SUSE
We adapt. You succeed.

Performance Analysis with ceph 雲儲存性能分析

Alex Lau 劉俊賢

Software Consultant 研發工程師 / 顧問

(AvengerMoJo) alau@suse.com

Agenda 議程

SES5 is base on Luminous – The Why? 為何分析性能？

Ceph performance – The How? 如何分析性能？

Ceph analysis – The What? 分析結果是怎樣？

The Future? 大家有什麼展望？

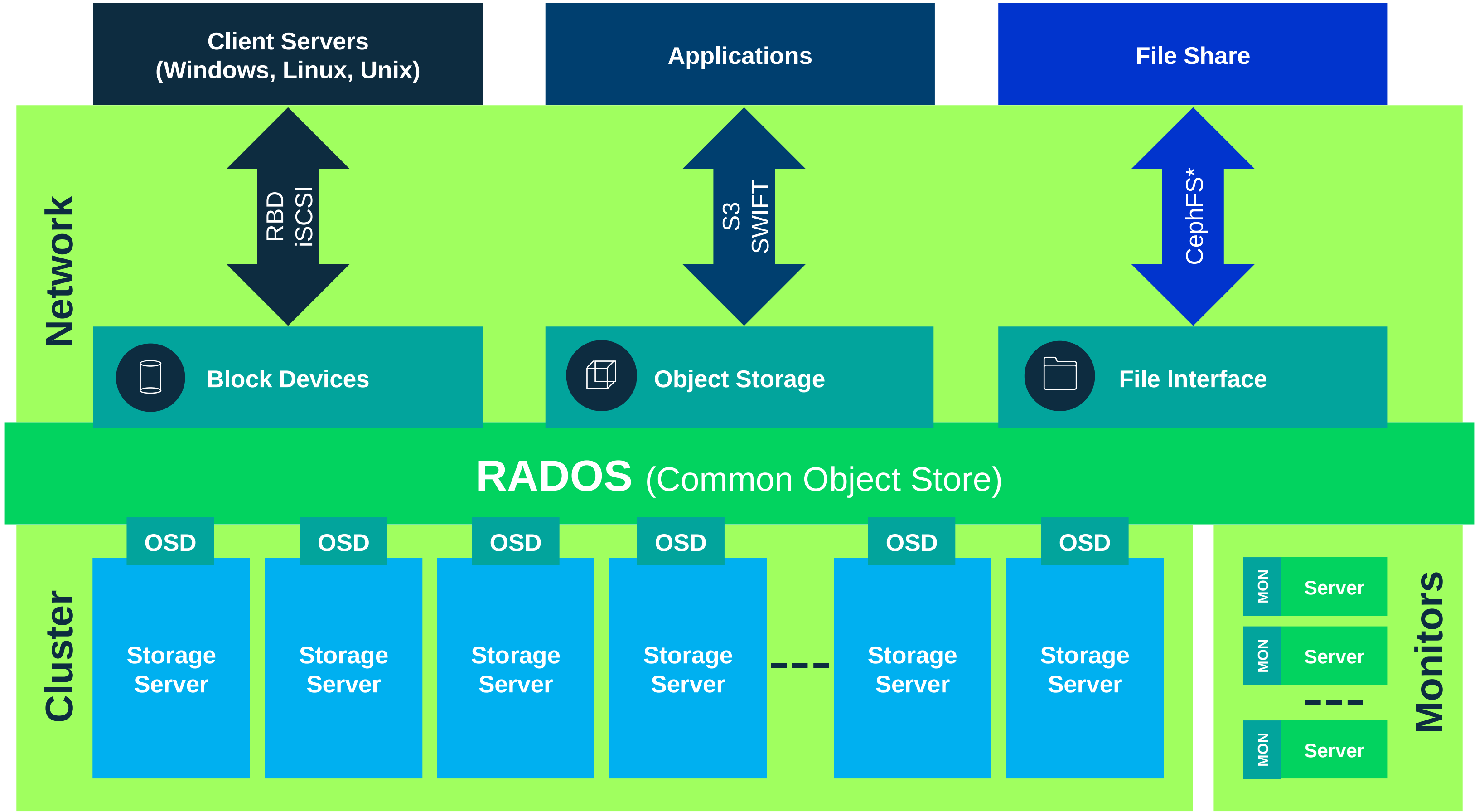
SES5 is base on Luminous – The Why?
為何分析性能？

SUSE Enterprise Storage 5 base on Ceph Luminous

- Support x86-64 and AArch64
- Easy to use WebUI openATTIC 3.x
- Simple DeepSea Cluster orchestration
- BlueStore ready with data compression
- Cephfs and NFS-Ganesha ready

STORAGE AWARDS THE STORRIES XII WINNER
Cloud Enabler of the Year
NETWORK computing AWARDS 2015 WINNER
PRODUCT OF THE YEAR
INNOVATIONSPREIS-IT FINALIST 2016 mittelstand
STORAGE / NETWORK
NETWORK computing AWARDS 2015 WINNER
STORAGE PRODUCT OF THE YEAR
STORAGE AWARDS THE STORRIES XII WINNER
Storage Magazine Editors Choice
STORAGE AWARDS THE STORRIES XIII WINNER
Availability Platform of the Year
NETWORK computing AWARDS 2016 WINNER
PRODUCT OF THE YEAR

Ceph Info 基本資料



Code Developers

782

Core	Regular	Casual
22	53	705

Total downloads

160,015,454

Unique downloads

21,264,047

SUSE CERN CISCO intel FUJITSU SanDisk redhat CANONICAL

SUSE Enterprise Storage 其實做了什麼？

基於 Ceph

2013: SUSE 加入 Ceph 社區

2015.01 : SUSE Enterprise Storage 1.0

2015.11: SUSE Enterprise Storage 2.0

2016.01: SUSE Enterprise Storage 2.1

2016.07: SUSE Enterprise Storage 3.0

2016: SUSE 收购 IT-Novum （主要開發存儲管理工具 openATTIC）

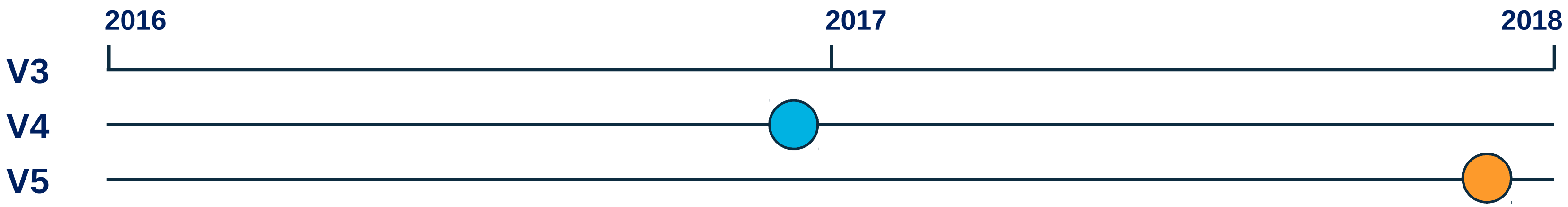
2016.09: SUSE 合併 HPE 軟體

2016.11: SUSE Enterprise Storage 4.0

2017.09: SUSE Enterprise Storage 5.0 (Beta Now)

SUSE Enterprise Storage

Roadmap



SUSE Enterprise Storage 4

Built On
- Ceph Jewel release
- SLES 12 SP 2 (Server)

Manageability
- SES openATTIC management
- Initial Salt integration DeepSea

Interoperability
- AArch64
- CephFS (production use cases)
- NFS Ganesha (Tech Preview)
- NFS access to S3 buckets (Tech Preview)
- CIFS Samba (Tech Preview)
- RDMA/Infiniband (Tech Preview)

Availability
- Multisite object replication
- Asynchronous block mirroring

SUSE Enterprise Storage 5*

Built On
- Ceph Luminous release
- SLES 12 SP 3 (Server)

Manageability
- SES openATTIC management phase 2
- SUSE Manager integration

Interoperability
- NFS Ganesha
- NFS access to S3 buckets

Availability
- Asynchronous block mirroring
- Erasure coded block pool

Efficiency
- BlueStore back-end
- Data compression

SUSE 和 Ceph 社區的關系

2013 年建立 Ceph 開發團隊 , 正式加入 Ceph 的代碼支持

SUSE 是 Ceph 理事會 8 大理事會員之一

SUSE CERN CISCO intel FUJITSU SanDisk redhat CANONICAL

一直以來代碼貢獻頭 3 名，上次 v12 Luminous 發報第 2 多，單個人第 1 Ricardo

1. 10648 Ricardo Dias rdias@suse.com
2. 6422 Sage Weil sweil@redhat.com

有收到 Inktank 和 Mirantis 開發人員

加大研發投入 : 2016 年中收購存儲管理工具 : it-novum (openATTIC)

Customer needs! 客戶的需要！

What’s our cluster IOPS?

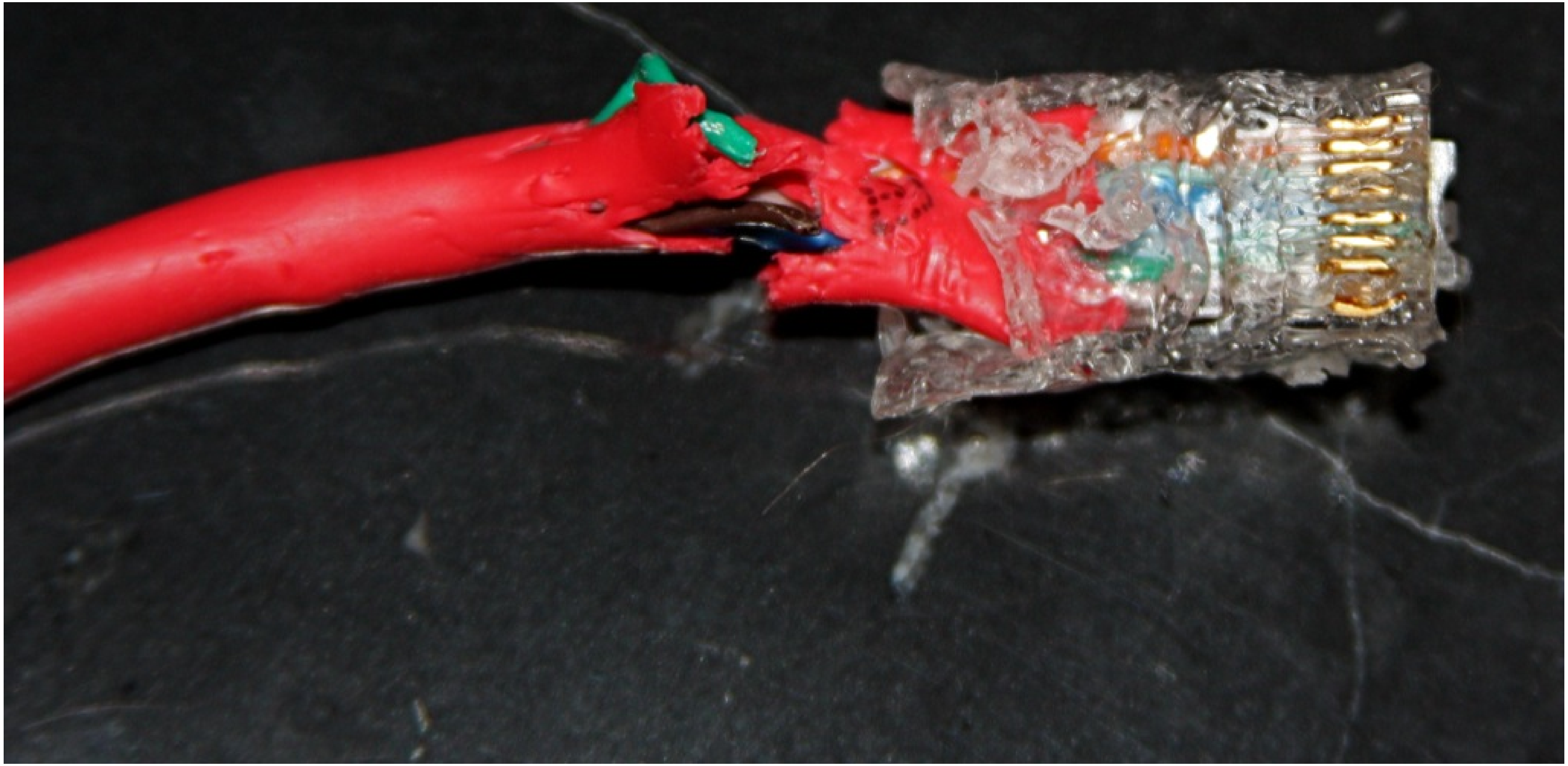
- How high do we need?
- How much do we want to pay?
- Are we counting 4K Random Write?
- Do we need Erasure Coding?
- Can we have the number on BlueStore vs FileStore?
- What about cephfs over nfs?
- ……

我們的存儲叢集能跑多少 IOPS?

- 我們需要多高的 IOPS?
- 我們想花多少錢？
- 我們只需要看 4K 隨機寫？
- 我們有糾刪碼的需要嗎？
- 我們能看看 BlueStore 對比 FileStore 嗎？
- 我們能看看 cephfs 上的 NFS 能怎樣？
- …….

System Engineer needs! 售前售後工程師的需要！

Putting them here is already show how much we care!



The DevOps Needs! 開發運維的需要！

Can we have some base line number?

Can we have tools to re-do performance analysis?

BTW we need all that to our live cluster!

能否告訴正常時叢集該跑多快？

能否每天再測試報告叢集現在的性能？

不是分開測而是實際運作中的叢集性能哦！

The Developers Needs! 開發者的需要 !

How much is BlueStore vs Filestore 4K, 16K, 64K, 1M, 4M write seq, random, read write performance in hdd vs ssd vs nvme pure and mixed pool, when an OSD is down, and the recovery process is started and meanwhile tracking how much memory needed when the capacity are 50% full without stopping the client keep writing to the cluster….

In short … very complicated analysis

…. 不寫了 .. 很多很多要求就是 .

p.s. 大家有硬體提供我們測試請找我，會後我請你… ☒

The Community Project Need! 社區的性能有關項目！

Ceph Brag

- It require to use CBT
  - https://github.com/ceph/cbt
- Setting up the CBT environment could be complicated
- It require another set of multiple nodes setup
- It is intrusive test that can’t be operate in a live cluster
- http://ceph.com/performance/

Ceph Brag

- CBT 是現在測試的標準
  - https://github.com/ceph/cbt
- 設定 CBT 環境並不簡單
- 需要多台機器進行測試
- 影響到內容不能向運作中的系統做測試
- 每週的性能有關會
  - http://ceph.com/performance/

The Goal

- Discover hardware, driver issue from time to time to check for degrade.
- Intrusive / Non-intrusive performance testing to do before and after the cluster being setup
- Able to perform to live and production cluster without stopping operation
- Low dependence, flexible, open source, and able to share eventually and align wth Ceph Brag
- Dynamic using different tools to do monitoring and tracing but able to orchestrated by a central admin that fit the container and cloud story

- 可以不定時發現硬體或驅動出現問題
- 在叢集設定前可以具破壞性的測試但在運作中的叢集可以用一樣的工具去做同類但不具破壞性的做法
- 測試可以在正在運作中的系統上進行而不影響一般運作
- 不用依賴太多三方面的需要，可以靈活使用，開源，最終可以和 Ceph Brag 共享一樣的發報方法
- 可以使用不同的管理和跟蹤工具，同時可以利用中央管理工具預備可以和容器和雲管理整合

Ceph Performance – The How?
如何分析性能？

Hardware Testing

Network:

```
   ping –c1 –q –W1
```

Specific for Jumbo Frame

```
   ping –Mdo –s8972 –c1 –q –W1
```

Bandwidth:

```
   iperf3 –fm –A0 –t10 –c<server ip> -p<port>
```

Harddisk:

Read:

```
   hdparm –t <osd_partition>
```

Write:

```
   dd if=/dev/zero of=`mount <osd_data_partition>/test/
conv=fdatasync bs=4K count=10000
```

Cluster Testing

OSD Bench (1G write with 4M block default)

```
ceph tell osd.0 bench
```

RADOS Bench

```
rados –p <pool> <time> <write,seq,rand> -t <thread> --no-cleanup
```

RBD bench

```
rbd –p <pool> bench-write <image> --io-size <e.g 4096>
--io-threads <1,4,16 etc> --io-total<total size e.g.209715200>
--io-pattern <rand | seq>
```

New version has io-type

```
rbd -p <pool> bench --io-type read rbd_test --io-size 4096
 --io-threads 1 --io-total 209715200 --io-pattern rand
```

Client Testing

FIO can be use against any of the following:

```
fio --ioengine=libaio --iodepth=32 --direct=1 --rw=write
--bs=4K --size=200MB --filename=/mnt/200M_data --numjobs=1
--name=cephfs_seqwrite_4K --output=cephfs_write_seq_4K.out
```

RBD Block

--ioengine can be either librbd or mount using libaio as default

ISCSI export

we can use it in windows or other OS that support iscsi

CephFS mount

```
mount.cephfs mon1,mon2,mon3:/ /mnt/
```

NFS/CIFS mount

```
mount.nfs nfs-server-ip:/ /mnt/
```

Lttng Testing

Kernel Tracing make easy!

```
lttng create –o .
lttng enable-channel --num-subbuf 16 --subbuf-size 8M  -k c0
lttng enable-event --kernel --all
lttng enable-event --syscall -a -k -c c0
lttng start
<do something>
lttng stop
lttng destroy
```

Mixing with Salt

It is already in next version of DeepSea

Network:

```
    salt-run net.ping exclude=<non-cluster-ip>
    salt-run net.jumbo_ping exclude=<non-cluster-ip>
```

Bandwidth:

```
    salt-run net.iperf cluster=ceph output=full
    salt-run net.iperf exclude=<non-cluster-ip>
```

Experiment

https://github.com/AvengerMoJo/Ceph-Saltstack

Since some of those test could be intrusive to OSD it is not in DeepSea yet.

```
salt “*” ceph_sles.disk_info
salt “*” ceph_slse.bench_disk /dev/sdb /dev/sdc /dev/sde
```

(dd write to a mount point a partition can mount to)

Lttng ust user space tracing

Client-side tracing only with librbd

compile ceph librbd with using `-finstrument-functions`

compile ceph with `-DWITH_LTTNG=ON`

```
export LD_PRELOAD=/usr/lib64/liblttng-ust-cyg-profile.so
```

Calling rbd to do something < write >

Stop tracing and collect the result for reporting

Can use Salt to do a multiple nodes tracing and collect all the report for you

Collecting Reporting with Salt

Calling salt to start lttng in every nodes of the cluster

```
salt-run lttng.run
    cmd="[fio --ioengine=libaio --iodepth=256 --direct=1
        --rw=write --bs=4K --size=20MB
        --filename=/mnt/cephfs/20M_data --numjobs=1
        –name=write_4K --output=/tmp/cephfs/write_seq_4K.out]"
    cmd_server=salt-master
```

Then collecting all the result back to the master.

Ceph Analysis – The What?
分析結是怎樣？

Example we use in the following hardware from SUSE Enterprise Storage Partners

HPE - OEM

- Apollo – Apollo 4200 * 3
  - osd with 18 hdd 2 ssd
- Proliant – DL380/DL360 * 3
  - mon

Network Ping

Regular Ping:

`ping –c1 –q –W1` **(rtt higher then 1ms maybe something wrong)**

```
--- 192.168.128.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.298/0.298/0.298/0.000 ms
```

Jumbo Frame:

`ping –Mdo –s8972 –c1 –q –W1` **(double normal ping)**

```
--- 192.168.128.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.649/0.649/0.649/0.000 ms
```

Network Bandwidth

Bandwidth: iperf3 –fm –A0 –t10 –c<server ip> -p<port>

```
Connecting to host 192.168.128.1, port 5201
[  4] local 192.168.128.77 port 41008 connected to 192.168.128.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   119 MBytes  1002 Mbits/sec    0    368 KBytes
[  4]   1.00-2.00   sec   118 MBytes   990 Mbits/sec    0    402 KBytes
[  4]   2.00-3.00   sec   118 MBytes   992 Mbits/sec    0    420 KBytes
[  4]   3.00-4.00   sec   118 MBytes   991 Mbits/sec    0    420 KBytes
[  4]   4.00-5.00   sec   118 MBytes   991 Mbits/sec    0    420 KBytes
[  4]   5.00-6.00   sec   118 MBytes   991 Mbits/sec    0    420 KBytes
[  4]   6.00-7.00   sec   118 MBytes   991 Mbits/sec    0    420 KBytes
[  4]   7.00-8.00   sec   118 MBytes   992 Mbits/sec    0    420 KBytes
[  4]   8.00-9.00   sec   118 MBytes   991 Mbits/sec    0    420 KBytes
[  4]   9.00-10.00  sec   118 MBytes   991 Mbits/sec    0    420 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.16 GBytes   992 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.15 GBytes   991 Mbits/sec                  receiver
```

DeepSea Network Test

```
salt-run net.ping exclude=<non-cluster-ip>
   Succeeded: 8 addresses from 7 minions average rtt 0.15 ms
salt-run net.jumbo_ping exclude=<non-cluster-ip>
   Succeeded: 8 addresses from 7 minions average rtt 0.26 ms
```

```
salt-run net.iperf exclude=192.168.128.9 cluster=ceph output=full
192.168.128.1:                    192.168.128.4:
    8644.0 Mbits/sec                  9588.56 Mbits/sec
192.168.128.2:                    192.168.128.5:
    10360.0 Mbits/sec                 10187.0 Mbits/sec
192.168.128.3:                    192.168.128.6:
    9336.0 Mbits/sec                  10465.0 Mbits/sec
```

Disk Direct Read Write

Read: `hdparm –t`

```
HDD:     Timing buffered disk reads: 618 MB in  3.01 seconds = 205.58 MB/sec

SSD:     Timing buffered disk reads: 1510 MB in  3.00 seconds = 503.25 MB/sec
```

Write: Direct dd right zero into disk mount point

```
/usr/bin/dd if=/dev/zero of=/var/lib/ceph/osd/ceph-0/test conv=fdatasync
bs=4K count=10000
    10000+0 records in
    10000+0 records out
HDD: 40960000 bytes (41 MB, 39 MiB) copied, 0.257385 s, 159 MB/s
SSD: 40960000 bytes (41 MB, 39 MiB) copied, 0.117944 s, 347 MB/s
```

With RAID Card Cache Enable

```
HDD: 40960000 bytes (41 MB, 39 MiB) copied, 0.038911 s, 1.1 GB/s
```

OSD Bench (without external communication)

OSD Bench (1G write with 4M block default)

```
 ceph tell osd.<num> bench
SSD
{
    "bytes_written": 1073741824, (1G)
    "blocksize": 4194304, (4M)
    "bytes_per_sec": 260063627 (248M)
}
RAID 0 HDD with cache
{
    "bytes_written": 1073741824,
    "blocksize": 4194304,
    "bytes_per_sec": 464233957 (442M)
}
```

RADOS Bench: rados –p hdd3 bench 20 write -t 32 –b 4096 --no-cleanup

```
Maintaining 32 concurrent writes of 4096 bytes to objects of size 4096 for up
to 20 seconds or 0 objects
2017-07-11 13:59:02.450884 min lat: 0.000861743 max lat: 0.193705 avg
lat: 0.0033328
  sec Cur ops   started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)
   20       9    191944    191935   37.4835   36.7539  0.00325612   0.0033328
Total time run:         20.008286
Total writes made:      191944
Write size:             4096
Object size:            4096
Bandwidth (MB/sec):     37.4735
Stddev Bandwidth:       2.88217
Max bandwidth (MB/sec): 44.332
Min bandwidth (MB/sec): 32.9258
Average IOPS:           9593
Stddev IOPS:            737
Max IOPS:               11349
Min IOPS:               8429
Average Latency(s):     0.00333329
Stddev Latency(s):      0.00611436
Max latency(s):         0.193705
Min latency(s):         0.000861743
```

Byte Size = 4M

```
Total time run:         20.186976
Total writes made:      4985
Write size:             4194304
Object size:            4194304
Bandwidth (MB/sec):     987.766
Stddev Bandwidth:       178.349
Max bandwidth (MB/sec): 1200
Min bandwidth (MB/sec): 596
Average IOPS:           246
Stddev IOPS:            44
Max IOPS:               300
Min IOPS:               149
Average Latency(s):     0.129235
Stddev Latency(s):      0.228253
Max latency(s):         6.50408
Min latency(s):         0.0230147
```

RBD Bench: rbd write with seq and rand

```
rbd –p ssd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern seq
    elapsed:     2  ops:    51200  ops/sec: 17947.54  bytes/sec: 73513129.85
rbd –p hdd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern seq
    elapsed:     2  ops:    51200  ops/sec: 22725.83  bytes/sec: 93085010.58
rbd –p ssd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern rand
    elapsed:    12  ops:    51200  ops/sec:  4010.68  bytes/sec: 16427739.46
rbd –p hdd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern rand
    elapsed:    13  ops:    51200  ops/sec:  3844.75  bytes/sec: 15748101.36
```

HDD with RAID card cache run faster in Sequence compare to Random

RBD Feature enable vs disable

To enable all the features in rbd: striping, object-map, fast-diff, deep-flatten, journaling

```
rbd –p ssd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern seq
    elapsed:     2  ops:    51200  ops/sec: 17143.89  bytes/sec: 70221393.28
rbd –p hdd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern seq
    elapsed:     2  ops:    51200  ops/sec: 22173.81  bytes/sec: 90823913.76
rbd –p ssd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern rand
    elapsed:    13  ops:    51200  ops/sec:  3711.78  bytes/sec: 15203449.44
rbd –p hdd bench-write rbd_test --io-size 4096 --io-threads 16
--io-total 209715200 --io-pattern rand
    elapsed:    33  ops:    51200  ops/sec:  1508.59  bytes/sec: 6179179.33
```

HDD with RAID cache run much worst in Random

RBD Block XFS mount hdd (cache) 4K IOPS benchmark

```
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=write --bs=4K --size=200MB
   --filename=/mnt/hdd/200M_data  --numjobs=1 –name=seq_write_4K
write: io=204800KB, bw=17667KB/s, iops=4416, runt= 11592msec
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=randwrite --bs=4K --size=200MB
   --filename=/mnt/hdd/200M_data --numjobs=1 --name=write_4K
write: io=204800KB, bw=104118KB/s, iops=26029, runt=  1967msec
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=read --bs=4K --size=200MB
   --filename=/mnt/hdd/200M_data --numjobs=1 --name=read_4K
read : io=204800KB, bw=71235KB/s, iops=17808, runt=  2875msec
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=randread --bs=4K --size=200MB
  --filename=/mnt/hdd/200M_data --numjobs=1 --name=read_4K
read : io=204800KB, bw=228317KB/s, iops=57079, runt=   897msec
```

RBD Block XFS mount ssd 4K IOPS benchmark

```
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=write --bs=4K --size=200MB
   --filename=/mnt/ssd/200M_data  --numjobs=1 –name=seq_write_4K
write: io=204800KB, bw=16394KB/s, iops=4098, runt= 12492msec
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=randwrite --bs=4K --size=200MB
   --filename=/mnt/ssd/200M_data --numjobs=1 --name=write_4K
write: io=204800KB, bw=104757KB/s, iops=26189, runt=  1955msec
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=read --bs=4K --size=200MB
   --filename=/mnt/ssd/200M_data --numjobs=1 --name=read_4K
read : io=204800KB, bw=90619KB/s, iops=22654, runt=  2260msec
fio --ioengine=libaio --iodepth=256 --direct=1 --rw=randread --bs=4K --size=200MB
  --filename=/mnt/ssd/200M_data --numjobs=1 --name=read_4K
read : io=204800KB, bw=302959KB/s, iops=75739, runt=   676msec
```

RBD Block XFS hdd (cache) 1M ThoughtPut benchmark

```
fio --ioengine=libaio --iodepth=16 --direct=1 --rw=write –bs=1M --size=4GB
   --filename=/mnt/hdd/4G_data  --numjobs=1 –name=seq_write_1M
write: io=4096.0MB, bw=1043.6MB/s, iops=1043, runt=  3925msec
fio --ioengine=libaio –iodepth=16 --direct=1 --rw=randwrite --bs=1M --size=4GB
   --filename=/mnt/hdd/4G_data --numjobs=1 --name=write_1M
write: io=4096.0MB, bw=1146.6MB/s, iops=1146, runt=  3574msec
fio --ioengine=libaio --iodepth=16 --direct=1 --rw=read --bs=1M –size=4GB
   --filename=/mnt/hdd/4G_data --numjobs=1 --name=read_1M
read : io=4096.0MB, bw=316790KB/s, iops=309, runt= 13240msec
fio --ioengine=libaio --iodepth=16 --direct=1 --rw=randread --bs=1M –size=4GB
  --filename=/mnt/hdd/4G_data --numjobs=1 --name=read_1M
read : io=4096.0MB, bw=183647KB/s, iops=179, runt= 22839msec
```

RBD Block XFS ssd 1M ThoughtPut benchmark

```
fio --ioengine=libaio --iodepth=16 --direct=1 --rw=write --bs=1M --size=4GB
   --filename=/mnt/ssd/4G_data  --numjobs=1 –name=seq_write_4G
write: io=4096.0MB, bw=540225KB/s, iops=527, runt=  7764msec
fio --ioengine=libaio –iodepth=16 --direct=1 --rw=randwrite --bs=1M --size=4GB
   --filename=/mnt/ssd/4G_data --numjobs=1 --name=write_4G
write: io=4096.0MB, bw=554142KB/s, iops=541, runt=  7569msec
fio --ioengine=libaio --iodepth=16 --direct=1 --rw=read --bs=1M --size=4GB
   --filename=/mnt/ssd/4G_data --numjobs=1 --name=read_4G
read : io=4096.0MB, bw=96998KB/s, iops=94, runt= 43241msec
fio --ioengine=libaio --iodepth=16 --direct=1 --rw=randread --bs=1M --size=4GB
  --filename=/mnt/ssd/4G_data --numjobs=1 --name=read_4G
read : io=4096.0MB, bw=113158KB/s, iops=110, runt= 37066msec
```

Lttng tracing regular RBD, Journaling enable RBD

Default:

```
          SEC          OPS  OPS/SEC        BYTES/SEC
elapsed:    0  ops:        1  ops/sec:   23.64  bytes/sec:   94.56
```

Journaling:

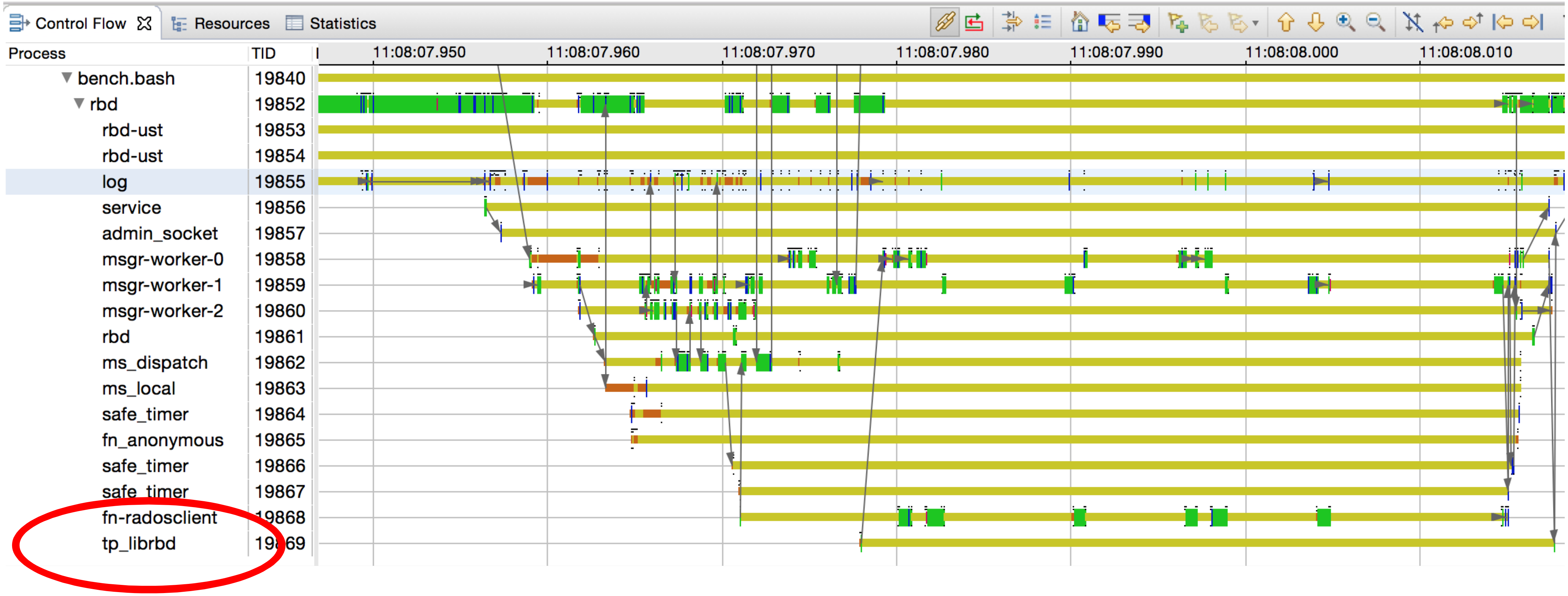
```
    bench  type write io_size 4 io_threads 1 bytes 4 pattern random
          SEC          OPS  OPS/SEC        BYTES/SEC
elapsed:    0  ops:        1  ops/sec:    9.54  bytes/sec:   38.14
```

Using Tracecompress

Eclipse base UI, easy to use, cross platform (java)

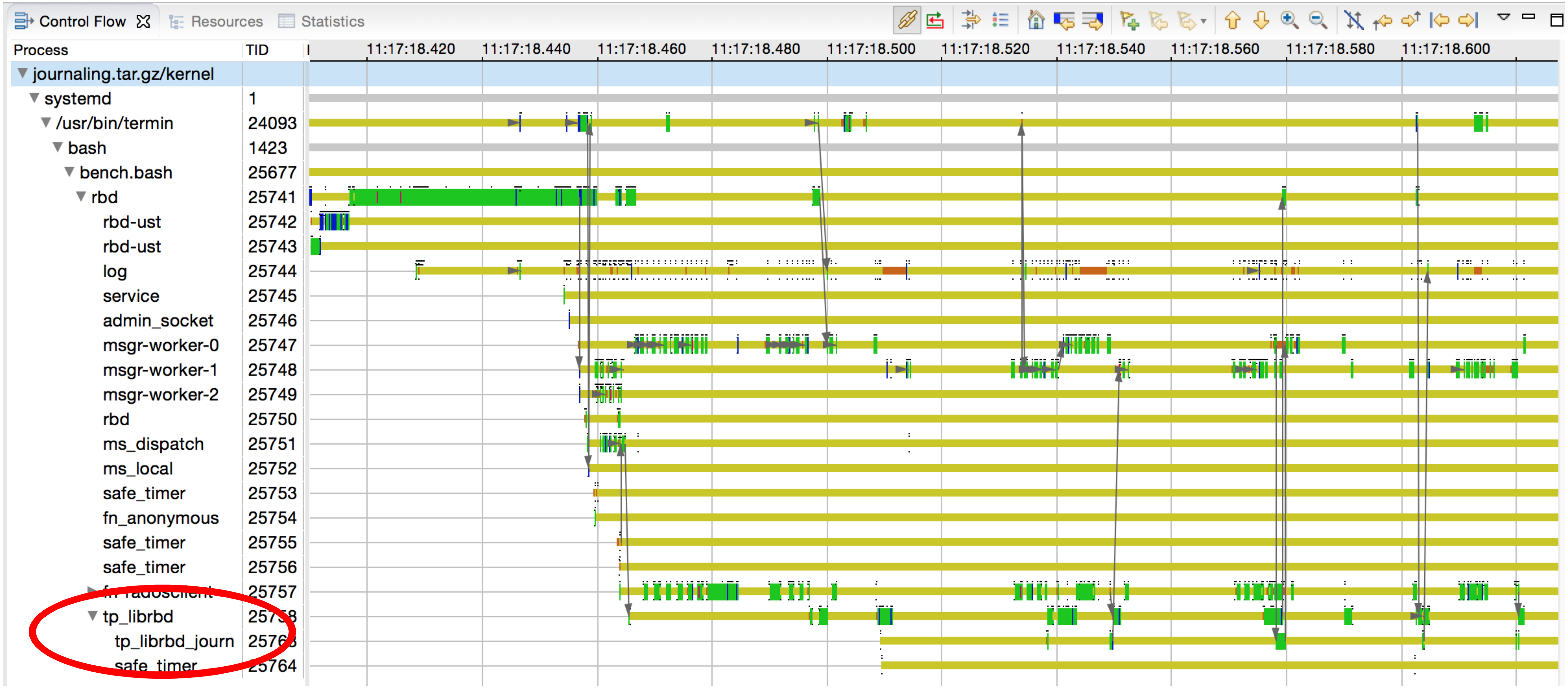
http://tracecompass.org/

Regular Layering RBD “tp_librbd”



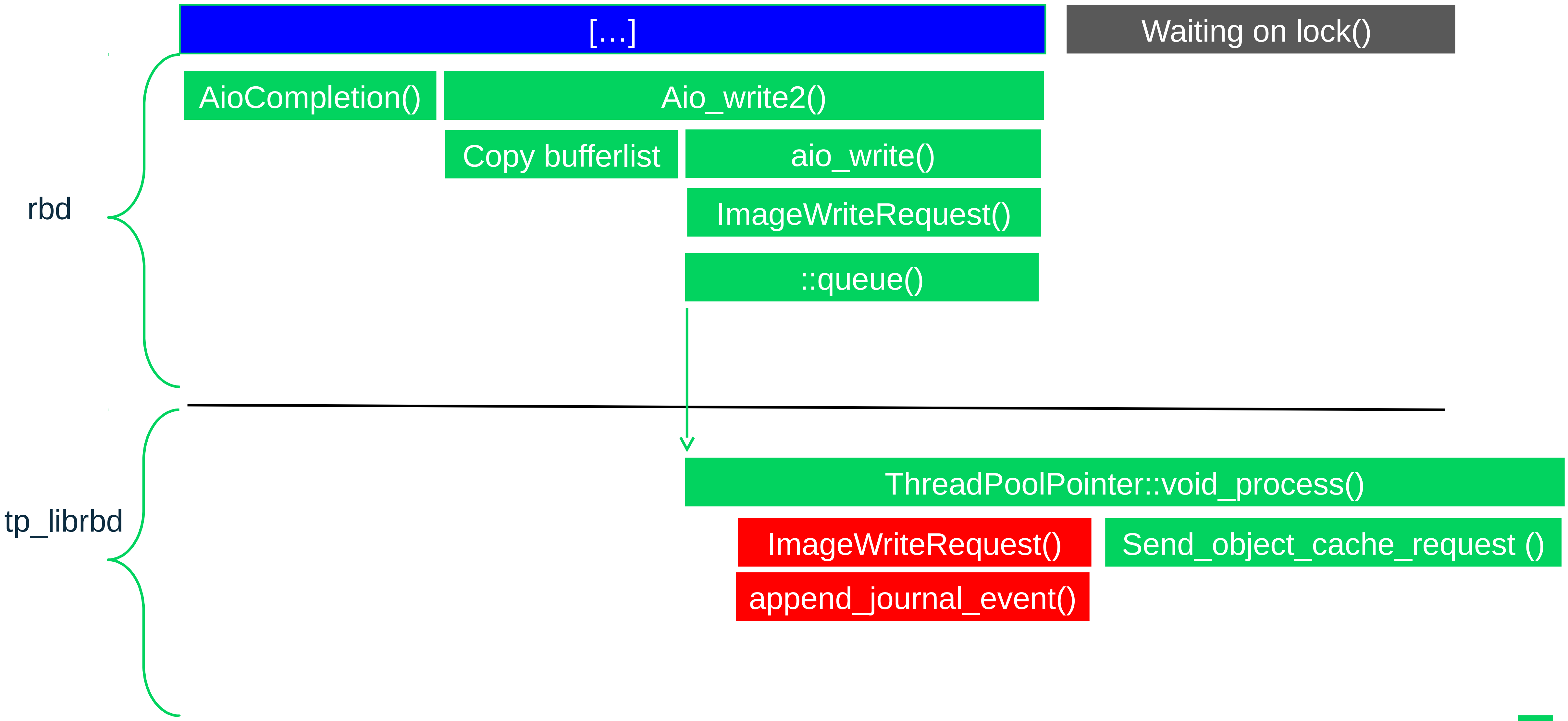
Process	TID
▼ bench.bash	19840
▼ rbd	19852
rbd-ust	19853
rbd-ust	19854
log	19855
service	19856
admin_socket	19857
msgr-worker-0	19858
msgr-worker-1	19859
msgr-worker-2	19860
rbd	19861
ms_dispatch	19862
ms_local	19863
safe_timer	19864
fn_anonymous	19865
safe_timer	19866
safe_timer	19867
fn-radosclient	19868
tp_librbd	19869

RBD with remote Journaling “tp_librbd_journ”

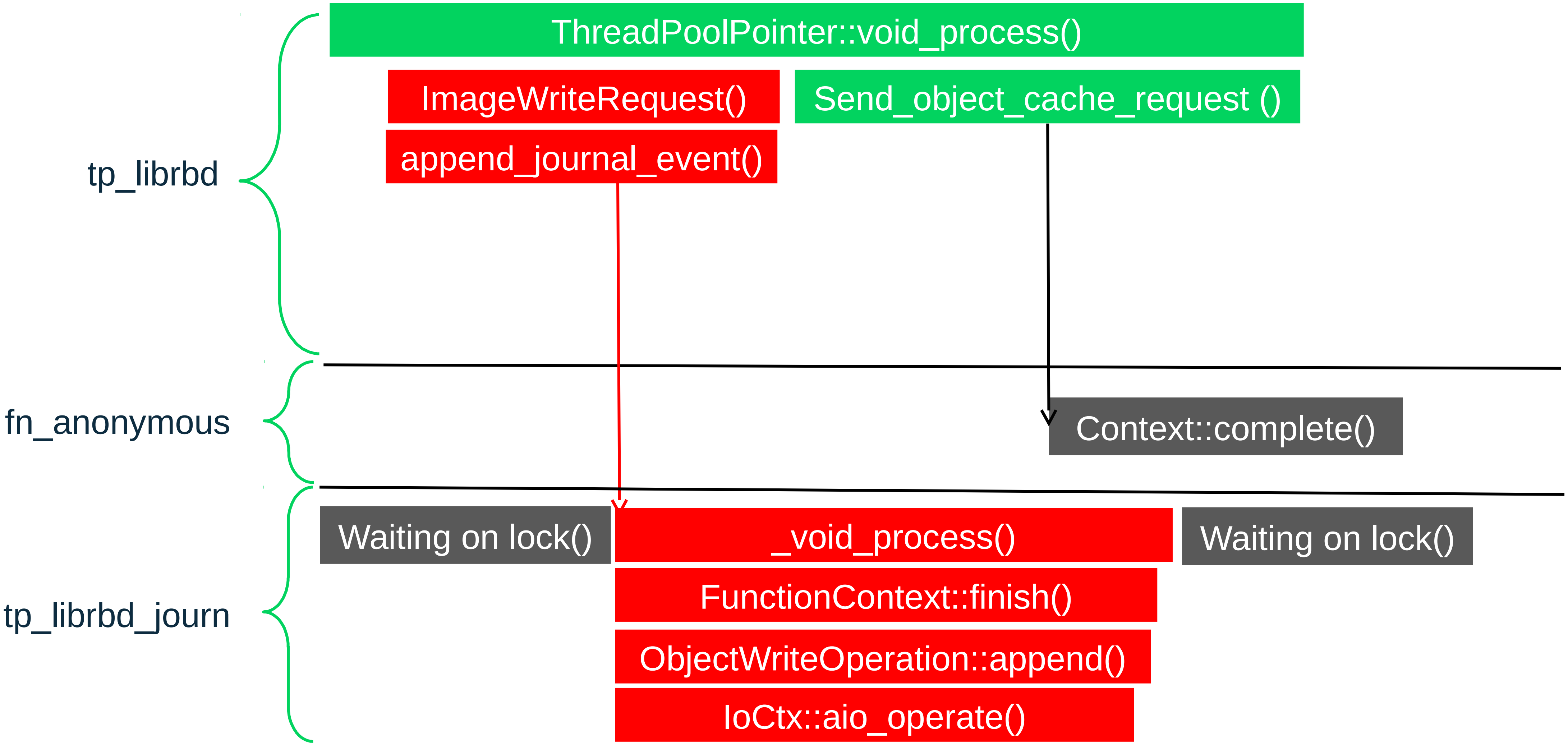


Process	TID
▼ journaling.tar.gz/kernel	
▼ systemd	1
▼ /usr/bin/termin	24093
▼ bash	1423
▼ bench.bash	25677
▼ rbd	25741
rbd-ust	25742
rbd-ust	25743
log	25744
service	25745
admin_socket	25746
msgr-worker-0	25747
msgr-worker-1	25748
msgr-worker-2	25749
rbd	25750
ms_dispatch	25751
ms_local	25752
safe_timer	25753
fn_anonymous	25754
safe_timer	25755
safe_timer	25756
▶ fn-radosclient	25757
▼ tp_librbd	25758
tp_librbd_journ	25763
safe_timer	25764

https://docs.google.com/drawings/d/1s9pRVygdPzBtQyYrvPueF8Cm6Xi1d4oy5gNMOjx2Wzo



https://docs.google.com/drawings/d/1s9pRVygdPzBtQyYrvPueF8Cm6Xi1d4oy5gNMOjx2Wzo



tp_librbd

ThreadPoolPointer::void_process()

ImageWriteRequest()

append_journal_event()

Send_object_cache_request ()

fn_anonymous

Context::complete()

tp_librbd_journ

Waiting on lock()

_void_process()

FunctionContext::finish()

ObjectWriteOperation::append()

IoCtx::aio_operate()

Waiting on lock()

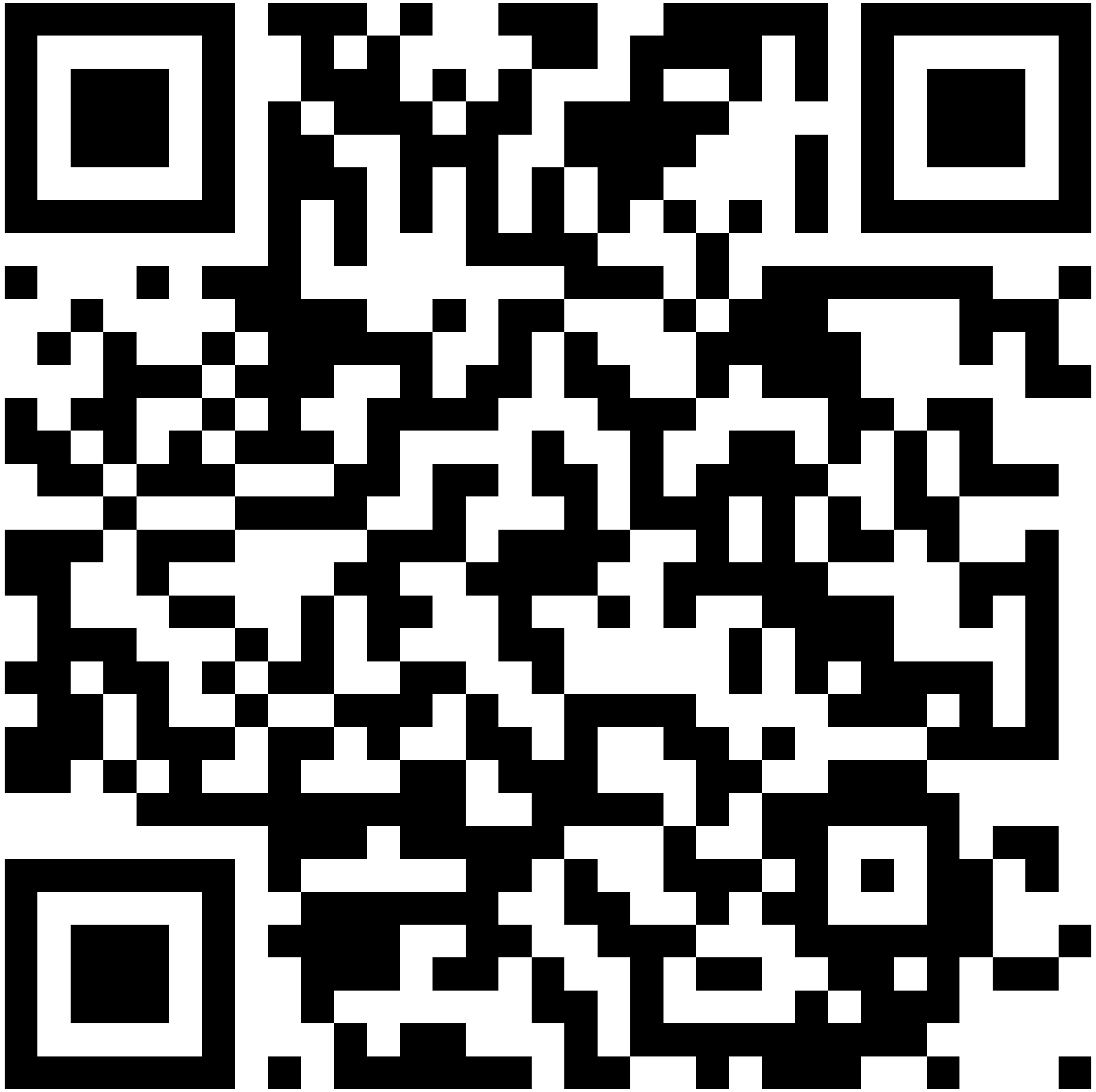
Let’s look at real trace

The Future?
以後的展望？

The Future

- Setup partner with production hardware in community that can run the non-intrusive test case in regular bases to compare with different ceph cluster.
- Include more client side multiple nodes tracing (container / cloud ready)
- Filestore and Bluestore with detail features on and off
- More in-depth ARM AArch64 ceph performance report
- Upstream is already doing a lot of recovery related to timing issue
- Chinese Translation related to performance analysis tools and method
- 提供服務給社區運作中的叢集非破壞式的測試方法 , 給大家參考和對比差異
- 提供更完整的多點跟蹤功能 (容器內 / 雲內)
- 針對 Filestore 和 Bluestore 做更深人的測試 , 觀察打開不同功能後的性能比較
- 再深人去了解 ARM64 運作 ceph 時的性能報告
- 上游正著手觀察 Recovery 有關的不同選項做分析
- 翻譯有關這麼多不同性能測試工具和方法的文檔

Questions and Answers



We adapt. You succeed.