Machine Learning Based Log Analysis of Computing Systems

By:

Anwesha Das

North Carolina State University

Introduction

- System Logs contain voluminous text messages whose semantic meaning can be leveraged
- Logs help in debugging, understanding performance behavior and localize faults in the systems
- Past work has leveraged PCA (Principal component analysis), automaton/modeling, clustering, flow reconstruction principle based approaches – few intrusive, few identifier based (e.g. Stitch)
- Aim – Post-mortem analysis of system logs without source code information (non-intrusive)
- Targets text based statistical machine learning techniques such as topic modeling.
- Demonstrate insights of both HPC systems and Distributed Systems.

Data Details

Serial No.	System	#Lines	#Files	System/Bug-Details
1.	Hadoop	28264	10	1911 Bug – 1 console and rest system logs
2.	Cassandra	892	1	11050 Bug – System Log
3.	Cassandra	73	1	5064 Bug – System Log
4.	Titan	655419	1	Integrated RAS log
5.	BlueGene-Intrepid	12500	1	Integrated RAS log
6.	Cray XC	12466	1	Console Log

Log Properties

- HPC logs are **denser** containing more number of columns for every line of the document compared to Distributed System (DS) Logs.
- Timestamps have granularity in seconds in all but **overall duration** is higher in HPC logs than DS. Selected events pertaining to anomalies are logged in console logs of DS, but HPC systems **encompass more components** since integrated RAS log is used.
- Log labels such as **Warn**, **Info**, **Fatal** is present in DS and BGL logs compared to Cray systems which have a few Critical Tags apart from "Error" log level.
- Source component based information rather than specific source code based file name (DS) displayed in HPC systems.

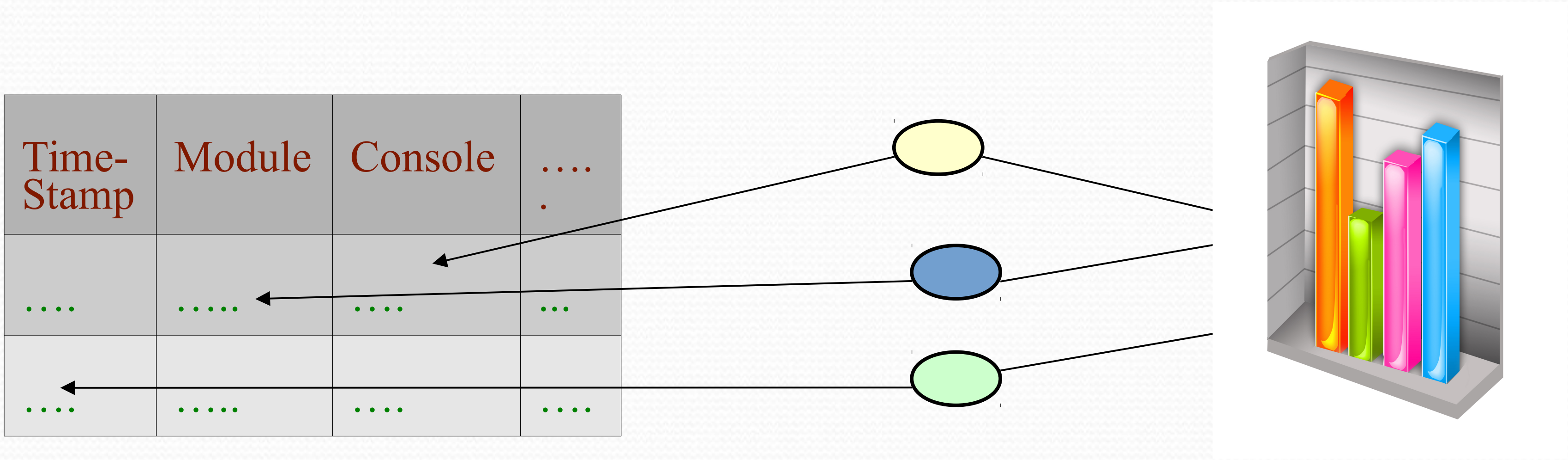
HPC – File System/Network Interconnect/Compute Node ID, e.g. Lustre, Lnet, HWERR, c0-0c0s1n1
DS – File Name, line number, Module name, e.g. Memtable.java, Line 470, Flushwriter

M/L Experiments

- Leveraged LDA (Latent Dirichlet Allocation) to perform topic modeling on text based logs. Estimates likelihood of texts in the overall data over a sampling rate. LDA is based on Gibbs Sampling.
- Used Mallet (Machine Learning for Language Toolkit) to analyze and perform experiments.
- Topic modeling, is suitable for unlabeled data required for unsupervised learning unlike classification etc.

Serial No.	**System**	**#Unit**	**Typical Failures**
1.	Hadoop/Cassandra	Storage service, JVM	No node available, Could not obtain block id, Exiting due to deserialization error, exception, server is degraded mode
2.	BlueGene (Intrepid)	Kernel, h/w, s/w	GPU errors, Machine check exception, Node Voltage Error, BGP chipkill error
3.	Titan/Cray	Alps, n/w, pciReport	NMI Fault, out of memory error, PCI Bus Error, LCB failure, corretable_dim_errors

LDA (Latent Dirichlet Allocation)

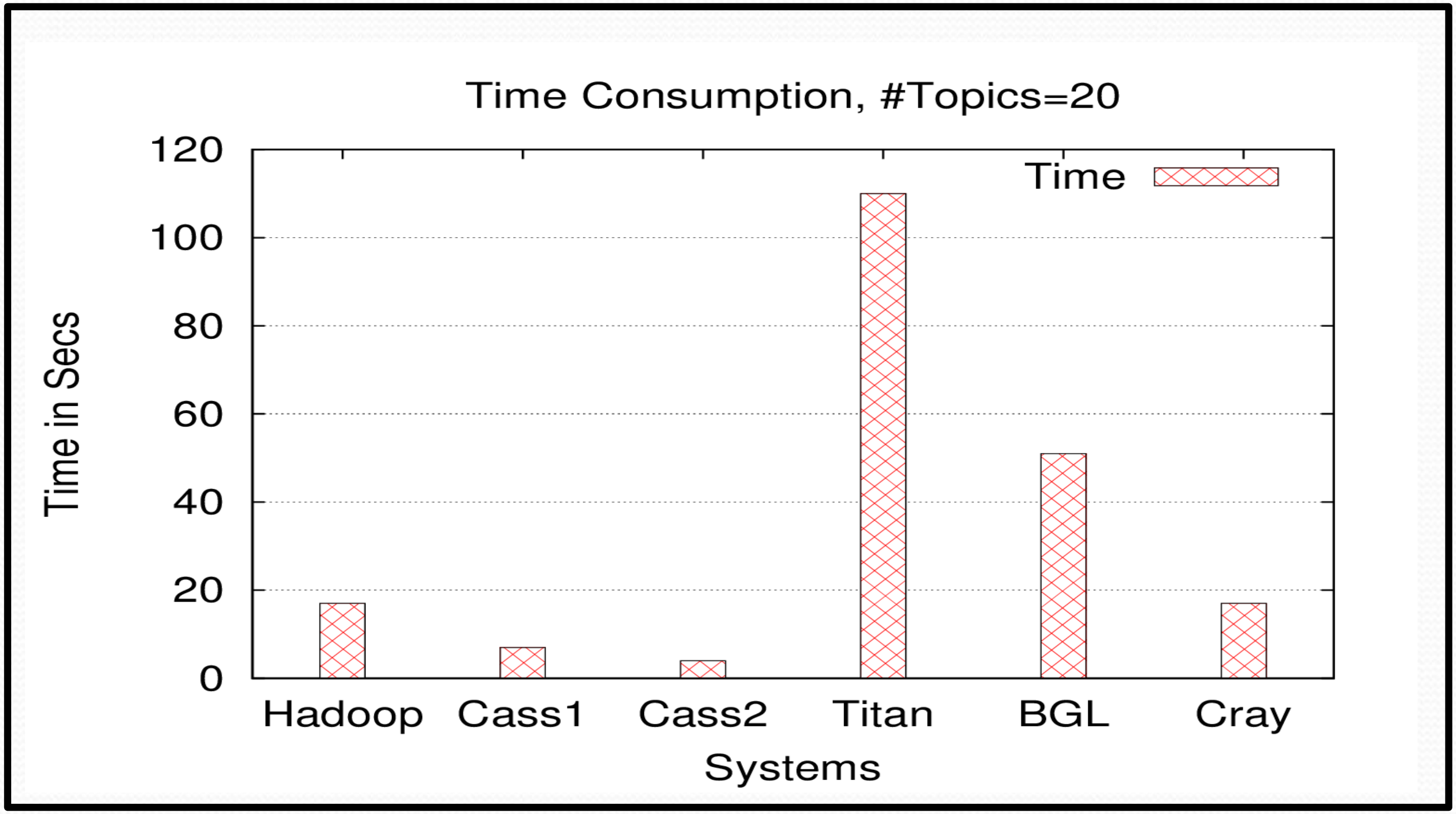


Time-Stamp	Module	Console	…..
….	…..	….	…
….	…..	….	….

Temporal LDA – Time-stamp based ordering, new topics evolve over time, we have temporally sequenced events, figure out if it can help with causal relationships to help predict future failures.

Spatial LDA – Cluster spatial structure together but spatial distribution needs to be taken into account, thoughts of preprocessing for data integration from raw logs needs more investigation. Could be used for correlating spatial and temporal properties of logs in Cray systems !!

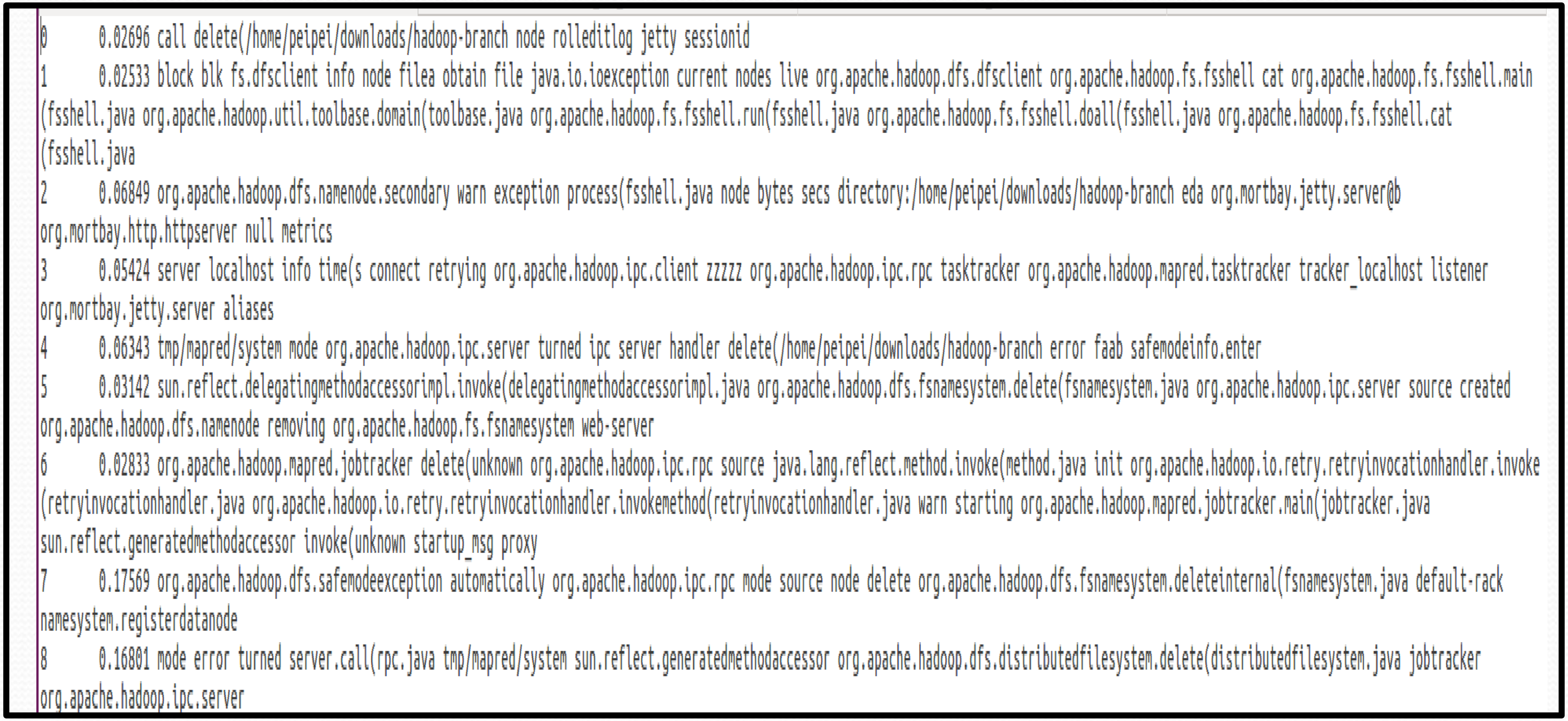
Time Complexity



Titan takes highest time even after limiting with fixed number of iterations due to largest log size.

Even if Hadoop logs have many lines, the text information in the lines are smaller and sparse, so takes less time to converge.

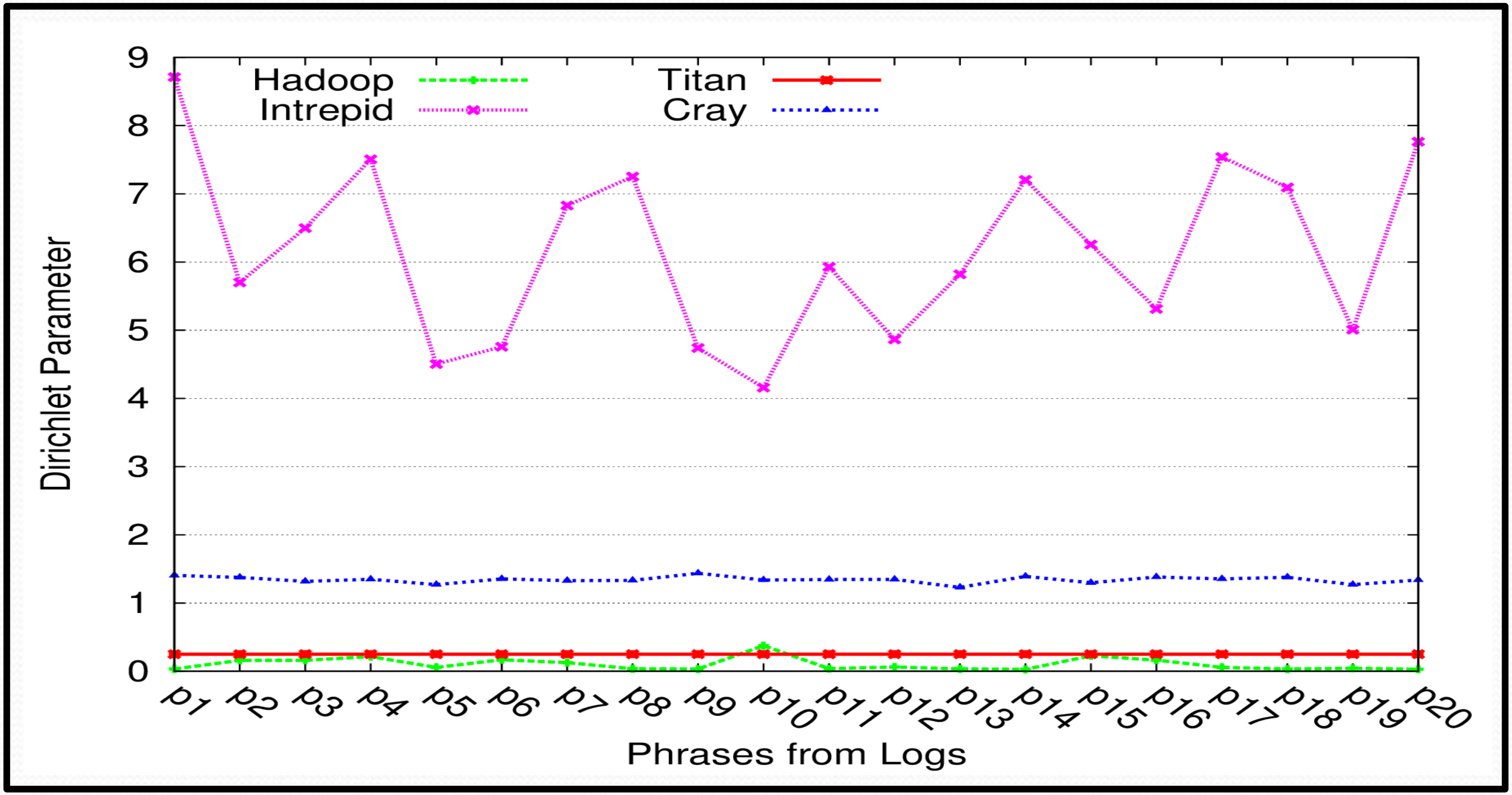
Time Complexity



```
0	0.02696 call delete(/home/peipei/downloads/hadoop-branch node rolleditlog jetty sessionid
1	0.02533 block blk fs.dfsclient info node filea obtain file java.io.ioexception current nodes live org.apache.hadoop.dfs.dfsclient org.apache.hadoop.fs.fsshell cat org.apache.hadoop.fs.fsshell.main
(fsshell.java org.apache.hadoop.util.toolbase.domain(toolbase.java org.apache.hadoop.fs.fsshell.run(fsshell.java org.apache.hadoop.fs.fsshell.doall(fsshell.java org.apache.hadoop.fs.fsshell.cat
(fsshell.java
2	0.06849 org.apache.hadoop.dfs.namenode.secondary warn exception process(fsshell.java node bytes secs directory:/home/peipei/downloads/hadoop-branch eda org.mortbay.jetty.server@b
org.mortbay.http.httpserver null metrics
3	0.05424 server localhost info time(s connect retrying org.apache.hadoop.ipc.client zzzzz org.apache.hadoop.ipc.rpc tasktracker org.apache.hadoop.mapred.tasktracker tracker_localhost listener
org.mortbay.jetty.server aliases
4	0.06343 tmp/mapred/system mode org.apache.hadoop.ipc.server turned ipc server handler delete(/home/peipei/downloads/hadoop-branch error faab safemodeinfo.enter
5	0.03142 sun.reflect.delegatingmethodaccessorimpl.invoke(delegatingmethodaccessorimpl.java org.apache.hadoop.dfs.fsnamesystem.delete(fsnamesystem.java org.apache.hadoop.ipc.server source created
org.apache.hadoop.dfs.namenode removing org.apache.hadoop.fs.fsnamesystem web-server
6	0.02833 org.apache.hadoop.mapred.jobtracker delete(unknown org.apache.hadoop.ipc.rpc source java.lang.reflect.method.invoke(method.java init org.apache.hadoop.io.retry.retryinvocationhandler.invoke
(retryinvocationhandler.java org.apache.hadoop.io.retry.retryinvocationhandler.invokemethod(retryinvocationhandler.java warn starting org.apache.hadoop.mapred.jobtracker.main(jobtracker.java
sun.reflect.generatedmethodaccessor invoke(unknown startup_msg proxy
7	0.17569 org.apache.hadoop.dfs.safemodeexception automatically org.apache.hadoop.ipc.rpc mode source node delete org.apache.hadoop.dfs.fsnamesystem.deleteinternal(fsnamesystem.java default-rack
namesystem.registerdatanode
8	0.16801 mode error turned server.call(rpc.java tmp/mapred/system sun.reflect.generatedmethodaccessor org.apache.hadoop.dfs.distributedfilesystem.delete(distributedfilesystem.java jobtracker
org.apache.hadoop.ipc.server
```

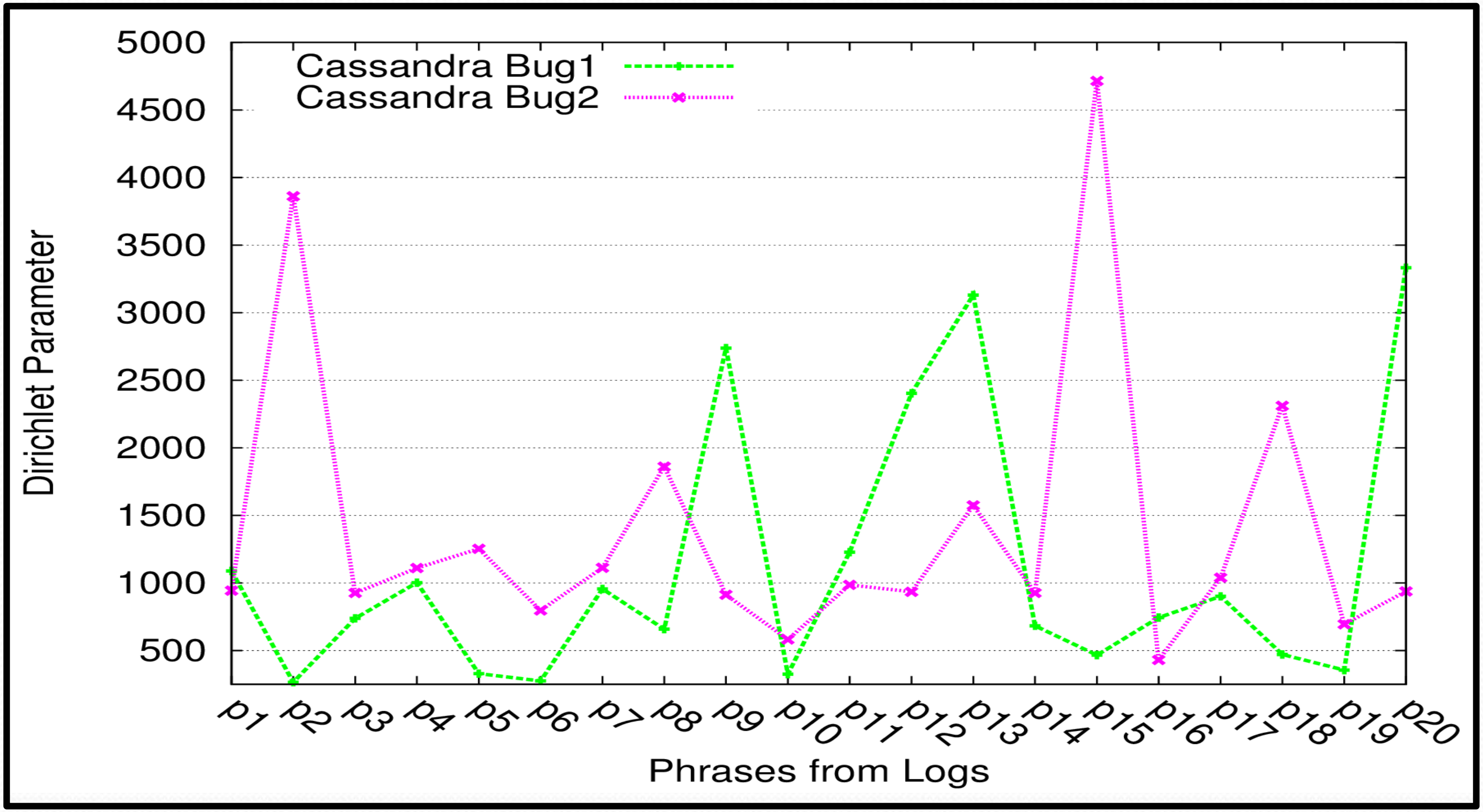
Example of Topics extracted from 28000 lines across 10 files of Hadoop logs.

Dirichlet Parameter Variation



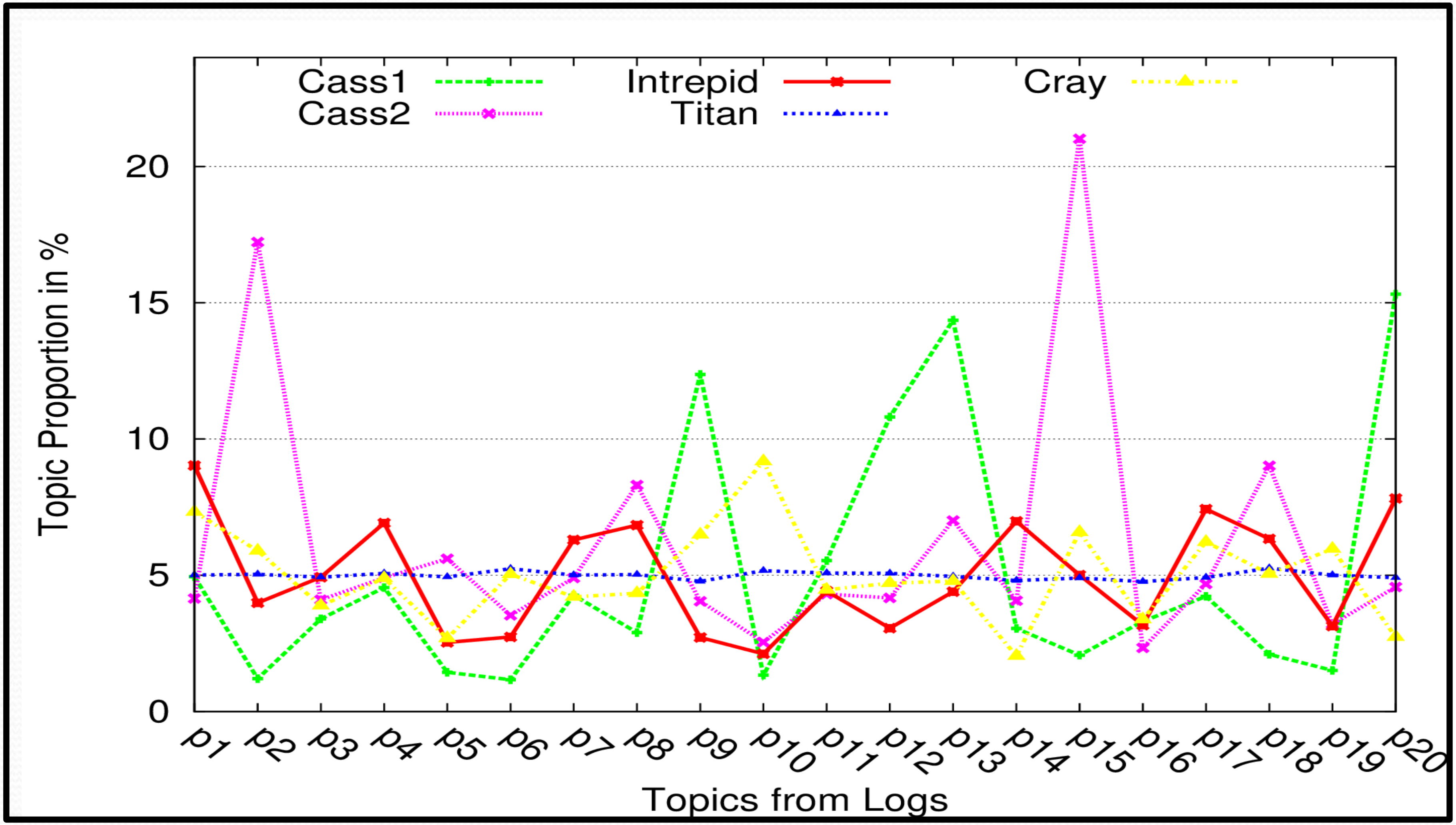
Among these 4 systems, the overall variation is not much, Intrepid has minor variations and Titan is same since the required parameter was not used, to reduce the convergence time.

Dirichlet Parameter Variation



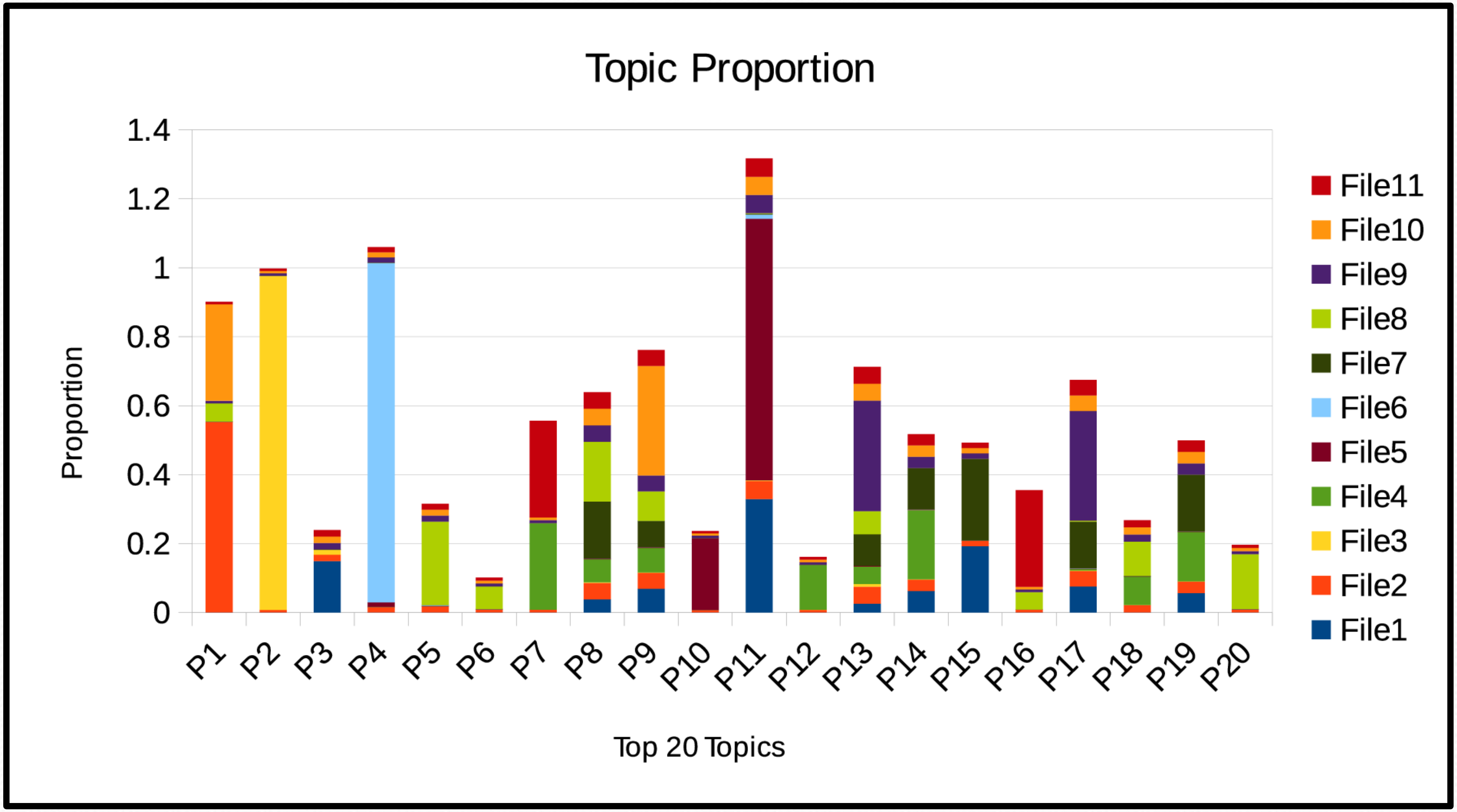
Cassandra has larger variations in the top 20 phrases sampled across the console log. Also, the overall values are high indicating similarity in the mixture of overall topics in the document.

Topic Proportion



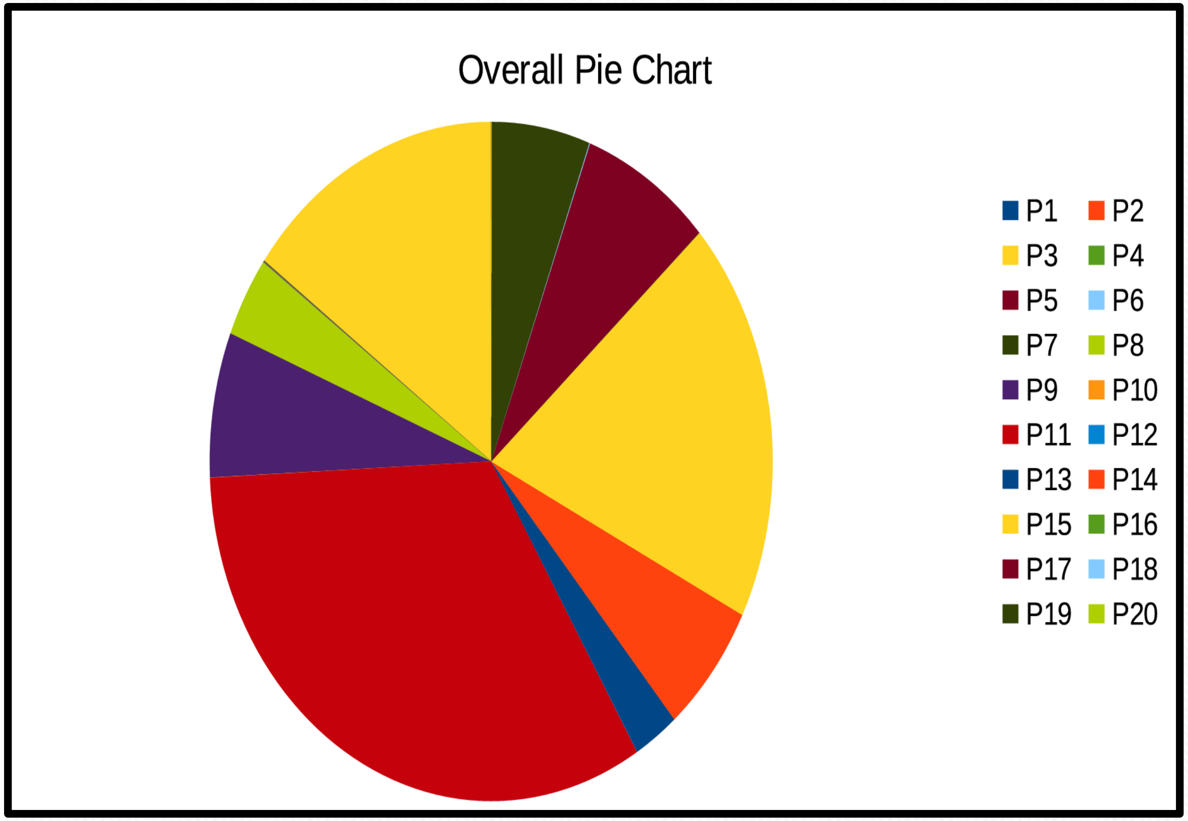
Conforms to Dirichlet parameter variation, some correlation exists, Cassandra has disparate proportions of topics, Titan has very similar phrases in all lines.

Topic Proportion



Overall proportion of a Topic in all the files of Hadoop: P11 is mostly in File1 & 5. P13 is largely in File9. P1 is concentrated in File2. Thus single versus multiple files changes the ratio which can help us track suitable bug locations.

Topic Proportion



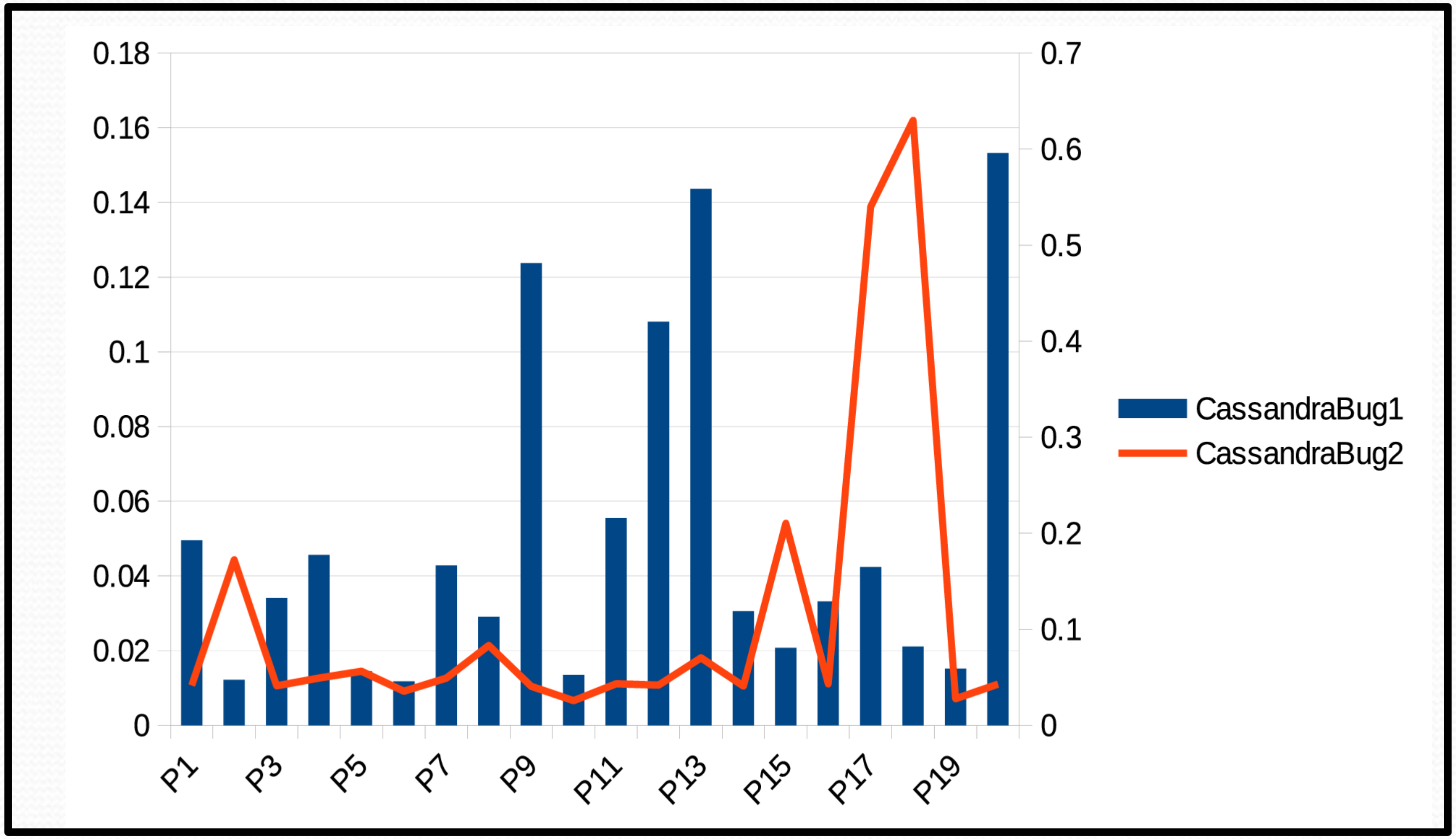
1. Certain Information dominant across multiple files.

2. Biased towards text messages which are trivial but repetitive such as java exceptions and startup/shutdown messages.

3. Few rare indicative messages may not be captured because of frequency bias.

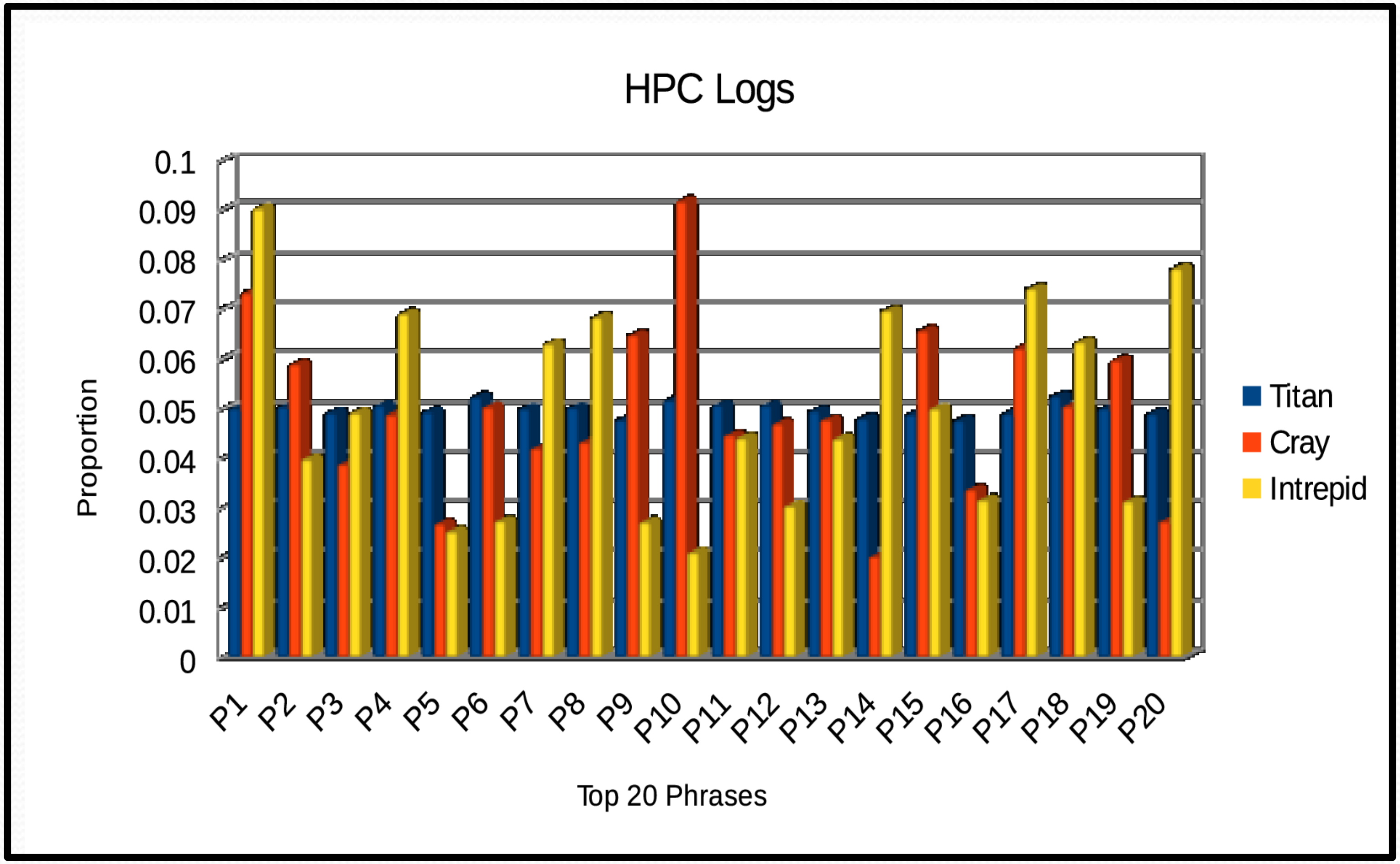
P11 has highest proportion, it corresponds to "container startup_msg starting shutdown_msg localhost shutting args...", P5: "ipc server handler delete", P15: "Thread exception", Relevant Messages indicative of failures.

Topic Proportion



Bug2, P18: heapbytebuffer, storageservice related messages, migration manager, Bug1, P20: ColumnDefinition, storage service, memtable flush, P12,13: Related to database descriptor, write-request timeout etc. If same system, based on bug specifics, similar phrase concentration is possible.

Topic Proportion



Unlike Hadoop and Cassandra Logs, HPC logs have similar overall distribution of Phrases in the document. In this case, the input files had less faults or diversity. In DS logs the input documents were not well per-processed.

Key Takeaways

- Careful use of topic modeling can help localize buggy messages in system logs.
- Unlike system logs, console messages alone may not contain lower level system information helpful in diagnosing anomalies in the system. Typically in Hadoop certain bugs have messages in the console logs, but we should system logs as well.
- How time-series based topic evolution happens over an interval has not been considered here since, small amount of logs have been used with no significant variation over time. These topic change as logs have pattern variations over a longer time-frame.
- The message distribution over different files can help locate the failure source such as *namenode* versus *tasktracker* but that depends on the data integration and pre-processing procedure used before running the experiments.

Conclusions

- Both HPC and DS logs have sufficient event information which can be leveraged using NLP, M/L techniques for better fault tolerance in such systems.

- The similarity of phrase mixture is more in HPC logs than DS logs since the core system logs were not sufficiently used in the experiments unlike Titan logs which were integrated across multiple files with relevant information. This makes anomaly detection work harder in raw HPC logs.

- Fault localization is possible with careful use of time-series based NLP and statistical estimation techniques and investigate new traits/patterns with newer systems and logging mechanisms.

- How to per-process and integrate is a crucial task to have efficacy in topic modeling variants in general since that this impacts the overall phrase estimates in the input documents.

Thank You