NVMe™ SSDs Future-proof Apache Cassandra®

Get More Insight from Datasets Too Large to Fit into Memory

Overview

When we scale a database—either locally or in the cloud—performance[1] is imperative. Without massive performance, a massive-scale database is little more than an active archive.

When an entire data set is small and fits into memory (DRAM), performance is straight-forward and storage system capability is less important. However, with immense data growth, a dwindling percentage of data affordably fits into memory.

By building with SSDs, we can future-proof Apache Cassandra deployments to perform soundly as active data sets grow, extending well beyond memory capacity.

Combined with the constant demand for faster and more detailed analytics, we have arrived at a data-driven crossroads: We need high performance, high capacity and affordability.

Cassandra combined with NVMe SSDs can help.

Cassandra's ability to support massive scaling, combined with multiterabyte, high IOPS NVMe SSDs, builds high-capacity NoSQL platforms offering extreme capacity, extreme agility and extreme capability.

This technical brief highlights the performance advantages we measured when we compared two 4-node Cassandra clusters: one built using legacy hard disk drives (HDDs); the second build using NVMe SSDs. We also explore some implications of these results.

Due to the broad range of Cassandra deployments, we tested multiple workloads and multiple thread counts. You may find some results more relevant than others for your deployment.

Fast Facts

- A four-node cluster using a single NVMe SSD eclipsed the capability of a multidrive legacy (HDD) four-node cluster across multiple workloads and thread counts
- A single NVMe SSD per node configuration measured up to 31X better performance, with more consistent, lower latency

NVMe SSDs Meet Growing Demands

When we built Cassandra nodes with legacy HDD storage, we scaled out by adding more nodes to the cluster. We scaled up by upgrading to larger drives. Sometimes we did both.

Adding more legacy nodes was effective (to a point), but it quickly became unwieldy. We gained capacity and a bit more performance, but as we added to the clusters, they became larger and more complex, consuming more rack space and support resources.

Upgrading to larger HDDs was somewhat effective (also to a point) since we got more capacity per node and more capacity per cluster, but these upgrades rarely augmented cluster performance.

With both techniques, performance stagnated while demand grew.

High capacity, lightning-quick NVMe SSDs are changing the design rules. With single SSD capacities measured in terabytes (TB), throughput in gigabytes per second (GB/s) and IOPS in hundreds of thousands[2], high-capacity NVMe SSDs enable new design opportunities and performance thresholds.

We used the Yahoo! Cloud Serving Benchmark (YCSB) workloads A–D and F[3] to compare two 4-node Cassandra test clusters: one built with NVMe SSDs and the other built with multiple legacy HDDs.

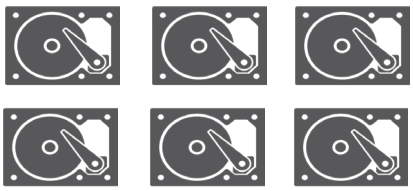
Note: Due to the broad range of Cassandra deployments, we tested multiple thread counts from 48 to 480. See the How We Tested section for details.

NVMe Clusters Build Capacity and Results

As you plan your next high-capacity, high-demand Cassandra cluster, NVME SSDs can support amazing capacity and provide compelling results.

Using a single NVMe SSD, each node in our SSD test cluster stores about 7.68TB. With six 15K RPM 300GB drives (RAID 0), our HDD test cluster stores about 1.8 TB per node.

SSD Test Cluster: One 7.68GB SSD with NVMe per node (4 cluster nodes)



Legacy Test Cluster: Six 300GB 15K RPM HDDs, RAID 0 poer node (4 cluster nodes)

With the same number of nodes and a single SSD in each node, the NVMe SSD test cluster offers a 4X capacity increase. We also measured a tremendous increase in performance over all the workloads and thread counts tested, ranging from a low of about 2X to a high of 31X, along with lower and more consistent latency.

Figure 1 shows YCSB performance for each configuration.

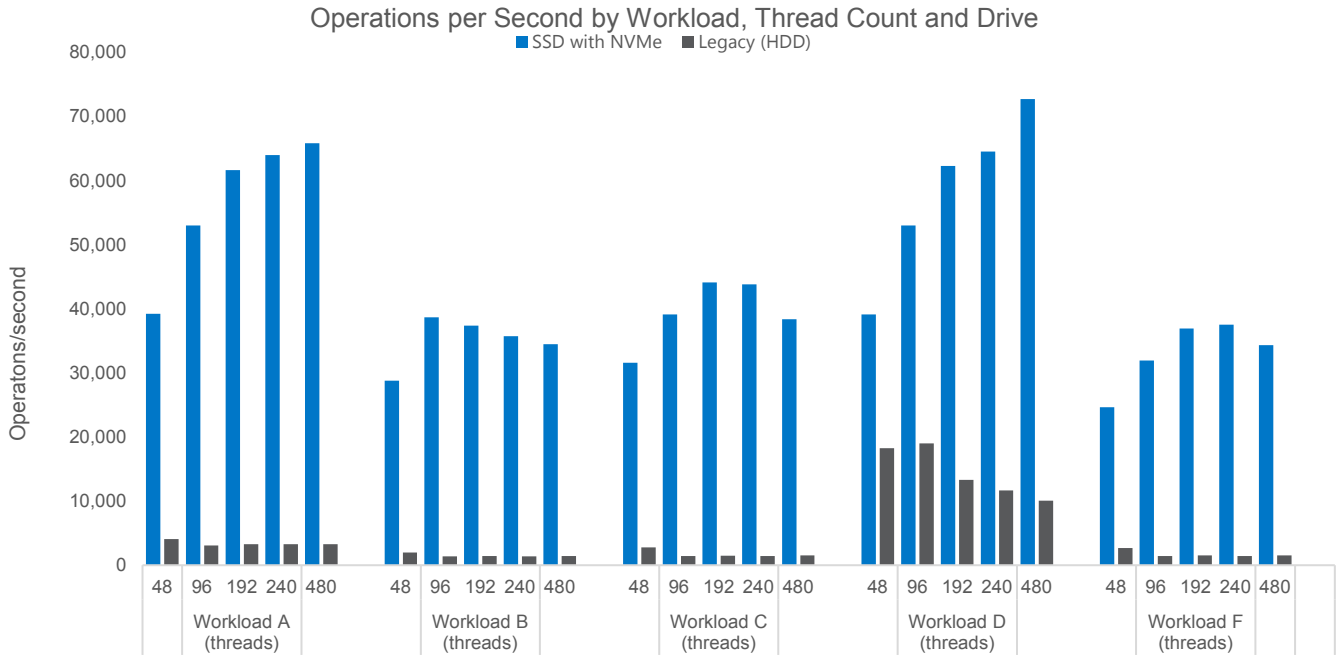


Figure 1: Relative Performance

NVMe Clusters Provide More Consistent Read Response

Since many Cassandra deployments rely heavily on fast, consistent read responses, we compared the 99th percentile read response times for each test cluster, workload and thread count. Figure 2 shows the results for each configuration.

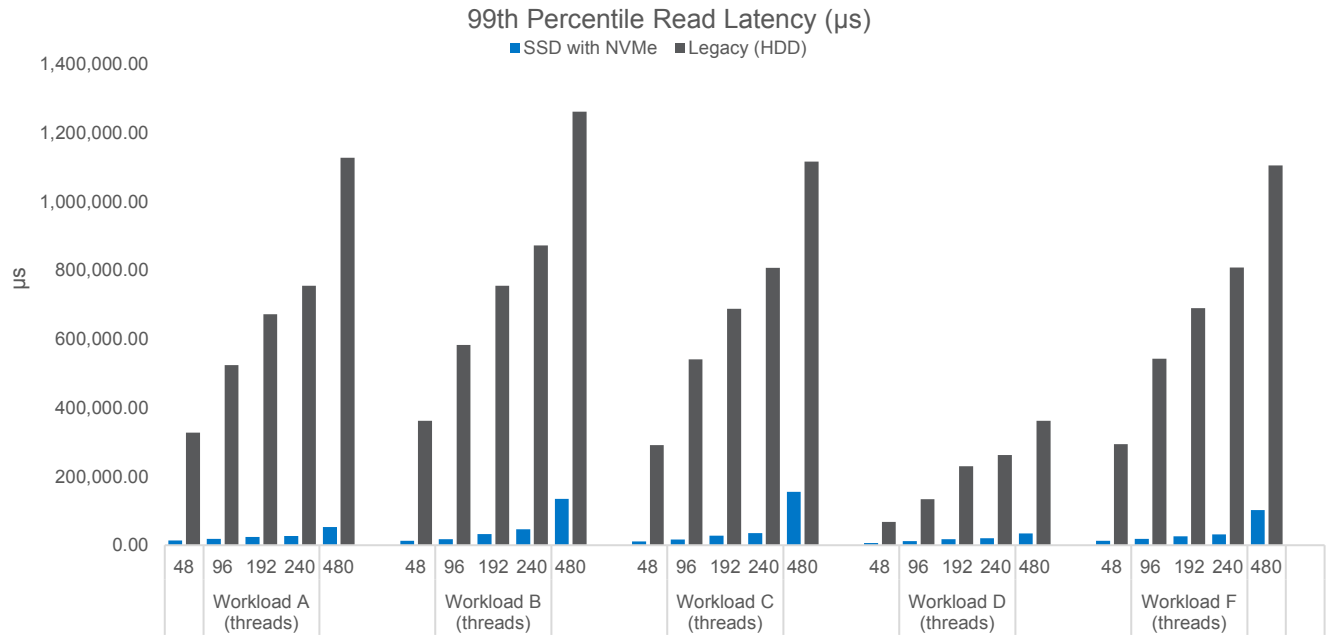


Figure 2: Relative Read Responsiveness

The Bottom Line

High-capacity, high-performance NVMe SSDs can produce amazing results with Cassandra. Whether you are scaling your local or cloud-based Cassandra deployment for higher performance or faster, more consistent read responses, NVMe SSDs are a great option.

We tested two clusters for database performance and read responsiveness across multiple workloads and thread counts. We built a legacy cluster using six 300GB 15K RPM HDDs (RAID 0) in each node and another cluster using a single 7.68TB NVMe SSD in each node.

The results were amazing.

The single SSD per node test cluster showed a tremendous increase in performance over all the workloads and thread counts tested, ranging from a low of about 2X up to a high of 31X. We also found that the SSD-based cluster read responses were much faster with far greater consistency despite using only one NVMe SSD in each node.

We expect great performance when our data set fits into memory, but immense data growth means that smaller and smaller portions of that data affordably fit into memory.

We are at a crossroads. Our demands drive us toward higher performance, and data growth drives us toward affordable capacity. When we combine these, the answer is clear: NVMe SSDs deliver Cassandra performance and capacity that's more approachable.

1. We use the terms database operations per second (OPS) and performance interchangeably in this paper.
2. Capacity, GB/s and IOPS vary by SSD. This paper focuses on our 7.68TB U.2 9200. Other NVMe SSD models and/or capacities may give different results.
3. We did not test YCSB workload E because it is not universally supported.

Note: We tested with Apache Cassandra Community Edition 3.11.1. Each node was equipped with 2x Intel Xeon E5-2690 v3 12 core processors and 256GB RAM.

How We Tested

Table 1 shows the tested configurations, types of storage devices used, the number and capacity of each as well as the number of nodes in each Cassandra test cluster. Table 2 shows the hardware and software configuration parameters used.

Configuration	Drive Type	Drives per Node	Capacity per Node	Nodes per Cluster
SSD with NVMe	7.68TB ECO	1	7.68TB	4
Legacy	300GB, 15K RPM	6	1.8TB	4

Table 1: Tested Configuration Capacities

Component	Test Cluster	Configuration/Description
Controller	NVMe SSD	Not used for database storage
	Legacy	Broadcom SAS 9300-8i (HBA)
Database Storage	NVMe SSD	1x Micron 9200 ECO 7.68TB 2.5" SSD with NVMe
	Legacy	6x 15K RPM 300GB HDD RAID 0 (mdadm)
OS Settings for Cassandra	NVMe SSD	**/etc/security/limits.d/cassandra.conf:** cassandra - memlock unlimited cassandra - nofile 100000 cassandra - nproc 32768 cassandra - as unlimited **/etc/sysctl.conf:** vm.max_map_count = 131072 **/etc/security/limits.d/20-nproc.conf:** * - nproc 32768
	Legacy	

Table 2: Configuration Parameters

Our test methodology approximates real-world deployments and uses for a Cassandra database. Although the test configuration is relatively small (four nodes in each cluster), Cassandra's scaling technology means these results are also relevant to larger deployments.

- Four nodes host the database.
- The replication factor for the database was set to 3 (there are three copies of the data and the cluster can sustain the loss of two data nodes and continue to function).

The database is initially created by utilizing YCSB workload A's load parameter, which generated a dataset of approximately 1.6TB, far exceeding available DRAM (ensuring we measure storage system IO). The database is then backed up to a separate location on the server for quick reload of data between test runs. For each configuration under test, the database was restored from this backup, starting every test from a consistent state.

Table 3 shows the percentage of data owned by each of the four nodes.

Node	Capacity	Tokens	Percent Owned
Node01	368.81GB	256	74.1%
Node02	362.62GB	256	72.9%
Node03	389.00GB	256	78.1%
Node04	373.42GB	256	74.9%

Table 3: Data Distribution across Nodes

Table 4 shows the testing parameters used in the tested workloads.

Parameter	Value	Description
Threads	48, 96, 144, 240, 480	Database load
Field Count	10	1K record size (standard)
Record Count	500 million	Number of database records
Operation Count	50 million	Dataset size within database

Table 4: Test Parameters

Dim_stat was used to capture statistics on the server running Apache Cassandra. It captures IOStat, VMStat, mpstat, network load, processor load, and several other statistics. Dim_stat was configured to capture statistics on a 10-second interval.

Table 5 shows the IO profiles for tested YCSB workloads (additional details are available at YCSB Core Workloads).

Name	Type	IO Profile
A	Update heavy	50% Read, 50% Write
B	Read mostly	95% Read, 5% Write
C	Read only	100% Read, 0% Write
D	Read latest	95% Read, 5% Insert
F	Read-modify-write (R/M/W)	50% Read, 50% R/M/W

Table 5: Workloads