On the Impact of Reservations from the Grid on Planning-Based Resource Management

Felix Heine[1], Matthias Hovestadt[1], Odej Kao[1], and Achim Streit[2]

[1] Paderborn Center for Parallel Computing (PC[2]),
Paderborn University, 33102 Paderborn, Germany
{fh, maho, okao}@upb.de
[2] Zentralinstitut fuer Angewandte Mathematik (ZAM),
Forschungszentrum Juelich (FZJ), 52425 Juelich, Germany
a.streit@fz-juelich.de

Abstract. Advance Reservations are an important concept to support QoS and Workflow Scheduling in Grid environments. However, the impact of reservations from the Grid on the performance of local schedulers is not yet known. Using discrete event simulations we evaluate the impact of reservations on planning-based resource management of standard batch jobs. Our simulations are based on a real trace from the parallel workload archive. By introducing a new option for scheduling reservations in planning-based resource management, less reservation requests are rejected. Our results are important for increasing the acceptability of the Grid technology. We show, that a limited number of additional resource reservations from the Grid have only a limited impact on the performance of the traditionally submitted batch jobs.

1 Introduction

Currently, a gap [2] exists between the demands of Grid middleware and the capabilities of the underlying resource management systems (RMS). While Grid middleware systems provide a certain amount of support for Quality of Service (QoS), underlying RMSs offer only limited functionality in this aspect. Service Level Agreements (SLA) [7] are powerful instruments for assuring QoS, but simply adding SLA negotiation mechanisms to an existing RMS is not sufficient. A demand for advanced abilities in runtime management and control exists. A basic mechanism for accomplishing a guaranteed resource usage are *advance reservations*. They denote the reservation of a fixed amount of resources for a given time span. In the context of Grid Computing, reservations are indispensable to realize the simultaneous availability of distributed resources and to orchestrate Grid workflows. In queuing-based systems reservations are complex to realize, as scheduling focuses only on the present resource usage. In contrast, planning-based resource management systems [3] do resource planning for the present and future. Their design implicitly supports advance reservations.

In this paper we evaluate the impact of reservations on the schedule quality of planning-based resource management systems. This impact is important to

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3516, pp. 155–162, 2005.

evaluate, as reservations affect the scheduling of the traditionally existing workload of batch jobs on supercomputers. Reservations occupy resources for a fixed time span, which gives the scheduler less opportunities to schedule batch jobs. With only some reservations in the system, this impact can be invisible, as the scheduler has still some free slots in the schedule. However, with an increased usage of Grid technology more reservations will be submitted to the underlying resource management systems. At a certain point, the impact will become visible and traditional batch jobs have to wait longer for execution.

We evaluate this impact by using discrete event simulations and generating a fixed amount of batch jobs from a workload trace. We increase the amount of submitted reservation requests and measure the average slowdown of batch jobs. Additionally, we measure the rejection rate of the submitted reservation requests and the utilization of the system.

The intuitive approach for scheduling reservations is to reject them, if the requested resources are already scheduled to batch jobs. We introduce a new approach for scheduling reservations, where such interfering batch jobs are scheduled to a later time and the reservation can be accepted.

The remainder of this paper is structured as follows: we first give an overview on related work, following an overview on the characteristics of planning-based scheduling. In section 4 we present the results of our simulations, thus measuring the impact of advance reservations. A brief conclusion closes this paper.

2 Related Work

Many publications deal with the implementation and applications of advance reservations. However, not much publications have been written about the impact of reservations on resource management.

In 2000, Smith, Foster and Taylor published their results on scheduling with advance reservations [8]. At this, they proposed and evaluated several algorithms, regarding utilization, mean wait time and the mean offset from requested reservation time, which is the time between the reservation request and the receipt of the reservation. All measurements were generated with an Argonne National Laboratory (ANL) workload trace. A basic assumption in their simulations is, that running jobs that have been reserved are not terminated to start other jobs. It is also assumed that reserved jobs will always start on time. These assumptions imply the necessity of run time estimates for each job to assure that resources are free on time. In fact, extending this approach to all types of jobs would lead to planning-based scheduling.

For systems which do not support resumable jobs (i. e. intermediate results of a job are not saved, so that a job can be restarted after termination), measurements show an superproportional increase of waiting time by increasing the percentage of reservations. Likewise, the mean offset from requested reservation time is increasing superproportional. By decreasing the percentage of queued jobs that can not be delayed by a reservation, waiting time as well as offset decrease. For resumable jobs, which can be terminated and restarted at a later

time without losing results, the jobs to terminate can be determined using an equation regarding number of nodes and used computing time. For this type of jobs, further decrease of waiting time and offset can be measured.

In [1] a queuing scheduler is enhanced to support advanced reservations. The authors state that users with lower priorities may gain start time advantages by using advanced reservations. To prevent this, the authors applied a shortest notice time for every reservation. It is defined using a predictive waiting time as if the reservation would have been submitted as a queued job. The prediction is based on historical data. A performance evaluation shows that advanced reservations prolong the queue waiting time, although no start time advantages are taken. Hence, longer shortest notice times have to be applied to prevent this.

Similar as before, this work reveals the restrictions of queuing-based scheduling. As only the present resource usage is scheduled, decisions on future reservations can not be based on future load information, instead predictions have to be done. With applying notice times to reservations, users can not specify arbitrary desired start times for their reservations. This is possible in planning-based scheduling presented in this paper.

3 Scheduling Reservations in Planning-Based Systems

If we look in depth at the way current resource management systems work, we find two different approaches: *queuing-based* and *planning-based* systems. In [3] we give a detailed overview on the two types of scheduling approaches. Examples for planning-based systems are the Maui scheduler [4] or CCS [5]. Choosing one of these approaches has a huge impact on the provided features and in particular on the possibility to support advance reservations.

Due to their design queuing-based systems do not support reservations by nature. Hence, work-arounds have to be used, e. g. the insertion of dummy-jobs in high-priority queues, or suspend and resume of low-priority jobs, but they all have drawbacks. Either, the number of reservations is limited, or in fact a second, planning-based scheduler is implemented on top of the queuing-based resource management system. In planning-based systems, on the other hand, reservations are implicitly available in a trivial and easy way.

We call standard batch jobs *variable jobs* as they can move on the time axis until they are started. The scheduler places such jobs at the earliest possible free slot in the schedule. If prior jobs finish earlier than expected, or get aborted, the scheduler might re-schedule the job to an earlier time. Depending on the scheduling policy (e. g. SJF, LJF), jobs may also be moved to a later point in time. *Reservations* have a user specified start time and the scheduler can either accept or reject the reservation for the given start time. Reservations limit the scheduler in placing variable jobs, as resources are occupied for a fixed time.

In case the scheduler cannot place a reservation due to a conflict with a not yet running variable job, we have two options. In the first one, which we call reject, a new reservation is only accepted, if the resources requested for the given time frame are available. The reservation is rejected, if the resources are

occupied by already running jobs, by previously accepted reservations, or by planned variable jobs.

With the new variable jobs move option the scheduler additionally accepts those reservations, for which the requested resources are only occupied by variable jobs which are planned, but not yet running. The scheduler plans the overlapping variable jobs at a later time, so that the reservations can be accepted. Still, a reservation is rejected, which collides with already running jobs or other reservations.

Note, starvation of variable jobs is possible with the variable jobs move option, in case newly submitted reservations constantly delay variable jobs. Appropriate mechanisms have to be implemented in this case to prevent starvation.

4 Evaluation

It is common practise to use discrete event simulations for the evaluation of job-scheduling strategies. For this purpose we implemented the reject and variable jobs move options in our MuPSiE system (Multi Purpose Scheduling Simulation Environment).

For evaluating the simulated schedules we use different metrics. We are focusing on the slowdown metric s_i (= response time divided by run time) and weight it by the job area a_i (= run time $\cdot$ requested resources), resulting in the $SLDwA$ metric defined as $SLDwA = \sum a_i \cdot s_i / \sum a_i$.

For measuring reservations other metrics have to be used than for variable jobs, as reservations do not have a waiting time which depends on the scheduler's performance and workload. As already mentioned in Section 3 reservations are either started at their requested start time or they are rejected. Hence, the total number of rejected reservations and respectively the rejection rate in percent of all submitted reservations is measured in our evaluations.

Finally, the overall utilization of the machine is used to measure the efficiency of the scheduler to place all simulated requests, variable jobs and accepted reservations. This information about utilization is of particular interest for system owners.

4.1 Workload

An evaluation of job scheduling policies requires to have job input. In this work a variable job (Section 3) is defined by the submission time, the number of requested resources, and the estimated run time, which is mandatory for planning-based resource management system. The use of a discrete event simulation environment for the evaluation requires additional data about the actual run time of a job. A reservation is defined by the same four parameters like a variable job, but additionally a requested start time is specified.

The variable jobs in this work are derived from a trace of the Parallel Workload Archive [6]. We are using the CTC trace from the Cornell Theory Center. It was taken from a 512-node IBM SP-2 machine during July 1996 to May 1997. It

contains 79,279 valid jobs and only 430 nodes of the 512 are available for batch processing.

Unfortunately, no traces are available which contain the appropriate data about reservations required for our simulations as described above. Hence, reservations have to be generated synthetically. In our work this is a two step process and we use the variable jobs from the CTC trace as a basis.

In a first step variable jobs are randomly chosen to get their properties (submission time, number of requested resources, estimated and actual run time). These properties are the basis for newly generated reservations. An input parameter defines, in percent of all variable jobs, how many reservations should be generated in total. If set to 100%, every variable job is used to copy its properties and generate a new reservation, hence the amount of submitted variable jobs and reservation is 79,279 each. With reservations added to the set of variable jobs, the overall workload submitted to the scheduler is increased.

The most important property of a reservation, its requested start time, is generated in the second step. At first a start delay is computed as follows: $\textit{start delay} = \textit{rand}[0,1] \cdot \textit{estimated run time} \cdot \textit{start factor}$. Assume that the start factor is initially set to 1, then the start delay is a random number between 0 and the estimated run time. This random start delay is added to the original submission time of the variable job and the resulting value is the requested start time for the reservation. The input parameter *start factor* is used to modify the start delay, so that the requested start time of reservations is potentially further (start factor > 1.0) or sooner (start factor < 1.0) in the future. With these two steps all necessary properties of a new reservation are copied and generated.

4.2 Results

Figure 1 and Figure 2 show the results for the two different scheduling options, three different start factors, and the three used performance metrics. Note that we repeated each simulation ten times and took the average result to exclude singular effects from generating random jobs.

With the variable jobs move option the scheduler moves interfering variable jobs to a future start time. This implies, that less reservations are rejected,

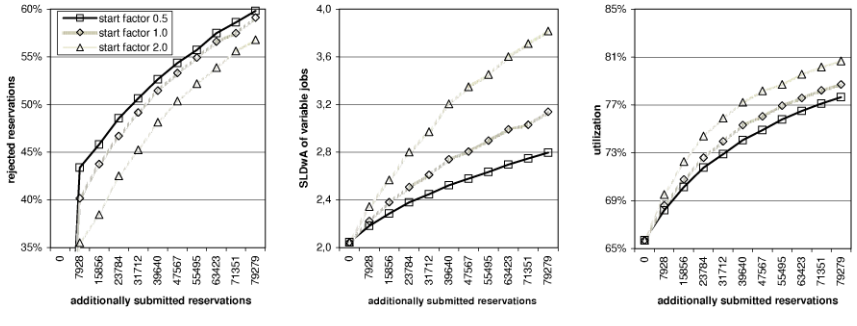

Fig. 1. Results with start factor 0.5, 1.0, and 2.0 for the reject option

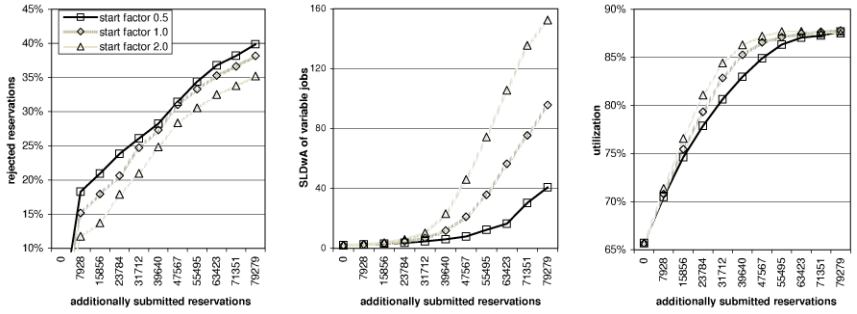

Fig. 2. Results with start factor 0.5, 1.0, and 2.0 for the variable jobs move option

but variable jobs have to wait longer for a start, i. e. their slowdown (SLDwA) increases. Still, if resources are occupied by previously accepted reservations, or by already started reservations, or variable jobs, new reservation requests are rejected with the variable jobs move option.

Focusing on the percentage of rejected reservation and the SLDwA metrics for the two scheduling options in Figures 1 and 2 shows this behavior. The more reservation requests are submitted, the more reservations the scheduler has to reject, as these additional reservations increase the workload and potentially more resources are occupied by other reservations or running jobs. This is the case for both scheduling options. Independent of how many reservations are submitted, the scheduler has to reject more reservations with the reject option, than with the variable jobs move option. For different amounts of submitted reservations the difference between both options is at least 20 percentage-points. With only some reservations submitted, this constant difference means, that more than twice as much reservations are rejected with the reject option. Note, with no reservations submitted at all, obviously no reservations are rejected (or accepted) by both options.

Taking a look at the SLDwA values for all scheduled variable jobs confirms the above made assumption. With the reject option, the scheduler does not replace variable jobs for new reservation requests. They are started at their originally assigned start time and they are not further delayed. However, made reservations may occupy resources for variable jobs submitted in the future, so that their (and thereby the overall) slowdown increases slightly. The curve for variable jobs move clearly shows, that moving variable jobs further in the future and thereby accepting more reservations, significantly increases the slowdown of variable jobs. The results become worse, the more reservations are submitted.

With additionally submitted reservations the workload increases and thereby the utilization, too. However, the utilization can not increase unbounded, which can be seen for the variable jobs move option in particular. Not more than approx. 88% is possible. The planned schedule is so compact, that additional reservations would only increase the slowdown (SLDwA) of variable jobs. Comparing the two scheduling options shows, that with the variable jobs move option the utilization is higher than with the reject option. This is based on the fact, that

more reservations can be accepted (cf. the percentages of rejected reservations) and that the scheduler has more freedom in planning variable jobs by placing them further in the future.

The start factor is used to modify the start delay, so that the requested start time of reservations is potentially further (start factor > 1.0) or sooner (start factor < 1.0). Increasing the start factor from to 0.5 to 1.0 and 2.0 means that the time difference between the submission and average requested start time of reservation requests gets larger. Combined with the random computation of the requested start times, they are equally distributed over a larger time span. This reduces the limitations for the scheduler to accept reservations and to (re)place variable jobs. Thus the scheduler can accept more reservations. This can be seen in the diagrams for both scheduling options. With a start factor of 2.0 less reservations are rejected, independent of the total amount of submitted reservations.

With a similar argumentation one could expect that the SLDwA values of variable jobs should decrease with increasing start factors: requested start times are further in the future and variable jobs might be placed at an earlier time in the schedule. However, the diagrams show different results. Increasing the start factor leads to increased SLDwA values for both scheduling options. On a first look the made assumption is correct and variable jobs are potentially placed earlier in the schedule. However, the reservations are still in the schedule, which means that the scheduler is still limited in placing variable jobs. As previously mentioned more reservations are accepted with increased start factors and this additionally limits the scheduler in placing variable jobs. In the end, the scheduler is more limited in placing variable jobs with an increased start factor, so that the SLDwA of variable jobs increases for both options.

As less reservations are rejected with increased start factors, the workload increases and the generated utilization is higher. This can be seen for both scheduling options, but observing the utilization diagram for the variable jobs move option in detail shows, that at the beginning the difference becomes larger and at the end almost the same utilization is achieved. With the variable jobs move option the system runs in a saturated state with many reservations submitted. Accepting more reservations as a result of increased start factors does only mean that the SLDwA values increase and variable jobs have to wait longer. Hence, the differences in the percentage of rejected reservations are smaller for the variable jobs move option than for the reject option. The saturated state is not reached with the reject option, so that a higher utilization is generated by the scheduler with different amounts of reservations submitted and the achieved utilization values are smaller.

5 Conclusion

At the beginning of this paper we revealed the demand for QoS and guaranteed resource usage support in modern resource management systems. To this end, planning systems are well suited for this endeavor due to their design. Planning

the present and future resource usage by assigning proposed start times to all waiting jobs makes reservations trivial to schedule. Although reservations are well-known, their impact on planning-based scheduling was not yet evaluated.

We introduced two scheduling options, called reject and variable jobs move, and evaluated the impact of different numbers of reservations on the performance of these options. As no real workload trace contains reservation requests, they were added synthetically. Based on the properties of randomly chosen jobs from the trace, we generated several job sets with different amounts of added reservations. Furthermore, we generated different start delays. The performance was measured by: the percentage of rejected reservations, the average slowdown of variable jobs weighted by their area (SLDwA), and the utilization of the system.

Our results show, that both scheduling options have their benefits. With reject the SLDwA of variable jobs does not increase much due to the additionally accepted reservations, but many reservations are rejected by the scheduler. With the variable jobs move option more reservations are accepted by the scheduler, as variable jobs are moved to a future start time. Hence, the result is an increased SLDwA of variable jobs. Generating reservations, which are further in the future, means that the scheduler can accept more reservations, so that the utilization increases. As more reservations are accepted the scheduler is more limited in placing variable jobs, hence their SLDwA increases.

In contrast to a queuing system, scheduling reservations in a planning system does not mean to terminate running jobs to guarantee the requested start time of the reservation. This is a direct consequence of planning the present and future resource usage not only of reservations, but also of all waiting variable jobs.

References

1. J. Cao and F. Zimmermann. Queue scheduling and advance reservations with cosy. In *Proc. of 18th Intl. Parallel and Distributed Processing Symposium*, 2004.
2. G. Fox and D. Walker. e-Science Gap Analysis. Technical report, Indiana University, USA, 2003.
3. M. Hovestadt, O. Kao, A. Keller, and A. Streit. Scheduling in HPC Resource Management Systems: Queuing vs. Planning. In *Proc. of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, 2003.
4. D. Jackson, Q. Snell, and M. Clement. Core Algorithms of the Maui Scheduler. In *Proc. of 7th Workshop on Job Scheduling Strategies for Parallel Processing*, 2001.
5. A. Keller and A. Reinefeld. Anatomy of a Resource Management System for HPC Clusters. In *Annual Review of Scalable Computing, Singapore University Press*, 2001.
6. Parallel Workloads Archive. `http://www.cs.huji.ac.il/labs/parallel/workload`.
7. A. Sahai, S. Graupner, V. Machiraju, and A. v. Moorsel. Specifying and Monitoring Guarantees in Commercial Grids trough SLA. Technical Report HPL-2002-324, Internet Systems and Storage Laboratory, HP Laboratories Palo Alto, 2002.
8. W. Smith, I. Foster, and V. Taylor. Scheduling with Advanced Reservations. In *Proc. of the 14th Intl. Conference on Parallel and Distributed Processing Symposium*, 2000.