Identifying Impactful Service System Problems via Log Analysis

Shilin He[1,2], Qingwei Lin[3], Jian-Guang Lou[3], Hongyu Zhang[4],
Michael R. Lyu[1,2], Dongmei Zhang[3]

[1]The Chinese University of Hong Kong, Hong Kong, China
[2]Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
[3]Microsoft Research, Beijing, China
[4]The University of Newcastle, Australia

Motivation & Background

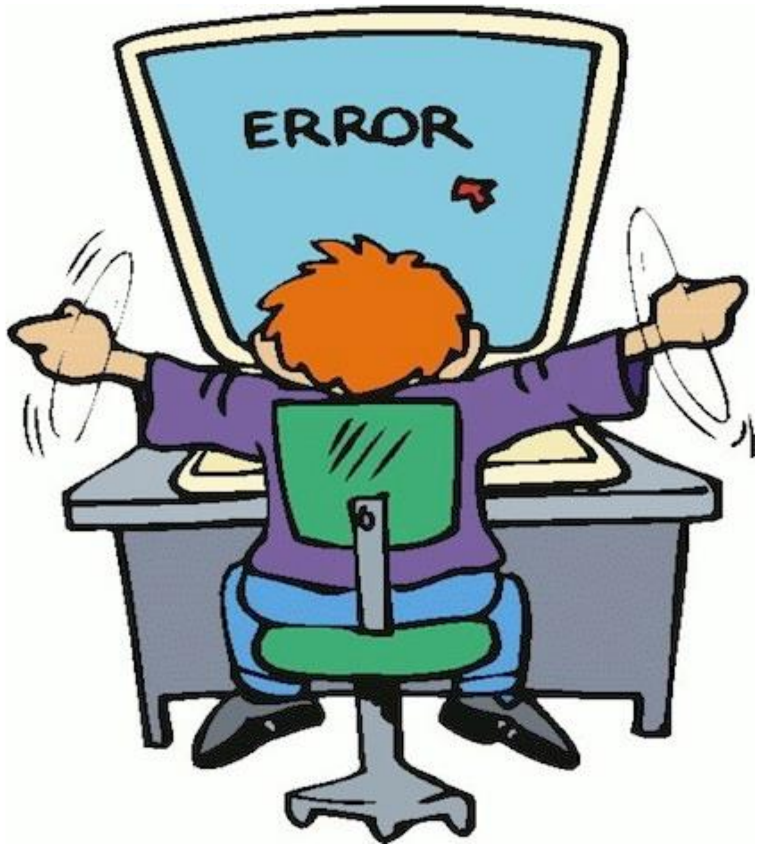
Modern systems are serving many aspects of our life

...

Search Engine	Social Network	Cloud Service	Office Software	...

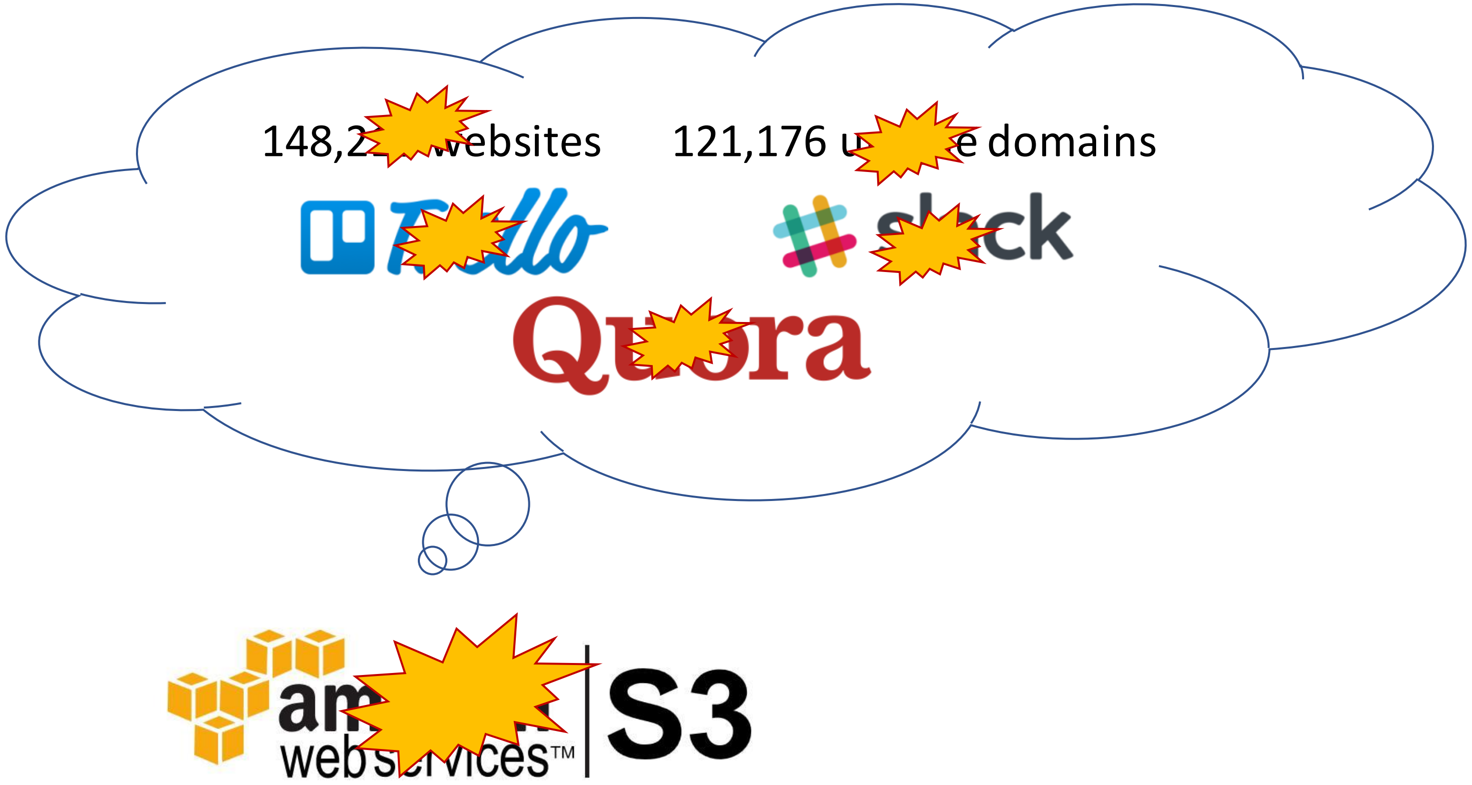
Motivation & Background

System reliability is very crucial!



Motivation & Background

An Real-World Example

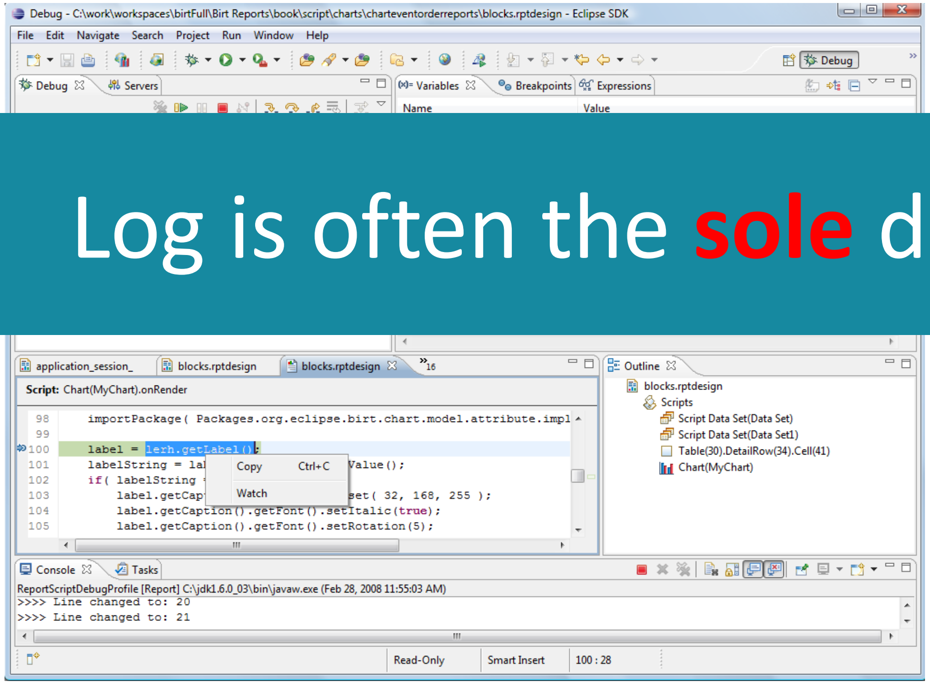


[Statistics from: https://techcrunch.com/2017/02/28/amazon-aws-s3-outage-is-breaking-things-for-a-lot-of-websites-and-apps/]

Motivation & Background

How to maintain these systems and keep them reliable?

Traditional tools (e.g., Java Debugger)



hard to apply in distributed systems

In practice:

Log is often the **sole** data source for troubleshooting

widely utilized by developers in system maintenance

Motivation & Background

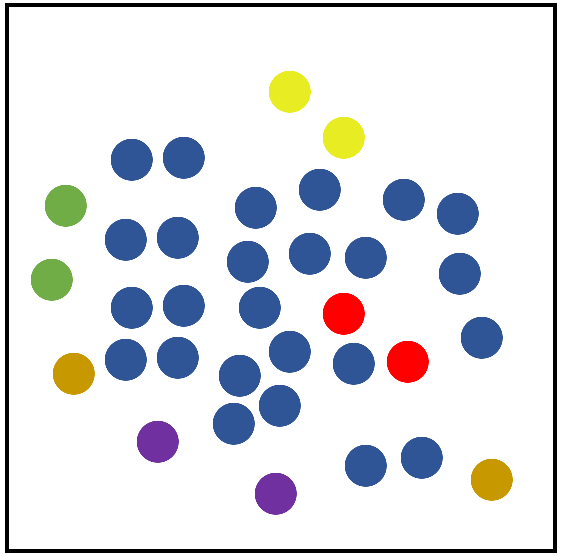
Manual Inspecting of logs is **infeasible!**

scale

complexity

Automated log analysis is **highly in demand**

Problem Identification



Normal

Different types of problem

Challenge:
In practice, no labels…

Clustering

Challenges

1) Huge log size

10+ Terabytes

2) Highly-imbalanced

3) Clustering alone cannot determine problematic or not

Imbalanced Log Data

Why is log data imbalanced?

Cloud-based online service systems

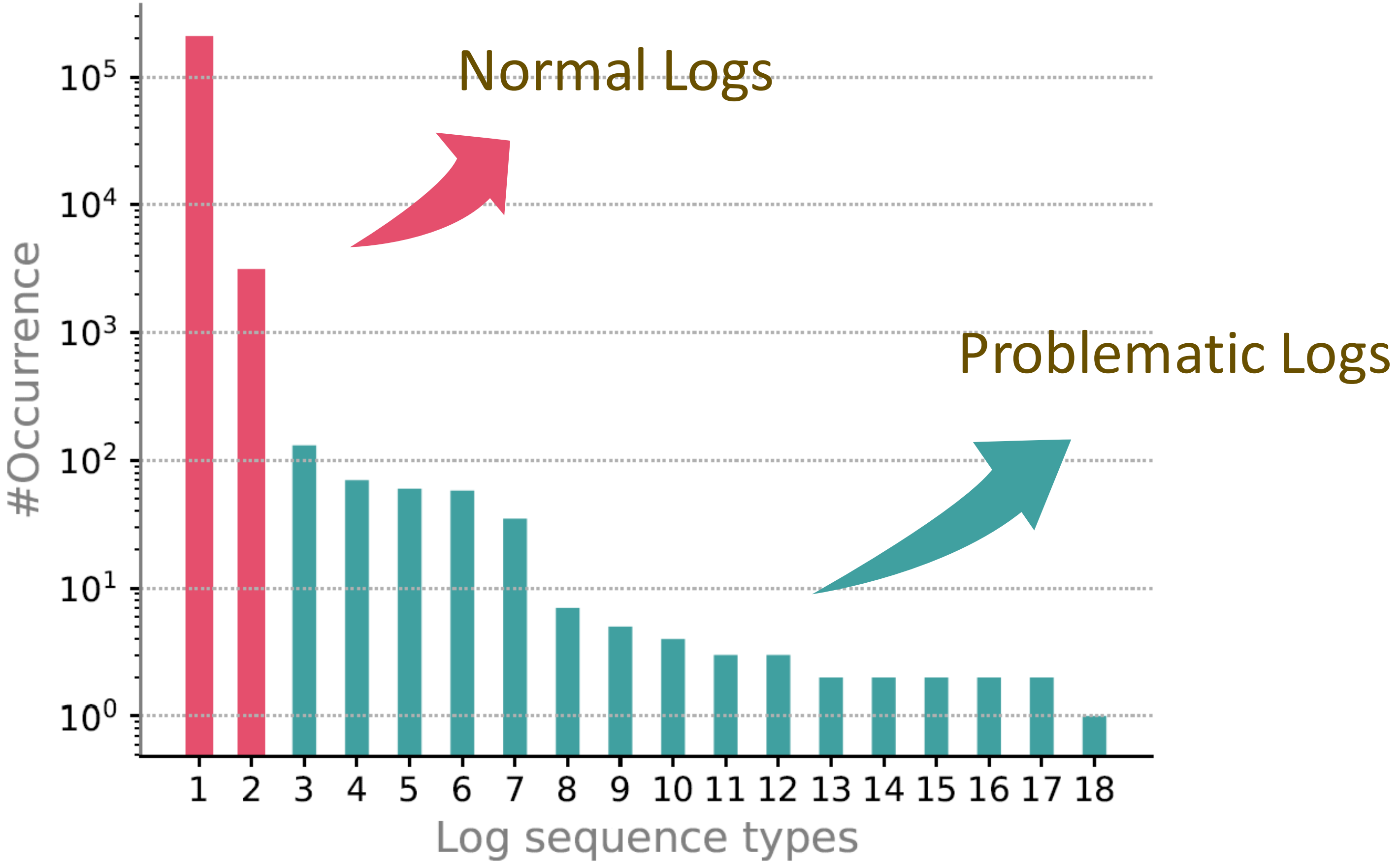
“Five Nines” of service availability

99.999%

Imbalanced Log Data

System executes normally in most cases and problems occasionally happen

Long tail distribution:
An example



KPIs

System KPIs (Key Performance Indicators)

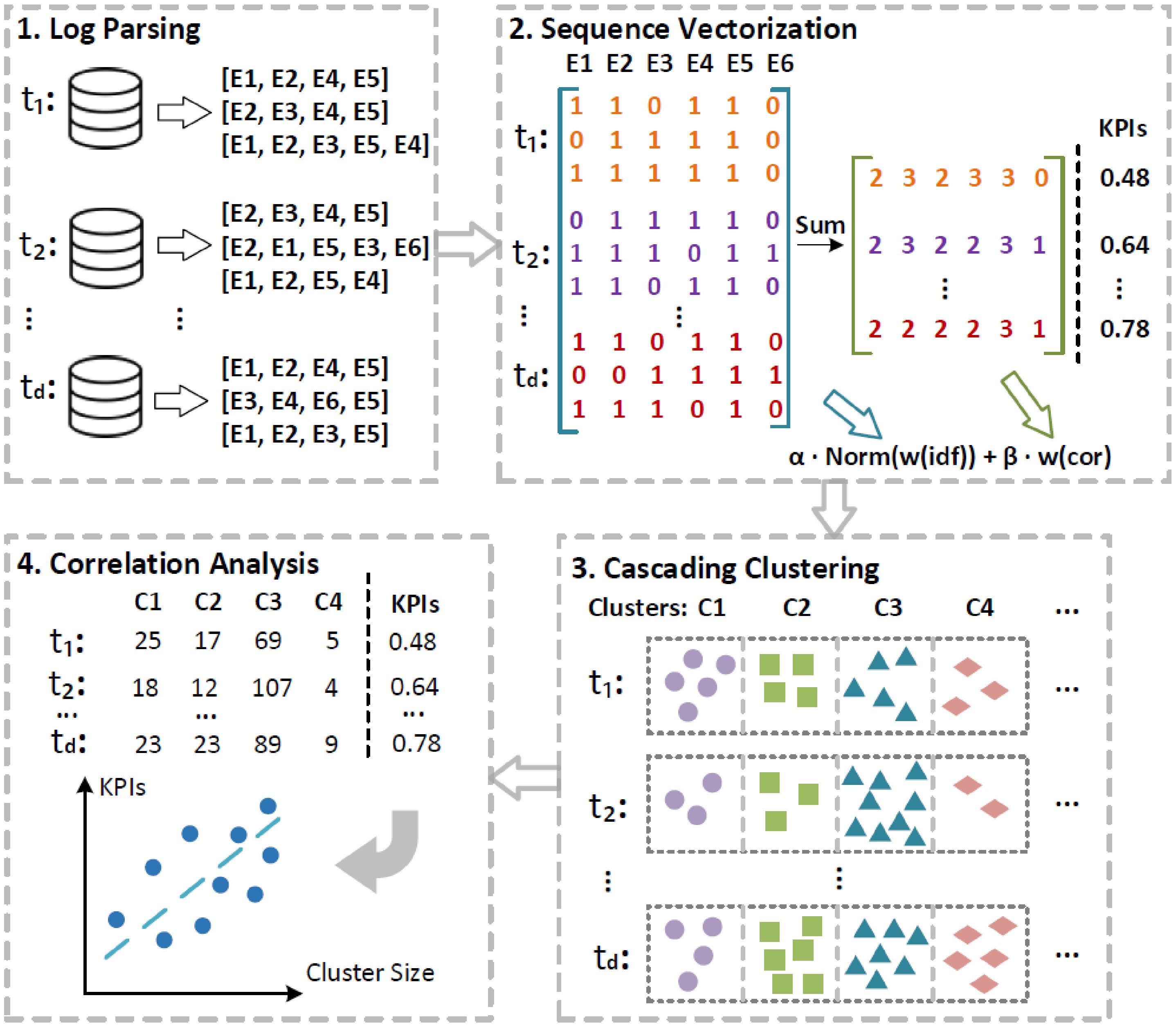
measure the system‘s **health status** in a certain time period, i.e.,

- ❑ Failure Rate
- ❑ Service Availability
- ❑ Average Request Latency

Periodically collected!

Framework

Framework of Log3C



Log Parsing

Raw Logs

01	Name=Request (GET:http://AAA:1000/BBBB/sitedata.html)
02	Leaving Monitored Scope (EnsureListItemsData) Execution Time=52.901315359
03	HTTP request URL: /14/teamX/Emails/MrX(MrX@mail.com)/20Private%-1b1c-48f0-b206-40a7279b2829.eml
04	HTTP Request method: GET
05	HTTP request URL: /55/RST/UVWX/YZ/ABCDE/Lists/Attachments/docXX.doc
06	Overridden HTTP request method: GET
07	HTTP request URL: http://AAA:1000/BBBB/sitedata.html
08	Leaving Monitored Scope (Request (POST:http://AAA:100/BBBB/sitedata.html)) Execution Time=334.319268903038

Log Parsing

Event Templates

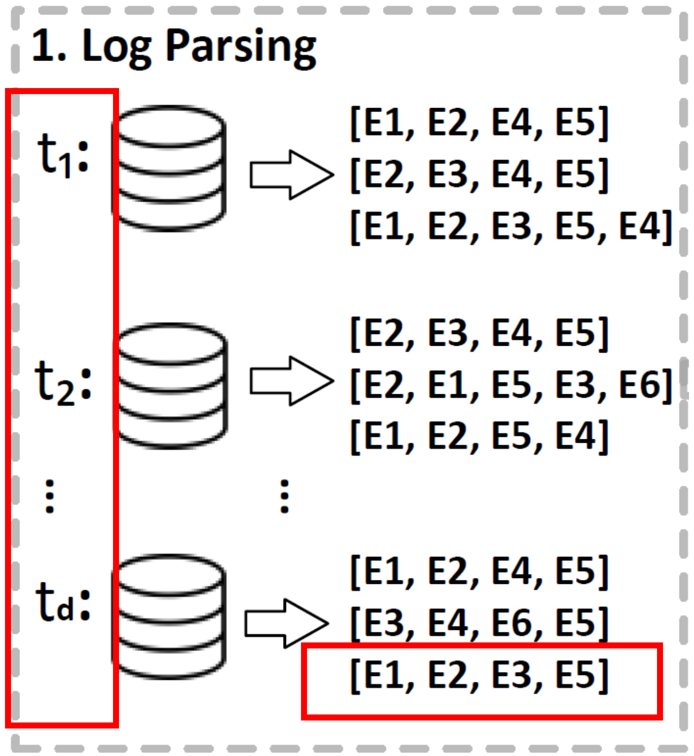
E1	Name=Request (*)
E2	Leaving Monitored Scope (*) Execution Time = *
E3	HTTP Request method: *
E4	HTTP request URL: *
E5	Overridden HTTP request method: *

Log3C-Log Parsing

Log Parsing

Logs in each time interval are parsed separately

Logs that share the same task ID are linked as a log sequence



Log3C– Sequence Vectorization

Weights from two perspectives:

IDF weighting

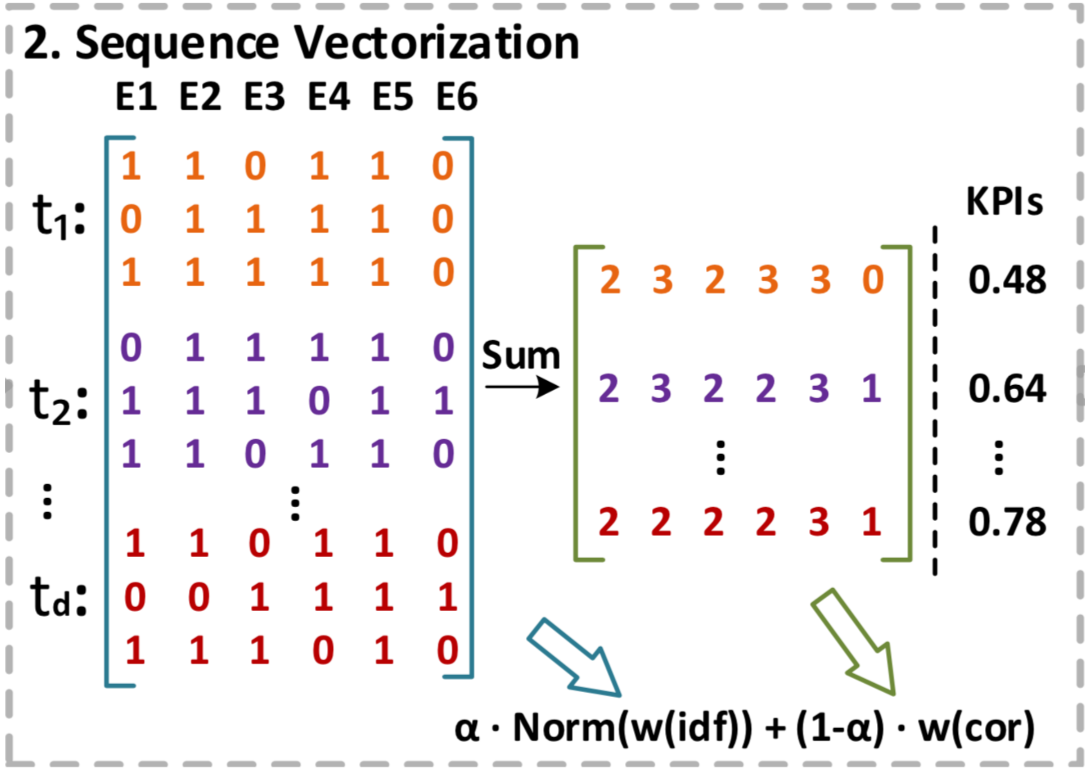
Importance weighting

1. IDF weighting

2. Importance weighting

$$w_{idf}(e) = \log\left(\frac{N}{n_e}\right) \quad (1)$$

$$w(e) = \alpha * Norm(w_{idf}(e)) + (1 - \alpha) * w_{cor}(e) \quad (2)$$

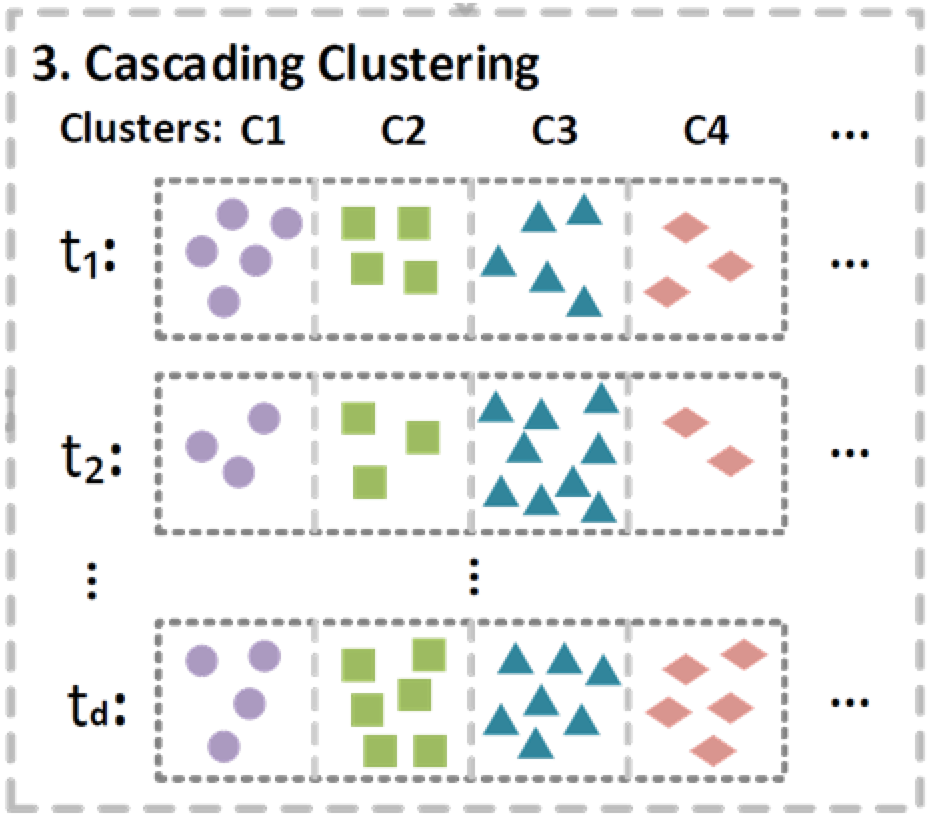


Log3C– Cascading Clustering

Target: Conduct clustering on log sequences from each time interval

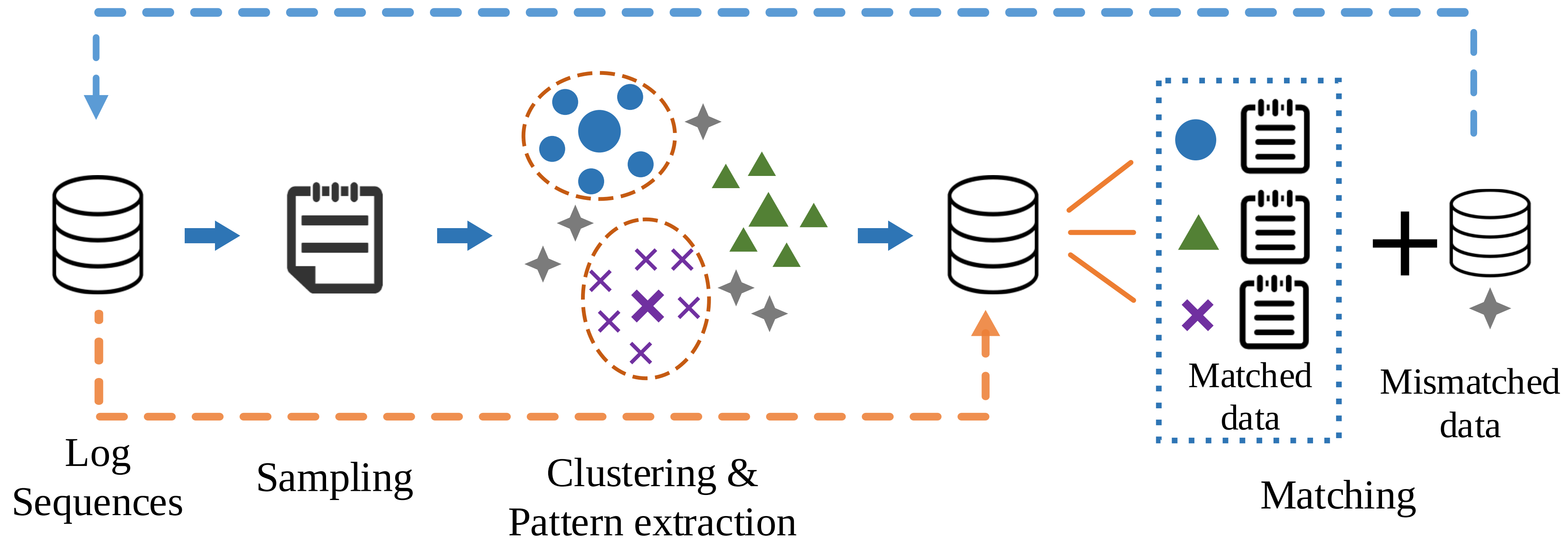
Challenge:
Huge amount of data

Traditional Clustering:
Distance calculation between any two data samples



Log3C– Cascading Clustering

Cascading Clustering: Efficient and Effective



Log3C – Correlation Analysis

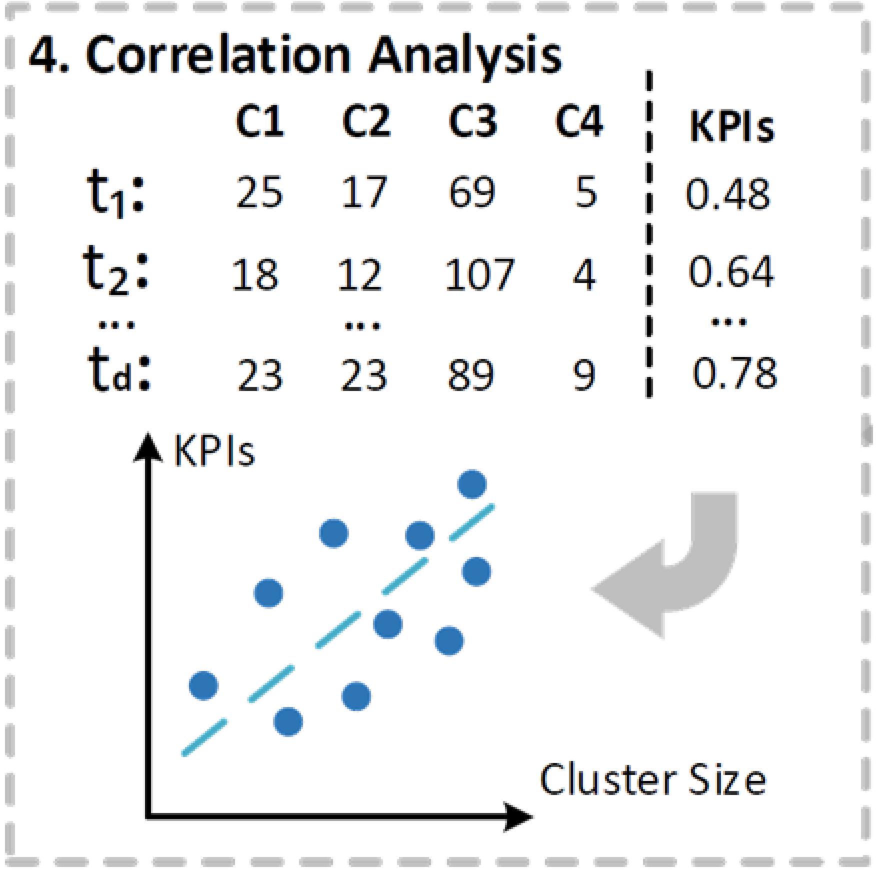
Impactful problems:
Can lead to the degradation of KPI

Target:
Identify clusters that are highly correlated with KPI's changes

Method:
Model *Cluster sizes—KPI values* relation

Multivariate Linear Regression (MLR)

t-statistic hypothesis test



Experiments

Datasets: Real-world data from the service system X

✓ Logs during a certain time period on three different days

Data	Snapshot starts	#Log Seq (Size)	#Events	#Types
Data 1	Sept 5th 10:50	359,843 (722MB)	365	16
Data 2	Oct 5th 04:30	472,399 (996MB)	526	21
Data 3	Nov 5th 18:50	184,751 (407MB)	409	14

Manual labelling from two aspects:

1. Does the log sequence indicate a problem?
2. What is the problem type?

Experiments

Evaluation Metrics:

1. **Problem Detection** (Binary Classification):

   Precision / Recall / F1-Measure

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

2. **Problem Identification** (Clustering)

   Normalized Mutual Information (NMI) ~ between [0, 1]

$$NMI(Y, C) = \frac{2 \times I(Y; C)}{[H(Y) + H(C)]}$$

Y = class labels

C = cluster labels

H(.) = Entropy

I(Y;C) = Mutual Information b/w Y and C

3. **Clustering Time** (in seconds)

Experiments

Accuracy of Problem **Detection**:

Data 1	Precision	Recall	F1-measure
PCA	0.465	0.946	0.623
Invariants Mining	0.604	1	0.753
Log3C	**0.900**	0.920	**0.910**
Data 2	Precision	Recall	F1-measure
PCA	0.142	0.834	0.242
Invariants Mining	0.160	0.847	0.269
Log3C	**0.897**	0.826	**0.860**
Data 3	Precision	Recall	F1-measure
PCA	0.207	0.922	0.338
Invariants Mining	0.168	0.704	0.271
Log3C	**0.834**	0.903	**0.868**

Experiments

Accuracy of Problem **Identification**:

Data 1	Size	10k	50k	100k	200k
	Log3C-SC	0.659	0.706	0.781	0.822
	Log3C	**0.720**	**0.740**	**0.798**	**0.834**
Data 2	Size	10k	50k	100k	200k
	Log3C-SC	0.610	0.549	0.600	0.650
	Log3C	**0.624**	0.514	**0.663**	**0.715**
Data 3	Size	10k	50k	100k	180k
	Log3C-SC	0.601	0.404	0.792	0.828
	Log3C	**0.680**	**0.453**	**0.837**	**0.910**

Log3C-SC is the comparison method, which replaces the *Cascading Clustering* with the *standard clustering* (HAC)

Experiments

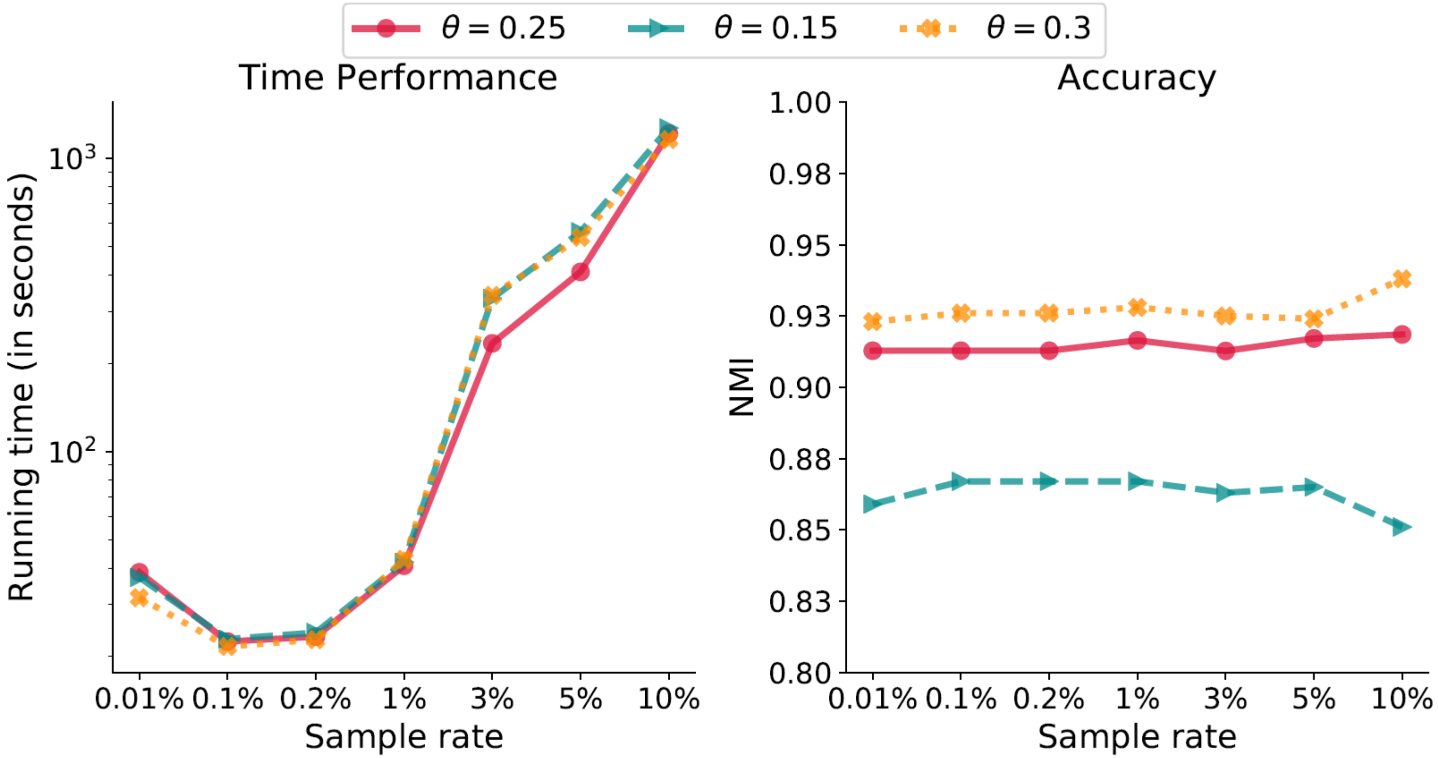
Time Performance of Cascading Clustering

Data 1	Size	10k	50k	100k	200k
	SC	127.6	2319.2	9662.3	38415.5
	CC	1.0	4.3	9.2	20.7
Data 2	Size	10k	50k	100k	200k
	SC	80.6	2469.1	8641.2	38614.0
	CC	0.7	3.8	9.5	18.9
Data 3	Size	10k	50k	100k	180k
	SC	81.5	2417.2	8761.2	33728.3
	CC	0.8	4.0	8.8	18.3

1800x faster on Data 1 of size 200k

Experiments

Cascading Clustering with Different Sample Rate



Time/NMI -- Sample Rate

Decreasing sample rate does not sacrifice the accuracy while greatly reducing the time

Conclusion

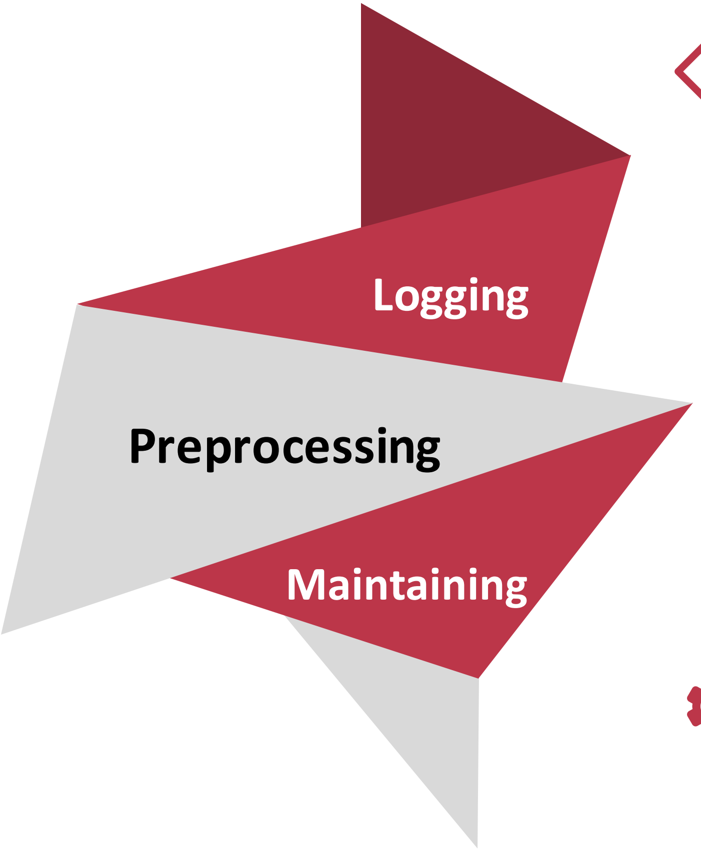
Contributions:

- ✓ Cascading Clustering, Efficient and Effective
- ✓ Propose Log3C by integrating cascading clustering and correlation analysis
- ✓ Log3C has been successfully applied in the **actual maintenance of online service** systems at Microsoft.

SharePoint Microsoft Azure

LOGPAI

— Log Analytics Powered by AI



LogAdvisor (ICSE'15)

- Learning to log: A framework for determining optimal logging points

LogHub (in submission)

- A collection of system log datasets for massive log analysis

Logizer (ISSRE'16)

- A log analysis toolkit for automated anomaly detection

LoggingDescriptions (ASE'18)

- A collection of Software Logging Statements in source code

LogParser (DSN'16)

- A toolkit for automated log parsing

Log3C (FSE'18)

- Log-based Problem Identification

https://github.com/logpai

Thanks!