IBM Db2 Warehouse MPP on Red Hat OpenShift Container Storage

Highlights:

- Deploy tested and proven Red Hat OpenShift Container Storage for IBM Db2 Warehouse MPP on OpenShift.
- Configure flexible software-defined storage that is designed for OpenShift applications like IBM Db2.
- Customize application deployments with software-defined storage that supports file, block, or object modalities to optimize performance for essential functionality.
- Embrace the open hybrid cloud, deploying Red Hat OpenShift Container Storage anywhere that OpenShift runs.

Table of contents

Introduction 2

IBM Db2 Warehouse MPP and Red Hat OpenShift Container Storage 2

- Db2 Warehouse MPP 2
- A disaggregated approach within one OpenShift cluster 3
- Performance summary 4

Red Hat OpenShift Container Storage overview 6

- Ceph 7
- The Rook Storage Operator 8
- OpenShift Multicloud Object Gateway 8
- The Red Hat OpenShift Storage Operator 9

Configuring Red Hat OpenShift Container Storage for IBM Db2 9

- The Db2 shared storage zone 10
- The Db2 database partition zone 10
- Cloud-based or direct-attached storage 11
- Shared or disaggregated operation 11
- Replication and distribution of replicas 12
- Sample test configuration 12

Performance testing 12

- Workload description and sizing 13
- System performance 14
- System scalability 16
- Comparison with existing cloud-based infrastructure 17

Conclusion 18

- Appendix A: Sizing an IBM Db2 Warehouse deployment 18
- Appendix B: Provisioning IBM Db2 Warehouse on Red Hat OpenShift 20
- Appendix C: Red Hat OpenShift Container Storage cluster CRD 22

Introduction

As organizations move their mission-critical IBM Db2 applications to the hybrid cloud, they need tested and proven solutions that offer scalability, performance, resilience, and security. To aid these efforts, the IBM Db2 team has spent the last several years transforming its delivery and infrastructure toward a Kubernetes native Db2—tailored for hybrid and multiclouds and managed by OpenShift®. Providing an ideal technology combination, Red Hat® OpenShift provides thought leadership across the entire stack, delivering a feature set and performance in line with IBM's requirements for Db2.

By redefining the storage layer around Kubernetes-based systems, the community has positioned software-defined storage (SDS) as the de facto storage solution for cloud-based deployments. In particular, Red Hat OpenShift Container Storage offers a complete data services platform for apps running on Red Hat OpenShift and OpenShift Virtualization. Through this agile, scalable, portable, and highly available platform, storage services can be provisioned and de-provisioned on demand. Container-native storage offers consistent data services so organizations can extract more value from their data—wherever it resides—complementing Red Hat's strategy of using OpenShift as the single control hub for apps and infrastructure life cycle management.

IBM and Red Hat engineers have worked closely to ensure that IBM Db2 performs well with Red Hat OpenShift and Red Hat OpenShift Container Storage. Two significant initiatives support this approach:

- An architectural initiative defines the integration and its associated performance, while reviewing protocols throughout the stack.
- A practical initiative explores IBM Db2 and Red Hat OpenShift Container Storage deployments, validating scenarios essential to critical database workloads.

This paper summarizes IBM Db2 performance on Red Hat OpenShift Container Storage. It provides insights into the test harness and architectural choices that were made as a part of the process. Finally, it highlights best practices that may be useful to database and storage administrators.

IBM Db2 Warehouse MPP and Red Hat OpenShift Container Storage

The sections that follow provide an overview of the architectural approach and a performance summary for IBM Db2 testing using the Big Data Intelligence (BDI) workload running on Db2 Warehouse MPP (Massive Parallel Processing). Based on the test results of Red Hat OpenShift Container Storage compared to tests including other software-defined storage options, Red Hat OpenShift Container Storage is currently the preferred solution for Db2 Warehouse MPP on OpenShift.

Db2 Warehouse MPP

IBM Db2 comes with two form factors, both built on the Db2 Common SQL Engine, and aligned with the latest version:

- IBM Db2 is best suited for on-line transaction processing workloads, with a strong focus on transaction volume.
- IBM Db2 Warehouse is best suited for on-line analytical processing (OLAP) workloads, with a strong emphasis on data volume and query performance.

An OLAP workload was chosen for testing—due to the demanding nature of the workload and the desire to provide a cross-sectional view of full-stack performance. As such, the Db2 Warehouse form factor was appropriately selected. Db2 Warehouse can be deployed in either a single-node (SMP) or multi-node deployment designed for massively parallelized processing (MPP). In MPP deployments, Db2 Warehouse segments a query into smaller tasks that are then spread across multiple database partitions (Figure 1). Db2 is also able to coordinate multiple cores per Db2 process at scale. An MPP deployment has a minimum of three nodes and a maximum of either 24 or 60 nodes (depending on database partitions).[1]

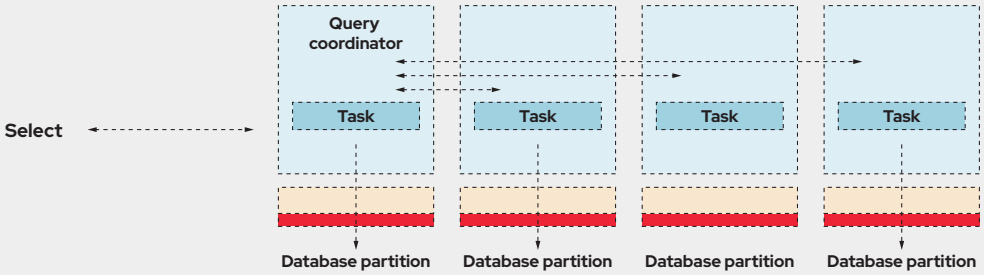

Figure 1. Db2 Warehouse MPP segments queries across multiple isolated database partitions.

A disaggregated approach within one OpenShift cluster

For this workload, IBM and Red Hat employed a *disaggregated* configuration where Db2 on OpenShift and Red Hat OpenShift Container Storage ran on physically separate nodes (Figure 2). Isolating compute from storage in this fashion provides advantages, including:

- The underlying capabilities of the worker nodes could be optimized for the respective workload. For example, memory- or CPU-intensive compute nodes could be configured as compared to storage-intensive nodes.
- Moreover, a disaggregated approach allows compute (or database) services to be conveniently scaled independently from storage services.

1 *https://www.ibm.com/support/knowledgecenter/en/SSCJDQ/com.ibm.swg.im.dashdb.doc/admin/local_prereqs.html*

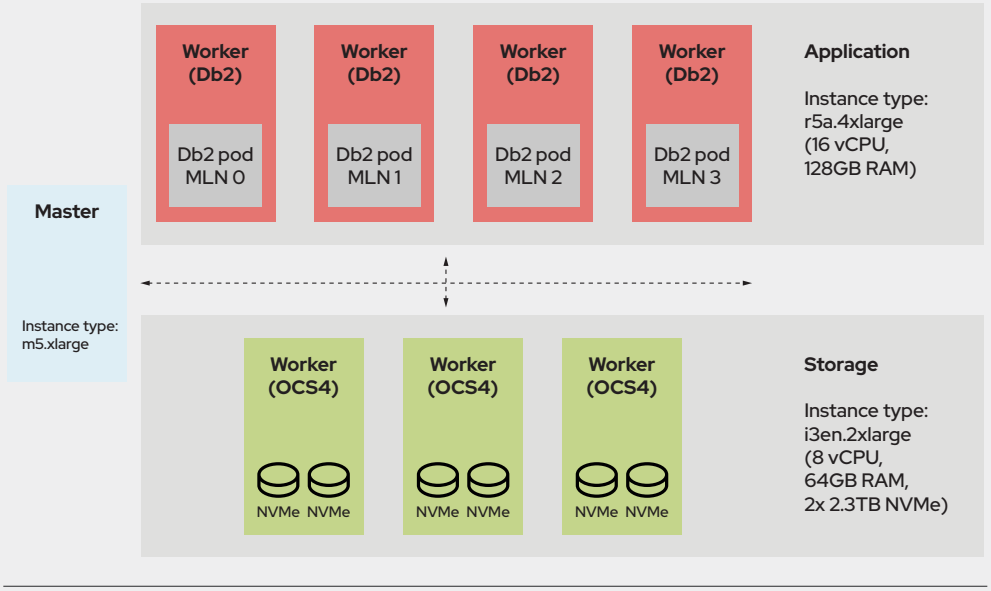

Figure 2. Testing took place on a disaggregated eight-node cluster.

Note: More information on the cluster configuration is provided in the section on performance testing.

Performance summary

Big Data Intelligence (BDI) is an IBM-defined workload that models a day in the life of a Business Intelligence application. The workload is based on a retail database with in-store, on-line, and catalog sales of merchandise. Three types of users are represented in the workload, running three types of queries:

- *Returns dashboard analysts* generate queries that investigate the rates of return and impact on the business bottom line
- *Sales report analysts* generate sales reports to understand the profitability of the enterprise
- *Deep-dive analysts* (data scientists) run analytics to answer questions identified by the returns dashboard and sales report analysts

The database can be generated at any scale factor. IBM testing included two databases, representing:

- A 1TB workload
- A 2TB workload

Several suites of tests were then run using the 1TB and 2TB workloads:

- Serial warmup runs (running all 100 queries end-to-end)
- Serial runs of three iterations (running through all 100 queries in serial mode)

- 16 concurrent heavy users running as many intermediate and complex queries as possible for one hour
- 32 concurrent heavy users running as many intermediate and complex queries as possible for one hour

The testing proved that Red Hat OpenShift Container Storage performs well at scale for production Db2 Warehouse MPP workloads running on OpenShift container clusters. For the 1TB workload, the tests ran mostly in memory. After doubling the workload to 2TB, performance only dropped by 44% demonstrating the scalability of the platform (Figure 3).

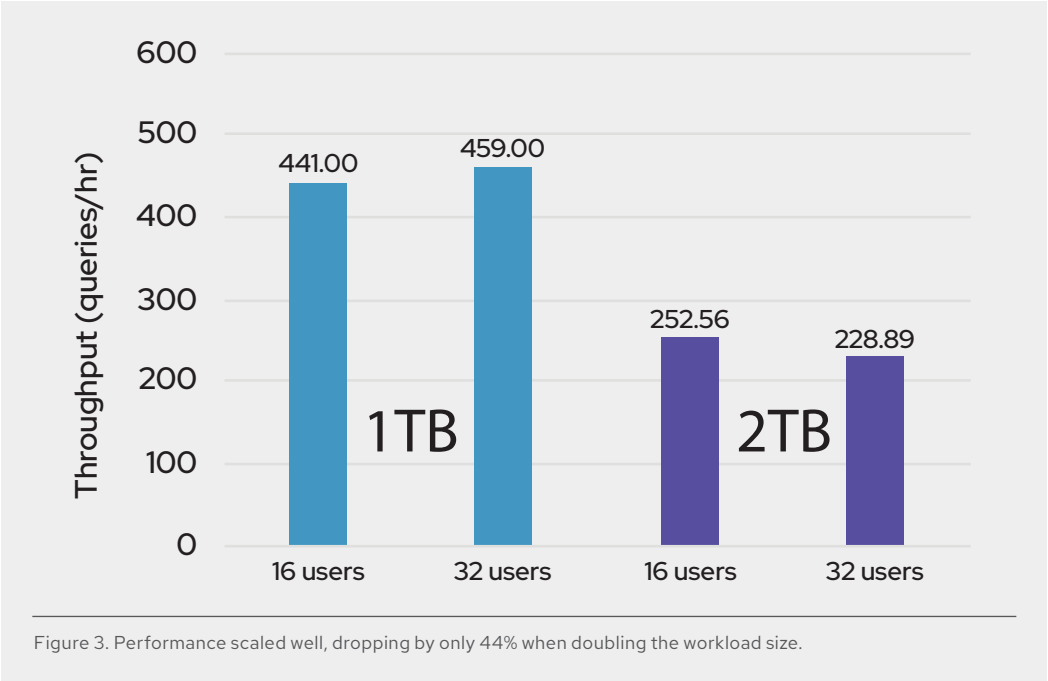

Figure 3. Performance scaled well, dropping by only 44% when doubling the workload size.

During testing, the cluster was instrumented to understand utilization and scalability. Results were then compared to other cloud-based configurations for Db2.

- **Resource utilization.** Testing showed that CPU, memory, disk, and network resources were all utilized very effectively during both the 1TB and 2TB runs. CPU and disk utilization showed available headroom on a per-system basis, and memory and Network I/O utilization rates were consistent across all four Db2 nodes and all three Red Hat OpenShift Container Storage nodes.
- **System scalability.** Utilization rates increased when moving from the 1TB to the 2TB workloads, demonstrating good system scalability to handle the larger workload. Higher CPU and disk utilization with the 2TB workloads indicate that system resources were used effectively. For most tests, runtime increased symmetrically with data volume. For the multi-user runs, a 2x increase in data size caused only a 1.75x reduction in throughput.

- **Performance comparison.** In order to better characterize the performance of Db2 Warehouse running on Red Hat OpenShift Container Storage, results were compared to tests performed with other cloud-based storage configurations for Db2, taken over the last two years. As some older results were not directly comparable, the results of the 1TB runs were compared and normalized with those of different cloud storage configurations. This comparison demonstrated that Db2 running on Red Hat OpenShift Container Storage performs very well compared to other software-defined storage approaches.

Red Hat OpenShift Container Storage overview

Red Hat OpenShift Container Storage offers the preferred persistent storage for cloud-native applications like Db2 running in OpenShift. Application teams can dynamically provision persistent volumes (PVs) for a wide variety of workload categories. The platform offers:

- Agility to streamline app/dev workflows across the hybrid cloud.
- Scale to support emerging data-intensive workloads for Kubernetes customers.
- Portability to provide data placement and access across clouds.

As Kubernetes has emerged and grown as the vehicle of modern development, complex stateful workloads like Db2 have driven the need for persistent container-native storage. OpenShift Container Storage is designed and built to supply all the needs of modern stateful applications in the Kubernetes domain, offering advantages that include:

- Platform agnostic, supporting cloud, bare-metal, and virtualized environments
- Open source, based on Ceph, Rook, and NooBaa
- Simple to install, update, and use
- Scalable
- Resilient
- Performant
- Support for block, object and file storage in a single product

One of the key features of OpenShift Container Storage is its simplicity and ease of installation stemming from the Kubernetes orchestration framework and the use of operators. Operators are custom software extensions to Kubernetes that make use of custom resources to automate and manage applications and their components. The *Red Hat OpenShift Container Storage operator* is a single meta-operator, providing one interface for installation and management for three components:

- **Ceph.** Red Hat Ceph® Storage is the central storage building block providing the storage services data plane inside Red Hat OpenShift Container Storage.
- **Rook.** Rook is the Kubernetes operator for Ceph, automating many Ceph deployment, operations, and upgrade tasks.
- **NooBaa.** The OpenShift Multicloud Object Gateway (MCG) is an object data service based on NooBaa technology that delivers policy-based data placement across hybrid and multicloud environments.

Ceph

Since its inception in 2007, Ceph has become one of the leading software-defined storage (SDS) solutions with clusters running on large production environments that can consist of hundreds of nodes. As an elastic storage services infrastructure, Ceph allows you to scale up with bigger/faster hardware and scale out for capacity and performance. Ceph also enables the federation of multiple clusters across sites with asynchronous replication and disaster recovery capabilities.

As a contributor to the upstream Ceph project, Red Hat ensures that features are ready for enterprise workloads and incorporates them into Red Hat Ceph Storage. As the foundational storage technology inside Red Hat OpenShift Container Storage, Red Hat Ceph Storage provides a robust and compelling data storage solution that can support all of your data, no matter the format or origin. A self-healing, self-managing platform with no single point of failure, Red Hat Ceph Storage can significantly lower the cost of storing enterprise data and help companies manage exponential data growth in an automated fashion.

Ceph is based on the Reliable Autonomic Distributed Object Store (RADOS, Figure 4). Unlike storage that only supports a single consumption mechanism, Red Hat Ceph Storage offers support for multiple storage modalities, including:

- Block storage via the RADOS Block Device (RBD).
- Object storage via the RADOS Gateway (RGW).
- File storage via CephFS.

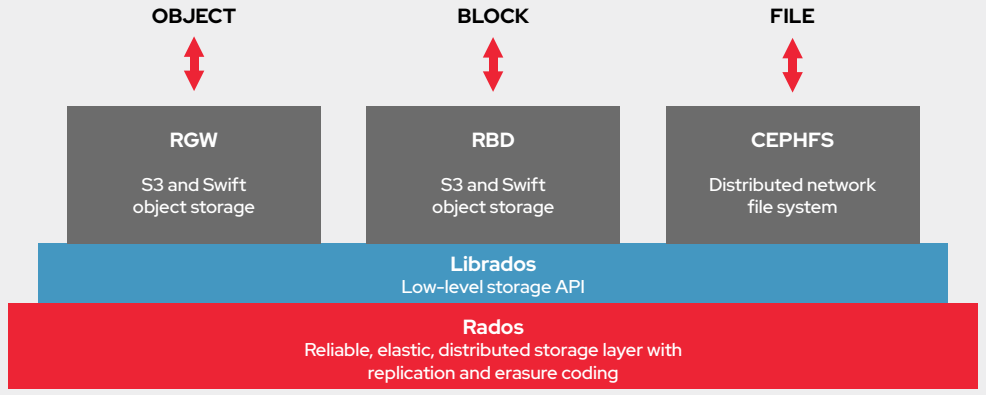

Figure 4. Ceph high-level block diagram.

From an architectural perspective, Ceph provides software modules that are designed for different tasks and responsibilities, including:

- **Object Storage Daemons (OSDs).** OSDs consume a storage device (e.g., a local drive, a partition, a SAN LUN, or a cloud provider volume) and map the device as part of a larger group of other OSDs to provide continuous storage to be used by applications.

- **Monitoring servers (MONs).** The MONs create the interface to the actual storage used by applications. MONs contain the current active Ceph cluster map. The MONs hold a list of OSDs and a list of MONs, distributing the load between the clients while assuring that specific quorum rules exist to make sure of data availability and accessibility.
- **Metadata servers (MDS).** The MDS maintains information about placement groups (PGs, a method to manage millions of storage objects efficiently) and also information on the metadata/host processes. The MDS adds POSIX metadata to objects so they can be consumed as files through a distributed file system (CephFS). The MDS also provides a RESTful API to monitor the cluster.

With Red Hat Ceph Storage inside, Red Hat OpenShift Container Storage users enjoy all the features of an integrated cloud-native persistent storage solution, while benefiting from the robust pedigree of an enterprise proven data services platform.

The Rook storage operator

Rook is a Kubernetes Operator designed to facilitate Kubernetes management of storage. It is an open source (Apache 2.0) Cloud Native Computing Foundation (CNCF) hosted project and includes code for several storage providers/consumers, including Ceph, Cassandra, CockroachDB, the Network File System (NFS), and EdgeFS. Ceph and EdgeFS are the most active providers in the Rook project.

To manage and automate storage, Rook follows the Kubernetes operator patterns that include:

- Automating processes that humans would typically perform.
- Observing and discovering the current state of the cluster.
- Analyzing and determining if any differences from the desired state exist, and performing operations to get to the desired state.

Rook uses Custom Resource Definitions (CRDs) to manage the storage provider, allowing automated deployment, configuration, provisioning, scaling, upgrading, and resource management of storage in the Kubernetes cluster. This approach retains Ceph principles so that Ceph components (OSDs, MONs, MDS, and RGW) are delivered in pods within the Kubernetes cluster.

OpenShift Multicloud Object Gateway

The OpenShift Multicloud Object Gateway (based on the NooBaa open source project) allows single scalable storage access to object storage. Using the Amazon Simple Storage Service (S3) API, OpenShift Multicloud Object Gateway lets you control the number of bucket copies and where to place the copies (on-prem, VM, or cloud provider) based cost considerations or security. For example, the technology supports:

- Multicloud buckets
- Hybrid buckets
- Multisite buckets

In Red Hat OpenShift Container Storage, the OpenShift Multicloud Object Gateway keeps a copy of the bucket in Red Hat Ceph Storage, but you can easily add other locations. The NooBaa Kubernetes Operator manages deployment and second-day operations, while the NooBaa core pod manages the data flow and provides the object as a service.

The Red Hat OpenShift Storage Operator

The Red Hat OpenShift Storage Operator provides the management and organization layer for the core components mentioned previously, including Ceph, Rook, and NooBaa (Figure 5). The operator allows you to install all three components at once. It acts as one place to control and monitor all storage layers as well as the layout of Red Hat OpenShift Container Storage core components. It also facilitates day-two management operations, such as coordinating software updates for each of the installed components.

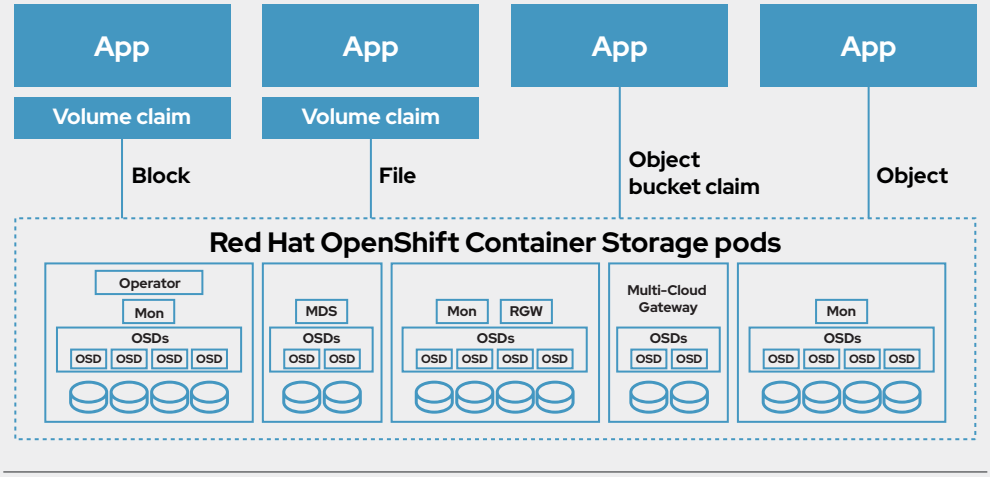

Figure 5. The Red Hat OpenShift Storage Operator provides a management and organization layer for all included software components.

Configuring Red Hat OpenShift Container Storage for IBM Db2

Db2 database architecture dictates how software-defined storage must be configured in an OpenShift context. While Db2 on OpenShift deployments support some storage variations, the correct configuration needs to be provisioned upfront to avoid outages later in the process. By supporting both file and block storage, Red Hat OpenShift Container Storage provides key flexibility that allows you to fine-tune storage configurations for the specific needs of Db2.

Db2 Warehouse represents a shared-nothing database architecture, in which database partitions are fully isolated from each other—visible only to a single Db2 pod in a topology of many. At the same time, some minimal information needs to be shared across the entire Db2 deployment. For instance, while partition data can be entirely local to achieve the fastest input/out (I/O) rates, the Db2 instance directory is a core installation requirement that needs to be shared across multiple pods. As such, two storage zones are required to achieve optimal Db2 performance (Figure 6):

- A shared storage zone (RWX access pattern) for when multiple Db2 pods need to read and write to a storage area from within the entire Db2 ecosystem.
- A dedicated storage zone (RWO access pattern) for each database partition in an individual pod to read and write to a storage area.

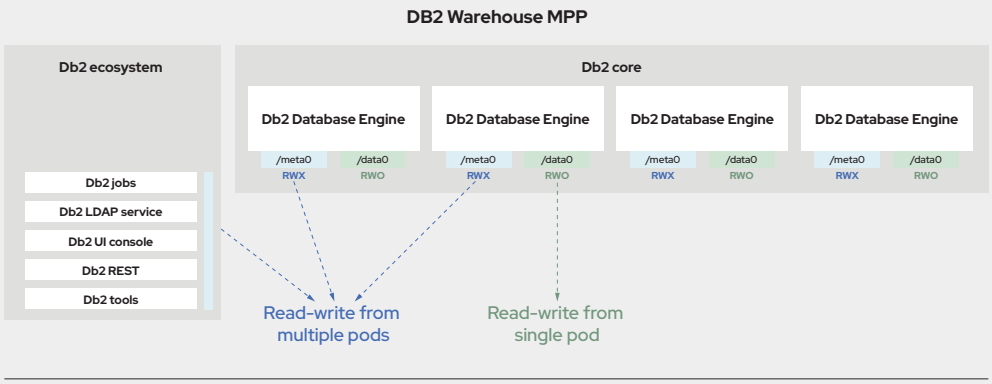

Figure 6. Db2 requires both dedicated (RWO access pattern) and shared (RWX access pattern) storage.

With Red Hat OpenShift Container Storage, both the type of storage (e.g., file or block) and the storage technology itself (e.g., cloud-based storage volumes or direct-attached storage) can be configured appropriately to match workload needs.

The Db2 shared storage zone

The Db2 shared storage zone has low I/O operations per second (IOPS). As such, this zone can be backed by file-type storage within Red Hat OpenShift Container Storage, yielding performance typical of network-attached storage (NAS) solutions. Db2 will use this storage space to host the following data:

- Db2 instance home directory
- Database manager configurations
- Db2 error messages file
- Db2 installation path
- Deployment metadata such as the key-value topology stored by each Db2 pod

The Db2 database partition zone

In contrast, the Db2 database partition storage zone has very high IOPS requirements and stringent performance expectations. In this implementation, this storage zone is backed by *block-type storage* within Red Hat OpenShift Container Storage, similar in performance profile to a storage area network (SAN). Storage for the database partition zone needs to be able to store:

- Storage path information
- Table space information
- Data

- Indexes
- Configuration files
- Transaction logs
- Database configuration files

Cloud-based or direct-attached storage

In a Ceph cluster, an OSD process represents a storage device that it can use to build the storage cluster. In Red Hat OpenShift Container Storage, however, an OSD is represented as a pod. For that pod to use a storage device, it needs a persistent volume (PV) and a persistent volume claim (PVC) for that volume. When designing an OpenShift cluster together with Red Hat OpenShift Container Storage, the type of the PV used for the OSD pods will determine or impact some of the storage characterizations.

In the cloud, you have the choice of storage options to use as the building block for Red Hat OpenShift Container Storage. You can use a portable volume provided by the cloud provider, such as an Amazon Elastic Block Storage (EBS) volume. You can also use direct-attached storage that is connected directly to the cloud-provider instance. When using the portable volume option (e.g., the default Amazon GP2 EBS volume), the volume is transferable between instances on the same availability zone. If an OSD pod fails, a new OSD pod can start quickly on the same or different instance in the cloud using the same data storage volume (i.e., the very same PV). This ability can significantly reduce the recovery point objective (RPO), since the Red Hat Ceph Storage data is still available to the new pod, with only some minor roll-forwards needed to get back to three full working replicas.

The disadvantages of using a cloud-provided volume include both performance and price. The performance capabilities of the volume usually derive from the size of the volume, with a larger volume equating to more IOPS. In general, the performance of these volumes is slower than regular storage devices such as a solid-state drive (SSD) directly connected to a server.

Another option for Red Hat OpenShift Container Storage is to use direct-attached storage, as is available with an AWS i3en instance. In this scenario, a storage device is directly connected to the instance running Red Hat OpenShift Container Storage. As mentioned, the performance can be significantly better for most use cases, even while the total storage price is less. When running on bare-metal clusters, using direct-attached storage devices is preferable. The beauty of Red Hat OpenShift Container Storage is that all of these options are just PVs that the OSDs will use to create the Ceph cluster—whether you use a volume from a cloud provider or a direct-attached storage device. This transparency helps accelerate hybrid cloud adoption.

Shared or disaggregated operation

As discussed, another important consideration is whether the OpenShift worker nodes will share resources with Red Hat OpenShift Container Storage or not. A truly *shared* solution—where each worker node both provides and consumes storage—is native to the Red Hat OpenShift Container Storage architecture. However, Kubernetes must have enough resources to run all the needed OpenShift container pods.

Another option is to run Red Hat OpenShift Container Storage services on their own OpenShift infrastructure nodes. In this *disaggregated* option, a set of OpenShift infrastructure (infra) nodes runs only OpenShift and Red Hat OpenShift Container Storage services. The rest of the worker nodes are reserved for the applications running on the cluster (Db2 in our case).

Choosing this disaggregated cluster approach allows you to optimize performance by selecting different types of nodes for storage services compared to those used for the rest of the applications running on the OpenShift cluster. Disaggregated operation also allows you to scale compute (app) or storage services (Red Hat OpenShift Container Storage) independently from each other.

Replication and distribution of replicas

Red Hat OpenShift Container Storage provides resiliency through Ceph replication. The current release employs a replication factor of three (3x), which implies that three copies of your data are distributed across all the available OSDs in the cluster. With Red Hat OpenShift Container Storage, you can specify the rules that determine how you want the copies distributed. For example, you could place a copy of your data in separate racks in an on-prem cluster, or within different Availability Zones in a cloud deployment.

Sample test configuration

As described, the testing described in this paper ran on AWS using Red Hat OpenShift Container Storage in a disaggregated configuration.

- *Storage providers* (Red Hat OpenShift Container Storage services) were built on three AWS *i3en.2xlarge* instances running as OpenShift infra nodes, each with two 2.3TB NVMe directly attached to the instance.
- *Storage consumers* (Db2 instances) were supported on four AWS *r5a.4xlarge* instances running the Db2 Warehouse MPP pods.

This configuration (with 3x replication) provided 4.6TB of available storage for Db2 Warehouse MPP, divided between RWO PVCs (Ceph RBD/block) and RWX PVCs (CephFS). The MON pods consumed very little storage and used a 10GB EBS GP2 volume for each pod. To allow direct-attached storage to be consumed by the OSD pods, the Local Storage Operator (LSO) was used to create local PVs matching each of the NVMe devices on each of the *i3en.2xlarge* instances. Instructions for this method are part of the product documentation.

Performance testing

IBM and Red Hat performance testing was conducted using AWS instances, as described elsewhere in this document. Consistent with best practices, each compute node was configured as follows:

- 1 database partition per node
- 104GB of memory
- 13 vCPUs

Total Db2 capacity was:

- Four Db2 compute nodes
- Four database partitions
- 52 vCPUs
- 416GB of memory

Workload description and sizing

The BDI workload database schema follows that of the TPC Benchmark DS (TPC-DS) decision support benchmark specification. It contains seven fact tables (store sales, store returns, catalog sales, catalog returns, web sales, web returns, and inventory) and 17 dimension tables. The workload contains 100 queries inspired by Cognos10-generated SQL for dashboards and reports, Cognos10-generated SQL for in-memory dynamic cubes, as well as industry-standard benchmark queries. The database can be generated at any scale factor. Two workloads were configured for IBM and Red Hat testing:

- A 1TB BDI workload with Scale Factor 1000 (1 TB of raw data) was initially established to analyze the performance of Db2 with Red Hat OpenShift Container Storage.
- A 2TB BDI workload was then configured to investigate the scalability of Db2 on Red Hat OpenShift Container Storage.

Three types of simulated users are represented in the workload, running three different types of queries:

- **Simple queries (70 queries).** *Returns dashboard analysts* are investigating rates of return and impact on the bottom line of the business. These users run simple queries with a narrow range of data with a runtime range of sub-second to one second.
- **Intermediate queries (25 queries).** *Sales report analysts* generate sales reports to understand the profitability of the retail enterprise. These users run queries of intermediate complexity with a broader range of data and a runtime range of up to one minute.
- **Complex queries (5 queries).** *Deep-dive analysts* (data scientists) handcraft more in-depth analysis to answer questions identified by the returns dashboards and sales report analysts. These users run the most complex SQL with an extensive data range and several minutes of runtime.

The BDI workload has two different modes of execution, including:

1. *Serial Execution mode*, where a single user iterates through all 100 queries in the workload to capture end-to-end elapsed time performance or isolate single-query performance.
2. *Concurrent Throughput test mode*, where the open source Apache jMeter load-generation tool drives the workload. Each connection (user) repeatedly executes a particular set of query scenarios (simple, intermediate, or complex) for as many times as possible in a given time interval. You can specify any number of users and pick from different user types. For example, heavy users might run only from the set of intermediate and complex queries. Each simulated user runs the same set of queries but in a shuffled order. The results are based on queries per hour.

The following suite of tests was run to analyze the performance of Db2 with Red Hat OpenShift Container Storage:

- **Serial warmup**—runs all 100 queries end-to-end
- **Serial run, three iterations**—runs through all 100 queries, three times in serial mode
- **16 heavy users static**—16 concurrent users run as many intermediate and complex queries as possible for 1 hour
- **32 heavy users static**—32 concurrent users run as many intermediate and complex queries for 1 hour

System performance

As shown in the Performance Summary (Figure 3), the system scaled predictably between the 1 TB and 2TB workloads. For a better understanding of the system performance during the BDI runs, nmon data were captured and analyzed with the NMONVisualizer, a Java graphical user interface (GUI) tool for analyzing nmon system files.

CPU utilization

This data revealed that the system under test made excellent use of the available system resources. Figure 7 shows the NMONVisualizer output for the 16 and 32 heavy multi-user runs, illustrating system utilization for all of the Db2 and Red Hat OpenShift Container Storage nodes in the cluster.

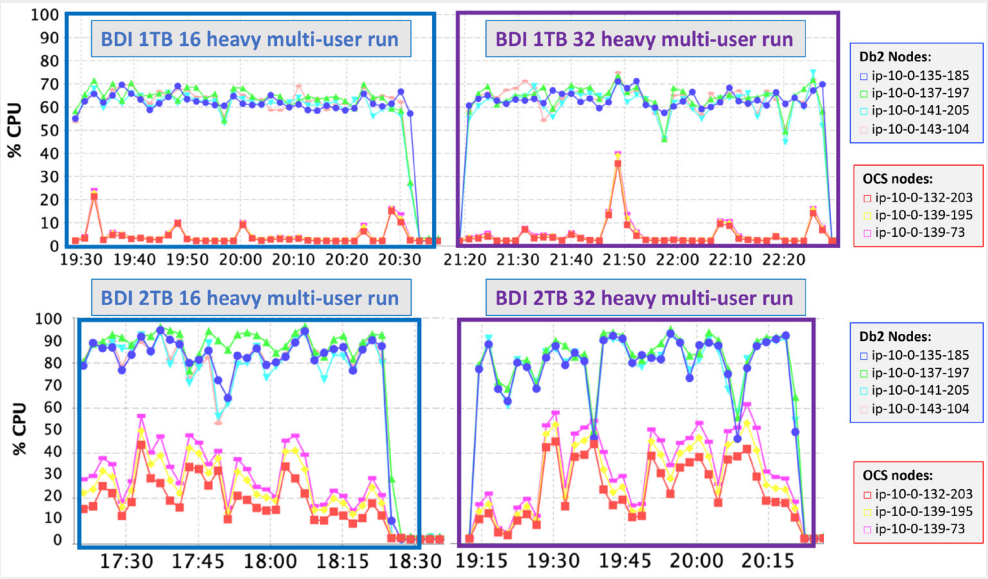

Figure 7. NMONVisualizer output for the 16 and 32 heavy multi-user runs showing system utilization for each node.

Overall, the available CPU resources were used effectively. Several conclusions can be drawn from the NMONVisualer display:

- CPU utilization averaged around 65% during the 1TB 16- and 32-heavy multi-user runs on the Db2 nodes.
- CPU utilization averaged around 90% during the 2TB 16- and 32-heavy multi-user runs on the Db2 nodes.
- CPU activity on the nodes running Red Hat OpenShift Container Storage increased with the workload size, and as the number of users increased.

Disk utilization

Disk utilization data is shown in Figure 8. The 1TB test runs occupied 42% of the available disk space and allowed most of the data to fit into the buffer pool. In contrast, the 2TB configuration occupied 85% of disk space and allowed only about 25% of the data to fit into the buffer pool.

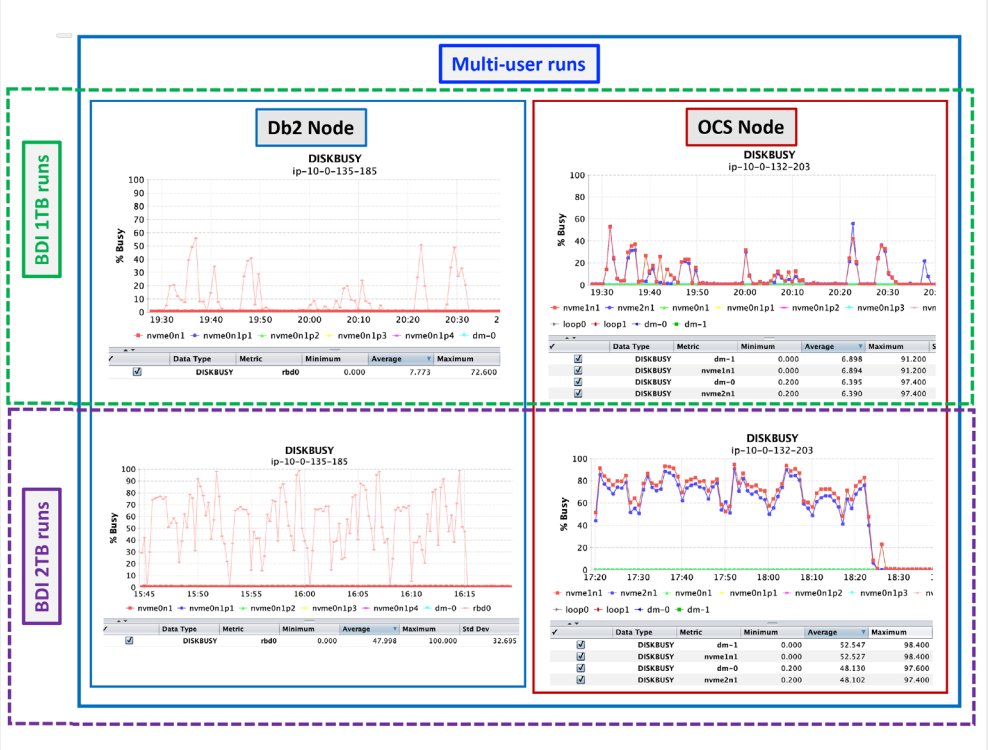

Figure 8. Disk utilization.

These data show that the available disk resources were used effectively and lead to several conclusions:

- The system is not disk-bound during the 1TB BDI runs.
- With increased data volume, the system shows significantly more disk activity during the 2TB BDI runs on both the Db2 and Red Hat OpenShift Container Storage nodes.
- Disk utilization is very similar across all four Db2 nodes and across all three Red Hat OpenShift Container Storage nodes.

Memory and network utilization

Network and Memory utilization appeared healthy and did not present a performance bottle-neck. Figure 9 depicts the memory and network utilization of a Db2 node and a Red Hat OpenShift Container Storage node during stress testing the system with the 2TB 16- and 32-heavy multi-user runs. Memory usage and Network I/O are very similar across all four Db2 nodes and across all three Red Hat OpenShift Container Storage nodes.

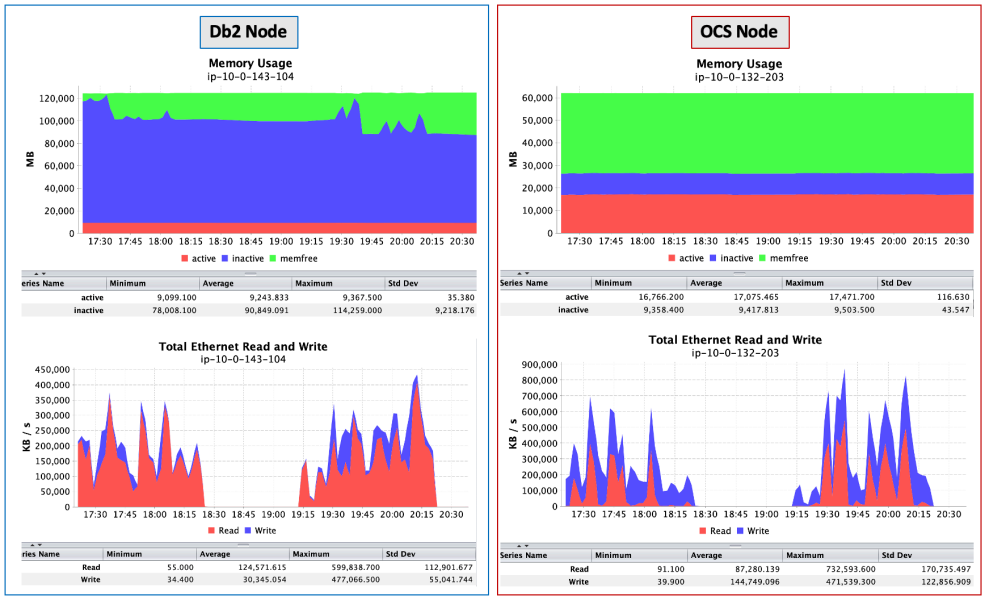

Figure 9. Memory and network utilization.

System scalability

To evaluate how well Db2 on Red Hat OpenShift Container Storage scales, the elapsed time from all of the 1GB runs were compared with those of the 2TB configuration. Doubling the workload size from 1TB to 2TB showed good system scalability. High CPU and disk utilization pointed to system resources being utilized effectively:

- **Serial runs.** CPU utilization increased from ~60% (1TB configuration) to ~80% on the Db2 nodes (2TB configuration).
- **Multi-user runs.** CPU utilization increased from ~65% (1TB configuration) to ~90% on the Db2 nodes (2TB configuration).
- **Disk utilization.** Disk utilization increased significantly in the 2TB runs, which is expected as most of the data no longer fits into the buffer pool.
- **Warmup run time.** Warmup run time increased by a factor of ~2x as the data volume increased by 2x.

- **Serial iterations.** Serial three-iteration run time increased by a factor of ~2x as data volume doubled.
- **Multi-user runs.** For the multi-user runs, a doubling in data size caused only a 1.75x reduction in throughput.

Comparison with existing cloud-based infrastructure

Engineers also wanted to compare the performance of Db2 Warehouse running on Red Hat OpenShift Container Storage with other cloud-based storage configurations. To that end, they compared the performance of Db2 running on Red Hat OpenShift Container Storage with that of Db2 running on a block storage solution provided by the cloud service provider. Both configurations utilized four nodes and 64 logical CPUs in total. The existing cloud infrastructure configuration had 50% more total RAM available.

The same 1TB BDI workload test suite of serial and multi-user runs was performed on both systems. The performance results were then normalized by leveling the numbers on the amount of memory and compared on a relative scale. The comparison demonstrated that there are no significant performance differences or concerns and that Db2 running on Red Hat OpenShift Container Storage performs on par with Db2 on existing cloud-based storage offerings today (Figure 10).[2]

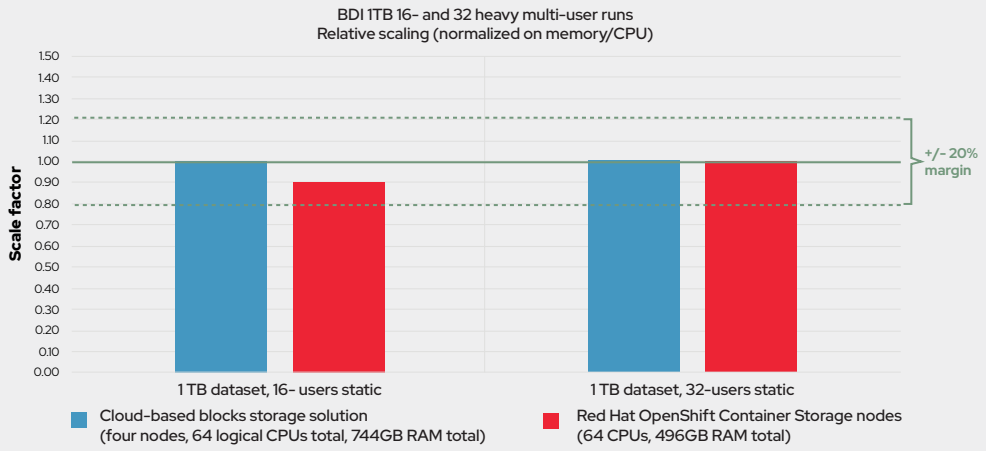

Figure 10. Comparison between Db2 on Red Hat OpenShift Container Storage and Db2 running on a cloud-based block storage solution.

2 *We allowed for a 20% margin because we are using extrapolation instead of direct experimentation, and the numbers will be influenced by other factors not in our control.*

Conclusion

Db2 has transformed its deployment model in order to provide a modern and cloud-native experience with Red Hat OpenShift. Built on Kubernetes, Db2 relies on storage classes and dynamic provisioning to support storage allocation. Red Hat OpenShift Container Storage is designed and tested in tandem with Red Hat OpenShift. Testing has shown that Red Hat OpenShift Container storage provides a credible and performant storage solution for IBM Db2, offering scalability, predictable system utilization, and an equitable comparison with other cloud-based block storage solutions. In addition, for the most resource-intensive version of Db2 Warehouse MPP, Red Hat OpenShift Container Storage is currently the preferred storage solution for IBM Db2 Warehouse MPP on OpenShift.

Appendix A: Sizing an IBM Db2 Warehouse deployment

In Db2 terminology, a physical database partition is managed by a dedicated Db2 process, named db2sysc, and appropriately sizing the resources dedicated to that process is necessary for operational efficiency. In addition, the Db2 deployment is designed to host multiple database partitions per OpenShift container, allowing for both intra-partition and inter-partition parallelism. While a higher partition number may boast better performance, an optimal number is not only dependent on the memory to core ratio, but is also heavily reliant on the ability to scale out or provide failover.

The Db2 node file

With every Db2 deployment, an internal file named *db2nodes.cfg* provides the deployment topology that Db2 will use at runtime. This essential file works by listing all the Db2 hosts as well as their owning partitions. It also binds the worker nodes, each hosting a Db2 Common SQL Engine container runtime, and establishes their participation as a distributed Db2 instance.

The Db2 nodes file is generated dynamically at deployment time, based on the replica and database partition values provided. Hostname values are dynamically resolved, based on the OpenShift scheduling assignment.

Horizontal scaling

When deploying Db2 Warehouse, having a database partition number able to scale out toward additional compute nodes is an important consideration. This day 2 operation, which would reappropriate database partitions to a new number of compute nodes, needs to be planned prior to deployment, in order to expand or contract the cluster without requiring data movement. Moving data, on the contrary, is a time-consuming and sensitive operation that isn't appealing for production environments.

For example:

- Three database partitions distributed over three worker nodes may comprise an ideal configuration today, but wouldn't leave any room for expanding compute any further.
- Six database partitions distributed over three worker nodes would allow compute resources to double in the future if desired.

The TOTAL_DB_PART_NUMBER needs to be divisible by the number of Db2 worker nodes to allow for reappropriation and scaling. Figure 11 illustrates an example where three database partitions per compute node are distributed across four compute nodes while avoiding any costly and time-consuming data movement. The configuration can be scaled out by moving to six Db2 worker nodes with two database partitions per node. The critical consideration is that the total number of database partitions stays constant (12 in this case).

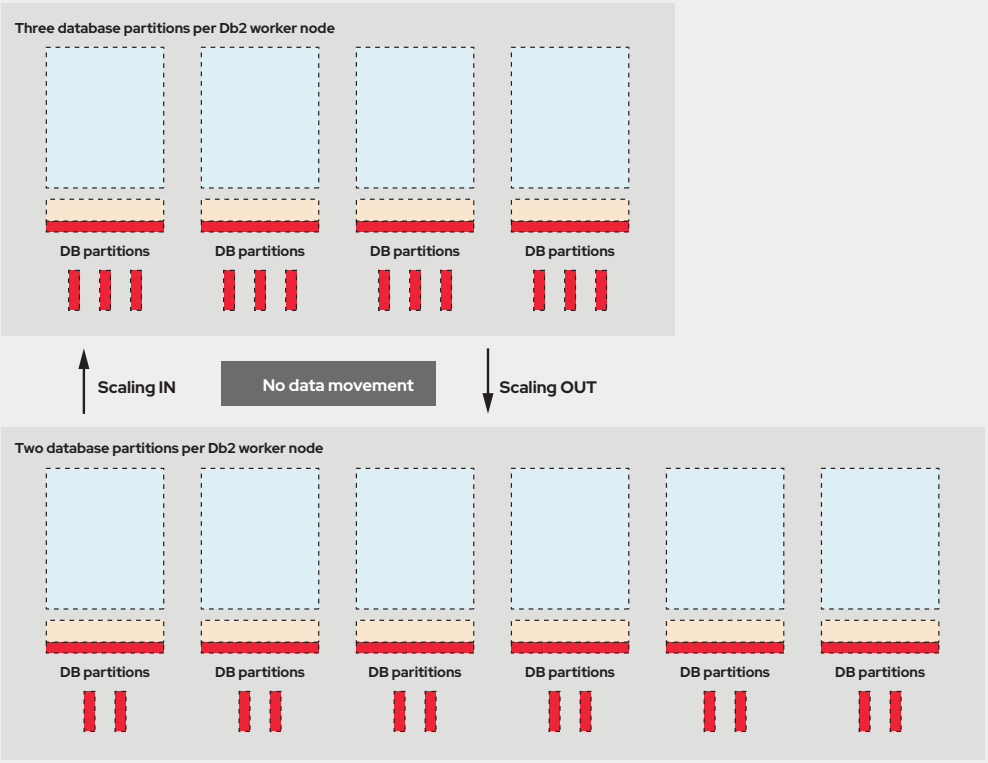

Figure 11. Maintaining TOTAL_DB_PART_NUMBER that is divisible by the number of Db2 worker nodes allows for horizontal scaling without data movement.

Parallelization with database partitions

A Db2 database partition, is designed to manage and host the following information:

- Data
- Indexes
- Configuration files
- Transaction logs

Best practice

Sizing the ideal partition number per Db2 Compute Node

A recommended range of [1-7] database partitions per OpenShift Db2 compute node

Compute and memory

Sizing for a database deployment requires a combination of best practices as well as a deep understanding of the underlying process and storage architecture. For a Db2 Warehouse deployment, the following considerations are paramount to a successful experience:

- Sufficient Memory and Core per database partition
- A range of [8GB-32GB] of memory per physical core with a recommended ratio of 16:1, or higher

Best practice

Sizing Db2 CPU & Memory

"High end system": CORE_PER_DB_PART = MEMORY_PER_DB_PART / 32

"Recommended system": CORE_PER_DB_PART = MEMORY_PER_DB_PART / 16

"Entry system": CORE_PER_DB_PART = MEMORY_PER_DB_PART / 8

Best practice

Sizing OpenShift Container Platform

Red Hat OpenShift Container Platforms requires two CPUs and 8GB per compute node

- It is generally desirable to have a number of cores evenly divisible by the number of database partitions, per host.

Finally, the OpenShift sizing also needs to be considered on each compute node.

Appendix B: Provisioning IBM Db2 Warehouse on Red Hat OpenShift

IBM Db2 on OpenShift is comprised of the following Kubernetes resources:

- The Db2 StatefulSet. This Kubernetes resource is ideally suited for databases as it allows for a scaling factor with strong semantic consistency around ordering, network, and storage stability. The Db2 StatefulSet includes the Db2 Common SQL Engine.
- The ETCD StatefulSet. The ETCD StatefulSet provides a performant key-value store able to store metadata on the Db2 distributed instance.
- The Db2 Toolbox. The Db2 toolbox provides the Db2 control plane for day-two operations.
- The Db2 Authentication pod. The Db2 authentication pod is currently backed by Lightweight Directory Access Protocol (LDAP).
- Db2 Jobs. The Db2 jobs provide the deployment configuration or post-deployment operations and drive the Db2 Toolbox.
- A Db2 UI Console. The optional Db2 UI Console provides monitoring and Structured Query Language (SQL) administration functions.

Db2 is available in the IBM Charts GitHub repository catalog. For this testing, IBM Db2 Warehouse was deployed using the following parameters:

Shared storage

The CephFS storage class backs the instance directory to provide a shared storage zone.

Exclusive database partition

The CephRBD storage class backs each database partition to provide the data zone.

Retain policy

This policy is essential and allows for a full redeployment of Db2, targeting an already persisted database.

4K sector size

As Db2 has implemented elaborate direct I/O and current I/O (DIO/CIO) technology, the Db2 write block needs size consistency with the underlying disk sector size.

- Writing a 512-byte block in a 4K sector would underperform.
- Writing a 4K block size in a 512-byte sector would not fit.

Providing `instance.db2Support4K="true"` during deployment configures Db2 with the proper setting to be used with Red Hat OpenShift Container Storage.

Full helm command

```
 helm install \
--name ${REL} \
--namespace ${NAMESPACE} \
--set dedicated="true" \
--set runtime="ICP4Data" \
--set images.pullPolicy="Always"  \
--set images.tools.image.repository="${REG}/db2u.tools"  \
--set images.tools.image.tag="11.5.2.0-${VERSION}"  \
--set images.db2u.image.repository="${REG}/db2u"  \
--set images.db2u.image.tag="11.5.2.0-${VERSION}"  \
--set images.instdb.image.repository="${REG}/db2u.instdb"  \
--set images.instdb.image.tag="11.5.2.0-${VERSION}"  \
--set images.auth.image.repository="${REG}/db2u.auxiliary.auth"  \
--set images.auth.image.tag="11.5.2.0-${VERSION}"  \
--set images.etcd.image.repository="${REG}/etcd"  \
--set images.etcd.image.tag="3.3.10-${VERSION}" \
--set images.etcd.storage.persisted="true"  \
--set etcd.storage.persisted="false"  \
--set storage.useDynamicProvisioning="true"  \
--set storage.storageLocation.metaStorage.enabled="true"  \
--set storage.storageLocation.metaStorage.volumeType="pvc"  \
--set storage.storageLocation.metaStorage.pvc.claim.
 storageClassName="ocs-storagecluster-cephfs-retain"  \
--set storage.storageLocation.metaStorage.pvc.claim.
 useDynamicProvisioning="true"  \
--set storage.storageLocation.metaStorage.pvc.claim.size="1.3Ti" \
--set storage.storageLocation.dataStorage.enabled="true"  \
--set storage.storageLocation.dataStorage.pvc.claim.size="250Gi"  \
--set storage.storageLocation.dataStorage.volumeType="pvc"  \
--set storage.storageLocation.dataStorage.pvc.claim.
 storageClassName="ocs-storagecluster-ceph-rbd-retain"  \
```

```
--set storage.storageLocation.dataStorage.pvc.claim.
 useDynamicProvisioning="true"  \
--set storage.storageLocation.dataStorage.enablePodLevelClaim="true"  \
--set storage.enableVolumeClaimTemplates="true"  \
--set ldap.enabled="true"  \
--set ldap.ldap_server="${REL}-db2u-ldap"  \
--set servicename="${REL}" \
--set limit.cpu="13"  \
--set limit.memory="104Gi" \
--set images.db2u.replicas="4"  \
--set global.dbType="db2wh"  \
--set mln.total="4"  \
--set subType="mpp"  \
--set instance.db2Support4K="true" \
ibm-db2wh-prod/
```

Appendix C: Red Hat OpenShift Container Storage cluster CRD

The storage cluster used the following CRD:

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
metadata:
  name: ocs-storagecluster
  namespace: openshift-storage
spec:
  manageNodes: false
  resources:
    mds:
      limits:
        cpu: 3
      requests:
        cpu: 1
    noobaa-core:
      limits:
```

```
      cpu: 1
    requests:
      cpu: 1
  noobaa-db:
    limits:
      cpu: 1
    requests:
      cpu: 1
monPVCTemplate:
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
    storageClassName: gp2
    volumeMode: Filesystem
storageDeviceSets:
- count: 2
  dataPVCTemplate:
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 1
      storageClassName: localblock
      volumeMode: Block
  name: ocs-deviceset
  placement: {}
  portable: false
  replica: 3
  resources: {}
```

About Red Hat

Red Hat is the world's leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high-performing Linux, hybrid cloud, container, and Kubernetes technologies. Red Hat helps customers integrate new and existing IT applications, develop cloud-native applications, standardize on our industry-leading operating system, and automate, secure, and manage complex environments. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500. As a strategic partner to cloud providers, system integrators, application vendors, customers, and open source communities, Red Hat can help organizations prepare for the digital future.

North America
1 888 REDHAT1
www.redhat.com

Europe, Middle East, and Africa
00800 7334 2835
europe@redhat.com

Asia Pacific
+65 6490 4200
apac@redhat.com

Latin America
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com
#F24114_0620