Computer Cluster Workload Analysis

I. Grudenic, I. Bakarcic and N. Bogunovic
Department of Electronics, Microelectronics, Computer and Intelligent Systems
Faculty of electrical engineering and computing, University of Zagreb
Unska 3, Zagreb, Croatia
Phone: 01-6129-999 int. 548 Fax: 01-6129-653 E-mail: igor.grudenic@fer.hr

Abstract - Performance of computer clusters is greatly affected by a nature of the submitted workload. Early characterization of different workload types allows for scheduler fine tuning as well as predictions on the system load. Statistical analysis and visual representation of the workload data provide valuable insight to the overall system utilization and may reveal potential bottlenecks and points for improvement. In this paper we describe important cluster job features and introduce a tool for statistical analysis and manipulation of the workload data that is a part of a cluster simulation and runtime prediction system.

I. INTRODUCTION

High Performance Computing (HPC) is an evolving trend in computing industry with predicted growth of 10% annually. System sizes are also being increased with petaflop systems becoming available. Scheduling in HPC systems is a vital part of harvesting available processing power. Scheduling for large computer systems is inherently a hard problem that is made even harder in HPC environment because of high task rates and potentially unstable environment.

The main reasons for the constantly changing environment are high job rates on the input of the HPC system, unpredictability of the user behavior that translates to the variation in workload, and potential system changes due to hardware malfunctions, upgrades or system updates. Computer clusters are the most dominant HPC architecture constituting 80% of the top 500 supercomputers [1].

In computer cluster jobs are scheduled according to the user predictions on the job runtime and the requested resources. User runtime predictions are usually not very accurate and they often serve as a deadline for the given job. Detailed workload analysis and data processing can be used to improve on user predictions and make scheduling more efficient.

In this paper we analyze a set of cluster workloads in order to get an insight into different workload characteristics. This information can be used to modify and tune up cluster schedulers in order to improve user computing experience. We also present a tool for statistical workload analysis and data preparation that is part of our runtime prediction and simulation system for computer clusters.

In section II we present architecture of our *Workload Analyzer tool* as well as the role of the tool in the cluster simulation and automated behavior prediction. Detailed analysis of 12 workload data files with focus on the job runtime and user runtime predictions is given in section III. Additionally we analyzed which results of the analysis can be used in order to improve computer cluster predictability together with the user visible performance of the cluster scheduling software.

II. WORKLOAD ANALYZER TOOL

Workload analyzer tool is a part of the simulation and cluster behavior prediction system. It is designed to perform descriptive statistical analysis of the workload data and to preprocess and reorganize data for data mining and cluster simulation. Fig. 1 presents both architecture and the role of the workload analyzer tool.

Workload analyzer is a starter application in the exploration of a cluster behavior and is composed of *Log file parser*, *Data processor*, *Chart builder* and *GUI*.

Exploration is triggered by the user instructing *Data processor* to load and parse log file (1). *Data processor* activates *Log file parser* (2) to load workload data (3). Parsed data are forwarded to the *Data processor* (4) to perform data filtering and other transformations defined by the user. *Chart Builder* uses processed data (5) and user defined chart characteristics (7) to create different data views (6).

Processed data can later be saved to the new log file (2,8) to be used for simulation (10) or data mining (9,11). Log file parser is used for loading workload data in the Workload Analyzer and the Cluster Simulator.

Workload analyzer is designed as a multithreaded

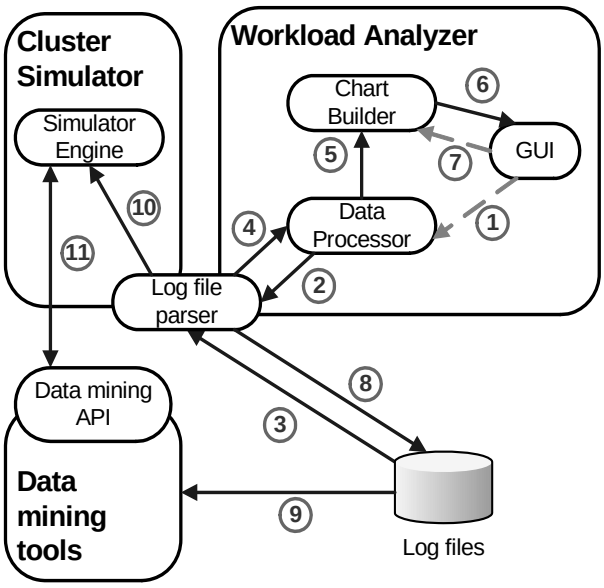

Fig. 1. Workload Analyzer architecture

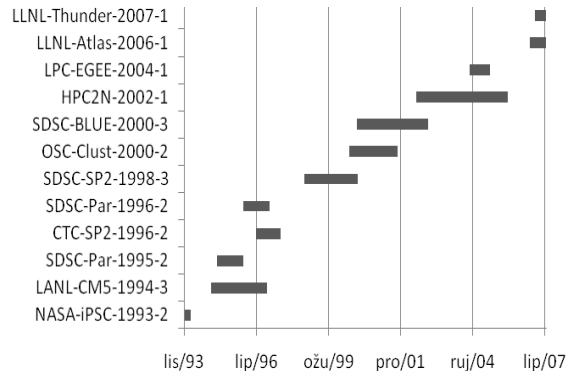

Fig. 2. Cluster workload timeline

application because data parsing and processing are CPU intensive due to the size of the log files. In order to improve user experience *GUI* component runs in one thread, while *Log File Parser* and *Data Processor* start a separate thread for each parsing/processing job. Multiple parsing and processing jobs running in parallel are allowed and supported by the user interface.

Results of the *Workload Analyzer* tool are used as an input parameter for the set of the *Data mining tools* that are used to predict behavior of the cluster system given the cluster usage history.

Outputs of the workload analyzer are also used as an input to the *Cluster Simulator* [2] that generates job sequences according to the existing preprocessed workload traces and evaluates different scheduling strategies. Cluster simulator uses *Data mining* in order to predict future cluster behavior, and then uses obtained predictions in the simulation of the scheduling process.

III. WORKLOAD DATA ANALYSIS

Workload data analyzed in this paper originate from the parallel workload archive [3]. Data is collected from 12 computer clusters over the last 10 years period and are presented in Fig.2. Logs cover different time frames starting from few months up to three years.

Number of attributes [4] is covered for each job in the log. Attributes can be divided into two groups – user provided attributes and system assigned attributes. User provided attributes include submit time, job size, user runtime prediction, application name, system queue and partition while system assigned attributes involve wait time, total runtime, job status and number of allocated resources. Most of the system assigned attributes are scheduler dependent meaning that changing a scheduler policy would produce a different log file.

Some of the attributes like identification of applications used and user runtime predictions are missing in some of the log files. These log files cannot be efficiently used for automated runtime prediction, and are excluded from some of the analysis in this paper.

A. Job Runtimes

Runtime is a most important property needed for efficient cluster scheduling.

Different distributions such as hyper-exponential [5], hyper-erlang [6], log-uniform [7] and hyper-gamma are used to describe workload sets. Ranges of job sizes are grouped together and modeled separately in [6], while others provide universal model for the entire job set. It is concluded [8] that load generators based on the mentioned distributions are more stable than real workloads because generated job sequences reach steady average runtime earlier than the corresponding real workloads.

Average runtime of the job has no correlation to cluster size, computing power or the number of users. It largely depends on the variability of applications of the given cluster which typically results in large standard errors for runtime distributions.

Although the average runtime gives no information on the typical cluster usage, it can be shown that a small number of distinct applications utilize rather large portions of available cluster time for all the analyzed clusters (Fig.3.). In majority of analyzed computer clusters 70% of total runtime is used by only 4% of applications, while 90% of the runtime is utilized by no more than 20% of applications.

Since most of the time consuming applications are used by a small amount of users with average number of applications per user ranging [1, 1.29], it is obvious that very limited group of users exploit most of the cpu time (Fig.3). It is noted that 70% of the total runtime is used by 7.5% of the users on the average, while 90% of the runtime is utilized by average 19.2% of the users.

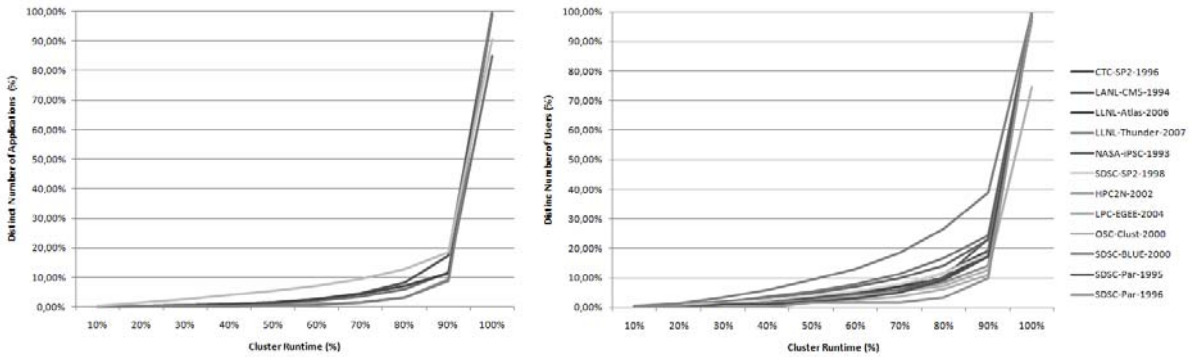

Fig. 3. Distribution of runtime over applications and users

This information is usually used by cluster schedulers in order to provide fair sharing of computer cluster, meaning that users that account for unfairly high amount of clusters cpu time are left to wait longer for their applications to execute. Some sort of anti-starvation mechanism is usually implemented in cluster schedulers in order to force such postponed applications to execute within the reasonable period of time thus preventing starvation.

When most time consuming applications are analyzed in detail, their average runtime is found to be higher than the system's runtime average. Average runtime for the applications consuming 70% of the total system runtime is 2.11 times higher than the overall system runtime average and this ratio drops to 1.67 for applications consuming up to 90% of the total runtime over all the analyzed computer cluster logs.

It may be intuitive that top time consuming applications on the computer cluster are highly parallel in nature but this seems not to be the case in any of the analyzed workloads. In some of the clusters like LLNL-Atlas-2006-1, average CPU allocation by the top 70% runtime consuming applications is 40% less than the global CPU allocation average. In all the other analyzed cluster logs top applications use up to 30% more than the average number of processors.

Increasing parallelism of applications does not inherently increase efficiency of the entire cluster system because parallelization of the application usually introduces overhead within the application, and additionally causes fragmentation of the CPU-time space that is hard to reduce by the cluster schedulers.

B. User Runtime Predictions

When users submit a job to the computer cluster they are often required to provide hard runtime estimate of the job duration. Jobs that exceed this estimate are usually automatically terminated. This forces users to overestimate job duration and causes cluster schedulers to underperform. Most of the users are not even aware of potential benefits of the tighter runtime estimations [9].

Fraction of the users doesn't even care to provide a runtime estimate and just uses the system defined default value. The chart on Fig.4. depicts a random sample from the *SDSC-SP2-1998-3* log file showing differences between user estimates and actual runtimes. It can be noted that there is no job with observable runtime exceeding user

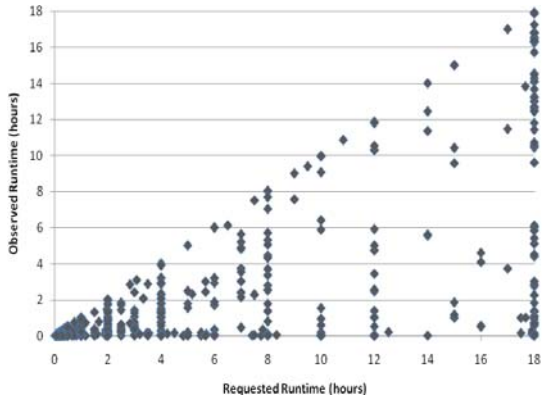

Fig. 4. Requested and observed job runtimes

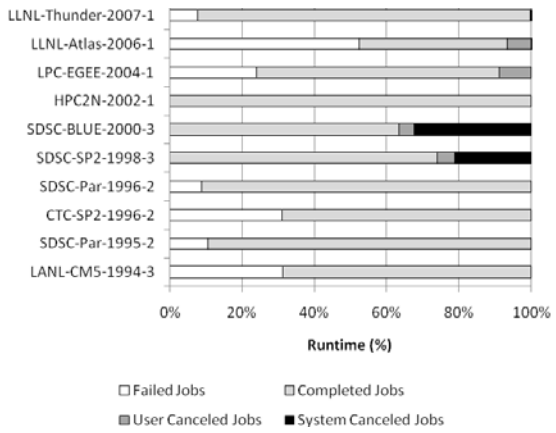

Fig. 5. Job statuses over runtime

runtime estimate due to scheduler automatic job termination. It can also be observed that the majority of jobs have short runtimes. Discrete time intervals for user runtime estimates are also noticeable as columns of values in the chart.

Every job that enters the cluster system is tagged with the status value that denotes the reason for the job termination. Status value can indicate failed jobs, successfully completed jobs, user canceled jobs and system canceled jobs. Failed jobs terminate because of the internal program error which can be caused by the bug in the software or intentional erroneous exit.

The analysis of job statuses in the standard workload archive is presented in Fig.5. where each of the potential job status is represented by the fraction of the entire system runtime. It is obvious that the majority of runtime is used by jobs that complete successfully. Four systems have very high runtime ratios of more than 24% of the total runtime spent on failed jobs, and the most probable reason for this is an exhaustion of the system resources since software errors are usually detected very early in the job execution. Exception to this is the software performing large multi stage calculations and failing in some of the late stages. Multistage calculations should be executed separately if possible in order to improve efficiency of the entire system.

User canceled jobs take minority of the total runtime with maximum of 8.7% for *LPC-EGEE-2004-1*. This is because most of the user canceled jobs are stopped due to improper parameter configuration. This often happens when users start an application that they haven't been using for a while.

The most interesting part of the status analysis is the runtime wasted on the system canceled jobs. These jobs are terminated because their runtime exceeded the provided user estimation. Two cluster systems that employ such a termination are *SDSC-SP2-1998-3* and *SDSC-BLUE-2000-3*, which spent more than 20% of the entire available time on the jobs terminated early by the scheduler. The results of early terminated jobs are often unusable.

In order to investigate the reason for two of the mentioned clusters that exhibit such a high job termination rate, a quality of user predictions are analyzed. Fig. 6. shows the difference in the quality of predictions for the *SDSC-SP2-1998-3* system with high job termination rates

and the *LLNL-Atlas-2006* which has a negligible runtime losses caused by runtime prediction.

Since overall quality of user predictions cannot be fully measured, because jobs that overrun estimated time are terminated and their real runtime is not known, only jobs that run shorter then requested runtime are analyzed. These jobs are then grouped based on fraction of the observed runtime in the user requested time.

It can be noted that *LLNL-Atlas-2006* users have more conservative predictions than *SDSC-SP2-1998-3* users. There are 61.3% of jobs at *LLNL-Atlas-2006* where observed runtime is less than 10% of the user predicted runtime. When this is compared to 38.5% of jobs at *SDSC-SP2-1998-3* with same prediction results it is obvious that users of *LLNL-Atlas-2006* are more likely to ask for more time then is really needed.

This also makes *SDSC-SP2-1998-3* users more accurate in their runtime predictions since 5% of their estimates are no more than 10% off the observed runtime, which more than doubles the fraction of predictions given by the *LLNL-Atlas-2006* users.

Although *SDSC-SP2-1998-3* users give more precise runtime estimates, the system hardly benefits from such a behavior due to the high number of job terminations caused by the inaccurate predictions. In this comparison conservative users do get a better service, but this is not easily expressible by the standard cluster quality metrics.

C. Potential benefits

In order to improve user experience with the computer clusters some of the simple metrics such as average wait time, application slowdown and predictability of the start time for applications must be improved. Some system properties like utilization are not directly visible by the user and should not be over optimized while disregarding system properties more visible to the users.

Running shorter jobs as early as possible and their compact packing in the CPU-time plane can improve both utilization of the system as well as shorten wait times. In order to improve packing of the jobs precise runtime must be known in advance or at least some degree of certainty must be employed. Since it is determined that majority of runtime is consumed by only several applications, these specific applications should be a primary target for automated target prediction.

Since some clusters experience virtual performance decrease because of the user runtime underestimations, special care should be taken to predict such a behavior and warn the users of the probable forced termination.

IV. CONCLUSION

In this paper we have analyzed data from 12 computer clusters in order to pinpoint some of the common properties that can later be used to tune up cluster schedulers. We have found that majority of computer clusters are being used by several users and applications.

Since user runtime prediction is an important parameter in cluster scheduler performance we analyzed the amount of resources that are wasted due to imperfect prediction.

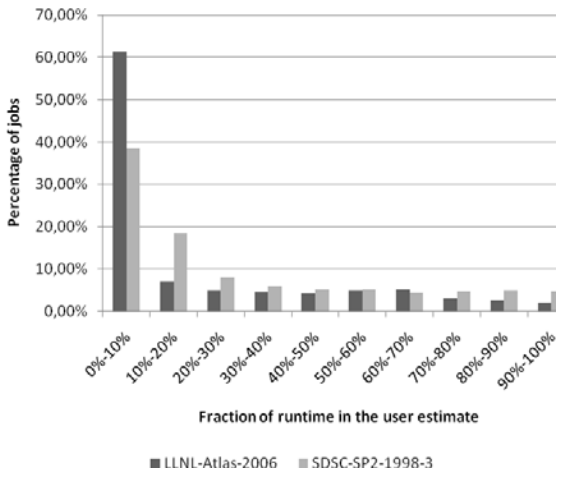

Fig. 6. Comparison of user runtime overestimation

Additionally, we presented architecture of the workload analyzer tool that is used to produce presented data analysis and is part of the runtime prediction and cluster simulation system which is under developement.

In the future we plan to simulate common scheduling techniques while trying to predict job runtimes as well as job arrivals in the system. We also plan to develop new scheduling techniques that will benefit from more precise runtime predictions and will be able to pack jobs more efficiently across the computer cluster.

IV. REFERENCES

[1] Top500 Supercomputer Sites, http://www.top500.org/

[2] I. Grudenic and N. Bogunovic, “Computer Cluster and Grid Simulator”, *Proceedings of the Joint Conferences Computers in Technical systems and Intelligent systems*, p. 49, 2009.

[3] Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload/

[4] S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby, “Benchmarks and Standards for the Evaluation of Parallel Job Schedulers”, *Lecture Notes in Compuer. Science*, vol. 1659, p. 66, 1999.

[5] D. G. Feitelson, “Packing Schemes for Gang Scheduling”, *Lecture Notes in Computer Science*, vol. 1162, p. 89, 1996.

[6] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira and J. Riordan, “Modeling of workload in MPPs”, *Lecture Notes in Computer Science*, vol. 1291, p. 95, 1997.

[7] A.B. Downey, “A parallel workload model and its implications”, *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing*, p. 112, 1997.

[8] D. G. Feitelson, “Metrics for parallel job scheduling and their convergence”, *Lecture Notes in Computer Science*, vol. 2221, p. 188, 2001.

[9] C.B. Lee, Y. Schwartzman, J. Hardy and A. Snavely, "Are user runtime estimates inherently inaccurate?", *Lecture Notes in Computer Science*, vol. 3277, p. 253, 2005.