

Fault Tolerance and Recovery of Scientific Workflows on Computational Grids

Gopi Kandaswamy, Anirban Mandal, and Daniel A. Reed

Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC - 27517, USA

Abstract—In this paper, we describe the design and implementation of two mechanisms for fault-tolerance and recovery for complex scientific workflows on computational grids. We present our algorithms for over-provisioning and migration, which are our primary strategies for fault-tolerance. We consider application performance models, resource reliability models, network latency and bandwidth and queue wait times for batch-queues on compute resources for determining the correct fault-tolerance strategy. Our goal is to balance reliability and performance in the presence of soft, real-time constraints like deadlines and expected success probabilities, and to do it in a way that is transparent to scientists. We have evaluated our strategies by developing a Fault-Tolerance and Recovery (FTR) service and deploying it as a part of the Linked Environments for Atmospheric Discovery (LEAD) production infrastructure. Results from real usage scenarios in LEAD show that the failure rate of individual steps in workflows decreases from about 30% to 5% by using our fault-tolerance strategies.

Index Terms—Computational grids, fault tolerance and recovery, resilient scientific workflows, scheduling.

I. INTRODUCTION

LARGE and complex scientific workflows rely on computational grids to satisfy their massive computational and data requirements. With increasing heterogeneity and complexity of computational grids, executing large scientific workflows reliably becomes a challenge. Although the mean time to failure of any entity in a computational grid is high, the large number of entities in a grid (hardware, network, software, grid middleware, core services etc) means that a grid will fail frequently. For example, in [1], the authors studied the failure data from several high performance computing systems operated by Los Alamos National Laboratory (LANL) over nine years. Although failure rates per processor varied from 0.1 to 3 failures per processor per year, systems with 4096 processors averaged as many as 3 failures per day. Thus, although the number of failures per processor is relatively low, the aggregate reliability of a system clearly deteriorates as the number of processors is increased. Since the failure rates are roughly proportional to the number of processors in the system, with over 112000 processors on a computational grid like the TeraGrid [2], [3], one would experience a failure

every two minutes. Unlike the LANL infrastructure, which is of high priority, very expensive and highly controlled with greater resources to maintain it, a computational grid is also susceptible to failures at the grid middleware level; software and services that tie the heterogeneous computing systems on a grid. Failure will be the norm rather than an exception. Hence workflow execution systems must be designed to execute workflows in a fault tolerant manner.

Current fault tolerance and recovery strategies like FT-MPI [4] can regularly take checkpoints of a workflow step, which is usually a parallel scientific application, and in case of a failure, restart the application from the last good checkpoint. However, many scientific workflows like the ones used in LEAD [5], [6] are deadline driven and must execute with a minimum success probability. Moreover, complex scientific workflows may assimilate large input data sets from a distributed set of data sources. Hence, moving the computation closer to the data sources could minimize the total time to completion of the workflow. Also, dynamic scientific workflows change their configuration rapidly and automatically in response to external events or decision-driven inputs from scientists and can initiate other workflows automatically. All these factors clearly necessitate new approaches to scheduling and fault tolerance for large and complex scientific workflows on computational grids, beyond what current technologies have to offer.

The main contributions of this paper are:

- An implementation of fault-tolerance and recovery strategies for workflows on computational grids
- An evaluation of the above strategies using the LEAD infrastructure for running dynamic and adaptive weather forecasting workflows
- Algorithms to find the degree of over-provisioning and the length of the migration path under the constraints of deadline and success probability
- Results from real usage examples in LEAD

The rest of this paper is organized as follows. In section II we give an overview of our fault tolerance and recovery service and discuss our two primary fault tolerance strategies viz. over-provisioning and migration. In section III we show results of our evaluation of the FTR service. In section IV we discuss some related work and in section V we present our future work and conclusions.

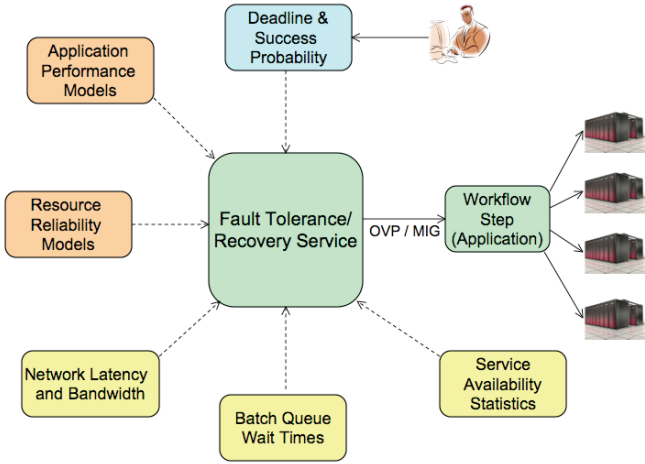


Fig. 1. Fault Tolerance and Recovery Service

II. FAULT TOLERANCE AND RECOVERY OF SCIENTIFIC WORKFLOWS

Large parallel computations, distributed data transfer and management and soft real-time constraints characterize complex scientific workflows. However, the computational grids on which these workflows may run, have distributed systems and software, virtual organizations and few guarantees of quality and availability of service. This necessitates the need for new approaches to scheduling complex scientific workflows on computational grids in order to meet the soft real-time constraints of deadline and reliability. Figure 1 shows our fault tolerance and recovery (FTR) service. It uses application performance models, network models, resource reliability models and batch-queue wait time predictions to decide what fault tolerance strategy viz. over-provisioning or migration to adopt for each workflow step/task in a workflow. Then the FTR service either runs copies of the workflow step on multiple resources (over-provisioning) or migrates the workflow step to another resource in case of a failure (migration) to meet the soft real-time constraints of deadline and reliability (success probability).

- **Over-provisioning:** Over-provisioning or replication is a fault-tolerance mechanism where multiple copies of an application (with the same input data-set) are executed in parallel. The idea is to maximize the probability of success for the application so that if one copy fails then another copy may succeed. This is particularly important in the presence of unreliable resources and where missing a deadline may incur a high penalty. The over-provisioning algorithm determines the best set of resources to replicate the application.
- **Migration:** In migration, an application is progressively restarted from the last good checkpoint (if available) on a different resource in case of failures. The migration algorithm

determines the best migration path.

The FTR service uses the following application and resources estimates and finds the best set of resources for over-provisioning or migration.

A. Application Performance Models

Application performance models are estimates of execution times of applications on different resources. Since grid resources are heterogeneous (in architecture, CPU speed and other system characteristics), the same application may have significantly different execution times on different resources. Hence, we take into account the performance models of the applications in the workflow to estimate the computational portion of the total expected completion time for the application.

B. Batch Queue Wait Times

Batch-queue systems like PBS and LSF, manage job prioritization and execution on most grid resources. An application may have to wait for a significant amount of time on a queue before it starts execution. Hence, an estimate of the batch-queue wait times is an essential component of the total expected completion time for an application. We use the methodology and software described in [7] for estimates of batch-queue wait times.

C. Network Latency and Bandwidth

Workflow execution may also involve moving large amounts of data between workflow steps, which takes significant amount of time. Hence, we also consider the data-transfer time while calculating the total expected completion time for an application. To estimate the data-transfer time, we use the Network Weather Service (NWS) [8] to obtain estimates of network latency and bandwidth values between resources pairs on the grid. We also find out the size of the data to be moved from the inputs to the application. Combining data-size with NWS estimates gives us an estimated data-transfer time.

D. Deadline and Success Probability

Apart from these, the fault-tolerance algorithms also take into account the specified deadline for the application. This is important for workflows that have real-time constraints.

Another input to the algorithm is the required success probability, which is the expected success probability from the scientist's perspective. It is the probability with which the scientist wants the workflow to run to completion. With application failures becoming more of a norm than exception on the grid, an expected success probability of 1 is not realistic. There is a trade-off between the value of required success probability and finding a fault-tolerance mechanism that can satisfy that value. A high value may result in the algorithms failing to find a resource set that can be used to run the application in a fault-tolerant way.

E. Resource Reliability Models

We use a simple model for modeling resource reliability. We assume that resource failures are independent over time,

implying that resource failures in the current time interval are

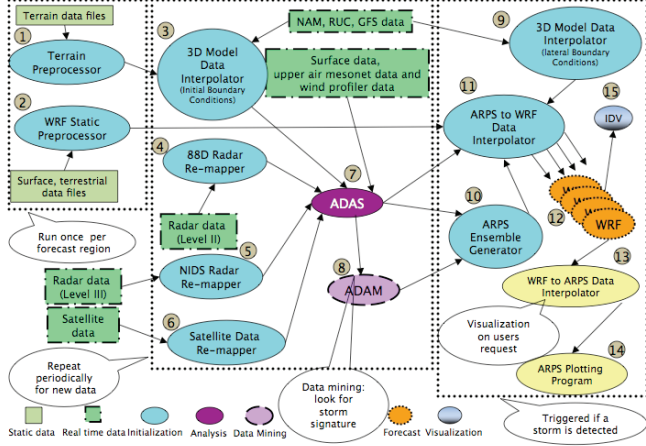


Fig. 2. A typical weather forecasting workflow in LEAD independent of the failures in the previous time interval. Hence, we can assume resource failures to follow a binomial distribution. To recap, binomial distribution is the discrete probability distribution of the number of successes in a sequence of n independent trials, each of which yields success with probability p . The probability of x successes in n trials is given by

$$\binom{n}{x} p^x (1-p)^{(n-x)}$$

In our case, if the failure probability of a resource is p for one time-unit and the expected execution time of an application on that resource is n time units, then the probability that the application succeeds on that resource is given by

$$\binom{n}{0} p^0 (1-p)^{(n-0)} = (1-p)^n$$

This is the probability of 0 successes in n trials. In our case, each trial asks the question whether the resource (and hence the application) failed in the time interval corresponding to the current trial. The application succeeds if the answer is no for all the n trials.

We understand that resource failures may be correlated over time, but we don't consider that in this work.

F. Service Availability Statistics

In computational grids a layer of software called the grid middleware ties the underlying heterogeneous hardware and software systems together. The grid middleware has numerous core grid services. These include security services like the MyProxy [9], execution and resource management services like Globus WS-GRAM [10], data management services like GridFTP [11] and RFT [12] and information services like MDS [13] [14]. These services are critical to many workflows running on computational grids. Reliability of these core grid services is paramount to the success of workflows. The FTR

service has plug-in modules to regularly monitor some core grid services like Globus WS-GRAM, GridFTP and MDS. The availability of these services is taken into consideration while scheduling workflow tasks on the grid. This further reduces the failure probability of workflows. Work is in progress by the Inca and BQP to monitor core grid services and provide estimates of their reliability. We plan to incorporate those in our future work.

G. Determining degree of Over-provisioning and Migration Path

Given these estimates, we developed an algorithm to find the degree and resources for over-provisioning and the

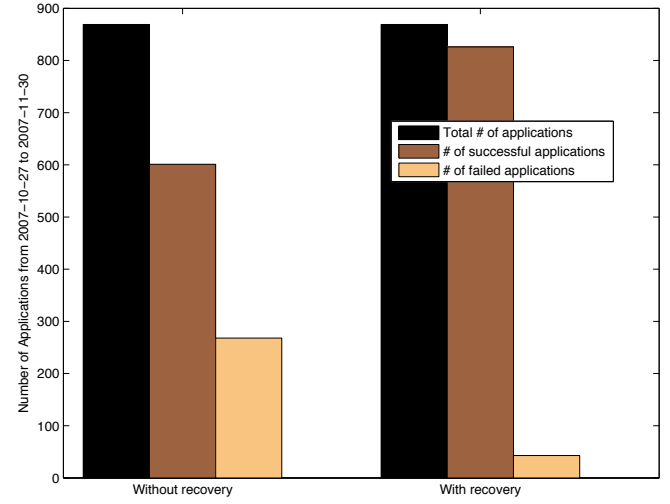


Fig. 3. Recovery statistics for applications in LEAD workflows running on the TeraGrid

migration path. Let the application deadline be d , the required success probability be x and $[1..M]$ be the set of available resources. We define h_{r_i} to be the expected total completion time for an application, which is obtained by aggregating performance models, batch-queue wait time estimates and data-transfer time estimates. We define m_{r_i} to be the probability of failure on resource r_i , which is obtained using the reliability model as described in sub-section E.

We solve the following optimization problem to find the best set of resources for over-provisioning. The problem is to find a partition, $P = \{r_1, r_2, \dots, r_n\}$ of $[1..M]$ such that the following holds true:

- $1 - m_{r_1} \times m_{r_2} \dots m_{r_n} \geq x$
- $|P|$ is minimum
- $\max(h_{r_1}, h_{r_2}, \dots, h_{r_n}) \leq d$

We use an efficient algorithm described in [15] to enumerate all subsets of $[1..M]$ and return the smallest subset of resources that satisfies these conditions.

To find the migration path, we solve a similar optimization problem with the last condition being substituted by

$\sum_{i=1}^n h_{r_i} \leq d$. In the absence of a deadline and success probability, we provide an ordering of resources with the best resource being the first resource on which the application should be executed.

III. EVALUATION

We have integrated the FTR service with the LEAD production infrastructure. All workflows run by scientists and students from the LEAD portal are automatically scheduled for fault tolerance and recovery on the TeraGrid by the FTR service.

In a period of about a month, a total of 165 complex weather forecasting workflows like the one shown in Figure 2, were run from the LEAD production portal, with a total of 869 applications (workflow steps) in them. Figure 3 shows that 268 of the total 869 applications failed on the TeraGrid without any fault tolerance and recovery strategies. These applications failed due to a variety of reasons like GridFTP failures during data transfer, file systems problems like NFS [23] and GPFS [24] errors, file systems running out of disk space due to very large data transfers, connection timeouts from Globus WS-GRAM, compute node crashes, transient downtimes of core grid services like MDS and RFT. None of the above application failures were due to problems with the applications themselves as they were all well tested applications in the LEAD production infrastructure. Out of these 268 failed applications, 225 applications were successfully recovered by the FTR service, thus bringing

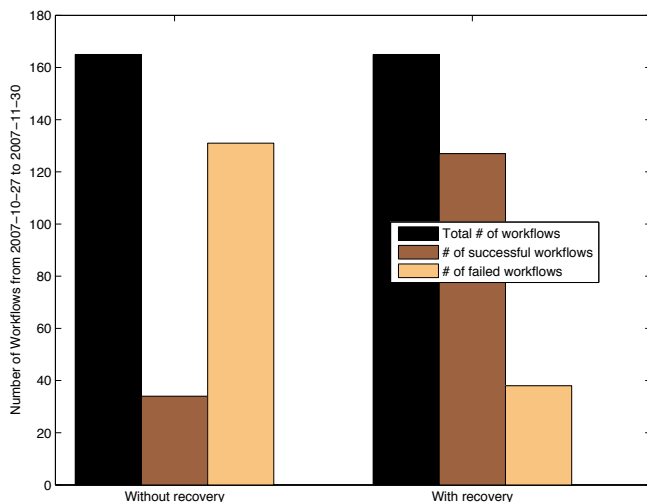


Fig. 4. Recovery statistics for LEAD workflows running on the TeraGrid

down the application failure rate from 30.84% to 4.95%. The average number of attempts to recover an application successfully was 2.395.

Figure 4 shows the number of workflows that failed because of the 268 failed applications above. A workflow consists of several workflow steps, where each step has usually one or more applications. Thus, even if one application in the workflow fails, the entire workflow fails. Although the FTR

service does scheduling, fault tolerance and recovery for each application in the workflow and not the entire workflow as a whole, it is still useful to see how the recovery strategies applied to the applications affect the workflow failure rates. We see from Figure 4 that of the 165 workflows, 131 failed on the TeraGrid. Of the 131 failed workflows, 93 were recovered because all the applications in those 93 workflows were successfully recovered by the FTR service. Thus the workflow failure rate was brought down from 79.39% to 23.03%. The reason why the workflow failure rates (79.39% and 23.03%) are greater than the application failure rates (30.84% and

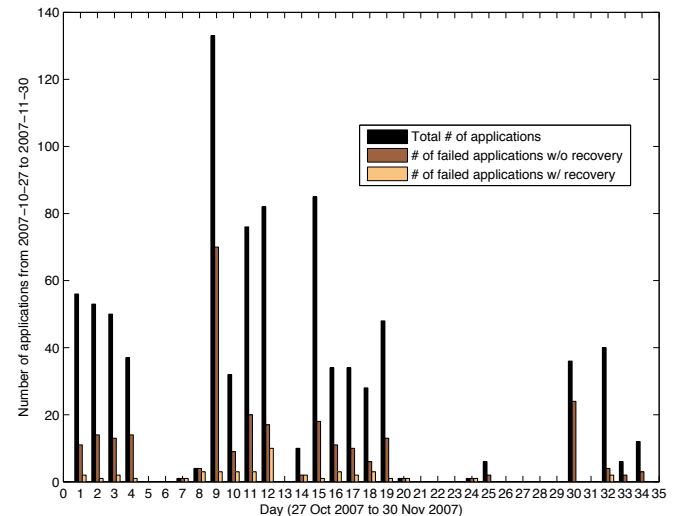


Fig. 5. Daily recovery statistics of applications in LEAD workflows running on the TeraGrid

4.95%) is because even if one application in workflow fails, the entire workflow fails, as the workflows are not inherently fault tolerant.

Figure 5 and Figure 6 show the daily failure rates of LEAD applications and workflows running on the TeraGrid from Oct 27, 2007 to Nov 30, 2007.

IV. RELATED WORK

In [1], the authors have analyzed failure data on 20 LANL systems collected over 9 years. They have categorized failure types and built statistical models for mean time between failures and mean time to repair. A similar study in [16] has developed statistical models for mean time between failures for individual nodes and system. These may form a basis for developing accurate reliability models for resources, which is essential for developing generic fault-tolerance mechanisms.

There are several techniques for fault-tolerance [17] of single parallel SPMD applications running on HPC systems, which include FT-MPI [4] and LA-MPI [18]. In [19], Reed et al. identify reliability challenges for large-scale HPC systems and provide adaptive techniques for failure detection (using performance contracts) and runtime adaptation for MPI applications. Lu et al. [20] discuss fault sensitivity for MPI applications using fault-injection. In [21], Weissman has proposed fault-tolerance techniques like wide-area replication and application level checkpointing for SPMD applications

based on performance models. The fault-tolerance methods discussed in this paper are applied at the workflow level and

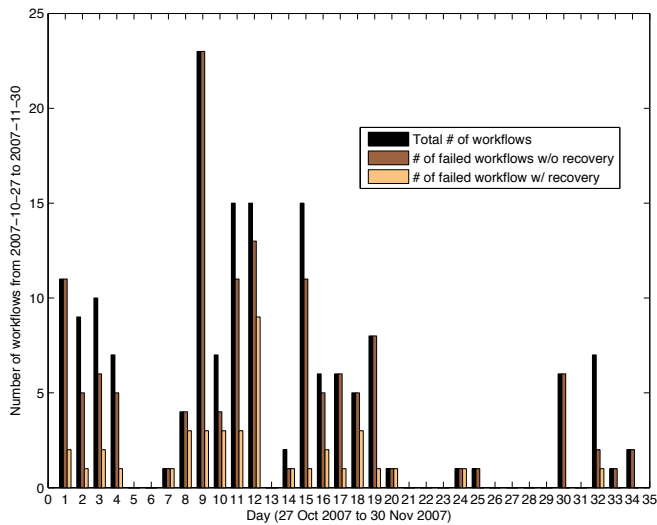


Fig. 6. Daily recovery statistics for LEAD workflows running on the TeraGrid

hence can leverage these techniques for fault-tolerance within a HPC site.

Khalili et al. [22] have done a study on reliability of computational grids using data gathered via a deployment of monitoring infrastructure on two production grid platforms, one of which is the TeraGrid. They have inferred that the success rates of application and benchmark runs on the grid range between 55% and 80%. This motivated the importance of including fault-tolerance mechanisms for workflows executing on computational grids.

One of the prevalent methods for fault-tolerance on computational grid-systems is simple retry, as in Condor-G [23] and Gridsolve [24]. The application is re-submitted on a resource in case of a failure. In Condor DAGMan [27], in case of a failure, the remaining portion of the workflow (the rescue DAG) is re-executed. The same is true for the Pegasus [28] workflow management framework. From a fault-tolerance perspective, our FTR framework is significantly better than the existing grid workflow frameworks because (a) it provides transparent mechanisms for application replication and migration and (b) the fault-tolerance decisions are based on several application and resource metrics, which balance reliability and application performance.

Hwang et al. [29] present a failure detection service (based on notifications) and a flexible framework for handling grid failures. However, their evaluation is simulation based.

Alonso et al. [30] present issues on fault-tolerance for commercial workflow management systems (WFMS). Current commercial WFMS lack in fault-tolerance features and they suggest using techniques borrowed from transactional systems like replication for increased availability and better exception handling.

Scheduling workflows on grids [31], [32], [33], [34], [35], [36], [37] is an active area of research. Most of these use

performance, cost or other QOS parameters as the optimization criteria. However, none of these use reliability as a criterion for resource selection. In our work, we balance reliability and performance for selecting the best resources for fault-tolerance.

In [38], the authors develop a strategy to select the most available and fastest resources. Jarvis et al. [39] identify issues in grid performability, the joint consideration of performance and dependability.

V. CONCLUSION

Owing to the complexity of computational grids, executing complex workflows reliably is a challenge. We have developed a fault tolerance and recovery (FTR) service that uses over-provisioning and migration for reliable execution of workflows on computational grids. It takes into consideration a) application performance models b) network latency and bandwidth c) batch-queue wait time d) user specified deadline and success probability of applications e) resource reliability models and f) availability of some core grid services, to determine the best fault tolerance strategy.

We have integrated and evaluated our FTR service with the LEAD production infrastructure. We found that in a time period of about a month in which a total of 869 applications were run on the TeraGrid from 165 weather forecasting workflows, the FTR service was able to reduce the rate of application failures due to numerous grid related problems, from 30.84% to 4.95%, with an average number of attempts to recover an application being about 2.395. This in turn reduced the failure rate of the workflows from 79.39% to 23.03%.

In future we plan to use reliability estimates of more core grid services to execute workflow tasks more reliably on grids. We also plan to extend our algorithms to other types of workflows like bio-informatics, which have different characteristics; a large number of small workflow tasks/applications unlike meteorological workflows, which have a small number of large workflow tasks/applications.

REFERENCES

- [1] B. Schroeder, and G. Gibson, "A Large Scale Study of Failures in High-performance-Computing Systems," *International Symposium on Dependable Systems and Networks*, 2006.
- [2] D. A. Reed, "Grids, the TeraGrid and beyond," *Computer*, vol. 36, no. 1, pp. 62- 68, Jan. 2003
- [3] C. Catlett, "The Philosophy of TeraGrid: Building an Open, Extensible, Distributed TeraScale Facility," *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, 2002
- [4] R. Wolski, N. Pjesivac-Grbovic, K. London, and J. Dongarra, "Extending the MPI Specification for Process Fault Tolerance on High Performance Computing Systems," *International Supercomputing Conference (ICS)*, Heidelberg Germany, Jun. 2004
- [5] K. K. Droege-meier, D. Gannon, D. Reed, B. Plale, J. Alameda, T. Baltzer, K. Brewster, R. Clark, B. Domenico, S. Graves, E. Joseph, D. Murray, R. Ramachandran, M. Ramamurthy, L. Ramakrishnan, J. A. Rushing, D. Weber, R. Wilhelmson, A. Wilson, M. Xue, and S. Yalda, "Service-Oriented Environments for Dynamically Interacting with Mesoscale Weather," *IEEE Computational Science and Engineering*, vol. 7, no. 6, pp. 12-29, Dec. 2005
- [6] B. Plale, D. Gannon, J. Brotzge, K. Droege-meier, J. Kurose, D. Mclaughlin, R. Wilhelmson, S. Graves, M. Ramamurthy, R. D. Clark, S.

- Yalda, D.A. Reed, E. Joseph, and V. Chandrasekar, "CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting," *IEEE Computer*, vol. 39, no. 11, pp. 56-64, Nov. 2006
- [7] J. Brevik, D. Nurmi, and R. Wolski, "Predicting Bounds on Queuing Delay for Batch-scheduled Parallel Machines," *Proceedings of the Eleventh ACM SIGPLAN Symposium on Principle and Practice of Parallel Programming*, 2006
- [8] G. E. Fagg, E. Gabriel, G. Bosilca, T. Spring, Z. Angskun, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *Future Generation Computer Systems*, vol. 15(5-6), pp. 757-768, 1999
- [9] J. Novotny, S. Tuecke, and V. Welch, "An Online Credential Repository for the Grid: MyProxy," 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01), pp. 1-4, 2001
- [10] M. Feller, I. Foster, and S. Martin, "GT4 GRAM: A Functionality and Performance Study," *TeraGrid 2007 Conference*, 2007
- [11] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus Striped GridFTP Framework and Server," *Proceedings of Super Computing 2005 (SC05)*, November 2005
- [12] R. K. Madduri, C. S. Hood, and W. E. Allcock, "Reliable file transfer in Grid environments," *27th Annual IEEE Conference on Local Computer Networks*, pp. 737-73, Nov. 2002
- [13] S. Fitzgerald, I. Foster, C. Kesselman, G. Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," *IEEE Symposium on High Performance Distributed Computing*, pp. 365-375, Aug. 1997
- [14] 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, no. 3, pp. 187-200, Jul. 2000.
- [15] J. Loughry, J. I. Hemert, and L. Schoofs, "Efficiently Enumerating the Subsets of a Set," applied-math.org/subset.pdf, Dec. 2000.
- [16] N. Raju, Gottumukkala, Y. Liu, B. C. Leangsuksun, R. Nassar, and S. Scott, "Reliability Analysis in HPC clusters," *Proceedings of the High Availability and Performance Computing Workshop*
- [17] E.N. Elnozahy, L. Alvisi, Y-M. Wang, and D.B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Computing Survey*, vol. 34, no. 3, pp. 375-408, 2002
- [18] R.T Aulwes, D. J. Daniel, N. N. Desai, R. L. Graham, L. D. Risinger, M. A. Taylor, T. S. Woodall, M. W. Sukalski, "Architecture of LA-MPI, a network-fault-tolerant MPI," *Proceedings of the International Parallel and Distributed Processing Symposium*, Apr. 2004
- [19] D. A. Reed, C. Lub, and C. L. Mendes, "Reliability challenges in large systems," *Future Generation Computer Systems*, vol. 22, no. 3, pp. 293-302, Feb. 2006
- [20] C. Lu, and D.A. Reed, "Assessing Fault Sensitivity in MPI Applications," *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, Nov. 2004
- [21] J. B. Weissman, "Fault Tolerant Computing on the Grid: What are My Options," *Proceedings of the Eighth International Symposium on High Performance Distributed Computing (HPDC)*, 1999.
- [22] O. Khalili, J. H. Olschanowsky, C. Snavely and H. Casanova, "Measuring the Performance and Reliability of Production Computational Grids," *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, Sep. 2006.
- [23] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)* San Francisco, California, Aug. 7-9, 2001.
- [24] R. Sandberg, D. Golgberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and implementation of the Sun network filesystem", *Innovations in Networking*, pp. 379-390, 1988.
- [25] F. Schmuck, and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, 2002
- [26] A. Yarkhan, J. Dongarra, and K. Seymour, "GridSolve: The Evolution of Network Enabled Solver," *Grid-Based Problem Solving Environments: Working Conference on Grid-Based Problem Solving Environments*, P. Gaffney, J. C. T. Pool Eds Springer, 2007, pp. 215-226.
- [27] Condor Team, "DAGMan Metascheduler," <http://www.cs.wisc.edu/condor/dagman>
- [28] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbre, R. Cavanaugh, and S. Koranda, "Mapping Abstract Complex Workflows onto Grid Environments," *Journal of Grid Computing*, vol. 1, no. 1, pp. 25-39, 2003.
- [29] S. Hwang, and C. Kesselman, "A flexible framework for fault tolerance in the grid," *Journal of Grid Computing*, vol. 1, no. 3, pp. 251-272, 2003.
- [30] G. Alonso, C. Hagen, D. Agrawal, A. E. Abbadi, and C. Mohan, "Enhancing the fault tolerance of workflow management systems," *IEEE Concurrency*, vol. 8, no. 3, pp. 74-81, 2000.
- [31] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, and L. Johnsson, "Scheduling Strategies for Mapping Application Workflows onto the grid," *Proceedings of High Performance Distributed Computing (HPDC 2005)*, pp. 125-134, 2005
- [32] J. Yu, R. Buyya, and C. Tham, "Cost-based Scheduling of Workflow Applications on Utility Grids," *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing*, Dec. 5-8, 2005.
- [33] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi, "Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources," *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, 2007.
- [34] Z. Shi, and J. J. Dongarra, "Scheduling workflow applications on processors with different capabilities," *Future Generation Computer Systems*, vol. 22, no. 6, pp. 665-675, 2006.
- [35] Y. Zhang, C. Koelbel, and K. Kennedy, "Relative Performance of Scheduling Algorithms in Grid Environments," *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, 2007.
- [36] J. Yu, and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, Technical Report, GRIDS-TR-2005-1*, Mar. 10, 2005.
- [37] M. Wiczcerek, R. Prodan, and A. Hoheisel, "Taxonomies of the Multi-criteria Grid Workflow Scheduling Problem," *CoreGRID Technical Report Number TR-0106*, Aug. 30, 2007
- [38] A. A. Sedrakian, R. M. Badia, J. M. Pérez, R. Sirvent, T. Kielmann, and A. Merzky, "Reliability and Trust Based Workflows' Job Mapping on the Grid," *Institute on Grid Systems, Tools and Environments, CoreGRID Technical Report, TR-0069*, Jan. 30 2007
- [39] S. Jarvis, N. Thomas, and A. Moorsel, "Open issues in grid performability," *International Journal of Simulation: Systems, Science and Technology*, vol. 5, no. 5, pp. 3-12, United Kingdom Simulation Society, 2004