White Paper

Bare-metal performance for Big Data workloads on Docker* containers

BlueData® EPIC™

Intel® Xeon® Processor

BlueData® and Intel® have collaborated in an unprecedented benchmark of the performance of Big Data workloads. These workloads are benchmarked in a bare-metal environment versus a container-based environment that uses the BlueData EPIC™ software platform. Results show that you can take advantage of the BlueData benefits of agility, flexibility, and cost reduction while running Apache Hadoop in Docker* containers, and still gain the performance of a bare-metal environment.*

ABSTRACT

In a benchmark study, Intel compared the performance of Big Data workloads running on a bare-metal deployment versus running in Docker* containers with the BlueData® EPIC™ software platform. This landmark benchmark study used unmodified Apache Hadoop* workloads. The workloads for both test environments ran on apples-to-apples configurations on Intel® Xeon® processor-based architecture. The goal was to find out if you could run Big Data workloads in a container-based environment without sacrificing the performance that is so critical to Big Data frameworks.

This in-depth study shows that performance ratios for container-based Hadoop workloads on BlueData EPIC are equal to — and in some cases, better than — bare-metal Hadoop. For example, benchmark tests showed that the BlueData EPIC platform demonstrated an average 2.33% performance gain over bare metal, for a configuration with 50 Hadoop compute nodes and 10 terabytes (TB) of data.[1] These performance results were achieved without any modifications to the Hadoop software.

This is a revolutionary milestone, and the result of an ongoing collaboration between Intel and BlueData software engineering teams.

This paper describes the software and hardware configurations for the benchmark tests, as well as details of the performance benchmark process and results.

DEPLOYING BIG DATA ON DOCKER* CONTAINERS WITH BLUEDATA® EPIC™

The BlueData EPIC software platform uses Docker containers and patent-pending innovations to simplify and accelerate Big Data deployments. The container-based clusters in the BlueData EPIC platform look and feel like standard physical clusters in a bare-metal deployment. They also allow multiple business units and user groups to share the same physical cluster resources. In turn, this helps enterprises avoid the complexity of each group needing its own dedicated Big Data infrastructure.

With the BlueData EPIC platform, users can quickly and easily deploy Big Data frameworks (such as Hadoop and Apache Spark*), and at the same time reduce costs. BlueData EPIC delivers these cost savings by

improving hardware utilization, reducing cluster sprawl, and minimizing the need to move or replicate data. BlueData also provides simplified administration and enterprise-class security in a multi-tenant architecture for Big Data.

One of the key advantages of the BlueData EPIC platform is that Hadoop and Spark clusters can be spun up on-demand. This delivers a key benefit: Data science and analyst teams can create self-service clusters without having to submit requests for scarce IT resources or wait for an environment to be set up for them. Instead, within minutes, scientists and analysts can rapidly deploy their preferred Big Data tools and applications — with security-enabled access to the data they need. The ability to quickly explore, analyze, iterate, and draw insights from data helps these users seize business opportunities while those opportunities are still relevant.

With BlueData EPIC, enterprises can take advantage of this Big-Data-as-a-Service experience for greater agility, flexibility, and cost efficiency. The platform can also be deployed on-premises, in the public cloud, or in a hybrid architecture.

In this benchmark study, BlueData EPIC was deployed on-premises, running on Intel® Architecture with Intel Xeon processors.

THE CHALLENGE: PROVE CONTAINER-BASED BIG DATA PERFORMANCE IS COMPARABLE TO BARE-METAL

Performance is of the utmost importance for deployments of Hadoop and other Big Data frameworks. To ensure the highest possible performance, enterprises have traditionally deployed Big Data analytics almost exclusively on bare-metal servers. They have not traditionally used virtual machines or containers because of the processing overhead and I/O latency that is typically associated with virtualization and container-based environments.

As a result, most on-premises Big Data initiatives have been limited in terms of agility. For example, up to now, infrastructure changes (such as provisioning new servers for Hadoop) often take weeks or even months to complete. This infrastructure complexity continues to slow the adoption of Hadoop in enterprise deployments. (Many of the same deployment challenges seen with Hadoop also apply to on-premises implementations for Spark and other Big Data frameworks.)

The BlueData EPIC software platform is specifically tailored to the performance needs for Big Data. For example, BlueData EPIC boosts the I/O performance and scalability of container-based clusters with hierarchical data caching and tiering.

In this study, the challenge for BlueData was to prove — with third-party validated and quantified benchmarking results — that BlueData EPIC could deliver comparable performance to bare-metal deployments for Hadoop, Spark, and other Big Data workloads.

COLLABORATION AND BENCHMARKING WITH INTEL®

In August 2015, Intel and BlueData embarked on a broad strategic technology and business collaboration. The two companies aimed at reducing the complexity of traditional Big Data deployments. In turn, this would help accelerate adoption of Hadoop and other Big Data technologies in the enterprise space. One of the goals of this collaboration was to optimize the performance of BlueData EPIC when running on the leading data-center architecture: Intel Xeon processor-based technology.

The Intel and BlueData teams worked closely to investigate, benchmark, test and enhance the BlueData EPIC platform in order to ensure flexible, elastic, and high-performance Big Data deployments. To this end, BlueData also asked Intel to help identify specific areas that could be improved or optimized. The main goal was to increase the performance of Hadoop and other real-world Big Data workloads in a container-based environment.

TEST ENVIRONMENTS FOR PERFORMANCE BENCHMARKING

To ensure an apples-to-apples comparison, the Intel team evaluated benchmark execution times in a bare-metal environment, and in a container-based environment using BlueData EPIC. Both the bare-metal and BlueData EPIC test environments ran on identical hardware. Both environments used the CentOS Linux*

Results apply to other Apache Hadoop* distributions

BlueData® EPIC™ allows you to deploy Big Data frameworks and distributions completely unmodified. In the benchmarking test environment, we used Cloudera* as the Apache Hadoop* distribution. However, because BlueData runs the distribution unmodified, these performance results can also apply to other Hadoop distributions — such as Hortonworks* and MapR.*

Table 1. Setup for test environments

Bare-metal test environment	BlueData® EPIC™ test environment
Cloudera Distribution Including Apache Hadoop* (CDH) 5.7.0 Express* with Cloudera Manager*	CDH 5.7.0 Express with Cloudera Manager
CentOS* 6.7[a]	CentOS 6.8[a]
2 Intel® Xeon® processor E5-2699 v3, 2.30 GHz	2 Intel Xeon processor E5-2699 v3, 2.30 GHz
256 GB DIMM	256 GB DIMM
7 Intel® SSD DC S3710, 400 GB: 2 SSDs allocated for Apache Hadoop MapReduce* intermediate data, and 5 allocated for the local Apache Hadoop* distributed file system (HDFS)	7 Intel SSD DC S3710, 400 GB: 2 SSDs allocated for node storage, and 5 allocated for local HDFS
One 10Gbit Ethernet for management, and another for data	One 10Gbit Ethernet for management, and another for access to the data via BlueData DataTap™ technology

[a] CentOS 6.7 was pre-installed in the bare-metal test environment. BlueData EPIC requires CentOS 6.8. After analysis, the benchmarking teams believe that the difference in OS versions did not have a material impact on performance between the bare-metal and BlueData EPIC test environments.

operating system and the Cloudera Distribution Including Apache Hadoop* (CDH). In both test environments, Cloudera Manager* software was used to configure one Apache Hadoop YARN* (Yet Another Resource Negotiator) controller as the resource manager for each test setup. The software was also used to configure the other Hadoop YARN workers as node managers.

Test environments used Intel® Xeon® processor-based servers

The Big Data workloads in both test environments were deployed on the Intel® Xeon® processor E5-2699 v3 product family. These processors help reduce network latency, improve infrastructure security, and minimize power inefficiencies. Using two-socket servers powered by these processors brings many benefits to the BlueData EPIC platform, including:

- **Improved performance and density.** With increased core counts, larger cache, and higher memory bandwidth, Intel Xeon processors deliver dramatic improvements over previous processor generations.
- **Hardware-based security.** Intel® Platform Protection Technology enhances protection against malicious attacks. Intel Platform Protection Technology includes Intel® Trusted Execution Technology, Intel® OS Guard, and Intel® BIOS Guard.
- **Increased power efficiency.** In Intel Xeon processors, per-core P states dynamically respond to changing workloads, and adapt power levels on each individual core. This helps them deliver better performance per watt than previous generation platforms.

Both test environments also used Intel® Solid-State Drives (Intel® SSDs) to optimize the execution environment at the system level. For example, the Intel® SSD Data Center (Intel® SSD DC) P3710 Series1 delivers high performance and low latency that help accelerate container-based Big Data workloads. This optimized performance is achieved with connectivity based on the Non-Volatile Memory Express (NVMe) standard and eight lanes of PCI Express* (PCIe*) 3.0.

Test environment configuration setups

The setups for both the bare-metal and BlueData EPIC test environments are described in Table 1 and figures 1 and 2. The container-based environment used BlueData EPIC version 2.3.2. The data in this environment was accessed from the Hadoop distributed file system (HDFS) over the network, using BlueData DataTap™ technology (a Hadoop-compliant file system service).

Test environment topologies

The performance benchmark tests were conducted for 50-node, 20-node, and 10-node configurations. However, for both environments, only 49 physical hosts were actually used in the 50-node configuration. This was because one server failed during the benchmark process, and had to be dropped from the environment. Both environments were then configured with 49 physical hosts. For simplicity in this paper, including in the figures, tables, and graphs, we continue to describe this configuration as a 50-node configuration.

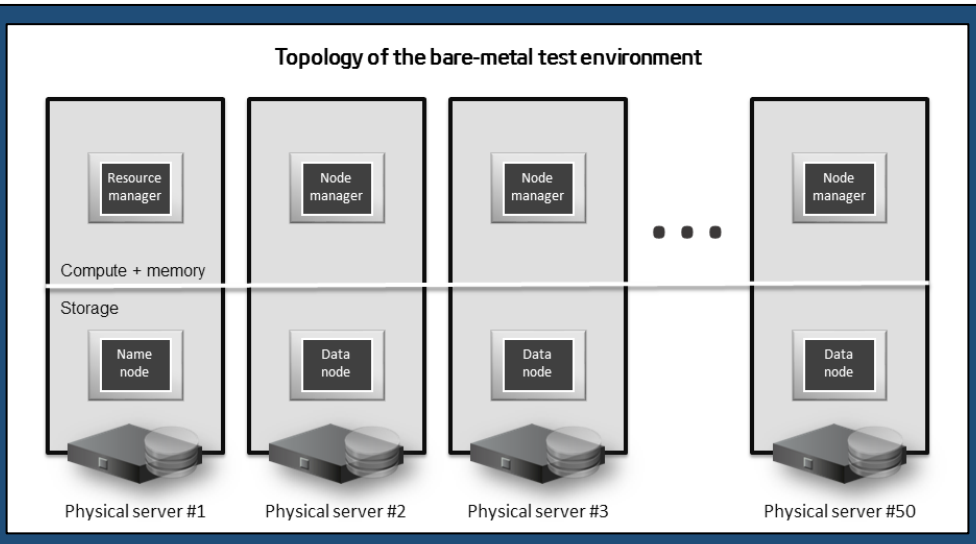

Figure 1. Topology of the bare-metal test environment. Apache Hadoop compute and memory services run directly on physical hosts. Storage for the Hadoop distributed file system (HDFS) is managed on physical disks.*

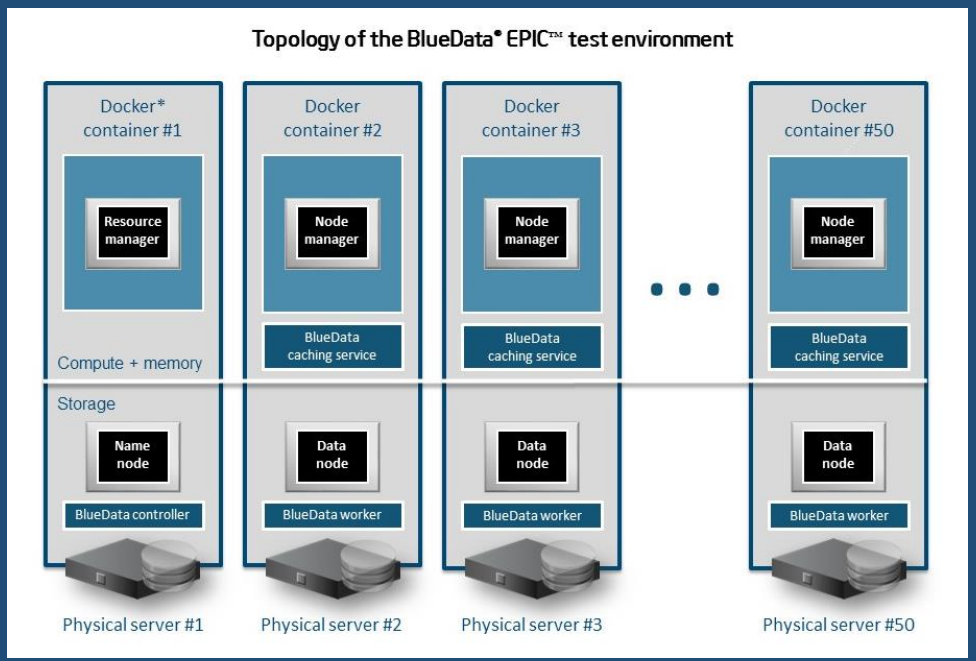

Figure 2. Topology of the BlueData® EPIC™ test environment. In this environment, the Apache Hadoop compute and memory services run in Docker* containers (one container per physical server). Like the bare-metal environment, storage for the Hadoop distributed file system (HDFS) is managed on physical disks. The container-based Hadoop cluster is auto-deployed by BlueData EPIC using the Cloudera Manager* application programming interface (API).*

In the bare-metal test environment, the physical CDH cluster had a single, shared HDFS storage service. Along with the native Hadoop compute services, additional services ran directly on the physical hosts. Figure 1 shows the 50-node bare-metal configuration with 1 host configured as the master node, and the remaining hosts configured as worker nodes.

Figure 2 shows the 50-node configuration for the BlueData EPIC test environment. In this configuration, 1 physical host was used to run the EPIC controller services, as well as run the container with the CDH master node. The other physical hosts each ran a single container with the CDH worker nodes. The unmodified CDH cluster in the test environment ran all native Hadoop compute services required for either the master node or worker nodes. On each host, services ran inside a single container. Separate shared HDFS storage was configured on physical disks for each of the physical hosts. In addition, a caching service (BlueData IOBoost™ technology) was installed on each of the physical hosts running the worker nodes. All source data and results were stored in the shared HDFS storage.

PERFORMANCE BENCHMARKING WITH BIGBENCH

The Intel-BlueData performance benchmark study used the BigBench benchmark kit for Big Data (BigBench).[2,3] BigBench is an industry-standard benchmark for measuring the performance of Hadoop-based Big Data systems.

The BigBench benchmark provides realistic, objective measurements and comparisons of the performance of modern Big Data analytics frameworks in the Hadoop ecosystem. These frameworks include Hadoop MapReduce,* Apache Hive,* and the Apache Spark Machine Learning Library* (MLlib).

BigBench designed for real-world use cases

BigBench was specifically designed to meet the rapidly growing need for objective comparisons of real-world applications. The benchmark's data model includes structured data, semi-structured data, and unstructured data. This model also covers both essential functional and business aspects of Big Data use cases, using the retail industry as an example.

Historically, online retailers recorded only completed transactions. Today's retailers demand much deeper insight into online consumer behavior. Simple shopping basket analysis techniques have been replaced by detailed behavior modeling. New forms of analysis have resulted in an explosion of Big Data analytics systems. Yet prior to BigBench, there have been no mechanisms to compare disparate solutions in real-world scenarios like this.

BigBench meets these needs. For example, to measure performance, BigBench uses 30 queries to represent Big Data operations that are frequently performed by both physical and/or online retailers. These queries simulate Big Data processing, analytics, and reporting in real-world retail scenarios. Although the benchmark was designed to measure performance for use cases in the retail industry, these are representative examples. The performance results in this study can be expected to be similar for other benchmarks, other use cases, and other industries.

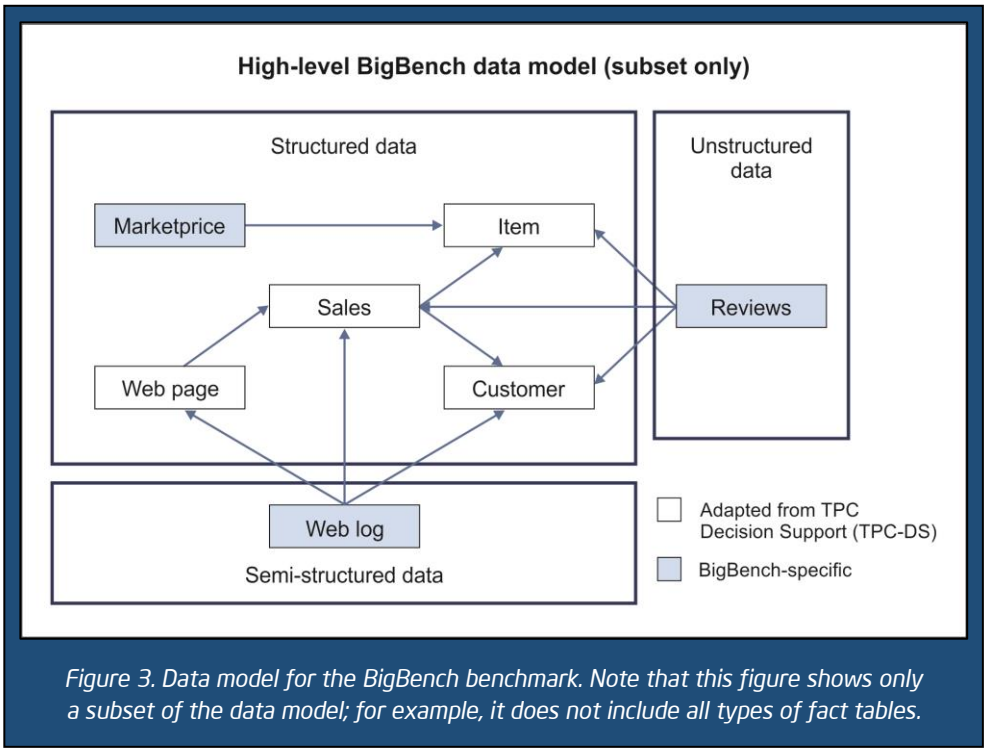

Figure 3. Data model for the BigBench benchmark. Note that this figure shows only a subset of the data model; for example, it does not include all types of fact tables.

BigBench performance metric: Query-per-minute (Qpm)

The primary BigBench performance metric is Query-per-minute (Qpm@Size), where size is the scale factor of the data. The metric is a measure of how quickly the benchmark runs (across various queries). The metric reflects three test phases:

- **Load test.** Aggregates data from various sources and formats.
- **Power test.** Runs each use case once to identify optimization areas and utilization patterns.
- **Throughput test.** Runs multiple jobs in parallel to test the efficiency of the cluster.

Benchmark data model

BigBench is designed with a multiple-snowflake schema inspired by the TPC Decision Support (TPC-DS) benchmark, using a retail model consisting of five fact tables. These tables represent three sales channels (store sales, catalog sales, and online sales), along with sales and returns data.

Figure 3 (above) shows a high-level overview of the data model. As shown in the figure, specific Big Data dimensions were added for the BigBench data model. Market price is a traditional relational table that stores competitors' prices.

Structured, semi-structured, and unstructured data

Structured, semi-structured, and unstructured data are very different.

- **Structured data.** Structured data typically accounts for only 20 percent of all data available. It is "clean" data, it is analytical, and it is usually stored in databases.
- **Semi-structured data.** Semi-structured data is a form of structured data that does not conform to the formal structure of data models.
- **Unstructured data.** Unstructured data is information that isn't organized in a traditional row-column database. For example, it could be text-oriented, like a set of product reviews.

The idea of using unstructured data for analysis has, in the past, been too expensive for most companies to consider. However, thanks to technologies such as Hadoop, unstructured data analysis is becoming more common in the business world.

Unstructured data is not useful when fit into a schema/table, unless there are specialized techniques that analyze some of the data and then store it in a column format. However, with the right Big Data analytics tools, unstructured data can add depth to data analysis that couldn't otherwise be achieved. In particular, using unstructured data to enhance its counterpart-structured data can provide deep insights.

Queries and semi-structured data

BigBench includes queries based on the TPC-DS benchmark that deals with structured data. BigBench also adds queries to address semi-structured and unstructured data for store and web sales channels. The semi-structured data represents a user's clicks from a retailer's website, to enable analysis of the user's behavior. This semi-structured data describes user actions that are different from a weblog, and so it varies in format.

The clickstream log contains data from URLs which are extracted from a webserver log. Typically, database and Big Data systems convert the webserver log to a table with five columns: DateID, TimeID, SalesID, WebPageID, and UserID. These tables are generated in advance to eliminate the need to extract and convert the webserver log information.

Unstructured data in the schema

The unstructured part of the schema is generated in the form of product reviews, which are, for example, used for sentiment analysis. Figure 3 (previous page) shows product reviews in the unstructured area (the right side of the figure). The figure shows their relationship to date, time, item, users, and sales tables in the structured area (left side of the figure). The implementation of product reviews is a single table with a structure similar to DateID, TimeID, SalesID, ItemID, ReviewRating, and ReviewText.

BENCHMARKING TEST RESULTS WITH BIGBENCH

For the comparison of the container-based environment versus bare-metal environment, each valid measurement included three phases: data load, power test, and throughput test. There are other phases in BigBench testing — such as raw data generation, and power/throughput validations — but these were not included in the final performance results for this study.

Overall, the results showed comparable performance for the bare-metal environment and the BlueData EPIC container-based environment. In fact, in some cases, the performance of BlueData EPIC is slightly better than that of bare-metal. For example, in the 50-node configuration with 10 terabytes of data, the container-based environment demonstrated an average 2.33% performance gain over bare-metal at Qpm@10TB.[1] This average was computed from the Qpm@10TB values for three runs ((2.43 + 3.37 +1.19)/3 = 2.33).[1]

Table 2 (next page) provides detailed results for the three test runs in each of the three phases, with the Qpm performance metric at 10TB. All performance results are based on well-tuned Hadoop, Hive, and Spark, with CDH in both environments. The positive percentages show that BlueData EPIC outperformed bare-metal in most categories.

In the load phase, negative percentages show where bare-metal outperformed BlueData EPIC. The load phase is dominated by data-write operations. Since Big Data workloads are typically dominated by read operations rather than write operations, the work Intel and BlueData have done to date has been focused on the data-read

.

Table 2: Summary of performance results for the 50-node test configuration[1]

BigBench Queries per minute (Qpm)	Test run #	Bare-metal (Q^M) (Qpm)	BlueData® EPIC™ (Q^E) (Qpm)	(Q^E–Q^M)/ Q^M [positive % means that BlueData EPIC out-performed bare-metal]
Qpm@10TB (queries per minute)	1	1264.596	1295.271118	2.43%
	2	1259.938	1302.347698	3.37%
	3	1270.047514	1285.106637	1.19%
BigBench test phase	**Test run #**	**Bare-metal (T^M) (execution time in seconds)**	**BlueData EPIC* (T^E) (execution time in seconds)**	**(T^M–T^E)/ T^M [positive % means that BlueData EPIC out-performed bare-metal]**
Load (seconds)	1	1745.237	1867.936	-7.03%
	2	1710.072	1850.185	-8.19%
	3	1700.569	1837.91	-8.08%
Power (seconds)	1	14843.442	14710.014	0.90%
	2	14854.622	14708.999	0.98%
	3	14828.417	14747.677	0.54%
Throughput (seconds)	1	26832.801	25274.873	5.80%
	2	26626.308	25563.988	3.99%
	3	26445.704	25911.958	2.02%

operations. Intel and BlueData are continuing to investigate load tests and optimize BlueData EPIC for these and other operations, and will publish new results when that work is complete.

Appendix A lists the settings used for these tests.

Figures 4 and 5 (next page) show some of the benchmark data from these tests. In Figure 4, the Qpm number is a measure of the number of benchmark queries that are executed per minute. In other words, the higher the number, the faster the benchmark is running.

As demonstrated in the chart, the workloads that ran in containers performed as well as or better than those that ran on bare-metal. The performance advantage for the container-based environment is due to the asynchronous storage I/O and caching provided by BlueData's IOBoost technology.

Figure 5 shows a performance comparison of the unmodified CDH distribution in environments with different numbers of nodes (10, 20, and 50). The performance ratios for Qpm, power, and throughput show equal or better performance for the BlueData EPIC container-based test environment than for the bare-metal test environment. These results were demonstrated for each configuration. In the load phase, results showed a slightly lower performance for the BlueData EPIC platform for the 20- and 50-node configurations.

Comparing test results for queries with BigBench

As mentioned earlier, BigBench features 30 complex queries. Ten of these queries are based on the TPC-DS benchmark, and the other 20 were developed specifically for BigBench. The 30 queries cover common Big Data analytics use cases for real-world retail deployments. These include merchandising pricing optimization, product return analysis, inventory management, customers, and product reporting. However, as noted previously, BigBench performance results can be applied to other industries.

In this performance study, we compared elapsed time, query-by-query for all 30 BigBench queries, for both the container-based and bare-metal environments.

Table 3 (next two pages) shows some of the results from the power test phase of this study. The results varied by query, but the average across all 30 queries was relatively comparable for the test two environments.

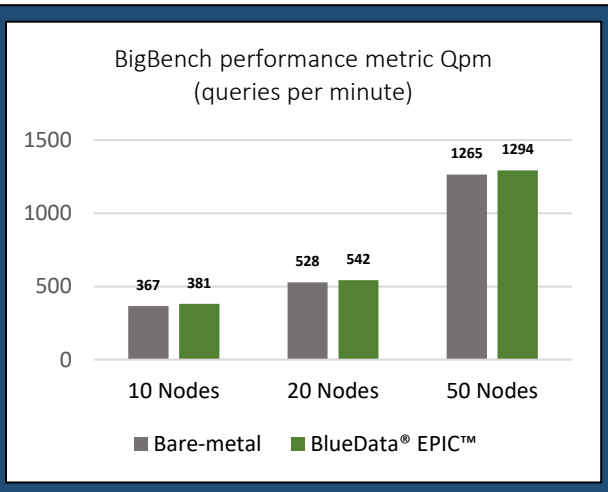

Figure 4. Performance comparison for Big Data workloads on bare-metal and BlueData® EPIC™ test environments.[1]

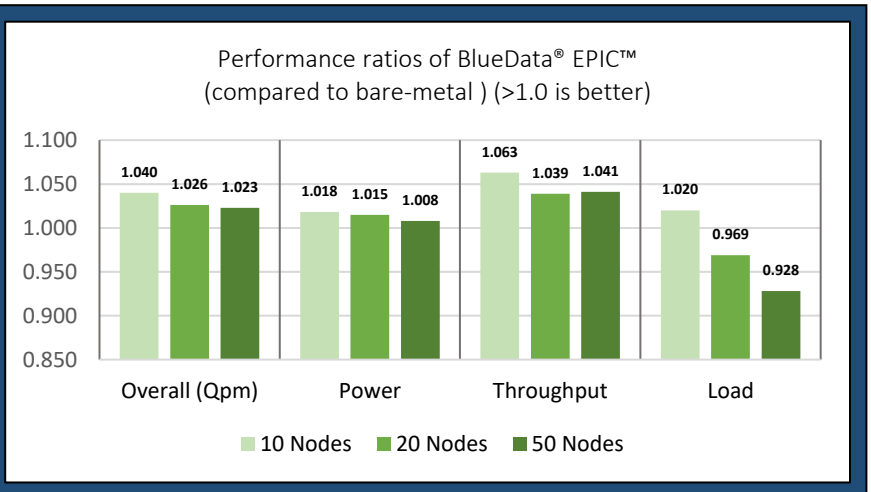

Figure 5. Ratio of BlueData® EPIC™-to-bare-metal performance on environments with different numbers of nodes.[1]

Table 3: Query-by-query results from the power test phase[1]

Query #	Type of query	Bare-metal (T^M)	BlueData® EPIC™ (T^E)	$(T^M-T^E)/T^M$ [positive % means BlueData EPIC out-performed bare-metal]
Q01	Structured, (UDP)/user-defined table function (UDTF)	249.973	228.155	8.70%
Q02	Semi-structured, MapReduce	1463.999	1264.605	13.62%
Q03	Semi-structured, MapReduce	866.394	817.696	5.62%
Q04	Semi-structured, MapReduce	1144.731	1134.284	0.91%
Q05	Semi-structured, machine language (ML)	2070.234	1978.271	4.44%
Q06	Structured, pure query language (QL)	412.234	389.261	5.57%
Q07	Structured, pure QL	188.222	195.323	-3.77%
Q08	Semi-structured, MapReduce	649.144	678.659	-4.55%
Q09	Structured, pure QL	378.873	379.262	-0.10%
Q10	Unstructured, UDF/UDTF/natural language processing (NLP)	574.235	564.193	1.75%
Q11	Structured, pure QL	147.582	159.467	-8.1%
Q12	Semi-structured, pure QL	550.647	573.972	-4.23%
Q13	Structured, pure QL	338.541	361.650	-6.82%
Q14	Structured, pure QL	79.201	85.000	-7.32%
Q15	Structured, pure QL	154.958	159.806	-3.13%
Q16	Structured, pure QL	1397.297	1307.499	6.42%
Q17	Structured, pure QL	325.894	307.258	5.72%
Q18	Unstructured, UDF/UDTF/NLP	1245.909	1281.291	-2.84%
Q19	Unstructured, UDF/UDTF/NLP	659.115	691.913	-4.98%
Q20	Structured, ML	366.52	367.806	-0.35%

Table 3: Query-by-query results from the power test phase[1] — *continued*

Query #	Type of query	Bare-metal (T^M)	BlueData® EPIC™ (T^E)	$(T^M-T^E)/T^M$ [positive % means BlueData EPIC out-performed bare-metal]
Q21	Structured, pure QL	1039.726	1011.864	2.68%
Q22	Structured, pure QL	232.01	215.346	7.18%
Q23	Structured, pure QL	197.69	207.341	-4.88%
Q24	Structured, pure QL	204.485	218.316	-6.76%
Q25	Structured, ML	905.222	895.200	1.10%
Q26	Structured, ML	1547.679	1543.607	0.26%
Q27	Unstructured, UDF/UDTF/NLP	65.569	62.309	4.97%
Q28	Unstructured	935.211	976.047	-4.37%
Q29	Structured, UDF/UDTF/NLP	892.346	845.908	5.20%
Q30	Semi-structured, UDF/UDTF/NLP MapReduce	2386.537	2206.385	7.55%

As you can see in Table 3, the queries used in the BigBench benchmark are grouped into structured, semi-structured, and unstructured queries; as well as into query categories. The four query categories are: Hive queries, Hive queries with MapReduce programs, Hive queries using natural language processing, and queries using Spark MLlib.

Appendix A lists brief descriptions of the queries used in this study. Appendix B lists the settings used to tune the workloads and queries for this study.

RESULTS SHOW COMPARABLE PERFORMANCE

The results detailed in the tables show that it is now possible to achieve similar performance for Hadoop in containers as compared to workloads that run in bare-metal environments. Specifically, there is no performance loss when running an unmodified Hadoop distribution on the BlueData EPIC software platform versus an identical setup on a bare-metal infrastructure. In some instances, the performance for BlueData EPIC can be slightly better than that of bare-metal.

The performance advantage of BlueData EPIC over bare-metal is due in large part to BlueData's IOBoost technology. IOBoost enhances the I/O performance with hierarchical tiers and data caching. IOBoost also improves the performance of single-copy data transfers from physical storage to the virtual cluster that is running in a container.

Other performance advantages of the BlueData EPIC platform are the result of performance improvements identified by the Intel research team, and implemented by BlueData as enhancements to the platform.

ENHANCEMENTS TO THE BLUEDATA EPIC PLATFORM

To identify possible performance improvements for BlueData EPIC software, Intel investigated potential bottlenecks (such as network maximum transmission unit, or MTU size). Intel also investigated a specific situation that caused YARN to launch jobs more slowly than on bare-metal.

Using that research, BlueData subsequently made enhancements to BlueData EPIC, which improved the parallelism when running Hadoop in Docker containers.

BlueData also implemented enhancements that took better advantage of data locality wherever possible. Additional investigations identified a number of useful changes that allowed BlueData to make further software enhancements to improve the platform's overall performance. Following are just a few of the improvements developed jointly by the Intel and BlueData teams.

Eliminating a network bottleneck due to MTU size

During benchmarking, Intel observed that the network bandwidth between the Docker containers on the BlueData EPIC platform was below expectations. Once the bottleneck was identified, BlueData investigated and discovered a configuration adjustment for BlueData EPIC that would improve performance. By reconfiguring the network MTU size from 1,500 bytes to

9,000 bytes, and by enabling jumbo-sized frames, BlueData was able to reap a solid increase in performance.

Reducing the latency of transferring storage I/O requests

Intel looked closely at the storage I/O requests being issued by the CDH software that ran on the BlueData EPIC platform. Intel noted that these I/O requests had a latency that was longer than when the same I/O requests were issued on the bare-metal configuration. The increase in latency was traced to the method being used to transfer the storage I/O requests. Specifically it was traced to the transfer of storage I/O requests from the BlueData implementation of the HDFS Java* client, to the BlueData caching node service (cnode). BlueData engineers reworked the way I/O requests were issued to the caching service, which improved performance by another substantial percentage.

Improving how quickly the YARN service launched jobs

Intel also carefully reviewed the Hadoop YARN statistics. They discovered that it was taking YARN longer to launch jobs when running on the BlueData EPIC platform than when running on bare-metal. Intel and BlueData found that this had to do with the calls to the remote HDFS that the platform was using to determine data block locality. BlueData used this information to implement changes that yielded a dramatic improvement in performance.

DEPLOYMENT CONSIDERATIONS AND GUIDANCE

As mentioned earlier, BlueData EPIC is distribution-agnostic for Big Data workloads. It runs with any Hadoop distribution. You do not need to modify the Hadoop distribution or other Big Data framework in order to run your workloads in Docker containers on the BlueData EPIC platform. The use of containers is also transparent, so Hadoop (and Spark) can be quickly and easily deployed in a lightweight container environment.

Because of this, enterprises can deploy Hadoop without requiring a detailed understanding of the intricacies of Docker and its associated storage and networking. Likewise, BlueData EPIC uses the underlying features of the physical storage devices for data backup, replication, and high availability. Because of this, enterprises do not need to modify existing processes to facilitate security and durability of data.

However, there are some best practices that may help enterprises with their Big Data deployments. On-going research and experimentation by Intel and BlueData have identified the following guidelines that could help system administrators achieve maximum performance when running Big Data workloads. These general guidelines are particularly relevant for I/O-bound workloads:

- **Configure systems to enhance disk performance.** The performance of the storage where the files are stored must be sufficient to avoid a bottleneck.
- **Provide sufficient network throughput.** The performance of the network connectivity between the hosts must be sufficient to avoid bottlenecks.
- **Deploy using current-generation Intel® processors.** The architecture of each generation of Intel Xeon processors delivers new advances in terms of performance and power efficiency. Using the newest generation of processor can mean significant benefits for Big Data workloads.

CONCLUSION

The BlueData EPIC software platform solves challenges that have traditionally slowed and/or stalled on-premises Big Data deployments. For example, BlueData EPIC provides the benefits of agility, flexibility, and cost efficiency of Docker containers while ensuring bare-metal performance. In turn, this helps eliminate the traditional barriers of complexity, and minimizes deployment time for Big Data adoption in the enterprise.

The extensive teamwork between BlueData and Intel has helped ensure ground-breaking performance for Hadoop and other Big Data workloads in a container-based environment. With this breakthrough, BlueData and Intel enable enterprises to take advantage of container technology to simplify and accelerate their Big Data implementations. Enterprises can now more easily and more effectively provide Big-Data-as-a-Service in their own data centers, powered by BlueData EPIC software and Intel Xeon processors. Data science teams can benefit from on-demand access to their Big Data environments, while leveraging enterprise-grade data governance and security in a multi-tenant architecture.

As a result, BlueData EPIC software, running on Intel architecture, is rapidly becoming the solution stack of choice for many Big Data initiatives.

APPENDIX A: BIGBENCH QUERIES USED IN THE STUDY'S PERFORMANCE BENCHMARKING

The following table briefly describes the thirty individual BigBench queries.

Table A-1. BigBench queries used in the performance benchmarking

Query	Description
Q01	For the given stores, identify the top 100 products that are frequently sold together.
Q02	For an online store, identify the top 30 products that are typically viewed together with the given product.
Q03	For a given product, get a list of last 5 products viewed the most before purchase.
Q04	Analyze web_clickstream shopping-cart abandonment.
Q05	Create a logistic regression model to predict if the visitor is interested in a given item category.
Q06	Identify customers who are shifting their purchase habit from in-store to web sales.
Q07	List top 10 states where, in last month, 10+ customers bought products costing 20% more than average product costs in same category.
Q08	Compare sales between customers who viewed online reviews versus those who did not.
Q09	Aggregate the total amount of sold items by different of combinations of customer attributes.
Q10	Extract sentences from a product's reviews that contain positive or negative sentiment.
Q11	Correlate sentiments on product monthly revenues by time period.
Q12	Identify customers who viewed product online, then bought that or a similar product in-store.
Q13	Display customers with both store and web sales where web sales are greater than store sales in consecutive years.
Q14	Calculate ratio between number of items sold for specific time in morning and evening, for customers with specific number of dependents.
Q15	Find the categories with flat or declining in-store sales for a given store.
Q16	For store sales, compute the impact of a change in item price.
Q17	Find the ratio of certain categories of items sold with and without promotions in a given month and year for a specific time zone.
Q18	Identify the stores with flat or declining sales in 3 consecutive months, and check any online negative reviews.
Q19	Identify the items with the highest number of returns in-store and online; and analyze major negative reviews.
Q20	Analyze customer segmentation for product returns.
Q21	Identify items returned by customer within 6 months, and which were subsequently purchased online within the next three years.
Q22	Compute the impact on inventory during the month before and the month after a price change.
Q23	Query with multiple iterations to calculate metrics for every item and warehouse, and then filter on a threshold.
Q24	Compute the crossprice elasticity of demand for a given product.
Q25	Group customers by date/time of last visit, frequency of visits, and monetary amount spent.
Q26	Group customers based on their in-store book purchasing histories.
Q27	Extract competitor product names and model names from online product reviews.
Q28	Build text classifier to classify the sentiments in online reviews.
Q29	Perform category affinity analysis for products purchased together online (order of viewing products doesn't matter).
Q30	Perform category affinity analysis for products viewed together online (order of viewing products doesn't matter).

APPENDIX B: CONFIGURATION SETTINGS FOR PERFORMANCE BENCHMARKING

The following configuration settings were used for both the bare-metal and BlueData EPIC test environments for the performance benchmarking.

Table B-1. Hadoop tuning settings

Component	Parameter	Setting used for performance benchmarking
Apache Hadoop YARN* resource manager	yarn.scheduler.maximum-allocation-mb	184GB
	yarn.scheduler.minimum-allocation-mb	1GB
	yarn.scheduler.maximum-allocation-vcores	68
	Yarn.resourcemanager.scheduler.class	Fair Scheduler
	yarn.app.mapreduce.am.resource.mb	4GB
YARN node manager	yarn.nodemanager.resource.memory-mb	204GB
	Java Heap Size of NodeManager in Bytes	3GB
	yarn.nodemanager.resource.cpu-vcores	68
YARN gateway	mapreduce.map.memory.mb	3GB
	mapreduce.reduce.memory.mb	3GB
	Client Java Heap Size in Bytes	2.5GB
	mapreduce.map.java.opts.max.heap	2.5GB
	mapreduce.reduce.java.opts.max.heap	2.5GB
	mapreduce.job.reduce.slowstart.completedmaps	0.8
Apache Hive* gateway	Client Java Heap Size in Bytes	3GB
	Java Heap Size of ResourceManager in Bytes	8GB
	Java Heap Size of JobHistory Server in Bytes	8GB
Apache Spark*	spark.io.compression.codec	org.apache.spark.io.LZ4CompressionCodec

Table B-2. Query tuning settings

Query #	Settings used to tune the test workloads
Global	set hive.default.fileformat=Parquet
1	set mapreduce.input.fileinputformat.split.maxsize=134217728;
2	set mapreduce.input.fileinputformat.split.maxsize=268435456; set hive.exec.reducers.bytes.per.reducer=512000000;
3	set mapreduce.input.fileinputformat.split.maxsize=268435456;

4	set mapreduce.input.fileinputformat.split.maxsize=536870912;
	set hive.exec.reducers.bytes.per.reducer=612368384;
	set hive.optimize.correlation=true;

Table B-2. Query tuning settings -- *continued*

Query #	Settings used to tune the test workloads

5	set mapreduce.input.fileinputformat.split.maxsize=268435456;
	set hive.mapjoin.smalltable.filesize=100000000;
	set hive.optimize.skew.join=true;
	set hive.skewjoin.key=100000;

6	set mapreduce.input.fileinputformat.split.maxsize=134217728;
	set hive.mapjoin.smalltable.filesize=10000000;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;
	set hive.optimize.ppd=true;
	set hive.optimize.ppd.storage=true;
	set hive.ppd.recognizetransivity=true;
	set hive.optimize.index.filter=true;
	set mapreduce.job.reduce.slowstart.completedmaps=1.0;

7	set mapreduce.input.fileinputformat.split.maxsize=536870912;
	set hive.exec.reducers.bytes.per.reducer=536870912;
	set hive.auto.convert.join=true;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.auto.convert.join.noconditionaltask.size=100000000;

8	set mapreduce.input.fileinputformat.split.maxsize=134217728;
	set hive.exec.reducers.bytes.per.reducer=256000000;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;
	set hive.auto.convert.join=true;
	set hive.mapjoin.smalltable.filesize=25000000;

9	set mapreduce.input.fileinputformat.split.maxsize=134217728;
	set hive.exec.reducers.bytes.per.reducer=256000000;
	set hive.exec.mode.local.auto=true;
	set hive.exec.mode.local.auto.inputbytes.max=1500000000;
	set hive.auto.convert.join=true;
	set hive.mapjoin.smalltable.filesize=25000000;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.auto.convert.join.noconditionaltask.size=10000;

10	set mapreduce.input.fileinputformat.split.maxsize=16777216;

11	set mapreduce.input.fileinputformat.split.maxsize=536870912;
	set hive.exec.reducers.bytes.per.reducer=512000000;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

12	set mapreduce.input.fileinputformat.split.maxsize=536870912;
	set hive.exec.reducers.bytes.per.reducer=16777216;
	set hive.auto.convert.join=true;
	set hive.mapjoin.smalltable.filesize=100000000;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.auto.convert.join.noconditionaltask.size=50000000;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

Table B-2. Query tuning settings -- *continued*

Query #	Settings used to tune the test workloads

13	set hive.exec.reducers.bytes.per.reducer=128000000;
	set hive.mapjoin.smalltable.filesize=85000000;
	set hive.auto.convert.sortmerge.join=true;
	set hive.auto.convert.join.noconditionaltask.size=10000;
	set mapreduce.input.fileinputformat.split.maxsize=134217728;

14	set mapreduce.input.fileinputformat.split.maxsize=2147483648;
	set hive.exec.reducers.bytes.per.reducer=128000000;
	set hive.exec.parallel=true;

15	set mapreduce.input.fileinputformat.split.maxsize=536870912;
	set hive.exec.reducers.bytes.per.reducer=128000000;
	set hive.mapjoin.smalltable.filesize=85000000;
	set hive.auto.convert.sortmerge.join=true;
	set hive.auto.convert.join.noconditionaltask.size=10000;

16	set hive.mapjoin.smalltable.filesize=100000000; set hive.exec.reducers.max=1200;

17	set mapreduce.input.fileinputformat.split.maxsize=134217728;
	set hive.exec.reducers.bytes.per.reducer=128000000;
	set hive.mapjoin.smalltable.filesize= 85000000;
	set hive.auto.convert.sortmerge.join=true;
	set hive.auto.convert.join.noconditionaltask.size=100000000;
	set hive.exec.parallel=true;

18	set mapreduce.input.fileinputformat.split.maxsize=33554432;
	set hive.exec.reducers.bytes.per.reducer=512000000;
	set hive.mapjoin.smalltable.filesize=85000000;
	set hive.auto.convert.sortmerge.join=true;
	set hive.auto.convert.join.noconditionaltask.size=10000;

19	set mapreduce.input.fileinputformat.split.maxsize=33554432;
	set hive.exec.reducers.bytes.per.reducer=16777216;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

20	set mapreduce.input.fileinputformat.split.maxsize=134217728;
	set mapreduce.task.io.sort.factor=100;
	set mapreduce.task.io.sort.mb=512;
	set mapreduce.map.sort.spill.percent=0.99;

21	set hive.auto.convert.join=true;
	set hive.auto.convert.join.noconditionaltask.size=100000000;
	set hive.exec.reducers.max=500;

22	set mapreduce.input.fileinputformat.split.maxsize=33554432;
	set hive.exec.reducers.bytes.per.reducer=33554432;
	set hive.auto.convert.join=true;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.mapjoin.smalltable.filesize=100000000;
	set hive.auto.convert.join.noconditionaltask.size=100000000;
	set hive.groupby.skewindata=false;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

Table B-2. Query tuning settings -- *continued*

Query #	Settings used to tune the test workloads

23	set mapreduce.input.fileinputformat.split.maxsize=33554432;
	set hive.exec.reducers.bytes.per.reducer=134217728;
	set hive.auto.convert.join=true;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.auto.convert.join.noconditionaltask.size=100000000;
	set hive.groupby.skewindata=false;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

24	set mapreduce.input.fileinputformat.split.maxsize=1073741824;
	set hive.auto.convert.join=true;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.auto.convert.join.noconditionaltask.size=100000000;
	set hive.groupby.skewindata=false;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

25	set mapreduce.input.fileinputformat.split.maxsize=268435456;
	set hive.exec.reducers.bytes.per.reducer=512000000;
	set hive.exec.mode.local.auto=true;
	set hive.exec.mode.local.auto.inputbytes.max=1500000000;

26	set mapreduce.input.fileinputformat.split.maxsize=134217728;
	set hive.exec.reducers.bytes.per.reducer=256000000;
	set hive.mapjoin.smalltable.filesize=100000000;

27	set mapreduce.input.fileinputformat.split.maxsize=33554432;
	set hive.exec.reducers.bytes.per.reducer=32000000;
	set hive.mapjoin.smalltable.filesize=10000000;
	set hive.exec.mode.local.auto=true;
	set hive.exec.mode.local.auto.inputbytes.max=1500000000;
	set mapreduce.job.ubertask.enable=true;

28	set mapreduce.input.fileinputformat.split.maxsize= 67108864;

29	set mapreduce.input.fileinputformat.split.maxsize=268435456;
	set hive.exec.reducers.bytes.per.reducer=256000000;
	set hive.auto.convert.join=true;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.auto.convert.join.noconditionaltask.size=100000000;
	set hive.groupby.skewindata=false;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

30	set mapreduce.input.fileinputformat.split.maxsize=536870912;
	set hive.exec.reducers.bytes.per.reducer=256000000;
	set hive.exec.reducers.max=3000;
	set hive.auto.convert.join=true;
	set hive.auto.convert.join.noconditionaltask=true;
	set hive.auto.convert.join.noconditionaltask.size=100000000;
	set hive.groupby.skewindata=false;
	set hive.exec.parallel=true;
	set hive.exec.parallel.thread.number=8;

Find out how to run your Big Data workloads on Docker containers and Intel Architecture with BlueData EPIC, by visiting intel.com/bigdata and bluedata.com

FIND OUT MORE:

Intel blog: "Simplify Big Data Deployment." Visit:
http://blogs.intel.com/evangelists/2015/08/25/simplify-big-data-deployment/

BlueData blog: "New Funding and Strategic Collaboration with Intel." Visit:
http://bluedata.com/blog/2015/08/new-funding-and-strategic-collaboration-with-intel

[1] Intel and BlueData EPIC Benchmark Results: https://goo.gl/RivvKI (179 MB). This .zip folder contains log files with detailed benchmark test results conducted in January, 2017.

[2] The BigBench benchmark kit used for these performance tests is not the same as TPCx-BigBench, the TPC Big Data batch analytics benchmark. As such, results presented here are not directly comparable to published results for TPCx-BigBench.

[3] BigBench benchmark kit: https://github.com/intel-hadoop/Big-Data-Benchmark-for-Big-Bench