
Lessons learned through driving science applications in the PRAGMA grid

Cindy Zheng*, Mason J. Katz
and Phil M. Papadopoulos

San Diego Supercomputer Center, USA

E-mail: zhengc@sdsc.edu

*Corresponding author

David Abramson, Shahaan Ayyub,
Colin Enticott, Slavisa Garic
and Wojtek Goscinski

Monash University, Australia

Peter Arzberger

University of California, San Diego, USA

Bu Sung Lee

Nanyang Technological University, Singapore

Sugree Phatanapherom,
Somsak Sriprayoonsakul and
Putchong Uthayopas

Thai National Grid Center

Software Industry Promotion Agency, Thailand

Yoshio Tanaka and Yusuke Tanimura

National Institute of Advanced Industrial
Science and Technology, Japan

Osamu Tatebe

University of Tsukuba, Japan

Abstract: This paper describes the coordination, design and implementation of the PRAGMA Grid. Applications in genomics, quantum mechanics, climate simulation, organic chemistry and molecular simulation have driven the middleware requirements, and the PRAGMA Grid provides a mechanism for

science and technology teams to collaborate, for grids to interoperate and for international users to share software beyond the essential, *de facto* standard Globus core. Several middleware tools developed by researchers within PRAGMA have been deployed in the PRAGMA grid and this has enabled significant insights, improvements and new collaborations to flourish. In this paper, we describe how human factors, resource availability and performance issues have affected the middleware, applications and the grid design. We also describe how middleware components in grid monitoring, grid accounting and grid file systems have dealt with some of the major characteristics of our grid. We also briefly describe a number of mechanisms that we have employed to make software easily available to PRAGMA and global grid communities.

Keywords: global grid; grid middleware; grid application; grid interoperation.

Reference to this paper should be made as follows: Zheng, C., Katz, M.J., Papadopoulos, P.M., Abramson, D., Ayyub, S., Enticott, C., Garic, S., Goscinski, W., Arzberger, P., Lee, B.S., Phatanapherom, S., Sriprayoosakul, S., Uthayopas, P., Tanaka, Y., Tanimura, Y. and Tatebe, O. (2007) 'Lessons learned through driving science applications in the PRAGMA grid', *Int. J. Web and Grid Services*, Vol. 3, No. 3, pp.287–312.

Biographical notes: Cindy Zheng is the Coordinator of the PRAGMA grid. David Abramson is an Australian Research Council Professorial Fellow, and Professor of Computer Science at the Monash University. Peter Arzberger is the Chair of the PRAGMA Steering Committee, the PI of the NSF funding to support the US participation in PRAGMA and the Director of the NIH-funded National Biomedical Computation Resource. Shahaan Ayyub is a PhD student at the Monash University, Australia. Colin Enticott, Slavisa Garic and Wojtek Goscinski are Research Scientists in the Monash eScience and Grid Engineering (MESSAGE) Laboratory at the Monash University. Mason Katz is the Group Lead for Cluster Development at the San Diego Supercomputer Center and is Co-investigator of the NSF Grant that supports UCSD participation in PRAGMA. Associate Professor Bu-Sung Lee is the founding President of Singapore Advance Research and Education Network (SingAREN), Associate Chair (Research) in the School of Computer Engineering, Nanyang Technological University and a member of the National Grid Advisory Council (Singapore). Dr. Philip Papadopoulos is the Program Director for Grid and Cluster Computing at the San Diego Supercomputer Center and is Co-investigator of the NSF Grant that supports the UCSD participation in PRAGMA. Sugree Phatanapherom is a Researcher in the Thai National Grid Center, Software Industry Promotion Agency. Somsak Sriprayoosakul is a Researcher and Group Leader in the Thai National Grid Center, Software Industry Promotion Agency. Yoshio Tanaka is a Principal Research Scientist of the Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology and the Chair of the International Grid Trust Federation and the Asia Pacific Grid Policy Management Authority. Yusuke Tanimura is a Researcher of the Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology. Dr. Osamu Tatebe is an Associate Professor in the Department of Computer Science at the University of Tsukuba. Puchong Uthayopas is the Director of the Thai National Grid Center, Software Industry Promotion Agency.

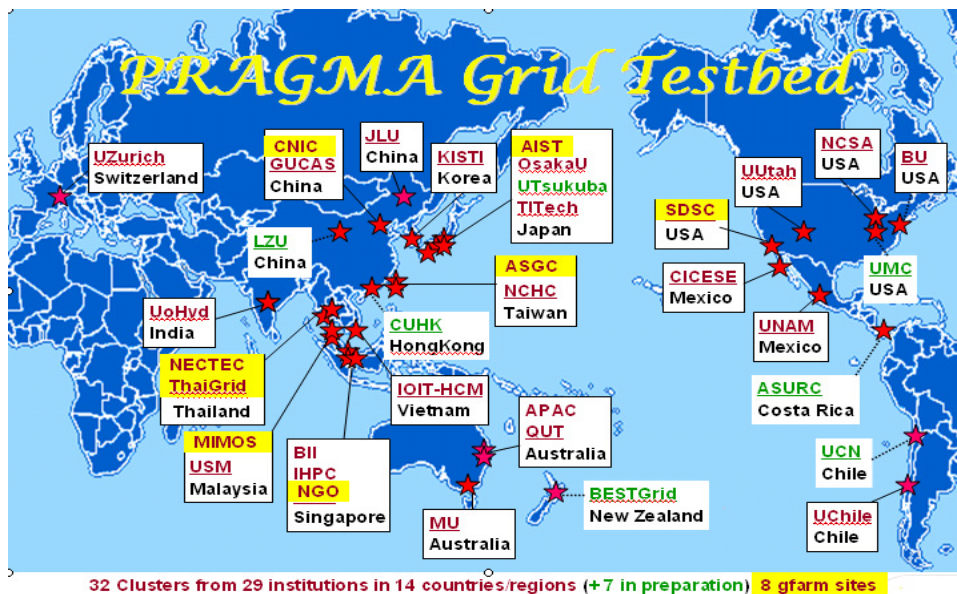
1 Introduction

The Pacific Rim Applications and Grid Middleware Assembly (PRAGMA), founded in 2002, is an open international organisation that focuses on a variety of practical issues of building international scientific collaborations in a number of application areas. PRAGMA operates four working groups – Resources and Data Computing, Telescience, Bioscience, and GEO – supporting global environmental observing and geosciences. The resources and data computing working group’s mission is to improve the interoperability of grid middleware and to enhance the usability and productivity of a global grid.

In May 2004 we established the PRAGMA Grid as a global grid and ran multiple applications. This experiment allowed us to learn about issues involved in building an easy-to-use production global grid and in running applications on a routine basis by users who do not necessarily know the inner workings of the middleware or the grid. The PRAGMA Grid itself is a ‘grass roots’ effort to build a long-term working experiment that allows a variety of applications and application scientists to work across international boundaries.

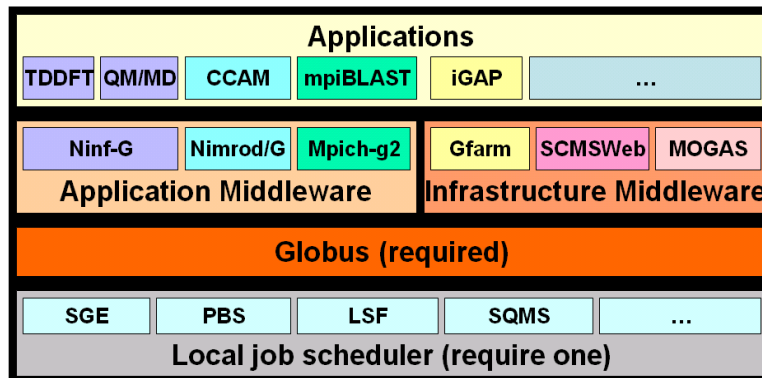
Today, 29 institutions from five continents and 14 countries have contributed both physical resources and, critical to success, people resources to the PRAGMA Grid (Figure 1). It is an instantiation of a useful, interoperable, and consistently available grid system that is neither dictated by the needs of a single science domain, nor funded by a single national agency. It differs significantly from experiments such as PlanetLab (Paterson and Roscoe, 2004) and production grids such as TeraGrid (Pennington, 2002) in that it does not have uniform infrastructure management. We started with minimum grid system requirements: a local job scheduler and Globus (Foster and Kesselman, 1997). Upper middleware layers and components are gradually built-in, driven by applications requirements and grid management needs.

Figure 1 PRAGMA grid geographical distribution (April 2007)



Even though this grid is not centrally managed, it is surprisingly robust and supports a wide range of scientific applications. Over three years we ran 12 applications – Time Dependent Density Functional Theory (TDDFT) (Yabana and Bertsch, 1999), Quantum Mechanics/Molecular Dynamics (QM/MD),¹ Fragment Molecular Orbital (FMO),² the CSIRO Conformal-Cubic Atmospheric Model (CCAM) (McGregor *et al.*, 1993), a genetic sequence alignment tool (mpiBLAST) (Darling *et al.*, 2003), mesoscale weather models (MM5/WRF),³ the integrated Genome Analysis Pipeline (iGAP) (Li *et al.*, 2005) and more. Each application creates its own sub-grid and middleware stack: TDDFT, QM/MD and FMO are based on Ninf-G (Tanaka *et al.*, 2004), CCAM on Nimrod (Abramson *et al.*, 2002), mpiBLAST on Mpich-g2 (Foster and Karonis, 1998), MM5/WRF on Mpich-gx⁴ and iGAP on Gfarm (Tatebe *et al.*, 2004) (Figure 2). Ninf-G, Nimrod, Mpich-g2 and Mpich-gx are application middleware which enable applications to run on a grid.

Figure 2 PRAGMA grid software layers



Simultaneously, we built the grid infrastructure by establishing a grid operation centre, deploying a grid monitoring system (SCMSWeb) (Chalakornkosol and Uthayopas, 2003), testing a grid accounting system (MOGAS) (Lee *et al.*, 2006), implementing a meta-scheduling service under Community Scheduling Framework (CSF)⁵ and building a grid file system (Gfarm). SCMSWeb, MOGAS, CSF and Gfarm are infrastructure middleware which provide and enhance the grid infrastructure services, thus benefit applications and grid management.

Throughout our experiments we encountered many problems, including system/network failures, trust management issues, complications in user and application environment setup, job submission difficulties, resources sharing, system/job monitoring and accounting problems. These problems were mainly caused by four factors: distance and time zone differences among sites, lack of infrastructure tools for heterogeneous global grid, non-uniform system and network environments, and diverse application requirements. Addressing these issues and problems has focused our efforts, thereby motivated collaborative innovations and accelerated our software enhancements.

The PRAGMA grid allows collaborations of application and middleware teams to be forged and to flourish. In addition, we also collaborate with many science and technology teams, working in the areas of grid security, grid infrastructure, datagrid, sensor and high performance networks, to leverage resources and community networks. Collectively

these collaborations quicken our advancements in science and technology and broaden our impact to grid communities at large. We also collaborate with other grids, thereby expanding our routine-use experiments to other grids, exploring grid interoperability issues, providing requirements to standardisation efforts, testing ideas and solutions, and paving a way for scientific applications to run across grids.

In this paper we discuss the setup and operation of the PRAGMA Grid (§2), our experiences with application middleware (§3) and infrastructure middleware (§4). We conclude with our key practical experience gained by running a grass-roots grid. It is our hope that our pragmatic approach and its lessons learned will benefit other emerging grid efforts or efforts to promote grid interoperation. And ultimately, we wish to make this and other grid efforts benefit science, engineering and education.

2 Operations and collaborations in the PRAGMA grid

Mobilising human resources was the first and most significant issue in building the PRAGMA Grid, which created some unique challenges because of the geographic distribution of our grid sites. Furthermore, each site had its own funding sources and chain of command, thus it was not always easy to allocate the human resources or to achieve the required priority to this project. Out of necessity, we based our relationship on mutual benefits and interests, and built consensus through active leadership. We formed a PRAGMA Grid working team by enlisting two or more contacts per site: at a minimum, a technical contact and a supervisor. One team member acted as the PRAGMA grid coordinator who maintained close communication between the team members and the PRAGMA leadership. The coordinator was responsible for soliciting and encouraging input from the team, organising discussions, summarising team decisions, facilitating collaborations, tracking task status and promoting speedy progress in implementing the team decisions.

The main communication tool used by the PRAGMA Grid support team was e-mail. We established mailing lists for both supporters and users. In addition, video teleconference (VTC) and audio teleconference via Skype⁶ were also used for group meetings. Whilst effective, these real-time communication methods were difficult to organise and chair, since they typically involved ten or more people from five or more groups distributed across multiple time zones (between 10 and 14 hour separations). Moreover, whilst we all speak English, it is not the native tongue for the majority sites. Over and above these electronic meetings, PRAGMA semi-annual workshops proved to be precious opportunities and allowed most sites get together for face-to-face meetings.

A key instrument for organising our activities was the Grid Operation Centre (GOC). Via the GOC we compiled and published a PRAGMA Grid member list, resource tables, application requirements, site support status and user guides. We installed the SCMSWeb meta-server (§4.2) to provide real-time PRAGMA Grid status on both systems and jobs. Further, Asian Pacific Area Network provided real-time network status and measurements on GOC. Finally, the GOC was used to archive and publish research data, reports, conference presentations and software packages.

A very good collaboration tool is the PRAGMA Grid WIKI site.⁷ All application drivers, site supporters, and collaborators involved with PRAGMA, jointly working together, document their plans, organisations, experiment status, results, various tools,

and instructions at this WIKI site. It has become an essential tool and a centre of activity where everyone can contribute, share information, and manage our sites, experiments and collaborations.

Compared to most well known grids, the PRAGMA Grid has the distinct characteristic of heterogeneity since it is a grass roots grid. Each site had its own policies, which precluded standardising system environments, policies, and procedures among PRAGMA grid sites. Each site specified its own software versions, security policies, and maintenance schedules, and we had to accept these differences and to try to adapt better technologies and solutions to manage and take advantage of this diversity. This is in stark contrast to the US TeraGrid in which a standard software stack is used for all grid sites.

To conduct our experiments, we developed the following (still evolving) procedures:

- Application or middleware drivers publish application or middleware requirements and deployment instructions at the PRAGMA Grid WIKI site; also file PRAGMA user account requests.
- The coordinator packages the user information and makes them available at secure GOC website, then sets up the user accounts and implements the documented requirements at one site, and work with the application or middleware driver to verify the resolve any problems.
- Then, the coordinator e-mails the PRAGMA grid supporters list, calls for general deployment of the requirements, including the required user access setup among all PRAGMA grid sites.
- When a site has implemented the requirements, the site technical contact informs both the application or middleware driver and the coordinator using e-mail.
- The application or middleware driver then tests access and requirements at the site and resolves problems directly with the site technical contact.
- When the application is working correctly at the site, the application driver includes the site in his/her routine long runs.
- All experiments and site status are documented and updated through out the process at our WIKI site by all site administrators.

In addition to rich and diverse physical resources, PRAGMA has more importantly assembled very talented expertise covering wide range of technologies. Individually, these development teams have created, for example, cluster management software Rocks (Papadopoulos *et al.*, 2001) (SDSC, USA); remote procedure call Ninf-G (AIST, Japan), parameter sweep codes Nimrod (Monash University, Australia); grid enabled MPI code Mpich-Gx (KISTI, Korea); grid monitoring software SCMSWeb (ThaiGrid, Thailand), distributed file system Gfarm (AIST and University of Tsukuba, Japan); grid accounting software MOGAS (Nanyang Technological University of Singapore); metascheduling code Community Scheduling Framework (CSF) (Jilin University of China); and many others.

The PRAGMA grid is built on collaborations. Besides collaborations among our routine operations, application and middleware experiments, we also collaborate with science and technology teams outside of PRAGMA and work in many focused areas.

In the area of grid security, we worked closely with Asia Pacific Grid (APGrid), developed near term and long term security infrastructure plans that to suit the characteristic of the PRAGMA Grid while still promoting interoperability with global grid communities. To form a grid while still allowing each site to maintain certain independence, we started with the most common Grid Security Infrastructure (GSI) authentication and authorisation as the base. Each institution runs its own Certificate Authority (CA) and issues certificates for users, hosts, and services which belong to the institution. At the beginning of building the PRAGMA grid, all institutions agreed to trust each other's CAs. Although most CAs were experimental, this decision was accepted within the PRAGMA institutions to accelerate the development of the PRAGMA grid. We also collaborated with National Research Grid Initiative (NAREGI)⁸ in Japan, tested and enhanced NAREGI-CA software; setup a catch-all PRAGMA experimental CA to serve the sites and users who do not have their own CA. Some of the sites later have setup their own CAs using NAREGI-CA. After a year, many PRAGMA grid sites' CAs have been upgraded to production-level and are accredited by or working to obtain the accreditation from the International Grid Trust Federation (IGTF) through the APGrid Policy Management Authority (PMA). One key motivation for IGTF accreditation comes from our Grid interoperability effort with other grids. As of March 2007, six sites in PRAGMA grid have IGTF-accredited CAs which enabled our multi-Grid interoperability experiments (see for example GIN below). In addition, the PRAGMA production CA has been setup. Working with APGrid PMA, we expect the PRAGMA production CA soon to be an IGTF-accredited catch-all CA for PRAGMA members.

To build robust grid infrastructure and services in PRAGMA grid, many technology teams work together. For example, the SCMSWeb grid monitoring team worked with MOGAS grid accounting team and Gfarm team to integrate their software; the Computational Science and Engineering Online (CSE-Online)⁹ cyber-environment team worked with Australian Partnership for Advanced Computing (APAC) and SCMSWeb team to develop a computational science and engineering application service; the Rocks team worked on integrations with Ninf-g, SCE (Uthayopas *et al.*, 2001), Gfarm and Open Science Grid (OSG)¹⁰ Virtual Data Toolkit (VDT) teams as well as many other middleware and scientific software teams, and created an easy-to-use cluster management system with a rich set of selectable grid middleware.

To build a robust Datagrid, we have been working with AIST and University of Tsukuba of Japan, deploying and testing Gfarm – a high performance scalable grid file system in PRAGMA grid. We are also collaborating with the GEON¹¹ data network, working to expand GEON US-based data network to a global geosciences data network, to interoperate with PRAGMA computational grid and other GEOgrids at PRAGMA grid member institutions, for example AIST GEO Grid¹² and to empower geosciences applications with both global computational resources and global data resources. Similarly, we are also collaborating with sensor network and application teams, such as Global Lake Ecological Observatory Network (GLEON) (Kratz *et al.*, 2006) and sensor network research teams at Binghamton University and the University of Wisconsin, to incorporate global sensor and data network in the PRAGMA grid to empower scientific applications based on sensor network data.

Many scientists work across different grid projects and their applications should not be restricted by grid boundaries. This requires grids to interoperate and to make it easier for applications to run across grids. To address this goal, we have collaborated with other grids and conducted multi-grid interoperation experiments and peer-grid interoperation experiments.

In early 2006, we joined the Grid Interoperation Now (GIN)¹³ effort under Open Grid Forum (OGF), collaborated with TeraGrid, OSG, NorduGrid¹⁴ and Enabling Grids for E-sciencE (EGEE),¹⁵ and formed the GIN-OPS group. Each grid contributes one or more clusters to form a GIN testbed while the clusters still remain parts of their own production grids. Each grid identified a supporter for their resources. Again, we let the applications drive the grid interoperation experiments. From February to September of 2006, we encountered and resolved many issues and successfully run TDDFT, a Ninf-G based quantum-chemistry application, among four grids – PRAGMA grid, TeraGrid, OSG and NorduGrid. At the end of 2006, we started a second application – a data-intensive Nimrod-based Savannah fire simulation application, to explore data related issues. In a few months, we have been able to run successfully among three grids – PRAGMA grid, OSG and TeraGrid. We also experimented with cross-grid monitoring, by implementing a GIN-testbed infrastructure testing matrix using SCMSWeb. Our SCMSWeb team at Thai National Grid Center led and collaborated with other grid monitoring software developers, such as Ganglia (Sacerdoti *et al.*, 2003), NAGIOS,¹⁶ MonAlisa¹⁷ and NorduGrid monitors teams, to develop a common schema, in order to facilitate data exchange among different monitoring software, thereby enable cross-grid monitoring.

Near the end of 2006, OSG and PRAGMA came together and agreed to start a peer-grid interoperation experiment. Each grid runs an application across both grids while developing minimum tools to enable the application run. PRAGMA runs a Ninf-G based FMO application and OSG runs a spatial interpolation application – GISolve.¹⁸ We started working on the application runs after the New Year in 2007. First, we identified all key persons responsible for this experiment – the overall coordinator and application drivers and resource supporters for each grid. The experiment coordinator created a WIKI space for this experiment, document the above responsibility assignments and ask application drivers on both grids to document their application descriptions and resource requirements at the PRAGMA grid WIKI site. Then we had an initial VTC, to clarify the goals and next milestone, and to exchange information on application requirements and resources between both grids. Immediately after the meeting, we start working on getting the applications to run on both grids. Resource supporters setup access to allow the peer-grid application drivers deploy, test and run their applications. From thereon, we rely on applications to drive the experiment. Application drivers from both grids report application deployment and run progress weekly via e-mail and also document it at our WIKI site. Any issues or problems either applications encounter, the application drivers will immediately e-mail the responsible resource supporters, then the resource supporters will work with the application drivers and try to resolve the issues immediately. All people involved in the peer-grid experiment are included in the e-mail communications and all findings, solutions, workarounds and status are documented at the WIKI site. By the end of February 2007, both applications were able to start running on the two grids and will continue for a long run (more than three weeks) to explore and resolve remaining issues and to allow applications to produce useful scientific results.

Running applications across grids takes advantage of the resources available and thus makes large scale calculations possible. Also grid environment differs and the differences can be learning opportunities for all involved. We picked up good ideas and tools from other grids and from working with our peers. On the other hand, the differences can also be a source of difficulty and confusion for applications and users. Top two areas of difficulty are security policies and job submission. Through the grid interoperation experiments, we were able to learn what the differences are and achieved a much better understanding of the differences. We have worked together to try to provide some interfaces and workarounds to enable interoperability in the short term.¹⁹ Long term and strategic solutions to grid interoperability require a lot more collaborative work and much better understanding of many issues. But, grid interoperation experiments like these are essential. We made a good start and plan to continue.

3 Application middleware

The PRAGMA grid is built on top of a Globus core, and this provides all of the standard low level services. Over and above this we have implemented an upper middleware layer consisting of tools, such as Nimrod/G and Ninf-G, more closely related to the applications (Figure 2, the Applications Middleware). Whilst Nimrod and Ninf-G both support very different programming methodologies, they both had to handle a number of similar issues that arose in the grid, namely fault tolerance, application deployment and setup, and resource configuration. Specifically, Ninf-G implements GridRPC, and thus application programs consist of a set of functions, some of which are invoked by a local procedure call, and some of which are invoked across the network by a remote procedure call. Nimrod/G, on the other hand, runs complete applications on a single grid resource, but achieves parallelism by executing more than one application at a time (although, Nimrod can run parallel applications as well).

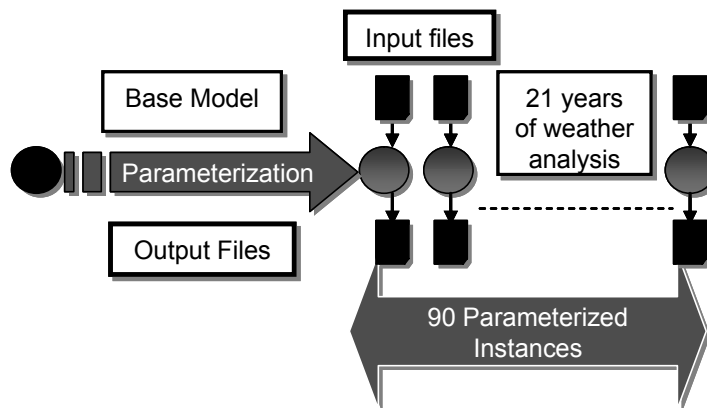
3.1 Fault tolerance

Faults are normal in the operation of the grid. Any resource and network connections can fail at any time. In a RPC based system like Ninf-G, a failure in a remote procedure would normally cause the calling client code to hang. However, Ninf-G monitors the remote procedure, and if a failure occurs, the return code of the call is set to error status, and the call fails accordingly. It is then up to the application programmer to insert code to handle the failure – possibly by causing the procedure to be retried elsewhere. A remote failure is detected by watching status of the TCP connection between the client process and the remote process. In addition to these explicit faults, the routine-basis experiments gave us insights that it is important to detect implicit faults. For example, job invocation requests may be stuck in a batch queue if the system is heavily utilised by the other users; an unstable network may cause a situation in which explicit network disconnection is not detected but no communication between the Ninf-G Client and Ninf-G Servers could be done. Such situations should be considered and detected as faults. Similar implicit faults are also detected and handled by Nimrod/G; in fact, a very large body of the Nimrod/G code is devoted to detect these implicit faults.

Ninf-G allows the programmer to set a timeout parameter on the call, and this prevents the application from hanging if no response occurs. In our routine-basis experiment, we found the timeout mechanism particularly helpful when the network was unstable. In addition, the experimental results guided us to improve the timeout mechanism so that Ninf-G uniquely detected three types of timeout: job start, job terminate, and session. One of the timeout mechanisms is implemented by heartbeat function. The function sends a small-sized packet periodically from server to client. It was originally designed to avoid disconnection by firewall software because there is no packet communication for a certain period of time. However, this function can also be used to check if a connection is still alive or if the server is operating correctly. In the case of that the client does not receive a heartbeat packet, the function regards the connection as dead and causes a timeout on the GridRPC function. This process prevents a client from being blocked to wait for receiving a result for a long time. For example, if the client stalls on a GridRPC API functions because of many packets drop on the network, the function will receive a timeout. When timeout has occurred, the function returns an error code and the server handle is invalidated. The application can detect a timeout error and the application can take appropriate action to recover from the error. It should be noted that the timeout can be set by a client configuration file, which is described in the next section, and application developers do not need to embed the code fragment for detecting timeout in the program. The routine-basis experiment provided Ninf-G developers with opportunities to improve Ninf-G at the points of stability and fault detection, which are not easy to test on a single cluster or a national-level grid interconnected via a high speed network.

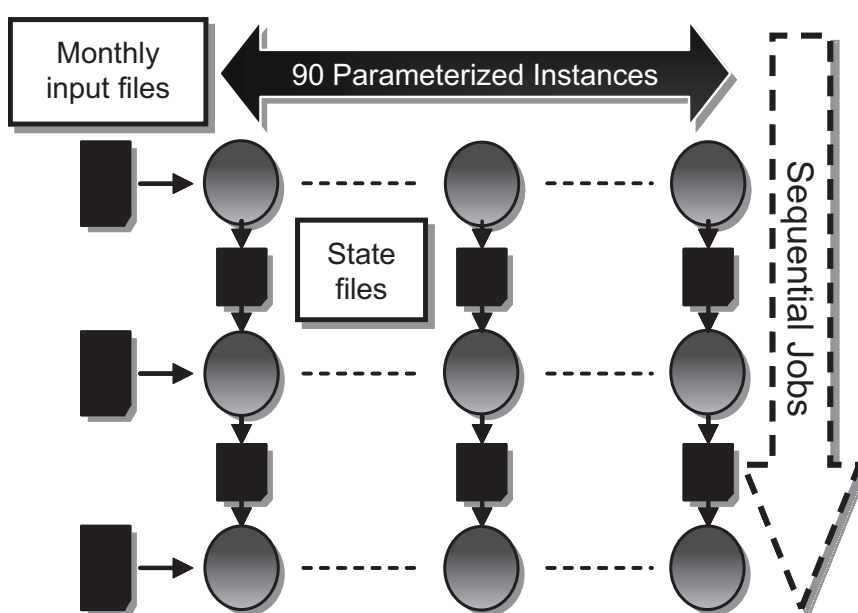
Nimrod/G was equally affected by network and processor faults. Because Nimrod only runs whole programs, a failure normally means that the entire program execution is lost. Accordingly, Nimrod/G includes a retry heuristic that launches the application again in the event of an individual job failure. The job status is reported back to the user through the job monitor, and a user can decide whether the job should be retried or not. To date, this simple retry technique has been powerful enough for all the applications we have run. However, the climate experiment used in this study required us to execute 90 independent model runs (Figure 3), where each one actually ran for a number of weeks! The problem with this structure is that a single job failure would stall the results whilst the single run was repeated, and this would add an unacceptable delay.

Figure 3 Climate run before Nimrod modifications



To solve this problem, we modified the Nimrod/G scheduler to support a new type of parameter called a Seqameter. Normally, parameters in Nimrod/G imply parallel execution of independent tasks. However, Seqameters which specify sequential execution, can be used to enforce dependencies between tasks. Moreover, any data that is produced by one computation and required by a downstream task is copied before the dependent task is started. Importantly, Seqameters and Parameters can be combined to specify quite complex task graphs. In the climate experiment we added a seqameter, called `time_step` to the experiment; thus, Nimrod/G runs each of the 90 simulations for each `time_step` before moving onto the next `time_step`, as depicted in Figure 4.

Figure 4 Nimrod run using a sequential timestep parameter



In this modified scheme. At the end of each time step, the results are copied back to the root node controlling the experiment, and are also left on the scratch storage of the machine that performed the work. When the next time step starts, Nimrod/G copies these output files as input to the next stage – either from the root node or the compute node, depending on which one has better connectivity to the downstream computation. In some circumstances, the scheduler will place the dependent computation on the same node as its predecessor, removing the need to copy the files altogether. In the event of a failure, the standard Nimrod/G retry mechanism is used, but for only one time step, and thus the losses are much smaller. Of course, it is possible to make the `time_steps` quite small, at the cost of increased network copies for the input and output files.

Whilst these changes had a significant effect on the internal structure of Nimrod, they require almost no changes to the application by the end-user scientist. This is because the climate models already understand how to restart from a set of initialisation files, and thus, it is simply a case of running the climate model with a set of restart files. Moreover, the changes required to the Nimrod plan file are minimal, and require the addition of only one new parameter, the time step.

3.2 Application deployment and setup

Both Ninf-G and Nimrod/G rely on executable code being available on a target system. There are two choices for application deployment. One is manual installation, in which the user logs into all remote computers, and installs the applications manually. The advantage is that users can configure the application for the machine architecture and its configuration. The disadvantage is that the work is tedious, repetitive and error prone may leads to version mismatches. The other method is automatic distribution by the staging function in Ninf-G or the remote copy function in Nimrod/G. When users specify the executable to be shipped to the remote in the configuration file of the client, the executable will be transferred before it is spawned on the remote resource. But, this method requires the user to prepare the executable for all remote architectures and to make them available on the root machine.

Ultimately the choice depends on how often the remote executable will be modified and how many target architectures there are. If the remote calculation is stable and does not need frequent modification, automatic distribution would be advantageous, especially if target systems are relatively homogeneous. Similarly, application setup can be done manually or automatically. This phase can also be time consuming. For example, in the Nimrod/G climate study we need to distribute 250 MByte files to each `time_step` – a total of 80 GBytes. In this instance, we chose to ship all files to all resources, giving the scheduler the flexibility to place the computation on any resource. In this case, distribute the file by GridFTP or FEDEX takes about the same amount of time (four days per site), the network was simpler to set up and control. The alternative was to copy each 250 MByte file automatically to the remote resource only when required, however, this adds a substantial startup cost to the computation and therefore was less desirable than just making the entire data set available. We hope to use Gfarm (§4) to resolve this type of problem in the future.

Recently, we have explored an alternative technique for deploying application binaries, and have developed an overarching infrastructure for the deployment of e-science applications called IDEA (Goscinski, 2007). This framework consists of three middleware layers:

- Distributed Ant (DistAnt) (Goscinski and Abramson, 2004; 2005), a user-oriented service-level infrastructure for grid-scale software deployment over heterogeneous uncharacterised resources, which provides a common deployment service and a tool for describing and enacting deployment steps. It allows developers to define and realise a grid which consists of two major components: the DistAnt service and the DistAnt workflow.
- Motor (Goscinski and Abramson, 2006), provides a common environment for software development, deployment and execution at application-runtime. It supports a runtime-internal MPI communication library for parallel processing.
- The Automatic Deployment Tool (ADT) automatically deploys a motor executable and its library dependencies.

This new framework provides users with an overall environment which supports the creation of application grids. DistAnt is user-oriented allowing a regular user to deploy a native or managed application over an uncharacterised and heterogeneous set of resources. The DistAnt service is significant because it defines a common access point and interface for deployment, while the DistAnt workflow provides a facility to describe

and enact the steps to define and realise a grid. In addition, Motor provides a common environment for high performance applications. Thus, users can develop and compile application software for the Motor environment rather than a range of platforms, simplifying application development and deployment. The integration of a message passing library guarantees that the runtime will support a well known parallel communication mechanism. ADT is significant because it automatically deploys a Motor executable and its libraries. Whilst it is early days, our initial experience with IDEA is that it can simplify deployment significantly.

3.3 Configuration of resources

Both Ninf-G and Nimrod/G are designed to handle resource heterogeneity. For example, in Ninf-G, the first argument of a client program must be a 'client configuration file', in which information required for running application is described. In order to compensate for the heterogeneity and unreliability of the grid, Ninf-G provides client configuration formats for detailed description of server attributes such as the port number of the Globus gatekeeper, the local scheduler type and its queue name for submitting jobs, a protocol for data transfers, library path for dynamic linking, disk quota limit, *etc.* In order to cope with the heterogeneity, the values of these server attributes can be specified in a client configuration file. In the routine-basis experiments, we checked the server attributes of all sites (*e.g.*, available Globus jobmanager and parameters which configure the batch queuing system) and prepared a client configuration file prior to the execution of the application. As the number of sites increases, this process causes troublesome overhead for application users. It is expected to implement a mechanism that collects information of these server attributes from information services and create a client configuration file automatically.

The detailed client configuration format enables Ninf-G applications to adapt to heterogeneous configuration of clusters. Similar techniques are available in Nimrod/G; however, the experiment performed here did require some changes that we had not predicted. For example, different file system structures required some minor modifications to Nimrod to run on clusters that did not have a shared file system across the nodes.

Most users choose to create and manage their Nimrod/G experiments using a special portal. The Nimrod Portal contains special interfaces for configuring a grid, and allows a user to associate different certificates with the various grid resources. This is useful when a grid consists of multiple CAs.

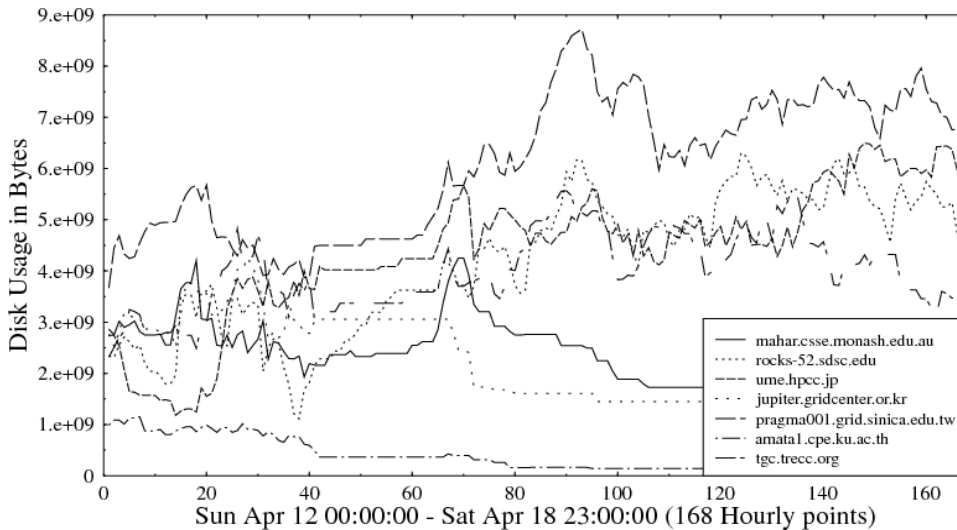
3.4 Storage management

Task-splitting, as described in Section 3.1 solves the fault tolerance problem, but adds a burden to storage management. Nimrod's existing scheduling heuristics assume unlimited disk space on Grid resources (Buyya *et al.*, 2002; Rich Wolski and Hayes, 1998), which is unrealistic on Grids where users are typically allocated limited disk space. In the climate study, each sub-computation generated about 70MB of output. Combined with the common input files of around 250MB in size, this exceeded the storage available on some grid resources. On the other hand, much of the output data is only required temporarily since the output of one sub-task is required only as input to the next sequential task.

To solve this problem we implemented garbage collection to remove all such files to comply with storage restrictions. This allowed unrestricted scheduling of multiple jobs on the same resource without the scheduler having to bother with space constraints. Collecting garbage in this manner solved data management problems to a large extent.

Figure 5 shows the disk space usage of the experiment running on different resources. The graph, which covers a few days of the whole experiment, illustrates dynamic data management activities as disk usage grows and shrinks. An increase in the y-coordinate represents data accumulation, whereas a decrease represents file removal. In the savannah experiment, the disk space usage was directly proportional to the number of jobs executing on the resource. Therefore, a low y-coordinate (disk usage) value in the graph means that fewer jobs executed on that resource at that moment. An oscillation shows that the data is being generated and removed uniformly. The job versus data relationship is important in order to understand the following behavioural analysis.

Figure 5 Storage management in Nimrod



It can be observed that before the 70th hour (approximately), all resources had a smooth oscillatory behaviour, but the disk usage of `jupiter.gridcenter.or.kr`, `mahar.csse.monash.edu.au` and `amata1.cpe.ku.ac.th` decreased afterwards, because these resources became unavailable. The constant disk usage at the end of the graph is due to the presence of common files, which were retained on the resources for subsequent use by other computations. Due to resource unavailability, the jobs were distributed among other available resources, which is evident from the increase in their disk usage patterns. Note how resources such as `ume.hpcc.jp` have shown a more noticeable increase in disk usage compared to others. This happened because these resources had a higher number of computational nodes available, which made them a preferable choice for job allocation. Note as well that the oscillation in disk usage curves became quite uniform after the 100th hour when the load was distributed again properly. This also illustrates the effectiveness of the garbage collection technique.

4 Infrastructure middleware

The infrastructure middleware (Figure 2) provide and enhance the grid infrastructure services, thus benefit applications and grid management. We discuss three examples in this section: Gfarm, SCMSWeb, and MOGAS.

4.1 Gfarm

Gfarm is open source infrastructure software we deployed and primarily tested with iGAP application in PRAGMA Grid. As a grid file system, Gfarm facilitates reliable and robust file sharing and supports high-performance data computing across different administrative domains. It is a scalable virtual file system that federates local file systems of cluster nodes in a grid. Instead of accessing a disk in a distant site, Gfarm utilises a disk in a near site or even in the same compute node. Efficient utilisation of local file system of compute nodes is a key technology for scalable parallel and distributed data computing. File replica management is a well known problem for data grids (Belalem and Slimani, 2007). Gfarm's solution is to manage file replicas in file system metadata for fault tolerance, for a better access to locality and for access concentration avoidance. Furthermore, Gfarm syscall hook library and GfarmFS-FUSE enable existing applications access a Gfarm global virtual file system without any modification. Gfarm syscall hook library emulates of mounting Gfarm file system on /gfarm by trapping system calls for files in applications, where as GfarmFS-FUSE physically mounts a Gfarm file system in user space via FUSE mechanism. This feature makes Gfarm file system a fundamental infrastructure of a shared file system for grid.

4.1.1 Application deployment

Gfarm is a network shared file system for grids that also supports binary execution and shared library loading. Not only application data but also application and shared libraries can be shared in a grid. Once an application is installed, and necessary input data is untarred in Gfarm file system, it can be executed on any cluster node in a grid. No explicit staging of applications and input data is required. Moreover, different kinds of binaries can be stored at the same pathname in Gfarm file system. When two kinds of binaries for Linux and Solaris are installed at /gfarm/bin/igap, appropriate binary is automatically selected and executed depending on the platform architecture. This feature helps to mitigate troublesome application deployment and setup in a heterogeneous environment.

One issue regarding application deployment is how to mount Gfarm file system on a remote node before application execution. Both Gfarm syscall hook library and GfarmFS-FUSE require some setting to mount Gfarm file system; Gfarm syscall hook library requires to set LD_PRELOAD environment variable and GfarmFS_FUSE requires executing mount.gfarmfs to mount Gfarm file system. In case of the syscall hook library, setting the environment variable can be done in a job script. One possible issue is how to execute applications stored in Gfarm file system. To execute program in Gfarm file system, this environment variable setting needs to be done before program execution somehow. We solve this problem by executing a wrapper script instead of executing an application program directly to define necessary environment variables before program execution.

In case of GfarmFS-FUSE, `mount.gfarmfs` needs to be executed before program execution. To do this, we also utilise another wrapper script to mount a Gfarm file system, to execute a specified program, and to unmount the Gfarm file system.

Another issue is related to authentication in Gfarm file system. In the PRAGMA Grid, GSI authentication is used among file servers in Gfarm. Every process needs to have a valid proxy certificate to mount and access to a Gfarm file system. This requires a job submission system to delegate a user proxy certificate to every remote process.

4.1.2 Performance

During an installation and setup process of iGAP, genome annotation software, we encountered a performance problem. Gfarm has been originally developed to improve reading and writing performance of large files. In this case the total I/O throughput is more critical than fast access to metadata. However, the installation and set-up process requires copying or untarring thousands of files including programs and input data. In this case, the metadata access overhead is crucial to performance. To reduce the overhead, we have improved the metadata cache management, and introduced a metadata cache server to share the metadata cache among different applications. Original implementation obtained pathname of all files several times for directory listing. This comes from a schema design of file system metadata that does not allow to obtain a list of a given specific directory, and naive implementation of directory listing. At first, we changed blocking database queries to non-blocking database queries, which reduces the metadata retrieval time especially when there are large number of files. Second, we eliminated unnecessary metadata access not to obtain the same data again by maintaining a modification time of a directory in the metadata cache. As a result, the performance improved dramatically. For example, directory listing of 16 393 files took 44.0 s. Improving the metadata cache management reduced this to 3.54 s. To improve the performance further, we have introduced a metadata cache server. It keeps file system metadata so that different applications share the metadata cache. Using the metadata cache server, every application does not need to acquire pathname of all files in directory listing. In the previous example, the introduction of a metadata cache server at the client further reduced this to 1.69 s.

4.1.3 Fault tolerance

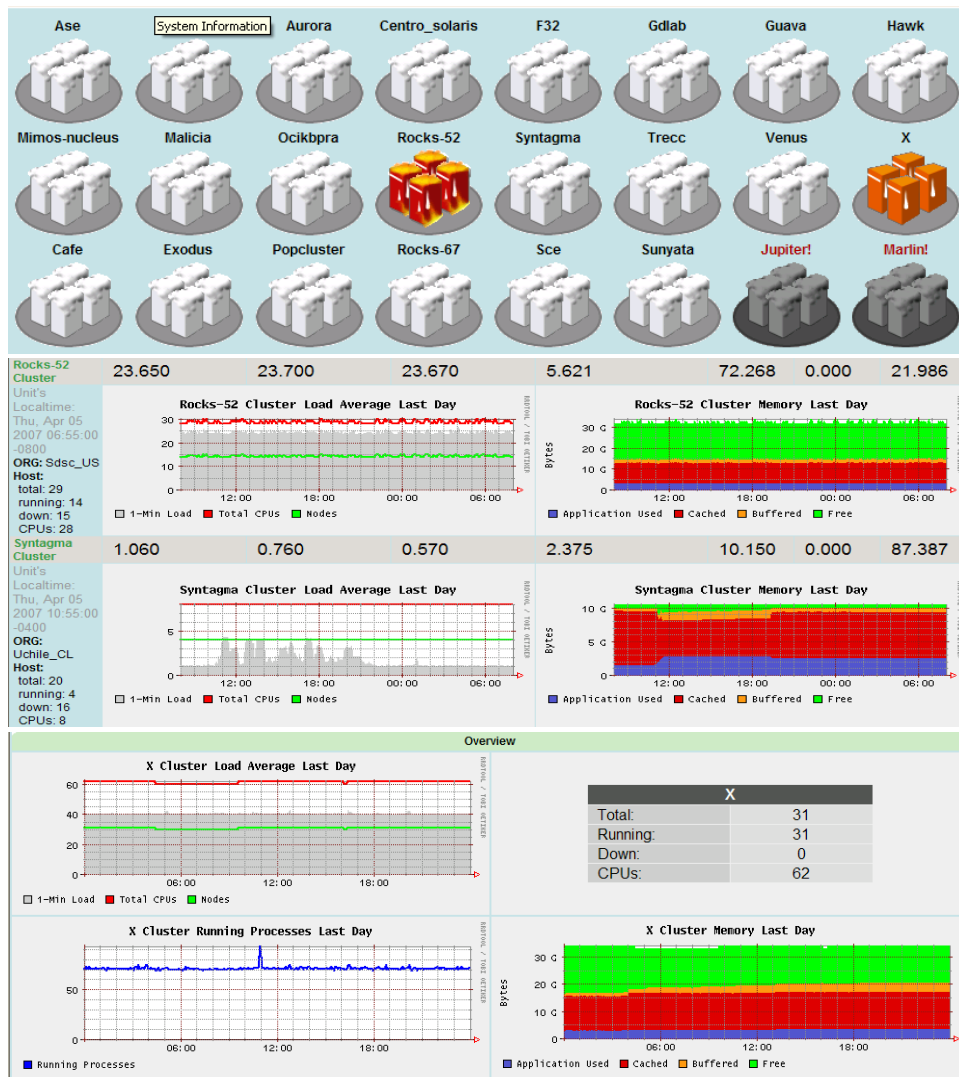
Gfarm file system supports fault tolerance in case of disk and network failure by automatic file replica selection in access time. On the other hand, this feature works only when there is an available file replica somewhere. To ensure this, automatic file replica creation mechanism is required. Since Gfarm implementation in PRAGMA Grid is a worldwide distributed file system, there are several requirements for the mechanism. One is to have at least two file replicas for each file in case of disk failure, and another is at least one file replica for each file at each site for performance and fault-tolerance. We enhanced feature of a file replica creation command to meet the requirements. To create the specified number of file replicas, it selects the required number of less loaded file system nodes, and creates a file replica on these nodes. To create a file replica at each site, introduce file selection to pick up files that one of file replica is stored in the specified domain name, and destination host selection by the domain name. If the required number of file replicas is already stored in the destination hosts in the

specified domain name, no action happens. Currently, we assume this command is regularly executed by cron to achieve requirements in file replica creation in the PRAGMA Grid.

4.2 SCMSWeb

SCMSWeb was the first infrastructure software deployed in the PRAGMA grid. It serves as the eyes to the grid, which in turn it helped SCMSWeb to discover problems and improve in stability, scalability and performance. SCMSWeb monitors and reports a wide variety of system usage and performance metrics (such as load average, CPU/memory usage, disk/network I/O activity, page/swap/context switching rate, system temperature) as shown in Figure 6.

Figure 6 Grid meta view and system information view



4.2.1 Heterogeneity

The PRAGMA grid heterogeneity arises from the diversity of site physical locations, network conditions, hardware architectures, operating systems (*e.g.*, Linux, Solaris), software stacks and versions, *etc.* This characteristic provides a realistic global grid environment and challenges to our software and also provides a collaborative environment for software improvements. For example, prior to testing in the PRAGMA grid, SCMSWeb supported limited platforms. Whence SCMSWeb being deployed in the PRAGMA grid, a Solaris site at Centro de Investigacion Cientifica y de Educacion Superior deEnsenada (CICESE), and an IA64 site at Computer Network Information Center (CNIC), volunteered to help porting SCMSWeb. With access to these platforms and help at these sites, we were able to port SCMSWeb to Solaris and IA64 in only a few weeks. Further, as SCMSWeb was deployed in the PRAGMA grid, we received a lot of feedback from sites, including problem reports, new feature and interface change requests. These valuable input stimulated fast improvements in software reliability and usability.

The PRAGMA grid heterogeneity also provides technical challenges in information exchange and storage. Driven by the demand of PRAGMA test-bed, the SCMSWeb team decided to use an XML format to describe most of the information processed, exchanged, and stored. The main advantage of XML is portability. We also found that using XML allowed us to easily extend the format of information to accommodate new features, while maintaining the compatibility between each version of software.

4.2.2 Reliability

Another major issue presented by the PRAGMA grid was reliability of the grid. This arose from the large distances between resources and the different levels of networking infrastructure in each country. Our experiences show that most grid reliability problems were caused by either network failure or poorly configured Globus software. To monitor such problems, a set of probes were developed and added to SCMSWeb to test network and Globus services, and also GFarm Grid file system services, which has been deployed as the main file system services in PRAGMA grid. This feature helps site administrators to detect and fix problems in a more timely fashion, substantially improved the reliability and productivity of the grid. An example of probes view is shown in Figure 7.

4.2.3 Software deployment

Software deployment in large-scale grid is also a challenging problem. Currently, software distribution is done using the Linux RPM format. However, dependency can be a problem on for some Linux distributions. So, we adopted the solution to regenerate the RPM source file (.src.rpm) and to install this source on the system. On systems such as Solaris, the desired format is source code packed as a compressed tar file. The installation order is also a concern, so, care must be taken to preinstall required packages before compiling SCMSWeb. For newly installed clusters, users can install the SCE roll that comes with Rocks, which includes a preinstalled and pre-configured version of SCMSWeb. This makes SCMSWeb installation and configuration very easy. With both the source distribution method and the roll distribution method, we have managed to reduce the installation effort while still cover wide range of demands arose within a heterogeneous grid.

Figure 7 Probe view show grid service status

::: PRAGMA Status:::								
Site	Summary	Authentication (20/2/7)	DNS (26/0/3)	Gfarm (6/0/23)	GridFTPFr (13/7/9)	GridFTPTo (13/7/9)	JobRun (18/3/8)	Mds (5/12/12)
sunyata.thaigrid.or.th	(7/0/0)	●	●	●	●	●	●	●
rocks-52.sdsc.edu	(6/1/0)	●	●	●	●	●	●	●
ase.nchc.org.tw	(6/0/1)	●	●	●	●	●	●	●
guava.ngpp.ngp.org.sg	(6/0/1)	●	●	●	●	●	●	●
malicia.super.unam.mx	(6/0/1)	●	●	●	●	●	●	●
pragma.sdg.ac.cn	(6/0/1)	●	●	●	●	●	●	●
pragma001.grid.sinica.edu.tw	(6/0/1)	●	●	●	●	●	●	●
venus.ioit-hcm.ac.vn	(6/0/1)	●	●	●	●	●	●	●
marlin.bii.a-star.edu.sg	(5/1/1)	●	●	●	●	●	●	●
ocikbpra.unizh.ch	(5/1/1)	●	●	●	●	●	●	●
fsvc001.asc.hpcc.jp	(5/0/2)	●	●	●	●	●	●	●
grid64.hpcc.nectec.or.th	(5/0/2)	●	●	●	●	●	●	●
xsvc.asc.hpcc.jp	(5/0/2)	●	●	●	●	●	●	●
hawk.usm.my	(3/3/1)	●	●	●	●	●	●	●
pop.cs.binghamton.edu	(3/3/1)	●	●	●	●	●	●	●
nucleus.mygridusbio.net.my	(3/2/2)	●	●	●	●	●	●	●

4.2.4 Performance

Initially, SCMSWeb was developed as a cluster monitoring system. As the number of sites in PRAGMA grid increased, there was a need to enhance both scalability and performance of SCMSWeb. To handle the scalability requirement of the grid, we enhanced SCMSWeb to support a hierarchical monitoring structure. In this structure, a higher level node can easily pull system status information from lower level systems. The topology of monitoring tree depends on the agreement among the participating sites. Currently, the root node of the monitoring tree for PRAGMA grid is located at <http://goc.pragma-grid.net/scmsweb>. With this enhancement, a system administrator can easily get most of the information needed to understand the grid operational characteristic from root node.

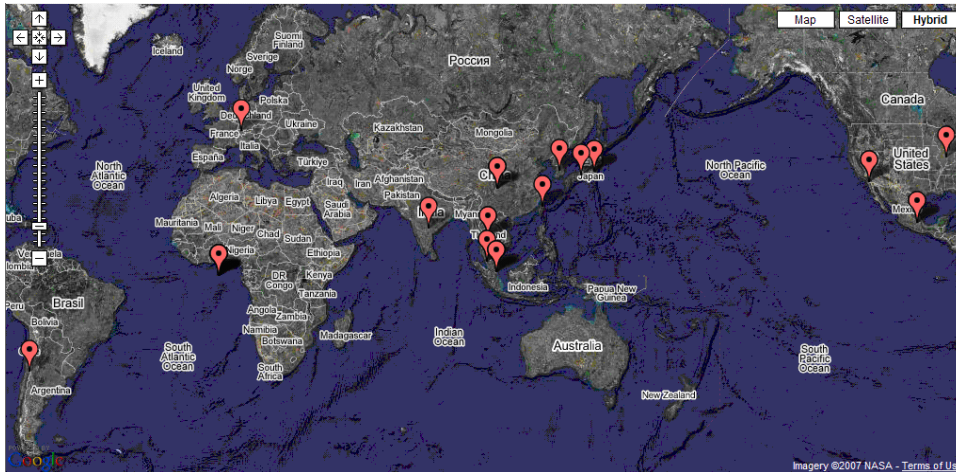
When network bandwidth is low, we found that the transfer of large monitoring information tends to be delayed or fails. This causes some information to be loss or not updated. To avoid this problem, we improved SCMSWeb by adding data compression during the transfer process. Since the original data is text based XML data, we successfully reduced the size of data being transferred up to ten times. As a result, the performance and stability has been improved substantially.

4.2.5 Features

During the course of SCMSWeb development, PRAGMA interaction has driven our development team to add many useful features into SCMSWeb. For example, during the interaction with the GIN project (see Section 2), SCMSWeb is fine-tuned to be much more portable such as the addition of GLUE schema²⁰ support that enable SCMSWeb to interact in a standard way to very heterogeneous Grid environment around the world. In

addition, a feature called GeoMap was developed to enable SCMS to interface with the popular Google map and exchange geographical information. As a result, users can now use SCMSWeb to see the exact geographical location of the grid site (Figure 8).

Figure 8 SCMSWeb GeoMap features the real geographical location of the Grid site



Another recent development is the addition of a bandwidth measurement infrastructure. As more data intensive applications are deployed, the need to discover a network bottleneck becomes more important. SCMSWeb routinely collects the bandwidth measured to each node using *iperf*²¹ tool and store the information in SQL database (MySQL).²² This information can be accessed directly from windows based PC using ODBC support. Hence, system administrator can quickly analyse the bandwidth information using standard tool such as excel.

During PRAGMA12 meeting in March 2007 (Bangkok) this new feature is used to demonstrate the bandwidth over Thailand Research and Education Network (ThaiREN)/ Trans-Eurasia Information Network (TEIN2) between PRAGMA nodes in Thailand to several grid nodes in Hong Kong, Singapore, Beijing (China), and Europe. Many interesting performance issues have been discovered such as TCP/IP acknowledgment limitation over a long distance high speed network. Some of the parameters such as TCP/IP buffer size has been tune up which results in a jump in performance. This feature is now being deployed and used extensively in ThaiGrid (Figure 9), a National Grid Infrastructure in Thailand, and will soon be widely deployed in PRAGMA grid.

4.3 MOGAS

Multiple Organization Grid Accounting System (MOGAS), developed by Nanyang Technological University in Singapore, has been deployed to 14 sites in PRAGMA grid. It has the objective that the tool should provide the resource owner with the overview of the resource usage across the grid test-bed based on the user organisation. Thus, different from just monitoring cluster usage MOGAS must also be able to detect the level of inter-organisation resource usage/sharing. MOGAS complimenting the SCMSWeb tool with the grid-wide view of resource usage and sharing.

Figure 9 Bandwidth measurement in SCMSWeb showing bandwidth among sites

Bandwidth Matrix (Mbits/sec)																		
From/To	ABAC	Araya	CSIM	DEQUEUE	ENQUEUE	Eastern	Gridmd	MAEKA	Ocean	Pluto	Radiant	Rdcgrid2	SRITHAN	Sutgrid	TubkaewGrid	cmu_grid	gcc.thaigrid.or.th	grid.uni.net.th
ABAC	X	8.47	0.07	10.60	-	-	0.14	12.90	-	6.53	10.80	7.20	-	0.07	0.14	7.25	10.10	0.17
Araya	4.74	X	48.90	69.10	-	-	0.76	826.00	-	74.00	94.20	93.20	-	35.40	57.90	27.60	754.00	482.00
CSIM	0.17	67.40	X	38.60	-	-	0.78	50.40	-	32.00	52.40	68.20	-	25.60	58.70	18.80	63.50	76.80
DEQUEUE	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENQUEUE	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-
Eastern	4.60	73.00	33.80	56.90	-	X	0.77	71.50	-	38.70	68.20	62.70	-	28.30	41.80	20.80	76.00	76.00
Gridmd	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-
MAEKA	4.75	788.00	53.60	87.50	-	-	0.78	X	-	72.20	93.40	93.40	-	42.60	49.60	28.10	849.00	415.00
Ocean	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-
Pluto	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
Radiant	4.39	94.20	54.60	77.10	-	-	0.76	94.20	-	58.40	X	92.10	-	31.60	60.10	24.90	94.20	94.90
Rdcgrid2	1.41	82.60	25.30	35.70	-	-	0.29	89.90	-	-	11.80	X	-	13.00	19.10	11.40	90.10	91.00
SRITHAN	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-
Sutgrid	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-
TubkaewGrid	0.02	75.80	56.00	68.50	-	-	0.78	73.80	-	48.40	75.50	63.30	-	29.70	X	18.70	77.20	78.70
cmu_grid	4.70	28.30	26.70	27.20	-	-	0.78	27.30	-	22.40	29.70	26.70	-	15.30	24.10	X	29.70	30.90
gcc.thaigrid.or.th	2.31	731.00	35.00	45.30	-	-	0.31	651.00	-	-	65.70	80.10	-	17.20	30.90	17.20	X	520.00
grid.uni.net.th	-	386.00	54.20	90.00	-	-	0.81	435.00	-	60.70	93.30	93.20	-	37.30	50.00	28.40	592.00	X
hanuman	0.63	3.67	11.10	2.73	-	-	0.78	3.74	-	5.59	2.80	2.98	-	2.96	4.97	2.56	3.66	3.28
inca	4.72	61.80	52.40	65.70	-	-	0.77	67.10	-	48.20	64.50	68.50	-	28.80	38.90	24.10	66.50	61.20
mu-hpc	-	151.00	18.00	10.60	-	-	0.18	174.00	-	60.60	8.24	7.45	-	14.10	14.90	22.90	243.00	226.00
sunyata	3.93	877.00	50.60	87.10	-	-	0.78	824.00	-	70.00	94.20	93.20	-	27.90	51.90	26.90	942.00	463.00

MOGAS was developed to log the job submission across multi-organisation clusters. It is built on top on of Globus core: Globus Resource Allocation Manager (GRAM), GSI and Grid File Transfer Protocol (GridFTP). MOGAS supports a number of well known local cluster schedulers. The PRAGMA grid provides a good, heterogeneous environment and diverse set of job schedulers (various flavours of PBS, LSF, and SGE) for MOGAS testing. Working in PRAGMA grid, we found that different GRAM scripts and different security policies among PRAGMA grid sites required us to improve our deployment procedure, automate and take into account all of the various requirements, and ease its installation and setup. Also, after the MOGAS deployment in the PRAGMA grid, we received significant feedback and suggestions, and these resulted in immediate improvements and provided us with clear directions for future enhancements.

Some of the features available in MOGAS include: overall consumer/provider usage view in a table matrix form (job and runtime); resource trading ratio view; resource workload view; the top 10 resource providers and consumers; and Automatic account report generation.²³ Figure 10 shows the consumer/provider table matrix. It shows the percentage of job submitted and the total number of jobs submitted by each organisation in last column in the table.

An analysis of the log data (Lee *et al.*, 2006) shows that the inter-organisation resource usage is high and the test-bed is quite stable, with some jobs running successfully for more than 50 hours. We also observe the job type and characteristics using the different middleware and fine tune the middleware and application to make effective use of the grid.

The deployment has also highlighted the scalability issue in MOGAS. The current design is based on a centralised database server. As more resources are added to the test-bed and more jobs submitted across the test-bed, the central server would not be able to cope with the log updates from all the sites. Thus, a decentralised system is proposed so that different sites can be federated together, *e.g.*, by country.

Figure 10 MOGAS consumer/provider matrix table

Consumer/Provider Overview of Job Numbers:

Rows are consumers and columns are providers

Org	ASCC	BH	CCESE	CNIC	DECAS	ILLU	KU	KISTI	MU	NCHC	ABT	NCSA	UNAM	SDSC	ZITFCU	UCLL	U-His	UMC	UZsich	USM	NTU	MIMOS	IOIT-ICM	Total
ASCC	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	7
BH	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
CCESE	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
CNIC	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
DECAS	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
ILLU	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
KU	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
KISTI	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
MU	4.384%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	97154
NCHC	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
ABT	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5679
NCSA	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	3
UNAM	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
SDSC	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.817%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1449
ZITFCU	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
UCLL	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
U-His	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
UMC	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
UZsich	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	4469
USM	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	62
NTU	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	3
MIMOS	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
IOIT-ICM	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	12
Total	299	0	0	0	0	0	0	14092	0	0	60478	19241	0	11507	0	0	0	0	0	0	0	126	87	108830

Legend:

- Jobs submitted by the Consumer (row)>50% of Total Jobs submitted to a particular Provider (col)
- 25%<Jobs submitted by the Consumer (row)<50% of Total Jobs submitted to a particular Provider (col)
- Jobs submitted by the Consumer (row)<25% of Total Jobs submitted to a particular Provider (col)

5 Conclusion

This paper described more than just the issues encountered when attempting to run applications across an international federation of loosely coupled machines. The shared vision of PRAGMA members has significantly lowered the resistance of the group to use others' software. Not only have PRAGMA software developers evaluated software developed in other institutions and countries, but also several of these have formed new development collaborations. While many of our observations are hardly surprising, the problems of heterogeneity (system, OS, network latency and performance), fault management, software coordination, language barriers, time zones, identity management, and storage performance are ones that must be dealt with to achieve a functional grid. More specifically, through our collective efforts we found and achieved the following:

- Faults (hardware, software and network) are normal and must be handled both by middleware and in some cases, applications themselves.
- The PRAGMA grid concurrently supports a rich variety of application needs, including high performance computation, and high bandwidth data access.
- The PRAGMA grid is large, allowing applications to explore issues that would need to be dealt with at massive scales such as, resource availability, network bandwidth and contention, distributed resource monitoring, and global name spaces.
- Coordination of resources, monitoring, inter-operable software deployments, and resource/user policies are significantly complicated because applications may have different requirements for middleware and resource allocation. Both application users and system administrators are located across 12 time zones. This presents a challenge in putting together a team with very diverse cultural backgrounds.

- The PRAGMA grid covers many countries with different levels of network speed, quality, and system resources. Thus, the situation that middleware has to deal with is more realistic, and more challenging than grids that are built only in countries with strong infrastructure in place. New types of middleware can be deployed and tested in a wide variety of administrative and physical environments.
- A telling value of the PRAGMA grid is how its use created strong feedback between application and middleware (*e.g.*, Ninf-G, Nimrod and Gfarm). Software had to be improved, then deployed quickly (*e.g.*, SCMSWeb, MOGAS). Finally, collaboration among PRAGMA grid sites helped to improve the software (*e.g.*, porting SCMSWeb to Solaris and ia64).

Applications in variety of fields have driven the middleware requirements and the PRAGMA grid provides a mechanism for users around the world to share and use software that goes beyond the essential, *de facto* standard Globus core. Some experiences from the application perspective are reported in CCGrid 2006 (Abramson *et al.*, 2006). We've found that at the initial stage, by setting absolute minimum grid-wide software requirements (Globus and a local scheduler) with additional middleware dependent on application needs, deployment across all institutions in a grass-roots global grid is more realistic and manageable. But the infrastructure middleware components could become part of the required software infrastructure in the future (Figure 2).

Given daunting issues such as lack of centralised control and routine system faults, it would seem that such a system would simply be unusable. However, our experience has shown just the opposite. Real science can be performed on such a system, and the environmental challenges greatly improve the middleware systems. We hope that our experience will inform other emerging grid efforts in operations of their grids, and concurrently will focus research development on the practical needs of these grids.

Acknowledgements

This work was supported in part by the US National Science Foundation under Awards INT-0216895, INT-0314015, SCI-0627026; by the Ministry of Education, Sports, Culture, Science and Technology of Japan through the National Research Grid Initiative Project; by Australian Government under a variety of awards including the Cooperative Research Centre scheme, the Departments of Education Science and Technology and Communications, Information Technology and the Arts, the Australian Research Council and Monash University; by the Thailand Research Fund and Kasetsart University Research and Development Institute SRU fund; by a grant from the Ministry of Information and Communication through the K*Grid project; and by National Grid Office of Singapore.

The authors also wish to acknowledge the support of all members of the PRAGMA Resources and Data Computing Working Group and of the PRAGMA grid. These include Academia Sinica Computing Center, Taiwan; Australia Partnership for Advanced Computing, Australia; Binghamton University, USA; Bioinformatics Institute, Singapore; Center of Investigation Science and Education at Superior Ensenada, Mexico; Computer Network Information Center, Chinese Academy of Science, China; Graduate University of Chinese Academy of Sciences, China; Institute of High

Performance Computing, Singapore; HCMC Institute of Information Technology, HCMC, Vietnam; Jilin University, China; Korea Institute of Science and Technology Information, Korea; MIMOS Berhad, Malaysia; Monash University, Australia; National Center for High-performance Computing, Taiwan; National Center for Supercomputing Applications, USA; National Electronics and Computer Technology Center, Thailand; National Grid Office of Singapore; National Institute of Advanced Industrial Science and Technology, Japan; National University of Mexico, Mexico; Osaka University, Japan; San Diego Supercomputer Center, USA; ThaiGrid, Thailand; Tokyo Institute of Technology, Japan; University of Chile, Chile; University of Hyderabad, India; University of Sains Malaysia, Malaysia; University of Utah, USA; University of Zurich, Switzerland.

References

- Abramson, D., Buuya, R. and Giddy, J. (2002) 'A computational economy for grid computing and its implementation in the Nimrod-G resource broker', *Future Generation Computer Systems*, October, Vol. 18, No. 8.
- Abramson, D., Lynch, A., Takemiya, H., Tanimura, Y., Date, S., Nakamura, H., Jeong, K., et al. (2006) 'Deploying scientific application on the PRAGMA grid testbed: ways, means and lessons', *CCGrid 2006*.
- Belalem, G. and Slimani, Y. (2007) *A Hybrid Approach to Replica Management in Data Grids IJWGS*, Vol. 3, No. 1, pp.2–18.
- Buyya, R., Abramson, D. and Murshed, M. (2002) 'A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids', *The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, Nevada, USA.
- Chalakovkosol, N. and Uthayopas, P. (2003) 'Monitoring the dynamics of grid environment using grid observer', *Poster Presentation in IEEE CCGRID2003*, Toshi Center, Tokyo, 12–15 May.
- Darling, A., Carey, L. and Feng, W. (2003) 'The design, implementation, and evaluation of mpiBLAST', *4th International Conference on Linux Clusters: The HPC Revolution*, San Jose, California, June.
- Foster, I. and Karonis, N. (1998) 'A grid-enabled MPI: message passing in heterogeneous distributed computing systems', *Proc. SuperComputing 98 (SC98)*, Orlando, FL: IEEE.
- Foster, I. and Kesselman, C. (1997) 'Globus: a metacomputing infrastructure toolkit', *International Journal of Supercomputer Applications*, Vol. 11, No. 2, pp.115–128.
- Goscinski, W. (2007) 'IDEA: an infrastructure for the deployment of e-science applications', PhD thesis, Monash University, Australia.
- Goscinski, W. and Abramson, D. (2004) 'Distributed ant: a system to support application deployment in the grid', *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pp.436–443.
- Goscinski, W. and Abramson, D. (2005) 'Application deployment over heterogeneous grids using distributed ant', *First International Conference on e-Science and Grid Technologies (e-Science 2005)*, pp.361–368.
- Goscinski, W. and Abramson, D. (2006) 'Motor: a virtual machine for high performance computing', *The Fifteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC)*.
- Kratz, T.K., Arzberger, P., Benson, B.J., Chiu, C-Y., Chiu, K., Ding, L., Fountain, T., et al. (2006) *Toward a Global Lake Ecological Observatory Network*, Publications of the Karelian Institute, Vol. 145, pp.51–63.

- Lee, B-S., Tang, M., Zhang, J., Soon, O.Y., Zheng, C., Arzberger, P. and Abramson, D. (2006) 'Analysis of job in a multi-organizational grid test-bed', *International Workshop of Grid Test-Bed Held in Conjunction with ccGrid'06*.
- Li, W.W., Arzberger, P.W., Yeo, C.L., Arg, L., Tatebe, O., Sekicughi, S., Jeong, K., Hwang, S., Date, S. and Kwak, J-H. (2005) 'Proteome analysis using iGAP in Gfarm', *Proceedings of 2nd International Workshop on Life Science Grid (LSGRID 2005)*.
- McGregor, J.L., Walsh, K.J. and Katzfey, J.J. (1993) 'Nested modelling for regional climate studies', in A.J. Jakeman, M.B. Beck and M.J. McAleer (Eds.) *Modelling Change in Environmental Systems*, J. Wiley and Sons, pp.367–386.
- Papadopoulos, P.M., Katz, M.J. and Bruno, G. (2001) 'NPACI rocks: tools and techniques for easily deploying manageable Linux clusters', *Proceedings of 2001 IEEE International Conference on Cluster Computing*, Newport, California, October.
- Paterson, L. and Roscoe, T. (2004) *The Design Principles of PlanetLab*, PlanetLab Consortium, PDN-04-021, June.
- Pennington, R. (2002) 'Terascale cluster and the TeraGrid', *Proc. of HPC Asia 2002*, pp.407–413.
- Rich Wolski, N.T.S. and Hayes, J. (1998) *The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing in Future Generation Computer Systems*.
- Sacerdoti, F.D., Katz, M.J., Massie, M.L. and Culler, D.E. (2003) 'Cluster computing, 2003', *Proceedings. 2003 IEEE International Conference*, 1–4 December.
- Tanaka, Y., Takemiya, H., Nakada, H. and Sekiguchi, S. (2004) 'Design, implementation and performance evaluation of GridRPC programming middleware for a large-scale computational grid', *Fifth IEEE/ACM International Workshop on Grid Computing*, November, pp.298–305.
- Tatebe, O., Soda, N., Morita, Y., Matsuoka, S. and Sekiguchi, S. (2004) 'Gfarm v2: a grid file system that supports high-performance distributed and parallel data computing', *Proceedings of the 2004 Computing in High Energy and Nuclear Physics (CHEP04)*, Interlaken, Switzerland, September.
- Uthayopas, P., Phatanapherom, S., Angskun, T. and Sriprayoosakul, S. (2001) 'SCE: a fully integrated software tool for Beowulf cluster system', *Proceedings of Linux Clusters: The HPC Revolution*, National Center for Supercomputing Applications, University of Illinois, Urbana, IL, 25–27 June.
- Yabana, K. and Bertsch, G.F. (1999) 'Time-dependent local-density approximation in real time: application to conjugated molecules', *Quantum Chemistry*, Vol. 75, pp.55–66.

Notes

- 1 QM/MD, <http://www.globusworld.org/program/abstract.php?id=64>.
- 2 FMO, <http://sc05.supercomputing.org/schedule/pdf/pap138.pdf>.
- 3 MM5/WRF, <http://box.mmm.ucar.edu/products/models.html>.
- 4 Mpich-gx, <http://www.moredream.org/MPICH-GX/pub.html>.
- 5 CSF, http://nbc.scd.edu/pub/wiki/index.php?title=Community_Scheduler_Framework.
- 6 Skype, www.skype.com.
- 7 <http://goc.pragma-grid.net/wiki>
- 8 NAREGI, http://www.naregi.org/index_e.html.
- 9 CSE-Online, <http://cseo.net/>.
- 10 OSG, <http://www.opensciencegrid.org/>.
- 11 GEON, <http://www.geogrid.org/>.
- 12 AIST GEO Grid, <http://www.geogrid.org/gridsphere/gridsphere>.

- 13 <https://forge.gridforum.org/projects/mgi> Charter document by C. Catlett and M. Satsuoka.
- 14 NorduGrid, <http://www.nordugrid.org/>.
- 15 EGEE, <http://public.eu-egee.org/>.
- 16 NAGIOS, <http://nagios.org/>.
- 17 MonAlisa, <http://monalisa.cern.ch/monalisa.html>.
- 18 GISolve, <http://grow.uiowa.edu/dokuwiki/doku.php/projects/gisolve/index>.
- 19 http://goc.pragma-grid.net/wiki/index.php/Main_Page#Grid_Inter-operations
- 20 GLUE schema, <http://glueschema.forge.cnaf.infn.it/>.
- 21 Iperf, <http://sourceforge.net/projects/iperf>.
- 22 MySQL, <http://www.mysql.com/>.
- 23 <http://ntu-cg.ntu.edu.sg/pragma>