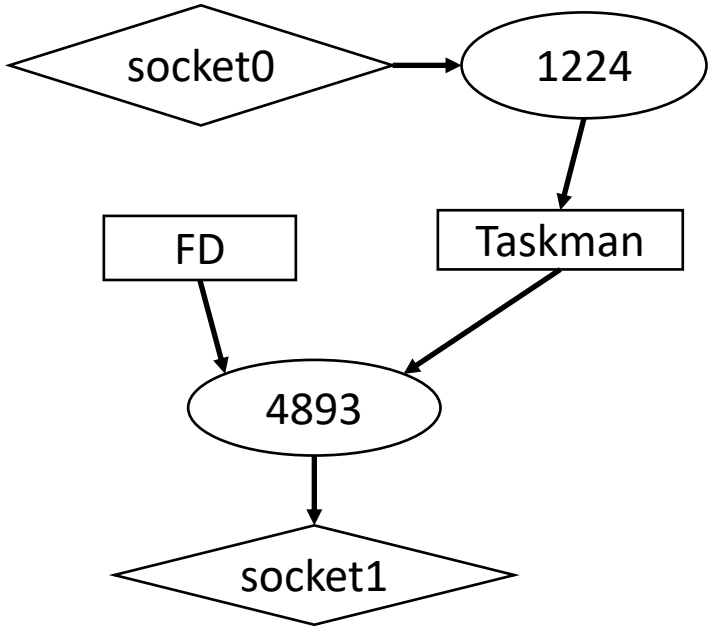
KCAL: Kernel-support Cost-effective Audit Logging for Causality Tracking

Shiqing Ma, Juan Zhai, Yonghwi Kwon, Kyu Hyung Lee, Xiangyu Zhang, Gabriela Ciocarlie, Ashish Gehani, Vinod Yegneswaran, Dongyan Xu, Somesh Jha



Background: Forensics Analysis with Audit Log

1. ……
2. *PID=1224*, Receives from *socket0*
3. *PID=1224*, Writes to File *Taskman*
4. ……
5. *PID=4893,* Starts from File *Taskman*
6. *PID=4893*, Reads file *FD*
7. *PID=4893*, Sends data to *socket1*
8. ……



Linux Audit Framework

Most Popular Audit Framework on Linux

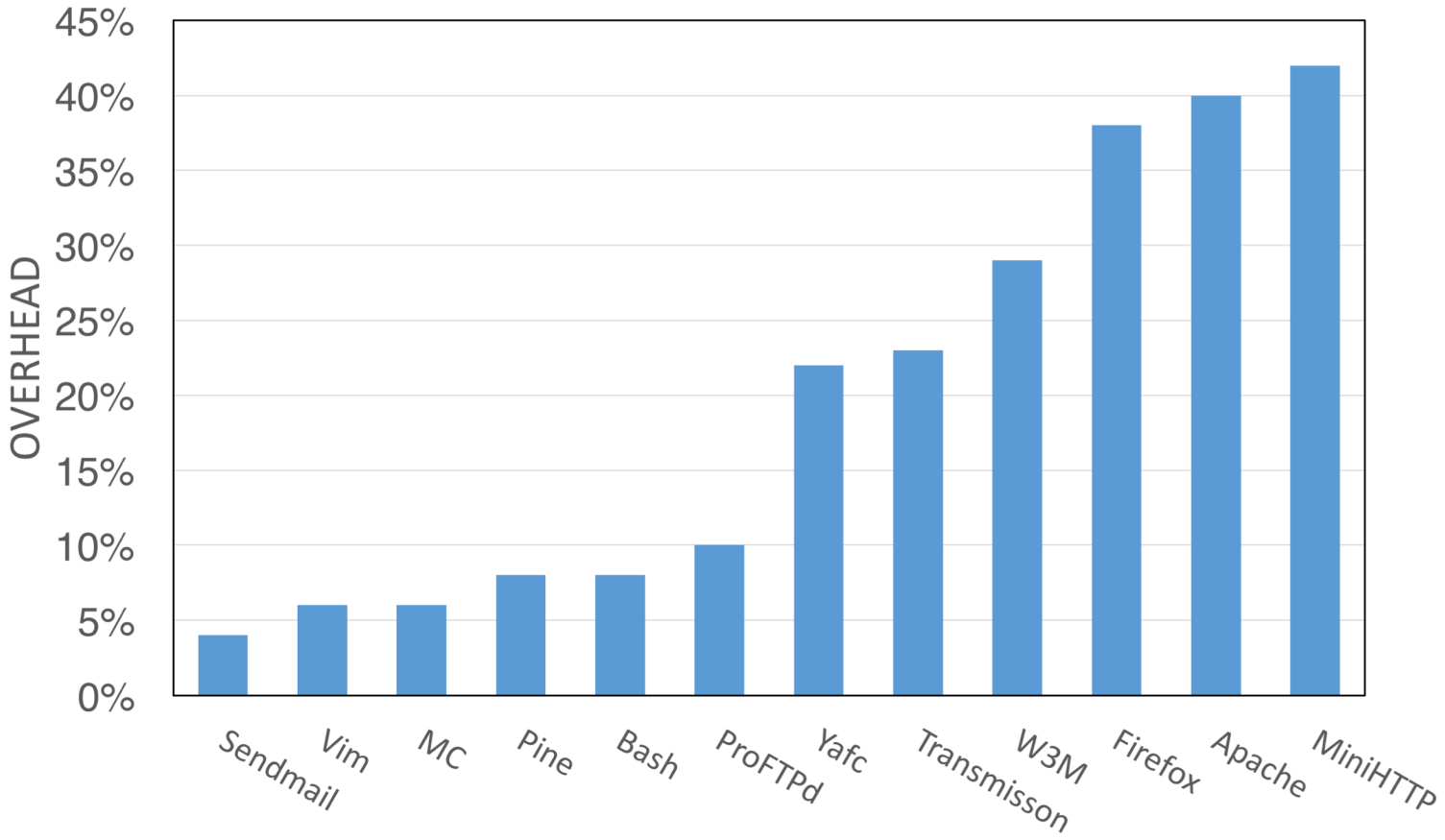
- Integrated with many other tools, e.g., Orchids, Prelude siem
- Kernel component is shipped with mainline kernel

The Linux Audit Framework

- Linux kernel component + user-space tools (e.g., Auditd, go-audit)
- Logging all system calls, monitoring file accesses etc.

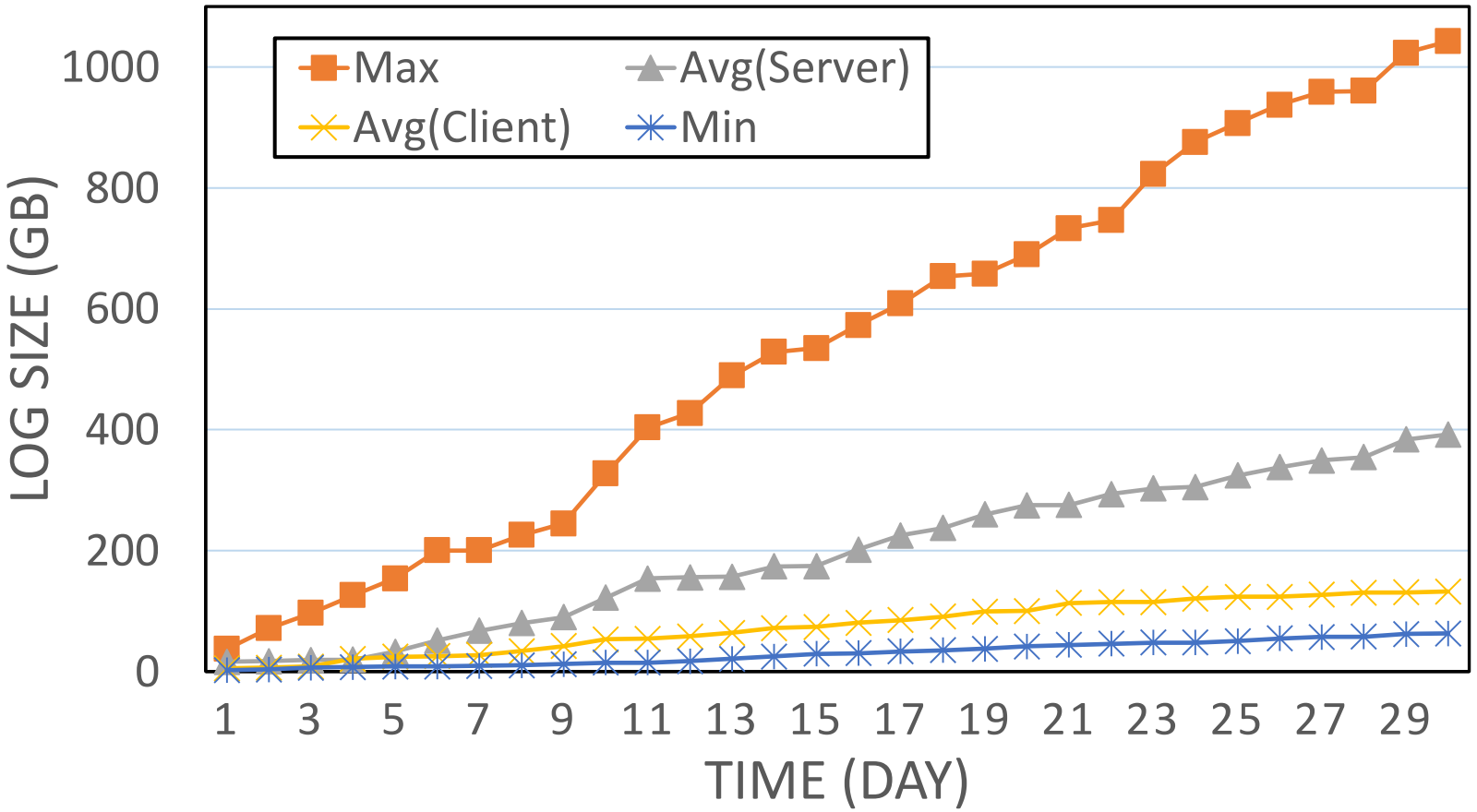
Problems: 1) Slow. 2) Large log files.

RT Overhead: 40% for some programs



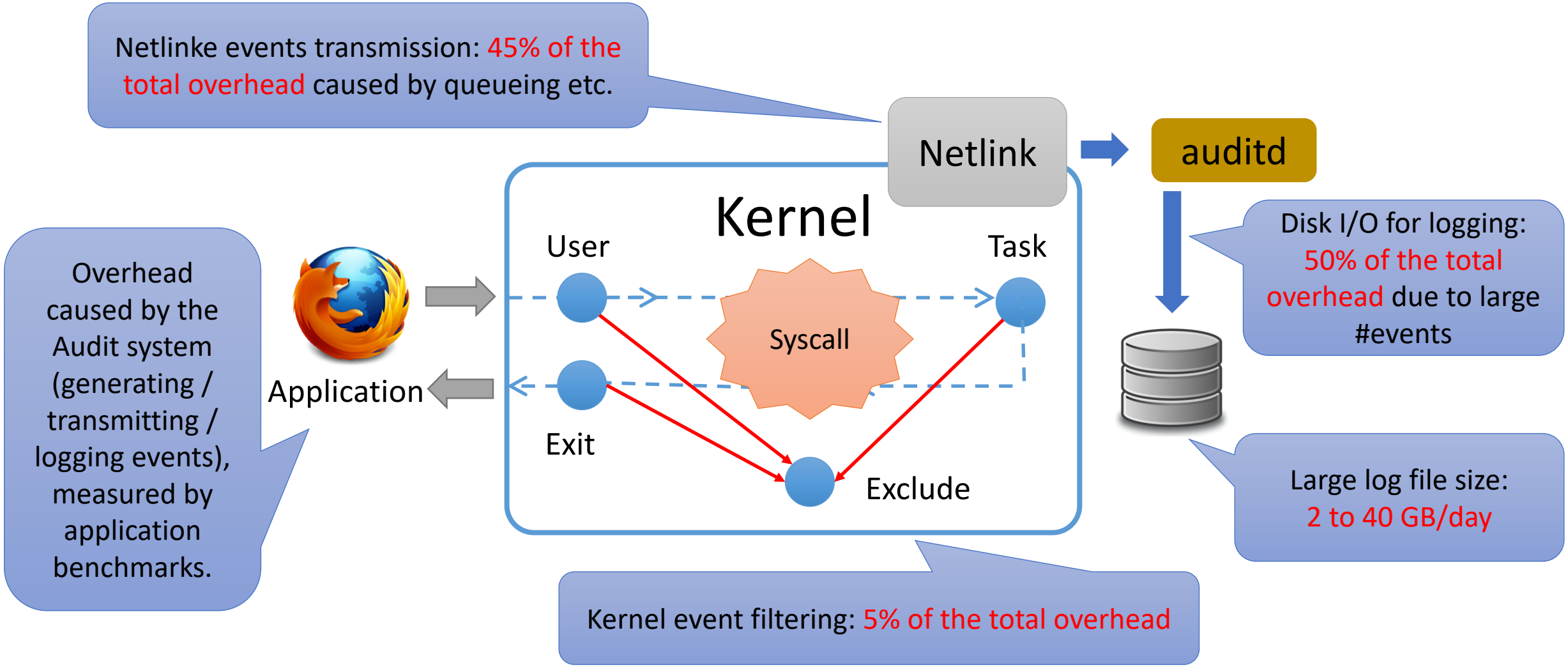


Space Overhead: 2 to 40 GB/day





Linux Audit System Overhead (Comparing with Linux with Audit disabled)



Problem: # Logged events is large



logged events → large log files



Large files are hard to investigate.

logged events → filter/transfer/writing



High RT overhead makes system not useable.

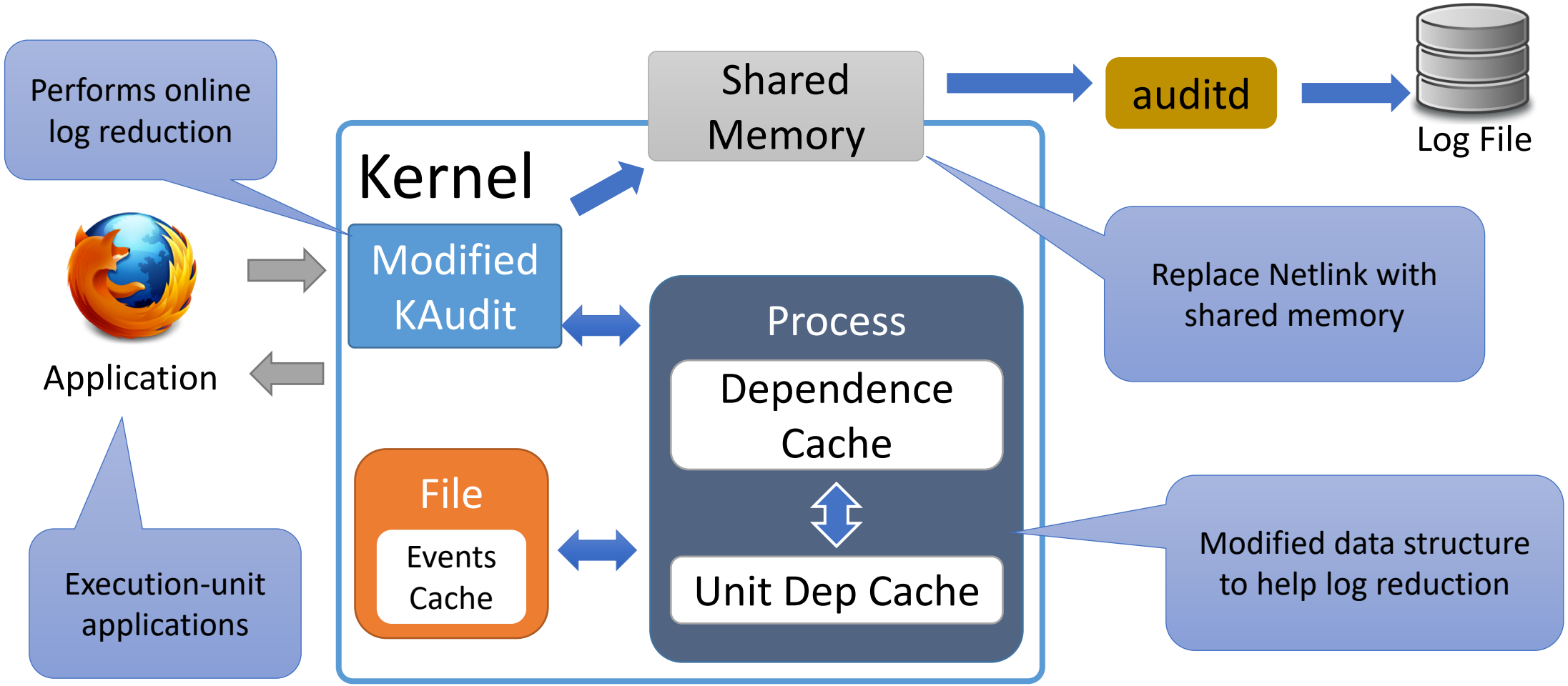
Question: is reducing the # log events possible?

Log Redundancy

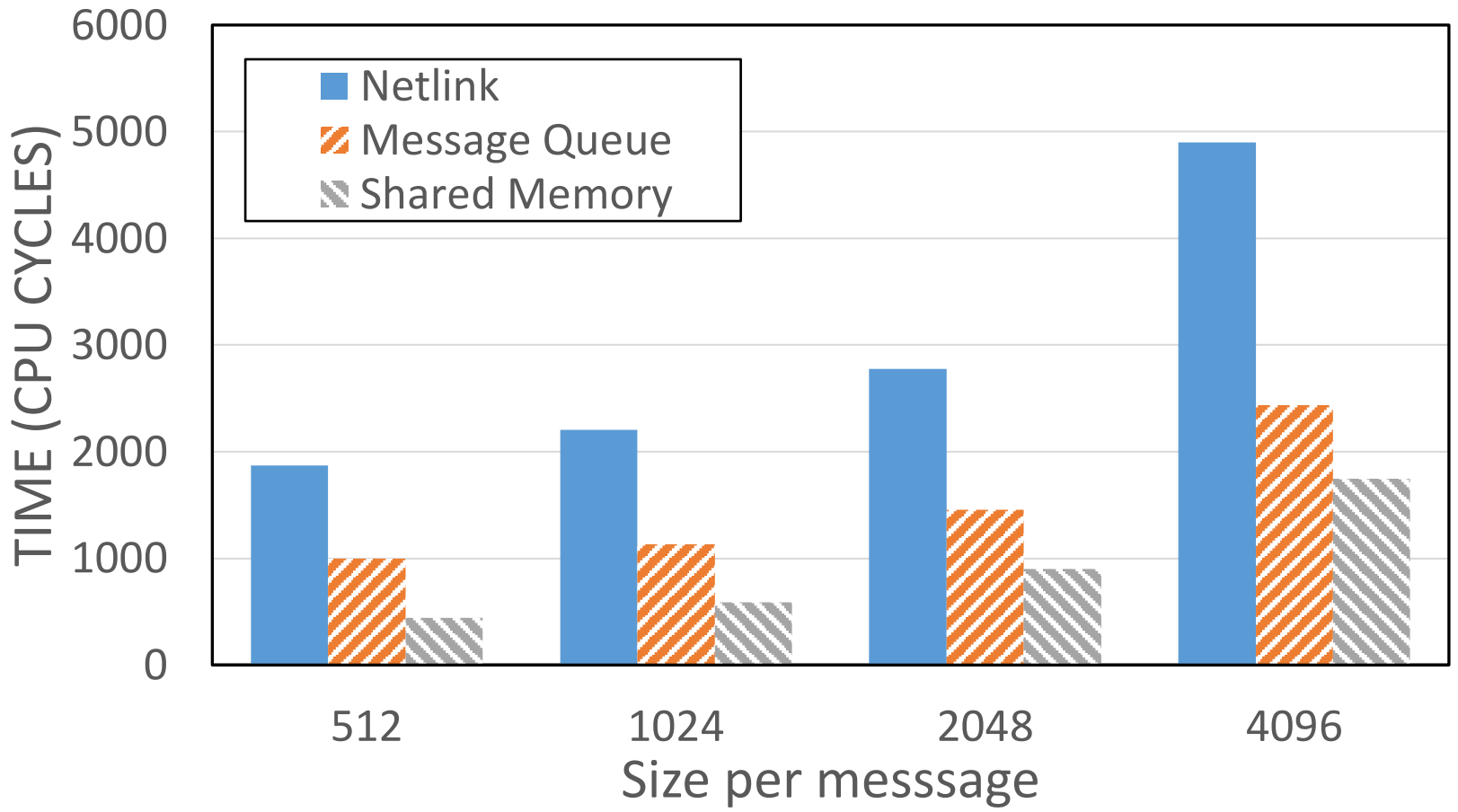
- Audit log is redundant for forensics analysis
- Redundant events: event that represent the same dependency relationships (from the forensics point of view)

```
1. PID=422, Event = Read (FD4)
2. PID=422, Event = Read (FD4)
3. PID=422, Event = Read (FD4)
4. PID=422, Event = Read (FD4)
5. PID=442, Event = Write(FD5)
6. PID=442, Event = Read (FD4)
7. PID=442, Event = Write(FD5)
8. PID=442, Event = Write(FD5)
9. PID=442, Event = Write(FD5)
```

KCAL: Do not generate redundant events.

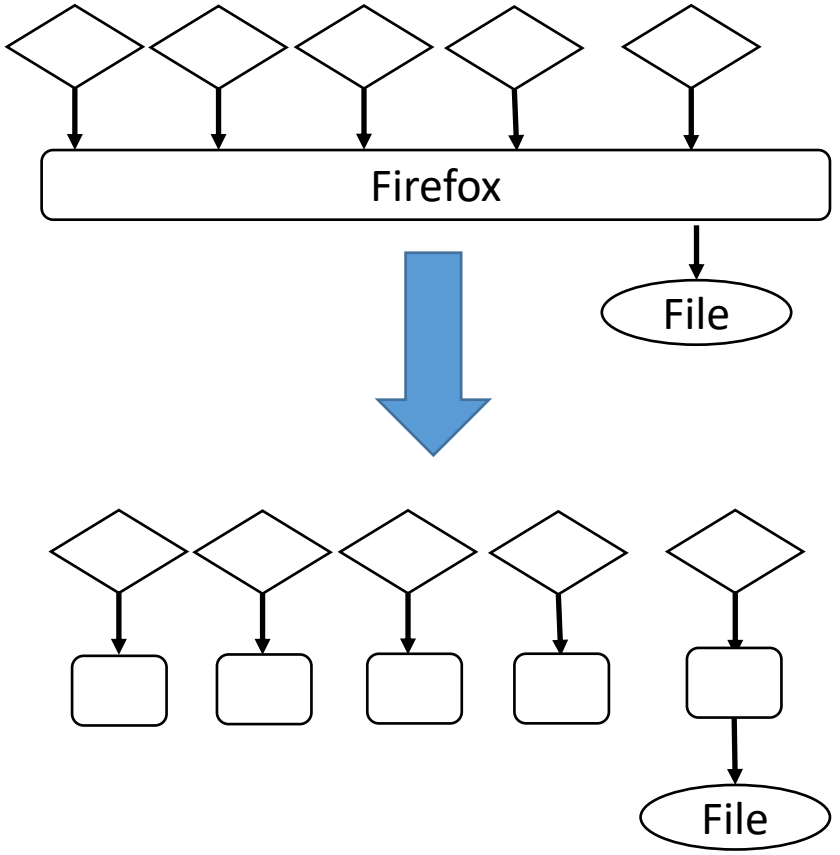


Data Channel: Shared Memory



Dependence Analysis: Execution-unit Based

- Dependence analysis is hard in forensics analysis
  - Dependence explosion
- Execution-unit
  - Part of a process execution
- More details, see
  - BEEP (NDSS'13)
  - MPI (USENIX Security'17, ***Distinguished Paper Award***)



Log Reduction: In-unit Redundancy

```
1. PID=442, Event=UNIT_ENTER
2. PID=422, Event=Read (FD4)
3. PID=422, Event=Read (FD4)
4. PID=422, Event=Read (FD4)
5. PID=422, Event=Read (FD4)
6. PID=422, Event=Read (FD4)
7. PID=422, Event=Read (FD4)
8. PID=422, Event=Read (FD4)
9. PID=422, Event=Read (FD4)
10.PID=442, Event=UNIT_EXIT
```

- The same operation(s) on the same object within the same unit
- Case: Vim loading a file
- Application logic or limited buffer size

```
while(content=read(fd) > 0) {
        insert_content(root, content)
}
```

Log Reduction: Cross-unit Redundancy

```
1. PID=442, Event=UNIT_ENTER
2. PID=422, Event=Read (FD4)
3. PID=422, Event=Read (FD4)
4. PID=422, Event=Write(FD5)
5. PID=442, Event=UNIT_EXIT
6. PID=442, Event=UNIT_ENTER
7. PID=422, Event=Read (FD4)
8. PID=422, Event=Read (FD4)
9. PID=422, Event=Write(FD5)
10.PID=442, Event=UNIT_EXIT
```

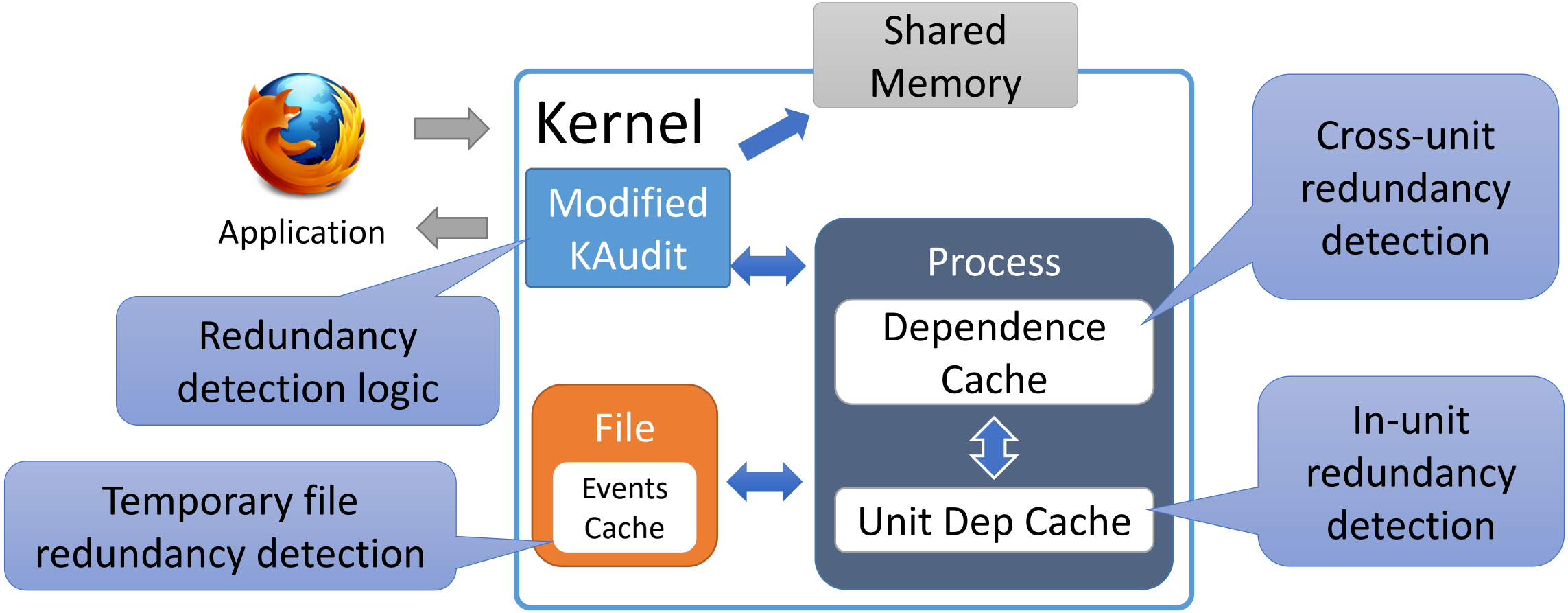
- Different units are doing the same thing
- Case: repeated operations
  - Vim save file
- Tracking to one of them is sufficient to build the graph

Log Reduction: Temporary Files

```
1. PID=442, Event=UNIT_ENTER
2. PID=422, Event=NewFD(FD5)
3. PID=422, Event=Write(FD5)
4. PID=442, Event=UNIT_EXIT
5. PID=442, Event=UNIT_ENTER
6. PID=422, Event=Write(FD5)
7. PID=442, Event=UNIT_EXIT
8. PID=442, Event=UNIT_ENTER
9. PID=422, Event=Delete(FD5)
10.PID=442, Event=UNIT_EXIT
```

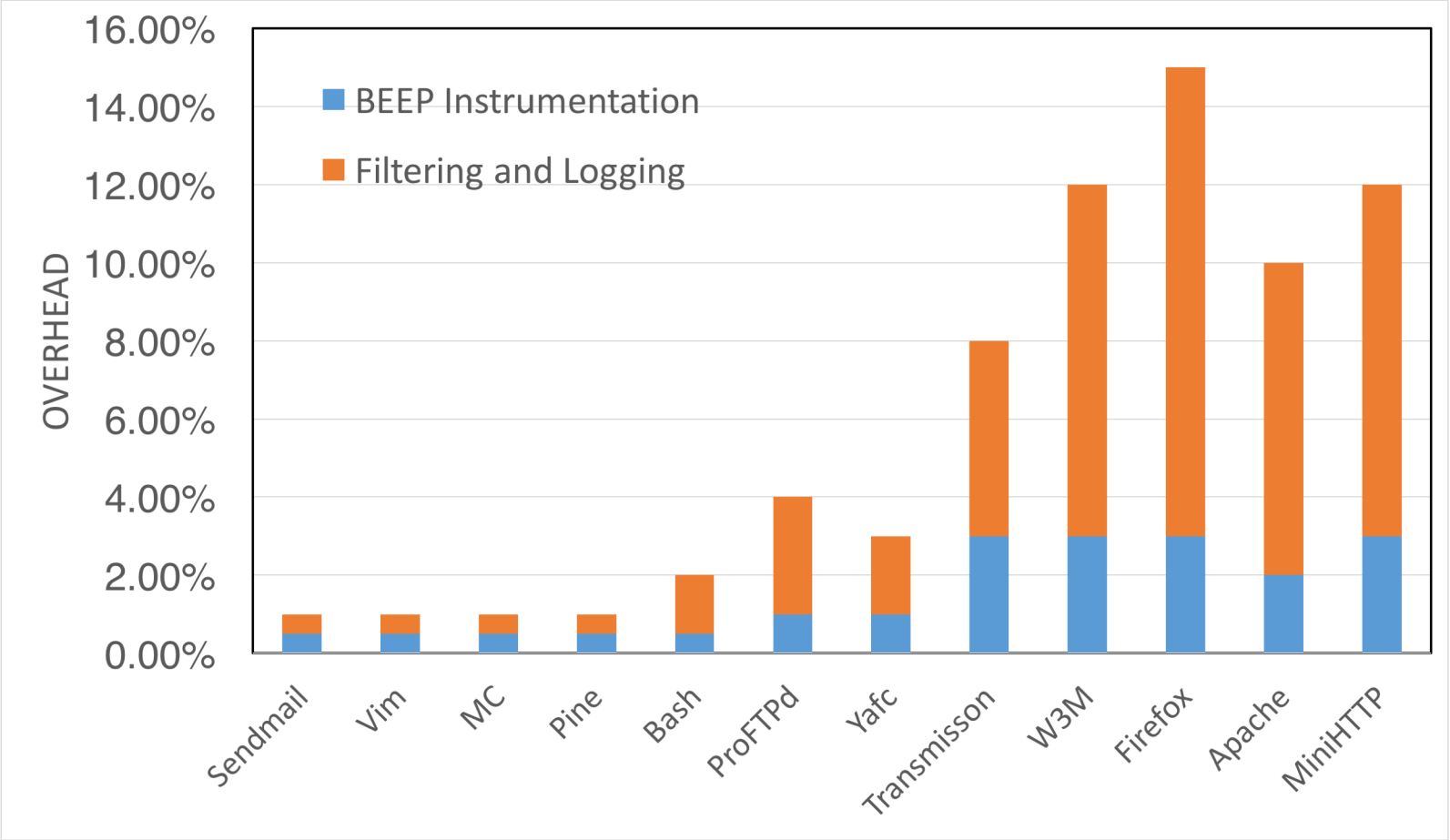
- Temporary file
  - Created/operated/deleted by the same process
- Case: web resources
  - If not stored explicitly, most will be deleted later
- Files that are too large for memory

KCAL: Log Redundancy Detection Components

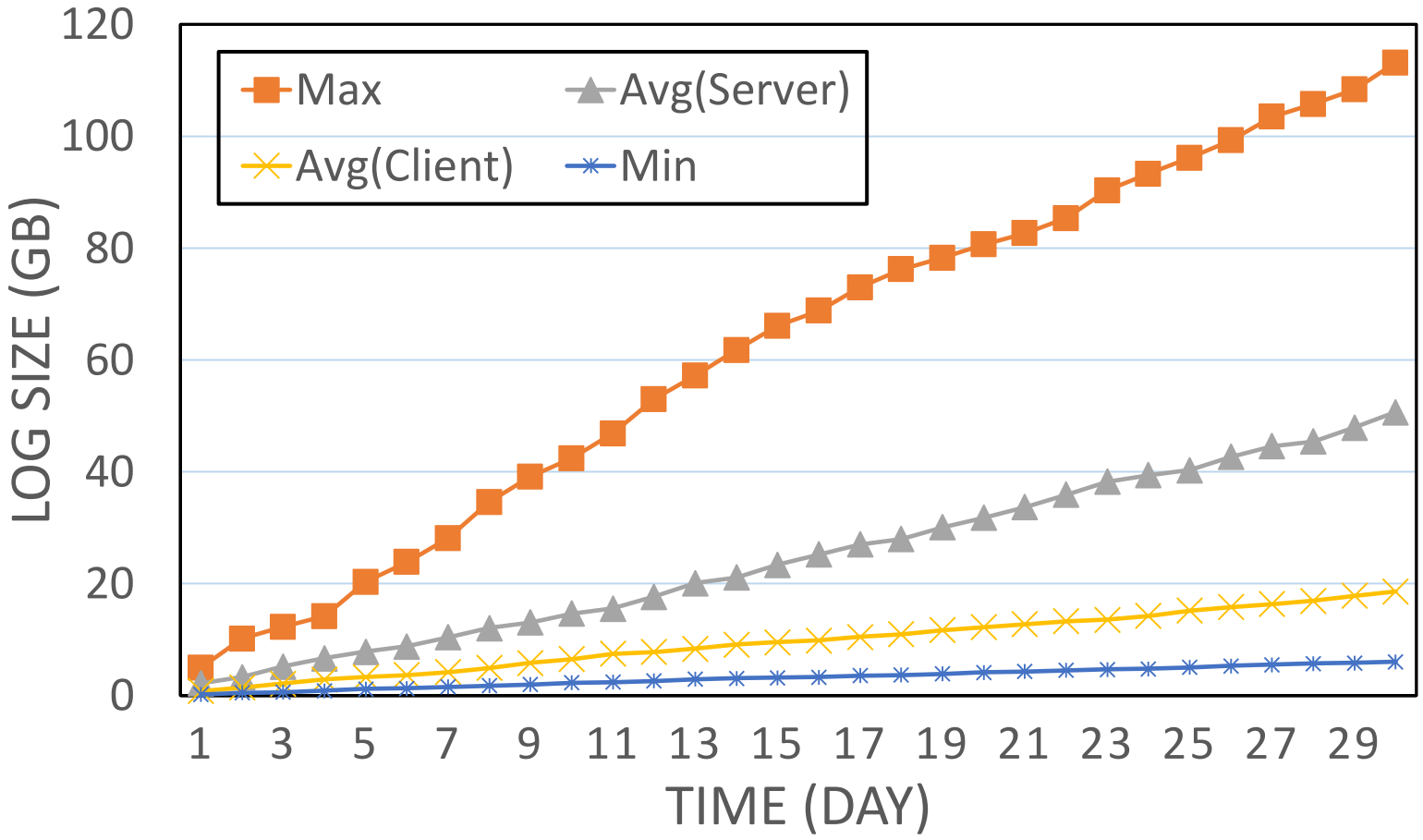


Idea: Dependency relationships are cached.

KCAL-RT Overhead: <16%



KCAL-Space Overhead: <4GB/day



KCAL: Redundancy Analysis

	In-Unit Redundency	Cross-Unit Redundancy	Temporary Files	KCAL Events
Machine 1	69%	7%	16%	8%
Machine 2	71%	10%	9%	10%
Machine 3	56%	10%	21%	13%
Machine 4	21%	46%	24%	9%
Machine 5	29%	43%	13%	14%
Avg.	51%	21%	16%	11%

More evaluation results in paper !

Discussion

- KCAL is for forensics analysis only
  - Linux Audit framework is a general audit framework
- KCAL requires instrumented applications to support execution-unit based log reduction
  - Leveraging other online log reduction algorithms is also possible
- KCAL modifies the kernel source code
  - Porting from 3.19 to 3.2 requires an addition 8 line patch

Summary

- KCAL: Kernel-support Cost-effective Audit Logging for Causality Tracking
  - Based on the Linux Audit Framework
  - For forensics analysis
  - Low runtime, space overhead
  - Reduce redundant events before generating them
    - Reduces overhead caused by transferring, writing, storing redundant events

THANK YOU!

Q&A

