Transforming PCIe-SSDs and HDDs with Infiniband into Scalable Enterprise Storage

Dieter Kasper
Fujitsu

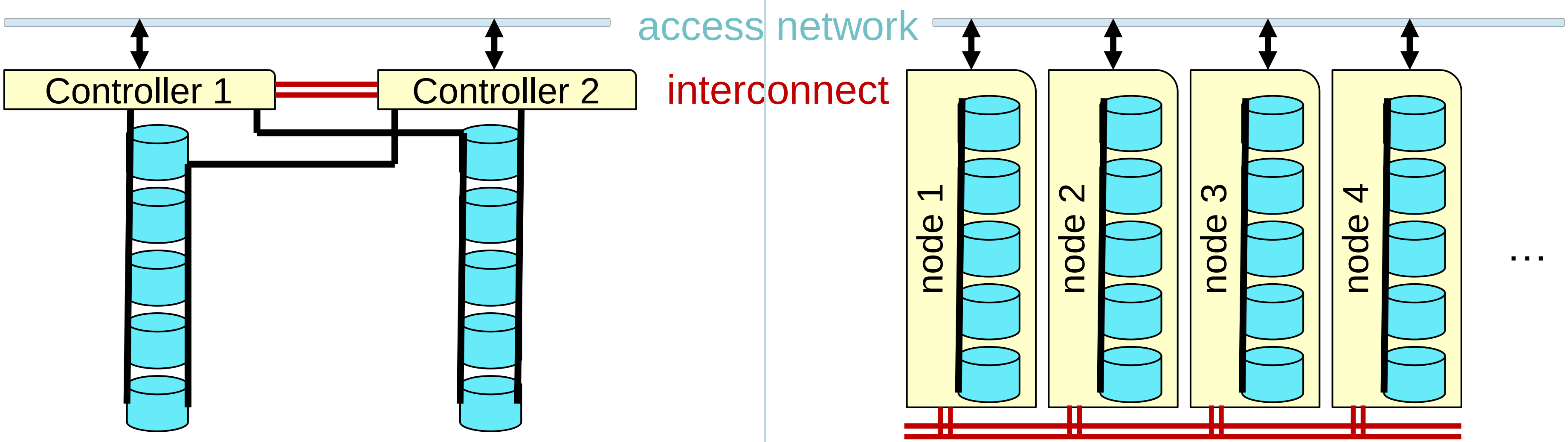
Agenda

- **Introduction**
- Hardware / Software layout
- Tools how to monitor
- Transformation test cases
- Conclusion

Challenges of a Storage Subsystem

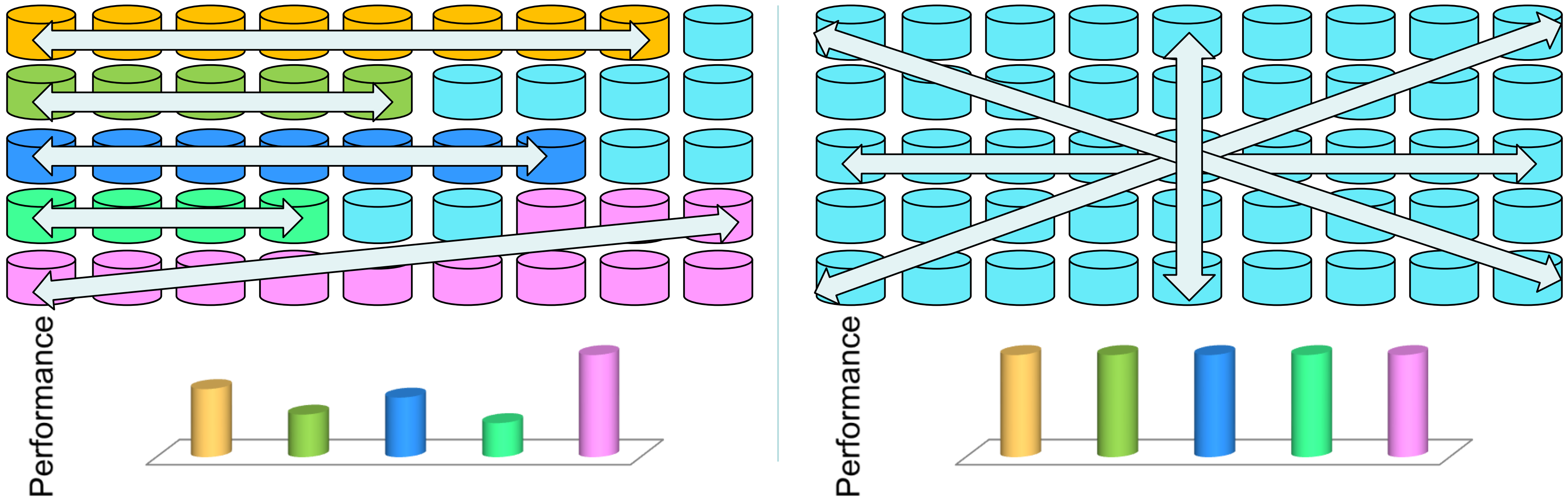
- **Transparency**
  - User has the impression of a single global File System / Storage Space
- **Scalable Performance, Elasticity**
  - No degradation of performance as the number of users or volume of data increases
  - Intelligent rebalancing on capacity enhancements
  - Offer same high performance for all volumes
- **Availability, Reliability and Consistency**
  - User can access the same file system / Block Storage from different locations at the same time
  - User can access the file system at any time
  - Highest MTTDL (mean time to data loss)
- **Fault tolerance**
  - System can identify and recover from failure
  - Lowest Degradation during rebuild time
  - Shortest Rebuild times
- **Manageability & ease of use**

Conventional vs. distributed model



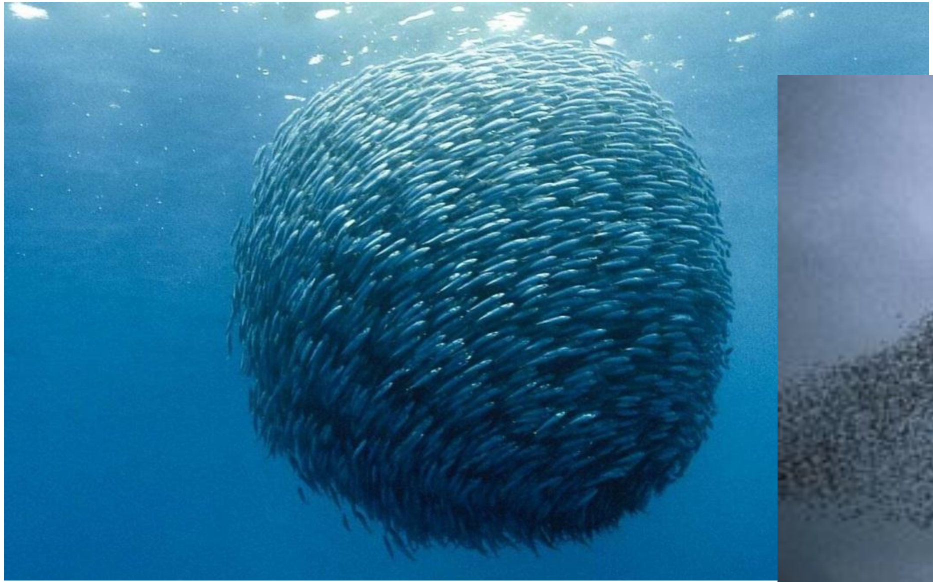
- Dual redundancy
  - 1 node/controller can fail
  - 2 drives can fail (RAID 6)
  - HA only inside a pair
- N-way resilience, e.g.
  - Multiple nodes can fail
  - Multiple drives can fail
  - Nodes coordinate replication, and recovery

Conventional vs. distributed model



- While traditional storage systems distribute Volumes across sub-sets of Spindles Scale-Out systems use algorithms to distribute Volumes across all/many Spindles and provide maximum utilization of all system resources
  - **Offer same high performance for all volumes and shortest rebuild times**

A model for dynamic “clouds” in nature



Swarm of birds or fishes
Source: wikipedia

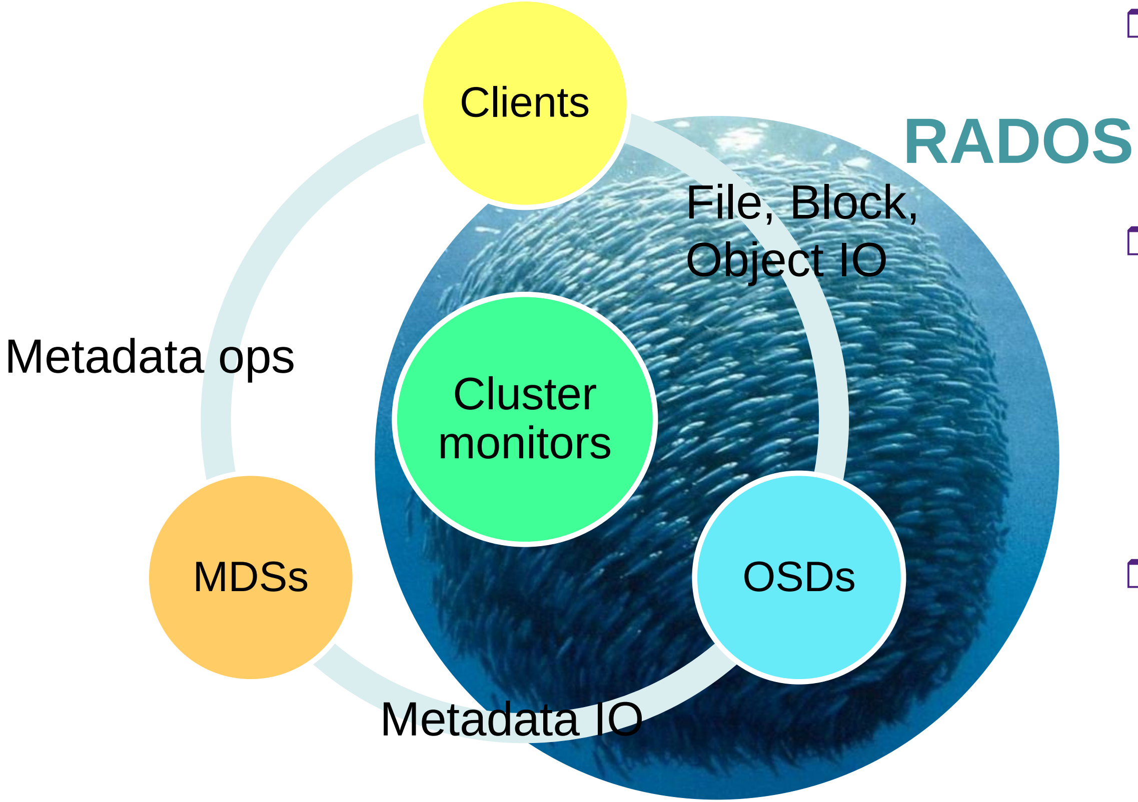
Distributed intelligence

- **Swarm intelligence** [Wikipedia]
  - (SI) is the collective behavior of decentralized, self-organized systems, natural or artificial.
- **Swarm behavior** [Wikipedia]
  - Swarm behavior, or swarming, is a collective behavior exhibited by animals of similar size (…) moving en masse or migrating in some direction.
  - From a more abstract point of view, swarm behavior is the collective motion of a large number of self-propelled entities.
  - From the perspective of the mathematical modeler, it is an (…) **behavior arising from simple rules that are followed by individuals and does not involve any central coordination.**

Ceph Key Design Goals

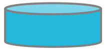
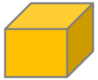
- The system is inherently dynamic:
  - Decouples data and metadata
  - Eliminates object list for naming and lookup by a hash-like distribution function – CRUSH (Controlled Replication Under Scalable Hashing)
  - Delegates responsibility of data migration, replication, failure detection and recovery to the OSD (Object Storage Daemon) cluster
- Node failures are the norm, rather than an exception
  - Changes in the storage cluster size (up to 10k nodes) cause automatic and fast failure recovery and rebalancing of data with no interruption
- The characters of workloads are constantly shifting over time
  - The Hierarchy is dynamically redistributed over 10s of MDSs (Meta Data Services) by Dynamic Subtree Partitioning with near-linear scalability
- The system is inevitably built incrementally
  - FS can be seamlessly expanded by simply adding storage nodes (OSDs)
  - Proactively migrates data to new devices -> balanced distribution of data
  - Utilizes all available disk bandwidth and avoids data hot spots

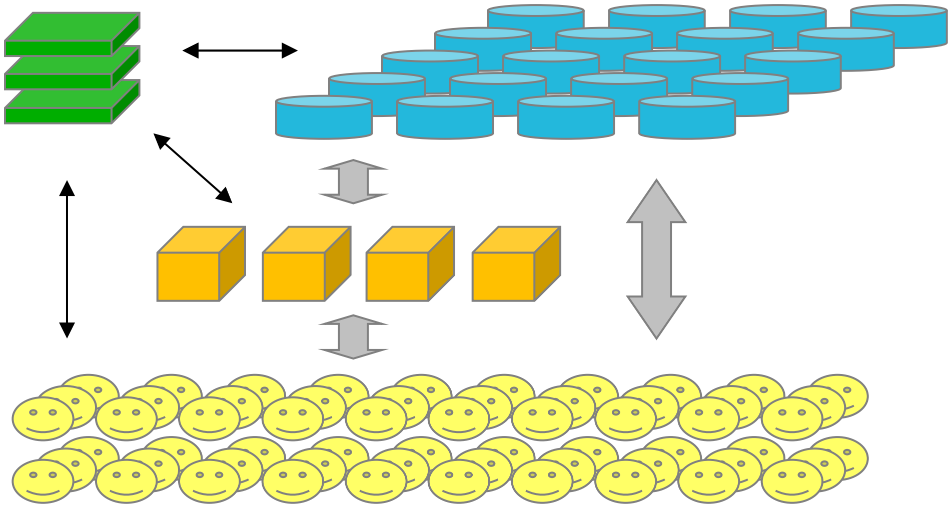
Architecture: Ceph + RADOS (1)



- Clients
  - Standard Interface to use the data (POSIX, Device, S3)
  - Transparent for Applications
- Metadata Server Cluster (MDSs)
  - Namespace Management
  - Metadata operations (open, stat, rename, …)
  - Ensure Security
- Object Storage Cluster (OSDs)
  - Stores all data and metadata
  - Organizes data into flexible-sized containers, called objects

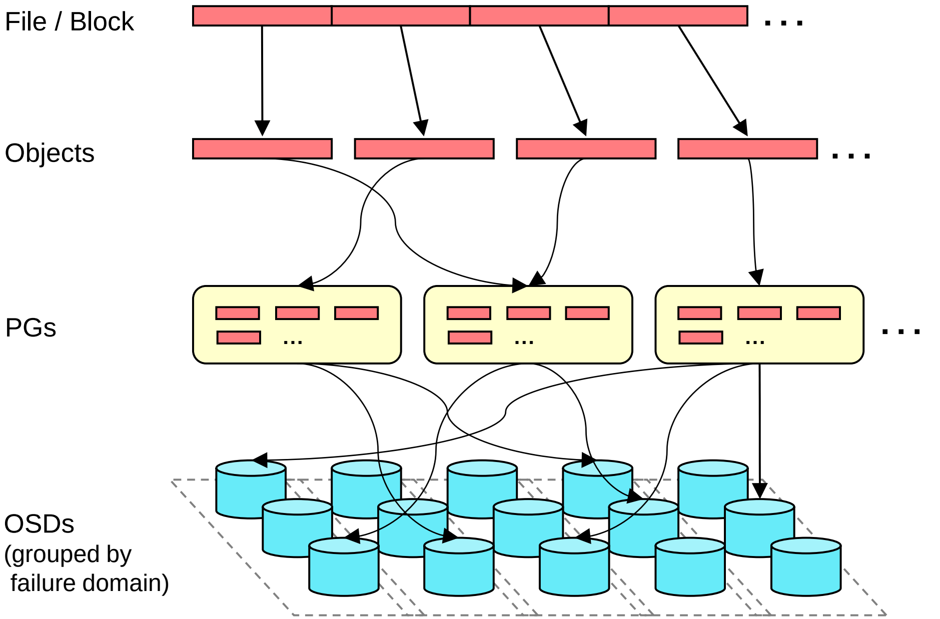
Architecture: Ceph + RADOS (2)

- MONs(monitors) 
  - 1s-10s, paxos
  - lightweight process
  - authentication, cluster membership, critical cluster state
- OSDs 
  - 1s-10,000s
- Ceph Clients 
  - Zillions
  - Smart, coordinate with peers
  - authenticate with monitors, talk directly to ceph-osds
- MDSs 
  - 1s-10s
  - Build POSIX file system on top of objects



Data placement with CRUSH

- Files/bdevs striped over objects
  - 4 MB objects by default
- Objects mapped to *placement groups (PGs)*
  - pgid = hash(object) & mask
- PGs mapped to sets of OSDs
  - crush(cluster, rule, pgid) = [osd2, osd3]
  - Pseudo-random, statistically uniform distribution
  - ~100 PGs per node



- **Fast:** O(log n) calculation, no lookups
- **Reliable:** replicas span failure domains
- **Stable:** adding/removing OSDs moves few PGs

- A deterministic pseudo-random hash like function that distributes data uniformly among OSDs
- Relies on compact cluster description for new storage target w/o consulting a central allocator

Unified Storage for Cloud based on Ceph – Architecture and Principles

App

App → Object

Host / VM → Virtual Disk

Client → Files & Dirs

Librados
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP

Ceph Object Gateway (RGW)
A bucket-based REST gateway, compatible with S3 and Swift

Ceph Block Device (RBD)
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver

Ceph File System (CephFS)
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE

Ceph Storage Cluster (RADOS)
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

The Ceph difference

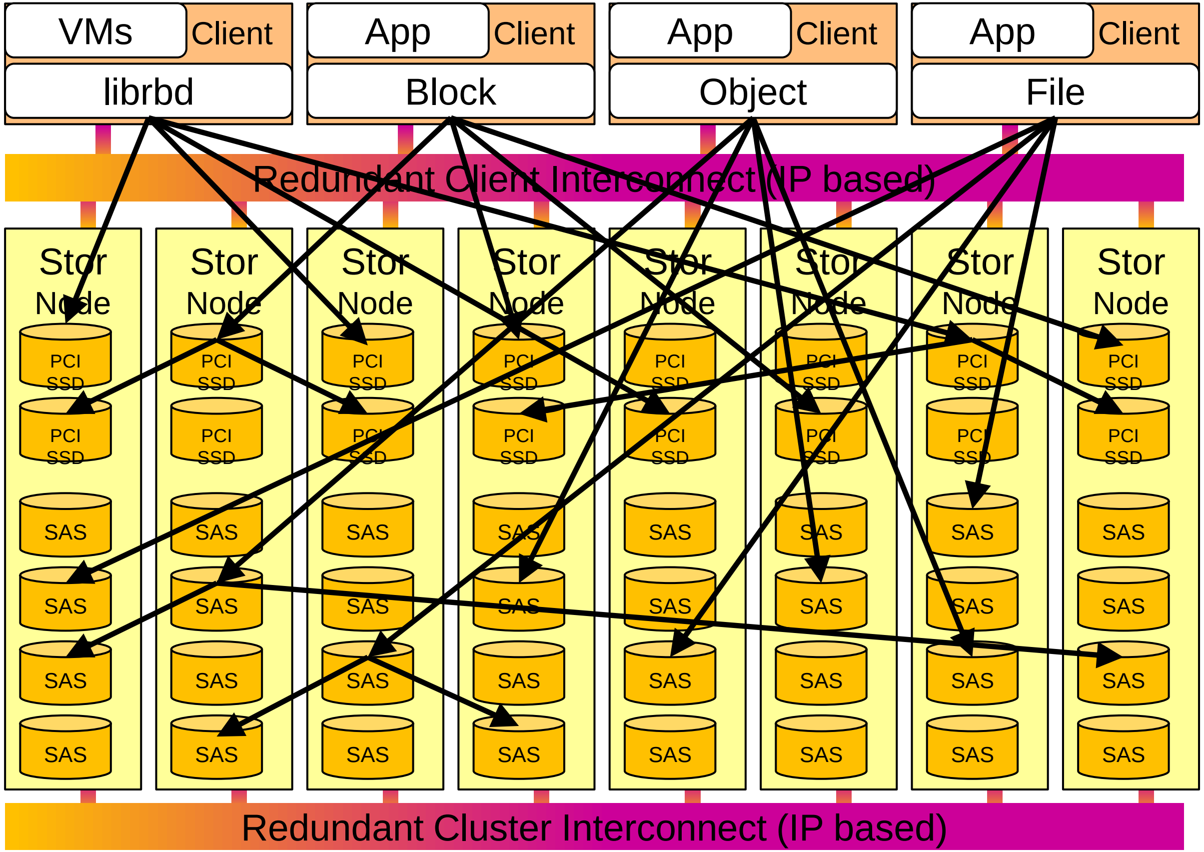
ceph

Ceph's CRUSH algorithm liberates storage clusters from the scalability and performance limitations imposed by centralized data table mapping. It replicates and re-balance data within the cluster dynamically - elminating this tedious task for administrators, while delivering high-performance and infinite scalability.

http://ceph.com/ceph-storage
http://www.inktank.com

Ceph is the most comprehensive implementation of Unified Storage

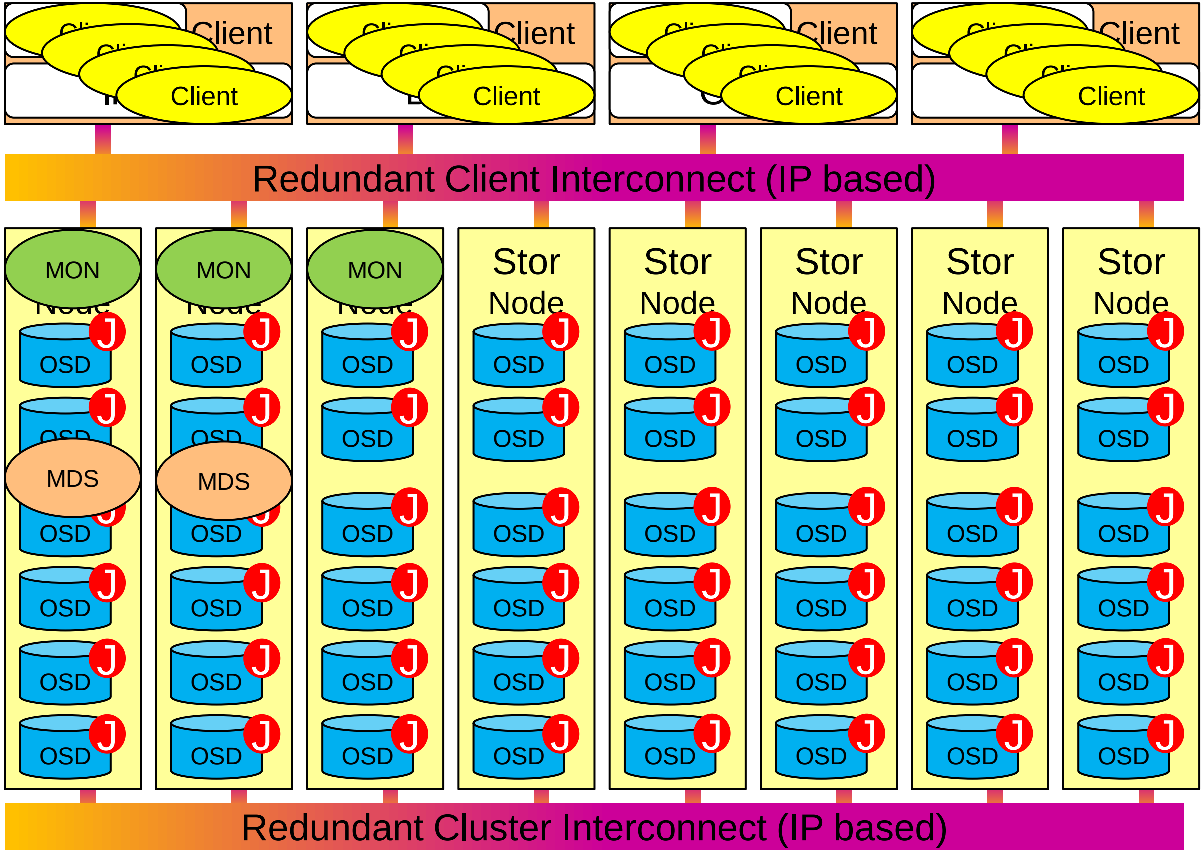
Ceph principles



Distributed Redundant Storage

- Intelligent data Distribution across all nodes and spindles = wide striping (64KB – 16MB)
- Redundancy with replica=2, 3 … 8
- Thin provisioning
- Fast distributed rebuild
- Availability, Fault tolerance
  - Disk, Node, Interconnect
  - Automatic rebuild
  - Distributed HotSpare Space
- Transparent Block, File access
- Reliability and Consistency
- Scalable Performance
- Pure PCIe-SSD for extreme Transaction processing

Ceph processes



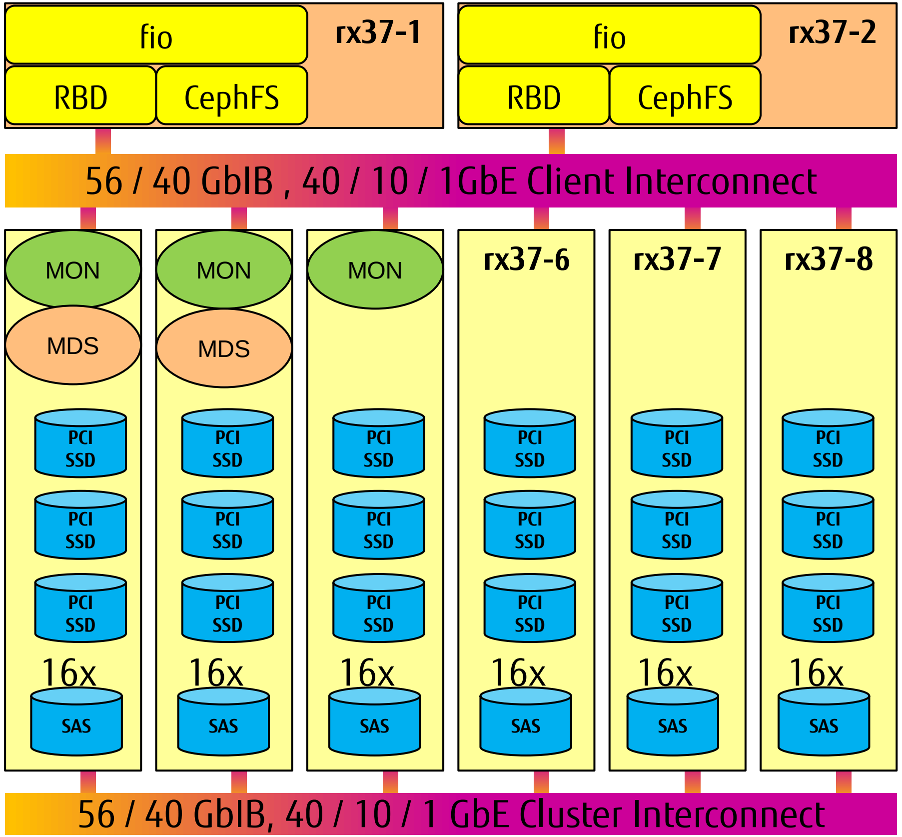
Distributed Redundant Storage

- Intelligent data Distribution across all nodes and spindles = wide striping (64KB – 16MB)
- Redundancy with replica=2, 3 … 8
- Thin provisioning
- Fast distributed rebuild
- Availability, Fault tolerance
  - Disk, Node, Interconnect
  - Automatic rebuild
  - Distributed HotSpare Space
- Transparent Block, File access
- Reliability and Consistency
- Scalable Performance
- Pure PCIe-SSD for extreme Transaction processing

Agenda

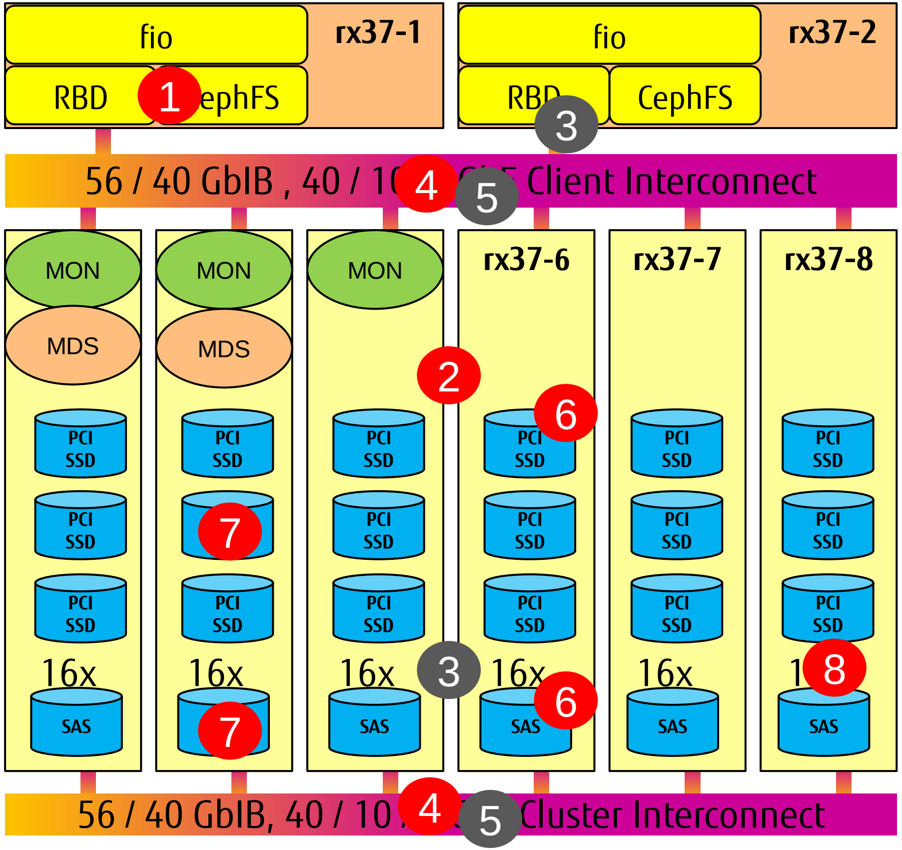
- Introduction
- **Hardware / Software layout**
- Tools how to monitor
- Transformation test cases
- Conclusion

Hardware test configuration



- rx37-[3-8]: Fujitsu 2U Server RX300
  - 2x Intel(R) Xeon(R) E5-2630 @ 2.30GHz
  - 128GB RAM
  - 2x 1GbE onboard
  - 2x 10GbE Intel 82599EB
  - 1x 40GbIB Intel TrueScale IBA7322 QDR InfiniBand HCA
  - 2x 56GbIB Mellanox MT27500 Family (configurable as 40GbE, too)
  - 3x Intel PCIe-SSD 910 Series 800GB
  - 16x SAS 6G 300GB HDD through
  - LSI MegaRAID SAS 2108 [Liberator]
- rx37-[12]: same as above, but
  - 1x Intel(R) Xeon(R) E5-2630 @ 2.30GHz
  - 64GB RAM
  - No SSDs, 2x SAS drives

Which parameter to change, tune



(1) Frontend interface: ceph.ko, rbd.ko, ceph-fuse, rbd-wrapper in user land

(2) OSD object size of data: 64k, 4m

(3) Block Device options for /dev/rbdX, /dev/sdY: scheduler, rq_affinity, rotational, read_ahead_kb

(4) Interconnect: 1 / 10 / 40 GbE, 40 / 56 GbIB CM/DG

(5) Network parameter

(6) Journal: RAM-Disk, SSD

(7) OSD File System: xfs, btrfs

(8) OSD disk type: SAS, SSD

Software test configuration

- CentOS 6.4
  - With vanilla Kernel 3.8.13
  - fio-2.0.13
  - ceph version 0.61.7 cuttlefish

```
 1GbE: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
 1GbE: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
10GbE: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
10GbE: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
40Gb:   ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520
56Gb:   ib1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520
56Gb:   ib2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520
40GbE: eth4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9216
40GbE: eth5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9216
```

```
# fdisk -l /dev/sdq
   Device Boot Start       End      Blocks   Id  System
/dev/sdq1              1     22754   182767616   83  Linux
/dev/sdq2          22754     24322    12591320    f  Ext'd
/dev/sdq5          22755     23277     4194304   83  Linux
/dev/sdq6          23277     23799     4194304   83  Linux
/dev/sdq7          23799     24322     4194304   83  Linux

#--- 3x Journals on each SSD
```

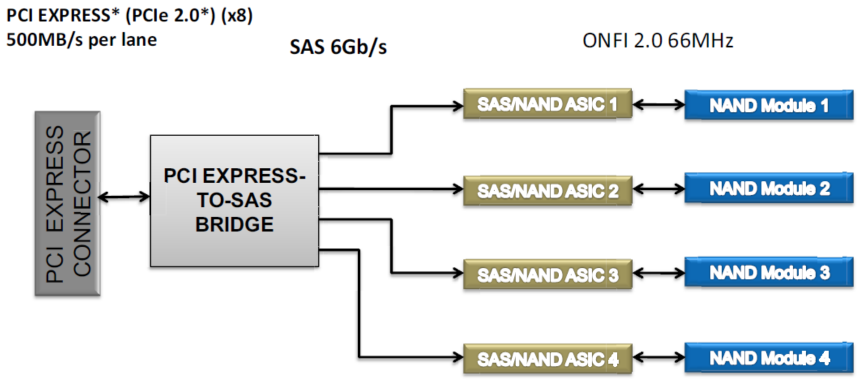
```
# lsscsi
[0:2:0:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sda
[0:2:1:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdb
[0:2:2:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdc
[0:2:3:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdd
[0:2:4:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sde
[0:2:5:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdf
[0:2:6:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdg
[0:2:7:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdh
[0:2:8:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdi
[0:2:9:0]    disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdj
[0:2:10:0]   disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdk
[0:2:11:0]   disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdl
[0:2:12:0]   disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdm
[0:2:13:0]   disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdn
[0:2:14:0]   disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdo
[0:2:15:0]   disk    LSI      RAID 5/6 SAS 6G  2.12  /dev/sdp
[1:0:0:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sdq
[1:0:1:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sdr
[1:0:2:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sds
[1:0:3:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sdt
[2:0:0:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sdu
[2:0:1:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sdv
[2:0:2:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sdw
[2:0:3:0]    disk    INTEL(R)  SSD 910 200GB  a411  /dev/sdx
[3:0:0:0]    disk    INTEL(R)  SSD 910 200GB  a40D  /dev/sdy
[3:0:1:0]    disk    INTEL(R)  SSD 910 200GB  a40D  /dev/sdz
[3:0:2:0]    disk    INTEL(R)  SSD 910 200GB  a40D  /dev/sdaa
[3:0:3:0]    disk    INTEL(R)  SSD 910 200GB  a40D  /dev/sdab
```

Agenda

- Introduction
- Hardware / Software layout
- **Tools how to monitor**
- Transformation test cases
- Conclusion

Intel PCIe-SSD 910 data sheet

- IOPS random rd/wr 4k: 180k/75k
  (queue depth 32 per NAND module)
- Bandwidth rd/wr 128k: 2/1 GB/s
  (queue depth 32 per NAND module)
- Latency rd 512 / wr 4k seq: <65µs
  (queue depth 1 per NAND module)



Recommended Settings in Linux:

- rq_affinity 1
- scheduler noop
- rotational 0
- read_ahead_kb 0

Intel © SSD Data Center Tool

https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=21099

```
| Supported Page List                     | 0 (0x00)    |
| Write Error Counter                     | 2 (0x02)    |
| Read Error Counter                      | 3 (0x03)    |
| Verify Error Counter                    | 5 (0x05)    |
| Non-medium Error Counter                | 6 (0x06)    |
| Temperature                             | 13 (0x0D)   |
| Manufacturing Date Information          | 14 (0x0E)   |
| Application Client Log                  | 15 (0x0F)   |
| Self Test Results                       | 16 (0x10)   |
| Solid State Media                       | 17 (0x11)   |
| Background Scan Medium Operation        | 21 (0x15)   |
| Protocol Specific Log Parameter         | 24 (0x18)   |
| Link Status                             | 26 (0x1A)   |
| SMART Status and Temperature Reading    | 47 (0x2F)   |
| Vendor Specific                         | 48 (0x30)   |
| Misc Data Counters                      | 55 (0x37)   |

# isdct -device 1 -drive 3 -log 0x11 | grep end
| Percentage used endurance indicator | 1 (0x01) |
# isdct -device 1 -drive 3 -log 0xD | grep Cel
| Temperature (Degress Celsius)           | 28 (0x1C)   |
| Reference Temperature (Degress Celsius) | 85 (0x55)   |
```

Infiniband


```
# # ibstatus | egrep 'Infi|stat|rate' | grep -v link_
Infiniband device 'mlx4_0' port 1 status:
        state:                  4: ACTIVE
        phys state:             5: LinkUp
        rate:                   56 Gb/sec (4X FDR)
Infiniband device 'mlx4_0' port 2 status:
        state:                  4: ACTIVE
        phys state:             5: LinkUp
        rate:                   56 Gb/sec (4X FDR)
Infiniband device 'qib0' port 1 status:
        state:                  4: ACTIVE
        phys state:             5: LinkUp
        rate:                   40 Gb/sec (4X QDR)
```

```
# ibhosts -C qib0 -P 1
Ca            : 0x00117500005a6aea ports 1 "rx37-8 qib0"
Ca            : 0x0011750000783984 ports 1 “rx37-7 qib0"
Ca            : 0x001175000078405e ports 1 "rx37-1 qib0"
Ca            : 0x00117500005a6ad2 ports 1 “rx37-3 qib0"
Ca            : 0x001175000077f6ec ports 1 "rx37-4 qib0"
Ca            : 0x001175000077740e ports 1 "rx37-5 qib0"
Ca            : 0x0011750000789c9e ports 1 "rx37-6 qib0"
Ca            : 0x00117500005a6a32 ports 1 "rx37-2 qib0“

# ibv_devinfo
# iblinkinfo –R
# perfquery -C qib0 -P 1
# ibdiagnet -p 1
```

```
# iblinkinfo
CA: rx37-8 mlx4_0:
     0x0002c90300218c81    14    1[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      3   35[  ] "MF0;switch-b79e58:SX6036/U1" ( )
     0x0002c90300218c82    15    2[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      3   33[  ] "MF0;switch-b79e58:SX6036/U1" ( )
CA: localhost mlx4_0:
     0x0002c90300218b91    12    1[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      3   29[  ] "MF0;switch-b79e58:SX6036/U1" ( )
     0x0002c90300218b92    13    2[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      3   31[  ] "MF0;switch-b79e58:SX6036/U1" ( )
CA: rx37-6 mlx4_0:
     0x0002c90300218dc1    17    1[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      3   25[  ] "MF0;switch-b79e58:SX6036/U1" ( )
     0x0002c90300218dc2    16    2[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      3   27[  ] "MF0;switch-b79e58:SX6036/U1" ( )
Switch: 0x0002c903008e8f00 MF0;switch-b79e58:SX6036/U1:
          3    1[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      1    1[  ] " HCA-1" ( )
          3    2[  ] ==(                 Down/ Polling)==>          [  ] "" ( )
          3    3[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      2    2[  ] " HCA-1" ( )
          3    4[  ] ==(                 Down/ Polling)==>          [  ] "" ( )
          3    5[  ] ==( 4X      14.0625 Gbps Active/  LinkUp)==>      4    1[  ] "rx37-1 mlx4_0" ( )
```

Agenda

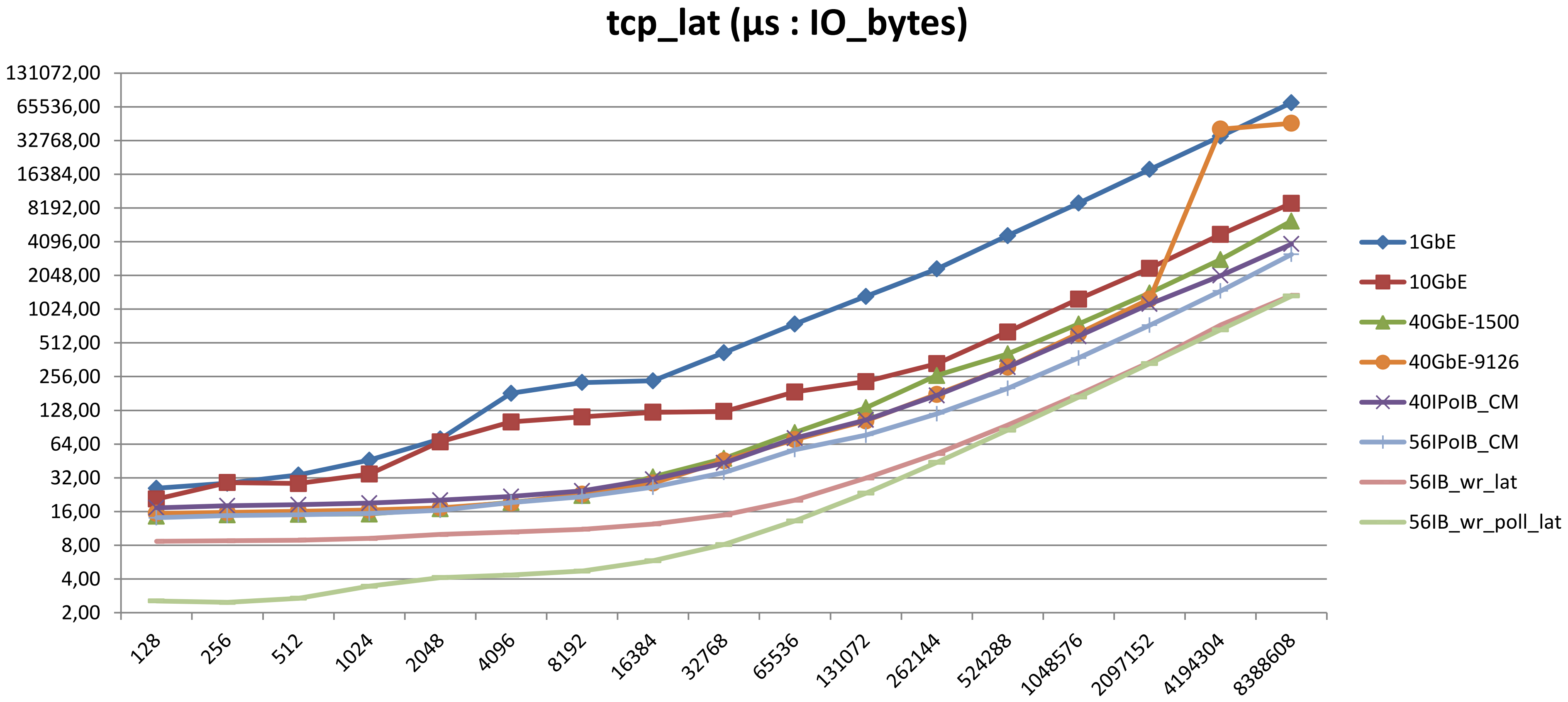
- Introduction
- Hardware / Software layout
- Tools how to monitor
- **Transformation test cases**
- Conclusion

Performance test tools

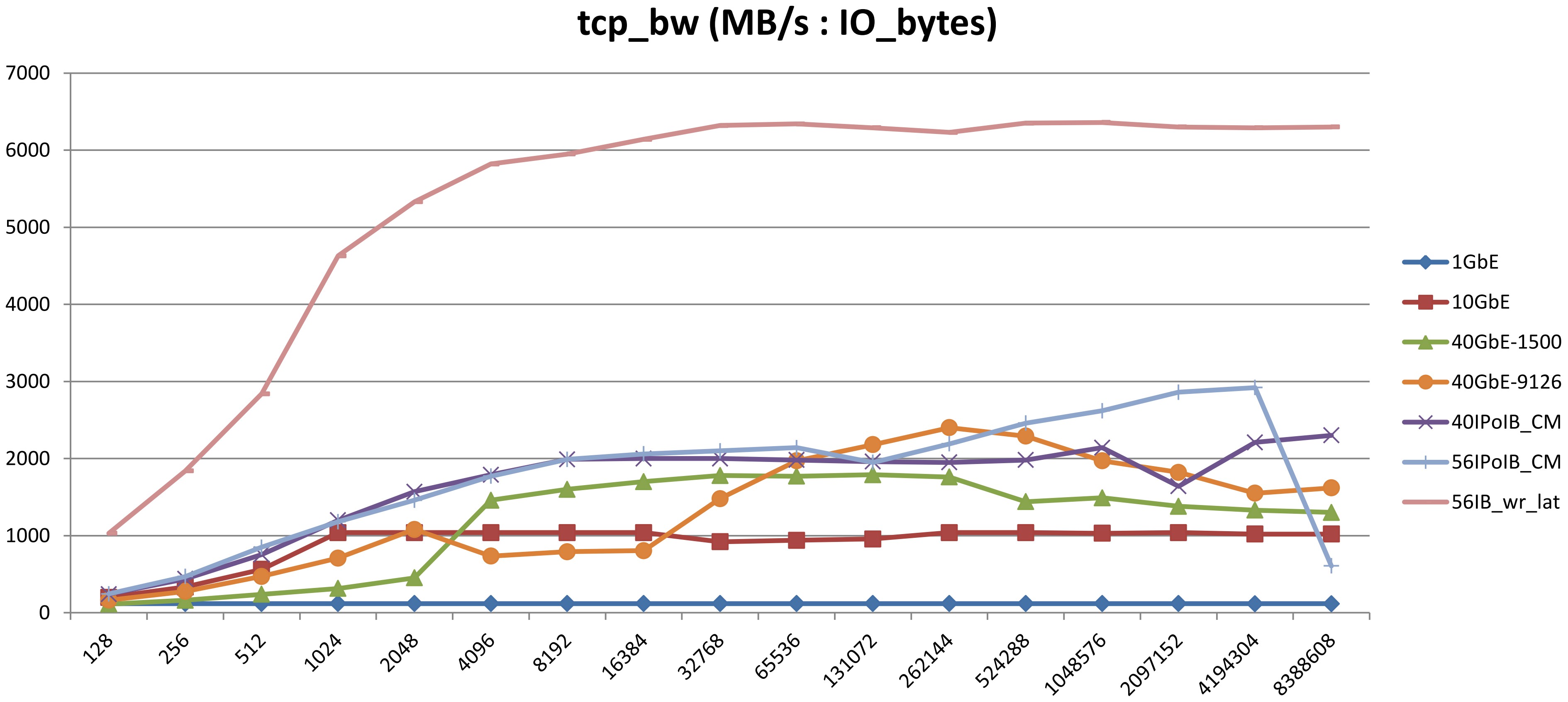
- qperf $IP_ADDR -t 10 -oo msg_size:1:8M:*2 –v tcp_lat | tcp_bw | rc_rdma_write_lat | rc_rdma_write_bw | rc_rdma_write_poll_lat

- fio --filename=$RBD|--directory=$MDIR --direct=1 --rw=$io --bs=$bs --size=10G --numjobs=$threads --runtime=60 --group_reporting --name=file1 --output=fio_${io}_${bs}_${threads}
  - RBD=/dev/rbdX, MDIR=/cephfs/fio-test-dir
  - io=write,randwrite,read,randread
  - bs=4k,8k,4m,8m
  - threads=1,64,128

qperf - latency



qperf - bandwidth

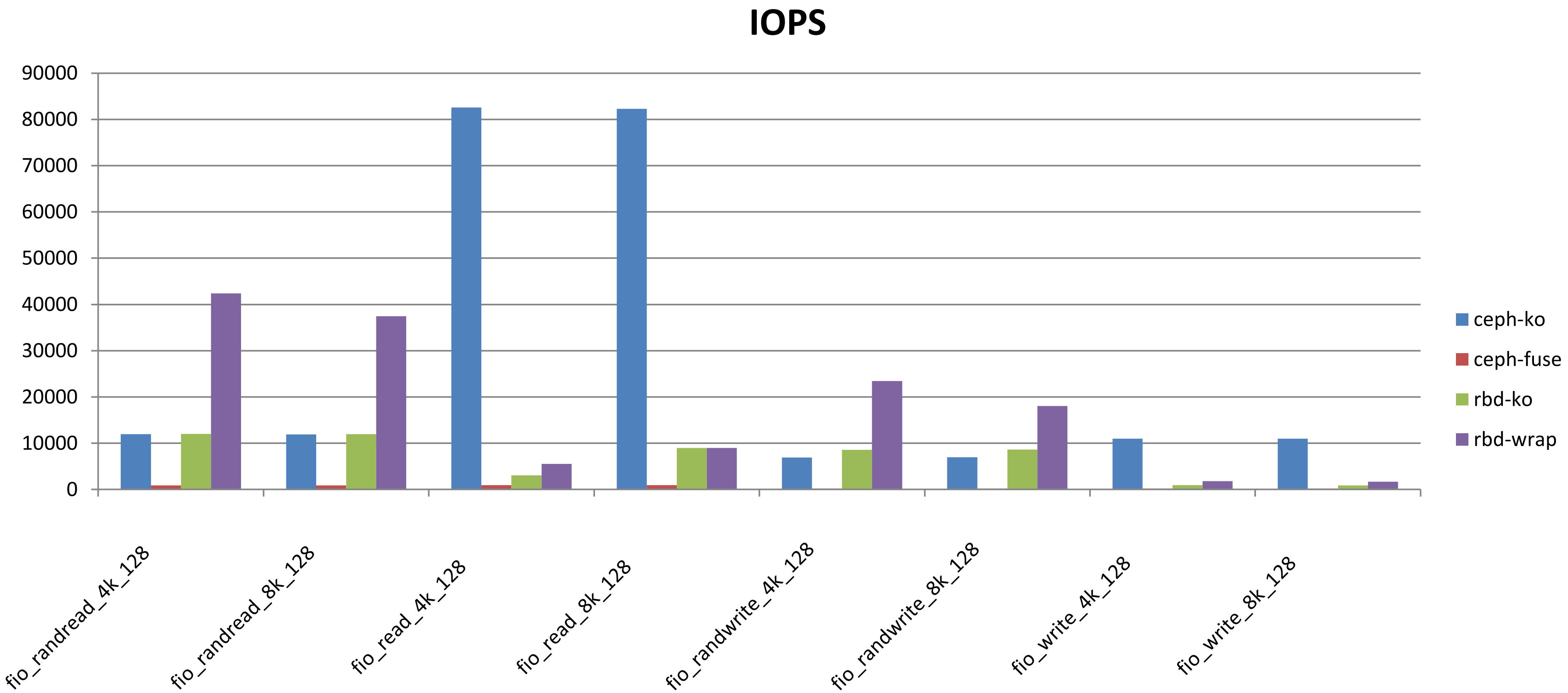


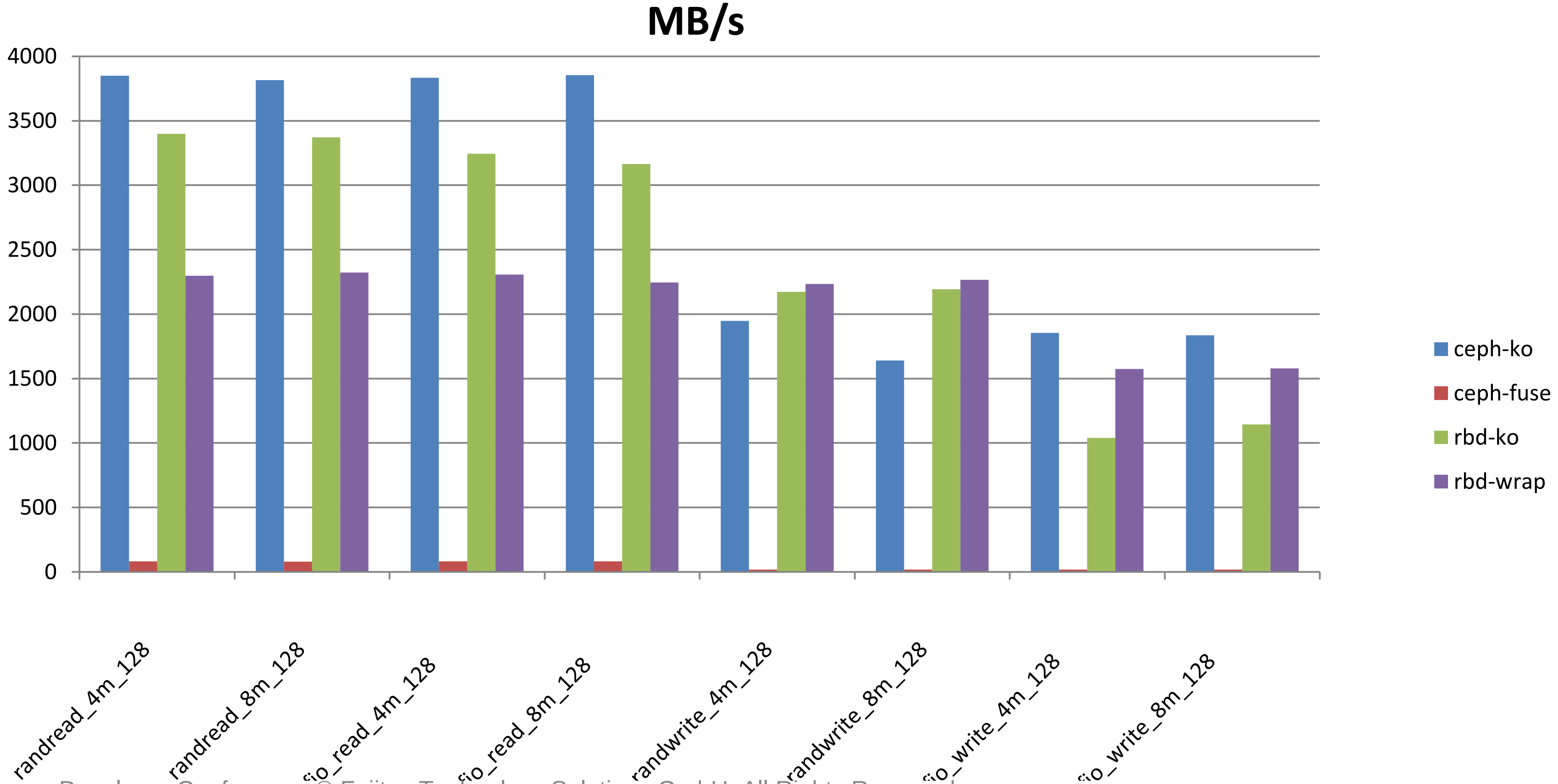
qperf - explanation

- tcp_lat almost the same for blocks <= 128 bytes
- tcp_lat very similar between 40GbE, 40Gb IPoIB and 56Gb IPoIB
- Significant difference between 1 / 10GbE only for blocks >= 4k
- Better latency on IB can only be achieved with rdma_write / rdma_write_poll

- Bandwidth on 1 / 10GbE very stable on possible maximum
- 40GbE implementation (MLX) with unexpected fluctuation
- Under IB only with RDMA the maximum transfer rate can be achieved

- Use Socket over RDMA
- Options without big code changes are: SDP, rsocket, SMC-R

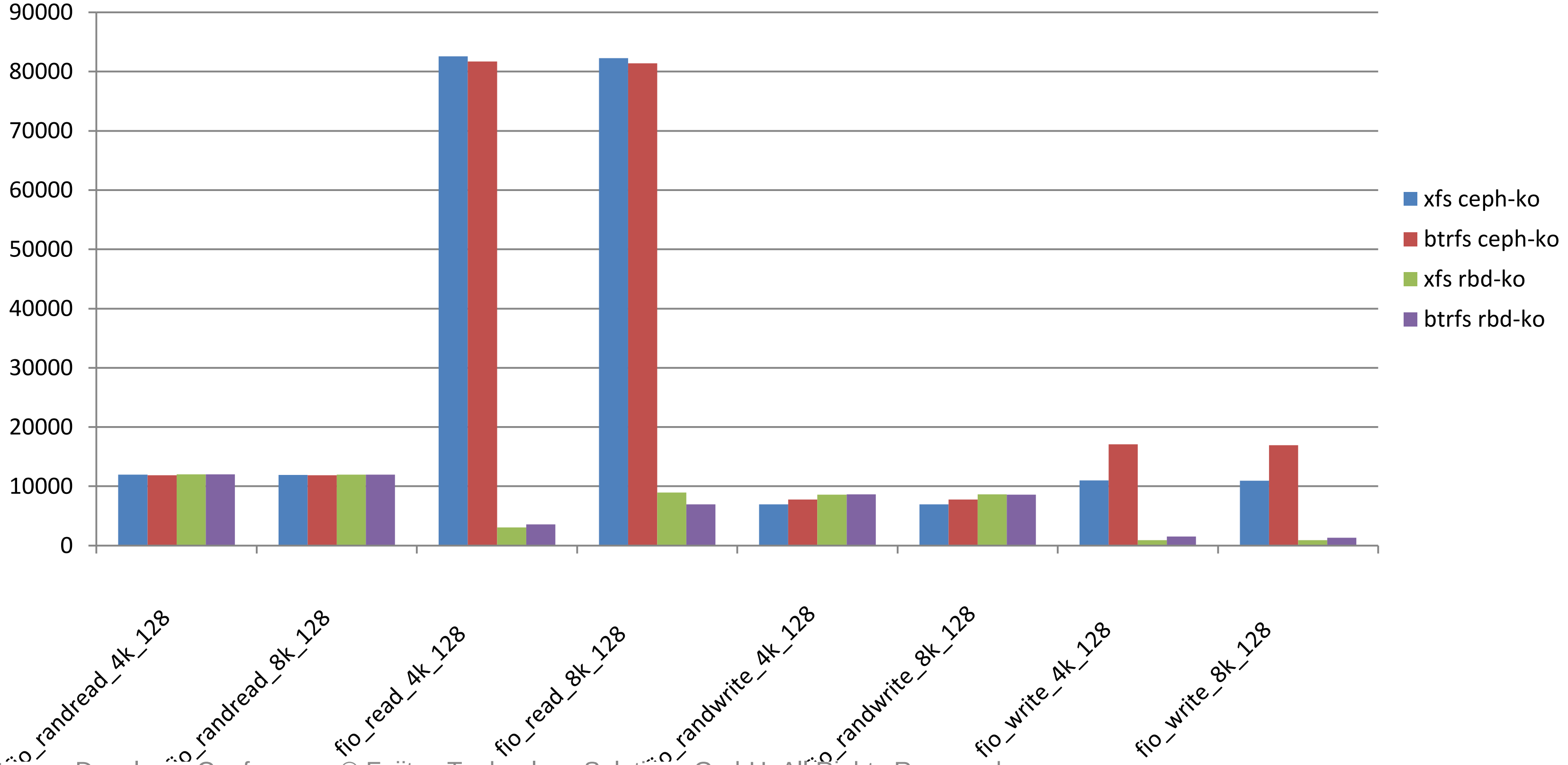

IOPS
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
fio_randread_4k_128
fio_randread_8k_128
fio_read_4k_128
fio_read_8k_128
fio_randwrite_4k_128
fio_randwrite_8k_128
fio_write_4k_128
fio_write_8k_128
ceph-ko
ceph-fuse
rbd-ko
rbd-wrap


MB/s
4000
3500
3000
2500
2000
1500
1000
500
0
randread_4m_128
randread_8m_128
fio_read_4m_128
fio_read_8m_128
randwrite_4m_128
randwrite_8m_128
fio_write_4m_128
fio_write_8m_128
ceph-ko
ceph-fuse
rbd-ko
rbd-wrap

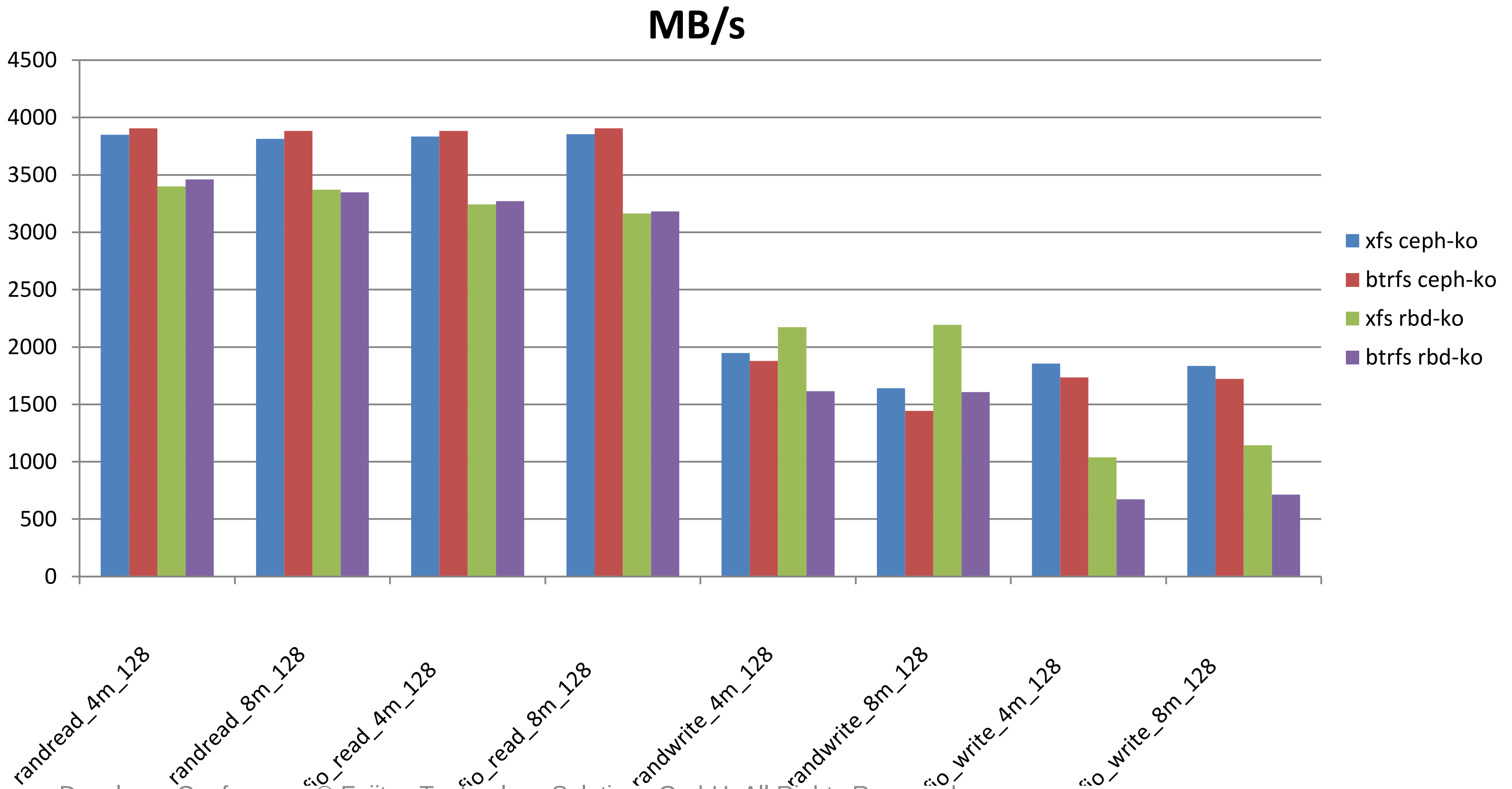
- Ceph-fuse seems not to support multiple I/Os in parallel with O_DIRECT which drops the performance significantly
- RBD-wrap (= rbd in user land) shows some advantages on IOPS, but not sufficient enough to replace rbd.ko
- Ceph.ko is excellent on sequential IOPS reads, presumable because of the read (ahead) of complete 4m blocks

- Stay with the official interfaces ceph.ko / rbd.ko to give fio the needed access to File and Block
- rbd.ko has some room for performance improvement for IOPS

IOPS

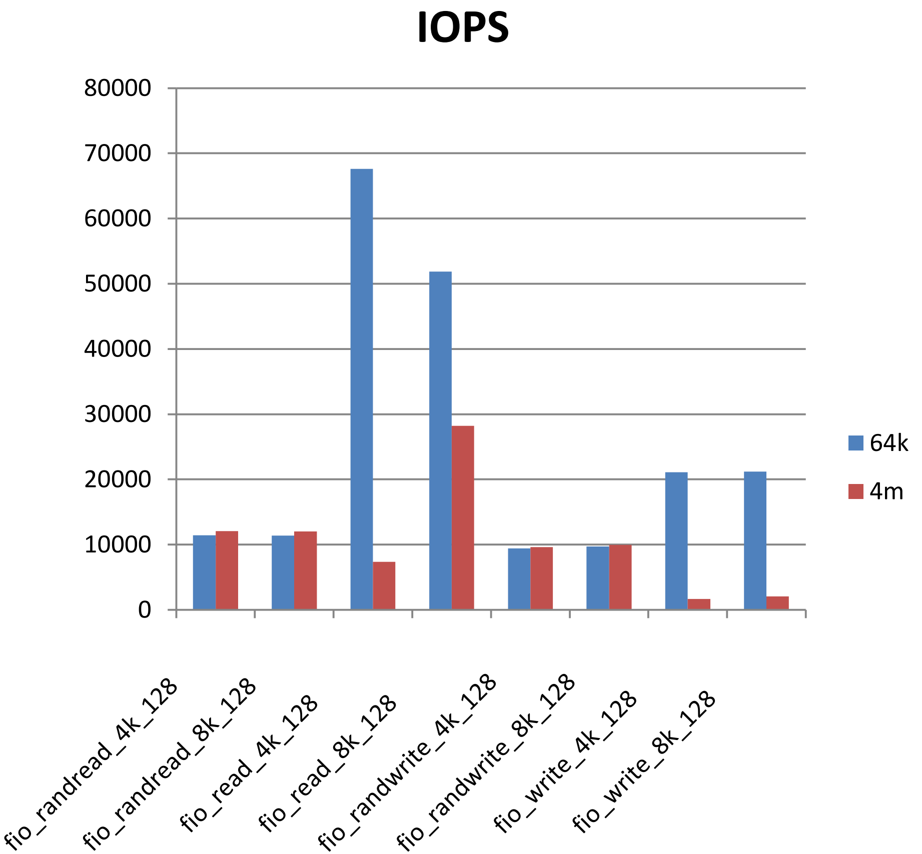


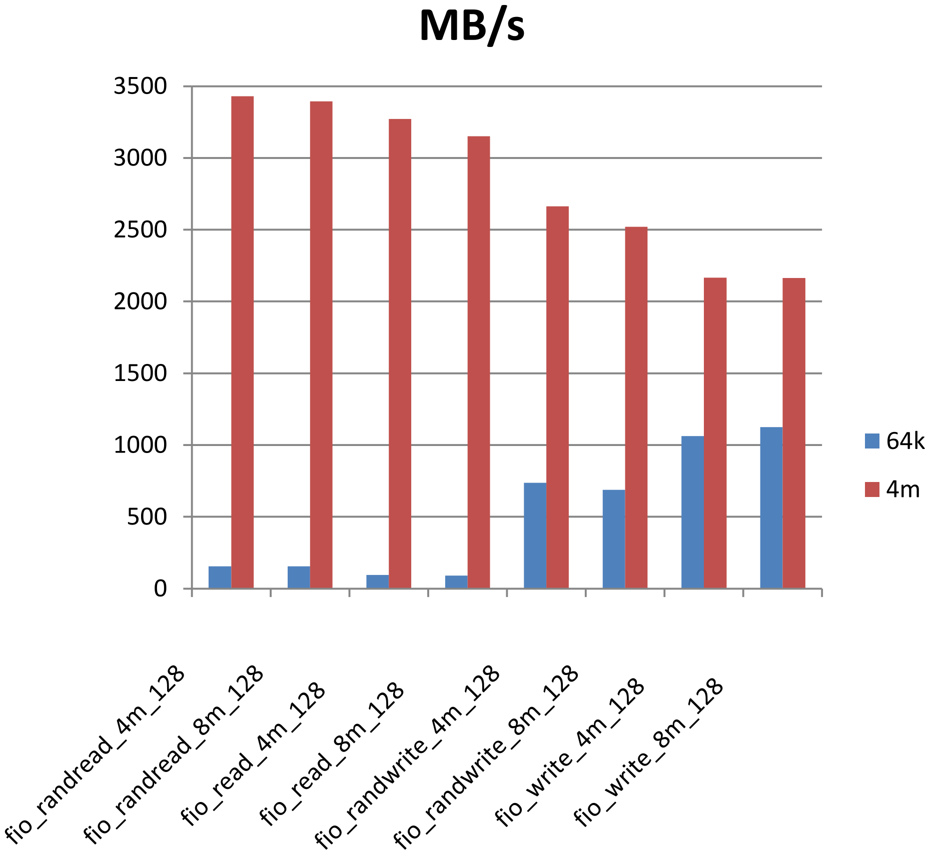
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
xfs ceph-ko
btrfs ceph-ko
xfs rbd-ko
btrfs rbd-ko
randread_4k_128
randread_8k_128
fio_read_4k_128
fio_read_8k_128
randwrite_4k_128
randwrite_8k_128
fio_write_4k_128
fio_write_8k_128


MB/s
4500
4000
3500
3000
2500
2000
1500
1000
500
0
randread_4m_128
randread_8m_128
fio_read_4m_128
fio_read_8m_128
randwrite_4m_128
randwrite_8m_128
fio_write_4m_128
fio_write_8m_128
xfs ceph-ko
btrfs ceph-ko
xfs rbd-ko
btrfs rbd-ko

- 6 months ago with kernel 3.0.x btrfs reveal some weaknesses in writes
- With kernel 3.6 some essential enhancements were made to the btrfs code, so almost no differences could be identified in our kernel 3.8.13

- ➢ Use btrfs in the next test cases, because btrfs has the more promising storage features: compression, data deduplication


IOPS
80000
70000
60000
50000
40000
30000
20000
10000
0
64k
4m
fio_randread_4k_128
fio_randread_8k_128
fio_read_4k_128
fio_read_8k_128
fio_randwrite_4k_128
fio_randwrite_8k_128
fio_write_4k_128
fio_write_8k_128

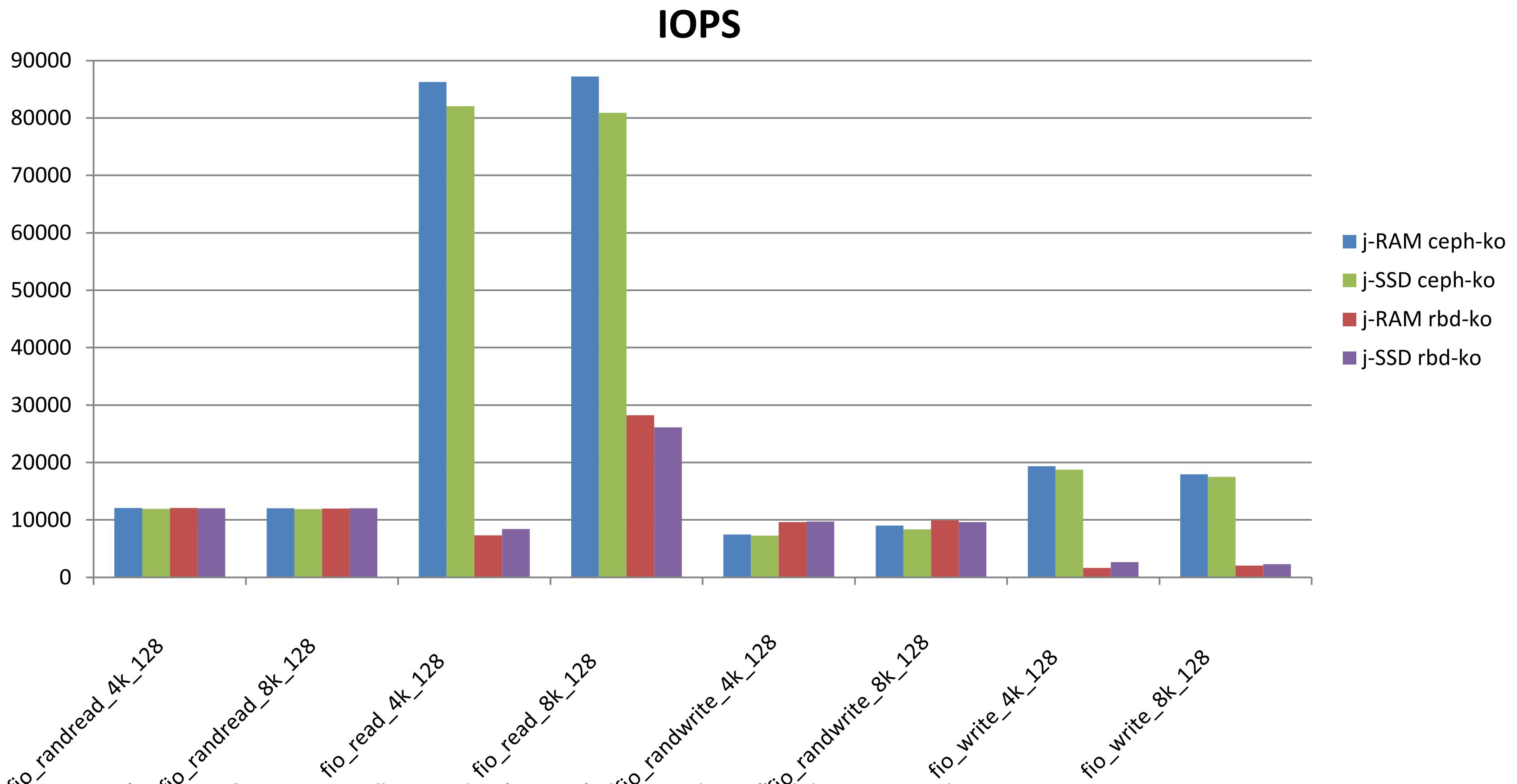

MB/s
3500
3000
2500
2000
1500
1000
500
0
64k
4m
fio_randread_4m_128
fio_randread_8m_128
fio_read_4m_128
fio_read_8m_128
fio_randwrite_4m_128
fio_randwrite_8m_128
fio_write_4m_128
fio_write_8m_128

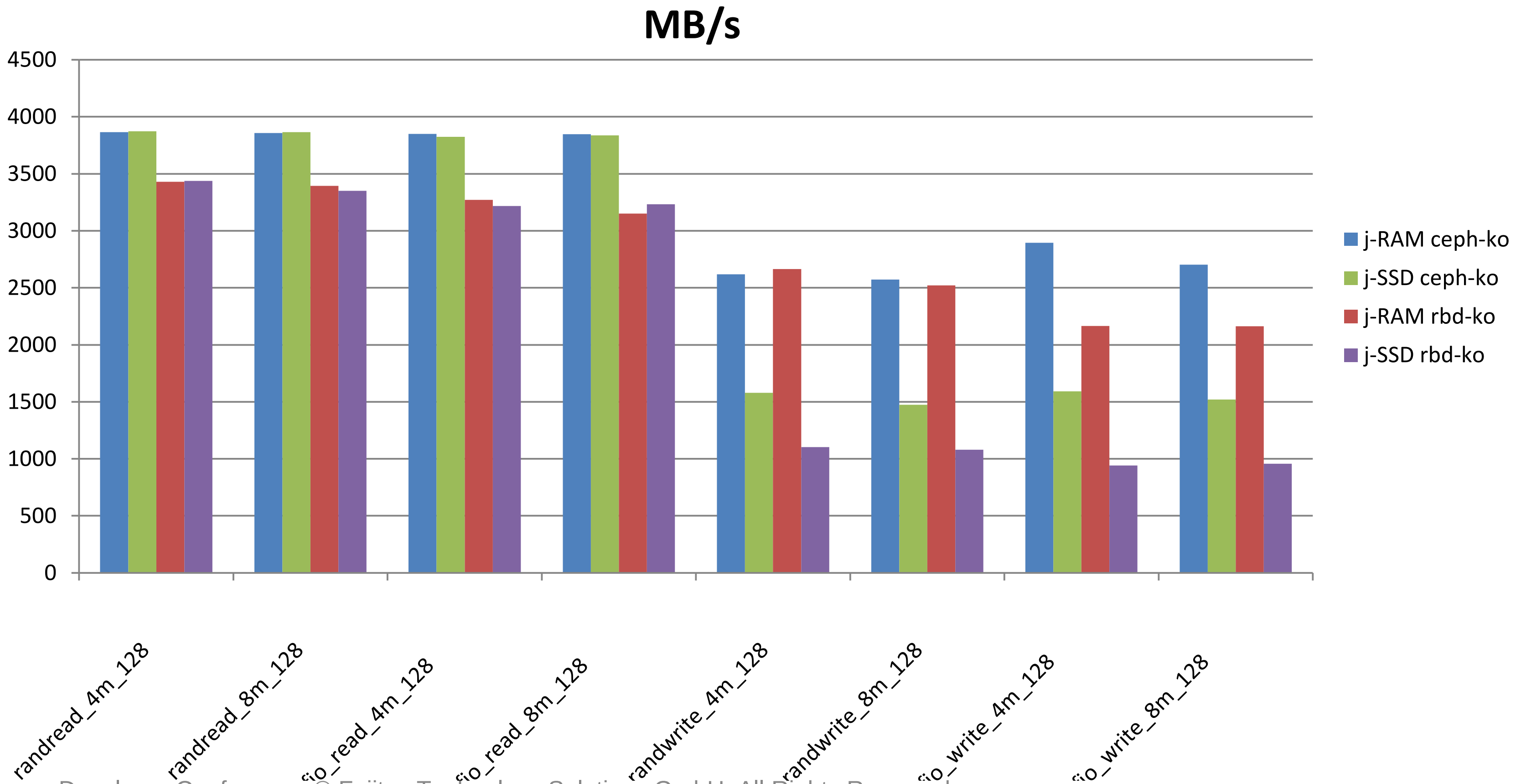
- Small chunks of 64k can especially increase 4k/8k sequential IOPS for reads as well as for writes
- For random IO 4m chunks are as good as 64k chunks

- But, the usage of 64k chunks will result in very low bandwidth for 4m/8m blocks

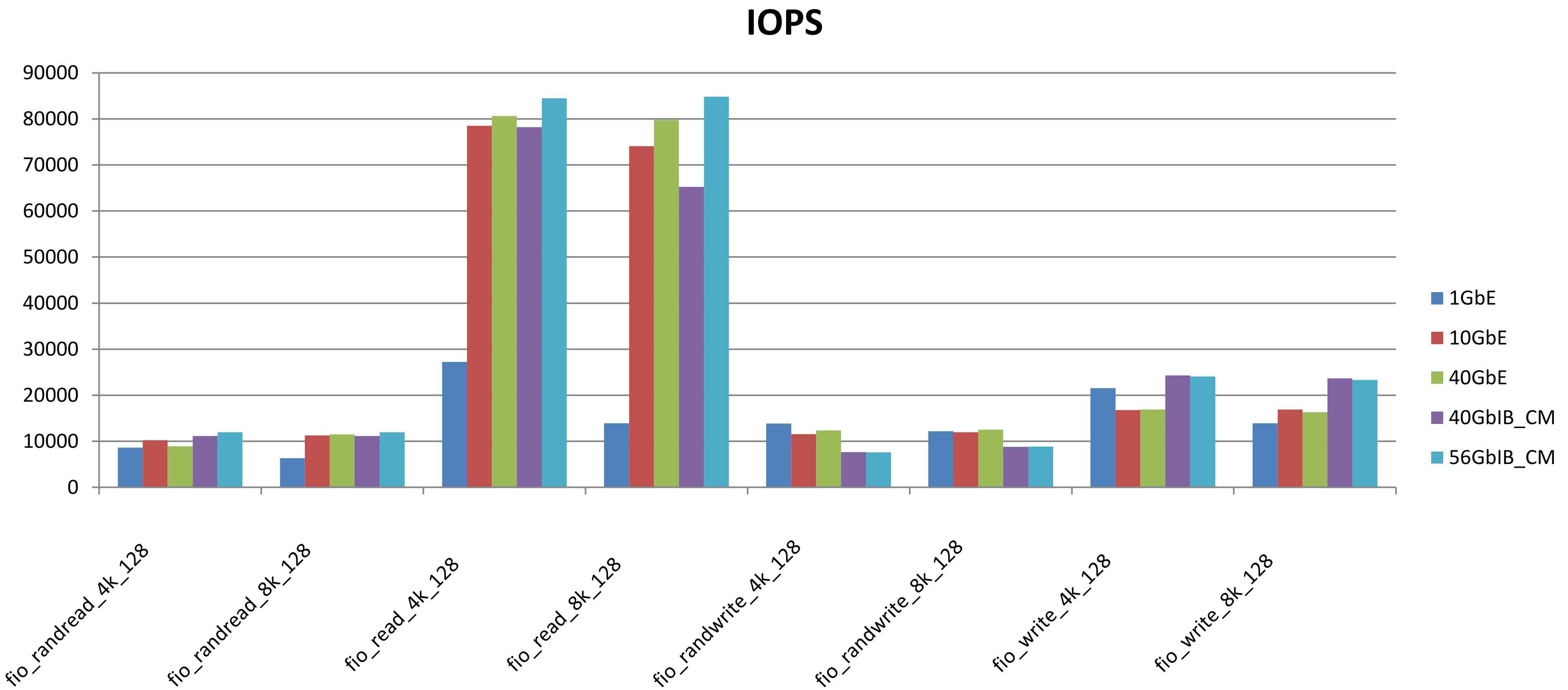
- Create each volume with the appropriate OSD object size: ~64k if small sequential IOPS are used, otherwise stay with ~4m

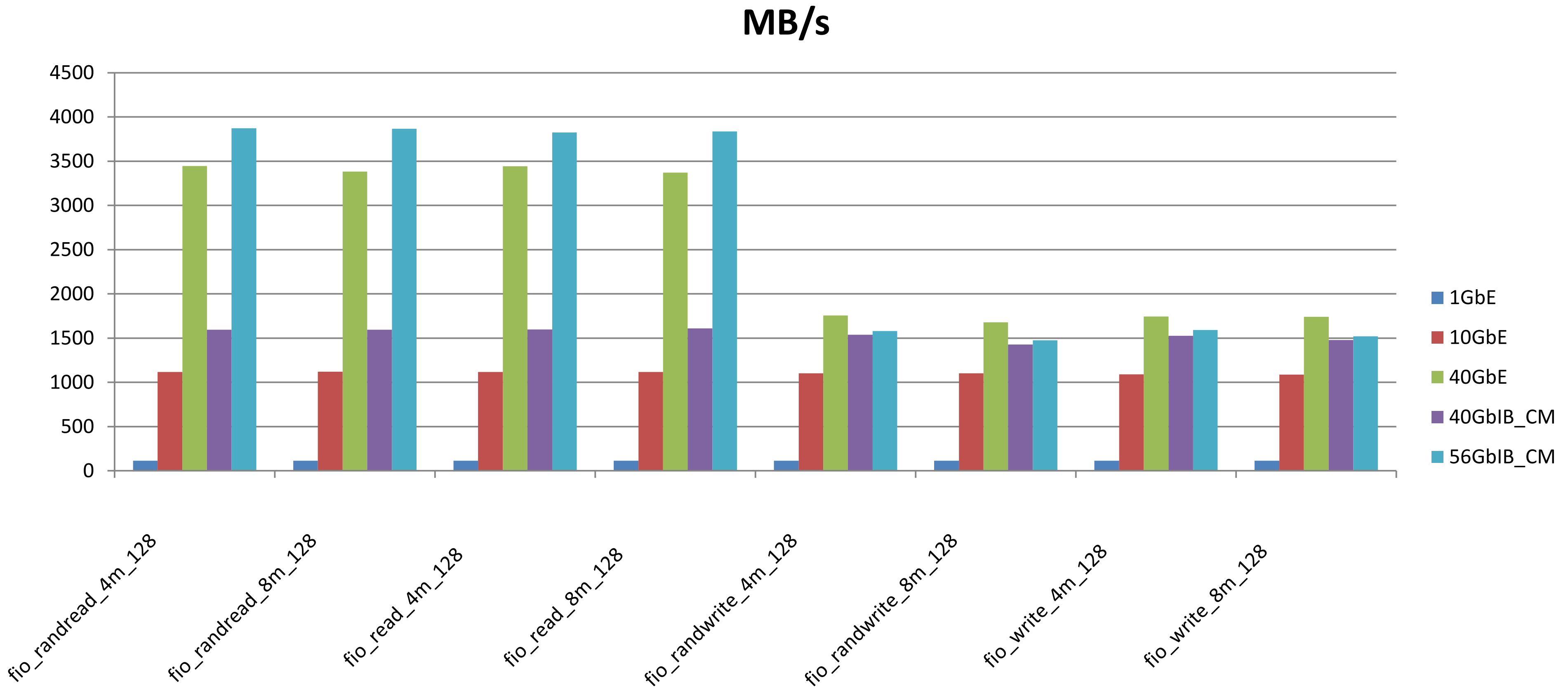
ceph-ko rbd-ko	OSD-FS btrfs	OSD object size 4m	Journal RAM / SSD	56 Gb IPoIB_CM	OSD-DISK SSD




MB/s
4500
4000
3500
3000
2500
2000
1500
1000
500
0
randread_4m_128
randread_8m_128
fio_read_4m_128
fio_read_8m_128
randwrite_4m_128
randwrite_8m_128
fio_write_4m_128
fio_write_8m_128
j-RAM ceph-ko
j-SSD ceph-ko
j-RAM rbd-ko
j-SSD rbd-ko

- Only on heavy write bandwidth requests RAM can show its predominance used as a journal.
- The SSD has to accomplish twice the load when the journal and the data is written to it.

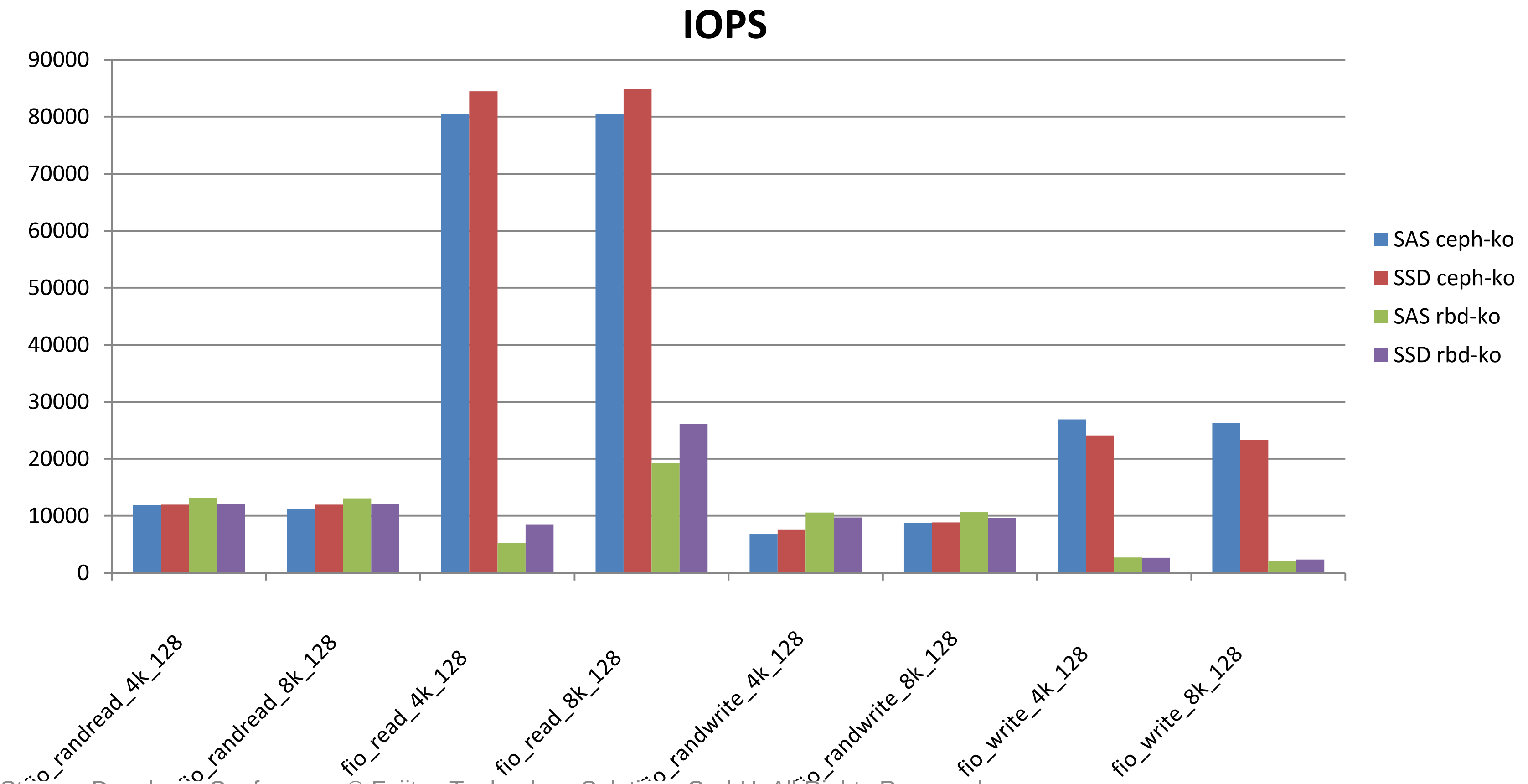

IOPS
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
fio_randread_4k_128
fio_randread_8k_128
fio_read_4k_128
fio_read_8k_128
fio_randwrite_4k_128
fio_randwrite_8k_128
fio_write_4k_128
fio_write_8k_128
1GbE
10GbE
40GbE
40GbIB_CM
56GbIB_CM


MB/s
4500
4000
3500
3000
2500
2000
1500
1000
500
0
fio_randread_4m_128
fio_randread_8m_128
fio_read_4m_128
fio_read_8m_128
fio_randwrite_4m_128
fio_randwrite_8m_128
fio_write_4m_128
fio_write_8m_128
1GbE
10GbE
40GbE
40GbIB_CM
56GbIB_CM

- In case of write IOPS 1GbE is doing extremely well
- On sequential read IOPS there is nearly no difference between 10GbE and 56Gb IPoIB

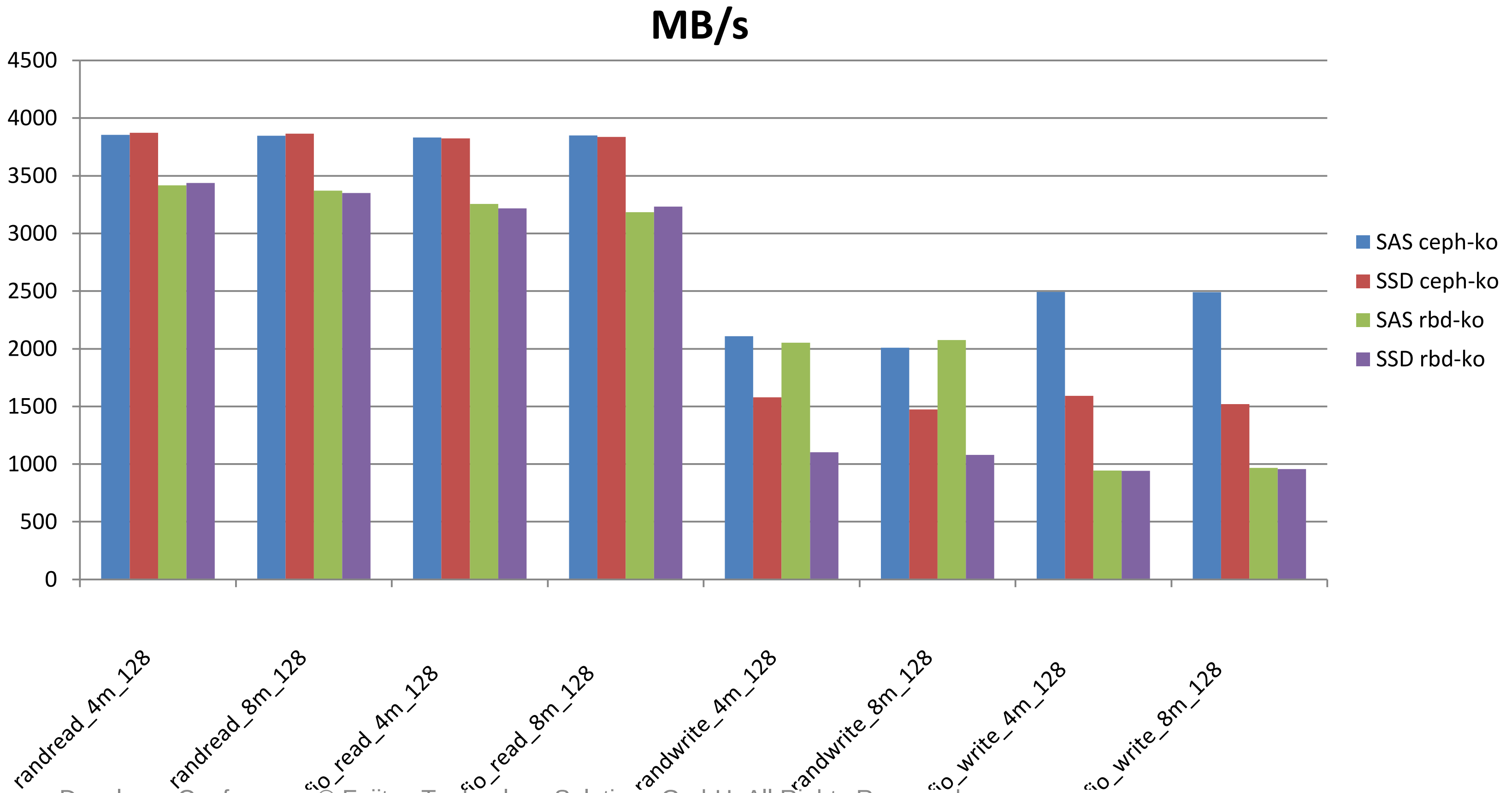
- On the bandwidth side with reads the measured performance is close in sync with the possible speed of the network. Only the TrueScale IB has some weaknesses, because it was designed for HPC and not for Storage/Streaming.

- If you only look or IOPS 10GbE is a good choice
- If throughput is relevant for your use case you should go for 56GbIB

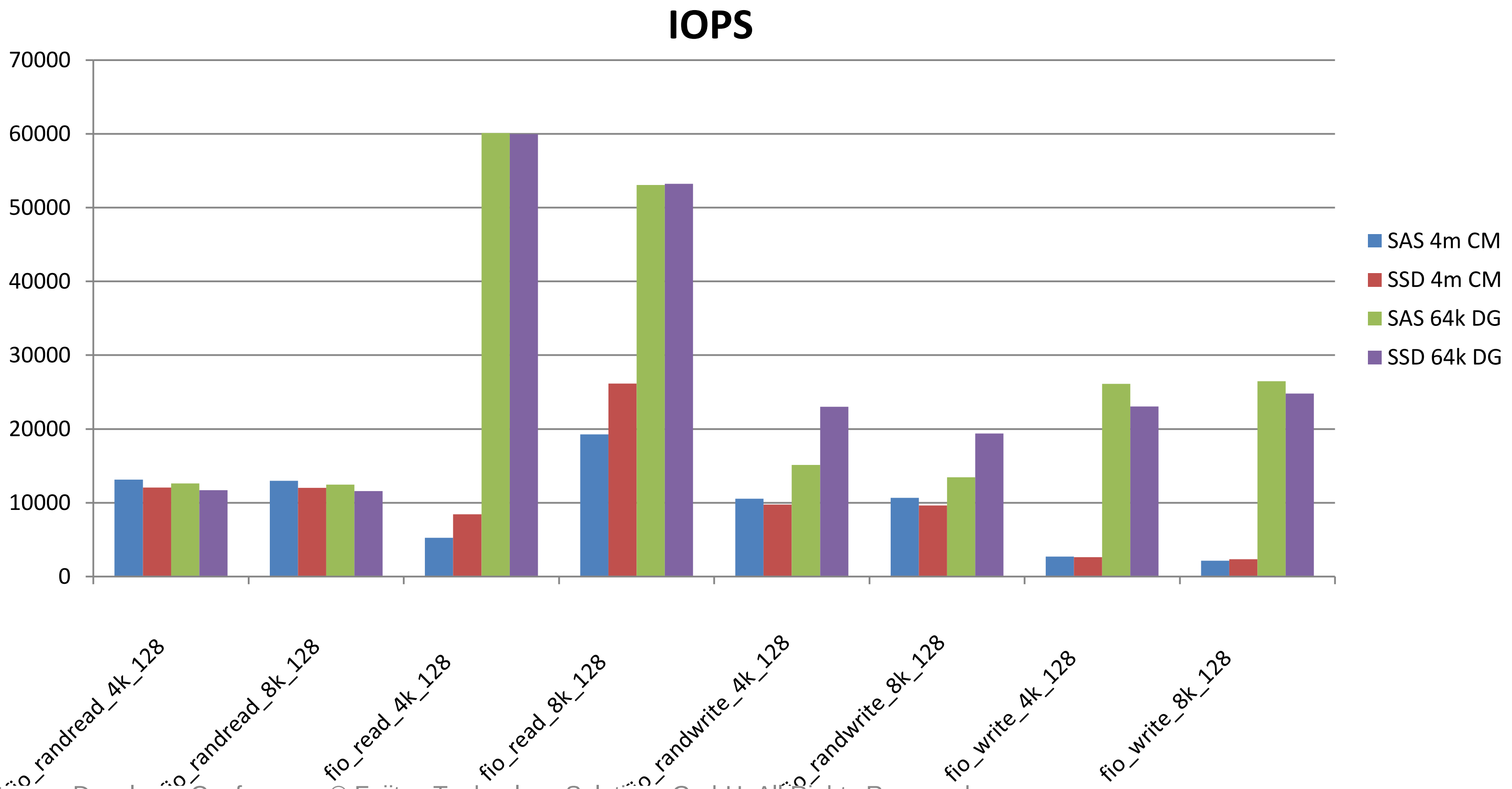


IOPS
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
SAS ceph-ko
SSD ceph-ko
SAS rbd-ko
SSD rbd-ko
fio_randread_4k_128
fio_randread_8k_128
fio_read_4k_128
fio_read_8k_128
fio_randwrite_4k_128
fio_randwrite_8k_128
fio_write_4k_128
fio_write_8k_128

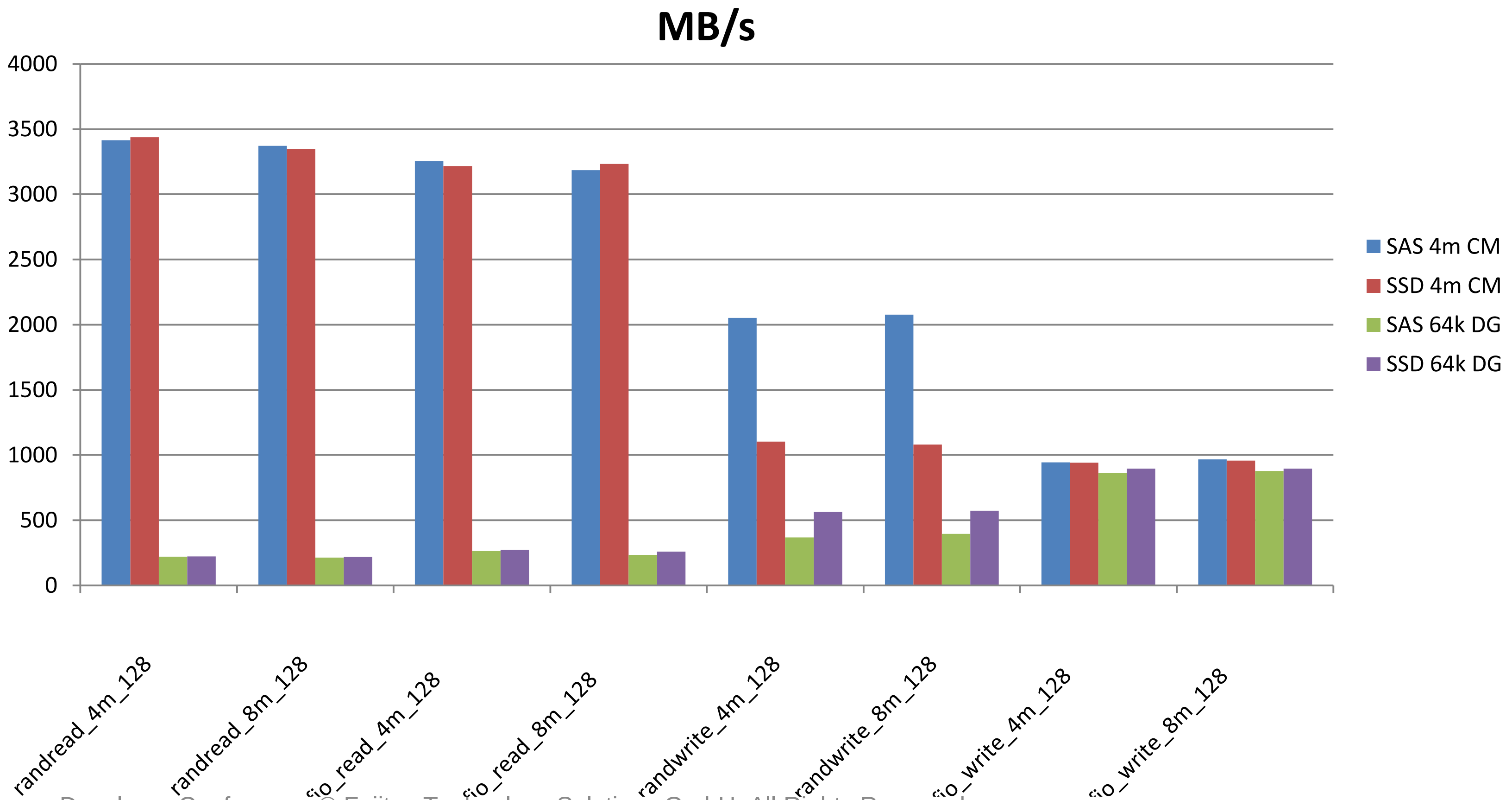
ceph-ko rbd-ko	OSD-FS btrfs	OSD object size 4m	Journal SSD	56 Gb IPoIB_CM	OSD-DISK SAS / SSD



rbd-ko	OSD-FS btrfs	OSD object 64k / 4m	Journal SSD	56 GbIPoIB CM / DG	OSD-DISK SAS / SSD



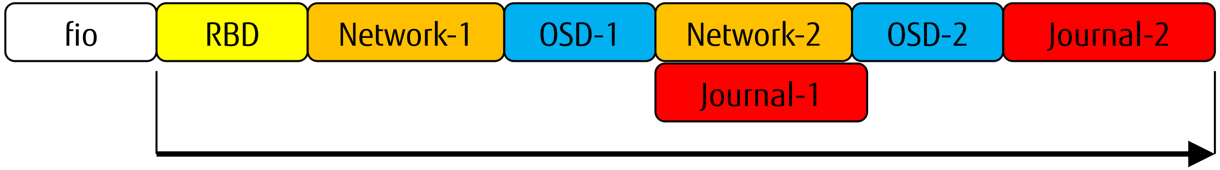
rbd-ko	OSD-FS btrfs	OSD object 64k / 4m	Journal SSD	56 GbIPoIB CM / DG	OSD-DISK SAS / SSD



- SAS drives are doing much better than expected. Only on small random writes they are significantly slower than SSD
- In this comparison is has to be taken into account, that the SSD is getting twice as much writes (journal + data)

- 2.5” 10k 6G SAS drives seems to be an attractive alternative in combination with SSD for the journal

Analysis of the TAT of a single 4k IO



Time = avg latency of one IO (queue-depth=1) with 5x ACK

	rbd		Network		Intel 910		ACK	Ceph code	
µsec	fio write		qperf lat		fio_randwrite		msg		
	4k	8k	4k	8k	4k	8k	128	4k	8k
1 GbE	2565	2709	182	227	54	64	26	2017	2061
10 GbE	2555	2584	109	122	54	64	21	2178	2171
40 GbE	2191	2142	19	22	54	64	15	2024	1959
40 Gb IPoIB	2392	2357	29	24	54	64	18	2190	2155
56 Gb IPoIB	1848	1821	19	37	54	64	14	1686	1613

- approximately 1600 µs on a single 4k/8k IO is spend in the Ceph code
  - The Ceph code has a lot of room for improvement

Agenda

- Introduction
- Hardware / Software layout
- Tools how to monitor
- Transformation test cases
- **Conclusion**

Summary and conclusion

- Ceph is the most comprehensive implementation of Unified Storage. Ceph simulates “distributed swarm intelligence” which arise from simple rules that are followed by individual processes and does not involve any central coordination.
- The Crush algorithm acts as an enabler for a controlled, scalable, decentralized placement of replica data.
- The usage of TCP/IP will slow down the latency capabilities of InfiniBand, but the better bandwidth mostly remains. DG has some advantage in small blocks, but overall CM is the better compromise.
- Only an optimal setting of block device parameter in Linux will ensure to get the maximum performance out of the SSD.
- 2.5” 10k 6G SAS drives for data are an attractive alternative for high performance in combination with SSDs for the journal.

Conclusion and Outlook

- Only with Socket over RDMA a better bandwidth and lower latency of Infiniband can be utilized: Options are: SDP, rsocket, SMC-R

- The Ceph code has a lot of room for improvement to achieve lower latency.

Fujitsu Technology Solutions

FTS CTO, Principal Architect
Dieter.Kasper@ts.fujitsu.com

v8