Handbook of Research on P2P and Grid Systems for Service-Oriented Computing:
Models, Methodologies, and Applications

Nick Antonopoulos
University of Surrey, UK

George Exarchakos
University of Surrey, UK

Maozhen Li
Brunel University, UK

Antonio Liotta
University of Essex, UK

Volume I

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey · New York

Director of Editorial Content: Kristin Klinger
Director of Book Publications: Julia Mosemann
Development Editor: Christine Bufton
Publishing Assistant: Kurt Smith
Typesetter: Carole Coulson
Quality control: Jamie Snavely
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com/reference

Library of Congress Cataloging-in-Publication Data

Handbook of research on P2P and grid systems for service-oriented computing : models, methodologies and applications / Nick Antonopoulos ... [et al.].
p. cm.
Includes bibliographical references and index.
Summary: "This book addresses the need for peer-to-peer computing and grid paradigms in delivering efficient service-oriented computing"--Provided by publisher.
ISBN 978-1-61520-686-5 (hardcover) -- ISBN 978-1-61520-687-2 (ebook) 1. Peer-to-peer architecture (Computer networks)--Handbooks, manuals, etc. 2. Computational grids (Computer systems)--Handbooks, manuals, etc. 3. Web services--Handbooks, manuals, etc. 4. Service oriented architecture--Handbooks, manuals, etc. I. Antonopoulos, Nick.
TK5105.525.H36 2009
004.6'52--dc22
2009046560

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 21
Self-Adjustment for Service Provisioning in Grids

Daniel M. Batista
University of Campinas, Brazil

Nelson L. S. da Fonseca
University of Campinas, Brazil

ABSTRACT

The fluctuation in resource availability, as well as the uncertainties in relation to requirements for applications, call for the implementation of grids that self-adjust resource allocations to avoid degradation in the quality of service provided for those applications. Various proposals have been made for grid systems that will react to changes in resource availability, which are the key for the creation of service-oriented grids. The purpose of this chapter is to present the main characteristics which are necessary for these systems to provide quality of service. Twelve grid systems are described, highlighting their differences and presenting their strong and weak points for the construction of service-oriented grids. The chapter also presents open research questions.

INTRODUCTION

Grids are systems, not subject to a central controller, that use open and standard protocols in the coordination of resources; their main objective is the provision of services or resources for applications (Foster, 2002). Thanks to such grids, research projects involving different areas of knowledge have emerged (Bethel et al., 2003; CERN, 2007; ESG, 2008]), and the uniqueness of these have led to the coinage of the term e-Science (e-Science, 2008) to describe the collaborative research that is now possible worldwide.

Grids are environments that provide services, rather than isolated resources, and they can be easily understood when applications that require simultaneous resource allocations are considered. Specific allocations must be made so that the final service offered by the grid meets the overall requirements of applications, such as minimum bandwidth and maximum delay.

Grids are usually classified as to services offered. Skillicorn (2002) presents the following classification: computational grids, which support

DOI: 10.4018/978-1-61520-686-5.ch021

high-performance processing; access grids, which provide specialized resources, such as scientific instruments shared by specific organizations; data grids, which furnish access to data sets measured in Terabytes via networks and datacentric grids. It should be clear, however, that the same grid can be used for more than one type of service.

Quality of Service (QoS) requirements and the consequent satisfaction of users have guided the search for mechanisms that will enable the creation of grids furnishing non-trivial QoS requirements. Without such mechanisms, grids would provide only best effort services.

Unlike conventional multiprocessing systems confined to local networks, grids extend throughout various domains and provide diverse services. Moreover, the lack of a central controller, the heterogeneity of resources, constant changes in the capacity available and uncertainties in application requirements make the management of grids a challenging activity.

As there are no guarantees that the capacity available will remain unchanged as time goes by, the **monitoring** of the state of resources and procedures for reacting to changes are major issues to be addressed by the provision of services in grids. Without mechanisms to monitor and to take reactive actions, there is no guarantee that the application requirements can be properly met. The cyclic procedure of monitoring and reaction represents the core of a **self-adjustment** system for resource allocation. In the literature, there are several proposals for mechanisms for use in the process of **self-adjustment** in grids (Wolski, Spring & Hayes, 1999, Buyya, Abramson & Giddy, 2001, Allen et al., 2001, Huedo, Montero & Llorent, 2002, Vadhiyar & Dongarra, 2003, Lowekamp, 2003, Montero, Huedo & Llorente, 2003, Al-Ali, Hafid, Rana & Walker, 2003, Sundararaj, Gupta & Dinda, 2004, Sun & Wu, 2005, Blythe et al., 2005, Prodan & Fahringer, 2005). Many of the solutions are specific for a small set of applications or are specific for grids that provide only one type of service (Montero et al., 2003, Sundararaj et al., 2004, Blythe et al., 2005, Prodan & Fahringer, 2005).

This chapter presents the main characteristics that grids with self-adjustment capability must have to furnish QoS for applications. It also compares twelve existing grid systems, highlighting the differences between them and presenting their characteristic strong and weak points which influence the construction of service-oriented grids. After the comparison of the systems, the chapter ends with the presentation of some problems open to research.

BACKGROUND

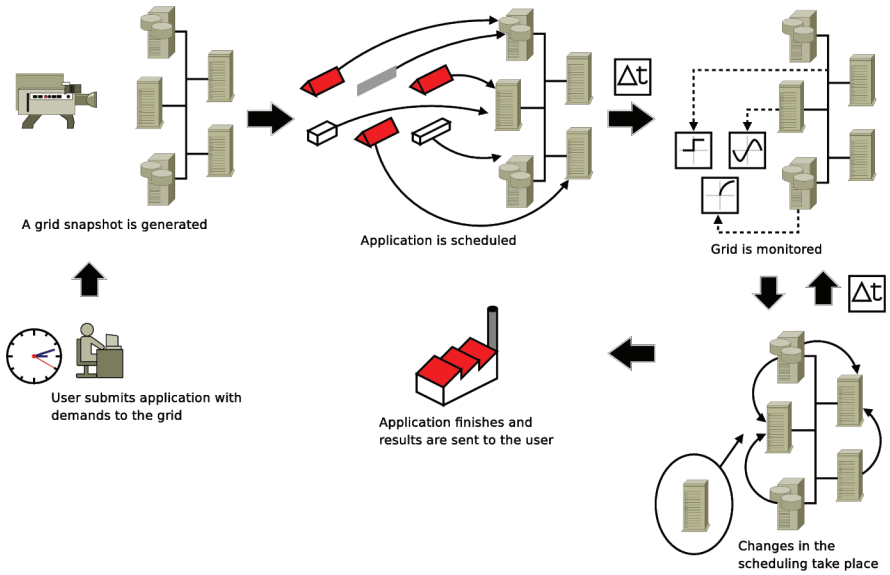
This section introduces the concepts necessary for the understanding of the rest of the chapter. Several issues that justify the need for the establishment of self-adjusting systems are identified, and the relationship between the provision of services and self-adjustment is also highlighted. The steps in the process of grid self-adjustment are also outlined. Previous work relating to service-oriented grids are then described.

Self-Adjustment of Resource Allocation

The **self-adjustment** of resource allocation in grids consists of several cyclic steps that must be undertaken as long as the grid exists (Batista, da Fonseca, Miyazawa & Granelli, 2008). Figure 1 summarizes the functioning of a self-adjustment system. In general, these systems implement steps responsible for monitoring the grid, scheduling applications and migrating tasks. Those steps designed to minimize the time of execution of applications are described first:

1. Given the description of the application and the description of the grid, which can be represented by a graph with vertices for the hosts and edges as links, a schedule is derived to answer the questions: "In which

Figure 1. A system with self-adjustment in execution (changes in scheduling occur during the execution)



host should a task be executed and at what instant in time should these executions and necessary data transfers occur?"

2. The code and the data necessary for each application task are transferred to the defined hosts and task executions are initiated
3. While the tasks are running, the links and hosts are monitored in order to detect changes that may modify the running time expected for the application
4. After information on the state of links and hosts is collected, a comparison must be made between the present state and the previous one. If there have been changes, a new schedule is defined for the tasks not yet terminated (Step 5). If there are no changes, monitoring is resumed (Step 3)
5. Based on the new state of the grid, a new schedule of tasks is established. Only those tasks that have not yet been executed are scheduled
6. This new schedule is compared to the previous one. If they are the same, monitoring is resumed (Step 3). If not, the possible gain to be obtained by migration is checked (Step 7)
7. With the new mapping, the gains and costs for the migration of tasks from the current site to the new one are analyzed. If the gains outweigh costs, migration is undertaken (Step 8). If not, monitoring is resumed (Step 3)
8. When a decision to migrate is made, migration is initiated and monitoring resumed. (Step 3)

The difference between Steps 4 and 6 is critical. The former detects changes in the state of resources and decides if a new schedule is necessary. The latter compares the new schedule with the previous schedule to detect the need for migrations. Step 6 is necessary because changes in the state of resources do not necessarily imply the need for a new schedule.

Each of the steps listed above can be implemented in various ways; the differences between these are fundamental for an understanding of the design of service-oriented grids.

Previous Work

Comparisons similar to those presented in this chapter are found in Krauter, Buyya and Maheswaran

(2002), Yu and Buyya (2005), Laure, Stockinger and Stockinger (2005) and Ranjan, Harwood and Buyya (2008). Krauter et al. (2002) provide a rating and comparison of various mechanisms for allocating grid resources. This classification is meant to facilitate the comparison of various functions performed by the mechanisms, such as resource discovery, scheduling and assessment of QoS requirements. What distinguishes this chapter from previous work is that it compares mechanisms designed for self -adjustment of resources to those carried out by various middlewares. Several of the mechanisms presented in Krauter et al. (2002), are specific for a limited situation and have not been implemented. Moreover, this chapter compares and evaluates eight separate mechanisms (including monitoring, scheduling and migration) whereas Krauter et al. (2002) analyze only three.

Yu and Buyya (2005) present a comparison of mechanisms specific to grids running applications described by workflows, whereas the present comparison is not restricted to a specific type of application.

In Laure et al. (2005), requirements necessary for the promotion of good performance in data grids are evaluated. Although these are mechanisms for the self-adjustment of resource allocation, the focus is on the evaluation of the requirements for the manipulation of data. Only four grids and middlewares are compared in relation to the requirements specified, whereas in this chapter we have included twelve systems.

Ranjan et al. (2008) compare different scheduling schemes and discuss how the techniques for resource discovery on Peer to Peer (P2P) networks can be extended to meet the needs of grid resource allocation. These grid systems are designed only to provide the processing capacity necessary for the applications. Here, however, the grid systems are examined in relation to various aspects which have an impact on the quality of service provided, not only in terms of resource discovery. Furthermore, this chapter does not focus on a single type of grid.

Proposals for the construction of service-oriented grids can be found in Al-Ali et al. (2003) and in Buyya et al. (2001). In the former, applications inform QoS requirements through a service level agreement (SLA), with a mechanism proposed to monitor the execution of the applications; if the SLA is broken, the behavior of the applications is adjusted, or more resources are allocated to these applications. In the latter, various issues related to service management in grids and the associated costs are analyzed.

Previous work has thus been limited in scope, whereas this account presents experiments conducted to evaluate the performance of twelve different proposals. The results are also compared, thus furnishing the reader with a general overview of the functions available and the benefits to be accrued from each specific mechanism. The information presented here expands the work of Batista and da Fonseca (2007) and that of Batista and da Fonseca (2008) in relation to new systems and the number of characteristics considered.

SYSTEMS IMPLEMENTING MECHANISMS FOR SELF-ADJUSTMENT IN GRIDS

Various characteristics of grids with **self-adjustment** capabilities must be considered in the evaluation of the performance of such systems: scope of application, measures monitored, forecast overhead, triggering events, complexity of reaction, complexity of (re)- scheduling, robustness, and experimental validity.

- **Scope of application:** Self-adjustment mechanisms should specify the application requirements for which they were designed. Certain mechanisms implement algorithms focusing on a specific type of application, whereas others try to address more complex applications composed of a large number of dependent tasks. Still other

solutions focus on applications consisting of independent tasks, regardless of the impact caused by the state of the link during execution. If self-adjustment mechanisms not designed for the type of application being run are used, however, the results may be worse than if no self-adjustment had been employed. In order to minimize the need for users to classify applications before executing them, it is preferable to have generic self-adjustment mechanisms oriented to all kinds of applications.

- **Measures monitored:** The **monitoring** of all attributes of resources that influence the execution of applications is quite desirable. For example, for applications that make use of both network and processing resources, it is important to monitor delay, available bandwidth and CPU availability. Various proposals for self-adjustment focus on the available bandwidth and CPU availability, but few consider a more-inclusive monitoring that includes delay on links, memory, and storage space available on hosts.
- **Forecast overhead:** Some mechanisms employ **forecasting** techniques to anticipate changes in the state of grid resources. By using a limited number of measurements, the state of resources can be predicted and an ideal allocation of resources conceptualized. Despite the fact that such forecasting should apparently be mandatory, for all self-adjustment mechanisms, the potential overhead generated by this forecasting must be kept in mind. The ideal would be to have a self-adjustment mechanism able to predict the state of resources based on a limited number of measurements and in a relatively short period of time in relation to the execution time of the application.
- **Triggering events:** Variations in the state of grid resources during the execution of applications can be used to trigger changes in resource allocation. All changes that might cause degradation during execution or that lead to under-utilization of grid resources can be dealt with by changes in resource allocation. Mechanisms that react only in the case of failure of initially allocated resources cannot adequately explore grid resources. The ideal would be to have mechanisms that consider degradation in the availability of initially allocated resources, as well as introducing new resources as potential triggering events for resource reallocation.
- **Complexity of reaction:** After the detection of events triggering new resource allocation, self-adjustment mechanisms must react. Many of the reactions proposed consist of migrating tasks to new resources. Many mechanisms react by changing the way the resources are interconnected in virtual organization, although maintaining the initial allocation unaltered from the point of view of the user. The complexity of the process used by a mechanism has a major influence on the advantages of that mechanism, since it must react before the proposed changes are no longer adequate. Depending on the dynamics of the grid, reactions that take a long time may be useless.
- **Complexity of (re) scheduling:** Scheduling and rescheduling of tasks should be carried out in time frames short enough not to invalidate the decisions taken. In a highly changeable environment, scheduling and rescheduling should occur in very brief intervals. Although several of the mechanisms which have been proposed use heuristics to ensure that critical applications are not penalized, some employ more complex methods for the scheduling of tasks and often require the use of their own grid to establish the schedule.

- **Robustness:** Due to the lack of tools to accurately describe applications, it is a common practice for allocation mechanisms to rely on descriptions provided by users when submitting applications to the grid. Self-adjustment mechanisms have low robustness if they fail to recognize the fact that the user may have provided misleading information about demands, since decisions are taken on the basis of this information rather than what may actually be the case. Therefore, an ideal self-adjustment mechanism would take decisions about application demands based on the Quality of Information (QoI) actually provided, rather than that supplied by the user. In such an ideal system, applications with identical descriptions but different QoIs would receive different treatments by the mechanism.
- **Experimental validity:** Various mechanisms have been evaluated experimentally (Wolski et al., 1999, Buyya et al., 2001, Allen et al., 2001, Huedo et al., 2002, Vadhiyar & Dongarra, 2003, Montero et al., 2003, Sundararaj et al., 2004, Sun & Wu, 2005, Blythe et al., 2005, Prodan & Fahringer, 2005). However, these experiments are often insufficient to draw adequate conclusions. Mechanisms that modify resource allocation must be evaluated for all possible combinations of variations in conditions. Besides considering different conditions that lead to a low performance, experiments to confirm the operation of a mechanism must use a representative set of applications, grids, and variations in the state of resources and uncertainties in the application descriptions made by users.

The rest of this section describes some of the systems used to implement mechanisms for the self-adjustment of resource allocation. Finally, a comparison of these is presented, highlighting the characteristics relevant for the construction of such self-adjustment systems.

Systems Description

This section will describe twelve grid systems that implement mechanisms for the self-adjustment of resource allocation. Each of the twelve has its own unique self-adjusting mechanism. This chapter does not consider all existing grid systems, but only those which are well known and documented.

Network Weather Service (NWS)

The NWS, presented in Wolski et al. (1999), provides distributed service for the forecast of grid performance. NWS has been adopted for monitoring and forecasting services by various scheduling proposals and the self-adjustment of resource allocation.

NWS focuses on **monitoring**, as well as the **forecasting** of the state of resources. Therefore, it does not specify techniques for use during scheduling or rescheduling. NWS measures and foresees the available processing capacity of the grid host, as well as the time necessary to establish a Transmission Control Protocol (TCP) connection; the end-to-end network delay and end-to-end bandwidth availability between hosts are also considered.

For forecasting, various time series models have been applied, with the predicted values compared to those that are measured. The model providing the least error is then the one used for future forecasting. Available processing capacity is monitored on the basis of the correlation of measurements made using both active and passive techniques. The frequency of active measurements is adjusted adaptively: if the latest values of the available processing capacity remain relatively stable, the time interval is increased; if not, this interval is decreased. The monitoring of the characteristics of an end-to-end network is achieved

only by active measurements. The measurement of delay and available bandwidth involves the transfer of a fixed amount of data from one grid host to another. The time needed to establish a TCP connection is determined by establishing and terminating an artificial connection. Network overloading and unreliable measurements can be avoided by the configuration of sensors in the hosts which are organized hierarchically so that active measurements will be made only of a representative subset of the available sensors.

The forecasting techniques of NWS are oriented toward applications that require information about the state of resources at time intervals of tens of seconds to minutes, i.e. short-term applications (Wolski et al., 1999). Such a short interval of time does not allow the NWS to obtain a long-term view of the state of the grid, although this can lead to the generation of inefficient initial schedules and the migration of tasks for applications which will takes hours to be executed.

Although NWS does use forecasting techniques to deal with uncertainty in resource availability, no mechanism treats uncertainty in application demands. The graphs resulting from experiments using NWS to evaluate forecasting techniques have been presented (Wolski et al., 1999), but no details as to the workload generated in these experiments are provided. For most NWS forecasts, the available bandwidth is underestimated.

Grid Architecture for Computational Economy (GRACE)

The GRACE presented in Buyya et al. (2001) is an architecture oriented to the regulation of the supply and demand of resources with associated financial costs. The goal of GRACE is to provide services to help owners and consumers of resources maximize their interests. While owners provide resources with the aim of making profit, consumers seek to obtain the required QoS at the lowest possible price. GRACE defines high-level blocks that can be implemented independent of middleware, type of resources and applications running on a grid.

GRACE was designed to make use of existing middleware services and requires a minimum set of resources. Among the various components that must be implemented is the resource broker, which is responsible for mediating between the user and the grid resources. It is responsible for resource discovery, application scheduling, and detection of changes in the grid state, as well as adaptations to these changes. GRACE leads users to perceive of the grid as a set of resources that meet specific requirements of QoS at a price that users agree to pay. Buyya et al., 2001) recommend the use of the Nimrod/G broker and use it in all the experiments they report.

One of contributions of GRACE is to allow adaptive resource allocation. Adaptation is beneficial only if users get to know the requirements of their applications, since only then will they be able to specify budgets for the use of the grid resources.

Since GRACE was designed to be as generic as possible, there is no limitation on the type of application. Several attributes of the hosts and of the network can be monitored and charged. The same can be said about rescheduling and migration of tasks.

The performance of GRACE was evaluated using the EcoGrid testbed, which aggregates resources from 4 continents. A hundred and sixty-five CPU-intensive tasks, each requiring approximately five minutes, were executed, with the service requested being that all of them be performed in a single hour. Performance was evaluated according to the price paid by the user for the utilization of resources. In a first experiment, resource choice was based on the price of each resource. It was found that most of the resources used were the cheapest ones. Although more expensive resources had a higher capacity, they were the least used. Even using the cheapest resources, however, the timing requirement of application was generally met successfully.

Without the use of algorithms to minimize the cost paid by the user, there was an increase of around 50% in the total value, which proved the effectiveness of the proposal when costs are associated with resources.

Two problems in the proposal made in Buyya et al. (2001) should, however, be considered: the lack of evaluation of the performance in relation to network links and migration due to variations in the cost of resource utilization (although the latter is mentioned an issue for future research in Buyya et al. (2001)).

Cactus Worm

The Cactus Worm (Allen et al., 2001) is a parallel computational framework for the execution of applications. Cactus Worm incorporates structures in the applications to promote adaptation in the face of changes in the state of grid resources. It also implements mechanisms for resource selection that will migrate tasks if the application performance drops below a certain pre-defined level.

Monitoring and the decision to migrate tasks are implemented on the basis of violations in QoS contracts and the inclusion of new resources which can potentially improve the execution of the application. Contracts in terms of the minimum requirements to be provided are made between users and providers. The requirements that can be evaluated are operating system, memory and available bandwidth. The monitoring of the state of resources is carried out by using the Monitoring and Discovery System (MDS) tool, implemented with Globus middleware. On the other hand, task scheduling is based on the algorithm of the Condor middleware, which selects the resources that satisfy the application requirements at the lowest price.

One distinctive characteristic of the Cactus Worm is its migration procedure. The data necessary for a task to continue execution on another host can be sent to a storage site before the migration to that new host. The problem with this approach is that the storage site may well become a bottleneck in the system and slow down migration. The advantage of this approach, however, is that it can allow task migration between resources that are in hosts which are not directly connected.

In Allen et al. (2001), the applications assigned to Cactus were those of large-scale simulations tending to take hours to complete. One disadvantage is that there is no mechanism to deal with uncertainty in either application demands or resource availability, nor for the prediction of the grid state.

The performance of the Cactus Worm was evaluated using a Grid Application Development Software (GrADS) testbed, which is a grid that aggregates processing resources from various universities in the United States. The results of the experiment reported did demonstrate the effectiveness of the system for detecting violations of contracts, but no details about the application used were provided except for the fact that the results were based on a simulation. Moreover, the framework as a whole can not be evaluated, since there is no information provided for comparing whether or not there was a true gain when using it.

Experimental Framework

The system presented in Huedo et al. (2002) is called a framework for executing applications in dynamic grid environments. The objective of this framework is to provide intelligent and autonomous grids that will allow users to execute applications in the mode “submit and forget”. The framework was presented as a solution for adapting the execution of applications; it promotes migrations in the following cases: i) degradation of application performance, ii) changes in availability of resources and iii) changes in application requirements.

In Huedo et al. (2002), the components of this framework are presented at the same high level as are those of GRACE, and the mode of functioning

is similar to that of GRACE, with implementation being independent of the applications, the grid and the middleware.

A greedy algorithm was adopted to schedule tasks for minimizing the execution time. However, although it provides rapid initial scheduling, the problem of this technique is that it does not take into consideration network costs, and this is aggravated by the need for task migration. There is no specific technique for the monitoring of resources, although some tools have been suggested for this purpose, such as the NWS. In the proposal, there was no reference to the consideration of network resources which could be requested by applications; however, processing capacity and memory can be allocated on hosts by applications. In Huedo et al. (2002), there is no restriction on application. However, some changes in applications are necessary to make them aware of the grid state.

The performance of this framework was evaluated using the Tidewater Research Grid Partnership (TRGP) testbed, which connects resources from two research institutions in the United States. A single CPU-intensive fluid dynamic application was executed. The framework revealed good results in 4 situations: i) when migration was to better resources than those initially allocated, which shortened the execution time by 42%, ii) when migration was due to performance degradation, which decreased execution time by 35%, iii) when changes occurred in application requirements and migrations were able to meet these new needs, and iv) when network links were disrupted, as the execution of the application was restarted with other resources. These results show the effectiveness of the proposal for the most common situations. However, it has not yet been evaluated for data-intensive applications, nor for the situation of a decrease in available bandwidth.

Moreover, no treatment of uncertainty in information about resource availability is incorporated, nor is any technique adopted to predict the future state of the grid. Uncertainties in application demands are handled through rescheduling and task migration. In the rest of this chapter, this framework is referred to as the "Experimental Framework".

Migration Framework for Grids

Vadhiyar and Dongarra (2003) present the migration framework for grids system. This system was designed to improve the response time of individual applications on the grid through the implementation of coupled techniques to suspend or migrate tasks in the event of fluctuation of resource availability. Information about the execution time of applications is used to evoke the most profitable migrations and avoid unnecessary migrations of those tasks for which termination is eminent. The proposal is oriented to applications with executions lasting for several minutes.

In Vadhiyar and Dongarra (2003), the monitoring of the progress and the migration of tasks is treated as two separate actions conducted jointly in order to avoid unnecessary migrations. A decision to migrate is taken after a comparison of the execution time of a task on the present host to that on one where it could theoretically migrate. Migration takes place if the execution time would be reduced more than 30%. The authors themselves admit that this is not a particularly good approach, since gains of less than 30% can still be attractive.

The scheduling mechanism adopted is based on the construction of a specific model for the execution of the application, which requires previous knowledge of its requirements. The NWS is used for the monitoring of resources. The proposal contains no explicit information about the resource attributes evaluated, although the experiments reported did consider both available bandwidth and available processing capacity.

Not only did the system compute the time of execution for the application, but also used fixed overheads from previous experiments in real environments to estimate the new execution time.

Experiments to evaluate the performance of this migration framework were carried out on the GrADS testbed. The ScaLAPACK application (ScaLAPACK, 2007) was used for these experiments and executed with different combinations of parameters to produce different execution times. The decision of the framework was reported to be adequate in all except one of the cases tested. In this single case, the application stayed on the original host, although the best option would have been to migrate to another host; this led to the failure to obtain the reduction which could have been obtained of around 35% in execution time. In general, the reduction in execution time was around 70%. Other experiments showed the effectiveness of the proposal when new resources became available during the lifetime of the application.

In the rest of this chapter, this framework is referred to as "Migration Framework".

Watching Resources from the Edge of the Network (Wren)

Another proposal based on **monitoring** is presented in Lowekamp (2003), but, unlike the NWS, it monitors only the state of the links that connect the hosts. This proposal is part of the project Wren, a project for the development of scalable solutions for network monitoring.

The Wren project monitors links and discovers the grid topology. It uses passive techniques when there is traffic on the network generated by grid applications but active ones when no grid application is on the network, or when the existing traffic does not consume much bandwidth. This approach tends to reduce the interference of monitoring on the network. The proposal promises to be scalable for various types of applications: from those which make heavy use of the network to those that are 100% CPU-intensive.

However, procedures for rescheduling, migration of tasks and dealing with uncertainties in application requirements and resource availability are not specified. Moreover, no experimental results are provided to prove the effectiveness of the proposal.

One contribution of Wren is the introduction of methods and functions that enable the use of low-overhead active probes to predict the network state when the transfer of a large amount of data takes place.

GridWay

The GridWay project (Montero et al., 2003) uses Globus middleware to make the execution of tasks easier and more efficient regardless of the dynamics of the environment. This project borrows some of the ideas and solutions implemented in the Experimental Framework discussed above.

The GridWay project (Montero et al., 2003) emphasizes the migration of application tasks when resources are available which will lead to a better performance than that initially allocated. Three aspects are examined before a decision to migrate is taken: i) performance of the new host, ii) time remaining to finish the execution of each task, iii) "distance" to the proposed new host where tasks will be terminated. These aspects distinguish the GridWay project from other existing mechanisms for self-adjustment of resource allocation.

The mechanisms employed by the GridWay are oriented to applications which make heavy use of the network, although these are quite common in some research areas, such as particle physics and bioinformatics. Because GridWay is oriented to these kinds of application, there is a genuine concern with the use of the network at the moment of determining the destination for migration, since a large amount of data will be transferred via the network. Migrations will only be performed if the gain, based on the execution time of the application as a whole, surpasses a certain threshold defined by the user.

The scheduling adopted utilizes requirements informed by the application. The hosts are modeled as resources to provide processing capacity,

whereas the network itself is modeled as the resources which provide bandwidth. The memory available does not affect the estimation of potential gains achieved from migration. No methods are specified for monitoring the state of resources, despite the fact that the NWS is considered to be an option for this function. There is no treatment of uncertainty in relation to application demands and resource availability, nor is there any consideration of predictions about the state of resources.

As in Huedo et al. (2002), the experiments carried out were based on the execution of a fluid dynamic processing application using a testbed, although no details of the topology formed by the resources of the grid were provided. Five hosts with heterogeneous capacities were used. An initial first experiment executed the application without migration, which generating a total execution time of about 350 seconds. In the following experiments, migration was carried out at different points during the execution of the application. A 10% threshold was adopted for the triggering of migrations. Migration was proposed only three times, with a reduction in execution time of around 13%. When migration was induced but not proposed by GridWay, the time of execution of the application became longer. In cases in which manual migration led to an execution time shorter than 350 seconds, no reductions less than 10% were recorded.

Grid QoS Management (G-QoSM)

The G-QoSM framework (Al-Ali et al., 2003) is based on heuristics for the adaptation of QoS. The algorithms adopted in this framework target the adjustment of resource allocation to meet the requirements defined by the SLA.

G-QoSM centers on the provision of mechanisms to support **QoS**. G-QoSM services have an associated cost, as in the GRACE system, and the goal of the algorithms adopted is to maximize the profit of service providers. This may not favor the user, who may wish to establish tradeoffs between QoS support and the cost of services.

The types of services supported by the G-QoSM are derived from the Internet Differentiated Services (DiffServ) architecture. For the first, the guaranteed-service class, the grid must provide applications with services exactly as they required. For the second, the controlled load class, applications define a range of values for the quality of service parameter. For the third, the best-effort class, no contract between users and the grid is involved; hence, any resource is potentially allocable for the execution of applications.

Three scenarios can activate allocation adjustment in the G-QoSM framework. The first is the need for new services which cannot meet the required QoS; in this case, allocation can be redefined so these requests for new services will have a higher level of priority than do existing ones. The second is finalization of services; in this case, resources are liberated, and allocations of other applications still in execution are evaluated to detect potential gains from movement. The third is QoS degradation; in this case, the requirements defined in the SLA are no longer met, and this triggers a search for better resources to meet the requirements of the degraded applications.

The available bandwidth is monitored by the Network Resource Manager (NRM), whereas the processing capacity of available hosts is monitored by the information service of Globus middleware. The NRM operates in a hierarchical structure similar to that of the NWS, with information measured more frequently within the same domain than between domains. Despite the fact that only information about bandwidth and available processing capacity are provided, Al-Ali et al. (2003) do state that the G-QoSM is capable of supporting any type of service provided that it can be quantified in an SLA and measured during the execution of applications.

No specification of the type of applications supported by the G-QoSM is provided. The same lack is also found in regard to the treatment of uncertainties in both application demands and estimation of grid state.

It is not clear in Al-Ali et al. (2003) whether the experiments were conducted on a real testbed or via simulations, since not enough details of the scenario are provided, nor do the results provide any information for drawing feasible conclusions about gains due to the use of the proposal.

Virtual Topology and Traffic Inference Framework (VTTIF)

Sundararaj et al. (2004) presents a proposal for managing grids built on overlay networks. In this proposal, an application sees the grid as a virtual machine, with resources interconnected via a virtual network. The virtual network is managed by a special mechanism, VNET, whereas issues referring to the topology of the virtual network and the characteristics of the traffic generated are under the responsibility of the Virtual Topology and Traffic Inference Framework (VTTIF) mechanism. The goal of both of these mechanisms is to adapt the topology of the virtual network to the traffic generated by application tasks. The execution time of an application can be shortened, since there is a tendency to decrease the distance between hosts that intercommunicate more frequently in the execution of tasks.

The target applications for this proposal are data-intensive Bulk Synchronous Parallel (BSP) applications which require large-scale network resources throughout their life cycle. For such applications, the proposal deals adaptively with uncertainties. Initially, a virtual star topology is defined, centered on the user's host, but with tasks being executed on various other hosts. During execution, the VTTIF monitors the traffic generated by the application and modifies the virtual topology in order to adapt to the pattern of communications performed.

The topology by VTTIF is measured by the analysis of a traffic matrix, which stores information about the packets transmitted between all hosts in the current virtual topology. This information is obtained passively.

No information is supplied about how the initial scheduling is performed, nor about the requirements of applications. Moreover, the processing capacity of hosts is ignored during the determination of the best virtual topology, although this can lead to an increase in execution time if the processing of an application task requires a long time. Furthermore, no technique is cited for forecasting the state of resources.

The performance of the proposal was evaluated using a grid built especially for the experiments. The first experiment evaluated the time spent in the selection of a topology for the application, which took about 1 minute. The second experiment evaluated the time saved by adaptation of the virtual topology. Several configurations were evaluated, with reductions in execution time ranging from 22% to 50%. The lowest savings were obtained when applications were allocated to hosts with little available processing capacity, because the proposal considers only the state of the network at the moment of resource allocation.

Grid Harvest Service (GHS)

The GHS monitoring system proposed in Sun and Wu (2005), like the NWS, focuses on the **monitoring** and prediction of the state of the grid.

The GHS was designed to provide measurements and forecasts for grids that execute applications which take hours to complete. Like the NWS, the GHS provides actual measurements and forecasts of the available processing capacity of hosts, as well as of the transmission capacity between pairs of grid hosts. Forecasts in relation to host capacity use statistical models of the resource demands of local processes. Delay and the available bandwidth of the network are predicted by artificial neural networks. GHS uses both active and passive techniques to measure the state of the hosts; however, no details of the techniques used to measure the state of the network are furnished in the proposal.

The GHS does, however, have modules for the rescheduling and migration of tasks. It adopts

two scheduling algorithms. The first distributes tasks among the hosts to minimize the difference of the mean execution times of tasks. The second scheduler attempts to reduce the execution time by allocating tasks to the same host so that the amount of data transferred can be minimized and the execution time kept to a minimum, thus facilitating data-intensive applications.

Preliminary experiments showed the advantages of the use of the GHS to forecast the long-term state of grids. For two different grids, the errors resulting from these forecasts were compared to those resulting from NWS forecasts, and in both cases the GHS forecast was closer to the actual value experienced by the application. The results of the GHS were compared to those of the Apples scheduler, and the former was found to provide 10 to 20% faster scheduling and migrations.

Workflow-Based Approach (WBA)

The scheduling approach presented in Blythe et al. (2005) self-adjusts resource allocation in response to changes in the environment. The proposed algorithm, the Workflow-based Approach or WBA, is oriented to applications described by workflows that are both data- and CPU-intensive. This scheduler distributes resources relatively evenly, thus avoiding the concentrated use of some at the expense of the others. It works for the whole life cycle of an application. If there is any variation in the execution environment that affects the predicted execution time for tasks, these are re-scheduled to other hosts.

No specific information about task migration is furnished in Blythe et al. (2005). The performance of the scheduler is compared to that of the Task Based Approach (TBA), a scheduler that does not consider the dependencies of workflows and that is presented in Blythe et al. (2005).

This scheduler considers both available processing capacity and available bandwidth. The algorithm does not, however, deal with monitoring nor the forecast of the state of the resources.

Uncertainty seems to be a problem, with some experiments demonstrating that the proposal is not scalable for applications with errors in computational and communication weights.

Unlike the other proposals, the experiments designed to demonstrate the effectiveness of the WBA were performed via simulation on a grid consisting of six fully interconnected hosts. The NS-2 network simulator was used to simulate the transfer of data between grid hosts, with resource demands based on a real application used in astronomy. For data-intensive applications, the time of execution decreased by about 50%. Experiments conducted with variations of up to 400% in the weight of tasks and data dependencies showed that the algorithm was sensitive to uncertainties, with the time of execution increasing up to 400%. Errors in communication weights led to a less dramatic increase in execution time of at most 50%. Results such as these, hardly ever found in literature, reinforce the importance of considering the uncertainty in mechanisms that allocate resources in grids.

No experiments were conducted to determine whether the WBA is capable of reacting to changes.

Dynamic Scheduler for Scientific Workflow

The dynamic scheduler presented in Prodan and Fahringer (2005) is for applications described by cyclic graphs. The scheduler employs techniques for the elimination of cycles and produces a modified Directed Acyclic Graph (DAG). The scheduler uses genetic algorithms, and it can be implemented on multiple parallel hosts to produce a schedule in a shorter time interval than that if it had been implemented on a single host.

Uncertainty in applications demands is accounted for by this scheduler. The negative impact of incorrect information is reduced by jointly evaluating the application details supplied by the users and those of a model created for the predic-

tion of the execution time of the most crucial tasks, even before they are executed. The construction of this model requires the execution of the application several times in a controlled environment before submission to the grid. It is assumed that applications have a linear dependence on input parameters so that linear regression methods can be used.

Task migration is triggered both in the case of failure and in the case of an unpredicted increase in execution time. Verification of performance degradation considers the forecast of execution time. If, after a given time interval, the execution of task has not reached a predicted state, it is migrated to another host. Unless there is a model of prediction for an application in execution, migrations will occur if the task does not finish its execution within a pre-established time interval, although this will not always result in gains, given the heterogeneity of applications.

Available processing capacity on hosts and available bandwidth are the resources accounted for by this scheduler, although neither monitoring nor forecasting the state of the grid are included.

The performance of the scheduler was evaluated on a grid testbed consisting of 314 machines spread around several organizations in Austria. The first experiment assessed the execution time of the scheduler. Without parallelism, the scheduler took about 7 minutes to return a schedule for the application used in the simulation. With parallelism, however, the execution time was 59 times faster, with the grid itself used to determine this schedule. The second experiment assessed the savings in time achieved with the scheduler in an environment subjected to variations in available processing capacity. Savings of around 25% were observed in time of execution for the applications studied.

Systems Overview

This subsection summarizes the main characteristics of each of the twelve systems described in the preceding sections. The information is distributed in three tables, with each developing certain related aspects. Table 1 provides a brief overview of the systems surveyed, while Table 2 presents the aspects related to application scheduling and Table 3 presents information related to the treatment of fluctuation and uncertainties, as well as summarizing the comparative studies conducted. The information provided in these tables is useful for the comparison of the systems, which will be presented in the following subsection.

Table 1 provides a brief description of the systems and identifies the proposed objectives, as well as the types of applications for which they were designed. Although all of the systems surveyed implement mechanisms related to the self-adjustment of resource allocation in grids, their objectives can be seen to be quite different from one another. Whilst some systems clearly specify the guarantee of QoS, such as GRACE and G-QoSM, others are concerned principally with monitoring and forecasting the state of the grid, such as the NWS and Wren. The majority of the systems do, however, aim at a reduction in the execution time of applications.

The terminology used to describe these systems varies considerably, as do the applications for which they were designed. Some systems are defined as services for grids, such as the NWS, while many are described as complete systems designed to deal with the events that justify the rescheduling and migration of tasks, such as the Experimental Framework, Migration Framework, GridWay and VTTIF. The types of applications targeted range from generic characteristics such as execution time, which is used to define the scope of NWS and Cactus, to specific characteristics such as the cyclic workflows targeted by the Dynamic Scheduler. No specifications as to type of applications for which they are relevant

Table 1.Brief summary of the systems surveyed

System	Description	Objective	Applications for which designed
NWS	Distributed service for prediction of resource performance	To foresee the state of hosts and the network accurately	Short-term applications
GRACE	Architecture based on economical computation for the management of grid resources	To guarantee QoS for applications without going over the user's cost limit	Unspecified
Cactus	Mechanisms for adaptive selection of resources	To allow efficient operation of grids	Large- scale simulations taking hours to complete
Experimental Framework	Framework for adaptation to grid dynamics	To allow efficient automatic execution of tasks	Unspecified
Migration Framework	Performance-oriented Migration Framework	To shorten execution time of individual grid applications	Applications taking several minutes for execution
Wren	Measurement system for scalable network clusters in Wide Area Networks (WANs)	To monitor a network independent of applications	i. Bulk data transfer ii. Interactive visualization iii. Optimistic computation
GridWay	Opportunistic "Migrator" for better resource use	To reduce the execution time of applications	Data-intensive applications
G-QoSM	Management framework based on grid QoS	To guarantee the service level of applications by maximizing provider gain	Unspecified
VTTIF	Dynamic adapter of virtual topology	To reduce the execution time of applications via changes in virtual topology	Bulk synchronous parallel applications
GHS	System for performance evaluation and task scheduling	To reduce the execution time of applications	Applications taking long periods of time for execution
WBA	Heuristic for resource allocation and optimization	To reduce the execution time of applications and reduce the idleness of resources	Applications described by workflows
Dynamic Scheduler	Hybrid approach for scheduling workflows based on directed graphs	To self-adjust applications described by graphs containing cycles	Applications described by workflows containing cycles

are mentioned in the discussion of GRACE, Experimental Framework and G-QoSM.

Table 2 summarizes the features of the systems related to the scheduling of applications. It lists the characteristics of the network and hosts in the grids that are considered during scheduling, the methods used for monitoring and predicting the state of resources and the methods employed to schedule applications. Most of the systems focus on the available bandwidth, although a few, such as the NWS and Wren, consider other metrics. The NWS, for example, determines the time required to establish a TCP connection and delays imposed by the host, whereas the Wren evaluates the links and hops that interconnect the grid resources in relation to nominal capacity and that of algorithms implemented to manage queues. Host characteristics considered by most include CPU availability, but some, such as Cactus, the Experimental Framework and GridWay, also consider availability of memory.

Differences in the methods employed to monitor the state of resources are also found. Some systems use other systems for **monitoring**. For example, the NWS is suggested for use by the Experimental Framework, the Migration Framework and the GridWay, and various systems, such as the NWS, Wren, VTTIF and GHS, specify details about the method to be employed in monitoring. However, only the NWS, Wren and GHS employ mechanisms for **forecasting** the state of resources. Some methods schedule applications, ranging

Table 2. Application scheduling

System	Network metrics considered	Hosts metrics considered	Resource monitoring techniques employed	Availability forecast method employed	Scheduling algorithm
NWS	i)TCP Connection Time ii)E2E Delay iii)E2E Bandwidth	i)Fraction of available CPU	i) Active for network ii) Active and passive for hosts	i) Historical + Series ii)Applies the best series based on the past	-
GRACE	Does not specify	Does not specify	Does not specify (Nimrod/G can be used)	-	Does not specify (Nimrod/G can be used)
Cactus	i) Bandwidth	i) Fraction of available CPU ii) Memory	State of allocated hosts and their links	-	Requirements Matching
Experimental Framework	-	i) Fraction of available CPU ii) Memory	Does not specify (NWS can be used)	-	Greedy
Migration Framework	Bandwidth	i) Fraction of available CPU	NWS	-	Requirements Matching
Wren	i) Capacity ii)Queue algorithm iii)Available bandwidth	-	i) Active without execution in grid ii)Passive with execution in grid	Forecasts long-term bandwidth using probes short-term	-
GridWay	i) Bandwidth	i)Fraction of available CPU ii) Memory	Does not specify (NWS can be used)	-	Requirements Matching
G-QoSM	i) Bandwidth	i) Fraction of available CPU	i) Hierarchic for network ii) Globus service for hosts	-	Requirements Matching
VTTIF	i)Bandwidth ii) Topology	-	i) Without information for hosts ii)Passive via traffic matrix for network	-	Does not specify
GHS	i) Bandwidth ii) Delay	i) Fraction of available CPU	i) Does not detail mechanism for network ii) Active and passive for hosts	i) Statistics for CPU ii) ANN for network i)	Heuristics based on the capacity of hosts
WBA	i) Bandwidth	i) Fraction of available CPU	Does not specify	-	Heuristics based on workflow as a whole
Dynamic Scheduler	i) Bandwidth	i) Fraction of available CPU	Does not specify	-	Genetic algorithm and cycle eliminator

from the simplest greedy searches for resources to meet minimum application requirements (Cactus, Migration Framework, GridWay and G-QoSM) to genetic algorithms such as that implemented by the Dynamic Scheduler.

Table 3 summarizes the characteristics of the systems in relation to the actions to be taken if the state of resources fluctuates, or if the information about applications and / or resources is incorrect (uncertainties). The experiments designed to evaluate their performance is also reported, as well as the specific type of event that will lead to changes in the allocation of resources, since some systems reevaluate allocations on a regular basis, regardless the state of the grid (the NWS and Wren), while others take action only when

Table 3. Fluctuations, uncertainties and experiments

System	Events which trigger any reaction	Reaction executed	Rescheduling algorithm	Methods employed to reduce impact of uncertainties	Experiments realized to evaluate performance
NWS	Frequency measurements i) adaptive for CPU ii) fixed for network	-	-	i) In resources: forecast ii) In application: none	Measurements in operational hosts and links (not in grids)
GRACE	Rules based on the application performance	Does not specify	Does not specify	-	Measurements in EcoGrid testbed
Cactus	Rules based on the application performance	Migration (can be done through an intermediary)	A better host than the current one	-	Measurements in GrADS testbed
Experimental Framework	i) Better resource comes up ii)Decrease in the resource performance iii) Resource or network failure iv) Application change v) User's decision vi) Owner's decision	Migration or execution restarts	Greedy	i) In the resources: none ii) In the application: Detects during execution	Measurements in TGRP testbed
Migration Framework	i) Better resource comes up ii) Decrease in the resource performance	Migration to the resource if the gain is higher than 30%	Requirement Matching	i) In the resources: none ii) In the application: builds specific models	Measurements in GrADS testbed
Wren	Measurement frequency	-	-	-	-
Gridway	i)Better resource comes up	Migration (estimate left) to resource if the gain is > threshold	Requirement Matching	-	Measurements in a testbed
G-QoSM	i) Impossibility to attend QoS ii) Liberation of busy resources iii) QoS degradation	Adjustment of allocated resources (does not mention migration)	Requirement Matching	-	Does not detail the scenario
VTTIF	Changes in the traffic pattern in virtual topology of application	Adaptation of virtual topology (creates and removes links in overlay network)	Does not specify	i) In the resources: none ii) In the application: infers the virtual topology by network use	Measurements in experimental grid
GHS	Rules based on state of resources (network and hosts)	Migration to idle resources	First host to attend the requirements	i) In resources: forecast ii) In application: none	Measurements in two grids
WBA	Does not detail	Does not detail	Heuristic based on workflow as a whole	Presented as non scalable	Simulations in NS-2
Dynamic Scheduler	i) Failure in execution ii) Performance decrease	Migration to better resources	Genetic algorithm and eliminator of cycles	Decreases the effect with preview execution and creation of specific model	Measurements in a testbed grid

specific events occur, such as the entry of new resources (Experimental Framework, Migration Framework, GridWay, G-QoSM, VTTIF and Dynamic Scheduler).

In most cases, the reaction to an adverse event will be task migration; most systems reschedule applications on the basis of the same procedure used to generate the initial scheduling. They also

ignore uncertainties and incorrect information in the description of applications and deal with such uncertainties by employing reactive techniques of self-adjustment: if the resources have a capacity different from that which was expected, tasks in execution are migrated.

Comparison of Systems

This subsection compares the grid characteristics of the twelve **self-adjustment** systems discussed. The information is presented in Table 4. A nominal scale based on the terms "Low", "Average" and "High" is used to classify the approximation of each characteristic of each system to an ideal self-adjustment system. If a particular characteristic is not addressed, "-" is used to indicate this.

The interpretation of the terms "Low", "Average" and "High" varies according to the characteristic being compared. For example, those grid systems oriented to a specific type of application are ranked as low for the "scope of application", whereas those reported to be appropriate for all types of applications are ranked as "high". All others are ranked as "average" for this characteristic. Moreover, systems **monitoring** only CPU and bandwidth availability are ranked as "low" in relation to "measures monitored", whereas those monitoring additional metrics are ranked as "average"; "high" is the score for systems allowing the monitoring of user-defined metrics without the need to change the system itself.

Forecasting overhead is concerned with the amount of interference in grid functioning. In this case, systems such as Wren and GHS received low ratings since they interfere very little with grid functioning, whereas the intrusion of NWS led to an average rating.

If triggering events are restricted to changes in resource availability and application requirements a low score for this metric is attributed; moreover, systems which only check potential changes at periodic intervals are also given a score of low. Systems which consider the other changes in the

Table 4. Comparison of critical characteristics of self-adjustment systems evaluated

Systems	Scope of Application	Measures monitored	Forecast overhead	Triggering Events	Complexity of Reaction	Complexity of (re-) scheduling	Robustness	Experimental validity
NWS	Average	Average	Average	Average	-	-	-	Low
GRACE	High	High	-	High	Average	-	-	Low
Cactus	Average	High	-	Average	Average	Low	-	Low
Experimental Framework	High	Average	-	High	Average	Low	High	Average
Migration Framework	Average	High	-	Average	Average	High	Average	Average
Wren	Average	Low	Low	Low	-	-	-	-
GridWay	Low	Low	-	Low	Average	Average	-	Average
G-QoSM	High	High	-	High	Average	Average	-	Low
VTTIF	Low	Low	-	Low	High	Low	Average	Average
GHS	Average	Average	Low	Average	Average	Low	-	Average
WBA	Low	Average	-	Average	Average	Low	-	Average
Dynamic Scheduler	Low	Average	-	Average	Average	Average	Average	Average

grid, like the entrance of better resources or the resources owner's decisions, besides changes in resource availability and application requirements, are given a score of high.

In terms of the complexity of reaction, systems managing both checkpoints and migrations are considered to have an average rating, whereas the restructuring of the overlay network of VTTIF led to its ranking of high complexity.

The **scheduling** and management of applications involving various hosts is one of the main concerns of self-adjustment systems, but such scheduling can approximate an ideal system for scheduling to a greater or lesser extent. If a system uses a greedy approach to (re-)scheduling, it receives a low score for the characteristic "Complexity of (re-) scheduling", whereas systems using more complex schedulers that consider the whole application and are based on more elaborated heuristics (such as genetic algorithms) are ranked as average. Systems which consider not only the whole application but also use pre-defined models have been rated as high.

The consideration of only bandwidth uncertainties is considered to be an average approximation to the ideal. Only when other uncertainties in the system are considered is the system ranked high for **robustness**. The use of predictive models leads to a rating of average if these models are not confirmed.

Any proposal must be verified, and these self-adjustment systems are no exception. If the assessment of the effectiveness of a system is not fully described, that system is given a low score for experimental validity. The report of a single, well-defined scenario for validation leads to the classification of that system as average. Only those systems evaluated using different scenarios and actual testbeds received a high rating for this metric.

An ideal system should rate high in relation to the characteristics "scope of application", "measures monitored", "triggering events", "robustness", and "experimental validity". This ideal system should rate low in relation to all the other characteristics. Table 4 shows that none of the twelve systems surveyed met this criterion. The mechanisms which come the closest to addressing all of them are the following: Experimental Framework, Migration Framework, VTTIF, Dynamic Scheduler and GHS. The first four go beyond processes to foresee the state of resources, whereas the GHS deals with uncertainties other than those of application descriptions submitted to a grid. Although seven out of the eight criteria are addressed by all of these systems, this does not mean that all of them are treated in the best possible way. For example, the experiments carried out to analyze the performance of these five systems do not require confirmation of proper functioning for all events leading to adjustments in resource allocation. Moreover, the scope of application of these systems, the measures monitored, and the triggering events vary greatly.

Except for VTTIF, all mechanisms have similar complexity in the reaction to **fluctuation** of the state of the grid. All these mechanisms react through task migration making necessary the implementation of processes necessary to deal with the treatment of checkpoints.

In short, the scope of applications in the evaluated systems ranges from systems that consider one single class of applications, such as the WBA that is aimed only at applications formed by dependent tasks and which have communication and processing requirements up to systems that employ generic and independent mechanisms, such as Experimental Framework.

In terms of measures monitored, there are systems that focus on a single type of resource as the VTTIF that only monitors the use of links by the applications, and systems that monitor different resources such as GRACE, which considers resource availability by their costs which should summarize the state and usefulness of the resources.

Regarding forecast overhead, it is noticed that the vast majority of systems do not implement any

mechanism to foresee the state of resources. The only systems that deal with this characteristic are the NWS, the Wren and GHS. Among the three systems, the NWS generates a higher overhead due to the need of simulation to foresee the state of network.

As the systems are evaluated in terms of the triggering events, it is noticed that the implemented solutions range from those that react to a set of small variations in grid, as the VTTIF, which only reacts to changes related to the state of links and to the requirements of tasks communication, up to those that react to practically all events such as the Experimental Framework, which includes failures, user's decisions, variations in the performance of current resource in grid, the output and input of grid resources as well as changes in requirements of applications in execution.

The (re) **scheduling** complexity varies greatly among the systems evaluated. There are highly complex solutions such as the one of the Migration Framework, which builds a specific model for the application to be scheduled and forecasts the value of the execution time of each task, up to solutions of low complexity, such as that presented by VTTIF, which chooses the idle resources at the submission time.

It is noticed that few systems implement solutions to deal with robustness. The few solutions implemented range from the change of the overlay topology among the resources, if there are changes in the communication pattern as proposed by VTTIF, up to task migration when there are differences between the application behavior during the execution and the behavior described at the submission time as proposed by the Experimental Framework.

In terms of the experiment validity, none of the systems carried out experiments that demonstrated their effectiveness in a number of scenarios that can be representative, given the inherent heterogeneity of the grids. The proposals basically do not detail the scenarios in which they were evaluated, such as in the case of G-QoSM, or they were evaluated on 1 or 2 testbeds, as in the GHS case.

Taking Table 4 as a reference, and considering all the criteria equally important in the design of an ideal self-adjustment system, the best systems to be used as a starting point for an organization are: Experimental Framework, GHS, Migration Framework, Dynamic Scheduler and VTTIF.

FUTURE TRENDS

This section explores the comparisons presented in Table 4 and identifies research issues related to the self-adjustment of resource allocations which have been left open. The criteria which were most ignored were the forecasting of the state of resources and the treatment of uncertainty in the application descriptions. Moreover, although the effectiveness of the proposals was evaluated, more results are lacking generalizable. These three criteria deserve special attention in future research involving the self-adjustment of service oriented grids so that the **quality of services** provided can be assured.

The need to forecast the state of grid resources can lead to specific investigations for each type of shared resources, since the standard behavior of communication resources is not necessarily the same as that of processing resources. A statistical analysis of grid traces might be a first step in the search for the distribution of probabilities that best describes the behavior of various resources over the life-cycle of applications. Another issue meriting research concerns the interference of **forecasting** tools in the availability of bandwidth, since some knowledge of the current and past states of a grid is necessary to predict a future state; this leads to the need for the installation of constant monitoring systems.

Errors and inaccuracies in the description of application demands can lead to unpredictability in relation to execution time (Blythe et al., 2005), which makes grid management more difficult and may leave the user dissatisfied because performance forecast may be quite different from what

is actually achieved. Three threads of research can be explored in order to reduce the impacts of **uncertainty** in the description of applications. The first concerns the insertion of Quality of Information directly in the task scheduler. Instead of receiving only the application and a description of the grid as input, schedulers would also be informed about the reliability of information provided. Based on this uncertainty, a scheduler can provide more flexible scheduling. In this way, a close to optimal schedule can be adopted, even if actual requirements are different from those present at the moment of submission.

The second thread of research involves decreasing uncertainty. Here, before being submitted, applications would undergo more precise evaluation of requirements (via a specific system). This research clearly represents a much greater challenge than does the first, since the requirements of applications may differ depending on the resource on which a task will be executed. (A task executed on hosts with differences in hardware or operational system may behave differently).

The final thread of investigation involves a comparison of reactive and preventive actions in dealing with uncertainties. Reactive actions may lead to a behavior different from that described upon submission (Huedo et al., 2002), whereas preventive ones involve the inclusion of uncertainty upon initial allocation, thus avoiding unnecessary reallocations.

The final criterion which should be explored is related to the fact that there are as yet no representative scenarios established for the study of self-adjustment mechanisms in grids. Although testbed measurements provide values very close to real, the performance statistics are only valid for a small number of applications and situations. Two approaches for addressing this problem would be the creation of complex benchmarks and the proposal of simple checklists to be followed in performance analysis. These benchmarks would have to be complex, since they should present not only a set of applications, but also a description of events to be used as resource input and output, as well as of the topology formed by the grid resources.

Checklists, though simpler in implementation than benchmarks, require intensive research investment in relation to the characterization and evaluation of events, applications and relevant situations faced by self-adjustment systems in grids all around the world.

CONCLUSION

The lack of a centralized grid control and dedicated resources is one of the factors leading to fluctuations in resource availability, consequently affecting the quality of service. It is thus necessary to implement techniques that can cope with such fluctuations. Self-adjustment of resource allocation are one of the possible mechanisms which can enable the maintenance of the quality of application services.

In the literature, various systems have been proposed to implement mechanisms for dealing with the development of self-adjustment systems. In this chapter, we have presented a summary of twelve of these systems and have compared various aspects of each of them. For this comparison, we have considered eight critical criteria involved in self-adjustment which can influence the quality of the services provided by grids for specific applications. Moreover, we have presented a classification of these systems in relation to the criteria addressed and use this classification to identify as yet unresolved issues. Our comparison has shown that most of the systems available neglect the forecasting of the state of resources and the treatment of uncertainties in application demands. It has also shown that most systems do not conduct representative experiments to test their effectiveness.

We hope that with the greater diffusion of grids, especially the LHC grid, the information provided here will be useful for grid users, as well as for the

designers who project them. Implementing task schedulers that are robust enough to deal with uncertainties in the descriptions of applications, as well as with unexpected changes in the capacity of the resources available, is an interesting development for the future. Moreover, the development of a representative set of benchmarks for the evaluation of self-adjustment systems for resource allocation is of paramount importance.

REFERENCES

Al-Ali, R., Hafid, A., Rana, O., & Walker, D. (2003). QoS adaptation in service-oriented Grids. In *Proceedings of the 1st International Workshop on Middleware for Grid Computing (MGC2003)*. Retrieved November 12, 2008, from http://www.wesc.ac.uk/resources/publications/pdf/MGC289_final.pdf

Allen, G., Angulo, D., Foster, I., Lanfermann, G., Liu, C., & Radke, T. (2001). The Cactus Worm: Experiments with dynamic resource discovery and allocation in a Grid environment. *International Journal of High Performance Computing Applications*, *15*(4), 345–358. doi:10.1177/109434200101500402

Batista, D. M., & da Fonseca, N. L. S. (2007). A Brief Survey on Resource Allocation in Service Oriented Grids. In *Globecom Workshops - 1st IEEE Workshop on Enabling the Future Service-Oriented Internet* (pp. 1-5). Washington, DC: IEEE.

Batista, D. M., & da Fonseca, N. L. S. (2008). Empowering Grids with flexibility to cope with uncertainties. In *ICC Workshops '08 - (CAMAD 2008)* (pp. 227-231). Washington, DC: IEEE.

Batista, D. M., da Fonseca, N. L. S., Miyazawa, F. K., & Granelli, F. (2008). Self-adjustment of resource allocation for Grid applications. *Computer Networks*, *52*(9), 1762–1781. doi:10.1016/j.comnet.2008.03.002

Bethel, W., Siegerist, C., Shalf, J., Shetty, P., Jankun-Kelly, T., Kreylos, O., & Ma, K.-L. (2003). *VisPortal: Deploying Grid-enabled visualization tools through a Web-portal interface*. Tech. Rep. No. LBNL-52940, Lawrence Berkeley National Laboratory.

Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., & Kennedy, K. (2005). Task scheduling strategies for workflow-based applications in Grids. In *IEEE International Symposium on Cluster Computing and Grids (CCGRID'05), volume 2* (pp. 759-767). Washington, DC: IEEE.

Buyya, R., Abramson, D., & Giddy, J. (2001). A case for economy Grid architecture for service oriented Grid computing. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium* (pp. 776-790). Washington, DC: IEEE.

CERN. (2007). *Worldwide LHC computing Grid*. Retrieved November 12, 2008, from http://lcg.web.cern.ch/LCG/

e-Science (2008). *4th IEEE International Conference on e-Science, 2008*. Retrieved November 12, 2008, from http://escience2008.iu.edu/

ESG. (2008). *Earth System Grid (ESG)*. Retrieved November 12, 2008, from http://www.earthsystemgrid.org/

Foster, I. (2002). What is the Grid? A three point checklist. *GRIDToday, 1*(6). Retrieved November 12, 2008, from http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

Huedo, E., Montero, R. S., & Llrorent, I. M. (2002). *An experimental framework for executing applications in dynamic Grid environments*. Tech. Rep. No. 2002-43, NASA Langley Research Center.

Krauter, K., Buyya, R., & Maheswaran, M. (2002). A taxonomy and survey of Grid resource management systems for distributed computing. [SPE]. *Software, Practice & Experience*, *32*(2), 135–164. doi:10.1002/spe.432

Laure, E., Stockinger, H., & Stockinger, K. (2005). Performance engineering in data Grids. *Concurrency and Computation: Practice and Experience, 17*(2- 4), 171-191.

Lowekamp, B. B. (2003). Combining active and passive network measurements to build scalable monitoring systems on the Grid. *SIGMETRICS Performance Evaluation Review*, *30*(4), 19–26. doi:10.1145/773056.773061

Montero, R. S., Huedo, E., & Llorente, I. M. (2003). Grid resource selection for opportunistic job migration. In *Proceedings of the 9th International EuroPar Conference* (pp. 366-373). Berlin: Springer.

Prodan, R., & Fahringer, T. (2005). Dynamic scheduling of scientific workflow application on the Grid: A case study. In *SAC'05: Proceedings of the 2005 ACM Symposium on Applied Computing* (pp. 687-694). New York: ACM Press.

Ranjan, R., Harwood, A., & Buyya, R. (2008). Peer-to-peer-based resource discovery in Global Grids: a Tutorial. *IEEE Communications Surveys & Tutorials*, *10*(2), 6–33. doi:10.1109/COMST.2008.4564477

ScaLAPACK. (2009). *ScaLAPACK Home Page*. Retrieved March 2, 2009, from http://www.netlib.org/scalapack/scalapack_home.html

Skillicorn, D. B. (2002). Motivating computational Grids. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'02)* (pp. 401-406).

Sun, X., & Wu, M. (2005). GHS: A Performance System of Grid Computing. In *19th IEEE International Parallel and Distributed Processing Symposium*. Washington, DC: IEEE. Retrieved November 12, 2008, from http://doi.ieeecomputersociety.org/10.1109/IPDPS.2005.234

Sundararaj, A. I., Gupta, A., & Dinda, P. A. (2004). Dynamic topology adaptation of virtual networks of virtual machines. In *LCR '04: Proceedings of the 7th Workshop on Workshop on Languages, Compilers, and Runtime Support for Scalable Systems* (pp. 1-8). New York: ACM Press.

Vadhiyar, S. S., & Dongarra, J. J. (2003). A performance oriented migration framework for the Grid. In *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)* (pp. 130-137).

Wolski, R., Spring, N. T., & Hayes, J. (1999). The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, *15*(5-6), 757–768. doi:10.1016/S0167-739X(99)00025-4

Yu, J., & Buyya, R. (2005). *A taxonomy of workflow management systems for Grid computing* (Tech. Rep. No. GRIDS-TR-2005-1). Grid Computing and Distributed Systems Laboratory, University of Melbourne.

KEY TERMS AND DEFINITIONS

Fluctuation: Variation in the capacity to provide resources for grid applications as time passes;

Grid: System using both open and standard protocols in the coordination of resources that are not subject to central control, with the goal of providing services or resources for applications (Foster, 2002);

Migration: Transfer of the codes and data of a task from one grid resource to another;

Robustness: Ability of a grid system to maintain a given quality of service for an application, despite fluctuations and uncertainties.

Scheduling: Procedure which defines the resources that will be used for application tasks,

as well as the intervals of time in which these resources will be used;

Self-Adjustment System: System which monitors grid resources and application execution and which reacts to events potentially leading to degradation of performance;

Task: Smallest unit for the execution of an application on a grid;

Uncertainties: Incorrect information about application requirements and / or the capacity of resources available;