Ceph Optimization on All Flash Storage

Somnath Roy

Lead Developer, SanDisk Corporation

Forward-Looking Statements

During our meeting today we may make forward-looking statements.

Any statement that refers to expectations, projections or other characterizations of future events or circumstances is a forward-looking statement, including those relating to market position, market growth, product sales, industry trends, supply chain, future memory technology, production capacity, production costs, technology transitions and future products. This presentation contains information from third parties, which reflect their projections as of the date of issuance.

Actual results may differ materially from those expressed in these forward-looking statements due to factors detailed under the caption “Risk Factors” and elsewhere in the documents we file from time to time with the SEC, including our annual and quarterly reports.

We undertake no obligation to update these forward-looking statements, which speak only as of the date hereof.

What is Ceph?

- A Distributed/Clustered Storage System
- File, Object and Block interfaces to common storage substrate
- Ceph is LGPL open source project
  - Part of Linux kernel and most distros since 2.6.x

Flash Readiness?

- Started with Ceph Emperor (~2 years ago)
- Not much scaling with increasing number of clients
- Minimal scaling if we increase number of SSDs
- No resources are saturated
- CPU core usage per IOPS is very high
- Double write due to Ceph journal is increasing WA by minimum 2X

SanDisk on Ceph

- We saw lot of potential on Ceph
- Decided to dig deep and optimize the code base for flash
- Ceph read path was comparatively easier to tackle, so, started with that first
- Also, lot of our potential customer workload was kind of WORM

Bottlenecks Identified and Fixed

- Optimized lot of CPU intensive code path
- Found out context switching overhead is significant if backend is very fast
- Lot of lock contention overhead popped up, sharding helped a lot
- Fine grained locking helped to achieve more parallelism
- New/delete overhead on IO path is becoming significant
- Efficient caching of indexes (placement groups) is beneficial
- Efficient buffering while reading from socket
- Need to disable Nagle's algorithm while scaling out
- Needed to optimize tcmalloc for object size < 32k

Time for a Benchmark of..

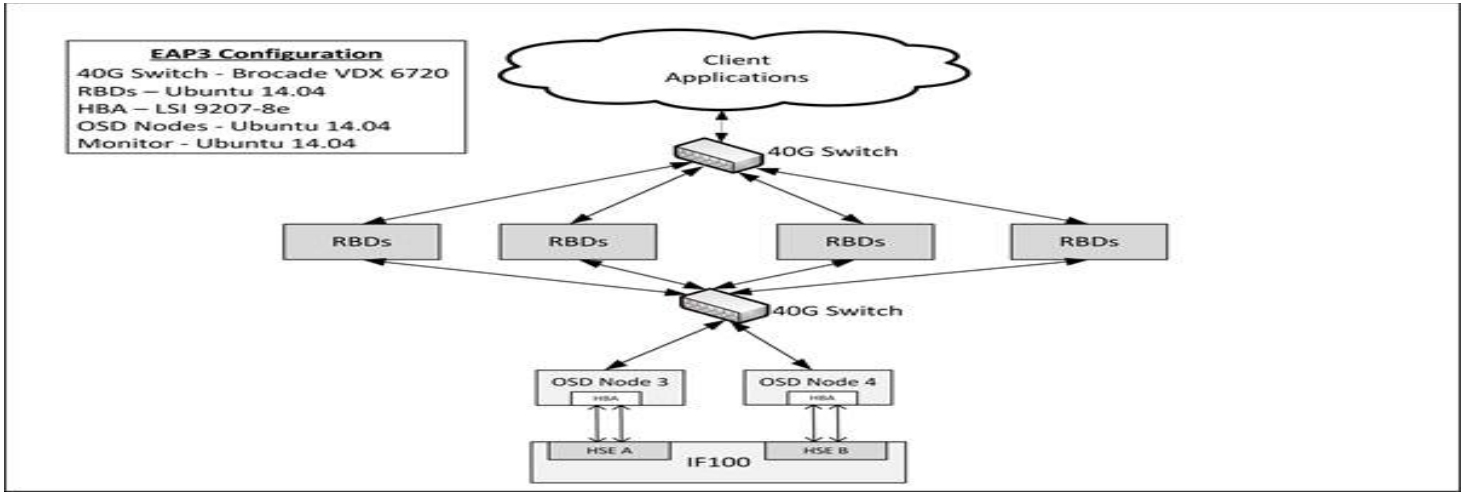
InfiniFlash™ System IF500

Ceph with all the optimizations mentioned on top of InfiniFlash IF100

+

Proper filesystem/kernel tuning optimized for InfiniFlash IF100

InfiniFlash IF500 Topology on a Single 512TB InfiniFlash IF100

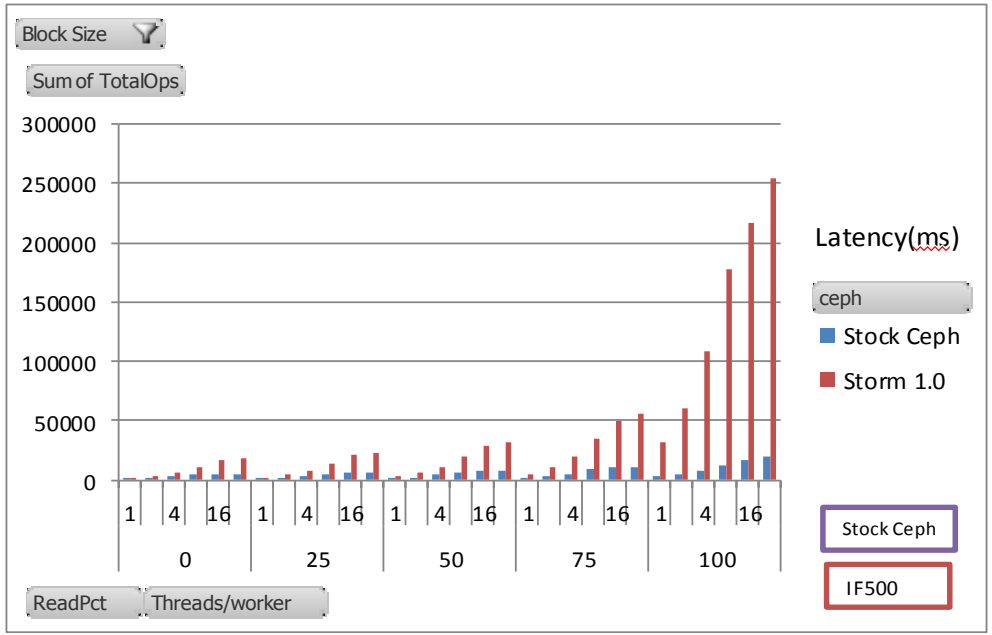


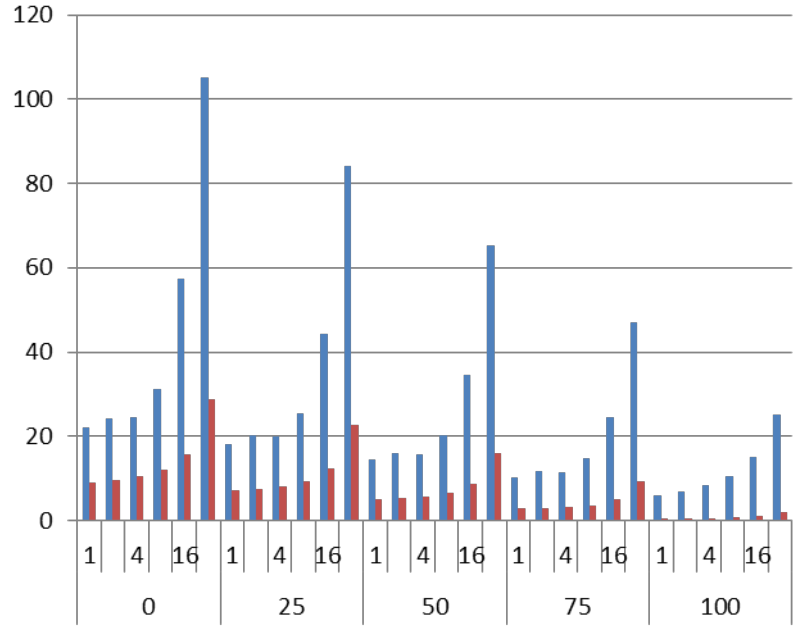
- IF100 BW is **~8.5GB/s (with 6Gb SAS, 12 Gb is coming EOY)** and **~1.5M** 4K RR IOPS
- We saw that Ceph is very resource hungry, so, need **at least** 2 physical nodes on top of IF100
- We need to connect all 8 ports of an HBA to saturate IF100 for bigger block size

Setup Details

Performance Config - InfiniFlash System IF500		
2 Node Cluster (32 drives shared to each OSD node)		
Node	2 Servers (Dell R620)	2 x E5-2680 12C 2.8GHz 4 x 16GB RDIMM, dual rank x 4 (64GB) 1 x Mellanox X3 Dual 40GbE 1 x LSI 9207 HBA card
RBD Client	4 Servers (Dell R620)	1 x E5-2680 10C 2.8GHz 2 x 16GB RDIMM, dual rank x 4 (32 GB) 1 x Mellanox X3 Dual 40GbE
Storage – InfiniFlash IF100 with 64 cards in A2 Config		
InfiniFlash IF100	IF100 is connected 64 x 1YX2 cards in A2 topology	Total storage - 64 * 8TB = 512TB
Network Details		
40G Switch	NA	
OS Details		
OS	Ubuntu 14.04 LTS 64bit	3.13.0-32
LSI card/ driver	SAS2308(9207)	mpt2sas
Mellanox 40gbps nw card	MT27500 [ConnectX-3]	mlx4_en - 2.2-1 (Feb 2014)
Cluster Configuration		
CEPH Version	sndk-ifos-1.0.0.04	0.86.rc.eap2
Replication (Default)	2 [Host]	Note: - Host level replication.
Number of Pools, PGs & RBDs	pool = 4 ;PG = 2048 per pool	2 RBDs from each pool
RBD size	2TB	
Number of Monitors	1	
Number of OSD Nodes	2	
Number of OSDs per Node	32	total OSDs = 32 * 2 = 64

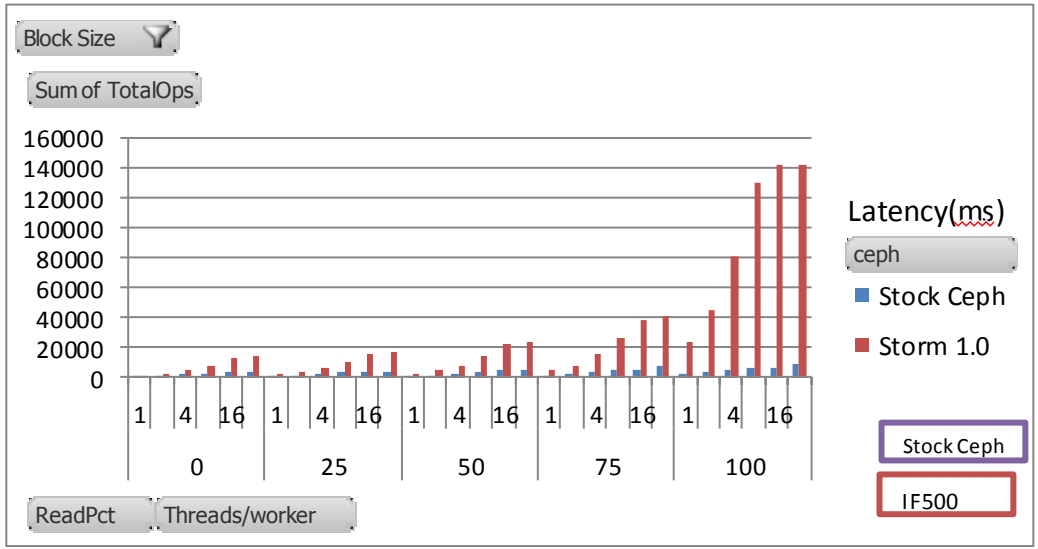
8K Random IO

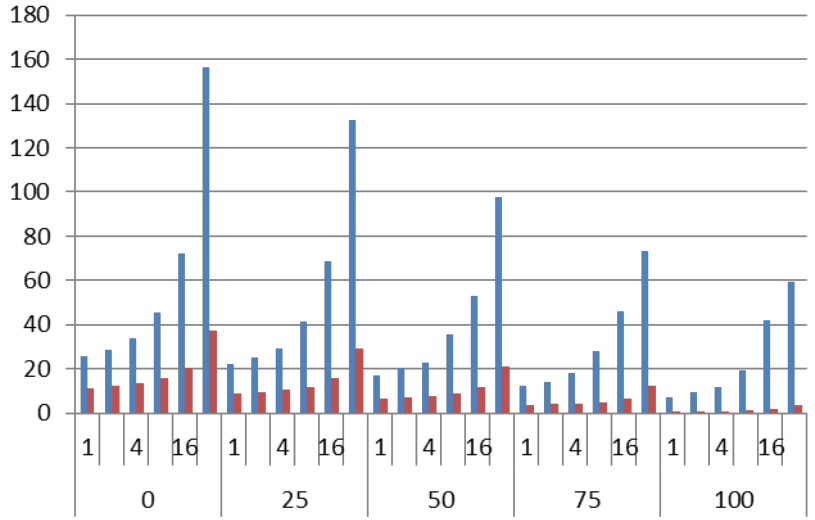




[Queue Depth]
Read Percent →

64K Random IO

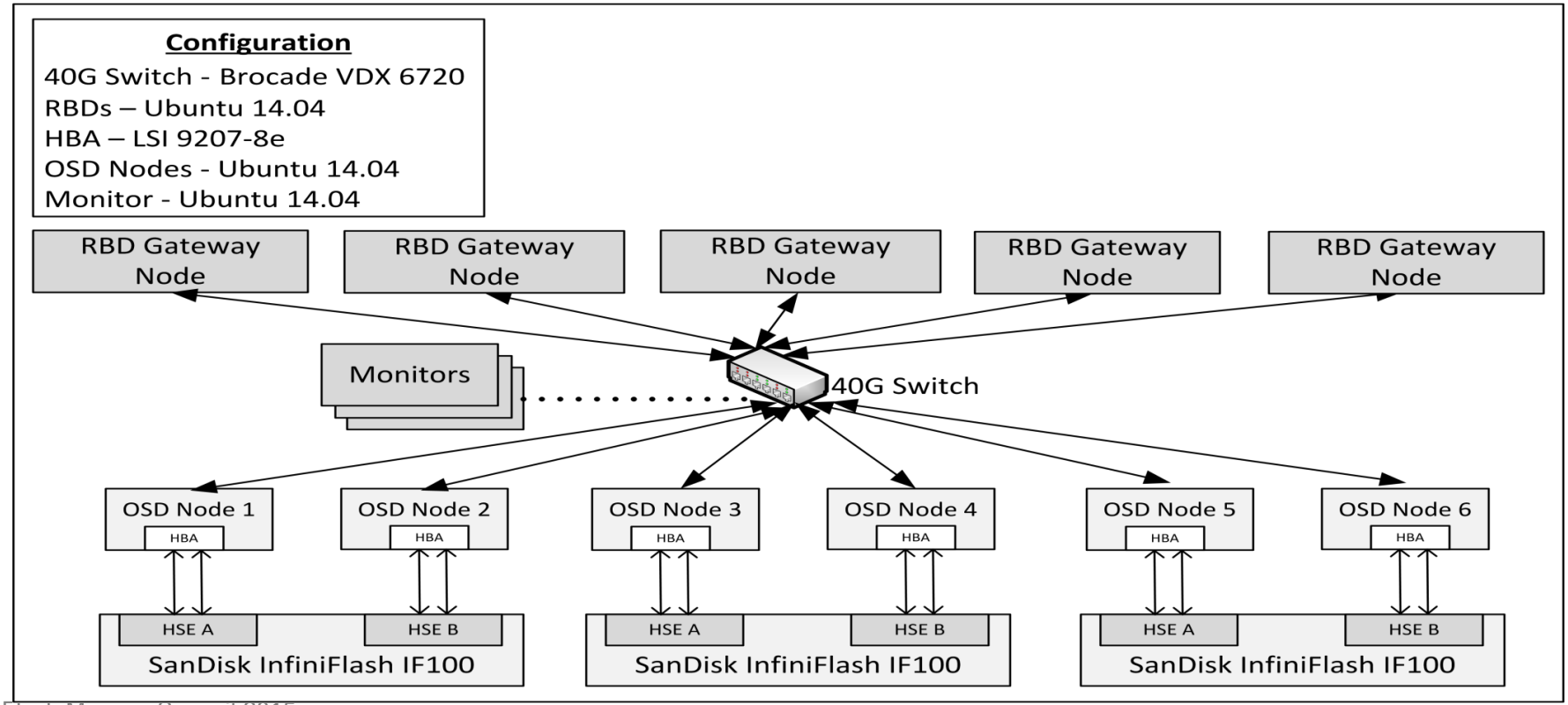




We are saturating ~8.5 GB/s IF100 BW here

[Queue Depth]
Read Percent →

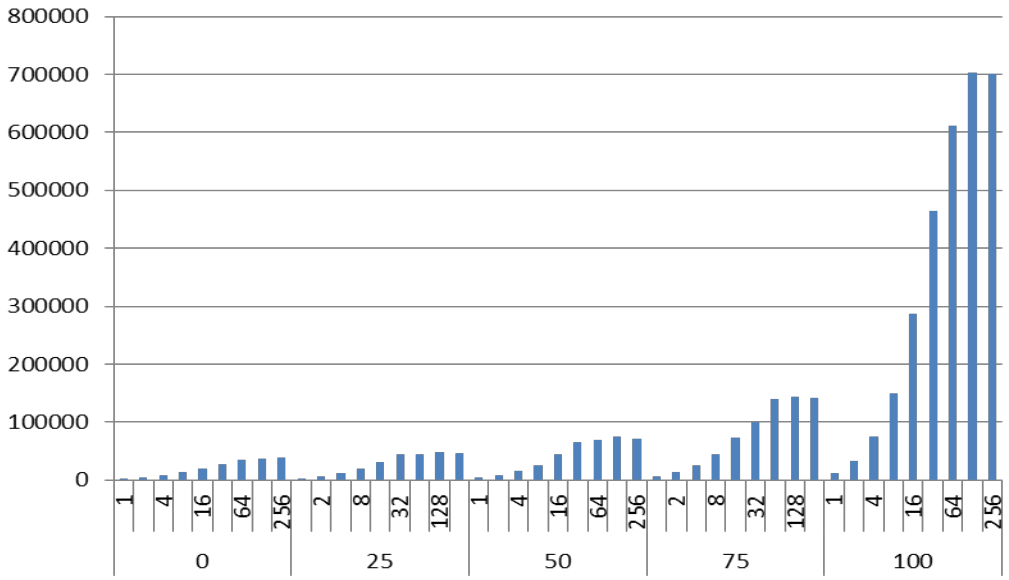
InfiniFlash IF500 Scale Out Topology

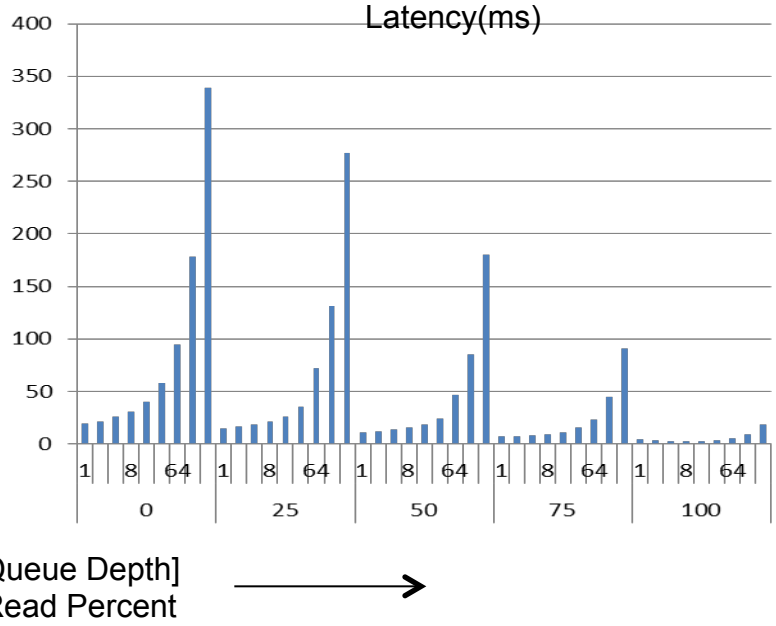


InfiniFlash IF500 HW Set Up

Performance Config		
6 Node Cluster (8 drives connected to each OSD node)		
Node	6 Servers (Dell R620)	2 x E5-2680 v2 2.8GHz 25M$ 8 x 16GB RDIMM, dual rank (128 GB) 1 x Mellanox X3 Dual 40GbE 1x LSI 9207 HBA card
RBD Client	5 Servers (Dell R620)	1 x E5-2680 v2 2.8GHz 25M$ 4 x 16GB RDIMM, dual rank (64 GB) 1 x Mellanox X3 Dual 40GbE
Network Details		
40G Switch	NA	
OS Details		
OS	Ubuntu 14.04 LTS 64bit	3.13.0-32
LSI card/ driver	SAS2308(9207) / mpt2sas	16.100.00.00
Mellanox 40gbps nw card	MT27500 [ConnectX-3] / MLX4_EN	mlx4_en - 2.2-1
Cluster Configuration		
CEPH Version	sndk-ifos-1.0.0.07	0.87-IFOS-1.0.0.7.beta
Replication (Default)	3 (CHASSIS)	Note: - chassis level replication
Number of Pools, PGs & RBDs	pool = 5 ; PG = 2048 per pool	2 RBDs from each pool
RBD size	3TB	total DATA SET size = 30TB
Number of Monitors	1	
Number of OSD Nodes	6	
Number of OSDs per Node	8	total OSDs = 6 * 8 = 48

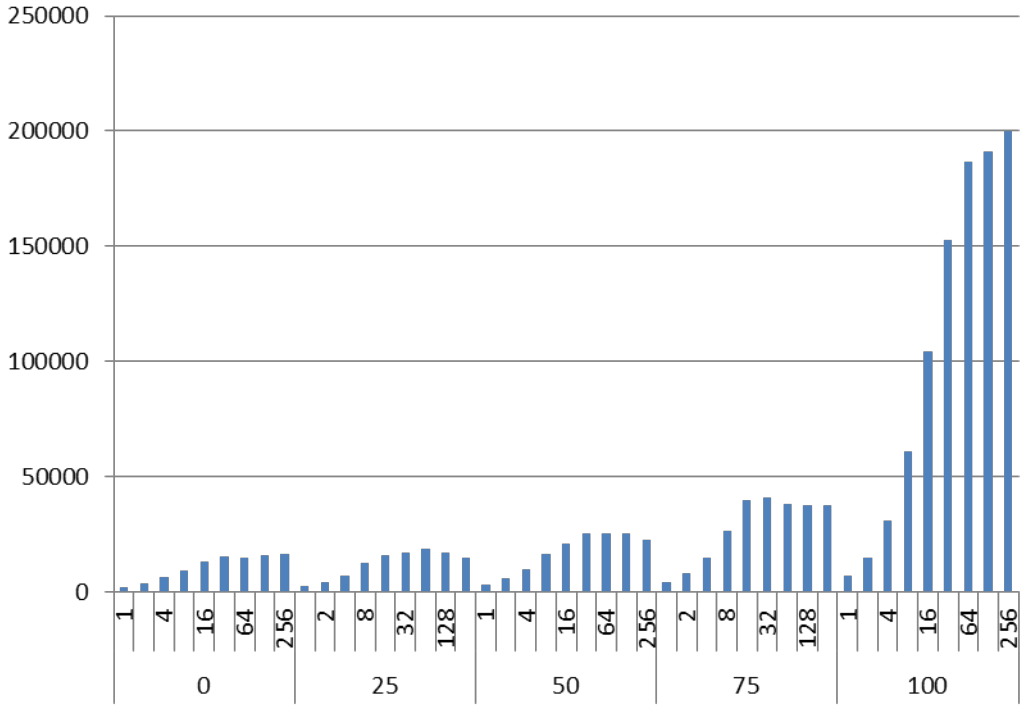
4K Random IOPS Performance



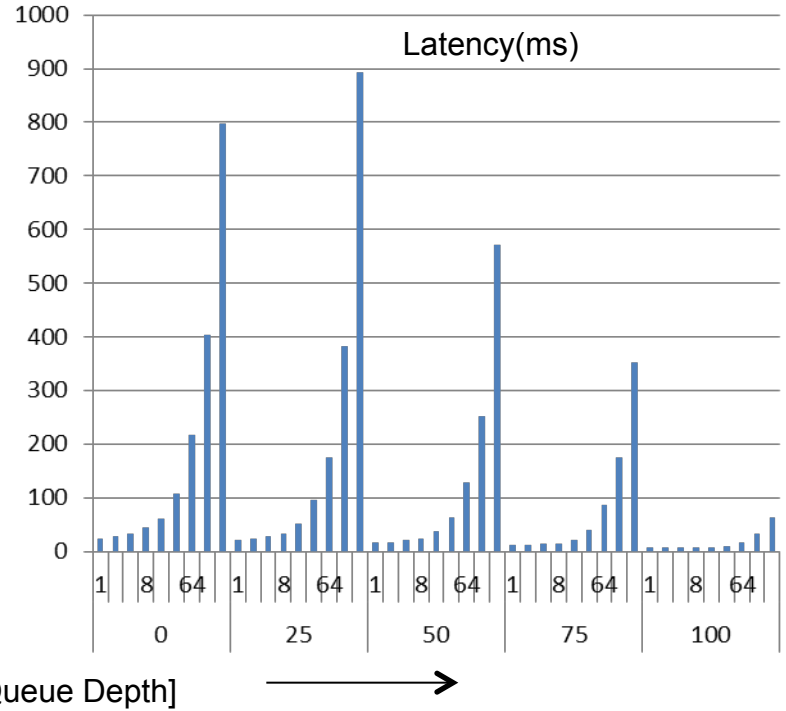


With tuning, maximum cumulative performance of ~**700K** IOPS measured towards 4KB blocks, saturating node CPUs at this point.

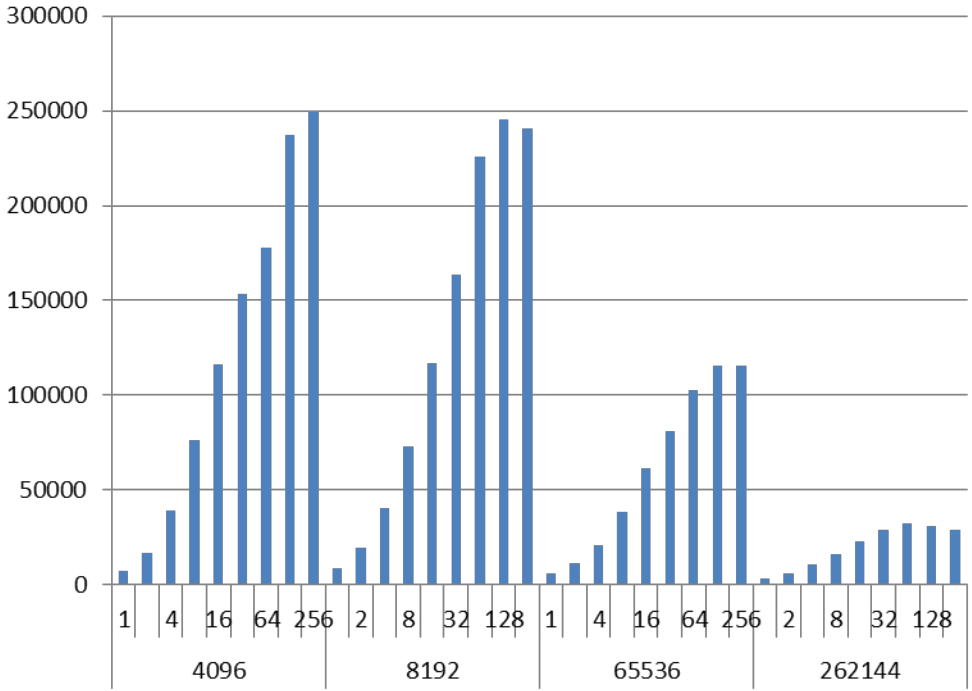
64K Random IOPS Performance

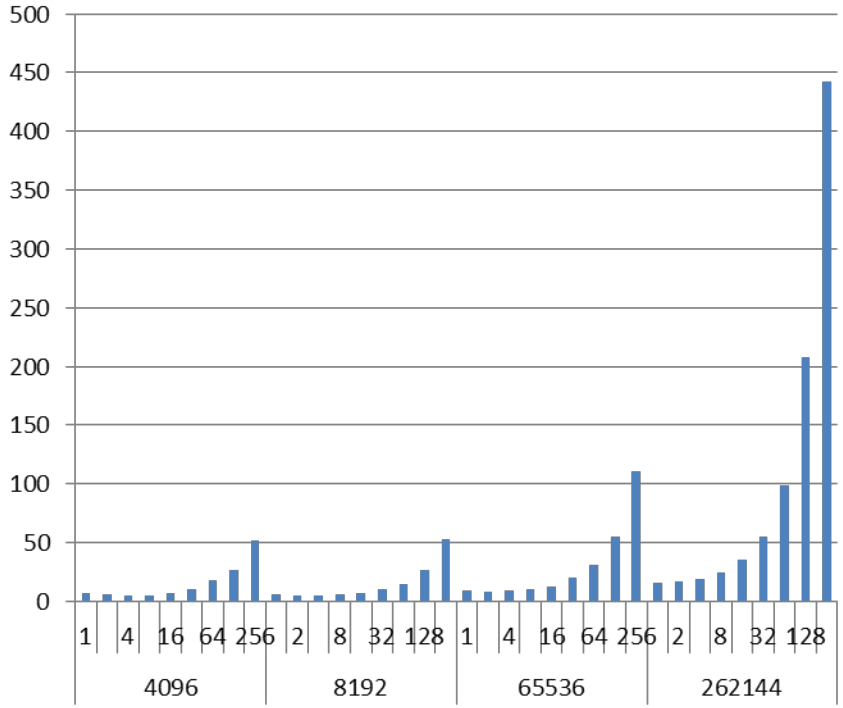


Maximum Bandwidth of **12.8GB/s** measured towards 64KB blocks



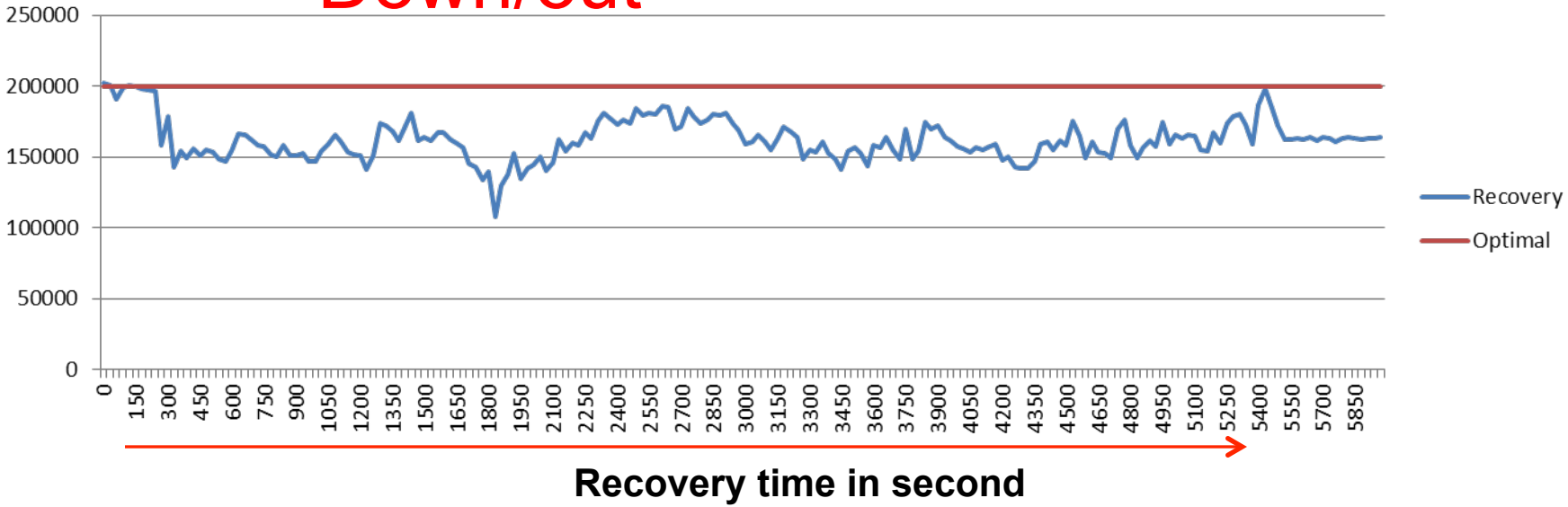
90-10 Random Read Performance





[Queue Depth]
block size

Client IO with Recovery after 1 OSD Down/out



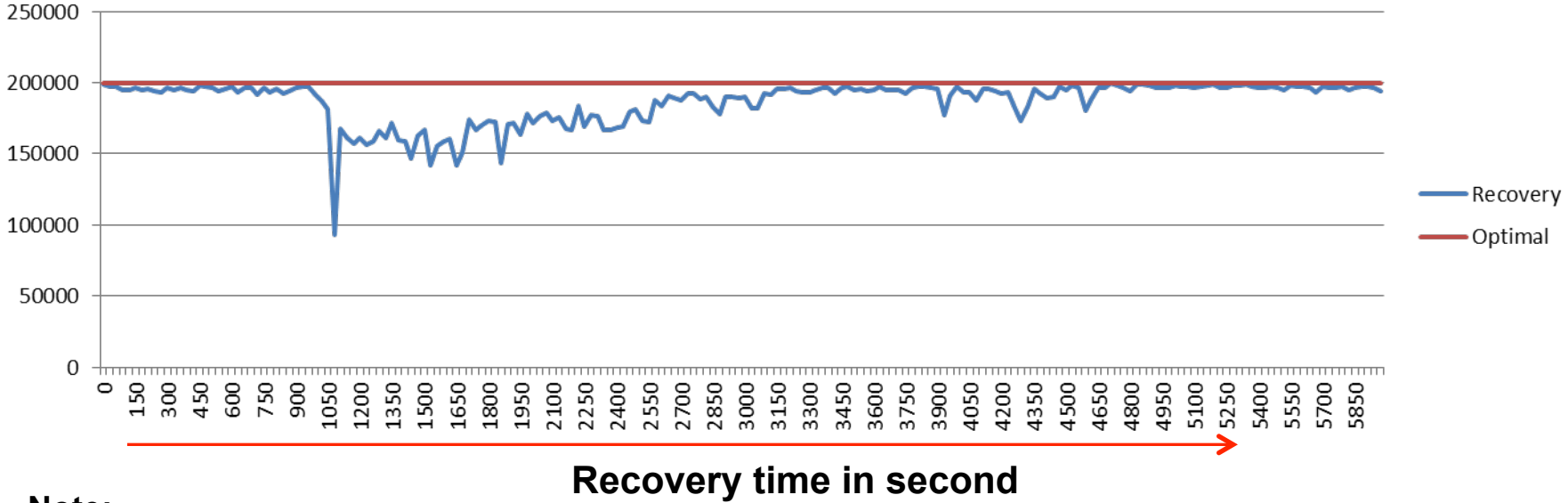
Note:

a. Recovery/rebalancing time is around 90 min when one osd down with IO load of 64k RR_qd64
(user data 10tb ; with 3 way replication total data on cluster is 30TB and each osd consists of around 1.2TB).

b. Recovery parameters with priority to client IO

c. Average perf degradation is around 15% when recovery is happening.

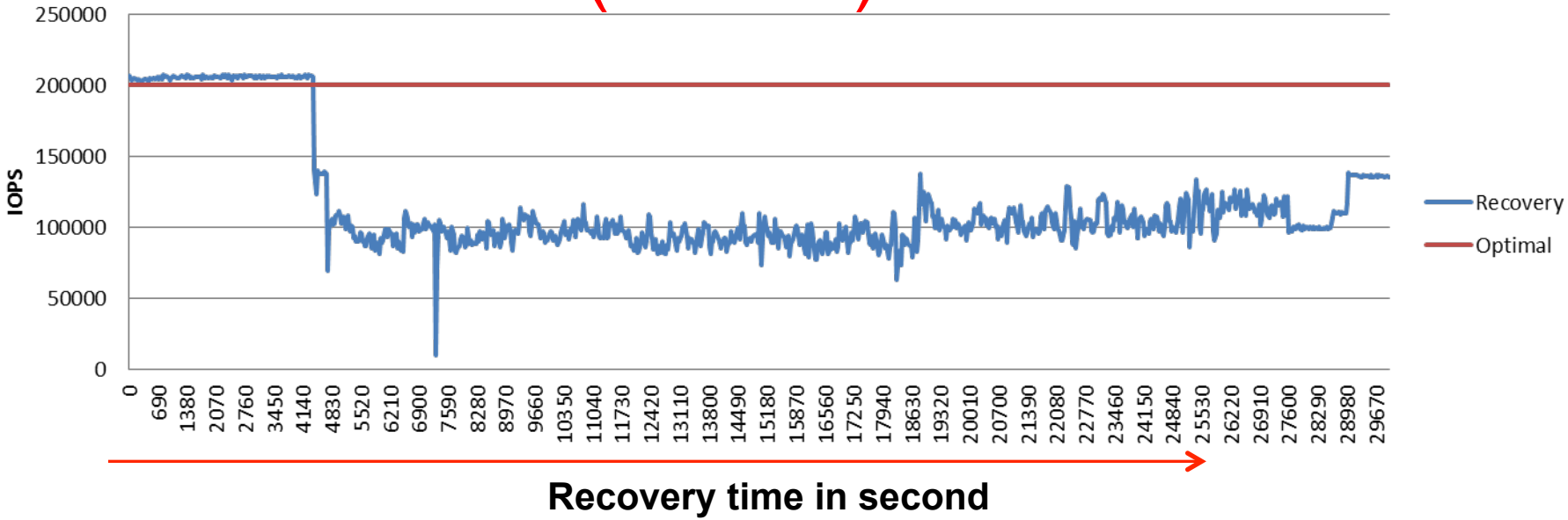
Client IO with Recovery after the ‘down’ OSD is ‘in’



Note:

a. Recovery/rebalancing time is around 60 min when one osd in with IO load of 64k RR_qd64
b. Average perf degradation is around 5% when recovery is happening.
c. Recovery parameters with priority to client IO

Client IO with Recovery after Entire Chassis (128 TB) Down/out



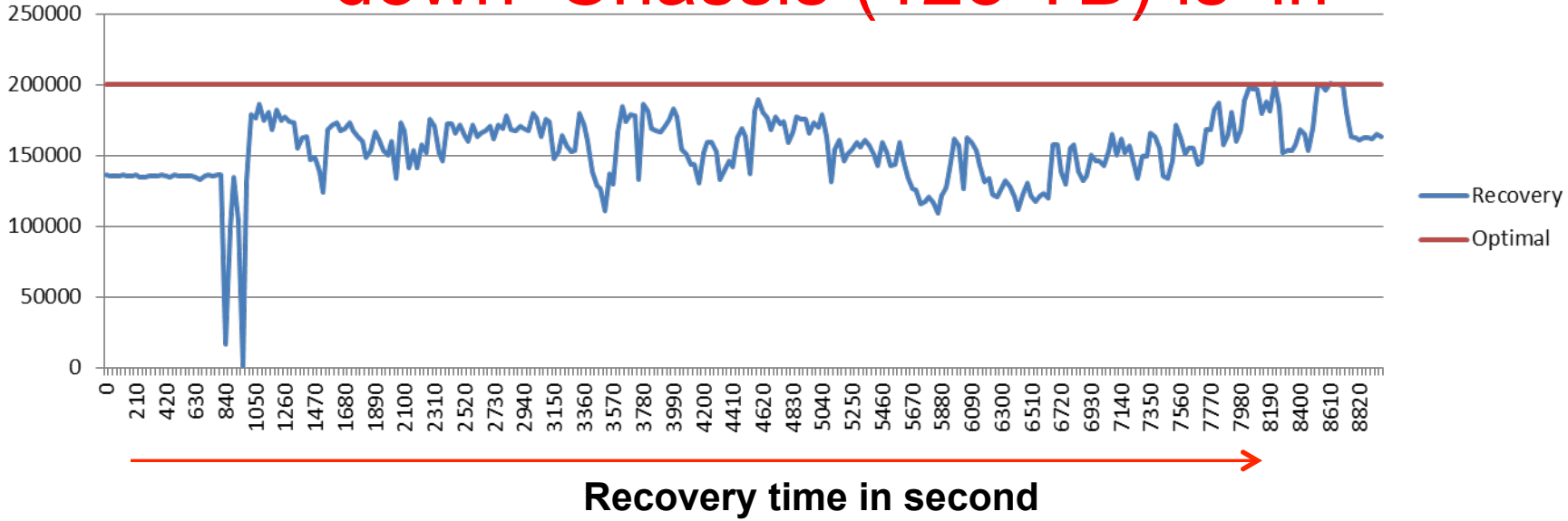
Note:

a. Recovery/rebalancing time is around 7 hours when one chassis down with IO load of 64k RR

b. User data 10tb ; with 3 way replication total data on cluster is 30tb and each chassis consists of around 10 TB).

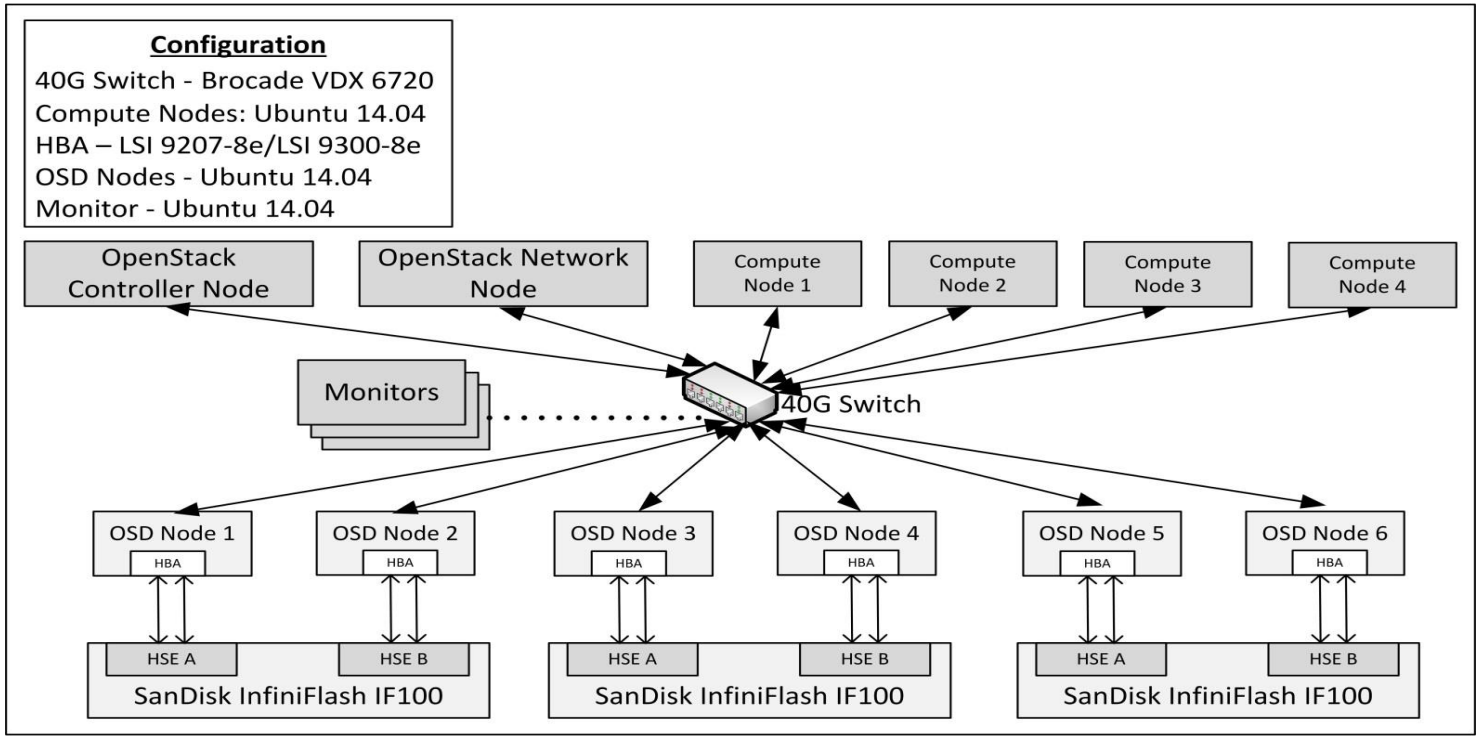
c. Recovery parameters with priority to client IO

Client IO with Recovery after Entire ‘down’ Chassis (128 TB) is ‘in’

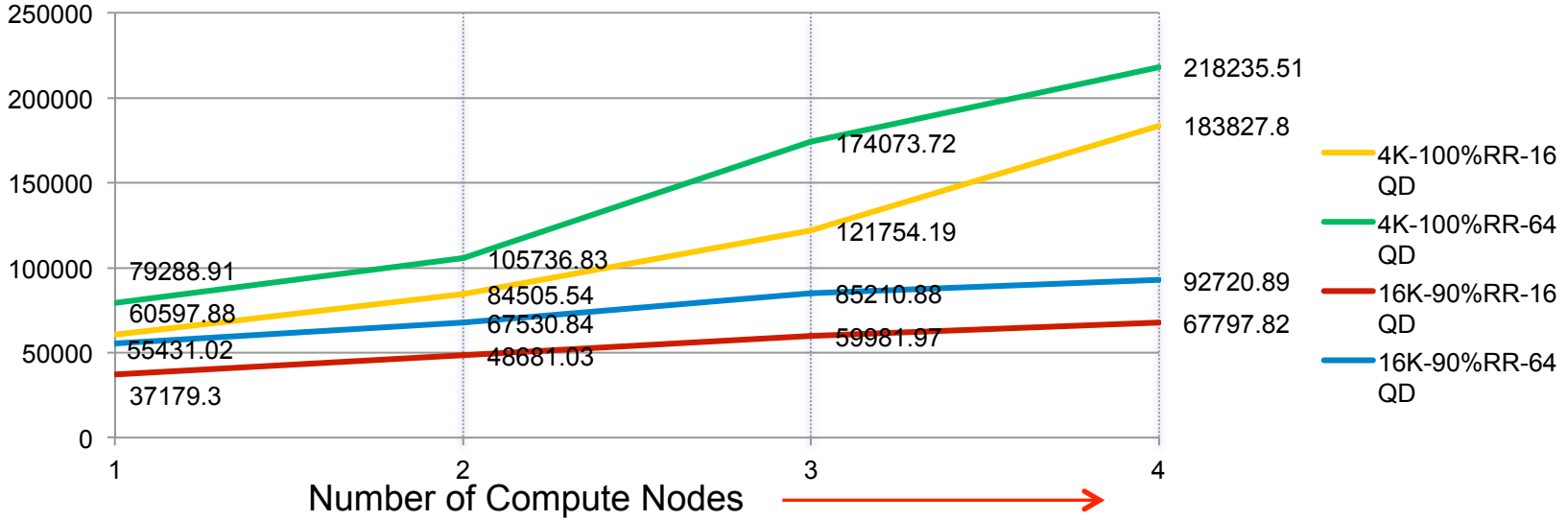


Note: Recovery/rebalancing time is around 2 hours after adding removed chassis with IO load of 64k RR_qd64 (user data 10tb ; with 3 way replication total data on cluster is 30tb and each chassis consists of around 10 TB).

InfiniFlash IF500 OpenStack Topology



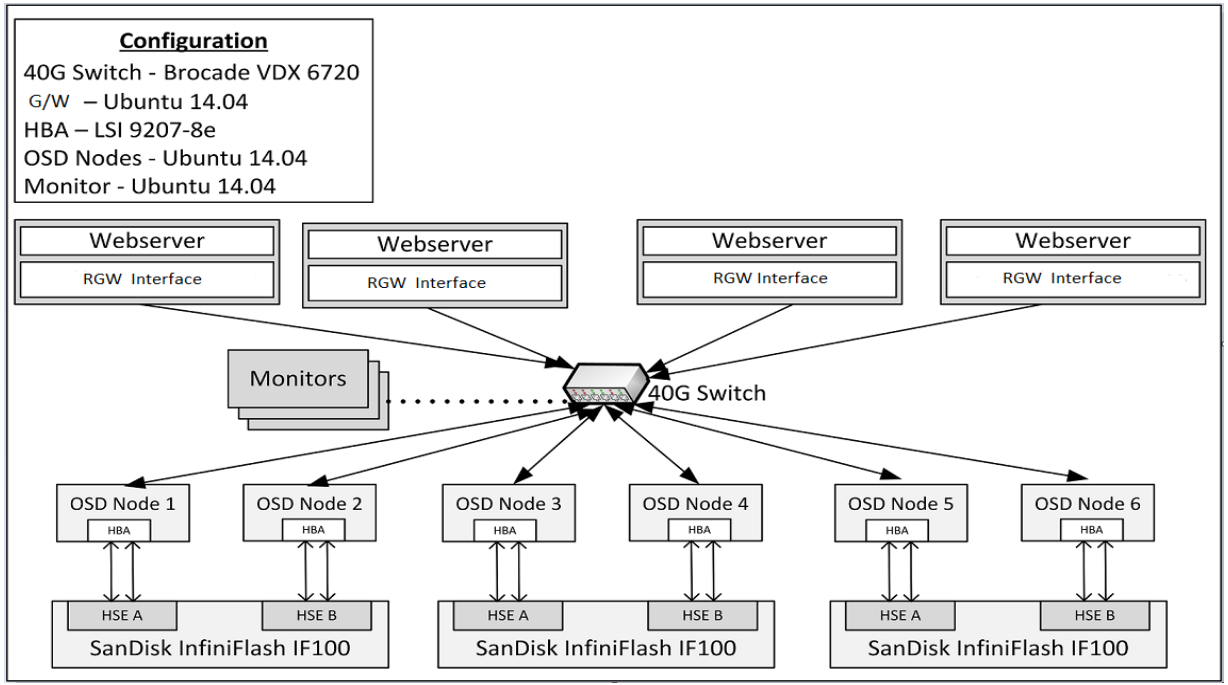
Performance Trends



VM configuration

- On single Compute node hosted 4 VMs and each vm is mapped four 1TB rbds.
- VMs root disks also from the same CEPH cluster.
- Each VM has 8 virtual CPU and 16GB RAM

InfiniFlash IF500 Object Storage Topology



Note: Clients and LBs are not shown in above topology. COSBench is used to generate the workload. Also, each Gateway server is running multiple RGW/WS combinations

Object Throughput with EC

Erasure Code Pool Configuration

- Host level Erasure coding 4:2 (k:m)
- Jerasure and technique is reed Solomon
- .rgw.buckets to be in Erasure coding pool others to be in replicated pool

Profile	Number of GW Servers	Total RGW Instances	COSBENCH Clients	Workers	BW(GB/s)	90% - Resp Time(ms)	Avg. CPU usage of OSD nodes(%)	Avg. CPU usage of RGW-CPU(%)	Avg. CPU usage of Client-CPU(%)	Avg.disk utilization of osds
4M_Write	4	16(4*4)	4	512	**2.86**	1,870	40	10	15	40
4M_Read	4	16(4*4)	4	512	**13.69**	300	55-60	40-45	70-75%	90-95

- Bandwidth of ~14GB/s achieved towards 4MB objects reads from 48 drives
- For writes ~3GB/s throughput achieved with host based EC pool configuration
- Network is saturating at around ~3.8GB/s from single Storage Node for **read** .

Write Performance with Scaling OSD

Write Object performance with EC pool scaling from 48 Drives to 96 drives

Profile	No of OSD Drives	No of RGW servers	No of RGW instances	Clients	Workers	BW(GB/s)	90% - RespTime (ms)	Avg. CPU usage of OSD nodes(%)	Avg. CPU usage of RGW-nodes(%)	Avg. CPU usage of COSBENCH-CPU(%)
4M_Write	48	4	16	4	512	2.86	1870	40	10	15
4M_Write	96	4	16	4	512	4.9	420	65-70	30-35	40-45

- With increase in OSD drives (2x) write performance improve by around 65-70%

Optimal EC Performance: Cauchy-good

Profile	No of RGW servers	No of RGW instances	Clients	Workers	BW(GB/s)	90% - RespTime (ms)	Avg. CPU usage of OSD nodes(%)	Avg. CPU usage of RGW-nodes(%)
4M_Write(ds:20 TB)	4	16	4	512	4.1	820	40-45	20-25
4M_Read (ds:20 TB)	4	16	4	512	14.33	220	50-55	50-52

- Cauchy-good Optimal write performance is at least 33% better Reed Solomon. In some runs got around 45% improvement
- Cauchy-good Optimal read performance 10% better however there was no much scope of improvement as we were already saturating the N/W bandwidth of RGW nodes.

Degraded EC Read Performance: Reed-Solomon vs Cauchy-good

	Cluster	Profile	BW(GB/s)	90% - RespTime(ms)	Avg. CPU usage of OSD nodes(%)
Cauchy-good	Degraded (1 Node Down)	4M_Read	12.9	230	71
Cauchy-good	Degraded (2 Node Down)	4M_Read	9.83	350	75-80
RS	Degraded (1 Node Down)	4M_Read	8.26	250	60
RS	Degraded (2 Node Down)	4M_Read	3.88	740	50-55

Cauchy-Good is doing far better than Reed Solomon technique for optimal and recovery IO performance as compared wrt write performance: getting almost double

InfiniFlash IF500 Future Goals

- Improved write performance
- Improved write amplification
- Improved mixed read-write performance
- Convenient end-to-end management/monitoring UI
- Easy deployment
- RDMA support
- EC with block interface

Flash Memory SUMMIT

Thank you ! Questions?

somnath.roy@sandisk.com

Visit SanDisk in Booth #207

@BigDataFlash
#bigdataflash
ITblog.sandisk.com
http://bigdataflash.sandisk.com