Parallel and Distributed Astronomical Data Analysis on Grid Datafarm

Naotaka YAMAMOTO, Osamu TATEBE and Satoshi SEKIGUCHI
Grid Technology Research Center, AIST
Tsukuba central 2, Umezono 1-1-1, Tsukuba, Ibaraki, Japan
naotaka@ni.aist.go.jp

Abstract

A comprehensive study of the whole petabyte-scale archival data of astronomical observatories has a possibility of new science and new knowledge in the field, while it was not feasible so far due to lack of enough data analysis environment. The Grid Datafarm architecture is designed for global petabyte-scale data-intensive computing, which provides a Grid file system with file replica management for fault tolerance and load balancing, and parallel and distributed data computing support for a set of files, to meet with the requirements of the comprehensive study of the whole archival data.

In the paper, we discuss about worldwide parallel and distributed data analysis in the observational astronomical field. The archival data is stored, replicated and dispersed in a Gfarm file system. All the astronomical data analysis tools successfully access files in Gfarm file system without any code modification, using a syscall hooking library regardless of file replica locations. Performance evaluation of the parallel data analysis in several ways shows file-affinity process scheduling plays an essential role for scalable and efficient parallel file I/O performance. A data calibration tools shows scalable file I/O performance, and achieved the file I/O performance of 5.9 GB/sec and 4.0 GB/sec for reading and writing FITS files, respectively, using 30 cluster nodes (60 CPUs). On-demand file replica creation mitigates the overhead of access concentration. Another tool shows the performance improvement at a factor of six for reading a shared file by creating file replicas.

1. Introduction

There are two major approaches in astronomical research; observational research by telescopes, and theoretical calculation by computers. In the observational astronomical research, a size of observed data taken by large telescopes and detectors has been dramatically increasing. The SUBARU telescope[6] on the summit of the Mauna Kea built by National Astronomical Observatory of Japan, has a primary mirror with the world largest class diameter of 8 meter. About 2 TBytes of data taken by its prime focus camera (Suprime-Cam[7]) from January, 1999 of the first light till June, 2002, has been published by a data archive system, SMOKA Science Archive[10]. Besides the SMOKA, there are several large archive systems such as HST Archive[12] and CFHT Archive[3]. The total amount of published archival data approaches petabyte scale. On the other hand, most researchers analyze the data of only interested targets, acquired by a telescope or an archive site. This is mainly due to lack of enough data analysis environment with respect to computational power and storage capacity.

A comprehensive study of the whole archival data has a possibility of new science and new knowledge. By re-analyzing the whole data, it is possible to build a new astronomical catalogue. A new astronomical catalogue may contain a new classification of galaxies and/or stars, which gives the information of birth of galaxies. New catalogues may also include hints of mysterious objects such as Gamma ray burster[16], QSOs[8], etc. The archival data may include forsaken objects such as solar system objects that are out of interest of the observers. Detecting new solar system objects gives the information of the structure of our solar system.

One of difficulties of the comprehensive study is that the total amount of data is increasing. For example, the amount of archival data taken by the SUBARU telescope is increasing at the rate of about 20 TBytes per year. Moreover, about 10 times more storage capacity is required the the original data size to analyze data taken by the Suprime-Cam. It is a critical demand in this study to analyze hundreds of terabytes or petabytes of data efficiently.

There are several projects for virtual astronomical observatory (VO) such as JVO[9], NVO, AVO and IVOA. Main purpose of each VO project is to integrate and federate archive systems dispersed in a Grid by standardizing XML schema, data access layer, and query language of astronomical archival data. Parallel and distributed analysis for the whole petabyte-scale archive data is out of scope of the VO activity at this time.

In order to analyze the whole archival data, a high-performance file system having petabyte-scale capacity is needed. Distributed file systems such as NFS and AFS target situations where many distributed clients efficiently access small files by using file caches, etc. It cannot achieve sufficient bandwidth for write operations requiring a GB/s bandwidth. Cluster and SAN file systems such as PVFS[4], Lustre[5] and CXFS may improve the file I/O bandwidth, although it is limited by the network bandwidth between storages and client systems.

The Grid Datafarm architecture[14] is designed for global petabyte scale data-intensive computing, which provides a Grid file system with file replica management (Gfarm file system), and parallel and distributed data processing support for a set of files (Gfarm file). Gfarm version 1.0, a reference implementation of the Grid Datafarm architecture, was released on November 25, 2003 from the Web site[1] (Version 1.0.3 was released on May 25, 2004). It provides scalable I/O bandwidth, and scalable parallel processing to exploit local I/O in a grid of clusters. The data is, physically, replicated and dispersed among cluster nodes across administrative domains, whereas it can be accessed transparently in file replica locations via POSIX file I/O interface by data analysis tools.

In this paper, we will discuss about astronomical data analysis environment using the Grid Datafarm architecture, and evaluate the performance of parallel and distributed data analysis.

2. Astronomical Data Analysis

2.1. Astronomical Data

Visible and infrared astronomical image data is often stored in the FITS[11] format. FITS file consists of a header part and a body part. The header part includes instrument information, time of observation, position in the sky, weather condition etc. The body part includes n-dimensional data sheet. The Suprime-Cam is a mosaic CCD camera consisting of 10 CCD detectors with 2000×4000 pixels and 16-bits depth, which has a capability to take wide field of view of about 33×27 arcmin. Since each detector generates a FITS file, the Suprime-Cam generates 10 FITS files of 170 Mbytes in one shot.

2.2. Data Analysis Flow

Generally, the observed raw data produced by equipments require various calibrations for astronomical research. For the SUBARU telescope, there are several data calibration tools shown in Table 1, Figure 1 depicts a flow of data calibration for N shots of images. The number in parenthesis on the arrow corresponds to the item number of data calibration tools in Table 1. Parallelism in each operation is shown below the arrow.

(1) osmed	subtract bias of an A/D converter
(2) medianimg	calculate median from N images
(3) uppercut	remove bad pixels
(4) arithimg	divide by flat field
(5) distcorr	correct optical distortion
(6) skysb	subtract sky background

Table 1. List of operations in data calibration tools

2.2.1. Data Calibration A data calibration consists of a flat image generation (hereafter Pa) and a data reduction (hereafter Pb) as shown in Figure 1. The Pa generates flat image files from N sets of images that will be used for rectifying sensitivity errors of CCD pixels in the data reduction phase. The operation (1) subtracts bias of an A/D converter. It analyzes $10 \times N$ images, and generates $10 \times N$ images independently. The operation (2) calculates a median image from N shots. Since there are 10 CCD chips in the Suprime-Cam, 10 median images can be generated in parallel, assuming each median computation is performed serially. The operation (3) removes bad pixels from a median image, and generates a flat image file in parallel.

The procedure Pb calibrates observed raw data using the flat image generated in the Pa. In the operation (4), every image file is divided by a flat image. Since there are 10 flat image files that correspond to 10 CCD chips, it is divided by the corresponding flat image file. The operation (5) corrects optical distortion, and the operation (6) subtracts sky background. Each image can be processed in parallel.

In the calibration phase of N shots, each operation generates new $10 \times N$ FITS images. Since the operation (1) generates image data with 32 bits depth from that with 16 bits depth, the calibration phase requires storage capacity at least 10 times larger than the original data size. Furthermore, the succeeding scientific data analysis requires more.

2.2.2. Extraction objects and FITS file viewer SExtractor[2] (Source Extractor) is a tool to build a catalogue of objects such as stars, galaxies, etc. from an astronomical image. This tool shows better performance than other source extraction tools on moderately crowded star fields. A generated catalogue includes luminosity, classification parameters, positions of objects, etc. It is widely used for large-scale galaxy survey, new object survey, etc.

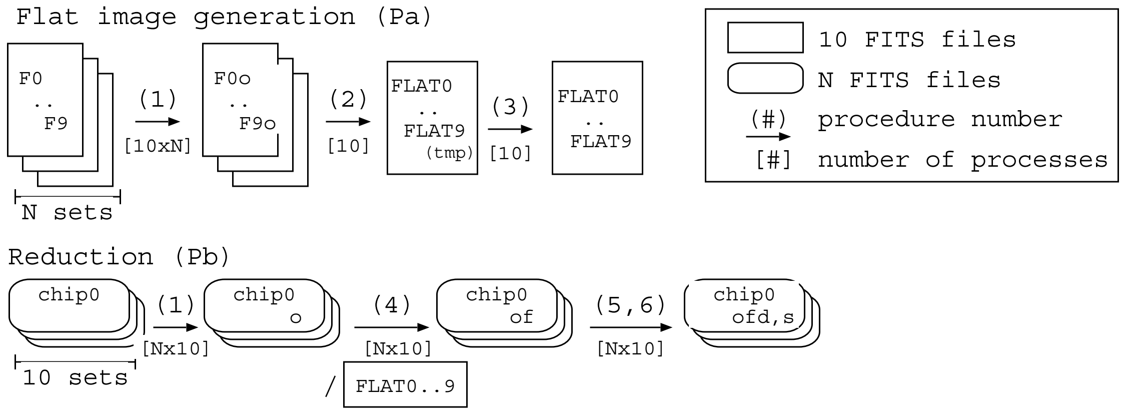

Figure 1. Typical flow of data calibration for N sets of images

In astronomical data analysis, in order to verify the result generated by the data analysis tools, a FITS file browser is frequently used. The SAOImage ds9 13) written in C++ and Tcl/Tk is one of graphical astronomical image and data visualization applications.

3. Distributed and parallel analysis on Grid Datafarm

3.1. Gfarm

Gfarm is a reference implementation of the Grid Datafarm that is designed for facilitating reliable file sharing and high-performance distributed and parallel data computing in a Grid across administrative domains by providing a Grid file system. A Grid file system is a virtual file system that federates multiple file systems.

The most time-consuming but also the most typical task in data computing such as astronomy, high energy physics, space exploration, human genome analysis, is to process a set of files in the same way. Such a process can be typically performed independently on every file in parallel, or at least have good locality. Gfarm supports high-performance distributed and parallel computing for such a process by introducing a *Gfarm file*, a new *file-affinity* process scheduling based on file locations, and new parallel file access semantics. An arbitrary group of files possibly dispersed across administrative domains can be managed as a single Gfarm file. Each member file will be accessed in parallel in a new file view called *local file view* by a parallel process possibly allocated by file-affinity scheduling based on replica locations of the member files. File-affinity scheduling and new file view enable the “owner computes” strategy, or “move the computation to data” approach for parallel and distributed data computing of member files of a Gfarm file in a single system image.

Gfarm file system also supports file replica management. Every file can be replicated dynamically and can be stored in any file system node. When a file is accessed, one of the file replicas is selected by CPU load average and response time. This not only enhances a capability of fault tolerance but also avoids access concentration and load imbalance.

Gfarm files in a Gfarm file system can be accessed by Gfarm parallel I/O APIs (hereafter referred to Gfarm APIs). On the other hand, this requires to modify application source code, and to catch up with the update of application, as well as the modification tends to introduce a new bug.

Instead, Gfarm provides a syscall hook library to utilize Gfarm file system without any code modification. It traps system calls for file I/O to investigate whether the specified operation is for a Gfarm file system or not. If it is for a Gfarm file system, it calls appropriate Gfarm APIs. For example, pseudocode of the syscall hook library for `open` system call is as follows.

```
_open(file, ...) {
    if (file is a Gfarm file)
        gfs_pio_open(file, ...)
        set appropriate file view
    else
        syscall(SYS_open, file, ...)
    endif
}
```

When a specified file is a Gfarm file, it will call `gfs_pio_open` and set appropriate file view. Otherwise, just call `open` system call as usual. Gfarm syscall hook library decides file view in the following policy; For newly created files, the default file view is a local file view. For existent files, if the number of processes and the number of file fragments are the same, the default file view is a local file view, otherwise, the default view is a global view.

3.2. Observational Data

For parallel computing by Gfarm, it is necessary to create a Gfarm file that consists of several FITS files that can be independently processed in parallel. In the procedure Pa, there is 10 degree of parallelism in the operations (2) and (3). We create N sets of Gfarm files that consists of 10 FITS images in one shot.

On the other hand, in the procedure Pb, all operations seem to have $N \times 10$ degree of parallelism, however, the operation (4) consists of 10 different operations. This means there is N degree of parallelism with respect to the same operation. We create 10 sets of Gfarm files that consists of N FITS images in the same CCD detector.

For distributed computing, each FITS file in a Gfarm file needs to be stored in a different file system node. Gfarm has a command to store a Gfarm file such that each member file is stored in a different file system node. To improve accessibility and availability of data, it is necessary to create file replicas. Gfarm also provides a command to create a file replica in any file system node.

3.3. Parallel and Distributed Data Analysis

Every analysis tool needs to access Gfarm files in a Gfarm file system. Instead of modifying application code, we use a syscall hook library to access Gfarm files. In this case, it is enough for each application to link with a syscall hook library and a Gfarm library without any source code modification.

After that, for example, the operation (1) in the procedure Pb can be executed in parallel and distributedly on N different file system nodes, assuming N FITS files in a Gfarm file are physically stored on different file system nodes. In this case, each process reads an input file from a local file system, and generates an output file to a local file system.

Here is a concrete example of SExtractor. SExtractor is executed by specifying an input FITS file, a parameter file and an output catalogue file.

```
% sex input.fits -c detect.param
                          -o out.cat
```

In a Gfarm system, remote parallel and distributed execution of SExtractor can by done by the `gfrun` command.

```
% gfrun -r gfarm:sex gfarm:input.fits
  -c gfarm:detect.param -o gfarm:out.cat
```

In this case, the execution command `gfarm:sex` is executed in parallel using the file-affinity process scheduling of `gfarm:input.fits`. The same number of processes as the number of member files of `gfarm:input.fits` are executed on file system nodes where the corresponding member files are stored. When there are redundant file replicas of member files, the least busy file system node is selected. Parameter file `gfarm:detect.param` is a Gfarm file that consists of a single member file. This file will be accessed in a global file view from every parallel process. The `-r` option of `gfrun` specifies the on-demand file replication mode. Because `gfarm:detect.param` will be accessed from every parallel process, it will be replicated on demand.

SAOImage ds9 FITS file browser written in C++ and Tcl/Tk can access a Gfarm file using a syscall hook library. Using a file selection window in SAOImage ds9, it is possible to browse and specify a Gfarm file.

4. Performance Evaluation of Astronomical Data Analysis Tools

This section evaluates the performance of parallel astronomical data analysis on Grid Datafarm. 30 nodes of the AIST Gfarm Cluster I[15] are used as file system nodes, which are a part of the Trans-Pacific Grid Datafarm testbed. Each cluster node has 2.8GHz dual Xeon processors, 1GB memory, Gigabit Ethernet interface, and a RAID controller with four 200GB HDDs configured in RAID-0. For parallel data analysis, we generate a Gfarm file that consists of different number of FITS files, and analyze member files of a Gfarm file in parallel. To remove every non-uniform factor in data analysis, we use the same FITS file for all member files. We measure the total time spent by Gfarm parallel I/O APIs listed as follows in each parallel process.

```
gfs_pio_create,  gfs_pio_open,
gfs_pio_read,    gfs_pio_write,
gfs_pio_seek,    gfs_pio_close,
gfs_pio_set_view_index,
gfs_stat,  gfs_pio_set_view_section
```

4.1. Gfarm Parallel I/O performance

This subsection evaluates the parallel I/O performance using the operation (1), `osmed`, in Table 1. It subtracts A/D converter bias from an observed FITS file with 16 bits depth, and generates a FITS file with 32 bits depth. The `osmed` reads an input FITS file with the size of about 17 Mbytes, and outputs a FITS file with the size of about 34 Mbytes.

For parallel data analysis performance evaluation, we use a Gfarm file that consists of different number of FITS image files for an input data. In case of 20 files, a Gfarm file becomes the size of 340 Mbytes. Parallel processes are allocated based on replica locations of each member file via the file-affinity scheduling. Each parallel process reads the corresponding member file that is expected to be stored in a local file system, and generate an output file also in the local file system.

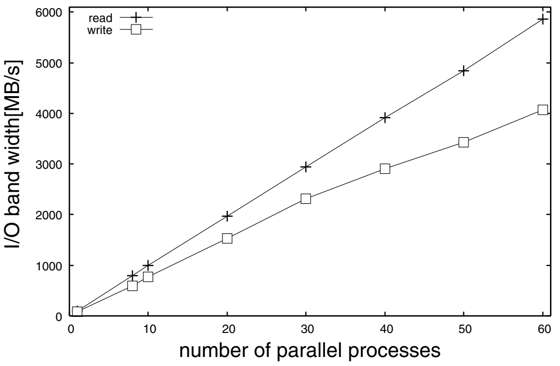

Figure 2. Gfarm parallel I/O performance of "osmed"

Figure 2 shows the parallel I/O performance of `osmed` with various number of parallel processes up to 60. We achieve the parallel I/O performance of 5.9 GB/sec and 4.0 GB/sec using 60 CPUs. When the number of processes is more than 30, the write performance was slightly degraded. That is because there is a node on which two `osmed` processes are running. On the other hand, the read performance is not affected by this.

4.2. Impact of file replicas

This section investigates the impact of file replicas using the operation (4), `arithimg`, in which every parallel process refers to the same flat image. We compare two cases; one case such that there is only one file replica for the flat image, and the other case such that every node has a file replica. In the first case, all parallel processes need to access a single file replica, which results in access concentration. To mitigate the access concentration, the flat image needs to be replicated. Gfarm has a capability of on-demand file replica creation that creates a file replica to a local file system dynamically before accessing remotely.

Figure 3 shows the breakdown of the elapsed time of `arithimg`, comparing two cases of 10 parallel processes. The left-side graph shows the first case such that every parallel process accesses a single file copy, and the right graph shows the second case such that every parallel process accesses its own local file copy of the flat image. Each graph shows the breakdown of the total elapsed time on each node into `gfs_pio_read`, `gfs_pio_set_view_section` and the others. `gfs_pio_set_view_section` changes a file view to a specified file fragment, and opens it. In the first case, every process opens the same remote file copy, and in the second case, it opens a local file copy. `gfs_pio_read` reads the contents of a file in a file view. There is no major difference in other APIs between two cases because the output file will be created in a local file system in both cases.

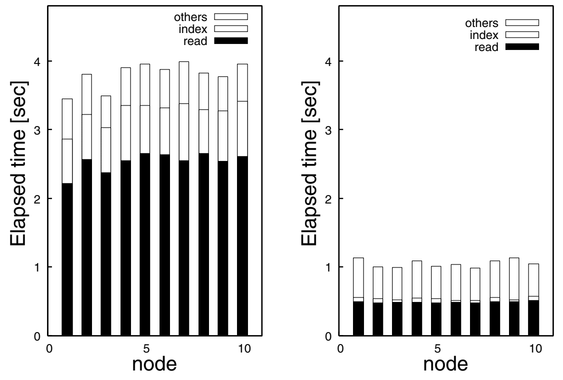

Figure 3. A comparison of `arithimg` in two cases. One case has only one file replica of a shared file, and the other case has a file replica on each node.

The second case improves the read time from 3.2 seconds to 0.5 seconds. With 10 processes, it improves the read performance six times from 200 MB/sec to 1250 MB/sec. Total elapsed time is improved from 3.8 seconds to 1.1 seconds. Total performance was improved 3.6 times by creating file replicas on each node.

5. Astronomical Results and Discussion

5.1. Astronomical Results

We searched minor bodies in our solar system using the data taken from November, 2002 to December, 2002. Solar system objects following the Kepler motion around the sun are detected by comparing several images taken succeedingly. Among several approaches to detect moving objects, we selected the simplest method. The method first picks all objects up in every image by SExtractor, then extracts moving objects from the list of selected objects. Note that all the input and output images, and data analysis tools are stored on a Gfarm file system provided by Trans-Pacific Grid Datafarm testbed.

5.2. Discussion

Worldwide astronomical data analysis environment was constructed using a Gfarm reference implementation of the Grid Datafarm architecture. Thanks to a system call trap functionality in a Gfarm library, no modification is required to execute every tool to search for new solar system objects. It was shown that parallel and distributed data analysis feature of the Grid Datafarm worked quite well in astronomi-

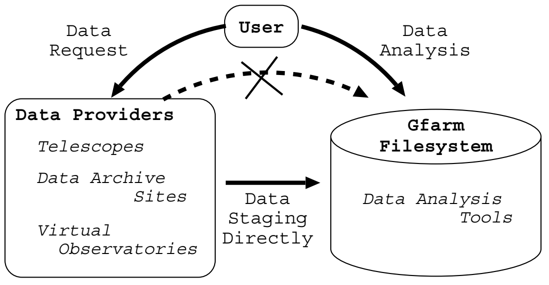

Figure 4. Future plan of Virtual Astronomical Data Analysis Center

cal data analysis. The same as before but parallel and distributed data analysis environment was constructed on the Grid Datafarm by introducing not only data analysis tools but also a FITS file browser, SAOImage ds9.

On the other hand, it is needed to store large-scale data to a Gfarm file system before starting parallel and distributed data analysis. Currently, it is done by hand from the SMOKA archive, which may become the bottleneck of the whole data analysis. In order to solve the problem, data requested by users need to be directly stored to a Gfarm file system from data providers such as telescopes, data archivers, and a virtual astronomical observatory, as shown in Figure 4. We have a future plan to collaborate with the Japanese VO 9) to investigate a possibility of a virtual astronomical data analysis center in Figure 4.

6. Conclusion

We discussed about worldwide astronomical data analysis using the Grid Datafarm architecture, and showed how it helped parallel and distributed astronomical data analysis for a comprehensive study. All the required analysis tools and a FITS file browser could successfully utilize a Gfarm file system. One of astronomical data calibration tools showed scalable file I/O performance up to 60 parallel executions; 5.9 GB/sec and 4.0 GB/sec for reading and writing FITS files, respectively, using 30 cluster nodes (60 CPUs). On-demand replication feature improved the performance at a factor of six for accessing a shared data such as a flat field image. A large-scale data analysis environment targeting the whole data was ready, which would help to obtain new astronomical results in the near future. In a future research plan, we would like to collaborate with a VO to investigate a possibility to provide a virtual astronomical data analysis center.

Acknowledgments

We are thankful to members of the Grid Datafarm project who give us precious comments in carrying out this research. We also thank members of Grid Technology Research Center, AIST, for their cooperation in this work. Especially, we thank Mr. Toshiyuki Shimizu, who takes care of the AIST Gfarm cluster I, and also thank all collaborators of the Trans-Pacific Grid Datafarm testbed and PRAGMA. We thank people who provide the data analysis tools, especially Dr. M. Yagi. All of data in the paper is either observed by Subaru Telescope or obtained from a data archive at the Astronomical Data Analysis Center operated by the National Astronomical Observatory of Japan.

References

[1] AIST. *Grid Datafarm*. http://datafarm.apgrid.org/.

[2] E. Bertin and S. Arnouts. Sextractor: Software for source extraction. *Astronomy and Astrophysics Supplement*, 117:393–404, 1996.

[3] The Canadian Astronomy Data Centre (CADC). *The CFHT Archive*. http://cadcwww.dao.nrc.ca/cfht/.

[4] P. H. Carns, W. B. L. III, R. B. Ross, and R. Thakur. A parallel file system for linux clusters. *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, 2000. http://www.parl.clemson.edu/pvfs/.

[5] Cluster File Systems, Inc. *Lustre*. http://www.lustre.org/.

[6] N. Kaifu. et al. The first light of the subaru telescope: A new infrared image of the orion nebula. *PASJ*, 52:1, 2000.

[7] Y. Komiyama et al. High-resolution images of the ring nebula taken with the subaru telescope. *PASJ*, 52:93, 2000.

[8] I. Marquez, J. Masegosa, A. Olmo, and et al. *Qso Hosts and Their Environments*. Kluwer Academic Publishers, 2001.

[9] NAOJ. *Japanese Virtural Observatory*. http://jvo.nao.ac.jp/.

[10] NAOJ. *Subaru Mitaka Okayama Kiso Archive system*. http://smoka.nao.ac.jp/.

[11] NASA. *FITS*. http://fits.gsfc.nasa.gov/.

[12] NASA. *HST Archive*. http://hst.nao.ac.jp/.

[13] SAO. *Image DS9*. http://hea-www.harvard.edu/RD/ds9/.

[14] O. Tatebe, Y. Morita, S. Matsuoka, N. Soda, and S. Sekiguchi. Grid datafarm architecture for petascale data intensive computing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, pages 102–110, 2002.

[15] O. Tatebe, H. Ogawa, Y. Kodama, T. Kudoh, S. Sekiguchi, S. Matsuoka, K. Aida, T. Boku, M. Sato, Y. Morita, Y. Kitatsuji, J. Williams, and J. Hickes. The second trans-pacific grid datafarm testbed and experiments for sc2003. In *Proceedings of 2004 International Symposium on Applications and the Internet Workshops (SAINT 2004 Workshops)*, pages 26–30, 2004.

[16] K. Weiler and K. W. Weiler. *Supernovae and Gamma-Ray Bursters*. Springer Verlag, 2003.