Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 1713 – 1722

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

Impact of urgent computing on resource management policies, schedules and resources utilization

Krzysztof Kurowski[a,*], Ariel Oleksiak[a], Wojciech Piatek[a], Jan Węglarz[a,b]

[a] *Poznan Supercomputing and Networking Center, Poznan, Poland*
[b] *Institute of Computer Science, Poznan University of Technology, Poland*

Abstract

In general resource management is often defined as a process of identifying application requirements, matching compute resources to jobs, allocating those resources, scheduling and monitoring them over time to run jobs as efficiently as possible. In case of urgent computing, additionally, in this process we have to deal with a set of specific event-driven or time-critical application requirements that must be satisfied by a resource provider to guarantee the immediate access to compute resources. There is no question that capabilities supporting urgent computing are required in distributed high performance and high throughput computing infrastructures like Grids. Nevertheless, it is difficult to predict if and how urgent jobs will disturb scheduling policies as well as coexist with less critical waiting and running jobs submitted by other end-users. To date many approaches have been proposed to deal with urgent computing, for instance based on preemption or high job priorities and recently based on advance reservations. However, there is a lack of comprehensive analysis of the impact of urgent computing on existing schedulers and their efficiency. Moreover, many resource providers use a single evaluation criterion - utilization or average load of compute resources. In fact the utilization of compute resources as a metric is in contrast to other evaluation criteria relevant for urgent jobs, e.g. waiting time, response time, or mean flow time. These additional criteria in our opinion can be treated as good metrics for urgent computing taking into account the end-users perspective rather than the resource provider centric view. As we demonstrate in this paper, the problem is not trivial and it requires workload analyses. Finally, in this paper we successfully used GSSIM simulator to run various experiments and benchmarks for different scheduling strategies where urgent jobs were additionally considered.

Keywords: urgent computing, on-demand computing, parallel job scheduling, hierarchical scheduling, HPC, Grids

1. Introduction

Large-scale, time-critical and event-driven application scenarios have been pushing demands for new capabilities available in distributed computing infrastructures to deal with urgent computing. In a nutshell, end-users submitting urgent jobs to remote compute resources do not expect that their jobs will waste time waiting in batch queues until compute resources become available. On the other hand, batch queues defined within local schedulers are core mechanisms for resource providers to manage and share compute resources among multiple end-users and their jobs. The

*Corresponding author
Email address: krzysztof.kurowski@man.poznan.pl (Krzysztof Kurowski)

doi:10.1016/j.procs.2012.04.189

problem is even more challenging in Grids linking together large collections of geographically distributed resource providers that usually span several organizations to share a variety of resources dynamically, depending on their availability, capability, end-users requirements, and any other predefined rules set by local resources owners. Thus, such computing environments could consist of independent resource providers, where each of them may imply different resource management policies and rules for responding to urgent computation. One should also note that most of Grids are organized hierarchically - that is, there is at least one grid scheduler or broker assigning jobs to a local schedulers without being able to fully control what happens next at the local level within a local scheduler. Taking into account the complexity of a two-level hierarchical structure of schedulers together with variety of grid middleware as well as constraints based on resources utilization, different levels of requested urgencies, end-users behavior and acceptance it is difficult or even impossible to come up with one unified resource management strategy for urgent computing.

Additionally, according to analyses of different workloads collected in production HPC systems or clusters, local schedulers are often configured to use a single evaluation criterion forced by a local resource provider - utilization of compute resources. As the utilization of compute resource criterion is in fact contradictory to other evaluation criteria relevant for end-users submitting urgent jobs, e.g. waiting time or a number of canceled jobs, the problem is even more difficult as it is multi-criteria by nature. On the other hand, in many production setups some end-users may have more influence on what compute resources are provided locally and how they are used than more objective measures of performance or hardware costs. We should also emphasize dynamic characteristics of end-users behaviors and job patterns. Preferences of end-users may vary in time and are correlated with resource requirements of submitted jobs. For example, small interactive or parameter sweep type jobs require often the immediate access to resources, whereas end-users submitting large-scale and long calculations can tolerate delays. Consequently, in our opinion both end-users and resource providers should be treated as stakeholders within the resource management process, in particular in case of urgent computing. Then, the resource management process should led to the improvement of all stakeholders satisfaction that can be measured and aggregated by multiple criteria.

Therefore, in this paper we show how multi-criteria approaches can be applied to validate the impact of different scheduling policies and their hierarchical configurations on urgent computing. We show how to model different and hierarchical scheduling setups in Grids where urgent jobs may appear irregularly. Then, we discuss achieved results in the light of key evaluation criteria relevant for urgent jobs and their users, in particular resources utilization vs. waiting time of both typical and urgent jobs. Moreover, based on analysis and performed experiments using real workloads we compare commonly used scheduling strategies and show if and how urgent jobs will disturb scheduling policies as well as coexist with less critical waiting and running jobs submitted by other end-users.

The rest of this paper is organized as follows. In Section 2 we introduce leading grid initiatives together with references to main capabilities introduced so far to deal with urgent computing. Section 3 presents briefly selected real workloads for basic analyses of job patterns as well as end-users preferences. The considered resource management problem is formulated in Section 4 together with the main assumptions. Section 5 describes a set of experiments we have performed to measure the impact of urgent jobs on resource management using high job priorities and advance reservations in local schedulers. Conclusions and future work are presented in Section 6.

2. Related work

Grid computing has become a popular way of providing distributed high performance, high throughput and parallel computing for advanced science and engineering based on interconnected networks of supercomputing centers, campuses and emerging service-oriented, Internet-based technologies. In other words, the Internet provides uniform access to World Wide Web resources, and Grids or recently Clouds provide their users with an access to all kinds of compute resources in a dynamic, geographically distributed fashion. Existing grid environments, such as PL-Grid in Poland [4], TeraGrid [6] in the United States, LHC/CERN in Switzerland [1], NorduGrid in Sweden [12], Grid5000 in France [5] to cite a few of them, provide thousands of compute resources, and offer similar or even better facilities for end-users when compared to supercomputers. Unfortunately, many of existing Grids offer relatively simple remote job submission, brokering and monitoring capabilities, whereas others support advanced capabilities that can be used for urgent computing, such as high job priorities or advance reservation. A set of interesting application scenarios and new requirements for capabilities that should be available for end-users have been collected in [26].

Probably the first complete high job priorities strategy for urgent computing has been proposed and prototyped in [2]. Then, a new system known as Special PRiority and Urgent Computing Environment (SPRUCE) framework was

successfully implemented, deployed and supported by some resource providers involved in TeraGrid [33]. Resource providers that wanted to support urgent jobs in SPRUCE had to reconfigure their local schedulers to support the elevated priority policies. In principle, the idea behind SPRUCE is relatively simply and does not require from end-users too many efforts. Only some basic extensions are required in case of urgent job submission requests that will then trigger elevated priority policies implemented in local schedulers. End-users have add to their submission requests a color-coded urgency token: critical (red), high (orange), and important (yellow), signifying the relative importance of the urgent job. Color-coded urgency tokens are typically created with a maximum urgency level attribute, giving resource providers the ability to limit the policies available to a given set of end-users. From the resource management perspective, one should note that local schedulers configured to support SPRUCE still use batch queues to manage jobs that can not be immediately allocated. However, to reduce the waiting time of urgent jobs in batch queues scheduling policies are moving the urgent job to the top of the queue or killing jobs that are currently allocated in order to free up the necessary compute resources for the urgent job.

As we demonstrate in this paper, an alternative approach to urgent computing can be based on a relatively new feature - advance reservation supported today by most of local schedulers, including Platform's Load Sharing Facility (Platform LSF) [28], PBS Pro/Torque [30], Maui [31], Sun Grid Engine (SGE) [29] and Load Leveler [32]. In fact, appropriate job managers to listed local schedulers have been also successfully modified to support SPRUCE [33]. The quality of service, especially related to a job start time in case of urgent computing, can be ensured by applying the advance reservation functionality. The advance reservation feature allows end-users to reserve a certain number of compute resources over time and preempt if needed waiting and running jobs to free up a number of compute resources required for the urgent job. Many resource providers believe that advance reservations may significantly deteriorate waiting time of jobs and the overall resource utilization, especially in hierarchical scheduling setups. However, as we demonstrated in [20], new online scheduling policies and generic advices could reduce significantly negative impact of advance reservations on a schedule quality. A comprehensive experimental analysis was presented to show the influence of advance reservations on multiple criteria, such as: resource utilization, mean waiting time, mean flow time, and mean tardiness.

Only a few groups worldwide are working on multicriteria approaches applied for grid environments, although it seems straightforward to consider grid resource management and scheduling as a multicriteria scheduling problem. In [10][11] the authors consider the bi-criteria algorithm for scheduling jobs on clusters of resources. Two pre-selected criteria are used, namely makespan and average completion time, for moldable jobs scheduling. However, there were not addressing issues related to advance reservation and urgent computing. In our opinion, additional evaluation criteria, in particular wait time and a number of preempted jobs are significant for urgent computing. Thus, one of the key motivations for presented research in this paper was to evaluate experimentally various interesting extensions to hierarchical scheduling strategies discussed in [20] using advance reservation capabilities for urgent jobs. Moreover, advance reservation capabilities are provided and supported today by resource providers using QosCosGrid middleware [18], [21] in the mentioned PL-Grid environment [4]. Thus, our another goal in this paper was to verify the applicability of advance reservation features for urgent computing in practical setups. One should note that advance reservation features supported by resource providers can be easily exposed to selected end-users that may occasionally require an urgent access to compute resources. In the next section we formulate the problem of resource management where urgent jobs may appear irregularly together with various basic assumptions for further analysis and experiments.

3. Workloads

It is generally agreed that the best approach to evaluate scheduling strategies is to apply realistic job mixes, obtained from large-scale supercomputing sites [25] [22] [27]. Probably the first detailed model of parallel workloads was proposed by Feitelson and then described in [7]. Since then, many workload data have been collected [34], analyzed and modeled [24], [25]. Real traces on production Grids are being collected [35] and recently have been analyzed, for instance in [13], [22]. Moreover, many analyses and experiments based on mentioned real workloads, representing both parallel and grid environments, indicate that the choice of workload trace alone did not affect the relative performance of the selected scheduling algorithms. Almost all workloads (real or synthetic, across sites, and for different time periods at the same time) ranked the algorithms in the same order from best to worst with respect to slowdown ratio and system utilization. However, many authors warned that researchers will need to be very careful about the use of real workloads, recommending that a workload derived from one system should not

be used to evaluate another [8], [9]. Real workload models could be affected by local site policies and constraints, polluted by bugs data, etc. Nevertheless, the access to real grid workloads is not straightforward even if a number of big Grid testbeds (e.g. Grid5000, EGEE, DAS2) have made accounting data available to the public [35]. Existing grid workloads usually come from single and often independent local clusters, collected under specific conditions, and do not contain information about many parameters that could affect the performance evaluation [23].

The standard workload format (SWF) is still considered as a main reference repository for researchers. The main aim of the common format was to ease the use of real workload logs and models. With it, programs that analyze workloads or simulate system scheduling need only be able to parse a single format and can be applied to multiple workloads. Real workloads sorted according to period of time when workloads were collected are shown in Table 1. Presented supercomputing systems denote respectively, San-Diego Supercomputer Center (SDSC SP2), High-Performance Computing Center North (HPC2N) in Sweden and Lawrence Livermore National Lab (LLNL) in the United States.

Table 1: Selected real workloads used for analysis and urgent computing experiments

Source	Duration	Jobs	Resources (CPUs)	File
SDSC SP2	1998-2001	73496	128	SDSC-SP2-1998-3.1-cln.swf
HPC2N	2002-2006	527371	240	HPC2N-2002-1.1-cln.swf
LLNL Atlas	2006-2007	60332	9216	LLNL-Atlas-2006-1.1-cln.swf

From the end-user perspective it is important not only to measure response time, but also to compare it with the job runtime. Thus, another criterion used in many performance analysis by scheduler is called slowdown. Slowdown as a metric captures the notion of fairness, as low slowdown implies that jobs are delayed in proportion to their size, hence smaller jobs are delayed less and large jobs are delayed proportionately more. As it is demonstrated in Fig. 1 in many systems end-users accept a relatively long system response time. We measured slowdown in all three of jobs SDSC SP2, HPC2N and LNLL Atlas systems for a relatively big fraction of jobs. The reader should notice that there are many peak frequencies presented in three histograms in Fig. 1 what might suggest common patterns during the job submission process, e.g. end-users may accept jobs delays in proportion to their job-size.

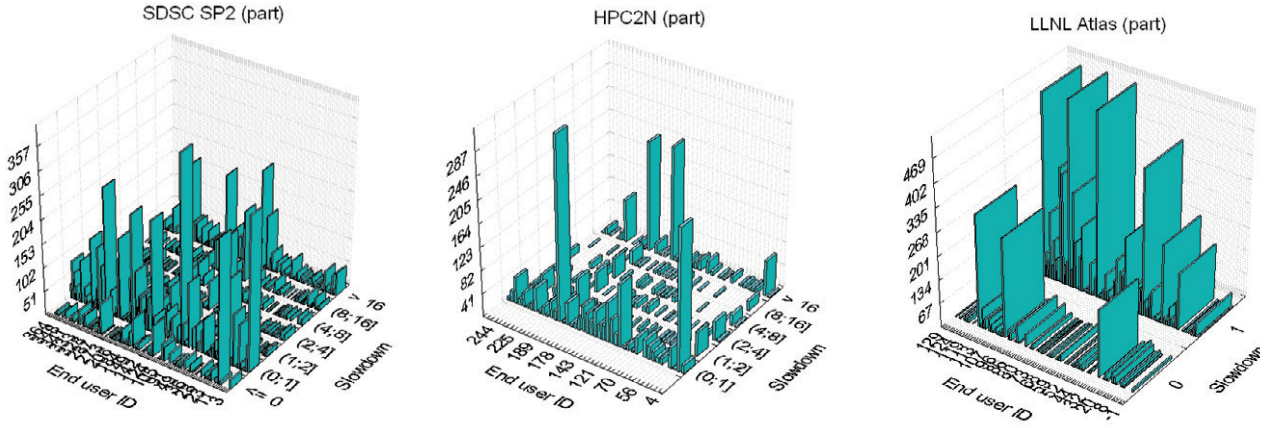

Figure 1: Slowdown of jobs in SDSC SP2, HPC2N and LNLL Atlas systems

A high percentage of job waiting time periods in SDSC SP2, HPC2N and LNLL Atlas systems is presented in Fig. 2. As different end-users run various computing simulations over different time periods and formulate a variety of resource requirements, in particular a number of required computing resources, their jobs end up in queues waiting before the execution as there is no available compute resources at job submission. Obviously, there are some end-users more active than others in terms of a number of generated job requests to selected systems and this observation is also

reflected in Fig. 2. In the context of urgent computing we should also emphasize the fact that in real workloads there are many typical job patterns and end-users behaviors. In fact Fig. 2 also shows that different end-users were using computing clusters or supercomputers in an urgent way by accepting only relatively small task waiting time periods, whereas there were also end-users tolerating longer task waiting time periods. Unfortunately, there is no information available in the analysed workload regarding priorities of jobs and end-users. According to this histogram jobs waiting time in the LNLL Atlas workload was zero with a high percentage of jobs with errors during their submission. The main reason that jobs were executed in the LNLL Atlas system immediately was probably its low utilization. Therefore, for further analysis and urgent computing experiments we selected the SDSC SP2 workload which was highly utilized.

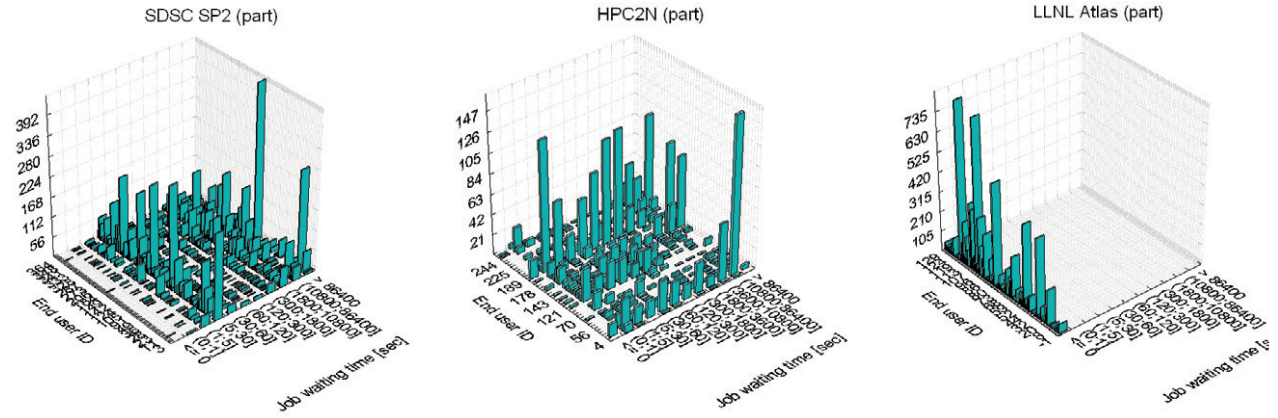

Figure 2: A high percentage of job waiting time periods in SDSC SP2, HPC2N and LNLL Atlas systems

4. Problem formulation and assumptions

In this paper we consider the classical batch type compute resources. Compute resources are not subject to individual access and usage policies, because they are provided by different resource owners. There is a hierarchical scheduling architecture with the following tightly connected two levels:

- a local scheduler located on top of the operating system that manages one cluster of nodes (each node has one or many CPUs) and performs scheduling for single and parallel (a job that requires many CPUs at the same time) jobs submitted for execution within a cluster or supercomputer,
- a Grid broker located on top of local schedulers and connected via middleware services to many administrative domains. The Grid broker performs scheduling for single, parallel and cross-domain jobs in many administrative domains.

We assume that the Grid broker has the knowledge about all available jobs, descriptions and characteristics of resource providers, however, it does not know when, what and how many jobs will arrive in the future. In other words the Grid broker is located within an interoperability layer and it acts as a 'mediator' among end-users and access geographically distributed compute resources managed by local schedulers. Each local scheduler generates its own schedule according to a local policy. We assume the following policies are considered for urgent computing at the level of local scheduler: next-to-run, preemption and preemption with advance reservation. In a nutshell, the next-to-run policy affect a job's position in a job queue. The preemption policy kills currently running jobs in order to free up resources for an urgent job. In case of the preemption with advance reservation policy an urgent job is always selected by a scheduler as a first job in a batch queue and running jobs are killed if and only if there are no available compute resources. Running urgent jobs on compute resources are not preempted. Other relevant assumptions to the considered problem are listed below:

- parallel job scheduling is considered where jobs belong to rigid and moldable and parameter sweep flexibility classes,
- there are two job classes: typical and urgent jobs submitted by end-users to batch queues at two levels, job resource requirements are known at submission time,
- urgent jobs can not be preempted and there is only one urgency token = critical,
- preempted typical jobs are resubmitted to a local batch queue,
- end-users submit both typical and urgent jobs to the system continuously, so jobs can arrive dynamically at any arbitrary time,
- compute resources are renewable and can be heterogeneous, their characteristics and attributes are discovered dynamically over time.

In our approach, we assume that with each urgent job there is advance reservation request attached. As a number of resource providers supporting advance reservation capabilities has been increasing over the last few years, we assume that there is no need to associate urgency tokens with urgent jobs as in the approach implemented in the SPRUCE system [33]. The main reason for that is that the way advance reservation mechanisms are implemented in local schedulers. In principle, advance reservation features are available for selected end-users and urgent jobs. Moreover, advance reservation requests have typically the highest priorities in batch queues, and finally can preempt typical running jobs without advance reservation. In other words urgent jobs and corresponding advance reservations have always higher priority than typical jobs. Additionally, if the advance reservation slot is not used by the urgent job, there are backfilling mechanisms that can be applied dynamically to increase the utilization of compute resources. Advance reservation requests for both urgent and typical jobs are forwarded to local schedulers whenever they arrive to the Grid broker.

The following resource management policies were proposed in the Grid broker to measure the impact of urgent jobs on various evaluation criteria:

- A load balancing (LB) policy is based on a number of both typical and urgent jobs in batch queues in local schedulers. Consequently, for all submitted typical and urgent jobs the resource provider with minimum queue length (a number of waiting jobs) is selected by the Grid broker.
- The second policy extends a bit LB by consolidating urgent jobs on one dedicated resource provider whereas typical jobs are distributed according to the LB policy.
- In the third policy we introduced prioritized resource providers for urgent jobs and we call it simply resource priority with LB. In other words, in this policy we try to consolidate urgent jobs on on preferred resource provider. However, if there are no available resources to run the urgent job immediately the Grid broker try to find a resource provider according to the LB policy.
- The Minimum Completion Time (MCT) policy allocates urgent jobs to a resource provider that guarantees the earliest completion time. Therefore, similar to above-mentioned two policies, the Grid broker is trying first to consolidate urgent jobs on one preferred resource provider. Then, in case a lack of available compute resources for the urgent job execution the Grid broker an alternative resource provider according to the MCT policy. Typical jobs are distributed according to the LB policy.

At the local level for all typical jobs the First-Come-First-Served (FCFS) with backfilling strategy was proposed. Typical jobs are taken from a batch queue in order of their arrival and allocated to compute resources. If a typical job can not be allocated at a given moment due to lack of compute resources the next typical job from the batch queue is checked. The decision about a new typical job to allocate is taken every time any job finishes or new either urgent or typical job arrives.

Scheduling the urgent job according to the next-to-run policy is not considered in this paper. In fact, this policy is easy to implement and support by resource providers within existing local schedulers. Nevertheless, urgent jobs scheduled according to this policy can wait up to hours or even days depending on the utilization of compute resources.

5. Experiments

5.1. Simulation environment

GSSIM has been designed as a simulation framework which enables easy-to-use experimental studies of various scheduling algorithms. Its goal is also to allow researchers to move the implementations of scheduling algorithms between simulation environments and real systems. GSSIM supports multilevel scheduling architectures with plugged-in algorithms both for Grid and local schedulers. It also enables both reading existing real workloads and generating synthetic Grid workloads based on given probabilistic distributions and constraints. These workloads are compliant with well known workload formats such as SWF [34] and GWF [35]. Additionally, GSSIM delivers several functionalities which are not, to the best of our knowledge, provided by any other distributed computing simulation tool available, namely: efficient network modeling including advance reservation capability, the possibility of adding customized application performance models related to both execution time and power usage, advanced modeling of various events including network failures and security issues, and the possibility of managing and executing simulations remotely in the cloud. GSSIM also contains a flexible workload generation tool allowing any number of jobs with sophisticated requirements to be created. A detailed architecture and comparison of GSSIM with other known simulators based on a classification of distributed computing simulator features are presented in [3].

5.2. Simulation parameters

All our experiments were performed using GSSIM. In our experiments we studied the performance of different two-level scheduling strategies introduced in the previous section using advance reservation. The real workload SDSC SP2 available at [34] with some minor modifications was applied. As there is a number of failed job submissions reported in this workload, we simply ignored such job submissions in our simulations. The number of compute resources reported originally for the SDSC SP2 workload is 128. As we wanted to use in our experiments two resource providers with the same number of resources we simply created two partitions, where each of them is available via a batch queue with 64 compute resources. In the considered workload there is only a small fraction of jobs requesting more than 64 compute resources, so those jobs were not used in our simulations. The selection of urgent jobs from the SDSC SP2 workload was made using discrete uniform distribution, assuming that roughly 10 percent of all jobs were defined as urgent. In the first set of simulations we did not modify a number of requested compute resources by urgent jobs. However, we wanted to observe the impact of bigger urgent jobs on various performance criteria, in particular an average resources utilization, a number of preempted typical jobs, total waiting time of typical jobs and finally total waiting time of urgent jobs. All performance tests and obtained results are briefly discussed in the next subsection. For experimental purposes we extracted one set of jobs from the original workload that contains jobs with IDs ranging from 68000 to 69000.

5.3. Results

First, by applying a basic configuration of resource management policies in our experiments, namely LB in the Grid broker and FCFS with backfilling in local schedulers, they generated the average utilization of compute resources in two resource providers at around 80%. It is worth emphasizing the fact that in general schedules generated by the simplest resource management policy - FCFS at both Grid and local levels are satisfactory if the average resources utilization is lower than 70 - 75 %. If the average resources utilization is higher than 70%, still FCFS with additional improvements is able to generate good results, e.g. conservative backfilling presented in [14]. Based on all the assumptions introduced in the previous section all jobs, where around 10% jobs were selected randomly as urgent, from the SDSC SP2 workload the first LB policy generated the high resources utilization between 82-84%, see Fig. 3. One should also note that schedules generated by other three policies led to a similar resources utilization.

In the next set of experiments we wanted to check if and how much bigger urgent jobs will affect the average resources utilization. Therefore, resource requirements of all previously selected urgent jobs (so called urgent job size) from the workload were modified to get the immediate access to 2, 4, 8, 16, 32, and 64 compute resources. According to obtained results, the first LB policy behaved quite well comparing to other policies for smaller urgent jobs. However, for bigger urgent jobs, especially for urgent jobs requesting 64 compute resources (all available resources offered by considered resource providers in experiments), we observed that the consolidation process can introduce some improvements reaching almost 93% of resources utilizations for the LB with consolidation strategy.

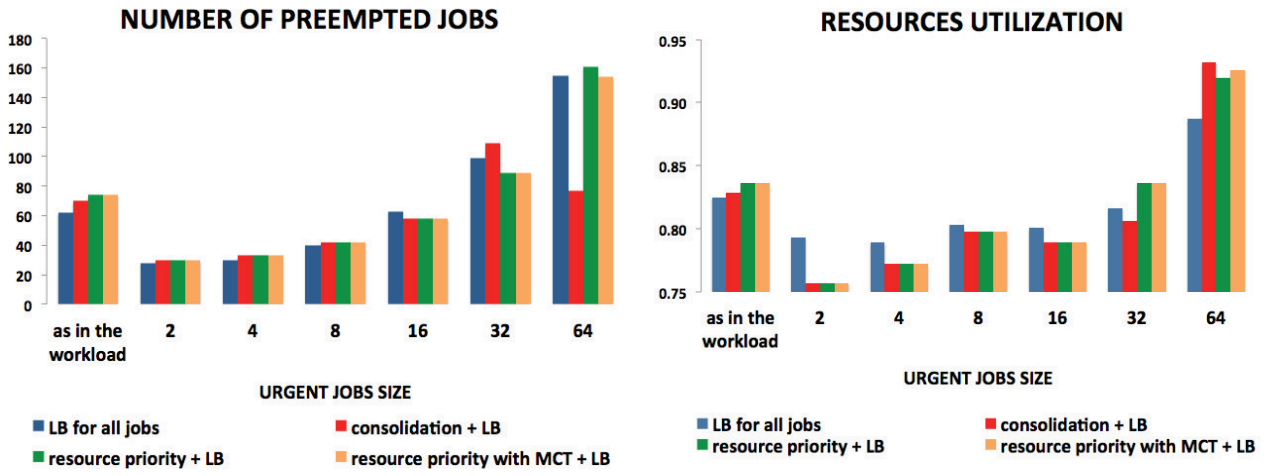

Figure 3: A number of preempted jobs and resources utilization generated by different resource management strategies for 10% of urgent jobs with different resource requirements (job sizes) in the SDSC SP2 workload.

This observation was similar to our recent results presented in [20] where we experimentally proved that for a mixed workload consisting of batch jobs and advance reservations it is worth to consolidate together advance reservation requests on preferred resource provider(s). In other words, if possible, advance reservations should be allocated after another advance reservation rather than after a batch job.

As we argued in previous sections we should not evaluate the efficiency of a resource management strategy from a perspective of only one criterion such as resources utilization. Consequently, the second criterion - a number of preempted jobs is also presented in Fig. 3. As expected we observed some fluctuations for big urgent jobs with job size 32 and 64. For the selected workload the best performance we achieved again for the LB with consolidation policy.

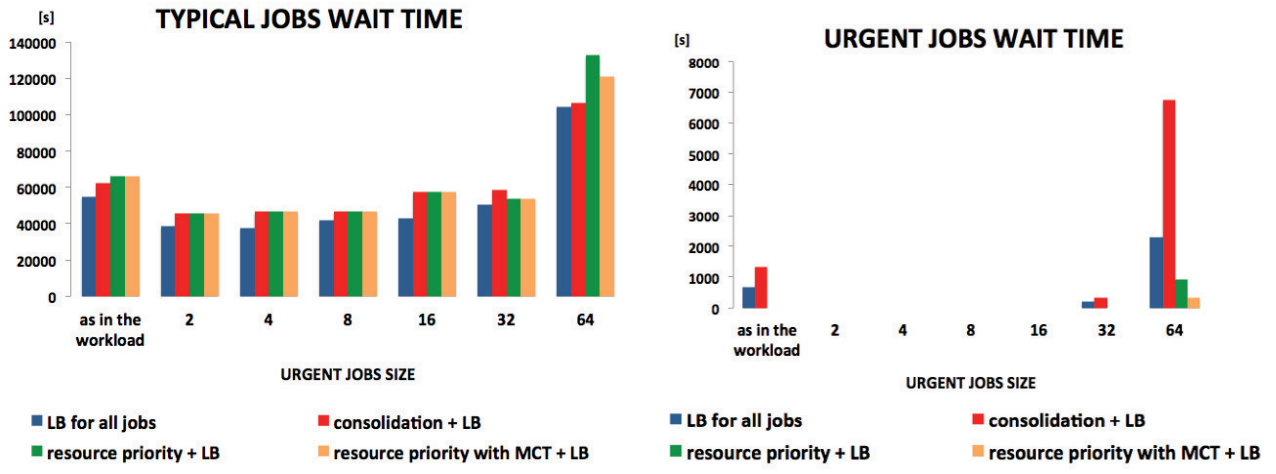

Figure 4: Total waiting time of both typical and urgent jobs generated by different resource management strategies for 10% of urgent jobs with different resource requirements (job sizes) in the SDSC SP2 workload.

Obtained schedules presented in the light of two criteria clearly suggested that the LB with consolidation policy should be applied at the Grid broker to deal with urgent computing in Grids. Nevertheless, our performance evaluation has changed once we compared results obtained on two additional criteria: total waiting time of typical and urgent jobs, see in Fig. 4. Surprisingly, total waiting time of typical jobs was relatively stable for different urgent job sizes,

except the last experiment where the biggest urgent jobs were considered. All four policies generated schedules where total waiting time of typical jobs was doubled or even achieved around 130000 seconds for the third policy (resource priority with LB) comparing to 58000 seconds for urgent job sizes set at 32. However, the most important criterion in the context of urgent computing is total waiting time of urgent jobs presented in Fig. 4. One can easily notice that around 10% of randomly selected urgent jobs from the real workload did not impact significantly the overall performance of resource providers as urgent jobs did not have to wait in local batch queues. A small number of relatively small urgent jobs requesting 2, 4, 8, or even 16 compute resources (25% of all available compute resources) were executed immediately by all four resource management strategies. Nevertheless, obtained results on the third criterion for bigger urgent jobs revealed disadvantages of basic LB and LB with simple consolidation strategies as urgent jobs had to wait 10 times or even longer comparing to results obtained thanks to the resource priority with MCT and LB policy.

6. Conclusions

In this paper based on performed experiments we discovered various interesting properties of two-level scheduling strategies with respect to urgent computing in distributed computing environments. We demonstrated that in real cases urgent jobs that appear rarely in a parallel computing environment in general may not disturb significantly the management of other less critical jobs. Moreover, we showed how an increasing fraction of urgent jobs impact different evaluation criteria considered by resource providers, administrators and end-users. Additionally, we presented how advance reservation capabilities can be easily applied for urgent computing as they were designed in principle to deal with special or more important job requests in local schedulers and are commonly supported today. We believe that there is a room for various improvements to the best resource management policy - the resource priority with MCT and LB policy. This policy was selected after various performance tests and our multicriteria evaluation process.

In this paper we also clearly indicated that generated schedules heavily depend on the particular workload, steering scheduling parameters and configuration of compute resources. Therefore, we performed most experiments on the real and well known workload adding as many as possible references to previous analysis and related work. In the near future we would like to perform some new experiments of recently collated real workloads where additional attributes, such as job priorities or advance reservation slots are available.

7. Copyright notes and acknowledgements

The presented work was sponsored by the UCoMS project under award number MNiSW (Polish Ministry of Science and Higher Education) 469 1 N USA/ 2009 in close collaboration with U.S. research institutions involved in the U.S. Department of Energy (DOE) funded grant under award number DE-FG02-04ER46136 and the Board of Regents, State of Louisiana, under contract no. DOE/LEQSF(2004-07).

References

[1] Avellino, G., Barale, S., Beco, S., Cantalupo, B., Colling, D., Giacomini, F., Gianelle, A., Guarise, A., Krenek, A., Kouril, D., Maraschini, A., Matyska, L., Mezzadri, M., Monforte, S., Mulac, M., Pacini, F., Pappalardo, M., Peluso, R., Pospisil, J., Prelz, F., Ronchieri, E., Ruda, M., Salconi, L. Salvet, Z., Sgaravatto, M., Sitera, J., Terracina, A., Vocu, M., Werbrouck, A., The EU DataGrid Workload Management System: towards the second major release. CHEP 2003, La Jolla, California, March 2003.

[2] Beckman, P., Nadella, S., Trebon, N., Beschastnikh, I. (2006). SPRUCE: A System for Supporting Urgent High-Performance Computing. In Proceedings of the IFIP WoCo9 Conference, pp. 295-316, Springer, USA, 2006

[3] Bak, S., Krystek, M. Kurowski, K., Oleksiak, A., Piatek, W., Węglarz, J. (2011). GSSIM - a Tool for Distributed Computing Experiments, Scientific Programming, vol. 19(4), pp. 231-251, 2011.

[4] Bubak, M., Szepieniec, T., Wiatr, K. (Eds.) (2012). Building a National Distributed e-Infrastructure PL-Grid. ISBN 978-3-642-28266-9, March 2012, Springer Lecture Notes in Computer Science, Vol. 7136

[5] Cappello, F., et. al. (2005). Grid'5000: A Large Scale and Highly Reconfigurable Grid Experimental Testbed. In Proc. of the 6th IEEE/ACM International Workshop on Grid Computing, pp. 99-106

[6] Catlett, C., et al. (2007). TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications. HPC and Grids in Action, Ed. Lucio Grandinetti, IOS Press 'Advances in Parallel Computing' series, Amsterdam, Holland
[7] Chapin, S., Cirne, W., Feitelson, D. G., Jones, P., Leutenegger, S., Schwiegelshohn, U., Smith, W., Talby, D. (1999). Benchmarks and Standards for the Evaluation of Parallel Job Schedulers. In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (Eds.), Lect. Notes Comput. Sci. vol. 1659, pp. 66-89
[8] Downey, A. B. (1997). A parallel workload model and its implications for processor allocation. In Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing
[9] Downey, A. B. (1997) Predicting queue times on space-sharing parallel computers. In proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, IPPS conference
[10] Dutot, P.-F., Eyraud, L., Mounie, G., Trystram, D. (2004). Bi-criteria algorithm for scheduling jobs on cluster platforms. In Proceedings of Symposium on Parallel Algorithm and Architectures, Barcelona, Spain
[11] Dutot, P.-F., Trystram, D., A best-compromise bicriteria scheduling algorithm for parallel tasks. In Proceedings of WEA'05 4th International Workshop on Efficient and Experimental Algorithms, Santorini Island, Greece
[12] Ellert, M, et al. (2007). Advanced Resource Connector middleware for lightweight computational Grids, Future Generation Computer Systems 23 (2007) 219-240
[13] Iosup, A., Dumitrescu, C., Epema, D. H, Li, H., Wolters, L. (2006). How are real grids used? The analysis of four grid traces and its implications. In GRID, pp. 262-270, IEEE Computer Society
[14] Jones, J., P., Nitzberg, B. (1999). Scheduling for parallel SUpercomputing: A historical perspective on achievable utilization. In Proceedings of the Workshop on Job Scheduling Strategies for Prallel Processing (JSSPP), Lecture Notes in Computer Science, vol. 1659, Springer, Berlin
[15] Kurowski, K., Nabrzyski, J., Oleksiak, A., Węglarz, J. (2003). Multicriteria Aspects of Grid Resource Management. In J. Nabrzyski, J. Schopf, and J. Węglarz (Ed.), Grid Resource Management (pp. 271 - 293). Kluwer Academic Publishers.
[16] Kurowski, K., Nabrzyski, J., Oleksiak A., Węglarz, J. (2008). Multicriteria Approach to Two-level Hierarchy Scheduling in Grids. Journal of Scheduling, 11(5), 371-379.
[17] Kravtsov, V., Schuster, A., Carmeli, D., Kurowski, K. Dubitzky, W. (2008). Grid-enabling complex system applications with QosCosGrid: An architectural perspective, in Proc. of The Int, Conference on Grid Computing and Applications (GCA'08), Las-Vegas, USA.
[18] Kurowski, K., de Back, W., Dubitzky, W., Gulyas, L., Kampis, G., Mamonski, M., Szemes, G., Swain, M. (2009). Complex System Simulations with QosCosGrid. Lecture Notes in Computer Science, 5544, 387-396.
[19] Kurowski, K., Oleksiak, A., Weglarz, J. (2010). Multicriteria, multi-user scheduling in Grids with advance reservation. Journal of Scheduling, 13(5), 493-508
[20] Kurowski, K., Oleksiak, A., Piatek, W., Weglarz, J. (2011). Hierarchical Scheduling Strategies for Parallel Tasks and Advance Reservations in Grids. Journal of Scheduling. 11(14), 1-20, DOI 10.1007/s10951-011-0254-9
[21] Dubitzky, W., Kurowski, K., Schott, B. (Eds.) (2012). Large-Scale Computing Techniques for Complex System Simulations, ISBN: 978-0-470-59244-1, 2011, Wiley-IEEE Computer Society Press
[22] Li, H., Groep, D. L., Wolters, L. (2004). Workload characteristics of a multi-cluster supercomputer. In JSSPP, vol 3277, 176-193.
[23] Li, H., Muskulus, M., Wolters, L. (2007). Modeling Correlated Workloads by Combining Model Based Clustering and A Localized Sampling Algorithm. In proc. of 21st ACM International Conference on Supercomputing (ICS07), Seattle, USA
[24] Lo, M., Mache, J., Windisch, K.J. (1998). A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling. In Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science; Vol. 1459, pp 25-46
[25] Lublin, U., Feitelson, D. (2003). The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. J. Parallel and Distributed Computing, 63(11), 1105-1122.
[26] Marru, S., Gannon, D., Nadella, S., Beckman, P., Weber, D. B., Brewster, K. A., Droegemeier, K. K.. CTWatch Quarterly, Volume 4, Number 1, March 2008.
[27] Sodan, A., C. (2005). Loosely coordinated coscheduling in the context of other approaches for dynamic job scheduling: a survey. Concurrency and Computation: Practice & Experience, 2005(17), 1725-1781.
[28] Platform LSF, http://www.platform.com/
[29] SGE, http://www.sun.com/software/sge/
[30] PBS, http://www.pbsgridworks.com/
[31] Maui, http://www.clusterresources.com/products/maui/
[32] Using and administering LoadLeveler for AIX 5L. Technical Report. http://publibfp.boulder.ibm.com/epubs/pdf/a2278810.pdf
[33] SPRUCE resources, http://spruce.teragrid.org/status.php
[34] Parallel Workload Archive. http://www.cs.huji.ac.il/labs/parallel/workload/
[35] Grid Workloads Archive. http://gwa.ewi.tudelft.nl/