

Benchmarks for Grid Computing: A Review of Ongoing Efforts and Future Directions

Allan Snaveley, Greg Chun, Henri Casanova
University of California, San Diego
9500 Gilman Dr.
La Jolla, CA 92093-0505, U.S.A.
(+1) 858 534 5158
{asnaveley,greg,casanova}@cs.ucsd.edu

Rob F. Van der Wijngaart, Michael A.
Frumkin
NASA Ames Research Center
T27A, NAS Division
Moffett Field, CA 94035-1000
{wijngaar,frumkin}@nas.nasa.gov

ABSTRACT

Grid architectures are collections of computational and data storage resources linked by communication channels for shared use. It is important to deploy measurement methods so that Grid applications and architectures can evolve guided by scientific principles. Engineering pursuits need agreed upon metrics—a common language for communicating results, so that alternative implementations can be compared quantitatively. Users of systems need performance parameters that describe system capabilities so that they can develop and tune their applications. Architects need examples of how users will exercise their system to improve the design. The Grid community is building systems such as the TeraGrid [1] and The Informational Power Grid [2] while applications that can fully benefit from such systems are also being developed. We conclude that the time to develop and deploy sets of Grid benchmarks is now. This article reviews fundamental principles, early efforts, and benefits of Grid benchmarks to the study and design of Grids.

Categories and Subject Descriptors

C.4 [Performance of Systems]: *Measurement techniques*. C.2.4 [Distributed Systems]: *Distributed applications*.

General Terms

Measurement, Performance, Design.

Keywords

Benchmarks. Grid Computing.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '00, Month 1-2, 2000, City, State.
Copyright 2000 ACM 1-58113-000-0/00/0000...\$5.00.

1. INTRODUCTION

Grid architectures can benefit substantially from standard means for measurement and for comparison of design alternatives. Provided here is an overview of Grid benchmarking activities in the Grid Benchmarking Research Group of the Global Grid Forum (GGF) [14]. The high-performance computing (HPC) community is building heterogeneous systems such as the TeraGrid [1], a Distributed Terascale Facility, and IPG, The Informational Power Grid [2], that link multiple, physically remote platforms via high-bandwidth channels. This leads to varying communication and computation speeds. These Grids are designed to solve scientific problems at unprecedented computational scales with complex, large, and distributed data inputs. Since these applications are being developed at the same time as the infrastructure, this is the appropriate time to begin defining and deploying Grid benchmarks. Such benchmarks should embody the anticipated requirements of applications. Although this may imply that the benchmarks themselves will have to evolve to capture changing requirements of developing applications, the alternative—that of building complex and expensive Grids now and later defining metrics to measure how applications and Grid infrastructure interact—is not reasonable. The Grid Benchmarking Research Group in GGF was recently formed with the goal to develop a set of benchmarks for design trade-off analysis, and to exchange information between infrastructure developers and Grid applications writers.

2. HISTORIC ROLE OF BENCHMARKS

Benchmarks for “traditional” microprocessors and HPC systems fall into at least two main categories:

1. Low-level “probes” for determining the rates at which a machine can perform fundamental operations. Examples of this class include MAPS [3], STREAM [4] the PALLAS [5] MPI benchmarks, and LINPACK [6].
2. Representative applications meant to capture the computational needs of a class of applications. In HPC, among the best known of these are the NAS Parallel Benchmarks (NPB) [7]. These broadly express the computational requirements of a class of computational fluid-dynamics problems. Some other influential suites of this kind are SPEC [8], ParkBench [9] and SPLASH [10].

Historically these benchmarks were used to assess all features of computer architecture. Similarly, both categories should be represented in a suite of Grid benchmarks.

3. GRID PROBES

Grid probes measure the rates at which computation can be carried out, and the rates at which data can be accessed as a function of access size and pattern through a deep and wide data-storage hierarchy. Traditional HPC probes such as STREAM and LINPACK primarily focused on ascertaining the issue rate of functional units and the costs of memory accesses and communications. But beyond hardware differences, a critical difference between traditional tightly coupled systems and distributed systems is the *software service layer*.

An expanded approach is needed for Grid computing; reporting on all overheads due to Grid middleware is necessary to fully assess the potential of Grid computing. A comprehensive software infrastructure is needed for Grid computing [11] as resources are shared among users and span several administrative domains. A common set of software services, including resource discovery, resource access, data management, and security, were identified [12], prototyped as part of the Globus project [13], and are being generalized and further specified as part of the Global Grid Forum activity [14]. The GrADS project [15,16] initiated an empirical assessment of certain software overheads associated with Grid information services [17,18]. It has been shown in that work and ongoing experiments that those overheads can be significant, especially for resource discovery and application start-up times. The recently proposed and already popular Open Grid Software Architecture (OGSA) standard advocates the use of Web services as an underlying technology for the Grid software infrastructure [19]. There is also interest in employing Grid information services based on relational database technology [20]. The SDSC Storage Resource Broker [27] employs a Metadata Catalog (MCAT) that supports full SQL query capability and sophisticated capabilities for schema extensibility. These or similar services will be used to deploy applications on Grid resources.

Therefore, there is a need to *quantify the overhead of using those services and to understand their impact on application performance*. Some of the components on which the GGF research group is initially focused is the Globus Metacomputing Directory Service (MDS-2), [21] Globus Resource Allocation Manager (GRAM) [22], MPICH-G [23,24] the Globus DataGrid [25] and the Globus Security Infrastructure (GSI) [26], and the SRB. Grid benchmarks should be able to characterize the performance implications of using various approaches for Grid information services. An effort is underway to develop a well-defined, concise, suite of probes for measuring basic service such as remote and local database access rates, overheads of Grid middleware (such as Globus [13], SRB [27], NWS [28]), as well as basic of networks, disks, and compute engines.

4. SYNTHETIC GRID APPLICATIONS

Representative Grid applications will be used to capture the resource needs of important classes of Grid applications. The identification of such applications for inclusion in a benchmark suite is somewhat complicated by the fact that Grid applications are mostly under development at this time. Nonetheless, there are clear trends in the type of applications most likely to benefit from

Grids in the near term. The following two sections provide overview descriptions of two Grid applications benchmark efforts that are ongoing under the auspices of GGF.

The benchmark efforts cover applications with variation in two important characteristics, namely *computation intensity* and *data intensity*. Computation intensity of a Grid application is defined as the amount of computational work per element of the data set(s) communicated between processes or read from a storage device. Data intensity is the reciprocal of computation intensity.

Useful metrics for both types of benchmark applications are *turnaround time* and *throughput*. Turnaround time is time between starting a job and obtaining the resulting data. Throughput is submission volume possible without affecting the turnaround time. While it would be desirable to capture both metrics for Grids, throughput poses special challenges. See discussion in Section 5.

Classification of Grid Applications: In analyzing the underlying algorithms and Grid usage scenarios of applications, it is useful to consider four broad classes delineated by their varying computational intensity as a function of their memory demands, data-locality, and inter-task communications requirements:

I: Loosely Coupled. Applications in this class are made up of “bags-of-tasks” with low memory requirements and small amounts of data for each task, and little communication required between tasks. Thus, they are compute intensive and are suitable for execution on wide-area clusters connected via low-bandwidth/high latency networks, as demonstrated by the [SETI@HOME](#) project [29]. Many examples of such applications arise in the fields of bioinformatics and molecular biology.

II: Pipelined. Applications in this class digest streaming and/or real-time data. These algorithms are often very memory and data intensive (requiring the memories of more than one machine), and have coarse-grained inter-task communication, while the constituent tasks are highly parallel. Their data and memory requirements are more challenging than CLASS I applications, while their task-communication demands are somewhat more challenging. Typical examples in this class are the real-time signal processing and subsequent storage of data captured from satellites, remote sensors including microscopes etc. as in BIRN, MADCAP and ROADNET applications. This class clearly belongs on Grids because the problems are by nature distributed; for example the data-acquisition devices may not be co-located with the compute engines or data storage silos.

III: Tightly Synchronized. Applications in this class have frequent inter-task synchronization. They may also have significant computation, and memory/data usage. Thus they *may* be data-intensive and have the challenges of CLASS II but they place additional demands on the communications infrastructure due to their frequent inter-task communication. Examples of applications in this class are climate, physics, and molecular models employing explicit iterative methods (stencil algorithms, binary cellular automata, etc.). CLASS III applications have traditionally been run on tightly-coupled HPC systems. They can certainly make use of Grid information services to locate the most appropriate HPC resource. Whether many in this class will evolve versions with sufficient latency tolerance to enable *truly distributed* versions—whereby one problem is solved across several Grid compute platforms exchanging messages—remains to be seen.

IV: Widely Distributed. Applications in this class search, update, and/or unify distributed databases. These typically have small compute, data, and memory requirements, but need to work seamlessly across the Grid environment to access databases that are variously owned and updated. These are “data-related” but (usually) not actually data-intensive. ?? See e-mail

While this classification does not capture every possible dimension of difference between Grid applications, and some applications fall in overlapping sets, it does usefully divide the space covered by most of the motivating applications. The two benchmarking efforts described below include synthetic applications from these four classes.

4.1 Computation Intensive Grid Benchmarks

A good candidate for a suite of computation intensive benchmarks is provided by the NAS Grid Benchmarks NGB³⁰. NGB tasks are defined in terms of *data flow graphs*, whose nodes and arcs represent computations and communications, respectively. An NGB measurement of Grid performance is a report on the execution trace, which may include durations of execution of each node and of transmission along each arc. Summation of such durations along a critical path in the instantiation of the NGB data flow graph on the Grid gives turnaround time, which must be reported.

NGB are *representative* of tasks typically executed on the Grid, and also specify well-defined, measurable quantities of work. This precludes, for example, any interactive or non-deterministic processes. A benchmark performance figure is meaningless if the results are wrong, so it is important that a reliable *verification test* be provided.

NGB also measure to some extent the data transfer capabilities of the communication network, particularly *latency* and *bandwidth*. Latencies are automatically included in the turnaround time if communicating tasks of the NGB are executed on a nontrivial subset of the Grid. In order to test bandwidth the benchmarks send sizeable data volumes. Suitable candidate applications for Grid computing are relatively coarse-grained, as latencies between geographically separated Grid platforms are often large. This characteristic is reflected in NGB.

NGB contain *little initialization data*. This is a consequence of the paper-and-pencil specification of NGB, which is described in just a few pages of typed text.

NGB neither measure nor require security and fault tolerance. Although these are crucial ingredients of a successful Grid, they are very hard to quantify. However, it is envisioned that they will become an informal part of NGB performance reporting. For example, turning security on/off may affect NGB turnaround time, and NGB failure rates can indicate Grid reliability. Similarly, while it is vital to know which resources were involved in a certain performance result, there is no way of formalizing their characterization at present. It is envisioned that resource usage will also become an informal part of NGB performance reporting, for studying the trade-off between turnaround time and consumed resources, and eventually for pricing Grid resources.

These properties of NGB were obtained through specific design. An NGB of a particular class (problem size) is specified by a data flow graph encapsulating NGB tasks (NPB problems) and communications between these tasks. The Report node of the

graph is endowed with a verification test to determine correctness of the computational result. This makes it easy to change parameters, organize the benchmarks into classes, and diversify the tasks in the future. The decision to use NPB problems in NGB, specifically BT, SP, LU, MG, and FT, is motivated as follows.

- The NPB problems are well-studied, well-understood, portable, and widely accepted as scientific benchmark codes.
- Solid verification procedures for NPB already exist.
- The NPB problems require no interaction, and no data files to start, in principle (but see next item).
- The NPB problems produce sizeable arrays representing solutions on discretization meshes. These can be used as input for any of the other problems, since each is based on structured discretization meshes covering the same physical domain. Hence, it is fairly straightforward to construct simple but plausible dependency graphs representing sets of interrelated tasks on the Grid, with significant data flows (solution arrays) between them.
- The granularity of the benchmark can easily be controlled by varying the number of iterations or time steps of each NPB problem.
- The NPB problems specify operations that can sensibly symbolize scientific computation (flow solvers: SP, BT, LU), post-processing (data smoother: MG), and visualization (spectral analysis: FT). Collections of such tasks are deemed suitable candidates for Grid computing.
- Good parallel versions of all NPB codes exist, which enables balancing the load of complicated Grid tasks by assigning different amounts of computational resources to different subtasks.

Below are shown the data flow graphs of two of the four families of NGB problems: Embarrassingly Distributed (ED), and Visualization Pipeline (VP). ED represents parameter studies, which constitute multiple independent runs of the same program (SP), but with different input parameters (CLASS I). VP represents chains of compound processes, like those encountered when visualizing flow solutions as the simulation progresses. These can be executed in pipeline fashion, where a post-processing (MG) and visualization step (FT) can be carried out while a new flow solution (BT) is being computed (CLASS II). Communications between tasks in the graphs are indicated by solid arrows. Dashed arrows signify control flow.

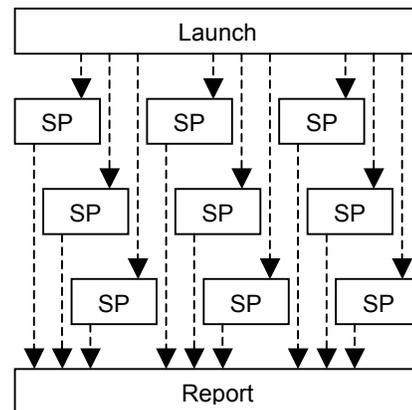


Figure 2. Data flow graph for Embarrassingly Distributed benchmark (ED)

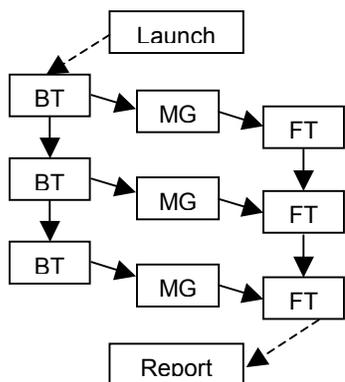


Figure 3. Data flow graph for Visualization Pipeline benchmark (VP)

4.2 Data-intensive Grid Benchmarks

Data intensive Grid applications are driven by two technology trends. Disks are getting denser and cheaper at a rate that exceeds Moore’s law for processors. By some estimates, these factors double every 9 months. But CPU speed is going up faster than disk access speed, so that applications that were formerly computationally intensive become data intensive if they perform significant disk accesses. The first trend has enabled scientists to store and process data in quantities that were unimaginable a few years ago. This has led to increased interest in acquiring and processing data on truly distributed systems that typically require the aggregation of physically separate data acquisition devices, data storage, computation and visualization resources. The second trend means that while the ability to store data grows exponentially, the ability to process it continues to challenge available Grid resources and infrastructure.

Some strategically important data-intensive applications come from:

- Biomedical Informatics Research Network (BIRN) [31] applications
- Genomic structure and interaction applications such as from EOL [32]
- GriPhyn [33] (Grid Physics Network) projects
- Cosmology applications (MADCAP) [34]
- Methods for modeling large molecular systems
- Coupled climate modeling applications
- Real-time observatories, applications, and data-management (ROADNet) [35]

These data-intensive applications fall mostly in CLASS II and CLASS III.

Sub-Classification of Data Intensive Applications: It is useful to consider subclasses of data intensive applications:

- Those with soft-real-time quality-of-service constraints (such as visualization applications)
- Those requiring access to central or distributed data repositories where the data-movement strategy (stage, prefetch, interleave with computation) may primarily determine performance
- Background processes that sweep massive amounts of data (such as satellite images) from disk to archival storage

The strategic applications above include representatives of each subclass. Thus a benchmark suite including representative calculations and data-usage scenarios from these applications should span an interesting and diverse set.

Figure 1 shows a problem from BIRN that includes components in CLASS II, CLASS III, and CLASS IV. The challenge is to acquire one image from a specified scanning electron microscope and another from a specified confocal microscope, compute an axial structure comparison using arbitrary compute resources, update a specified federated database (including the central archive), and then visualize the result of the computation on a continent remote from the central archive.

The ongoing development work is to synthesize some general data usage scenarios to capture the basic use-patterns of several applications across domain sciences. For each data-intensive application benchmark, there will be provided a “pencil and paper” version—with the problem specified but the implementation details left out, and also an example instantiation.

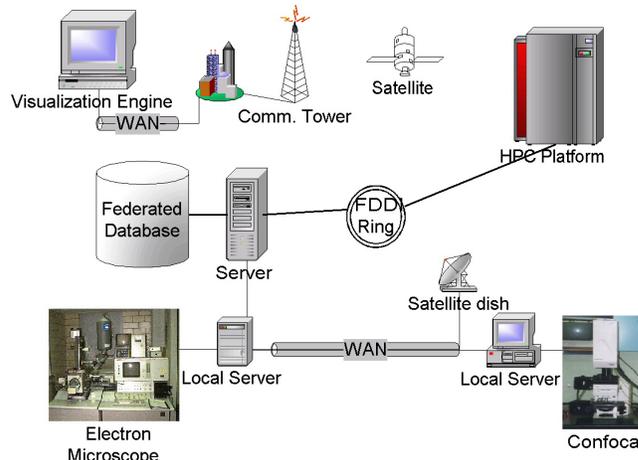


Figure 4. A “pencil and paper” configuration for a BIRN benchmark.

As a first step, a simple benchmark was created that assumes three nodes that may or may not be in separate administrative domains. The benchmark simulates Grid authentication, querying of resources, transfer of a dataset from node A to node B, computation on this dataset on node B, and transfer of the resultant output data to node C. Each of these steps can be timed. The benchmark allows the user to adjust parameters such as the size of the output data relative to the input data, as well as the amount of computation that is performed on the input data during the compute phase. This not only allows the user to quickly obtain

some rough estimates of grid performance, but it also provides the ability to vary the ratio of computation to communication. The reference implementation of the benchmark also serves effectively as a Grid validation script; for example, it requires that Grid information services be readily available and that DataGrid tools like GridFTP be configured and running properly. The next step is to extend this simple benchmark to include an arbitrary number of nodes, which will allow for an “at-a-glance” evaluation of interconnect speed and computing time for all nodes connected to a Grid. The plan is then to evolve this primitive benchmark into those that encompass the previously discussed benchmark classes.

5. OPEN QUESTIONS AND CONCLUSION

As mentioned above, two useful metrics for Grid applications are turnaround time and throughput. However, for benchmarking throughput, the Grid has to be stressed to the limit of one of its resources. This can be done, for example, by submitting a large number of instances of the same job with different initial data (parameter study). A throughput benchmark must be intrusive, consuming at least one of the Grid resources completely. While traditional HPC benchmarks are often intrusive (and may in fact be run on the whole machine), such “dedicated” testing of Grids that are comprised of geographically distributed assets under different administrative domains will be infrequent. Thus it is considered that intrusion will be an undesirable property for a Grid benchmark, one of whose important functions is to provide continual information on the health of the Grid environment.

With respect to turnaround time, a major challenge facing the Grid benchmarker is the issue of reproducibility. Grids are essentially batch systems. Generalizing from existing batch systems, it can be seen that to obtain reproducible results one must reserve all the resources needed to complete the application. This includes network bandwidth, disk space, CPU, instrument access, etc. If this has all been done and results are not reproducible, the Grid should be called defective. However, it is a real challenge in current Grid environments to measure resources consumed for a particular result (the result being the turnaround time). There is currently not even a standard way of describing resources, much less of aggregating them in a single cost figure, much less actually reserving portions of shared resource (as for example network bandwidth).

To further complicate matters, the Grid as a whole will always be changing due to the many hardware sites, administrative domains, and fluctuating workload with varying competition for resources. It is envisioned that the Grid Benchmarking Research Group work closely with other working groups in GGF to develop metrics and means for describing and reserving the resource needs of applications in order for benchmark results to make sense and be comparable.

It is desirable that the performance of a synthetic application benchmark be reported with additional information as to Grid state. For example, time spent in interaction with middleware will have to be separated from time spent in computation and communication. Particularly with respect to communication and data-base access, a measure of “network weather” and database-server load will have to be included with reported results in order for performance measurements taken at different times to be

comparable. Future plans include development of means to describe Grid state by parameters of network traffic, resource availability etc. and to enable correlation studies between the observed performance of repeated runs of an application/benchmark and the sensitivity of performance to variation in these parameters. Thus underway is an effort to standardize reporting methodology for Grid benchmarks to allow comparability of results when reproducibility is not possible.

The development of benchmarks for Grids is an important, ongoing activity and is in an early stage. Benchmarks for Grids will have some similarities and some important differences relative to traditional HPC benchmarks. This article has provided an overview of the issues and basic principles as well as an update on two coordinated Grid benchmarking activities in The Global Grid Forum. More information about the Grid Benchmarking Research Group in GGF is available at <http://www.nas.nasa.gov/GGF/Benchmarks/>.

6. ACKNOWLEDGMENTS

This work was supported in part by NSF STI award # 0230925.

REFERENCES

- [1] *TeraGrid*, <http://www.teragrid.org>
- [2] W. E. Johnston, D. Gannon, B. Nitzberg. “Grids as Production Computing Environments: The Engineering Aspects of NASA’s Information Power Grid.” Eighth IEEE International Symposium on High Performance Distributed Computing, Redondo Beach, CA, Aug. 1999
- [3] *MAPS*; <http://www.sdsc.edu/PMaC>
- [4] *STREAM*, <http://www.cs.virginia.edu/stream/>
- [5] *PALLAS*, <http://www.pallas.com>
- [6] *LINPACK*, <http://www.top500.org/lists/linpack.php>
- [7] D. Bailey et al. “The NAS Parallel Benchmarks.” NAS Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field, CA, 1994.
- [8] *SPEC*, <http://www.specbench.org/>
- [9] *PARKBENCH*, <http://www.netlib.org/parkbench/html/>
- [10] *SPLASH*, <http://www-flash.stanford.edu/apps/SPLASH/>
- [11] Foster, I. and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, San Francisco: Morgan Kaufmann Publishers, Inc., (1999).
- [12] Foster, I., C. Kesselman, and S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” *International Journal of High Performance Computing Applications*, Vol. 15, No. 3, 2001

- [13] *Globus*, <http://www.globus.org>.
- [14] *Grid Forum*, <http://www.Gridforum.org>
- [15] Dail, H. "A Modular Framework for Adaptive Scheduling in Grid Application Development Environments," *Masters. Thesis, University of California at San Diego* (2002), Available as UCSD Tech. Report CS2002-0698
- [16] Berman, F., A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski, "The GrADS Project: Software Support for High-Level Grid Application Development," *International Journal of High Performance Computing Applications*, Vol. 15, No. 4, (Winter 2001).
- [17] Petitet, A., S. Blackford, J. Dongarra, B. Ellis, G. Fagg, K. Roche, and S. Vadhiyar, "Numerical Libraries and the Grid: The GrADS Experiment with ScaLAPACK," *University of Tennessee Technical Report UT-CS-01-460*, (in proceedings *Supercomputing 2001*).
- [18] Song, Liu, Jakobsen, Bhagwan, Zhang, Taura, and Chien, "The MicroGrid: a Scientific Tool for Modeling Computational Grids," *Scientific Programming*, Vol. 8, No. 3, (2000), pp. 127-141.
- [19] Foster, I., Kesselman, C., Nick, J., Tuecke, S., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," [Provides a definition for implementing the Web services approach on the Grid], January, 2002.
- [20] P. Dinda, B. Plale, A Unified Relational Approach to Grid Information Services, Grid Forum Informational Draft GWD-GIS-012-1, 2002.
- [21] Czajkowski, C., S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," *10th IEEE Symposium on High-Performance Distributed Computing*, (2001), pp. 181-194.
- [22] Czajkowski, K., I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems," Proceedings of IPPS/SPDP'98 *Workshop on Job Scheduling Strategies for Parallel Processing*, (1998), pp. 62-82.
- [23] Foster, I., C. N. Karonis, "A Grid-enabled MPI: Message passing in heterogeneous distributed computing systems," *Proceedings of SC'98*.
- [24] *MPICH-G*, www.globus.org/mpi/
- [25] Allcock, W., A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, Vol. 23, (2001), pp. 187-200.
- [26] Foster, I., C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," *5th ACM Conference on Computer and Communications Security*, [Describes techniques for authentication in wide area computing environments], (1998), pp. 83-92.
- [27] *SRB*, <http://www.npaci.edu/DICE/SRB/>
- [28] *NWS*, <http://nws.npaci.edu/NWS/>
- [29] *SETI@home* Project, <http://setiathome.ssl.berkeley.edu>
- [30] M.A. Frumkin, R.F. Van der Wijngaart. "NAS Grid Benchmarks: A Tool for Grid Space Exploration." *Cluster Computing*, Vol. 5, No. 3, 2002
- [31] *BIRN*, <http://roadnet.ucsd.edu/>
- [32] *EOL*, <http://eol.sdsc.edu/>
- [33] *GriPhyn*, <http://www.griphyn.org/>
- [34] J. Borrill et al. MADCAP: The Microwave Anisotropy Dataset Computational Analysis Package, in "Proceedings of the 5th European SGI/Cray MPP Workshop"(1999)
- [35] *ROADNET*, <http://roadnet.ucsd.edu/>