Enabling Distributed Petascale Science

Andrew Baranovski[1], Shishir Bharathi[2], John Bresnahan[3,4], Ann Chervenak[2], Ian Foster[3,4,5], Dan Fraser[3,4], Tim Freeman[3,4], Dan Gunter[6], Keith Jackson[6], Kate Keahey[3,4], Carl Kesselman[2], David E. Konerding[6], Nick Leroy[7], Mike Link[3,4], Miron Livny[7], Neill Miller[3], Robert Miller[3], Gene Oleynik[1], Laura Pearlman[2], Jennifer M. Schopf[3,4], Robert Schuler[2], Brian Tierney[6]

[1] Fermi National Accelerator Laboratory
[2] Information Sciences Institute, University of Southern California
[3] Computation Institute, University of Chicago and Argonne National Laboratory
[4] Mathematics and Computer Science Division, Argonne National Laboratory
[5] Department of Computer Science, University of Chicago
[6] Lawrence Berkeley National Laboratory
[7] Department of Computer Science, University of Wisconsin, Madison

Contact Author: foster@mcs.anl.gov

Abstract. Petascale science is an end-to-end endeavour, involving not only the creation of massive datasets at supercomputers or experimental facilities, but the subsequent analysis of that data by a user community that may be distributed across many laboratories and universities. The new SciDAC Center for Enabling Distributed Petascale Science (CEDPS) is developing tools to support this end-to-end process. These tools include data placement services for the reliable, high-performance, secure, and policy-driven placement of data within a distributed science environment; tools and techniques for the construction, operation, and provisioning of scalable science services; and tools for the detection and diagnosis of failures in end-to-end data placement and distributed application hosting configurations. In each area, we build on a strong base of existing technology and have made useful progress in the first year of the project. For example, we have recently achieved order-of-magnitude improvements in transfer times (for lots of small files) and implemented asynchronous data staging capabilities; demonstrated dynamic deployment of complex application stacks for the STAR experiment; and designed and deployed end-to-end troubleshooting services. We look forward to working with SciDAC application and technology projects to realize the promise of petascale science. More details can be found at www.cedps.net.

1. Petascale science is an end-to-end problem

At a recent workshop on computational science, the chair noted in his introductory remarks that if the speed of airplanes had increased by the same factor as computers over the last 50 years, namely five orders of magnitude, then we would be able to cross the US in less than a second. This analogy communicates with great effectiveness the remarkable impact of continued exponential growth in computational performance, which along with comparable improvements in solution methods is arguably the foundation for the SciDAC program.

However, a participant was heard to exclaim following these remarks: "*yes—but it would still take two hours to get downtown*!" The serious point that this speaker was making is that science is an end-to-end problem and that accelerating just one single aspect of the problem solving process can inevitably achieve only limited returns in terms of increased scientific productivity.

These concerns become particularly important as we enter the era of petascale science, by which we mean science involving numerical simulations performed on supercomputers capable of a petaflop/sec or higher performance, and/or experimental apparatus—such as the Large Hadron Collider [4], DOE light sources and other Basic Energy Sciences user facilities [1], and ITER [3]—capable of producing petabytes of data. Successful science using such devices demands not only that we be able to construct and operate the simulation or experiment, but also that a distributed community of participants be able to access, analyze, and ultimately make sense of the resulting massive datasets. In the absence of appropriate solutions to the end-to-end problem, the utility of these unique apparatus can be severely compromised.

The following example illustrates issues that can arise in such contexts. A team at the University of Chicago recently used the FLASH3 code to perform the world's largest compressible homogeneous isotropic turbulence simulation [18]. Using 11 million CPU-hours on the LLNL BG/L computer over a period of a week, they produced a total of 154 terabytes of data, contained in 75 million files. Subsequently, they used GridFTP to move 23 terabytes of this data to computers at the University of Chicago; using four parallel streams, this took some three weeks at around 20 megabyte/sec. Next, they spent considerable time using local resources to analyze and visualize the data. In a final step, they are making this unique dataset available for use by the community of turbulence researchers. In each of these steps, they were ultimately successful—but they would be the first to argue that the effort required to achieve their end-to-end goals of scientific publications and publicly available datasets was excessive.

As this example illustrates, a complete solution to the end-to-end problem may require not only methods for parallel petascale simulation and high-performance parallel I/O (both handled by the FLASH3 code and associated parallel libraries), but also efficient and reliable methods for:

- high-performance *remote visualization*, to enable perusal of selected subsets and features of large datasets data prior to download;
- high-speed reliable *data placement*, to transfer data from its site of creation to other locations for subsequent analysis;
- terascale or faster *local data analysis*, to enable exploration of data that has been fetched locally;
- building and operating *scalable services*, so that many users can request analyses of data without having to download large subsets;
- *troubleshooting* the complex end-to-end system, which due to its myriad hardware and software components can fail in a wide range of often hard-to-diagnose ways;
- *securing* the end-to-end system, in a manner that prevents (and/or can detect) intrusions and other attacks, without preventing the high-performance data movement and collaborative access that is essential to petascale science; and
- *orchestrating* these various activities, so that they can be performed routinely and repeatedly.

Each of these requirements can be a significant challenge when working at the petascale. Thus, a new SciDAC Center for Enabling Technology, the Center for Enabling Distributed Petascale Science (CEDPS) was recently established to address three of these requirements: *data placement*, *scalable services*, and *troubleshooting*. This center is intended to support the work of any SciDAC program that involves the creation, movement, and/or analysis of large amounts of data.

2. Data placement

Large quantities of data must frequently be moved between computers in a petascale computing environment, whether because there are insufficient resources to perform analysis on the platform that generated the data, because analysis requires specialized resources or involves comparison with other data, or because the data must be "published" (moved and augmented with metadata) to facilitate use by the community. The requirement here is not simply high-performance data transfer, as supported for example by GridFTP [8]. For example, consider the following scenarios:

(1) A team running the CCSM climate simulation code at ORNL wants to publish its output data to the Earth System Grid (ESG) [11]. They must both transfer the output data to an HPSS archive at NERSC (perhaps while the model is running), and also register each file in a metadata catalog.

(2) Scientists who have run a combustion simulation at NERSC producing 100 TB of data want to explore that data using visualization tools. This task requires that data be replicated to five sites.

(3) Data produced by the CMS experiment at the LHC (at a sustained rate of 400 MB/s) must be delivered to a Tier 1 site in the US where it is further processed and then distributed among several US domestic and 20 non-US Tier-2 sites.

These scenarios, for which we can give many other examples across a wide range of DOE applications, frequently involve many of the following elements:

- data registration and metadata tagging as well as data movement;
- data movement over high-speed long-haul networks from a diversity of sources and sinks, including parallel file systems, running programs, and hierarchical storage;
- coordinated data movement across multiple sources, destinations, and intermediate locations, and among multiple users and applications;
- the use of techniques such as storage reservation, data replication, online monitoring, and operation retry to reduce the chances of failure, and/or to detect and recover from multiple failure modalities; and
- a need for predictability and coordinated scheduling in spite of variations in load and competing use of storage space, bandwidth to the storage system, and network bandwidth.

Not only must we be able to transfer data and manage end-point storage systems and resource managers; we must also be able to support the coordinated orchestration of data across many community resources. This requirement goes well beyond our current data transfer and storage resource management capabilities.

We are working to address this requirement in CEDPS by creating a new class of *data placement services* that can reliably position data appropriately across diverse systems and coordinate provisioning, movement, and registration across multiple storage systems to enable efficient and prioritized access by a community of users (see Figure 1). A single logical transfer may involve multiple sources and destinations necessitating the use of intermediate store and forward storage systems or the creation of optimized overlay networks such as user level multicast networks. Concurrent independent placement operations may be prioritized and monitored in case of failures.

In pursuing this goal, we build on much prior work, including our work on GridFTP [8] and associated data services [14]; the NeST [10] and SRM [31] storage management services; and a variety of research systems. We are extending the Data Replication Service (DRS), whose design was motivated by the Lightweight Data Replicator [17] that the LIGO experiment has been using to replicate one terabyte of data per day to eight sites in the US and Europe for over a year, creating 150 million file replicas to date.

Our data placement work addresses three classes of application requirements. First, we are interested in placing data at locations that will be advantageous for the execution of computations and workflows. By using a data placement service to perform staging operations asynchronously with respect to a workflow or execution engine, rather than explicitly staging data at run time, we hope to demonstrate improved application performance, as suggested in simulation [29].

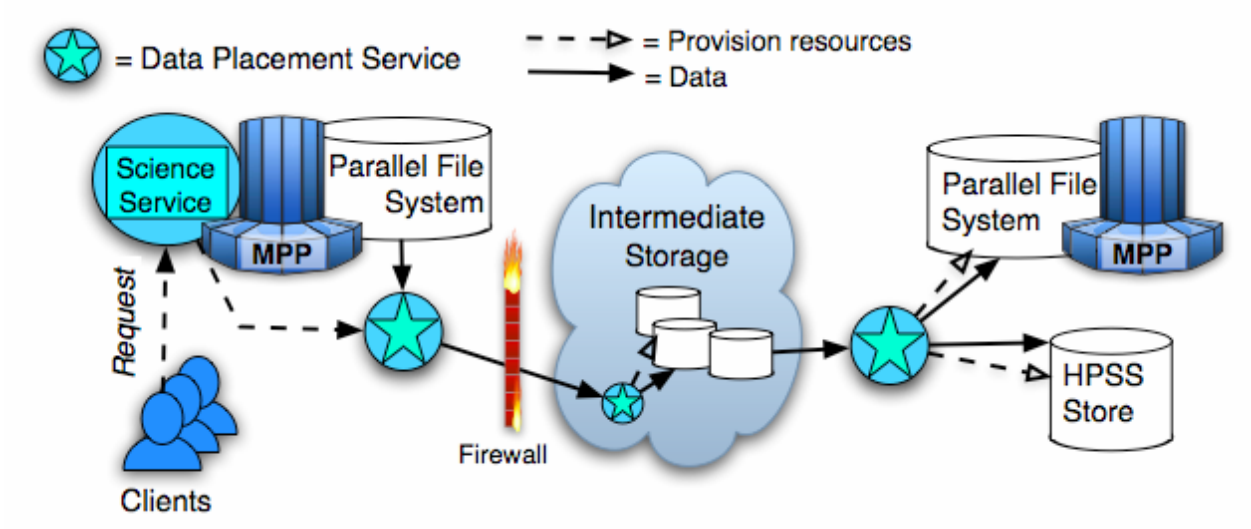

Figure 1: Elements involved in end-to-end data movement. Placement may require staging to intermediate storage. Users perform analysis by making requests to science services with access to placed data.

Second, we are interested in using the data placement service to assist with efficient staging out of data products. When an application runs on a compute resource such as a cluster or supercomputer, data products must often be staged off the storage system associated with that computational resource onto more permanent secondary or archival storage. These staging out operations can limit application performance, particularly if the compute resource is storage-limited; using an asynchronous data placement service to stage out data products should improve performance.

Finally, we are interested in data placement services that maintain required levels of redundancy in a distributed environment. For example, it might be the policy of the data placement service to enforce the requirement that there should always be three copies of every data item stored in the system. If the number of replicas of any data item falls below this threshold, the placement service is responsible for creating additional replicas to meet this requirement. We are investigating all three classes of data placement services, with an initial emphasis on the staging in and staging out scenarios [15].

In support of these scenarios, we have started work on a Managed Object Placement Service (MOPS), with the goal of transforming storage into a managed resource. MOPS allows users to negotiate access to a certain quantity of storage for a certain time and with certain performance characteristics. Its designa and implementation leverage GridFTP and NeST, among other technologies.

As work on MOPS proceeds, we also continue to implement improvements to the basic data management components on which MOPS depends. For example, recent work at Argonne incorporated pipelining in GridFTP server-to-server

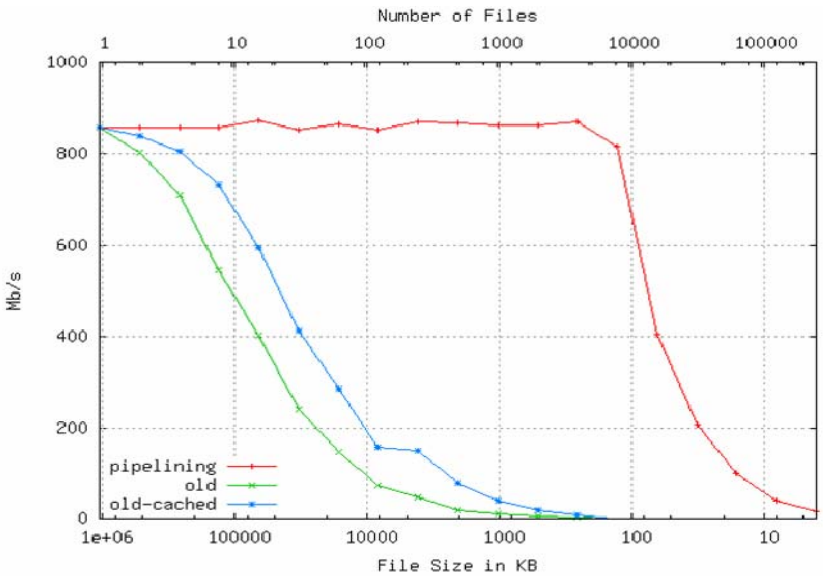

Figure 2: Pipelining optimizations in GridFTP: sending 1 GB partitioned into equi-sized files over 60 ms RTT, 1 Gbits WAN

transfers to achieve a significant increase in transfer performance for lots of small files (LOSF) [13], as illustrated in Figure 2. Previously, performance degraded rapidly for file size under 100 MB, but the current optimizations allow users to experience GridFTP's high performance transfer rates for any file size above 100 KB.

In another effort, staff at FermiLab are working to implement data transfer consistency verification features defined in the GridFTP-2 standard but not previously implemented in either Globus GridFTP or the storage software dCache [22], used on the Open Science Grid [5]. These features are important not only for verifying that individual transfers have completed correctly, but as a means of ensuring consistency in replicated data across many compliant yet independently maintained storage providers where consistency violations may occur because accidental or malicious data substitution as well as hardware failures.

These different efforts provide important building blocks that we will leverage in the coming year to create yet more ambitious end-to-end data placement solutions.

3. Server-side analysis

The role of the combined data placement service effort is to position data in physical locations that facilitates its subsequent analysis. Driven by the need to deliver analysis to an ever expanding non-expert user community, and a desire to incorporate analysis components as black boxes into complex analysis chains, we see increasing adoption of an approach in which programs are packaged as *scalable science services* that process (potentially many) requests to access and analyze data subsets via well-defined service interfaces [20].

Because science services define a community wide interface to a critical piece of data analysis, they must be able to process potentially many concurrent requests, as we illustrate in the following scenarios:

1. The fusion code TRANSP, deployed as a service by the fusion community [30], is accessed by tens to hundreds of remote users. Supporting such levels of access requires tools to manage the allocation of resources for TRANSP execution, so that all users get acceptable service.
2. The PUMA biology data resource must perform millions of BLAST, BLOCKS, and other computations to integrate new data into their database [32]. These tasks are memory- and CPU-intensive, and require many hundreds of CPUs for days.
3. ESG, which already supports downloads of terabytes of data per day by a community of 1000s, plans to provide server-side processing functions [11]. A single user request may require large amounts of computation and data access—and dozens of requests can be active at one time.

Such scenarios frequently involve the following elements.

- The need to integrate existing code into a services framework, to enable sharing of the code across community members, composition of analysis capability into end–to-end analysis chains, and the isolation of clients from details concerning the location and implementation of analysis functions.
- The ability to dynamically and reliably configure and deploy new analysis services and to dynamically vary the

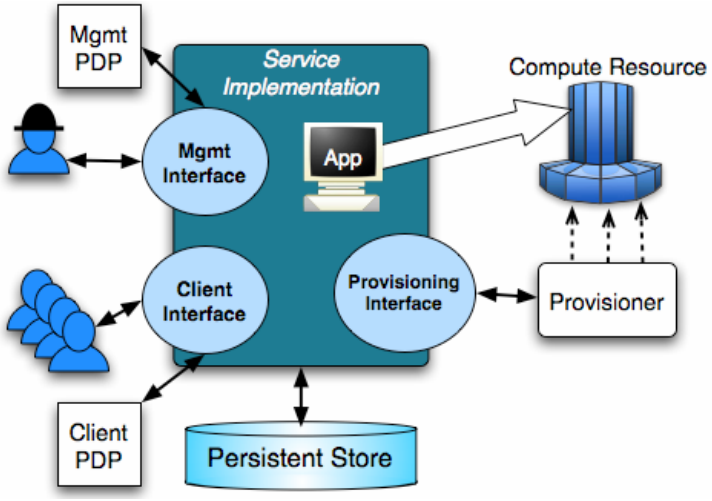

Figure 3: Architecture of a scalable science service.

computational resources used by analysis services, in response to changing community load.

- The ability to manage how services are used by communities on a user-by-user and request-by-request basis, and to monitor for performance degradation at that level.

As shown in Figure 3, a scalable science service that addresses these concerns will typically include a service implementation, end-user client and management interfaces, and mechanisms for the dynamic acquisition of computing, storage, and other resources in response to changing load.

Again, in developing tools to assist with the creation of such services we can build on considerable prior technology both within the CEDPS team and elsewhere. These include pyGlobus [25] and Introduce [24] for service authoring and the Workspace Service [26], Condor [27], MyCluster [34], and Falkon [28] for dynamic provisioning.

In one set of initial experiments, we have worked to enable dynamic provisioning for applications of the High Energy Nuclear Physics STAR experiment [6]. Our results illustrate both the challenges and potential of dynamic provisioning. Comprising nearly two million lines of code, developed over ten years by more than 100 scientists, and involving hundreds of dependencies, the STAR applications rely heavily on the right combination of compiler versions and available libraries to work. Even when the application compiles on a new platform, validating it is a controlled process subject to quality assurance and regression testing to ensure physics reproducibility and result uniformity across environments. These characteristics makes dynamic deployment of STAR applications onto new platforms challenging: even if an available resource can be found, adapting it to application needs is a complex and time-consuming process, making the investment cost-effective in practice only for dedicated platforms.

To overcome these challenges, we separate the two tasks of resource provisioning for STAR applications and managing application tasks. We use several technologies in conjunction to vary the set of platforms available to STAR scientists over time, as a function on demand. To experiment with this strategy, we used the Workspace Service to dynamically provision STAR-enabled resources. The Workspace Service uses the virtual machine technology (in this case, Xen [9]) to encapsulate a user environment (a "workspace"). A STAR environment, once configured and validated inside a virtual machine, can be deployed on many different platforms. In our experiment, the Workspace Service deployed virtual STAR application nodes on demand in response to requests from an authorized client (see Figure 4). On deployment, as part of the boot sequence, each virtual STAR node registered with a local resource manager (in this case, Condor). The newly provisioned virtual resources thus became available for local task deployment and management. In order to additionally enable remote clients to submit tasks to the newly provisioned nodes, the manager node was also configured with GRAM, allowing remote access to authorized Grid clients.

The scope of this experiment was limited to a small set of virtualization-enabled platforms. However, as more such platforms become available, we can expand the set of resources used, allowing full-scale production computations. We are working with the STAR community to run their applications on both

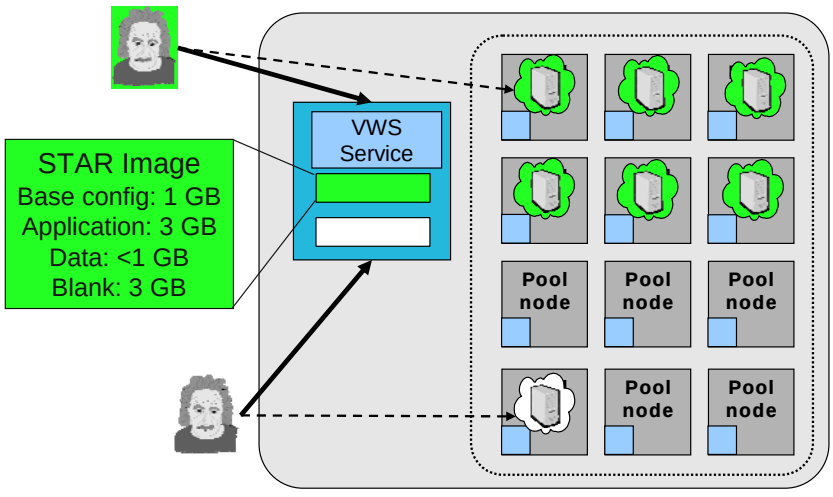

Figure 4: Dynamic provisioning using the workspace service. A user requests the workspace service to deploy STAR images and then interacts with those images to run STAR applications

commercially available resources (specifically, the Amazon EC2 service) and to enable similar modes of resource provisioning on Open Science Grid resources.

4. Troubleshooting

The end-to-end systems with which we are concerned comprise many different hardware and software components in different physical locations and administrative domains. Failures can occur and they can be hard to diagnose. Experience with current DOE distributed system deployments has shown that understanding behavior is a fundamental requirement, not just a desirable enhancement. For example, in 2003 about 30% of all grid jobs failed on Grid2003 (the precursor to Open Science Grid) for unknown reasons [21]. Middleware may also mask performance faults, when applications produce correct results but experience degradation in performance.

To address these challenges, we are developing methods and tools that allow us to monitor, collect, and respond to information about the individual and collective behavior of services in dynamic environments. These tools include log management services that can be associated with individual system and application components , a monitoring service to collect, archive, and analyze log and event data from those components, and a trigger service that performs warnings on errors.

As is the case with data placement, we have already made useful progress in several areas. We list three examples here. In the first, we have developed a first version of the log management service based on the syslog-ng system [7], as shown in Figure 5. We have used this service to detect performance anomalies [12], with NetLogger [33] used to access performance data.

A second activity has focused on improving the quality and consistency of available performance information. Specifically, we have codified a set of logging "Best Practices" [2], and are modifying the Globus Toolkit [19] to follow these practices. In defining these guidelines, we have worked with the European EGEE project to achieve compatibility with their security logging guidelines [23], an important requirement for LHC computing.

Third, we have extended the MDS4 Trigger Service to be more scalable and to allow better state management. This Globus component can be configured to check for specified failure conditions in monitored services and to notify system administrators when failures occur. It has been used by ESG for over three years for system failure notifications and to help diagnose errors [16]. We have re-architected this component to allow for additional trigger services, a separation of matching conditions and actions taken upon failure notification, and easier deployment through a Web interface.

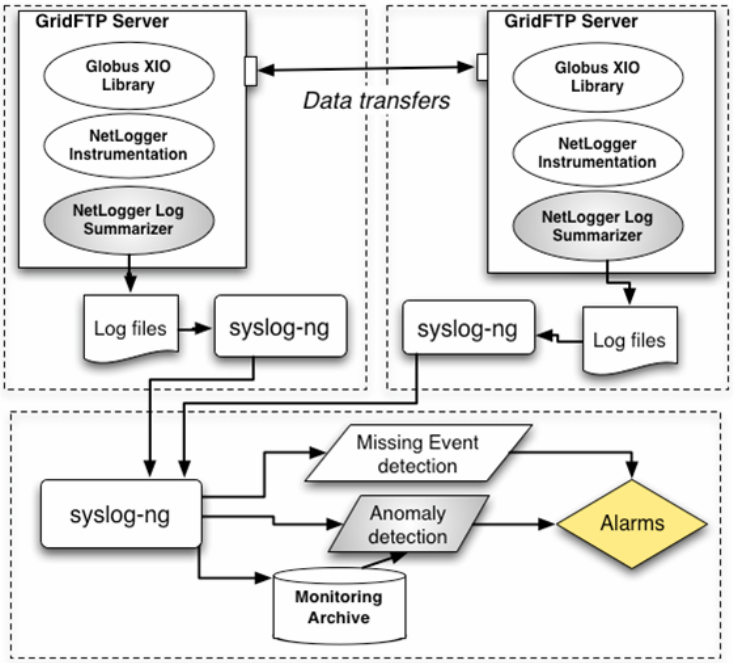

Figure 5: Log file collection across distributed sites.

These efforts represent useful steps towards our goal of end-to-end troubleshooting for data movement and service provisioning. Our next steps will be focused on deploying these tools more widely and gaining more extensive operational experience, as well as on incorporating additional functionality to help diagnose and prevent common errors seen on deployed systems.

5. Summary

We have introduced the SciDAC Center for Enabling Distributed Petascale Science, CEDPS, which is addressing three problems critical to enabling the distributed management and analysis of petascale datasets: data placement, scalable services, and troubleshooting.

In **data placement**, we are developing tools and techniques for reliable, high-performance, secure, and policy driven placement of data within a distributed science environment. We are constructing a managed object placement service (MOPS)—a significant enhancement to today's GridFTP—that allows for management of the space, bandwidth, connections, and other resources needed to transfer data to and/or from a storage system. Building on this base, we are developing end-to-end data placement services that implement different data distribution and replication behaviors.

In **scalable services**, we are developing tools and techniques for the *construction, operation, and provisioning of scalable science services*. We are creating *service construction tools* that take application code (simulation or data analysis, program or library) and wrap it as a remotely accessible service, with appropriate interfaces, authorization, persistence, provisioning, and other capabilities. We are also constructing *provisioning tools* for dynamic management of the computing, storage, and networking resources required to execute a service, and the configuration of those resources to meet application requirements.

Finally, in **troubleshooting**, we are developing tools for the *detection and diagnosis of failures* in end-to-end data placement and distributed application hosting configurations. We are constructing an end-to-end monitoring architecture that uses instrumented services to provide detailed data for both background collection and run-time event-driven collection. We are also constructing new monitoring analysis tools able to detect failures and performance anomalies and predict system behaviors using archived data and event logs.

In each area, we already have useful tools and are making good progress towards realizing project goals. We look forward to working with other SciDAC projects to apply and evaluate our tools.

Acknowledgements

This work is supported through the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, through the SciDAC program. Work at Argonne is supported under Contract DE-AC02-06CH11357; at FermiLab, under contract DE-AC02-07CH11359; and Lawrence Berkeley National Laboratory, under Contract DE-AC02-05CH11231.

References

1. BES Scientific User Facilities, http://www.sc.doe.gov/bes/BESfacilities.htm, 2007.
2. Grid Logging Best Practices Guide, CEDPS, http://www.cedps.net/wiki/images/6/6f/CEDPS-troubleshooting-bestPractices-16.doc, 2007.
3. ITER, www.iter.org, 2006.
4. LHC - The Large Hadron Collider Project, http://lhc.web.cern.ch, 2007.
5. Open Science Grid, http://www.opensciencegrid.org/, 2007.
6. The STAR Experiment, www.star.bnl.gov, 2007.
7. The syslog-ng Logging System, http://www.balabit.com/products/syslog-ng/, 2007.
8. Allcock, B., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I. and Foster, I. The Globus Striped GridFTP Framework and Server *SC'2005*, 2005.
9. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebar, R., Pratt, I. and Warfield, A. Xen and the Art of Virtualization *ACM Symposium on Operating Systems Principles*, 2003, 164-177.

10. Bent, J., Venkataramani, V., LeRoy, N., Roy, A., Stanley, J., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H. and Livny, M. NeST: A Grid Enabled Storage Appliance *Grid Resource Management: State of the Art and Future Trends*, 2004.
11. Bernholdt, D., Bharathi, S., Brown, D., Chanchio, K., Chen, M., Chervenak, A., Cinquini, L., Drach, B., Foster, I., Fox, P., Garcia, J., Kesselman, C., Markel, R., Middleton, D., Nefedova, V., Pouchard, L., Shoshani, A., Sim, A., Strand, G. and Williams, D. The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. *Proceedings of the IEEE, 93* (3). 485-495. 2005.
12. Bresnahan, J., Brown, A., Gunter, D., Schopf, J.M., Swany, M. and Tierney, B.L., Log Summarization and Anomaly Detection for Troubleshooting Distributed Systems in *IEEE/ACM International Conference on Grid Computing (Grid2007)*, (Austin, TX, 2007).
13. Bresnahan, J., Link, M., Kettimuthu, R., Fraser, D. and Foster, I. GridFTP Pipelining *TeraGrid Conference*, Madison, WI, 2007.
14. Chervenak, A., Deelman, E., Foster, I., Guy, L., Hoschek, W., Iamnitchi, A., Kesselman, C., Kunst, P., Ripenu, M., Schwartzkopf, B., Stockinger, H., Stockinger, K. and Tierney, B. Giggle: A Framework for Constructing Scalable Replica Location Services *SC'02: High Performance Networking and Computing*, http://www.globus.org/research/papers.html#giggle, 2002.
15. Chervenak, A., Deelman, E., Livny, M., Su, M.-H., Schuler, R., Bharathi, S., Mehta, G. and Vahi, K. Data Placement for Scientific Applications in Distributed Environments *8th IEEE/ACM International Conference on Grid Computing (Grid 2007)*, Austin, TX, 2007.
16. Chervenak, A., Schopf, J.M., Pearlman, L., Su, M.-H., Bharathi, S., Cinquini, L., D'Arcy, M., Miller, N. and Bernholdt, D. Monitoring the Earth System Grid with MDS4 *2nd IEEE Intl. Conference on e-Science and Grid Computing (e-Science 2006)*, Amsterdam, Netherlands, 2006.
17. Chervenak, A., Schuler, R., Kesselman, C., Koranda, S. and Moe, B., Wide Area Data Replication for Scientific Collaborations. in *6th IEEE/ACM Int'l Workshop on Grid Computing* (2005).
18. Fisher, R.T., Kadanoff, L., Lamb, D.Q., Constantin, P., Foster, I., Cattaneo, F., Papka, M.E., Dubey, A., Plewa, T., Rich, P., Antypas, K., Abarzhi, S.I., Asida, S.M., Calder, A.C., Reid, L.B., Sheeler, D., Gallagher, J.B., Glendenin, C.C. and Needham, S.G. Terascale Turbulence Computation on BG/L Using the FLASH3 Code. *IBM Systems Journal*. 2007.
19. Foster, I. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computational Science and Technology*, *21* (4). 523-530. 2006.
20. Foster, I. Service-Oriented Science. *Science*, *308*. 814-817. 2005.
21. Foster, I. and others. The Grid2003 Production Grid: Principles and Practice *IEEE International Symposium on High Performance Distributed Computing*, IEEE Computer Science Press, www.ivdgl.org/grid2003, 2004.
22. Fuhrmann, P., dCache, the commodity cache. in *12th NASA Goddard and Twenty First IEEE Conference on Mass Storage Systems and Technologies*, (2004).
23. Groep, D. *Middleware Security Audit Logging Guidelines* EGEE Document 2006-11-07, https://edms.cern.ch/document/793208, 2006.
24. Hastings, S.L., Oster, S., Langella, S., Ervin, D.W., Kurc, T.M. and Saltz, J.H. Introduce: An Open Source Toolkit for Rapid Development of Strongly Typed Grid Services. *Journal of Grid Computing*. 2007.
25. Jackson, K. pyGlobus: a Python Interface to the Globus Toolkit. *Concurrency and Computation: Practice and Experience*, *14* (13-15). 1075-1084. 2002.
26. Keahey, K., Foster, I., Freeman, T. and Zhang, X. Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. *Scientific Programming*, *13* (4). 265-275. 2005.
27. Litzkow, M.J., Livny, M. and Mutka, M.W. Condor - A Hunter of Idle Workstations. in *8th International Conference on Distributed Computing Systems*, 1988, 104-111.

28. Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I. and Wilde, M., Falkon: a Fast and Light-weight tasK executiON framework for Grid Environments. in *SC'2007*, (2007), ACM.
29. Ranganathan, K. and Foster, I. Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids. *Journal of Grid Computing, 1* (1). 2003.
30. Schissel, D.P., Keahey, K., Araki, T., Burruss, J.R., Feibush, E., Flanagan, S.M., Foster, I., Fredian, T.W., Greenwald, M.J., Klasky, S.A., Leggett, T., Li, K., McCune, D.C., Lane, P., Papka, M.E., Peng, Q., Randerson, L., Sanderson, A., Stillerman, J., Thompson, M.R. and Wallace, G., The National Fusion Collaboratory Project: Applying Grid Technology for Magnetic Fusion Research. in *Workshop on Case Studies on Grid Applications*, (2004).
31. Shoshani, A., Sim, A. and Gu, J. Storage Resource Managers: Essential Components for the Grid. in Nabrzyski, J., Schopf, J.M. and Weglarz, J. eds. *Grid Resource Management: State of the Art and Future Trends*, Kluwer Academic Publishers, 2003.
32. Sulakhe, D., Rodriguez, A., D'Souza, M., Wilde, M., Nefedova, V., Foster, I. and Maltsev, N. GNARE: An Environment for Grid-Based High-Throughput Genome Analysis. *Journal of Clinical Monitoring and Computing*. 2005.
33. Tierney, B. and Gunter, D. *NetLogger: A Toolkit for Distributed System Performance Tuning and Debugging*. Lawrence Berkeley National Laboratory, LBNL-51276, 2003.
34. Walker, E., Gardner, J.P., Litvin, V. and Turner, E.L. Creating personal adaptive clusters for managing scientific jobs in a distributed computing environment *Challenges of Large Applications in Distributed Environments*, IEEE, 2006.