Gaspra - software for astronomical image processing in high performance computing clusters

Fábio Andrijauskas [1]
André Leon Sampaio Gradvohl [1]

Resumo: Há uma enorme quantidade de informação gerada por telescópios como imagens astronômicas. Portanto, software e equipamentos podem processar essas imagens para encontrar novos fenômenos e obter novos conhecimentos sobre o espaço. Considerando a necessidade de um rápido processamento dessas imagens, apresentamos um software para processamento de imagens astronômicas em clusters de computação de alto desempenho, que utilizam sistemas de memória compartilhada e distribuída, chamado Gaspra. Projetamos o Gaspra para processamento em lote de grandes conjuntos de imagens astronômicas, permitindo aos pesquisadores criar workflows científicos para obter novos conhecimentos a partir desses conjuntos de dados de imagens do espaço. Experimentos com Gaspra mostram um aumento de velocidade de 3,5 vezes para processar uma única imagem em cinco nós de processamento, suportando, cada nó, 64 *threads* distintos.

***Palavras-chave*:** Imagens astronômicas. Processamento de imagens. Processamento de alto desempenho.

Abstract: There is a huge amount of information generated by telescopes as astronomical images. Therefore, software and equipment could process these images to find new phenomena and obtain new knowledge about the space. Considering the need of rapid processing of those images, we present a software for astronomical image processing in high-performance computing clusters, which use shared and distributed memory systems, called Gaspra. We designed Gaspra to batch process large sets of astronomical images, allowing researchers to create scientific workflows to obtain new knowledge from these astronomy imagery data sets. Experiments with Gaspra show a 3.5-fold speedup to process a single image in five processing nodes, each node supporting 64 different threads.

***Keywords*:** Astronomical images. Image processing. High-performance computing.

1 Introduction

The volume of information generated as astronomical images is constantly growing. The National Aeronautics and Space Administration (NASA), the National Laboratory for Astrophysics and other space agencies have plans for missions to gather space images, despite other active projects, which are still producing images from outer space. This huge volume of visual information should receive scientific attention. Due to several techniques that can improve, segment and analyze these images, we can obtain more information and knowledge from them.

As the amount of generated data increases, automatic processes become necessary to verify the relevance of certain information sets. The goal is to determine whether those images should be stored or not, since we must invest on large storage systems wisely.

On the other hand, techniques for image processing and analysis require large processing time. If we multiply that time by the number and size of images from space, the total processing time (wall time) may be unfeasible. Therefore, combining high-performance computing algorithms and image processing techniques, we can perform fast analyzes and improvements in space images in shorter time, making the process feasible. Additionally, the probability of finding phenomena (such as exploding stars or solar flares) increases with the implementation of image operations frequently used for astronomical images.

[1]School of Technology - University of Campinas. R. Paschoal Marmo, 1888 - CEP:13484-332 - Jd. Nova Itália - Limeira-SP, Brazil
{fabio.andrijauskas@pos.ft.unicamp.br; gradvohl@ft.unicamp.br}

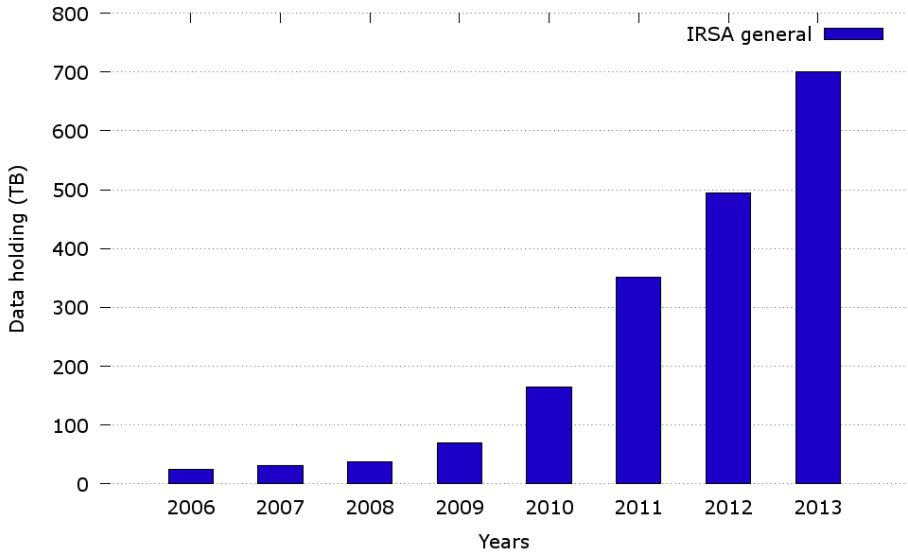
http://dx.doi.org/10.5335/rbca.2014.4072

Recent research about astronomic investigation highlight the data tsunami in this field. Lawrence [1] indicates that there will be a large volume of data and points the need for strategies to deal with that huge volume. Berriman and Groom [2] and Jones et al. [3] indicate a major challenge in the treatment of such information. They argued that the volume of stored data already surpasses one petabyte and grows about half petabyte per year. Therefore, Astronomy needs techniques to address big data issues.

Another example that shows the growth of the astronomical amount of processed data is the Solar Dynamic Observatory (SDO) project from NASA. SDO produces images with 6 megapixels every 10 seconds. This corresponds to 1.4 terabytes of information per day [4].

The InfraRed Science Archive (IRSA) at California Technology Institute is responsible to curate the science products of NASA's infrared and submillimeter missions, including many large-area and all-sky surveys. Missions such as WISE, Spitzer, 2Mass, IRAS and Plank generated in 2011, about 350 terabytes of images. In 2013, the IRSA itself has reached 700 terabytes of stored images and this number grows exponentially, as depicted in Figure 1.

Figure 1: The amount of data stored in IRSA from 2006 to 2013



Considering those aspects, we present Gaspra as a software developed for retrieving, processing, and displaying relevant information from astronomical images. Gaspra produces images files in Portable Network Graphics (PNG) format or files that the World Wide Telescope (WWT) can display. WWT is a Web 2.0 visualization software environment, which brings together imagery from the world's ground- and space-based telescopes for universe exploration [5].

Gaspra runs on high-performance computing (HPC) hybrid architecture, i.e. an architecture with shared and distributed memory systems [6]. Therefore, Gaspra focus on the e-science petascale Astronomy era, since it merges HPC with Astronomy. Incidentally, Gaspra is an S-type asteroid that orbits very close to the inner edge of the solar system asteroid belt.

1.1 Related work

There are other software to process astronomical images, such as MaxIm DL, PixInsight, ImagesPlus, AIP4Win, Nebulosity2, Iris, Astroart, Registax, AstroStack 3, K3 DS9 and Image Reduction and Analysis Facility (IRAF). These software contain a collection of algorithms for image processing and analysis. However, not all of them work on multicore architectures and none of them uses hybrid architectures. For this reason, these software cannot produce on a large scale, i.e. they cannot handle a long series of astronomical images in batch.

In turn, Powell [7] reports a processing strategy that divides images into tiles to increase efficiency and

scalability. Nevertheless, this strategy does not use hybrid systems.

Another interesting software is Montage [8], which allows the construction of mosaics of space imagery. Montage has similar features comparing with the Gaspra, including the use of high-performance processing with shared memory systems. Still, Montage does not use hybrid architectures, because it does not work on shared memory systems.

SkyQuery software, in other hand, consists of a federate database for astronomical data [9]. This federate database integrates several databases with information access features similar to Gaspra. Yet, SkyQuery does not integrates the techniques of image processing, event detection and hybrid memory systems. Thus, Gaspra distinguishes it by the integration of astronomical images processing in batch and high-performance computing. This feature allows researchers to automate the process by creating a scientific workflow to process astronomical images, and free them to work into high-level scientific tasks.

Gaspra has other features: it stores the processing time of the filters for better tuning of computational processing power (number of processing cores and processing nodes) and generates performance charts. Moreover, Gaspra send notices and processed data to web servers, and retrieves the most recent images from Global Oscillation Network Group (GONG – a community-based program to conduct a detailed study of solar internal structure and dynamics using helioseismology). Gaspra also reports position, size and area of detected solar flares.

Additionally, to present Gaspra, we organize this paper as follows: Section 2 presents the architecture of Gaspra software, including its modules and their integration; Section 3 shows how to use Gaspra and reports some results using it; and Section 4 presents the conclusion and future directions.

2 The Gaspra software

At a glance, the Gaspra software has six components designed to retrieve, process and display the information concisely and rapidly. The retrieval phase access information in a standardized and secure way. The image treatment phase improves the quality of image and extracts relevant information. The preview phase, in turn, provides means for user visualization of information with artifacts to display such data.

2.1 Software architecture

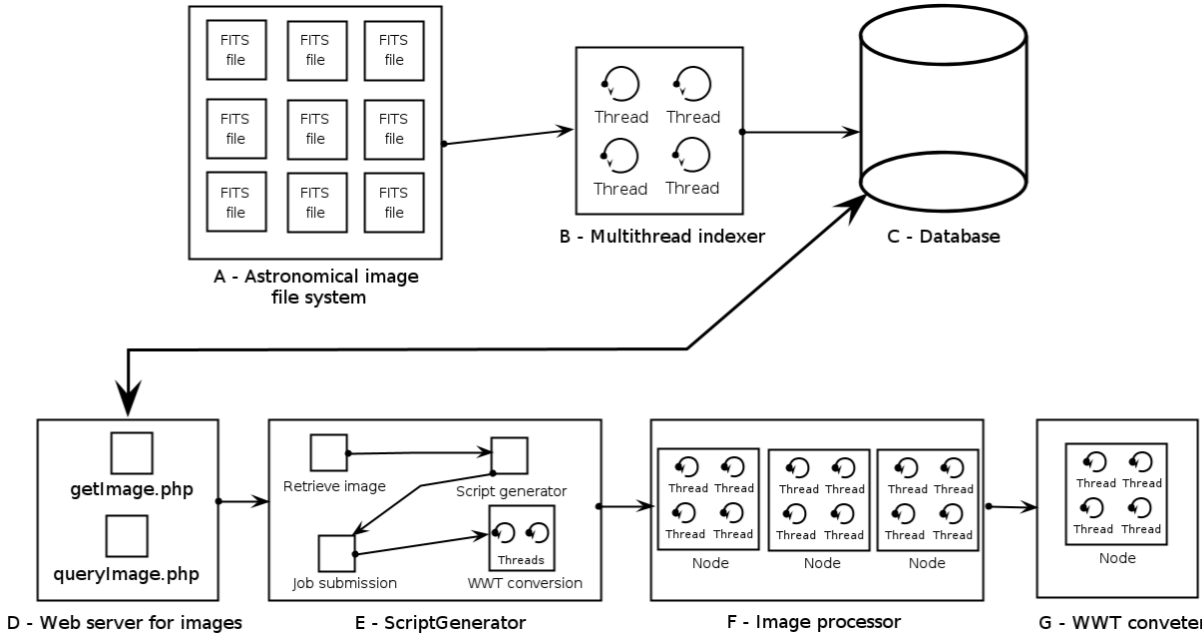
Figure 2 depicts the architecture of the Gaspra software that has five modules implemented as follows.

The file system device (Figure 2-A) consists of a high storage capacity, low latency, and high transmission rate using file types adherent to the nature of the data. It is necessary to use an image format that supports the richness and variation of astronomical images. The format used is the Flexible Image Transport System (FITS) [10], which is a specific image format recommended by NASA and International Astronomical Union to store astronomical images. Thus, all images stored on the device are in FITS format.

However, to standardize access, avoid concurrency problems and maximize the speed on fetching the images, the Multithreaded Indexer (Figure 2-B) will catalog these using the database (Figure 2-C). After indexing, the web server can access them (Figure 2-D).

We can access images manually or by software. An example of access by software is the ScriptGenerator on Figure 2-E. The ScriptGenerator retrieves images through the web server to filter, segment or perform other operation on FITS files using the Image Processor (Figure 2-F). To run these operations, ScriptGenerator automatically writes a job script, which runs on a computer cluster with a queue system (e.g. IBM Load Leveler or Portable Batch System) or even in a small cluster with the Message Passing Interface (MPI) library installed on. The ScriptGenerator also triggers the indexing module or generates the files to publish at WWT. Furthermore, it sends the images to the WWT conversion (Figure 2-G), which, in turn, makes image conversion, enabling the image display at WWT.

Figure 2: Gaspra software architecture. A-Astronomy images file system. B-Multithread indexer. C-Database. D-Web server for images. E-Script Generator. F-Image processor. G-WWT conversion



2.2 Multithread indexer

The Multithread Indexer consists in a pool of threads that extracts FITS files headers, analyzes them, makes necessary conversions, saves the information in the database and, at last, copies the images to the destination directory. It uses modules Astronomy images file system and Multithread indexer (Figs. 2-A and 2-B) to produce indexing.

These cataloging processes of images access the FITS files using the `cfitsio` library. After opening a file, it reads the header, which has much of the image's information, such as location and date, bits per pixel and others [11].

One of the most important data is the image position. We should standardize these data to provide access information for the coordinate system. The standardization process consists in conversion of all Right Ascension (RA) positions in decimal hours and Declination (DEC) in degrees, following conversions described in [12].

However, since the amount of data is too big, it is necessary to make the process as fast as possible. Then, the indexer uses a set of threads and a set of connections to use the full computing power available to minimize the time of indexing.

This subsystem uses the OpenMP library to produce the multiple threads of indexing. Each thread maintains a connection to the database. Thus, there is a one-to-one relationship between thread and connections. This relationship increases the demands for the database system and network connections.

2.3 Web server

The web server acts as a gateway to retrieve images. Apache Web PHP and MySQL database handle the requests for images. The service is based on the standards of the International Virtual Observatory Alliance, specifically in the Simple Image Access Specification [13].

This server receives HTTP requests (named GET requests) where the variables are dynamic defined. In other words, when we inform a variable in the uniform resource locator address, the server does the searches based on the names and values of the variable transmitted. Thus, if the Indexing Service of the images adds more information, there is no need to change the method `getImage` (from HTTP requests).

For instance, consider the request as the URL http://url.net/SIAP/getImage.php?naxis1=568. Then the system searches an image in the database that has axis 1 with 568 points. In addition to this dynamic way, the system accepts requests informed as RA and DEC positions or image size.

2.4 The ScriptGenerator

The Script Generator has several roles, such as recovers images through web server; generates a script configuring how many threads and processes should be used in the cluster to process the images; submits the job to cluster queue system; prepares images for to be displayed on WWT; and triggers the module for image indexing. Figure 2-D depicts these six tasks.

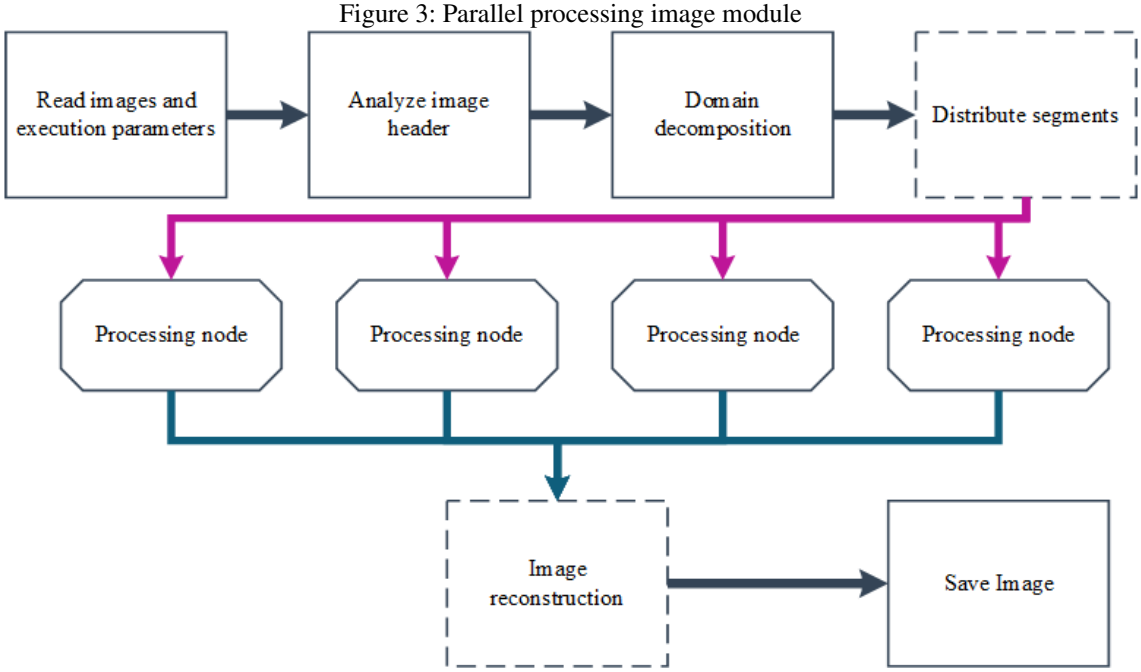
The image retrieval phase is done by calling the `wget` program. When the user retrieves a set of predefined files or a set of images, the system generates a script for each image to be processed.

After the creation of scripts, the cluster queue system receives the scripts and the images are processed. At this point, one can choose which algorithm to processes the images. On image processing phase, one could see the image in PNG format or send them to conversion and subsequent display in WWT. We use the Python programming language to program this module. Thus, a researcher can change or create scripts the way he or she desires.

2.5 Image processor

The module that processes images uses a hybrid system, which has shared and distributed memory systems. Figure 3 shows the diagram for this module.

Figure 3: Parallel processing image module



Given the size of the image and the respective volumes of data, it is important to use high-performance computing methods to reduce the overall processing time. Therefore, we use techniques for parallel programming hybrid systems, which are a combination of shared and distributed memory architectures. We use OpenMP for shared memory and MPI for distributed memory [14].

In a shared memory scenario, we divide the image into segments to apply filters and other processes. Therefore, we divide the image in latitudinal segments of equal length, and we send each one of these segments to the corresponding MPI process. We also send a small area of intersection to the MPI processes to provide information

for algorithms that use neighbor pixels for processing.

Figure 3 shows a diagram depicting the parallel application. In the first step, the application reads the image and some parameters for execution. After that, the application analyzes the image header to define the strategy used in domain decomposition for the image.

Then, the system splits the image and sends each segment to corresponding processes using non-blocking `MPI_Isend`. In turn, each OpenMP thread computes a segment. We parallelized the main loop, which iterates on image segments, using the OpenMP `#pragma omp parallel` annotation. In addition, other OpenMP annotations are used in algorithms to filter the brightest areas in images, such as `#pragma omp parallel for firstprivate` and `#pragma omp critical`.

The program receives each segment using non-blocking `MPI_Irecv`. Before the final reconstruct, the system uses a `MPI_Barrier` to synchronize each MPI process. Finally, the system saves it into the disk.

The ScriptGenerator is able to process images using the following algorithms in Gaspra: Binarization; Detection of the brightest areas of the image; Diffusion filter; Edge detection; Inversion of pixels; Median filter; Morphological filters (erosion and dilation in gray and binary images); Solar flares detection; Solar flares detection using morphological filters; and Transformation in pixel scale (logarithmic, exponential and normalization).

2.6 Converting to the WWT

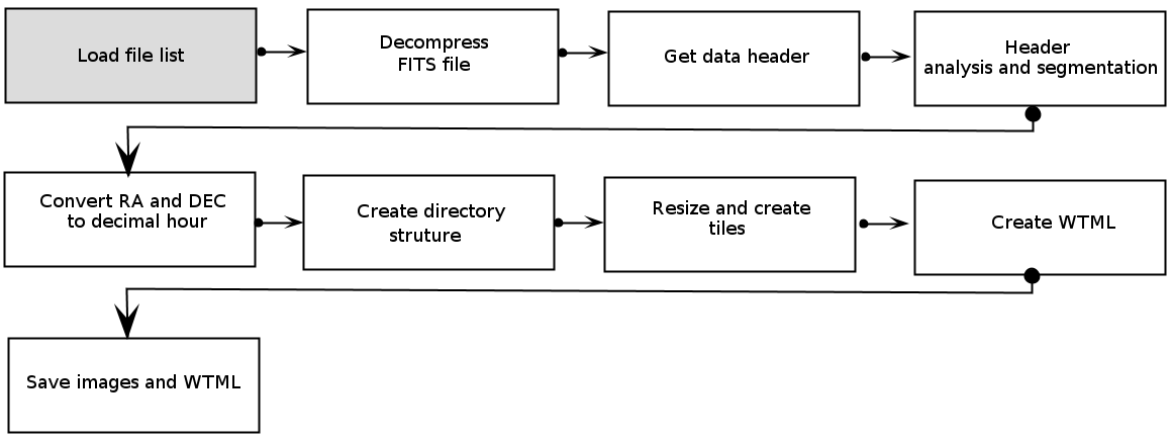
The conversion system uses several threads to process the images and prepare them to upload to WWT. A thread searches in a directory for predefined images and passes the job to other thread for conversion. This conversion creates a specific file for WWT (wtml file) and constructs the tiles. We used the basic functions of libpng (functions to read files, define pixels and save it to disk) to produce these tiles. Thus, this parallelization has a smaller granularity, allowing better use of computing power.

Figure 4 shows the procedure to create an image to display on the WWT. The first step in gray shade "Load file list" is processed in parallel, which means that different files are loaded simultaneously, and each image conversion is treated by one independent thread.

The "Decompress FITS file" and "Get data header" steps gather image information for the "Header analysis and segmentation" phase. After that, the "Convert RA and DEC to decimal hour" converts the right ascendant and the declination from the header image to decimal hour, if needed.

The "Create directory structure" assemblies the directory structure as used by the WWT. An essential step is the "Resize and create tiles", which segments and resizes the image tiles used on WWT. The last steps "Create WTML" and "Save images and WTML" creates the wtml files with the tiles recovered from the disk.

Figure 4: Procedure to create an image for display on WWT



3 Using Gaspra

The main interface of the system is the ScriptGenerator module, which integrates and triggers all system modules. We design the system to facilitate the integration and automation of indexing, retrieval and visualization.

There are three options to run the ScriptGenerator. The first option is to run the indexer from the command line, like "`./sg -i <dir> <#threads>`". Where the `<dir>` is the directory in which the images to be indexed are stored and `<#threads>` is the number of threads that will perform this process.

Another option is to transform images into the wtml files. To run this option, the user should replace the option `-i` by `-w` in the former command line. The result will be automatically stored at the WWT.

In addition to performing these actions independently, it is possible to create a Python script to automate the experiments, as the following code shows. The file should be stored at the "`src/experiments`" directory.

```
1  import os, time
2  import BDClass, GeraArquivoClass, ParLLClass, ParAstroClass, UtilClass
3
4  class Experimento:
5    id = "sun" # It must exist a single experiment with the following id
6    def roda(self):                    # Definition of the Experiment
7      util = UtilClass.Util            # Util class to help the image processing
8      astro = ParAstroClass.ParAstro() # Parameters for image processing
9      ll = ParLLClass.ParLL()    # Parameters to specify how to process the image
10
11     # Class to generate files that control the exec. of image processing
12     genFiles = GenFilesClass.GenFile()
13     while True:
14     util.getNSO(self) # Recover image from GONG
15     astro.qtyOpenMP = 12     # Process using 12 openmp threads
16     ll.node = 1              # Number of node to be used
17     ll.total_tasks = 2;      # Number of MPI process
18     astro.debug = 0          # Show debug info. during image proces. (0=no)?
19     astro.imgInput = '../file.fits' # Input image from GONG
20     astro.filter = 's'               # Apply Sobel filter
21     astro.outputPng = '../outputFits/filamentos.png' # Output PNG file name
22     astro.outputFits = '../outputFits/borda.fits'    # Output FITS file name
23
24     # Generate the file for LoadLeveler and execute in the cluster
25     genFiles.genFileLL(astro,ll,nome,1)
26     astro.filtro = 'le'                    # Apply Logarithmic scale filter
27     astro.outputPng = '../outputFits/escala.png'     # Output PNG file name
28     astro.outputFits = '../outputFits/escala.fits'   # Output FITS file name
29
30     # Generate the file for LoadLeveler and execute in the cluster
31     genFiles.genFileLL(astro,ll,nome,1)
32     astro.filter = 'd'                                # Filaments detection
33     astro.outputPng = '../outputFits/filamento.png'  # Output PNG file name
34     astro.outputFits = '../outputFits/filamento.fits'# Output FITS file name
35
36     # Generate the file for LoadLeveler and execute in the cluster
37     genFiles.genFileLL(astro,ll,nome,1)
38     util.sendGA(self)        # Send all png files to web server display
39     util.getRunData(self)    # Recover execution time
40     util.createGraph(self)   # Create time execution graph
41     util.sendEndEmail(self)  # Send by email the execution time graph
42     util.genWWT(self,'../outputFits/filamento.fits') # Create the WWT files
```

The user can specify the number of MPI process and the quantity of OpenMP threads per process as described in lines 17 and 15, respectively. With this simple change, the system can process the images with more or

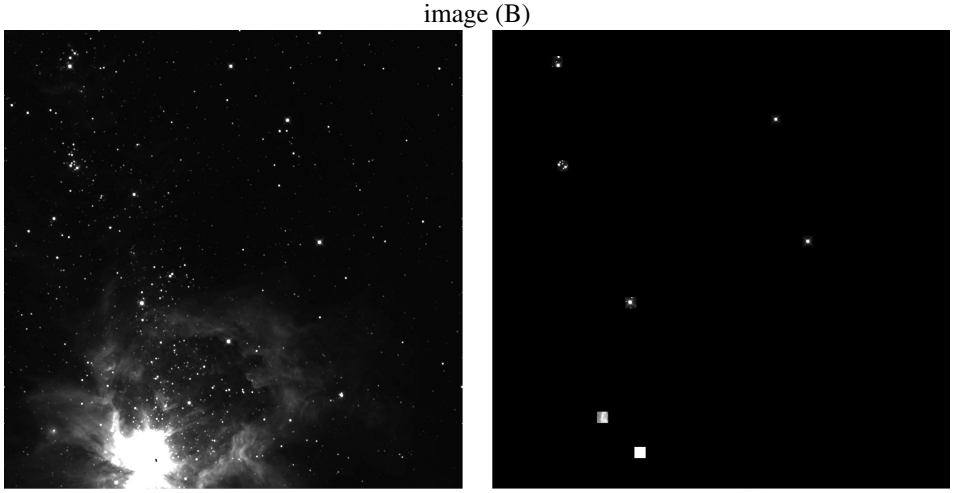
less threads and processes. This feature together with the ability to generate performance plots can help researchers to tune how much of the computer cluster should be used to maximize the performance. Using instructions as `util.getNSO` and `util.getImg` along Multithread Indexer provides access to large volume of information and methods to process this data.

3.1 Results

To present some results using Gaspra, we show two examples. For the first example, consider the image in Figure 5-A with 4095×4095 pixels. We use a computer cluster with five computing nodes, each one with four 3.0 GHz Power 755 processors, 12 GB memory, with Infiniband interconnection. This configuration is able to instantiate up to 32 simultaneous threads per processor, i.e. 8 cores per processor and 4 threads per core. The cluster has the following software installed: AIX 6.1 operating system, IBM XL C 1.11 compiler, MPI 2.1 and OpenMP 3.0 libraries, Python 2.6.7 interpreter and CFITSIO 3.29 library.

Using Gaspra, we call an algorithm that divides the image into areas and, using a median filter, displays only the segments of this area with greater brightness. In Figure 5-B, it is possible to identify the areas with the brightest spots of the image. These spots are the brightest considering a 32-bit floating point quantization. We compress the scale for displaying only.

Figure 5: Test image used to detect the areas with the brightest spots (A) and brightest spots detected in the image (B)



The chart depicted in Figure 6 shows the processing time for the brightest areas detection filter applied in an image. With hybrid processing, we used the ScriptGenerator to produce many configurations of threads, processes and number of processing nodes. We found that the best combination to minimize the processing time is to use 5 processing nodes (maximum number of nodes in our cluster), with 8 MPI processes per node and 64 threads for each MPI process. The graph in Figure 6 shows the processing time of sets with 2, 4, 8, growing exponentially up to 256 OpenMP threads for each MPI process.

For the second example, consider the image in Figure 7-A with 1024×1024 pixels and 32 bits per pixel, which depicts the solar activity. We acquired this image from the SDO on May 10, 2014, when a solar flare occurred with higher intensity than usual. Fig 7-B demonstrates the result of the image segmentation technique adapted from [15] and included in Gaspra to detect this sort of event. The black dots within the square in Figure 7-B indicate the occurrence of solar flares. The higher density of these points indicates greater intensity of the solar flares and consequently greater probability of coronal mass ejection.

Figure 8 shows the chart used to help the performance analysis. The experiment uses 8 MPI processes per node. Each MPI process uses 2, 4, 8 and growing exponentially up to 256 OpenMP threads. In this case, different from what we might expect, the best configuration for time uses 8 MPI processes, with 2 OpenMP threads for each

Figure 6: Time to process an astronomic image (Figure 5-A) using Gaspra, with 8 MPI processes in each processing node and a set of 2, 4, 8, 16, 32, 64, 128 and 256 OpenMP threads for each MPI process

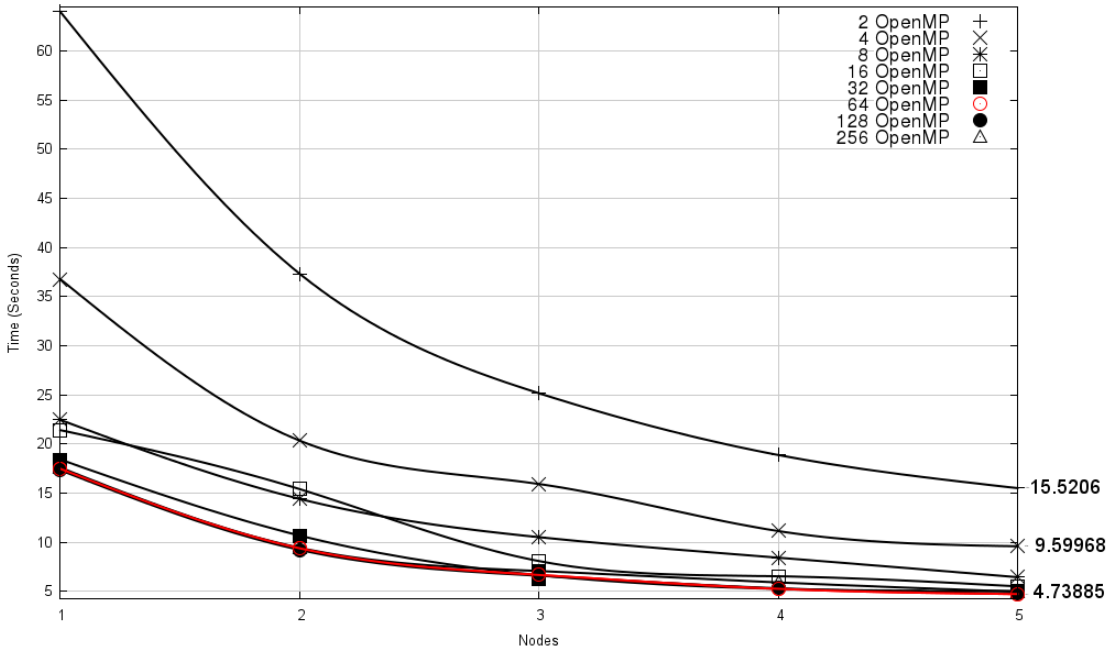
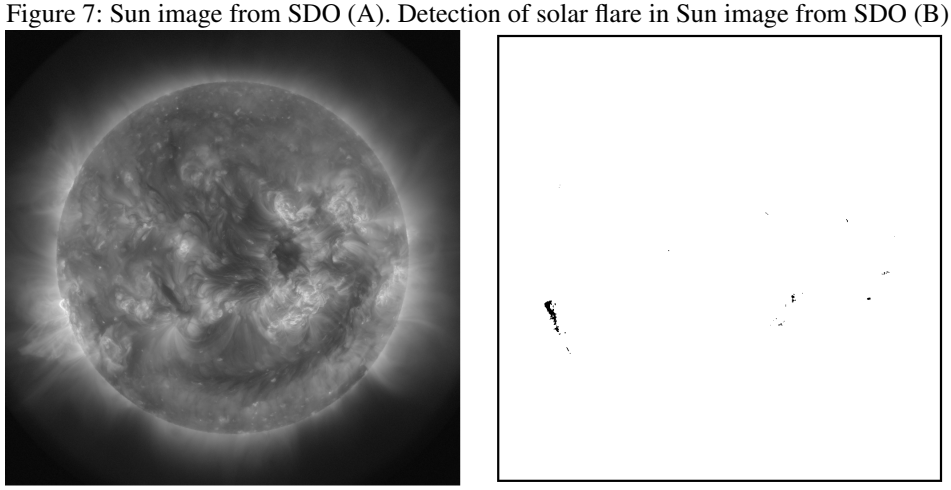


Figure 7: Sun image from SDO (A). Detection of solar flare in Sun image from SDO (B)



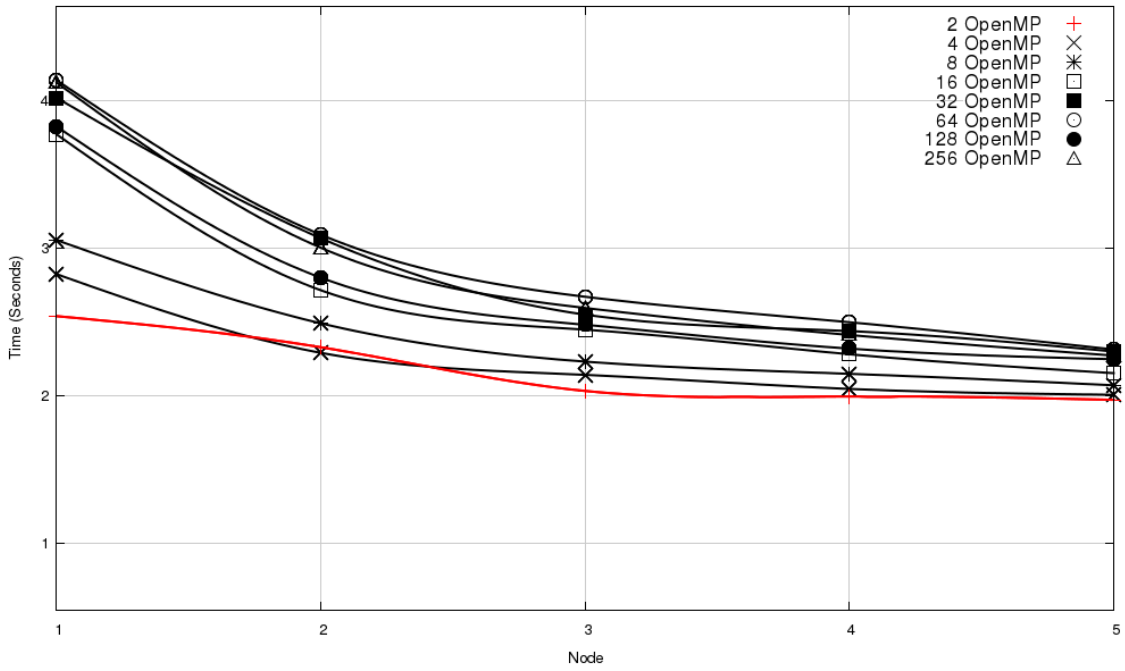
MPI process.

We analyzed the performance results for filament and brightest spots detection using two outputs. One output considers the speedup, while the other considers the system's throughput. The first output is useful for astronomical events that should be detected as soon as possible (solar storms, for instance).

However, other events or astronomical analysis could use more time to process more images. For that sort of event, we consider the system's throughput. For example, consider the filament detection in solar images using five nodes: the system processes one image within nearly 2 seconds. Nevertheless, processing five images (one per node) take approximately 2.5 seconds.

Gaspra generates performance plots, which let researchers to choose the best configuration for each sce-

Figure 8: Time to detect solar flares in the image of Figure 7-A using Gaspra, with 8 MPI processes in each processing node and a set of 2, 4, 8, 16, 32, 64, 128 and 256 OpenMP threads for each MPI process



nario. We achieved better speedup-folds using Gaspra with other filters and sequences in [14].

4 Conclusions

The Gaspra software is able to process large volumes of information, such as images from GONG observing sites. In all modules of the software, there is a focus on the processing performance. Therefore, the system tries to use full computing capacity to reduce the processing time. Some experiments show a speedup of 3.52-fold to process one single image in five different processing nodes, each node supporting 64 independent threads.

When we design Gaspra, we were concerned with its use in less computing performance infrastructures. We tried to ensure that even research groups with limited computer resources could realize their experiments. For this reason, Gaspra enables users generate surveys from the filters performance and object detections to use computing power in the most profitable way. After the execution of tasks, Gaspra does the surveys and enters performance data into a specific database. Afterwards, Gaspra executes new tasks in the shortest time possible, depending on the available infrastructure.

The main software interface consists of a program that guarantees their independence, agility and process automation. Jobs run automatically as specified in Python scripts. Thus, a researcher can input parameters such as a spatial position, make several runs and views, i. e., they can define a scientific workflow to process astronomical imagery and obtain new and relevant information about astronomy phenomena.

We designed the processing module to run in a simple computer with multicore processors or even in a more complex computing cluster with hybrid memory systems. In our future works, we plan to create a graphic interface to specify and select experiments, and we are considering incorporate manycore hardware to achieve even better performance.

Acknowledgment

The authors gratefully acknowledge the financial support from CAPES for providing the grants for research; FAPESP (projects 2010/50646-6 and 2011/00861-0) for providing computing and research facilities. In addition, the authors acknowledge the Big Bear Solar Observatory at New Jersey Institute of Technology and the InfraRed Science Archive (IRSA) at California Technology Institute for the images; and Laboratoire d'Informatique de Paris 6 for hosting the second author in his postdoctoral researches.

References

[1] LAWRENCE, A. .astronomy: Networked astronomy and the new media. In: ____. Bristol, England: Canopus Academic, 2009. cap. Drowning in Data: VO to the rescue, p. 197.

[2] 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.

[3] JONES, D. et al. Big data challenges for large radio arrays. In: *IEEE Aerospace Conference*. Big Sky, MT: [s.n.], 2012. p. 1–6. ISSN 1095-323X.

[4] MUELLER, D. et al. JHelioviewer: Visualizing large sets of solar images using JPEG 2000. *Computing in Science Engineering*, v. 11, n. 5, p. 38–47, Sept 2009. ISSN 1521-9615.

[5] GOODMAN, A. et al. WorldWide Telescope in Research and Education. In: BALLESTER, P.; EGRET, D.; LORENTE, N. P. F. (Ed.). *Astronomical Data Analysis Software and Systems XXI*. San Francisco: [s.n.], 2012. (Astronomical Society of the Pacific Conference Series, 1), p. 267.

[6] RABENSEIFNER, R.; HAGER, G.; JOST, G. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP Nodes. In: *17th Euromicro International Conference on Parallel, Distributed, and Network-based Processing*. Weimar: [s.n.], 2009. p. 427–436.

[7] POWELL, M.; ROSSI, R.; SHAMS, K. A Scalable Image Processing Framework for gigapixel Mars and other celestial body images. In: *IEEE Aerospace Conference*. Big Sky, MT: [s.n.], 2010. p. 1 –11. ISSN 1095-323X.

[8] JACOB, J. C. et al. Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking. *Int. J. Computational Science and Engineering.*, v. 2009, p. 73–87, maio 2010.

[9] BUDAVARI, T.; DOBOS, L.; SZALAY, A. S. Skyquery: Federating astronomy archives. *Computing in Science Engineering*, Institute of Electrical & Electronics Engineers (IEEE), v. 15, n. 3, p. 12–20, May 2013. ISSN 1521-9615.

[10] WELLS, D. C.; GREISEN, E. W.; HARTEN, R. H. FITS: a flexible image transport system. *Astronomy & Astrophysics supplement series*, v. 44, p. 363, 1981.

[11] BERRY, R.; BURNELL, J. *The Handbook of Astronomical Image Processing*. 2nd. ed. EUA: Willmann-Bell, Inc, 2005.

[12] DUFFETT-SMITH, P.; ZWART, J. *Practical astronomy with your calculator or spreadsheet*. 4th revised ed. ed. England: Cambridge University Press, 2011. ISBN 978-0-521-14654-8/pbk; 978-0-511-85561-0/ebook.

[13] TODY, D.; PLANTE, R.; HARRISON, P. *IVOA Recommendation: Simple Image Access Specification Version 1.0*. Washington, USA, 2011.

[14] 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. p. 41–48. ISBN 978-1-4503-1338-4.

[15] GAO, J.; WANG, H.; ZHOU, M. Development of an automatic filament disappearance detection system. *Solar Physics*, Kluwer Academic Publishers, v. 205, p. 93–103, 2002. ISSN 0038-0938.