HPC Andragogy: Automating Batch Scheduler Feedback

Kyriakos Tsoukalas
Colgate University
ktsoukalas@colgate.edu

ABSTRACT

This paper proposes a monitoring system that emails feedback to users about submitted jobs and has the capability to stop and resubmit jobs to a batch scheduler. The proposed system has been implemented for a small supercomputing environment with a mix of high-performance and high-throughput computing jobs. User feedback includes alerts for over- and under-utilization of CPU and physical memory. This paper also discusses how predefined system thresholds were chosen and proposes three algorithms. An algorithm for the proposed monitoring system and two algorithms for the prediction of CPU and physical memory utilization. The latter algorithms are based on users' input of the identification string (job ID) of a similar job that should have finished execution without errors. Lastly, a git repository is shared to make the code accessible for review.

KEYWORDS

HPC, HTC, Andragogy, Batch Scheduler, Feedback, Automation

1 INTRODUCTION

Batch scheduling on systems that combine High-Performance Computing (HPC) and High-Throughput (HTC) Computing has many challenges to overcome. The batch scheduler has a number of differing goals based on the mix of HPC and HTC jobs. Furthermore, resource optimization involves stohastic processes such as the demand for resources, the dynamic availability of resources, and the variability of job processing duration. Resource fragmentation is a particular challenge in systems with both HPC and HTC loads [3]. Moreover, different computer clusters may have different operating systems and support different scientific software.

An important factor in resource fragmentation is users' requested resources for their jobs. This paper proposes a system for monitoring the utilization of resources in comparison to the resources allocated to each running job. The proposed monitoring system automates feedback via email when a running job is over- or under-utilizing its allocated resources. The purpose of the proposed system is twofold. Firstly, to stop jobs that grossly under-utilize allocated resources and attempt to automatically resubmit them requesting appropriate resources. Secondly, to alert users for jobs that over-utilize or marginally under-utilize resources, prompting them to review the statistics for each job with misallocated resources. The focus of resource allocation extends beyond processing individual jobs. Variability in CPU and physical memory utilization affects all jobs submitted to a batch scheduler. Furthermore, possibly for long periods of time, each job's maximum and minimum resource utilization may vary greatly from the average resource utilization. Ideally, jobs that have known preparatory steps, such as moving files or preprocessing data, should be executed in a chain of jobs as opposed to a single job. Each job in the chain with an appropriate allocation of resources. Thus, the Colgate Supercomputer emails users when a predefined discrepancy (threshold) between allocated (requested) and utilized resources of CPU and physical memory has been exceeded.

The feedback emails aim to let researchers know when their job submissions are over- and under-utilizing allocated (requested) CPU and physical memory resources. The andragogical aspect of the proposed monitoring system is an early familiarization of new users of a supercomputer to the important aspect of resource allocation. More importantly, familiarization with the fact that the Colgate Supercomputer allows over-utilization when less resources have been requested and allocated. The overall goal is that users receive frequent feedback in order to learn to be more conservative with their requests for resources. Having knowledge that their jobs will be allowed to utilize more resources when then are able to, while a batch scheduler will not be able to allocate any allocated but unused resources to newly submitted jobs.

2 PROPOSED MONITORING SYSTEM

2.1 Rationale

The proposed system has been implemented on the Colgate Supercomputer [1]. Contemplate the following supercomputing academic context. A supercomputer is used by faculty, staff, and students alike, both for research and educational activities. The same supercomputer processes both HPC and HTC jobs, which include interactive applications and web services. Users of the supercomputer submit jobs that are queued to be processed while processing duration varies from a few minutes to weeks.

In the previously aforementioned context, it is extremely important to decrease the discrepancy between allocated (requested) and utilized resources. Firstly, to increase the number of jobs processed daily, weekly, monthly, and yearly. Secondly, to decrease the number of jobs competing for the same resources. Therefore, it is important to inform the users of a supercomputer about the difference between over-allocating resources versus under-allocating resources.

Over-allocating resources results in inability to schedule more jobs because there are unused but allocated resources that could have been allocated to newly submitted jobs. Under-allocating resources results in the possibility that multiple jobs may experience slower processing due to competing for the same resources. However, a batch scheduler may be configured to not fill a node when

CPU or memory utilization is higher that allocated, unless there is no other node left with enough available (unallocated) resources. Moreover, jobs could be allowed to utilize more than the allocated resources when there are more available.

2.2 Proposed Algorithms

A git repository is shared to make the code accessible for review [10]. Algorithms 1 and 2 are used to predict utilization of CPU cores and physical memory, respectively, given the identification string (job ID) of a job with similar characteristics that has finished without errors. Many users tend to submit groups of jobs at a time, which have similar characteristics, such as the same processing script but with different parameters. Algorithm 3 describes the proposed monitoring system. The proposed monitoring system could run as a CRON job (a time-based job scheduler in Unix-like systems).

Algorithm 1 Average Used CPU cores

```
Input: Job IDs, status, owner, and used CPU percentage
Output: Average Used CPU cores
$C_{max} \leftarrow 3$                                  ▷ Max counter
$I_{max} \leftarrow 11$                                 ▷ Max iterations
$I \leftarrow 0$                                        ▷ Iterations
$C \leftarrow 0$                                        ▷ Counter
ID ← (Job ID)                                           ▷ Job ID
$T_{cpu} \leftarrow 0$                                  ▷ Total used CPU percentage
$J_{cpu} \leftarrow 0$                                  ▷ Used CPU percentage
while $C < C_{max}$ and $I < I_{max}$ do
    if job with ID has status = F and owner = user then
        $T_{cpu} \leftarrow T_{cpu} + J_{cpu}$
        $I \leftarrow I + 1$
    else
        $C \leftarrow C + 1$
    ID ← ID − 1
Print: round$\left(\frac{T_{cpu}}{100I}\right)$
```

3 EMAIL FEEDBACK AUTOMATION

The Colgate Supercomputer automates feedback by monitoring all running jobs to alert users about under- or over- utilization of resources. Note that calculations should take into consideration how memory is reported (for example in MiB or MB). The monitoring system does not assign weights based on elapsed processing time of jobs, because the andragogical aspect of the proposed system aims to familiarize users with the implications of over-allocating resources. Email is a common medium for sending textual feedback.

The automation requires the capability to send emails. A common program for sending emails in Linux operating systems is mailx.

Figure 1 shows a basic Grafana dashboard for CPU utilization and Figure 2 shows a basic Grafana dashboard for memory utilization [5].

Algorithm 2 Average Used Physical Memory

```
Input: Job IDs, status, owner, and used physical memory
Output: Average Used Physical Memory
$C_{max} \leftarrow 3$                                  ▷ Max counter
$I_{max} \leftarrow 11$                                 ▷ Max iterations
$I \leftarrow 0$                                        ▷ Iterations
$C \leftarrow 0$                                        ▷ Counter
ID ← (Job ID)                                           ▷ Job ID
$T_{mem} \leftarrow 0$                   ▷ Total used physical memory in bytes
$J_{mem} \leftarrow 0$                   ▷ Used physical memory in bytes
procedure ConvertMemory($m$)
    if $m > 1$gb then
        Print: $m_{gb}$gb
    else if $m > 1$mb then
        Print: $m_{mb}$mb
    else if $m > 1$kb then
        Print: $m_{kb}$kb
    else
        Print: $m$
while $C < C_{max}$ and $I < I_{max}$ do
    if job with ID has status = F and owner = user then
        $T_{mem} \leftarrow T_{mem} + J_{mem}$
        $I \leftarrow I + 1$
    else
        $C \leftarrow C + 1$
    ID ← ID − 1
ConvertMemory($T_{mem}$)
```

Algorithm 3 Monitoring System

```
Input: Running jobs info ($\mathcal{F}$), logged job IDs ($\mathcal{L}$), job status,
owner, used walltime, used CPU percentage, and used physical
memory
Output: Action log
$c \leftarrow 2$                        ▷ Threshold for CPU core difference
$m \leftarrow 8$ GB                     ▷ Threshold for physical memory difference
procedure AllRunningJobs($\mathcal{F}$)
    for each Job Job_With_ID $\in \mathcal{F}$ do
        Process Job Log for ID
        if CPU cores requested - used > 2 then
            Increase counter in Job Log
            if counter > 1 then
                Stop Job_With_ID and email Owner
                Submit new job with Used_CPU_Cores
procedure AllLoggedJobs($\mathcal{L}$)
    for each Job Job_With_ID $\in \mathcal{L}$ do
        if $0 <$ CPU cores requested - used $< c$ then
            Email Owner
        if CPU cores requested - used $< 0$ then
            Email Owner
        if $0 <$ Physical memory requested - used $> m$ then
            Email Owner
        if Physical memory requested - used $< 0$ then
            Email Owner
AllRunningJobs($\mathcal{F}$)
AllLoggedJobs($\mathcal{L}$)
```

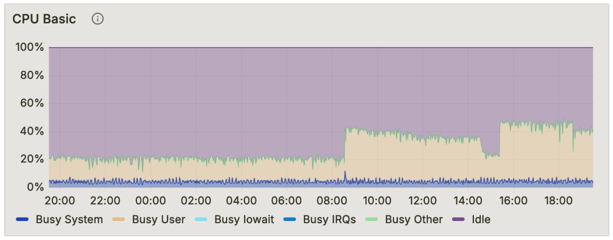

Figure 1: Example Grafana Dashboard for CPU Utilization

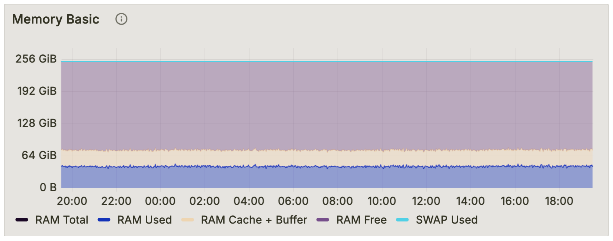

Figure 2: Example Grafana Dashboard for Memory Utilization

3.1 Email Feedback Templates

3.1.1 Example for Major CPU Under-utilization. The email subject is "CPU under-utilization Alert: Job xxxxx.<clustername> on Node n03 was stopped".

Job ID: xxxxx.<clustername>
Node: n03
Owner: <username>
Elapsed Time: 0h45m
Requested CPU cores: 31
Unused CPU cores: 4
Suggested CPU cores: 27
Metadata: qstat -xf xxxxx.<clustername>
Monitor: Monitor: https://monitor.domain.edu

This is an automated message.

Job xxxxx.<clustername> on Node n03 requested 31 CPU cores but was using an average of 2723% CPU after 45 minutes. Job xxxxx.<clustername> was stopped but resubmitted requesting 27 CPU cores. The new job's ID is yyyyy.<clustername>. It is suggested to request 27 CPU cores for future similar jobs instead of 31. Please review both jobs.

3.1.2 Example for Minor CPU Under-utilization. The email subject is "CPU under-utilization Alert: xxxxx.<clustername> on Node n03".

Job ID: xxxxx.<clustername>
Node: n03
Owner: <username>
Elapsed Time: 1d4h24m
Requested CPU cores: 31
Unused CPU cores: 2
Suggested CPU cores: 29
Metadata: qstat -xf xxxxx.<clustername>
Prediction of CPU cores: calccpu xxxxx.<clustername>
Monitor: https://monitor.domain.edu

This is an automated message.

Job xxxxx.<clustername> on Node n03 used 2903% CPU, therefore it is suggested to request 29 CPU cores for future similar jobs instead of 31. Please review your job.

3.1.3 Example for non Multi-threading. The email subject is "CPU under-utilization Alert: xxxxx.<clustername> on Node n03".

Job ID: xxxxx.<clustername>
Node: n03
Owner: <username>
Elapsed Time: 5h27m
Requested CPU cores: 3
Unused CPU cores: 2
Suggested CPU cores: 1
Metadata: qstat -xf xxxxx.<clustername>
Monitor: https://monitor.domain.edu

This is an automated message.

Job xxxxx.<clustername> on Node n03 used 36% CPU and did not multi-thread. When a job is not multi-threading it cannot use more that 1 CPU core and you will be notified when a job used more than the requested CPU cores. Therefore it is important to request only 1 CPU core for future similar jobs. Please review your job.

3.1.4 Example for CPU Over-utilization. The email subject is "CPU over-utilization Alert: xxxxx.<clustername> on Node n03".

Job ID: xxxxx.<clustername>
Node: n03
Owner: <username>
Elapsed Time: 13h13m
Requested CPU cores: 10
Unused CPU cores: -7
Suggested CPU cores: 17
Metadata: qstat -xf xxxxx.<clustername>
Prediction of CPU cores: calccpu xxxxx.<clustername>
Monitor: https://monitor.domain.edu

This is an automated message.

Job xxxxx.<clustername> on Node n03 used an average of 1707% CPU, therefore it is suggested to request 17 CPU cores for future similar jobs instead of 10. Please review your job.

3.1.5 Example for Memory Under-utilization. The email subject is "Memory under-utilization Alert: xxxxx.<clustername> on Node n03".

```
Job ID: xxxxx.<clustername>
Node: n03
Owner: <username>
Elapsed Time: 13h37m
Requested physical memory: 83886080kb
Unused physical memory: 50gb
Suggested physical memory: 35gb
Prediction of physical memory: calcmem xxxxx.<clustername>
Metadata: qstat -xf xxxxx.<clustername>
Monitor: https://monitor.domain.edu

This is an automated message.

Job xxxxx.<clustername> on Node n03 used 29gb of
physical memory and 60gb of virtual memory, there-
fore it is suggested to request 35gb of physical mem-
ory for future similar jobs instead of 83886080kb. Please
review your job.
```

3.1.6 Example for Memory Over-utilization. The email subject is "Memory over-utilization Alert: xxxxx.clustername on Node n03".

```
Job ID: xxxxx.<clustername>
Node: n03
Owner: <username>
Elapsed Time: 26h13m
Requested physical memory: 30gb
Unused physical memory: -21gb
Suggested physical memory: 55gb
Prediction of physical memory: calcmem xxxxx.<clustername>
Metadata: qstat -xf xxxxx.<clustername>
Monitor: https://monitor.domain.edu

This is an automated message.

Job xxxxx.<clustername> on Node n03 used 51gb of
physical memory and 83gb of virtual memory, there-
fore it is suggested to request 55gb of physical mem-
ory for future similar jobs instead of 30gb. Please re-
view your job.
```

4 PREDEFINED SYSTEM THRESHOLDS

On the Colgate supercomputer, jobs utilizing 1 to 3 CPU cores are considered HTC. There is a grey zone of CPU utilization between 4 and 7 CPU cores. A job which requires more that 12 hours of processing and an average of 4 CPU cores or more is considered HPC. Utilization of 8 or more CPU cores is also considered HPC. The goal of achieving a fair resource allocation between HPC and HTC job loads is a complicated matter.

The threshold for stopping a running job has been set to 3 cores. During job processing, jobs are not allowed to use, on average, less than 2 of the total CPU cores allocated. The proposed monitoring system will stop such jobs and try to resubmit them using the average CPU utilization within 25-45 minutes of job processing.

The proposed monitoring system will email alerts to users for jobs that use more or less than 8GB of physical memory than the memory requested and thus allocated by the batch scheduler.

5 EARLY RESULTS

The proposed monitoring system has been employed for about a year on the Colgate Supercomputer. The early findings from its use informed the supercomputer's documentation and a progressive refinement of the system during its trial period. The important findings from the trial period are presented in this section.

New users who submitted jobs requesting many CPU cores but their jobs did not multi-thread, received feedback emails. As a result, their subsequent submissions quickly changed to requesting just 1 CPU core, but also 2 to testing multi-threading. Testing multi-threading is not necessary as the proposed system will notify users when a job of theirs used more than the requested CPU cores. The relevant email alert was modified to clarify this capability in order to reduce unused but allocated CPU cores.

Some experienced users reached out to discuss the metric of average CPU utilization with regard to the maximum utilization during job execution. A result of these discussions was to update the documentation of the Colgate Supercomputer to emphasize the difference and how to interpret the relevant metadata given by the batch scheduler per job.

Another discussion with experienced users was about the differentiation between physical and virtual memory. Many users request physical memory that the batch scheduler allocates, even though their jobs end up using only a fraction of the allocation. Although the availability of large amounts of physical memory may help alleviate this problem, it remains an important issue in HPC andragogy. The relevant email feedback template was edited to clearly state the use of both physical and virtual memory.

The capability of the proposed monitoring system to stop and attempt to resubmit jobs that are under-utilizing allocated resources has been received with some skepticism from older users. Nevertheless, the system was able to attract attention to the important difference between under-allocating and over-allocating resources, and to the fact that jobs are allowed to use more resources than what was allocated by the batch scheduler.

Researcher training for supercomputing environments is not an easy endeavor. In a given academic institution, there can be a lot of different computing backgrounds as well as levels of experience with scientific software. The proposed monitoring system provides feedback as needed, rather than requiring users to undergo software carpentry and batch scheduler training to become familiar with a supercomputing environment, especially considering that many users either do not read the available documentation or only skim through it.

The most important result was that while new users were becoming familiar with the Colgate Supercomputer, any submitted jobs were restricted from excessively over-allocating resources.

6 DISCUSSION

In an effort to separate HPC and HTC jobs, the Colgate Supercomputer runs interactive applications on dedicated batch scheduling queues. Most interactive applications will run with a predefined number of CPU cores and physical memory. Achieving some level of separation between HTC and HPC job loads will reduce complexity and may result in fewer unused allocated resources while keeping higher job cancellation thresholds.

The thresholds chosen for the trial period, 3 CPU cores and 8GB of physical memory, would ideally be lowered in the future, as the resubmission process becomes more efficient. A common problem is that users could delete or rename files used in a job after its submission.

Formal training [4], informal training [7, 8], and educational activities [6] are based on proven paradigms, however they lack granularity due to the fact that learning is grouped in training modules, learning topics, or instructional activities. Consistent feedback for experiential learning provides increased granularity by matching one to one the job submissions of users of HPC/HTC systems.

A useful addition to the proposed monitoring system would be to automate a monthly report to each user that would present statistics about under- and over- utilization of requested resources. Supercomputing users would benefit from automatically receiving reports that highlight steps to improve subsequent job submissions, regarding resource requests.

Batch schedulers, such as Slurm [9], provide the statistics to be leveraged by the proposed monitoring system, hence shell scripting was used to implement the proposed monitoring system for the trial period. The approach is considering small supercomputers with less than 500 nodes. Newer resource managers have been proposed for supercomputers with more than 10k nodes [2]. A future integration with a large language model (LLM) would benefit from a python implementation.

The proposed monitoring system could assign predefined labels to each discrepancy between allocated (requested) and utilized resources (such as CPU cores and physical memory). An unsupervised machine learning algorithm, such as Bayesian learning, could then be employed to create user profiles regarding resource requests.

7 CONCLUSION

The proposed monitoring system offers a solution for managing resource misallocation leveraging the fault tolerance in scenarios of resource over-utilization. Supercomputing users can learn to be more conservative with their resource requests when there is little to no penalty for over-utilizing resources allocated by a batch scheduler.

Additionally, the proposed system is designed to efficiently reclaim resources that are allocated but under-utilized, particularly in an environment that integrates both HPC and HTC workloads. By addressing both over-utilization and under-utilization, the proposed system ensures more effective resource management, and promotes HPC andragogy.

Current work includes a refactoring of the system's code to query metadata from both PBS Pro and the Slurm batch schedulers. It also includes the development of Graphical Processing Unit (GPU) resource monitoring.

Future work includes integrating the monitoring system with Colgate's Supercomputer's AI chatbot which assists with documentation and coding. This integration aims to develop pipelines for more complex resource utilization predictions by analyzing previous job scripts as well as the scripts to be submitted. Parsing new scripts will also enable raising alerts about job dependencies and required POSIX permissions.

REFERENCES

[1] Colgate University. 2024. *Colgate Supercomputer (Partially supported by NSF grant OAC-2346664).* https://rcd.colgate.edu Accessed: 2024-08-12.

[2] Yiqin Dai, Yong Dong, Kai Lu, Ruibo Wang, Wei Zhang, Juan Chen, Mingtian Shao, and Zheng Wang. 2022. Towards scalable resource management for supercomputers. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Dallas, Texas) *(SC '22)*. IEEE Press, Article 24, 15 pages.

[3] Oliver Freyermuth, Peter Wienemann, Philip Bechtle, and Klaus Desch. 2021. Operating an HPC/HTC cluster with fully containerized jobs using HTCondor, Singularity, CephFS and CVMFS. *Computing and Software for Big Science* 5, 1 (2021), 9.

[4] Kai Himstedt, Nathanael Hubbe, Julian Kunkel, Hinnerk Stuben, Deutsches Klimarechenzentrum, Thomas Ludwig, Stephan Olbrich, Matthias Riebisch, Sandra Schröder, and Markus Stammberger. 2018. An HPC certification program proposal meeting HPC users' varied backgrounds. *Concept paper https://wr. informatik. uni-hamburg. de/research/projects/pecoh/start, DKRZ and RRZ* (2018).

[5] janakverma. 2024. *Linux Exporter Node.* https://grafana.com/grafana/dashboards/14513-linux-exporter-node/ Accessed: 2024-09-09.

[6] Bryan Johnston, Lara Timm, and Mabatho Hashatsi. 2023. Delivering Digital Skills across the Digital Divide. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC23).*

[7] Julia Mullen, Chansup Byun, Vijay Gadepally, Siddharth Samsi, Albert Reuther, and Jeremy Kepner. 2017. Learning by doing, High Performance Computing education in the MOOC era. *J. Parallel and Distrib. Comput.* 105 (2017), 105–115. https://doi.org/10.1016/j.jpdc.2017.01.015 Keeping up with Technology: Teaching Parallel, Distributed and High-Performance Computing.

[8] Julia Mullen and Lauren Milechin. 2023. A Data Driven Approach to Informal HPC Training Evaluation. In *Practice and Experience in Advanced Research Computing.* 378–381.

[9] Nikolay A. Simakov, Robert L. DeLeon, Martins D. Innus, Matthew D. Jones, Joseph P. White, Steven M. Gallo, Abani K. Patra, and Thomas R. Furlani. 2018. Slurm Simulator: Improving Slurm Scheduler Performance on Large HPC systems by Utilization of Multiple Controllers and Node Sharing. In *Proceedings of the Practice and Experience on Advanced Research Computing: Seamless Creativity* (Pittsburgh, PA, USA) *(PEARC '18)*. Association for Computing Machinery, New York, NY, USA, Article 25, 8 pages. https://doi.org/10.1145/3219104.3219111

[10] Kyriakos Tsoukalas. 2024. *Supercomputer Monitoring System.* https://github.com/tsoukalask/jobcheck Accessed: 2024-09-09.