Forecasting Storms in Parallel File Systems *

Ryan McKenna (Student)
University of Delaware
mckennar@udel.edu

Todd Gamblin (Advisor)
Lawrence Livermore National Laboratory
tgamblin@llnl.gov

Adam Moody (Advisor)
Lawrence Livermore National Laboratory
moody20@llnl.gov

Bronis de Supinski (Advisor)
Lawrence Livermore National Laboratory
bronis@llnl.gov

Michela Taufer (Advisor)
University of Delaware
taufer@udel.edu

ABSTRACT

Large-scale scientific applications rely on the parallel file system (PFS) to store checkpoints and outputs. When the PFS is over-utilized, applications can slow down significantly as they compete for scarce bandwidth. To prevent this type of "filesystem storm", schedulers must avoid running many IO-intensive jobs at the same time. To effectively implement such a strategy, schedulers must predict the IO workload and runtime of future jobs. In this poster, we explore the use of machine learning methods to forecast file system usage and to predict the runtimes of queued jobs using historical data. We show that our runtime predictions achieve over 80% accuracy to within 10 minutes of actual runtime.

Keywords

Machine Learning, High Performance Computing, Parallel File System

1. MOTIVATION

Most programs that run on large-scale parallel clusters interact with the parallel file system by writing large *checkpoints*, which save program state for recovery after a failure[1]. The parallel file system is a shared resource, and multiple applications read and write to it concurrently. File system performance can vary depending on how much data is being written or read, *or* by how many processes access it concurrently.

The goal of this research is to mitigate resource contention by identifying periods of time when the file system may be slow, and to identify IO-intensive jobs before they start. The contributions of this work focus on the assessment of different Machine Learning techniques to accurately and efficiently predict per-job runtime statistics such as runtime and bytes read/written to the file system. The results provide us with missing insights on effective prediction techniques for large HPC systems [2].

*This work performed under the auspices of the U.S. Department of Energy and an appointment to the Office of Science, Science Undergraduate Laboratory Internship (SULI) Program at the Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. IM #: LLNL-ABS-675739

2. METHODOLOGY

Historical Data. We have collected 3 months of historical job data from a production Linux cluster at Lawrence Livermore National Laboratory. Our data set includes Lustre filesystem counters recorded at the start and end of each jobs, as well as the submission scripts used to run the jobs. These job scripts can tell us the name of the application, its inputs, and the Lustre counters tell us about its IO behavior.

Regression Models. A regressor is a type of machine learning model that is specialized to predict continuous output. We use decision trees, random forests, and k-nearest neighbors methods in this work, because they are widely available, they can take non-continuous inputs, and they can model nonlinear behavior.

Predicting runtime with Job Scripts. Job scripts provide a wealth of data for understanding the behavior of simulation jobs. For example, they may contain the name of the application, number of nodes requested, an estimate of runtime provided by the user, the arguments to the job command, references to input files, and many more attributes. Often, jobs with similar inputs have similar runtimes. Our challenge is to extract and frame these attributes so that they can be used to train a model. Our poster briefly describes our job script parsing techniques, as well as the job features most useful for runtime.

Predicting Workload from Lustre Counters. In addition to job script data, we monitor Lustre counter data from each job. Because it is purely numerical, we can easily build a regressor from this data. Armed with both job runtime predictions and Lustre workload predictions, we can simulate

Error	User Requested	Decision Tree	Random Forest	K Neighbors
< 10 Mins	14.7%	80.0%	76.1%	78.9%
< 20 Mins	16.9%	82.0%	78.9%	81.5%
< 30 Mins	25.0%	83.2%	80.5%	83.1%
< 40 Mins	27.1%	84.1%	81.6%	84.2%
< 50 Mins	76.4%	85.6%	83.1%	85.5%
< 60 Mins	76.8%	86.2%	83.9%	86.4%

Table 1: Minimal interval error for runtime predictions when using ML algorithms and users' estimates.

Error	Bytes Written	Bytes Read
< 1 MB	74.9%	63.7%
< 16 MB	79.8%	68.6%
< 256 MB	86.6%	82.7%
< 4 GB	92.0%	90.0%
< 64 GB	97.4%	97.0%

Table 2: File system fingerprint predictions.

the job queue to determine which IO intensive jobs run at the same time.

3. RESULTS

We assess the effectiveness of the three ML models for the dataset of interest. Overall, we observe how the decision tree consistently gives the highest accuracy for predicting runtimes, improving the percentage of accurate predictions by over 65%. Table 1 summarizes the percentage of prediction successes measured for different error intervals ranging from 10 minutes to one hour when users define the runtime (User Requested) and when using the three ML models. The table outlines how the machine learning algorithms, and the decision tree in particular, substantially improve the number of accurate run time predictions as compared to the amount of time the user requested.

Figure 1 shows the prediction accuracy for different windows sizes when using the three ML models for the three error intervals (i.e., within 10, 20 and 30 minutes). Specifically, the figure indicates to us that, when starting the prediction with a window of historical data of one day, the prediction accuracy steadily improves and eventually levels off as the window size increases. We can claim that good predictions can be reached as long as the window size of historical data is at least two days in length, and a window size of five days is only 0.2% worse than the theoretical maximum achieved by using all available history. Table 2 summarizes the percentage of read and write bytes accurately predicted for a given error ranging from 1 MB to 64 GB and confirms our claim that the decision tree can provide accurate IO predictions for a large portion of the considered applications. The extensive set of results from our study is presented in our poster.

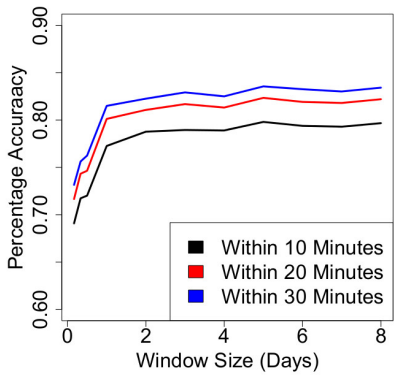

Figure 1: Prediction accuracy for different prediction window sizes when using the decision tree regressor.

4. CONCLUSION

Results of this work show that ML models have promising potential, and can serve as a viable solution for predicting IO utilization and, ultimately, more efficiently allocating jobs to minimize file system resource contention. For future work, we will analyze more fine grained data - that way, we will not have to rely on the assumption that the bytes read/written is uniform over the course of the job, as we did in this study.

5. REFERENCES

[1] A. Moody, G. Bronevetsky, K. Mohror, B. de Supinski. Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System. In *Proc. of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2010.

[2] K.E. Isaacs, T. Gamblin, A. Bhatele, M. Schulz, B. Hamann, and P-Timo Bremer. Ordering Traces Logically to Identify Lateness in Message Passing Programs. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2015.