IADIS International Journal on Computer Science and Information Systems
Vol. 13, No. 2, pp. 47-61
ISSN: 1646-3692

FEASIBILITY ANALYSIS OF USING THE MAUI SCHEDULER FOR JOB SIMULATION OF LARGE-SCALE PBS BASED CLUSTERS

Georg Zitzlsberger, Branislav Jansík, and Jan Martinovič
IT4Innovations, VŠB - Technical University of Ostrava, Czech Republic

ABSTRACT

For large-scale High Performance Computing centers with a wide range of different projects and heterogeneous infrastructures, efficiency is an important consideration. Understanding how compute jobs are scheduled is necessary for improving the job scheduling strategies in order to optimize cluster utilization and job wait times. This increases the importance of a reliable simulation capability, which in turn requires accuracy and comparability with historic workloads from the cluster. Not all job schedulers have a simulation capability, including the Portable Batch System (PBS) resource manager. Hence, PBS based centers have no direct way to simulate changes and optimizations before they are applied to the production system. We propose and discuss how to run job simulations for large-scale PBS based clusters with the Maui Scheduler. This also includes awareness of node downtimes, scheduled and unexpected. For validation purposes, we use historic workloads collected at the IT4Innovations supercomputing center. The viability of our approach is demonstrated by measuring the accuracy of the simulation results compared to the real workloads. In addition, we discuss how the change of the simulator's time step resolution affects the accuracy as well as simulation times. We are confident that our approach is also transferable to enable job simulations for other computing centers using PBS.

KEYWORDS

High Performance Computing (HPC), Simulation, PBS, Maui Scheduler, Job Scheduling

1. INTRODUCTION

For large-scale High Performance Computing (HPC) centers with a wide range of different projects and heterogeneous infrastructure, efficiency is an important consideration. Depending on the computing center and its objectives, efficiency comprises the optimization of the utilization of the available systems, or minimization of the wait times for users' jobs.

HPC centers continually strive to improve efficiency. Unconfirmed changes and experiments involving live production systems are not justifiable due to an increased risk of undesirable performance effects, such as job congestion or starvation of jobs, unexpected job scheduling behavior for users, etc., or simply because of a lack of comparability as usage patterns of users change with projects. This problem has already been addressed by HPC job schedulers and resource managers by adding simulation capabilities (e.g. Slurm Simulator (Trofinoff, and Benini, 2015), Maui Scheduler (Adaptive Computing, 2014), LSF (IBM, 2016)). However, not all solutions used today offer a simulator, which is also the case for PBS Professional (Altair, 2018). It hence does not allow the analysis of changes to the job scheduler configuration or effects of node reservations before deployment to the live system.

There exists a large range of simulation frameworks for HPC and grid systems. For research purposes, simulators like SimMatrix (Wang, 2014), GridSim (Buyya, and Murshed, 2002), SimGrid (Casanova, et al., 2014), and CQSim (Yang, X., et al., 2013) are widely used to study the effects of large-scale systems. Whilst SimMatrix demonstrated better resource usage than GridSim and SimGrid, and is capable of simulating Exascale level systems (Wang, et al., 2013), it was only validated against real systems of up to 4096 cores with artificial tasks. Another simulator is Alea (Klusáček, and Rudová, 2010), which is based on the GridSim toolkit. It is validated with real workloads (Klusáček, and Chlumský, 2016) for approximately 5,200 CPU cores, containing 102,657 jobs over a span of 4 months. CQSim enables plan based and energy aware simulations, and is validated with systems up to 164k CPU cores and 69k jobs. However, all of the above are only toolboxes, and require a manual implementation of the job scheduling algorithms to approximate an existing system. They are also not a suitable in situ replacement for the scheduling algorithms of HPC job management systems.

The Slurm Simulator is an extension of the HPC scheduler Slurm, originally developed by A. Lucero (Lucero, 2011) and improved further by S. Trofinoff and M. Benini. As such, Slurm based systems have the advantage of making use of the simulation capabilities directly. The Slurm Simulator has been validated for large-scale simulations of real workloads from TACC (Simakov, et al., 2017) with 6,400 nodes over 12 weeks.

For LSF based systems, simulations are so far only capable of predicting the start times of jobs. However, further plans to develop simulation capabilities have been announced (Ding and Li, 2017). The Cobalt Scheduler, developed by Argonne National Laboratory, comes with a simulator called Qsim (Argonne National Laboratory, 2018). It only is applicable within an existing Cobalt environment.

The Maui Scheduler (MS in the following) has existed since the 1990s, and even though there is no active development anymore it is open source, mature and highly configurable. Its simulation capability has already been used to analyze HPC workloads (Jackson, et al., 2000). Optionally, it can also be applied as a meta-scheduler to replace the job scheduling components of HPC resource managers like PBS (Bode, et al., 2000) and TORQUE (Advanced Computing, 2018). This allows the simulation and the job management system to use the same scheduling algorithms and configurations.

We use the MS to complement PBS with the analysis of historic workloads of large-scale clusters. To show the viability of MS for larger systems we describe the configuration and changes to approximate the behavior of the PBS job scheduler for a non-trivial 2 PetaFLOP/s system. We furthermore discuss the results and problems we faced whilst using real workload data of that system.

The paper has two main sections. Section 2 describes how to approximate an existing PBS configuration using an additive job priority formula with MS. Section 3 discusses MS simulation results with a real workload from a large-scale PBS based cluster. Along with a summary of our results in section 4, we also point out current limitations.

2. USING MS IN A PBS ENVIRONMENT

PBS and MS are independent solutions with different configuration options and philosophies. Hence, there is no direct mapping of a PBS configuration to a MS equivalent. PBS supports various job-scheduling policies like First In First Out (FIFO), resource allocation rules, by resource consumption, by time slots, and by priority. MS on the other hand supports a wide range of job priority factors, e.g. job credentials, fairshare usage, requested resources, etc.

Due to the wide range of possible configurations, we concentrate on a PBS setup, using the job priority formula as an additive term. We furthermore consider a PBS configuration without preemption, with strict ordering for priorities, neither round robin nor by queue selections, and no sorting of jobs or nodes. Without loss of generality, we are confident that for many other practical configurations of PBS an approximate MS setup exists. For example, MS supports FIFO based scheduling directly and offers Quality of Service (QoS) attributes to honor different resource categories in addition.

In the following, we explain how to map the PBS job priority formula and workloads to MS. We also discuss how to simulate node and system downtimes with MS.

2.1 PBS Job Scheduling Configuration for MS

If PBS uses priority based scheduling, a formula describes the computation of the job execution priorities. This kind of job scheduling by PBS is freely configurable yet it follows the principles of job priorities, fairshare, eligible time and backfilling. In the following we describe such terms and how to map them to MS.

2.1.1 Job Priorities

As an example, the priority p of a job can be defined as shown in Equation 1 by using a job priority formula. The priority of the queue (class for MS) a job is submitted to is q, w is the job wait time in seconds, and f is the fairshare priority. Equation 2 shows an example of a fairshare priority definition.

$$p(q, f, w) = 1{,}000 * q + \frac{f}{1{,}000} + \frac{w}{864{,}000} \qquad [1] \qquad\qquad f = 10^6 * \left(1 - \frac{u_p}{u_t}\right) \qquad [2]$$

In the above example, the queue priority has a weight of 1,000 to dominate the other terms. The fairshare term is used to give users of a specific project (i.e. account for MS) a higher priority to users of another account in the same queue, if their usage of resources had been lower in the past. It considers the ratio of a project's usage u_p to the total usage u_t and applies a weighting to set it as the second most prioritized term. The wait time term favors one job over others within the same queue and with the same fairshare value, should they have been waiting longer. The eligible time has the lowest weight in the formula and is noticeable only for long wait times with over 10 days (864,000 s). The weights are examples. PBS supports more coefficients (Altair PBS Works, 2011), such as job priority and resource usage depending on the number of CPUs, memory, wall time, and CPU time.

For MS, the closest equivalent to the job priority formula are the job priority factors (Supercluster R&D Group, 2002). They group into the categories: job credentials, fairshare usage, requested job resources, current service levels, target service levels, and consumed resources. There are further subcategories to access finer grained metrics, e.g. users or groups of the job credentials, and queue time of the current service level.

MS considers all such (sub-)-categories as weighted additive terms. Using Equation 1 we can express an analogous priority setup in MS using the following priority categories and subcategories[1]:

```
...
Priority += CREDWEIGHT[864000] * CLASSWEIGHT[1000] * J->C->Priority[q]
Priority += FSWEIGHT[8640000] * FSACCOUNTWEIGHT[1] * dAccountFS[f]
Priority += SERVICEWEIGHT[1] * QUEUETIMEWEIGHT[1] * QUEUETIME[w]
...
```

Assigned coefficients and variables are in brackets. All other unused weights MS offers are set to zero. Variables `J` and `C` denote a job and its class. Differences of the current fairshare value to a target within the range of [0...100] are expressed as `d*FS` (in the example for accounts only). The initial value for `Priority` is zero and accumulated over all categories, to result in the final job priority.

2.1.2 Queue Priorities and Constraints

Real systems typically have different queues for various purposes, like offering express queues to give short jobs a higher priority in order to minimize wait times, or have dedicated service queues with the highest global priority. Use of those queues can be subject to constraints. Constraints can be limitations of the available nodes for a specific queue, job, or user.

In MS, queue priorities are set via `CLASSCFG[q] PRIORITY=X`, with `q` specifying the queue name and `X` as its priority. Further attributes `MAXNODE=A`, `MAXNODEPERJOB=B` and `MAXNODEPERUSER=C` are available to limit the maximum number of available nodes globally (`A`), per job (`B`) or per user (`C`), respectively.

2.1.3 Fairshare

Fairsharing, if enabled, avoids the starvation of jobs of less active users caused by power users submitting large amounts of jobs. The fairshare configuration can be set in MS via `*CFG[DEFAULT] FSTARGET=X` for each partition, such as `USERCFG` or `ACCOUNTCFG` for user or account based fairshare, respectively. The target for the fairshare `X` is the usage in percent per partition. According to Equation 2, fairshare is among projects only, with a target of 100%. Thus, the MS equivalent would be `ACCOUNTCFG[DEFAULT] FSTARGET=100`. The variable `f` is the computed fairshare difference provided directly by MS. In MS, the computation of the fairshare difference can be parameterized by interval and decay factor (PBS' half-life setting), fairshare policy, and depth. The closest MS configuration to the default PBS fairshare would be:

```
FSPOLICY       DEDICATEDPS    # CPU time
FSDEPTH        1              # 24 h half-life...
FSINTERVAL     24:00:00
FSDECAY        0.50
```

[1] The formulas are documented in the Maui Administrator's Guide (Supercluster R&D Group, 2002)

It defines a 24h half-life, using dedicated processor seconds (**DEDICATEDPS**) as a resource usage indicator corresponding to PBS' `cput` metric. Only one fairshare window (**FSDEPTH 1**) is considered.

2.1.4 Backfilling

In addition to priority based scheduling, resources for which no suitable scheduled reservation exists can be used for backfilling. This allows lower priority jobs to fill up idle resources as long as their reservations do not delay any higher priority jobs. In contrast to MS, PBS does not have different backfilling policies. We assume that *FIRSTFIT* (Jackson, et al., 2001) is the closest approximation. It dispatches jobs as they fit on the remaining resources without any exhaustive search. In addition, the number of jobs for backfilling can be limited, which is typically considered together with the reservation depth to limit the number of reservations. The tradeoff between more or less reservations with less or more backfilled jobs allows for conservative or aggressive backfilling (Srinivasan, et al, 2002), respectively. An example MS configuration would be:

```
BACKFILLPOLICY FIRSTFIT
BACKFILLDEPTH  3 # Select only 3 jobs for backfilling
RESDEPTH       1 # Only one reservation per node (less conservative backfilling)
```

This resembles a liberal backfilling setup of PBS with a small backfill depth (PBS' `backfill_depth`).

2.2 Using Historic PBS Workloads for MS

A typical PBS based system does not provide simulation workloads in the form required for MS. We briefly describe the PBS data needed for simulations with MS.

2.2.1 Trace Files

MS uses two different trace files, one for the description of the cluster configuration (e.g. `Resource.Trace`) and one for the simulation workload (e.g. `Workload.Trace`). The documentation of their format is in the Maui Administrator's Guide (Supercluster R&D Group, 2002).

A description of the cluster configuration is required for the simulator, so it is aware of available resources. The list of available nodes can be directly queried from PBS via the `pbsnodes` command. The most important data fields for the resource trace file are the node's name, available resources, such as CPU cores, memory, etc., and a list of supported classes.

The workload trace file requires information about submitted jobs, which PBS typically stores in an SQL database[2]. It contains various information of which the following are needed for the simulation: submission, start and end times of jobs, requested number of nodes, user provided wall time, selected queue, and the user, group and account IDs. This information is reliable as PBS creates it directly. However, there are rare events of PBS failures or changes applied to the job scheduler itself that might corrupt the data and require validation.

[2] PBS uses PostgreSQL by default

2.2.2 Unexpected Node Downtimes

If a node goes offline unexpectedly, the job scheduler has to reschedule reservations that were made for that node. This cannot take place in advance, as those downtimes are unpredictable. It is also unclear for how long the downtime will last. Hence, the scheduler can only start planning reservations for it again if the node goes back online. On a real system, nodes are considered offline if a hardware defects occur, nodes need to be rebooted with a different configuration, or the load on nodes is high enough to make them appear unavailable. To simulate this effect in MS, the unexpected node offline times are required. Information about node downtimes can be retrieved from the PBS log file.

MS supports reservations but does not have the functionality to replay them as with workload trace files. We hence propose to use MS' `setres`, `releaseres`, `canceljob`, and `schedctl` commands. The simulation advances until it meets a time step in which the node goes offline, where it places a *user* reservation with infinite duration. When advancing further and reaching a time step where that node turns online again, the related reservation is removed. A node only has one such *user* reservation at a time. We further assume that a failing node terminates its associated job. The following pseudo code shows how to combine these commands:

```
1.  for step := 1 to last_step do begin
2.    ∀ nodes switching offline between step - 1 and step:
        a.  canceljob: delete node's job, and
        b.  setres: place unlimited reservation on node
3.    ∀ nodes switching back online between step - 1 and step:
        releasers: remove reservation from node
4.    schedctl: advance one time step
5.  End
```

2.2.3 Planned Maintenance

A full system (or site-wide) maintenance is typically known about weeks in advance and announced to all users in good time. Hence such periods are well documented even if not always machine-readable (e.g. in an internal ticket system). Compared to unexpected downtime, the cluster will become fully idle immediately before the system maintenance period (ramp down) and continue its normal operation after the downtime.

To simulate planned maintenance, *system* reservations are created for every one of these windows at the beginning of the simulation. This announces them to the scheduler sufficiently ahead of time. For example, the system maintenance for all nodes is created with the following `setres` command starting at 2017-01-29 8:00:00 and lasting 20,000 s: **`setres -s 08:00:00_01/29/2017 -d 20000 ALL`**

3. SIMULATION RESULTS WITH REAL WORKLOADS

We validate our MS configuration with historic workloads from a large-scale PBS based cluster system with a non-trivial configuration. First, we explain the real cluster setup and PBS configuration in terms of job scheduling. Second, we describe the historic workloads that were available and explain their use for MS. Lastly, we discuss early simulation results concerning accuracy and different time step resolutions with their simulation times.

3.1 Reference Cluster Setup

We use workloads collected from IT4Innovations' Salomon cluster (2 PetaFLOP/s) which uses PBS Professional 13.1. As of writing, it is ranked 139th in the Top500 list (Top500, 2018). Its nodes contain two Intel Xeon E5-2680v3 processors with 24 cores and 128 GB of memory in total. The entire cluster has 1,008 nodes, of which 432 are equipped with two Intel Xeon Phi 7120P coprocessors in addition. The job scheduling system does not offer node sharing, and as such only assigns entire nodes to a job (multiples of 24 cores/tasks).

Users submit jobs to selected queues and specify the type of nodes. Eligibility for using a queue depends on the queue type, wall time of the job, and free project resources (accounting). Certain restrictions apply to the queues as shown in Table 1. For the simulation, queue imposed wall time constraints are not relevant since the real system rejects jobs with wall time violations. Historic workloads contain no such rejected jobs.

During the simulated time window, use of the coprocessors required allocation of their nodes. Past the observed period an additional queue `qmic` was introduced to allocate coprocessors separately. Hence, our workloads do not contain the `qmic` queue yet.

PBS uses the job formulas shown in Equation 1 and 2 (IT4Innovations, 2018). The configuration of fairshare is account based with a half-life of 168h (`FSINTERVAL 168:00:00`). Liberal backfilling is used with a maximum of three jobs per cycle (`BACKFILLDEPTH 3`). The simulation required an individual increase of `RESDEPTH` depending on the simulation window. This is a side effect of emulating node downtimes using one *user* (unexpected downtime) and multiple *system* reservations (one per system maintenance). There is at most one *user* reservation per node if it is down. Reservations for the entire system, however, depend on the amount of system maintenance periods during the simulation window. Hence, `RESDEPTH` should be individually set by adding `1 + # of system maintenance periods` for each simulation window.

Table 1. Queues and constraints of the Salomon cluster

Queue name	Number of nodes	Priority q	Maximum wall time	MS configuration
`qexp`	1,006 + 2 dedicated (max. 8 per user, max. 32 total)	150	1h	`CLASSCFG[dexp] PRIORITY=150 MAXNODE=32 MAXNODEPERUSER=8`
`qprod`	1,006	0	48h	`CLASSCFG[qprod] PRIORITY=0`
`qlong`	576 (max. 256 total)	0	144h	`CLASSCFG[qlong] PRIORITY=0 MAXNODE=256`
`qmpp`	1,006	0	4h	`CLASSCFG[qmpp] PRIORITY=0`
`qfree`	1,006 (max. 752 total)	-1,024[3]	12h	`CLASSCFG[qfree] PRIORITY=-1024 MAXNODE=752`
`qmic`	2 * 432	0	24h	`CLASSCFG[qmic] PRIORITY=0`
`SERVICE`	1,008	512	N/A	`CLASSCFG[SERVICE] PRIORITY=512`

[3] Negative priorities require Maui's `ENABLENEGJOBPRIORITY TRUE` and `REJECTNEGPRIOJOBS FALSE`

3.2 Historic Workloads for the Simulation

For the simulation, we used three different data sources: job information from PBS stored in an SQL database for all submitted jobs, logs from PBS for unexpected node downtimes, and a list documenting the planned system maintenance downtimes.

For the job information in the SQL database, we found that less than 1% of all job records had incorrect information or were the subject of system changes. As such, we removed them from the workloads. We used a 50 week simulation window in 2017 and consecutively enumerated the jobs by assigning job IDs in ascending order by submission time.

Unexpected node downtime information is currently an approximation as there is no standardized source as with submitted jobs. We parsed the PBS logs to reconstruct the unplanned offline times, which showed good results but still had spurious false positives. Such false positives occur for example if nodes reboot for a different configuration or nodes are utilized so that they appear offline to PBS. Due to the large amount of offline reports over the 50 weeks window, manual validation is not feasible. We thus filtered out all offline times shorter than 4 minutes, which should account for the majority of sporadic and erroneous offline reports. Fortunately, planned maintenance had been well communicated to users, and consequently information is readily available and accurate.

Simulation results are available within the `simstat.out` file created by MS for each simulation run. In addition to the workload trace, it contains the job execution and end times. From those, it is possible to derive cluster utilization and job wait times.

3.3 Changes Applied to MS

To enable MS to simulate a cluster the size of Salomon (1,008 nodes, 24,192 cores), with a peak of approximately 60k jobs in the queues, some defaults need to be changed. For properly sized buffers we increase `MAX_MTASK` and `MMAX_JOB` to 32,768 and 80,000 respectively. In addition, `MAX_PRIO_VAL` needs to be changed from 10^9 to 10^{12} to account for the extended priority value range (compare Equation 1 and Table 1).

To simulate historic workloads, it is beneficial if the simulator uses the exact timestamps of each job. By default, MS will use the current time for the first job of the simulation (relative time). We have applied modifications to MS so it uses the absolute timestamp as provided in the workload trace. This helps to easily identify timeframes in the past.

3.4 Results

In the following, we highlight the initial results from our simulation runs. We compare the accuracy of the simulations and the real system. We do so on a week-by-week basis but prepend the preceding week as a warm-up phase. Furthermore, we explain the impact of different simulation time step resolutions with their simulation times.

3.4.1 Accuracy of the Simulation

For demonstration purposes, we selected week 7 because it contains one system maintenance period, and a large amount of unexpected downtimes. As shown in Figure 1 (left), the simulation shows similar utilization to the real workload. It is visible that during the warm-up week 6 (gray area) the utilization of the cluster builds up and follows the real utilization in week 7 (starting within day 46[4]).

Around the system maintenance period (green area), both the simulation and the real system show a ramp down of utilization. This is typical for a system shutdown, as queued jobs will not fit in the remaining time before all nodes go offline, according to the wall time specified by the users. At the end of the maintenance period, queued jobs are assigned to nodes again. In this example, the backlog of jobs accrued during the downtime is quite low, which results in a lower cluster utilization (<80%) directly after the maintenance period.

Unexpected downtimes (red area) reduce the total number of nodes on the system, which is visible by the utilization adjusting accordingly. Even though most of the unexpected downtimes happen early during the warm-up week, their effect is already visible. The node downtimes during the system maintenance period are also visible in red, but have no effect since there is a *system* reservation overlapping them.

Furthermore, Figure 1 (right) compares the wait times of jobs between the real workload and the simulation. Every data point is one executed job, colored depending on the queue used. The wait time is the difference between the execution and submission time of a job. It is visible that especially for the job bursts (to queue *qexp*) the patterns are identical and wait times are similar. The patterns result from how jobs are submitted, by which user/account/group, and the queue restrictions. There are also some deviations visible, which result from job dependencies as discussed below.

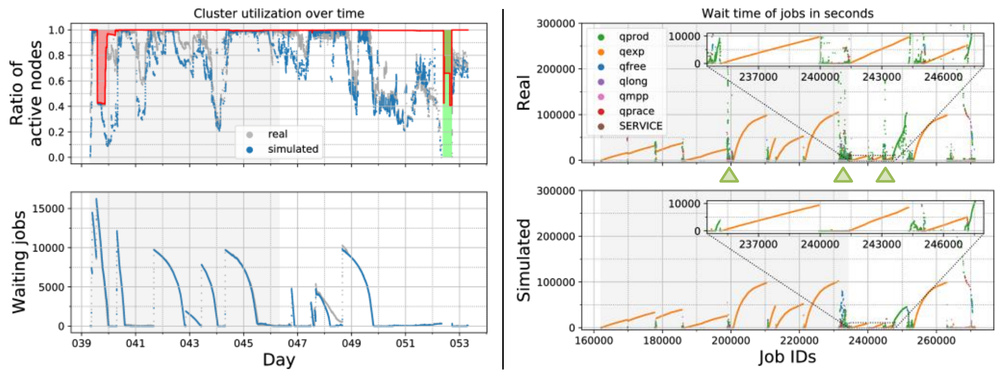

Figure 1. Comparison of the real workload of week 7 with a simulation using a 5 s time step resolution left: real (gray) and simulated (blue) cluster utilization with unexpected node downtimes (red) and a system maintenance period (green); right: job wait times of real (top) and simulated (bottom) workloads with jobs from different queues, in ascending order of submission time

As a contrast to week 7, we show week 29 (starting within day 200) in Figure 2, which does not have a system maintenance period, and has only minor unexpected system downtimes. It is a representative of the usual utilization of the cluster, with a larger number of shorter jobs and some job bursts.

Figure 3 shows a comparison of the wait times of jobs for weeks 6, 7, and 29 in detail. At the top, it uses Cumulative Density Functions (CDFs) for the real and simulated workloads with a 5 or 60 s time step resolution. For better readability, we only show those time step resolutions

[4] Weeks are aligned with first job of the historic workloads and hence are not calendar weeks

in the CDF diagrams. At the bottom of Figure 3 we show the errors with mean $\bar{x}$, median $\tilde{x}$, and standard deviation σ for all time step resolutions we evaluated. Errors are the difference between the real and simulated wait times. For all diagrams, only jobs submitted at the week of interest are considered, which excludes the warm-up phase.

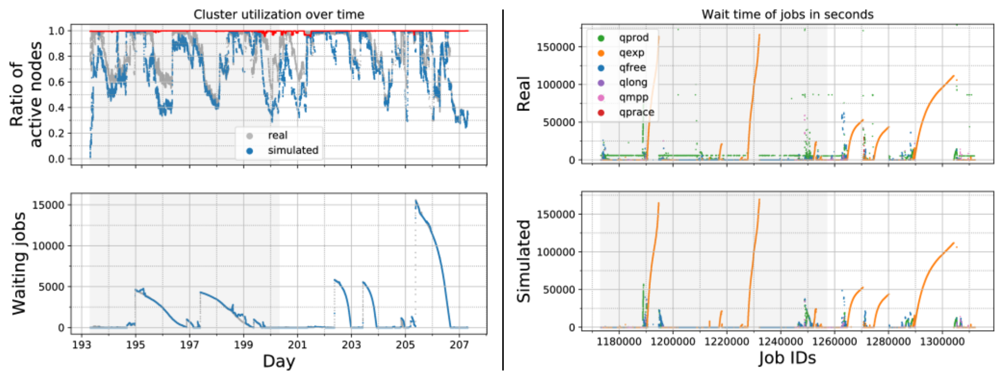

Figure 2. Comparison of the real workload of week 29 with a simulation using a 5 s time step resolution left: real (gray) and simulated (blue) cluster utilization with unexpected node downtimes (red); right: job wait times of real (top) and simulated (bottom) workloads with jobs from different queues, in ascending order of submission time

The wait times of the simulations are tentatively lower than for the real system, since the simulation made the assumption of ideal job dispatching where nodes are immediately available. A job does not instantly execute if the job scheduler selects it. Prologue and epilogue scripts are executed to prepare and validate a node before a job is executed, and to clean up after the job is finished, respectively. The runtime of such scripts is dependent on the resources (i.e. nodes) selected and the job itself. Nodes without an Intel Xeon Phi coprocessor have a shorter prologue as validating the coprocessors takes additional time. In addition, a job that creates a significant amount of temporary data on a node increases the cleanup time compared to a job that keeps persistent data on the user's home drive. The runtimes of such scripts are not part of the PBS logs or otherwise recorded. The typical runtime of such scripts is between seconds and a few minutes. In the simulation we assume an ideal scenario, where nodes become available instantly. This results in differences that are more visible for jobs with a shorter runtime. For longer runtimes, the simplification has less relative effect.

Comparing the errors of the different time step resolutions, it becomes clear that an increase in resolution results in less error. The error converges to a minimum at the highest resolution of 1 s. Using a 60 s time step yields a large error for week 6 ($\bar{x}$: 103,386, $\tilde{x}$: 116,932, σ: 53,415), week 7 ($\bar{x}$: 39,617, $\tilde{x}$: 18,984, σ: 53,245), and week 29 ($\bar{x}$: 27,951, $\tilde{x}$: 951, σ: 70,094). In contrast, the error when using a 5 s time step are reduced for week 6 ($\bar{x}$: -52, $\tilde{x}$: 171, σ: 7,557), week 7 ($\bar{x}$: -2,223, $\tilde{x}$: -10, σ: 21,137), and week 29 ($\bar{x}$: 167, $\tilde{x}$: 2, σ: 2,952).

In addition, the CDFs for the wait times further demonstrate two additional properties of the workloads. First, towards the lower end of the wait times, within the range of 10 to 100 s, the simulation tends to have much lower wait times than the real system. This is due to the ideal job dispatching in the simulation. Second, there are more outliers in the real system, whose wait times are about one order of magnitude larger compared to the simulation. This is mostly due to job dependencies. Job dependencies are used to overcome the wall time limitations of jobs by submitting a batch of shorter running dependent jobs at once. Such jobs are executed one after the other and enable users to run computations longer than the maximum wall time of the targeted queue. For weeks 6 and 7, mostly jobs from the queue `qprod` showed job dependency patterns - they have a constant wait time difference (e.g. 24h interval). In the real workload, they show up as vertical spikes, which are marked with green triangles in Figure **1** and Figure **5** below. Job dependency information is not available in the current PBS logs and hence preventing its simulation. In the current simulations, such jobs are not considered dependent and are scheduled like any other job.

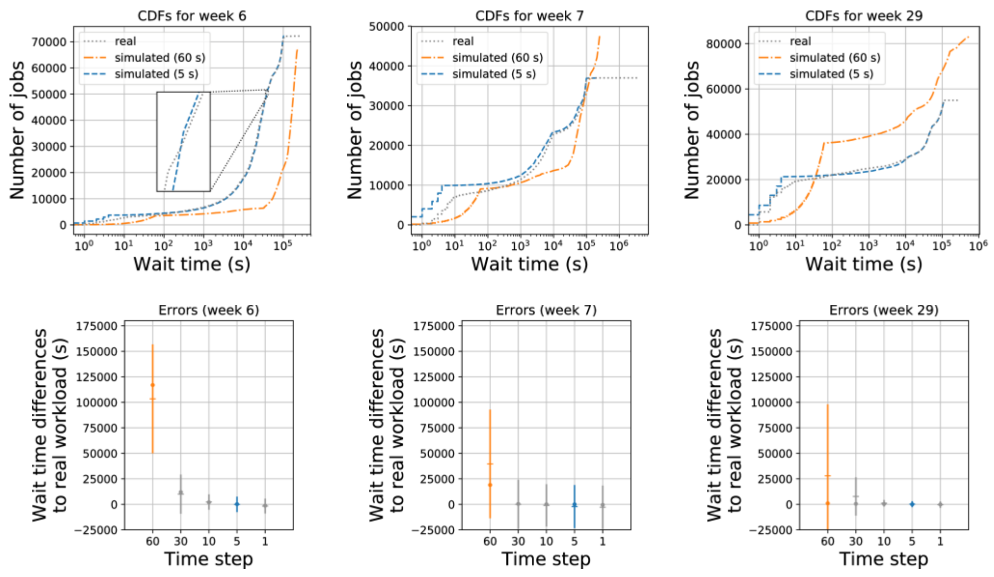

Figure 3. Cumulative density functions (CDF) of job wait times and errors for weeks 6, 7 and 29
For weeks 6 (top left), 7 (top middle), and 29 (top right), the CDF for real and simulated workload with 5 and 60 s time steps are shown; corresponding errors compared to the real workload are shown with mean (bar), median (dot) and standard deviation (range) for each simulation time step resolution (bottom)

3.4.2 Time step Resolution

Jobs submitted to PBS immediately trigger a scheduling event. The historic logs have a resolution of one second, which allows for a fine-grained replay. The MS simulator is capable of handling arbitrary simulation time steps down to a second. However, such a fine-grained resolution increases the simulation time proportionally. For example, the simulation of week 7 took approximately 42h and 239h (in the order of 5 times difference) with a resolution of 5 and 1 s, respectively. Comparing resolutions of 10 to 5 s, the simulation slowdown is about a factor of 2, following the increased time step complexity. The slowdowns with increasing resolutions are not caused by our approach to handling unexpected node downtimes, with placing and removing reservations from nodes. That handling is a constant overhead, independent of the time step resolution. Figure 4 compares the simulation times over the full 50 week range for each week (including the warm-up week) and different time step resolutions. It is visible that differences in simulation times are proportional to the time step resolutions but not to the number of submitted jobs. Additional factors influence the complexity of the simulation such as the amount of waiting jobs in the queues, mean job runtime, cluster utilization, and the amount of reservations during the simulation window. We have observed a variation of simulation times by up to two orders of magnitude within the same time step resolution across different weeks. In addition, the memory usage per simulator instance varies between approximately 2 and 10 GB. The memory requirements primarily depend on the number of waiting jobs in the queues, which changes over time. During our simulations, the queue lengths range from a few thousand up to over 60k jobs waiting at peak times. We ran our simulation on a two socket Intel Xeon E5-2665 with 8 cores per socket, enabled Hyper-threading (simultaneous multithreading) and with 64 GB of memory. Since simulation instances are not dependent on each other, simulations are embarrassingly parallel, providing enough main memory is available.

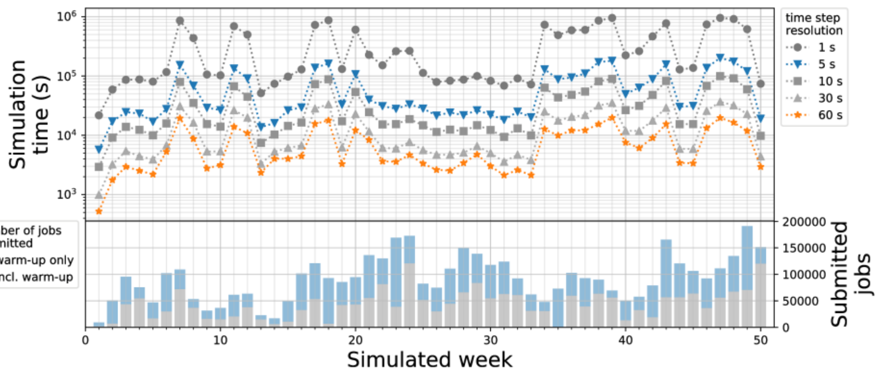

Figure 4. Comparison of simulation times for different time step resolutions
Each week is simulated independently and contains its previous week for the simulator warm-up; simulation time grows proportionally as the time step resolution is increased; measurement was done on the same system with one simulation instance per core

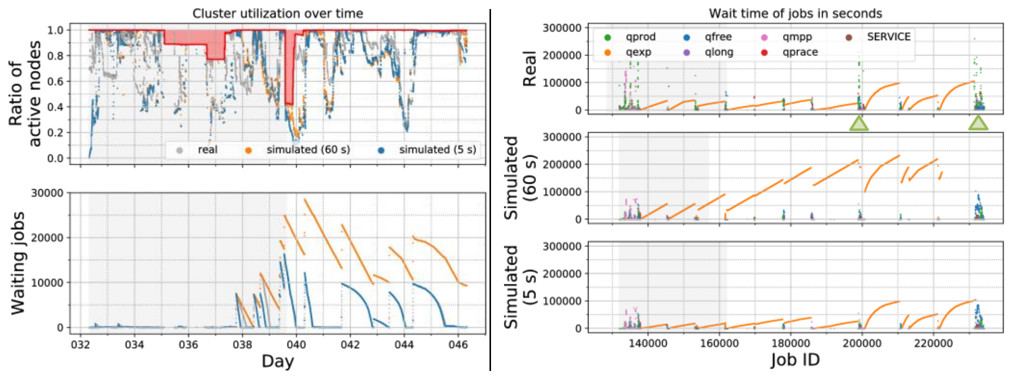

Figure 5. Comparison of the real workload of week 6 with simulations using different time step resolutions of 60 and 5 s
Left: real (gray) cluster utilization compared to simulations with 60 s (orange) and 5 s (blue) time step resolution; right: job wait times of real (top) and simulated workloads with 60 s (middle) and 5 s (bottom) time step resolution for jobs from different queues, in ascending order of submission time

For a shorter time-to-solution, it is desirable to reduce the resolution, at the cost of increasing the overall error. Such errors increase as (re-)scheduling only happens at the end of every time step. The time passed during the time step leaves resources idle if jobs finish within that window. When resources remain unused, it increases the risk of congestion, and later jobs may become delayed further (error propagation). Figure 5 shows an example for week 6 (starting within day 39). Even though the node utilization of the simulated system with both 5 and 60 s resolutions is similar, the number of waiting jobs increases in the latter. This results in a gradually increasing wait time for jobs, offsetting the execution time of successive jobs and skewing the results. Therefore, reducing the resolution is not always appropriate, and needs a case-by-case decision. We have found that 5 s time steps offer the best tradeoff between time-to-solution and accuracy. However, some weeks with longer phases of high cluster loads likely require a shorter time step at the cost of an increased simulation time.

4. CONCLUSION

We have demonstrated a method for approximating non-trivial PBS Professional configurations using an additive job priority formula with different queue priorities and constraints, including fairsharing and backfilling with MS. With this setup, we are able to simulate HPC job scheduling effects with historic workloads from a large-scale 2 PetaFLOP/s cluster, including planned and unplanned node down times. Even though PBS does not have any simulation capabilities, early results suggest MS is a valuable simulation tool for PBS. The quality of approximation is sufficient for analysis of changes to a PBS production system, and allows a trade-off between accuracy and time-to-solution by varying the time step resolution. In addition, MS offers the option to replace the job scheduler from PBS, which would further minimize the remaining gaps between simulation and real workloads by using identical configurations for the real system and the simulation.

Limitations currently apply to the quality of historic workloads, such as inconsistent node downtime information, which requires manual validation. In addition, the assumption of ideal job dispatching adds a larger error for jobs with a short wall time. Currently missing information about the real dispatching time and job dependencies requires improvement to reduce errors further. Whilst it is easy to collect such additional data on a real system, two deficits require an extension of MS itself. The simulation of unexpected node downtime currently uses a simplified approach by terminating the entire job associated to nodes taken offline. This is not always correct and, together with the modeling of job dependencies, MS currently does not have suitable functionalities built-in.

In future projects, we plan to use the discussed simulation framework to study the effects of applying changes to the PBS job scheduler, such as changes to queue priorities or constraints. Furthermore, we plan to study the effects of job dependencies and different scheduling strategies.

At our *Github* repository (Zitzlsberger, 2018) we provide a *Dockerfile* to build MS with the changes we applied. We also provide the environment we used for our simulations. The environment contains the MS configuration and the resource trace file, as well as an example implementation for handling node downtimes. Interactive versions of the diagrams used in this paper are also available for individual study.

ACKNOWLEDGEMENT

This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070", by the ERDF in the IT4Innovations national supercomputing center - path to exascale project (CZ.02.1.01/0.0/0.0/16_013/0001791) within the OPRDE , and ANTAREX, a project supported by the EU H2020 FET-HPC program under grant agreement No. 671623 .

REFERENCES

Adaptive Computing, 2014, "Maui Scheduler, Release 3.3.1," https://github.com/LabAdvComp/maui/ [24-10-2018].

Advanced Computing, 2018, "Torque Resource Manager," http://www.adaptivecomputing.com/products/open-source/torque/ [24-10-2018].

Altair PBS Works, 2011, "PBS Professional 13.1, Administrator's Guide," https://pbsworks.com/pdfs/PBSProAdminGuide13.1.pdf [24-10-2018].

Altair, 2018, "PBS Works Suite, PBS Professional," https://pbsworks.com/ [24-10-2018].

Argonne National Laboratory, 2018, "Qsim," https://trac.mcs.anl.gov/projects/cobalt/wiki/qsim [24-10-2018].

Bode, B. Halstead, D. M., Kendall, R., Lei, Z., and Jackson, D., 2000, "The portable batch scheduler and the maui scheduler on linux clusters," in Proceedings of the 4th Annual Linux Showcase & Conference - Volume 4, ALS'00, (Berkeley, CA, USA), pp. 27–27, USENIX Association.

Buyya, R., and Murshed M., 2002, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," Concurrency and Computation: Practice and Experience (CCPE, vol. 14, no. 13, pp. 1175–1220.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F., June 2014, "Versatile, scalable, and accurate simulation of distributed applications and platforms," Journal of Parallel and Distributed Computing, vol. 74, pp. 2899–2917.

Ding, Z. H., and Li, X. Q., 2017, "IBM Spectrum LSF 10 — Taking the guesswork out." https://developer.ibm.com/storage/2016/10/19/ibm-spectrum-lsf-10-taking-the-guesswork-out/[24-10-2018].

IBM, 2016, "IBM Spectrum LSF, Version 10 Release 1.0, Configuration Reference."

IT4Innovations, 2018, "Job Scheduling," https://docs.it4i.cz/salomon/job-priority/ [24-10-2018].

Jackson, D. B., Jackson, H. L., and Snell, Q. O., 2000, "Simulation based hpc workload analysis," in Parallel and Distributed Processing Symposium., Proceedings 15th International, IEEE.

Jackson, D. B., Snell, Q., and Clement, M. J., 2001, "Core Algorithms of the Maui Scheduler," in Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing, JSSPP '01, (London, UK, UK), pp. 87–102, Springer-Verlag.

Klusáček, D., and Rudová, H., 2010, "Alea 2 – Job Scheduling Simulator," in Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010), ICST.

Klusáček, D., and Chlumský, V., 2016, "Planning and metaheuristic optimization in production job scheduler," in Job Scheduling Strategies for Parallel Processing (N. D. Walfredo Cirne, ed.), pp. 1–19, Neuveden.

Lucero, A., 2011, "Slurm Simulator, Slurm User Group Meeting, September 2011."

Simakov, N. A., Innus, M. D., Jones, M. D., DeLeon, R. L., White, J. P., Gallo, S. M., Patra, A. K., and Furlani, T. R., 2017, “A Slurm simulator: Implementation and parametric analysis,” in PMBS@SC, vol. 10724 of Lecture Notes in Computer Science, pp. 197–217, Springer.

Srinivasan, S., Kettimuthu, R., Subramani, V., and Sadayappan, P., 2002, “Characterization of backfilling strategies for parallel job scheduling,” in ICPP Workshops, pp. 514–522, IEEE Computer Society.

Supercluster R&D Group, 2002, “Maui Administrator’s Guide, Maui 3.2,” http://docs.adaptivecomputing.com/maui/pdf/mauiadmin.pdf [24-10-2018].

Top500, 2018, “Top500 List,” from June 2018, https://www.top500.org/site/50561 [24-10-2018].

Trofinoff, S., and Benini, M., 2015, “Using and Modifying the BSC Slurm Workload Simulator, Slurm User Group Meeting, September 2015,” https://github.com/beninim/slurm_simulator/ [24-10-2018].

Wang, K., Brandstatter, K., and Raicu, I., 2013, “Simmatrix: Simulator for many-task computing execution fabric at exascale,” in Proceedings of the High Performance Computing Symposium, HPC ’13, (San Diego, CA, USA), pp. 9:1–9:9, Society for Computer Simulation International.

Wang, K., 2014, “SimMatrix,” https://github.com/kwangiit/SimMatrix/ [24-10-2018].

Yang, X., et al., 2013, "Integrating Dynamic Pricing of Electricity into Energy Aware Scheduling for HPC Systems", Proc. of SC'13.

Zitzlsberger, G., 2018, “Maui simulator configuration and environment for IT4Innovations’ Salomon cluster,” https://github.com/It4innovations/Maui-Simulation [24-10-2018]