Advanced Scheduling Strategies and Grid Programming Environments

Bruno Schulze[1], Geoffrey C Fox[2]

1. National Laboratory for Scientific Computing (LNCC), Av. Getulio Vargas 333, Petropolis – RJ, Brazil
2. Indiana University, USA

schulze@lncc.br, gcf@indiana.edu

1. BACKGROUND

This special issue focuses on "advanced scheduling strategies and Grid programming environments", and is based on selected papers from the 6th International Workshop on Middleware for Grid Computing (MGC 2008) and 2nd Latin American Grid (LAGrid 2008) workshop . The authors were invited to provide extended versions of their original papers taking into account comments and suggestions raised during the peer review process and comments from the audience during the workshops. MGC 2008 relevant contributions have been provided by Kane and Dillaway, Abbes et al., Reis and Cerqueira, Finger et al. and Kuijl et al. From the LAGrid 2008 side relevant contribuitions as well have been provided by Martins et al., Costa and Cardozo, Mury et al. and Cáceres et al. These contributions focus mainly on:

- performance of running parallel algorithms with considerable communication among processors on the InteGrade grid;
- prediction of the availability of a resource (CPU, RAM, disk space) as a fundamental activity in opportunistic grids;
- Cyclotron - a prototype cycle-scavenging grid that addresses the security and usability concerns typical of enterprises considering deployment of such systems;
- a decentralized and fault-tolerant software system, called PastryGrid, for Desktop Grid resources management;
- integration of the MPI programming model as part of the InteGrade grid computing middleware;
- Grid scheduling algorithm using multi-attribute utility theory and multi-objective optimization;
- diagnosis model for detecting and excluding misbehaving units by using mechanisms such as reputation-based analysis, voting scheme and honeypots;
- task distribution, load balancing and failure treatment in a grid environment and
- a tool called CPUReserve which guarantees processing performance isolation among applications and makes resource sharing more flexible.

Grid-enabled environments have been motivated by work in service-oriented computing, and designed as a set of services that are provided by individuals or institutions. In this scenario, a level of automation is necessary to achieve scalability, requiring intelligent infrastructure management capabilities to be utilized alongside existing approaches. Despite the emerging complexity and capability made available in Grid environments, it is worthy to mention that developing cross-grid applications that span different virtual organizations (VOs) has remained difficult. For instance, there is no co-scheduling of resources across more than one VO.

Cloud computing concepts have recently emerged to address limitations of Grid environments, a key focus is the use of virtualization technologies (at both platform and middleware levels) and replication through machine/service imaging. However limitations initially found with Grid computing remain. It is useful to note that a computing Cloud can be designed on top of an existing Grid, as it also comprises heterogeneous components that have been integrated using a top-down approach. Similarly, Cloud computing systems are mostly created with the objective of providing a determined set of capabilities to a user, so the interface is an important part of the design.

Scheduling challenges for enabling the deployment and execution of Grid-based parallel applications have been addressed by the selected papers in this special issue, with the development of models and techniques evaluated accordingly by prototyping and simulations.

2. SPECIAL ISSUE PAPERS

Kane discusses Cyclotron a prototype cycle-scavenging grid that addresses the security and usability concerns typical of enterprises considering deployment of such systems. Although enterprises have large numbers of workstations that remain idle outside of working hours, and are thus ideal for use in a cycle-scavenging grid, current solutions have failed to address the security requirements of many enterprises. Cycle-scavenging grids have been far more successful in academic environments and volunteer computing projects where security considerations are typically less stringent. Consequently, enterprises have continued to favor the deployment of dedicated and centrally administered compute clusters, incurring the cost of the cluster, its infrastructure, cooling, power, and personnel on top of the cost of the workstations on employees' desks that remain idle two-thirds of the time. To address these enterprise requirements, Cyclotron executes jobs inside a virtual machine environment on employee’s workstations. This provides a greater degree of isolation from the workstation host than do other solutions, and allows the use of a homogenous operating environment for grid application development and execution. Access control is managed by a declarative security-policy based infrastructure, that supports flexible authorization rules and the constrained delegation of access rights. Code objects, both virtual machine images and individual code bundles, are treated as first-class entities with a unique identity described by their hash values. This allows Cyclotron to both verify the integrity of deployed code and provide policy-based control over what is allowed to execute in the grid.

Abbes et al. discuss that Desktop Grid systems are attractive when running distributed applications with significant computational requirements. While the increasing number of users of such systems does demonstrate the potential of Desktop Grid, current implementations, for instance Boinc, United Devices, Seti@Home, Distributed.Net and Xtrem-Web still follow the client-server or master/slave paradigm. The computing power which can be obtained from these systems is constrained by the performance criteria of the master. This is particularly the case for data-intensive applications. Then, depending on the performance of the master, with many thousands of workers (or slaves), the central scheduler could become a bottleneck. Such problem does not occur with a decentralized resources

management. Furthermore, centralized platforms require a full supervision by an administrative staff which guarantees the operation of the master. Although the crash of the master is infrequent and replication techniques can resolve this problem when it occurs, we still believe in the need to decentralized approaches, since all computing departments do not have the same type of equipment quality and resources. This paper proposes a decentralized and fault-tolerant software system, called PastryGrid, for Desktop Grid resources management. Its main principle is to eliminate the need for a centralized server, therefore to remove the single point of failure and bottleneck of existing Desktop Grids. Indeed, each node can play alternatively the role of client or server. Our main contribution is to conceive PastryGrid protocol (based on Pastry) for Desktop Grid in order to support a wider class of applications, especially the distributed application with precedence between tasks. Comparing to a centralized system, we evaluate our approach on the Grid'5000 testbed using more than 205 machines and a workload composed of 100 distributed applications (more than 2500 tasks). Obtained results show that our decentralized system outperforms a centralized one (XtremWeb-CH), with respect to the turnaround time.

Reis and Cerqueira describe CPUReserve, a tool which guarantees processing performance isolation among applications and makes resource sharing more flexible. Unlike traditional approaches to isolate performance, CPUReserve is entirely implemented at user level, so it does not incur kernel recompiling or system overload by instancing virtual machines. CPUReserve allows the configuration of the active scheduling policy and simplifies the development of new ones, as policies can be implemented in a scripting language as well as in C. Considering CPUReserve's characteristics, the authors believe that it can be useful in a broad range of situations, including when reservation is desired for isolating performance, saving energy in mobile devices and large scale clusters, reducing the heating produced by machines, and providing different testbeds for scalability tests.

Finger et al. discuss the prediction of the availability of a resource (CPU, RAM, disk space) as a fundamental activity in opportunistic grids, that is, computer grids which employs the idle time of its components to perform high-performance computations. In this work they describe a method to perform such predictions employing Use Pattern Analysis (UPA). This prediction method is based on the assumption of the existence of several classes of computational resource use patterns, which can be used to predict resource availability. For example, one such class can represent a typical working day, another can represent a holiday. Experiments made with an implementation of the UPA method show the feasibility of its use in the scheduling of grid tasks with very little overhead. The experiments also demonstrate the method's superiority over other predictive and non-predictive methods.

Kuijl et al. propose a Grid scheduling algorithm using multi-attribute utility theory and multi-objective optimization. Grid computing emerges as an infrastructure for large-scale data processing, resource sharing, and scientific computing. The algorithm makes the optimal decisions based on the available set of objectives. By comparing to a deadline-and-budget algorithm (DBC) with three objectives, we show that the proposed Multi-Objective Optimization (MOO) scheduling algorithm is

capable of obtaining a broader set of non-dominated solutions. The obtained solutions are also of higher quality, which are in close proximity to the Pareto optimal front.

For **Martins** et al., in a P2P computational grid there is no real assurance that intelligent cheating nodes will not damage neither the grid applications nor the other nodes into the environment. With the aim to handle the existence of these rational nodes, the diagnosis model here presented detects and excludes misbehaving units by using mechanisms such as reputation-based analysis, voting scheme and honeypots. In order to validate the diagnosis model, experiments were run in the a grid simulator, where results show that all intelligent cheating nodes can be detected, with an accuracy of 99.4% of jobs being correctly processed. Besides, a graphical user interface were implemented for visualizing the simulations.

Costa and Cardozo describe in their paper, the integration of the MPI programming model as part of the InteGrade grid computing middleware. Parallel processing is among the first motivations for the introduction of grid computing as well as for a major part of its success. Parallel applications can benefit from the larger amount of resources that is available on the grid as compared to the usual environments based on clusters and supercomputers. The ability to run unmodified parallel applications on grid environments is thus an important goal for grid computing research. MPI is one of the most popular programming models for parallel applications, meaning that a large number of existing parallel applications can benefit from this integration and the ability to be executed on top of resources provided by a grid infrastructure. In addition, because InteGrade is primarily aimed at opportunistic grids, such infrastructure can be composed of existing, non-dedicated, computing resources, which provide their idle capacity for use in the grid. Another important issue addressed in their work is related to fault tolerance, as non-dedicated resources may become unavailable at any time. The execution management of MPI applications on InteGrade thus includes a checkpointing and recovery mechanism, which allows an application's tasks to be paused and resumed on different nodes of the grid when they experience any kind of failure in their previous execution sites. The authors have evaluated the performance of their execution engine, which shows very little overhead compared to MPI applications running on the MPICH2 platform. They argue that such overhead can be compensated by the larger amount of resources available on a grid environment.

According to **Mury** et al., task management in grid computing has always been a challenge. Several papers proposed adaptive models to deal with this issue, taking into account the characteristics of the resources and their behavior in this heterogeneous environment, without linking this analysis to the contribution that could be obtained when also considering the behavior of users or groups of users. The work "Task distribution models in grid: towards a profile-based approach" seeks to deal with this influence and shows the benefits that can be obtained for the solution of problems such as task distribution, load balancing and failure treatment in a grid environment.

Finally, **Cáceres** et al., in the paper Performance Results of Running Parallel

Applications on the InteGrade, investigate the performance of running parallel algorithms with considerable communication among processors on the InteGrade grid. The InteGrade is a multi-institution project that intends to exploit the idle time of computing resources in computer laboratories. Since costly communication on a grid can be prohibitive, the authors explore the so-called systolic or wavefront paradigm to design the parallel algorithms in which no global communication is used. To evaluate the InteGrade middleware, three parallel algorithms were considered: to compute the matrix chain product problem, the 0-1 knapsack problem, and the local sequence alignment problem. Their results show that these three applications running under the InteGrade middleware and MPI take slightly more time than the same applications running on a cluster with only LAM-MPI support. The results can be considered promising and the time difference between the two is not substantial. The conclusion is that the overhead of the InteGrade middleware is acceptable, in view of the benefits obtained to facilitate the use of grid computing by the user.

3. CLOUDS VISION AND BEYOND [MGC,08]

Computing Clouds provide a useful basis for improving current Grid developments (current examples include the Nimbus project, which utilizes Globus Toolkit 4 (GT4)). For instance, the requirement of fault tolerance and resilience (supported through redundancy) can be achieved by automatically deploying more instances of a resource or service image. Also, the need for considering system emulation can be avoided as the service entities will naturally conserve their interfaces. As information and knowledge evolve, different encoding formats may arise, and migration will eventually be necessary to interpret them correctly. However, developers only have to consider maintaining image files of the software representing services in the appropriate layers. A distributed but unique image file repository inside a Cloud could potentially simplify development and maintenance processes. This image file repository would publish system profiles of services, allowing for new equipment to more easily adopt the Grid operating system by downloading an appropriate image file. This scheme has the potential of facilitating maintenance and enforcing levels of homogenization. Image file managers like VirtualBox offer the possibility of using image snapshots, which save smaller changes that affect base image files and thus provide customized images. This is particularly useful for deploying clones that differ only in their network configuration or security profile.

The purpose of using a computing Cloud is to provide a narrower interface while abstracting away implementation details. Installing a Cloud on top of an existing Grid supposes the loss of flexibility, because the layer on top of it (applications, portals) will not have access to the wide range of Grid capabilities. However, different levels of abstraction can be produced to satisfy specific needs. Specialist software, hereby referred to as “agents” can operate intelligently over this infrastructure. Hence, a user should only be aware of the instantiation of the image files containing the agents, and the Virtual Machines (VMs) containing data processing and resource management capabilities, may be managed and scaled automatically.

The concept of ontology needs to be used to characterize knowledge metadata according to the situation. A particular ontology provides a framework for the development and knowledge sharing by establishing a common terminology, and can layout an interface of an element that is part of that environment. A Cloud’s usage mode may be described by an ontology, thereby constraining the interface of any agent participating in the Cloud to cover

aspects of that ontology. This constraint can be enforced in the image files available in the Cloud repository. In an e-Science scenario, multiple Clouds may be necessary depending on the ontology used, each with a defined usage mode according to the particular field or need. Under a Web Services (using WSDL) implementation, a common data terminology among images or snapshots can be attained by using shared XSD files to declare and define data types. However, WSDL cannot currently characterize a common terminology for functions or roles of services. The OWL Web Ontology Language has the necessary capabilities to express this terminology.

Agents participating in the marketplace can be designed to meet a particular usage mode and ontology, allowing them to participate in a Cloud that enforces the use of that ontology. Hence, a Coud can be considered as the implementation of a marketplace. In a brokering scenario, managing the computing profile of a service instance allows dynamic offering of services. Computing resources associated with a service can escalate or diminish according to user needs in a particular time. If a Cloud is implemented on third-party infrastructure, the problem of delivering user needs can be optimized to save unnecessary payment costs. Also, instances of the same Grid services can be deployed with different computing profiles to widen the spectrum of offers and stimulate competition.

In this context, scaling can support two situations: (a) to deal with increased processing power, additional low-level services can be deployed automatically, and (b) to deal with increased number of users, additional high-level services can be instantiated to serve the interface. The first solution can be implemented with Grid middleware that incorporates an index service to register Grid services, so that the high-level services can discover additional low-level services deployed in the Cloud and take them into account. The second solution can be achieved by using load balancing at the interface nodes. For instance, Cloud computing is particularly useful in the implementation of the master-worker paradigm, as multiple workers can be spawned dynamically by creating more instances. These workers need only register themselves with the index service to become available to the master. This approach has the advantage of having to configure the worker's environment only once.

Let's now consider three examples of Clouds which rely heavily on virtualization of hardware, operating systems, and applications. We can say that Amazon's EC2 enables the use of different image instances and has a simple interface for managing them. A similar architecture can be adopted in computational science, primarily for abstracting the deployment and configuration of services. The S3 Cloud provides automatic provision of a namespace and intrinsic redundancy, providing the necessary abstraction of physical location and the reliability to handle scientific data. However, it does not offer a way to seamlessly attach metadata to it, making it an incomplete solution for the management of information and knowledge. This architecture should be improved to be able to participate in the e-Science field, but it could be considered as a step in the right direction given the simplicity of use. The architecture provided by the 3Tera Cloud is ideal for deploying high level services and creating new ones by aggregation. Its graphical interface offers an easy way to manage services. Similar graphical interfaces can be created to represent information and knowledge, as metadata provides a meaningful graphical representation. The user would have direct access only to this information, while lower level information would be dealt with by a brokering entity. This kind of *end-to-end* virtualization should be adopted by developers, to offer e-Scientists the possibility of simplified, yet useful interfaces, to enable them to pursue their goals more effectively.

In summary, the emergence of computing Clouds has already caused an impact on IT

industry. Many enterprises are deciding to make use of virtual data centres to facilitate infrastructure management and reduce the need for hardware maintenance. This type of *cyberinfrastructure* reduces the complexity involved in deployment of services, at the cost of losing flexibility with a narrower interface, a cost that many users may be willing to pay to deploy applications in a distributed environment.

The intrinsic virtualization gained by using image files offers support for redundancy and a more effective scaling to maximize usage of computing resources. A higher level of homogenization can be obtained by providing a standard operating system throughout the Cloud, and it can be made available in a distributed image file repository to facilitate the inclusion of new systems into the Cloud. However, the time spent to download and deploy an image file in a host remains a factor that needs to be considered. Image snapshots come to aid in this situation and can also be used to provide similar services with different computing profiles and Qualities of Service. The real-time management of virtual instance capabilities can produce a market of services that adapts dynamically to demand.

The concept of an ontology can be used to further describe the usage mode that characterizes a Cloud, and the interfaces that its entities should expose. A particular ontology used can be embedded in the image files, creating specialized services. This ontology-based classification will hep differentiate between different Cloud instances and deployments; however, the underlying implementation will be the same, constituted of the Grid fabric and the Grid middleware. The user would be aware of the Cloud selection according to the profile of the task, but the deployment and execution of the task should be transparent and easy, through the use of a graphical interface. One Cloud may represent a virtual organization, or many virtual organizations that agree to common semantics. The graphical interface may only deal with VMs. Management of the lower-level services may be hidden from the user and dealt with by a brokering service with VM instantiation and destroying capabilities, achieving automated scaling of the application processing power.

Existing Grid middleware can be deployed in a Cloud environment, as Grid services can run inside image instances, and multiple agents performing the same functions can be spawned from a single image easily. If an index service is available for discovering these services, the network configuration of each VM may be unknown, adding flexibility to the design. In this fashion, the master-worker paradigm can be easily deployed in a Cloud environment.

Cloud computing may be introduced in TeraGrid to encourage development of applications through a simplified interface. Clouds could potentially fill in the gap of the knowledge layer management with more abstract descriptions of content, marking a step towards the grid vision. Security in the cloud and applications that span over different Clouds remain an open issue, but the higher level of homogenization throughout the nodes and standardization of interfaces should improve development in these areas.

4. ACKNOWLEDGEMENTS

We would like to thank the authors for contributing papers on their research in middleware for Grid computing for this special issue, and thank all the reviewers for providing constructive reviews and in helping to shape this special issue. Finally we would like to thank Prof. Geoffrey Fox for providing us an opportunity to bring this special issue to the research community. Previous issues can be found at [CPE,04], [CPE,06], [CPE,07], [CPE,08] and [MGC,03], [MGC,04], [MGC,05], [MGC,06], [MGC,07] and [MGC,08].

5. REFERENCES

[CPE,04] Schulze, B., Nandkumar, R., Magedanz, T., Concurrency and Computation: Practice and Experience - Special Issue on Middleware for Grid Computing, Wiley Interscience; 16(5), 2004.

[CPE,06] Schulze, B., Nandkumar, R., Concurrency and Computation: Practice and Experience - Special Issue on Middleware for Grid Computing, Wiley Interscience; 18(6), 2006.

[CPE,07] Schulze, B., Coulson, G., Nandkumar, R., Henderson, P., Concurrency and Computation: Practice and Experience - Special Issue on Middleware for Grid Computing, Wiley Interscience; 19(14), 2007.

[CPE,08] Schulze, B., Nandkumar, R., Abramson, D., Buyya, R., Concurrency and Computation: Practice and Experience - Special Issue on Middleware for Grid Computing, Wiley Interscience; 20(9), 2008.

[MGC,03] Schulze, B., Nandkumar, R., Magedanz, T., Proceedings of the 1st International Workshop on Middleware for Grid Computing, PUC-Rio, pp. 169-266; Rio de Janeiro, 2003.

[MGC,04] Schulze, B., Nandkumar, R., Proceedings of the 2nd International Workshop on Middleware for Grid Computing, ACM International Conference Proceeding Series, Vol.76, 2004.

[MGC,05] Schulze, B., Nandkumar, R., Henderson, P., Proceedings of the 3rd International Workshop on Middleware for Grid Computing, ACM International Conference Proceeding Series;, Vol.117, 2005.

[MGC,06] Schulze, B., Nandkumar, R., Abramson, D., Buyya, R., Proceedings of the 4th International Workshop on Middleware for Grid Computing, ACM International Conference Proceeding Series;, Vol.194, 2006.

[MGC,07] Schulze, B., Rana, O., Meyer, J., Cirne, W., Proceedings of the 5th International Workshop on Middleware for Grid Computing, ACM International Conference Proceeding Series;, Vol.64, 2007.

[MGC,08] G V. Mc Evoy, B Schulze, Using clouds to address grid limitations, Proceedings of the 6th international workshop on Middleware for Grid Computing Leuven, Belgium, Article No. 11, Vol 72.