Towards Pragmatic Pre-processing of logs

Weiyi Shang

https://ece.uwaterloo.ca/~wshang/

UNIVERSITY OF WATERLOO

Academia

Industry

M.Sc., Ph.D., Post-Doc
Sept. 2008 - July. 2015

Performance Engineer
Sept. 2010-Aug. 2014

Associate Prof.
Concordia University Research Chair
Aug. 2015 - June 2023

Associate Prof.
July 2023 - Present

Logs are the most common data to be used in AIOps

what is the typical data to be analyzed in AIOps

In **AIOps** (Artificial Intelligence for IT Operations), the **typical data** analyzed includes a wide variety of IT operational data, usually coming from multiple sources. Here's a breakdown:

---

1. Logs

- Application logs
- Server logs
- Database logs
- Security logs
- Event logs
- Network device logs (routers, firewalls, switches)

Logs are one of the only resources of information

Operator

Developer

"The Bone of the System"
[ICSE SEIP 2016]

Microsoft Research

Ma
log
de
s

How to make logging statements?

Re

Produce at run-time

System issues

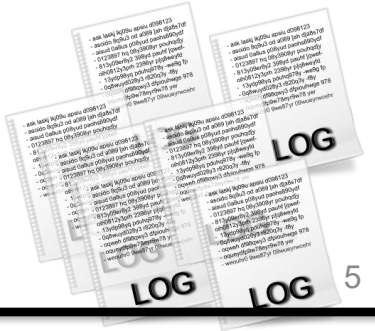
Log Processing Apps

How to analyze logs

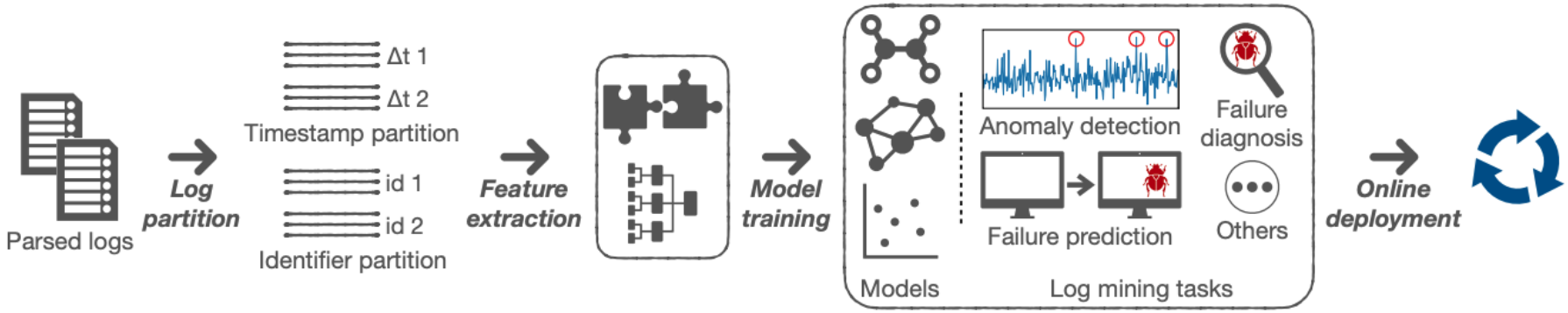
Perl

RegEx
[^...]

Report



General flow of log analysis



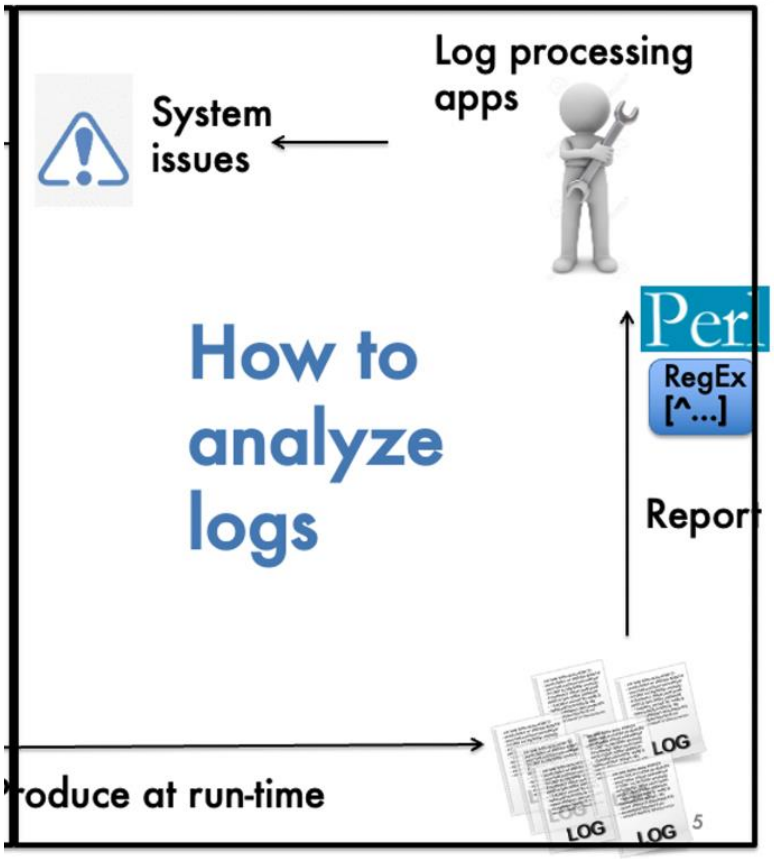
[He et al. CSUR21]

A Survey on Automated Log Analysis for Reliability Engineering

SHILIN HE, Microsoft Research
PINJIA HE, Department of Computer Science, ETH Zurich
ZHUANGBIN CHEN, TIANYI YANG, YUXIN SU, and MICHAEL R. LYU, Department of Computer Science and Engineering, The Chinese University of Hong Kong

Logs are semi-structured text generated by logging statements in software source code. In recent decades, software logs have become imperative in the reliability assurance mechanism of many software systems because they are often the only data available that record software runtime information. As modern software is evolving into a large scale, the volume of logs has increased rapidly. To enable effective and efficient usage of modern software logs in reliability engineering, a number of studies have been conducted on automated log analysis. This survey presents a detailed overview of automated log analysis research, including how to automate and assist the writing of logging statements, how to compress logs, how to parse logs into structured event templates, and how to employ logs to detect anomalies, predict failures, and facilitate diagnosis. Additionally, we survey work that releases open-source toolkits and datasets. Based on the discussion of the recent advances, we present several promising future directions toward real-world and next-generation automated log analysis.

CCS Concepts: • **Software and its engineering → Software maintenance tools; Software creation and management.**

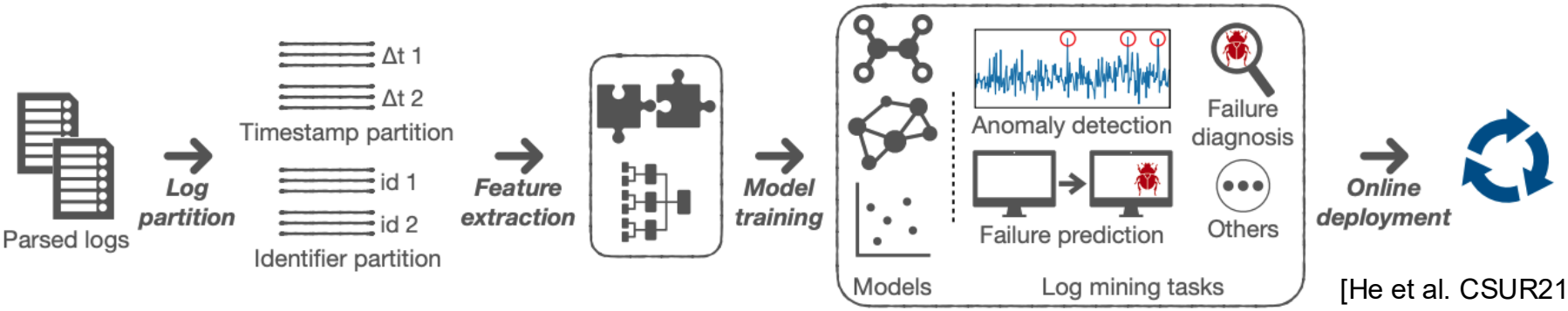


Very successful research area

Why?

Limited generalized toolsets in practice

General flow of log analysis



[He et al. CSUR21]

Pre-processing of logs

Most of the research is here

LOGPAI

Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics

Jieming Zhu*, Shilin He*, Pinjia He†✉, Jinyang Liu‡, Michael R. Lyu‡

†School of Data Science, The Chinese University of Hong Kong, Shenzhen (CUHK Shenzhen), China
‡Department of Computer Science and Engineering, The Chinese University of Hong Kong, China
jiemingzhu@ieee.org slhe@link.cuhk.edu.hk hepinjia@cuhk.edu.cn {jyliu, lyu}@cse.cuhk.edu.hk

LOGPAI

Tools and Benchmarks for Automated Log Parsing

Jieming Zhu¶, Shilin He†, Jinyang Liu‡, Pinjia He§, Qi Xie‖, Zibin Zheng‡, Michael R. Lyu†

¶Huawei Noah's Ark Lab, Shenzhen, China
†Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
‡School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
§Department of Computer Science, ETH Zurich, Switzerland
‖School of Computer Science and Technology, Southwest Minzu University, Chengdu, China

jmzhu@ieee.org, slhe@cse.cuhk.edu.hk, liujy@logpai.com, pinjiahe@gmail.com
qi.xie.swun@gmail.com, zhzibin@mail.sysu.edu.cn, lyu@cse.cuhk.edu.hk

Log Parsing

logInfo("Found block $blockId locally")

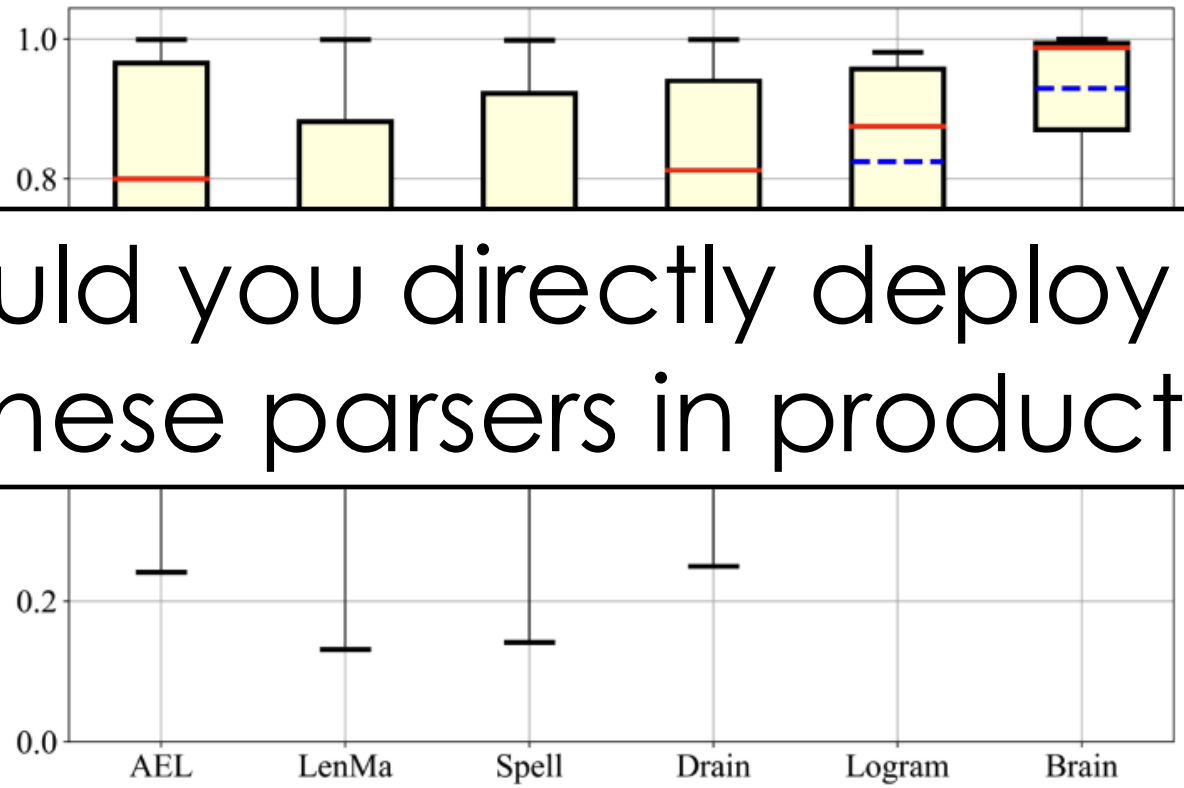
Generate

17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_20 locally

Contain

Timestamp: 17/06/09 20:11:11; Level: INFO
Logger: storage.BlockManager
Static template: Found block <*> locally
Dynamic variable(s): rdd_42_20

Log parsing does not seem to be a big issue on papers



Would you directly deploy any of these parsers in production?

Fig. 9. Boxplot of word-level parsing accuracy on 16 benchmark datasets.

[Yu et al TSC 2023]

The data is way too pretty

2015-10-18 18:01:47,978 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Created MRAppMaster for application appattempt_1445144423722_0020_000001

081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating

BGL

- 1117838570 2005.06.03 R02-M1-N0-C:J12-U11 2005-06-03-15.42.50.675872 R02-M1-N0-C:J12-U11 RAS KERNEL INFO instruction cache parity error corrected

The ugly truth of logs in real life

Anything can be
in there!

The ugly truth of logs in real life

```
{ Error: Request failed with status code 500
    at createError (...)
    at settle (...)
...
        [Symbol(isCorked)]: false,
        [Symbol(outHeadersKey)]: [Object] },
     data:
      ... },
  isAxiosError: true,
  toJSON: [Function] }
```

```
/home/travis/.nvm/versions/node/v6.4.0/lib
└─┬ markdown-spellcheck@0.11.0
  ├── async@1.5.2
  ├─┬ chalk@1.1.3
...
    │ └── unique-concat@0.2.2
    └── native-promise-only@0.8.1
```

```
GORACE=""
GOROOT="/home/travis/.gimme/versions/go1.8.linux.amd64"
GOTOOLDIR="/home/travis/.gimme/versions/go1.8.linux.amd64/pkg/tool/linux_amd64"
```

The ugly truth of logs in real life

```
GORACE=""
GOROOT="/home/travis/.gimme/versions/go1.8.linux.amd64"
GOTOOLDIR="/home/travis/.gimme/versions/go1.8.linux.amd64/pkg/tool/linux_amd64"
```

After parsing

```
GORACE=$
GOROOT=$
GOTOOLDIR=$
```

Is this correct?

Is this useful?

The ugly truth of logs in real life

```
{ Error: Request failed with status code 500
    at createError (...)
    at settle (...)
...
        [Symbol(isCorked)]: false,
         [Symbol(outHeadersKey)]: [Object] },
      data:
       ... },
  isAxiosError: true,
  toJSON: [Function] }
```

How about this?

We need a second thought on how to pre-process these logs

```
GORACE=""
GOROOT="/home/travis/.gimme/versions/go1.8.linux.amd64"
GOTOOLDIR="/home/travis/.gimme/versions/go1.8.linux.amd64/pkg/tool/linux_amd64"
```

Log Parsing

logInfo("Found block $blockId locally")

Generate

17/06/09 20:11:11 INFO storage.BlockManager: Found block rdd_42_20 locally

Contain

Timestamp: 17/06/09 20:11:11; Level: INFO
Logger: storage.BlockManager
Static template: Found block <*> locally
Dynamic variable(s): rdd_42_20

```
Error: Request failed with status code 500
    at createError (...)
    at settle (...)
...
        [Symbol(isCorked)]: false,
        [Symbol(outHeadersKey)]: [Object] },
     data:
      ... },
  isAxiosError: true,
  toJSON: [Function] }
```

What are the other structures of logs?

We manually studied some "unconventional" logs

```
Packer v1.0.2
Your version of Packer is out of date! The latest version
is 1.1.2. You can update by downloading from www.packer.io
```

Simple

```
[-] - mock defined in beforeall is counted independently -> Expected $true but gc
at Verify-True
at <ScriptBlock
...
at <ScriptBlock
at <ScriptBlock
```

Maybe logs should be pre-processed by chunks instead of lines!

```
Huge version
+acl
+arabic          +file_in_path     +mouse_sgr        +tag_binary
+autocmd         +find_in_path     -mouse_sysmouse   +tag_old_static
```

Tabular

```
[(DefiniteUnitId (DefUnitId {unDefUnitId = UnitId
"base-4.8.2.0-0d6d1084fbc041e1cded9228e80e264d"}),DefaultRenaming),(DefiniteUnitI
...
(DefUnitId {unDefUnitId = UnitId
"file-embed-0.0.11-5eblKl2yNd7K74FblwfEQk"}),DefaultRenaming)]
```

Recursive

What are the formats of data in each chunk?

Code

Diff

Paragraph

Stack trace

Ite

How do we handle these in the context of AIOps?

Variable assignment

JSON like

Tree like

YAML like

The first attempt would be automatically identifying these chunks and their formats

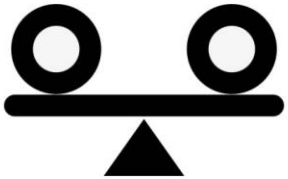
Results based on an AI-based solution.

Name	F1	Precision	Recall	ROC AUC
Code	0.56	0.83	0.42	0.98
Diff Hunk	0	0	0	1
Items	0.87	0.91	0.83	0.98
Tree Like	1	1	1	1
Variable Assignments	0	0	0	0.99
YAML Like	0.93	0.95	0.9	1
End of Line	0.88	0.84	0.93	0.97

A good start, but there is big rooms of improvement.

Is that all for log parsing?

Non-functional attributes are detrimental to the success of adoption in industry



Accuracy

Performance

Stability

Most of the research is here

Practitioners care about these too!

Logram: A fast and scalable parser

Raw log (Unstructured)	Found block rdd_42_20 locally Found block rdd_42_22 locally Found block rdd_42_23 locally Found block rdd_42_24 locally

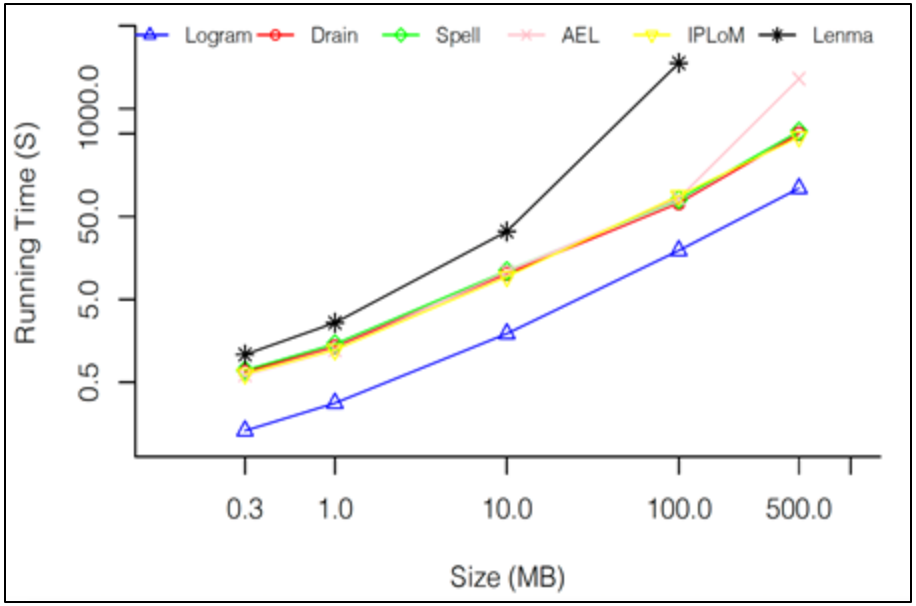
Each static token has a higher number of appearance.
Token "Found" appears 4 times.

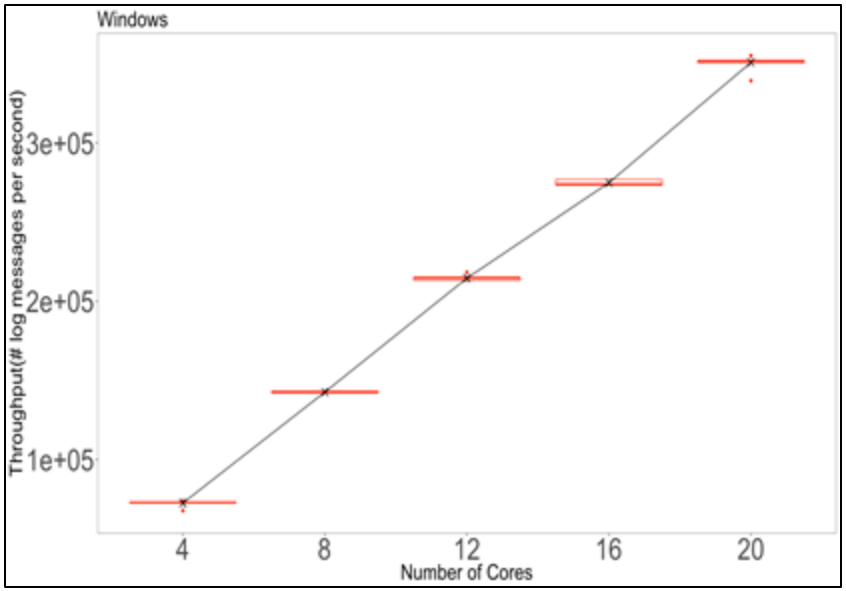
Each dynamic token has a lower number of appearance.
Token "rdd_42_20" appears only once.

We use the number of appearances to distinguish static and dynamic tokens.

[Dai et al. TSE]

Logram: A fast and scalable parser

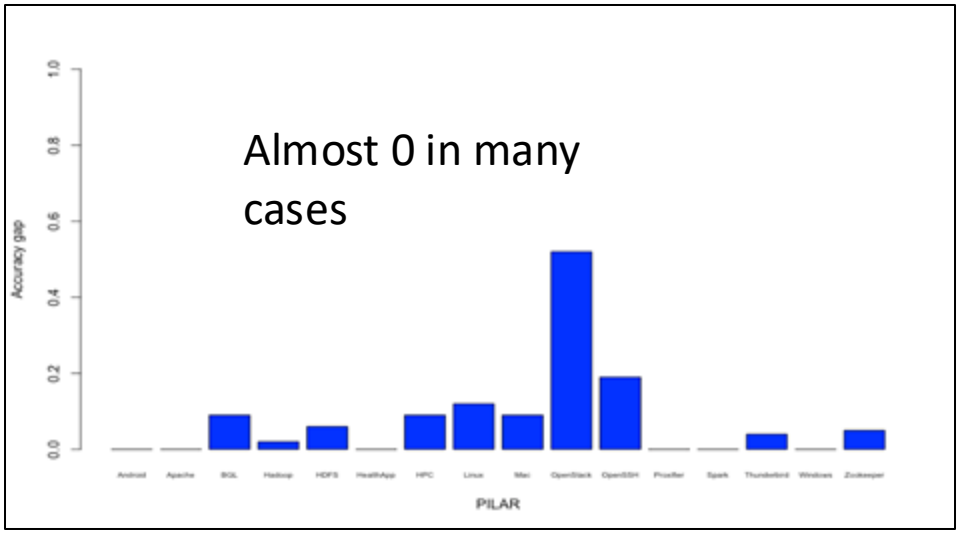




Basic idea: using the count of n-grams to parse.

[Dai et al. TSE]

PILAR: A parser that is parameter insensitive

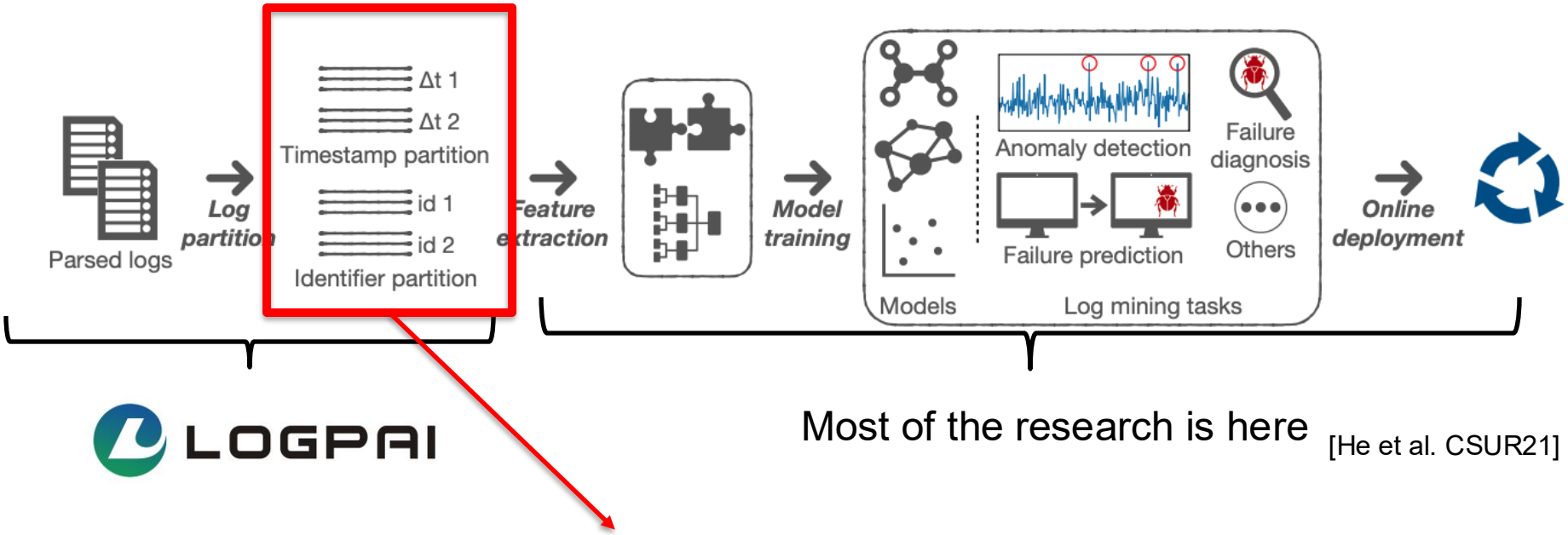


Accuracy gaps with varying parameter on the same datasets

Basic idea: using probability of token to parse.

[Dai et al. ICSE]

Preprocessing of logs is not just parsing

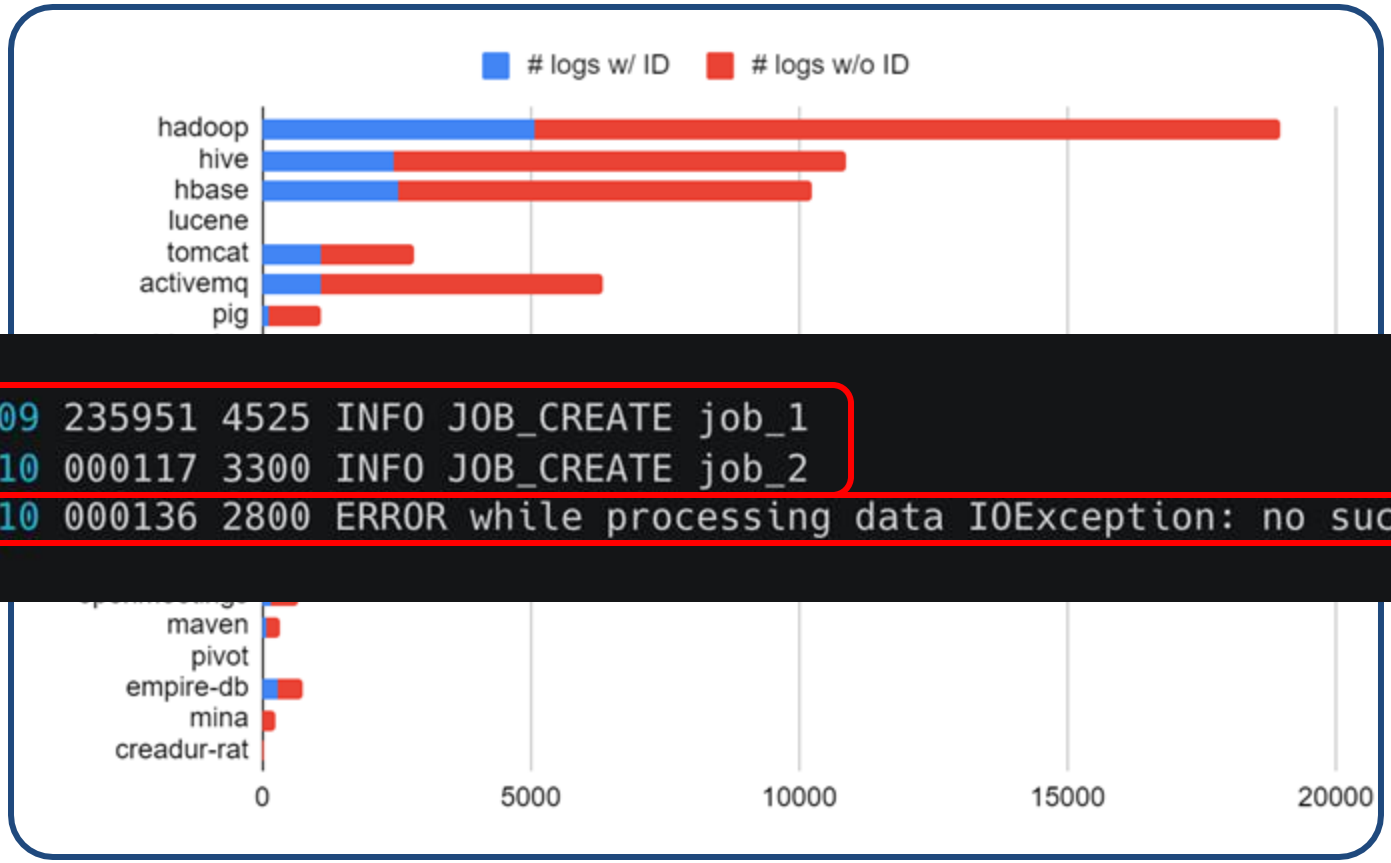


LOGPAI

Most of the research is here [He et al. CSUR21]

Partition logs is extremely important to the AIOps tasks

Only 21% of logging statements contain IDs.



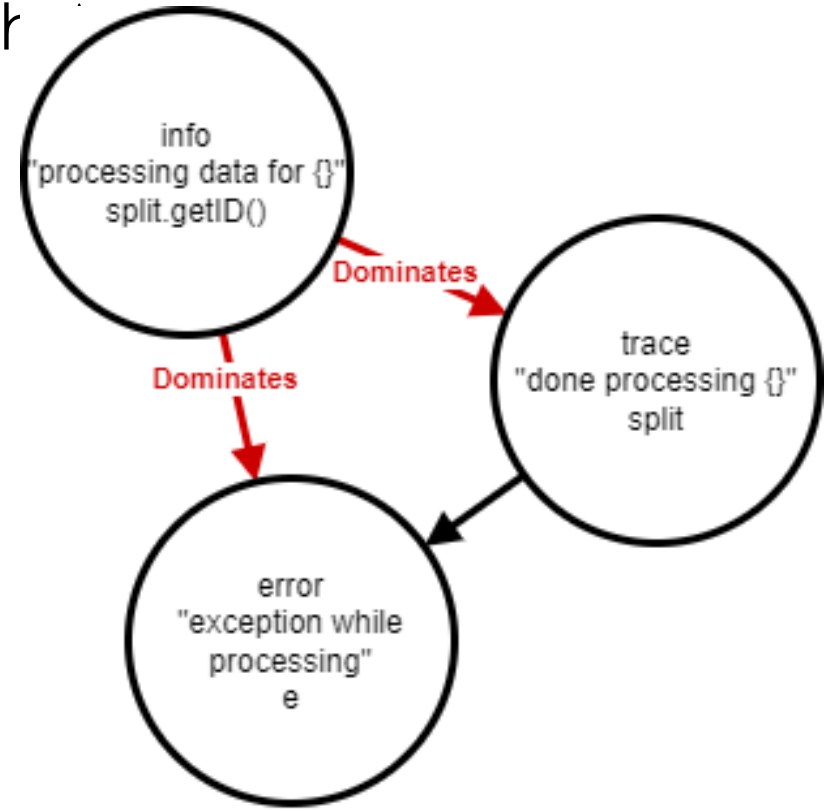

```
1 081109 235951 4525 INFO JOB_CREATE job_1
2 081110 000117 3300 INFO JOB_CREATE job_2
3 081110 000136 2800 ERROR while processing data IOException: no such file
```

Most relationship between logs are lost at runtime. This makes log analysis difficult.

[Zhao et al. SANER]

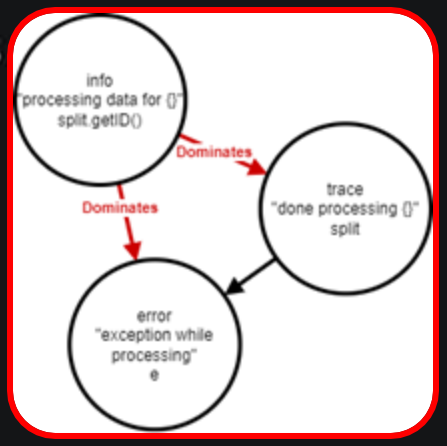
We leverage node dominations to create relationship between logs.

Node A dominates node B means that: to get to B, one must go through



Here is an example of how the IDs are propagated:

```
1  void process(Split split) throws IOException {
2    try {
3
4      LOG.info("Processing data for {}", split.getID());
5
6      // processing some data...
7
8      if (LOG.isTraceEnabled()) {
9        LOG.trace("done processing {}", split);
10     }
11   } catch (IOException e) {
12     LOG.error("Exception while processing", e);
13     throw e;
14   }
15 }
```



Thanks to all my collaborators

Testing

ERICSSON Software testing

WIND

Mobeewave Mobile apps

System

Alibaba.com
ERA ENVIRONMENTAL Management Solutions
SE for AI
AIOps
AI Performance

Source

ASML
Mobeewave
Source code performance

WIND Quality

ERA ENVIRONMENTAL Management Solutions Exception Handling

BlackBerry Make better logging

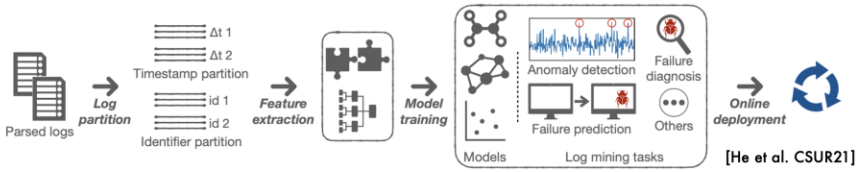
ERA ENVIRONMENTAL Management Solutions Bugs

Development

BlackBerry
ERICSSON
Alibaba.com
Log analytics

ERA ENVIRONMENTAL Management Solutions
Performance analytics

General flow of log analysis



[He et al. CSUR21]

Pre-processing of logs

Most of the research is here

LOGPAI

The ugly truth of logs in real life

```
{ Error: Request failed with status code 500
    at createError (...)
    at settle (...)
...
        [Symbol(isCorked)]: false,
        [Symbol(outHeadersKey)]: [Object] },
      data:
        ... },
  isAxiosError: true,
  toJSON: [Function] }
```

```
/home/travis/.nvm/versions/node/v6.4.0/lib
└┬ markdown-spellcheck@0.11.0
  ├── async@1.5.2
  ├┬ chalk@1.1.3
...
  │ └── unique-concat@0.2.2
  └── native-promise-only@0.8.1
```

```
GORACE=""
GOROOT="/home/travis/.gimme/versions/go1.8.linux.amd64"
GOTOOLDIR="/home/travis/.gimme/versions/go1.8.linux.amd64/pkg/tool/linux_amd64"
```


Weiyi Shang https://ece.uwaterloo.ca/~wshang/

We manually studied some "unconventional" logs

```
Packer v1.0.2
Your version of Packer is out of date! The latest version
is 1.1.2. You can update by downloading from www.packer.io
```

Simple

```
[-] - mock defined in beforeall is counted independently -> Expected $true but gc
at Verify-T[1]
at <ScriptBlock
...
at <ScriptBlock
at <ScriptBlock
```

Maybe logs should be pre-processed by chunks instead of lines!

```
Huge version
+acl
+arabic          +file_in_path     +mouse_sgr        +tag_binary
+autocmd         +find_in_path     -mouse_sysmouse   +tag_old_static
```

Tabular

```
[(DefiniteUnitId (DefUnitId {unDefUnitId = UnitId
"base-4.8.2.0-0d6d1084fbc041e1cded9228e80e264d"})),DefaultRenaming),(DefiniteUnitI
...
(DefUnitId {unDefUnitId = UnitId
"file-embed-0.0.11-5eblKl2yNd7K74FblwfEQk"})),DefaultRenaming)]
```

Recursive

Adopting AIOps in practice is more than a simple pipeline

We need more than better AI tools.

We need data! Real industrial data!

LOGPAI

We need industrial-involved solutions!