A Brief Survey on Resource Allocation in Service Oriented Grids

Daniel M. Batista and Nelson L. S. da Fonseca
Institute of Computing
State University of Campinas
Avenida Albert Einstein, 1251 – 13084-971 – Campinas – SP – Brazil
Email: {batista, nfonseca}@ic.unicamp.br

Abstract— **Grids are systems that involve coordinate resource sharing and problem solving in heterogeneous dynamic environments. Resource allocation is central to service provisioning in grids. In this paper, a brief survey of resource allocation of twelve existing systems is presented. Moreover, directions for future research are suggested.**

I. Introduction

Grids are systems that involve coordinated resource sharing and problem solving in heterogeneous dynamic environments to meet the needs of a generation of researchers requiring large amounts of bandwidth and more powerful computational resources [1]. Grids enable virtual organizations / computing environment which allows the offering of a variety of services. This has motivated the development of different grid systems for highly demanding services and applications. Although in its infancy, cooperative problem solving via grids has become a reality, and various areas [2] [3] [4] [5] have benefited from this novel technology.

Depending on the main type of service a grid offers, it can be classified as: computational grid, access grid, data grid or datacentric grid [6]. Computational grids provide high performance computing; access grids allow the access to a small number of specific resources; data grids store and move large data sets; and datacentric grids enable distributed repositories of data that cannot be stored in a single one.

Grids are typically composed of heterogeneous resources. The availability of these resources fluctuates during the execution of a grid application due to the lack of ownership of resources by the application. Moreover, grid applications typically demand large amount of resources and have diverse quality of service requirements. Thus, the ability to allocate resources and to cope with fluctuations of the availability of these resources is central to the provisioning of services in grid networks. Adaptive scheduling and dynamic scheduling [7] deal with such fluctuation by scheduling tasks composing an application just before the instant when the task should initiate. However, changes occurred during a task execution are not accounted for.

Existing grid systems differ by the way resources are allocated to furnish the desired quality of service. This paper briefly surveys aspects of resource allocation in grid systems. Moreover, it points out directions for future research on service provisioning in grid networks. This paper differs from previous surveys on resource allocation for grid networks by the larger number of systems surveyed as well as by the report on their performance. The survey in [8] focus in data grid, whereas the survey in [9] considers only applications described by workflows. Moreover, the survey in [10] neither takes into account heterogeneous resources nor describes approaches to deal with fluctuation of resource availability.

This paper is organized as follows: Section II introduces the problem of resource allocation in grid networks. Section III presents a survey of aspects of resource allocation in current systems. Section IV briefly compare the proposals surveyed. Finally, Section V points out some directions for future research.

II. Resource Allocation in Grid Networks

The allocation of resources to a grid application involves several actions. Scheduling maps the tasks composing an application to the available resources. Code migration transfers the code of a task and its computational context to a host where it will be executed. Data transmission transfers, between two remote hosts, the data needed by a task. Monitoring keeps track of resources availability and forecasting tries to predict the application performance. Figure 1 [11], illustrates a proposal for resource allocation by showing the flow of actions taken in a scheme based on monitoring. The scheme involves the following steps:

1-) Mapping the application description onto the graph describing the grid resources and production of a schedule for the beginning of task execution and data transfer; **2-)** Transfer of the codes and data to the hosts where the tasks will run. The execution of the tasks begins as soon as transfer is completed; **3-)** Monitoring the resources of the grid to detect any variation in the availability of hosts and links; **4-)** Gathering of the data collected in Step 3 and comparison with the scenario used for previous task scheduling. If no change is detected, periodic monitoring of the grid (Step 3) is continued; **5-)** Production of a new scheduling considering the current grid state. Only the unfinished tasks of the application must be scheduled; **6-)** Verification of whether the schedule derived is the same as the current one; **7-)** Comparison of the cost of the solution derived in Step 5 with the cost of the current solution. The cost of the solution derived in Step 5 should include the cost of migration of the tasks. If the predicted schedule length

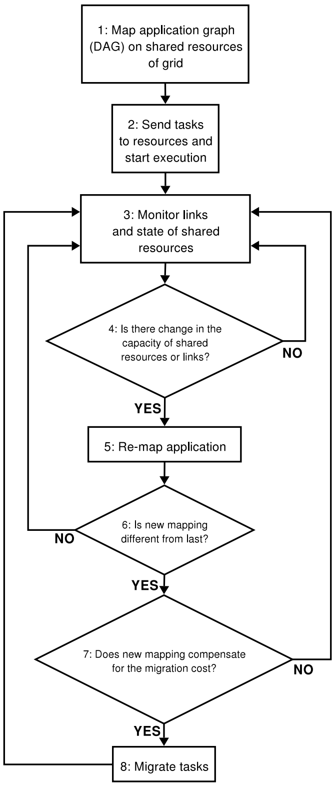
1: Map application graph (DAG) on shared resources of grid
2: Send tasks to resources and start execution
3: Monitor links and state of shared resources
4: Is there change in the capacity of shared resources or links?
NO
YES
5: Re-map application
6: Is new mapping different from last?
NO
YES
7: Does new mapping compensate for the migration cost?
NO
YES
8: Migrate tasks

Fig. 1. Procedure of Resources Engineering for Grids

produced by the new schedule is greater than that obtained by the current schedule, monitoring the grid resources (Step 3) shall continue. The cost of migration of a task involves the time needed to complete the execution, as well as the time to transfer the data. A task is only worth moving if a reduction in execution time of the entire application compensates for the cost; **8-)** Migration of tasks to the designated hosts on the basis of the most recent schedule.

III. Existing Grids

This section provides a brief overview of twelve existing grid systems and how resource allocation is pursued in each system.

A. NWS

The Network Weather Service (NWS) [12] is a distributed system used by several grid systems for producing short term performance predictions of computational and network resources. It involves monitoring and prediction but does not include (re)scheduling of tasks.

Current implementation of NWS collects measurements on the availability of CPU, TCP connection establishment time, end-to-end latency and available bandwidth. A set of different time series are applied to recent collected data and the one which produced the most accurate result is used to predict performance in the short term. The frequency at which probes are sent for measuring the grid resources is periodically adjusted in order to minimize the interference with the application. Moreover, it is sent with a frequency to produce representative measurement data sets. Sensors are organized in a hierarchical manner to optimize the generation of predicted values.

NWS works on small time scales and it does work well for long term predictions. Therefore, it is not proper to applications which takes hours to run. Besides that, uncertainties on applications demands are not accounted in prediction. Errors in estimating the execution time on the order of 25% were reported. Moreover, NWS produces graphics for bandwidth availability predictions in which the differentiation between predicted and measured values are not easy to evaluate.

B. GRACE

The GRid Architecture for Computational Economy (GRACE) [13] allocates resource on the basis of supply and demand dynamics. A resource broker deals with resource discovery and adaptation of resource allocation given changes in availability. It presents the grid to the users as a single computational system. The `Nimrod/G` resource broker is recommended and it was used in experiments conducted.

GRACE allows the monitoring of several parameters, including: CPU process power, memory, storage capacity and network bandwidth. Moreover, detailed measurements about software libraries access and memory pages can be generated.

Performance of GRACE was evaluated on the `EcoGrid`, a testbed which covers resources in four continents. CPU intensive applications were executed during a one-hour period. A reduction of the order of 30% on the cost due to intelligent utilization of resources was observed. However, the major drawback of this proposal is the lack of flexibility to adjust the resource allocation given changes of resource availability.

C. Cactus Worm

The Cactus Worm system [14] employs code migration and monitoring to allow applications to adapt its resource allocation when required performance level is not achieved. The Cactus Worm system employs the Globus Toolkit Monitoring and Discovery System (MDS) [15] to monitor the current resource status. The Condor middleware matchmaking algorithm schedules tasks to resources considering hosts memory and link bandwidth availability.

Migration can use intermediate nodes, allowing, in this way, data-transfer between non-connected hosts. However, intermediate nodes can become a bottleneck. Cactus Worm does not deal with uncertainties. Is was evaluated on the GrADS testbed, a grid composed of American universities. Experimental results show the advantage of Cactus Worm adaptive mechanism. However, there is no concluding data on the performance of the whole system.

D. Framework for Dynamic Grid Environments

The framework presented in [16] was target to promote changes when fault occurs as well as when resource availability increases. It was evaluated on TRGP testbed and a decrease of execution time of the order of 30% was observed.

Experiments were conducted involving only CPU intensive applications and, therefore, there is a need to consider data intensive applications to derive broader conclusions on its performance.

E. Migration Framework for Grids

The migration framework for grids presented in [17], adopts a policy which determines that migration should be pursued in case the gain in the execution time exceeds 30% of the estimated one, which is derived by using a pre-defined model. Link bandwidth and processing power are the major metrics used in this estimation. The NWS system is employed in this framework. It was was tested on GrADS testbed. An application run several times and a gain of 70% in the execution time was obtained. Similar gain values were found when the availability of resources increases. The major drawback of this proposal is the need of CPU time to produce the estimations given the model adopted.

F. Wren

The Wren system [18] uses both passive and active measurement. Passive measurement is carried out when applications are executing and active measurements when either no application is running or when the network load is low. The major contribution of this proposal is the introduction of active probing with low overhead. No mechanism for dealing with uncertainties and (re)scheduling were derived. It is our best knowledge that no experimental results are available.

G. GridWay

The GridWay system [19] utilizes the Globus middleware and the Framework for Dynamic Grid Environments (Subsection III-D) to build a system capable of adapting itself to environment changes, specially to CPU-intensive applications. Application requirements, bandwidth availability, migration overhead and processing power of potential new host are considered in the migration decision making process. Uncertainties on the application requirements and on estimations are not accounted in the decision process. Although not mandatory, the NWS is used but no forecasting of resource availability is carried out. Experiments with CPU-intensive applications pointed out a gain in execution time of about 15%. Manual migration, contrary to non-migration decisions, led to performance degradation.

H. G-QoSM

The Grid-QoS Management (G-QoSM) framework [20] allocates resources based on the Service Level Agreement made between users and providers. The grid is seen as capable of furnishing QoS and three classes similar to the classes of Internet Diffserv QoS framework are employed: the QoS guaranteed class, the QoS controlled-load class and the best effort one. Both performance degradation and incoming new services are adopted in the reallocation of resources. The Network Resource Manager (NRM) is employed to estimate the available bandwidth and the information gathering procedure of the Globus middleware is used to monitor the availability of processing power. Sampling of intra domain resources is more frequent than the sampling of inter domain resources. Although only link bandwidth and processing power are accounted for, G-QoSM is supposed to be able to monitor several QoS-related metrics. Both uncertainty on resource availability and on application demands are not considered. Moreover, G-QoSM does not involve forecasting.

I. VNET and VTTIF

In the system presented in [21], the grid network is seen as an overlay network and the VNET and the Virtual Topology and Traffic Inference Framework (VTTIF) mechanisms are used for managing and for defining the grid topology, respectively. Adaptation to resource availability is carried out by dynamically changing the overlay network topology which has an initial configuration of a star. VTTIF monitors the traffic pattern passively and measurement results dictate topology changes. Only the communication pattern is considered in rearranging the overlay topology. In [21], no information is provided about initial scheduling and mechanisms to deal with application and resource availability uncertainties. A grid to evaluate the concept was built. It was observed that gains varying from 20 to 50% in execution time were achieved after the changes in topology. These changes took on the average about one minute to complete.

J. GHS

The Grid Harvest Service (GHS) [22], as the NWS system (Subsection III-A), focuses on monitoring and prediction of the grid state. The purpose of GHS is to achieve higher levels of scalability and precision of predictions than the ones obtained by the NWS system, specially for applications which run for long periods. Passive and active monitoring techniques are used to evaluate the end-to-end bandwidth availability and neural networks are used to predict the available bandwidth and latency experienced. GHS re-schedules tasks to enhance the performance of applications. Two scheduling algorithms try to achieve such goal. One schedules tasks in order to minimize mean difference of execution time of tasks and the other one tries to maximize the number of tasks mapped onto a single resource. Experimental results point out the advantage of using GHS when compared to both the NWS system and to the AppLeS scheduler [23] in relation to the gain of execution time.

K. Workflow Based Approach (WBA)

The algorithm proposed in [24] is oriented to workflow based applications which are data intensive. Changes on resource availability trigger the re-scheduling of tasks but no migration of process context of processes is carried out. The schedule produced by the Task Based Approach guides the scheduling of tasks but it does not consider dependencies in the workflow. Schedulers take into account existing processing power and bandwidth. Simulations using the NS-2 simulator indicate the need to adopt mechanisms for dealing with uncertainties on the estimation of resource availability. Execution time half of

TABLE I

COMPARISON OF MECHANISMS IN TERMS OF SCHEDULING, MIGRATION, UNCERTAINTY HANDLING AND PERFORMANCE BASED EXPERIMENTS

Ref	Scheduling	Triggering	Reactions to change	Rescheduling	Treatment of uncertainty	Performance Results
[12]	–	Frequency of measurements i) adaptive to CPU ii) constant to network	–	–	Prediction of hosts state	Measurements in non-grid hosts and links
[13]	Not specified (Can use Nimrod/G)	Rules based on the application performance	Not specified	Not specified	–	Measurements on testbed
[14]	Requirements matching (Condor)	Rules based on the application performance	Migration (can use intermediate node)	Any host with better performance	–	Measurements on testbed
[16]	Greedy	i) New better resources ii) Performance degradation iii) Resource faults iv) Change in application requirement v) User decision	Migration or re-execution of task	Greedy	–	Measurements on testbed
[17]	Requirements matching	i) New better resources ii) Performance degradation	Migration if gain higher than 30%	Requirements matching	On applications	Measurements on testbed
[18]	–	Frequency of measurements	–	–	–	–
[19]	Requirements matching	i) New better resources	Migration if gain $>$ threshold	Requirements matching	–	Measurements on testbed
[20]	Requirements matching	i) Infeasibility of support QoS ii) Release of previously occupied resources iii) QoS degradation	Adjust of allocation (does not mention migration)	Requirements matching	–	Scenario not detailed
[21]	Not specified	Changes in traffic of the virtual topology	Topology adaptation	Not specified	Inference of virtual topology	Measurements on testbed
[22]	Host capacity based heuristics	Rules based on link and host status	Migration to idle resources	To first host found which support the requirements	Prediction of hosts state	Measurements in two grids
[24]	Workflow based heuristics	Not specified	Not specified	Workflow based heuristics	Mechanism not scalable	Simulations on NS-2
[25]	Cycle elimination and genetic algorithm	i) Execution fault ii) Performance degradation	Migration to better resources	Cycle elimination and genetic algorithm	Ameliorate negative impact of wrong decisions employing specific models	Measurements on testbed

those produced when the grid does not employs WBA were found.

L. Dynamic Scheduler of Scientific Workflows

The dynamic scheduler presented in [25] is able to schedule tasks described by graph with cycles, by eliminating cycles first. The scheduler uses genetic optimizations and can be parallelized. Uncertainties on the applications demands are assumed when predicting the execution times. Migration decisions are taken in case the observed execution time exceeds the predicted value. No monitoring is employed. Experimental results indicate that a 25% reduction on the execution time is possible.

IV. Comparison of Existing Proposals

Table I displays the main aspects of resource allocation of the proposals surveyed for service provisioning. Most of the proposals considers processing power and link bandwidth to (re)-schedule tasks. This is not sufficient for all types of grid applications, specially those requiring large amount of storage space and those requiring low end-to-end latencies such as interactive visualization.

Several proposals use the NWS system which was shown to be ineffective for applications requiring long execution times. Furthermore, existing systems are oriented to specific types of applications which implies on the lack of transparency to grid users. Moreover, only the G-QoSM system takes into account classes of services and QoS requirements in the resource allocation process. Besides that, uncertainties on applications demands are largely ignored in the proposals surveyed which can make ineffective resource allocation/scheduling.

V. Conclusions and Research Topics

The emerging technologies of grid networks will allow a diversity of highly demanding new applications and services which were not possible before. Resource allocation is the key to effective and efficient service provisioning. This paper surveyed the resource allocation schemes in twelve existing grid network proposals.

However, several challenges need to be overcome in order to make these systems transparent for service provisioning. One major challenge is to make grids general enough to efficiently accommodate a large spectrum of applications, releasing users from the need to understand the limits and capabilities of specific existing systems.

A critical aspect to all of the proposals presented is the lack of mechanism to deal with uncertainties on applications demands which can seriously degrade the system performance. Finally, there is an urgent need to adopt standard benchmarks, to compare existing proposals and to assess specific resource allocation mechanisms for service provisioning.

References

[1] I. Foster, "What is the Grid? A Three Point Checklist," *GRIDToday*, vol. 1, no. 6, Jul 2002. [Online]. Available: {http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf}

[2] W. Bethel, C. Siegerist, J. Shalf, P. Shetty, T. Jankun-Kelly, O. Kreylos, and K.-L. Ma, "VisPortal: Deploying Grid-Enabled Visualization Tools through a Web-Portal Interface," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-52940, Jun 2003.

[3] "iVDGL - International Virtual Data Grid Laboratory," 2006, http://www.ivdgl.org/. Accessed at 28 Jun 2007.

[4] "LCG - LHC Computing Grid Project," 2007, http://lcg.web.cern.ch/LCG/. Accessed at 28 Jun 2007.

[5] "Earth System Grid (ESG)," http://www.earthsystemgrid.org/. Accessed at 28 Jun 2007.

[6] D. B. Skillicorn, "Motivating Computational Grids," in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'02)*, May 2002, pp. 401–406.

[7] T. L. Casavant and J. G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Transactions on Software Engineering*, vol. 14, no. 2, pp. 141–154, Fev 1988.

[8] E. Laure, H. Stockinger, and K. Stockinger, "Performance Engineering in Data Grids," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2–4, pp. 171–191, 2005.

[9] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Grid Computing and Distributed Systems Laboratory, University of Melbourne, Tech. Rep. GRIDS-TR-2005-1, Mar 2005.

[10] K. Krauter, R. Buyya, and M. Maheswaran, "A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing," *Software: Practice and Experience (SPE)*, vol. 32,, no. 2, pp. 135–164, Fev 2002.

[11] D. M. Batista, N. L. S. da Fonseca, F. Granelli, and D. Kliazovich, "Self-Adjusting Grid Networks," in *Proceedings of the IEEE International Conference on Communications – ICC 2007*, Jun 2007.

[12] R. Wolski, N. T. Spring, and J. Hayes, "The Network Weather Service: a Distributed Resource Performance Forecasting Service for Metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5–6, pp. 757–768, 1999.

[13] R. Buyya, D. Abramson, and J. Giddy, "A Case for Economy Grid Architecture for Service Oriented Grid Computing," in *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, Abr 2001, pp. 776–790.

[14] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf, "The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment," *International Journal of High Performance Computing Applications*, vol. 15, no. 4, pp. 345–358, Nov. 2001.

[15] "About the Globus Toolkit," http://www.globus.org/toolkit/about.html. Acessed at 08 Ago 2007.

[16] E. Huedo, R. S. Montero, and I. M. Llrorent, "An Experimental Framework for Executing Applications in Dynamic Grid Environments," NASA Langley Research Center, Tech. Rep. 2002-43, 2002.

[17] S. S. Vadhiyar and J. J. Dongarra, "A Performance Oriented Migration Framework for the Grid," in *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'03)*, 2003, pp. 130–137.

[18] B. B. Lowekamp, "Combining Active and Passive Network Measurements to Build Scalable Monitoring Systems on the Grid," *SIGMETRICS Performation Evaluation Review*, vol. 30, no. 4, pp. 19–26, 2003.

[19] R. S. Montero, E. Huedo, and I. M. Llorente, "Grid Resource Selection for Opportunistic Job Migration," in *Proceedings of the 9th International Euro-Par Conference*. Springer Berlin / Heidelberg, 2003, pp. 366–373.

[20] R. Al-Ali, A. Hafid, O. Rana, and D. Walker, "QoS Adaptation in Service-Oriented Grids," in *Proceedings of the 1st International Workshop on Middleware for Grid Computing (MGC2003)*, 2003.

[21] A. I. Sundararaj, A. Gupta, and P. A. Dinda, "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*. New York, NY, USA: ACM Press, 2004, pp. 1–8.

[22] X. Sun and M. Wu, "GHS: A Performance System of Grid Computing," in *19th IEEE International Parallel and Distributed Processing Symposium*, 2005, http://doi.ieeecomputersociety.org/10.1109/IPDPS.2005.234. Accessed at 21/05/2006.

[23] F. Berman, R. Wolski, H. Casanova, W. W. Cirne, H. H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov, "Adaptive computing on the Grid using AppLeS," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 369–382, Apr 2003.

[24] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy, "Task Scheduling Strategies for Workflow-based Applications in Grids," in *IEEE International Symposium on Cluster Computing and Grids (CCGRID'05)*, vol. 2, May 2005, pp. 759–767.

[25] R. Prodan and T. Fahringer, "Dynamic Scheduling of Scientific Workflow Application on the Grid: a Case Study," in *SAC'05: Proceedings of the 2005 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 2005, pp. 687–694.