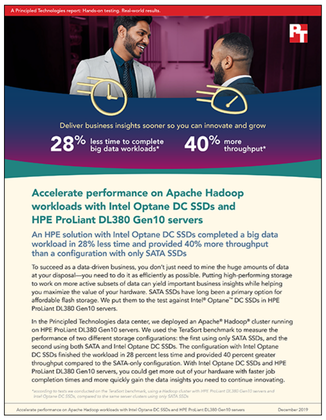


A Principled Technologies report: Hands-on testing. Real-world results.

Deliver business insights sooner so you can innovate and grow
28% less time to complete big data workloads*
40% more throughput*

Accelerate performance on Apache Hadoop workloads with Intel Optane DC SSDs and HPE ProLiant DL380 Gen10 servers

An HPE solution with Intel Optane DC SSDs completed a big data workload in 28% less time and provided 40% more throughput than a configuration with only SATA SSDs

To succeed as a data-driven business, you don't just need to mine the huge amounts of data at your disposal—you need to do it as efficiently as possible. Putting high-performing storage to work on more active subsets of data can yield important business insights while helping you maximize the value of your hardware. SATA SSDs have long been a primary option for affordable flash storage. We put them to the test against Intel® Optane™ DC SSDs in HPE ProLiant DL380 Gen10 servers.

In the Principled Technologies data center, we deployed an Apache® Hadoop® cluster running on HPE ProLiant DL380 Gen10 servers. We used the TeraSort benchmark to measure the performance of two different storage configurations: the first using only SATA SSDs, and the second using both SATA and Intel Optane DC SSDs. The configuration with Intel Optane DC SSDs finished the workload in 28 percent less time and provided 40 percent greater throughput compared to the SATA-only configuration. With Intel Optane DC SSDs and HPE ProLiant DL380 Gen10 servers, you could get more out of your hardware with faster job completion times and more quickly gain the data insights you need to continue innovating.

*according to tests we conducted on the TeraSort benchmark, using a Hadoop cluster with HPE ProLiant DL380 Gen10 servers and Intel Optane DC SSDs, compared to the same server clusters using only SATA SSDs

Accelerate performance on Apache Hadoop workloads with Intel Optane DC SSDs and HPE ProLiant DL380 Gen10 servers December 2019

The science behind the report:

Accelerate performance on Apache Hadoop workloads with Intel Optane DC SSDs and HPE ProLiant DL380 Gen10 servers

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Accelerate performance on Apache Hadoop workloads with Intel Optane DC SSDs and HPE ProLiant DL380 Gen10 servers.

We concluded our hands-on testing on October 30, 2019. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on October 25, 2019 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

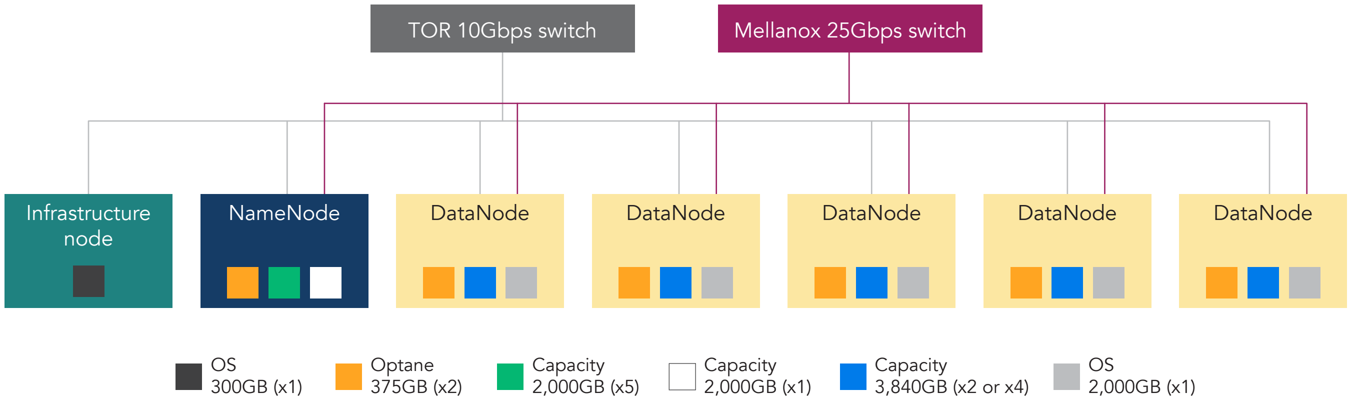
Our results

The table below presents our findings in detail.

Drive type	Iteration	Test runtime (min:sec)	Median time (min:sec)	Intel Optane DC SSDs win (percentage)	Throughput (GB/sec)	Median throughput (GB/sec)	Intel Optane DC SSDs win (percentage)
SATA SSDs	0	19:20	19:20	**28.62%**	736.99	736.99	**40.06%**
	1	19:45			721.38		
	2	19:04			747.30		
Intel® Optane™ DC SSDs	0	13:44	13:48		1,037.34	1,032.26	
	1	13:48			1,032.26		
	2	13:49			1,030.89		

Configuration chart

The chart below depicts how we configured the drives in our Apache® Hadoop® cluster. On the DataNodes, for the Intel Optane DC SSD testing, we enabled the Intel Optane DC SSDs and disabled two of the SATA SSDs, as shown below; for the SATA SSD-only testing, we did not enable the Intel Optane DC SSDs and left all SATA SSDs enabled. For the full details of the configuration, see the How we tested section.



System configuration information

The table below presents detailed information on the systems we tested.

Server configuration information	Hadoop NameNode	Hadoop DataNodes
Tested by	Principled Technologies	Principled Technologies
Test date	2019/10/30	2019/10/30
Workload and version	Intel HiBench 7.0 - Terasort	Intel HiBench 7.0 - Terasort
Server platform	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10
BIOS name and version	U30 v2.10 (05/21/2019)	U30 v2.10 (05/21/2019)
BIOS settings	Performance	Performance
Operating system name and version/build number	CentOS 7 x86_64 18.10 3.10.0-1062.4.1.el7.x86_64	CentOS 7 x86_64 18.10 3.10.0-1062.4.1.el7.x86_64
Date of last OS updates/patches applied	2019/10/30	2019/10/30
Power management policy	Static High Performance Mode	Static High Performance Mode
Processor		
Number of processors	2	2
Vendor and model	Intel Xeon® Platinum 8180	Intel Xeon Gold 6240M
Core count (per processor)	28	18
Core frequency (GHz)	2.5	2.6
Stepping	4	7
Hyper-threading	Yes	Yes
Turbo	Yes	Yes
Memory module(s)		
Total memory in system (GB)	768	384
Number of memory modules	24	12
Vendor and model	Samsung M393A4K40BB1-CRC0Q	Samsung M393A4K40BB2-CTD6Q
Size (GB)	32	32
Type	PC4-2400T	PC4-2666V
Speed (MHz)	2,400	2,666
Speed running in the server (MHz)	2,400	2,666
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM)	768	384
Local storage (OS)		
Number of drives	1	1
Drive vendor and model	HPE 2TB NVMe x4 RI SFF SSD (Intel SSD DC P4500)	HPE 2TB NVMe x4 RI SFF SSD (Intel SSD DC P4500)
Drive size (TB)	2	2
Drive information (interface, type)	NVMe PCIe SSD	NVMe PCIe SSD

Server configuration information	Hadoop NameNode	Hadoop DataNodes	
Local storage (capacity drives))			
Number of drives	5	In scenario using Intel Optane DC SSDs	2
		In scenario using SATA SSDs	4
Drive vendor and model	HPE 2TB NVMe x4 RI SFF SSD (Intel SSD DC P4500)	HPE 3.84TB SATA RI SFF SC DS SSD (Intel SSD D3-S4510)	
Drive size (TB)	2	3.84	
Drive information (interface, type)	NVMe PCIe SSD	SATA SSD	
Local storage (Intel Optane SSD)			
Number of drives	2	In scenario using Intel Optane DC SSDs	2
		In scenario using SATA SSDs	0
Drive vendor and model	HPE 375GB NVMe x4 WI SFF SCN DS SSD (Intel Optane SSD DC P4800X)	HPE 375GB NVMe x4 WI SFF SCN DS SSD (Intel Optane SSD DC P4800X)	
Drive size (GB)	375	375	
Drive information (interface, type)	Intel Optane DC SSD	Intel Optane DC SSD	
Network adapter (OS)			
Vendor and model	HPE Ethernet 1Gb 4-port 331i	HPE Ethernet 1Gb 4-port 331i	
Number and type of ports	4 x 1GbE	4 x 1GbE	
Network adapter (Hadoop data)			
Vendor and model	Intel XXV710	HPE Ethernet 10/25Gb 2-port 661SFP28	
Number and type of ports	2 x 25GbE	2 x 25GbE	
Cooling fans			
Vendor and model	Delta PFM0612XHE	Delta PFM0612XHE	
Number of cooling fans	6	6	
Power supplies			
Vendor and model	HPE 865414-B21	HPE 865414-B21	
Number of power supplies	2	2	
Wattage of each (W)	800	800	

How we tested

For this series of tests, we compared the performance of two Apache Hadoop cluster configurations: one with four SATA drives per DataNode, and the other with two Intel Optane drives and two SATA SSDs per DataNode. For deploying the clusters, we used the Hortonworks Data Platform (v3.1.4) with Apache Ambari (v2.7.3). We performed the comparison using the TeraSort workload from Intel HiBench, a suite of benchmarking software targeting Apache Hadoop and Apache Spark™. Other than the disks, we configured both clusters identically.

Hot and cold data

Storage comprised a hot tier and a cold tier, with the inbuilt assumption that large datasets would live on slower-capacity storage while a smaller subset of that data would be active at any one point in time. While the Hadoop Distributed File System (HDFS) defines explicit HOT and COLD policy options, we used different but similar options to control data location. Due to the failover semantics associated with SSD/DISK versus HOT/COLD, we designated hot-tier data as SSD drives, and designated cold-tier drives as DISK.

Erasure coding and the Intel Intelligent Storage Acceleration Library (ISA-L)

We performed tests using erasure coding with three data blocks, two parity blocks, and a 1024K striping size (In Hadoop, this is RS-3-2-1024K). Erasure coding is a space-efficient alternative to the default three-way replication policy in HDFS. Savings in storage size come at the cost of additional CPU utilization. The version of Hadoop we deployed includes support for erasure coding using either a pure Java™ implementation or an accelerated native implementation using Intel ISA-L. Intel ISA-L uses Intel Advanced Vector Extensions 512 (AVX-512) to efficiently compute the parity required by erasure coding. In our clusters, we used the Intel ISA-L library to reduce the additional CPU burden of erasure coding.

For production Hadoop deployments, RS-10-4-1024K or RS-6-3-1024K policies offer greater data storage efficiency than the RS-3-2-1024K policy we used in our experimental cluster. The RS-3-2-1024K policy has a storage efficiency of 60 percent; while this is an improvement over the 33 percent offered by three-way replication, the RS-10-4-1024K policy, with four parity blocks for every 10 data blocks, has a storage efficiency of 71 percent. This reduced storage size could translate into performance benefits, depending on the specifics of the workload.

Workload sizing and customization

Because the Intel Optane DC SSDs had a smaller capacity than the SATA drives, we sized the workload to fit into the available space (at maximum utilization, we filled the Intel Optane DC SSDs to 94%). By default, Intel HiBench does not provide direct access to configure the workload size, so we modified the terasort.conf configuration file. We also modified the prepare.sh and run.sh scripts to configure erasure coding and drive policy (hot/cold) on the input and output folders of the tests.

Overview of deployment and testing

1. Install operating systems.
2. Configure 25G networking in dual-link LACP groups.
3. Format data drives.
4. Configure hostnames, SSH keys, and hosts file.
5. Install EPEL repository.
6. Install Java, Python, and misc. support.
7. Disable SELinux and firewall.
8. Configure system settings: UMASK, ULIMIT, Frequency Scaling, and Transparent Huge Pages.

9. Install MySQL Server and MySQL JDBC Connector.
10. Install Apache Ambari Server and Ambari Agents.
11. Generate a generic Ambari blueprint for the cluster.
12. Customize the Ambari blueprint's performance tuning and storage settings.
13. Deploy the blueprint and cluster.
14. Build and install the Intel ISA-L library.
15. Install and configure Intel HiBench.
16. Customize the TeraSort workload (size, erasure coding, and storage policy).
17. Initialize HDFS Storage.
18. Prepare the Terasort workload.
19. Execute the Terasort workload.

We describe these steps in greater detail in the following sections.

Installing CentOS 7

1. Boot the server to the CentOS 7 installation media.
2. Select Install CentOS 7, and press Enter.
3. Select English (United States) as the installation language, and click Continue.
4. At the Installation Summary screen, select the following options:
    a. Set the Date and Time settings to the local time zone.
    b. Set the Software Selection to Minimal Install.
    c. Set the Installation Destination as Local Disk, and automatically configure partitioning.
    d. Set the Network and Hostname to DHCP, and turn the Ethernet device on.
5. Click Begin Installation.
6. Enter a root password, and click Done twice.

Configuring networking

On the network switch

1. Identify the ports that are connected to the 25G NIC's ports.
2. Create a Link Aggregation Group (LAG) port group.
3. Configure the LAG for Active Link Aggregation Control Protocol (LACP).
4. Configure the LAG LACP Rate to Fast.
5. Add each of the ports identified in step 1 to the LAG created in step 2.

On each NameNode or DataNode host

1. Boot the server, and log in.
2. Add each participating 25G NIC to a bond device. For example, edit the network script as follows:

```
DEVICE=ens1f0
BOOTPROTO=none
ONBOOT=yes
SLAVE=yes
USERCTL=no
NM_CONTROLLED=no
MASTER=bond1
```

3. Create the bond NIC's network script:

```
DEVICE=bond1
TYPE=Ethernet
ONBOOT=yes
USERCTL=no
NM_CONTROLLED=no
MTU=9000
BOOTPROTO=static
NETWORK=172.16.172.0
IPADDR=172.16.172.17
PREFIX=24
BONDING_OPTS="mode=802.3ad miimon=100 lacp_rate=fast xmit_hash_policy=layer3+4"
```

4. Configure the hostname (if not set during OS installation):

```
vi /etc/hostname
```

5. Enter the hostname, and save the file.
6. Reboot the system:

```
sudo reboot
```

7. Add the static IP addresses assigned in step 3 with the corresponding hostnames to /etc/hosts:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.208.1.151 INFRASTRUCTURE_HOSTNAME
172.16.172.12 DATANODE1_HOSTNAME slave-1.hadoop.local
172.16.172.13 DATANODE2_HOSTNAME slave-2.hadoop.local
172.16.172.14 DATANODE3_HOSTNAME slave-3.hadoop.local
172.16.172.15 DATANODE4_HOSTNAME slave-4.hadoop.local
172.16.172.16 DATANODE5_HOSTNAME slave-5.hadoop.local
172.16.172.17 NAMENODE_HOSTNAME master-1.hadoop.local
```

8. Confirm that all hosts may reach all other hosts:

```
ping -w 1 -t1 172.16.172.12 alpha
```

Configuring storage

On each DataNode host

Note: depending on the scenario you are testing, the type and number of drives will vary. For the baseline configuration, we used two SATA drives for the cold tier and two SATA drives for the hot tier. For the Intel Optane SSD configuration, we used two SATA drives for the cold tier and two Intel Optane SSDs for the hot tier.

1. Boot the server, and log in.
2. Identify the SATA SSD and Intel Optane DC SSD identifiers.
3. For each identified device, format the device using xfs:

```
mkfs -t xfs /dev/XXX
```

4. For each device, create an appropriate mountpoint. We used /data/speed/XXX for hot-tier drives, and /data/capacity/XXX for cold-tier drives:

```
mkdir -p /data/capacity/XXX
mkdir -p /data/speed/XXX
```

5. For each device, mount the device at the specified mountpoint:

```
mount -t xfs -o nodiratime,noatime /dev/XXX /data/capacity/XXX
mount -t xfs -o nodiratime,noatime /dev/XXX /data/speed/XXX
```

6. Add each device to /etc/fstab. For example:

```
/dev/nvme0n1    /data/speed/nvme0n1     xfs     noatime,nodiratime      0 0
/dev/nvme1n1    /data/speed/nvme1n1     xfs     noatime,nodiratime      0 0
/dev/sda        /data/capacity/sda      xfs     noatime,nodiratime      0 0
/dev/sdb        /data/capacity/sdb      xfs     noatime,nodiratime      0 0
```

Upgrading the operating system

1. Boot the server, and log in.
2. Update the operating system:

```
yum -y update
```

3. Reboot the server:

```
reboot
```

Installing utilities

1. Boot the server, and log in.
2. Enable the EPEL repository:

```
yum -y install epel-release
```

3. Install prerequisites:

```
yum -y install git ntp ntpdate openssh-server htop wget curl expect
```

Configuring SSH

On the infrastructure host

1. Boot the server, and log in.
2. Install ssh, ssh-keygen, ssh-copy-id, and sshpass:
   a. For Ubuntu/Debian hosts:

   ```
   apt-get install -y openssh-server openssh-client sshpass
   ```

   b. For CentOS/Red Hat hosts:

   ```
   yum -y install openssh openssh-clients sshpass
   ```

3. Generate a passphrase-less SSH key:

```
mkdir ~/.ssh
chmod 0600 ~/.ssh
ssh-keygen -t rsa -b 4096 -N "" -f ~/.ssh/id_rsa
```

4. Distribute the public key to all hosts:

```
for hostname in DATANODE1_HOSTNAME DATANODE2_HOSTNAME DATANODE3_HOSTNAME DATANODE4_HOSTNAME
DATANODE5_HOSTNAME NAMENODE_HOSTNAME; do
ssh-copy-id -i ~/.ssh/id_rsa root@${hostname}
done
```

5. Distribute the private key to all hosts:

```
for hostname in DATANODE1_HOSTNAME DATANODE2_HOSTNAME DATANODE3_HOSTNAME DATANODE4_HOSTNAME
DATANODE5_HOSTNAME NAMENODE_HOSTNAME; do
ssh root@${hostname} "mkdir ~/.ssh"
ssh root@${hostname} "chmod 0600 ~/.ssh"
rsync ~/id_rsa root@${hostname}:~/.ssh/id_rsa
rsync ~/id_rsa.pub root@${hostname}:~/.ssh/id_rsa.pub
done
```

6. Confirm that each host may SSH to the other hosts without using a password.

Installing Python

On NameNode, DataNode, and infrastructure hosts if using CentOS/Red Hat Enterprise Linux®

1. Boot the server, and log in.
2. Install Development Tools:

```
yum -y groupinstall "Development Tools"
```

3. Install Python:

```
yum -y install python python-devel
```

On infrastructure host, if using Ubuntu/Debian

1. Boot the server, and log in.
2. Install Development Tools:

```
apt-get install -y build-essential
```

3. Install Python:

```
apt-get install -y python python-dev
```

Installing Python pip

On all hosts

1. Boot the server, and log in.
2. Download the pip installer:

```
wget https://bootstrap.pypa.io/get-pip.py -O ~/get-pip.py
```

3. Install pip:

```
python ~/get-pip.py
```

Disabling SELinux

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Disable SELinux:

```
setenforce 0
```

3. Disable SELinux Persistently.
4. Edit /etc/selinux/config to look like the following:

```
SELINUX=disabled
SELINUXTYPE=targeted
```

5. Reboot the server:

```
reboot
```

Disabling Firewall

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Stop Firewall:

```
systemctl stop firewalld
```

3. Disable Firewall:

```
systemctl disable firewalld
```

Setting file system ulimit

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Configure soft ulimit (immediate):

```
ulimit -S 16384
```

3. Configure soft ulimit (persistent):

```
echo '* soft nofile 16384' >> /etc/security/limits.conf
```

4. Configure hard ulimit (immediate):

```
ulimit -H 16384
```

5. Configure hard ulimit (persistent):

```
echo '* hard nofile 16384' >> /etc/security/limits.conf
```

Setting umask

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Configure umask (immediate):

```
umask 0022
```

3. Configure umask (persistent):

```
echo 'umask 0022' >> /etc/profile
```

Disabling frequency scaling

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Disable CPU Frequency Scaling:

```
for CPUFREQ in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do
    echo -n performance > $CPUFREQ
done
```

Disabling Transparent Huge Pages

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Disable Transparent Huge Pages (Immediate):

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

3. Disable Transparent Huge Pages (Persistently):

```
touch /etc/rc.local
chmod
echo 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' >> /etc/rc.local
echo 'echo never > /sys/kernel/mm/transparent_hugepage/defrag' >> /etc/rc.local
```

Installing Java

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Install OpenJDK 8:

```
yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

3. Set JAVA_HOME environment variable system-wide:

```
[ -f /etc/environment ] || (echo "#!/bin/sh" > /etc/environment; chmod +x /etc/environment)
echo "export JAVA_HOME=\"/etc/alternatives/java_sdk/\"" >> /etc/environment
[ -f /etc/profile.d/java.sh ] || (echo "#!/bin/sh" > /etc/profile.d/java.sh; chmod +x /etc/profile.d/
java.sh)
echo 'export JAVA_HOME="/etc/alternatives/java_sdk/"' >> /etc/profile.d/java.sh
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /etc/profile.d/java.sh
```

Installing MySQL community repository

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Download the repository:

```
wget http://repo.mysql.com/mysql-community-release-el7-7.noarch.rpm -O ~/mysql-community-release-
el7-7.noarch.rpm
```

3. Install the repository:

```
rpm -ivh ~/mysql-community-release-el7-7.noarch.rpm
```

Installing MySQL common

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Install MySQL client and common tools:

```
yum -y install mysql-community-client.x86_64 mysql-community-common.x86_64
```

3. Install Python tools for MySQL:

```
pip install PyMySQL
```

Installing MySQL server

On NameNode host

1. Boot the server, and log in.
2. Install MySQL server:

```
yum -y install mysql-server
```

3. Set MySQL Binding IP Address:

```
sed -i 's/^\s*bind-address\s*=.*/bind-address = 0.0.0.0/'
```

4. Set the root password:

```
mysql_secure_installation
```

5. Enable root access from anywhere:

```
mysql -u root
```

   a. Enter the following:

```
GRANT ALL PRIVILEGES ON * . * TO 'root'@'HOSTNAME';
GRANT ALL PRIVILEGES ON * . * TO 'root'@'FQDN';
GRANT ALL PRIVILEGES ON * . * TO 'root'@'127.0.0.1';
GRANT ALL PRIVILEGES ON * . * TO 'root'@'::1';
GRANT ALL PRIVILEGES ON * . * TO 'root'@'localhost';
GRANT ALL PRIVILEGES ON * . * TO 'dba'@'%';
COMMIT;
FLUSH PRIVILEGES;
```

6. Create a database user account:

```
mysql -u root
```

   a. Enter the following:

```
CREATE USER 'dba'@'localhost' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON * . * TO 'dba'@'HOSTNAME';
GRANT ALL PRIVILEGES ON * . * TO 'dba'@'FQDN';
GRANT ALL PRIVILEGES ON * . * TO 'dba'@'127.0.0.1';
GRANT ALL PRIVILEGES ON * . * TO 'dba'@'::1';
GRANT ALL PRIVILEGES ON * . * TO 'dba'@'localhost';
GRANT ALL PRIVILEGES ON * . * TO 'dba'@'%';
COMMIT;
FLUSH PRIVILEGES;
```

Installing MySQL Java Database Connectivity (JDBC) connector

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Install MySQL connector:

```
yum -y install mysql-connector-java*
```

Installing Ambari repository

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Download the repository:

```
wget http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.7.3.0/ambari.repo -O /etc/yum.
repos.d/ambari.repo
```

3. Import the repository's GPG Key:

```
rpm --import http://s3.amazonaws.com/dev.hortonworks.com/ambari/centos7/RPM-GPG-KEY/RPM-GPG-KEY-
Jenkins
```

Installing Ambari server

On NameNode host

1. Boot the server, and log in.
2. Install the package:

```
yum -y install ambari-server
```

3. Stop the Ambari server service:

```
systemctl stop ambari-server
```

4. Disable mandatory installation of Apache SmartSense:

```
sed -ir 's@^\s*<selection>MANDATORY</selection>\s*$//' /var/lib/ambari-server/resources/stacks/
HDP/3.0/services/SMARTSENSE/metainfo.xml
```

5. Create initial database content:

```
mysql -u root
```

   a. Enter the following

```
DROP DATABASE IF EXISTS druid;
CREATE DATABASE druid DEFAULT CHARACTER SET utf8;
CREATE USER 'druid'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'druid'@'%' WITH GRANT OPTION;
DROP DATABASE IF EXISTS superset;
CREATE DATABASE superset DEFAULT CHARACTER SET utf8;
CREATE USER 'superset'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'superset'@'%' WITH GRANT OPTION;

DROP DATABASE IF EXISTS streamline;
CREATE DATABASE streamline;
CREATE USER 'streamline'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'streamline'@'%' WITH GRANT OPTION;
DROP DATABASE IF EXISTS registry;
CREATE DATABASE registry;
CREATE USER 'registry'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'registry'@'%' WITH GRANT OPTION;

CREATE USER 'hive'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'hive'@'%' WITH GRANT OPTION;
CREATE USER 'oozie'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'oozie'@'%' WITH GRANT OPTION;
```

```
CREATE USER 'amb_ranger_admin'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'amb_ranger_admin'@'%' WITH GRANT OPTION;
CREATE USER 'admin'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%' WITH GRANT OPTION;

CREATE USER 'rangerdba'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost' WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%' WITH GRANT OPTION;
CREATE USER 'rangeradmin'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'rangeradmin'@'localhost' WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON *.* TO 'rangeradmin'@'%' WITH GRANT OPTION;
CREATE USER 'rangerkms'@'%' IDENTIFIED BY 'Password_1234';
GRANT ALL PRIVILEGES ON *.* TO 'rangerkms'@'localhost' WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON *.* TO 'rangerkms'@'%' WITH GRANT OPTION;

COMMIT;

FLUSH PRIVILEGES;
```

6. Configure Ambari Server for use with MySQL:

```
ambari-server setup                            \
--databasehost=localhost                       \
--databasename=ambari                          \
--databaseusername=ambari                      \
--databasepassword=Password_1234               \
--databaseport=5432                            \
--database=embedded                            \
-j /etc/alternatives/java_sdk                  \
--enable-lzo-under-gpl-license                 \
--stack-java-home=/etc/alternatives/java_sdk  \
-s > /opt/ambari-install/install.1.log
```

7. Configure Ambari server JDBC connector:

```
ambari-server setup                                     \
--jdbc-db=mysql                                         \
--jdbc-driver=/usr/share/java/mysql-connector-java.jar \
--enable-lzo-under-gpl-license                          \
-s > /opt/ambari-install/install.1.log
```

8. Start Ambari server service:

```
systemctl enable ambari-server
systemctl start ambari-server
```

9. To confirm Ambari server is running, navigate to http://NAMENODE_HOSTNAME:8080/#login.
10. Log in using the username `admin` and the password `admin`

Installing Ambari agent

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Install the package:

```
yum -y install ambari-agent
```

3. Configure the Ambari agent server with the following, replacing `NAMENODE_HOSTNAME` with the hostname of the NameNode.

```
sed -ir 's/^.*hostname=.*$/hostname=NAMENODE_HOSTNAME/' /etc/ambari-agent/conf/ambari-agent.ini
```

4. Start the Ambari agent service:

```
systemctl enable ambari-agent
systemctl start ambari-agent
```

Generating basic Ambari blueprint

1. Navigate to http://NAMENODE_HOSTNAME:8080/#login.
2. Log in using the username `admin` and the password `admin`
3. Click Launch Install Wizard.
4. Enter a cluster name, and click Next.
5. Select HDP-3.1.4.0 if this is not selected by default.
6. Click Use Public Repository.
7. Remove all repositories except Redhat7 by clicking the Remove button that appears when hovering over the repositories.
8. Click Next.
9. In the Target Hosts box, add all of the DataNodes and NameNodes to the list.
10. Click Provide your SSH Private Key to automatically register hosts.
11. Enter the SSH Private key created in the Configuring SSH section.
    Note: you can use `cat ~/.ssh/id_rsa` at the command line to get the key content.
12. Click Register and confirm.
13. Wait for the checks to pass.
14. Click Next.
15. On the Choose Services page, select only the following:
    - YARN + Mapreduce2
    - Tez
    - Hive
    - Pig
    - ZooKeeper
    - Infra Solr
    - Ambari Metrics
    - Kafka
    - Knox
16. Click Next.
    Note: Ignore Limited Functionality Warning messages that pop up by clicking Proceed Anyway.
17. On the Assign Masters page, move all services to the NameNode by selecting the NameNode's hostname in each dropdown. For ZooKeeper Server or other services assigned to multiple hosts, delete all but one entry by clicking the "-" button, and then assign the remaining instance to the NameNode.
18. Click Next.
19. On the Assign Slaves and Clients page, ensure that Client is installed on all hosts, while both DataNode and NodeManager are installed on all DataNode hosts but not NameNode hosts.
20. Click Next.
21. On Credentials page, enter the database passwords used in step 5 of Installing Ambari Server.
22. Click Next.
23. On Databases page, select the Hive database.
24. Select New MySQL from the drop-down menu.
25. Enter MySQL connection details as follows:
    - Database Name: `hive`
    - Database UserName: `hive`
    - Database password: `Password_1234`
    - Database URL: `jdbc:mysql://NAMENODE_HOSTNAME/hive?createDatabaseIfNotExist=true`
    - Hive Database Type: `mysql`
26. Click Next until presented with the Review page.
27. Click Generate Blueprint.
28. Click OK.
29. Save the generated blueprint .zip file.
30. Extract the blueprint.json and clustertemplate.json files. (We will modify this blueprint in the next section.)
31. Close the web browser window.
    Note: Do not click Deploy. We will deploy the blueprint after customization.

Computing general tuning settings

On infrastructure host

1. Download the cloudera yarn-tuning-guide:

```
wget http://tiny.cloudera.com/yarn-tuning-guide -O yarn-tuning-guide.xlsx
```

2. Open the spreadsheet and fill out the required information about cluster geometry.
   In our configuration this was as follows:
   a. Under section labeled Step 1:
      - RAM: `384G`
      - CPU count: `2`
      - CPU cores: `18`
      - Hyperthreading: `yes`

   b. Under section labeled Step 2, set up reservations such that the total amount of reserved matches:
      - Reserve Memory: 12 G
      - Reserve Cores: 2
      - Physical cores to vcores multiplier: 1

   c. Under section labeled Step 3:
      - Workers: `5`

3. Take note of the values that are calculated in steps 6 and 7 in the workbook:

```
yarn.scheduler.minimum-allocation-vcores
yarn.scheduler.maximum-allocation-vcores
yarn.scheduler.increment-allocation-vcores
yarn.scheduler.minimum-allocation-mb
yarn.scheduler.maximum-allocation-mb
yarn.scheduler.increment-allocation-mb

yarn.app.mapreduce.am.resource.cpu-vcores
yarn.app.mapreduce.am.resource.mb
yarn.app.mapreduce.am.command-opts
Use task auto heap sizing
mapreduce.job.heap.memory-mb.ratio
mapreduce.map.cpu.vcores
mapreduce.map.memory.mb
mapreduce.task.io.sort.mb
mapreduce.reduce.cpu.vcores
mapreduce.reduce.memory.mb
```

Customizing the Ambari blueprint

1. In the blueprint.json file, make the following changes.
   Note: You may wish to use an online JSON linter/formatter in order to make the file easier to edit.
2. In core-site.properties:

```
"fs.defaultFS":                                 "hdfs://NAMENODE_HOSTNAME:8020",
"hadoop.tmp.dir":                               "/tmp/hadoop",
"io.compression.codecs":                        "org.apache.hadoop.io.compress.GzipCodec,org.apache.
hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.BZip2Codec,com.hadoop.compression.lzo.
LzoCodec,com.hadoop.compression.lzo.LzopCodec",
"io.compression.codec.lzo.class":               "com.hadoop.compression.lzo.LzoCodec",
"io.compression.codec.lzo.compressor":          "LZO1X_999",
"io.compression.codec.lzo.compression.level":   "3",
"io.erasurecode.codec.rs-legacy.rawcoders":     "rs-legacy_java",
"io.erasurecode.codec.rs.rawcoders":            "rs_native,rs_java",
"io.file.buffer.size":                          "131072",
```

3. In hdfs-site.properties:
   Note: Where applicable, substitute the computed results from steps 6 and 7 of the spreadsheet for the yarn-tuning-guide.xlsx.

```
"dfs.replication":                              "3",
"dfs.replication.max":                          "50",
"dfs.blocksize":                                "536870912",
"dfs.permissions.ContentSummary.subAccess":     "true",
"dfs.permissions.enabled":                      "true",
"dfs.socket.timeout":                           "600000",
"dfs.namenode.name.dir":                        "NAMENODE_PATH_TO_DATA/hadoop/local/name",
"dfs.namenode.checkpoint.dir":                  "NAMENODE_PATH_TO_DATA/hadoop/local/checkpoint",
"dfs.namenode.ec.system.default.policy":        "RS-3-2-1024k",
"dfs.namenode.handler.count":                   "46",
"dfs.namenode.service.handler.count":           "23",
"dfs.datanode.handler.count":                   "80",
"dfs.datanode.max.transfer.threads":            "47999",
"dfs.datanode.socket.write.timeout":            "600000",
"dfs.permissions.superusergroup":               "hdfs",
"dfs.cluster_administrators":                   " hdfs",
"dfs.cluster.administrators":                   " hdfs",
"dfs.datanode.du.reserved":                     "34359738368",
"dfs.namenode.fslock.fair":                     "false",
```

   a. For Optane scenarios:
      Note: This assumes your Intel Optane devices are /dev/nvme1n1 and /dev/nvme2n1.
      Note: This assumes your SATA devices are /dev/sda and /dev/sdb.

```
"dfs.datanode.data.dir":                        "[SSD]/data/speed/nvme1n1/hadoop/hdfs,[SSD]/data/
speed/nvme2n1/hadoop/hdfs,[DISK]/data/capacity/sdc/hadoop/hdfs,[DISK]/data/capacity/sdd/hadoop/
hdfs"
```

   b. For SATA SSD-only scenarios:
      Note: this assumes your SATA devices are /dev/sda, /dev/sdb, /dev/sdc, and /dev/sdd.

```
"dfs.datanode.data.dir":                        "[SSD]/data/capacity/sda/hadoop/hdfs,[SSD]/data/
capacity/sdb/hadoop/hdfs,[DISK]/data/capacity/sdc/hadoop/hdfs,[DISK]/data/capacity/sdd/hadoop/
hdfs"
```

4. In yarn-site.properties:
   Note: where applicable, substitute the computed results steps 6 and 7 of the spreadsheet for the yarn-tuning-guide.xlsx.

```
"yarn.scheduler.minimum-allocation-mb":                    "8192",
"yarn.scheduler.maximum-allocation-mb":                    "360448",
"yarn.scheduler.increment-allocation-mb":                  "512",
"yarn.scheduler.maximum-allocation-vcores":                "70",
"yarn.scheduler.minimum-allocation-vcores":                "1",
"yarn.log-aggregation-enable":                             "False",
"yarn.log-aggregation.retain-seconds":                     "7200",
"yarn.nodemanager.resource.memory-mb":                     "360448",
"yarn.nodemanager.resource.cpu-vcores":                    "70",
"yarn.nodemanager.localizer.cache.target-size-mb":         "65536",
"yarn.nodemanager.vmem-check-enabled":                     "false",
"yarn.nodemanager.pmem-check-enabled":                     "false",
"yarn.nodemanager.aux-services":                           "mapreduce_shuffle",
"yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage": "99",
```

   a. For Optane scenarios:
      Note: This assumes your Intel Optane devices are /dev/nvme1n1 and /dev/nvme2n1.
      Note: This assumes your SATA devices are /dev/sda and /dev/sdb.

```
"yarn.nodemanager.local-dirs":                             "/data/speed/nvme1n1/hadoop/local/
yarn/local,/data/speed/nvme2n1/hadoop/local/yarn/local"
"yarn.nodemanager.log-dirs":                               "/data/speed/nvme1n1/hadoop/local/
yarn/log,/data/speed/nvme2n1/hadoop/local/yarn/log"
```

   b. For SATA SSD-only scenarios:
      Note: This assumes your SATA devices are /dev/sda, /dev/sdb, /dev/sdc, and /dev/sdd.

```
"yarn.nodemanager.local-dirs":                             "/data/capacity/sda/hadoop/local/
yarn/local,/data/capacity/sdb/hadoop/local/yarn/local"
"yarn.nodemanager.log-dirs":                               "/data/capacity/sda/hadoop/local/
yarn/log,/data/capacity/sdb/hadoop/local/yarn/log"
```

5. In mapred-site.properties:
Note: Where applicable, substitute the computed results steps 6 and 7 of the spreadsheet for the yarn-tuning-guide.xlsx.
Note: For "mapreduce.map.java.opts," "yarn.app.mapreduce.am.command-opts," and "mapreduce.reduce.java.opts," compute the maximum amount of memory allowed (-XmxNNNNm) from the corresponding value of "mapreduce.map.memory.mb," "yarn.app.mapreduce.am.resource.mb," or "mapreduce.reduce.memory.mb," respectively. Multiply that number by 0.8 and round it to an integer. Calculate the initial amount of memory allocated (-XmsMMMMm) by dividing the -XmxNNNNm value by 2. For example, "mapreduce.map.memory.mb" is 6144, so Xmx = floor(6144*0.8) = 4915, and Xms = floor(4915 / 2) = 2457.

```
"mapreduce.framework.name":                                         "yarn",
"mapreduce.am.max-attempts":                                        "2",
"yarn.app.mapreduce.am.resource.cpu-vcores":                        "1",
"yarn.app.mapreduce.am.resource.mb":                                "6144",
"mapreduce.map.cpu.vcores":                                         "1",
"mapreduce.map.memory.mb":                                          "6144",
"mapreduce.reduce.cpu.vcores":                                      "1",
"mapreduce.reduce.memory.mb":                                       "12288",
"mapreduce.job.ubertask.enable":                                    "false",
"mapreduce.job.heap.memory-mb.ratio":                               "0.8",
"mapreduce.job.reduce.slowstart.completedmaps":                     "1",
"mapreduce.task.io.sort.mb":                                        "2047",
"mapreduce.task.io.sort.factor":                                    "100",
"mapreduce.reduce.shuffle.memory.limit.percent":                  "0.0001",
"yarn.app.mapreduce.am.job.task.allocation.delay.threshold":  "600",
"mapreduce.reduce.shuffle.parallelcopies":                        "80",
"mapreduce.map.output.compress":                                    "true",
"mapreduce.map.output.compress.codec":                              "org.apache.hadoop.io.compress.
Lz4Codec",
"mapreduce.map.sort.spill.percent":                                 "0.7",
"mapreduce.reduce.speculative":                                     "false",
"mapreduce.map.java.opts":                                          "-Dhdp.version=${hdp.version}
-Djava.net.preferIPv4Stack=true -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps
-XX:+PrintGCDateStamps -Xms2457m -Xmx4915m -XX:ParallelGCThreads=1 -XX:+UseParallelGC
-XX:+UseParallelOldGC -XX:NewSize=1024m -XX:MaxNewSize=2048m",
"yarn.app.mapreduce.am.command-opts":                               "-Dhdp.version=${hdp.version}
-Djava.net.preferIPv4Stack=true -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps
-XX:+PrintGCDateStamps -Xms2457m -Xmx4915m",
"mapreduce.reduce.java.opts":                                       "-Dhdp.version=${hdp.version}
-Djava.net.preferIPv4Stack=true -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps
-XX:+PrintGCDateStamps -Xms4915m -Xmx9830m -XX:ParallelGCThreads=1 -XX:+UseParallelGC
-XX:+UseParallelOldGC -XX:NewSize=1024m -XX:MaxNewSize=2048m",
```

Deploying the Ambari blueprint

On infrastructure host

1. Boot the server, and log in.
2. Deploy the blueprint.
Note: Substitute the NameNode hostname for NAMENODE_HOSTNAME, the path to the blueprint file for path/to/blueprint.json, and the cluster name for BLUEPRINT_NAME:

```
curl                                                \
-H "X-Requested-By: ambari-script"              \
-X POST                                         \
-u admin:admin                                  \
--data-binary "@path/to/blueprint.json"         \
http://NAMENODE_HOSTNAME:8080/api/v1/blueprints/BLUEPRINT_NAME
```

   a. Check that the http response code was 201.

3. Deploy the cluster.
   Note: Substitute the NameNode hostname for NAMENODE_HOSTNAME, the path to the cluster file for path/to/clustertemplate.json, and the cluster name for CLUSTER_NAME:

   ```
   curl                                                 \
   -H "X-Requested-By: ambari-script"               \
   -X POST                                          \
   -u admin:admin                                   \
   --data-binary "@path/to/blueprint.json"          \
   http://NAMENODE_HOSTNAME:8080/api/v1/clusters/CLUSTER_NAME
   ```

   a. Check that the http response code was 202.

4. Wait for the cluster to deploy. You may check the status of the operation by logging into the Ambari server (http://NAMENODE_HOSTNAME:8080/#login), and clicking on the background tasks icon.

Building and installing Intel I-SAL

On NameNode and DataNode hosts

1. Boot the server, and log in.
2. Install Development Tools:

   ```
   yum -y groupinstall "Development Tools"
   ```

3. Install prerequisite packages:

   ```
   yum -y install git autoconf automake libtool yasm
   ```

4. Clone the ISA-L repository:

   ```
   mkdir /opt/isal
   git clone https://github.com/intel/isa-l /opt/isal
   ```

5. Change directories into the cloned repository:

   ```
   cd /opt/isal
   ```

6. Configure the build:

   ```
   ./autogen.sh && ./configure
   ```

7. Build the library:

   ```
   make
   ```

8. Install the library:

   ```
   make install
   ```

9. Copy the library into the location expected by Hadoop:

   ```
   cp /opt/isal/.libs/* /usr/hdp/current/hadoop-client/lib/native/.
   ```

10. Check that HDFS shows native support for erasure coding:

    ```
    sudo -u hdfs hadoop checknative
    ```

    a. If you see either of the following, ISA-L is NOT installed correctly:

       ```
       WARN erasurecode.ErasureCodeNative: ISA-L support is not available in your platform... using
       builtin-java codec where applicable
       ISA-L: false libhadoop was built without ISA-L support
       ```

Installing Intel HiBench

On NameNode host

1. Boot the server, and log in.
2. Install the package:

```
yum -y install maven git vim numpy blas64 lapack64
```

3. Clone the repository:

```
mkdir /opt/hibench
git clone https://github.com/intel-hadoop/HiBench.git /opt/hibench
```

4. Build Intel HiBench:

```
cd /opt/hibench
mvn -Phadoopbench -Dspark=2.1 -Dscala=2.11 clean package
```

Configuring Intel HiBench

On NameNode host

1. Boot the server, and log in.
2. Modify /opt/HiBench/conf/hibench.conf:
    - Set hibench.scale.profile to `large`
    - Set hibench.default.shuffle.parallelism to the number of cores (yarn.scheduler.maximum-allocation-vcores) times the number of nodes.
    - Set hibench.default.map.parallelism to the number of cores (yarn.scheduler.maximum-allocation-vcores) times the number of nodes.
    - Set hibench.masters.hostnames to the hostname of the NameNode.
    - Set hibench.slaves.hostnames to the hostnames of the DataNodes.

3. Modify /opt/HiBench/conf/hadoop.conf. Make the file look like this, substituting the NameNode's hostname for NAMENODE_HOSTNAME:

```
hibench.hadoop.home               /usr/hdp/current/hadoop-client
hibench.hadoop.executable         ${hibench.hadoop.home}/bin/hadoop
hibench.hadoop.configure.dir     ${hibench.hadoop.home}/etc/hadoop
hibench.hdfs.master               hdfs://NAMENODE_HOSTNAME:8020
hibench.hadoop.release            hdp
```

Customizing Intel HiBench TeraSort workload

On NameNode host

1. Boot the server, and log in.
2. Modify /opt/HiBench/conf/workloads/micro/terasort.conf:

```
#datagen
hibench.terasort.tiny.datasize          9375000
hibench.terasort.small.datasize         93750000
hibench.terasort.large.datasize         8550000000
hibench.terasort.huge.datasize          9375000000
hibench.terasort.gigantic.datasize      18750000000
hibench.terasort.bigdata.datasize       150000000000

hibench.workload.datasize               ${hibench.terasort.${hibench.scale.profile}.datasize}

# export for shell script
hibench.workload.input                  ${hibench.hdfs.data.dir}/Terasort/Input
hibench.workload.output                 ${hibench.hdfs.data.dir}/Terasort/Output
```

3. Modify /opt/HiBench/bin/workloads/micro/terasort/prepare/prepare.sh:

```
current_dir='dirname "$0"'
current_dir='cd "$current_dir"; pwd'
root_dir=${current_dir}/../../../../..
workload_config=${root_dir}/conf/workloads/micro/terasort.conf
. "${root_dir}/bin/functions/load_bench_config.sh"
hdfs="${HADOOP_HOME}/bin/hdfs"
enter_bench HadoopPrepareTerasort ${workload_config} ${current_dir}
show_bannar start
rmr_hdfs $OUTPUT_HDFS || true
rmr_hdfs $INPUT_HDFS || true
${hdfs} dfs -mkdir -p $INPUT_HDFS
${hdfs} storagepolicies -setStoragePolicy -path $INPUT_HDFS -policy ALL_SSD
${hdfs} ec -setPolicy -policy RS-3-2-1024k -path $INPUT_HDFS
START_TIME='timestamp'
run_hadoop_job ${HADOOP_EXAMPLES_JAR} teragen -D mapreduce.job.maps=${NUM_MAPS} -D mapreduce.job.
reduces=${NUM_REDS} ${DATASIZE} ${INPUT_HDFS}/data
END_TIME='timestamp'
show_bannar finish
leave_bench
```

4. Modify /opt/HiBench/bin/workloads/micro/terasort/hadoop/run.sh:

```
current_dir='dirname "$0"'
current_dir='cd "$current_dir"; pwd'
root_dir=${current_dir}/../../../../..
workload_config=${root_dir}/conf/workloads/micro/terasort.conf
. "${root_dir}/bin/functions/load_bench_config.sh"
hdfs="${HADOOP_HOME}/bin/hdfs"
enter_bench HadoopTerasort ${workload_config} ${current_dir}
show_bannar start
rmr_hdfs $OUTPUT_HDFS || true
${hdfs} dfs -mkdir -p $OUTPUT_HDFS
${hdfs} storagepolicies -setStoragePolicy -path $OUTPUT_HDFS -policy ALL_SSD
${hdfs} ec -setPolicy -policy RS-3-2-1024k -path $OUTPUT_HDFS
SIZE='dir_size $INPUT_HDFS'
START_TIME='timestamp'
run_hadoop_job ${HADOOP_EXAMPLES_JAR} terasort -D mapreduce.job.reduces=${NUM_REDS}  ${INPUT_HDFS}/
data ${OUTPUT_HDFS}/data
END_TIME='timestamp'
gen_report ${START_TIME} ${END_TIME} ${SIZE}
show_bannar finish
leave_bench
```

Preparing HDFS storage

On NameNode host

1. Boot the server, and log in.
2. Clean out any HDFS trash:

```
sudo -u hdfs hdfs dfs -expunge
sudo -u hdfs hdfs dfs -rmr '/user/*/.Trash'
```

3. Create data directories on HDFS:

```
sudo -u hdfs hdfs dfs -mkdir /HiBench
sudo -u hdfs hdfs dfs -mkdir /user/root
```

4. Set HDFS data directory permissions:

```
sudo -u hdfs hdfs dfs -chown -R root:hadoop /HiBench
sudo -u hdfs hdfs dfs -chown root /user/root
```

Preparing HiBench TeraSort workload

On NameNode host

1. Boot the server, and log in.
2. Run TeraGen to prepare the workload

```
/opt/HiBench/bin/workloads/micro/terasort/prepare/prepare.sh
```

3. Results will be produced in console output, as well as various files in /opt/HiBench/report/terasort. Collect these files.

Running HiBench TeraSort workload

On NameNode host

1. Boot the server, and log in.
2. Run TeraGen to prepare the workload:

```
/opt/HiBench/bin/workloads/micro/terasort/hadoop/run.sh
```

3. The results will appear in console output, as well as various files in /opt/HiBench/report/terasort. Collect these files.

Read the report at http://facts.pt/erw42r3 ▶

This project was commissioned by HPE.

Principled Technologies®

Facts matter.®