Modeling High Performance Computing System Log Messages for Early Prediction of Job Outcome

Alexandra DeLucia
Ultrascale Systems Research Center
Los Alamos National Laboratory

Elisabeth Moore
Ultrascale Systems Research Center
Los Alamos National Laboratory

Introduction

- Semi-supervised application of machine learning to the monitoring of high performance computing jobs
- Predicting the outcome of jobs using features from system log (syslog) produced by the compute nodes running the job

Research Questions

1. How accurately can syslog features predict job outcome?
2. Which features from syslog are most informative?
3. Can the features be used across platforms?

Background

Job Log

- Jobs are recorded by the job scheduler (e.g. Moab, Slurm) in a job log file

JobID=# UserID=# GroupID=# Name=<program name> **JobState=[COMPLETED,FAILED, NODE_FAIL,CANCELLED,TIMEOUT]** Partition=<> TimeLimit=# **StartTime=<time> EndTime=<time> NodeList=[]** NodeCnt=# ProcCnt=# WorkDir=../../

Job log entry format. The highlighted fields are used in our study.

- The job state indicates normal or problematic outcomes

Job State	Description	"Okay" or "Problem"
CANCELLED*	User cancelled the job. *These jobs are not used in this experiment.	Okay
COMPLETED	Job completed successfully	Okay
FAILED	Job did not complete for some reason (e.g. program bug)	Problem
NODE FAIL	One or more of the jobs compute nodes failed (e.g. filesystem error)	Problem
TIMEOUT	Job did not finished in the allocated time limit	Okay

System Log (Syslog)

- Syslogs give insight to process activities and are crucial for failure analysis

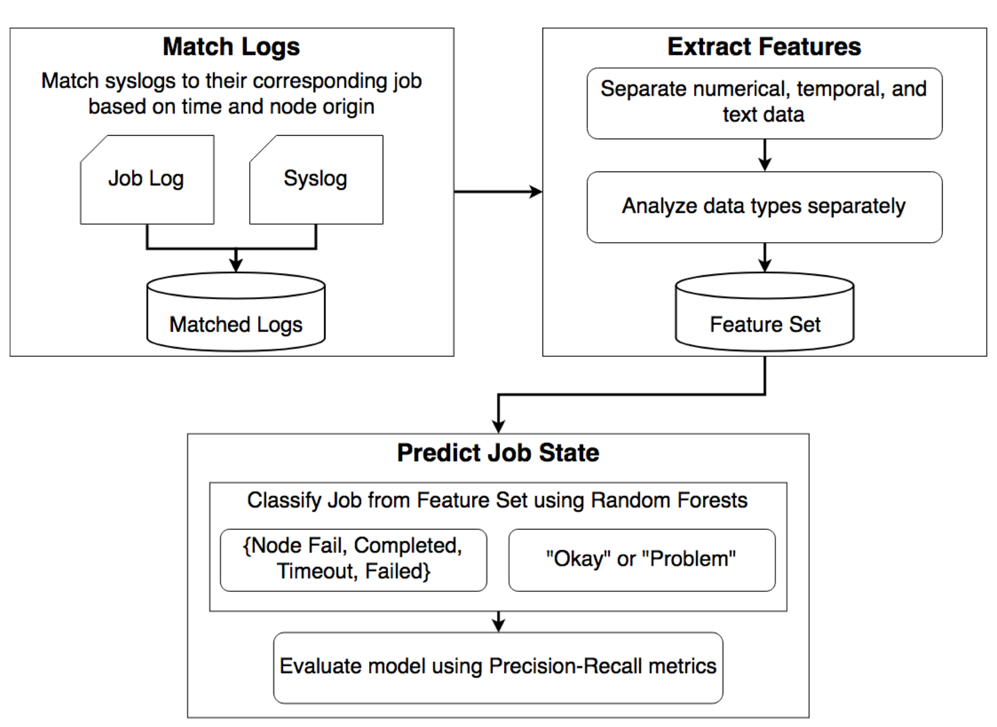
<Datetime> <Node> <Process Tag> <Message>
Mar 26 03:45:02 wf001 TEMP_SENSORS: coretemp +27.0°C

The syslog line format and an example syslog line corresponding to a core temperature check

Approach

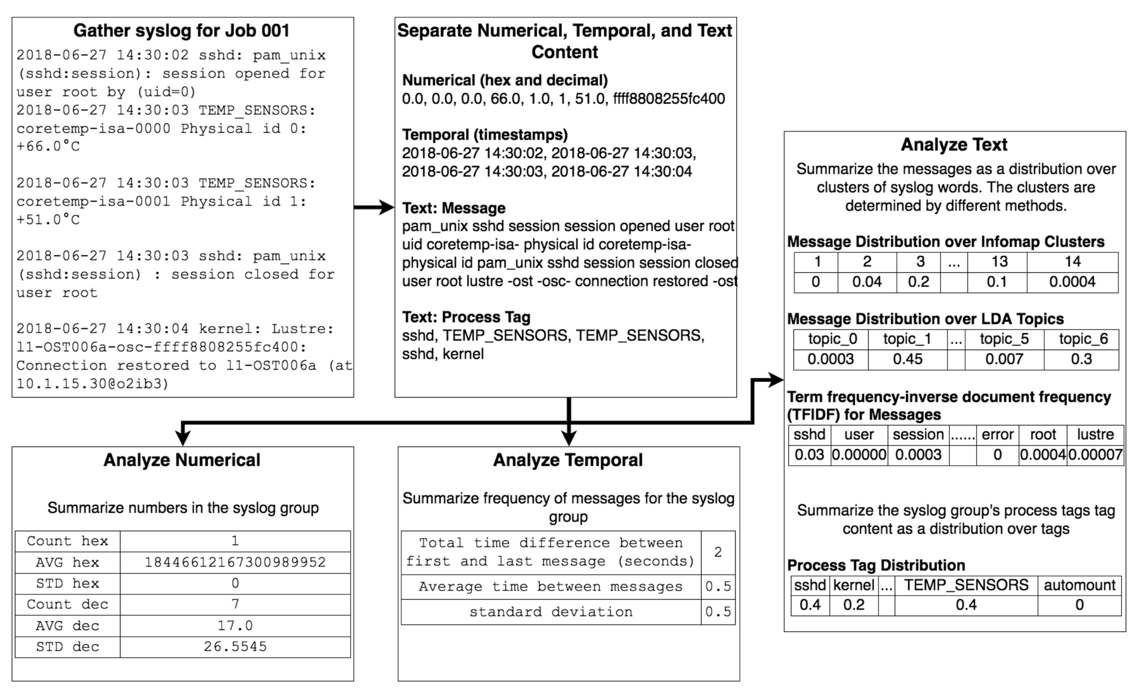
Data Set

A sample of 10,000 jobs from Los Alamos National Laboratory clusters Wolf and Grizzly (5,000 each) over the month of June 2018



Feature Engineering and Extraction

Overview



- For each job we analyze the group of syslogs associated with the job
- Syslogs contain an inhomogeneous mixture of numerical, temporal, and text content
- Separated the numerical, temporal, and text content for individual analysis

Text Content Analysis

- Analyzed the text content using different methods from the fields of systems, natural language processing, and graph analysis
- The text content consists of the syslog messages and the process tags

Infomap[1]
- Graph clustering algorithm
- Each node is a token and each edge is weighted by the number of times tokens appear in a syslog message together
- Distribution of clusters across each syslog group

Term Frequency-Inverse Document Frequency (TFIDF)[2]
- Distribution of terms across the syslog message for all jobs
- Identifies unique words by giving rare words more weight

Latent Dirichlet Allocation (LDA)[3, 4]
- Generative statistical model that finds latent "topics" across documents
- Distribution of topics across each syslog group

Process Tag Distribution
- Distribution of process tags across each syslog group

Standard Keywords (Baseline)
- Counts of commonly searched troubleshooting terms in each syslog group
- `err, warn, fail, time, shutdown, and kill`

Selected Text Clusters

Infomap Clusters

Cluster	Tokens († usernames removed)
1†	user, session, opened, pam_unixsshdsession, closed, root, granted, access, pam_unixsulsession, stam, denied, rtc, pam, account, configur
2	id, c, physical, memory, dimm, event, channel, assertion, sensor, cpu, warn, rank, number, correctable, ecc, d, mmry, b, machine, check, exception, handling, mce, label, unknown, errors, ce, row, amanzi, system, bank, sbridge, nominal, edac, evt, timestamp, clock, high, going, upper, temperature, ctrl, bios, deassertion, oem, critical, synch, noncritical, boot, therm, type, offset, code, log, ac, lostpower, input, extended, list, hardware, mod, direction, name, state, s, conf, rdnc, mem, temp, ssb, qidxmcnp, hermite
3†	read, remov, mountstats, qidxsec, codeexit, binpagosa, exit, metrics, bingen, coderun, bingd_es, bindumpcmp, terminat, symcartesian,, addr, time, socket, processor, misc, area, fatal, vers, process, kill, term, main, signal, apic, tsc, tty, symcylindrical, rpcbind, thru, threshold, requested, restart, via, global_error_check_log, ntpd, serial, w, lnet, lni, fail, lock, page, ib_qib, errno, mesh, generation, date, symmetry, global_error_check_int, cartesian, init_environment, scrubbing, dimension, zonecount, ÿÿÿÿ, resetcleared, communicating, operation, ldlm_enqueue, codemcnp, detected, overflow, cylindrical, dt
5	not, found, map, tainted, includ, sources, key, modulerc, lanldata, device, commodel, busy, about, processes, that, use, some, cases, useful, crestone, umount, toolsrh, tools, return, ask, enabled, svn, turquoiseusrprojects, umount_autofs_indirect, graphics, dotfiles, share

LDA Topics

Topic	Tokens († usernames removed)
0	system lustre not session user ptlrpc root pam_unixsshdsession message tainted trace call disabl procsyskernelhung_task_timeout_secs echo seconds than more blocked task
1	memory dimm event assertion sensor warn channel number rank correctable ecc mmry cpu signal process term kill main system tty
2†	user pam_unixsshdsession session root closed opened segfault lustreerror ldlm_cli_enqueue cookies send port
4	exit qidxsec metrics codeexit binpagosa bingen coderun terminat symcartesian bingd_es bindumpcmp user pam_unixsshdsession root session vers closed
7†	pam_unixsshdsession session root user physical opened closed log hardware event scrubbing access

Experiment and Results

Experimental Setup

- Trained and tested a Random Forest model on all feature sets to predict job outcome (state)
- Results were compared to a baseline of standard keywords
- The model was evaluated on two prediction tasks
  1. Multiclass: classifying a job's state
  2. Okay vs. Problem: classifying a job as "okay" or a "problem"
- Experiment was repeated 200 times using stratified random permutations cross-validation
- Evaluated using Precision-Recall metric

Wolf Results Summary

- All feature sets performed best on the Okay vs. Problem task
- The combined feature sets performed better than alone
- Best performing feature sets across all tasks:
  - Infomap and Temporal & Numerical
  - LDA and Temporal & Numerical
  - Tag and Temporal & Numerical

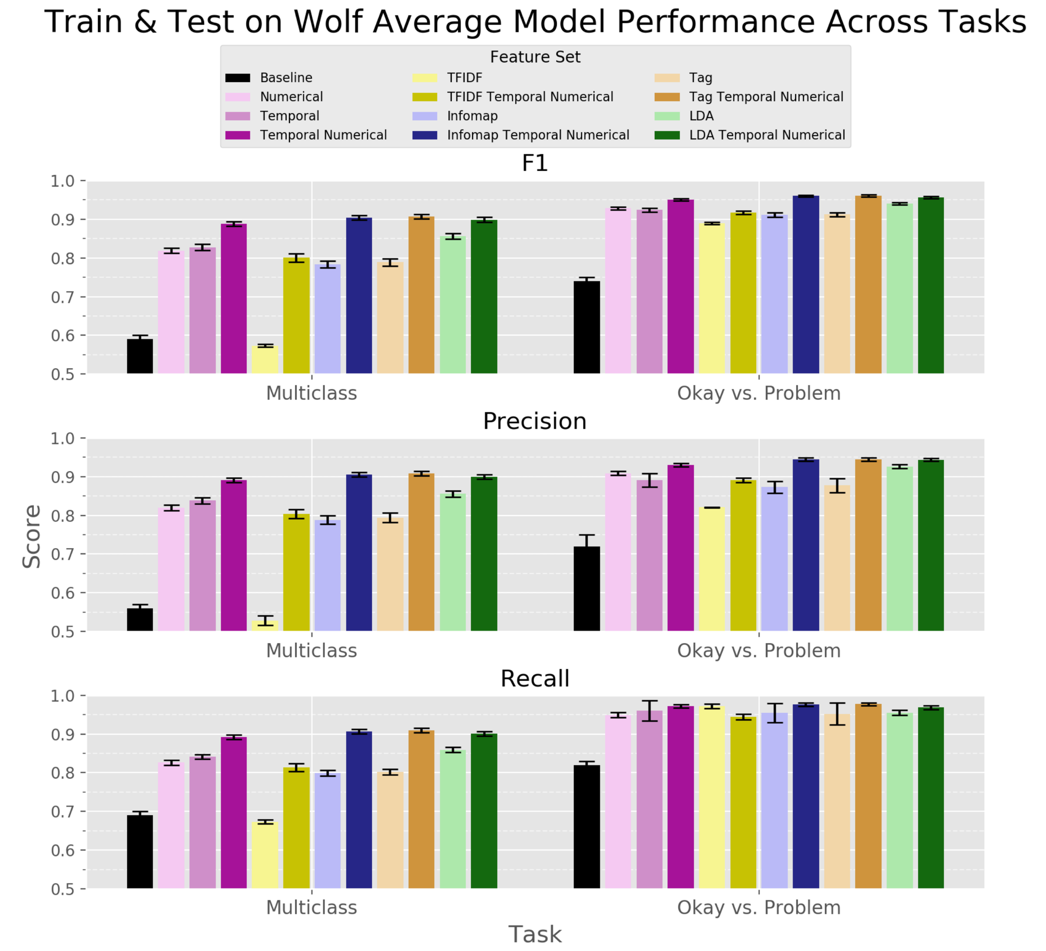
Cross-Platform Experimental Setup and Results

- Setup is the same as for the regular experiment except the random forest model is trained on Wolf and tested on another LANL cluster, Grizzly
- The model did not perform as well as on just Wolf, suggesting that the model performs best when developed for a single cluster

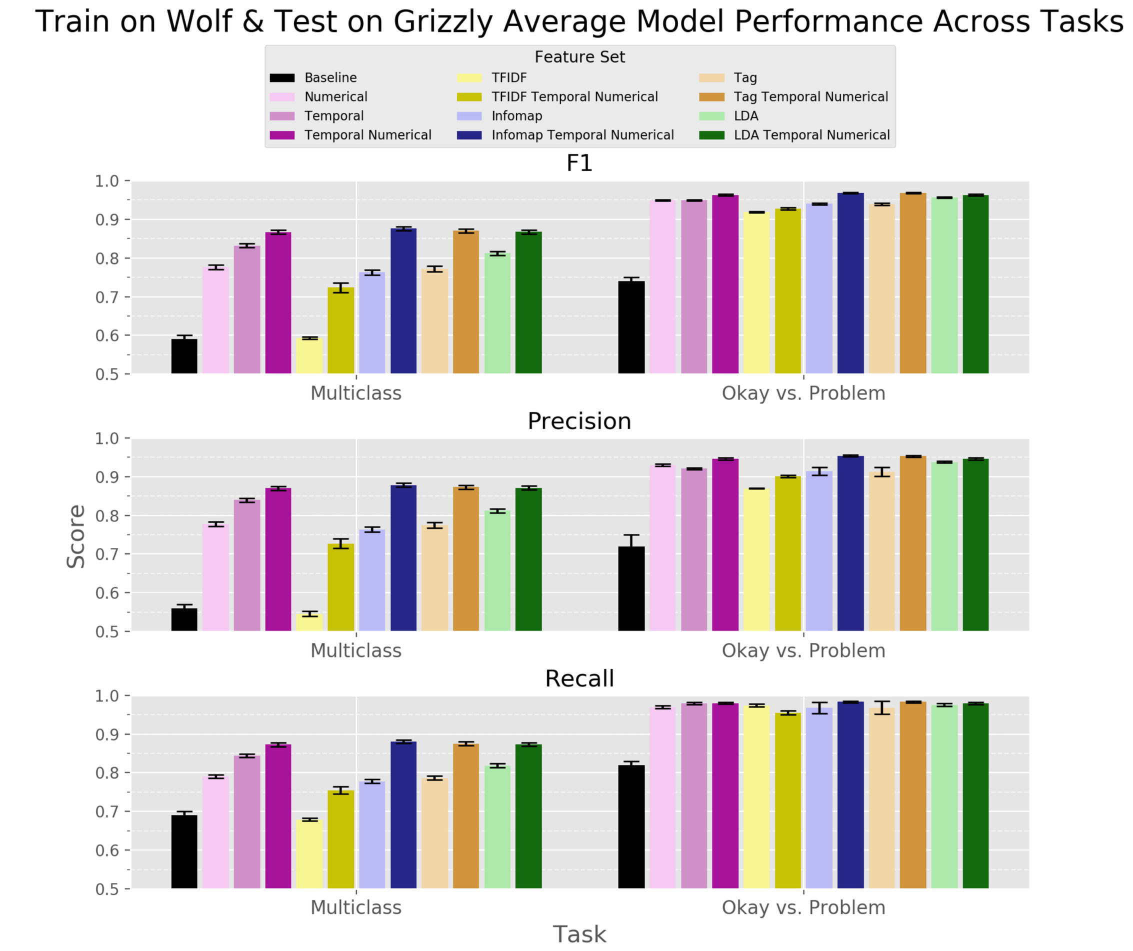
Feature Importance

- Feature importance for the single cluster experiment (Wolf)
- Temporal features were important across all feature sets
- Numerical features, specifically decimal features, were also important

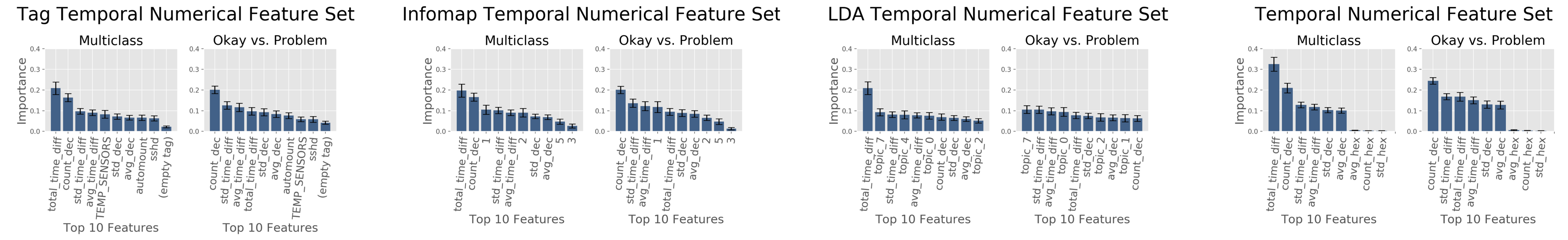
Wolf Average Model Performance Across Tasks



Cross Platform Average Model Performance Across Tasks



Feature Importance for Top Performing Feature Sets



Acknowledgements

We thank Sean Blanchard and Nathan DeBardeleben, both at the Ultrascale Systems Research Center, Los Alamos National Laboratory, for domain expert input to this work. We also thank Hugh Greenberg for providing easy log access.

References

1. D. Edler and M. Rosvall, The MapEquation software package, available online at http://www.mapequation.org.
2. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.
3. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, Mar. 2003.
4. A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

LA-UR-18-31152