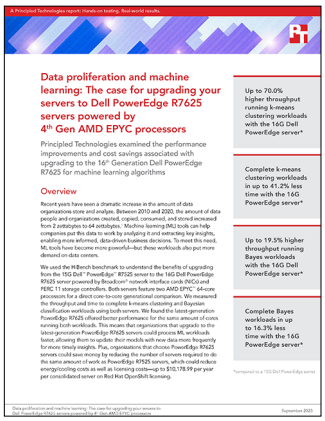

The science behind the report:

Data proliferation and machine learning: The case for upgrading your servers to Dell PowerEdge R7625 servers powered by 4th Gen AMD EPYC processors

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 Data proliferation and machine learning: The case for upgrading your servers to Dell PowerEdge R7625 servers powered by 4th Gen AMD EPYC processors.

We concluded our hands-on testing on July 21, 2023. 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 June 23, 2023 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

We used the HiBench benchmark to understand the benefits of upgrading from the 15G Dell™ PowerEdge™ R7525 server to the 16G Dell PowerEdge R7625 server powered by 4th Gen AMD EPYC™ processors along with Broadcom® network interface cards (NICs) and PERC 11 storage controllers. To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Results of our testing using HiBench version 7.1.1.

	16G Dell™ PowerEdge™ R7625	15G Dell PowerEdge R7525	% Difference
Bayes			
Throughput (B/s)	51,364,850	42,956,387	19.5%
Duration (s)	25.988	31.075	-16.3%
K-means			
Throughput (B/s)	1,384,859,005	814,173,092	70.0%
Duration (s)	172.470	293.361	-41.2%

CPU utilization

K-means clustering workload

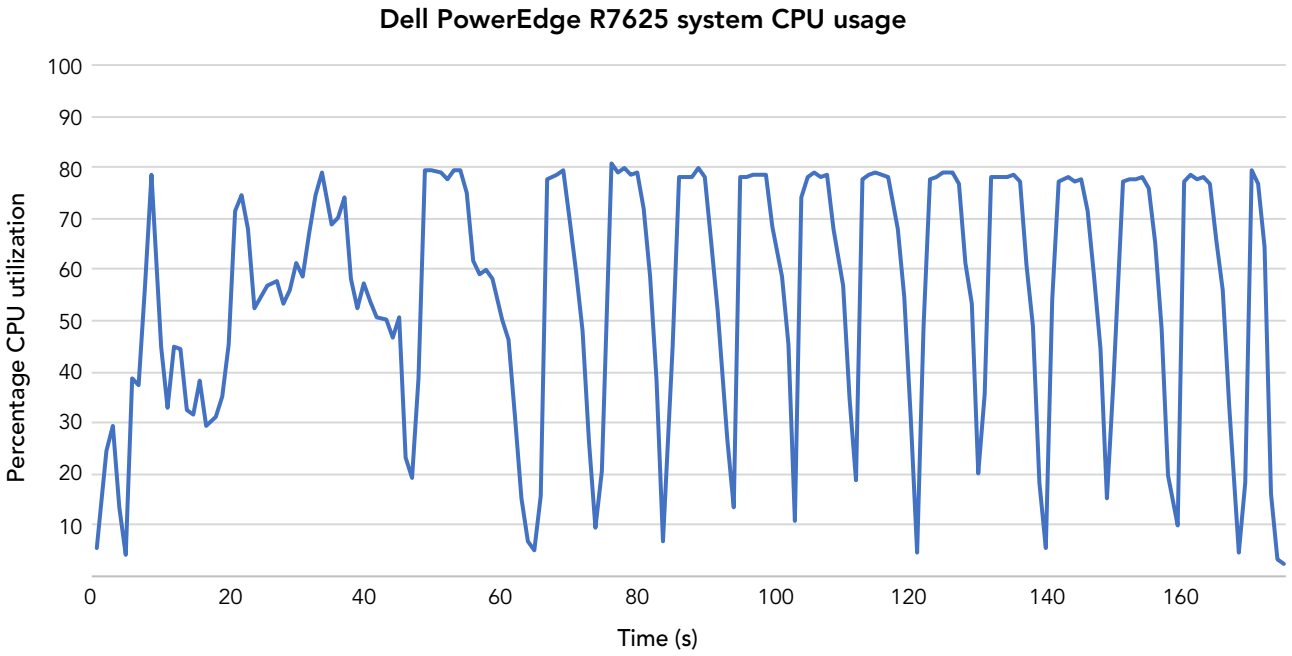

Figure 1: CPU utilization percentage over time in seconds for the 16G Dell PowerEdge R7625 during testing.
Source: Principled Technologies.

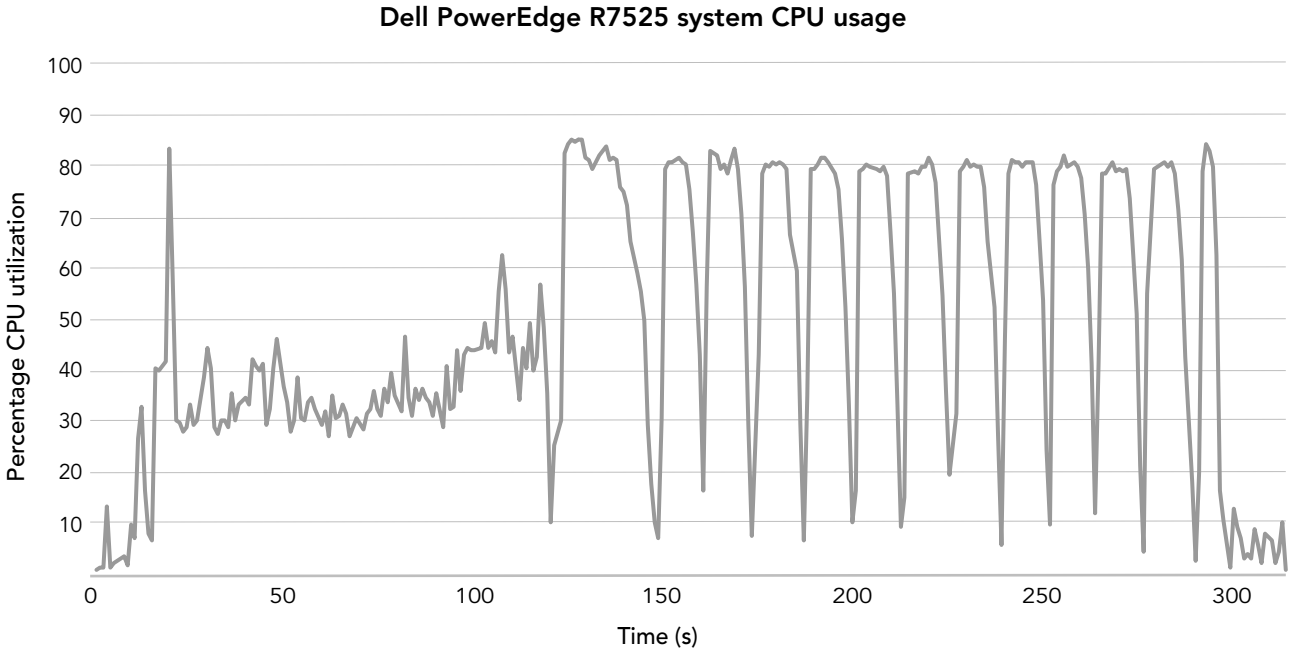

Figure 2: CPU utilization percentage over time in seconds for the 15G Dell PowerEdge R7525 during testing.
Source: Principled Technologies.

Bayesian analysis workload

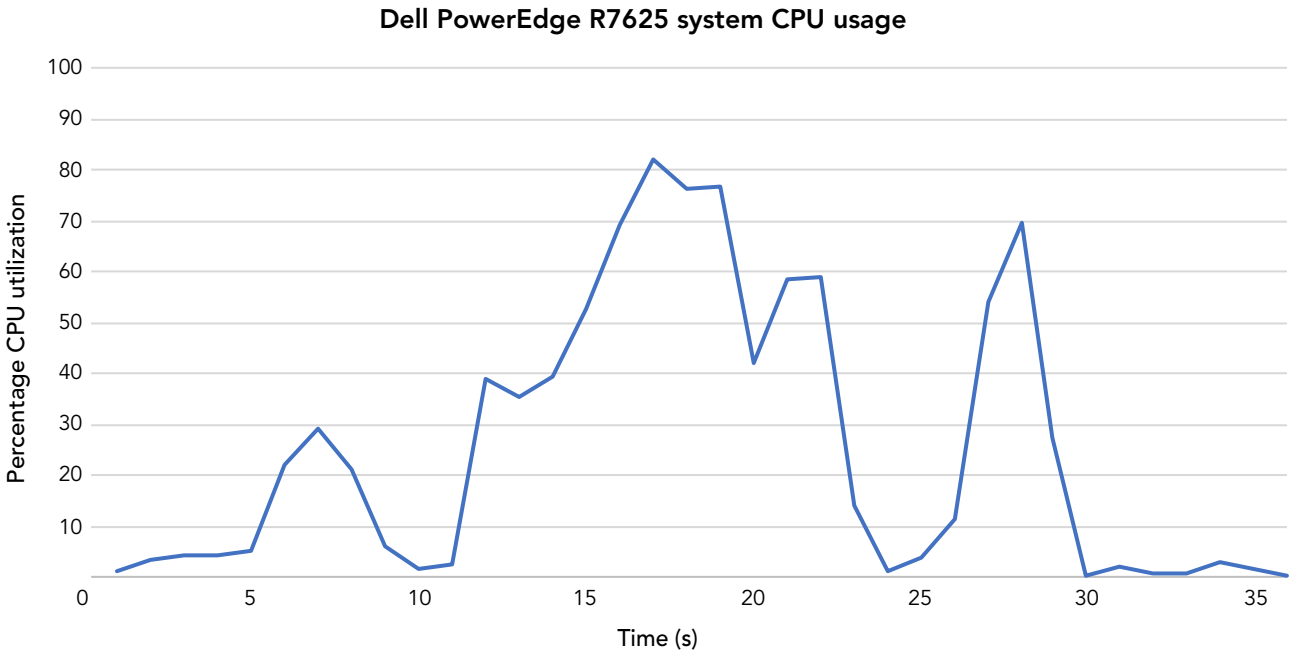

Figure 3: CPU utilization percentage over time in seconds for the 16G Dell PowerEdge R7625 during testing.
Source: Principled Technologies.

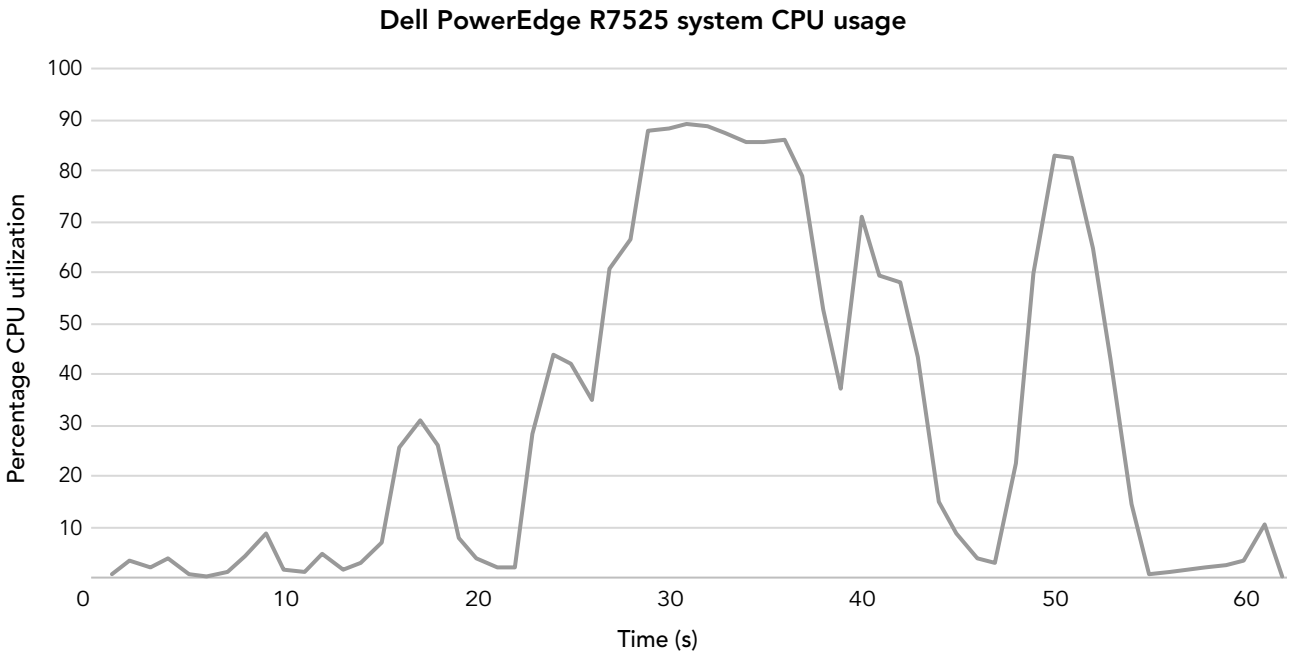

Figure 4: CPU utilization percentage over time in seconds for the 15G Dell PowerEdge R7525 during testing.
Source: Principled Technologies.

System configuration information

Table 2: Detailed information on the systems we tested.

System configuration information	Dell PowerEdge R7625	Dell PowerEdge R7525
BIOS name and version	Dell 1.3.8	Dell 2.8.4
Non-default BIOS settings	N/A	N/A
Power management policy	Performance per watt (OS)	Performance per watt (OS)
Bare metal server operating system		
OS name and version/build number	Red Hat® Enterprise Linux® CoreOS 4.12 4.18.0-372.49.1.el8_6.x86_64	Red Hat Enterprise Linux CoreOS 4.12 4.18.0-372.49.1.el8_6.x86_64
Date of last OS updates	March 24, 2023	March 24, 202
Virtual machine properties – Spark manager		
OS name and version/build number	Red Hat Enterprise Linux 8.8 4.18.0-477.13.1.el8_8.x86_64	Red Hat Enterprise Linux 8.8 4.18.0-477.13.1.el8_8.x86_64
Date of last OS updates	June 23, 2023	June 23, 2023
Number of cores	16	16
Memory size (GB)	32	32
Total storage (GB)	30	30
Number of VMs	1	1
Virtual machine properties – Spark worker		
OS name and version/build number	Red Hat Enterprise Linux 8.8 4.18.0-477.13.1.el8_8.x86_64	Red Hat Enterprise Linux 8.8 4.18.0-477.13.1.el8_8.x86_64
Date of last OS updates	June 23, 2023	June 23, 2023
Number of cores	24	24
Memory size (GB)	96	96
Total storage (GB)	30	30
Number of VMs	10	10
Processor		
Number of processors	2	2
Vendor and model	AMD EPYC™ 9554	AMD EPYC 7763
Core count (per processor)	64	64
Core frequency (GHz)	3.10	2.45
Stepping	1	1
Memory module(s)		
Total memory in system (GB)	3,072	2,048
Number of memory modules	24	16
Vendor and model	Samsung® M321RAGA0B20-CWKZH	Micron® 72ASS16G72LZ-3G2B3

System configuration information	Dell PowerEdge R7625	Dell PowerEdge R7525
Size (GB)	128	128
Type	DDR-5	DDR-4
Speed (MHz)	4,800	3,200
Speed running in the server (MHz)	4,800	3,200
Storage controller		
Vendor and model	Dell PERC H755 Front	Dell PERC H745 Front
Cache size (GB)	8	4
Firmware version	52.21.0-4606	51.16.0-4076
Driver version	megaraid_sas 07.719.03.00-rh1	megaraid_sas 07.719.03.00-rh1
Local storage (OS)		
Number of drives	5	5
Drive vendor and model	Seagate ST1000NX0453	Seagate ST1000NX0453
Drive size (GB)	1,024	1,024
Drive information (speed, interface, type)	7.2K, 12Gb SAS, HDD	7.2K, 12Gb SAS, HDD
Local storage (data)		
Number of drives	4	4
Drive vendor and model	Dell Ent NVMe v2 AGN MU U.2 6.4TB	Dell Ent NVMe v2 AGN MU U.2 6.4TB
Drive size (GB)	6,144	6,144
Drive information (speed, interface, type)	NVMe v2, PCIe, SSD	NVMe v2, PCIe, SSD
Network adapter		
Vendor and model	Broadcom® Gigabit Ethernet BCM5720	Broadcom Gigabit Ethernet BCM5720
Number and type of ports	2 x 1GbE	2 x 1GbE
Firmware version	22.00.5	22.00.6
Cooling fans		
Vendor and model	Dell Gold	Dell Gold
Number of cooling fans	12	12
Power Supplies		
Vendor and model	Dell 01CW9GA05	Dell 01CW9GA07
Number of power supplies	2	2
Wattage of each (W)	1,400	1,400

How we tested

For this project, we tested Spark performance on Dell PowerEdge servers using OpenShift KVM virtualization 4.12. We ran the k-means and Bayes tests from the machine-learning section of the HiBench suite. Our results reflect what customers can expect to see using the latest-generation Dell EMC PowerEdge servers vs. an older generation.

Using our methodology to aid your own deployments

While the methodology below describes in great detail how we accomplished our testing, it is not a deployment guide. However, because we include many basic installation steps for operating systems and testing tools, reading our methodology may help with your own installation.

Creating the base environment

This section contains the steps we took to install and configure Red Hat OpenShift Virtualization. Prior to this, we setup the prerequisite environment for OpenShift. We created DNS A records for the Kubernetes API, internal API, and a DNS A Wildcard for the ingress route. We also created a pfSense gateway VM to our private network traffic out to the public network.

1. On the system you are doing the install from, click the link in the single node installation guide to get to the assisted installer.
2. Click Create Cluster.
3. Give the cluster a cluster name.
4. In the Base Domain field, enter the name for the domain you created in the DNS server.
5. From the drop-down menu, select an OpenShift version, next to Install single node OpenShift (SNO) check the box, and click Next.
6. Click Next.
7. Click Add Host.
8. From the Provisioning type drop-down menu, select Full image file - Download a self-contained ISO.
9. Enter the SSH public key of the account on the server running the assisted installer, and click Generate Discovery ISO.
10. Click Download Discovery ISO.
11. Connect the downloaded discovery ISO to the iDRAC virtual media, and power on the server.
12. Boot the server to the ISO.
13. When the server has booted into Red Hat Enterprise Linux Core OS, return to the assisted installer GUI and wait for the node to be discovered.
14. Once the server status changes to Ready, click Next.
15. On the Storage tab, click Next.
16. On the Networking tab, click Next.
17. On the Review and Create tab, click Install cluster.
18. Once the installation is complete, click Launch OpenShift Console, and follow the DNS entry instructions to get to the Web Console URL.
19. Login to the OpenShift Console with the provided username and password.
20. Under the Operators tab on the left-hand pane, click on OperatorHub.
21. In the search field type Local Storage, and click on the Local Storage operator.
22. Click Install.
23. Leave the defaults, and click Install.
24. Repeat step 20.
25. In the search field type Local Storage, and click on the LVM Storage operator.
26. Click Install.
27. Leave the defaults, and click Install.
28. Once the LVM operator is installed, click Create LVMCluster.
29. Give the cluster a name, and click Create.
30. Repeat step 20.
31. In the search field type OpenShift Virtualization, and click on the OpenShift Virtualization operator.
32. Click Install.
33. Leave the defaults, and click Install.
34. After the operator installs the OpenShift Virtualization page will appear.
35. Click Create HyperConverged.
36. In the Create HyperConverged screen, to add virtualization functionality to your cluster click Create.

37. From the OpenShift Web console, select OperatorHub under the Operators group.
38. In the search box, enter nmstate.
39. Click on the Kubernetes NMState Operator box.
40. Click Install.
41. On the Install Operator options page, click Install.
42. After the operator installation completes, Click on Installed Operators under the Operators Group.
43. Go to the NMState tab, and click Create NMState.
44. Click Create.
45. Open a terminal on the physical server
46. Apply the following manifest to create a bridge to the second network.

```
cat << EOF | oc apply -f -

apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: br1-eth1-policy
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
  maxUnavailable: 3
  desiredState:
    interfaces:
      - name: br1
        description: Linux bridge with eth1 as a port
        type: linux-bridge
        state: up
        ipv4:
          dhcp: true
          enabled: true
          auto-dns: false
        bridge:
          options:
            stp:
              enabled: false
          port:
            - name: eno8303
    dns-resolver:
      config:
        search:
        - example.com
        - example.org
        server:
        - 8.8.8.8
EOF
46. Log off the physical server.
```

Creating the first VM

For each environment, we created and configured one VM to serve as the basis for the others. We installed and configured the Apache Hadoop and Spark distribution on the base VM and then cloned it 10 times to form 10 worker nodes, and one management node. The management node required additional configuration for the HiBench suite, tools for managing the cluster, and running the HiBench tests.

We performed the following steps for each server environment.

Creating the virtual machine

1. Open the Red Hat OpenShift web console.
2. From the Virtualization pulldown menu, open a list of virtual machines by selecting VirtualMachines.
3. Click Create, and select From Catalog.
4. Select the Red Hat Enterprise Linux 8 VM.
5. Click customize machine.

6. Change the VM name to worker1.
7. Click Storage, and set Disk source to Blank, and Disk Size to 30 GiB.
8. Click Optional parameters, and set the CLOUD_USER_PASSWORD to a convenient password.
9. Click NEXT.
10. Click edit by CPU | Memory, and set the CPUs to 24, and the Memory to 96 GiB. Click SAVE.
11. Select Start this VirtualMachine after Creation, and click Create Virtual Machine.

Creating structure for a second network interface

Wait until the VM is created and runs.

1. From the VirtualMachine dashboard, select worker1.
2. Click Network interfaces.
3. Click Add network interface.
4. Keep the generated name, select virtio for the Model, select the Network spark-app, which you created in the second-network section above, and select Bridge for the type.
5. Click Save.
6. Under Actions restart the VM.

Configuring the VM's OS and installing Hadoop and Spark

Wait until the VM restarts.

1. From the VirtualMachine dashboard, select worker1.
2. Select Open web console.
3. Log onto the VM as user cloud-user using the password you entered above.
4. Set the root password, and switch to root:

```
sudo passwd
sudo -s
```

5. Configure SSH for passwordless logins:

```
ssh-keygen -t rsa -q
cp id_rsa.pub authorized_keys
printf "%s\n%s\n" "Host *" "StrictHostKeyChecking no" > ~/.ssh/config
```

6. Disable the firewall:

```
systemctl stop firewalld
systemctl disable firewalld
```

7. Disable SELinux:

```
setenforce 0
sed -i 's/^SELINUX=.*/SELINUX=disabled/' /etc/selinux/config
```

8. Configure the second network interface, enp2s0:

```
nmcli c add type ethernet con-name enp2s0 ifname enp2s0 ipv4.method manual\
  ipv4.addresses 100.67.114.33/24
nmcli c up enp2s0 ifname enp2s0
```

9. Update the hosts file with the IP addresses for the 11 VMs. For example:

```
cat << EOF >> /etc/hosts
#
100.67.114.33   worker1
100.67.114.34   worker2
100.67.114.35   worker3
100.67.114.36   worker4
100.67.114.37   worker5
```

```
100.67.114.38   worker6
100.67.114.39   worker7
100.67.114.40   worker8
100.67.114.41   worker9
100.67.114.42   worker10
100.67.114.31   manager
EOF
```

10. Update the OS:

```
dnf upgrade -y
```

11. Install the prerequisite packages:

```
dnf install -y maven bc python2 wget java-1.8.0-openjdk git blas64 lapack64
```

12. Download Apache Hadoop and Apache Spark:

```
cd ~
wget https://archive.apache.org/dist/spark/spark-3.0.3/spark-3.0.3-bin-hadoop3.2.tgz
wget https://archive.apache.org/dist/hadoop/common/hadoop-3.2.4/hadoop-3.3.1.tar.gz
```

13. Extract the Hadoop and Spark files:

```
cd ~
tar xzf /root/hadoop-3.3.1.tar.gz
tar -xzf /root/spark-3.2.1-bin-hadoop3.3.tgz
```

14. Append the following lines to your bashrc:

```
cat << 'EOF' >> ~/.bashrc
#
export SPARK_HOME=/root/spark-3.0.3-bin-hadoop3.2
export HADOOP_HOME=/root/hadoop-3.2.4
export HADOOP_CONF=$HADOOP_HOME/etc/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_COMMON_LIB_NATIVE_DIR"
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:$HADOOP_COMMON_LIB_NATIVE_DIR"
export YARN_HOME=$HADOOP_HOME
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.372.b07-4.el8.x86_64
export PATH="$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin:$SPARK_HOME/bin"
EOF
```

15. Source the bash file to make use of these variables for the current login session:

```
. ~/.bashrc
```

16. Modify the Spark environmental variable file (`$SPARK_HOME/conf/spark-env.sh`) to contain:

```
export SPARK_LOCAL_DIRS=/root/tmp-dir
export SPARK_MASTER_HOST=manager
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:$HADOOP_HOME/lib/native"
```

17. Modify the Hadoop environment file (`$HADOOP_HOME/etc/conf/hadoop-env.sh`) to contain:

```
export HADOOP_OS_TYPE=${HADOOP_OS_TYPE:-$(uname -s)}
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.372.b07-4.el8.x86_64
```

18. Modify the following Hadoop configuration files in `$HADOOP_HOME/etc/conf/hadoop-env.sh` to contain:

core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://manager:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/root/tmp-dir</value>
  </property>
```

hdfs-site.xml

```
configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/root/hdfs/nn</value>
  </property>
  <property>
    <name>fs.checkpoint.dir</name>
    <value>file:/root/hdfs/snn</value>
  </property>
  <property>
    <name>fs.checkpoint.edits.dir</name>
    <value>file:/root/hdfs/snn</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/root/hdfs/dn</value>
  </property>
</configuration>
```

mapred-site.xml

```
configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>
</configuration>
```

yarn-site.sh

```
configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>manager</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

19. Shutdown the VM:

```
shutdown -h now
```

Creating the remaining nine worker VMs

1. From the OpenShift web console, go to the VirtualMachine dashboard.
2. From the three dots on the right for VM worker1, click Clone.
3. Change the VM name to worker2.
4. Click Clone.
5. Repeat steps 2 through 4 for worker3, worker4, …, worker10.
6. For VMs worker2 through worker10:
    a. Log into the VM's web console as root
    b. Change the hostname to match the VM name; e.g.,:

```
hostnamectl set-hostname worker2 # match the VM name
```

    c. Change the IP address of the second interface to match that in the hosts file; e.g.:

```
nmcli c mod enp2s0 ip4 100.67.114.34
```

7. Reboot the VM:

```
shutdown -r now
```

Creating the remaining the Spark manager VM

1. From the OpenShift web console, go to the VirtualMachine dashboard.
2. From the three dots on the right for VM worker1, click Clone.
3. Change the VM name to manager.
4. Click Clone.
5. After the new VM starts, click on its name from the VirtualMachine Dashboard.
6. On the VirtualMachine Details page, click Details.
7. Click on the pencil icon next to CPU | Memory.
8. Change the number of CPUs to 16, and the Memory to 32GB. Click Save.
9. 9. On the VirtualMachine details page, select Action, and Restart.
10. Wait for the VM to restart.
11. Log into the VM's web console as root.

12. Change the hostname to manager:

```
hostnamectl set-hostname manager
```

13. Change the IP address of the second interface to match that in the hosts file:

```
nmcli c mod enp2s0 ip4 100.67.114.31
```

14. Install HiBench:

```
cd ~
git clone https://github.com/intel-hadoop/HiBench.git
```

15. Compile HiBench for Apache Spark 3.2:

```
cd HiBench
mvn -Dspark.version=3.0.3 -Dscala=2.11 clean package |& tee hibench_build.log
```

16. Modify the HiBench configuration files (`~/HiBench/conf`) with the following information. We use the `bigdata` profile for k-means, and the `gigantic` profile for Bayes. For the latter, the value of the first variable should be changed to `gigantic`:

hibench.conf

```
# note only these three variables are modified
hibench.scale.profile                bigdata
hibench.default.map.parallelism      390
hibench.default.shuffle.parallelism  390
```

hadoop.conf

```
hibench.hadoop.home           /root/hadoop-3.2.4
hibench.hadoop.executable     ${hibench.hadoop.home}/bin/hadoop
hibench.hadoop.configure.dir  ${hibench.hadoop.home}/etc/hadoop
hibench.hdfs.master           hdfs://manager:9000
hibench.hadoop.release        apache
```

spark.conf

```
hibench.spark.home            /root/spark-3.0.3-bin-hadoop3.2
hibench.spark.master          spark://manager:7077
spark.executor.memory         20g
spark.driver.memory           16g
spark.executor.instances      39
spark.executor.cores           5
spark.driver.cores            12
spark.default.parallelism     ${hibench.default.map.parallelism}
spark.sql.shuffle.partitions  ${hibench.default.shuffle.parallelism}
```

Running the tests

In this section, we list the steps to run the k-means and Bayes benchmarks. The scripts that format the HDFS volumes and start/stop the daemons are found at the end of the section. They are installed in director /root/scripts.

1. Log into the manager VM as root.
2. Ensure the Spark and Hadoop daemons are stopped:

```
~/scripts/stop_all.sh
```

3. Format the Hadoop HDFS backing storage:

```
~/scripts/format_hdfs.sh
```

4. Start the Hadoop and Spark daemons:

```
~/scripts/start_all.sh
```

5. To perform one run of the k-means test:
   a. Make sure the HiBench profile in hibench.conf is set to `bigdata`.
   b. Clear the filesystem caches on each VM:

```
  for i in manager worker{1..10} ; do
  printf "\n%s " $i
  ssh $i "free -m ;echo 3 > /proc/sys/vm/drop_caches; free -m"
done
```

   c. Create the test data for k-means:

```
  cd ~/HiBench
./bin/workload/ml/kmeans/prepare/prepare.sh
```

   d. Run the test:

```
  ./bin/workload/ml/kmeans/spark/run.sh
```

   e. Results are in the file report/hibench.report.
6. To perform one run of the Bayes test:
   a. Make sure the HiBench profile in hibench.conf is set to `gigantic`.
   b. Clear the filesystem caches on each VM:

```
  for i in manager worker{1..10} ; do
  printf "\n%s " $i
  ssh $i "free -m ;echo 3 > /proc/sys/vm/drop_caches; free -m"
done
```

   c. Create the test data for Bayes:

```
  cd ~/HiBench
./bin/workload/ml/bayes/prepare/prepare.sh
```

   d. Run the test:

```
  ./bin/workload/ml/bayes/spark/run.sh
```

   e. Results are in the file report/hibench.report.

Testing scripts

format_hdfs.sh

```
#!/bin/bash
for i in manager worker{1..10}; do
  printf "\n%s\n\n" "Formatting HDFS on ${i}"
  ssh ${i} "
    mkdir -p /root/tmp-dir/nonce
    rm -rf /root/hdfs/* /root/tmp-dir/*
    mkdir -p /root/hdfs/{nn,snn,dn}
    mkdir -p /var/log/hadoop/yarn
    $HADOOP_HOME/bin/hdfs namenode -format
  "
done
```

start_all.sh

```
#!/bin/bash
echo manager
ssh manager "
  $HADOOP_HOME/bin/hdfs --daemon start namenode
  $HADOOP_HOME/bin/hdfs --daemon start secondarynamenode
  $HADOOP_HOME/bin/yarn --daemon start resourcemanager
  $HADOOP_HOME/bin/yarn --daemon start nodemanager
  $SPARK_HOME/sbin/start-master.sh
"

sleep 10

for i in worker{1..10}; do
  echo $i
  ssh $i "
    $HADOOP_HOME/bin/hdfs --daemon start datanode
    $SPARK_HOME/sbin/start-slave.sh spark://manager:7077
  "
done
```

stop_all.sh

```
#!/bin/bash
for i in worker{1..10}; do
  echo $i
  ssh $i "
    $SPARK_HOME/sbin/stop-slave.sh
    $HADOOP_HOME/bin/hdfs --daemon stop datanode
  "
done

echo manager
ssh manager "
  $SPARK_HOME/sbin/stop-master.sh
  $HADOOP_HOME/bin/yarn --daemon stop nodemanager
  $HADOOP_HOME/bin/yarn --daemon stop resourcemanager
  $HADOOP_HOME/bin/hdfs --daemon stop secondarynamenode
  $HADOOP_HOME/bin/hdfs --daemon stop namenode
"
```

Read the report at https://facts.pt/HqXEuP2

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®