Designing and Tuning for All-Flash Ceph RBD Storage

Bloomberg Engineering

Ceph Days NYC 2024
April 26, 2024

Tyler Stachecki
Cloud Infrastructure

TechAtBloomberg.com

Designing for Scale

TechAtBloomberg.com

Bloomberg

Engineering

Designing for Scale

- Our experience with OpenStack and Ceph dates all the way back to 2013 (Essex and Giant, respectively)

- Design goals: Large clusters and dense, power-efficient compute and storage
  — ~22 OSD/servers, CPU overcommit on hypervisors

- First foray into Ceph and OpenStack taught us a lot of valuable lessons… most importantly, that L2 networks **do not scale** to many thousands of of VMs and are **hard to debug**; lots of weird interoperability issues between different vendors

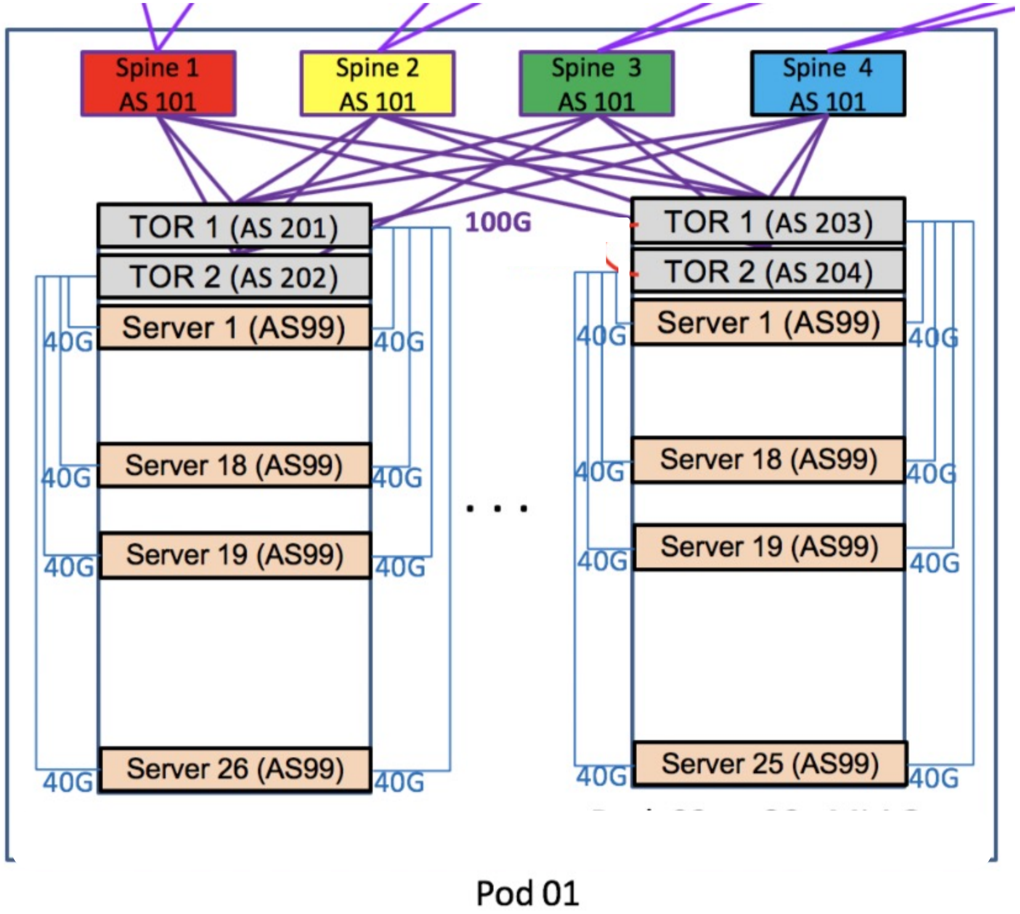
- In 2018, we began re-architecting our cloud and network…

Designing for Scale: Pure L3 fabrics

- Pure, BGP-based IP fabric with Layer 3 routing all the way down to the host for fast re-convergence on failure, stretched routing domains, no subnet constraints for tenants

- Redundancy and maximal use of links achieved through BGP and ECMP

- Multiple, disparate planes, each capable of >>1 Tbps of traffic

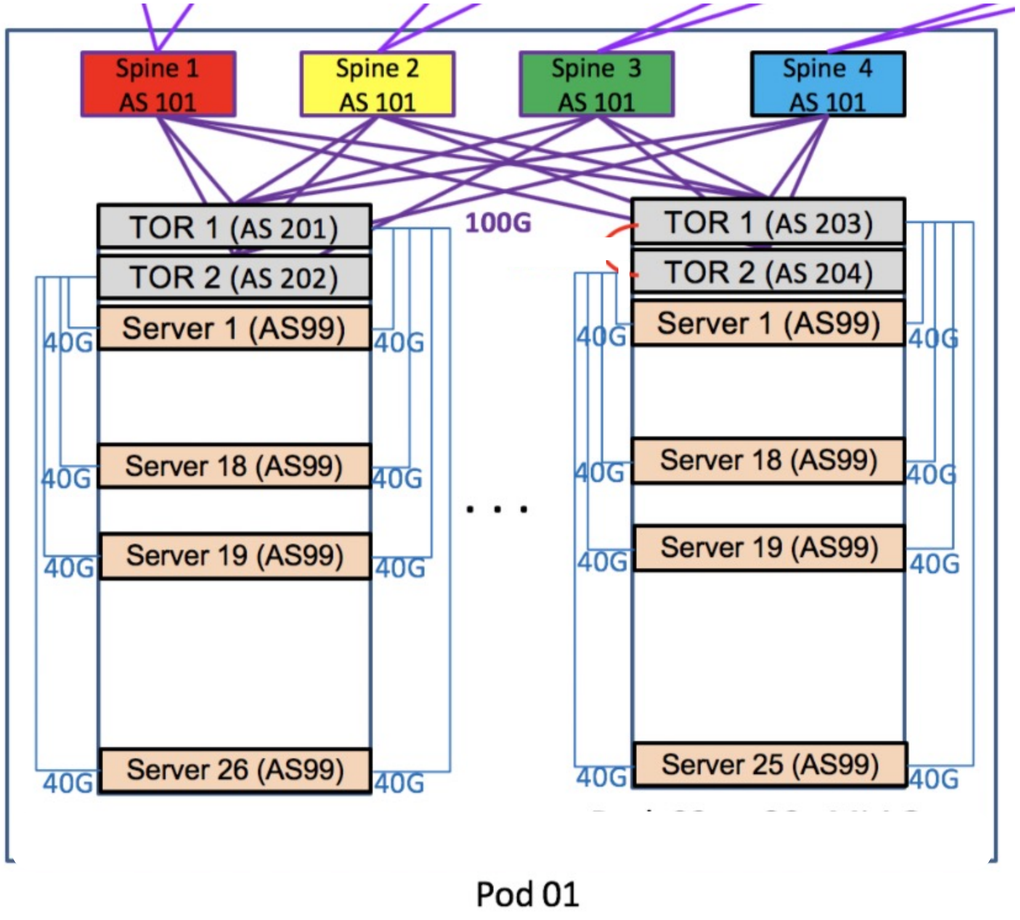


TechAtBloomberg.com

Bloomberg Engineering

Designing for Scale: Pure L3 fabrics

- Formatted and rebuilt all ~20K of our production OSDs when upgrading to Quincy: the network made it possible
- L3 fabric unifies disparate networks (control, data, storage planes, …) usually prominent in VM+RBD deployments, thus allowing for either fewer links or more available bandwidth
- Ancillary benefits incl. making rack/host movement in the data center more trivial



TechAtBloomberg.com

Bloomberg Engineering

A Story on Swapping

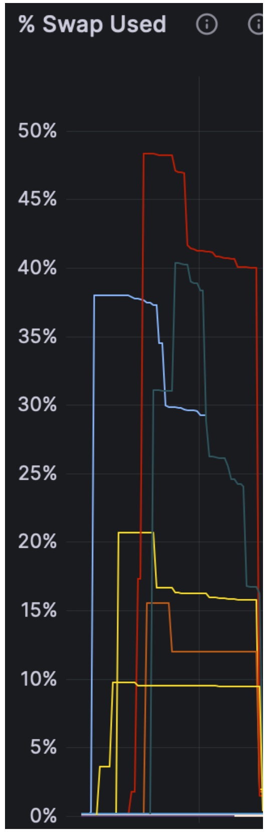
TechAtBloomberg.com

Bloomberg

Engineering

A Story on Swapping

- In 2022-2023, we underwent an initiative to renovate our Ceph clusters running RBD workloads...



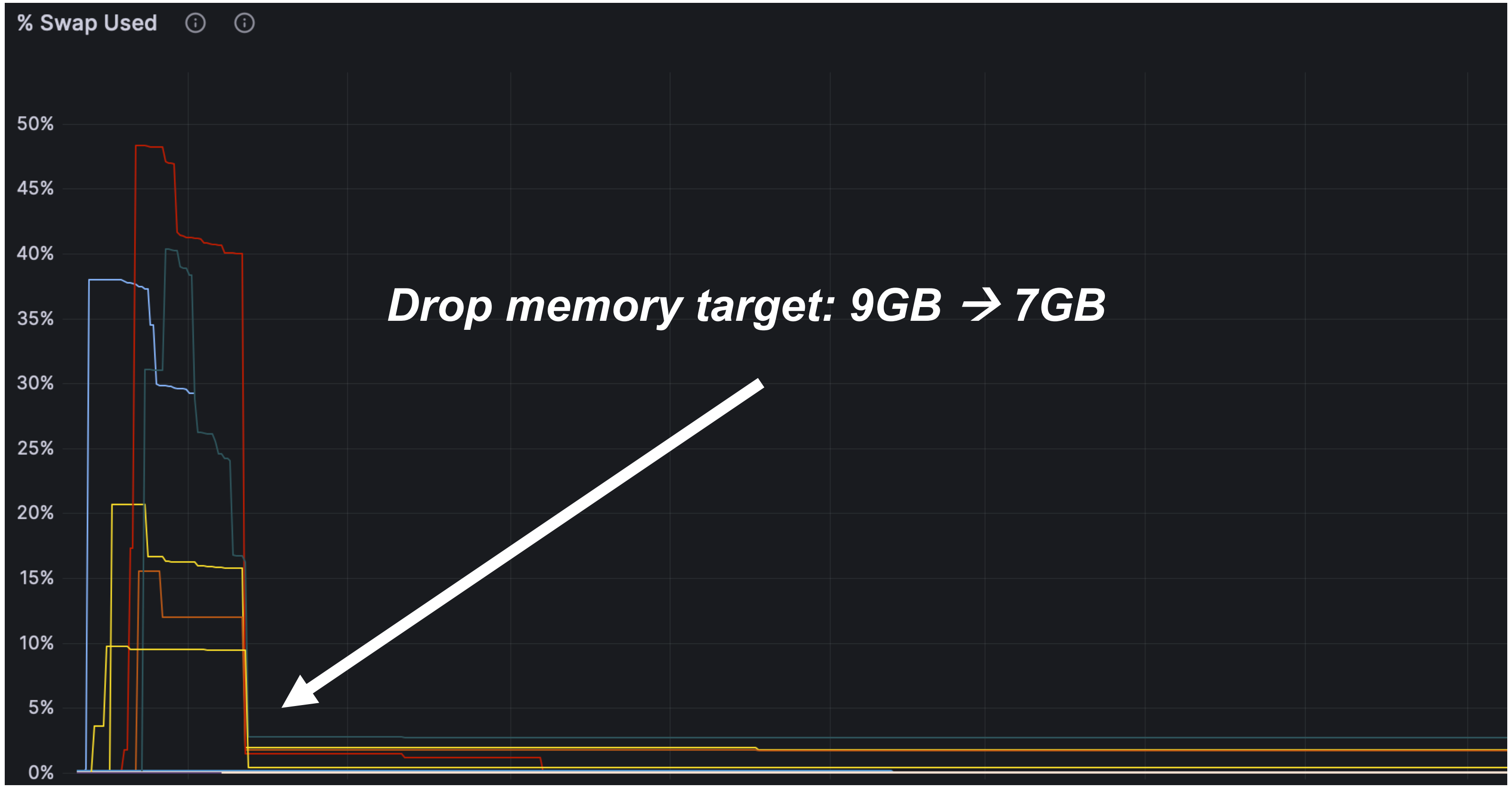
TechAtBloomberg.com

Bloomberg

Engineering

A Story on Swapping

- In 2022-2023, we underwent an initiative to renovate our Ceph clusters running RBD workloads...

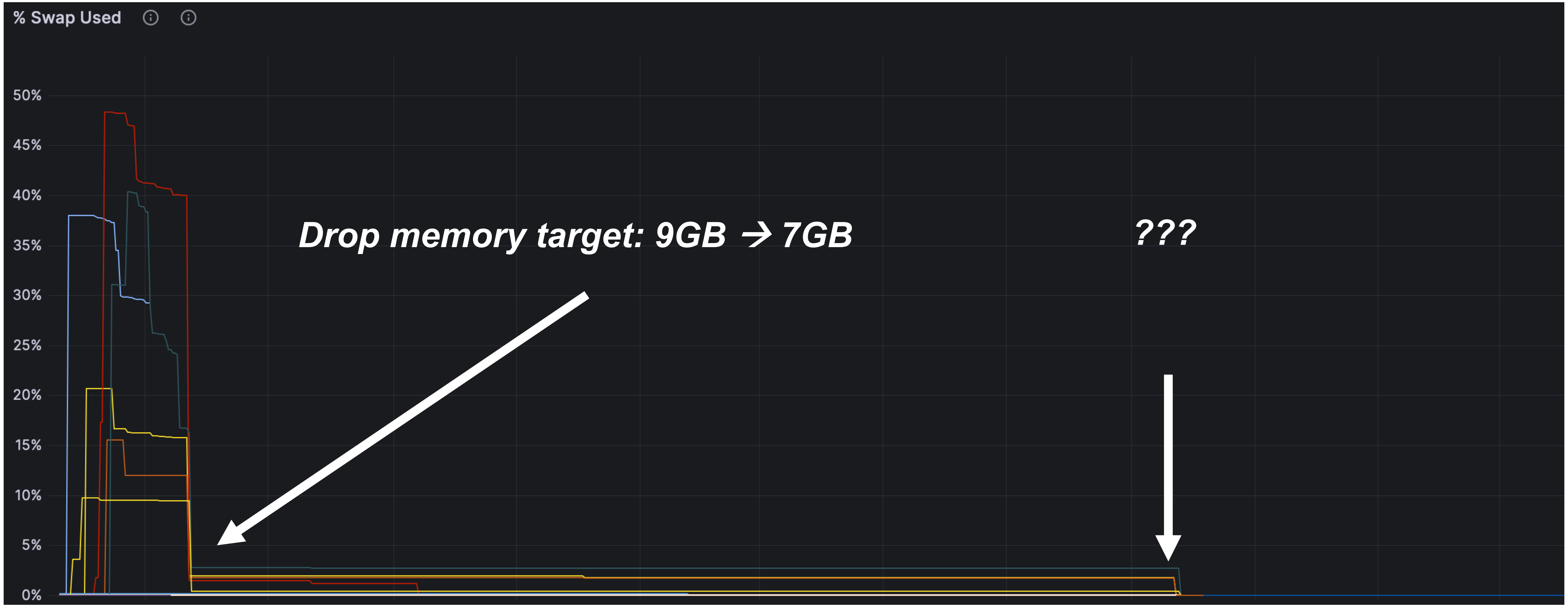


TechAtBloomberg.com

Bloomberg Engineering

A Story on Swapping

- In 2022-2023, we underwent an initiative to renovate our Ceph clusters running RBD workloads...

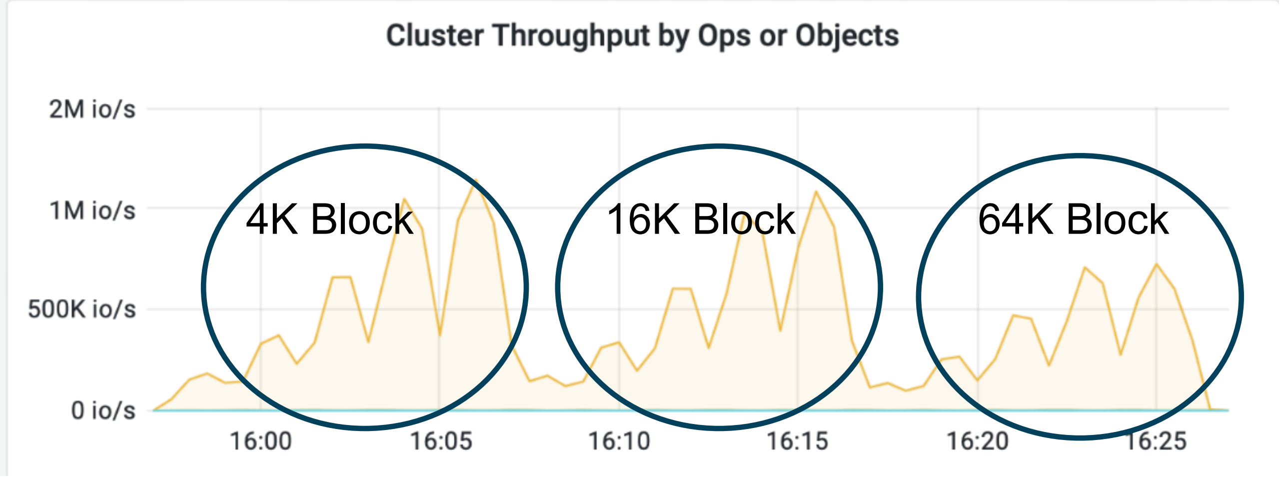


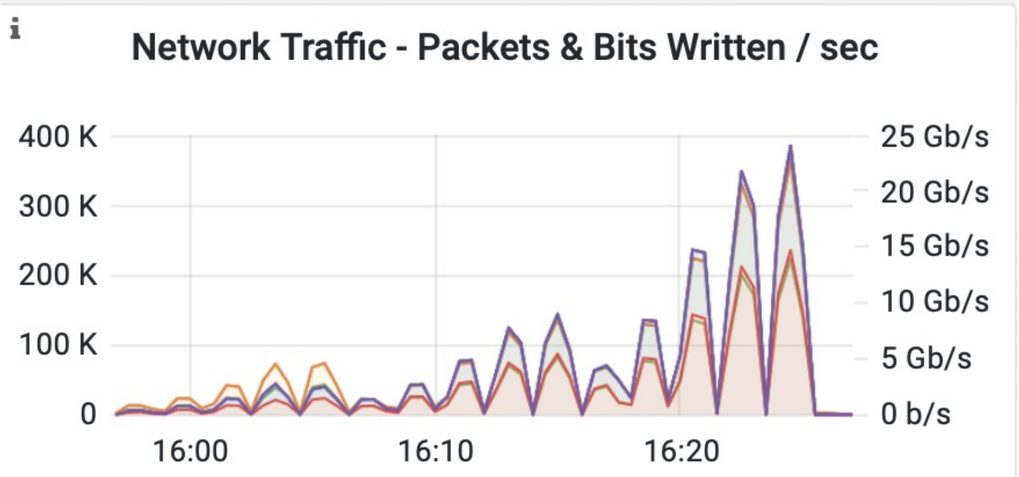
TechAtBloomberg.com

Bloomberg Engineering

A Story on Swapping

- Started setting up benchmarks on our lab cluster
  - — Metrics shown are not indicative of performance figures of our production systems
  - — 120 RBD clients running fio, 330 OSDs, 2x25Gbps networking
  - — Different r/w mixes (0/100, 100/0, 70/30), block sizes, queue depths, IOPS limits…





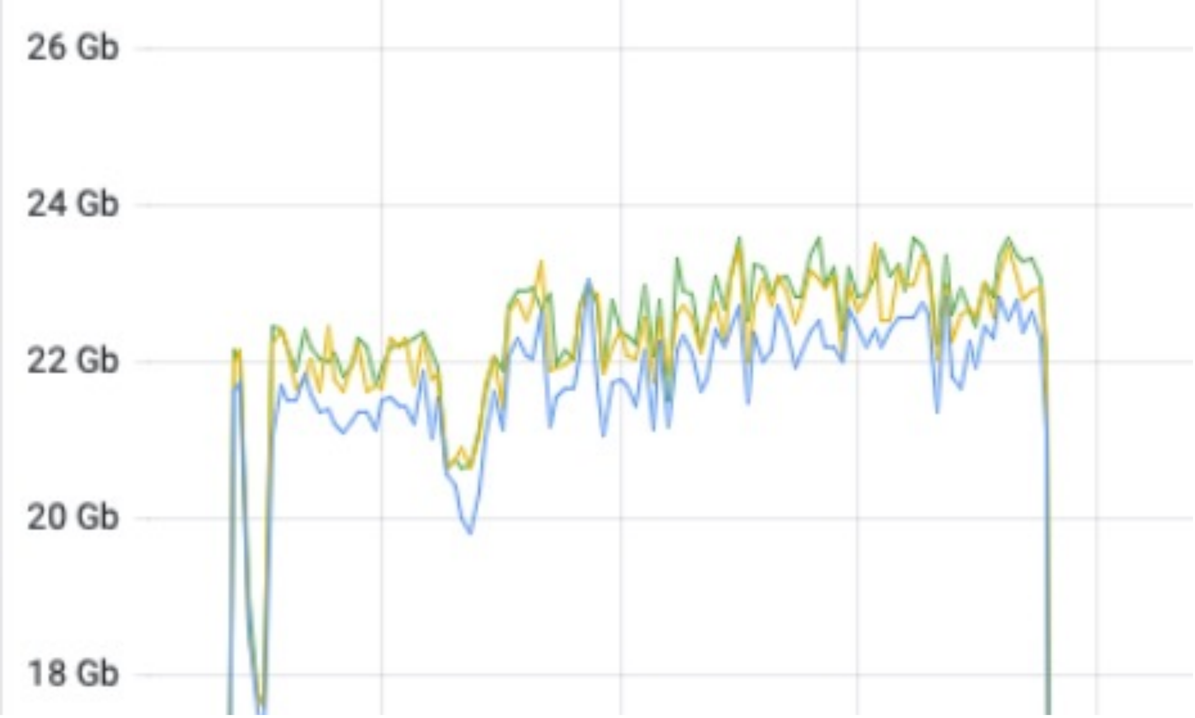
TechAtBloomberg.com

Bloomberg

Engineering

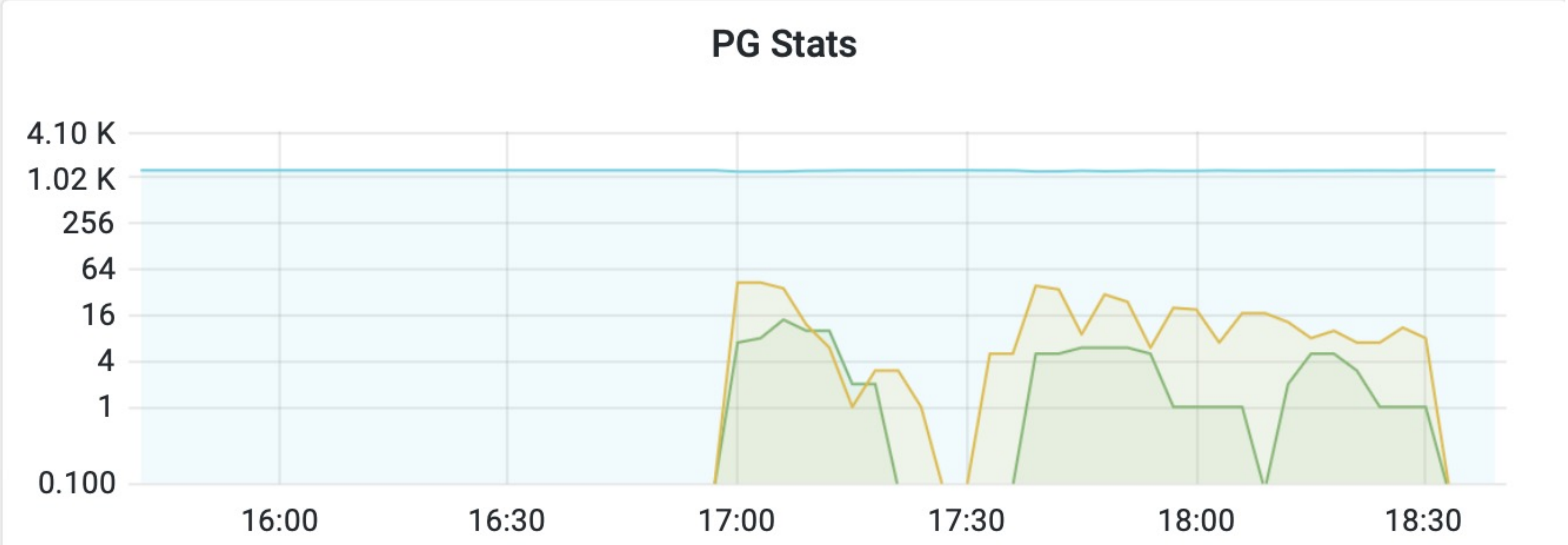
A Story on Swapping

- Early on, we identified a big problem with our benchmarking setup:
  - — 4K, 1QD Read: ~186K IOPS (**5% run-to-run variance**)
  - — 64K, 16QD Write: ~150K IOPS (**10% run-to-run-variance**)



A Story on Swapping

- Do not forget to turn off scrubbing when benchmarking…
- But, do not set it ‘off’ indefinitely either

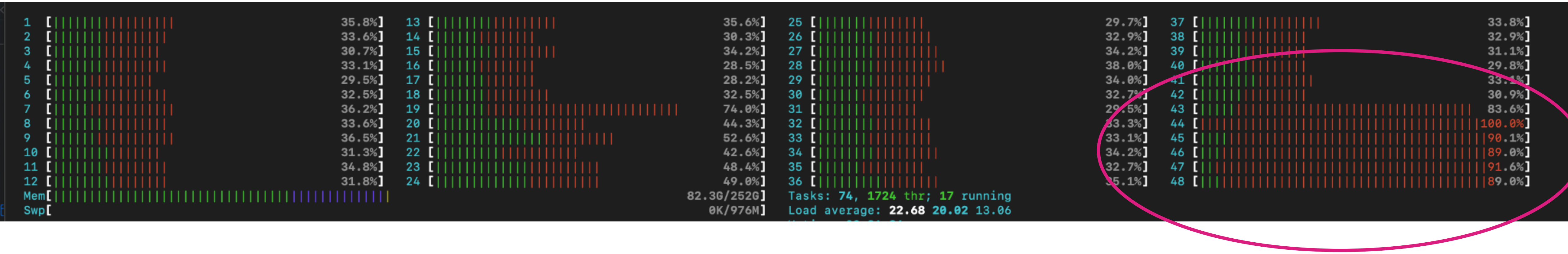


TechAtBloomberg.com

Bloomberg
Engineering

A Story on Swapping

- Why are some cores so overloaded compared to others?

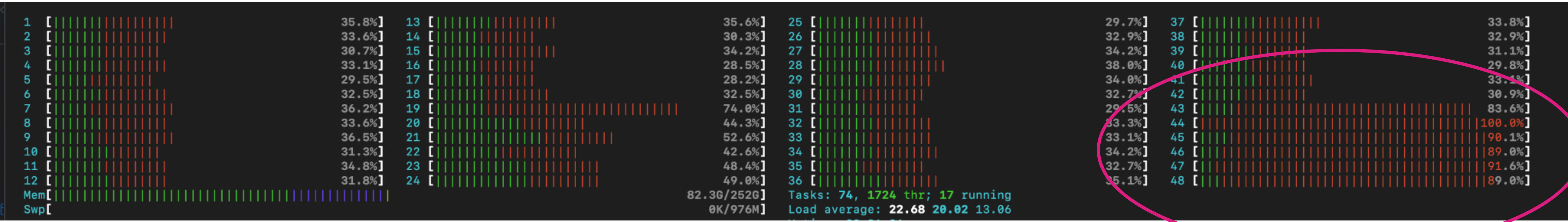


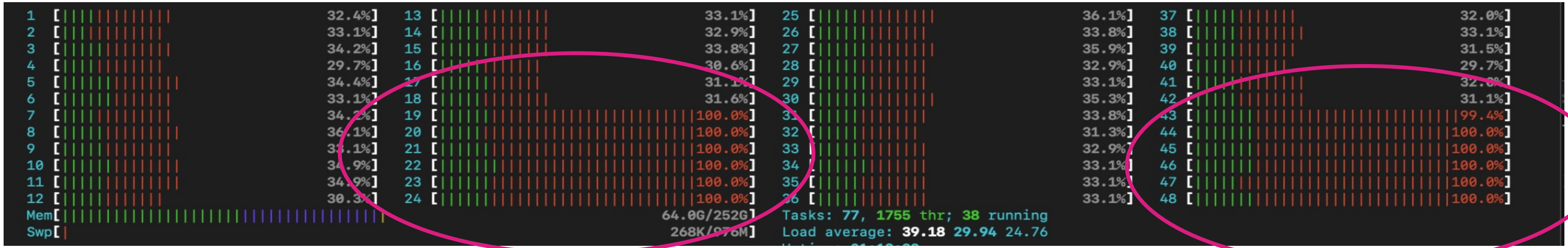
TechAtBloomberg.com

Bloomberg
Engineering

A Story on Swapping

- Why are some cores so overloaded compared to others?
- Change NUMA setting in NIC driver from local cores (default) to local threads





Bloomberg Engineering

A Story on Swapping

- `numastat`:

	node0	node1	node2	node3
numa_hit	**881983537**	**888572582**	**745775975**	**32314773050**
numa_miss	0	12848350	6769274	415898270
numa_foreign	**117258094**	**172283056**	**145974729**	**0**
interleave_hit	25851	26008	25827	25999
local_node	881974471	888538666	745745346	32314760894
other_node	9066	12882266	6799903	415910359

A Story on Swapping

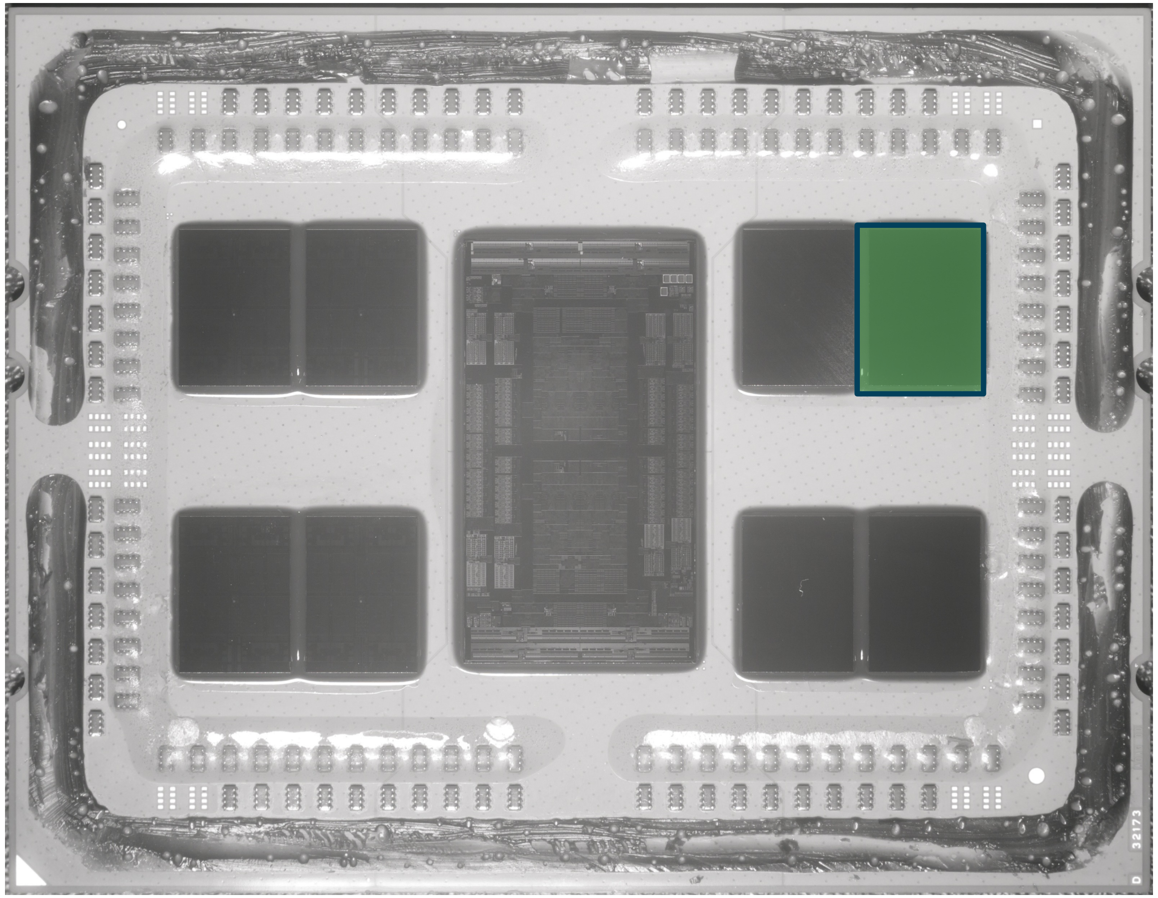
- `numastat`:

	node0	node1	node2	node3
numa_hit	**881983537**	**888572582**	**745775975**	**32314773050**
numa_miss	0	12848350	6769274	415898270
numa_foreign	**117258094**	**172283056**	**145974729**	**0**
interleave_hit	25851	26008	25827	25999
local_node	881974471	888538666	745745346	32314760894
other_node	9066	12882266	6799903	415910359

- Eureka: we were swapping in production because **some** zones were starving for memory and reacting accordingly!

- Disabling swap is the **wrong** thing to do: kernel will still page out (and likely more aggressively with swap disabled), leading to lots of page churning and memory accesses spilling into other NUMA nodes

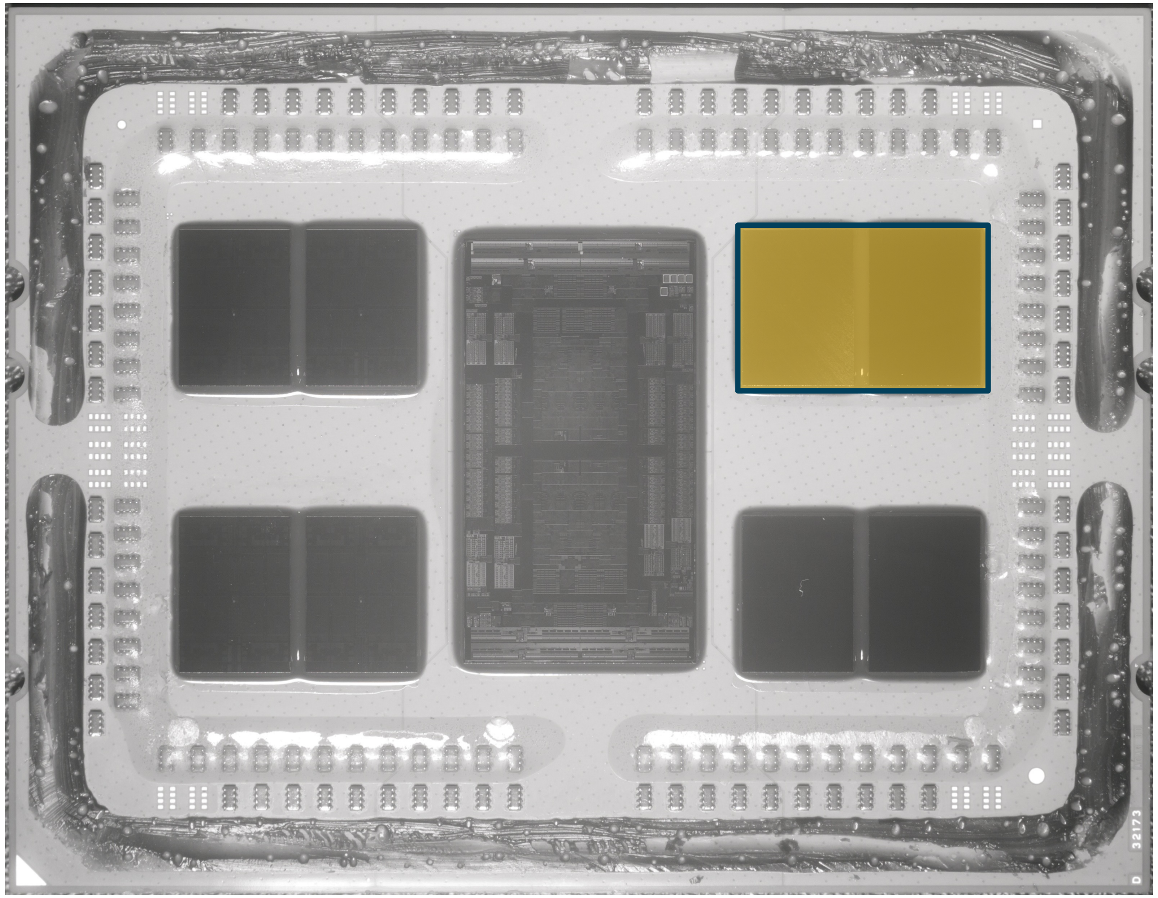
A Story on Swapping



- Intra-chiplet latency: **20-30ns**

(“AMD Epyc 7702 ES” by Fritzchens Fritz licensed under CC0 1.0 DEED)

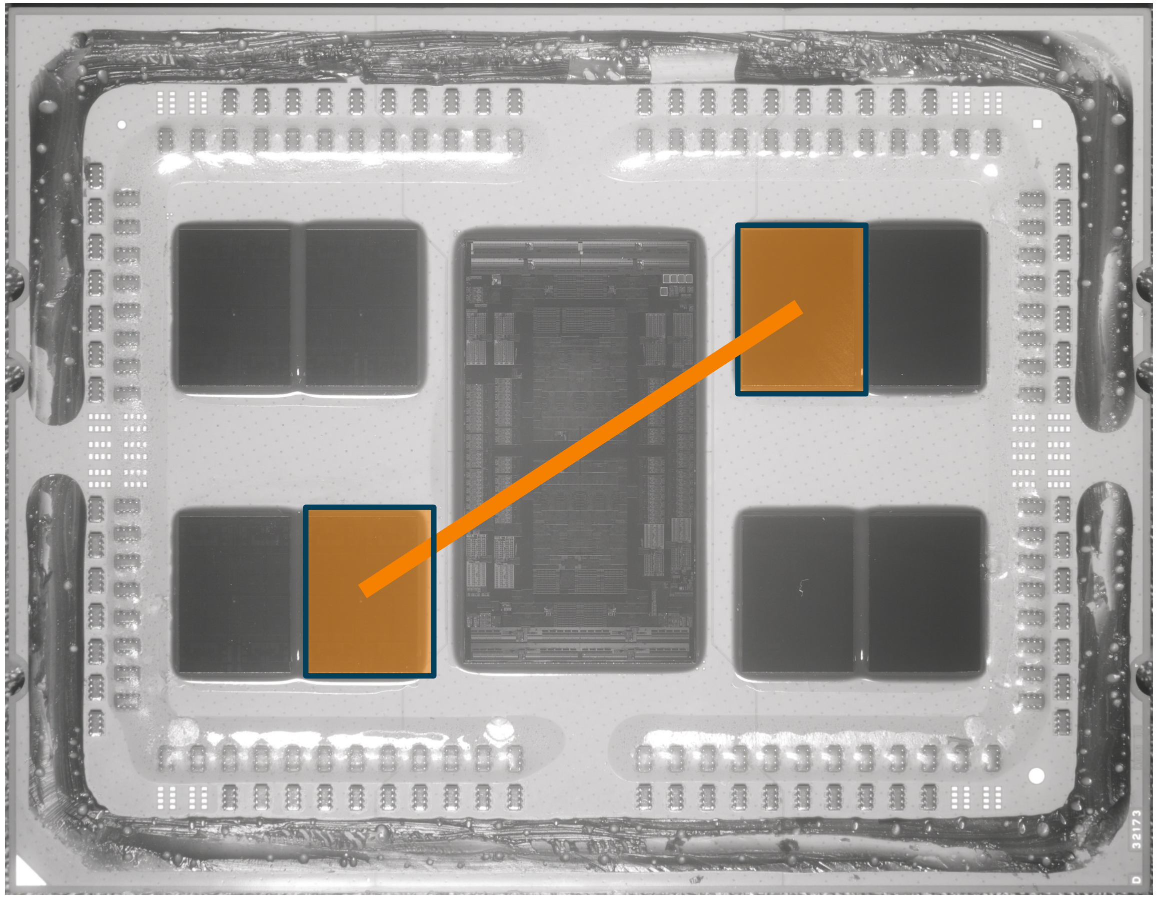
A Story on Swapping



- Intra-chiplet latency: 20-30ns
- Adjacent-chiplet latency: 80-90ns

(“AMD Epyc 7702 ES” by Fritzchens Fritz licensed under CC0 1.0 DEED)

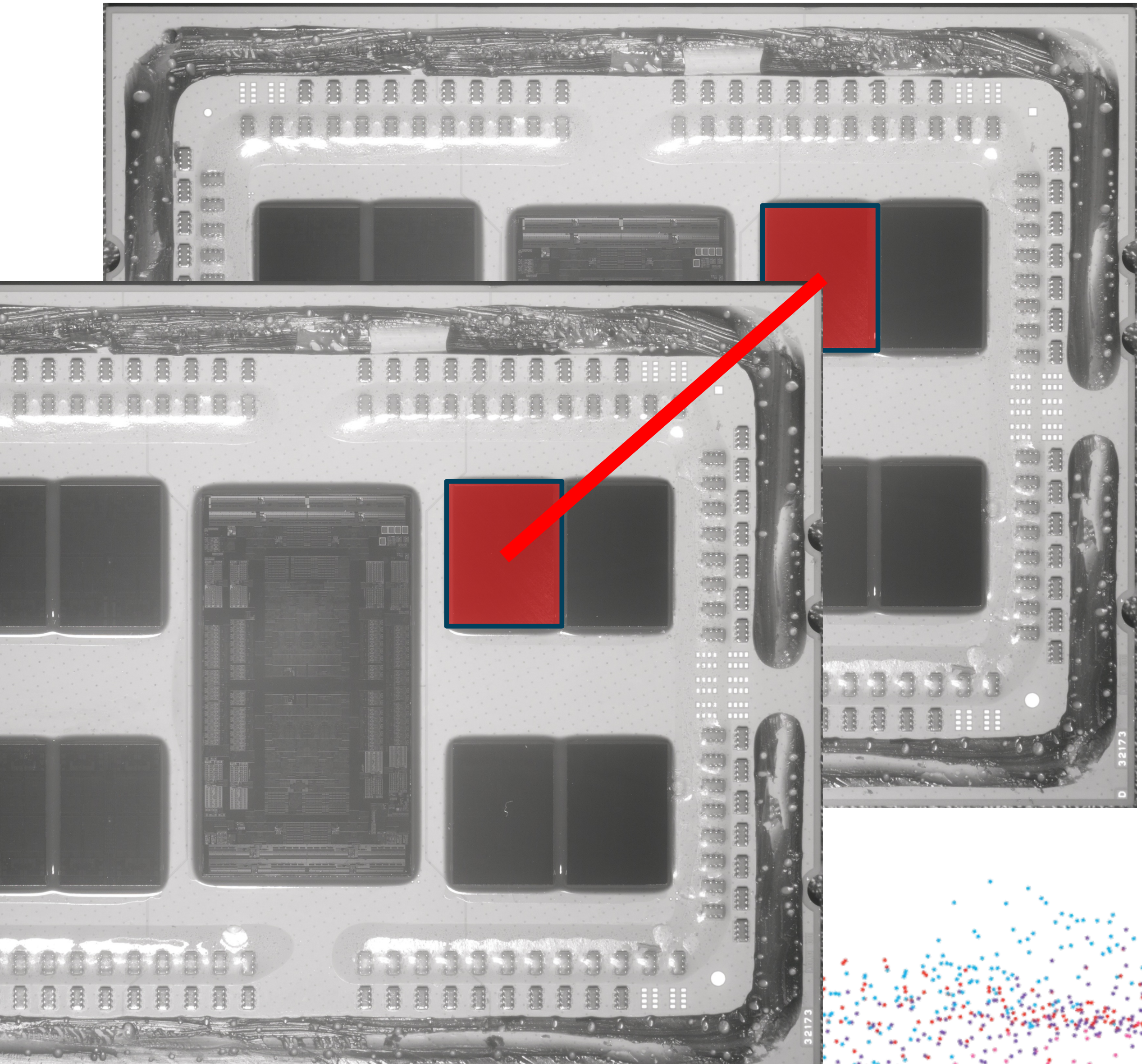
A Story on Swapping



- Intra-chiplet latency: **20-30ns**
- Adjacent-chiplet latency: **80-90ns**
- Chiplet across I/O die: **110-120ns**

(“AMD Epyc 7702 ES” by Fritzchens Fritz licensed under CC0 1.0 DEED)

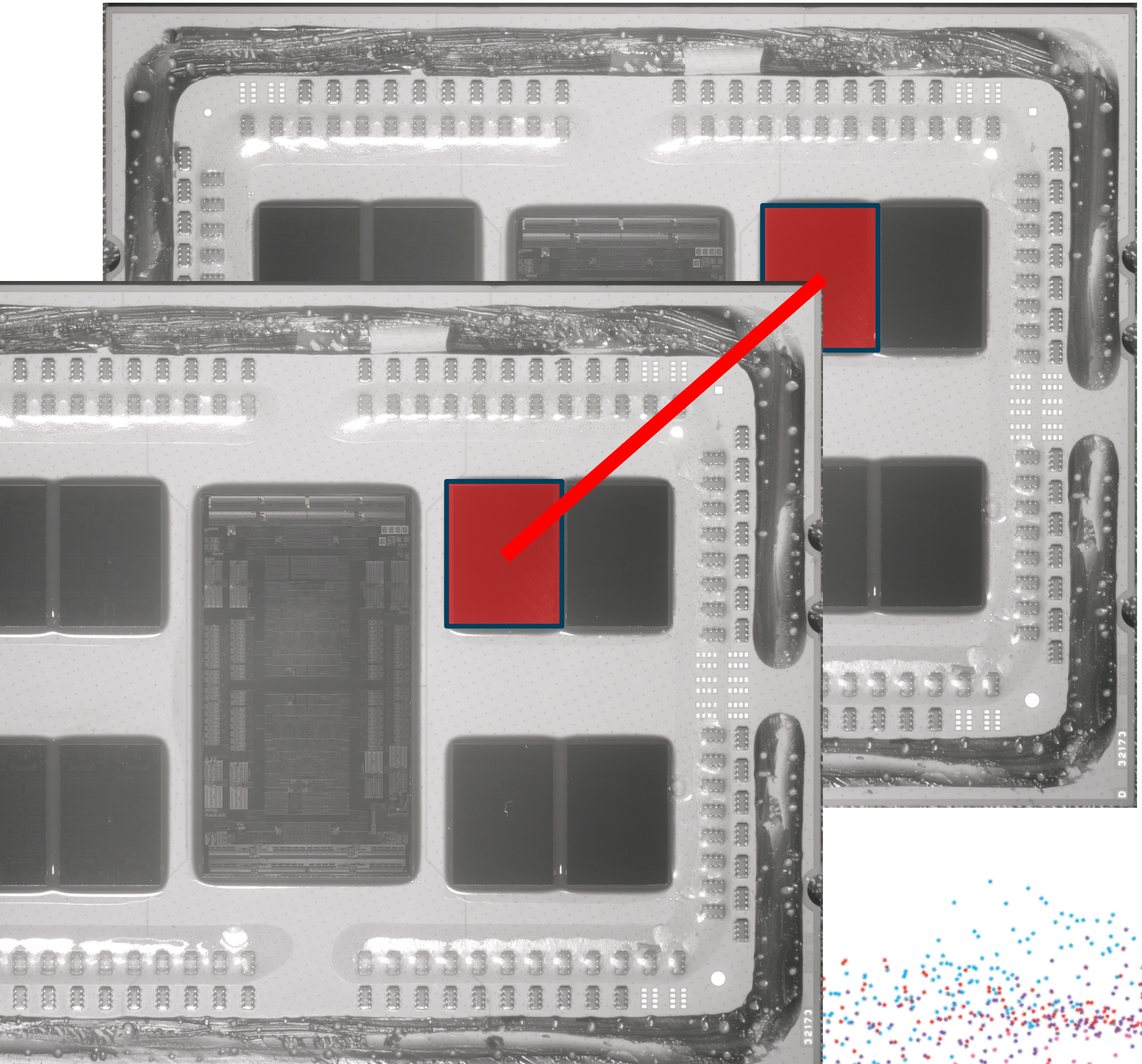
A Story on Swapping



- Intra-chiplet latency: **20-30ns**
- Adjacent-chiplet latency: **80-90ns**
- Chiplet across I/O die: **110-120ns**
- Chiplet across package: **~200ns**

Bloomberg Engineering

A Story on Swapping

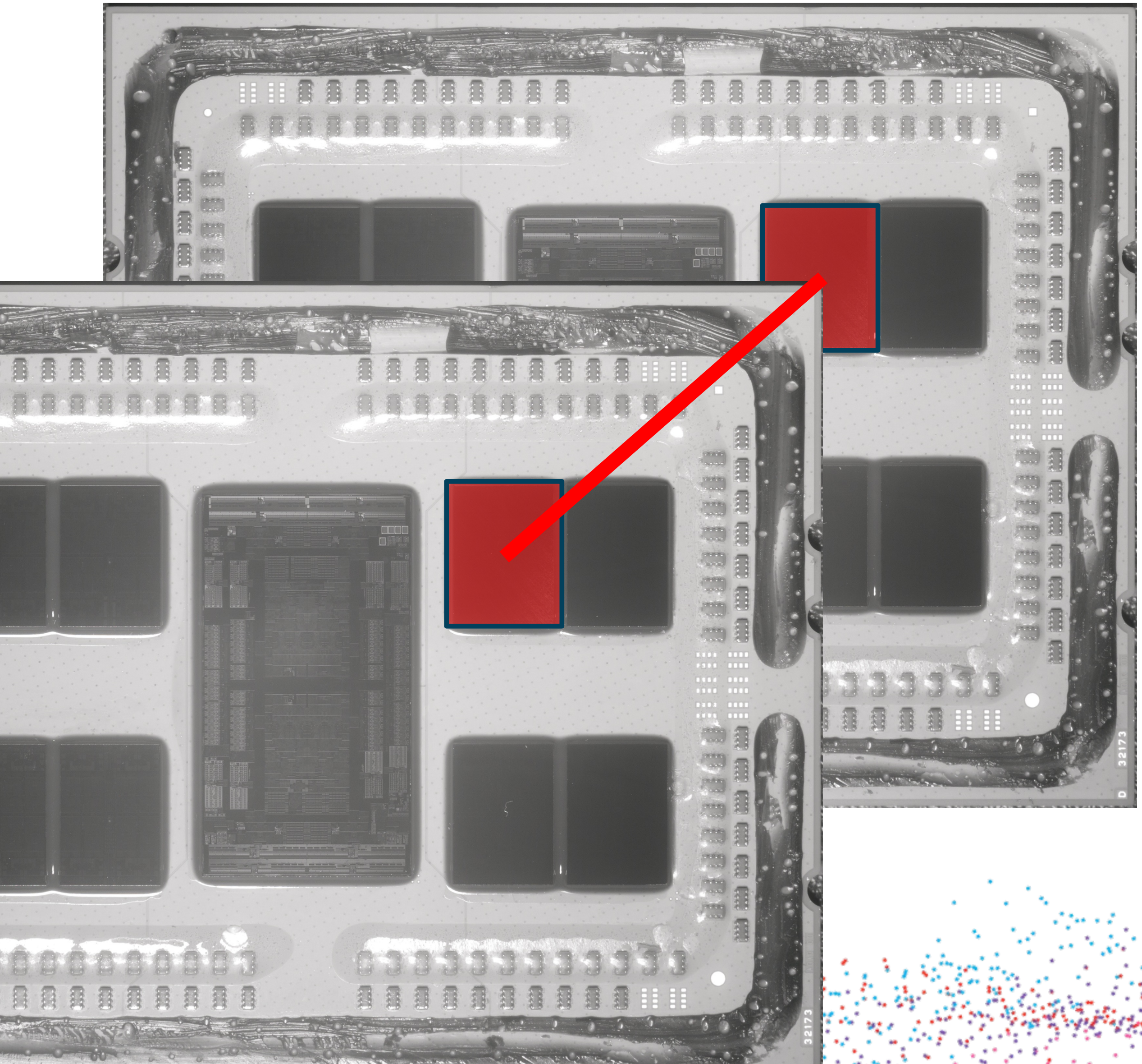


- Intra-chiplet latency: **20-30ns**
- Adjacent-chiplet latency: **80-90ns**
- Chiplet across I/O die: **110-120ns**
- Chiplet across package: **~200ns**
- Up to **~5x** latency in a 1P system
- Up to **~10x** latency in a 2P system

Bloomberg

Engineering

A Story on Swapping



- Intra-chiplet latency: **20-30ns**
- Adjacent-chiplet latency: **80-90ns**
- Chiplet across I/O die: **110-120ns**
- Chiplet across package: **~200ns**
- Up to **~5x** latency in a 1P system
- Up to **~10x** latency in a 2P system
- *Most HW vendors "hide" the chiplet-level topology (8 NUMA zones) from the OS by default and only show 4 NUMA zones!*

Bloomberg Engineering

A Story on Swapping

- Kernel does not know how to optimally configure multi-process workloads like Ceph
- Our Ceph block servers use a systemd service shim to configure Ceph with a-priori knowledge:

```
$ cat /etc/systemd/system/ceph-osd@.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/bin/bbcephtool exec_osd -f --cluster ${CLUSTER} --id %i --setuser ceph --setgroup ceph
```

- `bbcephtool` probes the system for *all* OSDs and performs global scheduling
- It instructs the kernel to assign ***this*** OSD to a particular chiplet in order to leverage the locality and restricted coherence domain of modern chiplet-based microarchitectures

A Story on Swapping

Before:

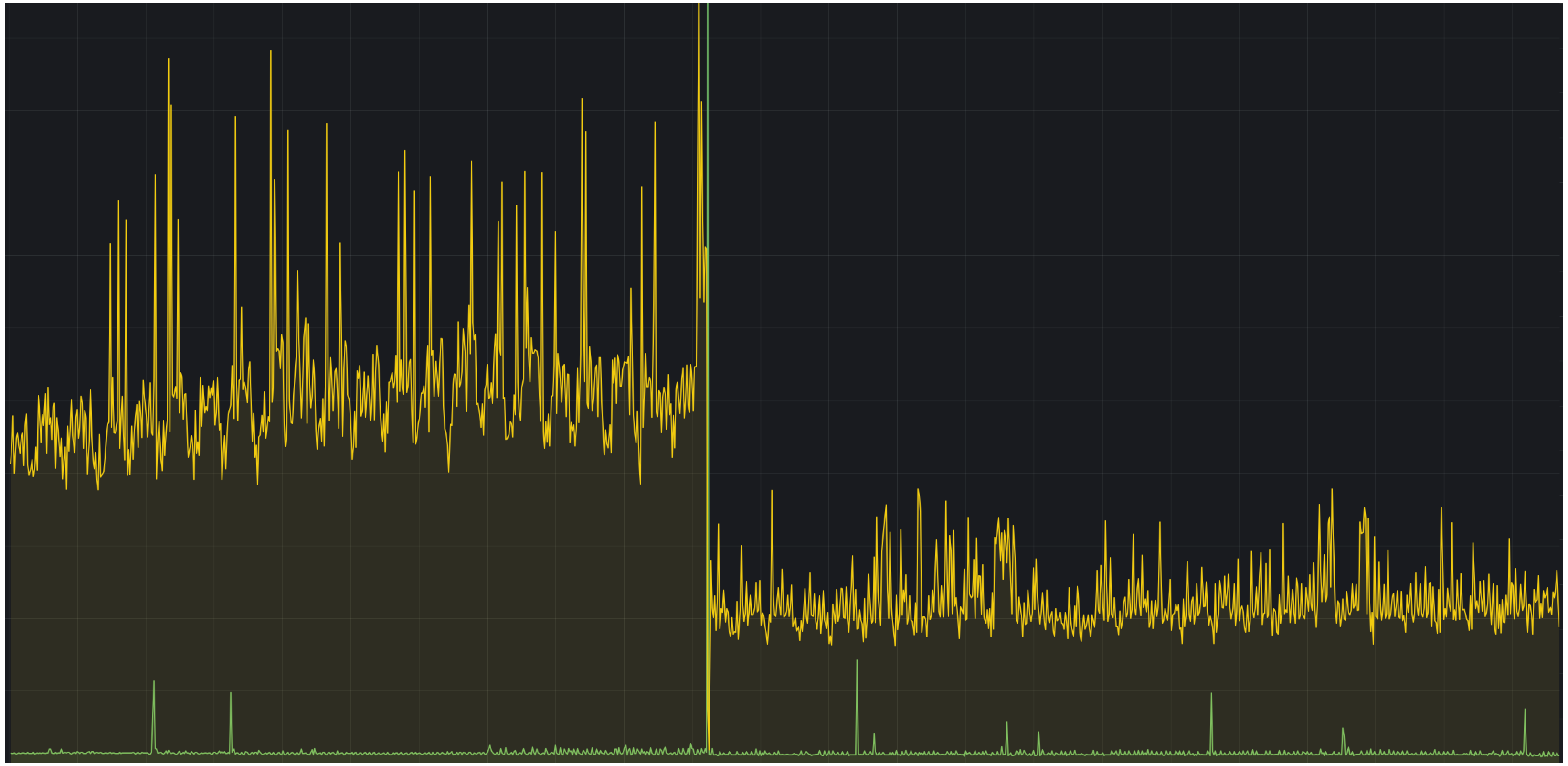
	node0	node1	node2	node3
numa_hit	**881983537**	**888572582**	**745775975**	**32314773050**
numa_miss	0	12848350	6769274	415898270
numa_foreign	**117258094**	**172283056**	**145974729**	**0**
interleave_hit	25851	26008	25827	25999
local_node	881974471	888538666	745745346	32314760894
other_node	9066	12882266	6799903	415910359

After:

	node0	node1	node2	node3
numa_hit	**45697885068**	**818961266817**	**57096939504**	**60297664855**
numa_miss	723914	4654022	27846460	407840
numa_foreign	**4629624**	**27868400**	**387372**	**746840**
interleave_hit	17523	17106	17532	17094
local_node	45697875953	818938581714	57096806186	60297320431
other_node	797371	4867960	27951674	872478

A Story on Swapping

- Number of context switches on a production host as we deploy the changes…



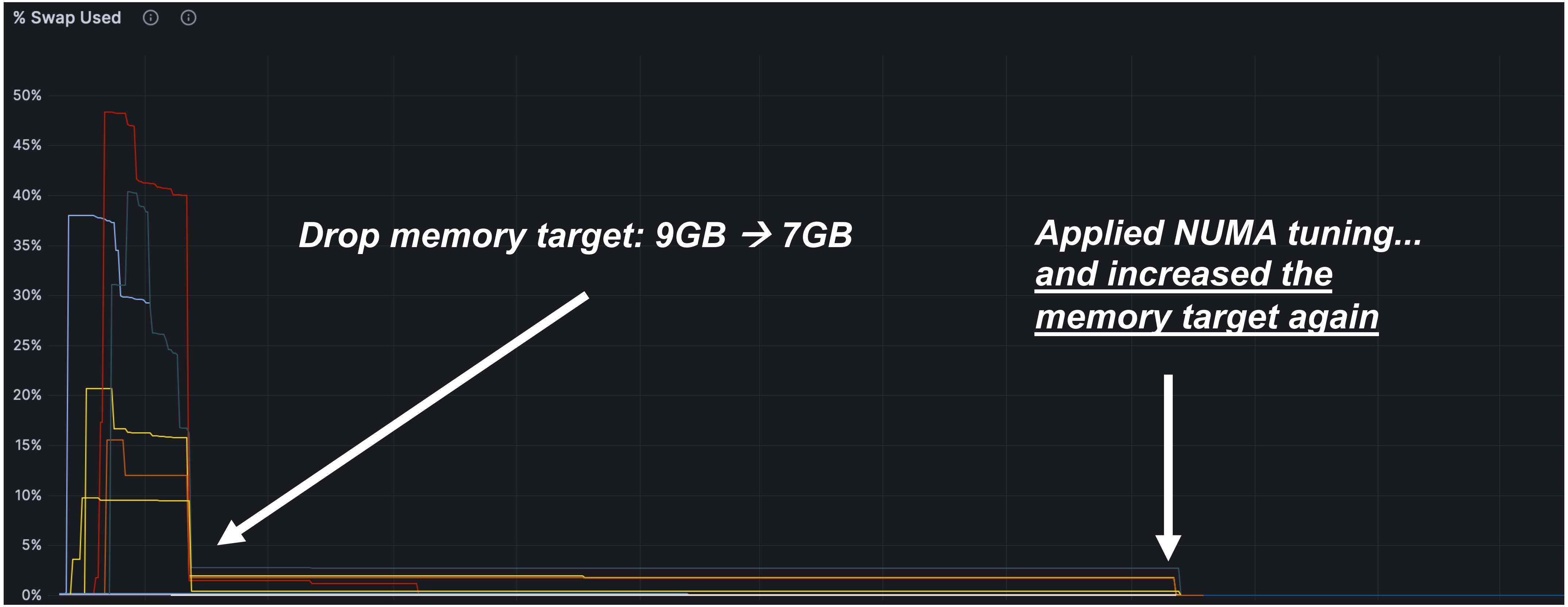
A Story on Swapping

- Remember that variance we saw in our lab cluster before?

	Before Tuning	After Tuning
4K, 1QD Read IOPS	186K (5% variance)	215K (<1% variance)
64K, 16QD Write IOPS	150K (10% variance)	181K (~2.5% variance)

A Story on Swapping

- NUMA tuning's purpose is not only memory latency:



A Story on Swapping

- Have a solid understanding of what you are changing and why – Do not make reactionary decisions

- NUMA tuning is quintessential for **consistent** performance in deployments targeting a dense number of OSDs/host

- Look at your BIOS settings to see if you are forgoing “sub-” NUMA optimizations; you probably are, unless you looked already

Hey, we saw this, too!

- *“That's when I noticed that we were not, in fact, building RocksDB with the correct compile flags. It's not clear how long that's been going on…”* – [ceph.io blog post](), Jan 19, 2024

Hey, we saw this, too!

- *“That's when I noticed that we were not, in fact, building RocksDB with the correct compile flags. It's not clear how long that's been going on…”*

– ceph.io blog post, Jan 19, 2024

Tyler Stachecki (ts achecki) wrote on 2020-09-10:

```
Checked out ceph-15.2.3 source on a up-to-date focal VM.

When running dpkg-buildpackage, CMakeCache.txt gets generated as expected. However, the generated CMAKE_BUILD_TYPE is "None".

A bit odd, but going a step further, one can grep for CMAKE_C_FLAGS in that same file and see something to the effect of:
CMAKE_C_FLAGS_DEBUG:STRING=<sensible compiler settings for debug builds>
CMAKE_C_FLAGS_NONE:STRING=
...
...
CMAKE_C_FLAGS_RELWITHDEBINFO:STRING=<sensible compiler settings for release builds>

It seems, though, that since "None" is used, all of those are actually ignored and
CMAKE_C_FLAGS:STRING=<compiler flags from dpkg-buildpackage>
is used
```

Hey, we saw this, too!

- *“That's when I noticed that we were not, in fact, building RocksDB with the correct compile flags. It's not clear how long that's been going on…”*
  – ceph.io blog post, Jan 19, 2024

- (oops… Bloomberg had known about this, and we *really* should have made the upstream contributions to fix it)

- Hopefully, today, we will make up for that by sharing some of the findings we have discovered since then… and share more going forward

Here Today, Gone Tomorrow

TechAtBloomberg.com

Bloomberg Engineering

Here Today, Gone Tomorrow

- Your cluster is only as fast as your slowest OSD
- In an all-flash cluster, if you have a sluggish OSD, this becomes especially noticeable!
- Sometimes, the slowest OSD right now is a “quick” OSD just shortly later (examples to come)
- Telemetry can put you in front of what’s slow *right now*
- Fixing what’s infrequently very slow improves your worst-case scenarios

Here Today, Gone Tomorrow

- Your cluster is only as fast as your slowest OSD
- In an all-flash cluster, if you have a sluggish OSD, this becomes especially noticeable!
- Sometimes, the slowest OSD right now is a “quick” OSD just shortly later (examples to come)
- Telemetry can put you in front of what’s slow *right now*
- Fixing what’s infrequently very slow improves your worst-case scenarios
- <u>Your “worst case” is what your users really care about</u>

Here Today, Gone Tomorrow

- Before we “pick” on Ceph, let’s first walk through a mistake we made ourselves

- There’s a popular RocksDB setting online that looks similar to this:

```
default[‘bcpc’][‘ceph’][‘bluestore_rocksdb_options’] = [
  ‘compression=kNoCompression’,
  ‘max_write_buffer_number=4’,
  ‘min_write_buffer_number_to_merge=1’,
  ‘recycle_log_file_num=4’,
  ‘write_buffer_size=268435456’,
  ’writeable_file_max_buffer_size=0’,
  ’compaction_readahead_size=2097152’,
  ‘max_background_compactions=4’,
]
```

- Similar settings appear in multiple vendor whitepapers, online searches, … and actually work quite well for Ceph versions from the era for which the setting was published

Here Today, Gone Tomorrow

- Then came RocksDB column families (a good thing!)
- … but they necessitate an additional option to keep WAL sizes at sane levels (#35277)

Here Today, Gone Tomorrow

```
2 src/common/options.cc

@@ -4413,7 +4413,7 @@ std::vector<Option> get_global_options() {
    .set_description("max duration to force deferred submit"),

    Option("bluestore_rocksdb_options", Option::TYPE_STR, Option::LEVEL_ADVANCED)
-
  .set_default("compression=kNoCompression,max_write_buffer_number=4,min_write_buffer_numb
  ion_readahead_size=2097152,max_background_compactions=2")
+
  .set_default("compression=kNoCompression,max_write_buffer_number=4,min_write_buffer_numb
  ion_readahead_size=2097152,max_background_compactions=2,max_total_wal_size=1073741824")
    .set_description("Rocksdb options"),

    Option("bluestore_rocksdb_cf", Option::TYPE_BOOL, Option::LEVEL_ADVANCED)
```

TechAtBloomberg.com

Bloomberg

Engineering

Here Today, Gone Tomorrow

- Then came RocksDB column families (a good thing!)
- … but they necessitate an additional option to keep WAL sizes at sane levels (#35277)
- without `max_total_wal_size` being appended to your RocksDB settings, you will experience insufferably bad latencies when OSDs need to compact WALs (that grow to ~100GB…)
- The setting is not additive or a default, so without specifying it explicitly as part of your overridden RocksDB options, your cluster will suffer
- In most cases, the effects of not including the setting will take >1d to manifest

Here Today, Gone Tomorrow

- Extreme case of this: once in a blue moon on a specific platform, we see hardware failures that manifest as missed interrupts from the NVMe. The IOP is then polled and completes.

- NVMe timeout in Linux defaults to… what?

```
$ cat /sys/module/nvme_core/parameters/io_timeout
30
```

- After 30 seconds, the kernel polls the NVMe to say hey… about that I/O access…

- If this keeps happening, Ceph is self-healing and marks the OSD out, right?

Here Today, Gone Tomorrow

- Extreme case of this: once in a blue moon on a specific platform, we see hardware failures that manifest as missed interrupts from the NVMe. The IOP is then polled and completes.

- NVMe timeout in Linux defaults to… what?

```
$ cat /sys/module/nvme_core/parameters/io_timeout
30
```

- After 30 seconds, the kernel polls the NVMe to say hey… about that I/O access…

- If this keeps happening, Ceph is self-healing and marks the OSD out, right?

- … right?

```
$ sudo ceph daemon osd.X config get osd_op_thread_suicide_timeout | jq -r
.osd_op_thread_suicide_timeout
150
```

Here Today, Gone Tomorrow

- Different case: Let us look at an outlier in a cluster of 4K+ OSDs & >10K+ RADOS clients

- `perf top -p 66111`

```
Samples: 28K of event 'cycles', 4000 Hz, Event count (approx.): 6274841861 lost: 0/0 drop: 0/0
Overhead  Shared Object          Symbol
  19.09%  ceph-osd               [.] crc32_iscsi_00
   3.13%  [kernel]               [k] copy_user_generic_string
   2.70%  libtcmalloc.so.4.5.9   [.] operator new[]
   1.65%  [kernel]               [k] nft_do_chain
   0.88%  [kernel]               [k] native_write_msr
   0.85%  libtcmalloc.so.4.5.9   [.] operator delete[]
   0.84%  libtcmalloc.so.4.5.9   [.] tcmalloc::CentralFreeList::FetchFromOneSpans
   0.80%  [kernel]               [k] clear_page_rep
   0.80%  libtcmalloc.so.4.5.9   [.] tcmalloc::ThreadCache::ReleaseToCentralCache
   0.79%  [kernel]               [k] iommu_v1_map_page
   0.76%  ceph-osd               [.] rocksdb::InlineSkipList<rocksdb::MemTableRep::KeyComparator const&>::R
   0.67%  [kernel]               [k] memset
```

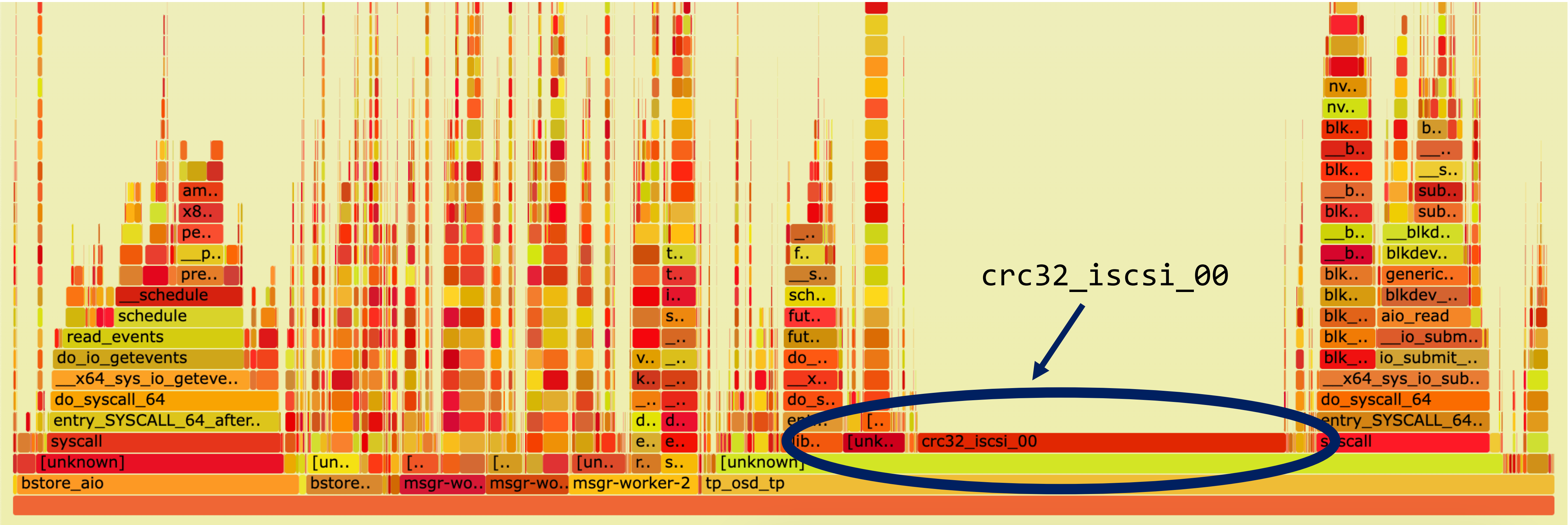
Here Today, Gone Tomorrow

- Different case: Let us look at an outlier in a cluster of 4K+ OSDs & >10K+ RADOS clients
- `perf top -p 66111`

```
Samples: 28K of event 'cycles', 4000 Hz, Event count (approx.): 6274841861 lost: 0/0 drop: 0/0
Overhead  Shared Object          Symbol
  19.09%  ceph-osd               [.] crc32_iscsi_00
   3.13%  [kernel]               [k] copy_user_generic_string
   2.70%  libtcmalloc.so.4.5.9   [.] operator new[]
   1.65%  [kernel]               [k] nft_do_chain
   0.88%  [kernel]               [k] native_write_msr
   0.85%  libtcmalloc.so.4.5.9   [.] operator delete[]
   0.84%  libtcmalloc.so.4.5.9   [.] tcmalloc::CentralFreeList::FetchFromOneSpans
   0.80%  [kernel]               [k] clear_page_rep
   0.80%  libtcmalloc.so.4.5.9   [.] tcmalloc::ThreadCache::ReleaseToCentralCache
   0.79%  [kernel]               [k] iommu_v1_map_page
   0.76%  ceph-osd               [.] rocksdb::InlineSkipList<rocksdb::MemTableRep::KeyComparator const&>::R
   0.67%  [kernel]               [k] memset
   0.67%  libc.so.6              [.] pthread_mutex_lock
```

Here Today, Gone Tomorrow

- `perf record -gp 66111 -- sleep 30; perf script > stack.out`
- `./stackcollapse-perf.pl < stack.out | ./flamegraph.pl > osd.svg`

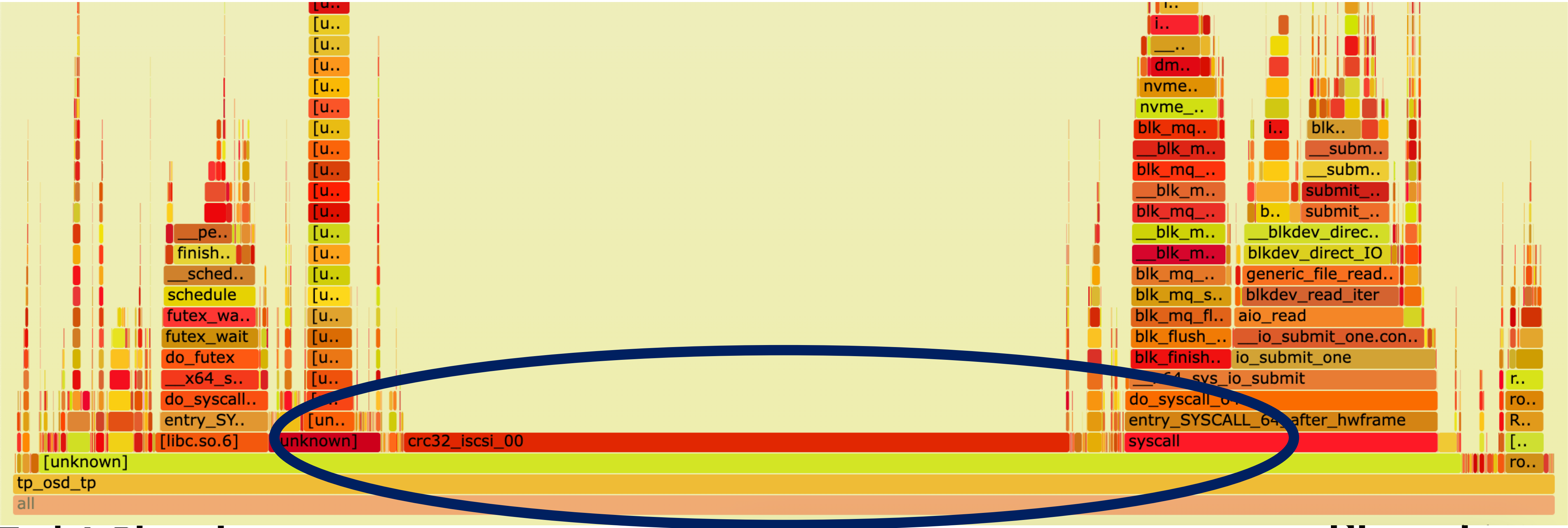


TechAtBloomberg.com

Bloomberg
Engineering

Here Today, Gone Tomorrow

- OSD thread pool was spending **43%** of it's time calculating CRCs over a 30-second period!



TechAtBloomberg.com

Bloomberg

Engineering

Here Today, Gone Tomorrow

- Now the kicker: Let's look at this *same* process just minutes later:

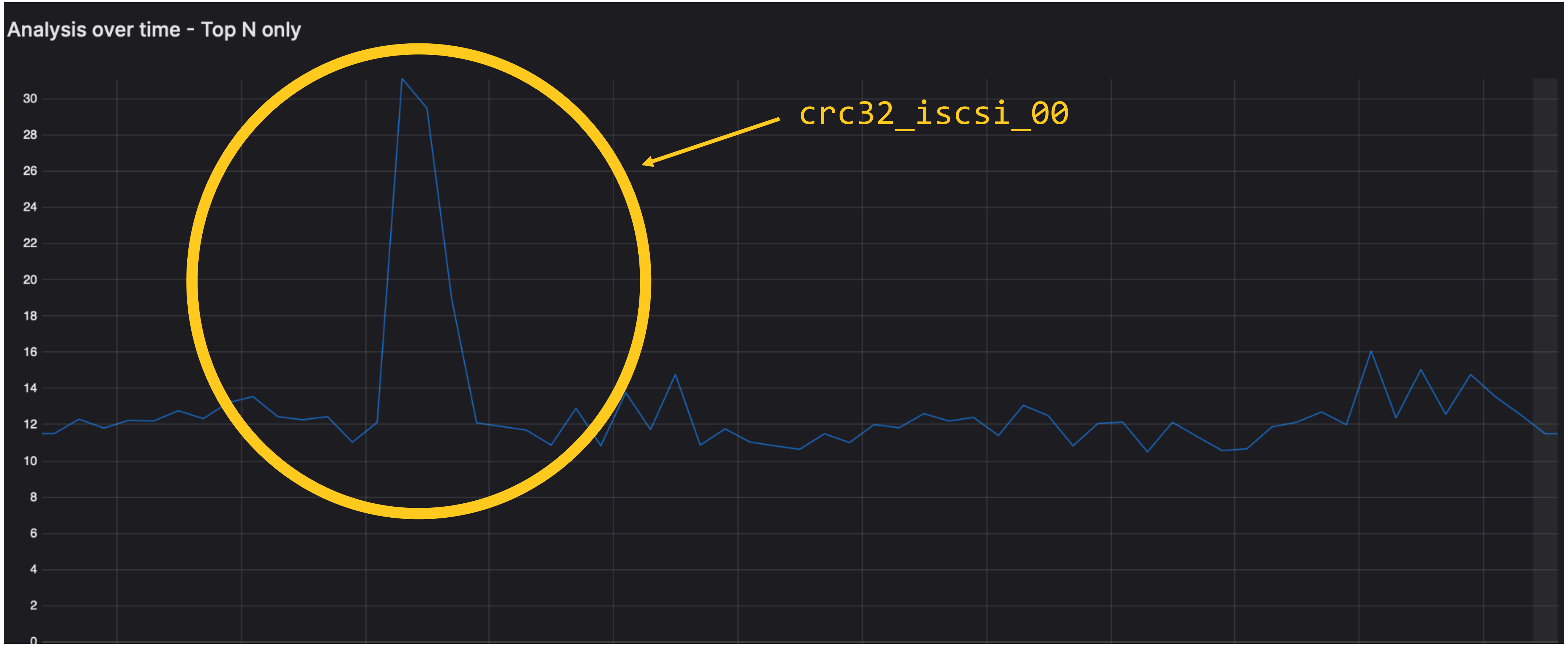
- `perf top -p 66111`

```
Samples: 8K of event 'cycles', 4000 Hz, Event count (approx.): 2577042790 lost: 0/0 drop: 0/0
Overhead  Shared Object          Symbol
   3.90%  libtcmalloc.so.4.5.9   [.] operator new[]
   3.58%  [kernel]               [k] copy_user_generic_string
   3.28%  ceph-osd               [.] crc32_iscsi_00
   2.10%  [kernel]               [k] nft_do_chain
   1.10%  libtcmalloc.so.4.5.9   [.] operator delete[]
   1.09%  ceph-osd               [.] rocksdb::InlineSkipList<rocksdb::MemTableRep::KeyComparator const&>::R
   0.91%  libc.so.6              [.] pthread_mutex_lock
   0.86%  libtcmalloc.so.4.5.9   [.] aligned_alloc
   0.82%  [kernel]               [k] native_read_msr
```

TechAtBloomberg.com

Bloomberg
Engineering

Here Today, Gone Tomorrow



TechAtBloomberg.com

Bloomberg

Engineering

Here Today, Gone Tomorrow

- ISA-L has multiple CRC32 implementations – ceph uses `crc32_iscsi_00`

- `crc32_iscsi_00`: Uses CPU's native `crc32` instructions
- `crc32_iscsi_01`: Uses `pclmulqdq` to vectorize folding of the message buffer

Here Today, Gone Tomorrow

- ISA-L has multiple CRC32 implementations – ceph uses `crc32_iscsi_00`

- `crc32_iscsi_00`: Uses CPU's native `crc32` instructions
- `crc32_iscsi_01`: Uses `pclmulqdq` to vectorize folding of the message buffer

CPU	`crc32_iscsi_00`	`crc32_iscsi_01`
Vendor A, Server Gen 2 uArch	10241 MB/s	13500 MB/s (+31%)
Vendor A, Server Gen 3 uArch	21645 MB/s	21469 MB/s (wash)
Vendor B, Personal Laptop	14691 MB/s	21084 MB/s (+43%)
Vendor B, Low Power/Edge	2887 MB/s	3664 MB/s (+26%)

- `pclmulqdq` version also benefits from a smaller look-up table and hence pollutes L1D$ less

Here Today, Gone Tomorrow

- Do not copy ceph.conf changes from online unless you understand why and what they do!
- Adapt `osd_op_thread_suicide_timeout` to HDD/SSD use cases, set SSD case to something less than 30s
- Is it time to change the default CRC32 implementation?
- It is important to have time-series- based telemetry to identify issues that come and go

Thank You!

We're hiring!
https://www.bloomberg.com/careers

Bloomberg Engineering

TechAtBloomberg.com