Diagnosing applications' I/O behavior through system call observability

Tânia Esteves, Ricardo Macedo, Rui Oliveira and João Paulo

INESC TEC & University of Minho

5th Workshop on Data-Centric Dependability and Security (DCDS'23)

INESCTEC

Universidade do Minho

FCT Fundação para a Ciência e a Tecnologia

Diagnosing applications storage I/O

Problem

- Applications often exhibit inefficient or erroneous I/O behaviors
  - Costly access patterns
    - Small-sized I/O requests or random accesses
  - I/O contention
    - Concurrent requests to shared resources
  - Erroneous usage of I/O calls
    - Accessing wrong file offsets

Diagnosing applications storage I/O

Problem

- Applications often exhibit inefficient or erroneous I/O behaviors
  - Costly access patterns
    - Small-sized I/O requests or random accesses
  - I/O contention
    - Concurrent requests to shared resources

      RocksDB
  - Erroneous usage of I/O calls
    - Accessing wrong file offsets

Diagnosing applications storage I/O

Problem

◉ Applications often exhibit inefficient or erroneous I/O behaviors

- Costly access patterns
  - Small-sized I/O requests or random accesses
- I/O contention
  - Concurrent requests to shared resources

RocksDB

- Erroneous usage of I/O calls
  - Accessing wrong file offsets

Fluent Bit

Diagnosing applications storage I/O

Problem

- Applications often exhibit inefficient or erroneous I/O behaviors
  - Costly access patterns
    - Small-sized I/O requests or random accesses
  - I/O contention
    - Concurrent requests to shared resources RocksDB
  - Erroneous usage of I/O calls
    - Accessing wrong file offsets Fluent Bit

Can compromise the performance, correctness and dependability of applications!

Diagnosing applications storage I/O

Current approaches

◉ Source code instrumentation

- ▸ Intrusive
  - Source code may be unavailable
- ▸ Complex & time-consuming
  - Large codebases to understand and modify

✓ Provides accurate information about applications' actions

Diagnosing applications storage I/O

Current approaches

- ◉ Source code instrumentation
  - ▸ Intrusive
    - Source code may be unavailable
  - ▸ Complex & time-consuming
    - Large codebases to understand and modify

✓ Provides accurate information about applications' actions

RocksDB 440K LoC

SILK [1]

Fluent Bit 1M LoC

[1] BALMAU, Oana, et al. SILK: Preventing Latency Spikes in Log-Structured Merge Key-Value Stores. In: USENIX Annual Technical Conference. 2019. p. 753-766.

Diagnosing applications storage I/O

Current approach

◉ Tracing

✓ Transparent to the application

- High overhead vs data loss
  - High overhead can camouflage erroneous behaviors
- Lack of analysis pipelines
  - Large number of events to analyze manually
- Lack of flexibility
  - Solutions designed for rigid analysis scenarios

Diagnosing applications storage I/O

Current approach

◉ Tracing

✓ Transparent to the application

- High overhead vs data loss
  - High overhead can camouflage erroneous behaviors
- Lack of analysis pipelines
  - Large number of events to analyze manually
- Lack of flexibility
  - Solutions designed for rigid analysis scenarios

RocksDB
Requires a benchmark that generates >500M system calls

Diagnosing applications storage I/O

Current approach

◉ Tracing

✓ Transparent to the application

- High overhead vs data loss
  - High overhead can camouflage erroneous behaviors
- Lack of analysis pipelines
  - Large number of events to analyze manually
- Lack of flexibility
  - Solutions designed for rigid analysis scenarios

RocksDB
Requires a benchmark that generates >500M system calls

Fluent Bit
Requires accessed offsets and inodes

DIO

This work

◉ A generic tool for observing and diagnosing I/O interactions between applications and in-kernel POSIX storage systems

- Transparency
- Comprehensive and flexible tracing
- Practical and timely analysis
- Data querying and correlation
- Customized visualization

DIO

This work

◉ A generic tool for observing and diagnosing I/O interactions between applications and in-kernel POSIX storage systems

- Transparency
- Comprehensive and flexible tracing
- Practical and timely analysis
- Data querying and correlation
- Customized visualization

✓A new eBPF-based tracer

DIO

This work

- A generic tool for observing and diagnosing I/O interactions between applications and in-kernel POSIX storage systems
  - Transparency
  - Comprehensive and flexible tracing
  - Practical and timely analysis
  - Data querying and correlation
  - Customized visualization

✓A new eBPF-based tracer

✓ Contextual information from kernel & Filters

DIO

This work

◉ A generic tool for observing and diagnosing I/O interactions between applications and in-kernel POSIX storage systems

- Transparency
- Comprehensive and flexible tracing
- Practical and timely analysis
- Data querying and correlation
- Customized visualization

✓A new eBPF-based tracer

✓ Contextual information from kernel & Filters

✓ Data sent directly to a remote analysis pipeline

DIO

This work

- A generic tool for observing and diagnosing I/O interactions between applications and in-kernel POSIX storage systems

▸ Transparency	✓A new eBPF-based tracer
▸ Comprehensive and flexible tracing	✓ Contextual information from kernel & Filters
▸ Practical and timely analysis	✓ Data sent directly to a remote analysis pipeline
▸ Data querying and correlation	✓ Query, filter and correlate captured data
▸ Customized visualization	

DIO

This work

◉ A generic tool for observing and diagnosing I/O interactions between applications and in-kernel POSIX storage systems

▸ Transparency	✓A new eBPF-based tracer
▸ Comprehensive and flexible tracing	✓ Contextual information from kernel & Filters
▸ Practical and timely analysis	✓ Data sent directly to a remote analysis pipeline
▸ Data querying and correlation	✓ Query, filter and correlate captured data
▸ Customized visualization	✓ Explore data and build customized visualizations

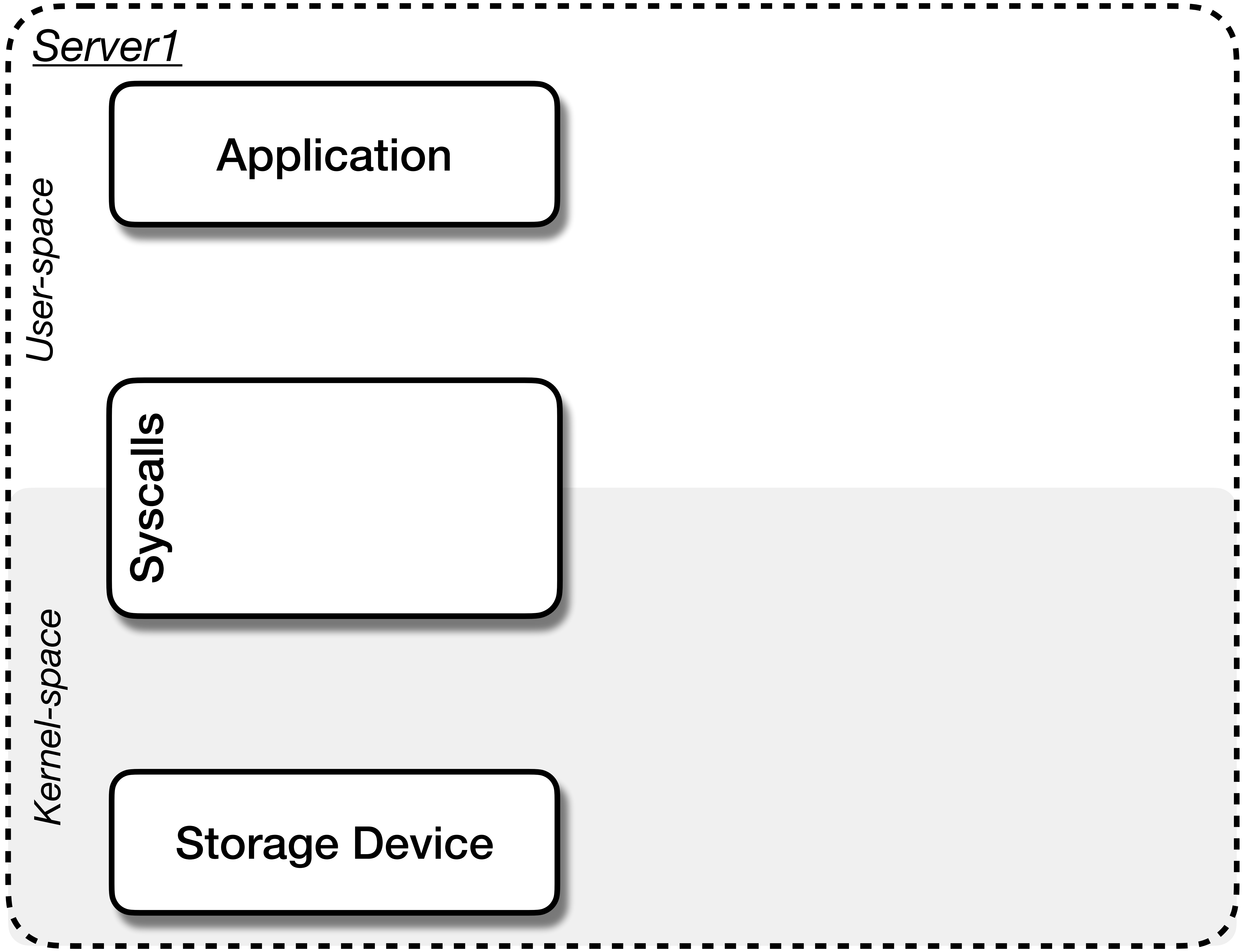
DIO

System overview

■ DIO’s components → DIO main flow ⇢ App flow

DIO

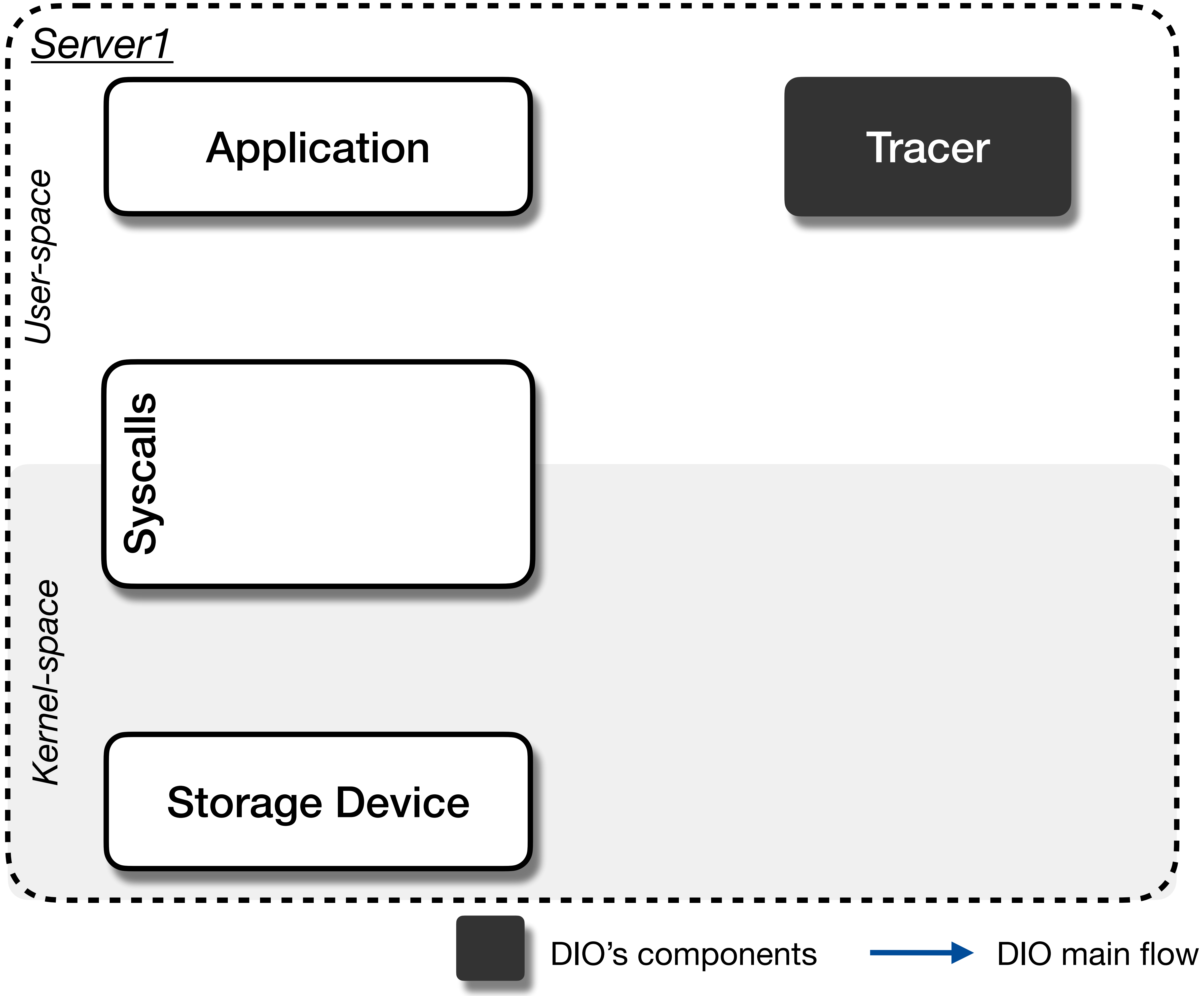
System overview



⬛ DIO's components → DIO main flow ⇢ App flow

DIO

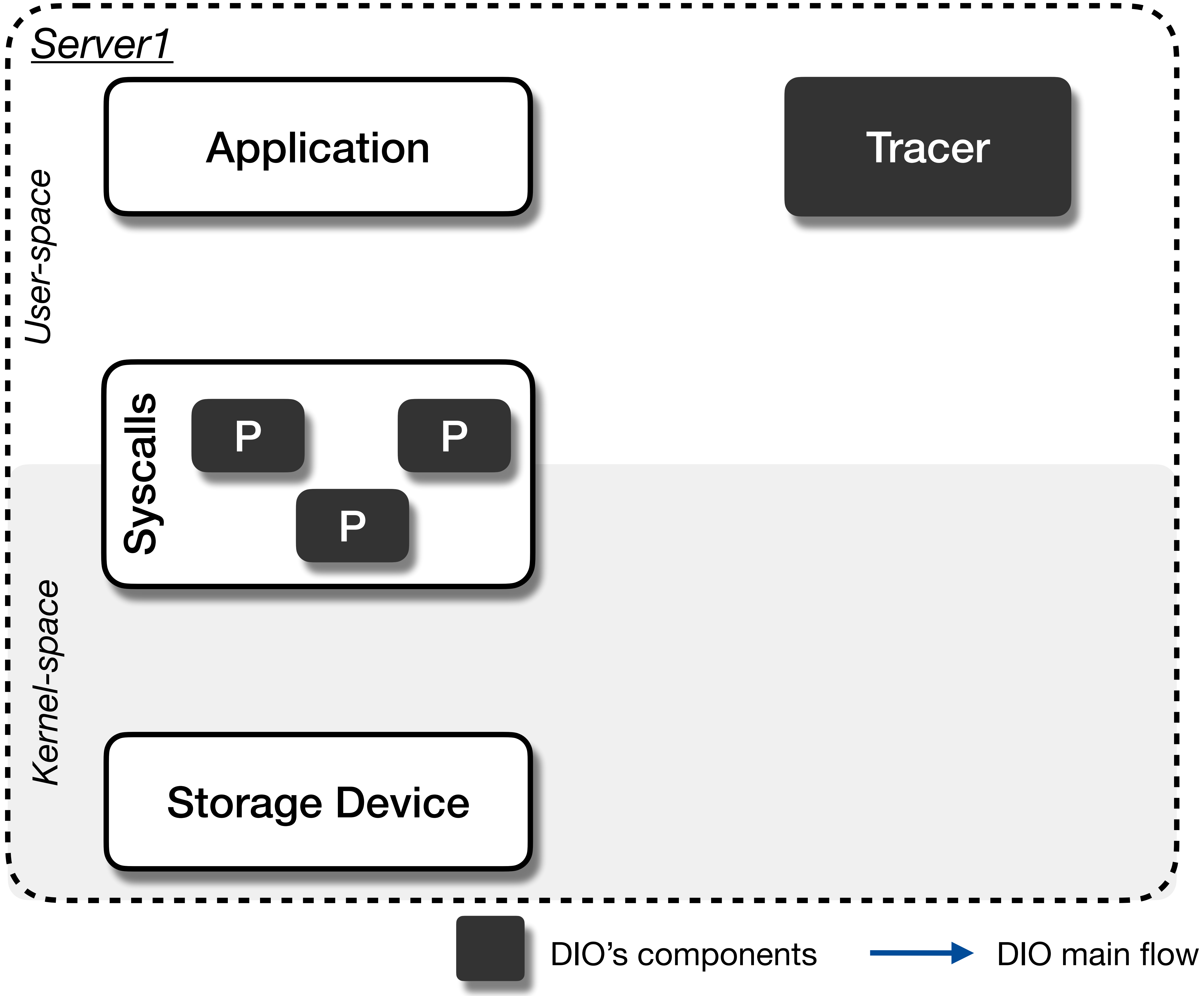
System overview



DIO's tracer runs along the targeted application, intercepting its syscalls

DIO

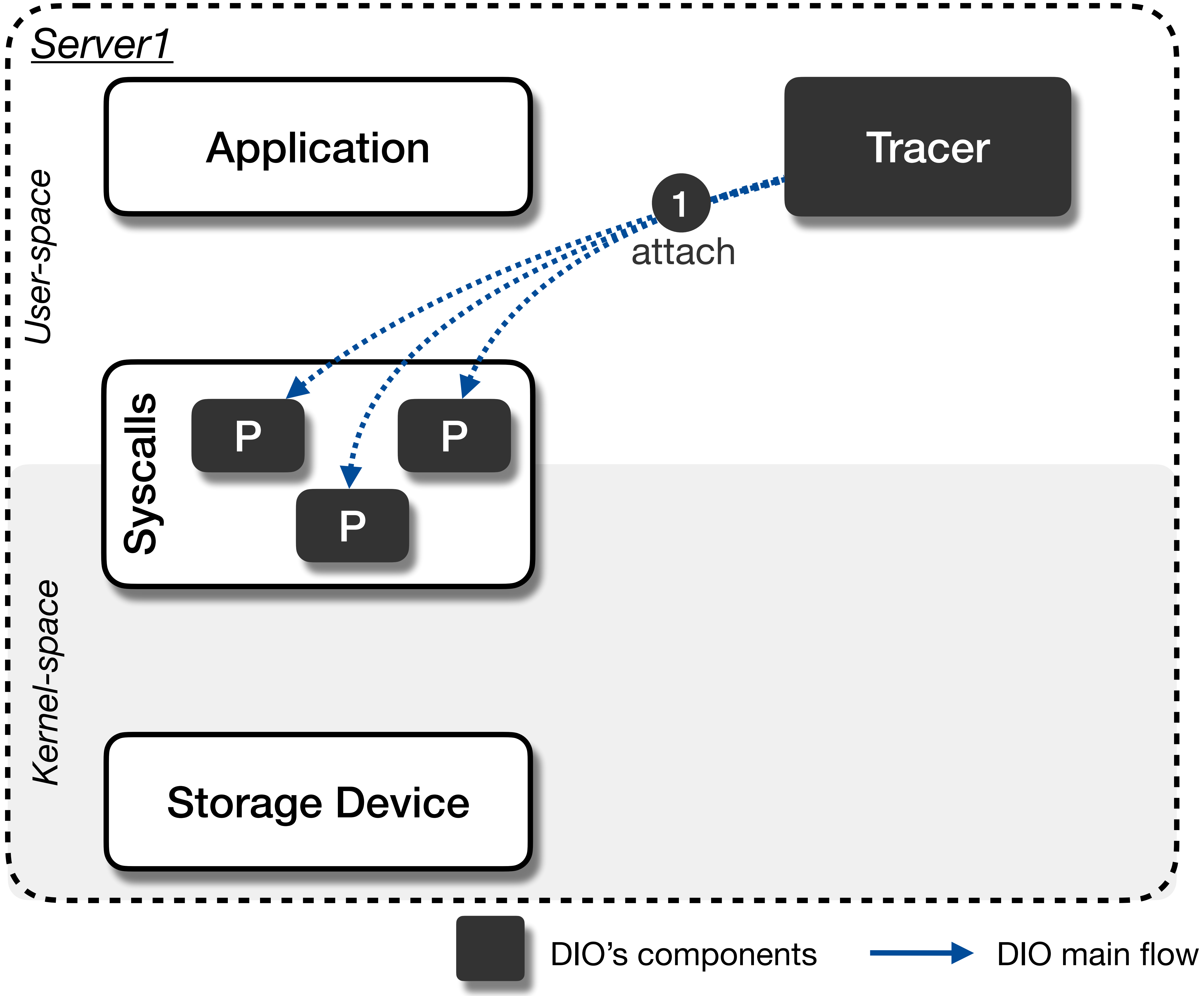
System overview



DIO's tracer runs along the targeted application, intercepting its syscalls

DIO

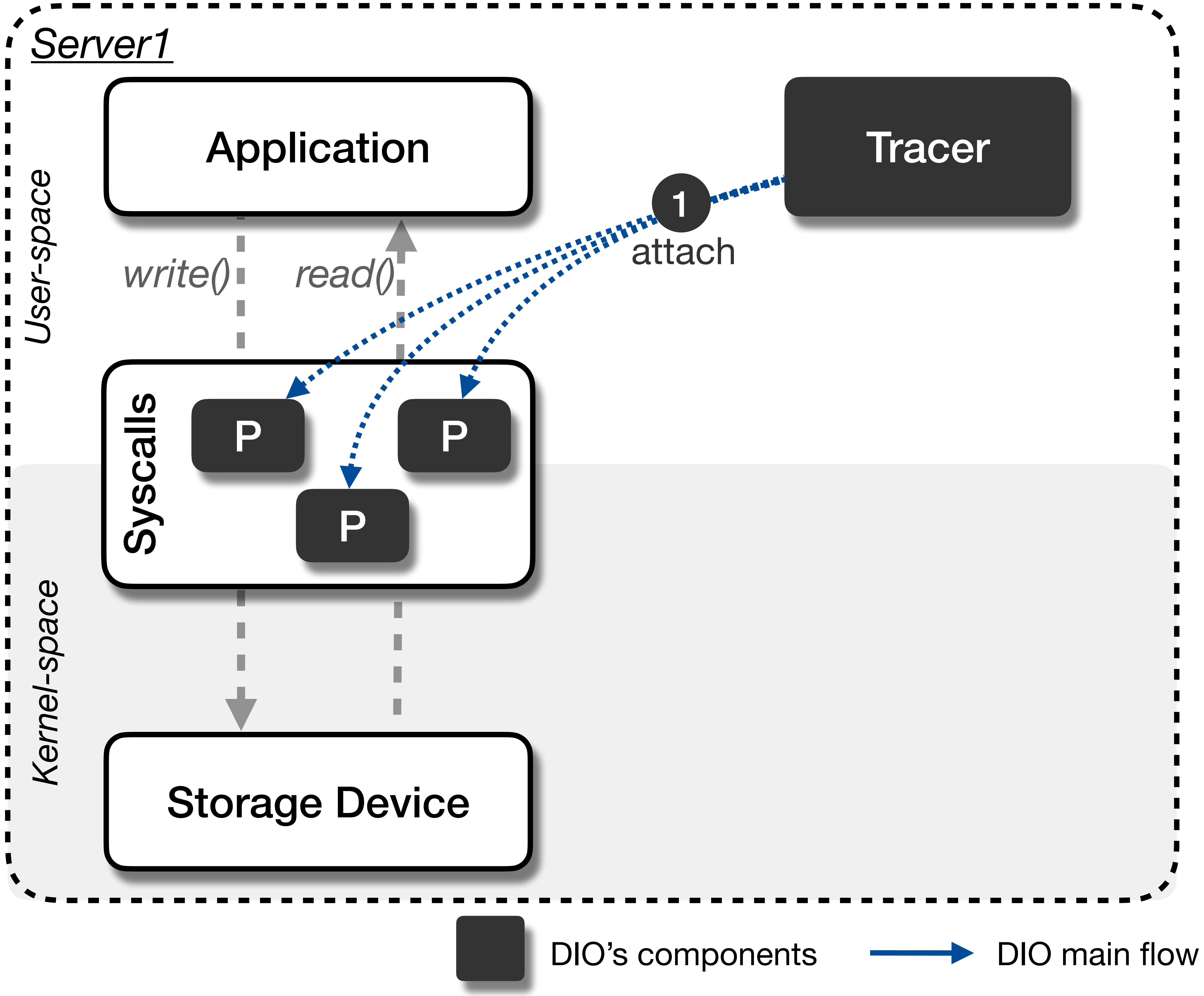
System overview



DIO's tracer runs along the targeted application, intercepting its syscalls

DIO

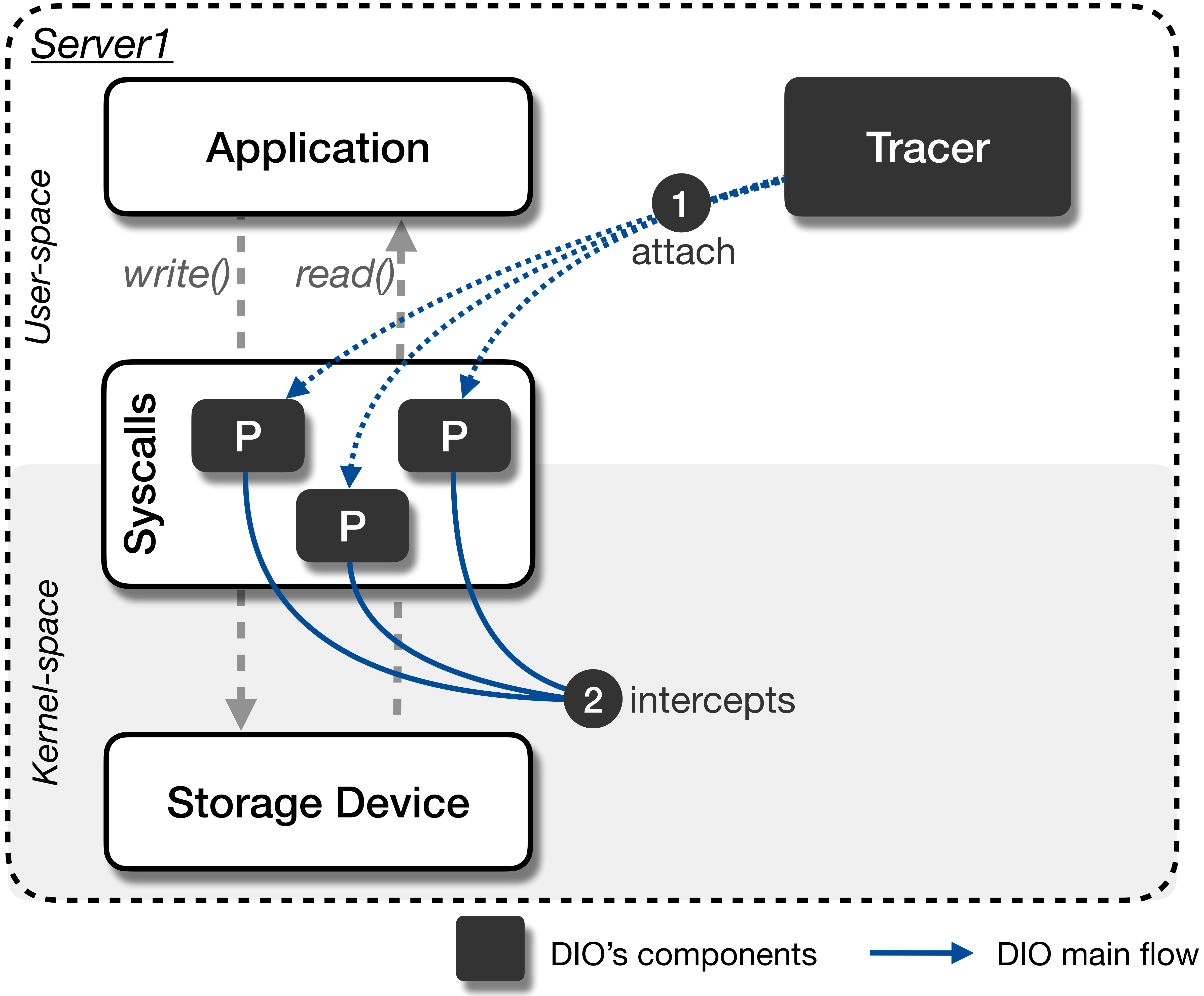
System overview



DIO's tracer runs along the targeted application, intercepting its syscalls

DIO

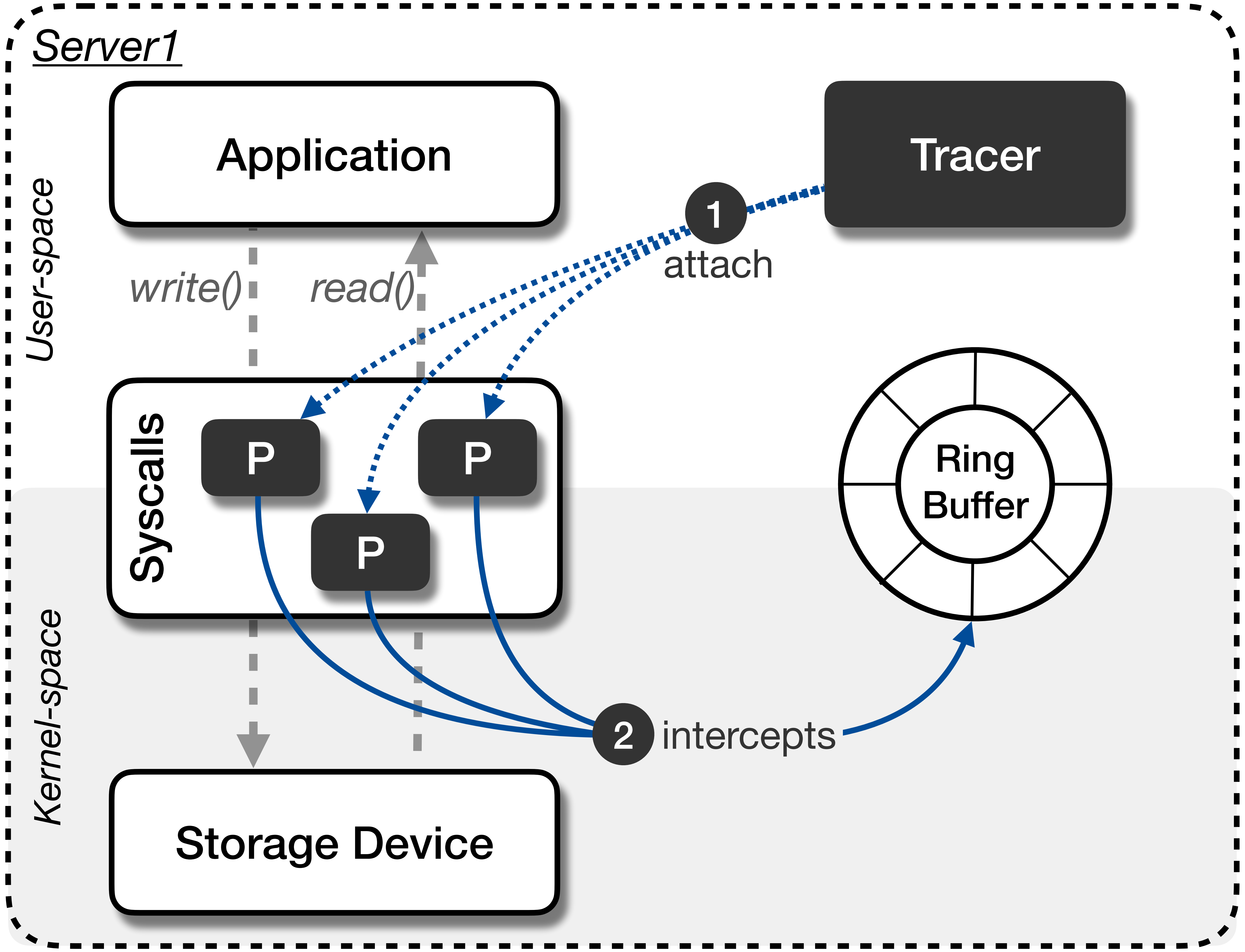
System overview



DIO's tracer runs along the targeted application, intercepting its syscalls

DIO

System overview

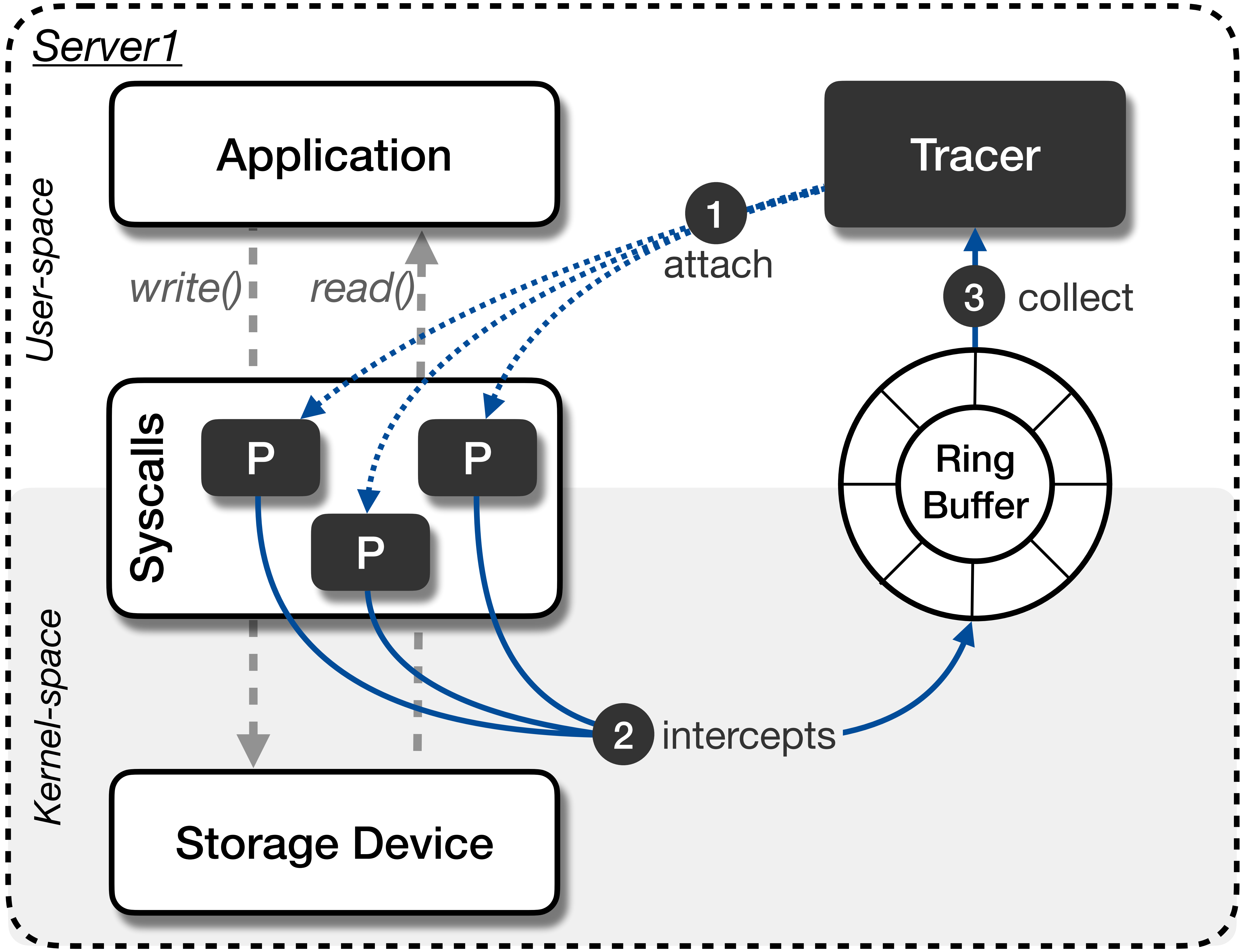


■ DIO's components → DIO main flow - - -> App flow

DIO's tracer runs along the targeted application, intercepting its syscalls

DIO

System overview

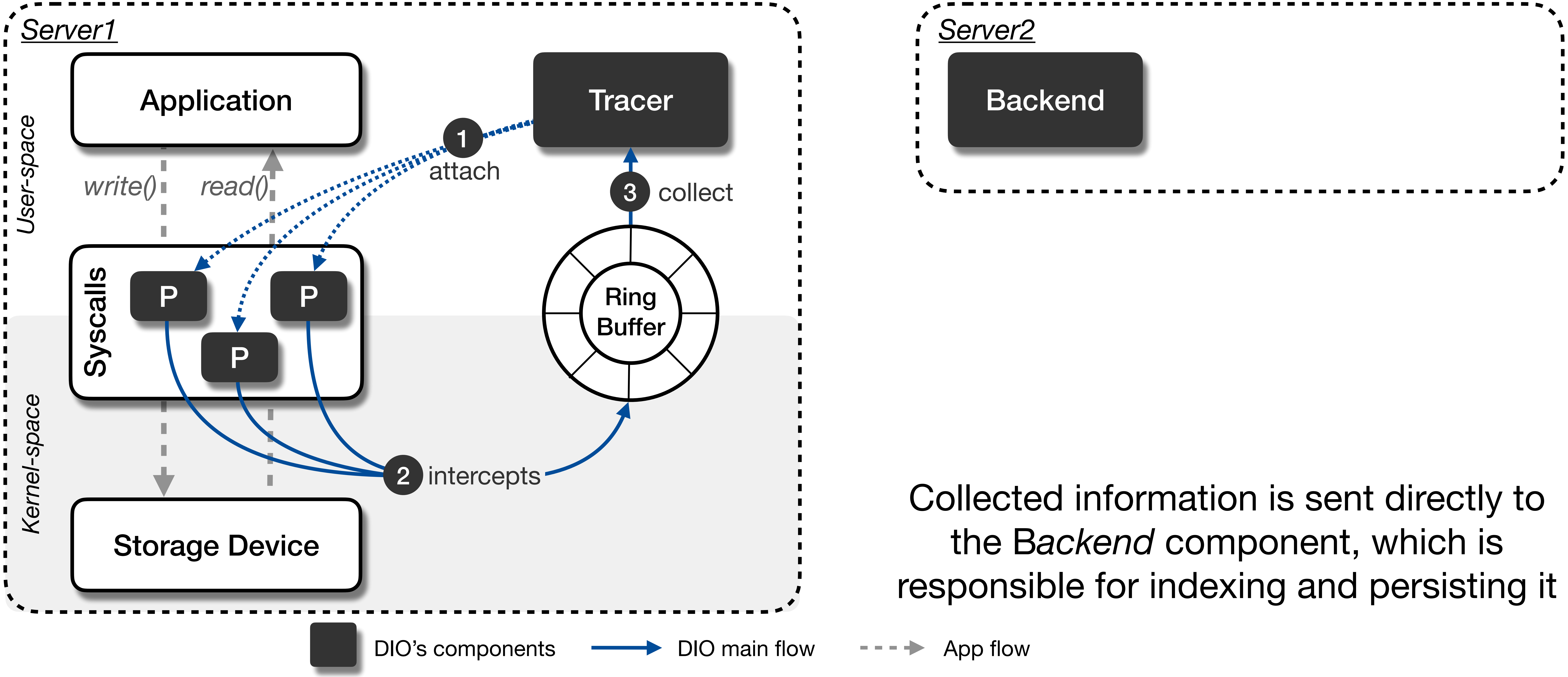


■ DIO's components ⟶ DIO main flow - - -▸ App flow

DIO's tracer runs along the targeted application, intercepting its syscalls

DIO

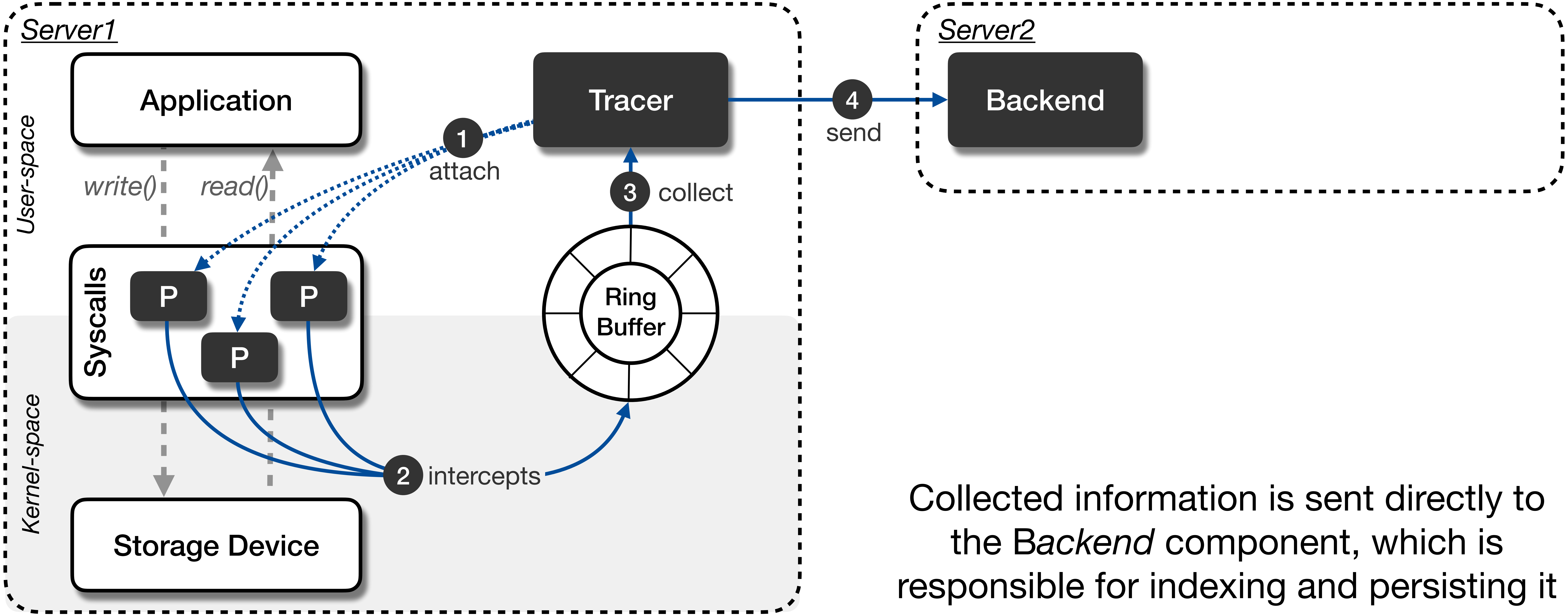
System overview



Collected information is sent directly to the *Backend* component, which is responsible for indexing and persisting it

DIO

System overview

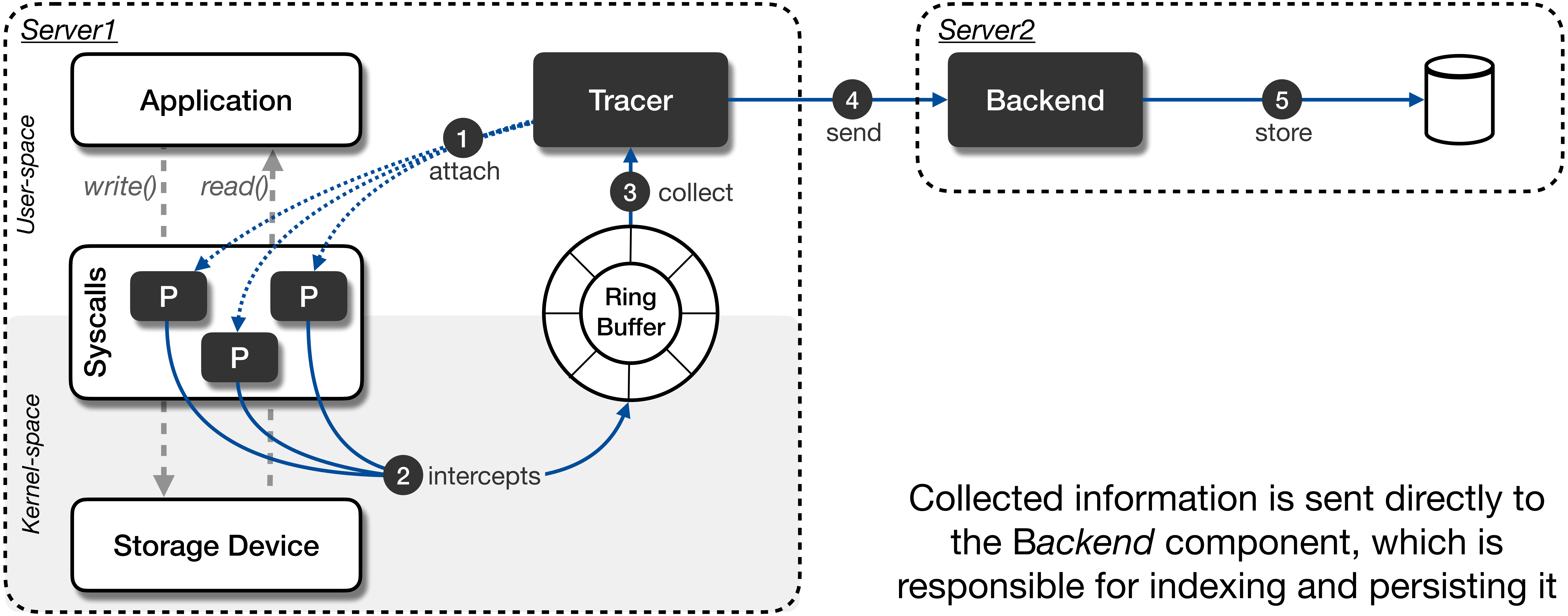


Collected information is sent directly to the *Backend* component, which is responsible for indexing and persisting it

■ DIO's components ⟶ DIO main flow - - -▸ App flow

DIO

System overview

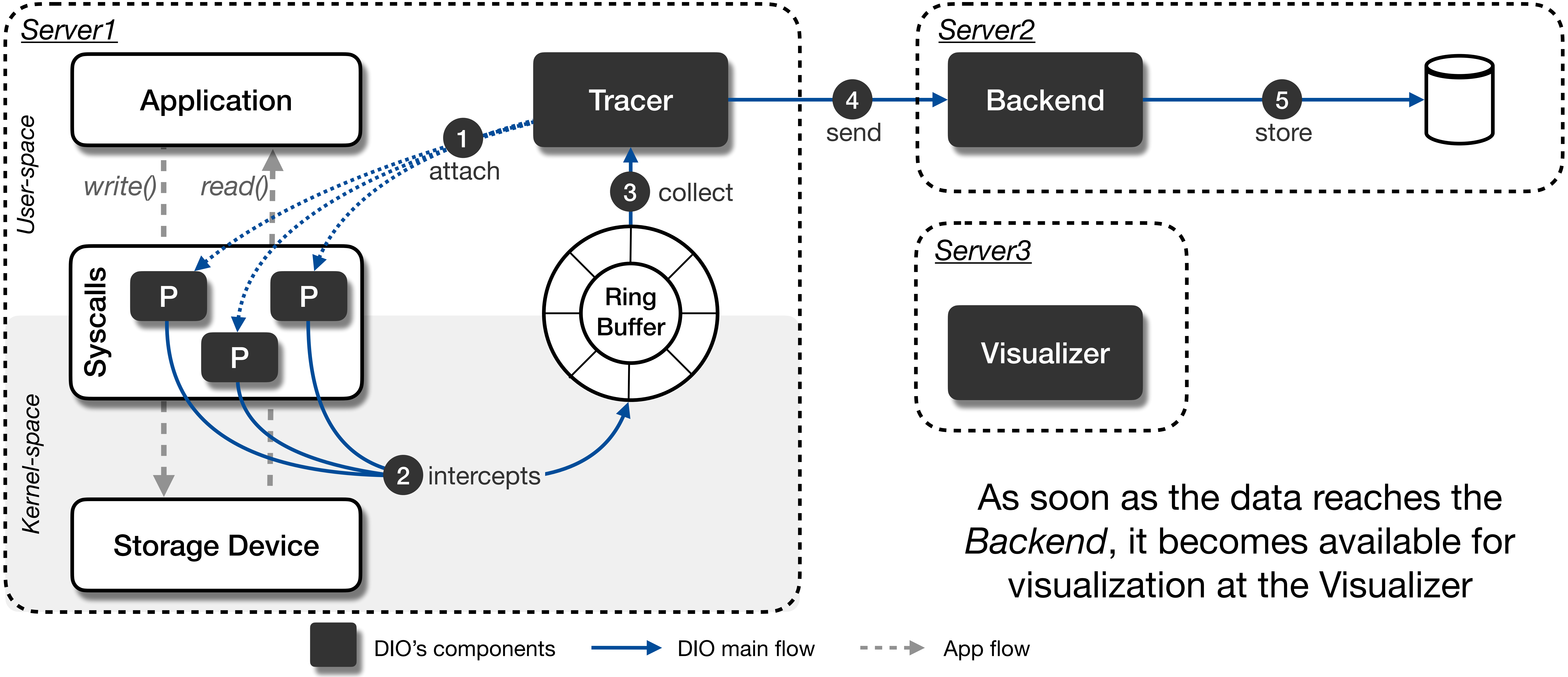


Collected information is sent directly to the *Backend* component, which is responsible for indexing and persisting it

■ DIO's components → DIO main flow - - -→ App flow

DIO

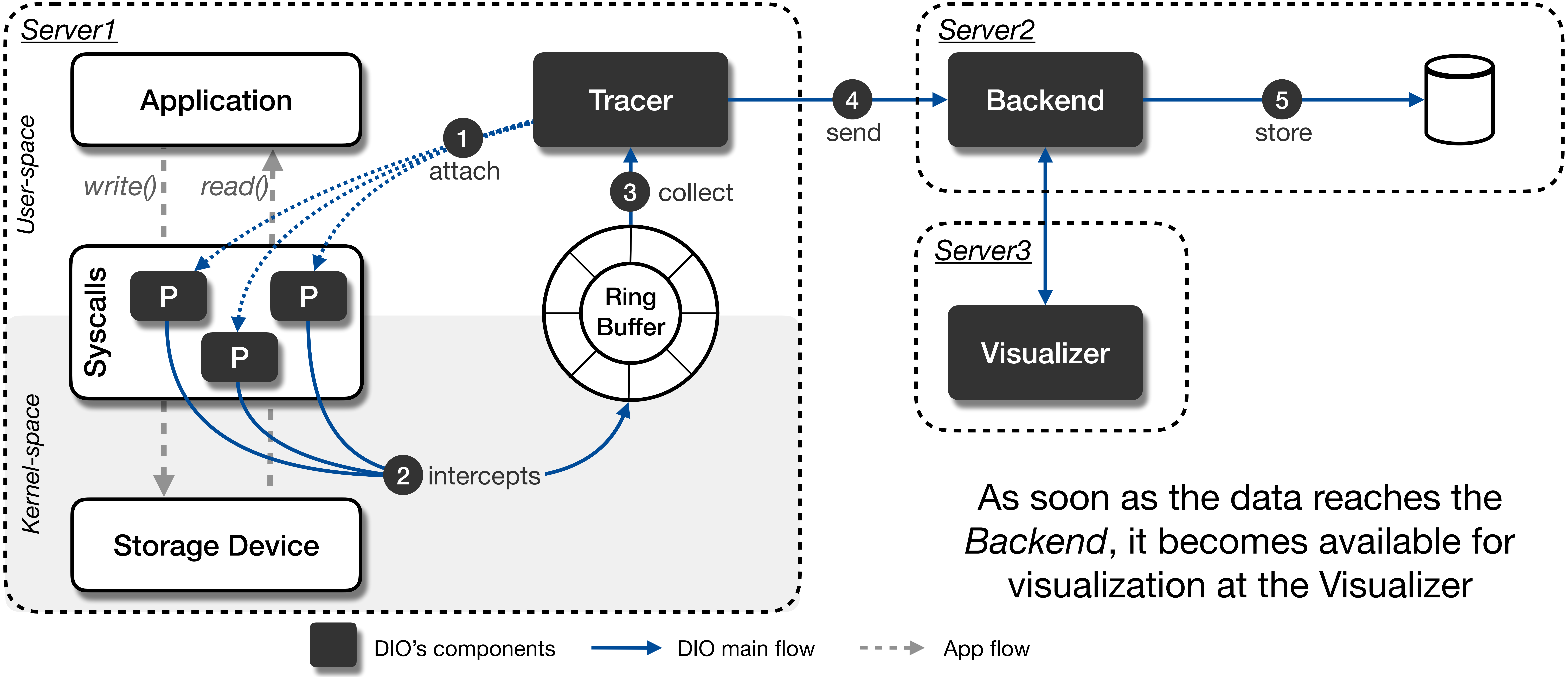
System overview



As soon as the data reaches the *Backend*, it becomes available for visualization at the Visualizer

DIO

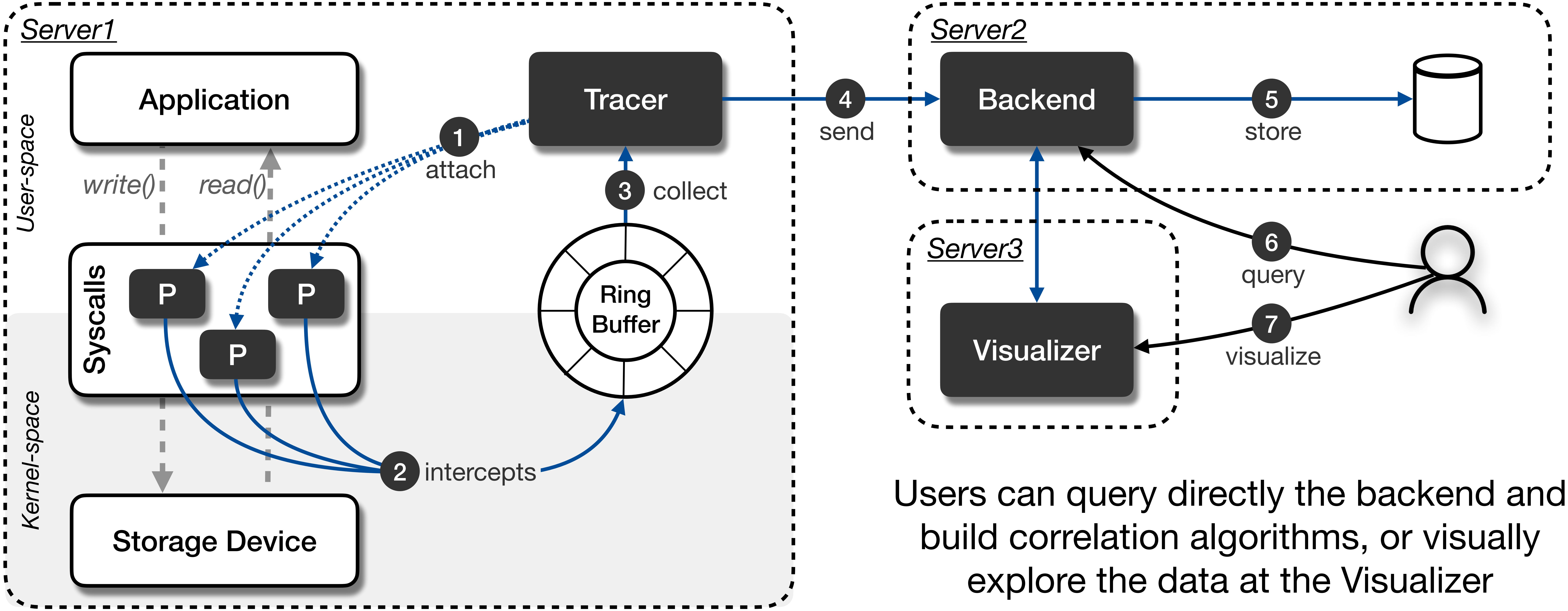
System overview



As soon as the data reaches the *Backend*, it becomes available for visualization at the Visualizer

DIO

System overview



Users can query directly the backend and build correlation algorithms, or visually explore the data at the Visualizer

■ DIO's components → DIO main flow - - → App flow

DIO

Implementation

- ◉ **Tracer**
  - ‣ Uses eBPF technology
  - ‣ Currently supports 42 storage-related system calls
  - ‣ Implemented in ≈8K LoC (restricted C & Go)

- ◉ **Backend & Visualizer**
  - ‣ Elasticsearch and Kibana (v8.5.2)
  - ‣ File path correlation algorithm
    - Correlates file descriptors with their corresponding file paths
  - ‣ Pre-defined dashboards and visualizations

Evaluation

Goals

- Showcase how DIO eases the observation of storage issues
  - Identifying erroneous actions that lead to data loss
  - Finding the root cause of performance anomalies

- Understand the performance impact induced by DIO
  - Comparison with two state-of-the-art tracers
    - Unlike other tracers, DIO collects, parses, and forwards the traced information to the analysis pipeline while imposing reduced performance overhead

Evaluation

Goals

- Showcase how DIO eases the observation of storage issues
  - Identifying erroneous actions that lead to data loss
  - Finding the root cause of performance anomalies

Fluent Bit

- Understand the performance impact induced by DIO
  - Comparison with two state-of-the-art tracers
    - Unlike other tracers, DIO collects, parses, and forwards the traced information to the analysis pipeline while imposing reduced performance overhead

Evaluation

Goals

- Showcase how DIO eases the observation of storage issues
  - Identifying erroneous actions that lead to data loss Fluent Bit
  - Finding the root cause of performance anomalies RocksDB

- Understand the performance impact induced by DIO
  - Comparison with two state-of-the-art tracers
    - Unlike other tracers, DIO collects, parses, and forwards the traced information to the analysis pipeline while imposing reduced performance overhead

Evaluation

Goals

- Showcase how DIO eases the observation of storage issues
  - Identifying erroneous actions that lead to data loss
  - Finding the root cause of performance anomalies

Fluent Bit

RocksDB

- Understand the performance impact induced by DIO
  - Comparison with two state-of-the-art tracers
    - Unlike other tracers, DIO collects, parses, and forwards the traced information to the analysis pipeline while imposing reduced performance overhead

Strace

Evaluation

Goals

- Showcase how DIO eases the observation of storage issues
  - Identifying erroneous actions that lead to data loss
  - Finding the root cause of performance anomalies

Fluent Bit

RocksDB

- Understand the performance impact induced by DIO
  - Comparison with two state-of-the-art tracers
    - Unlike other tracers, DIO collects, parses, and forwards the traced information to the analysis pipeline while imposing reduced performance overhead

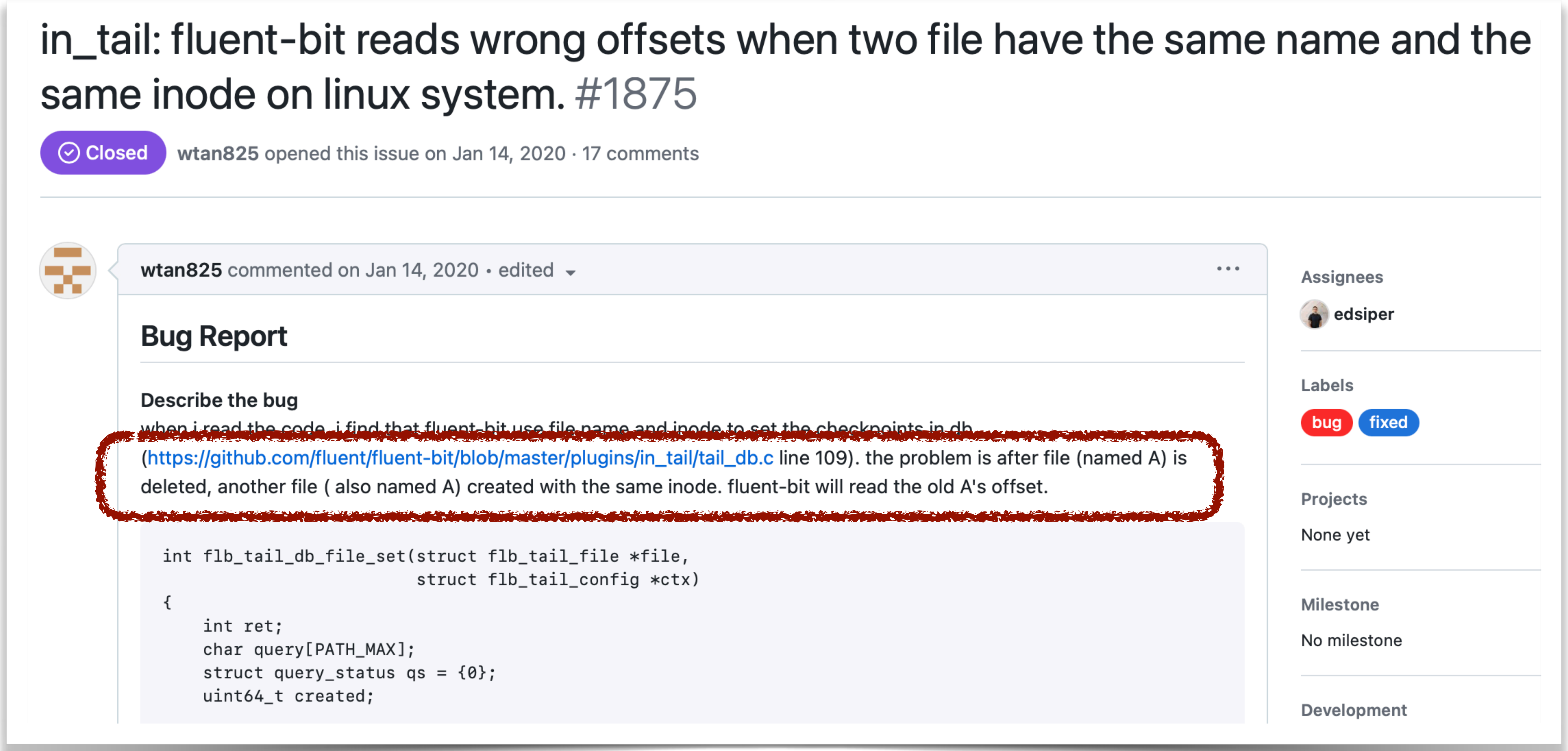
Strace

Sysdig

Evaluation - Fluent Bit

Identifying erroneous actions that lead to data loss

- ◉ **Fluent Bit:** a high-performance logging and metrics processor and forwarder
- ◉ **Problem:** clients observe data loss when using Fluent Bit's tail input plugin (v1.4.0)



in_tail: fluent-bit reads wrong offsets when two file have the same name and the same inode on linux system. #1875

Closed wtan825 opened this issue on Jan 14, 2020 · 17 comments

wtan825 commented on Jan 14, 2020 • edited

Bug Report

Describe the bug

when i read the code, i find that fluent-bit use file name and inode to set the checkpoints in db (https://github.com/fluent/fluent-bit/blob/master/plugins/in_tail/tail_db.c line 109). the problem is after file (named A) is deleted, another file (also named A) created with the same inode. fluent-bit will read the old A's offset.

```
int flb_tail_db_file_set(struct flb_tail_file *file,
                         struct flb_tail_config *ctx)
{
    int ret;
    char query[PATH_MAX];
    struct query_status qs = {0};
    uint64_t created;
```

Assignees

edsiper

Labels

bug fixed

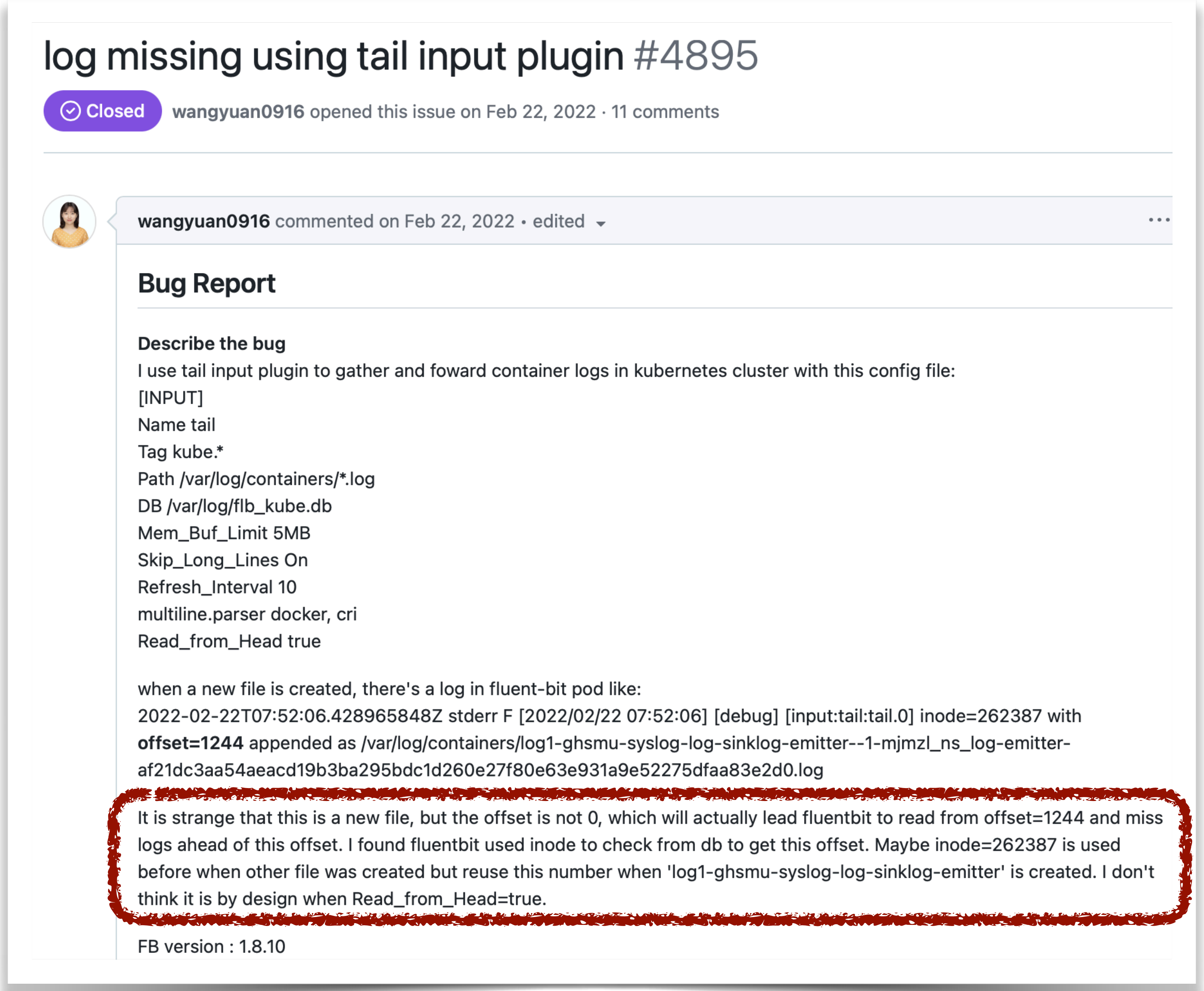
Projects

None yet

Milestone

No milestone

Development



log missing using tail input plugin #4895

Closed wangyuan0916 opened this issue on Feb 22, 2022 · 11 comments

wangyuan0916 commented on Feb 22, 2022 • edited

Bug Report

Describe the bug

I use tail input plugin to gather and foward container logs in kubernetes cluster with this config file:

```
[INPUT]
Name tail
Tag kube.*
Path /var/log/containers/*.log
DB /var/log/flb_kube.db
Mem_Buf_Limit 5MB
Skip_Long_Lines On
Refresh_Interval 10
multiline.parser docker, cri
Read_from_Head true
```

when a new file is created, there's a log in fluent-bit pod like:
2022-02-22T07:52:06.428965848Z stderr F [2022/02/22 07:52:06] [debug] [input:tail:tail.0] inode=262387 with **offset=1244** appended as /var/log/containers/log1-ghsmu-syslog-log-sinklog-emitter--1-mjmzl_ns_log-emitter-af21dc3aa54aeacd19b3ba295bdc1d260e27f80e63e931a9e52275dfaa83e2d0.log

It is strange that this is a new file, but the offset is not 0, which will actually lead fluentbit to read from offset=1244 and miss logs ahead of this offset. I found fluentbit used inode to check from db to get this offset. Maybe inode=262387 is used before when other file was created but reuse this number when 'log1-ghsmu-syslog-log-sinklog-emitter' is created. I don't think it is by design when Read_from_Head=true.

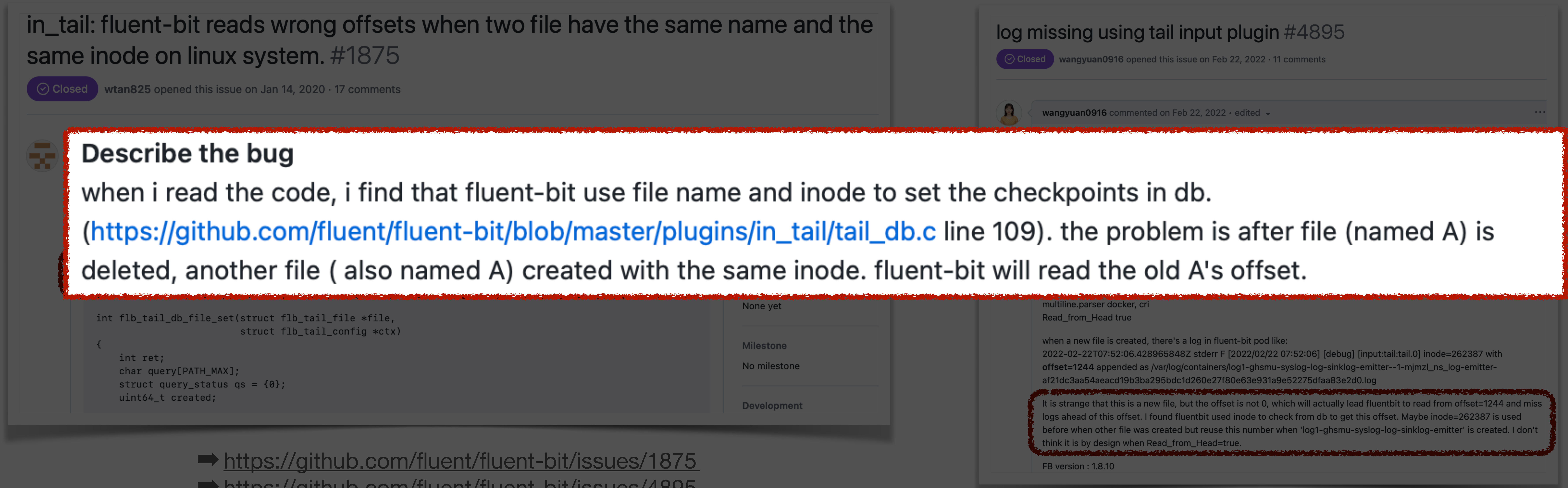
FB version : 1.8.10

➡ https://github.com/fluent/fluent-bit/issues/1875
➡ https://github.com/fluent/fluent-bit/issues/4895

Evaluation - Fluent Bit

Identifying erroneous actions that lead to data loss

- **Fluent Bit:** a high-performance logging and metrics processor and forwarder
- **Problem:** clients observe data loss when using Fluent Bit's tail input plugin (v1.4.0)



in_tail: fluent-bit reads wrong offsets when two file have the same name and the same inode on linux system. #1875

Closed wtan825 opened this issue on Jan 14, 2020 · 17 comments

Describe the bug

when i read the code, i find that fluent-bit use file name and inode to set the checkpoints in db. (https://github.com/fluent/fluent-bit/blob/master/plugins/in_tail/tail_db.c line 109). the problem is after file (named A) is deleted, another file (also named A) created with the same inode. fluent-bit will read the old A's offset.

```
int flb_tail_db_file_set(struct flb_tail_file *file,
                         struct flb_tail_config *ctx)
{
    int ret;
    char query[PATH_MAX];
    struct query_status qs = {0};
    uint64_t created;
```

log missing using tail input plugin #4895

Closed wangyuan0916 opened this issue on Feb 22, 2022 · 11 comments

wangyuan0916 commented on Feb 22, 2022 • edited

multiline.parser docker, cri
Read_from_Head true

when a new file is created, there's a log in fluent-bit pod like:

```
2022-02-22T07:52:06.428965848Z stderr F [2022/02/22 07:52:06] [debug] [input:tail:tail.0] inode=262387 with offset=1244 appended as /var/log/containers/log1-ghsmu-syslog-log-sinklog-emitter--1-mjmzl_ns_log-emitter-af21dc3aa54aeacd19b3ba295bdc1d260e27f80e63e931a9e52275dfaa83e2d0.log
```

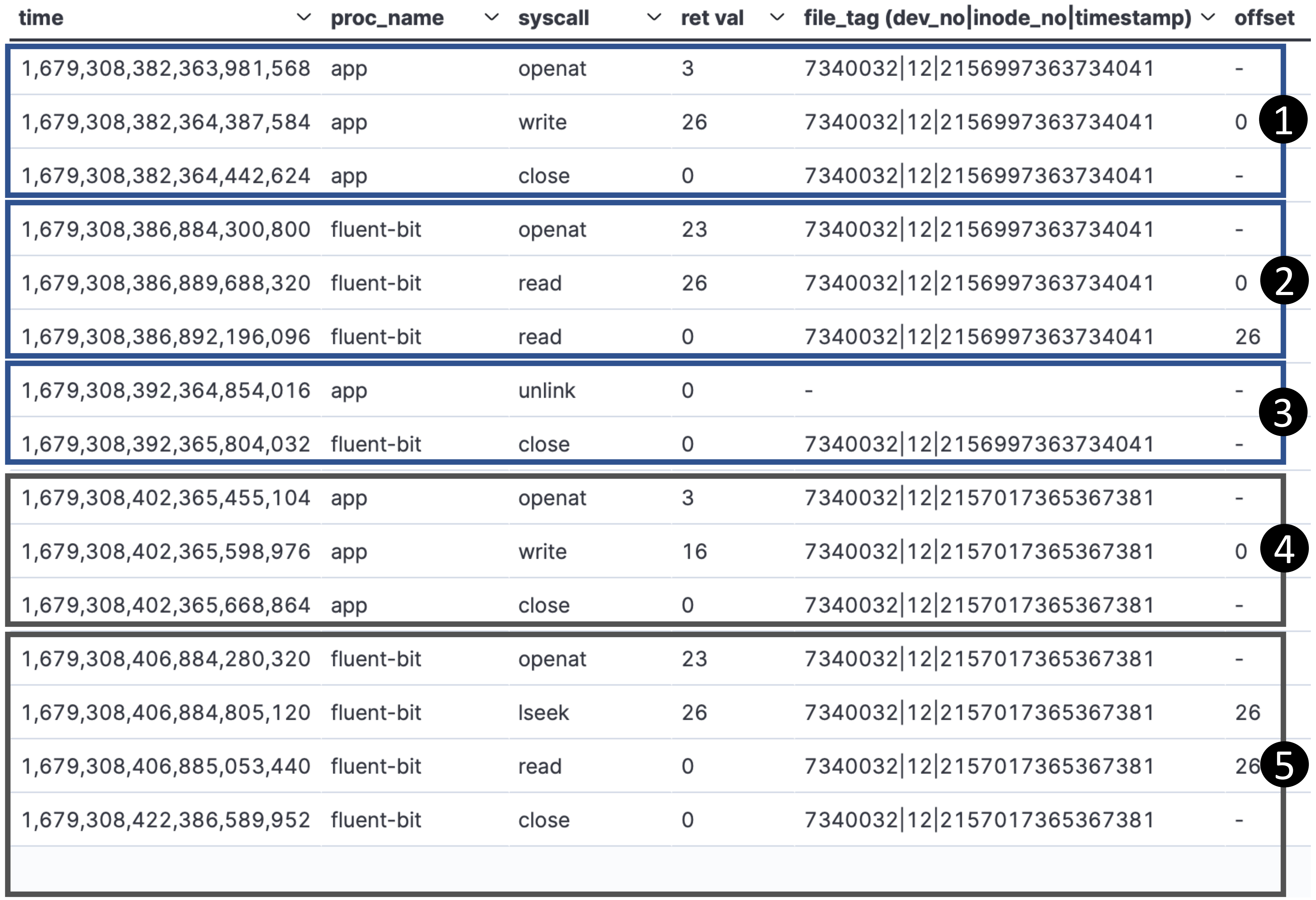
It is strange that this is a new file, but the offset is not 0, which will actually lead fluentbit to read from offset=1244 and miss logs ahead of this offset. I found fluentbit used inode to check from db to get this offset. Maybe inode=262387 is used before when other file was created but reuse this number when 'log1-ghsmu-syslog-log-sinklog-emitter' is created. I don't think it is by design when Read_from_Head=true.

FB version : 1.8.10

➡ https://github.com/fluent/fluent-bit/issues/1875
➡ https://github.com/fluent/fluent-bit/issues/4895

Evaluation - Fluent Bit (v1.4.0)

Identifying erroneous actions that lead to data loss



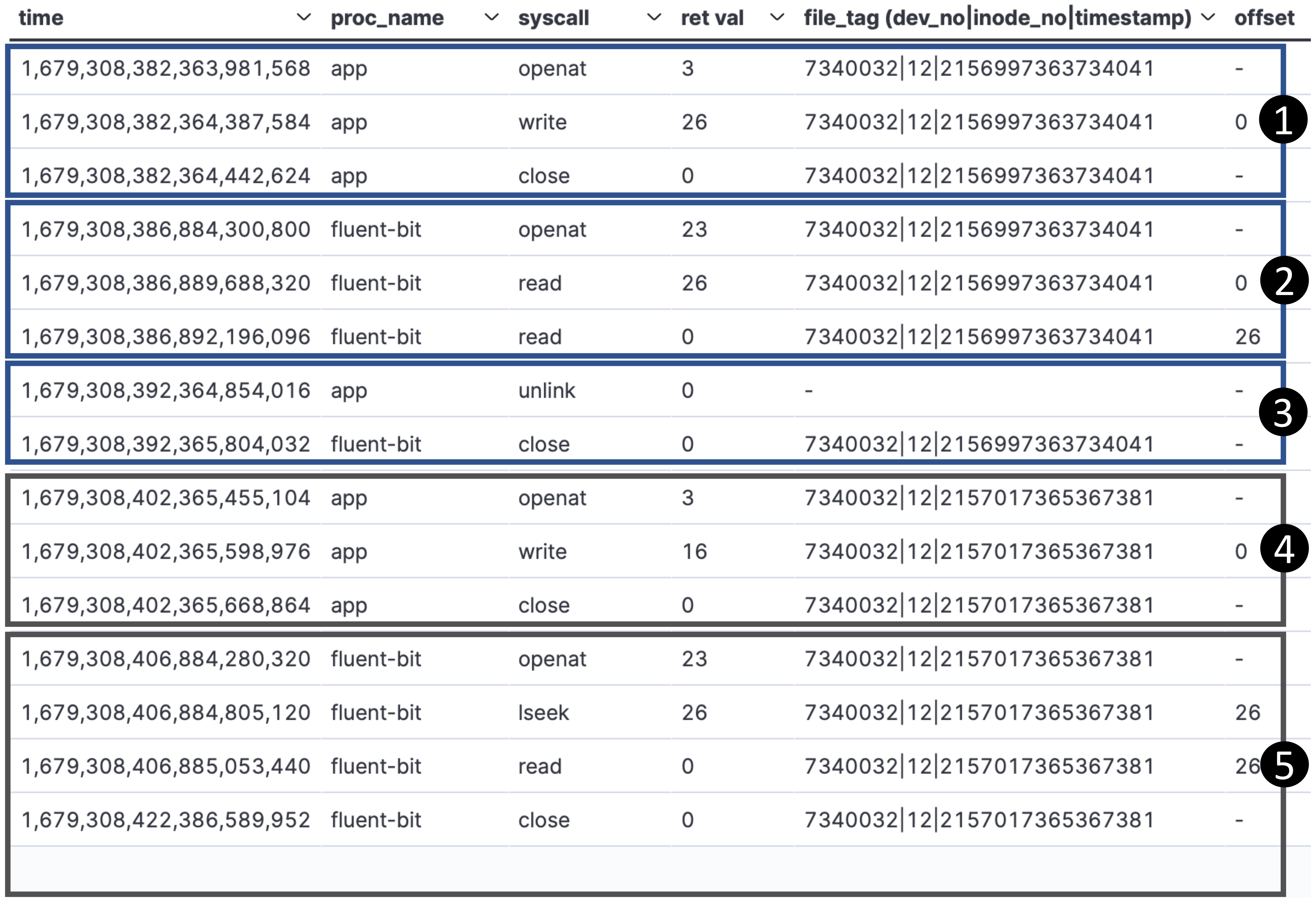
time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,308,382,363,981,568	app	openat	3	7340032|12|2156997363734041	-
1,679,308,382,364,387,584	app	write	26	7340032|12|2156997363734041	0
1,679,308,382,364,442,624	app	close	0	7340032|12|2156997363734041	-
1,679,308,386,884,300,800	fluent-bit	openat	23	7340032|12|2156997363734041	-
1,679,308,386,889,688,320	fluent-bit	read	26	7340032|12|2156997363734041	0
1,679,308,386,892,196,096	fluent-bit	read	0	7340032|12|2156997363734041	26
1,679,308,392,364,854,016	app	unlink	0	-	-
1,679,308,392,365,804,032	fluent-bit	close	0	7340032|12|2156997363734041	-
1,679,308,402,365,455,104	app	openat	3	7340032|12|2157017365367381	-
1,679,308,402,365,598,976	app	write	16	7340032|12|2157017365367381	0
1,679,308,402,365,668,864	app	close	0	7340032|12|2157017365367381	-
1,679,308,406,884,280,320	fluent-bit	openat	23	7340032|12|2157017365367381	-
1,679,308,406,884,805,120	fluent-bit	lseek	26	7340032|12|2157017365367381	26
1,679,308,406,885,053,440	fluent-bit	read	0	7340032|12|2157017365367381	26
1,679,308,422,386,589,952	fluent-bit	close	0	7340032|12|2157017365367381	-

Evaluation - Fluent Bit (v1.4.0)

Identifying erroneous actions that lead to data loss

This is the first log line

app.log

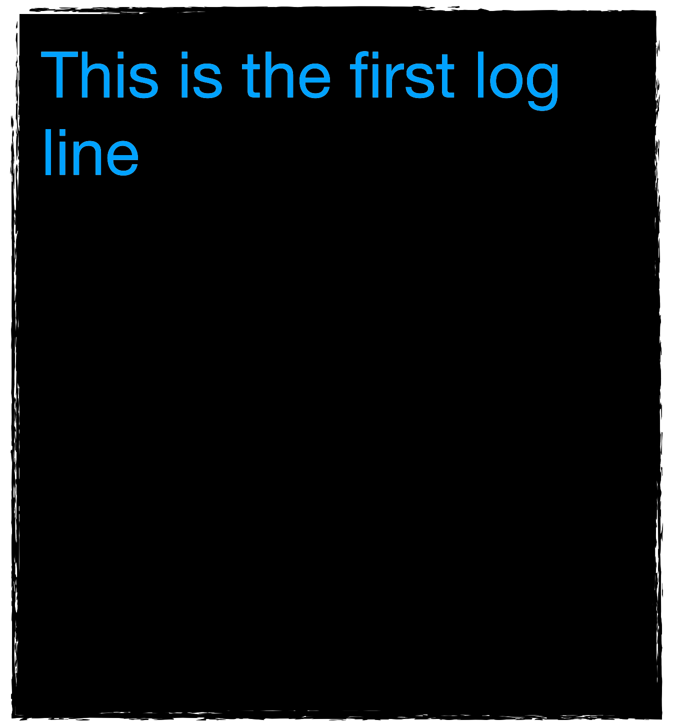


time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset	
1,679,308,382,363,981,568	app	openat	3	7340032|12|2156997363734041	-	
1,679,308,382,364,387,584	app	write	26	7340032|12|2156997363734041	0	1
1,679,308,382,364,442,624	app	close	0	7340032|12|2156997363734041	-	
1,679,308,386,884,300,800	fluent-bit	openat	23	7340032|12|2156997363734041	-	
1,679,308,386,889,688,320	fluent-bit	read	26	7340032|12|2156997363734041	0	2
1,679,308,386,892,196,096	fluent-bit	read	0	7340032|12|2156997363734041	26	
1,679,308,392,364,854,016	app	unlink	0	-	-	3
1,679,308,392,365,804,032	fluent-bit	close	0	7340032|12|2156997363734041	-	
1,679,308,402,365,455,104	app	openat	3	7340032|12|2157017365367381	-	
1,679,308,402,365,598,976	app	write	16	7340032|12|2157017365367381	0	4
1,679,308,402,365,668,864	app	close	0	7340032|12|2157017365367381	-	
1,679,308,406,884,280,320	fluent-bit	openat	23	7340032|12|2157017365367381	-	
1,679,308,406,884,805,120	fluent-bit	lseek	26	7340032|12|2157017365367381	26	
1,679,308,406,885,053,440	fluent-bit	read	0	7340032|12|2157017365367381	26	5
1,679,308,422,386,589,952	fluent-bit	close	0	7340032|12|2157017365367381	-	

app opens a file, **writes 26 bytes** from **offset 0** and closes it

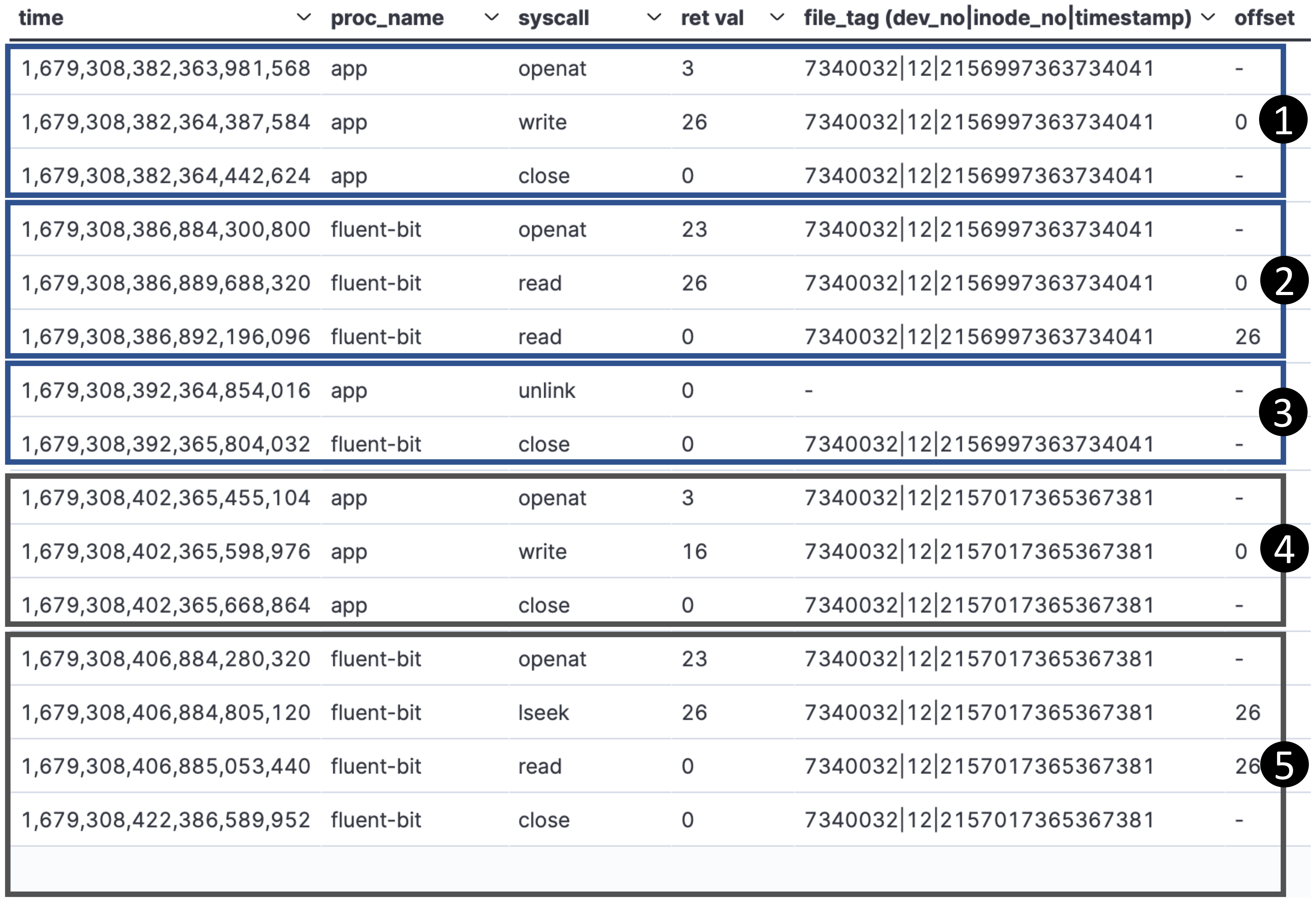
Evaluation - Fluent Bit (v1.4.0)

Identifying erroneous actions that lead to data loss



This is the first log line

app.log



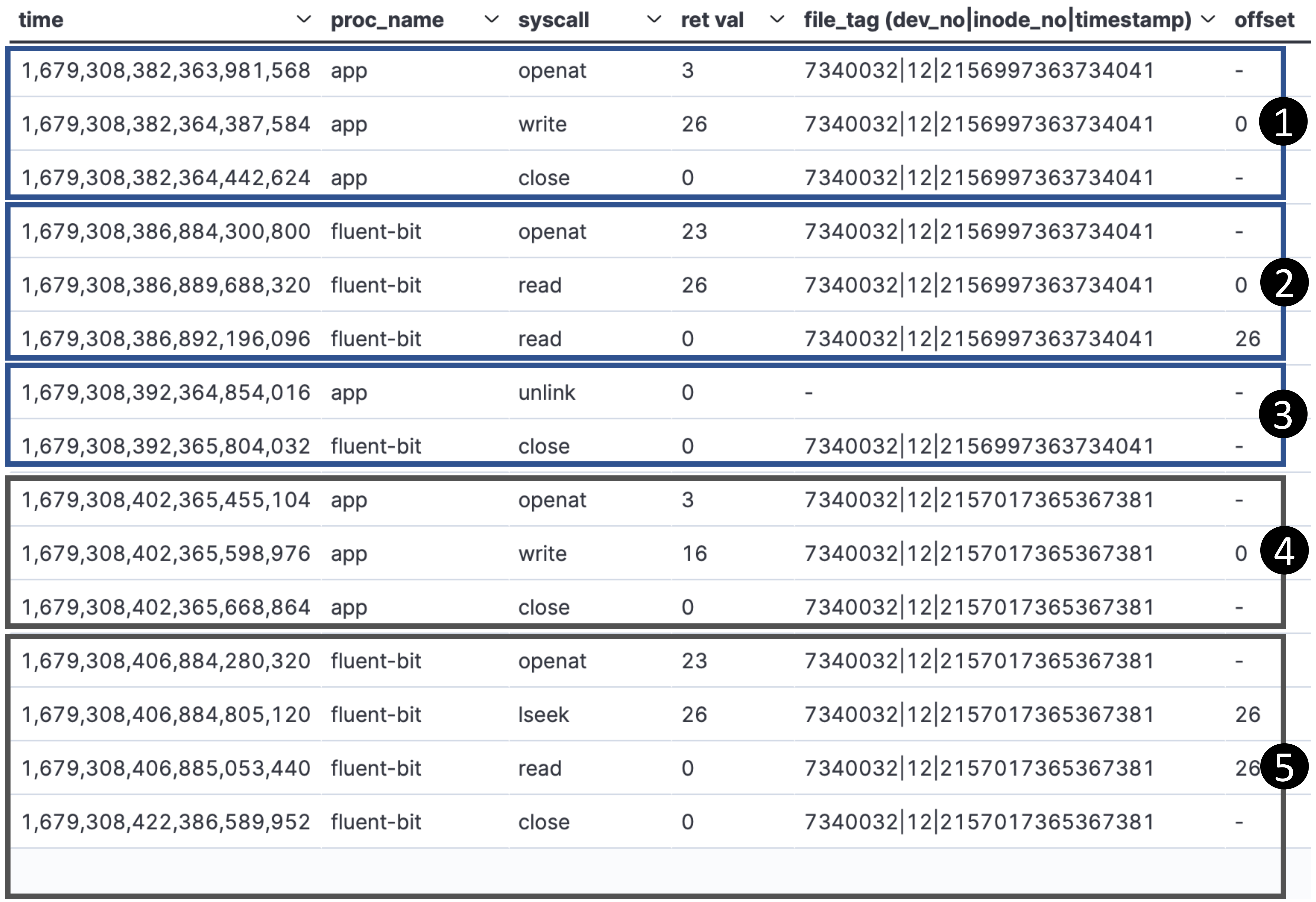
time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,308,382,363,981,568	app	openat	3	7340032|12|2156997363734041	-
1,679,308,382,364,387,584	app	write	26	7340032|12|2156997363734041	0 ❶
1,679,308,382,364,442,624	app	close	0	7340032|12|2156997363734041	-
1,679,308,386,884,300,800	fluent-bit	openat	23	7340032|12|2156997363734041	-
1,679,308,386,889,688,320	fluent-bit	read	26	7340032|12|2156997363734041	0 ❷
1,679,308,386,892,196,096	fluent-bit	read	0	7340032|12|2156997363734041	26
1,679,308,392,364,854,016	app	unlink	0	-	- ❸
1,679,308,392,365,804,032	fluent-bit	close	0	7340032|12|2156997363734041	-
1,679,308,402,365,455,104	app	openat	3	7340032|12|2157017365367381	-
1,679,308,402,365,598,976	app	write	16	7340032|12|2157017365367381	0 ❹
1,679,308,402,365,668,864	app	close	0	7340032|12|2157017365367381	-
1,679,308,406,884,280,320	fluent-bit	openat	23	7340032|12|2157017365367381	-
1,679,308,406,884,805,120	fluent-bit	lseek	26	7340032|12|2157017365367381	26
1,679,308,406,885,053,440	fluent-bit	read	0	7340032|12|2157017365367381	26 ❺
1,679,308,422,386,589,952	fluent-bit	close	0	7340032|12|2157017365367381	-

app opens a file, **writes 26 bytes** from **offset 0** and closes it

fluent-bit opens the file and **reads 26 bytes** from **offset 0**

Evaluation - Fluent Bit (v1.4.0)

Identifying erroneous actions that lead to data loss



time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,308,382,363,981,568	app	openat	3	7340032|12|2156997363734041	-
1,679,308,382,364,387,584	app	write	26	7340032|12|2156997363734041	0
1,679,308,382,364,442,624	app	close	0	7340032|12|2156997363734041	-
1,679,308,386,884,300,800	fluent-bit	openat	23	7340032|12|2156997363734041	-
1,679,308,386,889,688,320	fluent-bit	read	26	7340032|12|2156997363734041	0
1,679,308,386,892,196,096	fluent-bit	read	0	7340032|12|2156997363734041	26
1,679,308,392,364,854,016	app	unlink	0	-	-
1,679,308,392,365,804,032	fluent-bit	close	0	7340032|12|2156997363734041	-
1,679,308,402,365,455,104	app	openat	3	7340032|12|2157017365367381	-
1,679,308,402,365,598,976	app	write	16	7340032|12|2157017365367381	0
1,679,308,402,365,668,864	app	close	0	7340032|12|2157017365367381	-
1,679,308,406,884,280,320	fluent-bit	openat	23	7340032|12|2157017365367381	-
1,679,308,406,884,805,120	fluent-bit	lseek	26	7340032|12|2157017365367381	26
1,679,308,406,885,053,440	fluent-bit	read	0	7340032|12|2157017365367381	26
1,679,308,422,386,589,952	fluent-bit	close	0	7340032|12|2157017365367381	-

1 *app* opens a file, **writes 26 bytes** from **offset 0** and closes it

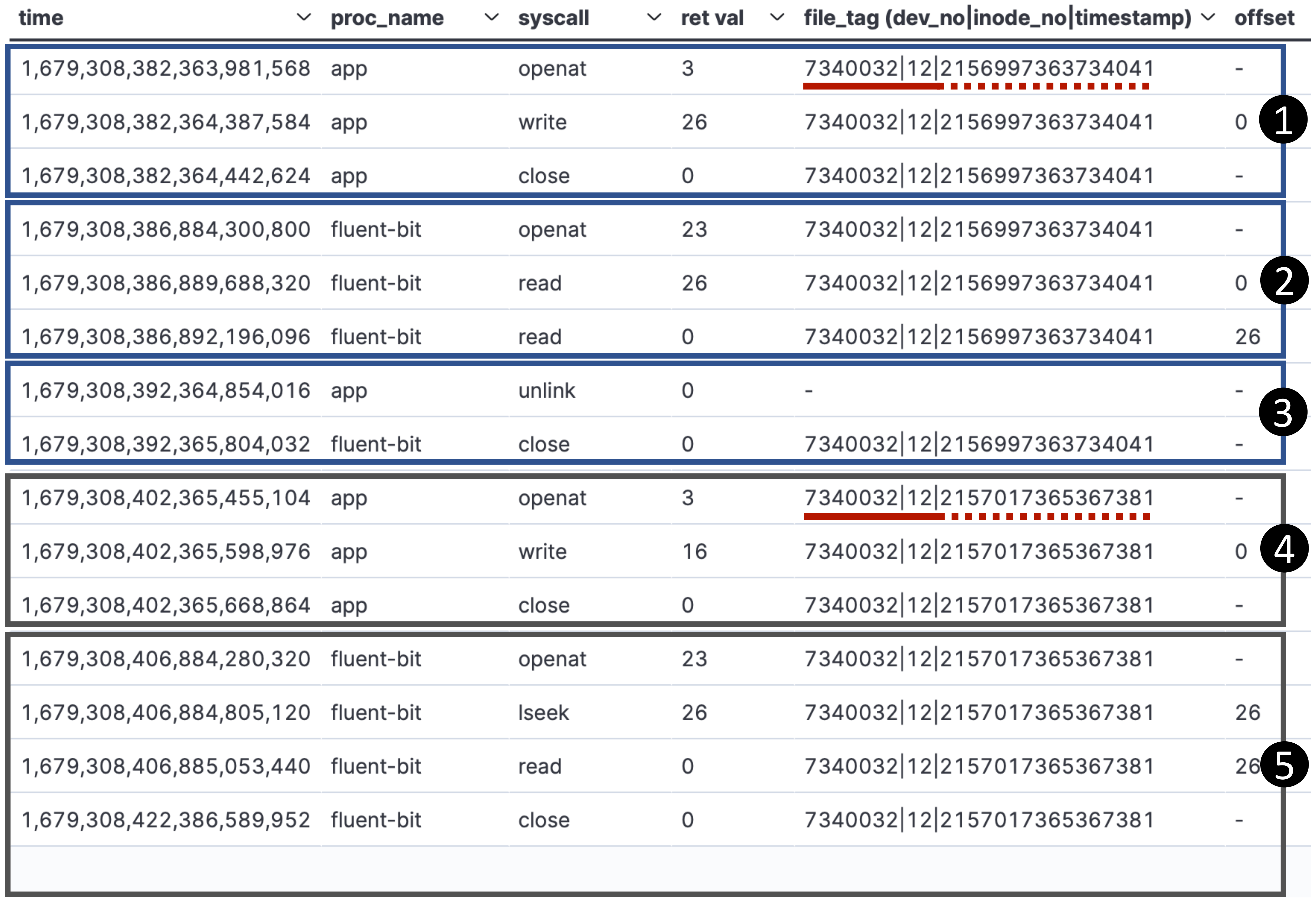
2 *fluent-bit* opens the file and **reads 26 bytes** from **offset 0**

3 *app* **removes** the file and *fluent-bit* closes its file descriptor

Evaluation - Fluent Bit (v1.4.0)

Identifying erroneous actions that lead to data loss

app.log

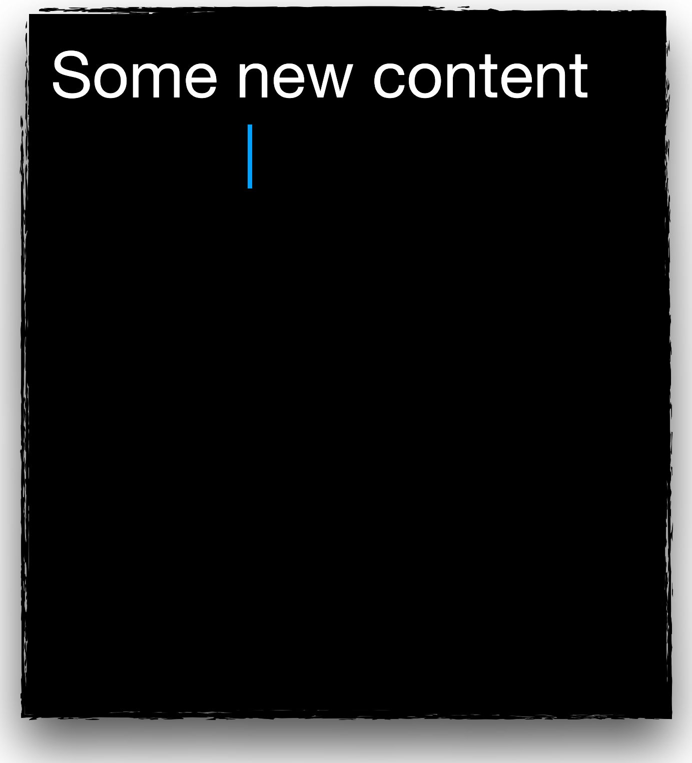


time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,308,382,363,981,568	app	openat	3	7340032|12|2156997363734041	-
1,679,308,382,364,387,584	app	write	26	7340032|12|2156997363734041	0
1,679,308,382,364,442,624	app	close	0	7340032|12|2156997363734041	-
1,679,308,386,884,300,800	fluent-bit	openat	23	7340032|12|2156997363734041	-
1,679,308,386,889,688,320	fluent-bit	read	26	7340032|12|2156997363734041	0
1,679,308,386,892,196,096	fluent-bit	read	0	7340032|12|2156997363734041	26
1,679,308,392,364,854,016	app	unlink	0	-	-
1,679,308,392,365,804,032	fluent-bit	close	0	7340032|12|2156997363734041	-
1,679,308,402,365,455,104	app	openat	3	7340032|12|2157017365367381	-
1,679,308,402,365,598,976	app	write	16	7340032|12|2157017365367381	0
1,679,308,402,365,668,864	app	close	0	7340032|12|2157017365367381	-
1,679,308,406,884,280,320	fluent-bit	openat	23	7340032|12|2157017365367381	-
1,679,308,406,884,805,120	fluent-bit	lseek	26	7340032|12|2157017365367381	26
1,679,308,406,885,053,440	fluent-bit	read	0	7340032|12|2157017365367381	26
1,679,308,422,386,589,952	fluent-bit	close	0	7340032|12|2157017365367381	-

1. *app* opens a file, **writes 26 bytes** from **offset 0** and closes it
2. *fluent-bit* opens the file and **reads 26 bytes** from **offset 0**
3. *app* **removes** the file and *fluent-bit* closes its file descriptor
4. *app* opens a **new file** with **same name** and **inode** number (12), **writes 16 bytes** from **offset 0** and closes the file
5.

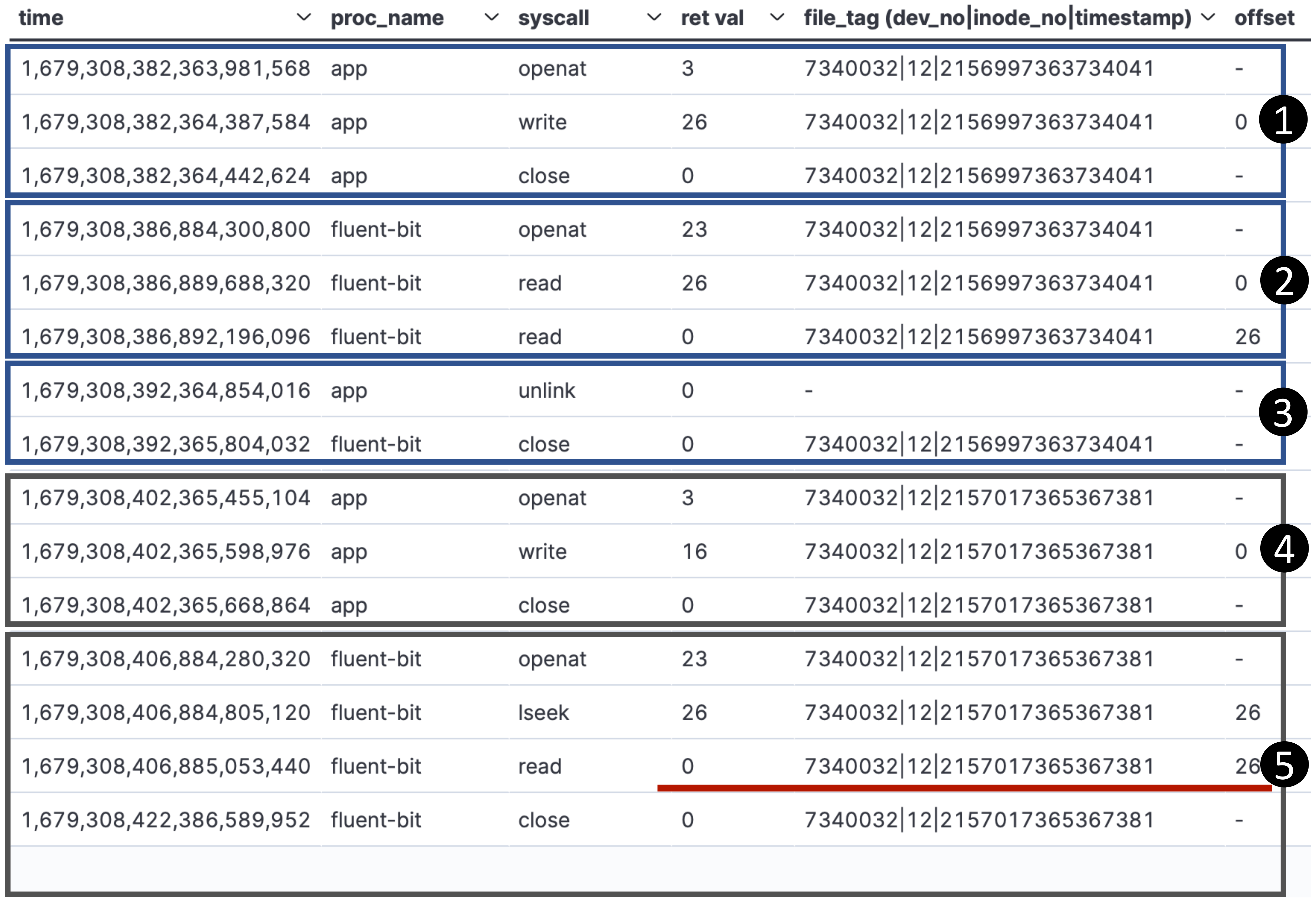
Evaluation - Fluent Bit (v1.4.0)

Identifying erroneous actions that lead to data loss



Some new content

app.log

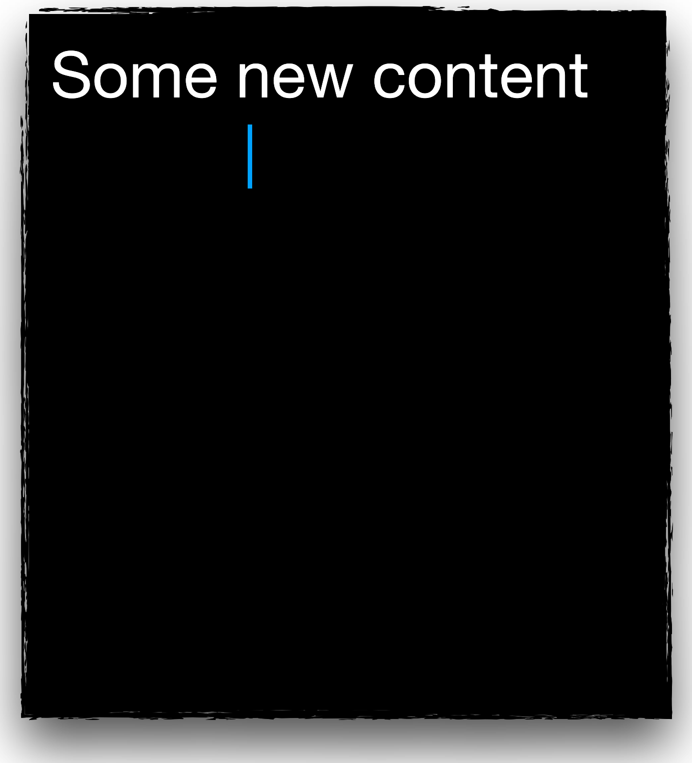


time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,308,382,363,981,568	app	openat	3	7340032|12|2156997363734041	-
1,679,308,382,364,387,584	app	write	26	7340032|12|2156997363734041	0
1,679,308,382,364,442,624	app	close	0	7340032|12|2156997363734041	-
1,679,308,386,884,300,800	fluent-bit	openat	23	7340032|12|2156997363734041	-
1,679,308,386,889,688,320	fluent-bit	read	26	7340032|12|2156997363734041	0
1,679,308,386,892,196,096	fluent-bit	read	0	7340032|12|2156997363734041	26
1,679,308,392,364,854,016	app	unlink	0	-	-
1,679,308,392,365,804,032	fluent-bit	close	0	7340032|12|2156997363734041	-
1,679,308,402,365,455,104	app	openat	3	7340032|12|2157017365367381	-
1,679,308,402,365,598,976	app	write	16	7340032|12|2157017365367381	0
1,679,308,402,365,668,864	app	close	0	7340032|12|2157017365367381	-
1,679,308,406,884,280,320	fluent-bit	openat	23	7340032|12|2157017365367381	-
1,679,308,406,884,805,120	fluent-bit	lseek	26	7340032|12|2157017365367381	26
1,679,308,406,885,053,440	fluent-bit	read	0	7340032|12|2157017365367381	26
1,679,308,422,386,589,952	fluent-bit	close	0	7340032|12|2157017365367381	-

1. *app* opens a file, **writes 26 bytes** from **offset 0** and closes it
2. *fluent-bit* opens the file and **reads 26 bytes** from **offset 0**
3. *app* **removes** the file and *fluent-bit* closes its file descriptor
4. *app* opens a **new file** with **same name** and **inode** number (12), **writes 16 bytes** from **offset 0** and closes the file
5. *fluent-bit* opens new file, **jumps to offset 26** and tries to read from there, which **results in 0 bytes** (EOF)

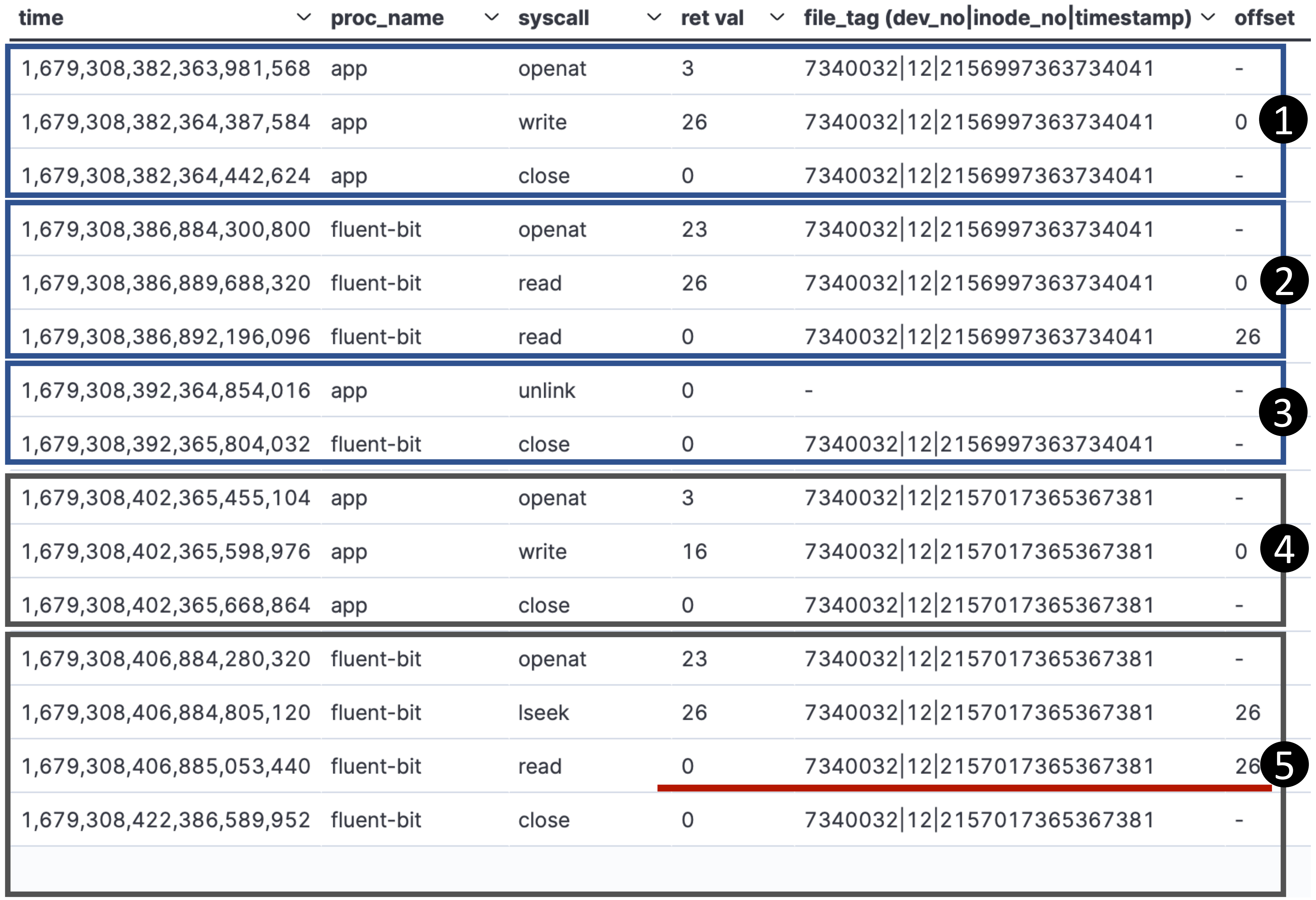
Evaluation - Fluent Bit (v1.4.0)

Identifying erroneous actions that lead to data loss



Some new content

app.log



time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,308,382,363,981,568	app	openat	3	7340032|12|2156997363734041	-
1,679,308,382,364,387,584	app	write	26	7340032|12|2156997363734041	0
1,679,308,382,364,442,624	app	close	0	7340032|12|2156997363734041	-
1,679,308,386,884,300,800	fluent-bit	openat	23	7340032|12|2156997363734041	-
1,679,308,386,889,688,320	fluent-bit	read	26	7340032|12|2156997363734041	0
1,679,308,386,892,196,096	fluent-bit	read	0	7340032|12|2156997363734041	26
1,679,308,392,364,854,016	app	unlink	0	-	-
1,679,308,392,365,804,032	fluent-bit	close	0	7340032|12|2156997363734041	-
1,679,308,402,365,455,104	app	openat	3	7340032|12|2157017365367381	-
1,679,308,402,365,598,976	app	write	16	7340032|12|2157017365367381	0
1,679,308,402,365,668,864	app	close	0	7340032|12|2157017365367381	-
1,679,308,406,884,280,320	fluent-bit	openat	23	7340032|12|2157017365367381	-
1,679,308,406,884,805,120	fluent-bit	lseek	26	7340032|12|2157017365367381	26
1,679,308,406,885,053,440	fluent-bit	read	0	7340032|12|2157017365367381	26
1,679,308,422,386,589,952	fluent-bit	close	0	7340032|12|2157017365367381	-

1 *app* opens a file, **writes 26 bytes** from **offset 0** and closes it

2 *fluent-bit* opens the file and **reads 26 bytes** from **offset 0**

3 *app* **removes** the file and *fluent-bit* closes its file descriptor

4 *app* opens a **new file** with **same name** and **inode** number (12), **writes 16 bytes** from **offset 0** and closes the file

5 *fluent-bit* opens new file, **jumps to offset 26** and tries to read from there, which **results in 0 bytes** (EOF)

Erroneous access pattern!

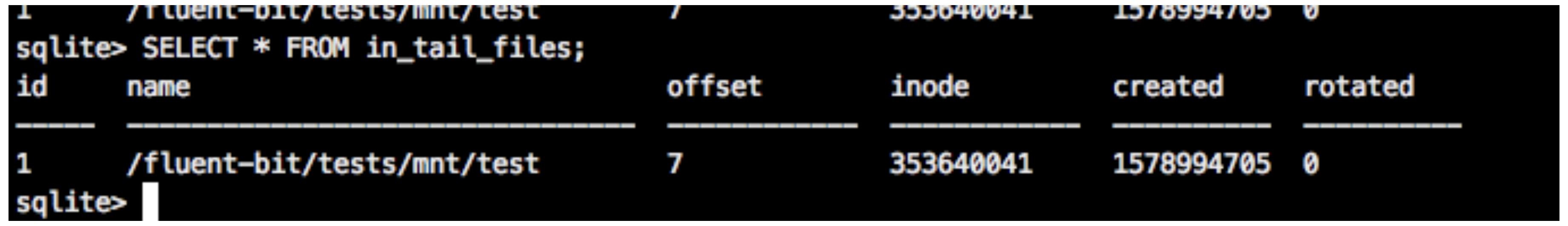
Evaluation - Fluent Bit

Identifying erroneous actions that lead to data loss

- **Root cause:** Fluent Bit tracks the last processed offset for each file, which is not reset when the file is removed

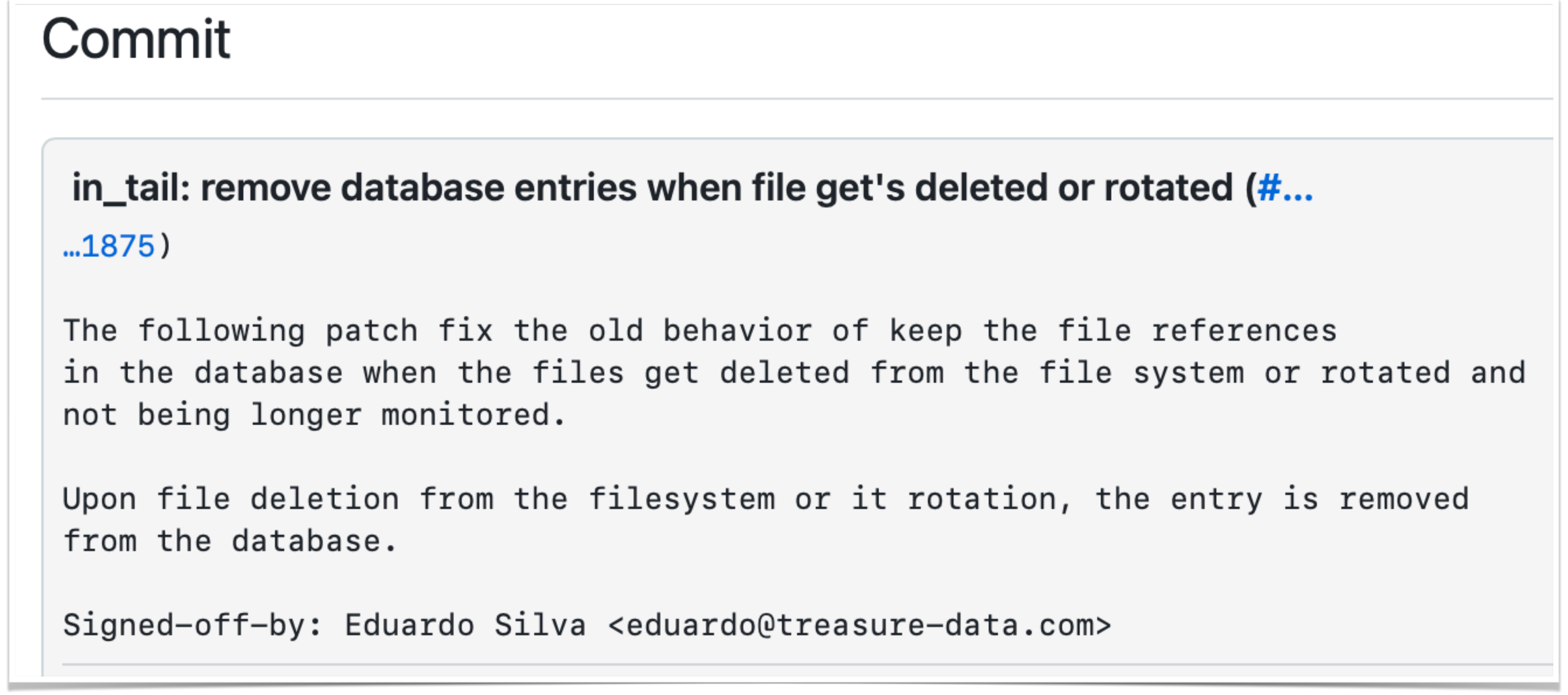
- **Solution:** Upon file deletion or rotation, remove the entry from the database

- **Validation:** Use DIO to validate the correction of this erroneous pattern in a recent version



```
1      /fluent-bit/tests/mnt/test       7           353640041    1578994705  0
sqlite> SELECT * FROM in_tail_files;
id     name                             offset      inode        created     rotated
-----  -------------------------------  ----------  -----------  ----------  ---------
1      /fluent-bit/tests/mnt/test       7           353640041    1578994705  0
sqlite>
```

Database



Commit

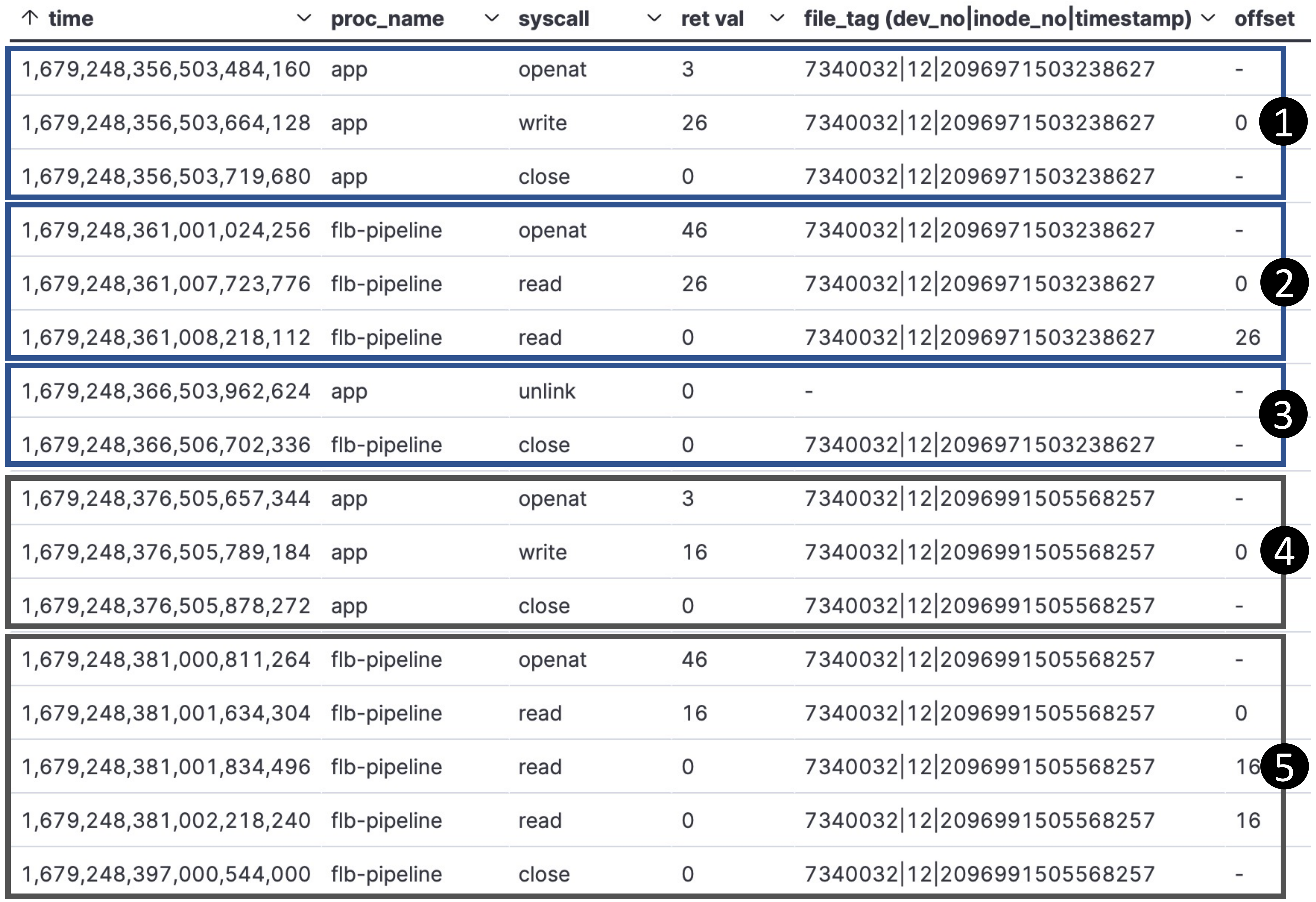
in_tail: remove database entries when file get's deleted or rotated (#… …1875)

```
The following patch fix the old behavior of keep the file references
in the database when the files get deleted from the file system or rotated and
not being longer monitored.

Upon file deletion from the filesystem or it rotation, the entry is removed
from the database.

Signed-off-by: Eduardo Silva <eduardo@treasure-data.com>
```

Fix

Evaluation - Fluent Bit (v2.0.5)

Identifying erroneous actions that lead to data loss



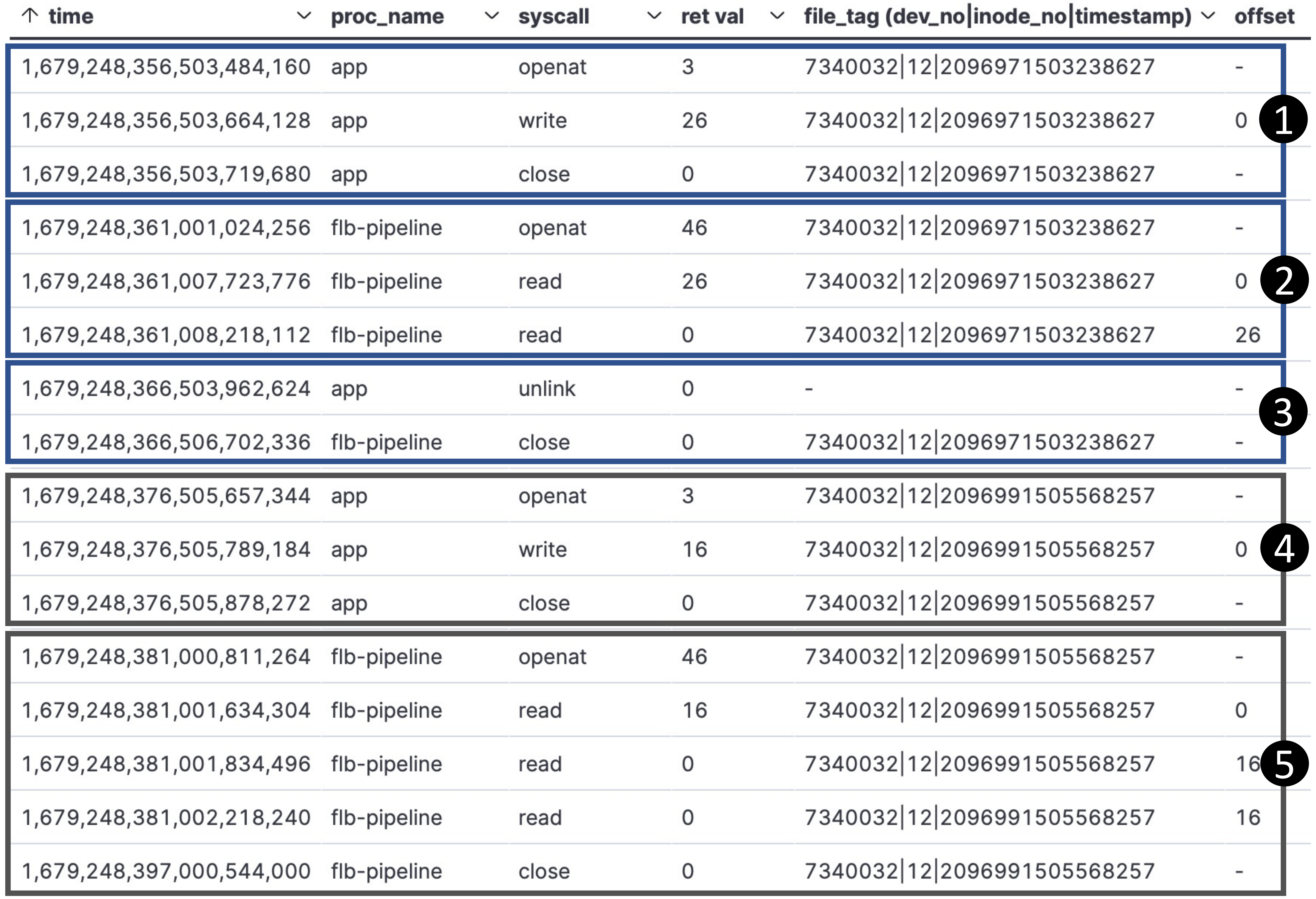
↑ time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,248,356,503,484,160	app	openat	3	7340032|12|2096971503238627	-
1,679,248,356,503,664,128	app	write	26	7340032|12|2096971503238627	0
1,679,248,356,503,719,680	app	close	0	7340032|12|2096971503238627	-
1,679,248,361,001,024,256	flb-pipeline	openat	46	7340032|12|2096971503238627	-
1,679,248,361,007,723,776	flb-pipeline	read	26	7340032|12|2096971503238627	0
1,679,248,361,008,218,112	flb-pipeline	read	0	7340032|12|2096971503238627	26
1,679,248,366,503,962,624	app	unlink	0	-	-
1,679,248,366,506,702,336	flb-pipeline	close	0	7340032|12|2096971503238627	-
1,679,248,376,505,657,344	app	openat	3	7340032|12|2096991505568257	-
1,679,248,376,505,789,184	app	write	16	7340032|12|2096991505568257	0
1,679,248,376,505,878,272	app	close	0	7340032|12|2096991505568257	-
1,679,248,381,000,811,264	flb-pipeline	openat	46	7340032|12|2096991505568257	-
1,679,248,381,001,634,304	flb-pipeline	read	16	7340032|12|2096991505568257	0
1,679,248,381,001,834,496	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,381,002,218,240	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,397,000,544,000	flb-pipeline	close	0	7340032|12|2096991505568257	-

Evaluation - Fluent Bit (v2.0.5)

Identifying erroneous actions that lead to data loss

This is the first log line

app.log

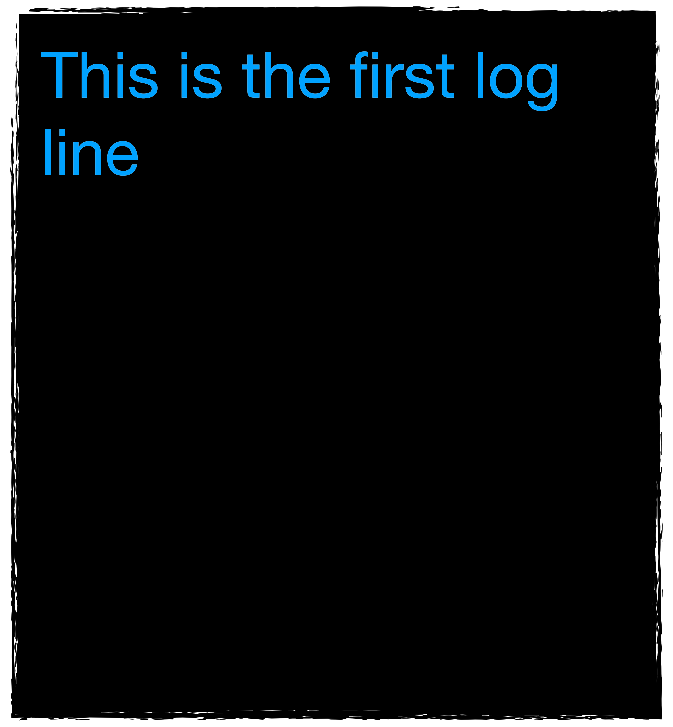


↑ time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset	
1,679,248,356,503,484,160	app	openat	3	7340032|12|2096971503238627	-	
1,679,248,356,503,664,128	app	write	26	7340032|12|2096971503238627	0	1
1,679,248,356,503,719,680	app	close	0	7340032|12|2096971503238627	-	
1,679,248,361,001,024,256	flb-pipeline	openat	46	7340032|12|2096971503238627	-	
1,679,248,361,007,723,776	flb-pipeline	read	26	7340032|12|2096971503238627	0	2
1,679,248,361,008,218,112	flb-pipeline	read	0	7340032|12|2096971503238627	26	
1,679,248,366,503,962,624	app	unlink	0	-	-	3
1,679,248,366,506,702,336	flb-pipeline	close	0	7340032|12|2096971503238627	-	
1,679,248,376,505,657,344	app	openat	3	7340032|12|2096991505568257	-	
1,679,248,376,505,789,184	app	write	16	7340032|12|2096991505568257	0	4
1,679,248,376,505,878,272	app	close	0	7340032|12|2096991505568257	-	
1,679,248,381,000,811,264	flb-pipeline	openat	46	7340032|12|2096991505568257	-	
1,679,248,381,001,634,304	flb-pipeline	read	16	7340032|12|2096991505568257	0	
1,679,248,381,001,834,496	flb-pipeline	read	0	7340032|12|2096991505568257	16	5
1,679,248,381,002,218,240	flb-pipeline	read	0	7340032|12|2096991505568257	16	
1,679,248,397,000,544,000	flb-pipeline	close	0	7340032|12|2096991505568257	-	

app opens a file, **writes 26 bytes** from **offset 0** and closes it

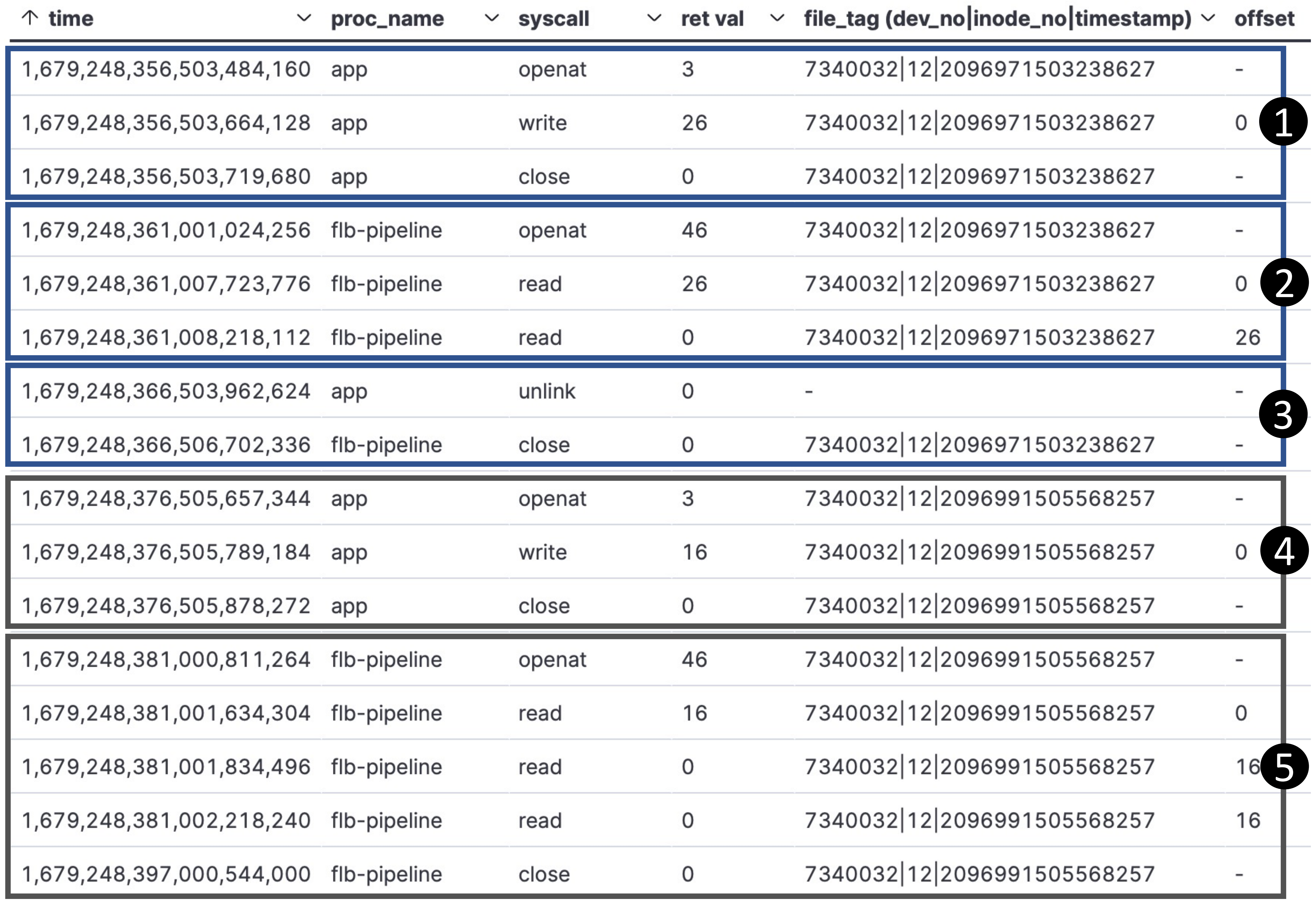
Evaluation - Fluent Bit (v2.0.5)

Identifying erroneous actions that lead to data loss



This is the first log line

app.log



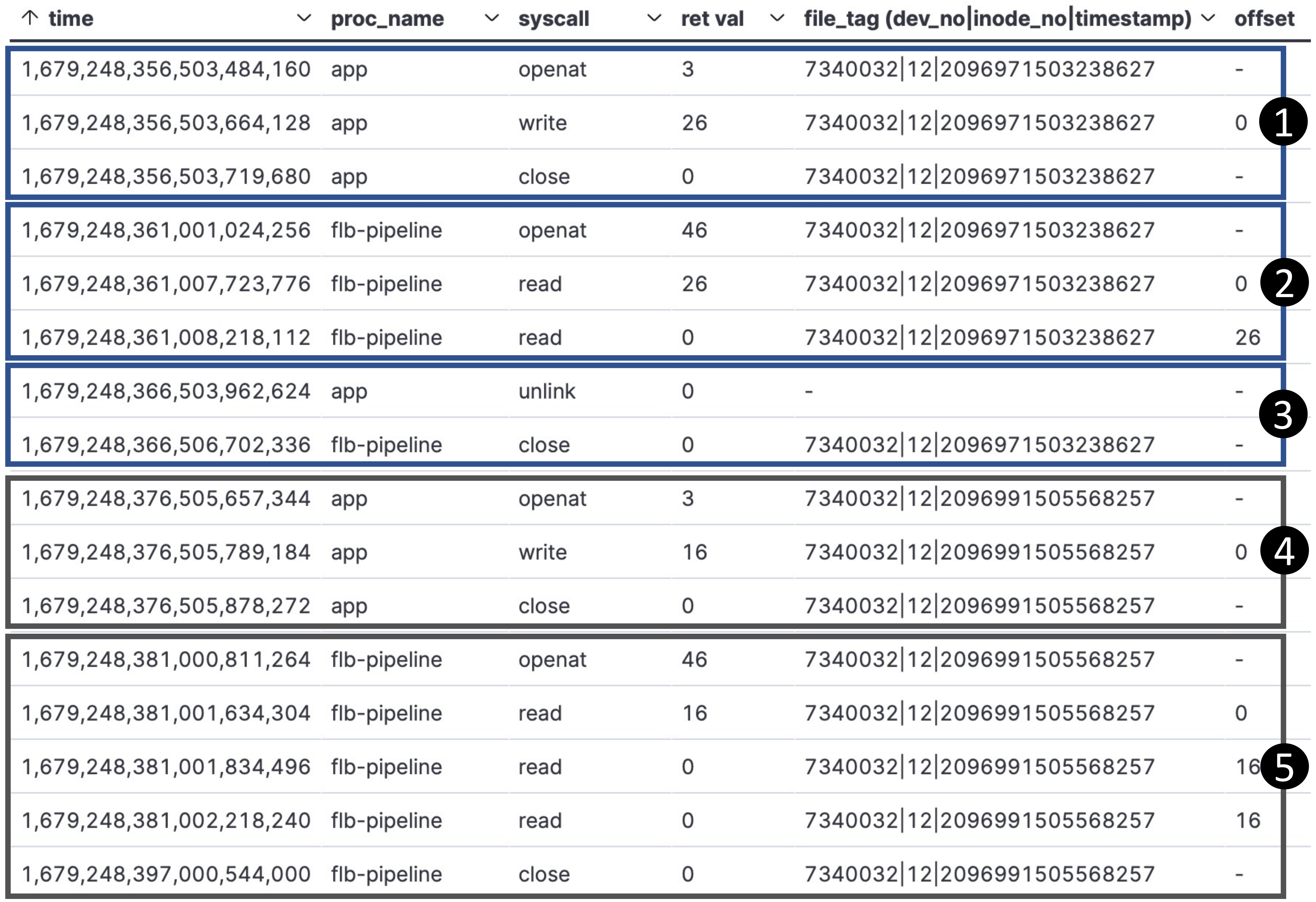
↑ time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset	
1,679,248,356,503,484,160	app	openat	3	7340032|12|2096971503238627	-	
1,679,248,356,503,664,128	app	write	26	7340032|12|2096971503238627	0	❶
1,679,248,356,503,719,680	app	close	0	7340032|12|2096971503238627	-	
1,679,248,361,001,024,256	flb-pipeline	openat	46	7340032|12|2096971503238627	-	
1,679,248,361,007,723,776	flb-pipeline	read	26	7340032|12|2096971503238627	0	❷
1,679,248,361,008,218,112	flb-pipeline	read	0	7340032|12|2096971503238627	26	
1,679,248,366,503,962,624	app	unlink	0	-	-	❸
1,679,248,366,506,702,336	flb-pipeline	close	0	7340032|12|2096971503238627	-	
1,679,248,376,505,657,344	app	openat	3	7340032|12|2096991505568257	-	
1,679,248,376,505,789,184	app	write	16	7340032|12|2096991505568257	0	❹
1,679,248,376,505,878,272	app	close	0	7340032|12|2096991505568257	-	
1,679,248,381,000,811,264	flb-pipeline	openat	46	7340032|12|2096991505568257	-	
1,679,248,381,001,634,304	flb-pipeline	read	16	7340032|12|2096991505568257	0	
1,679,248,381,001,834,496	flb-pipeline	read	0	7340032|12|2096991505568257	16	❺
1,679,248,381,002,218,240	flb-pipeline	read	0	7340032|12|2096991505568257	16	
1,679,248,397,000,544,000	flb-pipeline	close	0	7340032|12|2096991505568257	-	

app opens a file, **writes 26 bytes** from **offset 0** and closes it

fluent-bit opens the file, **reads 26 bytes** from **offset 0**

Evaluation - Fluent Bit (v2.0.5)

Identifying erroneous actions that lead to data loss



↑ time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,248,356,503,484,160	app	openat	3	7340032|12|2096971503238627	-
1,679,248,356,503,664,128	app	write	26	7340032|12|2096971503238627	0
1,679,248,356,503,719,680	app	close	0	7340032|12|2096971503238627	-
1,679,248,361,001,024,256	flb-pipeline	openat	46	7340032|12|2096971503238627	-
1,679,248,361,007,723,776	flb-pipeline	read	26	7340032|12|2096971503238627	0
1,679,248,361,008,218,112	flb-pipeline	read	0	7340032|12|2096971503238627	26
1,679,248,366,503,962,624	app	unlink	0	-	-
1,679,248,366,506,702,336	flb-pipeline	close	0	7340032|12|2096971503238627	-
1,679,248,376,505,657,344	app	openat	3	7340032|12|2096991505568257	-
1,679,248,376,505,789,184	app	write	16	7340032|12|2096991505568257	0
1,679,248,376,505,878,272	app	close	0	7340032|12|2096991505568257	-
1,679,248,381,000,811,264	flb-pipeline	openat	46	7340032|12|2096991505568257	-
1,679,248,381,001,634,304	flb-pipeline	read	16	7340032|12|2096991505568257	0
1,679,248,381,001,834,496	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,381,002,218,240	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,397,000,544,000	flb-pipeline	close	0	7340032|12|2096991505568257	-

app opens a file, **writes 26 bytes** from **offset 0** and closes it

fluent-bit opens the file, **reads 26 bytes** from **offset 0**

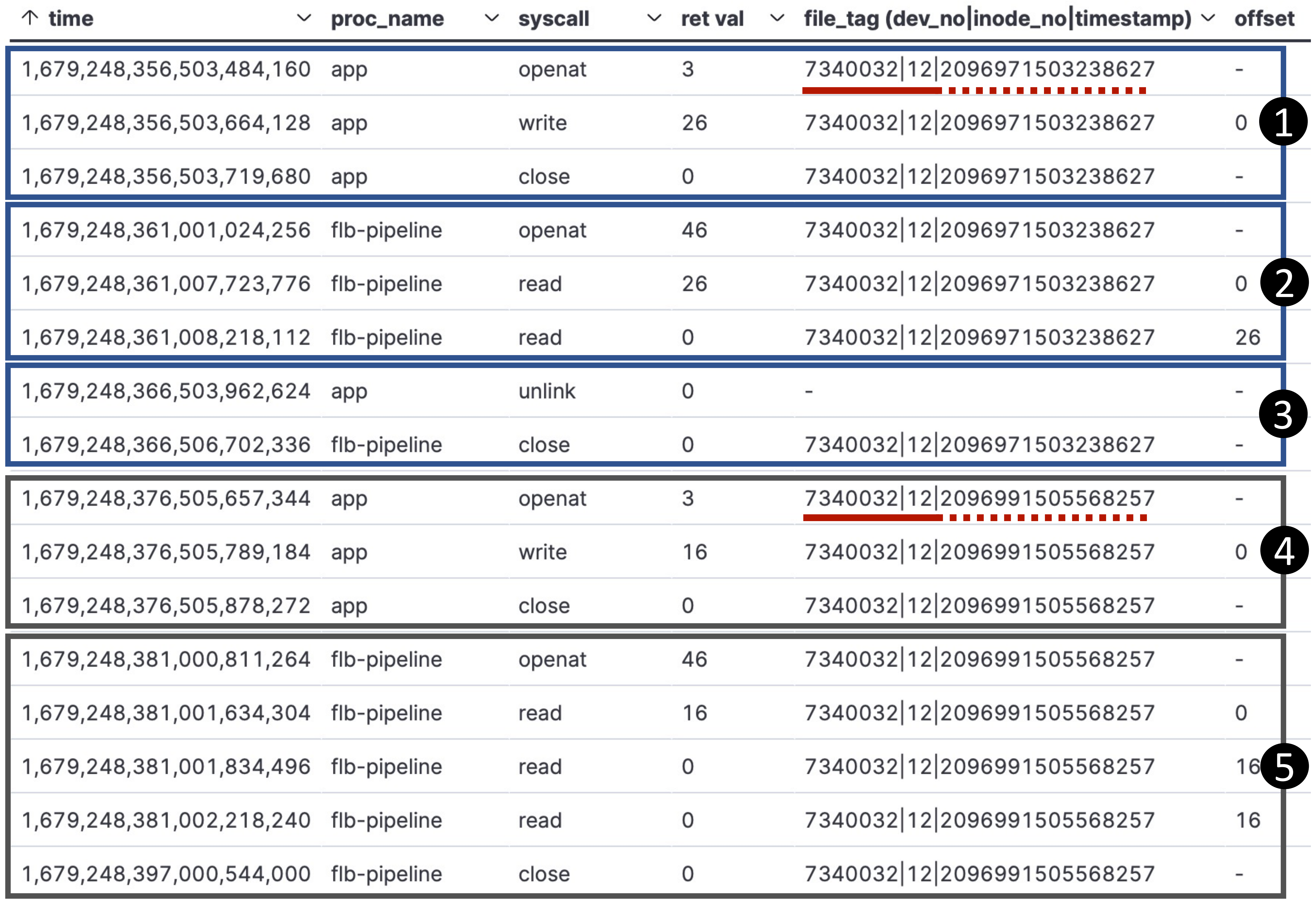
app **removes** the file and *fluent-bit* closes its file descriptor

Evaluation - Fluent Bit (v2.0.5)

Identifying erroneous actions that lead to data loss

Some new content

app.log

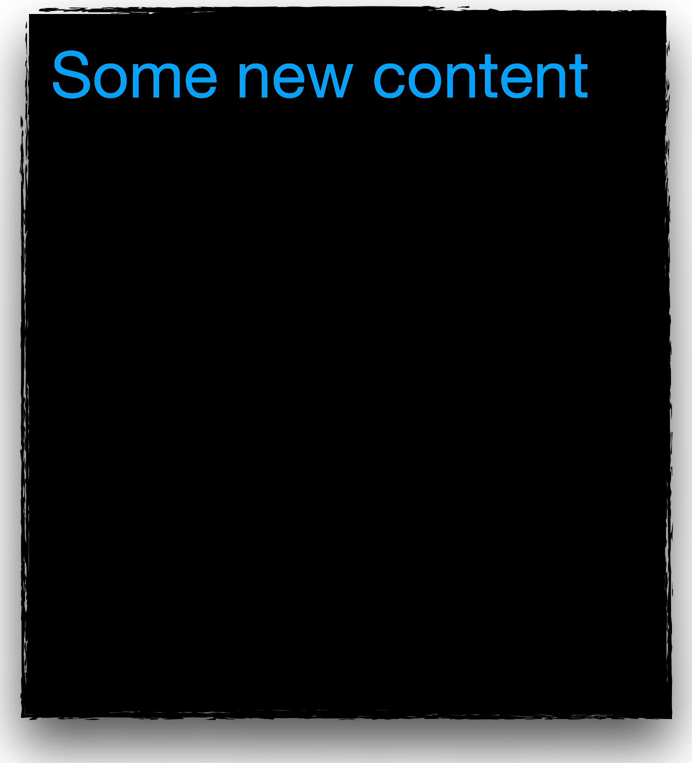


↑ time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,248,356,503,484,160	app	openat	3	7340032|12|2096971503238627	-
1,679,248,356,503,664,128	app	write	26	7340032|12|2096971503238627	0
1,679,248,356,503,719,680	app	close	0	7340032|12|2096971503238627	-
1,679,248,361,001,024,256	flb-pipeline	openat	46	7340032|12|2096971503238627	-
1,679,248,361,007,723,776	flb-pipeline	read	26	7340032|12|2096971503238627	0
1,679,248,361,008,218,112	flb-pipeline	read	0	7340032|12|2096971503238627	26
1,679,248,366,503,962,624	app	unlink	0	-	-
1,679,248,366,506,702,336	flb-pipeline	close	0	7340032|12|2096971503238627	-
1,679,248,376,505,657,344	app	openat	3	7340032|12|2096991505568257	-
1,679,248,376,505,789,184	app	write	16	7340032|12|2096991505568257	0
1,679,248,376,505,878,272	app	close	0	7340032|12|2096991505568257	-
1,679,248,381,000,811,264	flb-pipeline	openat	46	7340032|12|2096991505568257	-
1,679,248,381,001,634,304	flb-pipeline	read	16	7340032|12|2096991505568257	0
1,679,248,381,001,834,496	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,381,002,218,240	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,397,000,544,000	flb-pipeline	close	0	7340032|12|2096991505568257	-

1. *app* opens a file, **writes 26 bytes** from **offset 0** and closes it
2. *fluent-bit* opens the file, **reads 26 bytes** from **offset 0**
3. *app* **removes** the file and *fluent-bit* closes its file descriptor
4. *app* opens a **new file** with **same name** and **inode** number (12), **writes 16 bytes** from **offset 0** and closes the file
5.

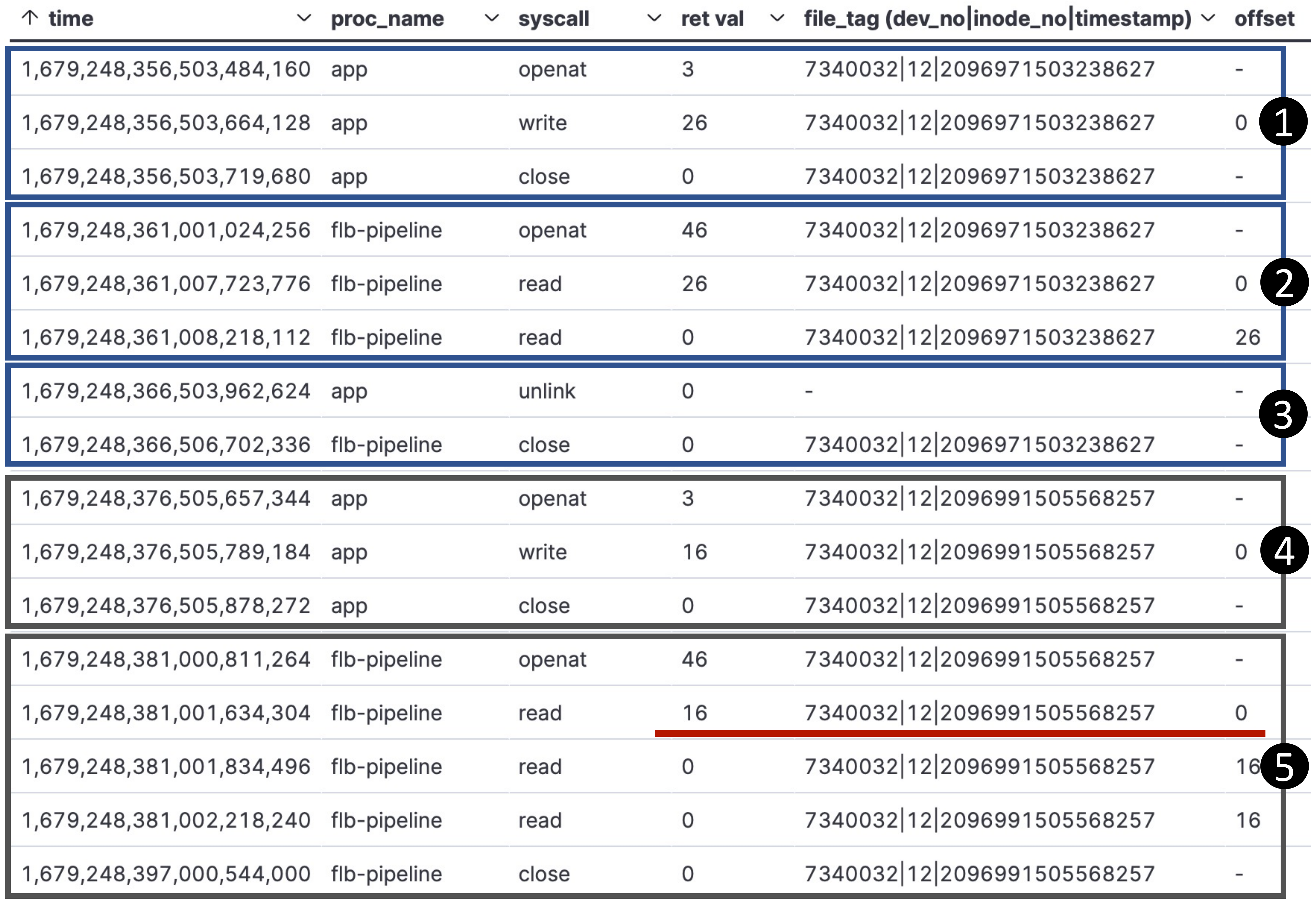
Evaluation - Fluent Bit (v2.0.5)

Identifying erroneous actions that lead to data loss



Some new content

app.log



↑ time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,248,356,503,484,160	app	openat	3	7340032|12|2096971503238627	-
1,679,248,356,503,664,128	app	write	26	7340032|12|2096971503238627	0
1,679,248,356,503,719,680	app	close	0	7340032|12|2096971503238627	-
1,679,248,361,001,024,256	flb-pipeline	openat	46	7340032|12|2096971503238627	-
1,679,248,361,007,723,776	flb-pipeline	read	26	7340032|12|2096971503238627	0
1,679,248,361,008,218,112	flb-pipeline	read	0	7340032|12|2096971503238627	26
1,679,248,366,503,962,624	app	unlink	0	-	-
1,679,248,366,506,702,336	flb-pipeline	close	0	7340032|12|2096971503238627	-
1,679,248,376,505,657,344	app	openat	3	7340032|12|2096991505568257	-
1,679,248,376,505,789,184	app	write	16	7340032|12|2096991505568257	0
1,679,248,376,505,878,272	app	close	0	7340032|12|2096991505568257	-
1,679,248,381,000,811,264	flb-pipeline	openat	46	7340032|12|2096991505568257	-
1,679,248,381,001,634,304	flb-pipeline	read	16	7340032|12|2096991505568257	0
1,679,248,381,001,834,496	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,381,002,218,240	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,397,000,544,000	flb-pipeline	close	0	7340032|12|2096991505568257	-

1 *app* opens a file, **writes 26 bytes** from **offset 0** and closes it

2 *fluent-bit* opens the file, **reads 26 bytes** from **offset 0**

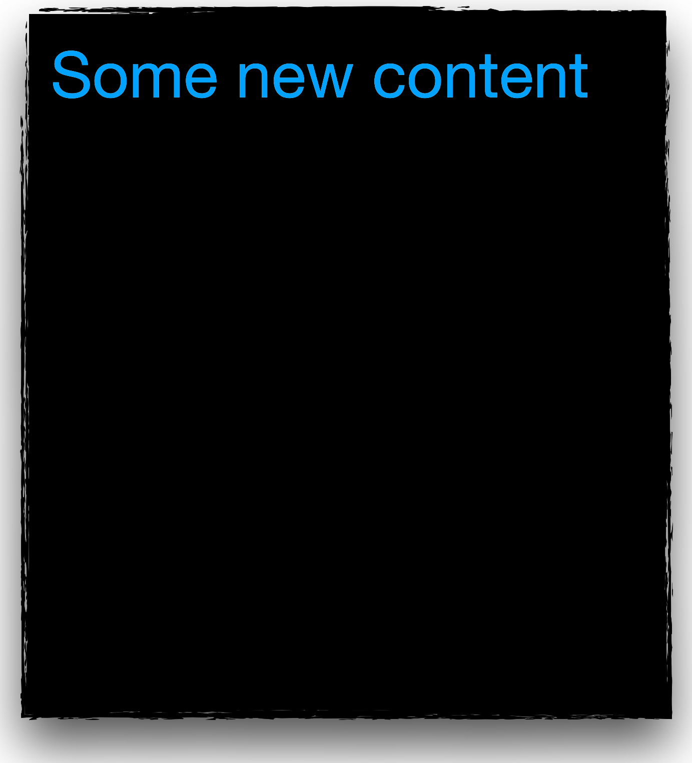
3 *app* **removes** the file and *fluent-bit* closes its file descriptor

4 *app* opens a **new file** with **same name** and **inode** number (12), **writes 16 bytes** from **offset 0** and closes the file

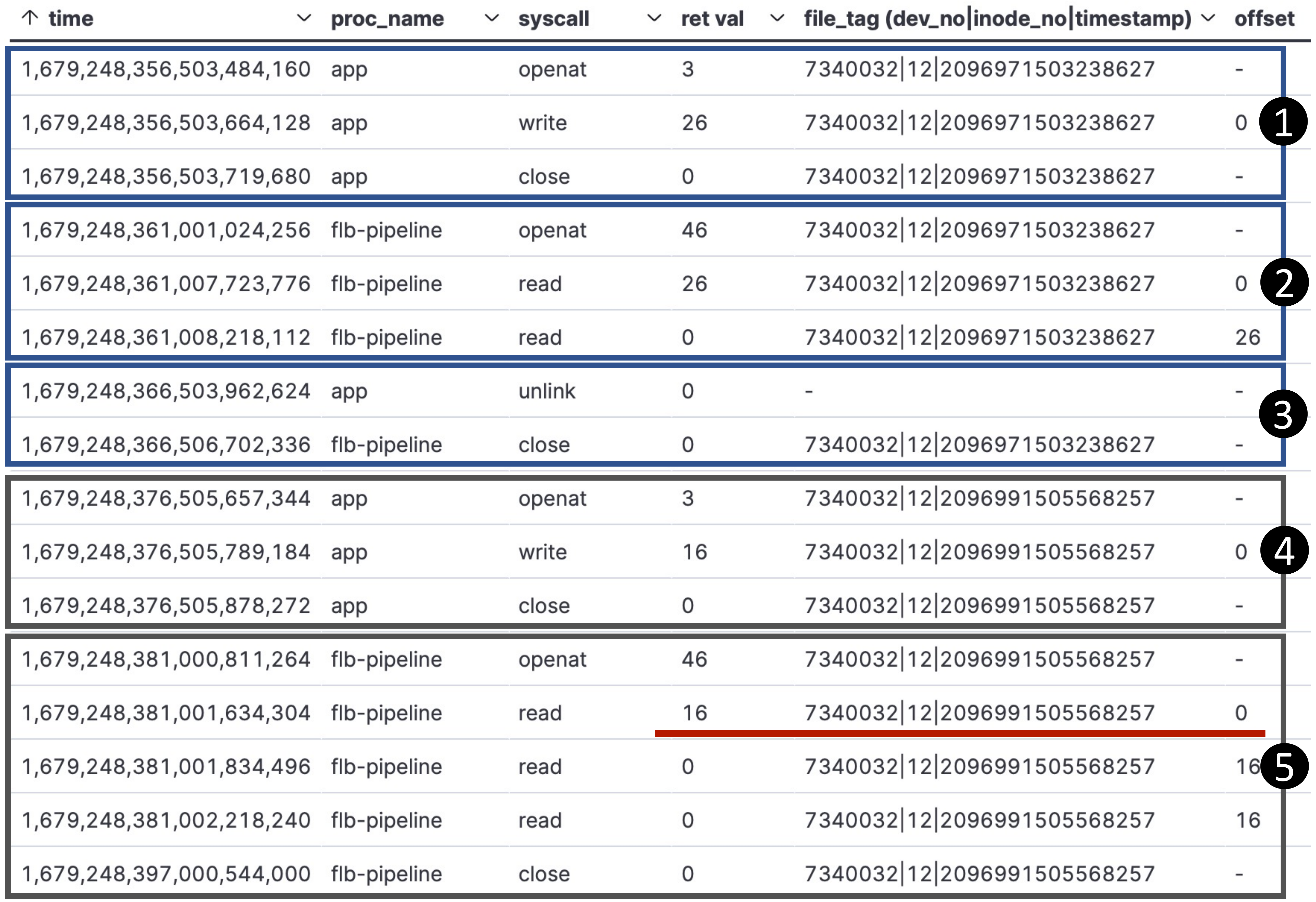
5 *fluent-bit* opens new file and **reads 16 bytes** from **offset 0**

Evaluation - Fluent Bit (v2.0.5)

Identifying erroneous actions that lead to data loss



app.log



↑ time	proc_name	syscall	ret val	file_tag (dev_no|inode_no|timestamp)	offset
1,679,248,356,503,484,160	app	openat	3	7340032|12|2096971503238627	-
1,679,248,356,503,664,128	app	write	26	7340032|12|2096971503238627	0
1,679,248,356,503,719,680	app	close	0	7340032|12|2096971503238627	-
1,679,248,361,001,024,256	flb-pipeline	openat	46	7340032|12|2096971503238627	-
1,679,248,361,007,723,776	flb-pipeline	read	26	7340032|12|2096971503238627	0
1,679,248,361,008,218,112	flb-pipeline	read	0	7340032|12|2096971503238627	26
1,679,248,366,503,962,624	app	unlink	0	-	-
1,679,248,366,506,702,336	flb-pipeline	close	0	7340032|12|2096971503238627	-
1,679,248,376,505,657,344	app	openat	3	7340032|12|2096991505568257	-
1,679,248,376,505,789,184	app	write	16	7340032|12|2096991505568257	0
1,679,248,376,505,878,272	app	close	0	7340032|12|2096991505568257	-
1,679,248,381,000,811,264	flb-pipeline	openat	46	7340032|12|2096991505568257	-
1,679,248,381,001,634,304	flb-pipeline	read	16	7340032|12|2096991505568257	0
1,679,248,381,001,834,496	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,381,002,218,240	flb-pipeline	read	0	7340032|12|2096991505568257	16
1,679,248,397,000,544,000	flb-pipeline	close	0	7340032|12|2096991505568257	-

1 *app* opens a file, **writes 26 bytes** from **offset 0** and closes it

2 *fluent-bit* opens the file, **reads 26 bytes** from **offset 0**

3 *app* **removes** the file and *fluent-bit* closes its file descriptor

4 *app* opens a **new file** with **same name** and **inode** number (12), **writes 16 bytes** from **offset 0** and closes the file

5 *fluent-bit* opens new file and **reads 16 bytes** from **offset 0**

Correct access pattern!

Evaluation - Fluent Bit

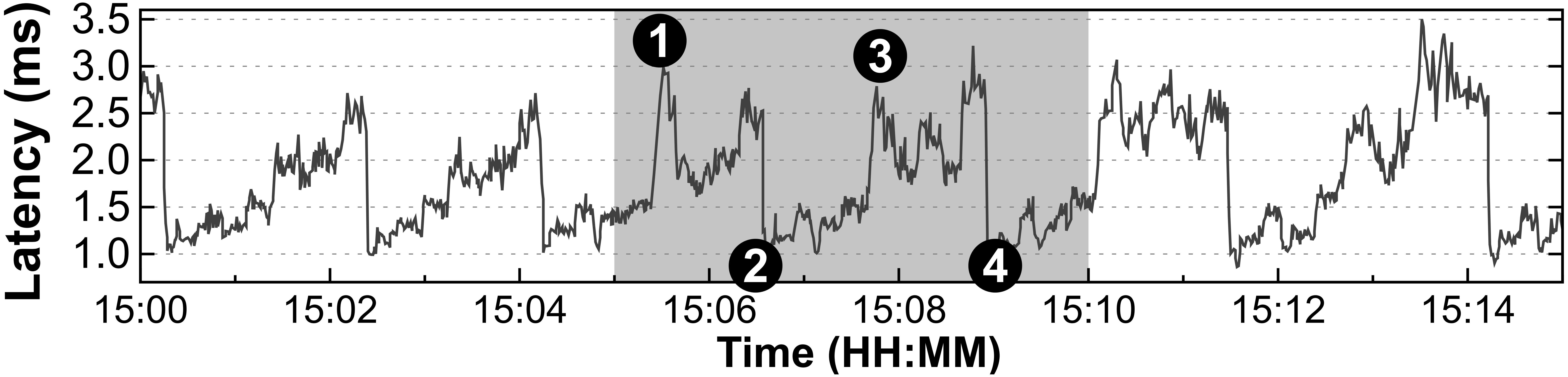
Identifying erroneous actions that lead to data loss

- ◉ **DIO helps users diagnose** incorrect I/O behavior from applications and find the root cause for dependability issues such as data loss
- ◉ **DIO helps validate** the corrections applied to the applications' implementation

Evaluation - RocksDB

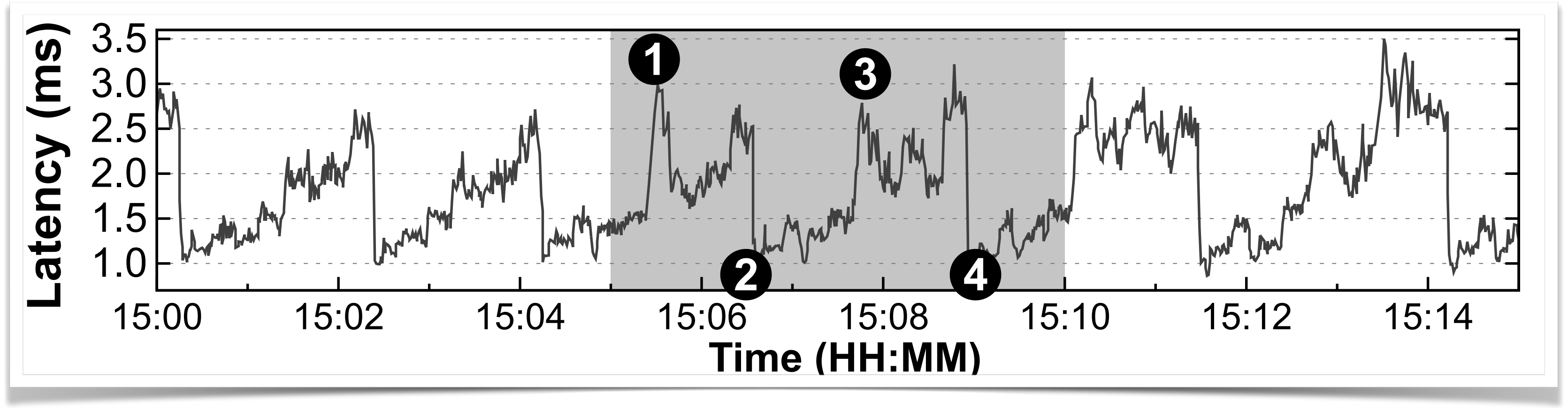
Finding the root cause of performance anomalies

- **RocksDB:** an embedded key-value store
- **Problem:** RocksDB clients observe high tail latencies (1 & 3)
  - Reproducible with db_bench benchmark

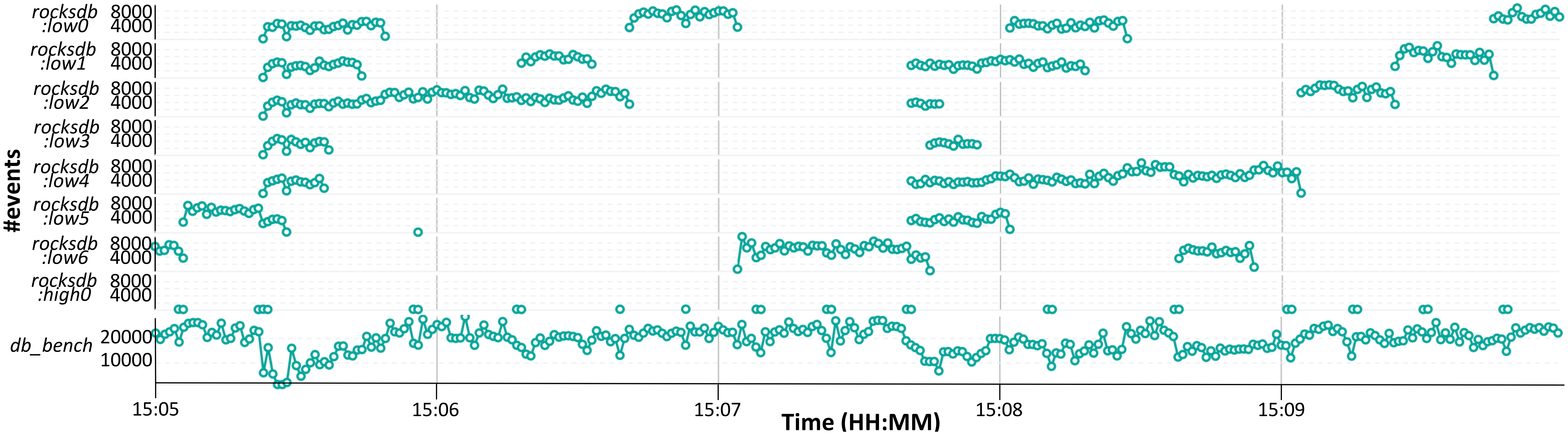


99th percentile latency for RocksDB client operations.

Evaluation - RocksDB

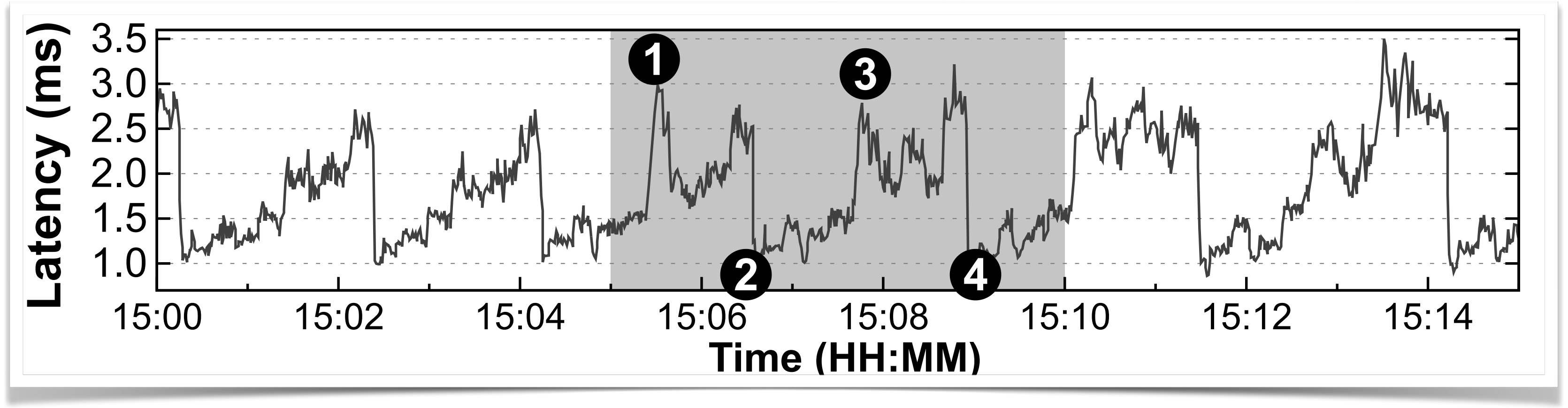


Finding the root cause of performance anomalies

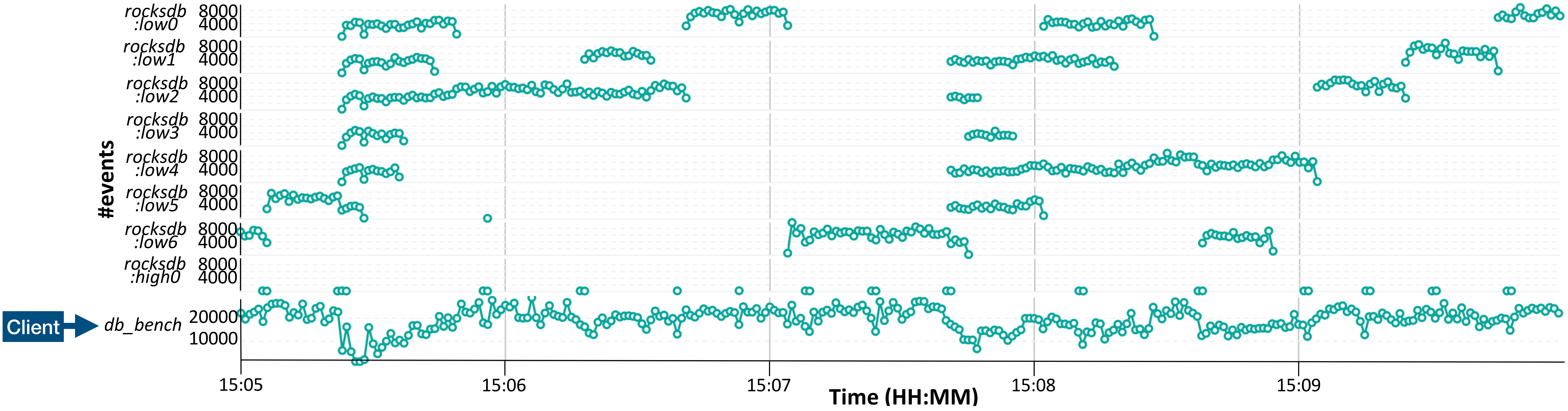


Syscalls issued by RocksDB over time, aggregated by thread name.

Evaluation - RocksDB



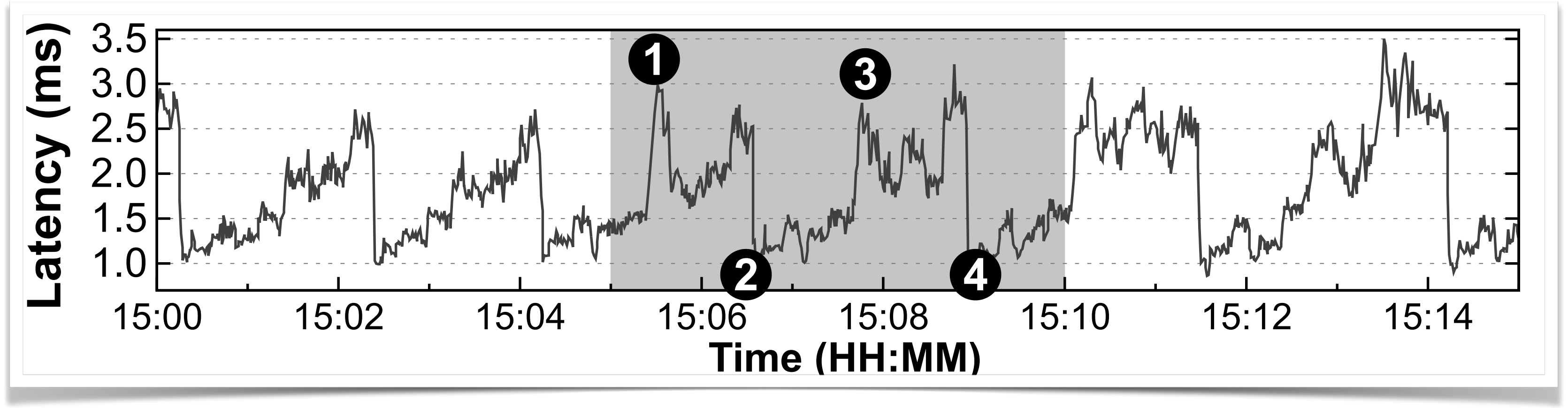
Finding the root cause of performance anomalies

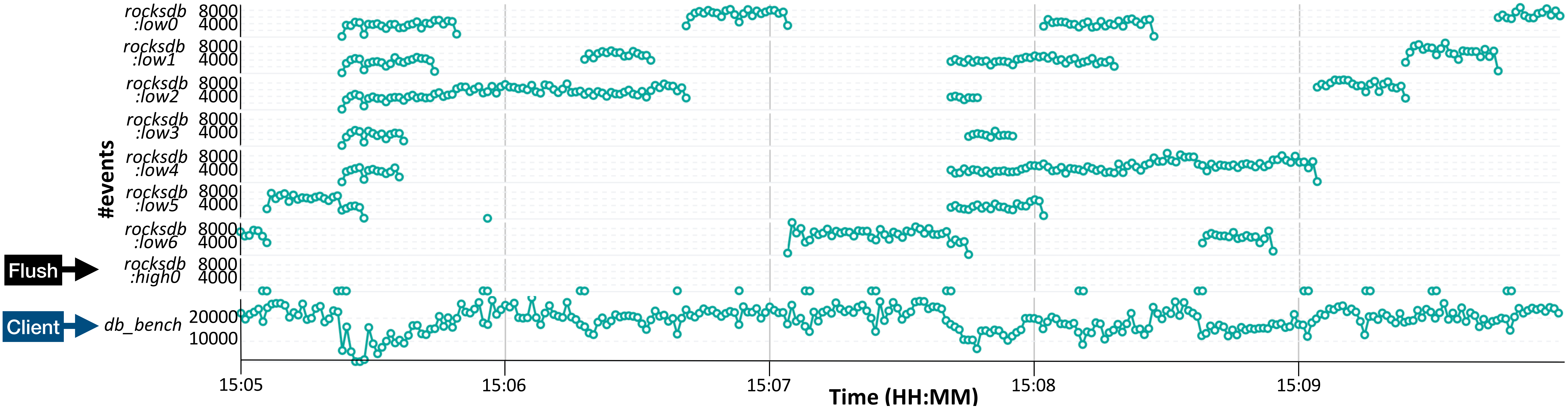


Syscalls issued by RocksDB over time, aggregated by thread name.

Evaluation - RocksDB

Finding the root cause of performance anomalies

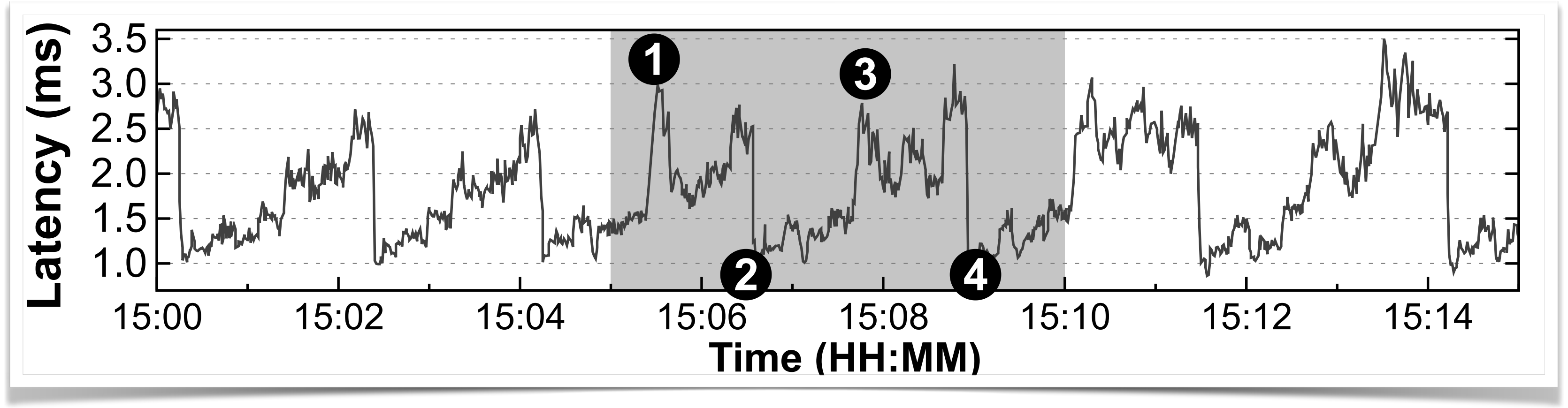


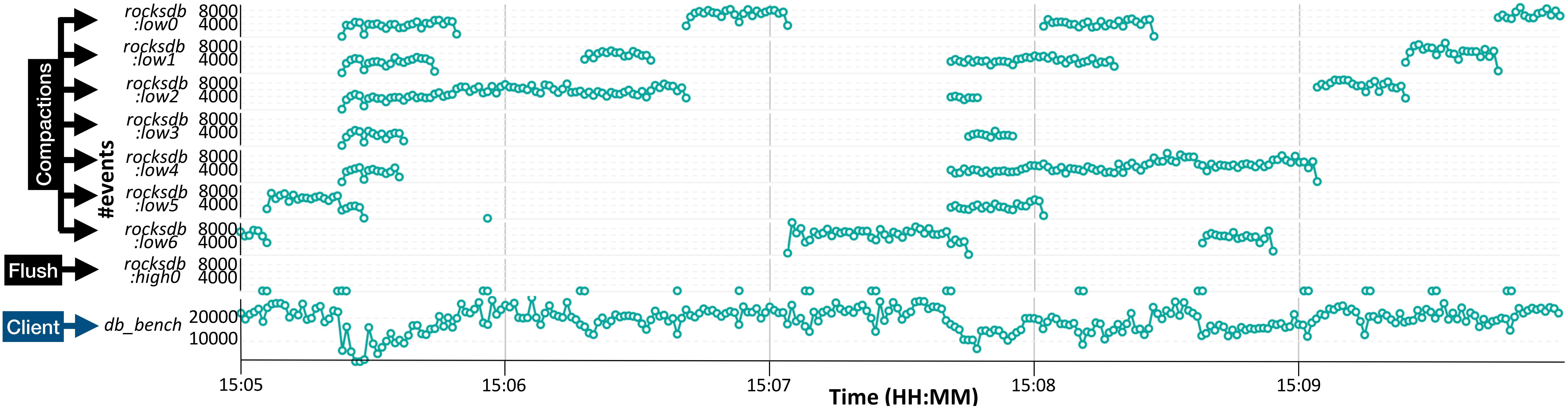


Syscalls issued by RocksDB over time, aggregated by thread name.

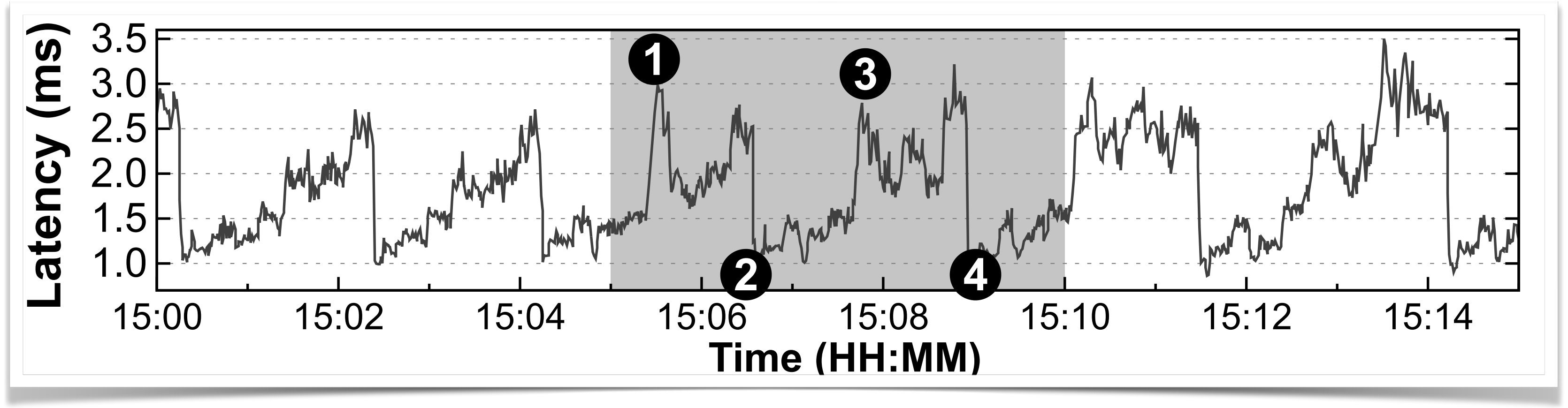
Evaluation - RocksDB

Finding the root cause of performance anomalies



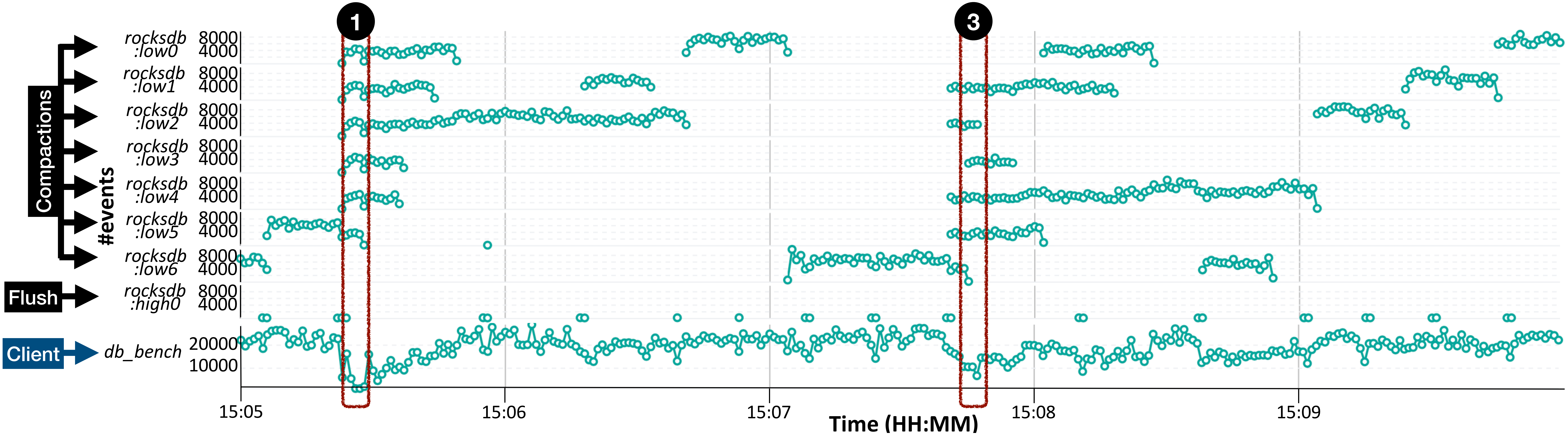


Syscalls issued by RocksDB over time, aggregated by thread name.



Evaluation - RocksDB

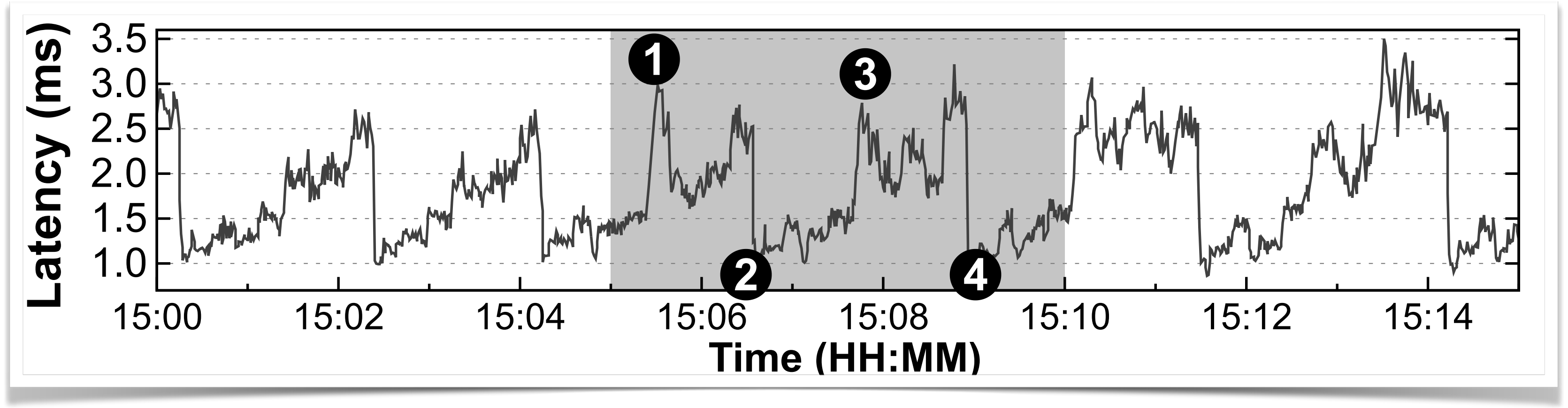
Finding the root cause of performance anomalies



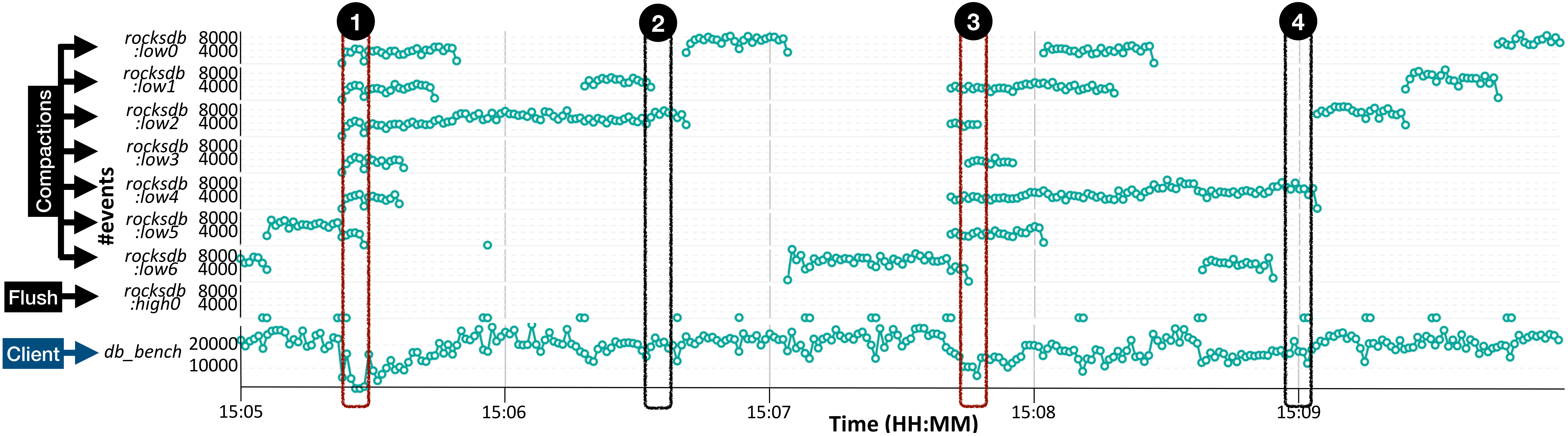
Syscalls issued by RocksDB over time, aggregated by thread name.

- (1&3) multiple background threads perform I/O simultaneously, db_bench performance decreases

Evaluation - RocksDB



Finding the root cause of performance anomalies



Syscalls issued by RocksDB over time, aggregated by thread name.

- (1&3) multiple background threads perform I/O simultaneously, db_bench performance decreases
- (2&4) few background threads perform I/O simultaneously, db_bench performance improves

Evaluation - RocksDB

Finding the root cause of performance anomalies

- **Root cause:** Latency spikes occur when threads compete for shared disk bandwidth, leading to performance contention

- This is the phenomenon identified in SILK[1] and **observable with DIO without** any **code instrumentation**

[1] BALMAU, Oana, et al. SILK: Preventing Latency Spikes in Log-Structured Merge Key-Value Stores. In: USENIX Annual Technical Conference. 2019. p. 753-766.

Conclusion

- DIO is a generic tool for observing and diagnosing I/O interactions between applications and in-kernel POSIX storage systems
- Helps observe I/O issues, find their root causes and validate their fixes
- Experiments, with two widely-used systems, show that DIO enables
  - observing erroneous I/O access patterns that lead to data loss
  - identifying I/O contention that leads to high tail latency

Future directions

- Simplify analysis with new automated correlation algorithms
- Explore other applications for uncovering new I/O issues
- Further analyze DIO’s performance overhead and explore new optimizations

DIO

Diagnosing applications’ I/O behavior through system call observability

- DIO is publicly available at
  - **Github**: github.com/dsrhaslab/dio
  - **Website**: dio-tool.netlify.app
  - **Contact**: tania.c.araujo@inesctec.pt

Diagnosing applications' I/O behavior through system call observability

Tânia Esteves, Ricardo Macedo, Rui Oliveira and João Paulo

INESC TEC & University of Minho

5th Workshop on Data-Centric Dependability and Security (DCDS'23)

INESCTEC

Universidade do Minho

FCT Fundação para a Ciência e a Tecnologia