oomd

Daniel Xu
Production Engineer

Out of memory daemon

- User-space
- Dependency free
- Deterministic, faster, and more flexible than kernel OOM killer
- Open source (GPL2)
  - https://github.com/facebookincubator/oomd
  - https://facebookmicrosites.github.io/oomd/

Agenda

- Motivation
- Mechanism
- Results
- Questions

Motivation

Kernel OOM killer

- Memory overcommit
- Configuration not very intuitive (what's with all the numbers?)
  - */proc/[pid]/oom_adj*
  - */proc/[pid]/oom_score*
  - */proc/[pid]/oom_score_adj*
  - */proc/sys/vm/oom_kill_allocating_task*
  - */proc/sys/vm/panic_on_oom*
- Slow to act; often it's already too late by the time the kernel reacts
  - Tries to protect kernel health; user-space could be livelocked but the kernel could still be happily churning pages in and out

Kernel OOM killer (cont)

- No context on logical composition of system
  - What should be killed together, what shouldn't be, etc.
- No way to customize kill action (modulo OOM eventfd; still slow though)
  - For some processes, a SIGTERM/SIGKILL is fine. Other might want a song and dance
    - Eg. a persistent process that manages containers
- Non-determinstic (Or at least really hard to get deterministic)

OOM problems @ FB

- Continuous build and test platform
- Container & service platform (Tupperware)
- White box (commodity) top-of-rack switches (FBOSS)
- Other multi-tenant platforms
- *vm.panic_on_oom*

Fbtax2

- See "Resource Control @ FB" talk
  - Work-conserving full-OS resource isolation across applications
  - Uses cgroup2, PSI, io.latency, btrfs, and oomd to isolate workloads on the same machine
    - https://facebookmicrosites.github.io/cgroup2
    - http://git.cmpxchg.org/cgit.cgi/linux-psi.git/
    - https://facebookmicrosites.github.io/btrfs

Mechanism

How does it work?

- PSI
  - New kernel feature (by FB's Johannes Weiner) that quantifies lost wall clock time due to resource shortages
- Core of oomd is the plugin system
  - All detection and action behavior is customizable
  - Default OomDetector and OomKiller plugins are sensible and work well across a variety of workloads
- Not just memory, also monitors IO pressure and swap depletion
  - There to remediate when kernel resource isolation isn't enough

```
{
    "target": "system.slice",
    "kill_list": [
        {"chef.service": { "max_usage": "1000" } },
        {"sshd.service": { "max_usage": "inf" } },
    ],
    "oomdetector": "default",
    "oomkiller": "default"
}
```

Original oomd config

oomd2

- Oomd as it currently exists works well
- More users, more use cases, new oomd design
- Still iterating and playing around with details
  - In general, we're onto something very useful here

```
1  {
2      "rulesets": [
3          <FIRE IF WORKLOAD.SLICE SLOWS BY > 5%>,
4          <FIRE IF SYSTEM.SLICE SLOWS BY > 40%>
5      ],
6      "actions": [
7          <KILL A MEMORY HOG IN SYSTEM.SLICE>
8      ]
9  }
10
```

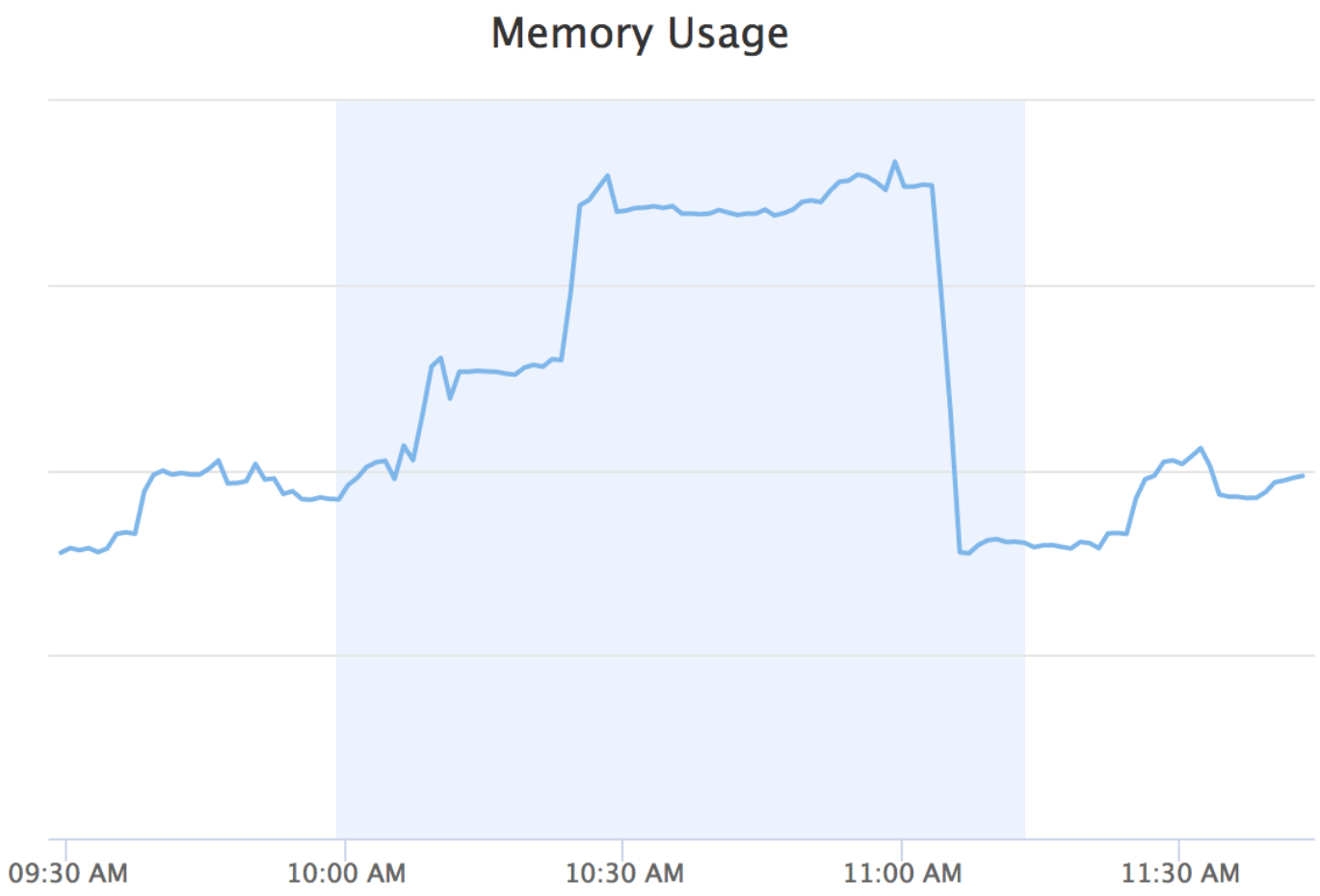
pseudocode

oomd2 config (pseudo code)

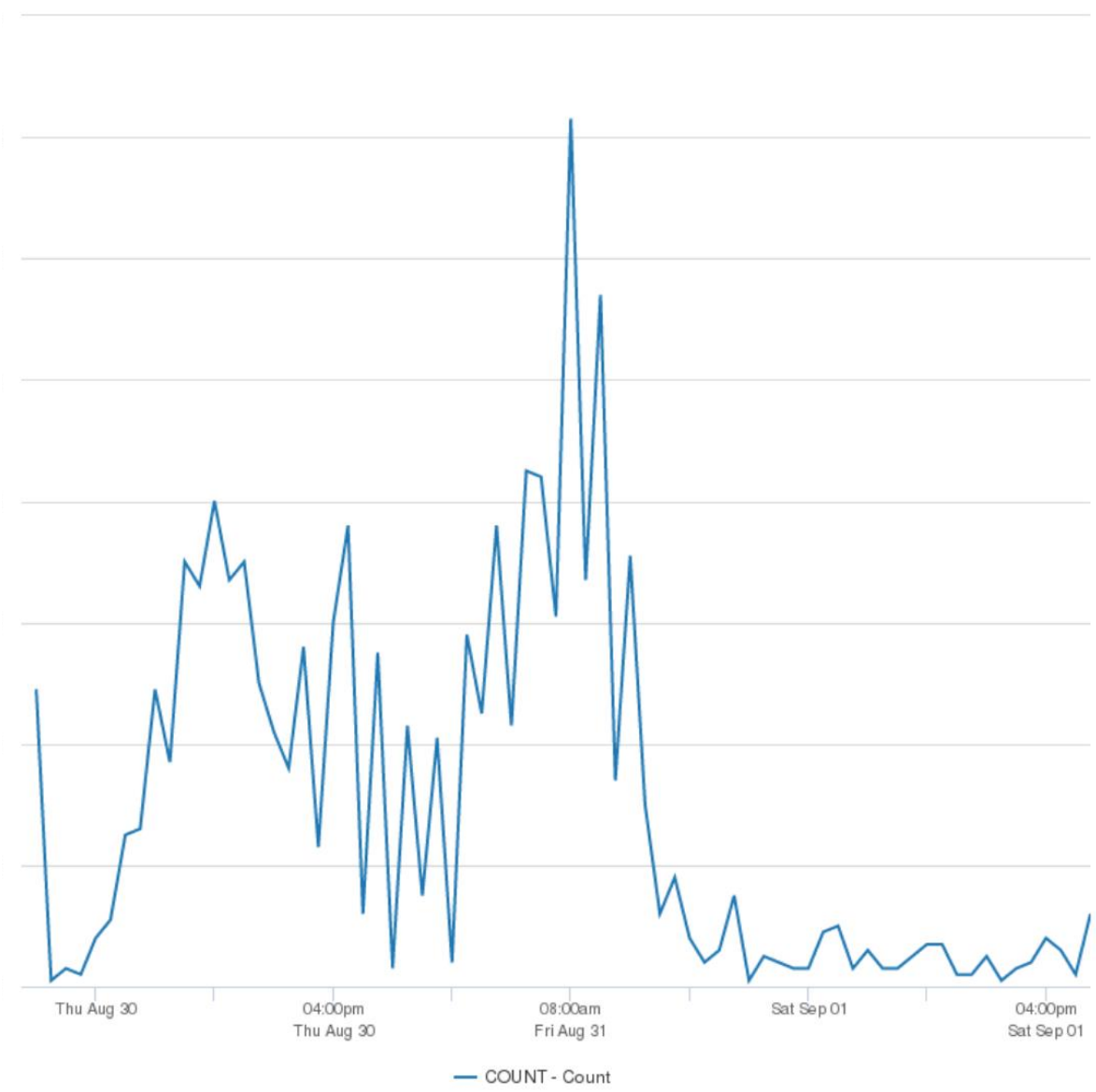
```
{
    "rulesets": [
        [
            "name": "my first ruleset",
            "detectors": [
                [
                    "group1",
                    [
                        "pressure_rising_beyond",
                        "cgroup=workload.slice",
                        "resource=memory",
                        "threshold=5"
                    ],
                    [
                        "pressure_rising_beyond",
                        "cgroup=system.slice",
                        "resource=memory",
                        "threshold=40"
                    ]
                ]
            ],
            "actions": [
                [
                    "kill_by_memory_size_or_growth",
                    "cgroup=system.slice"
                ]
            ]
        ]
    ]
}
```

oomd2 config (real)

Results



An oomd kill on a build host (kills memory hogging job)



panic_on_oom rate before and after oomd rollout

Questions