HPC System Lifetime Story: Workload Characterization and Evolutionary Analyses on NERSC Systems

Gonzalo P. Rodrigo, Per-Olov Östberg,
Erik Elmroth
Dept. Computing Science, Umeå University
SE-901 87, Umeå, Sweden
{gonzalo, p-o, elmroth}@cs.umu.se

Katie Antypas, Richard Gerber,
Lavanya Ramakrishnan
Lawrence Berkeley National Lab
Berkeley, CA 94720, USA
{kantypas, ragerber,
lramakrishnan}@lbl.gov

ABSTRACT

High performance computing centers have traditionally served monolithic MPI applications. However, in recent years, many of the large scientific computations have included high throughput and data-intensive jobs. HPC systems have mostly used batch queue schedulers to schedule these workloads on appropriate resources. There is a need to understand future scheduling scenarios that can support the diverse scientific workloads in HPC centers. In this paper, we analyze the workloads on two systems (Hopper and Carver) at the National Energy Research Scientific Computing (NERSC) Center. Specifically, we present a trend analysis towards understanding the evolution of the workload over the lifetime of the two systems.

Categories and Subject Descriptors

D4.1 [Operating Systems]: Process Management—*Scheduling*

Keywords

Scheduling; workload; trend analysis; HPC

1. INTRODUCTION

Traditionally, scientific applications running at HPC centers have been large monolithic MPI applications which require high bandwidth, low latency interconnects. However, large scientific computations that include high-throughput, data-intensive jobs, and stream-processing are increasingly becoming more common in HPC centers. These applications are assigned HPC resources through batch queue schedulers with objectives such as short wait times, high utilization, and selective prioritization [5].

The diversity of workloads in supercomputing centers requires us to investigate the right workload scheduling model for the next-generation systems addressing the needs of diverse workloads. As a first step, we need to understand the

evolution of the workload on the current systems in depth. Previous works have analyzed workloads on various grid [6] and cloud [3] systems. However, these works and associated comparisons are based on single points in time of HPC workloads that are several years before the current trend in workloads.

In this paper, we analyze the workloads at the National Energy Research Scientific Computing Center (NERSC), a supercomputing center that supports the broad scientific workload of the Department of Energy Office of Science. We consider two systems: Carver and Hopper, selected due to differences in their hardware and timeline characteristics. Carver is a traditional high performance Linux cluster while Hopper is a specialized Cray supercomputer with a custom interconnect. These systems allow us to capture the workloads on high-end clusters and supercomputers over a period of four years.

Specifically, in this paper we provide an evolutionary analysis of the NERSC systems workloads. We study the trend in job geometry (i.e. allocated cores, wall clock time and consumed core-hours), job wait times, and wall clock time accuracy. Our analyses help in understanding the evolution of workload patterns over the lifetime of each system to facilitate short- and long-term decisions at HPC centers.

2. BACKGROUND

In this section, we present the background on workloads and current scheduling policies in HPC centers.

2.1 Evolution of HPC Workloads

Data-intensive applications are becoming increasingly more common in HPC workloads, partly following the increase of HPC use in scientific domains such as biology or astrophysics [8]. We are seeing an increase in the need for stream processing of large amounts of experimental data. For example, the Advanced Light Source is a particle accelerator that produces high quality energy beams. Large amounts of data produced are transferred to NERSC to be processed. A number of these experiments would benefit from processing the data in real-time to have low latency feedback [2]. To support this, advance resource reservations could be a means, but they are known to significantly reduce the overall utilization. Such changes in applications over time require us to analyze the workloads at supercomputers to understand their characteristics.

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the United States Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

HPDC'15, June 15 - 19, 2015, Portland, OR, USA.

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 978-1-4503-3550-8/15/06.

DOI: <http://dx.doi.org/10.1145/2749246.2749270>.

System	Vendor	Model	Built	Nodes	Cores/N	Cores	Memory	Network	TFlops/s	Service	Charging
Hopper	Cray	XE6	2010	6,384	24	154,216	212 TB	Gemini	144	Jan'10	Jan'10
Carver	IBM	iDataPlex	2010	1,120	8/12/32	9,984	147 TB	Infiniband	106.5	Apr'10	May'10

Table 1: Hopper and Carver characteristics

2.2 Scheduling

HPC systems are designed to run multiple jobs in parallel over large compute infrastructures. Their schedulers commonly use the FCFS (First-Come, First-Served) model [4], selecting jobs in arrival order. Additionally, they use backfilling [7] to avoid low resource utilization. Jobs submitted to the queue can have different priorities associated with them. Priority of a job is used by the scheduler in each scheduling pass to determine the speed at which a job makes its way through the queue (i.e. jobs belonging to higher priority users or projects are executed faster).

The quality of the results of backfilling algorithms depend on user wall clock estimations [4]. Under and over-estimation of job execution times may lead to lower utilization of systems, motivating the need to study wall clock accuracy (relationship between estimated and actual wall clock, more in Section 4.3).

3. METHODOLOGY

In this section, we present the system and workloads in focus for our investigation and elaborate on the key parameters studied.

3.1 Systems Description

NERSC is a HPC center at the Lawrence Berkeley National Lab, that provides computing infrastructure and tools for scientists performing research of relevance to the United States Department of Energy (DOE). Over 5000 users and 700 distinct projects use the NERSC supercomputing infrastructures [1]. The workload is composed of applications that belong to a wide range of scientific fields including Chemistry, Material Science, Climate Research, Astrophysics, Life Sciences, and Nuclear Physics.

This work considers two systems at NERSC (summary of characteristics in Table 1): Hopper, built on multicore processors, packaged together on customized blades and connected with a high speed proprietary interconnect; and Carver, a high performance Linux cluster with a fat-tree Infiniband interconnect matrix. Both systems use the Moab scheduler and the Torque [9] resource manager.

NERSC users receive an allocation of compute hours that can be used to run on multiple NERSC systems, including Carver and Hopper. Each system has its own charging factor depending on its performance across a standard set of benchmarks. Both Carver and Hopper are heavily utilized systems ($\sim 90\%$ for the years considered in this study)

During the deployment and initial testing phase, users run jobs without charges on their allocations. Users' behavior on the systems can be different during these times and hence it is important to note them: Hopper was deployed in December 2009 with a small Phase 1 system that was later upgraded to the full system in November 2010. Users started getting charged for running jobs in April 2011.

Carver was deployed in early 2010 and set into production in May 2010. Additionally, Carver was expanded in January 2012. Carver will be decommissioned in September 2015 and

its jobs will be migrated to Edison, a new Cray system similar to Hopper. The staff at NERSC tweak scheduling and queuing policies a few times per year based on the characteristics of the system, user input, scientific requirements, DOE priorities, and observations of job backlog and usage across systems. Our analysis considers the period January 2010 to June 2014 for Hopper and Carver.

In addition to serving the typical workloads, Carver provides a *serial* queue: a special queue that allows users to submit and execute jobs with a very low degree of parallelism (i.e. one node). Since our analyses, serial queues have also been added to Hopper.

3.2 Data source

All workload analysis is performed on the job summary entries from the systems' Torque logs. The data includes 4.5 years and 4,326,870 jobs for Hopper and 4.5 years and 9,508,054 jobs for Carver. The raw data size is 45 GB, which, after filtering and parsing, is reduced to 6 GB of net data.

3.3 Analysis variables

Our analysis focuses on understanding the evolution of the two systems' lifetime workload. First, we try to understand the workload, i.e. the job geometry (wall clock time and degree of parallelism). The job geometry defines the upper bound of application resource requirements.

Second, we study the trend in job wait time and wall clock time accuracy. Job wait time represents the time a job spends in queue waiting to start its execution, and is a measure of the system's degree of overload. The wall clock estimation accuracy has an important impact on the scheduler decision quality.

The evolutionary trends of these variables are interesting as their characterization can help build more suitable future systems from the perspective of infrastructure and resource management.

3.4 Time Period of Trend Analysis

The trend analysis studies the workload in sequential time periods. The size of the periods is calculated using a Fourier transform analysis on the number of tasks submitted per hour, which identifies cycles in the job submission [10] and thus, the user behavior. The most powerful cycles correspond to the periods of one day, one week, three month and six months, matching calendar work periods and allocation year. Each project has a number of core hours to be consumed during a year, divided in four allocation quarters in which the project has to consume (or forfeit) the corresponding allocated time. The yearly allocation policy motivated the choice of one year as the analysis time period for the trend analysis.

4. TREND ANALYSIS

In this section, we present our trend analyses on job geometry, job wait time, and wall clock time accuracy.

Figure 1: Job wall clock time for each workload year. Trend: Hopper jobs become longer, Carver jobs shorter. Majority of jobs under one hour.

Figure 2: Allocated number of cores for each workload year. Trend: Hopper jobs allocate less cores. In 2011-2013, most Carver jobs used one core.

4.1 Job geometry

The evolution of the first job geometry variable is presented in Figure 1 as a box plot of job wall clock time for each system in each year. Hopper shows a significantly low wall clock median in 2010 (< 1 minute), which might be related to the fact that it was a smaller testbed system that year. In 2011, the median increased to ~ 5 minutes and subsequently increased to ~ 12 minutes by 2014. Carver shows a different trend: the median, upper and lower quartile decrease effectively over the period studied. The median decreased from ~ 20 minutes (2010) to ~ 6 minutes (2014). However, there is some variation from year to year: It is observed that in the first year in production, Carver ran longer jobs than Hopper, a fact that slowly changed in 2014 when Hopper ran longer jobs than Carver. More generally, Hopper presents fairly short jobs, as the highest upper quartile is around the one hour value. Carver presents a similar behavior as the upper quartiles of the last years are under one hour.

The evolution of the width of jobs (number of allocated cores per job) is shown in Figure 3. For Hopper, the median decreases from 100 cores (2010) to under 30 cores (2014).

Figure 3: Allocated core hours for each workload year. Trend: No changes on Hopper. Carver jobs become smaller.

Figure 4: Jobs' wait time evolution for each workload year. Trend: All systems increase wait time. Carver lower wait time in 2011.

Carver presents an opposite pattern. Except for 2010, the median of the rest of the years is one core, showing the predominance of single core serial jobs. In 2014, the upper quartile increased to 8 cores.

The core-time i.e., total clock time across all cores was also studied. In the case of Hopper, it remains nearly the same through time with a median of ~ 20 core hours and the upper quartile slightly under 200 core hours in most years. In the case of Carver, it slowly decreases from a median of almost 1 core hour to ~ 6 core minutes (and a last upper quartile of 1 core hour).

In summary, Hopper jobs (shorter jobs, with a higher degree of parallelism, bigger than Carver's) seem to be showing an increase in their wall clock time. As the effective job's core hours remain the same, they must be using fewer cores. Carver jobs (longer jobs, lower degree of parallelism, fewer core hours than Hopper's) have decreasing wall clock time and use more cores, but the increase is not sufficient to keep the job's core hours steady over the years.

Figure 5: Jobs’ wall clock time accuracy evolution for each workload year. In all systems wall clock time remains low.

4.2 Job wait time

According to Figure 4, for Hopper, the median of the wait time is steadily increasing from under 100 seconds to over 20 minutes (a pattern also present in the upper and lower quartiles). On Carver, the effective wait time increases over the four years from ~ 10 minutes in 2010 to ~ 20 minutes in 2014. However we notice a zigzag pattern trend in between. In 2011, Carver presented significantly shorter wait times, which could be attributed to a known increase of resources in the system. The steady increase of wait time over the lifetime fits with the growth of the user community of the systems.

4.3 Wall clock time accuracy

The wall clock accuracy is calculated as *real/estimated* wall clock time. The results are shown in Figure 5. Hopper does not show a clear trend: 2011 to 2013 presents a higher accuracy than 2010 and 2014, with a median variation between 0.2 and 0.4. For Carver, the median decreases over time, with significant changes between 2010 (~ 0.25) and 2011 (< 0.1). In 2014, the median is under 0.1 and the last quartile it is under 0.2. For Carver, there is a clear pattern of worse estimations as the time proceeds. In general, both systems present very low values with medians under 0.4.

Wall clock accuracy does not show a noticeable pattern beyond the fact that accuracy is low. On Hopper, 50% of all jobs run less than 40% of the estimated time. Similarly on Carver, 50% of all jobs less than 20% of the estimated time. These values indicate that the decisions made by the back-filling algorithms are based on inaccurate user estimations.

5. SUMMARY

Our work describes the evolution of jobs over the lifetime of two systems: Hopper and Carver. The results show that job wait times increased over time on both systems. Similarly, job size estimation accuracy was low on both systems. However, differences were found on the job geometry evolution. In the beginning of the time period analyzed, Carver’s job run times were significantly higher than Hopper’s. However, run time decreased on Carver and increased on Hopper. Thus the workloads became more similar towards the end of the period analyzed. Hopper jobs allocate significantly more

cores than Carver in all years, but the difference is slightly reduced over time. Core hours consumed by Hopper jobs did not change, but Carver’s decreased over time.

In summary, this work shows that the systems were used differently at different points of their lifetimes. This can be attributed to the evolution of the user behavior as users adapt applications towards the most optimal configuration. Another possibility is that the results reflect user adaptation to policy changes over the lifetime. More detailed analyses and reasoning of these trends are topics for future work.

6. ACKNOWLEDGMENTS

Financial support has been provided in part by the Swedish Government’s strategic effort eSSENCE, the European Union’s Seventh Framework Programme under grant agreement 610711 (CACTOS), and the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR) and the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

7. REFERENCES

- [1] K. Antypas, B. A. Austin, T. L. Butler, and R. A. Gerber. NERSC workload analysis on Hopper. Technical report, LBNL Report: 6804E, October 2014.
- [2] M. A. Bauer, A. Biem, S. McIntyre, N. Tamura, and Y. Xie. High-performance parallel and stream processing of x-ray microdiffraction data on multicores. In *Journal of Physics: Conference Series*, volume 341, page 012025. IOP Publishing, 2012.
- [3] S. Di, D. Kondo, and W. Cirne. Characterization and comparison of cloud versus grid workloads. In *2012 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 230–238. IEEE, 2012.
- [4] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn. Parallel job scheduling, a status report. In *Job Scheduling Strategies for Parallel Processing*, pages 1–16. Springer, 2005.
- [5] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE’08*, pages 1–10. Ieee, 2008.
- [6] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema. The grid workloads archive. *Future Generation Computer Systems*, 24(7):672–686, 2008.
- [7] D. A. Lifka. The ANL/IBM SP scheduling system. In *Job Scheduling Strategies for Parallel Processing*, pages 295–303. Springer, 1995.
- [8] S. N. Srirama, P. Jakovits, and E. Vainikko. Adapting scientific computing problems to clouds using MapReduce. *Future Generation Computer Systems*, 28(1):184–192, 2012.
- [9] G. Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 8. ACM, 2006.
- [10] W. W.-S. Wei. *Time series analysis*. Addison-Wesley publ, 1994.