Planning and testing DastaStax Enterprise deployments

Updated: 2019-01-19-08:00

Table of Contents

Selecting hardware for DataStax Enterprise implementations

General guidelines

Attention: These recommendations are guidelines. To ensure your implementation meets expectations, DataStax recommends contacting the DataStax Services team and testing your configuration thoroughly before deployment.

Follow these guidelines when choosing hardware for DataStax Enterprise:

- Hardware choices depends on your particular use case. The right balance of memory, CPU, disks, number of nodes, and network are vastly different for environments with static data that are accessed infrequently than for volatile data that is accessed frequently.
- The suggested guidelines are the minimum required. You may need to increase memory, CPU, disk from the recommended minimums.
- Be sure to read Anti-patterns in DataStax Enterprise *(page 14)* for important information about SAN storage, NAS devices, and NFS.
- **Thoroughly test your configuration before deployment.**

Stress testing

To help make certain your that hardware selections meet your business needs, DataStax provide stress testing tools:

cassandra-stress
: The cassandra-stress tool at your desired configuration. Be sure to test common administrative operations, such as bootstrap, repair, and failure. See Testing your DataStax Enterprise cluster before production *(page 22)*.

DSE Search stress demo
: The solr_stress demo is a benchmark tool that simulates the number of requests for reading and writing data on a DSE Search cluster using a specified sequential or random set of iterations. See MBeans search demo.

The default location of the `demos` directory depends on the type of installation:

- Package installations: `/usr/share/dse/demos`
- Tarball installations: *`installation_location`*`/demos`

Caution: The hardware used for load testing should mimic production environment, by parameters and number of nodes. Performing load testing on a single node, or in environments where the number of nodes is equal to the replication factor, will result in incorrect measurements.

Memory

The more memory a DataStax Enterprise node has, the better read performance. More RAM also allows memory tables (memtables) to hold more recently written data. Larger memtables

lead to a fewer number of SSTables being flushed to disk, more data held in the operating system page cache, and fewer files to scan from disk during a read. The ideal amount of RAM depends on the anticipated size of your hot data.

Table 1: Recommended minimum memory for dedicated hardware and virtual environments

<table>
<tr><th colspan="3">Production</th></tr>
<tr><td>Node type</td><td>System memory</td><td>Heap</td></tr>
<tr><td>Transactional only</td><td>32 GB</td><td>8 GB</td></tr>
<tr><td>DSE Analytics</td><td>32 GB to 512 GB</td><td rowspan="3">System memory less than 64 GB: 24 GB
System memory greater than 64 GB: 30 GB</td></tr>
<tr><td>DSE Search

Also see Capacity planning for DSE Search (page 10).</td><td>32 GB to 512 GB</td></tr>
<tr><td>DSE Graph</td><td>Add 2 to 4 GB to your particular combination of DSE Search or DSE Analytics. For a large dedicated graph cache, add more RAM.</td></tr>
<tr><td colspan="3">Development (non-load testing environments)</td></tr>
<tr><td>Any</td><td>8 GB</td><td>4 GB</td></tr>
<tr><td>Any</td><td>16 GB</td><td>8 GB</td></tr>
</table>

CPU

Insert-heavy workloads are CPU-bound in DataStax Enterprise before becoming memory-bound. (All writes go to the commit log, but the database is so efficient in writing that the CPU is the limiting factor.) DSE is highly concurrent and uses as many CPU cores as available. Recommendations:

- Minimum dedicated hardware for production: 16-core CPU processors (logical).
- Dedicated hardware in development in non-loading testing environments: 2-core CPU processors (logical) are sufficient.
- DSE Graph: a slightly larger CPU is recommended as graph queries can be a CPU bottleneck due to query optimization and result set preparation.

Spinning disks versus solid state drives (local only)

Note: For cloud deployments, contact the DataStax Services team.

SSDs are recommended for all DataStax Enterprise nodes. The NAND Flash chips that power SSDs provide extremely low-latency response times for random reads while supplying ample sequential write performance for compaction operations. In recent years, drive manufacturers have improved overall endurance, usually in conjunction with spare (unexposed) capacity. Additionally, because PBW/DWPD ratings are probabilistic estimates based on worst case scenarios, such as random write workloads, and because the database does only large sequential writes, drives significantly exceed their endurance ratings. However, it is important to plan for drive failures and have spares available. A large variety of SSDs are available from server vendors and third-party drive manufacturers.

Endurance is a key factor when purchasing SSDs. The best recommendation is to make the decision based on how difficult it is to change drives when they fail, not on workload of the drive. Remember, your data is protected because the database replicates data across the cluster. Buying strategies include:

- If drives are quickly available, buy the cheapest drives that provide the performance you want.
- If it is more challenging to swap the drives, consider higher endurance models, possibly starting in the mid range, and then choose replacements of higher or lower endurance based on the failure rates of the initial model chosen.

For additional help in determining the most cost-effective option for a given deployment and workload, contact the DataStax Services team.

Disk space

Disk space depends on usage, so it's important to understand the mechanism. The database writes data to disk when appending data to the commit log for durability and when flushing memtables to SSTable data files for persistent storage. The commit log has a different access pattern (read/writes ratio) than the pattern for accessing data from SSTables. This is more important for spinning disks than for SSDs.

SSTables are periodically compacted. Compaction improves performance by merging and rewriting data and discarding old data. However, depending on the type of compaction and size of the compactions, during compaction disk utilization and data directory volume temporarily increases. For this reason, be sure to leave an adequate amount of free disk space available on a node.

For large compactions:

Compaction strategy	Minimum requirements
SizeTieredCompactionStrategy (STCS)	Sum of all the SSTables compacting must be smaller than the remaining disk space. **Note:** Worst case: 50% of free disk space. This scenario can occur in a manual compaction where all SSTables are merged into one giant SSTable.
LeveledCompactionStrategy (LCS)	Generally 10%. Worse case: 50% if the Level 0 backlog exceeds 32 SSTables (LCS uses STCS for Level 0).

Compaction strategy	Minimum requirements
TimeWindowCompactionStrategy (TWCS)	TWCS requirements are similar to STCS. TWCS requires approximately 50% extra disk space for the total size of SSTables in the last created bucket. To ensure adequate disk space, determine the size of the largest bucket ever generated and add 50% extra disk space.

Note: DateTieredCompactionStrategy (DTCS) is deprecated.

Additional resources:

- How is data maintained?
- Calculating usable disk capacity
- Configuring compaction

Minimum disk space recommendations

Important: Use these recommendations as a starting point. **Be sure to thoroughly test before production deployment.** DataStax highly recommends testing with the cassandra-stress tool at your desired configuration. Be sure to test common administrative operations, such as bootstrap, repair, and failure, to make certain your hardware selections meet your business needs. See Testing your DataStax Enterprise cluster before production *(page 22)*. For DSE Search, also test using `solr_stress`.

Capacity per node (node density)

Node capacity is highly dependent on the environment. Determining node density depends on many factors, including:

- Whether data changes frequently or infrequently.
- Access frequency.
- Compaction strategy: choice of compaction strategy depends of whether the workload is write-intensive or read-intensive or time dependent. See Disk space *(page 6)* above.
- Using HDDs or SSDs.
- Network performance: remote links likely limits storage bandwidth and increase latency.
- Storage speed and whether the storage is local or not.
- Replication factor: See About data distribution and replication.
- SLAs (service-level agreements) and ability to handle outages.
- Whether the data is compressed or not.

To avoid problems, DataStax recommends keeping data per node near or below 1 TB. Exceeding this value has the following effects:

- Extremely long times (days) for bootstrapping new nodes.
- Impacts maintenance (day-to-day operations), such as recovering, adding, and replacing nodes.

- Reduces efficiency when running repairs.
- Significantly extends the time it takes to expand datacenters.
- Substantially increases compactions per node.

Higher capacity nodes works best with static data and low access rates.

Search node capacity

For best performance, use a dedicated drive for search indexes.
DataStax recommends the following maximum index sizes:

- Single index: 250 GB maximum

  Once a single index exceeds 250 GB or if performance degrades, consider adding nodes to further distribute the search index.

- Multiple indexes: collective index size: 500 GB maximum

  Supporting more than one index is hardware dependent with respect to the number of physical CPU cores available. DataStax recommends a minimum of two physical cores per search index where the maximum number of search indexes is the number of physical cores divided by two.

  For example, if a machine has 16 virtual CPUs on 8 physical cores, the recommended maximum number of search indexes is 4.

See Set the location of search indexes.

Note: DataStax recommends extensive testing *(page 22)* or consulting assistance from the DataStax Services team. Also see Capacity planning for DSE Search *(page 10)*.

Capacity and I/O

When choosing disks for your nodes, consider both capacity (how much data you plan to store) and I/O (the write/read throughput rate). Some workloads are best served by using less expensive SATA disks and scaling disk capacity and I/O by adding more nodes (with more RAM).

Number of disks - SATA

Ideally DataStax Enterprise needs at least two disks per node, one for the commit log and the other for the data directories. At a minimum the commit log should be on its own partition.

Commit log disk - SATA

The disk need not be large, but it should be fast enough to receive all of your writes as appends (sequential I/O).

Commit log disk - SSD

Unlike spinning disks, it's alright to store both commit logs and SSTables are on the same mount point.

DSE Search - SSD

Because DSE Search is very IO intensive, transactional data and search data must be on different SSDs. Otherwise, the SSD can be overrun from both workloads.

Data disks

Use one or more disks per node and make sure they are large enough for the data volume and fast enough to both satisfy reads that are not cached in memory and to keep up with compaction.

RAID on data disks

It is generally not necessary to use RAID for the following reasons:

- Data is replicated across the cluster based on the replication factor you've chosen.
- DataStax Enterprise includes a JBOD (Just a bunch of disks) feature to take care of disk management. Because the database responds according to your availability/consistency requirements to a disk failure either by stopping the affected node or by blacklisting the failed drive, you can deploy nodes with large disk arrays without the overhead of RAID 10. You can configure the database to stop the affected node or blacklist the drive according to your availability/consistency requirements. Also see Recovering from a single disk failure using JBOD.

RAID on the commit log disk

Generally RAID is not needed for the commit log disk. Replication adequately prevents data loss. If you need extra redundancy, use RAID 1.

Extended file systems

DataStax recommends deploying on XFS or ext4. On ext2 or ext3, the maximum file size is 2 TB even using a 64-bit kernel. On ext4 it is 16 TB.

Because the database can use almost half your disk space for a single file when using `SizeTieredCompactionStrategy`, use XFS when using large disks, particularly if using a 32-bit kernel. XFS file size limits are 16 TB max on a 32-bit kernel, and essentially unlimited on 64-bit.

Network

Minimum recommended bandwidth: 1000 Mb/s (gigabit).

Because DataStax Enterprise is a distributed data store, it puts load on the network to handle read/write requests and replication of data across nodes. Be sure that your network can handle inter-node traffic without bottlenecks. DataStax recommends binding your interfaces to separate Network Interface Cards (NIC). You can use public or private NICs depending on your requirements.

The database efficiently routes requests to replicas that are geographically closest to the coordinator node and chooses a replica in the same rack when possible. The database will always choose replicas located in the same datacenter over replicas in a remote datacenter.

Firewall

If using a firewall, make sure that nodes within a cluster can communicate with each other. See Securing DataStax Enterprise ports table.

Capacity planning for DSE Search

Using DSE Search is memory-intensive and rereads the entire row when updating indexes which can cause a significant performance hit on spinning disks. Use solid-state drives (SSD) for applications that have very aggressive insert and update requirements.

This capacity planning discovery process helps you develop a plan for having sufficient memory resources to meet the operational requirements.

Overview

First, estimate how large your search index will grow by indexing a number of documents on a single node, executing typical user queries, and then examining the memory usage for heap allocation. Repeat this process using a greater number of documents until you get a solid estimate of the size of the index for the maximum number of documents that a single node can handle. You can then determine how many servers to deploy for a cluster and the optimal heap size. Store the index on SSDs or in the system IO cache.

Capacity planning requires a significant effort by operations personnel to:

- Set the optimal heap size per node.
- Estimate of the number of nodes that are required for your application.
- Increase the replication factor to support more queries per second.
- Distributed queries in DSE Search are most efficient when the number of nodes in the queried data center (DC) is a multiple of the replication factor (RF) in that DC.

Note: The Preflight check tool can detect and fix many invalid or suboptimal configuration settings.

Prerequisites:

A node with:

- The amount of RAM that is determined during capacity planning.
- DataStax recommends the following:
  - One data/logs disk. (If using spinning disks, a separate disk for the commit log. See Disk space *(page 6)*.)
  - One disk for DSE Search.

Input data:

- *N* documents indexed on a single test node
- A complete set of sample queries to be executed
- The maximum number of documents the system will support

1. Create the `schema.xml` and `solrconfig.xml` files.

2. Start a node.
3. Add *N* docs.
4. Run a range of queries that simulate a production environment.
5. View the size of the index (on disk) included in the status information about the Solr core.
6. Based on the server's system IO cache available, set a maximum index size per server.
7. Based on the available system memory, set a maximum heap size required per server.

   DataStax recommends the following heap sizes:

   - System memory less than 64 GB: 24 GB
   - System memory greater than 64 GB: 30 GB

   For faster live indexing you should configure live indexing ((RT) postings to be allocated offheap.

   Note: Enable live indexing on only one search core per cluster.

8. Calculate the maximum number of documents per node based on 6 *(page 11)* and 7 *(page 11)*.

   When the system is approaching the maximum docs per node, add more nodes.

Estimating partition size for DataStax Enterprise

For efficient operation, partitions must be sized within certain limits in DataStax Enterprise. Two measures of partition size are the number of values in a partition and the partition size on disk. The practical limit of cells per partition is two billion. Sizing the disk space is more complex, and involves the number of rows and the number of columns, primary key columns and static columns in each table. Each application will have different efficiency parameters, but a good rule of thumb is to keep the maximum number of rows below 100,000 items and the disk size under 100 MB.

Estimating usable disk capacity for DataStax Enterprise

To estimate how much data your DataStax Enterprise nodes can hold, calculate the usable disk capacity per node and then multiply that by the number of nodes in your cluster. Typically in a production cluster, the commit log and data directories are on different disks.

1. Start with the raw capacity of the physical disks:

   ```
   raw_capacity = disk_size * number_of_data_disks
   ```

2. Calculate the usable disk space accounting for file system formatting overhead (roughly 10 percent):

   ```
   formatted_disk_space = (raw_capacity * 0.9)
   ```

3. Calculate the recommended working disk capacity:

   During normal operations, the database routinely requires disk capacity for compaction and repair operations. For optimal performance and cluster health, DataStax recommends not filling your disks to capacity, but running at 50% to 80% capacity depending on the compaction strategy and size of the compactions.

   ```
   usable_disk_space = formatted_disk_space * (0.5 to 0.8)
   ```

Anti-patterns in DataStax Enterprise

The following implementation or design patterns are ineffective and/or counterproductive in DataStax Enterprise production installations. Correct patterns are suggested in most cases.

Storage area network

DataStax **strongly recommends against** using SAN storage for on-premises deployments.

> **Note:** Storage in clouds works very differently and is not an issue.

Although used frequently in Enterprise IT environments, SAN storage has proven to be a difficult and expensive architecture to use with distributed databases for a variety of reasons, including:

- SAN ROI (return on investment) does not scale along with that of DataStax Enterprise, in terms of capital expenses and engineering resources.
- In distributed architectures, SAN generally introduces a bottleneck and single point of failure because the database's IO frequently surpasses the array controller's ability to keep up.
- External storage, even when used with a high-speed network and SSD, adds latency for all operations.
- Heap pressure is increased because pending I/O operations take longer.
- When the SAN transport shares operations with internal and external DataStax Enterprise traffic, it can saturate the network and lead to network availability problems.

Taken together these factors can create problems that are difficult to resolve in production. In particular, new users deploying DataStax Enterprise with SAN must first develop adequate methods and allocate sufficient engineering resources to identify these issues before they become a problem in production. For example, methods are needed for all key scaling factors, such as operational rates and SAN fiber saturation.

Caution: If you still intend to use SAN, contact the DataStax Services team for detailed information about why it's not recommended.

Network attached storage

DataStax does **not** recommend storing SSTables on a network attached storage (NAS) device. Using a NAS device often results in network related bottlenecks resulting from high levels of I/O wait time on both reads and writes. The causes of these bottlenecks include:

- Router latency.
- The Network Interface Card (NIC) in the node.
- The NIC in the NAS device.

Attention: If you are required to use NAS for your environment, contact the DataStax Services team.

CPU frequency scaling

Recent Linux systems include a feature called *CPU frequency scaling* or *CPU speed scaling*. It allows a server's clock speed to be dynamically adjusted so that the server can run at lower clock speeds when the demand or load is low. This reduces the server's power consumption and heat output (which significantly impacts cooling costs). Unfortunately, this behavior has a detrimental effect on servers running DataStax Enterprise because throughput can get capped at a lower rate.

For more information, see Disable CPU frequency scaling.

Excessive heap space size

This information applies only to single-token architecture, not to virtual nodes. See Tuning Java Virtual Machine.

DSE's rack feature in single-token architecture deployments

This information applies only to single-token architecture, not to virtual nodes. SeeAbout data distribution and replication.

Defining one rack for the entire cluster is the simplest and most common implementation. Multiple racks should be avoided for the following reasons:

- Most users tend to ignore or forget rack requirements that racks should be organized in an alternating order. This order allows the data to get distributed safely and appropriately.
- Many users are not using the rack information effectively. For example, setting up with as many racks as nodes (or similar non-beneficial scenarios).
- Expanding a cluster when using racks can be tedious. The procedure typically involves several node moves and must ensure that racks are distributing data correctly and evenly. When clusters need immediate expansion, racks should be the last concern.

To use racks correctly:

- Use the same number of nodes in each rack.

- Use one rack and place the nodes in different racks in an alternating pattern. The rack feature benefits from quick and fully functional cluster expansions. Once the cluster is stable, you can swap nodes and make the appropriate moves to ensure that nodes are placed in the ring in an alternating fashion with respect to the racks.

SELECT ... IN or index lookups

SELECT ... IN and index lookups (formerly secondary indexes) should be avoided except for specific scenarios. See *When not to use IN* in SELECT and *When not to use an index* in Indexing.

Using the Byte Ordered Partitioner

The Byte Ordered Partitioner (BOP) is not recommended.

Use virtual nodes instead. Vnodes allow each node to own a large number of small ranges distributed throughout the cluster. Using vnodes saves you the effort of generating tokens and assigning tokens to your nodes. If not using vnodes, these partitioners are recommended because all writes occur on the hash of the key and are therefore spread out throughout the ring amongst tokens range. These partitioners ensure that your cluster evenly distributes data by placing the key at the correct token using the key's hash value.

Reading before writing

Reads take time for every request, as they typically have multiple disk hits for uncached reads. In work flows requiring reads before writes, this small amount of latency can affect overall throughput. All write I/O in the database is sequential so there is very little performance difference regardless of data size or key distribution.

Load balancers

The database was designed to avoid the need for load balancers. Putting load balancers between DataStax Enterprise and clients is harmful to performance, cost, availability, debugging, testing, and scaling. All high-level clients, such as the Java and Python drivers for DataStax Enterprise, implement load balancing directly. For available drivers, see DataStax drivers.

Insufficient testing

Be sure to test at scale and production loads. This the best way to ensure your system will function properly when your application goes live. The information you gather from testing is the best indicator of what throughput per node is needed for future expansion calculations.

To properly test, see Testing your DataStax Enterprise cluster before production *(page 22)*. Also look at the Netflix case study, which shows an excellent testing example.

Too many tables

Substantial performance degradation can be caused by having too many tables in a cluster. Because performance is influenced by a range of other factors, it is difficult to determine a universal table threshold; however, Datastax provides the following guidelines:

- **Warning threshold**: 200 tables. The cluster may run smoothly above this threshold, but once your database reaches the warning threshold, it's time to start monitoring and planning a re-architecture. If possible, remove unused and underutilized tables.
- **Failure threshold**: 500 tables. On a cluster that has exceeded 500 tables, expect problems and errors, including (but not limited) issues related to high memory usage and compactions.

Note: The table thresholds have additional dependencies on JVM Heap and the byte count. Each table uses approximately 1 MB of memory. For each table being acted on, there is a memtable representation in JVM Heap. Tables with large data models increase pressure on memory. Each keyspace also causes additional overhead in JVM memory; therefore having lots of keyspaces may also reduce the table threshold.

Lack of familiarity with Linux

Linux has a great collection of tools. Become familiar with the Linux built-in tools. It will help you greatly and ease operation and management costs in normal, routine functions. The essential list of tools and techniques to learn are:

- Parallel SSH and Cluster SSH: The pssh and cssh tools allow SSH access to multiple nodes. This is useful for inspections and cluster wide changes.
- Passwordless SSH: SSH authentication is carried out by using public and private keys. This allows SSH connections to easily hop from node to node without password access. In cases where more security is required, you can implement a bastion host and/or VPN.
- Useful common command-line tools include:
  # dstat: Shows all system resources instantly. For example, you can compare disk usage in combination with interrupts from your IDE controller, or compare the network bandwidth numbers directly with the disk throughput (in the same interval).
  # top: Provides an ongoing look at CPU processor activity in real time.
  # System performance tools: Tools such as iostat, mpstat, iftop, sar, lsof, netstat, htop, vmstat, and similar can collect and report a variety of metrics about the operation of the system.
  # vmstat: Reports information about processes, memory, paging, block I/O, traps, and CPU activity.
  # iftop: Shows a list of network connections. Connections are ordered by bandwidth usage, with the pair of hosts responsible for the most traffic at the top of list. This tool makes it easier to identify the hosts causing network congestion.

Running without the recommended settings

Be sure to use the Recommended settings.

Using DSE Search on DSE Tiered Storage tables

DataStax recommends not using DSE Search on DSE Tiered Storage tables. The data sets used by DSE Tiered Storage can be very large. Common use cases appropriate for DSE Tiered Storage are described in this DataStax blog post.

Planning a DataStax Enterprise cluster on Amazon EC2

Before planning an Amazon EC2 cluster, read Amazon EC2 - Virtual Server Hosting.

> **Attention:** These recommendations are guidelines. To ensure your implementation meets expectations, DataStax recommends contacting the DataStax Services team and testing your configuration thoroughly before deployment.

DataStax AMI deployments

DataStax (DSE) no longer hosts the DataStax ComboAMI. You can install DataStax Enterprise in two ways:

- Create your instances using an AMI for a supported platform and from a trusted source *(page 18)*. Then use the appropriate install method for your platform.
- Use the Lifecycle Manager in DSE OpsCenter to easily provision a DataStax Enterprise cluster for versions 4.7 and later:

  1. Create your instances using an AMI for a supported platform and from a trusted source *(page 18)*.
  2. Use the Lifecycle Manager to provision and configure your cluster.

Use AMIs from trusted sources

Use only AMIs for supported platforms and from a trusted source. Random AMIs pose a security risk and may perform slower than expected due to the way the EC2 install is configured. The following are examples of trusted AMIs:

- Ubuntu Amazon EC2 AMI Locator
- Debian AmazonEC2Image
- CentOS-6 images on Amazon's EC2 Cloud

EC2 deployments for multiple regions/availability zones

For these deployments use any of the supported platform on each node:

It is best practice to use the same platform on all nodes. If your cluster was instantiated using the DataStax AMI (no longer supported), use Ubuntu for the additional nodes. Configure the cluster as a multiple datacenter cluster using the Configuring Amazon EC2 multi-region snitch.

Guidelines for EC2 production clusters

DSE requires 10,000 IOPS (Input/Ouput Operations Per Second) minimum per node. The AWS storage choices for achieving this performance level are:

EBS General Purpose SSD (gp2) volumes

To achieve IO required by DSE, you must use 3.5 TB volumes, regardless of the actual space used, because gp2 provides 3 IOPS per GB.

Amazon EBS Provisioned IOPS SSD (io1) volumes
EBS io1 with 10,000 provisioned IOPS (PIOPS) provides the same performance level as gp2 using smaller volumes but at a higher cost.

Directly attached local SSDs
Also called ephemeral or instance SSDs. This storage type makes i3 instances the optimal cost versus performance choice. See pricing in Amazon AWS.

Use these guidelines for choosing the instance types:

- Light production with only transactional nodes and very light-weight usage: m4.2xlarge (absolute minimum). Also suitable for development.
- Moderate production: i3.4xlarge
- Large production: i3.8xlarge
- DSE Search and DSE Analytic nodes: i3.4xlarge or i3.8xlarge

Note: In EC2, each vCPU is a hyperthread of an Intel Xeon core, which means that two virtual cores exist on one physical core. For example, an i3.8xlarge instance type has 32vcCPUs, which is the equivalent of 16 physical cores.

EBS volumes recommended for m4 instance types

SSD-backed general purpose volumes (**GP2**) or provisioned IOPS volumes (**io1**) are suitable for production workloads when using m4 instances (transactional nodes with very light-weight usage). These volume types are designed to deliver consistent, low latency performance:

GP2	PIOPS
• The best choice for most workloads and have the advantage of guaranteeing 10,000 IOPS when volumes larger than 3.5TB are attached to instances. • Designed to deliver single-digit millisecond latencies. • Designed to deliver the provisioned performance 99.0% of the time.	• Designed to deliver single-digit millisecond latencies. • Designed to deliver the provisioned performance 99.9% of the time.

EBS magnetic volumes not recommended

EBS magnetic volumes are not recommended for DataStax Enterprise data storage volumes for the following reasons:

- EBS magnetic volumes contend directly for network throughput with standard packets. This contention means that EBS throughput is likely to fail when a network link is saturated.
- EBS magnetic volumes have unreliable performance. I/O performance can be exceptionally slow, causing the system to back load reads and writes until the entire cluster becomes unresponsive.

- Adding capacity by increasing the number of EBS volumes per host does not scale. You can easily surpass the ability of the system to keep effective buffer caches and concurrently serve requests for all of the data it is responsible for managing.

  Note: Use only ephemeral **instance-store** or the recommended EBS volume types for DataStax Enterprise data storage.

For more information and graphs related to ephemeral versus EBS performance, see Systematic Look at EC2 I/O.

Disk Performance Optimization

To ensure high disk performance to mounted drives, it is recommended that you pre-warm your drives by writing once to every drive location before production use. Depending on EC2 conditions, you can get moderate to enormous increases in throughput. See Optimizing Disk Performance in the *Amazon Elastic Compute Cloud Documentation*.

Storage recommendations

DataStax Enterprise supports JBOD (just a bunch of disks). JBOD excels at tolerating partial failures in a disk array. Configure the disk_failure_policy in cassandra.yaml. See Recovering from a single disk failure using JBOD.

Note: JBOD support allows you to use standard disks. However, RAID0 may provide better throughput because it splits every block to be on another device so that writes are written in parallel fashion instead of written serially on disk.

EC2 security group

When deploying DataStax Enterprise on EC2, you must create security rules that open ports to other nodes in the same security group. An EC2 Security Group acts as a firewall that allows you to choose which protocols and ports are open in your cluster. You can specify the protocols and ports either by a range of IP addresses or by security group. For more information, see the Amazon EC2 help on Security Groups.

Warning: Specifying a Source IP of 0.0.0.0/0 opens externally accessible ports to incoming traffic from any IP address. The risk of data loss is high.

The Securing DataStax Enterprise ports table provides a list of ports that should be opened to internode and client communications.

Note: Generally, when you have firewalls between machines, it is difficult to run JMX across a network and maintain security. This is because JMX connects on port 7199, handshakes, and then uses any port within the 1024+ range. Instead use SSH to execute commands remotely to connect to JMX locally or use the DSE OpsCenter.

Other resources

- Amazon Web Services EC2 documentation
- What is the story with AWS storage

- Get in the Ring with Cassandra and EC2

Testing your DataStax Enterprise cluster before production

To ensure that your selections of memory, CPU, disks, number of nodes, and network settings meet your business needs, test your cluster prior to production. By simulating your production workload, you can avoid unpleasant surprises and ensure successful production deployment.

Your testing workload should match the production workload as closely as possible. Variances from production in testing must fully account for any differences between workloads.

Use the following guidelines for testing:

Considerations if you're coming from a relational database world

DataStax Enterprise-based applications and clusters are much different than relational databases and use a data model based on the types of queries, not on modeling entities and relationships. Besides learning a new database technology and API, writing an application that scales on a distributed database instead of a single machine requires a new way of thinking.

A common strategy new users employ for making the transition from single machine to distributed node is to make heavy use of materialized views, secondary indexes, and UDF (user defined functions). It’s important to realize that in most applications at large scale these techniques are not heavily used, and if they are, allowances must be made for the performance penalty in node sizing and SLAs.

Service level agreements targets

Before going into production, decide what constitutes fast and slow Service level agreements (SLA), and how often slow SLAs are allowed to occur.

Table 2: Basic rules for determining request percentiles

Percentile	Use case
100	Not recommended: The worst measurement is often irrelevant to testing and based on factors external to DataStax Enterprise.
99.9	Aggressive: Use only for latency sensitive use cases. Be aware that hardware expenses will be high if you need to keep this close to average.
99	Sweet spot and the most common.
95	Loose: Use when where latency is not easily noticed this is a good metric to measure.

The following metrics are based on clusters with more data on disk than RAM. Measurement is in the 99th percentile using nodetool tablehistograms.

Table 3: Latency metrics

Hardware	Latency
Top-of-the-line high CPU nodes with NVM flash storage	< 10ms
Common SATA based SSDs	< 150ms Factors, such as testing, controller, and contention, can shift this measurement significantly. This can result in some hardware approaching NVM specifications.
Common SATA based-systems	500ms is common. This metric varies significantly due to controller, contention, data model, and so on. Tuning can shift this figure an order of magnitude.

Establish TPS targets

Determine how many transactions per second (TPS) you expect your cluster to handle. Ideally, match this number with the same node count in your test cluster. Alternately, test with ratio of TPS to nodes to evaluate how the cluster behaves.

If the cluster does not meet your SLA requirements, look for the root cause. Helpful questions include:

- Does the cluster meet SLAs below that level?
- At what TPS level do your SLAs fail?

Monitor the important metrics

Use DSE OpsCenter to monitor cluster metrics. You can also use JMX MBeans or Linux `iostat` or `dstat` commands. Monitor the following metrics:

System hints
Generated system hints should not be continually high. If a node is down or a datacenter link is down, expect this metric to accumulate (by design).

Pending compactions
The number of acceptable pending compactions depends on the type of compaction strategy.

Pending writes
Should not go over 10,000 writes for more than short periods. Otherwise, a tuning or sizing problem exists.

heap usage
If huge spikes with long pauses exist, adjust the heap.

I/O wait
Metrics greater than 5 indicates that the CPU is waiting too long on I/O and is a sign of being I/O bound.

CPU Usage
For CPUs using hyper-threading, it is helpful to remember that for each processor core that is physically present, the operating system addresses two virtual (logical) cores and shares the workload between them. This means that if half your threads are at 100% CPU usage, the CPU capacity is at maximum even if the tool reports 50% usage. To see the actual thread usage, use the ttop command.

Monitor your SLAs
Monitor according to the targets you set above *(page 22)*.

SSTable counts
If SSTable counts go over 10,000, see CASSANDRA-11831.

> **Note:** DateTieredCompactionStrategy (DTCS) is deprecated.

Log warnings and errors
Look for the following:

- SlabPoolCleaner
- WARN
- ERROR
- GCInspector
- hints

Simulate real outages

Simulating outages is key to ensuring that your cluster functions well in production. To simulate outages:

1. Take one or two nodes offline (more is preferable, depending on the cluster size).
   - Do your queries start failing?

- Do the nodes start working without any change when they come back online?

2. Take a datacenter offline while taking load.

   How long can it be offline before system.hints build up to unmanageable levels?

3. Run an expensive task that eats up CPU on a couple of nodes, such as find "a /

   This scenario helps you prepare for the effect on your nodes in case this happens.

4. Do expensive range queries in CQLSH while taking load to see how the cluster behaves. For example:

   ```
   PAGING OFF;
   SELECT * FROM FOO LIMIT 100000000;
   ```

   This test ensures your cluster is ready for naive or unintended actions by users.

Simulate production operations

Do the following **during load testing** and repeat **under outages** as described above:

1. Run repair to ensure that you have enough overhead to complete the repair.
2. Bootstrap a new node into the cluster.
3. Add a new datacenter. This exercise gives you an idea of the process when you need to do it in production.
4. Change the schema by adding new tables, removing tables, and altering existing tables.
5. Replace a *failed* node. Note how long it takes to bootstrap a new node.

Run a real replication factor on at least 5 nodes

Ideally run the same number of nodes that you expect to run in production. Moreover, if you're planning on running only 3 nodes in production, test with 5. You may find that the queries that were fast with 3 nodes are much slower with 5 nodes. In this scenario, secondary indexes as a common query path are exposed as being slower at scale.

Be sure to use the same replication factor as in production.

Simulate the production datacenter setup

Each datacenter adds additional write costs. If your production cluster has more datacenters than your test cluster, your writes might be greater than you anticipate. For example, if you use RF 3 and test your cluster with 2 datacenters, each write goes to 4 nodes in 2 data centers. If your production cluster uses the same RF and has 5 datacenters, each write goes to 7 nodes with 5 data centers. This means writes are now almost twice as expensive on the coordinator

node because in each remote datacenter, the forwarding coordinator sends the data to the other replicas in it's datacenter.

Simulate the network topology

Simulate your network topology with the same latency and pipe. Test with the expected TPS. Be aware that remote links can be a limiting factor for many use cases and expected bandwidth.

Simulate expected per node density

> **Note:** Generally, unless you're using SSDs, a high-numbered CPUs, significant RAM, using more than a terabyte (TB) per node is rarely a good idea. See Capacity per node recommendations *(page 7)*.

Once you've determined your target per node density, add data to the test cluster until the target capacity is reached. Test for the following:

- What happens to your SLAs as the target density is approached?
- What happens if you go over your the density?
- How long does bootstrap take?
- How long does with repair take?

Use the cassandra-stress tool for testing

The cassandra-stress tool is a Java-based stress testing utility for cluster load testing and basic benchmarking. Use this tool to test hardware choices and discover issues with data modeling. The cassandra-stress tool also supports a YAML-based profile for testing reads, writes, and mixed workloads, plus the ability to specify schemas with various compaction strategies, cache settings, and types.