Learning and Recognizing Failure Causes on OSG

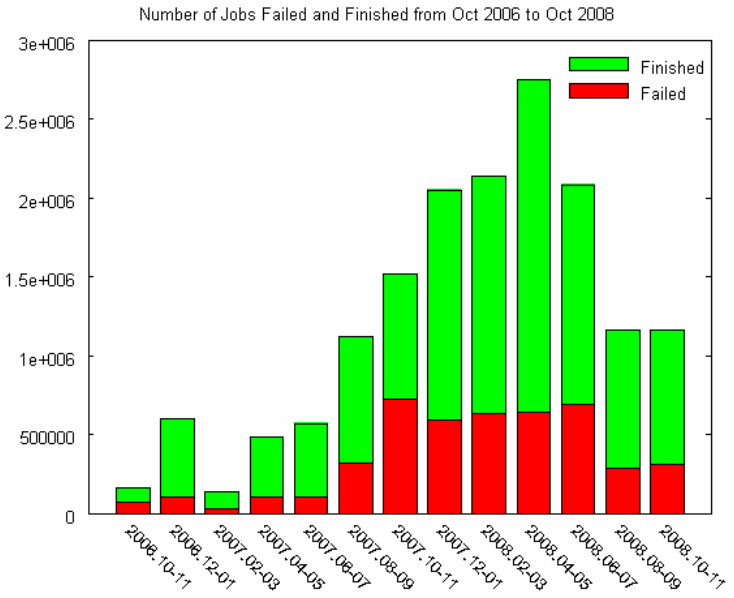
Abstract

From a survey of job failures in US-ATLAS, we found that jobs on OSG constantly fail because of many different reasons, some are known and others are not. System operators troubleshoot these failures mostly by examining various logs and probing all the possible small tests based on their experiences. Obviously, it's a complicated and time-consuming task. What's more, some hidden causes are very difficult to discover if they are beyond system operators' experiences. Therefore, it would be greatly of value to learn and build models on tons of monitoring and logging data, for recognizing failures and discovering hidden causes. A mechanism of Learning and Recognizing Failure Causes (LRFC) is proposed to index system states on the fly, and retrieval similar ones in the future. Some experiment results of partial implementation testing on US-ATLAS monitoring data look promising. In the end, the research plan and resources needed from OSG are listed.

1 US-ATLAS Job Failures and Troubleshooting

In this section, we first look at failure distributions of ATLAS jobs running on OSG, and find out jobs constantly fail because of many different reasons, some of which are known and some are not. Up to my knowledge, troubleshooting and summarizing causes are mostly done manually. Even for Panda system which has been running for several years and having been developed hundreds of error codes, sometimes it still cannot tell the exact problem before system operators examining various logs. Some examples will be given later.

Figure 1displays the total number of jobs failed and finished, and failure rates from October 2006 to October 2008, and Table 1 show the top 10 Error Codes during that time period. It's clear to see that many different types of failures are happening on OSG. And Table 2 in appendix illustrates a portion of different causes for the same failure type (error code) in 24 hours.



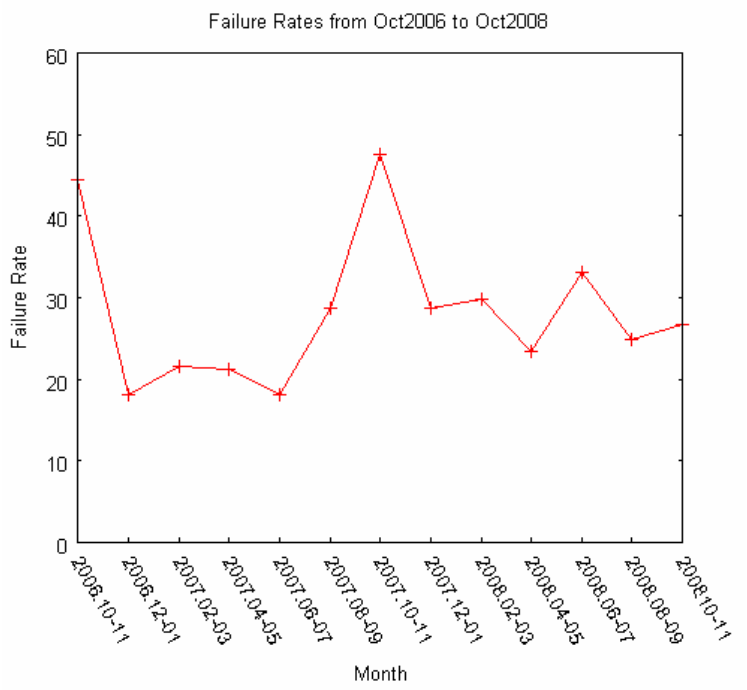


Figure 1. Number of Jobs failed and finished in two years grouped by every two months (upper graph) and failure rate (nether graph). Failure Rate was higher than 20% most of the time.

Error Code	Failed #	Occurrences #	Error Diagnostics
taskBufferErrorCode=100	697032	12	Job expired and killed three days after submission (or killed by user)
jobDispatcherErrorCode=100	474225	13	Lost heartbeat
pilotErrorCode=1099	389740	9	Get error: Staging input file failed
taskBufferErrorCode=102	264196	7	Expired three days after submission
pilotErrorCode=1137	210870	6	Put error: Error in copying the file from job workdir to local SE
transExitCode =40	128293	8	Athena crash - consult log file
transExitCode=86	123745	2	
ddmErrorCode=100	106944	7	DQ2 server error
taskBufferErrorCode=101	99959	4	Transfer timeout (2weeks)
exeErrorCode=99	85831	5	
pilotErrorCode=1150	74793	8	Looping job killed by pilot

Table 1. Top 10 Error Codes in two years. Occurrences # is the times that error code appears in the top10 lists of 13 two-month periods.

From the figures and tables, we could see that a large number of different failures happen on OSG constantly and simultaneously, and error codes are far from enough to identify the causes even they have being developed for years (e.g. Panda). Logs can help a lot, but need too much time and labor for these huge number of repeated arisen failures. Learning and recognizing failure causes from the large data set will make the troubleshooting more efficient, and also can discover internal hidden correlations among system metrics.

2 LRFC Overview

This section will illustrate what the final LRFC might look like, and point out how it helps OSG troubleshooting.

2.1 Definitions

Metric Vector (MV) is a vector of metrics. Each instance of MV is an input of LRFC, and also a training sample for a single model. The last metric in MV is Failure Type (zero if succeeds).

Balanced Accuracy (BA) is defined as the average value of correctly classified rates of all the classes. It is an evaluation criterion for the models [1].

BA = { p(classified as failed | failed) + p(classified as finished | finished) } / 2

Models Set (MS) is a collection of models generated so far. Each model is a Bayesian Network, and the nodes in the network are chosen greedily based on Balanced Accuracy from Metric Vector. The training dataset for each model is inside a Training Window, where majority of the samples should belong to one cause of system failure or one state of success [2].

Signature is a 1/-1/0 sequence having the same length as Metric Vector, indicating the status of each metric - abnormal/normal/irrelevant [3].

Cluster is a clustering of similar Signatures. Ideally, all the signatures in a cluster should have the same cause of failure, or state of success [3].

2.2 Illustration of the Final System I/O

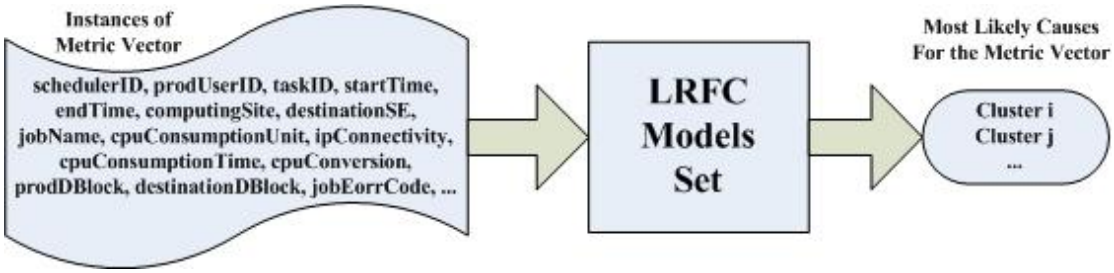


Figure 2. LRFC system I/O

Reading in an instance of Metric Vector, LRFC updates its Models Set and outputs the nearest Clusters of current instance, where each Cluster represents a cause of failure or a state of success. For example: the system returns one cluster which indicates “prodUserID = A” is the main suspect of current “transExitCode = 40” failure.

2.3 How it Helps OSG Troubleshooting

The first time we see a cluster in LRFC, we don’t know what the cause is. But the learned model still tells us current abnormal, normal and irrelevant metrics, which of course give us very useful clues on the possible causes. After system operators confirm the learned cause, associate it with

the output cluster. Then the next time we see the cluster, we know that what the cause could be, and what the metrics should correlate to the problem.

At the same time, the ongoing research will show hidden correlations among collected data, give OSG feedback on specific problems detected, and how to collect data of value in troubleshooting. For example, we found the following rules:

CpuConversion = 0 <=> cpuConsumptionUnit = 'NULL' or 'none + CPU TYPE' (1)

PilotErrorCode = 1099 => cpuConversion = 0 => cpuConsumptionTime = 0 (2)

From them, we can guess that pilotErrorCode = 1099 might have the same reason why we can't get CPU conversion but can get CPU type sometimes. Furthermore, we could show that the first part in cpuConsumptionUnit is always useless.

3 Methodology

The main idea is correlating OSG instrumental data, application monitoring and logging data to different system states (including failures and successes), extracting Signatures from Models Set, and clustering Signatures into several Clusters. When a new data comes in, Models Set updates itself and returns the most possible Clusters of the new data, indicating the states during that time.

3.1 Dataset Splitting based on Failure Type

On OSG, there are many different problems happened at the same time. In order to learn correct correlations for these failures, i.e. get high accuracies on the models, we should split the dataset into different regions representing different types of failures or states of success. One method is roughly categorizing samples based on failure types or error codes, such as "pilot error code = 1099" in Panda and "authentication fail" in site verify scanner.

3.2 Models Set Learning for Each Failure Type Dataset

Learn the most correlated metrics for each failure type dataset in a moving training window; add in the new model to the models set if it has much better accuracy [1-3].

Update Models Set (Dataset, minimum number of samples per class)
For each FT (Failure Type) dataset **do**
    Initialize MS (Models Set) to empty, and TW (Training Window) to empty
    For each new sample **do**
        Add the sample to TW
        If TW has minimum number of samples per class **then**
            Train Bayesian Network Model M on current TW
            Compute Balanced Accuracy on M
            If M has much better BA than other models in MS **then**
                Add M to MS, and set TW to empty
            End-If
        End-If
        Extract Signature (the sample, the most accurate models in MS). See 3.3.

End-For
End-For

3.3 Signature Extracting

Extract Signature for every sample by assigning 1/-1/0 to each metric indicating its status: abnormal/normal/irrelevant [3].

Extract Signature (sample, models)
If metric m_i in the sample is selected in models, and the value of m_i appears much more often when the sample fails than when it succeeds **then**
$S_i = 1$
Else if m_i is selected in models and the value of m_i appears much more often when the sample succeeds than when it fails **then**
$S_i = -1$
Else
$S_i = 0$
End-If

3.4 Signature Clustering and Cluster Retrieval

Cluster all the signatures by K-Means until the clusters are pure enough. Then identify the causes of different clusters with OSG system maintainers, and associate them with each other for future failure diagnosis [3].

For each new sample, update MS, extract Signature of the sample and find out the nearest clusters. Explain the possible causes of current failure with these clusters.

4 Some Experiments on Job Monitoring Data in US-ATLAS

Learning a single Bayesian Network Model on a dataset for one failure type (the train part in 3.2) is implemented. Some experiments are done to see: (1) how good are the Balanced Accuracies on the testing sets; (2) what are the features chosen in these models learning on multiple random time periods; (3) how many failure and success samples are needed to generate stable models or high accuracies.

Fortunately, the results show that the accuracies are higher than 90% most of the time when the numbers of failure samples are larger than 35, and the top chosen metrics are almost the same for the same failure type and different for different failure types. It indicates that each failure type correlates to some constant metrics, which of course reveals the cause of the failure. And for each model, at most two and mostly one out of 26 metrics are selected to be in the Bayesian Network (except the class – failure type), which shows the model captures the key metrics responding for the failure.

Right now the experiments are done for 3 different failure types, each on 12 randomly selected

time periods on 2 months data (August 2008 and February 2008). The three failure types are the top 1, 3, 6 error codes in table 1. Failure samples are the records of jobs failed with the error code, and success samples are records of jobs finished. Each record is a list of selected metrics from Panda Archive Database (listed in table 3 in Appendix). For each randomly chosen data, I enlarged the numbers of failure and success samples incrementally, and built models on the increasing dataset. The results reveal that the number of failure samples is the key to get high BA.

The following subsections display the results of 3 different failure types one by one.

4.1 Job Expired and Killed 3 Days after Submission (or Killed by User)

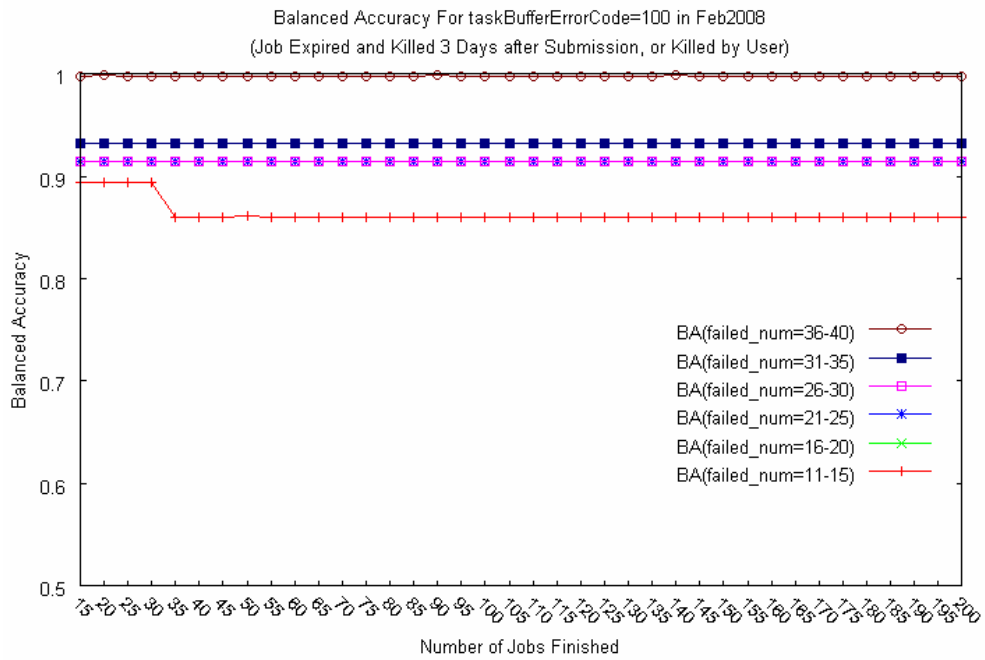


Figure 5. We could see that when the number of failures is larger than 25, BA is higher than 90%. The number of success samples won't influence the accuracy much. So the minimum number of failure and success samples should be 25 and 15.

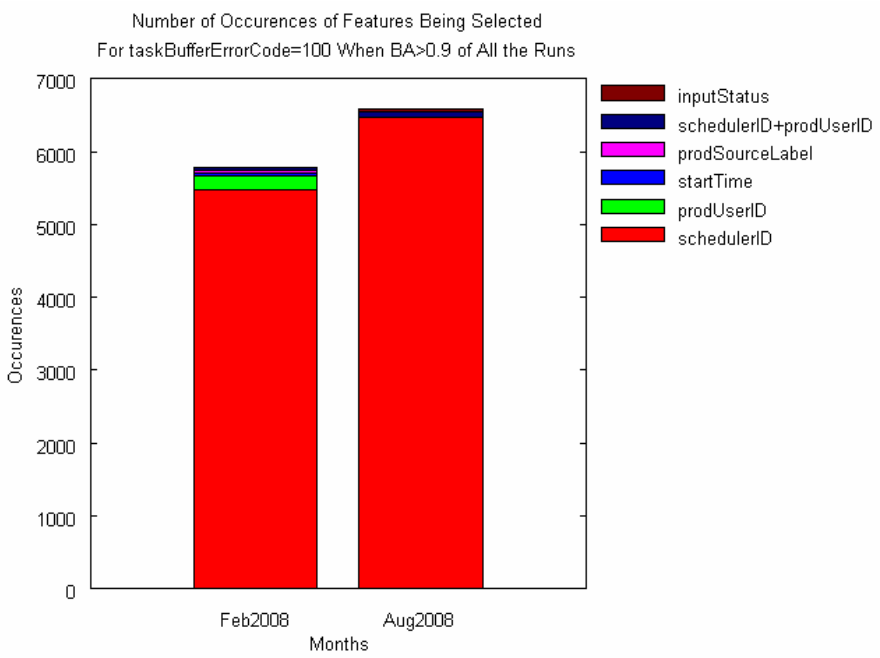


Figure 6. It shows that schedulerID is the most correlated feature for the failure type at that time, i.e. the problem schedulers might be the cause when jobs expired or killed long time after submission.

4.2 Get Error: Staging Input File Failed

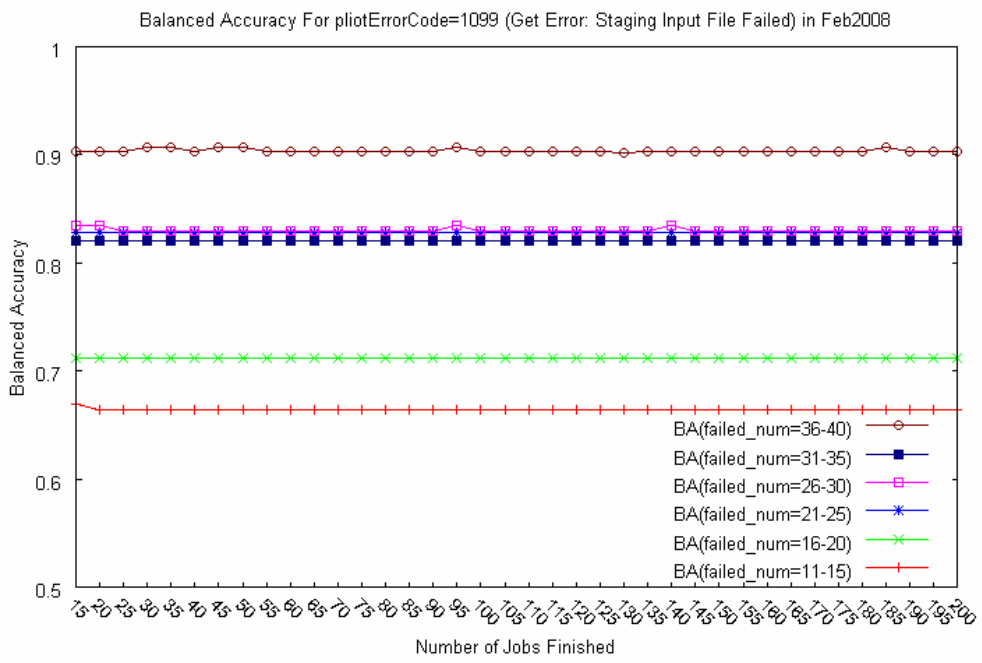


Figure 6. If we want BA to be higher than 90%, the number of failure samples should larger than 35. So the minimum number of failure and success samples should be 35 and 15. But a strange thing is that BA decreases when the number of failure samples increases from 25 to 35. It might because there is more than one cause in the dataset; and a model is trained on one type of cause samples but tested on others. We can prove the suspect in Figure 7.

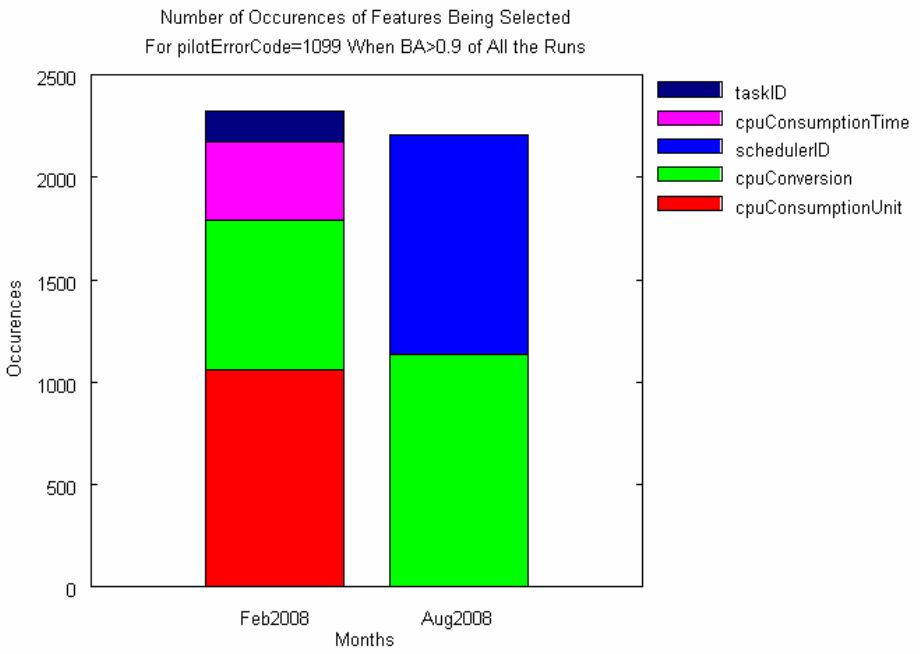


Figure 7. The top chosen metrics are cpuConversion, cpuConsumptionUnit and schedulerID. cpuConversion and cpuConsumptionUnit are duplicate metrics when classifying pilotErrorCode = 1099 (see the end of 2.3). It also shows that the cause of the failure could be the same reason why we can't get CPU conversion but can get CPU type sometimes. Scheduler might be another cause, and even taskID.

4.3 Athena Crash – Consult Log File

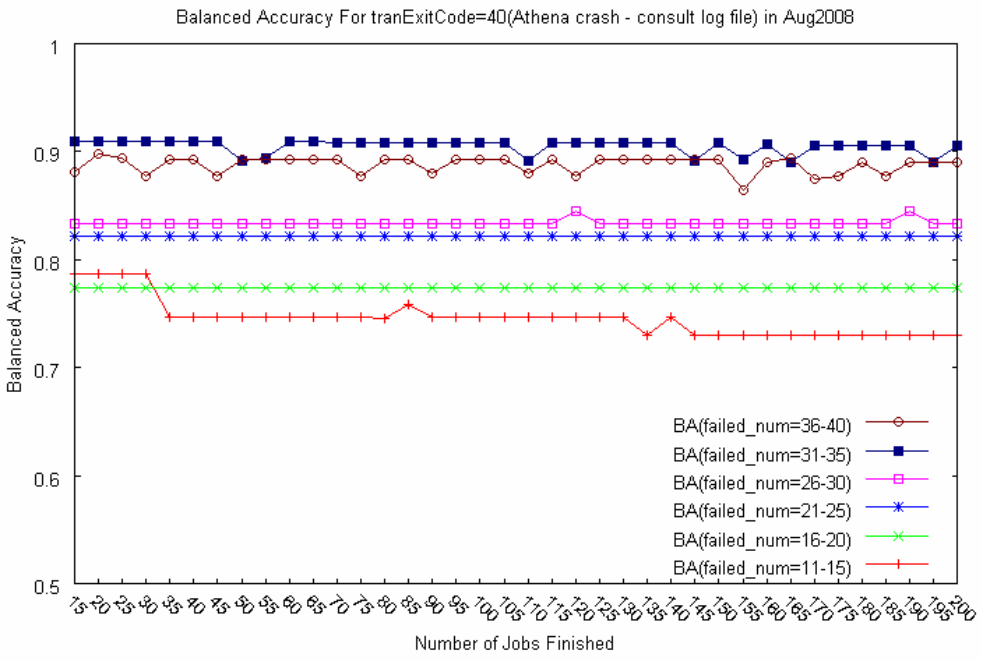


Figure 8. BA decreases when the number of failure samples goes from 35 to 40, which also might because of more than one cause in the samples. It should be interesting to see what happens if we continuously enlarge the number of failure samples.

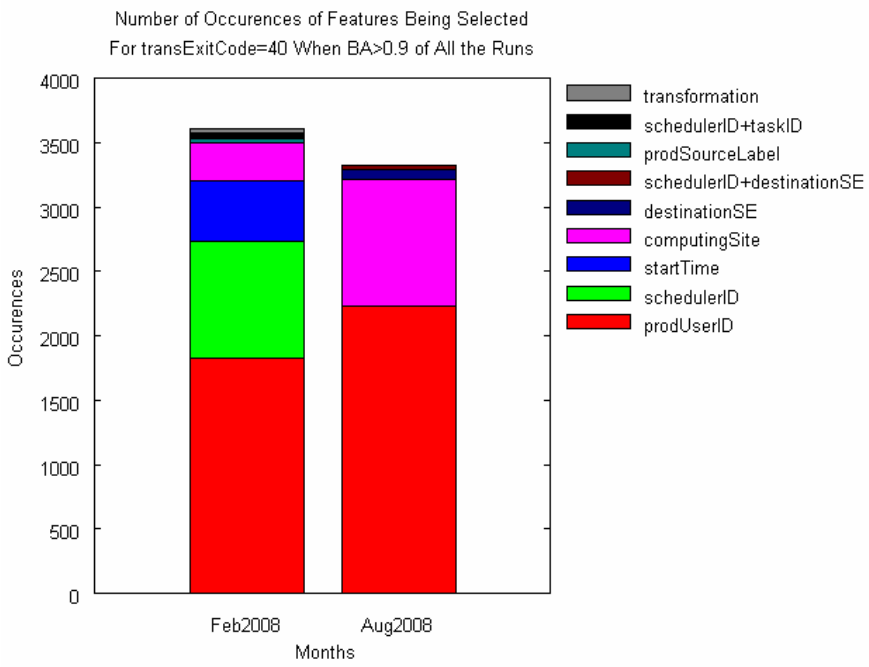


Figure 9. The top metrics are prodUserID, computingSite and schedulerID. User should be a cause for Athena crash; the user might submit a group of jobs using a particular part of Athena core which happened to be broken. ComputingSite and schedulerID might be other causes.

When Models Set is fully implemented, different models will be generated in the training window on the fly. Since most of the failures are temporal correlated, i.e. the same failure is more likely to happen again if it happened just now, we can guess that different causes are likely to be separated in different training windows, and then be learned separately. In that case, I consider BA will be even higher than the experiments. And even if different causes of the same failure type appear in the same training window, the model won't be added into Models Set if BA is not high enough; but old models having high BA will be selected for extracting Signature.

There are only 26 metrics in the training set, no logs info at all. If we include more metrics and extract useful metrics from logs, the causes learned should be much more detailed.

5 Research Plan and Resources Needed from OSG

5.1 Reaching an Agreement on Problems Set and Data Set

I have talked with some OSG system operators on the problems and difficulties at present, and list them as follows:

- Debug the same problems from scratch again, again and again, on different sites and time, for different jobs and users;
- Can't find the causes of some problems even after examining all the logs and probing all the possible tests, and have no clue on the causes, then mark "unknown";
- Wait for tickets replies because of not having the rights to get remote information needed.

We would like to discuss more about these difficulties in detail. Then define the specific problems set we should first work on and its corresponding datasets, also the ways to verify and use these learning results.

5.2 More, Qualified, Long-Term Data

Only good data can bring interesting and useful results. Thus it's great to navigate all the qualified data or potential ones on OSG. If some specific data are very critical in the learning but not there, then we might need to plug in sensors on OSG. Databases are preferred.

I know some monitoring or troubleshooting systems. For instance, Panda is a good one, although some values in the database are not filled up by system administers. Grid site verify scanner is not so regular (every ~6 hours). RSV, ReSS, and DDM Dashboard are great, but I am not sure how to get the data from their historical datasets. I also know a little about PhEDEx, CDF, CMS, etc., but haven't done anything related yet.

5.3 Help from OSG Experts and System Operators

- Explain the meaning of data in the datasets, and the methods of collecting them;
- Identify whether the results can indeed explain the problems, or why not;
- Provide some representative problems on OSG so far.

References

[1] Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In Proc. 6th USENIX OSDI, 2004.

[2] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In DSN, 2005.

[3] Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system history. In Proc. 20th ACM SOSP, 2005.

[4] Chengdu Huang, Ira Cohen, Julie Symons, Tarek Abdelzaher, Achieving Scalable Automated Diagnosis of Distributed Systems Performance Problems, HP Lab Tech Paper 2007

[5] Remco R. Bouckaert, Bayesian Network Classifier in Weka, July 2008

[6] http://www.opensciencegrid.org/

Appendix

Table 2. Some of the jobs failed with pilotErrorCode=1137 (Put error: Error in copying the file from job workdir to local SE) on 12/19/2008, collected from "Panda Based Distributed Analysis Dashboard - Analysis job error report, last 24 hours"

PandaID	Job name	Log
21626105	valid1.105015.J6_pythia_jetjet.recon.e344_s479_d147_r599_tid030068._00018.job #4	**pilot: Job failed: Non-zero failed job return code: 6|Log put error: Copy command returned error code 256** and output: /usr/local/grid/glite/3.1.19-0/WN32/lcg/bin/lcg-cr lcg_util-1.6.15 GFAL-client-1.10.17 [SE][**GetSpaceTokens**] httpg://ccsrm.in2p3.fr:8443/srm exe: **TRF_SEGFAULT** | Caught signal 11(Segmentation fault). Details: | (pid=27511 ppid=27510) received fatal signal 11 (Segmentation fault) | ./csc_recoESD_runathena: line 1: 27511 Segmentation fault (core dumped) /afs/in2p3.fr/group/atlas/sw/prod/rele
21625962	pathena jobID=193 buildJob	**pilot: Log put error: Copy command returned error code 256** and output: /usr/local/grid/glite/3.1.19-0/WN32/lcg/bin/lcg-cr lcg_util-1.6.15 GFAL-client-1.10.17 [SE][**StatusOfPutRequest**] httpg://ccsrm.in2p3.fr:8443/srm/managerv2: CGSI-gSOAP: **Could not open connection!**
21624563	valid1.105015.J6_pythia_jetjet.recon.e344_s479_d147_r599_tid030068._00007.job #5	**pilot: Job failed: Non-zero failed job return code: 99|Log put error: Copy command returned error code 256** and output: /usr/local/grid/glite/3.1.19-0/WN32/lcg/bin/lcg-cr lcg_util-1.6.15 GFAL-client-1.10.17 [SE][**GetSpaceTokens**] httpg://ccsrm.in2p3.fr:8443/sr **exe: TRF_UNKNOWN** | std exception thrown from algorithm (basic error)St9bad_alloc | std exception thrown from algorithms (basic error)St9bad_alloc
21623664	pathena jobID=1 buildJob	**pilot: Log put error: Copy command returned error code 256** and output: /usr/local/grid/glite/3.1.19-0/WN32/lcg/bin/lcg-cr lcg_util-1.6.15 GFAL-client-1.10.17 0 bytes 0.00 KB/sec avg 0.00 KB/sec inst 20895 bytes 38.75 KB/se - Errors from athena_stdout.txt - WARNING: **Certificate verification** error for gridui05.usatlas.bnl.gov: self signed certificate ERROR: ld.so: object '/usr/local/products/xrootd/prod/lib/libXrdPosixPreload.so' from **LD_PRELOAD** cannot be preloaded: ignored.
21623269	data08_cosmag.00091900.physics_RPCwBeam.recon.o4_r600_tid030237._03350.job #5	**pilot: Job failed: Non-zero failed job return code: 34|Log put error: Copy command returned error code 256** and output: /usr/local/grid/glite/3.1.19-0/WN32/lcg/bin/lcg-cr lcg_util-1.6.15 GFAL-client-1.10.17 [SE][**GetSpaceTokens**] httpg://ccsrm.in2p3.fr:8443/sr **exe: TRF_EXC** | RuntimeError: ('no tree [POOLContainer_EventInfo_.]',)

21622640	pathena jobID=9750 runAthena	**pilot: Job failed: Non-zero failed job return code: 141|Log put error: Copy command returned error code 256** and output: /usr/local/grid/glite/3.1.19-0/WN32/lcg/bin/lcg-cr lcg_util-1.6.15 GFAL-client-1.10.17 [SE][GetSpaceTokens] httpg://ccsrm.in2p3.fr:8443/s **trans: No input file available** - check availability of input dataset at site
…	…	…

Table 3 Metrics Selected as Training Data from Panda Archive Database

Metric Name	Notes
schedulerID	ID created by scheduler
prodUserID	ID of the user defined the job (user's certificateDN or e-mail address)
taskID	task ID
creationTime	generated by MySQL function (client side)
modificationTime	generated by MySQL function (client)
startTime	will be filled after job finishes
endTime	will be filled after job finishes
stateChangeTime	timestamp of the last state change
modificationHost	can be extracted from MySQL function (client)
computingSite	example: BU_ATLAS_Tier2
transformation	example: share/rome.1001.reco.MuonDigit.OverrideRE.trf
homepackage	example: JobTransforms-10.0.1.5
prodSourceLabel	possible values: "managed, regional, user, panda, test"
cpuConsumptionUnit	will be filled after job finishes
ipConnectivity	comes from CERN ProdDB for managed production jobs
attemptNr	number of job-attempt, ("0" in jobsDefined table for new jobs,"1" when job is moved to jobsActive table)
cpuConsumptionTime	will be filled after job finishes
assignedPriority	comes from CERN ProdDB for managed production jobs (assigned by production manager)
currentPriority	indexed, assigned by production manager, usually the same as assignedPriority
cpuConversion	CPU conversion factor
relocationFlag	flag for submitting special jobs (M5reco) to a given cluster
destinationDBlock	name of job's destination datablock; is used to register the outputs of an associated set of jobs as belonging to one block to be saved at an archival destination, index
cloud	Cloud (associated with Tier 1) job is submitted to (US,Canada,etc.)
inputSize	The total size of input files, computed from filesTable, in bytes
inputStatus	The status of input files; whether job's inputs are at computingSite, job's output is available in destination datablock, index
destinationSE	destination storage element (archival destination) of job's output file(s)
errorCode	Combination of 7 error codes (transExitCode, pilotErrorCode, exeErrorCode, ddmErrorCode, brokerageErrorCode, jobDispatcherErrorCode, taskBufferErrorCode)