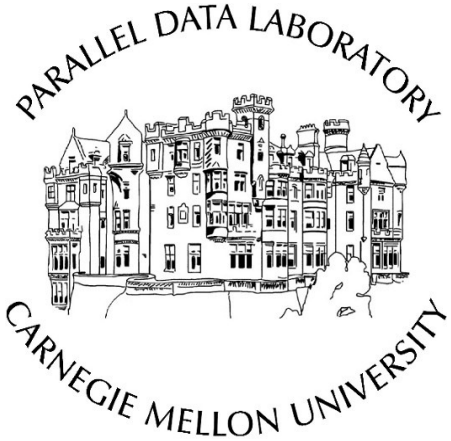


This is why ML-driven cluster scheduling remains widely impractical

Michael Kuchnik[1†], Jun Woo Park[1†], Chuck Cranor[†], Elisabeth Moore[‡], Nathan DeBardeleben[‡], and George Amvrosiadis[†]

[†]Carnegie Mellon University
[‡]Los Alamos National Laboratory

CMU-PDL-19-103
May 2019

Parallel Data Laboratory
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Using learning algorithms in cluster schedulers has the potential to increase the effectiveness of a computing cluster, yet deployments are limited. Our analysis of a diverse set of workload traces from national labs and industry points to three challenges that have received little attention in the literature. First, lack of data diversity can adversely affect the design of prediction systems. We demonstrate how we detected limitations, such as impractical feature engineering, unreliable prediction performance, and inconspicuous overfitting in prior work. Second, workload changes can negatively affect predictor performance. For the first time, we quantify how accuracy degrades over time due to the non-stationarity that is common across most traces. We further examine the effectiveness of adaptive techniques, such as online training, at addressing this problem. Third, aiming for high prediction accuracy alone does not guarantee dramatically better end-to-end performance. Our experimental results indicate that the relationship between prediction error and end-to-end performance can be small for common cluster deployments.

[1]Contributed Equally.

Acknowledgements: We thank the members and companies of the PDL Consortium (Alibaba Group, Amazon, Datrium, Dell EMC, Facebook, Google, Hewlett Packard Enterprise, Hitachi, IBM Research, Intel, Micron, Microsoft Research, NetApp, Oracle, Salesforce, Samsung, Seagate, Two Sigma, Veritas and Western Digital) for their interest, insights, feedback, and support. This work is supported in part by a National Defense Science and Engineering Graduate Fellowship.

Keywords: Machine Learning, Cluster Scheduling, Cloud Computing, High Performance Computing

1 Introduction

Machine learning (ML) has the potential to change the way we build efficient datacenters [6, 20, 18, 22]. Successful ML deployments exist for resource allocation [4], failure prediction [35], and data center cooling [7, 15, 16]. Despite active research, such successes are not present in the area of cluster job scheduling. In this paper, we identify three key reasons why this gap has formed between research and practice in the context of job duration prediction using ML models.

Our first concern is *data availability and diversity* (Section 2). Most publicly available traces have fewer than 30 features and span only a few months at most [12]. While systems like Microsoft's Resource Central [4] leverage over 500 features [5] describing virtual machine runtimes, most researchers are not privy to such a wealth of data. Other systems, such as Jockey [14], utilize job structure information, such as dependencies between jobs, to inform scheduling decisions. However, commonly available traces lack such information, and predictors must resort to utilizing what is known to the scheduler at job submission time. *We show that only a few features are crucial when predicting a job's resource allocation, but these features are not the ones commonly available.* For example, removing jobs submitted by fewer than 4% of users can *halve* model performance on job runtime prediction, which agrees with typical Zipf distributional assumptions [10]. We find that broadly available features such as job size and submission time are poorly predictive compared to features that can be used to identify users or applications.

Our second concern is *workload variability* and its effects on model performance (Section 3). A slew of prior work on job scheduling has focused on building models that predict job characteristics such as duration, resource usage, and outcome [2, 30, 28, 25, 19, 17, 31, 21, 33, 3]. But because of the aforementioned lack of data, most models are only evaluated on short traces that do not expose variations in model performance. We use longer workload traces (up to 5 years) that contain natural variability to demonstrate its effects on model performance. For example, we show that performance can fluctuate by more than 40x on a day-to-day basis. This leads us to conclude that *longer traces are necessary in model evaluation, as well as metrics that capture variation in model performance over time.* We also show that even though the data is non-stationary, predictive performance increases as more training data is added.

Our third concern is understanding the *effect of accuracy on end-to-end performance*, that is effect of predictions on scheduler-level metrics (Section 4). The hope of using machine learning in a scheduling system is that it yields lower latencies or higher goodput (which measures useful machine utilization). However, as we described earlier, model performance is highly variable over time, and there are many cases, such as when a new users enters the system, when predictions are likely to be wrong. Because of this, there is a need to understand when model predictions are "good enough", and when prediction error actually impacts system performance. We attempt to understand this at a fundamental level by studying the effect of prediction accuracy to end-to-end performance. *A key result of our study is that both timing and magnitude of prediction errors are needed to characterize end-to-end performance.*

Data and prediction model used. In this paper we focus on predictors that use job descriptions to generate predictions at job submission time. Our data set consists of three traces with workloads from national labs and industry. We use traces captured at a Google cluster [34, 27], the Los Alamos National Laboratory (LANL) Mustang cluster [1], and the Pittsburgh Supercomputing Center (PSC) Bridges cluster[1]. These traces bring with them a fresh perspective: Mustang is the longest trace ever released (5 years), and the *newly released* PSC trace spans 4000 users, pushing the envelope on the number of unique users in a trace. Our analysis utilizes the recent 3Sigma cluster scheduler [26], which uses a collection of independent predictors to generate predictions for jobs which appear to have been repeated. Each predictor is conditioned on a feature, such as user ID, or a combination of features, such as both user ID and tasks requested. The predictors are evaluated on prior data to find ones that have historically performed well, and these are used

[1]The LANL Mustang and PSC Bridges traces will be available online through the ATLAS repository at http://www.project-atlas.org.

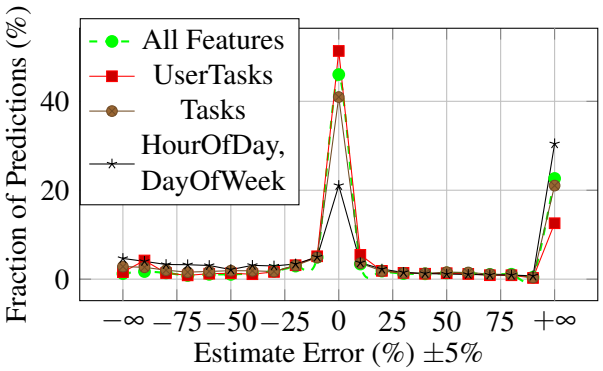


Figure 1: Features affect model performance unequally. The fraction of accurate job duration predictions, concentrated in the middle of the curve, is significantly lower for the LANL Mustang trace when user information is removed.

for the final prediction.

2 Lack of diversity

The first step towards practical ML-driven schedulers is to ensure that the input features are easily available. In this section we test the efficacy of such features for achieving high prediction accuracy, and attempt to better understand the role each plays in the final model performance.

After examining tens of job traces [12, 34, 27, 1] we find that several features are commonly available across them: user and group name of the submitter; requested and allocated number of processors; job outcome; submission, start, and end time of the job. Of those, only the user and group ID, the requested number of processors, and the job submission time are available at the time a job is submitted to the scheduler. This is also the time when predictions are most valuable. These features and their engineered variants, such as the hour of the day and day of the week when the job was submitted, are the features we investigate first.

In addition to these features that are common across the board, two other features appear frequently. First, several traces include a *job name* corresponding to the name given to the scheduler upon job submission. Unfortunately, systems such as MapReduce auto-generate unique names for each job, making it a useless feature for predictions. To address this issue, a variant of job name, the *logical job name*, may be used instead of the *job name* if it's available. A *logical job name* is used to identify an application by using a hash of the executable name and parameters [34, 27], which avoids the uniqueness problem mentioned previously. Second, several traces include user-defined runtime limits. We refrain from using them in our models, since they are not always available to bound predictions. Regardless, we have experimented with using them and did not witness a significant increase in performance.

Observation 1: *User behavior, as expressed by grouping jobs by user name, is the most predictive common feature. Logical job names, as generated by combining the executable name and parameters, are a reliable way to detect job repeats but are often missing from traces.*

To gauge the effect of a feature on model performance it is common to perform *ablation studies* in the machine learning community, i.e., remove features to measure their effect on prediction accuracy. Our ablation study of the Mustang trace found that using personalized features was enough to match the results of using all the features, as shown in Figure 1. The tails of estimate error (i.e., larger than $\pm 100\%$) are captured in the leftmost and rightmost bins. We analyze these plots by assuming that jobs with low estimate error (e.g., within $\pm 5\%$) are are accurate enough to make sensible scheduling decisions; the effects of larger errors is hard to predict and is further discussed in Section 4. We use combinations of the user name and

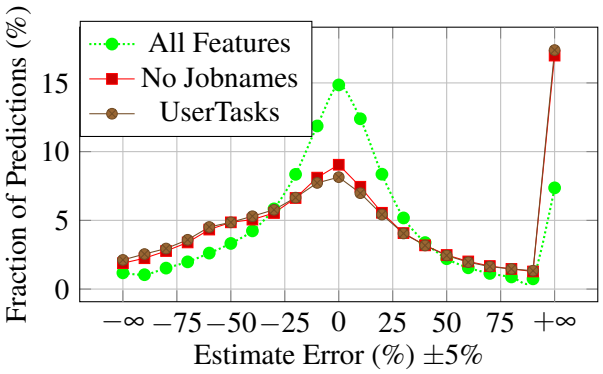


Figure 2: At times, one feature can be single-handedly responsible for model performance. By leaving our job names generated using the executable name and parameters, accuracy for the Google trace drops significantly.

number of tasks requested (*UserTasks*) to get the highest performance; using the number of tasks alone (*Tasks*) results in a 6.5% drop in accuracy. Using only features such as the hour of day (*HourOfDay*) and day of the week (*DayOfWeek*) the job was submitted results in more than 25% lower accuracy, since they fail to reliably identify users. Our results are in line with prior work that has shown both the user name and the number of tasks to have high predictive performance on LANL Mustang [1]. This observation, however, does not hold for the Google trace, as shown in Figure 2. Instead, the logical names of jobs are needed to achieve improved performance. While not shown, the PSC Bridges trace does not contain logical job names and is robust to removal of user names, losing only 3% accuracy. We suspect this is due to the PSC trace having 4000 active users in the system within a span of 1 year, a number that is at least an order of magnitude higher than what we encounter in typical traces.

Observation 2: *Extreme user skew, where most of the users only submit 5–10 jobs, and high user arrival rates are the norm across traces and over time.*

Having observed that user names is predictive, we investigate factors which may affect the reliability of predictions. First, we notice that the users submitting the most jobs in each cluster make up a significant fraction of the workload. Specifically, the top 10 users in Mustang, Google, and Bridges account for 78.9%, 55.7%, and 44.9% of the jobs, respectively, and this trend holds for CPU hours as well. Second, we find that new users arrive at high rates and use the cluster sporadically. This means that for many users we have little history to use when trying to predict their future behavior. In Figure 3 we show the arrival rates of unique users in all three clusters, and observe that the arrival rates remain stable both across traces and over time.

On Bridges, most of the 4000 unique users do not have sufficient per-user job history to learn anything interesting. A quarter of Bridges users submit fewer than five jobs, and 42% of the users submit fewer than ten. We can see this in Figure 4, where we plot a heatmap of user submissions over time on a weekly basis. In general, we observe that user activity is often highly intermittent and does not extend throughout the trace. This means learning from user behavior poses challenges, though this is obscured by the user activity skew found in the datasets.

Observation 3: *Removing a few active users can significantly affect model performance.*

Knowing that user behavior can be important and that individual users may have varying amounts of history, we remove the top N users from the traces, ranked by the number of submitted jobs. In Figure 5 we show model performance after removing the top 20 users while using all features. We see that performance degrades significantly as this results in the bulk of user history being removed for both Mustang and Bridges traces. The Google trace (not shown) is not as significantly affected by user removal. This calls into question

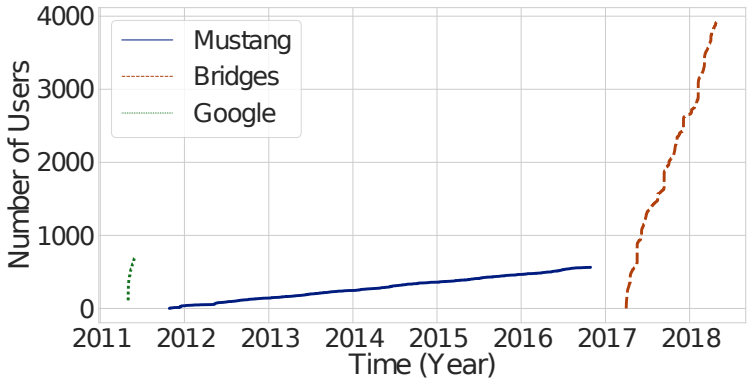


Figure 3: The amount of unique users in traces over time. For all three traces, the rate of arrival of new users remains stable over time.

what machine models actually learn when trained on real traces: do they learn the underlying behavioral trends of users and applications, or are results strongly biased toward the most common outcomes? Bridges suffers just as Mustang does from these modifications; the top few users can dominate performance even if there are thousands of users in total.

We conclude that there is a strong case for increased feature diversity. Using solely user-based behavior for prediction is not reliable, since user behavior is highly variable. A steady stream of new users enter the system, and a few users dominate each trace. Instead, we encourage future traces to be released with features that describe submitted jobs, like the described logical job name does, but without losing the generality found with features that aren't categorical and trace specific (e.g., hour of day or tasks requested).

3 Inability to adapt to change

Our analysis confirms that workload characteristics, such as the number of jobs submitted, are *non-stationary*. This means that these metrics exhibit variable variance over time, without returning to a long-run mean. To cope with changes in trace properties over time, existing work incorporates adaptation techniques such as sliding windows [35, 26]. However, even with such approaches there is error in predictions over time. In this section we investigate how model performance may fluctuate over time. We expect that there is a *ramp up* period before a model reaches good accuracy due to an initial lack of training data, but we would like to measure the length of such a period. To do this we analyze the time it takes for a model to perform at "full capacity", which we define as obtaining similar cumulative accuracy as a model trained using the full trace.

Observation 4: *Model performance can fluctuate by more than 40x on a day-to-day basis.*

Model performance is highly variable, as seen in Figure 6. We group prediction errors by day, allowing us to capture means and confidence intervals. We show performance can fluctuate daily by more than 40x in Mean Absolute Percentage Error (MAPE). It's worth noting that individual errors can be negative, and we take the absolute values of these errors. We further normalize the error by the actual runtime and capped values to 100%. This means that the variations could potentially be *higher* without capping. The Google trace is only a few months long, while the other two traces span years. This allows us to measure meaningful variations in model performance. We also note that there are some correlations between misprediction rates and trace features — the low rates of mispredictions for Mustang around 2014 correlate with the introduction of shorter jobs. One conclusion to make from this is that a single accuracy number is not sufficient to capture model performance. Rather, models should be evaluated over long traces and summaries of their performance should be reported. Further, it may not be sufficient to use only recent data, as larger training

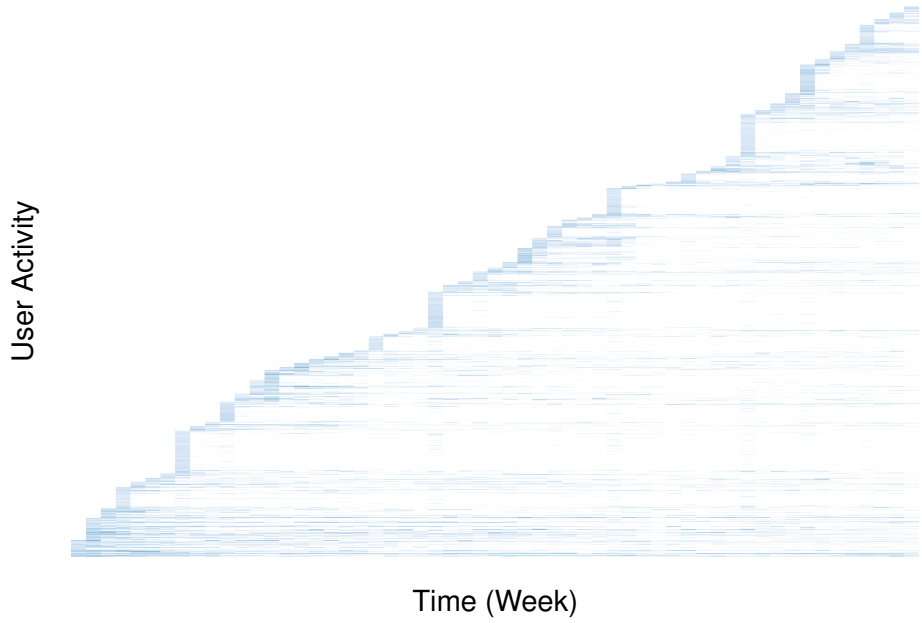


Figure 4: The count of user weekly job submissions (log scale) for the PSC Bridges trace over time. Users appear over time, enforcing a diagonal-like pattern. Many users do not submit many jobs and therefore have 0 (white) or a low amount of submissions (light blue) for most weeks. Users sometimes have many jobs submitted in a week (dark blue).

sets seem to boost performance.

Observation 5: *Learning can take between a hundred thousand to a million trace events to level off at a stable state. This represents months of data before a model is ready to be deployed.*

Predictor performance is highly variable. How much of this variability is due to difficulties in the trace compared to difficulties caused by a lack of learning? To answer this, we evaluate the cumulative accuracy of 3Sigma using all features to show how 3Sigma predictions evolve over time. We show results for the Mustang trace in Figure 7 and for the Google trace in Figure 8. In the Mustang trace, we can see that it takes 3Sigma about *one million* records (around 850 days) to reach the same cumulative accuracy as using the whole trace. This shows that if rich features are not available, then real ML systems may not be deployable until years after a cluster comes online, which is impractical. The Google trace follows a similar trend, though at a much shorter time scale, since the trace itself is much shorter. Interestingly, when we ran the same test with job names removed, the learning over time is relatively constant and poor in performance. This points to job names being learned over time as they appear. Doing the same evaluation, i.e., using all features, with PSC Bridges results in highly variable performance, with accuracy sharply dropping off at about 10^5 records, representing roughly 3 months of data. This is an indication that some parts of the trace are harder to learn from, but the learning period is short.

4 Accuracy is not enough

Observation 6: *Errors seem to matter more if the system is under load. Under high load, latencies may scale proportionally with error. In general, the relationship between error and latency/goodput is not necessarily monotonic or easily characterizable.*

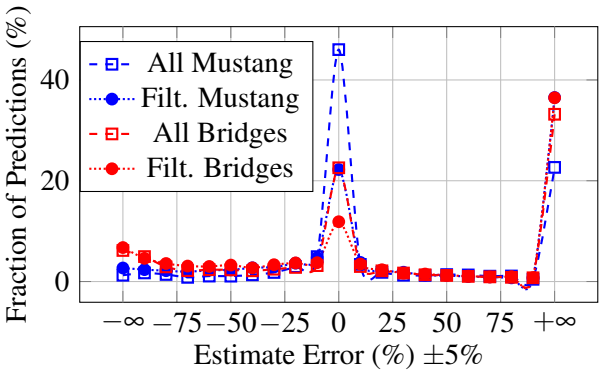


Figure 5: Certain values of a given feature may determine the effectiveness of the model itself. For our traces, performance suffers once power users are removed. The figure shows prediction accuracy when the 20 most active users are removed from the LANL Mustang and PSC Bridges traces.

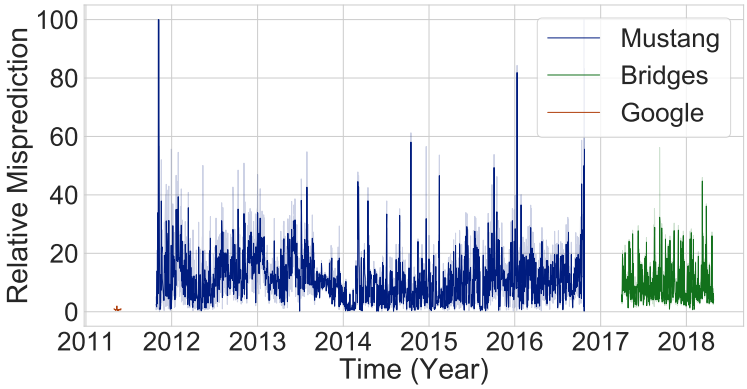


Figure 6: Misprediction variance is the norm. Shown are the *daily* relative misprediction scores of 3Sigma for each trace, over time. 95% Confidence Intervals are also shown. The mean estimation error varies wildly, often by more than 40x.

Previous work has evaluated the effect of error on system performance. TetriSched runs tests with error injected to measure SLOs/latency [32]. 3Sigma measures the effect of runtime shift and width on a distribution-aware and distribution-unaware scheduler [26]. Similarly, Stratus injects multiplicative error[2] in the context of cloud scheduling and finds that error controlling mechanisms are necessary for good system performance (i.e., heuristics are used to fix model errors) [3]. In the context of backfilling with user estimates, prior work indicates that inaccurate estimates of a user's job duration can *improve* cluster efficiency under backfilling [13, 24, 36]. To make matters more complicated, inaccurate estimates can be either harmful or beneficial depending on how aggressive backfilling is [8, 9]. These effects are due to scheduling holes being occupied by small jobs. This favors small jobs in the same way as shortest job first (SJF) policies do [11]. More aggressive backfilling allows large jobs to block small jobs. Theoretical analysis has recently been conducted on the robustness of latency-driven scheduling policies, i.e., ability to minimize expected service time [29, 23].

We next provide experimental results using the 3Sigma system on the Mustang trace. The objective of 3Sigma is to maximize the total work completed (goodput), as well as minimizing the best effort job latency. This is done via utility functions, which incorporate reward completed jobs while adding aggressive penalties for missing SLOs. We report the performance of the system measured by goodput (machine-

[2]Job task runtimes are scaled by a factor sampled from a uniform distribution.

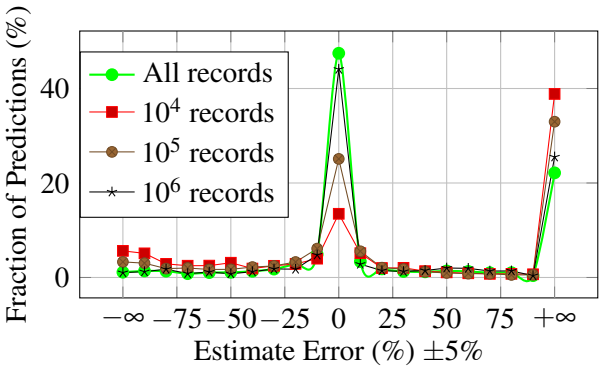


Figure 7: How much data does it take to train an accurate model? We find that high cumulative accuracy, relative to training with the whole trace, is not achieved until millions of job samples are seen for the LANL Mustang trace.

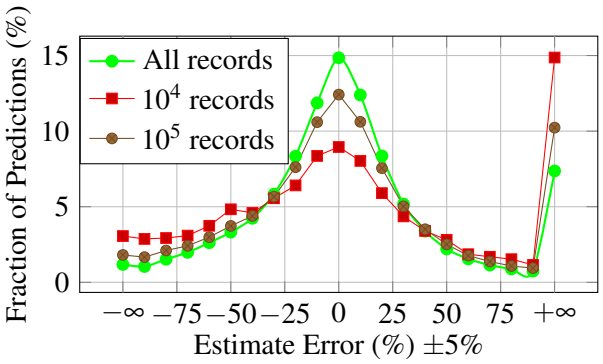
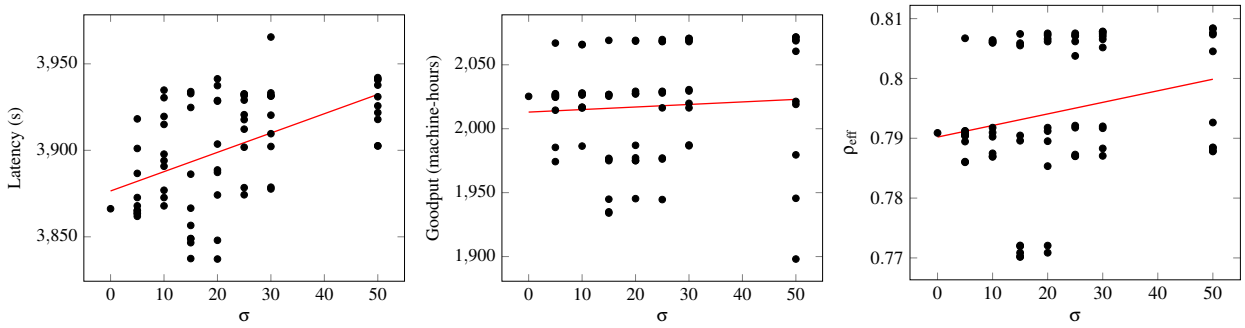


Figure 8: Similar to Figure 7, we find that high cumulative accuracy for the Google trace is not achieved, until hundreds of thousands of job samples are seen. Note that the entire trace contains a few million jobs.

seconds), job completion latency (seconds), and effective utilization of the cluster (fraction of cluster), as we perturb the job's true runtime with (additive) Gaussian noise. This error model differs from prior work, and it allows us to measure the effect of the magnitude of errors in an intuitive way. A job's true duration is affected by some noise, such as network congestion, interference between services, and the state of caches in the system. By adding noise to the true duration, we can model how the system breaks down under variability which is impossible to avoid.

Formally, let T_j be the actual runtime of a job (in seconds), indexed by j, and let our perturbation $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. The job runtime estimate provided to the scheduler is then $T_j^* = T_j + \varepsilon$. We sample from T_j^* to obtain point estimates and clamp negative values to 0. Since Mustang is a large HPC cluster, the number of tasks submitted to be packed can be larger than is feasible to simulate. We divide the number of tasks by 24 and we simulate the trace on a 512-node cluster. As expected, we do not see any variations when the cluster is not being highly utilized; when all jobs fit in the cluster, scheduling can be done first-come-first-serve and the predictor is not used. Therefore, we chose a window of time in the trace corresponding to 5 hours where a large number of completed jobs were submitted (allowing for 95 jobs to complete when the scheduler is provided with perfect oracular predictions) and fits well to our cluster. We re-ran each test 10 times with different random seeds and report the mean performance. Latency, goodput, and ρ_{eff} plots are shown in Figure 9.

Our results show some patterns; namely, latency is increased with these induced errors, but goodput and ρ_{eff} are more robust. Latency is improved by scheduling jobs conditioned on their runtime (e.g., favor



(a) Latency (b) Goodput (c) ρ_{eff}

Figure 9: The effect of Gaussian error with various level of σ on latency, goodput, and ρ_{eff}. $\sigma = 0$ corresponds to perfect (oracular) knowledge of runtimes. Latency increases notably with higher values of σ. Goodput and ρ_{eff} stay relatively constant. The variance of variables changes with σ.

small jobs), and we perturb all runtimes, causing some short jobs to be given lower priority than longer jobs. This effect is potentially even worse for very small jobs, since a small error can lead to a large percentage change in the job runtime. Goodput and ρ_{eff} are relatively unaffected (i.e., goodput correlation is close to 0 and ρ_{eff}=0.79 ± 0.02) by the induced errors, because any work done by the cluster contributes equally to goodput. The only factor that can improve goodput is packing density, and the induced error may lead to more fragmented use of resources by causing suboptimal packing decisions to be made. Although predicting the effect of error on end-to-end performance may be difficult, there are some situations where error has no effect. One example of this is when utilization is close to 0, which is when no packing decision must be made. To illustrate this effect, we pick a segment of the trace that has two jobs allowed to finish in five hours. We observe that goodput and latency is constant (variance is zero) for *all* settings of σ. Since this results in a utilization of 0.0675, there is little room for queuing to occur, and therefore prediction error does not matter. Therefore, there is a strong connection between the expected utilization of the cluster and the utility of a prediction system. There is little incentive to have a predictor for a poorly utilized cluster, which is often the case if the cluster is sized for peak utilization.

However, the latency trends are not universal — especially for intermediate values of ρ_{eff}. When we re-run this experiment on a segment of the trace with only sixty-nine jobs being allowed to finish, we see that latency is lowest with $\sigma = 50$. This is due to two runs having latencies that were nearly 100 points lower than the average, thus leading to both high variance and lower than average latency. These results agree with prior work that inaccuracy can both hurt and help cluster efficiency. We reached the same counter-intuitive results when we downscaled the cluster to 256 nodes and used the same portion of the trace — utilization was boosted from 0.65 to 0.68, but it was not enough to result in sensible latency trends. Our conclusion is that it is not just the magnitude of errors that matter: both the *time* in the trace and *magnitude* of the errors matter. At the very least, low utilization periods are not impacted by model predictions. Some clusters may see massive benefits from high accuracy (such as virtual clusters that can manipulate their own utilization [3], or clusters that benefit from backfilling), while others may not see much of a benefit at all. Many clusters, such as those found in HPC contexts, aim to maximize the goodput of the cluster rather than latency; we do not see a benefit of ML-driven scheduling in these contexts. One improvement that can be made to metrics is to explicitly account for utilization effects in accuracy results (e.g., discounting accuracy numbers obtained from trace segments corresponding to low system utilization). Doing so could help account for cluster-level utilization variations, as well as temporal accuracy variations, such as those discussed in Section 3, when predicting end-to-end performance from accuracy alone.

5 Conclusion

The continuing rise of machine learning in both research and practice invites its use in real systems. However, unlike image and text datasets that have been continously enabling more sophisticated machine learning models by growing in both size and diversity, scheduling datasets of today are few in number and look shockingly close to those of two decades ago. The state of publicly available data has created challenges with both data diversity and the length of traces, and it is impacting the design and evaluation of current models. In this paper, we explored model performance on current datasets under a microscope, and we uncovered sources of concern with limited features, data skew, non-stationarity, and the impact of a predictor on scheduling performance. The next steps in making ML-driven schedulers practical are to address the problems commonly found in datasets while also formulating concrete measures of goodness for predictors so that progress can be measured and improved.

Note. This manuscript has been approved for unlimited release and has been assigned LA-UR-19-25043. This work has been authored by an employee of Triad National Security, LLC which operates Los Alamos National Laboratory under Contract No. 89233218CNA000001 with the U.S. Department of Energy/National Nuclear Security Administration.

References

[1] George Amvrosiadis, Jun Woo Park, Gregory R. Ganger, Garth A. Gibson, Elisabeth Baseman, and Nathan DeBardeleben. On the diversity of cluster workloads and its impact on research results. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 533–546, Boston, MA, 2018. USENIX Association.

[2] Xin Chen, Charng-Da Lu, and Karthik Pattabiraman. Predicting job completion times using system logs in supercomputing clusters. In *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*, pages 1–8. IEEE, 2013.

[3] Andrew Chung, Jun Woo Park, and Gregory R. Ganger. Stratus: Cost-aware container scheduling in the public cloud. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, pages 121–134, New York, NY, USA, 2018. ACM.

[4] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167. ACM, 2017.

[5] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms, 2018. URL: https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15719-s18/web/lectures/719_S18-L26_Fontoura.pdf.

[6] Jeff Dean. Machine learning for systems and systems for machine learning. 2017. http://learningsys.org/nips17/assets/slides/dean-nips17.pdf.

[7] Richard Evans and Jim Gao. Deepmind ai reduces google data centre cooling bill by 40%, 2014. https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/.

[8] Dror G. Feitelson. Metric and workload effects on computer systems evaluation. *Computer*, (9):18–25, 2003.

[9] Dror G. Feitelson. Experimental analysis of the root causes of performance evaluation results: a backfilling case study. *IEEE Transactions on Parallel and Distributed Systems*, 16(2):175–182, 2005.

[10] Dror G. Feitelson. *Workload modeling for computer systems performance evaluation*. Cambridge University Press, 2015.

[11] Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Parallel job scheduling—a status report. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–16. Springer, 2004.

[12] Dror G. Feitelson, Dan Tsafrir, and David Krakov. Experience with using the parallel workloads archive. *Journal of Parallel and Distributed Computing*, 74(10):2967–2982, 2014.

[13] Dror G. Feitelson and Ahuva Mu'alem Weil. Utilization and predictability in scheduling the ibm sp2 with backfilling. In *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pages 542–546. IEEE, 1998.

[14] Andrew D. Ferguson, Peter Bodik, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca. Jockey: guaranteed job latency in data parallel clusters. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 99–112. ACM, 2012.

[15] Jim Gao and Ratnesh Jamidar. Machine learning applications for data center optimization. 2014.

[16] Amanda Gasparik, Chris Gamble, and Jim Gao. Safety-first ai for autonomous data center cooling and industrial control, 2018. https://www.blog.google/inside-google/infrastructure/safety-first-ai-autonomous-data-center-cooling-and-industrial-control/.

[17] Eric Gaussier, David Glesser, Valentin Reis, and Denis Trystram. Improving backfilling by using machine learning to predict running times. In *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–10, Nov 2015.

[18] Milad Hashemi, Kevin Swersky, Jamie A. Smith, Grant Ayers, Heiner Litz, Jichuan Chang, Christos Kozyrakis, and Parthasarathy Ranganathan. Learning memory access patterns. *arXiv preprint arXiv:1803.02329*, 2018.

[19] Seung-Hye Jang, Valerie E. Taylor, Xingfu Wu, Mieke Prajugo, Ewa Deelman, Gaurang Mehta, and Karan Vahi. Performance prediction-based versus load-based site selection: Quantifying the difference.

[20] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pages 489–504. ACM, 2018.

[21] Andréa Matsunaga and José A.B. Fortes. On the use of machine learning to predict the time and resources consumed by applications. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 495–504. IEEE Computer Society, 2010.

[22] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2430–2439. JMLR. org, 2017.

[23] Michael Mitzenmacher. Scheduling with Predictions and the Price of Misprediction. *arXiv e-prints*, page arXiv:1902.00732, Feb 2019.

[24] Ahuva W. Mu'alem and Dror G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. *IEEE transactions on parallel and distributed systems*, 12(6):529–543, 2001.

[25] Anne-Cécile Orgerie, Laurent Lefèvre, and Jean-Patrick Gelas. Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. In *2008 14th IEEE international conference on parallel and distributed systems*, pages 171–178. IEEE, 2008.

[26] Jun Woo Park, Alexey Tumanov, Angela Jiang, Michael A. Kozuch, and Gregory R. Ganger. 3sigma: Distribution-based cluster scheduling for runtime uncertainty. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, pages 2:1–2:17. ACM, 2018.

[27] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7. ACM, 2012.

[28] Brent Rood and Michael J. Lewis. Grid resource availability prediction-based scheduling and task replication. *Journal of Grid Computing*, 7(4):479, 2009.

[29] Ziv Scully and Mor Harchol-Balter. SOAP Bubbles: Robust scheduling under adversarial noise. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 144–154. IEEE, 2018.

[30] Warren Smith, Ian Foster, and Valerie Taylor. Predicting application run times using historical information. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 122–142. Springer, 1998.

[31] Warren Smith, Valerie Taylor, and Ian Foster. Using run-time predictions to estimate queue wait times and improve scheduler performance. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 202–219. Springer, 1999.

[32] Alexey Tumanov, Timothy Zhu, Jun Woo Park, Michael A Kozuch, Mor Harchol-Balter, and Gregory R. Ganger. Tetrisched: global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters. In *Proceedings of the Eleventh European Conference on Computer Systems*, page 35. ACM, 2016.

[33] Daniel Vladušič, Aleš Černivec, and Boštjan Slivnik. Improving job scheduling in GRID environments with use of simple machine learning methods. In *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, pages 177–182. IEEE, 2009.

[34] John Wilkes. More google cluster data. https://ai.googleblog.com/2011/11/more-google-cluster-data.html, 2011. Accessed: 2018-10-14.

[35] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, Murali Chintalapati, and Dongmei Zhang. Improving service availability of cloud systems by predicting disk error. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 481–494, Boston, MA, 2018. USENIX Association.

[36] Dmitry Zotkin and Peter J. Keleher. Job-length estimation and performance in backfilling schedulers. In *Proceedings. The Eighth International Symposium on High Performance Distributed Computing (Cat. No. 99TH8469)*, pages 236–243. IEEE, 1999.