PROWESS

Technical Research Report

Java-Based Benchmarking Shines a Light on How Underlying Architecture Impacts Cloud Performance

Use Java Development Kit (JDK®) benchmarking results to make evidence-based management decisions that help you achieve optimal performance from cloud-native applications.

Executive Summary

The point of benchmarking is to translate lab results into actionable business decisions. For example, say you want improved performance from cloud-native, Java-based applications. Benchmarking can help you make decisions about purchasing software and hardware and provisioning resources backed by evidence.

It is a given that improving workload performance in the cloud can help satisfy your customers' ever-growing appetite for fast, accurate data delivery. Customers who get stuck with a slow or faltering experience while performing search queries, watching media streams, or accessing medical or financial information might soon take their business elsewhere. Of course, improved performance is only one reason why so many organizations have migrated workloads such as Elasticsearch®, Apache Cassandra®, and Apache Spark™ to the cloud. Organizations of all sizes are using cloud-based platforms to stay competitive, meet service-level agreements (SLAs), and lower total cost of ownership (TCO).

Applying benchmarking best practices, Prowess Consulting tested cloud-native applications to surface underlying factors affecting workload performance, including Java Development Kit (JDK®) version, CPU type, and instance size. Benchmark testing of Java-based workloads in an Amazon® Elastic Compute Cloud™ (Amazon EC2®) environment revealed that using the latest long-term support (LTS) release of JDK improved the overall performance of Java-based applications across CPU types and instance sizes. Testing revealed that cloud instances powered by Intel® Xeon® Scalable processors consistently outperformed those powered by Amazon Web Services® (AWS®) Graviton2 and AMD EPYC™ processors.

The workload performance improvements showed variations among different instance sizes, which suggests that when provisioning cloud instances, you should consider instance-size-related memory and networking requirements, in addition to compute performance.

Upgrading to the latest supported LTS release of JDK® can immediately and significantly boost the performance of cloud-native applications up to …

10%
for Elasticsearch®[1]

4%
for Apache Cassandra®[2]

19%
for Apache Spark™[3]

Cloud-native applications run better on **3rd Gen Intel® Xeon® Scalable processors** than on AWS® Graviton2 processors, up to …

26%
for Elasticsearch®[4]

27%
for Apache Cassandra®[5]

49%
for Apache Spark™[6]

Cloud-native applications run better on **3rd Gen Intel® Xeon® Scalable processors** than on AMD EPYC™ processors, up to …

8%
for Elasticsearch®[7]

13%
for Apache Cassandra®[8]

12%
for Apache Spark™[9]

Market and Technology Trends

At Prowess, the point of benchmarking is translating lab results into actionable decisions that your business can implement in the real world. More than ever before, organizations need optimal performance from their cloud-native and web-based applications built on Java. As an IT decision maker, you are tasked with making decisions about application management and workload placement to achieve the desired performance outcomes. Before making consequential decisions, you want to be informed by evidence, such as workload benchmarking.

It's no secret that successful organizations count on cloud-native applications such as Elasticsearch, Apache Cassandra, and Apache Spark to process vast volumes of data coming from a variety of sources and packaged in different formats. They provision cloud-based resources to support social media and messaging services, Internet of Things (IoT) sensors, threat analysis and fraud-detection monitoring, user-activity tracking and analytics, confidential computing for medical and financial databases, streaming media user personalization, ecommerce search engines and purchase recommendations, artificial intelligence (AI)-driven language processing and translation, high-performance computing (HPC) for scientific research, and many other services.

Even with these robust cloud-based solutions at your disposal, getting the insights you need and delivering the services your customers demand can be a challenge. To meet your business goals, you need answers to best-practices questions, such as: How is cloud-native application performance affected by software version? What hardware configuration delivers optimal Java-based workload performance? Do these performance differences exist over a range of cloud-instance sizes?

To get some of these questions answered, Prowess designed benchmarking tests that would examine how cloud-native application performance is impacted by these key variables:

1. JDK version
2. Data workload
3. Processor type

We selected Amazon EC2 as the benchmarking environment because AWS is the most widely used cloud platform across the globe, with Amazon EC2 being one of AWS' fastest growing services.[10]

Measuring Workload Performance for JDK Version, Application, Processor Hardware, and Instance Size

Prowess engineers selected three Java-based workloads for benchmark testing, all of which require the installation of JDK. To examine the effects of the latest supported LTS release of JDK, we set up an Amazon EC2 testing environment using a 3rd Gen Intel Xeon Scalable processor with LTS JDK 8 or LTS JDK 11 as the baseline, depending on the benchmark being run. Next, we upgraded the Amazon EC2 instance on the same Java virtual machine (JVM) to the latest LTS JDK 11 or LTS JDK 17, depending on the workload. This helped identify performance improvements that could be gained from a simple Java upgrade.

To analyze cloud-native applications' performance, we ran workload benchmarks for three popular Java-based applications. Elasticsearch is an open-source search and analytics engine used to rapidly access and analyze huge volumes of data. Apache Cassandra is an open-source, scalable NoSQL database favored for its fast read-/write-operations on large volumes of data. The open-source Apache Spark framework is particularly well-suited for handling HPC and big data analytics, which process massive volumes of data, packaged in a variety of formats, and coming from multiple sources.

Our engineers selected the M6 instance for testing because this Amazon EC2 instance family is popular for running a diverse variety of general-purpose workloads. Also, their inherent stability makes M6 instances useful for comparing baseline and testing performance.[11] We tested three instance sizes—xlarge, 4xlarge, and 16xlarge—to facilitate the analysis of performance changes for small, medium, and large instance sizes.

To analyze workload performance for three of the most popular cloud-platform CPUs, we tested Intel Xeon Scalable, AMD EPYC, and AWS Graviton2 processors. Intel Xeon Scalable and AMD EPYC processors are designed for x86 data center architectures, and AWS Graviton2 processors for ARM® architectures. Table 1 summarizes the hardware, JDK versions, workloads, and instances that we tested.

Table 1 | Hardware (platform architecture), Java Development Kit (JDK®) versions, data workloads, and Amazon® Elastic Compute Cloud™ (Amazon EC2®) instance sizes used in testing

Hardware (architecture)	JDK® version	Workload (benchmark)	Instance size
3rd Gen Intel® Xeon® Scalable processor (x86)	JDK 11, JDK 17	Elasticsearch® (Rally)	m6i.xlarge m6i.4xlarge m6i.16xlarge
	JDK 8, JDK 11	Apache Cassandra® (Cassandra)* Apache Spark™ (HiBench)*	
Amazon Web Services® (AWS®) Graviton2 processor (ARM®)	JDK 17	Elasticsearch (Rally)	m6g.xlarge m6g.4xlarge m6g.16xlarge
	JDK 11	Apache Cassandra (Cassandra)* Apache Spark (HiBench)*	
AMD EPYC™ processor (x86)	JDK 17	Elasticsearch (Rally)	m6a.xlarge m6a.4xlarge m6a.16xlarge
	JDK 11	Apache Cassandra (Cassandra)* Apache Spark (HiBench)*	

*The Apache Cassandra data platform and the HiBench benchmark do not currently support JDK 17, so the control and testing workloads were run on JDK 8 and JDK 11.

Testing Assumptions and System Configurations

Prowess engineers established a number of conditions and assumptions to help ensure workload testing emulated enterprise-scale deployments and that the results could be applied to real-world conditions. Following network-security best practices, we deployed authentication and Transport Layer Security (TLS)/Secure Sockets Layer (SSL) encryption during testing. We used separate Amazon EC2 logins for individual sets of testing parameters. After testing, we powered down the cloud instances to avoid incurring compute costs that would inflate the results.

Our engineers chose to compare JDK 8, 11, and 17 because these are the LTS versions typically run in enterprise data centers.[12] Elasticsearch was tested on JDK 11 to establish baseline values, and then on JDK 17 for comparison. The Apache Cassandra data platform and the HiBench benchmark do not currently support JDK 17, so we ran these baseline and testing workloads on JDK 8 and JDK 11, respectively.[13]

We tested the smallest instance size first, with incrementally larger instance sizes tested next using the same JDK and hardware configurations. For example, Elasticsearch benchmark testing for JDK 11 running on a 3rd Gen Intel Xeon Scalable processor started with an M6i.xlarge instance. Using the same software and hardware configurations, we tested the benchmark on an M6i.4xlarge instance size next, followed by an M6i.16xlarge instance size. Table 2 shows the system configurations we used for the benchmark testing.

Table 2 | System configurations and benchmarks used for performance testing

	Intel® Xeon® Scalable processor	AMD EPYC™ processor	Amazon Web Services® (AWS®) Graviton2 processor
Architecture	x86	x86	ARM®
CPU	1 x Intel Xeon Platinum 8375C processor	1 x AMD EPYC 7R13 processor	1 x AWS Graviton2 processor
Frequency (Base/SCT/MCT)	2.9 GHz/3.5 GHz turbo	2.9 GHz/3.5 GHz turbo	2.5 GHz
Instance sizes: vCPUs	m6.xlarge: 4 m6.4xlarge: 16 m6.16xlarge: 64		
Instance sizes: cores/threads	m6i.xlarge: 8/16 m6i.4xlarge: 128/256 m6i.16xlarge: 2,048/4,096	m6a.xlarge: 8/128 m6a.4xlarge: 32/512 m6a.16xlarge: 128/2,048	m6g.xlarge: 4/1 m6g.4xlarge: 16/1 m6g.16xlarge 64/1
Storage controller	Amazon® Elastic Block Store (EBS)		
Disk	/dev/nvme0		
Number of disks	1 x 200 GB VHD		
Operating system (OS) version	Ubuntu® 20.04		
OS kernel	5.15.0-1011-aws	5.13.0-1029-aws	5.13.0-1029-aws
Amazon® Elastic Compute Cloud™ (Amazon EC2®) locations	Oregon	Northern California	Northern California
JDK® versions	JDK 8 version "1.8.0_312" JDK 11.0.15 2022-04-19 JDK 17.0.3 2022-04-19	JDK 11.0.15 2022-04-19 JDK 17.0.3 2022-04-19	
Workloads (benchmarks)	Elasticsearch® (Rally 2.6.0) Apache Cassandra® (Cassandra 3.11.13) Apache Spark™ (HiBench 7.1.1)		
Security	Encryption, TLS/SSL, authentication		

Imagine What You Could You Do with Faster Cloud Performance

In the war for market share, latency kills. This is what drives top social-media sites, news outlets, and other cloud-native companies to rely on the Elasticsearch search engine to sort through massive volumes of data and deliver answers in milliseconds.[14,15] These companies understand that users are notoriously impatient and will switch to other tasks—or worse, another app—if they do not get their answers delivered in near real time. Based on benchmark testing, upgrading from JDK 11 to JDK 17 will deliver up to 13 percent faster search query results for cloud instances powered by 3rd Gen Intel Xeon Scalable processors.[1]

Apache Cassandra is a NoSQL database that delivers high-speed data storage and access,[16] which is why streaming-media companies count on Apache Cassandra in the cloud to meet customers' strict SLAs. They need a database that can meet strict media-delivery requirements, such as latency of no more than a few seconds and failure of one-thousandth of a percent.[17] Based on Prowess testing, Instagram users could experience smoother video streaming that runs up to 42 percent faster using JDK 11 on cloud instances powered by 3rd Gen Intel Xeon Scalable processors, compared to AWS Graviton2 processors, and up to 50 percent faster compared to AMD EPYC processors.[18]

Apache Spark is deployed in life-sciences research to handle complex data analytics, such as high-throughput genome sequencing, using a hybrid cloud. For example, an Apache Spark pipeline can reduce genomic analysis time by more than 4.5x.[19] With 3rd Gen Intel Xeon Scalable processors, researchers could potentially reduce pipeline latency up to an additional 49 percent compared to using AWS Graviton2 processors, and up to 27 percent compared to using AMD EPYC processors.[20]

Benchmarking Results Reveal Standouts in Amazon EC2 Performance

For this study, Prowess engineers proposed the following hypotheses:

1. Upgrading to the latest JDK LTS version would improve workload performance for the following cloud-native applications: Elasticsearch, Apache Cassandra, and Apache Spark.
2. Running on the latest JDK LTS versions, cloud instances powered by Intel Xeon Scalable processors would outperform cloud instances powered by AWS Graviton2 and AMD EPYC processors.

Prowess engineers used best-practices application-level testing to help surface underlying architecture, software, and infrastructure limitations that might impact cloud performance. To evaluate cloud-native applications' performance after upgrading the JDK version, we compared the performance of Elasticsearch, Apache Cassandra, and Apache Spark workloads on Intel Xeon Scalable processor–powered instances. We compared LTS releases of JDK 8 (baseline) and JDK 11 (testing) for Elasticsearch workloads and JDK 11 (baseline) and JDK 17 (testing) for Apache Cassandra and Apache Spark workloads. To analyze the comparative workload performance of other mainstream data-center CPUs, our engineers tested Intel Xeon Scalable processors against AMD EPYC and AWS Graviton2 processors.

Benchmark testing confirmed our engineers' original hypothesis. Overall, the workloads showed improved performance after upgrading the JDK versions. Workloads on instances powered by Intel Xeon Scalable processors performed consistently better than instances powered by AMD EPYC or AWS Graviton2 processors.

Upgrading to the Latest JDK LTS Version Improves Cloud-Based Workload Performance

Elasticsearch, Apache Cassandra, and Apache Spark benchmarking results supported Prowess engineers' first hypothesis—upgrading to the latest JDK LTS release improves workload performance across a variety of Java-based data platforms. For the purposes of this technical research report, the charts present 4xlarge instance-size results. Out of the nine workload benchmarks run in total (three workloads x three instance sizes), only one workload test showed a minor performance dip after the JDK LTS upgrade.[21] Raw data values for all workloads and instance sizes are included in the endnotes at the end of this report.

As illustrated in Figure 1, our test results reveal that after upgrading to the latest LTS JDK release:

- Elasticsearch workloads performed up to 10 percent better.[1]
- Apache Cassandra workloads performed up to 4 percent better.[2]
- Apache Spark workloads performed up to 19 percent faster.[3]

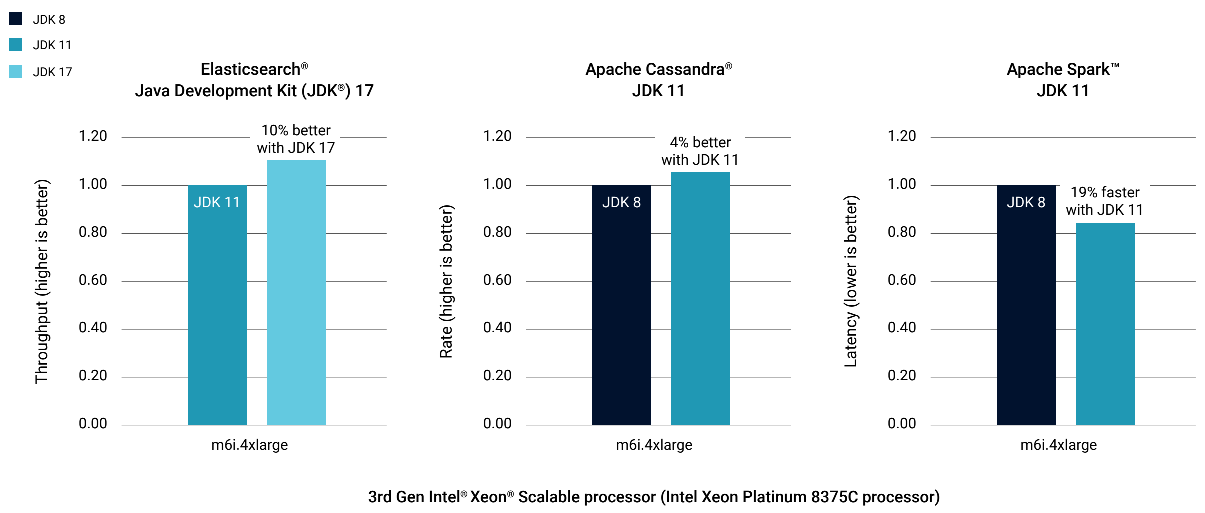

Figure 1. The performance of Elasticsearch®, Apache Cassandra®, and Apache Spark™ workloads improved after upgrading to the latest LTS Java Development Kit (JDK®) release

The Difference Is Clear … Cloud-Native Applications Running on Instances with Intel® Processors Significantly Outperform Those on AWS Graviton® and AMD EPYC™ Processors

Testing results also supported Prowess engineers' second hypothesis—with the latest JDK LTS versions, cloud instances powered by Intel Xeon Scalable processors outperformed cloud instances powered by AWS Graviton2 and AMD EPYC processors.

As with the JDK LTS upgrade results, our charts and discussion here present 4xlarge instance size results. Out of 18 benchmarks run in total (three workloads x three instance sizes x two CPU-versus-CPU comparisons), the results revealed that all workloads running on Intel Xeon Scalable processor–powered instances consistently outperformed workloads on AWS Graviton2 processor–powered instances and AMD EPYC processor–powered instances. Raw data values for all workloads and instance sizes are included in the endnotes at the end of this report. These results suggest that optimizations for Intel® architecture included in the latest JDK releases could be providing Intel Xeon Scalable processors with additional benefits that might not be accessible to other CPUs.

As illustrated in Figure 2, our test results reveal that with the latest LTS JDK release, Intel Xeon Scalable processor–powered instances outperformed AWS Graviton2 processor–powered instances:

- Elasticsearch workloads performed up to 26 percent faster.[4]
- Apache Cassandra workloads performed up to 27 percent better.[5]
- Apache Spark workloads performed up to 49 percent better.[6]

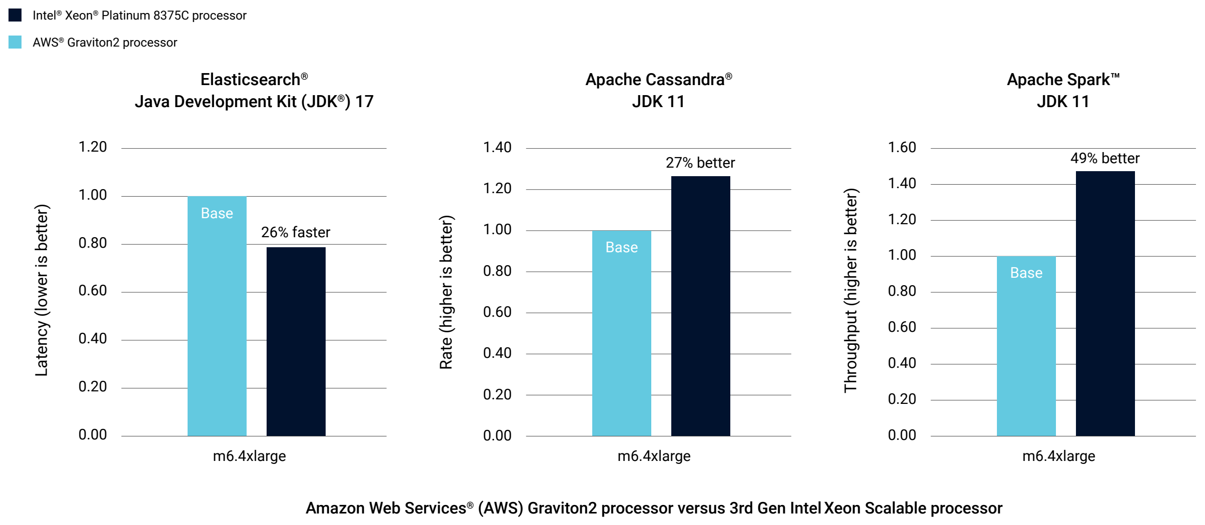

Figure 2. Elasticsearch®, Apache Cassandra®, and Apache Spark™ workloads run faster and better on M6i instances powered by 3rd Gen Intel® Xeon® Scalable processors than on M6g instances powered by Amazon Web Services® (AWS®) Graviton2 processors

As illustrated in Figure 3, our test results reveal that with the latest LTS JDK release, Intel Xeon Scalable processor–powered instances outperformed AMD EPYC processor–powered instances:

- Elasticsearch workloads performed up to 8 percent faster.[7]
- Apache Cassandra workloads performed up to 13 percent faster.[8]
- Apache Spark workloads performed up to 12 percent better.[9]

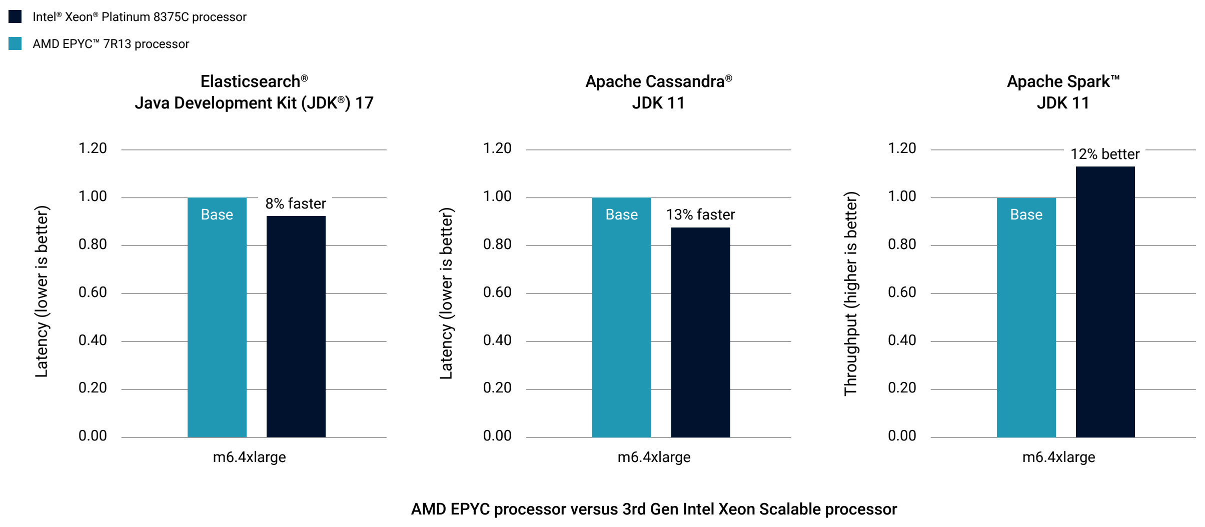

Figure 3. Elasticsearch®, Apache Cassandra®, and Apache Spark™ workloads run faster and better on M6i instances powered by 3rd Gen Intel® Xeon® Scalable processors than on M6a instances powered by AMD EPYC™ processors

A Compelling Case for Running Cloud-Native Applications on the Latest JDK Version

Prowess benchmark testing of Elasticsearch, Apache Cassandra, and Apache Spark workloads confirmed the original hypothesis, that web-based applications will likely run best on the latest supported JDK LTS version. The test results also indicate that general-computing cloud instances will perform better when powered by Intel Xeon Scalable processors than when powered by AWS Graviton2 and AMD EPYC processors. These results translate to a faster, more responsive user experience for your customers.

Prowess best-practices guidance for cloud-instance deployments recommends using application-level benchmarking results as a reliable starting point, and then provisioning the instance size to meet the computing, networking, or memory requirements of your specific workloads, datasets, or virtual machines (VMs).

Learn More

Learn more about how Intel Xeon Scalable processors can unlock the potential of your cloud computing.

[1] **Elasticsearch®** results for the 4xlarge instance size. xlarge: Up to 13 percent performance improvement as measured by ES Rally HTTP Logs Median Throughput benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor. JDK® 11: m6i.xlarge = 81,975.0700. JDK 17: m6i.xlarge = 92,789.0700. 4xlarge: Up to 10 percent performance improvement as measured by ES Rally HTTP Logs Median Throughput benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor. JDK 11: m6i.4xlarge = 265,696.7000. JDK 17: m6i.4xlarge = 292,641.6400. 16xlarge: Up to 13 percent performance improvement as measured by ES Rally HTTP Logs Median Throughput benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor. JDK 11: m6i.16xlarge = 329,257.5000. JDK 17: m6i.16xlarge = 372,954.5100. See Table 2 for full system configuration and benchmark testing information.

[2] **Apache Cassandra®** results for the 4xlarge instance size. xlarge: Up to 5 percent performance improvement as measured by Cassandra Read-Op Rate benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor. JDK® 8: m6i.xlarge = 26,793.0000. JDK 11: m6i.xlarge = 28,145.0000. 4xlarge: Up to 4 percent performance improvement as measured by Cassandra Read-Op Rate benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor. JDK 8: m6i.4xlarge = 54,597.0000. JDK 11: m6i.4xlarge = 56,922.0000. 16xlarge: Up to 6 percent performance regression as measured by Cassandra Read-Op Rate benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor. JDK 8: m6i.16xlarge = 107,294.0000. JDK 11: m6i.16xlarge = 101,108.0000. See Table 2 for full system configuration and benchmark testing information.

[3] **Apache Spark™** results for the 4xlarge instance size. xlarge: Up to 4 percent performance improvement as measured by HiBench Spark Duration Linear Regression benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor. JDK® 8: m6i.xlarge = 188.9880. JDK 11: m6i.xlarge = 182.2210. 4xlarge: Up to 19 percent performance improvement as measured by HiBench Spark Duration Linear Regression benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor. JDK 8: m6i.4xlarge = 66.6430. JDK 11: m6i.4xlarge = 56.0280. 16xlarge: Up to 112 percent performance improvement as measured by HiBench Spark Duration Linear Regression benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor. JDK 8: m6i.16xlarge = 65.5560. JDK 11: m6i.16xlarge = 31.4330. See Table 2 for full system configuration and benchmark testing information.

[4] **Elasticsearch®** results for the 4xlarge instance size. xlarge: Up to 4 percent performance improvement as measured by ES Rally NYC Taxi 50th Percentile Latency benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an Amazon Web Services® (AWS®) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK® 11): m6i.xlarge = 1,617.5520. AWS Graviton2 (JDK 11): m6g.xlarge = 1,672.1400. 4xlarge: Up to 26 percent performance improvement as measured by ES Rally NYC Taxi 50th Percentile Latency benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an Amazon Web Services (AWS) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.4xlarge = 434.6317. AWS Graviton2 (JDK 11): m6g.4xlarge = 546.8501. 16xlarge: Up to 61 percent performance improvement as measured by ES Rally NYC Taxi 50th Percentile Latency benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an Amazon Web Services (AWS) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.16xlarge = 347.1408. AWS Graviton2 (JDK 11): m6g.16xlarge = 551.3300. See Table 2 for full system configuration and benchmark testing information.

[5] **Apache Cassandra®** results for the 4xlarge instance size. xlarge: Up to 42 percent performance improvement as measured by Cassandra Write-Op Rate benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an Amazon Web Services® (AWS®) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK® 11): m6i.xlarge = 21,900.0000. AWS Graviton2 (JDK 11): m6g.xlarge = 15,377.0000. 4xlarge: Up to 27 percent performance improvement as measured by Cassandra Write-Op Rate benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an Amazon Web Services (AWS) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.4xlarge = 51,733.0000. AWS Graviton2 (JDK 11): m6g.4xlarge = 40,446.00. 16xlarge: Up to 31 percent performance improvement as measured by Cassandra Write-Op Rate benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an Amazon Web Services (AWS) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.16xlarge = 92,840.0000. AWS Graviton2 (JDK 11): m6g.16xlarge = 70,622.00. See Table 2 for full system configuration and benchmark testing information.

[6] **Apache Spark™** results for the 4xlarge instance size. xlarge: Up to 16 percent performance improvement as measured by HiBench Spark Throughput K-means benchmark testing in August 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an Amazon Web Services® (AWS®) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK® 11): m6i.xlarge = 101,090.0000. AWS Graviton2 (JDK 11): m6g.xlarge = 87,029.00. 4xlarge: Up to 49 percent performance improvement as measured by HiBench Spark Throughput K-means benchmark testing in August 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an Amazon Web Services (AWS) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.4xlarge = 154,470.0000. AWS Graviton2 (JDK 11): m6g.4xlarge = 103,280.00. 16xlarge: Up to 38 percent performance improvement as measured by HiBench Spark Throughput K-means benchmark testing in August 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an Amazon Web Services (AWS) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.16xlarge = 148,546.0000. AWS Graviton2 (JDK 11): m6g.16xlarge = 106,937.00. See Table 2 for full system configuration and benchmark testing information.

[7] **Elasticsearch®** results for the 4xlarge instance size. xlarge: Up to 7 percent performance improvement as measured by ES Rally HTTP Logs Median Cumulative Indexing Time Across Primary Shards benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an AMD EPYC™ 7R13 processor. Intel Xeon Platinum 8375C processor (JDK® 11): m6i.xlarge = 1.1257. AMD EPYC 7R13 processor (JDK 11): m6a.xlarge = 1.1980. 4xlarge: Up to 8 percent performance improvement as measured by ES Rally HTTP Logs Median Cumulative Indexing Time Across Primary Shards benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.4xlarge = 1.3513. AMD EPYC 7R13 processor (JDK 11): m6a.4xlarge = 1.4599. 16xlarge: Up to 19 percent performance improvement as measured by ES Rally HTTP Logs Median Cumulative Indexing Time Across Primary Shards benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.16xlarge = 1.1672. AMD EPYC 7R13 processor (JDK 11): m6a.16xlarge = 1.3838. See Table 2 for full system configuration and benchmark testing information.

[8] **Apache Cassandra®** results for the 4xlarge instance size. xlarge: Up to 36 percent performance improvement as measured by Cassandra Write Latency Median benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an AMD EPYC™ 7R13 processor. Intel Xeon Platinum 8375C (JDK® 11): m6i.xlarge = 5.1000. AMD EPYC 7R13 (JDK 11): m6a.xlarge = 6.90. 4xlarge: Up to 13 percent performance improvement as measured by Cassandra Write Latency Median benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.4xlarge = 1.5000. AMD EPYC 7R13 processor (JDK 11): m6a.xlarge = 1.70. 16xlarge: Up to 100 percent performance improvement as measured by Cassandra Write Latency Median benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.16xlarge = 0.6000. AMD EPYC 7R13 processor (JDK 11): m6a.16xlarge = 1.20. See Table 2 for full system configuration and benchmark testing information.

[9] **Apache Spark™** results for the 4xlarge instance size. xlarge: Up to 27 percent performance improvement as measured by HiBench Spark Throughput K-means benchmark testing in August 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an AMD EPYC™ 7R13 processor. Intel Xeon Platinum 8375C processor (JDK® 11): m6i.xlarge = 101,090.0000. AMD EPYC 7R13 processor (JDK 11): m6a.xlarge = 79,072.00. 4xlarge: Up to 12 percent performance improvement as measured by HiBench Spark Throughput K-means benchmark testing in August 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.4xlarge = 154,470.0000. AMD EPYC 7R13 processor (JDK 11): m6a.4xlarge = 137,211.00. 16xlarge: Up to 3 percent performance improvement as measured by HiBench Spark Throughput K-means benchmark testing in August 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.16xlarge = 148,546.0000. AMD EPYC 7R13 processor (JDK 11): m6a.16xlarge = 143,000.00. See Table 2 for full system configuration and benchmark testing information.

[10] MindMajix. "Top 30 AWS Services List in 2022." https://mindmajix.com/top-aws-services.

[11] Nakivo. "The Definitive Guide to AWS EC2 Instance Types." www.nakivo.com/blog/the-definitive-guide-to-aws-ec2-instance-types/.

[12] Apache Cassandra. "Cassandra Documentation: Support for Java 11." https://cassandra.apache.org/doc/latest/cassandra/new/java11.html.

[13] GitHub. "test/cql-pytest/run-cassandra doesn't work with Java 17 #10946." July 2022. https://github.com/scylladb/scylladb/issues/10946. And: GitHub. "JDK11: HiBench to support JDK11, >=Spark2.4 and >=scala2.12.4 #567." February 2019. https://github.com/Intel-bigdata/HiBench/issues/567.

[14] ZDNet. "Elastic, search company for Uber and Tinder, nearly doubles in IPO." October 2018. www.zdnet.com/article/elastic-search-company-for-uber-and-tinder-nearly-doubles-in-ipo/.

[15] Sematext Group. "Elasticsearch Tutorial: A complete guide to getting started with the basic concepts: what it is, how it works, and what it's used for." https://sematext.com/guides/elasticsearch/#what-is-elasticsearch-used-for-applications-examples.

[16] Ubuntu. "What is Cassandra and why are big tech companies using it?" May 2020. https://ubuntu.com/blog/apache-cassandra-top-benefits.

[17] Instagram Engineering. "Open-sourcing a 10x reduction in Apache Cassandra tail latency." March 2018. https://instagram-engineering.com/open-sourcing-a-10x-reduction-in-apache-cassandra-tail-latency-d64f86b43589.

[18] Up to 42 percent performance improvement as measured by Cassandra Write-Op rate benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an AMD EPYC™ 7R13 processor. Intel Xeon Platinum 8375C processor (JDK® 11): m6i.xlarge = 21,900.0000. Amazon Web Services® (AWS®) Graviton2 (JDK 11): m6g.xlarge = 15,377.0000. Up to 50 percent performance improvement as measured by Cassandra Write Latency Median benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.16xlarge = 0.6000. AMD EPYC 7R13 (JDK 11): m6a.16xlarge = 1.2000. Up to 100 percent performance improvement as measured by Cassandra Write Latency Median benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and an AMD EPYC 7R13 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.4xlarge = 0.6000. AMD EPYC 7R13 (JDK 11): m6a.4xlarge = 1.2000. See Table 2 for full system configuration and benchmark testing information.

[19] Data Centre Dynamics. "Cray announces Urika-GX, a supercomputing platform for big data." May 2016. www.datacenterdynamics.com/en/news/cray-announces-urika-gx-a-supercomputing-platform-for-big-data/.

[20] Up to 49 percent performance improvement as measured by HiBench Spark Throughput k-means benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor and an Amazon Web Services® (AWS®) Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK® 11): m6i.4xlarge = 154,470.0000. AWS Graviton2 processor (JDK 11): m6g.4xlarge = 103,280.0000. Up to 27 percent performance improvement as measured by HiBench Spark Throughput k-means benchmark testing in June 2022 of a 3rd Gen Intel Xeon Platinum 8375C processor and AWS Graviton2 processor. Intel Xeon Platinum 8375C processor (JDK 11): m6i.xlarge = 101,090.0000. AMD EPYC™ 7R13 (JDK 11): m6a.xlarge = 79,072.0000. See Table 2 for full system configuration and benchmark testing information.

[21] **Apache Cassandra®** results for the 16xlarge instance size. Up to 6 percent performance regression as measured by Cassandra Read-Op Rate benchmark testing in June 2022 of a 3rd Gen Intel® Xeon® Platinum 8375C processor. JDK® 8: m6i.16xlarge = 107,294.0000. JDK 11: m6i.16xlarge = 101,108.0000. See Table 2 for full system configuration and benchmark testing information.

PROWESS