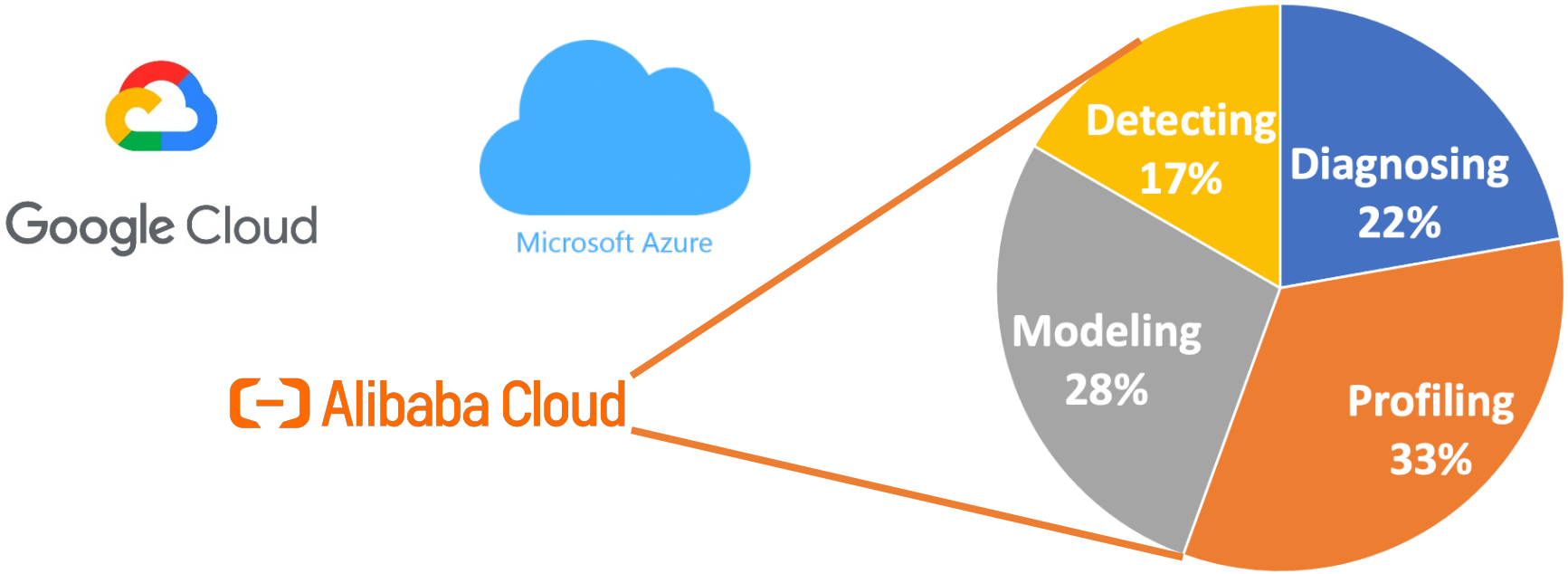
FAST '21
19th USENIX Conference on File and Storage Technologies

On the Feasibility of Parser-based Log Compression in Large-Scale Cloud Systems

Junyu Wei, Guangyan Zhang, Yang Wang,
Zhiwei Liu, Zhanyang Zhu, Junchao Chen, Tingtao Sun, Qi Zhou

Log compression in cloud systems

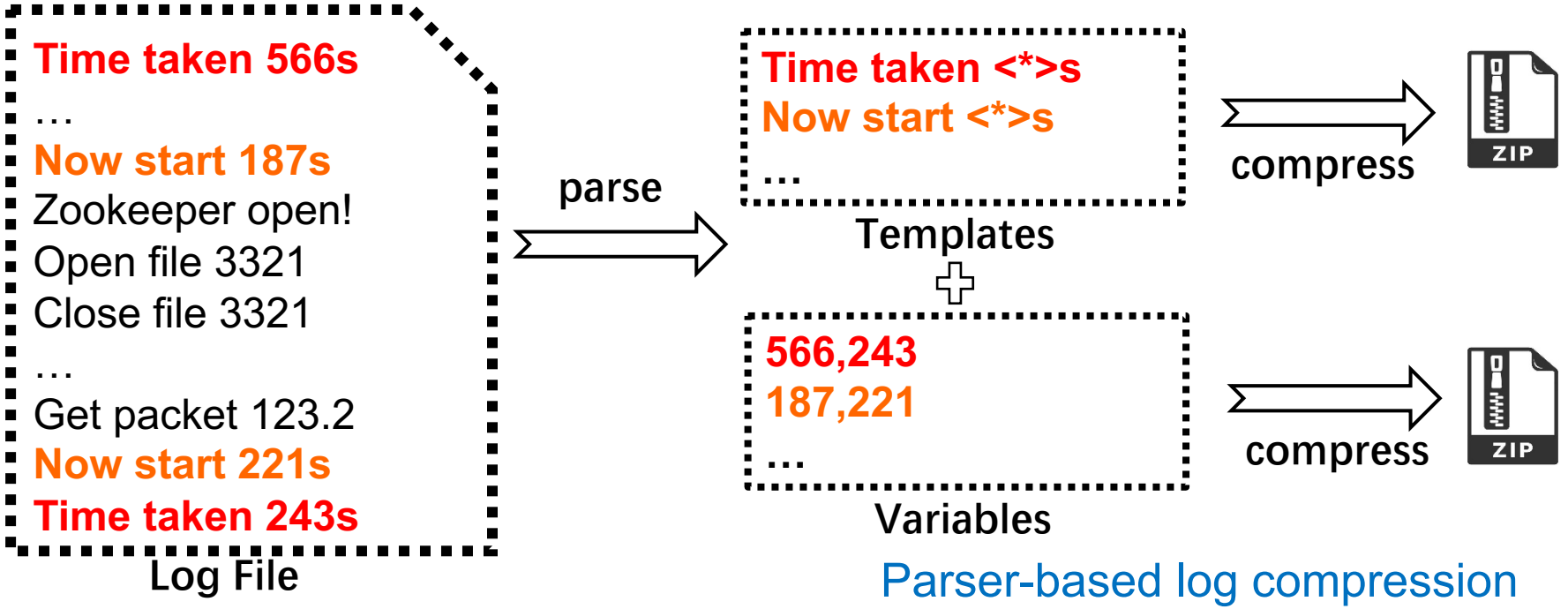
- Logs are widely used in cloud systems
- Large amounts of logs are produced per day - ≈1PB at AliCloud
- Need to store these logs for at least several months
- Compression is desirable to save storage cost



Cloud logs and its application ratio in AliCloud

General-purpose vs log-specific methods

- General-purpose compression methods (e.g. gzip, LZMA, PPMd, bzip …)
- Log-specific compression methods (e.g. LogArchive (Christensen et. SIGMOD'13), Drain (He et. ICWS'17), Logzip (Liu et. ASE'19) …)
- Parser-based methods reported to have promising performance
- Is it feasible to use parser-based methods on large-scale cloud logs?



Production logs from Alicloud

- We collected 18 types of logs (1.76TB in total)
  - User behavior tracing
  - Infrastructure monitoring
  - Warning and error reporting
  - Periodical summary

```
2019-11-03T23:59:59.885+08:00 232.230.24.16
GET snull null 200 0 306 907 null
2019-11-03T23:59:59.904+08:00 220.252.56.14
GET null null 200 0 306 922 null
```

User behavior tracing (Log L)

```
Aug 28 03:09:02 h10c10322.et15 su[57118]: (to
nobody) root on none
Aug 28 03:09:02 h10c10322.et15 su[57118]: session
opened for user nobody by (uid=0)
```

Warning and error reporting (Log Q)

```
[2018-01-12 08:53:12.188370] project:393 logstore:
XDoFiqnlmZd shard:78 inflow:3376 dataInflow: 18
[2018-01-12 08:53:12.188390] project:656 logstore:
lOdMafL31Pg shard:37 inflow:7506 dataInflow: 42
```

Infrastructure monitoring (Log D)

```
[2020-04-24 10:35:00.541708] TraceType: Summary
CountAll:5 CountFail:6 UsedTimeAvg:23669
[2020-04-24 10:35:00.542081] TraceType: Summary
CountAll:884 CountFail:9 UsedTimeAvg:243509
```

Periodical summary (Log P)

Existing methods do not work well

- Latest parser-based method (Logzip) is sub-optimal
  - **Compression ratio:** worse than LZMA on 13 out of 18 types of logs
  - **Compression speed:** need over 200 days to compress 1 PB logs
- Mismatch between production log features and Logzip design

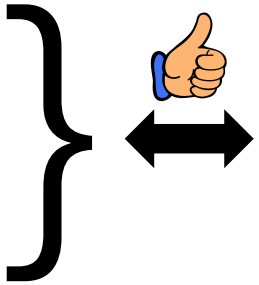
Production log features		**Logzip design**
Large-scale logs		Implement with slow Python libraries
Up to 176 variables per template		Limit to 5 variables per template
Numerical variables take a large part		No specific consideration for compressing numerical variables
…		…

Overview of LogReducer

Production logs features

Large-scale logs

Up to 176 variables per template



LogReducer design

- Better Engineering efforts

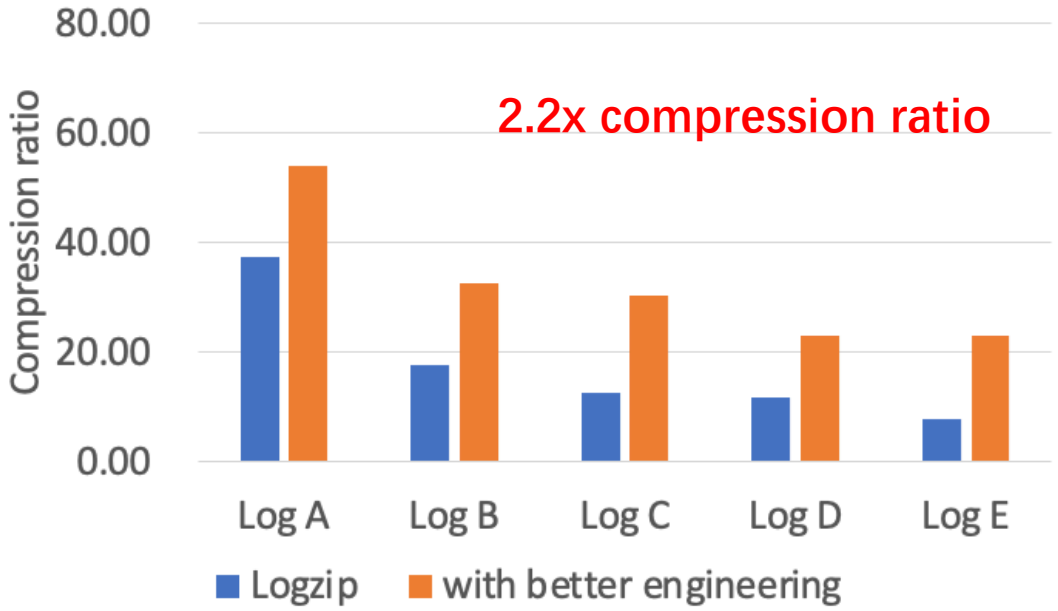
Numerical variables take a large part

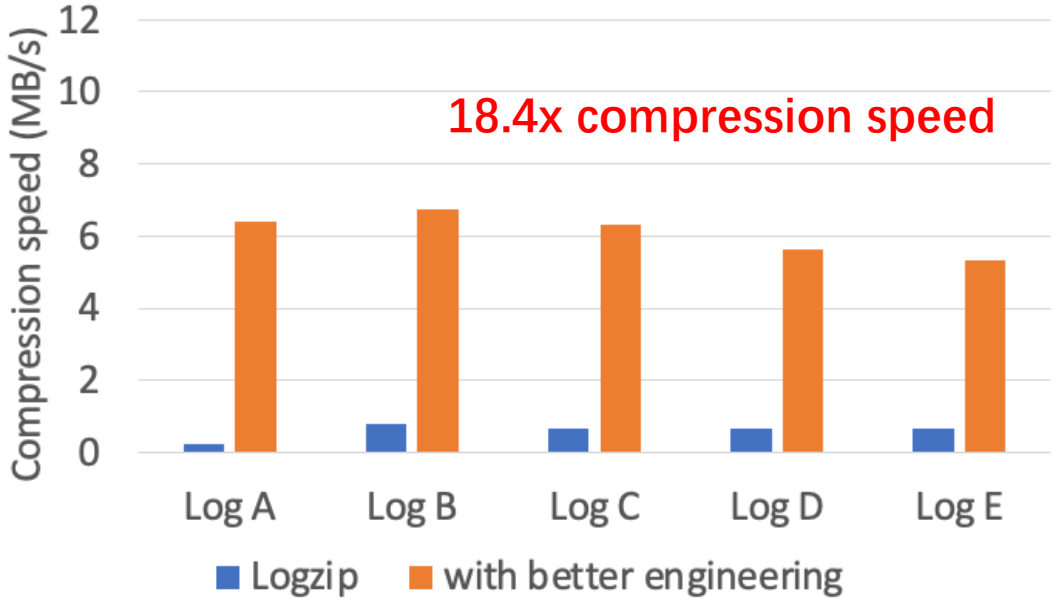
- Numerical variables compression
  - Delta timestamp
  - Correlation identification
  - Elastic encoding

Up to 4x compression ratio and 180x compression speed compared with Logzip

Engineering efforts matter

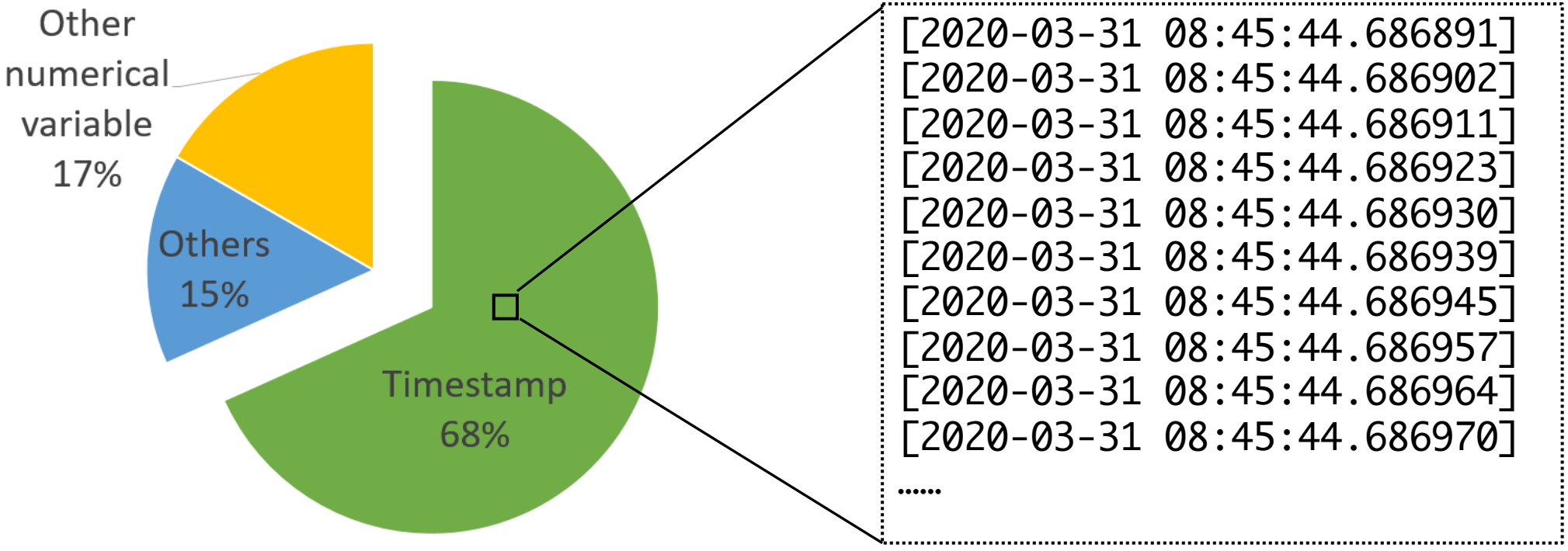
Observation		Engineering efforts
Large-scale logs	→	Efficient implementation with C++
Fewer templates but more variables per template	→	Removing limit on # of variables Pruning parser tree Tuning threshold
A large number of log files	→	Batch processing





Numerical variable compression (1): Delta timestamps

- Observation: Timestamps takes a very large space
  - ➢ Up to 1M log entries per second
  - ➢ Require micro-second level timestamp to debug
- Technique: Differential encoding of time stamps

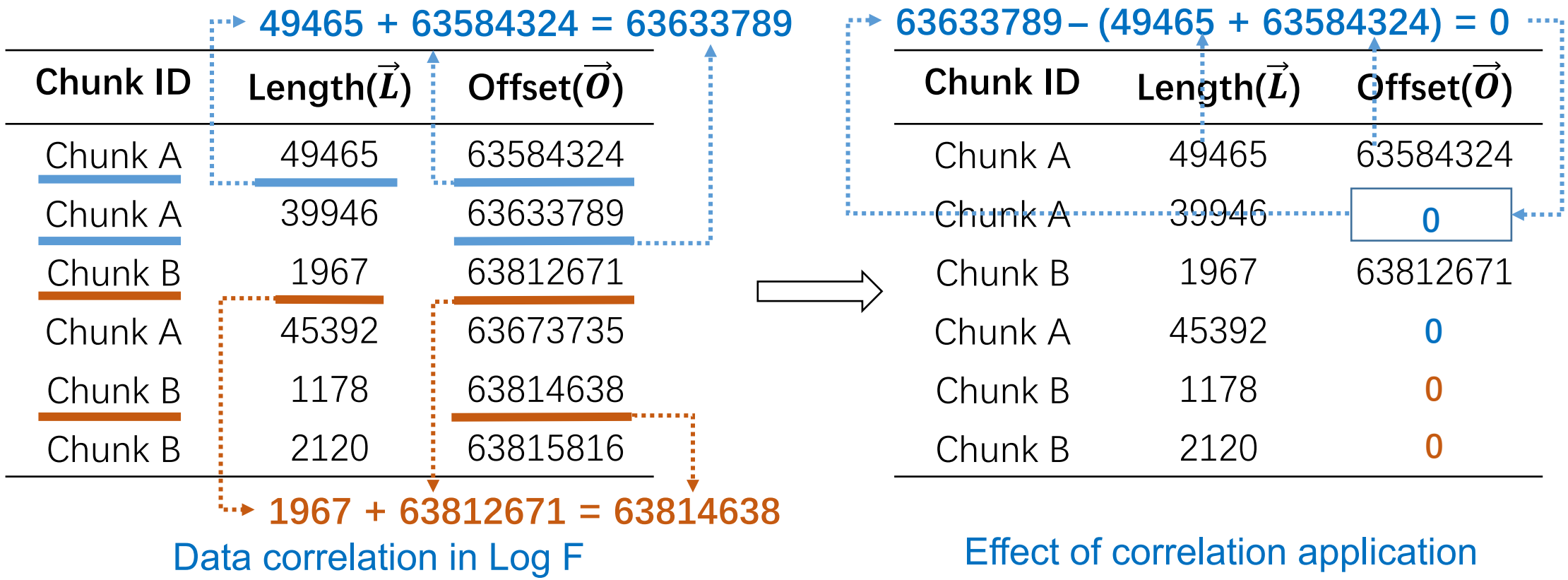


Space consumption of Log R

Micro-second level timestamps

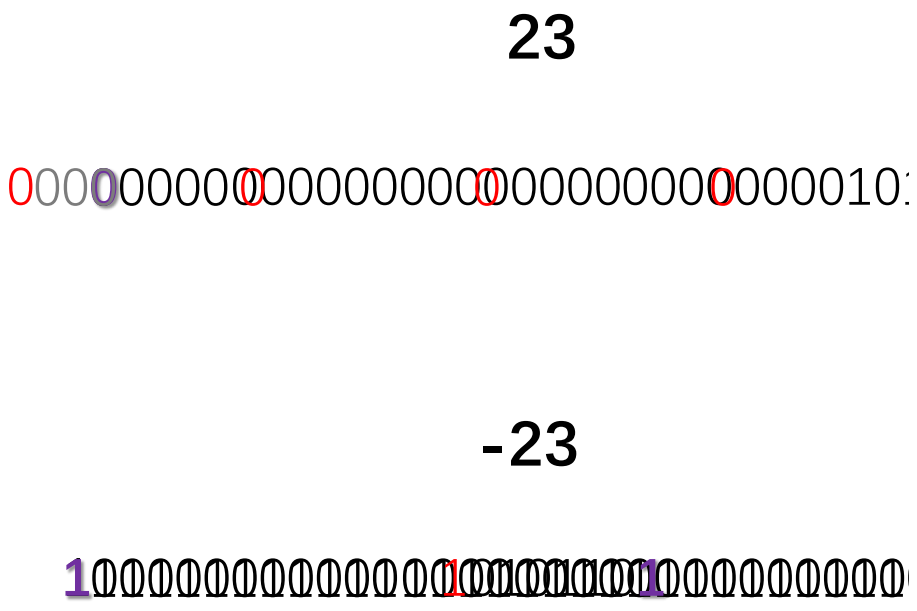
Numerical variable compression (2): Correlation identification

- Observation: Some numerical variables are correlated
- Technique:
  - ➢ Identify correlation in training phase
  - ➢ Apply correlation in compression phase

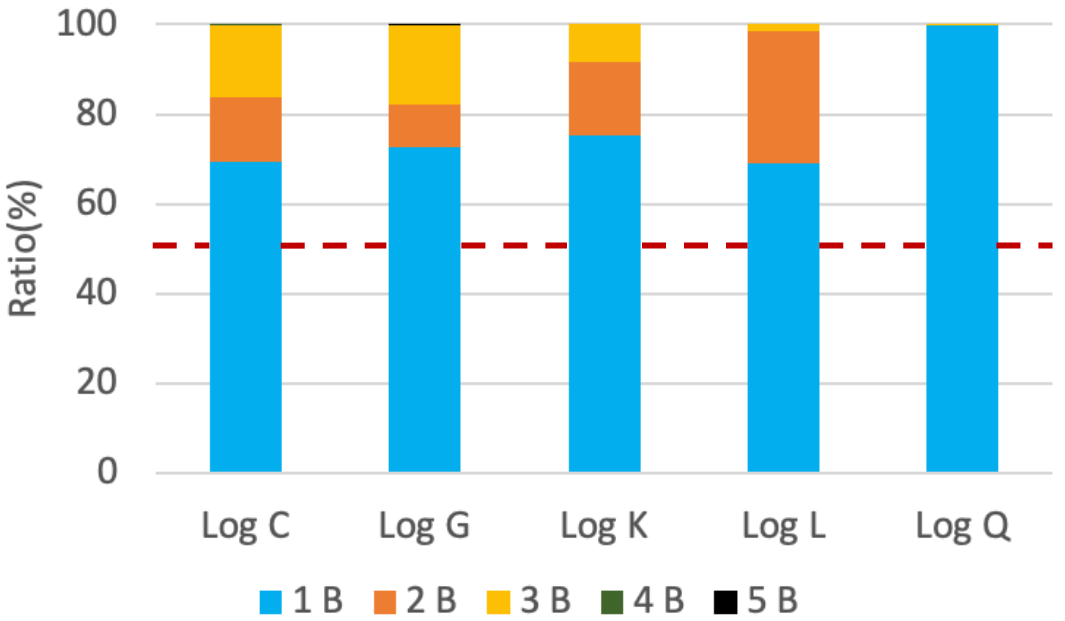


Numerical variable compression (3): Elastic encoding

- Observation: Integers in numerical variables are usually small
- Technique: Use elastic encoding to save space.

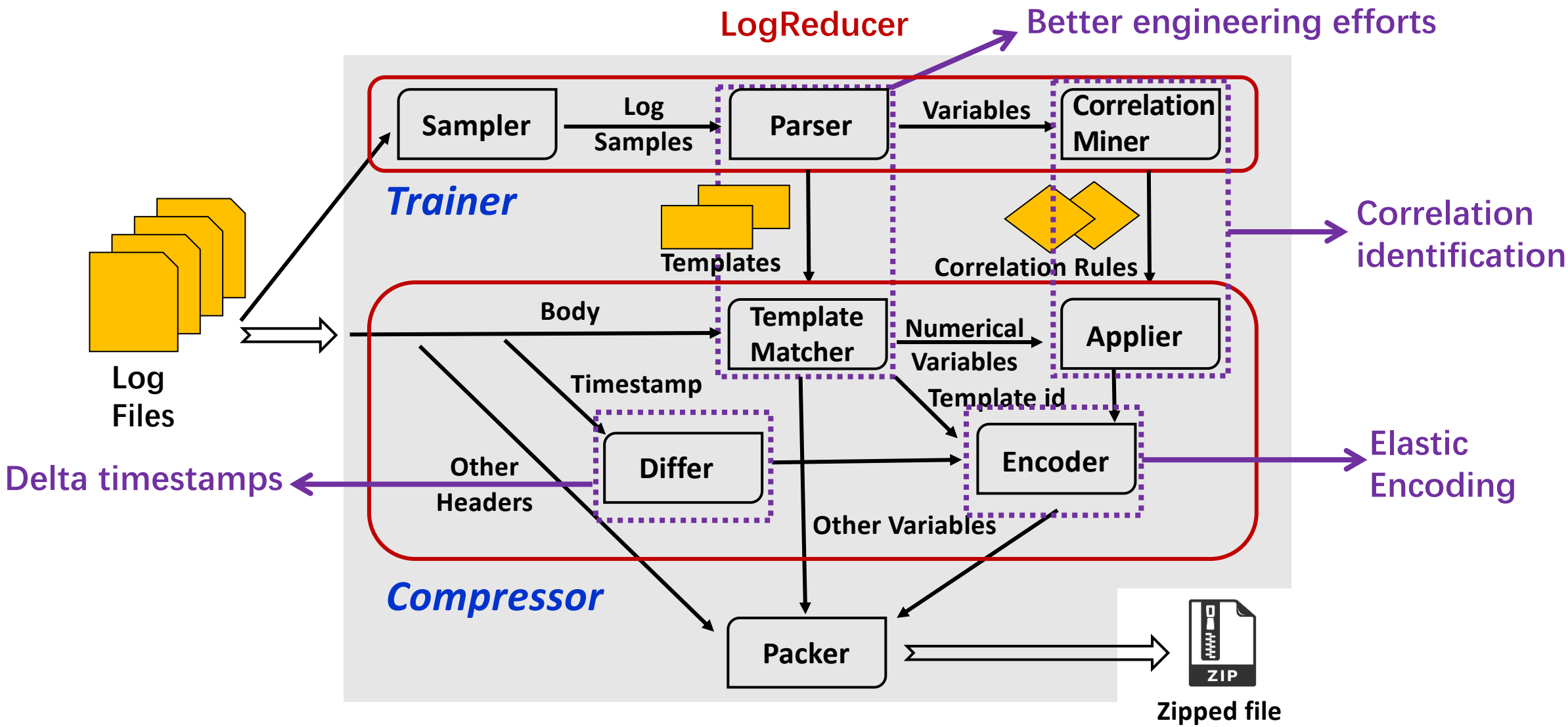


Over 50% need only 1 Byte



How many bytes are needed per integer after encoding

LogReducer Architecture

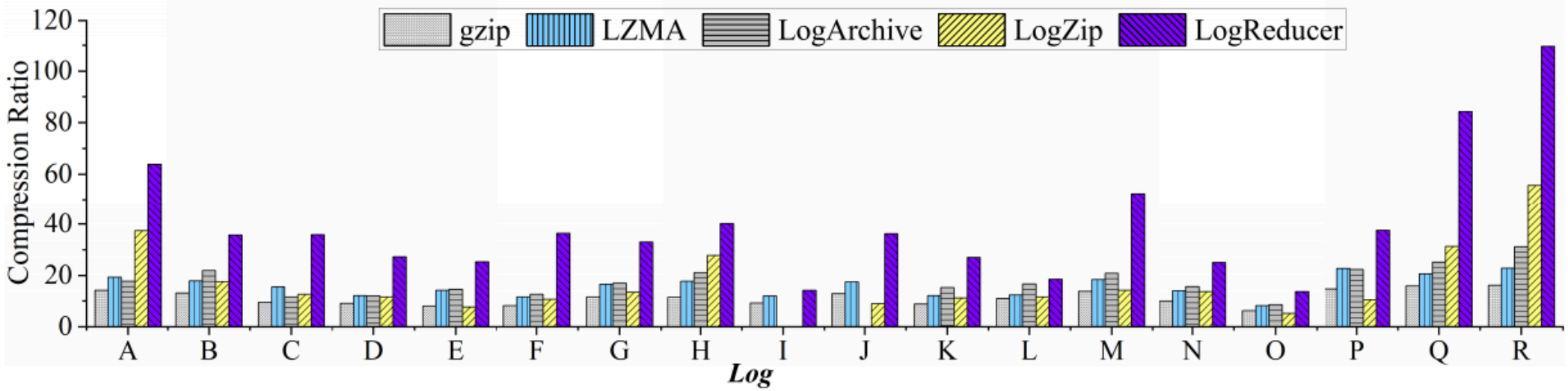


Experiment settings

- TestBed (Linux server)
  - 2x Intel Xeon E5-2682 2.5GHz CPUs
  - 188GB RAM
- Dataset
  - Real-world production log dataset from AliCloud (18 types, 1.76TB in total)
  - Public log dataset (16 types, 77GB in total)
- Baseline
  - General-purpose compression methods: gzip (high-speed) and LZMA(high compression ratio)
  - Log-specific compression methods: LogArchive (bucket-based methods) and Logzip (latest parser-based method)

Compression ratio

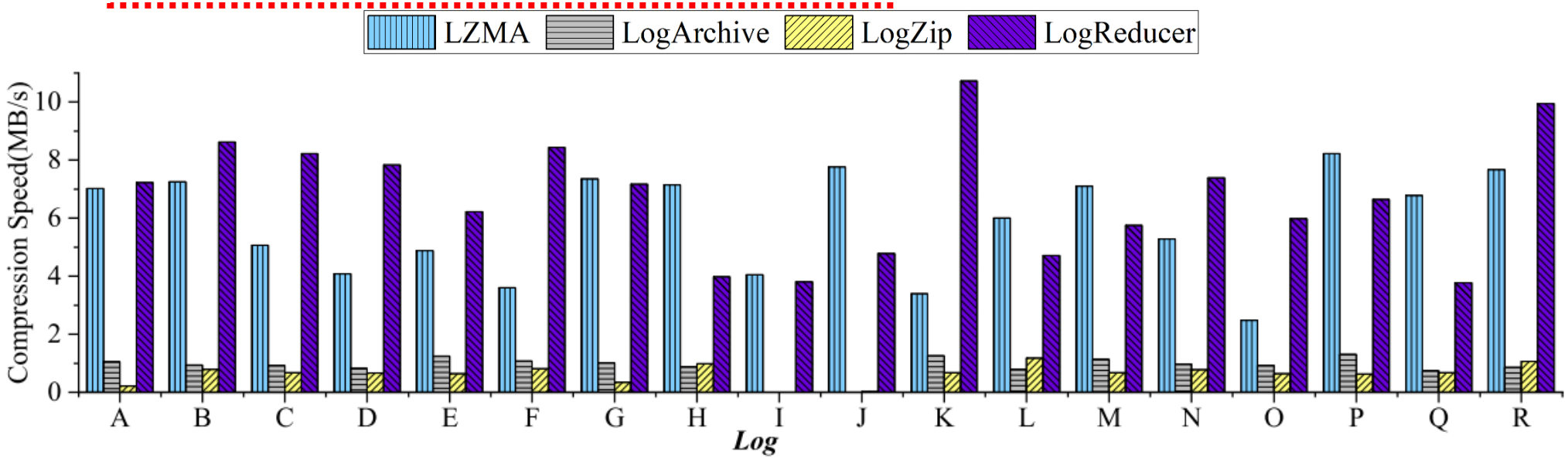
- LogReducer can achieve the highest compression ratio on production logs, it can compress all 1.76TB log dataset into 34.25GB, takes 1.9% space
  - 1.54× - 6.78× compared to gzip
  - 1.19× - 4.80× compared to LZMA
  - 1.11× - 3.60× compared to LogArchive
  - 1.45× - 4.01× compared to Logzip



Compression ratio on production logs

Compression speed

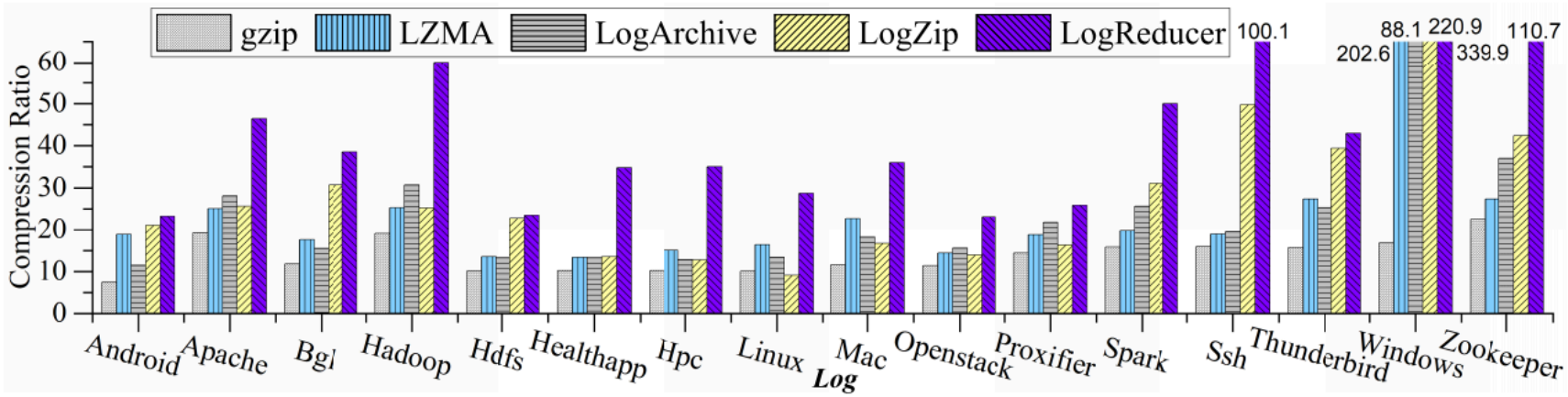
- The compression speed is up to two orders of magnitude higher than Logzip and is comparable to LZMA
  - 0.07× - 0.53× compared to gzip
  - 0.56× - 3.16× compared to LZMA
  - 4.49× - 11.65× compared to LogArchive
  - 4.01× - 182.31× compared to Logzip



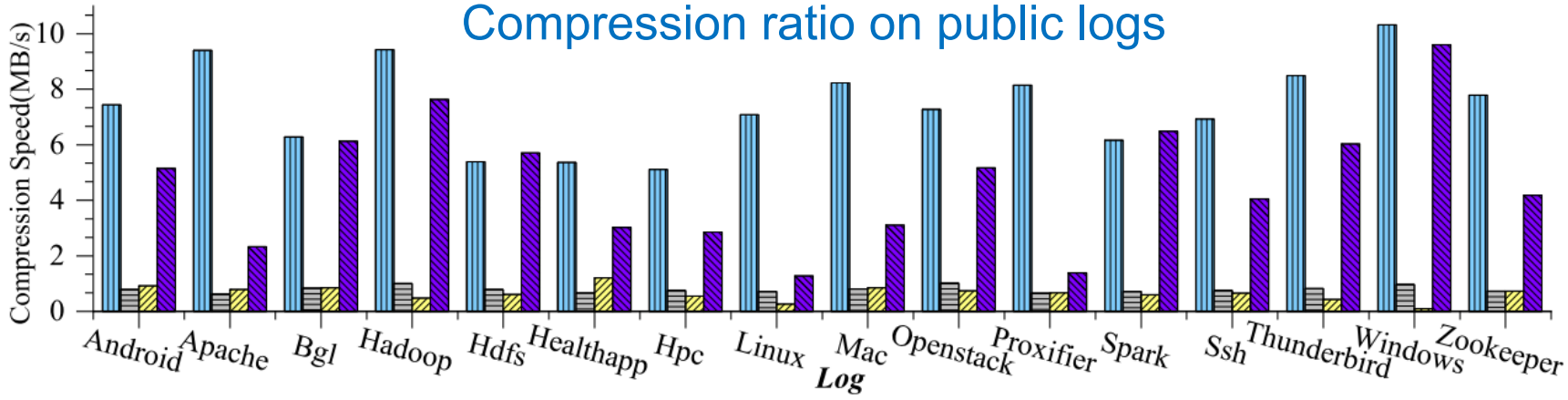
Compression speed on production logs

Evaluation on public dataset

- LogReducer has the highest compression ratio on all public logs
- The compression speed is comparable to LZMA on large logs (over 100MB)



Compression ratio on public logs



Compression speed on public logs

Is it feasible to use parser-based methods on large-scale logs?

Yes

But…

- An efficient implementation is critical to realize its potential
- More opportunities to compress numerical variables
- A significant improvement in compression ratio with a satisfactory speed

Source code of LogReducer can be found at
https://github.com/THUBear-wjy/LogReducer
Samples of production logs can be found at
https://github.com/THUBear-wjy/openSample

Thank you

Q&A