TECHNICAL PAPER

HITACHI
Inspire the Next

Build a Big Data Hadoop Infrastructure using Hitachi Compute Rack 220S Servers and Cloudera Hadoop

Reference Architecture Guide

By Madhura Limaye

June 4, 2013

Hitachi Data Systems

Feedback

Hitachi Data Systems welcomes your feedback. Please share your thoughts by sending an email message to SolutionLab@hds.com. To assist the routing of this message, use the paper number in the subject and the title of this white paper in the text.

Table of Contents

Build a Big Data Hadoop Infrastructure using Hitachi Compute Rack 220S Servers and Cloudera Hadoop

Reference Architecture Guide

This is a reference architecture guide for building a scalable big data infrastructure using the Cloudera Hadoop Distribution on Hitachi Compute Rack 220S servers. It provides guidance on how to build a big data infrastructure to satisfy the large scale data processing and analytics requirements of your organization using Hitachi Compute Rack 220S servers and Cloudera Hadoop Distribution[1].

Hitachi Compute Rack 220S is an Intel Xeon processor-based midrange rack mountable server platform, providing advanced systems management and redundancy options. It is data center friendly, with a 2U footprint, while delivering the performance that is required to meet enterprise level challenges — all characteristics valuable for a big data infrastructure.

Cloudera is the leader in Apache Hadoop-based software and services and offers a big data platform for enterprises and organizations. Cloudera Manager is the industry's first end-to-end management application for Apache Hadoop. See more at http://www.cloudera.com.

You need familiarity with the use and configuration of the following to use this reference architecture guide:

- Hitachi Compute Rack 220S
- Red Hat Enterprise Linux 6.3
- Cloudera Hadoop distribution (Hadoop 2.0.0-cdh4.1.2)
- Hitachi Compute Rack 210H
- Cloudera Manager 4.1.3

1. Numerous third parties may have claims to intellectual property relevant to an implementation of HADOOP. Hitachi Data Systems requests that anyone reading this white paper ensure that they have sufficient rights to use any intellectual property relevant to their particular application. Hitachi Data Systems provides no warranty, indemnity or other protection against third party claims of intellectual property infringement arising from the use of HADOOP-compliant software on HDS equipment.

Solution Overview

This reference architecture uses Hitachi Compute Rack 220S servers in a Cloudera Hadoop Distribution cluster, and the cluster is used to provide guidelines on how to build a Hadoop infrastructure for big data projects.

This solution also validates the integration of Hitachi hardware with the Cloudera Hadoop Distribution software stack using a 10 node cluster.

Note— Testing of this configuration was in a lab environment. Many things affect production environments beyond prediction or duplication in a lab environment. Follow the recommended practice of conducting proof-of-concept testing for acceptable results in a non-production, isolated test environment that otherwise matches your production environment before your production implementation of this solution.

Figure 1 illustrates the high-level design of this reference architecture.

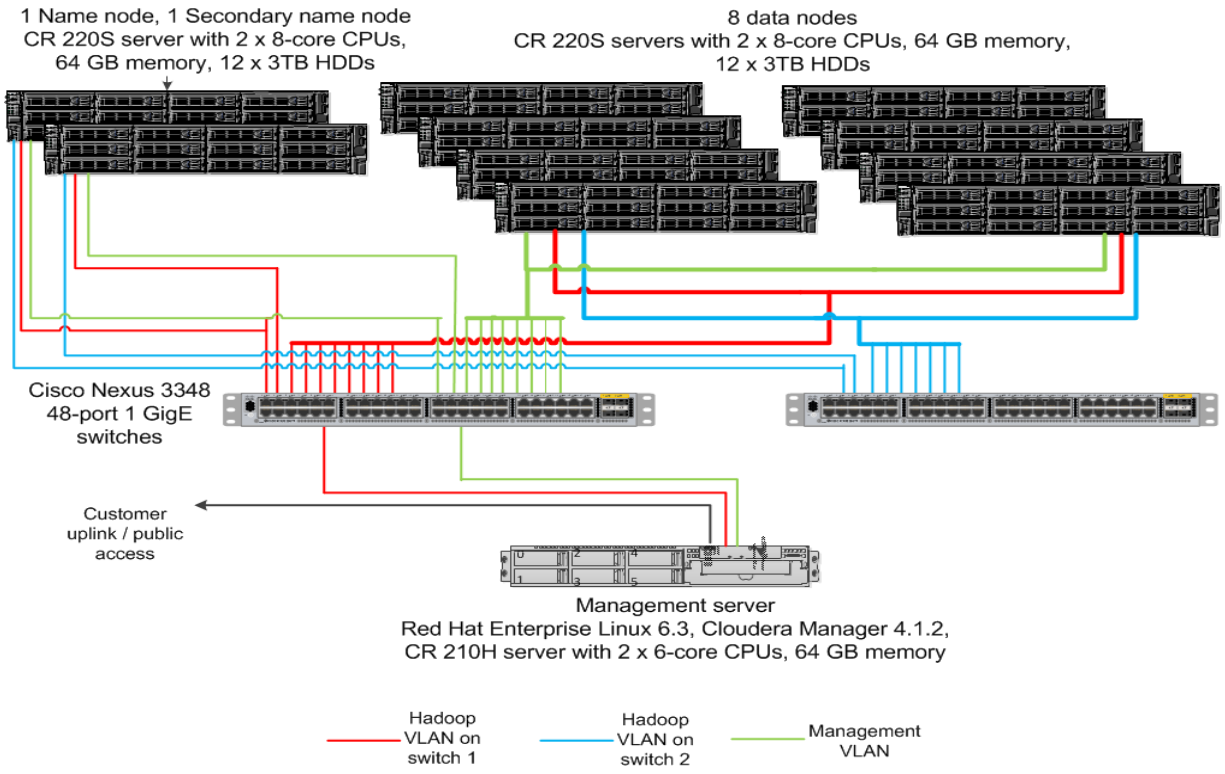

Figure 1

Key Solution Components

Following are descriptions of the key hardware and software components used in the test environment for the Cloudera Hadoop Distribution on Hitachi Compute Rack 220S servers reference architecture testing.

Hardware Components

Table 1 lists detailed information about the hardware components used in the Hitachi Data Systems lab to validate this reference architecture solution.

Table 1. Hardware Components

Hardware	*Description*	*Version*	*Quantity*
Hitachi Server CR 220S	Cloudera Name Node primary and Job tracker ■ 2 × 8-Core Intel Xeon E2470 @ 2.3 GHz ■ 64 GB Main Memory ■ 2 × 1 GigE (onboard) ■ 2 × 300 GB drives ■ 12 x 3.5” 3 TB NL-SAS 7200 RPM drives	64-bit Red Hat Enterprise Linux 6.3, Cloudera Hadoop Distribution	1
Hitachi Server CR 220S	Cloudera Name Node secondary ■ 2 × 8-Core Intel Xeon E2470 @ 2.3 GHz ■ 64 GB Main Memory ■ 2 × 1 GigE (onboard) ■ 2 × 300 GB drives ■ 12 × 3.5” 3 TB NL-SAS 7200 RPM drives	64-bit Red Hat Enterprise Linux 6.3, Cloudera Hadoop Distribution	1

Table 1. Hardware Components (Continued)

Hardware	Description	Version	Quantity
Hitachi Server CR 220S	Cloudera data nodes and task trackers ■ 2 × 8-Core Intel Xeon E2470 @ 2.3 GHz ■ 64 GB Main Memory ■ 2 × 1 GigE (onboard) ■ 2 × 300 GB drives ■ 12 × 3.5” 3 TB NL-SAS 7200 RPM drives	64-bit Red Hat Enterprise Linux 6.3, Cloudera Hadoop Distribution	8
Cisco Nexus 3348 (with LACP including sFlow)	■ 48 × 1 GigE		2
Hitachi Server CR 210H	Management Node for Cloudera Manager ■ 2 × 6-Core Intel Xeon E2620 @ 2 GHz ■ 64 GB Main Memory ■ 2 × 1 GigE (onboard) ■ 2 × 2.5” 300 GB drives	64-bit Red Hat Enterprise Linux 6.3, Cloudera Manager 4.1.3	1

Hitachi Compute Rack 220S

Hitachi Compute Rack 220S is an Intel Xeon processor-based midrange rack mountable server platform, providing advanced systems management and redundancy options. It is data center friendly, with a 2U footprint, while delivering the performance that is required to meet enterprise level challenges.

The benefits of Hitachi Compute Rack 220S are the following:

- Web-based management interface
- RAID level configuration, with up to twelve 3.5 inch internal drives
- Eco-friendly power-saving capabilities
- 2 socket Intel based server
- Configuration flexibility to meet business needs
- Dense 2U rack mountable design

Hitachi Compute Rack 210H

Hitachi Compute Rack 210H is a midrange rack mountable server platform, providing advanced systems management and redundancy options. It is data center friendly, with a 1U footprint, while delivering the performance that is required to meet enterprise level challenges.

The benefits of Hitachi Compute Rack 210H are the following:

- Web-based management interface
- RAID level configuration, with up to six 2.5 inch internal drives
- Eco-friendly power-saving capabilities
- Configuration flexibility to meet business needs
- Dense 1U rack mountable design

Compared to commodity rack servers, Hitachi Compute Rack servers offer enterprise class redundant features and superior customer support at price points that are competitive with commodity servers. By using Hitachi Compute servers for Hadoop infrastructure, customers can focus more on developing innovative applications and less on infrastructure maintenance and support.

Software Components

Table 2 lists the software components deployed for this reference architecture.

Table 2. Software Components

Software	*Version*
Red Hat Enterprise Linux	6.3 (64-bit)
Cloudera Hadoop distribution	Hadoop 2.0.0-cdh4.1.2
Cloudera Manager	4.1.3

Solution Design

Test Environment

The hardware and software components for the reference architecture are configured as follows:

- A Cloudera Hadoop Distribution cluster is configured with 10 cluster nodes on the 10 Hitachi Compute Rack 220S servers using Cloudera Manager.
  - All cluster nodes have Red Hat Enterprise Linux (RHEL) 6.3 installed on them.
  - Using Cloudera Manager, the Cloudera Hadoop Distribution (version Hadoop 2.0.0-cdh4.1.2) core configuration is also installed on these cluster nodes. The core configuration includes the Hadoop distributed file system (HDFS), Hadoop Map-Reduce infrastructure (MapReduce) and 2 other Hadoop components called Hue and Oozie, which are not used nor analyzed in this reference architecture.
  - Two of the 10 cluster nodes host the HDFS name node and the secondary name node. The name node also hosts the job tracker for Hadoop MapReduce.
  - The remaining eight cluster nodes are used as HDFS data nodes and Hadoop MapReduce task tracker nodes.
- Hitachi Compute Rack 210H is installed with 64-bit Red Hat Enterprise Linux 6.3 and the Cloudera Manager 4.1.3. Cloudera Manager is a web based management platform for Cloudera Hadoop cluster installation and management/administration.

Storage Architecture

Cluster Data Nodes and Task Trackers

Apache Hadoop offers scale-out Hadoop distributed file system (HDFS) with built-in data redundancy and is designed to work with direct attached storage in just a bunch of disks (JBOD) configuration on the Hadoop data nodes. As a result, for this reference architecture, the physical disks available on the cluster data nodes are used in a JBOD configuration. Each cluster data node also runs a Hadoop task tracker role.

- Each data node has 12 x 3TB disks in addition to the operating system disks. These additional 12 disks are formatted using the ext4 file system with 4K block size and are mounted via /etc/fstab as /data1, /data2 … /data12. The ext4 file system was chosen for this reference architecture. The primary intention behind this choice was to use a commonly used Linux file system. Other file systems like ext3 or xfs can also be used. Testing will be required to compare the impact of using the different file systems, and this testing is outside of the scope of this current reference architecture.
- These disks, /data1 to /data12, are used within the Cloudera Hadoop Distribution HDFS and Hadoop MapReduce configuration for data node directories and task tracker local data directories.

Cluster Name Node, Secondary Name Node and Job Tracker

Cloudera Hadoop Distribution also uses the local storage on the Hadoop name node and secondary name node of the cluster. For resiliency, it is recommended that the name node and secondary name node use RAIDed local disks. As a result, for this reference architecture, on the name node and secondary name node, a RAID group of the physical disks available on these nodes is setup as follows:

- The name node and secondary name node have 12 x 3TB disks in addition to the operating system disks. Note that the reference architecture testing helps us conclude that the name node and secondary name node disk configuration can be significantly downsized in terms of number of disks, from what is being used here, since the resource utilization seen on these nodes turned out to be fairly low.
- A RAID 5 group (11D+1P) is created for these 12 additional disks on the name node and the secondary name node. The MegaRAID manager available for Hitachi Compute Rack 220S is used to create the RAID group on the name node and secondary name node.
- The RAID group is then used as a logical disk and is formatted with the xfs file system. The original choice of the file system was ext4 to match what is used on the data nodes, but due to its 16TB maximum file system size limitation, it could not be used. This logical disk is mounted automatically via an entry in /etc/fstab as /data1. ext3 or ext4 file systems can be used with a downsized disk configuration with smaller file system size on the name node and secondary name node. Additional testing will be required to compare the impact of using different file systems, and this testing is outside the scope of this current reference architecture.
- /data1 is used within the Cloudera Hadoop distribution configuration for the Hadoop name node and the secondary name node data directories. Also, the name node hosts the role of the Hadoop job tracker and defines the jobtracker local data directory on /data1. Resiliency is provided for the name node and job tracker data directories by the RAID group that make up /data1.

Cloudera Hadoop Distributed File System (HDFS) Configuration

For Cloudera HDFS, the default block size of 128 MB and the default replication of 3 is used.

Network Architecture

In this reference architecture, each cluster node has 2 active 1 GigE interfaces in addition to a management interface. There are 2 Cisco Nexus 48-port 1 GigE switches included in the reference architecture. The Cisco Nexus switches and 1 GigE interfaces are configured as follows:

- 2 VLANs are set up, one on each of the 2 Cisco Nexus switches. These are primarily used as the Hadoop data VLANs. On one of the Cisco Nexus switches, a management VLAN is also set up.
- For each cluster node, two active 1 GigE interfaces are connected to the Hadoop data VLAN ports, one interface connected to one switch and the other interface connected to the other switch.
- The two active 1 GigE interfaces on the cluster nodes are then bonded to form an active-active channel (bond0) via the NIC bonding driver available as part of Red Hat Enterprise Linux (RHEL) 6.3.
- The active-active channel, bond0, hosts a single IP address on the Hadoop data VLAN and this is the primary network for the Cloudera Hadoop Distribution cluster nodes and for all Hadoop distributed file system and Hadoop MapReduce traffic.
- Each cluster node also has a management interface connected to the management VLAN ports. The management interface has an IP address hosted on it for administrative purposes.
- The Hitachi Compute Rack 210H management server has a management interface connected to a management VLAN port and an interface connected to the customer uplink or the public network.

Note that for this reference architecture, a 1 GigE test network is used within the cluster nodes, where all the nodes belong to the same rack. With a single rack design, this test network is sufficient. But, when growing the cluster across multiple racks, the recommendation is to use a 10 GigE core switch between the two 1 GigE networks used within the individual racks, to connect the nodes across the racks.

Administration and Monitoring of Cloudera Cluster

The free edition of the Cloudera Manager used in this reference architecture provides management of the Cloudera Hadoop distribution cluster, but does not provide performance monitoring of the cluster nodes. Cloudera Manager enterprise edition provides performance monitoring of the cluster nodes in addition to the administration and management of the cluster itself and thus can be used instead of the free edition of the Cloudera Manager specifically for performance monitoring.

Alternatively, there are other options for monitoring the performance the cluster nodes, ranging from open source products to Hitachi IT Operations Analyzer or Hitachi Compute Systems Manager.

Engineering Validation

The following section describes the test methodology used for this reference architecture testing and the results, their analysis and conclusions. The goal of the testing is to determine guidelines for building a big data infrastructure to satisfy the requirements of customers, using Hitachi Compute Rack 220S servers and Cloudera Hadoop Distribution.

Test Methodology

The reference architecture testing involves using standard Hadoop benchmarks, namely Terasort and TestDFSIO, to test the Cloudera Hadoop Distribution cluster described in “Test Environment” on page 6. Testing involves varying the cluster configuration and the input dataset (as related to the benchmark), thus providing multiple test cases and therefore multiple data points which can then be used to suggest guidelines about configuring big data infrastructures using Hitachi Compute Rack 220S servers and Cloudera Hadoop Distribution.

All testing is conducted on the Cloudera Hadoop distribution cluster using the private Hadoop data VLAN with no other workload running.

Varying the Cluster Configuration

The cluster configuration for the testing is changed specifically in terms of the number data nodes / task trackers being used in the cluster. For all cluster configurations, there is also one name node and one secondary name node. The job tracker is hosted on the same node as the name node. The reference architecture testing focuses on the following cluster configurations:

- Configuration 1: 3 data nodes / task trackers
- Configuration 2: 4 data nodes / task trackers
- Configuration 3: 5 data nodes / task trackers
- Configuration 4: 6 data nodes / task trackers
- Configuration 5: 7 data nodes / task trackers
- Configuration 6: 8 data nodes / task trackers

The first configuration is chosen to have enough number of data nodes to support the default HDFS replication factor of 3. Beyond that, the number of data nodes are increased by 1 till we reach a total of 8 data nodes. Note that each of these data nodes also hosts a task tracker role, therefore this document refers to them as data nodes / task trackers.

To switch between cluster configurations, the Cloudera Hadoop Distribution configuration is changed by commissioning the additional data nodes / task trackers in the HDFS or Hadoop MapReduce configuration.

Additionally, some HDFS and Hadoop MapReduce parameters have been changed from their defaults for the testing across different cluster configurations. These parameters were changed based on a limited parameter evaluation. The focus of the evaluation was to determine values for specific cluster configuration parameters which allow better than average performance for the tests. The parameter evaluation provided the recommended values for specific parameters as listed in Table 3 and Table 4.

Cluster configuration parameters for the Hadoop distributed file system (HDFS)

Table 3 summarizes the HDFS configuration parameters:

Table 3. HDFS Parameters

HDFS configuration parameter	*Value used*	*Description of parameter*
dfs.data.dir or dfs.datanode.data.dir	/data1/dfs/dn, /data2/dfs/dn, /data3/dfs/dn, /data4/dfs/dn, /data5/dfs/dn, /data6/dfs/dn, /data6/dfs/dn, /data7/dfs/dn, /data8/dfs/dn, /data9/dfs/dn, /data10/dfs/dn, /data11/dfs/dn, /data12/dfs/dn	Comma-delimited list of directories on the local file system where the DataNode stores HDFS block data.
dfs.datanode.failed.volumes.tolerated	4	The number of volumes that are allowed to fail before a DataNode stops offering service.
dfs.name.dir or dfs.namenode.name.dir	/data1/dfs/nn for name node /data1/dfs/snn for secondary name node Note: /data1 is hosted on a RAID5 disk group for resiliency purposes	Determines where on the local file system the NameNode should store the name table
dfs.blocksize	128 MB	Default block size for HDFS files
dfs.replication	3	Default block replication. Configures the number of replications to make when the file is created.
Java heap size on data and name nodes	1 GB for data node, 2 GB for name node	Maximum size for the Java Process heap.

Cluster Configuration Parameters for the Hadoop MapReduce Infrastructure

Table 4 summarizes the Hadoop MapReduce configuration parameters:

Table 4. Hadoop MapReduce Parameters

Hadoop MapReduce configuration parameter	*Value used*	*Description of parameter*
mapred.local.dir on task trackers	/data1/mapred/local, /data2/mapred/local, /data3/mapred/local, /data4/mapred/local, /data5/mapred/local, /data6/mapred/local, /data7/mapred/local, /data8/mapred/local, /data9/mapred/local, /data10/mapred/local, /data11/mapred/local, /data12/mapred/local	List of directories on the local filesystem where a TaskTracker stores intermediate data files.
mapred.local.dir on job tracker	/data1/mapred/jt Note: /data1 is hosted on a RAID5 disk group for resiliency purposes	Directory on the local filesystem where the JobTracker stores job configuration data.
mapred.tasktracker.map.tasks.maximum	32	The maximum number of map tasks that a TaskTracker can run simultaneously.
mapred.tasktracker.reduce.tasks.maximum	16	The maximum number of reduce tasks that a TaskTracker can run simultaneously.
mapred.job.reuse.jvm.num.tasks	-1	Number of tasks to run per JVM. If set to -1, there is no limit.
io.sort.mb	Values depend on benchmark. See test specific sections “Terasort Benchmark” on page 13 and “TestDFSIO Benchmark” on page 16.	The total amount of memory buffer, in megabytes, to use while sorting files. Note that this memory comes out of the user JVM heap size.
io.sort.rec	Values depend on benchmark. See test specific sections “Terasort Benchmark” on page 13 and “TestDFSIO Benchmark” on page 16.	The percentage of 'io.sort.mb' dedicated to tracking record boundaries.

Switching between Cluster Configurations

When switching between the different Cloudera Hadoop Distribution cluster configurations, the following steps are followed. Note that these steps allow fair comparison between the runs, to the extent that they can be controlled and managed.

1. Cleanup files created by previous test runs using benchmark specific commands for cleanup of test data.
2. Stop all Cloudera Hadoop Distribution services (includes HDFS and Hadoop MapReduce) via the Cloudera Manager.
3. Update the cluster configuration via the Cloudera Manager by commissioning the additional data nodes and task trackers.
4. Reboot all cluster nodes.
5. Once cluster nodes are up, start all Cloudera Hadoop Distribution services (including HDFS and Hadoop MapReduce) via the Cloudera Manager.
6. Check the health of the Hadoop distributed file system via the 'hadoop fsck /' command and ensure it is 'HEALTHY'. If not 'HEALTHY' (typically this is when fsck shows it to be 'CORRUPTED'), fix it to make it 'HEALTHY' via the 'hadoop fsck' command.

Varying the Input Dataset

A fixed input data set is used for the reference architecture testing for each of the benchmarks (Terasort, TestDFSIO) being used in the testing. The input data sets are discussed in “Terasort Benchmark” on page 13 and “TestDFSIO Benchmark” on page 16, where more details about the benchmarks are provided. But in summary:

The Terasort benchmark takes as input, the size of the data to be sorted. The reference architecture testing focuses on the input data set of:

- 3.5 TB

For the TestDFSIO benchmark, the input arguments include the number of files to read/write and the size of each of those files. The data set that is used for this benchmark is as follows:

- 50,000 files of 200 MB size

These input data sets were chosen based on typical datasets seen among customers working on big-data initiatives.

Terasort Benchmark

A Terasort benchmark test for this reference architecture consists of the following two steps:

1. Generating the input data via Teragen
2. Running the actual Terasort on the input data

The command to run Teragen to generate the required amount of data for the input to sorting is as follows. The command below generates 3.5 TB of input data in the directory listed in the command. Note that the input data size is given in multiples of 100 bytes.

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar
teragen 35000000000 /user/hduser/terasort-input
```

The command to run Terasort and sort the data generated using Teragen is as follows. This command uses as input, the directory into which Teragen wrote the input data. The output of the sorting is written into the output directory given in the command.

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar
terasort /user/hduser/terasort-input /user/hduser/terasort-output.
```

The above commands are executed with the 'hdfs' user credentials. Simple scripts are used to wrap other commands around the teragen and terasort test commands. The other commands specifically include commands that collect the execution time for the test commands, collect the output of these test commands, and collect resource utilization information via the 'sar' command.

It is important to clean up the test files created by Teragen and Terasort before running them for a different dataset to ensure fair comparisons between the tests. This is done using the 'hdfs dfs -rm -r <terasort-input-dir>' and 'hdfs dfs -rm -r <terasort-output-dir>' commands.

High Level Test Cases

For Terasort, the input data set consists mainly of the size of the input data, which is set to 3.5 TB. Each test case involves first running Teragen to generate the data, followed by Terasort. For the chosen dataset, the cluster configuration is varied, thus resulting in the test cases shown in Table 5 for the Terasort benchmark.

Table 5. Terasort Benchmark Test Cases

Test Case	*Size of Input data*	*Cluster configuration (as per the Test Plan section above)*
SORT-TC-1	3.5 TB	Configuration 1 - 3 data nodes / task trackers
SORT-TC-2	3.5 TB	Configuration 2 - 4 data nodes / task trackers
SORT-TC-3	3.5 TB	Configuration 3 - 5 data nodes / task trackers
SORT-TC-4	3.5 TB	Configuration 4 - 6 data nodes / task trackers
SORT-TC-5	3.5 TB	Configuration 5 - 7 data nodes / task trackers
SORT-TC-6	3.5 TB	Configuration 6 - 8 data nodes / task trackers

Benchmark Specific Cluster Configuration Parameters

Before executing the test cases described above, some time was spent on cluster configuration parameter evaluation. The focus of the evaluation was to determine values for specific cluster configuration parameters which allow better than average performance for the tests. One specific test case was chosen and it was evaluated with different values of specific parameters to get the parameter values that were eventually used for the test cases.

The cluster configuration parameters that were changed for the Terasort benchmark include the following. These are Hadoop MapReduce parameters:

- io.sort.mb. This sets the total amount of memory to use for sorting files. Based on initial parameter evaluation, a value of 400 MB was chosen for the Terasort benchmark.
- Specifically for the teragen part of this benchmark, the mapred.map.tasks parameter was changed to match the maximum number of map tasks allowed across all the data nodes. With 32 maximum map tasks allowed per data node, the value that was chosen was 32 x number of datanodes, which varied depending on cluster configuration. This was set by using the '-D mapred.map.tasks=XXX' argument in the teragen command line.

TestDFSIO Benchmark

The TestDFSIO benchmark is a read and write benchmark test for Hadoop Distributed File System (HDFS). It is helpful for sanity testing HDFS and to identify potential performance bottlenecks in the Hadoop cluster.

```
Usage: TestDFSIO -read | -write | -clean [-nrFiles N] [-fileSize MB]
[-resFileresultFileName] [-bufferSize Bytes]
```

The command to run a write test that generates 50000 files of size 200 MB for a total of 10 TB is as follows. Note that the file size is always given in MB, so appropriate conversion is required for sizes in TB or GB.

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-
2.0.0-cdh4.1.2-tests.jar TestDFSIO -write -nrFiles 50000 -fileSize 200
```

The command to run the corresponding read test using 10 files of size 1 GB each is as follows. Again, the file size is always given in MB.

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-
2.0.0-cdh4.1.2-tests.jar TestDFSIO -read -nrFiles 50000 -fileSize 200
```

The command to clean up the data set written previously is:

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-
2.0.0-cdh4.1.2-tests.jar TestDFSIO -clean
```

One important note about TestDFSIO is that it executes a read or write test exclusively. It cannot do a test that includes both reads and writes, and therefore may not map directly to real-world workloads.

The above commands are executed with the 'hdfs' user credentials. Simple scripts are used to wrap other commands around the TestDFSIO test commands. The other commands specifically include commands that collect the execution time for the test commands, collect the output of these test commands, and collect resource utilization information via the 'sar' command.

It is important to use a TestDFSIO –clean test in between sets of read/ write tests for different data sets to ensure fair comparisons between the test runs. Also, it is recommended to first run a write test and follow it up with a read test since the read test relies on existing data and the write test can provide this data and no additional steps are required to generate the data for a read test.

High Level Test Cases

For TestDFSIO, the input data set consists of the number of files and the size of each of those files. The specific values used are 50,000 files of size 200 MB each. For this data set, on each cluster configuration, a write test is attempted, followed by a read test. This results in the test cases shown in Table 6 for the TestDFSIO benchmark.

Table 6. TestDFSIO Test Cases

Test Case	*Number of files*	*Size of files*	*Cluster configuration (as per the Test Plan section above)*
IO-TC-1	50000	200 MB	Configuration 1 - 3 data nodes / task trackers
IO-TC-2	50000	200 MB	Configuration 2 - 4 data nodes / task trackers
IO-TC-3	50000	200 MB	Configuration 3 - 5 data nodes / task trackers
IO-TC-4	50000	200 MB	Configuration 4 - 6 data nodes / task trackers
IO-TC-5	50000	200 MB	Configuration 5 - 7 data nodes / task trackers
IO-TC-6	50000	200 MB	Configuration 6 - 8 data nodes / task trackers

Benchmark Specific Cluster Configuration Parameters

Before executing the test cases described above, some time was spent on cluster configuration parameter evaluation. The focus of the evaluation was to determine values for specific cluster configuration parameters which allow better than average performance for the tests. One specific test case was chosen and it was evaluated with different values of specific parameters to get the parameter values that were eventually used during the test runs.

The cluster configuration parameters that were changed for the TestDFSIO benchmark include:

- io.sort.mb - This sets the total amount of memory to use for sorting files. Based on initial parameter evaluation, a value of 400 MB was chosen for the TestDFSIO benchmark testing.
- io.sort.rec - This sets the percentage of the io.sort.mb memory that is used for tracking record boundaries. Based on initial parameter evaluation, a value of 0.5 was chosen for the TestDFSIO benchmark testing.

Test Results Collection

When conducting the tests for the input datasets across the cluster configurations, the cluster nodes resource utilization (for CPU, memory, disk I/O and network) is collected using the Red Hat Enterprise Linux 6.3 ‘sar’ command. Also, the execution time for the all tests is collected using the ‘time’ command. Finally, the output of the benchmark test provides information about the test and this is also recorded.

The test cases are conducted in a sequential fashion, with the necessary steps taken for cleanup when switching between cluster configurations and input data sets as discussed in “Switching between Cluster Configurations” on page 13. This was to avoid any inconsistencies while testing to the extent possible so that comparisons between the test runs would be fair.

Test Results Summary

The following sections discuss the results for some of the test cases listed in Table 5, “Terasort Benchmark Test Cases,” on page 15 and Table 6, “TestDFSIO Test Cases,” on page 17. First the Terasort benchmark results are shown followed by the TestDFSIO benchmark results.

For the Terasort benchmark, the variation of the execution time against the cluster configuration is shown. For the TestDFSIO benchmark, the variation of the execution time as well as the throughput against the cluster configuration is shown. This is followed by an analysis of the test results as it relates to customer requirements.

Teragen/Terasort Benchmark Results

Execution time results

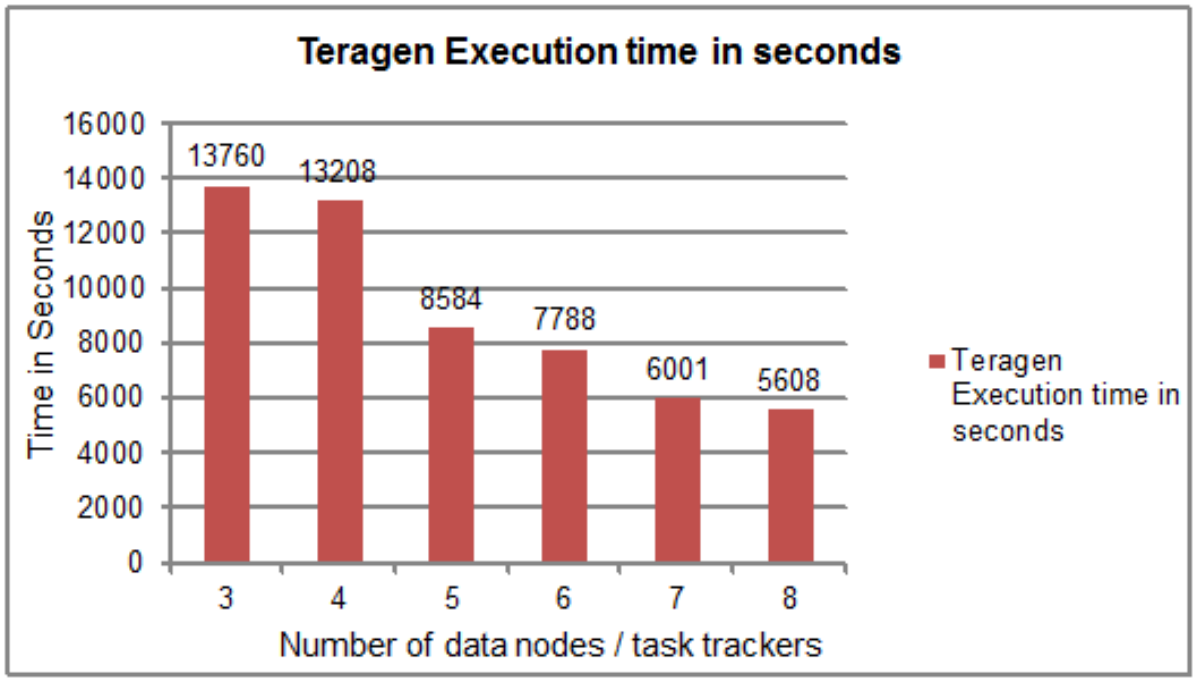

Figure 2

Figure 2 shows the variation of the execution time for Teragen for a 3.5 TB input dataset. The Y-axis shows the time in seconds and the X-axis shows the different cluster configurations in terms of number of data nodes / task trackers.

The execution time for Teragen decreases as we increase the number of data nodes / task trackers. This can be explained by the fact that each additional data node / task tracker provides additional resources for the same amount of input data, thus enabling quicker execution of the Teragen benchmark.

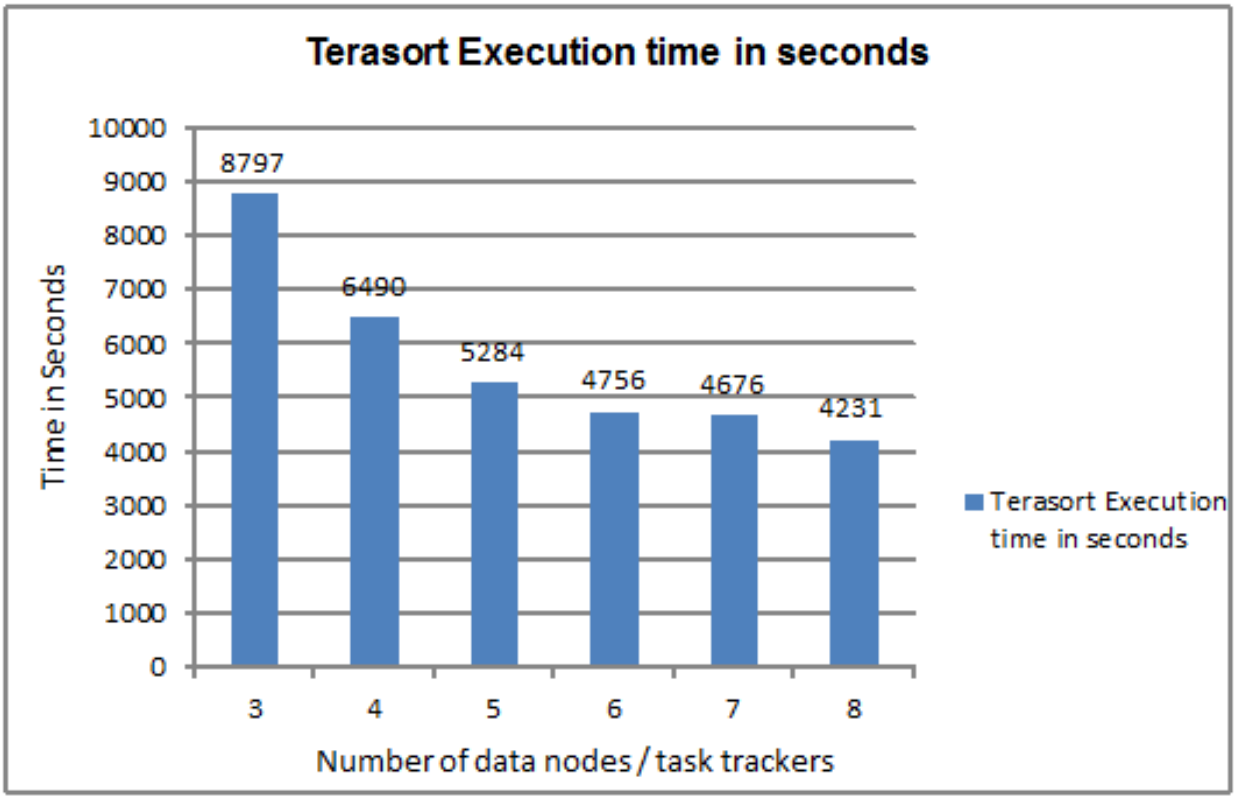

Figure 3

Figure 3 shows the variation of the execution time for Terasort for a 3.5 TB input dataset. The Y-axis shows the time in seconds and the X-axis shows the different cluster configurations in terms of number of data nodes / task trackers.

Again, similar to the Teragen benchmark, there is an improvement in the execution time for doing Terasort on a 3.5 TB input data set as the number of data nodes / task trackers are increased. This is understandable due to the additional resources provided by the additional data nodes / task trackers to process the same amount of data.

TestDFSIO Benchmark Results

Execution time and throughput results

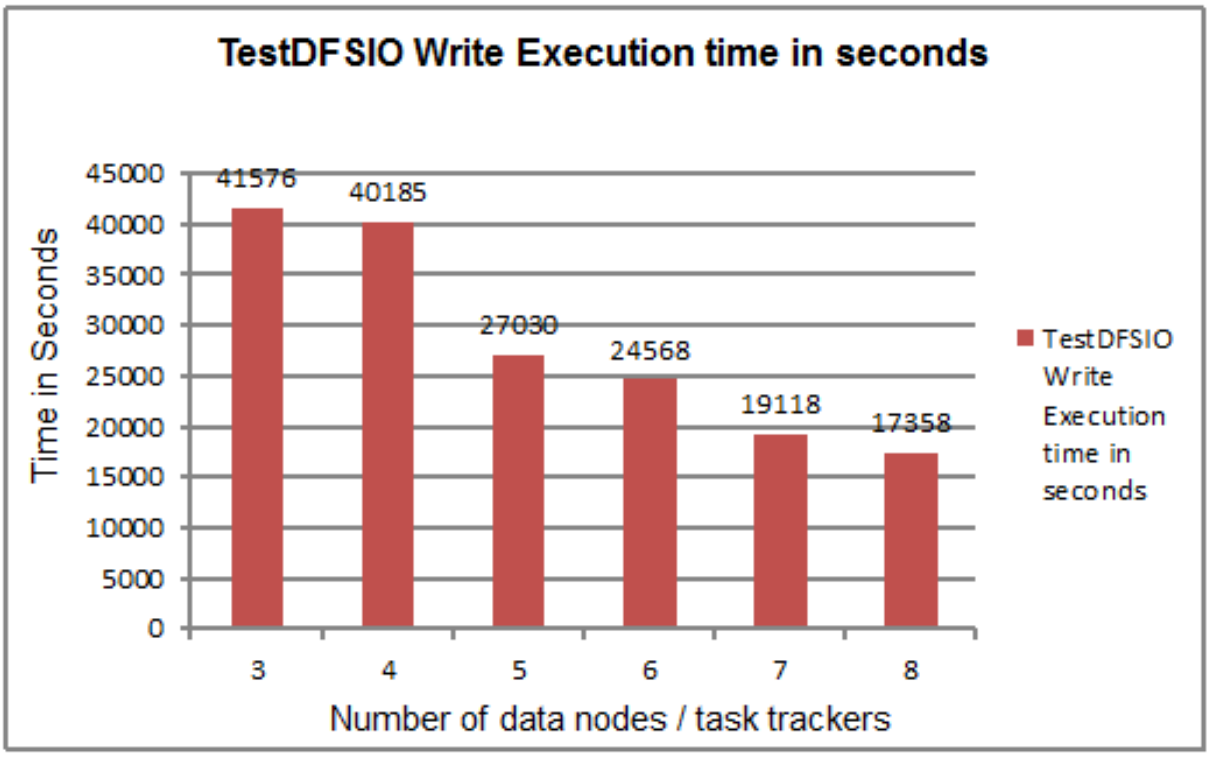

Figure 4

Figure 4 shows the variation of the execution time for the TestDFSIO write benchmark for the input data set of 50,000 files each of 200 MB size. The Y-axis shows the time in seconds and the X-axis shows the different cluster configurations in terms of number of data nodes / task trackers being used.

For the TestDFSIO write benchmark, there is a decrease in the execution time as the number of data nodes / task trackers are increased. The additional CPUs, memory, disks and network resources allow for more parallelism of map and reduce tasks, thus allowing faster execution times with increasing number of data nodes / task trackers.

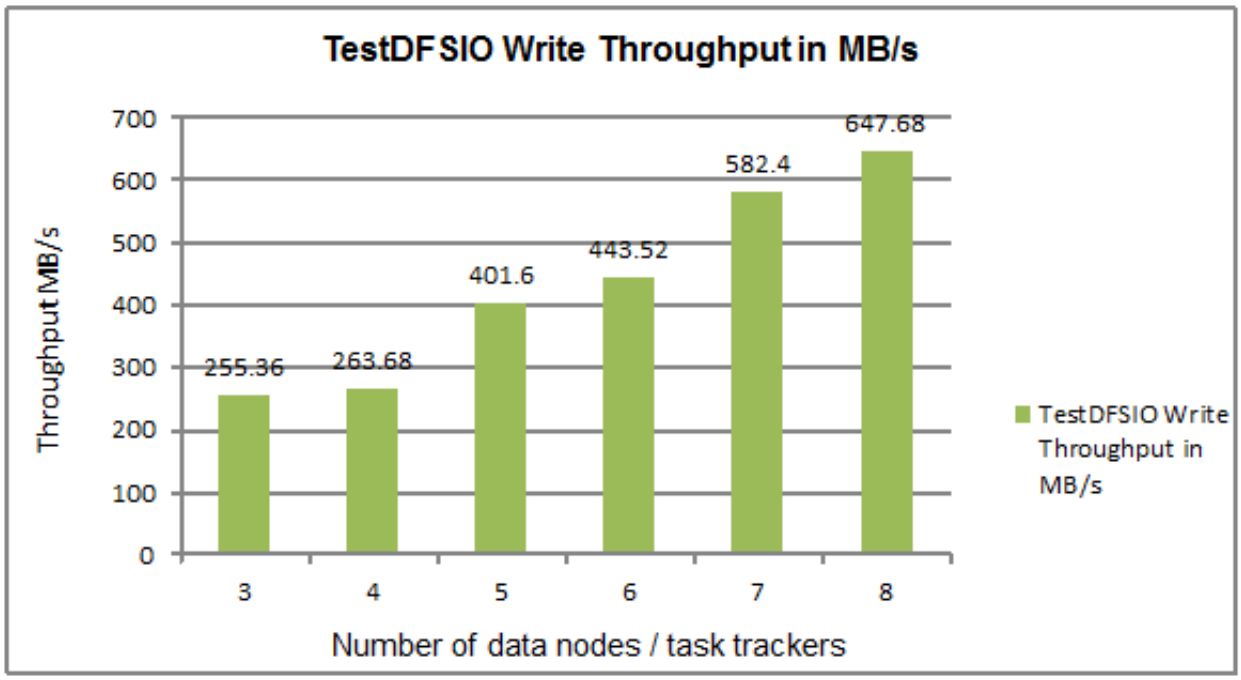

Figure 5

Figure 5 shows the variation of the throughput seen for the TestDFSIO write benchmark for the input data set of 50,000 files each of 200 MB size. The Y-axis shows the throughput in MB/sec and the X-axis shows the different cluster configurations in terms of number of data nodes / task trackers being used. The throughput for the TestDFSIO benchmark is reported as part of the output of the test.

For the TestDFSIO write benchmark, there is an increase in the throughput as the number of data nodes / task trackers are increased. This is as expected due to additional resources with each additional data node / task tracker.

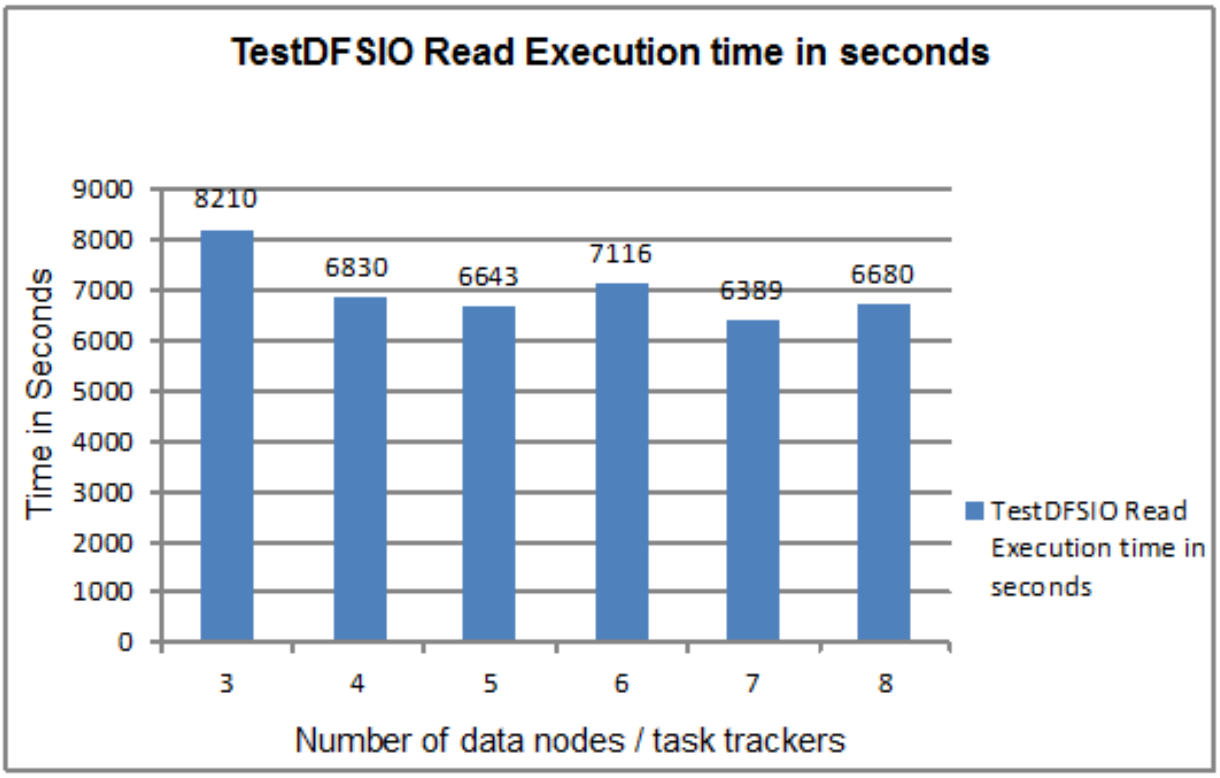

Figure 6

Figure 6 shows the variation of the execution time for the TestDFSIO read benchmark for the input data set of 50,000 files each of 200 MB size. The Y-axis shows the time in seconds and the X-axis shows the different cluster configurations in terms of number of data nodes / task trackers being used.

For the TestDFSIO read benchmark, there isn't a straightforward trend showing a decrease in the execution time as the number of data nodes / task trackers are increased, and this could likely be related to known open issues with respect to the reliability of the results reported by the TestDFSIO read benchmark.

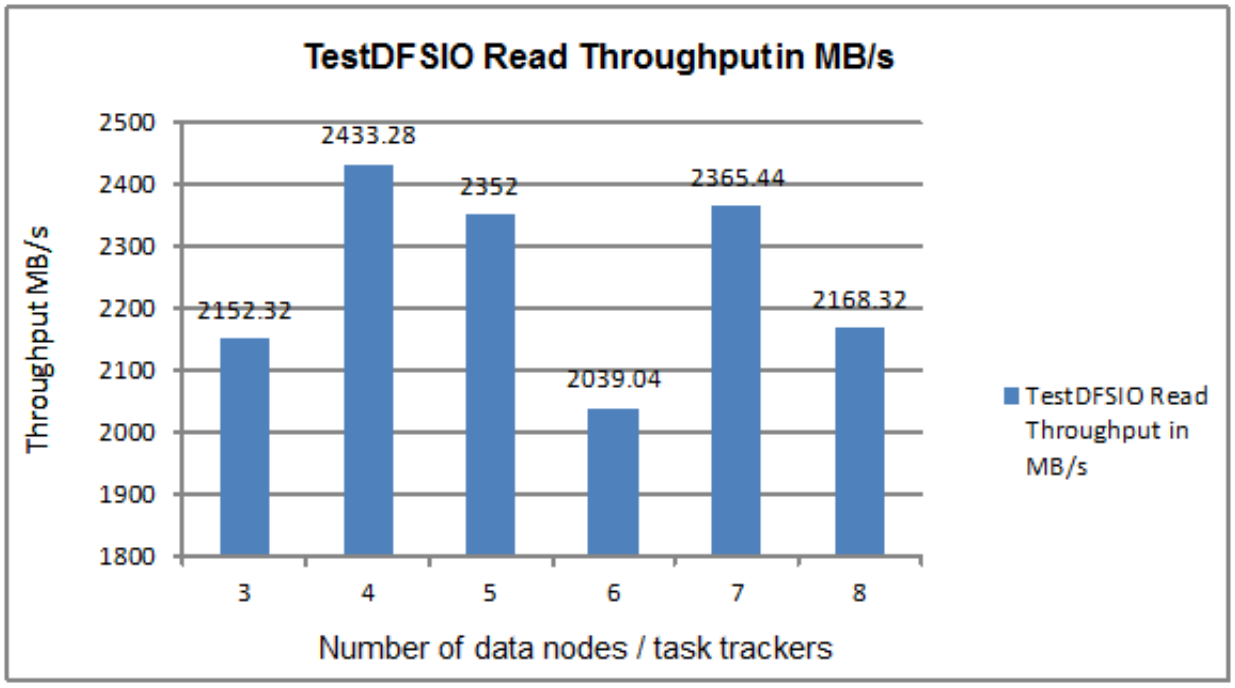

Figure 7

Figure 7 shows the variation of the throughput for the TestDFSIO read benchmark for the input data set of 50,000 files each of 200 MB size. The Y-axis shows the throughput in MB/sec and the X-axis shows the different cluster configurations in terms of number of data nodes / task trackers being used. The throughput for the TestDFSIO benchmark is reported as part of the output of the test.

For the TestDFSIO read benchmark, there isn't a straightforward trend showing a consistent increase in the throughput as the number of data nodes / task trackers are increased and this could likely be related to known open issues with respect to with respect to the reliability of the results reported by the TestDFSIO read benchmark.

Resource Utilization on Data Nodes / Task Trackers

CPU, memory, disk and network utilization information was collected on all data nodes / task trackers throughout the duration of the benchmark tests.

With a smaller number of data nodes or task trackers (3 like in the minimal configuration), average CPU utilization is 70% for compute intensive benchmarks such as Terasort and lower (around 20%) for the storage intensive TestDFSIO benchmark. Average memory utilization is around 65%, again for the compute intensive Terasort benchmark, and around 50% for TestDFSIO benchmark. Average disk utilization (which is in terms of throughput on the disk) shows up in the range of 30% to 40%. Network utilization reaches a maximum of 500 MB/sec including transmits and receives.

Increasing the number of data nodes / task trackers to 6 doesn't change the average CPU utilization much, since the benchmarks are still using up all the available CPUs on each node. With more data nodes / task trackers, there is a slight decrease in the memory, disk and network utilization on each node, since the total amount of data is distributed to a larger number of nodes, and each node therefore has to process a smaller amount of data. The average memory utilization drops to 57% for the Terasort benchmark. For TestDFSIO, there isn't a significant change in the memory utilization. The average disk utilization, in terms of throughput seen on the disks, shows a range of 20% to 30% across the benchmarks. Network utilization still reaches a maximum of around 500 MB/sec for both transmits and receives combined, but the average network utilization drops by around 10%.

Resource Utilization on Name Nodes / Job Tracker

CPU, memory, disk and network utilization information was also collected on the name nodes / job tracker throughout the duration of the benchmark tests.

The name nodes / job tracker nodes were lightly utilized during all the tests across all the cluster configurations. The CPU utilization and memory utilization is less than 10%. The disk utilization, in terms of the throughput seen by the disk, reaches a peak of 80% intermittently for the initial stages of the TestDFSIO write and read benchmarks, but was less than 10% for the remainder of the TestDFSIO benchmark. For Terasort, disk utilization does not reach above 20%. Network utilization does not go beyond 10 MB/sec including transmits and receives.

The variation in the resource utilization for the name nodes across different cluster configurations was very small.

The name node / job tracker, as configured within the reference architecture testing, is under-utilized and therefore to save costs, a smaller server configuration (like Hitachi Compute Rack 210H, with fewer CPU and less memory) can be used for the name node and job tracker cluster nodes.

Test Results Analysis

The goal of this reference architecture is to determine guidelines on building a big data infrastructure to satisfy the requirements of customers using Hitachi Compute Rack 220S servers and Cloudera Hadoop Distribution. Multiple test cases, as referenced in the previous sections, provide results, which will be used in this section to suggest such guidelines.

Customer data processing requirements that will play an important role when sizing a big data infrastructure, will likely include the following:

- The amount of data that needs processing
- The type of processing, namely compute-intensive or storage-intensive
- Limits on the execution time or throughput which can be satisfied by the configuration

These customer requirements can be used in conjunction with the analysis in this section to provide Cloudera Hadoop Distribution cluster sizing guidelines when using Hitachi Compute Rack 220S servers.

Minimal Cluster Configuration

A minimal cluster configuration can be used as a starting point for cluster sizing. The minimal configuration uses 5 Hitachi Compute Rack 220S servers with the following specifications:

- 2 × 8-Core Intel Xeon E2470 @ 2.3 GHz
- 64 GB Main Memory
- 2 × 1 GigE (onboard)
- 12 × 3.5” 3 TB NL-SAS 7200 RPM drives

Three of the 5 servers will be used as data nodes / task trackers, with the remaining two being used for the name node, secondary name node and job tracker.

This minimal configuration can perform compute intensive processing (such as sorting) of up to 3.5TB of data within 2.5 hours with a throughput of close to 400 MB/sec. The minimal configuration can also support a write-intensive workload of around 10 TB of data and complete processing within 12 hours with a throughput of 260 MB/sec.

Table 7 summarizes expected execution time and throughput for various benchmarks using the minimal configuration. The throughput for the Teragen and Terasort benchmarks is approximated using the total input data size and the execution time. For the TestDFSIO benchmark, the throughput is reported as part of the test.

Table 7. Data Nodes / Task Trackers Configuration

Benchmark	*Input dataset size*	*Execution time in hours*	*Throughput in MB/sec*
Teragen	3.5 TB	3.8	255
Terasort	3.5 TB	2.5	398
TestDFSIO write	10 TB (50000 * 200 MB)	11.5	255
TestDFSIO read	10 TB (50000 * 200 MB)	2.3	2152

The range of execution times and throughputs for specific input datasets from Table 7 can be compared against customer requirements to determine if a minimal configuration as discussed here is sufficient to satisfy the customer requirements.

The minimal configuration is conservative in terms of cost due to the minimal number of data nodes / task trackers. But, customers will likely have requirements in terms of limits on the execution time or a range of throughput for their data processing, and these requirements may require a bigger cluster configuration.

A Bigger Cluster Configuration

To achieve better execution times and throughputs, the minimal configuration can be extended to use one or more additional data nodes / task trackers, thus providing more resources for data processing. It is recommended that the additional data nodes / task trackers have similar specifications as the existing ones.

Adding nodes incrementally (one at a time) can lead to an incremental improvement in the execution times and throughputs, but for a much more substantial improvement, multiple data nodes / task trackers should be added.

For example, growing the cluster such that it is double the size of the minimal configuration will require adding 3 more data nodes / task trackers with the same specifications as the Hitachi Compute Rack 220S servers used in the minimal configuration. This cluster growth results in finishing the compute intensive processing of up to 3.5TB of data within 1.3 hours with a throughput of 735 MB/sec and finishing the write-intensive workload of 10 TB within 6.8 hours with a throughput of 445 MB/sec.

Table 8 summarizes expected execution times and throughput for various benchmarks using 6 data nodes / tasks trackers in the cluster configuration. Again, the throughput for the Teragen and Terasort benchmarks is approximated using the total input data size and the execution time. For the TestDFSIO benchmark, the throughput is reported as part of the test.

Table 8. Data Nodes / Task Trackers Configuration

Benchmark	*Input dataset size*	*Execution time in hours*	*Throughput in MB/sec*
Teragen	3.5 TB	2.2	449
Terasort	3.5 TB	1.3	735
TestDFSIO write	10 TB (50000 * 200 MB)	6.8	445
TestDFSIO read	10 TB (50000 * 200 MB)	2.0	2039

The range of execution times and throughputs for specific input datasets from Table 8 can be compared against customer requirements to determine cluster configuration suitable to satisfy the customer requirements.

3 Data Nodes / Task Trackers -> 6 Data Nodes / Task Trackers - Comparison

Table 9 summarizes the percentage improvement in the execution times and the throughput for the various benchmarks when using 6 data nodes / task trackers, when compared to the minimal configuration using 3 data nodes.

Table 9. Data Nodes Compared to 3 Data Nodes

Benchmark test	*Execution time in seconds*			*Throughput in MB/sec*		
	3 Data nodes / task trackers	6 Data nodes / task trackers	*% Decrease*	3 Data nodes / task trackers	6 Data nodes / task trackers	*% Increase*
Teragen - 3.5 TB	13760	7788	*43.4%*	254.35	449.40	*76.68%*
Terasort - 3.5 TB	8797	4756	*45.93%*	397.86	735.91	*84.96%*
TestDFSIO write 50000 200 MB files	41576	24568	*40.90%*	255.36	443.52	*73.68%*
TestDFSIO read 50000 200 MB files	8210	7116	*13.32%*	2152.32	2039.04	*-5.26%*

Cluster Sizing Guidelines

The data shared in this reference architecture document can help provide guidelines about sizing a cluster which will satisfy customer execution time or throughput requirements for data processing of specific input data sets. A minimal configuration can be chosen initially and it can be expanded to a bigger cluster configuration to achieve better execution times and throughput. In addition, this data could be extrapolated (using the graphs shown in the section “Test Results Summary” on page 18) to obtain cluster sizing estimates for input data sets or execution time and throughput requirements not tested within the reference architecture testing.

The cluster sizing guidelines are provided based on testing done in a lab environment using benchmarks. They need to be verified by conducting proof-of-concept testing for acceptable results in the appropriate environment, with the appropriate applications, before any production implementation.

Resource utilization metrics on the cluster nodes should be observed as part of a proof-of-concept testing to ensure appropriate use of cluster nodes and adjustments should be made to the cluster sizing based on these metrics.

Conclusion

This section summarizes the sizing guidelines which can be used when designing a big-data solution using Hitachi Compute Rack 220S servers and Cloudera Hadoop Distribution.

1. A minimal Cloudera Hadoop Distribution cluster configuration using 5 Hitachi Compute Rack 220S servers (with 2 name nodes and 3 data nodes) can support compute and storage intensive workloads for up to 10 TB to satisfy specific execution time or throughput ranges.

2. A bigger Cloudera Hadoop Distribution cluster configuration using more data nodes / task trackers helps improve the execution time and throughput for processing a specific data set when compared with the minimal configuration. For example, doubling the number of data nodes / task trackers from 3 in a minimal configuration to 6 results in reducing the execution time by close to half of what it is with the minimal configuration. So, cost permitting, a bigger cluster will help with getting the data processing done faster with higher throughput.

3. The throughput for the Terasort benchmark (which is a mix of compute and storage intensive operations, with a bigger proportion of the compute intensive operations) averages around 120 MB/sec per node. This can be used as a guide to size the cluster to satisfy specific throughput requirements for a mixed workload similar to the Terasort benchmark.

4. The throughput for the storage intensive TestDFSIO benchmark averages around 75 MB/sec per node for write operations. The average per data node throughput can be used as a guide to size the cluster to satisfy specific throughput requirements for read/write intensive workloads.

5. Resource utilization for the CPU, memory, disks and network on the cluster data nodes / task trackers need to be monitored for saturation.

6. During compute intensive benchmarks like Terasort, the CPU and memory utilization needs to be observed for saturation especially during the initial part of the benchmark which is when the data is being sorted. During the latter part of the benchmark, the disk utilization must be observed for saturation since this is when the sorted data is being written out to disk.

7. During storage intensive benchmarks like TestDFSIO, for resource utilization, more attention needs to be given to the disk and network utilization to see if there is saturation, since throughout the TestDFSIO tests, data is being written to or read from the disks on all the data nodes. With storage intensive benchmarks, the CPU utilization should be expected to be moderate.

8. Resource utilization for the name nodes or job tracker also needs to be monitored. When using the Hitachi Compute Rack 220S servers with the specifications listed in this document, the name nodes are very lightly loaded. As a result, a lower end server having lower end CPUs and slightly less memory can be used for the cluster, thus

reducing the cost of the configuration. For example, a Hitachi Compute Rack 210H server or another equivalent server with 8 or more CPU cores, 32 GB of memory and 6 local disks can likely be sufficient as a name node and secondary name node for a cluster with 8 data nodes / task trackers.

9. The data shared in this reference architecture document can help provide guidelines about sizing a cluster which will satisfy customer execution time or throughput requirements for data processing of specific input data sets tested for this reference architecture.

10. The results discussed in this document could be extrapolated to obtain cluster sizing estimates for input data sets or execution time and throughput requirements not tested for this reference architecture.

For More Information

Hitachi Data Systems Global Services offers experienced storage consultants, proven methodologies and a comprehensive services portfolio to assist you in implementing Hitachi products and solutions in your environment. For more information, see the Hitachi Data Systems Global Services website.

Live and recorded product demonstrations are available for many Hitachi products. To schedule a live demonstration, contact a sales representative. To view a recorded demonstration, see the Hitachi Data Systems Corporate Resources website. Click the **Product Demos** tab for a list of available recorded demonstrations.

Hitachi Data Systems Academy provides best-in-class training on Hitachi products, technology, solutions and certifications. Hitachi Data Systems Academy delivers on-demand web-based training (WBT), classroom-based instructor-led training (ILT) and virtual instructor-led training (vILT) courses. For more information, see the Hitachi Data Systems Services Education website.

For more information about Hitachi products and services, contact your sales representative or channel partner or visit the Hitachi Data Systems website.

Hitachi Data Systems Corporation

Corporate Headquarters
2845 Lafayette Street, Santa Clara, California 95050-2627 USA
www.HDS.com

Regional Contact Information
Americas: +1 408 970 1000 or info@HDS.com
Europe, Middle East and Africa: +44 (0) 1753 618000 or info.emea@HDS.com
Asia-Pacific: +852 3189 7900 or hds.marketing.apac@HDS.com

AS-226-00, May 2013