Scaling Analytics Engine powered by Apache Spark on OpenShift

Table of contents

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

redhat.com

Highlights

A joint technology solution from IBM, Red Hat, and Intel offers analytics scale and concurrency with up to twice the performance of previously tested solutions.

Analytics Engine, powered by Apache Spark running on IBM Cloud Pak for Data, delivers the best-reported analytics performance with Red Hat OpenShift Container Platform and Red Hat OpenShift Data Foundation.

High-performance and high-endurance Intel Optane Solid State Drives (SSDs) for the Data Center (DC) accelerate Spark shuffle operations as well as serving as an effective metadata cache for OpenShift Data Foundation.

Intel data node configurations for Red Hat OpenShift allow rapid procurement of tested configurations to serve as OpenShift Data Foundation data nodes, easing the transition to Kubernetes data services.

Executive summary

Data analysis solutions are changing rapidly, shifting from relatively static and rigid analytics clusters toward more dynamic and scalable environments. Software-defined data services are a big part of this transition, allowing on-demand provisioning of clusters for analytics and data processing, sharing of data sets between these clusters, and independent and granular scaling of compute and storage resources. Working closely together IBM, Red Hat, and Intel engineers designed an optimized reference architecture and performed extensive testing on a cluster equipped with:

- Analytics Engine powered by Apache Spark.
- IBM Cloud Pak® for Data.
- Red Hat® OpenShift® Data Foundation.
- Red Hat OpenShift Container Platform.
- Intel® Xeon® Scalable Processors and Intel® Solid State Drives for the Data Center (including Intel® Optane™ SSD DC Series).

This document describes the test cluster reference architecture in detail along with the performance results achieved in the testing. The compelling performance results realized in laboratory testing indicate that this is an effective technology combination. Providing separate optimized resources for compute and storage clusters lets organizations share data sets easily even as they scale performance for Apache Spark-based workloads. Test results with a tuned storage cluster using Intel® processors and SSDs demonstrated why OpenShift Data Foundation is a preferred storage solution for Analytics Engine powered by Apache Spark.

Agility, performance at scale, and sharable data sets for Apache Spark

Traditional Hadoop-based Spark solutions were specialized and rigid. Creating new analytics clusters was time consuming and complex, and data sets could not be easily shared between clusters. Some teams needed access to the same data sets, but required different analytics tools or versions resulting in duplicate Hadoop clusters. Multiple separate Hadoop clusters soon proliferated in large organizations as a result. Static and inflexible analytics infrastructure impeded organizational agility and limited responses to rapidly changing world conditions.

Updating Apache Spark to run in containers governed by Kubernetes liberated analytics clusters from rigid Hadoop infrastructure towards easily extensible and declarative infrastructure. Kubernetes lets applications express desired state and operators make sure the application state is met, allowing applications to be dynamically configured and reconfigured in response to rapidly changing circumstances. Kubernetes allows application teams to dynamically configure and extend their applications through custom resource (CR) files.

While Kubernetes has solved the problem of managing distributed systems resiliently and allowed on-demand management and provisioning of containerized Spark clusters, it did not address sharing data sets between clusters. Sharing is most often made possible through the use of a common object store, in combination with filesystem clients like the S3A client from Hadoop common.[1] This client allowed Spark and other analytics tools to access shared data sets via a RESTful object storage protocol instead of traditional Hadoop Distributed File System (HDFS). Multiple Spark clusters can now share data sets from a single external object storage data lake.

[1] *For compatibility with the Amazon Simple Storage Service (S3).*

Analytics Engine powered by Apache Spark on Cloud Pak for Data (referred to subsequently as Analytics Engine for brevity) provides a fully supported Spark distribution running on Red Hat OpenShift Container Platform. As an industry-leading Kubernetes distribution within a full-featured DevOps platform, OpenShift Container Platform enables applications to be developed once, and then deployed on any underlying on-premise or public cloud running OpenShift Container Platform. This write-once-run-anywhere characteristic satisfies the needs of many organizations looking for common hybrid cloud infrastructure.

Red Hat OpenShift Data Foundation[2] is persistent software-defined storage integrated with and optimized for OpenShift Container Platform. It provides a fully supported Kubernetes container storage interface (CSI) driver for Kubernetes persistent volume (PV) services, as well as an S3-compatible object storage service. OpenShift Data Foundation relies on Ceph® as the underlying data plane technology. For data intensive applications, OpenShift Data Foundation provides a highly recommended Ceph RADOS gateway (RGW) object storage service that also supports a S3A Hadoop filesystem client. OpenShift Data Foundation exhibits high performance at scale as a result.

Within OpenShift Container Platform, disaggregating compute and storage resources makes the overall solution architecture extremely flexible. This architecture allows the compute cluster to be stateless, with all the data, metadata, and even container images stored in volumes provided by the external OpenShift Data Foundation cluster. Multiple Analytics Engine (Spark) compute clusters can be spun up and down on demand, while the data is persisted on highly available external OpenShift Data Foundation object clusters (Figure 1).

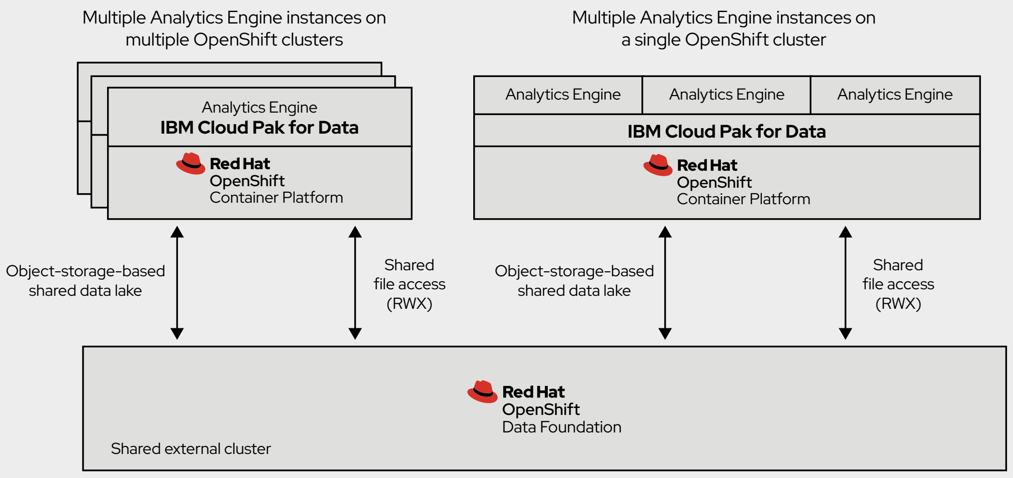

Multiple Analytics Engine instances on multiple OpenShift clusters
Analytics Engine
IBM Cloud Pak for Data
Red Hat OpenShift Container Platform
Multiple Analytics Engine instances on a single OpenShift cluster
Analytics Engine
Analytics Engine
Analytics Engine
IBM Cloud Pak for Data
Red Hat OpenShift Container Platform
Object-storage-based shared data lake
Shared file access (RWX)
Object-storage-based shared data lake
Shared file access (RWX)
Red Hat OpenShift Data Foundation
Shared external cluster

Figure 1. Red Hat OpenShift Data Foundation provides a shared external cluster for OpenShift Container Platform applications like Analytics Engine powered by Apache Spark

[2] *Red Hat OpenShift Data Foundation was formerly known as Red Hat OpenShift Container Storage.*

Combining these solutions into a common reference architecture provides a number of benefits.

- With compute and storage no longer bound together, data processing and analytical workloads can be scheduled on demand, accessing and persisting data as an inter/intra cluster shared resource.
- I/O-heavy workloads become more cost effective, with the ability to steer storage services towards optimized hardware.
- Analytics clusters become more elastic as both data and compute nodes can be added as needed.
- Analytics clusters can be spun up to meet the needs of the job, without waiting for lengthy software package installation times, lengthy data hydration cycles, and brittle scripting to maintain data consistency between multiple copies of the same data sets.

Performance highlights

The combined IBM, Red Hat, and Intel analytics solution delivered high-performance results with separation of storage and compute resources. The separation adds flexibility to the architecture by allowing both storage and compute to scale independently. A low-footprint, four-node OpenShift Data Foundation external-mode cluster deployed on Intel Optane SSD NVMe storage media delivered high performance. No bottlenecks were observed using simulated single and multiuser TeraGen, TeraSort, and TPC Decision Support (TPC-DS) workloads with data sets as large as 10TB.

- **Supporting concurrent analytics at scale.** This jointly architected state-of-the-art analytics solution successfully completed all 104 queries of the TPC-DS benchmark suite without any failures across different scale factors of up to 10TB. Multistream simulation with five concurrent users showed only a minimal 15% performance degradation exercising all 104 TPC-DS queries at the 1TB scale factor.
- **Best reported analytics performance.** The solution delivered up to twice the data processing performance compared to Red Hat's previously reported results in a similar Intel lab.[3] Testing employed Analytics Engine powered by Apache Spark in IBM Cloud Pak for Data v3.5 with Red Hat OpenShift Container Platform, Red Hat OpenShift Data Foundation, and Intel Optane SSD NVMe devices for Spark shuffle and Ceph metadata storage.
- **Performance improvements over previous release.** With optimized data layout and workload-specific tuning, Analytics Engine powered by Apache Spark delivered substantial performance improvements over previous releases across all 104 TPC-DS queries on a 1TB scale factor. With the optimizations tested in this project, Apache Spark 3.0 on IBM Cloud Pak for Data 3.5 delivered up to three times higher performance for the 1 TB TPC-DS scale factor when compared to previous testing on an un-optimized configuration based on Apache Spark 2.4 on IBM Cloud Pak for Data 3.0.[4]
- **Performance improvements with Dynamic Data Skipping.** With the Dynamic Data Skipping feature enabled, Analytics Engine showed a significant, additional performance improvement of up to 70% when compared to community Spark for several TPC-DS queries.

[3] *https://www.redhat.com/en/resources/openshift-container-storage-detail*

[4] *IBM Cloud Pack for Data 3.0 testing was based on a different hardware configuration.*

- **Higher endurance and performance with Intel Optane SSDs.** The Spark analytics workload access pattern for local Spark Shuffle space is write-intensive, as is the pattern for OpenShift Data Foundation metadata access. As a result, we recommend Intel Optane SSDs as the underlying media for both due to their extremely high write endurance and performance.

Solution overview and components

The components of the solution are described in this section.

Analytics Engine powered by Apache Spark

Analytics Engine powered by Apache Spark can be used as a compute engine to run analytical and machine learning jobs. An administrator must install Analytics Engine on the IBM Cloud Pak for Data platform as it is not available by default.[5]

Analytics Engine allows users to store data in an object storage layer such as OpenShift Data Foundation, spinning up Spark clusters when needed. Each time a job is submitted, a dedicated Spark cluster is created for the job. Users can specify the size of the Spark driver, the size of the executor, and the number of executors for the job, enabling predictable and consistent performance.

When a job completes, the Spark cluster is automatically deleted so that computational resources are available for other jobs. The service also includes interfaces that enable you to analyze the performance of your Spark applications and debug problems. Jobs can be submitted to Spark clusters in two ways:

- Specifying a Spark environment definition for a job in an analytics project
- Using the Spark job application programming interface (API)

IBM Cloud Pak for Data

IBM Cloud Pak for Data is a cloud-native solution that fosters productivity by allowing users to automate and govern the entire data and AI life cycle on a single unified interface. Running on OpenShift Container Platform you can run IBM Cloud Pak for Data on any cloud or on-premise environment. Regardless of where you deploy IBM Cloud Pak for Data, you can connect to your data no matter where it lives.

With integrated data and AI services from proprietary, third party, and open-source services, IBM Cloud Pak for Data fosters collaboration and enables you to choose from services to help you catalog, govern, transform and analyze your data, and operationalize artificial intelligence (AI). With IBM Cloud Pak for Data, raw data becomes trusted data that you can analyze to gain insights and maximize business outcomes.

Common use cases IBM Cloud Pak for Data addresses include:

- Data privacy and security.
- Data quality and governance.
- Optimized data access and availability.
- ModelOps.
- AI governance.

[5] *To determine whether the service is installed, open the services catalog and determine whether the service is enabled. Refer to the Apache Spark section of the IBM Cloud Pak for Data product hub for more information.*

For a full list of IBM Cloud Pak for Data services, visit the Cloud Pak for Data product hub.

Red Hat OpenShift Data Foundation

OpenShift Data Foundation is software-defined storage that is optimized for container environments. It runs as an operator on OpenShift Container Platform to provide highly integrated and simplified persistent storage management for containers. OpenShift Data Foundation uses Ceph to provide the file, block, and object persistent storage to applications running on OpenShift Container Platform. This project employed OpenShift Data Foundation in its external-mode configuration to provide scalable object storage service as a separate storage cluster to serve Analytics Engine as well as persistent RWO and RWX volumes required for IBM CloudPak for Data.

The solution demands a variety of data persistence options for different applications and requires a flexible container storage solution. IBM Cloud Pak for Data Analytics Engine requires read-write many (RWX) volumes, read-write once (RWO) volumes, and S3 shared object storage access for Spark. Similarly the OpenShift infrastructure services, such as image registry and monitoring require RWX persistent storage. Using OpenShift Data Foundation with IBM Cloud Pak for Data Analytics Engine fulfills all of these requirements.

A Spark application on Analytics Engine typically spawns several hundreds of Spark pods (100+ pods per Spark job in our testing). These pods are distributed across OpenShift Container Platform worker nodes. The Analytics Engine requires multiple shared RWX persistent volume claims (PVCs) to serve instance, log, and application volume mounts – on each Spark pod across all worker nodes, concurrently. A container storage solution that hosts these RWX volumes must satisfy the performance demand as well as a very high provisioning/deprovisioning rate. Any delay in provisioning of PVs or performance bottlenecks at the storage layer would cause unacceptable degradation in the Spark job application life cycle. In this testing, we used OpenShift Data Foundation RWX capabilities, which delivered a high performance and high provisioning/deprovisioning rate for several hundreds of PVs across the cluster.

OpenShift Data Foundation external mode provides the flexibility needed to optimize the storage infrastructure. For instance, we recommend running multiple Ceph RADOS gateway (RGW) instances per node to achieve high-aggregated bandwidth with no additional solution cost. Higher storage bandwidth can also help analytics applications in faster processing.

Red Hat OpenShift Container Platform

OpenShift Container Platform offers a consistent hybrid cloud foundation for building and scaling containerized applications. As a trusted enterprise Kubernetes platform, it is self-managed, and includes an enterprise-grade Red Hat Enterprise Linux® CoreOS operating system, container runtime, networking, monitoring, container registry, authentication, and authorization solutions. These components are tested together for unified operations on a complete Kubernetes platform, supported both on major cloud providers and on-premise. With OpenShift Container Platform, applications and the datacenters that support them can expand from just a few machines and applications to thousands of machines that serve millions of clients.

Intel Optane SSDs and NAND media

Intel® triple layer cell (TLC) and quad layer cell (QLC) SSDs combine with Intel® Optane™ technology to accelerate the speed of frequently accessed data. A variety of Intel datacenter SSDs were used in IBM and Red Hat testing, including:

- **Intel® Optane™ SSD DC P4800X.** Intel Optane SSDs combine the attributes of memory and storage with high throughput, low latency, high quality of service (QoS), and high endurance. The Intel Optane SSD DC P4800X is designed for high write environments and can withstand intense write traffic. The life of the DC P4800X is extended with its extremely high endurance. The SSD is rated for 60 drive writes per day (DWPD), making it suitable for write-intensive applications such as online transaction processing, high-performance computing, write caching, boot, and logging.
- **Intel® Optane™ SSD P5800X.** The Intel Optane SSD P5800X is the next-gen Intel Optane media on PCIe 4.0. It has several improvements over PCIe gen 3-based Intel Optane SSD DC P4800X with 3x greater random 4K R/W IOPS, 3x higher sequential 4K-128K r/w bandwidth, 40% greater QoS for 4K random read at QD=1 and 67% higher DWPD endurance (100 DWPD).
- **Intel TLC SSDs.** The Intel® SSD DC P4610 Series is built on NVMe specification 1.2. The increased density is key to supporting broader workloads, bandwidth, and performance allowing cloud and enterprise service providers to increase users and improve data service levels. Better QoS is ensured with an intelligent firmware algorithm that keeps host and background data reads and writes at an optimum balance. The P4600 series are available in 1.6TB, 3.2TB, 6.4TB, and 7.68TB in the U.2 2.5" (15mm) form factor providing an endurance of 3 drive writes per day.
- **Intel QLC SSDs.** The Intel® SSD D5-P4320 delivers big, affordable, and reliable storage. With 33% more bits per cell than TLC, it enables 3x storage consolidation compared to hard disk drives (HDDs) delivering high capacity and lower operational cost for the capacity or "warm" data storage layer traditionally served by hybrid or HDD arrays. In addition to massive capacities, the D5-P4320 is built on the PCIe interface, which delivers higher maximum performance than the SATA interface.

Intel Xeon Scalable processors

Intel Xeon Scalable processors were used in IBM and Red Hat testing because of their scalability and performance characteristics. Intel Xeon Gold processors offer up to 22 cores/44 threads and feature Intel® Turbo Boost Technology 2.0 that ramps up to 4.4GHz. With up to four-socket scalability, these processors deliver a significant improvement in performance, advanced reliability, and hardware-enhanced security. Intel Xeon Scalable processors are optimized for demanding mainstream datacenters, multicloud compute, and network and storage workloads. The Intel Xeon Scalable processors have been designed to accelerate analytics as well as AI, providing a more scalable, agile, and efficient platform with increased security features for all enterprise use cases.

Lab configuration overview

The overall laboratory testing configuration evaluated by IBM and Red Hat engineers is depicted from a high-level perspective in Figure 2. A 10-node OpenShift Container Platform cluster was interconnected with a four-node OpenShift Data Foundation storage cluster. The OpenShift Container Platform cluster was interconnected with 100Gb Ethernet networking while the OpenShift Data Foundation cluster was connected via 80Gb networking.

Based on this and other extensive testing with OpenShift Data Foundation, Intel has released data node configurations for Red Hat OpenShift. These tested configurations offer a vastly simplified evaluation and deployment process for data services infrastructure with configurations optimized for edge, capacity, or performance workloads. OpenShift Data Foundation data nodes used in this testing are highly similar to performance-optimized data node configurations specified by Intel in terms of processor, memory, and Intel Optane SSDs DC series.

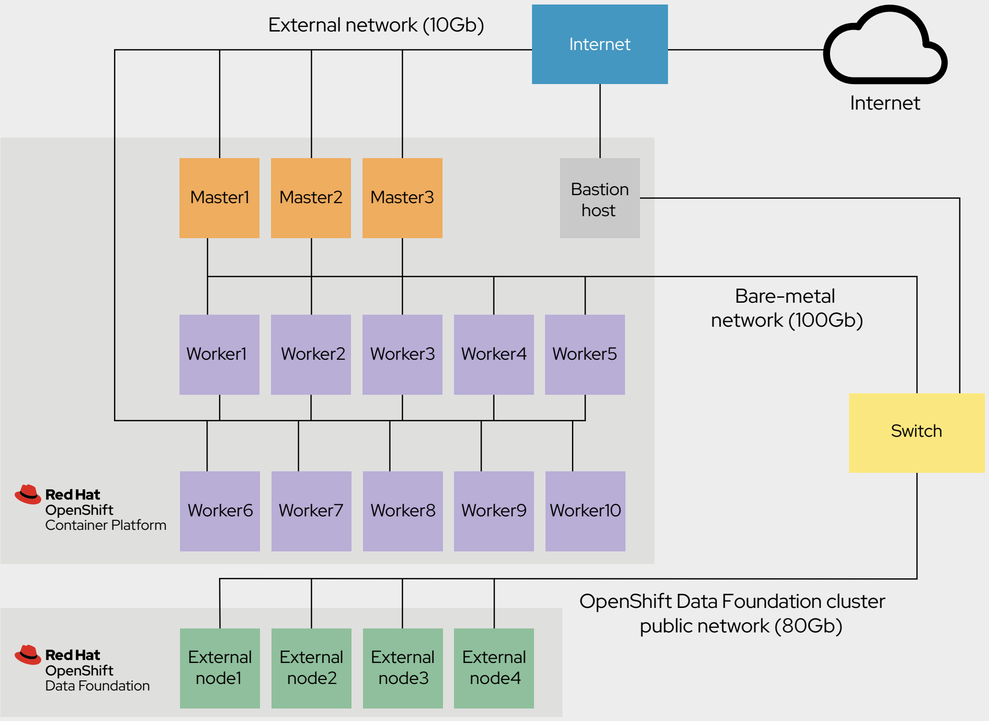


Figure 2. High-level laboratory testing configuration

In addition to evaluating OpenShift Data Foundation external mode, engineers wanted to understand the performance impact of using different media for OpenShift Data Foundation metadata and for Spark shuffle data. Engineers tested three configurations as described in Table 1. Following a scientific performance benchmarking methodology, only one variable was changed at a time throughout the testing. For example, the only difference between Config-1 and Config-2 was the media type used for the OpenShift Data Foundation metadata device. The only difference between Config-2 and Config-3 was the Spark shuffle media type.

Table 1. Tested configurations

Tested config	Worker nodes	Storage nodes	Spark local shuffle media type	Ceph metadata media type	Ceph data media Type
Config-1	10	4	1x 1.6TB Intel® SSD DC P4610 (TLC)	**Intel® SSD DC P4610 (TLC)**	Intel® SSD DC P4320 (QLC)
Config-2	10	4	1x 1.6TB Intel® SSD DC P4610 (TLC)	**Intel® Optane™ SSD DC P4800X**	Intel® SSD DC P4320 (QLC)
Config-3	10	4	**1x 1.5TB Intel® Optane™ P5800X**	Intel® Optane™ SSD DC P4800X	Intel® SSD DC P4320 (QLC)

Table 2 provides full details on the control plane, compute plane, and storage plane used in testing, including details on the three configurations.

Table 2. Configuration details

	Control plane	Compute plane	Storage plane
Resources	Helper node, bootstrap node, OpenShift Container Platform supervisor node	OpenShift Container Platform worker nodes	OpenShift Data Foundation external cluster nodes
Node count	1+3	10	4
CPU	2x Intel® Xeon® E5-2699 v4 @2.20 GHz	2x Intel® Xeon® Gold 6238R @2.2GHz	2x Intel® Xeon® Gold 6238R @2.2GHz
Memory	192GB	512GB	192GB
Intel High Performance drives (shuffle or metadata)	N/A	Config 1: 1x Intel® SSD DC P4610 Series (1.6TB, 2.5in PCIe 3.1 x4, 3D2, TLC Config 3: 1x Intel® Optane™ SSD P5800X Series 800GB, 2.5in PCIe x4, 3D XPoint™	Config 1: 4x Intel® SSD DC P4610 Series (1.6TB, 2.5in PCIe 3.1 x4, 3D2, TLC) Config 2: 2x Intel® Optane™ SSD DC P4800X Series 750GB, 2.5in PCIe x4, 3D XPoint™
Data drives	N/A	N/A	8x Intel® SSD D5-P4320 Series (7.68TB, 2.5-inch PCIe 3.1 x4, 3D2, QLC)

	Control plane	Compute plane	Storage plane
Red Hat OpenShift cluster network	1x 100Gb Ethernet	1x 100Gb Ethernet	2x 40Gb Ethernet SFP28 (bonded)
Provisioning network	1x 1Gb Ethernet	1x 1Gb Ethernet	1x 1Gb Ethernet

Table 3 lists the software elements used in testing.

Table 3. Test cluster software stack

Software stack	Compute plane	Storage plane
Operating system	Red Hat Enterprise Linux CoreOS (kernel 4.18.0)	Red Hat Enterprise Linux 8.2 (kernel 4.18.0)
Container and storage platform	OpenShift Container Platform 4.5.8	OpenShift Data Foundation 4.6 external mode
Application / software-defined storage	IBM Cloud Pak for Data v3.5	Red Hat Ceph Storage 4.1
	Stocator 1.1.3	
	Spark 3.0.1 and Spark 2.4.7	
Workload	TPC-DS , TeraGen, TeraSort, COSBench	

Design considerations

During the course of testing, engineers made a number of observations leading to design considerations and recommendations for various solution components, as described in this section.

Analytics Engine configuration

Various options were considered while designing this testing approach from an Analytics Engine perspective. Analytics Engine requires several types of volumes:

- **Log and instance volumes.** The log volume is used to persist logs from different Spark pods. The instance volume is used to persist Spark history events and share libraries required by the jobs. Analytics Engine expects a Kubernetes RWX volume for both log and instance volumes as these volumes are mounted concurrently on several of the Spark application pods without an error or delay. Testing employed an OpenShift Data Foundation external-mode cluster to provide the RWX volumes. Volumes were created with the ocs-external-storagecluster-cephfs storage class.
- **Ephemeral volume.** Analytics Engine uses this volume to persist intermediate Spark shuffle and spill data. To read/write data from/to OpenShift Data Foundation-based S3 object stores, Analytics Engine uses the Stocator driver. This driver uses the ephemeral volume for buffering the

object data before writing it to the object store. All the data stored in this volume is ephemeral in nature. If data is lost due to a node or a drive failure, Spark will automatically recompute the data on another node.

- From a performance perspective, these volumes should be local to the pod and should use fast solid state devices (e.g., NVMe). Traditionally, Spark applications run on Hadoop clusters alongside other big data applications. On these clusters a small (10% of disk size) partition from each of the disks on the node is used to store Spark intermediate data. A similar solution can be achieved in Kubernetes by mounting these smaller partitions as host paths on the Spark pod as ephemeral volumes. However, usage of host paths is discouraged on OpenShift Data Foundation clusters for security reasons. Native Kubernetes volumes are recommended instead. We considered following three solution alternatives for Kubernetes-based local volumes:
  - **OpenShift Container Platform local storage operator.** We discarded this option as it only supports static volume provisioning. As such, we would have to create fixed-size PVCs up front and keep track of which PVCs are available and which are not. Another significant issue with local volumes is that the pod bound to this volume can't be scheduled to any other node due to hard pod volume collocality constraints. In the event of a node going down, the pod won't be scheduled, which is unacceptable.
  - **Kubernetes RWO volumes.** These volumes don't meet the pod volume locality criteria.
  - **Native Kubernetes `emptyDir` volume.** Kubernetes `emptyDir` volumes are natively ephemeral in nature. Essentially the volume is around for the pod's lifetime, which exactly matches with both the locality and ephemeral requirements of the Spark application. Additionally, while mounting `emptyDir` volumes, it is also possible to specify the size of the volume allowing us to control the size of Spark ephemeral volume.

  For this testing, we chose the Kubernetes emptyDir volume as the ephemeral volume for Analytics Engine jobs. Kubernetes emptyDir volumes are backed by multiple local fast and performant NVMe TLC and Optane drives.

Considerations for Spark optimization

A number of observations and adjustments were made to optimize Apache Spark, including:

- **Adopting Apache Hive-style partitioning.** Using Hive-style partitioning while creating data and the up-front discovery of partitions significantly reduces the number of objects being read from the object store and subsequently improves query performance.
- **Using appropriate partition sizes.** While running TeraSort workload we observed that varying partition sizes from 128MB to 512MB to 1024MB caused the job to run faster due to fewer container start-up delays and lower concurrency against the object store. For example, A 10TB TeraSort job with a partition size of 1024MB ran twice as fast as a job with a partition size of 128MB.
- **Allocating appropriate hardware.** While running Spark jobs we observed that a TeraSort job ran slower when given excessive resources in terms of CPU and memory. We observed that 10TB TeraSort with 800 cores ran 30% slower when compared to one equipped with only 320 cores.

Storage for data under analysis

Before Analytics Engine jobs can analyze data, it needs to be stored in an object store that supports the S3 API, a Hadoop Distributed File System (HDFS) cluster, or even on a persistent volume mounted on the Spark pods. Consistent with the philosophy of separating compute and storage, engineers wanted to keep data external to the compute cluster, so that both the compute and storage tiers could be scaled independently. With external storage, Analytics Engine jobs run on a separate OpenShift Container Platform cluster. Other systems in the enterprise can access the same external OpenShift Data Foundation external mode cluster for their data needs.

As described, an OpenShift Data Foundation external-mode configuration was chosen as the storage cluster. S3 storage provided by OpenShift Data Foundation was used to store the data for test workloads and RWX persistent volumes were used to store metadata for various Analytics Engine and IBM Cloud Pak for Data control plane components. Other things to consider from an external object storage cluster perspective include:

- **Object store connector.** By default (and for this testing) Analytics Engine is configured to use the Stocator connector to read and write data objects from and to OpenShift Data Foundation S3 buckets. The Stocator connector is custom built with Spark object access patterns in mind. It outperformed other open source S3 connectors on the object write path.
- **Scalable connectivity.** Big data analytics workloads require high bandwidth access to object storage clusters. For good performance, scalable connectivity must exist to the external OpenShift Data Foundation cluster. Additional considerations are provided in the OpenShift Data Foundation external mode configuration section below.

Image repository

Continuing the philosophy of using OpenShift Container Platform as a pure stateless compute cluster, engineers configured the cluster image registry to use an external OpenShift Data Foundation PVC to store image layers. All Analytics Engine and IBM Cloud Pak for Data control-plane and data-plane images were served by this image registry. Details are provided in the OpenShift Data Foundation external mode configuration section below.

Intel SSD NVMe considerations

Slower storage can easily become a performance bottleneck. Garbage collection for NAND SSDs requires background writes. NAND SSDs also read and write in pages, which get stale. Current pages are copied (written) to an empty block to free space. Obsolete and copied pages are erased to create an empty block leading to slower, less consistent responses. In contrast, Intel Optane DC SSDs do not use garbage collection. Instead, Intel Optane media is byte addressable with write-in-place updates, delivering a consistent, fast response and QoS, lower latency, greater I/O operations per second (IOPS), and higher endurance.[6]

Intel Optane media allows for a much larger number of lifetime write cycles, and thus a higher endurance than NAND memory. Moreover, because Intel Optane media enables write-in-place updates, it avoids the extra writes that NAND-based systems incur, at the cost of endurance. For these reasons, Intel Optane DC SSDs feature much higher endurance than NAND SSDs. High endurance storage is also cost effective, as shown in Figure 3.

[6] *Endurance refers to the number of times the memory capacity can be written before it is considered worn out and unsuitable for continued reliable use.*

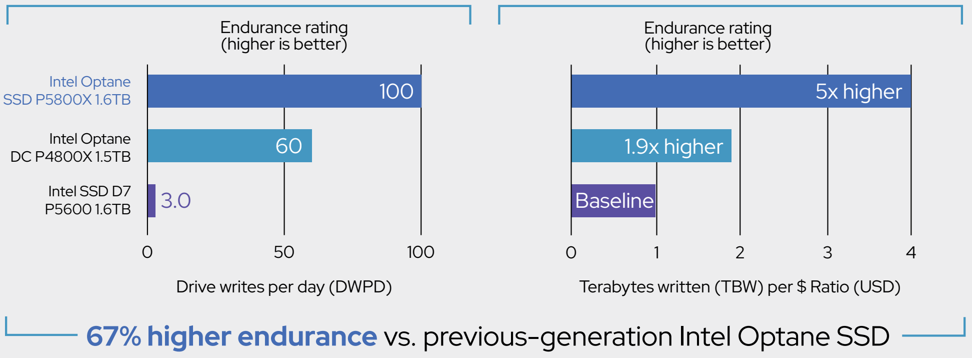


Figure 3. Intel Optane DC SSDs offer high performance, endurance, and are cost effective

In the tested configurations, Intel Optane SSD 5800X were used to accelerate the write-intensive Spark shuffle operation on OpenShift Container Platform worker nodes. Intel Optane DC SSD 4800X were used and are recommended for metadata storage on OpenShift Data Foundation nodes. This approach provides quicker access to metadata to accelerate I/O. In addition, the high endurance of the Intel Optane DC SSDs can absorb writes and extend the life of the backend NAND devices used for Ceph data. For example:

- A 500GB NAND drive warrantied for three drive writes per day (DWPD) over five years supports a maximum of 3PB of total writes over the five-year term.
- A 500GB Intel Optane DC SSD is warrantied at 60 DWPD for five years, supporting a maximum of 55PBs of total writes over the five-year term.

The Intel Optane SSD DC P4800X maintains consistent read-response times regardless of the write throughput applied to the drive. Figure 4 shows the dramatically lower latency of an Intel Optane SSD DC P4800X compared with an Intel® 3D NAND SSD, especially under the pressure of increasing random-write operations.[7] Unlike NAND-based SSDs, the latency of Intel Optane SSDs remains consistently low for all write requests.

[7] *Intel Optane SSD P5800X can deliver more than twice the IOPS of Intel Optane DC SSD P4800X, yielding still better endurance per dollar.*

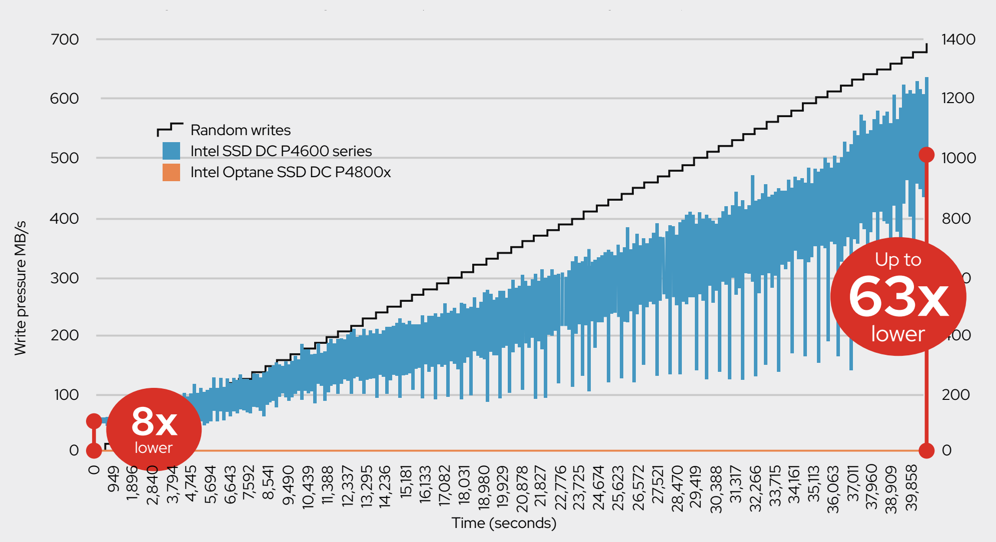


Figure 4. Intel Optane SSDs exhibit very low write latency

Over-provisioning is another important consideration. While over-provisioning SSDs sacrifices user-addressable capacity, it can provide a positive effect on NAND-based SSD endurance, write amplification factor (WAF), and random write performance. In general, over-provisioning allows flexibility for endurance and capacity when using NAND-based SSDs. Intel Optane DC SSDs are still a better option when higher endurance is required for a hot data tier, caching, or DRAM displacement. Due to the nature of the media, Intel Optane DC SSDs do not gain benefits with over provisioning as they already exhibit very high endurance.

On the OpenShift Container Platform compute nodes, Intel Optane SSD 5800X were used for sort, merge, and shuffle activity. The shuffle drives experience a mix of heavy read/write cycles depending on load, sort-merge, and shuffle activity. The read/write pattern also varies with different shufflers. Given varying block size and queue depth, Intel Optane helps accelerate application performance through low latency, high endurance, greater IOPS.

OpenShift Data Foundation external mode configuration

For IBM and Red Hat testing, OpenShift Data Foundation was configured in external mode, allowing the use of a storage cluster that is external and independent from OpenShift Container Platform.[8] Depending upon the workload, OpenShift Data Foundation external mode provides deployment flexibility in configuring resources for Ceph components. Table 4 defines the resources that were configured for Ceph components within OpenShift Data Foundation during testing.

[8] *See the Red Hat documentation for detailed instructions on setting up OpenShift Data Foundation in external mode.*

Table 4. Ceph component configurations

	CPU	Memory	Instances	Data device	Metadata device
Ceph MON	Default		3	N/A	
Ceph OSD	5	4GB	64	3.4TB	365GB
Ceph RGW	8	4GB	16	N/A	
	Default		3	N/A	
	Default		2	N/A	

By default, external mode OpenShift Data Foundation clusters are configured with a single RGW instance. Because big data analytics workloads require high bandwidth access to object storage clusters, multiple RGW instances need to be configured on the Red Hat OpenShift nodes. As shown in Figure 5 engineers made use of Kubernetes service and endpoint objects to bind to multiple RGW instances. A total of 16 Ceph RGW instances were bound under one Kubernetes endpoint, which was then used by one Kubernetes service.

In addition to delivering far better performance over the default single RGW configuration, there are several benefits of this configuration approach. Analytics Engine (Spark) only needs to be aware of a single object storage service endpoint, that then transparently resolves to multiple Ceph RGW instances. This abstraction also provides high availability of the object storage service.

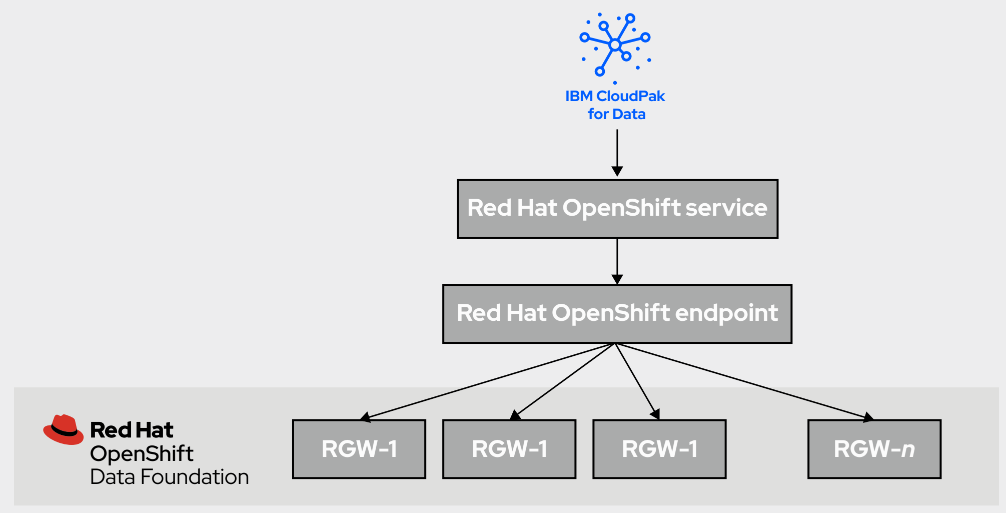

IBM CloudPak for Data
Red Hat OpenShift service
Red Hat OpenShift endpoint
Red Hat OpenShift Data Foundation
RGW-1
RGW-1
RGW-1
RGW-*n*

Figure 5. Multiple Ceph RGW instances bind to a single Kubernetes (OpenShift Container Platform) endpoint

Red Hat OpenShift Container Platform considerations

Analytics Engine benefits from fast local storage used as a spill/shuffle device for Spark. As such, OpenShift Container Platform nodes were configured to consume a fast local device as container ephemeral storage for Spark shuffle. NVMe media is recommended for Spark shuffle space. For OpenShift Container Platform nodes to use additional NVMe devices as shuffle devices, we needed to create a custom OpenShift machine config and define how the device should be mounted on each worker node.

To learn how to apply custom machine configuration in your environment, refer to the OpenShift Container Platform environment documentation on post-installation machine configuration tasks and managing nodes.

Workload-specific performance results

To gauge performance, engineers chose standard benchmark workloads that are a good representation of actual use cases. TeraGen/TeraSort tests were first run across the three storage server hardware configurations defined in Table 1 (Config-1, Config-2, and Config-3). The best-performing configuration was then used to run TPC-DS and COSBench benchmarks.

TeraGen/TeraSort

Engineers used TeraSort to simulate Spark applications with large shuffle stages. As one of the most popular MapReduce-based benchmarking programs, TeraSort samples the input data and sorts that data into a total order. Generally, TeraSort is used to sort the data set of desired size generated by TeraGen. In our testing, TeraSort was run across 1TB, 2TB, 4TB, and 10TB scale factors to simulate Spark applications with different sizes of shuffle data. TeraSort was also run concurrently to simulate Spark applications running with varied size shuffle stages.

TeraGen

Figure 6 compares TeraGen performance across the three test configurations. The team made the following observations from these results:

- Between Config-1 and Config-2 there was a consistent 16-17% improvement in TeraGen execution time. This is a result of the OpenShift Data Foundation write operations on Config-2 becoming significantly faster by using the Intel Optane NVMe drives for Ceph metadata.
- Between Config-2 and Config-3, engineers swapped out an Intel TLC-based Spark intermediate drive for a faster Intel Optane SSD. They didn't observe much of a performance difference until the 4TB scale factor. For 10TB they observed a little dip in performance.
- No errors were observed in the Spark logs while generating data.

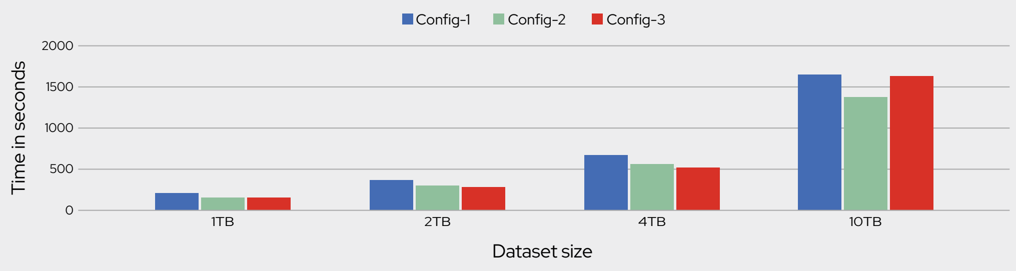


Figure 6. TeraGen performance across all three test configurations (smaller is better)

To appreciate the graph completely, it is important to understand some aspects of Spark TeraGen applications. The TeraGen application generates a data set of a specific size. The data set is then persistent and can later be sorted using the TeraSort application. The generated data set consists of several objects (files) of the same specific size. The TeraGen Spark application runs one job with two stages:

- In the first stage (Stage#0), all the executors generate objects of the specified size and write them to the specified OpenShift Data Foundation bucket.
- In the second stage (Stage#1), the job verifies if the required amount of data has been generated or not.

As mentioned previously, the team used the Stocator connector to write/read objects to/from Ceph S3 buckets. Stocator buffers the objects on the ephemeral volumes before writing them to OpenShift Data Foundation. As a result, TeraGen performance depends on the drives where Stocator buffers the intermediate data as well as the throughput of OpenShift Data Foundation object operations. Table 5 describes the Spark configuration and the object size being used across all the three tested configurations.

Table 5. Scale factors and executor configurations

Scale factor	Executor configuration	Number of executors	Total cores	Total memory	Object size
1TB	8 cores, 40GB	100	800	4000GB	512MB
2TB	8 cores, 40GB	100	800	4000GB	512MB
3TB	8 cores, 40GB	100	800	4000GB	512MB
10TB	8 cores, 40GB	100	800	4000GB	1024MB

After trying sample jobs on several executor configurations (e.g., 4 cores/32GB, 5 cores/25GB, 8 cores/40GB), engineers settled on using executors with 8 cores and 40GB. This configuration delivered the best performance and maximum utilization of the hardware available across the compute cluster. With executors configured for 8 cores and 40GB, 10 executors fit on each OpenShift Container Platform node, allowing scalability up to 100 Spark executors in the 10-node cluster. The same executor configuration was used across both TeraGen and TeraSort.

TeraSort

Figure 7 compares TeraSort performance across the three test configurations. The team made the following observations:

- There were no errors in any Spark events/logs during the TeraSort runs.
- There was no observable change across the three hardware configurations. At higher scale factors (e.g., 4TB and 10TB) runs on Config-3 were slightly faster than those on Config-1 and Config-2. In Config-3, Spark intermediate data volumes are based on Intel Optane SSD NVMe drives.
- Disk utilization was 30% less on Intel Optane drives when compared to TLC-based drives in both Config-1 and Config-2. For exact disk utilization refer to the Appendix section on shuffle device utilization.

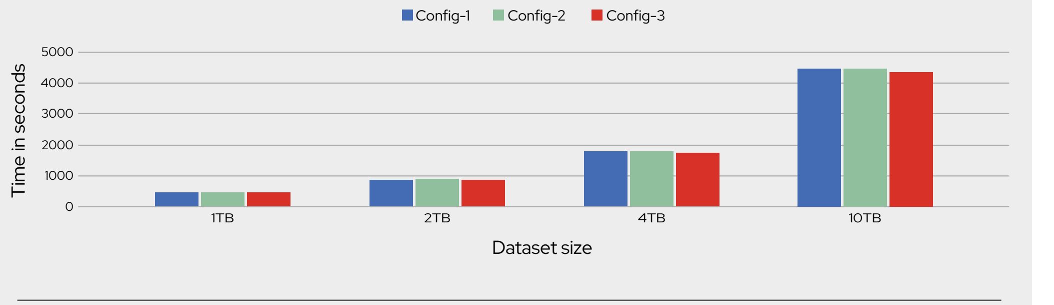


Figure 7. TeraSort performance across the three test configurations (lower is better)

Figure 8 compares two TeraSort runs on Config-3 across scale factors of 1TB, 2TB, 4TB, and 10TB. In this chart, the red bars represent runs that were given all of the resources of the OpenShift Container Platform cluster (800 cores, 100 executors of 8 cores each). The lavender bars represent runs that were given a more appropriate amount of resources (320 cores, 40 executors of 8 cores each). Running with right-sized resources clearly resulted in better performance. Engineers made the following observations:

- Up to the 2TB scale factor, TeraSort jobs ran ~20% faster. Beyond that point they ran ~30% faster with appropriate parallelism.
- From Spark events in the shuffle stage, engineers observed that significant task time was spent waiting for remote shuffle blocks to be served. At the 10TB scale factor, the average **Shuffle Read Blocked Time** for the yellow run was 25 seconds vs. 6ms for the faster red run with more appropriate resources.

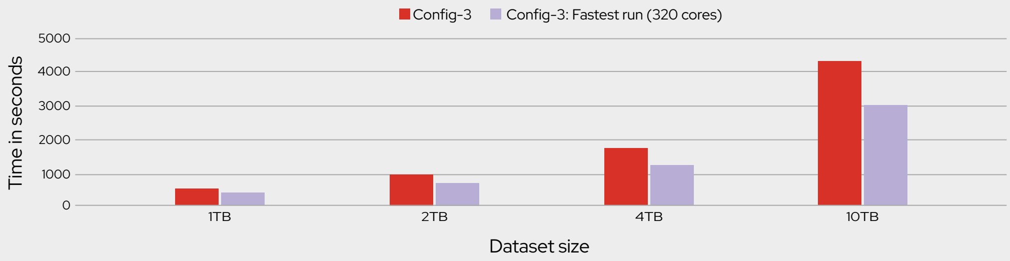


Figure 8. Right-sizing cluster resources improves performance for TeraSort jobs (lower is better)

TPC Benchmark Decision Support (TPC-DS)

The Transaction Processing Performance Council (TPC) offers benchmarks that provide relevant and objective performance data to industry users. TPC-DS is a decision support benchmark that models several generally applicable aspects of a decision support system, including queries and data maintenance. These results are derived for a given hardware, operating system, and data processing system configuration under a controlled, complex, multiuser decision support workload.

To simulate Spark SQL workloads, the team modified TPC-DS v2.4, enhancing it to test dynamic data skipping. All 104 TPC-DS tests were across 1TB, 3TB and 10TB scale factors. In a typical data warehouse environment, different streams might represent different users. Since Analytics Engine is not a multiuser system, the team ran multiple Spark applications against the same data set. All 104 TPC-DS tests were run with queries randomised across streams to simulate multiple concurrent users running different query types (e.g., filter, aggregation, groupby). Testing used up to five streams to simulate five concurrent users issuing different sets of queries.

All the TPC-DS tests were run on Config-3 described in the test configuration overview section, with test configuration details as follows:

- **Data generation.** The dsdgen tool from the Spark application was used to generate TPC-DS data across all the chosen scale factors. All the generated data (objects) were stored in the OpenShift Data Foundation cluster.
- **Data layout.** During data generation, we partitioned the following tables using Apache Hive-style partitioning: web_returns, catalog_returns, store_returns, web_sales, catalog_sales, and store_sales. Data (records) in each partition were further range-partitioned and divided into objects of 32MB each.
- **Statistics collection.** The team used an external Postgres database to store metadata. We ran a Spark application up front to create the Hive metastore schema, discover the partitions of partitioned tables, and collect the table and column level statistics. All these statistics are populated to the external Postgres metastore. Statistics were collected up front across all the scale factors. We ran all the TPC-DS queries by pointing to the pre-populated Hive metastore. This upfront discovery of partitions drastically improves the performance of queries.

Analytics scale and concurrency

The jointly architected analytics solution successfully completed all 104 queries of the TPC-DS benchmarking suite without any failures across different scale factors of up to 10TB. The execution times for all 104 TPC-DS queries across 1TB, 3TB, and 10TB scale factors are shown in Figure 9, along with the number of CPU cores that were used for each.

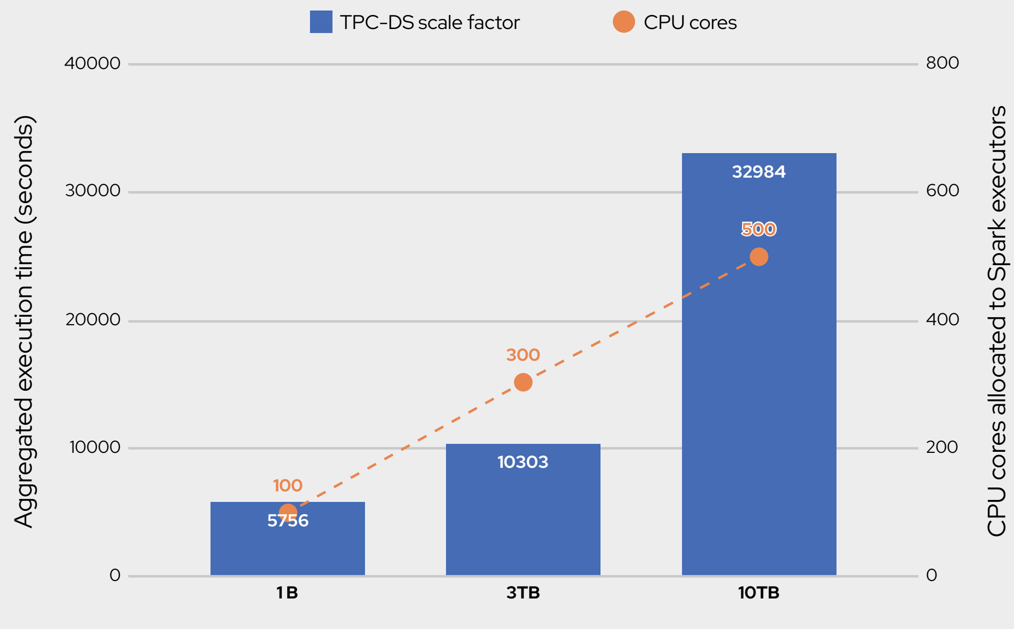


Figure 9. Aggregated query execution times for 104 TPC-DS queries with the number of CPU cores used for each

The team was able to run all the 104 TPC-DS queries across 1TB, 3TB and 10TB scale factors, including the more demanding TPC-DS queries such as q14a, q14b, q23a, q23b, q95 and the most demanding q67 with large skewed partitions. All of the queries were run on Spark 3.0.1 with adaptive query execution(AQE) and adaptive partition coalesce on.

A number of adjustments were made:

- At 10TB scale, we observed that several queries were failing or retrying during the object listing stage. To overcome this issue the value of spark.sql.hive.filesourcePartitionFileCacheSize was increased to 10GB.
- At 10TB scale, we observed several queries stages were either failing or retrying with Spark default memory management distribution. We adjusted the executor memory distribution by tuning the following two configurations: spark.memory.fraction was adjusted to 0.8 and spark.memory.storageFraction was adjusted to 0.2.

- To smoothly shuffle data across 100 executors, we had to increase the various network timeouts, shuffle, and rpc threads pool. Table 6 lists configurations with tuned values. Table 7 lists Spark executor sizes and numbers of executors.

Table 6. Tuned configuration values

Configuration	Value
spark.network.timeout	3600s
spark.shuffle.io.maxRetries	10
spark.shuffle.io.retryWait	60s
spark.shuffle.io.serverThreads	64
spark.shuffle.io.clientThreads	64
spark.shuffle.io.threads	64
spark.rpc.io.serverThreads	64
spark.rpc.io.clientThreads	64
spark.rpc.io.threads	64

Table 7. Spark executor sizes and number of executors

Scale factor	Executor configuration	Number of executors	Total cores	Total memory
1TB	5 cores 40GB	20	100	800GB
3TB	5 cores 40GB	60	300	2400GB
10TB	5 cores 40GB	100	500	4000GB

Figure 10 compares a single-stream run with a multistream simulation using five concurrent users exercising all 104 TPC-DS on a 1TB TPC-DS data set. This study was done to compare how the system reacts to a single user running all 104 TPC-DS queries with five concurrent users running all 104 TPC-DS queries in a randomized manner. Both the single stream and multistream runs were conducted with 100 cores. Engineers found that with five concurrent users (five times the load) the system degraded by only approximately 15%. This very slight performance decrease was mostly attributed to contention on shared resources like shuffle drives and networking.

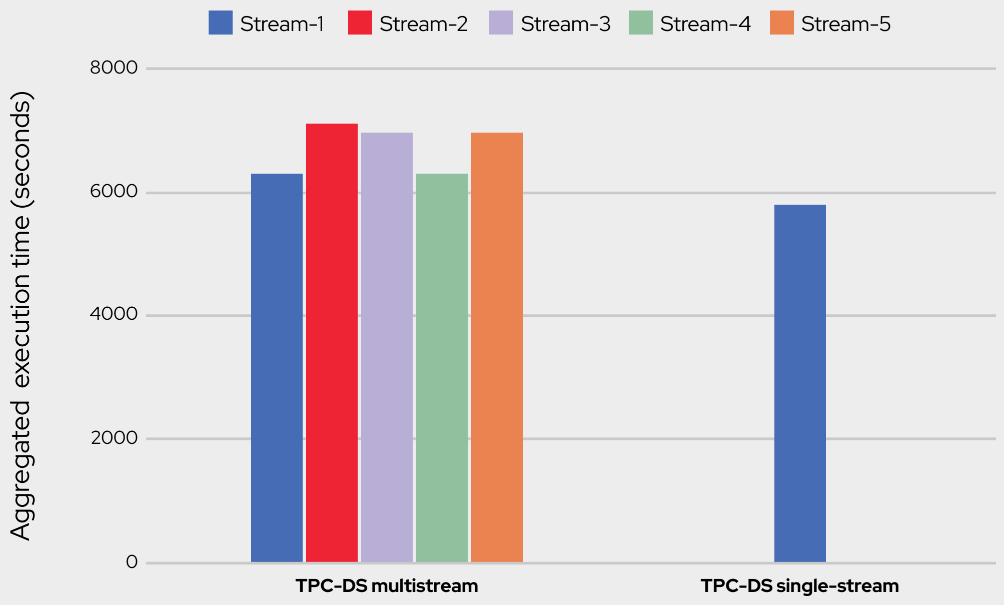


Figure 10. Multistream performance only introduced a slight ~15% performance penalty over single-stream performance (lower is better)

Performance compared to previous results

Comparing previous results can be helpful to verify forward progress relative to performance, even if the comparisons are imperfect due to differing hardware configurations and software versions. Specifically, the most recent testing demonstrated:

- Analytics Engine using Intel Optane for Spark shuffle delivered 2x higher performance (Figure 11) when compared to an alternative query engine, with both running on OpenShift Container Platform with OpenShift Data Foundation.
- Analytics Engine powered by Apache Spark with optimized data layout and partitioning and workload-specific tuning delivered a 3x performance increase for 1 TB TPC-DS benchmark over an unoptimized configuration of Apache Spark on a previous version of IBM Cloud Pak for Data.

Figure 11 illustrates the up to 2x performance increase found using Analytics Engine powered by Apache Spark in IBM Cloud Pak for Data v3.5 with OpenShift Data Foundation and Intel Optane SSD NVMe devices.[9] These results were compared across 54 I/O-intensive TPC-DS queries[10] at both 1TB and 3TB scale factors.

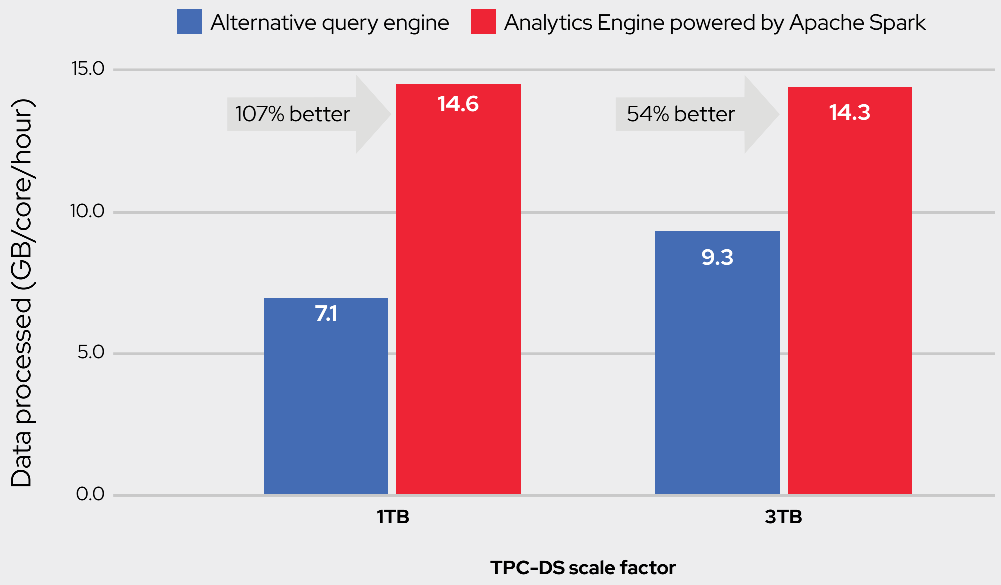


Figure 11. Analytics Engine powered by Apache Spark provided twice the throughput per core per hour as compared with an alternative query engine (higher is better).

Figure 12 shows that IBM Cloud Pak for Data 3.5 benchmarked in this project delivered up to 3x higher data processing performance when compared against the previous version (IBM Cloud Pak for Data 3.0) across all 104 TPC-DS queries at the 1TB scale factor.

[9] *The alternative query engine 1TB and 3TB runs were done using 320 cores whereas the Spark 1TB and 3TB runs were done using 100 and 300 cores respectively. To treat the results equitably, we compare the throughputs vs. the absolute time taken to run the queries. Throughput, or the amount of data processed per core in an hour was calculated using the following formula: (data-size/cores/time-taken-in-seconds)*3600. While the 1TB TPC-DS results for both analytics engines were similar, the 3TB Spark throughput result was 55% higher.*

[10] *As part of another study, Red Hat identified the following 54 i/o intensive TPC-DS queries: q3 ,q7 ,q12 ,q13 ,q15 ,q17 ,q19 ,q20 ,q21 ,q24 ,q25 ,q26 ,q27 ,q28 ,q29 ,q31 ,q32 ,q34 ,q39 ,q40 ,q42 ,q43 ,q45 ,q46 ,q48 ,q49 ,q51 ,q52 ,q55 ,q56 ,q60 ,q63 ,q64 ,q65 ,q66 ,q68 ,q71 ,q73 ,q75 ,q76 ,q79 ,q82 ,q83 ,q84 ,q85 ,q87 ,q88 ,q89 ,q90 ,q91 ,q92 ,q93 ,q96 and q97.*

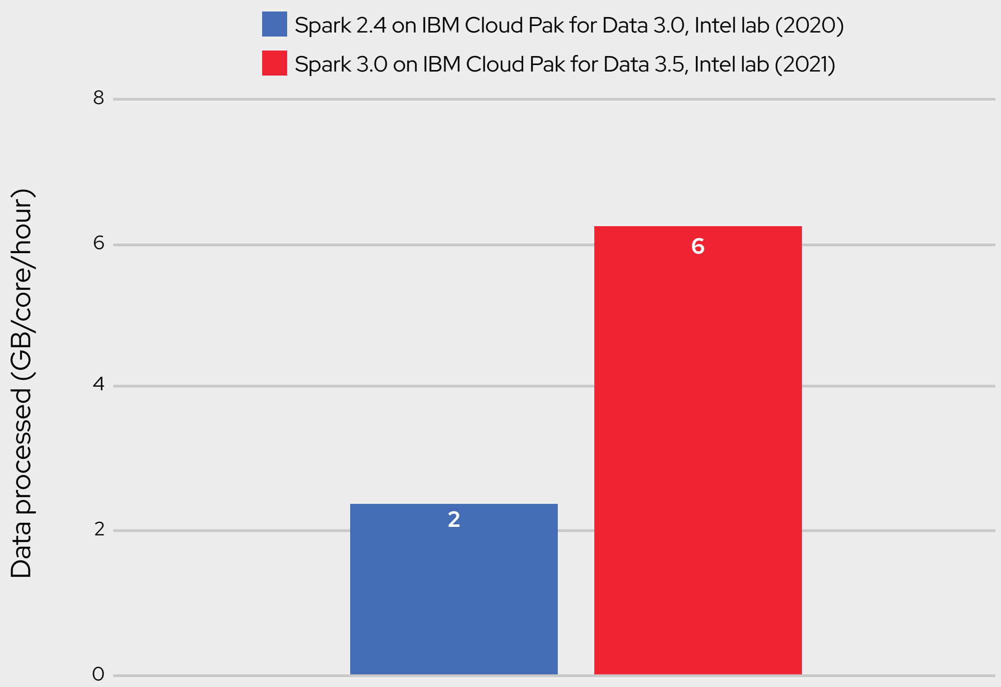


Figure 12. A tuned Spark 3.0 on IBM Cloud Pak for Data 3.5 provided up to 3x higher performance for 1 TB TPC-DS workload than an earlier untuned test run on Spark 2.4 on IBM Cloud Pak for Data 3.0 (higher is better)

Performance improvements with dynamic data skipping

Data skipping can often significantly boost the performance of queries by skipping over irrelevant data objects or files based on summary metadata associated with each object. Figure 13 shows query performance gain achieved using Spark 3.0.1 with Dynamic data skipping over unmodified Spark 3.0.1. With the dynamic data-skipping feature enabled, Analytics Engine showed a significant performance improvement of up to 70% for several TPC-DS queries.

The team observed 20+ TPC-DS queries with better performance with dynamic data skipping enabled. While the chart only captures queries that showed a performance improvement of 5% and above, fully nine out of 20 queries showed more than 40% improvement. Query-72 demonstrated a maximum performance improvement of 72%.

As described, the data was partitioned in this testing. Partitions were discovered and other statistics about the data were collected up front. Using these statistics, community Spark itself could skip many objects (data). As a result, we didn't observe many objects being skipped by data skipping alone. Most of the performance improvement that we observed is due to an IBM add-on dynamic data-skipping capability used in conjunction with the xskipper open-source data-skipping library.

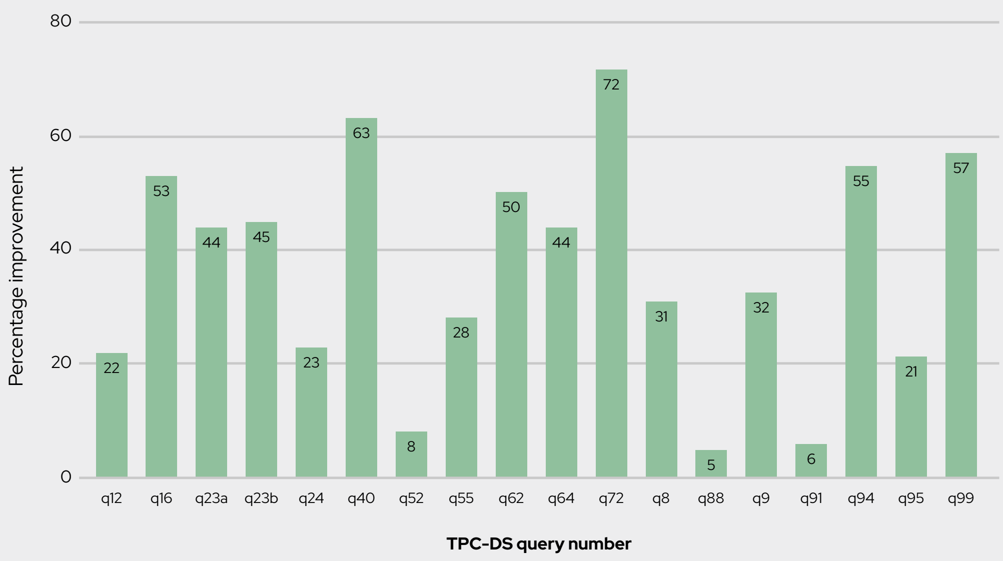


Figure 13. Dynamic data skipping provides performance improvements for many TPC-DS queries.

Achieving optimal price performance

Right-sizing IBM Cloud Pak for Data Spark clusters is important to optimize the price/performance ratio and keep the overall cost of the analytics service low. Empirical evidence has shown that oversized Spark clusters do not necessarily provide a better price/performance ratio. IBM and Red Hat testing revealed that for TPC-DS workloads, allocating four times more resources to the Spark cluster delivered only a 6% higher performance—diminishing CAPEX returns. Figure 10 compares 1TB TPC-DS execution times across different numbers of cores along with relative hardware cost.

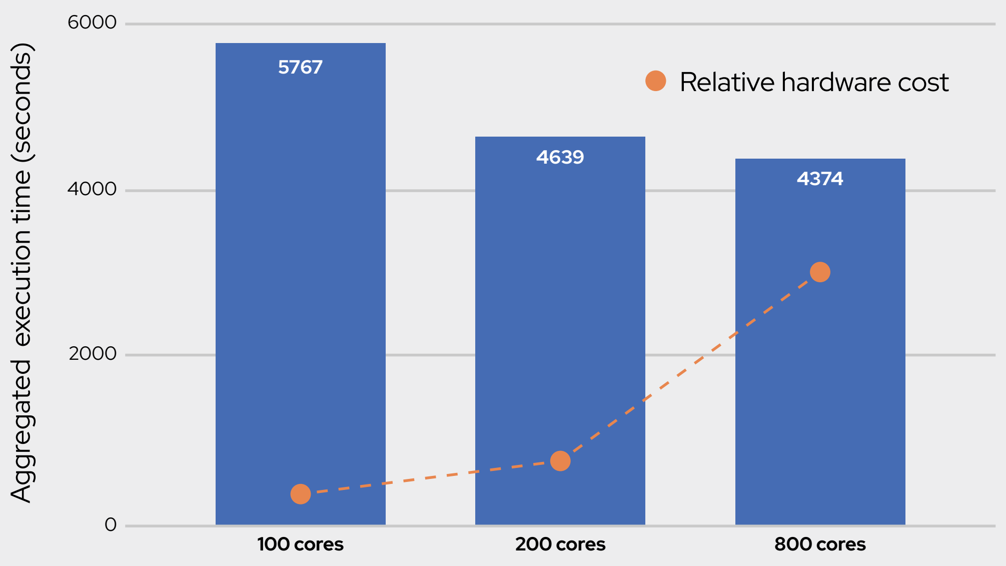


Figure 14. Relative hardware cost vs. aggregated execution time (1TB scale factor) for 104 TPC-DS queries across different numbers of cores.

As shown in the chart, a 1TB scale factor TPC-DS run with 800 cores ran only 25% faster when compared to the run with 100 cores. By increasing the hardware by a factor of eight, execution time was only reduced by 25%. In contrast, a 1TB TPC-DS run with 200 cores ran 20% faster when compared to the run with 100 cores. So by doubling spending on hardware the execution time was reduced by 20%, which is much more economical. In our lab, a 200 core configuration seems to be the optimal amount of hardware to run a 1TB scale factor TPC-DS Spark application.

COSBench

COSBench is a benchmarking tool used to measure the performance of cloud object storage services. Engineers used COSBench to measure the high-watermark performance of the Ceph S3 object access interface provided in OpenShift Data Foundation. COSBench was run against both the OpenShift Container Platform and OpenShift Data Foundation clusters to expose any hardware, software, and configuration related issues, before we began exercising the analytics workloads. For detailed COSBench performance results, refer to the Appendix.

To understand the high-watermark performance of the cluster, engineers ran the COSBench workload with a large object size of 128M and upto 80 COSBench Driver Instances to simulate concurrent multiuser workload. A four-node OpenShift Data Foundation external-mode cluster was used with BlueStore metadata deployed on different solid state media.

- During the first iteration, BlueStore metadata was configured on Intel SSD DC P4610 Series (TLC) media.
- During the second iteration, Intel Optane SSD DC P4800x media type was used to store BlueStore metadata.
- The Ceph OSD data partition was based on Intel SSD DC P4320 (TLC).

As shown in Figure 15, large-object 100% HTTP PUT workload testing exhibited near linear scalability when incrementing the number of client instances. The highest observed PUT throughput for 128MB objects size was found to be 11.9GBps before encountering bottlenecks. As is clear from the test data, Intel Optane SSD DC P4800x media delivered up to twice the performance compared to Intel SSD DC P4610 media. When configured to be used as BlueStore metadata significantly improves with write performance as it sits in the write path of the IO operation.

The cluster aggregated throughput was found to be constrained by the network on the OpenShift Data Foundation nodes. The team believes that they could have achieved even higher throughput with more available network bandwidth between the OpenShift Container Platform nodes and the OpenShift Data Foundation nodes. The Intel SSD DC P4320, Intel SSD DC P4610 and Intel Optane SSD DC P4800X and other system resources had sufficient headroom for additional performance.

Engineers also ran a similar test with a GET workload and as expected, there is no observed performance difference between the two media types. Please refer to the Appendix for GET performance results.

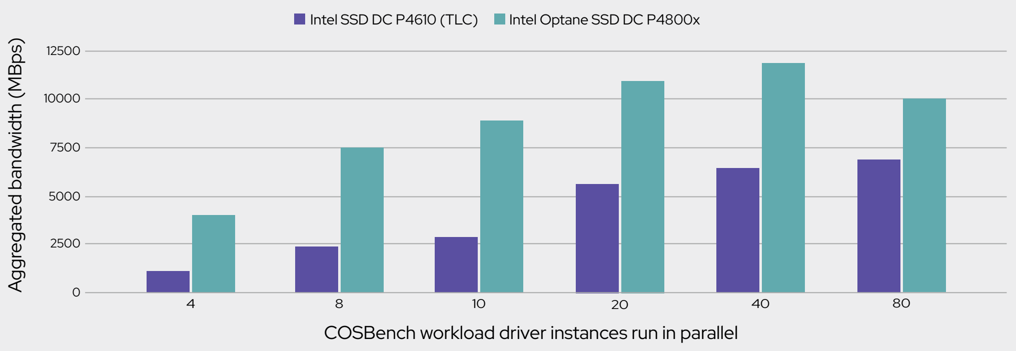


Figure 15. COSBench PUT workloads scale linearly with Intel Optane SSD DC P4800X providing significant performance advantages for Ceph BlueStore metadata

Summary

Red Hat Data Services provides an ideal platform for Analytics Engine powered by Apache Spark. In testing performed by IBM and Red Hat, engineers showed that OpenShift Container Platform and OpenShift Data Foundation deliver analytics performance at scale, supporting TPC-DS runs at 1TB, 3TB, and 10TB scale factors without errors. Multistream runs were also easily supported with very low performance degradation while running multiple streams. TPC-DS results compared favorably with previous testing delivering the best reported analytics performance with Red Hat OpenShift Container Platform and Red Hat OpenShift Data Foundation to date.

Intel technology is a vital component of this solution. Insights from this testing and others has led to verified external data node configurations for OpenShift Container Platform. These tested configurations are fast to deploy, and ease the transition to providing scalable data services for OpenShift Container Platform applications. Testing also validated that Intel Optane SSDs are ideal as underlying storage media—both for write-intensive Spark shuffle operations and as a storage cache for OpenShift Data Foundation—due to both their high performance and their extremely high write endurance. Intel Xeon Scalable Processors are also ideal for heavy analytics infrastructure, handling concurrent workloads and delivering fast execution times.

Appendix

Environment configuration and benchmarking workload files

All of the environment/cluster configuration and benchmarking workload files are available at the following Github repository: https://github.com/red-hat-data-services/ibm-ae-ocs-refarch.git

COSBench : GET performance

Unlike the PUT workload, the COSBench GET workload did not show any performance difference when Ceph is configured with Intel SSD DC TLC or Intel Optane SSD-based BlueStore metadata devices (Figure 16). This is understandable because all of the GET operations in OpenShift Data Foundation are serviced from the OSD data device. The BlueStore metadata device does not enter the read path of OpenShift Data Foundation. As a result, there is no observable performance difference across the BlueStore metadata devices with GET workload.

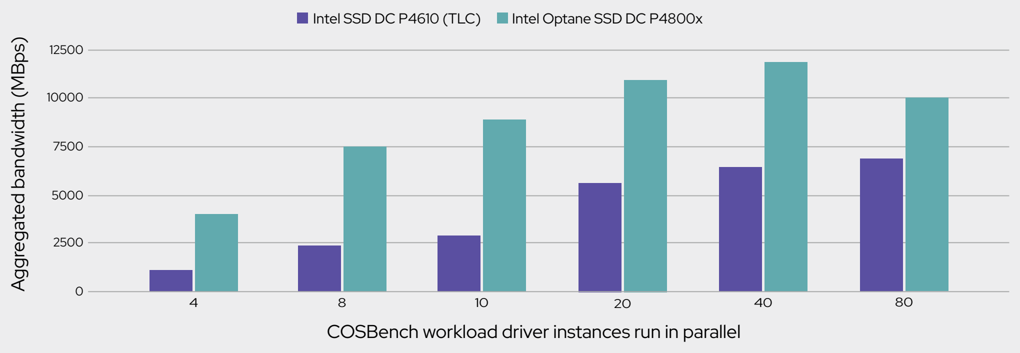


Figure 16. Metadata storage has no impact on GET workload performance

Shuffle device utilization

The Spark shuffle workload generated significantly lower device utilization for Intel Optane SSD P5800x (averaging 30%, Figure 17) than with TLC media (Intel SSD DC P4610) which averaged 60% utilization (Figure 18).

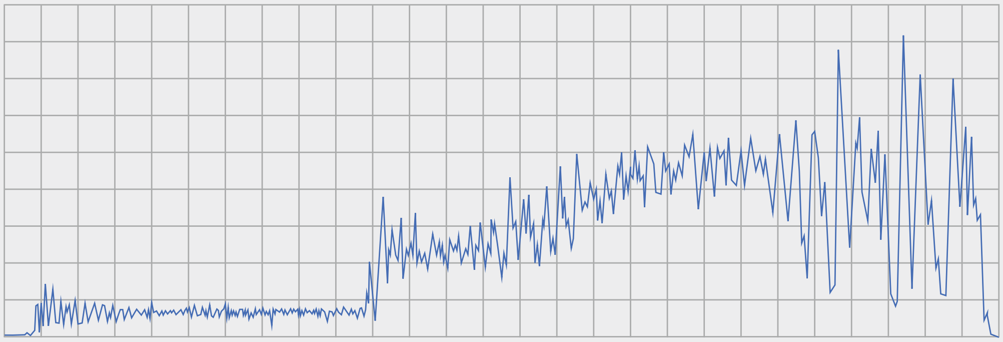


Figure 17. Intel Optane SSD device utilization for Spark shuffle operations, averaging ~30%

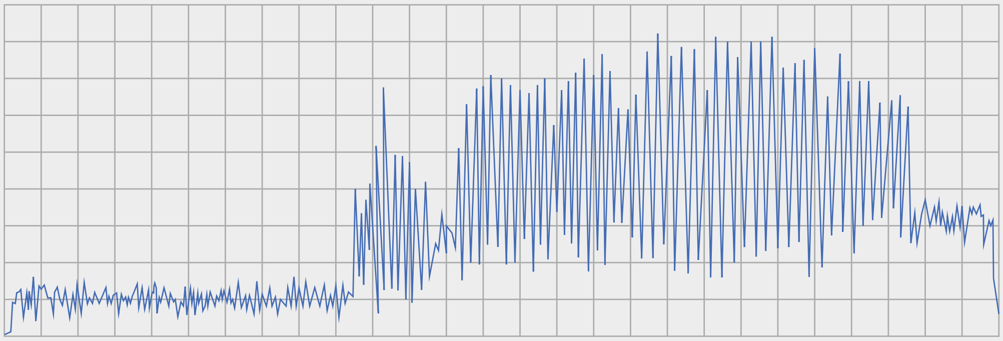


Figure 18. Intel SSD with TLC technology device utilization for Spark shuffle operations, averaging ~60%

Previous Analytics Engine testing environment

As cited for performance comparisons above, previous benchmarking of Analytics Engine included the following details:

- **Compute cluster.** Spark applications used 375 cores from an OpenShift Container Platform 4.3.1 cluster.
- **Object store.** Ceph Mimic 13.2.10 storage cluster with 14 nodes.
- **Software stack.** Cloud pack for Data 3.0 , Spark 2.4, and the Stocator driver were used to read/write objects from/to object stores.
- **Data layout and format:** Data was in parquet format. Data was not partitioned nor was it placed in any optimized layout.

Optimal RGW instance count per node

The Ceph RGW component that provides object storage interface access is stateless and scalable. Engineers wanted to understand an optimal RWG instance count per node that results in the best performance as it pertains to this testing. Multiple iterations of the TeraSort test were run across different dataset sizes and with different Ceph RGW instance counts per node. As shown in Figure 19, we have not observed a significant performance difference between four RGW instances per node as compared to two RGW instances per node, demonstrating diminishing returns. As such we recommend running two RGW instances per OSD node to achieve maximum aggregated throughput from the Ceph object storage service.

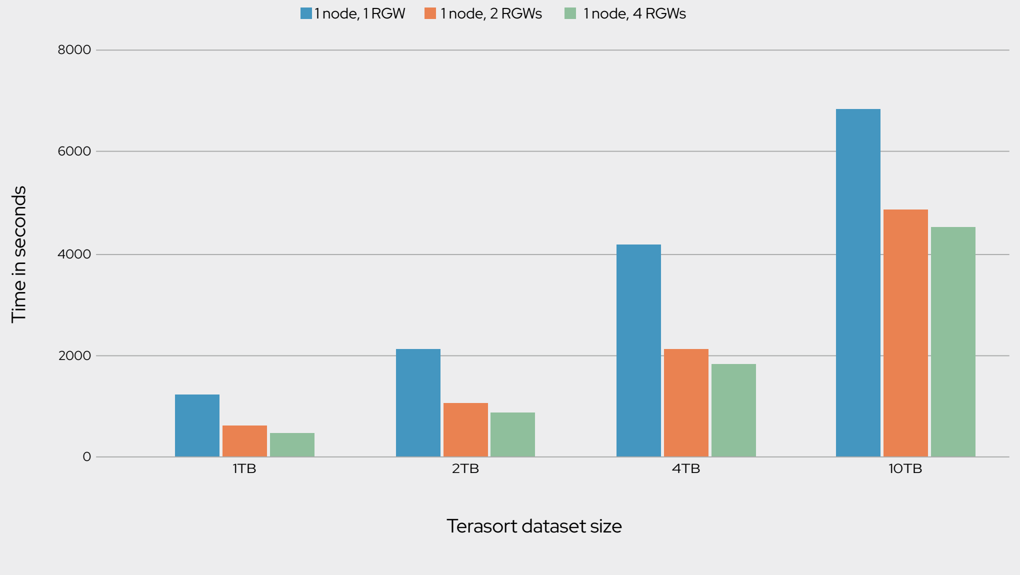


Figure 19. Comparing different numbers of RGW instances per node for the TeraSort test (lower is better)

Hardware configuration

Host	CPU	Server board
Master001-003	Intel® Xeon® Gold 6140 Processor @2.30 GHz	Intel® Server Board S2600WFT
Worker1-10	Intel® Xeon® Gold 6238R Processor @2.2GHz	Intel® Server Board S2600WFT
Ceph I001-008	Intel® Xeon® Gold 6238R Processor @2.2GHz	Intel® Server Board S2600WFT
Jumphost, Bootstrap	Intel® Xeon® CPUE5-2699v4 @2.20GHz	Intel® Server Board S2600WFT

Switch settings:

Host	Cable connector	Speed	Setting on switch	Nic
Master001-003	SFP	25gbe	Single Port Breakout: 1/4 cable 25gbe/line 100gbe at port	XXV710-QDA2
Worker1-10	QSFP+	100gbe	Single port: Speed forced 100gfull	E810-CQ
Ceph-node 001-008 (OpenShift Data Foundation)	QSFP+	40gbe	Single Port: Speed forced 40gfull MTU 9000 Channel Group 1** mode active** lacp timer fast	2x XL710-QD
Jumphost	SFP	25gbe	Single Port Breakout: 4/4 cable 25gbe/line 100gbe at port	XXV710-QDA2
Bootstrap	SFP	25gbe	Single port: 25gbe Speed forced 25gbfull	XXV710-QDA2

**Each channel group can contain only two ports before the switch will shut off excessive ports for port availability.

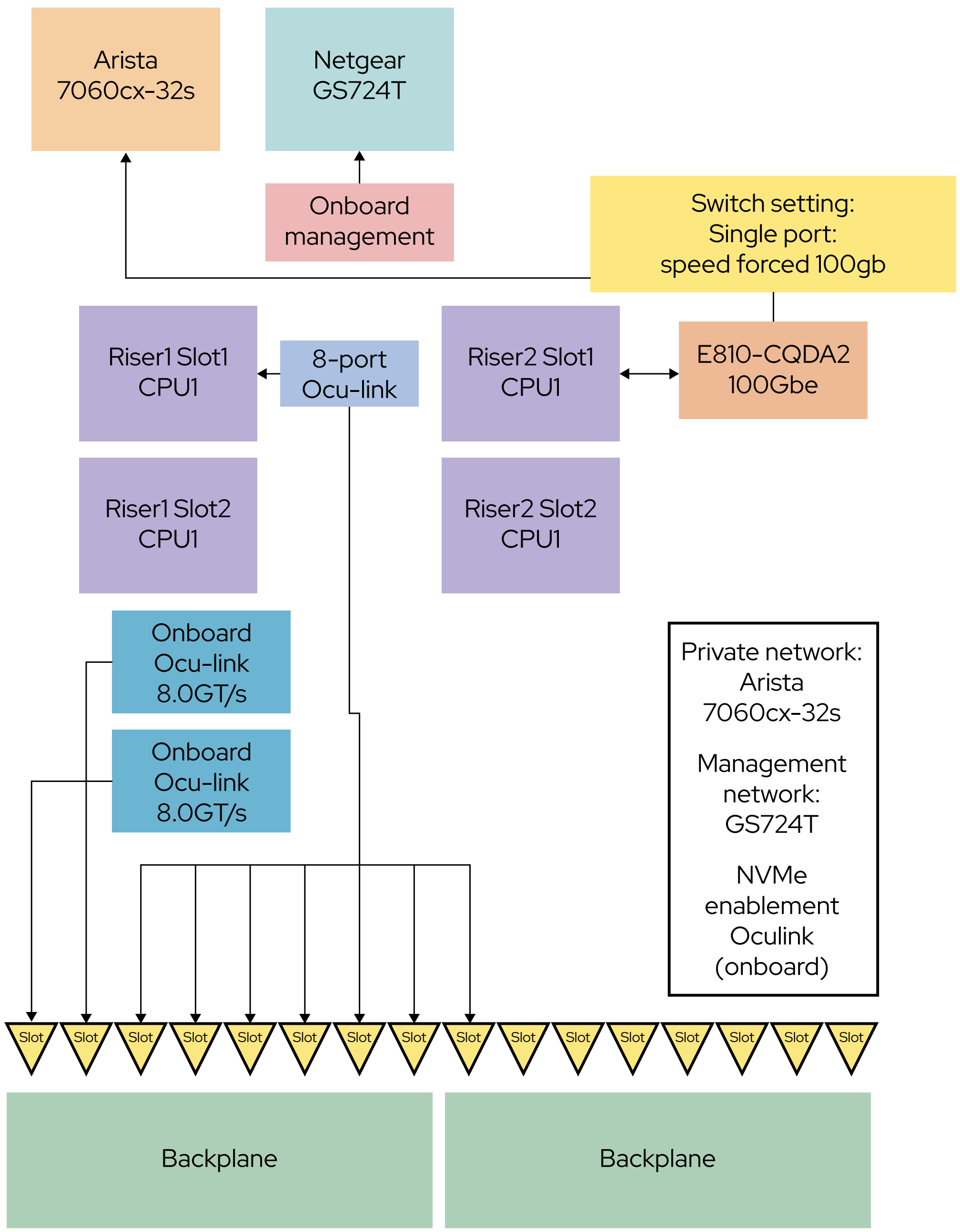
** Mode active required for successful bonding

OS bonding settings	BONDING_OPTS= “ mode=4 lacp_rate=fast xmlt_hash_policy=1” -Mode 4 = 802.3ad -lacp_rate=fast is your package rate -xmit_hash_policy=1 is your layer 3+4 policy
BIO’s version	Requires reboot to confirm
Memory speed	Requires reboot to confirm

Drives:

Model	Size	Type	Links:
P5800X	1.6tb	Optane	https://ark.intel.com/content/www/us/en/ark/products/201859/intel-optane-ssd-dc-p5800x-series-1-6tb-2-5in-pcie-x4-3d-xpoint.html
P5800X	800gb	Optane	https://ark.intel.com/content/www/us/en/ark/products/201860/intel-optane-ssd-dc-p5800x-series-800gb-2-5in-pcie-x4-3d-xpoint.html
P4610	1.6tb	TLC	https://ark.intel.com/content/www/us/en/ark/products/140103/intel-ssd-dc-p4610-series-1-6tb-2-5in-pcie-3-1-x4-3d2-tlc.html
P4800X	750gb	QLC	https://ark.intel.com/content/www/us/en/ark/products/97154/intel-optane-ssd-dc-p4800x-series-750gb-2-5in-pcie-x4-3d-xpoint.html
P4320	7.68tb	QLC	https://ark.intel.com/content/www/us/en/ark/products/186679/intel-ssd-d5-p4320-series-7-68tb-2-5in-pcie-3-1-x4-3d2-qlc.html
S3510	1.6tb	Boot Drive	https://ark.intel.com/content/www/us/en/ark/products/86192/intel-ssd-dc-s3510-series-1-6tb-2-5in-sata-6gb-s-16nm-mlc.html

Compute node configuration



Arista 7060cx-32s

Netgear GS724T

Onboard management

Switch setting: Single port: speed forced 100gb

Riser1 Slot1 CPU1

8-port Ocu-link

Riser2 Slot1 CPU1

E810-CQDA2 100Gbe

Riser1 Slot2 CPU1

Riser2 Slot2 CPU1

Onboard Ocu-link 8.0GT/s

Onboard Ocu-link 8.0GT/s

Private network: Arista 7060cx-32s

Management network: GS724T

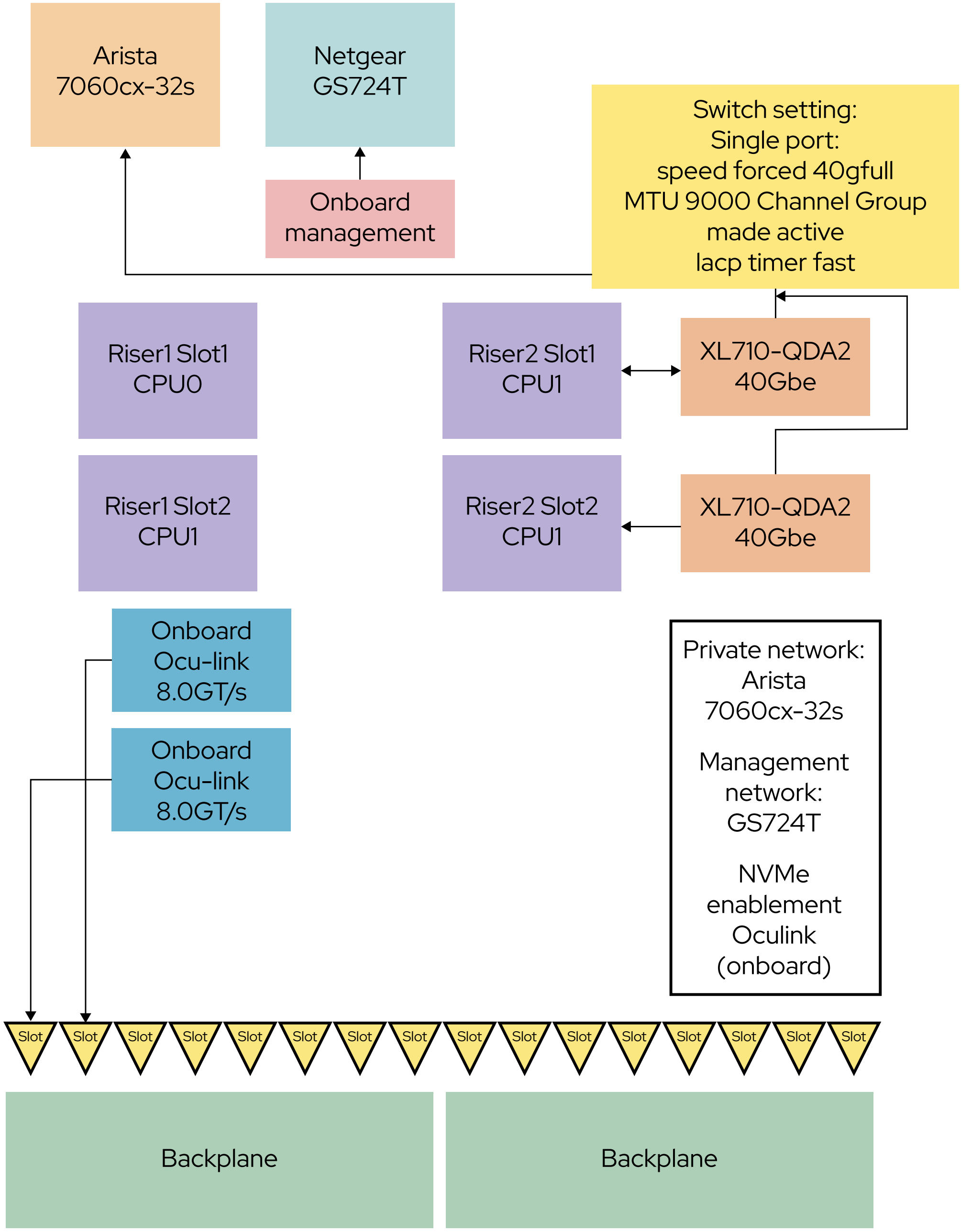
NVMe enablement Oculink (onboard)

Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot

Backplane

Backplane

Storage node configuration



Arista 7060cx-32s

Netgear GS724T

Onboard management

Switch setting:
Single port:
speed forced 40gfull
MTU 9000 Channel Group made active
lacp timer fast

Riser1 Slot1 CPU0

Riser2 Slot1 CPU1

XL710-QDA2 40Gbe

Riser1 Slot2 CPU1

Riser2 Slot2 CPU1

XL710-QDA2 40Gbe

Onboard Ocu-link 8.0GT/s

Onboard Ocu-link 8.0GT/s

Private network:
Arista
7060cx-32s

Management
network:
GS724T

NVMe
enablement
Oculink
(onboard)

Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot Slot

Backplane

Backplane

NVMe connections

Intel Optane SSDs are connected at the front cage backplane directly to onboard Oculink NVMe connections. Additional NVMe devices are connected via an 8-port Oculink switch on an x16 riser connected to the front cage backplane.

Server board link:

Intel® Server Board S2600WFT	https://ark.intel.com/content/www/us/en/ark/products/89015/intel-server-board-s2600wft.html
Intel® Server Board S2600WTT	https://ark.intel.com/content/www/us/en/ark/products/82156/intel-server-board-s2600wtt.html

CPU links:

Intel® Xeon® Gold Processor 6140 @2.30 GHz	https://ark.intel.com/content/www/us/en/ark/products/120485/intel-xeon-gold-6140-processor-24-75m-cache-2-30-ghz.html
Intel® Xeon® Processor E5-2699v4 @2.20GHz	https://ark.intel.com/content/www/us/en/ark/products/91317/intel-xeon-processor-e5-2699-v4-55m-cache-2-20-ghz.html
Intel® Xeon® Gold Processor 6238R @2.2GHz	https://ark.intel.com/content/www/us/en/ark/products/199345/intel-xeon-gold-6238r-processor-38-5m-cache-2-20-ghz.html

NIC links:

XXV710-DA2 25gbe	https://ark.intel.com/content/www/us/en/ark/products/95260/intel-ethernet-network-adapter-xxv710-da2.html
E810-CQDA2 100gbe	https://ark.intel.com/content/www/us/en/ark/products/192558/intel-ethernet-network-adapter-e810-cqda2.html
XL710-QDA2 40gbe	https://ark.intel.com/content/www/us/en/ark/products/83967/intel-ethernet-converged-network-adapter-xl710-qda2.html

References

TPCDS specification:
http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-ds_v2.4.0.pdf

TeraGen and TeraSort:
https://www.ibm.com/docs/en/platform-symphony/7.1.1?topic=applications-terasort-benchmark

IBM Cloud Pack for Data:
https://www.ibm.com/support/producthub/icpdata/docs/content/SSQNUZ_latest/cpd/overview/overview.html

Data skipping:
https://xskipper.io/master/

https://www.computer.org/csdl/proceedings-article/big-data/2020/09377740/1s64k8kunyo

Intel product specifications:
https://ark.intel.com/content/www/us/en/ark/products/201859/intel-optane-ssd-dc-p5800x-series-1-6tb-2-5in-pcie-x4-3d-xpoint.html

https://ark.intel.com/content/www/us/en/ark/products/187937/intel-optane-ssd-dc-p4800x-series-with-intel-memory-drive-technology-750gb-2-5in-pcie-x4-3d-xpoint.html

https://ark.intel.com/content/www/us/en/ark/products/140103/intel-ssd-dc-p4610-series-1-6tb-2-5in-pcie-3-1-x4-3d2-tlc.html

https://www.intel.com/content/www/us/en/products/sku/186679/intel-ssd-d5p4320-series-7-68tb-2-5in-pcie-3-1-x4-3d2-qlc/specifications.html?wapkw=P4320%207.68%20TB

https://ark.intel.com/content/www/us/en/ark/products/199345/intel-xeon-gold-6238r-processor-38-5m-cache-2-20-ghz.html?wapkw=6238r

Intel Scalable Processors product brief:
https://www.intel.com/content/www/us/en/products/docs/processors/xeon/3rd-gen-xeon-scal-able-processors-brief.html

Red Hat resources:

Red Hat OpenShift:
https://www.redhat.com/en/technologies/cloud-computing/openshift

Red Hat OpenShift Data Foundation:
https://www.redhat.com/en/technologies/cloud-computing/openshift-data-foundation

Red Hat OpenShift Data Foundation external mode:
https://www.redhat.com/en/resources/openshift-storage-deployment-agility-brief

Intel data node configurations for Red Hat OpenShift Data Foundation:
https://www.redhat.com/en/resources/easily-scale-apps-intel-brief

https://www.redhat.com/en/resources/data-node-config-openshift-overview

Red Hat OpenShift Data Foundation data node configurations from Dell Technologies:
https://www.redhat.com/en/resources/data-nodes-openshift-intel-dell-datasheet

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

NORTH AMERICA
1 888 REDHAT1

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