Behind the Report:
Accelerate Big Data and Database Workloads in Healthcare

This document provides the system-configuration details and step-by-step procedures that Prowess used to perform benchmark testing on two Dell Technologies™ platforms:

- Dell™ PowerEdge™ R730
- Dell™ PowerEdge™ R7515

For the full analysis, read the report "Accelerate Big Data and Database Workloads in Healthcare."

Testing was concluded on August 5, 2022.

Server Configurations

	Dell™ PowerEdge™ R730	Dell™ PowerEdge™ R7515
Hardware		
Processor	2 x Intel® Xeon® processor E5-2683	1 x AMD EPYC™ 7543 processor
Number of CPUs	2	1
Cores	16	32
Cores/threads total	32/64	32/64
Frequency (base/SCT/MCT)	2,100 MHz	2,800 MHz
Storage controller 01	No operating system (OS) boot-optimized storage controller (BOSS)	Dell™ BOSS-S1
Disk	Not applicable (N/A)	223.57 GB
Number of disks	N/A	2
Storage controller 02	N/A	Dell™ PowerEdge™ RAID Controller (PERC) H730P
Disk	1,787.88 GB	1,787.88 GB
Number of disks	4	4
Installed memory	128 GB	128 GB
Memory DIMM	Error correcting code (ECC) DDR4	Error correcting code (ECC) DDR4
Memory speed	2,133 MT/s	3,200 MT/s
Number of memory DIMMs	8	8
BIOS version	2.13.0	2.7.3
OS performance profile	Performance	Performance

Software	
OS	Red Hat® Enterprise Linux® 8.6
OS kernel	4.18.0-372.13.1.el8_6.x86_64
Database	Microsoft® SQL Server® 2019—15.0.4236.7
Benchmarking tools	
Database performance	HammerDB TPROC-H benchmark
Big data (artificial intelligence [AI]) performance	Spark-Bench KMeans workload

Testing Procedures

Baseline

Baseline using Sysbench:

- CPU Baseline:

```
sysbench cpu --cpu-max-prime=10000 run
```

- Storage Baseline:

```
sysbench fileio --threads=16 --file-total-size=5G --file-test-mode=rndrw prepare
sysbench fileio --threads=16 --file-total-size=5G --file-test-mode=rndrw run
sysbench fileio --file-test-mode=rndrw cleanup
```

HammerDB with TPC-H

With HammerDB, a TPC-H–like workload can be run to determine the server performance with Microsoft® SQL Server®.

1. Set up the server under test (SUT) with Red Hat® Enterprise Linux®.
    a. System 1 (Dell PowerEdge R7515):
        i. Configure RAID
            1. Enter Dell™ Lifecycle Controller.
            2. Select **Hardware Configuration**.
            3. Under **Storage Configuration Wizards**, click **RAID Configuration**.
            4. Select the controller, and then click **Next**.
            5. Select the RAID level, and then click **Next**.
            6. Select the physical drives, and then click **Next**.
            7. Select the virtual disk parameters, and then click **Next**.
            8. Select **Finish** to apply the RAID configuration.

ii. Install Red Hat Enterprise Linux 8.6:

1. PXE boot and select Red Hat Enterprise Linux 8.6.

iii. Change to the correct time zone.

iv. Change **Software Selection** to **Server** and add **Hardware Monitoring Utilities**.

v. Create a root password.

vi. Create an additional user and make them an admin.

vii. Manually set partitions on the RAID 1 volume created in Dell Lifecycle Controller:

1. **/home = 130.92 GB**
2. **/rhel-root = 75 GB**
3. **/boot/efi = 600 MiB**
4. **/boot = 1,024 MiB**
5. **/swap = 16 GB**

viii. Enable wired network connection.

b. System 2 (Dell PowerEdge R730):

i. Configure RAID:

1. Software RAID was used and configured in Red Hat Enterprise Linux.

ii. Install Red Hat Enterprise Linux.

1. PXE boot and select Red Hat Enterprise Linux 8.6.

iii. Change to the correct time zone.

iv. Change **Software Selection** to **Server** and add **Hardware Monitoring Utilities**.

v. Create a root password.

vi. Create an additional user and make them an admin.

vii. Manually set partitions; for this step, all drives (4) were selected:

1. **/home = 100.09 GB with RAID5**
2. **/mssql/data = 3.5 TB with RAID5**
3. **/mssql/logs = 1.5 TB with RAID10**
4. **/rhel-root = 1,024 MB with RAID1**
5. **/boot/efi = 600 MiB with RAID1**
6. **/boot = 70 MiB with RAID5**
7. **/swap = 16 GB with RAID5**

viii. Enable wired network connection.

2. Install and optimize Microsoft SQL Server.

    a. Run the following commands to install prerequisites:

```
sudo yum install python2 compat-openssl10
sudo alternatives --config python
```

    b. Run the following commands to enable TuneD and install the SQL Server TuneD profile:

```
systemctl enable tuned
dnf install tuned-profiles-mssql
```

    c. The tuned.conf file in /usr/lib/tuned/mssql has the following configuration:

```
#
# tuned configuration
#

[main]
summary=Optimize for Microsoft SQL Server
include=throughput-performance

[cpu]
force_latency=5

[vm]
# For multi-instance SQL deployments use 'madvise' instead of 'always'
transparent_hugepages=always

[sysctl]
vm.swappiness=1
vm.dirty_background_ratio=3
vm.dirty_ratio=80
vm.dirty_expire_centisecs=500
vm.dirty_writeback_centisecs=100
vm.max_map_count=1600000
net.core.rmem_default=262144
net.core.rmem_max=4194304
net.core.wmem_default=262144
net.core.wmem_max=1048576
kernel.numa_balancing=0

[scheduler]
sched_latency_ns=60000000
sched_migration_cost_ns=500000
sched_min_granularity_ns=15000000
sched_wakeup_granularity_ns=2000000
```

    d. Run the following command to set the SQL Server repository:

```
sudo curl -o /etc/yum.repos.d/mssql-server.repo
https://packages.microsoft.com/config/rhel/8/mssql-server-2019.repo
```

    e. Run the following command to install SQL Server:

```
sudo dnf install -y mssql-server
```

f. Run the following command to configure SQL Server:

```
/opt/mssql/bin/mssql-conf setup
```

g. Select **1** for evaluation.

h. Enter **Yes** to accept the license terms.

i. Enter a **SQL Server Admin Password**.

j. Run the following command to verify that SQL Server is running:

```
systemctl status mssql-server
```

k. For testing purposes only, disable SELINUX by modifying the config file in /etc/selinux/ to set SELINUX to "permissive:"

```
vi /etc/selinux/config
SELINUX=permissive
```

l. Install the SQL Server tools by using the following commands:

```
curl -o /etc/yum.repos.d/msprod.repo
https://packages.microsoft.com/config/rhel/8/prod.repo
dnf install -y mssql-tools unixODBC-devel
```

m. Configure .bash_profile and .bashrc to source the tools and SQL Server install paths by using the following commands:

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
echo 'export PATH="$PATH:/opt/mssql/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql/bin"' >> ~/.bashrc
source ~/.bashrc
```

n. Test connectivity to SQL Server by running the following command:

```
sqlcmd -s localhost -U SA -P <sa password>
```

o. Run the following SQL command to verify the version:

```
Select @@version
go
```

p. Enable trace flag 3979 to support SQL Server and the Forced Unit Access (FUA) input/output (I/O) subsystem by using the following commands:

i. Enter the following command to enable **traceflag 3979**:

```
mssql-conf traceflag 3979 on
```

ii. Enter the following command to set **control.writethrough** in the **mssql-conf configuration** option to **1**:

```
mssql-conf set control.writethrough 1
```

iii. Enter the following command to set **control.alternatewritethrough** in the **mssql-conf configuration** option to **0**:

```
mssql-conf set control.alternatewritethrough 0
```

q. Run the following commands to complete setup of SQL Server:

```
mssql-conf set telemetry.customerfeedback false
sysctl -w kernel.numa_balancing=0
sysctl -w vm.max_map_count=262144
mssql-conf set network.tlsprotocols 1.2
```

r. Set SQL Server memory to 90 percent of available memory by using the following command:

```
mssql-conf set memory.memorylimitmb 230400
```

s. Run the following command to create the SQL Server directory:

```
mkdir -p /mssql/data /mssql/log/log /mssql/log/tempdb
```

t. Run the following command to change ownership of the newly created directories:

```
sudo chown mssql:mssql /mssql/data
sudo chown mssql:mssql /mssql/log
```

u. Run the following command to enable execution on the directories:

```
sudo chmod 777 /mssql/data
sudo chmod 777 /mssql/log
```

v. Update the SQL Server configuration data and log file locations by using the following commands:

```
mssql-conf set filelocation.defaultdatadir /mssql/data/
mssql-conf set filelocation.defaultlogdir /mssql/log/log
```

w. Restart the SQL Server service by using the following command:

```
systemctl restart mssql-server.service
```

x. Launch the SQL Server management console from a client system.

y. Connect to the Linux SQL Server instance.

z. Run the following commands to modify the location of TempDB:

```
ALTER DATABASE tempdb MODIFY FILE
(NAME = tempdev, FILENAME = '/mssql/log/tempdb/tempdb01.mdf', SIZE = 1024,
FILEGROWTH = 8192MB)
GO
ALTER DATABASE tempdb MODIFY FILE
(NAME = templog, FILENAME = '/mssql/tempdb/templog.ldf', SIZE = 1024,
FILEGROWTH = 8192MB)
GO
ALTER DATABASE tempdb REMOVE FILE tempdev2
GO
ALTER DATABASE tempdb REMOVE FILE tempdev3
GO
ALTER DATABASE tempdb REMOVE FILE tempdev4
GO
ALTER DATABASE tempdb REMOVE FILE tempdev5
GO
```

```
ALTER DATABASE tempdb REMOVE FILE tempdev6
GO
ALTER DATABASE tempdb REMOVE FILE tempdev7
GO
ALTER DATABASE tempdb REMOVE FILE tempdev8
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev2, FILENAME = '/mssql/tempdb/tempdb02.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev3, FILENAME = '/mssql/tempdb/tempdb03.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev4, FILENAME = '/mssql/tempdb/tempdb04.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev5, FILENAME = '/mssql/tempdb/tempdb05.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev6, FILENAME = '/mssql/tempdb/tempdb06.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev7, FILENAME = '/mssql/tempdb/tempdb07.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev8, FILENAME = '/mssql/tempdb/tempdb08.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
```

aa. Enter the following command to set **Max Degree of Parallelism** to **0**:

```
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
EXEC sp_configure 'max degree of parallelism', 0;
GO
RECONFIGURE WITH OVERRIDE;
```

3. Download HammerDB to the SUT:

```
curl -OL https://github.com/TPC-Council/HammerDB/releases/download/v4.4/HammerDB-4.4-
Linux.tar.gz
```

4. Deploy the HammerDB TPROC-H test database.

```
tar -xvf HammerDB-4.4-Linux.tar.gz
```

5. Modify the “mssqls_pass” variable in the mssqlserver.xml file to use the correct password:

```
vi config/mssqlserver.xml
```

6. Deploy the HammerDB TPROC-H test database:
    a. Enter the following command to start HammerDBcli:

```
./hammerdbcli
```

    b. Enter the following command to set the database type:

```
dbset db mssqls
```

    c. Enter the following command to verify the database type:

```
print db
```

    d. Enter the following command to set the benchmark workload type to TPROC-H:

```
dbset bm TPROC-H
```

    e. Enter the following command to set Scale Factory to 300:

```
diset tpch mssqls_scale_fact 300
```

    f. Enter the following command to set the number of virtual users to build schema to 40:

```
diset tpch mssqls_num_tpch_threads 40
```

    g. Enter the following command to set **Maxdop** to **0**:

```
diset tpch mssqls_maxdop 0
```

    h. Enter the following command to set **Clustered Columnstore** to **true**:

```
diset tpch mssqls_colstore true
```

    i. Enter the following command to build the schema:

```
buildschema
```

    j. Enter the following command to verify the status of the build:

```
vustatus
```

    k. Enter the following command to destroy the build virtual users:

```
vudestroy
```

7. Run through experimentation to determine Scale Factor and Streams for HammerDB testing.
8. Start HammerDBcli by running the following command where HammerDB is installed:

```
Hammerdbcli
```

9. Configure HammerDB to use the following settings by running the commands below:
    a. Set the database test to use SQL Server:

```
dbset dbmssqls
```

b. Set the benchmark type to use TPROC-H:

```
dbset bm TPROC-H
```

c. Set the scale factor (1, 100, and 300):

```
diset tpch mssqls_scale_fact 1
```

d. Set the number of threads to 40:

```
diset tpch mssqls_num_tpch_threads 40
```

e. Set the **Maxdrop** to **0**:

```
diset tpch mssqls_maxdop 0
```

f. Set **Clustered Columnstore** to **true**:

```
diset tpch mssqls_colstore true
```

g. HammerDB is now configured and ready to run.

10. Configure the TCL script that HammerDB will run. Create a new file called mssqltest.tcl with the following configuration, and save it where you have HammerDB installed:

```
#!/bin/tclsh
proc runtimer { seconds } {
set x 0
set timerstop 0
while {!$timerstop} {
incr x
after 1000
if { ![ expr {$x % 60} ] } {
set y [ expr $x / 60 ]
puts "Timer: $y minutes elapsed"
}
update
if { [ vucomplete ] || $x eq $seconds } { set timerstop 1 }
}
return
}
puts "SETTING CONFIGURATION"
dbset db mssqls
dbset bm TPROC-H
diset tpch mssqls_scale_fact 1
vuset iterations 1
vuset showoutput 1
vuset logtotemp 1
vuset timestamps 1
vuset unique 1
loadscript
foreach z { 1 5 10 20 50  } {
puts "$z iteration"
vuset vu $z
vucreate
vurun
```

```
runtimer 1800
vudestroy
after 1920
}
puts "TESTING COMPLETE"
```

11. Download and install atop to capture system performance information by using the following commands:

```
wget https://www.atoptool.nl/download/atop-2.6.0-1.el8.x86_64.rpm
chmod +x atop-2.6.0-1.el8.x86_64.rpm
rpm -ivh atop-2.6.0-1.el8.x86_64.rpm
service atop start
```

12. Capture atop data by using the following command:

```
atop -r -b <beginning time> -e <ending time> > /tmp/pass#.txt
example: atop -r -b 12:06 -e 12:38 > /tmp/atop/pass3.txt
```

13. Capture DSTAT data by using the following command:

```
dstat -trdlD total,sdb,sdc 60 --output /tmp/pass#.csv
```

14. Run the benchmark:

    a. Open three separate command windows.

    b. Run the following command in the first command window:

```
atop
```

    c. Run the following command in the second command window (making sure to update output file name to match current pass):

```
dstat -trdlD total,sdb,sdc 60 --output /tmp/dstat_pass#.csv
```

d. Run the following commands in the third command window:

```
cd /Hammerdb
./hammerdbcli
source mssqltest.tcl
```

    e. When the test has completed, capture a screenshot of the atop window.

    f. Stop the DSTAT command by using **Ctrl+C** and make note of the start and stop times.

    g. Use the times from DSTAT and run the following command:

```
atop -r -b beginning time -e ending time > /tmp/pass#.txt
```

    h. Highlight and capture the output results of the Spark-Bench pass, and then save them to a .txt file.

    i. Repeat this process three more times, rebooting between each pass, and keeping the last three pass scores.

Spark-Bench

1. Create a RAID configuration:
    a. System 1 (Dell PowerEdge R7515):
        i. Configure RAID:
            1. Enter Dell Lifecycle Controller.
            2. Select **Hardware Configuration**.
            3. Under **Storage Configuration Wizards**, click **RAID Configuration**.
            4. Select the controller, and then click **Next**.
            5. Select the RAID level, and then click **Next**.
            6. Select the physical drives, and then click **Next**.
            7. Select the virtual disk parameters, and then click **Next**.
            8. Select **Finish** to apply the RAID configuration.
    b. System 2 (Dell PowerEdge R730):
        i. Software RAID was used and configured in Red Hat Enterprise Linux.
2. Install Red Hat Enterprise Linux.
    a. System 1 (Dell PowerEdge R7515):
        i. PXE boot and select Red Hat Enterprise Linux 8.6.
        ii. Change to the correct time zone.
        iii. Change **Software Selection** to **Server** and add **Hardware Monitoring Utilities**.
        iv. Create a root password.
        v. Create an additional user and make them an admin.
        vi. Manually set partitions on the RAID 1 volume created in Dell Lifecycle Controller:
            1. **/home = 130.92 GB**
            2. **/rhel-root = 75 GB**
            3. **/boot/efi = 600 MiB**
            4. **/boot = 1,024 MiB**
            5. **/swap = 16 GB**
        vii. Enable wired network connection.

b. System 2 (Dell PowerEdge R730):

i. PXE boot and select Red Hat Enterprise Linux 8.6:

ii. Change to the correct time zone.

iii. Change **Software Selection** to **Server** and add **Hardware Monitoring Utilities**.

iv. Create a root password.

v. Create an additional user and make them an admin.

vi. Manually set partitions; for this process, all drives (4) were selected.

1. **/home = 500.09 GB with RAID5**
2. **/spark = 3.8 TB with RAID5**
3. **/rhel-root = 2 GB with RAID1**
4. **/boot/efi = 600 MiB with RAID1**
5. **/boot = 70 MiB with RAID5**
6. **/swap = 16 GB with RAID5**

vii. Enable wired network connection.

3. Stop and disable the firewall.

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

4. Disable SELINUX.

```
vi /etc/selinux/config

SELINUX=permissive
```

5. Download and install Java JDK.

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

6. Set JAVA_HOME:

```
export JAVA_HOME=~/jres/java-8
```

7. Verify JAVA_HOME is set correctly:

```
printenv | grep JAVA_HOME
```

8. Download Spark:

```
wget https://archive.apache.org/dist/spark/spark-2.4.8/spark-2.4.8-bin-hadoop2.7.tgz
```

9. Unpack the tarball:

```
tar -xvf spark-2.4.8-bin-hadoop2.7.tgz
```

10. Create a systemd unit file for the master service:

```
vi /etc/systemd/system/spark-master.service

       [Unit]
       Description=Apache Spark Master
       After=netowrk.target

       [Service]
       Type=forking
       User=root
       Group=root
       ExecStart=/opt/spark/sbin/start-master.sh
       ExecStop=/opt/spark/sbin/stop-master.sh

       [Install]
       WantedBy=multi-user.target
```

11. Create a systemd unit file for the slave service:

```
vi /etc/systemd/system/spark-slave.service

       [Unit]
       Description=Apache Spark Slave
       After=netowrk.target

       [Service]
       Type=forking
       User=root
       Group=root
       ExecStart=/opt/spark/sbin/start-slave.sh spark://127.0.0.1:7077
       ExecStop=/opt/spark/sbin/stop-slave.sh

       [Install]

       WantedBy=multi-user.target
```

12. Ask systemd to read the new service files:

```
systemctl daemon-reload
```

13. Start the services:

```
systemctl start spark-master.service
systemctl start spark-slave.service
```

14. Verify the Spark services are running:

```
systemctl status spark-master.service
systemctl status spark-slave.service
```

15. Download and extract Spark-Bench:

```
wget https://github.com/CODAIT/spark-bench/releases/download/v99/spark-
bench_2.3.0_0.4.0-RELEASE_99.tgz
```

16. Unpack the tarball:

```
tar -xvzf spark-bench_2.3.0_0.4.0-RELEASE_99.tgz
```

17. Create the kmeansworkloadgenerate.conf file with the following command

```
vi kmeansworkloadgenerate.conf
```

18. Enter the following information into the file, and then save

```
spark-bench = {
	spark-home = "/opt/spark/"
	spark-submit-config = [{
		spark-args = {
		master = "spark://127.0.0.1:7077"
		}
		spark-bench-jar = "/spark-bench/spark-bench_2.3.0_0.4.0-
	RELEASE/lib/spark-bench-2.3.0_0.4.0-RELEASE.jar"workload-suites = [
	{
		descr = "KMean data generator"
	benchmark-output = "console"
	workloads = [
	{
		name = "data-generation-kmeans"
		rows = 100000
		cols = 99
		output = "/opt/spark/temp/kmeans-data.csv"
		k = 10
		scaling = 1.6
		partitions = 10
		}
	]
	}
]
}]
}
```

19. Set the environment for Spark-Bench by generating workload files for Spark-Bench with 100,000 rows, and then recreate the environment with 50,000,000 rows:

```
/bin/spark-bench.sh kmeansworkloadgenerate.conf
```

20. Create the workload script:

```
vi kmeansworkload.conf
	spark-bench =   {
		spark-home = "/spark/spark/"
		spark-submit-config = [
			{
			spark-args = {
				master = "spark://127.0.0.1:7077"
				num-executors = 31
					executor-memory = 32g
```

```
                                        }
                                        workload-suites = [
                                                {
                                                descr = "KMean data generator"
                                                benchmark-output = "console"
                                                workloads = [
                                                        {
                                                        name = "kmeans"
                                                        input = "/spark/spark/temp/kmeans-data.csv"
                                                        rows = 100000
                                                        cols = 24
                                                        scaling = 1.6
                                                        partitions = 10
                                                        output = /spark/benchmark/results.txt
                                                        k = 10
                                                        maxiterations = 5
                                                        }
                                                ]
                                        }
                                ]
                        }
                        ]
                        }
```

21. Modify "spark-bench-env.sh" by setting the following:

```
vi /spark-bench-env.sh
       export SPARK_HOME=<spark location>
       export SPARK_MASTER_HOST=<IP address of the Master>
```

22. Download and install atop to capture system performance information by using the following commands:

```
wget https://www.atoptool.nl/download/atop-2.6.0-1.el8.x86_64.rpm
chmod +x atop-2.6.0-1.el8.x86_64.rpm
rpm -ivh atop-2.6.0-1.el8.x86_64.rpm
service atop start
```

23. Capture atop data by using the following command:

```
atop -r -b <beginning time> -e <ending time> > /tmp/pass#.txt
example: atop -r -b 12:06 -e 12:38 > /tmp/atop/pass3.txt
```

24. Install DSTAT:

```
sudo yum -y install dstat
```

25. Capture DSTAT data by using the following command:

```
dstat -trdlD total,sdb,sdc 60 --output /tmp/pass#.csv
```

26. Run the workload three times, capturing atop and DSTAT results:

    a. Run the benchmark:

        i. Open three separate command windows.

ii. Run the following command in the first command window:

```
atop
```

iii. Run the following command in the second command window (making sure to update output file name to match current pass):

```
dstat -trdlD total,sdb,sdc 60 --output /tmp/dstat_pass#.csv
```

iv. Run the following command in the third command window:

```
/{spark-bench dir location}/bin/spark-bench.sh kmeansworkload.conf
```

v. When the test has completed, capture a screenshot of the atop window.

vi. Stop the DSTAT command by using **Ctrl+C** and make a note of the start and stop times.

vii. Use the times from DSTAT and run the following command:

```
atop -r -b beginning time -e ending time > /tmp/pass#.txt
```

viii. Highlight and capture the output results of the Spark-Bench pass, and then save them to a .txt file.

ix. Repeat this process three more times, rebooting between each pass, and keeping the last three pass scores.