Optimizing User Oriented Job Scheduling within TORQUE

Dalibor Klusáček[†*] klusacek@cesnet.cz
Václav Chlumský[†*] vchlumsky@cesnet.cz
Hana Rudová[*] hanka@fi.muni.cz

[*]Faculty of Informatics
Masaryk University
Brno, Czech Republic

[†]CESNET
Association of Legal Entities
Prague, Czech Republic

ABSTRACT

We presents major extension of the widely used TORQUE Resource Manager. Unlike common resource managers that use queuing approach, our solution uses planning (job schedule construction) and schedule optimization in order to achieve better predictability, performance and fairness with respect to common queue-based approaches. Moreover, additional important features are supported, e.g., so called multi resource fairness that is used to fairly prioritize users subject to their (highly) heterogeneous demands concerning various system resources. Also, new textual and graphical interfaces allow users to better control and plan computation of their jobs. Our solution is currently undergoing experimental deployment in Czech national Grid *MetaCentrum*.

1. INTRODUCTION

Job scheduling methods using planning and schedule optimization have been studied in the past and their suitability and better performance have been demonstrated with respect to standard queue-based solutions, that are widely used in Grid and cluster systems [5, 3]. However, many of these works used simplified models together with a simulator, while realistic and complex implementations in actual mainstream resource managers were not available for various reasons, e.g., due to huge complexity of such systems.

2. CONTRIBUTION

The main contribution of this work is that we have successfully developed a plan-based scheduling system within the production TORQUE Resource Manager [1] system. Compared to classical queue-based schedulers, our system supports several important features. First of all, it allows better *predictability* as every job's execution is plan ahead by constructing job execution plan (job schedule). Next, such a schedule is periodically *optimized using metaheuristic* in order to improve schedule's quality with respect to considered optimization criteria. Here we build upon our previous research on fast, multi-criteria optimization methods [3].

Both *performance and fairness-related criteria* are optimized. We support so called *multi resource fairness* that properly prioritize users of the system with respect to their (highly) heterogeneous demands concerning system resources [2]. Furthermore, we also provide new *textual and graphical user interfaces* that allow users to better control and plan execution of their jobs. Prior deployment, our solution has been experimentally evaluated, demonstrating good performance, fairness and adequate accuracy of created schedules.

These features represent major benefits compared to classical queue-based techniques like backfilling where predictability is typically very limited and scheduling decisions are neither evaluated nor further optimized with respect to various optimization criteria.

3. APPLIED SOLUTION

Let us briefly describe the main parts of our solution.

3.1 New TORQUE Scheduler

The core part is the newly developed TORQUE scheduler (see `pbs_sched` module in Figure 1) that contains *complete job schedule* data structures and the *schedule construction* algorithm which works in a backfill-like fashion, i.e., it adds newly incoming jobs at the earliest available time slots in the schedule. We emphasize realistic aspects by considering inaccurate job runtime estimates as well as complex job characteristics including, e.g., RAM and HDD requirements along with common CPU-related demands. Beside that, there are new *maintenance routines* that adjust the schedule in time subject to dynamic events such as (early) job completions, machine failures, etc. Finally, `pbs_sched` hosts newly developed schedule optimization algorithm which we describe in Section 3.2.

3.2 Schedule Optimization

Thanks to the use of schedule, an expected start time is known for every job prior its actual execution. Therefore, the schedule can be periodically evaluated in order to identify possible inefficiencies. Then, a *local search-inspired metaheuristic* is used to produce better schedules with respect to applied optimization criteria. We consider both performance and fairness-related criteria. First of all, we minimize *wait time (WT), response time (RT)* and *bounded slowdown (SD)* to improve overall performance.

Moreover, *user-to-user fairness* is also optimized. We use a per user metric called *Normalized User Wait Time (NUWT)*.

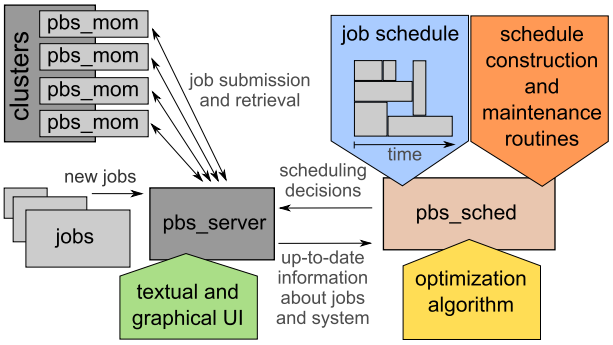

Figure 1: Extended TORQUE Resource Manager.

For a given user, NUWT is the total user wait time divided by the amount of previously consumed system resources by that user. Then, the user-to-user fairness is optimized by minimizing the mean and the standard deviation of all NUWT values. Calculation of NUWT reflects consumption of multiple resources as we explain in Section 3.3.

3.3 Multi Resource Fairness

In our solution we use recent multi resource-aware methods to keep *user-to-user fairness* at a good level [2]. The solution follows classical fairshare principles, i.e., a user with lower resource usage and/or higher total wait time gets higher priority over more active users and vice versa. Notably, the solution reflects *heterogeneous jobs' demands* concerning different system resources. It means that both *CPU and RAM consumption* is taken into account when measuring given user's resource utilization. By using multi resource fairness we can better react on users with (highly) heterogeneous demands on available resources.

Figure 2 illustrates the importance of multi resource-based fairness. It shows two variants of penalties (Z-axis) that can be used for prioritizing users according to their relative consumption of available CPU and RAM resources (X and Y-axes). Single resource-based (CPU only) penalty is on the left while multi resource-based (CPU and RAM) penalty is on the right. Clearly, single resource-based penalty (left) cannot properly penalize those users who, e.g., use (nearly) all RAM but only a small fraction of available CPUs.

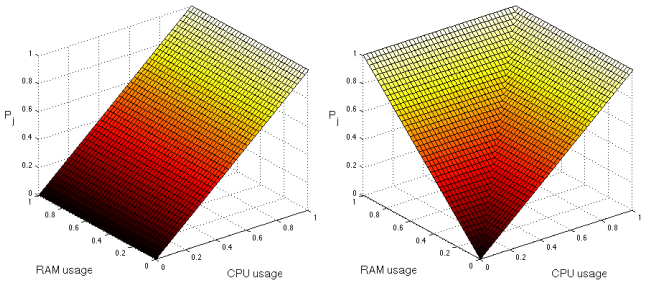

Figure 2: Examples of different penalties (P_j) distributions wrt. CPU and RAM consumption.

3.4 User Interfaces

As the schedule increases predictability, our users can obtain information about planned start times of their jobs using either textual or web-based interfaces. The first one is an *extended* `qstat` *command* while the second one is a newly developed *web application*. Both methods display planned start times of waiting jobs by communicating with the `pbs_server` (see Figure 1) that provides up-to-date information using current job schedule.

4. EVALUATION AND DEPLOYMENT

Prior an experimental deployment, the system has been evaluated using historic workloads from Czech National Grid Infrastructure *MetaCentrum* [4], analyzing its performance, fairness and the accuracy of start time predictions.

Compared to existing MetaCentrum scheduler, the average wait time (WT), response time (RT) and bounded slowdown (SD) have decreased by 73%, 26% and 74%, respectively. Concerning fairness, we have used two indicators (see Section 3.2) which are the mean NUWT and the std. deviation of all NUWT values. Here, the average improvements has been 55% and 34%, respectively. We have also measured how accurate is a job's initial start time prediction (established when a job arrives in the system) with respect to its real start time. Here we have observed that 33.8% of jobs start earlier (typically due to earlier completions of preceding jobs) and 48.5% start on time. There are 17.7% of jobs that start later than were their initial predictions. In most cases, this is a natural result of schedule optimization, which (beside other factors) tries to improve user-to-user fairness. To do that, jobs belonging to "low priority" users must be delayed to improve performance for users with high priority.

The solution presented in this work is currently being *experimentally deployed* in *MetaCentrum*. During this summer it will be further tested by selected users and then it will be freely available to users as an alternative to the current queue-based solution. Our system can be freely obtained at: `http://www.metacentrum.cz/docs/pbts`.

5. ACKNOWLEDGMENTS

We acknowledge the support provided by "Projects of Large Infrastructure for Research, Development, and Innovations" LM2010005 funded by the Ministry of Education, Youth, and Sports of the Czech Republic and the support provided by the grant No. P202/12/0306 of the Grant Agency of the Czech Republic.

6. REFERENCES

[1] Adaptive Computing Enterprises, Inc. *TORQUE Admininstrator Guide, version 4.2.4*, July 2013. `http://docs.adaptivecomputing.com`.

[2] D. Klusáček, M. Jaroš, and H. Rudová. Multi resource fairness: Problems and challenges. In *Job Scheduling Strategies for Parallel Processing*, Boston, USA, 2013.

[3] D. Klusáček and H. Rudová. Efficient Grid scheduling through the incremental schedule-based approach. *Computational Intelligence*, 27(1):4–22, 2011.

[4] MetaCentrum, October 2013. `http://www.metacentrum.cz/`.

[5] F. Xhafa and A. Abraham. *Metaheuristics for Scheduling in Distributed Computing Environments*, volume 146 of *Studies in Computational Intelligence*. Springer, 2008.