Chi's Trivial OOM Killer

A low-overhead, low-latency and fault-tolerant OOM killer

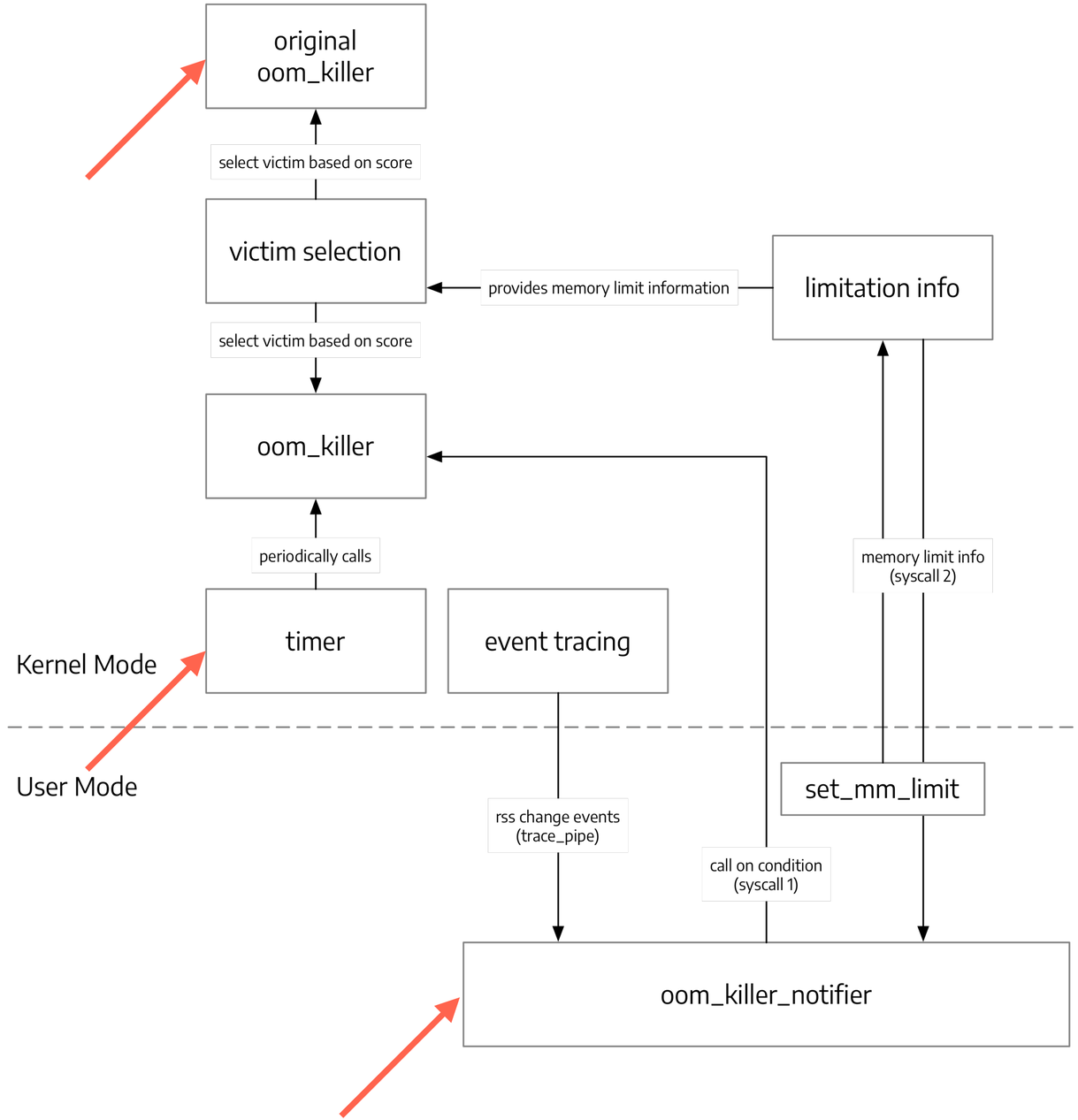
Chi Zhang, aka. Alex Chi

Common Approaches

- Iterate all processes when there's a new page allocated
  - Problem: slow (200% performance regression)
- Kernel timer or user-space daemon
  - Problem: latency

Overall Design

- 3 triggers
- kernel + user + last defense
- 3 + 2 syscalls



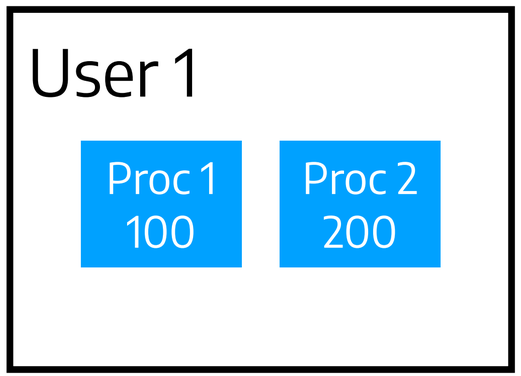
Fine-grained notifier in user-space

Fine-grained notifier in user-space

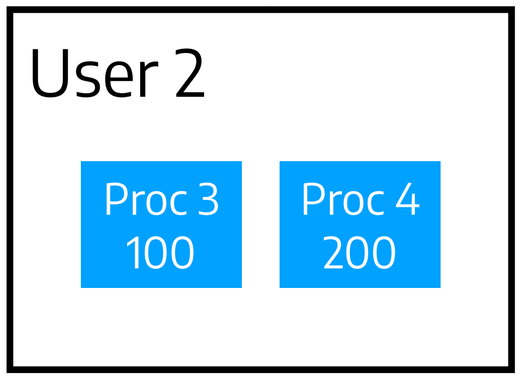
Fine-grained notifier

Basic Idea: only scan 1 process and 1 user

current usage: 300



current usage: 300



Fine-grained notifier

Basic Idea: only scan 1 process and 1 user

current usage: 300/300

User 1

Proc 1
100

Proc 2
200

current usage: 300/400

User 2

Proc 3
100

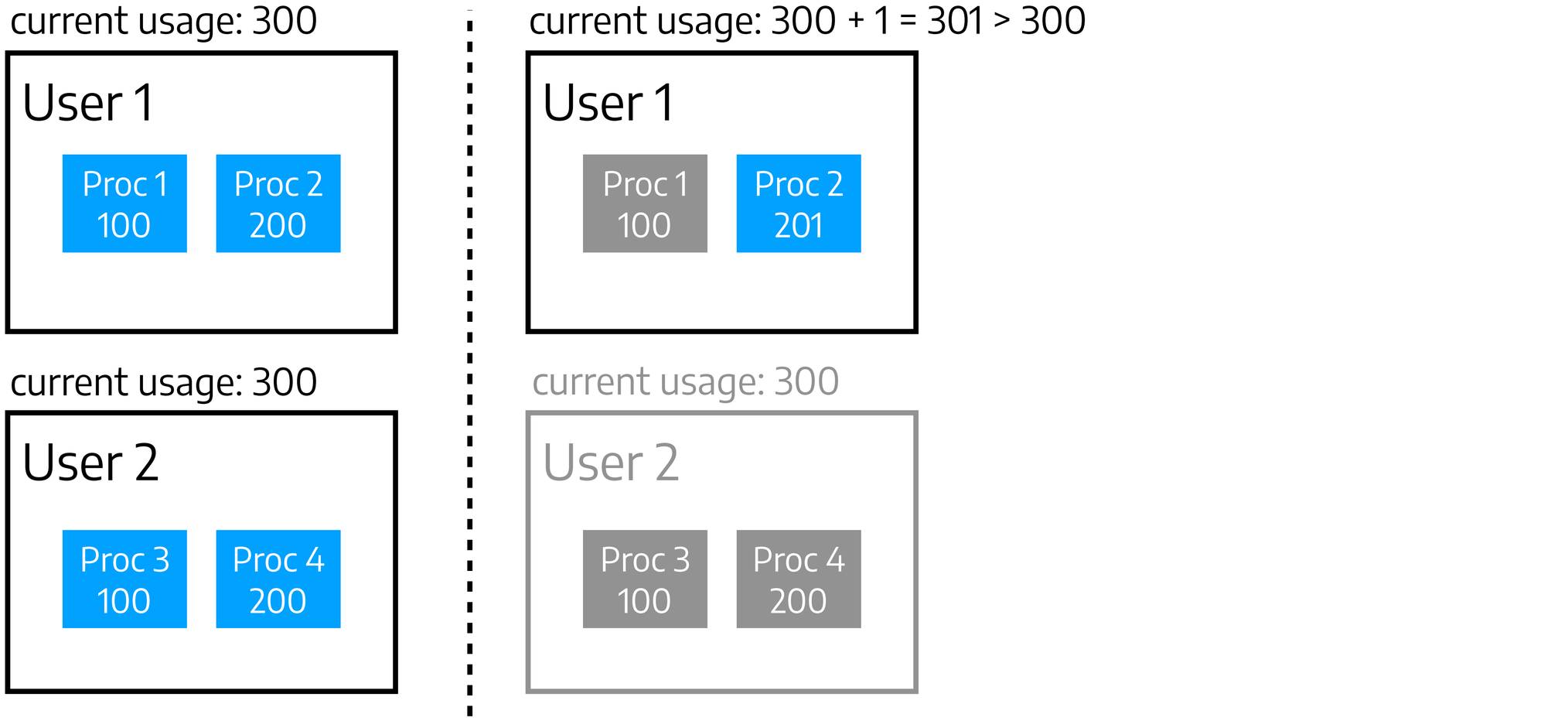
Proc 4
200

Proc 2 allocates 1 page

Listen for every memory allocation

Fine-grained notifier

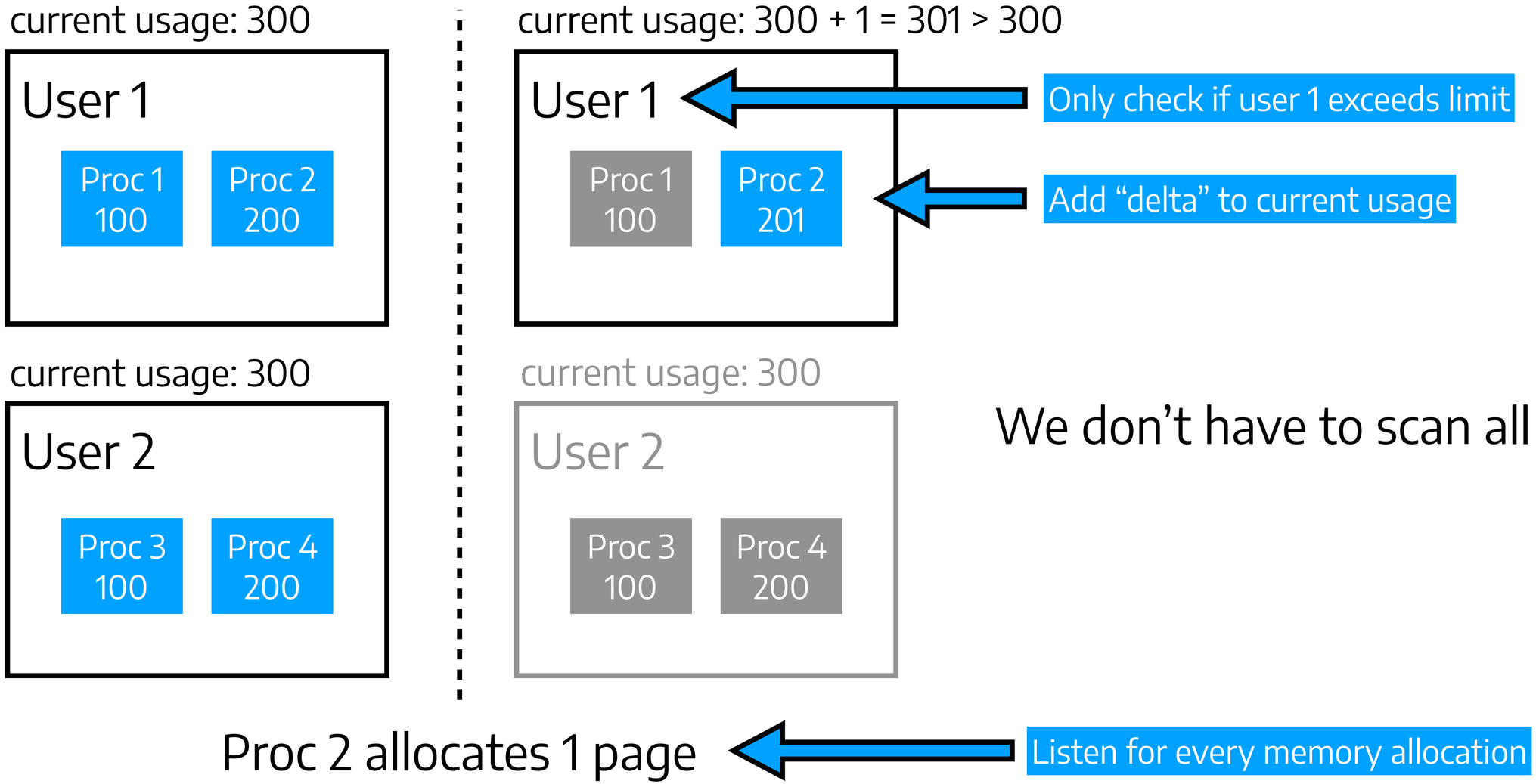
Basic Idea: only scan 1 process and 1 user



Proc 2 allocates 1 page ⟵ Listen for every memory allocation

Fine-grained notifier

Basic Idea: only scan 1 process and 1 user



Fine-grained notifier in user-space

Offload kernel task into user space

How to export kernel events to user space?

- Place tracepoints in kernel
  - Per-CPU ring-buffer inside
  - Very efficient
- Read from trace pipe

```
530 --- a/include/linux/mm.h
531 +++ b/include/linux/mm.h
532 @@ -1090,19 +1090,24 @@ static inline unsigned long get_mm_counter(struct mm_struct *mm, int member)
533         return (unsigned long)val;
534  }
535
536 +void mm_trace_rss_stat(int member, long count);
537 +
538  static inline void add_mm_counter(struct mm_struct *mm, int member, long value)
539  {
540 -       atomic_long_add(value, &mm->rss_stat.count[member]);
541 +       long count = atomic_long_add_return(value, &mm->rss_stat.count[member]);
542 +       mm_trace_rss_stat(member, count);
543  }
544
545  static inline void inc_mm_counter(struct mm_struct *mm, int member)
546  {
547 -       atomic_long_inc(&mm->rss_stat.count[member]);
548 +       long count = atomic_long_inc_return(&mm->rss_stat.count[member]);
549 +       mm_trace_rss_stat(member, count);
550  }
551
552  static inline void dec_mm_counter(struct mm_struct *mm, int member)
553  {
554 -       atomic_long_dec(&mm->rss_stat.count[member]);
555 +       long count = atomic_long_dec_return(&mm->rss_stat.count[member]);
556 +       mm_trace_rss_stat(member, count);
557  }
558
```

Offload kernel task into user space

How to export kernel events to user space?

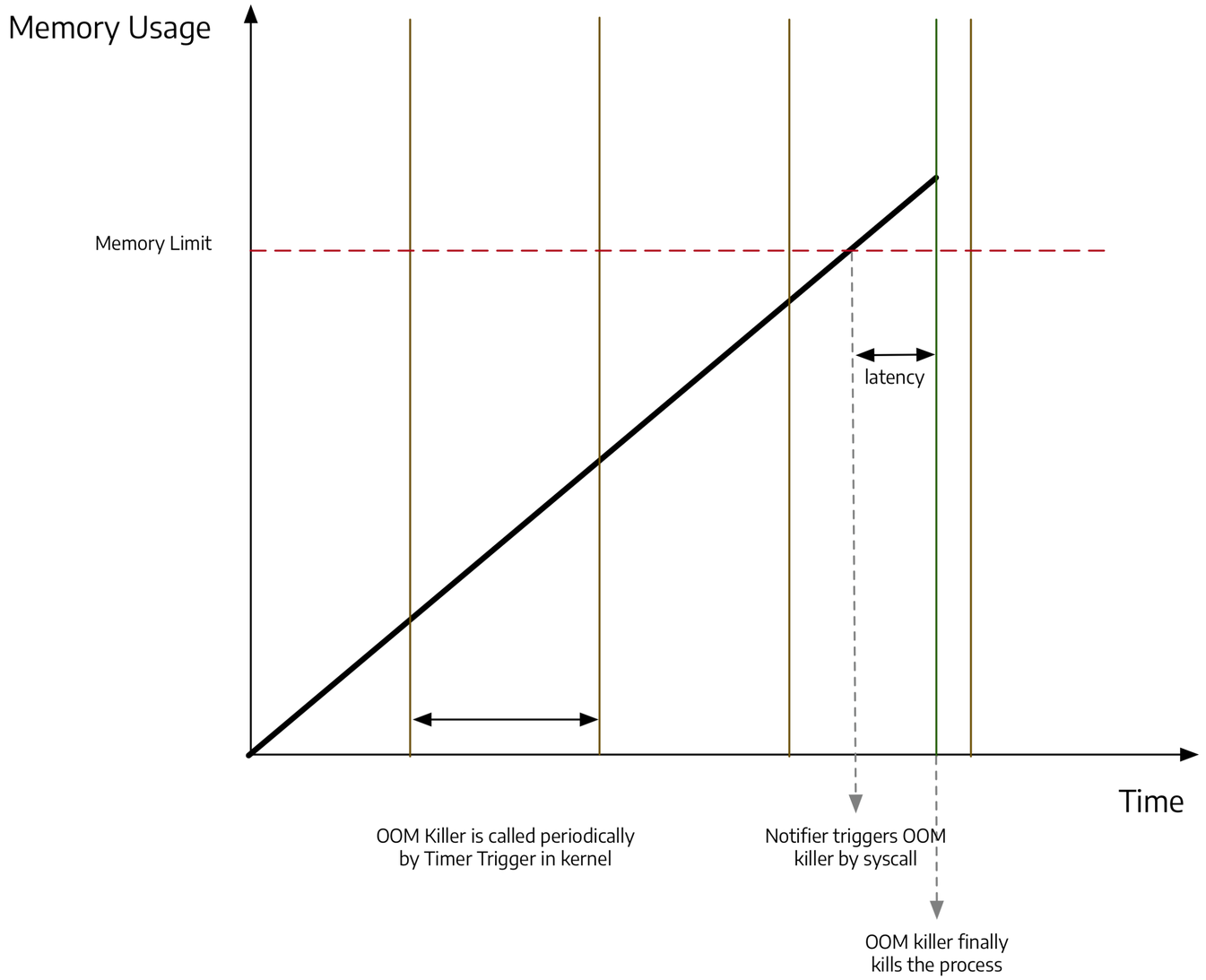
- Place tracepoints in kernel
- Read from trace pipe
  - PID & RSS
  - Real-time

```
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6681
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6682
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6683
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6684
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6685
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6686
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6687
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6688
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6689
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6690
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6691
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6692
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6693
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6694
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6695
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6696
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6697
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6698
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6699
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6700
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6701
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6702
ndroid.systemui-888   [000] ...2   597.590000: rss_stat: member=1 size=6703
ndroid.systemui-888   [000] ...2   597.610000: rss_stat: member=1 size=6639
ndroid.systemui-888   [000] ...2   597.610000: rss_stat: member=1 size=6442
          adbd-77     [000] ...2   597.640000: rss_stat: member=1 size=58
          adbd-172    [000] ...2   597.680000: rss_stat: member=1 size=53
          adbd-173    [000] ...2   597.750000: rss_stat: member=1 size=54
          adbd-173    [000] ...2   597.850000: rss_stat: member=1 size=55
          adbd-173    [000] ...2   597.950000: rss_stat: member=1 size=56
          adbd-173    [000] ...2   598.050000: rss_stat: member=1 size=57
```

Early Bootstrap to Reduce Latency

Reduce Latency in User-Space

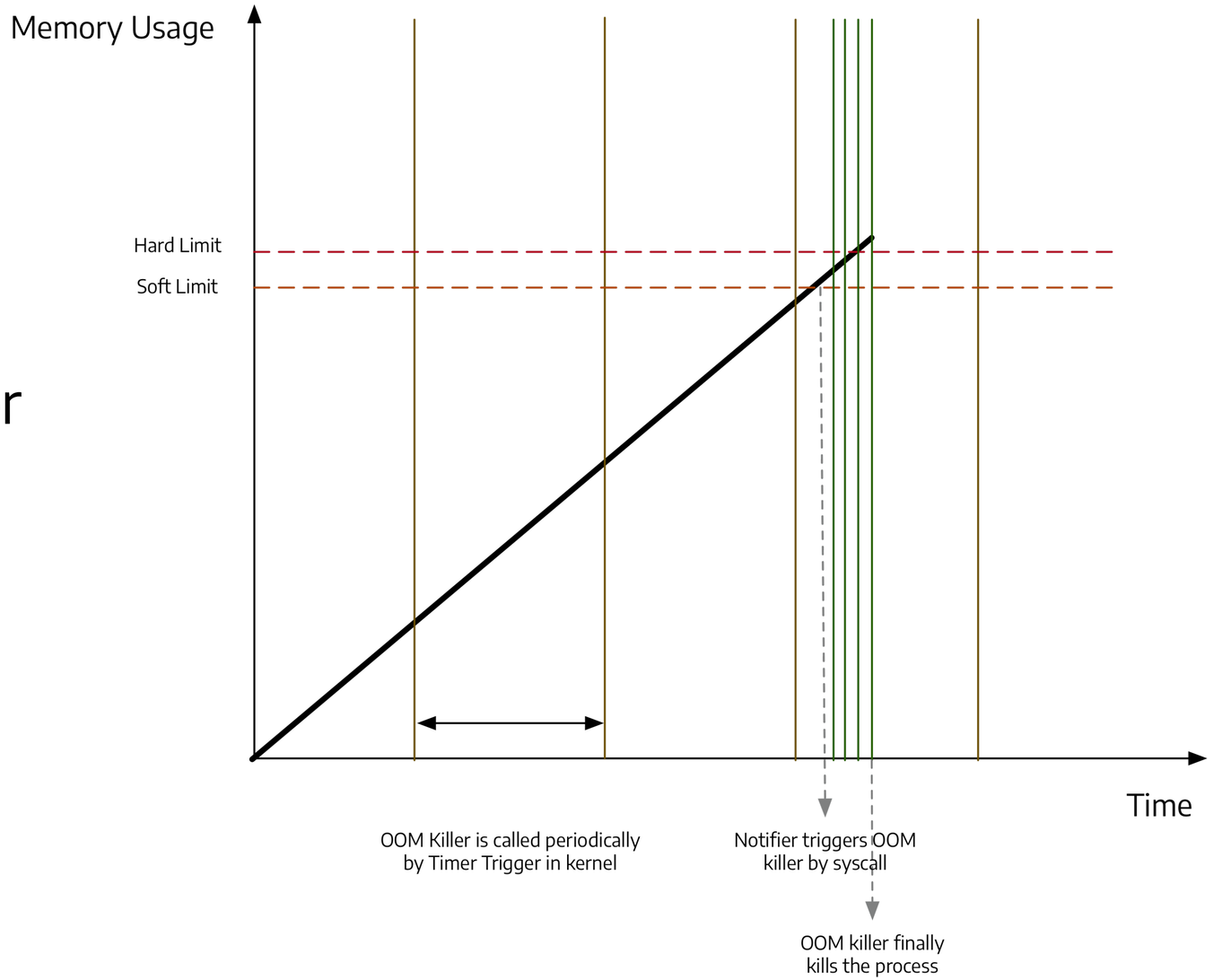
- Latency between trigger OOM killer and actually killing



Early Bootstrap to Reduce Latency

Reduce Latency in User-Space

- Latency between trigger OOM killer and actually killing
- When a user reaches soft limit (90% of hard limit), call OOM killer every 10 events



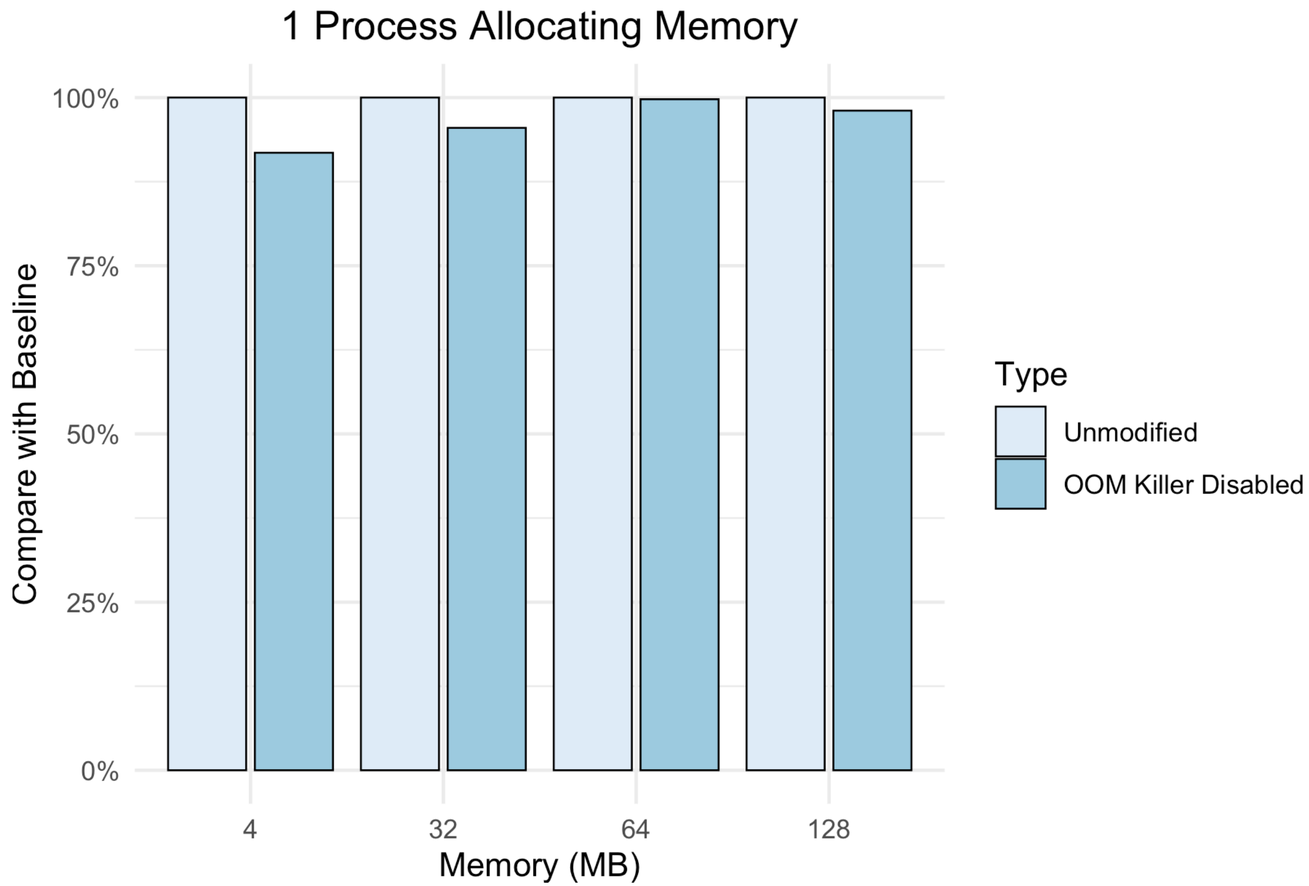
Fine-grained notifier in user-space

= low-overhead + low-latency + fault-tolerant

Benchmark

Memory Allocation Speed

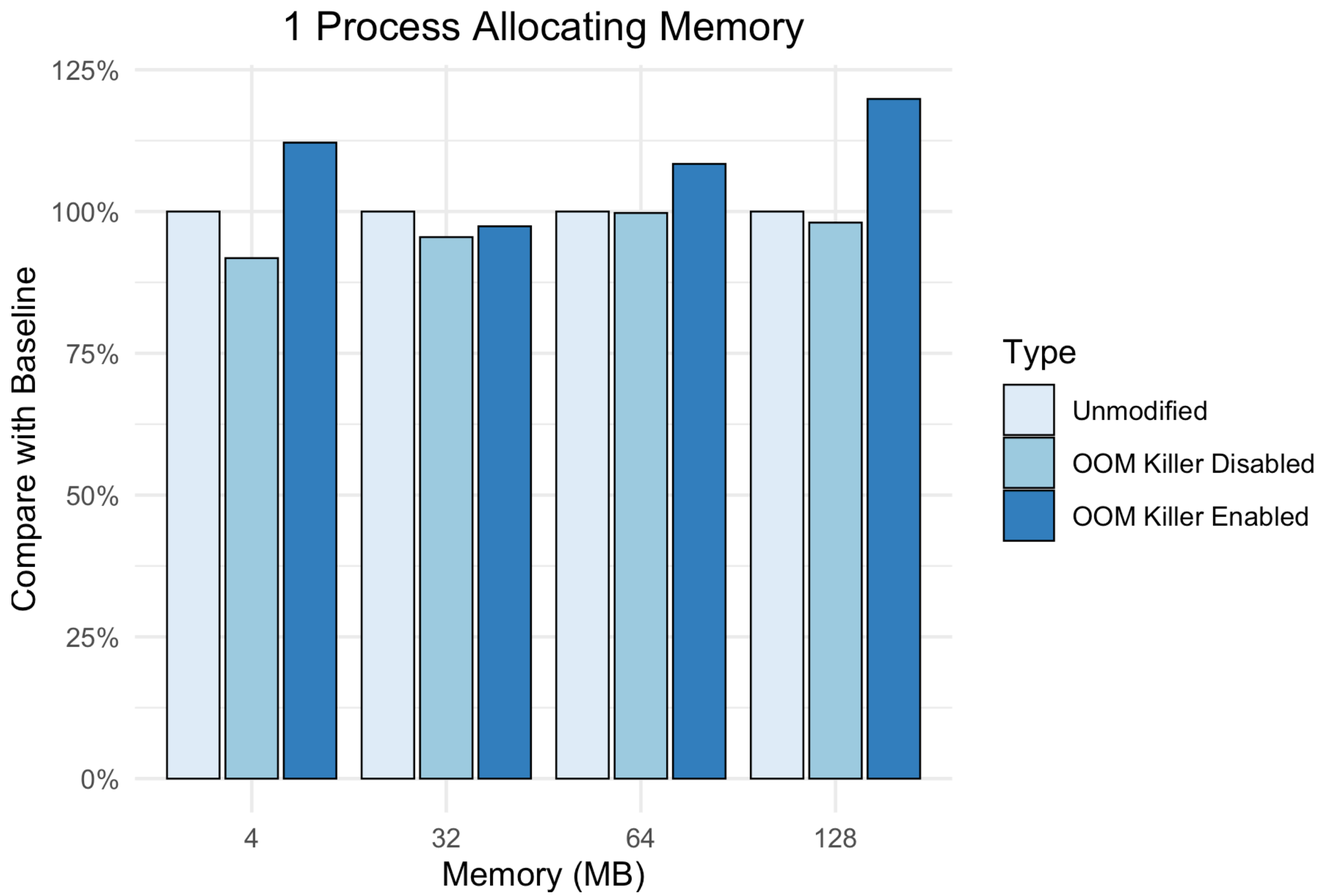
- No limit = no performance regression



Benchmark

Memory Allocation Speed

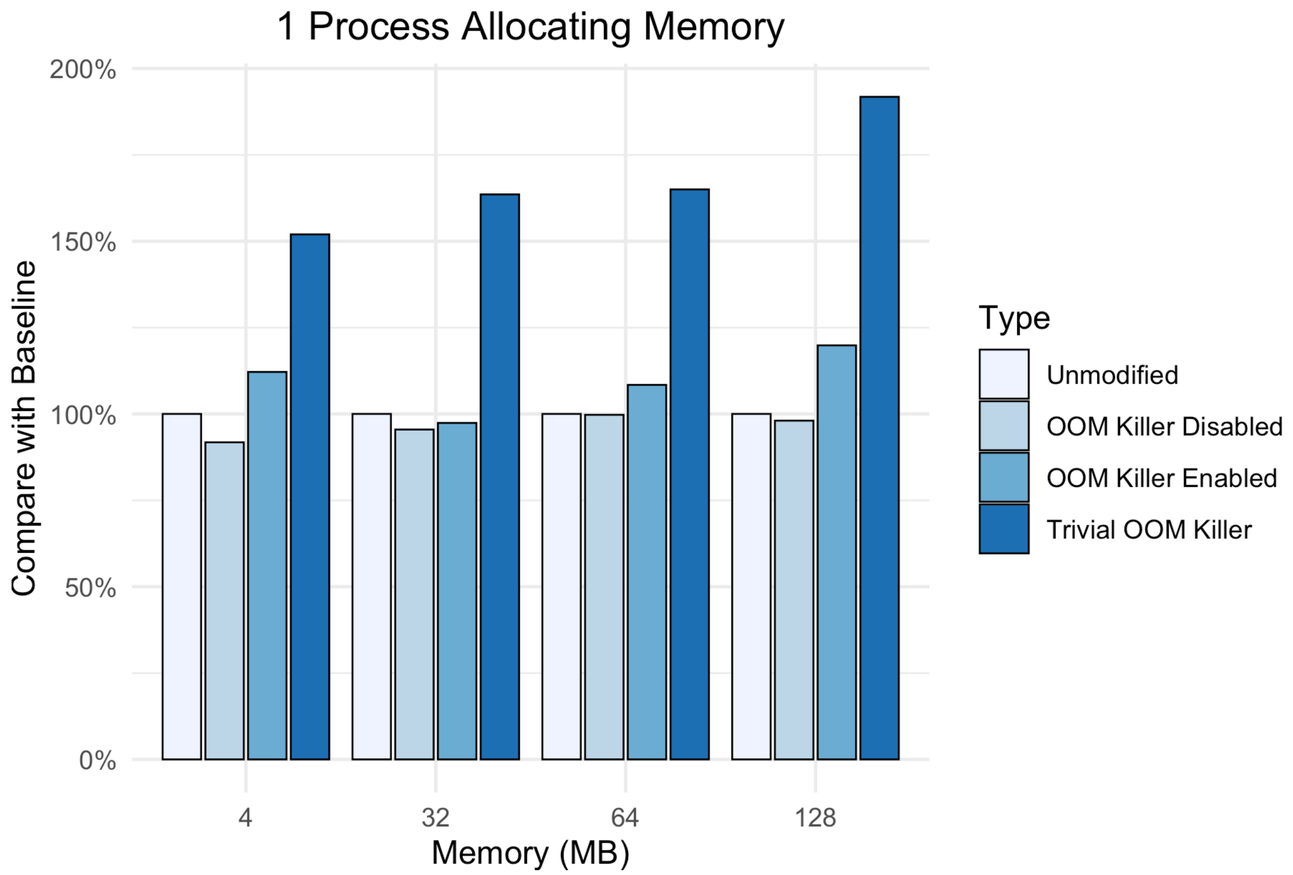
- No limit = no performance regression
- One Process = nearly no performance regression



Benchmark

Memory Allocation Speed

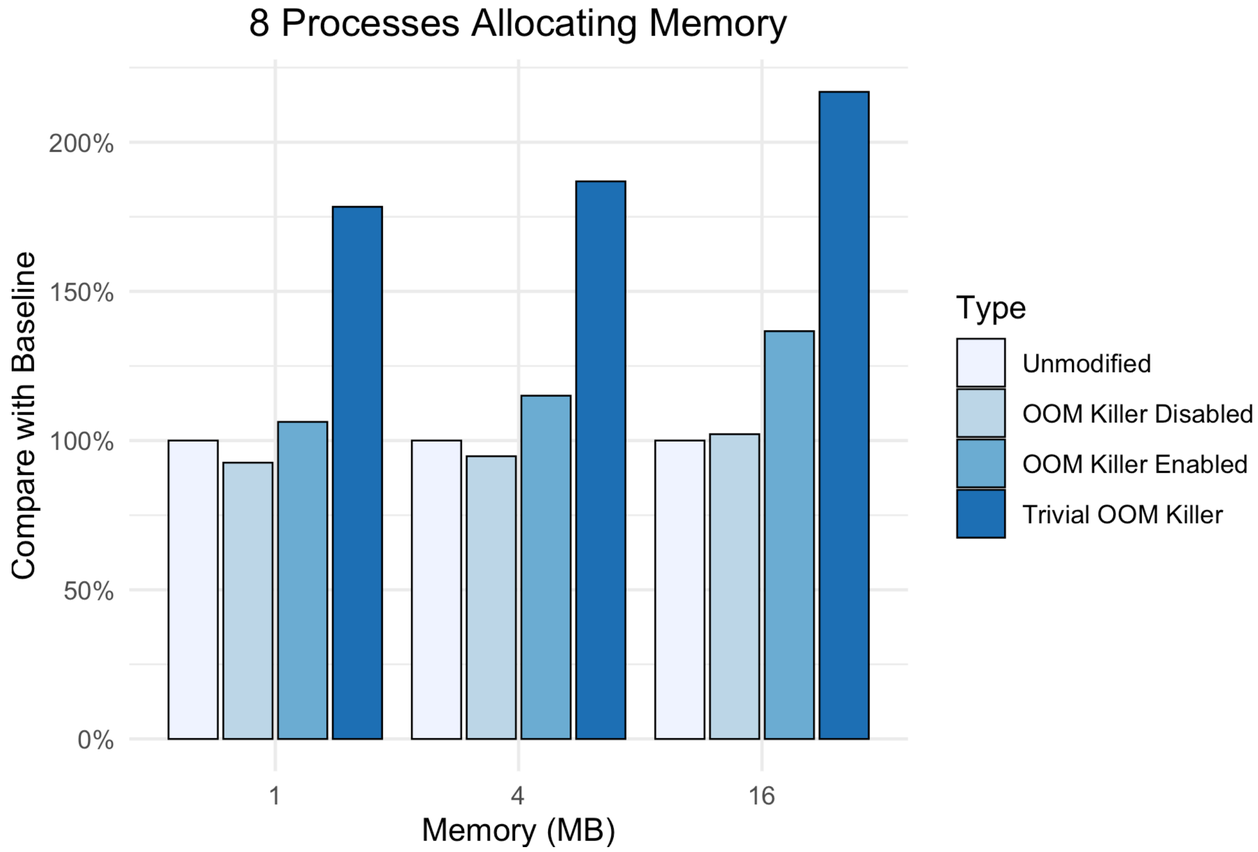
- No limit = no performance regression
- One Process = nearly no performance regression
- 2x speedup than trivial approach



Benchmark

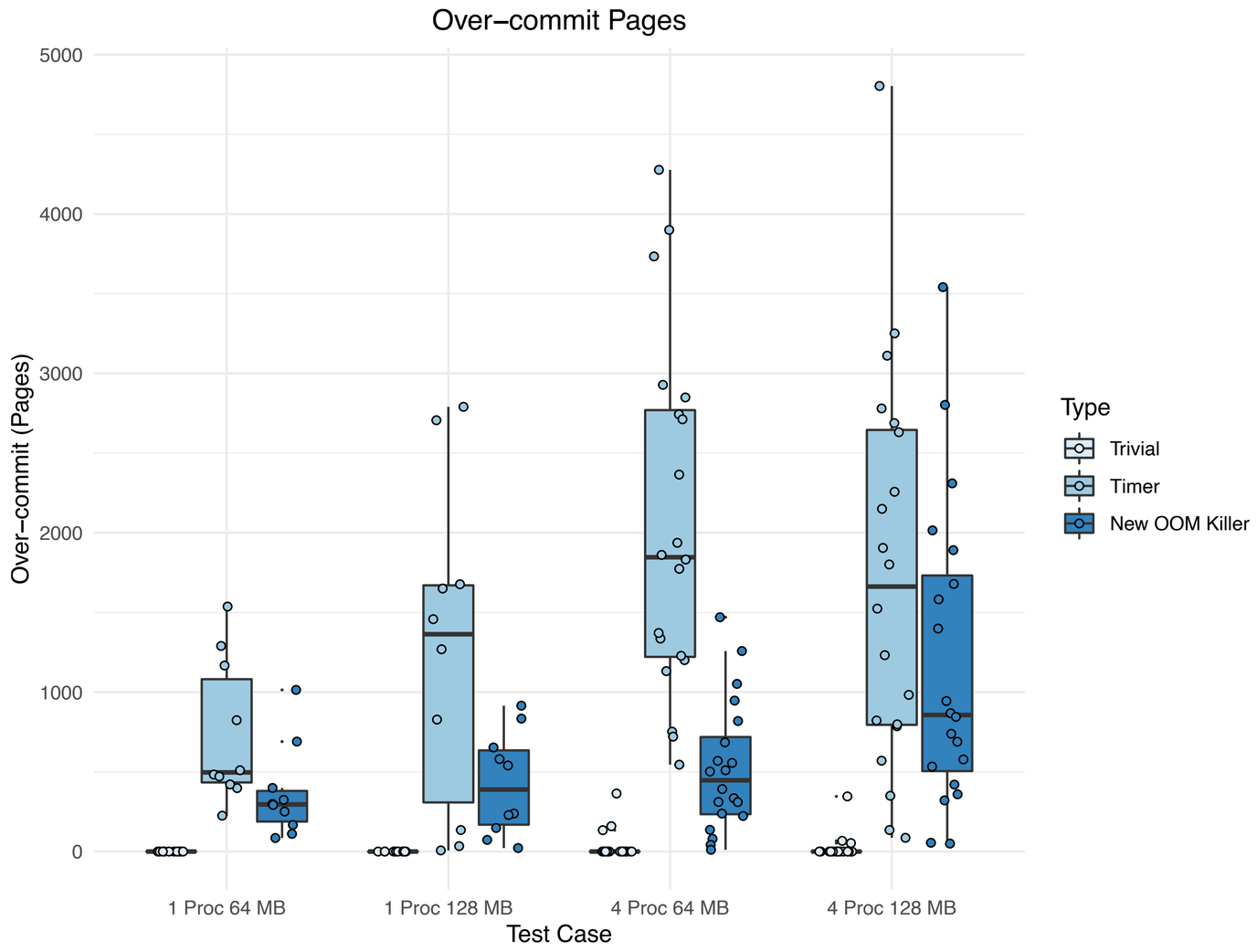
Memory Allocation Speed

- No limit = no performance regression
- One Process = nearly no performance regression
- 2x speedup than trivial approach
- Multiple Processes = 10%~20% performance regression



Benchmark
OOM Kill Latency

- Respond to events 3x more quickly than timer approach
- Lower variance, better stability



Memory Usage Awareness

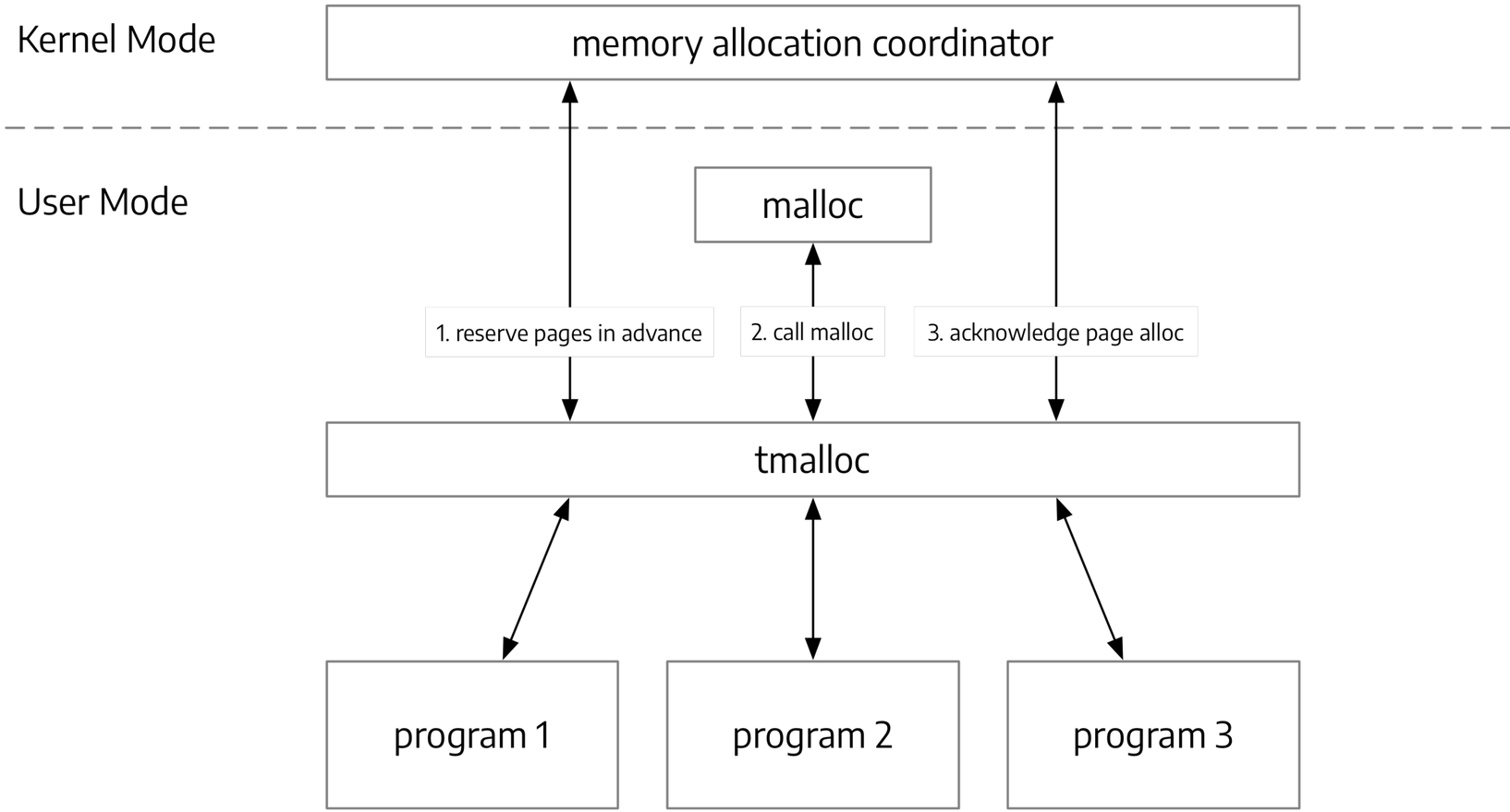
How can user process know they’ve used up their memory?

Memory Usage Awareness

- Linux’s approach: return NULL on malloc if there’s no space
- Our approach: return NULL on “tmalloc”

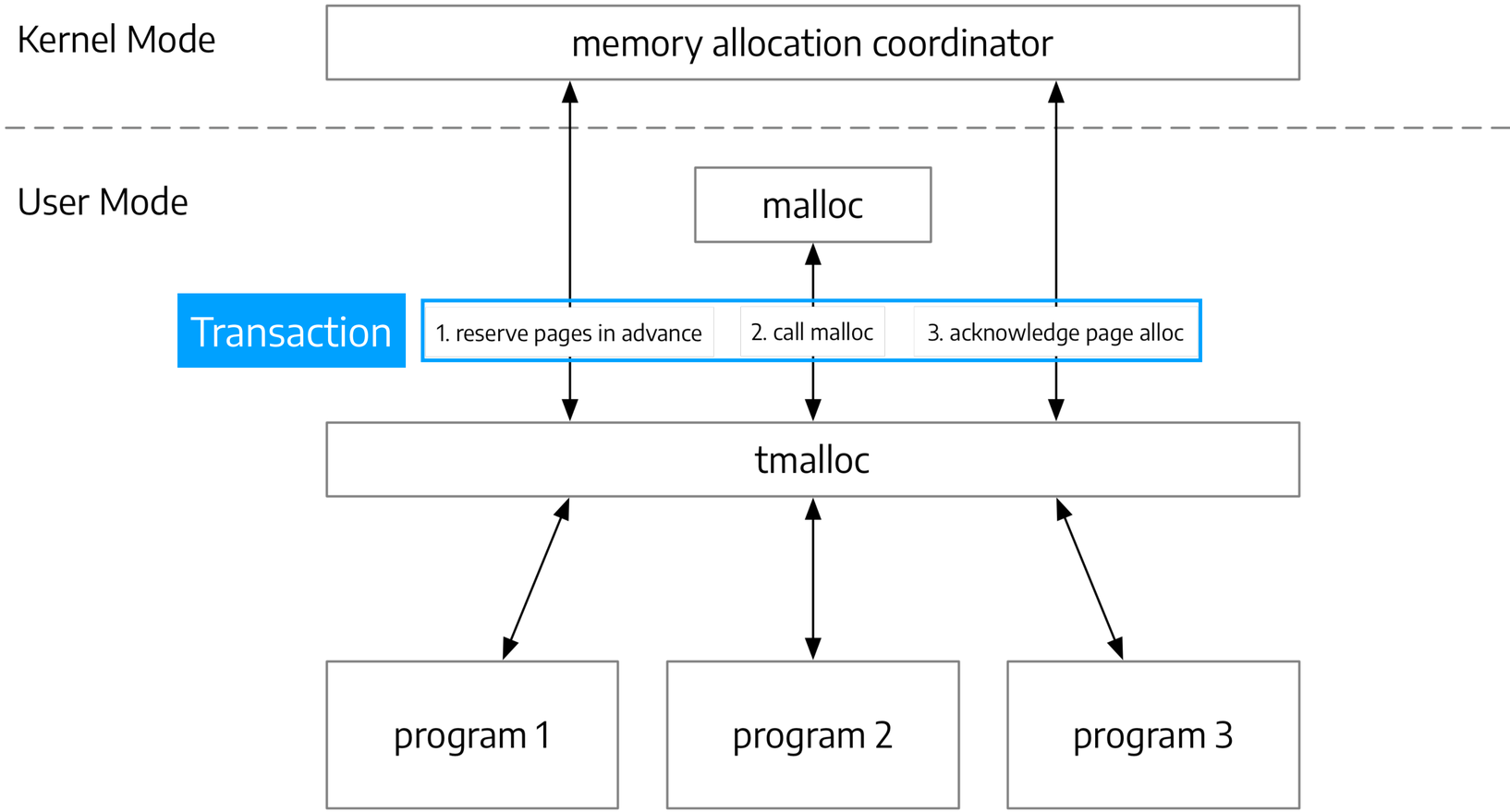
Transactional Malloc

- Introduce a “memory allocation coordinator”
- User process calls “tmalloc”



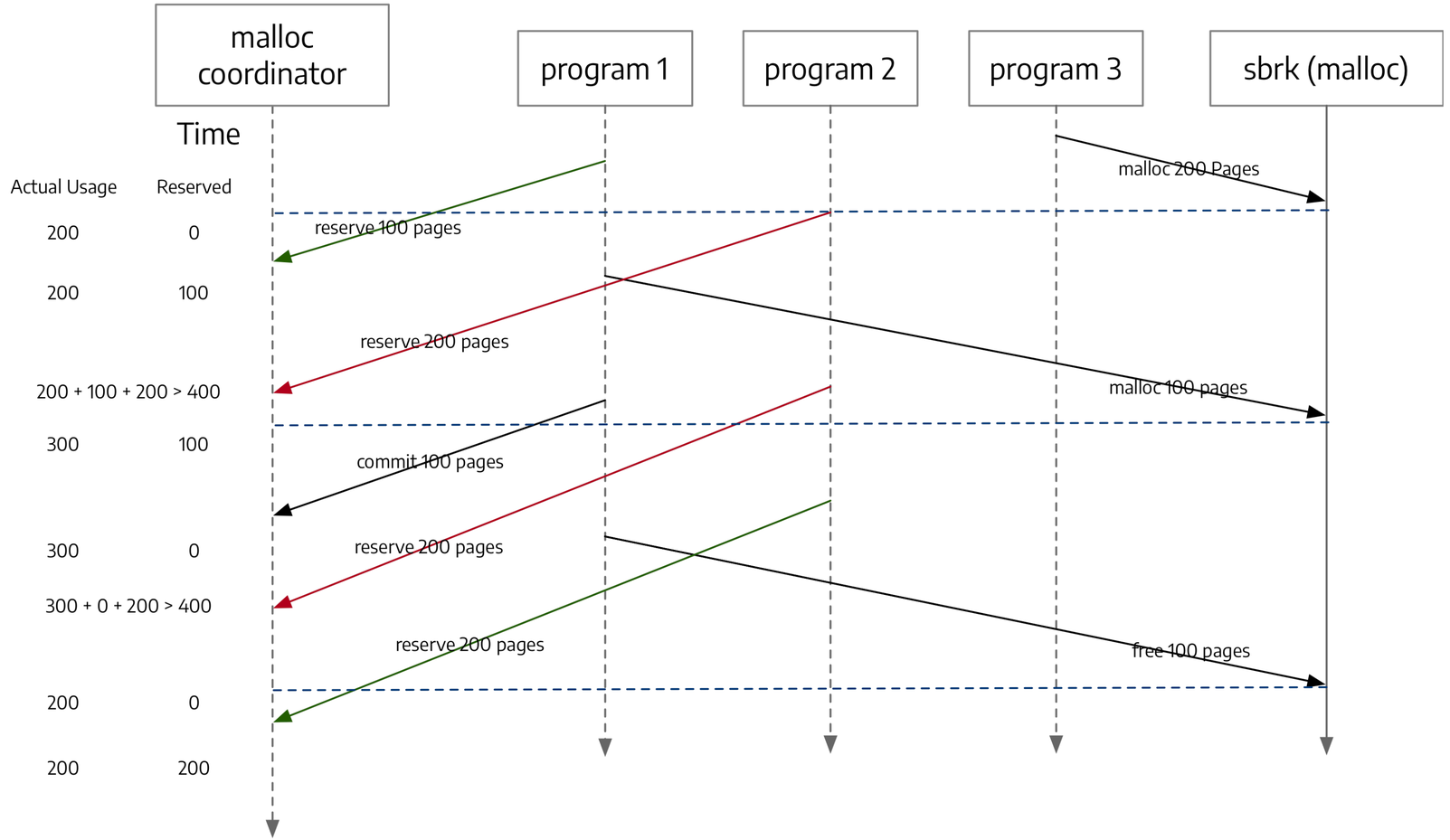
Transactional Malloc

- Introduce a “memory allocation coordinator”
- User process calls “tmalloc”



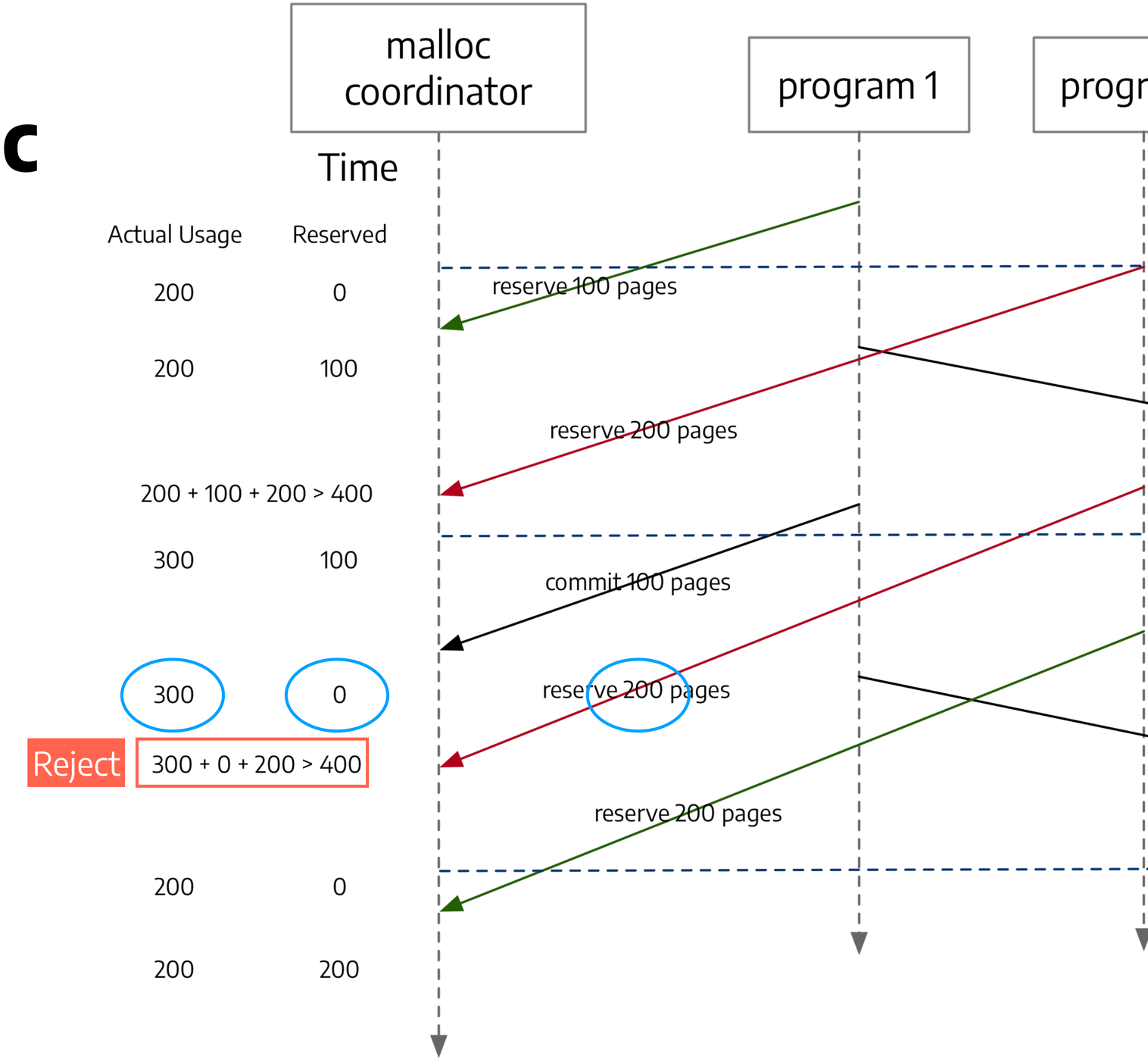
Transactional Malloc

- Similar to “Deadlock Avoidance”
- Memory as a kind of resource
- Grant request when we are sure that no process will be killed



Transactional Malloc

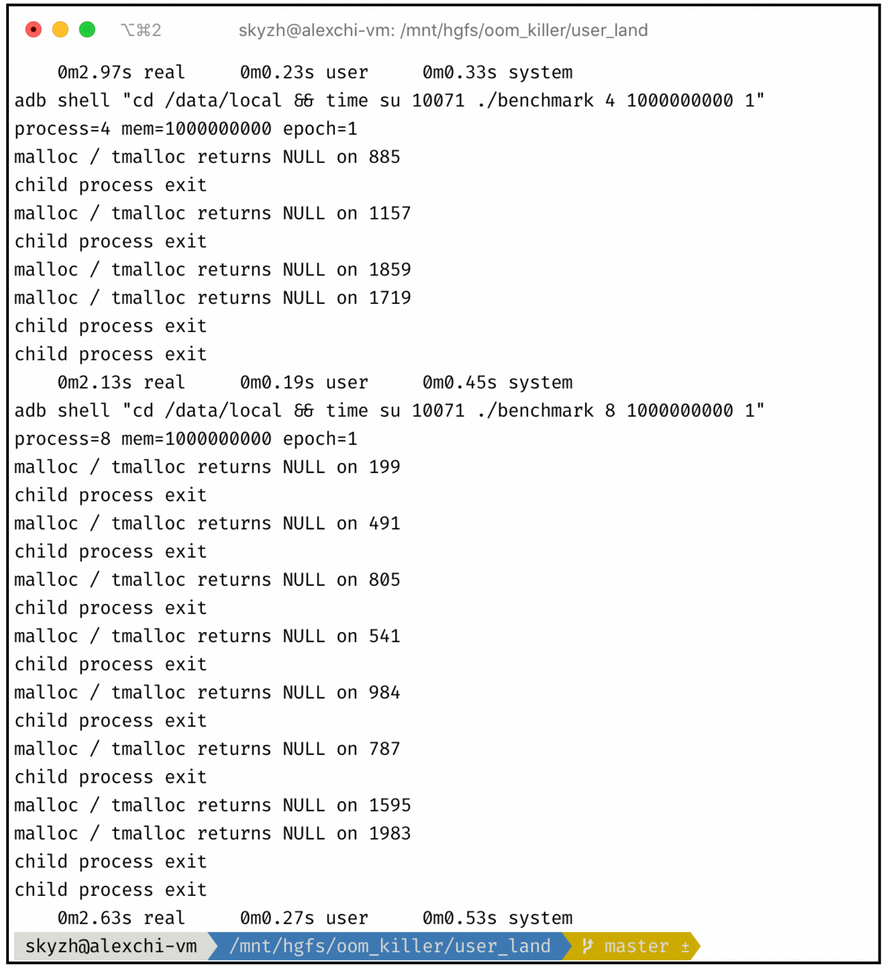
- Similar to “Deadlock Avoidance”
- Memory as a kind of resource
- Grant request when we are sure that no process will be killed



Transactional Malloc

Never exceed limit

```
[NEW_OOM] limit set: uid=10071, mm_max=16777216B or 4096 pages
[NEW_OOM] request not granted: uid=10071 pid=209 4086+11>4096
[NEW_OOM] request not granted: uid=10071 pid=219 4075+22>4096
[NEW_OOM] request not granted: uid=10071 pid=218 4086+11>4096
[NEW_OOM] request not granted: uid=10071 pid=221 4075+22>4096
[NEW_OOM] request not granted: uid=10071 pid=220 4075+22>4096
warning: `main' uses 32-bit capabilities (legacy support in use)
[NEW_OOM] request not granted: uid=10071 pid=230 4053+44>4096
[NEW_OOM] request not granted: uid=10071 pid=235 4042+55>4096
[NEW_OOM] request not granted: uid=10071 pid=237 4053+44>4096
[NEW_OOM] request not granted: uid=10071 pid=233 4065+33>4096
[NEW_OOM] request not granted: uid=10071 pid=232 4053+44>4096
[NEW_OOM] request not granted: uid=10071 pid=234 4075+22>4096
[NEW_OOM] request not granted: uid=10071 pid=236 4089+11>4096
```



Acknowledgement

- Prof. Wu & TA
- Linux Community & Linux Kernel Documentation
- Zefu Wang, Yimin Zhao and Ziqi Zhao

Thanks!

Q&A