Unlock Bigdata Analytic Efficiency With Ceph Data Lake

Jian Zhang,
Yong Fu,

March, 2018

Agenda

- Background & Motivations
- The Workloads, Reference Architecture Evolution and Performance Optimization
- Performance Comparison with Remote HDFS
- Summary & Next Step

BACKGROUND AND MOTIVATION

Challenges of scaling Hadoop* Storage

BOUNDED Storage and Compute resources on Hadoop Nodes brings challenges

Data Capacity

Silos

Costs

Performance & efficiency

Typical Challenges

Data/Capacity	Multiple Storage Silos
Space, Spent, Power, Utilization	Upgrade Cost
Inadequate Performance	Provisioning And Configuration

Source: **451 Research**, Voice of the Enterprise: Storage Q4 2015

Options To Address The Challenges

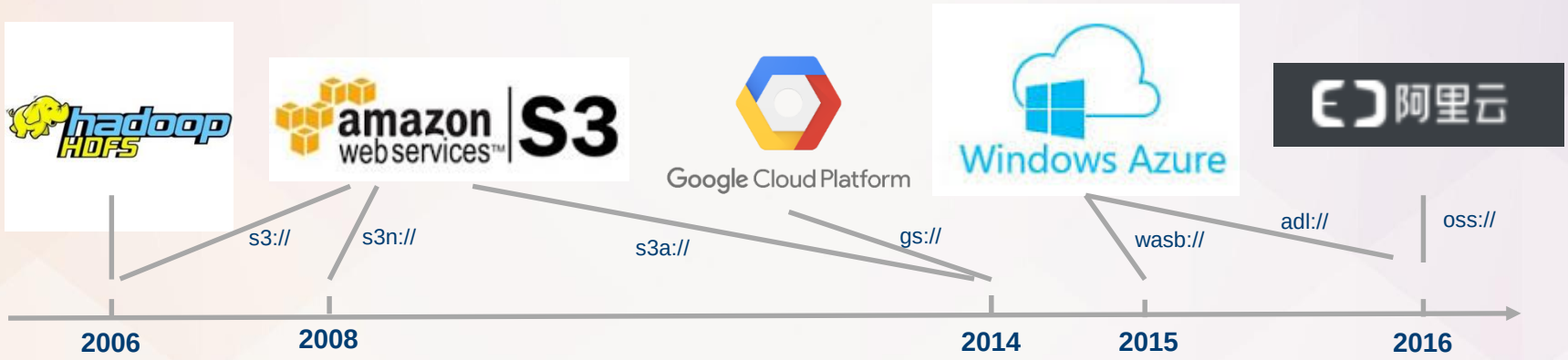
Large Cluster	More Clusters	Compute and Storage Disaggregation
• Lacks isolation - noisy neighbors hinder SLAs • Lacks elasticity - rigid cluster size • Can't scale compute/storage costs separately	• Cost of duplicating datasets across clusters • Lacks on-demand provisioning • Can't scale compute/storage costs separately	• Isolation of high-priority workloads • Shared big datasets • On-demand provisioning • compute/storage costs scale separately

Compute and Storage disaggregation provides Simplicity, Elasticity, Isolation

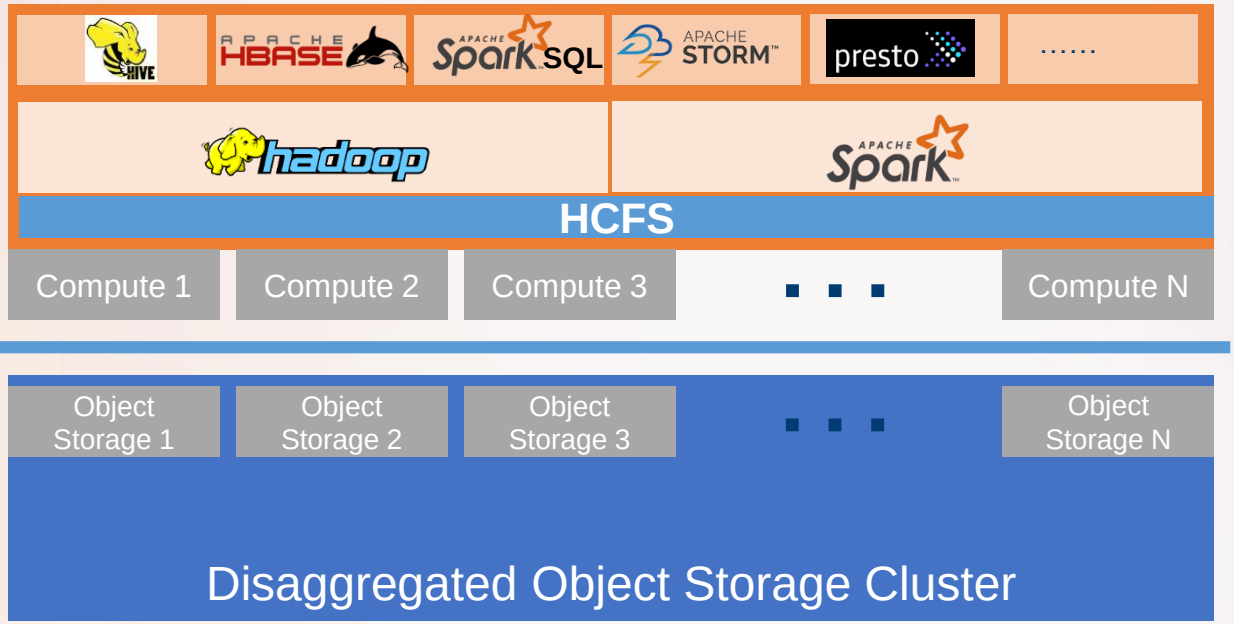
ceph Unified Hadoop* File System and API for cloud storage

Spark HIVE hadoop MapReduce IMPALA presto

Hadoop Compatible File System abstraction layer: Unified storage API interface Hadoop fs –ls s3a://job/



Proposal: Apache Hadoop* with disagreed Object Storage



Hadoop Services
- Virtual Machine
- Container
- Bare Metal

Object Storage Services
- Co-located with gateway
- Dynamic DNS or load balancer
- Data protection via storage replication or erasure code
- Storage tiering

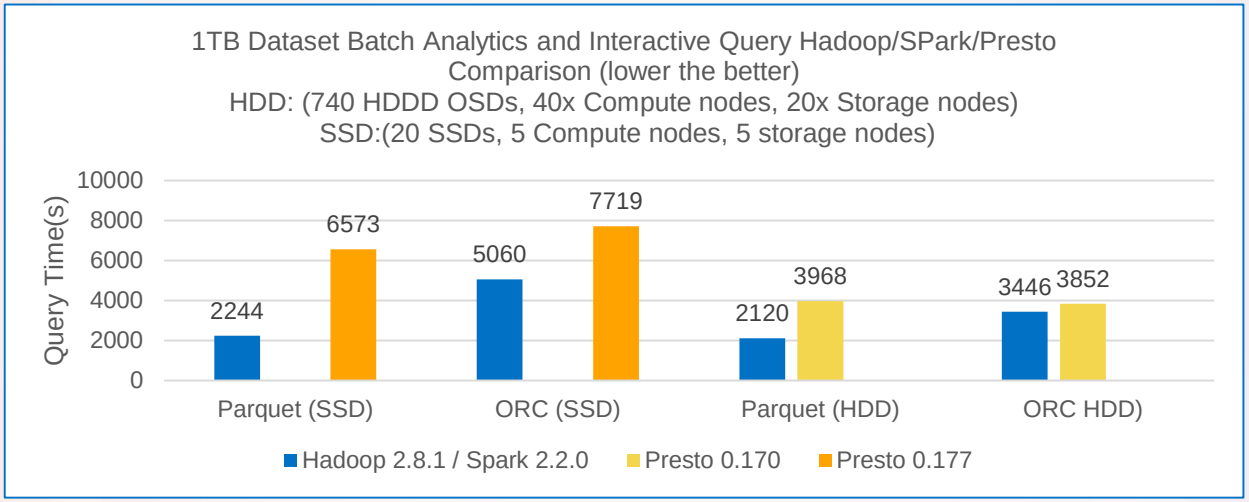
THE WORKLOADS , REFERENCE ARCHITECTURE AND PERFORMANCE OVERVIEW

Workloads

ceph

- **Simple Read/Write**
  - DFSIO: TestDFSIO is the canonical example of a benchmark that attempts to measure the Storage's capacity for reading and writing bulk data.
  - Terasort: a popular benchmark that measures the amount of time to sort one terabyte of randomly distributed data on a given computer system.
- **Data Transformation**
  - ETL: Taking data as it is originally generated and transforming it to a format (Parquet, ORC) that more tuned for analytical workloads.
- **Batch Analytics**
  - To consistently executing analytical process to process large set of data.
  - Leveraging 54 derived from TPC-DS * queries with intensive reads across objects in different buckets

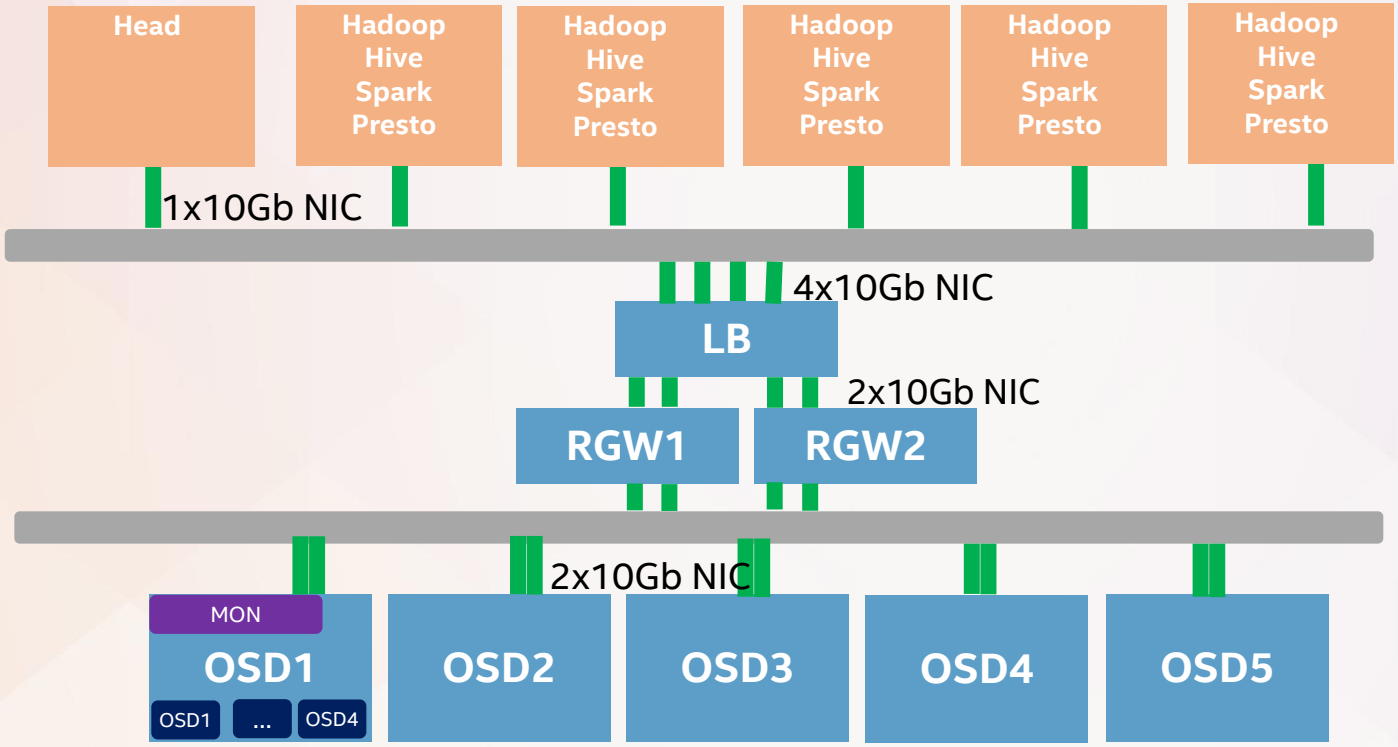
Bigdata on Object Storage Performance overview
--Batch analytics



- Significant performance improvement from Hadoop 2.7.3/Spark 2.1.1 to Hadoop 2.8.1/Spark 2.2.0 (improvement in s3a)
- Batch analytics performance of 10-node Intel AFA is almost on-par with 60-node HDD cluster

Hardware Configuration & Topology

--Dedicate LB



5x Compute Node

- Intel® Xeon™ processor E5-2699 v4 @ 2.2GHz, 64GB mem
- 2x10G 82599 10Gb NIC
- 2x SSDs
- 3x Data storage (can be emliminated)

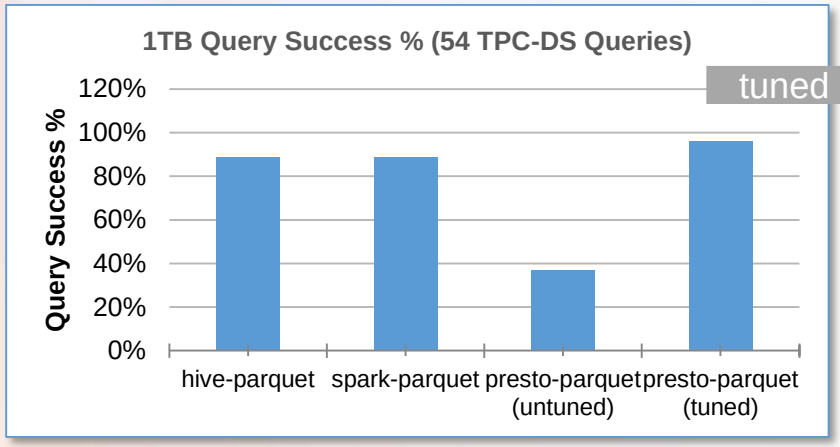
Software:

- Hadoop 2.7.3
- Spark 2.1.1
- Hive 2.2.1
- Presto 0.177
- RHEL7.3

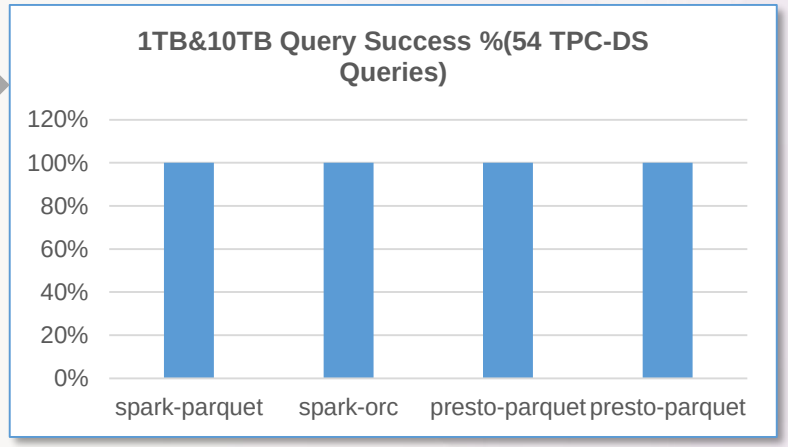
5x Storage Node, 2 RGW nodes, 1 LB nodes

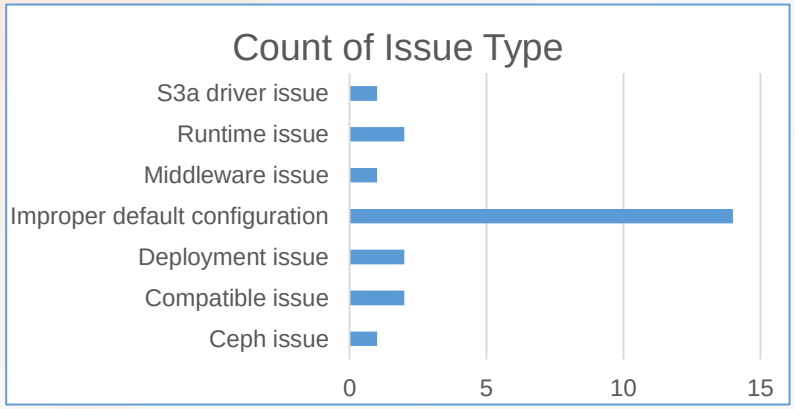
- Intel(R) Xeon(R) CPU E5-2699v4 2.20GHz
- 128GB Memory
- 2x 82599 10Gb NIC
- 1x Intel® P3700 1.0TB SSD as WAL and rocksdb
- 4x 1.6TB Intel® SSD DC S3510 as data drive
- 2x 400G S3700 SSDs
- 1 OSD instances one each S3510 SSD
- RHEl7.3
- RHCS 2.3

Improve Query Success Ratio with trouble-shootings



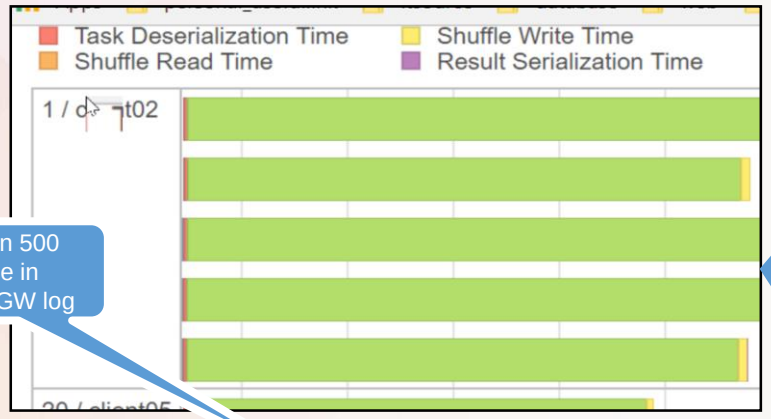
tuned





- 100% selected TPC-DS query passed with tunings
- Improper Default configuration
  - small capacity size,
  - wrong middleware configuration
  - improper Hadoop/Spark configuration for different size and format data issues

Optimizing HTTP Requests
-- The bottlenecks



Compute time take the big part. (compute time = read data +sort)

Return 500 code in RadosGW log

New connections out every time, Connection not reused

```
2017-07-18 14:53:52.259976 7fddd[illegible]fc700  1 ====== starting new request req=0x7fddd67[illegible]
2017-07-18 14:53:52.271829 7fddd5ffb[illegible]  1 ====== starting new request req=0x7fddd5ff5710 =====
2017-07-18 14:53:52.273940 7fddd7fff700  0 ERROR: flush_read_list(): d->client_c->handle_data() returned -5
2017-07-18 14:53:52.274223 7fddd7fff700  0 WARNING: set_req_state_err err_no=5 resorting to 500
2017-07-18 14:53:52.274253 7fddd7fff700  0 ERROR: s->cio->send_content_length() returned err=-5
2017-07-18 14:53:52.274257 7fddd7fff700  0 ERROR: s->cio->print() returned err=-5
2017-07-18 14:53:52.274258 7fddd7fff700  0 ERROR: STREAM_IO(s)->print() returned err=-5
2017-07-18 14:53:52.274267 7fddd7fff700  0 ERROR: STREAM_IO(s)->complete_header() returned err=-5
```

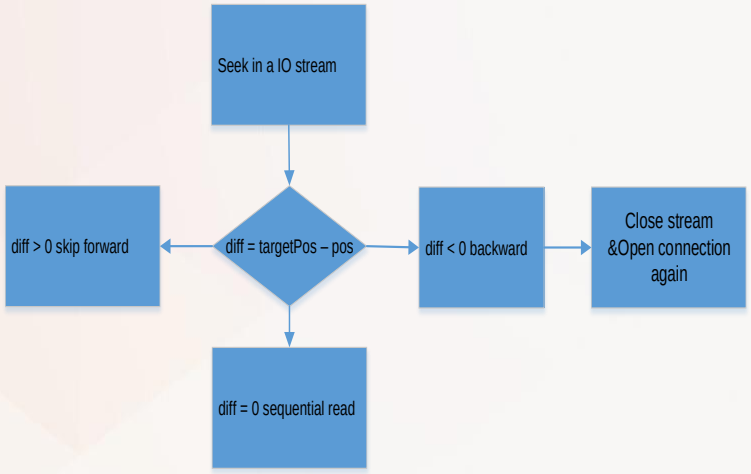
```
ESTAB      0      0       ::ffff:10.0.2.36:44446          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44454          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44374          ::ffff:10.0.2.254:80
ESTAB      159724 0       ::ffff:10.0.2.36:44436          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44448          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44338          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44438          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44414          ::ffff:10.0.2.254:80
ESTAB      0      480     ::ffff:10.0.2.36:44450          ::ffff:10.0.2.254:80
                          timer:(on,170ms,0)
ESTAB      0      0       ::ffff:10.0.2.36:44442          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44390          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44326          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44452          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44394          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44444          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44456          ::ffff:10.0.2.254:80      2
                          seconds interval ==========================
ESTAB      0      0       ::ffff:10.0.2.36:44508          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44476          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44524          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44374          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44500          ::ffff:10.0.2.254:80
           0      0       ::ffff:10.0.2.36:44504          ::ffff:10.0.2.254:80
           0      0       ::ffff:10.0.2.36:44512          ::ffff:10.0.2.254:80
           0      0       ::ffff:10.0.2.36:44506          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44464          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44518          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44510          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44442          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44526          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44472          ::ffff:10.0.2.254:80
ESTAB      0      0       ::ffff:10.0.2.36:44466          ::ffff:10.0.2.254:80
```

Unresued connection cause high read time and block performance

Optimizing HTTP Requests
-- S3a input policy

Background

- The S3A filesystem client supports the notion of input policies, similar to that of the POSIX fadvise() API call. This tunes the behavior of the S3A client to optimize HTTP GET requests for various use cases. To optimize HTTP GET requests, you can take advantage of the S3A experimental input policy fs.s3a.experimental.input.fadvise.
- **Ticket**: https://issues.apache.org/jira/browse/HADOOP-13203



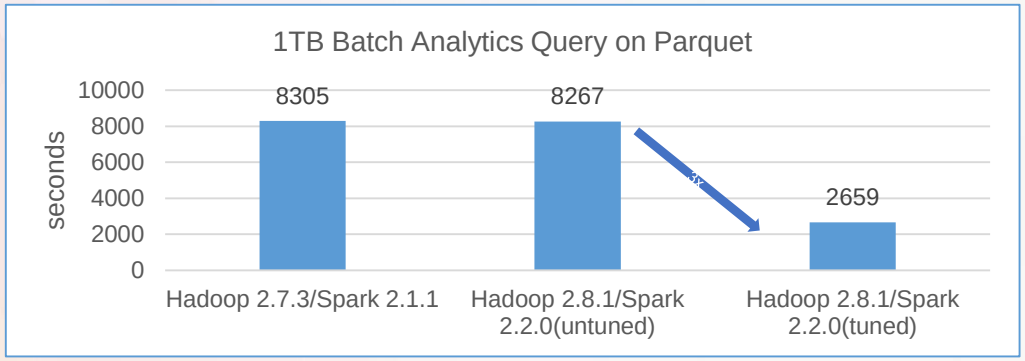
Solution

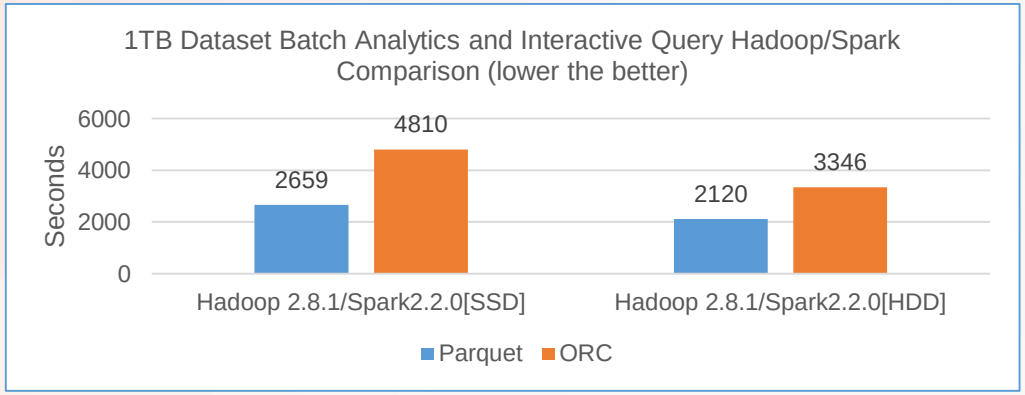
Enable random read policy in core-site.xml:

```
<property>
   <name>fs.s3a.experimental.input.fadvise</name>
   <value>random</value>
 </property>
 <property>
   <name>fs.s3a.readahead.range</name>
   <value>64K</value>
 </property>
```

- By reducing the cost of closing existing HTTP requests, this is highly efficient for file IO accessing a binary file through a series of `PositionedReadable.read()` and `PositionedReadable.readFully()` calls.

Optimizing HTTP Requests -- Performance

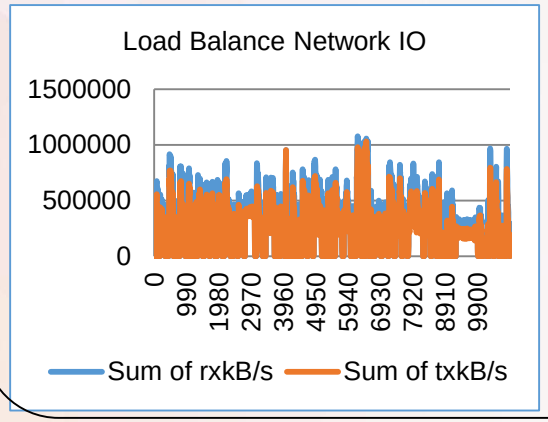


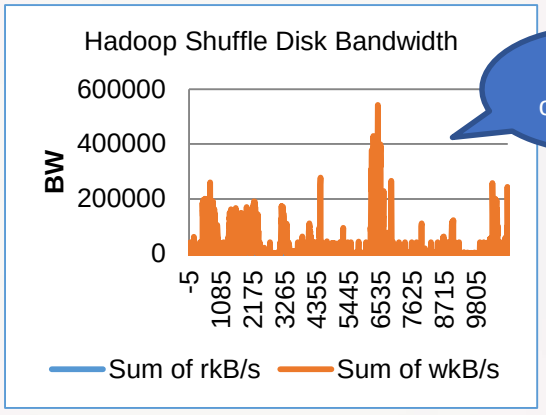


- Readahead feature support from Hadoop 2.8.1, but not enabled by default. Apply random read policy, 500 issue gone and performance improved 3x than before
- All Flash storage architecture also show great performance benefit and low TCO which compared with HDD storage

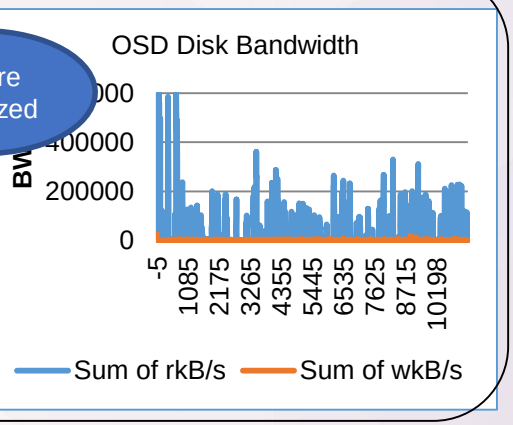
Optimizing HTTP Requests

--Resource Utilization Comparison

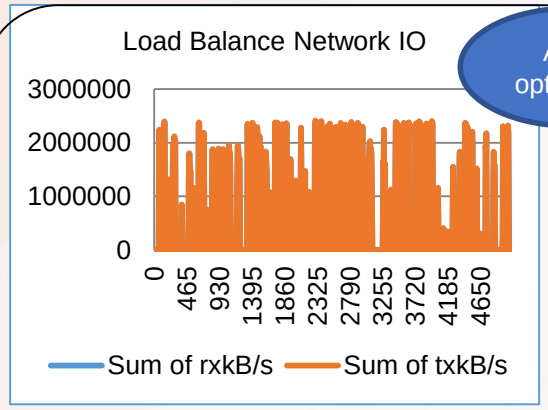


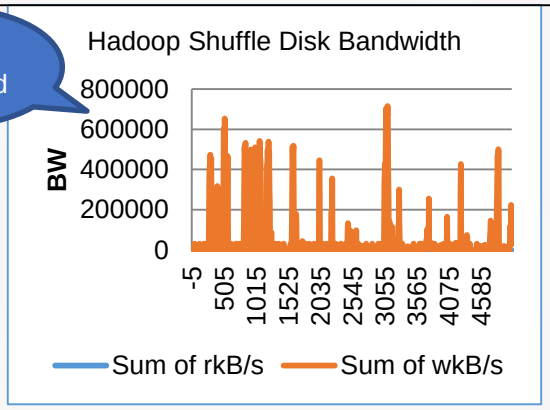


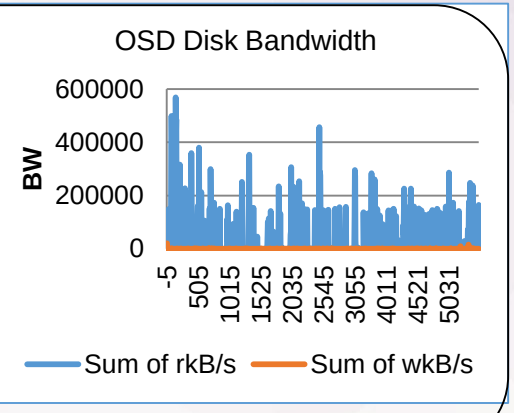
Before optimized



After optimized



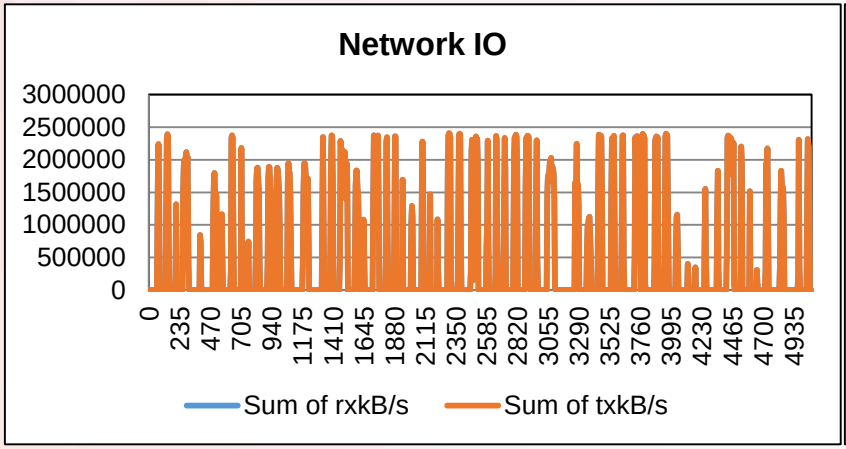


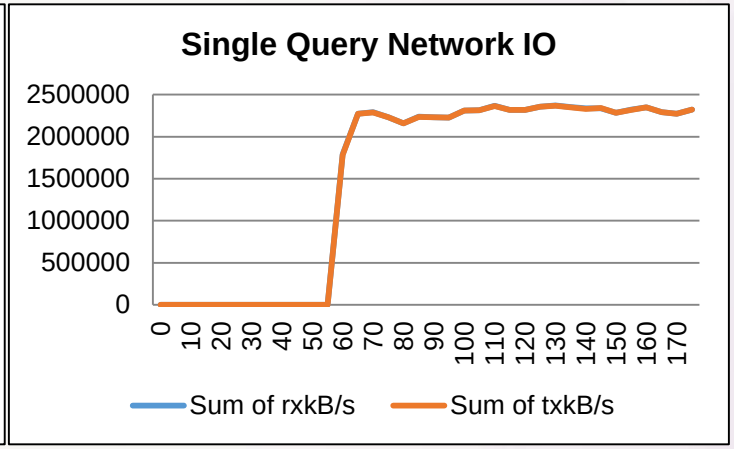


16

Resolving RGW BW bottleneck

--The bottlenecks

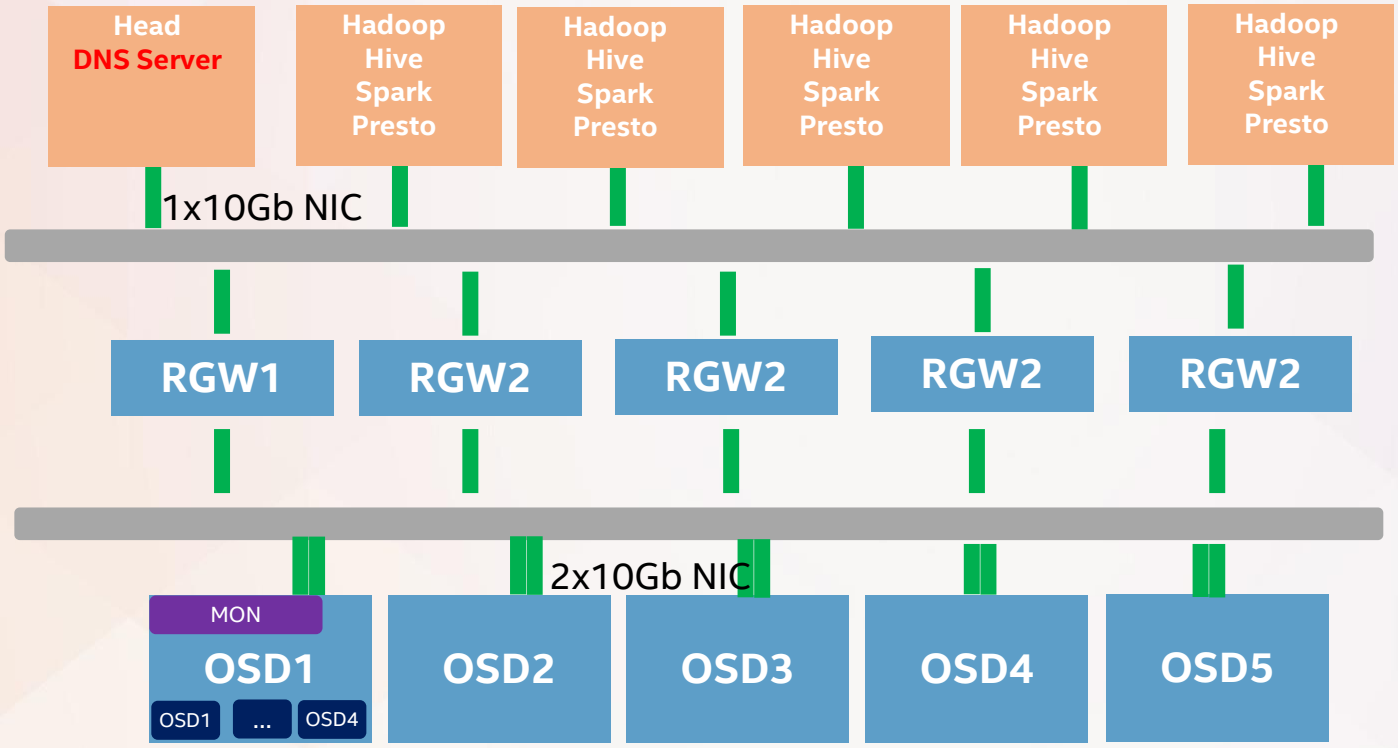




- LB BW has became the bottleneck
- Observed many messages blocked at load balancer server(send to s3a driver), but not much blocked at receiving on s3a driver side

Hardware Configuration

--No LB with more RGWs and round-robin DNS



5x Compute Node

- Intel® Xeon™ processor E5-2699 v4 @ 2.2GHz, 64GB mem
- 2x10G 82599 10Gb NIC
- 2x SSDs
- 3x Data storage (can be emliminated)

Software:

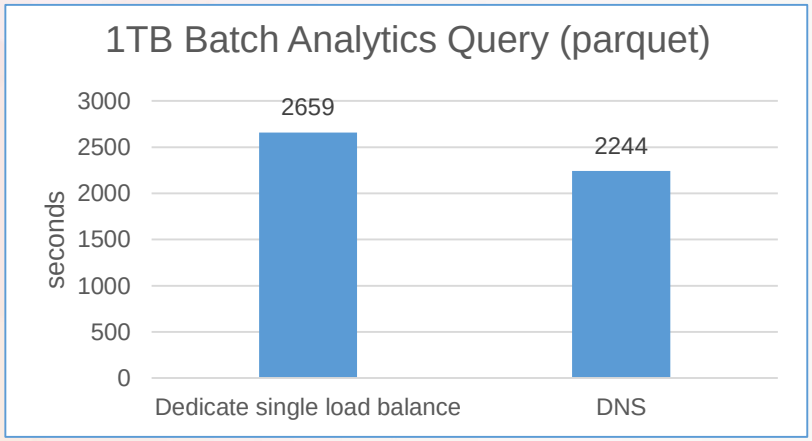
- Hadoop 2.7.3
- Spark 2.1.1
- Hive 2.2.1
- Presto 0.177
- RHEL7.3

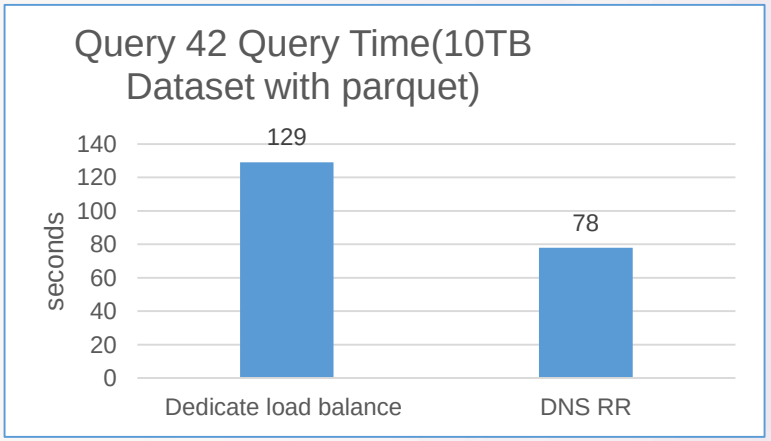
5x Storage Node, 2 RGW nodes, 1 LB nodes

- Intel(R) Xeon(R) CPU E5-2699v4 2.20GHz
- 128GB Memory
- 2x 82599 10Gb NIC
- 1x Intel® P3700 1.0TB SSD as WAL and rocksdb
- 4x 1.6TB Intel® SSD DC S3510 as data drive
- 2x 400G S3700 SSDs
- 1 OSD instances one each S3510 SSD
- RHEl7.3
- RHCS 2.3

Performance evaluation

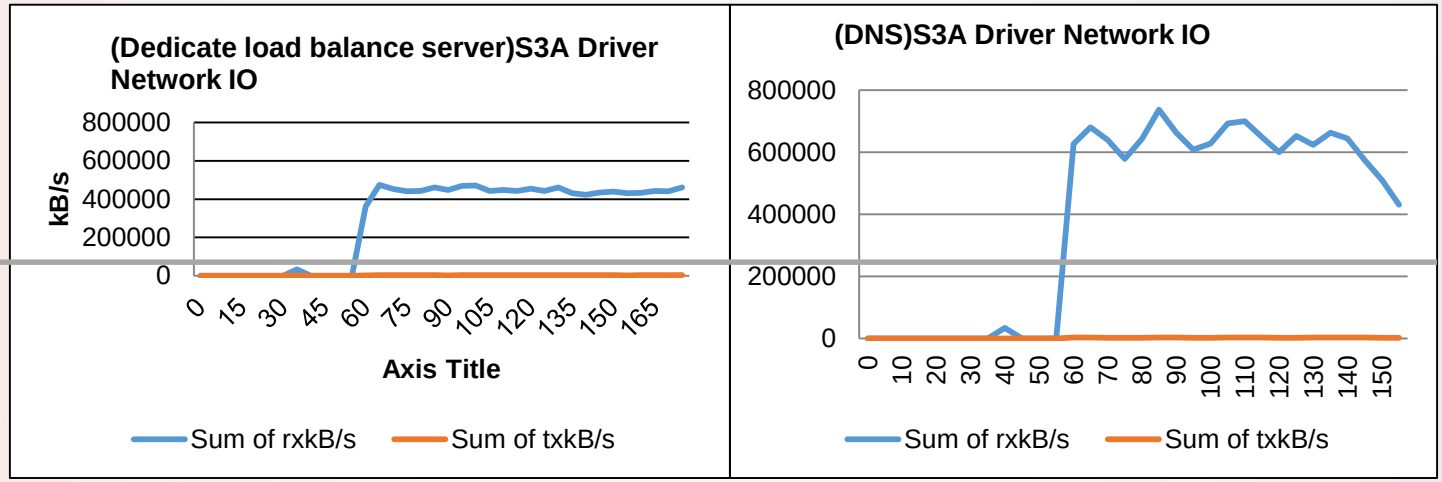
--No LB with more RGWs and round-robin DNS





- 18% performance improvement with more RGWs and round-robin DNS
- Query42(has less shuffle) **is 1.64x faster** in the new architecture

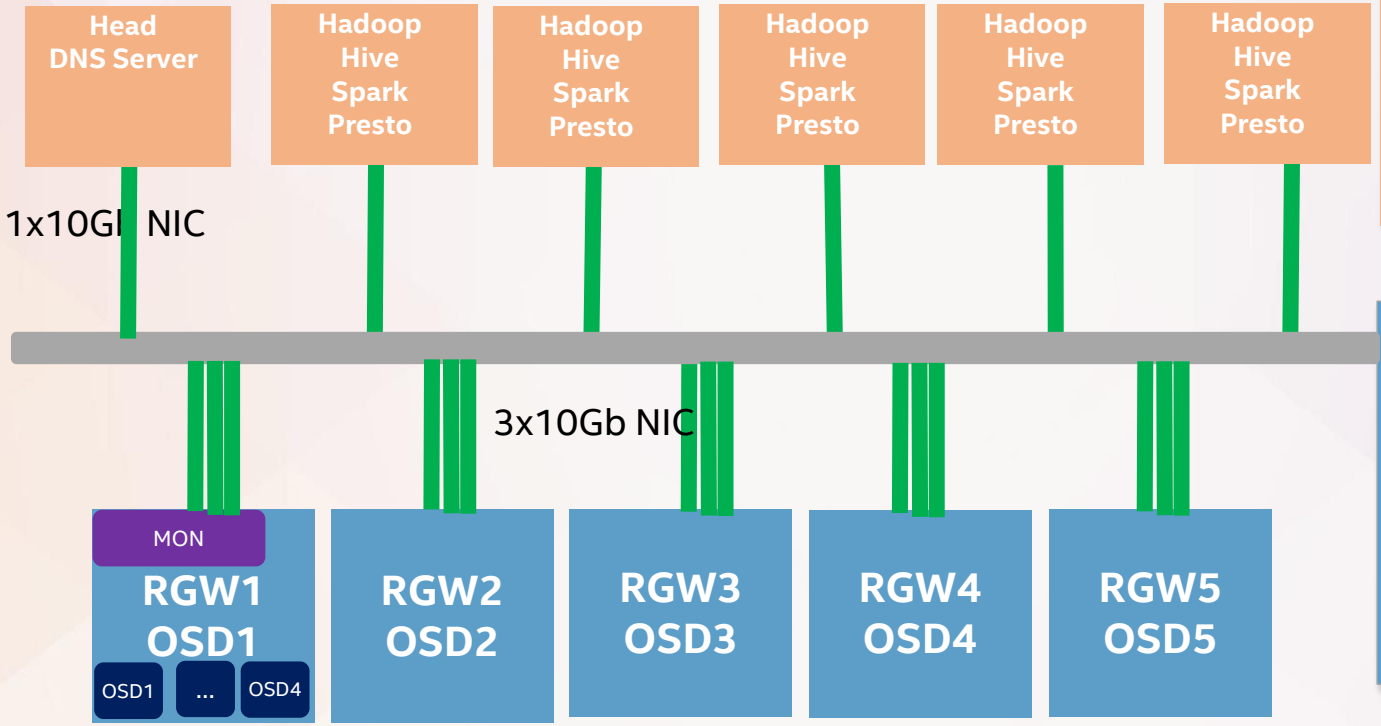
Key Resource Utilization Comparison



- Compute side(Hadoop s3a driver) can read more data from OSD, which represent DNS deployment really can gain network throughput performance than single gateway with bonding/teaming technology

Hardware Configuration

--RGW and OSD Collocated



5x Compute Node

- Intel® Xeon™ processor E5-2699 v4 @ 2.2GHz, 64GB mem
- 2x10G 82599 10Gb NIC
- 2x SSDs
- 3x Data storage (can be emliminated)

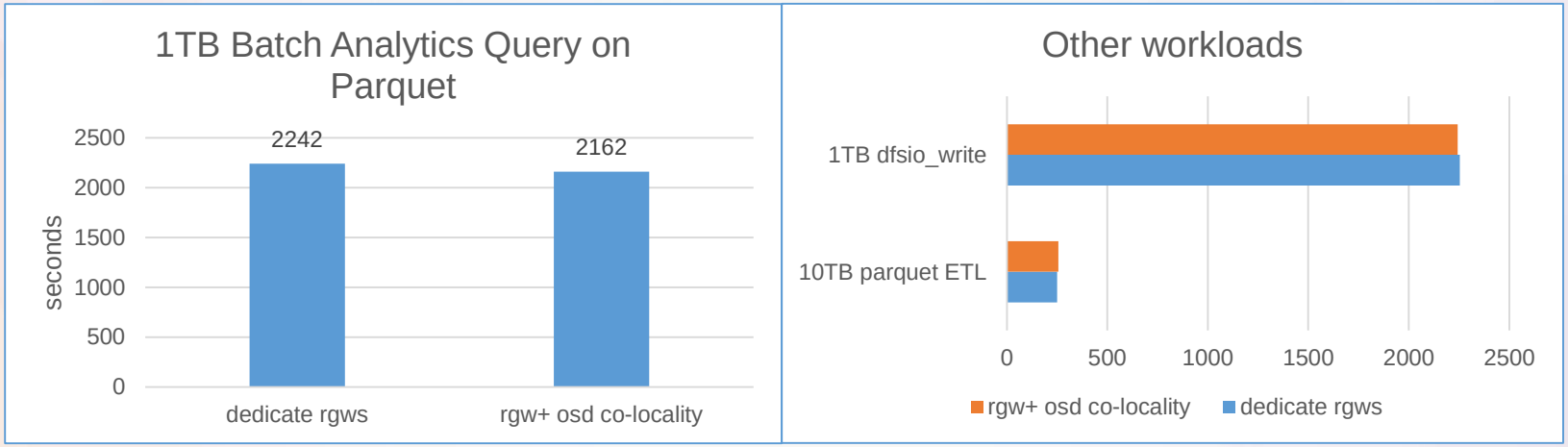
Software:

- Hadoop 2.7.3
- Spark 2.1.1
- Hive 2.2.1
- Presto 0.177
- RHEL7.3

5x Storage Node, 2 RGW nodes, 1 LB nodes

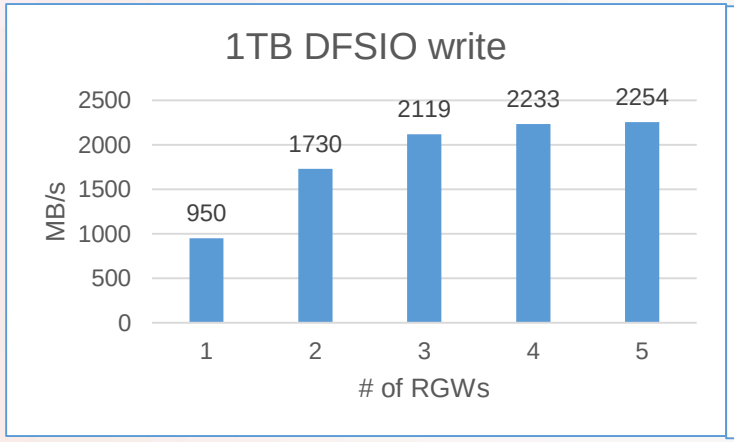
- Intel(R) Xeon(R) CPU E5-2699v4 2.20GHz
- 128GB Memory
- 2x 82599 10Gb NIC
- 1x Intel® P3700 1.0TB SSD as WAL and rocksdb
- 4x 1.6TB Intel® SSD DC S3510 as data drive
- 2x 400G S3700 SSDs
- 1 OSD instances one each S3510 SSD
- RHEl7.3
- RHCS 2.3

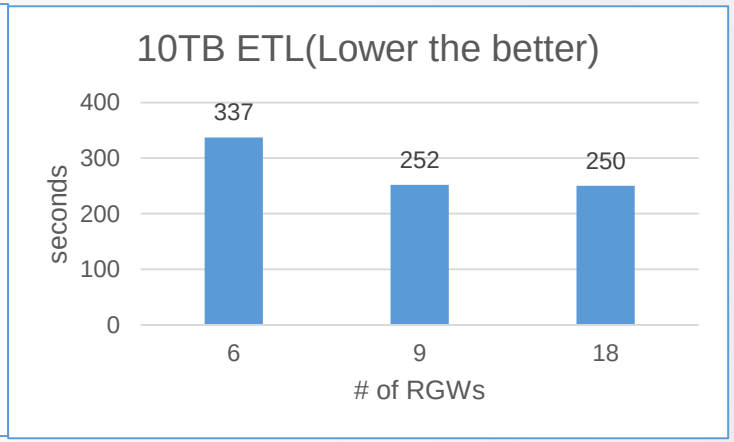
Performance evaluation with RGW & OSD collocated



- No need extra dedicate RGW servers, RGW instance and OSD go through different network interface by enable ECMP
- No performance degradation, but **more less TCO**

RGW & OSD collocated – RGW scaling

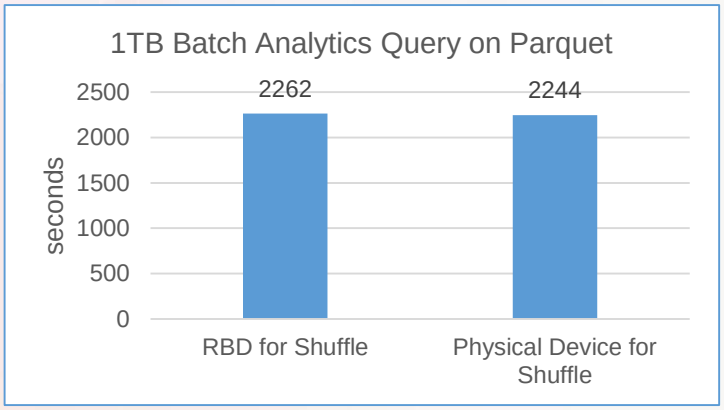


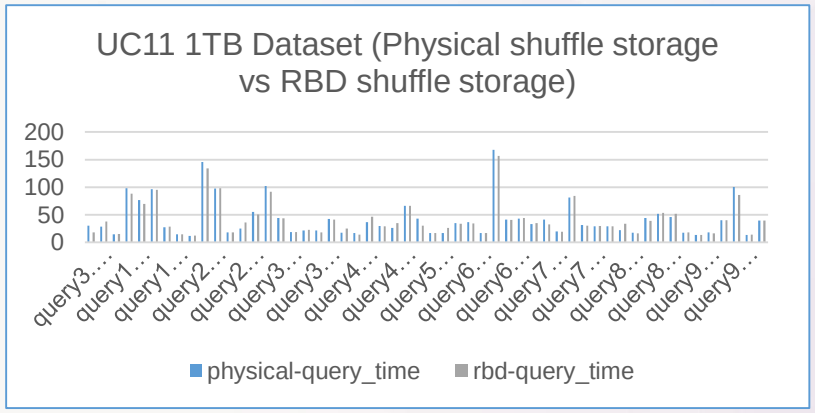


- Scale out RGWs can improve performance before OSD(storage) saturating
- So How many RGWs can win the best performance should be decided by the bandwidth of each RGW server and throughput of OSDs

Using Ceph RBD as shuffle Storag

-- Eliminate the physical drive on the compute!

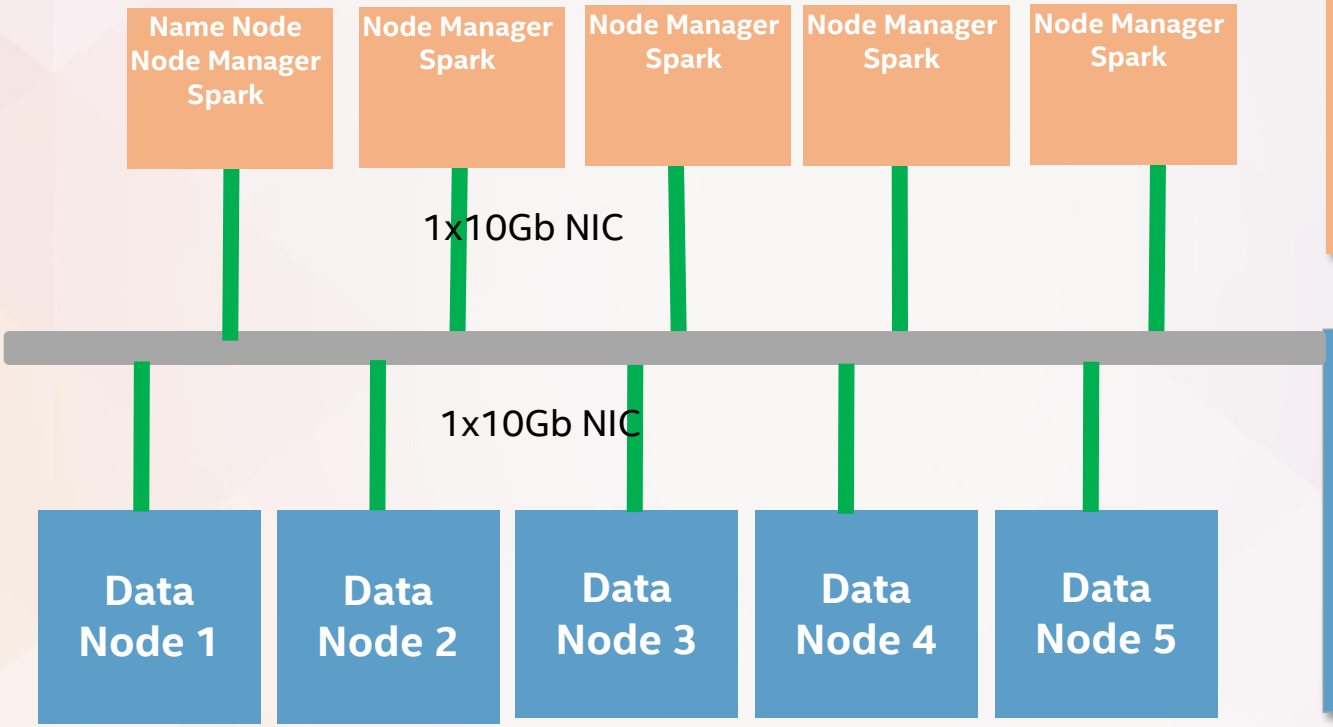




- Mount two RBDs on compute node remotely instead of physical shuffle device
- Ideally, the latency on remote RBDs larger than local physical device, but the bandwidth of remote RBD is not smaller than local physical device too
- So final performance of a TPC-DS query set on RBDs maybe close or even better than on physical device while there are heavy shuffles

PERFORMANCE COMPARISON WITH REMOTE HDFS

Hardware Configuration
--Remote HDFS

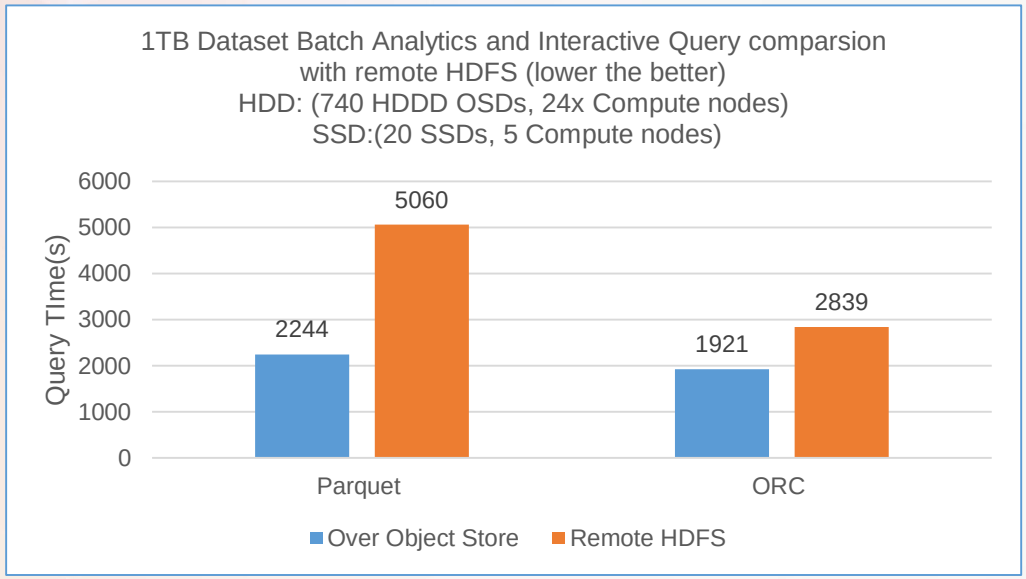


5x Compute Node
- Intel® Xeon™ processor E5-2699 v4 @ 2.2GHz, 64GB mem
- 2x10G 82599 10Gb NIC
- 2x S3700 as shuffle storage

Software:
- Hadoop 2.7.3
- Spark 2.1.1
- Hive 2.2.1
- Presto 0.177
- RHEL7.3

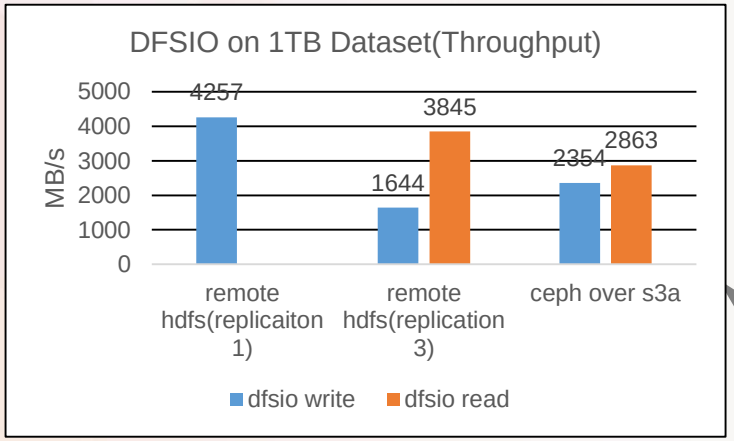
5x Data Node
- Intel(R) Xeon(R) CPU E5-2699v4 2.20GHz
- 128GB Memory
- 2x 82599 10Gb NIC
- 1x Intel® P3700 1.0TB SSD as WAL and rocksdb
- 7x 400G S3700 SSDs as data stire
- RHEl7.3
- RHCS 2.3

Bigdata on Cloud vs. Remote HDFS
--Batch Analytics



- On-par performance compared with remote HDFS
  - With optimizations, bigdata analytics on object storage is onpar with remote, especially on parquet format data
  - performance of s3a driver close to native dfsclient , and demonstrate compute and storage separate solution has a considerable performance compare with combination solution

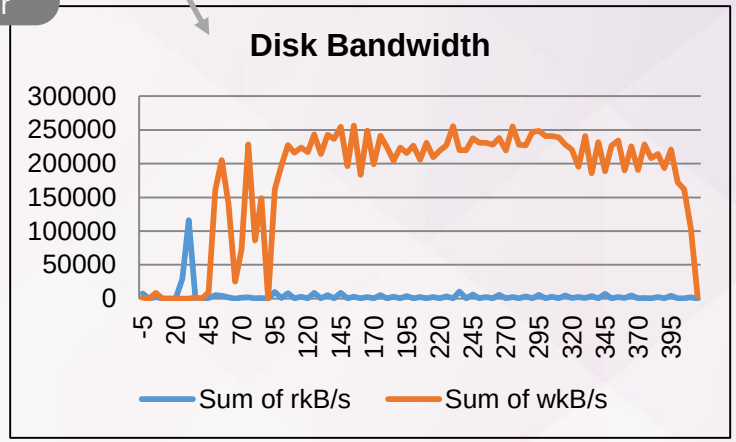
Bigdata on Cloud vs. Remote HDFS
--DFSIO



DFSIO write performance on Ceph is better than remote hdfs(43%), but read performance is 34% lower

Device:	rrqm/s	wrqm/s	r/s	w/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util
nvme0n1	0.00	0.00	0.00	4943.00	0.00	316696.00	128.14	0.27	0.06	0.00	0.06	0.03	13.25
sdb	0.00	0.00	5.00	100.50	20.00	15076.00	286.18	0.09	0.82	0.10	0.85	0.48	5.10
sdc	0.00	0.00	5.50	118.50	52.00	19969.50	322.93	0.13	1.03	0.09	1.08	0.62	7.70
sdh	0.00	0.00	4.50	120.50	108.00	17768.00	286.02	0.12	0.92	0.33	0.95	0.54	6.70
sda	0.00	0.00	23.50	474.00	274.00	66450.00	268.24	0.82	1.65	1.60	1.66	0.83	41.20
sdd	0.00	0.00	22.00	455.50	208.00	61194.00	257.18	2.59	5.43	4.89	5.45	1.94	92.70
sdg	0.00	0.00	10.00	461.50	100.00	102906.00	436.93	83.39	176.86	15.95	180.35	2.04	96.15
sdf	0.00	43.50	19.50	922.50	168.00	110178.00	234.28	7.49	7.95	2.26	8.07	0.54	50.55
sdi	0.00	0.00	0.00	18.00	0.00	931.75	103.53	0.10	5.56	0.00	5.56	4.86	8.75
sde	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

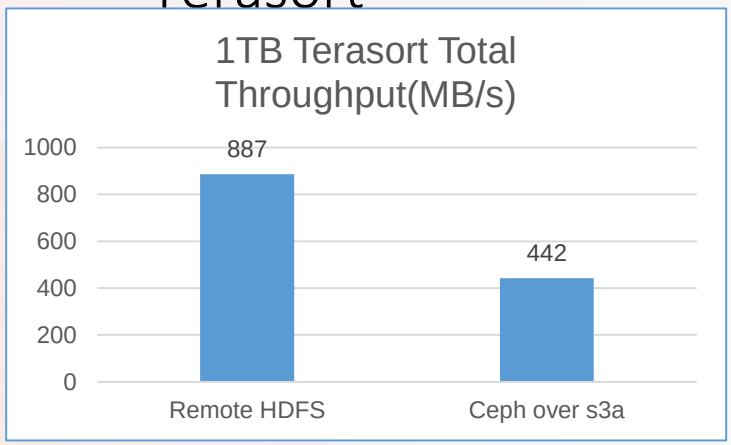
Write to Ceph hit disk bottleneck



Shuffle storage block at consuming data from Data Lake

ESTAB	2135680 0	10.0.2.16:8080	10.0.2.32:48046	timer:(keepalive,119min,0)
ESTAB	2135680 0	10.0.2.16:8080	10.0.2.32:47974	timer:(keepalive,119min,0)
ESTAB	2135680 0	10.0.2.16:8080	10.0.2.32:48006	timer:(keepalive,119min,0)
ESTAB	562816 0	10.0.2.16:8080	10.0.2.32:47956	timer:(keepalive,119min,0)
ESTAB	0 0	10.0.2.16:8080	10.0.2.32:47972	timer:(keepalive,119min,0)
ESTAB	2135680 0	10.0.2.16:8080	10.0.2.32:47978	timer:(keepalive,119min,0)
ESTAB	1611392 0	10.0.2.16:8080	10.0.2.32:48012	timer:(keepalive,119min,0)
ESTAB	2135680 0	10.0.2.16:8080	10.0.2.32:47942	timer:(keepalive,119min,0)
ESTAB	0 0	10.0.2.16:8080	10.0.2.32:47992	timer:(keepalive,119min,0)
ESTAB	0 0	10.0.2.16:8080	10.0.2.32:48030	timer:(keepalive,119min,0)
ESTAB	562816 0	10.0.2.16:8080	10.0.2.32:48004	timer:(keepalive,119min,0)

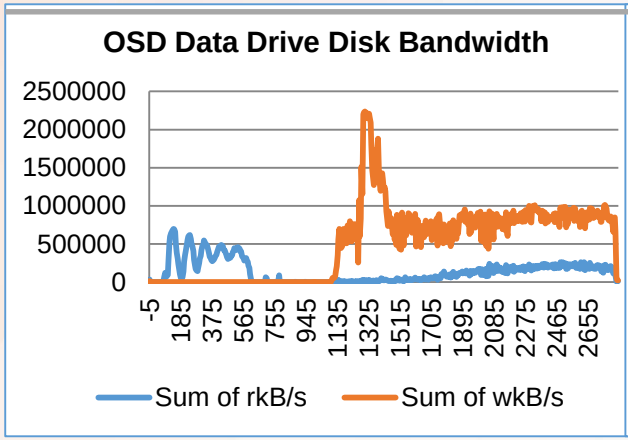
Bigdata on Cloud vs. Remote HDFS --Terasort

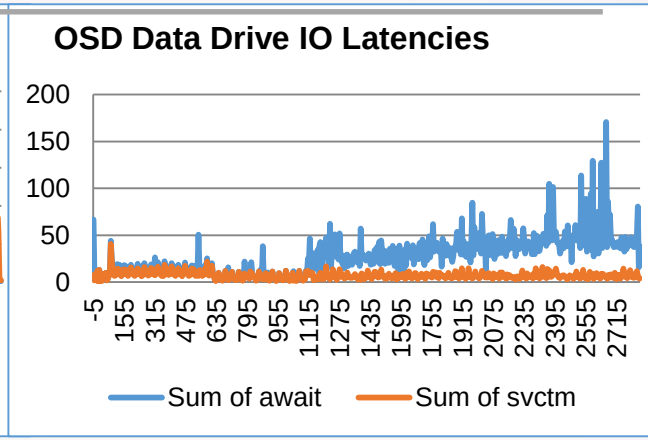


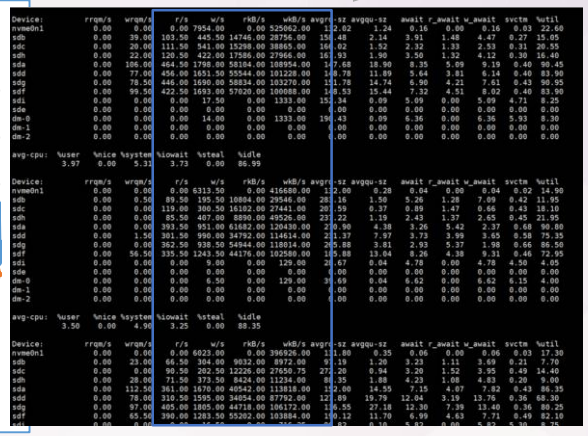
Job Name:	TeraSort
User Name:	root
Queue:	root.root
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Nov 02 09:38:18 CST 2017
Started:	Thu Nov 02 09:38:50 CST 2017
Finished:	Thu Nov 02 10:24:56 CST 2017
Elapsed:	46mins, 5sec
Diagnostics:	
Average Map Time	1mins, 37sec
Average Shuffle Time	8mins, 53sec
Average Merge Time	30sec
Average Reduce Time	23mins, 35sec

Time cost at Reduce stage is big part

Read and write concurrently







Bigdata on Cloud vs. Remote HDFS

--Ongoing rename optimizations

DirectOutputCommitter	An implementation in Spark 1.6, that return the destination address as working directory then no need to rename/move task output, no good robustness for failures, removed in Spark 2.0
IBM's "Stocator" committer	Targets Openstack Swift, good robustness, but it is another file system for s3a
Staging committer	A choice of new s3a committer*, need large capacity of hard disk for staged data
Magic committer	A choice of new s3a committer*, if you know your object store is consistent or use s3gurad, this committer has higher performance

Rename operation can be improved!

* New s3a committer has merged into trunk, and will release in Hadoop 3.1 or later

SUMMARY & NEXT STEP

Summary and Next Step

Summary

- Bigdata on Ceph data lake is functionality ready validated by industry standard decision making workloads TPC-DS
- Bigdata on the Cloud delivers on-par performance with remote HDFS for batch analytics, intensive write operations still need further optimizations
- All flash solutions demonstrated significant TCO benefit compared with HDD solutions

Next

- Expand analytic workloads scope
- Rename operations optimizations to improve the performance
- Accelerating the performance with speed up layer for shuffle

Q&A

BACKUP

Experiment environment

Cluster	Hadoop head	Hadoop slave	Load balancer	OSD	RGW
Roles	Hadoop name node Secondary name node Resource manager Data node Node manager Hive metastore service Yarn history server Spark history server Presto server	Data node Node manager Presto server	Haproxy	Ceph osd	Ceph rados gateway
# of node	1	5	1	5	5
Processor	Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz 44 cores HT enabled				Intel(R) Xeon(R) CPU E31280 @ 3.50GH 4 cores HT enabled
Memory	128GB		64GB	128GB	32GB
Storage	4x 1TB HDD 2x Intel S3510 480GB SSD(vs s3700 metrics)		1x Intel S3510 480 GB SSD	• 1x Intel® P3700 1.6TB as jounal • 4x 1.6TB Intel® SSD DC S3510 2X 400GB s370 as data store	1x Intel S3510 480 GB SSD
Network	10GB		40GB	10GB+10GB	10GB

ceph

SW Configuration	
Hadoop version	2.7.3/2.8.1
Spark version	2.1.1/2.2.0
Hive version	2.2.1
Presto version	0.177
Executor memory	22GB
Executor cores	5
# of executor	24
JDK version	1.8.0_131
Memory.overhead	5GB

S3A Key Performance Configuration	
fs.s3a.connection.maximum	10
fs.s3a.threads.max	30
fs.s3a.socket.send.buffer	8192
fs.s3a.socket.recv.buffer	8192
fs.s3a.threads.keepalivetime	60
fs.s3a.max.total.tasks	1000
fs.s3a.multipart.size	100M
fs.s3a.block.size	32M
fs.s3a.readahead.range	64k
fs.s3a.fast.upload	true
fs.s3a.fast.upload.buffer	array
fs.s3a.fast.upload.active.blocks	4
fs.s3a.experimental.input.fadvise	radom

Legal Disclaimer & Optimization Notice

- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.
- INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
-

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804