This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Self-Adjusting Grid Networks

Daniel M. Batista*, Nelson L. S. da Fonseca*, Fabrizio Granelli†, Dzmitry Kliazovich†
*Institute of Computing
State University of Campinas, Brazil
Email: {batista,nfonseca}@ic.unicamp.br
†DIT
University of Trento, Italy
Email: {granelli,klezovic}@dit.unitn.it

***Abstract*— This paper introduces a procedure called Traffic Engineering for grids for enabling grid networks to self-adjust to resource availability. The proposal is based on monitoring the state of resources and on task migration. It involves several layers of the Internet architecture. Experiments executed in NS-2 are used to illustrate the efficacy of the procedure proposed.**

I. Introduction

Grid Networks (Grids) were designed to provide a distributed computational infrastructure for advanced science and engineering [1] [2]. They involve coordinated resource sharing and problem solving in heterogeneous dynamic environments to meet the needs of a generation of research workers requiring large amounts of bandwidth and more powerful computational resources. Although in its infancy, cooperative problem solving via grids has become a reality, and various areas from aircraft engineering to bioinformatics have benefited from this novel technology. Grids are expected to evolve from pure research information processing to e-commerce, as has happened with the World Wide Web.

Resource sharing was one of the main objectives of Arpanet, the ancestor of Internet, developed in the 60's with network links used to access remote resources. In grids, however, network links are considered to be resources for allocation by grid applications. The network is seen as the bus of a virtual computing system used for the exchange of massive amount of data, potentially in the order of petabytes. Indeed, it has been observed that the exponential growth in the size of the data sets of current grid applications is rapidly approaching this.

In [3], a layered architecture for grid networks was proposed. Figure 1 illustrates the mapping of this architecture to the layers of that of the Internet. The Resource layer serves as an intermediary between the application and the infrastructure of the grid. One of its main functionalities is the evaluation of application requirements as well as the gathering of information about the status of shared resources. The Collective layer is responsible for the selection of resources to meet the application requirements, as well as the allocation of these resources. The combination of these two layers corresponds to the Application layer in Internet architecture. The Connectivity layer, which houses protocols for communication and data transfer, corresponds to a combination of the Transport and Internet layers of the architecture of the Internet. The Fabric layer is then responsible for communication at the link level.

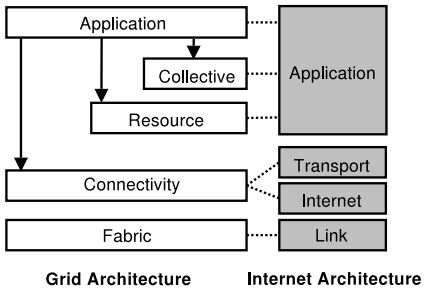


Fig. 1. Relationship between Grid and Internet architectures.

The lack of resource ownership by grid schedulers and fluctuations in resource availability require mechanisms which will enable grids to adjust themselves to cope with fluctuations. A sudden increase in link load can, for example, increase the time for the transfer of data between the computers where two tasks reside, thus leading to the necessity of relocating the tasks to a third computer. Furthermore, the lack of a central controller implies a need for self-adaptation. The ability to discover, monitor and manage the use of network resources is fundamental for the autonomous operation of a grid.

The contribution of this paper is a procedure for enabling grids to adapt themselves to fluctuation of resource availability. It involves task scheduling, resource monitoring and task migration, called Traffic Engineering for Grids in analogy to the Traffic Engineering procedure for networks [4] [5]. This procedure involves fundamental concepts of monitoring and self-adjustment as Traffic Engineering for networks. The distinct characteristic of the procedure introduced here is the consideration of network resource costs at all times of task mapping and migration. The proposed procedure involves different layers of the Internet architecture as well as the Grid architecture.

This paper is organized as following. Section II introduces a procedure for Traffic Engineering for Grids. Section III presents a summary of schedulers used in this paper. Numerical examples are shown in Section IV. Related works are shown in Section V and Section VI provides some conclusions.

1-4244-0353-7/07/$25.00 ©2007 IEEE

II. Proposal of Traffic Engineering

Key to the performance of grid applications is the choice of resources composing the virtual organization (computing system) to be used to execute the application. This choice is made by schedulers. Figure 2 illustrates the phases in the execution of a grid application. The bottom of the left side of Figure 2 illustrates the steps needed for scheduling. Determination of resource availability and application needs constitutes the first phase of the process. Applications are usually described as Direct Acyclic Graphs (DAG) in which nodes represent the tasks to be performed and arcs the dependence between two tasks. The weights of the arcs represent the amount of data to be exchanged by the tasks and the weights of the nodes the amount of processing required for a task. Figure 3 illustrates the DAG of a visualization application [6] that will be used to illustrate the Traffic Engineering proposal introduced here. The main question in scheduling is how to map the tasks to the network resources so that the execution time of application, usually called schedule length or makespan, is minimized.

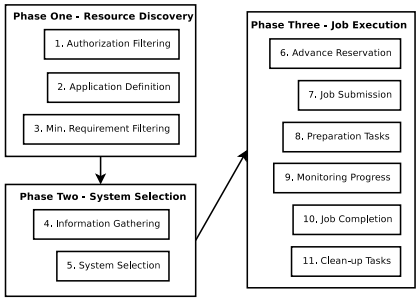


Fig. 2. A grid scheduling process

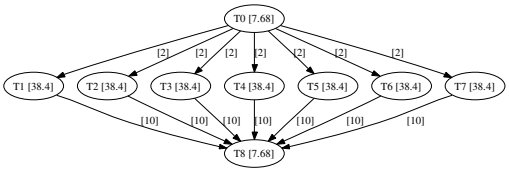


Fig. 3. A grid application DAG

After the tasks are allocated to hosts (grid nodes) according to a schedule, tasks are executed until all have been terminated. However, due to the lack of ownership of resources, their availability can change dynamically, and the original schedule may become sub-optimal. If, for instance, the load of a processor decreases, this processor may become an interesting choice for decreasing the execution time of the application. Therefore, if changes in resource availability lead to changes in the predicted schedule length, the schedule should be redefined so that a shorter schedule than those originally predicted can be achieved. Indeed, the proposal of applying traffic engineering to grids is to enable them to self-adjust to current resource availability.

In order to provide this capability, it is necessary to monitor the network resources periodically and perform code migration accordingly. The present proposal involves the following steps:

- *Step 1* Map the DAG describing the tasks that represent an application to the graph describing the network resources. Produce a schedule for the beginning of task execution and data transfer;
- *Step 2* Migrate the task codes and data to the nodes where the tasks will run. The execution of the tasks begins as soon as migration is completed;
- *Step 3* Monitor the resources of the grid to detect any variation in availability of resources, either decrease or increase;
- *Step 4* Gather the data collected in Step 3 and compare it to the scenario used for previous scheduling of tasks. If no change is detected, continue monitoring the grid periodically (Step 3);
- *Step 5* Derive a new DAG representing current computation and data transfer demand and produce a schedule for these tasks;
- *Step 6* Check whether the schedule derived is equal to the current one;
- *Step 7* Compare 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 tasks. If the predicted schedule length produced by the new schedule is greater than that obtained by the current schedule, continue monitoring the grid resources (Step 3). The cost of migration of a task involves the time needed to complete the execution, as well as the time to transfer data. A task is only worth moving if a reduction in execution time compensates for the cost;
- *Step 8* Migrate tasks to the designated processors on the basis of the most recent schedule and return to monitor the grid resources (Step 3).

Figure 4 shows a diagram portraying Traffic Engineering for Grids.

The mapping of tasks to grid nodes and their scheduling (Steps 1 and 5) demands efficient schedulers. Section III presents a set of schedulers [7] used in this paper.

Code and data migration can be performed using the GridFTP [8] (Steps 2 and 8). It is assumed in Step 8 that it is possible to resume the execution of an interrupted task. One method for the resume of task execution is provided in [9]. Techniques for monitoring the available bandwidth [10] [11] as well for predicting the network capacity with low computational overhead are available [12] [13] [14] and can be used in Step 3.

The schedulers can be used for both the initial scheduling of the Application (Step 1 of Traffic Engineering for Grids) and the re-scheduling of tasks due to changes in resource availability (Step 5). However, the tasks DAG to be used as input to the schedulers must reflect already effected computation. For each task in the DAG, a new task should be created with the same output arcs as the associated task. These two tasks will be connected by a link with the weight of the new task representing the amount of processing still to be done. Those

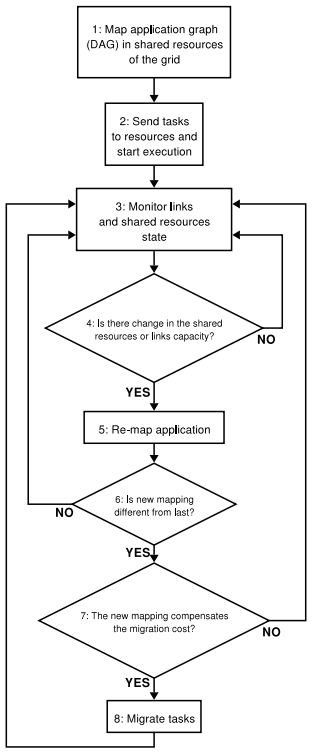


Fig. 4. A Flow Diagram of the Traffic Engineering for Grids Procedure

tasks which have already begun at the time of re-assignment evaluation are fixed at the hosts where they are processed.

Algorithm 1 redefines the application DAG so that the same scheduler can be used at both Steps 1 and 5. Such redefinition of the application DAG is necessary since between decision times about task migration, the computation progress and the change of demands need to be considered.

Algorithm 1 Tasks re-scheduling and migration

Input: Previous schedule; Description of current resource availability status

1: **for** each task i **do**
2: Assign the number of instructions already executed to the weight of task i.
3: Create a task i' and assign the backlog of instructions yet to be executed to the weight of the corresponding vertex.
4: Move all the outgoing arcs of the vertex i to the vertex i'.
5: Create an arc between vertex i and vertex i' with weight equivalent to the amount of bytes that need to be transferred in case task i' migrates.
6: Assign task i to the same host to which it was mapped previously to the re-scheduling decision.
7: **end for**
8: **for** each task k which either has already completed execution or is presently receiving data from others tasks **do**
9: Enforce that task k execute in the host it was executing previously to the re-scheduling decision step.
10: **end for**
11: Execute the scheduler with the new DAG.
12: Migrate all tasks which the new schedule indicates a different host than the one it was executing previously to the re-scheduling decision.

The self-adjusting capacity introduced by the Traffic Engineering procedure allows great flexibility and can be introduced in middlewares for grids such as [15] [16] [17]. Figure 5 illustrates the introduction of the traffic engineering procedure into the scheme proposed in [18] which is represented on both sides of the figure. Note that according to the procedure in [18], once a task is scheduled to a node it is executed until completion regardless of the fluctuation of resources availability. The central part of the figure is the procedure introduced here and it replaces the dashed part of the scheme in [18].

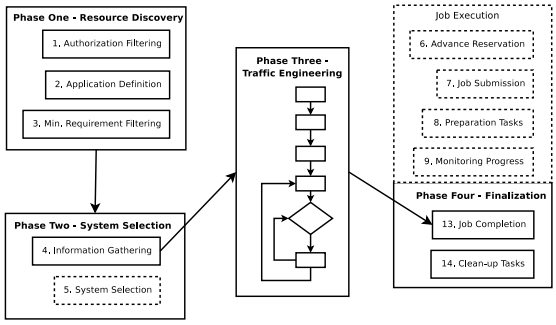


Fig. 5. Inclusion of Traffic Engineering for Grids in the process shown in Figure 2

III. Schedulers for the Traffic Engineering Procedure

Central to grid processing is the allocation of tasks to resources. In contrast to what happens in multiprocessor systems, scheduling in grids involves heterogeneous resources and irregular topologies. In heterogeneous systems, the scheduling problem is a NP-complete problem [19], and feasible solutions in real time require either heuristics or approximations.

The authors developed a set of schedulers which differ by the quality of schedules produced as well as by the computational demand. Six of the schedulers proposed use mixed linear programming (LP) and integer linear programming whereas the other two are heuristics randomized algorithms. The approach used in each scheduler is listed in Table I. These schedulers can be executed in parallel during finite period which duration represents the maximum tolerable time so that the application can be executed properly.

Scheduler	Approach
MLPCT	Mixed linear programming that considers a real time line
CT-IRR	MLPCT with relaxation of integer variables. The LP is executed n times
CT-RR	MLPCT with relaxation of integer variables. The LP is executed 1 time
ILPDT	Integer linear programming that considers a integer time line
DT-IRR	ILPDT with relaxation of integer variables. The LP is executed n times
DT-RR	ILPDT with relaxation of integer variables. The LP is executed 1 time
DG	Drawing based in grid- and DAG-aware probabilities
RDU	Drawing based in uniform probabilities

TABLE I

Task schedulers (n is the number of tasks in the DAG)

The performance of these schedulers were exhaustively addressed using various application DAGs and network topologies for both scheduling and re-scheduling scenarios. The results of this evaluation indicate that the DT-RR presents the best compromise between quality of scheduling and computation demand. However, the schedulers MPLCT and ILPDT produce the best schedules under loose time restrictions.

These schedulers can be used in Steps 1 and 5 of the traffic engineering procedure proposed in this paper.

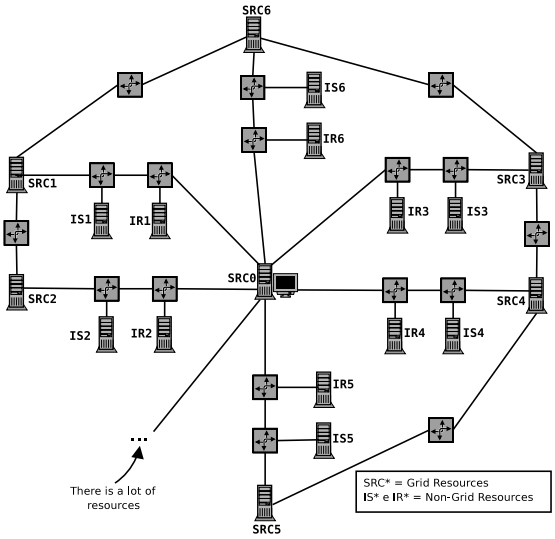
IV. Examples of the use of the Traffic Engineering Procedure

This section illustrates the use of Traffic Engineering procedure to minimize the execution time of grid applications (schedule length). A simulator, called Gridsim-NS, developed at the University of Trento, was used to generate examples. Gridsim-NS is actually a module incorporated into the widely used NS-2 [20] simulator. Gridsim-NS receives as input a task DAG and allows users to define a schedule to be employed for the input DAG. In the following examples, the schedules were produced by the schedulers presented in the Section III.

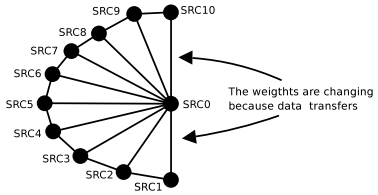
The application is the one described in Figure 3, whereas the grid is illustrated in Figure 6. The arc weights in the DAG represent the amount of data to transfer in GigaBytes, whereas the node weights represent the amount of instructions in a 10^{12} scale. Figure 6(a) shows the network topology whereas Figure 6(b), the grid nodes. The network has 34 nodes arranged around a central node, *SRC0* (This topology represents partially CERN's LHC Computing Grid[21]). The available processing rate of this host, node, is $1600MIPS$, whereas all others can process at the rate of $8000MIPS$. The link connecting *SRC0* to the other nodes has the capacity of $100Mbps$, whereas all the other links are limited to $33.33Mbps$. All links involve a propagation delay of $0.2ms$.

In the first example, the application (Figure 3) is mapped using MLPCT. The resulting mapping is $0 \rightarrow$ *SRC0*, $1 \rightarrow$ *SRC2*, $2 \rightarrow$ *SRC5*, $3 \rightarrow$ *SRC8*, $4 \rightarrow$ *SRC4*, $5 \rightarrow$ *SRC1*, $6 \rightarrow$ *SRC9*, $7 \rightarrow$ *SRC10*, $8 \rightarrow$ *SRC0*. Similar mapping could have involved other nodes, since the topology is symmetrical. In the actual schedule derived, Tasks 1 to 7 start running at the time of $82.66min$, whereas task number 8 starts running at $175.96min$ and finishes at $255.96min$.

In a second experiment, the same scenario and initial mapping were used. However, at $90min$, UDP streams with a rate of $90Mbps$ were added as interfering traffic between nodes *SRC2* and *SRC0* and between nodes *SRC5* and *SRC0*. Monitoring the resources of the grid was carried out every 120 minutes. Thus, at the time of the first data collection, the need to re-evaluate the current schedule had become evident. At that time, the DAG for the remaining tasks was modified to the one shown in Figure 7. For that DAG, both MLPCT and ILPDT produced the same schedule. Since the cost involved in task migration includes that of time needed to complete the execution, as well as that required to transfer data, a task is worth moving only if a reduction in time of execution will compensate for this cost. The new schedule determined that Tasks 1 and 2 should be migrated to nodes *SRC3* and *SRC6*, respectively. These migrations were designed to avoid the interfering traffic for the transfer of $10GB$ of data to task 8. The new execution time was 281 *min*. If the tasks had not migrated, the execution time would have been 358 *min*, i.e., an increase of about 27.4%. Figures 8 and 9 show, respectively, the time of execution of Task 1 and the Round Trip Time



(a) Network Topology



(b) Virtual Organization Graph

Fig. 6. Grid used in the examples

(RTT) between *SRC1* and *SRC2*. These figures illustrate task migration; it can be seen that between the times 120 *min* and 150 *min*, no processing activity took place in either of the two nodes.

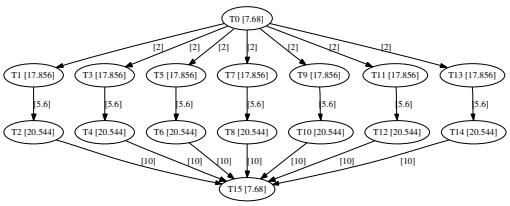
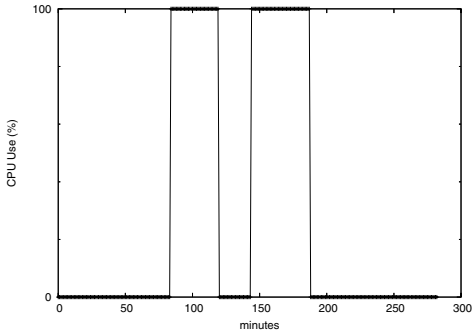
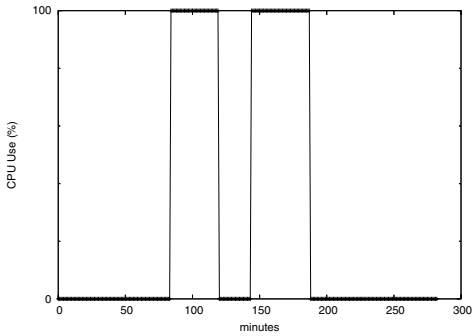


Fig. 7. DAG for migration at $120min$

In another set of experiments, resources were added to the grid. Such additions are not necessarily due to the acquisition of new resources, as they may be due to the release of resources by other applications. Figure 10 illustrates the addition of the node *SRC16*; the link capacity joining it to node *SRC6* is 1Gbps, with an available processing rate of $8000MIPS$. Similar nodes were also added to nodes *SRC1* to *SRC10*. With this extra resource, the execution time decreases to 247 min. This example shows that task migration should not only

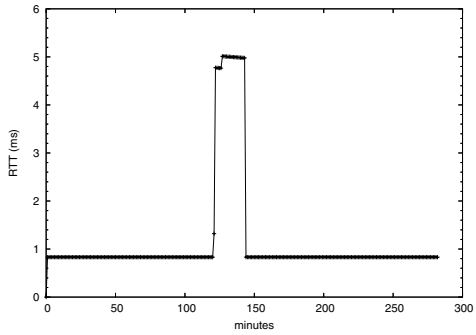


(a) Use of CPU for Task 1

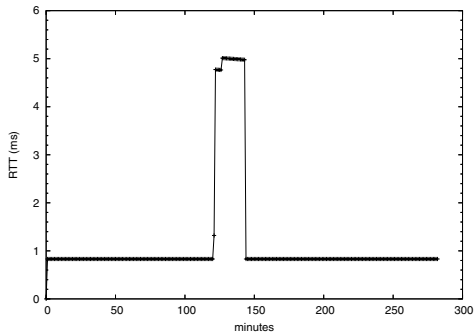


(b) Use of CPU for Task 2

Fig. 8. Use of CPU for Tasks 1 and 2



(a) RTT between *SRC1* and *SRC2*



(b) RTT between *SRC5* and *SRC6*

Fig. 9. RTT between *SRC1* and *SRC2* as well as between *SRC5* and *SRC6*

be investigated under conditions of a shortage of resources, but also whenever increased resources become available. If, for example, the processing rate available were $4000MIPS$, migration would not be advisable since, execution time would have increased to 291 min if migration were carried out.

One of the key issues involved in the Traffic Engineering procedure is the frequency of re-evaluation of the adequacy of the schedule under modified resource constraint. To get an idea of the importance of the frequency of this procedure, various simulations were carried out. A source of interfering traffic (60 $Mbps$) was introduced to the same links as in the previous example. Both MLPCT and ILPDT were used for the experiments. First, a simulation with no task migration was run; execution time was 279 min. Then, the recommendations of the scheduler were followed. Table II shows the execution time required when task migration is undertaken. It is clear that the frequency of evaluation plays a major role in the execution time. If a long period between changes in resources availability and the decision to migrate a tasks occurs, computation may have progressed to a point in which migration would no longer be an interesting option. Moreover, it can be seen that the ILPDT may produce schedules which yield longer execution times than those where no migration is pursued, as can be in the results when intervals of 120 min and 130 min were used. Such imprecision is critical when approaching the "ideal" time for re-evaluation due to the approximations introduced by time discretization. In fact, the ideal frequency for re-evaluation is system dependent, since it is influenced by the frequency of changes in the resource pool.

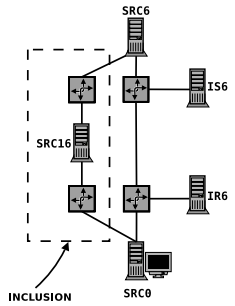


Fig. 10. Inclusion of new resource linked to $SRC6$

V. Related Work

Various techniques for monitoring and performance prediction have been employed for systems such as that of the Network Weather Service (NWS) [14], which uses active monitoring techniques, as well as temporal series, to predict performance. One distinct characteristic of the NWS system is its hierarchical monitoring approach. Applications such as those supported by NWS require performance feedback in short periods of time, typically in the order of minutes. Another system for applications which run for long periods is the Grid Harvest Service (GHS) [13] which is more scalable than NWS. In GHS, performance prediction is carried out by neural networks and these predictions are employed to determine task migration. A different monitoring system used in the Wren systems was introduced in [11]; this adopts either active or passive monitoring techniques, depending on the

Interval	MLPCT	ILPDT
100	269 (migration)	269 (migration)
110	275 (migration)	275 (migration)
120	276 (no migration)	281 (migration)
130	276 (no migration)	287 (migration)
140	276 (no migration)	276 (no migration)
150	276 (no migration)	276 (no migration)

TABLE II

EXECUTION TIMES AS A FUNCTION OF MONITORING INTERVAL DURATION (MINUTES)

network load. All these proposals for monitoring status of resources can be incorporated in Steps 3 and 4 of the Traffic Engineering procedure introduced in Section II. However, the prediction of performance in Traffic Engineering for Grids involves schedulers based on optimization for determining potential re-configuration of a grid.

Several self-adjusting systems based on monitoring and task migration have been proposed [16] [17] [15] in the literature. In [16] task migration is pursued whenever the service level agreement between a user and a network provider is not being supported. Scheduling is based on a high-level definition of requirements and NWS is used for monitoring. This system differs from the one proposed in this paper by task migration which is carried out in two steps. First, a task is migrated to an intermediate storage node and then to its final definition. The drawback of this approach is the network bottleneck introduced and the enlargement of migration period. The procedure presented in [17] also migrates tasks whenever resource availability changes but only disconnection of resources is considered. In this procedure, scheduling is based on a greedy algorithm without the consideration of data dependencies. Moreover, the cost of task migration is not considered. The proposal introduced in this paper differ from the one in [17] by the potential resource changes considered. The scheme introduced in [15] can also be used to migrate tasks if the cost gains of migration exceed a certain threshold value, although the authors acknowledge that setting this threshold value is quite difficult task which may prevent the rapid execution of certain tasks.

VI. CONCLUSIONS

Grid networks can accommodate a new generation of users with high computational and data transfer demands. Although several grid systems already exist, this technology is still in its infancy. One of the major challenges of grids networks is the fluctuation in availability of resources which has a definite impact on the performance of an application. Enabling grid systems for self-adjustment in response to changing scenarios is crucial for autonomy and will facilitate their use. This paper has introduced a multi layer traffic engineering approach for the empowerment of grids in this direction. The effectiveness of this new procedure has been illustrated in several simulation experiments involving various changes in the simulated grid. In the future, the dynamic determination of the duration of intervals for re-evaluation of schedule needs to be pursued. Moreover, traffic engineering proposed here should be introduced into existing systems to cope with uncertainties on both bandwidth estimation and estimation of processing power.

REFERENCES

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

[2] H. Casanova, "Distributed Computing Research Issues in Grid Computing," *SIGACT News*, vol. 33, no. 3, pp. 50–70, 2002.

[3] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *The International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, Fall 2001.

[4] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "RFC 3272: Overview and Principles of Internet Traffic Engineering," 2002.

[5] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "RFC 2702: Requirements for Traffic Engineering Over MPLS," 1999.

[6] L. Renambot, T. van der Schaaf, H. E. Bal, D. Germans, and H. J. W. Spoelder, "Griz: Experience with Remote Visualization over an Optical Grid." *Future Generation Computer Systems*, vol. 19, no. 6, pp. 871–882, 2003.

[7] D. M. Batista, N. L. S. da Fonseca, and F. K. Miyazawa, "Escalonadores de Tarefas em Grades," in *Anais do XXVI Congresso da Sociedade Brasileira de Computação – V Wperformance*, Jul 2006, pp. 73–92.

[8] M. Cannataro, C. Mastroianni, D. Talia, and P. Trunfio, "Evaluating and Enhancing the Use of the GridFTP Protocol for Efficient Data Transfer on the Grid," *Lecture Notes in Computer Science*, vol. 2840, pp. 619 – 628, Jan 2003.

[9] A. Roy and M. Livny, "Condor and Preemptive Resume Scheduling," in *Grid Resource Management: State of the Art and Future Trends (1st Edition)*. Springer, 2003, pp. 135–144.

[10] F. Montesino-Pouzols, "Comparative Analysis of Active Bandwidth Estimation Tools," *Lecture Notes in Computer Science*, vol. 3015, pp. 175 – 184, May 2004.

[11] 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.

[12] J. M. Schopf and L. Yang, "Using Predicted Variance for Conservative Scheduling on Shared Resources," in *Grid Resource Management: State of the Art and Future Trends (1st edition)*. Springer, 2003, pp. 215–236.

[13] X. Sun and M. Wu, "GHS: A Performance System of Grid Computing." in *Proc. 19th IEEE International Parallel and Distributed Processing Symposium*, Apr 2005, pp. 228a–228a.

[14] 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.

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

[16] 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.

[17] 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. [Online]. Available: {http://techreports.larc.nasa.gov/ltrs/PDF/2002/cr/icase-2002-43.pdf}

[18] J. M. Schopf, "Ten Actions when Grid Scheduling," in *Grid Resource Management: State of the Art and Future Trends (1st edition)*. Springer, 2003, pp. 15–23.

[19] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization – Algorithms and Complexity*. Dover Publications, 1998, pp. 363–366.

[20] K. Fall and K. Varadhan, "The ns Manual," UC Berkeley, The VINT Project, Tech. Rep., Aug 2002. [Online]. Available: {http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf}

[21] "LCG - LHC Computing Grid Project," European Organization for Nuclear Research, 2006, http://lcg.web.cern.ch/LCG/. Accessed on 06/09/2006. [Online]. Available: {http://lcg.web.cern.ch/LCG/}