A Survey of Self-Adaptive Grids

Daniel M. Batista and Nelson L. S. da Fonseca, University of Campinas

ABSTRACT

Grid systems allow the execution of a class of highly demanding services and applications. These grids involve communication networks, and their links are essential resources for massive data transfers. However, the management of current grid systems requires intervention for efficient service provisioning. Moreover, this need increases with the increase in demand for grid services. Therefore, grid systems will become effective only when they are capable of self-managing resource allocation to cope with fluctuations in resource availability. At present, however, very few integrated self-adaptive mechanisms have been implemented in existing grid systems. The aim of this article is to provide a survey of existing mechanisms and suggest directions for enabling autonomic operation of grid systems.

INTRODUCTION

Highly demanding applications have traditionally been processed on supercomputers or clusters of computers, which are confined systems typically belonging to a single owner. This has limited the type of application that can be processed, since even an incremental expansion of the computing system can be quite expensive.

The introduction of high capacity optical links in the Internet led to the growth of a global communication infrastructure with increased connectivity by several orders of magnitude, shorter delays and decreased loss of information. This global infrastructure has enabled the interconnection of remote computing systems, thus creating grids with resources belonging to different organizations, but which are shared in a cooperative way. In grids, the network links serve as the data bus for the virtual computing system, resulting in a global span shared by a multitude of users (Fig. 1). A distinct difference between grids and clusters of computers is the vital role played by the communication network in transferring data between remote systems. This data transfer, however, is subject to dynamic fluctuation in link loads.

The increase in availability of resources has made possible the processing of a class of applications for various sciences, including chemistry, physics, and pharmacology. One example is the processing of information generated by the Large Hadron Collider (LHC), the largest scientific instrument so far created for high-energy physics research. Its operation demands the storage and transfer of 15 Petabytes per year, a volume that can only be achieved by the use of grids.

Grids are systems that coordinate resources which are not subject to centralized control using standard, open general-purpose protocols and interfaces to deliver nontrivial qualities of service [1]. The autonomic management of these grids aim at provisioning transparent services. Despite the advantages of such a paradigm, several challenges must be met to make the management of grids autonomic. The lack of resource ownership by grid schedulers, the fluctuations in resource availability, and the uncertainty in application demands require that grid systems to employ management mechanisms that enable them to adapt to environment changes; such mechanisms are known as self-adaptive mechanisms and involve the ability to discover, monitor and manage the use of network resources. They should react to the occurrence of events and make autonomous adequate decisions.

The identification of self-adaptive mechanisms that will enable the desired level of automation is, thus, necessary to design the next generation of grid networks. Understanding the benefits and limitations of the mechanisms actually employed in current grids should facilitate the design of more robust and transparent grids.

This article provides a brief survey of self-adaptive mechanisms and the major challenges involved in the autonomous management of grids. It differs from previous papers [2, 3] by providing a broader coverage of existing systems. Although other surveys of the autonomous operation of grids have been published, they are limited to specific type of grids, such as grids for workflows [2] and data grids [3]. In addition to the analysis of a larger set of grids, this article is not restricted to specific types of services and applications. Furthermore, the information supplied here can help users determine which of the existing grid systems best fits their needs, as well as assist grid administrators in pinpointing what needs to be improved in their systems to make them more autonomic.

The rest of this article is organized as follows. First, the motivation for self-adaptive mechanisms for resource allocation is presented, and the desired self-adaptive characteristics are described. Then seven grid systems are identified and analyzed. Finally, these grids are compared and recommendations designed to promote autonomic management are made.

0163-6804/10/$25.00 © 2010 IEEE

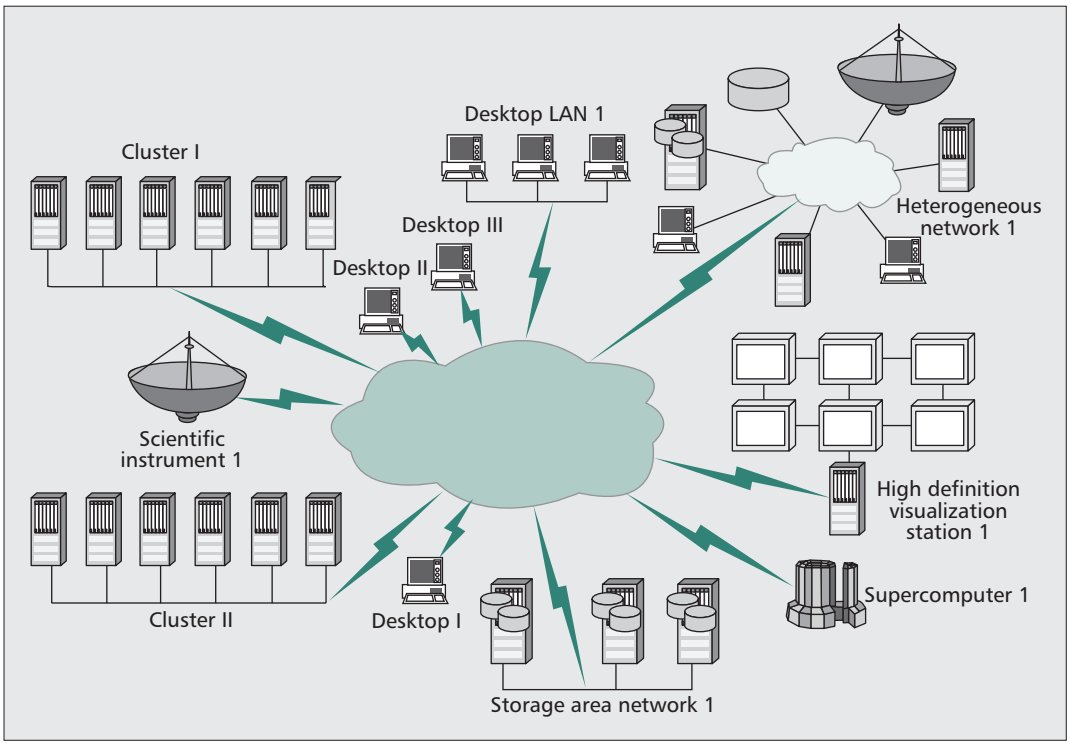

Figure 1. *Example of grid resources.*

The performance of a grid application relies on the efficiency of the scheduling of the tasks, i.e. the efficiency of mapping of these tasks onto available resources and the coordination of the execution of tasks on these resources.

Techniques for Resource Allocation In Grids

Before their execution, grid applications are segmented into smaller processing units, called tasks. Data dependencies among tasks determine the demand for bandwidth, and typically these tasks transfer huge amounts of data among themselves.

The performance of a grid application relies on the efficiency of the scheduling of the tasks (i.e., the efficiency of mapping these tasks onto available resources and the coordination of the execution of tasks on these resources). In other words, to achieve enhanced performance, proper resource allocation is necessary. Moreover, the availability of shared resources can change after the scheduling of tasks, with scheduling decisions becoming ineffective for the new scenario. Furthermore, the lack of precision in the estimation of application demands and resource availability introduces unavoidable uncertainties into scheduling decisions. As a consequence, various approaches have been adopted by different grid systems to deal with these problems, including both dynamic scheduling and adaptive scheduling.

Dynamic scheduling is useful when some of the resource requirements of an application are not known at the time of the scheduling of the first tasks of that application. In a directed acyclic graph (DAG) representation, such a situation is represented by unknown edge and node weights; this prevents the definition of a schedule for all tasks at the initial scheduling time. The unknown demands will be discovered only after the completion of certain tasks, and the decision about resource allocation for these tasks will be postponed until these dependencies are resolved. Thus, the scheduling of tasks is pursued in several steps, providing a certain flexibility in relation to the availability of resources. Adaptive scheduling, on the other hand, is useful for coping with fluctuations in resource availability, since resources are monitored continuously to provide a precise view of their availability at the time when a task is scheduled. Adaptive scheduling can be used for any type of application, whereas dynamic scheduling is designed for applications with unknown demands.

Although both dynamic and adaptive scheduling take into consideration the dynamics of resource availability, this availability is verified only at specific instants. Dynamic scheduling verifies this availability only when previously unknown demands are identified, whereas adaptive scheduling checks the state of the grid whenever scheduling a task. Both schemes are quite restrictive, however, and fail to exploit various opportunities, thus preventing a dynamic search to guarantee the minimum execution time of an application. Changes during the execution of a task are also neglected, which can increase the execution time. Furthermore, both approaches fail to address another important issue: the necessity of task migration to decrease execution time.

The deficiencies in these two approaches can be illustrated by the example in Figs. 2 and 3. This numerical example was derived via simulation using a special `NS-2` module for grid networks. In Fig. 2 the edge weights in the DAG represent the amount of data to be transferred, in gigabytes, and the node weights represent the quantity of instructions on a 10^{12} scale. In Fig. 3, the network has 34 hosts arranged around a central host (SRC_0), and the grid has 11 nodes ($SRC_{\{0...10\}}$). The available processing rate of the host SRC_0 is 1600 MIPS, whereas that of all

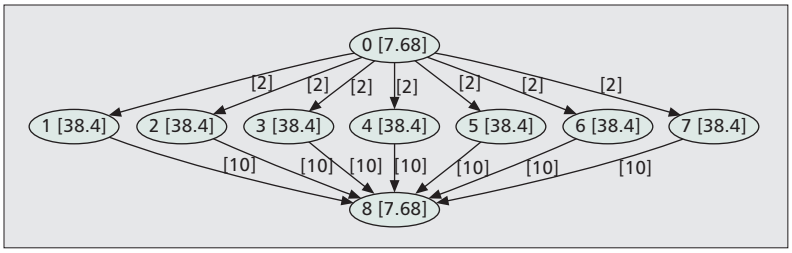

Figure 2. *Example of DAG.*

the other nodes is 8000 MIPS. The links connecting SRC_0 to the other hosts have a capacity of 100 Mb/s each, whereas all the others are limited to 33.33 Mb/s. Note that the topology is not centralized around SRC_0, so the hosts can communicate with each other without going through the central node. Moreover, the processing capacity of node SRC_0 is less than that of the other hosts, which means tasks must be executed in parallel on the other hosts.

Ninety minutes after the initial scheduling of the tasks, there is an increase in UDP traffic between hosts IR_2 and IS_2 and between hosts IR_5 and IS_5 at a rate of 90 Mb/s. When a dynamic scheduling approach is employed, the execution time is 300 min; whereas with adaptive scheduling no migration takes place, and the execution time of the application is 358 min. The use of self-adaptive mechanisms to manage this grid, however, triggers task migration as a response to the increase in traffic, leading to an execution time of only 281 min. In the next section, grid management using self-adaptive procedures will be described.

Characteristics of Self-Adaptive Grids

Figure 4 illustrates the execution of an ideal cycle for the processing of grid applications. This figure shows the submission, execution, and finalization procedures of an application on a grid that adopts self-adaptive mechanisms for resource allocation. The grid management system monitors the availability of grid resources as well as the performance of task execution. If a different mapping of tasks on the resources is found to reduce the execution time, then task migration will be implemented unless migration overhead exceeds the gain obtained from migration. Such a cycle of management needs to account not only for the fluctuation in resource availability, but also for uncertainty in application demands.

The employment of self-adaptive mechanisms in grid management has been implicit since its inception. Actually, the success of this emerging technology, as of any technology, depends on the level of autonomy. The less the users need to know about its operation, the greater the chances of the new technology being established in the long term.

The characteristics of grid systems leading to a capacity for self-management of resource allocation were identified by the authors. They are used in the next section to evaluate the degree of autonomy of existing systems. These characteristics include:

Breadth of scope: Adopting self-adaptive mechanisms oriented toward specific types of applications can lead to poor performance of other applications not covered by the mechanism. One example of this would be certain schedulers that treat applications as if there were no data dependencies between their tasks or ignore link bandwidth as a resource.

Monitored metrics: The number of metrics monitored by a grid system can limit its ability to make the most adequate scheduling decisions. For instance, some systems monitor the available bandwidth but ignore the delay induced by data transfer.

Forecasting overhead: Forecasting the performance based on the measured history of the execution of an application should be possible at all times so that it can be used to trigger changes to improve performance.

Triggering information: Information that enables the triggering of resource allocation should include not only performance degradation, but also opportunities for performance enhancement.

Reaction complexity: Reactions to detected needs for changes in resource allocation vary from partial migration of all tasks to the redefinition of the grid overlay connectivity. The overhead of the action taken should not surpass the benefits arising from the change.

Complexity of (re)scheduling: Schedulers need to provide a schedule that is as close as possible to the optimal one, in a timeframe for which that schedule will still be valid and useful for decreasing the execution time.

Robustness: Schedulers need to produce near-optimal solutions, and should consider the uncertainty of application demands and resource availability.

Effectiveness under diverse scenarios: A scheduler must be effective for various network and application scenarios.

A Brief Description of the Autonomy of Various Grids

The major characteristics to implement autonomy are described here for seven grid systems. All are well-known systems, and were chosen since they each implement at least four of the eight self-adaptive characteristics discussed above.

GRACE

In Grid Architecture for Computational Economy (GRACE) [4], each resource has an associated cost, and it is assumed that users will try to minimize expenditures for resource allocation.

In relation to breadth of scope, GRACE accepts any type of application. In addition to being able to monitor all types of resources, GRACE also allows the monitoring of the cost of these resources. GRACE migrates tasks in order to cope with fluctuations in resource availability. Migration occurs either when the performance degrades or the cost of a resource increases so much that it becomes unattractive. To make effective migration decisions, check-

points need to be monitored. The problem with this system is that it cannot deal with uncertainties in demand. Its efficiency under heterogeneous environments has not been assessed, since experiments for the validation of GRACE were limited to CPU-intensive applications. In GRACE there is no mechanism to predict future usage of resources; nor does it have mechanisms to deal with the (re-)scheduling of tasks; unknown application demands are also ignored.

Experimental Framework

Huedo *et al.*'s experimental framework [5] adopts the concept of "submit and forget about it." It is not restricted to specific types of applications.

Performance is measured only by CPU utilization and memory utilization. Neither the state of the network links nor storage availability are monitored. The experimental framework promotes changes in resource allocation under various circumstances, such as when performance degrades, new resources become available, faults occur, or application requirements change. Moreover, it allows users to change the schedule during the execution of an application. The changes in resource allocation promote the migration of tasks. Scheduling is based on a greedy algorithm, and checkpoints are monitored to determine task migration. Discrepancies between the predicted and the observed performance of the tasks trigger migration, which provides robustness for the applications. Although a single application was used to evaluate the system efficiency, it was able to adapt itself to various environmental changes. In fact, the only self-adaptive characteristic lacking in this framework is the ability to predict future resource utilization.

Migration Framework for Grids

Migration and rescheduling are the two major mechanisms adopted for minimizing the execution time of applications in the Migration Framework presented by Vadhiyar and Dongarra [6].

Although this framework can handle any performance metric and use any monitor, performance is estimated on the basis of link bandwidth and processing power. Migration involves a contract between users and grid providers, which requires a certain commitment to guarantee the availability of a minimum of resources. Migration is pursued if at least a 30 percent reduction in execution time can be expected. Special attention is given to the computation of the overhead, which includes checkpoint computation. Migration can be triggered by various events, including contract violations, unexpectedly long task execution times, and availability of new resources. Both scheduling and rescheduling employ models constructed for the specific applications submitted to the system. These models allow the prediction of performance under various scenarios. The major drawback of this proposal is the CPU time needed to produce estimations. In terms of robustness, the negative impact of misleading information about application demands is attenuated by taking into consideration the history of the performance of different classes of applications. The problem of this technique is that it is useless for applications executed only a few times on the grid. In terms of efficiency under heterogeneous scenarios, this Migration Framework was tested using the GrADS testbed [6]. Up to 70 percent of reduction in execution time was obtained, and similar reductions were found when the availability of resources increased. However, few experiments have been conducted, and it is difficult to generalize the results obtained [6]. As with the experimental framework described above, the only mechanism not available in this framework is the capacity to predict future resource utilization.

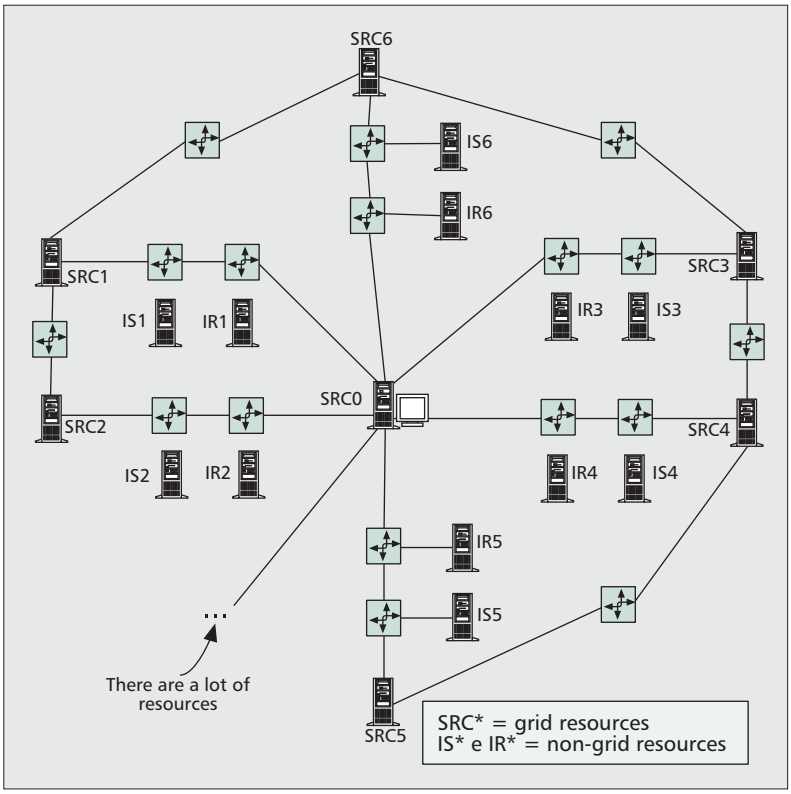

Figure 3. *Example of grid.*

Grid-QoS Management

The Grid-QoS Management (G-QoSM) system [7] allocates resources based on Service Level Agreements between users and service providers. The grid is capable of furnishing quality of service (QoS) and adopts three classes similar to those of the Internet differentiated services (DiffServ) QoS framework: the QoS guaranteed, controlled-load, and best effort classes.

This system does not restrict the type of application that can run on it. The network resource manager (NRM) is employed to estimate the available bandwidth, and the information gathering procedure of Globus middleware is used to monitor the availability of processing power. To avoid overhead, the sampling of intradomain resources is more frequent than that of interdomain ones. Although only link bandwidth and processing power are accounted for, G-QoSM is presumed to be able to monitor several other QoS-related metrics. Both performance (QoS) degradation and new incoming services are considered in the triggering of reallocation of resources. G-QoSM employs heuris-

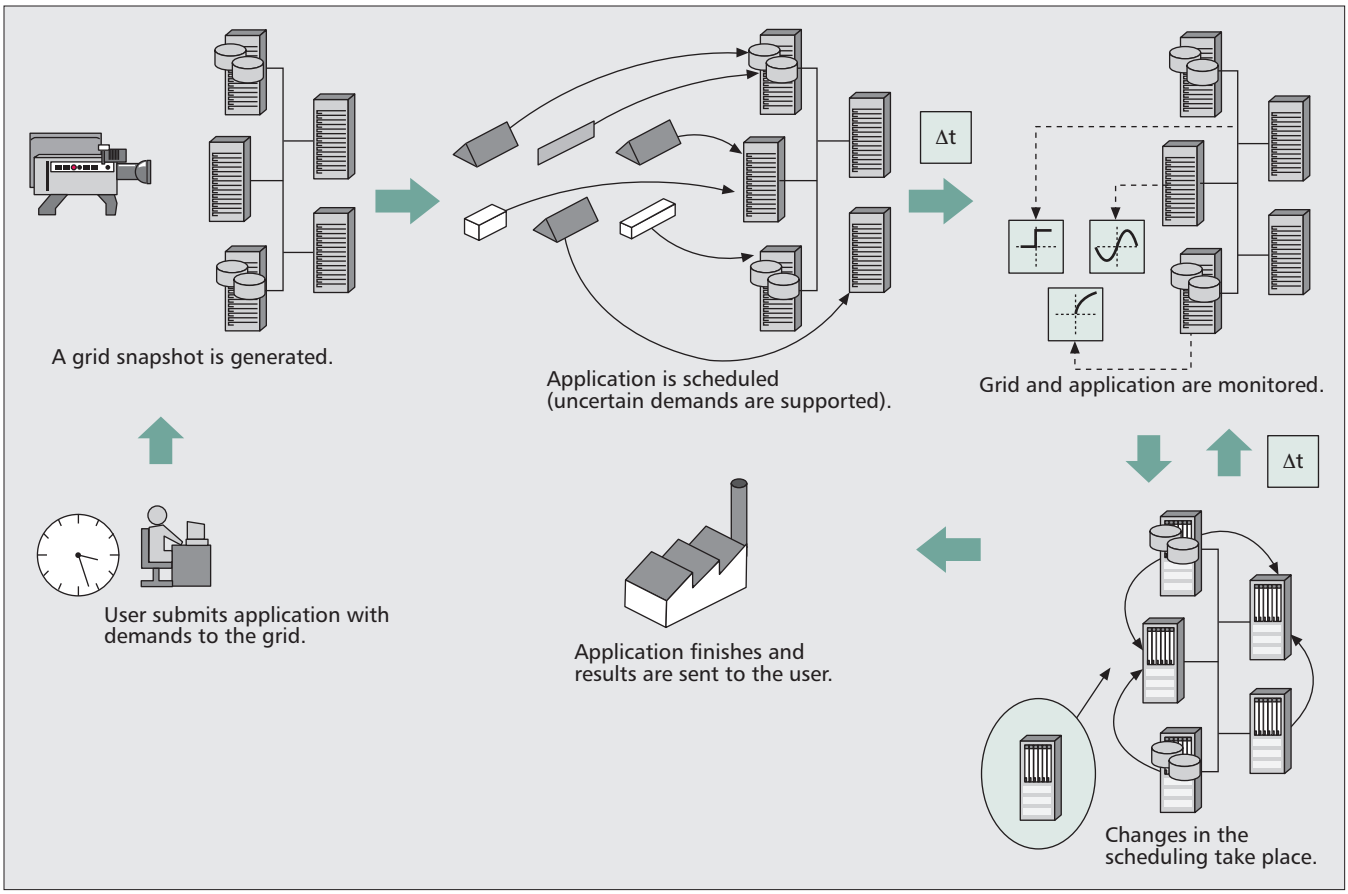

Figure 4. *Ideal execution of the application from the user's point of view.*

tics to find the most adequate set of resources for both scheduling and rescheduling for any QoS class. The major difficulties are the translation of the class of service requirements into resource allocation and the monitoring of the fulfillment of such demands. Moreover, this procedure increases the complexity of the rescheduling of tasks. No consideration is given to robustness under uncertainty in resource availability or application demands. Moreover, G-QoSM does not provide forecasting. No experiments to evaluate its effectiveness were found.

VNET+VTTIF

In the system presented by Sundararaja *et al.* [8], the grid network is seen as an overlay network, with the Virtual Network (VNET), and the Virtual Topology and Traffic Inference Framework (VTTIF) mechanisms used for the management and definition of the grid topology, respectively. VNET deals with the migration of virtual machines so that they remain in the same virtual organization, regardless of physical location, while VTTIF builds overlay networks to satisfy data transfer demands of the applications inferred by the analysis of traffic matrices.

This system is meant for data-intensive applications, and the metrics monitored are those related to data transfer: the number of bytes to be transferred and the available bandwidth. Fluctuations in resource availability change the overlay network topology, which is initially configured as a star. VTTIF passively monitors traffic patterns. The detection of changes in link utilization and application demands results in topology changes. The initial schedule considers idle resources; after the initial allocation, the system seeks the best overlay topology by using self-monitoring data. Such self-adaptability is presumed to make the system more robust, although it ignores uncertainties related to processing capacity. Results from experiments involving various applications on a single grid show that reductions varying from 20 to 50 percent in execution time were achieved after changes in topology, which took an average of one minute to complete. The lack of predictions of resource availability is also characteristic of VNET+VTTIF.

Grid Harvest Service

The Grid Harvest Service (GHS) [9] focuses on monitoring and predicting the state of the grid. It is designed to achieve higher levels of scalability and precision of predictions than those obtained by the Network Weather Service system (NWS), especially for applications that run for long periods.

GHS accepts both processing- and communication-intensive applications. It monitors the available bandwidth and processing capacity during the execution of applications. Data collected through monitoring is used for scheduling and detecting of changes in the grid state. In contrast to the other systems surveyed, GHS does employ

	GRACE	Experimental Framework	Migration Framework	G-QoSM	VNET+VTTIF	GHS	WBA
Breadth of scope	High	High	Medium	High	Low	Medium	Low
Monitored metrics	High	Medium	High	High	Low	Medium	Medium
Forecasting overhead	—	—	—	—	—	Low	—
Triggering information	High	High	Medium	High	Low	Medium	Medium
Reaction complexity	Medium	Medium	Medium	Medium	High	Medium	Medium
Complexity of (re)scheduling	—	Low	High	Medium	Low	Low	Low
Robustness	—	High	Medium	—	Medium	—	—
Effectiveness under diverse scenarios	Low	Medium	Medium	Low	Medium	Medium	Medium

Table 1. *Comparison of grid systems.*

mechanisms for predicting future resource usage. Historical data of resource consumption is used in statistical and neural network models for predicting both processing and bandwidth demands. Both the increase in resource availability and the degradation of application performance affect the grid performance, so GHS migrates tasks in reaction to these changes. This requires the management of checkpoints, thus increasing the management cycle complexity. GHS employs two (re-)scheduling algorithms. One of them, especially adequate for applications with independent tasks, tries to minimize the mean difference in execution time, whereas the other, which is more adequate for applications with dependent tasks, tries to maximize the number of tasks mapped onto a single resource. Although experiments evaluating the performance of GHS have demonstrated its effectiveness, only two grids were actually used in these experiments. Moreover, no mechanism to guarantee robustness in the face of uncertain application demands is provided.

Workflow-Based Approach

The Workflow-Based Approach system (WBA) proposed by Blythe *et al.* [10] is oriented to dealing with data-intensive applications that involve dependent tasks described by workflows. Changes in resource availability trigger the rescheduling of tasks, but no actual migration of process context is implemented.

This system is defined for services and applications described as workflows. WBA monitors both the available processing capacity and bandwidth. Its scheduler, in contrast to what other schedulers do, periodically reschedules an application, trying to allocate as many resources as it can to increase the level of parallelism. Along with grid monitors, this scheduler takes into account existing processing capacity and bandwidth. The scheduler is periodically executed. If two consecutively produced schedules indicate a different resource allocation, task migration is undertaken. However, no mechanism is implemented to ensure robustness under uncertain application demands. Simulation was used to evaluate the performance of WBA [10], although a grid with just six hosts was simulated in the experiments.

Brief Comparison

The degree of self-adaptation of the grid systems surveyed has been classified as either high, medium, or low. The greater the number of classes of applications that can be dealt with under different scenarios, the higher the score for *breadth of scope*. Similarly, the greater the number of metrics monitored and the greater the quantity of information to trigger changes, the higher the scores for *monitored metrics* and *triggering information*, respectively. Conversely, a less complex scheduler, a lower migration overhead, and the use of a smaller amount of control information are linked to an increased score for *complexity of (re)scheduling, reaction complexity* and *forecasting overhead*, respectively. Moreover, the less prone to misleading information and the more flexible to diverse scenarios a grid system is, the higher are the scores for *robustness* and *effectiveness under diverse scenarios*.

Table 1 compares the grid systems surveyed in relation to characteristics of self-adaptation.

Different grid systems involve different characteristics regarding self-adaptation. While WBA is limited to applications with dependent tasks, the Experimental Framework deals with a wide range of applications. The number of metrics monitored varies from one (VNET+VTTIF), to a dozen, as in GRACE.

While the Experimental Framework bases the triggering of changes on information from a variety of sources (e.g., faults, resource requirement changes and new resource availability), VNET+VTTIF considers only that from variation in link capacity. Moreover, scheduling can be quite complex, as with the construction of specific application models in the Migration Framework, to very simple, as with the VNET+VTTIF scheduler, which merely schedules tasks to the least used resource.

Fully autonomic grids are still in their infancy. The adoption of standard benchmarks for fine-tuning mechanisms of self-adaptation would help grid administrators assess the performance and suitability of different grid systems.

Grid systems also differ significantly in relation to monitoring procedures when change-triggering criteria and the scope of applications are considered. The Experimental Framework, Migration Framework, VNET+VTTIF, and GHS have a larger number of the characteristics required of self-adaptation, although these are not always implemented in the most effective way in the seven systems surveyed. The Experimental Framework seems to be the system with the greatest number of these characteristics that are adequately employed.

The comparison provided in Table 1 can be used by grid administrators to move their systems toward a fully autonomic system by replacing manual or semi-automated procedures with autonomic ones. Special attention should be paid by administrators to those mechanisms classified as having a high value in monitored metrics and triggering information, as well as those classified as having low reaction complexity, since the former characteristics lead to efficacy of automation, while the latter leads to efficiency in implementing a management cycle. According to our study, the grid systems that present the greatest degree of autonomy are GRACE and G-QosM.

Conclusions

This survey of existing grid systems takes into account the mechanisms contributing to making grids autonomic. It is shown that most existing systems do not employ procedures for predicting the future state of the grid resources, which would require constant monitoring of the network and CPU/memory resources, and the use of adequate tools for each type of resource. Moreover, the tools should be as non-intrusive as possible to avoid affecting the performance of the grid.

Another aspect that deserves attention is the consideration of uncertainty in estimating application demands. Schedules based on misleading information can result in longer execution times and cause a large number of migrations, which increases the overhead.

Fully autonomic grids are still in their infancy. The adoption of standard benchmarks for fine-tuning mechanisms of self-adaptation would help grid administrators assess the performance and suitability of different grid systems.

References

[1] I. Foster, "What is the Grid? A Three Point Checklist," *GRIDToday*, vol. 1, no. 6, July 2002.

[2] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Grid Computing and Distrib. Sys. Lab., Univ. of Melbourne, tech. rep. GRIDS-TR-2005-1, Mar. 2005.

[3] E. Laure, H. Stockinger, and K. Stockinger, "Performance Engineering in Data Grids," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2–4, Feb. 2005, pp. 171–91.

[4] R. Buyya, D. Abramson, and J. Giddy, "A Case for Economy Grid Architecture for Service Oriented Grid Computing," *Proc. 15th Int'l. Parallel and Distrib. Proc. Symp.*, San Francisco, CA, Apr. 2001, pp. 776–90.

[5] E. Huedo, R. S. Montero, and I. M. Llrorent, "An Experimental Framework for Executing Applications in Dynamic Grid Environments," NASA Langley Research Ctr., tech. rep. ICASE-2002-43, Nov. 2002.

[6] S. S. Vadhiyar and J. J. Dongarra, "A Performance Oriented Migration Framework for the Grid," *Proc. 3rd IEEE/ACM Int'l. Symp. Cluster Computing and the Grid*, Tokyo, Japan, May 2003, pp. 130–37.

[7] R. Al-Ali *et al.*, "QoS Adaptation in Service-Oriented Grids," *Proc. 1st Int'l. Wksp. Middleware for Grid Computing*, Rio de Janeiro, Brazil, June 2003.

[8] A. I. Sundararaj, A. Gupta, and P. A. Dinda, "Dynamic Topology Adaptation of Virtual Networks of Virtual Machines," *Proc. 7th Wksp. Languages, Compilers, and Runtime Support for Scalable Sys.*, Houston, TX, Oct 2004, pp. 1–8.

[9] X. Sun and M. Wu, "GHS: A Performance System of Grid Computing," *Proc. 19th IEEE Int'l. Parallel and Distrib. Proc. Symp.*, Denver, CO, Apr. 2005, p. 228.

[10] J. Blythe *et al.*, "Task Scheduling Strategies for Workflow-Based Applications in Grids," *Proc. 5th IEEE Int'l. Symp. Cluster Computing and the Grid*, vol. 2, Cardiff, UK, May 2005, pp. 759–67.

Biographies

Daniel M. Batista (batista@ic.unicamp.br) received a B.Sc. degree in computer science from the Federal University of Bahia in 2004 and his M.Sc. degree in computer science from the State University of Campinas in June 2006. He is now a Ph.D. candidate at the Institute of Computing, State University of Campinas, Brazil, and is affiliated with the Computer Networks Laboratory at the same university. His research interests include traffic engineering and grid networks.

Nelson L. S. da Fonseca [F] (nfonseca@ic.unicamp.br) received his Ph.D. from the University of Southern California. He is full professor at the State University of Campinas, Brazil. He is the Editor-in-Chief of *IEEE Communications Surveys and Tutorials* and past Editor-in-Chief of the *IEEE ComSoc Electronic Newsletter*. Currently, he is a Member at Large of the ComSoc Board of Governors. He has served as ComSoc Director for Latin America and Director of On-line Services.