Grades Computacionais Baseadas em Modelos Econômicos

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Ricardo da Rosa e aprovada pela Banca Examinadora.

Campinas, 22 de fevereiro de 2010.

Maria Beatriz Felgar de Toledo

Profa. Dra. Maria Beatriz Felgar de Toledo
IC/UNICAMP - Universidade Estadual de Campinas (Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Crisllene Queiroz Custódio – CRB8 / 7966

Rosa, Ricardo da
R71g Grades computacionais baseadas em modelos econômicos / Ricardo da Rosa -- Campinas, [S.P. : s.n.], 2010.

Orientadora : Maria Beatriz Felgar de Toledo
Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Computação em grade (Sistema de computador). 2. Modelos econômicos. 3. Serviços Web. 4. Sistemas distribuídos. 5. Leilão. I. Toledo, Maria Beatriz Felgar de. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Grid computing based on economic models.

Palavras-chave em inglês (Keywords): 1. Grid computing (Computer system). 2. Economic models. 3. Web services. 4. Distributed systems. 5. Auctions.

Área de concentração: Sistemas de computação – Sistemas distribuídos

Titulação: Mestre em Ciência da Computação

Banca examinadora: Profa. Dra. Maria Beatriz Felgar de Toledo (IC-Unicamp)
Profa. Dra. Islene Calciolari Garcia (IC-Unicamp)
Prof. Dr. Marcelo Fantinato (EACH/USP)

Data da defesa: 22/02/2010

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 22 de fevereiro de 2010, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Marcelo Fantinato
EACH / USP

Profª. Drª. Islene Calciolari Garcia
IC / UNICAMP

Profª. Drª. Maria Beatriz Felgar de Toledo
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Grades Computacionais Baseadas em Modelos Econômicos

Ricardo da Rosa[1]

Fevereiro de 2010

Banca Examinadora:

- Profa. Dra. Maria Beatriz Felgar de Toledo
  IC/UNICAMP - Universidade Estadual de Campinas (Orientadora)
- Profa. Dra. Islene Calciolari Garcia
  IC/UNICAMP - Universidade Estadual de Campinas
- Prof. Dr. Marcelo Fantinato
  EACH/USP - Universidade de São Paulo
- Prof. Dr. Edmundo Roberto Mauro Madeira (Suplente)
  IC/UNICAMP - Universidade Estadual de Campinas
- Prof. Dr. Fabio Luciano Verdi (Suplente)
  UFSCAR - Universidade Federal de São Carlos

[1]Suporte financeiro de: Bolsa FUNCAMP 2007–2008 (Protocolo: 117086-07), Bolsa CAPES (processo 01-P-08503-2008) 2008–2009.

Resumo

Computação em grade é um paradigma que permite o compartilhamento de recursos heterogêneos, geograficamente distribuídos e sob administrações independentes. Esse compartilhamento deve ser realizado para otimizar a utilização de recursos e atender os requisitos de qualidade de serviço. Modelos econômicos podem ser aplicados para fornecer uma alocação justa desses recursos e incentivar a disponibilização de um maior número de recursos na grade.

Nesta dissertação, será discutida uma arquitetura de grade baseada em modelos econômicos, em especial, os vários modelos de leilões para permitir negociação entre um fornecedor e vários consumidores de recursos. Será realizada uma análise sobre as diversas modalidades de leilão para verificar o comportamento de consumidores e fornecedores de recursos em um ambiente de grade.

Abstract

Grid computing is a paradigm that allows the sharing of heterogeneous resources, geographically distributed and under independent administration. Sharing must be done to optimize the use of resources and meet quality of service requirements. Economic models can be applied to provide a fair allocation of these resources and to promote the entry of a greater number of resources into a grid.

In this dissertation, a grid architecture based on economic models will be discussed, in particular, several auction models to allow negotiation between a provider and many consumer of resources. Different types of auction models will be analyzed to verify the behavior of consumers and providers of resources in a grid environment.

Agradecimentos

Agradeço primeiramente a Deus, por possibilitar a realização de meus estudos e desenvolvimento desta dissertação, obtendo o grau de Mestre em Ciência da Computação pelo Instituto de Computação da Unicamp.

Agradeço aos meus pais, Celomar e Tânia, pelo apoio incondicional durante todos os anos durante a realização dos meus estudos de mestrado, principalmente pelo suporte dado nos momentos mais difíceis desta jornada. Agradeço também aos meus irmãos Renata e Rodrigo da Rosa pelo apoio sempre presente.

Agradeço a minha orientadora, Profa. Maria Beatriz Felgar de Toledo, por me aceitar no programa de pós-graduação e me ajudar durante meus estudos e desenvolvimento da dissertação.

Agradeço a todo o corpo docente e funcionários do Instituto de Computação da Unicamp, pelo conhecimento e experiências adquiridas.

Aos inúmeros amigos feitos em Campinas, sejam eles aqueles que estudaram comigo (Danilo, Leonel, Leonelo, e tantos outros), sejam eles aqueles que moraram comigo na república onde fizemos muitos churrascos (Luciano Granella, Fernando Ito, Diego Samir).

A todos aqueles que, de uma forma ou de outra, passaram pela minha vida durante essa jornada.

Muito obrigado.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A disseminação da Internet e as taxas de transmissão cada vez maiores permitiram a criação de um modelo computacional denominado computação em grade [33]. O desenvolvimento de aplicações que exigem um grande poder computacional incentivou o aparecimento desse modelo que provê o compartilhamento de recursos entre sistemas heterogêneos e geograficamente dispersos. Contudo, o compartilhamento de recursos exige políticas de alocação visando atender os requisitos de diversos consumidores sem prejudicar os fornecedores de recursos. O comportamento egoísta dos participantes (aqueles que desejam apenas consumir recursos sem a preocupação de fornecer os seus) pode prejudicar o crescimento da grade computacional.

Diversos sistemas como *Condor* [56] e *Legion* [35] foram desenvolvidos utilizando políticas de escalonamento baseadas em parâmetros de sistema como, por exemplo, *throughput*. Um segundo enfoque baseado em modelos econômicos se concentra em requisitos de qualidade de serviço (QoS) estabelecidos pelos próprios usuários. Entre os atributos de QoS estão a disponibilidade, tempo de resposta, segurança, interoperabilidade e custo.

Através de modelos econômicos, é possível regular a oferta e demanda de recursos pelo estabelecimento de preços, e, principalmente, incentivar a oferta de novos recursos para a grade.

Uma desvantagem de muitos sistemas está relacionada com aspectos como configurabilidade e extensibilidade. Esses sistemas são na maior parte estaticamente configuráveis e não adaptáveis em tempo de execução.

No sistema proposto, tentaremos obter maior grau de configurabilidade através de maior flexibilidade para a alteração de parâmetros de recursos na grade. Quanto à adaptação, o sistema oferece adaptação de parâmetros como preço de recursos de forma dinâmica.

A seguir são descritos os objetivos e a estrutura dessa dissertação.

1.1 Objetivos

Este trabalho tem por objetivo estender uma arquitetura para grades baseada em modelos econômicos [62] como forma de incentivo para aumentar o poder computacional da grade. A arquitetura proposta, que estende uma arquitetura proposta em [62], utiliza alguns dos serviços já oferecidos pelo Globus Tool Kit [48] que já é bastante aceito e utilizado na comunidade de grades. A extensão desenvolvida nessa dissertação inclui um novo elemento, o leiloeiro [1] que trata de vários modelos de leilão. Elementos da arquitetura original também foram estendidos para possibilitar a negociação de um recurso usando leilões.

A arquitetura apresenta elementos consumidores e fornecedores de recursos. Os consumidores realizarão buscas por recursos que atendam suas necessidades por um preço adequado, enquanto os fornecedores disponibilizarão seus recursos com o objetivo de obter lucro com este compartilhamento. O processo de negociação do recurso será realizado na forma de algum modelo de leilão que possibilita a negociação de um-para-muitos. Assim, um consumidor poderá decidir pagar um preço aceitável pelo recurso ou desistir de uma negociação, e o fornecedor poderá receber um valor justo para seu recurso baseando-se nas proporções de oferta e demanda de mercado, ou não realizar a venda e rever se o preço cobrado pelo seu recurso está condizente com os valores praticados pelos demais fornecedores.

Para a avaliação desses modelos econômicos, é apresentado um simulador do comportamento de consumidores e fornecedores de recursos.

As contribuições principais esperadas são o projeto de alguns modelos econômicos que podem ser utilizados conforme o comportamento da grade e de suas aplicações. A avaliação do comportamento dos modelos de leilão poderá indicar qual modelo tem um melhor desempenho, dependendo das condições e intenções dos participantes encontrados na grade computacional.

1.2 Organização da Dissertação

Essa dissertação está estruturada da seguinte maneira:

- O Capítulo 2 apresenta alguns conceitos básicos referentes a grades computacionais e modelos econômicos para negociação de recursos, os quais serão utilizados nesta dissertação;
- O Capítulo 3 apresenta alguns trabalhos relacionados na área em questão, mostrando

[1] Auctioneer

tanto as soluções que abordam modelos econômicos quanto tecnologias básicas para o desenvolvimento da arquitetura proposta;

- O Capítulo 4 apresenta a extensão da arquitetura proposta para grades computacionais, utilizando modelos econômicos de leilão para a negociação de recursos;
- O Capítulo 5 apresenta detalhes de implementação de um simulador do comportamento dos participantes em uma negociação de um recurso utilizando modelos econômicos de leilão, assim como a análise do comportamento desses participantes;
- O Capítulo 6 finaliza a dissertação apresentando as conclusões e contribuições desse trabalho, assim como os trabalhos futuros.

Capítulo 2

Fundamentos

Nas próximas seções são apresentados os fundamentos necessários para a compreensão desta dissertação.

2.1 Grades Computacionais

O termo grade foi concebido nos anos 1990 para denotar uma infra-estrutura para computação distribuída voltada para ciência avançada e engenharia[52]. Esse paradigma se distingue da computação distribuída atual por ser voltado para o compartilhamento de recursos em larga escala, aplicações inovadoras e, em alguns casos, orientação para processamento de alto desempenho.

Ao agruparmos recursos que estão distribuídos em uma forma de grade computacional, temos a possibilidade de compartilhar recursos entre os participantes dessa, surgindo então um sistema poderoso o suficiente para resoluções de problemas de forma colaborativa, evitando, quando possível, a aquisição de novos equipamentos. Como, ao redor do mundo, existem muitos computadores cujas capacidades não são exploradas totalmente, o agrupamento desses percentuais de ociosidade pode fornecer um poder computacional considerável para a resolução de muitos problemas.

Temos então que o principal problema relacionado a grades computacionais é o gerenciamento de recursos e a resolução de problemas em organizações virtuais dinâmicas e multi-institucionais. Esses recursos não são simplesmente troca de arquivos, mas também acesso direto a computadores, softwares, dados e outros recursos que sejam necessários para solucionar, de forma colaborativa, uma gama de problemas do mundo da indústria, da ciência e da engenharia. Assim, uma das questões centrais da computação em grade [52] é a interoperabilidade, pois o compartilhamento de recursos envolve participantes utilizando diversas plataformas, linguagens e ambientes de programação.

Os tipos de trabalhos no ambiente de computação em grade são os seguintes:

Computacional: Esse é o tipo de trabalho que é mais associado, pelas pessoas, a grades computacionais. Os recursos mais comuns fornecidos pelos processadores de máquinas na grade são processamento de tarefas. O processador pode variar em diversos aspectos: velocidade, arquitetura, entre outros fatores associados.

Podemos, então, dividir em três meios primários de explorar um recurso em uma grade [44]. A forma mais simples é utilizar a grade para executar uma aplicação existente em uma máquina participante da grade, com uma taxa de execução melhor do que a execução local da aplicação. A segunda forma consiste em dividir uma grande tarefa de processamento em pequenos problemas que podem ser distribuídos entre os vários participantes da grade a fim de minimizar o tempo gasto para realizar a tarefa e maximizar a utilização dos recursos computacionais disponíveis. Uma terceira forma é executar uma aplicação, que necessita ser executada várias vezes, em diversas máquinas diferentes da grade.

Virtualização de dados: Nesse caso, o recurso está associado ao armazenamento de dados. Através da agregação de dados, um sistema de arquivos inteligente que opere através da Internet pode criar um cenário virtual de dados a partir da utilização de espaços de discos não utilizados. Nesse caso, o sistema precisa manter referências para os locais onde os dados foram armazenados para permitir recuperação posterior. Em alguns sistemas especiais de banco de dados, é possível integrar vários bancos de dados individuais e arquivos, de uma forma ampla, compreensiva e acessível [44].

Tolerância a falhas: Proporciona redundância às instituições que precisam de alta disponibilidade de recursos. O sistema em uma grade pode ser relativamente barato e geograficamente disperso [44]. Então, se algum tipo de falha ocorrer em algum lugar, as outras partes da grade não serão diretamente afetadas. Um gerenciador da grade poderá automaticamente submeter novamente os trabalhos (*jobs*) para outro participante da grade quando uma falha é detectada. Em sistemas críticos, situações de tempo-real, múltiplas cópias de trabalhos importantes podem ser executadas em diferentes participantes através da grade.

A seguir, apresentamos o conceito de organizações virtuais e arquiteturas para grades computacionais.

2.1.1 Organizações Virtuais

Um dos principais problemas em grades computacionais é gerenciar os recursos disponibilizados nelas [52]. Recursos que podem ser compartilhados em uma grade são os seguintes: arquivos, acesso direto a computadores ou softwares, dados, licenças, entre outros. Esse

compartilhamento de recursos necessita ser controlado, com provedores e consumidores desses claramente definidos, cuidando do que é compartilhado, quem tem permissão para compartilhar, além de condições para haver tal compartilhamento. Ou seja, é necessário que regras sejam bem definidas para que indivíduos e/ou instituições possam compartilhar seus recursos. O conjunto de indivíduos e/ou instituições que compartilham recursos de acordo com certas regras forma o que é chamado de *Organização Virtual* [52].

Organizações Virtuais podem variar muito no que diz respeito a seu propósito, escopo, tamanho, estrutura, entre outros aspectos [52]. O proprietário de um recurso é quem define se esse está disponível e quais as restrições para seu uso, podendo limitar o uso por parte dos participantes de uma Organização Virtual da qual ele participa. Além disso, participantes de uma Organização Virtual podem decidir consumir recursos de algum consumidor, conforme requisitos como, por exemplo, um certificado de segurança. Implementar cada restrição requer mecanismos para expressar políticas, seja para estabelecer uma identificação dos consumidores ou dos recursos (autenticação), ou para determinar quando uma operação é consistente com os relacionamentos de compartilhamento (autorização).

Vemos então que as Organizações Virtuais possibilitam que grupos de organizações e/ou indivíduos compartilhem recursos de modo controlado [52], o que é uma característica interessante a uma arquitetura de grades computacionais.

2.1.2 Arquitetura de Grade Computacional

Uma arquitetura de Grade Computacional é responsável por definir mecanismos para o compartilhamento de recursos, identificando os componentes fundamentais, especificando o propósito e as funções desses componentes, e indicando como esses componentes interagem uns com os outros [52], além de cobrir a segurança no compartilhamento desses recursos.

Uma década dedicada a pesquisa e desenvolvimento e experimentações produziu um consenso sobre os requisitos e a arquitetura de Grade Computacional [47]. Protocolos padrão, utilizados para comunicação entre consumidores e fornecedores de recursos, emergiram como um meio importante e essencial para alcançar a interoperabilidade de que um sistema em uma grade computacional depende, assim como APIs (*Application Programming Interfaces*) padrão que facilitam a construção e o reuso de componentes de uma grade.

Os protocolos e APIs podem ser categorizadas em uma estrutura de camadas de acordo com o papel que cada componente possui na grade [47]. A especificação dessas várias camadas da arquitetura da Grade segue o princípio de um modelo em ampulheta, em que as camadas mais internas possuem um número menor de elementos, como pode ser visto

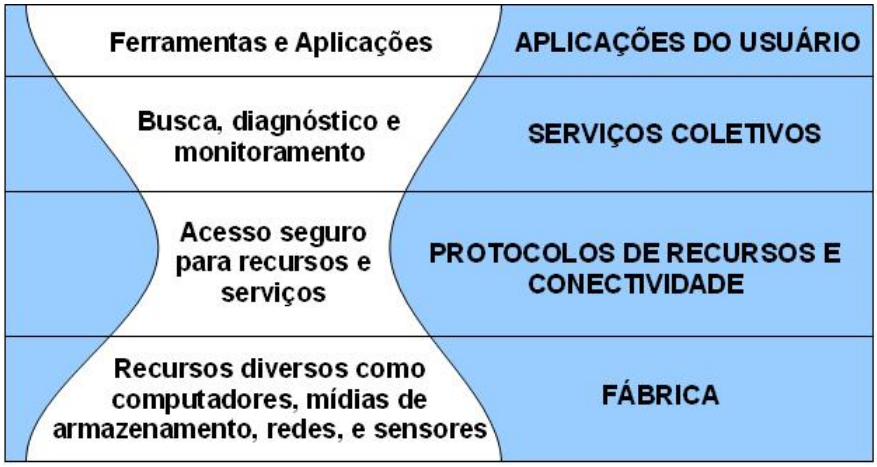

Figura 2.1: Modelo em Ampulheta da Aquitetura de Grades Computacionais [47].

na Figura 2.1 da página 7.

No nível mais baixo da arquitetura, encontramos a **Fábrica** (*Fabric*) [47], com os dispositivos ou recursos físicos que os usuários da Grade desejam compartilhar e acessar, incluindo computadores, sistemas de armazenamento, catálogos, redes, entre outros, ou ainda algum recurso lógico, como sistema de arquivos e licenças de software. Os componentes dessa camada oferecem os protocolos e mecanismos que implementam as operações fornecidas pelos recursos. As implementações desses mecanismos podem ser fornecidas tanto pelo fabricante do recurso quanto pelo *Middleware* da Grade Computacional.

Acima da Fábrica, encontramos a camada de **Protocolos de Recursos e Conectividade** (*Resource and Connectivity Protocols*)[47], que define o conjunto básico de protocolos para atender os requisitos de comunicação e autenticação para operações da Grade Computacional. Para fornecer segurança, mecanismos de autenticação e criptografia para verificação de identidades de usuários são aplicados aos protocolos de comunicação. Em Organizações Virtuais, as soluções de autenticação devem fornecer as seguintes características ([52]*apud*[27]):

- **Autenticação única no sistema**: O usuário deve fazer seu "*log on*"apenas uma vez e ter acesso a múltiplos recursos da Grade Computacional definidos na camada de Fábrica, sem a necessidade de nova autenticação;

- **Delegação**: O usuário deve conseguir atribuir a algum outro sistema, seus direitos de maneira que esse sistema consiga utilizar recursos em seu nome. Esse sistema também deve ser capaz de atribuir seus direitos a um outro, desde que respeite restrições de direitos;

- **Integração com várias soluções locais de segurança**: Diversos sistemas podem utilizar soluções locais de segurança, sendo então necessário que existam mecanismos na Grade que possibilitem a interação com eles;

- **Relacionamentos confiáveis baseados no usuário**: Quando um usuário utiliza recursos de múltiplos provedores juntos, o sistema de segurança não deve requerer que cada um dos provedores de recursos interajam com outros para configurar a segurança do ambiente. Por exemplo, suponhamos um usuário que tem o direito de usar dois recursos A e B; esse usuário deve ser capaz de utilizar ambos os recursos sem que haja interação entre os gerenciadores desses recursos.

Esta camada (Protocolo de Recursos e Conectividade) utiliza os mecanismos da camada Fábrica para realizar operações sobre recursos compartilhados com segurança, como monitoração por exemplo. Os protocolos dessa camada que são voltados para recursos, estão inteiramente preocupados com recursos individuais, ignorando ocorrências de eventos no estado global e ações atômicas em coleções distribuídas de recursos da Grade Computacional. Temos então, duas principais classes de protocolos na camada de Recursos e Connectividade [52]:

- **Protocolos de Informação**: são usados para obter informações sobre a estrutura e estado de um recurso;

- **Protocolos de Gerenciamento**: são usados para negociar acesso ao recurso compartilhado, especificando requisitos do recurso, como qualidade de serviço, operações a serem realizadas, políticas de uso, entre outros.

A camada **Serviços Coletivos** (*Collective Services*) contém protocolos, serviços e APIs que implementam a interação sobre coleções de recursos [47]. Exemplos de serviços coletivos incluem [52]:

- **Serviço de diretórios**: permitem aos participantes de uma Organização Virtual descobrir a existência ou propriedades de recursos em Organizações Virtuais. Um serviço de diretório pode permitir que seus usuários consultem recursos por nome e/ou por atributos como tipo, disponibilidade, entre outros;

- **Co-alocação, escalonamento e serviços de *mediação***[1]: permitem aos participantes da Organização Virtual requisitar a alocação de recursos para um propósito específico e fazer o escalonamento de tarefas em recursos apropriados;

[1] **Do inglês *brokering***

- **Monitoração e serviços de diagnósticos**: apóiam a monitoração de recursos de Organizações Virtuais por falhas, ataques adversos (detecção de intrusos), sobrecarga, entre outros.

Os componentes da camada Coletivo podem ser desenvolvidos para requisitos de uma comunidade específica de usuários, Organização Virtual ou domínio de aplicação. Por outro lado, componentes da camada Coletivos podem ter mais de um propósito geral, como por exemplo, um serviço de diretórios projetado para possibilitar a descoberta de Organizações Virtuais [52]. Desse modo, se o alvo é a comunidade de usuários, é importante que os protocolos e APIs de componentes sejam baseados em padrões.

A camada final na arquitetura de Grades compreende as **Aplicações do Usuário**, que operam utilizando os recursos da Grade Computacional, podendo também operar com uma Organização Virtual [52]. Ou seja, são aplicações são construídas baseadas em chamadas a componentes de qualquer outra camada.

Assim, uma aplicação que deseja utilizar um recurso de uma Grade (recurso na camada Fábrica) deveria, por exemplo [47]:

1. **obter** credenciais de autenticação necessárias (protocolos da camada de Conectividade);

2. **consultar** algum sistema de informação e réplica de catálogos para determinar a disponibilidade de computadores, sistemas de armazenamento, redes, e a localização de arquivos de entrada requeridos (serviços na camada Coletivos);

3. **submeter** requisições a computadores apropriados, sistemas de armazenamento e redes para iniciar a computação (protocolos de Recuros);

4. **monitorar** o progresso de várias computações e transferência de dados, notificando o usuário quando as tarefas estiverem completas e detectando e respondendo a condições de falhas (protocolos de Recursos).

2.2 Arquitetura Orientada a Serviços

Sistemas distribuídos são compostos por conjuntos de elementos (*hardware e software*) que se comunicam utilizando troca de mensagens [39]. Como a comunicação entre homem e máquina bastante difundida com a proliferação da *Web*, a comunicação máquina-máquina (ou aplicação-aplicação) vem ganhando um espaço considerável na área da computação. Temos como exemplos, tecnologias que implementam mecanismos que possibilitam a

execução de alguma funcionalidade remota, como *Sun RPC*[39] e *Java RMI*[55]. Com essas tecnologias, foi possível que aplicações invocassem outras aplicações remotas passando parâmetros de processamento e recebendo respostas como resultado da invocação.

Porém, algumas tecnologias são dependentes de plataformas e linguagens de programação. A Arquitetura Orientada a Serviços (*Service Oriented Architecture* - SOA) é uma abordagem arquitetural em que uma aplicação é composta de componentes de *software* independentes e distribuídos chamados *serviços eletrônicos* [59].

O conceito chave de SOA é que as funcionalidades implementadas por um serviço são expostas por meio de declarações de interfaces baseadas em padrões. Os detalhes de implementação ficam escondidos dos usuários do serviço, os quais apenas podem invocar serviços baseados em operações expostas nas interfaces [45]. Temos, então, uma estrutura independente de plataforma e linguagem de programação.

A seguir, apresentamos conceitos e padrões da tecnologia de serviços *Web*.

2.2.1 Serviços *Web*

Um Serviço *Web* [23][46] é um componente de software projetado para fornecer formas de interação entre aplicações através de uma rede de computadores. Uma aplicação pode ser implementada na forma de um serviço e disponibilizada para ser invocada remotamente.

O termo Serviço *Web* é descrito como um importante paradigma de computação distribuída que emerge diferente de outras abordagens como DCE [4], CORBA [1] e Java RMI [55]. Serviços *Web* são baseados em padrões da Internet, direcionados para computação distribuída heterogênea [51]. Serviços *Web* definem padrões para a descrição de componentes de software para serem acessados, descobertos e invocados em uma rede.

Como a interoperabilidade é uma questão de grande importância no âmbito de um sistema distribuído, torna-se necessário utilizar diversas especificações e protocolos. Podemos fazer, de maneira abstrata, uma análise de um modelo utilizado pelos serviços *Web* em uma estrutura de camadas, como pode ser visto na Figura 2.2 da página 11.

Na camada de ***Descoberta*** (Figura 2.2), encontramos protocolos de descoberta de serviços, como o UDDI (*Universal Description, Discovery and Integration*)[23], o qual fornece mecanismos para armazenamento de descrições e descoberta de serviços *Web*. Quando o UDDI é consultado, podemos encontrar serviços e as informações necessárias para sua utilização. O UDDI fornece aos usuários um meio unificado e sistemático para encontrar provedores de serviços através de um diretório de serviços [40], onde as informações sobre tais serviços podem ser pesquisadas em:

- **Páginas Brancas:** informações como organizações e detalhes para contato;
- **Páginas Amarelas:** fornecem uma categorização baseada no tipo de negócio e serviço;

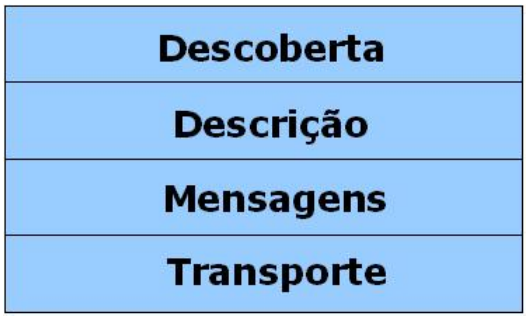

Figura 2.2: Uma Visão Abstrata do Modelo de Camadas de Serviços *Web*.

- **Páginas Verdes:** são informações que incluem dados técnicos sobre os serviços.

Tendo descoberto o serviço, é necessário agora que uma aplicação consiga interagir com o serviço *Web*, obtendo informações de suas interfaces, de maneira que certos detalhes importantes para a interação sejam conhecidos (quantidade e tipos dos parâmetros, retornos, entre outros). O WSDL (*Web Services Description Language*) [23], na camada ***Descrição***, é baseado em XML (*eXtensible Markup Language*) [53] (desenvolvido e mantido pela W3C - *World Wide Web Consortium*), que descreve as interfaces e requisitos de uso de um serviço *Web* de modo que uma aplicação possa trocar informações com esse serviço.

Obtendo o conhecimento sobre as interfaces e formas de comunicação com o serviço *Web*, é necessário agora interagir com esse serviço. O protocolo SOAP (*Simple Object Access Protocol*)[23] da camada ***Mensagens***, como pode ser visto na Figura 2.2, é utilizado para a troca de mensagens e invocações remotas dos serviços. Essa troca de mensagem deve ocorrer mesmo quando exista diferença de plataformas ou linguagens de programação entre as aplicações. Para existir esta interoperabilidade, o protocolo SOAP faz uso de XML para definir o formato de uma mensagem trocada entre duas aplicações e utiliza outros protocolos, da camada de ***Transporte*** (Figura 2.2) já existentes, para o envio da mensagem, como HTTP (*Hypertext Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*), entre outros [40].

A mensagem SOAP é organizada então, em um arquivo XML, com um elemento raiz chamado envelope, contendo uma área para cabeçalho e outra para o corpo da mensagem, como pode ser visto na Figura 2.3 da página 12.

O elemento *Header*, opcional, pode ser utilizado para funções específicas da aplicação durante o transporte da mensagem. Essas funções que podem ser realizadas por intermediários no processo de comunicação, podem incluir autenticação, controles de transação, pagamentos, ou qualquer outra funcionalidade [64].

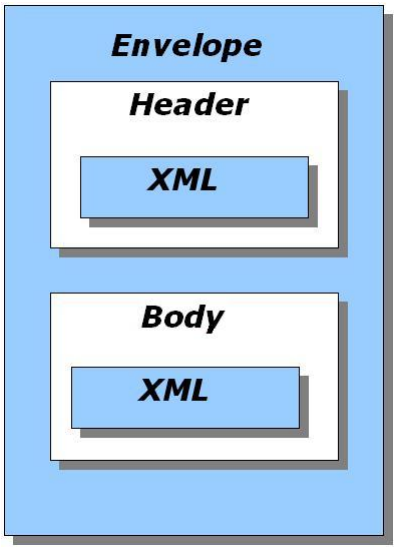

Figura 2.3: Mensagem SOAP.

O elemento *Body* é a parte do envelope SOAP que contém a mensagem propriamente dita, direcionada ao último ponto final da mensagem [64][40]. O corpo da mensagem pode ser implementada em duas formas[26]:

- ***Document-Style:*** o elemento *Body* transporta uma mensagem XML que está relacionada a um documento que está sendo trocado entre as aplicações. Exemplo: pedido de compra contendo todos os itens, quantidade e preço, em que o destino processa este documento;
- ***RPC-Style:*** o elemento *Body* transporta um documento que carrega informações sobre uma chamada de um procedimento remoto, contendo o procedimento a ser invocado e os valores dos parâmetros.

Definidos os principais componentes da arquitetura de serviços *Web*, a Figura 2.4 da página 13 exibe a estrutura de utilização de um serviço. Inicialmente, o serviço é desenvolvido por um fornecedor e publicado em um ***Repositório UDDI***. Essa publicação utiliza mensagens ***SOAP*** para enviar a descrição do serviço *Web* definido em ***WSDL***. Um consumidor pode, então, buscar algum serviço de seu desejo no repositório UDDI, novamente utilizando uma mensagem SOAP. O repositório UDDI envia as informações sobre o serviço ao cliente e este pode mapear as interfaces do serviço pelas descrições contidas no WSDL. Finalmente, o cliente pode interagir com o serviço *Web* por meio do protocolo SOAP.

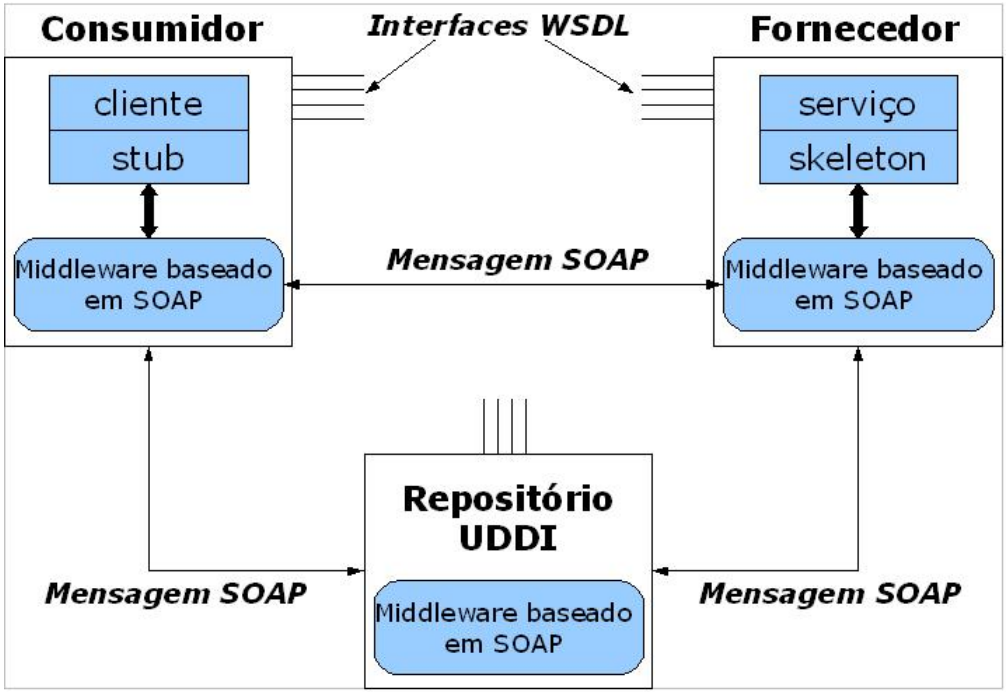

Figura 2.4: Estrutura de Utilização de um Serviço *Web*.

A Figura 2.4 mostra no consumidor os módulos *cliente* e *stub*, onde o primeiro é a aplicação do consumidor que deseja utilizar o serviço remoto, e o segundo é o módulo responsável pela abstração da implementação necessária para invocação do serviço remoto, o qual é gerado pelo compilador através do processamento da interface do serviço *Web* descrito em WSDL. Já no fornecedor, temos os módulos *serviço* e *skeleton*, em que o primeiro é a aplicação que será executada por uma chamada, e o segundo, com um papel semelhante ao *stub*, será responsável pela abstração da implementação necessária para tratamento da invocação, recebimento de parâmetros, entre outras funções. O módulo ***Middleware baseado em SOAP*** é encontrado tanto no consumidor, fornecedor e repositório UDDI, onde é o responsável pelo tratamento das mensagens SOAP. As mensagens SOAP fazem uso de algum protocolo de transporte, como o HTTP por exemplo.

2.3 Grades Computacionais e Economia

Grades Computacionais são um paradigma para resolver problemas de larga escala na ciência, engenharia e comércio [32][33], provendo o compartilhamento de recursos heterogêneos que estejam distribuídos, além de possibilitar a criação de Organizações Virtuais para o compartilhamento e a agregação de diversos recursos. Elas possuem um potencial

maior do que um sistema distribuído típico que fornece serviços de provedores para consumidores e possibilitam a computação sob demanda e compartilhamento dos recursos [24]. Uma Grade baseada em economia deve oferecer serviços atingindo diversas empresas. Em um ambiente destes, empresas podem tomar decisões de forma independente, gerando uma economia real.

Os fornecedores (*proprietários de recursos*) e consumidores (*consumidores de recursos*) possuem diferentes objetivos, estratégias e padrões de comportamento. Ao colocar valores em recursos, os participantes de uma grade computacional (fornecedores ou consumidores) podem sentir-se motivados a compartilhar/utilizar com a intenção de ganhar dinheiro ou créditos para utilzar outros serviços da grade [34]. Para que isso ocorra, é necessário utilizar algum modelo econômico para alocação e distribuição dos recursos. A abordagem de economia provê uma base justa de gerenciamento da descentralização e heterogeneidade que está presente em uma economia real [33].

A seguir é apresentado alguns modelos econômicos, além de técnicas de contabilização e pagamento de recursos.

2.3.1 Modelos Econômicos

Modelos econômicos [43] oferecem um conjunto de ferramentas para limitar a complexidade devida à descentralização do controle de recursos. Além disso, oferecem um conjunto de modelos matemáticos que podem produzir novas alternativas aos problemas de compartilhamento de recursos.

Em uma economia, modelos econômicos consistem de agentes que de forma egoísta querem alcançar seus objetivos. Existem dois tipos de agentes: *fornecedores e consumidores*. O consumidor procura otimizar seu critério de desempenho individual obtendo os recursos necessários sem se preocupar com o desempenho do sistema como um todo. O fornecedor provê seus recursos individuais para consumidores. O principal objetivo do fornecedor é oferecer recursos visando sua satisfação individual. Alguns exemplos de consumidores são aplicações como transações, tarefas computacionais, aplicações de transferência de arquivos, teleconferências multimídia e distribuição de notícias. Computadores são fornecedores primários e controlam recursos locais como tempo de UCP, memória, discos, largura de banda e recursos de comunicação. Fornecedores também incluem servidores que oferecem serviços lógicos como servidores de arquivos, servidores de páginas amarelas, servidores de nomes, servidores de notícias e servidores *Web*.

Em uma grade computacional baseada em modelos econômicos, as decisões de escalonamento de recursos são feitas dinamicamentes em tempo de execução: enquanto o modelo de custo convencional foca em custos de *hardware* e *software* para executar uma aplicação, um modelo econômico tem como prioridade atender os usuários finais de

serviços, baseados em valores de requisitos definidos por estes [33].

Nas grades computacionais com uma grande variedade de recursos compartilhados e usuários interessados nestes, a alocação de recursos pode ser uma tarefa complexa, necessitando de algum mecanismo que torne possível analisar e tomar decisões baseadas em políticas definidas [33]. Algumas das políticas que podem ser adotadas são:

- **Políticas centradas no sistema**: a preocupação está focada no melhor uso do sistema como um todo, com base em aspecto de funcionamento como *throughput*, taxa de utilização de memória, entre outros;
- **Políticas centradas no usuário**: a preocupação está em atender requisitos de processamento do usuário, como qualidade de serviço (QoS[2]);

Quando políticas centradas no usuário são utilizadas, é necessário um sistema de recompensas e penalidades aos participantes [33]. Em uma requisição de recurso para execução de uma tarefa, o usuário pode definir diversos critérios para execução ([31]). Em alguns casos, o tempo de execução de uma tarefa é relacionado com o poder de processamento utilizado na sua solução. Quanto menor o tempo desejado para obter a solução, maior será a necessidade de poder computacional. Caso os valores de processamento não sejam corretamente definidos, pode haver desperdícios de processamento ou sobrecarga do sistema.

Muitos modelos econômicos introduzem créditos (ou moeda) e uma técnica para coordenar o comportamento egoísta dos participantes (fornecedores ou consumidores). Cada consumidor recebe uma doação de crédito que ele usa para comprar os recursos requeridos. Cada produtor possui um conjunto de recursos e cobra dos consumidores pelo uso desses recursos. O preço que um produtor cobra por um recurso é determinado por algum sistema de geração de preços. O sistema de preços assegura que uma alocação factível de recursos seja obtida.

Para a determinação de preços, os fornecedores de recursos de uma Grade Computacional baseada em modelos econômicos devem possuir meios de atribuir e controlar valores para seus recursos [33]. Diversos parâmetros podem ser utilizados na definição de preços para utilização de recursos. Exemplos de esquemas de definição de preços são:

- **Preço fixado** [31]: um preço é pré-determinado para a utilização de um recurso;
- **Oferta e demanda:** preços são estabelecidos de acordo com a situação atual do mercado. Se a utilização do recurso cai, o seu preço tende a diminuir, enquanto se seu uso cresce, o seu preço tende a subir;

[2]Do inglês *Quality of Service*

- **Preço por períodos:** identificam-se picos de utilização do recurso. Em momentos de maior utilização o preço é maior, enquanto em períodos de menor utilização, o preço é menor. Podemos fazer uma analogia ao sistema de tributação da telefonia, em que fins de semanas e feriados os custos de ligação são mais baixos.

Dado que um consumidor de serviços definiu seus requisitos e aspectos de QoS para um recurso desejado, ele pode iniciar uma busca de um serviço que atenda suas necessidades. A figura de um mediador pode ser utilizada para a negociação com o fornecedor de um recurso. Este mediador pode buscar a otimização de algum critério de escalonamento ou de custo [33]:

- **Otimização de escalonamento:** procura o melhor prazo de execução possível sem que o valor a ser cobrado ultrapasse o estipulado pelo consumidor;
- **Otimização de custos:** procura um recurso com o menor valor possível sem que o prazo de execução seja ultrapassado.

Tanto o fornecedor quanto o consumidor pode definir porcentagens para relaxamento de critérios durante uma negociação como, por exemplo, flexibilizar prazo de execução, aumentar do valor a ser pago pela utilização do recurso, entre outros. Esse relaxamento possibilita que seja encontrado um ponto de interesse de ambas as partes para finalizar positivamente a negociação ou então cancelar o processo de negociação.

Alguns dos modelos econômicos para negociação de recursos em sistemas computacionais e estratégias para estabelecimento de preços são os seguintes [30]:

- *Modelo de mercado de produto*[3]: Os fornecedores, de forma competitiva, estabelecem preços para seus recursos e anunciam seus serviços. Os consumidores podem escolher o serviço a ser adquirido através de uma análise de custo-benefício;
- *Modelo de preço afixado*[4]: Esse modelo é semelhante ao anterior, no entanto fornecedores geram promoções com a intenção de atrair novos consumidores ou motivar seus atuais consumidores;
- *Modelo de negociação*[5]: Fornecedores e consumidores negociam por prazos e preços que maximizem seus objetivos, ou então a negociação é abortada. Os participantes possuem seus próprios objetivos definindo as metas da negociação. Assim, o fornecedor procura alocar seu recurso com o maior valor possível, e o consumidor busca um recurso que atenda suas necessidades com o menor valor possível;

[3]Tradução de Commodity Market Model
[4]Tradução de Posted Price Model
[5]Tradução de Bargaining Model

- *Modelo de oferta/contrato*[6]: O consumidor solicita ofertas fechadas de vários fornecedores e seleciona aquele que possui menor preço pelo serviço e que se encaixe com seu prazo e orçamento;
- *Modelo de compartilhamento proporcional à oferta*[7]: Nesse modelo, a quantidade de recursos alocados ao consumidor dependerá do valor de sua oferta;
- *Modelo de cooperativa*[8]: Nesse modelo, um conjunto de participantes que sejam consumidores e fornecedores de recursos forma uma cooperativa na qual quem contribui oferecendo um recurso pode utilizar recursos alheios quando necessário;
- *Modelo de leilão*[9]: Os fornecedores solicitam ofertas de vários consumidores, que são livres para gerar suas ofertas de acordo com suas necessidades e possibilidades. O leilão acaba quando não chegarem novas ofertas após um determinado tempo.

O Modelo de Leilão, que é o modelo focado neste trabalho, possui variações na forma de execução. Seu modelo suporta a negociação de "*um-para-muitos*", entre um fornecedor e muitos consumidores [32], e redução da negociação a um único valor de preço. Geralmente, leilões são constituídos por três entidades: o *seller*[10], os *bidders*[11] e o *auctioneer*[12], o qual é responsável por todo o gerenciamento do leilão [33]. Um leilão pode ser conduzido de forma aberta (onde todos os participantes conhecem todos os lances gerados) ou fechada (os valores de lances não são divulgados), dependendo de como são permitidas as ofertas. Um consumidor pode atualizar um lance e um fornecedor pode atualizar o preço de venda oferecido [32]. Dependendo dos parâmetros, leilões podem ser classificados em [28]:

- **Leilão Inglês**[13]: o *auctioneer* inicia o processo com um valor inicial para o bem (ou recurso) a ser leiloado, e todos os compradores (*bidders*) são livres para aumentarem suas ofertas em relação aos outros concorrentes, buscando oferecer o melhor valor ao *auctioneer*. O processo termina quando nenhum outro *bidder* lançar ofertas;
- **Primeiro Preço**[14]: todos os *bidders* enviam uma oferta ao *auctioneer* sem terem conhecimendo das ofertas dos outros *bidders*. Quando o *auctioneer* recebe todas as ofertas, ele verifica qual possui o melhor valor e efetua a venda para quem fez tal oferta;

[6]Tradução de Tendering/Contract Model
[7]Tradução de Bid-based Proportional Resource Sharing Model
[8]Tradução de Community/Coalision/Bartering Model
[9]Tradução de Auction Model
[10]Indivíduo que está colocando um recurso seu para leilão
[11]Indivíduos com intenção de aquisição de um recurso em um leilão
[12]Leiloeiro
[13]Tradução de English Auction
[14]Tradução de First-price Sealed-bid Auction

- **Segundo Preço**[15]: todos os *bidders* enviam ofertas ao *auctioneer* sem conhecimento das ofertas dos outros *bidders*. Quando o *auctioneer* recebe todas as ofertas, ele efetua a venda para o *bidder* que fornecer o maior valor ofertado, porém com o preço do segundo maior valor ofertado;

- **Leilão Holandês**[16]: neste modelo, o *auctioneer* inicia o leilão com um valor alto para o recurso em questão. O *auctioneer* vai então diminuindo o valor requerido até que algum *bidder* faça um lance. O primeiro *bidder* que fizer o lance adquire o recurso (ou o direito de utilizar). Uma vantagem desse modelo é que ele força a oferta-demanda. Enquanto não houver demanda suficiente, o preço é reduzido, e por outro lado, quando há muita demanda e pouca oferta, lances são dados mais cedo no leilão;

2.3.2 Contabilizando e Realizando Pagamentos

Ao utilizamos uma Grade Computacional baseada em modelos econômicos, é necessário obter mecanismos para efetuar os pagamentos da utilização dos recursos envolvidos em um processo de venda. Estes mecanismos podem fornecer um controle sobre a capacidade de pagamento para uso de recursos de cada um dos participantes, assim como transferências de créditos na utilização de um recurso compartilhado.

As formas de pagamentos que podem ser aplicadas são [30]:

- Pré-pagas: o consumidor de um recurso realiza o pagamento para utilização deste antes de começar a utilizá-lo;

- Pós-pagas: o consumidor de um recurso realiza o pagamento para utilização deste após usá-lo.

Para utilizar mecanismos de controle de créditos e pagamentos, importantes características devem ser observadas [58]:

- **Segurança:** diz respeito a transferências de créditos entre participantes que devem trafegar de forma segura pela Internet.

- **Confiabilidade:** significa que mecanismos de pagamentos devem estar sempre disponíveis, apesar da ocorrência de falhas.

- **Escalabilidade:** siginifica que novos participantes não devem degradar o desempenho do sistema.

[15]Tradução de Second-price Sealed-bid Auction
[16]Tradução de Dutch Auction

- **Aceitação:** todos os participantes da grade devem aceitar o mesmo mecanismo de pagamento. Assim, créditos podem ser utilizados para o pagamento de qualquer recurso.
- **Base de consumidores:** a quantidade potencial de consumidores pode influir na aceitação de um mecanismo de pagamento por parte de um determinado fornecedor de um recurso.
- **Flexibilidade:** significa que o sistema deve aceitar diversas formas de pagamento.
- **Eficiência:** o funcionamento do mecanismo de pagamento não deve sofrer degradação independentemente da quantidade ou valor de pagamentos realizados.
- **Facilidade de uso:** o mecanismo de pagamento deve ser de fácil uso por parte do usuário, possibilitando que os usuários definam valores máximos que desejam gastar, e solicitando autorizações de pagamentos para valores que ultrapassem esse valor máximo.

Com os requisitos para mecanismos de pagamentos definidos [58], pode-se agrupar estes nas seguintes classes: moeda eletrônica, crédito e débito, e apresentação segura de números de cartão de crédito. O pagamento utilizando *moeda eletrônica* ocorre quando um usuário adquire um certificado eletrônico de moeda que representa um determinado valor. Ao efetuar o pagamento, o usuário apresenta tal certificado para o credor fazer o depósito do valor em sua própria conta. No pagamento na forma *crédito e débito*, cada usuário deve possuir uma conta com saldo positivo em um servidor. Assim, contas de consumidores e fornecedores são atualizadas conforme as transações são realizadas. A *apresentação segura de cartão de crédito* ocorre quando o usuário deseja efetuar o pagamento através do uso de um número de cartão de crédito. Técnicas de criptografia assimétrica devem ser utilizadas para que somente pessoas autorizadas tenham acesso ao número do cartão.

2.4 Conclusões do Capítulo

Grades computacionais possibilitam o compartilhamento de recursos heterogêneos e geograficamente dispersos, onde os participantes colaboram na resolução de problemas computacionais.

Uma grade computacional voltada para o compartilhamento de serviços, com controle de qualidade em requisitos do usuário, pode aplicar modelos econômicos para o escalonamento justo de recursos, evitando o comportamento egoísta dos participantes.

O próximo capítulo apresenta os trabalhos relacionados com esta dissertação.

Capítulo 3

Trabalhos Relacionados

Nas próximas seções são apresentados os trabalhos relacionados a grades computacionais.

3.1 OGSA e WSRF

Em uma grade computacional existem diversos participantes, os quais desenvolvem aplicações para solucionar diversos problemas. Para um extensivo uso destas aplicações, todos os participantes devem possuir a capacidade de interagir com diversas soluções sem uma sobrecarga de conhecimento sobre aspectos particulares de cada uma destas. Desse modo, é necessário a adoção de padrões para o desenvolvimento e disponibilização de serviços em uma Grade Computacional.

A *Open Grid Services Architecture* (OGSA) [21], desenvolvida pela *Global Grid Forum* (http://www.ggf.org), busca definir uma arquitetura aberta, comum e padrão para aplicações baseadas em Grades Computacionais [61]. O principal objetivo da OGSA é padronizar praticamente todos os serviços com finalidades comuns em uma aplicação de grade (serviços gerenciadores de tarefas e recursos, serviços de segurança, entre outros) pela especificação de um conjunto padrão de interfaces para estes serviços.

A OGSA tem como premissa que qualquer recurso compartilhado seja representado por um serviço de grade [51]. Serviços de grade estendem conceitos de serviços *Web* [44], fazendo uso de padrões bem definidos de interfaces para capitalizar diversas propriedades destes, como serviços de descrição e descoberta (fazendo uso de WSDL para obter autodescrição e protocolos interoperáveis), gerenciamento do tempo de vida, notificação, entre outros. Como a arquitetura é aberta, temos extensiva comunicação, neutralidade de fornecedores, além do comprometimento com o uso de padrões aceitos pela comunidade.

Serviços da OGSA podem ser criados e destruídos dinamicamente, podendo existir diversas instâncias de um mesmo serviço sendo executadas ao mesmo tempo. Com isso, é necessário distinguir uma instância de um serviço de uma outra instância do mesmo

serviço. Assim, toda instância de um serviço de grade é identificada com um único nome global, o *Grid Service Handler* (GSH), o qual não varia durante o tempo de vida do serviço. Um Serviço de Grade pode ser atualizado durante seu tempo de vida, podendo, por exemplo, suportar uma nova versão de protocolo ou protocolos alternativos [51]. O GSH não possui informações sobre protocolos ou outras informações específicas sobre uma instância de um serviço. Tais informações são encapsuladas em uma abstração chamada *Grid Service Reference* (GSR), o qual pode ser atualizado durante o tempo de vida da instância de Serviço de Grade. O GSR possui um tempo de expiração explícito. Assim, a OGSA define mecanismos para a obtenção de novos GSR a partir de um GSH [44].

A arquitetura de Serviços *Web* é extensivamente utilizada pela OGSA, mas, por padrão, ela não atende um de seus mais importantes requisitos: prover serviços com estado [61]. Serviços *Web* são usualmente sem estado, o que significa que não é possível guardar informações ou manter algum estado entre uma invocação e outra do mesmo serviço. A *Web Services Resource Framework* (WSRF)[17], desenvolvida pela OASIS (http://www.oasis-open.org) em um esforço das comunidades de Grades e Serviços *Web*, é uma especificação que define como produzir serviços *Web* com estado. Para manter o estado de um serviço *Web*, a WSRF fornece o WS-*Resource*, que é uma composição de um Serviço *Web* e um recurso com capacidade de manter seu estado [49]. A WS-*Resource Framework* utiliza puramente WSDL 1.1, garantindo a compatibilidade com as ferramentas existentes e futuras para Serviços *Web* [37].

Temos então que a relação entre a OGSA e a WSRF é que a WSRF fornece serviços com capacidade de manter seus estados o que a OGSA necessita. Podemos ainda expressar esta relação na forma que, enquanto a OGSA é uma arquitetura, a WSRF é uma infra-estrutura na qual a arquitetura é construída. A Figura 3.1 da página 21 mostra esta relação.

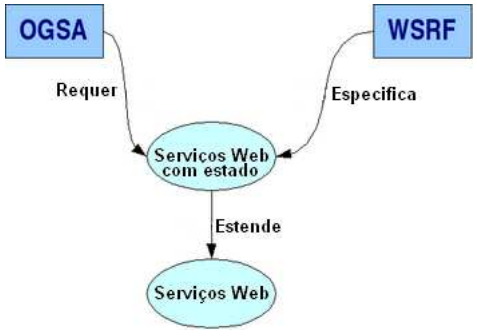

Figura 3.1: Relacionamento entre OGSA, WSRF e Serviços *Web*[61].

O Globus Toolkit 4 implementa as especificações da WSRF e atende requisitos da OGSA. A Figura 3.2 da página 22 mostra os relacionamentos entre Globus Toolkit 4,

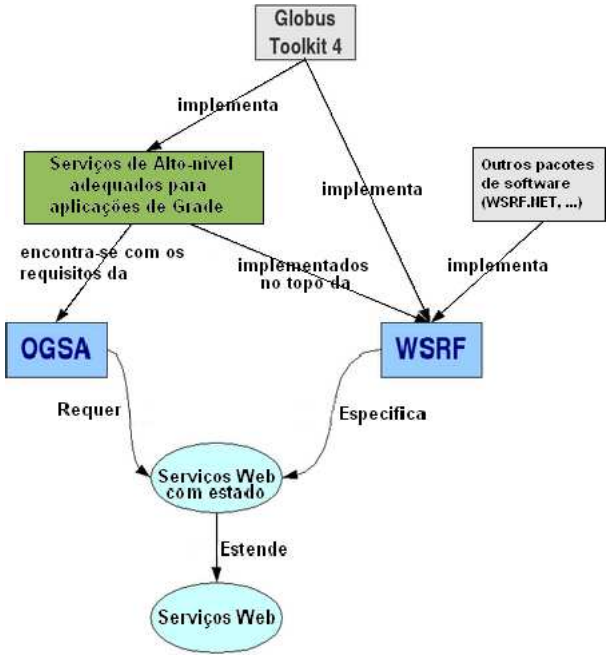

Figura 3.2: Relacionamento entre OGSA, Globus Toolkit 4 (GT4), WSRF e Serviços *Web*[61].

WSRF, OGSA e serviços *Web*.

3.1.1 Serviço de Grade

Para virtualizar e compor serviços, são necessários definições padrões de interfaces [50]. Também é necessário uma semântica padrão para interação com estes serviços, que, por exemplo, forneçam mecanismos padrões para descoberta de propriedades de serviços além de diferentes serviços que seguem uma mesma convenção para notificação de erros.

Para atingir estas necessidades, a OGSA define um serviço de grade: um serviço *Web* que fornece um conjunto de interfaces bem definidas que seguem específicas convenções. Estas interfaces possibilitam a descoberta, criação dinâmica de serviços, gerenciamento do tempo de vida e notificações [45]. Este conjunto de interfaces consistentes sobre as quais são construidos os serviços de grade, facilitam a construção de serviços de forma hierárquica e de alto nível, que podem ser tratadas uniformemente através de camadas de abstrações [50]. Desse modo, toda implementação de serviço de grade deve fornecer as interfaces definidas pela OGSA para serviços de grades.

Os serviços de grade podem manter o seu estado interno durante o seu tempo de vida. A existência do estado distingue uma instância de um serviço de outra instância

que fornece a mesma interface. O termo instância de serviço de grade refere-se a uma instância particular de um serviço de grade. As interfaces e convenções que definem um serviço de grade estão relacionadas, em particular, com os comportamentos relacionados ao gerenciamento de instâncias de serviços transientes.

Mais informações sobre serviços de grades podem ser encontrados em [50].

3.2 Globus Toolkit

O Globus Toolkit é desenvolvido desde o final da década de 1990 para dar suporte ao desenvolvimento de aplicações e infra-estruturas de computação distribuída orientadas a serviços [48]. Possui uma comunidade de usuários e desenvolvedores que colaboram no uso e desenvolvimento do *software* de código aberto e da documentação.

O Globus é um *software* projetado para possibilitar que aplicações "federem" recursos distribuídos, dados, serviços, redes, entre outros. Uma federação é normalmente motivada pela necessidade de acessar recursos ou serviços que não são facilmente replicados [48]. Os primeiros trabalhos com o Globus foram motivados pela demanda de "*Organizações Virtuais*" na ciência, mas esta demanda também se tornou importante em aplicações comerciais.

Ferramentas como o Globus Toolkit têm sido empregadas para resolver os cinco problemas [41] de gerenciamento de recursos introduzidos por grades computacionais [30]:

- Autonomia dos participantes: recursos são tipicamente possuídos e gerenciados por organizações diferentes em diferentes domínios administrativos. Dessa forma, as políticas de uso, políticas de escalonamento e mecanismos de segurança, entre outros, não são comuns a todos os participantes;

- Heterogeneidade: diferentes participantes podem utilizar diferentes sistemas de gerenciamento de recursos. Mesmo quando o mesmo sistema é utilizado por vários participantes, configurações distintas podem acarretar mudanças significativas de funcionalidade;

- Políticas de extensibilidade: como uma solução para gerenciamento de recursos pode ser projetada para um grande número de domínios, é necessário que essa solução ofereça novas estruturas de gerenciamento específicas para um domínio sem a necessidade de alteração do código instalado no participante;

- Co-Alocação de recursos: como muitas aplicações possuem requisitos que só podem ser atendidos através da alocação paralela de recursos em muitos participantes, a autonomia desses e a possibilidade de falhas durante a alocação exige um mecanismo

para alocar múltiplos recursos, iniciar o processamento nesses recursos e monitorar e gerenciar esse processamento;

- Controle *on-line*: negociações podem ser necessárias para adaptar os requisitos de uma aplicação à disponibilidade do recurso, particularmente quando as características dos requisitos e dos recursos se alteram durante a execução.

Além disso, o gerenciamento econômico [30] tem sido proposto como um outro aspecto chave a ser tratado no desenvolvimento de grades orientadas a serviços.

Uma variedade de componentes e funcionalidades estão disponíveis no *toolkit*, incluindo um conjunto de implementações de serviços voltados ao gerenciamento de recursos, movimentação de dados, descoberta de serviços, entre outros. Também encontramos uma ferramenta para construção de novos serviços *Web* em linguagens de programação como Java, C e Python. O Globus Toolkit ainda possui uma infra-estrutura de segurança poderosa baseada em padrões para autenticação e autorização. Todos esses vários componentes possuem documentações detalhadas.

O Globus Toolkit 4 faz uso extensivo de serviços *Web* para definir suas interfaces e estruturar seus componentes, pois fornecem mecanismos flexíveis e extensíveis utilizando padrões amplamente adotados baseados em XML para prover descrição, descoberta e invocação de serviços [48].

3.2.1 A Arquitetura do Globus Toolkit 4

O Globus Toolkit 4 possui diversos aspectos arquiteturais, como podem ser vistos na Figura 3.3 da página 25. Um destaque pode ser dado a três componentes [48]: um conjunto de implementações de serviços, *containers* e bibliotecas clientes.

O conjunto implementações de serviços fornece uma infra-estrutura para gerenciamento de execução, acesso e alteração de dados, gerenciamento de réplicas, monitoração e descoberta, entre outros. A maioria destes são serviços *Web* em Java, mas podendo existir implementações utilizando outras linguagens de programação.

Três *containers* podem ser utilizados para hospedar serviços desenvolvidos pelo usuário, desde que sejam escritos em Java, Python, ou C. Os *containers* atendem requisitos necessários para a construção de serviços, como implementações de segurança, gerenciamento, descoberta, entre outros mecanismos, além de estenderem padrões abertos dando suporte a especificações de serviços *Web*, como o WSRF.

As bibliotecas clientes possibilitam que os clientes programem em Java, C e Python, podendo invocar operações de serviços do Globus Toolkit 4 ou de serviços desenvolvidos por clientes.

Os principais componentes do Globus Toolkit 4 podem ser estruturados em uma perspectiva, a qual separa componentes de execução comum, segurança, gerenciamento de

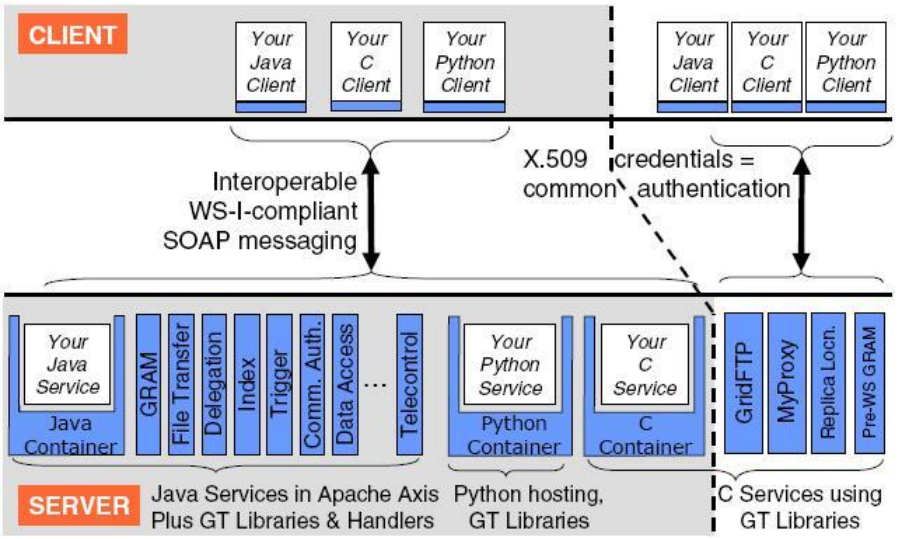

Figura 3.3: A Arquitetura do Globus Toolkit 4 [48].

execução, gerenciamento de dados e informações de serviços [48]. A figura 3.4 da página 26 mostra esta estrutura.

Os componentes de ***execução comum***(*Common runtime*) fornecem bibliotecas e ferramentas que são necessárias para a construção de serviços para execução na Grade Computacional. Encontram-se os módulos de execução Java, C e Python.

Os componentes de ***segurança*** (*Security*), baseados nas especificações da *Grid Security Infrastructure* (GSI), garantem que as comunicações sejam seguras através do uso de cifragem com chaves públicas [9], além de identificar usuários e suas permissões de acesso na Grade [46].

Os componentes de ***gerenciamento de dados*** (*Data management*), permitem gerenciar, fornecer acesso e integrar grandes quantidades de dados de uma ou várias fontes. O Globus possui uma implementação do GridFTP [7] para movimentação de dados [48]. Para gerenciar múltiplas transferências de dados de GridFTP de forma confiável é utilizado o serviço *Reliable File Transfer* (RFT). Para manter e fornecer acesso a informações sobre localizações de dados replicados é utilizado o *Replica Location Service* (RLS). O *Data Access and Integration* (OGSA-DAI) é uma ferramenta para prover acesso a dados relacionais e XML, além de processamento destes dados no lado do servidor. O *Data Replication* possibilita a replicação de dados.

Os componentes de ***informação de serviços*** (*Information services*) possibilitam monitorar e descobrir recursos na Grade Computacional. O *Monitoring and Discovery System 4* (MDS4) fornece informações sobre disponibilidade de recursos na Grade [12]. Esta versão do MDS inclui implementações:

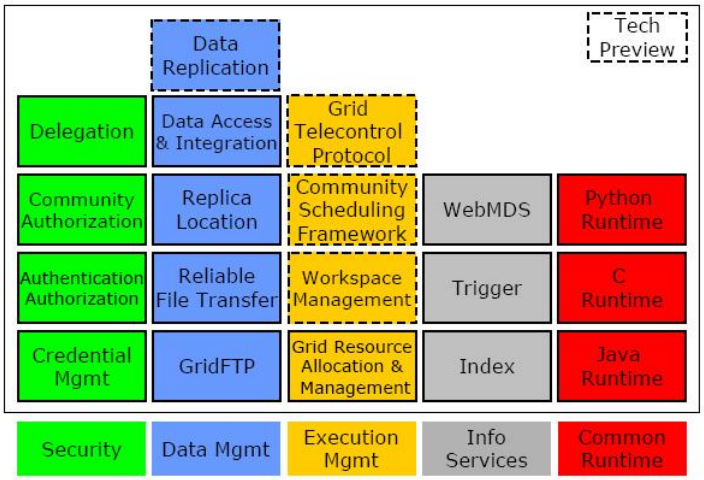

Figura 3.4: Estrutura dos principais componentes do Globus Toolkit 4 (GT4)[48].

- de serviços de registros (***Index***) que coletam informações de recursos e disponibilizam para consulta em um único local [8];
- de serviços de eventos (***Trigger***) que coletam dados de recursos na Grade e, se um administrador define regras, podem ser realizadas diversas ações, como enviar um *email* quando uma fila de computação em um recurso estiver cheia [11];
- do ***WebMDS***, o qual possibilita que usuários finais visualizem e monitorem informações através de uma interface *Web* padrão. WebMDS obtém as informações da monitoração dos recursos, transformado estas informações em XSLT [22], e apresentando os dados em um formato legível aos usuários [6].

Nos componentes de ***gerenciamento de execução*** (*Execution management*) encontramos o *Grid Resource Allocation Manager* (GRAM), que fornece uma interface de serviço *Web* para inicializar, monitorar e gerenciar a execução de qualquer computação em computadores remotos [48]. Por esta interface, é possível expressar tipos e quantidades de recursos desejados, argumentos, credenciais, entre outros. Quando uma tarefa é submetida por um cliente, a requisição é enviada ao servidor remoto e então é manipulada por um *gatekeeper* localizado neste. O *gatekeeper* cria um gerenciador para a tarefa para inicializá-la e monitorá-la. Quando a tarefa é finalizada, o gerenciador envia uma informação de estado de volta ao cliente e termina [44].

Os componentes do Globus Toolkit 4 mais importantes para a OGSA são o GRAM, que fornece meios para a criação e o gerenciamento de serviços remotos; o MDS, o qual fornece dados sobre registros; e o GSI, possibilitando um único registro de entrada na grade, delegação, além de mapeamento de credenciais[51].

3.3 Grid Architecture for Computational Economy (GRACE)

Grid Architecture for Computational Economy GRACE [33] é uma arquitetura concebida para grades computacionais, onde o gerenciamento de recursos é baseado na utilização de modelos econômicos. Essa arquitetura é genérica o bastante para acomodar diferentes modelos econômicos usados para o comércio de recursos. Ela disponibiliza serviços que ajudam tanto os proprietários quanto os usuários de recursos a maximizarem suas funções de objetivo. Os provedores de recursos podem disponibilizar seus recursos na grade e cobrar pelos serviços utilizados. Os usuários interagem com a grade através de definições (que serão geradas por ferramentas de alto nível) de suas necessidades.

Alguns dos componentes dessa arquitetura formam o que é chamado de Grid Resource Broker [31]. O GRB trabalha para os consumidores agindo como mediador entre o usuário e os recursos da grade usando serviços de *middleware*. Ele é responsável pela localização de recursos, seleção e iniciação de computação entre outras funções. Ele ainda apresenta a grade ao usuário como sendo um único recurso. A Figura 3.5 mostra a arquitetura GRACE e os componentes que compõem o GRB.

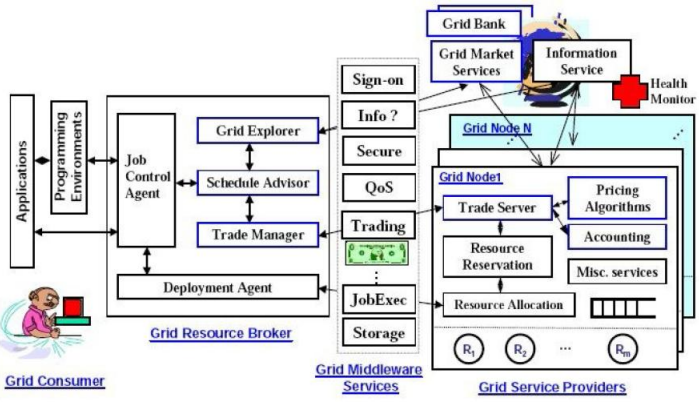

Figura 3.5: Arquitetura GRACE [33].

O ***Job Control Agent*** é o mecanismo de controle responsável pela condução de uma tarefa no sistema. Ele faz a coordenação com o *Schedule Advisor* para controlar o escalonamento, manipula a criação de tarefas, mantém informações sobre o estado das tarefas, além de interagir com clientes/usuários.

O ***Schedule Advisor*** é o responsável pelo escalonamento de tarefas, localizando recursos existentes utilizando o *Grid Explorer* de forma que estes estejam de acordo com os requisitos do usuário.

O ***Grid Explorer*** faz descobertas de recursos através de interações com servidores de informações de grades, mantendo também informações de estado de recurso.

O ***Trade Manager*** trabalha sob direção do *Schedule Advisor* para identificar custos de acesso a recursos. Ele é o responsável pela negociação do uso de algum recurso disponibilizado na grade.

O ***Deployment Agent*** é o responsável pela ativação da execução de tarefas em um recurso selecionado pelo *Schedule Advisor*, além de periodicamente atualizar as informações de estado de uma tarefa ao *Job Control Agent*.

Os GRB's executam uma análise de custo/benefício baseando-se em prazos (tempo no qual os resultados são esperados) e na verba disponibilizada pelo usuário para resolver seu problema. Os proprietários de recursos decidem seus preços baseando-se em diversos fatores. Eles podem cobrar diferentes preços de diferentes usuários pelo mesmo recurso, ou o preço pode variar, dependendo da demanda de utilização.

Em um ambiente de grade, cada usuário possui seu próprio GRB. Os GRB's podem ter objetivos diferentes.

3.4 Nimrod/G

O sistema *Nimrod* [29] foi desenvolvido com o objetivo de ser utilizado para efetuar experimentos paramétricos em uma grade computacional. Esse sistema fornece uma linguagem declarativa simples de modelagem de parâmetros para especificar um experimento paramétrico. O sistema de execução permite a submissão, execução e coleta de resultados de processamento de múltiplos computadores utilizando um conjunto estático de recursos computacionais. Contudo, não é adequado para um contexto dinâmico em larga escala em que os recursos estão espalhados por muitos domínios administrativos. Essa restrição foi tratada pelo sistema denominado *Nimrod/G* que usa o *middleware* do Globus para a localização dinâmica de recursos e despacho de tarefas na grade.

O *Nimrod/G* distribui o processamento entre os participantes da grade computacional através de um escalonamento econômico, possibilitando definir requisitos de tempo e custo a serem empregados na alocação de recursos para a execução de tarefas [28]. Para isso, ele faz uso de três algoritmos com diferentes estratégias de otimização, abordando as restrições de tempo e orçamento:

- **Otimização no custo:** busca minimizar o custo de execução respeitando os requisitos de prazos de execução;
- **Otimização em tempo:** busca minimizar o tempo de execução respeitando os limites de custos definidos;

- **Otimização em tempo e custo:** busca completar a execução com restrições de tempo e custos, minimizando o tempo quando há um alto orçamento disponível. Esse orçamento é utilizado cuidadosamente e garante um valor mínimo por tarefa do total do orçamento disponível para tarefas ainda não processadas.

A arquitetura do Nimrod/G (Figura 3.6) possui os seguintes componentes:

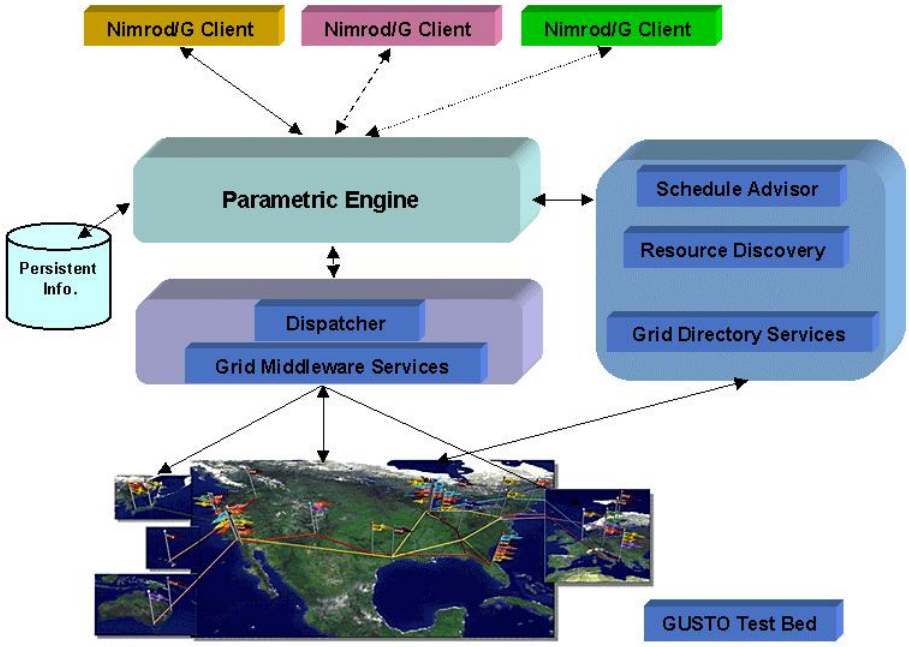

Figura 3.6: Arquitetura Nimrod/G [29].

- ***Client/ User Station:*** Interface que provê ao usuário meios para controlar e supervisionar o experimento em execução, fornecendo opções para que o cliente possa alterar parâmetros como tempo e custo que podem influenciar o escalonador na escolha de recursos;

- ***Parametric Engine:*** Funciona como um controlador persistente de tarefas e é o componente central de onde o experimento é gerenciado. Responsável pela parametrização do experimento, interação com clientes, criação de tarefas, controle do estado da tarefa. Ele recebe o plano de execução, descrito utilizando uma linguagem declarativa para modelagem paramétrica, e envia essas informações ao escalonador. Mantém ainda o estado de todo o experimento em memória persistente, o que permite que este possa ser reiniciado mesmo que um nó executando o Nimrod seja desabilitado;

- ***Scheduler:*** Responsável pela localização de recursos, análises para definir qual dos recursos finalizaria o trabalho de uma determinada tarefa dentro do prazo com um menor custo e atribuição de recursos às tarefas;

- ***Dispacher:*** Responsável por iniciar a execução de uma tarefa em um determinado recurso (através da invocação de um componente remoto chamado Job Wrapper) de acordo com o escalonador. Periodicamente, envia informações de estado da tarefa ao *parametric-engine*;

- ***Job Wrapper:*** Inicia a execução de uma tarefa no recurso local e envia os resultados de volta ao *parametric-engine* através do *dispacher*;

3.5 OurGrid

O *OurGrid* [25][36] é uma solução para computação em grade desenvolvido pela Universidade Federal de Campina Grande, que segue uma política de melhor esforço. Seu principal foco está na execução de aplicações *Bag-of-Task*, em que as tarefas são independentes entre si.

O *OurGrid* explora o conceito de *Rede de Favores*, em que a grade computacional é composta por participantes que têm interesse em trocar favores computacionais entre si. Dessa forma, uma rede *peer-to-peer* de troca de favores é formada, permitindo que recursos ociosos de um participante sejam fornecidos para outros quando solicitados. Para manter o equilíbrio de oferta e consumo de recursos no sistema, níveis de prioridades são definidos. Assim, os participantes que doaram mais recursos enquanto estavam ociosos terão preferência no processo de solicitação de recursos na grade computacional [37].

A Figura 3.7 da página 31 mostra os três principais componentes do *OurGrid*. O *MyGrid*[38] é um intermediador entre o usuário e a grade computacional. O *OurGrid Community* é responsável pela montagem da grade a ser utilizada por instâncias do *MyGrid*. O *SWAN* é o componente que garante acesso seguro aos recursos.

O *MyGrid* busca abstrair o máximo possível esta interação entre o usuário e a grade, simplificando o seu uso. São definidos *GuM's* (*Grid Machine Interface*), que abstraem conjuntos de máquinas, e *GuMP's* (*Grid Machine Provider Interface*), que representam um conjunto de máquinas.

O *MyGrid* possui três implementações nativas de *GuM's* [38]:

- *UserAgent*: é uma implementação baseada em Java, projetada para situações em que é desejado que o usuário instale facilmente o *software* na máquina da grade. A comunicação entre o *MyGrid* e o *UserAgent* é feita via Java RMI (*Remote Method Invocation*);

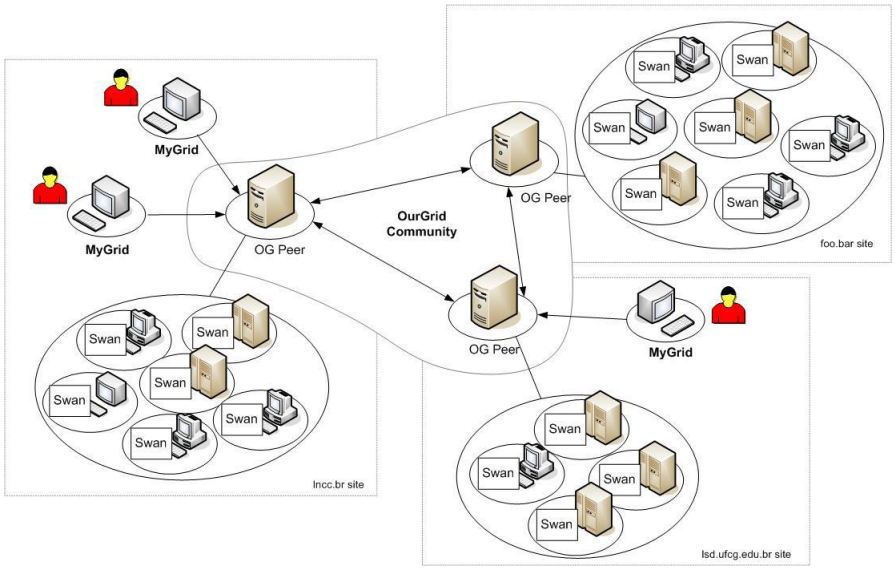

Figura 3.7: Componentes do *OurGrid* [36].

- *GridScript*: fornece as mesmas funcionalidades do *UserAgent* utilizando scripts no lugar da execução de processos Java;
- *GlobusGridMachine*: permite o redirecionamento de requisições recebidas pelo *MyGrid* para uma máquina Globus.

3.6 Conclusões do Capítulo

A OGSA e a WSRF fornecem especificações e padrões para o desenvolvimento de serviços para grades com estado. O Globus Toolkit apresenta diversas ferramentas para a construção, publicação, descoberta, execução e monitoramento de recursos em uma grade. Outros trabalhos relacionados abordam arquiteturas e soluções para grades computacionais utilizando modelos econômicos para o escalonamento de recursos.

O próximo capítulo apresenta o modelo econômico de leilões em uma arquitetura para grades coputacionais.

Capítulo 4

Leilões em Uma Arquitetura para Grades Computacionais

Esse capítulo apresenta uma proposta de arquitetura direcionada para o gerenciamento de recursos disponíveis em uma grade computacional, fazendo uso de modelos econômicos. Esta arquitetura é uma extensão da arquitetura de Teixeira [62]. A extensão aqui proposta faz uso do modelo de *Leilão*, atribuindo valores base para um recurso a partir de uma análise de demanda e oferta dos recursos disponíveis em uma grade computacional.

A arquitetura possibilita que os usuários da grade sejam notificados sobre a realização de futuros leilões de um recurso, avaliando se este atende suas necessidades técnicas, de tempo e/ou de custo. O usuário pode também disponibilizar um recurso próprio para uso na grade através de um leilão. Desse modo, um componente intermediador existente tanto no consumidor quanto no fornecedor do recurso é responsável pela negociação do recurso em questão.

A seguir serão apresentados em maiores detalhes os componentes que fazem parte da arquitetura original e estendida, assim como suas funcionalidades.

4.1 Arquitetura de Grades Computacionais Baseada em Modelos Econômicos

A arquitetura proposta em [62] utiliza uma extensão do modelo de Preço Afixado e consiste em atribuir preços base aos recursos para diferentes tipos de dias e horários, pois a demanda pode possuir um comportamento variado. Assim, ajustando o preço em função desses fatores haverá uma maior distribuição da demanda, possibilitando que haja um possível equilíbrio entre a quantidade de recursos ofertados e a quantidade demandada.

A seguir será apresentado detalhadamente os mecanismos que fazem parte da arquite-

tura como: o gerenciamento de Grupos de Trabalho para dar mais flexibilidade às organizações para manipular seus créditos, formas de pagamento, controle do comportamento honesto do participante, entre outros.

4.1.1 Grupos de Trabalho

Ao utilizar modelos econômicos para o compartilhamento de recursos em uma grade computacional, os agentes participantes (provedores e consumidores) necessitam fazer uso de créditos para a negociação de uso de um recurso.

Os consumidores devem possuir créditos suficientes para realizar o pagamento do uso de um recurso. Para um agente obter créditos é necessário que ele disponibilize seus próprios recursos na grade, onde outros consumidores vão pagar para utilizá-los. Em alguns momentos, um consumidor que necessita utilizar um recurso específico pode não possuir créditos suficientes, devido a um alto valor de custo. A criação de grupos de trabalho pode ser uma possível solução para esse problema.

O principal objetivo de grupos de trabalho é agrupar vários agentes cujos créditos recebidos são tratados como um único montante pertencente ao grupo. Isso possibilita que um participante do grupo faça uso dos valores de créditos do grupo para realizar os pagamentos de recursos que necessita utilizar e aos quais não teria acesso trabalhando independentemente.

Em um grupo de trabalho, os participantes podem ser organizados de forma independente, livres para gerar e consumir créditos. Podem também exercer um ou mais papéis. Estes papéis são:

- **Gerador de Créditos:** a função do participante com este papel é oferecer seus recursos para a grade computacional, com o objetivo de obter créditos para o grupo de trabalho em que participa;
- **Consumidor de Recursos:** o participante com este papel tem a capacidade de utilizar créditos do Grupo de Trabalho em que participa para usar recursos remotos;
- **Coordenador do Grupo:** o participante do grupo de trabalho que possui o papel de Coordenador do Grupo pode acessar as informações do grupo (recursos disponibilizados pelo grupo, créditos disponíveis no grupo, entre outros) e utilizá-las para a definição de parâmetros, além de definir limites de consumo de créditos dos Consumidores de Recursos do grupo.

Em algumas organizações de grupos de trabalho, alguns participantes possuem apenas o papel de geradores de créditos para o grupo, enquanto é possível existir um ou mais participantes com o papel de consumidores de recursos. Nos casos em que todos os

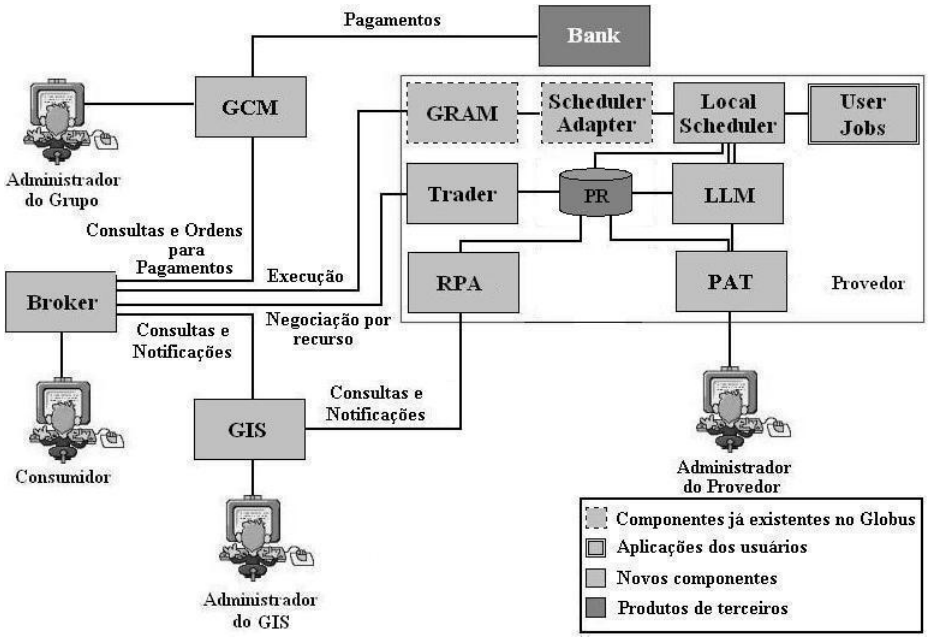

Figura 4.1: Arquitetura de Grades Computacionais - Modelo de Preço Afixado [62].

participantes são geradores de crédito e consumidores de recursos, o Controlador do Grupo define os valores de créditos que cada consumidor de recurso pode utilizar.

As identificações de participantes e de Grupos de Trabalho são fornecidos pelo componente GIS (*Grid Information Service*), o qual mantém informações sobre a grade. Os identificadores são:

- ***GGID:*** o *Grid Group ID* é uma identificação de um determinado Grupo de Trabalho na grade;
- ***GPID:*** o *Grid Participant ID* é uma identificação de um determinado participante. O participante de um grupo de trabalho possui, além de seu próprio e único GPID, a identificação do Grupo de Trabalho em que ele participa. Os participantes que não participam de um Grupo de Trabalho também possuem os identificadores GGID e GPID, porém são tratados como um grupo de um único participante.

4.1.2 Arquitetura dos Participantes

A arquitetura proposta por [62] tem como base o Globus Toolkit e inclui componentes para o escalonamento baseado em modelos econômicos. Os componentes da arquitetura (Figura 4.1) são listados a seguir.

Os componentes do Globus Toolkit na arquitetura:

- **GRAM [46] [10]:** componente fornecido pelo *Globus Toolkit*, possui, além de outras, a função de receber requisições de execução remota. Possibilita as transferências de executáveis para processamento, arquivos de entrada que alimentam o processamento, assim como arquivos de saída, como resultados do processamento.

- ***Scheduler Adapter* [46] [10]:** para a execução de uma requisição, o GRAM necessita de um escalonador local para a execução dos serviços. O *Scheduler Adapter* é um *script* com a função de ser uma interface entre o escalonador local e o GRAM.

Os componentes do provedor são os seguintes:

- **LLM:** O Local Load Manager é um serviço que permanece em execução enquanto o provedor permanece conectado à grade ou tempo integral (a critério do administrador do provedor). Com uma freqüência definida pelo administrador, o LLM verifica a carga do processador local e armazena essa informação no repositório de informações locais do provedor.

- **PR:** o *Provider Repository* é o componente responsável pelo armazenamento persistente das informações locais de um participante. Pode ser implementado utilizando-se um sistema de gerenciamento de banco de dados (SGBD) como o *PostgreSQL*[19], o *MySQL*[14], *Oracle*[18], entre outros;

- ***Local Scheduler:*** é o escalonador local, o qual tem a função de iniciar e gerenciar a execução da aplicação de um consumidor. O *Schedule Adapter* será a interface entre ele o GRAM;

- **RPA:** o *Resource Price Agent* é o componente responsável pela análise de, entre outros, dados colhidos pelo LLM, a oferta e a demanda por um recurso (obtidos através de uma consulta ao GIS, *Grid Information Service*) e determinar os preços-base de um recurso, baseando-se em parâmetros definidos pelo administrador do provedor;

- **PAT:** o *Participant Administration Tool* é uma ferramenta *Web* por onde o administrador de um participante da grade tem mecanismos e interfaces para configuração;

- ***Trader***: componente do provedor responsável por negociar com um consumidor, a utilização de um recurso local. Através dos dados e preços gerados pelo RPA (*Resource Price Agent*) para um determinado recurso, o *Trader* procura alocá-lo de forma a maximizar os lucros do provedor. Esse processo de negociação é estabelecido com o componente *Broker* de um consumidor interessado;

O componente do lado do consumidor é o seguinte:

- ***Broker***: componente utilizado pelo consumidor no momento de submeter uma aplicação para execução. Considerando parâmetros estabelecidos pelo usuário (como por exemplo, limite de preço e prazo de execução), o *Broker* localiza e negocia com provedores de maneira a identificar aqueles que melhor atendam os critérios estabelecidos;

Os outros componentes da arquitetura são descritos a seguir:

- ***Bank:*** é o responsável por controlar a quantidade de créditos de cada participante ou Grupo de Trabalho, realizando as transações de pagamentos referentes ao uso de recursos;

- **GCM:** o *Group Credit Manager* é o componente responsável por coordenar as operações financeiras de grupos. É ele quem determina os valores de crédito que cada participante do Grupo de Trabalho pode gastar, além de autorizar e efetuar os pagamentos relacionados ao uso de recursos por parte dos participantes do grupo;

- **GIS:** o *Grid Information Service* é o componente que mantém um serviço de informações sobre recursos compartilhados. Através dele, o consumidor pode identificar o provedor, obter históricos sobre oferta, demanda, preço médio de um recurso, além da reputação dos participantes;

A próxima seção aborda os mecanismos de geração de preços para os recursos que um provedor deseja disponibilizar para negociação.

4.1.3 Determinação de Preços

A determinação de preços aqui apresentada é voltada para os recursos de processamento.

Um preço-base é definido para cada segundo de UCP (Unidade Central de Processamento), o qual pode sofrer modificações devido ao esforço computacional necessário para atender um prazo de execução de uma tarefa.

A definição do preço-base do recurso é realizada no momento em que se deseja compartilhar o mesmo na grade. Inicialmente, um preço médio é calculado após uma consulta a um GIS. O resultado desta consulta fornece informações que ajudam o administrador do recurso definir um valor condizente com os demais provedores. Para disponibilizar um recurso para leilão na grade, o administrador define um parâmetro de variação do preço-base, podendo deixar o preço abaixo da média para atrair consumidores ao leilão.

Nas subseções seguintes são discutidos os padrões de carga e demanda e o ajuste de preços.

4.1.4 Padrões de Carga e Demanda

A carga de utilização de um recurso em um provedor pode sofrer variação, dependendo do período do dia, dia da semana, dia do mês, entre outros. Um recurso que faça fechamento de horas de um projeto de uma empresa pode ser muito solicitado em períodos finais dos meses. Um recurso que efetua a realização de cópias de segurança (*back-up*) de arquivos de informação pode ser muito solicitado em momentos específicos do dia. Ao mesmo tempo, outros recursos podem ficar ociosos ou com baixa utilização por um dado período.

Devido a essa variação de oferta e demanda por um recurso é interessante que os valores cobrados pelo uso destes também variem de acordo com o período.

O administrador de um recurso pode definir a duração dos períodos. Ele ainda pode definir um dia nas seguintes formas:

- **Dia Normal:** um dia comum sem características significativas;
- **Dia da Semana:** em diferentes dias da semana, o administrador pode definir uma diferenciação dos preços. Supondo que durante o fim de semana (sábado ou domingo) o recurso tem baixa demanda, um valor atrativo para os consumidores pode ser estabelecido;
- **Dia do Mês:** em determinados dias do mês, a demanda por recursos pode ser diferente dos demais dias, sendo necessário aplicar diferentes preços.

Para identificar padrões de demanda em diferentes dias ou períodos, é necessária a existência de históricos para a realização de uma análise. Desse modo, quando um consumidor deseja adquirir o direito de uso de algum recurso remoto, o seu componente *Broker* realiza uma busca no componente GIS por um recurso que atenda seus requisitos.

O componente GIS mantém informações referentes à demanda de algum recurso e processa as análises para a identificação de padrões de demanda. Os dados sobre carga local de um provedor de recurso são tratados pelo componente LLM, que é o responsável por monitorar e armazenar estas informações no PR. Os componentes GIS e RPA podem, em determinados períodos, realizar uma busca pelos históricos de carga e demanda de recursos para os diferentes tipos de dias. O provedor do recurso estabelece os períodos n em que o GIS pode realizar a análise da demanda, e então efetua os cálculos de média aritmética de demanda e o respectivo desvio padrão. O cálculo da demanda média de um recurso considera as demandas dos últimos n períodos, enquanto para os desvios padrão são considerados estes mesmos últimos n períodos e a demanda média.

O administrador do recurso é o responsável por definir o percentual máximo do desvio padrão durante um período, para assim controlar se o comportamento do recurso pode ser considerado padrão.

4.1.5 Ajuste de Preços

O administrador de um provedor define, baseado em valores de consulta ao GIS e parâmetros por ele estabelecido, o preço-base de um recurso que será disponibilizado na grade computacional. O valor definido para tal recurso pode ser um dos fatores a influenciar a decisão de um consumidor em utilizá-lo. Os valores de outros recursos concorrentes podem variar durante o tempo, sendo necessário realizar algum ajuste no valor deste.

O mecanismo de ajuste automático de preços é implementado pelo RPA, que busca otimizar a taxa de utilização do recurso. Portanto, o preço deve ser proporcional à demanda e taxa de utilização, e inversamente proporcional à oferta. Para isso, o RPA realiza análises nos diversos períodos que apresentam uma demanda padrão. Essa análise considera cada um dos registros de preço, o comportamento da oferta e demanda, além do percentual de utilização do recurso disponibilizado. Uma alta taxa de utilização em um determinado período acarretará aumento do preço do recurso naquele período, enquanto quando ocorrer uma baixa taxa de utilização do recurso em outro período, o seu preço poderá sofrer uma queda. Esta diferenciação de preços em períodos variados aumenta a concorrência entre os provedores, gerando um cenário atrativo para os consumidores.

O RPA faz a identificação da variação de demanda e oferta realizando uma consulta ao GIS. Caso a oferta por um recurso aumente, o preço tende a diminuir, enquanto se a oferta diminuir, o preço tende a aumentar. Quando analisada a relação oferta e demanda, os preços também podem sofrer alterações: caso a oferta por um recurso for maior do que a demanda, a tendência é que os preços sofram redução, porém quando a demanda for maior que a oferta, a tendência é o aumento dos preços praticados. Mais detalhes sobre o ajuste de preços podem ser encontrados em [62].

4.1.6 Serviço de Informação de Grade

O componente GIS (*Grid Information Service*) é o elemento responsável por fornecer informações a respeito da grade computacional.

Os provedores que desejam compartilhar recursos próprios na grade fazem um registro dos mesmos no GIS, de modo que estes recursos possam ser localizados por consumidores que desejam utilizar um recurso remoto para realizar suas tarefas. Durante o registro de um recurso no GIS, o provedor deve definir em que tipo de categoria o seu recurso se enquadra, os valores de preço-base do recurso, tempo disponibilizado para processamento (por exemplo, o recurso realiza processamento de uma tarefa que necessite ser finalizada em até 1200 segundos). Quando o consumidor requisita um recurso remoto, ele deve informar o tipo de recurso desejado e/ou as faixas de valores aceitáveis que atendam seus requisitos (como, por exemplo, capacidade de processamento).

O GIS fornece uma tabela com a classificação dos possíveis tipos de recursos que

podem ser consultados, a qual é utilizada pelo consumidor para definir qual recurso será buscado. Novos tipos de recursos podem ser adicionados pelo administrador do GIS.

Além de retornar os resultados das consultas de recursos ao consumidor, o GIS fornece informações sobre a situação econômica da grade como, por exemplo, informações sobre oferta, demanda e preços praticados. A cada vez que um consumidor efetua uma busca por um recurso, a demanda é registrada no GIS, pois neste momento, o consumidor fornece informações sobre o tipo de recurso desejado, o tempo de UCP necessário, além do valor do seu FLOPS (*Floating point Operations Per Second*). Quando a consulta ao GIS é realizada especificando apenas o tipo do recurso desejado, a demanda é registrada apenas para este tipo de recurso. Do mesmo modo, quando a consulta é efetuada baseando-se em faixa de valores, a demanda é registrada para esta mesma faixa de valores pesquisada. O cálculo da demanda por um determinado recurso, então, considera a demanda específica do recurso somada à demanda por faixas.

Em uma organização virtual, os participantes podem ter velocidade de processamento que são diferentes uns dos outros. Devido a esta diferença, o administrador do GIS define um valor de FLOPS a ser utilizado nos registros de demanda por faixa, usando uma medida comum para todos os participantes.

Quando uma negociação é finalizada com sucesso, o componente *Broker* do consumidor informa o componente GIS o tempo de UCP adquirido, preço e tipo de recurso utilizado. Com estas informações, o GIS mantém o preço médio negociado de determinado tipo de recurso. As informações de demanda, oferta e preço médio de um tipo de recurso são armazenadas em um histórico, a partir do qual podem-se realizar análises do comportamento da grade no decorrer do tempo.

A partir do momento em que o processo de leilão de um recurso foi finalizado com sucesso, obtendo-se um vencedor, o componente *Auctioneer* não participa mais da negociação. Assim, já ocorre uma comunicação direta entre *bidder* e *seller*. O GIS oferece informações sobre o comportamento honesto dos participantes, a qual será detalhada na próxima seção.

4.1.7 Sistema de Pagamentos

O mecanismo de pagamento está no componente *Bank*, que é um banco virtual, em que cada Grupo de Trabalho ou o próprio participante autônomo possui uma conta para controlar seu saldo.

Depois da finalização da negociação de um recurso no modelo de leilão envolvendo *bidders*, *seller* e *Auctioneer*, o consumidor (*bidder* vencedor) deve efetuar uma ordem de pagamento junto ao componente *Bank* para o provedor do recurso leiloado (*seller*). O *Bank* emite um recibo que deve ser apresentado ao *Local Scheduler* do provedor. Dessa

forma, o provedor obtém conhecimento que o preço negociado foi pago e, então, inicia a execução das tarefas requisitadas pelo consumidor do recurso.

Mecanismos de segurança são aplicados nas transações de pagamentos para evitar possíveis fraudes. Para isso, a emissão de um recibo deve conter as seguintes informações:

- **Número da transação:** número único que representa a transação efetuada.
- **Número da negociação:** número da negociação efetuada entre provedor e consumidor gerado pelo *Auctioneer* no processo de leilão;
- **GGID do depositante:** identificação do Grupo de Trabalho que efetuou o depósito;
- **GGID do favorecido:** identificação do Grupo de Trabalho para qual foi efetuado o crédito em conta;
- **Valor do depósito:** valor debitado da conta do consumidor e creditado na conta do provedor.

A garantia de autenticidade deste recibo é feita através da autenticação, utilizando a chave privada do banco. Ao receber o recibo, o provedor do recurso pode então verificar a sua veracidade fazendo uso da chave pública do banco. Desse modo, é garantido ao provedor o pagamento do recurso que será utilizado.

Entretanto, a execução de uma aplicação pode não ter sucesso por diversos fatores: houve o desligamento do provedor durante a execução da tarefa, falhas de comunicação, ou o resultado do processamento não foi entregue ao consumidor que estava inoperante no momento, entre outros fatores.

Quando uma negociação é finalizada e o pagamento é efetuado, o provedor envia uma confirmação ao consumidor notificando que a sua aplicação será executada. Ao enviar esta mensagem, o provedor cria um registro de estado de execução que é armazenado no componente LLM do provedor. O registro possui as seguintes informações:

1. **Número da negociação:** número único no provedor, gerado pelo *Auctioneer*, que identifica o processo de negociação em questão;
2. **Provedor ativo:** informa se o provedor de serviços encontrava-se ativo no momento previsto para execução;
3. **Execução iniciada:** informa se a execução da aplicação do consumidor foi iniciada;
4. **Execução concluída:** informa se a execução da aplicação do consumidor foi concluída.

Nos casos em que um pedido de execução de um consumidor não é realizado com sucesso pelo provedor do recurso, o valor pago deve ser restituído. O consumidor deve entrar em contato com o provedor para solicitar a restituição que será realizada dependendo das informações armazenadas no registro de execução. Os possíveis registros de execução e da ocorrência da respectiva restituição, podem ser visto na Tabela 4.1[62].

O administrador do Grupo de Trabalho pode definir critérios de restituição automática de valores quando as falhas ocorrem por parte do provedor ou quando todas as operações não são executadas. Outras restituições solicitadas que não são atendidas automaticamente pelo sistema são apresentadas ao administrador do recurso, para que esse possa, então, autorizá-las manualmente.

Provedor Ativo	**Execução Iniciada**	**Execução Concluída**	**Restituir**
SIM	SIM	SIM	NÃO
SIM	SIM	NÃO	SIM
SIM	NÃO	NÃO	NÃO
NÃO	NÃO	NÃO	SIM

Tabela 4.1: Possíveis valores do *Registro de Estado de Execução* [62].

Quando um consumidor pedir a restituição de valores, se uma execução não teve sucesso por problemas no provedor, pode ocorrer que o mesmo não esteja ativo, devendo-se então esperar um período de tempo e realizar a solicitação novamente. Após um determinado número de tentativas de restituição não atendidas, o provedor deve armazenar um registro de execuções perdidas no GIS. Este registro possui as seguintes informações:

- **Número do registro:** número único que identifica o registro de execução perdida;
- **GGID do provedor:** número de identificação do Grupo de Trabalho do provedor;
- **GGID do consumidor:** número de identificação do Grupo de Trabalho do consumidor;
- **Número da negociação:** número gerado pelo *Auctioneer*, que identifica o processo de negociação entre consumidor e provedor;
- **Valor:** valor da operação.

Quando o GIS armazenar um registro de execução perdida, uma notificação é enviada ao componente GCM do Grupo de Trabalho sobre a reclamação para que o administrador tenha conhecimento do fato ocorrido. O registro possui tempo de vida determinado pelo administrador, após o qual é removido do GIS. Um Grupo de Trabalho que possui um

registro de execução perdida pode retirá-lo efetuando uma restituição do valor junto ao banco do consumidor apresentando o recibo ao GIS.

Quando um consumidor efetuar uma consulta por provedores de recursos, o GIS fornece, para cada provedor, a relação de Grupos de Trabalho que tiveram reclamações. A partir destas informações, o *Broker* pode decidir entre não utilizar recursos de provedores que tenham um número superior a n reclamações.

4.2 Estendendo uma Arquitetura de Grades Computacionais

Esta seção apresenta uma arquitetura para o escalonamento de recursos em uma grade computacional utilizando modelos econômicos. Essa arquitetura é uma extensão do modelo proposto por Teixeira [62], empregando o modelo econômico de *Leilão*.

4.2.1 Arquitetura dos Participantes

A arquitetura aqui proposta, Figura 4.2 da página 43, assim como na arquitetura proposta por [62], também possui componentes do Globus Toolkit, componentes para o lado do provedor, componente do lado cliente e outros componentes que executam em máquinas independentes de provedor e cliente. Enquanto alguns componentes existentes na arquitetura de [62] foram alterados, outro componente para leilão foi incluído. Esses componentes são descritos a seguir.

Os componentes alterados do lado do provedor são:

- **RPA:** o *Resource Price Agent*, responsável pela determinação dos preços-base de um recurso, incorpora informações de configuração do administrador do provedor para disponibilizar um recurso em um leilão para o componente GIS;
- ***Trader:*** o *Trader* é o componente do provedor responsável por interagir com o componente *Auctioneer* durante o processo de leilão[1] de um recurso local. Baseados nos dados gerados pelo RPA para um determinado recurso e parâmetros definidos pelo administrador do provedor, ele participa do processo de leilão disponibilizando o recurso local para o componente *Auctioneer* iniciar um leilão, gerencia os limites de lances[2] e taxas de variação de preços do recurso, analisando a demanda por parte dos consumidores[3];

[1] *Auction*
[2] *Bid*
[3] *Bidders*

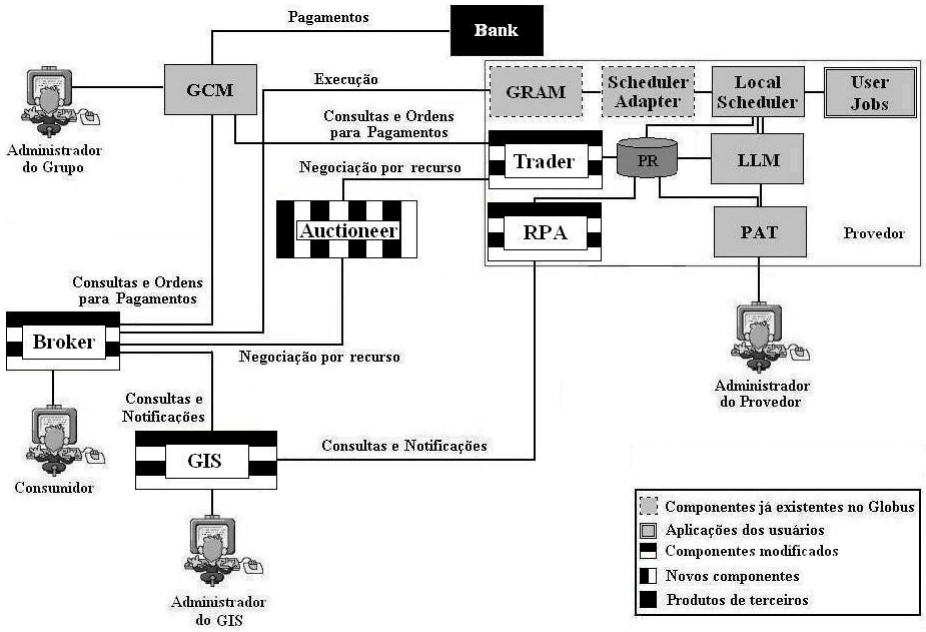

Figura 4.2: Arquitetura de Grades Computacionais - Modelo Econômico de Leilão.

O componente alterado do lado consumidor é o seguinte:

- ***Broker:*** o componente *Broker* encontrado no consumidor, é o responsável pela busca e negociação de um recurso para execução de uma aplicação. A partir de parâmetros estabelecidos pelo usuário, como por exemplo, limite de preço e tempo de execução, ele localiza recursos disponíveis para leilão que atendam seus critérios. A negociação é feita através do componente *Auctioneer*;

O outro componente modificado da arquitetura é o seguinte:

- **GIS:** o *Grid Information Service* é o componente que mantém um serviço de informações sobre recursos compartilhados em processos de negociação em um modelo econômico de leilão. Através dele, o consumidor pode identificar o provedor, obter históricos sobre oferta, demanda, preço médio de um recurso, a forma de leilão e parâmetros de negociação de um recurso, além da reputação dos participantes;

O novo componente inserido na arquitetura é o seguinte:

- ***Auctioneer:*** o *Auctioneer* (leiloeiro) é o componente intermediador entre o provedor de um recurso e os consumidores interessados na utilização deste. Ele é o responsável por receber um recurso do componente *Trader* do provedor para iniciar

um leilão e realizar o processo de negociação com os consumidores através do componente *Broker* existente em cada um dos consumidores. O *Auctioneer* mantém ainda uma lista de *Brokers* que já participaram de leilões por ele intermediado, podendo informar futuros leilões de recursos que possam ser interessantes a um determinado consumidor.

A seguir é apresentado o modelo de leilão na negociação de recursos disponíveis em uma grade computacional.

4.2.2 Leilões na Negociação de Recursos

Para se ter acesso a um recurso disponibilizado em uma Grade Computacional baseada em modelos econômicos, é necessário que as partes (consumidores e provedores) realizem um processo de negociação, com o objetivo de determinar o preço do recurso.

O modelo econômico de leilão é empregado na arquitetura proposta como a forma de negociação do preço de um recurso. Inicialmente o administrador do provedor disponibiliza um recurso na grade, informando o componente GIS sobre o novo recurso e o tipo de leilão em que ele será negociado (Inglês, Segundo Preço, Holandês, entre outros). Neste momento, o administrador do provedor já define para o componente *Trader* todos os parâmetros necessários para o processo de leilão, como preço-base, taxa de variação do valor do recurso para lance, tipo de leilão, além de possíveis limites dos valores mínimos/máximos que o recurso pode ser negociado. Essas informações são repassadas para o componente *Auctioneer*.

Quando um consumidor deseja executar alguma aplicação utilizando um recurso remoto, ele faz uso do componente *Broker*, que conduzirá o processo de localização e negociação. O consumidor define para o componente *Broker* os parâmetros necessários para a participação em um leilão, como faixa de preço aceitos para negociação, limites para lances, além da preferência por um tipo de leilão. Encontrado um recurso desejado, o componente *Broker* comunica ao componente *Auctioneer* sobre o interesse na participação em um leilão de um dado recurso.

O componente *Auctioneer* será o árbitro durante o processo de leilão. O *Auctioneer* dispara o leilão após um período pré-determinado pelo administrador a partir do momento em que o provedor disponibiliza um recurso para leilão. Outro requisito que poderia ser considerado para iniciar um leilão seria um número mínimo de participantes.

Quando um processo de leilão finaliza com sucesso e um *bidder* é vencedor, o *Auctioneer* gera um número único de identificação da negociação, o qual é fornecido ao provedor e consumidor para fins de transação de valores (os pagamentos de recursos serão abordados na sessão 4.1.7), e armazenado em seu histórico, para uma eventual futura auditoria.

Quando analisa-se o modelo de leilão sendo aplicado a um ambiente de grades computacionais, é necessário fazer uma correlação entre ativos em leilões tradicionais e recursos em uma grade computacional. No modelo de leilão tradicional, o *Auctioneer* manipula todas as transações entre *sellers* e *bidders*, enquanto em um leilão de um recurso de grade, os *bidders* fazem lances pelo direito de uso de um recurso. Desse modo, assim que um leilão é finalizado, o *Auctioneer* não continua a mediação entre provedores e consumidores [33]. Desse modo, o componente *Auctioneer* aqui proposto, não manipulará as negociações após a definição de um vencedor de um processo de negociação de leilão para um recurso.

A seguir são apresentados os diferentes tipos de leilão que o componente *Auctioneer* pode conduzir.

4.2.3 Leilão Inglês

O administrador do provedor pode em algum momento compartilhar um recurso na grade computacional, disponibilizando o mesmo através do modelo econômico Leilão Inglês. A modalidade de Leilão Inglês é uma das mais utilizadas [33]. O modelo de leilão em questão é aberto permitindo que os *bidders* aumentem os valores dos próprios lances baseando-se nos valores de lances de outros *bidders*.

A Figura 4.3 mostra o diagrama de seqüência dos eventos gerados durante o processo de negociação na modalidade Leilão Inglês.

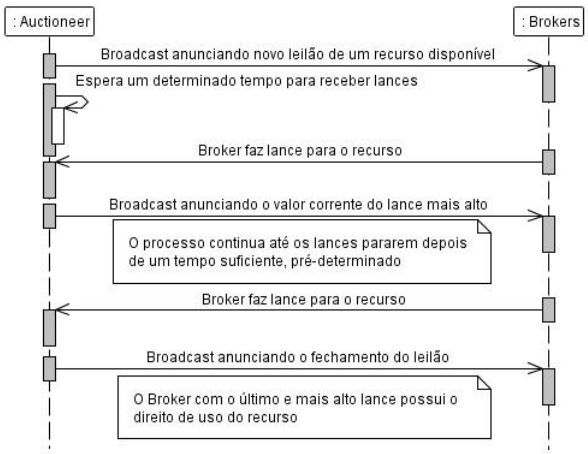

Figura 4.3: Diagrama de Seqüência - Leilão Inglês.

O processo de negociação através do Leilão Inglês inicia-se com o *Auctioneer* transmitindo uma mensagem (via *broadcast*) a todos os *bidders* interessados que o recurso está disponível para receber lances (*bids*). A partir desse momento, o *Auctioneer* pode receber vários lances de diversos participantes dentro de um período de tempo (tempo suficiente

para receber lances de participantes muito remotos), sempre enviando um *broadcast* do lance mais alto corrente aos *bidders*. Este processo de receber lances e fazer *broadcast* do maior lance corrente continua até que o envio de novos lances parem, após um determinado período de tempo suficiente pré-determinado. Ao final do processo, o *bidder* com o lance mais alto tem o direito de utilizar o recurso leiloado. Em caso de empates, o primeiro lance gerado é o vencedor.

Se não ocorrerem lances, o componente *Trader* pode refazer uma avaliação do valor que está cobrando por seu recurso, atualizando seu preço de acordo com os preços praticados na grade computacional, como foi visto na sessão 4.1.5 deste capítulo.

Aspectos positivos e negativos podem ser encontrados no Leilão Inglês. Uma das vantagens do leilão inglês é que, por ser um leilão aberto, os consumidores têm a possibilidade de ver os lances dos demais consumidores. Esta visibilidade permite que os lances de um consumidor sejam ajustados sensivelmente de acordo com os lances que estão sendo praticados. Outra vantagem do uso do Leilão Inglês é que, quando existem muitos *bidders* interessados, a negociação será finalizada com um preço mais interessante ao provedor do recurso. Por outro lado, o alto nível de comunicação necessária neste modelo é uma desvantagem, já que para cada lance dado por um *bidder*, o *Auctioneer* necessita enviar uma mensagem para cada um dos demais *bidders* participantes. Outro problema encontrado é que o uso de recursos pode ser restringido aos grupos de trabalho/consumidores com maior poder aquisitivo. Este comportamento pode ser prevenido limitando um *Broker* a participar de um limitado número de leilões em um período de tempo.

4.2.4 Leilão Holandês

O administrador do provedor pode disponibilizar um recurso através do Leilão Holandês, que também é um leilão aberto. O processo publicação do recurso e solicitação de participação no leilão por parte do *Broker* ao *Auctioneer* ocorre de forma semelhante ao Leilão Inglês. Entretanto, o processo de leilão é feito de forma diferente.

A Figura 4.4 mostra o diagrama de seqüência dos eventos gerados durante o processo de negociação na modalidade Leilão Holandês.

No início do processo do Leilão Holandês, o *Auctioneer* anuncia a todos os *bidders* participantes (via *broadcast*) que o recurso está disponível para receber lances. Dado um período de tempo sem lances, o *Auctioneer* reduz o valor do preço do recurso, novamente enviando um *broadcast* do novo preço para todos os *bidders* envolvidos. Esse processo continua até que um *bidder* faça um lance ou o valor do recurso chegue a um limite mínimo. O primeiro lance de um *bidder* a chegar ao *Auctioneer* adquire o direito de utilizar o recurso leiloado.

Assim como na modalidade de Leilão Inglês, se o recurso não recebeu algum lance

até a redução do valor ao preço mínimo estabelecido pelo administrador do provedor, o componente *Trader* pode atualizar o preço de acordo com os valores praticados na grade, como visto na seção 4.1.5.

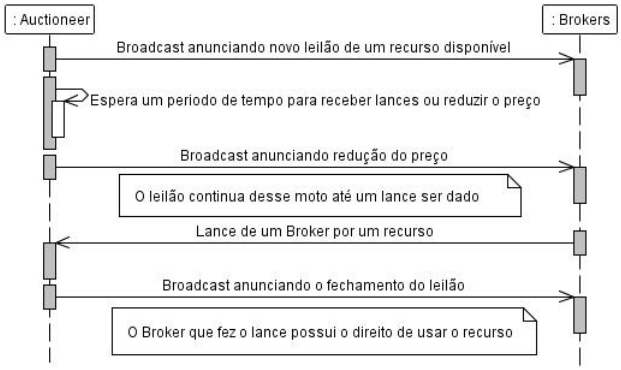

Figura 4.4: Diagrama de Seqüência - Leilão Holandês.

Uma importante vantagem do Leilão Holandês, é que ele força que o conceito de oferta-demanda tenha um importante papel no leilão. Quando um recurso possui uma alta demanda, os *bidders* farão lances mais cedo, conseqüentemente pagando um valor maior pelo recurso. Por outro lado, quando a oferta de recursos é maior que a demanda, os *bidders* não darão lances até que os preços sejam reduzidos.

Apesar de o Leilão Holandês ter menos comunicação do que o Leilão Inglês, o número de mensagens geradas ainda é alta. A cada redução de preço de um recurso, o *Auctioneer* necessita realizar o *broadcast* do novo preço do recurso aos *bidders* envolvidos no leilão, gerando uma grande quantidade de mensagens. A supervalorização de um recurso não é um problema, mas existe um potencial de subvalorização do mesmo [33].

4.2.5 Primeiro Preço

Os modelos de leilão vistos anteriormente (Inglês e Holandês) são chamados de leilões abertos, pois todos os participantes têm conhecimento do lance mais alto, podendo decidir dar um lance maior do que está sendo ofertado no momento.

Contudo, encontramos também os leilões fechados, em que os lances são apresentados simultaneamente ao *Auctioneer* em mensagens fechadas. A modalidade de leilão Primeiro Preço é um tipo de leilão fechado, em que ganham os *bidders* que fizerem os melhores lances. Os leilões fechados exigem que cada *bidder* faça o lance considerando exclusivamente informações de conhecimento do mercado, pois ele somente toma conhecimento dos demais lances quando o leilão está encerrado [57].

A Figura 4.5 mostra o diagrama de seqüência dos eventos gerados durante o processo de negociação na modalidade Leilão Primeiro Preço.

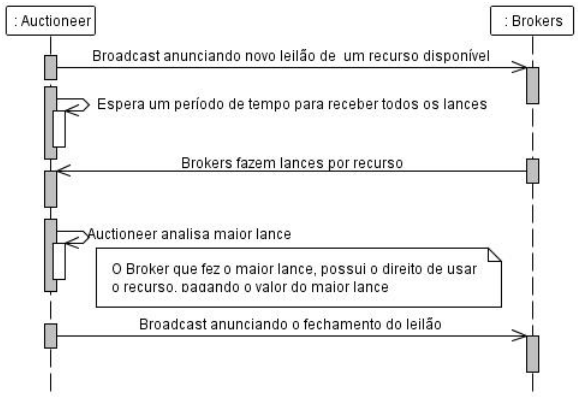

Figura 4.5: Diagrama de Seqüência - Leilão Primeiro Preço.

O processo para requisitar participação no leilão é semelhante ao da modalidade Leilão Inglês e Holandês. O processo do leilão Primeiro Preço inicia-se com o componente *Auctioneer* enviando uma mensagem para cada *bidder* participante, informando que o recurso está disponível para receber lances. Cada *bidder* envia seu lance, sem conhecimento dos valores dos demais *bidders* concorrentes. Após um determinado período de tempo, o *Auctioneer* analisa todos os lances recebidos, e então atribui o direito de uso do recurso ao *bidder* que efetuou o maior lance. Em caso de empates, o primeiro lance gerado é o vencedor.

Esta forma de leilão faz com que os provedores (*sellers*) submetam os preços-base dos recursos com valores similares ao seu custo, e os *bidders* façam lances de quantidades equivalentes à sua capacidade de aquisição e conhecimento de mercado [42], fazendo análises sobre resultados de consultas ao GIS.

No modelo de leilão Primeiro Preço, os lances produzem um comportamento semelhante ao Leilão Holandês, com apenas um único maior lance que define o *bidder* com direito de uso do recurso. Entretanto, não existe a carga de comunicação para envio de mensagens para cada *bidder* sobre redução de preços, pois não há atualização de valores durante o leilão.

4.2.6 Segundo Preço

O modelo de leilão Segundo Preço, também conhecido como *Vickrey* [63], é um leilão fechado e semelhante ao modelo de leilão Primeiro Preço.

A Figura 4.6 mostra o diagrama de seqüência dos eventos gerados durante o processo de negociação na modalidade Leilão Segundo Preço.

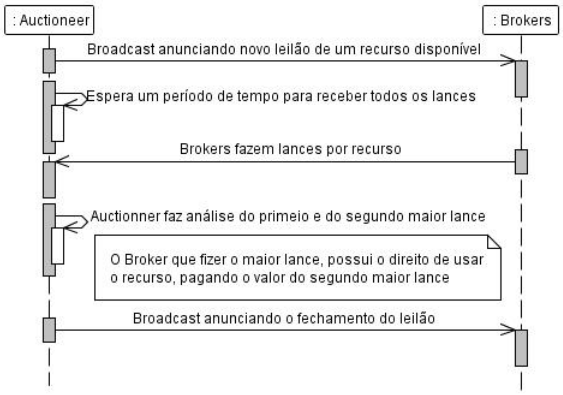

Figura 4.6: Diagrama de Seqüência - Leilão Segundo Preço.

O processo de negociação se dá com os *bidders* enviando lances para o *Auctioneer* sem haver conhecimento do lance dos *bidders* concorrentes. Ao *bidder* que fizer o maior lance de todos, é atribuído o direito de utilizar o recurso leiloado. Entretanto, o valor a ser pago pelo recurso será o preço do segundo maior lance recebido pelo *Auctioneer*. Em caso de empates, o primeiro lance gerado é o vencedor.

4.3 Conclusões do Capítulo

A arquitetura proposta estende uma arquitetura proposta por [62], em que alguns componentes são alterados e outros são criados para suportar o modelo econômico de leilão em um ambiente de grades computacionais.

O novo componente *Auctioneer* fornece negociações em diferentes modalidades de leilão: leilão inglês, holandês, primeiro e segundo preço. A utilização de modelos de leilão possibilita a negociação justa de um fornecedor para muitos consumidores, onde todos os participantes possuem chances iguais de concorrer ao direito de uso de um recurso.

O próximo capítulo apresenta a implementação de um simulador do comportamento dos participantes em um modelo econômico de leilão.

Capítulo 5

Implementação do Modelo Econômico de Leilão

Modelos econômicos de leilão são amplamente utilizados em negociações de bens. Eles fornecem um ambiente justo para negociações, pois sua formalidade tende a garantir que todos os potenciais compradores tenham chances iguais [60]. Além disso, o modelo em questão apoia negociações de um-para-muitos, entre um provedor de um recurso e vários consumidores, em um processo que reduz a negociação a um único valor (preço) [32].

O modelo econômico de leilão pode ser aplicado também em um ambiente de grade computacional, onde recursos disponíveis podem ser alocados remotamente, gerando créditos para os fornecedores e prestando serviços para os consumidores.

A implementação foi realizada através de uma simulação do comportamento do modelo econômico de leilão em uma grade computacional, baseada na arquitetura proposta no Capítulo 4. Essa opção foi adotada pois não compromete a avaliação da arquitetura e por não termos disponível um ambiente com vários computadores utilizando um *middleware* de grades como o Globus Toolkit. Nas próximas seções são descritos mais detalhes sobre a implementação.

5.1 Tecnologias e Ferramentas

Como grades computacionais agregam ambientes heterogêneos, tecnologias para o seu desenvolvimento devem ser portáveis entre diversas plataformas. Por essa razão, o simulador para a arquitetura proposta no Capítulo 4 foi implementado utilizando a tecnologia *Java* [54]. Como ambiente de desenvolvimento, foi escolhida a ferramenta *NetBeans*6.7 [16].

O ambiente de simulação do modelo econômico de leilão necessita que os participantes possuam independência uns dos outros, além de concorrência para tomadas de decisões: enquanto um participante está participando de um processo de leilão, outro participante

poderia atualizar informações de seus recursos. Para prover este comportamento, foi adotada a implementação de *Java Threads* [13]. A *Máquina Virtual Java* [20] permite que aplicações possuam múltiplas *threads* de execução trabalhando concorrentemente.

Para guardar as informações da simulação, a ferramenta *db4o* [2] foi escolhida. Esta ferramenta é um banco de objetos de código aberto sob a licença GPL [5], com suporte tanto para tecnologias *Java* quanto para .Net [15].

A tecnologia *Java* aliada ao ambiente de desenvolvimento *NetBeans* possibilita a escolha de diversos sistemas operacionais como base. Dessa forma, optou-se para execução da simulação de modelos econômicos de leilão o sistema operacional *Debian Linux* [3].

No que diz respeito a *hardware*, foi utilizado um computador com processador Intel Celeron 440 de 1.86 GHz, com 1024 MB de memória RAM e disco rígido de 100 GB.

5.2 Detalhes de Implementação

Os principais módulos existentes na execução da simulação do modelo econômico de leilão são:

- ***Participant:*** elemento consumidor ou provedor de um recurso;
- ***GIS: Grid Information Service:*** serviço de informação sobre serviços da grade computacional;
- ***Auctioneer:*** leiloeiro - conduz a negociação de um recurso.

Foi asbtraído da arquitetura o componente *Bank*. A seguir serão detalhados os módulos acima listados.

5.2.1 *Participant*

Dentro de um ambiente de um leilão, um agente pode ter tanto o papel de fornecedor quanto de consumidor de recursos, dependendo das regras do grupo de trabalho em que ele se encontra.

Desse modo, para o ambiente de simulação, é definido o elemento *Participant*, o qual possui os módulos ***Broker***, ***Trader*** e **RPA**. O *Participant* mantém informações sobre valores monetários, como o total de dinheiro disponível, além de um valor limite de orçamento para participação em cada leilão (*Budget*). Também são guardadas as informações sobre o recurso que pode ser disponibilizado por ele, seu preço e regras para venda, além de uma lista dos recursos que o participante pode desejar consumir durante seu tempo de vida.

O ***Broker*** é o módulo responsável pela negociação em um leilão quando o participante deseja consumir um recurso. A partir de regras definidas previamente por um administrador, o *Broker* realiza buscas no ***GIS*** por algum recurso disponível que atenda suas necessidades, e informa o ***Auctioneer*** sobre o interesse na participação do leilão de um recurso. Quando o *Broker* se comunica com o *Auctioneer*, ele recebe um objeto *SalesRule*(regras de venda), o qual possui todas as informações referentes ao leilão do recurso desejado (preço-base, taxa de elevação de lances, tipo do recurso, entre outros). Assim que sinaliza ao *Auctioneer* a participação em um leilão, o *Broker* espera até que o recurso que ele deseja adquirir esteja aberto para receber lances.

O ***Trader*** é o módulo responsável pela negociação em um leilão quando o participante deseja vender (fornecer) um recurso. O *Trader* realiza a comunicação com o *Auctioneer*, com o objetivo de prover um recurso para ser leiloado. Ele também é responsável por controlar as regras de venda definidas pelo seu administrador, decidindo por exemplo, se o preço de um recurso em um leilão deve sofrer redução, respeitando um limite estabelecido.

O módulo **RPA** (*Resource Price Agent*) é responsável por informar o componente GIS sobre um recurso disponível para consultas, além de manter os valores iniciais dos recursos. De tempos em tempos, os preços dos recursos são atualizados: através de consultas ao componente GIS, ele obtém informações sobre a oferta e a demanda dos recursos concorrentes também disponibilizados no GIS, aplicando pesos sobre essas informações para variar o preço cobrado.

O funcionamento dos módulos *Broker*, *Trades* e RPA deve ser de forma concorrente. Para que isto ocorra, cada instância do elemento *Participant* instancia uma *thread* para cada um destes módulos. Com isso, um participante pode, ao mesmo tempo, estar comprando um recurso em um leilão e vendendo o próprio recurso em outro. Da mesma forma o preço do recurso disponibilizado estará, de períodos em períodos, sendo atualizado pela *thread* do RPA.

5.2.2 *GIS: Grid Information Service*

Para que um consumidor consiga localizar um recurso desejado, é necessário uma forma de busca em um local comum, onde os fornecedores destes recursos possam disponibilizá-los.

É no módulo GIS que o componente RPA de um fornecedor cadastra o recurso que deseja compartilhar, e o *Broker* do consumidor busca um recurso que atenda seus requisitos. O GIS fornece uma lista com informações de recursos que estão disponíveis para uso remoto, além de informações sobre o estado da grade computacional. Algumas dessas informações, como preço médio, demanda e consumo de um tipo de recurso, necessitam de atualizações periódicas. Para isso, uma *thread* é designada para realizar as atualizações de informações de estado da grade.

O RPA calcula o novo preço de um recurso baseando-se em valores de oferta e demanda corrente com valores de oferta e demanda de um período anterior. Esses valores são calculados e fornecidos pelo GIS, o qual informa a demanda e oferta corrente e de um período n anterior.

5.2.3 *Auctioneer*

Durante o leilão de um recurso, toda a negociação é intermediada pelo *Auctioneer* (leiloeiro). Desse modo, um consumidor deve manifestar ao *Auctioneer* sua intenção em adquirir o direito de uso de algum recurso, enquanto um fornecedor lhe comunica da intenção de leiloar um recurso próprio.

O módulo *Auctioneer* necessita estar ativo durante o tempo de vida dos participantes. Para isso, uma *thread* é designada para controlar os processos de leilão.

O módulo *Trader* de um fornecedor registra um recurso para leiloar junto ao *Auctioneer*, fornecendo informações sobre o recurso e regras de venda. O módulo *Broker* do consumidor, após uma consulta ao GIS, solicita ao *Auctioneer* a participação no leilão de um recurso registrado em sua base de dados. Após um tempo determinado pelo administrador, o *Auctioneer* busca um recurso para leiloar e avisa todos os interessados sobre o início do leilão.

Quatro diferentes variações de leilão foram desenvolvidos: leilão inglês, leilão holandês, primeiro preço e segundo preço.

Durante o leilão inglês, os módulos *Brokers* dos participantes interessados no recurso enviam seus lances (*bids*) ao componente *Auctioneer*, e este por sua vez anuncia para todos os participantes o valor atual do recurso em leilão. Outros *Brokers* podem enviar mais lances se possuírem condições para isso. Este processo repete-se até que, depois de um tempo estabelecido, não ocorram novos lances. Se não ocorrer nenhum lance, o preço do recurso pode sofrer redução, de acordo com as regras de venda mantidas pelo módulo *Trader* do participante que está leiloando o próprio recurso.

No leilão holandês, o módulo *Broker* do participante interessado no recurso que enviar o primeiro lance ao componente *Auctioneer* vence o leilão. Caso não ocorra nenhum lance, o preço do recurso sofre redução até um limite definido nas regras de venda do recurso. Se ocorrer esta redução de preço, o *Auctioneer* anuncia para todos os participantes o novo preço do recurso com seu valor reduzido.

Para o leilão na modalidade primeiro preço, cada um dos módulos *Brokers* dos participantes enviam uma única proposta para o *Auctioneer*. O *Broker* que enviar o maior lance obtém o direito de utilizar o recurso. Caso não ocorram lances, o preço do recurso poderá ser reduzido, novamente respeitando as regras de venda do recurso.

O processo do leilão na modalidade segundo preço segue semelhante ao primeiro preço,

porém o *Broker* vencedor irá pagar o valor do segundo maior lance.

O componente *Auctioneer* possui responsabilidades até a definição de um vencedor do leilão. Assim, ele apenas informa o fornecedor (*Trader*) e consumidor (*Broker*) sobre a finalização do leilão de um recurso com seu respectivo valor negociado. O *Auctioneer* não se envolve nos pagamentos para uso do recurso recém leiloado.

5.3 Execução da Simulação de Leilão

Para a execução da simulação dos processos de leilão, foram previamente definidos um número de participantes em uma base de dados. Para que se tenha uma comparação equivalente, a mesma base de dados (participantes) é utilizada nos quatro diferentes tipos de leilão.

Três diferentes tipos abstratos de classificação de recursos foram definidos para gerar uma diversificação dentro do processo de leilão. Desse modo, o componente GIS poderá manter uma lista de vários recursos que atendam diferentes requisitos. Os tipos abstratos de classificação são: *TYPE_A, TYPE_B* e *TYPE_C*.

A Figura 5.1 exibe a interface gráfica para o cadastro de um participante no ambiente para leilão.

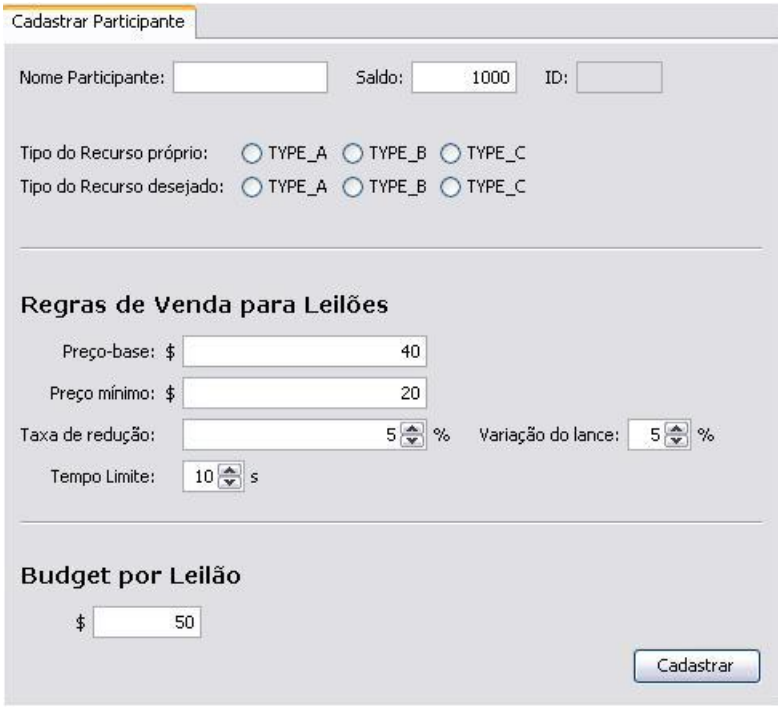

Figura 5.1: Interface gráfica para cadastro de participantes.

O administrador pode definir um nome e saldo disponível para um participante. O identificador ID é gerado automaticamente. Ele também escolhe o tipo de recurso que irá disponibilizar, assim como os recursos que o participante poderá consumir. Um saldo monetário e um valor de orçamento para participação em cada leilão também deve ser fornecido (*budget*).

Nas regras de venda para leilões, o administrador define o preço-base para um recurso. Ele também define um preço mínimo que limita as reduções do valor do preço-base cobrado. Para realizar as reduções de preço, uma taxa de redução deve ser estabelecida. Assim, esta taxa de redução é aplicada ao preço praticado se o novo valor não for menor do que o preço mínimo anteriormente definido.

É estabelecido um tempo limite para os participantes. Este tempo limite será utilizado pelo *Auctioneer* para controlar o tempo de espera por lances em um leilão: quando um recurso receber um lance, após um tempo de espera t sem receber mais algum lance, a negociação é finalizada, ou caso não receba nenhum lance dentro deste mesmo tempo de espera t, o preço-base pode sofrer redução se possível.

Para realizar a simulação do comportamento de leilão, foram gerados 30 participantes, 10 participantes são fornecedores de recursos *TYPE_A* e possíveis consumidores de recursos *TYPE_B* e *TYPE_C*, 10 participantes são fornecedores de recursos *TYPE_B* e possíveis consumidores de recursos *TYPE_A* e *TYPE_C*, e 10 participantes são fornecedores de recursos *TYPE_C* e possíveis consumidores de recursos *TYPE_A* e *TYPE_B*.

A execução das simulações são realizadas em diferentes proporções de oferta e demanda por recursos para cada um dos quatro tipos de modalidades de leilão, os quais estão listados na Tabela 5.1.

Oferta	**Demanda**
100%	100%
100%	75%
100%	25%
75%	100%
25%	100%

Tabela 5.1: Proporções de oferta e demanda na execução dos modelos econômicos de leilão.

Durante a inicialização de um participante, de acordo com as proporções de oferta e demanda, ele pode ser habilitado a trabalhar ou não como um consumidor de recursos e/ou fornecedor de recursos. Caso ele esteja habilitado, durante seu tempo de vida, ele decidirá quando disponibilizará ou buscará consumir um recurso na grade. No ambiente de simulação, esta decisão é feita de forma aleatória, utilizando a classe *Math.Ramdomize()* da *API Java* [54].

Na próxima sessão são apresentados os resultados da simulação do comportamento do modelo econômico de leilões nas modalidades leilão inglês, leilão holandês, primeiro e segundo preço.

5.4 Análise dos Resultados das Simulações

A partir de uma base de dado comum, foram realizadas simulações das diferentes modalidades de leilão, em diferentes relações de oferta e demanda, em um único computador, seguindo os valores listados na Tabela 5.1.

Para cada proporção de oferta e demanda de cada modalidade de leilão, foram realizadas simulações com duração de 8 horas cada. A seguir serão apresentados os gráficos das simulações dos modelos de leilão implementados.

5.4.1 Leilão Inglês

A Figura 5.2 mostra o preço médio negociado dos tipos de recursos em diferentes proporções de oferta e demanda em um leilão inglês.

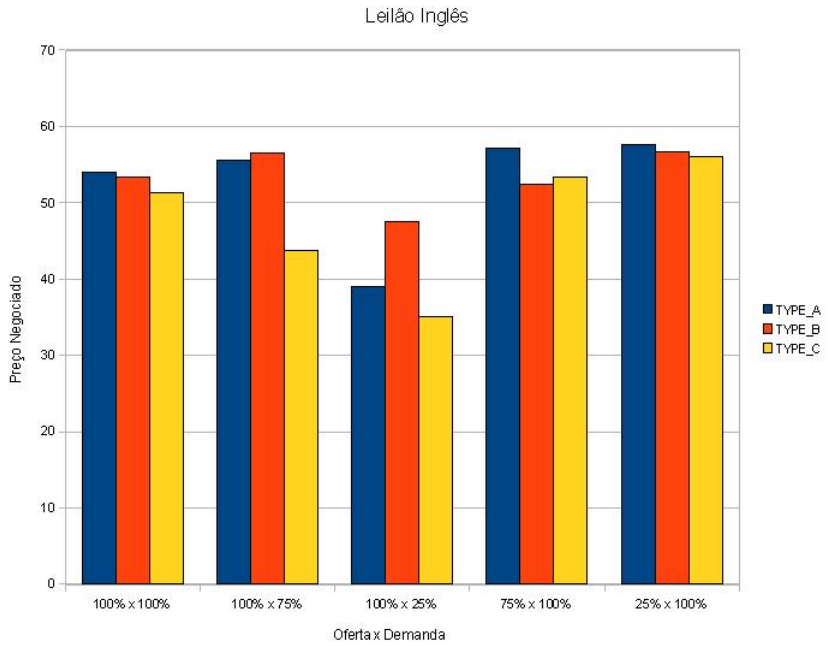

Figura 5.2: Preço médio na negociação de recursos no Leilão Inglês.

No leilão inglês para os recursos classificados como *TYPE_A*, *TYPE_B* e *TYPE_C*, os preços negociados seguem a tendência de oferta e demanda: quando a oferta é maior

que a demanda, o preço reduz, e quando a oferta é menor que a demanda, o preço sofre elevação.

5.4.2 Leilão Holandês

A Figura 5.3 mostra o preço médio negociado dos tipos de recursos em diferentes proporções de oferta e demanda em um leilão holandês.

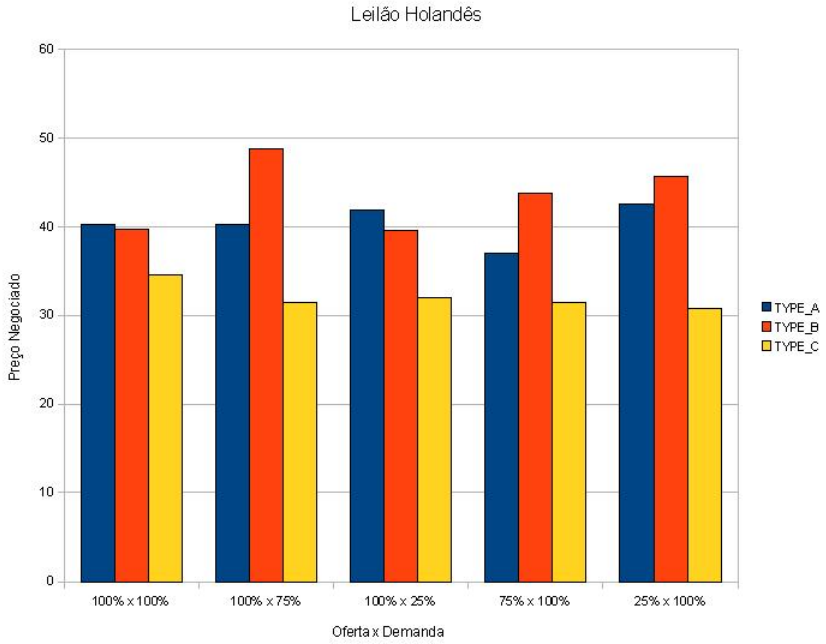

Figura 5.3: Preço médio na negociação de recursos no Leilão Holandês.

Nota-se que, para os recursos classificados como *TYPE_A*, *TYPE_B* e *TYPE_C*, os preços negociados seguem, na média, a tendência de oferta e demanda: quando a oferta é maior que a demanda, o preço reduz, e quando a oferta é menor que a demanda, o preço sofre elevação.

No leilão holandês, o primeiro interessado em um recurso que fizer um lance obtém o direito de uso. Em algumas situações, por mais que a demanda seja baixa, alguns consumidores podem não esperar redução de preços e imediatamente enviam lances para garantir um recurso.

O comportamento do preço de negociação dos recursos no leilão inglês é diferente do comportamento no leilão holandês. Uma das razões para isso é que, enquanto no leilão holandês o primeiro lance é o vencedor, no leilão inglês vários lances podem ser gerados,

aumentando os valores médios negociados quando a demanda é extremamente maior que a oferta.

5.4.3 Leilão Primeiro Preço

A Figura 5.4 mostra o preço médio negociado dos tipos de recursos em diferentes proporções de oferta e demanda na modalidade leilão primeiro preço.

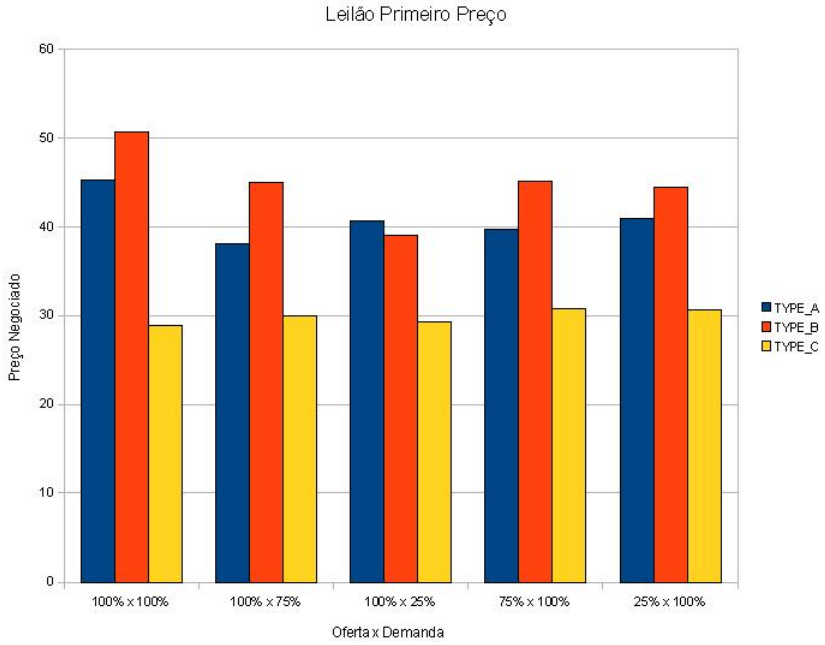

Figura 5.4: Preço médio na negociação de recursos no Leilão Primeiro Preço.

Para os recursos classificados como *TYPE_A*, *TYPE_B* e *TYPE_C*, os preços negociados seguem a tendência de oferta e demanda (oferta maior que a demanda resultando em menores preços, e demanda maior que oferta resultando e aumento nos preços), com o preço médio negociado de *TYPE_C* variando pouco entre as proporções de oferta e demanda.

O comportamento do leilão de primeiro preço é semelhante ao comportamento do leilão holandês, pois um participante não tem conhecimento de outros lances de concorrentes ao recurso. Dessa forma, o interessado em enviar um lance necessita de um maior conhecimento de mercado, ao invés de basear-se nos lances concorrentes. Com isso, se o participante não possui um bom conhecimento de mercado (acessando o GIS e realizando análise nas informações adquiridas) e necessita muito de um certo recurso, um lance alto pode ser enviado, mesmo que a oferta seja maior.

5.4.4 Leilão Segundo Preço

A Figura 5.5 mostra o preço médio negociado dos tipos de recursos em diferentes proporções de oferta e demanda na modalidade leilão segundo preço.

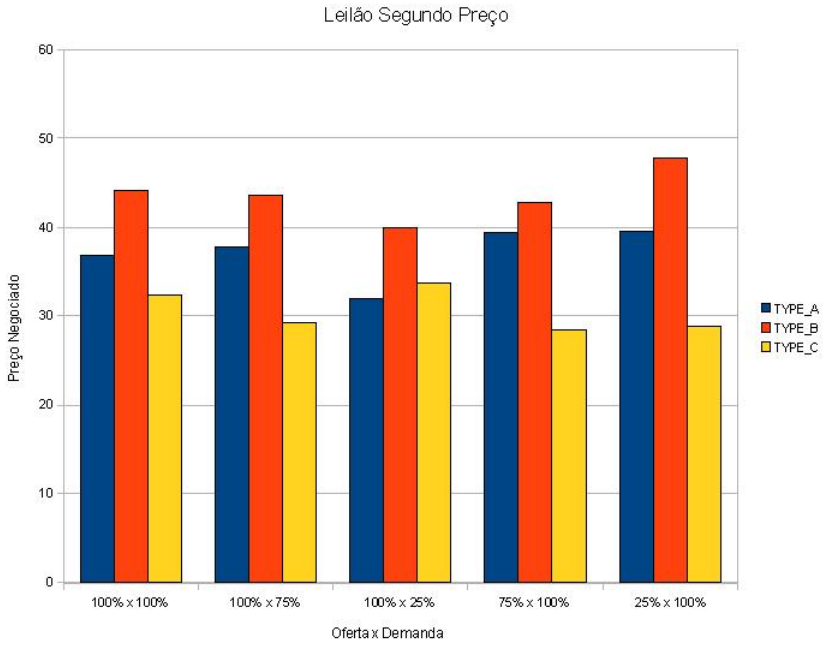

Figura 5.5: Preço médio na negociação de recursos no Leilão Segundo Preço.

Para os recursos classificados como *TYPE_A*, *TYPE_B* e *TYPE_C*, os preços negociados seguem a tendência de oferta e demanda, com o preço médio negociado de *TYPE_C* variando pouco entre as proporções de oferta e demanda.

O leilão de segundo preço segue o mesmo comportamento do leilão primeiro preço, com a diferença que o participante que enviar o maior lance pagará o recurso com o valor do segundo maior lance. Novamente, os participantes necessitam de um maior conhecimento de mercado para gerar lances.

5.4.5 Negociações de Recursos

Durante a execução do simulador de leilões, várias negociações foram efetuadas e registradas no componente GIS. O gráfico da Figura 5.6 mostra o número de negociações efetuadas em cada modalidade de leilão, em diferentes padrões de oferta e demanda listados na Tabela 5.1.

Nota-se um padrão do comportamento dos participantes, mesmo esse comportamento sendo incentivado de forma aleatória. Enquanto a oferta de um recurso é alta e a demanda

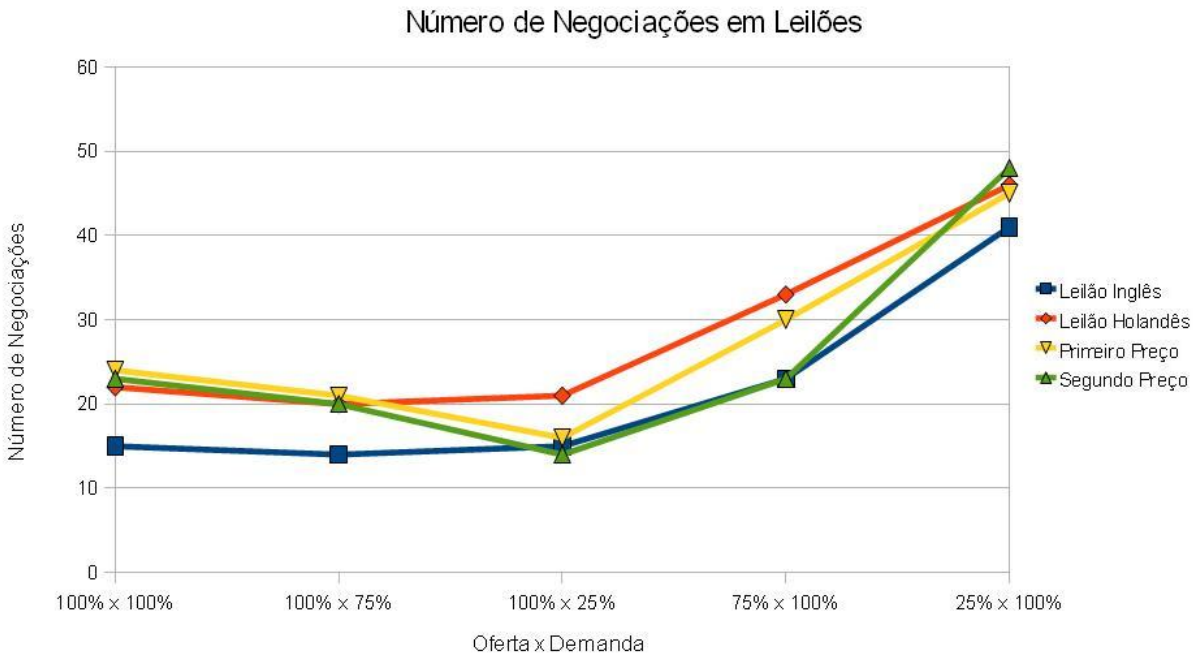

Figura 5.6: Número de negociações em diferentes modalidades de Leilão.

é baixa, o número de negociações é significativamente menor que o número de negociações de quando a demanda pelos recursos é maior que a oferta dos mesmos, pois temos mais interessados em adquirir o direito de uso dos mesmos.

5.4.6 Análise do Número de Mensagens Trocadas

Durante o processo de qualquer modalidade de leilão, mensagens que contém o preço do recurso a ser leiloado ou de valores de lances são trocadas entre o *Auctioneer* e os módulos *Broker* dos participantes.

Quando o *Auctionner* avisa n participantes sobre o início de um leilão, são geradas n mensagens. Caso não ocorra algum lance, o preço do recurso poderá ser reduzido, fazendo-se necessário avisar novamente todos os participantes com mais n mensagens. Caso ocorram m reduções de preço, serão geradas mn mensagens. Ao final do leilão, n mensagens são enviadas para avisar sobre o fim do processo de negociação. Desse modo, todo processo de leilão irá gerar pelo menos $2n+mn$ mensagens, sendo $m = 0..L$, onde L é o número máximo de reduções de preço possível.

As diferentes modalidades de leilão podem gerar número de mensagens diferentes. Essa diferença no número de mensagens pode influenciar no desempenho do sistema em que é realizado o leilão.

Para o leilão inglês, cada lance gerado por um participante deve ser avisado aos demais participantes, para que estes decidam enviar um novo lance. Assim, para cada lance gerado, n mensagens de aviso sobre novo lance serão geradas. Se houver k lances, então haverá ao menos kn mensagens trocadas. Desse modo, para o leilão inglês, serão geradas ao menos $2n+mn+ kn$ mensagens.

No processo de leilão holandês, o número de mensagens é menor do que no leilão inglês. Quando houver um lance, o processo de negociação é finalizado, e o *Auctioneer* avisa todos os participantes sobre o lance vencedor e o fim do processo de leilão do recurso. Dessa forma, ocorrerão ao menos $2n+mn+1$ (um lance enviado define o vencedor).

O processo de leilão primeiro e segundo preço geram o mesmo número de mensagens. O *Auctioneer* avisa os participantes que está aberto para receber uma proposta de cada participante. Se houver n participantes, serão geradas no máximo n mensagens com lances dos participantes. Desse modo, o total de mensagens geradas no sistema será de $(2n+mn)+n$ (número máximo de mensagens de lances enviados).

A Tabela 5.2 lista resumidamente o número de mensagens geradas por modalidade de leilão. Ao analisar a Tabela 5.2, verifica-se que o leilão holandês, primeiro e segundo preço têm o número de mensagens dependendo do número de participantes e número de reduções de preços.

Modalidade de Leilão	**Número de Mensagens**
Leilão Inglês	$n(2+m+k)$
Leilão Holandês	$n(2+m)$ $+1$
Primeiro Preço	$n(3+m\ )$
Segundo Preço	$n(3+m\ )$

Tabela 5.2: Número de mensagens geradas por modalidade de leilão.

Suponha que um recurso qualquer será disponibilizado para negociação em um processo de leilão. Suponha ainda que as regras de venda do recurso limitam o número de reduções de preço a 10, e que o número de participantes no processo de leilão é 30. Por mais que, em teoria, o número de lances em um leilão inglês possa ser extremamente elevado, vamos limitá-lo ao total de 50 lances gerados no máximo.

A partir desses valores e, baseando-se no número de mensagens geradas para cada modelo de leilão como pode ser visto na Tabela 5.2, a Tabela 5.3 mostra qual seria o número mínimo e máximo de mensagens geradas em cada modalidade.

No leilão inglês, para se obter um número máximo de mensagens com os valores acima citados, será necessário que o *Auctioneer* anuncie todos os 30 participantes sobre o novo leilão (30 mensagens). Antes de o recurso receber algum lance, o preço sofreu 10 reduções de preço, e a cada redução é enviado o novo preço para os participantes (300 mensagens). Depois das reduções de preços, os participantes começam a gerar lances, num total de 50

ofertas, onde todos os participantes ficam sabendo de todos os lances (1500 mensagens). Define-se um vencedor, e então o *Auctioneer* avisa todos os participantes sobre o fim do processo de negociação (30 mensagens). Desse modo, o total de mensagens gerada é 30+300+1500+30=1860. O número mínimo de mensagens ocorrerá, no leilão inglês, quando não houver redução de preços e apenas um participante gerar um lance. Assim, o *Auctioneer* gera 30 mensagens para avisar sobre o inicio do leilão, o participante gera uma mensagem que é seu lance, e o *Auctioneer* gera mais 30 novas mensagens para informar o fim da negociação. O total de mensagens é de 30+1+30=61.

No leilão holandês, o número máximo de mensagens ocorrerá quando forem realizadas todas as reduções de preço possíveis. Dessa forma, o *Auctioneer* anunciará o leilão do recurso (30 mensagens), realizará todas as reduções de preços possíveis (300 mensagens), um participante irá gerar um lance (uma mensagem), e o *Auctioneer* informará o fim do processo de negociação (30 mensagens). O total de mensagens no sistema será 30+300+1+30=361. Para ocorrer o número mínimo de mensagens no processo do leilão holandês, nenhuma redução de preço poderá ocorrer. Dessa forma, o *Auctioneer* anuncia o inicio do leilão (30 mensagens), não há redução de preço, um participante gera um lance (uma mensagem). O *Auctioneer* anuncia o fim do processo de negociação (30 mensagens). Assim, temos o número mínimo de mensagens no sistema de 30+1+30=61.

Tanto para o leilão primeiro preço quanto para o leilão segundo preço, o número de mensagens será semelhante. Para ocorrer o número máximo de mensagens no sistema, o *Auctioneer* irá informar todos os participantes sobre o início do leilão (30 mensagens). O preço sofrerá o número máximo de reduções (300 mensagens). Cada um dos participantes enviará seus lances (30 mensagens). O *Auctioneer* avisará o fim do processo de negociação (30 mensagens). Assim, o total de mensagens geradas no sistema será de 30+300+30+30=390. Para ocorrer o número mínimo de mensagens, o *Auctioneer* avisa o início do leilão (30 mensagens). Não ocorrem reduções de preço, e um participante envia um lance (uma mensagem). O *Auctioneer* notifica o fim do processo de negociação. Dessa forma, temos o total de mensagens geradas no sistema igual a 30+1+30=61.

Na modalidade leilão inglês, o número de mensagens pode ser extremamente maior do que os demais modelos, devido ao possível grande número de lances gerados em um processo de leilão de um recurso muito concorrido. Um número de mensagens para leilão

Modalidade de Leilão	**Nº Máximo (msg)**	**Nº Mínimo (msg)**
Leilão Inglês	30(2+10+50)=1860	30(2)+1=61
Leilão Holandês	30(2+10)+1=361	30(2)+1=61
Primeiro Preço	30(3+10)=390	30(2)+1=61
Segundo Preço	30(3+10)=390	30(2)+1=61

Tabela 5.3: Número máximo e mínimo de mensagens geradas por modalidade de leilão.

inglês será apenas próximo do número de mensagens das outras modalidades de leilão se existir somente um lance em todo o processo de negociação.

Capítulo 6

Conclusões

A disseminação da Internet e o aumento das taxas de transmissão, aliados à crescente demanda por processamento, permitiram a criação do modelo computacional denominado computação em grades. A heterogeneidade dos ambientes dos participantes e sua situação geograficamente dispersa exige que o compartilhamento de recursos seja guiado por políticas de alocação visando atender os requisitos de diversos consumidores sem prejudicar os fornecedores de recursos.

Diversos trabalhos desenvolvidos na área da computação em grades buscam formas de gerenciar o comportamento dos participantes ([25], [29] e [33]), sejam eles consumidores ou provedores de recursos. Algumas tecnologias foram desenvolvidas com a intenção de prover infra-estruturas de computação distribuída orientada a serviços, como o Globus Toolkit [48]. Para que exista um entendimento comum no uso destas tecnologias, evitando sobrecarga de conhecimento técnico, alguns padrões para desenvolvimento e descoberta de recursos são aplicados a elas ([17], [21]).

Em um ambiente onde os participantes atuam de forma independente, é necessário formas de conter o comportamento egoísta de cada um deles. Este comportamento egoísta dos participantes pode ter um efeito significativo no desempenho computacional da grade. Modelos econômicos aplicados no tratamento do comportamento egoísta dos participantes possibilitam uma alocação justa dos recursos disponíveis.

O modelo econômico de leilão possibilita que um recurso seja negociado entre um fornecedor e muitos consumidores. Nesse modelo, os fornecedores são incentivados a compartilhar seus recursos em troca de lucros. As informações de qualidade de serviço (QoS) podem ser fornecidas na descrição de um recurso (ou serviço), auxiliando um consumidor a tomar a decisão de participar ou não de um processo de leilão de um recurso desejado.

A arquitetura apresentada, que faz uso de componentes disponíveis no Globus Toolkit, trata o controle do comportamento egoísta dos participantes aplicando o modelo econômico de leilão. Um fornecedor que deseja compartilhar um recurso próprio publica

informações deste em um componente que mantém informações da grade computacional (GIS), e então aciona um elemento *Auctioneer* (leiloeiro) para conduzir o processo de negociação do recurso com os consumidores interessados. Do outro lado, um consumidor que necessita utilizar algum recurso remoto procura um fornecedor que atenda seus requisitos e então aciona o *Auctioneer* responsável pelo recurso, informando a intenção de participar do processo de negociação.

Para avaliar o comportamento dos participantes em um processo de negociação por um recurso, foi implementado um simulador para quatro diferentes modalidades de leilão:

- Leilão Inglês;
- Leilão Holandês;
- Leilão Primeiro Preço;
- Leilão Segundo Preço.

O leilão inglês e o leilão holandês são considerados leilões abertos, pois todos os participantes têm conhecimento de todos os lances gerados.

No leilão inglês, a cada lance enviado, outro lance com um valor maior pode ser gerado. O maior lance gerado, após um certo tempo, é o vencedor. Por mais que a modalidade de leilão inglês seja bastante justa, possibilitando oportunidade para todos os participantes, ela implica em uma grande quantidade de mensagens geradas, pois a cada lance, uma mensagem com o valor do novo lance deve ser enviada a todos os participantes. O leilão holandês reduz o número de mensagens geradas, pois na negociação de um recurso o primeiro lance gerado é o vencedor.

O leilão primeiro e segundo preço são considerados leilões fechados, pois os participantes não têm conhecimento dos lances dos consumidores concorrentes por um recurso.

No leilão primeiro preço, o participante que enviar o maior lance vence o processo de negociação, pagando o valor que ele propôs. Já no leilão segundo preço, o participante que enviar o maior lance vence o processo de negociação, porém o valor a ser pago pelo uso do recurso será o do segundo maior lance enviado. Nesses processos de leilão fechado, o participante não possui informação dos lances dos concorrentes para gerar o seu valor de lance. Desse modo, o consumidor interessado em um recurso necessita de um maior conhecimento de mercado para gerar seu lance.

A decisão sobre a utilização de uma modalidade de leilão deverá levar em consideração a estrutura de comunicação da grade computacional, além do comportamento dos participantes.

As contribuições principais deste trabalho são:

- Uma proposta de arquitetura para grades computacionais baseada em modelos econômicos de leilão;
- Uma análise do comportamento das diferentes modalidades do modelo econômico de leilão.

Como trabalho futuro, pretende-se desenvolver um componente *Auctioneer* que possa ser integrado a diversos *frameworks* que implementem conceitos da OGSA e WSRF. Este componente *Auctioneer* poderá manipular as negociações de diversas modalidades de leilão em um mesmo ambiente, onde o fornecedor de um recurso poderá definir em qual modalidade de leilão deve ser negociado, dependendo do estado da grade. Da mesma forma, um consumidor pode escolher o tipo de modalidade de leilão em que ele deseja participar.

Outros trabalhos futuros incluem:

- Desenvolvimento de componente para modelo de leilão da forma "um consumidor para vários fornecedores";
- Estudo sobre a composição de serviços em uma grade computacional utilizando coordenação ou coreografia, considerando modelos econômicos na alocação de recursos.

Referências Bibliográficas

[1] CORBA. Disponível em: http://www.corba.org/ Acessado em Junho de 2009.

[2] db4o - open source object database. Disponível em: http://www.db4o.com/ Acessado em Novembro de 2009.

[3] Debian.org. Disponível em: http://www.debian.org/ Acessado em Novembro de 2009.

[4] Distribuited Computing Environment. Disponível em: http://www.opengroup.org/dce/ Acessado em Junho de 2009.

[5] GNU GENERAL PUBLIC LICENSE. Disponível em: http://www.gnu.org/licenses/gpl.html Acessado em Novembro de 2009.

[6] GT 4.0 Component Fact Sheet: WS MDS WebMDS. Disponível em: http://www-unix.globus.org/toolkit/docs/4.0/info/webmds/WSMDSWebMDSFacts.html Acessado em Maio de 2009.

[7] GT 4.0 GridFTP. Disponível em: http://www.globus.org/toolkit/docs/4.0 /data/gridftp/ Acessado em Maio de 2009.

[8] GT 4.0 Index Service: Fact Sheet. Disponível em: http://www-unix.globus.org/toolkit/docs/4.0/info/index/WSMDSIndexFacts.html Acessado em Maio de 2009.

[9] GT 4.0 Security: Key Concepts. Disponível em: http://www.globus.org/toolkit/docs/4.0/security/key-index.html. Acessado em Maio de 2009.

[10] GT 4.0 WS GRAM Approach. Disponível em: http://www.globus.org/toolkit/docs/4.0/execution/wsgram /WS_GRAM_Approach.html Acessado em Julho de 2009.

[11] GT 4.0 WS MDS Trigger Service: Fact Sheet. Disponível em: http://www-unix.globus.org/toolkit/docs/4.0/info/trigger/WSMDSTriggerFacts.html Acessado em Maio de 2009.

[12] GT Information Services: Monitoring & Discovery System (MDS). Disponível em: http://www.globus.org/toolkit/mds/ Acessado em Maio de 2009.

[13] Java 2 Platform. Disponível em: http://java.sun.com/j2se/1.4.2/docs/api /java/lang/Thread.html Acessado em Novembro de 2009.

[14] MySQL. Disponível em: http://www.mysql.com/ Acessado em Julho de 2009.

[15] .NET Framework. Disponível em: http://www.microsoft.com/NET/overview.aspx Acessado em Novembro de 2009.

[16] NetBeans IDE. Disponível em: http://www.netbeans.org/ Acessado em Novembro de 2009.

[17] OASIS Web Services Resource Framework (WSRF). Disponível em: http://www.oasis-open.org/committees/wsrf Acessado em Julho de 2009.

[18] Oracle. Disponível em: http://www.oracle.com/database/index.html Acessado em Julho de 2009.

[19] PostgreSQL. Disponível em: http://www.postgresql.org/ Acessado em Julho de 2009.

[20] The Java Virtual Machine Specification. Disponível em: http://java.sun.com/docs/books/jvms/ Acessado em Novembro de 2009.

[21] The Open Grid Services Architecture . Disponível em: http://www.globus.org/ogsa/ Acessado em Julho de 2009.

[22] XSL Transformations (XSLT) Version 1.0. Disponível em: http://www.w3.org/TR/xslt Acessado em Maio de 2009.

[23] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer, November 2003.

[24] J. Altmann and S. Routzounis. Economic Modeling of Grid Services. *Proceedings of the e-Challenges 2006 Conference*, 2006.

[25] N. Andrade, L. Costa, G. Germóglio, and W. Cirne. Peer-to-Peer grid computing with the OurGrid Comunity. *SBRC*, 2005.

[26] R. Butek. Which style of WSDL should I use? Disponível em: http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl. Acessado em Abril de 2009.

[27] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch. Desing and Deployment of a NationalScale Authentication Infrastructure. *IEEE Computer*, 33(12):60–66, 2000.

[28] R. Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD thesis, Monash University, 2002.

[29] R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. pages 283–289 vol.1, 2000.

[30] R. Buyya, D. Abramson, and J. Giddy. A Case for Economy Grid Architecture for Service-Oriented Grid Computing. In *IPDPS*, page 83. IEEE Computer Society, 2001.

[31] R. Buyya, D. Abramson, and J. Giddy. An Economy Grid Architecture for Service Oriented Grid Computing. Technical Report 2001/84, School of Computer Science and Software Engineering, Monash University, Australia 3168, January 2001.

[32] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic Models for Resource Management and Scheduling in Grid Computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[33] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. In *Proceedings of the IEEE*, volume 93, pages 698–714, 2005.

[34] R. Buyya and M. M. Murshed. A Deadline and Budget Constrained Cost-Time Optimisation Algorithm for Scheduling Task Farming Applications on Global Grids. *CoRR*, cs.DC/0203020, 2002. informal publication.

[35] S. J. Chapin, D. Katramatos, J. F. Karpovich, and A. S. Grimshaw. The legion resource management system. In *IPPS/SPDP'99/JSSPP'99: Proceedings of the Job Scheduling Strategies for Parallel Processing*, pages 162–178, London, UK, 1999. Springer-Verlag.

[36] W. Cirne, F. Brasileiro, J. Sauvé, N. Andrade, D. Paranhos, E. Santos-Neto, and R. Medeiros. Grid Computing for Bag of Task. *Third IFIP Conference on E-Commerce, E-Business and E-Goverment*, 2003.

[37] W. Cirne and E. Santos-Neto. Grids Computacionais: da Computação de Alto Desempenho a Serviços sob Demanda. *Minicurso SBRC*, 2005.

[38] L. Costa, L. Feitosa, E. Araújo, G. Mendes, R. Coelho, W. Cirne, and D. Fireman. MyGrid - A complete solution for running Bag-of-Task Applications,.

[39] G. F. Coulouris and J. Dollimore. *Distributed Systems: Concepts and Design.* Addison-Wesley, 2000.

[40] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 6(2):86–93, March 2002.

[41] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82. Springer-Verlag LNCS 1459, 1998.

[42] R. Ethier, R. Zimmerman, T. Mount, W. Schulze, and R. Thomas. A uniform price auction with locational price adjustments for competitive electricity markets. *Electrical Power and Energy Systems*, (21):103–110, 1999.

[43] D. F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic Models for Allocating Resources in Computer Systems, November 15 1996.

[44] L. Ferreira and et al. *Introduction to Grid Computing with Globus*. IBM Redbooks, second edition edition, September 2003.

[45] L. Ferreira and et al. *Grid Services Programming and Application Enablement*. IBM RedBooks, May 2004.

[46] I. Foster. A Globus Primer. Or, Everthing You Wanted to Know about Globus, but Were Afraid To Ask. Describing Globus Toolkit 4. Disponível em http://www.globus.org/toolkit/docs/latest-stable/key/GT4_Primer_0.6.pdf Acessado em Abril de 2009.

[47] I. Foster. The Grid: A New Infrastructure for 21st Centuty Science. *Physics Today*, 55(2):42 – 47, 2002.

[48] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *J. Comput. Sci. Technol*, 21(4):513–520, 2006.

[49] I. Foster and et al. The WS-Resource Framework. Disponível em: http://www.globus.org/wsrf/specs/ws-wsrf.pdf. Acessado em Maio de 2009.

[50] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer*, 35(6):37–46, June 2002.

[51] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, June 28 2002.

[52] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computer Applications*, 15(3):200–222, 2001.

[53] C. F. Goldfarb and P. Prescod. *The XML Handbook*. Prentice Hall PTR, third edition, 2001.

[54] C. Horstman. *Big Java*. Bookman, 2004.

[55] I. Horton. *Begining Java*. Wrox, 1997.

[56] M. Litzkow, M. Livny, and M. Mutka. Condor, a hunter of idle workstations. *8th International Conference of Distributed Computing Systems*, 1998.

[57] G. S. Masili. Metodologia e Software para Simulação de Leilões de Energia Elétrica do Mercado Brasileiro. Master's thesis, Universidade Estadual de Campinas - Faculdade de Engenharia Mecânica, 2004.

[58] C. Neuman and G. Medvinsky. Requirements for Network Payment: The NetCheque Prespective. In *Proceedings of IEEE Compcon '95*, San Francisco, March 1995.

[59] Papazoglou, M. P. and Heuvel, W. van den. Service Oriented Architectures: Approaches, Technologies and Research Issues. *VLDB J*, 16(3):389–415, 2007.

[60] M. H. Rothkopf and S. Park. An elementary introduction to auctions. *Interfaces*, 31(6):83–97, 2001.

[61] B. Sotomayor. The Globus Toolkit 4 Programmer's Tutorial. Disponível em: http://gdp.globus.org/gt4-tutorial/multiplehtml/index.html. Acessado em Maio de 2009.

[62] F. C. Teixeira. Um Escalonador para Grades Computacionais Utilizando Modelos Econômicos. Master's thesis, Universidade Estadual de Campinas - Instituto de Computação, 2006.

[63] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[64] w3schools.com. SOAP Tutorial. Disponível em: http://www.w3schools.com/soap/default.asp. Acessado em Abril de 2009.