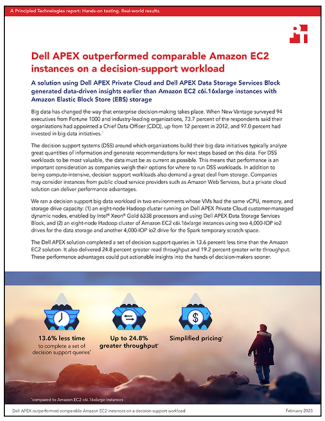

The science behind the report:

Dell APEX outperformed comparable Amazon EC2 instances on a decision-support workload

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 Dell APEX outperformed comparable Amazon EC2 instances on a decision-support workload.

We concluded our hands-on testing on December 7, 2022. 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 December 2, 2022 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

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.

	Dell APEX solution	Amazon EC2 solution	Percentage advantage with Dell APEX solution
Query time in seconds (lower is better)	11,561.82	13,393.29	13.67%
Throughput - Read MB/s (higher is better)	3,477.45	2,784.90	24.87%
Throughput - Write MB/s (higher is better)	1,634.36	1,370.21	19.28%

Pricing

Our goal in this study was to explore the performance potential of two comparably priced solutions. We began by determining monthly pricing our Private Cloud and Data Storage Services solution. As Table 2 shows, the monthly cost of this solution was $13,062.00.

Table 2: The monthly cost of our Dell APEX Private Cloud solution.

Monthly cost for Dell APEX Private Cloud General Purpose 8GB/core 3yr	Monthly cost for Dell APEX Data Storage Services Block Balanced 3yr customer managed	Total monthly cost for Dell APEX solution
$8,911.50	$4,150.50	$ 13,062.00

Next, we set out to configure an Amazon EC2 solution with a matching cost. We first configured a solution with two 1TB drivers per worker, which together formed a large cluster. As Table 3 shows, the monthly cost of this solution was $13,788.67, which is 5.27 percent higher than the pricing of the Dell APEX solution.

Table 3: The monthly cost of the Amazon EC2 solution.

Instance	c6i.16xlarge	1 x c6i.4xlarge
Monthly charge for instance	$874.54	$218.27
Monthly charge for EBS capacity	$256.00	N/A
Monthly charge for EBS provisioned IOPS	$520.00	N/A
Monthly charge for gp3 capacity	$4.80	$327.68
Total monthly charge per instance	$1,655.34	$545.95
Number of instances	8	1
Total monthly charge	$13,242.72	$545.95
Total monthly charge for Amazon EC2 solution		**$13,788.67**

We believed this would provide sufficient space for our data with additional extra room, but once testing began, we learned that Spark used up the extra room for temporary storage and we would need more space. In the interest of time, we added one more 1TB drive per worker. As Table 4 shows, the monthly cost of this solution was $16,892.67, which is 22.67 percent higher than the pricing of the Dell APEX solution.

Table 4: The monthly cost of the Amazon EC2 solution with additional storage per worker.

Instance	c6i.16xlarge	1 x c6i.4xlarge
Monthly charge for instance	$874.54	$218.27
Monthly charge for EBS capacity	$384.00	N/A
Monthly charge for EBS provisioned IOPS	$780.00	N/A
Monthly charge for gp3 capacity	$4.80	$327.68
Total monthly charge per instance	$2,043.34	$545.95
Number of instances	8	1
Total monthly charge	$16,346.72	$545.95
Total monthly charge for Amazon EC2 solution		**$16,892.67**

This extra allocation of space was mostly likely higher than necessary, and the true monthly cost of the Amazon EC2 configuration would fall somewhere between $13,788 and $16,892.

System configuration information

Table 5: Detailed information on the systems we tested.

System configuration information	Dell APEX Private Cloud customer-managed dynamic nodes x 4
BIOS name and version	Dell APEX Private Cloud customer-managed dynamic nodes
Non-default BIOS settings	N/A
Operating system name and version	VMware vSphere® 7.0.3 Build 19193900
Date of last OS updates/patches applied	05/29/2022
System Profile Settings	Performance
Processor	
Number of processors	2
Vendor and model	Intel® Xeon® Gold 6338R
Core count (per processor)	32
Core frequency	2.00
Memory module(s)	
Total memory in system	512
Number of memory modules	16
Vendor and model	Hynix® HMAA4GR7CJR8N-XN
Size (GB)	32
Type	DDR4 DIMM
Speed (MHz)	3,200
Speed running in the server (MHz)	3,200
Storage controller	
Vendor and model	Dell PERC H355i Front
Cache size (MB)	0
Firmware version	17.15.08.00
Driver version	N/A
Local storage	
Number of drives	2
Drive vendor and model	Micron® MTFDDAV480TDS
Drive size (GB)	480
Drive information (speed, interface, type)	6Gbps, M.2, SSD
Network adapter	
Vendor and model	Broadcom® Gigabit Ethernet BCM5720
Number and type of ports	2 x 1Gb & 2 x 10Gb
Firmware version	21.81.3

System configuration information	Dell APEX Private Cloud customer-managed dynamic nodes x 4
Storage adapter	
Vendor and model	Emulex LPe35002-M2-D
Number and type of ports	2-port 32 Gb Fibre Channel
Firmware version	03.05.23
Power Supply	
Vendor and model	Dell® 01CW9GA05
Number of power supplies	2
Wattage of each (W)	1,400

Table 6: Detailed information on the storage we tested.

Storage configuration information	Dell APEX Data Storage Services Block, Balanced Performance Tier
Controller firmware version	2.1.1.1
Performance tier	Balanced
Number of storage controllers	4
Number of storage shelves	2
Total number of drives	20
Drive vendor and model number	Dell 005053081
Drive size (TB)	7.0
Drive information (speed, interface, type)	NVMe SSD

Table 7: Detailed information on the AWS instances we tested.

AWS instance configuration information	c6i.4xlarge	c6i.16xlarge
Tested by	Principled Technologies	Principled Technologies
Test date	12/06/2022	12/06/2022
CSP / Region	AWS® / us-east-1	AWS / us-east-1
Workload & version	Spark™ v3.2.0 tpc-ds-like	Spark v3.2.0 tpc-ds-like
WL specific parameters	4000 scale, 64 executors, 6 executor cores, 12g executor memory	4000 scale, 64 executors, 6 executor cores, 12g executor memory
Iterations and result choice	3 runs, median	3 runs, median
Server platform	c6i.4xlarge	c6i.16xlarge
BIOS name and version	Amazon® EC2 1.0, 10/16/2017	Amazon EC2 1.0, 10/16/2017
Operating system name and version/build number	Ubuntu™ 20.04.5 LTS, kernel 5.15.0-1026-aws	Ubuntu 20.04.5 LTS, kernel 5.15.0-1026-aws
Date of last OS updates/patches applied	12/01/2022	12/01/2022

AWS instance configuration information	c6i.4xlarge	c6i.16xlarge
Processor		
Number of processors	1	1
Vendor and model	Intel® Xeon® Platinum 8375C	Intel Xeon Platinum 8375C
Core count (per processor)	32	32
Core frequency (GHz)	2.9	2.9
Stepping	6	6
Hyper-Threading	Yes	Yes
Turbo	Yes	Yes
Number of vCPU per VM	16	64
Memory module(s)		
Total memory in system (GB)	32	128
NVMe memory present?	No	No
Total memory (GB) (DDR+NVMe RAM)	32	128
General HW		
Storage: NW or Direct Att / Instance	NW	NW
Network BW / Instance (Gbps)	Up to 12.5	25
Storage BW / Instance (Gbps)	Up to 10	20
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	60	60
Drive information (speed, interface, type)	gp3, EBS, 125 MiB/s, 3,000 IOPs	gp3, EBS, 125 MiB/s, 3,000 IOPs
Data drive		
Number of drives	N/A	2
Drive size (GB)	N/A	1,024
Drive information (speed, interface, type)	N/A	io2, EBS, 4000 IOPs
Temporary drive		
Number of drives	1	1
Drive size (GB)	1,024	1,024
Network adapter		
Vendor and model	Amazon Elastic Network Adapter	Amazon Elastic Network Adapter
Number and type of ports (Gbps)	1 x 12.5	1 x 25

How we tested

Testing overview

The goal of our study was to explore the performance capabilities and interoperability of the Dell APEX Data Storage Services solution, and to compare that performance to a similar configuration in Amazon AWS. We were given unfettered and uninterrupted access to the Dell remote lab to perform configuration and testing. We received the Dell environment with a cluster of four APEX Private Cloud nodes, each with two Intel Xeon Gold 6338R processors and 512 GB of RAM. Our cluster also came with VMware vSphere and vCenter installed, and 50 TB of balanced Dell APEX Data Storage Services block storage presented to the cluster as a single datastore via eight 32Gb FC paths (two per node). Client and node network traffic used two redundant 10GBE connections per node.

We started by setting up an Hadoop HDFS cluster with a manager node and eight worker nodes. We installed Apache Spark and Hive on each node to create a big-data decision support database and tables, and to query the database using SQL. We configured the manager node with 16 vCPUs, 32 GB of memory, and two virtual disks (one 4TB virtual disk to house the dataset and one 2TB virtual disk for Spark temporary scratch space). We configured each of our worker nodes with 64 vCPUs, 128 GB of memory, and three virtual disks (two 1TB virtual disks for each data directory and one 1TB virtual disk for Spark temporary scratch space). On AWS, we used the same configuration for CPU, RAM, and storage, using a c6i.4xlarge instance for the manager node and c6i.16xlarge instances for the worker nodes. For data and temporary volumes in AWS, we used EBS io2 volumes with 4,000 IOPs allocated per volume to give us the potential to reach the maximum throughput at large block sizes. Finally, we used Apache Hadoop's built-in yarn resource manager to manage the cluster's resources.

We ran a TPC-DS like workload that ran through a series of 99 queries on each setup and measured the total time it took to complete the queries. To generate the 4,000-scale dataset and queries, we downloaded the TPC-DS tools from the tpc.org website and compiled using their instructions. We also used IBM's spark-tpcds toolkit from https://github.com/IBM/spark-tpc-ds-performance-test as a framework for our custom scripts to run the test. We ran our tests using 64 Spark executors, each with six CPU cores and 12 GB of memory. We ran our test three times on each setup and took the median total time to complete the queries for each.

Creating the Ubuntu 20.04 baseline image on Amazon Web Services (AWS)

This section contains the steps we took to create our baseline image.

1. Download and install the AWS CLI for your OS using instructions provided here: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html.
2. Configure AWS by typing **aws configure** and entering the appropriate details for your AWS account.
3. Create the baseline VM instance:

```
aws ec2 run-instances --image-id ami-0a6b2839d44d781b2 --count 1 \
--instance-type c6i.4xlarge --key-name [AWS Key] \
--block-device-mappings DeviceName=/dev/sda1,VirtualName=Manager-OS,EBS={VolumeSize=60,VolumeType=
gp3} \
--security-group-ids [AWS security group] \
--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=Hadoop-Gold}]'
```

4. Record the Instance Id.

Configuring Ubuntu 20.04 and installing Apache Hadoop and Spark on AWS

1. Log into your instance:

```
ssh -i [key file] ubuntu@[Public IP Address]
```

2. Set the hostname by editing **/etc/hostname**:
3. Modify your hosts file at **/etc/hosts** and add manager IP addresses.
4. Turn off and disable your firewall:

```
sudo ufw disable
```

5. Update your OS:

```
sudo apt update
sudo apt upgrade
```

6. Install Java 8:

```
sudo apt install openjdk-8-jdk
```

7. Reboot.
8. SSH into your manger instance VM.
9. Download Hadoop, Spark, and Hive:

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
wget https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

10. Create default Hadoop directories:

```
mkdir -p /home/ubuntu/hdfs/namenode
mkdir -p /home/ubuntu/hdfs/datanode1
mkdir -p /home/ubuntu/hdfs/datanode2
mkdir -p /home/ubuntu/hdfs/tmp
```

11. Extract the Hadoop, Spark, and Hive compressed files:

```
tar -xzf /home/ubuntu/hadoop-3.3.1.tar.gz
tar -xzf /home/ubuntu/spark-3.2.0-bin-hadoop3.2.tgz
tar -xzf /home/ubuntu/apache-hive-3.1.2-bin.tar.gz
```

12. Set up the Hadoop environment by editing **~/.profile** to match what we have in Scripts we used for testing.
13. Load the Hadoop environment:

```
source ~/.profile
```

14. Edit **core-site.xml**, **hdfs-site.xml**, and **mapred-site.xml** in $HADOOP_HOME/etc/hadoop to match what we have in Scripts we used for testing.
15. Edit **spark-defaults.conf** to match what we have in Scripts we used for testing.
16. Uncomment the JAVA_HOME line in hadoop-env.sh and edit to match the line below:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

17. Shut down the instance.

Creating an image of your baseline instance VM on AWS

1. Create an image with awscli:

```
aws ec2 create-image \
--instance-id [baseline instance id] --name hadoop-gold-ami \
--tag-specifications 'ResourceType=image,Tags=[{Key=Name,Value=hadoop-gold-ami}]'
```

2. Record the image Id.

Creating your instance VMs with the baseline image on AWS

Creating the worker instance VMs from your image

1. Create the worker baseline with awscli:

```
aws ec2 run-instances --image-id [ImageId] --count 1 --instance-type c6i.16xlarge \
--key-name [AWS Key] --security-group-ids [AWS Security Group]
```

2. Record the Instance Id.
3. Create the HDFS data volume:

```
aws ec2 create-volume --availability-zone us-east-1c \
--iops 4000 --volume-type io2 --size 1024
```

4. Record the Volume Id.
5. Repeat steps 3 and 4 two more times for the second data volume and spark temp volume.
6. Attach volumes to the worker instance:

```
aws ec2 attach-volume --device /dev/xvdb \
--instance-id [Worker Instance Id] --volume-id [Data1 Volume Id]

aws ec2 attach-volume --device /dev/xvdc \
--instance-id [Worker Instance Id] --volume-id [Data2 Volume Id]

aws ec2 attach-volume --device /dev/xvdc \
--instance-id [Worker Instance Id] --volume-id [Temp Volume Id]
```

7. SSH into the worker instance.
8. Format the added volumes with an XFS filesystem:

```
sudo mkfs.xfs /dev/<DATA DRIVE 1>
sudo mkfs.xfs /dev/<DATA DRIVE 2>
sudo mkfs.xfs /dev/<TEMP DRIVE>
```

9. Mount the drives to the HDFS data and temp directories:

```
sudo mount /dev/<DATA DRIVE 1> /home/ubuntu/hdfs/datanode1
sudo mount /dev/<DATA DRIVE 2> /home/ubuntu/hdfs/datanode2
sudo mount /dev/<TEMP DRIVE> /home/ubuntu/hdfs/tmp
```

10. Get the UUID of each data and temp drives:

```
lsblk -o NAME,SIZE,MOUNTPOINT,UUID
```

11. Add the drives to **/etc/fstab**:

```
echo "UUID=[Data1 UUID] /home/ubuntu/hdfs/datanode1 xfs defaults 0 0" | sudo tee -a /etc/fstab
echo "UUID=[Data2 UUID] /home/ubuntu/hdfs/datanode2 xfs defaults 0 0" | sudo tee -a /etc/fstab
echo "UUID=[Temp UUID] /home/ubuntu/hdfs/tmp xfs defaults 0 0" | sudo tee -a /etc/fstab
```

12. Create another AMI image using the instructions in the section above for the worker instance VM.
13. Create the remaining seven worker instance VMs:

```
aws ec2 run-instances --image-id [Image Id] --count 7 --instance-type c6i.16xlarge \
--key-name [AWS Key] --security-group-ids [AWS Security Group]
```

Creating the Ubuntu baseline VM in vSphere

This section contains the steps we took to create our baseline VM.

1. Log into the vCenter.
2. Right-click the cluster, and click New Virtual Machine.
3. Select Create a new virtual machine, and click Next.
4. Select a name for the VM and click Next.
5. Select the cluster as the compute resource, and click Next.
6. Select the Dell Apex block storage datastore, and click Next.
7. Select the compatibility, and click Next.
8. Select Ubuntu Linux (64-bit), and click Next.
9. On the Customize hardware tab set the following:
   - 16 vCPU
   - 32 GB memory
   - 60 GB OS disk
   - VMware Paravirtual SCSI controller
10. Click Next.
11. Review, and click Finish.
12. Right-click the newly created VM, and click Edit Settings.
13. For the CD/DVD drive, select Datastore ISO file, navigate to the location with the Ubuntu ISO, and click Connect.
14. Click OK.
15. Power on the VM, and launch the Web Console.
16. Select the Language, and click Enter.
17. Click Continue without updating.
18. Verify the keyboard layout, and click Done.
19. Configure the network, and click Done.
20. To bypass the proxy page, click Done.
21. To use the Mirror address, click Done.
22. Select Use an entire disk, and click Done.
23. Edit the ubuntu-lv and expand to use all available space, and click Done.
24. Click Continue.
25. Enter ubuntu for the name and username, enter manager for the server's name, select a password, and click Done.
26. Check the box to install OpenSSH server, and click Done.
27. Click Done.
28. Click Reboot

Configuring Ubuntu 20.04 and installing Apache Hadoop and Spark in vSphere

1. Log into your VM.
2. Set the hostname by editing **/etc/hostname**:
3. Modify your hosts file at **/etc/hosts** and add manager IP addresses.
4. Turn off and disable your firewall:

```
sudo ufw disable
```

5. Update your OS:

```
sudo apt update
sudo apt upgrade
```

6. Install Java 8:

```
sudo apt install openjdk-8-jdk
```

7. Reboot.
8. SSH into your VM.
9. Download Hadoop, Spark, and Hive:

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
wget https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

10. Create default hadoop directories:

```
mkdir -p /home/ubuntu/hdfs/namenode
mkdir -p /home/ubuntu/hdfs/datanode1
mkdir -p /home/ubuntu/hdfs/datanode2
mkdir -p /home/ubuntu/hdfs/tmp
```

11. Extract the Hadoop, Spark, and Hive compressed files:

```
tar -xzf /home/ubuntu/hadoop-3.3.1.tar.gz
tar -xzf /home/ubuntu/spark-3.2.0-bin-hadoop3.2.tgz
tar -xzf /home/ubuntu/apache-hive-3.1.2-bin.tar.gz
```

12. Set up the Hadoop environment by editing **~/.profile** to match what we have in Scripts we used for testing.
13. Load the Hadoop environment:

```
source ~/.profile
```

14. Edit **core-site.xml**, **hdfs-site.xml**, and **mapred-site.xml** in $HADOOP_HOME/etc/hadoop to match what we have in Scripts we used for testing.
15. Edit **spark-defaults.conf** to match what we have in Scripts we used for testing.
16. Uncomment the JAVA_HOME line in hadoop-env.sh and edit to match the line below:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

17. Shut down the VM.

Cloning baseline VM into worker VMs in vSphere

1. Right-click the VM.
2. Select Clone → Clone to Virtual Machine.
3. Give the VM a name, and click Next.
4. Select the cluster as the compute resource, and click Next.
5. Select the Dell Apex datastore for the storage, and click Next.
6. Click Next.
7. Click Finish.
8. Right-click the cloned VM, and click Edit settings.
9. Change the number of vCPUs to 64.
10. Change the memory to 128 GB.
11. Add three VMware Paravirtual SCSI controllers, and click OK.
12. Right-click the cloned VM, and click Edit settings.
13. Click Add new device, and select Hard Disk.
14. Set the New Hard disk size to 1TB.
15. Set the Virtual Device Node to one of the new SCSI controllers.
16. Repeat steps 13 through 15 two more times for a total of three drives.
17. Click OK.
18. SSH into the worker VM.

19. Format the added volumes with an XFS filesystem:

```
sudo mkfs.xfs /dev/<DATA DRIVE 1>
sudo mkfs.xfs /dev/<DATA DRIVE 2>
sudo mkfs.xfs /dev/<TEMP DRIVE>
```

20. Mount the drives to the HDFS data and temp directories:

```
sudo mount /dev/<DATA DRIVE 1> /home/ubuntu/hdfs/datanode1
sudo mount /dev/<DATA DRIVE 2> /home/ubuntu/hdfs/datanode2
sudo mount /dev/<TEMP DRIVE> /home/ubuntu/hdfs/tmp
```

21. Get the UUID of each data and temp drive:

```
lsblk -o NAME,SIZE,MOUNTPOINT,UUID
```

22. Add the drives to **/etc/fstab**:

```
echo "UUID=[Data1 UUID] /home/ubuntu/hdfs/datanode1 xfs defaults 0 0" | sudo tee -a /etc/fstab
echo "UUID=[Data2 UUID] /home/ubuntu/hdfs/datanode2 xfs defaults 0 0" | sudo tee -a /etc/fstab
echo "UUID=[Temp UUID] /home/ubuntu/hdfs/tmp xfs defaults 0 0" | sudo tee -a /etc/fstab
```

23. Power off the VM.
24. Repeat steps 1 through 7 on the cloned VM for a total of eight worker VMs.

Configuring and starting the cluster

1. Power on the manager and workers.
2. Create and attach two more 4TB drives to the manager.
3. Mount one of the drives to the hdfs tmp directory:

```
sudo mount /dev/<TEMP DRIVE> /home/ubuntu/hdfs/tmp
```

4. Set the hostname on the manager and each of the worker Instances by editing **/etc/hostname**.
5. Add the FQDN, hostname, and IP address of each instance VM to the /etc/hosts file on the manager and worker instances.
6. Verify that you can do passwordless SSH into each instance.
7. Run the **format_hdfs.sh** script in Scripts we used for testing to format the HDFS filesystem on the manager node and each of the worker nodes.
8. Run the **start_all.sh** script in Scripts we used for testing to start the Hadoop namenode, secondary namenode, yarn resource manager, Spark master on the manager node, and the Hadoop datanode, yarn nodemanager, and Spark worker process on each of the worker instances.

Installing the PostgreSQL database for Apache Hive and configuring Hive

1. On the Manager instance VM, install PostgreSQL:

```
sudo apt install postgresql-12
```

2. Edit /etc/postgresql/pga_hba.conf and add the following two lines:

```
local  all ubuntu  trust
Local  all hive    trust
```

3. Uncomment the listen_addresses line in **/etc/postgresql/12/main/postgresql.conf** and change it to the following:

```
listen_addresses = '*'
```

4. Restart PostgreSQL:

```
sudo systemctl restart postgresql
```

5. Edit the hive-site.xml to match what we have in Scripts we used for testing.
6. Create the Hive user and metastore database:

```
psql -u Ubuntu
CREATE USER hive;
ALTER ROLE hive WITH PASSWORD '<PASSWORD>';
CREATE DATABASE metastore;
GRANT ALL PRIVILEGES ON DATABASE metastore TO hive;
\q
```

7. Initialize the hive database:

```
schematool -dbType ubuntu -initSchema
```

8. List tables under the metastore schema to make sure it worked:

```
psql -u Ubuntu -d metastore
\d
\q
```

9. Add the Hive path in ~/.profile, and source the environment:

```
export HIVE_HOME=/home/ubuntu/apache-hive-3.1.2-bin
source ~/.profile
```

10. Download the JDBC driver for PostgreSQL, and copy it to the following locations:

```
/home/ubuntu/spark-3.2.0-bin-hadoop3.2/jars/postgresql-42.5.0.jar
/home/ubuntu/apache-hive-3.1.2-bin/jdbc/postgresql-42.5.0.jar
/home/ubuntu/apache-hive-3.1.2-bin/lib/postgresql-42.5.0.jar
```

Generating the 4TB dataset and creating the TPC-DS like database

1. On the manager instance, create a directory for the dataset creation and the IBM database toolkit:

```
mkdir /home/ubuntu/db
```

2. Create a directory for the dataset:

```
mkdir /home/ubuntu/db/data
```

3. Create an XFS filesystem on the remaining data drive attached to the manager instance:

```
sudo mkfs.xfs /dev/<DATA DRIVE>
```

4. Mount the data drive to the data directory you created above:

```
sudo mount /dev/<DATA DRIVE> /home/ubuntu/db/data
```

5. Download the TPC-DS toolkit to generate the dataset from here: https://www.tpc.org/tpc_documents_current_versions/download_programs/tools-download-request5.asp?bm_type=TPC-DS&bm_vers=3.2.0&mode=CURRENT-ONLY
6. Place the file you just downloaded in **/home/ubuntu/db** and unzip it.
7. Move into the tools subdirectory of the toolkit directory you just unzipped:

```
cd DSGen-software-code-3.2.0rc1/tools
```

8. Install prerequisites:

```
sudo apt install -y gcc make bison flex
```

9. Compile dsdgen and dsqgen:

```
make clean
make
```

10. Create the dataset:

```
dsdgen -scale 4000 -dir /home/ubuntu/db/data -parallel 4 -child 1 &
dsdgen -scale 4000 -dir /home/ubuntu/db/data -parallel 4 -child 2 &
dsdgen -scale 4000 -dir /home/ubuntu/db/data -parallel 4 -child 3 &
dsdgen -scale 4000 -dir /home/ubuntu/db/data -parallel 4 -child 4 &
```

11. Clone the github repository containing the IBM toolkit:

```
git clone https://github.com/IBM/spark-tpc-ds-performance-test.git
```

12. Change directories into the toolkit:

```
cd /home/ubuntu/db/spark-tpc-ds-performance-test
```

13. Copy the necessary files into the working directory:

```
cp src/properties/log4j.properties work/
cp src/ddl/create_database.sql work/
cp src/ddl/create_tables.sql work/
```

14. Create the following HDFS directory for Hive data:

```
hdfs dfs -mkdir -p /user/hadoop/warehouse
```

15. Change into the working directory:

```
cd work/
```

16. Edit **log4j.properties**, **create_database.sql**, and **create_tables.sql** to match what we have in Scripts we used for testing.
17. Go back up a directory to the toolkit home:

```
cd ../
```

18. Load the dataset data into the HDFS filesystem by running the **load_hdfs.sh** script in Scripts we used for testing.
19. Run the following command to create the database:

```
spark-sql \
--driver-memory 6g \
--driver-java-options -Dlog4j.configuration=file:///$PWD/work/log4j.properties \
--executor-cores 2 \
--executor-memory 32g \
--conf \ spark.executor.extraJavaOptions=-Dlog4j.configuration=file:///$PWD/work//log4j.properties \
-f $PWD/work/create_database.sql
```

20. Run the following command to create the tables:

```
spark-sql \
--driver-memory 6g \
--driver-java-options -Dlog4j.configuration=file:///$PWD/work/log4j.properties \
--executor-cores 2 \
--executor-memory 32g \
--conf \ spark.executor.extraJavaOptions=-Dlog4j.configuration=file:///$PWD/work//log4j.properties \
-f $PWD/work/create_tables.sql
```

Generating the query stream for the test

1. Navigate to the tpcds toolkit tools directory:

```
cd /home/ubuntu/db/DSGen-software-code-3.2.0rc1/tools
```

2. Create a directory for the query streams to be placed in:

```
mkdir /home/ubuntu/db/queries
```

3. Generate the query streams:

```
./dsqgen -scale 4000 \
-input ~/db/DSGen-software-code-3.2.0rc1/query_templates/templates.lst \
-dir ~/db/DSGen-software-code-3.2.0rc1/query_templates -streams 1 \
-output_dir ~/db/queries/ -dialect sqlserver
```

4. Rename the query streams:

```
mv ~/db/queries/query_0.sql ~/db/queries/stream0.sql
```

Running the power test

1. Navigate to the testing toolkit:

```
cd /home/ubuntu/db/spark-tpc-ds-performance-test
```

2. Copy the power run query script to run_query.sh:

```
cp run_query.sh.power run_query.sh
```

3. Run the **run.sh** script that we have in Scripts we used for testing, and enter the appropriate configuration values. We used the following:
   - Number of executors: 64
   - Executor cores: 6
   - Executor memory (k, m, g): 12g
4. Repeat steps 1 through 3 two more times for a total of three runs.

Scripts we used for testing

Configuration files

~/.profile

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
      . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

export SPARK_HOME=/home/ubuntu/spark-3.2.0-bin-hadoop3.2

export HADOOP_HOME=/home/ubuntu/hadoop-3.3.1
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_INSTALL=$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_HOME/lib/native"

export HIVE_HOME=/home/ubuntu/apache-hive-3.1.2-bin

export YARN_HOME=$HADOOP_HOME

export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin:$HIVE_HOME/bin:$SPARK_HOME/bin
```

core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://[Manager IP Address]:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/ubuntu/hdfs/tmp</value>
</property>
</configuration>
```

hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
<property>
<name>dfs.name.dir</name>
<value>file:///home/ubuntu/hdfs/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>file:///home/ubuntu/hdfs/datanode1,file:///home/ubuntu/hdfs/datanode2</value>
</property>
<property>
  <name>dfs.client.read.shortcircuit</name>
  <value>true</value>
</property>
<property>
  <name>dfs.domain.socket.path</name>
  <value>/home/ubuntu/hdfs/socket</value>
</property>
</configuration>
```

mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

spark-defaults.conf

```
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Default system properties included when running spark-submit.
# This is useful for setting default environmental settings.

# Example:
# spark.master                     spark://master:7077
# spark.eventLog.enabled           true
# spark.eventLog.dir               hdfs://namenode:8021/directory
# spark.serializer                 org.apache.spark.serializer.KryoSerializer
# spark.driver.memory              5g
# spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"

spark.master                  yarn
spark.eventLog.enabled                true
spark.eventLog.dir                    hdfs://[Manager IP Address]:9000/user/ubuntu/logs
spark.serializer                      org.apache.spark.serializer.KryoSerializer
spark.sql.warehouse.dir               /user/ubuntu/warehouse
```

hive-site.xml

```
<configuration>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:postgresql://localhost:5432/metastore</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>org.postgresql.Driver</value>
</property>

<property>
<name>javax.jdo.option.ConnectionUserName</name>
  <value>hive</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>Password1!</value>
</property>

</configuration>
```

format_hdfs.sh

```
#!/bin/bash

rm -rf /home/ubuntu/hdfs/tmp/*
rm -rf /home/ubuntu/hdfs/namenode/*
rm -rf /home/ubuntu/hdfs/datanode1/*
rm -rf /home/ubuntu/hdfs/datanode2/*
$HADOOP_HOME/bin/hdfs namenode -format

for i in $(cat workers);
do
      ssh ${i} 'rm -rf /home/ubuntu/hdfs/tmp/*'
      ssh ${i} 'rm -rf /home/ubuntu/hdfs/namenode/*'
    ssh ${i} 'rm -rf /home/ubuntu/hdfs/datanode1/*'
    ssh ${i} 'rm -rf /home/ubuntu/hdfs/datanode2/*'
      ssh ${i} '/home/ubuntu/hadoop-3.3.4/bin/hdfs namenode -format'
done
```

load_hdfs.sh

```
#!/bin/bash

cd src/data/

for i in *;
do
      echo "$i, /home/ubuntu/db/data/${i}_[1-4]*.dat"
      hdfs dfs -mkdir -p /user/ubuntu/db/4tb/data/$i/
      hdfs dfs -put -t 4 /home/ubuntu/db/data/${i}_[1-4]*.dat /user/ubuntu/db/4tb/data/$i/
done
```

start_all.sh

```
#!/bin/bash


sudo $HADOOP_HOME/bin/hdfs --daemon start namenode
sudo $HADOOP_HOME/bin/hdfs --daemon start secondarynamenode
sudo $HADOOP_HOME/bin/yarn --daemon start resourcemanager
sudo $SPARK_HOME/sbin/start-master.sh

for i in $(cat ~/scripts/workers);
do
      ssh ${i} 'sudo /home/ubuntu/hadoop-3.3.1/bin/hdfs --daemon start datanode'
      ssh ${i} 'sudo /home/ubuntu/hadoop-3.3.1/bin/yarn --daemon start nodemanager'
      ssh ${i} 'sudo /home/ubuntu/spark-3.2.0-bin-hadoop3.2/sbin/start-worker.sh spark://[Manager
IP Address]:7077'
done
```

stop_all.sh

```
#!/bin/bash

for i in $(cat ~/scripts/workers);
do
      ssh ${i} "sudo /home/ubuntu/hadoop-3.3.1/bin/hdfs --daemon stop datanode"
      ssh ${i} "sudo /home/ubuntu/hadoop-3.3.1/bin/yarn --daemon stop nodemanager"
      ssh ${i} "sudo /home/ubuntu/spark-3.2.0-bin-hadoop3.2/sbin/stop-worker.sh spark://manager:7077"
done

sudo $SPARK_HOME/sbin/stop-master.sh
sudo $HADOOP_HOME/bin/yarn --daemon stop resourcemanager
sudo $HADOOP_HOME/bin/hdfs --daemon stop secondarynamenode
sudo $HADOOP_HOME/bin/hdfs --daemon stop namenode
```

log4j.properties

```
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Set everything to be logged to the console
###Custom log file
log4j.rootCategory=INFO, file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=/home/ubuntu/db/spark-tpc-ds-performance-test/log/spark-tpcds.log
log4j.appender.file.ImmediateFlush=true
## Set the append to false, overwrite
log4j.appender.file.Append=false
log4j.appender.file.MaxFileSize=100MB
log4j.appender.file.MaxBackupIndex=10
##Define the layout for file appender
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

# Set the default spark-shell log level to WARN. When running the spark-shell, the
# log level for this class is used to overwrite the root logger's log level, so that
# the user can have different defaults for the shell and regular Spark apps.
log4j.logger.org.apache.spark.repl.Main=WARN

# Settings to quiet third party logs that are too verbose
log4j.logger.org.spark_project.jetty=WARN
log4j.logger.org.spark_project.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
log4j.logger.org.apache.parquet=ERROR
log4j.logger.parquet=ERROR

# SPARK-9183: Settings to avoid annoying messages when looking up nonexistent UDFs in SparkSQL
with Hive support
log4j.logger.org.apache.hadoop.hive.metastore.RetryingHMSHandler=FATAL
log4j.logger.org.apache.hadoop.hive.ql.exec.FunctionRegistry=ERROR
```

create_database.sql

```
CREATE DATABASE IF NOT EXISTS TPCDS
  COMMENT 'For TPCDS at 4TB scale factor';
```

create_tables.sql

```
--------------------------------------------------------------------------------
-- Licensed Materials - Property of IBM
--
-- (C) COPYRIGHT International Business Machines Corp. 2014
-- All Rights Reserved.
--
-- US Government Users Restricted Rights - Use, duplication or
-- disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
--------------------------------------------------------------------------------

USE TPCDS;

drop table if exists call_center_text;
create table call_center_text
(
cc_call_center_sk         int,
cc_call_center_id         string,
cc_rec_start_date         string,
cc_rec_end_date           string,
cc_closed_date_sk         int,
cc_open_date_sk           int,
cc_name                   string,
cc_class                  string,
cc_employees              int,
cc_sq_ft                  int,
cc_hours                  string,
cc_manager                string,
cc_mkt_id                 int,
cc_mkt_class              string,
cc_mkt_desc               string,
cc_market_manager         string,
cc_division               int,
cc_division_name          string,
cc_company                int,
cc_company_name           string,
cc_street_number          string,
cc_street_name            string,
cc_street_type            string,
cc_suite_number           string,
cc_city                   string,
cc_county                 string,
cc_state                  string,
cc_zip                    string,
cc_country                string,
cc_gmt_offset             double,
cc_tax_percentage         double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/call_center"
;
drop table if exists call_center;
create table call_center
using parquet
as (select * from call_center_text)
;
drop table if exists call_center_text;

drop table if exists catalog_page_text;
create table catalog_page_text
(
cp_catalog_page_sk        int,
cp_catalog_page_id        string,
cp_start_date_sk          int,
cp_end_date_sk            int,
cp_department             string,
cp_catalog_number         int,
cp_catalog_page_number    int,
```

```
cp_description            string,
cp_type                   string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/catalog_page"
;
drop table if exists catalog_page;
create table catalog_page
using parquet
as (select * from catalog_page_text)
;
drop table if exists catalog_page_text;

drop table if exists catalog_returns_text;
create table catalog_returns_text
(
cr_returned_date_sk       int,
cr_returned_time_sk       int,
cr_item_sk                int,
cr_refunded_customer_sk   int,
cr_refunded_cdemo_sk      int,
cr_refunded_hdemo_sk      int,
cr_refunded_addr_sk       int,
cr_returning_customer_sk  int,
cr_returning_cdemo_sk     int,
cr_returning_hdemo_sk     int,
cr_returning_addr_sk      int,
cr_call_center_sk         int,
cr_catalog_page_sk        int,
cr_ship_mode_sk           int,
cr_warehouse_sk           int,
cr_reason_sk              int,
cr_order_number           int,
cr_return_quantity        int,
cr_return_amount          double,
cr_return_tax             double,
cr_return_amt_inc_tax     double,
cr_fee                    double,
cr_return_ship_cost       double,
cr_refunded_cash          double,
cr_reversed_charge        double,
cr_store_credit           double,
cr_net_loss               double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/catalog_returns"
;
drop table if exists catalog_returns;
create table catalog_returns
using parquet
as (select * from catalog_returns_text)
;
drop table if exists catalog_returns_text;


drop table if exists catalog_sales_text;
create table catalog_sales_text
(
cs_sold_date_sk           int,
cs_sold_time_sk           int,
cs_ship_date_sk           int,
cs_bill_customer_sk       int,
cs_bill_cdemo_sk          int,
cs_bill_hdemo_sk          int,
cs_bill_addr_sk           int,
cs_ship_customer_sk       int,
cs_ship_cdemo_sk          int,
cs_ship_hdemo_sk          int,
```

```
cs_ship_addr_sk           int,
cs_call_center_sk         int,
cs_catalog_page_sk        int,
cs_ship_mode_sk           int,
cs_warehouse_sk           int,
cs_item_sk                int,
cs_promo_sk               int,
cs_order_number           int,
cs_quantity               int,
cs_wholesale_cost         double,
cs_list_price             double,
cs_sales_price            double,
cs_ext_discount_amt       double,
cs_ext_sales_price        double,
cs_ext_wholesale_cost     double,
cs_ext_list_price         double,
cs_ext_tax                double,
cs_coupon_amt             double,
cs_ext_ship_cost          double,
cs_net_paid               double,
cs_net_paid_inc_tax       double,
cs_net_paid_inc_ship      double,
cs_net_paid_inc_ship_tax  double,
cs_net_profit             double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/catalog_sales"
;
drop table if exists catalog_sales;
create table catalog_sales
using parquet
as (select * from catalog_sales_text)
;
drop table if exists catalog_sales_text;

drop table if exists customer_text;
create table customer_text
(
c_customer_sk             int,
c_customer_id             string,
c_current_cdemo_sk        int,
c_current_hdemo_sk        int,
c_current_addr_sk         int,
c_first_shipto_date_sk    int,
c_first_sales_date_sk     int,
c_salutation              string,
c_first_name              string,
c_last_name               string,
c_preferred_cust_flag     string,
c_birth_day               int,
c_birth_month             int,
c_birth_year              int,
c_birth_country           string,
c_login                   string,
c_email_address           string,
c_last_review_date        string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/customer"
;
drop table if exists customer;
create table customer
using parquet
as (select * from customer_text)
;
drop table if exists customer_text;
```

```
drop table if exists customer_address_text;
create table customer_address_text
(
ca_address_sk             int,
ca_address_id             string,
ca_street_number          string,
ca_street_name            string,
ca_street_type            string,
ca_suite_number           string,
ca_city                   string,
ca_county                 string,
ca_state                  string,
ca_zip                    string,
ca_country                string,
ca_gmt_offset             double,
ca_location_type          string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/customer_address"
;
drop table if exists customer_address;
create table customer_address
using parquet
as (select * from customer_address_text)
;
drop table if exists customer_address_text;

drop table if exists customer_demographics_text;
create table customer_demographics_text
(
cd_demo_sk                int,
cd_gender                 string,
cd_marital_status         string,
cd_education_status       string,
cd_purchase_estimate      int,
cd_credit_rating          string,
cd_dep_count              int,
cd_dep_employed_count     int,
cd_dep_college_count      int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/customer_demographics"
;
drop table if exists customer_demographics;
create table customer_demographics
using parquet
as (select * from customer_demographics_text)
;
drop table if exists customer_demographics_text;

drop table if exists date_dim_text;
create table date_dim_text
(
d_date_sk                 int,
d_date_id                 string,
d_date                    string,
d_month_seq               int,
d_week_seq                int,
d_quarter_seq             int,
d_year                    int,
d_dow                     int,
d_moy                     int,
d_dom                     int,
d_qoy                     int,
d_fy_year                 int,
d_fy_quarter_seq          int,
d_fy_week_seq             int,
d_day_name                string,
```

```
d_quarter_name            string,
d_holiday                 string,
d_weekend                 string,
d_following_holiday       string,
d_first_dom               int,
d_last_dom                int,
d_same_day_ly             int,
d_same_day_lq             int,
d_current_day             string,
d_current_week            string,
d_current_month           string,
d_current_quarter         string,
d_current_year            string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/date_dim"
;
drop table if exists date_dim;
create table date_dim
using parquet
as (select * from date_dim_text)
;
drop table if exists date_dim_text;

drop table if exists household_demographics_text;
create table household_demographics_text
(
hd_demo_sk                int,
hd_income_band_sk         int,
hd_buy_potential          string,
hd_dep_count              int,
hd_vehicle_count          int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/household_demographics"
;
drop table if exists household_demographics;
create table household_demographics
using parquet
as (select * from household_demographics_text)
;
drop table if exists household_demographics_text;

drop table if exists income_band_text;
create table income_band_text
(
ib_income_band_sk         int,
ib_lower_bound            int,
ib_upper_bound            int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/income_band"
;
drop table if exists income_band;
create table income_band
using parquet
as (select * from income_band_text)
;
drop table if exists income_band_text;

drop table if exists inventory_text;
create table inventory_text
(
inv_date_sk               int,
inv_item_sk               int,
inv_warehouse_sk          int,
```

```
inv_quantity_on_hand      bigint
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/inventory"
;
drop table if exists inventory;
create table inventory
using parquet
as (select * from inventory_text)
;
drop table if exists inventory_text;

drop table if exists item_text;
create table item_text
(
i_item_sk                 int,
i_item_id                 string,
i_rec_start_date          string,
i_rec_end_date            string,
i_item_desc               string,
i_current_price           double,
i_wholesale_cost          double,
i_brand_id                int,
i_brand                   string,
i_class_id                int,
i_class                   string,
i_category_id             int,
i_category                string,
i_manufact_id             int,
i_manufact                string,
i_size                    string,
i_formulation             string,
i_color                   string,
i_units                   string,
i_container               string,
i_manager_id              int,
i_product_name            string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/item"
;
drop table if exists item;
create table item
using parquet
as (select * from item_text)
;
drop table if exists item_text;

drop table if exists promotion_text;
create table promotion_text
(
p_promo_sk                int,
p_promo_id                string,
p_start_date_sk           int,
p_end_date_sk             int,
p_item_sk                 int,
p_cost                    double,
p_response_target         int,
p_promo_name              string,
p_channel_dmail           string,
p_channel_email           string,
p_channel_catalog         string,
p_channel_tv              string,
p_channel_radio           string,
p_channel_press           string,
p_channel_event           string,
p_channel_demo            string,
p_channel_details         string,
```

```
p_purpose                 string,
p_discount_active         string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/promotion"
;
drop table if exists promotion;
create table promotion
using parquet
as (select * from promotion_text)
;
drop table if exists promotion_text;

drop table if exists reason_text;
create table reason_text
(
r_reason_sk               int,
r_reason_id               string,
r_reason_desc             string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/reason"
;
drop table if exists reason;
create table reason
using parquet
as (select * from reason_text)
;
drop table if exists reason_text;

drop table if exists ship_mode_text;
create table ship_mode_text
(
sm_ship_mode_sk           int,
sm_ship_mode_id           string,
sm_type                   string,
sm_code                   string,
sm_carrier                string,
sm_contract               string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/ship_mode"
;
drop table if exists ship_mode;
create table ship_mode
using parquet
as (select * from ship_mode_text)
;
drop table if exists ship_mode_text;

drop table if exists store_text;
create table store_text
(
s_store_sk                int,
s_store_id                string,
s_rec_start_date          string,
s_rec_end_date            string,
s_closed_date_sk          int,
s_store_name              string,
s_number_employees        int,
s_floor_space             int,
s_hours                   string,
s_manager                 string,
s_market_id               int,
s_geography_class         string,
s_market_desc             string,
```

```
s_market_manager          string,
s_division_id             int,
s_division_name           string,
s_company_id              int,
s_company_name            string,
s_street_number           string,
s_street_name             string,
s_street_type             string,
s_suite_number            string,
s_city                    string,
s_county                  string,
s_state                   string,
s_zip                     string,
s_country                 string,
s_gmt_offset              double,
s_tax_precentage          double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/store"
;
drop table if exists store;
create table store
using parquet
as (select * from store_text)
;
drop table if exists store_text;

drop table if exists store_returns_text;
create table store_returns_text
(
sr_returned_date_sk       int,
sr_return_time_sk         int,
sr_item_sk                int,
sr_customer_sk            int,
sr_cdemo_sk               int,
sr_hdemo_sk               int,
sr_addr_sk                int,
sr_store_sk               int,
sr_reason_sk              int,
sr_ticket_number          int,
sr_return_quantity        int,
sr_return_amt             double,
sr_return_tax             double,
sr_return_amt_inc_tax     double,
sr_fee                    double,
sr_return_ship_cost       double,
sr_refunded_cash          double,
sr_reversed_charge        double,
sr_store_credit           double,
sr_net_loss               double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/store_returns"
;
drop table if exists store_returns;
create table store_returns
using parquet
as (select * from store_returns_text)
;
drop table if exists store_returns_text;

drop table if exists store_sales_text;
create table store_sales_text
(
ss_sold_date_sk           int,
ss_sold_time_sk           int,
ss_item_sk                int,
ss_customer_sk            int,
```

```
ss_cdemo_sk               int,
ss_hdemo_sk               int,
ss_addr_sk                int,
ss_store_sk               int,
ss_promo_sk               int,
ss_ticket_number          int,
ss_quantity               int,
ss_wholesale_cost         double,
ss_list_price             double,
ss_sales_price            double,
ss_ext_discount_amt       double,
ss_ext_sales_price        double,
ss_ext_wholesale_cost     double,
ss_ext_list_price         double,
ss_ext_tax                double,
ss_coupon_amt             double,
ss_net_paid               double,
ss_net_paid_inc_tax       double,
ss_net_profit             double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/store_sales"
;
drop table if exists store_sales;
create table store_sales
using parquet
as (select * from store_sales_text)
;
drop table if exists store_sales_text;

drop table if exists time_dim_text;
create table time_dim_text
(
t_time_sk                 int,
t_time_id                 string,
t_time                    int,
t_hour                    int,
t_minute                  int,
t_second                  int,
t_am_pm                   string,
t_shift                   string,
t_sub_shift               string,
t_meal_time               string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/time_dim"
;
drop table if exists time_dim;
create table time_dim
using parquet
as (select * from time_dim_text)
;
drop table if exists time_dim_text;

drop table if exists warehouse_text;
create table warehouse_text
(
w_warehouse_sk            int,
w_warehouse_id            string,
w_warehouse_name          string,
w_warehouse_sq_ft         int,
w_street_number           string,
w_street_name             string,
w_street_type             string,
w_suite_number            string,
w_city                    string,
w_county                  string,
w_state                   string,
```

```
w_zip                     string,
w_country                 string,
w_gmt_offset              double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/warehouse"
;
drop table if exists warehouse;
create table warehouse
using parquet
as (select * from warehouse_text)
;
drop table if exists warehouse_text;

drop table if exists web_page_text;
create table web_page_text
(
wp_web_page_sk            int,
wp_web_page_id            string,
wp_rec_start_date         string,
wp_rec_end_date           string,
wp_creation_date_sk       int,
wp_access_date_sk         int,
wp_autogen_flag           string,
wp_customer_sk            int,
wp_url                    string,
wp_type                   string,
wp_char_count             int,
wp_link_count             int,
wp_image_count            int,
wp_max_ad_count           int
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/web_page"
;
drop table if exists web_page;
create table web_page
using parquet
as (select * from web_page_text)
;
drop table if exists web_page_text;

drop table if exists web_returns_text;
create table web_returns_text
(
wr_returned_date_sk       int,
wr_returned_time_sk       int,
wr_item_sk                int,
wr_refunded_customer_sk   int,
wr_refunded_cdemo_sk      int,
wr_refunded_hdemo_sk      int,
wr_refunded_addr_sk       int,
wr_returning_customer_sk  int,
wr_returning_cdemo_sk     int,
wr_returning_hdemo_sk     int,
wr_returning_addr_sk      int,
wr_web_page_sk            int,
wr_reason_sk              int,
wr_order_number           int,
wr_return_quantity        int,
wr_return_amt             double,
wr_return_tax             double,
wr_return_amt_inc_tax     double,
wr_fee                    double,
wr_return_ship_cost       double,
wr_refunded_cash          double,
wr_reversed_charge        double,
wr_account_credit         double,
```

```
wr_net_loss               double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/web_returns"
;
drop table if exists web_returns;
create table web_returns
using parquet
as (select * from web_returns_text)
;
drop table if exists web_returns_text;

drop table if exists web_sales_text;
create table web_sales_text
(
ws_sold_date_sk           int,
ws_sold_time_sk           int,
ws_ship_date_sk           int,
ws_item_sk                int,
ws_bill_customer_sk       int,
ws_bill_cdemo_sk          int,
ws_bill_hdemo_sk          int,
ws_bill_addr_sk           int,
ws_ship_customer_sk       int,
ws_ship_cdemo_sk          int,
ws_ship_hdemo_sk          int,
ws_ship_addr_sk           int,
ws_web_page_sk            int,
ws_web_site_sk            int,
ws_ship_mode_sk           int,
ws_warehouse_sk           int,
ws_promo_sk               int,
ws_order_number           int,
ws_quantity               int,
ws_wholesale_cost         double,
ws_list_price             double,
ws_sales_price            double,
ws_ext_discount_amt       double,
ws_ext_sales_price        double,
ws_ext_wholesale_cost     double,
ws_ext_list_price         double,
ws_ext_tax                double,
ws_coupon_amt             double,
ws_ext_ship_cost          double,
ws_net_paid               double,
ws_net_paid_inc_tax       double,
ws_net_paid_inc_ship      double,
ws_net_paid_inc_ship_tax  double,
ws_net_profit             double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/web_sales"
;
drop table if exists web_sales;
create table web_sales
using parquet
as (select * from web_sales_text)
;
drop table if exists web_sales_text;

drop table if exists web_site_text;
create table web_site_text
(
web_site_sk               int,
web_site_id               string,
web_rec_start_date        string,
web_rec_end_date          string,
web_name                  string,
```

```
web_open_date_sk          int,
web_close_date_sk         int,
web_class                 string,
web_manager               string,
web_mkt_id                int,
web_mkt_class             string,
web_mkt_desc              string,
web_market_manager        string,
web_company_id            int,
web_company_name          string,
web_street_number         string,
web_street_name           string,
web_street_type           string,
web_suite_number          string,
web_city                  string,
web_county                string,
web_state                 string,
web_zip                   string,
web_country               string,
web_gmt_offset            double,
web_tax_percentage        double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION "/user/ubuntu/db/4tb/data/web_site"
;
drop table if exists web_site;
create table web_site
using parquet
as (select * from web_site_text)
;
drop table if exists web_site_text;
```

workers

```
worker-1
worker-2
worker-3
worker-4
worker-5
worker-6
worker-7
worker-8
```

yarn-site.xml

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<configuration>
<property>
<name>mapreduceyarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>[Manager IP Address]</value>
</property>
<property>
<name>yarn.nodemanager.resource.memory-mb</name>
<value>131072</value>
</property>
<property>
<name>yarn.scheduler.maximum-allocation-mb</name>
<value>128450</value>
</property>
<property>
<name>yarn.nodemanager.resource.cpu-vcores</name>
<value>64</value>
</property>
<property>
<name>yarn.scheduler.maximum-allocation-vcores</name>
<value>64</value>
</property>
</configuration>
```

Scripts

run_query.sh.power

```
#!/bin/bash

date=$(date '+%Y%m%d_%H%M%S')
num_executors=${1}
exec_cores=${2}
exec_mem=${3}

results=/home/ubuntu/results
mkdir -p ${results}

#Copy yarn-site.xml file for the correct size instance
#cp ~/yarn-site-${instance_size}.xml $HADOOP_HOME/etc/hadoop/yarn-site.xml

#for i in {1..8};
#do
#     scp ~/yarn-site-${instance_size}.xml worker-${i}:$HADOOP_HOME/etc/hadoop/yarn-site.xml
#done
#
#sleep 5

#Start spark, hadoop, yarn on manager and worker nodes
~/scripts/start_all.sh

#Cleanup previous runs
pkill nmon
rm *.nmon

for i in {1..8};
do
        ssh worker-${i} 'pkill nmon'
      ssh worker-${i} 'rm /home/ubuntu/*.nmon'
done

#Make results directory
mkdir -p ${results}/${date}

#Start nmon on Manager node
nmon -f -t -s 5 -c 2000

#Start nmon on Worker nodes
for i in {1..8};
do
      ssh worker-${i} 'nmon -f -t -s 5 -c 2147483647'
done

sleep 10

#Run power test
echo "starting at ${date} with 64 vCPUs, ${num_executors} execs with ${exec_cores} cores and
${exec_mem} mem."
stream=0
nohup spark-sql --driver-memory 4g --driver-java-options -Dlog4j.configuration=file:///$PWD/work/
log4j.properties \
   --executor-memory ${exec_mem} --executor-cores ${exec_cores} --num-executors ${num_executors} \
   --conf spark.executor.extraJavaOptions=-Dlog4j.configuration=file:///${PWD}/work/log4j.properties \
   --conf spark.sql.crossJoin.enabled=true -database tpcds -f /home/ubuntu/db/queries/
stream${stream}.sql \
  > work/throughput_stream_${stream}.res 2>&1 &


echo "waiting"
wait
echo "done at ${date}"

#Stop nmon on Manager node
pkill nmon
```

```
#Stop nmon on Worker nodes
for i in {1..8};
do
        ssh worker-${i} 'pkill nmon'
done

#Copy nmon files and query results to results folder
mv ./*.nmon ${results}/${date}/manager.nmon

for stream in {0..0}; do
mv work/throughput_stream_${stream}.res ${results}/${date}
done

for i in {1..8};
do
      scp worker-${i}:/home/ubuntu/*.nmon ${results}/${date}/worker-${i}.nmon
      ssh worker-${i} 'rm /home/ubuntu/*.nmon'
done

mkdir ${results}/${date}/hadoop_config
mkdir ${results}/${date}/spark_config
cp $HADOOP_HOME/etc/hadoop/* ${results}/${date}/hadoop_config
cp $SPARK_HOME/conf/* ${results}/${date}/spark_config
cp /home/ubuntu/db/spark-tpc-ds-performance-test/run_query.sh ${results}/${date}/

touch ${results}/${date}/config.txt
config=${results}/${date}/config.txt
echo "Date: ${date}" >> ${config}
echo "Number of executors: $num_executors" >> ${config}
echo "Executor cores: $exec_cores" >> ${config}
echo "Executor memory: $exec_mem" >> ${config}

#Parse results and generate run summary
SUMMARY=${results}/${date}/run_summary.txt
echo "Run summary for tpc-ds run ${date}" >> ${SUMMARY}
echo "" >> ${SUMMARY}
cat ${config} | tee -a ${SUMMARY}
echo ""



for i in {0..0};
do
      touch ${results}/${date}/throughput_stream_${i}.txt
      cat ${results}/${date}/throughput_stream_${i}.res | grep "Time taken" | awk '{ print $3 }' >>
${results}/${date}/throughput_stream_${i}.txt
done

#Parse nmon results
for i in {1..8};
do
      ~/scripts/nmonchart ${results}/${date}/worker-${i}.nmon ${results}/${date}/worker-${i}.html
done

~/scripts/nmonchart ${results}/${date}/manager.nmon ${results}/${date}/manager.html

#sleep 5


#Stop hadoop, spark, yarn
~/scripts/stop_all.sh
sleep 5

#Shutdown workers and manager
~/scripts/shutdown_all.sh
```

run.sh

```
#!/bin/bash

read -p 'Number of executors: ' num_executors
read -p 'Executor cores: ' exec_cores
read -p 'Executor memory (k,m,g): ' exec_mem

./run_query.sh $num_executors $exec_cores $exec_mem &
```

View the original version of this report at https://facts.pt/XJvpK9D

Read the report ▶

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®