FÁBIO ANDRIJAUSKAS

DETECÇÃO DE FILAMENTOS SOLARES UTILIZANDO PROCESSAMENTO PARALELO EM ARQUITETURAS HÍBRIDAS

DETECTION OF SOLAR FILAMENTS USING PARALLEL PROCESSING IN HYBRID ARCHITECTURES

LIMEIRA

2013

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE TECNOLOGIA

FÁBIO ANDRIJAUSKAS

DETECÇÃO DE FILAMENTOS SOLARES UTILIZANDO PROCESSAMENTO PARALELO EM ARQUITETURAS HÍBRIDAS

Orientador/Supervisor: Prof. Dr. André Leon Sampaio Gradvohl

Co-Orientador/Co-Supervisor: Prof. Dr. Vitor Rafael Coluci

DETECTION OF SOLAR FILAMENTS USING PARALLEL PROCESSING IN HYBRID ARCHITECTURES

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Tecnologia da Faculdade de Tecnologia da Universidade Estadual de Campinas para obtenção do título de Mestre em Tecnologia, área de concentração em Tecnologia e Inovação.

Master's Dissertation presented to the Graduate Program in Technology of the School of Technology of the University of Campinas to obtain the title of Master in Technology, concentration area in Technology and Innovation.

ESTE EXEMPLAR CORRESPONDE VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO FÁBIO ANDRIJAUSKAS E ORIENTADO PELO PROF. DR. ANDRÉ LEON SAMPAIO GRADVOHL

Assinatura do Orientador

_____

Limeira

2013

FICHA CATALOGRÁFICA ELABORADA POR SILVANA MOREIRA DA SILVA SOARES – CRB-8/3965

BIBLIOTECA UNIFICADA FT/CTL

UNICAMP

Andrijauskas, Fábio,1986-
An28d Detecção de filamentos solares utilizando processamento paralelo em arquiteturas híbridas / Fábio Andrijauskas. – Limeira, SP : [s.n.], 2013.

Orientador: André Leon Sampaio Gradvohl.
Coorientador: Vitor Rafael Coluci.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Processamento de imagens. 2. Computação de alto de desempenho. 3. Filamentos solares. 4. Análise de imagem. I. Gradvohl, André Leon Sampaio. II. Coluci, Vitor Rafael. III. Universidade Estadual de Campinas. Faculdade de Tecnologia. IV. Título.

Informações para Biblioteca Digital

Título em inglês: Detection of solar filaments using parallel processing in hybrid architectures

Palavras-chave em inglês (Keywords):

1- Image processing

2- High performance computing

3- Solar filaments

4- Image analysis

Área de concentração: Tecnologia e Inovação

Titulação: Mestre em Tecnologia

Banca examinadora: André Leon Sampaio Gradvohl, Marco Antonio Garcia de Carvalho, Stephan Stephany

Data da Defesa: 21-02-2013

Programa de Pós-Graduação em Tecnologia

DISSERTAÇÃO DE MESTRADO EM TECNOLOGIA

ÁREA DE CONCENTRAÇÃO: TECNOLOGIA E INOVAÇÃO

Detecção de filamentos solares utilizando processamento paralelo em arquiteturas híbridas

Autor: Fábio Andrijauskas

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. Andre Leon Sampaio Gradvohl, Presidente
FT/UNICAMP

Prof. Dr. Marco Antonio Garcia de Carvalho
FT/UNICAMP

Prof. Dr. Stephan Stephany
INPE

Dedicatória

Dedico este trabalho a todos que de uma forma ou de outra, contribuíram para que tudo se realiza-se. Em especial a minha esposa que sempre me apoiou junto às adversidades, dando o suporte necessário para a conclusão de mais essa etapa em minha jornada acadêmica.

Agradecimento

Agradeço a todos que nessa jornada contribuíram para conclusão de mais essa etapa, que veio somente engrandecer-me como acadêmico e ser humano. Ao meu orientador Prof. Dr. Andre Leon Sampaio Gradvohl que sempre me conduziu de forma exemplar, com excelência em suas diretrizes de importância imensurável para conclusão dessa pesquisa e o mais importante, acreditou que poderia cursar o mestrado. A Prof. Dr. Regina pelo enriquecimento obtido em minha jornada docente, abrindo visão a um novo leque de oportunidades e desafios a serem desvendados no processo ensino - aprendizagem. Aos Professores: Prof. Dr. Celmar, Prof. Dr. Varese, Prof. Dr. Marcos Borges e Prof. Dr. Vitor pela oportunidade de absorver os conhecimentos transmitidos durante esse período. A minha esposa pela compreensão e dedicação constante, sempre presente em todos os momentos. Sou o resultado da credibilidade e da força de cada um.

Agradeço também a Karen e a Fátima da secretaria, resolvendo os problemas sempre com boa vontade. Ao meus amigos de mestrado Ximenes, Pedro Ivo e em especial ao Koji. Que sempre estavam prontos a ouvir minhas opiniões. Também agradeço a CAPES e Faculdade de Tecnologia da Unicamp pela bolsa de mestrado concedida e aos projetos FAPESP 2010/50646-6 e 2011/00861-0 pela estrutura computacional fornecida.

Resumo

A quantidade de imagens astronômicas geradas cresce diariamente, além da quantidade já obtida e armazenada. Uma grande fonte de dados são imagens solares, cujo estudo pode detectar eventos que têm a capacidade de afetar as telecomunicações, transmissão de energia elétrica e outros sistemas na Terra. Para que tais eventos sejam detectados, torna-se necessário analisar essas imagens de forma eficiente, levando em conta os aspectos de armazenamento, processamento e visualização. Agregar algoritmos de processamento de imagem e técnicas de computação de alto desempenho facilita o tratamento da informação de forma correta e em tempo reduzido. As técnicas de computação para alto desempenho utilizadas neste trabalho foram desenvolvidas para sistemas híbridos, isto é, aqueles que utilizam uma combinação de sistemas de memórias compartilhada e distribuída. Foram produzidas versões paralelas para sistemas híbridos de técnicas já estabelecidas. Além disso, novas técnicas foram propostas e testadas para esse sistema tais como o *Filamento Diffusion Detection*. Para avaliar a melhora no desempenho, foram feitas comparações entre as versões serial e paralela. Esse texto também apresenta um sistema com capacidade para armazenar, processar e visualizar as imagens solares. Em uma das técnicas de detecção de filamentos, o processo foi acelerado 120 vezes e um processo auxiliar para a detecção de áreas mais brilhantes foi 155 vezes mais rápido do que a versão serial.

Palavras-chave: Processamento de imagens, Computação de alto desempenho, Filamentos solares, Análise de imagem.

Abstract

The number of astronomical images produced grows daily, in addition to the amount already stored. Great sources of data are solar images, whose study can detect events which have the capacity to affect the telecommunications, electricity transmission and other systems on Earth. For such events being detected, it becomes necessary to treat these images in a coherent way,considering aspects of storage, processing and image visualization. Combining image processing algorithms and high performance computing techniques facilitates the handling of information accurately and in a reduced time. The techniques for high performance computing used in this work were developed for hybrid systems, which employ a combination of shared and distributed memory systems. Parallel version of some established techniques were produced for hybrid systems. Moreover, new techniques have been proposed and tested for this system. To evaluate the improvement in performance, comparisons were made between serial and parallel versions. In addition to the analysis, this text also presents a system with capacity to store, process and visualize solar images. In one of the techniques for detecting filaments, the process was accelerated 120 times. Also an auxiliary process for the detection of brighter areas was 155 times faster than the serial version.

Keywords: Image processing, High performance computing, Solar filaments, Image analysis.

Sumário

Lista de Abreviaturas

API	*Application Programming Interface* – Interface para Programação de Aplicações.
ASCII	*American Standard Code for Information Interchange* – Código Padrão Americano para o Intercâmbio de Informação.
BBO	*Big Bear Observatory*.
BITPIX	*BITs per PIXel* – BITs por PIXel.
CCD	*Charge-coupled devices* – Dispositivo de Carga Acoplada.
CME	*Coronal Mass Ejection* – Ejeção de Massa Coronal.
DCT	*Discrete Cosine Transform* – Transformada Discreta dos Cossenos.
DEC	*Declination* – Declinação.
DEC	Divisão de tarefas em um nó computacional.
DPC	Divisão de tarefa por nó computacional.
EMBARACE	Estudo e Monitoramento Brasileiro do Clima Espacial.
FDD	*Filamento Diffusion Detection*.
FITS	*Flexible image transport system* – Sistema Flexível de transporte de Imagem.
FN	falso negativo.
FP	falso positivo.
FT	Faculdade de Tecnologia.
GCC	*GNU Compiler Collection* – Coleção de Compiladores GNU.

GIF	*Graphics Interchange Format* – Formato Intercambiável Gráfico.
GONG	*Global Oscillation Network Group*.
GPU	*Graphics Processing Unit* – Unidade de Processamento Gráfico.
HDU	*Header Data Units* – Cabeçalho de Informação Principal.
HST	*Hubble Spacial Telescope* – Telescópio Espacial Hubble.
HTTP	*Hypertext Transfer Protocol* – Protocolo de Transferência Hypertext.
IAU	*International Astronomical Union* – União Astronômica Internacional.
INPE	Instituto nacional de pesquisas espaciais.
IPAC	*Infrared Processing and Analysis Center* – Centro de Processamento e Análise Infravermelho.
IRAF	*Image Reduction and Analysis Facility* – Ambiente para Redução e Análise de Imagem.
IRSA	*Infrared Science Archive* – Arquivo Cientifico Infravermelho.
JFIF	*JPEG File Interchange format* – Intercâmbio Arquivo Formato JPEG.
JPEG	*Joint Photographic Experts Group* – Grupo especialista em Imagem Articulada.
LaSCADo	*Laboratório de Simulação e de Computação de Alto Desempenho*.
LNA	Laboratório Nacional de Astrofísica.
MD	*Morph Detection*.
MPI	*Message Passing Interface* – Interface de Troca de Mensagem.
MVAPICH	*MPI-1 over OpenFabrics/Gen2, OprnFabrics/Gen2-UD, uDAPL, InfiniPath, VAPI and TCP/IP*.

NASA	*National Air and Space Administration* – Administração Nacional de Aeronáutica e do Espaço.
NRAO	*Nation Radio Astronomy Observatory* – Observatório Nacional de Rádio Astronomia.
OpenMP	*Open Multi-Processing*.
PAD	Processamento de Alto Desempenho.
PHP	*Hypertext Preprocessor*.
POSIX	*Portable Operating System Interface* – Interface portátil entre sistemas operacionais.
RA	*Right ascension* – Ascensão Reta.
SDO	*Solar Dynamics Observatory* – Observatório Solar Dinâmico.
SGBD	sistema de gerenciamento do banco de dados.
SIAP	*Simple Image Access Protocol* – Protocolo Simples de Acesso a Imagem.
SIF	*single image FITS* – Imagem Única FITS.
SOAR	*SOuthern Astrophysical Research Telescope*.
TIFF	*Tagged Image File Format* – Arquivos com formato de imagem marcado.
UNICAMP	Universidade Estadual de Campinas.
WCS	*World Coordinates System* – Sistema de Coordenadas Mundial.
WISE	*Wide-Field Infrared Survey Explorer* – Explorador Infrevermelho de Largo Campo.
WWT	*World Wide Telescope*.

Lista de Figuras

Lista de Tabelas

Lista de Equações

Lista de Símbolos

1 Introdução

A maioria da luz que se observa no céu vem de uma estrela anã que se chama Sol (MOCHÉ, 2009). Tal estrela provê energia em diversas formas. Todavia, bloqueia a observação no espectro visível de milhares de objetos da Terra. Quando a face da Terra em que o observador encontra-se não recebe mais essa luz, é possível visualizar diversos pontos luminosos no céu. Esses pequenos pontos podem ser planetas, estrelas ou outros objetos celestes.

Para observar esses objetos com precisão necessita-se do uso de telescópios (ROY; CLARKE, 2003). Esses equipamentos têm a capacidade de concentrar certa quantidade de luz e ainda propiciar a visualização dessa luz por um observador (MORISON, 2008). Os tipos de telescópios que podem ser construídos ou comprados por um valor razoável (cerca de R$10.000,00) apenas capturam a parte visível do espectro da luz. Porém, diversos objetos celestes propagam energia eletromagnética em diversas frequências que não são visíveis ao olho humano (INGLIS, 2007). Há telescópios que capturam ondas de rádio ou raios X, como o *Hubble Spacial Telescope* – Telescópio Espacial Hubble (HST) que tem a capacidade de capturar faixas do infravermelho e do ultravioleta que não são visíveis ao olho humano (FOSBURY, 2006). Outro telescópio, o *Wide-Field Infrared Survey Explorer* – Explorador Infrevermelho de Largo Campo (WISE) da *National Air and Space Administration* – Administração Nacional de Aeronáutica e do Espaço (NASA) captura 4 faixas do infravermelho (MAINZER et al., 2005).

O WISE e o HST estão fora da Terra. Utilizar filmes fotográficos ou a observação direta é completamente inviável. Ainda é necessária uma sensibilidade maior a luz. Para isso, utilizam-se *Charge-coupled devices* – Dispositivo de Carga Acoplada (CCD) (MORISON, 2008). Tais dispositivos são parecidos com os de câmeras digitais comuns, porém com maior sensibilidade, controle e precisão. A facilidade de geração de imagens criada pelo CCD propiciou a geração de grandes volumes de informação.

Pesquisas recentes ressaltam a avalanche de dados que são gerados e armazenados. Berriman e Groom (2011) apontam um grande desafio no tratamento dessas informações. O gráfico da Figura 1 mostra a quantidade de dados armazenados no decorrer do tempo das missões WISE, *Spitzer*, outras missões e suas projeções (a partir do ano de 2012 no gráfico, os valores são estimados). Outra pesquisa que descreve em partes como essa informação deve ser tratada foi descrita por Lawrence (2009). CALTECH (2010) e Kaiser (2009) também apresentam diversas observações em que a quantidade de informação astronômica está crescendo rapidamente.

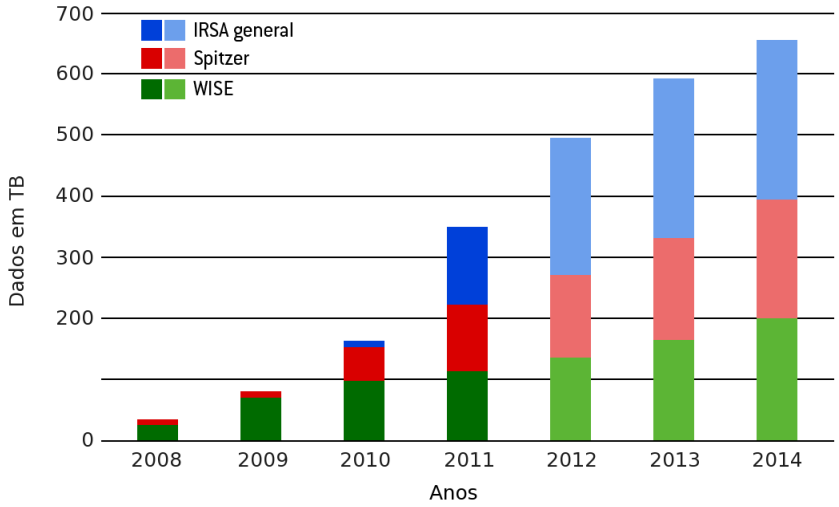

Figura 1: Gráfico com a quantidade de dados armazenados pelas missões *WISE*, *Spitzer* e outras missões no decorrer do tempo e suas projeções. Adaptado de (BERRIMAN; GROOM, 2011).

O *Infrared Processing and Analysis Center* – Centro de Processamento e Análise Infravermelho (IPAC) possui no seus discos de produção 5,5 *PetaBytes*: são imagens das missões *Spitzer*, WISE, *Planck*, *2MASS*, *Akari*, *IRAS*, *NED*, *Kepler*, *CoRoT* entre outras bases menores (CALTECH, 2010). Outro exemplo é o *Solar and Heliospheric Observatory* que transfere cerca de 200 *MegaBytes* de imagens do Sol todos os dias ou *Solar Dynamics Observatory* – Observatório Solar Dinâmico (SDO) que envia cerca de 1,4 *TeraBytes* de imagens por dia (MUELLER et al., 2009). O *Big Bear Observatory* (BBO) produz aproximadamente 600 imagens do Sol em 2048×2048 pixels por dia (GAO et al., 2002). O acompanhamento do Sol é de interesse dos serviços como telecomunicações e transmissão de energia elétrica que podem ser afetados por tempestades solares. Para tal tarefa, pode-se utilizar técnicas de processamento de imagem (SHIH, 2010) para detectar ou evidenciar eventos, inclusive tempestades solares.

As tempestades solares são caracterizadas por diversos eventos: entre eles o surgimento de filamentos na superfície do Sol. Esses filamentos, visíveis em imagens Hα, são aglomerados de plasma Solar sustentados por campos magnéticos. Filamentos estão associados a *Coronal Mass Ejection* – Ejeção de Massa Coronal (CME). As CME podem causar diversos problemas nos sistemas terrestres de comunicação, transmissão de energia, linhas de transporte de petróleo e outros. Diversas pesquisas demonstram a importância e necessidade de fazer a monitoração desses eventos (MOORE, 2003) e (CUI et al., 2010).

Essa necessidade é de tamanha importância que o Instituto nacional de pesquisas espaciais (INPE) em 2011 iniciou o Estudo e Monitoramento Brasileiro do Clima Espacial (EMBRACE). Esse Projeto visa monitorar e modelar e difundir as informações do clima espacial utilizando diversas tecnologias no Brasil e no mundo (INPE, 2011).

A fim de que o volume de informação seja processado em um tempo razoável é necessário utilizar técnicas para que a capacidade computacional disponível seja utilizada de forma plena (RABENSEIFNER, 2003) e (RABENSEIFNER et al., 2009). Tais técnicas, como as de sistemas híbridos, devem ser capazes de aproveitar todos os meandros das arquiteturas de memória compartilhada e memória distribuída.

Nesse cenário, agregar o processamento de imagem e técnicas de processamento de alto desempenho para detectar os filamentos solares é vital para que os eventos sejam processados utilizando de forma plena a capacidade computacional, propiciando um menor tempo de trabalho. Esse decréscimo no tempo leva a utilizar mais dados, habilitando aumentar a quantidade de imagens utilizadas com o objetivo de obter mais informação. Essa motivação vai além da necessidade clara de uso dos dados; também observa-se que a área possui poucas publicações no âmbito da Computação e Astronomia (ACCOMAZZI et al., 2012).

1.1 Motivação

O grande volume de informação astronômica deve receber atenção, pois com os investimentos nos projetos de telescópios e afins espera-se contrapartida cientificas, e se os dados não forem manipulados corretamente os problemas com localização da informação, armazenamento e análise serão evidentes. Existem diversos métodos que podem ser empregados para melhor interpretação humana, detecção automática de eventos como os filamentos ou outros eventos

solares. Contudo, diversas técnicas de processamento de imagem ou análise de imagem requerem um alto tempo de processamento. Se esse tempo for multiplicado pela grande quantidade de imagens, o tempo total de processamento (*wall time*) pode ser inviável. Unir algoritmos de processamento de imagem e técnicas de alto desempenho pode produzir análises e melhorias em imagens utilizando um tempo menor, tornando o processo viável. Unir as duas disciplinas abre caminhos para comparações e análises de grande volume de informação, em que a probabilidade de se encontrar fenômenos aumenta e ainda é possível explorar relacionamentos não triviais entre os dados.

A área que envolve a pesquisa combinada em Computação e Astronomia é considerada nova. Accomazzi et al. (2012) descreve a importância e necessidade de um periódico específico da área. Que a computação passou de apenas uma ferramenta coadjuvante, de enfoque puramente auxiliar, a um patamar essencial para as descobertas. O termo *astroinformatic* apresenta essa mudança de postura.

1.2 Hipótese

A hipótese que norteou este trabalho baseia-se na questão a seguir: há redução significativa no tempo de processamento de imagens solares com acurácia ao se utilizar algoritmos desenvolvidos especificamente para computação de alto desempenho em arquiteturas híbridas face a grande quantidade de dados astronômicos ?

1.3 Objetivo

O objetivo deste trabalho foi desenvolver um *software* para o processamento de imagens solares em arquiteturas de Processamento de Alto Desempenho (PAD) com sistemas híbridos, isto é, que dispõem de memória compartilhada e distribuída em uma única arquitetura computacional.

1.3.1 Objetivos específicos

Para atingir o objetivo geral do projeto, os seguintes objetivos específicos foram determinantes:

1. identificar métodos para a detecção de eventos solares e sistemas auxiliares na indexação, visualização a automatização;
2. identificar seções de códigos passíveis de paralelização, de acordo com os paradigmas escolhidos, nos algoritmos que efetuam as operações selecionadas;
3. desenvolver ou adaptar os algoritmos que efetuam as operações identificadas para utilização em ambiente de processamento de alto desempenho;
4. obter medidas quantitativas e qualitativas das imagens tratadas (comparações do desempenho da versão serial e paralela);
5. analisar os resultados das medidas obtidas.

1.4 Estrutura do texto

O Capítulo 2 descreve o formato FITS, mostra suas afinidades e compatibilidades com imagens astronômicas. O Capítulo 3 descreve as imagens de uso astronômico geral, as áreas e técnicas que são utilizadas no processamento de imagens astronômicas, elencando operações a serem aplicadas em imagens solares. As técnicas de alto desempenho direcionadas para o processamento de imagens astronômicas são abordadas no Capítulo 4. Tais técnicas são implementadas em um software para sistemas híbridos, descrito no Capítulo 5. Os experimentos do sistemas são descritos no Capítulo 6. O Capítulo 7 apresenta as conclusões da pesquisa descrita neste texto e apresenta algumas propostas para trabalhos futuros.

2 *Formato de imagem para astronomia*

As imagens astronômicas podem possuir duas ou mais dimensões espaciais e ainda contar com mais do que três dimensões espectrais (imagem multiespectral). Os instrumentos de captura podem obter diferentes comprimentos de onda de uma mesma região, revelando diversas informações diferentes para cada frequência. Para transportar e armazenar imagens astronômicas é necessária uma atenção especial aos arquivos, pois formatos regulares não têm suporte às características citadas. Além disso, é necessário um mapeamento de cada pixel para coordenadas espaciais. O formato mais utilizado para esses fins é o FITS (BERRY, 2005).

Neste capítulo, a Seção 2.1 explica dois formatos comuns para imagem; a Seção 2.2 mostra um formato específico para imagens astronômicas; e a Seção 2.3 mostra uma comparação entre os formatos.

2.1 *JPEG* e *TIFF*

Um formato popular de imagem é o *Joint Photographic Experts Group* – Grupo especialista em Imagem Articulada (JPEG). O nome formal é *JPEG File Interchange format* – Intercâmbio Arquivo Formato JPEG (JFIF) (WALLACE, 1992). Em linhas gerais, o processo de codificação consiste nos seguintes passos: (i) utilizar os componentes de cada cor separadamente; (ii) cada componente é tratado em blocos de 8×8; (iii) extrair a sua representação na frequência utilizando a *Discrete Cosine Transform* – Transformada Discreta dos Cossenos (DCT). Utilizando a informação obtida pela DCT é possível definir quais partes da imagem devem receber a compressão dos dados e por fim uma compactação baseada na codificação de *Huffman* (WALLACE, 1992).

Armazenar informações astronômicas em JPEG é um processo que pode ocultar informações importantes. Um exemplo de como essa limitação pode afetar diretamente a análise dos dados pode ser observado na Figura 2, que possui as galáxias *M81* e *M82* e foram obtidas das mesmas coordenadas espaciais, porém em frequências diferentes. A Figura 2-(a) possui um objeto que é visível na sua frequência de captura. Porém não é visível na 2-(b). Contudo, pode-se argumentar que a produção de uma imagem para cada canal tornaria factível a utilização do formato *JPG*.

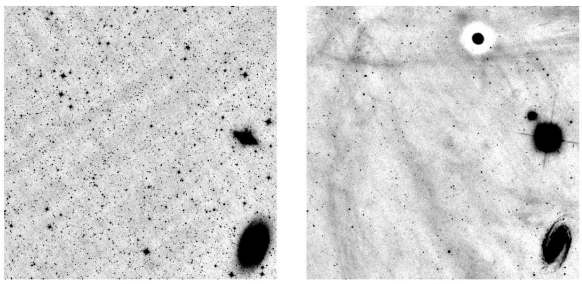

Figura 2: Imagem feita pelo Telescópio *WISE*, a Figura (a) utilizando $4,618\mu$m e a imagem (b) com $12,082\mu$m de comprimento de onda para a captura.

A abordagem para gerar uma imagem para cada canal poderia ser válida para o propósito de transmissão. Além da limitação da quantidade de dados, é o fato da própria compressão dos dados admitir pequenas perdas (compactação ou quantidade de bits por pixels limitada). Em vários casos alguns pequenos pontos possuem grande informação, como no caso da descoberta de uma das estrelas mais frias já observadas, as anãs Marrons Y, que possuem temperaturas próximas do corpo humano. A Figura 3 exibe esse conceito.

Figura 3: Imagem feita pelo Telescópio *WISE*, uma das estrelas mais frias já observadas (dentro do círculo branco, adicionado manualmente).

O *Tagged Image File Format* – Arquivos com formato de imagem marcado (TIFF) é utilizado em imagens astronômicas, pois é um formato muito popular na indústria de publicação e nas artes gráficas. As características do TIFF são: suporte a diversos sistemas de cor (i.e. *L*a*B*, *YCbCr*, etc); suporte a compressão com ou sem perdas e possibilidade de extensões e ainda conta com as informações extras nos cabeçalhos.

O processo de conversão para que uma imagem seja formatada em TIFF consiste em escolher quantos *bits* serão utilizados por canal e qual o sistema de cores. Um arquivo em TIFF pode possuir diversas imagens e ainda um conjunto de informações em seu cabeçalho. A Figura 4 dá um exemplo do cabeçalho de um arquivo codificado em TIFF.

Tag	Field	Count	Value	Comment
49 49				Byte order
2A 00				TIFF Version
08 00 00 00				Offset of first image file directory
0F 00				Number of tag entries (= 15)
FE 00	04 00	01 00 00 00	00 00 00 00	Tag 0FEh: New subfile type (0 = no)
00 01	04 00	01 00 00 00	XX XX XX XX	Tag 100h: Image width (= samples)
01 01	04 00	01 00 00 00	XX XX XX XX	Tag 101h: Image depth (= lines)
02 01	03 00	01 00 00 00	08 00 00 00	Tag 102h: Bits per sample (= 8)
03 01	03 00	01 00 00 00	01 00 00 00	Tag 103h: Compression (1 = none)
06 01	03 00	01 00 00 00	01 00 00 00	Tag 106h: Photometric interpretation (1 = grayscale)
11 01	04 00	01 00 00 00	C2 00 00 00	Tag 111h: Image offset (= 194)
15 01	03 00	01 00 00 00	01 00 00 00	Tag 115h: Samples per pixel (= 1)
16 01	04 00	01 00 00 00	XX XX XX XX	Tag 116h: Rows/strip (= samples)
17 01	05 00	01 00 00 00	XX XX XX XX	Tag 117h: Strip byte count = (Tag 100h * Tag 101h)
1A 01	05 00	01 00 00 00	XX XX XX XX	Tag 11Ah: X resolution offset = (Tag 111h + Tag 117h)
1B 01	05 00	01 00 00 00	XX XX XX XX	Tag 11Bh: Y resolution offset = (Tag 111h + Tag 117h + 8))
1C 01	03 00	01 00 00 00	01 00 00 00	Tag 11Ch: Planar configuration (=1)
28 01	03 00	01 00 00 00	02 00 00 00	Tag 128h: Resolution unit (= 2)
3D 01	03 00	01 00 00 00	01 00 00 00	Tag 13Dh: Predictor
00 00 00 00				Offset of next IFD (0 = none)

Figura 4: Cabeçalho do arquivo *TIFF* (BERRY, 2005).

As informações do cabeçalho garantem os dados básicos de como fazer o acesso às imagens que foram formatadas em TIFF. Os dados a seguir do cabeçalho, por sua vez, são gravados como um *byte stream* (Aldus Corporation, 1992). Se o arquivo contiver mais que uma imagem, antes de se iniciar a leitura dos pixels é necessário informar quantas imagens o arquivo contém e seus tamanhos. Porém, com todas as características o TIFF é um formato passível de leitura apenas por *software* (o JPEG compartilha tal característica). Além disso, a leitura ou gravação não são tarefas simples e obrigam ao programador utilizar algum tipo de coletânea de software para obter o acesso a essa imagem (BERRY, 2005).

2.2 *FITS*

Por volta de 1980, o Observatório *Kitt Peak National* e o *Nation Radio Astronomy Observatory* – Observatório Nacional de Rádio Astronomia (NRAO) iniciaram a utilização das imagens do *Very Large Telescope* (VLT). Esse telescópio produziu diversas imagens do céu em alta qualidade. Na mesma época, diversos telescópios pelo mundo iniciaram a produção de imagens digitais graças aos *CCDs*. O observatório de *Kitt Peak* e o NRAO tinham um projeto conjunto para a produção de um catálogo estelar: uma produção totalmente constituída de imagens digitais (BERRY, 2005).

Era necessário um formato de imagem intercambiável entre os observatórios, universidades e afins, além de uma forma de armazenar as imagens para que a leitura e a escrita fossem feitas de modo universal. Para dar força a esse impulso, na seção de Astronomia na Fundação Nacional de Ciência, em Janeiro de 1979, um grupo de cientistas propôs que um formato de imagem para astronomia deveria ter as seguintes características:

- grande flexibilidade (sobre a relação de imagens multiespectrais e multidimensionais);
- possuir espaço para uma autodocumentação.

Para resolver essa questão, deu-se início em 1981 ao FITS. De forma rápida, o FITS tornou-se o padrão de *facto* para transportar e armazenar informações astronômicas. Os maiores observatórios o utilizam (LBTC, 2011), (ESO, 2011), (IAC, 2011) e (WMKO, 2011). Além da *International Astronomical Union* – União Astronômica Internacional (IAU) e a NASA encorajam e endossam o uso desse padrão. Um ponto forte do FITS é sua descrição completa que define a representação binária dos dados (PENCE, W. D. et al., 2010). Outras informações que podem ser definidas são:

- tamanho das palavras chave (Metadados);
- descrições no cabeçalho da imagem;
- definições de quais unidades podem ser utilizadas com o formato;
- descrição de como devem ser formatadas as datas;

- suporte à compactação;
- suporte ao *World Coordinates System* – Sistema de Coordenadas Mundial (WCS) (GREISEN; CALABRETTA, 2002).

Essa última característica garante que sua informação possa ser comparada com outras fontes. Um exemplo é o sistema *Infrared Science Archive* – Arquivo Cientifico Infravermelho (IRSA) (CALTECH, 2010). A plataforma consiste em um *website* onde é possível buscar diversas imagens de diversas missões. As imagens podem ser recuperadas em FITS com as informações de localização de cada pixel. O FITS consiste em três componentes: o cabeçalho (*header*), os dados da imagem (*image data*) e o rodapé (*tailer*). A junção desses componentes é conhecida como FITS básico ou *single image FITS* – Imagem Única FITS (SIF) (PENCE, W. D. et al., 2010). As próximas seções explicarão os três principais componentes do FITS: o *FITS Header*, vetor de informações e o *tailer*.

2.2.1 *FITS Header*

O cabeçalho do *FITS* é composto por um conjunto de *Header Data Units* – Cabeçalho de Informação Principal (HDU). Tais unidades armazenam informações sobre a imagem codificada em *American Standard Code for Information Interchange* – Código Padrão Americano para o Intercâmbio de Informação (ASCII), bem como a quantidade de dimensões, quantidade de *bits* por canal (informações essenciais para a leitura da imagem) e informações complementares como origem da foto, etc (PENCE, W. D. et al., 2010). O tamanho dos arquivos deve ser múltiplo de 2880 *bytes*. Esse tamanho deve ser seguido para que o cabeçalho possa ser armazenado de forma integral em antigas fitas magnéticas (BERRY, 2005). Atualmente, o tamanho de discos rígidos (*hard disks*) e o preço por *Megabyte* é tão baixo que o tamanho dos cabeçalhos pode ser relevante em sistemas que devotam muito esforço para o alto desempenho. Se o sistema de arquivos estiver configurado e arranjado de forma a suportar um tamanho próximo a 2880 *bytes*, o acesso a tais imagens pode ter uma resposta mais rápida.

Cada linha do cabeçalho pode conter 80 *bytes*. De forma geral são colocadas 36 linhas. Cada um desses itens são chamados de *card image* (relacionado aos antigos cartões perfurados) (PENCE, W. D. et al., 2010). A Figura 5 exibe um exemplo de um cabeçalho de uma imagem codificada com o formato FITS, a estrutura mostra a relação *keyword* = *value* / *comment*.

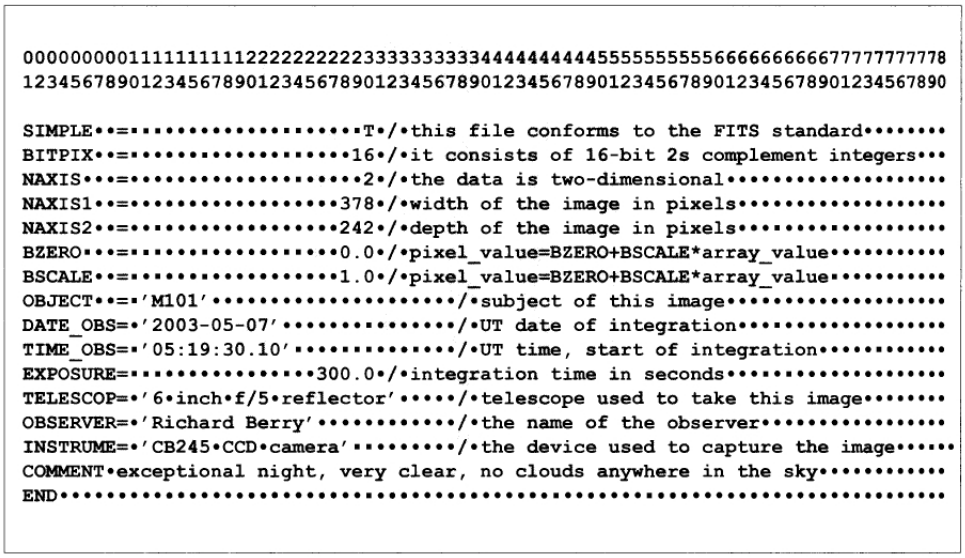

Figura 5: *FITS Header* (BERRY, 2005).

O tamanho máximo para as palavras-chave (*keyword*) é de 8 caracteres em caixa alta, alfanuméricos e um conjunto muito limitado de caracteres especiais (hífen e *underscore*), espaço em branco não é permitido. Algumas palavras-chave são obrigatórias, como as *SIMPLE*, *BITs per PIXel* – BITs por PIXel (BITPIX), *NAXIS*, *NAXIS1*, *NAXIS2* (até *NAXISn*) e *END* (BERRY, 2005).

A palavra-chave *SIMPLE* deve ser a primeira a ser encontrada no *header*, associada a um valor booleano (“T” para verdadeiro ou “F” para falso). Esse valor informará se o arquivo está formatado de acordo com o formato *FITS* ou se existe alguma desconformidade (FITS-OFFICE, 1997). BITPIX é preenchida por um valor inteiro e deve ser a segunda no HDU. Esse item informa a quantidade de *bits* por *pixel* na mesma imagem. O padrão especifica 6 possibilidades que estão na Tabela 1. Os valores negativos informam que será utilizado ponto flutuante a quantidade de bits é informada pelo valor absoluto do BITPIX.

NAXIS informa o número de eixos que o *array* de dados foi codificado. Para isso é utilizado um valor inteiro de 0 (sem dados) até 999. Para que o acesso aos dados seja feito de forma plena as palavras-chave *NAXIS1...NAXISn* informam o tamanho de cada eixo. E por fim, *END* apenas informa que o cabeçalho se encerra naquela posição. Existem diversas palavras chaves que são utilizadas para melhor descrever o conteúdo. A palavra chave *BSCALE* é utilizada juntamente com a chave *BZERO* para recuperar o valor físico de um pixel (PENCE, W. D. et al., 2010). Esse valor físico representa o número que o instrumento de captura gerou.

Tabela 1: Possíveis valores para *BITPIX* (PENCE, W. D. et al., 2010).

BITPIX	Tipo	Intervalo
8	8-bits sem identificador de sinal	0 até 255
16	16-bit com identificador de sinal (complemento de dois)	-32768 até 32767
32	32-bit com identificador de sinal (complemento de dois)	-2147483648 até 2147483647
64	64-bit com identificador de sinal (complemento de dois)	-9223372036854775808 até 9223372036854775807
-32	32-bit *IEEE* ponto flutuante	1.2×10^{-38} até 3.4×10^{38}
-64	64-bit *IEEE* ponto flutuante	2.2×10^{-308} até 1.8×10^{308}

Com as informações de *NAXIS*, *NAXIS1...NAXISn* e *BITPIX* já é possível acessar a imagem de forma simples. Com as informações de *BZERO* e *BSCALE* é possível mapear para os valores que foram produzidos pelos instrumentos de captura. Informações extras também fazem parte no cabeçalho do *FITS*. *OBJECT* consiste em uma *string* com o nome do objeto que está sendo observado (*i.e. M81 e M82 Galaxy*). *TELESCOP* descreve o nome do telescópio que foi utilizado para capturar a imagem. (*e.g. VLT*). *INSTRUME* contém a descrição do instrumento utilizado no telescópio para produzir a imagem (*e.g. CCD*). *OBSERVER* descreve o nome do autor ou do sistema que produziu a imagem.

2.2.2 Vetor de informação

A imagem propriamente dita é armazenada em uma estrutura denominada "*data array*". Tal vetor é interpretado como um *byte stream* (PENCE, W. D. et al., 2010). A Figura 6 ilustra como essa informação é organizada.

```
A(1, 1, . . . . , 1),
A(2, 1, . . . . , 1),
.
.
.,
A(NAXIS1, 1, . . . . , 1),
A(1, 2, . . . . , 1),
A(2, 2, . . . . , 1),
.
.
.,
A(NAXIS1, 2, . . . . , 1),
.
.
.,
A(1, NAXIS2, . . . . , NAXISm),
.
.
.,
A(NAXIS1, NAXIS2, . . . . , NAXISm)
```

Figura 6: Organização da informação em arquivo regular codificado em FITS. Adaptada de (PENCE, W. D. et al., 2010).

Para acessar a informação da imagem é possível a utilização de diversas bibliotecas para uma gama de linguagens de programação. As principais linguagens suportadas são: C, *C#* (.Net), C++, *Fortran*, *IDL*, *IGOR Pro*, Java, *LabVIEW*, *Mathematica*, *MatLab*, *Perl*, *Photoshop plug-in*, *Python*, R e *TCL.*

2.3 Comparação de formatos para uso astronômico

Considerando o cenário de dados astronômicos/científicos, existem algumas características que são necessárias para um bom sistema de armazenamento: flexibilidade no número de bits por canal; suporte a imagens multidimensionais; cabeçalho com vasta quantidade de informação e passível de leitura por humanos e computadores; padronização entre observatórios, astrônomos, etc e a leitura dos dados não ser demasiadamente complexa ou essencialmente atrelada a uma biblioteca.

Sobre a quantidade de *bits* por *pixels*, o FITS tem suporte a seis tipos diferentes (sem dependência ao sistema de cores). O JPEG ou TIFF tem relações diretas ao sistema de cor e ainda uma quantidade limitada de bits por canal. O FITS pode suportar até 999 eixos de dados em apenas um arquivo. O JPEG e TIFF podem suportar apenas os 2 eixos tradicionais. A "leitura" do FITS pode ser feita facilmente com um editor de texto pleno. Um dos pontos mais importantes que o FITS é endossado pela NASA e pela IAU e diversos observatórios utilizam esse formato para armazenar suas imagens. TIFF ou JPEG ainda são utilizados, mas de forma geral apenas para exibir as imagens. A Tabela 2 resume as principais vantagens do FITS sobre os outros formatos.

Tabela 2: Comparação entre formatos de imagem.

Característica	*FITS*	*JPEG*	*TIFF*
flexibilidade no número de bits por canal	de 8 até 64 bits (com possibilidade de ponto flutuante)	dependente do sistema de cor	dependente do sistema de cor
Quantidade de eixo destinado a dados	0 a 999 eixos	2 eixos	2 eixos
cabeçalho com vasta quantidade de informação e passível de leitura por humanos e computadores	quantidade limitada, porém padronizada	possibilidade de comentários	possibilidade de comentários
padronização entre observatórios, astrônomos, etc	padrão de *facto*	utilizada para visualização	utilizado para visualização e impressão
leitura dos dados não ser demasiadamente complexa ou essencialmente atrelada a uma biblioteca	a leitura pode ser feita de forma descomplicada por funções básicas de entrada/saída	atrelada a biblioteca	atrelada a biblioteca

2.4 Considerações

Este capítulo mostrou o quanto o formato utilizado para representar imagens astronômicas é importante. Em diversos casos, poucos *pixels* podem representar a detecção de diversos eventos. Se utilizado um formato incompatível, existem diversas chances de perder informações. O FITS é o padrão de armazenamento mais indicado para o trato dos dados astronômicos garantir que será mantida a riqueza da informação. Os formatos JPEG e TIFF, por exemplo, são adequados somente para a exibição de imagens.

3 *Imagem digital*

Cada projeto de telescópios (seja terrestre ou espacial) tem um custo muito alto e, por conta disso, deve se converter em resultados satisfatórios. Como todos são equipados com CCD, garante que o estudo de diversos objetos que antes não eram passíveis de observação por possuírem um tamanho reduzido ou pouco brilho.

Neste capítulo, a Seção 3.1 aponta quais áreas de processamento de imagem e análise de imagem possuem aplicabilidade direta e específica em imagens astronômicas. A Seção 3.2 apresenta um levantamento de quais técnicas, algoritmos e filtros são mais utilizados em *software* para processar imagens astronômicas. A etapa descrita na Seção 3.3 mostra a aplicabilidade de operações com *kernels*. Os processos para tratamento do contraste e escala de *pixels* para as imagens são descritos na Seção 3.4. A detecção de filamentos solares é descrita na Seção 3.5 e a detecção de áreas mais brilhantes do Sol é descrita na Seção 3.6.

3.1 Aplicações de processamento de imagem em astronomia

Algumas aplicações ou áreas do processamento/análise de imagem podem ser muito bem utilizadas para tratar dados astronômicos. Tais áreas são (STARCK; MURTAGH, 2006):

1. visualização - Técnicas para exibir as informações de forma adequada dada a natureza dos dados ou ainda transformações que evidenciem os dados (MUELLER et al., 2009);

2. filtragem - Filtragem dos sinais enviados pode ser utilizada para melhorar a visualização ou ainda segmentar certos tipos de objetos para um melhor estudo (STARCK; MURTAGH, 2006);

3. deconvolução - Transformações inversas para retirar as áreas “embaçadas” da imagem. Tais processamentos geram resultados que melhoram as imagens (STARCK et al., 2002) e (NUNEZ et al., 2009);

4. compressão - Sondas enviadas para o espaço devem utilizar algoritmos que consigam minimizar o tamanho dos arquivos sempre preocupados com a qualidade (RUEFFER; JASPERS, 2007) e (BOBIN et al., 2008);

5. segmentação e reconhecimento de padrão - A detecção de padrões é muito útil em diversas pesquisas como, por exemplo, a detecção de grandes explosões em estrelas (super novas) (BARRA et al., 2007) e (CRISTO et al., 2008);

6. transformadas - Transformadas de forma geral (*Fourier* e *Wavelet*, entre outras). Tais operações geram grandes ganhos no âmbito do processamento de imagens para astronomia por propiciar a interpretação das informações contidas de formas diferentes (ANDS J. ANDS et al., 2010) e (LAVANYA et al., 2011).

3.2 *Software* e aplicações para processamento de imagens astronômicas

Existe um conjunto de *software* para processar imagens astronômicas. Alguns possuem funções que abrangem controles de observatórios e outros apenas são focados no processamento das imagens já obtidas. A Figura 7 mostra a quantidade das operações disponíveis nas ferramentas: *MaxIm DL 5*, *PixInsight*, *ImagesPlus*, *AIP4Win*, *Nebulosity2*, *Iris*, *Astroart*, *Registax*, *AstroStack 3*, *K3 DS9* e *Image Reduction and Analysis Facility* – Ambiente para Redução e Análise de Imagem (IRAF). O gráfico da Figura 7 mostra a agregação das funcionalidades que os *software* pesquisados possuem.

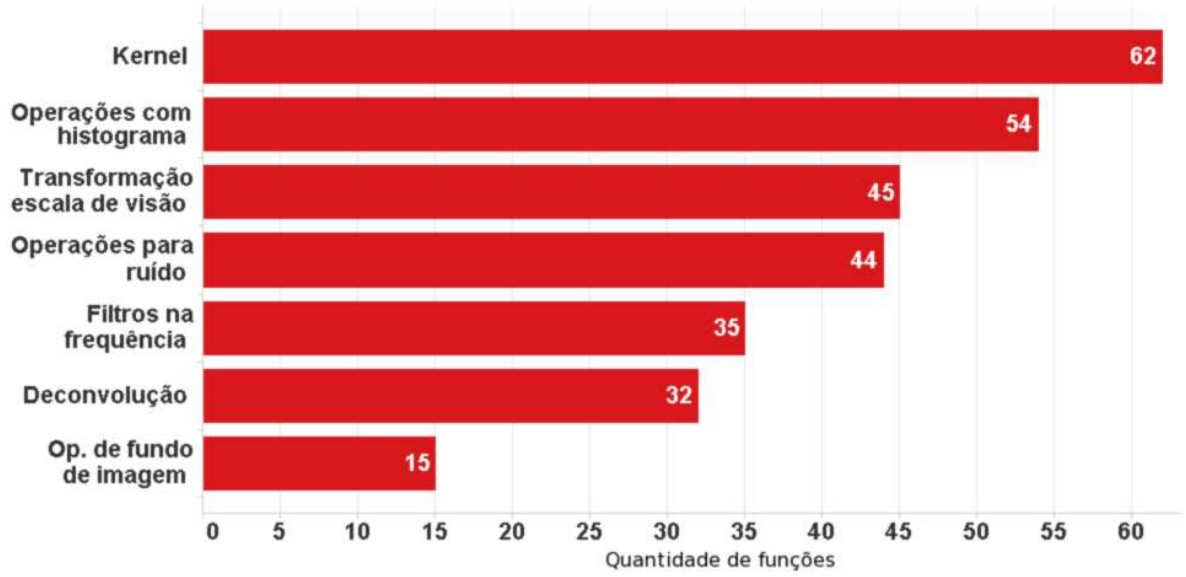

Figura 7: Gráfico com as quantidades das funções de 14 ferramentas para processamento de imagem para astronomia.

Processos para a detecção de eventos não foram encontrados. Operações com *Kernels* (tais como Gaussianas, detecção de bordas e afins) têm grande popularidade, pois propiciam efeitos tanto de remoção de ruído quanto técnicas para melhorar o contraste. Transformação de histograma e transformação de escala de *pixels* têm grande apelo ao uso, pois diversas imagens possuem escalas diferentes das utilizadas nos monitores ou ainda não possuem contraste razoável para serem exibidas. Porém, outro grupo muito importante e representado por uma quantidade menor de filtros são aqueles classificados como os de redução de ruído. Tais interferências podem ter diversas fontes e cada uma delas têm tratamentos diferentes.

Dois *software* são detalhados a fim de exemplificar o funcionamento geral de tais ferramentas. Um deles é o *MaxIm DL*, produzido pela *Diffraction Limited*, que possui suporte para diversos formatos de imagem. Desde os mais comuns, como JPEG ou *Graphics Interchange Format* – Formato Intercambiável Gráfico (GIF) e afins, até o próprio FITS. A Figura 8 mostra a interface do *software* processando uma imagem do Sol.

Além de fornecer suporte a diversos arquivos, o *MaxIm DL* possui diversos filtros para melhorar e recuperar imagens. Tais filtros comportam funções como remoção de ruídos, remoção de *Bloom* (o excesso de brilho de estrelas por exemplo, que causam efeitos de espalhamento do brilho), remoção do efeito de *pedestal* (corrigir os limites dos *pixels* da imagem). Controles

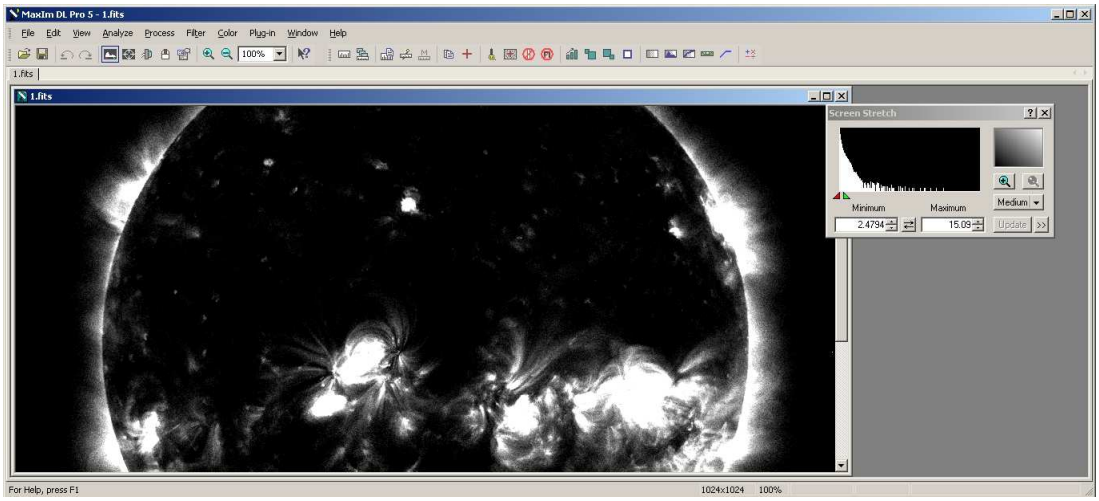

Figura 8: Interface do *MaxIm DL*.

de contraste baseados em extensão de histograma e especificações de histograma, *threshold* e operações de transformação de escala fazem parte do grupo de operações.

Existe um conjunto de filtros disponíveis *MaxIm DL*, alguns deles baseados em *kernels*. São eles: (i) passa-Alta e passa-baixa; (ii) remoção de *dead pixel*; (iii) remoção de *hot pixel*; (iv) média; (v) moda; (vi) dilatação e erosão; (vii) filtro Gaussiano; e (viii) interface para escolher um *kernel* personalizado. Os filtros baseados em frequência utilizam a transformada de *Fourier* (passa alta e baixa frequências) e *Wavelet*.

Outros filtros amplamente utilizados são *sharpening*, *rank filter* (para controlar o contraste), o filtro local adaptativo (para melhora do contraste), filtro de gradiente rotacional (para melhora de detalhes e contraste), deconvolução (para remoção e controle do ruído) e, por fim, funções para remoção e fusões do fundo da imagem. Há funções para fusões de sistemas de cores com o objetivo de produzir imagens de um mesmo objeto, utilizando frequências diferentes.

Outro *software* para processamento de imagens astronômicas é o *PixInsight* que é descrito como um programa *avançado* e modular, produzido pela *Pleiades Astrophoto*. A Figura 9 mostra a interface do *software* processando a imagem das galáxias M81 e M82.

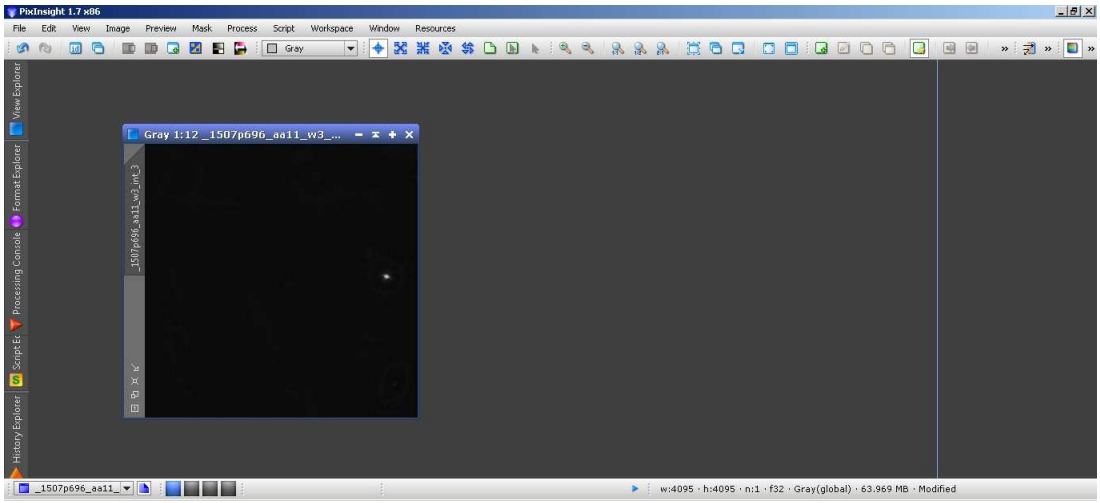

Figura 9: Interface do *PixInsight*.

O *PixInsight* possui interface de texto para executar os comandos, múltiplas visões e um grande conjunto de operações, além de suporte ao FITS e diversos outros arquivos. O *PixInsight* tem funções parecidas com as do *MaxIm DL*. Porém, possui alguns processamentos peculiares, tais como: (i) *ACDNR* (para redução de ruído), (ii) funções para registro de imagem e (iii) funções para diversos espaços de cores.

Alguns desses *software* possuem capacidade para funcionar em máquinas com diversos núcleos. Porém, em nenhum dos casos foi observado métodos para o processamento em memória distribuída. Como foi possível verificar no Capítulo 1, o volume de informação gerado de imagens é muito grande e o processamento apenas em abordagens de memória compartilhada pode não ser suficiente, já que diversas arquiteturas possuem limites de memória (*slots* físicos para os *chips* ou impedimentos lógicos para o controle). Dessa forma, uma abordagem com sistemas híbridos se faz necessária para que os limites locais de um computador não se tornem limites globais da aplicação.

A seção a seguir descreve as operações de maior relevância e as operações em imagens que foram identificados como as de maior ocorrências nos *software* para processamento de imagens astronômicas.

3.3 Operações com *Kernels*

As operações com *kernels* ou convoluções representam um mecanismo matemático que utiliza dois sinais para gerar um terceiro sinal. Isto modifica a imagem de uma forma específica, por exemplo, detectando variações abruptas de contraste (JÄHNE, 1997). Em uma abordagem mais formal, F representa um filtro de convolução e I é uma imagem. Assim assume-se a Equação 1.

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \tag{1}$$

As operações de convolução, podem ser aplicadas tanto na frequência (utilizando transformada de *Fourier* ou *Wavelets*) ou de forma espacial (diretamente nos *pixels*). Nesta etapa do trabalho, abordam-se apenas as operações no espaço. Técnicas mais clássicas a serem aplicadas com os *kernels* são as operações de média (moda e média utilizam o conceito dos vizinhos da convoluçã; Porém, não a operação de forma direta), detecção de bordas (*Sobel*, *Roberts*, etc) entre outras diversas. Um exemplo simples de como são os *kernels* e quais são seus resultados é o filtro de *Sobel*. Esse filtro tem a capacidade de realçar as bordas de uma imagem. Na Figura 10 pode-se ver o resultado da aplicação de tal filtro.

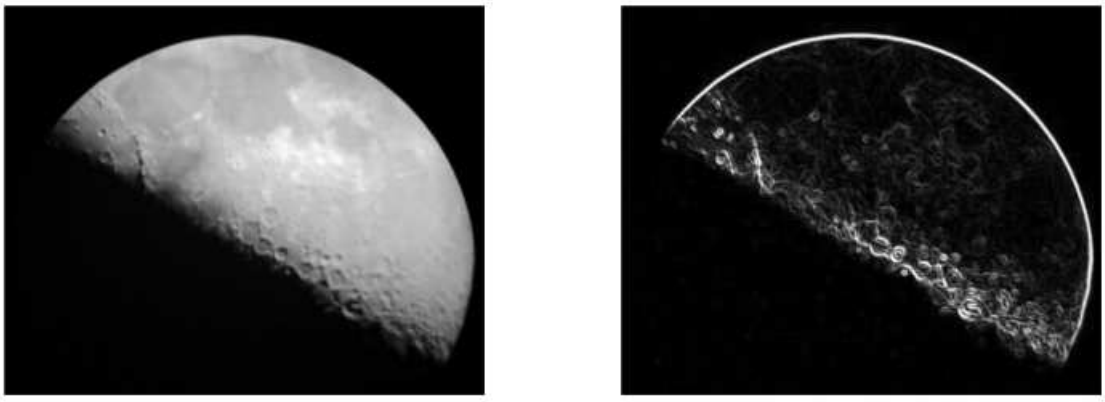

Figura 10: (a) Imagem capturada da Lua. (b) Detecção de bordas.

O filtro *Sobel* utiliza dois *kernels* que podem ser aplicados de forma simultânea na imagem e serem combinados para produzir bordas em todas as direções. Tais *kernels* são representados pelas matrizes na Equação 2.

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2}$$

Os filtros de detecção de borda são utilizados para realçar características que sejam convenientes em processos de segmentação ou a fim de melhorar a visualização. A detecção de crateras planetárias que foi desenvolvida por Troglio et al. (2010) além de utilizar modelos estatísticos, utiliza os filtros de detecção de borda, filtros *Gaussianos* e filtros de média. Outra operação que utiliza operações com *kernels* é um processo de segmentação de céu feito por Qin et al. (2010) que utiliza filtros Gaussianos e filtros de média em uma imagem que contém apenas o fundo, para que ruídos não influenciem na detecção e segmentação da imagem.

A operação de convolução por utilizar os pixels vizinhos podem indicar como é o comportamento de certa região da imagem. Com isso pode-se estender a ideia dos vizinhos da convolução para aplicar filtros mais específicos. Os filtros de difusão utilizam a convolução. Tais filtros têm características parecidas com a convolução. Porém, uma abordagem consiste em utilizar janelas circulares, algoritmos de classificação e a própria função *Gaussiana*. A convolução tem característica para a filtragem de um sinal. Contudo, em diversos casos é necessário obter uma visão mais global da imagem e assim aplicar algum tipo de operação.

3.4 Contraste e transformações de escala

Com o formato FITS é possível armazenar imagens com quantizações variadas. Isso garante que a informação capturada será armazenada de forma coerente, mantendo toda a riqueza dos dados obtidos. Entretanto, para a visualização por monitores, que em sua maioria conseguem apenas mostrar 8 *bits* por canal, isso se torna um problema.

Utilizando as transformações de escala e afins é possível efetuar alterações na variação da imagem transformando de 32 *bits* para 8 *bits* por canal. Dessa forma ocorre uma perda de informação. Contudo, a imagem ganha uma visualização mais confiável ao olho humano. Uma

figura codificada em FITS, utilizando 32 *bits* por canal, com um tamanho de 1024×1024 *pixels*, obtida pelo SDO, capturada em um comprimento de onda de 94 Å tem apenas alguns pontos visíveis. Porém, com uma mudança de escala para 8 *bits* é possível a visualização de diversos pontos, como na imagem da Figura 11. A compressão de escala baseado na função logarítmica (Equação 3 e 4) já possibilita uma visualização com maiores detalhes dos pontos onde o Sol está com maior atividade. A transformação é dada por (GONZALEZ; WOODS, 2007):

$$s = c\ log(1 + |r|) \tag{3}$$

$$c = H/log(1 + |R|) \tag{4}$$

Onde s é o resultado para cada *pixel* (aplicado em cada canal separadamente) na imagem, r é valor original, c é a constante da escala e, por fim, R que representa o maior número possível da nova escala. Tal transformação simples aplicada na imagem gera um resultado interessante que é apresentado na Figura 11. Aplicada a transformação, já é possível visualizar diversos detalhes que não eram apresentados na imagem original. Todavia, existem diversos eventos no Sol que devem ser monitorados. Um deles é a aparição de filamentos.

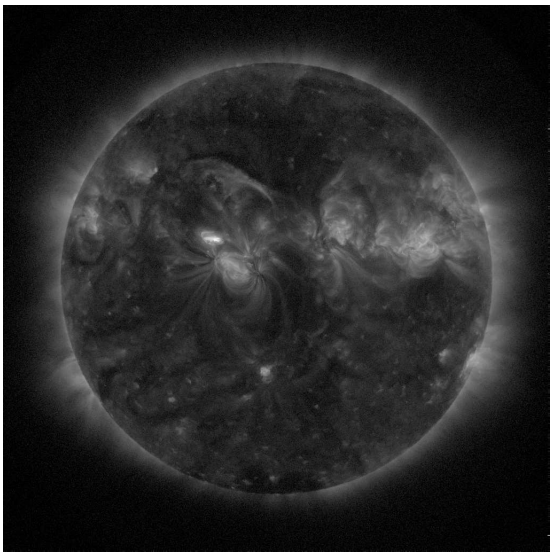

Figura 11: Imagem do Sol obtida do *SDO* processada com a transformação logarítmica. A imagem original apresentava apenas pixels tendendo a zero.

3.5 Detecção de filamentos

O Sol é uma estrela anã, categorizada como corpo central do Sistema Solar, que produz grande quantidade de energia que chega na forma de luz à Terra. Essa energia é de suma importância para o suporte a vida. Porém, mesmo com a distância aproximada de 150 milhões de quilômetros (RIDPATH, 2007), eventos extremos que acontecem em sua superfície podem prejudicar diversos sistemas terrestres.

Tais eventos podem ser classificados como: *Sunspot* (manchas solares), *plages*, *faculae*, flares (explosões), filamentos e condensação coronal (RIDPATH, 2007). Áreas no Sol que possuem esses eventos são conhecidas como regiões ativas (JENKINS, 2009). Detectar os filamentos é de grande importância, pois são relacionados as CME. Esses eventos podem afetar a transmissão de energia, sistemas de telecomunicações e transporte aéreo, entre outros.

3.5.1 Detecção baseada em grafos

O algoritmo de detecção de filamentos solares utilizado foi descrito em (GAO et al., 2002). Este método foi melhorado ao adicionar um passo de pré-processamento que compreende o filtro de difusão e um passo de pós-processamento para associar diferentes tons de cinza para cada filamento. Ambos os passos, bem como o algoritmo de detecção de filamentos, foram desenvolvidos com a abordagem de programação paralela. O método, chamado de *Filamento Diffusion Detection* (FDD), está representado na Figura 12. Os passos em cinza indicam as etapas adicionadas.

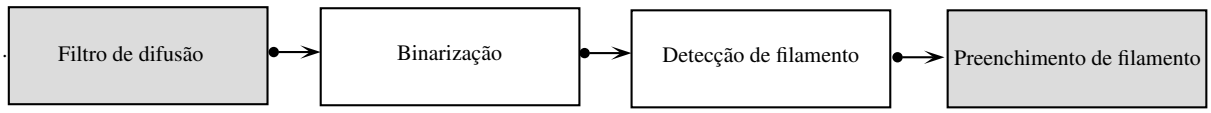

Figura 12: Diagrama do método *FDD* (ANDRIJAUSKAS; GRADVOHL, 2012).

O processo de segmentação consiste em aplicar um limiar para evidenciar os filamentos. Esse limiar é aplicado na etapa de "Binarização" Figura 12. O valor escolhido consiste na metade da média da intensidade dos *pixels* (GAO et al., 2002). Os filamentos apresentam-se nas imagens Hα de maneira peculiar. Possuem temperatura muito menor que do Sol, com isso indicados por *pixels* escuros. O passo seguinte consiste em visitar os vizinhos espaciais

de cada *pixels* de forma recursiva com o objetivo de verificar quais estão conectados (GAO et al., 2002). Uma vizinhança circular foi escolhida dado a natureza do problema. Após essa caracterização, o tamanho e distância entre os filamentos é avaliado com o objetivo de verificar se aquele filamento é relevante ou se deve ser agrupado com outro. O último passo consiste na atribuição de diferentes tons de cinza para cada filamento.

Para evitar o ruído nas imagens e maximizar a operação de binarização, um filtro de média ou filtro Gaussiano pode ser aplicado. Contudo, esses filtros podem atenuar bordas na imagem, o que não é interessante. Para evitar esse problema pode-se utilizar um filtro de difusão.

3.5.2 Filtro de difusão

No processo de segmentação dos filamentos é muito interessante que o interior dos filamentos e as outras partes do Sol estejam o mais uniforme possível e as bordas bem definidas. Ao tentar aplicar um filtro Gaussiano ou um filtro de média, além de suavizá-las eles deterioram as bordas. Para resolver esse problema pode-se utilizar o filtro de difusão.

Considerem-se os vértices como os *pixels* da imagem e seus arcos uma ligação levando em conta posições relativas da imagem ou propriedades por *pixel*, tais como brilho e outras. É possível desenvolver um filtro que consiga suavizar o interior de objetos e ainda manter as bordas. A base desse filtro de difusão foi obtida da *Image Foresting Transform* (FALCAO et al., 2004).

Na Equação 5, p é o *pixel* de origem e q é adjacente a p, sendo que $d(p,q)$ é a distância entre os dois nós e t é um limiar que delimitará se o *pixel* q e adjacente a p. Essa distância pode ser espacial ou paramétrica. O filtro de difusão utiliza uma adjacência circular espacial. A Figura 13 demonstra como essa vizinhança foi definida dado um *pixels* central (*pixel* central mais claro). A distância euclidiana foi utilizada para construir os vizinhos e para calcular a distância entre os *pixels*.

$$(p,q) \sim d(p,q) <= t \tag{5}$$

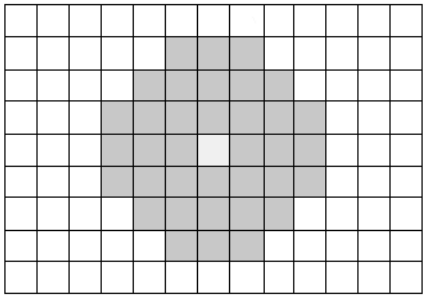

Figura 13: Vizinhança circular dos pixels.

Dados os *pixels* vizinhos, são utilizados apenas os k mais próximos considerando agora uma distância paramétrica. Para os k mais próximos, faz-se necessária a ordenação do vetor. Nesse caso, o valor dessa distância dado pelo módulo da diferença do valor do *pixel* adjacente e do *pixel* central. Após ordenar o vetor de distância paramétrica é feita uma convolução - Equação 6 - utilizando apenas os k mais próximos.

$$J[x_p, y_p] = \sum_{i=0}^{k} w[i] I[x_p + dx[i], y_p + dy[i]]. \tag{6}$$

A convolução dos *pixels* mais próximos é feita com um *kernel* gaussiano (w). Para gerar os coeficientes do filtro Gaussiano, percorre-se a adjacência do *pixel* p, e para cada um dos deslocamentos (dx e dy representam os deslocamentos nos eixos x e y, respectivamente) que geram os q adjacentes, aplica-se a Equação 7.

$$w(q-p) = \exp\left(\frac{-(|dx+dy|^2)}{2\sigma^2}\right). \tag{7}$$

Um exemplo do resultado do filtro de difusão aplicado está na Figura 14, onde é possível observar que a imagem está mais homogênea, porém suas bordas ainda estão bem definidas.

A Figura 15 apresenta o resultado de uma classificação com o *FDD*. Neste método é possível observar tons de cinza diferentes para cada filamento.

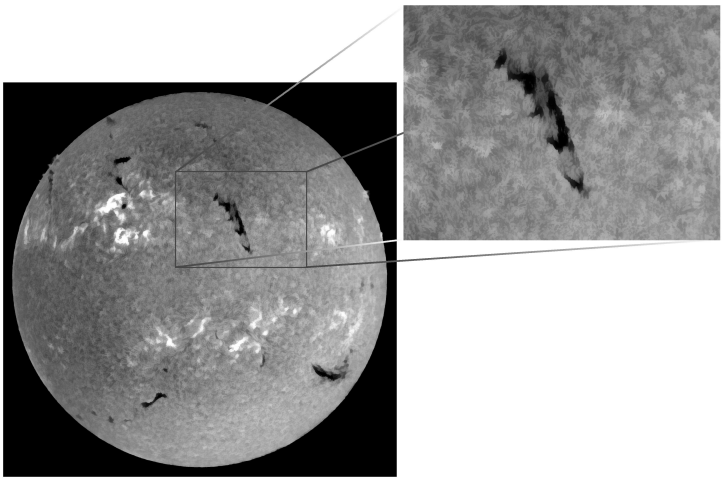

Figura 14: Imagem processada com o filtro de difusão. O Filamento foi evidenciado manualmente.

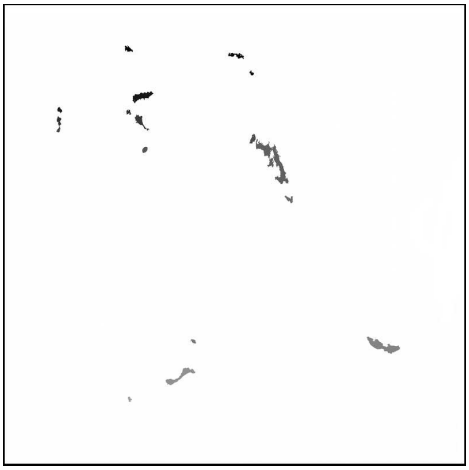

Figura 15: Filamentos solares detectados com o método *FDD*. Os valores dos *pixels* foram invertidos para melhor visualização.

3.5.3 Operadores morfológicos

O método *Morph Detection* (MD) é baseado em operadores morfológicos. O diagrama da Figura 16 descreve como esse método é composto.

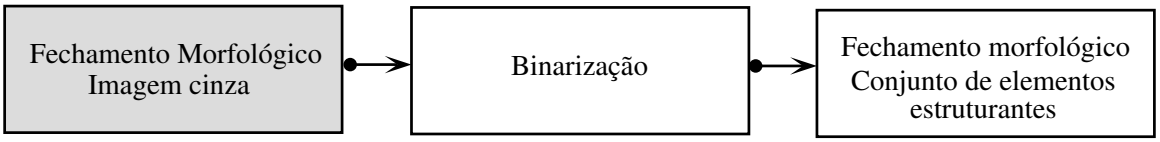

Figura 16: Diagrama do Método MD.

A morfologia matemática é aplicada no processamento de imagens em diversas áreas (GONZALEZ; WOODS, 2007). Diversas técnicas podem ser aplicadas para realce, filtragem ou detecção de objetos. Foram utilizadas duas operações morfológicas neste método. A primeira consiste em um pré-processamento. Executa-se quatro vezes o fechamento morfológico na imagem em escala de cinza como um pré-processamento para fechar pequenos orifícios na imagem.

O elemento estruturante utilizado é de tamanho 11×11 *pixels* em que todos os elementos são o número 1. Após transformar a imagem em binária com um liminar que é metade da média da intensidade dos *pixels* (GAO et al., 2002) é aplicado um conjunto de operadores morfológicos na imagem binária. Esse conjunto de operadores é descrito em (SHIH, 2010). Na Figura 17 há um exemplo desses elementos.

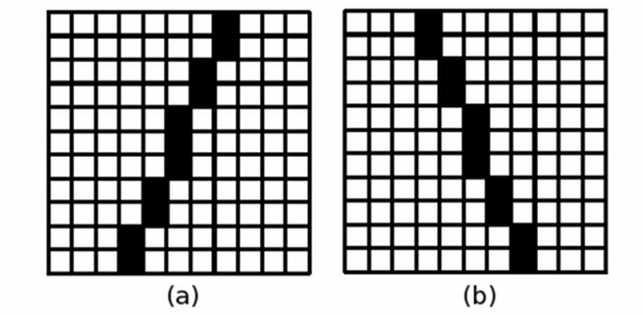

Figura 17: Exemplos do conjunto de elementos aplicados (SHIH, 2010).

A Figura 18 descreve um exemplo do filtro de pré-processamento aplicado. A imagem do Sol tem menor variação. Assim, a fase de binarização pode ser realizada com maior precisão. No entanto, algumas áreas do filamento são obliteradas pelo processo. A Figura 19 mostra os filamentos detectados pelo método MD.

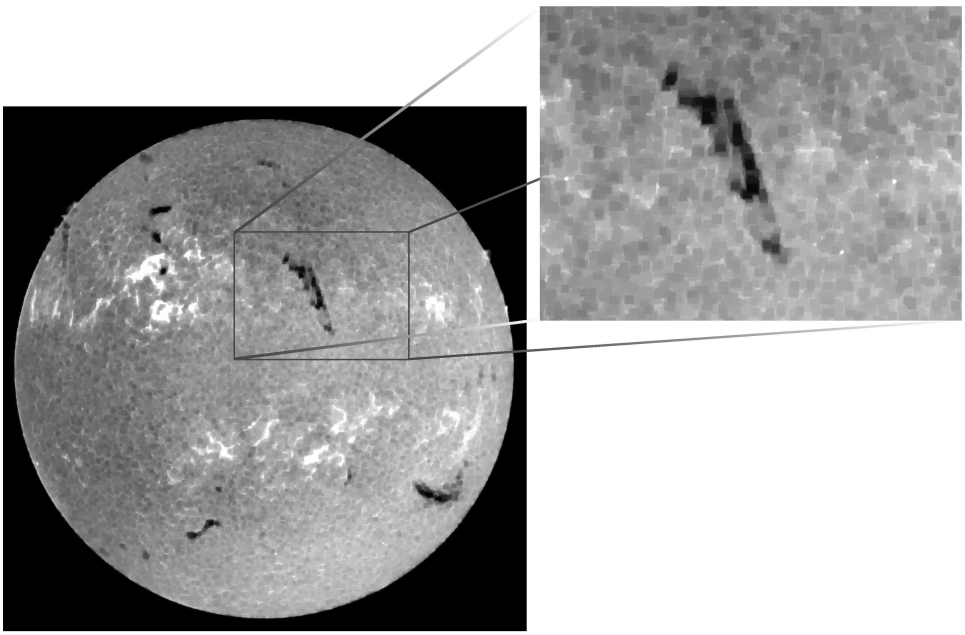

Figura 18: Imagem processada com o pré-processamento do método MD. O Filamento foi evidenciado manualmente.

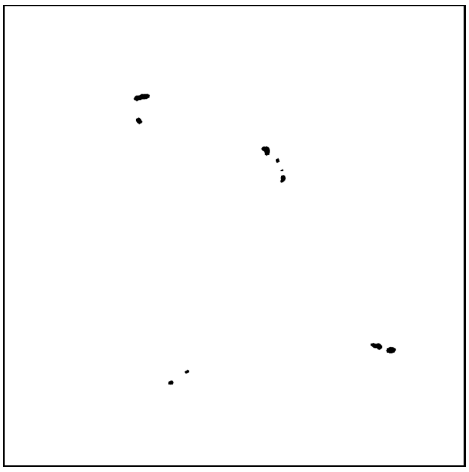

Figura 19: Filamentos solares detectados com o método MD. Os pixels foram invertidos para melhor visualização.

3.6 Detecção das áreas mais brilhantes por segmento

A detecção das áreas mais brilhantes é de grande utilidade para a melhor visualização de diversos eventos. *Flares* e *faculaes* são representadas por áreas de grande brilho e ambos os eventos estão ligados direto ou indiretamente ao CME. Por esta razão, seus estudos são de grande importância (JENKINS, 2009).

O método de detecção dessas áreas é baseado em janelas de pixels denominado *HArea*. É utilizada uma vizinhança espacial ao redor de cada pixel e avalia-se como é o somatório dessa região. A Equação 8 apresenta esse processo. Onde T é a soma dos valores do pixels, w e h representam respectivamente a largura e altura da janela, I é a imagem e dx e dy são os deslocamentos. Essa função é aplicada para cada pixel de um segmento horizontal da imagem. O usuário pode escolher quantos segmentos a imagem deve ter.

$$T = \sum_{i=1}^{w \times h} I[x_p + dx[i], y_p + dy[i]]. \tag{8}$$

Aplicada a Equação 8, é necessário escolher o maior valor encontrado. A Equação 9 escolhe o maior entre todos os somatórios, sendo que $\vec{T}$ representa a coleção dos somatórios para cada pixel e T_l o maior valor.

$$T_l = max(\vec{T}) \tag{9}$$

A Figura 20 demonstra o processo aplicado em uma imagem do Sol. As áreas que estão na imagem representam aquelas com os maiores valores no somatório.

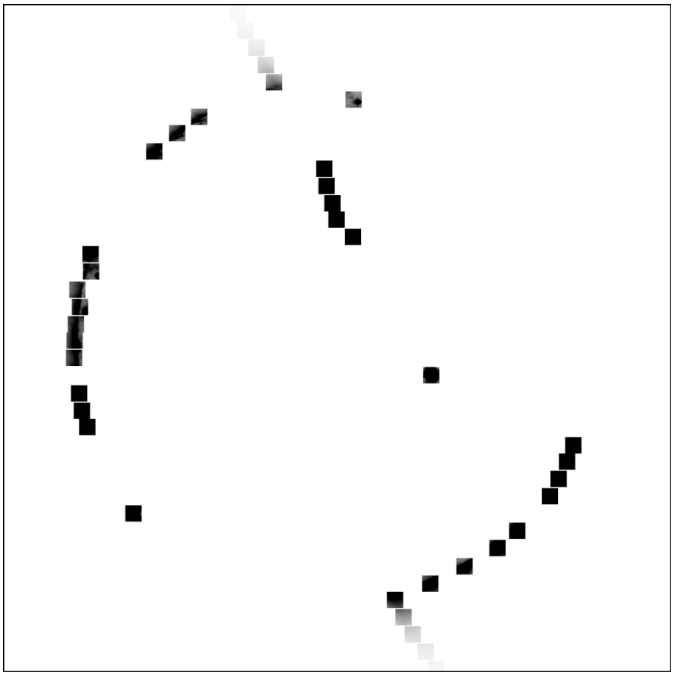

Figura 20: Detecção das áreas mais brilhantes do Sol. Os pixels da imagem foram invertidos.

3.7 Considerações

O processos de imagem utilizados na astronomia são de grande importância para viabilizar a observação. Processos simples como a compressão de escala de *pixels* já podem evidenciar informações para um usuário. Mas é necessário utilizar mais informação além do que o usuário pode ver. Processos de detecção em que o computador utiliza toda a gama de informação disponível também são de grande importância como a detecção de filamentos ou a detecção de áreas mais brilhantes.

4 *Processamento de alto desempenho*

A informação colhida para uso astronômico consiste, em grande parte, de imagens digitais. Essas imagens possuem alta resolução espacial e espectral. Para processar todo esse conteúdo independente dos objetivos - podendo variar de tratamentos para melhorar a visualização até procurar por exoplanetas - computadores domésticos não possuem a capacidade necessária para tamanho processamento.

Os computadores de uso específico para processamento de alto desempenho, de forma geral, podem utilizar duas abordagens para obter um poder computacional superior: (i) aumento do *clock* dos processadores, capacidade de memórias e afins (novas arquiteturas); ou (ii) dividir o processo em diversos computadores (ou diversos núcleos) (HERLIHY; SHAVIT, 2008). A primeira forma não requer grande esforço dos programadores, pois as instruções dos programas executarão de forma mais rápida. Porém, diversos limites (físicos e técnicos) mostram-se grandes demais para serem transpostos em um tempo hábil. Por essa razão, a segunda forma mostra-se mais plausível. Todavia, requer um esforço maior dos seus idealizadores para: analisar as possibilidades de paralelismo; projetar e implementar; testar a corretude; e ajustes de desempenho (BRESHEARS, 2009).

O ambiente que comporta um sistema de alto desempenho para processar as imagens astronômicas consiste em computadores interligados por uma rede de alta velocidade. Cada computador receberá uma parte da tarefa, assim utilizando os recursos computacionais de maneira mais abundante, com o objetivo de reduzir o tempo total de processamento.

Neste Capítulo, a Seção 4.1 apresenta os requisitos de um ambiente para processamento distribuído; a Seção 4.2 descreve a programação em memória compartilhada; os ambientes em memória distribuída são descritos na Seção 4.3; e uma junção dos dois modelos, um sistema híbrido, é descrito na Seção 4.3.

4.1 Ambientes de processamento de alto desempenho para imagens astronômicas

Em um ambiente de processamento de alto desempenho, sua arquitetura de *software* ou de *hardware* deve estar intimamente ligada ao tipo de dado que se quer processar. Em diversos cenários, o *hardware* não pode ser modificado ou especializado para um dado problema. Com essa visão, as energias devem ser direcionadas a arquitetura do *software* a ser produzida. Tal arquitetura possui dois focos gerais a serem explorados: (i) Divisão de tarefa por nó computacional (DPC); (ii) Divisão de tarefas em um nó computacional (DEC).

A DPC consiste em um processo de segmentação de áreas na imagem, correlacionadas ou não. Um fator a ser evidenciado é a correlação entre dois pontos (ou conjunto de pontos) de uma imagem. Se a operação de segmentação correlacionada for feita de forma coerente e o processo a ser aplicado depende desse tipo de relação, então a quantidade de transmissão de dados entre nós tende a cair. Tal processo pode ser feito de forma a dividir a imagem em grandes áreas de mesmo tamanho como descrito em Powell et al. (2010). A imagem da Figura 21 ilustra o conceito.

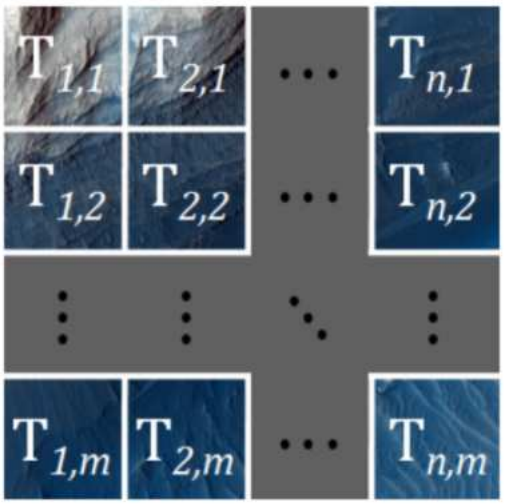

Figura 21: Divisão da imagem em pequenas *tiles*. Imagem adaptada de (POWELL et al., 2010).

A divisão que é atribuída à imagem consegue satisfazer as necessidades (DEC e DPC) para que os dados sejam distribuídos em diferentes máquinas. A variação do tamanho de cada parte pode ser dimensionada de acordo a disponibilidade de recursos de cada nó.

Outra forma consiste na divisão das imagens em porções latitudinais, apenas tendo foco na DPC (ANDRIJAUSKAS; GRADVOHL, 2012). A Figura 22 mostra um exemplo. As faixas amarelas são sobreposições relacionadas aos segmentos. Tais regiões são necessárias para satisfazer as necessidades de informação quando se utiliza processos que interagem com *pixels* vizinhos.

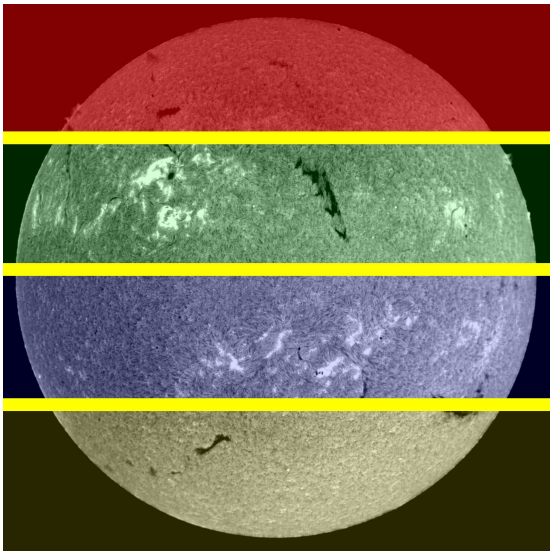

Figura 22: Áreas atribuídas aos diferentes processos (4 processos). Cada processo pode estar em um nó diferente dos demais.

Para contemplar a decomposição de domínio por e entre nós, é necessário dois tipos de modelos de programação: (i) Programação por memória compartilhada; e (ii) Programação por troca de mensagens.

4.2 Programação utilizando memória compartilhada

A programação por memória compartilhada consiste em utilizar técnicas em que diversos núcleos ou processadores compartilham a mesma memória (CHAPMAN et al., 2007). Dessa forma a comunicação é feita através da memória do nó. O diagrama da Figura 23 ilustra essa técnica.

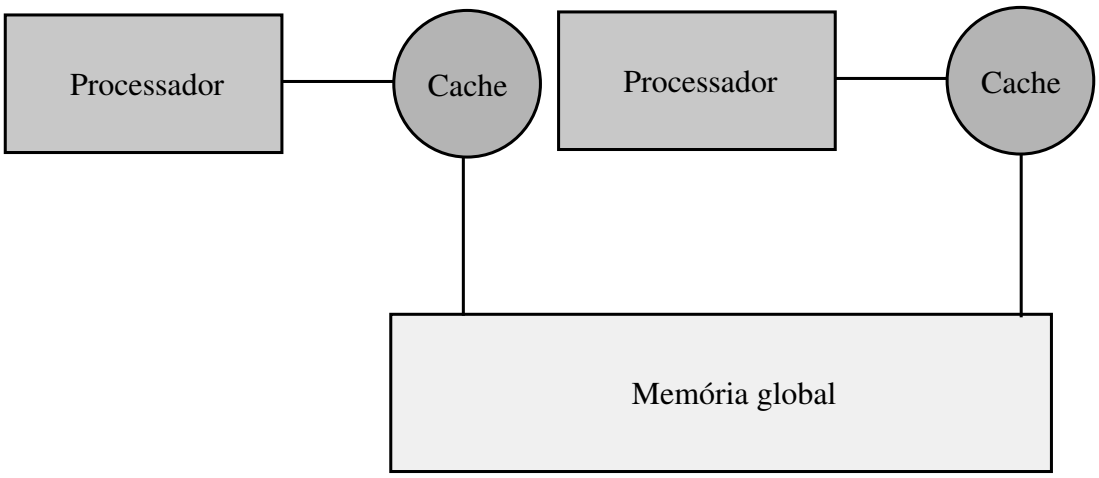

Figura 23: Diagrama que representa o processamento distribuído por memória compartilhada.

Quando um dos processadores precisa de algum dado que está na memória, pode acessar de forma direta. Essa técnica pode ser utilizada em conjunto com diversos tipos de biblioteca e *frameworks*. Dois deles são as *POSIX Threads* e o *Open Multi-Processing* (OpenMP) (GEBALI, 2011). As *POSIX Threads* consistem em uma *Application Programming Interface* – Interface para Programação de Aplicações (API) para processos leves (*threads*) que seguem o padrão *Portable Operating System Interface* – Interface portátil entre sistemas operacionais (POSIX) (MITCHELL et al., 2001). O processo para utilizar as *POSIX Threads* consiste na declaração de uma variável que irá representá-la, construir uma função que será executada por essa *thread* e por fim, utilizar a função `pthread_create` para invocar seu início (MATTHEW; STONES, 2011).

A criação de uma ou mais *threads* não é um processo complexo. Porém, os cenários onde é necessário esse paralelismo, em sua grande maioria, não consistem em problemas simples. Para que a utilização das *POSIX Threads* seja factível são necessários diversos outros recursos como *Semáforos*, *Mutexes* e afins. Toda essa complexidade tem de ser gerenciada pelo programador. Para que tal tarefa seja menos árdua e para que a produtividade aumente, existem bibliotecas que auxiliam no processo de paralelismo de um código (CHAPMAN et al., 2007).

Uma das bibliotecas a facilitar o desenvolvimento de *software* paralelo é o OpenMP. Ela consiste em uma *API* que tem por objetivo facilitar a programação que utiliza memória compartilhada (CHAPMAN et al., 2007). Diversos compiladores possuem suporte às marcações que o OpenMP provê, tais como: *GNU Compiler Collection* – Coleção de Compiladores GNU (GCC), *XL* da *IBM* entre outros. A criação de códigos paralelos com OpenMP consiste basicamente na marcação de áreas que devem ser paralelizadas. Um exemplo está na Figura 24.

```
1  /* Valor do iterador i, limiar l e n threads */
2  int i, l = 127, n = 10;
3
4  /* define n threads para o OpenMP */
5  omp_set_num_threads(n);
6
7  /* O for será paralelizado, percorre todos os pontos da imagem */
8  #pragma omp parallel for
9  for(i = 0; i < totalPontos; i++)
10 {
11   /* guarda o pixel na variável d */
12   int d = img->pixels[i];
13
14   /* se D maior ou igual a l, define o pixel com 255, senão 0 */
15   if( d >= l)
16     img->pixels[i] = 255;
17   else
18     img->pixels[i] = 0;
19 }
```

Figura 24: Trecho de código que demonstra o paralelismo com *OpenMP*.

A marcação *#pragma omp parallel for* no Código fonte da Figura 24 já provê o paralelismo dessa limiarização. A quantidade de *Threads* que será utilizada é definida pela função *omp_set_num_threads(n);* onde *n* é o número de *threads*.

Existem diversas outras marcações e tipos de funções que o *OpenMP* fornece (CHANDRA, 2001), por exemplo: Proteção e compartilhamento de variáveis dentro de contexto paralelo; sincronização e semáforos; entre outras funções.

Com todas essas funcionalidades, o OpenMP aparenta ser uma solução ótima para todos os casos em que é necessário incrementar o uso computacional, com o objetivo de diminuir o tempo de processamento. Entretanto, é necessário um planejamento e análise dos algoritmos a serem implementados. A primeira vista o OpenMP resolve qualquer problema de paralelismo com marcações simples.

Em aplicações bem desenhadas, a inserção do OpenMP tem por característica ser bem sucedida e ainda ser simples (CHAPMAN et al., 2007). Porém, no momento em que é entregue a complexidade do paralelismo à biblioteca é de se esperar a utilização de alguns ciclos para resolver o que deve ser processado e onde. Dado esse fato, é de se esperar uma pequena variação no desempenho, dependendo do tipo de problema e do número de *threads* para solucioná-lo.

De qualquer forma, todas essas funcionalidades do OpenMP têm grande valia em um contexto fechado de um computador (com a possibilidade de múltiplos núcleos). Entretanto, em cenários onde não é possível adicionar mais núcleos de processamento em uma máquina, a solução *OpenMP* fica limitada. Para resolver o dilema, existem outros modelos de programação para distribuir a carga. Porém, essa carga será distribuída entre diversos computadores.

4.3 Programação por troca de mensagens

A limitação física, técnica, financeira, entre outras impedem que as informações sejam processadas em apenas um computador. Para suplantar tais limitações, é possível interconectar os computadores para que a carga seja distribuída. A Figura 25 ilustra o conceito do processamento distribuído.

Figura 25: Diagrama que representa o processamento por memória distribuída.

No modelo distribuído, a Figura 25 mostra que os computadores não trocam dados pelo sistema de memória e sim por um barramento - rede - específico de comunicação. Para que seja possível a comunicação, é necessário um protocolo que defina como serão as mensagens.

Além do protocolo, é necessário que seja utilizado uma tecnologia para a comunicação. Com isso, o cenário se repete como no caso do *POSIX Thread versus* OpenMP. Porém, nesse caso a competição é entre *Sockets* e o *Message Passing Interface* – Interface de Troca de Mensagem (MPI).

O MPI consiste em um protocolo de comunicação padronizado específico para processamento distribuído e que possui diversas implementações (HAGER; WELLEIN, 2010). Utilizando o MPI é possível que diversas máquinas estabeleçam uma troca de mensagens. Entre as diversas implementações estão *OpenMPI*, *MPI-1 over OpenFabrics/Gen2, OprnFabrics/Gen2-UD, uDAPL, InfiniPath, VAPI and TCP/IP* (MVAPICH). Todas as implementações seguem um protocolo que estabelece que as mensagens sejam passíveis de interpretação. A troca de mensagens informa diversos itens relevantes à tarefa, tais como (HAGER; WELLEIN, 2010):

- qual processo está enviando a mensagem;
- onde estão os dados a serem processados;
- qual é o tipo de dado a ser processado;
- qual é a quantidade de dados;
- qual processo receberá a mensagem;
- onde as informações devem ser armazenadas durante o processo de recepção.

Todas essas informações estabelecem que cada nó, no momento em que receber as mensagens, saiba como proceder. Porém, as máquinas que são interligadas ainda podem conter diversos processadores (ou diversos núcleos). Nesse novo cenário, é necessária uma abordagem diferente para que a capacidade computacional seja utilizada em sua plenitude.

4.4 Sistemas de processamento híbridos

Em um cenário onde existam diversas máquinas e onde cada máquina possui diversos processadores ou núcleos é necessário um cuidado especial de como distribuir as tarefas para cada nó. A Figura 26 ilustra esse cenário.

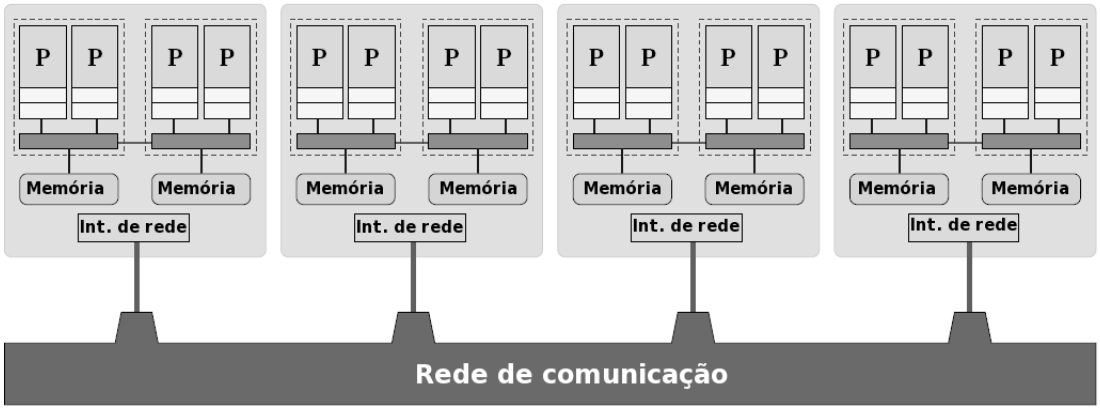

Figura 26: Modelo composto por processamento distribuído e por memória compartilhada. Adaptado de (HAGER; WELLEIN, 2010).

O cenário constituído pela Figura 26 apresenta um conjunto de computadores (nós), em que cada nó possui diversos processadores e é conectado por uma rede. Dessa forma cada nó pode receber tarefas e segmentá-las para que cada processador receba uma parte desse trabalho (HAGER; WELLEIN, 2010).

Nesse modelo híbrido é necessário explorar o máximo de cada nó e ainda dividir a tarefa entre diferentes máquinas. O MPI fornece a forma de como ocorre comunicação entre processos (que estarão em máquinas diferentes) e o OpenMP provê as características para que diversos núcleos de processamento recebam partes dessa tarefa. Essa abordagem é pertinente, pois o número de núcleos por máquina aumenta cada vez mais e ainda existem diversas limitações para se colocar uma grande quantidade de núcleos em um computador.

A taxonomia de sistemas paralelos em sistemas híbridos descrita por Rabenseifner et al. (2009) . Os sistemas MPI puros utilizam um processo MPI por núcleo. Dessa forma, a tarefa está na memória. Porém, são necessários arranjos para compartilhar. Em processos *OpenMP* puros são necessários sistemas externos para o compartilhamento da memória com os *Cluster OpenMP for Intel Compilers* (INTEL, 2010). Quando utilizados sistemas híbridos, a divisão de

dados entre nós é feita pelo MPI e a divisão de processamento é feita pelo *OpenMP*. Com base no modelo híbrido, o modelo *MasterOnly* define que a troca de informação feita pelo MPI não é executada pela *threads*, na sobreposição comunicação e computação as chamadas MPI são feitas na região paralela.

Para processar imagens astronômicas, o modelo *MasterOnly* é mais indicado. Rabenseifner (2003) descreve as vantagens dos sistemas híbridos utilizando a abordagem *MasterOnly*.

4.5 Considerações

A forma de se utilizar os recursos computacionais em sua plenitude varia de acordo com os tipos de conjunto de computadores a ser utilizado. Os modelos baseados em memória compartilhada recorrem a formas de comunicação mais rápida. Porém, o limite de recursos de apenas um nó pode ser um limitante. Os modelos de memória distribuída garantem que o limite global seja o do conjunto de máquinas e não de cada nó. Entretanto, requer mais atenção para o fluxo de dados que deve ser enviado para cada máquina.

Para os modelos de memória compartilhada e memória distribuída, podem ser aplicadas bibliotecas ou *frameworks* para lidar com essa complexidade. No caso de memória compartilhada recomenda-se o uso do OpenMP. No modelo de memória distribuída é possível preferível o MPI. Em ambos os casos é possível utilizar bibliotecas mais rústicas que garantam a comunicação ou distribuição da carga. Porém, o uso de bibliotecas de mais alto nível como OpenMP e MPI reduzem uma parte da complexidade do desenvolvimento do programador com um custo pequeno.

5 *Gaspra - Sistema de processamento de imagens astronômicas*

O *SOuthern Astrophysical Research Telescope* (SOAR) sobre responsabilidade do Laboratório Nacional de Astrofísica (LNA) e com parceria do INPE propôs o Desafio Bravo. Tal desafio consistia em desenvolver um sistema para catalogar e visualizar imagens geradas pelo SOAR. Para participar, sistema *Gaspra* foi desenvolvido. O *Gaspra* possui 6 componentes que têm por objetivo recuperar, tratar e visualizar a informação da forma mais concisa, rápida e relevante. A etapa de recuperação consiste nos processos de acessar a informação de forma padronizada e segura. Os processos de tratamento de imagens podem simplesmente melhorar a qualidade de exibição da imagem ou extrair informações relevantes. A etapa de visualização, por sua vez, consiste em propiciar para o usuário maneiras de observar a informação com artefatos propícios a exibição de dados dessa natureza. A Figura 27 mostra um diagrama geral do sistema.

O sistema de arquivos (item A do Diagrama da Figura 27) consiste em um dispositivo de armazenamento de alta capacidade, baixa latência e alta taxa de transmissão. Toda as imagens serão armazenadas neste dispositivo no formato FITS. A padronização e organização do acesso resolvem problemas de concorrência e maximizam a velocidade de acesso das imagens. Essas imagens serão catalogadas pelo Indexador *Multithread* (item B do Diagrama da Figura 27). Após a indexação, o acesso de forma padronizada e organizado pode ser feito pelo Servidor Web (item D do Diagrama da Figura 27).

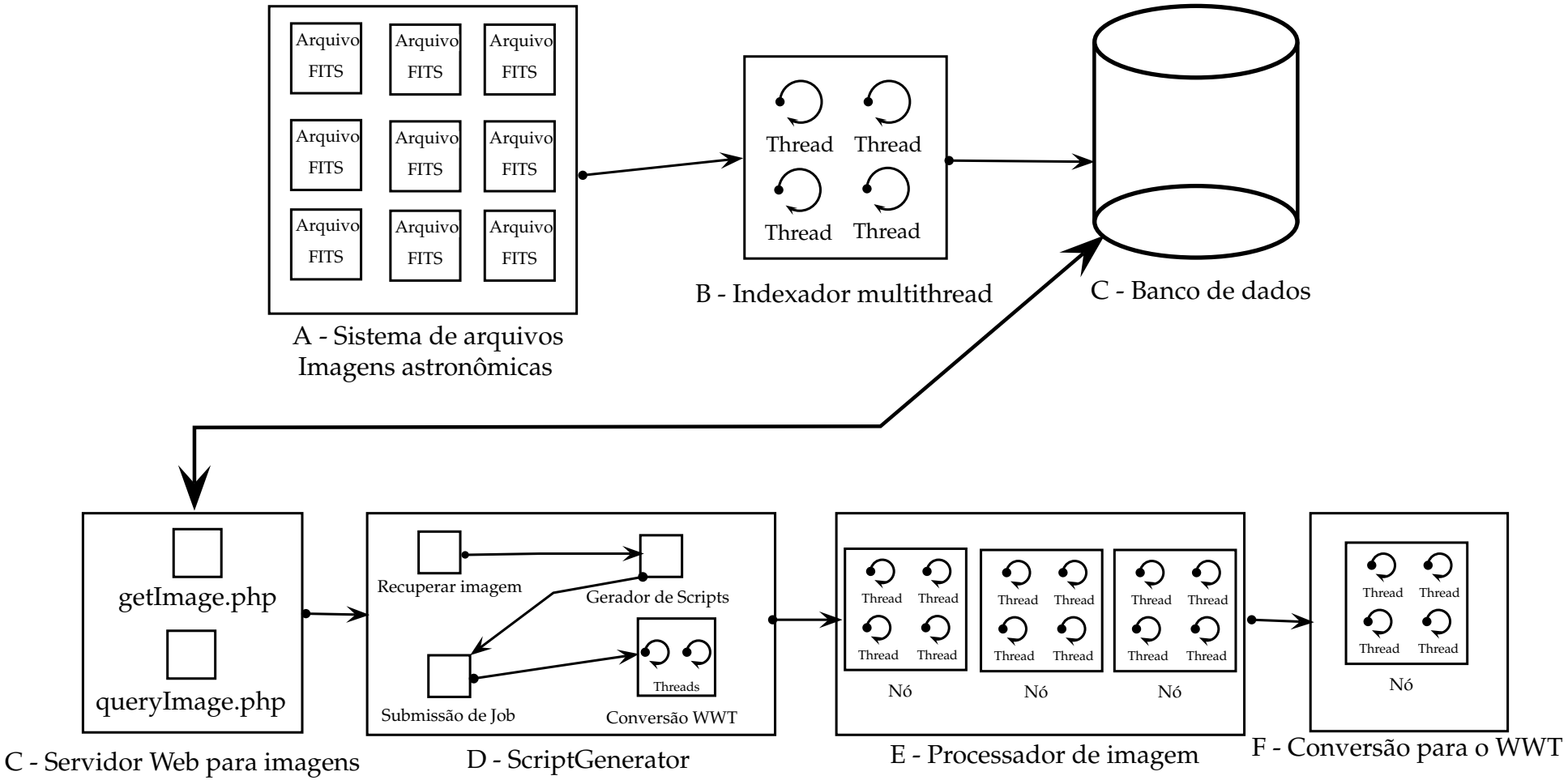

Figura 27: Diagrama da aplicação.

Pode-se fazer acesso às imagens manualmente ou por um *software*. Um exemplo de *software* é o *ScriptGenerator* - item E do Diagrama da Figura 27. O *ScriptGenerator* recupera as imagens através do Servidor *Web* e gera um *Job* a ser executado em um *Cluster IBM* com o *LoadLeveler* ou um *Cluster* simples para filtrar, segmentar ou efetuar outras operações nos arquivos *FITS*, além de acionar o módulo de indexação ou gerar os arquivos para o *World Wide Telescope* (WWT). Após a submissão e o processamento no Item E da Figura 27, as imagens são enviadas para o Item F, que por sua vez, faz a conversão da imagem, propiciando a exibição das imagens no WWT.

5.1 Indexador *multithread*

O indexador *multithread* consiste em um sistema que utiliza diversas *threads* para extrair o cabeçalho dos arquivos FITS, analisá-los, fazer as conversões necessárias, inserir as informações no banco de dados e por fim copiar as imagens para o diretório final. O indexador utiliza os Itens A e B da Figura 27.

Este processo de catalogação das imagens consiste em acessar os arquivos FITS utilizando a biblioteca *CFITSIO*. Após o processo de abertura do arquivo é possível acessar seu cabeçalho. O cabeçalho possui diversas informações da imagem, como local de captura, data da captura, bits por *pixels*, etc (BERRY, 2005). Um dos dados mais importantes é a consulta da posição da imagem. Esses dados devem ser padronizados propiciando o acesso da informação pelo sistema de coordenadas. A padronização adotada consiste na conversão de todas a posições de *Right ascension* – Ascensão Reta (RA) em hora decimal e *Declination* – Declinação (DEC) em graus seguindo as conversões de (DUFFETT-SMITH, 1989). Tais conversões são utilizadas para garantir a compatibilidade com o protocolo *Simple Image Access Protocol* – Protocolo Simples de Acesso a Imagem (SIAP) (TODY et al., 2011).

Porém, como a quantidade de dados é muito grande, é necessário realizar o processo o mais rápido possível. O indexador utiliza um conjunto de *threads* e um conjunto de conexões para utilizar a capacidade computacional de forma plena com o objetivo de minimizar o tempo de indexação. Esse subsistema utiliza o OpenMP para produzir as múltiplas execuções da indexação. Cada *thread* possui uma conexão com o banco de dados. Dessa forma, existe uma relação um-para-um entre *threads* e conexões. Essa relação aumenta a demanda do sistema de gerenciamento do banco de dados (SGBD) e as conexões de rede.

No diagrama da Figura 28, o passo "Carregar lista de arquivos" em cinza é o único que é feito sem paralelismo. O passo "Descompactar arquivo *FITS*" consiste em verificar e descompactar se a extensão do arquivo informa a compactação. O próximo passo "Acessar cabeçalho com dados" consiste em analisar a quantidade e quais HDU possuem dados: apenas esses serão utilizados na conversão. A etapa "Analisar e segmentar cabeçalho" lê todas as variáveis e aloca em estruturas para acomodar cada item. A fase "Converter RA e DEC para hora decimal" consiste em verificar se os dados de RA e DEC estão ou não em hora decimal e se necessário convertê-los. A etapa "Inserir banco de dados" consiste em utilizar os dados segmentados da imagem e colocá-los no banco e por fim, na etapa "Copiar imagem" a imagem é copiada para que o Servidor *web* possa enviá-las quando requisitado.

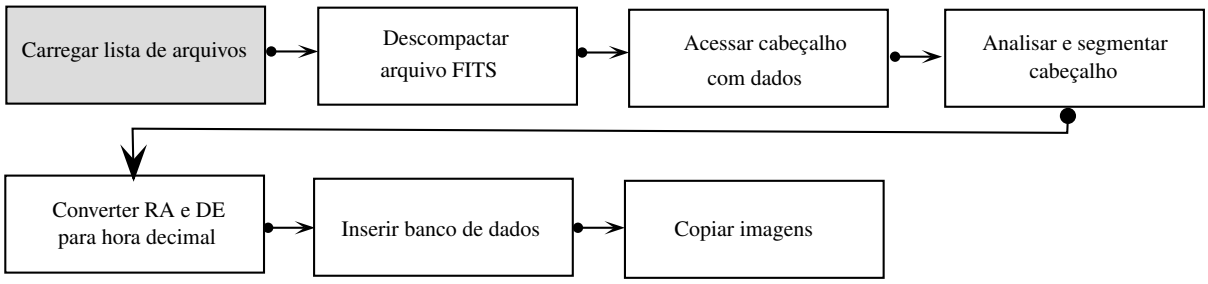

Figura 28: Processo para indexação de imagens.

A Figura 29 mostra a função que faz a segmentação das variáveis do cabeçalho do FITS. Esta função é chamada para cada linha do cabeçalho passando a variável que se quer encontrar, por exemplo `extrairDados(dadosFits[z],bitpix,"BITPIX");`. Neste caso, quer se encontrar a entrada `"BITPIX"`, dentro da matriz de *string* `dadosFits[z]` e guardar o valor na variável `bitpix`.

```
1  /* Recebe os dados  - PdadosFITS, a variável de destino - Pdestino
2  e o que sera buscado - PvariavelBuscada */
3  void extrairDados(char* PdadosFITS,
4                                  char* Pdestino,char* PvariavelBuscada ) {
5
6    /* Se a variável ja esta preenchida não fazer a busca */
7    if(Pdestino[0] != '0')
8      return;
9
10   /* Verifica se variável e a procurada */
11   int a = 0;
12   for (a = 0; a < strlen(PvariavelBuscada); a++) {
13     if (PvariavelBuscada[a] != PdadosFITS[a]) {
14       break;
15     }
16   }
17
18   /* Verifica se a variável possui conteúdo */
19   if (a == strlen(PvariavelBuscada) &&
20                          (PdadosFITS[a+1] == ' '|| PdadosFITS[a+1] == '=')
                               ) {
21     int c = 0;
22     /* copia a variável */
23     for (a = strlen(PvariavelBuscada) + 1; a < strlen(PdadosFITS); a++) {
24       if (PdadosFITS[a] != '/') {
25           /* Se a variável for a RA ou DEC, copia apenas os dados
                relativos as posições*/
26         if(strcmp(PvariavelBuscada,"RA") == 0 || strcmp(PvariavelBuscada,
              "DEC") == 0)
27         {
28           if(isdigit(PdadosFITS[a]) != 0
29                      || PdadosFITS[a] == ':'
30                      ||  PdadosFITS[a] == '-'
31                      ||  PdadosFITS[a] == '.')
32           {
33             Pdestino[c] = PdadosFITS[a];
34             c++;
35           }
36         }else{
37           /* Se for outros dados, apenas copia */
38           Pdestino[c] = PdadosFITS[a];
39           c++;
40         }
41       } else
42         break;
43     }
44   }
45 }
```

Figura 29: Trecho de código que extrai e segmenta o cabeçalho da imagem.

5.2 Servidor Web

O Servidor Web serve como porta de entrada para as requisições de recuperação de imagem. As requisições de *getImage* são tratadas por um servidor *Web Apache* utilizando *Hypertext Preprocessor* (PHP) e a base de dados o *Mysql*. O serviço foi baseado nos padrões do *International Virtual Observatory Alliance*, especificamente no SIAP (TODY et al., 2011).

Esse servidor recebe as requisições *Hypertext Transfer Protocol* – Protocolo de Transferência Hypertext (HTTP) do tipo *GET*. As variáveis passadas para o serviço são definidas de forma dinâmica. Quando passada uma variável, faz-se a busca com base nos nomes e valores transmitidos. Desta forma, se o serviço de indexação das imagens adicionar mais informações não será necessário alterar o método *getImage*. Como exemplo, a requisição pode ser `http://127.0.0.1/SIAP/getImage.php?naxis1=568`. O sistema irá buscar no banco a imagem que tenha o eixo 1 com 568 pontos. As variáveis que podem ser passadas para acesso da informação estão na Tabela 3, as descrições das variáveis são provenientes do *National Optical Astronomy Observatory* e pelo (WELLS; HARTEN, 1981).

Tabela 3: Variáveis disponíveis para consulta.

Variável	Descrição
date	Data de criação do arquivo
crval1 e *crval2*	Coordenadas do centro da imagem
bitpix	bits por pixels
naxis	número de eixos na imagem
naxis1 e *naxis2*	número de pixels no eixo 1 e 2
origin	responsável pela geração do arquivo
exptime	tempo de exposição da imagem em segundos
epoch	equinócio de eixo óptico
ra	ascensão direita
decp	declinação
date-obs	data de observação da imagem
time-obs	hora da observação
observat	observatório responsável pela imagem
airmass	Massa de ar durante a captação da imagem
ctype1 e *ctype2*	tipo de coordenada
cd1_1, *cd2_1*, *cd1_2* e *cd2_2*	utilizados para conversão de pontos para WCS

5.3 ScriptGenerator

O *ScriptGenerator* possui diversos papéis que têm como ideia central fornecer uma forma de automatizar o processamento da informação. São elas:

1. recuperar as imagens através do Servidor *Web*;
2. gerar *script* configurando quantas *threads* e processos devem ser utilizados no *cluster* para processar as imagens;
3. submeter o *Job* para o *cluster*;
4. preparar as imagens para serem exibidas no WWT;
5. acionar o módulo de indexação de imagem;
6. colher a informação e gerar gráficos com a ferramenta *gnuplot*;
7. recuperar imagens solares do projeto *Global Oscillation Network Group* (GONG);
8. enviar os resultados para um servidor remoto e avisar sobre o término do processamento.

Essas oito tarefas são retratadas no diagrama da Figura 27 no Item D. A etapa de recuperação de imagem é feita através de chamadas do *wget*. Quando um conjunto de arquivos pré-definidos ou quando o conjunto de imagem que o usuário determinou são recuperados, o sistema gera um *script* para cada imagem a ser processada. Após a criação, cada *script* é enviado ao *cluster* e as imagens são processadas. Neste momento, é possível escolher qual será o algoritmo utilizado para processar. Com a imagem tratada, é possível já visualizar o arquivo em *png* ou enviá-lo para conversão e posterior visualização no WWT. Esse módulo é escrito em *Python*. Desta forma, o pesquisador pode alterar ou criar *scripts* da forma que desejar de maneira simples.

5.4 Processador de imagens

O módulo que faz o processamento das imagens consiste em um sistema de memória compartilhada e memória distribuída. Neste caso, a junção desses dois itens são denominados sistema híbridos (RABENSEIFNER et al., 2009). A Figura 30 mostra o diagrama desse módulo.

Dadas as dimensões da imagem e os respectivos volumes de dados, é importante utilizar métodos de computação de alto desempenho para reduzir o tempo de processamento total. Portanto, utiliza-se técnicas paralelas de programação para sistemas híbridos. Utilizou-se OpenMP para a programação paralela em memória compartilhada e a biblioteca MPI para programação em memória distribuída (RABENSEIFNER, 2003).

Em um cenário de memória compartilhada, a imagem deve ser dividida em segmentos, a fim de aplicar os filtros e outros processos. Esta etapa consiste na decomposição de domínio. Portanto, a imagem pode ser dividida em segmentos latitudinal de igual comprimento, uma para cada processo MPI. Eventualmente, uma pequena área de intersecção é enviada para os processos com o objetivo de fornecer informação para algoritmos que utilizam *pixels* vizinhos. A Figura 30 mostra um diagrama que descreve a aplicação paralela. No primeiro passo, a aplicação lê a imagem e alguns parâmetros para a execução. Em seguida, analisa-se o cabeçalho de imagem para definir a estratégia a ser empregada na decomposição de domínio, com a área de interseção de cada processo MPI, se for necessário.

Depois, o sistema divide a imagem e envia cada segmento utilizando funções não bloqueantes do MPI. Cada segmento é processado utilizando OpenMP. A Figura 31 mostra o processo de envio. Finalmente, reconstrói-se a imagem final e a armazena em disco.

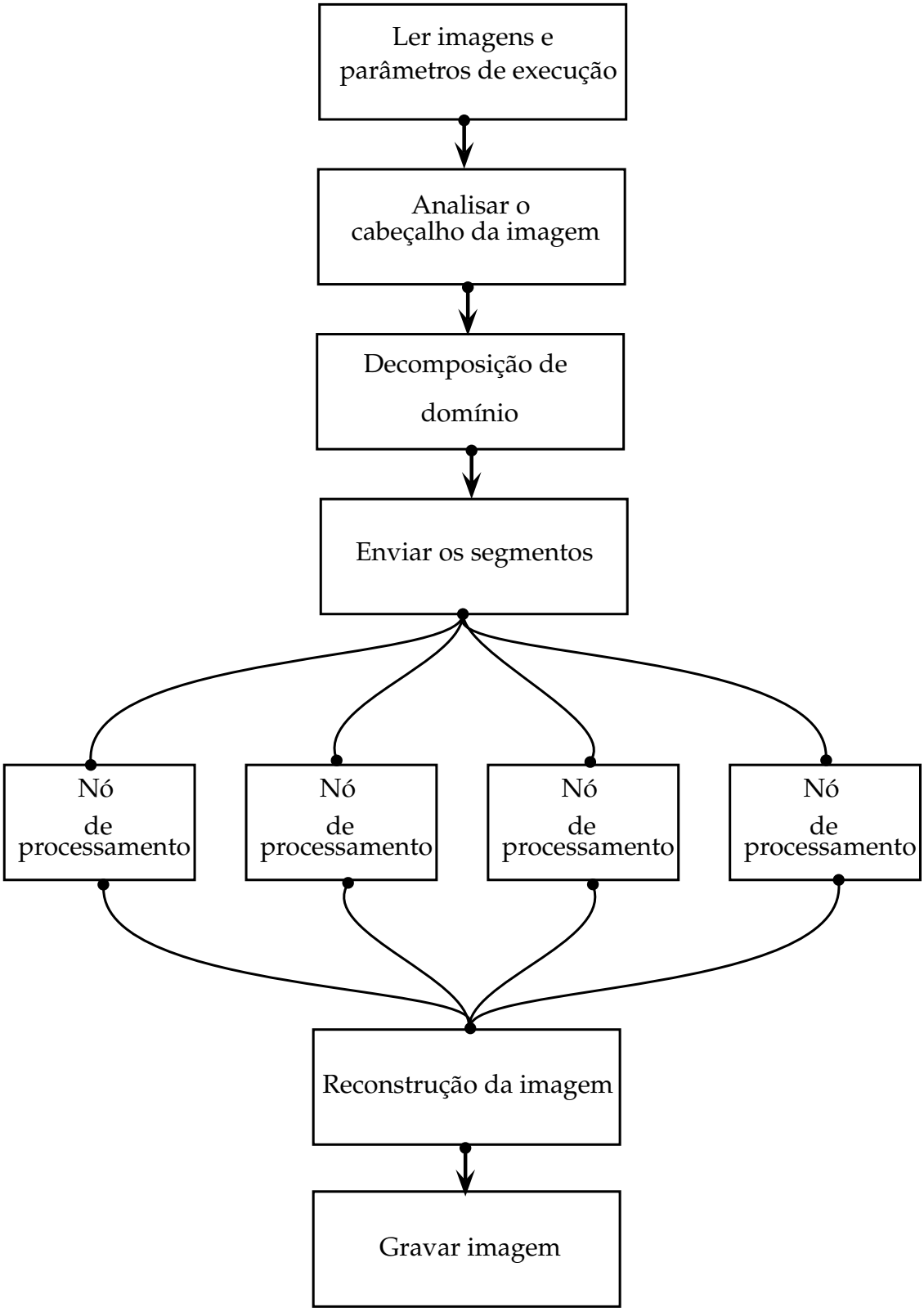

Figura 30: Diagrama do processamento das imagens.

```
1 /* Para cada processo MPI */
2 for(ti = 0; ti < qtdThreadMPI; ti++,qtdRequest+=2)
3 {
4     /* Recupera o tamanho do segmento */
5   int tam = args[ti].total;
6
7     /* Um vetor com os dados do segmento, com o início
8     e fim dos dados e afins */
9   dados[0] = args[ti].posInic;     /* posição inicial dos dados */
10  dados[1] = args[ti].posFinal;    /* posição final dos dados */
11  dados[2] = tam;                  /* total dados */
12  dados[3] = args[ti].posInicGap;  /* posição inicial Gap */
13  dados[4] = args[ti].posFinalGap; /* posição final Gap */
14  dados[5] = sizeD[0];             /* nx */
15  dados[6] = sizeD[1];             /* ny */
16
17  /* Enviar o total de pontos e o tamanho das imagem */
18  status = MPI_Isend(&dados,7,MPI_INT,ti+1,1,MPI_COMM_WORLD,
19                                &request[qtdRequest]);
20
21  /* Envia o segmento para o processo MPI com
22  função não bloqueantes */
23  status = MPI_Isend(&pixels1[args[ti].posInicGap],args[ti].total,
        MPI_DOUBLE,ti+1,2,MPI_COMM_WORLD,&request[qtdRequest+1]);
24
25  /* guarda em um vetor de quais requisições foram feitas */
26  requestIF[qtdRequest+1] =1;
27 }
```

Figura 31: Trecho de código que envia segmento de imagens para outros processos MPI.

Foram implementados diversos algoritmos para a detecção de eventos e tratamento das imagens. Todos esses processos podem ser utilizados diretamente no *cluster* ou com o *scriptGenerator*. A Tabela 4 apresenta quais são os filtros e processos disponíveis.

Tabela 4: Filtros e processos disponíveis.

Filtro	Descrição	Utilização
Binarização	Aplica o processo de binarização compactando a escala para 8 bits	bXXXXXX - XXXXXX é um número inteiro que representa o valor do limiar
Transformação de escala de pixel	Compacta a escala aplicando uma transformação	lg: logarítmica, le: exponencial e ln: normalização
bArea	Detecção das áreas mais brilhantes da imagem	a
Inversão	Inversão dos pixels e compacta a escala para 8 bits	i
Filtro de média	Aplicando um filtro de média com janela variável	mXXX - XXX é um número que representa o lado da janela quadrada de *pixels*
Detecção de filamentos solares	Detecção de filamentos utilizando técnica baseada no trabalho Gao et al. (2002)	f
Detecção de bordas	Detector de bordas *Sobel*	s
Filtros morfológicos	Abertura e fechamento para imagens binárias e em cinza	m
Filtro de difusão	Filtro de difusão para atenuação de ruídos	d

5.5 Criação de arquivos para o WorldWide Telescope *(WWT)*

O sistema de conversão para o WWT baseado nas especificações fornecidas pela *Microsoft* (TELESCOPE, 2012) utiliza diversas *threads* para processar as imagens com o objetivo de prepará-las para o WWT. Uma *thread* busca as imagens em um diretório predefinido e repassa o trabalho para outra *thread* livre fazer a conversão. Esta conversão consiste na criação do arquivo *wtml* e na construção dos *tiles*.

A criação dos arquivos *wtml* implica na divisão da imagem em diversos segmentos de tamanhos pré-definidos (*tiles*) em diferentes níveis. No nível 0 a imagem deve ter 256×256 *pixels* e nenhuma divisão. O tamanho da imagem e suas divisões aumenta conforme o nível aumenta. No caso do nível 2, a imagem deve ter 1024×1024 *pixels* e ter 16 divisões. Este processo é necessário para a visualização da imagem em diversas posições de zoom. O diagrama da Figura 32 apresenta esses passos.

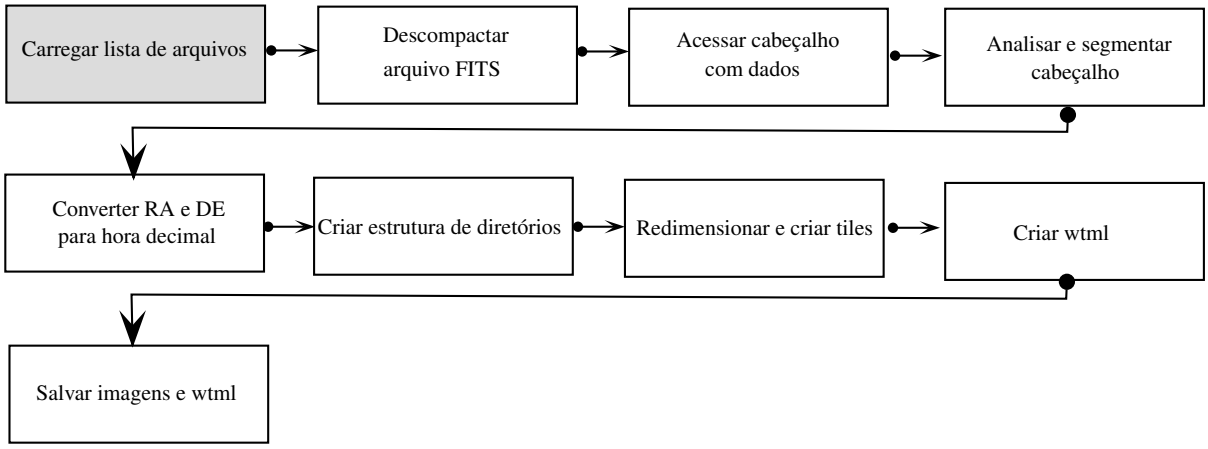

Figura 32: Processo de conversão para o *WWT*.

Os passos anteriores a "Criar estrutura de dados"são descritos na Seção 5.1. Esta estrutura é necessária, pois o WWT utiliza hierarquia específica para os *tiles*. O passo "Redimensionar e criar *tiles*"representa a interpolação e criação dos segmentos para exibição no WWT. O processo "Criar wtml"consiste em utilizar as informações das imagens para criar o *xml* para o WWT e por fim "Salvar imagens e wtml"consiste em salvar os dados no disco.

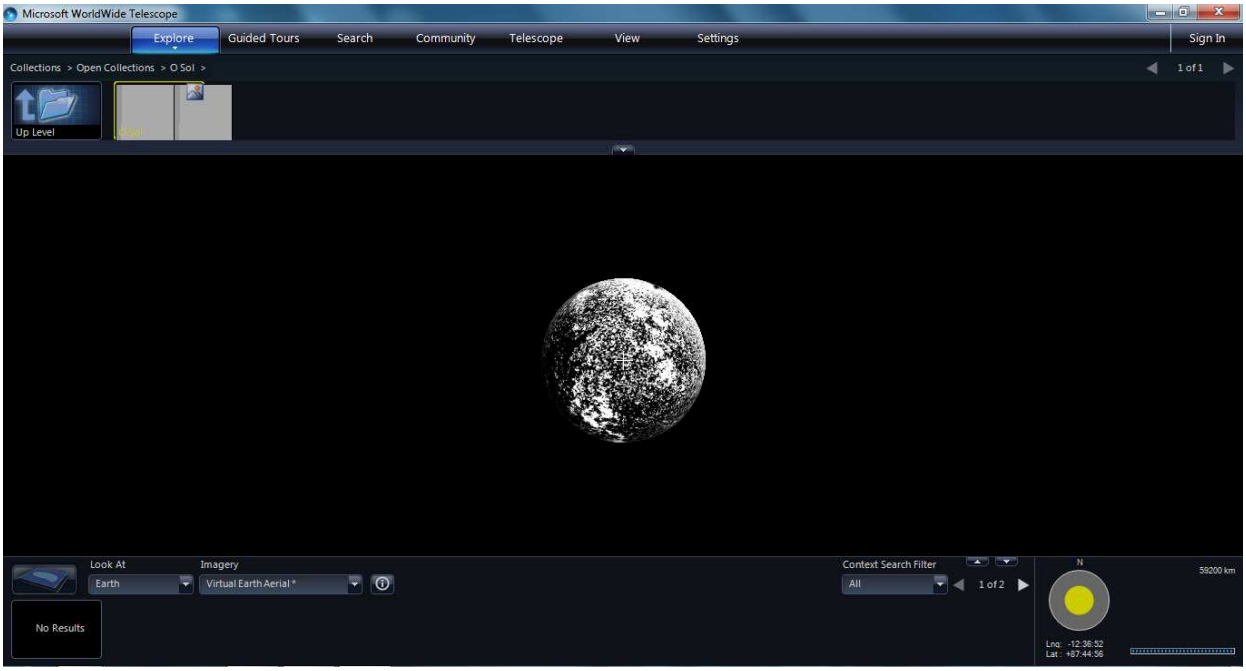

Figura 33: Imagem do Sol processada com filtro de binarização convertida para o *WWT*.

Todo o processo de criação dos *tiles* foi produzido a partir de funções básicas da *libpng* (apenas funções para ler os arquivos, definir *pixels* e salvar no disco). Desta forma a paralelização pode ter uma granularidade menor, possibilitando um uso maior da capacidade computacional. A Figura 34 mostra uma trecho de código responsável por dividir a imagem em *tiles* e criar o arquivo *whtml*. Utilizando apenas a biblioteca *libpng* e a *Mini-XML*. A Figura 33 mostra uma imagem do Sol que estava no formato FITS convertida para o WWT. Foi aplicado um filtro de binarização da imagem.

```
1  /* Redimensiona as imagens para 2 níveis de tiles */
2  double* t1 = resizeImageSetSize(pixels1,w,h,512,512);
3  double* t0 = resizeImageSetSize(pixels1,w,h,128,64);
4
5  /* Cria a miniatura  */
6  char n[] = "wwt/t%i/t.Png";
7  sprintf(buffer,n,z);
8  criaPNG(buffer,128,64,t0);
9
10 /* Cria o nível 0 item x0y0 */
11 t0 = resizeImageSetSize(pixels1,w,h,256,256);
12 char n1[] = "wwt/t%i/Pyramid/0/0/L0X0Y0.Png";
13 sprintf(buffer,n1,z);
14 criaPNG(buffer,256,256,t0);
15
16 /* Cria o nível 1 item x0y0 */
17 char n2[] = "wwt/t%i/Pyramid/1/0/L1X0Y0.Png";
18 sprintf(buffer,n2,z);
19 sliceImage(t1,512,512,0,0,256,256,buffer);
20
21 /* Cria o arquivo XML para o WWT */
22 mxml_node_t *xml;
23 mxml_node_t *folder;
24 mxml_node_t *place;
25
26 xml = mxmlNewXML("1.0");
27 xmlElementSetAttr(folder,"Name",dadosdb->name); /* Nome do objeto */
28 mxmlElementSetAttr(place,"RA",dadosdb->ra); /* Posição RA */
29 mxmlElementSetAttr(place,"DEC",dadosdb->dec); /* Posição DEC */
```

Figura 34: Trecho de código que cria o *tiles* e o arquivo *wtml*.

5.6 Utilização do sistema *Astronomus*

Existem três opções para executar o *ScriptGenerator* no sistema *Astronomus*. A primeira opção é executar o indexador, com o comando: `./sg -i diretório qtdThreads`. Onde a palavra `diretório` deve ser alterada para o diretório onde se encontram as imagens a serem indexadas e a palavra *qtdThreads* deve ser substituída pelo número de *threads* que irão executar esse processo.

Outra opção é executar a ação de transformar a imagem para o formato do WWT. O comando para isso é: `./sg -w diretório qtdThreads`, em que o diretório contém os arquivos FITS a serem convertidos e a palavra *qtdThreads* deve ser substituída pelo número de *threads* que irão executar esse processo. O resultado estará no diretório WWT.

Além de executar essas ações de forma independente, é possível montar um experimento que faça o processo automático. Os arquivos devem ficar na pasta: `src/experimentos`. Utiliza-se a reflexão do *Python* para carregar os experimentos. Um exemplo desta utilização está na Figura 35. Os comentários nos códigos da Figura 35 explicam a utilidade de cada linha. Para executar esse *script*, utilize o comando: `./sg -s E1`, onde o código E1 é referente ao experimento criado. Após a conversão para WWT o resultado é compactado (com o nome de wwt.zip) e disponibilizado no servidor web.

```
# bibliotecas utilizadas, podem ser tomadas como padrão
import BDClass
import GeraArquivoClass
from time import sleep
import os
import UtilClass
import ParLLClass
import ParAstroClass
class Experimento:
    # deve existir apenas um experimento com esse ID
  id = "E1"
  def roda(self):
    # classe útil, comandos para inciar o processamento das imagens
    util = UtilClass.Util
    # parâmetros relativos ao processamento da imagem
    astro = ParAstroClass.ParAstro()
    # parâmetros relativos da forma como a imagem sera processada
        ll = ParLLClass.ParLL()
    # responsável em gerar os arquivo que de execução do processamento
    geraArquivo = GeraArquivoClass.GeraArquivo()
    # recupera uma lista de imagens dado um valor
    dados = util.recuperaListaImagens(self,"naxis=2");
    for item in dados:
      print item
    # imagem em questão, pode ser os nome, a posição etc.
    imgB = "POS=26.67306,-8.433555&SIZE=00.1,00.1"
    util.recuperaImagem(self,imgB,"img/1.fits")
    # processar utilizando 5 thread openmp
    astro.qtdOpenMP = 5
    # saída do arquivo
    astro.outputFits = 'outputFits/output.fits'
    # filtro à ser utilizado
    astro.filtro = 'a'
    # a quantidade de tarefas por nó
    astro.tasks_per_node =  ll.tasks_per_node = 4
    # nome do script
    nome ="processar.cmd"
    # gera o arquivo para ser enviado
    geraArquivo.geraArquivoLL(astro,ll,nome,0)
    # converte para o WWT o resultado
    util.geraWWT(self,'outputFits/',1)
```

Figura 35: Recuperação, processamento e visualização.

5.7 Considerações

O sistema *Astronomus* compreende uma grande gama de aspectos necessários para o estudo de imagens astronômicas. Com a indexação, passando por etapas de detecção/processamento e por fim a visualização. Tal característica promove a utilização da informação de forma plena.

Os *ScriptGenerator* garante que um conjunto de tarefas possa ser feito de forma automática e que seja submetido a diferente conglomerados de computadores. Dessa forma, a detecção de eventos e até processos de *data mining* podem ser implantados. Por fim, implementou-se a visualização em um sistema que garante a colocação dos dados de forma correta.

6 Resultados

Os testes foram feitos inicialmente em um computador doméstico para a prova de conceito (ANDRIJAUSKAS; GRADVOHL, 2011). Experimentos inicias para demonstrar que um sistema híbrido tinha ganhos em relação a uma abordagem baseada em sistemas de memória distribuída foram executados no inicio do projeto. Os experimentos mais elaborados foram feitos utilizando a estrutura computacional do *Laboratório de Simulação e de Computação de Alto Desempenho* (LaSCADo) pertencente a Faculdade de Tecnologia (FT) da Universidade Estadual de Campinas (UNICAMP) descritos na Seção 6.1. Todos os testes de detecção de filamentos possuem duas etapas. A primeira etapa consiste em verificar quais são as taxas de detecção do eventos em questão. A segunda etapa consiste em mostrar quais foram os ganhos no tempo computacional (*speedup*) com diversas configurações.

6.1 Computador de alto desempenho

Um dos *clusters* disponíveis no LaSCADo utilizado nos testes é equipado com cinco nós, cada um deles com quatro processadores *Power 755* de 3.0 GHz, 12 gigabytes de RAM e rede *Infiniband* para interligar os nós. Conta com o sistema operacional *AIX* 6.1, *IBM XL C* versão 1.11, *MPI* versão 2.1, *OpenMP* versão 3.0 e *CFITSIO* 3.29 (PENCE, 1999). Cada nó nesse *cluster* possui quatro processadores, cada processador possuir oito núcleos e cada núcleo gerência diretamente quatro *threads*. A Equação 10 mostra quantos *threads* o sistema pode administrar nativamente.

$$\begin{aligned} 5\ \text{(nós)} \times 4\ \text{(processadores por nó)} \times 8\ \text{(núcleos por processadores)} \\ \times 4\ \text{(thread por núcleo)} = 640\ \text{threads} \end{aligned} \tag{10}$$

Em um sistema híbrido com 640 *threads* disponíveis, é necessário fazer o ajuste da quantidade de processos MPI e *threads OpenMP*. Rabenseifner et al. (2009) afirma que é necessário uma análise complexa para encontrar a melhor combinação entre *threads* e processos. Hager e Rabenseifner (2009) faz afirmação semelhante definindo que não é uma tarefa trivial encontrar esse modelo ótimo. Para encontrar a melhor relação *threads OpenMP* e processos *MPI* utilizou-se o *scriptGenerator*. Com ele foi possível fazer a automatização das submissões e geração de gráficos de desempenho do sistema. Foram formulados três testes que estão na Tabela 5.

Tabela 5: Configuração dos experimentos

ID do código	*MPI* por nó	Variação de *thread OpenMP*
1	1	2, 4, 8, 16, 32, 64, 128 e 256
2	4	2, 4, 8, 16, 32, 64, 128 e 256
3	8	2, 4, 8, 16, 32, 64, 128 e 256

O teste ID de código 1 da Tabela 5 envia um processo *MPI* para cada nó do cluster; o de código 2 envia quatro processos para nó; e o de código 3 envia oito processos para cada nó. Em todos os testes diversas quantidade de *threads OpenMP* são avaliadas.

Foram utilizados cinco imagens produzidas pela *Big Bear Observatory* para testes. Tais imagens do Sol foram obtidas em Janeiro de 2012. Para cada imagem, foi construído manualmente um *ground truth*, em que marcou-se a área de cada filamento com pontos brancos e tais imagens foram utilizadas como referências.

6.1.1 Detecção de filamentos

Para cada imagem escolhida foi produzido um *ground truth* com o objetivo de avaliar as taxas de detecção de filamentos. Os testes consistiram em verificar se os pontos detectados pelos métodos eram coincidentes com os do *ground truth*. Desta forma, conseguiu-se avaliar os falso positivo (FP) e falso negativo (FN). Além dessa abordagem, verificou-se que o método foi capaz de detectar a existência do filamento.

A Tabela 6 contém a taxa de acertos de cada imagem processada, bem como as taxas de falsos negativos (FN) e falsos positivos (FP), quando comparada com o *ground truth*. A taxa de 100% de acertos ocorre quando a imagem processada por qualquer método é igual ao *ground truth*.

Tabela 6: Resultados de detecções nos algoritmos de comparação de todos os pontos contra o *ground truth*.

	FDD			MD		
	Acertos	FN	FP	*Hits*	FN	FP
Imagem 1	78%	21%	1%	50%	50%	0%
Imagem 2	72%	24%	4%	57%	41%	2%
Imagem 3	83%	16%	1%	58%	42%	0%
Imagem 4	87%	10%	3%	56%	40%	4%
Imagem 5	80%	17%	3%	70%	29%	1%

A média de acertos do método FDD é 80% com um coeficiente de variação de 7%. A média de FN para o método FDD foi de 17%, enquanto a média de FP foi de 2%. Para o método MD, a média foi de 58% com um coeficiente de variação de 14%. O FN do MD foi de 40% e o FP de 1%. Outra análise dos dados, que avalia se o método consegue detectar a existência do filamento é apresentada na Tabela 7. Neste caso o método FDD teve uma taxa de 94% de acertos e o método MD de 66%.

Tabela 7: Resultados da detecção considerando se o método detecta a existência do filamento.

	Quantidade de filamentos	FDD	FDD%	MD	MD %
Imagem 1	7	6	85%	4	57%
Imagem 2	11	11	100%	6	54%
Imagem 3	8	7	87.5%	5	62%
Imagem 4	7	7	100%	6	86%
Imagem 5	8	8	100%	6	75%

Os testes de desempenho foram baseados na Tabela 5. O teste da Figura 36 mostra como foi o desempenho do método FDD utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com apenas um processo MPI por nó com um total de 5 nós. A linha com o $+$ que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com $\times$; utilizando 8 *threads* OpenMP está marcada com $\divideontimes$; a abordagem com 16 *threads* está marcada com $\square$; com 32 *threads* marcado com $\blacksquare$; 64 *threads* está marcado com $\bigcirc$; 128 *threads* esta destacada com $\bullet$; e 256 *threads* marcado com $\triangle$. A linha de 128 *threads* está destacada em vermelho por ter alcançado o menor tempo utilizando 9,58 segundos.

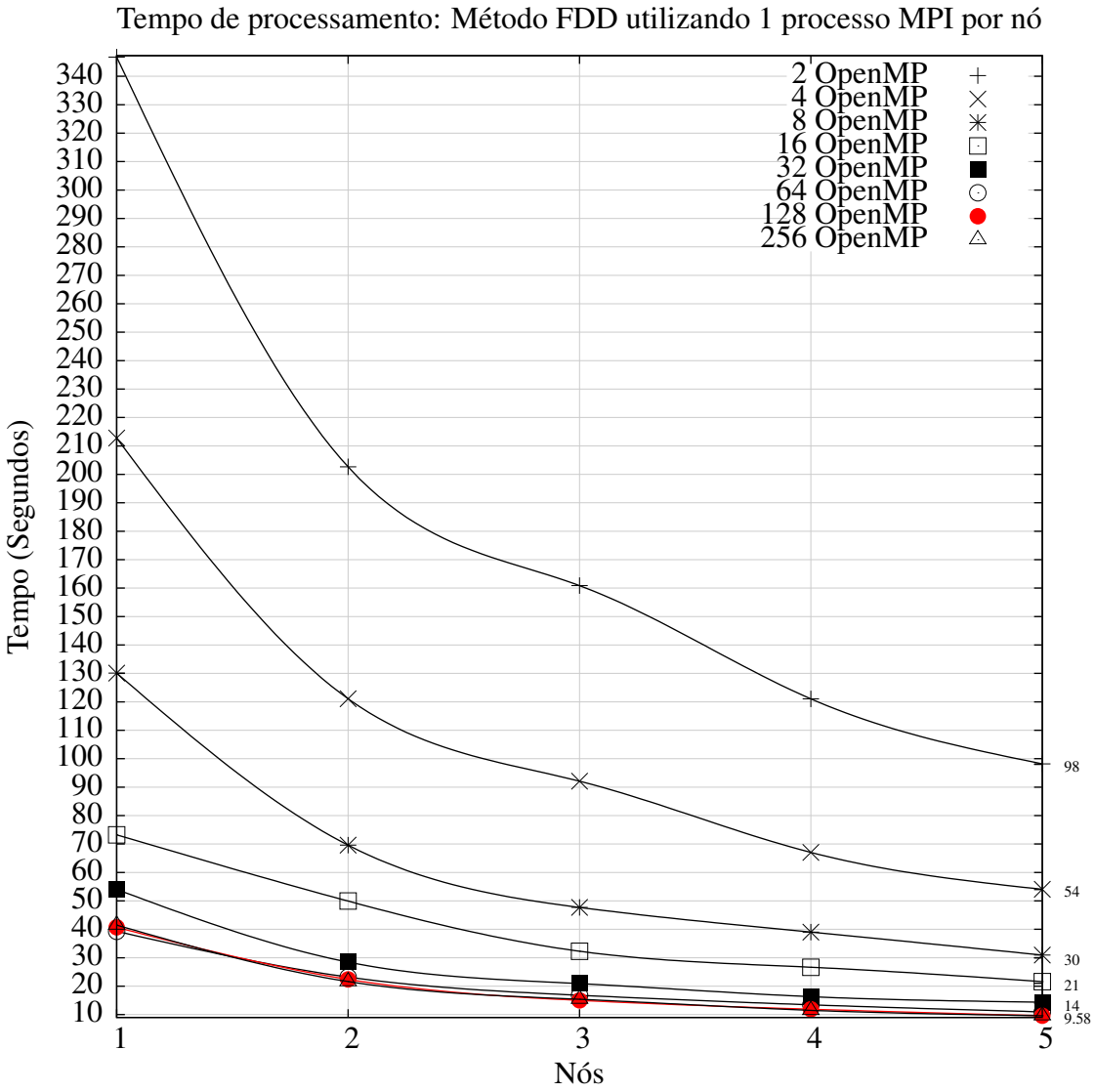

Figura 36: Tempo de processamento para o método *FDD* utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com um processo *MPI* por nó.

O teste da Figura 37 mostra como o foi o desempenho do método FDD utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 4 processos MPI por nó com um total de 5 nós. A linha com o $+$ que utiliza 2 *threads* OpenMP, com 4 *threads* OpenMP está marcada com $\times$, utilizando 8 *threads* OpenMP está marcada com $\ast$, a abordagem com 16 *threads* está marcada com $\square$, com 32 *threads* marcado com $\blacksquare$; 64 *threads* está marcado com $\bigcirc$; 128 *threads* esta destacada com $\bullet$ e 256 *threads* marcado com $\triangle$. A linha de 128 *threads* está destacada em vermelho por ter alcançado o menor tempo em 5,8 segundos.

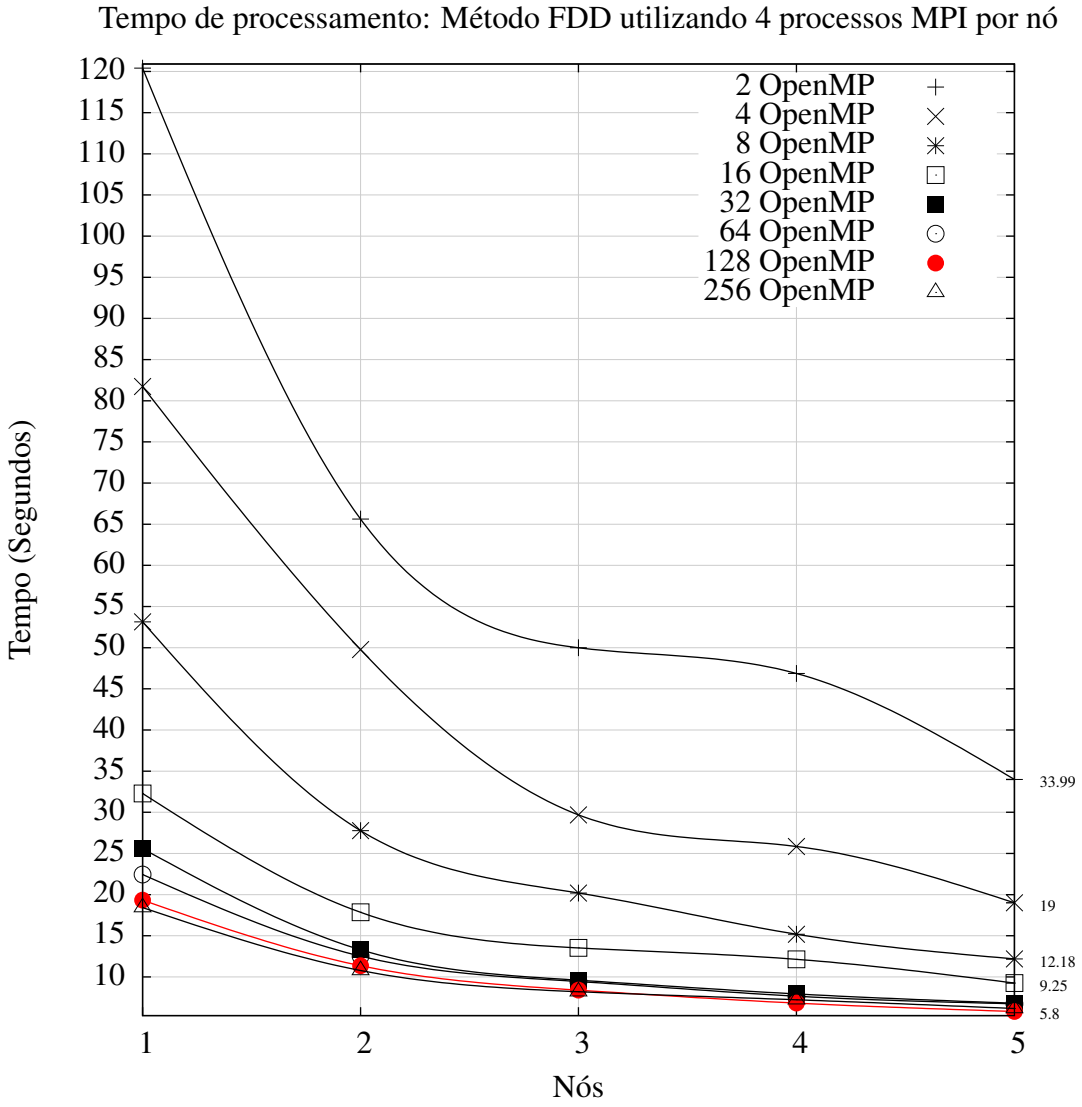

Figura 37: Tempo de processamento para o método *FDD* utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com quatro processos *MPI* por nó.

O teste da Figura 38 mostra como o foi o desempenho do método FDD utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 8 processos MPI por nó com um total de 5 nós. A linha com o $+$ que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com $\times$; utilizando 8 *threads* OpenMP está marcada com $\ast$; a abordagem com 16 *threads* está marcada com $\square$; com 32 *threads* marcado com $\blacksquare$; 64 *threads* está marcado com $\bigcirc$; 128 *threads* esta destacada com $\bullet$; e 256 *threads* marcado com $\triangle$. A linha de 32 *threads* está destacada em vermelho por ter alcançado o menor tempo em 6,0 segundos.

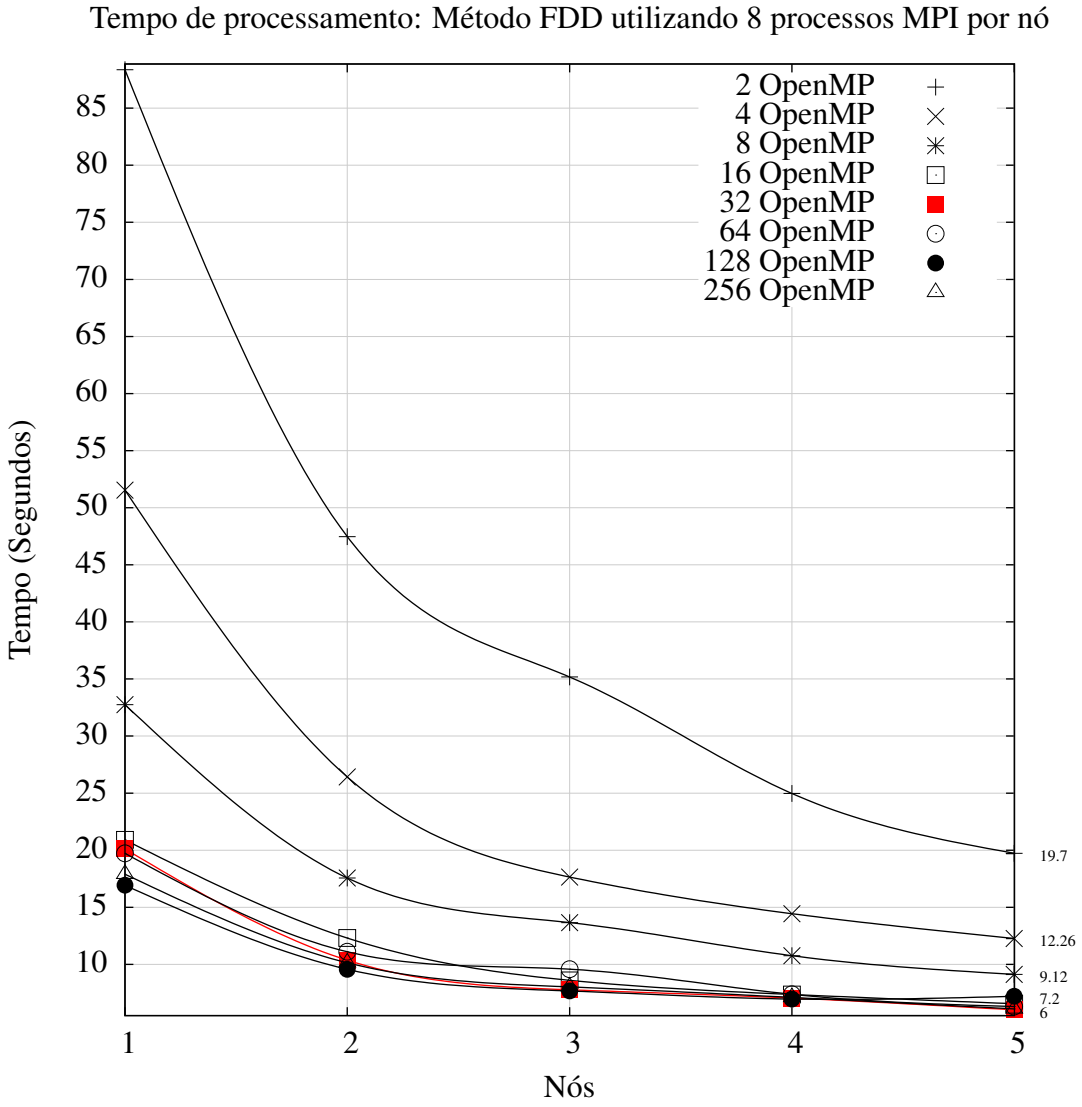

Figura 38: Tempo de processamento para o método *FDD* utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com oito processos *MPI* por nó.

Os testes com o MD utilizaram a mesma abordagem utilizada para o método FDD. O teste da Figura 39 mostra como o foi o desempenho do método FDD utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 1 processo MPI por nó com um total de 5 nós. A linha com o $+$ que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com $\times$; utilizando 8 *threads* OpenMP está marcada com $\ast$; a abordagem com 16 *threads* está marcada com $\square$; com 32 *threads* marcado com $\blacksquare$; 64 *threads* está marcado com $\bigcirc$; 128 *threads* esta destacada com $\bullet$; e 256 *threads* marcado com $\triangle$. A linha de 4 *threads* está destacada em vermelho por ter alcançado o menor tempo em 4,12 segundos.

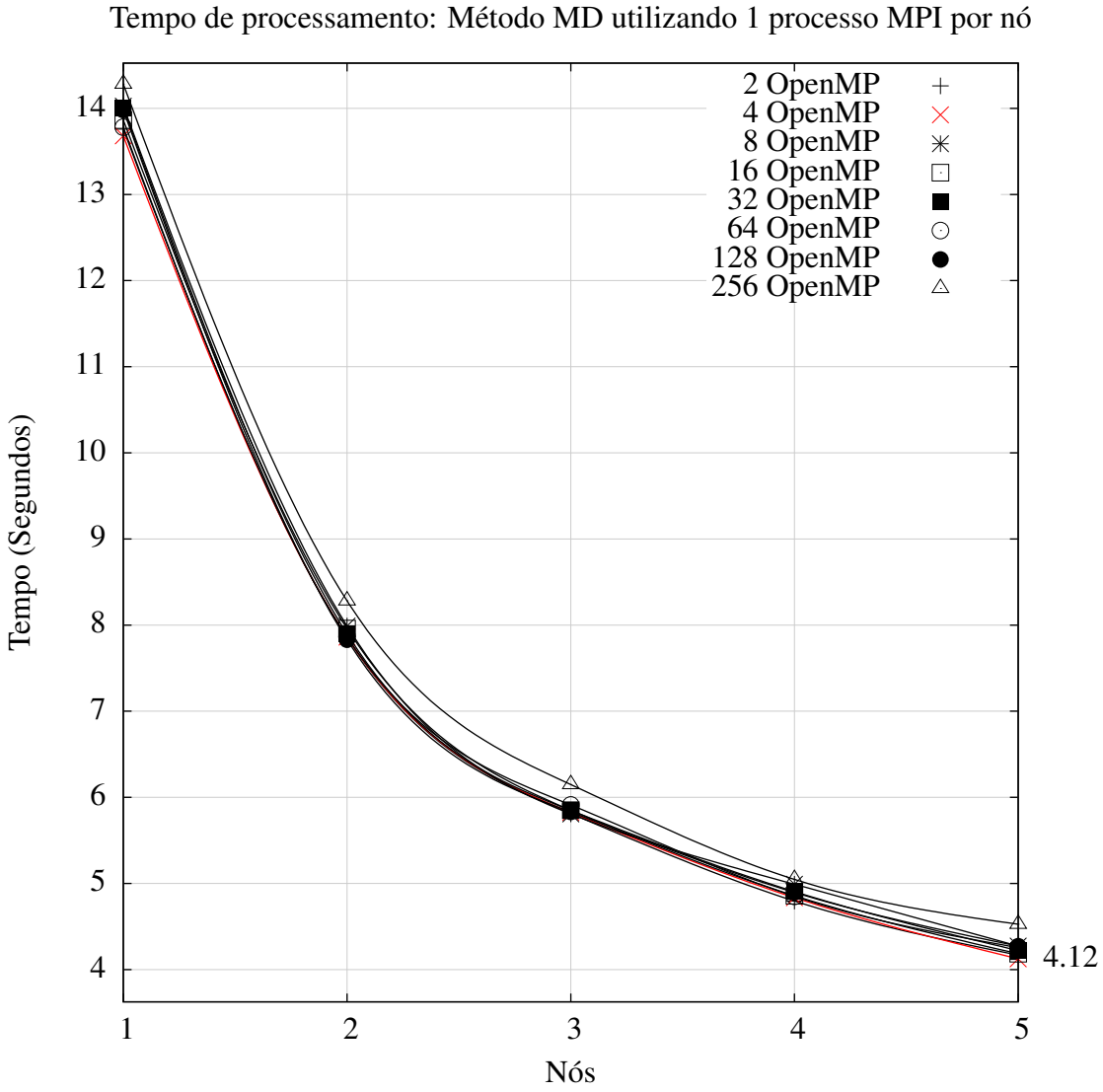

Figura 39: Tempo de processamento para o método *MD* utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com um processos *MPI* por nó.

O teste da Figura 40 mostra como o foi o desempenho do método MD utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 4 processos MPI por nó com um total de 5 nós. A linha com o $+$ que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com $\times$; utilizando 8 *threads* OpenMP está marcada com $\ast$; a abordagem com 16 *threads* está marcada com $\square$; com 32 *threads* marcado com $\blacksquare$; 64 *threads* está marcado com $\bigcirc$; 128 *threads* esta destacada com $\bullet$; e 256 *threads* marcado com $\triangle$. A linha de 2 *threads* está destacada em vermelho por ter alcançado o menor tempo em 2,74 segundos.

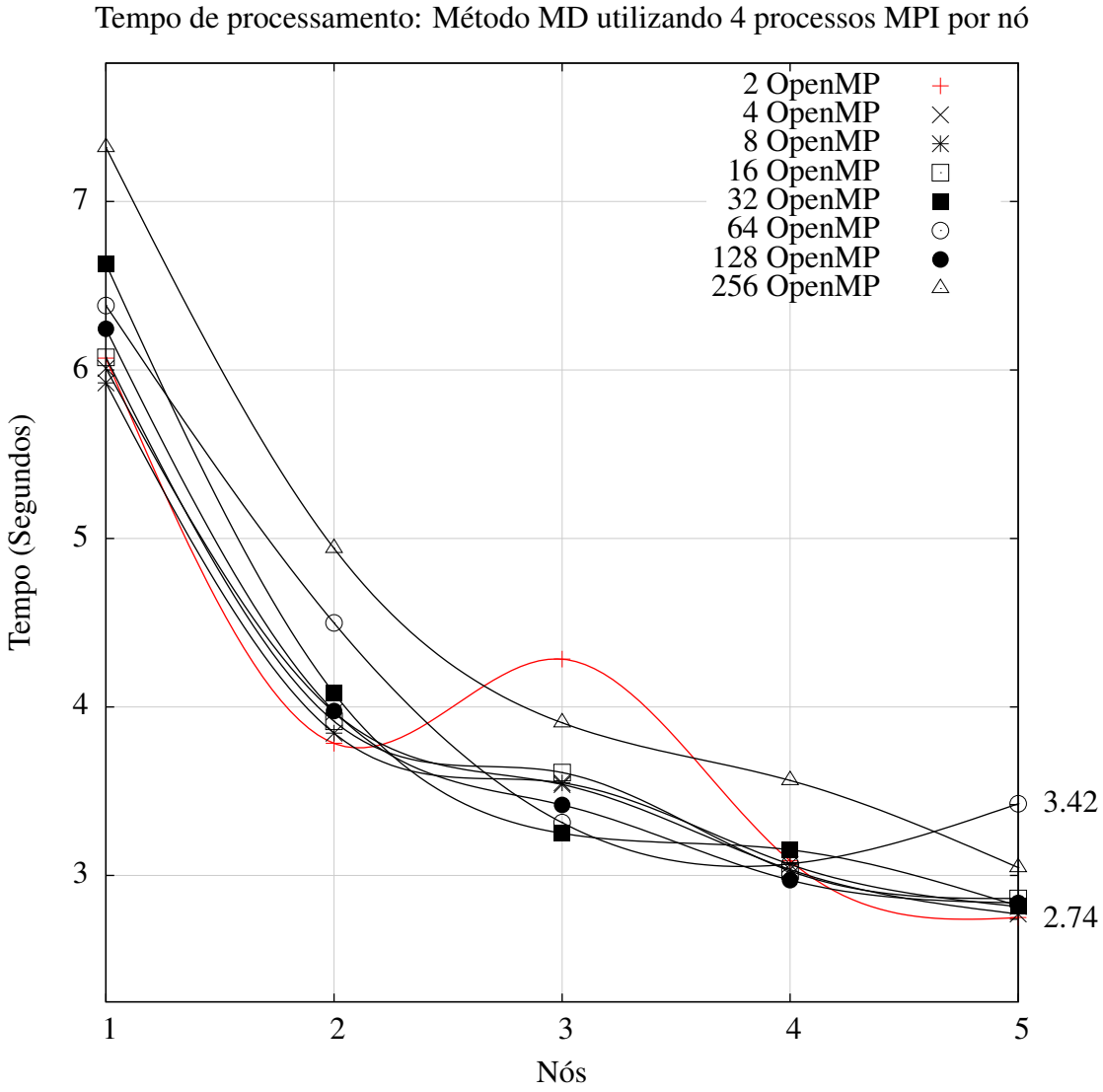

Figura 40: Tempo de processamento para o método *MD* utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com quatro processos MPI por nó.

O teste da Figura 41 mostra como o foi o desempenho do método FDD utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 8 processos MPI por nó com um total de 5 nós. A linha com o $+$ que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com $\times$; utilizando 8 *threads* OpenMP está marcada com $\ast$; a abordagem com 16 *threads* está marcada com $\square$; com 32 *threads* marcado com $\blacksquare$; 64 *threads* está marcado com $\bigcirc$; 128 *threads* esta destacada com $\bullet$; e 256 *threads* marcado com $\triangle$. A linha de 64 *threads* está destacada em vermelho por ter alcançado o menor tempo em 2,25 segundos.

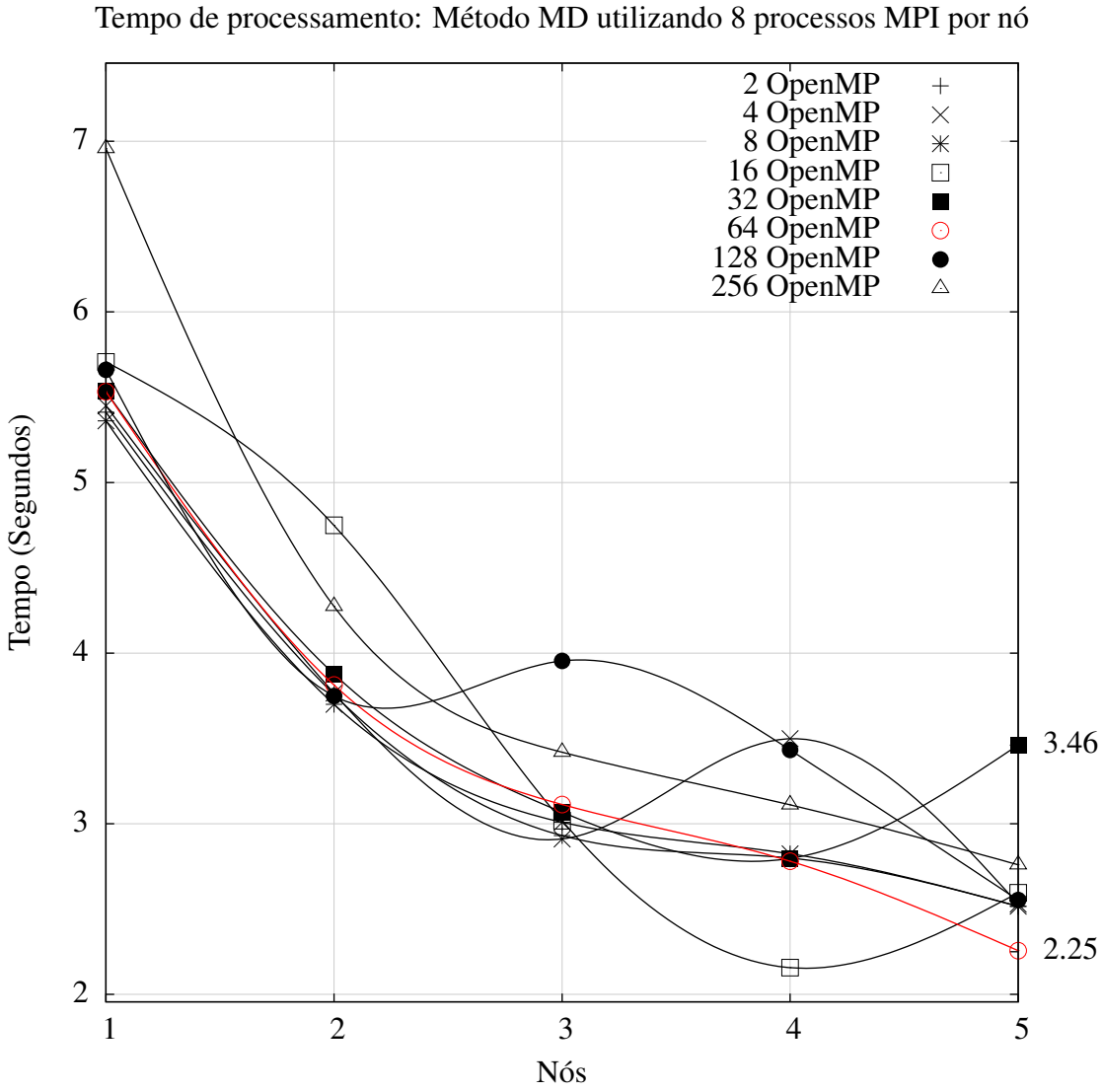

Figura 41: Tempo de processamento para o método *MD* utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com oito processos MPI por nó.

6.1.2 Detecção de áreas brilhantes

Os testes com para a detecção de áreas brilhantes, *HArea*, utilizaram a mesma abordagem utilizada para o método FDD. A Figura 42 mostra como o foi o desempenho do método *HArea* utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com apenas um processo MPI por nó com um total de 5 nós.

Na Figura 42, a linha com o + que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com ×; utilizando 8 *threads* OpenMP está marcada com ✳; a abordagem com 16 *threads* está marcada com □; com 32 *threads* marcado com ■; 64 *threads* está marcado com ○; 128 *threads* esta destacada com ●; e 256 *threads* marcado com △. A linha de 128 *threads* está destacada em vermelho por ter alcançado o menor tempo em 8,33 segundos.

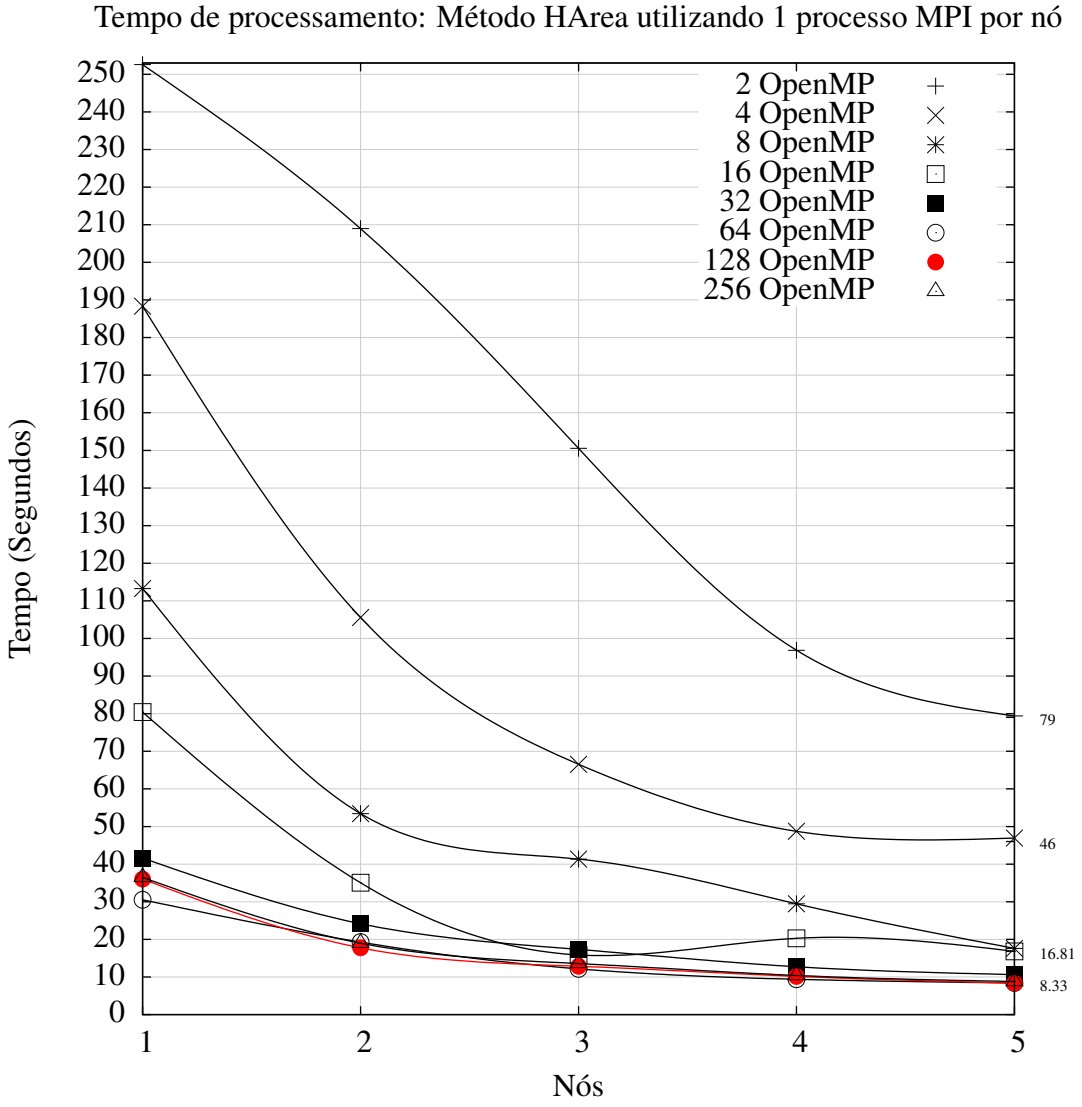

Figura 42: Tempo de processamento para o método para detecção de áreas brilhantes do Sol utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 1 processo *MPI* por nó.

O teste da Figura 43 mostra como o foi o desempenho do método *HArea* utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 4 processos MPI por nó com um total de 5 nós. A linha com o + que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com ×; utilizando 8 *threads* OpenMP está marcada com ✳; a abordagem com 16 *threads* está marcada com □; com 32 *threads* marcado com ■; 64 *threads* está marcado com ○; 128 *threads* esta destacada com ●; e 256 *threads* marcado com △. A linha de 128 *threads* está destacada em vermelho por ter alcançado o menor tempo em 4,82 segundos.

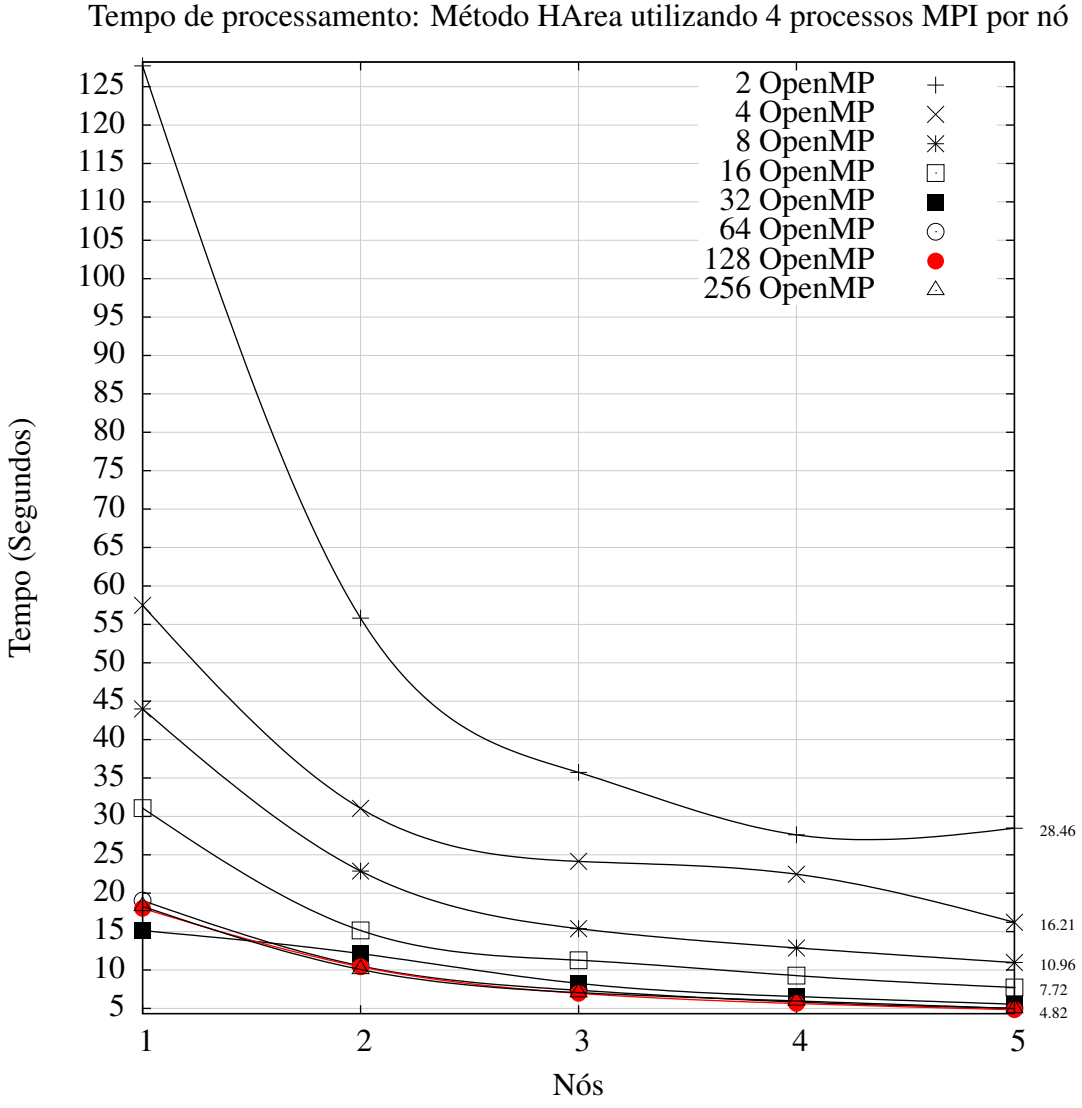

Figura 43: Tempo de processamento para detecção de áreas brilhantes do Sol 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com 4 processos MPI por nó.

O teste da Figura 44 mostra como o foi o desempenho do método *HArea* utilizando a combinação de 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com apenas um processo MPI por nó com um total de 5 nós. A linha com o $+$ que utiliza 2 *threads* OpenMP; com 4 *threads* OpenMP está marcada com $\times$; utilizando 8 *threads* OpenMP está marcada com $\ast$; a abordagem com 16 *threads* está marcada com $\square$; com 32 *threads* marcado com $\blacksquare$; 64 *threads* está marcado com $\bigcirc$; 128 *threads* esta destacada com $\bullet$; e 256 *threads* marcado com $\triangle$. A linha de 64 *threads* está destacada em vermelho por ter alcançado o menor tempo em 4,73 segundos.

Tempo de processamento: Método HArea utilizando 8 processos MPI por nó

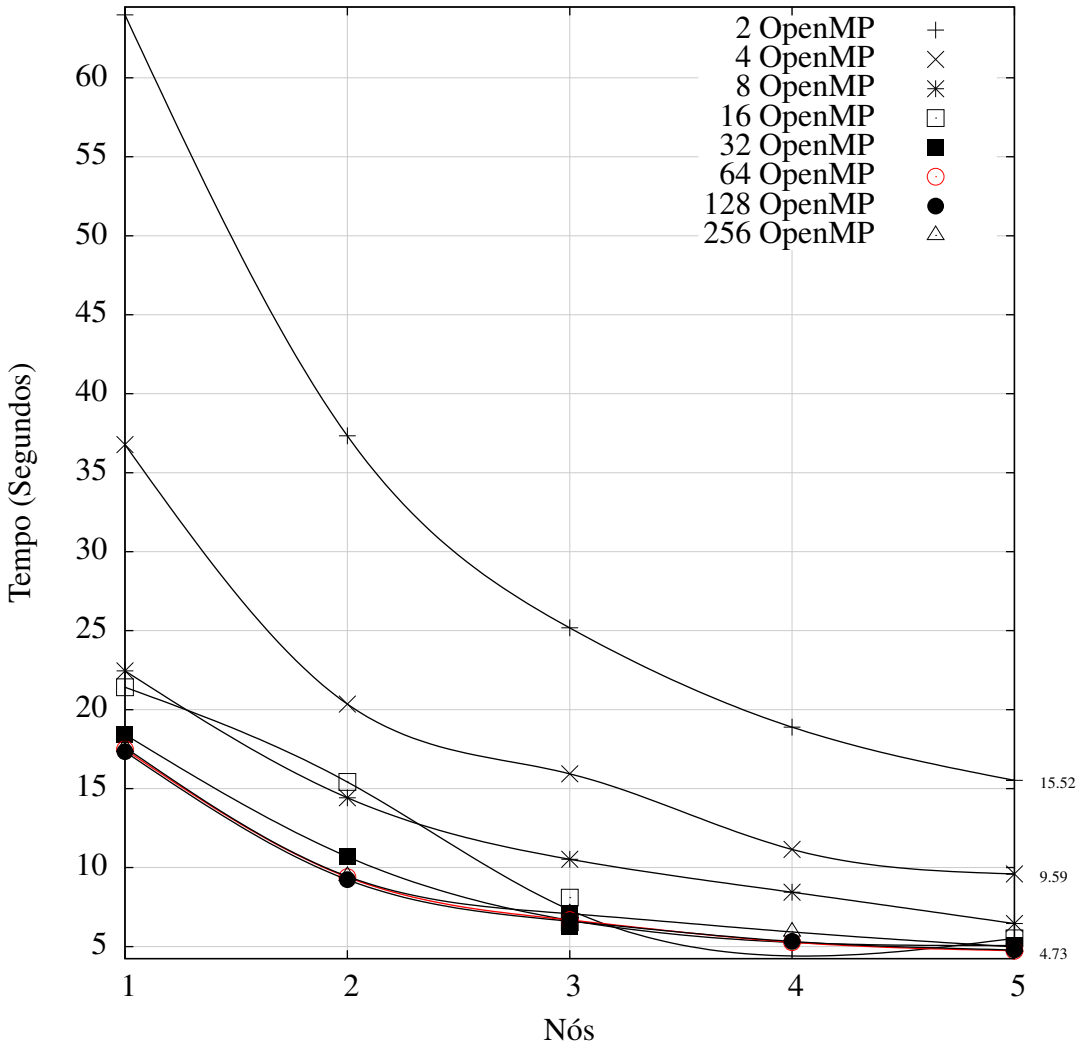

Figura 44: Tempo de processamento para detecção de áreas brilhantes do Sol utilizando 2, 4, 8, 16, 32, 64, 128 e 256 *threads OpenMP* com oito processos MPI por nó.

6.2 Discussão

Nesta seção discute-se os experimentos realizados. Os dois experimentos serão analisados de forma separada, pois se tratam de eventos diferentes.

6.2.1 Detecção de filamentos

Filamentos solares são importantes, pois têm forte relação com as ocorrências de CME. Utilizar programação de alto desempenho foi primordial para que fosse possível processar as imagens utilizando técnicas robustas e ainda utilizar todo o montante de dados disponíveis.

O método FDD com apenas uma *thread* utilizou 698,34 segundos. Com o melhor tempo de 5,8 segundos na abordagem de 32 *threads OpenMP*, com 5 nós e 8 processos MPI por nó foi alcançado um *speed-up* de 120 vezes, com mais de 80% de taxa de acerto (comparando *pixels* a *pixels* com a *ground truth*) em um *cluster* com 5 nós. No método MD foi alcançado um *speed-up* de 54 vezes com taxa de acerto de 58% nos mesmos cinco nós.

Verificou-se que número de *threads* OpenMP deve considerar o tamanho de cada segmento de imagem a ser processada. Quando a quantidade de *threads* aumenta o tamanho dos segmentos diminui. Portanto, constatou-se que o número de *threads* OpenMP deve ser inversamente proporcional ao tamanho do segmento, a fim de minimizar a mudança de contexto e o tempo para a criação das *threads*. Também é importante, de acordo com os testes realizados, destacar que o número de *threads OpenMP* deve ser proporcional ao número de núcleos do processador.

O processamento de tais imagens e a quantidade de informação disponível demonstrou que utilizar técnicas de alto desempenho e processamento de imagem é possível e necessário. Mostrou inclusive que existem técnicas de processamento/análise de imagem direcionadas para as imagens astronômicas e tais técnicas são passíveis de paralelismos em arquiteturas híbridas.

6.2.2 Detecção de áreas brilhantes

A detecção das áreas mais brilhantes puderam indicar a aparição de diversos eventos, tais como *flares* ou *faculae*. Tais eventos estão ligados as CME. Como o algoritmo é bem simples não conseguiu detectar esses eventos. Porém, pode ser utilizada para gerar sementes para *Optimum-Path Forest* ou outro algoritmos de classificação. Para processar utilizando apenas um nó, foram necessários 705 segundos. Utilizando 5 nós e 64 *threads* foram consumidos 25 segundos. Isso implica em um *speedup* 149,04 vezes.

7 Conclusões

Existem muitas imagens astronômicas. Essa quantidade tende a aumentar a cada nova missão, sempre com maiores amostragens e quantizações. Esse volume de informação deve ser processado de forma rápida e precisa para que toda sua riqueza seja aproveitada. Com os resultados obtidos nesse projeto, foi possível apresentar um sistema que teve por objetivo analisar as imagens astronômicas. O sistema era composto por módulos responsáveis por armazenamento, processamento e exibição dessas informações. Esse sistema considerou sempre a questão do tempo utilizado para cada ação e na padronização de acesso aos dados.

Para padronizar o acesso aos dados, o formato de imagem verificou-se que deve ser escolhido de forma precisa. O FITS foi encontrado em todos os repositórios de imagens astronômicas e provou ser o padrão *de facto* para esse tipo de informação. Outro ponto da padronização é o acesso ao sistema. O sistema *Astronomus* garantiu que o processamento e disponibilização da informação fosse feito de forma automática. Constatou-se a existência de diversas técnicas que podem ser aplicadas em imagens solares para detectar eventos e melhorar a visualização, como por exemplo, transformações na escala dos pixels, detecção de filamentos, detecção de áreas brilhantes e outras. Segundo esse estudo, tais técnicas podem ser paralelizadas especialmente para sistemas híbridos. No entanto, para os sistemas híbridos é importante balancear a quantidade de processos *MPI* e *threads OpenMP* para obter um melhor desempenho. O tamanho de cada segmento de imagem enviado para o processo MPI necessita ser relacionado à quantidade de *threads OpenMP*. Essas quantidades impactaram diretamente no desempenho do sistema.

Para que as técnicas fossem adaptadas foi necessário atenção ao fato da grande riqueza de detalhes que essas imagens possuem. A utilização de diversas faixas de frequência provenientes do mesmo objeto puderam gerar grandes melhorias na visualização e até na detecção de eventos celestes. Algumas técnicas que foram listadas no Capítulo 3 já possuem objetivos específicos em imagens astronômicas e a implementação de tais técnicas em ambientes de alto desempenho tem grandes chances de serem proveitosa.

Porém, o processo para paralelizar técnicas direcionadas a imagens astronômicas deve ser feito com muita precisão, pois em diversos casos existe a utilização de janelas de *pixels*. Exemplos são os casos em que são necessárias ações para que todos os processos (que podem estar em máquinas separadas) tenham acesso a todas as informações necessárias e ainda recebam as modificações decorrentes de outras porções da imagem.

As técnicas de processamento de alto desempenho já possuem um legado e são muito bem estabelecidas nos ramos científicos e comercias. As técnicas de processamento direcionadas para processamento distribuído ou para o processamento em memória compartilhada utilizadas de forma separada não conseguiram um desempenho tão bom quanto são utilizadas de forma complementar, bem orquestrada. O sistema híbrido, por sua vez, possui vantagens que permitem o uso mais eficiente de todos os recursos. Essas vantagens consistem em utilizar técnicas para tratar o problema de forma específica em ambientes de memória compartilhada e distribuída. Todavia, detém uma complexidade maior porque foi necessário dividir o conjunto de informações diversas vezes.

O sistema Astronomus, implementado para automatizar o processamento de imagens astronômicas, possui capacidade de processar grandes volumes de informação. Em todos os passos do sistema existiu a preocupação na agilidade do processamento. Com isso, utilizando a capacidade computacional de forma plena tornou-se factível a redução do tempo de processamento. A interface principal do sistema consiste em um *software* que garante a independência, agilidade e automatização do processo. Este módulo propiciou que o processamento fosse feito automaticamente na forma de *scripts*. Dessa forma, um pesquisador pode informar parâmetros como uma posição espacial e fazer diversos processamentos e visualizações. O módulo de processamento tem capacidade para funcionar em um *cluster* simples ou até em um equipamento mais elaborado.

Pode-se processar uma imagem utilizando todo o *cluster* ou utilizar cada nó para processar uma ou mais imagens. Dessa forma o usuário pode adequar a suas necessidades. Novos filtros podem ser adicionados de forma simples e ainda é possível a construção de sequência de filtros tanto em relação ao *scriptGenerator* ou na própria aplicação.

7.1 Trabalhos futuros

Alguns aspectos do sistema podem ser explorados dando continuidade ao trabalho. Um módulo que leva em conta diversos fatores, tais como: ocupação do *cluster*, quantidade de imagens, *etc* para determinar quantos processos MPI e *threads OpenMP*. Além dessa escolha de *threads* e processos, a possibilidade de utilizar sistemas heterogêneos (CPU + *Graphics Processing Unit* – Unidade de Processamento Gráfico (GPU)). Outros filtros podem ser implementados para a detecção de diversos eventos do Sol, como: *flare*, *facule*, buracos coronais e outros. Com essa gama de filtros, um módulo automático poderia detectar e fazer análises completas de dados atuais e passados do Sol.

7.2 Produções Decorrentes da Pesquisa

Durante a pesquisa realizada para a confecção desse trabalho ocorreram algumas publicações e registro de *software*. Estas ações tinham o objetivo de criar um sistema mais completo para o processamento de imagens astronômicas, com o foco direcionado nas imagens solares.

1. Resumo expandido e poster na Escola Regional de Alto Desempenho de São Paulo (ERAD-SP 2011) (ANDRIJAUSKAS; GRADVOHL, 2011);
2. Artigo publicado e apresentado no Workshop on High-Performance Computing for Astronomy Data (*AstroHPC*) do *International Symposium on High Performance Distributed Computing* (*HPDC*) (ANDRIJAUSKAS; GRADVOHL, 2012);
3. Registro no INPI do *software Astronomus*;
4. Sistema de indexação de imagens astronômicas;
5. Sistema de *script* para recuperação, processamento e visualização (conversão para o *Microsoft WorldWide Telescope*) de imagens astronômicas.

Referências Bibliográficas

ACCOMAZZI, A. et al. *Astronomy and Computing: a New Journal for the Astronomical Computing Community*. out. 2012. Disponível em: <http://arxiv.org/abs/1210.8030>.

Aldus Corporation. *TIFF — Revision 6.0*. June 1992. Seattle, Washington.

ANDRIJAUSKAS, F.; GRADVOHL, A. L. S. Tratamento de imagens astronômicas utilizando arquiteturas de alto desempenho. *ERAD-SP 2011*, jul 2011.

ANDRIJAUSKAS, F.; GRADVOHL, A. L. S. Solar filaments detection using parallel programming in hybrid architectures. In: *Proceedings of the 2012 workshop on High-Performance Computing for Astronomy Date*. New York, NY, USA: ACM, 2012. (Astro-HPC '12), p. 41–48. ISBN 978-1-4503-1338-4.

ANDS J. ANDS, J. G. et al. Acceleration of a dwt-based algorithm for short exposure stellar images processing on a hprc platform. In: *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*. [S.l.: s.n.], 2010. p. 113 –116.

BARRA, V.; DELOUILLE, V.; HOCHEDEZ, J.-F. Segmentation of extreme ultraviolet solar images using a multispectral data fusion process. In: *FUZZ-IEEE 07*. [S.l.: s.n.], 2007. p. 1–6.

BERRIMAN, G. B.; GROOM, S. L. How will astronomy archives survive the data tsunami? *Communications of the ACM*, ACM, New York, NY, USA, v. 54, n. 12, p. 52–56, dez. 2011. ISSN 0001-0782.

BERRY, R. e. J. B. *The Handbook of Astronomical Image Processing*. 2º. ed. EUA: Willmann-Bell, Inc, 2005.

BOBIN, J.; STARCK, J.-L.; OTTENSAMER, R. Compressed sensing in astronomy. *Selected Topics in Signal Processing, IEEE Journal of*, v. 2, n. 5, p. 718 –726, oct. 2008. ISSN 1932-4553.

BRESHEARS, C. *The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications*. [S.l.]: O'Reilly, 2009. (O'Reilly Series). ISBN 9780596521530.

CALTECH. *NASA/IPAC Infrared Science Archive*. 2010. Disponível em: <http://irsa.ipac.caltech.edu/>.

CHANDRA, R. *Parallel programming in OpenMP*. [S.l.]: Morgan Kaufmann Publishers, 2001. (High performance computing). ISBN 9781558606715.

CHAPMAN, B.; JOST, G.; PAS, R. van van der. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. [S.l.]: The MIT Press, 2007. Paperback. ISBN 0262533022.

CRISTO, A.; PLAZA, A.; VALENCIA, D. A novel thresholding method for automatically detecting stars in astronomical images. In: *Signal Processing and Information Technology, 2008. ISSPIT 2008. IEEE International Symposium on*. [S.l.: s.n.], 2008. p. 180 –185.

CUI, M. et al. Preliminary study of the influence caused by solar storm on sichuan power grid. In: *Power System Technology (POWERCON), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 1 –6.

DUFFETT-SMITH, P. *Practical Astronomy with your Calculator*. [S.l.]: Cambridge University Press, 1989. ISBN 9780521356992.

ESO, E. S. O. *The Very Large Telescope*. February 2011. Disponível em: <http://www.eso.org/public/teles-instr/vlt.html>.

FALCAO, A.; STOLFI, J.; LOTUFO, R. de A. The image foresting transform: theory, algorithms, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 26, n. 1, p. 19 – 29, jan 2004. ISSN 0162-8828.

FITS-OFFICE, A. D. F. A user's guide for the flexible image transport system (fits). 1997.

FOSBURY, R. Lars Lindberg Christensen e. *Hubble: 15 years of discovery*. [S.l.]: Springer, 2006. ISBN 9780387285993.

GAO, J.; WANG, H.; ZHOU, M. Development of an automatic filament disappearance detection system. *Solar Physics*, Springer Netherlands, v. 205, p. 93–103, 2002. ISSN 0038-0938. 10.1023/A:1013851808367.

GEBALI, F. *Algorithms and Parallel Computing*. [S.l.]: John Wiley & Sons, 2011. (Wiley Series on Parallel and Distributed Computing). ISBN 9780470902103.

GONZALEZ, R. C.; WOODS, R. E. *Digital image processing*. 3º. ed. EUA: Prentice-Hall, 2007.

GREISEN, E. W.; CALABRETTA, M. R. Representations of world coordinates in FITS. *Astron. Astrophys.*, v. 395, p. 1061–1075, dez. 2002.

HAGER, G.; WELLEIN, G. *Introduction to High Performance Computing for Scientists and Engineers*. [S.l.]: Taylor & Francis, 2010.

HAGER, G. J. G.; RABENSEIFNER, R. Communication characteristics and hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. *Proc. Cray User Group Conf.*, 2009.

HERLIHY, M.; SHAVIT, N. *The art of multiprocessor programming*. [S.l.]: Elsevier/Morgan Kaufmann, 2008. (Safari Books Online). ISBN 9780123705914.

IAC, I. d. A. d. C. *Gran Telescopio CANARIAS(GTC)*. February 2011. Http://www.gtc.iac.es/en/.

INGLIS, M. *Astrophysics is easy!: an introduction for the amateur astronomer*. [S.l.]: Springer, 2007. (Patrick Moore's practical astronomy series). ISBN 9781852338909.

INPE. *EMBRACE - Estudo e Monitoramento Brasiliero do Clima Espacial*. 2011. Http://www.inpe.br/climaespacial/.

INTEL. *Cluster OpenMP for Intel compilers*. 2010. Http://software.intel.com/en-us/articles/cluster-openmp-for-intel-compilers.

JÄHNE, B. *Digital Image Processing*. [S.l.]: Springer, 1997. ISBN 9783540240358.

JENKINS, J. *The Sun and How to Observe It*. [S.l.]: Springer, 2009. (Astronomers' Observing Guides). ISBN 9780387094977.

KAISER, N. Moore's law takes on the universe; new astronomy with giga-pixel imagers and peta-byte data archives. In: *Aerospace conference, 2009 IEEE*. [S.l.: s.n.], 2009. p. 1 –2.

LAVANYA, A. et al. Image fusion of the multi-sensor lunar image data using wavelet combined transformation. In: *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 920 –925.

LAWRENCE, A. Drowning in data : VO to the rescue. *Dot Astronomy*, maio 2009.

LBTC, L. B. T. C. *Large Binocular Telescope Observatory*. Fevereiro 2011. Disponível em: <http://lbtwww.arcetri.astro.it/>.

MAINZER, A. K. et al. *Preliminary Design of The Wide-Field Infrared Survey Explorer (WISE)*. [S.l.], Aug 2005.

MATTHEW, N.; STONES, R. *Beginning Linux Programming*. [S.l.]: John Wiley & Sons, 2011. ISBN 9781118058619.

MITCHELL, M.; OLDHAM, J.; SAMUEL, A. *Advanced Linux programming*. [S.l.]: New Riders, 2001. (Landmark Series). ISBN 9780735710436.

MOCHÉ, D. L. *Astronomy - A self-teaching Guide*. 7º. ed. EUA: Wiley, 2009.

MOORE, S. Extreme solar storm strikes earth. *Spectrum, IEEE*, v. 40, n. 12, p. 15 –16, 2003. ISSN 0018-9235.

MORISON, I. *Introduction to Astronomy and Cosmology*. [S.l.]: Wiley, 2008. (Manchester Physics Series). ISBN 9780470033333.

MUELLER, D. et al. JHelioviewer - visualizing large sets of solar images using JPEG 2000. jun. 2009. Disponível em: <http://arxiv.org/abs/0906.1582>.

NUNEZ, J.; FORS, O.; PRADES, A. Using image deconvolution to increase the ability to detect stars and faint orbital objects in ccd imaging. In: *Digital Image Processing, 2009 International Conference on*. [S.l.: s.n.], 2009. p. 195 –199.

PENCE, W. CFITSIO, v2.0: A New Full-Featured Data Interface. In: D. M. Mehringer, R. L. Plante, & D. A. Roberts (Ed.). *Astronomical Data Analysis Software and Systems VIII*. [S.l.: s.n.], 1999. (Astronomical Society of the Pacific Conference Series, v. 172), p. 487.

PENCE, W. D. et al. Definition of the flexible image transport system (fits), version 3.0. *A&A*, v. 524, p. A42, 2010.

POWELL, M.; ROSSI, R.; SHAMS, K. A scalable image processing framework for gigapixel mars and other celestial body images. In: *Aerospace Conference, 2010 IEEE*. [S.l.: s.n.], 2010. p. 1 –11. ISSN 1095-323X.

QIN, L. et al. A more precise method of sky subtraction in sdss astronomical image processing. In: *Image and Signal Processing (CISP), 2010 3rd International Congress on*. [S.l.: s.n.], 2010. v. 5, p. 2458 –2460.

RABENSEIFNER, R. Hybrid parallel programming on hpc platforms. In: *5th European Workshop on OpenMP*. [S.l.: s.n.], 2003. p. 185–194.

RABENSEIFNER, R.; HAGER, G.; JOST, G. Hybrid MPI/OpenMP parallel programming on clusters of Multi-Core SMP nodes. *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 427–436, 2009. ISSN 1066-6192.

RIDPATH, I. *A dictionary of astronomy*. [S.l.]: Oxford University Press, 2007. (Oxford paperback reference). ISBN 9780199214938.

ROY, A.; CLARKE, D. *Astronomy: principles and practice*. [S.l.]: Institute of Physics Pub., 2003. ISBN 9780750309172.

RUEFFER, P.; JASPERS, J.-P. Image data compression scheme for a future mars lander. In: *Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International*. [S.l.: s.n.], 2007. p. 1037 –1040.

SHIH, F. Image processing and pattern recognition:fundamentals and techniques. In: ____. 1. ed. [S.l.]: Wiley-IEEE Press, 2010. cap. Solar image processing and analysis, p. 496 –533.

STARCK, J.; MURTAGH, F. *Astronomical image and data analysis*. [S.l.]: Springer, 2006. (Astronomy and astrophysics library). ISBN 9783540330240.

STARCK, J.-L.; PANTIN, E.; MURTAGH, F. Deconvolution in astronomy: a review. *PASP*, v. 114, p. 1051–1069, 2002.

TELESCOPE, M. W. *WorldWide Telescope Data Files Reference*. 2012. Http://www.worldwidetelescope.org/Docs/WorldWideTelescopeDataFilesReference.html.

TODY, D.; PLANTE, R.; HARRISON, P. *IVOA Recommendation: Simple Image Access Specification Version 1.0*. [S.l.], Oct 2011.

TROGLIO, G. et al. Crater detection based on marked point processes. In: *Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International*. [S.l.: s.n.], 2010. p. 1378 –1381. ISSN 2153-6996.

WALLACE, G. The jpeg still picture compression standard. *Consumer Electronics, IEEE Transactions on*, v. 38, n. 1, p. xviii –xxxiv, fev. 1992. ISSN 0098-3063.

WELLS, E. W. G. D. C.; HARTEN, R. H. Fits: A flexible image transport system. *ASTRONOMY & ASTROPHYSICS SUPPLEMENT SERIES*, v. 44, p. 363, 1981.

WMKO, W. M. K. O. *W. M. Keck Observatory*. February 2011. Disponível em: <http://www.keckobservatory.org/>.