DELL Technologies

Technical White Paper

Dell EMC Unity XT: Microsoft SQL Server 2019 Big Data Clusters

Abstract

This document includes architecture and deployment guidance for Microsoft® SQL Server® 2019 Big Data Clusters with Dell EMC™ Unity XT storage. It also includes a deployment example on the Red Hat® OpenShift Container Platform.

February 2021

Revisions

Date	Description
July 2020	Initial release: Dell EMC Unity XT
Feb 2021	Legal disclaimer update

Acknowledgments

Author: Doug Bernhardt

Support:

- Microsoft: Mihaela Blendea, Sinisa Knezevic, Jamie Reding
- Red Hat: Dave Cain, Abhinav Joshi

Table of contents

Executive summary

The Microsoft® SQL Server® 2019 release introduced the SQL Server 2019 Big Data Clusters feature. Big Data Clusters enable deploying scalable clusters of not only SQL Server, but also Apache® Spark™ and Hadoop® Distributed File System (HDFS), as containers running on Kubernetes. This feature has different requirements compared to traditional versions of SQL Server. This document provides recommendations, tips, and other guidelines for architecting and deploying SQL Server 2019 Big Data Clusters on Dell EMC™ Unity XT™ storage. For general best practices using Dell EMC Unity systems, see the Dell EMC: Unity Best Practices Guide.

These guidelines are intended to cover most use cases. We recommend these guidelines, but they are not strictly required.

This paper was developed using the Dell EMC Unity 880F all-flash array, but is also applicable when using the 350F, 450F, 550F, 380F, 480F, 680F, and 880F Dell EMC Unity all-flash arrays.

If you have questions about the applicability of these guidelines in your environment, contact your Dell Technologies representative to discuss the appropriateness of the recommendations.

Audience

This document is intended for Dell EMC Unity administrators, database administrators, architects, partners, and anyone responsible for configuring Dell EMC Unity storage systems. Some familiarity with Dell EMC unified storage systems is assumed.

We welcome your feedback along with any recommendations for improving this document. Send comments to StorageSolutionsFeedback@dell.com.

1 Introduction

This section provides an overview for Dell EMC Unity XT and SQL Server 2019 Big Data Clusters. Dell EMC Unity arrays are virtually provisioned, flash-optimized storage systems that are designed for ease of use. This paper covers the all-flash array models which are well suited for SQL Server 2019 Big Data Clusters.

1.1 Dell EMC Unity XT overview

Dell EMC Unity XT all-flash and hybrid-flash arrays set new standards for storage with compelling simplicity, all-inclusive software, blazing speed, optimized efficiency, and multicloud enablement. All these features are combined in a modern NVMe-ready solution that meets the needs of resource-constrained IT professionals in large or small companies. Designed for performance and efficiency, and built for hybrid-cloud environments, these systems are the perfect fit to support demanding virtualized applications, deploying unified storage, and addressing remote-office and branch-office requirements.

1.2 SQL Server 2019 Big Data Clusters overview

SQL Server 2019 introduced a groundbreaking data platform with SQL Server 2019 Big Data Clusters (BDC). This platform addresses big-data challenges in a unique way, and solves many of the traditional challenges with building big-data and data-lake environments. See an overview of SQL Server 2019 Big Data Clusters on the Microsoft page SQL Server 2019 Big Data Cluster Overview and on the GitHub page SQL Server Big Data Cluster Workshops.

In addition to the product documentation, the following subsections cover specific benefits when deploying BDC on Unity XT.

1.2.1 Data virtualization

Typically, in big-data and data-analytics environments, data must be prepared for analysis. Often, this preparation includes data extraction, transformation, and load (ETL) processes in a separate data store. These processes can be expensive and time consuming in terms of development, maintenance, and administration. SQL Server 2019 Big Data Clusters enable choice in how to analyze data and access data with expanded PolyBase capabilities. Big Data Clusters can be used as a data store, but they can also be used to analyze data where it resides. This data could reside in existing relational databases, Hadoop clusters, or unstructured storage. This BDC capability enables scaling compute and storage separately, horizontally, and dynamically.

1.2.2 Data lake

Besides enabling access to virtualized data, SQL Server Big Data Clusters also includes a scalable HDFS **storage pool** for storing big data within the cluster. When the big data is stored in the BDC storage pool, you can analyze and query the data and combine it with your relational data.

Also, Dell EMC PowerScale OneFS allows NAS storage to be presented to Big Data Clusters using HDFS tiering. This allows a NAS folder to be presented as a mount to the HDFS storage pool, and it appears as another HDFS folder to the user. HDFS tiering allows PowerScale and Isilon™ customers to use their existing data environment inside Big Data Clusters with zero data movement.

This paper explores this feature running Spark workloads on large datasets that are stored in the storage pool. This paper also describes running a series of Spark SQL queries against that data to assess scalability.

1.2.3 Scale-out data mart

For data that is queried repetitively, it can be beneficial to store a copy of that data locally to improve performance. It may also be necessary to have a storage area for data that has been transformed or aggregated. SQL Server Big Data Clusters include a scalable **data pool** which you can use for this purpose. SQL Server Big Data Clusters provide scale-out compute and storage to improve the performance of analyzing any data. Data from various sources can be ingested and distributed across data pool nodes as a cache for further analysis.

1.2.4 Artificial intelligence and machine learning

SQL Server Big Data Clusters enable artificial intelligence (AI) and machine learning (ML) tasks on the data that is stored in HDFS storage pools and the data pools. You can use Spark and integrated AI tools in SQL Server using R, Python, Scala, or Java.

2 Planning and sizing

SQL Server 2019 BDC deploys on the Kubernetes (K8s) platform. Several distributions for Kubernetes are supported, and various Linux® distributions run Kubernetes. While you can deploy SQL Server 2019 BDC either in the public cloud or on-premises, this paper focuses on the Unity XT on-premises deployments. Besides the design that is addressed in this paper, Dell Technologies also provides many Kubernetes hosting platforms and validated designs, depending on the required solution. Regardless of the deployment, cluster management and user experience are largely the same.

2.1 Choosing a Kubernetes distribution

For administrators and IT professionals transitioning from Microsoft SQL Server on Windows Server, the Kubernetes platform can make the transition to SQL Server Big Data Clusters a bit daunting. At the time of publication, there over 100 certified Kubernetes offerings from the Cloud Native Computing Foundation. Also, the Kubernetes platform is rapidly evolving, and updates are published on a quarterly basis. These factors can make finding, setting up, and running a solution extremely challenging.

In the context of this solution, the K8s distribution must be supported by both Microsoft SQL Server 2019 Big Data Clusters and Dell EMC Unity XT storage. Since customers may require a fully supported enterprise solution, support for Red Hat OpenShift Container Platform is a priority for both Dell Technologies and Microsoft.

As of SQL Server 2019 CU5, OpenShift is a fully supported platform for Big Data Clusters. To accelerate the deployment of BDC on OpenShift, Dell Technologies provides step-by-step instructions about setting up and deploying Red Hat OpenShift with Dell EMC Unity XT on the Dell Technologies OpenShift Platform page. Red Hat OpenShift 4.3 was used as the deployment platform for developing this paper. General recommendations apply to all Kubernetes platforms including OpenShift. This document covers the differences between deployment platforms where applicable.

Multiple other combinations of Kubernetes platforms and Linux versions are available, with new additions continuing to be released. Consult the Microsoft SQL Server 2019 Big Data Clusters website for K8s distributions supported. Also, the Linux versions supported by the Dell EMC Unity CSI Driver can be found in the CSI Driver for Dell EMC Unity Product Guide.

Regardless of the combination chosen, consult the list of supported platforms and operating systems before deployment.

2.2 Dell EMC Unity XT sizing

Dell EMC Unity XT is available in various models and configurations to accommodate Big Data Clusters of any size. SQL Server Big Data Clusters can be used for different types of activities ranging from traditional online transaction processing (OLTP) workloads to big data analytic workloads using PolyBase and Spark. Each of these workloads can have a drastically different I/O profile, so understanding the workload is important.

2.2.1 OLTP workloads

A traditional OLTP workload typically consists of many small I/O requests (8 KB to 32 KB). For maximum performance, the workload is sized according to the number of requests per second (IOPS) and the latency required. Users typically wait for these transactions to occur in real time, so to fulfill many concurrent requests quickly, the storage must provide many IOPS at low latency. Besides total storage capacity, a Dell EMC Unity

XT system that is sized for this workload is sized primarily for optimal IOPS and latency performance rather than bandwidth.

2.2.2 Analytic workloads

The other extreme workload scenario involves large analytic queries that are processed through SQL Server or Apache Spark. These queries can process massive amounts of data and are often submitted and run as background jobs where users are not interactively waiting for a result. In this scenario, the I/O sizes can be large (1 MB to 2 MB) and performance may or may not be a primary concern. When performing large I/O, bandwidth can quickly become a bottleneck. Unity XT systems that are sized for analytic workloads should be optimized for capacity and bandwidth performance.

2.2.3 Sizing and selection

To fully use the capabilities of Big Data Clusters, most workloads will likely be a combination of the workloads mentioned previously. A Dell Technologies representative can help you analyze the various scenarios and determine the workload mix and the priority. This analysis can help you choose the proper Unity XT model and size the configuration based on your workload criteria. In an ideal scenario, a similar workload is running in one or more environments. In these cases, you can use tools such as Live Optics to gather workload data and input it into Dell Technologies sizing calculators.

2.2.4 Scale

When planning a big-data environment, scaling can sometimes be an afterthought. When scalability is not planned for an environment that will inevitably grow, this scenario can create problems in the future. SQL Server 2019 Big Data Clusters have been designed with scalability in mind. The default installation creates a cluster of three nodes, enabling performance and scale from the start. Using proven components such as SQL Server, Spark, and Kubernetes provides massive compute and scale. To add power to the cluster, just add nodes to the cluster. Dell EMC Unity XT storage enables scaling up to 16 PB of storage to accommodate the largest Big Data Clusters environments.

3 Deployment

SQL Server 2019 Big Data Clusters is a powerful data-analytics platform. It can be used for a wide variety of big-data and data-analytics tasks including AI and ML. As organizations discover the various ways that BDC can be deployed and used, they can define the best compute and storage requirements for specific use cases. The performance and scale of Unity XT models provide many benefits that are outlined in the following subsections.

Building out a big-data environment typically requires defining a stack of products that provide the required capabilities. It also involves configuring multiple components such as Hadoop and Spark, and selecting and installing monitoring and analytical components. SQL Server 2019 BDC simplifies a complex deployment process. Using a containerized architecture on the Kubernetes platform can simplify deployment, since Kubernetes manages networking, resiliency, and load balancing. The SQL Server 2019 BDC installation tools enable deploying an entire BDC cluster on Kubernetes with a single command.

The following sections discuss the Big Data Cluster deployment process on Unity XT storage. However, the general process for deploying Big Data Clusters in an on-premises deployment is as follows:

1. Deploy a Kubernetes environment.
2. Configure the persistent storage using the Container Storage Interface (CSI) in Kubernetes.
3. Deploy SQL Server 2019 Big Data Clusters.

For general guidance with deploying SQL Server BDC on various platforms, see the Microsoft article How to deploy SQL Server Big Data Clusters on Kubernetes. The following subsections provide extra guidance for deploying BDC on Unity XT.

3.1 Deploying Kubernetes

Kubernetes supports most major distributions of Linux. The Linux distribution that is used depends on the persistent storage method that is chosen. After the Linux operating system is installed, some customizations are required before installing Kubernetes and configuring the cluster. These customizations, and instructions for installing and configuring Kubernetes, are detailed in the Microsoft article Configure Kubernetes on multiple machines for SQL Server big data cluster deployments.

After completing these deployment instructions and before deploying SQL Server 2019 BDC on Kubernetes, you must configure the persistent storage.

3.2 Configuring persistent storage

Storage in Kubernetes environments works differently than with applications running directly on an operating system such as Microsoft Windows Server® or Linux. In K8s, applications are deployed in pods. The storage that is used by a K8s pod is ephemeral, and it is deleted and re-created each time the pod is stopped and started. For storage to exist beyond the lifetime of a pod, persistent storage must be created and presented to the pod.

Pods can also move around in the cluster. The K8s scheduler is responsible for finding a suitable node for the pod to run on. The scheduler accounts for node failures, resource constraints, and other rules that are applied to control which nodes are available for a pod to run on. When the pod moves around in the cluster, its persistent storage needs to follow it.

Kubernetes allows for volume provisioning using the Container Storage Interface (CSI). This interface allows storage vendors such as Dell Technologies to implement plug-ins or drivers to implement provisioning functionality in K8s

The Dell EMC Unity CSI driver implements CSI functionality for Dell EMC Unity XT storage. The CSI driver interprets the generic K8s storage commands that are implemented with the CSI, and translates the commands into the appropriate Dell EMC Unity XT operations. The first step to configuring persistent storage is to install and configure the Dell EMC Unity CSI driver.

The OpenShift platform uses operators for deploying applications. When deploying on OpenShift, the Dell CSI Operator deploys the Dell EMC Unity CSI driver. You can find the CSI driver and complete installation instructions on github.com/dell/dell-csi-operator.

You can directly deploy the Dell EMC Unity CSI driver on vanilla Kubernetes using the Dell EMC Unity CSI driver. The CSI driver and complete instructions are on https://github.com/dell/csi-unity.

Deploying the Unity CSI driver creates a StorageClass within the Kubernetes cluster. This StorageClass is used for dynamic-storage volume provisioning during the deployment of SQL Server BDC.

Note: Complete the testing steps in the CSI driver installation. If the CSI driver is not installed properly, the BDC installation will become unresponsive or fail.

For complete instructions for deploying SQL Server 2019 BDC, see the Microsoft article How to deploy SQL Server Big Data Clusters on Kubernetes > Deployment overview section.

For more information about data persistence in Kubernetes in the context of SQL Server 2019 BDC, see the Microsoft article Data persistence with SQL Server big data cluster in Kubernetes.

3.3 Deploying SQL Server 2019 Big Data Clusters

Once the Dell EMC Unity XT CSI driver is installed and configured properly, you can deploy a SQL Server Big Data Cluster. The BDC installation experience is largely the same regardless of the K8s distribution it is being deployed on. During BDC installation, the StorageClass created by the Dell EMC Unity XT CSI driver is specified either as an input parameter or in a configuration file, depending on the BDC installation method that is chosen. Complete instructions for deploying Big Data Clusters are in the Microsoft article How to deploy SQL Server Big Data Clusters on Kubernetes.

4 Big Data Clusters workload example on Dell EMC Unity XT

Big Data Clusters contain many tools and features for working with big data environments. For a complete overview of all the available components, see the Microsoft article What are SQL Server Big Data Clusters. One new area with Big Data Clusters is the storage pool which allows you to run Spark workloads on data that is stored in HDFS within the cluster.

As part of the SQL Server 2019 CU5 release which introduced support for OpenShift, Dell Technologies partnered with Microsoft and Red Hat to test the scalability of running Spark workloads on Big Data Clusters running on OpenShift.

4.1 Cluster configuration settings

4.1.1 Hardware configuration

For this testing, twelve Dell EMC PowerEdge™ R640 servers were used to configure a Red Hat OpenShift 4.3 cluster. One server was used for cluster-management tasks, three servers were used as primary nodes in the cluster, and the remaining eight servers were used as worker nodes. Each PowerEdge R640 server was configured with dual Intel® Xeon® Gold 6154 processors and 576 GB of memory.

A Dell EMC Unity XT 880F system was used for storage with 50 drives configured as a single storage pool.

For complete instructions to configure Dell EMC servers, storage, and networking for an OpenShift cluster deployment, see the Dell EMC OpenShift deployment guide.

4.1.2 Expanding container storage

When running big-data workloads inside containers, besides sizing the persistent storage, it is likely that you must expand the container storage also. When migrating data into Big Data Clusters or running workloads such as Spark, a considerable amount of space can be required for temporary operations. Kubernetes environments monitor resources and fail pods that exceed resource limits. If disk space utilization exceeds 85%, the pod receives a **NodeHasDiskPressure** alert, and the pod (and related workload) restarts or possibly fails.

In most default installations, a relatively small amount of storage is allocated to the root partition. For our cluster, a second 2 TB volume was created in addition to the boot volume, and the boot partition was extended onto this volume. This configuration allowed for ample container-storage working space. With OpenShift 4.3, container storage space is in **/var/lib/containers.** For other K8s distributions, the location may differ.

Since Dell EMC Unity XT allows thin provisioning of storage, a generous amount of space can be allocated for these operations, and it is only consumed if needed.

4.1.3 Maximum threads per container

When allocating more than 24 virtual CPUs to a container, you must increase the number of threads for a container beyond the default of 1024. Reaching this limitation with our workload resulted in out-of-memory errors returned from Spark jobs, but this event could also result in other unusual behavior. For instructions to change this value, see the article Change pids_limit in OpenShift. For our workload, we increased the **pids_limit** to 4096.

4.1.4 BDC deployment settings

Instructions for deploying Big Data Clusters on OpenShift describe how to create base configuration files for deployment. For this solution, **openshift-dev-test** was used as the source for the configuration template.

The BDC configuration can be modified from the default settings to use cluster resources and to address the workload requirements. For complete instructions about modifying these settings from the defaults, see the Microsoft BDC website > Customize deployments section. To scale out the BDC resource pools, the number of replicas were adjusted to fully leverage the resources of the cluster. The values in Table 1 were adjusted in the **bdc.json** file.

Table 1 BDC replica settings changed from the default values

Resource	Replicas
nmnode-0	2
Sparkhead	2
Zookeeper	3
compute-0	2
data-0	4
storage-0	8

Note: Although data and compute replicas were adjusted, they were not used for the Spark workload.

4.1.5 Spark and YARN settings

The configuration values for running Spark and Apache Hadoop YARN were also adjusted to the compute resources available per node. In this configuration, sizing was based on 576 GB of RAM and 72 virtual CPU cores available per PowerEdge R640 server. Much of this configuration is estimated and adjusted based on the workload. In this scenario, it was assumed that the worker nodes were dedicated to running Spark workloads. If the worker nodes are performing other operations or other workloads, you may need to adjust these values. You can also override Spark values as job parameters.

For further guidance about configuring settings for Apache Spark and Apache Hadoop in Big Data Clusters, see BDC documentation section Configure Apache Spark & Apache Hadoop.

Table 2 Spark and YARN settings and values

Setting	Value
spark-defaults-conf.spark.executor.memoryOverhead	512
yarn-site.yarn.nodemanager.resource.memory-mb	512000
yarn-site.yarn.nodemanager.resource.cpu-vcores	64
yarn-site.yarn.scheduler.maximum-allocation-mb	512000

Setting	Value
yarn-site.yarn.scheduler.maximum-allocation-vcores	8
yarn-site.yarn.scheduler.capacity.maximum-am-resource-percent	0.10

Note: The value for spark-defaults-conf.spark.executor.memoryOverhead was left at the 512 MB default, and it was overridden for all tests. Typically, you should set the default to 10% of the system memory. For more information, see the article Spark Configuration.

4.1.6 Storage pod scheduling

The Spark and YARN configuration settings that are used for the cluster assume that a node does not run more than one storage pod. If this node configuration is present, resource contention could occur. While the OpenShift scheduler attempts to balance this utilization, it is not guaranteed. For example, during a sustained node failure, the scheduler moves the pod to another node, and it would need to be manually rebalanced. Kubernetes affinity and anti-affinity rules may be useful in this situation. BDC also enables pod allocation to specific nodes using node labels. In this configuration, the goal is for each worker node to run a single storage pod.

4.1.7 HDFS replication

Dell EMC Unity XT storage has several advanced features that offer excellent levels of data protection. For data protection and to preserve space, you may reduce the HDFS replication factor (**hdfs-site.dfs.replication)** from the default value of 3. While Dell EMC Unity XT storage can perform data protection, this is only one purpose for this setting. Another purpose of the data replication is to make copies of the data that is available to Spark for processing. When reducing this setting from the default value of 3 to 1, there was a 25% average decrease in performance on this workload. Consider whether the priority should be storage savings or performance.

4.1.8 Persistent storage

You can configure persistent storage for BDC in the **control.json** file that is in the configuration template that was described previously. The storage class used was created during the Dell EMC Unity XT CSI driver installation, and you can display the class by using the **oc get sc** command. For the purposes of our testing, the class is named **test-unity**. Data and log volumes were sized at **8Ti** and **1Ti**, respectively. Dell EMC Unity XT thin provisioning allows you to provision ample space, while only consuming space as required, and simplify storage configuration. You can also customize each pool in the **bdc.json** file.

4.2 Dell EMC Unity XT considerations

4.2.1 Host mapping

The Dell EMC Unity CSI Driver performs host-mapping activities. When the CSI driver is installed, it creates host entries if they do not exist. Existing host entries must match the naming format that is used by the Dell EMC Unity CSI driver. Otherwise, the driver installation will fail. See the Unity CSI Driver product guide for details about how to verify or modify these parameters.

4.2.2 Volume creation

The deployment of Big Data Clusters dynamically creates volumes on Dell EMC Unity XT storage that are based on the installation parameters that are provided to the deployment. At a minimum, 26 volumes are created. Using the number of pod replicas specified previously, 54 volumes were dynamically created for this deployment.

As shown in Figure 1, you can filter the volume list in Unisphere by **pvc** to display all dynamically created volumes.

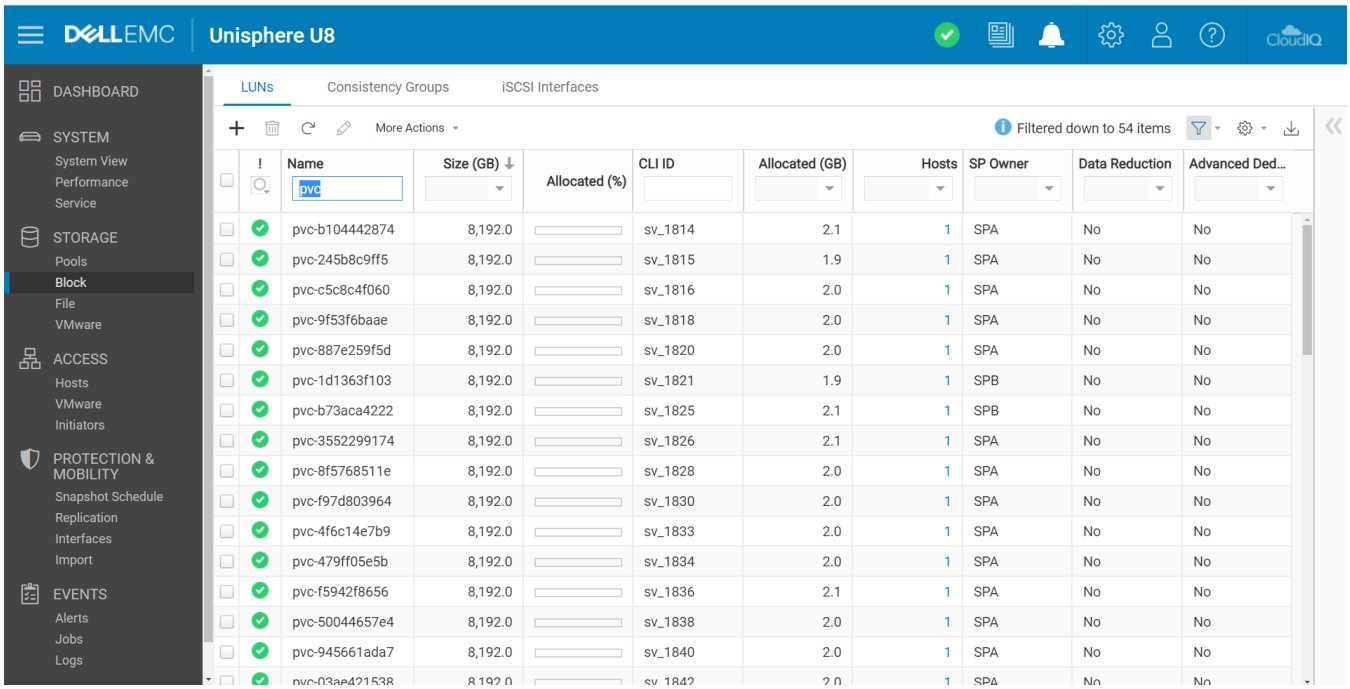

Figure 1 List volumes dynamically created by Dell EMC Unity XT CSI driver

The OpenShift storage class that is created upon CSI driver installation controls the file system that is used as and some volume-provision parameters. For the testing performed in this paper, the xfs file system was used because it is the default file system for Red Hat Enterprise Linux.

4.2.3 Volume ownership

Dell EMC Unity XT volumes are owned by a specific storage processor (SP). For best performance, the I/O workload should be balanced across both SPs. During volume provisioning, the Dell EMC Unity XT system attempts to balance the I/O workload by using a round-robin approach to assigning volumes to SPs when they are created. In a traditional setting, this works well. When deploying Big Data Clusters, several data

services and applications are deployed simultaneously. The Dell EMC Unity XT system is not able to balance the volumes, and you must perform this balancing manually.

In this deployment, the storage pool uses eight replicas, and each replica has a data volume and a log volume. To evenly distribute data volumes, identify the data volumes for the storage pool. For example, use the command **oc get pvc -n mssql-cluster | grep data-storage** to list all volumes that belong to the mssql-cluster namespace. This namespace is the default one that BDC deploys into. Then, filter the results by **data-storage.** The output is shown as follows.

PVC List for the storage pool data volumes:

```
data-storage-0-0   Bound    pvc-c3978f4521   8Ti        RWO            test-
unity   124m
data-storage-0-1   Bound    pvc-aaf0bb200f   8Ti        RWO            test-
unity   124m
data-storage-0-2   Bound    pvc-caab559697   8Ti        RWO            test-
unity   124m
data-storage-0-3   Bound    pvc-fe43a6c312   8Ti        RWO            test-
unity   124m
data-storage-0-4   Bound    pvc-d1275c5375   8Ti        RWO            test-
unity   124m
data-storage-0-5   Bound    pvc-b33191712f   8Ti        RWO            test-
unity   124m
data-storage-0-6   Bound    pvc-1755fd71ab   8Ti        RWO            test-
unity   124m
data-storage-0-7   Bound    pvc-fd81d2f582   8Ti        RWO            test-
unity   124m
```

When the list of relevant volumes is obtained, you can verify the volume ownership for Dell EMC Unity XT storage in Unisphere. From the storage menu on the left, click **Block** to display all available LUNs.

For each volume in the list, ensure that they are evenly divided between storage processor A (SPA) and storage processor B (SPB).

In the following example, SBP owns the volume (see Figure 2).

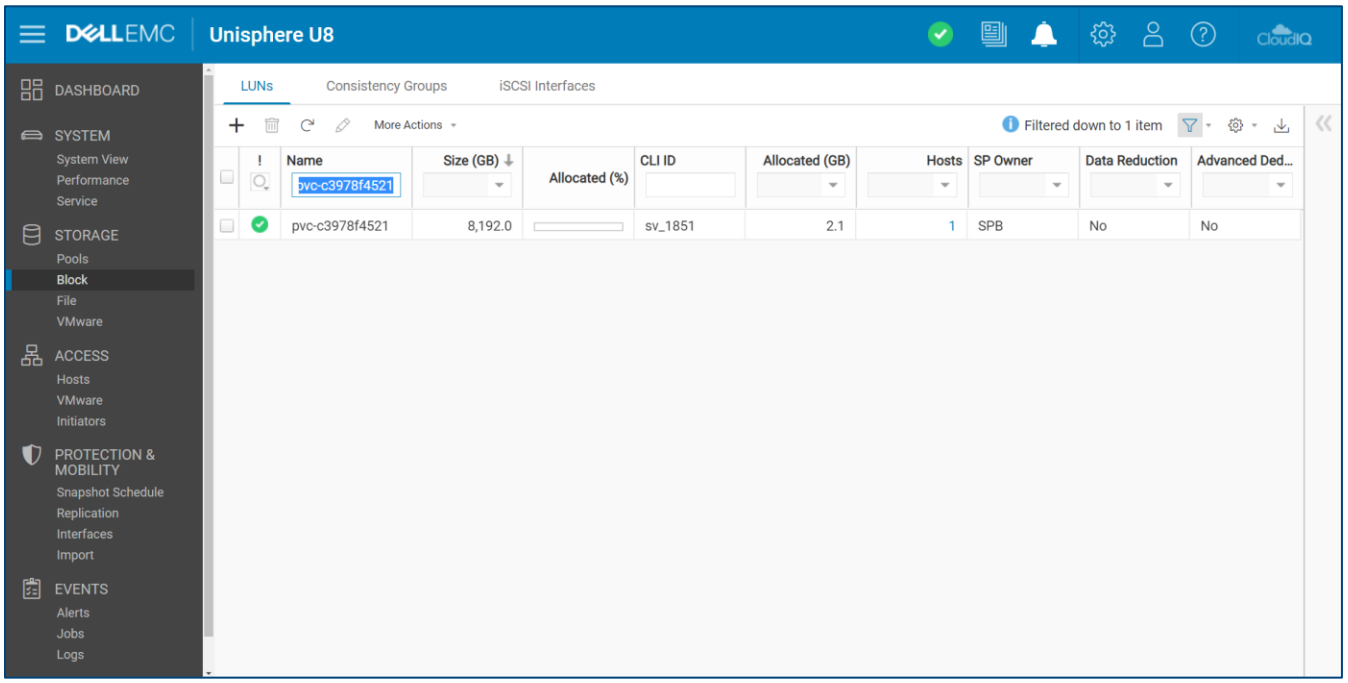

Figure 2 Searching for a specific volume

If ownership must be changed, edit the volume properties and change the **SP Owner** at the bottom of the tab (see Figure 3).

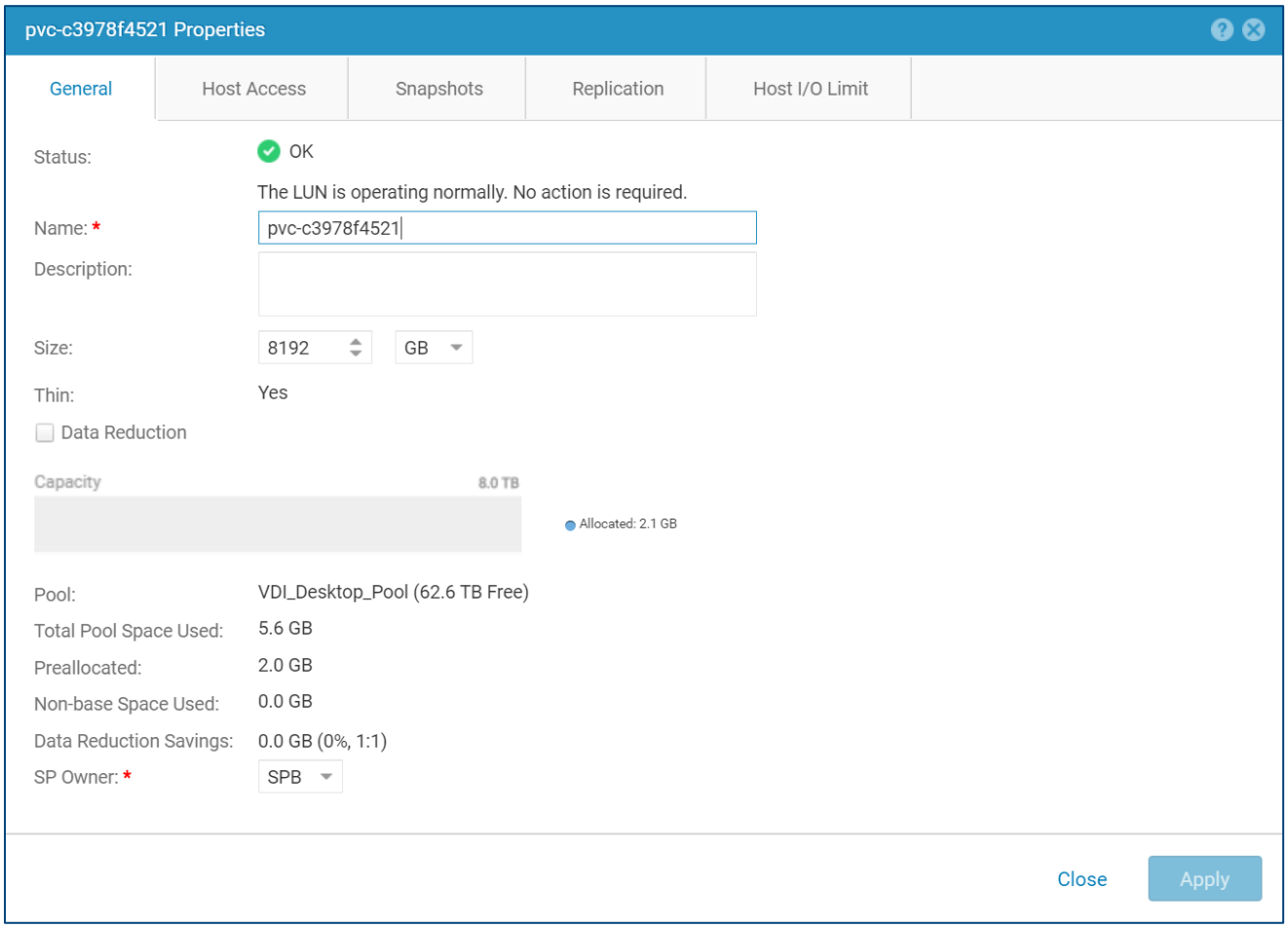

Figure 3 Modifying volume ownership

You can change the volume ownership at any time. Monitor workload performance in the Unisphere Performance Dashboard, and balance volume ownership until LUN bandwidth is relatively equal.

4.3 Big Data Clusters workload testing

To validate the configuration of the Big Data Cluster that is running on OpenShift and to test its scalability, we ran a TPC-DS workload on the cluster using the Databricks TPC-DS Spark SQL kit. This toolkit allows an entire TPC-DS benchmark to be submitted as a Spark job that generates the test dataset and runs a series of analytics queries across it. This workload was run for the 10 TB, 20 TB, and 30 TB dataset sizes. Since this workload runs entirely inside the storage pool of the SQL Server Big Data Cluster, the environment was scaled to run the recommended maximum of eight storage pods. We assigned one storage pod to each worker node in our OpenShift environment as shown in Table 3.

Table 3 BDC pool layout among worker nodes

<table>
<tr><th>Worker 1</th><th>Worker 2</th><th>Worker 3</th><th>Worker 4</th><th>Worker 5</th><th>Worker 6</th><th>Worker 7</th><th>Worker 8</th></tr>
<tr><td></td><td></td><td></td><td></td><td>Compute pool</td><td></td><td></td><td>Compute pool</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="4">Data pool</td></tr>
<tr><td></td><td></td><td></td><td></td><td>SQL primary instance</td><td></td><td></td><td></td></tr>
<tr><td colspan="8">Storage pool</td></tr>
<tr><td colspan="8">Dell EMC UnityCSI</td></tr>
</table>

Depending on the use of the various components and features within Big Data Clusters and the entire K8s cluster, the pools may be configured differently.

4.3.1 Workload balancing

SQL Server 2019 Big Data Clusters and Kubernetes are efficient at running intensive workloads such as Spark. In this environment, we customized Spark jobs to fully utilize the compute, memory, and storage resources available. In this OpenShift environment, our goal was to run our workloads up to ~80% CPU and memory, which we were easily able to do. See Figure 4 and Figure 5.

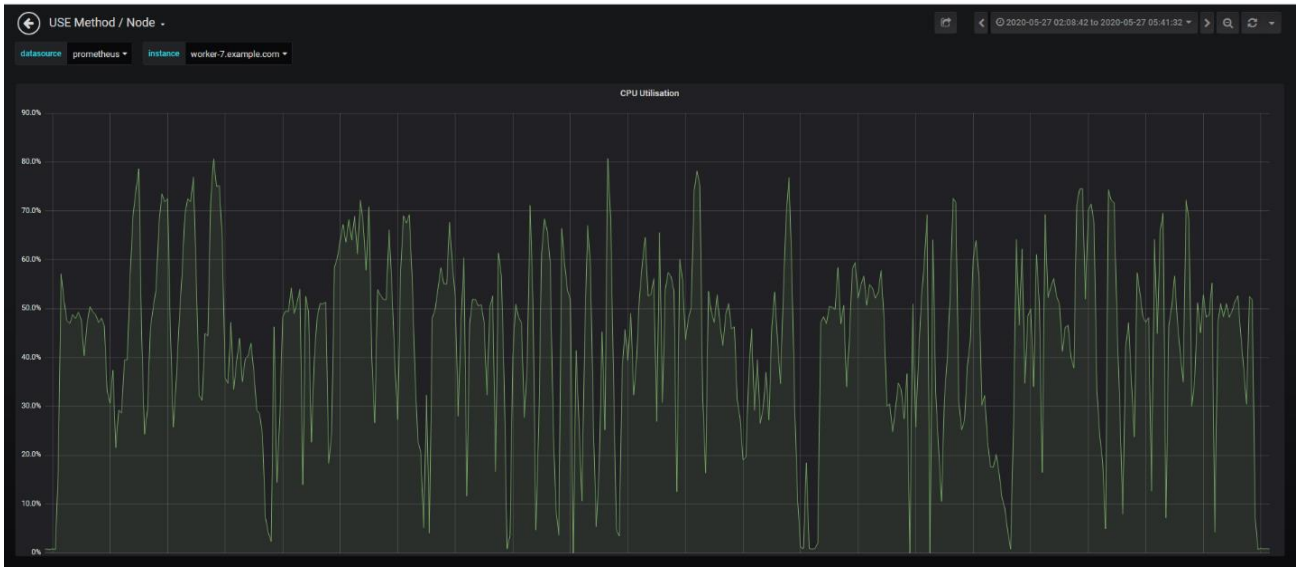

Figure 4 Worker node CPU utilization in the OpenShift Grafana dashboard

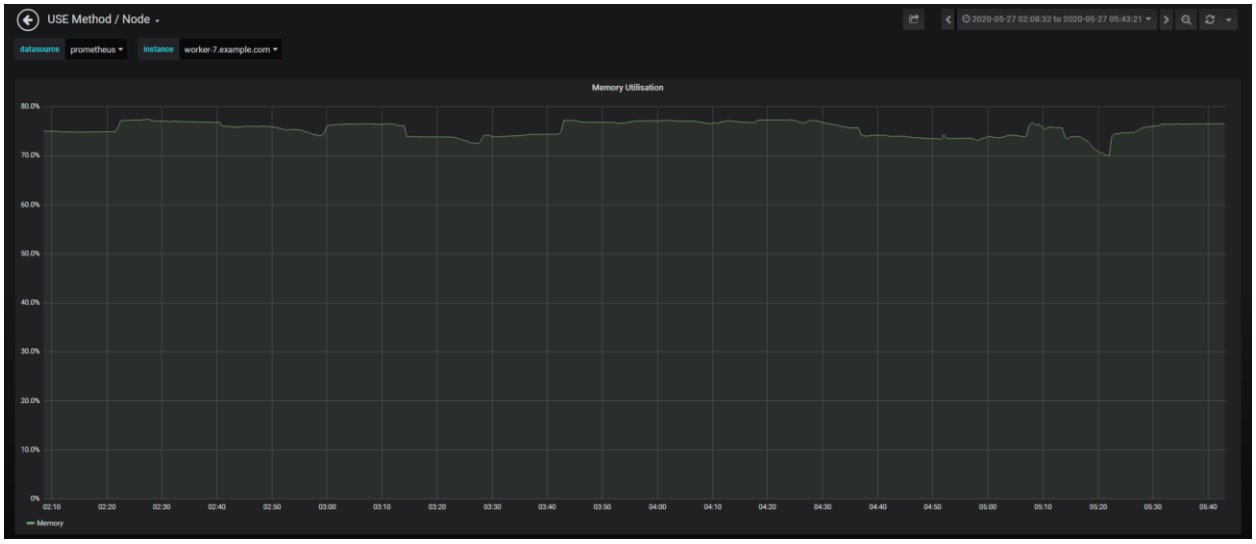

Figure 5 Worker node memory utilization in the OpenShift Grafana dashboard

The workload statistics were also balanced across the cluster. Figure 6 shows the CPU utilization of each node in the cluster. The utilization of each worker is consistent from node to node.

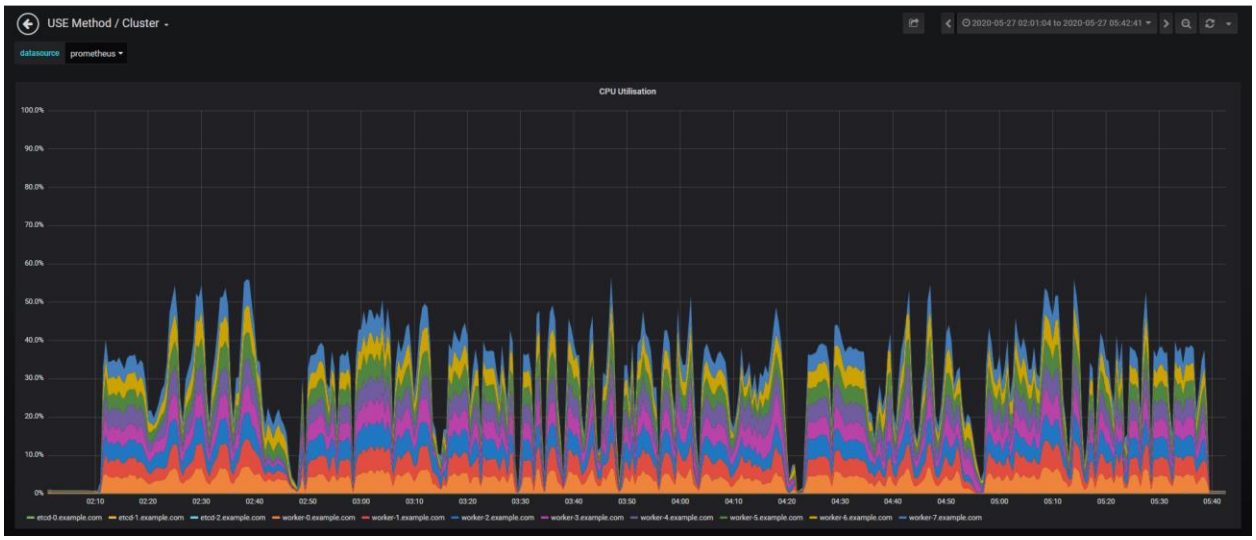

Figure 6 Balanced workload across worker nodes in the cluster in the OpenShift Grafana dashboard

4.3.2 I/O profile

The I/O profile of Spark workloads (see Figure 7) is different than the typical I/O profile of SQL Server environments. Usually with SQL Server OLTP workloads, 8 KB I/O is common, and large I/O is 128 KB or 256 KB. Over the course of testing our workload running with Spark, I/O sizes of 1.7 Mb were produced, with 0.5 Mb being the average. As discussed in section 2.2.3, storage sizing for high bandwidth is now an important consideration.

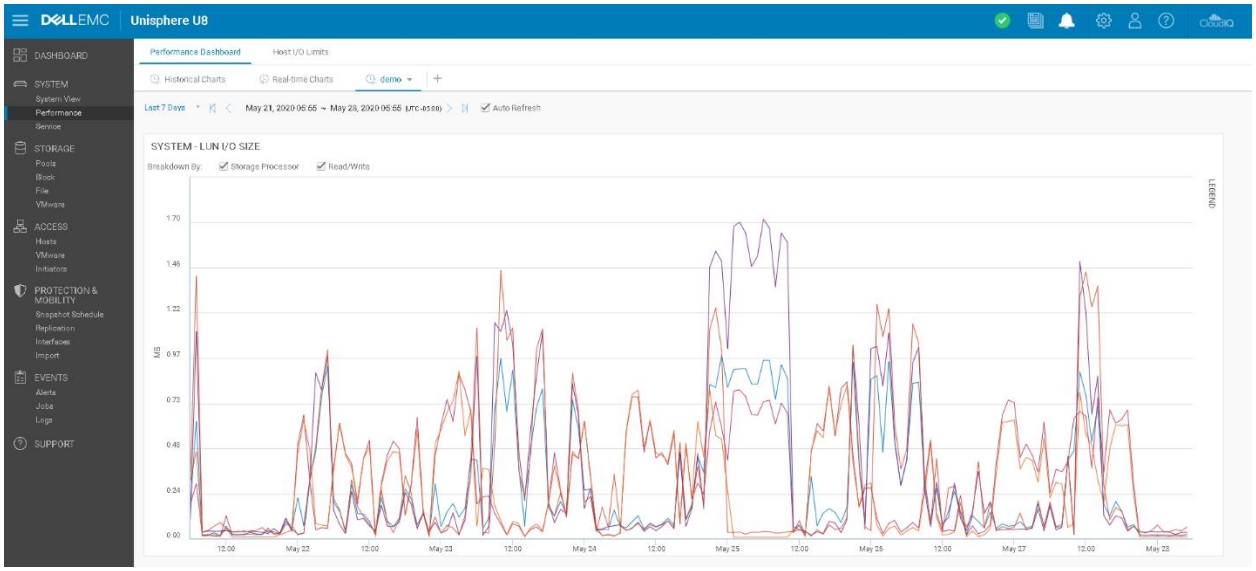

Figure 7 Spark workload LUN I/O sizes

4.3.3 Workload tests

Using the Databricks TPC-DS Spark SQL kit, the workload was run as Spark jobs for the 10 TB, 20 TB, and 30 TB workloads. For each workload, only the size of the dataset was changed. Several parameters can be specified on the job for tuning the Spark environment. The parameters used for each job are specified in Table 4.

Table 4 Spark job parameters

Parameter	Value
driverMemory	8G
driverCores	1
executorCores	8
executorMemory	66G
numExecutors	64
spark.driver.memoryOverhead	6g
spark.driver.maxResultSize	16g
spark.sql.broadcastTimeout	4800
spark.network.timeout	900s

Parameter	Value
spark.executor.memoryOverhead	6g
spark.sql.statistics.histogram.enabled	True
spark.sql.cbo.enabled	True
spark.sql.cbo.joinReorder.enabled	True
spark.sql.cbo.joinReorder.dp.star.filter	False
spark.sql.cbo.starSchemaDetection	True
spark.sql.optimizer.nestedSchemaPruning.enabled	True
spark.sql.cbo.joinReorder.dp.threshold	18
query	q1,q2,q3,q4,q5,q6,q7,q8,q9,q10, q11,q12,q13,q15,q16,q17,q18,q19,q20, q21,q22,q23a,q23b,q24a,q24b,q25,q26,q27,q28,q29,q30, q31,q32,q33,q34,q35,q36,q37,q38,q39a,q39b,q40, q41,q42,q43,q44,q45,q46,q47,q48,q49,q50, q51,q52,q53,q54,q55,q56,q57,q58,q59,q60, q61,q62,q63,q64,q65,q66,q67,q68,q69,q70, q71,q73,q74,q75,q76,q77,q78,q79,q80 ,q81,q82,q83,q84,q85,q86,q87,q88,q89,q90, q91,q92,q93,q94,q95,q96,q97,q98,q99 (queries 1-99 except for q14a, q14b, q72)

See the Microsoft article Configure Apache Spark and Apache Hadoop in Big Data Clusters for more information about tuning parameters and values.

4.3.4 Workload scalability

Using the Databricks Spark SQL kit, we demonstrated the scalability of Spark workloads in SQL Server Big Data Clusters. We ran the workload at dataset levels of 10 TB, 20 TB, and 30 TB. Figure 8 shows the sum of the runtime for 97 queries that were run on the various dataset sizes.

As shown in the figure, as we increased the size of the dataset, the workload scaled well across the cluster. This outstanding linear scale is a testament to all the solution components from Dell Technologies, Microsoft, and Red Hat.

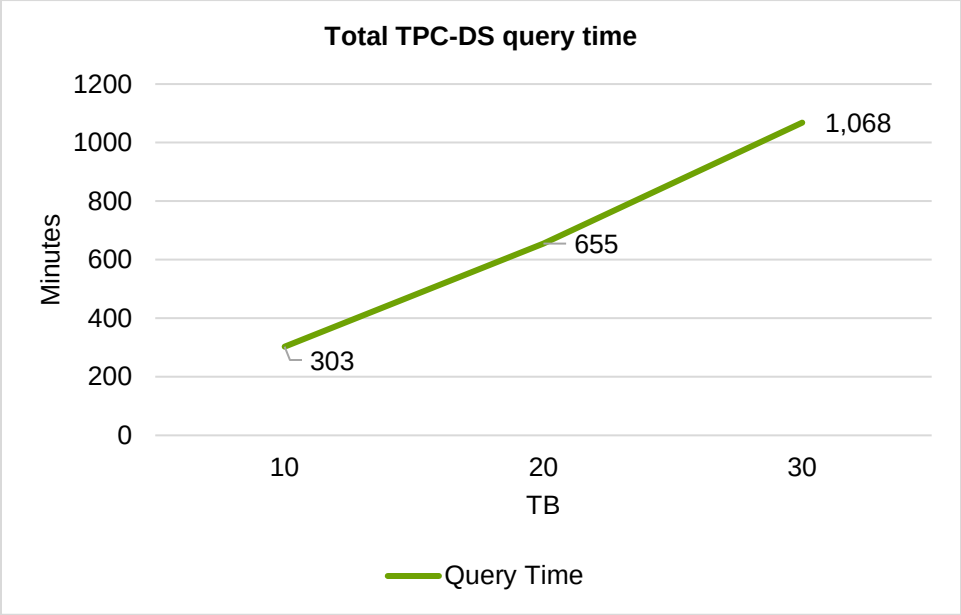

Figure 8 Workload scalability

5 Summary

The Dell EMC Unity XT storage platform provides outstanding scalable performance for Microsoft SQL Server 2019 Big Data Clusters. This paper demonstrates that Big Data Clusters running on the Red Hat Container Platform powered by PowerEdge servers and Dell EMC Unity XT storage provide a powerful, scalable big-data solution. Also, Dell Technologies, Microsoft, and Red Hat fully support this solution.

For more information about running SQL Server Big Data Clusters on Dell EMC Unity XT, contact your Dell Technologies representative.

A Configuration files

A.1 Bdc.json

```
{
    "apiVersion": "v1",
    "metadata": {
        "kind": "BigDataCluster",
        "name": "mssql-cluster"
    },
    "spec": {
        "resources": {
            "nmnode-0": {
                "spec": {
                    "replicas": 2
                }
            },
            "sparkhead": {
                "spec": {
                    "replicas": 2
                }
            },
            "zookeeper": {
                "spec": {
                    "replicas": 3
                }
            },
            "gateway": {
                "spec": {
                    "replicas": 1,
                    "endpoints": [
                        {
                            "name": "Knox",
                            "serviceType": "NodePort",
                            "port": 30443
                        }
                    ]
                }
            },
            "appproxy": {
                "spec": {
                    "replicas": 1,
                    "endpoints": [
                        {
                            "name": "AppServiceProxy",
                            "serviceType": "NodePort",
                            "port": 30778
                        }
                    ]
                }
            },
            "master": {
                "metadata": {
                    "kind": "Pool",
                    "name": "default"
```

```
            },
            "spec": {
                "type": "Master",
                "replicas": 1,
                "endpoints": [
                    {
                        "name": "Master",
                        "serviceType": "NodePort",
                        "port": 31433
                    }
                ],
                "settings": {
                    "sql": {
                        "hadr.enabled": "false"
                    }
                }
            }
        },
        "compute-0": {
            "metadata": {
                "kind": "Pool",
                "name": "default"
            },
            "spec": {
                "type": "Compute",
                "replicas": 2
            }
        },
        "data-0": {
            "metadata": {
                "kind": "Pool",
                "name": "default"
            },
            "spec": {
                "type": "Data",
                "replicas": 4
            }
        },
        "storage-0": {
            "metadata": {
                "kind": "Pool",
                "name": "default"
            },
            "spec": {
                "type": "Storage",
                "replicas": 8,
                "settings": {
                                        "spark": {
                    "includeSpark": "true"
                                            }
                },
                            "affinity": {
                                    "podAntiAffinity": {
  "requiredDuringSchedulingIgnoredDuringExecution": [
                                            {
                                                    "labelSelector": {
                                                            "matchExpressions": [
```

```
                                                                                {
                                                                                    "key":
"app",
        "operator": "In",
        "values": [
        "storage-0"
                                                                                ]
                                                                            }
                                                                        ]
                                                                    },
                                                                    "topologyKey":
"kubernetes.io/hostname"
                                                                }
                                                            ]
                                                        }
                                                    }


                }
            }
        },
        "services": {
            "sql": {
                "resources": [
                    "master",
                    "compute-0",
                    "data-0",
                    "storage-0"
                ]
            },
            "hdfs": {
                "resources": [
                    "nmnode-0",
                    "zookeeper",
                    "storage-0",
                    "sparkhead"
                ],
                "settings": {
                    "hdfs-site.dfs.replication": "3"
                }
            },
            "spark": {
                "resources": [
                    "sparkhead",
                    "storage-0"
                ],
                "settings": {
                    "spark-defaults-conf.spark.driver.cores": "1",
                    "spark-defaults-conf.spark.driver.memory": "1664m",
                    "spark-defaults-conf.spark.driver.memoryOverhead": "384",
                    "spark-defaults-conf.spark.executor.instances": "1",
                    "spark-defaults-conf.spark.executor.cores": "2",
                    "spark-defaults-conf.spark.executor.memory": "4500m",
                    "spark-defaults-conf.spark.executor.memoryOverhead": "512",
                    "yarn-site.yarn.nodemanager.resource.memory-mb": "512000",
```

```
                    "yarn-site.yarn.nodemanager.resource.cpu-vcores": "64",
                    "yarn-site.yarn.scheduler.maximum-allocation-mb": "512000",
                    "yarn-site.yarn.scheduler.maximum-allocation-vcores": "8",
                    "yarn-site.yarn.scheduler.capacity.maximum-am-resource-percent":
"0.10"
                }
            }
        }
    }
}
```

A.2 Control.json

```
{
  "apiVersion": "v1",
  "metadata": {
    "kind": "Cluster",
    "name": "mssql-cluster"
  },
  "spec": {
    "docker": {
      "registry": "mcr.microsoft.com",
      "repository": "mssql/bdc",
      "imageTag": "2019-CU5-ubuntu-16.04",
      "imagePullPolicy": "IfNotPresent"
    },
    "storage": {
      "data": {
        "className": "test-unity-fc",
        "accessMode": "ReadWriteOnce",
        "size": "12Ti"
      },
      "logs": {
        "className": "test-unity-fc",
        "accessMode": "ReadWriteOnce",
        "size": "550Gi"
      }
    },
    "endpoints": [
      {
        "name": "Controller",
        "serviceType": "NodePort",
        "port": 30080
      },
      {
        "name": "ServiceProxy",
        "serviceType": "NodePort",
        "port": 30777
      }
    ],
    "settings": {
      "controller": {
        "logs.rotation.size": "5000",
        "logs.rotation.days": "7"
      },
```

```
      "ElasticSearch": {
        "vm.max_map_count": "-1"
      }
    }
  },
  "security": {
    "allowDumps": true,
    "allowNodeMetricsCollection": false,
    "allowPodMetricsCollection": false
  }
}
```

B Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Storage technical white papers and videos provide expertise that helps to ensure customer success on Dell EMC storage platforms including Dell EMC Unity XT and many others.

B.1 Related resources

- Dell EMC Unity Performance Best Practices
- Dell Technologies OpenShift Platform
- Kubernetes
- SQL Server Big Data Cluster Workshops
- Dell Technologies GitHub
- Dell Technologies SQL Server Solutions