

Troubleshooting and Fault Tolerance in Grid Environments Workshop^{*}

Chicago, Dec 11th 2002

Final version: Mar 4th 2003

1.1 Introduction and Overview

The workshop brought a broad range of application and computer science researchers and practitioners to the table to exchange ideas on what can and should be done to enhance the ability of computational Grids to deliver dependable and sustained performance given the current “state-of-the-art” of Grid technology. Recent experience in deploying Grid middleware demonstrated the challenges one faces in delivering robust services in a distributed environment. While end-to-end performance issues are clearly part of these challenges, the main focus of the workshop was issues related to keep a Grid and the services it provides “up and running”. See Appendix for the list of participants.

While the focus was on the high energy physics applications of the organizers from the physics grid projects, contributions from the NASA Information Power Grid, NCSA, DOE science grid, LBNL, Oracle and several university Computer Science groups (University of Chicago, Wisconsin, Cornell) helped ensure that attention was broadly addressed to the general problems.

The scope of PPDG especially is to bring short to medium term benefits of deployed grids to the Monte Carlo production, data processing and data analysis needs of the collaborating experiments, with additionally providing platforms for alpha testing to provide early feedback of requirements to CS groups. The collaborating Computer Science groups were well represented, and there were contributions from other researchers in the SciDAC and Grid middleware programs. Talks from D0, BaBar, CMS, ATLAS showed that these experiments are looking for short-term solutions for the problem they are facing today while thinking about long-term solutions that will address their sustainable needs for the longer term.

This report provides a summary of the scope, issues and directions stemming from the workshop.

1.2 Requirements for Grid Monitoring and Troubleshooting

Physics experiments are starting to deploy Grid software on complex distributed facility and fabric infrastructures. A single experiment (for example BaBar) has dedicated resources amounting to thousands of processors, hundreds of servers hosting hundreds of terabytes of disk, and several robotic mass storage systems. The facility is spread over major centers in 5 countries plus 10 – 20 significant university-based clusters. Major centers are often used primarily for data-intensive processing and physics analysis whereas many of the university-based clusters concentrate on the less data-intensive simulation task.

The distributed facility and user community is a reality. The challenge of Grid technology is to make much more effective use of the complete set of resources, the support personnel and the physicists, than can be achieved through the use of pre-grid and *ad-hoc* tools.

Users, systems support, and operations support all have their goals and requirements for Grid technology. Like for any other computing facility, at the highest level these converge to a simple goal that a computational Grid should appear as a seamless resource where failures that require

^{*} <http://www.ppdg.net/mtgs/Troubleshooting/agenda.htm>

human intervention are rare and monitoring and troubleshooting tools allow problems to be identified and fixed rapidly and with minimal effort. What makes this goal hard to achieve is the distributed nature, both in terms of ownership and location, of the Grid.[†]

Experience reported at the workshop indicated that, with substantial additional automation of failure detection and response, some existing Grid tools could be very cost-effective in well-controlled production environments running a limited range of applications. However, the experience from production testbeds with greater heterogeneity of site fabric elements or applications indicated that a more advanced monitoring and troubleshooting framework is needed.

The workshop reinforced the view that the time is right to increase the focus of the Grid community on the system wide and application deep exception handling, recovery procedures and monitoring and analysis frameworks.

1.2.1 Physicists' Requirements

1. A way for an application to reliably and recoverably (ie upon error not hang or crash) request a service, obtain status information and receive notification upon completion or failure.
2. A way for an application to reliably log information about its activities.
3. A way for the Grid middleware to distinguish between a fabric or a middleware failure and an application exception, with control of the tradeoff between this capability and performance
4. A way for an application to be automatically and cleanly (all garbage from the previous run removed) restarted in the presence of a failure in Grid middleware, application or fabric, with control of the tradeoffs between the capability and performance.
5. A way to discover expected Grid performance for an application. Is the performance worth investigating or complaining about?
6. A suite of monitoring and diagnostic tools integrated with the application, the Grid software and the fabric to help in locating the origins of a problem. Few physicists like to make vague "it doesn't work properly" complaints. Most are well aware that the problem may lie in their own or their colleagues code. This also requires careful instrumentation of the application not just the Grid services.
7. Excellent configuration control at all levels. Knowing what has changed can short-circuit a resource-intensive investigation.
8. Someone to call.

1.2.2 Systems and Operations Requirements

9. A way to discover the expected behavior and performance of Grid services and fabric subsystems. If the performance is as expected given the existing fabric, where are the bottlenecks?

[†] The issue of complexity limiting the capabilities of computing resources is identified by IBM in an initiative called Autonomic Computing (ref).

10. A suite of monitoring and diagnostic tools integrated with the application, Grid middleware and the fabric.
11. Excellent configuration control at all levels.
12. Automation of monitoring, diagnostic and recovery procedures. Comments on the requirements

The promise of the Grid is that of revolutionary improvements in the cost-effectiveness of distributed computing systems. No requirement is fully meaningful until the costs and benefits of meeting it are well demonstrated and understood.

All these requirements express a heartfelt wish of users and providers. Meeting these requirements would enable computational Grids to deliver dependable and sustained performance and thus would save resources and lessen frustration. However, this is acknowledged to be an unsolvable problem, almost certainly requiring detailed understanding of the tradeoffs in terms of functionality and performance of fabric elements and Grid middleware.

1.3 Deployment, Operation and Troubleshooting

Tom Roney (NCSA), Anzar Afaq (CMS, Fermilab), Surendra Reddy (Oracle), Andy Hanushevsky (BaBar, SLAC)

From this session we learned:

1. Deployment and configuration are effort intensive and robustness and stability are hard to achieve.
2. The current situation results in misleading errors and faults that require a combined team of experts from application and middleware groups working together to solve.
3. Information is available in many different places, in many different formats. There are insufficient tools available for synthesis, archiving, short term and trend analysis.
4. Unless there is tight configuration control on the complete system, scaling in any direction of Grid application deployment induces new problems and an unpredictable increase in operations effort.
5. Deploying legacy applications is fragile. Additional errors occur that are more difficult to diagnose because of the lack of end-to-end monitoring instrumentation and designs taking account of the widely distributed nature of the execution environment.

Discussion and presentations brought forward the following needs:

6. Any information collected needs to be selectable and archivable, with dynamic control of level and length of history to be kept, based on the operating characteristics of the running system.
7. Standard error message formats and interpretation from the middleware would aid in interpretation and response.
8. Care must be taken to balance the impact of monitoring and fault response systems on the overall performance. Allowance must be made for errors and faults in the monitoring

and error reporting systems themselves. Requirements must be specified for expectation in event delivery and quality.

9. Need improved tools and techniques for deployment and configuration control. Configuration management, archiving and versioning of all deployed software is required to allow investigation of changes of performance over time. The length of history needed can be long (3 years for oracle). Support for both complete rebuilds from source code and distribution of binaries and executables over the distributed system is required.
10. The widely distributed nature of the Grid systems brings issues and requirements of its own: Remote management of failures must be well supported required. Handoff and collaboration between operations groups and sites is necessary – especially as these cross national and organizational boundaries. Time-zone and operational mind-set differences must be accommodated.
11. Need to abstract and trace what has been happening end to end. Need comprehensive state tracking, tracing, error output retrieval and interpretation tools. There were no identified needs for replay of application operations. This should be further considered.
12. New challenges for the security infrastructure will result from the existence of comprehensive monitoring and performance archives. The grid needs access control services giving equivalent functionality to that of process groups in unix. One needs a way to associate all derived processes and activities across the distributed system.
13. System operators, application deployers and users want a single user interface to all the available information with intuitive interfaces and capabilities for drill downs and references to all levels and details of the information available.
14. HENP does not currently use rule based or AI systems to interpret or automate response to errors. This might be a fruitful area to reconsider, but careful consideration of the tradeoffs in performance need to be included.

1.4 Propagation, Logging, Interpretation and Response

Warren Smith (NASA Ames), Igor Terekhov (D0 FNAL), Doug Thain (Condor), Brian Tierney (LBL)

Issues and requirements in the domain of this panel session fall naturally into six categories:

- definition of events, log messages and errors
- transport, delivery, subscription and filter mechanisms
- logging and archiving of messages and errors
- analysis and presentation tools
- active response and actions to handle error conditions
- security, access control, authorization

There are existing implementations of systems in that provide useful examples if not directly usable software in the grid domain. Some study of existing implementations is certainly a necessary exercise but delineation and analysis of these systems is not within the scope of this panel. In the context of GGF there is also work in the architectural domain helpful for designing implementations, such as GMA, or naming schemes as developed in the damed working group.

We include a bulletized list of notes from this session to show the scope of the discussion and indicate requirements:

1.4.1 Definition of events, log messages and errors

need extensible set of attributes

need to associate user or workflow manager job identifiers

interactions with users, developers, operators

will need naming standards

is QOS needed here?

amendable to defining logging levels

1.4.2 Transport, delivery, subscription and filter mechanisms

need to propagate messages from source to all consumers

support for logging levels

need for filters for routing & suppression (different from log levels?)

probably need subscription mechanism for consumers

subscription implies registry

are there performance issues?

it is useful to watch streams of events, certain classes or categories (perhaps defined with filters)

quality of service or guaranty on delivery issues, it is necessary for a consumer to know if they get all events or may miss some, consumers and/or producers may need to specify a level of delivery guaranty

it is necessary to have access control mechanism integrated with GSI, probably issues about who managers access control & how (producers, users, facility owners, ...)

1.4.3 Logging and archiving of messages and errors

it is necessary to be able to save events, messages, errors to an archive

it is necessary to be able to query the archive

archive may need historical filtering (all messages < 1 day old, only certain classes < 1 week, summaries > 1 month, ...)

access control on archive, what about access control to individual or classes of events in the archive?

1.4.4 Analysis and presentation tools

it is necessary to be able to view events and summaries of events, either live or from an archive

it is necessary to analyze messages from an archive for post-mortem diagnostics of faults

necessary to analyze messages for diagnostics of live systems

need to analyze messages for defining automated error recovery actions

1.4.5 Active response and actions to handle error conditions

It was generally thought useful to be able to automatically handle some errors within a given domain of the system. For example, if it is clear that a job crashed on a given node because of some fault on the node, automatically resubmitting it to another node is likely a useful feature. This can be thought of a similar to a RAID filesystem where certain types of disk errors are handled transparently to the user though there may be an alarm to an operator for preventive maintenance.

1.4.6 Security, access control, authorization

It is clear that all aspects of inspecting, subscribing, viewing summaries of messages and errors will need access control mechanisms that are integrated with the rest of the grid security infrastructure. However, it does not appear that an “extra special” mechanisms are required.

1.5 System Instrumentation, Probing, Performance Problem Solving

Kaushik De (Atlas, University of Texas Arlington), Keshav Pingali (Cornell University), Don Petravick (Fermilab), Jenny Schopf (Globus, Argonne National Lab). Additionally: Les Cotrell (SLAC)

Issues from this panel session covered:

- Spanning of information from short to long term as a mechanism for performance problem identification.
- Visualization of monitoring and performance for human pattern matching and problem notice.
- New methods for fault tolerance and application design to aid in stability and performance.
- Development of monitoring software must be informed by experience, and is disruptive and expensive. Software planning must allow for this.
- Ensembles and typical statistical information
- Application responsibilities and Expectations
- Attributes to describe normal behavior across the layers.

Below are notes from the discussion and presentations in this session.

1.5.1 Monitoring and Presentation

- Black box that appears very simple to users, like an Ethernet router, that presents a very rich monitoring interface to people that need to debug information. It is important to be able to visualize this information so that operators can figure out what is wrong.
- Monitoring interfaces are important to developing architectural and administrative talent for understanding and debugging.
- Adding monitoring can be disruptive to the system doing the monitoring and the system being monitored, and may require changes to design of software. For example, information may be hidden but need to be exposed, one might need to have real-time data flows, and the data may need to conform to standard schema.
- Visualizations of data help people understanding monitoring results.
- Debugging problems often requires cooperation between experts in different domains.
- Some clever automatic analysis can be very useful. We have a lot to learn, and need to attack the problems in different systematic ways: analyzing small cases, getting people together to analyze real problems as they happen, and continuing to develop the abilities of our monitoring systems
- Monitoring tools must be easy to deploy to be ubiquitous. Attention must be paid to usability - e.g. for inter-regional and poor links, single compute nodes and large farms, developing countries

1.5.2 Problem Solving

- Performance can be obtained through manual means and intervention: resubmit failed jobs; Manually check log files to see what happened
- Repositories of failures and performance can provide valuable information for mining and analysis. Flat files and databases can equally be useful
- How do we know there is a problem with a system? This is a hard question, and monitoring doesn't answer the question. For instance, how long should a file transfer take? It is hard to give a single answer. Transferring a 1G file could take 90 seconds or 15 minutes, depending on software and hardware configurations, and it would be acceptable, but 15 minutes might be unacceptable for some configurations. It is very hard to figure out automatically when there is a problem.
- Globus can figure out when a problem exists in some cases, but the general case is very hard, particularly when you ask "why is the performance the way it is?"

- MDS is being extended to have more service and testing data. This sort of data will be much more available in the third version of the Globus Toolkit.

1.5.3 Fault Tolerance

Mechanisms for fault tolerance include:

- file redundancy
- independent verification processes
- 80% of code is error handling. Think about the implications of this.
- Self optimization through monitoring and capture of parameterized performance.
- Application optimized checkpointing.

1.6 Necessary Areas of Research and Development

We summarise the results of the workshop into three broad areas that we believe would most benefit the building and operation of effective computational Grids. Many of the Grid projects already address some corners of the problem space (e.g. GMA, Netlogger, instrumentation of GridFTP, Hawkeye.etc.). Developing and implementing a comprehensive solution to the challenges identified by the workshop requires a large effort that at this point in time will be disruptive to Grid middleware and application developers alike and will offer timely solutions to existing and soon to be deployed Grids. However, our view is that there needs to be significant increase in the effort devoted by application and middleware developers to implementing localized solutions.

1) Instrumentation and Analysis Infrastructure with Local through Global Scope

Research and development of tools that support the generation, collection, archiving, analysis and presentation of behavioral information about all Grid layers – application, middleware and fabric. The design of these tools must accommodate the complexities inherent to the loosely coupled, error-prone, evolving and multi-dimensional heterogeneity of a Grid environment. They must accommodate the overwhelming volume and potential lack of completeness and accuracy of the information that must be managed and understood, without compromising the performance of the applications and robustness of the grid itself. The system must provide for:

1. Flexible, extensible and easy to manage instrumentation, collection and storage of behavioral information throughout all software layers and hardware components.
2. Equivalent and matched local and global recording and analysis capabilities to allow local community operation and support and complete functionality and interfaces for system wide and/or cooperative management and tracking.
3. Real time characteristics (ie responsiveness to short term changes to match the performance needs of the system) to provide both expected as well as encountered performance data. Behavioral information over a wide range of time periods from the very short to the historical to allow detection of transitional behavior at all time scales.
4. Controlled ability for Grid activities (.eg. a job, a transaction, a service request) to be tracable and identifiable in all stages of its propagation and execution.

5. Fine and course grained control of the volume, nature and lifetime of the information, at all stages of its lifecycle – generation, collection, storage, presentation and analysis. The needs to be easy and ubiquitous mechanisms to match the characteristics and configuration of the information and functionality to the goals of the monitoring goals. It will be only too easy to generate and record vast quantities of information for which there are insufficient resources - programmatic and human – to synthesize and interpret.
6. Infrastructure for the storage of, management, containment, storage (temporary, durable and long term), mining and analysis of the suite of distributed, structured, heterogenous log files and/or databases. This should allow tuning of the cost-benefit and risk ratios from the additional load and complexities introduced by monitoring activities.
7. Means to ensure that the information is uniquely defined, machine decipherable and understandable. Examples would include a hierarchy of uniquely specific job identifiers throughout the distributed system; universal timestamps etc.
- 8.
9. Engagement of application and infrastructure architects in engineering the necessary software and system instrumentation capabilities.
10. Curation of long term monitoring and information repositories for long term trend analysis, prediction and archeology.

2) Consistent and Interpretable System and Component Fault and Error Reporting

This activity area should aim towards standards based fault, error and performance information definition, encoding, transport, and interpretation. It should specify the details of an infrastructure for the transport, aggregation, filtering and dissemination of fault and error information. It should promote and provide mechanisms for ubiquitous adoption of the error and fault reporting, interpretation and presentation.

Following issues should be addressed:

1. Universal naming, interfaces, principals and protocols for error and fault information and codes. This should include standards efforts through such organizations as the Global Grid Forum and W3C.
2. Definition and design of error and fault reporting and handling, synthesis and response mechanisms through all layers of the middleware and application infrastructure. Provision for automated rule, algorithm and human based response and reasoning mechanisms at the component and the system level.
3. Standards for and development of a Grid event service, including the delivery, the interface semantics, and the underlying transport. This should specifically include attention to the expectations of reliability, completeness and fault tolerance of the fault handling and management systems; including allowance for inconsistent state reporting, system guarantees, potential incompleteness of the information, latencies in transport and provision for archeology of the historical record.
4. Definition of aggregation and filtering semantics for reduction and synthesis of information to eliminate redundancy, provide for summary, interpretation and reasoning.

3) Agreement on policy of response to errors

Research is needed into the required operational support and response infrastructures – both holistic and technical. The global Grid infrastructure will span the barriers of technology and human boundaries and encompass the breadth of national and regional capabilities. We might eventually go to the extent of infrastructures such as for telephony and network systems to accommodate cooperation, policies, procedures and supporting frameworks necessary to sustain and deliver cost effective, or even ultimately affordable, computational benefits to the consumers and users. Research is needed both in broad technology and enterprise areas to :

1. Understand the sociological and organizational constraints and realities of operating integrated and transparent computational infrastructures through national and international locations.
2. Analyse the legal, procedural, and political ramifications of a global cyber infrastructure.
3. Negotiate, define and standardize on policies and procedures for operation, support, and incident response at the geo-political scale.
4. Provide pervasive collaborative environments for distributed, independent investigatory teams faced with the full range of hard intermittent, global and/or localized faults and performance anomalies. Often a full complement of experts from every layer of software and hardware is required to work together to identify and solve peripherally coupled but ultimately catastrophic sequences from cause to effect.
5. Develop strategies - both human and automated - to respond to, mitigate and repair fault and failure conditions.
6. Apply automated reasoning and response technologies to grid operations, support and response requirements.

4) Modelling of the System Performance and Response Characteristics

1. Develop models of the distributed system and through the injection of faults and anomalies explore the system and component behavior and performance profiles
2. Through these models develop metrics for the performance and response behaviors.
3. Develop automated comparison and validation tools between the system models and operationing infrastructures.

1.7 Background

1. Troubleshooting and Fault Tolerance in Grid Environments Workshop agenda and attendees
<http://www.ppdg.net/mtgs/Troubleshooting/agenda.htm>

2. Pre-workshop white paper on Troubleshooting Production Grids
<http://www.ppdg.net/mtgs/Troubleshooting/TroubleshootingWhitePaper-rev1.pdf>

1.8 Appendix: Participants