Cluster Workload Analytics Revisited

Subrata Mitra, Suhas Javagal, Todd Gamblin, Adam Moody, Stephen Harrell, Saurabh Bagchi

Cluster Workload Analytics Revisited

Subrata Mitra*, Suhas Javagal*
{mitra4, sjavagal}@purdue.edu
Purdue University, IN, USA
* Equal contributors

Todd Gamblin, Adam Moody
Lawrence Livermore National Laboratory

Stephen Harrell, Saurabh Bagchi
{sharrell, sbagchi}@purdue.edu
Purdue University, IN, USA

Abstract—**Community compute clusters are now common in big organizations such as in universities and industries. These are excellent targets for analyzing resource usage and failure events because of the diverse user base and corresponding diversity of workloads. We introduce an public repository that hosts such rich dataset and would be useful for systems and dependability research communities. In this paper, we discuss our goals, provide details of the dataset, and present some statistical analysis.**

I. Introduction

Large high performance computing (HPC) clusters have become common in academic institutes, industries, and government labs. These clusters cater to the need for compute-intensive scientific applications and various kinds of big-data analytics. The large scale applications that run on these clusters, tackle problems that are infeasible to run on personal computers. However, managing such large and shared resources can be challenging due to the following four reasons.

Diversity of users: Often the user-base of these clusters are quite diverse both in terms of the nature of jobs they run, as well as the level of experience the users have in dealing with large, shared community cluster.

Diversity of execution environments: These diverse set of applications often use special libraries (e.g., particular versions of MPI), run on specialized hardware(e.g., Nvidia GPUs, Intel MICs) or show different resource usage patterns (e.g., many small jobs, strong or weak scaling).

Privacy and security: Cluster managers often have very limited visibility inside users' development and execution environment. Users usually write their own job submission scripts that wrap various applications. The scripts allow customization of the execution in a wide variety of ways. Due to the complexity of some scripts and the lack of any standard, it is often difficult to parse/identify what are the core applications that are being invoked from a script and how it is run.

Failures: Failures are not infrequent in large clusters. These can be either transient (in the OS, hardware or network) or permanent failures (e.g., hard-drive crash, memory failures). The diversity of applications and its user-base also increases the chance of workload-related soft (or performance) failures. For example, if one application creates network congestion that might lead to slowdown of other applications.

We have initiated efforts to build an open data repository [1] of system usage and failure information from community cluster at Purdue. Our final goal is to develop tools and techniques that would improve cluster management through data-driven divisions. We also hope our workload data repository would enable dependability researchers to perform analyses on real-world, large-scale data. There have been previous efforts to create public failure repositories, the most notable of which is the Computer Failure Data Repository (CFDR) [2] which has 13 datasets from different clusters from HPC and Internet services domain. However, there are few issues with this dataset. Firstly, it is not up-to-date and hence offer very limited insights on today's workload patterns. Secondly, the dataset is focused on node-level hardware or OS failures. Thus, it does not capture failure situations that might occur due to interplay between different workloads. Finally, in addition to failures, cluster administrators are also interested in job performance and optimal usage of the cluster resources and these are not covered by the CFDR dataset. To address these issues, we have created a new *publicly available workload dataset* [1]. It contains workload traces for $489,971$ jobs from *Conte* at Purdue, one of the largest university-wide community clusters with a diverse user base and diverse workloads, as presented in Table I. We also made sure that the data is fully anonymized. In addition, we plan to periodically update our repository by adding latest dataset from *Conte* and also from other university and government lab clusters.

Details	University community cluster
Duration	Oct 2014 – Mar 2015
Total number of jobs	489,971
Number of unique users	306
No. of unique application behaviors	3,373

TABLE I: Summary of workload data in the open repository

II. Description of our community cluster, *Conte*

Node hardware, network and filesystem: *Conte* has 580 nodes each with two 16 core Intel Xeon E5-2670 processors, two Xeon Phi accelerator cards and 64GB of memory. Nodes are connected through a 40GB/s Infiniband (IB) network. These filesystems are used university wide and are accessed through the IP over IB egress. The scratch filesystem used by the jobs is a Lustre 2.4 installation that can sustain upto 23GB/s and is connected via the above-mentioned IB network. The total capacity is 1.4PB with current utilization being 49%. Every node has a local filesystem but only usable by the OS.

Software and Scheduling: Each node runs RHEL 6.6 OS. The nodes are administrated using the Kickstart installers and Puppet configuration management software. Along with default RHEL libraries, the environment also provides many other important pre-installed libraries and applications which can be used through command: `module load <name>`. The scheduling is done by TORQUE 4, an open source implementation of Portable Batch System (PBS) and with Moab as the resources manager. At submission time, each job requests a time duration of execution (a wall clock time), number of nodes and, optionally, amount of memory needed using PBS submission scripts. A job is killed when it exceeds the specified time limit or by an out-of-memory (OOM) killer(a kernel level memory manager). Job scheduling uses a community cluster allocation method in which, some research groups purchase nodes and get *semi-dedicated* access to the nodes that they purchase through their own queue and can also access a far greater number of nodes from the general pool through a shared queue (called standby queue), on demand and opportunistically. For the purchased nodes, research groups get a service level agreement (SLA) of maximum 4 hours wait time for the job at the head of their group-specific queue. When those nodes are not in use, jobs from the standby queue are scheduled on those nodes with a maximum walltime limit of 4 hours. By default, a single job is scheduled on an entire node but sharing can be enabled using a configuration parameter in the job submission scripts.

III. Data Sources

Our dataset has five major components. For privacy reasons, only the first two of these five data sources could be made available in our public repository.

Accounting logs: The accounting logs extracted from TORQUE provides job scheduling related details such as the job id, queue, start and end timestamps, user id, requested resources, resources consumed, the nodes on which job was actually run and an exit status.

TACC Stats: TACC stats [3] data provides more fine-grained resource usage profile on all the nodes used by the job. For each node, TACC Stats data has periodic snapshots of various system metrics which include: usage metrics for the local disk and the Lustre filesystem, Infiniband and IP network traffic, and process and memory statistics.

Syslog: The Syslog data comprises of kernel and system messages of all nodes. It contains memory errors, OOM killer messages, filesystem status, etc. all in free form text.

Library lists: The library list data captures the periodic snapshots of the shared libraries accessed by the jobs in each computing node. Consequently, many such snapshots will be created corresponding to a long running job. Also a job using multiple nodes will have files for libraries loaded in each node.

Job scripts: Job script data contains the user's job submission scripts written with PBS directives to specify the resource requirements and the layout of the tasks a job would perform.

IV. Data Analysis

The existing dataset in our repository [1] can facilitate various data-driven analysis such as to find hidden trends in job failures, resource wastage or degraded performance, thus enabling better management of community clusters. As a representative example we analyze *if certain libraries can be implicated in jobs failures*. The actual reason for a job failure can be anything from node level hardware problems, bugs in the software stack to problems in the network fabric. Therefore, a comprehensive analysis of job failure patterns would help to understand the root-causes.

Methodology: The exit code of a job is useful to find the cause of the failure. Usually these exit codes are set by the job scheduling system or by the kernel but users can also set their own exit status. However, after analyzing job scripts and discussing with our cluster administrators, we identified the exit code convention as: (1) 0 denotes a successful run, (2) negative error codes usually indicate a failure of the scheduler or the nodes, (3) in the absence of user specified code, exit code from the last executed command in the job script is reported. On analyzing user-defined exit codes inside the job scripts, we found that the most common user-defined exit code is 1(26%). 5 users specified negative exit codes and 44 users explicitly returned 0 to mark success. Exit code $\geq$ 128 or $\geq$ 256 can be decomposed as `128 (or 256) + a system signal` where the system signal can be SIGTERM/SIGKILL (memory exhaustion), SIGSEGV (segment violation), SIGBUS (file system error), SIGILL/SIGFPE (bad operation), etc. indicating the root case of an unsuccessful job termination. However, in TORQUE, exit code 1 indicates a generic error and cannot be assigned to any particular category.

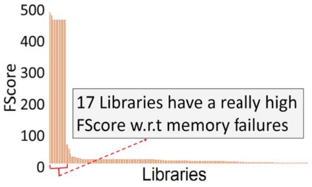

Fig. 1: Distribution of *FScore* values w.r.t failures due to memory problems in *Conte*.

Reason	% of *failed* jobs
Time expired (timeout)	20.3
Memory exhaustion	15.2
Segmentation Fault	9.3
Quit/keyboard interrupt	5.3
File system/path problem	3.7
Self abort/assert failure	0.6

Fig. 2: Few major types of job failures showing percentage of *failed jobs*. We could not classify 40% of the failed jobs and about 6% of jobs had generic exit code (-1).

Classification of job failures with the help of exit codes: The analysis of the job's exit codes revealed in *Conte* 16.2% of the jobs had failed (exit code not equal to 0). Based on exit code and runtime of the job, we have summarized the jobs with various problem manifestations in Figure 2.

Which library to blame? While executing a job uses a variety of libraries: libraries from the environment, written by the user or third-party. It is worthwhile to consider if a failure is caused by a particular library or due to interaction among a set of libraries. We now illustrate how the source of a certain kind of failure can be narrowed down to few libraries using statistical analysis. We introduce the *FScore* scoring metrics as:

$$\mathit{FScore} = \frac{P_r(L|F)}{P_r(L|F)+P_r(L|job\ succeeded)} \times Freq_{L|F}$$

Where for library L and failure type F, $P_r(L|F)$ and $P_r(L|job\ succeeded)$ denotes the probability that a job using library L failed with type F and succeeded respectively, and $Freq_{L|F}$ is the number of jobs with failure F that also uses library L. It should be noted that $P_r(L|F)$ and $P_r(L|job\ succeeded)$ are not complementary to each other because library L might be also be associated with *other* type of failures. Suspicious libraries can be identified for detailed investigation by choosing the top K, ordered by the *FScore*. Fig. 1 shows the FScore distribution for jobs failed due to memory problems in *Conte*. The names of the libraries are not shared due to security reasons.

Threats to validity: Using syslog messages, we found that 92% of the jobs that failed with exit codes representing memory exhaustion, was also preceded by syslog with OOM related messages and 77% of the jobs that output the memory exhaustion message, ultimately exited with exit codes for memory exhaustion. But a drawback of the exit code based or scheduler dependent failure analysis is that more fine-grained failure classification cannot be achieved. A thorough analysis of syslog might unravel more nuanced causes of failure.

V. Conclusion

We introduce a public data repository that currently holds a variety of resource usage and application level metrics from a large number of jobs submitted to a big community cluster at Purdue. Based on this data set, we present some statistical analysis that help diagnosing the root cause of job failures. We strongly believe the research community would benefit from this open workload data repository and wish it would enable more insightful analyses in future.

References

[1] "FRESCO - The Open Data Repository for Workloads and Failures in Large-scale Computing Clusters," https://diagrid.org/resources/247.

[2] "Computer Failure Data Repository: http://usenix.org/cfdr."

[3] T. Evans, W. L. Barth, J. C. Browne, R. L. DeLeon, T. R. Furlani, S. M. Gallo, M. D. Jones, and A. K. Patra, "Comprehensive resource use monitoring for hpc systems with tacc stats," in *HPC User Support Tools (HUST)*, 2014.