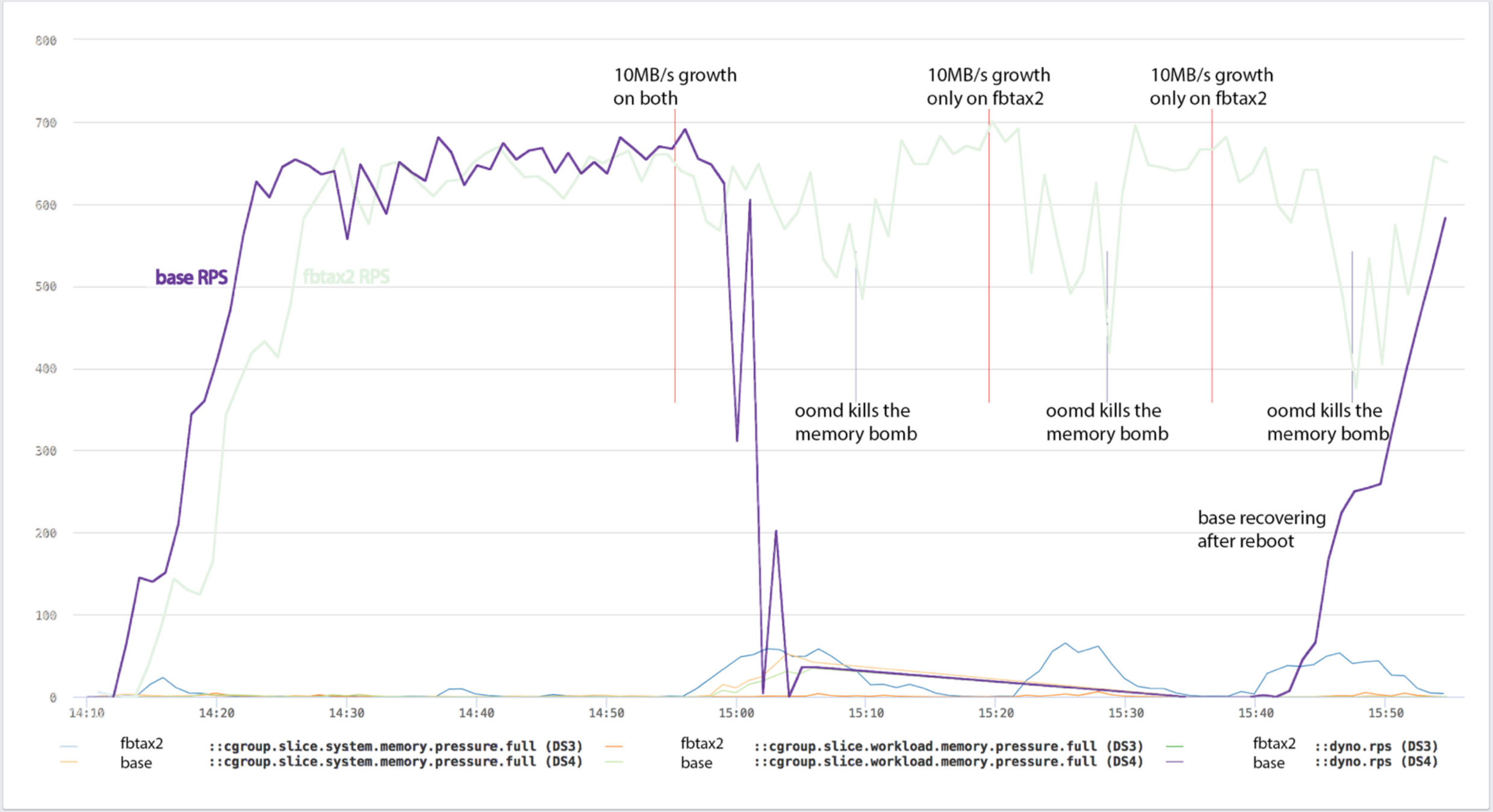
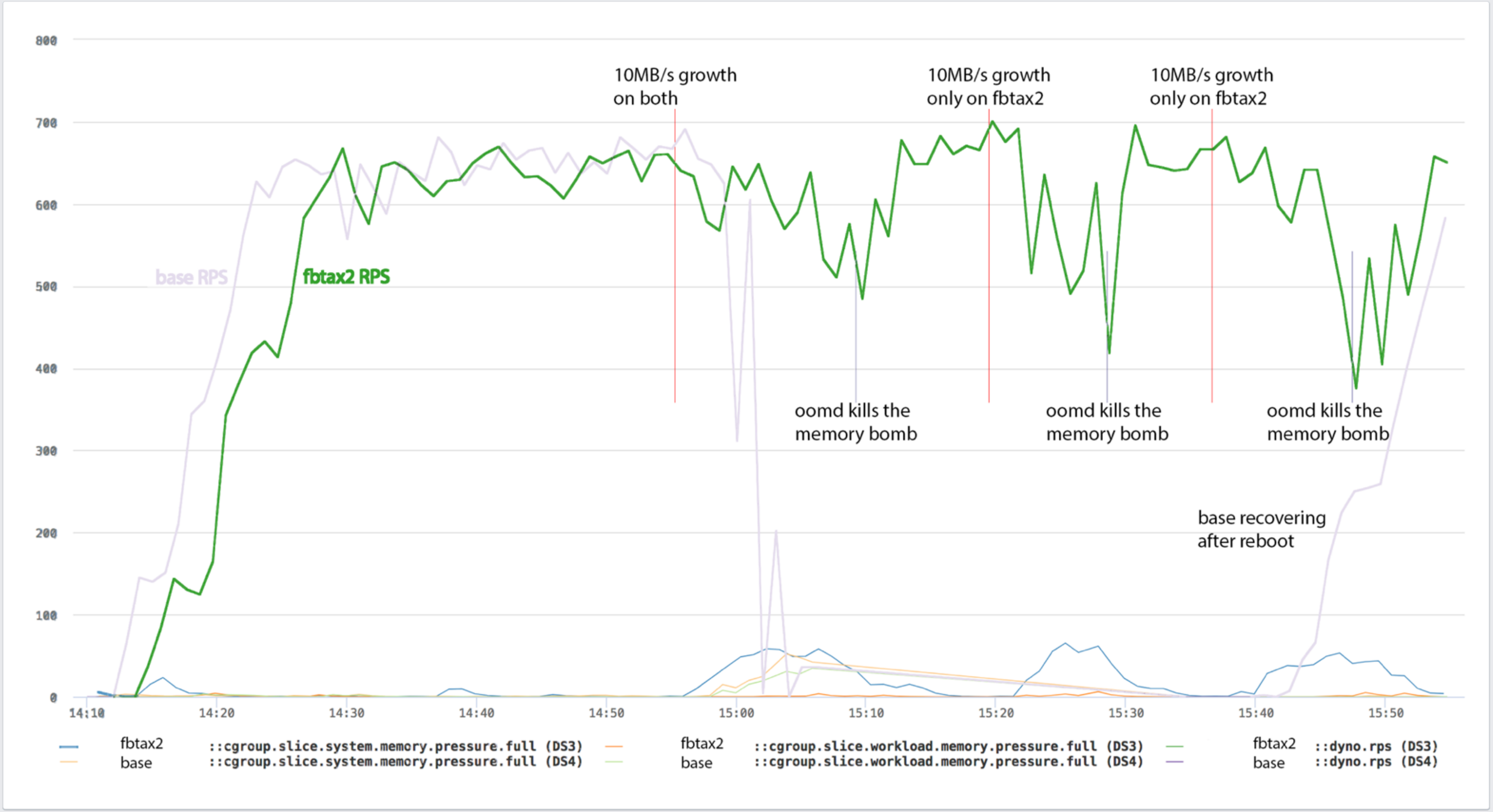
facebook

Resource Control @ FB

Tejun Heo

Software Engineer


800
700
600
500
400
300
200
100
0
10MB/s growth
on both
10MB/s growth
only on fbtax2
10MB/s growth
only on fbtax2
base RPS
fbtax2 RPS
oomd kills the
memory bomb
oomd kills the
memory bomb
oomd kills the
memory bomb
base recovering
after reboot
14:10
14:20
14:30
14:40
14:50
15:00
15:10
15:20
15:30
15:40
15:50
fbtax2 ::cgroup.slice.system.memory.pressure.full (DS3)
base ::cgroup.slice.system.memory.pressure.full (DS4)
fbtax2 ::cgroup.slice.workload.memory.pressure.full (DS3)
base ::cgroup.slice.workload.memory.pressure.full (DS4)
fbtax2 ::dyno.rps (DS3)
base ::dyno.rps (DS4)


800
700
600
500
400
300
200
100
0
10MB/s growth
on both
10MB/s growth
only on fbtax2
10MB/s growth
only on fbtax2
base RPS
fbtax2 RPS
oomd kills the
memory bomb
oomd kills the
memory bomb
oomd kills the
memory bomb
base recovering
after reboot
14:10
14:20
14:30
14:40
14:50
15:00
15:10
15:20
15:30
15:40
15:50
fbtax2 ::cgroup.slice.system.memory.pressure.full (DS3)
base ::cgroup.slice.system.memory.pressure.full (DS4)
fbtax2 ::cgroup.slice.workload.memory.pressure.full (DS3)
base ::cgroup.slice.workload.memory.pressure.full (DS4)
fbtax2 ::dyno.rps (DS3)
base ::dyno.rps (DS4)


800
700
600
500
400
300
200
100
0
10MB/s growth
on both
10MB/s growth
only on fbtax2
10MB/s growth
only on fbtax2
base RPS
fbtax2 RPS
oomd kills the
memory bomb
oomd kills the
memory bomb
oomd kills the
memory bomb
base recovering
after reboot
14:10
14:20
14:30
14:40
14:50
15:00
15:10
15:20
15:30
15:40
15:50
fbtax2 ::cgroup.slice.system.memory.pressure.full (DS3)
base ::cgroup.slice.system.memory.pressure.full (DS4)
fbtax2 ::cgroup.slice.workload.memory.pressure.full (DS3)
base ::cgroup.slice.workload.memory.pressure.full (DS4)
fbtax2 ::dyno.rps (DS3)
base ::dyno.rps (DS4)

The Goal

Work-conserving full-OS resource isolation

What does that mean?

- Work-conserving:
  - Don’t keep machine idle if there’s work to do
- Full-OS:
  - Transparent
  - Keep doing what you’ve been doing and overlay isolation
  - No need for direct IO, hard-allocate mem, separate FS

fbtax2

The Challenges

Memory Control

- *memory.high* and *.max* aren’t work-conserving
- Adding restrictions to already over-subscribed systems

  -> more brittle systems
- Kernel OOM killer doesn’t protect workload health

The Challenges

IO Control

- No good IO controller to use
- Accounting of FS metadata and swap IOs

The Challenges

Priority Inversions

- Filesystem operations (e.g. ext4)
- FS metadata, swap IO spikes lead to priority inversions
- *mmap_sem* and readahead
- Misc – squashfs, fuse…

The Solutions

memory.low and memory.min

Lift up, don’t push down

- Work-conserving best-effort protection
- More forgiving, allowing for ball-park configurations
- Proportional pressure (being worked on)

PSI – resource pressure metric

Who's slowing us down?

If I had more of this resource, I might have been able to run this percentage faster

- On memory, IO and CPU
- System-wide and per-cgroup
- Reliable and intuitive understanding of workload health
- Used for resource allocation, load-shedding, *oomd*

oomd

The gentler and more perceptive grim reaper

- Watch workload health with PSI, remediate contentions
- Workload QoS and context-aware decisions and actions
- Helps kernel when resource isolation breaks down

io.latency

Completion latency based IO control

- Best-effort avg (or p90) completion latency guarantee
- More work-conserving
- Can be used both on hard disks and SSDs
- Supports do-first-pay-later for metadata and swap IOs
- Works on blk-mq

The Hunt for Priority Inversions

Kernel gotta be able to handle a part of system being really slow

- Switch to btrfs and fix priority inversions
- *mmap_sem*: readahead aborts, early breakout
- Shared IOs: do-first-pay-later, dirtier/allocator throttling
- Other misc config change and fixes

The Setup

fbtax2 - btrfs

No FS priority inversions, easier management

- Multi-100k machines running on btrfs (HDD and SSD)
- All priority inversions fixed, all metadata annotated

fbtax2 - swap

If there’s no swap, all anon memory is memlocked

- Better use of memory
- Allows memory pressure to build up gracefully
- Enabled everywhere except for the main workload

fbtax2 - cgroup

- *hostcritical.slice* (mem.min=352M, io.latency=50ms)

  oomd, sshd, systemd-journald, rsyslog

- *workload.slice* (mem.low=17G, io.latency=50ms)

  workload-wdb.slice (mem.low=2.5G)

  workload-tw.slice (mem.low=max)

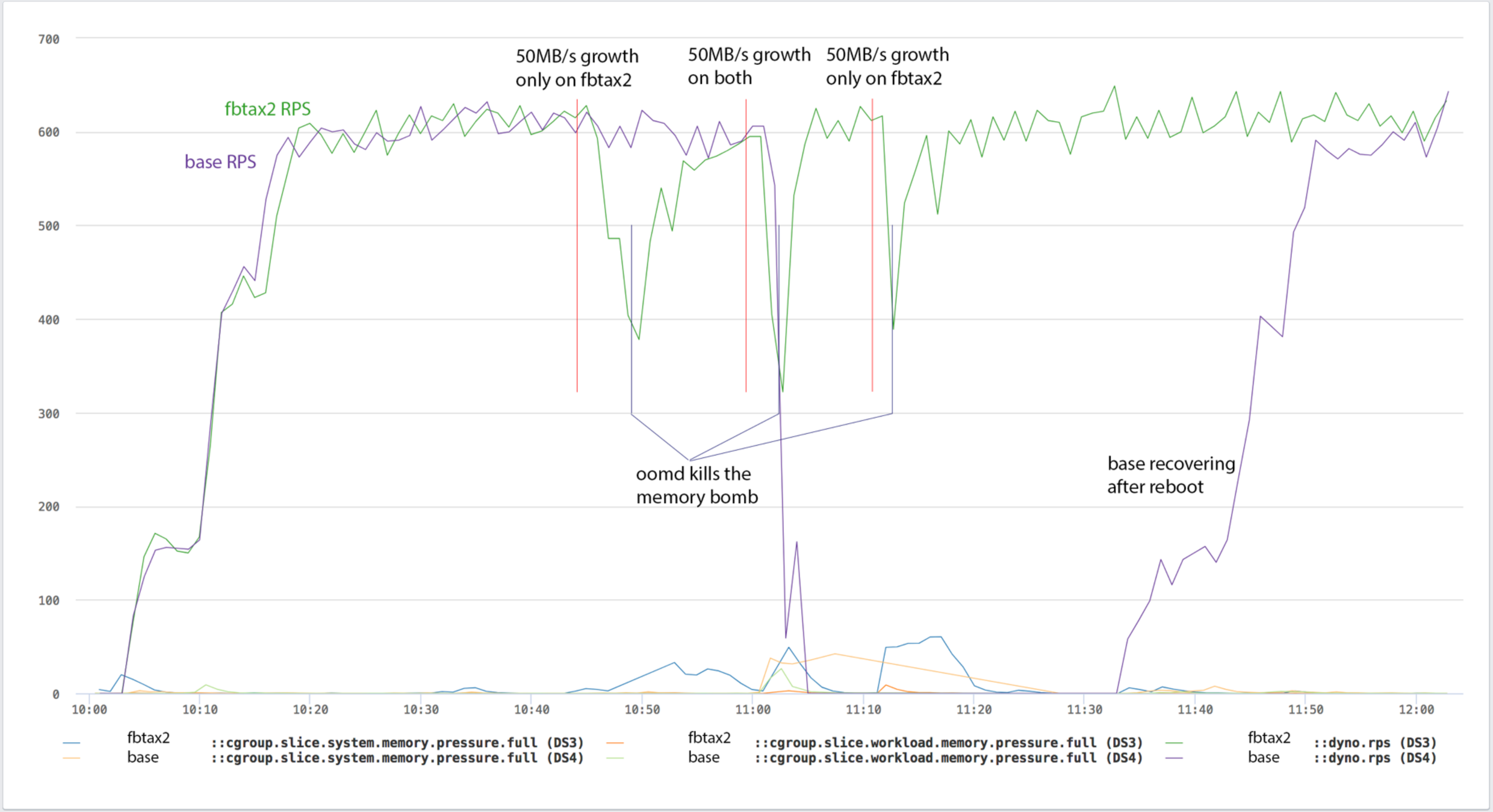
- *system.slice* (io.latency=75ms)

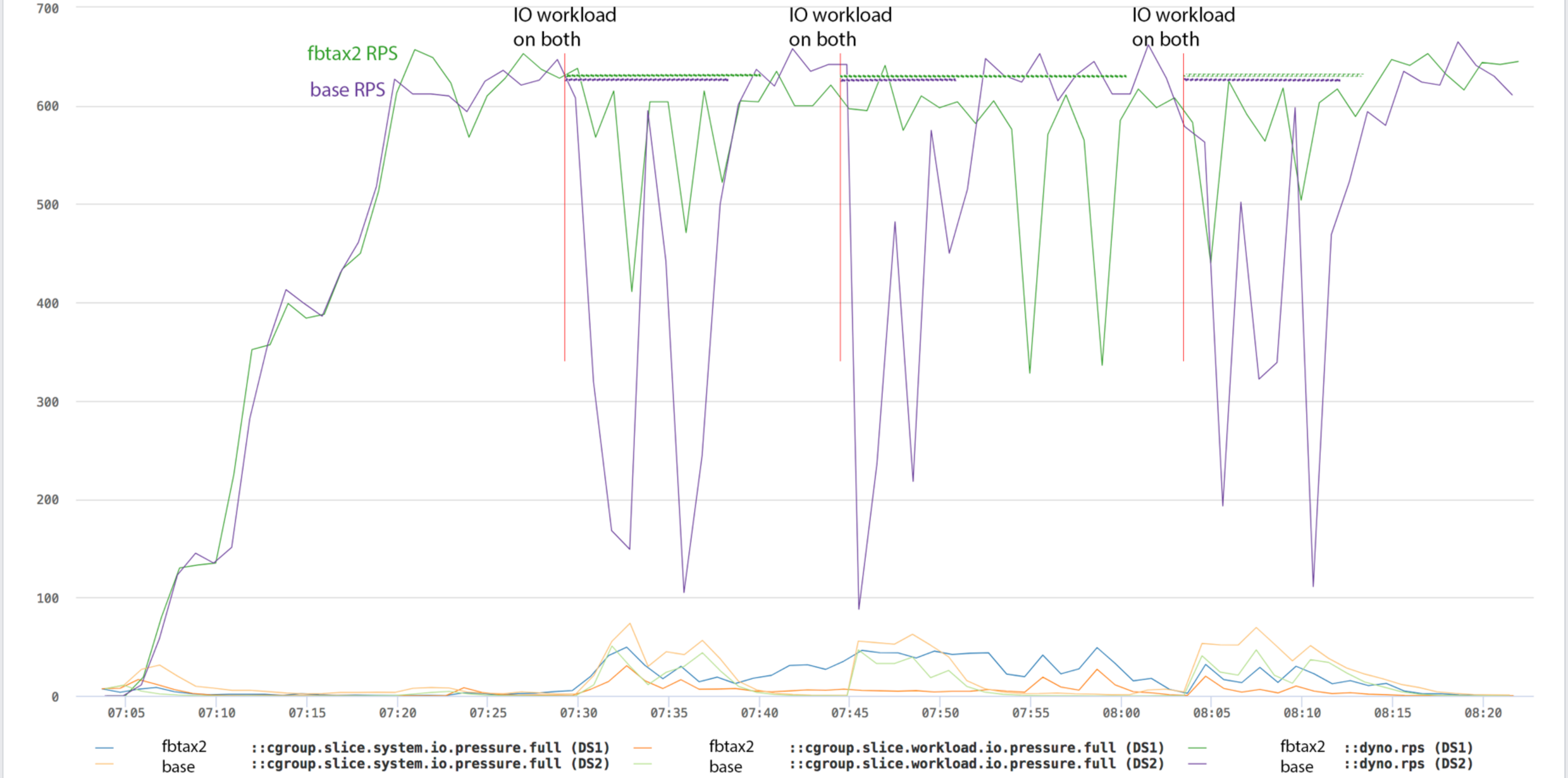
fbtax2 - oomd

- Kill a memory hog in *system* if
  - *workload* under moderate and *system* under high mempress
  - *system* under prolonged high mempress
- Kill an IO hog in *system* if
  - *workload* under moderate and *system* under high iopress
- Kill a swap hog from *system* or *workload-wdb* if
  - Swap is running out

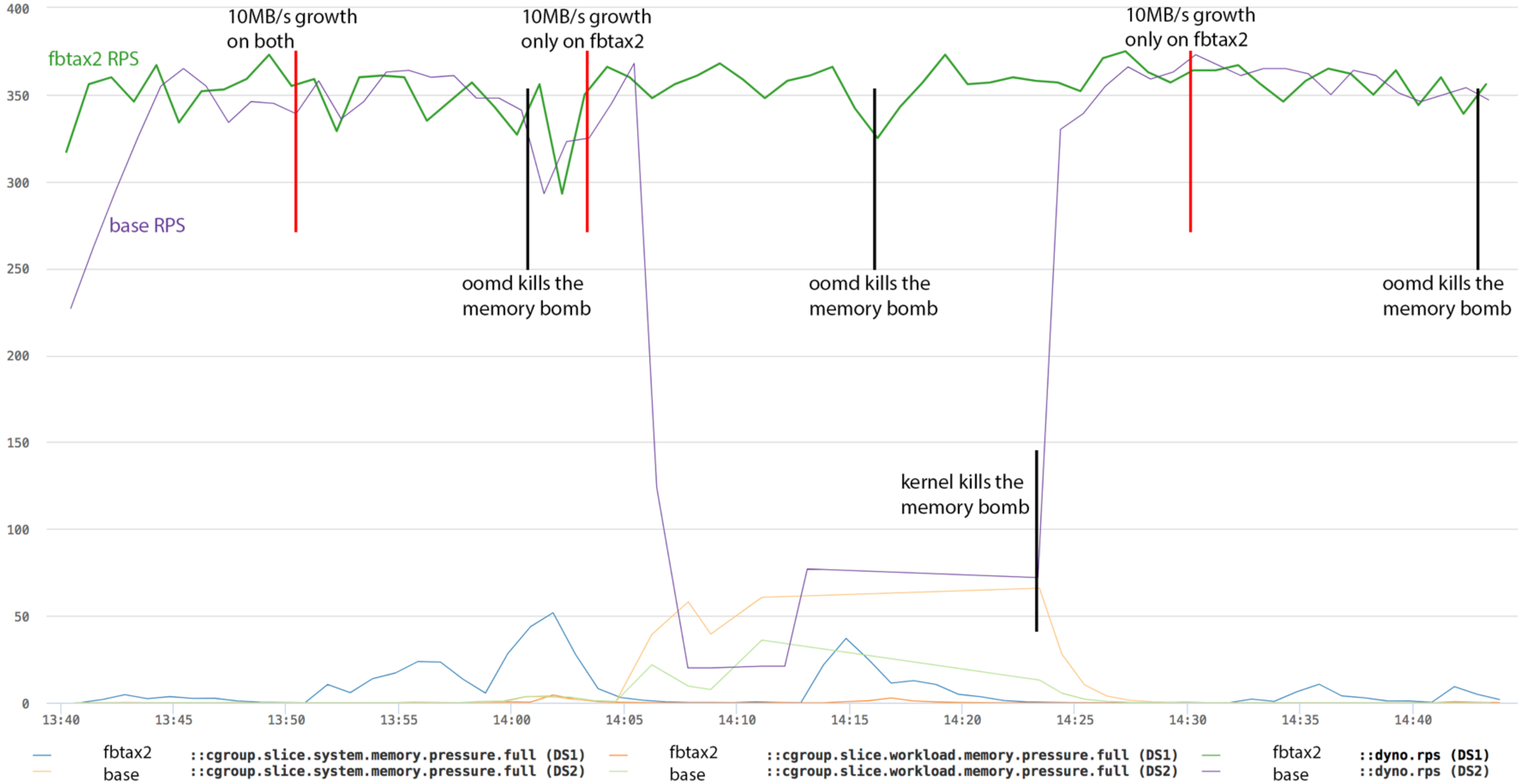
The Results on HDDs

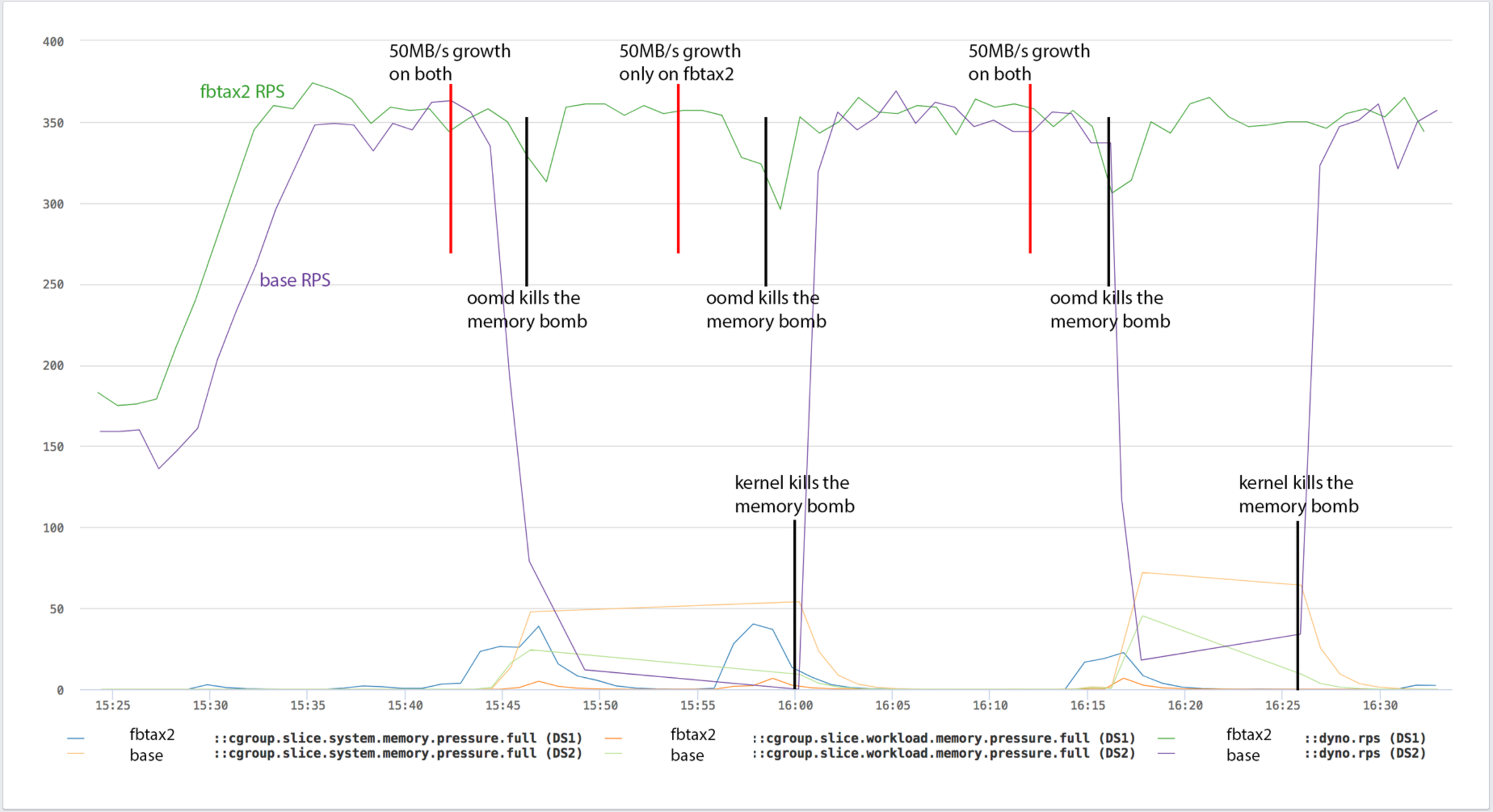

800
700
600
500
400
300
200
100
0
10MB/s growth
on both
10MB/s growth
only on fbtax2
10MB/s growth
only on fbtax2
base RPS
fbtax2 RPS
oomd kills the
memory bomb
oomd kills the
memory bomb
oomd kills the
memory bomb
base recovering
after reboot
14:10
14:20
14:30
14:40
14:50
15:00
15:10
15:20
15:30
15:40
15:50
fbtax2 ::cgroup.slice.system.memory.pressure.full (DS3)
base ::cgroup.slice.system.memory.pressure.full (DS4)
fbtax2 ::cgroup.slice.workload.memory.pressure.full (DS3)
base ::cgroup.slice.workload.memory.pressure.full (DS4)
fbtax2 ::dyno.rps (DS3)
base ::dyno.rps (DS4)


700
600
500
400
300
200
100
0
50MB/s growth
only on fbtax2
50MB/s growth
on both
50MB/s growth
only on fbtax2
fbtax2 RPS
base RPS
oomd kills the
memory bomb
base recovering
after reboot
10:00
10:10
10:20
10:30
10:40
10:50
11:00
11:10
11:20
11:30
11:40
11:50
12:00
fbtax2 ::cgroup.slice.system.memory.pressure.full (DS3)
base ::cgroup.slice.system.memory.pressure.full (DS4)
fbtax2 ::cgroup.slice.workload.memory.pressure.full (DS3)
base ::cgroup.slice.workload.memory.pressure.full (DS4)
fbtax2 ::dyno.rps (DS3)
base ::dyno.rps (DS4)


IO workload on both
IO workload on both
IO workload on both
fbtax2 RPS
base RPS
700
600
500
400
300
200
100
0
07:05
07:10
07:15
07:20
07:25
07:30
07:35
07:40
07:45
07:50
07:55
08:00
08:05
08:10
08:15
08:20
fbtax2 ::cgroup.slice.system.io.pressure.full (DS1)
base ::cgroup.slice.system.io.pressure.full (DS2)
fbtax2 ::cgroup.slice.workload.io.pressure.full (DS1)
base ::cgroup.slice.workload.io.pressure.full (DS2)
fbtax2 ::dyno.rps (DS1)
base ::dyno.rps (DS2)

The Results on SSDs


400
350
300
250
200
150
100
50
0
fbtax2 RPS
base RPS
10MB/s growth
on both
10MB/s growth
only on fbtax2
10MB/s growth
only on fbtax2
oomd kills the
memory bomb
oomd kills the
memory bomb
oomd kills the
memory bomb
kernel kills the
memory bomb
13:40
13:45
13:50
13:55
14:00
14:05
14:10
14:15
14:20
14:25
14:30
14:35
14:40
fbtax2 ::cgroup.slice.system.memory.pressure.full (DS1)
base ::cgroup.slice.system.memory.pressure.full (DS2)
fbtax2 ::cgroup.slice.workload.memory.pressure.full (DS1)
base ::cgroup.slice.workload.memory.pressure.full (DS2)
fbtax2 ::dyno.rps (DS1)
base ::dyno.rps (DS2)


400
350
300
250
200
150
100
50
0
fbtax2 RPS
base RPS
50MB/s growth
on both
50MB/s growth
only on fbtax2
50MB/s growth
on both
oomd kills the
memory bomb
oomd kills the
memory bomb
oomd kills the
memory bomb
kernel kills the
memory bomb
kernel kills the
memory bomb
15:25
15:30
15:35
15:40
15:45
15:50
15:55
16:00
16:05
16:10
16:15
16:20
16:25
16:30
fbtax2
base
::cgroup.slice.system.memory.pressure.full (DS1)
::cgroup.slice.system.memory.pressure.full (DS2)
fbtax2
base
::cgroup.slice.workload.memory.pressure.full (DS1)
::cgroup.slice.workload.memory.pressure.full (DS2)
fbtax2
base
::dyno.rps (DS1)
::dyno.rps (DS2)

The Possibilities & Todos

- We now have working full-OS resource isolation
- Batch workload side-loading
- Better thread-pool and resource consumption mgmt
- Upstreaming
- Proportion IO control for complex workload stacking

opensource.fb.com/linux

facebook