Early Prediction of High Performance Computing Job Outcomes via Modeling System Text Logs

Alexandra DeLucia*
Center for Language and Speech Processing
Johns Hopkins Whiting School of Engineering
Baltimore, Maryland, USA
aadelucia@jhu.edu

Elisabeth Moore
Information Sciences Group
Los Alamos National Laboratory†
Los Alamos, New Mexico, USA
lissa@lanl.gov

***Abstract*—The massive scale of high performance computing (HPC) machines necessitates using automatic statistical methods to assist human operators in monitoring day-to-day behavior. We address the problem of identifying problematic compute jobs by modeling system logs, which record all activities on the machine in near-natural language form. We apply techniques from relational learning and human language technology, incorporated with domain knowledge, to extract features from system logs produced by approximately 10,000 HPC jobs. We evaluate the usefulness of these features via a random forest model to predict job outcome state. We compare our models to a baseline which mimics state-of-the-art human operator behavior, and find that the best-performing feature set is one which combines domain knowledge with simple aggregate metrics. Our method predicts job outcomes with an F1 score approaching 0.9 after a job has been running for 30 minutes, giving an average lead time of 3 hours before failure.**

***Index Terms*—HPC Job Prediction, System Modeling, Machine Learning**

I. Introduction

High performance computing (HPC) machines produce on the order of terabytes of monitoring information daily, ranging from user activity to hardware telemetry, making efficient monitoring solely by human operators intractable. One crucial aspect of monitoring large-scale computing systems is the identification of problematic user-submitted jobs, such as jobs that are likely to fail. In our sample dataset of 10,000 jobs from HPC clusters, approximately 13% of jobs did not complete successfully, occupying roughly 14,000 compute hours[1].

In this study, we expand upon previous work applying machine learning techniques to predict the outcome of a job based on textual system log (*syslog*) messages. Previous work found that topics learned from syslog messages are highly predictive of job outcome [1]. We expand the variety of features engineered and extracted from syslog messages, and use these features for early prediction of job outcomes — i.e., before a job ends. We compare our approach to a baseline using features drawn from keywords most commonly used by system administrators, and test our technique's generalizablity across different compute clusters. The ultimate goal is to develop a tool that will raise alerts in real time for compute jobs which are likely to experience a failure.

Our work presented here makes the following contributions:

- A machine learning approach for early detection of problematic HPC jobs that surpasses current state-of-the-art methods
- Exploration of human language technology and other feature engineering/extraction techniques for analysis of computer-generated text logs

II. Related Work

The HPC community relies heavily on computer-generated text logs, especially syslog, to monitor system health and provide insights regarding failures [2], [3]. In general, previous work on statistical analysis of system log messages focused on fault diagnosis [4], anomaly detection [5]–[10], and node failure prediction [11], which is distinct from our goal of job outcome prediction. Prior work also exists using I/O and other compute node activity data (e.g. workload traces) to monitor and predict job outcome [12], but our work is novel in using syslog messages to predict job outcome. Our work achieves better prediction performance than other existing approaches.

Prior work on analysis of syslog falls roughly into two groups: (1) systems-oriented and (2) data science-oriented. The most common technique for log message analysis in the systems community is to create regular expressions for messages, either by hand or automatically [3], and then use these patterns to assign labels to syslog messages. The data science approach views the logs as a rich inhomogeneous combination of text, numerical, and temporal data. Common approaches in this space include graph analysis and human language technology [6], [11], [13], [14].

Our work also distinguishes itself from prior work by accomplishing a clear predictive task rather than detecting anomalies. For this reason, we not only propose a predictive technique, but also report quantitatively on its performance (as opposed to the anomaly detection task where quantitative evaluation is difficult).

*This work was performed during an internship at Los Alamos National Laboratory.

†This manuscript has been approved for unlimited release and has been assigned LA-UR-20-27025. 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.

[1]Calculated by multiplying the total compute time of jobs with a `Failed` or `Node Fail` outcome by the number of compute nodes it used.

III. Available Data

We use a sample of 10,000 jobs from two HPC clusters, roughly 5,000 per cluster, located at Los Alamos National Laboratory (LANL) from June 2018. The first cluster, Wolf, contains 616 compute nodes running Clustered High Availability Operating System with a total of 9,856 Dual 8-core Intel Xeon (Sandy Bridge) processors with InfiniBand [15]. The second cluster, Grizzly, contains 1,490 compute nodes running Tri-Lab Operating System Stack 3 with a total of 53,640 Dual 18-core Intel Xeon Broadwell processors with Intel OmniPath Interconnect [16]. Both clusters use Slurm workload manager for job scheduling [17]. Wolf and Grizzly are available for use by scientists at LANL, Lawrence Livermore National Laboratory, and Sandia National Laboratories.

A. System Logs (Syslogs)

Syslog messages are short near-natural language text that describe most activity occurring on a compute node. These textual log messages are extremely important for understanding the overall state of a cluster or a particular compute node, especially in the event of a failure. Although a single compute node may generate a variety of textual logs, a human operator's starting place for root cause analysis is usually the syslog, as it records all processes and system events. Each syslog message contains a timestamp, a node name, a "tag" — i.e., the name of the daemon or other system process that generated the message, and the textual message itself.

B. Job Logs

HPC job schedulers generate a record for each job that is submitted and store these records in a "job log." Each job's record contains detailed information, including the resources it requested, the job's start and end times, the allocated nodes, and the job's final outcome state. Possible outcomes are: `COMPLETED`, `FAILED`, `NODE_FAIL`, `TIMEOUT`, and `CANCELLED` (Table I). In this work we consider `COMPLETED` and `TIMEOUT` jobs to be "okay" because there were no issues with the computation, and `FAILED` and `NODE_FAIL` jobs to be "problematic." Jobs that are `CANCELLED` are not used in this experiment because cancellations occur due to an external decision made by a user or administrator, and therefore are not predictable via syslog.[2]

Job State	Description	Okay or Problem
COMPLETED	Job completed successfully	Okay
TIMEOUT	Job did not finish in the allowed time limit	Okay
FAILED	Job did not complete for some reason (e.g. program bug)	Problem
NODE_FAIL	One or more of the job's compute nodes failed (e.g. filesystem error)	Problem

TABLE I: Enumeration and description of relevant job states.

[2]We have confirmed that our prediction scheme mis-predicts cancelled jobs predominantly as completed. Our best performing model predicts cancelled jobs as follows: 65% (+/- 2%) completed, 19% (+/- 4%) failed, 16% (+/- 4%) timeout, 0% node fail.

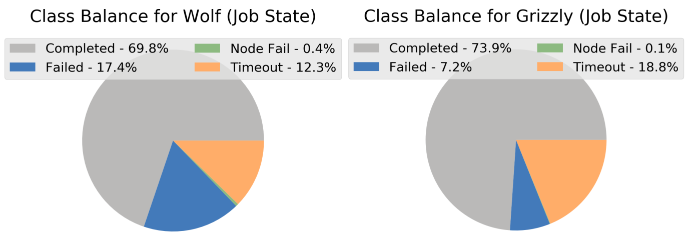


Fig. 1: Class balances on Wolf and Grizzly. Jobs considered "okay", i.e. `COMPLETED` and `TIMEOUT` jobs, greatly outnumber the "problem" classes, i.e. `FAILED` and `NODE_FAIL`.

C. Data Preparation

We start with syslog messages stored in an internal LANL system, Tivan [18]. We query Tivan's ElasticSearch database for syslogs and job logs from Wolf and Grizzly from June 2018, and then sample 5,000 jobs from each cluster using the Python Pandas package sampling function [19], excluding `CANCELLED` jobs.

We organize our data by grouping syslog messages by the job which produced them (identified by matching the syslog message's timestamp and node origin to the job that was running on those nodes at that time) and labeling the resulting syslog chunks with the corresponding job outcome label provided in the job log. We parse out separate components of the syslog messages (raw text, hexadecimal values, and decimal values) as shown in Table II.

D. Raw Data Exploration

To get a sense of what's included in our syslog/joblog dataset, we investigate the distributions of a few basic properties. First, we note the proportion of job outcomes in each cluster; the large majority of jobs complete successfully or timeout, while fewer experience general failures, and a small minority experience node failures. The class balances for Wolf and Grizzly are shown in Figure 1.

Second, we investigate a potentially problematic confounding variable. We raise the concern that job outcome may be correlated with job size (e.g. larger jobs may simply be more prone to timing out). To remove this concern, we show the distribution of job size (measured in number of nodes allocated) across job outcomes in Figure 2. We observe that there does not appear to be strong enough signal in job size alone to distinguish between job outcomes, so we proceed with extracting more detailed features.

IV. Approach

Our strategy for accurate and actionable prediction of job outcome involves two phases: (1) feature extraction from a job's syslog messages and (2) early prediction of job outcome.

A. Feature Extraction

We investigate the usefulness of a variety of feature set types for the prediction of job outcome. Our techniques are borrowed from the fields of relational learning, human

Job ID	Job State	Syslog Tags	Syslog Times	Syslog Text	Syslog Hex	Syslog Dec
0001	COMPLETED	temp_sensors, temp_sensors, sshd, sshd, kernel	2018-06-27 14:30:02, 2018-06-27 14:30:03, 2018-06-27 14:30:03, 2018-06-27 14:30:04	pam_unix sshd session session opened user root uid, coretemp-isa- physical id, coretemp-isa- physical id, pam_unix sshd session session closed user root, lustre -ost -osc- connection restored -ost	ffff8808255fc400	0.0, 0.0, 0.0, 1.0 66.0, 1.0, 51.0

TABLE II: Organized and cleaned data, ready for feature extraction. Syslog messages are grouped by corresponding job, and components of syslog messages are separated via simple parsing.

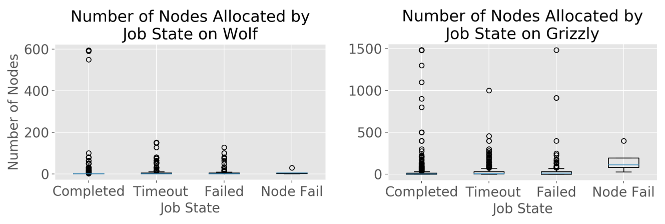


Fig. 2: Distribution of job size (number allocated of nodes) by job outcome. While some variation exists, there is not enough signal to use job size alone for prediction.

#	Feature Set	Description
1	Numerical	Average, standard deviation, and count for decimal and hexademical values present in syslog (normalized by number of syslog messages in the job)
2	Temporal	Average and standard deviation of time between messages, and total time between first and last message using syslog timestamps
3	LDA	Distribution of syslog message text across LDA topics
4	Infomap	Distribution of a job's syslog text across Infomap clusters
5	TFIDF	Term frequency-inverse document frequency of a job's syslog text
6	Process Tag	Distribution of process tags within a job's syslog messages
7	Numerical & Temporal	Combined temporal and numerical features
8-11	Combined sets	Each text feature set combined with the Numerical & Temporal set, e.g. LDA and Numerical & Temporal
12	Baseline	Frequency and counts of common human operator-defined keywords in a job's syslog

TABLE III: A complete list of feature sets we investigated for early prediction of job outcome.

language technology, and systems. Using the raw data from syslog, cleaned as described above, we extract features from each component of syslog separately, following previous work [6]. All feature sets we investigated are listed and summarized in Table III, and detailed in the following subsections.

1) Numerical and Temporal Features: The numerical features use simple aggregate metrics, listed in Table III, to summarize the numerical content of the job's syslog. We separate the hexadecimal- and decimal-encoded values to preserve the significance of their presence in a message. For example, as in Figure 3a, hexadecimal usually represents a memory address, whereas a decimal could be from a temperature monitoring process. Therefore, for each of hexadecimal and decimal, we extract the average value, standard deviation of the value, and count of values. As for the temporal analysis, we extract the average and standard deviation of time between syslog messages, as well as the total time difference between the first and last syslog messages, as summarized in Table III. These temporal features indicate how "chatty", or active, a job is. An example calculation of the metrics are in Figure 3b.

2) Text Analysis: In order to incorporate the content included in the actual text of the syslog message, we compare four text-based feature extraction techniques, all of which focus on clustering words from syslog and summarizing a job's syslog content as a distribution over clusters.

a) Latent Dirichlet allocation (LDA): LDA is a popular topic modeling technique which learns the distribution of latent topics across a "document" in a "corpus" [20]. In this work we define a group of cleaned syslog message text from a single job as a document. LDA clusters words, or tokens, into "topics." We then summarize the syslog message content for a job via the distribution of topics contained within the job's syslog text. Example topics learned from our syslog text are shown in Table IV.

b) Term frequency-inverse document frequency (TFIDF): TFIDF is a natural language processing technique. A basic improvement upon token counts, TFIDF gives more weight to words that are unique to a document. For example, in Table II the word "sshd" would have less weight than "connection" because it appears more often across the corpus.

c) Infomap: We use Infomap, a graph clustering algorithm based on the behavior of a random walker [21], as another text clustering algorithm. First, we construct a graph from the syslog text, representing the tokens as nodes and the number of times tokens co-occur within a message as weighted edges. This approach has previously been found useful for anomaly detection in syslog [6]. Then, Infomap is used to find clusters of words in the graph, analogous to topics. We then summarize a job's syslog text content via a distribution vector across these learned clusters. Example clusters learned from our syslog text are shown in Table V.

d) Process Tag: We incorporate systems domain knowledge by summarizing the process tags present in a job's syslog messages. We first remove symbols from tags to group them into "tag types" (e.g. `sshd[200]` becomes `sshd`. Then, we use a vector of the normalized frequency of occurrence of the process tags within a job as a feature. The belief is that including the process tag frequencies should help provide more high-level but still domain-relevant information regarding job behavior (e.g., we would be able to tell that a job is interacting with the network frequently, but not the details of that interaction).

3) Baseline: The current state-of-the-art for analysis of syslog is simply human operators searching through millions of messages for previously known keywords. Our baseline model uses counts and frequencies of human-defined keywords within each job's syslog text as features. Specifically, our baseline includes counts and frequencies of the character patterns `err`, `warn`, `fail`, `time` (for indications of a timeout), `shutdown`, and `kill`.

Count Hex	AVG Hex	STD Hex	Count Dec	AVG Dec	STD Dec
1	18446612167300989952	0	7	17.0	26.5545

(a) Features summarizing numerical content of a job's syslog.

Total time difference between first and last message	AVG time between messages	STD time between messages
2	0.5	0.5

(b) Temporal features, measured in seconds.

Fig. 3: Example numerical and temporal analyses using the example from Table II and relevant features from Table III.

LDA Topic	Tokens (* usernames removed)
0	`system lustre not session user ptlrpc root pam_unixsshdsession message tainted`
1	`memory dimm event assertion sensor warn channel number rank correctable`
2*	`user pam_unixsshdsession session root closed opened segfault lustreerror ldlm_cli_enqueue cookiess`

TABLE IV: Example clusters learned by latent Dirichlet allocation (LDA) on the Wolf dataset. Only the top ten words in each group are shown.

Infomap Cluster	Tokens (* usernames removed)
1*	`user session opened pam_unixsshdsession closed root granted access pam_unixsulsession stam`
2	`id c physical memory dimm event channel assertion sensor cpu`
3*	`read remov mountstats qidxsec codeexit binpagosa exit metrics bingen coderun`

TABLE V: Example clusters learned by Infomap on the Wolf dataset. Only the top ten words in each group are shown.

B. *Job Outcome Prediction*

We use random forest models to predict job outcome at a variety of times during the job's runtime. For a proof of concept, we perform prediction after the job has finished and allow the model to use syslog messages from the entire run of the job, assuming that this is the point where the model would have the most information and be the most accurate. However, in order to discover whether the model can provide an early alert with enough lead time before failure (or timeout) for a human operator to take mitigating actions, we walk back from the end of the job (i.e. "time before end") and perform prediction at a variety of timesteps, *using only the features that would have been available at that timepoint*. We also perform prediction at a variety of timesteps measured from the beginning of the job (i.e. "time after start") in order to determine how long the model must wait to make a sufficiently accurate prediction. In this way, we aim to answer two questions: (1) How much time does a human operator have after a failure prediction is made to fix the issue, and (2) At what point in the runtime of a job should the model be run?

V. Experimental Setup

We examine the usefulness of our extracted features for predicting job outcome using random forest models. We choose to use a simple random forest model, rather than a more complicated model such as a deep network, because our focus is on proof-of-concept and investigation of feature sets. For the text content analyses which involve learning clusters, the entire text corpus was used to learn clusters, before separating into train and test sets.

We evaluate our models on two tasks: (1) predicting specific job outcome (i.e. "Multiclass" task), and (2) classifying a job as "okay" or a "problem" as defined in Section III-B (i.e. "Okay vs. Problem" task). We evaluate each of our model / feature set pairings using weighted F1, precision, and recall scores from 200 iterations with stratified shuffle split for training and testing.

On each task, we ask four questions of each of our model / feature set pairings:

1) How accurately can we accomplish the task if the model has access to the job's full syslog (proof of concept)?
2) How well does the model generalize?
3) How much lead time can the model provide for a human operator between prediction of a failure/problem and the occurrence of the failure/problem?
4) At what point during the runtime of the job should the model be run for a sufficiently accurate prediction?

Question (1) is a simple proof-of-concept question, and for question (2) we attempt to train on the Wolf system and test on the Grizzly system ("cross-platform test"). The results from questions (1) and (2) will guide our selection of the best model/feature set pairings to use for questions (3) and (4).

To answer question 3, we create multiple subsets of the syslog data based on "time before end" of the job, taking only messages from the job before its last half hour, before its last hour, etc., and try to predict job outcome with this limited information. Note that as we restrict the amount of syslog available to the model, we necessarily also alter the size of the dataset because each job runs for a different amount of time (almost all jobs run for at least 1 minute, but fewer jobs run for more than 10 hours). To avoid overfitting, we do not perform any predictions if the size of the dataset falls below 200 total jobs. Figure 4 shows our dataset size for each timestep.

To answer question 4, we similarly create multiple subsets of the syslog data, but now based on "time after start" of the job, including only messages of the job during its first half hour, first hour, etc., and try to predict job outcome using

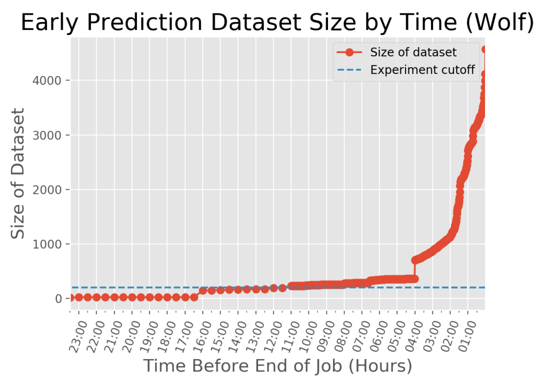


Fig. 4: Dataset size for early prediction experiment when restricted by time before end of job. Only jobs that were not completed by "time before end" were included. Only datasets larger than 200 jobs were used.

only this data. As above, as we restrict the amount of available syslog, we necessarily decrease the size of the dataset, and only run a prediction experiment if more than 200 jobs are included in the dataset. The size of the dataset decreases exponentially in a similar way to that described above.

For the analysis of our results for questions 3 and 4, we use a randomly sampled set of 200 jobs as a validation set. This validation set is the set of jobs on which we compare our learned models' early prediction capabilities at each timestep. (Using a randomly selected test set each time would not allow a direct comparison between models.)

VI. Results

A. *Proof-of-concept, Generalizability, and Model Selection*

First we discuss the results from our investigation of our experimental questions 1 and 2, detailed in Section V. Figure 5 shows F1, precision, and recall scores for random forest models trained with each of our feature set combinations, when the model is given access to the full syslog from a job. Note that in Figure 5, we show the results for training and testing on the same system: either train and test on Wolf, or train and test on Grizzly. Results for both the multiclass (predict specifically `COMPLETED`, `TIMEOUT`, `FAILED` or `NODE FAIL`) and "Okay vs. Problem" ({`COMPLETED`, `TIMEOUT`} vs. {`FAILED`, `NODE FAIL`}) tasks are shown. We find that almost all models and feature sets significantly out-perform the baseline model, meaning that the features calculated and extracted from syslog are more informative of job outcome than the human-designed keywords. This is surprising because one would expect that the final syslog messages generated by a job that fails or times out would include distinctive keywords. However, it appears that statistical information regarding the content of the full syslog from a job is more information-rich, and useful specifically for outcome prediction.

We also find that on both Wolf and Grizzly, the Okay vs. Problem task appears to be easier than the multiclass task, probably because it is quite difficult r to distinguish between a job that completes and a job that times out. From these results, it appears that the top three performing feature sets across Grizzly and Wolf are Tag and Temporal & Numerical, Infomap and Temporal & Numerical, and LDA and Temporal & Numerical. We also note that the combination including tag information, which incorporates domain knowledge is competitive with the combinations that only include statistically learned features. This is certainly an argument for including information about the task's particular domain, rather than blindly applying statistical tools.

To test generalizability, we report the result metrics for models trained on Wolf and tested on Grizzly, still with access to each job's full syslog. Figure 6 shows these results. We still see quite high F1, precision, and recall scores, although not quite as high as training and testing on the same system. This suggests that although the model may be learning some general characteristics of job behavior via syslog information, the most accurate model would be one deployed to monitor the same system it was trained on. However, the Okay vs. Problem task still appears to be much easier than the Multiclass task, almost all models out-perform the baseline, and Tag and Temporal & Numerical, Infomap and Temporal & Numerical, and LDA and Temporal & Numerical remain in the top performing feature sets. We proceed to investigate early prediction using only these three feature set combinations.

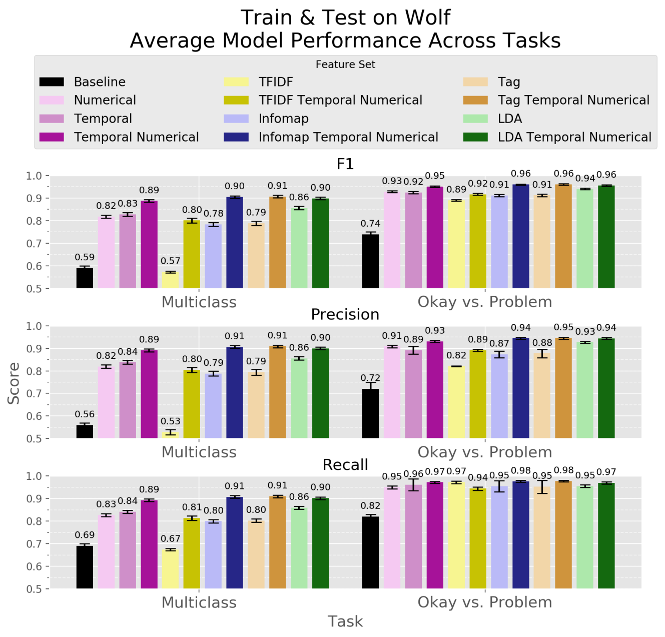
B. *Early Prediction of Job Outcome*

To determine our models' ability to accurately predict job outcome before the end of a job, we look at the models' performances if we subset syslog information based on "time before end" of a job, and if we subset based on "time after start" of a job, as described in our experimental questions in Section V.

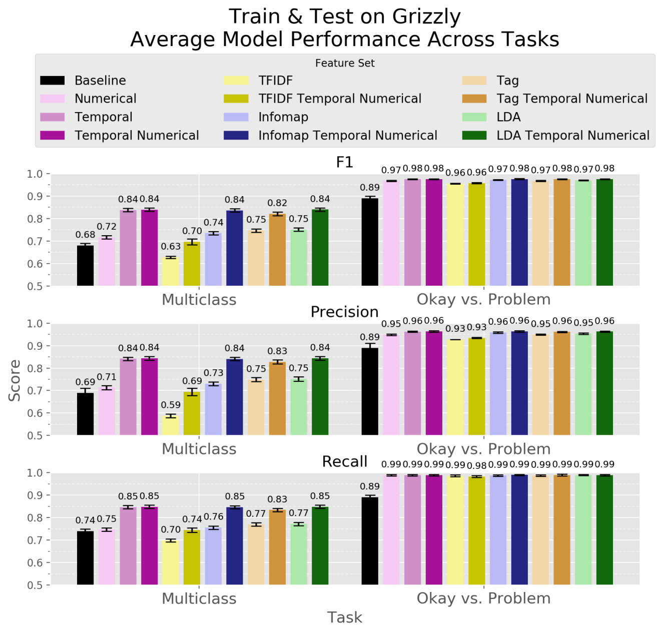
We investigate question (3): how much lead time can our model provide to human operators who may be able to take action to mitigate problems? Figure 7a shows F1, precision, and recall results at different timesteps when the syslog available to the model is restricted based on the time before the end of the job. Note that this does not give useful information for the design of a tool, because when a job starts running there is no reliable way to know when it will end[3], therefore as a job is running we do not know the "time before end." However, we can conclude that in the average case for a model trained on any of the three best-performing feature set combinations, an accurate prediction can be made with a conservative minimum of approximately 3 hours of lead time, meaning that a human operator would have at least 3 hours to take mitigating action. This would almost certainly be enough time to trigger a checkpoint or other action before the job ultimately failed or timed out.

Next, we investigate the answer to question (4): at what point during a job's runtime should a model be run to achieve sufficient accuracy? The answer to this question would drive

[3]In our dataset, the jobs ran for an average of 38% (+/- 33%) of requested time.



(a) Random forest model trained and tested on Wolf, with access to complete syslog from jobs.



(b) Random forest model trained and tested on Grizzly, with access to complete syslog from jobs.

Fig. 5: F1, precision, and recall, for each feature set combination using random forests trained and tested on data from the same system, with access to each job's full syslog information. Error bars show standard deviation.

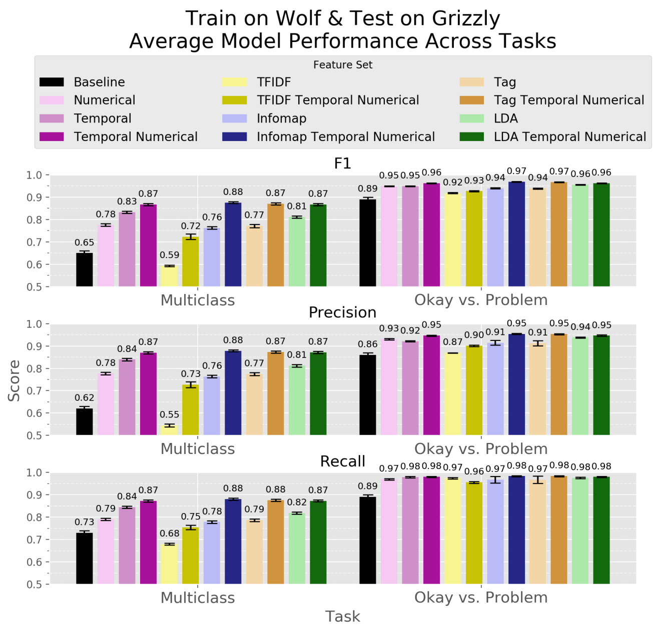
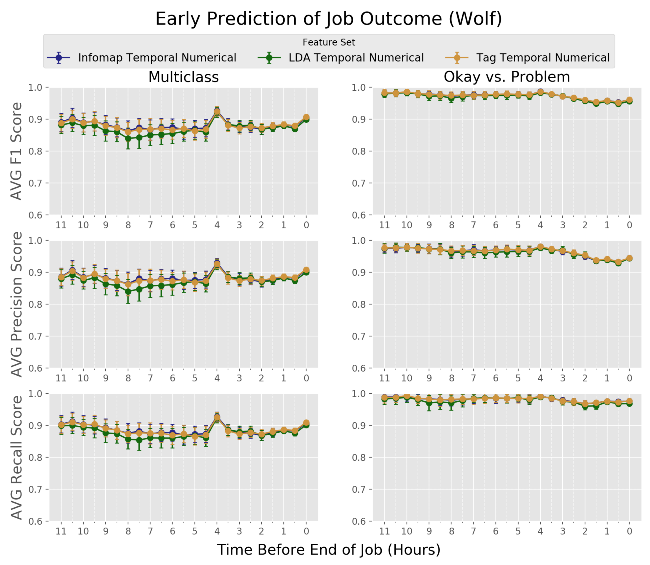


Fig. 6: F1, precision, and recall, for each feature set combination using random forests trained on Wolf and tested on Grizzly, with access to each job's full syslog information. Error bars show standard deviation.

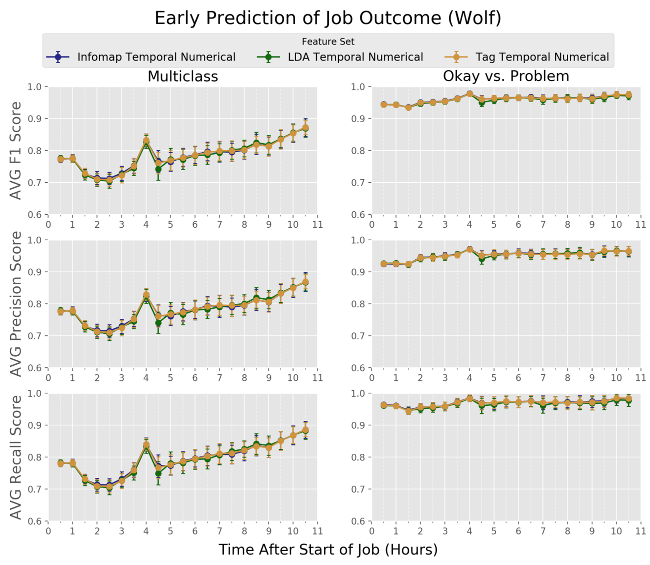
the development of a tool for job outcome prediction. Figure 7b shows F1, precision, and recall results when we limit the available syslog based on the amount of time since the start of each job, and provides insight into when during a job's run it would be reasonable to run a prediction. As expected for the more difficult Multiclass task, we see that all metrics start fairly low at the beginning of the job, and then steadily increase (roughly linearly) as the job progresses. Unfortunately, because the scores do not seem to converge quickly, this does not provide a simple answer as to when to run a prediction. However, scores for the Okay vs. Problem task are relatively stable and suggest that running a prediction even within the first 30 minutes of a job could provide accurate information regarding whether or not the job will be problematic. Therefore, one approach to building a tool would be to run an Okay vs. Problem prediction early on for a first-pass triage, and then to run a Multiclass prediction every so many minutes or hours as a problematic job progresses so that a human operator could receive more insight into the job's behavior. The model's ability to detect a problematic job within the first half hour of its running is a significant improvement over current state-of-the-art monitoring, which would usually be a post-hoc root cause analysis of the failure.

Finally, we look at predicted classes over time as two example jobs progress — one which times out, and one which fails. For this case study, we use one of our best-performing feature sets, Tag and Temporal & Numerical. Figure 8 shows the predicted class probabilities over time for both predictive tasks for these two example jobs. We note the extremely strong



(a) F1, precision, and recall scores for both predictive tasks, and for models trained using the top three performing feature set combinations, when the amount of syslog for each job is restricted by the time remaining in the job's actual runtime (not the time requested by the user submitting the job, which is usually wildly inaccurate)[a].

[a]See footnote 3.



(b) F1, precision, and recall scores for both predictive tasks, and for models trained using the top three performing feature set combinations, when the amount of syslog for each job is restricted by the amount of time elapsed in the job.

Fig. 7: F1, precision, and recall results for early prediction of job outcome. Error bars indicate standard deviation. The anomaly at 4 hours is from a data artifact (a large amount of failed jobs ended). Note: the y-axis scale starts at 0.6.

priors at work in the Okay vs. Problem task, but that the probabilities for the problem job do switch and converge to the correct answer. In general, we observe that over time the predictions in each case become more sure and converge to the correct answer. In fact, the failed job could have been identified as problematic about 190 minutes before the actual failure, and the fact that it would be a `FAIL` could have been predicted at the same time — about 10 minutes into its runtime. Similarly, the timeout job could have been identified as non-problematic as early as 1 minute into its runtime, and identified as a timeout at about 7 minutes into its runtime, which would have provided a lead time of about 50 minutes.

Note that we can use analyses as in Figure 8 to investigate at what time the most likely predicted class no longer changes. We find that on average, the most likely class stabilizes at 6.85 (+/- 23.3) minutes into the job for the Okay vs. Problem task, and at 31.43 (+/- 67.9) minutes into the job for the Multiclass task. Although these standard deviations are high, even two standard deviations above the mean gives in the average case, multiple hours of lead time before failure.

VII. Conclusion

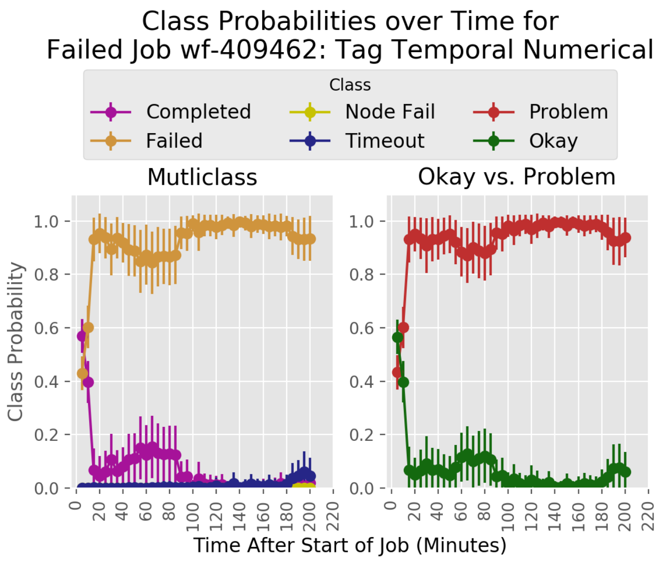
We investigate the ability of a variety of features extracted from the system log messages produced by a job running on a high performance computing system to accurately predict the final outcome of the job (successful completion, timeout, failure, or node failure). We investigate the usefulness of each of these feature sets individually and in combination, and find that while all trained models outperform our baseline (intended to mimic state-of-the-art human operator behavior), the best-performing model is a random forest trained on numerical, temporal, and process tag distribution information. Identifying non-problematic (completions and timeouts) versus problematic jobs (failures and node failures) appears to be a much easier problem than the multiclass case. However, even in the multiclass case, a model which combines temporal, numerical, and text information, and is trained and tested on messages from different HPC compute clusters can achieve an F1 score of 0.9 when given access to the jobs' full syslog messages. We also find that models trained on significantly fewer syslog messages, restricted by wallclock time during the job's run, can correctly predict job outcome on average at approximately 30 minutes into a job's runtime, giving a conservative minimum of approximately 3 hours of lead time before failure. This points to the development of a tool that would raise alerts to human operators with enough time to take mitigating action before a failure or timeout.

Acknowledgments

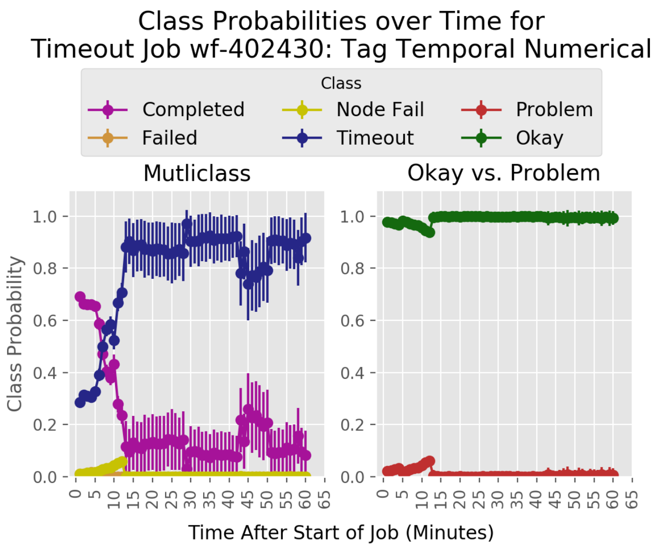
The authors thank Nathan DeBardeleben, Sean Blanchard, and Hugh Greenberg for domain expert input and data access.

References

[1] A. DeLucia and E. Baseman, "Work in progress: Topic modeling for hpc job state prediction," in *Proceedings of the First Workshop*



(a) Prediction probabilities over time for a `Failed` job. At first the model classifies the job incorrectly, but then classifies it correctly after the first 10 minutes.



(b) Prediction probabilities over time for a `Timeout` job. At first the model classifies the job incorrectly, but then classifies it correctly after the first 7 minutes.

Fig. 8: Early prediction of job outcome for two example jobs on Wolf using the best performing model, the Tag and Temporal & Numerical feature set model. Note that as time progresses, the model's probabilities seem to converge on the correct answer; in the future, this could be used to raise an early alert to a human operator. Time is measured in minutes elapsed since the start of the job, and error bars show standard deviation.

on Machine Learning for Computing Systems, ser. MLCS'18. New York, NY, USA: ACM, 2018, pp. 4:1–4:4. [Online]. Available: http://doi.acm.org/10.1145/3217871.3217874

[2] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Commun. ACM*, vol. 55, no. 2, pp. 55–61, Feb. 2012. [Online]. Available: http://doi.acm.org/10.1145/2076450.2076466

[3] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, June 2007, pp. 575–584.

[4] D.-Q. Zou, H. Qin, and H. Jin, "Uilog: Improving log-based fault diagnosis by log analysis," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 1038–1052, Sep 2016. [Online]. Available: https://doi.org/10.1007/s11390-016-1678-7

[5] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1285–1298.

[6] E. Baseman, S. Blanchard, Z. Li, and S. Fu, "Relational synthesis of text and numeric data for anomaly detection on computing system logs," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec 2016, pp. 882–885.

[7] W. Xu, L. Huang, and M. I. Jordan, "Experience mining google's production console logs." in *SLAML*, 2010.

[8] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, "Mining console logs for large-scale system problem detection." *SysML*, vol. 8, pp. 4–4, 2008.

[9] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 117–132.

[10] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online system problem detection by mining patterns of console logs," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 588–597.

[11] A. Das, F. Mueller, P. Hargrove, E. Roman, and S. Baden, "Doomsday: Predicting which node will fail when on supercomputers," in *Supercomputing 2018*, Nov 2018.

[12] X. Chen, C. Lu, and K. Pattabiraman, "Failure prediction of jobs in compute clouds: A google cluster case study," in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, Nov 2014, pp. 341–346.

[13] C. Bertero, M. Roy, C. Sauvanaud, and G. Tredan, "Experience report: Log mining using natural language processing and application to anomaly detection," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, Oct 2017, pp. 351–360.

[14] R. Vaarandi and M. Pihelgas, "Logcluster - a data clustering and pattern mining algorithm for event logs," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 1–7.

[15] Los Alamos National Laboratory, "Tri-lab computing resources," 2018. [Online]. Available: http://www.lanl.gov/asc/tri-lab-resources.php

[16] Los Alamos National Laboratory (internal), "Grizzly," 2018. [Online]. Available: https://hpc.lanl.gov/grizzly$_$home

[17] SchedMD, "Slurm workload manager." [Online]. Available: https://slurm.schedmd.com

[18] H. N. Greenberg and N. DeBardeleben, "Tivan: A scalable data collection and analytics cluster," in *The 2nd International Industry/University Workshop on Data-center Automation, Analytics, and Control, SuperComputing'18*, 2018.

[19] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445. Austin, TX, 2010, pp. 51–56.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944937

[21] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.