Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Uma ferramenta para monitoramento do sistema JoiN de processamento maciçamente paralelo virtual

Autor: Ana Maria de Seixas Pereira
Orientador: Prof. Dr. Marco Aurélio Amaral Henriques

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Engenharia de Computação**.

Banca Examinadora

Edmundo Roberto Mauro Madeira, Prof. Dr. . . . . IC/UNICAMP
Marco Aurélio Amaral Henriques, Prof. Dr. . . . . DCA/FEEC/Unicamp
Maurício Ferreira Magalhães, Prof. Dr. . . . . DCA/FEEC/Unicamp

Campinas, SP
2008

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Pereira, Ana Maria de Seixas
P414f Uma ferramenta para monitoramento do sistema JoiN de processamento maciçamente paralelo virtual / Ana Maria de Seixas Pereira. – Campinas, SP: [s.n.], 2008.

Orientador: Marco Aurélio Amaral Henriques.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Computação em Grade (Sistema de computador). I Henriques, Marco Aurélio Amaral. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título

Título em Inglês:	A monitoring tool for the massively parallel virtual processing system JoiN
Palavras-chave em Inglês:	Grid application monitoring, Grid computing, JoiN
Área de concentração:	Engenharia de Computação
Titulação:	Mestre em Engenharia Elétrica
Banca Examinadora:	Marco Aurélio Amaral Henriques, Edmundo Roberto Mauro Madeira, Mauricio Ferreira Magalhães
Data da defesa:	27/08/2008
Programa de Pós Graduação:	Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Ana Maria de Seixas Pereira

Data da Defesa: 27 de agosto de 2008

Título da Tese: "Uma Ferramenta para Monitoramento do Sistema Join de Processamento Maciçamente Paralelo Virtual"

Prof. Dr. Marco Aurélio Amaral Henriques (Presidente): _____

Prof. Dr. Edmundo Roberto Mauro Madeira: _____

Prof. Dr. Maurício Ferreira Magalhães: _____

Resumo

O gerenciamento e a utilização de ambientes de computação em grade requerem um grande esforço na obtenção das informações necessárias para a administração e para a identificação e resolução de problemas. Acompanhar o desempenho das aplicações, determinar a origem de problemas, identificar e eliminar gargalos são funções que requerem informações detalhadas sobre a plataforma e sobre as aplicações sendo executadas. Neste trabalho apresentamos uma proposta para implementação de uma ferramenta para monitoramento do sistema JoiN de processamento maciçamente paralelo virtual. Após a análise de algumas das ferramentas hoje existentes para monitoramento de ambientes distribuídos não identificamos entre elas uma solução que satisfaça os requisitos do sistema JoiN e, por esta razão, implementamos uma nova ferramenta que procura integrar as melhores características encontradas nas ferramentas analisadas. As principais dificuldades encontradas na implementação desta ferramenta são relacionadas à obtenção e à publicação de um grande número de informações que permitam a observação e o monitoramento do sistema JoiN e que facilitem seu uso e sua administração, interferindo o mínimo possível com seu desempenho. Os resultados obtidos mostram que a ferramenta implementada no sistema JoiN oferece uma boa relação custo/benefício no monitoramento das principais funções deste sistema sem causar impactos significativos na execução de suas aplicações paralelas.

Palavras-chave: monitoramento em grades computacionais, computação em grade, JoiN.

Abstract

Grid Computing environments management and utilization require a great effort in gathering necessary information for site administration and identification/resolution of problems. Discovering applications performance, determining the origin of problems, identifying and eliminating bottlenecks are functions thar require detailed information about the platform and the applications running. In this work we present a proposal for implementation of a monitoring tool for JoiN system, a massively virtual environment for parallel processing. After the analysis of some of the available monitoring tools for distributed environments, we could not identify among them a solution that meets JoiN requirements and, therefore, we decided to implemented a new tool which seeks to incorporate the best features found on analyzed tools. The main difficulties found in the implementation of this tool are related to the collection and publication of a large number of information that allow JoiN observation and monitoring and that helps its administration and use, interfering as little as possible with its performance. The results show that the tool implemented in JoiN system offers a good cost/benefit relationship in monitoring the main functions of this system without causing significant impacts in the execution of parallel applications.

Keywords: grid application monitoring, grid computing, JoiN.

Agradecimentos

Pesquisa desenvolvida com o auxílio do CENAPAD-SP (Centro Nacional de Processamento de Alto Desempenho em São Paulo), projeto UNICAMP / MCT.

Ao meu orientador, Prof. Marco Aurélio Amaral Henriques, sou grata pela orientação, amizade, apoio e sobretudo pela paciência.

Aos colegas do grupo JoiN, em especial à Cristina, à Miriam, ao Fábio e ao Roberto, pela solidariedade, amizade, críticas e sugestões.

Aos amigos do CENAPAD-SP, pelo apoio e incentivo.

Aos muitos amigos, presentes nas alegrias e nas tristezas, que foram e são muito importantes na minha vida.

Aos que partiram antes que eu pudesse terminar este trabalho, companheiros de um longo caminho, mas que estão sempre presentes no meu coração.

À minha família pelo apoio e compreensão.

A Deus, que permitiu a realização de mais esta etapa na minha tão abençoada vida.

Ao Cesar, Mateus, João e aos meus pais,

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

O Apêndice E complementa este glossário com definições de alguns conceitos mais recorrentes no texto.

ANSI - American National Standards Institute

API - Application Programming Interface

ASP - Active Server Pages

AWT - Abstract Window Toolkit

BSP - Bulk Synchronous Parallel

BXXP - Blocks Extensible Exchange Protocol

DDL - Data Definition Language

DMF - Distributed Monitoring Framework

EDG - European DataGrid

EGEE - Enabling Grids for E-sciencE

FLOPS - FLoating point Operations Per Second

GGF - Global Grid Forum

GGF-Perf - GGF Performance Working Group

GIS - Grid Information Service

GMA - Grid Monitoring Architecture

GSTR - Generational Scheduling with Task Replication

HCI - Human-Computer Interaction

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

IDS - Intrusion Detection System

IHC - Interação Humano-Computador

IIS - Internet Information Services

IMAP - Internet Message Access Protocol

JAMM - Java Agents for Monitoring and Management

JDBC - Java Database Connectivity

JSP - Java Server Page

JVM - Java Virtual Machine

LB - Logging and Bookkeeping

LDAP - Lightweight Directory Access Protocol

LHC - Large Hadron Collider

MA-GMA - Mobile Agent-based Grid Monitor Architecture

MDS - Monitoring & Discovery System

MIME - Multipurpose Internet Mail Extensions

MonALISA - MONitoring Agents using a Large Integrated Services Architecture

MPI - Message Passing Interface

MRTG - Multi Router Traffic Grapher

NNTP - Network News Transfer Protocol

PAS - Parallel Application Specification

PASL - Parallel Application Specification Language

PDU - Protocol Data Unit

PHP - Hypertext Preprocessor

POP3 - Post Office Protocol 3

PRC - Public-Resource Computing

pyGMA - Grid Monitoring Architecture Web-Services Interfaces in Python

R-GMA - Relational Grid Monitoring Architecture

RMI - Remote Method Invocation

RRDtool - Round Robin Database Tool

SEQUEL - Structured English QUEry Language

SGBD - Sistemas Gerenciadores de Bancos de Dados

SGML - Standard Generalized Markup Language

SIG - Sistema de Informações Gerenciais

SMTP - Simple Mail Transfer Protocol

SNMP - Simple Network Monitoring Protocol

SOAP - Simple Object Access Protocol

SQL - Structured Query Language

TID - JoiN Task IDentification

ULM - Universal Logger Message

URL - Uniform Resource Locator

W3C - World Wide Web Consortium

WMS - EDG Workload Management System

WSDL - Web Services Description Language

XML - eXtensible Markup Language

Capítulo 1

Introdução

A computação de alto desempenho é necessária para pesquisas em diversas áreas, onde problemas computacionalmente complexos demandam processamento intensivo e a manipulação de grandes volumes de informações. Existem grandes centros de computação que oferecem recursos para a solução deste tipo de problema; entretanto, o acesso a estes centros pode ter um custo muito alto. Em muitos casos o alcance da pesquisa científica é limitado pela disponibilidade de poder computacional e de recursos instalados.

Algumas das aplicações que requerem processamento de alto desempenho podem utilizar sistemas paralelos, beneficiando-se da distribuição de dados e de processamento. Nestes sistemas paralelos o poder computacional total disponível é definido pelo número e capacidade dos processadores utilizados e pela velocidade de comunicação entre eles. Neste cenário surgem ambientes em que um grande número de computadores é utilizado para compor sistemas de processamento de alto desempenho.

Por trás de todos estes esforços está a necessidade crescente do uso de ambientes com grande poder computacional e a dificuldade de aquisição e manutenção de computadores deste porte por grupos de pesquisa. A utilização de forma colaborativa de ciclos de *cpu*, seja de computadores de grande porte, seja de computadores pessoais, é hoje em dia um grande desafio na área de processamento de alto desempenho.

Objeto de muitas pesquisas, a Computação em Grade (*Grid Computing*) [1, 2, 3] permite o compartilhamento de recursos geograficamente distantes e cria ambientes heterogêneos e dinâmicos de processamento distribuído em que o controle dos recursos e dos serviços não é feito de forma centralizada.

Ambientes de processamento baseados na Internet compõem um tipo de Computação em Grade e são também chamados de ambientes de Computação em Grade com Recursos Públicos (*PRC - Public-Resource Computing*) [4], em que a utilização do tempo ocioso de computadores quaisquer viabiliza a execução de aplicações paralelas que requerem ambientes de computação de alto desempenho, mas que podem se adequar a uma infra-estrutura fracamente acoplada. O sistema *JoiN* [5], em desenvolvimento na Faculdade de Engenharia Elétrica e de Computação da UNICAMP, permite a exploração de ciclos ociosos de computadores conectados à Internet, disponibilizando poder computacional para aplicações que possam tirar proveito de um grande paralelismo de dados.

A administração de um ambiente computacional é uma função de controle, que envolve a observação e reação a eventos ocorridos neste ambiente, e é responsável por manter a capacidade do ambiente de oferecer os serviços a que se propõe. Acompanhar o desempenho de aplicações, deter-

minar a origem de problemas, identificar e eliminar gargalos são algumas das muitas atividades que requerem obtenção de informações detalhadas.

A partir da necessidade de ferramentas que facilitem o monitoramento e a administração do sistema JoiN, este trabalho busca soluções para um problema complexo: a obtenção e publicação de informações, que permitam a observação e o monitoramento deste ambiente, e que também facilitem seu uso e sua administração, interferindo o mínimo possível com seu desempenho. As características do sistema JoiN, ambiente de processamento baseado na Internet, são: maciçamente paralelo, porque é composto por muitos computadores, geograficamente distribuídos; heterogêneo, porque agrega computadores de diferentes arquiteturas e sistemas operacionais, administrados de forma independente por seus proprietários; dinâmico, porque os computadores que o compõem podem conectar-se e desconectar-se a qualquer momento. Tais características tornam difícil seu monitoramento e gerenciamento e a implementação de uma ferramenta de monitoramento que auxilie a administração do sistema JoiN envolve o tratamento de um grande volume de informações, obtidas pela ocorrência freqüente de eventos nos muitos componentes de um grande ambiente distribuído. Assim, a ferramenta para monitoramento deste ambiente deve ser escalável, flexível, configurável e deve manipular informações produzidas por muitos computadores em um ambiente onde mudanças ocorrem freqüentemente.

Os objetivos deste trabalho serão discutidos na seção 1.2 que trata da definição, implementação e avaliação de uma ferramenta para monitoramento do sistema JoiN. Antes disto vamos apresentar em mais detalhes a computação em grade.

1.1 Cenário: Computação em Grade

A computação em grade tem se destacado na comunidade acadêmica e também na indústria, tanto no desenvolvimento de sistemas e ferramentas, quanto na sua utilização para solução de problemas. Muitas implementações de ambientes em grade têm sido feitas, resultando em sistemas com um grande poder computacional, para uso nas mais diversas áreas de conhecimento, como por exemplo: astronomia, química, engenharia civil, estudos climáticos, ecologia, geologia, medicina, física, geologia e ciência da computação. A Fig. 1.1 exemplifica uma grade, composta por computadores de vários tipos e pertencentes a diferentes organizações, geograficamente dispersos e conectados via Internet. A seguir são apresentados alguns destes sistemas, representando uma pequena parcela dos sistemas existentes.

A comunidade participante do *Globus Alliance* [6] trabalha na proposta de um conjunto de especificações técnicas e ferramentas de software para computação em grade, definindo padrões que são importantes para garantir a compatibilidade entre os computadores participantes da grade. A infraestrutura *Globus Toolkit* [7, 8, 9] em desenvolvimento por esta comunidade, é utilizada na implementação de ambientes de computação em grade, entre os quais podemos destacar: *AccessGrid* [10]; *Open Science Grid* [11]; *TeraGrid* [12] e *EGEE* [13] (uma relação mais completa pode ser consultada em *http://www.globus.org/alliance/projects.php*).

O *Globus Toolkit* é um software de domínio público, resultado de um projeto iniciado em 1997 pelas instituições Argonne National Labs, University of Chicago e University of Southern California, com o objetivo de oferecer software de código aberto para a construção de sistemas e aplicações em grade. Permite o compartilhamento de poder computacional, bases de dados e outras ferramen-

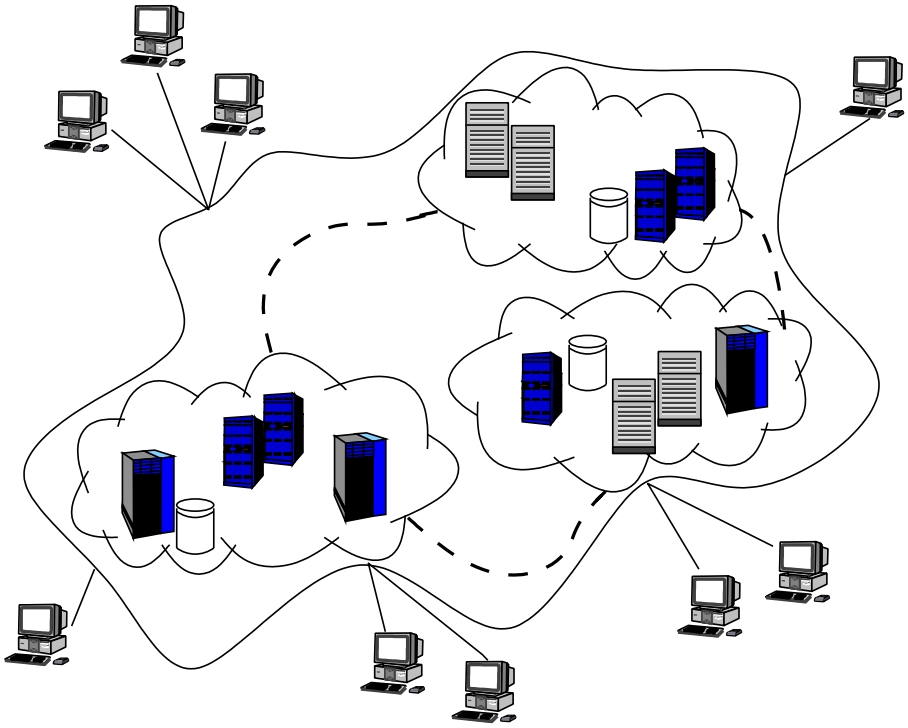

Fig. 1.1: Possível cenário de computação em grade

tas, mesmos quando estes recursos estão geograficamente distribuídos, e são administratados individualmente por seus proprietários. Este software inclui serviços e bibliotecas para monitoramento, descoberta e gerenciamento de recursos, além de segurança, gerenciamento de dados, comunicação, detecção de falhas e portabilidade. Seus componentes podem ser utilizados, no desenvolvimento de aplicações, de forma independente ou juntos. Pode-se destacar como principal caraterística deste software a possibilidade de seus usuários acessarem recursos remotos como se estivessem localizados em sua própria máquina: as diferentes formas de operação, administração e colaboração existentes entre múltiplas organizações, além da incompatibilidade de computadores, sistemas operacionais, sistemas de armazenamento de dados e conexões de rede, são observadas pelos diversos componentes do *Globus Toolkit*, que removem os obstáculos existentes, facilitando a interação e o compartilhamento de recursos. Isto é feito respeitando-se o controle exercido por cada organização sobre seus recursos, determinando quem pode utilizar estes recursos, e quando. O *Globus Toolkit* tem sido desenvolvido e distribuído como software livre e, com estratégia similar à do sistema operacional Linux, estimula e amplia sua utilização com a participação de muitos membros da comunidade de pesquisa em computação em grade. Isto também propicia uma maior inovação técnica e melhorias contínuas ao produto. O monitoramento em ambientes que utilizam o *Globus Toolkit* é feito por meio de consultas a informações mantidas pelo serviço MDS (*Monitoring & Discovery System*) [14]. Existe um servidor MDS que mantém informações sobre os recursos do ambiente, e um módulo MDS em cada nó da grade para manutenção de informações relativas ao nó.

O *Globus Toolkit* é o software mais utilizado na implementação de ambientes de grade computacional, mas existem também outras infraestruturas para computação em grade, não baseadas no *Globus Toolkit*:

- o projeto *Sardes (System Architectur for Reflective Distributed Computing Environments)* [15, 16], do *Institut National de Recherche en Informatique et en Automatique (Inria Rhône-Alpes)*, que tem por objetivo desenvolver conceitos, metódos e ferramentas para a construção de sistemas distribuídos, incluindo infraestrutura e aplicações. Entre os principais desafios identificados neste projeto estão a escalabilidade, a administração e a qualidade de serviço. Para atender estes desafios, o projeto Sardes propõe que tanto a infraestrutura quanto as aplicações sejam adaptáveis, ou seja, tenham arquitetura modular, onde as interfaces e dependências entre as diversas partes sejam claramente identificadas e os componentes sejam reconfiguráveis e adaptáveis a alterações no ambiente, em especial com relação a sistemas operacionais. Para isto são utilizados programação reflexiva e componentização. O monitoramento no projeto Sardes é feito por meio de ferramanenta baseada na infraestrutura Dream [17];
- o projeto *Unicore (UNiform Interface to COmputing REsources)* [18], do governo alemão, tem por objetivo oferecer uma *interface* gráfica para facilitar o acesso uniforme a recursos distribuídos e heterogêneos. Sua arquitetura consiste em três camadas: usuário, servidor e sistema alvo. Por meio do cliente *Unicore* o usuário tem acesso a uma interface gráfica que permite a exploração dos serviços oferecidos pelo servidor e facilita a criação e submissão de trabalhos complexos e interdependentes. Os serviços disponíveis, ou sistemas alvo, são constituídos por um ou mais *Vsite (UNICORE Virtual site)* que, por sua vez, são computadores de diversas arquiteturas e pertencentes a diferentes organizações. O Unicore é base de muitos projetos de pesquisas e também é utilizado para o acesso a diversos supercomputadores em todo o mundo, como por exemplo: *EUROGRID* [19]; *OpenMolGRID* [20]; *VIOLA* [21]; e o japonês *NaReGI* [22]. SIMON (*Site Functionality Monitoring Tool*) [23] é a ferramenta utilizada para monitoramento no sistema UNICORE.

Para utilização de computadores ociosos na Internet podemos destacar os seguintes projetos, entre as diversas implementações e *frameworks* existentes.

- *Seti@Home* [24], um dos mais conhecidos e difundidos. O projeto SETI (*Search for Extra-Terrestrial Intelligence*) tem por objetivo analisar os sinais de rádio captados pelo radiotelescópio instalado em Arecibo (Porto Rico), na busca por alguma forma de vida inteligente no universo. Para analisar o grande volume de informações captadas em Arecibo, foi desenvolvido o projeto *Seti@Home*, que consiste na divisão destes dados em pequenos lotes, que possam ser analisados pelos computadores pessoais. Desta forma o projeto conta com a participação voluntária de pessoas que possuem computadores conectados à Internet e que permitem o processamento em seus computadores para a análise dos sinais captados em Arecibo. O voluntário cadastrado no *Seti@Home*, ao conectar-se à Internet, carrega um lote de informações coletadas pelo radiotelescópio em seu computador. Estas informações são analisadas durante o tempo em que o computador estiver ocioso e, após essa análise, os resultados são transmitidos ao controle do projeto. A popularidade deste projeto pode ser medida pelo grande número de voluntários cadastrados, que em conjunto compõem a maior capacidade de processamento que já existiu: são 170 mil voluntários permanentes e 320 mil computadores cadastrados. O projeto *Seti@Home* [25] teve início em 1999 e ainda não foram encontradas provas da existência de vida extraterrestre. Este projeto foi desenvolvido para resolver apenas este problema e, durante o ano de 2004 o projeto migrou para a plataforma *Boinc*, apresentada a seguir. Foram desenvolvidos diversos *add-ons* para monitoramento do projeto *Seti@Home*,

entre os quais podem ser destacados Msetimon (*Multi SETI@home Monitor*) [26] e *SETI Stats Analyser* [27], além da página *web* onde pode ser acompanhada a utilização do sistema *http://setiathome.berkeley.edu/stats.php*;

- *Boinc (Berkeley Open Infrastructure for Network Computing)* [28], versão de propósito geral do *Seti@Home*. Devido ao sucesso alcançado na distribuição de processamento com o projeto *Seti@home*, e com os conhecimentos adquiridos no desenvolvimento deste projeto, a Universidade de Berkeley desenvolveu uma plataforma para permitir a utilização da tecnologia disponível para outras áreas de pesquisa além do projeto *SETI*: nasceu assim o *Boinc*. O objetivo desta plataforma é conseguir mais voluntários, propondo a cada um deles a colaboração na cura de doenças, no estudo do aquecimento global, no descobrimento de pulsares e em outras pesquisas científicas; os projetos que utilizam esta plataforma são listados na página *http://boinc.berkeley.edu/projects.php* e abrangem as áreas de matemática, estratégia de jogos, astronomia, física, química, predição climática, medicina e biologia. Os pesquisadores e cientistas são convidados a desenvolver novos projetos que utilizem a plataforma *Boinc*, para distribuir processamento e conseguir grande poder computacional. Para universidades e empresas a plataforma oferece a possibilidade de utilização dos computadores, para criação de *Virtual Campus Supercomputing Center* ou então de *Desktop Grid Computing*. Qualquer aplicação cujos dados possam ser divididos em um grande número de lotes, sobre os quais o processamento possa ser feito de forma independente, pode utilizar a plataforma *Boinc*. As principais características desta plataforma são: a autonomia de cada projeto, que pode ser executado de forma independente dos demais; a flexibilidade oferecida a cada voluntário, que pode participar de um ou mais projetos, à sua escolha; a possibilidade de executar aplicações já existentes, em C, C++ ou Fortran, com nenhuma ou poucas modificações; a segurança, para voluntários e aplicações, por exemplo com o uso de codificação com chave pública; o desempenho eficiente e a escalabilidade; a distribuição sob a licença *Lesser General Public License*; o suporte a grandes volumes de dados; a possibilidade de participação de computadores de diversas platformas (Mac OS X, Windows, Linux and other Unix systems); a arquitetura aberta, e a documentação das interfaces dos principais componentes, possibilitando o desenvolvimento de extensões. Entre estas extensões existem algumas que foram desenvolvidas para monitoramento dos projetos implementados no sistema *Boinc*, entre as quais podem ser destacadas *Boinctray* [29] e *BOINCprog* [30], além da página *web* onde pode ser acompanhada a utilização do sistema *http://boincstats.com/*;

- *Condor* [31], um dos pioneiros na utilização de recursos ociosos. O projeto *Condor* teve início em 1984 com o objetivo de explorar o poder computacional existente nas instituições, e que se encontrava espalhado em muitos computadores de pequeno porte, devido à utilização crescente de computadores pessoais, buscando atender as necessidades de processamento em ambientes onde os recursos são heterogêneos e distribuídos. O sistema *Condor* gerencia a distribuição de carga para a execução de aplicações e, nos moldes da maioria dos sistemas para execução em *batch*, possui mecanismos para enfileiramento, escalonamento, atribuição de prioridades, monitoramento e gerenciamento de recursos. Os usuários submetem suas aplicações ao sistema *Condor*, que as enfileira e então determina quando e onde as mesmas serão executadas. Para isto procura encontrar, entre os recursos disponíveis, aqueles necessários para a execução de cada aplicação. O *Condor* monitora as execuções e envia notificações aos usuários quando as mes-

mas são completadas. O conjunto de computadores onde as aplicações podem ser executadas é chamado de *Condor pool*. Neste conjunto co-existem os três componentes do sistema: agentes, recursos e *mathcmaker*. Os agentes estabelecem a relação entre recursos e as necessidades para execução das aplicações; nos recursos são definidos quais são os usuários que podem acessá-los; e o *matchmaker* é responsável pela implementação das políticas previamente determinadas. O projeto *Condor* é anterior à idéia de grades computacionais, mas existe uma implementação do escalonador *Condor* que usa recursos de um *Condor pool* associado a recursos acessíveis via *Globus*: *Condor-G* [32], que implementa um *front-end* para grades computacionais por meio de interação com os protocolos *Globus*. Para monitoramento de ambientes baseados no Condor é utilizada a ferramenta *Hawkeye* [33] que coleta, armazena e disponibiliza informações sobre os computadores conectados a estes ambientes;

- *Legion* [34], um conjunto de interfaces para constituição de um ambiente distribuído orientado a objetos. Projeto iniciado em 1993 na Universidade da Virginia, que apresenta como principais características: autonomia para diferentes domínios, ou seja, podem ser criados diversos agrupamentos independentes de computadores, onde cada grupo pode ser utilizado para execução de aplicações específicas; núcleo extensível e arquitetura escalável; ambiente de computação homogêneo, isto é, os computadores componentes de um agrupamento são apresentados para seus usuários como um grupo homogêneo, mesmo quando forem de arquiteturas distintas ou utilizarem sistemas operacionais diferentes; suporte a múltiplas linguagens; tolerância a falhas; segurança baseada em direitos e privilégios atribuídos aos usuários; execução sem a necessidade de privilégios de administrador do sistema operacional. A arquitetura do sistema *Legion* é orientada a objetos, que se comunicam por meio de chamadas assíncronas de métodos, utilizando interfaces e protocolos próprios. Alguns objetos centrais implementam funcionalidades básicas como: hospedeiro, que representa um processador na grade; cofre, que representa um objeto *Legion* qualquer serializado em disco; implementação, que representa um programa a ser executado. Possui suporte a aplicações paralelas que utilizam troca de mensagens por meio das bibliotecas MPI e PVM. Estas aplicações podem ser programadas nas linguagens MPL (extensão de C++), Fortran e Java; as aplicações já existentes podem ser encapsuladas em objetos. O sistema Legion implementa um mecanismo de notificação baseado em eventos e o monitoramento é feito pela ferramenta *Resource management* [35].
- *Ourgrid* [36, 37], uma ferramenta para ambientes de computação em grade, desenvolvida na Universidade Federal de Campina Grande, que tem como objetivo facilitar a utilização destes ambientes heterogêneos e dinâmicos no processamento de aplicações do tipo "*bag-of-tasks*", compostas por tarefas independentes que não se comunicam. A ferramenta OurGrid Toolkit foi desenvolvida em Java, implementa a comunicação entre processos por meio da invocação de métodos remotos *(RMI - Remote Method Invocation)* e é composta por três peças: a comunidade OurGrid, que permite o compartilhamento de recursos com base na tecnologia "*peer-to-peer*"; a ferrramenta MyGrid, que possibilita ao usuário trabalhar com abstrações de alto nível na utilização da grade; a plataforma Swan, que permite o processamento em máquinas virtuais isoladas do sistema operacional. Os usuários podem fazer o monitoramento da execução de suas aplicações por meio de scripts e programas distribuídos com a ferramenta OurGrid. É um software gratuito e de código aberto, que pode ser baixado em *http://www.ourgrid.org*.
- *Integrade* [38, 39], uma infra-estrutura que permite a utilização não dedicada de computadores.

Aproveita recursos ociosos destes computadores, sem comprometer sua utilização por seus proprietários, com o objetivo de viabilizar a execução tanto de aplicações do tipo "*bag-of-tasks*" como aplicações paralelas nos modelos BSP (*Bulk Synchronous Parallel*) e MPI (*Message Passing Interface*) que requeiram comunicação entre tarefas. Implementa um mecanismo de tolerância a falhas, por meio de *checkpoints* incorporados ao código das aplicações, que tem por objetivo garantir a execução das mesmas uma vez que os computadores utilizados alternam constantemente de estado entre ocioso e ocupado. Uma iniciativa de diversas universidades (USP, PUC-Rio, UFMS, UFG e UFMA), Integrade é um software aberto, multi-plataforma e orientado a objetos, desenvolvido em CORBA. Implementa ferramentas gráficas para o monitoramento da utilização de recursos e da execução de aplicações em grade. O ambiente de grade computacional de Integrade é estruturado como um conjunto hierárquico de *clusters*, onde cada *cluster* é composto por grupos de até cem computadores, podendo potencialmente agregar milhões de máquinas. Também é um software gratuito e de código aberto, que pode ser baixado em *http://www.integrade.org.br/portal/software*.

Além destes projetos e infraestruturas citados para implementação de ambientes de grade computacional e para utilização de computadores ociosos na Internet, existem esforços que buscam a integração entre ambientes de computação em grade e de processamento baseados na Internet [40]. O objetivo desta integração é criar um ambiente híbrido, onde grades computacionais, normalmente compostas por computadores de grande porte, possam interagir com ambientes compostos por computadores pessoais e, desta forma, utilizar o poder de um grande número de computadores agregados.

1.2 Motivação e Objetivos

Entre os fatores que distinguem um ambiente de computação de alto desempenho estão o poder computacional disponibilizado para os usuários, a facilidade de uso e a existência de ferramentas para gerenciamento, administração e observação da execução de aplicações. Neste contexto, a existência de uma ferramenta de monitoramento eficiente e flexível é essencial para que o ambiente possa ser utilizado da melhor forma possível.

As ferramentas para monitoramento de ambientes de processamento baseados na Internet e de computação em grade devem possibilitar o acompanhamento da utilização e do desempenho de cada equipamento integrante do ambiente, por meio da obtenção e apresentação de informações relevantes. Além disto, devem permitir o acesso a informações relativas ao estado corrente e a estados passados, de forma a permitir análises estatísticas sobre comportamento do ambiente e de seus componentes no decorrer do tempo. Desta forma, estas ferramentas são também muito importantes para a administração dos computadores participantes, pois estes foram originalmente projetados para uso e administração como sistemas individuais. Além disto, o monitoramento deve ser um facilitador para a descoberta de possíveis problemas e gargalos, coletando periodicamente informações sobre o estado dos componentes do ambiente e sobre parâmetros de seu desempenho.

O sistema *JoiN* [5] para computação em grade, em desenvolvimento na Faculdade de Engenharia Elétrica e de Computação da UNICAMP, tem por objetivo explorar ciclos ociosos de computadores conectados à Internet, disponibilizando poder computacional para aplicações fracamente acopladas que possam tirar proveito de um grande paralelismo de dados. Com o uso do sistema *JoiN* é possível criar e utilizar ambientes de grade computacionais com recursos públicos, e com isto, viabilizar

a execução de aplicações maciçamente paralelas. Desenvolvido em linguagem Java, com estrutura baseada em componentes, o sistema *JoiN* tem como característica principal a grande portabilidade, pois pode ser executado em qualquer computador em que houver uma implementação da Máquina Virtual Java (JVM). A escalabilidade, essencial para ambientes de grade, é obtida pela organização hierárquica dos computadores em grupos. Além disto, o *JoiN* implementa eficientemente funções importantes para este tipo de sistemas, tais como: escalonamento, replicação, tolerância a falhas e segurança. A implementação do sistema JoiN é baseada no conceito de serviços que podem ser combinados de diversas formas para compor ambientes bastante flexíveis. O sistem JoiN é apresentado no capítulo 3.

O objetivo deste trabalho é implementar e avaliar uma ferramenta de monitoramento para o ambiente JoiN de computação em grade. Para isto, vamos identificar os principais desafios encontrados no projeto e implementação deste tipo de ferramenta e, por meio da análise de algumas das soluções existentes, destacar as funcionalidades e características desejadas na ferramenta a ser implementada. Os principais desafios estão relacionados à obtenção, ao tratamento e à publicação do grande volume de informações, produzidas pelo extenso e diversificado grupo de integrantes da grade. Assim, nas soluções existentes, a eficiência das ferramentas de monitoramento passa a ser fundamental: tais soluções devem preocupar-se com a qualidade, o conteúdo e a atualidade das informações, obtidas a partir do tratamento de um grande volume de dados, de forma a satisfazer as necessidades dos administradores e usuários do ambiente.

A estratégia adotada para atingirmos estes objetivos compreende diversas etapas, que podemos resumir em:

- levantamento dos fundamentos teóricos e das barreiras tecnológicas existentes para a implementação deste tipo de ferramenta;
- avaliação de ferramentas de monitoramento já existentes para ambientes de computação em grade, identificando as abordagens adotadas na sua implementação e suas principais características;
- definição e implementação de uma ferramenta para monitoramento de um ambiente de computação em grade que seja eficiente, adequada às necessidades do sistema JoiN, e que facilite a administração, a avaliação e a otimização deste sistema;
- avaliação da solução adotada e dos resultados obtidos com a ferramenta implementada.

1.3 Organização do trabalho

O restante deste trabalho está organizado da seguinte maneira. No Capítulo 2 são apresentadas algumas definições, introduzidos alguns fundamentos teóricos desta área de pesquisa e discutidos os principais desafios, dificuldades e barreiras tecnológicas a serem vencidos no projeto e implementação de ferramentas para monitoramento de ambientes de computação em grade. São também apresentados alguns trabalhos relacionados a este estudo e, entre estes, destaca-se a GMA (*Grid Monitoring Architecture*), proposta para padronizar ferramentas de monitoramento em *Grid Computing*.

No Capítulo 3 é apresentado o sistema JoiN de processamento paralelo baseado na Internet: sua arquitetura e implementação, além de aspectos importantes como segurança, portabilidade, e escalabilidade. São também discutidas as características desejadas para uma ferramenta de monitoramento

para o sistema JoiN e definidos os critérios que devem ser considerados na avaliação desta ferramenta. É feita a análise de algumas das ferramentas disponíveis, avaliando a possibilidade de sua utilização no ambiente JoiN. A seguir são apresentados o projeto e a implementação da ferramenta JoiNMon para monitoramento do sistema JoiN.

No Capítulo 4 é apresentada a análise do impacto da ferramenta de monitoramento no sistema JoiN, tanto para os usuários, que executam suas aplicações neste ambiente, como para os colaboradores, que disponibilizam seus computadores para o sistema.

No Capítulo 5 são apresentadas discussões sobre possíveis trabalhos futuros.

Os Apêndices A, B, C, D e E contêm informações adicionais: no primeiro são apresentadas algumas ferramentas existentes para monitoramento de ambientes de computação em grade; no segundo é documentada a base de dados utilizada pela ferramenta de monitoramento para armazenar as informações coletadas; no terceiro são apresentadas as funcionalidades da interface gráfica, desenvolvida para facilitar a obtenção das informações coletadas e armazenadas pela ferramenta de monitoramento; no quarto são apresentados todos os resultados obtidos nos experimentos realizados para análise da ferramenta de monitoramento; no quinto são apresentados alguns conceitos e definições citados neste trabalho que, por não serem objeto primário deste estudo, não são explorados em detalhes no texto principal.

Capítulo 2

Aspectos tecnológicos e trabalhos relacionados

Neste capítulo são discutidos os desafios encontrados na implementação de uma ferramenta de monitoramento para ambientes distribuídos, heterogêneos e dinânicos, e também os fundamentos teóricos que devem ser observados nesta implementação. São apresentadas as principais barreiras tecnológicas que devem ser superadas e satisfeitas, e também os conceitos e terminologia utilizados na apresentação das ferramentas de monitoramento estudadas.

É destacada a padronização GMA (*Grid Monitoring Architecture*), proposta pelo grupo de trabalho *Performance Working Group* do GGF (*Global Grid Forum*), que tem o objetivo de definir padrões e interoperabilidade entre os softwares desenvolvidos para monitoramento em *Grid Computing*.

2.1 Barreiras tecnológicas

Funções de administração, como monitoramento, gerenciamento de recursos e tolerância a falhas, são baseadas em informações, obtidas e propagadas por componentes do sistema. Em ambientes heterogêneos e dinâmicos, muitas vezes conectados por meio de redes instáveis, são comuns as falhas e freqüentes as mudanças de configurações da plataforma [41]. A administração deste tipo de ambiente requer ferramentas adequadas e informações atualizadas.

Em grandes sistemas distribuídos e baseados na Internet, estas informações podem ser geradas em taxas muito elevadas. Assim, são necessários canais eficientes para recolher e filtrar estes acontecimentos, e para propagá-los aos seus destinatários. A coleta e o tratamento de informações são críticos pois, neste ambiente com muitos e diversificados colaboradores e aplicações, grande parte das informações coletadas tem vida útil muito curta, ou seja, refletem um estado atual e efêmero. As ferramentas de monitoramento devem ser eficientes, estar constantemente ativas e ser tolerantes a falhas.

Algumas condições devem ser observadas no desenvolvimento de ferramentas de monitoramento para que seus objetivos sejam atingidos. Entre estas condições, as listadas a seguir são essenciais para seu sucesso e, como descrito por Röder, Ludwig e Bode [42], devem estar presentes em toda ferramenta de monitoramento:

- escalabilidade: não deve introduzir gargalos no sistema;

- não intrusão: não deve consumir de forma excessiva os recursos disponíveis, evitando que as aplicações sofram alterações em seu desempenho;
- baixo tráfego de rede: o uso da rede deve ser reduzido para evitar a disputa por este recurso. Isto pode ser feito por meio de uma taxa de atualizações de informações não elevada e de um baixo volume de dados transmitidos (as informações a serem monitoradas devem ser selecionadas e pré-processadas localmente na origem sempre que possível);
- interface gráfica: deve prover facilidade de uso e clareza na exibição de informações, de forma a facilitar a análise das mesmas;
- disponibilidade das informações: as informações devem ser disponibilizadas para análise em tempo de execução; também deve ser possível o arquivamento destas para permitir sua análise após o término da execução dos aplicativos, o que é chamado de análise *post-mortem*.

2.2 Fundamentos teóricos

Entre as possíveis abordagens para implementações de monitoramento, as técnicas passiva (ou indireta) e ativa (ou direta) são as mais freqüentemente utilizadas. Cada uma delas apresenta características próprias, vantagens e desvantagens. Como descrito por Zangrilli e Lowekamp [43], estas técnicas podem ainda ser combinadas e utilizadas de forma complementar, com o objetivo de reduzir o impacto e o custo causado pelo monitoramento ativo sem que seja sacrificada a acuidade das informações que podem ser obtidas desta forma.

2.2.1 Monitoramento indireto ou passivo

O monitoramento pode ser feito de forma indireta, ou passiva, [44], por meio da análise do conteúdo das informações que trafegam normalmente pelo sistema ou que são registradas em *logs*. Por exemplo, no monitoramento passivo de redes [45], podem ser introduzidos dispositivos para análise do tráfego que passa por eles, como *sniffers*, ou então o monitoramento pode ser implementado em dispositivos já existentes na rede, como roteadores, switches ou os próprios computadores conectados. Esta abordagem tem como vantagens a independência e a transparência, pois os componentes não precisam ter implementações específicas para gerar informações de monitoramento, além de não gerar tráfego adicional, pois a abordagem é baseada em informações já existentes. As desvantagens são: a possibilidade da existência de mudanças de estados que não estejam associadas ao tráfego de informações ou ao registro de *logs* e a necessidade de análise de todo o tráfego de informações e dos *logs*, o que pode comprometer a eficiência e o desempenho do sistema de processamento como um todo. Por exemplo, no caso de monitoramento de ambientes de computação em grade, se o término de uma aplicação não gerar tráfego de informações ou não for registrado nos *logs* do sistema, não será possível monitorar este evento.

Outro aspecto importante com relação ao monitoramento passivo é necessidade de maior segurança ou privacidade das informações, uma vez que a análise abrangerá todo o tráfego ou *log*. Isto requer uma maior proteção ao acesso das informações obtidas e armazenadas.

2.2.2 Monitoramento direto ou ativo

O monitoramento ativo requer a geração explícita das informações a serem observadas. Desta forma introduz tráfego extra, o que pode comprometer o desempenho do sistema. Por outro lado, é possível o controle da freqüência e do volume de geração de informações, bem como a seleção do tipo e conteúdo do monitoramento a ser feito. Por ser direto, pode-se obter apenas as informações desejadas, quando necessário, com maior rapidez e eficiência. Por exemplo, são deste tipo os sistemas para detecção de invasões (*Intrusion Detection System* - IDS) que têm como objetivos principais detectar se existe alguma tentativa de invasão no sistema ou se algum usuário está fazendo mau uso do mesmo. Estas ferramentas permanecem em execução no sistema a ser monitorado e somente geram notificações quando detectam alguma atividade considerada suspeita ou ilegal.

A seguir apresentamos as principais formas de implementação deste tipo de monitoramento.

Modelo produtor/consumidor

Uma forma de fazer o monitoramento direto utiliza o modelo produtor/consumidor [46]: os componentes do sistema (sensores) geram explicitamente as informações necessárias e as enviam a um outro componente (produtor) que as coleta, trata e disponibiliza de forma a permitir o monitoramento das informações relevantes por outros componentes (consumidores).

Esta opção tem como desvantagem a necessidade de gerar e formatar adequadamente as informações desejadas em cada componente sensor e produtor. Entre as vantagens destaca-se a possibilidade de obtenção de informações de forma mais completa, que podem ser previamente selecionadas e processadas, além de não haver a necessidade de análise de todas informações trafegadas.

A transmissão das informações pode ser feita por meio de *pooling* ou por meio de eventos. Na transmissão por *pooling* os consumidores solicitam informações aos produtores (*pull*) em intervalos regulares de tempo. Na transmissão por meio de eventos são os produtores que enviam as informações aos consumidores (*push*). O método baseado em *pooling* não é apropriado para o monitoramento em ambientes muito grandes pois pode afetar o desempenho da rede em função de um grande volume de tráfego.

Em geral, ferramentas de monitoramento que implementam o modelo produtor/consumidor utilizam eventos para a transmissão das informações a serem monitoradas. Podemos definir evento como um acontecimento, em algum tempo e lugar não determinados (ocorrência assíncrona). Além disto, um evento pode alterar o comportamento de, ou o relacionamento entre, indivíduos e objetos.

Em computação, utilizamos eventos para transmitir informações que denotam o resultado de uma ocorrência no sistema. Utiliza-se a implementação de *Event Listeners* para a captura e distribuição de informações associadas a eventos: criam-se listas de eventos e, para cada um deles, uma relação de objetos que devem ser notificados da ocorrência do mesmo. Assim, as informações são transmitidas assincronamente: quando um produtor tiver informações a serem enviadas, produz um evento que, devidamente tratado pelo *Event Listener*, é transmitido a todos os consumidores que tiverem previamente registrado seu interesse neste evento. Mesmo que os programadores de sistemas estejam habituados à implementação de *Event Listeners* [47], sua utilização em ferramentas de monitoramento requer alguns cuidados, importantes para a manutenção de suas características essenciais, pois, se o tratamento de eventos não for eficiente, podem ocorrer atrasos e perturbações indesejáveis no ambiente.

Sistemas e infraestruturas baseadas em eventos são, há muito tempo, estudados e desenvolvidos em diversos ramos da computação. Apesar disto, a sua utilização em ambientes distribuídos introduz desafios inerentes ao grande volume de informações a serem tratadas e ao dinamismo destes ambientes. Como apresentado em [48], na implementação de infraestruturas para tratamento de eventos em ambientes distribuídos deve-se dar especial atenção a dois principais desafios: manter a escalabilidade do ambiente e possibilitar o tratamento adequado de diversos tipos de informação.

2.2.3 Combinação das abordagens direta e indireta

Utilizando a combinação das duas formas de abordagem é possível explorar as vantagens de ambas, e construir uma ferramenta mais eficiente. As informações obtidas de forma direta, ou ativa, podem ser complementadas com informações obtidas de forma indireta. Quando uma informação for obtida de forma direta, pode-se disparar um mecanismo que obtenha informações complementares, relativas à primeira, de forma indireta ou passiva. Desta forma o monitoramento passivo contribui com novas informações, já disponíveis no ambiente, sem contudo coletar dados desnecessários e onerar o sistema. As informações, além de complementares, podem ser utilizadas para co-validação dos dados obtidos.

2.3 Conceitos e terminologia

Neste trabalho são usados conceitos sobre os quais temos definições informais, derivadas de diversas áreas e mesmo do nosso cotidiano, tais como produtores, consumidores e eventos. Estes conceitos são definidos a seguir, para que não sejam causa de ambigüidade ou interpretações equivocadas.

- Eventos, como descrito na sub-seção 2.2.2, são acontecimentos assíncronos, utilizados para transmitir informações que denotam o resultado de uma ocorrência no sistema.
- Origem é o local onde os eventos são gerados, e pode ser subdividida em:
  - sensor é o coletor das informações no local onde são geradas;
  - produtor é um processo que disponibiliza, por meio de eventos, as informações coletadas pelos sensores.

  As informações são obtidas pelos sensores e, a seguir, são formatadas, padronizadas e disponibilizadas pelos produtores.
- Destino (ou consumidor) é o local onde os eventos são consumidos, ou seja, consumidor é um processo que recebe dados gerados pelos produtores.
- Usuários (também chamados observadores) são pessoas ou processos que examinam as informações coletadas e exibidas por estas ferramentas. Esta observação pode ser feita em tempo real ou por meio da análise de informações previamente armazenadas.
- Serviço de diretório é o responsável pelo registro e busca de informações disponíveis.
- Dados são as informações contidas em um evento, permitindo que o consumidor tenha uma reação baseada no contexto em que o evento ocorreu.
- Interações são trocas de informações entre produtores e consumidores, que podem ser classificadas em:

- Registro: é um mecanismo usado por um produtor para registrar a existência de um determinado evento;
- Assinatura: é um mecanismo usado por um consumidor para registrar seu interesse na ocorrência de um determinado evento;
- Notificação: é o envio de eventos da origem (produtores) ao destino (consumidores).

2.3.1 Modelos de eventos

Meier e Cahill [49] apresentam uma classificação de modelos de eventos onde são considerados o sistema, o serviço, o modelo de eventos e a relação entre estes. Cada sistema de eventos tem um serviço e um modelo de eventos, e estes termos são definidos como:

- sistema de eventos: é uma aplicação que usa o serviço de eventos para a comunicação de eventos;
- modelo de eventos: consiste em regras que descrevem a forma de tratamento de eventos;
- serviço de eventos: é a infraestrutura que implementa um modelo de eventos e provê comunicação dos mesmos para o sistema de eventos.

O modelo de eventos define a interface entre o serviço de eventos e a aplicação que o utiliza, e pode ser classificado como *peer to peer*, mediador ou implícito.

Um modelo *peer to peer* permite que consumidores assinem diretamente informações de um produtor e este as envie também diretamente aos consumidores: este é o modelo implementado pelo *Java distributed event model*.

O modelo de eventos que faz uso de mediadores permite que consumidores assinem informações em um mediador e que os produtores enviem as informações (eventos) ao mesmo mediador, e este faz o encaminhamento aos consumidores: este é o modelo implementado pelo *event channel* do CORBA. O uso de mediadores tem como objetivo fazer a comunicação entre os consumidores e os produtores para que os produtores não precisem ter conhecimento dos seus consumidores e vice-versa. Desta forma não é necessário que o produtor gere o mesmo evento para todos os consumidores, bastando gerar um evento que será replicado e distribuído para seus consumidores pelo mediador.

No modelo implícito os consumidores fazem assinatura de um tipo de evento, sem uso de mediadores e sem a identificação do seu produtor, e são notificados diretamente pelo produtor quando este evento ocorre. Este mecanismo requer um serviço de eventos mais sofisticado, que deve manter as informações necessárias para associação entre produtores, eventos e consumidores.

2.4 Grid Monitoring Architecture

O GGF (*Global Grid Forum* - www.ggf.org) reúne a comunidade de usuários, desenvolvedores de software e fabricantes de computadores, para a definição de padrões e formas de interoperabilidade entre os softwares desenvolvidos para *Grid Computing*.

Em 1999 foi criado neste forum o grupo de trabalho GGF-Perf (*Performance Working Group*) com o objetivo de estabelecer padrões, para protocolos e representações, relativos ao armazenamento e distribuição de dados de desempenho. Entre estes padrões, há aqueles para o monitoramento do

estado de computadores e redes, e também do desempenho dos serviços oferecidos por uma grade computacional.

A arquitetura GMA [50] foi proposta pelo GGF-Perf como padrão para o monitoramento de recursos e de aplicações em ambientes de computação em grade.

Na proposta GMA são descritos alguns dos principais requisitos e características, destacando pontos importantes que devem ser considerados durante o projeto de uma ferramenta de monitoramento.

2.4.1 Características

O volume, o tempo de vida dos dados e a freqüência com que são gerados e acessados definem as principais características do monitoramento:

- em um ambiente de computação em grade, envolvendo um grade número de computadores, redes e usuários, o volume de dados coletados pode ser muito grande;
- o tempo de vida de uma informação, entretanto, é relativamente curto, pois sua importância é restrita a curtos intervalos de tempo;
- uma informação deve ser retida, ou armazenada, por mais tempo quando for necessária para contabilização da utilização de recursos, ou para o uso por outras ferramentas que executam análises de processos já encerrados;
- a freqüência com que são executadas as coletas e atualizações das informações é muito maior que a freqüência da leitura ou análise das mesmas.

2.4.2 Requisitos

Os principais requisitos a serem observados por uma ferramenta de monitoramento, adequada ao ambiente de computação em grade, são:

- baixa latência - devido ao curto tempo de vida de uma informação, a sua transmissão da origem (onde é gerada) para o destino (onde é armazenada/tratada) deve ter baixa latência;
- capacidade para tratamento de um grande volume de dados gerados;
- *overhead* mínimo na obtenção dos dados, para não ser um fator de degradação de desempenho;
- as informações obtidas devem ser transmitidas, armazenadas e tratadas com a segurança necessária para garantir apenas acessos autorizados ;
- a escalabilidade é muito importante, pois o monitoramento deve acompanhar possíveis aumentos de magnitude do ambiente.

2.4.3 Arquitetura

Na arquitetura GMA, a geração e o tratamento de eventos são feitos no esquema Produtor/Consumidor, utilizando eventos para transmissão de informações. São propostos três componentes, como pode ser visto na Fig. 2.1:

- Produtor: obtém as informações a serem monitoradas, produzidas pelos sensores, formata, padroniza e gera eventos;

- Consumidor: recebe notificações sobre eventos, gerados pelos produtores, e reage a estes eventos;
- Registro ou Serviço de Diretório: responsável pelo registro e busca de informações disponíveis; identifica os eventos, produtores e consumidores associados.

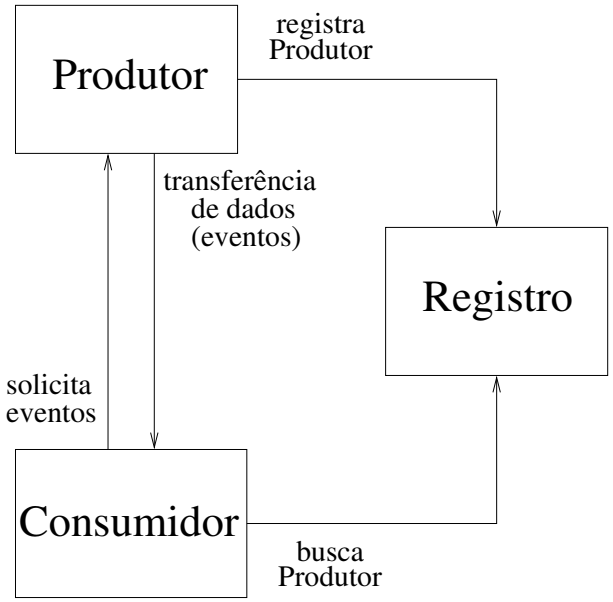

Fig. 2.1: Componentes da Grid Monitoring Architecture

Funcionalidades

As funcionalidades e *interfaces* previstas para estes componentes são:

- Registro:
  - diretório onde são registradas as informações disponíveis ou necessárias. Deve ser utilizado para busca das informações e suas origens, ou então para publicação das informações coletadas;
  - funções que devem ser suportadas: inclusão, alteração e remoção de entradas do diretório; busca de determinado produtor ou consumidor, baseada em critérios de seleção;
- Produtor:
  - envia dados a um consumidor por meio de uma interface. Um produtor pode ter diversas interfaces utilizadas simultânea e independentemente;
  - a interface de um produtor usa um protocolo padrão e gera dados (eventos) em um formato padrão. Existem diversos protocolos-padrão, e um produtor pode suportar múltiplas opções;

  - funções básicas: manutenção das informações no serviço de diretório; tratamento de solicitações de consumidores; tratamento e cancelamento de assinaturas de consumidores; localização de consumidores; notificação de eventos a consumidores;
  - pode ainda implementar: políticas de acesso aos dados; seleção de dados; caching; pré-processamento dos dados;

- Consumidor:
  - componente que utiliza uma interface para receber dados de um produtor
  - a interface de um consumidor usa um protocolo padrão e espera receber dados em um formato padrão;
  - funções básicas: localização de eventos e produtores; manutenção das informações no serviço de diretório; solicitações de eventos a produtores; solicitação e cancelamento de assinaturas em produtores; aceitação da notificação de eventos de produtores;
  - podem existir diversos tipos de consumidores, como por exemplo, bases de dados de eventos, ou monitores de tempo-real.

Interações

As interações, ou seja, a forma de comunicação entre produtores, consumidores e registro são:

- publicação/assinatura - esta interação é feita em três estágios:
  1. um consumidor demonstra interesse por determinado tipo de evento, previamente publicado, e faz uma assinatura do mesmo;
  2. o consumidor recebe os eventos do produtor correspondente;
  3. o produtor, a seu critério, ou o consumidor quando o desejar, finaliza a interação, indicando o final da assinatura;
- solicitação/resposta - esta interação é feita em dois estágios:
  1. um consumidor solicita um determinado tipo de evento para seu produtor;
  2. recebe uma única resposta do produtor, contendo todos os eventos deste tipo existentes;
- notificação - um produtor envia todos os eventos de um determinado tipo para um consumidor, em uma única notificação.

Republicadores ou Intermediários

A principal característica da GMA é a separação entre a descoberta e a obtenção de informações, o que permite a existência de intermediários que podem ser responsáveis pela entrega, filtragem ou armazenagem dos eventos, criando uma hierarquia de componentes, como pode ser visto na Fig. 2.2, a qual é importante para a manutenção da escalabilidade do sistema. Estes componentes intermediários são também chamados de componentes compostos ou republicadores.

Podem existir diversos tipos de consumidores, como por exemplo:

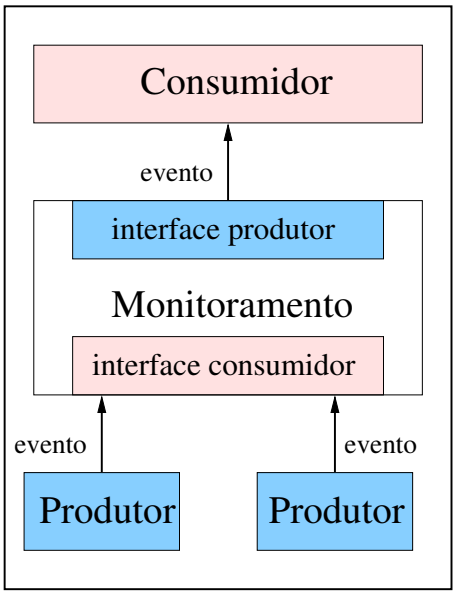

Fig. 2.2: Hierarquia de produtores e consumidores na arquitetura GMA

- *Archiver*: obtém as informações a serem monitoradas e as arquiva para recuperação posterior. Pode atuar também como um produtor, oferecendo as informações armazenadas para outros consumidores;
- *Real-time*: coleta e exibe as informações a serem monitoradas em tempo real;
- *Overview*: coleta informações de diversos produtores e as compõe para disponibilizar um conjunto de informações agregadas.

Implementação

Por se tratar de uma proposta de arquitetura, a GMA não define detalhes de implementação como o modelo de dados, o *schema* dos eventos, protocolos, serviço de registro etc.

Por exemplo, ferramentas implementadas conforme esta arquitetura podem suportar diversos protocolos como SOAP/HTTP, LDAP ou ainda XML/BXXP, entre outros. Estes protocolos não serão discutidos em detalhes, pois não são objetos deste estudo; entretanto uma breve descrição dos mesmos pode ser encontrada no Apêndice E.

No exemplo de implementação e uso da GMA da Fig. 2.3 podemos identificar um possível fluxo:

- informações são coletadas por sensores existentes em três computadores;
- o Produtor recebe as informações dos sensores e registra os eventos no Serviço de Diretório;
- um Consumidor (tempo real) faz a assinatura de todos os tipos de eventos disponíveis;
- outro Consumidor (*archiver*) faz a assinatura de eventos que devem ser armazenados;
- o Produtor envia os Eventos para os Consumidores, conforme as assinaturas existentes;
- o Consumidor (tempo real) utiliza os dados contidos nos eventos recebidos para visualização e análise de desempenho;

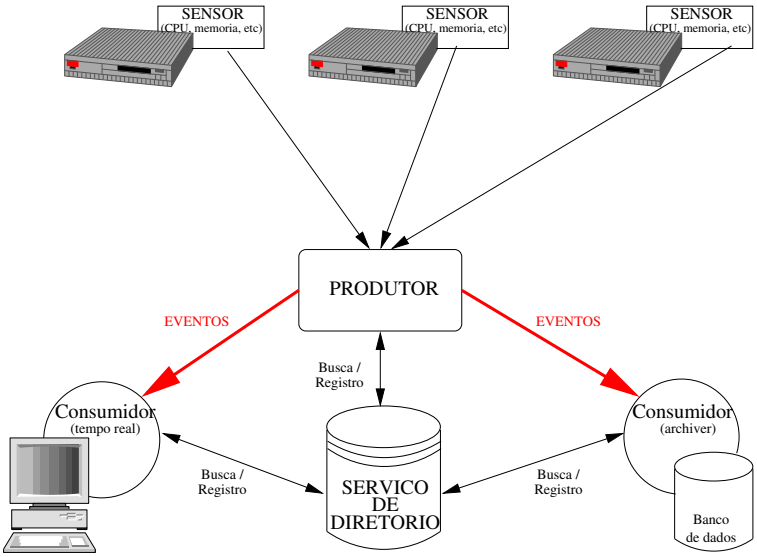

Fig. 2.3: Exemplo de implementação da GMA

- o Consumidor (*archiver*) armazena em um banco de dados as informações contidas nos eventos recebidos.

2.5 Revisão bibliográfica

Vários grupos de pesquisa têm explorado e apresentado diferentes abordagens para o desenvolvimento de ferramentas de monitoramento.

Podemos dividir as ferramentas de monitoramento de ambientes de grade computacional em três grandes categorias:

- ferramentas para plataformas específicas:

  desenvolvidas para atender características e necessidades de um sistema em particular. Implementam as funções necessárias para este ambiente, de forma personalizada e otimizada;
- ferramentas de uso genérico:

  desenvolvidas de forma independente do ambiente em que são usadas. Em geral, implementam parte das funcionalidades necessárias e o desempenho da implementação pode ser comprometido pela forma genérica de obtenção e tratamento das informações (não há otimização);
- ferramentas de base, para combinações com outras ferramentas:

  implementam subconjuntos de funcionalidades que precisam ser combinados para que sejam obtidas todas as informações desejadas por um ambiente.

Por ser uma área de pesquisa em plena atividade, existem muitas publicações relacionadas ao monitoramento de ambientes de grade, bem como um grande número de ferramentas. Aqui é apresentada uma breve revisão bibliográfica e feita uma análise de algumas das ferramentas disponíveis.

Entre as diversas publicações disponíveis na bibliografia, duas são evidenciadas por apresentar propostas de classificação das ferramentas existentes: "A Taxonomy of grid monitoring systems" [51] e "APART-2 White Paper on Performance Tools for the Grid: State of the Art and Future" [52]. Utilizando as formas de classificação apresentadas nestes artigos podemos comparar e avaliar algumas das ferramentas de monitoramento existentes.

No Apêndice A são apresentadas brevemente algumas das ferramentas de monitoramento existentes para ambientes de computação em grade, identificando características importantes e abordagens adotadas na sua implementação. Foram analisadas as ferramentas que implementam arquiteturas e soluções diferenciadas como, por exemplo, aquelas que oferecem suporte à linguagem Java (JAMM, *EDG Logging and Bookkeeping*, MDS e NetLogger) e, desta forma, atendem o requisito de portabilidade, essencial para monitoramento de ambientes heterogêneos. Também são apresentadas ferramentas de propósito geral (Ganglia, Nagios, MonALISA e MapCenter), de uso mais abrangente e não vinculadas a um ambiente ou sistema em particular e algumas que implementam a arquitetura GMA (R-GMA, pyGMA e MA-GMA).

2.5.1 Padrões e parâmetros de comparação

A avaliação da qualidade e do desempenho de ambientes de grade computacional por seus usuários, desenvolvedores de aplicações e administradores depende da disponibilidade de informações sobre os recursos integrantes da grade e sobre a execução de aplicações nestes ambientes. E são as ferramentas de monitoramento que devem prover estas informações de forma clara e objetiva.

Os ambientes de grade computacional são caracterizados por uma grande diversidade de recursos, de aplicações e também de usuários, que têm diferentes expectativas com relação à utilização da grade. Devido a esta amplitude de requisitos a serem satisfeitos, as ferramentas de monitoramento são, em geral, desenvolvidas e personalizadas para os ambientes onde serão implementadas, adequando-se às arquiteturas e aos usuários destes ambiente e utilizando diferentes técnicas e abordagens. Desta forma, as ferramentas de monitoramento existentes, ou em desenvolvimento, diferem também pelos sistemas operacionais suportados, pelo público atendido e pela abrangência: enquanto algumas ferramentas pretendem abordar todos os aspectos do monitoramento, outras mantêm o foco em porções específicas.

Por estas razões, não existem padronizações ou parâmetros que permitam uma comparação direta de ferramentas de monitoramento para ambientes de grade computacional, tanto com relação às informações coletadas, quanto à forma de apresentação das mesmas.

As ferramentas listadas a seguir exemplificam ferramentas desenvolvidas e implementadas de forma personalizada, adequadas às necessidades dos ambientes monitorados e às preferências dos seus usuários.

- A ferramenta Condor Hawkeye [33] coleta, armazena e disponibiliza informações sobre os computadores conectados a ambientes que utilizam o sistema Condor, por meio da execução periódica de programas para coleta de informações nestes computadores. A ferramenta Hawkeye configura o sistema Condor para a execução de *scripts* que produzem como resultados pares contendo atributo/valor no formato ClassAd [53] que faz o mapeamento entre nomes de atributos e valores associados a estes. Utilizando facilidades do sistema Condor e das ClassAds, que são utilizadas pra descrever *jobs*, computadores e demais recursos, a ferramenta

Condor Hawkeye permite o monitoramento de diversos atributos e pode ser personalizada para cada computador conectado ao ambiente Condor.

- Para o sistema Sardes foi desenvolvida uma ferramanenta de monitoramento baseada na infraestrutura Dream [17]. Esta infraestrutura permite o desenvolvimento de serviços distribuídos e dinamicamente configuráveis, os quais se comunicam de forma hierárquica e assíncrona. Este sistema de monitoramento [54] possibilita uma visão geral dos recursos disponíveis na grade computacional, e não das aplicações, com o objetivo de manter informações sobre os recursos e seus estados. Este monitoramento é feito em três fases: a primeira delas coleta indicadores e sinais de alarme dos recursos monitorados; a segunda processa as informações coletadas, agregando e filtrando, e encaminha-as à terceira fase, responsável pelo seu arquivamento.
- O sistema GridICE [55], *"the eyes of the Grid"*, foi desenvolvido com o objetivo de oferecer fácil integração aos softwares de *grid computing* utilizados no EGEE (Enabling Grids for E-sciencE). Possui estrutura modular, em cinco camadas que implementam desde sensores e produtores até os consumidores finais das informações coletadas. Estas camadas são implementadas como:

  - sensores - utiliza a ferramenta Lemon [56];
  - serviços de diretório - utiliza o Globus MDS (Monitoring & Discovery System) [57] e R-GMA;
  - coletores - utiliza o Nagios [58] para obter dados e um servidor de base de dados PostgreSQL [59] para armazenamento das informações;
  - serviços de detecção, notificação e análise de dados - foi desenvolvida ferramenta própria;
  - apresentação - feita por meio de interface gráfica baseada em web, desenvolvida em PHP (Hypertext Preprocessor) [60].

- A ferramenta visPerf [44] foi desenvolvida no projeto NetSolve [61] para monitoramento e vizualização de informações sobre recursos de grades computacionais. Estas informações são obtidas por meio de análise de arquivos de log ou de API disponível no software de *grid*. É um exemplo do monitoramento indireto, ou passivo, apresentado no ítem 2.2.1.
- SIMON [23] é uma ferramenta desenvolvida para monitoramento do sistema UNICORE. Permite que sejam definidos os testes e os executa automaticamente, em intervalos regulares, nos ambientes especificados. Pode ser configurado para, quando detectar falhas, enviar mensagens para um *e-mail* indicado. Esta ferramenta permite diferentes configurações e pode ser facilmente estendida para atender requisitos específicos de um ambiente.

Entre as ferramentas listadas, o software GridICE utiliza outras ferramentas para a implementação de funcionalidades específicas: Lemon, MDS e Nagios, entre outras.

Existem também outras ferramentas para monitoramento que podem ser usadas em conjunto, por serem complementares, como é o caso do MonALISA e Ganglia. Estas ferramentas são de uso mais abrangente, portanto mais complexas. Segue uma breve descrição de algumas delas.

- Ganglia Cluster Toolkit [62]: uma ferramenta muito utilizada para monitoramento de *clusters* e agrupamentos hierárquicos de *clusters*. Coleta informações sobre o estado dos computadores e sistemas, mas não sobre aplicações, e as disponibiliza por meio de uma interface *web*.

- MonALISA (MONitoring Agents using a Large Integrated Services Architecture) [63]: baseado em Java, JINI [64] e Web services. Como um sistema autônomo e *multi-threaded* baseado em agentes, pode ser integrado ao Ganglia Toolkit e a alguns sistemas de gerenciamento de filas, com o objetivo de oferecer informações para o monitoramento de grandes sistemas distribuídos.
- Lemon [56]: infra-estrutura para monitoramento escalável e flexível, obtém informações por meio de sensores executados em clientes de forma distribuída. Estas informações são armazenadas em uma base de dados Oracle ou arquivos texto, e posteriormente processadas por outras ferramentas, como GridIce.
- R-GMA (Relational Grid Monitoring Architecture) [65] é uma das implementações da arquitetura GMA com os componentes consumidor, que solicita informações, produtor, que provê informações, e registro, que intermedia a comunicação entre consumidores e produtores. O R-GMA também é desenvolvido como parte do projeto EGEE (*Enabling Grids for E-sciencE*) e tem por objetivo representar as informações coletadas como uma grande base de dados relacional, que podem ser obtidas por meio comandos SQL (*Structured Query Language*). O R-GMA é utilizado como um serviço de diretório pelo sistema GridICE.

2.5.2 Taxonomia conforme a GMA

Esta seção discute a proposta de classificação das ferramentas para monitoramento de ambientes de grade apresentada no artigo "A Taxonomy of grid monitoring systems" [49]. A proposta é baseada na padronização definida pela GMA (Grid Monitoring Architecture) do GGF (Global Grid Forum), considerando a abrangência, a escalabilidade, o uso geral e a flexibilidade das ferramentas. A classificação proposta naquele artigo facilita a comparação e a avaliação das ferramentas analisadas, como apresentado a seguir.

Conceitos e terminologia

Para classificar as ferramentas de monitoramento são usados conceitos e terminologia descritos a seguir [49].

- Entidades: recursos compartilhados pela rede como, por exemplo, processadores, memórias, dispositivos de armazenamento, conexões de rede, aplicações e processos.
- Evento: um conjunto de informações, com carimbo de tempo, representadas em uma estrutura específica.
- Tipo de evento: um identificador único de uma estrutura de evento.
- Schema de evento: define a estrutura e semântica de todos os eventos de forma que, dado um tipo de evento, é possível obter-se a estrutura e semântica do evento.
- Sensor: é um processo que monitora recursos e produz eventos. Existem sensores passivos, ou seja, que obtêm informações disponíveis, normalmente por meio de facilidades do sistema operacional, e sensores ativos, ou seja, que obtêm medidas por meio de algum tipo de processamento e que, por esta razão, são mais intrusivos que os passivos.

Os requisitos de um sistema de monitoramento são:

- escalabilidade - deve suportar de forma eficiente o aumento do número de recursos, eventos e usuários;
- extensibilidade - deve permitir o monitoramento de diferentes tipos de eventos e a inclusão de novos tipos de eventos deve ser feita facilmente;
- políticas de medição - podem ser periódicas ou sob demanda, solicitadas ou enviadas sem requisição prévia;
- portabilidade - a ferramenta de monitoramento e, em particular, os sensores e eventos devem ser independentes de plataforma;
- segurança - deve implementar controle de acesso a informações e tranporte seguro das informações monitoradas.

Fases do processo de monitoramento

No artigo, o processo de monitoramento é representado por quatro fases.

1. geração de eventos pelos sensores;
2. processamento de eventos gerados, que em geral é aplicação de algum tipo de filtro;
3. distribuição ou transmissão dos eventos aos interessados, ou consumidores;
4. apresentação das informações de forma a torná-las disponíveis aos usuários.

Tab. 2.1: Relacionamento entre fases de monitoramento e GMA

GMA	$Fase$			
	1	2	3	4
Sensor	X	[X]		
Produtor	[X]	[X]	X	
Consumidor		[X]		X

(Obs: os colchetes indicam fases opcionais)

A Tab. 2.1 relaciona as fases de um sistema de monitoramento (geração, processamento, distribuição, armazenamento e apresentação das informações) e a GMA:

- sensor: gera eventos (primeira fase) e pode, opcionalmente, processá-los e torná-los disponíveis para consumidores locais (segunda fase);
- produtor: pode ter sensores próprios (primeira fase) e pode processar os eventos (segunda fase), devendo distribui-los a consumidores remotos (terceira fase);
- consumidor: pode processar os eventos (segunda fase), devendo apresentar as informações (quinta fase).

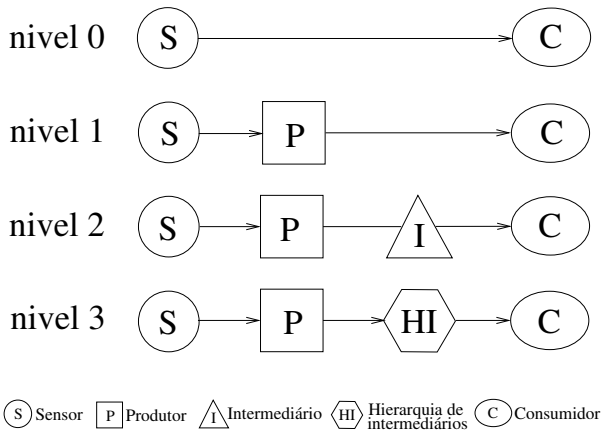

Fig. 2.4: Categorias da classificação em 4 níveis

Classificação

A classificação é baseada na arquitetura GMA, apresentada na seção 2.4 (Fig. 2.2), e composta de quatro níveis, determinados pela forma de implementação de produtores e intermediários (ou republicadores).

Na Fig. 2.4 é apresentado um esquema que resume esta classificação.

- **Nível 0**

  Os eventos são enviados dos sensores aos consumidores, tanto *online* quanto *offline*. No caso *online* os sensores armazenam localmente as informações que são recuperadas e exibidas por uma aplicação *web*. Estes sistemas são auto-contidos e não permitem a distribuição de informações. O sistema MapCenter [66] é classificado neste nível.

- **Nível 1**

  Os sensores são implementados nas mesmas máquinas que os produtores, ou a funcionalidade dos sensores está embutida nos produtores. O sistema Autopilot [67] é classificado neste nível.

- **Nível 2**

  Além de produtores, os sistemas deste nível implementam pelo menos um tipo de republicador, ou intermediário, que tem função pré-definida. São classificados neste nível os sistemas com republicador centralizado: CODE [68], GridICE [55], GridRM [69] e Hawkeye [33]; os sistemas com republicadores distribuídos: HBM [70], JAMM [71], NetLogger [72], OCM-G [73], Remos [74] e SCALEA-G [75]; e os sistemas com republicadores distribuídos e replicados: NWS [76].

- **Nível 3**

  São os sistemas de monitoramento mais flexíveis, que implementam diversos republicadores, que podem ser configurados e organizados em uma estrutura hierárquica. São os sistemas que permitem melhor escalabilidade, em função da hierarquia implementada. São classificados neste nível os sistemas: Ganglia [62], Globus MDS [14], Mercury [77], MonALISA [63], Paradyn-MRNet [78] e R-GMA [65].

2.5.3 Classificação conforme características

Nesta seção é discutida a classificação apresentada no artigo "APART-2 White Paper on Performance Tools for the Grid: State of the Art and Future" [52], onde vinte e seis ferramentas de monitoramento são classificadas com base em características consideradas necessárias, ou recomendadas, para o ambiente de *grid computing*. Os autores avaliam as características utilizadas para classificação conforme os seguintes grupos:

- Público alvo: as ferramentas são classificadas conforme o público atendido pela mesma. É verificado se a ferramenta pode oferecer informações para usuários do ambiente de *grid*, para desenvolvedores de aplicações, desenvolvedores de softwares para infraestrutura de *grids*, administradores de *grids* ou ainda para outros componentes dos softwares de *grid*;
- Funcionalidades: na classificação por esta característica são analisadas as implementações das funcionalidades para o monitoramento, para a análise das informações coletadas, para a predição de desempenho, para o direcionamento na utilização da grade e também para a visualização das informações;
- Recursos: nesta classificação são analisadas a forma de apresentação das informações, a forma de uso da ferramenta e as linguagens de programação suportadas. A apresentação das informações pode ser feita com base: nas aplicações; no estado dos jobs; na infraestrutura de *grid*; nos recursos de processamento, de armazenamento ou de rede. A forma de uso pode ser interativa, *off-line* ou *online*.
- Inclusão de sensores, ou instrumentação: para a obtenção de dados é necessária a existência de sensores, que coletam as informações desejadas. Aqui é analisada a forma como a inclusão dos sensores para obtenção de informações sobre recursos e aplicações pode ser feita: de forma automática, manual, estática ou dinâmica, se codificada no software, ou por meio de hardware acoplado ao ambiente;
- Arquitetura: neste grupo de características são analisados aspectos relacionados à arquitetura implementada, além dos serviços disponíveis: tipo de grade suportada, organização hierárquica ou *peer-to-peer*, análise centralizada ou distribuída, autenticação, autorização, uso de criptgrafia, firewalls, entre outros;
- Interfaces: como a aquisição de informações pelas ferramentas de monitoramento não é feita de forma independente da infraestrutura de *grid computing*, neste item são analisadas a conexão e a integração das ferramentas com a infraestrutura, como: formatação dos dados, quais são as operações suportadas sobre os dados armazenados e quais as linguagens suportadas pela API.

A classificação apresentada no artigo não tem por objetivo comparar ou avaliar as ferramentas de monitoramento, devendo ser utilizada como fonte de informações para a verificação da adequação e possibilidade de implementação de uma ferramenta em um ambiente de grade.

2.6 Conclusões

Neste capítulo foram apresentados alguns desafios que devem ser superados na implementação de uma ferramenta de monitoramento como: a manutenção da escalabilidade do sistema, a necessidade de não intrusão, o pequeno consumo de recursos e o baixo tráfego de rede. Também foi destacada a

importância de uma interface gráfica, onde as informações possam ser obtidas de forma eficiente e clara, e que permita o acesso ao estado atual e estados passados do ambiente. Estes desafios devem ser considerados como fatores de sucesso na implementação de uma ferramenta de monitoramento.

Foram discutidas possíveis abordagens para a implementação de uma ferramenta de monitoramento: de forma direta ou ativa, quando são criados os elementos que têm a função de obter as informações; ou de forma indireta ou passiva, que busca informações em elementos já existentes como *logs* ou mensagens trocadas entre componentes do sistema. Também foram apresentados alguns conceitos, terminologia e definições que são utilizadas no restante do trabalho, como produtores, consumidores e eventos.

Destacou-se a GMA (*Grid Monitoring Architecture*), que propõe padrões e interoperabilidade entre os softwares desenvolvidos para monitoramento em *Grid Computing*.

Na revisão bibliográfica foram destacados os artigos "A Taxonomy of grid monitoring systems" [51] e "APART-2 White Paper on Performance Tools for the Grid: State of the Art and Future" [52]. Estes dois artigos propõem critérios que permitem a classificação das principais ferramentas de monitoramento existentes. As características de algumas das ferramentas para monitoramento encontradas atualmente na literatura são apresentadas no Apêndice A.

Capítulo 3

Monitoramento no sistema JoiN

Neste capítulo são apresentadas as principais características relacionadas ao sistema JoiN para computação em grade, em desenvolvimento na Faculdade de Engenharia Elétrica e de Computação da UNICAMP: sua classificação como sistema de computação em grade, sua arquitetura e implementação, as características das aplicações processadas neste sistema, o perfil de seus colaboradores, além de aspectos mais específicos do sistema como escalonamento, segurança, portabilidade e escalabilidade.

Este sistema não dispõe de uma ferramenta para monitoramento e, neste ambiente de processamento baseado na Internet, tão heterogêneo e dinâmico, a existência de uma ferramenta eficiente e flexível para monitoramento é essencial para sua administração, avaliação e otimização.

São também discutidas as características desejadas para uma ferramenta de monitoramento para este sistema e definidos os critérios que devem ser considerados na avaliação desta ferramenta.

Com base nos estudos realizados, este trabalho procura aproveitar as melhores características das ferramentas de monitoramento existentes para implementação de uma ferramenta de monitoramento no sistema JoiN, de forma a satisfazer as necessidades dos usuários e administradores deste sistema. A ferramenta de monitoramento implementada, chamada JoiNMon, apresenta algumas características que a diferenciam de outras propostas e permitem que sejam mantidas as premissas básicas do sistema JoiN: trabalhar com um grande número de computadores heterogêneos e com grande volume de dados, mantendo a escalabilidade e o desempenho. O projeto e a implementação da ferramenta JoiNMon também são descritos neste capítulo.

3.1 JoiN e computação em grade

Antes da classificação do sistema JoiN como um sistema para computação em grade é importante observar que *Grid Computing* continua sendo apresentada como uma tecnologia emergente, evolução do processamento distribuído. Ainda estão sendo padronizadas e implementadas as ferramentas e *middlewares* necessários para a superar os desafios que esta tecnologia oferece, como citado por Jacob [79], com relação a:

- segurança;
- interface com os usuários;
- escalonamento de jobs;

- gerenciamento de carga;
- gerenciamento de dados;
- gerenciamento de recursos.

Ainda hoje é difícil definir *Grid Computing*, pois este termo tem significados diferentes para diferentes pessoas.

Usando como argumento de pesquisa no Google "define: grid computing" obtem-se algumas definições interessantes:

- *"Applying the resources of many computers in a network to a single problem at the same time . . ."*

  (www.bl.uk/about/strategic/glossary.html)
- *"Grid computing is a model for allowing companies to use a large number of computing resources on demand, no matter where they are located . . ."*

  (www.informatica.com/solutions/resource_center/glossary/default.htm)
- *"The activity of using a computational grid . . ."*

  (books.nap.edu/html/up_to_speed/appD.html)
- *"Grid computing provides clustering of remotely distributed computing . . ."*

  (www.aarohi.net/info/glossary.html)
- *"Grid computing uses the resources of a many separate computers connected by a network (usually the internet) to solve large-scale computation problems. The SETI@home project, launched in the mid-1990s, was the first widely-known grid computing project . . ."*

  (en.wikipedia.org/wiki/Grid_computing)

Uma definição simples, herança do processamento distribuído, e que engloba as definições mais comuns pode ser: *computação em grade é um modelo que permite a apresentação para seus usuários de um enorme sistema de computação virtual, composto por conjuntos de servidores, sistemas de armazenamento e redes*. Desta forma é possível unir o poder de múltiplos recursos e apresentá-los aos usuários como se fosse apenas um sistema.

Existe ainda a definição de PRC (Public Resouce Computing) [4], também apresentada como Computação Voluntária [80], ou então Computação distribuída pública [81], que consiste na utilização de recursos ociosos de computadores conectados a Internet, para a criação de um ambiente virtual de processamento maciçamente paralelo (MPVC) [82]. É nesta última definição que classificamos o sistema JoiN.

3.2 JoiN: arquitetura e características

O mundo assiste a uma crescente utilização de computadores para as mais diversas atividades e, com a popularização do uso da Internet, assiste também à integração destes computadores em uma imensa rede mundial. Os computadores conectados à Internet vão de supercomputadores, dos grandes laboratórios de pesquisa, a computadores pessoais. O poder computacional deste grande número de computadores conectados, se agregados, criaria um supercomputador de proporções imensas, abrindo

perspectivas para a solução de problemas extremamente complexos, de solução até então impossível. O desafio é permitir a utilização do poder computacional de muitos destes computadores conectados à Internet, de forma transparente, tirando proveito do que cada computador tem a oferecer.

O sistem JoiN [83] está sendo desenvolvido para permitir a criação e utilização de ambientes de grade computacionais com recursos públicos e, com isto, viabilizar a execução de aplicações maciçamente paralelas.

Desenvolvido em linguagem Java, com uma estrutura baseada em componentes, o sistema JoiN tem como característica principal a grande portabilidade, pois pode ser executado em qualquer computador em que houver uma implementação da Máquina Virtual Java (JVM) [84].

A escalabilidade, essencial para ambientes de grade, é obtida pela organização hierárquica dos computadores em grupos, gerenciados de forma independente. Além disto, o sistema JoiN trata de funções importantes para este tipo de sistemas como escalonamento, replicação, tolerância a falhas e segurança.

O sistema JoiN utiliza um escalonamento de tarefas dinâmico e flexível, que procura adaptar-se ao dinamismo do ambiente, pois colaboradores conectam-se e deixam o sistema freqüentemente, e usa replicação de tarefas para prover um certo nível de tolerância a falhas [85].

A utilização da linguagem Java e a execução na arquitetura de JVM permite um controle sobre as autorizações concedidas ao código que está sendo executado na VM, e oferece uma forma de proteção contra a execução de códigos maliciosos. Internamente, a segurança do sistema JoiN é implementada por meio de mecanismos de autenticação e regras codificadas no próprio sistema.

A implementação do sistema é baseada no conceito de componentes (também chamados de serviços): peças de software que podem ser combinadas de diversas formas para compor ambientes bastante flexíveis.

3.2.1 Arquitetura

A arquitetura do sistema JoiN [5], como pode ser visto na Fig. 3.1, é baseada em componentes hierarquicamente organizados: um mediador (*server*), módulos de administração (*jack*) e grupos interconectados. Cada um destes grupos é formado por um *coordinator* e diversos *workers*. Os *workers* são responsáveis pela execução das aplicações; portanto o número deste tipo de componente pode ser muito grande.

Cada computador conectado ao JoiN executa pelo menos um dos seguintes componentes, que têm as seguintes funções:

- *server*: gerencia as operações de alto nível, como a inclusão de um novo computador ou a submissão de uma aplicação para execução; sua principal função é servir como ponto de entrada para computadores participantes;
- *coordinator*: gerencia os grupos de componentes *worker* e a execução das aplicações submetidas; também é responsável pela comunicação externa ao grupo, ou seja, entre grupos e do grupo com o *server*;
- *worker*: realiza o trabalho computacional, ou seja, processa as tarefas das aplicações;
- *jack (Join Administration and Configuration Kit)*: módulo de administração básica, com interface para configuração e administração do sistema.

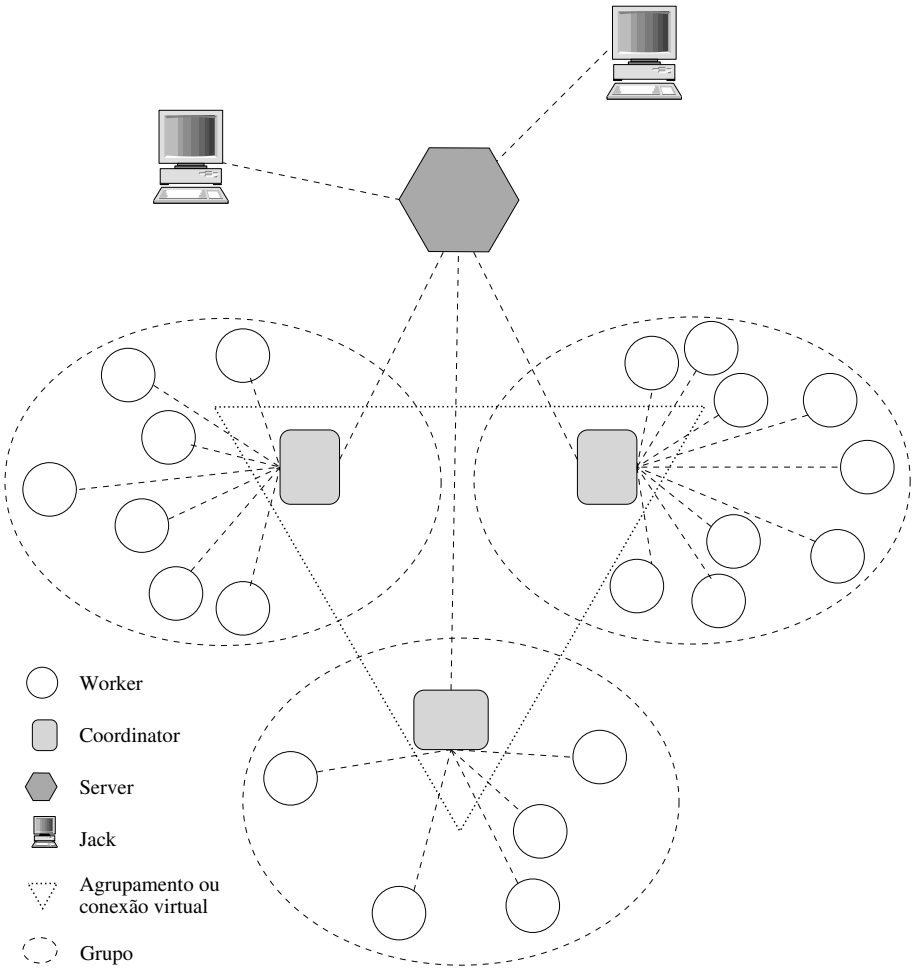

Fig. 3.1: Estrutura do sistema JoiN

Comunicação Interna

A comunicação entre os componentes do sistema JoiN é feita da seguinte forma:

- *coordinators* comunicam-se com outros *coordinators* e com o *server*;
- *workers* podem comunicar-se apenas com seus *coordinators*, e a conexão deve ser sempre iniciada pelo *worker*. A comunicação direta entre *workers* não é permitida.

Estas restrições têm como objetivo evitar as barreiras criadas pelo uso de *firewalls*, pois apenas o *server* e os *coordinators* são acessados remotamente. Além disso, estas restrições viabilizam a implementação de mecanismos de tolerância a falhas mais eficientes.

Serviços

Cada componente do JoiN (*server, coordinators, workers e jacks*) compartilha a mesma arquitetura, baseada no conceito de serviços. Um serviço tem uma função específica como, por exemplo, monitoramento e é formado por um conjunto de módulos.

Cada um dos serviços do sistema JoiN é implementado por meio de módulos que contêm o código a ser executado nos diferentes componentes do sistema (*server, coordinator, worker* e *jack*). Desta forma, cada serviço é implementado por:

- um módulo a ser executado no componente *server*, em geral identificado como S_*service_id*;
- um módulo a ser executado nos componentes *coordinator*, em geral identificado como C_*service_id*;
- um módulo a ser executado nos componentes *worker*, em geral identificado como W_*service_id*;
- um módulo a ser executado nos componentes *jack*, em geral identificado como J_*service_id*;
- um ou mais módulos que podem ser executados em qualquer dos componentes, para os quais não há regra para identificação.

A associação entre os módulos que implementam um serviço e os componentes em que são executados é feita quando o serviço é instalado no sistema JoiN. Esta instalação é feita por meio do componente *jack*.

Além da organização em módulos, a estrutura interna de cada serviço é definida por um conjunto de *interfaces* de *software* correspondentes às funcionalidades que implementa. Uma interface básica, denominada *ServiceCode*, deve obrigatoriamente ser implementada por todos os módulos de serviços para que o sistema JoiN controle o acesso destes módulos a certos recursos compartilhados. Este controle de acesso é feito quando serviços realizam chamadas entre si, por meio de um mecanismo de *proxy* de classes disponível na plataforma Java. Este mecanismo permite que chamadas a métodos de uma determinada classe sejam interceptados para que seja realizado um tratamento qualquer antes da invocação efetiva do método em questão. A interação entre os diversos serviços é então gerenciada pelo *ServiceManager*, como ilustrado na Fig. 3.2.

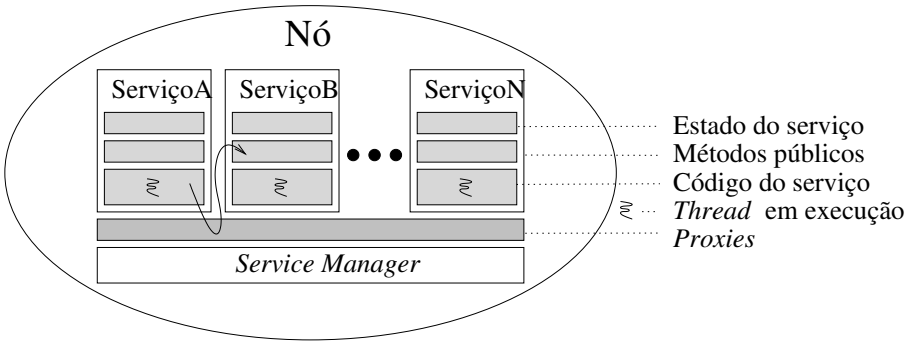

Fig. 3.2: Interação entre serviços do sistema JoiN

A utilização de *proxies* de classes nos serviços do sistema JoiN também permite que sejam realizadas funções como *class profiling*, que consiste na avaliação da freqüência com que os métodos de uma determinada classe são invocados e qual o tempo médio dispendido na execução destes métodos. Trata-se portanto de uma ferramenta que pode ser utilizada para monitoramento dos serviços.

3.2.2 Aplicações JoiN

As aplicações devem ser programadas em Java, para manter a portabilidade do sistema, e sua estrutura especificada por meio de um arquivo *Parallel Application Specification* (PAS). Neste arquivo são descritas as dependências entre os lotes de tarefas, as precedências e o fluxo de dados, por meio de uma linguagem de especificação de aplicações paralelas chamada *Parallel Application Specification Language* (PASL) [85].

O modelo de aplicação consiste basicamente em relações de dependência entre lotes de tarefas.

Na codificação e especificação de uma aplicação é importante observar as seguintes definições:

- uma tarefa é uma seqüência ordenada de operações aritméticas, matemáticas, de desvio condicional ou de atribuição/entrada/saída de dados que implementam algum tipo de processamento determinístico e finito sobre um bloco de dados de entrada a fim de produzir um bloco de dados de saída, ou seja, é um código de programa que trata dados de entrada e produz resultados;
- os dados a serem enviados (entrada ou saída) devem ser codificados em uma estrutura de dados serializável, de maneira a permitir o uso de recursos de comunicação Java;
- um lote de tarefas consiste na aplicação de uma determinada tarefa sobre blocos de dados de entrada;
- a relação de dependência entre lotes de tarefas define quais blocos de saída, de um ou vários lotes, serão utilizados como dados de entrada para os lotes subseqüentes.

As aplicações são instaladas e submetidas para execução por meio dos componentes *jack* conectados ao *server*. As aplicações são instaladas no *server* e, ao comando de submissão de uma aplicação para execução, a mesma é enviada ao *coordinator* de um dos grupos. Este *coordinator* divide e gerencia a execução desta aplicação nos componentes *worker* conectados ao grupo. São necessários três passos para que um usuário tenha sua aplicação processada no sistema JoiN, descritos a seguir. Os dois primeiros são executados pelos usuários por meio de interações com o componente *jack*. Uma vez que uma aplicação está instalada, poderá ser executada quantas vezes forem necessárias.

1. Instalação - o usuário indica a localização do arquivo PAS e das classes que serão utilizadas no processamento da aplicação.
2. Submissão - o usuário informa que a aplicação previamente instalada deverá se processada.
3. Execução - o sistema JoiN processa as informações contidas no arquivo PAS e distribui o processamento da aplicação.

3.2.3 Escalonamento

A eficiência do sistema de computação em grade depende de um escalonamento eficiente, que deve determinar a melhor forma de executar as aplicações nos computadores disponíveis, onde o ambiente é, além de heterogêneo, muito dinâmico, pois os computadores que compõem o ambiente não são dedicados a esta função e podem ter o desempenho alterado, bem como podem ser desconectados do ambiente.

Para manter informações sobre o desempenho de cada um dos computadores é executada uma série de algoritmos de *benchmark*, no momento em que o computador conecta-se ao sistema, e esta informação é mantida pelo *coordinator*.

O algoritmo de escalonamento do JoiN é chamado *Generational Scheduling with Task Replication* (GSTR) [85] e tem como principais caracteríticas:

- a distribuição de tarefas não é feita exclusivamente de forma estática, pois uma parte das tarefas é enviada aos componentes *worker* no início da execução da aplicação e as demais tarefas são distribuídas sob demanda, ou seja, os componentes *worker* que terminam a execução das tarefas inicialmente alocadas a eles recebem um novo conjunto de tarefas para execução. Nesta redistribuição é novamente avaliado o desempenho dos computadores e o escalonamento adapta-se ao dinamismo do ambiente;
- implementa a replicação da execução de uma tarefa ainda não concluída, introduzindo um esquema de tolerância a falhas pela réplica da execução e aproveitamento do primeiro resultado obtido; assim, os problemas de comunicação ou de desconexão de colaboradores são atenuados e o sistema se torna adaptável ao dinamismo do ambiente e alterações de desempenho dos componentes *worker*.

3.2.4 Heterogeneidade e portabilidade

A possibilidade de executar aplicações paralelas de granularidade grossa, com alta razão entre computação e comunicação, lidando com a heterogeneidade do ambiente e a necessidade de portabilidade é muito importante em ambientes de computação em grade com recursos públicos.

A heterogeneidade é resultado da utilização de computadores que diferem em arquiteturas de processadores, recursos disponíveis, sistemas operacionais e meios de interconexão utilizados no ambiente.

O ambiente de computação multiplataforma Java foi adotado no desenvolvimento do sistema JoiN e, com a portabilidade desta plataforma, buscou-se superar as barreiras impostas pela heterogeneidade dos computadores participantes. O ambiente Java permite a execução do mesmo código compilado (bytecodes) em máquinas virtuais disponíveis para os ambientes operacionais mais presentes na Internet.

O desenvolvimento de serviços e aplicações em Java permite a utilização de máquinas de diferentes arquiteturas e sistemas operacionais.

3.2.5 Escalabilidade

Um sistema que se propõe a utilizar um número muito grande de computadores, formando um sistema maciçamente paralelo deve ter como ponto forte a escalabilidade. A arquitetura hierárquica do sistema JoiN tem como objetivo garantir sua escalabilidade: o componente *server*, que poderia estabelecer um ponto de gargalo, não participa da execução de aplicações e, se necessário, pode ser reiniciado sem prejuízo da execução das aplicações em curso. Este componente atua como ponto de entrada para novos colaboradores (*workers*) e para os componentes *jack* usados na submissão de aplicações. Os componentes *worker*, que existem em grande número, são distribuidos em grupos, evitando a sobrecarga de um único componente *coordinator* e gargalos no sistema.

3.3 Ferramenta para monitoramento do sistema JoiN

A administração do sistema JoiN sem uma ferramenta que disponibilize e organize informações adequadamente só pode ser feita baseada na análise de arquivos de *log*. Além de ser de difícil interpretação, estes *logs* contêm um volume de dados muito grande e, na maioria das vezes, não relevantes ao que se pretende observar.

Uma ferramenta para monitoramento do sistema JoiN, que abra a possibilidade de obtenção e análise de informações sobre sua configuração e seu uso, é essencial para a utilização eficiente deste sistema. Esta ferramenta deve monitorar os serviços e as aplicações do sistema JoiN, coletar e tornar disponíveis informações sobre:

- os componentes do sistema: *server*, *coordinators* e *workers*;
- os usuários do sistema, suas aplicações e tarefas;
- os colaboradores do sistema e seus computadores;
- o ambiente de processamento, incluindo computadores, conexões e tráfego de rede.

Desta forma, por meio das informações apresentadas, a ferramenta de monitoramento deve oferecer subsídios aos usuários, administradores e desenvolvedores de serviços do sistema para:

- identificar mais rapidamente a existência de problemas ou gargalos;
- identificar os recursos disponíveis para a execução das aplicações;
- identificar parâmetros importantes para o escalonamento de processos e balanceamento de carga.

Como consequência, o uso desta ferramenta deve facilitar o desenvolvimento de aplicações de melhor qualidade e a implementação de serviços mais eficientes para o sistema JoiN.

É importante lembrar que o dinamismo dos ambientes criados pelo sistema Join, onde colaboradores podem conectar-se e desligar-se a todo instante e os computadores não têm uso dedicado ou exclusivo, torna pouco provável a re-execução de uma aplicação nas mesmas condições de uma execução anterior. Portanto, além do monitoramento em tempo real, a ferramenta de monitoramento deve armazenar informações e permitir sua análise posteriormente por meio de uma interface gráfica.

3.3.1 Critérios de avaliação

Na avaliação da ferramenta para monitoramento do sistema JoiN serão considerados os vários aspectos descritos a seguir, relativos às características e funcionalidades da ferramenta de monitoramento.

- Público alvo: a ferramenta deve atender usuários, administradores, desenvolvedores de aplicações e serviços, e também outros serviços do sistema.
- Organização: deve adequar-se à arquitetura do sistema JoiN, apresentada na seção 3.2, seguindo a hierarquia de componentes, o modelo e os requisitos do sistema JoiN para implementação de serviços, além de atender as restrições impostas por esta arquitetura com relação à comunicação entre componentes e serviços.

- Funcionalidade: o foco da ferramenta deve ser o monitoramento, ou seja, coleta, exibição e armazenamento de dados de forma eficiente de todas as informações identificadas neste trabalho como relevantes para o monitoramento do JoiN; além disto o armazenamento deve ser feito de forma eficiente para facilitar a recuperação das informações.
- Recursos: o sistema JoiN tem como premissa a utilização da linguagem Java pelas aplicações e pelos serviços, para satisfazer o objetivo da portabilidade. Desta forma, a ferramenta de monitoramento deve ser implementada na linguagem de programação Java e, desta forma, poder ser executada em qualquer computador em que houver uma implementação de JVM.
- Dados disponíveis: devem ser coletadas e armazenadas informações, relativas às aplicações e aos recursos computacionais, que reflitam não só o estado atual do ambiente e das aplicações em execução mas também de informações históricas sobre estados e aplicações passados.
- Visualização: as informações devem ser apresentadas de forma "orientada às aplicações", ou seja, devem retratar o comportamento da execução das aplicações no sistema, e a exibição das mesmas deve ser feita de forma adequada, por meio de uma interface gráfica onde possam ser facilmente obtidas as informações desejadas.
- Escalabilidade - requisito muito importante em ambientes de computação em grade, a ferramenta de monitoramento deve suportar o crescimento do número de usuários, computadores, aplicações, serviços e eventos.
- Intrusão - o processamento da ferramenta não deve sobrecarregar os componentes do sistema JoiN, nem a execução das aplicações; também deve obter e transmitir as informações de forma a não sobrecarregar o tráfego de dados, pois este deve ser reduzido em ambientes fracamente acoplados como o JoiN;
- Segurança - são necessários mecanismos para autenticação de usuários e autorização no acesso a informações;
- Tolerância a falhas - neste ambiente, computadores podem conectar-se e desligar-se a qualquer momento, e a ferramenta deve suportar este dinamismo.

3.4 Avaliação de ferramentas para uso no sistema JoiN

Para ser utilizada no ambiente JoiN a ferramenta deve ser classificada, entre as categorias que apresentamos na seção 2.5, como "de uso genérico", ou então é possível a utilização de um conjunto de ferramentas classificadas como "base para a combinação com outras ferramentas". A ferramenta, ou o conjunto de ferramentas, deve:

- atender os critérios necessários para integração ao ambiente;
- apresentar o estado de cada componente do ambiente JoiN em um determinado momento;
- monitorar aplicações e serviços.

Na busca por uma ferramenta de monitoramento adequada para o ambiente JoiN devem ser avaliadas: as ferramentas mais utilizadas nos ambientes de grade computacional existentes, as ferramentas que apresentam arquitetura que possa ser integrada ao ambiente e também as ferramentas que satisfazem restrições do ambiente JoiN, como o suporte a linguagem Java.

Iniciamos com uma avaliação das ferramentas existentes e apresentadas no Apêndice A.

1. Ganglia [62]

   Este sistema não atende os requisistos do JoiN pois:

   - o uso da base de dados de tamanho fixo da ferramenta RRDtool, utilizada de forma rotativa (*round-robin*), é inadequado para o armazenamento de dados históricos (persistentes) e também para a busca de correlação entre informações;
   - não é possível sua implementação como um serviço do sistema JoiN, por não ser codificado em Java e por não ser possível sua adequação aos requisitos da hierarquia de componentes do JoiN.

2. Nagios [58]

   Este sistema também não atende os requisistos do sistema JoiN pois:

   - assume que os recursos são estáticos, ou seja, não permite a inclusão e exclusão dinâmica de recursos, o que é feito constantemente no ambiente JoiN com a adesão e desconexão de colaboradores;
   - não provê um mecanismo de assinatura/notificação de eventos;
   - requer a instalação de software e criação de usuário em cada computador para sua execução, o que dificulta a instalação, portabilidade e distribuição do sistema JoiN;
   - permite o monitoramento de serviços de rede como por exemplo HTTP, SMTP, SSH e Telnet, e de recursos dos computadores, como espaço em disco e carga de processamento, mas não permite o monitoramento de aplicações executadas em uma JVM;
   - não é possível sua implementação como um serviço do sistema JoiN, por não ser codificado em Java e por não ser possível sua adequação aos requisitos da hierarquia de componentes do JoiN.

3. MonALISA [63]

   Este sistema não atende os requisitos do sistema JoiN pois:

   - em cada computador a ser monitorado deve-se executar um servidor MonALISA para obtenção de dados (produtor), e este não pode ser implementado como um serviço JoiN;
   - implementa a comunicação direta entre agentes, o que também não é permitido na arquitetura de serviços e componentes do JoiN;
   - depende da instalação e configuração de outros sistemas ou pacotes, como Ganglia ou SNMP, para coleta de informações.

4. MapCenter [66]

   Este sistema não atende os requisistos do sistema JoiN pois:

   - não coleta informações sobre os computadores conectados à grade;
   - depende da instalação e configuração de outros sistemas ou pacotes, que implementem o GIS, para obtenção de dados;
   - estabelece conexões TCP para coletar informações sobre a rede, o que não é permitido na arquitetura de serviços e componentes do JoiN.

5. R-GMA [65]

   Este sistema não atende os requisistos do sistema JoiN pois:

   - não permite a configuração das informações a serem coletadas;
   - é baseado em tecnologia *servlet*, inadequada às restrições de comunicação e implementação de serviços no sistema JoiN.

6. JAMM [71]

   Esta é a ferramenta que mais se aproxima do ambiente JoiN: desenvolvida em linguagem Java, implementa uma hierarquia de componentes e permite o monitoramento de computadores, sistemas e aplicações. Entretanto, não é possível utilizá-la diretamente no sistema JoiN pois:

   - em cada computador a ser monitorado devem ser executados agentes (*sensor* e *sensor manager*) para obtenção de dados, que não podem ser implementados como serviços JoiN, por não seguirem a arquitetura de serviços do JoiN;
   - implementa a comunicação direta entre agentes, o que também não é permitido na arquitetura de serviços e componentes do JoiN;
   - depende da utilização e configuração de comandos do sistema, como os comandos usados para obtenção de informações em sistema Linux, o que pode necessitar de previlégios para execução destes agentes, contrariando características fundamentais do JoiN, como a portabilidade e a execução sem privilégios de administrador.

7. pyGMA [86]

   Esta ferramenta também não pode ser implementada conforme as restrições do ambiente JoiN, pois:

   - requer a instalação do Python em todos os computadores conectados ao sistema JoiN. Esta condição contraria a característica de portabilidade do sistema JoiN, que pode ser executado em qualquer computador sem que seus colaboradores precisem instalar ou configurar qualquer *software* além da máquina virtual Java;
   - não pode ser implementada como um serviço do sistema JoiN, pois realiza a comunicação, por meio do protocolo SOAP, entre os componentes do sistema, o que também não é permitido na arquitetura de serviços e componentes do JoiN.

8. MA-GMA [87]

   Esta ferramenta, baseada na tecnologia de agentes móveis, utiliza a arquitetura GMA, implementa uma hierarquia de componentes que podem ser customizados e, desta forma, possibilita o monitoramento de computadores, sistemas e aplicações. Entretanto, da mesma forma que a JAMM, esta ferramenta não poderá ser utilizada no sistema JoiN pois:

   - não pode ser implementada como um serviço do sistema JoiN: não é implementada em Java e realiza comunicação direta entre os agentes que a compõem, por meio do protocolo SOAP, o que não é permitido na arquitetura de serviços e componentes do JoiN;

- requer a instalação de software em cada computador pois alguns componentes são necessários para a criação, migração e execução dos agentes móveis e também para interface entre os agentes e o computador. Coom vimos anteriormente, esta condição contraria a característica de portabilidade do sistema JoiN, pois requer que seus colaboradores instalem ou configurem *software* adicional.

9. EDG Logging and Bookkeeping [88]

   O sistema EDG Logging and Bookkeeping tem características que inviabilizam sua utilização no sistema JoiN, tais como:

   - não oferecer suporte à linguagem Java em sua API, o que dificulta a integração como um serviço do sistema JoiN;
   - necessitar dos componentes do Workload Management System (WMS) para geração de eventos;
   - ser organizado como *peer-to-peer* o que é inadequado para a utilização no sistema JoiN, pois não atende as restrições de comunicação entre serviços no sistema JoiN.

10. Monitoring and Discovery System [14]

    Os serviços MDS2 e MDS3 implementam, além do serviço de monitoramento, o serviço de diretório utilizado pelo Globus para descoberta de recursos. As funções implementadas nestes serviços atendem características da arquitetura Globus, que diferem do sistema JoiN, pois:

    - não são implementados em Java;
    - apesar do suporte a aplicações desenvolvidas em Java, sua adaptação à linguagem de descrição de aplicações do JoiN (PASL) não é direta nem trivial;
    - requerem diversos outros componentes da implementação Globus Toolkit, para geração, transferência e tratamento de eventos.

11. NetLogger [72]

    Como já apresentado nas descrições das ferramentas pyGMA e JAMM, a ferramenta NetLogger também não é adequada para utilização no ambiente JoiN pois:

    - para o monitoramento de aplicações utilizando o NetLogger é necessário que os desenvolvedores das aplicações codifiquem chamadas à sua API, em todos os pontos críticos de execução;
    - o monitoramento dos recursos, como computadores e redes, utiliza recursos do sistema Linux, e o JoiN deve ser independente de plataforma;
    - o NetLogger requer uma porta específica para comunicação entre os produtores e o consumidor (um *daemon* servidor que recebe os eventos e grava em disco). Isto não atende as restrições de comunicação entre serviços no sistema JoiN.

3.4.1 Resultado da avaliação

Avaliando as principais ferramentas analisadas neste trabalho, com relação às suas características e às dificuldades de serem portadas para o sistema JoiN, em função dos critérios descritos na seção 3.3.1 e das restrições encontradas para sua adaptação ao sistema, conclui-se que não existe

atualmente na literatura uma ferramenta que possa ser implementada diretamente como um serviço de monitoramento no sistema JoiN.

A arquitetura GMA (*Grid Monitoring Architecture*), proposta pelo grupo de trabalho GGF-Perf (*Performance Working Group*) e apresentada na seção 2.4, reúne as principais características necessárias para monitoramento de recursos do sistema JoiN, pois:

- na GMA são definidos apenas três componentes: produtores, consumidores e serviços de diretório. Estes componentes podem ser implementados como um serviço do sistema JoiN e executado, conforme a hierarquia da arquitetura JoiN, nos componentes *worker*, *coordinator* e *server*;
- o monitoramento é baseado em eventos, e os modelos de publicação/assinatura e solicitação/resposta suportados são adequados às necessidades do JoiN;
- os dados podem ser transferidos diretamente de um produtor para um consumidor, e isto pode ser feito por meio do serviço de comunicação do sistema JoiN;
- o monitoramento pode feito por meio de sensores distribuídos, que captam as informações necessárias, centralizando os dados em um serviço de diretório para busca posterior.

A complexidade do monitoramento no sistema JoiN é conseqüência da heterogeneidade e da distância dos computadores que o compõem, do dinamismo causado pela adesão e desligamento não previsto de colaboradores, e da escalabilidade necessária para a manutenção da qualidade do sistema. Assim, parece ser mais adequado que o monitoramento deste sistema seja feito por uma ferramenta projetada e desenvolvida exclusivamente para ele, e que satisfaça as necessidades dos usuários, administradores e desenvolvedores de aplicações e serviços.

O projeto de uma ferramenta específica para o sistema JoiN é apresentado nas seções seguintes, com base nas informações obtidas durante a análise e avaliação das ferramentas apresentadas e procurando aproveitar os melhores pontos de cada uma delas, em especial a forma como são tratados os problemas relacionados a escalabilidade, dinamismo e heterogeneidade. Entre estes pontos destacam-se os seguintes:

- no sistema Ganglia há uma hierarquia e uma distribuição de componentes, além da utilização de uma interface *web* para apresentação das informações;
- no Nagios há a possibilidade de notificações assíncronas, que identificam a ocorrência de problemas;
- no MonALISA tem-se a ação dos coletores, que combinam e agrupam as informações coletadas;
- nas ferramentas que implementam a GMA, como R-GMA, pyGMa e MA-GMA, tem-se a eficiência e a escalabilidade obtidas com o uso do modelo produtor-consumidor;
- o software JAMM distribui o gerenciamento do grande volume de dados por meio da utilização de agentes, que podemos conseguir por meio da distribuição da função de sensor/produtor nos componentes *worker* do sistema JoiN;
- no NetLogger existe a manutenção de dados históricos utilizados para avaliação do desempenho dos sistemas monitorados.

3.5 JoiNMon - Uma ferramenta de monitoramento para o sistema JoiN

Nesta seção são apresentados o projeto e a implementação da ferramenta JoiNMon para monitoramento do sistema JoiN. Esta ferramenta vem suprir a necessidade de acesso a informações sobre o sistema JoiN, possibilitando a observação em tempo real do estado do sistema e das aplicações, bem como a exibição de informações coletadas ao longo do tempo e armazenadas em uma base de dados.

Diante dos conhecimentos obtidos nos estudos sobre monitoramento, na análise das principais ferramentas para monitoramento de ambientes de grade computacional ou baseados na Internet e na identificação das características desejadas para este tipo de ferramenta, procuramos na realização deste projeto corrigir as eventuais falhas e integrar as boas idéias de maneira que sejam adequadas às restrições e características do sistema JoiN.

As soluções adotadas para atender as necessidades de um ambiente heterogêneo e dinâmico como o sistema JoiN têm como principais objetivos manter a escalabilidade do sistema e prover informações claras e atualizadas sobre aplicações, usuários, colaboradores e o estado do sistema de uma forma geral.

3.5.1 Objetivos

Como descrito nas seções anteriores, foi identificada a necessidade de desenvolvimento de uma ferramenta de monitoramento específica para o sistema JoiN pois, devido a aspectos de sua arquitetura, as ferramentas existentes não podem ser facilmente integradas ou adaptadas a este sistema. Sendo assim, foi desenvolvida a ferramenta JoiNMon, baseada na arquitetura GMA apresentada na seção 2.4, para monitoramento do sistema JoiN.

A padronização proposta na arquitetura GMA consiste de um modelo produtor-consumidor que pode ser adaptado à arquitetura do sistema JoiN e, como não impõe protocolos ou modelo de dados a serem utilizados, é também possível adequar a ferramenta de monitoramento às características exigidas dos serviços do sistema JoiN. Assim, no desenvolvimento da ferramenta de monitoramento JoiNMon pôde ser utilizado o modelo de dados mais adequado às informações que se deseja monitorar e manter, e a comunicação entre os componentes da ferramenta pôde se adequar às restrições impostas pela arquitetura do sistema JoiN.

Esta ferramenta foi desenvolvida com o objetivo de fornecer dados para que os administradores e usuários do sistema JoiN possam gerenciar de forma eficiente os recursos disponíveis e a execução de aplicações neste sistema, possibilitando o acompanhamento da utilização e do desempenho de cada componente e de cada aplicação presentes no sistema JoiN, por meio da obtenção das informações relevantes e da apresentação das mesmas de forma adequada.

O monitoramento da execução das aplicações e dos serviços implementados no sistema oferece subsídios para a descoberta de possíveis problemas e gargalos, fornecendo informações sobre os componentes do sistema, facilitando a análise do desempenho do sistema, e da eficiência na distribuição e execução das aplicações. Também facilita o acompanhamento da execução das aplicações distribuídas ou paralelas. Além disto, as informações poderão no futuro ser utilizadas por outros serviços do sistema, como por exemplo, por um escalonador de processos e balanceador de carga dinâmicos.

Assim, por meio das informações apresentadas, deve ser possível:

- identificar mais rapidamente a existência de problemas ou gargalos;
- identificar os recursos disponíveis para a execução das aplicações;
- identificar parâmetros importantes para o escalonamento de processos e balanceamento de carga;
- acompanhar a execução de aplicações com mais facilidade;
- identificar e acompanhar a utilização do sistema pelos seus usuários;
- identificar e acompanhar a utilização de cada um dos computadores que foram disponibilizados para uso pelo sistema;
- obter informações sobre aplicações já encerradas.

O escalonamento é um processo de tomada de decisões, responsável pela otimização na alocação de recursos e na distribuição de processamento entre os computadores integrantes do sistema, com o objetivo de maximizar o desempenho por meio do balanceamento de carga entre estes computadores. O uso das informações obtidas pela ferramenta de monitoramento, que permitem avaliar o desempenho de cada um dos computadores, é muito importante para que se alcance melhores resultados nas decisões de escalonamento em sistemas heterogêneos como o JoiN, onde o problema de balanceamento de carga é mais complexo.

O processo de escalonamento é um dos possíveis observadores das informações coletadas pelas ferramentas de monitoramento. Além dele, os usuários e os administradores do sistema JoiN têm interesse nestas informações e, por este motivo, a forma de apresentação das informações recebeu atenção especial: deve facilitar a análise das informações, tanto pelos administradores do sistema como pelos seus usuários, para que estes possam compreender os dados apresentados.

3.5.2 Visão Geral

Os principais desafios encontrados no projeto da ferramenta JoiNMon foram relativos à obtenção, tratamento e disponibilização das informações pois, como o volume de dados pode ser muito grande, a eficiência da ferramenta está diretamente relacionada ao tratamento destes dados.

O projeto e desenvolvimento foi composto de três etapas principais: planejamento, implementação e avaliação de resultados. A seguir, são descritos os principais fatores considerados em cada uma destas etapas.

Planejamento

A etapa de planejamento é muito importante para o sucesso da especificação e desenvolvimento desta ferramenta, devido à complexidade da coleta de dados. Os passos seguidos no planejamento foram:

1. identificação das informações relevantes para o monitoramento;
2. avaliação da melhor forma de obtenção e transmissão destas informações;
3. identificação de como deve ser feita a manutenção de informações que reflitam o estado atual do sistema e aplicações em execução, bem como informações sobre estados e aplicações passados;
4. definição do armazenamento destas informações de forma eficiente para facilitar sua recuperação;

5. definição de uma interface gráfica onde possam ser obtidas facilmente as informações desejadas.

Implementação

As informações obtidas, para cada computador conectado ao sistema JoiN e para cada aplicação em execução no sistema, são processadas pela ferramenta de monitoramento JoiNMon e exibidas por meio de uma interface gráfica, responsável pela interação entre o serviço de coleta de dados e os usuários do sistema, obtendo e apresentando adequadamente as informações solicitadas. As informações coletadas são também gravadas em um banco de dados relacional, para que possam ser recuperadas e analisadas posteriormente.

A ferramenta de monitoramento foi desenvolvida em dois módulos: o primeiro módulo efetua a coleta, o tratamento e o armazenamento das informações, e foi implentado como um serviço do sistema JoiN; o segundo módulo implementa uma interface gráfica, que possibilita a análise das informações previamente armazenadas, e foi implementado como uma aplicação *web*. Desta forma, a ferramenta permite o monitoramento dos componentes e aplicações do sistema JoiN.

Na implementação do módulo para coleta, foram observadas as condições estabelecidas para a implementação de serviços no sistema JoiN, descritas na seção 3.2 seguindo definições da arquitetura GMA, descritas na seção 2.4, referentes à utilização de eventos, produtores e consumidores.

Avaliação

Na avaliação da ferramenta de monitoramento foram considerados o seu impacto sobre o desempenho do sistema JoiN e as facilidades que oferece para usuários e colaboradores do sistema. Esta avaliação é apresentada no capítulo 4.

3.6 JoiNMon - Planejamento

Os usuários da ferramenta, as informações e a forma de apresentá-las a estes usuários foram o foco desta etapa de planejamento. Também foram consideradas a arquitetura, as características e as restrições do sistema JoiN, bem como o modelo produtor-consumidor proposto pela arquitetura GMA, baseado na produção e tratamento de eventos.

3.6.1 Usuários do sistema JoiN

Com o objetivo de apresentar as informações de monitoramento de forma a satisfazer quem as busca, começamos identificando as características de cada um dos indivíduos que devem fazer uso da ferramenta de monitoramento. A classificação abaixo está de acordo com a proposta feita pelo grupo de estudos que tem a cargo o controle de acesso e a segurança do sistema JoiN. Este grupo desenvolve a Ferramenta de Segurança do sistema que é responsável pelas identificações, autenticações e autorizações necessárias para segurança do JoiN.

- Usuários:

são os indivíduos que acessam o sistema JoiN para a execução de suas aplicações. Estes indivíduos devem ter sido cadastrados no sistema pela Ferramenta de Segurança e obtido uma identificação única e senha.

- Administradores:

  são os indivíduos responsáveis pela administração do sistema JoiN. O administrador é responsável por um conjunto de componentes: *server*, *coordinators*, *workers* e *jacks*. Os administradores também devem ser cadastrados no sistema pela Ferramenta de Segurança e possuir uma identificação única e senha.

- Visitantes:

  são indivíduos que acessam JoiN por meio da Internet com o objetivo de obter informações genéricas sobre o sistema. Acessam o sistema de forma não identificada, ou seja, sem que tenham sido previamente cadastrados e têm acesso restrito às informações.

- Colaboradores:

  são indivíduos que colocam seus computadores à disposição do sistema JoiN, para que os mesmos sejam utilizados para a execução de aplicações. Um colaborador pode ser previamente cadastrado e identificado pelo seu e-mail, ou ainda contribuir para o sistema de forma anônima, o que não requer cadastramento prévio.

3.6.2 Usuários da ferramenta JoiNMon

A seguir foram estabelecidas associações entre cada um destes diferentes indivíduos, usuários do sistema JoiN, e o que podem buscar na ferramenta JoiNMon.

- Usuário:

  1. consultar o estado de suas aplicações;
  2. consultar informações sobre as suas aplicações em execução, lotes e tarefas que as compõem;
  3. consultar o estado do sistema JoiN e de seus componentes;
  4. consultar o histórico de suas aplicações já encerradas.

- Administrador:

  1. consultar o estado das aplicações;
  2. consultar informações sobre as aplicações em execução, lotes e tarefas que as compõem;
  3. consultar o estado do sistema JoiN e de seus componentes;
  4. consultar o histórico de aplicações já encerradas;
  5. consultar informações sobre os colaboradores e o computadores que foram colocados a disposição do sistema JoiN;
  6. consultar informações sobre os usuários e suas aplicações.

- Visitante:

  1. consultar informações genéricas (estatísticas) sobre as aplicações em execução e encerradas;

2. consultar informações genéricas (estatísticas) sobre o estado do sistema JoiN e de seus componentes.

- Colaborador:

1. consultar informações sobre os computadores que colocou a disposição do sistema JoiN;
2. consultar o estado do sistema JoiN e de seus componentes.

3.6.3 Informações coletadas

Com base nas necessidades destes usuários e nas características do sistema JoiN, foi feito um levantamento das informações necessárias para que estes indivíduos consigam, por meio da ferramenta de monitoramento, obter os dados sobre o sistema, seus colaboradores e aplicações. Para isto devem ser obtidas informações sobre as aplicações executadas no sistema JoiN e sobre os computadores conectados a este sistema. Devido a características de JoiN, e também porque este é executado em uma máquina virtual Java, não é possível obter informações sobre o uso de cpu, de memória e de rede de cada uma das aplicações executadas neste sistema, uma vez que as informações disponíveis por meio das APIs padrão do Java, como, por exemplo, aquelas que podem ser obtidas por meio do pacote *java.lang.management* (introduzido no Java 1.5 para monitoramento da JVM), são referentes à maquina virtual.

As informações sobre aplicações devem ser baseadas na estrutura das mesmas. Como descrito na seção 3.2.2 e representado na Fig. 3.3, uma aplicação do JoiN é formada por:

1. um lote inicial com uma tarefa (lote 1), responsável por organizar os dados que servirão de entrada para o próximo lote;
2. um ou mais lotes de tarefas com várias tarefas paralelas (lote 2 a lote n-1);
3. um lote final com uma tarefa (lote n) responsável pela consolidação dos resultados.

Desta forma, as informações coletadas sobre as aplicações devem conter os itens seguintes.

- Aplicação (estado de cada aplicação):
  - identificação da aplicação;
  - usuário responsável (dono);
  - data/horário da instalação da aplicação;
  - data/horário da submissão da aplicação;
  - data/horário do início do processamento da aplicação;
  - data/horário do término do processamento da aplicação;
  - número de lotes.
- Lotes (estado de cada lote):
  - data/horário do início do processamento do lote;
  - data/horário do término do processamento do lote;
  - número de tarefas que compõem o lote;
  - tarefas deste lote já executadas;

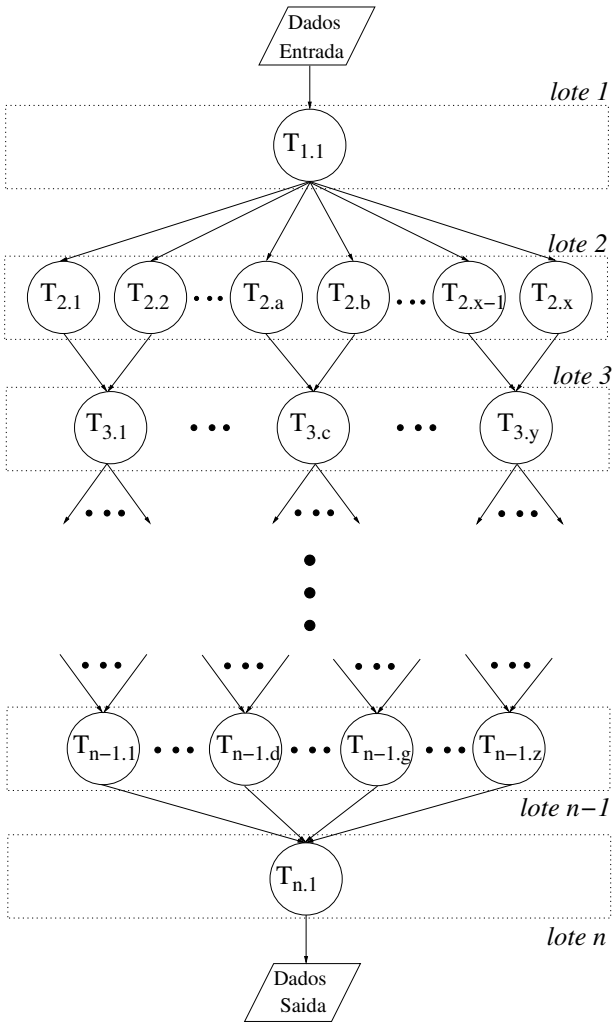

Fig. 3.3: Exemplo da estrutura de uma aplicação JoiN

  - tarefas deste lote em execução;
  - tarefas deste lote em espera;
  - tarefas que estão replicadas e grau de replicação.

- Tarefas (estado de cada tarefa):

  - estado: em execução, encerrada, em espera, réplicas;
  - data/horário do envio da tarefa para processamento;
  - data/horário do retorno (término) da tarefa;
  - computador onde a tarefa está sendo executada (se houver réplicas, deverá ser apresentada uma lista de computadores).

As informações sobre os computadores conectados ao sistema devem permitir a identificação do computador e seu estado.

- Computadores conectados:

  - identificação do computador: definida por sua conexão à Internet (endereço IP ou nome/DNS);
  - informações do sistema operacional em uso no computador;
  - informações da JVM em uso no computador;
  - índice de desempenho associado ao computador (obtido via *benchmark* feito após conexão inicial);
  - estado do computador: processando aplicações ou inativo;
  - relação das aplicações em execução neste computador.

3.6.4 Obtenção e transmissão das informações

Os sensores, componentes responsáveis pela coleta de dados, devem ter fácil acesso às informações de interesse para o monitoramento. No sistema JoiN estas informações são mantidas pelos serviços, ou seja, os dados estão disponíveis em estruturas internas destes componentes. Desta forma, as informações devem ser obtidas por meio de interações com sub-módulos dos serviços que as mantêm e gerenciam. Por exemplo, para obter as informações sobre as aplicações, lotes e tarefas, que são mantidas pelo serviço *Application Manager* a ferramenta de monitoramento deverá interagir com este serviço.

As opções existentes para a ferramenta de monitoramento obter as informações são:

1. análise das informações que trafegam por *proxies* existentes no sistema JoiN, como apresentado em 3.2.1 e ilustrado na Fig. 3.2, baseando-se nos métodos que estes chamam:

   esta opção tem como vantagem a independência e a transparência com relação aos demais serviços, pois os serviços não precisariam de implementações específicas para obter e transmitir as informações de monitoramento; a desvantagem é a possibilidade de existência de mudanças de estado que não estejam associadas à chamada de métodos e, desta forma, não seria possível obter estas informações;

2. modificação de cada um dos serviços que mantêm as informações, de forma que estes serviços implementem os sensores:

   esta opção, tem como desvantagem a necessidade de mudanças em alguns dos serviços existentes, para que sejam adequados ao novo serviço de coleta de informações; entretanto tem a vantagem de possibilitar a obtenção de informações de forma mais completa;
3. monitoramento indireto, por meio da análise de mensagens gravadas nos *logs* dos sistema:

   as informações podem não estar disponíveis nos arquivos de *log*, pois podem existir mudanças de estado que não sejam registradas nestes arquivos, e desta forma o monitoramento pode ficar incompleto. Tem ainda como desvantagem a necessidade de análise de um volume muito grande de mensagens, não padronizadas, para obtenção dos dados disponíveis. Em contrapartida, não requer modificações nos serviços já implementados no sistema possibilitando independência e transparência com relação aos demais serviços.

Avaliando as vantagens e desvantagens de cada uma das opções e tendo o objetivo de obter e monitorar todas as informações desejadas, concluiu-se que os sensores devem ser implementados em cada serviço onde exista uma informação a ser coletada, por meio da alteração do código destes serviços. Além disto, a existência do serviço *EventManager* que provê parte das funções necessárias para o gerenciamento de eventos, e do serviço *Communicator* que é responsável pela comunicação entre componentes do sistema e deverá ser utilizado para transmissão dos eventos, deverá facilitar a implementação. Esta abordagem é possível pois temos acesso ao código fonte do sistema e assim podemos obter as informações de forma mais completa.

Para a transmissão das informações serão utilizados eventos, seguindo o modelo produtor/consumidor da proposta da GMA. Estes eventos são gerados pelos sensores/produtores e consumidos pela ferramenta de monitoramento. Como um serviço do sistema JoiN, esta ferramenta deve atender as restrições da arquitetura deste sistema, descritas na seção 3.2.1, onde não são permitidas comunicações diretas entre componentes *worker*. As comunicações entre um componente *coordinator* e os componentes *worker* a ele associados, bem como a comunicação entre sub-módulos de serviços diferentes em execução no mesmo componente, devem ser feitas por meio de um serviço específico (*Communicator*), já implementado e utilizado pelos demais serviços do sistema JoiN,

Gerenciamento de eventos

Os eventos produzidos devem ser encaminhados aos seus consumidores. Conforme a arquitetura proposta pela GMA o relacionamento entre eventos e seus consumidores é mantido em um serviço de diretório. É portanto necessária a existência de um gerenciador de eventos, que administre os relacionamentos entre produtores, eventos e consumidores, e que também faça o encaminhamento dos eventos ocorridos aos seus consumidores.

No sistema JoiN, este gerenciador, assim como a ferramenta de monitoramento, devem ser implementados como serviços. Na proposta da GMA, o serviço de registro é utilizado para identificar os eventos, produtores e consumidores disponíveis: uma vez que um consumidor identifique o produtor do evento em que está interessado (informação obtida no registro), as informações (eventos) são recebidas diretamente do produtor. Entretanto, um serviço de registro como o proposto pela GMA não é adequado às características do sistema JoiN, pois a comunicação direta entre dois serviços contrapõe-se à arquitetura deste sistema. Desta forma, o serviço *EventManager* já existente é utilizado como

intermediário e, como descrito na seção 2.4.3 e apresentado na Fig. 3.4, ele faz o registro dos eventos, de seus consumidores e também a transmissão de eventos entre produtores e consumidores. Assim, os produtores sinalizam as ocorrências de eventos ao gerenciador de eventos (serviço *EventManager*), que os encaminha ao consumidor (serviço *JoiNMonitor*).

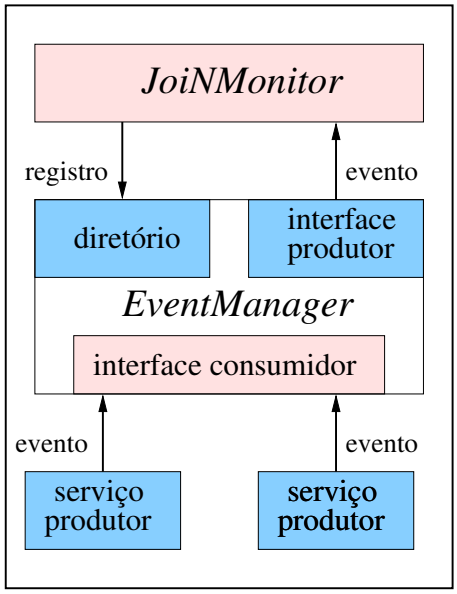

Fig. 3.4: JoiNMon: registro, produtores e consumidores de eventos

Os eventos devem ser tratados de forma assíncrona para que os serviços que os produzem não tenham que aguardar a execução de seus consumidores e, desta forma, não fiquem bloqueados.

3.6.5 Armazenamento das informações

Em ambientes de processamento que utilizam recursos ociosos de computadores conectados a Internet, o número de colaboradores em potencial é muito grande: em julho de 2008 o projeto SETI@Home contabiliza [89, 90] 1.998.709 computadores cadastrados e 846.661 colaboradores de 252 países, sendo 353.139 computadores e 214.059 colaboradores ativos, atingindo a marca de 54.158,27 GigaFLOPS, ou 54,158 TeraFLOPS (FLoating point Operations Per Second). Também no caso do sistema JoiN, o volume de informações a serem coletadas, que contêm dados que permitem acompanhar o estado das aplicações executadas e dos computadores conectados ao sistema, pode ser muito grande e, por esta razão, seu armazenameto deve ser feito de forma eficiente.

Entre as formas de armazenamento possíveis, foram avaliadas as opções de armazenamento em arquivos simples (*flat files*) e a utilização de um banco de dados relacional. O uso de arquivos tem como vantagens não requerer *software* adicional e proporcionar velocidade na gravação de dados; como desvantagens apresentam a dificuldade e a lentidão na recuperação das informações para permitir sua exibição para os usuários, pois estes arquivos têm que ser lidos seqüencialmente até que se encontre as informações desejadas. Como no caso do uso de arquivos de *log* para obtenção de informações para monitoramento, a eficiência da gravação não compensa a dificuldade de obtenção de informações. Desta forma a utilização de banco de dados relacional se mostrou mais indicada

para este caso, facilitando o acesso aos dados e possibilitando diversas abordagens no tratamento dos mesmos, beneficiando também as consultas, apesar da necessidade de instalação de um *software* para gerenciamento da base de dados.

Esta necessidade não é um problema de difícil solução, pois existem muitos *softwares* para servidores de bancos de dados relacionais. Existem servidores comerciais, proprietários e reconhecidos no mercado, como: Microsoft SQL Server [91], DB2 [92], Oracle [93] e Informix [94]. Também existem opções de código aberto, muito utilizadas atualmente, como MySQL [95] e PostgreSQL [59]. Existe ainda HSQLDB [96], um banco de dados em código aberto muito utilizado por ferramentas simples desenvolvidas em Java.

Optou-se pela utilização do HSQLDB, por ser de fácil instalação e adequado ao ambiente de testes, favorecendo o desenvolvimento e, principalmente, a avaliação da ferramenta proposta. A substituição deste servidor de banco de dados por outro servidor mais robusto, que ofereça melhor desempenho pode ser feita de forma simples e rápida, alterando-se apenas a conexão JDBC [97], já que o mesmo segue a sintaxe padrão SQL.

Para que se obtenha a flexibilidade de um banco de dados relacional é essencial o uso de uma estrutura de dados normalizada, ou seja, estruturas de dados não redundantes que representem as entidades e os relacionamentos descritos pelos dados. A modelagem da base de dados é apresentada no Apêndice B e foi feita com as seguintes premissas e requisitos:

- ser simples e de fácil compreensão;
- ser flexível e suportar modificações estruturais para inserção de novos dados, sem que a estrutura existente seja comprometida;
- refletir a visão que o usuário tem dos dados e suas medidas;
- suportar a grande volume de dados;
- manter dados históricos que reflitam situações ao longo do tempo;
- manter a integridade dos dados de forma eficiente;
- o acesso aos dados deve ser feito com bom desempenho, tanto para atualização quanto para recuperação das informações.

Manutenção de informações atuais e históricas

A ferramenta de monitoramento tem como objetivo, além de permitir o acompanhamento do estado do sistema JoiN em tempo real, possibilitar também a observação de estados passados. Isto é possível pela manutenção na base de dados de informações relativas à execução de aplicações já encerradas e de todas as conexões e utilização de computadores colaboradores.

3.6.6 Interface gráfica

O planejamento de interfaces gráficas é uma tarefa bastante complexa e objeto de estudo na área Interação Humano-Computador (IHC), que envolve profissionais das áreas de computação, psicologia, ergonomia e *design* gráfico. Com o uso cada vez maior de computadores no cotidiano das pessoas, desenvolvem-se na área de IHC diversas teorias, métodos, técnicas e ferramentas com o objetivo de obter produtos de qualidade, como bem descreve o livro *Contextual Design* [98].

Não é objetivo deste trabalho explorar aspectos teóricos de IHC mas sim desenvolver uma interface gráfica simples, intuitiva e de fácil uso. O usuário moderno é capaz de interagir com as interfaces gráficas de forma cognitiva, utilizando recursos mnemônicos, e fazer associações a aspectos culturais e empíricos. Portanto o usuário deve ser o foco principal do desenvolvimento da interface gráfica da ferramenta.

Esta interface pode ser implementada como uma aplicação gráfica, que o usuário instala em seu computador ou executa remotamente. A instalação de um *software* adicional pelo usuário não é interessante pois requer ações para instalação e atualizações. A execução remota, por sua vez, também é comprometida pelo alto tráfego de rede que aplicações gráficas executadas remotamente geram. A implementação de uma interface gráfica baseada em *web* apresenta-se como uma boa solução, pois não requer instalação de *software* adicional no equipamento do usuário: a obtenção de informações é feita por meio de *browsers*, que hoje em dia integram o *software* básico de qualquer computador, com a vantagem de permitir o acesso às informações a partir de qualquer computador conectado a Internet.

Foram analisadas algumas das tecnologias existentes para a criação de páginas dinâmicas em aplicações *web*: Java Server Pages (JSP), Microsoft Active Server Pages (ASP) e Hypertext Preprocessor (PHP). Estas tecnologias apresentam simbologias parecidas e muitas semelhanças, como permitir a separação entre a parte estática e a parte dinâmica. Existem entretanto diferenças que devem ser consideradas na adoção de uma ou outra tecnologia. A tecnologia ASP tem como desvantagens a dependência do sistema operacional Windows e do servidor *web* Internet Information Services (IIS) além de ser um padrão fechado, proprietário da Microsoft e, por estas razões, foi descartada. As tecnologias JSP e PHP são, por sua vez, independentes de plataforma, de sistema operacional e de servidor *web*, e têm código aberto. Enquanto a principal característica do PHP é a facilidade de desenvolvimento devido a uma sintaxe flexível, a tecnologia JSP é uma extensão da plataforma Java e indicada para uma programação robusta, com as vantagens da orientação a objetos.

Devido a estas razões decidiu-se pela implementação da interface *web* em JSP e, para uso desta tecnlogia, é necessária a utilização de um servidor de aplicações Java para *web* (servidor *web* especializado, com suporte para a execução de *servlets*) como, por exemplo, Jakarta Tomcat [99] e Glassfish [100], que são distribuidos como *software* livre. O servidor Jakarta Tomcat foi utilizado por ser robusto, de fácil instalação e configuração.

As características desejadas e avaliadas na interface gráfica são: conteúdo direcionado ao usuário e usabilidade. Desta forma, as informações exibidas pela interface gráfica devem estar organizadas hierarquicamente, conforme a arquitetura do sistema JoiN (componentes *server*, *coordinators* e *workers*), e também conforme as características das aplicações executadas neste sistema (aplicação, lotes, tarefas). As informações são assim organizadas:

- componente *server*:

  identificação, características, horário de início do sistema neste componente, número de componentes *coordinator* ativos, número de componentes *worker* ativos, número de aplicações concluídas, em execução ou em espera;
- componentes *coordinator*:

  identificação, características, horário de início deste componente, número de componentes *worker* ativos e conectados a cada componente *coordinator*, número de aplicações em execução/espera associadas a cada componente *coordinator*;

- componentes *worker*:

  identificação, características, horário de início deste componente, número de aplicações em execução/espera associadas a cada componente *worker*;
- aplicação:

  usuário que a instalou e submeteu, horário da submissão, do início e do término do processamento, número de componentes *worker* utilizados durante o processamento, número de lotes que compõem esta aplicação;
- lotes:

  aplicação a que pertence, horário do início e do término do processamento, número de tarefas que compõem este lote;
- tarefas:

  aplicação e lote a que pertence, horário do início e do término do processamento, componentes *worker* utilizados no processamento;
- usuários:

  identificação, aplicações submetidas, em execução e já executadas, horário do último acesso ao sistema;
- colaboradores:

  identificação, componentes *worker* que controla, total de tempo cedido para o sistema para a execução de aplicações.

O desenho das páginas usadas para apresentação destas informações pela interface gráfica encontra-se no Apêndice C.

3.7 JoiNMon - Implementação

Nesta seção são apresentados, com base no modelo de eventos proposto pela arquitetura GMA (seção 2.4.3) e no planejamento apresentado:

- os eventos produzidos;
- as modificações realizadas nos serviços do sistema JoiN para a produção destes eventos;
- a adequação do serviço *EventManager* a esta arquitetura ;
- a implementação do novo serviço *JoiNMonitor*.

O serviço *EventManager* é responsável pelo registro dos eventos e pelo encaminhamento dos mesmos ao módulo consumidor, quando gerados por seus produtores; o serviço *JoiNMonitor* implementa o módulo consumidor, responsável pela coleta, tratamento e armazenamento das informações a serem monitoradas.

Pode-se apresentar, de forma resumida, a composição da ferramenta de monitoramento JoiNMon como:

- produtores de eventos (diversos serviços do sistema JoiN);

- serviço para gerenciamento dos eventos (*EventManager*);
- serviço para consumo dos eventos (*JoiNMonitor*);
- interface gráfica (aplicação web *JoiNMonitor*).

3.7.1 Produtores de eventos

Os produtores obtêm as informações a serem monitoradas, formatam, padronizam e geram os eventos. Cada evento contém informações que permitem ao consumidor reações baseadas no contexto em que o mesmo ocorreu. Todos os serviços do sistema JoiN que mantêm informações a serem coletadas e tratadas pelo sistema de monitoramento foram alterados de forma a gerar os eventos. Na Tab. 3.1 são apresentados os eventos e os seus respectivos produtores.

Tab. 3.1: Eventos implementados e serviços produtores

Identificação	Descrição	Produtor	Componente
JOIN_START	Sistema JoiN iniciado	*S_JoinMonitor*	*server*
JOIN_SHUTDOWN	Sistema JoiN finalizado	*S_JoinMonitor*	*server*
NEW_COORD	Conexão de *coordinator*	*C_JoiNMonitor*	*coordinator*
COORD_DOWN	Desconexão de *coordinator*	*S_Communicator*	*server*
NEW_WORKER	Conexão de *worker*	*W_JoiNMonitor*	*worker*
WORKER_PERF	Fator de desempenho de *worker*	*TaskSchedulerer*	*coordinator*
WORKER_DOWN	Desconexão de *worker*	*C_Communicator*	*coordinator*
APP_SUB	Submissão de aplicação	*S_ApplicationManager*	*server*
APP_START	Início processamento aplicação	*ExecutionEngine*	*coordinator*
APP_END	Término processamento aplicação	*C_ApplicationManager*	*coordinator*
APP_KILL	Cancelamento aplicação	*S_ApplicationManager*	*server*
TSK_START	Início processamento tarefa	*W_ApplicationManager*	*worker*
TSK_END	Término processamento tarefa	*W_ApplicationManager*	*worker*

3.7.2 Serviço *EventManager*

Na programação em Java é geralmente usado o modelo de eventos do Java AWT [101] onde, além da interação entre produtores e consumidores, a geração e consumo dos eventos é feita de forma sincronizada: a mesma *thread* que gera o evento executa o seu *handler*. Este modelo não é adequado ao sistema JoiN devido à necessidade de interação entre serviços e componentes, o que contraria a arquitetura do sistema, e também devido à possibilidade de interferência no processamento de diferentes serviços ou no núcleo do sistema, por problemas no tratamento dos eventos. Para adequar a produção e tratamento de eventos à arquitetura do sistema JoiN e ao modelo proposto pela GMA é utilizado o serviço *EventManager*. Apesar deste serviço já existir no sistema JoiN, o mesmo precisou de algumas adaptações para desempenhar melhor sua função e para adequação à arquitetura GMA.

O serviço *EventManager* tem a função de atuar como intermediário entre produtores e consumidores de eventos pois, no modelo implementado, os produtores e consumidores não interagem diretamente. As principais razões para isto são a adequação ao modelo proposto pela GMA e a possibilidade de introduzir modificações nos produtores de eventos sem que seja necessário alterar também os consumidores. Além disto, a produção e consumo de eventos é feita de forma assíncrona e não bloqueante, ou seja, os produtores de eventos não precisam aguardar o processamento de todos os consumidores de um evento gerado. O consumo de um evento é feito por uma *thread* distinta daquela em que foi produzido: a geração de um evento corresponde à sua inclusão em uma fila e uma *thread* especializada obtém os eventos incluídos nesta fila e executa o *handler* de forma assíncrona e independente.

O serviço *EventManager* é executado em cada um dos computadores participantes do sistema JoiN, e é responsável também pela adequação da arquitetura GMA ao sistema JoiN, onde não é possível a comunicação direta entre dois serviços.

Ele mantém em uma estrutura de dados os relacionamentos entre produtores, eventos e consumidores e em uma fila os eventos ocorridos. Quando um consumidor indica seu interesse em um determinado evento, esta informação é armazenada nesta estrutura de dados, indexada pela identificação do evento.

A geração de um evento por um produtor corresponde à sua inclusão em uma fila, e uma *thread* especializada obtém eventos desta fila e dispara a execução de todos os consumidores que estão associados a este evento na estrutura de dados que mantém estas informações. A produção e o consumo dos eventos são tratados de forma assíncrona e não bloqueante: tanto o serviço que produz um evento não tem que aguardar a execução dos consumidores deste evento, como os consumidores não permanecem bloqueados aguardando a ocorrência de um evento.

As funcionalidades do serviço *EventManager* são implementadas por meio do módulo *EventManager* e podem ser descritas como:

1. registrar o interesse de um consumidor na ocorrência de um determinado evento;
2. notificar cada consumidor previamente registrado sobre a ocorrência de um evento.

Módulo EventManager

O serviço *EventManager* é implementado por meio de um único módulo. Desta forma, em todos estes componentes do sistema JoiN é executado o mesmo código, implementado pelo módulo também chamado *EventManager*, que é responsável pelo registro de consumidores dos eventos e pela notificação a estes consumidores sobre a ocorrência de um evento. Neste módulo *EventManager* são implementados os métodos: *registerMyListener* (para registro de consumidores), *FireEvent* (para produção) e *handleEvent* (para consumo).

Por ser o intermediário entre consumidores e produtores, o serviço *EventManager* interage com estes da seguinte forma:

- Consumidores:
  - os serviços consumidores de eventos do sistema JoiN devem implementar a interface *Listener*, que permite o recebimento da notificação da ocorrência de eventos. Na inicialização

do sistema em cada um dos seus componentes (*server, coordinator* ou *worker*), é criada uma *thread EventExecutor* para cada consumidor;
  - os consumidores devem ainda registrar seu interesse na ocorrência de cada evento, e isto é feito por meio da função *registerMyListener*: este registro é mantido em uma estrutura de dados, indexada por *eventType*, onde são mantidas as informações sobre todos os consumidores de cada evento;
  - a *thread* que executa *EventExecutor* manipula todos os eventos nos quais este consumidor registra interesse e, ao identificar a ocorrência de um evento (por meio da inspeção de uma fila de eventos) executa a função *handleEvent*, onde os eventos são consumidos;

- Produtores:

  a geração de um evento é feita por meio da execução da função *fireEvent* que consiste:

  - na verificação da permissão deste serviço para gerar eventos (isto é feito pelo serviço *SecurityManager* do sistema JoiN);
  - na colocação do evento na fila de eventos de cada *EventExecutor* associado a este evento (identificados pela tabela onde estão registrados os consumidores de cada evento).

3.7.3 Serviço JoiNMonitor

Recebe notificações sobre eventos, gerados pelos produtores, e reage a estes eventos.

O serviço JoiNMonitor é o consumidor dos eventos que contêm informações selecionadas para monitoramento, e foi implementado em dois módulos distintos, como pode ser observado no esquema apresentado na Fig. 3.5:

- módulo coletor: para coleta de informações em cada uma das máquinas participantes do sistema, executado em todas as máquinas do sistema (*server, coodinators* e *workers*) e responsável pelo tratamento dos eventos gerados pelos demais serviços;
- módulo agrupador: para agrupar as informações obtidas, seguindo a hierarquia de componentes do sistema JoiN, executado nos componentes *server* e *coodinator*. No componente *server* este módulo faz o arquivamento das informações, utilizando para isto a base de dados apresentada na seção 3.6.5 e no Apêndice B. No componente *coordinator* este módulo executa a transferência de informações entre os componentes *coodinator* e *server*. Esta transferência é feita de duas formas:

  - quando o *server* solicitar o envio (sob demanda);
  - quando o *coordinator* disparar o envio devido ao buffer de armazenamento ter atingido um volume previamente determinado.

Desta forma, a transferência de informações entre os módulos coletores e o módulo agrupador correspondente é feita usando o mecanismo *push*. A transferência de informações entre os módulos agrupadores dos componentes *coordinator* e *server* pode ser feita tanto por meio do mecanismo *pull* (sob demanda) quanto utilizando o mecanismo *push* (quando o *buffer* atingir o volume pré-determinado).

As funcionalidades implementadas pelo serviço *JoiNMonitor* podem ser descritas como:

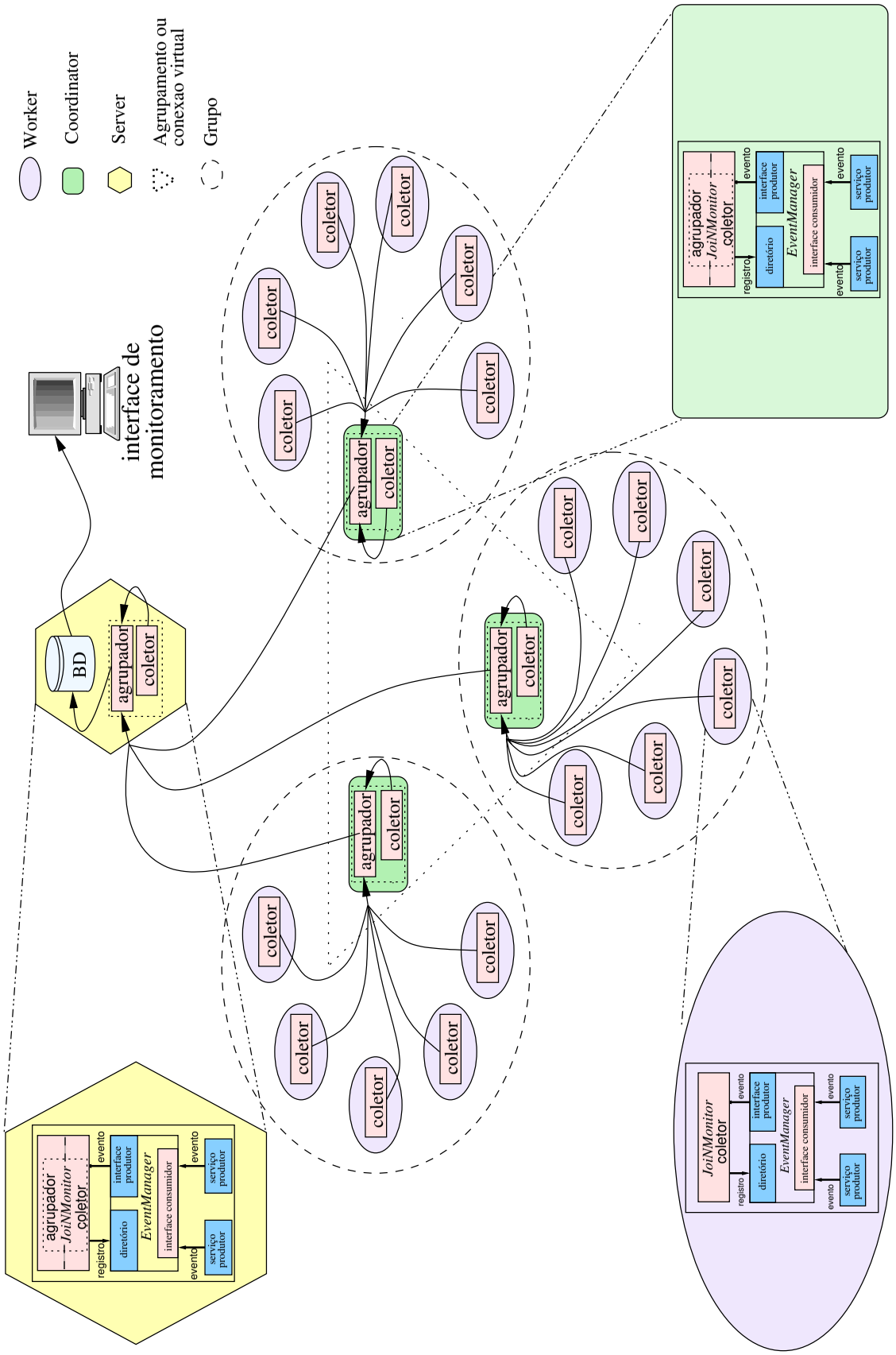

Fig. 3.5: *JoiNMonitor* - arquitetura e hierarquia

- componentes *worker*:

    1. na inicialização do serviço, obter as informações sobre o computador onde está sendo executado e enviá-las para o serviço JoiNMonitor do componente *coordinator* a que está subordinado;
    2. enviar eventos recebidos dos produtores do componente *worker* para o componente *coordinator*;

- componentes *coordinator*:

    1. na inicialização do serviço, obter as informações sobre o computador onde está sendo executado e enviá-las para o serviço JoiNMonitor do componente *server* a que está subordinado;
    2. receber e armazenar eventos enviados pelo serviço JoiNMonitor dos componentes *worker* subordinados a este componente *coordinator*;
    3. receber e armazenar eventos enviados pelos produtores deste componente *coordinator*;
    4. enviar eventos armazenados para o componente *server*;

- componentes *server*:

    1. na inicialização do serviço, obter as informações sobre o computador onde está sendo executado e armazená-las;
    2. requisitar o envio de eventos armazenados no serviço JoiNMonitor *coordinator*;
    3. receber e armazenar eventos enviados pelo serviço JoiNMonitor dos componentes *coordinator* subordinados a este *server*;
    4. receber e armazenar eventos enviados pelos produtores deste componente *server*.

Estas funcionalidades são implementadas de forma diferenciada para cada tipo de componente onde são processadas por meio dos módulos *S_JoiNMonitor* (componente *server*), *C_JoiNMonitor* (componentes *coordinator*) e *W_JoiNMonitor* (componentes *worker*). A seguir são apresentados, de forma mais detalhada, estes módulos do serviço *JoiNMonitor*, descrevendo a implementação das funcionalidades planejadas.

Módulo S_JoiNMonitor

É o módulo do serviço JoiNMonitor executado no componente *server*, que recebe os eventos gerados pelos demais serviços deste componente e obtém e trata as informações a serem monitoradas. Também é responsável pelo armazenamento das informações no banco de dados, para possibilitar a recuperação destas informações pela interface gráfica.

Este serviço é iniciado na ativação do componente *server* e executa determinadas funções em determinados instantes.

- Na ativação deve:

    – obter parâmetros, por meio da leitura de um arquivo de propriedades;
    – registrar no serviço *EventManager (registerMyListener)* os eventos sobre os quais deverá ser notificado.

- De forma assíncrona, na ocorrência de algum dos eventos para os quais se registrou, será notificado da ocorrência de um evento. Então deve:
  - identificar o evento ocorrido;
  - obter as informações associadas a este evento;
  - tratar, formatar, e guardar as informações obtidas.
- Periodicamente (o período é configurado em um arquivo de propriedades), solicita aos coordenadores (*peers*) as informações sobre os eventos mantidos pelos mesmos. Para isto deve:
  - enviar uma mensagem *JOINMON_REQUEST_COORD_EVENTS*;
  - receber as respostas dos coordenadores, por meio de mensagens *JOINMON_RETURN_COORD_EVENTS*;
  - gravar as informações recebidas dos coordenadores.
- De forma assíncrona, quando a transmissão de informações é disparada pelo componente *coordinator*, deve:
  - gravar as informações recebidas dos coordenadores.

Módulo C_JoiNMonitor

É o módulo da ferramenta de monitoramento executado no componente *coordinator* que envia ao *server*, mediante solicitação ou quando o *buffer* de armazenamento estiver cheio, a lista contendo os eventos ocorridos e registrados, bem como as informações a eles associadas.

Este serviço é iniciado na ativação do componente *coordinator* e executa as funções seguintes.

- Na ativação:
  - registra no serviço *EventManager (registerMyListener)* os eventos sobre os quais deverá ser notificado;
- Em um laço infinito:
  - recebe mensagens por meio do serviço *Communicator*;
  - se for uma mensagem identificada como *JOINMON_REQUEST_COORD_EVENTS*, ou seja, uma mensagem do componente *server* requisitando o envio dos eventos, ou então quando o *buffer* de eventos estiver cheio:
    - envia ao servidor a resposta à solicitação, por meio de uma mensagem *JOINMON_RETURN_COORD_EVENTS*, contendo a lista de eventos que mantém;
    - re-inicia a lista de eventos.
- De forma assíncrona, na ocorrência de algum dos eventos para os quais se registrou:
  - identifica o evento ocorrido;
  - obtém as informações associadas a este evento;
  - trata, formata e guarda as informações obtidas.
- Se o buffer para armazenamento do eventos estiver cheio:
  - envia as informações ao componente *server*.

Módulo W_JoiNMonitor

É o módulo da ferramenta de monitoramento executado no componente *worker* que envia ao *coordinator* as informações associadas a este componente.

Este serviço será iniciado na ativação do componente *worker* e executa a função seguinte.

- Envia uma mensagem ao *coordinator (C_JoiNMonitor) JOINMON_WORKER_INFO*, contendo a identificação do computador onde está sendo processado. Esta informação será armazenada no *coordinator* associada a TID do componente *worker*, para que os eventos gerados por este *worker* possam ser identificados.

3.7.4 Interface Gráfica JoiNMonitor

A interface gráfica implementa as seguintes funcionalidades:

- verificar informações de aplicações:

  exibe informações sobre as aplicações em execução ou já encerradas. No caso das aplicações em execução, são exibidas as informações sobre cada lote e tarefa. No caso de aplicações já encerradas, são exibidas informações que resumem sua execução;
- verificar informações do sistema JoiN e de seus componentes:

  exibe informações sobre o estado dos componentes *server*, *coordinators* e *workers*;
- verificar informações de computadores colaboradores do sistema JoiN:

  exibe informações para os colaboradores do sistema, relativas a utilização de seus computadores pelo sistema.

Como já foi explicado anteriormente, esta interface foi implementada em JSP para permitir a visualização das informações por meio de *browsers*, de forma organizada, e assim facilitar o acesso a estas pelos usuários e administradores do sistema JoiN. Uma página *web* criada com a tecnologia JSP, quando instalada no servidor de aplicações, é transformada em um *servlet*, o que permite a geração de conteúdo dinâmico por meio do modelo *request/response*. O servidor de aplicações utilizado, Jakarta Tomcat [99], gerencia o ciclo de vida e a segurança das aplicações. As páginas a serem exibidas são geradas em HTML e é usado o protocolo HTTP para transferência de dados.

Páginas JSP são compostas de código HTML e *tags* especiais em sintaxe Java e são identificadas pela presença da extensão *.jsp* em uma URL. Quando uma página JSP é solicitada, o servidor de aplicações *web* aciona o *servlet* tradutor de páginas e, se for a primeira vez que o arquivo contendo a página JSP é acessado, este arquivo é traduzido em um *servlet*. Nesta tradução são analisados os *tags* JSP e o código HTML. Os *tags* são convertidos em código Java, que deverá produzir a parte dinâmica das páginas, e o código HTML (porção estática) é convertido em cadeias Java que serão gravadas na seqüência original e sem alterações. Enquanto o arquivo contendo a página JSP permanecer inalterado o código Java correspondente permanece o mesmo e não há necessidade de nova tradução. Após a criação do *servlet*, que contém o código Java, o *servlet* tradudor de páginas aciona o compilador Java e este gera um arquivo de classes Java. Este arquivo é então executado para gerar a resposta à solicitação original.

Para a interface gráfica JoiNMonitor foram codificadas as seguintes aplicações JSP, correspondentes às páginas apresentadas no Apêndice C:

Fig. 3.6: Parte superior das páginas da interface gráfica JoiNMonitor

- *pageheader.jsp*: para construção do código HTML correspondente à parte superior das páginas, onde estão imagens de identificação do sistema JoiN e botões para navegação (Fig. 3.6);
- *index.jsp*: para construção da página onde são exibidas as informações relativas ao componente *server* do sistema JoiN;
- *coordinators.jsp*: para construção da página onde são exibidas as informações relativas aos componentes *coordinator* do sistema JoiN;
- *coorddetail.jsp*: para construção das páginas que contêm informações detalhadas sobre cada um dos componentes *coordinator* do sistema JoiN;
- *workers.jsp*: para construção da página onde são exibidas as informações relativas aos componentes *workers* do sistema JoiN;
- *workerdetail.jsp*: para construção das páginas que contêm informações detalhadas sobre cada um dos componentes *worker* do sistema JoiN;
- *users.jsp*: para construção da página onde são exibidas as informações relativas aos usários do sistema JoiN;
- *applications.jsp*: para construção da página onde são exibidas as informações relativas às aplicações em execução ou já processadas no sistema JoiN;
- *appdetail.jsp*: para construção das páginas que contêm informações detalhadas sobre cada uma das aplicações em execução ou já processadas no sistema JoiN;
- *collaborators.jsp*: para construção da página onde são exibidas as informações relativas aos colaboradores, do sistema JoiN.

Os usuários da interface gráfica selecionam as informações que desejam analisar pelo acionamento de botões e pelo preenchimento de campos existentes em cada uma das páginas. Por exemplo, na página exibida na Fig. 3.7 podem ser vistos os botões (identificados como *Server*, *Coordinators*, *Workers*, *Applications*, *Collaborators* e *Users*) e a área para filtro das informações a serem exibidas (campos *Hostname* e *Status*). A aplicação correspondente ao botão acionado obtém as informações selecionadas por meio de conexão estabelecida com o servidor de banco de dados e formata adequadamente o código HTML que é enviado como resposta à solicitação do usuário da interface. No Apêndice C são apresentados as descrições e alguns exemplos das páginas exibidas pela interface gráfica.

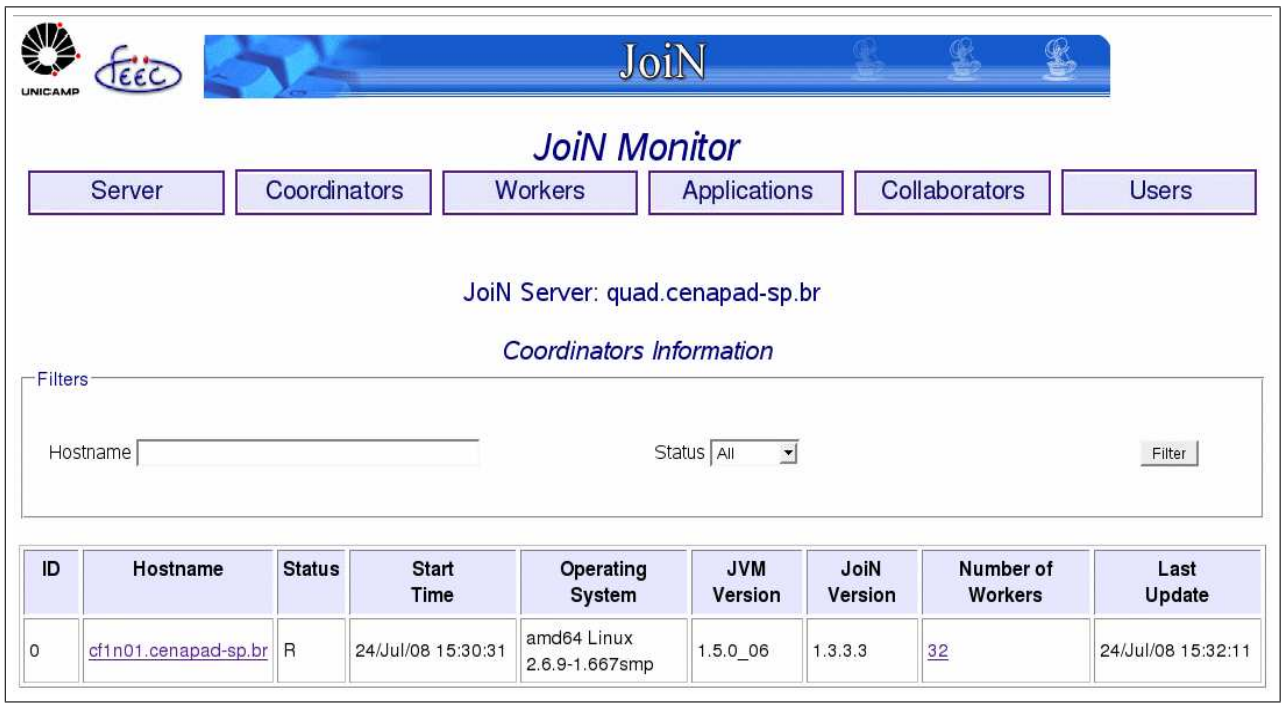

Fig. 3.7: Exemplo de página da interface gráfica JoiNMonitor - botões e filtros

3.8 Conclusões

Neste capítulo foi apresentado o sistema JoiN, destacando suas principais características e as soluções adotadas para evitar problemas com a escalabilidade, a heterogeneidade e o dinamismo deste tipo de sistema. A arquitetura hierárquica utilizada e a estrutura de implementação baseada em serviços são essenciais para o crescimento do ambiente, evitando gargalos e sobrecargas, e para a manutenção da segurança e integridade do ambiente.

Também foram apresentadas as características necessárias em uma ferramenta para monitoramento deste sistema: obter, manter e tornar disponíveis informações sobre serviços, componentes, usuários, aplicações, tarefas, colaboradores e ambiente de execução. Foram apresentados ainda critérios para avaliação desta ferramenta referentes à adequação da mesma à arquitetura e características do sistema JoiN.

Foi avaliada a possibilidade de implentação no sistema JoiN de alguma das ferramentas existentes, ou mesmo a combinação de algumas delas. Entretanto, devido à arquitetura particular do sistema JoiN e às suas características específicas, concluiu-se que não existe ferramenta que possa ser utilizada diretamente pelo JoiN, tornando-se necessário o desenvolvimento de uma ferramenta específica para este ambiente, aproveitando os pontos fortes das ferramentas existentes.

Foram apresentados a arquitetura e os componentes da ferramenta de monitoramento JoiNMon e também as soluções adotadas para superar os desafios relativos à obtenção, ao tratamento e à disponibilização de um grande volume de informações para o monitoramento. Esta ferramenta utiliza conceitos apresentados na arquitetura GMA, do GGF, e foi implementada no sistema JoiN de forma a manter as premissas básicas deste sistema com relação à escalabilidade, ao desempenho e à tolerância a falhas.

Para adequação à arquitetura GMA e às necessidades e à arquitetura do sistema JoiN, a ferramenta de monitoramento JoiNMon foi implementada por meio do serviço *JoiNMonitor* e da interface gráfica *JoiNMonitor*, complementados por modificações no serviço *EventManager* e nos serviços que têm acesso às informações necessárias para produção dos eventos a serem monitorados. As alterações no serviço *EventManager* foram feitas com o objetivo de adequar o gerenciamento de eventos à arquitetura GMA. O serviço *JoiNMonitor* foi desenvolvido de acordo com as condições previstas na arquitetura do sistema JoiN e implementado conforme a hierarquia de componentes deste sistema: *server*, *coordinators* e *workers*. A interface gráfica *JoiNMonitor* foi implementada como uma aplicação *web* usando tecnologia JSP e, para sua execução, utilizamos o servidor *Apache Tomcat*.

Neste capítulo foram também destacadas as funcionalidades implementadas pela ferramenta JoiNMon e pela interface gráfica. Esta interface é responsável pela interação entre o serviço de coleta de dados e seus usuários e facilita o acesso às informações de forma clara e objetiva.

No próximo capítulo é feita análise de resultados obtidos com a utilização da ferramenta de monitoramento JoiNMon no sistema JoiN.

Capítulo 4

Análise da ferramenta de monitoramento no sistema JoiN

Neste capítulo é feita a classificação da ferramenta de monitoramento JoiNMon de acordo com os requisitos, as características e a taxonomia apresentados no Capítulo 2. Nesta classificação são consideradas a adequação da ferramenta à arquitetura GMA e as facilidades que o monitoramento oferece, por meio das informações disponibilizadas para usuários e colaboradores do sistema. Também foram consideradas a disponibilidade e a forma de apresentação destas informações.

A seguir são analisados os resultados obtidos na avaliação do impacto da ferramenta de monitoramento JoiNMon sobre o sistema JoiN, conforme os requisitos apresentados na seção 3.4, onde são discutidas as características que devem estar presentes na ferramenta de monitoramento para este sistema. São também apresentados e discutidos experimentos que foram conduzidos com o objetivo de avaliar a sobrecarga que a ferramenta JoiNMon introduz na execução de aplicações no sistema JoiN. Para isto avaliamos se a implementação e utilização da ferramenta de monitoramento perturba a execução de aplicações típicas. Por fim é avaliado o desempenho do sistema, ou seja, se a utilização da ferramenta de monitoramento interfere na escalabilidade ou na eficiência do sistema JoiN.

4.1 Classificação da ferramenta JoiNMon

A ferramenta de monitoramento JoiNMon se enquadra na categoria "ferramenta para plataforma específica" pois foi desenvolvida para atender as características e necessidades do sistema JoiN, implementando as funções necessárias para monitoramento deste sistema de forma personalizada e otimizada.

No contexto de técnicas e modelos, JoiNMon utiliza monitoramento direto e modelo produtor/consumidor com mediador: o sistema de eventos é composto pelos serviços do sistema JoiN (produtores) que obtêm as informações a serem observadas e geram explicitamente os eventos que as contêm; o serviço de eventos é implementado pelo serviço *EventManager* que trata a assinatura e encaminhamento dos eventos, atuando como mediador entre produtores e consumidores; o serviço *JoiNMonitor* implementa o componente consumidor e é também responsável pelo armazenamento das informações coletadas. A ferramenta JoiNMon segue desta forma a proposta da arquitetura GMA, pois implementa todas as fases previstas na mesma.

Conforme a classificação em níveis, apresentada na seção 2.5.2, na qual as ferramentas que seguem a padronização definida pela GMA são agrupadas de acordo com a forma de implementação de produtores e republicadores, a ferramenta JoiNMon pode ser considerada "nível 2". Para esta classificação consideramos que a funcionalidade dos sensores é embutida nos produtores e os republicadores têm como função pré-definida tratar, filtrar e repassar os eventos para o nível hierárquico superior, de acordo com a arquitetura do sistema JoiN. Os republicadores de JoiNMon não são configuráveis e organizáveis em estrutura hierárquica, que é o requisito necessário para classificação como "nível 3", onde estão os sistemas que permitem maior escalabilidade. Entretanto, a adequação da ferramenta à arquitetura hierárquica do sistema JoiN foi feita de maneira a permitir a manutenção da escalabilidade e facilitar sua implementação de acordo com a proposta da GMA.

A classificação da ferramenta de monitoramento JoiNMon com base nos critérios apresentados na seção 2.5.3 é apresentada a seguir.

- público alvo: a ferramenta oferece informações para usuários colaboradores, administradores e visitantes do sistema JoiN. Não oferece, de forma direta, informações para desenvolvedores de serviços para o sistema JoiN ou para outros componentes dos sistema;
- funcionalidades implementadas: a ferramenta JoiNMon implementa funcionalidades para monitoramento e visualização das informações coletadas. Não implementa ainda funcionalidades para facilitar a análise destas informações, a predição de desempenho ou o direcionamento na utilização da grade;
- recursos monitorados: o monitoramento é baseado na execução das aplicações e na utilização dos computadores conectados ao sistema. Nesta versão da ferramenta não são monitorados os recursos de armazenamento ou de rede;
- facilidade na inclusão de sensores: a inclusão de sensores, para obtenção de novas informações a serem monitoradas deve ser feita de forma manual e estática, por meio de nova codificação ou de alteração de *software* já existente no sistema JoiN;
- arquitetura: a ferramenta JoiNMon segue a organização hierárquica da arquitetura do sistema JoiN e deve utilizar os serviços deste sistema para autenticação, autorização e uso de criptografia;
- interfaces: como uma ferramenta específica para o sistema JoiN, a ferramenta JoiNMon foi implementada por meio serviços deste sistema e, desta forma, está plenamente integrada a ele. Ainda não foram implementadas APIs que virão facilitar a inclusão e tratamento de novos eventos e permitir a integração com demais serviços do sistema JoiN. Esta integração deve possibilitar a utilização pelos demais serviços das informações coletadas e armazenadas por JoiNMon.

4.2 Comparação entre JoiNMon e algumas das principais ferramentas de monitoramento

Depois da classificação da ferramenta JoiNMon apresentada na seção 4.1 podemos compará-la com algumas das principais ferramentas para monitoramento de ambientes de grade computacional. Entre as ferramentas apresentadas no Apêndice A selecionamos para comparação: Ganglia, que é

muito utilizado no monitoramento de *clusters* e de ambientes corporativos; R-GMA, que tem se tornado o "padrão de mercado" nos ambientes de grade computacional pois, mais que uma ferramenta de monitoramento, tem sido utilizado como um diretório de serviços onde são registrados os recursos disponíveis na grade; JAMM, que tem como características a arquitetura distribuída e o fato de ser implementado em Java, tendo desta forma, muitos pontos em comum com JoiNMon; e MonALISA, que faz a coleta de informações por meio de agentes e possibilita a incorporação destas informações por outros sistemas. Na tabela Tab. 4.1 apresentamos um quadro comparativo entre as principais características destas ferramentas de monitoramento e de JoiNMom.

Tab. 4.1: Comparação entre ferramentas de monitoramento

Característica	$Ferramenta$				
	$Ganglia$	$R-GMA$	$JAMM$	$MonALISA$	$JoiNMon$
Requer instalação	sim	sim	sim	sim	não
Mecanismo de portabilidade	módulos diferenciados	módulos diferenciados	Java	Java	Java
Arquitetura	hierárquica	hierárquica	distribuída	distribuída	hierárquica
Vizualização	tempo-real e histórica	tempo-real e histórica	tempo-real e histórica	tempo-real e histórica	tempo-real e histórica
Coleta	cíclica	cíclica e sob demanda	sob demanda	cíclica e sob demanda	cíclica e sob demanda
Extensibilidade	comando inclusão métricas	codificação APIs	objetos RMI e arquivos configuração	via Ganglia	requer modificação código fonte
Tolerância a falhas	replicação informações	replicação de registro e de schema	base dados LDAP replicada	replicação informações	sistema JoiN
Análise das informações	gráfica e textual	via consumidor (GridICE e outros)	via NetLogger	gráfica e textual	textual
Pontos positivos	hierarquia, gráficos gerados por RRDtool	uso SQL, eficiência, escalabilidade	portabilidade, base dados LDAP	interface gráfica, agrupamento de informações, integração com outras ferramentas	não requer instalação, eficiência, portabilidade, monitoramento de aplicações

continua na próxima página

Tab. 4.1: Comparação entre ferramentas de monitoramento (continuação)

Característica	$Ferramenta$				
	$Ganglia$	$R-GMA$	$JAMM$	$MonALISA$	$JoiNMon$
Pontos negativos	complexidade, base de dados, não monitora aplicações	complexidade, vizualização	requer NetLogger, privilégios administrador	requer Ganglia/SNMP	extensilibilidade, análise gráfica

Podemos observar que cada uma destas ferramentas apresenta características próprias e específicas, ressaltando o foco dado em seu projeto, destacadas a seguir.

Em Ganglia podemos destacar o monitoramento feito de forma distribuída em que cada nó de monitoramento mantém uma cópia do estado atual de todo o ambiente monitorado, garantindo a tolerância a falhas com a replicação de informações. O armazenamento dos dados é feito por um módulo centralizador, que capta as informações periodicamente de qualquer um dos nós, em uma base de dados circular chamada RRDtool (Round Robin Database Tool). Os dados desta base de dados podem ser visualizados graficamente por meio de uma interface *web*. A utilização de uma base de dados circular restringe o volume de informações que pode ser armazenado. Para manter informações históricas Ganglia as agrupa e sumariza em períodos de tempo cada vez maiores, diminuindo o nível de detalhamento em função do tempo. Para a medição de novas informações em Ganglia deve ser usado o comando *gmetric* que, executado em um computador sob monitoramento, pode adicionar medidas ao conjunto de métricas.

De R-GMA destacam-se a utilização da arquitetura GMA e o modelo produtor/consumidor em que os produtores publicam os dados em forma de tuplas e os consumidores as obtêm por meio de comandos SQL, possibilitando comandos bastante elaborados para seleção de informações. Por esta razão R-GMA é utilizado como serviço de diretório em implementações da grades computacionais. O serviço de monitoramento GridICE utilizado nos projetos EGEE e EELA é implementado como um componente consumidor das informações coletadas e arquivadas por R-GMA.

Na ferramenta JAMM, destacamos a arquitetura de agentes distribuídos, o modelo produtor/consumidor e a publicação das informações em uma base de dados LDAP. O uso de LDAP possibilita o acesso a estas informações por meio de um protocolo simples e aberto, com suporte ao TCP/IP.

O ponto forte de MonALISA é a possibilidade de utilização de informações obtidas por outras ferramentas. Pode utilizar o SNMP, Ganglia ou MRTG (Multi Router Traffic Grapher) [102] para obter informações sobre a rede e recursos do ambiente. MRTG é um *software* livre que obtém informações SNMP e gera gráficos que permitem o monitoramento do tráfego em recursos de rede.

Em todas estas ferramentas é necessária a instalação de *software* com porções das ferramentas de monitoramento em cada computador a ser monitorado. Esta é uma desvantagem destas ferramentas com relação a JoiNMon, que foi implementada por meio de serviços do sistema JoiN e tem como principal vantagem a plena integração a este sistema. As características da ferramenta JoiNMon, pontos positivos e negativos, são discutidos mais detalhadamente na seção 4.3. Os pontos fortes presentes em outras ferramentas e não contemplados nesta implementação de JoiNMon, como tolerância a falhas, análise gráfica das informações e extensibilidade, podem ser incorporados a JoiNMon em

novos trabalhos, aproximando-a das demais ferramentas.

4.3 Avaliação do impacto da ferramenta JoiNMon sobre o sistema JoiN

Nesta seção usaremos diretrizes estabelecidas pela norma ISO 9126 (NBR 13596 na versão brasileira) que define seis características para avaliação da qualidade de um *software*. Entretanto, como não temos por objetivo atestar a qualidade ou a certificação do *software* desenvolvido, usaremos estas características como parte do roteiro para a avaliação do impacto da ferramenta JoiNMon sobre o sistema JoiN e seus usuários, atentando para as vantagens e possíveis desvantagens que esta ferramenta apresenta. As características analisadas são:

- Funcionalidade - finalidade do *software* e atendimento dos requisitos;
- Confiabilidade - tolerância a falhas e freqüência com que estas ocorrem;
- Usabilidade - esforço necessário para aprender e utilizar o *software*;
- Eficiência - desempenho do *software*;
- Manutenibilidade - esforço necessário para modificar o *software*;
- Portabilidade - possibilidade de usar o *software* em ambientes com diferentes configurações.

4.3.1 Funcionalidade - atendimento dos requisitos

Nesta análise verificamos se a ferramenta JoiNMon atende as expectativas e satisfaz os principais requisitos de uma ferramenta de monitoramento adequada ao ambiente de computação em grade, como proposto no decorrer do Capítulo 2, que são:

- coleta eficiente dos dados e possibilidade de medições periódicas ou sob demanda;
- capacidade para tratamento de grande volume de dados;
- baixa latência no tratamento das informações;
- transmissão, armazenamento e tratamento das informações coletadas com segurança;
- monitoramento de diferentes tipos de eventos;
- disponibilidade das informações;
- apresentação das informações de forma clara.

Coleta de dados para medição

A obtenção dos dados de medição pela ferramenta JoiNMon pode ser feita periodicamente ou sob demanda. Na obtenção de dados sob demanda, os consumidores, implementados por meio do serviço *JoiNMonitor*, recebem de forma assíncrona as informações a serem monitoradas no momento em que são notificados sobre a ocorrência de algum dos eventos para os quais estão registrados. Um mecanismo de *cache* foi criado por meio da manutenção de informações pelos componentes da hierarquia do sistema JoiN. Esta é uma outra forma de obtenção das informações, onde o serviço *JoiNMonitor*

faz solicitações periódicas aos serviços correspondentes nos demais componentes, conforme a hierarquia do sistema. A criação deste mecanismo de *cache* teve por objetivo diminuir o impacto da ferramenta no sistema, permitindo o agrupamento de informações que, transmitidas e manipuladas em bloco, evitam tráfego de rede desnecessário e contribuem para a eficiência da ferramenta.

Tratamento de grande volume de dados

A ferramenta JoiNMon prevê o tratamento de um grande volume de informações, coletadas por diferentes sensores com o objetivo de permitir o acompanhamento do estado das aplicações executadas e dos computadores conectados ao sistema JoiN.

A eficiência no tratamento de grandes volumes de dados é um problema que demanda mecanismos para organização, armazenamento, busca e avaliação destes dados. Para enfrentar este problema a ferramenta JoiNMon utiliza uma base de dados relacional. A modelagem deste banco de dados, apresentada no Apêndice B, tem por objetivo manter e facilitar o acesso às informações coletadas pela ferramenta JoiNMon, tanto na gravação como na recuperação das mesmas. A utilização do servidor de banco de dados HSQLDB foi adotada com o objetivo de facilitar o processo de instalação e favorecer o desenvolvimento e a avaliação da ferramenta de monitoramento. Em avaliações divulgadas pelo projeto PolePosition [103], (um conjunto de testes desenvolvidos para avaliação e comparação de servidores de banco de dados) o HSQLDB apresenta, de modo geral, um desempenho igual ou superior a outros bancos de dados distribuídos como *software* livre. Nestas avaliações o HSQLDB apresenta bom desempenho na operação com centenas de milhares de registros, o que atende as necessidades das instalações típicas do sistema JoiN. Além disto, como pode operar embutido em vez de executar como um servidor de rede, facilita a distribuição do código do sistema JoiN. HSQLDB é o servidor de banco de dados utilizado, por exemplo, no OpenOffice 2.0, no Mathematica e no InstallAnywhere. Em instalações muito grandes do sistema JoiN, com centenas de milhares de colaboradores, em que as aplicações também tenham centenas de milhares de tarefas, o servidor de banco de dados pode ser substituído por outro servidor mais robusto. Isto deve ser feito por meio de alteração na conexão JDBC. Nestes casos podemos optar, entre os gerenciadores distribuídos como software livre, entre MySQL, recentemente adquirido pela Sun Microsystems, que tem foco na agilidade e no processamento rápido dos dados, e PostgreSQL que é otimizado para aplicações complexas que envolvem grandes volumes de dados.

Baixa latência no tratamento das informações

A observação do estado do sistema e das aplicações em tempo real requer baixa latência tanto na coleta quanto no tratamento e apresentação das informações. Para coletar e tratar grandes volumes de dados e minimizar a latência, a ferramenta JoiNMon separa a lógica computacional da camada de apresentação. O pré-processamento dos dados, incluindo a formatação e geração de eventos, é executado na mesma máquina onde as informações são obtidas, minimizando, desta forma, a latência. Nesta abordagem, a organização dos dados, a forma de comunicação e o tratamento das informações permitem que se obtenha redução significativa na latência e melhor desempenho.

O mecanismo de *cache*, que permite a manipulação e a transmissão das informações em bloco, seguindo a hierarquia do sistema JoiN, dispensa a necessidade de compartilhamento de resultados parciais e prioriza a redução no tráfego de rede podendo, todavia, causar um aumento na latência

para atualização das informações. Entretanto, o possível aumento de latência não tem conseqüências graves, uma vez que a taxa de amostragem pode ser ajustada por meio de parâmetros de configuração da ferramenta e, como envolve um grande volume de dados, um pequeno atraso no início da transferência não compromete a operação, e assim é possível verificar o estado do sistema.

Segurança no tratamento das informações coletadas

A segurança no tratamento das informações tem o objetivo de garantir a integridade, o sigilo, a autenticidade e a disponibilidade destas informações. A integridade, a autenticidade e o sigilo são relacionados com os controles de acesso às informações.

A ferramenta JoiNMon não implementa mecanismos de segurança na coleta e transmissão das informações pois, como um serviço do sistema JoiN, utiliza-se da segurança prevista neste sistema. Estes mecanismos estão sendo implementados pelo grupo que desenvolve a ferramenta de segurança.

Esta ferramenta, o gerenciador de segurança, deve implementar as políticas de acesso entre serviços, garantir e controlar o acesso a serviços que desempenham funções críticas, como é o caso da ferramenta de monitoramento. Além disto, o gerenciador de seguranca é responsável pela identificação, autenticação e autorização de usuários e colaboradores, e também pelo uso de criptografia na transmissão de dados.

As informações disponíveis devem também ser íntegras, o que pressupõe que os dados não podem ser alterados, de forma proposital ou acidental, seja por meio de trocas, inclusões ou exclusões. O sigilo das informações é mantido pelo controle de acesso aos dados armazenados na base de dados. Este controle é implementado pelo gerenciador de bancos de dados (HSQLDB), com base nas políticas definidas, por meio da identificação de usuário e autenticação mediante senha.

No uso da interface *web* também deve ser implementado o controle de acesso, mediante autenticação de usuário pela sua senha. Além disto, de acordo com o tipo de usuário, deve ser feito o controle do nível de informações que podem ser exibidas. Esta implementação deverá ser feita posteriormente pois depende da disponibilidade do gerenciador de segurança do sistema JoiN. Ainda com relação à interface *web*, o servidor Jakarta Tomcat [99] é responsável pelo gerenciamento do ciclo de vida e a segurança da interface *JoiNMonitor*.

Monitoramento de diferentes tipos de eventos

Na fase de planejamento da ferramenta JoiNMon foi feita a identificação das informações relevantes: aquelas que devem ser monitoradas para que os usuários e administradores do sistema JoiN tenham acesso ao estado, atual e histórico, das aplicações e do sistema JoiN. A seguir, o código fonte do sistema JoiN foi analisado e foram identificados os serviços que têm acesso a estas informações relevantes. Um código único foi associado a cada uma destas informações; este código identifica o evento a ser monitorado. Cada evento contém informações que permitem à ferramenta JoiNMon realizar o monitoramento das aplicações e do sistema. Os diferentes tipos de eventos monitorados por JoiNMon são apresentados na tabela Tab. 3.1.

Disponibilidade e apresentação das informações

A ferramenta de monitoramento possibilita o acesso a informações que podem ser analisadas *on-line* e que refletem o estado atual do sistema, dos computadores a ele conectados e das aplicações

em execução. Também permite o acesso a informações históricas, mantidas na base de dados, que viabilizam análises *post-mortem* da execução de aplicativos já encerrados e da utilização de computadores que já estiveram conectados ao sistema em algum momento. Estas informações históricas são importantes para a contabilização da utilização de recursos para, por exemplo, a adoção de políticas de recompensas aos participantes do esforço de processamento.

Por disponibilidade das informações entende-se a garantia de que as mesmas estarão sempre disponíveis para as pessoas e os processos autorizados a acessá-las, a qualquer momento. Ou seja, para manter a disponibilidade das informações é necessário garantir o acesso a elas, sem interrupções.

Para que as informações coletadas pela ferramenta JoiNMon sejam armazenadas na base de dados e possam ser sempre exibidas pela interface gráfica, é necessário que o servidor de bancos de dados HSQLDB e o servidor de aplicações *web* Apache Tomcat estejam permanentemente ativos no componente *server* do sistema JoiN.

A avaliação da usabilidade e da forma de apresentação das informações de monitoramento pela interface gráfica é apresentada na seção 4.3.4.

4.3.2 Confiabilidade - tolerância a falhas

Conforme mencionado anteriormente, o sistema JoiN constitui um ambiente heterogêneo e dinâmico. Diferentes tipos de computadores, de colaboradores distintos, podem conectar-se ou desconectar-se a qualquer momento. Este fraco acoplamento entre os componentes *worker* e seu *coordinator* faz com que o ambiente esteja exposto a falhas. A ocorrência de falhas, permanentes ou temporárias, causadas pela desconexão de componentes *worker* não deve comprometer o funcionamento do sistema JoiN e deve ser identificada e devidamente tratada pela ferramenta de monitoramento. Isto é feito por meio do evento identificado como *WORKER_DOWN*, disparado pelo módulo *C_Communicator* dos componentes *coordinator* toda vez que a desconexão de um *worker* é identificada. Da mesma forma, o evento identificado como *COORD_DOWN* é disparado pelo módulo *S_Communicator* quando verificada a desconexão de um componente *coordinator*.

A manutenção da integridade dos serviços no sistema JoiN e a conclusão da execução das aplicações, mesmo quando houver falhas, devem ser feitos por um um mecanismo de tolerância a falhas, a ser agregado ao sistema para torná-lo mais robusto à ocorrência de falhas, em particular àquelas que ocorram nos componentes *server* e *coordinator*. A ferramenta JoiNMon, como um serviço implementado no sistema JoiN, deverá ser beneficiada com os futuros mecanismos de tolerância a falhas.

4.3.3 Manutenibilidade e extensibilidade

A manutenibilidade requer conhecimento do código fonte e acesso à documentação do sistema JoiN pois, como a ferramenta JoiNMon foi implementada por meio de serviços deste sistema, é necessário o conhecimento da arquitetura e restrições do mesmo. O entendimento, a manutenção e eventuais modificações nos serviços são facilitados pela inclusão de comentários no código fonte e pela documentação das classes Java implementadas. Esta documentação é produzida por meio do sistema Javadoc [104] criado pela Sun Microsystems para documentar a interface de programação (API - *Application Programming Interface*) de programas em Java a partir do código-fonte. O programador insere marcações nos comentários do programa e o sistema Javadoc produz a documentação correspondente no formato HTML.

Quanto à extensibilidade, ou seja, a capacidade de se estender o monitoramento a novos tipos de eventos, constatou-se que também neste caso é exigido o conhecimento do sistema JoiN, do mecanismo de geração e tratamento de eventos, e da base de dados utilizada, para a criação de novas tabelas onde as informações serão armazenadas. Além disto, é necessária a alteração da interface gráfica e o desenvolvimento de novas aplicações para exibição das informações obtidas. Entretanto, acreditamos que praticamente todos os eventos necessários para o monitoramento das principais características do sistema já foram considerados e incluídos na atual implementação. Portanto, o ponto que poderia necessitar de mais extensões seria aquele relativo à extração e apresentação de informações mais elaboradas (gerenciais) a partir dos dados brutos presentes na base de dados como, por exemplo, indicar a parcela de participação de um dado colaborador na conclusão de uma determinada aplicação, ou então indicar a eficiência de um determinado computador conectado ao sistema a partir da utilização dos resultados produzidos pelo mesmo na execução de tarefas de uma aplicação. Para obter estas informações gerenciais não será necessária nenhuma manutenção no sistema, mas apenas um novo planejamento sobre como usar os dados já disponíveis no banco de dados, o que pode ser feito por um SIG (Sistema de Informações Gerenciais).

4.3.4 Usabilidade - Interface gráfica

Como já salientado na seção 3.6.6, o tema "interfaces gráficas" envolve profissionais de diversas capacitações e é objeto de estudo da área de Interação Humano-Computador (IHC). Dentro desta área existem diversos métodos para avaliação da usabilidade e para a classificação de páginas *web*. Não vamos seguir a risca todas as classificações da área de IHC, pois o nosso objetivo é que a interface gráfica seja simples, intuitiva e de fácil uso. Assim, avaliamos a usabilidade da interface gráfica por meio de critérios mais práticos, verificando que a mesma segue as principais recomendações para o desenho de páginas e aplicações *web*:

- as informações devem ser apresentadas pela interface gráfica de forma concisa e eficiente;
- na construção das páginas não devem ser utilizados fontes ou tabelas de tamanhos fixos;
- os textos devem ser apresentados em formato simples, para reduzir o tamanho das páginas e, consequentemente, o tempo necessário para *download* das mesmas;
- as páginas não devem apresentar *"links quebrados"*, ou seja, apontadores para páginas inexistentes;
- a estética e desenho das páginas deve ser minimalista, privilegiando as informações e não os artefatos gráficos. Deve-se procurar encontrar soluções simples e funcionais, limpas de complementos e excessos;
- as páginas devem permitir navegação fácil;
- as páginas devem utilizar linguagem adequada aos seus usuários.

Entendemos que a interface gráfica implementada atende estas recomendações, devido ao fato de ter sido implementada por meio de aplicação *web*. Tanto usuários do JoiN, como seus colaboradores voluntários e administradores, podem valer-se desta interface para obter e analisar as informações desejadas. Na Fig. 4.1 é apresentada, como exemplo, a imagem da página onde são exibidas as informações sobre os componentes *worker* conectados ao sistema JoiN.

ID	Hostname	Coordinator	Status	Start Time	Tasks Running
0	cf1n02.cenapad-sp.br	cf1n01.cenapad-sp.br	R	15/Jul/08 10:00:58	0
1	cf1n04.cenapad-sp.br	cf1n01.cenapad-sp.br	R	15/Jul/08 10:01:01	0
2	cf1n07.cenapad-sp.br	cf1n01.cenapad-sp.br	R	15/Jul/08 10:00:58	0
3	cf1n03.cenapad-sp.br	cf1n01.cenapad-sp.br	R	15/Jul/08 10:00:56	0
4	cf2n03.cenapad-sp.br	cf1n01.cenapad-sp.br	R	15/Jul/08 10:00:58	0
5	cf2n01.cenapad-sp.br	cf1n01.cenapad-sp.br	R	15/Jul/08 10:00:58	0
6	cf2n08.cenapad-sp.br	cf1n01.cenapad-sp.br	R	15/Jul/08 10:00:58	0

Fig. 4.1: JoiNMonitor - Exemplo ilustrativo da interface gráfica

A apresentação das informações é textual, o que pode requerer certo esforço para sua interpretação. Há um amplo espaço para melhorias aqui, especialmente na parte relativa à criação de gráficos e análises estatísticas mais sofisticadas dos dados monitorados e, conforme ressaltado no item anterior, apresentar informações gerenciais de forma mais elaborada.

4.3.5 Portabilidade

A portabilidade do sistema JoiN, característica obtida pela utilização da linguagem Java, é mantida após a implementação da ferramenta JoiNMon pois esta também foi desenvolvida totalmente em Java, utilizando sensores e eventos independentes de plataforma. O novo código desenvolvido, bem como as modificações efetuadas no código dos serviços já existentes no sistema JoiN não criaram novas exigências para compilação ou instalação deste sistema. Também não é requerido nenhum esforço extra por parte do programador de aplicações, uma vez que a ferramenta de monitoramento é totalmente transparente para ele.

4.3.6 Eficiência - realização de experimentos

Dentre os requisitos necessários para ferramentas de monitoramento de ambientes de grade, a sobrecarga mínima na obtenção e tratamento dos dados e a geração de pouco tráfego de rede são de grande importância para a eficiência da ferramenta, de forma a não degradar o desempenho e manter a escalabilidade da grade.

A avaliação da eficiência da ferramenta JoiNMon foi feita pela medição da sobrecarga que a mesma introduz no sistema JoiN, no tempo de execução das aplicações e no tráfego de rede. Para isto foram feitos experimentos com diferentes configurações do sistema, variando o número de componentes *worker*, o tipo de aplicação e o número de tarefas. Com a realização destes experimentos foi possível avaliar o impacto causado pela ferramenta no processamento das aplicações e no tráfego de rede. Aqui é apresentada a forma como estes experimentos foram conduzidos. Os resultados obtidos e a análise dos mesmos são apresentados na seção 4.4.

Cada uma das aplicações foi executada em diferentes configurações e, para cada configuração, os experimentos foram feitos em duas etapas: na primeira etapa foi utilizada a plataforma JoiN sem a ferramenta de monitoramento; na segunda etapa a ferramenta de monitoramento foi ativada e os experimentos repetidos. Além disto, cada experimento foi feito cinco vezes, para que pudesse ser calculada a média dos valores obtidos, que é o valor utilizado nas avaliações.

Configuração dos computadores

Para execução destes experimentos foi utilizado um ambiente homogêneo, composto por um *cluster* com 24 nós, cada um dos nós com a seguinte configuração:

- 2 processadores AMD-Opteron, 1.6GHz e 2GB RAM;
- interconexão por rede Gigabit Ethernet;
- JoiN Version 1.3.3.3 (com um grupo, ou seja, um componente *coordinator*);
- sistema operacional Fedora3/amd64 Linux_2.6.9-1.667smp;
- JVM Version 1.5.0_06.

Foi utilizado um computador, com a mesma configuração e ligado na mesma rede, para a execução dos componentes *server* e *coordinator*. Foram realizados experimentos com 4, 8, 12, 16, 24 e 32 componentes *worker*. Para as configurações entre 4 e 24 componentes *worker*, foi executado um *worker* em um processador de cada nó. Para a configuração com 32 *workers*, foram ativados dois *workers* em 8 dos nós, utilizando a configuração com 2 processadores destes nós, o que manteve a relação de um componente *worker* por processador.

Para as análises de tráfego de rede foram utilizadas informações capturadas pela ferramenta Wireshark Version 1.0.0 [105]. Wireshark é uma ferramenta para análise de protocolos de rede, utilizada por diversas indústrias e instituições de ensino, que facilita a seleção e análise de dados capturados por meio do comando *dumpcap* existente em sistemas *Unix*.

Classificação das aplicações

As aplicações podem ser classificadas em três grupos conforme a razão entre o tempo consumido em computação e o tempo consumido para a comunicação necessária para sua execução (R_{cc} = *tempo_de_computação / tempo_de_comunicação*).

1. $R_{cc} >> 1$ - onde o tempo de computação é muito maior que o tempo consumido para comunicação. Nesta classe estão as aplicações que melhor utilizam os recursos da plataforma JoiN, pois melhor exploram o paralelismo.
2. $R_{cc} > 1$ - onde o tempo de computação ainda é superior ao tempo de comunicação. As aplicações desta classe também exploram o paralelismo disponível, mas não tão bem como no caso anterior e, assim, não se consegue uma alta eficiência; caso mais comum quando há uso de redes heterogêneas e compartilhadas.
3. $R_{cc} <= 1$ - onde o tempo de comunicação é maior ou igual ao tempo gasto em computação. As aplicações desta classe não são adequadas para o uso de ambientes paralelos e não são consideradas.

Com base nesta classificação, a Tab. 4.2 apresenta as aplicações utilizadas nos experimentos realizados para avaliação da eficiência da ferramenta JoiNMon.

Tab. 4.2: Aplicações utilizadas nos experimentos

Aplicação	Razão computação/comunicação (R_{cc})
Cálculo do número π pelo método Monte Carlo	$R_{cc} >> 1$
Multiplicação de matrizes	$R_{cc} > 1$

Estas aplicações foram selecionadas para que os experimentos permitissem a observação e a análise do comportamento da ferramenta JoiNMon nas distintas condições em que o sistema JoiN pode ser utilizado.

Para estimar o valor de R_{cc} destas duas aplicações e classificá-las adequadamente, realizamos alguns experimentos específicos nos quais utilizamos os mesmos computadores e configurações apresentados anteriormente: ativamos os componentes *server* e *coordinator* em um computador e um componente *worker* em um dos nós do cluster. Para cada uma das aplicações, cálculo do número π e multiplicação de matrizes, executamos um conjunto reduzido de tarefas (apenas 3). Cada experimento foi feito três vezes e foi calculada a média dos valores obtidos, que foi o valor utilizado como resultado dos experimentos. Desta forma pudemos estimar os tempos de computação e de comunicação para calcular, em valores aproximados, os valores de R_{cc} destas aplicações.

A utilização de apenas um componente *worker* e de um conjunto de três tarefas foi adotada como forma de eliminar as dificuldades na obtenção de informações bem como os fatores não relevantes que poderiam ser introduzidos pela utilização de um grande número de tarefas e de componentes *worker*. Por exemplo, como cada componente *worker* do sistema JoiN pode processar até três tarefas simultaneamente e o escalonamento de tarefas deste sistema faz a replicação de tarefas ainda não concluídas, o uso de mais do que um componente *worker* ou de uma aplicação com um número maior do que três tarefas causaria replicações de tarefas e, conseqüentemente, teríamos mais dificuldades para identificar o tráfego das tarefas originais (não réplicas) que são o objetivo destes experimentos.

No gráfico apresentado na Fig. 4.2, obtido por meio da ferramenta *Wireshark* durante a execução dos experimentos com a aplicação para cálculo do número π, é possível observar que são raros e pequenos os intervalos de tempo em que ocorre algum tipo de comunicação. Neste gráfico o eixo *X* (tempo) está na escala de 0,1 segundos para que seja possível observar os instantes em que ocorre

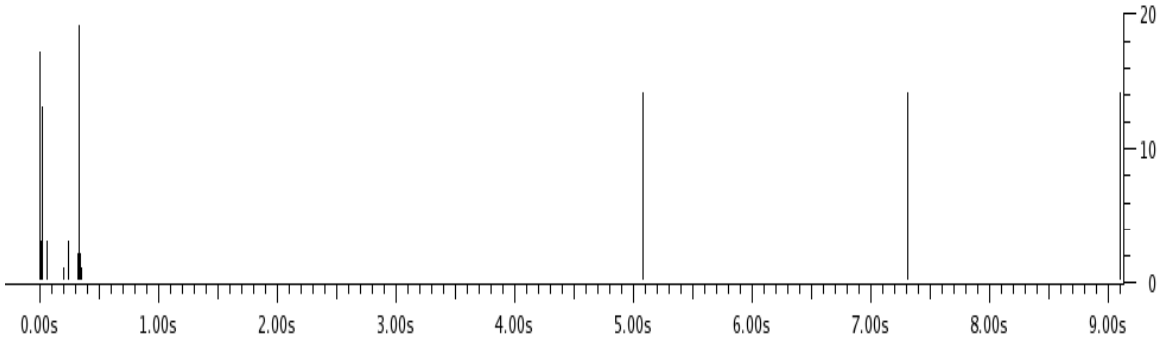

Fig. 4.2: Cálculo de π - Pacotes transmitidos X Tempo (s)

comunicação durante a execução de toda a aplicação. O eixo *Y* representa o número de pacotes transmitidos. Na seção D.1 do Apêndice D é apresentado um gráfico do início da execução da aplicação na escala de milissegundos.

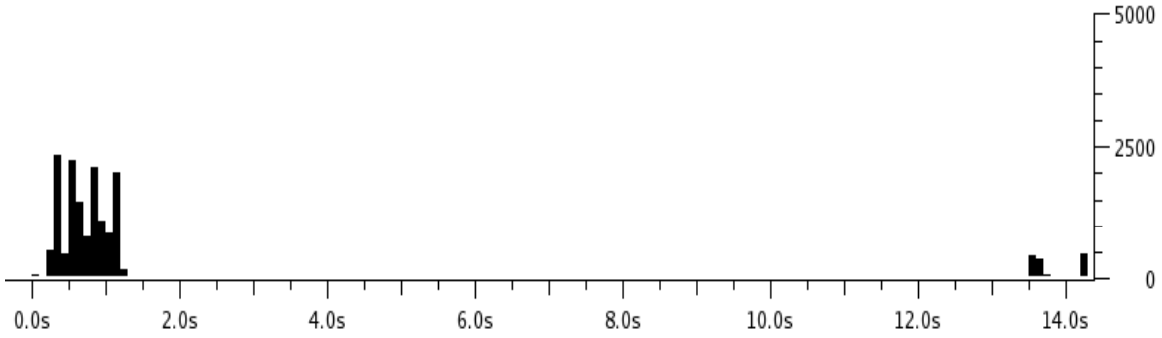

Fig. 4.3: Multiplicação de matrizes - Pacotes transmitidos X Tempo (s)

No gráfico da Fig. 4.3, também obtido por meio da ferramenta *Wireshark*, são apresentadas as informações obtidas durante a execução destes experimentos com a aplicação para multiplicação de matrizes. Neste gráfico o eixo *X* (tempo) está na escala de 0,1 segundos para que seja possível observar os instantes em que ocorre comunicação durante a execução de toda a aplicação. O eixo *Y* representa o número de pacotes transmitidos. Na seção D.1 do Apêndice D são apresentados gráficos referentes aos instantes em que ocorre comunicação na escala de milissegundos.

Tab. 4.3: Cálculo estimado de R_{cc}

Aplicação	Tempo execução (ms)	Intervalo (ms)		R_{cc}	
		SEM comunicação	COM comunicação	(pior caso)	(melhor caso)
Cálculo de π	8870	9078	20	456,57	457,57
Multiplicação matrizes	14011	13510	957	14,14	15,14

Na Tab. 4.3 são apresentados os valores utilizados para o cálculo estimado do R_{cc} para as aplicações cálculo de π e multiplicação matrizes. Nas colunas identificadas como "Tempo execução" e

"Intervalo" são exibidos os valores médios, calculados com base nos resultados obtidos em cada um dos experimentos (os valores obtidos nestes experimentos são apresentados na seção D.1 do Apêndice D). A coluna "Intervalo" indica períodos nos gráficos da Fig. 4.2 e da Fig. 4.3. A coluna R_{cc} apresenta os valores calculados para cada uma das aplicações. Esta coluna é subdividida em duas: na primeira subcoluna são apresentados os valores obtidos no "pior caso", onde consideramos que não há computação enquanto ocorre comunicação (este é o pior cenário para sistemas paralelos); na segunda subcoluna são apresentados os valores obtidos no "melhor caso", onde consideramos que computação e comunicação ocorrem simultaneamente. Podemos ainda observar nesta tabela que os valores dos tempos para execução das aplicações, obtidos por meio do sistema JoiN ao término do processamento das mesmas, são menores que as somas dos respectivos intervalos de tempos de computação e de comunicação. Isto ocorre porque a ativação e o encerramento da captura de informações pela ferramenta *Wireshark* são feitos manualmente, logo antes da submissão das aplicações e logo após o término da sua execução, o que faz com que o monitoramento seja feito por um intervalo maior que o tempo de processamento da aplicação. A imprecisão no processo de aquisição de dados para o cálculo estimado do valor de R_{cc} não introduz grandes diferenças nos valores obtidos.

No caso da aplicação para cálculo de π, as informações da Fig. 4.2 e da Tab. 4.3 nos permitem concluir que o tempo gasto em comunicação é muito pequeno se comparado ao tempo em gasto com computação. Se considerarmos que não há computação enquanto ocorre comunicação teremos, no pior caso, $R_{cc} \sim= 456,57$. Como sabemos que a comunicação acontece simultaneamente a algum processamento, a aplicação para cálculo do número π está classificada adequadamente no primeiro grupo ($R_{cc} >> 1$).

Utilizamos as informações da Fig. 4.3 e da Tab. 4.3 para estimar o valor de R_{cc} para a aplicação para multiplicação de matrizes. Da mesma forma que fizemos anteriormente, se considerarmos que não há computação enquanto ocorre comunicação teremos $R_{cc} \sim= 14,14$. Se, no melhor caso, considerarmos que computação e comunicação ocorrem simultaneamente, ainda teríamos $R_{cc} \sim= 15,14$. Com base nestas informações é possível concluir que o tempo gasto em comunicação é da mesma ordem de grandeza do tempo em gasto com computação, e assim podemos classificar a aplicação de multiplicação de matrizes na classe $R_{cc} > 1$.

Aplicação para cálculo de π

Para a execução da aplicação para cálculo de π pelo método de Monte Carlo foram executados experimentos utilizando as configurações apresentadas na Tab. 4.4, que têm como principal característica a manutenção da carga total fixa, ou seja, manteve-se constante a multiplicação entre o número de tarefas e o número de pontos em cada tarefa.

Cada um dos 18 experimentos foi executado 5 vezes, portanto foram realizados 90 experimentos em cada uma das duas etapas (a primeira sem a ferramenta de monitoramente e a segunda com a ferramenta ativada), totalizando 180 experimentos. O resultado obtido em cada um deles é apresentado no Apêndice D. A avaliação dos resultados obtidos é feita na seção 4.4.

Aplicação para multiplicação de matrizes

Para a execução da aplicação para multiplicação de matrizes foram executados experimentos utilizando as configurações apresentadas na na Tab. 4.5, que têm como principal característica a manu-

Tab. 4.4: Configurações para os experimentos com cálculo de π

Número *workers*	Número tarefas	Número pontos/tarefa
4	1000	20.000.000
	2000	10.000.000
	4000	5.000.000
8	1000	20.000.000
	2000	10.000.000
	4000	5.000.000
12	1000	20.000.000
	2000	10.000.000
	4000	5.000.000
16	1000	20.000.000
	2000	10.000.000
	4000	5.000.000
24	1000	20.000.000
	2000	10.000.000
	4000	5.000.000
32	1000	20.000.000
	2000	10.000.000
	4000	5.000.000

Tab. 4.5: Configurações para os experimentos com multiplicação de matrizes

Número *workers*	Número tarefas		
4	100	200	400
8	100	200	400
12	100	200	400
16	100	200	400
24	100	200	400
32	100	200	400

tenção constante das dimensões das matrizes utilizadas. Também neste caso a carga total é mantida constante, pois a primeira matriz é dividida em um número de submatrizes igual ao número de tarefas a serem executadas. Nestes experimentos foi feita a multiplicação de uma matriz de dimensão (4.000 X 1.000) por uma matriz de dimensão (1.000 X 4.000). Nos experimentos com 100 tarefas são feitas multiplicações de 100 submatrizes de dimensão (40 X 1.000) pela matriz de dimensão (1.000 X 4.000), Nos experimentos com 200 tarefas são feitas multiplicações de 200 submatrizes de dimensão (20 X 1.000) pela matriz de dimensão (1.000 X 4.000), E nos experimentos com 400 tarefas são feitas multiplicações de 400 matrizes de dimensão (10 X 1.000) pela matriz de dimensão (1.000 X 4.000),

Também para esta aplicação, cada um dos 18 experimentos foi executado 5 vezes, portanto foram realizados 90 experimentos em cada uma das duas etapas, totalizando 180 experimentos. O resultado obtido em cada um deles é apresentado no Apêndice D. A avaliação dos resultados obtidos é feita na seção 4.4.

4.4 Análise dos resultados experimentais

Nesta seção vamos analisar os resultados obtidos nos experimentos realizados com as aplicações para cálculo de π e para multiplicação de matrizes. Utilizamos nestas análises o valor médio dos resultados obtidos em cada uma das 5 vezes que cada um dos experimentos foi executado. Para validar a utilização da média, calculamos o desvio padrão e o coeficiente de variação, que é igual ao desvio padrão dividido pela média e é uma medida que facilita a comparação de experimentos diferentes. Os valores de desvio padrão e coeficiente de variação de cada experimento são apresentados nas tabelas detalhadas do Apêndice D. Nestas mesmas tabelas observamos que o coeficiente de variação dos experimentos são relativamente pequenos, o que representa uma alta repetibilidade apresentada pelo sistema. Por esta razão, podemos considerar que os valores médios são apropriados para a avaliação de resultados.

Na execução da aplicação para o cálculo de π pelo método de Monte Carlo o tempo de computação é muito maior que o tempo consumido para comunicação ($R_{cc} >> 1$), pois para cada uma das tarefas é enviado apenas um parâmetro (o número de pontos no plano cartesiano que a tarefa deve sortear para cálculo de π), e cada tarefa também retorna apenas um valor (o número de pontos sorteados que se encontram dentro de uma dada circunferência).

Por outro lado, na execução da aplicação para multiplicação de matrizes o tempo de computação não é tão superior ao tempo consumido para comunicação ($R_{cc} > 1$), pois para cada uma das tarefas são enviadas as matrizes a serem multiplicadas: são enviadas todas as linhas e colunas da segunda matriz e a primeira matriz é subdivida conforme o número de tarefas a serem executadas e cada porção da mesma é enviada para a tarefa correspondente. Cada tarefa retorna uma submatriz contendo o resultado da multiplicação. Assim sendo, o volume de informações a serem enviadas e recebidas pode ser grande e, portanto, o tempo consumido em comunicação pode ser relativamente grande se comparado ao tempo exigido para o processamento da multiplicação.

4.4.1 Avaliação do impacto no desempenho

Na Tab. 4.6 e na Tab. 4.7 são exibidos, resumidamente, os dados obtidos nos experimentos realizados para observação do impacto da ferramenta de monitoramento JoiNMon no tempo de execução

destas aplicações. Em cada uma destas tabelas são apresentados os dados obtidos na primeira etapa, feita sem a ferramenta de monitoramento, na coluna intitulada *JoiNMon Inativo*, e os dados obtidos na segunda etapa, com a ferramenta JoiNMon ativada, na coluna *JoiNMon ativo*. Para estas duas etapas é apresentado o valor médio, em milissegundos, do tempo de processamento da aplicação obtido em cada uma das configurações do sistema JoiN apresentadas na Tab. 4.4 e na Tab. 4.5. Nas duas últimas colunas são apresentadas a diferença entre os valores médios de tempo de execução e a sobrecarga que a ferramenta de monitoramento acarreta no tempo de execução destas aplicações. Estas informações são também representadas em Fig. 4.8, Fig. 4.9, Fig. 4.14 e Fig. 4.15. Em cada uma destas figuras são mostrados dois gráficos: o primeiro com as curvas relativas às diferenças entre os valores médios (em milissegundos) e o segundo com as curvas de "sobrecarga relativa" (porcentagem da sobrecarga sobre os valores obtidos com *JoiNMon Inativo*).

Impacto na aplicação para cálculo de π

Tab. 4.6: Cálculo de π - Tempo médio processamento (ms)

Número *workers*	Número tarefas	JoiNMon		Comparativo	
		Inativo	Ativo	Diferença	Sobrecarga
4	1000	740744	751638	10894	1.47%
	2000	746631	754101	7470	1.00%
	4000	771479	775437	3958	0.51%
8	1000	345839	356818	10979	3.17%
	2000	353007	360904	7897	2.24%
	4000	366108	367573	1465	0.40%
12	1000	234378	242675	8297	3.54%
	2000	234806	242400	7594	3.23%
	4000	242054	249095	7041	2.91%
16	1000	177367	183127	5760	3.25%
	2000	178814	182350	3536	1.98%
	4000	183012	185463	2450	1.34%
24	1000	114222	117796	3573	3.13%
	2000	115706	119159	3454	2.98%
	4000	121673	125140	3467	2.85%
32	1000	93388	94679	1291	1.38%
	2000	93837	94600	763	0.81%
	4000	98154	98563	409	0.42%

Verificamos, nos dados obtidos nos experimentos realizados com a aplicação para o cálculo de π e apresentados na Tab. 4.6, que a sobrecarga observada na execução não foi, em qualquer dos casos, superior a 3,54% e, na maioria (55%) dos experimentos foi inferior a 2,25%. A mediana dos valores obtidos é 2,11% e a porcentagem média de sobrecarga para todos os casos foi de 2,03%. Nota-se uma tendência de queda na sobrecarga quando se aumenta o número de tarefas, o que sugere que a ferramenta de monitoramento não interfere na escalabilidade do sistema JoiN para a execução desta

classe de aplicações ($R_{cc} >> 1$), que são as aplicações que melhor utilizam os recursos do sistema JoiN.

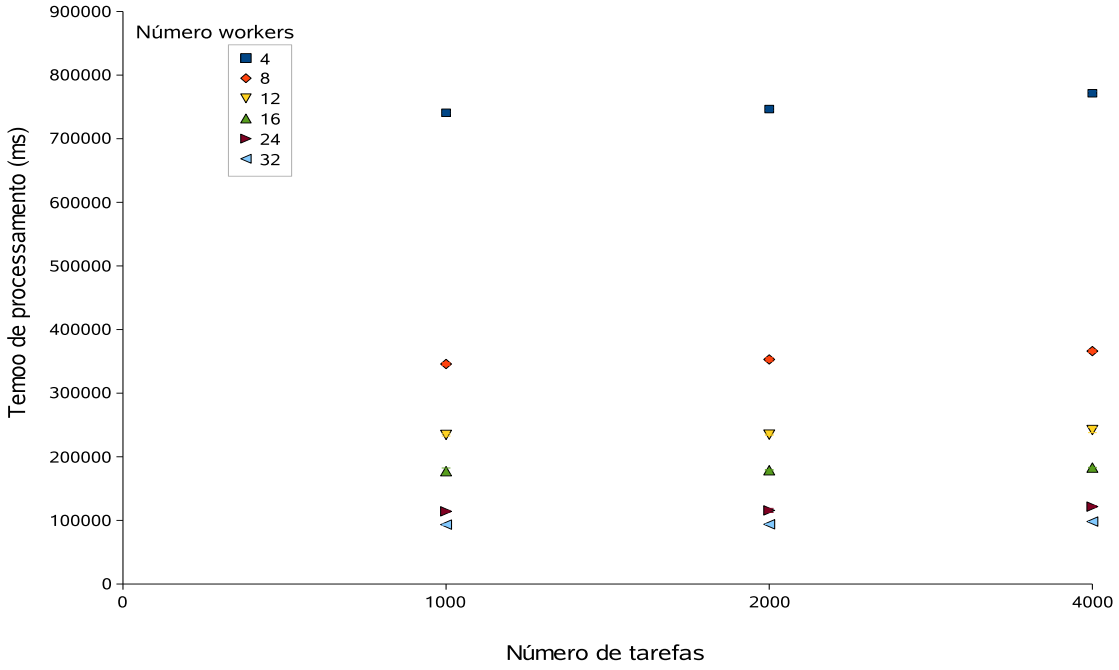

Fig. 4.4: Cálculo de π sem monitoramento - Tempo médio de processamento (ms)

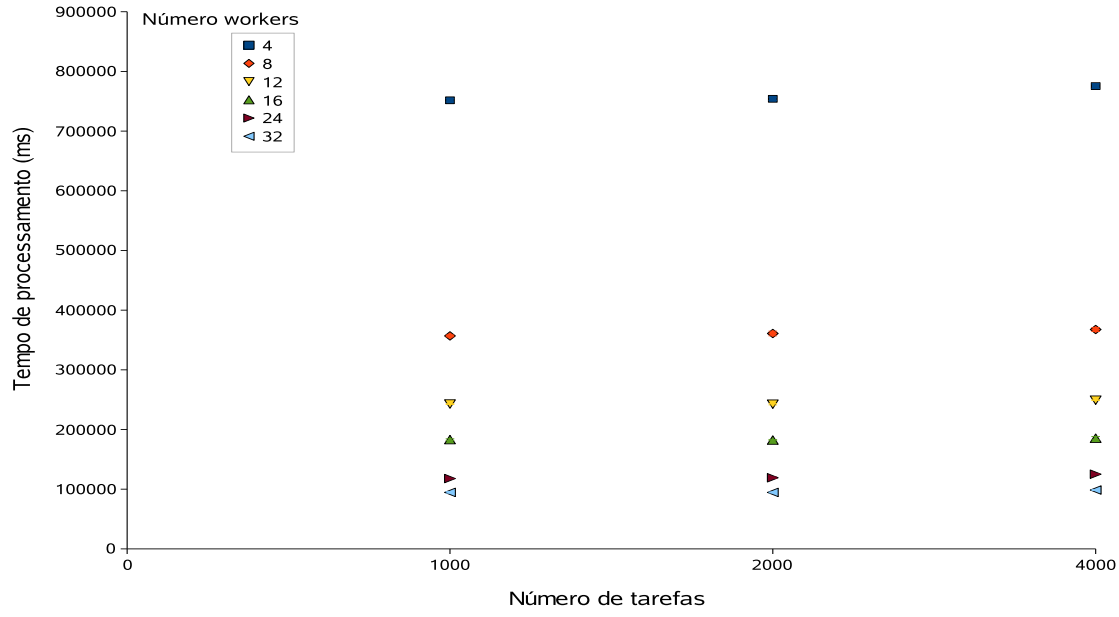

Fig. 4.5: Cálculo de π com monitoramento - Tempo médio de processamento (ms)

Nos gráficos da Fig. 4.4 e da Fig. 4.5 são representados os valores médios do tempo de processamento obtidos nos experimentos com a aplicação para cálculo de π e os respectivos intervalos de confiança (estes valores são apresentados na Tab. D.3 do Apêndice D). Os intervalos de confiança, que nos indicam se as médias obtidas são boas estimativas dos valores médios, não podem ser facilmente visualizados nestas figuras, pois seus valores são muito pequenos quando comparados aos tempos de processamento representados. Desta forma, para permitir a observação de parte dos intervalos de confiança, no gráfico da Fig. 4.6 são apresentados os valores obtidos nos experimentos com e sem monitoramento usando 4 componentes *worker*. Nota-se que, para 4000 tarefas, temos uma intersecção de intervalos de confiança, o que indica um empate técnico entre os tempos obtidos com e sem a ferramenta de monitoramento. Ou seja, neste caso, o monitoramento está causando um impacto inferior aos erros de medida. Este empate técnico ocorre também em tempos medidos sob outras condições, confirmando o baixo impacto causada pela ferramenta de monitoramento, como será discutido mais a frente.

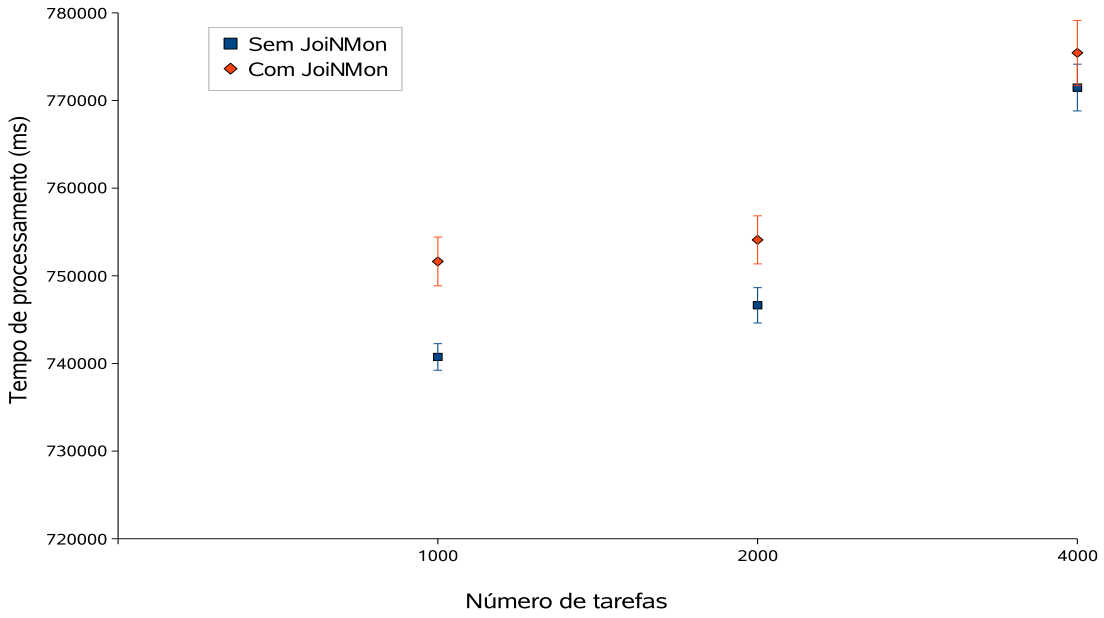

Fig. 4.6: Cálculo de π com 4 *workers* - Tempo médio de processamento (ms) com e sem monitoramento

A Fig. 4.7 apresenta um gráfico onde pode ser observado que a sobrecarga imposta pela ferramenta de monitoramento não é significativa para a execução deste tipo de aplicação no sistema JoiN. Nota-se ainda a diminuição desta sobrecarga à medida que mais componentes *worker* são adicionados ao sistema. Entretanto, a análise dos gráficos da Fig. 4.8 mostra que a sobrecarga relativa varia mas não apresenta tendência clara de queda quando se adiciona *workers*.

No primeiro gráfico da Fig. 4.8 (sobrecarga em ms) podemos observar que a sobrecarga absoluta tende a cair quando aumentamos o número de componentes *worker*. Entretanto, ao analisar o segundo gráfico da Fig. 4.8 (sobrecarga em %) podemos observar que as três curvas apresentam comportamento similar, e não refletem a tendência de queda observada anteriormente, pois os valores

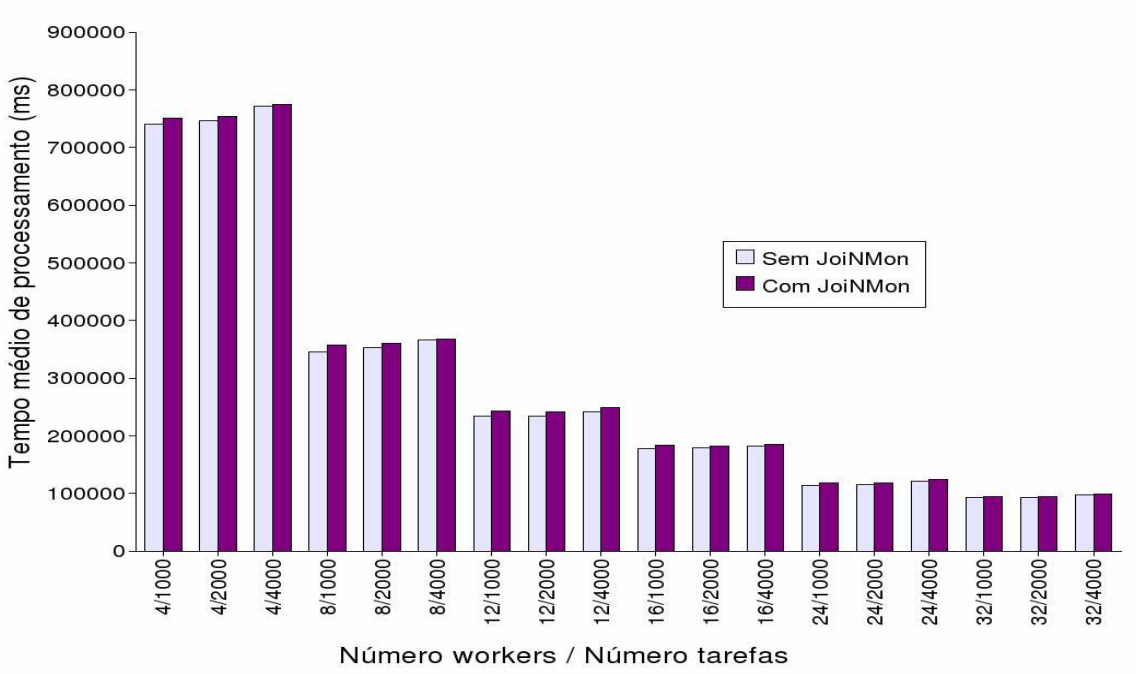

Fig. 4.7: Cálculo de π - Tempo médio de processamento (ms) com e sem monitoramento

apresentados oscilam em torno de percentuais baixos.

Na Fig. 4.9 podemos observar uma regularidade maior na forma das seis curvas de cada um dos gráficos. Verificamos que a sobrecarga diminui com o aumento do número tarefas. Isto acontece devido à eficiência na produção, na transmissão e no tratamento de eventos, e porque o "custo fixo" da ferramenta de monitoramento, representado pelo armazenamento das informações no banco de dados, ocorre no componente *server* e interfere pouco no processamento das aplicações, e se dispersa com o aumento do número de tarefas.

Estas figuras reforçam a conclusão de que JoiNMon não interfere na escalabilidade do sistema JoiN para a execução de aplicações da classe ($R_{cc} >> 1$).

Impacto na aplicação para multiplicação de matrizes

Verificamos, nos dados obtidos nos experimentos realizados com a aplicação para multiplicação de matrizes e apresentados na Tab. 4.7, que a sobrecarga observada na execução não foi, em qualquer dos casos, superior a 5,52% e, na maioria (72%) dos experimentos foi inferior a 3,81%. A mediana dos valores obtidos é 3,47% e a porcentagem média de sobrecarga para todos os casos foi de 3,54%. Também nestes experimentos há tendência de queda na sobrecarga do tempo de processamento quando se aumenta o número de tarefas. Isto pode ser verificado na maior parte das configurações utilizadas, o que sugere que a ferramenta de monitoramento também não interfere na escalabilidade do sistema JoiN para a execução desta classe de aplicações ($R_{cc} > 1$).

Nos gráficos da Fig. 4.10 e da Fig. 4.11 são representados os valores médios do tempo de processamento obtidos nos experimentos com a aplicação para multiplicação de matrizes e os respectivos intervalos de confiança (estes valores são apresentados na Tab. D.4 do Apêndice D). Nestas figuras

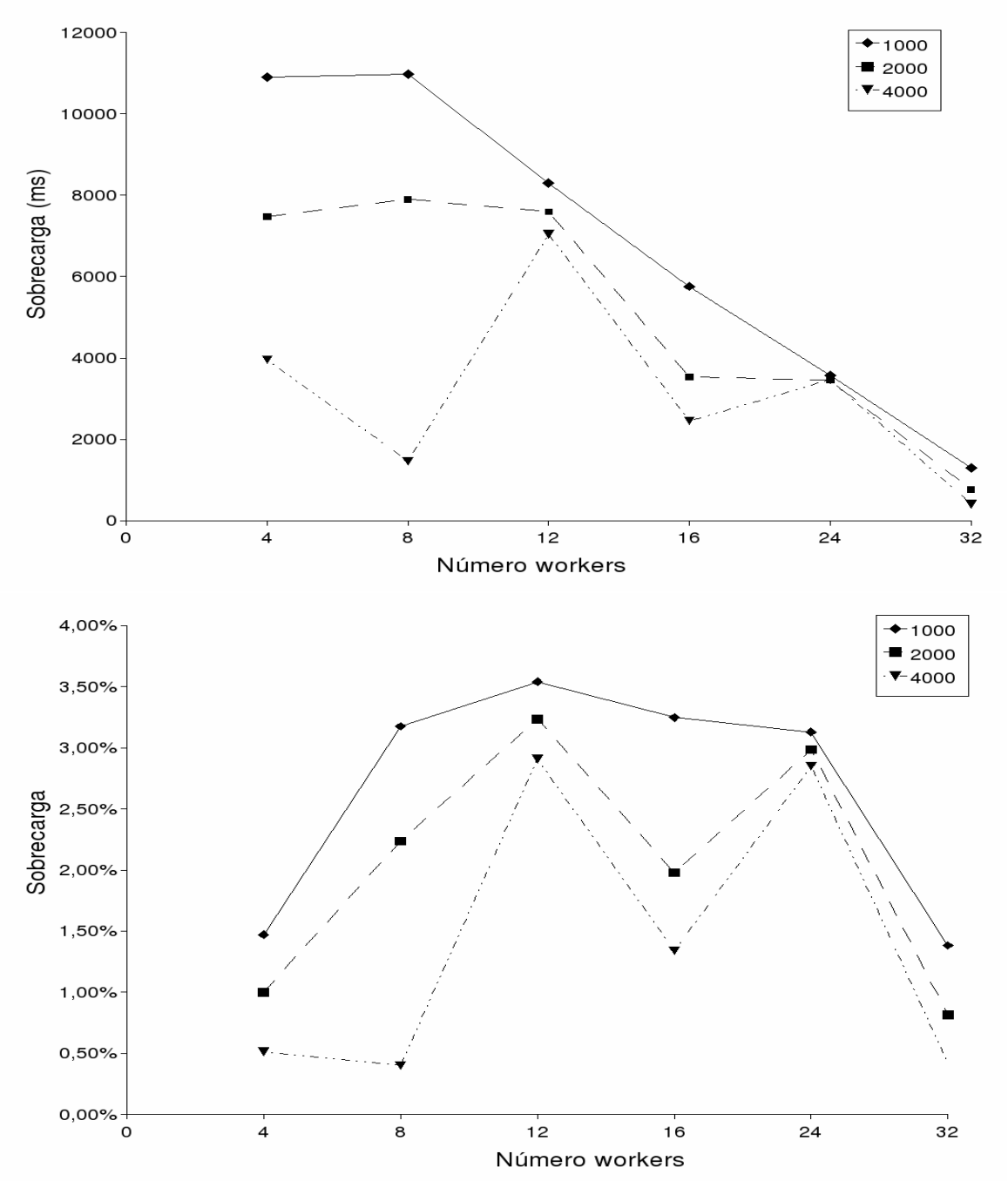

Fig. 4.8: Cálculo de π - Sobrecarga tempo médio processamento (em função do número de *workers*)

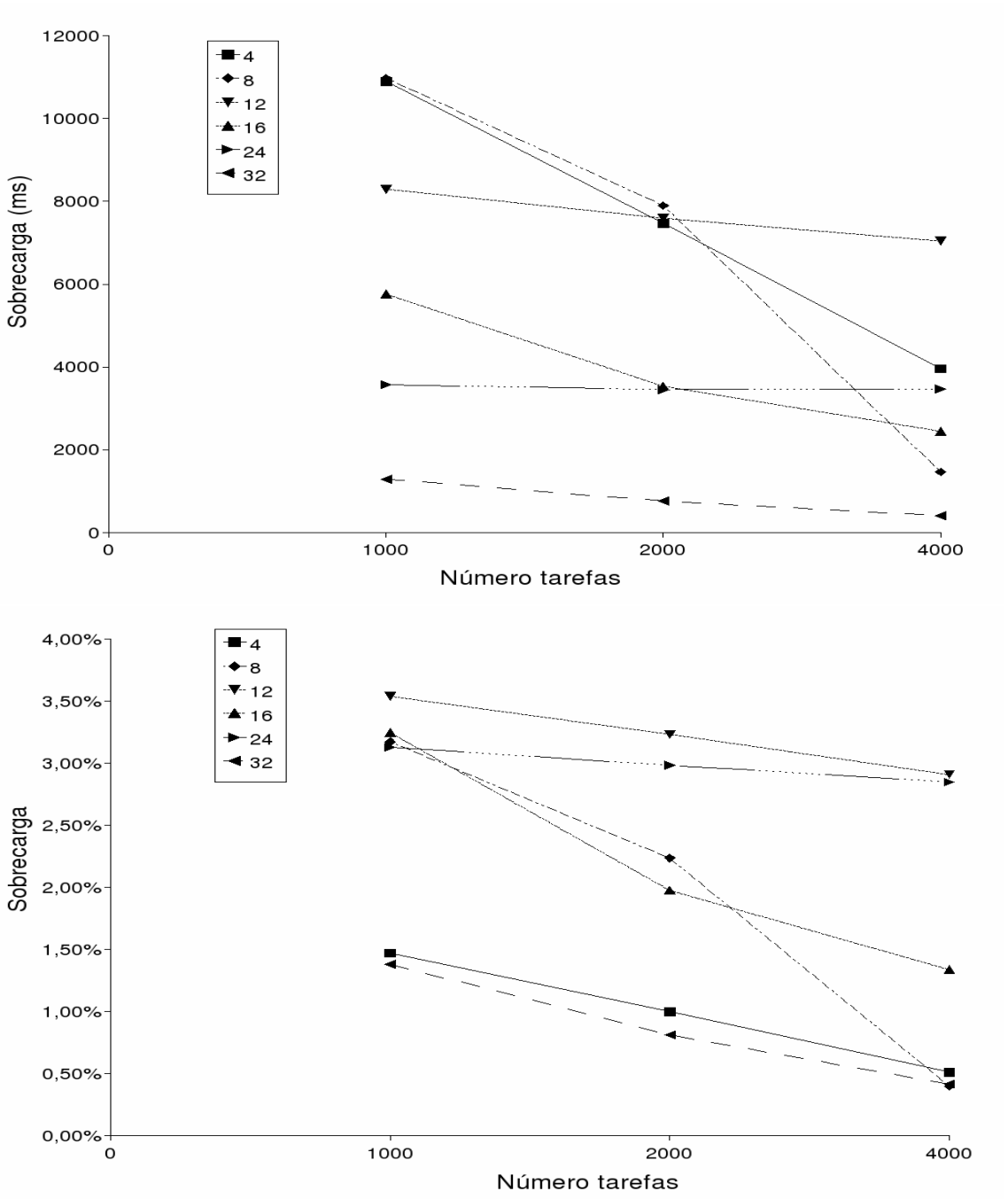

Fig. 4.9: Cálculo de π - Sobrecarga tempo médio processamento (em função do número de tarefas)

Tab. 4.7: Multiplicação de matrizes - Tempo médio de processamento (ms)

Número *workers*	Número tarefas	JoiNMon		Comparativo	
		Inativo	Ativo	Diferença	Sobrecarga
4	100	117918	121902	3984	3,38%
	200	120471	124219	3749	3,11%
	400	125064	128819	3755	3,00%
8	100	62373	64013	1641	2,63%
	200	65426	67069	1643	2,51%
	400	73941	75260	1319	1,78%
12	100	49414	51422	2008	4,06%
	200	53150	55032	1882	3,54%
	400	60408	62685	2277	3,77%
16	100	43707	45589	1882	4,31%
	200	47453	49255	1801	3,80%
	400	57591	59549	1958	3,40%
24	100	35776	36816	1040	2,91%
	200	38708	40847	2139	5,52%
	400	45068	46713	1645	3,65%
32	100	30078	31523	1446	4,81%
	200	31130	32142	1013	3,25%
	400	36977	38600	1623	4,39%

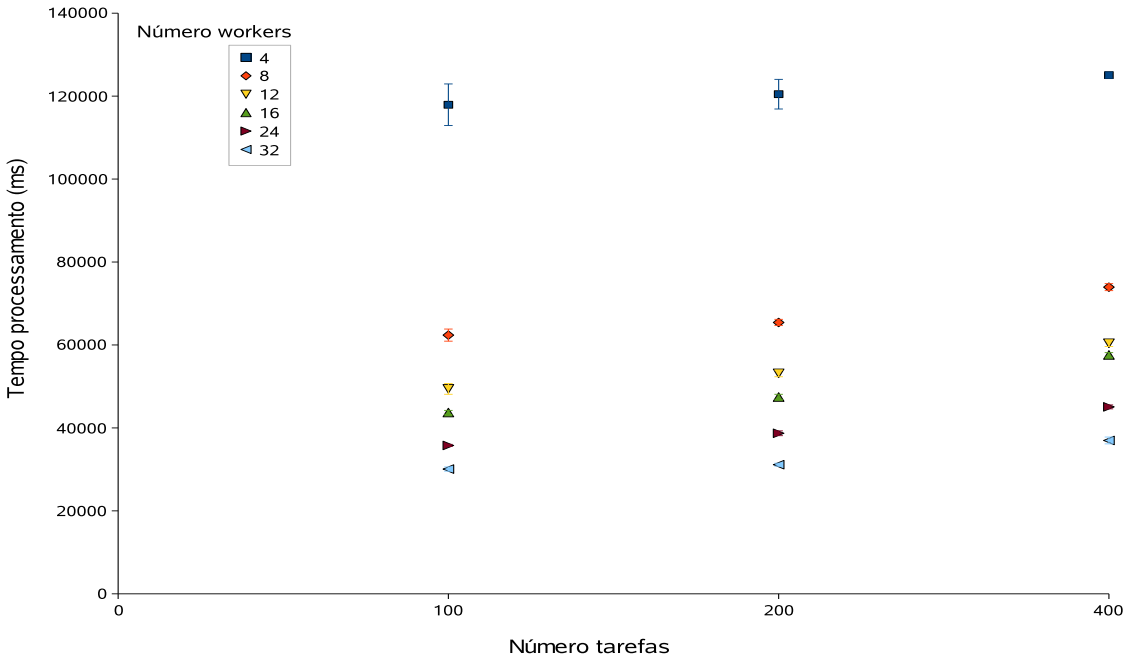

Fig. 4.10: Multiplicação de matrizes sem monitoramento - Tempo médio de processamento (ms)

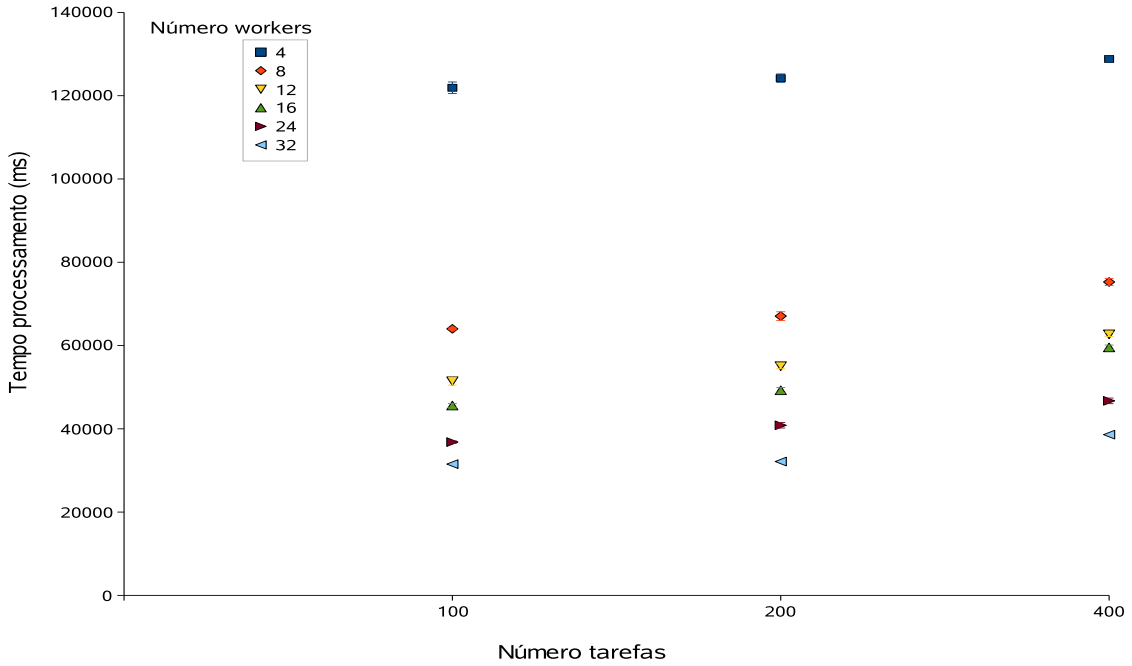

Fig. 4.11: Multiplicação de matrizes com monitoramento - Tempo médio de processamento (ms)

podem ser mais facilmente observadas as representações de alguns dos intervalos de confiança. Isto é possível porque os valores dos intervalos de confiança, apesar de serem pequenos, não são insignificantes quando comparados aos tempos de processamento obtidos nos experimentos. Também neste

caso, para permitir a melhor observação dos intervalos de confiança e possibitar a análise comparativa entre os valores obtidos sem monitoramento e com a ferramenta de monitoramento ativada, no gráfico da Fig. 4.12 são apresentados os valores obtidos nos experimentos usando 4 componentes *worker*. Assim como ocorreu com o cálculo de π, aqui também existe um empate técnico entre os tempos com e sem monitoramento para 100 e 200 tarefas, o que mostra que o impacto introduzido pela ferramenta de monitoramento, neste caso, é inferior ao erro das medidas de tempo. Uma análise mais detalhada dos tempos em outras condições de processamento revela outros empates técnicos, mostrando o baixo impacto da ferramenta de monitoramento nestes casos também.

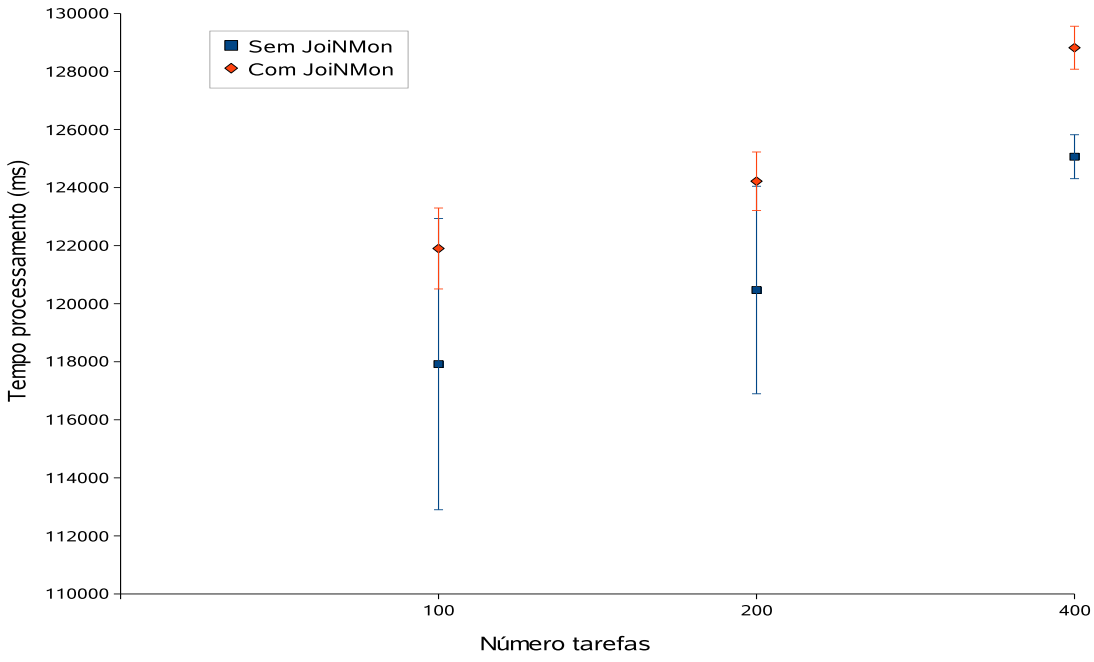

Fig. 4.12: Multiplicação de matrizes com 4 *workers* - Tempo médio de processamento (ms) com e sem monitoramento

A Fig. 4.13 apresenta um gráfico onde pode ser observado que a sobrecarga imposta pela ferramenta de monitoramento não é significativa em relação ao tempo total de processamento.

No primeiro gráfico da Fig. 4.14 (sobrecarga em ms) observamos que, na maioria dos casos, a sobrecarga absoluta tende a cair quando aumentamos o número de componentes *worker*. As maiores sobrecargas absolutas, observadas neste gráfico nos experimentos com 4 componentes *worker* e que parecem ser muito grandes, são proporcionais aos tempos médios de processamento medidos nestes experimentos, o que pode ser visto no segundo gráfico desta mesma figura. Nos experimentos com 8 componentes *worker* seriam esperados valores entre aqueles obtidos nos experimentos com 4 e com 12 componentes *worker*. Creditamos esta aparente inconsistência ao melhor desempenho dos computadores utilizados durante a execução dos experimentos com 8 componentes *worker*, e isto se deve à variação de carga destes computadores que não foram utilizados de forma dedicada durante os experimentos. A visão de que a sobrecarga absoluta tende a cair quando aumentamos o número de componentes *worker* é ilusória, pois os tempos médios de processamento também estão diminuindo, como pode ser visto na Tab. 4.7. Esta ilusão é confirmada pela análise do segundo gráfico da Fig. 4.14

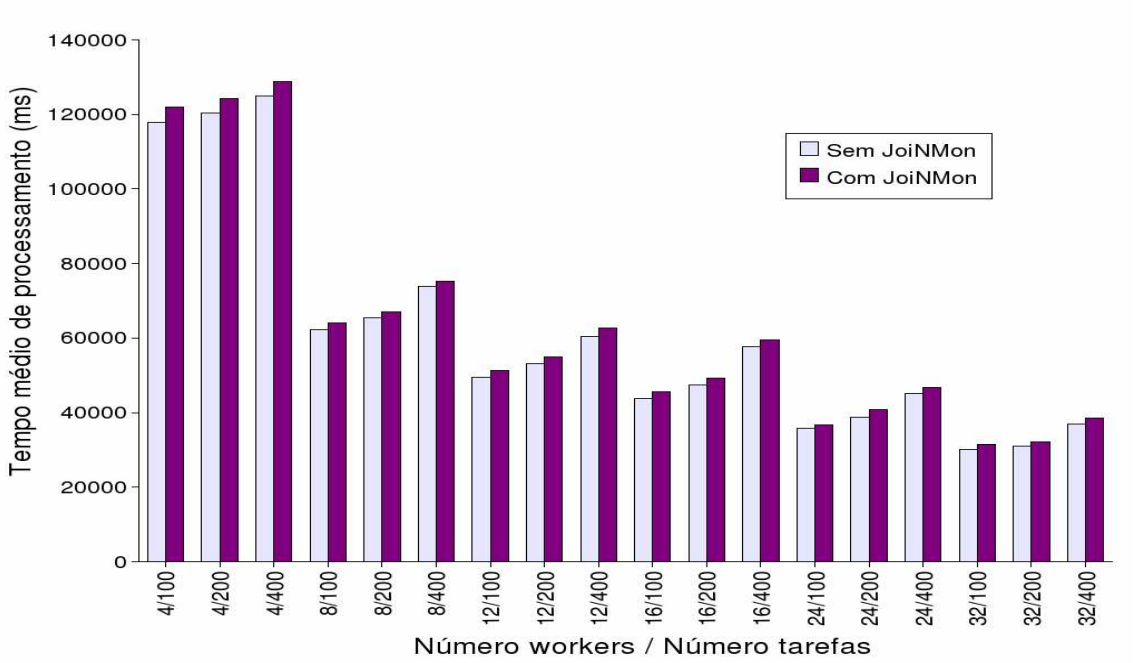

Fig. 4.13: Multiplicação de matrizes - Tempo médio de processamento (ms) com e sem monitoramento

(sobrecarga em %). Neste gráfico podemos observar que as três curvas apresentam comportamento similar onde, em geral, o percentual da sobrecarga aumenta conforme aumentamos o número de componentes *worker*. Isto ocorre porque, no caso da aplicação para multiplicação de matrizes, o aumento do número de *workers* é acompanhado pelo aumento do tráfego de rede, o que será discutido mais detalhadamente na seção 4.4.2.

No primeiro gráfico da Fig. 4.15 podemos observar que a sobrecarga absoluta se manteve estável e isto se deve ao fato de não ter havido uma redução significativa no tempo de processamento quando aumentamos o número de componentes *worker* (como pode ser visto na Tab. 4.7 e na Fig. 4.13). No segundo gráfico desta figura podemos observar uma regularidade na forma das curvas relativas a 4 das 6 configurações utilizadas (nos experimentos com 4, 8, 12 e 16 componentes *worker*) e, nestes casos, uma clara tendência de queda na sobrecarga relativa à medida que aumentamos o número tarefas. Nestes casos é observado um ruído na medida obtida nos experimentos com 12 componentes *worker* e 400 tarefas, que podemos desprezar por ser pequeno. As curvas relativas aos experimentos com 24 e 32 componentes *worker* não são regulares e atribuímos esta irregularidade à dificuldade de obtermos valores precisos nestes casos, onde o tempo de processamento é pequeno e, portanto, mais sujeito a distorções. Também para a aplicação para multiplicação de matrizes, como analisado no caso da aplicação para cálculo de π, a tendência de queda na sobrecarga relativa à medida que aumentamos o número tarefas, ocorre devido à forma eficiente com que os eventos são produzidos, transmitidos, consumidos e armazenados. O armazenamento das informações no banco de dados feito no componente *server* pouco influencia o tempo de processamento das aplicações pois é diluído com o aumento do número de tarefas.

Com base nestas figuras, a sobrecarga relativa que JoiNMon acarreta no tempo de execução de

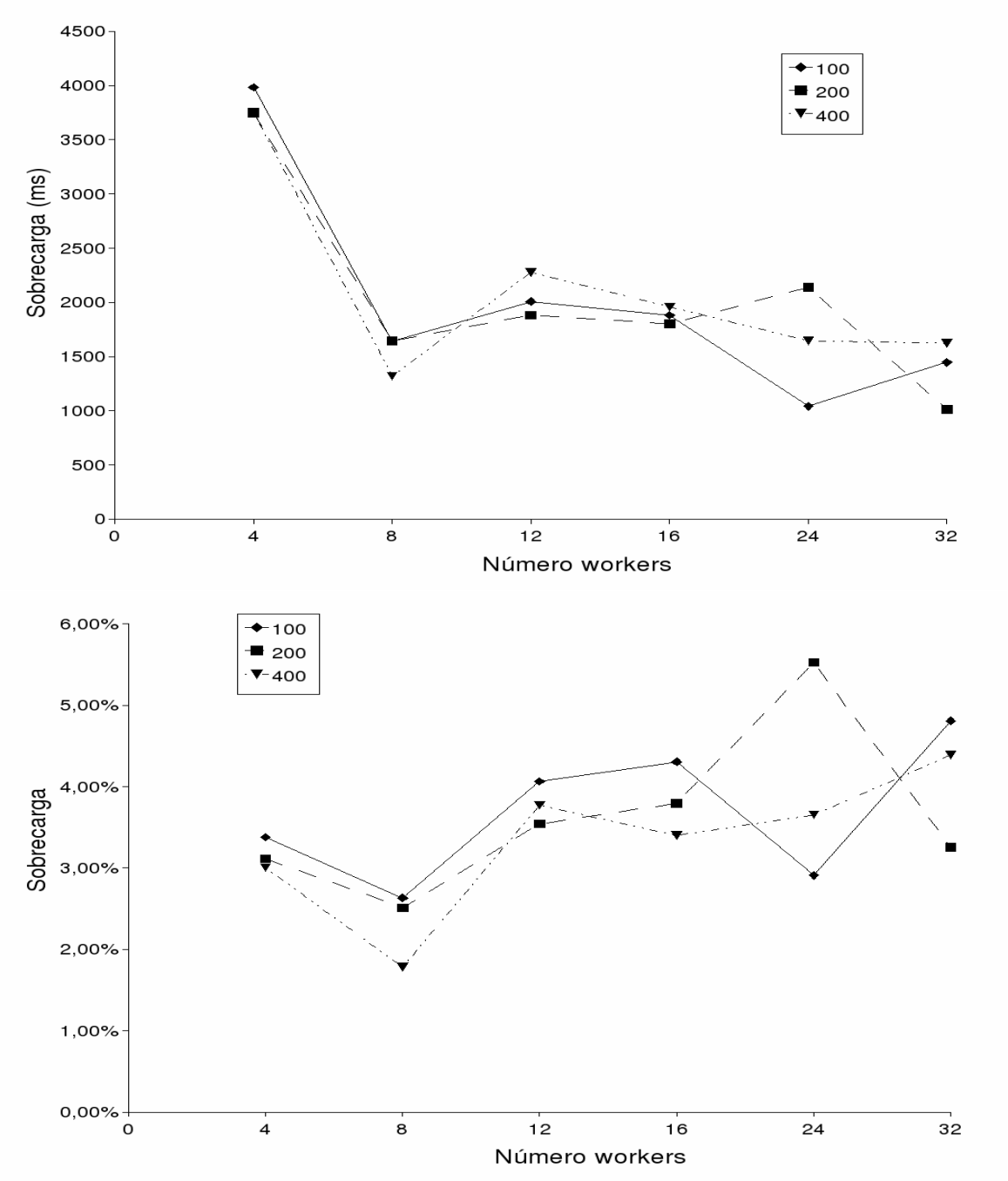

Fig. 4.14: Multiplicação de matrizes - Sobrecarga no tempo médio de processamento (em função do número de *workers*)

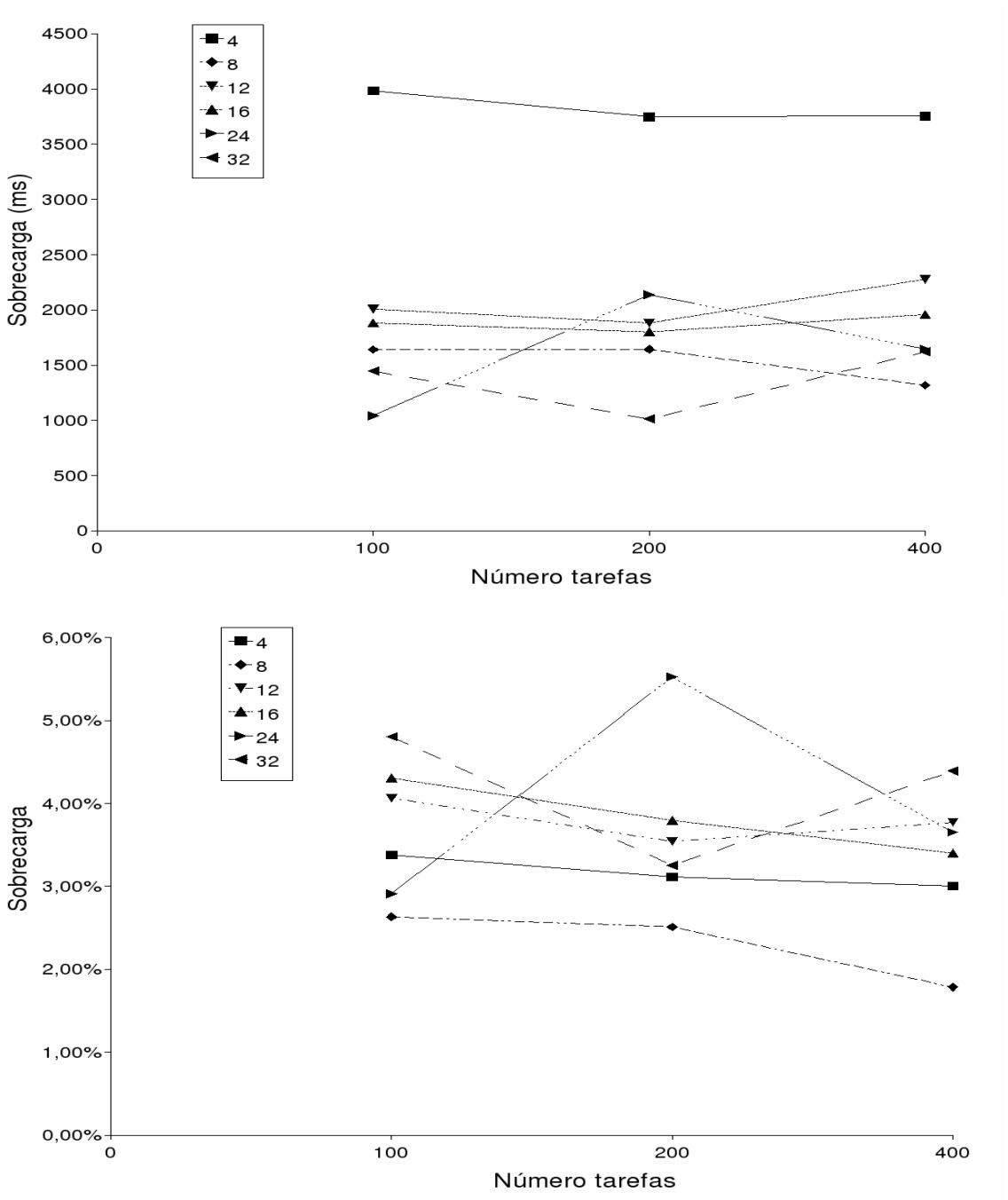

Fig. 4.15: Multiplicação de matrizes - Sobrecarga no tempo médio de processamento (em função do número de tarefas)

aplicações da classe ($R_{cc} > 1$) aumenta à medida que aumentamos o número de componentes *worker*, como observamos no segundo gráfico da Fig. 4.14. Entretanto, a observação do outro gráfico desta mesma figura, onde apresentamos o comportamento da sobrecarga absoluta, sugere que a ferramenta JoiNMon, apesar de interferir na eficiência do sistema JoiN, permite que seja mantida uma certa escalabilidade para a execução desta classe de aplicações.

Considerações sobre a avaliação do impacto no desempenho

Podemos constatar que, para a faixa de valores usados nestes experimentos, a ferramenta JoiNMon não introduziu gargalos no sistema e suportou o aumento do número de recursos (componentes *worker*) e de eventos (tarefas); em outras palavras, a implementação da ferramenta JoiNMon não aumentou significativamente o consumo de recursos disponíveis e, com isto, as aplicações não sofreram alterações perceptíveis no seu desempenho em função do monitoramento.

4.4.2 Avaliação do impacto no tráfego de rede

Durante a execução dos experimentos foram capturadas informações sobre o tráfego nas portas utilizadas pelo sistema JoiN e pela ferramenta de monitoramento. Esta captura foi feita por meio da ativação da ferramenta Wireshark [105] no computador utilizado para a execução dos componentes *server* e *coordinator*. A ferramenta Wireshark permite, além da captura das informações, a seleção e o resumo dos dados a serem analisados. As portas monitoradas foram:

- TCP/8000: utilizada para comunicação entre os componentes *server* e *coordinator*;
- TCP/8100: utilizada para comunicação entre os componentes *coordinator* e *workers* a ele associados;
- TCP/9001: utilizada para acesso ao servidor de banco de dados (HSQLDB).

As portas TCP/8000 e TCP/8100 são as portas default de JoiN. A porta TCP/9001 é a porta default de HSQLDB.

Tráfego gerado pela ferramenta de monitoramento

Os eventos, onde estão contidas as informações que são monitoradas pela ferramenta JoiNMon, são gerados pelos produtores e consumidos pelo serviço JoiNMonitor. Como apresentado em 3.7.3, este serviço é composto pelos módulos coletor e agrupador. O primeiro coleta e trata as informações em cada uma das máquinas, o segundo as agrupa, conforme a hierarquia do sistema JoiN.

- Tráfego na porta TCP/8100:

  o módulo coletor executado nos componentes *worker* envia suas informações para o módulo agrupador correspondente, executado no componente *coordinator* ao qual este *worker* está associado. Para isto utiliza o serviço *Communicator* do sistema JoiN, e gera tráfego de rede na porta TCP/8100.

- Tráfego na porta TCP/8000:

  o módulo agrupador executado nos componentes *coodinator*, além de agrupar as informações, executa a transferência de dados entre os componentes *coodinator* e *server*, por meio do serviço *Communicator*, e gera tráfego de rede na porta TCP/8000.

- Tráfego na porta TCP/9001:

  o módulo agrupador executado no componente *server* agrupa e organiza todas as informações coletadas pela ferramenta de monitoramento e as armazena na base de dados. A transferência de informações entre o serviço JoiNMonitor e o servidor de banco de dados é feito por meio da porta TCP/9001.

Dados obtidos nos experimentos com a aplicação para cálculo de π

O tráfego de dados para transmissão de informações entre as tarefas que processam esta aplicação é muito pequeno, pois cada tarefa recebe um número como parâmetro (a quantidade de pontos que a tarefa deve sortear) e devolve um número como resultado (o número de pontos sorteados dentro de uma circunferência). Para esta classe de aplicações ($R_{cc} >> 1$) é esperado que a ferramenta de monitoramento introduza a maior sobrecarga relativa no tráfego de rede.

Na Tab. 4.8 são exibidos os dados relativos ao tráfego de rede total das portas TCP/8000, TCP/8100 e TCP/9001 obtidos nos experimentos realizados com a aplicação para o cálculo de π pelo método de Monte Carlo.

São apresentados resumos dos dados obtidos, subdivididos conforme a etapa: a primeira, na coluna *Sem JoiNMon*, feita com o sistema JoiN sem a ferramenta de monitoramento; e a segunda com a ferramenta JoiNMon ativada, na coluna *JoiNMon ativo*. Para cada uma destas etapas, é apresentado o valor médio, em bytes, do tráfego de rede observado nestas portas durante a execução da aplicação, capturado para as configurações do sistema JoiN apresentadas na Tab. 4.4. Nas duas últimas colunas são apresentadas a diferença entre os valores médios de bytes trafegados e a sobrecarga que a ferramenta de monitoramento acarreta no tráfego de rede durante a execução desta aplicação.

O volume de tráfego observado durante a execução desta aplicação é baixo e, por este motivo, as medições são mais sujeitas a imprecisões. Mesmo assim, os valores de coeficiente de variação destes experimentos, apresentados nas tabelas Tab. D.6 e Tab. D.7, e discutidos na na seção D.4.1 do Apêndice D, são relativamente pequenos.

Podemos observar na Tab. 4.8 que a sobrecarga no tráfego de rede introduzida pela ferramenta de monitoramento durante os diversos experimentos conduzidos com a aplicacação para cálculo de π não foi inferior a 116,75%, ou seja, o tráfego foi mais que duplicado. A maior sobrecarga observada foi de 155,32%, que corresponde a um tráfego total de 2,5 vezes o original (sem monitoramento). O valor médio da sobrecarga introduzida foi de 139,30%.

Estes valores parecem sugerir uma sobrecarga muito grande no tráfego de rede. Se considerarmos que, no contexto desta aplicação, o tráfego original é pequeno, podemos avaliar como satisfatórios os resultados obtidos uma vez que, como vimos em 4.4.1, a ferramenta de monitoramento não introduziu sobrecarga superior a 3,55% no tempo total de execução da aplicação, que inclui o tempo para comunicação. Como (R_{cc} = *tempo_de_computação / tempo_de_comunicação* $>> 1$), se dobrarmos o tempo gasto em comunicação, ainda assim a execução da aplicação para cálculo de π no sistema JoiN apresenta boa relação custo/benefício.

Tab. 4.8: Cálculo de π - Tráfego total (bytes) pelas portas TCP 8000, 8100 e 9001

Num. workers	*Num. tasks*	*Sem JoiNMon*	*JoiNMon ativo*	*Comparativo*	
		Média (bytes)	*Média (bytes)*	*Diferença*	*Sobrecarga*
4	1000	2299257	5111225	2811968	122,30%
	2000	4427220	9770586	5343365	120,69%
	4000	7768649	19052709	11284059	145,25%
8	1000	2226505	5053188	2826682	126,96%
	2000	3902450	9600096	5697646	146,00%
	4000	7098767	18124585	11025818	155,32%
12	1000	2281480	4945212	2663732	116,75%
	2000	3729910	9300777	5570866	149,36%
	4000	7208269	18019664	10811394	149,99%
16	1000	2164455	4947631	2783175	128,59%
	2000	3715807	9296602	5580795	150,19%
	4000	7030726	17287309	10256583	145,88%
24	1000	2074745	4763200	2688455	129,58%
	2000	3708900	8834883	5125983	138,21%
	4000	6928067	17532965	10604897	153,07%
32	1000	2138950	4949733	2810783	131,41%
	2000	3703673	9087978	5384305	145,38%
	4000	6832446	17258274	10425827	152,59%

Entretanto, como as aplicações da classe $R_{cc} >> 1$ não necessariamente apresentam um volume de tráfego de dados insignificante, é possível termos um impacto relativo reduzido da ferramenta de monitoramento no tráfego de rede se houver um certo volume de dados sendo transmitido, já que o tráfego extra de JoiNMon pode ser considerado estável para um mesmo número de tarefas.

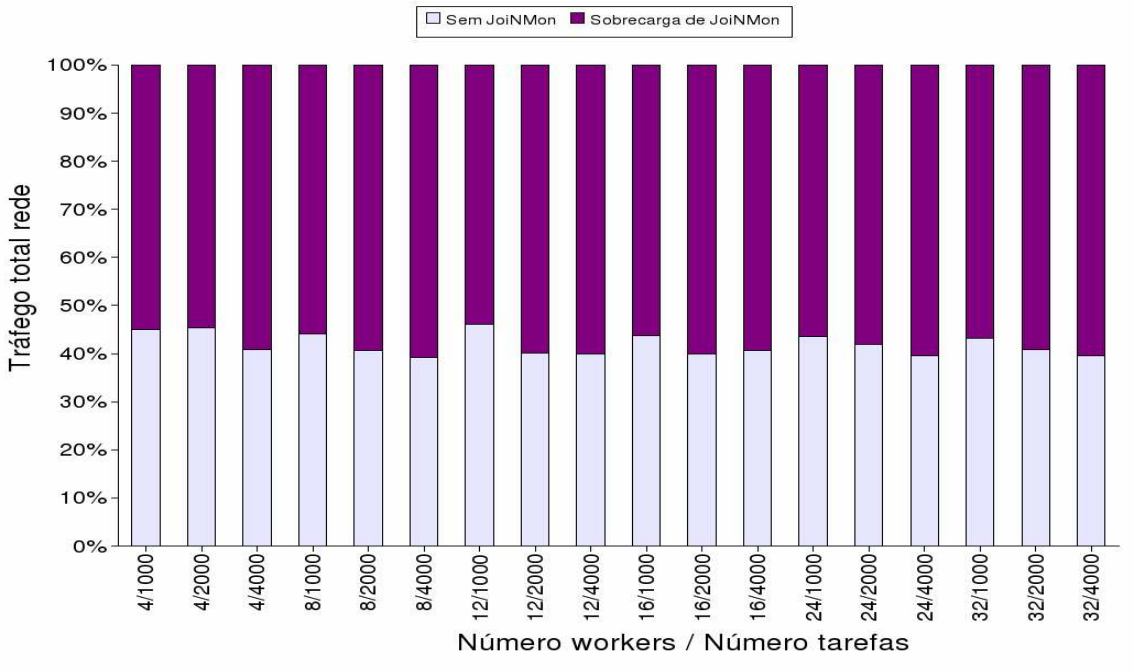

Fig. 4.16: Cálculo de π - Sobrecarga no tráfego de rede

A Fig. 4.16 apresenta um gráfico onde pode ser observado que o tráfego de rede gerado pela ferramenta na execução desta aplicação está compreendido no intervalo de 55% a 60% do tráfego total observado durante a execução de uma aplicação da classe $R_{cc} >> 1$ no sistema JoiN.

Na Fig. 4.17 e na Fig. 4.18 são apresentados gráficos referentes aos experimentos realizados com a aplicação para cálculo de π, onde são representadas as curvas de sobrecarga no tráfego de rede acumulado nas portas TCP 8000, 8100 e 9001.

No primeiro gráfico da Fig. 4.17 observamos que a sobrecarga absoluta (em bytes) permanece quase constante quando aumentamos o número de componentes *worker*. Este resultado é muito interessante e sugere que a ferramenta de monitoramento deve causar sobrecarga absoluta semelhante mesmo quando o sistema JoiN for utilizado com um número muito grande de componentes *worker*. No segundo gráfico desta mesma figura podemos observar que as três curvas apresentam comportamento similar, e que a sobrecarga da ferramenta JoiNMon no tráfego de rede oscila entre 115% e 155%.

No primeiro gráfico da Fig. 4.18 podemos observar que as seis curvas da sobrecarga (em bytes) praticamente se sobrepõem. Independentemente do número de componentes *worker* utilizados, a sobrecarga cresce na mesma proporção que o número de tarefas. Isto se deve ao tráfego relativo aos eventos para sinalização do início e término de cada tarefa e para manutenção destas informações na base de dados. No segundo gráfico desta figura podemos observar que a sobrecarga relativa aumenta com o aumento do número tarefas e de componentes *worker*.

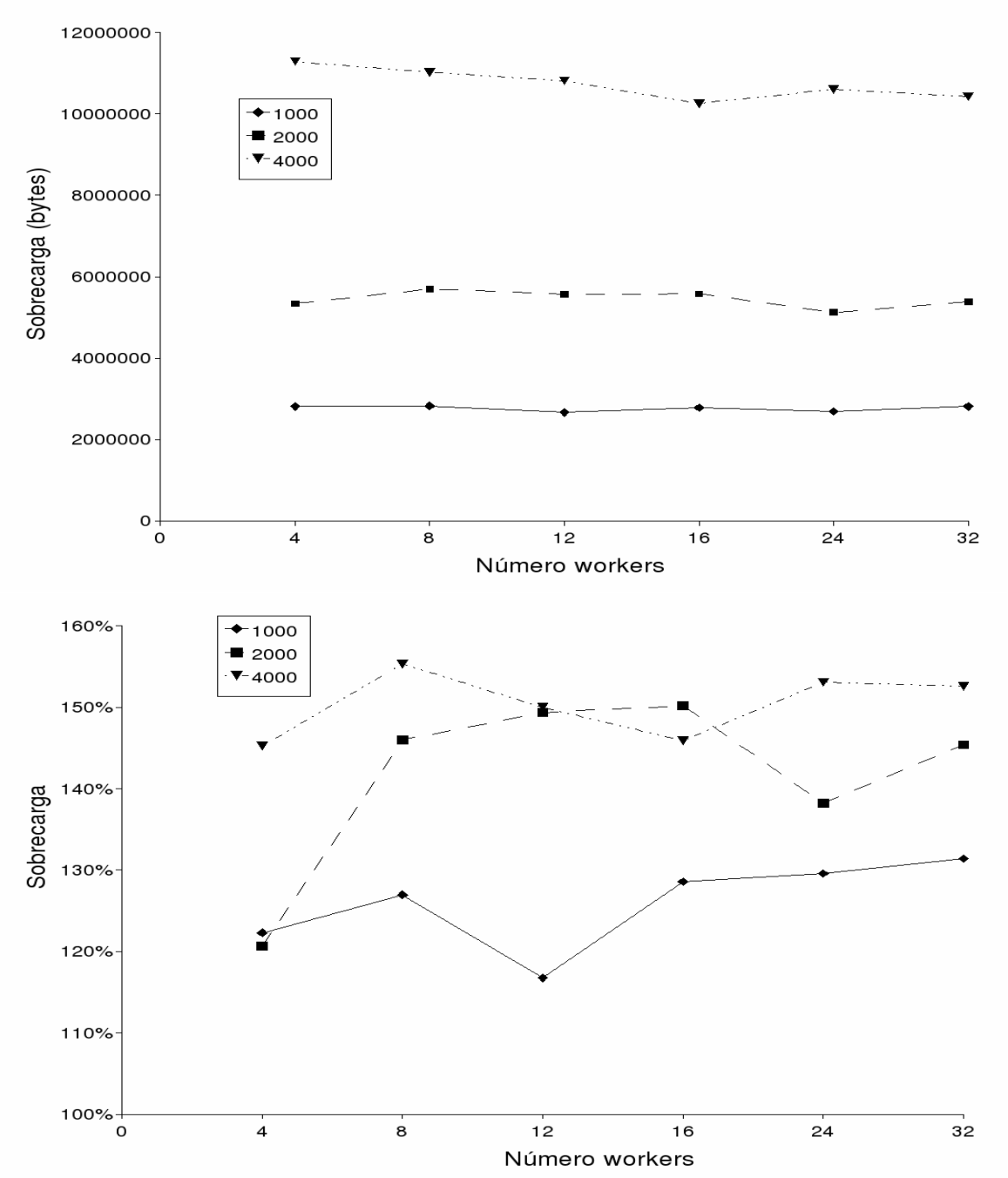

Fig. 4.17: Cálculo de π - Sobrecarga tráfego de rede total (em função do número de *workers*)

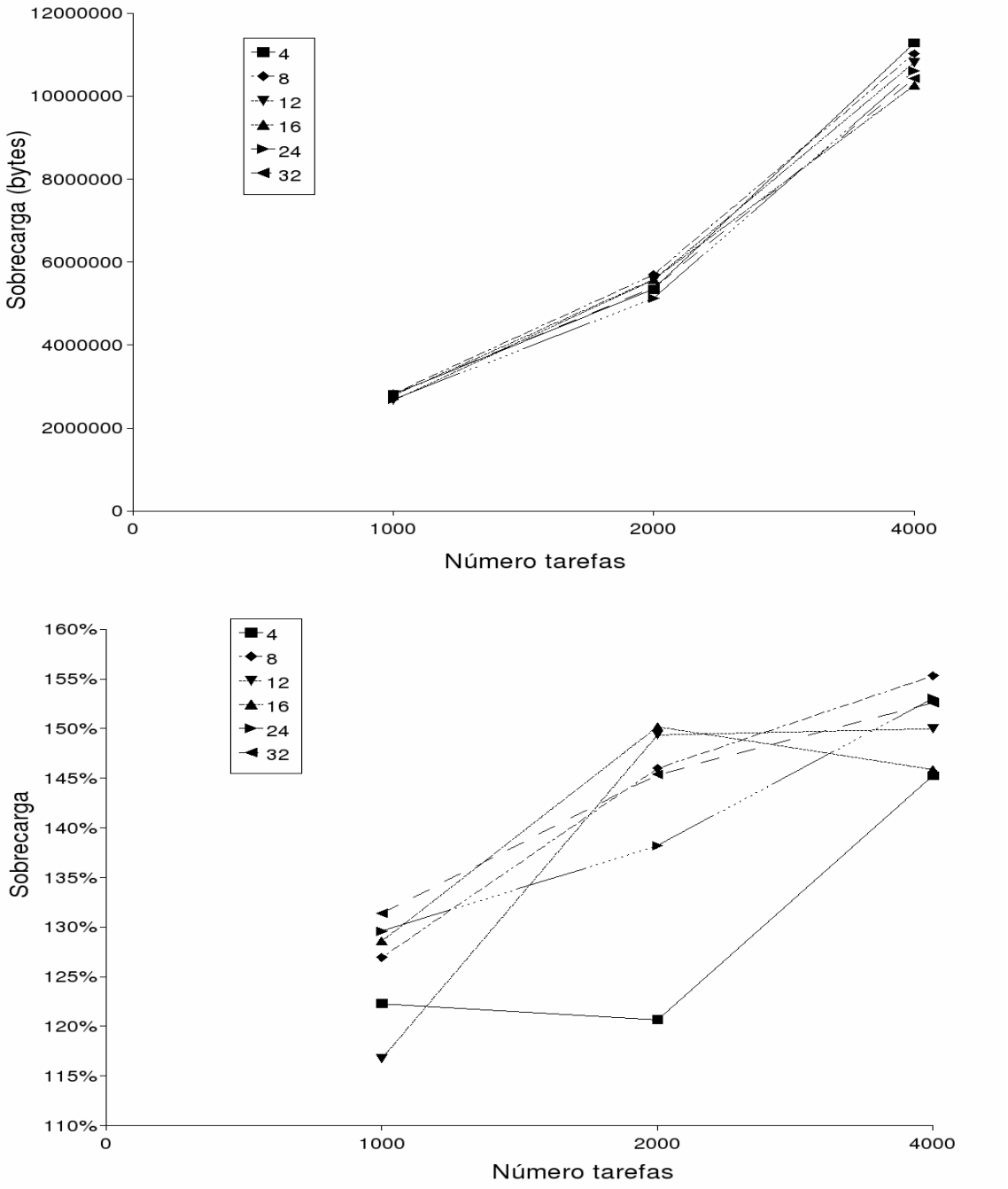

Fig. 4.18: Cálculo de π - Sobrecarga tráfego de rede total (em função do número de tarefas)

Tab. 4.9: Cálculo de π - Tráfego (bytes) em cada porta TCP com 4, 16 e 32 *workers*

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*	*JoiNMon ativo*	*Comparativo*	
			Média	*Média*	*Diferença*	*Sobrecarga*
4	1000	8000	13803	156669	142865	1035,02%
		8100	2285454	2187195	-98259	-4,30%
		9001	-	2767361	-	-
		total	2299257	5111225	2811968	122,30%
	2000	8000	13094	230704	217610	1661,91%
		8100	4414126	4065256	-348870	-7,90%
		9001	-	5474625	-	-
		total	4427220	9770586	5343365	120,69%
	4000	8000	15408	378648	363240	2357,47%
		8100	7753241	7860775	107533	1,39%
		9001	-	10813287	-	-
		total	7768649	19052709	11284059	145,25%
16	1000	8000	11601	113829	102228	881,20%
		8100	2152855	1986326	-166529	-7,74%
		9001	-	2847476	-	-
		total	2164456	4947631	2783175	128,59%
	2000	8000	12988	186175	173186	1333,41%
		8100	3702819	3645044	-57775	-1,56%
		9001	-	5465384	-	-
		total	3715807	9296603	5580795	150,19%
	4000	8000	10596	352206	341610	3224,02%
		8100	7020131	6678462	-341669	-4,87%
		9001	-	10256642	-	-
		total	7030727	17287310	10256583	145,88%
32	1000	8000	12617	105755	93138	738,22%
		8100	2126333	2086264	-40069	-1,88%
		9001	-	2757714	-	-
		total	2138950	4949733	2810783	131,41%
	2000	8000	13435	187747	174312	1297,43%
		8100	3690237	3710979	20742	0,56%
		9001	-	5189252	-	-
		total	3703673	9087978	5384305	145,38%
	4000	8000	10841	342224	331383	3056,87%
		8100	6821606	6652754	-168852	-2,48%
		9001	-	10263296	-	-
		total	6832446	17258274	10425827	152,59%

Na Tab. 4.9 são exibidos os dados relativos ao tráfego de rede separado por porta, além do tráfego total nestas portas, para os experimentos realizados com 4, 16 e 32 componentes *worker*. Estes dados são suficientes para ilustrar a análise do tráfego em cada uma das portas, pois não há variação significativa com relação ao comportamento observado nos demais experimentos. No Apêndice D podem ser encontradas todas as informações coletadas nos experimentos.

Na Tab. 4.9 é possível observar que:

- o tráfego na porta TCP/8000, pela qual são enviadas as informações coletadas e agrupadas pela ferramenta de monitoramento, aumenta da ordem de dezenas de vezes quando se ativa o monitoramento. Contudo, mesmo no caso em que este aumento é de 363240 bytes (máximo observado), o volume de tráfego não pode ser considerado um fator que venha a alterar a eficiência ou o desempenho do sistema, pois este volume de tráfego é considerado baixo e suportado com facilidade pela tecnologia de rede existente atualmente;
- o tráfego na porta TCP/9001, utilizada para acesso ao servidor de banco de dados (HSQLDB) usado pela ferramenta de monitoramento JoiNMon, é da ordem de 2700 bytes por tarefa que compõe a aplicação. Se o número de tarefas for muito grande (muitos milhares), este tráfego pode interferir no desempenho do sistema JoiN. Entretanto é possível executar o servidor de bancos de dados no mesmo computador onde é executado o componente *server* e, desta forma, transformar este tráfego em comunicação interna no mesmo computador, sem sobrecarga no tráfego de rede. Esta é uma alternativa viável visto que o componente *server* não participa do processamento das tarefas e pode cuidar do serviço de banco de dados sem maiores problemas;
- o tráfego na porta TCP/8100, utilizada para comunicação entre os componentes *coordinator* e *workers* a ele associados, diminuiu quando JoiNMon foi ativado em 7 dos 9 experimentos apresentados. Apesar de parecer um contra-senso, esta redução é justificada pelo comportamento do mecanismo de replicação da execução de tarefas ainda não concluídas do algoritmo de escalonamento do sistema JoiN, apresentado na seção 3.2.3. Os atrasos nas comunicações causados por JoiNMon estão tendo o efeito de inibir o disparo de réplicas de tarefas que já estão em execução. Para confirmar este comportamento foram realizados experimentos com o sistema JoiN configurado com apenas um componente *worker*, cujos resultados são apresentados na Tab. 4.10. A configuração com apenas um componente *worker* reduz ao mínimo a replicação de tarefas, pois este único *worker* permanece ocupado com a execução de tarefas originais durante toda a execução da aplicação e, desta forma, não são escalonadas tarefas réplicas. Nota-se neste novo experimento que o tráfego na porta TCP/8100 aumenta marginalmente quando se ativa o JoiNMon, como era esperado.

Na Fig. 4.19 são apresentados gráficos correspondentes à sobrecarga no tráfego de rede na porta TCP/8000 referentes aos resultados apresentados na Tab. 4.9. No primeiro gráfico podemos observar que a sobrecarga absoluta (em bytes) da ferramenta JoiNMon no tráfego de rede nesta porta aumenta quando aumentamos o número de tarefas. No segundo gráfico desta figura podemos observar que, para o mesmo número de tarefas, a sobrecarga diminui à medida que aumentamos o número de componentes *worker*. A porta TCP/8000 é utilizada para comunicação entre os componentes *server* e *coordinator*, e por esta porta são enviadas as informações coletadas e agrupadas pela ferramenta de monitoramento, para que sejam armazenadas no banco de dados mantido no componente *server* e, aumentando o número de tarefas, aumentamos o volume de tráfego nesta porta. Este comportamento

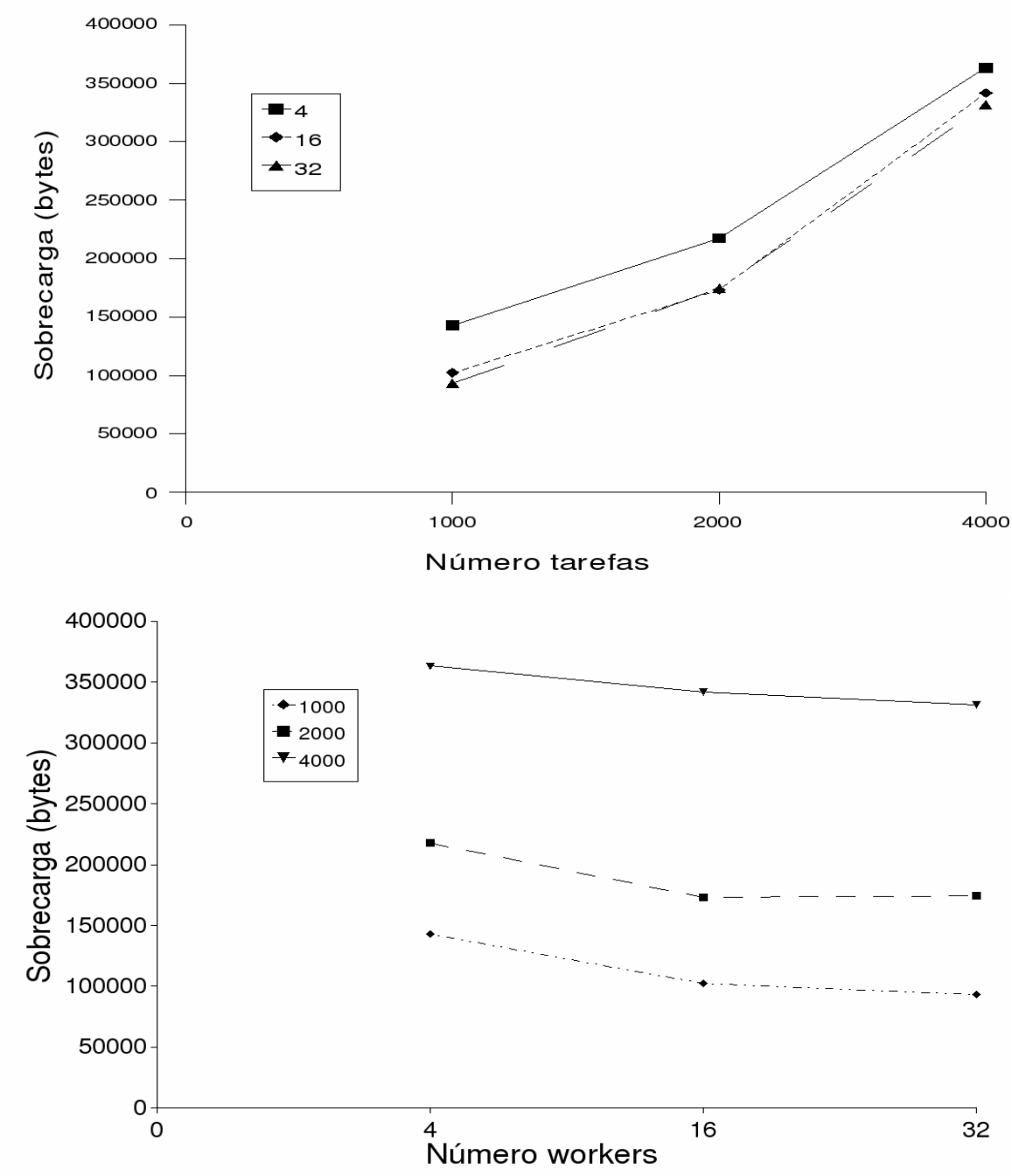

Fig. 4.19: Cálculo de π - Sobrecarga tráfego na porta TCP 8000

Tab. 4.10: Cálculo de π - Tráfego (bytes) em cada porta TCP com 1 *worker*

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*	*JoiNMon ativo*	*Comparativo*	
			Média	*Média*	*Diferença*	*Sobrecarga*
1	1000	8000	19535	270402	250867	1284,19%
		8100	1803634	1842619	38985	2,16%
		9001	-	2823926	-	-
		total	1823169	4936947	3113778	170,79%
	2000	8000	18631	308182	289551	1554,14%
		8100	3383538	3484205	100667	2,98%
		9001	-	5586526	-	-
		total	3402169	9378913	5976744	175,67%
	4000	8000	22879	415061	392182	1714,16%
		8100	6125042	6403058	278016	4,54%
		9001	-	10829552	-	-
		total	6147921	17647671	11499750	187,05%

pode introduzir um gargalo na comunicação entre os componentes *server* e *coordinator*. Nas configurações típicas do sistema JoiN a ocorrência deste gargalo é pouco provável, devido à recomendação de que o componente *server* e os componentes *coordinator* sejam executados em computadores robustos, interligados por redes confiáveis e de alta velocidade.

Gostaríamos de ressaltar que a alta sobrecarga relativa observada no tráfego de rede nos experimentos com esta aplicação é pouco significativa no desempenho global da mesma, como já demonstrado pela Fig. 4.7.

Dados obtidos nos experimentos com a aplicação para multiplicação de matrizes

O tráfego de dados para transmissão de informações entre as tarefas que processam esta aplicação pode ser grande pois para cada uma das tarefas são enviadas as matrizes a serem multiplicadas: são enviadas todas as linhas e colunas da primeira matriz e a segunda matriz é subdivida conforme o número de tarefas a serem executadas e cada porção da mesma é enviada para a tarefa correspondente. Cada tarefa retorna uma submatriz contendo o resultado da multiplicação. Para esta classe de aplicações ($R_{cc} > 1$) é esperado que a sobrecarga relativa introduzida pela ferramenta de monitoramento no tráfego de rede não seja muito expressiva.

Na Tab. 4.11 são exibidos os dados relativos ao tráfego de rede total das portas TCP/8000, TCP/8100 e TCP/9001 obtidos nos experimentos realizados com a aplicação para o multiplicação de matrizes.

Podemos observar na Tab. 4.11 que a sobrecarga no tráfego de rede introduzida pela ferramenta de monitoramento durante os diversos experimentos conduzidos com a aplicacação para multiplicação de matrizes não chegou a 10%, o que representa um resultado satisfatório uma vez que, como vimos em 4.4.1, a ferramenta de monitoramento não introduziu sobrecarga superior a 5,52% no tempo total de execução da aplicação, que inclui o tempo para comunicação. O valor médio da sobrecarga introduzida na comunicação foi de 5,75%.

Tab. 4.11: Multiplicação de matrizes - Tráfego total (bytes) pelas portas TCP 8000, 8100 e 9001

Num. workers	*Num. tasks*	*Sem JoiNMon*	*JoiNMon ativo*	*Comparativo*	
		Média (bytes)	*Média (bytes)*	*Diferença*	*Sobrecarga*
4	100	859458050	933142668	73684618	8.57%
	200	1405869881	1454930819	49060938	3.49%
	400	2114298003	2261447769	147149766	6.96%
8	100	895860471	982578918	86718447	9.68%
	200	1610178481	1628475182	18296701	1.14%
	400	2164569250	2290321737	125752487	5.81%
12	100	732872939	788326140	55453201	7.57%
	200	921908010	982779698	60871688	6.60%
	400	1449744459	1477668124	27923664	1.93%
16	100	731024542	747245575	16221033	2.22%
	200	849998306	870066249	20067943	2.36%
	400	1292466994	1348651037	56184044	4.35%
24	100	561845022	602622727	40777705	7.26%
	200	723424700	795224986	71800286	9.93%
	400	1155046141	1178038466	22992325	1.99%
32	100	683810855	737762598	53951744	7.89%
	200	804971164	859552460	54581296	6.78%
	400	1207334347	1314728253	107393906	8.90%

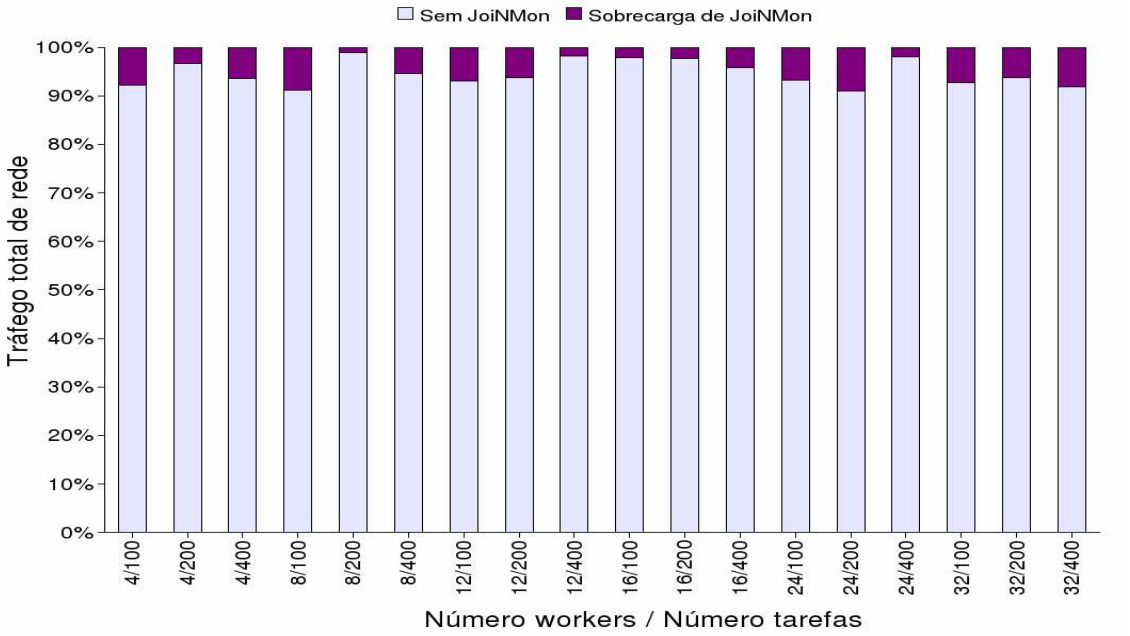

Fig. 4.20: Multiplicação de matrizes - Sobrecarga no tráfego de rede

A Fig. 4.20 apresenta um gráfico onde também pode ser observado que o tráfego de rede gerado pela ferramenta de monitoramento na execução desta aplicação representa no máximo 10% do tráfego total observado durante a execução de uma aplicação da classe $R_{cc} > 1$ no sistema JoiN. Na Tab. 4.3 apresentamos a estimativa de $R_{cc} \sim= 14,14$ para a aplicação para multiplicação de matrizes. Como vimos anteriormente, R_{cc} = *(tempo_de_computação / tempo_de_comunicação)*. Se considerarmos a mesma sobrecarga de 10% sobre o tempo de comunicação e desprezarmos a sobrecarga no tempo de processamento, podemos concluir que JoiNMon altera R_{cc} desta aplicação de 14,14 para 12,7. Ainda assim a utilização do sistema JoiN para processamento desta aplicação apresenta boa relação custo/benefício.

Em Fig. 4.21, Fig. 4.22 e Fig. 4.23 são apresentados gráficos referentes às informações contidas na Tab. 4.11 sobre o tráfego de rede total nas portas TCP 8000, 8100 e 9001.

Em Fig. 4.21 e Fig. 4.22 são apresentados gráficos referentes a sobrecargas observadas. Nestes gráficos não é possível observar um padrão de comportamento e a imprevisibilidade dos dados absolutos se mantém nos dados relativos. A sobrecarga pode variar devido a diversos fatores, como já apresentado anteriormente, entre os quais citamos os diferentes fatores de desempenho dos componentes *worker* e o escalonamento de tarefas realizado pelo sistema JoiN, dos quais depende diretamente o fluxo da execução das tarefas e, consequentemente, o tráfego de rede. Analisando a Fig. 4.23, onde é apresentado o gráfico correspondente ao tráfego médio por cada tarefa da aplicação para multiplicação de matrizes, podemos observar o mesmo comportamento em cada par de curvas, composto por: uma linha contínua, que representa o experimento sem a ferramenta de monitoramento; uma linha pontilhada, que representa o JoiNMon ativado. Pela observação das informações exibidas nesta figura, concluimos que a ferramanta de monitoramento não altera o comportamento do tráfego de rede do sistema JoiN na execução desta aplicação.

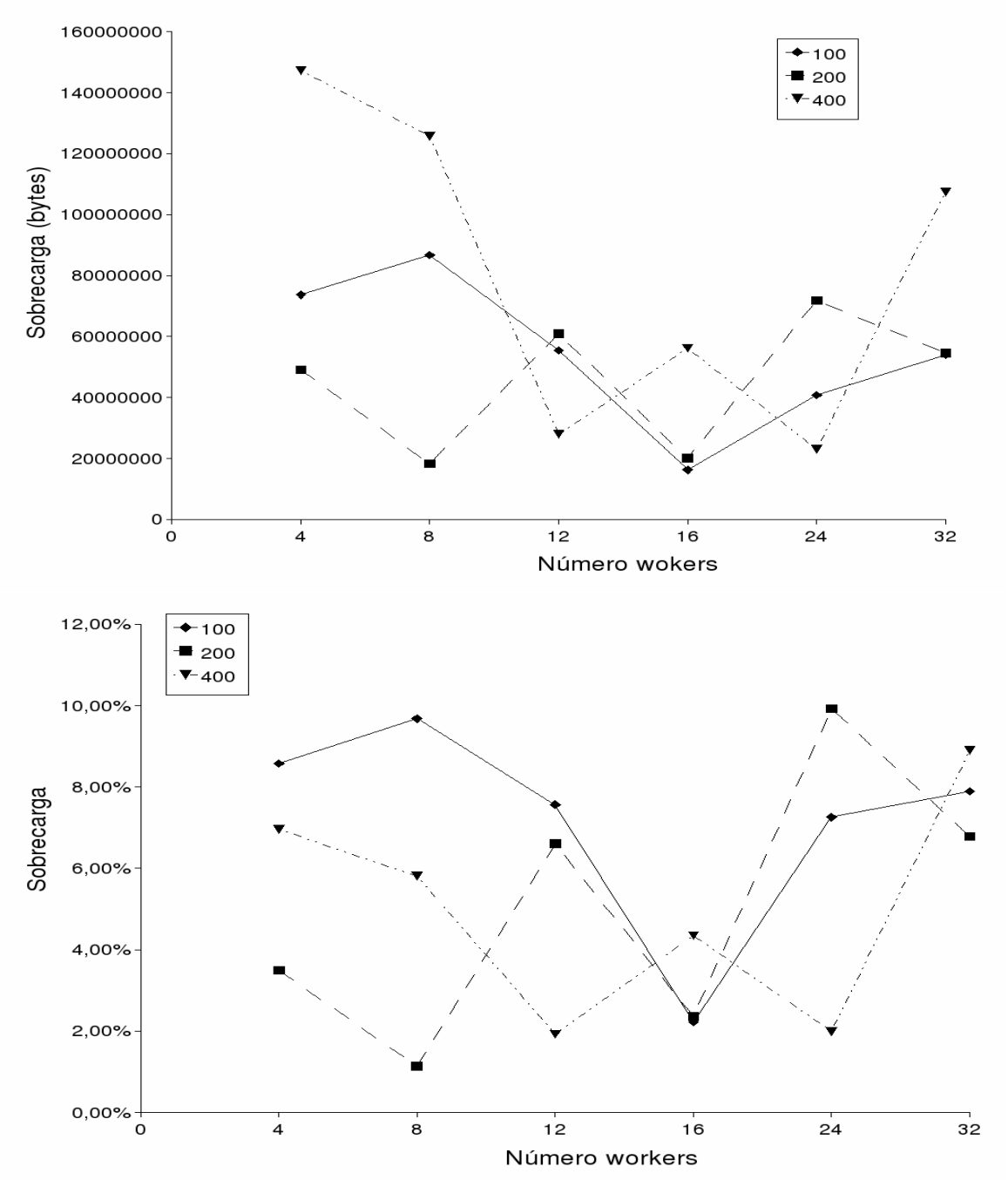

Fig. 4.21: Multiplicação de matrizes - Sobrecarga tráfego de rede total (em função do número de *workers*)

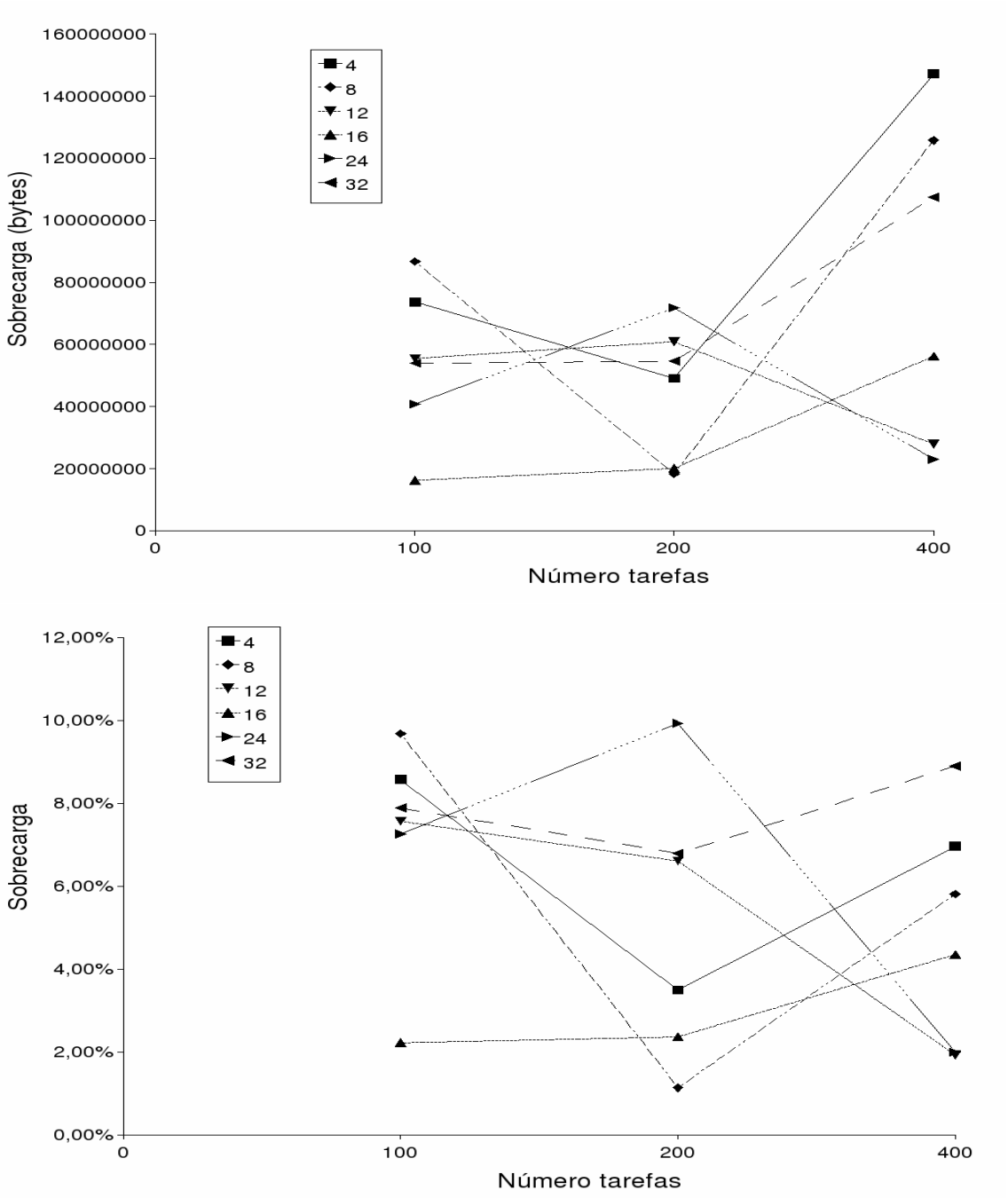

Fig. 4.22: Multiplicação de matrizes - Sobrecarga tráfego de rede total (em função do número de tarefas)

Tab. 4.12: Multiplicação de matrizes - Tráfego (bytes) em cada porta TCP com 4, 16 e 32 *workers*

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*	*JoiNMon ativo*	*Comparativo*	
			Média	*Média*	*Diferença*	*Sobrecarga*
4	100	8000	9527	24757	15230	159.87%
		8100	859448523	932833207	73384684	8.54%
		9001	-	284704	-	-
		total	859458050	933142668	73684618	8.57%
	200	8000	10044	38759	28715	285.89%
		8100	1405859837	1454531475	48671637	3.46%
		9001	-	360585	-	-
		total	1405869881	1454930819	49060938	3.49%
	400	8000	14129	53352	39223	277.61%
		8100	2114283874	2260665600	146381726	6.92%
		9001	-	728817	-	-
		total	2114298003	2261447769	147149766	6.96%
16	100	8000	9282	42899	33617	362.17%
		8100	731015260	746890646	15875386	2.17%
		9001	-	312030	-	-
		total	731024542	747245575	16221033	2.22%
	200	8000	8574	34903	26329	307.06%
		8100	849989731	869646061	19656330	2.31%
		9001	-	385285	-	-
		total	849998306	870066249	20067943	2.36%
	400	8000	9167	36052	26885	293.27%
		8100	1292457827	1348614985	56157159	4.34%
		9001	-	912528	-	-
		total	1292466994	1348651037	56184044	4.35%
36	100	8000	9799	42389	32590	332.60%
		8100	683801056	737720210	53919154	7.89%
		9001	-	368595	-	-
		total	683810855	737762598	53951744	7.89%
	200	8000	10045	49658	39613	394.37%
		8100	804961119	859502802	54541683	6.78%
		9001	-	463948	-	-
		total	804971164	859552460	54581296	6.78%
	400	8000	11546	50688	39142	339.01%
		8100	1207322801	1314677564	107354763	8.89%
		9001	-	934642	-	-
		total	1207334347	1314728253	107393906	8.90%

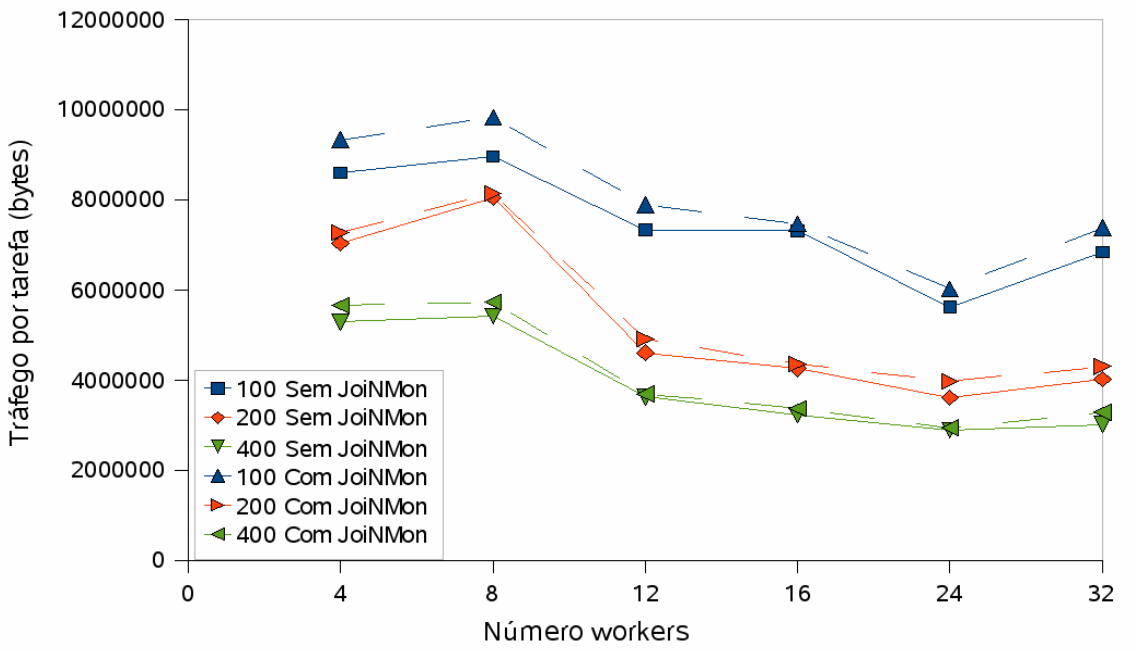

Fig. 4.23: Multiplicação de matrizes - Tráfego médio por tarefa

Na Tab. 4.12 são exibidos os dados relativos ao tráfego de rede separado por porta, além do tráfego total nestas portas, para os experimentos realizados com 4, 16 e 32 componentes *worker*. Estes dados são suficientes para ilustrar a análise do tráfego em cada uma das portas, pois não há variação significativa com relação ao comportamento observado nos demais experimentos. No Apêndice D podem ser encontradas todas as informações coletadas nos experimentos.

Na Tab. 4.12, onde são apresentadas as informações sobre o tráfego em cada uma das portas TCP, é possível observar que:

- o tráfego na porta TCP/8000, pela qual são enviadas as informações coletadas e agrupadas pela ferramenta de monitoramento, aumenta da ordem de 2 a 3 vezes quando se ativa o monitoramento. Contudo, mesmo no caso em que este aumento é de 39613 bytes (máximo observado), o volume de tráfego não pode ser considerado um fator que venha a alterar a eficiência ou o desempenho do sistema, pois este volume de tráfego é considerado baixo e suportado com facilidade pela tecnologia de rede existente atualmente;
- o tráfego na porta TCP/9001, utilizada para acesso ao servidor de banco de dados (HSQLDB) é da ordem de 2500 bytes por tarefa que compõe a aplicação. Se o número de tarefas for muito grande (muitos milhares), este tráfego pode interferir no desempenho do sistema JoiN. Aqui também vale a observação da análise do tráfego da aplicação para cálculo de π relativa a possibilidade de executarmos no mesmo computador o servidor de bancos de dados e o componente *server*, transformando a comunicação com o servidor de banco de dados em comunicação interna no mesmo computador, sem sobrecarga no tráfego de rede;
- a sobrecarga de tráfego na porta TCP/8100, utilizada para comunicação entre os componentes *coordinator* e *workers* a ele associados, não foi superior, nestes 9 experimentos apresentados,

a 8,9%, com valor médio de 5,7%. É o tráfego nesta porta que contribui em maior parte para o sobrecarga imposta pela ferramenta de monitoramento. Cabe aqui uma observação para que, em futuros trabalhos com a ferramenta JoiNMon, sejam observadas com atenção a formatação dos eventos, a melhor utilização ou reavaliação do mecanismo de *cache* e a possibilidade de utilização de um mecanismo de *piggybacking* [106], anexando os eventos em pacotes de dados que trafegam normalmente no sistema JoiN. Diferentemente dos resultados obtidos nos experimentos com a aplicação para cálculo de π, para a aplicação de multiplicação de matrizes não observamos redução do tráfego na porta TCP/8100. No caso da aplicação para cálculo de π a redução foi causada pela diminuição do número de tarefas replicadas, como apresentado anteriormente. No caso da aplicação para multiplicação de matrizes utilizamos menores números de tarefas (100, 200 e 400 no lugar de 1000, 2000 e 4000) e, por esta razão, a replicação de tarefas é menor em todas as condições de execução dos experimentos, sem a ferramenta de monitoramento ou com a mesma ativada.

Em Fig. 4.24, Fig. 4.25, Fig. 4.26 e Fig. 4.27 são apresentados gráficos referentes às informações contidas na Tab. 4.12 sobre o tráfego de rede nas portas TCP 8000 e 8100 para os experimentos com 4, 16 e 32 componentes *worker*.

Na Fig. 4.24 são apresentados gráficos referentes a sobrecarga no tráfego de rede observado na porta TCP 8000. No primeiro gráfico desta figura é possível observar que, em geral, a sobrecarga absoluta da ferramenta de monitoramento não apresenta grandes variações quando aumentamos o número de componentes *worker*. Este é o comportamento esperado, uma vez que nesta porta trafegam eventos relacionados ao início e término de tarefas. No segundo gráfico, onde é apresentada a sobrecarga relativa, observamos uma tendência de aumento à medida que aumentamos o número de componentes *worker*, entretanto este comportamento não causa prejuízos à execução da aplicação. Alguns dos pontos destes gráficos refletem a dificuldade de obtermos estas informações com precisão, devido aos baixo volume de tráfego.

Na Fig. 4.25 são apresentados gráficos referentes a sobrecarga no tráfego de rede observado na porta TCP 8100. Se não considerarmos os pontos relativos aos experimentos com 16 componentes *worker* observamos no primeiro gráfico (sobrecarga absoluta) uma tendência de queda e no segundo gráfico (sobrecarga relativa) uma tendência de aumento da sobrecarga à medida que aumentamos o número de *workers*. No segundo gráfico, onde é apresentada a sobrecarga relativa, observamos uma tendência de aumento à medida que aumentamos o número de componentes *worker*, entretanto este comportamento não causa prejuízos à execução da aplicação. Como pela porta TCP/8000 trafegam as informações coletadas e agrupadas pela ferramenta de monitoramento, na transmissão de dados de monitoramento dos componentes *coordinator* para o componente *server* para que este faça o armazenamento das mesmas no banco de dados, aumentando o número de tarefas ou o número de componentes *worker*, aumentamos o volume de tráfego nesta porta. É importante lembrar que este tráfego pode introduzir um gargalo na comunicação entre os componentes *server* e *coordinator*.

Na Fig. 4.26 e na Fig. 4.27 são apresentados gráficos referentes ao tráfego médio por cada tarefa da aplicação para multiplicação de matrizes observado, respectivamente, nas nas portas TCP 8000 e 8100. Analisando os gráficos apresentados em cada uma destas figuras observamos que o comportamento das curvas é o mesmo tanto para os experimentos sem a ferramenta de monitoramento, quanto para aqueles em que JoiNMon foi ativado. As informações contidas nesta figura nos levam a concluir que a ferramanta de monitoramento não altera o comportamento do tráfego de rede nas portas TCP

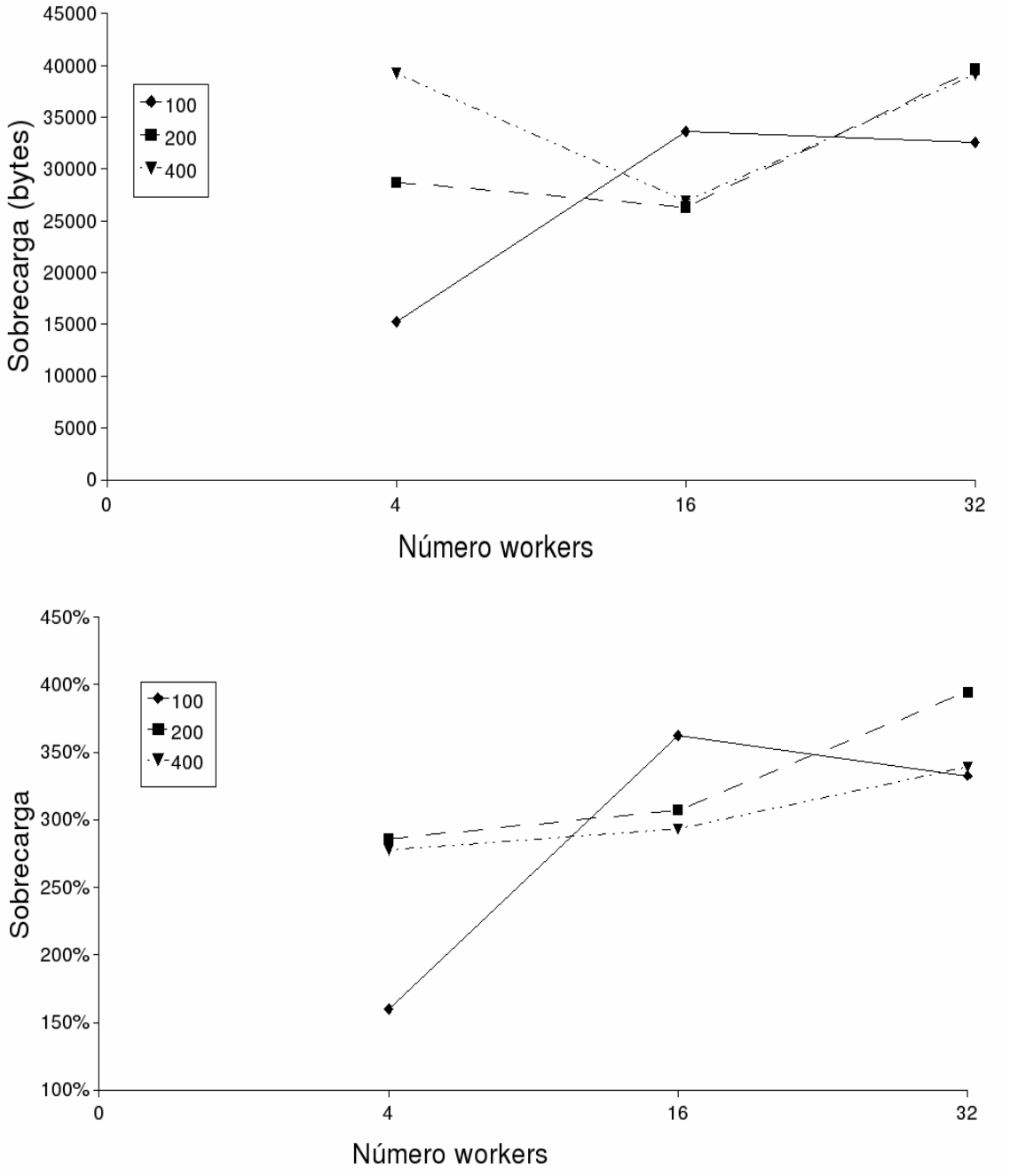

Fig. 4.24: Multiplicação de matrizes - Sobrecarga tráfego na porta TCP 8000

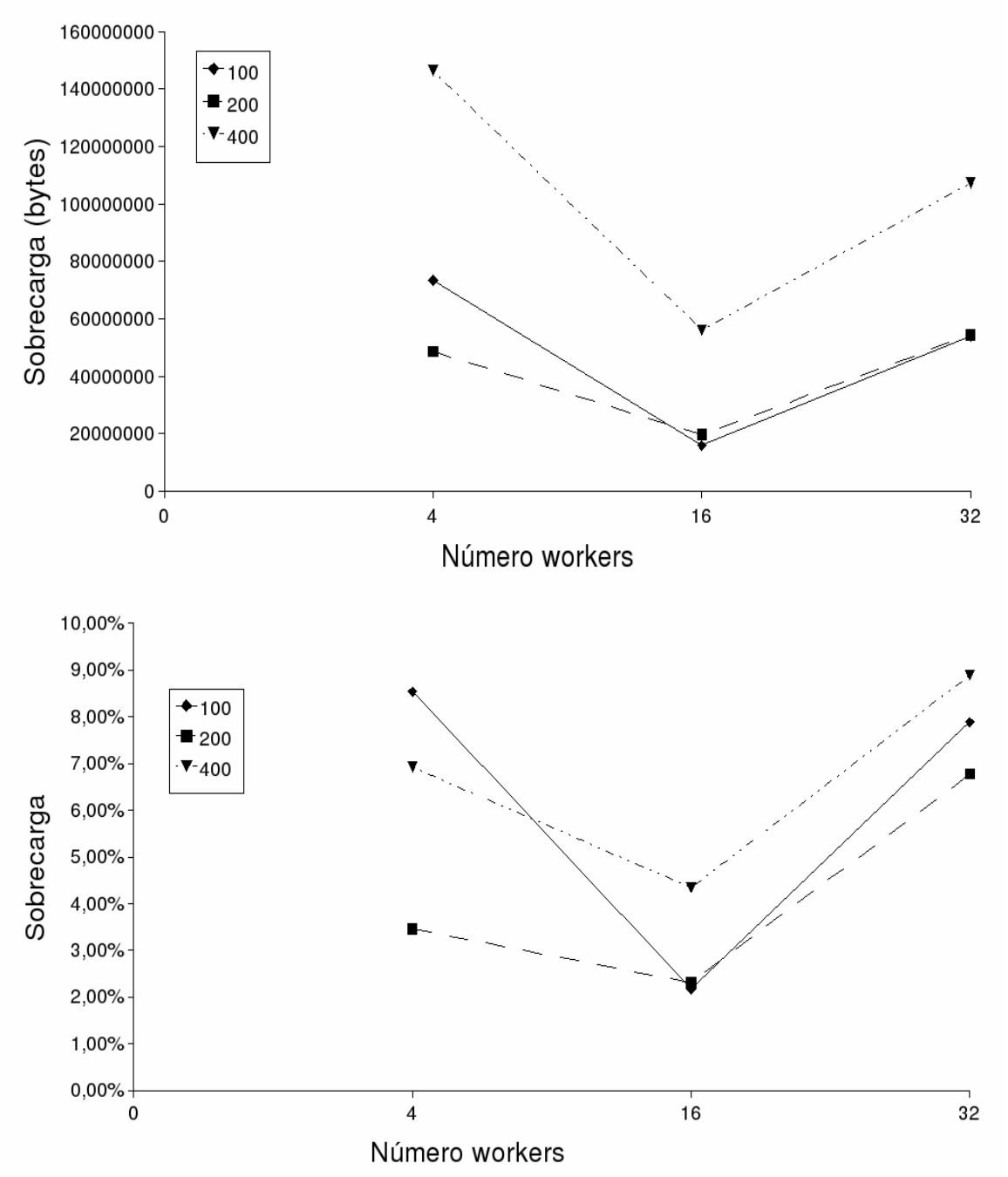

Fig. 4.25: Multiplicação de matrizes - Sobrecarga tráfego na porta TCP 8100

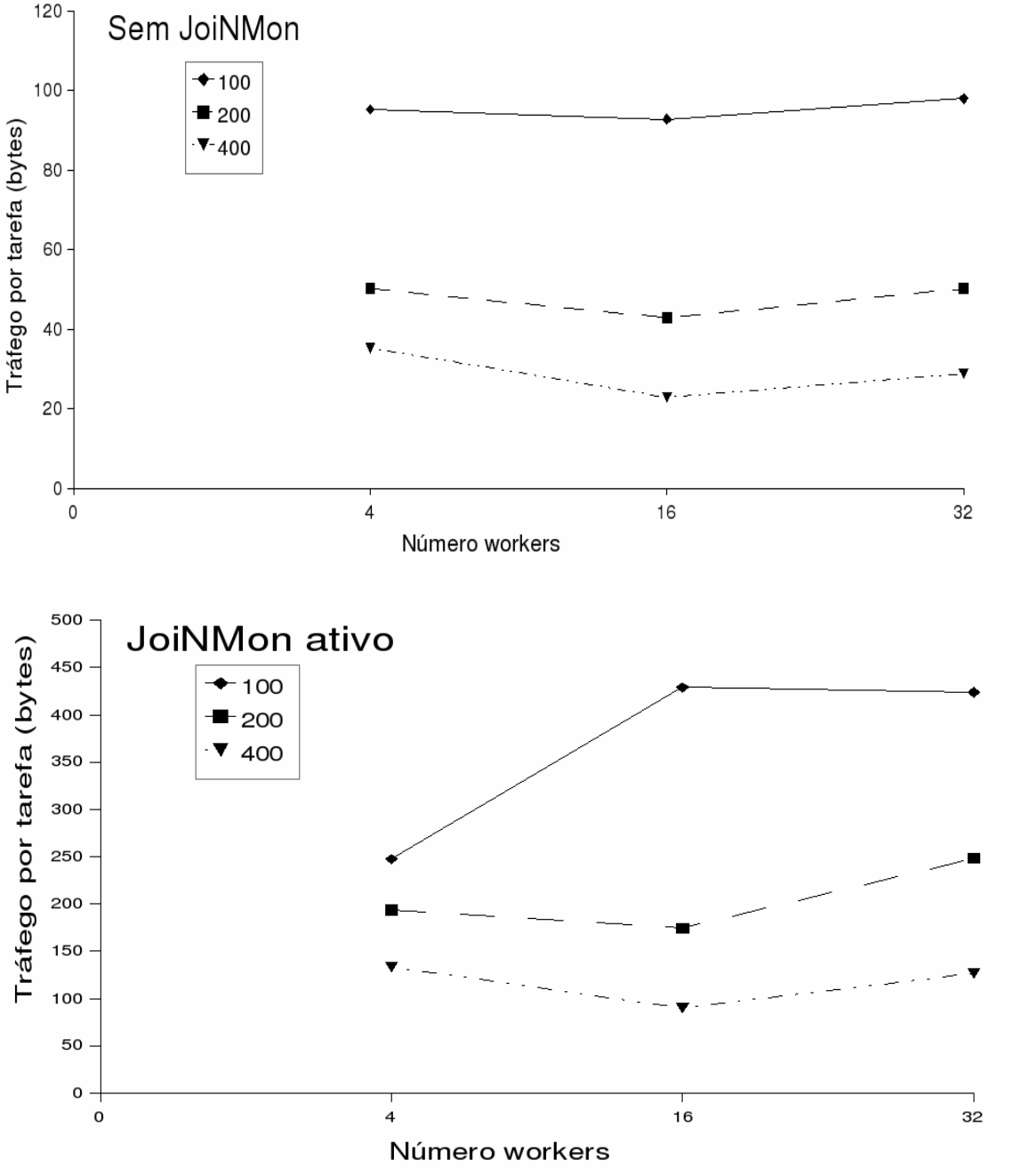

Fig. 4.26: Multiplicação de matrizes - Tráfego médio por tarefa na porta TCP 8000

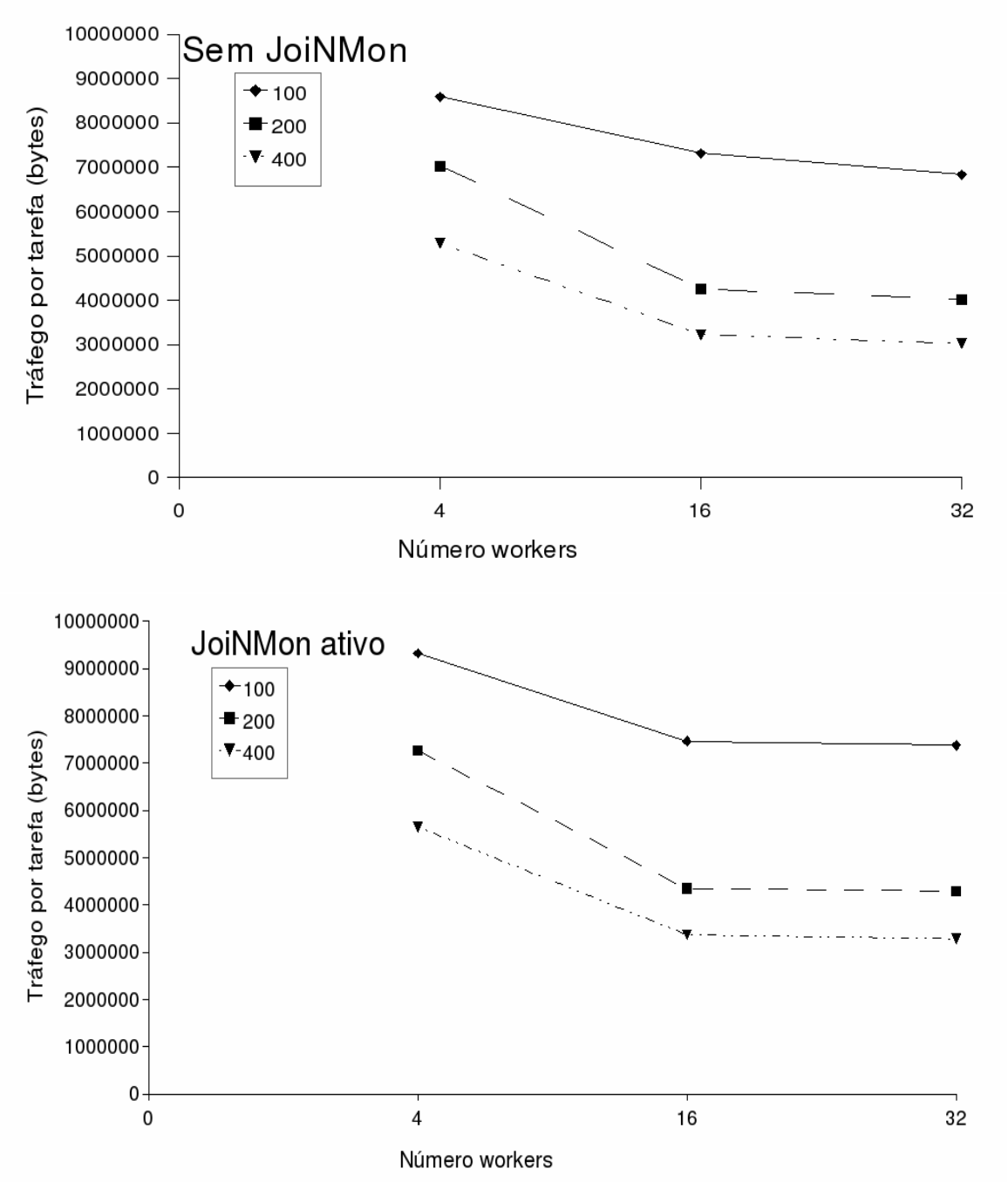

Fig. 4.27: Multiplicação de matrizes - Tráfego médio por tarefa na porta TCP 8100

8000 e 8100 do sistema JoiN na execução desta aplicação.

4.4.3 Aplicação multiplicação de matrizes com razão R_{cc} constante

Nesta seção são apresentados resultados obtidos em experimentos realizados com a aplicação para multiplicação de matrizes onde o tamanho das sub-matrizes é mantido fixo. Desta forma, o tamanho do problema cresce quando aumentamos o número de tarefas, tanto no processamento quanto no tráfego de dados, pois a carga de cada tarefa permanece a mesma. O objetivo destes experimentos é fornecer dados para a análise da da eficiência da ferramenta de monitoramento na execução da classe de aplicações $R_{cc} > 1$ quando esta razão permanece constante em diferentes configurações do sistema JoiN (número de componentes *worker*) e do número de tarefas da aplicação.

Configurações para os experimentos

Para execução destes experimentos foi utilizado o mesmo ambiente apresentado na seção 4.3.6, e as configurações apresentadas na Tab. 4.13. Os experimentos foram feitos em duas etapas: na primeira etapa foi utilizada a plataforma JoiN sem a ferramenta de monitoramento; na segunda etapa a ferramenta de monitoramento foi ativada e os experimentos repetidos. Além disto, cada experimento foi feito três vezes, para que pudesse ser calculada a média dos valores obtidos, que é o valor utilizado nas avaliações.

Cada uma das tarefas, em qualquer das configurações, executa a multiplicação de 2 matrizes de dimensões 400x400.

Tab. 4.13: Configurações para os experimentos com multiplicação de matrizes de tamanho fixo

Número *workers*	Número tarefas		
4	100	200	400
16	100	200	400
32	100	200	400

Avaliação do impacto no desempenho

Na Tab. 4.14 são exibidos, resumidamente, os dados obtidos nos experimentos realizados para observação do impacto da ferramenta de monitoramento JoiNMon no tempo de execução da aplicação para multiplicação de matrizes de tamanho fixo. A apresentação dos dados obtidos é subdividida conforme a etapa: a primeira, na coluna intitulada *Sem JoiNMon*, feita com o sistema JoiN sem a ferramenta de monitoramento; e a segunda, na coluna *JoiNMon ativo*, feita com a ferramenta JoiNMon ativada. Para cada uma destas etapas, são apresentados o valor médio (em milissegundos), e o número de tarefas replicadas, calculados com base nos valores obtidos nas três execuções do experimento, com mesmo número de componentes *worker* e de tarefas. Nas duas últimas colunas são apresentados a diferença entre os valores médios de execução, e a sobrecarga que a ferramenta de monitoramento acarreta no tempo médio de execução desta aplicação. Os valores de desvio padrão e coeficiente de variação (expresso como a porcentagem do desvio padrão dividido pela média) de cada

Tab. 4.14: Multiplicação de matrizes de tamanho fixo - Tempo de processamento (ms)

Numero *workers*	Numero *tasks*	Sem JoiNMon		JoiNMon ativo		Comparativo	
		Média	Num Rép	Média	Num Rép	Diferença	Sobrecarga
4	100	16719	9	17108	10	389	2.33%
	200	28539	9	29031	9	492	1.72%
	400	59165	8	61039	9	1874	3.17%
16	100	6485	41	6616	38	131	2.02%
	200	11373	40	11762	38	389	3.42%
	400	23924	44	24671	46	747	3.12%
32	100	3741	90	3825	99	84	2.25%
	200	6327	79	6511	76	184	2.91%
	400	11081	81	11483	85	402	3.63%

Num Rép = número réplicas

experimento, bem como o erro padrão e o intervalo de confiança da sobrecarga, são apresentados na Tab. D.5 do Apêndice D, onde podemos observar que os valores de coeficiente de variação são relativamente pequenos. Por esta razão, podemos considerar que os valores médios de tempo de execução das aplicações são apropriados para a avaliação de resultados.

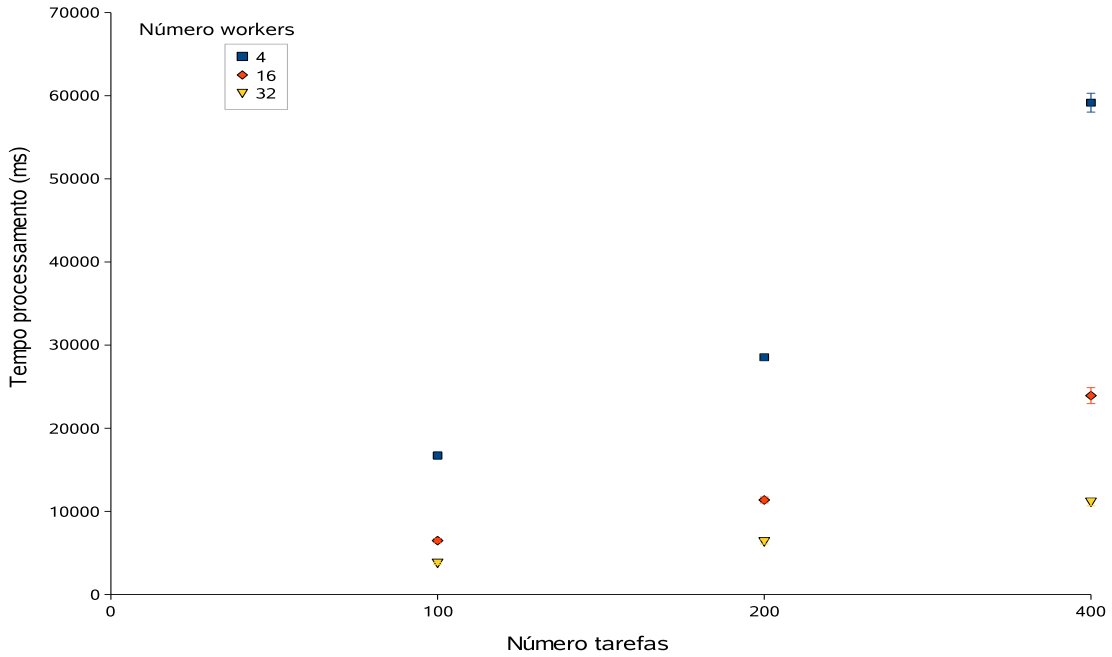

Fig. 4.28: Multiplicação de matrizes de tamanho fixo, sem monitoramento - Tempo médio de processamento (ms)

Os valores médios do tempo de processamento obtidos nestes experimentos para multiplicação

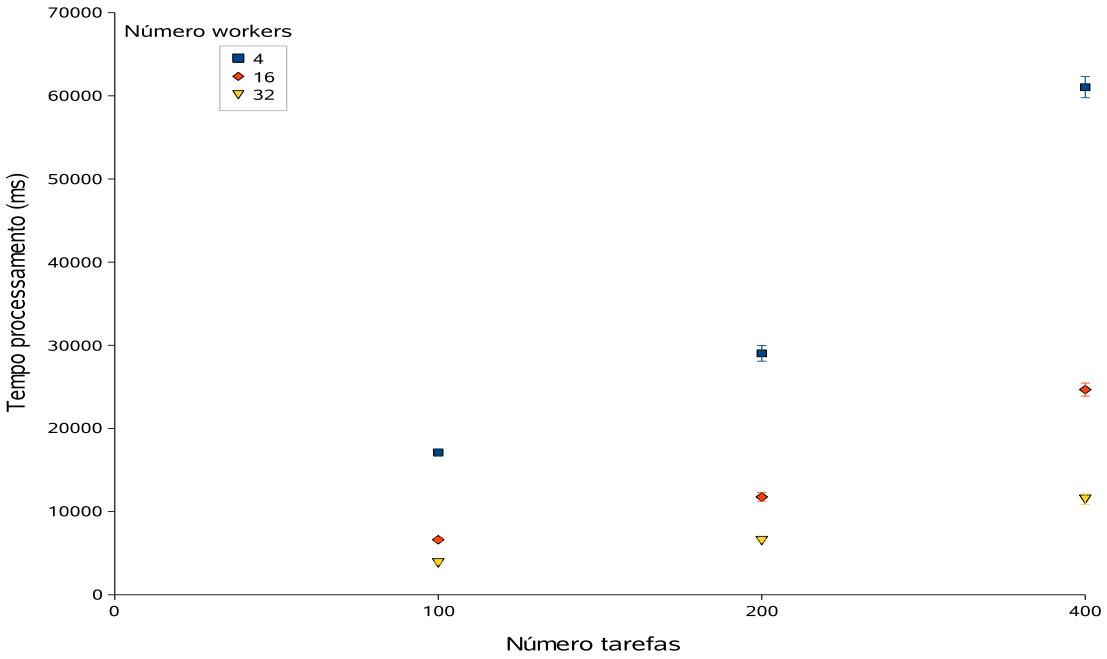

Fig. 4.29: Multiplicação de matrizes de tamanho fixo, com monitoramento - Tempo médio de processamento (ms)

de matrizes de tamanho fixo, e os respectivos intervalos de confiança, são apresentados na Tab. D.5 do Apêndice D e estão representados nos gráficos da Fig. 4.28 e da Fig. 4.29. Nestas figuras podem ser observadas as representações de alguns dos intervalos de confiança. Isto acontece porque estes intervalos, apesar de pequenos, não são insignificantes quando comparados aos tempos de processamento obtidos. Para permitir a comparação de parte dos valores obtidos nos experimentos sem monitoramento e com a ferramenta de monitoramento ativada, no gráfico da Fig. 4.30 são apresentados os valores observados nos experimentos utilizando 4 componentes *worker*. Neste caso, os tempos medidos sem e com a ferramenta de monitoramento são muito parecidos e observamos empates técnicos para 100, 200 e 400 tarefas, mostrando que, também nestes casos, o impacto da ferramenta de monitoramento é baixo.

Verificamos, nos dados obtidos nos experimentos realizados com a aplicação para multiplicação de matrizes de tamanho fixo que a sobrecarga observada na execução não foi, em qualquer dos casos, superior a 3,63%. A mediana dos valores obtidos é 2,91% e a porcentagem média de sobrecarga para todos os casos foi de 2,73%. Na Fig. 4.31, que representa o tempo médio de processamento da aplicação para multiplicação de matrizes de tamanho fixo nas duas etapas, também é possível observar que a sobrecarga imposta pela ferramenta de monitoramento não é significativa para a execução desta aplicação.

A sobrecarga observada nestes experimentos são representadas na Fig. 4.32 e na Fig. 4.33. Em cada figura são mostrados dois gráficos: o primeiro com as curvas relativas às diferenças entre os valores médios (em milissegundos) e o segundo com as curvas de "sobrecarga relativa" (porcentagem da sobrecarga sobre os valores obtidos com *JoiNMon Inativo*). Nos gráficos da Fig. 4.32 nota-se uma tendência de queda na sobrecarga absoluta quando se aumenta o número de componentes *worker*

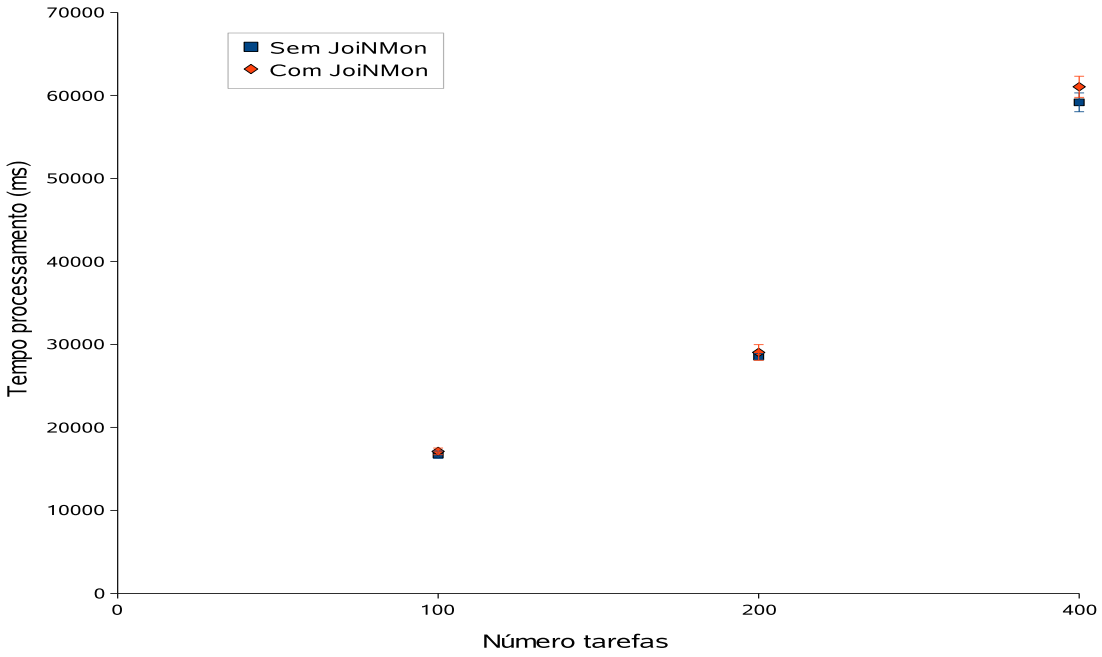

Fig. 4.30: Multiplicação de matrizes de tamanho fixo, com 4 *workers* - Tempo médio de processamento (ms) com e sem monitoramento

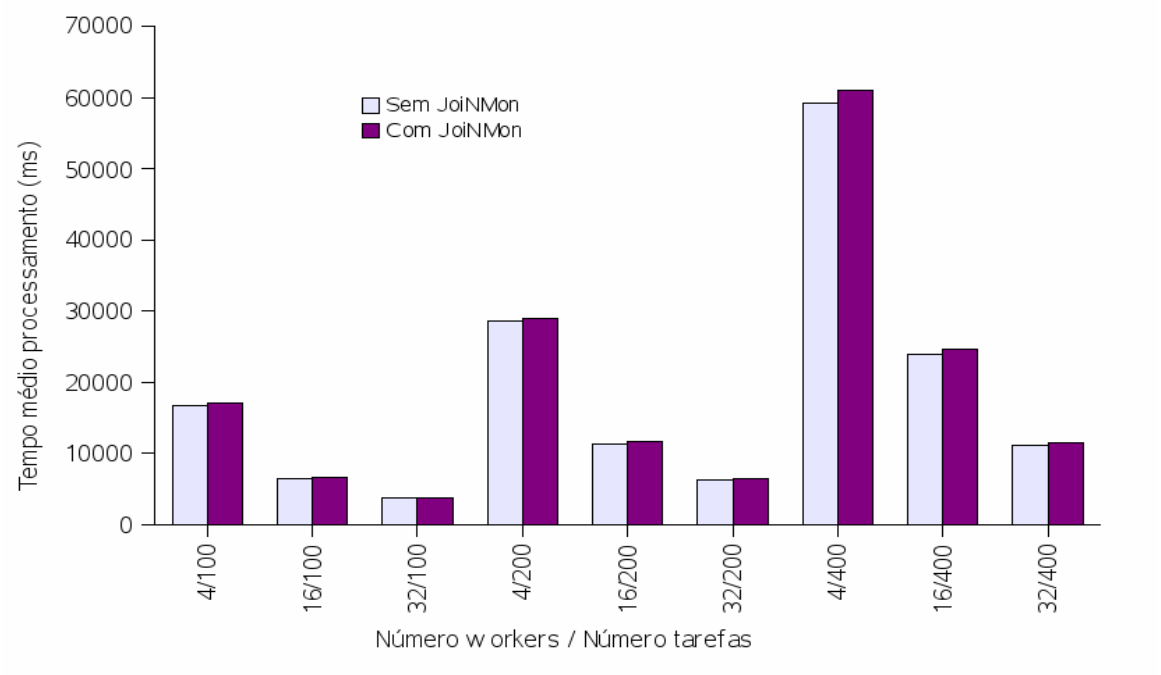

Fig. 4.31: Multiplicação de matrizes de tamanho fixo - Tempo de processamento (ms) com e sem monitoramento

mas, ao observar o gráfico da sobrecarga relativa, percebe-se que a mesma se mantém praticamente constante. Nos gráficos da Fig. 4.33 nota-se uma pequena tendência de aumento na sobrecarga, tanto absoluta quanto relativa, quando se aumenta o número de tarefas, que é o comportamento esperado uma vez que o número de eventos aumenta com o número de tarefas. Este comportamento que sugere que a ferramenta de monitoramento não interfere significativamente no desempenho do sistema JoiN para a execução desta aplicação.

Avaliação da sobrecarga no tráfego de rede

Na Tab. 4.15 são exibidos, resumidamente, os dados relativos ao tráfego de rede em cada uma das portas TCP/8000, TCP/8100 e TCP/9001, além do tráfego acumulado nestas portas.

Tab. 4.15: Multiplicação de matrizes de tamanho fixo - Tráfego de rede (bytes) em cada porta TCP

Num.	Num.	Porta	Sem JoiNMon	JoiNMon ativo	Comparativo	
workers	*tasks*	TCP	Média	Média	Diferença	Sobrecarga
4	100	8000	7392	20887	13495	182.56%
		8100	55303901	55796420	492519	0.89%
		9001	0	309726		
		total	55311293	56127033	815740	1.47%
4	200	8000	7157	32222	25065	350.22%
		8100	116878281	117669102	790821	0.68%
		9001	0	471312		
		total	116885438	118172636	1287198	1.10%
4	400	8000	7721	48575	40854	529.13%
		8100	253001495	258229026	5227531	2.07%
		9001	0	726007		
		total	253009216	259003608	5994392	2.37%
16	100	8000	7392	36090	28698	388.23%
		8100	55971593	56111231	139638	0.25%
		9001	0	282689		
		total	55978985	56430010	451025	0.81%
16	200	8000	9158	54169	45011	491.49%
		8100	112604028	113438176	834148	0.74%
		9001	0	537164		
		total	112613186	114029509	1416323	1.26%
16	400	8000	9509	76977	67468	709.52%
		8100	252249016	255052942	2803926	1.11%
		9001	0	1124804		
		total	252258525	256254723	3996198	1.58%
32	100	8000	7305	32976	25671	351.42%
		8100	59865596	61360031	1494435	2.50%
		9001	0	373433		
		total	59872901	61766440	1893539	3.16%

continua na próxima página

Tab. 4.15: Multiplicação de matrizes de tamanho fixo - Tráfego de rede (bytes) em cada porta TCP (continuação)

Num.	*Num.*	*Porta*	*Sem JoiNMon*	*JoiNMon ativo*	*Comparativo*	
workers	*tasks*	*TCP*	*Média*	*Média*	*Diferença*	*Sobrecarga*
32	200	8000	7528	44219	36691	487.39%
		8100	108189997	109855070	1665073	1.54%
		9001	0	773274		
		total	108197525	110672563	2475038	2.29%
32	400	8000	8052	53953	45901	570.06%
		8100	246599017	246952335	353318	0.14%
		9001	0	938985		
		total	246607069	247945273	1338204	0.54%

São apresentados os resultados obtidos, subdivididos conforme a etapa: a primeira, intitulada *Sem JoiNMon*, feita com o sistema JoiN sem a ferramenta de monitoramento; e a segunda, intitulada *JoiNMon ativo*, feita com a ferramenta JoiNMon ativada. Para cada uma destas etapas, são apresentados os valores médios, em número de bytes, calculados com base nos valores obtidos nas três execuções do experimento, com mesmo número de componentes *worker* e de tarefas. Nas duas últimas colunas são apresentados a diferença entre os valores médios de cada etapa, e a sobrecarga que a ferramenta de monitoramento acarreta no tráfego de rede durante a execução desta aplicação. Os valores de desvio padrão e coeficiente de variação (expresso como a porcentagem do desvio padrão dividido pela média) de cada experimento, bem como o erro padrão e o intervalo de confiança da sobrecarga, são apresentados nas tabelas detalhadas do Apêndice D.

Os valores de coeficiente de variação apresentados na Tab. D.10 do Apêndice D são sempre inferiores a 4,2%. Desta forma, aqui também podemos considerar os valores médios nas avaliações sem introduzir grandes erros.

O tráfego de dados para transmissão de informações entre as tarefas que processam esta aplicação pode ser grande pois para cada uma das tarefas são enviadas as matrizes a serem multiplicadas: são enviadas todas as linhas (400) e colunas (400) das duas matrizes. Cada tarefa retorna uma matriz de dimensões (400x400) contendo o resultado da multiplicação. A replicação de tarefas, mecanismo do escalonamento que garante a execução de todas as tarefas no menor tempo possível, evitando que falhas em componentes *worker* interfiram no processamento das aplicações, tem grande influência no tráfego de redes no caso de aplicações da classe $R_{cc} > 1$, pois cada tarefa replicada gera um volume de tráfego que não deve ser desprezado.

Nas informações da Tab. 4.15 é possível observar que a maior sobrecarga no tráfego de rede ocorre na configuração em 100 tarefas foram processadas utilizando 32 componentes *worker*. Ao observarmos o número de tarefas replicadas para esta configuração na Tab. 4.14, identificamos que neste caso ocorreu o maior número de réplicas e que foi ainda maior quando a ferramenta JoiNMon está ativa. Estas informações reforçam a conclusão de que o número de réplicas influencia o volume de dados que trafegam pela rede no caso desta classe de aplicações.

Na Fig. 4.34 é apresentada a sobrecarga no tráfego de rede imposta pela ferramenta de monitoramento durante a execução desta aplicação.

Nos experimentos com a aplicação para multiplicação de matrizes de tambanho fixo obtivemos resultados similares aos observados nos experimentos com a aplicação para multiplicação de matrizes

apresentados na seção 4.4.2.

Na Fig. 4.35 e na Fig. 4.36 são apresentados gráficos referentes a sobrecargas observadas. Nestes gráficos não é possível observar um padrão de comportamento. Isto ocorre porque a sobrecarga imposta pela ferramenta de monitoramento é tão pequena em relação ao volume total de tráfego que as medidas são influenciadas por fatores externos a JoiNMon, como já apresentado anteriormente, entre os quais citamos os diferentes fatores de desempenho dos componentes *worker* e o escalonamento de tarefas realizado pelo sistema JoiN, dos quais depende diretamente o fluxo da execução das tarefas e, consequentemente, o tráfego de rede. Na Fig. 4.35, quando aumentamos o número de componentes *worker*, a sobrecarga absoluta tende a subir para os experimentos com 100 e 200 tarefas, e tende a cair para os experimentos com 400 tarefas.

Na Fig. 4.36, quando aumentamos o número de tarefas, a sobrecarga absoluta parece subir para os experimentos com 4 e 16 componentes *worker* e, para os experimentos com 32 componentes *worker* não apresenta comportamento regular. Entretanto, ao observarmos os gráficos da sobrecarga relativa nestas duas figuras, identificamos que a sobrecarga oscila em torno de valores pequenos.

Estas informações nos levam a concluir que a ferramanta de monitoramento também não impacta significativamente o tráfego de rede do sistema JoiN na execução desta aplicação.

4.5 Conclusões

Neste capítulo foi apresentada a avaliação da ferramenta JoiNMon e do impacto que o monitoramento causa no sistema JoiN.

Foi feita a classificação da ferramenta de monitoramento conforme requisitos, características e taxonomia apresentados anteriormente, buscando avaliar a adequação da ferramenta à arquitetura GMA e as facilidades que o monitoramento oferece. Nesta avaliação a ferramenta foi enquadrada na categoria que engloba as ferramentas desenvolvidas para atender características e necessidades de um sistema em específico (JoiN), projetado e implementado conforme o modelo produtor/consumidor com mediador em que é utilizado um sistema de eventos. Conforme padronização definida pela GMA, a ferramenta JoiNMon pode ser considerada "nível 2", pois a funcionalidade dos sensores é embutida nos produtores e os republicadores têm como função pré-definida tratar, filtrar e repassar os eventos para o nível hierárquico superior, de acordo com a arquitetura do sistema JoiN.

A seguir foi feita uma comparação dos pontos fortes e fracos da ferramenta JoiNMon e das ferramentas Ganglia, MonALISA, JAMM e R-GMA. Estas ferramentas foram selecionadas para comparação devido a sua ampla utilização. Pode-se perceber desta comparação que a ferramenta JoiNMon apresenta características próprias, necessárias para integração ao sistema JoiN, e características em comum com as outras ferramentas e também que é possível, por meio de futuras implementações, que JoiNMon venha contemplar outros pontos fortes destacados nas demais ferramentas.

Na avaliação do impacto da ferramenta JoiNMon sobre o sistema JoiN foram considerados e discutidos: funcionalidade; confiabilidade e tolerância a falhas; manutenibilidade e extensibilidade; usabilidade, por meio da interface gráfica; portabilidade.

Para avaliação da eficiência foram realizados experimentos e analisados os resultados obtidos, em especial com relação ao impacto no desempenho de diferentes tipos de aplicações e no no tráfego de rede das mesmas.

Após a análise dos resultados obtidos nestes experimentos podemos ressaltar alguns pontos rele-

vantes. Do ponto de vista do usuário, permite o acompanhamento da execução de suas aplicações, do escalonamento e replicação de tarefas entre os componentes *worker*, sem causar impactos significativos no tempo de execução destas aplicações. Do ponto de vista dos colaboradores, permite acompanhar a utilização de seus computadores na execução das aplicações e avaliar a sua contribuição para o sistema. Do ponto de vista de seus administradores, facilita a obtenção de informações sobre o estado e o desempenho do sistema. Sendo assim, podemos concluir que a ferramenta possui uma relação custo/benefício reduzida, considerando a baixa sobrecarga imposta sobre o tempo de processamento das aplicações.

Com relação ao tráfego de rede, podemos observar que a sobrecarga introduzida por JoiNMon pode variar em função de fatores relativos ao escalonamento e à replicação de tarefas do sistema JoiN e isto foi enfatizado na avaliação dos experimentos com a aplicação para multiplicação de matrizes. Contudo, é difícil determinar os limites aceitáveis em um ambiente tão dinâmico. Após a análise dos resultados obtidos nos experimentos com as aplicações para cálculo de π, da classe $R_{cc} >> 1$, e multiplicação de matrizes, da classe $R_{cc} > 1$, podemos concluir que a sobrecarga no tráfego de rede causada pela ferramenta de monitoramento não interfere na escalabilidade do sistema JoiN. Este resultado pôde ser obtido devido à arquitetura da ferramenta JoiNMon e à eficiência no tratamento do grande volume de dados monitorados.

A configuração do sistema JoiN, em todos os experimentos conduzidos para avaliação da eficiência da ferramenta JoiNMon, contém um único grupo e, conseqüentemente, um componente *coordinator*. Embora prevista na arquitetura do sistema JoiN, a configuração com mais de um grupo ainda não disponível na versão do sistema utilizada nestes experimentos. A arquitetura e a implementação da ferramenta de monitoramento prevêem sua utilização em configurações com diversos grupos e, desta forma, espera-se que a eficiênca da ferramenta seja mantida também nesta condição.

Assim sendo, é possível afirmar que a ferramenta JoiNMon possibilita o monitoramento da execução de aplicações e da utilização dos computadores no sistema JoiN sem interferir na escalabilidade do sistema e causando baixo impacto no tempo necessário para o processamento das aplicações.

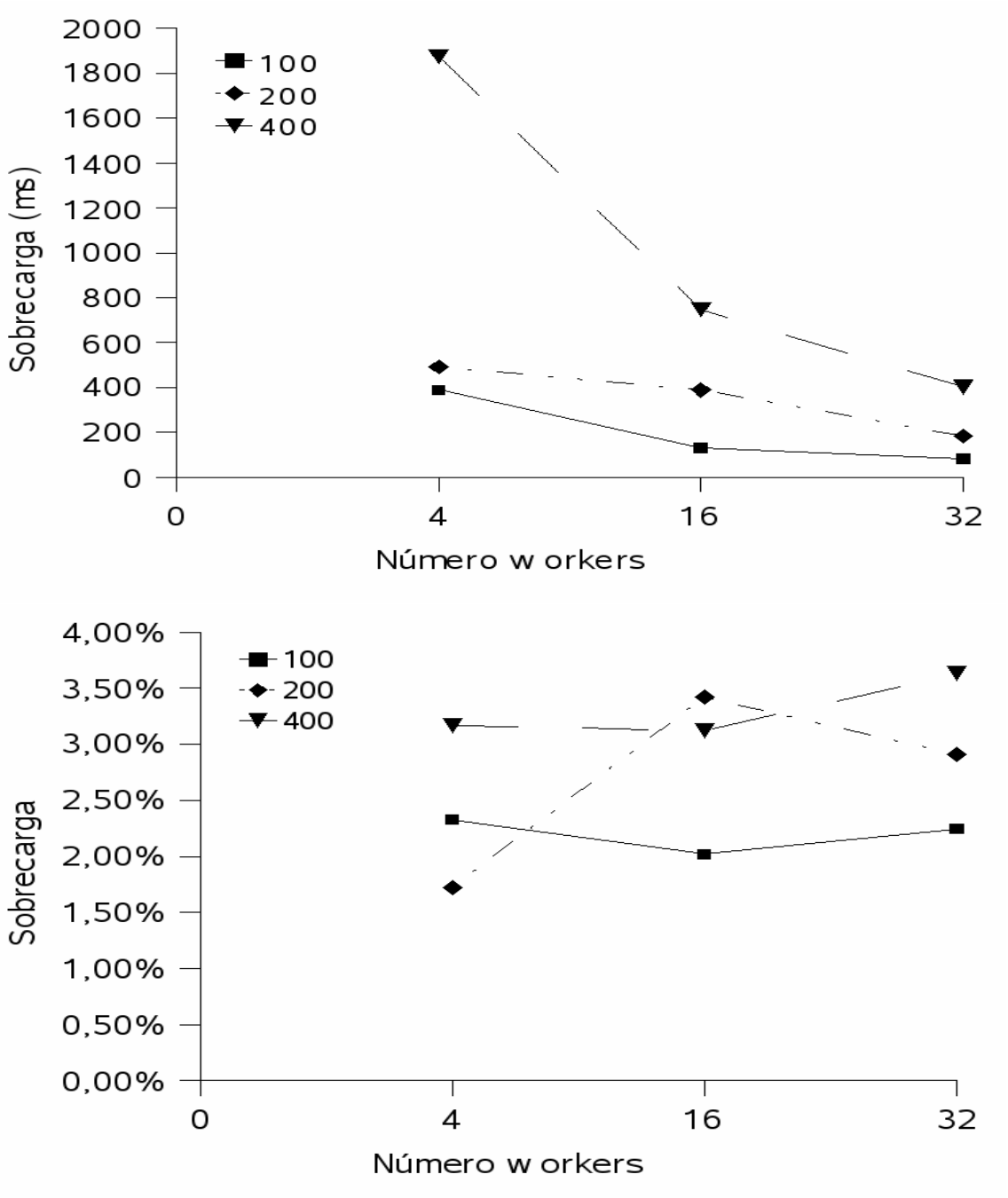

Fig. 4.32: Multiplicação de matrizes de tamanho fixo - Sobrecarga tempo processamento (em função do número de *workers*)

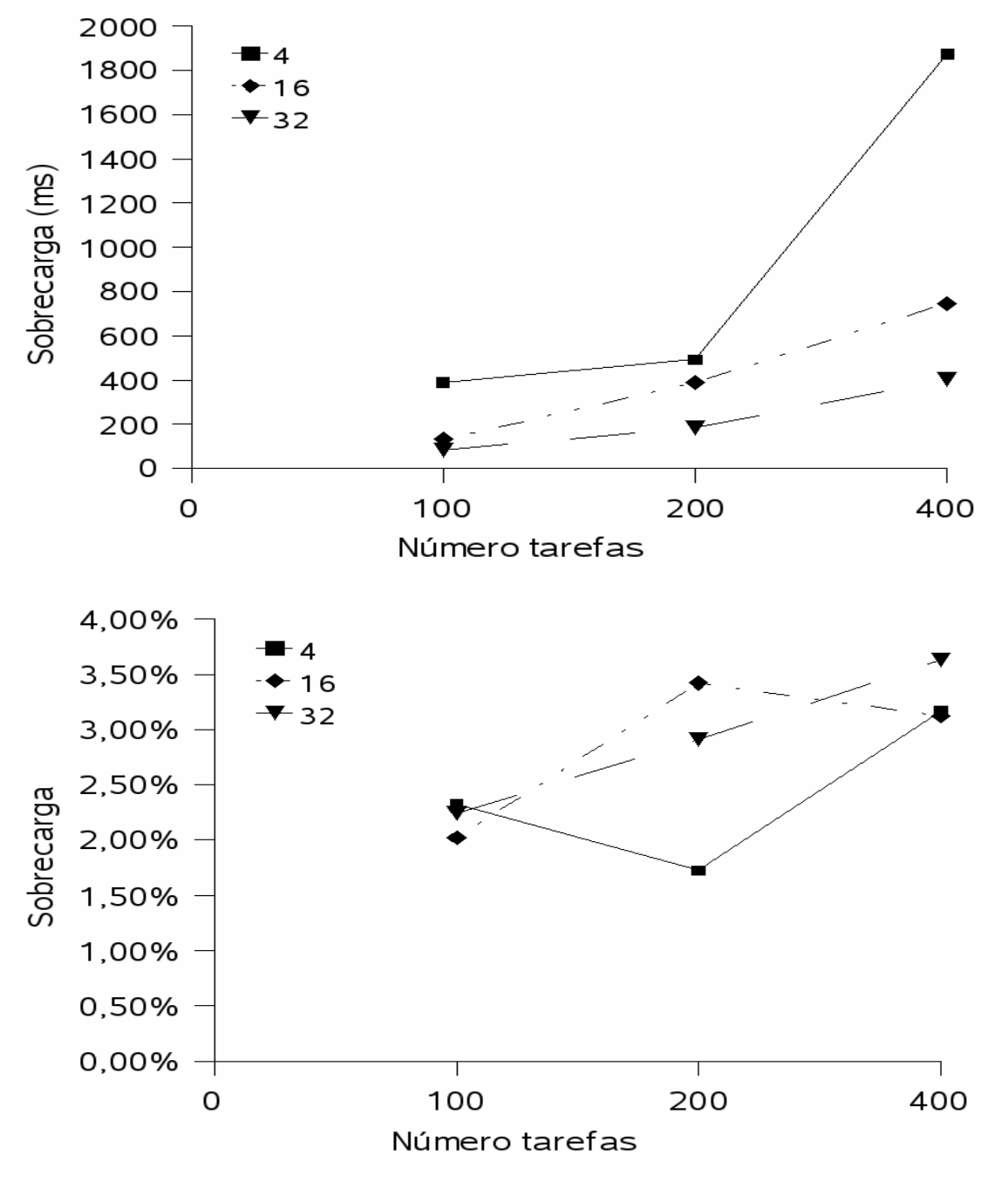

Fig. 4.33: Multiplicação de matrizes de tamanho fixo - Sobrecarga tempo processamento (em função do número de tarefas)

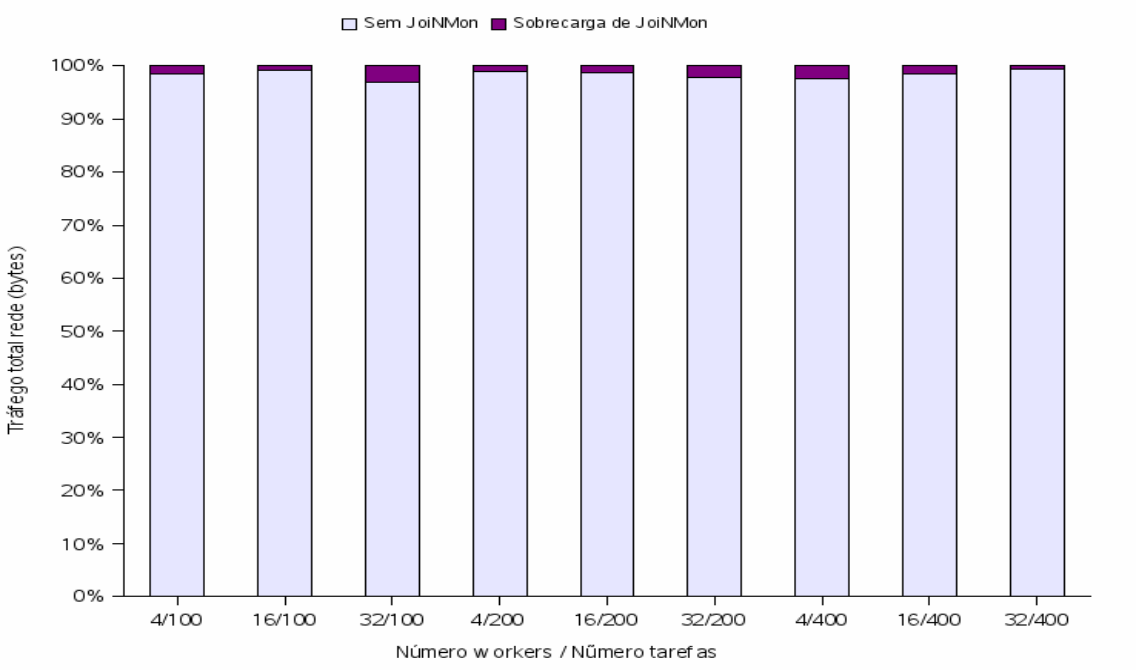

Fig. 4.34: Multiplicação de matrizes de tamanho fixo - Sobrecarga no tráfego de rede

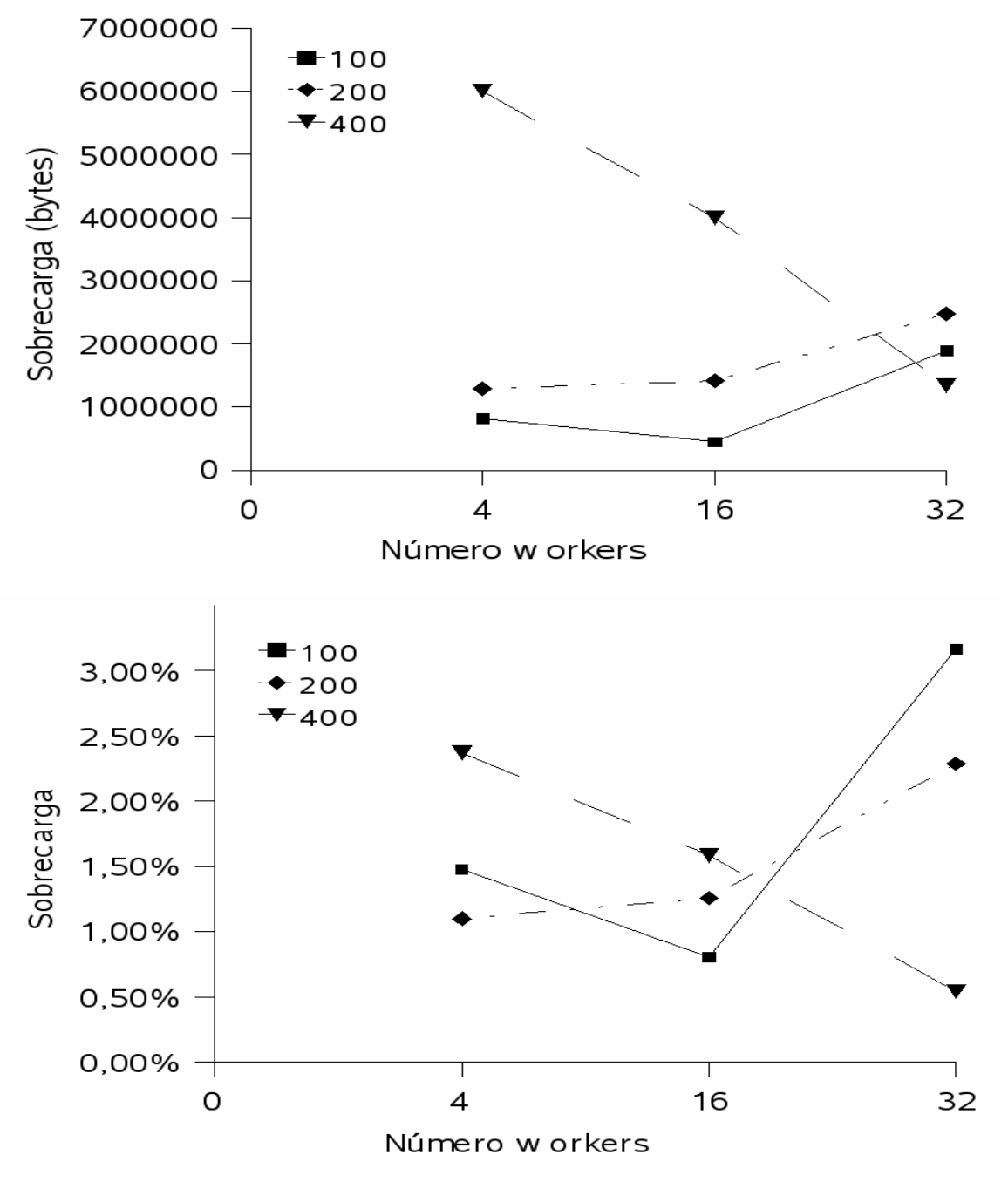

Fig. 4.35: Multiplicação de matrizes de tamanho fixo - Sobrecarga tráfego rede (em função do número de *workers*)

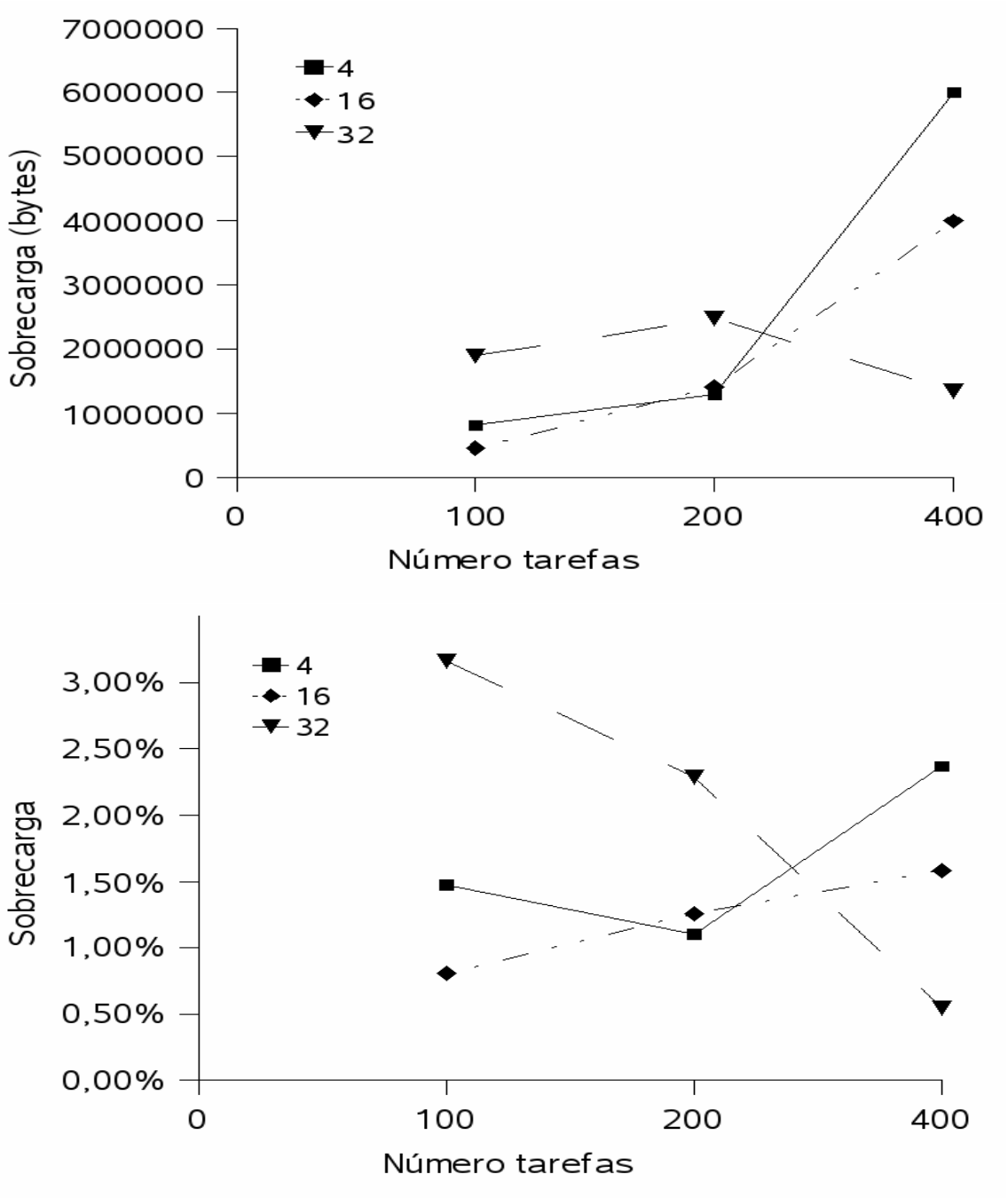

Fig. 4.36: Multiplicação de matrizes de tamanho fixo - Sobrecarga tráfego rede (em função do número de tarefas)

Capítulo 5

Conclusões e trabalhos futuros

Neste trabalho foram apresentadas a proposta, a implementação e a avaliação de uma ferramenta, baseada na arquitetura GMA, para monitoramento do sistema JoiN, em desenvolvimento na FEEC/Unicamp.

Neste capítulo são apresentadas as conclusões deste trabalho destacando a relevância dos resultados obtidos e a contribuição para a identificação das características e para a análise de requisitos de ferramantas para monitoramento de ambientes heterogêneos e dinâmicos. São também apresentadas sugestões e otimizações que poderão dar continuidade ao trabalho realizado.

5.1 Conclusões

5.1.1 Como avaliar uma ferramenta de monitoramento para um ambiente de grade computacional?

Esta pergunta esteve presente desde o início deste trabalho e serviu como orientação na condução dos estudos e na identificação dos principais requisitos e das características que devem ser avaliadas neste tipo de ferramenta. Não é possível recomendar um roteiro para avaliação de ferramentas de monitoramento, uma vez que cada ambiente de grade computacional a ser monitorado tem também características e necessidades próprias. Apresentamos então uma estratégia que permite abordar este tema complexo de uma maneira mais simples.

Em geral, a primeira análise que fazemos diz respeito à forma de apresentação e ao conteúdo das informações que a ferramenta torna disponíveis. Para que uma ferramenta seja bem avaliada por seus usuários deve apresentar com clareza as informações que ele procura e estas informações devem poder ser facilmente encontradas. Como não pretendemos explorar aspectos teóricos de IHC, que trata da apresentação das informações por meio de uma interface que seja entendida por todos os usuários, fizemos a avaliação da disponibilidade das informações e do conteúdo apresentado. Com relação à disponibilidade, avaliamos se a ferramenta de monitoramento permite o acesso a informações em tempo-real, de maneira a viabilizar a análise do estado atual da grade computacional, bem como de estados passados. Isto possibilita analisar o comportamento do ambiente e a utilização deste na execução de aplicações.

Esta primeira avaliação é feita em alto nível, pois estamos analisando o resultado final da ferra-

menta de monitoramento. Precisamos então avaliar cuidadosamente a implementação da ferramenta, em especial os aspectos relacionados à sua arquitetura e à capacidade para tratamento de um grande volume de dados. É muito importante analisar como são feitos a obtenção, a transmissão e o tratamento dos dados, pois a ferramenta de monitoramento não deve degradar o desempenho e não deve interferir na escalibilidade do sistema. Outro aspecto a que devemos estar atentos é relativo à segurança que a ferramenta oferece no tratamento da das informações coletadas. Finalmente, mas não menos importante que os demais, devemos avaliar a flexibilidade da ferramenta, ou seja, se é possível o monitoramento de novas informações e se isto pode ser feito facilmente.

5.1.2 Qual ferramenta de monitoramento utilizar no sistema JoiN?

Após o estudo de conceitos e técnicas para monitoramento em ambientes de computação em grade, da identificação e avaliação de trabalhos relacionados e ferramentas existentes, foi possível levantar pontos fortes e fracos e concluir que não havia, entre as ferramentas analisadas, uma que se adequasse às necessidades e à arquitetura do sistema JoiN.

Partimos então para o projeto de uma nova ferramenta que buscasse agregar as vantagens identificadas nesta análise. Acabamos nos baseando na arquitetura GMA que foi proposta pelo Performance Working Group do GGF como padrão para o monitoramento de recursos e de aplicações em ambientes de computação em grade.

Os principais desafios encontrados no projeto da ferramenta, chamada JoiNMon, foram relativos à obtenção, tratamento e disponibilização das informações pois, como o volume de dados pode ser muito grande, a eficiência da ferramenta depende da forma de tratamento dos dados. Assim como proposto na arquitetura GMA, a geração e o tratamento de eventos na ferramenta JoiNMon são feitos de acordo com o modelo produtor/consumidor, utilizando eventos para transmissão de informações.

Começamos pela identificação das informações relevantes para o monitoramento, seguida da avaliação da melhor forma de obtenção e transmissão destas informações. Como temos acesso ao código fonte do sistema JoiN, implementamos os sensores/produtores por meio de alterações nos serviços JoiN. As informações são obtidas, padronizadas e formatadas por estes serviços, gerando eventos. Estes, por sua vez, são transmitidos por meio do serviço *Communicator*. Foram feitas modificações no serviço *EventManager* para tratamento dos eventos e para sua adequação à arquitetura GMA. O *EventManager* é o responsável pelo serviço de diretório, identificação dos eventos, produtores e consumidores associados. O componente consumidor foi implementado por meio do novo serviço *JoiNMonitor* que recebe as notificações sobre a ocorrência dos eventos e reage a estes: obtém as informações a serem monitoradas e faz o armazenamento das mesmas de forma eficiente em uma base de dados relacional. Nesta base de dados são mantidas informações que refletem o estado atual do sistema e aplicações, bem como informações históricas sobre estados e aplicações passados. A interface gráfica, para visualização das informações, foi desenvolvida em JSP, o que permite a utilização em navegadores *web* e facilita o acesso às informações previamente armazenadas.

5.1.3 Como avaliar a ferramenta de monitoramento para o sistema JoiN?

Para avaliação da ferramenta JoiNMon analisamos as funcionalidades implementadas e também verificamos se, mesmo após sua implementação, foram mantidas as premissas básicas do sistema JoiN com relação à escalabilidade, à portabilidade e ao desempenho. Por meio da ferramenta JoiNMon os

usuários e colaboradores do sistema JoiN passaram a ter acesso em tempo-real a informações sobre o estado do sistema, e também a informações sobre o histórico da execução de aplicações, sem que isto cause prejuízos ao desempenho ou escalabilidade do sistema. A sobrecarga que a ferramenta introduz no tempo de execução das aplicações foi avaliada por meio da realização de experimentos cujos resultados nos permitem concluir que a ferramenta JoiNMon não causa gargalos no sistema e suporta o aumento do número de recursos (colaboradores) e de eventos (tarefas).

Podemos então verificar se foram atendidos os objetivos apresentados na seção 3.5.1:

1. identificar mais rapidamente a existência de problemas ou gargalos:

   a ferramenta JoiNMon não implementa funcionalidades para a identificação de problemas ou gargalos, entretanto, com base nas informações que disponibiliza sobre a execução de aplicações e a utilização dos componentes *worker* pode auxiliar os administradores e usuários nesta tarefa;
2. identificar os recursos disponíveis para a execução das aplicações:

   a ferramenta JoiNMon disponibiliza as informações sobre recursos disponíveis para a execução de aplicações;
3. identificar parâmetros importantes para o escalonamento de processos e balanceamento de carga:

   por meio das informações que a ferramenta JoiNMon disponibiliza sobre a utilização de componentes *worker* na execução de aplicações, um programador do sistema JoiN pode avaliar a eficiência do escalonamento de processos e do balanceamento de carga e, se necessário, configurar parâmetros que alterem o mecanismo de distribuição de tarefas;
4. acompanhar a execução de aplicações com mais facilidade:

   a ferramenta JoiNMon disponibiliza as informações sobre a execução de aplicações;
5. identificar e acompanhar a utilização do sistema pelos seus usuários:

   a ferramenta JoiNMon disponibiliza informações sobre a utilização do sistema JoiN pelos seus usuários;
6. identificar e acompanhar a utilização de cada um dos computadores que foram disponibilizados para uso pelo sistema:

   a ferramenta JoiNMon disponibiliza informações sobre a participação dos colaboradores no sistema JoiN, bem como da utilização dos computadores por eles disponibilizados;
7. obter informações sobre aplicações já encerradas:

   a ferramenta JoiNMon disponibiliza informações sobre aplicações já executadas no sistema JoiN.

Os resultados esperados ao iniciarmos este trabalho foram obtidos de forma satisfatória com a implementação da ferramenta JoiNMon. Além disto, os conhecimentos adquiridos nos estudos e análises realizados contribuem para que possamos avaliar futuramente funcionalidades e características de outras ferramantas, já existentes ou em desenvolvimento, para monitoramento de ambientes de computação em grade.

5.2 Trabalhos futuros

Após a conclusão do desenvolvimento da ferramenta proposta neste trabalho é possível apresentar sugestões e otimizações que podem ser implementadas, com o objetivo de aumentar a eficiência e a abrangência da ferramenta:

- desenvolvimento de uma *interface* para acesso ao banco de dados relacional e recuperação das informações, com o objetivo de permitir o acesso às informações por meio de outras ferramentas de apresentação de dados;
- inclusão de funcionalidades para monitoramento dos recursos de armazenamento e de rede utilizados pelo sistema JoiN;
- desenvolvimento de ferramentas para análise de desempenho e identificação de problemas. Estas ferramentas podem valer-se das informações armazenadas na base de dados mantida pela ferramenta de monitoramento, tratá-las e apresentá-las sob a perspectiva necessária para estudos de desempenho do sistema, identificação de problemas, predição de desempenho e direcionamento na utilização da grade;
- implementação de APIs que facilitem a inclusão e tratamento de novos eventos, pois apesar do planejamento realizado e da identificação das informações mais relevantes para a ferramenta de monitoramento, poderão surgir novas necessidades e, nestes casos, o uso de APIs para tratamento destes novos eventos será muito importante;
- implementação de funcionalidades para o monitoramento de informações que auxiliem o desenvolvimento de novos serviços para o sistema JoiN;
- implementação de mecanismos de segurança na coleta, distribuição e acesso aos dados monitorados. Apesar de sua importância, este aspecto não foi explorado pela ferramenta de monitoramento, pois a ferramenta de segurança ainda não estava operacional no momento de desenvolvimento e implementação do monitoramento;
- implementação de mecanismos para replicação das informações monitoradas para, desta forma, aumentar a tolerância a falhas;
- avaliar a formatação dos eventos e a forma de utilização do mecanismo de *cache* e analisar a possibilidade de utilização de mecanismos de *piggybacking* para transmissão de eventos, anexando-os a pacotes de dados que trafegam normalmente no sistema JoiN;
- implementar novas funcionalidades na interface gráfica relativas à apresentação de gráficos, de análises estatísticas e de informações gerenciais.

Referências Bibliográficas

[1] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–4, 2001. http://www.globus.org/alliance/publications/papers/anatomy.pdf.

[2] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. 2002. http://www.globus.org/research/papers/ogsa.pdf.

[3] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2005.

[4] David P. Anderson. Public computing: Reconnecting people to science. *Conference on Shared Knowledge and the Web*, November 2003. http://boinc.berkeley.edu/boinc2.pdf.

[5] Eduardo Javier Huerta Yero, Fabiano de Oliveira Lucchese, Francisco Sérgio Sambatti, Miriam von Zuben, and Marco Aurélio Amaral Henriques. Join: The implementation of a java-based massively parallel grid. *Future Generation Computer Systems*, 21(5):791–810, May 2005.

[6] Globus Alliance. The globus alliance. Página na internet, University of Chicago, Outubro 2005. http://www.globus.org/alliance/about.php/.

[7] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997. ftp://ftp.globus.org/pub/globus/papers/globus.pdf.

[8] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In Springer-Verlag LNCS 3779, editor, *IFIP International Conference on Network and Parallel Computing*, pages 2–13, 2005. http://www.globus.org/alliance/publications/papers/IFIP-2005.pdf/.

[9] I. Foster. A globus toolkit primer. draft, Globus Allience, 2005. http://www.globus.org/toolkit/docs/4.0/key/GT4_Primer_0.6.pdf.

[10] AccessGrid. The access grid project. Página na internet, Argonne National Laboratory - Mathematics and Computer Science Division, Julho 2008. http://www.accessgrid.org/.

[11] OSG. Open science grid (osg) consortium. Página na internet, U.S. universities and national laboratories, Julho 2008. http://www.opensciencegrid.org/.

[12] TeraGrid. Teragrid project by the national science foundation. Página na internet, National Science Foundation, Julho 2008. http://www.teragrid.org/.

[13] EGEE. Enabling grids for e-science (egee). Página na internet, European Commission, Julho 2008. http://public.eu-egee.org/.

[14] Karl Czajkowski, Carl Kesselman, Steven Fitzgerald, and Ian Foster. Grid information services for distributed resource sharing. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01)*, page 0181, 2001. http://www.globus.org/alliance/publications/papers/MDS-HPDC.pdf/.

[15] Sardes. Sardes - system architecture for reflective distributed environments. Página na internet, Inria - Institute National de Recherche en Informatique et en Automatique, Julho 2008. http://ralyx.inria.fr/2006/Fiches/sardes/sardes.html.

[16] Project-Team Sardes. System architecture for reflective distributed environments. Activity report, Inria - Institute National de Recherche en Informatique et en Automatique, 2003. http://www.inria.fr/rapportsactivite/RA2003/sardes2003/sardes.ps.gz.

[17] Dream. A component-based communication framework. Página na internet, Julho 2008. http://dream.objectweb.org/.

[18] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder. Unicore – from project results to production grids. Technical report, 2005. http://unicore.sourceforge.net/docs/paper_streit_et_al.pdf.

[19] D. Erwin. Unicore and eurogrid: Grid computing in europe. *TERENA Networking Conference 2001*, 2001. http://www.eurogrid.org/documents/Terena2001.pdf.

[20] Open computing grid for molecular science and engineering. Página na internet, Forschungszentrum Jülich GmbH (FZJ), Germany, Julho 2008. http://www.openmolgrid.org/.

[21] Vertically integrated optical testbed for large applications in dfn. Página na internet, Deutsches Forschungsnetz (Germany's National Research and Education Network), Germany, Julho 2008. http://www.viola-testbed.de/index.php?id=goalsengl.

[22] National research grid initiative. Página na internet, NII- National Institute of Informatics, Japan, Julho 2008. http://www.naregi.org/index_e.html.

[23] Unicore. Simon - site functionality monitoring tool. Página na internet, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany, Julho 2008. http://www.unicore.eu/download/unicore5/.

[24] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, November 2002.

[25] Seti@home. Search for extra-terrestrial intelligence. Página na internet, University of California - Berkeley, Julho 2008. `http://setiweb.ssl.berkeley.edu/`.

[26] Msetimon - multi seti@home monitor. Página na internet, Julho 2008. `http://msetimon.sourceforge.net/`.

[27] Seti stats analyser. Página na internet, Julho 2008. `http://members.lycos.co.uk/chrisdavis/seti/stats/about.html`.

[28] David P. Anderson. Boinc: A system for public-resource computing and storage. In IEEE Computer Society Press, editor, *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages xxx–yyy, Pittsburgh, USA, November 2004. `http://boinc.berkeley.edu/grid_paper_04.pdf/`.

[29] Boinctray. Página na internet, Julho 2008. `http://www.iki.fi/edu/programs/boinctray/`.

[30] Boincprog. Página na internet, Julho 2008. `http://www2.gol.com/users/trane/programming/programming.html`.

[31] Michael Litzkow, Miron Livny, and Matt Mutka. Condor - a hunter of idle workstations. In IEEE Computer Society Press, editor, *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, Washington, D.C., USA, June 1988.

[32] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke. Condor-g: A computation management agent for multi-institutional grids. Technical report, San Francisco, California, USA, August 2001. `http://www.cs.wisc.edu/condor/doc/condorg-hpdc10.pdf/`.

[33] Hawkeye. Hawkeye - a monitoring and management tool for distributed systems. Página na internet, Condor Hawkeye Project - University of Wisconsin - Madison, Julho 2008. `http://www.cs.wisc.edu/condor/hawkeye`.

[34] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, and Jr. Paul F. Reynolds. Legion: The next logical step toward a nationwide virtual computer. Technical Report CS-94-21, Charlottesville, VA, USA, 1994. `http://legion.virginia.edu/papers/CS-94-21.ps`.

[35] Steve J. Chapin, Dimitrios Katramatos, John Karpovich, and Andrew Grimshaw. Resource management in legion. *Future Generation Computer Systems*, 15(5-6):583–594, 1999. `http://www.legion.virginia.edu/papers/legionrm.pdf`.

[36] Ourgrid. Open source software for peer-to-peer grid computing. Página na internet, UFCG, Agosto 2008. `http://www.ourgrid.org/`.

[37] Nazareno Andrade, Lauro Costa, Guilherme Germoglio, and Walfredo Cirne. Peer-to-peer grid computing with the ourgrid community. In *in 23rd Brazilian Symposium on Computer Networks (SBRC 2005) - 4th Special Tools Session*, 2005. http://www.ourgrid.org/twiki-public/pub/OG/OurPublications/salao2005.pdf.

[38] InteGrade. Página na internet, Agosto 2008. http://www.integrade.org.br/portal.

[39] Andrei Goldchleger, Fabio Kon, Alfredo Goldman, and Marcelo Finger. Integrade: Object-oriented grid middleware leveraging idle computing power of desktop machines. *Concurrency and Computation: Practice and Experience*, 16:16–449, 2004.

[40] Jakob Gregor Pedersen and Christian Ulrik Søttrup. Developing distributed computing solutions combining grid computing and public computing. Tese de mestrado, Department of Computer Science, University of Copenhagen, Março 2005. http://www.fatbat.dk/thesis/boincThesis.pdf.

[41] Ken'ichiro Shirose, Satoshi Matsuoka, Hidemoto Nakada, and Hirotaka Ogawa. Autonomous configuration of grid monitoring systems. In *SAINT-W '04: Proceedings of the 2004 Symposium on Applications and the Internet-Workshops (SAINT 2004 Workshops)*, pages 651–657, Washington, DC, USA, 2004. IEEE Computer Society.

[42] C. Röder, T. Ludwig, and A. Bode. Flexible status measurement in heterogeneous environments. In H. R. Arabnia, editor, *Proceedings of the Fifth International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA'98), Las Vegas, USA, 1998*, volume volume I, pages 247–254. CSREA Press, 1998. http://www.in.tum.de/~ludwig/LN/papers/tl9806/PS/article.ps.gz.

[43] Marcia Zangrilli and Bruce B. Lowekamp. Comparing passive network monitoring of grid application traffic withs active probes. In *Proceedings of the Fourth International Workshop on Grid Computing, 2003*, pages 84–91, 2003. http://www.cs.wm.edu/~mazang/passive-active-grid2003.pdf.

[44] D. Lee, J. Dongarra, and R. Ramakrishna. Visperf: Monitoring tool for grid computing. In Peter M. A. Sloot, David Abramson, Alexander V. Bogdanov, Jack Dongarra, Albert Y. Zomaya, and Yuri E. Gorbachev, editors, *Computational Science - ICCS 2003, International Conference, Melbourne, Australia and St. Petersburg, Russia, June 2-4, 2003. Proceedings, Part III*, volume 2659 of *Lecture Notes in Computer Science*, pages 233–243. Springer, 2003. http://icl.cs.utk.edu/projectsfiles/netsolve/pubs/visperf.pdf.

[45] E. V. Zinov'ev and N. A. Piziks. Design principles for an interactive monitor of network information resources. *Autom. Control Comput. Sci.*, 24(1):59–63, 1990.

[46] D. Gunter. An architecture and protocols for grid performance monitoring. Página na internet, CERN: Geneva, Switzerland, Julho 2008. http://www-didc.lbl.gov/presentations/CERN.GMAFuture.ppt/.

[47] Inc Sun Microsystems. *The Java Tutorial*, página na internet Writing Event Listeners. Julho 2008. http://java.sun.com/docs/books/tutorial/uiswing/events/.

[48] Antonio Carzaniga. Architectures for an event notification service scalable to wide-area networks. Phd thesis, Politecnico di Milano, Italy, Dec 1998. http://serl.cs.colorado.edu/~carzanig/papers/.

[49] Rene Meier and Vinny Cahill. Taxonomy of distributed event-based programming systems. *The Computer Journal*, 48(5):602–626, January 2005. http://comjnl.oxfordjournals.org/cgi/reprint/48/5/602?ijkey=mveLC7U2u33I2ad\verb!&!keytype=ref.

[50] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski. A grid monitoring architecture. Informational document, GGF - Global Grid Forum, 2002. http://www.ggf.org/documents/GFD/GFD-I.7.pdf.

[51] Serafeim Zanikolas and Rizos Sakellariou. A taxonomy of grid monitoring systems. *Future Gener. Comput. Syst.*, 21(1):163–188, 2005. http://citeseer.ist.psu.edu/foster96globus.html.

[52] M. Gerndt, R. Wismüller, Z. Balaton, G. Gombás, P. Kacsuk, Zs. Németh, N. Podhorszki, H-L. Truong, T. Fahringer, M. Bubak, E. Laure, and T. Margalef. Performance tools for the grid: State of the art and future - version 1.0, 7.01.04. White paper, APART - Automatic Performance Analysis: Real Tools, 2004. http://urza.lpds.sztaki.hu/~zsnemeth/apart/repository/gridtools.pdf.

[53] Rajesh Raman, Miron Livny, and Marvin Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998, Chicago, IL.*, Chicago, IL. http://www.cs.wisc.edu/condor/doc/hpdc98.ps.

[54] Vivien Quéma, Renaud Lachaize, and Emmanuel Cecchet. An asynchronous middleware for grid resource monitoring. *Concurrency and Computation: Practice and Experience*, 16(5):523–534, March 2004. http://sci-serv.inrialpes.fr/papers/files/03-Quema-Mgc.pdf.

[55] S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G.L. Rubini, G. Tortone, and M.C. Vistoli. Gridice: a monitoring service for grid systems. *Future Generation Computer Systems*, 21(4):559–571, April 2005. http://infnforge.cnaf.infn.it/gridice/doc/gridice4FGCS_revised_1.pdf.

[56] Lemon. Lemon - lhc era monitoring. Página na internet, CERN - ELFms - Extremely Large Fabric management system, Julho 2008. http://lemon.web.cern.ch/lemon/.

[57] Globus Alliance. Monitoring & discovery system (mds). Página na internet, Julho 2008. http://www.globus.org/toolkit/mds/.

[58] Nagios. Network monitor. Página na internet, Julho 2008. http://www.nagios.org/.

[59] PostgreSQL. open source relational database system. Página na internet, Julho 2008. http://www.postgresql.org/.

[60] PHP. Hypertext preprocessor. Página na internet, Julho 2008. http://www.php.net/.

[61] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Sagi, Z. Shi, and S. Vadhiyar. Users' guide to netsolve v1.4. Technical report, Computer Science Dept., University of Tennessee, Knoxville, TN, Julho 2008. http://icl.cs.utk.edu/netsolvedev/documents/ugv1.4/html/index.html.

[62] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, July 2004. http://www.cs.berkeley.edu/~massie/papers/science.pdf.

[63] H. B. Newman, I.C. Legrand, P. Galvez, R. Voicu, and C. Cirstoiu. Monalisa : A distributed monitoring service architecture. In *Proceedings of the Conference on Computing in High Energy and Nuclear Physics (CHEP 2003)*, La Jolla, California (USA), March 2003. http://monalisa.caltech.edu/documentation/MOET001.pdf.

[64] Inc Sun Microsystems. *Java Technology Java 2 Platform, Enterprise Edition (J2EE)*, página na internet Jini Specifications and API Archive. Julho 2008. http://java.sun.com/products/jini/.

[65] Andrew W. Cooke, Alasdair J. G. Gray, Lisha Ma, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Rob Byrom, Laurence Field, Steve Hicks, Jason Leake, Manish Soni, Antony J. Wilson, Roney Cordenonsi, Linda Cornwall, Abdeslem Djaoui, Steve Fisher, Norbert Podhorszki, Brian A. Coghlan, Stuart Kenny, and David O'Callaghan. R-gma: An information integration system for grid monitoring. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *Proceedings of the 11th International Conference on Cooperative Information Systems - CoopIS/DOA/ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 462–481, Catania, Sicily (Italy), November 2003. Springer.

[66] F. Bonnassieux, R. Harakaly, and P. Primet. Mapcenter: an open grid status visualization tool. In *Proceedings of the 15th ISCA International Conference on Parallel and Distributed Computing Systems (PDCS 2002), Louisville, KY, USA, Sep 2002*, 2002. http://www.atlasgrid.bnl.gov/mapcenter/mapcenter.pdf.

[67] R. L. Ribler, J. S. Vetter, Huseyin Simitci, and Daniel A. Reed. Autopilot: Adaptive control of distributed applications. In *Proceedings of the Seventh IEEE Symposium on High-Performance Distributed Computing (HPDC)*, pages 172–179, 1998. http://citeseer.ist.psu.edu/ribler98autopilot.html.

[68] W. Smith. A framework for control and observation in distributed environments. Technical report, NASA Advanced Supercomputing Division, July 2001. http://www.nas.nasa.gov/News/Techreports/2001/PDF/nas-01-006.pdf.

[69] M.A. Baker and G.C. Smith. Gridrm: an extensible resource monitoring system. In *Proceedings of the IEEE International Cluster Computing Conference*, pages 207–214, 2003. http://ieeexplore.ieee.org/iel5/8878/28041/01253317.pdf.

[70] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski. A fault detection service for wide area distributed computations. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, pages 268–278, 1998. http://www.springerlink.com/index/N4437020K35X6350.pdf.

[71] Brian Tierney, Brian Crowley, Dan Gunter, Jason Lee, and Mary Thompson. A monitoring sensor management system for grid environments. *Cluster Computing*, 4(1):19–28, 2001. http://www-didc.lbl.gov/papers/JAMM.HPDC00.pdf.

[72] Brian Tierney and Dan Gunter. Netlogger: A toolkit for distributed system performance tuning and debugging. draft, Distribute Systems Department - Lawrence Berkeley National Laboratory, 2002. http://dsd.lbl.gov/publications/NetLogger.overview.pdf.

[73] B. Balis, M. Bubak, T. Szepieniec, R. Wismüller, and M. Radecki. Monitoring grid applications with grid-enabled omis monitor. In *Proceedings of the First European Across Grids Conference*, pages 230–239, 2004. http://citeseer.ist.psu.edu/738135.html.

[74] T. DeWitt, T. Gross, B. Lowekamp, N. Miller, P. Steenkiste, J. Subhlok, and D. Sutherland. Remos: A resource monitoring system for network-aware applications. Technical report, Carnegie Mellon University, December 1998. http://www-2.cs.cmu.edu/afs/cs/project/cmcl/www/remulac/remos-tr-2.ps.

[75] H.L. Truong and T. Fahringer. Scalea-g: A unified monitoring and performance analysis system for the grid. Technical report, Institute for Software Science, University of Vienna, November 2003. http://citeseer.ist.psu.edu/truong03scaleag.html.

[76] R. Wolski, N. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5/6):757–768, 1999. http://citeseer.ist.psu.edu/wolski98network.html.

[77] Z. Balaton and G. Gombás. Resource and job monitoring in the grid. In *Proceedings of the Ninth International Euro-Par Conference*, pages 404–411, 2003. http://www.springerlink.com/content/3nhclwv9bcdj36m1/.

[78] P.C. Roth, D.C. Arnold, and B.P. Miller. Mrnet: a software-based multicast/reduction network for scalable tools. In *Proceedings of the Supercomputing 2003 (SC 2003)*, 2003. http://citeseer.ist.psu.edu/roth03mrnet.html.

[79] Bart Jacob. Grid computing: What are the key components? IBM White Paper - developerWorks, Julho 2008. http://http://www.ibm.com/developerworks/grid/newto/.

[80] Luis F. G. Sarmenta. Volunteer computing. Ph.d. thesis, Dept. of Electrical Engineering and Computer Science, MIT, March 2001. http://www.cag.lcs.mit.edu/bayanihan/papers/phd/sarmenta-phd-mit2001.pdf.

[81] Michael Stanton. Computação distribuída pública. Jornal O Estado de São Paulo, 28 de Novembro 2004. http://www21.estadao.com.br/tecnologia/coluna/stanton/2004/nov/28/107.htm.

[82] Eduardo J. Huerta Yero and Marco A. Amaral Henriques. A method to solve the scalability problem in managing massively parallel processing on the internet. In *Seventh Euromicro Workshop on Parallel and Distributed Processing*, pages 256–262, 1998.

[83] Eduardo Javier Huerta Yero. Estudo sobre processamento maciçamente paralelo na internet. Tese de doutorado, Universidade Estadual de Campinas. Faculdade de Engenharia Eletrica e de Computação, Julho 2003.

[84] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification*. Addison Wesley, 1999.

[85] Fabiano de Oliveira Lucchese. Um mecanismo para distribuição de carga em ambientes virtuais de computação maciçamente paralela. Tese de mestrado, Universidade Estadual de Campinas. Faculdade de Engenharia Eletrica e de Computação, Dezembro 2002.

[86] pyGMA. Grid monitoring architecture web-services interfaces in python. Página na internet, The Data Intensive Distributed Computing Research Group (DIDC) - Lawrence Berkeley National Laboratory, Julho 2008. http://www-didc.lbl.gov/pyGMA/.

[87] Guoqing Dong and Weiqin Tong. Ma-gma: A mobile agent-based grid monitor architecture. In *International Conference on Computer Systems and Applications, 2007. AICCSA07. IEEE/ACS*, pages 293–300, 2007. http://ieeexplore.ieee.org/iel5/4230920/4230921/04230972.pdf.

[88] Ludek Matyska Jan Pospísil Jirí Sitera Miroslav Ruda Zdenëk Salvet Michal Vocu Ales Krenek, Daniel Kouril. Logging and bookkeeping ser vice for the datagrid. DataGrid Public documents, Julho 2008. http://lindir.ics.muni.cz/dg_public/DataGrid-01-TEN-0109-1_0.pdf.

[89] Boinc. Berkeley open infrastructure for network computing (boinc). Página na internet, University of California - Berkeley, Julho 2008. http://boinc.netsoft-online.com//.

[90] Allprojectstats.com. Boinc all project stats. Página na internet, Julho 2008. http://www.allprojectstats.com/po.php?projekt=15/.

[91] Microsoft SQL Server. Microsoft sql server. Página na internet, Julho 2008. http://www.microsoft.com/sql//.

[92] DB2. Db2 - database management. Página na internet, IBM, Julho 2008. http://www.ibm.com/db2/.

[93] Oracle. Oracle database. Página na internet, Julho 2008. http://www.oracle.com/database/index.html.

[94] Informix. Informix dynamic server. Página na internet, IBM, Julho 2008. http://www-306.ibm.com/software/data/informix/.

[95] MySQL. Mysql community server. Página na internet, MySQL AB, Julho 2008. http://dev.mysql.com/downloads/mysql/.

[96] Hsqldb. Hsqldb database engine. Página na internet, Julho 2008. http://www.hsqldb.org/.

[97] JDBC. Java database connectivity. Página na internet, Sun, Julho 2008. http://java.sun.com/javase/technologies/database/index.jsp/.

[98] Hugh Beyer and Karen Holtzblatt. *Contextual design : defining customer-centered systems*. San Francisco, Calif. : Morgan Kaufmann, 1998.

[99] Tomcat. Apache tomcat. Página na internet, Apache Jakarta Project, Julho 2008. http://jakarta.apache.org/tomcat/index.html.

[100] Glassfish. Glassfish. Página na internet, GlassFish community, Julho 2008. https://glassfish.dev.java.net/.

[101] Inc Sun Microsystems. *Java SE Desktop Technologies*, página na internet Abstract Window Toolkit (AWT). Julho 2008. http://java.sun.com/j2se/1.5.0/docs/guide/awt/.

[102] Tobi Oetiker. Mrtg - the multi router traffic grapher. Página na internet, OETIKER+PARTNER AG, Julho 2008. http://http://oss.oetiker.ch/mrtg/.

[103] PolePosition. Poleposition. Página na internet, Julho 2008. http://polepos.sourceforge.net/results/html/index.html.

[104] Inc Sun Microsystems. *Java Technology Java 2 Platform, Standard Edition (J2SE)*, página na internet Javadoc Tool. Julho 2008. http://java.sun.com/j2se/javadoc/.

[105] Wireshark. Wireshark. Página na internet, Julho 2008. http://www.wireshark.org/.

[106] Chee Yong Chan and Yuan Ni. Efficient xml data dissemination with piggybacking. In *SIGMOD 07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 737–748, New York, NY, USA, 2007. ACM.

[107] WMS. Workload management system (wms). Página na internet, CERN, Julho 2008. http://glite.web.cern.ch/glite/wms/.

[108] EDG. The european datagrid project (edg). Página na internet, CERN, Julho 2008. http://eu-datagrid.web.cern.ch/.

[109] LCG. Lhc computing grid project(lcg). Página na internet, CERN, Julho 2008. http://lcg.web.cern.ch/LCG/.

[110] Crossgrid. Crossgrid project. Página na internet, Julho 2008. http://www.crossgrid.org/.

[111] RRDtool. Round robin database. Página na internet, Julho 2008. http://oss.oetiker.ch/rrdtool/.

[112] DMF. Distributed monitoring framework (dmf). Página na internet, Lawrence Berkeley National Laboratory, Julho 2008. http://dsd.lbl.gov/DMF/.

[113] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970. http://portal.acm.org/ft_gateway.cfm?id=362685&type=pdf&coll=GUIDE&dl=&CFID=56138463&CFTOKEN=26543357.

[114] Silvia Shimakura. Notas do curso ce003 - estatística ii. Página na internet, Departamento de Estatística-UFPR, Agosto 2008. http://leg.ufpr.br/~silvia/CE003/notes.html.

Apêndice A

Análise de algumas ferramentas de monitoramento para computação em grade

Neste apêndice são apresentadas as principais caracteríticas de algumas das ferramentas existentes atualmente na literatura para monitoramento de ambientes de computação em grade. As ferramentas apresentadas são de "propósito geral", ou seja, não foram desenvolvidas para atender as necessidades de um ambiente ou sistema em particular, e foram analisadas com o objetivo de avaliarmos suas características, suas arquiteturas e as abordagens adotadas nas suas implementações.

JAMM - Java Agents for Monitoring and Management

JAMM [71] (*Java Agents for Monitoring and Management*) é um software desenvolvido em Java que implementa um conjunto de componentes distribuídos que coletam e publicam informações sobre computadores. A arquitetura destes agentes implementa o modelo produtor/consumidor, e é baseada em Java e RMI para obtenção das informações desejadas, que são então publicadas em uma base de dados LDAP. Estes agentes são executados em cada computador conectado ao ambiente e utilizam comandos, ou sensores, para extrair as informações para o monitoramento de redes, de carga dos computadores, de execução de aplicações. As informações coletadas são carregadas na base de dados LDAP, com uma freqüência previamente estabelecida, em geral a cada minuto. A utilização de agentes tem por objetivo a escalabilidade, distribuindo o gerenciamento do grande volume de dados necessários para monitorar ambientes de grade. A ferramenta JAMM é utilizada para a geração de informações que serão posteriormente analisadas por meio da ferramenta NetLogger [72]. Os componentes da ferramenta JAMM são:

- *Sensor*: executado em cada computador conectado ao sistema para obtenção de dados ou eventos a serem monitorados. As informações são transmitidas no formato em que são obtidas, ou então os sensores podem ser configurados para produzir informações, ou mensagens, no formato utilizado pelo Netlogger. Existem alguns sensores pré-codificados, que interagem com comandos Unix, como por exemplo *vmstat*, *netstat*, *ping*, *iperf*, *xntpdc*, *ps* e *iostat*;
- *Logger* ou *sensor manager*: é o componente responsável pela execução dos sensores, além de sua ativação e desativação. O agente *Portmonitor*, por exemplo, monitora um conjunto de portas e, com base na atividade destas, dispara a execução de sensores. Um arquivo de configuração contém informações sobre os sensores que devem ser ativados. Em geral o componente

sensor manager é executado em cada computador conectado ao sistema, e é responsável pela transmissão das informações coletadas pelos sensores para o componente *Gateway*;

- *Gateway*: é o componente onde os clientes (consumidores) fazem a assinatura das informações disponíveis; é também responsável pela intermediação entre sensores e clientes e pela execução da maior parte do processamento;
- *Directory service*: utilizado para a publicação da localização dos sensores existentes e dos *gateways* associados a estes sensores. Possibilita que os consumidores encontrem as informações desejadas e para isto utiliza o serviço de diretórios LDAP, por ser uma solução padrão que possibilita hierarquia e replicação de servidores;
- *Consumer* ou *Data Collector*: é o componente que obtém, combina e trata as informações de diversos *gateways*. Existem por exemplo, consumidores que obtêm as informações em tempo real e consumidores que armazenam informações em bases de dados, para visualização ou análises.

EDG Logging and Bookkeeping

O serviço LB (*Logging and Bookkeeping*) [88] é um componente do sistema *EDG Workload Management System* (WMS) [107], responsável pela obtenção e armazenamento de eventos que contêm informações sobre a execução de *jobs*. Este serviço recebe informações de outros componentes do sistema WMS, mantém o histórico sobre a execução de jobs, armazenando estes dados, que podem ser posteriormente processados para que sejam apresentadas informações de "alto-nível" sobre o estado dos *jobs*, como por exemplo: submetido, em execução, encerrado etc. Os eventos são recebidos e armazenados pelo componente LB que está localizado mais próximo, para evitar problemas de comunicação. Este componente armazena estas informações no disco local e, posteriormente, as envia ao componente servidor hierarquicamente superior. Este servidor é responsável pelo registro e manutenção de informações sobre cada *job*, e é associado estaticamente ao *job* no momento da sua submissão. O LB É um serviço distribuído, e pode ser usado em conjunto com a ferramenta R-GMA [65], para utilização dos mecanismos de notificação desta ferramenta. Desenvolvido originalmente no projeto *European DataGrid* (EDG) [108], continua sendo utilizado nos projetos que sucederam o EDG, como *LHC Computing Grid Project* (LCG) [109] e CrossGrid [110].

MDS - Monitoring and Discovery System

MDS (*Monitoring and Discovery System*) [14] [57] é o serviço de informação utilizado pelo *Globus Toolkit*; utiliza uma estrutura extensível e hierárquica para gerenciar informações, estáticas e dinâmicas, sobre o estado dos componentes da grade computacional. As informações são geradas pelos "provedores de informação" e recebidas, tratadas e armazenadas pelo MDS. Existem duas versões do MDS: o MDS2 construído sobre o LDAP, e o MDS3 que é baseado nos *Information Services* do Globus Toolkit 3. O MDS2 é baseado em dois protocolos definidos pelo GGF: GRIP (Grid Information Protocol), que permite interações para obtenção de informações (*query/response*) e de busca; e GRRP (Grid Resgistration Protocol), que mantém o registro de relacionamentos entre os componentes MDS. É composto por: sensores (provedores de informação), produtores (implementados como GRIS - Grid Resource Information Services) e intermediários (implementados como GIIS

- Grid Index Information Services). Os produtores obtêm as informações geradas pelos sensores, que podem ser um conjunto de *shell scripts* ou módulos carregados por meio de uma API, e as publicam utilizando o GRRP. Os intermediários permitem que seja construída uma estrutura hierárquica, onde cada nó filtra e agrega as informações recebidas. Os consumidores podem requisitar informações dos produtores ou dos intermediários, utilizando o protocolo GRIP. Além disto, o MDS oferece suporte ao controle de acesso, por meio do GSI (Grid Security Infraestructure). O MDS3 é baseado em *web-services*, seguindo a arquitetura OGSA (Open Grid Services Architecture) utilizada no Globus Toolkit 3. Nesta arquitetura cada serviço atende algumas convenções e implementa interfaces definidas: por meio da interface *portTypes* cada serviço disponibiliza o seu estado e seus atributos, que podem ser, opcionalmente, notificados por meio de eventos. *portTypes* é uma terminologia *Web Services Description Language* (WSDL) utilizada para descrever o serviço, a forma de acessá-lo, as operações ou métodos disponíveis. Os serviços MDS2 e MDS3 implementam, além do serviço de monitoramento, o serviço de diretório utilizado pelo Globus para descoberta de recursos.

NetLogger - Network Application Logger Toolkit

NetLogger [72] (*Network Application Logger Toolkit*) é uma ferramenta utilizada para análise de desempenho de ambientes complexos, como aplicações cliente/servidor e/ou *multi-thread*, executadas em sistemas distribuídos. É parte do projeto Distributed Monitoring Framework (DMF), assim como o JAMM e o pyGMA. A ferramenta NetLogger manipula informações previamente registradas em arquivos de *log* contendo dados sobre o desempenho dos sistemas. Inclui módulos para geração de eventos e para vizualização de dados em tempo-real, e consiste de quatro componentes:

- API e bibliotecas, utilizadas pelas aplicações para geração de eventos;
- sensores para obtenção de informações sobre os computadores e redes, normalmente implementados como *wrappers* de comandos de monitoramento de sistemas Linux;
- ferramentas para coleta e manipulação de registros de *logs*;
- interface gráfica, para visualização em tempo real ou de informações previamente armazenadas.

Para permitir que a análise de desempenho seja feita de forma mais precisa, é importante que seja coletado o maior número de informações disponíveis, que devem incluir dados sobre aplicações, sistemas operacionais e redes de comunicação, e que estas informações sejam padronizadas. Para isto o NetLogger utiliza o formato ULM (Universal Logger Message). Para a geração de eventos e, conseqüentemente, a obtenção de informações sobre a execução de aplicações, estas precisam ser instrumentadas, ou seja, é preciso incluir no código das aplicações a geração de *logs* que identifiquem as ocorrências a serem observadas. Os eventos sobre os sistemas e redes são gerados pelos sensores (*wrappers*). Os eventos podem ser gravados em arquivos locais, utilizando um *daemon* que executa esta função, ou transmitidos para um servidor por meio de uma porta específica. Em geral, o NetLogger é utilizado para tratar eventos e informações geradas por meio das ferramentas pyGMA ou JAMM, além de dados gerados pelas próprias aplicações e *wrappers*.

Ganglia

Ganglia [62] é um serviço de monitoramento distribuído e escalável, baseado em arquitetura hierárquica, para sistemas de computação de alto desempenho, como *clusters* e *grids*. É uma ferramenta

muito utilizada, disponível para diversas arquiteturas e sistemas operacionais. Utiliza protocolo baseado em *multicast listen/announce*, que permite a entrega de informações para diversos destinatários simultaneamente, de forma eficiente, onde as mensagens passam por um "caminho" uma única vez, e são duplicadas quando o "caminho" para os destinatários se bifurca: usa uma árvore de conexões ponto-a-ponto para representar os *clusters*. Usa o sistema RRDtool (Round Robin Database) [111] para armazenamento dos dados monitorados e também para geração de gráficos de visualização das informações coletadas.

Nagios

Nagios [58] é um sistema de monitoramento que executa *plug-ins* externos para obter informações sobre os computadores e serviços sob observação. Executa, em cada computador a ser monitorado, um progama com *plug-ins*, que permite que os clientes enviem mensagens de verificação de serviços por conta própria ao servidor onde é exectuado o Nagios. Permite também o envio de notificações quando identifica problemas.

MonALISA - MONitoring Agents using a Large Integrated Services Architecture

MonALISA [63] é um serviço de monitoramento que permite a distribuição de dados de forma escalável. Pode incorporar informações obtidas por outros sistemas, como o SNMP para obtenção de informações sobre a rede, o Ganglia, MRTG ou Hawkeye para informações sobre recursos do ambiente. Sua arquitetura é baseada em quatro camadas:

- a primeira camada implementa um serviço de registro distribuído, para identificação e registro de outros agentes e serviços;
- a segunda camada implementa a coleta de informações, por meio de agentes que obtêm, analisam e armazenam estas informações em tempo real;
- a terceira camada, serviços de *proxy*, implementa a comunicação confiável entre agentes, na obtenção de informações solicitadas pelos clientes;
- na quarta camada são implementados os clientes, consumidores das informações coletadas;

MapCenter

A forma de apresentação das informações, que permite a visualização das mesmas em um mapa geográfico, é o diferencial do sistema MapCenter [66]. As informações podem ser visualizadas em diferentes níveis de abstração, defindos por meio de um modelo autônomo e extensível. Todas as informações são armazenadas em *flat-files* que contêm o estado dos recursos sendo monitorados. Utiliza o GIS (Grid Information Service) para localizar os recursos da grade computacional e estabelece conexões TCP para obter informações sobre recursos da rede.

R-GMA - Relational Grid Monitoring Architecture

R-GMA [65] é uma implementação da arquitetura GMA com o armazenamento das informações em uma base de dados relacional. Implementa o modelo consumidor-produtor na coleta informações. Utiliza um subconjunto da SQL: os produtores publicam tuplas (linhas de base de dados) e os consumidores as obtêm utilizando comandos SQL select. O R-GMA é utilizado atualmente no projeto EGEE como catálogo de informações, ou registro, e não como ferramenta de monitoramento.

pyGMA - Grid Monitoring Architecture Web-Services Interfaces in Python

pyGMA [86] (*Grid Monitoring Architecture Web-Services Interfaces in Python*) é uma ferramenta desenvolvida pelo mesmo grupo do JAMM, como parte do projeto Distributed Monitoring Framework (DMF) [112]: ferramentas para monitoramento e análise de performance em ambientes distribuídos. O objetivo do pyGMA é permitir a manipulação de dados gerados por um grande número de eventos e prover mecanismos escaláveis e flexíveis para a publicação e assinatura destes eventos. Na verdade, pyGMA não é uma ferramenta de monitoramento, mas uma infra-estrutura que manipula a comunicação, usando o protocolo SOAP, entre os componentes propostos na arquitetura GMA: produtores, consumidores e serviço de diretório, ou registro. O uso de Python deve-se à facilidade de uso e ao suporte a um grande número de funções, como por exemplo smtp, ldap, XML, Web Services. Seus usuários são os desenvolvedores de ferramentas de monitoramento, que devem programar os procedimentos a serem executados quando ocorrerem assinaturas, buscas ou notificações de eventos.

MA-GMA - Mobile Agent-based Grid Monitor Architecture

MA-GMA [87] (*Mobile Agent-based Grid Monitor Architecture*) é uma ferramenta desenvolvida com base na tecnologia de agentes móveis com o objetivo de suportar a heterogeneidade dos ambientes de grade computacional e de permitir o desenvolvimento de procedimentos customizados nos computadores que compõem a grade. A ferramenta MA-GMA é composta por módulos coletores baseados na tecnologia de agentes móveis e módulos produtores com mecanismos de *cache*, e pode ser descrita como uma infraestrutura que possibilita a implementação de procedimentos customizados para monitoramento. Por meio de um dos paradigmas da tecnologia de agente móveis, que consiste em mover a lógica do processamento para perto dos dados de que necessita, um agente MA-GMA que tenha a função de produtor é capaz de realizar operações complexas sobre as informações coletadas antes de transferi-las e isto, em geral, reduz significativamente o tráfego de rede além de possibilitar o uso de algoritmos customizados, encapsulados em agentes, para filtro e análise das informações. É composta por serviços que provêem gerenciamento dos agentes e mecanismos para comunicação entre os agentes por meio do protocolo SOAP, por módulos para produção e tratamento de informações a serem monitoradas, e também por bibliotecas onde são mantidas as funções utilizadas no monitoramento. Compõem a MA-GMA:

- *Monitoring Service* que coleta as informações desejadas, por meio de sensores, e as envia ao agente estacionário que as solicitou;
- *Monitoring Task Pattern Center*, composto uma biblioteca onde são incluidas funções padrão de monitoramento e por uma segunda onde são mantidas as funções customizadas pelos usuários;

- *Mobile Agent Manager* que é responsável por todo o gerenciamento dos agentes, como por exemplo, migração, comunicação e segurança;
- *Mobile Agent Factory*, onde os agentes são gerados e enviados as componentes da grade;
- *SOAP Tunnel* que conecta todos os computadores da grade para que sejam transmitidas todas as informações, incluindo os próprios agentes, formatadas em mensagens do protocolo SOAP;
- *Agent Executing Environment* que é o ambiente onde os agentes são executados e que provê interfaces para comunicação entre agentes e computadores, além de serviços para criação, migração e encerramento dos agentes.

Apêndice B

Base de Dados

Aqui é apresentada a base de dados relacional utilizada para armazenamento das informações coletadas pela ferramenta de monitoramento JoiNMon.

Para que se obtenha a flexibilidade e o desempenho desejados desta base de dados é usada uma estrutura de dados normalizada, ou seja, estruturas de dados não redundantes que representam as entidades e os relacionamentos descritos pelos dados.

B.1 O modelo relacional

O modelo de dados relacional foi introduzido por Codd, em 1970, com a publicação do artigo "A Relational Model of Data for Large Shared Data Banks" [113]. O sucesso dos Sistemas Gerenciadores de Bancos de Dados (SGBD), que assumiram o lugar antes ocupado por bases de dados hierárquicas, deve-se à simplicidade e ao desempenho do modelo relacional. O modelo relacional é um modelo formal baseado em cálculos relacionais e na teoria dos conjuntos. Neste modelo os dados são organizados em tabelas, que são um conjunto de objetos do mesmo tipo. No modelo relacional:

- cada tabela, chamada de relação, tem um nome único;
- cada linha da tabela é conhecida como tupla;
- cada coluna da tabela é conhecida como atributo, e os valores de uma coluna da tabela são do mesmo tipo.

Nas consultas à base de dados pode ser feita a seleção de linhas ou de colunas de uma tabela, ou ainda pela combinação de informações dispersas em diferentes tabelas.

B.2 Tabelas utilizadas por JoiNMon

As tabelas criadas para o armazenamento das informações coletadas pela ferramenta de monitoramento JoiNMon foram projetadas com o objetivo de ajudar no desempenho de JoiNMon, tanto na gravação como na recuperação das informações de monitoramento. Estas tabelas e seu atributos são apresentados a seguir. As informações sobre o tipo e tamanho dos campos são apresentadas na Fig. B.1, na seção B.3.

B.2.1 Tabela *SERVER*

Nesta tabela são mantidas as informações sobre o componente *server* do JoiN bem como características do computador em que este componente é executado. Esta tabela é alimentada com informações obtidas pelo módulo *S_JoinMonitor* na ocorrência dos eventos JOIN_START e JOIN_SHUTDOWN.

Atributos da tabela *SERVER*

Identificação	Descrição
SERVERNAME	Chave primária. Nome da máquina onde é executado o componente *server*
SERVEROPSYS	Sistema operacional
SERVERJVMVER	Versão da máquina virtual Java
SERVERJOINVER	Versão do sistema JoiN
SERVERSTARTTIME	Horário de início de execução do sistema JoiN
SERVERCOORDN	Número de componentes *coordinator* conectados a este *server*
SERVERWORKERN	Número total de componentes *worker* conectados a este *server*
SERVERSTATUS	Estado do sistema JoiN (R=ativo D=inativo)
SERVERLASTUPD	Horário da última atualização das informações contidas nesta tabela

O atributo SERVERNAME contém o nome do computador onde é executado este componente. Este nome é obtido por meio da execução do comando Java
`(InetAddress.getLocalHost()).getCanonicalHostName()`
que retorna o nome canônico associado ao computador.

O atributo SERVERSTATUS indica se existe alguma aplicação em execução no sistema JoiN (SERVERSTATUS=R) ou não (SERVERSTATUS=D).

B.2.2 Tabela *COORDINATOR*

Nesta tabela são mantidas as informações sobre os componentes *coordinator* do JoiN bem como características dos computadores em que estes componentes são executados. Esta tabela é alimentada com informações obtidas pelos módulos *C_JoinMonitor* na ocorrência do evento NEW_COORD e *S_Communicator* no evento COORD_DOWN.

Atributos da tabela *COORDINATOR*

Identificação	Descrição
COORDID	Chave primária. Identificação única atribuída ao componente *coordinator* na sua ativação
COORDNAME	Nome da máquina onde é executado este *coordinator*
COORDJOOKIE	JoiNCookie - identificação única associada pelo JoiN ao colaborador que disponibiliza o computador onde este *coordinator* é executado
COORDSTARTTIME	Horário de início de execução deste componente
COORDOPSYS	Sistema operacional
COORDJVMVER	Versão da máquina virtual Java
COORDJOINVER	Versão do sistema JoiN
COORDSTATUS	Estado do componente (R=ativo D=inativo)
COORDWORKERN	Número de componentes *worker* conectados a este *coordinator*
COORDLASTUPD	Horário da última atualização das informações contidas nesta tabela

O atributo COORDNAME contém o nome do computador onde é executado este componente. Este nome é obtido por meio da execução do comando Java
`(InetAddress.getLocalHost()).getCanonicalHostName()`
que retorna o nome canônico associado ao computador.

O atributo COORDSTATUS indica se existe alguma aplicação em execução no grupo coordenado pelo componente *coordinator* (COORDSTATUS=R) ou não (COORDSTATUS=D).

B.2.3 Tabela *WORKER*

Nesta tabela são mantidas as informações sobre os componentes *worker* do JoiN bem como características dos computadores em que estes componentes são executados. Esta tabela é alimentada com informações obtidas pelos módulos *W_JoinMonitor* na ocorrência do evento NEW_WORKER, *TaskSchedulerer* no evento WORKER_PERF e *C_Communicator* no evento WORKER_DOWN.

Atributos da tabela *WORKER*

Identificação	Descrição
WORKID	Chave primária. Identificação única atribuída ao componente *worker* na sua ativação
WORKJOOKIE	JoiNCookie - identificação única associada pelo JoiN ao colaborador que disponibiliza o computador onde este *worker* é executado
WORKNAME	Nome da máquina onde é executado este *worker*
WORKSTARTTIME	Horário de início de execução deste componente
WORKOPSYS	Sistema operacional
WORKJVMVER	Versão da máquina virtual Java
WORKPERFORM	Índice de desempenho do componente
WORKSTATUS	Estado do componente (R=ativo D=inativo)
WORKCOORDID	Identificação componente *coordinator* a que este *worker* está conectado
WORKLASTUPD	Horário da última atualização das informações contidas nesta tabela
WORKJOINVER	Versão do sistema JoiN

O atributo WORKERNAME contém o nome do computador onde é executado este componente. Este nome é obtido por meio da execução do comando Java
`(InetAddress.getLocalHost()).getCanonicalHostName()`
que retorna o nome canônico associado ao computador.

O atributo WORKSTATUS indica se o componente *worker* está processando alguma aplicação (WORKSTATUS=R) ou não (WORKSTATUS=D).

B.2.4 Tabela *COLABORATOR*

Nesta tabela são mantidas as informações sobre os colaboradores do sistema JoiN. Cada novo colaborador é registrado nesta tabela, quando efetua seu cadastramento no sistema JoiN, por meio da ferramenta de segurança, autenticação e controle de usuários e colaboradores.

Esta tabela é alimentada com informações para contabilização dos recursos utilizados pelo sistema JoiN nos computadores disponibilizados por estes colaboradores. Estas informações são obtidas pelos módulos *W_JoiNMonitor* na ocorrência do evento NEW_WORKER, *C_Communicator* no evento WORKER_DOWN e *W_ApplicationManager* no evento TSK_END.

O atributo COLSTATUS indica se algum dos computadores disponibilizados por este colaborador está conectado ao sistema JoiN (COLSTATUS=R) ou não (COLSTATUS=D).

B.2.5 Tabela *USER*

Nesta tabela são mantidas as informações sobre os usuários do JoiN. Cada novo usuário é registrado nesta tabela, quando efetua seu cadastramento no sistema JoiN, por meio da ferramenta de segurança, autenticação e controle de usuários e colaboradores. Esta tabela é alimentada com informações obtidas pelo módulo *S_ApplicationManager* na ocorrência do evento APP_SUB. A ferramenta

Atributos da tabela *COLABORATOR*

Identificação	Descrição
COLJOOKIE	Chave primária. JoiNCookie - identificação única associada pelo JoiN ao colaborador
COLACTNUMCONN	Número de conexões ativas deste colaborador ao JoiN
COLFIRSTCONNSTART	Data e hora da primeira conexão do colaborador ao JoiN
COLACTCONNSTART	Data e hora de iníco da última conexão do colaborador ao JoiN
COLLASTCONNEND	Data e hora da última desconexão do colaborador do JoiN
COLTOTALNUMCONN	Número de conexões já realizadas ao JoiN pelo colaborador
COLTOTALCONNTIME	Tempo total que o colaborador já esteve conectado ao JoiN
COLTOTALPROCTIME	Tempo total de processamento que o JoiN já utilizou deste colaborador
COLTOTALTSKEXEC	Número de tarefas já executadas por este colaborador
COLSECSTAT	Estado deste colaborador com relação à ferramenta de segurança
COLSTATUS	Estado deste colaborador (R=ativo D=inativo)

de monitoramento atualiza a informação sobre a data e horário da última submissão de uma aplicação para execução no sistema JoiN pelo usuário (USERLASTCONN). Os demais atributos são descritos e manipulados pela ferramenta de segurança, em desenvolvimento.

Atributos da tabela *USER*

Identificação	Descrição
USERID	Chave primária - identificação única associada pelo JoiN ao usuário
USERNAME	Nome do usuário
USERPASSWD	Senha cifrada do usuário para autenticação no JoiN
USEREMAIL	e-mail cadastrado
USERLASTCONN	Data e horário da última submissão de uma aplicação para execução no sistema JoiN

B.2.6 Tabela *USERAPP*

Nesta tabela são mantidas as informações sobre as aplicações executadas do sistema JoiN pelos seus usuários. Esta tabela é alimentada com informações para contabilização dos recursos utilizados pelas aplicações executadas. Estas informações são obtidas pelos módulos *S_ApplicationManager* na ocorrência do evento APP_SUB, *ExecutionEngine* no evento APP_START, *C_ApplicationManager* no evento APP_END, *S_ApplicationManager* no evento APP_KILL, *W_ApplicationManager* no evento TSK_START e *W_ApplicationManager* no evento TSK_END.

Atributos da tabela *USERAPP*

Identificação	Descrição
UAPPUSERID	Chave primária - Identificação do usuário que submeteu a aplicação
UAPPNAME	Chave primária - Nome da aplicação
UAPPSUBTIME	Chave primária - Data e horário de submissão
UAPPNUMB	Identificação única associada pelo JoiN às aplicações ativas
UAPPSTARTIME	Data e horário de início da execução
UAPPPROCTIME	Tempo de processamento
UAPPNUMBATCH	Número de lotes que compõem a aplicação
UAPPNUMTASKS	Número total de tarefas que compõem a aplicação
UAPPNUMWBEGIN	Número de componentes *worker* ativos no início da execução da aplicação
UAPPNUMWFINISH	Número de componentes *worker* ativos no término da execução da aplicação
UASTATUS	estado da aplicação (W=espera R=execução C=completada K=cancelada)

O atributo UASTATUS indica o estado da aplicação: se foi submetida mas ainda não iniciou a execução (UASTATUS=W), se está sendo executada (UASTATUS=R), se terminou a execução (UASTATUS=C) ou se sua execução foi cancelada pelo usuário (UASTATUS=K).

B.2.7 Tabela *APPRUN*

Nesta tabela são mantidas as informações sobre as aplicações ativas (em execução, em espera ou recentemente completadas) no sistema JoiN. Esta tabela é alimentada com informações obtidas pelos módulos *S_ApplicationManager* na ocorrência do evento APP_SUB, *ExecutionEngine* no evento APP_START, *C_ApplicationManager* no evento APP_END e *S_ApplicationManager* no evento APP_KILL.

Atributos da tabela *APPRUN*

Identificação	Descrição
APPNUMB	Chave primária - Identificação única associada pelo JoiN às aplicações ativas
APPUSERID	Identificação do usuário que submeteu a aplicação
APPNAME	Nome da aplicação
APPSUBTIME	Data e horário de submissão

B.2.8 Tabela *APPBATCH*

Nesta tabela são mantidas as informações sobre os lotes das aplicações em execução no sistema JoiN. Esta tabela é alimentada com informações obtidas pelos módulos *W_ApplicationManager* na ocorrência do evento TSK_START e *W_ApplicationManager* no evento TSK_END.

Atributos da tabela *APPBATCH*

Identificação	Descrição
APPBNUMB	Chave primária - Identificação única associada pelo JoiN às aplicações ativas
APPBBATCHNUMB	Chave primária - Número do lote
APPBSTARTTIME	Data e horário de início da execução do lote
APPBENDTIME	Data e horário de término da execução do lote
APPBNUMTSK	Número de tarefas que compõem o lote
APPBSTATUS	Estado do lote (W=espera R=execução C=completado)

O atributo APPBSTATUS indica o estado do lote de uma aplicação: se ainda não iniciou a execução (APPBSTATUS=W), se está sendo executado (APPBSTATUS=R) ou se terminou a execução (APPBSTATUS=C).

B.2.9 Tabela *APPTASK*

Nesta tabela são mantidas as informações sobre as tarefas que compõem os lotes das aplicações em execução no sistema JoiN. Esta tabela é alimentada com informações obtidas pelos módulos *W_ApplicationManager* na ocorrência do evento TSK_START e *W_ApplicationManager* no evento TSK_END.

Atributos da tabela *APPTASK*

Identificação	Descrição
APPTNUMB	Chave Primária - Identificação única associada pelo JoiN às aplicações ativas
APPTBATCHNUMB	Chave Primária - Número do lote
APPTTASKNUMB	Chave Primária - Número da tarefa
APPTSTARTTIME	Data e horário de início da execução da tarefa
APPTENDTIME	Data e horário de término da execução da tarefa
APPTSTATUS	Estado da tarefa (W=espera R=execução C=completada)
APPTEXECHOSTS	Relação dos componentes *worker* onde esta tarefa está sendo executada (indica se há réplicas)

O atributo APPTSTATUS indica o estado de uma tarefa da aplicação: se ainda não iniciou a execução (APPTSTATUS=W), se está sendo executada (APPTSTATUS=R) ou se terminou a execução (APPTSTATUS=C).

B.2.10 Tabela *JOOKIE*

Nesta tabela são mantidos os JoiNCookies, que contêm a identificação única associada pelo JoiN ao colaborador que disponibiliza computadores para uso no sistema. A ferramenta de monitoramento não acessa esta tabela, cujos atributos são descritos e manipulados pela ferramenta de segurança.

Atributos da tabela *JOOKIE*

Identificação	Descrição
JOOKIESN	Chave primária - Número serial atribuído a este Jookie
JOOKIEMAC	Senha cifrada para autenticação do Jookie
JOOKIEEMAIL	E-mail do colaborador
JOOKIEVER	Versão do Jookie
JOOKIESECLVL	Nível de segurança
JOOKIELASTUPD	Data e horário da última atualização desta tupla

B.2.11 Tabela *JHISTORY*

Nesta tabela são mantidas as informações históricas, sobre estados passados, relativas à execução de aplicações já encerradas. Esta tabela é alimentada com informações obtidas pelo módulo *C_ApplicationManager* na ocorrência do evento APP_END.

Atributos da tabela *JHISTORY*

Identificação	Descrição
JHUSERID	Chave primária - Identificação do usuário
JHAPPNAME	Chave primária - Identificação da aplicação
JHSUBTIME	Chave primária - Data e horário de submissão da aplicação
JHCOLDATA	Objeto contendo dados coletados na execução da aplicação

O atributo JHCOLDATA armazena um objeto onde estão contidas as seguintes informações sobre a aplicação: data e horário de início da execução, tempo de processamento, número de lotes que a compõem, número total de tarefas processadas, número de componentes *worker* ativos no início e no final da execução da aplicação.

B.3 Representação gráfica tabelas e relacionamentos

A modelagem da base de dados é representada na Fig. B.1, onde podem ser vistas as tabelas e relacionamentos definidos para armazenar as informações monitoradas.

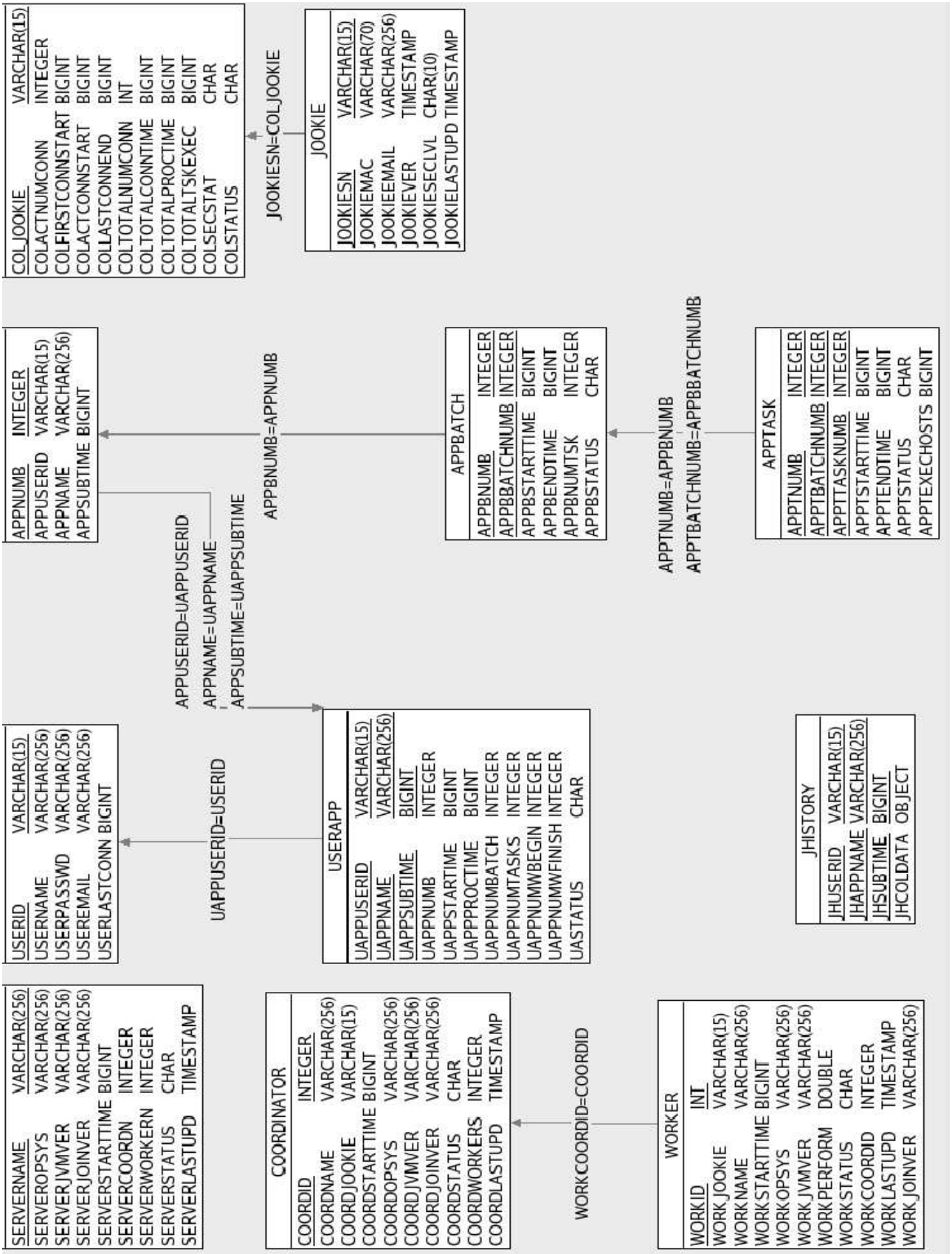

Fig. B.1: Esquema de tabelas e relacionamentos

B.4 Código para a criação da base de dados

No Código B.1 são apresentados os comandos DDL (Data Definition Language) utilizados para a criação das tabelas na base de dados. Além de permitirem a criação das tabelas eles servem também de complemento à documentação das mesmas.

Código B.1: Comandos DDL para criação das tabelas

```
\begin{small}

CREATE TABLE SERVER (
        SERVERNAME VARCHAR(256) NOT NULL
      , SERVEROPSYS VARCHAR(256)
      , SERVERJVMVER VARCHAR(256)
      , SERVERJOINVER VARCHAR(256)
      , SERVERSTARTTIME BIGINT
      , SERVERCOORDN INTEGER
      , SERVERWORKERN INTEGER
      , SERVERSTATUS CHAR
      , SERVERLASTUPD TIMESTAMP
      , PRIMARY KEY (SERVERNAME)
);

CREATE TABLE COORDINATOR (
        COORDID INTEGER NOT NULL
      , COORDNAME VARCHAR(256)
      , COORDJOOKIE VARCHAR(15)
      , COORDSTARTTIME BIGINT
      , COORDOPSYS VARCHAR(256)
      , COORDJVMVER VARCHAR(256)
      , COORDJOINVER VARCHAR(256)
      , COORDSTATUS CHAR
      , COORDWORKERS INTEGER
      , COORDLASTUPD TIMESTAMP
      , PRIMARY KEY (COORDID)
);

CREATE TABLE WORKER (
        WORKID INT NOT NULL
      , WORKJOOKIE VARCHAR(15)
      , WORKNAME VARCHAR(256)
      , WORKSTARTTIME BIGINT
      , WORKOPSYS VARCHAR(256)
      , WORKJVMVER VARCHAR(256)
      , WORKPERFORM DOUBLE
      , WORKSTATUS CHAR
      , WORKCOORDID INTEGER
      , WORKLASTUPD TIMESTAMP
      , WORKJOINVER VARCHAR(256)
      , PRIMARY KEY (WORKID)
      , CONSTRAINT FK_WORKER_1 FOREIGN KEY (WORKCOORDID)
                   REFERENCES COORDINATOR (COORDID) ON DELETE CASCADE
);
```

```
CREATE INDEX SYS_IDX_5 ON WORKER (WORKCOORDID);

CREATE TABLE COLABORATOR (
        COLJOOKIE VARCHAR(15) NOT NULL
      , COLACTNUMCONN INTEGER
      , COLFIRSTCONNSTART BIGINT
      , COLACTCONNSTART BIGINT
      , COLLASTCONNEND BIGINT
      , COLTOTALNUMCONN INT
      , COLTOTALCONNTIME BIGINT
      , COLTOTALPROCTIME BIGINT
      , COLTOTALTSKEXEC BIGINT
      , COLSECSTAT CHAR
      , COLSTATUS CHAR
      , PRIMARY KEY (COLJOOKIE)
);

CREATE TABLE USER (
        USERID VARCHAR(15) NOT NULL
      , USERNAME VARCHAR(256)
      , USERPASSWD VARCHAR(256)
      , USEREMAIL VARCHAR(256)
      , USERLASTCONN BIGINT
      , PRIMARY KEY (USERID)
);

CREATE TABLE USERAPP (
        UAPPUSERID VARCHAR(15) NOT NULL
      , UAPPNAME VARCHAR(256) NOT NULL
      , UAPPSUBTIME BIGINT NOT NULL
      , UAPPNUMB INTEGER
      , UAPPSTARTIME BIGINT DEFAULT '0'
      , UAPPPROCTIME BIGINT DEFAULT '0'
      , UAPPNUMBATCH INTEGER
      , UAPPNUMTASKS INTEGER
      , UAPPNUMWBEGIN INTEGER
      , UAPPNUMWFINISH INTEGER
      , UASTATUS CHAR
      , PRIMARY KEY (UAPPUSERID, UAPPNAME, UAPPSUBTIME)
      , CONSTRAINT FK_USERAPP_1 FOREIGN KEY (UAPPUSERID)
                  REFERENCES USER (USERID) ON DELETE NO ACTION ON UPDATE NO
                      ACTION
);
CREATE UNIQUE INDEX SYS_PK_USERAPP
                        ON USERAPP (UAPPUSERID, UAPPNAME);

CREATE TABLE APPRUN (
        APPNUMB INTEGER NOT NULL
      , APPUSERID VARCHAR(15)
      , APPNAME VARCHAR(256)
      , APPSUBTIME BIGINT
      , PRIMARY KEY (APPNUMB)
      , CONSTRAINT FK_APPRUN_1
```

```
                    FOREIGN KEY (APPUSERID, APPNAME, APPSUBTIME)
                    REFERENCES USERAPP (UAPPUSERID, UAPPNAME, UAPPSUBTIME)
);

CREATE TABLE APPBATCH (
        APPBNUMB INTEGER NOT NULL
      , APPBBATCHNUMB INTEGER NOT NULL
      , APPBSTARTTIME BIGINT
      , APPBENDTIME BIGINT
      , APPBNUMTSK INTEGER
      , APPBSTATUS CHAR
      , PRIMARY KEY (APPBNUMB, APPBBATCHNUMB)
      , CONSTRAINT FK_APPBATCH_1 FOREIGN KEY (APPBNUMB)
                    REFERENCES APPRUN (APPNUMB) ON DELETE CASCADE
);

CREATE TABLE APPTASK (
        APPTNUMB INTEGER NOT NULL
      , APPTBATCHNUMB INTEGER NOT NULL
      , APPTTASKNUMB INTEGER NOT NULL
      , APPTSTARTTIME BIGINT
      , APPTENDTIME BIGINT
      , APPTSTATUS CHAR
      , APPTEXECHOSTS BIGINT
      , PRIMARY KEY (APPTNUMB, APPTBATCHNUMB, APPTTASKNUMB)
      , CONSTRAINT FK_APPTASK_1 FOREIGN KEY (APPTNUMB, APPTBATCHNUMB)
                    REFERENCES APPBATCH (APPBNUMB, APPBBATCHNUMB) ON DELETE
                        CASCADE
);

CREATE TABLE JOOKIE (
        JOOKIESN VARCHAR(15) NOT NULL
      , JOOKIEMAC VARCHAR(70)
      , JOOKIEEMAIL VARCHAR(256)
      , JOOKIEVER TIMESTAMP
      , JOOKIESECLVL CHAR(10)
      , JOOKIELASTUPD TIMESTAMP
      , PRIMARY KEY (JOOKIESN)
      , CONSTRAINT FK_JOOKIE_1 FOREIGN KEY (JOOKIESN)
                    REFERENCES COLABORATOR (COLJOOKIE) ON DELETE NO ACTION ON
                        UPDATE NO ACTION
);

CREATE TABLE JHISTORY (
        JHUSERID VARCHAR(15) NOT NULL
      , JHAPPNAME VARCHAR(256) NOT NULL
      , JHSUBTIME BIGINT NOT NULL
      , JHCOLDATA OBJECT
      , PRIMARY KEY (JHUSERID, JHAPPNAME, JHSUBTIME)
);

\end{small}
```

Apêndice C

Interface gráfica de monitoramento

A interface gráfica *JoiNMonitor*, desenvolvida em JSP, acessa as informações na base de dados mantida pelo serviço *JoiNMonitor* e as exibe de forma organizada.

O *layout* das páginas foi projetado para facilitar a navegação e a obtenção de informações. As páginas são divididas em 4 seções:

1. a parte superior é composta por botões, por meio dos quais é possível navegar entre as telas que contêm informações sobre os componentes do sistema (*server*, *coordinators* e *workers*), sobre as aplicações, colaboradores e usuários;
2. logo abaixo destes botões de navegação é apresentada a identificação do computador em que o componente *JoiN Server* está sendo executado;
3. a seguir encontra-se uma área para seleção e filtro das informações a serem exibidas;
4. na quarta seção encontra-se a área onde são apresentadas as informações selecionadas. Algumas das informações desta área são *links* para outras páginas, que trazem informações mais detalhadas.

Ao acessar a página inicial, por meio do endereço
`http://<nome ou endereço do servidor>/JoiNMonitor`
é exibida a página contendo informações sobre o componente *server*, a partir da qual é possível navegar para obter as informações desejadas. A seguir são apresentadas as informações exibidas em cada uma das páginas, exemplificadas pela exibição de modelos das mesmas.

Server - Página inicial

Na página inicial (Fig. C.1) são exibidas as informações sobre o componente *server*:

- data e horário da última atualização de informações do *server*;
- identificação do computador onde o componente está sendo processado;
- versão do sistema JoiN neste computador;
- sistema operacional utilizado;
- versão da máquina virtual Java;

Fig. C.1: JoiNMonitor - informações sobre o componente *server*

- data e horário da ativação do sistema JoiN neste *server*;
- número de *coordinators* conectados ao sistema;
- número de *workers* conectados ao sistema;
- número de aplicações em processamento;
- número de aplicações em espera;
- número de aplicações já processadas.

Coordinators

Ao selecionar o botão *Coordinators* na parte superior de qualquer página em exibição, o usuário será direcionado para uma página onde são listados os componentes *coordinator* e também um resumo das informações de cada um deles, como na Fig. C.2:

- *ID* - número serial que identifica o componente *coordinator*;
- identificação do computador onde o componente está sendo processado;
- estado do componente, que pode estar ativo (R) ou inativo (D);
- data e horário da ultima conexão do *coordinator* ao *server*;
- sistema operacional utilizado neste computador;
- versão da máquina virtual Java;
- versão do sistema JoiN;

- número de *workers* conectados a este *coordinator*;
- data e horário da última atualização de informações deste *coordinator*.

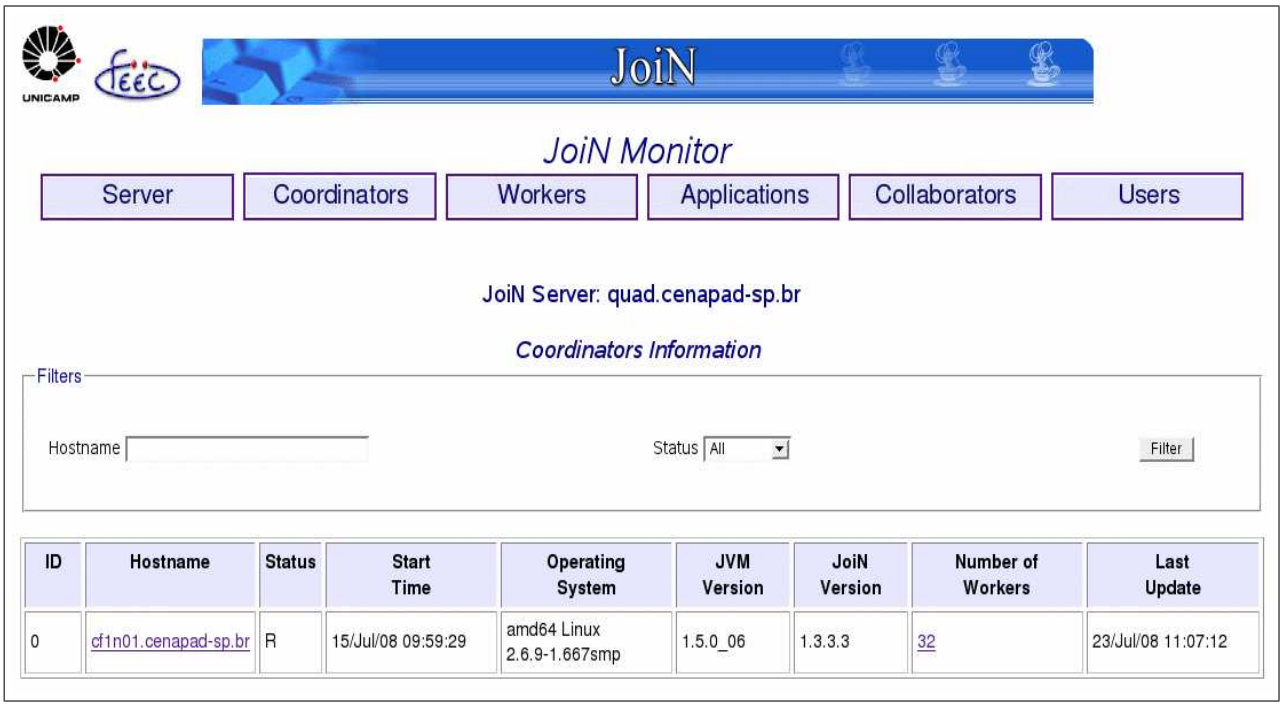

Fig. C.2: JoiNMonitor - informações sobre os componentes *coordinator*

A partir desta página pode-se obter informações detalhadas sobre cada um dos componentes *coordinator* do sistema, por meio dos *links* existentes na coluna *Hostname* de cada componente apresentado. A página com estas informações detalhadas é exemplificada pela Fig. C.3 e contém, além das informações sobre o *coordinator*, dados sobre as tarefas processadas neste componente, contendo:

- identificação da aplicação, do lote e da tarefa (*Application ID, Batch* e *Task*);
- data e horário do início da execução desta tarefa;
- data e horário do término da execução desta tarefa (se a tarefa ainda estiver em execução é exibido o caracter "-").

Workers

Ao selecionar o botão *Workers* na parte superior de qualquer página em exibição, ou então ao selecionar o *link* existente nas colunas *Number of Workers* das páginas já apresentadas, o usuário será direcionado para uma página onde são listados os componentes *workers* e também um resumo das informações de cada um deles, como na Fig. C.4:

- *ID* - número serial que identifica o componente *worker*;
- identificação do computador onde o componente está sendo processado;

Fig. C.3: JoiNMonitor - informações detalhadas sobre um componente *coordinator*

- ID do componente *coordinator* a que este *worker* está conectado;
- estado do componente, que pode estar ativo (R) ou inativo (D);
- data e horário da ultima conexão deste *worker* ao *server*;
- sistema operacional utilizado neste computador;
- versão da máquina virtual Java;
- versão do sistema JoiN;
- índice de desempenho atribuído a este componente *worker* (*Performance*);
- data e horário da última atualização de informações deste *worker*.

A partir desta página pode-se obter informações detalhadas sobre cada um dos componentes *worker* do sistema, por meio dos *links* existentes na coluna *Hostname* de cada componente apresentado. A página com estas informações detalhadas é exemplificada pela Fig. C.5 e contém, além das informações sobre o componente *worker*, dados sobre as tarefas processadas neste componente, contendo:

- identificação da aplicação; do lote e da tarefa (*Application ID, Batch* e *Task*);
- data e horário do início da execução desta tarefa;
- data e horário do término da execução desta tarefa (se a tarefa ainda estiver em execução é exibido o caracter "-").

Fig. C.4: JoiNMonitor - informações sobre os componentes *worker*

Aplicações

Ao selecionar o botão *Applications* na parte superior de qualquer página em exibição, ou então ao selecionar o *link* existente nas colunas *Application ID* das páginas já apresentadas, o usuário será direcionado para uma página onde são listadas as aplicações e também um resumo das informações de cada uma delas, como na Fig. C.6:

- identificação do usuário que submeteu a aplicação para execução;
- nome da aplicação;
- estado da aplicação que pode ser: aguardando para execução (W); em execução (R); execução completada (C) ou cancelada (K);
- data e horário de submissão da aplicação para execução;
- data e horário de início da execução (se estado diferente de W) (se o estado da aplicação for W é exibido o caracter “-”);
- tempo de processamento (se estado C) (se o estado da aplicação for diferente de C é exibido o caracter “-”);
- número de lotes que compõem a aplicação;
- número total de tarefas que compõem a aplicação (de todos os lotes);

Fig. C.5: JoiNMonitor - informações detalhadas sobre um componente *worker*

- número de componentes *worker* ativos no início da execução da aplicação;
- número de componentes *worker* ativos no término da execução da aplicação (se a aplicação ainda estiver em execução é exibido o caracter "-").

A partir desta página pode-se obter informações detalhadas sobre cada uma das aplicações em execução no sistema, por meio dos *links* existentes na coluna *Application ID* de cada aplicação apresentada. A página exemplificada pela Fig. C.8 contém informações sobre uma aplicação já encerrada. A página exemplificada pela Fig. C.7 apresenta informações detalhadas sobre uma aplicação em execução e contém, além das informações sobre a aplicação, dados sobre os lotes e as tarefas que compõem esta aplicação:

- identificação do lote e da tarefa;
- data e horário do início da execução da tarefa (se não foi iniciada a execução da tarefa é exibido o caracter "-");

Status	Name	User	Submit Time	Start Time	Process Time
C	Matrix Multiplication 100/4000x1000	ANONYMOUS	15/Jul/08 10:36:38	15/Jul/08 10:36:52	0d +0h:0m:54s:818ms
C	Matrix Multiplication 100/4000x1000	ANONYMOUS	15/Jul/08 10:47:04	15/Jul/08 10:47:34	0d +0h:0m:51s:912ms
C	Matrix Multiplication 100/4000x1000	ANONYMOUS	15/Jul/08 10:54:30	15/Jul/08 10:55:03	0d +0h:0m:55s:973ms
C	Matrix Multiplication 100/4000x1000	ANONYMOUS	15/Jul/08 11:44:06	15/Jul/08 11:45:06	0d +0h:1m:4s:215ms
C	Matrix Multiplication 100/4000x1000	ANONYMOUS	15/Jul/08 15:19:31	15/Jul/08 15:20:01	0d +0h:1m:9s:953ms
C	Matrix Multiplication 200/4000x1000	ANONYMOUS	15/Jul/08 11:01:31	15/Jul/08 11:01:33	0d +0h:0m:56s:445ms
C	Matrix Multiplication 200/4000x1000	ANONYMOUS	15/Jul/08 11:07:58	15/Jul/08 11:08:01	0d +0h:0m:46s:335ms
C	Matrix Multiplication 200/4000x1000	ANONYMOUS	15/Jul/08 11:50:57	15/Jul/08 11:51:48	0d +0h:0m:53s:44ms
C	Matrix Multiplication 200/4000x1000	ANONYMOUS	15/Jul/08 15:25:32	15/Jul/08 15:25:36	0d +0h:0m:50s:861ms
C	Matrix Multiplication 400/4000x1000	ANONYMOUS	15/Jul/08 11:16:12	15/Jul/08 11:16:13	0d +0h:1m:18s:577ms

Fig. C.6: JoiNMonitor - informações sobre as aplicações

- data e horário do término da execução da tarefa (se não foi encerrada a execução da tarefa é exibido o caracter “-”);
- estado da tarefa, que pode estar aguardando a execução (W), em processamento (R) ou completada (C);
- lista dos componentes *worker* onde esta tarefa está sendo processada e, no caso de tarefas já encerradas, identificação do componente *worker* que completou sua execução (se não foi iniciada a execução da tarefa é exibido o caracter “-”).

Usuários

Ao selecionar o botão *Users* na parte superior de qualquer página em exibição, ou então ao selecionar o *link* existente nas colunas *User* das páginas já apresentadas, o usuário será direcionado para uma página onde são listadas informações sobre os usuários do sistema, como na Fig. C.9:

- identificação do usuário;
- nome do usuário;
- e-mail cadastrado;

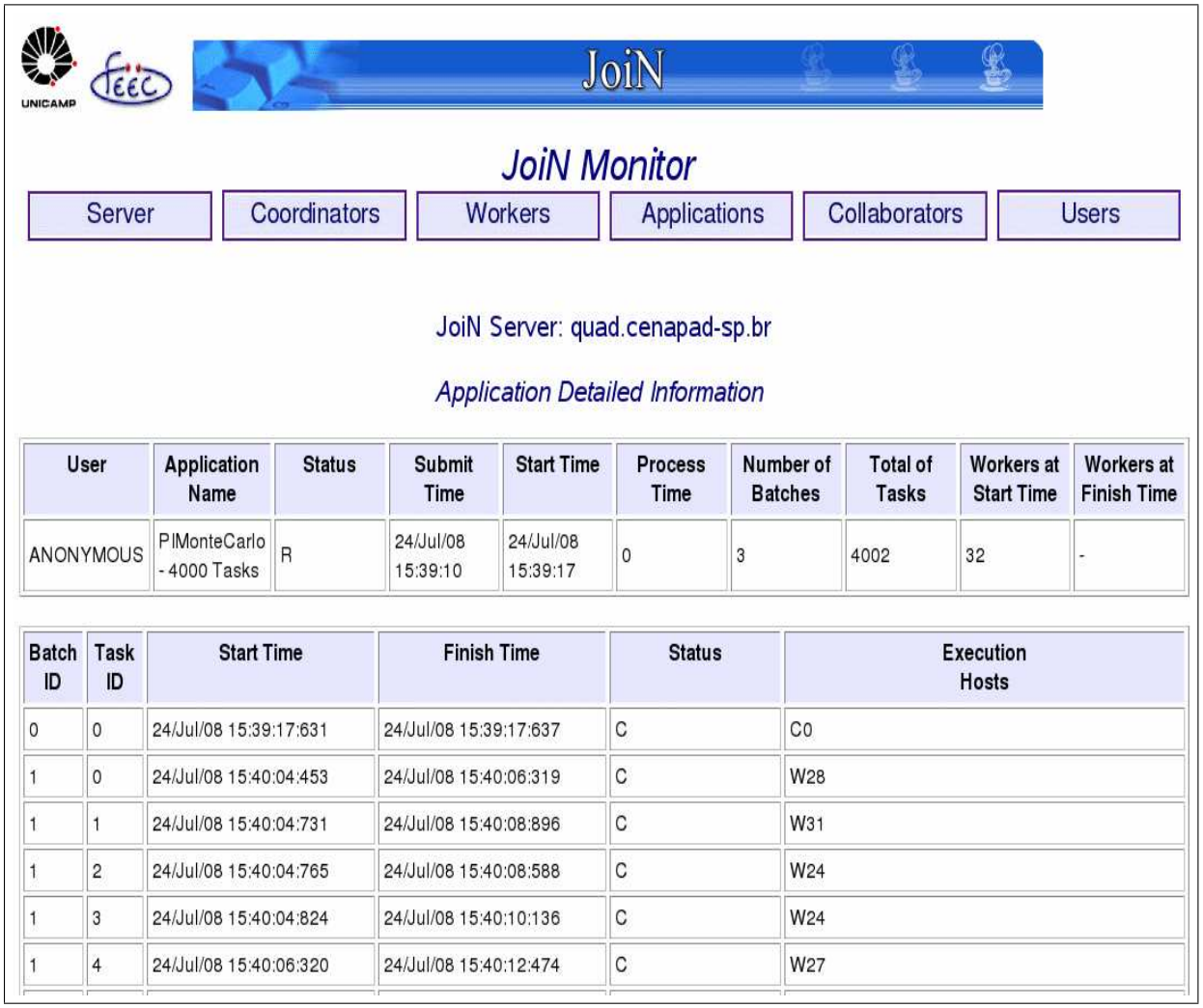

Fig. C.7: JoiNMonitor - informações detalhadas sobre uma aplicação em execução

- data e horário da última submissão de aplicação deste usuário no sistema JoiN (se o usuário nunca submeteu aplicações é exibido o caracter "-").
- número de aplicações em execução;
- número de aplicações aguardando execução;
- número de aplicações já processadas.

Colaboradores

Ao selecionar o botão *Collaborators* na parte superior de qualquer página em exibição, o usuário será direcionado para uma página onde são listadas informações sobre os colaboradores do sistema JoiN, como na Fig. C.10:

- identificação do colaborador;

Fig. C.8: JoiNMonitor - informações sobre uma aplicação encerrada

- estado do colaborador no sistema de segurança, que pode sinalizar a confiança que o sistema JoiN tem neste colaborador (funcionalidade a ser implementada pela ferramenta de segurança, ainda não disponível nesta versão do sistema JoiN e, por esta razão é exibido o caracter "-").
- data e horário da primeira conexão do colaborador ao sistema;
- tempo total que o colaborador já esteve conectado ao sistema, contribuindo para a execução de aplicações;
- número de conexões já realizadas;
- tempo total de processamento utilizado para a execução de aplicações no computador deste colaborador;
- número total de tarefas já processadas por este colaborador;
- número de conexões ativas;
- data e horário da primeira conexão ativa;
- data e horário de encerramento da última conexão.

Id	Name	Email	Last Connection	Applications Running	Applications Waiting	Applications Completed
ANONYMOUS	USER FOR ANONYMOUS ACCESS	user@dca.fee.unicamp.br	15/Jul/08 15:48:55	0	0	3

Fig. C.9: JoiNMonitor - informações sobre usuários

Jookie	Security Status	Status	First Connection	Total Connection Time	Total of Connections	Total Processing Time	Total Tasks Processed	Active Connections	Start Active Connection	End of Last Connection
NOTDEF	A	A	24/Jun/08 01:51:15	24d +14h:47m:23s:367ms	21945673	2d +5h:16m:52s:865ms	732885	32	15/Jul/08 11:40:42	-

Fig. C.10: JoiNMonitor - informações sobre colaboradores

Apêndice D

Dados obtidos nos experimentos

Neste apêndice são apresentados na íntegra os dados obtidos nos nos experimentos realizados para análise da ferramenta de monitoramento.

D.1 Dados para estimativa de R_{cc} das aplicações utilizadas

Na seção 4.3.6 do Capítulo 4 apresentamos a classificação das aplicações conforme a razão entre o tempo consumido em computação e o tempo consumido para a comunicação necessária para sua execução (R_{cc} = *tempo_de_computação / tempo_de_comunicação*).

Para estimar o valor de R_{cc} das aplicações utilizadas para avaliação da ferramenta JoiNMon foram realizados experimentos específicos, utilizando os mesmos computadores e configurações apresentados na seção4.3.6. Os componentes *server* e *coordinator* foram ativados em um computador, um componente *worker* foi ativado em um dos nós do cluster, e cada uma das aplicações foi executada com três tarefas.

O gráfico da Fig. D.1 apresenta os pacotes transmitidos durante os primeiros milissegundos da execução da aplicação para cálculo do número π.

Os gráficos da Fig. D.2 apresentam os pacotes transmitidos durante os intervalos, em milissegundos, em que se observou comunicação durante a execução da aplicação para multiplicação de matrizes.

Na Tab. D.1 são exibidos os dados obtidos nos experimentos realizados para estimativa de R_{cc} das aplicações para cálculo de π pelo método de Monte Carlo e para multiplicação de matrizes.

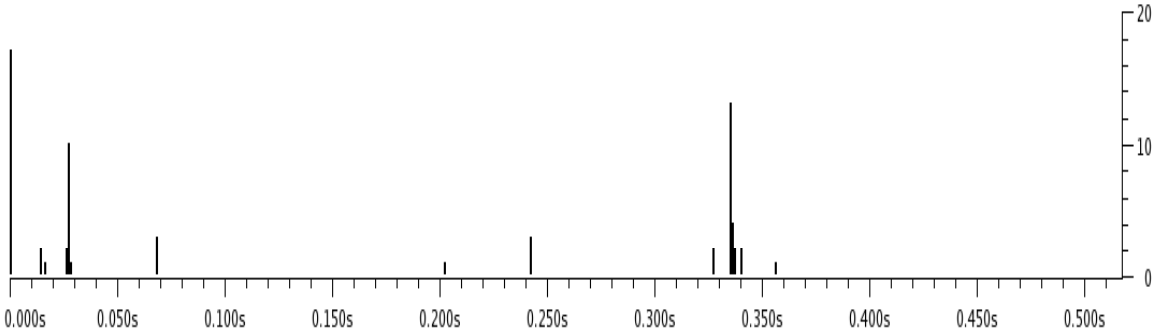

Fig. D.1: Cálculo de π - Pacotes transmitidos X Tempo (ms)

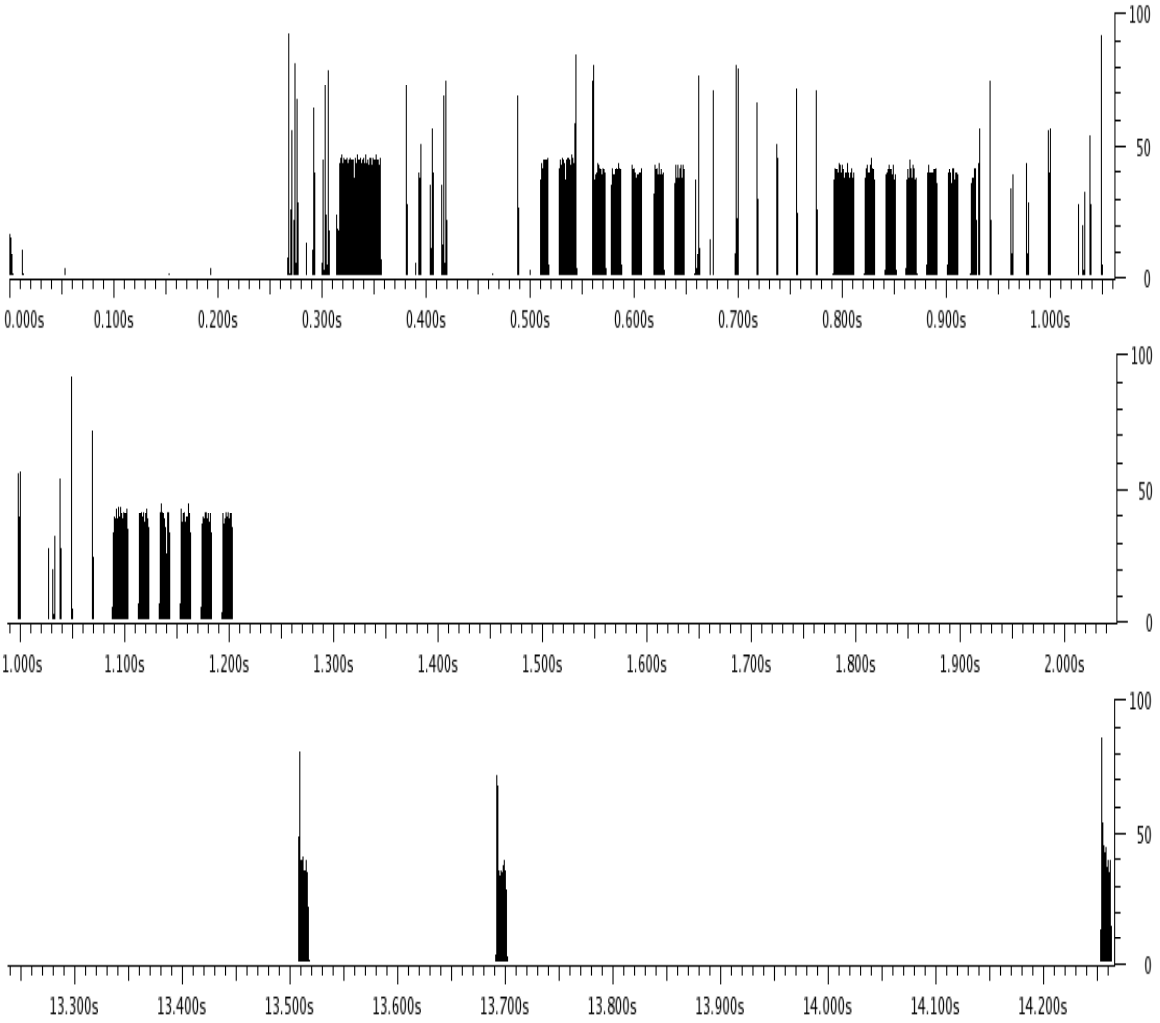

Fig. D.2: Multiplicação de matrizes - Pacotes transmitidos X Tempo (ms)

Tab. D.1: Dados utilizados para cálculo estimado de R_{cc}

Cálculo do π pelo método Monte Carlo								
	Tempo execução (ms)				Intervalo (ms)		R_{cc}	
	tarefa 1	tarefa 2	tarefa 3	total	SEM comunicação	COM comunicação	(pior caso)	(melhor caso)
1^o exp	4666	6926	8714	8876	9092	19	478,53	479,53
2^o exp	4572	6763	8542	8662	8861	22	402,77	403,77
3^o exp	6335	6984	8845	9072	9280	19	488,42	489,42
média				8870	9078	20	456,57	457,57

Multiplicação de matrizes								
	Tempo execução (ms)				Intervalo (ms)		R_{cc}	
	tarefa 1	tarefa 2	tarefa 3	total	SEM comunicação	COM comunicação	(pior caso)	(melhor caso)
1^o exp	12635	11759	12295	14079	13370	910	14,69	15,69
2^o exp	12565	12478	12295	14237	13450	990	13,59	14,59
3^o exp	12104	12500	13347	13718	13710	970	14,13	15,13
média				14011	13510	957	14,14	15,14

D.2 Funções estatítiscas utilizadas

Para possibilitar a análise e a comparação dos dados obtidos nos experimentos, foram utilizadas algumas funções estatíticas [114] descritas a seguir, embora esta abordagem seja mais adequada para grandes amostras.

- Média aritmética (μ):

  Utilizada para resumir dados aproximadamente simétricos. Se $v_1, v_2, \ldots, v_n$ são os valores dos dados, então calculamos a média como:

  $$\mu = \frac{\sum_{i=1}^{n} v_i}{n}$$

- Mediana (Md):

  É o valor que divide o conjunto ordenado de dados ao meio, ou seja, metade dos dados têm valores maiores do que a mediana e a outra metade tem valores menores do que a mediana.

- Variância (σ^2):

  A variância é definida como o "desvio quadrático médio da média" e é calculada como:

  $$\sigma^2 = \frac{\sum_{i=1}^{n}(v_i - \mu)^2}{(n-1)} = \frac{\sum_{i=1}^{n}(v_i^2) - n\mu^2}{(n-1)}$$

- Desvio padrão (σ):

  É a raiz quadrada positiva da variância, calculda como:

  $$\sigma = \sqrt{\sigma^2}$$

- Coeficiente de variação (CV):

  Expresso como a porcentagem do desvio padrão dividido pela média:

  $$CV = \frac{\sigma}{\mu} 100$$

- Erro padrão da média (*SE*):

  Define a variabilidade associada a uma estimativa e é calculado como:

  $$SE = \frac{\sigma}{\sqrt{n}}$$

- Intervalo de confiança:

  É usado para indicar a confiabilidade de uma estimativa. Em geral os intervalos de confiança são calculados no nível de 95% e fornecem intervalos de valores onde, na repetição de amostras, em 95% dos casos a média estará entre os valores fornecidos. O intervalo de confiança de 95% para experimentos com um grande número de amostras é calculado como:

$$(\mu - 1.96 \times \frac{\sigma}{\sqrt{n}}, \mu + 1.96 \times \frac{\sigma}{\sqrt{n}})$$

Para a obtenção de intervalos de confiança mais adequados a amostras pequenas devemos substituir o valor *1.96* por um valor obtido na tabela de distribuição t de Student, correspondente à linha $r = (n - 1)$ graus de liberdade. Nos resultados apresentados realizamos cinco repetições de cada experimento e, por esta razão, usamos o valor *2.776* (Tab. D.2) para calcular o intervalo de confiança de 95%.

Tab. D.2: Distribuição t de Student (parcial)

r	Nível de confiança				
	80%	90%	95%	99%	99,9%
1	3.078	6.314	12.706	63.657	636.619
2	1.886	2.920	4.303	9.925	31.599
3	1.638	2.353	3.182	5.841	12.924
4	1.533	2.132	2.776	4.604	8.610
5	1.476	2.015	2.571	4.032	6.869
10	1.372	1.812	2.228	3.169	4.587
30	1.310	1.697	2.042	2.750	3.646
∞	1.282	1.645	1.960	2.576	3.291

Média aritmética e mediana são medidas de tendência central. Erro padrão, desvio padrão, variância e coeficiente de variação são medidas de dispersão.

D.3 Dados para avaliação do impacto no desempenho

Na Tab. D.3, na Tab. D.4 e na Tab. D.5 são exibidos os dados obtidos nos experimentos realizados para observação do impacto da ferramenta de monitoramento JoiNMon no tempo de execução das aplicações, respectivamente, para cálculo de π pelo método de Monte Carlo, para multiplicação de matrizes com carga total fixa e para multiplicação de matrizes com tamanho das sub-matrizes fixas. Em cada uma destas tabelas são apresentados os dados obtidos, subdivididos conforme a etapa: a primeira, na coluna intitulada *Sem JoiNMon*, feita com o sistema JoiN sem a ferramenta de monitoramento; e a segunda, na coluna *JoiNMon ativo*, feita com a ferramenta JoiNMon ativada. Em cada linha das tabelas são apresentados os resultados obtidos nas execuções dos experimentos, para cada uma destas etapas, com mesmo número de componentes *worker* e de tarefas. São apresentados o valor médio do tempo de processamento da aplicação, o desvio padrão, o coeficiente de variação (expresso como a porcentagem do desvio padrão dividido pela média) e o intervalo de confiança de 95%, calculados com base nos valores obtidos nas repetições dos experimentos. Nas últimas colunas são apresentados a diferença entre os valores médios de execução e a sobrecarga que a ferramenta de monitoramento acarreta no tempo médio de execução destas aplicações.

Os valores dos coeficientes de variação e dos intervalos de confiança apresentados nas tabelas D.3, D.4 e D.5 são relativamente pequenos e, por esta razão, podemos considerar que os valores médios de tempo de execução das aplicações são apropriados para a avaliação de resultados.

Tab. D.3: Cálculo de π - Tempo de processamento (ms)

Num.	*Num.*	*Sem JoiNMon*				*JoiNMon ativo*				*Comparativo*	
workers	*tasks*	*Média*	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Diferença*	*Sobrecarga*
4	1000	740744	1220	0,16%	±1515	751638	2241	0,30%	±2782	10894	1.47%
	2000	746631	1622	0,22%	±2014	754101	2209	0,29%	±2742	7470	1.00%
	4000	771479	2149	0,28%	±2668	775437	2983	0,38%	±3704	3958	0.51%
8	1000	345839	1703	0,49%	±2115	356818	1273	0,36%	±1580	10979	3.17%
	2000	353007	2302	0,65%	±2858	360904	3154	0,87%	±3915	7897	2.24%
	4000	366108	1733	0,47%	±2152	367573	2719	0.74%	±3376	1465	0.40%
12	1000	234378	1493	0,64%	±1854	242675	521	0,21%	±647	8297	3.54%
	2000	234806	728	0,31%	±904	242400	955	0,39%	±1186	7594	3.23%
	4000	242054	3313	1,37%	±4113	249095	499	0,20%	±620	7041	2.91%
16	1000	177367	4008	2,26%	±4976	183127	1101	0,60%	±1367	5760	3.25%
	2000	178814	765	0,43%	±949	182350	862	0,47%	±1070	3536	1.98%
	4000	183012	361	0,20%	±448	185463	2222	1,20%	±2758	2450	1.34%
24	1000	114222	1220	1,07%	±1515	117796	231	0,20%	±286	3573	3.13%
	2000	115706	2597	2,24%	±3224	119159	1057	0,89%	±1313	3454	2.98%
	4000	121673	1201	0,99%	±1491	125140	562	0,45%	±698	3467	2.85%
32	1000	93388	746	0,80%	±926	94679	1082	1,14%	±1343	1291	1.38%
	2000	93837	371	0,40%	±460	94600	657	0,69%	±816	763	0.81%
	4000	98154	71	0,07%	±88	98563	666	0,68%	±827	409	0.42%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança

Tab. D.4: Multiplicação de matrizes - Tempo de processamento (ms)

Num.	*Num.*	*Sem JoiNMon*				*JoiNMon ativo*				*Comparativo*	
workers	*tasks*	*Média*	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Diferença*	*Sobrecarga*
4	100	117918	4042	3,43%	±5018	121902	1123	0,92%	±1394	3984	3,38%
	200	120471	2880	2,39%	±3575	124219	811	0,65%	±1007	3749	3,11%
	400	125064	610	0,49%	±757	128819	596	0,46%	±740	3755	3,00%
8	100	62373	1177	1,89%	±1461	64013	179	0,28%	±222	1641	2,63%
	200	65426	604	0,92%	±750	67069	835	1,24%	±1036	1643	2,51%
	400	73941	677	0,92%	±841	75260	696	0,93%	±865	1319	1,78%
12	100	49414	1050	2,13%	±1304	51422	757	1,47%	±940	2008	4,06%
	200	53150	650	1,22%	±807	55032	493	0,90%	±612	1882	3,54%
	400	60408	634	1,05%	±787	62685	403	0,64%	±500	2277	3,77%
16	100	43707	417	0,95%	±518	45589	540	1,18%	±671	1882	4,31%
	200	47453	527	1,11%	±655	49255	845	1,72%	±1049	1801	3,80%
	400	57591	451	0,78%	±560	59549	748	1,26%	±929	1958	3,40%
24	100	35776	213	0,60%	±264	36816	218	0,59%	±271	1040	2,91%
	200	38708	464	1,20%	±575	40847	531	1,30%	±660	2139	5,52%
	400	45068	381	0,85%	±473	46713	542	1,16%	±673	1645	3,65%
32	100	30078	406	1,35%	±504	31523	508	1,61%	±630	1446	4,81%
	200	31130	289	0,93%	±358	32142	436	1,36%	±541	1013	3,25%
	400	36977	661	1,79%	±820	38600	388	1,00%	±481	1623	4,39%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança

Tab. D.5: Multiplicação de matrizes de tamanho fixo - Tempo de processamento (ms)

Num. workers	*Num. tasks*	*Sem JoiNMon*					*JoiNMon ativo*					*Comparativo*	
		Média	*DesvPad*	*CV*	*IntConf*	*Rép*	*Média*	*DesvPad*	*CV*	*IntConf*	*Rép*	*Diferença*	*Sobrecarga*
4	100	16719	351	2.10%	±436	9	17108	326	1.91%	±405	10	389	2.33%
	200	28539	290	1.02%	±360	9	29031	755	2.60%	±937	9	492	1.72%
	400	59165	914	1.54%	±1135	8	61039	1025	1.68%	±1273	9	1874	3.17%
16	100	6485	198	3.05%	±246	41	6616	169	2.55%	±210	38	131	2.02%
	200	11373	281	2.47%	±349	40	11762	416	3.54%	±516	38	389	3.42%
	400	23924	773	3.23%	±960	44	24671	631	2.56%	±783	46	747	3.12%
32	100	3741	95	2.54%	±118	90	3825	88	2.30%	±109	99	84	2.25%
	200	6327	203	3.21%	±252	79	6511	129	1.98%	±160	76	184	2.91%
	400	11081	335	3.02%	±416	81	11483	514	4.48%	±638	85	402	3.63%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança; Rép = número réplicas

D.4 Dados para avaliação do impacto no tráfego de rede

D.4.1 Tráfego de rede nos experimentos com aplicação para o cálculo de π

Na Tab. D.6 são exibidos os dados relativos ao tráfego de rede acumulado nas portas TCP/8000, TCP/8100 e TCP/9001 obtidos nos experimentos realizados com a aplicação para o cálculo de π pelo método de Monte Carlo. Na Tab. D.7 são exibidos os dados relativos ao tráfego de rede em cada uma das portas TCP/8000, TCP/8100 e TCP/9001, além do tráfego acumulado nestas portas.

Em cada uma destas tabelas são apresentados resumos dos resultados obtidos, subdivididos conforme a etapa: a primeira, intitulada *Sem JoiNMon*, feita com o sistema JoiN sem a ferramenta de monitoramento; e a segunda, intitulada *JoiNMon ativo*, feita com a ferramenta JoiNMon ativada. Para cada uma destas etapas, na coluna *Média* é apresentada a média, em número de bytes, dos valores obtidos nas cinco execuções do experimento, com mesmo número de componentes *worker* e de tarefas. Nas colunas seguintes são apresentados, também em número de bytes, o desvio padrão e o coeficiente de variação (expresso como a porcentagem do desvio padrão dividido pela média) e o intervalo de confiança de 95%, calculados com base nos valores obtidos nas cinco execuções. Nas últimas colunas são apresentados a diferença entre os valores médios de cada etapa e a sobrecarga que a ferramenta de monitoramento acarreta no tráfego de rede durante a execução desta aplicação.

Os valores dos coeficientes de variação e dos intervalos de confiança apresentados nas tabelas D.6 e D.7 são relativamente pequenos. Os valores dos coeficientes de variação são sempre inferiores a 10%. Dos 126 valores existentes, 94 são inferiores a 5%, e os valores mais altos são obtidos quando o volume de tráfego é menor e, por este motivo, as medições são mais sujeitas a imprecisões. Estes valores de coeficiente de variação nos fornecem uma idéia da precisão dos dados obtidos nestes resultados experimentais e, como são relativamente pequenos, também neste caso podemos trabalhar com os valores médios.

Tab. D.6: Cálculo de π - Tráfego de rede (bytes)

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
			Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
4	1000	total	2299257	65840	2.86%	±81739	5111225	106585	2,09%	±132322	2811968	122,30%
4	2000	total	4427220	154343	3,49%	±191611	9770586	191126	1,96%	±237276	5343365	120,69%
4	4000	total	7768649	116226	1,50%	±144290	19052709	330041	1,73%	±409735	11284059	145,25%
8	1000	total	2226505	121581	5,46%	±150939	5053188	105126	2,08%	±130510	2826682	126,96%
8	2000	total	3902450	162715	4,17%	±202006	9600096	175106	1,82%	±217388	5697646	146,00%
8	4000	total	7098767	54605	0,77%	±67791	18124585	529059	2,92%	±656809	11025818	155,32%
12	1000	total	2281480	128876	5,65%	±159995	4945212	201692	4,08%	±250394	2663732	116,75%
12	2000	total	3729910	102863	2,76%	±127702	9300777	260753	2,80%	±323717	5570866	149,36%
12	4000	total	7208269	24816	0,34%	±30809	18019664	478333	2,65%	±593834	10811394	149,99%
16	1000	total	2164455	138029	6,38%	±171358	4947631	107854	2,18%	±133897	2783175	128,59%
16	2000	total	3715807	76195	2,05%	±94593	9296602	229014	2,46%	±284313	5580795	150,19%
16	4000	total	7030726	55551	0,79%	±68965	17287309	780373	4,51%	±968806	10256583	145,88%
24	1000	total	2074745	126832	6,11%	±157459	4763200	183099	3,84%	±227311	2688455	129,58%
24	2000	total	3708900	30120	0,81%	±37394	8834883	268378	3,04%	±333182	5125983	138,21%
24	4000	total	6928067	73221	1,06%	±90901	17532965	460503	2,63%	±571699	10604897	153,07%
32	1000	total	2138950	34214	1,60%	±42475	4949733	157351	3,18%	±195346	2810783	131,41%
32	2000	total	3703673	95222	2,57%	±118214	9087978	257672	2,84%	±319891	5384305	145,38%
32	4000	total	6832446	75987	1,11%	±94335	17258274	436873	2,53%	±542363	10425827	152,59%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança
Dif = diferença; SobreC = sobrecarga

Tab. D.7: Cálculo de π - Tráfego de rede (bytes) em cada porta TCP

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
			Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
4	1000	8000	13803	1322	9,58%	±1641	156669	5898	3,76%	±7323	142865	1035,02%
		8100	2285454	63578	2,78%	±78930	2187195	81161	3,71%	±100759	-98259	-4,30%
		9001	-	-	-	-	2767361	33018	1,19%	±40991	-	-
		total	2299257	65840	2,86%	±81739	5111225	106585	2,09%	±132322	2811968	122,30%
4	2000	8000	13094	518	3,96%	±643	230704	3851	1,67%	±4781	217610	1661,91%
		8100	4414126	154548	3,50%	±191866	4065256	74194	1,83%	±92110	-348870	-7,90%
		9001	-	-	-	-	5474625	118947	2,17%	±147669	-	-
		total	4427220	154343	3,49%	±191611	9770586	191126	1,96%	±237276	5343365	120,69%
4	4000	8000	15408	1523	9,88%	±1891	378648	3683	0,97%	±4572	363240	2357,47%
		8100	7753241	115230	1,49%	±143054	7860775	161435	2,05%	±200416	107533	1,39%
		9001	-	-	-	-	10813287	334272	3,09%	±414988	-	-
		total	7768649	116226	1,50%	±144290	19052709	330041	1,73%	±409735	11284059	145,25%
8	1000	8000	12189	1131	9,28%	±1404	123323	4255	3,45%	±5282	111134	911,77%
		8100	2214317	122996	5,10%	±140281	2161991	110316	5,10%	±136954	-52326	-2,36%
		9001	-	-	-	-	2767875	38995	1,41%	±48411	-	-
		total	2226505	121581	5,46%	±150939	5053188	105126	2,08%	±130510	2826683	126,96%
8	2000	8000	15119	1004	6,64%	±1246	195920	7002	3,57%	±8692	180801	1195,87%
		8100	3887331	163185	4,20%	±202588	3919612	36028	0,92%	±44727	32280	0,83%
		9001	-	-	-	-	5484565	156784	2,86%	±194642	-	-
		total	3902450	162716	4,17%	±202006	9600097	175106	1,82%	±217388	5697646	146,00%
8	4000	8000	12394	1144	9,23%	±1420	351526	12082	3,44%	±15000	339132	2736,35%
		8100	7086373	55453	0,78%	±68842	7127065	47200	0,66%	±58597	40692	0,57%
		9001	-	-	-	-	10645994	512830	4,82%	±636660	-	-
		total	7098767	54606	0,77%	±67791	18124585	529060	2,92%	±656809	11025818	155,32%
12	1000	8000	11081	609	5,50%	±757	124115	9152	7,37%	±11362	113034	1020,11%
		8100	2270399	128928	5,68%	±160060	2095407	139395	6,65%	±173054	-174993	-7,71%
		9001	-	-	-	-	2725691	110473	4,05%	±137148	-	-

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança
Dif = diferença; SobreC = sobrecarga

continua na próxima página

Tab. D.7: Cálculo de π - Tráfego de rede (bytes) em cada porta TCP (continuação)

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
			Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
		total	2281480	128876	5,65%	±159995	4945213	201693	4,08%	±250394	2663733	116,75%
12	2000	8000	15539	1531	9,85%	±1901	195875	6525	3,33%	±8101	180336	1160,52%
		8100	3714371	103562	2,79%	±128568	3723322	98400	2,64%	±122160	8951	0,24%
		9001	-	-	-	-	5381579	192915	3,58%	±239498	-	-
		total	3729911	102864	2,76%	±127702	9300777	260754	2,80%	±323717	5570866	149,36%
12	4000	8000	11487	1090	9,49%	±1354	352521	3973	1,13%	±4933	341033	2968,76%
		8100	7196782	25885	0,36%	±32135	7209382	84650	1,17%	±105090	12600	0,18%
		9001	-	-	-	-	10457762	483130	4,62%	±599788	-	-
		total	7208270	24817	0,34%	±30809	18019664	478334	2,65%	±593834	10811395	149,99%
16	1000	8000	11601	1149	9,91%	±1427	113829	5544	4,87%	±6882	102228	881,20%
		8100	2152855	139101	6,46%	±172689	1986326	120876	6,09%	±150063	-166529	-7,74%
		9001	-	-	-	-	2847476	43049	1,51%	±53443	-	-
		total	2164456	138029	6,38%	±171358	4947631	107854	2,18%	±133897	2783175	128,59%
16	2000	8000	12988	1288	9,92%	±1599	186175	2643	1,42%	±3281	173186	1333,41%
		8100	3702819	76182	2,06%	±94577	3645044	43192	1,18%	±53621	-57775	-1,56%
		9001	-	-	-	-	5465384	212123	3,88%	±263344	-	-
		total	3715807	76195	2,05%	±94593	9296603	229014	2,46%	±284313	5580795	150,19%
16	4000	8000	10596	926	8,74%	±1149	352206	16359	4,64%	±20309	341610	3224,02%
		8100	7020131	56044	0,80%	±69577	6678462	185527	2,78%	±230325	-341669	-4,87%
		9001	-	-	-	-	10256642	712981	6,95%	±885140	-	-
		total	7030727	55551	0,79%	±68965	17287310	780373	4,51%	±968806	10256583	145,88%
24	1000	8000	10621	1003	9,45%	±1246	103836	451	0,43%	±559	93215	877,62%
		8100	2064124	127861	6,19%	±158736	2006872	82773	4,12%	±102760	-57251	-2,77%
		9001	-	-	-	-	2652492	236504	8,92%	±293612	-	-
		total	2074745	126833	6,11%	±157459	4763201	183099	3,84%	±227311	2688456	129,58%
24	2000	8000	13531	811	5,99%	±1007	189143	8564	4,53%	±10632	175612	1297,83%
		8100	3695369	30579	0,83%	±37963	3628281	167614	4,62%	±208087	-67088	-1,82%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança
Dif = diferença; SobreC = sobrecarga

continua na próxima página

Tab. D.7: Cálculo de π - Tráfego de rede (bytes) em cada porta TCP (continuação)

Num.	*Num.*	*Porta*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
workers	*tasks*	*TCP*	*Média*	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
		9001	-	-	-	-	5017460	275948	5,50%	±342580	-	-
		total	3708900	30121	0,81%	±37394	8834884	268378	3,04%	±333182	5125984	138,21%
24	4000	8000	9723	905	9,31%	±1123	344773	8739	2,53%	±10850	335050	3446,10%
		8100	6918345	72181	1,04%	±89610	6605870	34455	0,52%	±42775	-312476	-4,52%
		9001	-	-	-	-	10582323	474314	4,48%	±588844	-	-
		total	6928068	73221	1,06%	±90901	17532965	460504	2,63%	±571699	10604897	153,07%
32	1000	8000	12617	1206	9,56%	±1497	105755	4528	4,28%	±5621	93138	738,22%
		8100	2126333	36085	1,70%	±44799	2086264	78787	3,78%	±97811	-40069	-1,88%
		9001	-	-	-	-	2757714	113902	4,13%	±141405	-	-
		total	2138950	34214	1,60%	±42475	4949733	157351	3,18%	±195346	2810783	131,41%
32	2000	8000	13435	1153	8,58%	±1431	187747	3240	1,73%	±4023	174312	1297,43%
		8100	3690237	95079	2,58%	±118037	3710979	55559	1,50%	±68974	20742	0,56%
		9001	-	-	-	-	5189252	242873	4,68%	±301519	-	-
		total	3703673	95222	2,57%	±118214	9087978	257672	2,84%	±319891	5384305	145,38%
32	4000	8000	10841	1029	9,49%	±1277	342224	15395	4,50%	±19112	331383	3056,87%
		8100	6821606	73756	1,08%	±91565	6652754	75334	1,13%	±93524	-168852	-2,48%
		9001	-	-	-	-	10263296	416139	4,05%	±516622	-	-
		total	6832446	75987	1,11%	±94335	17258274	436873	2,53%	±542363	10425827	152,59%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança
Dif = diferença; SobreC = sobrecarga

D.4.2 Tráfego de rede nos experimentos com aplicação para multiplicação de matrizes

Na Tab. D.8 são exibidos os dados relativos ao tráfego de rede acumulado nas portas TCP/8000, TCP/8100 e TCP/9001 obtidos nos experimentos realizados com a aplicação para multiplicação de matrizes. Na Tab. D.9 são exibidos os dados relativos ao tráfego de rede em cada uma das portas TCP/8000, TCP/8100 e TCP/9001, além do tráfego acumulado nestas portas.

Em cada uma destas tabelas são apresentados resumos dos resultados obtidos, subdivididos conforme a etapa: a primeira, intitulada *Sem JoiNMon*, feita com o sistema JoiN sem a ferramenta de monitoramento; e a segunda, intitulada *JoiNMon ativo*, feita com a ferramenta JoiNMon ativada. Para cada uma destas etapas, na coluna *Média* é apresentada a média, em número de bytes, dos valores obtidos nas cinco execuções do experimento, com mesmo número de componentes *worker* e de tarefas. Nas colunas seguintes são apresentados, também em número de bytes, o desvio padrão e o coeficiente de variação (expresso como a porcentagem do desvio padrão dividido pela média) e o intervalo de confiança de 95%, calculados com base nos valores obtidos nas cinco execuções. Nas últimas colunas são apresentados a diferença entre os valores médios de cada etapa e a sobrecarga que a ferramenta de monitoramento acarreta no tráfego de rede durante a execução desta aplicação.

Os valores dos coeficientes de variação e dos intervalos de confiança apresentados nas tabelas D.8 e D.9 são relativamente pequenos. Os valores dos coeficientes de variação são sempre inferiores a 8,2%. Dos 126 valores existentes, 118 são inferiores a 5% sendo que 91 são inferiores a 2%. Também neste caso os valores mais altos são obtidos quando o volume de tráfego é menor e, por este motivo, as medições são mais sujeitas a imprecisões. Desta forma, aqui também podemos considerar os valores médios nas avaliações sem introduzir grandes erros.

Tab. D.8: Multiplicação de matrizes - Tráfego total de rede (bytes)

Num. workers	*Num. tasks*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
		Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
4	100	859458050	2660674	0.31%	±3303134	933142668	6941855	0.74%	±8618070	73684618	8.57%
4	200	1405869881	19292213	1.37%	±23950606	1454930819	2944124	0.20%	±3655027	49060938	3.49%
4	400	2114298003	43749575	2.07%	±54313564	2261447769	49221208	2.18%	±61106404	147149766	6.96%
8	100	895860471	2554109	0.29%	±3170837	982578918	7898802	0.80%	±9806086	86718447	9.68%
8	200	1610178481	15283130	0.95%	±18973470	1628475182	12945910	0.79%	±16071893	18296701	1.14%
8	400	2164569250	57408069	2.65%	±71270105	2290321737	36145147	1.58%	±44872933	125752487	5.81%
12	100	732872939	2176866	0.30%	±2702503	788326140	2206703	0.28%	±2739544	55453201	7.57%
12	200	921908010	3571823	0.39%	±4434293	982779698	7633941	0.78%	±9477270	60871688	6.60%
12	400	1449744459	6928392	0.48%	±8601356	1477668124	5308848	0.36%	±6590749	27923664	1.93%
16	100	731024542	8722025	1.19%	±10828088	747245575	3189858	0.43%	±3960097	16221033	2.22%
16	200	849998306	11700804	1.38%	±14526138	870066249	1341699	0.15%	±1665672	20067943	2.36%
16	400	1292466994	12560915	0.97%	±15593936	1348651037	9802818	0.73%	±12169855	56184044	4.35%
24	100	561845022	6509386	1.16%	±8081174	602622727	1517660	0.25%	±1884122	40777705	7.26%
24	200	723424700	8884855	1.23%	±11030236	795224986	4078615	0.51%	±5063458	71800286	9.93%
24	400	1155046141	21767701	1.88%	±27023838	1178038466	5358361	0.45%	±6652217	22992325	1.99%
32	100	683810855	9461327	1.38%	±11745906	737762598	3585050	0.49%	±4450714	53951744	7.89%
32	200	804971164	8072436	1.00%	±10021647	859552460	1688491	0.20%	±2096202	54581296	6.78%
32	400	1207334347	11267340	0.93%	±13988008	1314728253	1937890	0.15%	±2405823	107393906	8.90%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança
Dif = diferença; SobreC = sobrecarga

Tab. D.9: Multiplicação de matrizes - Tráfego de rede (bytes) em cada porta TCP

workers/ tasks	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
		Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
4 / 100	8000	9527	211	2.22%	±262	24757	874	3.53%	±1085	15230	159.87%
	8100	859448523	2660546	0.31%	±3302975	932833207	6938147	0.74%	±8613467	73384684	8.54%
	9001					284704	6291	2.21%	±7810		
	total	859458050	2660674	0.31%	±3303134	933142668	6941855	0.74%	±8618070	73684618	8.57%
4 / 200	8000	10044	351	3.50%	±436	38759	1033	2.66%	±1282	28715	285.89%
	8100	1405859837	19292228	1.37%	±23950625	1454531475	2943824	0.20%	±3654655	48671637	3.46%
	9001					360585	3181	0.88%	±3949		
	total	1405869881	19292213	1.37%	±23950606	1454930819	2944124	0.20%	±3655027	49060938	3.49%
4 / 400	8000	14129	644	4.56%	±800	53352	668	1.25%	±829	39223	277.61%
	8100	2114283874	43749108	2.07%	±54312984	2260665600	49213310	2.18%	±61096599	146381726	6.92%
	9001					728817	12347	1.69%	±15328		
	total	2114298003	43749575	2.07%	±54313564	2261447769	49221208	2.18%	±61106404	147149766	6.96%
8 / 100	8000	11712	740	6.31%	±918	24246	450	1.86%	±559	12535	107.03%
	8100	895848759	2554309	0.29%	±3171085	982554672	7898474	0.80%	±9805679	86705913	9.68%
	9001					309255	8686	2.81%	±10784		
	total	895860471	2554109	0.29%	±3170837	982578918	7898802	0.80%	±9806086	86718447	9.68%
8 / 200	8000	10595	223	2.11%	±277	35595	729	2.05%	±905	25000	235.96%
	8100	1610167886	15283170	0.95%	±18973519	1628439586	12946164	0.80%	±16072208	18271701	1.13%
	9001					395687	5918	1.50%	±7347		
	total	1610178481	15283130	0.95%	±18973470	1628475182	12945910	0.79%	±16071893	18296701	1.14%
8 / 400	8000	16095	923	5.73%	±1146	49223	1295	2.63%	±1607	33128	205.83%
	8100	2164553155	57408958	2.65%	±71271209	2290272514	36144448	1.58%	±44872065	125719359	5.81%
	9001					802402	3868	0.48%	±4802		
	total	2164569250	57408069	2.65%	±71270105	2290321737	36145147	1.58%	±44872933	125752487	5.81%
12 / 100	8000	8130	599	7.37%	±744	31838	278	0.87%	±345	23708	291.62%
	8100	732864809	2176884	0.30%	±2702526	788294302	2206928	0.28%	±2739823	55429493	7.56%
	9001					317567	8682	2.73%	±10778		

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança; Dif = diferença; SobreC = sobrecarga

continua na próxima página

Tab. D.9: Multiplicação de matrizes - Tráfego de rede (bytes) em cada porta TCP (continuação)

workers/ tasks	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
		Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
	total	732872939	2176866	0.30%	±2702503	788326140	2206703	0.28%	±2739544	55453201	7.57%
12 / 200	8000	8474	542	6.40%	±673	19804	531	2.68%	±659	11330	133.71%
	8100	921899536	3571419	0.39%	±4433791	982759894	7633719	0.78%	±9476995	60860358	6.60%
	9001					458563	8583	1.87%	±10656		
	total	921908010	3571823	0.39%	±4434293	982779698	7633941	0.78%	±9477270	60871688	6.60%
12 / 400	8000	9433	243	2.58%	±302	38311	831	2.17%	±1032	28878	306.13%
	8100	1449735026	6928391	0.48%	±8601354	1477629813	5309295	0.36%	±6591303	27894787	1.92%
	9001					888482	8275	0.93%	±10273		
	total	1449744459	6928392	0.48%	±8601356	1477668124	5308848	0.36%	±6590749	27923664	1.93%
16 / 100	8000	9282	511	5.50%	±634	42899	569	1.33%	±706	33617	362.17%
	8100	731015260	8721658	1.19%	±10827633	746890646	3191078	0.43%	±3961611	15875386	2.17%
	9001					312030	6641	2.13%	±8245		
	total	731024542	8722025	1.19%	±10828088	747245575	3189858	0.43%	±3960097	16221033	2.22%
16 / 200	8000	8574	699	8.15%	±868	34903	716	2.05%	±889	26329	307.06%
	8100	849989731	11700794	1.38%	±14526126	869646061	1341036	0.15%	±1664849	19656330	2.31%
	9001					385285	8503	2.21%	±10556		
	total	849998306	11700804	1.38%	±14526138	870066249	1341699	0.15%	±1665672	20067943	2.36%
16 / 400	8000	9167	663	7.23%	±823	36052	580	1.61%	±720	26885	293.27%
	8100	1292457827	12561142	0.97%	±15594218	1348614985	9802739	0.73%	±12169757	56157159	4.34%
	9001					912528	3394	0.37%	±4213		
	total	1292466994	12560915	0.97%	±15593936	1348651037	9802818	0.73%	±12169855	56184044	4.35%
24 / 100	8000	9658	383	3.96%	±475	35082	469	1.34%	±582	25424	263.26%
	8100	561835364	6509737	1.16%	±8081611	602587645	1517549	0.25%	±1883983	40752281	7.25%
	9001					351981	3017	0.86%	±3746		
	total	561845022	6509386	1.16%	±8081174	602622727	1517660	0.25%	±1884122	40777705	7.26%
24 / 200	8000	8264	415	5.02%	±515	35533	947	2.67%	±1176	27269	329.97%
	8100	723416436	8884981	1.23%	±11030392	795189454	4079070	0.51%	±5064022	71773017	9.92%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança; Dif = diferença; SobreC = sobrecarga

continua na próxima página

Tab. D.9: Multiplicação de matrizes - Tráfego de rede (bytes) em cada porta TCP (continuação)

workers/ tasks	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
		Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
	9001					432188	572	0.13%	±710		
	total	723424700	8884855	1.23%	±11030236	795224986	4078615	0.51%	±5063458	71800286	9.93%
24 / 400	8000	8107	159	1.96%	±198	36046	305	0.85%	±378	27939	344.65%
	8100	1155038034	21767784	1.88%	±27023941	1178002420	5358345	0.45%	±6652198	22964386	1.99%
	9001					890565	5184	0.58%	±6435		
	total	1155046141	21767701	1.88%	±27023838	1178038466	5358361	0.45%	±6652217	22992325	1.99%
32 / 100	8000	9799	111	1.13%	±138	42389	738	1.74%	±916	32590	332.60%
	8100	683801056	9461248	1.38%	±11745808	737720210	3584561	0.49%	±4450107	53919154	7.89%
	9001					368595	4737	1.29%	±5880		
	total	683810855	9461327	1.38%	±11745906	737762598	3585050	0.49%	±4450714	53951744	7.89%
32 / 200	8000	10045	84	0.84%	±104	49658	2118	4.27%	±2630	39613	394.37%
	8100	804961119	8072492	1.00%	±10021716	859502802	1688931	0.20%	±2096748	54541683	6.78%
	9001					463948	4499	0.97%	±5585		
	total	804971164	8072436	1.00%	±10021647	859552460	1688491	0.20%	±2096202	54581296	6.78%
32 / 400	8000	11546	336	2.91%	±418	50688	887	1.75%	±1101	39142	339.01%
	8100	1207322801	11267075	0.93%	±13987679	1314677564	1938222	0.15%	±2406235	107354763	8.89%
	9001					934642	1575	0.17%	±1955		
	total	1207334347	11267340	0.93%	±13988008	1314728253	1937890	0.15%	±2405823	107393906	8.90%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança; Dif = diferença; SobreC = sobrecarga

D.4.3 Tráfego de rede nos experimentos com aplicação para multiplicação de matrizes de tamanho fixo

Na Tab. D.10 são exibidos os dados relativos ao tráfego de rede em cada uma das portas TCP/8000, TCP/8100 e TCP/9001, além do tráfego acumulado nestas portas.

São apresentados os resultados obtidos, subdivididos conforme a etapa: a primeira, intitulada *Sem JoiNMon*, feita com o sistema JoiN sem a ferramenta de monitoramento; e a segunda, intitulada *JoiNMon ativo*, feita com a ferramenta JoiNMon ativada. Para cada uma destas etapas, com mesmo número de componentes *worker* e de tarefas, são apresentados, em número de bytes, o valor médio, o desvio padrão e o coeficiente de variação (expresso como a porcentagem do desvio padrão dividido pela média) e o intervalo de confiança de 95%, calculados com base nos valores obtidos nas três execuções. Nas duas últimas colunas são apresentados a diferença entre os valores médios de cada etapa e a sobrecarga que a ferramenta de monitoramento acarreta no tráfego de rede durante a execução desta aplicação.

Os valores dos coeficientes de variação e dos intervalos de confiança apresentados nesta tabela também são relativamente pequenos. Os valores de coeficiente de variação são sempre inferiores a 4,2%. Também neste caso os valores mais altos são obtidos quando o volume de tráfego é menor e, por este motivo, as medições são mais sujeitas a imprecisões. Desta forma, aqui também podemos considerar os valores médios nas avaliações sem introduzir grandes erros.

Tab. D.10: Multiplicação de matrizes de tamanho fixo - Tráfego de rede (bytes) em cada porta TCP

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
			Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
4	100	8000	7392	201	2.72%	±250	20887	508	2.43%	±631	13495	182.56%
		8100	55303901	156592	0.28%	±194403	55796420	175665	0.31%	±218082	492519	0.89%
		9001	0				309726	3086	1.00%	±3831		
		total	55311293	156894	0.28%	±194778	56127033	176052	0.31%	±218562	815740	1.47%
4	200	8000	7157	199	2.78%	±247	32222	968	3.00%	±1202	25065	350.22%
		8100	116878281	1056147	0.90%	±1311169	117669102	1146982	0.97%	±1423938	790821	0.68%
		9001	0				471312	3368	0.71%	±4181		
		total	116885438	1056336	0.90%	±1311404	118172636	1147751	0.97%	±1424893	1287198	1.10%
4	400	8000	7721	248	3.21%	±308	48575	1352	2.78%	±1678	40854	529.13%
		8100	253001495	2214522	0.88%	±2749251	258229026	2498567	0.97%	±3101883	5227531	2.07%
		9001	0				726007	8974	1.24%	±11141		
		total	253009216	2214493	0.88%	±2749215	259003608	2499671	0.97%	±3103254	5994392	2.37%
16	100	8000	7392	258	3.49%	±320	36090	703	1.95%	±873	28698	388.23%
		8100	55971593	600478	1.07%	±745472	56111231	712543	1.27%	±884597	139638	0.25%
		9001	0				282689	2501	0.88%	±3105		
		total	55978985	600365	1.07%	±745332	56430010	712988	1.26%	±885150	451025	0.81%
16	200	8000	9158	339	3.70%	±421	54169	1038	1.92%	±1289	45011	491.49%
		8100	112604028	1458721	1.30%	±1810951	113438176	1568893	1.38%	±1947726	834148	0.74%
		9001	0				537164	4984	0.93%	±6187		
		total	112613186	1458996	1.30%	±1811292	114029509	1569592	1.38%	±1948593	1416323	1.26%
16	400	8000	9509	298	3.13%	±370	76977	1284	1.67%	±1594	67468	709.52%
		8100	252249016	3012879	1.19%	±3740384	255052942	3445288	1.35%	±4277204	2803926	1.11%
		9001	0				1124804	11475	1.02%	±14246		
		total	252258525	3013097	1.19%	±3740654	256254723	3446274	1.34%	±4278428	3996198	1.58%
32	100	8000	7305	245	3.35%	±304	32976	691	2.10%	±858	25671	351.42%
		8100	59865596	786547	1.31%	±976471	61360031	867942	1.41%	±1077520	1494435	2.50%
		9001	0				373433	5065	1.36%	±6288		

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança; Dif = diferença; SobreC = sobrecarga

continua na próxima página

Tab. D.10: Multiplicação de matrizes de tamanho fixo - Tráfego de rede (bytes) em cada porta TCP (continuação)

Num. workers	*Num. tasks*	*Porta TCP*	*Sem JoiNMon*				*Com JoiNMon*				*Comparativo*	
			Média	*DesvPad*	*CV*	*IntConf*	*Média*	*DesvPad*	*CV*	*IntConf*	*Dif*	*SobreC*
		total	59872901	786424	1.31%	±976318	61766440	868388	1.41%	±1078073	1893539	3.16%
32	200	8000	7528	311	4.13%	±386	44219	1156	2.61%	±1435	36691	487.39%
		8100	108189997	1896545	1.75%	±2354494	109855070	1982355	1.80%	±2461024	1665073	1.54%
		9001	0				773274	7478	0.97%	±9284		
		total	108197525	1896823	1.75%	±2354839	110672563	1983200	1.79%	±2462073	2475038	2.29%
32	400	8000	8052	274	3.40%	±340	53953	1510	2.80%	±1875	45901	570.06%
		8100	246599017	2998700	1.22%	±3722781	246952335	3209873	1.30%	±3984945	353318	0.14%
		9001	0				938985	10861	1.16%	±13484		
		total	246607069	2998854	1.22%	±3722972	247945273	3211109	1.30%	±3986479	1338204	0.54%

DesvPad = desvio padrão; CV = coeficiente de variação (%); IntConf = intervalo de confiança;
Dif = diferença; SobreC = sobrecarga

Apêndice E

Definições

Neste apêndice são apresentados alguns conceitos e definições citados neste trabalho que, por não serem objeto deste estudo, não são explorados em detalhes. Estas definições foram obtidas no endereço http://www.webopedia.com/ e traduzidas livremente.

- BXXP - acrônimo para *Blocks Extensible Exchange Protocol*, é um protocolo para aplicações orientadas à conexão, que executam interações assíncronas para requisições e respostas. Foi projetada como um conjunto de ferramentas para manipulação de documentos XML na Internet. As aplicações com suporte a BXXP estabelecem e mantêm conexões *peer-to-peer* entre dois usuários, que podem alternar atuações como servidores e clientes. Estes dois usuários podem transmitir e receber dados por meio da conexão estabelecida, bem como responder a requisições de dados. O protocolo BXXP situa-se logo acima da camada TCP e é uma alternativa ao protocolo HTTP ou a outros protocolos para troca de dados. Como o HTTP não é eficiente na transferência de dados XML, o protocolo BXXP permite o desenvolvimento de protocolos específicos para suportar o tráfego de trocas múltiplas e simultâneas de dados XML. Por exemplo, os usuários podem trocar mensagens por meio de *chat* e transferir arquivos ao mesmo tempo, por meio de um única aplicação que emprega esta conexão.
- IHC - acrônimo para Interação Humano-Computador, é a área dedicada ao estudo, ao projeto, à construção e à execução de sistemas onde os processos interativos são centrados no ser humano. O ser humano interage com o computador por meio de uma *interface*. A IHC vai além de projetar telas e menus mais fáceis de usar. Ela estuda o raciocínio necessário para implementação de cada funcionalidade específica e os efeitos que os sistemas terão nos seres humanos a longo prazo. IHC é uma disciplina muito extensa, que abrange diferentes especialidades, com interesses distintos a respeito do uso do computador: na informática interessa-se pelo projeto da aplicação e a engenharia das interfaces; na sociologia e na antropologia preocupa-se com as interações entre tecnologia, trabalho e organização e com a maneira que os humanos e os sistemas se adaptam mutuamente; na ergonomia é avaliada a segurança dos sistemas e os limites seguros da cognição e da sensação humanas; a preocupação na área de psicologia é com os processos cognitivos dos seres humanos e o comportamento dos usuários; na lingüística o interesse é o desenvolvimento de línguagens humanas e de máquina e relacionamento entre elas.
- HTML - acrônimo para *HyperText Markup Language*, é a linguagem utilizada para criação de

documentos a serem compartilhados na *World Wide Web*. A estrutura e o formato dos documentos são definidos por meio de *tags* e atributos, que são interpretados pelos navegadores (*browsers*).

- HTTP - acrônimo para *HyperText Transfer Protocol* o protocolo por baixo da *World Wide Web*. O HTTP define como as mensagens são formatadas e transmitidas, e as ações dos servidores e dos navegadores *web* em resposta aos comandos. Por exemplo, quando você informa um URL (*Uniform Resource Locator*) ao navegador, isto emite um comando HTTP ao servidor *web* para buscar e transmitir o página solicitada. O outro padrão principal que controla o funcionamento da *World Wide Web* é o HTML, que define como as páginas *web* são formatadas e exibidas. O HTTP é chamado um protocolo *stateless* porque cada comando é executado independentemente, sem nenhum conhecimento dos comandos que vieram antes dele. Esta é a principal razão da dificuldade de implementar sítios *Web* que reajam de forma inteligente às informações fornecidas pelos usuários. Esta deficiência do HTTP está sendo suprida por outras tecnologias, incluindo ActiveX, Java, Javascript e *cookies*.
- Instrumentação - é a implementação de procedimentos para coleta de informações, no software ou no hardware, para medir o estado de um componente de hardware, como o processador, os discos, a rede, ou de um componente de software, como o sistema operacional, *middleware*, ou aplicação. Um procedimento para coleta de informações é frequentemente chamado de sensor.
- Jini network technology - a tecnologia de rede de Jini é uma arquitetura de software aberto que possibilita o uso de Java em redes dinâmicas, para o desenvolvimento de sistemas distribuídos adaptáveis a mudanças. A tecnologia Jini pode ser usada para criar sistemas escaláveis e flexíveis, que são características necessárias em ambientes dinâmicos. A tecnologia Jini foi criada originalmente pela Sun Microsystems, e cedida pela Sun à *Jini Community* em 1999. Está disponível para uso livre e continua sendo aperfeiçoada pelos membros da *Jini Community* por meio do *Jini Community Decision Process*.
- JSP - acrônimo para *Java Server Page*, é uma tecnologia para servidores *Web*, extensão da tecnologia de *servlets* do Java que foi desenvolvida pela Sun. JSPs têm a capacidade de *scripting* dinâmico que trabalha em conjunto com código HTML, separando a lógica da página dos elementos estáticos - o desenho e a forma de exibição da página - para tornar o HTML mais funcional (por exemplo, o tratamento de requisições dinâmicas a bases de dados). O código JSP é traduzido em um *servlet* Java antes de ser executado, processa as solicitações HTTP e gera as respostas como todo o *servlet*. Entretanto, a tecnologia JSP permite uma maneira mais conveniente de codificar um *servlet*. A tradução ocorre na primeira vez que a aplicação é executada. Um tradutor de JSP é acionado pela existência da extensão *.jsp* em uma URL. Os *servlets* JSP são interoperáveis com *servlets*: é possível incluir a saída de um outro *servlet* em uma página gerada por um *servlet* JSP, ou enviar a saída do *servlet* JSP a um outro *servlet*. JSPs não são restritos a nenhuma plataforma ou servidor *Web* específico. Foi criado originalmente como uma alternativa a ASPs (*Active Server Pages*) da Microsoft. Recentemente, entretanto, a Microsoft propôs a tecnologia ASP.NET, parte da iniciativa de .NET, como alternativa ao JSP.
- JVM - acrônimo para *Java Virtual Machine*, uma máquina de computação abstrata, ou uma máquina virtual, é um ambiente de execução independente de plataforma que converte o *bytecode* Java na linguagem de máquina e o executa. A maioria das linguagens de programação compilam o código fonte dos programas diretamente no código de máquina, que é projetado

para funcionar na arquitetura específica de um microprocessador (Pentium, PowerPC, etc) ou de um sistema operacional, tal como Windows ou UNIX. Uma JVM - uma máquina dentro de outra máquina - imita um processador Java real , permitindo que o *bytecode* Java seja executado como ações ou chamadas ao sistema operacional em qualquer processador, independente do sistema operacional. Por exemplo, estabelecer o *socket* de uma conexão de uma estação de trabalho com uma máquina remota envolve uma chamada ao sistema operacional. Como os sistemas operacionais diferentes manipulam os *sockets* em formas diferentes, a JVM traduz o código de programação único (em Java) de modo que as duas máquinas, que podem estar em plataformas diferentes, possam conectar-se.

- LDAP - acrônimo para *Lightweight Directory Access Protocol*, é um conjunto de protocolos para acesso a diretórios de informação. LDAP é baseado nos padrões contidos no padrão X.500, mas é significativamente mais simples. Ao contrário do X.500, o LDAP suporta o TCP/IP, que é necessário para qualquer tipo de acesso à Internet. Por ser uma versão mais simples do X.500, o LDAP é às vezes chamado de X.500-lite. Embora ainda não seja extensamente implementado, o LDAP deve eventualmente possibilitar a quase toda aplicação que funciona em virtualmente qualquer plataforma, a obtenção de informações de diretórios, tal como endereços de email e chaves públicas. Por ser um protocolo aberto, as aplicações não necessitam preocupar-se sobre o tipo de servidor onde o diretório está hospedado.
- PHP - acrônimo para Hypertext Preprocessor, é uma linguagem de código aberto, em que é possível a inserção de código HTML, usada para a criação de páginas Web dinâmicas. Seu processamento ocorre no servidor da aplicação. Os comandos PHP, que têm sintaxe similar a linguagem C e Perl, são incluídos em documentos HTML por meio de *tags* especiais. Desta forma, o autor das páginas pode alternar entre HTML e PHP, evitando a necessidade de códigos pesados para obtenção dos resultados em HTML. Além disto, como o código PHP é processado no servidor, o cliente não pode visualizar o código PHP, portanto não é possível analisar o programa executado. Programas em PHP podem executar as mesmas funções que programas CGI (*Common Gateway Interface*) com a grande vantagem de serem compatíveis com muitos tipos de gerenciadores de bancos de dados. PHP pode também comunicar-se entre redes distintas utilizando IMAP (*Internet Message Access Protocol*), SNMP, NNTP (*Network News Transfer Protocol*), POP3 (*Post Office Protocol 3*), ou HTTP. O PHP foi criado em 1994 por Rasmus Lerdorf. A partir de 1997 seu desenvolvimento passou a contar com outros colaboradores e, entre eles, com Zeev Suraski e Andi Gutmans que reescreveram o *parser* para criar o PHP versão 3 (PHP3). Eles também fundaram a empresa Zend Technologies, em Israel, que atualmente é responsável pelo gerenciamento do desenvolvimento do PHP. Em maio de 2000 foi lançado o PHP4 e em julho de 2004 o PHP5.
- SNMP - acrônimo para *Simple Network Management Protocol*, é um conjunto de protocolos para gerenciamento de redes complexas. A primeira versão do SNMP foi desenvolvida no início dos anos 80. O protocolo SNMP trabalha com o envio de mensagens, chamadas *Protocol Data Unit*, para diferentes componentes da rede. Dispositivos que implementam SNMP, chamados agentes, armazenam dados sobre si mesmos em MIBs (*Management Information Bases*) e os enviam aos demais componentes que requisitarem informações SNMP.
- SOAP - acrônimo para *Simple Object Access Protocol*, é um protocolo simples, baseado em XML, usado para codificar as informações, contidas em mensagens de requisição e resposta de

serviços Web, antes do seu envio pela rede. As mensagens SOAP são independentes do sistema operacional e do protocolo em uso, e podem ser transportadas por meio de vários protocolos da Internet, entre eles SMTP (*Simple Mail Transfer Protocol*), MIME (*Multipurpose Internet Mail Extensions*) e HTTP.

- SQL - acrônimo para *Structured Query Language*, é uma linguagem padronizada para requisição de informações armazenadas em bases de dados. A versão original, chamada SEQUEL (*Structured English QUEry Language*) foi projetada pelo centro de pesquisas da IBM entre 1974 e 1975. A primeira implementação comercial da SQL em em sistema de banco de dados foi em 1979, pela *Oracle Corporation*. Historicamente, SQL tem sido a linguagem favorita para acesso a sistemas gerenciadores de bancos de dados, em minicomputadores e *mainframes*. Tem também aumentado sua utilização para acesso a bancos de dados em microcomputadores porque oferece suporte a bases de dados distribuídas, ou seja, bases de dados quem mantêm os dados espalhados em diversos computadores, possibilitando aos usuários de uma rede local o acesso simultâneo à mesma base de dados; Apesar da existência de diversos dialetos da linguagem SQL, este é, sem dúvidas, o exemplo mais próximo de uma linguagem padrão para consultas que existe. Em 1986, o ANSI (*American National Standards Institute*) aprovou uma versão rudimentar de SQL como o padrão oficial, mas desde então a maioria das versões de SQL incluíram muitas extensões a este padrão. SQL é adotado como padrão ANSI desde 1986 e como padrão ISO desde 1987. A revisão mais recente, denominada SQL:2006 define formas para utilização de SQL em conjunto com XML. As especificações do padrão SQL:2006 não estão disponíveis livremente e devem ser adquiridas do ANSI ou ISO.
- XML - acrônimo para *eXtensible Markup Language*, é uma linguagem especificada pelo W3C (*World Wide Web Consortium*). Trata-se de uma versão reduzida da linguagem SGML *Standard Generalized Markup Language*, projetada para tratamento de documentos Web. O uso da linguagem XML possibilita aos projetistas a criação de *tags* customizadas, permitindo a definição, transmissão, validação e interpretação de dados entre aplicações e entre organizações.