Technical Report

NetApp E-Series and Cassandra NoSQL Database

Stephen Carl, Mitch Blackburn, NetApp
April 2017 | TR-4584

Abstract

This technical report describes the integrated architecture of the NetApp® E-Series and Cassandra NoSQL database design. Optimized for node storage balance, reliability, performance, storage capacity, and density, this design employs the Cassandra clustered node model, with higher scalability. Separating storage from compute provides the ability to scale each separately, saving the cost of overprovisioning one or the other. In addition, this document summarizes the performance test results obtained from a Cassandra simulated client machine reference architecture using the I/O load generation tool Yahoo Client Server Benchmark (YCSB) in simulated customer workloads.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

NetApp E-Series enables NoSQL database environments such as Cassandra to maintain the highest levels of performance and uptime for workloads by providing advanced fault recovery features and easy in-service growth capabilities to meet ever-changing business requirements. By disaggregating storage from compute, you gain the ability to scale capacity and compute separately, saving the cost of overprovisioning one or the other.

The E-Series is designed to handle the most extreme application workloads with very low latency. Typical use cases include application acceleration; improving the response time of latency-sensitive applications; and improving the power, environmental, and capacity efficiency of overprovisioned environments. E-Series storage systems leverage the latest solid-state disk (SSD) and SAS drive technologies and are built on a long heritage of serving diverse workloads to provide superior business value and enterprise-class reliability.

NoSQL encompasses a wide variety of different database technologies that were developed in response to a rise in the volume of data stored about users, objects, and products; the frequency with which this data is accessed; and performance and processing needs. Relational databases, in contrast, were not designed to cope with the scale and agility challenges that face modern or third platform applications, nor were they built to take advantage of the inexpensive storage and processing power available today.

This technical report describes the integrated architecture of the NetApp E-Series and Cassandra NoSQL database design. This design employs the Cassandra clustered node model, with higher scalability. In addition, this document summarizes the performance test results obtained from a Cassandra simulated client machine reference architecture using the I/O load generation tool Yahoo Client Server Benchmark (YCSB) in simulated customer workloads. Two areas of E-Series E5600 performance are tested:

- All SSDs (equivalent to a NetApp E-Series EF560 all-flash array)
- E5600 configured with Dynamic Disk Pools (DDP) for enhanced recovery times when a drive fails (performance on failure)

2 Cassandra Overview

NoSQL databases, including wide column store types such as Apache Cassandra, are providing companies with analytics capabilities for faster data insights in real time, enabling quick response to customer data-related needs and requirements. Online applications focused on customer interaction while collecting customer data at any location require zero tolerance of downtime and data unavailability. Cassandra is a distributed and scalable database with enterprise search, security, integrated analytics, and in-memory capabilities. An open-source NoSQL platform, Apace Cassandra can handle requests for concurrent writes with low-latency response times in a distributed environment of users. Cassandra's distributed node architecture makes sure that data is accessible to users even in the event of hardware failures and offers robust support for clusters spanning multiple data centers, with asynchronous masterless replication of data.

2.1 Primary Use Cases

Cassandra is a general purpose nonrelational database that can be used for a variety of different applications and use cases. Where the Cassandra database excels over other options includes the following:

Messaging

Cassandra serves as the database backbone for mobile phone and messaging providers' applications. With an increase in mobile device adoption, storing, managing, and performing analyses on messaging systems that include email, chat, commenting, and user notifications need large scalability and availability capabilities, without sacrificing performance.

Fraud Detection

Businesses lose billions of dollars due to fraudulent activity. Storing, managing, and analyzing large datasets in real time to identify fraud are difficult and costly. Storing data patterns in Cassandra can detect fraudulent activity quickly, accurately, and without the complexity and expense that relational environments would require.

Product Catalog and Playlists

Cassandra is a popular database choice for many retailers that need durable shopping cart protection, fast product catalog input and lookups, and similar retail application support. Businesses that require handling retail/e-commerce transactions, managing a user's media playlist, or storing collections of user-selected and -curated items have performance, scalability, and availability implications.

Internet of Things

Cassandra is well suited for consuming large amounts of fast incoming data from devices, sensors, and similar mechanisms from a multitude of different geographic locations.

Recommendation and Personalization

Personalized content and recommendations to customers are currently a feature expected by most users. Cassandra can ingest, analyze, and provide analysis and recommendations to its customers, providing fast, low-cost, and scalable capabilities managing any volume of user activity and data.

Retail

The consumer retail environment is a varied and complex mixture of customer touchpoints and data collection opportunities for real-time interactions occurring both in store and online containing a unique perspective into consumer behavior. Cassandra offers a scalable solution enabling retailers to provide a user experience that not only allows a better connection with their customers, but also delivers consumer insight in ways that were unimaginable even a few years ago.

Digital Media

Media companies depend on Cassandra to build a strong connection with their customers and develop levels of user engagement. Building on Cassandra's highly available peer-to-peer architecture with fast writes and linear scale, digital media companies can build a better, more personalized, and seamless user experience for their consumers. Nodes can be added at a moment's notice to handle all levels of user demand, ready to scale for the latest releases.

Finance

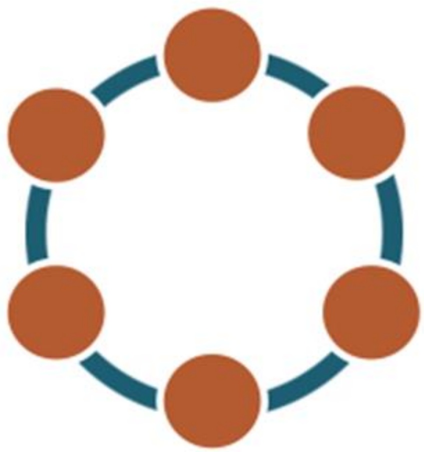
Cassandra's feature set makes it possible to offer the best in risk management and fraud detection while also making sure of a superior user experience, which are critical for the financial services industry. Whether your application is for banking, mobile payment services, market investments, or insurance, Cassandra delivers a natural advantage over legacy and other NoSQL database systems by maintaining performance at scale to handle often changing regulations and a user base that grows 24/7, with more accounts and devices being added every day.

2.2 Cassandra Architecture Overview

Cassandra is designed to handle big data workloads across multiple nodes with no single point of failure. Its architecture is based on the understanding that system and hardware failures can and do occur. Cassandra addresses the problem of failures by employing a peer-to-peer distributed system across homogeneous nodes where data is distributed among all nodes in the cluster. Each node frequently exchanges state information about itself and other nodes across the cluster using a peer-to-peer communication protocol.

Cassandra's architecture is responsible for its ability to scale, perform, and offer continuous uptime. Rather than using a legacy master-slave or a manual and difficult-to-maintain sharded design, Cassandra has a masterless "ring" architecture that is elegant, easy to set up, and easy to maintain, as shown in Figure 1. All nodes in the cluster play an identical role and communicate using a distributed, scalable protocol called "gossip." This distributed environment has no single point of failure and therefore can offer true continuous availability and uptime.

Figure 1) Cassandra ring architecture.

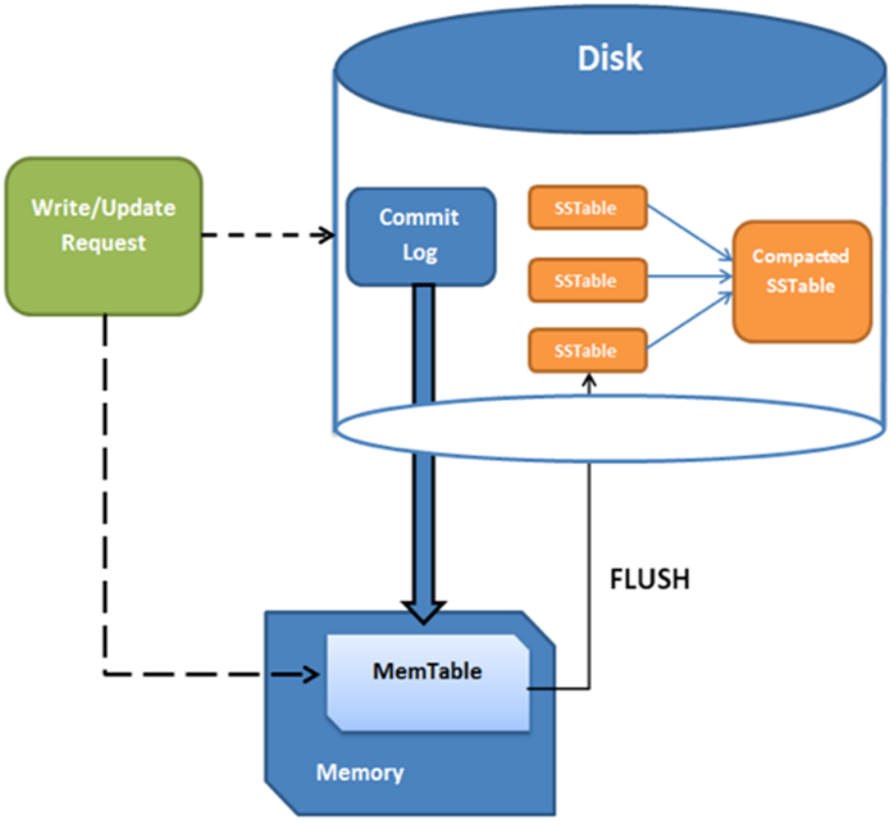


Cassandra replicates data to multiple nodes in a cluster, which enables reliability, continuous availability, and fast I/O operations. The number of data copies that are replicated is referred to as the replication factor. A replication factor of 1 means that there is only one copy of each row in a cluster; a replication factor of 3 means three copies of the data are stored across the cluster.

After a keyspace and its replication have been created, Cassandra automatically maintains that replication even when nodes are removed, are added, or fail.

Each Cassandra cluster node uses a sequentially written commit log to capture write activity to make sure of data availability. Data is distributed and indexed in an in-memory structure called a memtable, which resembles a writeback cache. When the memtable is full, the data is written to an immutable file on disk called an SSTable. Buffering writes in memory allows writes to always be a fully sequential operation of many megabytes of disk I/O happening simultaneously. This approach enables high write performance and durability. The write path diagram is shown in Figure 2.

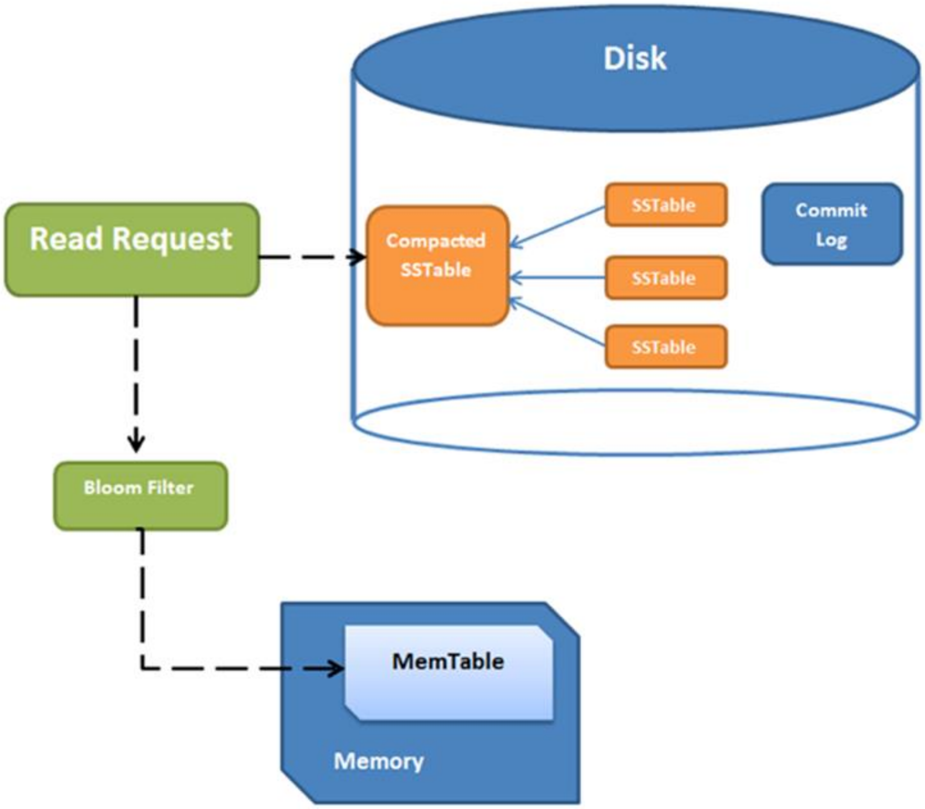
Figure 2) Cassandra write path.



More than one SSTable can exist for a single Cassandra logical data table. A process named compaction occurs periodically, combining multiple SSTables into one for faster read access.

For a read request, Cassandra consults an in-memory data structure called a Bloom filter that checks the probability of an SSTable having the needed data. The Bloom filter can tell very quickly whether the file probably has the needed data or certainly does not have it. If the answer is a tentative yes, Cassandra consults another layer of in-memory caches, then fetches the compressed data on disk. If the answer is no, Cassandra doesn't trouble with reading that SSTable at all and moves on to the next. The read path diagram is shown in Figure 3.

Figure 3) Cassandra read path.



For more information about Cassandra capabilities, go here.

3 NetApp E-Series Overview

The E-Series E5600 is an industry-leading storage system that delivers high input/output operations per second (IOPS) and bandwidth with consistently low latency to support the demanding performance and capacity needs of science and technology, simulation modeling, and decision support environments. In addition, the E5600 is equally capable of supporting primary transactional databases, general mixed workloads, and dedicated workloads such as video analytics in a highly efficient footprint with extreme simplicity, reliability, and scalability.

The E5600 provides the following benefits:

- Support for wide-ranging workloads and performance requirements
- Fully redundant I/O paths, advanced protection features, and proactive support monitoring and services for high levels of availability, integrity, and security
- Increased IOPS performance by up to 35% compared to the previous high-performance generation of E-Series products
- A level of performance, density, and economics that leads the industry
- Interface protocol flexibility to support FC host and iSCSI host workloads simultaneously
- Support for private and public cloud workloads behind virtualizers such as FlexArray®, Veeam Cloud Connect, and StorageGRID®

3.1 E-Series Hardware Overview

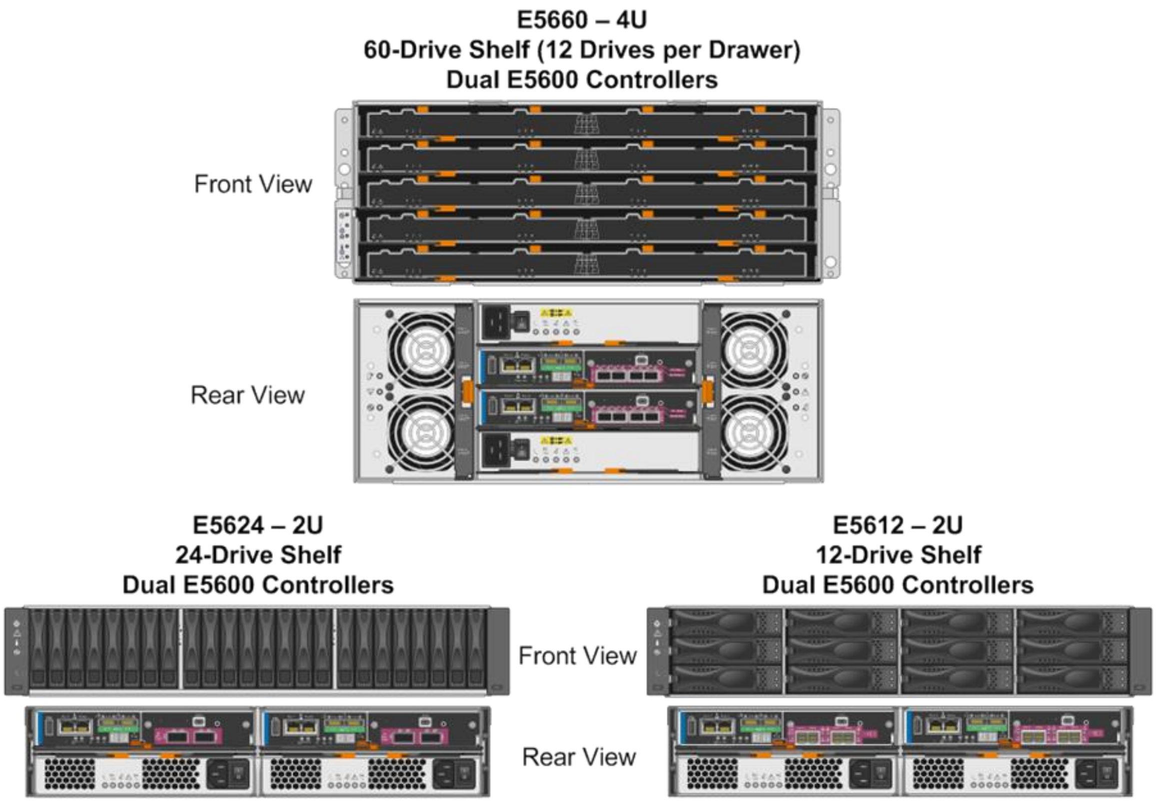
As shown in Table 1, the E5600 is available in three shelf options, which support both hard-disk drives (HDDs) and SSDs to meet a wide range of performance and application requirements.

Table 1) E5600 controller shelf and drive shelf models.

Controller Shelf Model	Drive Shelf Model	Number of Drives	Type of Drives
E5660	DE6600	60	2.5” and 3.5” SAS drives (HDDs and SSDs)
E5624	DE5600	24	2.5” SAS drives (HDDs and SSDs)
E5612	DE1600	12	3.5” SAS drives (HDDs only)

All three shelf options include dual-controller modules, dual power supplies, and dual fan units for redundancy (the 12-drive and 24-drive shelves have integrated power and fan modules). The shelves are sized to hold 60 drives, 24 drives, or 12 drives, as shown in Figure 4.

Figure 4) E5600 controller drive shelf options.



Each E5600 controller shelf includes two controllers, with each controller providing two Ethernet management ports for out-of-band management. The system also supports in-band management access and has two 6Gbps wide-port SAS drive expansion ports for redundant drive expansion paths. The E5600 controllers do not include built-in host ports, but must be ordered with one of the following host interface cards (HICs) installed in each controller:

Note: Both controllers in an E5600 array must be identically configured.

- 4-port 12Gb SAS HIC.
- 2-port 56Gb InfiniBand (IB) HIC. This HIC runs the iSCSI Extensions for RDMA (iSER) protocol as shipped, but it can be converted to SCSI RDMA Protocol (SRP) before initial use by applying a software feature pack in the field at no additional cost.
- 4-port optical HIC, which can be factory-configured as either 16Gb Fibre Channel or 10Gb iSCSI. A software feature pack can be applied in the field to change the host protocol of this HIC:
  - From FC to iSCSI
  - From iSCSI to FC
  - From either FC or iSCSI to FC-iSCSI split mode
  - From FC-iSCSI split mode back to FC or iSCSI

Note: In FC-iSCSI split mode, ports 1 and 2 operate as iSCSI, and ports 3 and 4 operate as FC.

3.2 SANtricity Software

E-Series systems are managed by the SANtricity® Storage Manager application. Simple to download and install, SANtricity Storage Manager provides an intuitive, wizard-led GUI as well as full support for a CLI. SANtricity Storage Manager can be installed on a Microsoft Windows, Solaris, or Linux operating system (OS) platform for out-of-band management of the storage array.

To create volume groups on the array, the first step when configuring SANtricity is to assign a redundant array of inexpensive disks (RAID) level. This assignment is then applied to the disks selected to form the volume group. The E5600 storage systems support RAID levels 0, 1, 3, 5, 6, and 10 or DDP. DDP was used for all configurations described in this document.

To simplify the storage provisioning, NetApp provides a SANtricity automatic configuration feature. The configuration wizard analyzes the available disk capacity on the array. It then selects disks that maximize array performance and fault tolerance while meeting capacity requirements, hot spares, and any other criteria specified in the wizard.

Dynamic Storage Functionality

From a management perspective, SANtricity offers a number of capabilities to ease the burden of storage management, including the following:

- New volumes can be created and are immediately available for use by connected servers.
- New RAID sets (volume groups) or disk pools can be created at any time from unused disk devices.
- Dynamic volume expansion allows capacity to be added to volumes online as needed.
- Dynamic capacity expansion allows disks to be added to volume groups and disk pools online to meet any new requirements for capacity or performance.
- Dynamic RAID migration allows the RAID level of a particular volume group to be modified online if new requirements dictate a change, for example, from RAID 10 to RAID 5.
- Flexible cache block and dynamic segment sizes enable optimized performance tuning based on a particular workload. Both items can also be modified online.
- Online controller firmware upgrades and drive firmware upgrades are possible.
- Path failover and load balancing (if applicable) between the host and the redundant storage controllers in the E5600 are provided. See the Multipath Drivers Guide for more information.

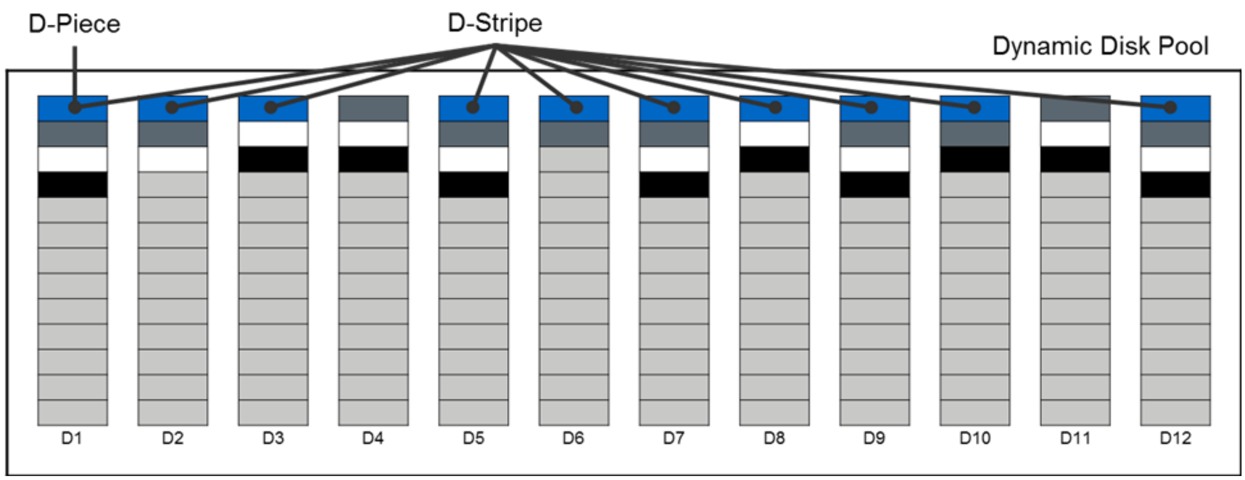
Dynamic Disk Pools

With seven patents pending, the DDP feature dynamically distributes data, spare capacity, and protection information across a pool of disk drives. These pools can range in size from a minimum of 11 drives to all the supported drives in a system. In addition to creating a single DDP, storage administrators can opt to mix traditional volume groups and DDP or even multiple DDPs, offering an unprecedented level of flexibility.

Dynamic Disk Pools are composed of several lower-level elements. The first of these is a D-piece. A D-piece consists of a contiguous 512MB section from a physical disk that contains 4,096 128KB segments. Within a pool, 10 D-pieces are selected using an intelligent optimization algorithm from selected drives within the pool. Together, the 10 associated D-pieces are considered a D-stripe, which is 4GB of usable capacity in size. Within the D-stripe, the contents are similar to a RAID 6 8+2 scenario. There, 8 of the underlying segments potentially contain user data, 1 segment contains parity (P) information calculated from the user data segments, and 1 segment contains the Q value as defined by RAID 6.

Volumes are then created from an aggregation of multiple 4GB D-stripes as required to satisfy the defined volume size up to the maximum allowable volume size within a DDP. Figure 5 shows the relationship between these data structures.

Figure 5) Dynamic Disk Pool components.

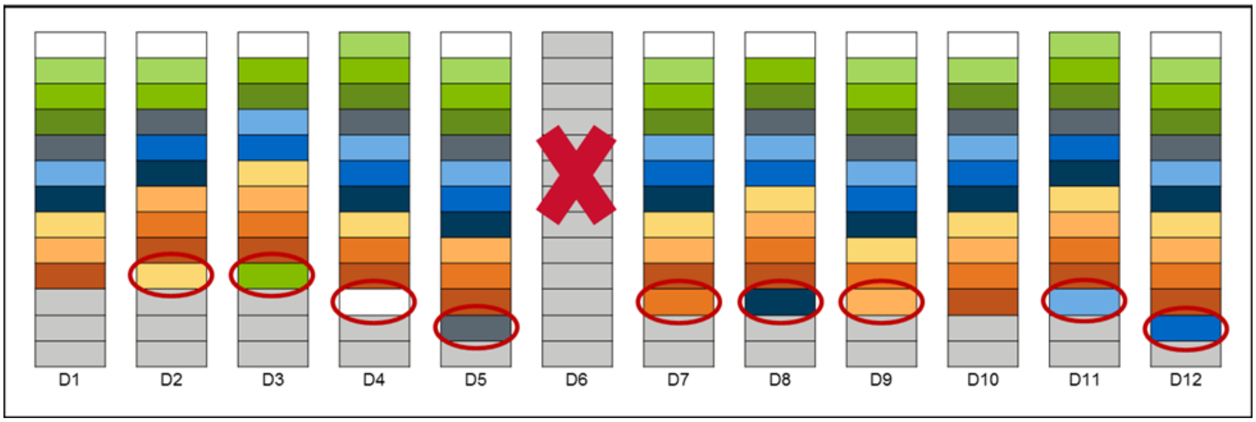


Another major benefit of a DDP is that, rather than using dedicated stranded hot spares, the pool contains integrated preservation capacity to provide rebuild locations for potential drive failures. This approach simplifies management, because individual hot spares no longer need to be planned or managed. The approach also greatly improves the time for rebuilds, if required, and enhances volume performance during a rebuild, as opposed to traditional hot spares.

When a drive in a DDP fails, the D-pieces from the failed drive are reconstructed to potentially all other drives in the pool using the same mechanism normally used by RAID 6. During this process, an algorithm internal to the controller framework verifies that no single drive contains two D-pieces from the same D-stripe. The individual D-pieces are reconstructed at the lowest available logical block address (LBA) range on the selected disk drive.

In Figure 6, disk drive 6 (D6) is shown to have failed. Subsequently, the D-pieces that previously resided on that disk are recreated simultaneously across several other drives in the pool. Because there are multiple disks participating in the effort, the overall performance impact of this situation is lessened, and the length of time needed to complete the operation is dramatically reduced.

Figure 6) Dynamic Disk Pool drive failure.



When multiple disk failures occur within a DDP, priority for reconstruction is given to any D-stripes missing two D-pieces to minimize data availability risk. After those critically affected D-stripes are reconstructed, the remainder of the necessary data is reconstructed.

From a controller resource allocation perspective, there are two user-modifiable reconstruction priorities within DDP:

- Degraded reconstruction priority is assigned to instances in which only a single D-piece must be rebuilt for the affected D-stripes; the default for this value is high.
- Critical reconstruction priority is assigned to instances in which a D-stripe has two missing D-pieces that need to be rebuilt; the default for this value is highest.

For very large disk pools with two simultaneous disk failures, only a relatively small number of D-stripes are likely to encounter the critical situation in which two D-pieces must be reconstructed. As discussed previously, these critical D-pieces are identified and reconstructed initially at the highest priority. Doing so returns the DDP to a degraded state quickly so that further drive failures can be tolerated.

In addition to improving rebuild times and providing superior data protection, DDP can also greatly improve the performance of the base volume when under a failure condition compared to the performance of traditional volume groups.

For more information about DDP, see TR-4115: SANtricity Dynamic Disk Pools BPG.

E-Series Data Protection Features

E-Series has a reputation for reliability and availability. Many of the data protection features in E-Series systems can be beneficial in a Cassandra environment. The following highlights key E-Series protection features that can improve Cassandra reliability and availability.

Encrypted Drive Support

E-Series storage systems provide at-rest data encryption through self-encrypting drives. These drives encrypt data on writes and decrypt data on reads regardless of whether the FDE feature is enabled. Without the SANtricity feature enabled, the data is encrypted at rest on the media, but it is automatically decrypted on a read request.

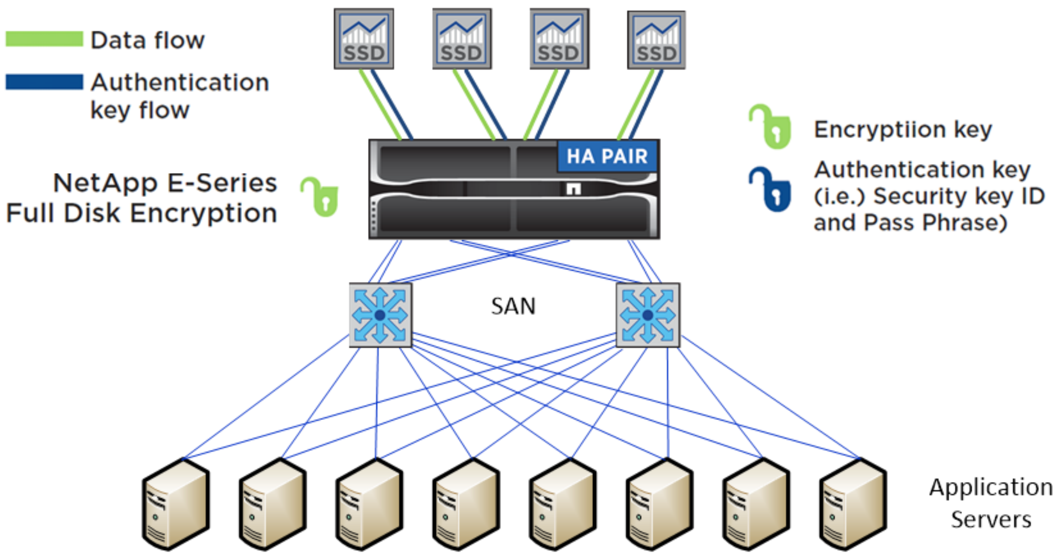
When the FDE feature is enabled on the storage array, the drives protect the data at rest by locking the drive from reads or writes unless the correct security key is provided. This process prevents another array from accessing the data without first importing the appropriate security key file to unlock drives. It also prevents any utility or operating system from accessing the data.

The encryption and decryption performed by the hardware in the drive are invisible to the user and do not affect the performance or user workflow. Each drive has its own unique encryption key, which cannot be transferred, copied, or read from the drive. The encryption key is a 256-bit key as specified in the NIST Advanced Encryption Standard (AES). The entire drive, not just a portion, is encrypted.

Security can be enabled at any time by selecting the Secure Drives option in the Volume Group or Disk Pool menu. This selection can be made either at volume group or disk pool creation or afterward. It does not affect existing data on the drives and can be used to secure the data after creation. However, the option cannot be disabled without erasing all the data on the affected drive group or pool.

Figure 7 shows the technical components of NetApp E-Series Full Disk Encryption.

Figure 7) NetApp E-Series Full Disk Encryption.



For more information about disk encryption, see TR-4474: SANtricity Full Disk Encryption.

Background Media Scan

Media scan is a background process that is performed by the controllers to provide error detection on the drive media. The main purpose of the feature is to detect and repair media errors on disk drives that are infrequently read by user applications and where data loss might occur if other drives in the volume group fail. A secondary purpose is to detect redundancy errors such as data/parity mismatches. A background media scan can find media errors before they disrupt normal drive reads and writes.

Data Assurance (T10 PI)

The data assurance feature provides controller-to-drive data integrity protection through the SCSI direct-access block device protection information model. This model protects user data by appending protection information to each block of user data. The protection model is sometimes referred to as data integrity field protection or T10 PI. This model makes sure that an I/O has completed without any bad blocks written to or read from disk. It protects against displacement errors, data corruption resulting from hardware or software errors, bit flips, and silent drive errors, such as when the drive delivers the wrong data on a read request or writes to the wrong location.

You need both data assurance and media scan. They work complementarily to protect your data.

Unreadable Sector Management

This feature provides a controller-based mechanism for handling unreadable sectors detected both during normal I/O operation of the controller and during long-lived operations such as reconstructions. The feature is transparent to the user and requires no special configuration.

Proactive Drive Health Monitor

Proactive drive health monitoring examines every completed drive I/O and tracks the rate of error and exception conditions returned by the drives. It also tracks drive performance degradation, which is often associated with unreported internal drive issues. Using predictive failure analysis technology, when any error rate or degraded performance threshold is exceeded—indicating that a drive is showing signs of impending failure—SANtricity software issues a critical alert message and takes corrective action necessary to protect the data.

Data Evacuator

With data evacuator, nonresponsive drives are automatically power-cycled to see if the fault condition can be cleared. If the condition cannot be cleared, the drive is flagged as failed. For predictive failure events, the evacuator feature removes data from the affected drive in an effort to move the data before the drive actually fails. If the drive fails, rebuild picks up where the evacuator was disrupted, thus reducing the rebuild time.

Hot Spare Support

The system supports global hot spares that can be automatically used by the controller to reconstruct the data of the failed drive if enough redundancy information is available. The controller selects the best match for the hot spare based on several factors, including capacity and speed.

SSD Wear Life Monitoring and Reporting

If an SSD supports wear life reporting, the GUI provides this information to the user to allow monitoring how much of the useful life of an SSD remains. For SSDs that support wear life monitoring, the percentage of spare blocks remaining in solid-state media is monitored by controller firmware at approximately one-hour intervals. Think of this approach as a fuel gauge for SSDs.

SSD Read Cache

The SANtricity SSD read cache feature uses SSD storage to hold frequently accessed data from user volumes. It is intended to improve the performance of workloads that are performance limited by HDD IOPS. Workloads with the following characteristics can benefit from using the SANtricity SSD read cache feature:

- Read performance is limited by HDD IOPS.
- There is a high percentage of read operations relative to write operations, that is, greater than 80% read.
- A large number of reads are repeat reads to the same or adjacent areas of disk.
- The size of the data that is repeatedly accessed is smaller than the SSD read cache capacity.

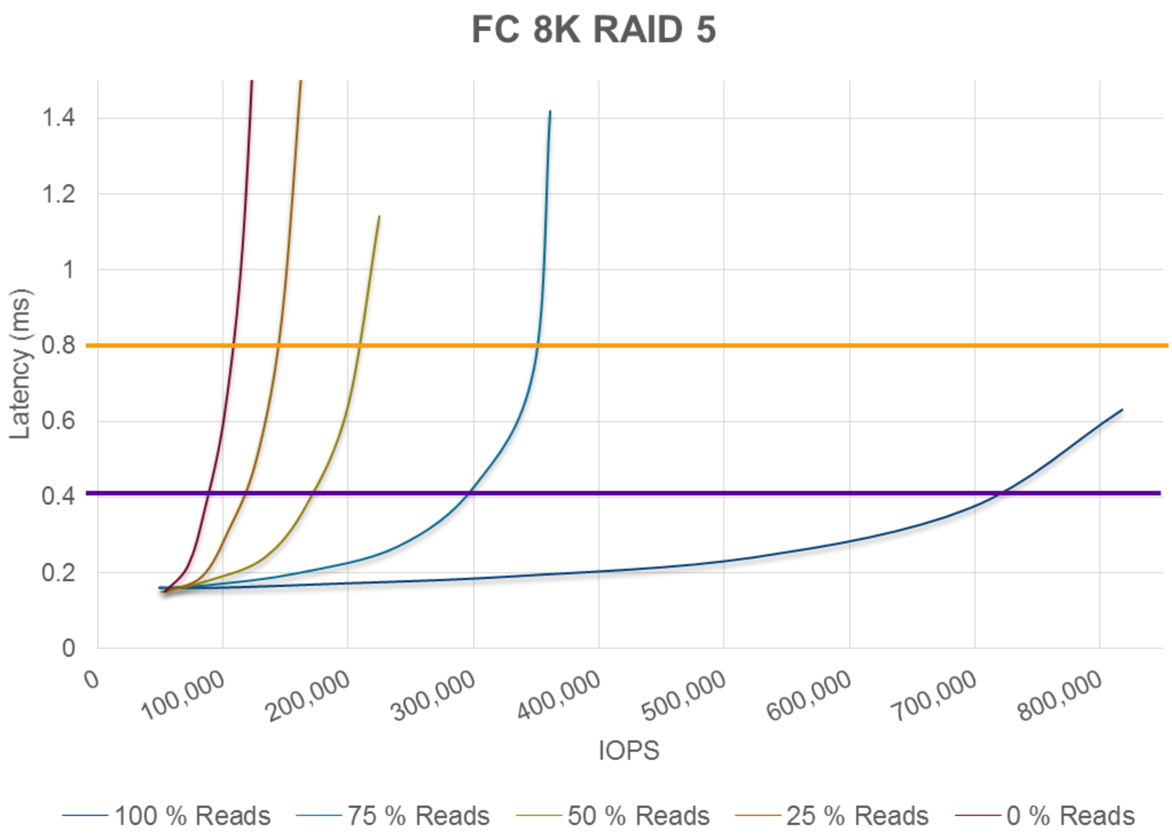
For more information about SSD read cache, see TR-4099: NetApp SANtricity SSD Cache for E-Series.

3.3 Performance

An E5600 with SSDs installed can perform at very high levels, in both IOPS and throughput, while still providing extremely low latency. In many Cassandra deployments, internal SSDs located in the server are often deployed to provide the level of performance required of an expected high-performance NoSQL cluster. The E5600, through its ease of management, higher degree of reliability, and exceptional performance, can meet the extreme performance requirements expected when a document is not located in the memory of a Cassandra cluster server.

An E5600 is capable of providing over 800,000 8KB random read IOPS at less than 800µs average response time in a RAID 5 configuration, as shown in Figure 8.

Figure 8) Performance of the E5600 using 48 SSDs.



Many factors can affect the performance of the E5600, including different volume group types or the use of DDP, the average I/O size, and the read versus write percentage provided by the attached servers. Figure 8 also provides performance graphs across various read percentages for the same RAID 5 8KB I/O size with a Fibre Channel host interface.

4 Disaggregating Storage from Compute

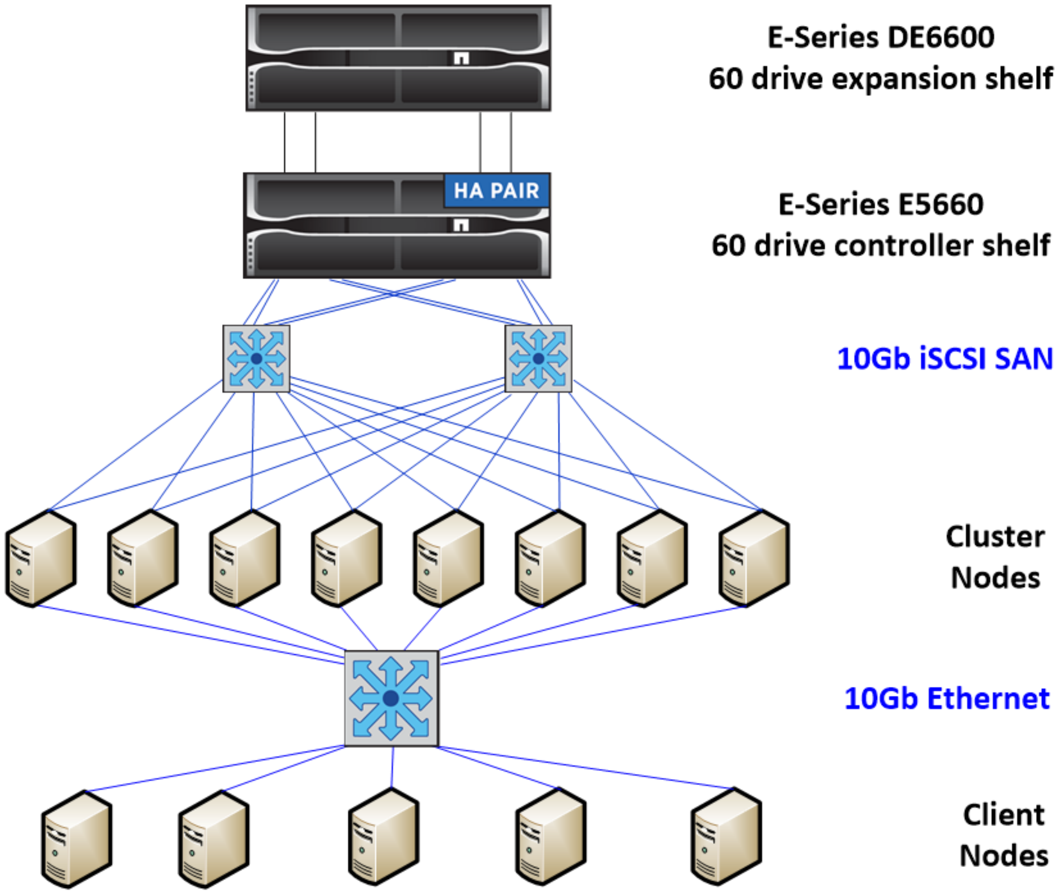
With the advent of larger and larger SSDs, up to 15.3TB SSDs now available, and the expanded use of specialized compute for data analytics such as GPUs, the ability to disaggregate storage and compute separately with NoSQL databases such as Cassandra is becoming an economic necessity.

Figure 9 shows a sample NoSQL architecture where an E-Series E5660 is connected over iSCSI to eight cluster nodes, and then client nodes are connected over Ethernet to the cluster nodes. With NetApp's E-Series it's possible to begin with 20 SSDs and scale all the way up to 120 SSDs without needing to add more cluster nodes in an all-SSD system. If HDDs are being deployed, it is possible to scale up to 384 drives by adding additional drive shelves to the system shown. With E-Series, 4 to 10 nodes per E-Series array work well depending on storage and performance requirements.

The advantages of disaggregating storage from compute include:

- Ability to scale capacity and compute separately, saving the cost of overprovisioning one or the other
- Flexibility to use excess top-of-rack switches bandwidth for the storage network, use a wholly different storage network such as Fibre Channel, or connect the array as direct-attached storage (DAS)

Figure 9) Sample of storage and compute separation.



This type of disaggregation, for example, can allow a company with 100 nodes to reduce its number of nodes to 10, if 100 nodes of compute aren't required. This change provides a significant reduction in rack space required and the associated cooling and power requirements. In contrast, if the need is more compute, then less expensive servers can be purchased that don't require space for additional storage and have a smaller footprint.

After the decision has been made to use a separate storage array, sizing and configuring it are straightforward.

4.1 Sizing

Storage Overhead

The size of raw data is larger after it is loaded into Cassandra due to storage overhead. Data is organized into sorted strings tables (SSTables) in key/value pair format.

On average, raw data is about two times larger on disk after it is loaded into the database. However, it could be much smaller or larger depending on the characteristics of the data and tables.

On disk storage, overhead for Cassandra is the sum of the following:

- Column overhead
- Row overhead
- Primary key index
- Replication overhead
- Compaction overhead

DB Compaction Strategies and Impact to Disk Space and Performance

An operational storage requirement is the need to keep sufficient disk space free and available for Cassandra's compaction. Compaction is a mechanism used to improve data locality and flush stale data from the system. The default compaction strategy is size tiered, which temporarily might require up to double the size of the table it is compacting. Because multiple tables might be compacted simultaneously, it is prudent to keep free space in excess of your live data volume to make sure of optimal performance.

Leveled compaction strategy (LCS) dramatically reduces the temporary disk space requirements, while providing tighter bounds on the latency of read operations. The tradeoff is that LCS generally produces twice as many write operations over time. LCS is generally not recommended for write-heavy workloads for this reason.

Size-tiered compaction strategy (STCS):

- Recommended for write-intense workloads
- Compaction done by merging similar-sized SSTables into one large SSTable
- Requires 50% more disk space than the data size

LCS:

- Recommended for read-intensive workloads
- Requires about 10% more disk space in addition to the space occupied by the data
- LCS is significantly more I/O intensive than other compaction strategies; as a result, it might introduce additional latency for reads

Date-tiered compaction strategy (DTCS):

- Recommended for time series data
- Compaction done based on SSTable age/timestamp
- Requires 50% more disk space than the data size

Sizing Calculations

Database volume size = (total_data_size + replication_overhead + compaction_overhead)

```
total_data_size = (column_overhead + row_overhead + primary_key_index)
column_overhead = regular_total_column_size + counter_and_expiring_total_column_size
```

Every column in Cassandra incurs 15 bytes of overhead. For counter columns and expiring columns, an additional 8 bytes of storage overhead needs to be added (that is, 15+8 = 23 bytes of overhead).

Calculate the column overhead from the user-provided column name size and value size input values in the following manner:

```
regular_total_column_size = column_name_size + column_value_size + 15 bytes
counter_and_expiring_total_column_size = column_name_size + column_value_size + 23 bytes
```

Every row in Cassandra incurs 23 bytes of overhead.

```
row_overhead = number of rows x 23 bytes
```


Every row of the primary key index incurs 32 bytes of overhead.

```
primary_key_index = number of rows x (32 bytes + average_key_size)
```

For a replication factor of 1, there is no overhead for replicas because only one copy of data is stored in the cluster. Therefore, if the replication factor is greater than 1:

```
replication_overhead = total_data_size * ( replication_factor - 1 )
```

From the earlier discussion about compaction strategy and the overhead each requires, we have:

If STCS or DTCS is used:

```
compaction overhead = (0.5) * total_data_size
```

If LCS is used:

```
compaction overhead = (0.1) * total_data_size
```

Now that we know the database volume size, we can create a Dynamic Disk Pool such that:

```
the number of database volumes = the number of nodes sharing the array
```

Capacity of DDP = number of database volumes x database volume size

We also need to consider the size of the commit log:

- The commit log is used to periodically back up in-memory SSTables to storage volumes. For 32-bit JVM, Java heap size is 32MB. For 64-bit JVM, Java heap size is 8GB.
- The default size for the commit log is governed by commitlog_total_space_in_mb in the Cassandra.yaml configuration file.
- The storage for the commit log must be on a separate disk group different from the database volumes and can be provisioned using HDDs (no SSDs are required).

Only one commit log volume is needed per DB instance.

Commit log size = Cassandra JVM heap size + 25%

4.2 Other Considerations

E-Series

To prepare your server for storage access, see:

- Installing and Configuring for Linux Express Guide
- Installing and Configuring for Linux Power Guide for Advanced Users

These documents guide you through:

- Installing SANtricity Storage Manager host-side applications
- Configuring multipath
- Installing NetApp Host Utilities
- Using the iscsiadm open-iscsi utility with E-Series products

The E-Series Interoperability Matrix Tool (IMT) has some 85,000 entries to not only connect to any SAN but also support it. To verify that your configuration is supported and check for any changes that might be required for correct functioning of your E-Series, see the Interoperability Matrix Tool.

Linux Configuration

All servers in the Cassandra cluster were tested with CentOS 7 with default kernel settings.

To increase performance, jumbo frames should be set on the network. Setting jumbo frames for the storage is explained in the E-Series documentation. On the server, they are configured by adding an entry of MTU=9000 to the interface file in the /etc/sysconfig/network-scripts directory and restarting the interface. To validate that jumbo frames have been set, use the `ip link show` command:

```
[root@ictk0103r720-4 ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT link/loopback
00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode DEFAULT qlen 1000
link/ether b0:83:fe:d5:ae:62 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode DEFAULT qlen 1000
link/ether b0:83:fe:d5:ae:64 brd ff:ff:ff:ff:ff:ff
```

For persistent deployments, administrators should consider the following flags when adding a mount into /etc/fstab:

- `nobarrier`: Allows data to sit in cache instead of being flushed. There is a large performance gain on particular workloads by allowing `nobarrier`. This option should only be used for E-Series storage, because internal disks might not have battery backup.
- `_netdev`: Required for configurations using iSCSI and iSER network protocols. The `_netdev` option forces the mount to wait until the network is up before trying to mount. Without this option, the OS attempts to mount the disk prior to the network being completely available, and it could lead to various timeouts or the OS entering recovery mode.
- `discard`: If the storage volume is thinly provisioned, providing the `discard` flag allows the file system to reclaim space. This flag can cause performance degradation. Administrators who want to control when discards take place (for example, nightly) should consider using `fstrim` or an equivalent command for the OS.
- `noatime`: Forces file reads to not record their access times to disk, which can increase I/O dramatically on heavy read loads. Setting the `noatime` flag is only recommended for file systems or dependent applications where a record of the last access time of a file for reading is unnecessary.

If you use multipathing, you need to edit the timeout value in the iSCSI configuration file, /etc/iscsi/iscsid.conf. NetApp recommends using a value of 5 seconds.

```
node.session.timeo.replacement_timeout = 5
```

This amount is the length of time to wait for session reestablishment before failing SCSI commands back to the application when running the Linux SCSI layer error handler. The default value is 120 seconds.

Cassandra

See the following deployment information for DataStax Cassandra to make sure that your Linux environment is set up correctly:

DataStax Cassandra Recommended Production Settings for Linux

5 Cassandra DataStax Enterprise Edition and E5600 Testing

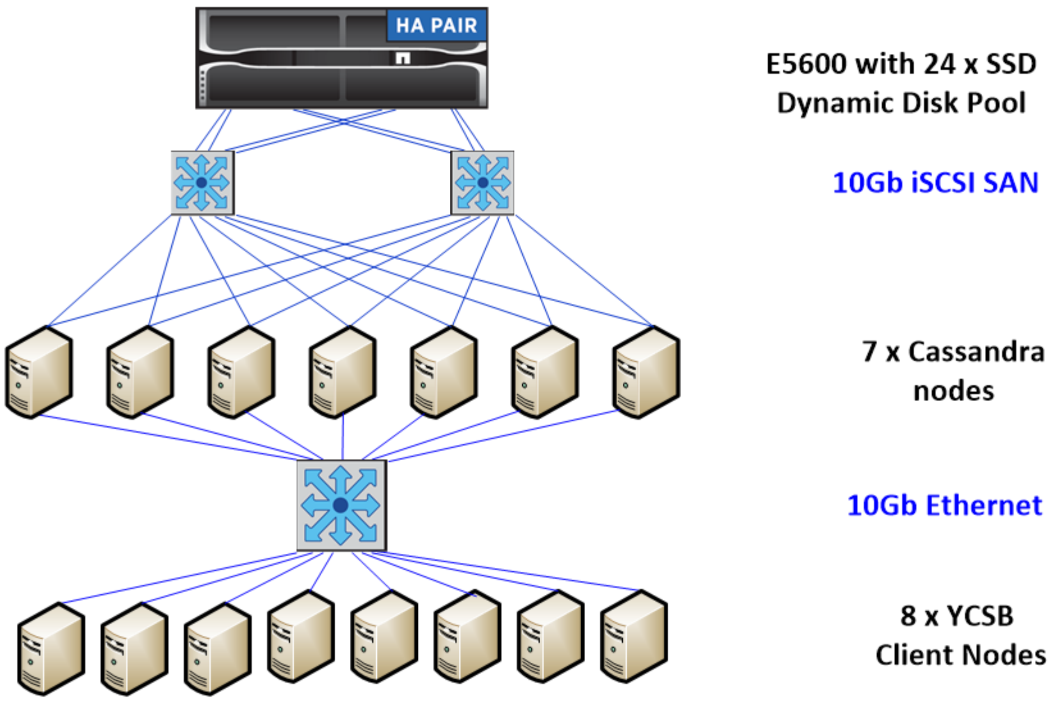
NetApp recently tested a simulated cluster environment with both E-Series and commodity servers configured in a Cassandra cluster of nodes and YCSB client nodes using replication of data throughout the C* cluster. This configuration enabled testing the E-Series compared to the commodity server DAS for the indexing and search functions a Cassandra cluster of nodes requires. The server hardware was chosen following recommendations from the DataStax Cassandra Reference Architecture System Requirements. The Cassandra cluster node server hardware used is listed in Table 2.

Table 2) Cassandra node cluster server hardware.

Cassandra Nodes Cluster	Qty	Type	CPU	CPUs	Cores/CPU	Speed	RAM	OS
Cassandra nodes	7	Dell 730xd	E2-2670 v3	2	8	2.3GHz	128GB	CentOS 7
YCSB client nodes	8	Dell 730	E2-2670 v3	2	8	2.3GHz	128GB	CentOS 7

Figure 10 shows the diagram for the E-Series configuration used for testing: an E5600 with 24 800GB SSDs configured as a DDP using SANtricity Storage Manager. The pool is then shared across the Cassandra server nodes.

Figure 10) E-Series and Cassandra test cluster configuration.



Cassandra Cluster

A seven-node Cassandra cluster was used for the purposes of testing. All nodes were located in one data center and one rack.

To explore the disk failure recovery capabilities of the E-Series system, two different types of storage deployment were tested. The first deployment was a DAS installation with internal SSDs, while the second one was a SAN installation of SSDs configured as a DDP.

For the all-SSD E-Series deployment, the E5600 under test was configured with a single DDP of 24 800GB SSDs, with a pool preservation capacity of one drive offering ~12TiB of usable capacity. Seven 1.2TB volumes were created, with one volume per each Cassandra cluster server node.

The following data model was used:

- Each row of testing data consisted of 11 fields.
- The key field size was 24 bytes.

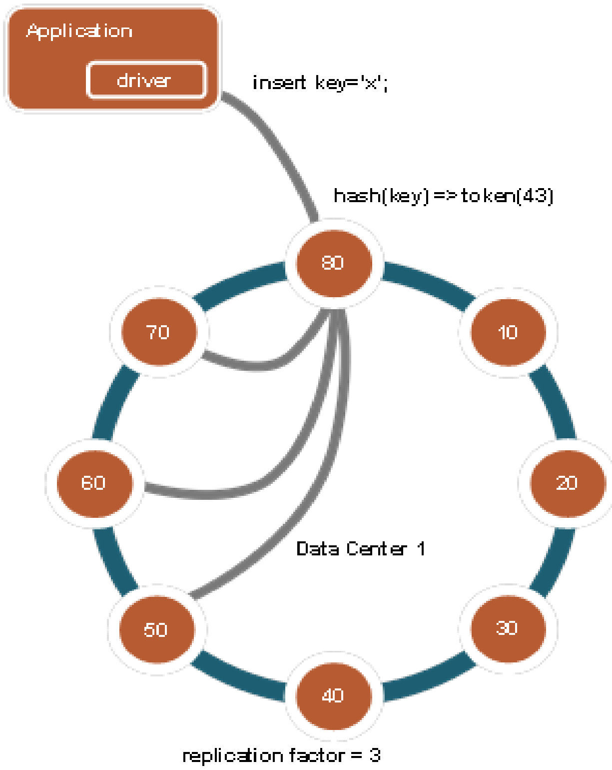
- The total record size was 2,024 bytes.
- The total number of records to read and update was 600 million.
- The total data size was approximately 1130GB.
- Each row had two additional copies (replication factor 3), so the cluster contained approximately 3400GB of data.
- Each node owned approximately 500GB to 560GB of data.

Cassandra configured in a clustered environment uses a deployment strategy where all the nodes are equal members of the cluster. Data copies are created and spread throughout the cluster to enable redundancy and data availability when a node is down and the primary copy of data is not available. This process is known as replication and stores copies of rows on different nodes. Row copies are called replicas. When a row is first written, it is also referred to as a replica (first replica). A set of nodes that store replicas with the same partition key is called replica nodes.

The total number of replicas across the system is referred to as the replication factor. The replication factor is less than the total number of system nodes but greater than zero, but it is possible to increase the replication factor and add the desired number of nodes afterward. Replication factor 1 means there is only the first replica. Figure 11 shows a graphical representation of Cassandra cluster replication when the replication factor is 3. When the replication factor is greater than the total number of system nodes, writes are rejected, but reads are served if the desired consistency level can be met. Consistency level specifies on how many replicas the write/read must succeed before returning an acknowledgement to the client application. There is no leading, or master, replica. All replicas have equal roles.

In addition to the replication factor system, a replica placement strategy is used. The strategy sets the distribution of the replicas across the nodes in the system based on topology snitch. Snitches provide the possibility to group nodes into logical units called racks and data centers. Through snitches, requests are routed efficiently, and the Cassandra cluster distributes data according to data centers and racks.

Figure 11) Cassandra cluster replication factor 3 example.



5.1 Building the Cassandra Cluster

Cassandra Deployment

Having 100% of inserts ("writes only") is a typical scenario of using Cassandra. In such a case, however, the Cassandra storage engine doesn't guarantee high performance for read operations. So, while executing the test, the Cassandra cluster nodes were configured to optimize the cluster performance for insert operations.

Therefore:

- The number of concurrent compaction processes was low.
- The size-tiered compaction strategy was used.
- The number of concurrent writers and memtable flush writers was increased.
- To create balance between the disk throughput and the speed of arriving client requests, the memtables heap space was increased.
- For reducing JVM GC pauses, the garbage first (G1) collector was used with several additional options.

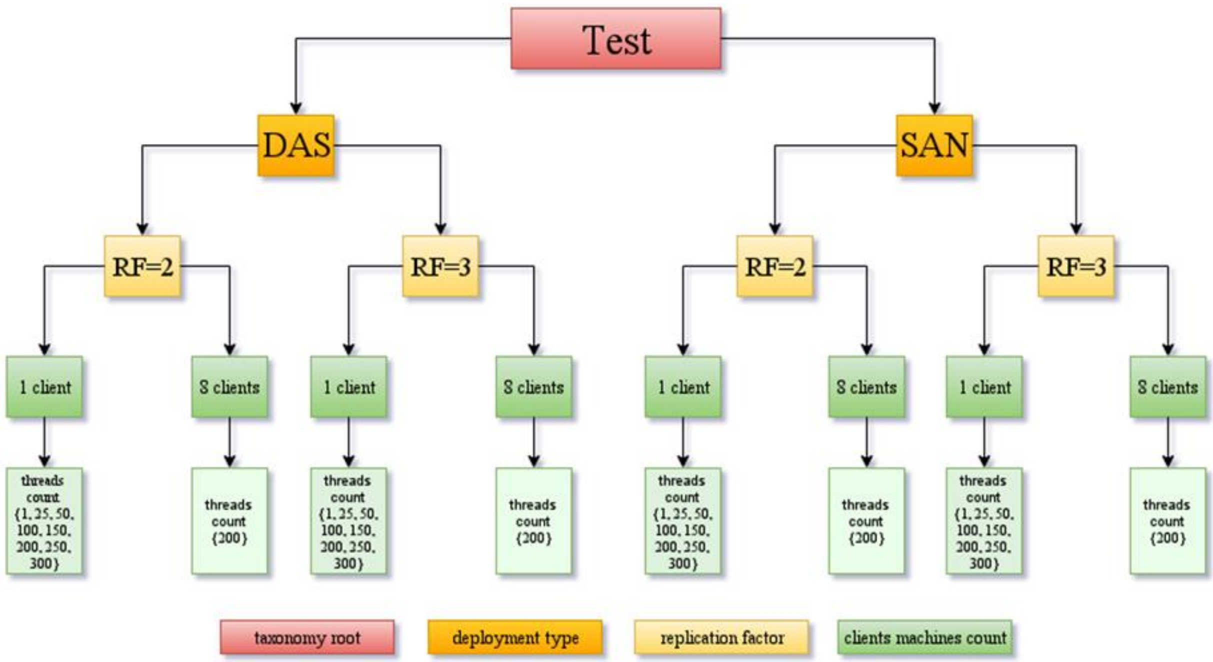
The test included the following set of attributes (see Figure 12):

- Cluster replication factor value (RF=2 or RF=3)
- Storage type (SAN or DAS [internal server RAID 0 columns])
- Test type (using a single YCSB client and varying the number of threads or using multiple [eight] YCSB clients with 200 threads per each of them)

- For single client testing, tests were run using the YCSB tool on one machine and varying the number of threads that sent requests simultaneously from 1 to 300
- For multiple client testing, eight YCSB machines were used, starting from one machine and adding one more every 200 seconds

Running eight YCSB clients helps to understand how Cassandra works under a high load: 300 versus 2,000 connections. (You can think about the eight YCSB clients as an office with 2,000 machines.) As a result, we can see how a high load affects the disk performance (what the write speed and latency delta are as compared to one client) and the overall performance of the Cassandra cluster.

Figure 12) Cassandra cluster and clients.



Data was inserted into a keyspace with one table consisting of 11 fields. The key field size was 24 bytes. The total record size was 4,024 bytes.

The number of records to insert was 310 million. The total data size was 1162GB. With replication (2 or 3), it was 2324GB or 3486GB, respectively, where each node owned approximately 340GB or 500GB of the data. The numbers mentioned in the paragraph are related to one YCSB client testing.

5.2 Performance Testing

In the test environment, Casandra cluster performance was measured using the YCSB tool. A particular test is defined by the following choices:

- **YCSB workload type.** Which create, read, update, delete (CRUD) operation to perform on the document and which document is selected as the target
- **Document size.** Document size in bytes, balance between data and metadata
- **Total number of documents.** Total data size in cluster, percentage of data that fits in memory
- **Write durability options.** How many nodes have the result of the operation persisted in memory and on disk before it is considered successful

Database performance is defined by the speed at which a database computes basic CRUD operations. A basic operation is an action performed by the workload executor that drives multiple client threads. Each thread executes a sequential series of operations by making calls to the database interface layer, both to load the database (the load phase) and to execute the workload (the transaction phase). The threads throttle the rate at which they generate requests, so that we may directly control the offered load against the database. In addition, the threads measure the latency and achieved throughput of their operations and report these measurements to the statistics module.

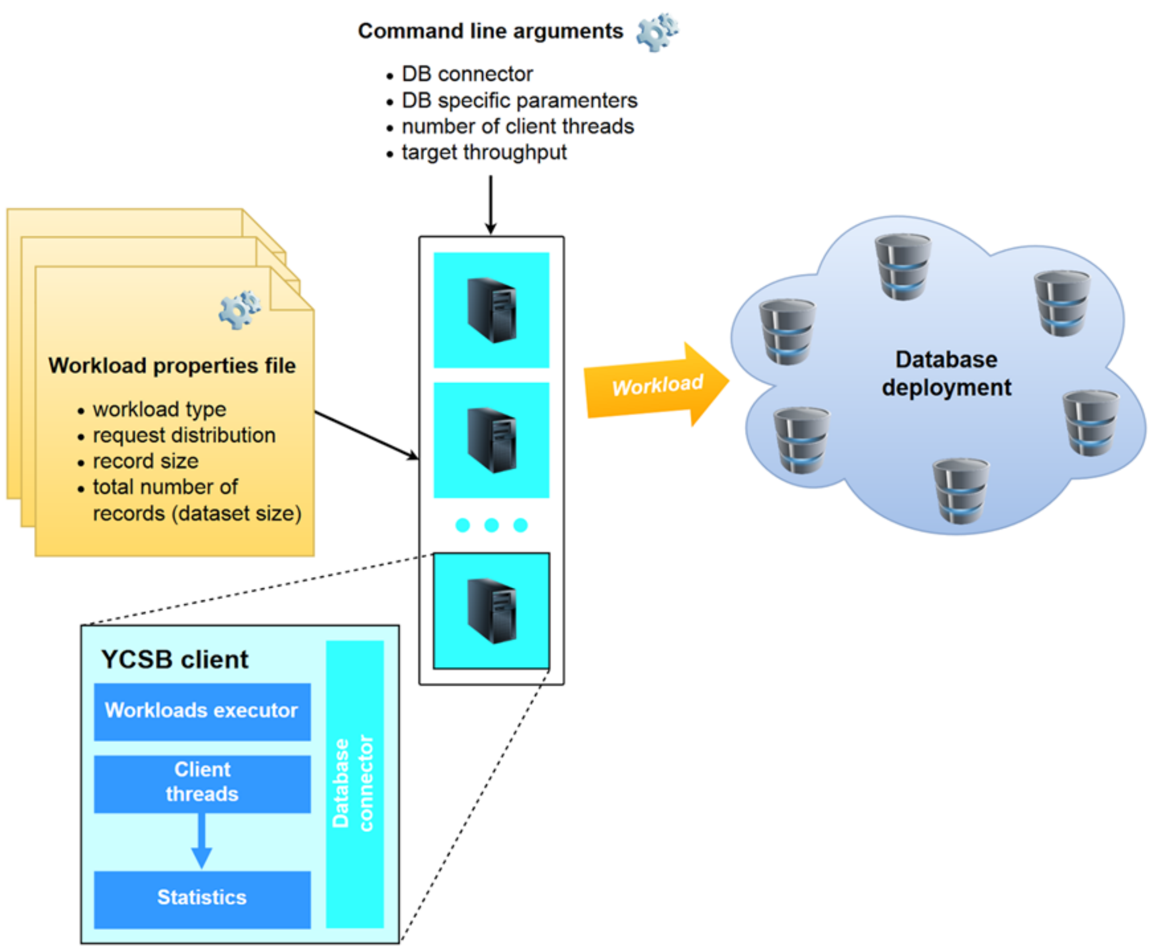
A very common use case for applications deploying Cassandra is environments where heavy write operations having 100% of inserts ("writes only") is a typical scenario are the most prevalent. YCSB was configured during all tests with the four following workloads to documents within the database:

- 100% inserts
- 100% reads
- 50% read, 50% updates
- 95% reads, 5% updates

These tests were conducted on seven Cassandra cluster servers. Performance and scalability were tested with a compaction sequence introduced before increasing the number of cluster servers after the update tests were complete.

The typical YCSB workflow into a cluster is depicted in Figure 13.

Figure 13) YCSB deployment diagram.



YCSB client testing used a single YCSB client and varied the number of threads or used multiple [eight] YCSB clients with 300 threads each.

Example parameter variations for the write-heavy workload:

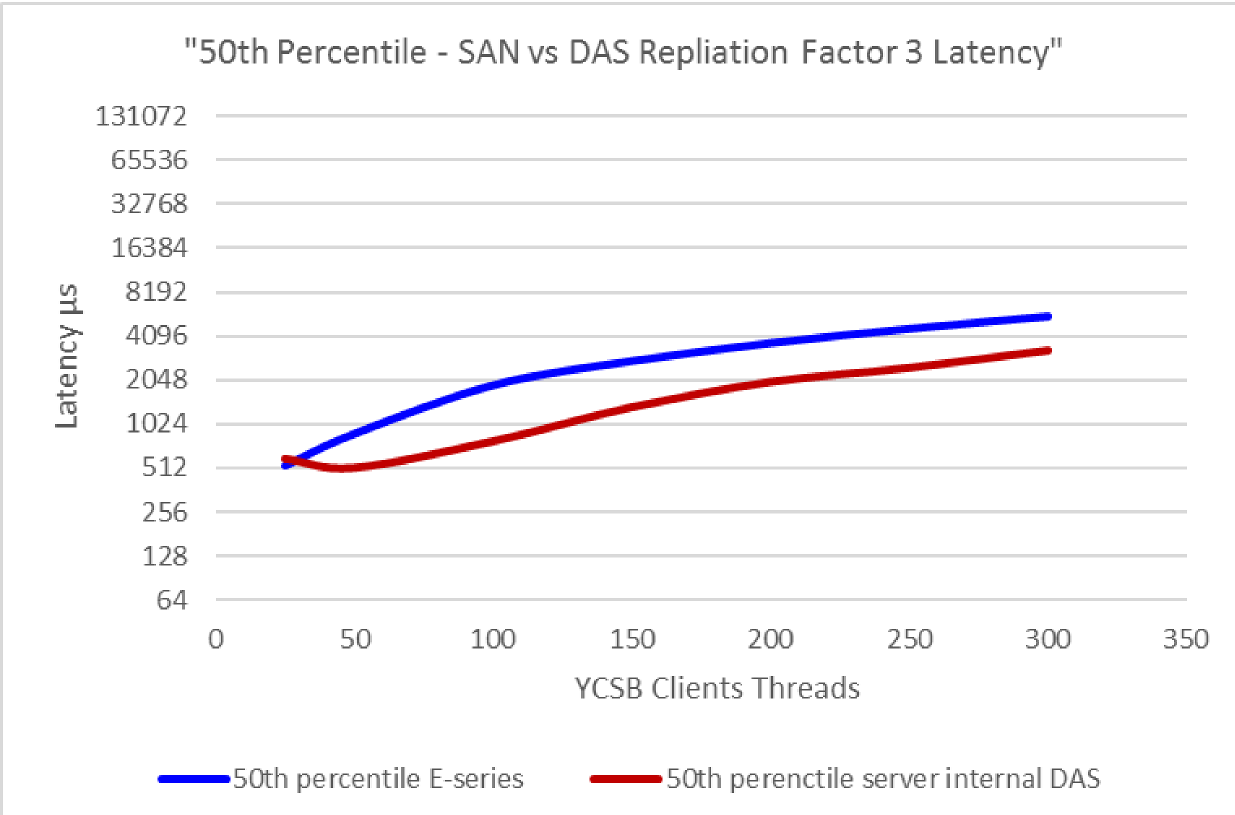
Data was inserted into a keyspace with one table consisting of 11 fields. The key field size was 24 bytes. The total record size was 4,024 bytes. The number of records to insert was 310 million. The total data size was 1162GB. With replication (2 or 3), it was 2324GB or 3486GB, respectively, where each node owned approximately 340GB or 500GB of the data. The numbers mentioned in the paragraph are related to one YCSB client testing.

Example of the parameter variations for the read heavy workload:

Each row of testing data consisted of 11 fields. The key field size was 24 bytes. The total record size was 2,024 bytes. The total number of records to read and update was 600 million. The total data size was approximately 1162GB. Each row had two additional copies (replication factor 3), so the cluster contained approximately 3486GB of data. Each node owned approximately 560GB of data.

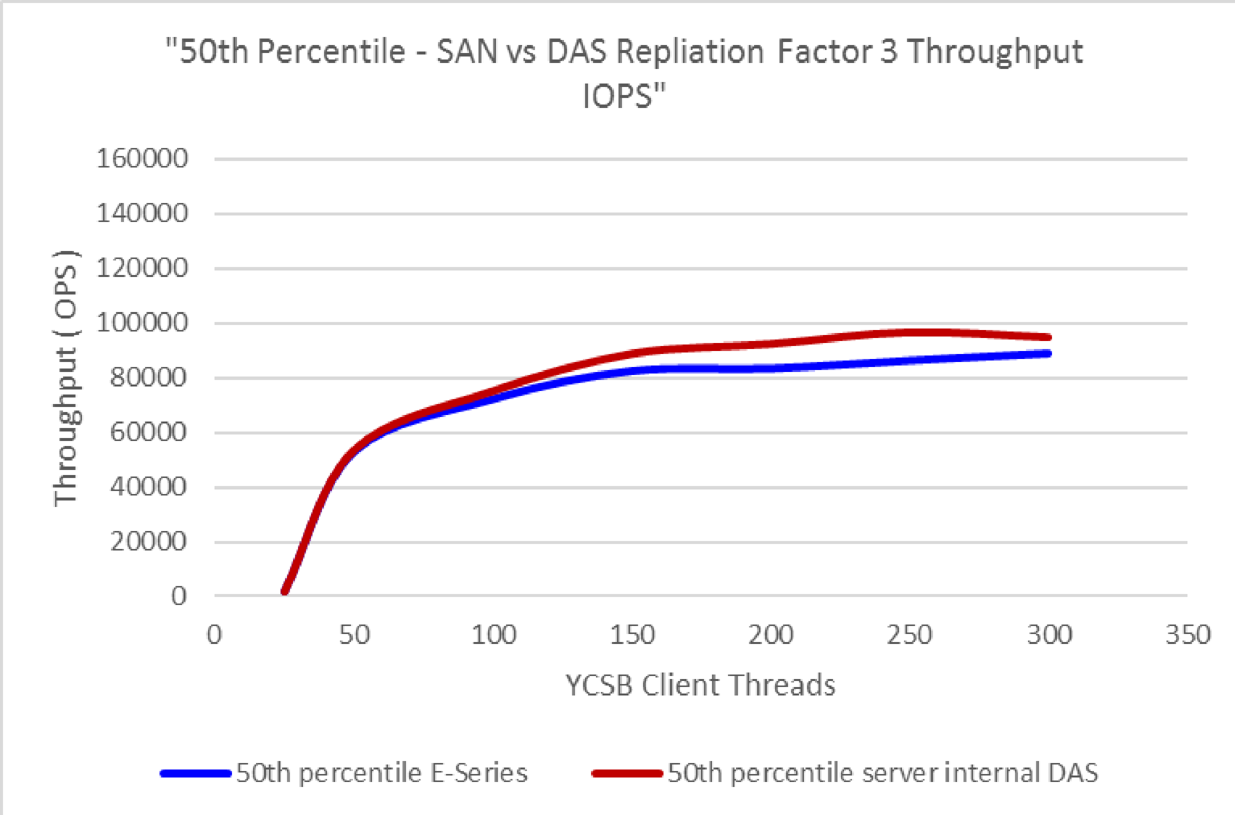
The YCSB tool then loaded the database into the cluster, and we ran from a single YCSB client with thread count up to a maximum of 300. Measuring the latency and throughput for the 8-node Cassandra cluster, we collected the performance using the 50th percentile for the median, which is an important measure of database workloads. We compared the SAN (E-Series DDP volumes) versus the DAS (commodity server RAID 0 volumes) with a replication factor of 3. In Figure 14 and Figure 15, the E-Series has nearly the same latency and throughput as the DAS.

Figure 14) E-Series latency: single YCSB client.



The preceding graph measures the latency in microseconds per YCSB client thread.

Figure 15) E-Series throughput: single YCSB client.



The preceding graph shows the database throughput ops per YCSB client thread. See [Blog: Why Averages Suck and Percentiles Are Great] for a discussion about the use of percentiles.

5.3 Cassandra Cluster Node Failure Testing

In addition to baseline performance testing, additional testing was conducted on the E5600 under failure conditions, including controller failure and drive failure.

In a typical Cassandra cluster deployment using only internal drives within the server, there is no redundancy at the RAID controller level with a RAID 0 internal disk configuration. A failure of the internal controller would be similar to a complete server failure and would require rebalancing of the data across the remaining cluster servers. With the E5600 and the redundancy provided through the dual redundant controller design, no rebalancing is required because all volumes simply transition to the remaining controller. A test was conducted in which a controller within the E5600 was failed under an active workload.

The following set of tests was designed to show E-Series disk failure recovery capabilities. Two different storage deployments were tested. The first deployment was the all-SSD installation, and the second one was built of enterprise-level SAS HDDs and an SSD read cache, as described previously. Our goal was to estimate the Cassandra cluster performance negative impact of a single disk drive failure under heavy read workload.

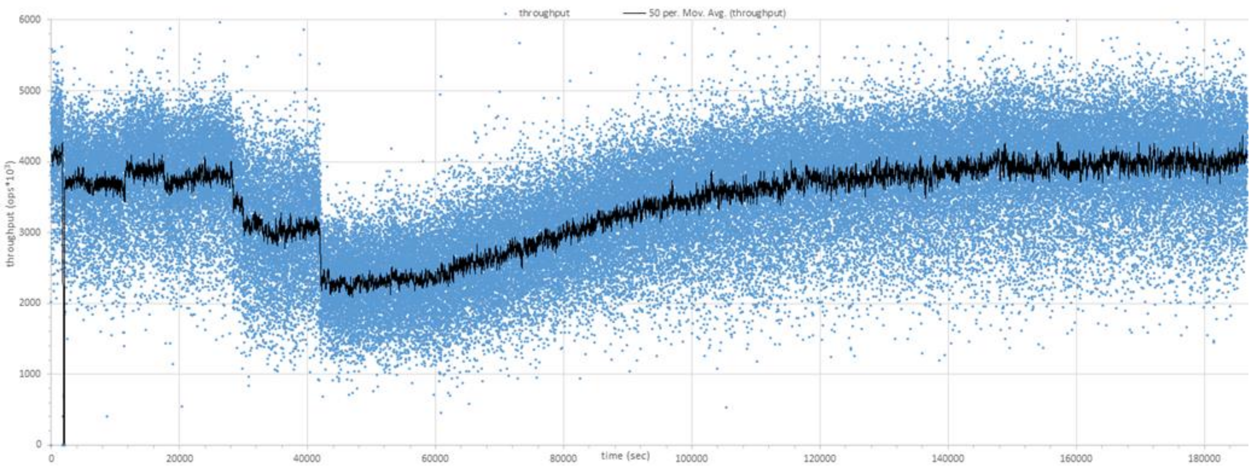
During the tests, data was read uniformly at the read consistency level equal to 2. To create a load, eight YCSB client machines were used simultaneously. One disk in the E-Series flash array (or one disk in the DAS RAID 0) was failed after 20 to 30 minutes. As a result, one cluster node running on the DAS deployment went down and remained inaccessible.

After the failure, we simulated the disk replacement process and kept measuring until all data was recovered and the cluster performance was back to its original state.

The test results are presented in Figure 16 and Figure 17. As you can see, for the SAN deployment, rebalancing took approximately 2.5 hours when the disk was pulled out and then replaced (the pit on the graph). After the rebalancing was finished, the cluster performance came back to its original state (approximately 2,000 ops on average). In contrast, the DAS deployment came back to its original state only after approximately 50 hours. The cluster rebalancing included data recovery on the failed node and data compaction. It took about 40 minutes to replace the failed disk, restart the Cassandra node, and start the recovery process for the DAS deployment. While recovering, the DAS throughput degraded almost two times.

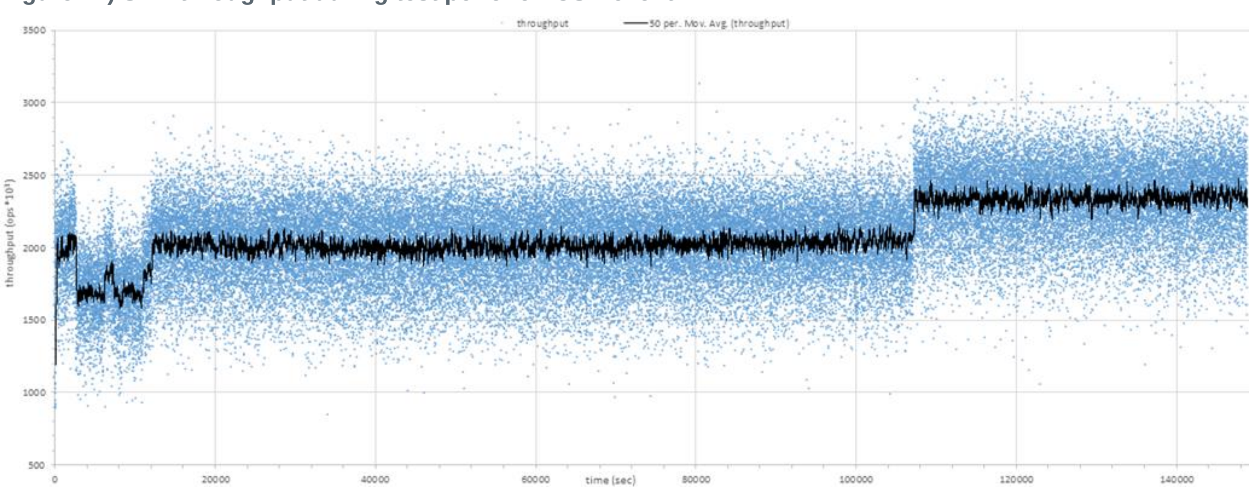
It is obvious that if a client requires consistent data reads (consistency level equal to 3), the DAS cluster will fail many operations.

Figure 16) DAS throughput during test per one YCSB client.



The preceding graph shows the YCSB throughput for the DAS configuration from slightly before a cluster node has the RAID 0 volume fail from a drive failure to when the volume is created and brought back online for the node. The prefailure performance is reached again approximately 50 hours after the failure.

Figure 17) SAN throughput during test per one YCSB client.



The preceding graph shows the YCSB throughput when a drive failure occurs on the E-Series DDP volume. The cluster nodes do not know about the failure as the E-Series rebuilds and rebalances the DDP. This process takes approximately 2.5 hours for prefailure performance to return.

Summary

The NetApp E-Series provides a number of significant advantages over internal DAS for Cassandra deployments. The E5600 configured with all SSD enables high performance and availability of Cassandra data, as seen in the test results using YCSB client testing simulation of real customer workloads.

NetApp SANtricity provides compelling customer solutions for read-heavy workloads in a clustered Cassandra environment. These advantages include Dynamic Disk Pools, which provide rapid rebuild under disk failures so the Cassandra database degradation is minimal. Dynamic Disk Pools can be increased in size dynamically to provide additional capacity and/or performance as required for the Cassandra cluster environment for both a current deployment and when the cluster needs to scale out to meet additional use case requirements.

The NetApp E-Series integrated architecture for Cassandra is optimized for node storage balance, reliability, performance, storage capacity, and density. From an administrative standpoint, the E-Series offers simplified storage management with a centralized user interface. This solution enables new volumes, volumes groups, and Dynamic Disk Pools to be created easily and provisioned immediately for use by the Cassandra cluster servers.

By disaggregating storage from compute, you gain the ability to scale capacity and compute separately, saving the cost of overprovisioning one or the other.

References

TR-4494: Introduction to E-Series E5600 Hardware Using SANtricity 11.25

TR-4115: SANtricity Dynamic Disk Pools BPG

TR-4099: NetApp SANtricity SSD Cache for E-Series

TR-4474: SANtricity Full Disk Encryption

Interoperability Matrix Tool

Installing and Configuring for Linux Express Guide

Installing and Configuring for Linux Power Guide for Advanced Users

DataStax Cassandra Recommended Production Settings for Linux

DataStax Cassandra Reference Architecture System Requirements

Blog: Why Averages Suck and Percentiles Are Great

Version History

Version	Date	Document Version History
Version 1.0	April 2017	Original release.

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

NetApp