Apache Cassandra : cassandra.yaml (Latest)

cassandra.yaml file configuration

cassandra.yaml file configuration

cluster_name

NOTE: This file is provided in two versions: - cassandra.yaml: Contains configuration defaults for a "compatible" configuration that operates using settings that are backwards-compatible and interoperable with machines running older versions of Cassandra. This version is provided to facilitate pain-free upgrades for existing users of Cassandra running in production who want to gradually and carefully introduce new features. - cassandra_latest.yaml: Contains configuration defaults that enable the latest features of Cassandra, including improved functionality as well as higher performance. This version is provided for new users of Cassandra who want to get the most out of their cluster, and for users evaluating the technology. To use this version, simply copy this file over cassandra.yaml, or specify it using the -Dcassandra.config system property, e.g. by running cassandra -Dcassandra.config=file://$CASSANDRA_HOME/conf/cassandra_latest.yaml /NOTE

The name of the cluster. This is mainly used to prevent machines in one logical cluster from joining another.

Default Value: 'Test Cluster'

num_tokens

This defines the number of tokens randomly assigned to this node on the ring The more tokens, relative to other nodes, the larger the proportion of data that this node will store. You probably want all nodes to have the same number of tokens assuming they have equal hardware capability.

If you leave this unspecified, Cassandra will use the default of 1 token for legacy compatibility, and will use the initial_token as described below.

Specifying initial_token will override this setting on the node's initial start, on subsequent starts, this setting will apply even if initial token is set.

See https://cassandra.apache.org/doc/latest/getting-started/production.html#tokens for best practice information about num_tokens.

Default Value: 16

allocate_tokens_for_keyspace

This option is commented out by default.

Triggers automatic allocation of num_tokens tokens for this node. The allocation algorithm attempts to choose tokens in a way that optimizes replicated load over the nodes in the datacenter for the replica factor.

The load assigned to each node will be close to proportional to its number of vnodes.

Only supported with the Murmur3Partitioner.

Replica factor is determined via the replication strategy used by the specified keyspace.

Default Value: KEYSPACE

allocate_tokens_for_local_replication_factor

Replica factor is explicitly set, regardless of keyspace or datacenter. This is the replica factor within the datacenter, like NTS.

Default Value: 3

initial_token

This option is commented out by default.

initial_token allows you to specify tokens manually. While you can use it with vnodes (num_tokens > 1, above) — in which case you should provide a comma-separated list — it's primarily used when adding nodes to legacy clusters that do not have vnodes enabled.

hinted_handoff_enabled

May either be "true" or "false" to enable globally

Default Value: true

hinted_handoff_disabled_datacenters

This option is commented out by default.

When hinted_handoff_enabled is true, a black list of data centers that will not perform hinted handoff

Default Value (complex option):

```
#    - DC1
#    - DC2
```

max_hint_window

this defines the maximum amount of time a dead host will have hints generated. After it has been dead this long, new hints for it will not be created until it has been seen alive and gone down again. Min unit: ms

Default Value: 3h

hinted_handoff_throttle

Maximum throttle in KiBs per second, per delivery thread. This will be reduced proportionally to the number of nodes in the cluster. (If there are two nodes in the cluster, each delivery thread will use the maximum rate; if there are three, each will throttle to half of the maximum, since we expect two nodes to be delivering hints simultaneously.) Min unit: KiB

Default Value: 1024KiB

max_hints_delivery_threads

Number of threads with which to deliver hints; Consider increasing this number when you have multi-dc deployments, since cross-dc handoff tends to be slower

Default Value: 2

hints_directory

This option is commented out by default.

Directory where Cassandra should store hints. If not set, the default directory is $CASSANDRA_HOME/data/hints.

Default Value: /var/lib/cassandra/hints

hints_flush_period

How often hints should be flushed from the internal buffers to disk. Will **not** trigger fsync. Min unit: ms

Default Value: 10000ms

max_hints_file_size

Maximum size for a single hints file, in mebibytes. Min unit: MiB

Default Value: 128MiB

max_hints_size_per_host

This option is commented out by default.

The file size limit to store hints for an unreachable host, in mebibytes. Once the local hints files have

reached the limit, no more new hints will be created. Set a non-positive value will disable the size limit.

Default Value: 0MiB

auto_hints_cleanup_enabled

Enable / disable automatic cleanup for the expired and orphaned hints file. Disable the option in order to preserve those hints on the disk.

Default Value: false

transfer_hints_on_decommission

This option is commented out by default.

Enable/disable transfering hints to a peer during decommission. Even when enabled, this does not guarantee consistency for logged batches, and it may delay decommission when coupled with a strict hinted_handoff_throttle. Default: true

Default Value: true

hints_compression

This option is commented out by default.

Compression to apply to the hint files. If omitted, hints files will be written uncompressed. LZ4, Snappy, and Deflate compressors are supported.

Default Value (complex option):

```
    #   - class_name: LZ4Compressor
    #     parameters:
    #         -
```

heap_dump_path

This option is commented out by default. Directory where Cassandra should store results of a One-Shot troubleshooting heapdump for uncaught exceptions. Note: this value can be overridden by the -XX:HeapDumpPath JVM env param with a relative local path for testing if so desired. If not set, the default directory is $CASSANDRA_HOME/heapdump

Default Value: /var/lib/cassandra/heapdump

dump_heap_on_uncaught_exception

This option is commented out by default.

Enable / disable automatic dump of heap on first uncaught exception If not set, the default value is false

Default Value: true

hint_window_persistent_enabled

This option is commented out by default.

Enable / disable persistent hint windows.

If set to false, a hint will be stored only in case a respective node that hint is for is down less than or equal to max_hint_window.

If set to true, a hint will be stored in case there is not any hint which was stored earlier than max_hint_window. This is for cases when a node keeps to restart and hints are not delivered yet, we would be saving hints for that node indefinitely.

Defaults to true.

Default Value: true

batchlog_replay_throttle

Maximum throttle in KiBs per second, total. This will be reduced proportionally to the number of nodes in the cluster. Min unit: KiB

Default Value: 1024KiB

batchlog_endpoint_strategy

Strategy to choose the batchlog storage endpoints.

Available options:

- random_remote Default, purely random, prevents the local rack, if possible.
- prefer_local Similar to random_remote. Random, except that one of the replications will go to the local rack, which mean it offers lower availability guarantee than random_remote or dynamic_remote.
- dynamic_remote Using DynamicEndpointSnitch to select batchlog storage endpoints, prevents the local rack, if possible. This strategy offers the same availability guarantees as random_remote but selects the fastest endpoints according to the DynamicEndpointSnitch. (DynamicEndpointSnitch

currently only tracks reads and not writes - i.e. write-only (or mostly-write) workloads might not benefit from this strategy.) Note: this strategy will fall back to random_remote, if dynamic_snitch is not enabled.

- dynamic Mostly the same as dynamic_remote, except that local rack is not excluded, which mean it offers lower availability guarantee than random_remote or dynamic_remote. Note: this strategy will fall back to random_remote, if dynamic_snitch is not enabled.

Default Value: dynamic_remote

authenticator

Authentication backend, implementing IAuthenticator; used to identify users Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllAuthenticator, PasswordAuthenticator}. Optional parameters can be specified in the form of: parameters: param_key1: param_value1 ...

- AllowAllAuthenticator performs no checks - set it to disable authentication.
- PasswordAuthenticator relies on username/password pairs to authenticate users. It keeps usernames and hashed passwords in system_auth.roles table. Please increase system_auth keyspace replication factor if you use this authenticator. If using PasswordAuthenticator, CassandraRoleManager must also be used (see below)

authorizer

MutualTlsAuthenticator can be configured using the following configuration. One can add their own validator which implements MutualTlsCertificateValidator class and provide logic for extracting identity out of certificates and validating certificates. class_name : org.apache.cassandra.auth.MutualTlsAuthenticator parameters : validator_class_name: org.apache.cassandra.auth.SpiffeCertificateValidator

Authorization backend, implementing IAuthorizer; used to limit access/provide permissions Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllAuthorizer, CassandraAuthorizer}. Optional parameters can be specified in the form of: parameters: param_key1: param_value1 ...

- AllowAllAuthorizer allows any action to any user - set it to disable authorization.
- CassandraAuthorizer stores permissions in system_auth.role_permissions table. Please increase system_auth keyspace replication factor if you use this authorizer.

role_manager

Part of the Authentication & Authorization backend, implementing IRoleManager; used to maintain grants and memberships between roles. Out of the box, Cassandra provides org.apache.cassandra.auth.CassandraRoleManager, which stores role information in the system_auth keyspace. Most functions of the IRoleManager require an authenticated login, so unless the configured IAuthenticator actually implements authentication, most of this functionality will be unavailable.

Optional parameters can be specified in the form of: parameters: param_key1: param_value1 ...

- CassandraRoleManager stores role data in the system_auth keyspace. Please increase system_auth keyspace replication factor if you use this role manager.

network_authorizer

Network authorization backend, implementing INetworkAuthorizer; used to restrict user access to certain DCs Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllNetworkAuthorizer, CassandraNetworkAuthorizer}. Optional parameters can be specified in the form of: parameters: param_key1: param_value1 ...

- AllowAllNetworkAuthorizer allows access to any DC to any user - set it to disable authorization.
- CassandraNetworkAuthorizer stores permissions in system_auth.network_permissions table. Please increase system_auth keyspace replication factor if you use this authorizer.

cidr_authorizer

CIDR authorization backend, implementing ICIDRAuthorizer; used to restrict user access from certain CIDRs Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllCIDRAuthorizer, CassandraCIDRAuthorizer}. Optional parameters can be specified in the form of: parameters: param_key1: param_value1 ...

- AllowAllCIDRAuthorizer allows access from any CIDR to any user - set it to disable CIDR authorization.
- CassandraCIDRAuthorizer stores user's CIDR permissions in system_auth.cidr_permissions table. Please increase system_auth keyspace replication factor if you use this authorizer, otherwise any changes to system_auth tables being used by this feature may be lost when a host goes down.

traverse_auth_from_root

This option is commented out by default.

Depending on the auth strategy of the cluster, it can be beneficial to iterate from root to table (root → ks → table) instead of table to root (table → ks → root). As the auth entries are whitelisting, once a permission is found you know it to be valid. We default to false as the legacy behavior is to query at the table level then move back up to the root. See CASSANDRA-17016 for details.

Default Value: false

roles_validity

Validity period for roles cache (fetching granted roles can be an expensive operation depending on the role manager, CassandraRoleManager is one example) Granted roles are cached for authenticated

sessions in AuthenticatedUser and after the period specified here, become eligible for (async) reload. Defaults to 2000, set to 0 to disable caching entirely. Will be disabled automatically for AllowAllAuthenticator. For a long-running cache using roles_cache_active_update, consider setting to something longer such as a daily validation: 86400000 Min unit: ms

Default Value: 2000ms

roles_update_interval

This option is commented out by default.

Refresh interval for roles cache (if enabled). After this interval, cache entries become eligible for refresh. Upon next access, an async reload is scheduled and the old value returned until it completes. If roles_validity is non-zero, then this must be also. This setting is also used to inform the interval of auto-updating if using roles_cache_active_update. Defaults to the same value as roles_validity. For a long-running cache, consider setting this to 60000 (1 hour) etc. Min unit: ms

Default Value: 2000ms

roles_cache_active_update

This option is commented out by default.

If true, cache contents are actively updated by a background task at the interval set by roles_update_interval. If false, cache entries become eligible for refresh after their update interval. Upon next access, an async reload is scheduled and the old value returned until it completes.

Default Value: false

permissions_validity

Validity period for permissions cache (fetching permissions can be an expensive operation depending on the authorizer, CassandraAuthorizer is one example). Defaults to 2000, set to 0 to disable. Will be disabled automatically for AllowAllAuthorizer. For a long-running cache using permissions_cache_active_update, consider setting to something longer such as a daily validation: 86400000ms Min unit: ms

Default Value: 2000ms

permissions_update_interval

This option is commented out by default.

Refresh interval for permissions cache (if enabled). After this interval, cache entries become eligible for refresh. Upon next access, an async reload is scheduled and the old value returned until it completes. If permissions_validity is non-zero, then this must be also. This setting is also used to inform

the interval of auto-updating if using permissions_cache_active_update. Defaults to the same value as permissions_validity. For a longer-running permissions cache, consider setting to update hourly (60000) Min unit: ms

Default Value: 2000ms

permissions_cache_active_update

This option is commented out by default.

If true, cache contents are actively updated by a background task at the interval set by permissions_update_interval. If false, cache entries become eligible for refresh after their update interval. Upon next access, an async reload is scheduled and the old value returned until it completes.

Default Value: false

credentials_validity

Validity period for credentials cache. This cache is tightly coupled to the provided PasswordAuthenticator implementation of IAuthenticator. If another IAuthenticator implementation is configured, this cache will not be automatically used and so the following settings will have no effect. Please note, credentials are cached in their encrypted form, so while activating this cache may reduce the number of queries made to the underlying table, it may not bring a significant reduction in the latency of individual authentication attempts. Defaults to 2000, set to 0 to disable credentials caching. For a long-running cache using credentials_cache_active_update, consider setting to something longer such as a daily validation: 86400000 Min unit: ms

Default Value: 2000ms

credentials_update_interval

This option is commented out by default.

Refresh interval for credentials cache (if enabled). After this interval, cache entries become eligible for refresh. Upon next access, an async reload is scheduled and the old value returned until it completes. If credentials_validity is non-zero, then this must be also. This setting is also used to inform the interval of auto-updating if using credentials_cache_active_update. Defaults to the same value as credentials_validity. For a longer-running permissions cache, consider setting to update hourly (60000) Min unit: ms

Default Value: 2000ms

credentials_cache_active_update

This option is commented out by default.

If true, cache contents are actively updated by a background task at the interval set by credentials_update_interval. If false (default), cache entries become eligible for refresh after their update interval. Upon next access, an async reload is scheduled and the old value returned until it completes.

Default Value: false

partitioner

The partitioner is responsible for distributing groups of rows (by partition key) across nodes in the cluster. The partitioner can NOT be changed without reloading all data. If you are adding nodes or upgrading, you should set this to the same partitioner that you are currently using.

The default partitioner is the Murmur3Partitioner. Older partitioners such as the RandomPartitioner, ByteOrderedPartitioner, and OrderPreservingPartitioner have been included for backward compatibility only. For new clusters, you should NOT change this value.

Default Value: org.apache.cassandra.dht.Murmur3Partitioner

data_file_directories

This option is commented out by default.

Directories where Cassandra should store data on disk. If multiple directories are specified, Cassandra will spread data evenly across them by partitioning the token ranges. If not set, the default directory is $CASSANDRA_HOME/data/data.

Default Value (complex option):

```
#     - /var/lib/cassandra/data
```

local_system_data_file_directory

This option is commented out by default. Directory were Cassandra should store the data of the local system keyspaces. By default Cassandra will store the data of the local system keyspaces in the first of the data directories specified by data_file_directories. This approach ensures that if one of the other disks is lost Cassandra can continue to operate. For extra security this setting allows to store those data on a different directory that provides redundancy.

commitlog_directory

This option is commented out by default.

commit log. when running on magnetic HDD, this should be a separate spindle than the data

directories. If not set, the default directory is $CASSANDRA_HOME/data/commitlog.

Default Value: /var/lib/cassandra/commitlog

cdc_enabled

Enable / disable CDC functionality on a per-node basis. This modifies the logic used for write path allocation rejection (standard: never reject. cdc: reject Mutation containing a CDC-enabled table if at space limit in cdc_raw_directory).

Default Value: false

cdc_block_writes

This option is commented out by default.

Specify whether writes to the CDC-enabled tables should be blocked when CDC data on disk has reached to the limit. When setting to false, the writes will not be blocked and the oldest CDC data on disk will be deleted to ensure the size constraint. The default is true.

Default Value: true

cdc_on_repair_enabled

This option is commented out by default.

Specify whether CDC mutations are replayed through the write path on streaming, e.g. repair. When enabled, CDC data streamed to the destination node will be written into commit log first. When setting to false, the streamed CDC data is written into SSTables just the same as normal streaming. The default is true. If this is set to false, streaming will be considerably faster however it's possible that, in extreme situations (losing > quorum # nodes in a replica set), you may have data in your SSTables that never makes it to the CDC log.

Default Value: true

cdc_raw_directory

This option is commented out by default.

CommitLogSegments are moved to this directory on flush if cdc_enabled: true and the segment contains mutations for a CDC-enabled table. This should be placed on a separate spindle than the data directories. If not set, the default directory is $CASSANDRA_HOME/data/cdc_raw.

Default Value: /var/lib/cassandra/cdc_raw

disk_access_mode

This option is commented out by default.

Policy for accessing disk:

auto Enable mmap on both data and index files on a 64-bit JVM.

standard Disable mmap entirely.

mmap Map index and data files. mmap can cause excessive paging if all actively read SSTables do not fit into RAM.

mmap_index_only Similar to mmap but maps only index files. Using this setting might also help if you observe high number of page faults or steals along with increased latencies. This setting is default.

Default Value: mmap_index_only

disk_failure_policy

Policy for data disk failures:

die shut down gossip and client transports and kill the JVM for any fs errors or single-sstable errors, so the node can be replaced.

stop_paranoid shut down gossip and client transports even for single-sstable errors, kill the JVM for errors during startup.

stop shut down gossip and client transports, leaving the node effectively dead, but can still be inspected via JMX, kill the JVM for errors during startup.

best_effort stop using the failed disk and respond to requests based on remaining available sstables. This means you WILL see obsolete data at CL.ONE!

ignore ignore fatal errors and let requests fail, as in pre-1.2 Cassandra

Default Value: stop

commit_failure_policy

Policy for commit disk failures:

die shut down the node and kill the JVM, so the node can be replaced.

stop shut down the node, leaving the node effectively dead, but can still be inspected via JMX.

stop_commit shutdown the commit log, letting writes collect but continuing to service reads, as in pre-2.0.5 Cassandra

ignore ignore fatal errors and let the batches fail

Default Value: stop

prepared_statements_cache_size

Maximum size of the native protocol prepared statement cache

Valid values are either "auto" (omitting the value) or a value greater 0.

Note that specifying a too large value will result in long running GCs and possbily out-of-memory errors. Keep the value at a small fraction of the heap.

If you constantly see "prepared statements discarded in the last minute because cache limit reached" messages, the first step is to investigate the root cause of these messages and check whether prepared statements are used correctly - i.e. use bind markers for variable parts.

Do only change the default value, if you really have more prepared statements than fit in the cache. In most cases it is not neccessary to change this value. Constantly re-preparing statements is a performance penalty.

Default value ("auto") is 1/256th of the heap or 10MiB, whichever is greater Min unit: MiB

key_cache_size

Maximum size of the key cache in memory.

Each key cache hit saves 1 seek and each row cache hit saves 2 seeks at the minimum, sometimes more. The key cache is fairly tiny for the amount of time it saves, so it’s worthwhile to use it at large numbers. The row cache saves even more time, but must contain the entire row, so it is extremely space-intensive. It’s best to only use the row cache if you have hot rows or static rows.

NOTE	if you reduce the size, you may not get you hottest keys loaded on startup.

Default value is empty to make it "auto" (min(5% of Heap (in MiB), 100MiB)). Set to 0 to disable key cache.

This is only relevant to SSTable formats that use key cache, e.g. BIG.

This version of the configuration is intended for new installs and disables the key cache. Do not do this if you are upgrading from an older Cassandra version and still have sstables in BIG format as it can degrade performance. Min unit: MiB

Default Value: 0MiB

key_cache_save_period

This option is commented out by default.

Duration in seconds after which Cassandra should save the key cache. Caches are saved to saved_caches_directory as specified in this configuration file.

Saved caches greatly improve cold-start speeds, and is relatively cheap in terms of I/O for the key cache. Row cache saving is much more expensive and has limited use.

This is only relevant to SSTable formats that use key cache, e.g. BIG. Default is 14400 or 4 hours. Min unit: s

Default Value: 4h

key_cache_keys_to_save

This option is commented out by default.

Number of keys from the key cache to save Disabled by default, meaning all keys are going to be saved This is only relevant to SSTable formats that use key cache, e.g. BIG.

Default Value: 100

row_cache_class_name

This option is commented out by default.

Row cache implementation class name. Available implementations:

org.apache.cassandra.cache.OHCProvider Fully off-heap row cache implementation (default).

org.apache.cassandra.cache.SerializingCacheProvider This is the row cache implementation available in previous releases of Cassandra.

Default Value: org.apache.cassandra.cache.OHCProvider

row_cache_size

Maximum size of the row cache in memory. Please note that OHC cache implementation requires some additional off-heap memory to manage the map structures and some in-flight memory during operations before/after cache entries can be accounted against the cache capacity. This overhead is usually small compared to the whole capacity. Do not specify more memory that the system can afford in the worst usual situation and leave some headroom for OS block level cache. Do never allow your system to swap.

Default value is 0, to disable row caching. Min unit: MiB

Default Value: 0MiB

row_cache_save_period

Duration in seconds after which Cassandra should save the row cache. Caches are saved to saved_caches_directory as specified in this configuration file.

Saved caches greatly improve cold-start speeds, and is relatively cheap in terms of I/O for the key cache. Row cache saving is much more expensive and has limited use.

Default is 0 to disable saving the row cache. Min unit: s

Default Value: 0s

row_cache_keys_to_save

This option is commented out by default.

Number of keys from the row cache to save. Specify 0 (which is the default), meaning all keys are going to be saved

Default Value: 100

counter_cache_size

Maximum size of the counter cache in memory.

Counter cache helps to reduce counter locks' contention for hot counter cells. In case of RF = 1 a counter cache hit will cause Cassandra to skip the read before write entirely. With RF > 1 a counter cache hit will still help to reduce the duration of the lock hold, helping with hot counter cell updates, but will not allow skipping the read entirely. Only the local (clock, count) tuple of a counter cell is kept in memory, not the whole counter, so it’s relatively cheap.

NOTE	if you reduce the size, you may not get you hottest keys loaded on startup.

Default value is empty to make it "auto" (min(2.5% of Heap (in MiB), 50MiB)). Set to 0 to disable counter cache. NOTE: if you perform counter deletes and rely on low gcgs, you should disable the counter cache. Min unit: MiB

counter_cache_save_period

Duration in seconds after which Cassandra should save the counter cache (keys only). Caches are saved to saved_caches_directory as specified in this configuration file.

Default is 7200 or 2 hours. Min unit: s

Default Value: 7200s

counter_cache_keys_to_save

This option is commented out by default.

Number of keys from the counter cache to save Disabled by default, meaning all keys are going to be saved

Default Value: 100

saved_caches_directory

This option is commented out by default.

saved caches If not set, the default directory is $CASSANDRA_HOME/data/saved_caches.

Default Value: /var/lib/cassandra/saved_caches

cache_load_timeout

This option is commented out by default.

Number of seconds the server will wait for each cache (row, key, etc ...) to load while starting the Cassandra process. Setting this to zero is equivalent to disabling all cache loading on startup while still having the cache during runtime. Min unit: s

Default Value: 30s

commitlog_sync_group_window

This option is commented out by default.

commitlog_sync may be either "periodic", "group", or "batch."

When in batch mode, Cassandra won't ack writes until the commit log has been flushed to disk. Each incoming write will trigger the flush task.

group mode is similar to batch mode, where Cassandra will not ack writes until the commit log has been flushed to disk. The difference is group mode will wait up to commitlog_sync_group_window between flushes.

Min unit: ms

Default Value: 1000ms

commitlog_sync

the default option is "periodic" where writes may be acked immediately and the CommitLog is simply synced every commitlog_sync_period milliseconds.

Default Value: periodic

commitlog_sync_period

Min unit: ms

Default Value: 10000ms

periodic_commitlog_sync_lag_block

This option is commented out by default.

When in periodic commitlog mode, the number of milliseconds to block writes while waiting for a slow disk flush to complete. Min unit: ms

commitlog_segment_size

The size of the individual commitlog file segments. A commitlog segment may be archived, deleted, or recycled once all the data in it (potentially from each columnfamily in the system) has been flushed to sstables.

The default size is 32, which is almost always fine, but if you are archiving commitlog segments (see commitlog_archiving.properties), then you probably want a finer granularity of archiving; 8 or 16 MB is reasonable. Max mutation size is also configurable via max_mutation_size setting in cassandra.yaml. The default is half the size commitlog_segment_size in bytes. This should be positive and less than 2048.

NOTE	If max_mutation_size is set explicitly then commitlog_segment_size must be set to at least twice the size of max_mutation_size

Min unit: MiB

Default Value: 32MiB

commitlog_compression

This option is commented out by default.

Compression to apply to the commit log. If omitted, the commit log will be written uncompressed. LZ4, Snappy, and Deflate compressors are supported.

Default Value (complex option):

```
#   - class_name: LZ4Compressor
#     parameters:
#         -
```

commitlog_disk_access_mode

Set the disk access mode for writing commitlog segments. The allowed values are: - auto: version dependent optimal setting - legacy: the default mode as used in Cassandra 4.x and earlier (standard I/O when the commitlog is either compressed or encrypted or mmap otherwise) - mmap: use memory mapped I/O - available only when the commitlog is neither compressed nor encrypted - direct: use direct I/O - available only when the commitlog is neither compressed nor encrypted - standard: use standard I/O - available only when the commitlog is compressed or encrypted The default setting is legacy when the storage compatibility is set to 4 or auto otherwise.

Default Value: auto

table

This option is commented out by default.

Compression to apply to SSTables as they flush for compressed tables. Note that tables without compression enabled do not respect this flag.

As high ratio compressors like LZ4HC, Zstd, and Deflate can potentially block flushes for too long, the default is to flush with a known fast compressor in those cases. Options are:

none : Flush without compressing blocks but while still doing checksums. fast : Flush with a fast compressor. If the table is already using a fast compressor that compressor is used.

Default Value: Always flush with the same compressor that the table uses. This

flush_compression

This option is commented out by default. was the pre 4.0 behavior.

Default Value: fast

seed_provider

any class that implements the SeedProvider interface and has a constructor that takes a Map<String, String> of parameters will do.

Default Value (complex option):

```
      # Addresses of hosts that are deemed contact points.
      # Cassandra nodes use this list of hosts to find each other and learn
      # the topology of the ring.  You must change this if you are running
      # multiple nodes!
      - class_name: org.apache.cassandra.locator.SimpleSeedProvider
        parameters:
          # seeds is actually a comma-delimited list of addresses.
          # Ex: "<ip1>,<ip2>,<ip3>"
          - seeds: "127.0.0.1:7000"
          # If set to "true", SimpleSeedProvider will return all IP addresses for a DNS
name,
          # based on the configured name service on the system. Defaults to "false".
          #  resolve_multiple_ip_addresses_per_dns_record: "false"
```

concurrent_reads

For workloads with more data than can fit in memory, Cassandra's bottleneck will be reads that need to fetch data from disk. "concurrent_reads" should be set to (16 * number_of_drives) in order to allow the operations to enqueue low enough in the stack that the OS and drives can reorder them. Same applies to "concurrent_counter_writes", since counter writes read the current values before incrementing and writing them back.

On the other hand, since writes are almost never IO bound, the ideal number of "concurrent_writes" is dependent on the number of cores in your system; (8 * number_of_cores) is a good rule of thumb.

Default Value: 32

concurrent_writes

Default Value: 32

concurrent_counter_writes

Default Value: 32

concurrent_materialized_view_writes

For materialized view writes, as there is a read involved, so this should be limited by the less of concurrent reads or concurrent writes.

Default Value: 32

networking_cache_size

This option is commented out by default.

Maximum memory to use for inter-node and client-server networking buffers.

Defaults to the smaller of 1/16 of heap or 128MB. This pool is allocated off-heap, so is in addition to the memory allocated for heap. The cache also has on-heap overhead which is roughly 128 bytes per chunk (i.e. 0.2% of the reserved size if the default 64k chunk size is used). Memory is only allocated when needed. Min unit: MiB

Default Value: 128MiB

file_cache_enabled

This option is commented out by default.

Enable the sstable chunk cache. The chunk cache will store recently accessed sections of the sstable in-memory as uncompressed buffers.

Default Value: false

file_cache_size

This option is commented out by default.

Maximum memory to use for sstable chunk cache and buffer pooling. 32MB of this are reserved for pooling buffers, the rest is used for chunk cache that holds uncompressed sstable chunks. Defaults to the smaller of 1/4 of heap or 512MB. This pool is allocated off-heap, so is in addition to the memory allocated for heap. The cache also has on-heap overhead which is roughly 128 bytes per chunk (i.e. 0.2% of the reserved size if the default 64k chunk size is used). Memory is only allocated when needed. Min unit: MiB

Default Value: 512MiB

buffer_pool_use_heap_if_exhausted

This option is commented out by default.

Flag indicating whether to allocate on or off heap when the sstable buffer pool is exhausted, that is when it has exceeded the maximum memory file_cache_size, beyond which it will not cache buffers but allocate on request.

Default Value: true

disk_optimization_strategy

This option is commented out by default.

The strategy for optimizing disk read Possible values are: ssd (for solid state disks, the default) spinning (for spinning disks)

Default Value: ssd

memtable

Supported memtable implementations and selected default. Currently Cassandra offers two memtable implementations: - SkipListMemtable is the legacy memtable implementation provided by earlier versions of Cassandra. - TrieMemtable is a new memtable that utilizes a trie data structure. This implementation significantly reduces garbage collection load by moving more of the sstable metadata off-heap, fits more data in the same allocation and can reliably handle higher write throughput. Because the trie memtable is a sharded single-writer solution, it can perform worse when the load is very unevenly distributed, e.g. when most of the writes access a very small number of partitions or with legacy secondary indexes. The memtable implementation can be selected per table by setting memtable property in the table definition to one of the configurations specified below. If the memtable property is not set, the "default" configuration will be used. See src/java/org/apache/cassandra/db/memtable/Memtable_API.md for further information.

memtable_heap_space

This option is commented out by default.

Total permitted memory to use for memtables. Cassandra will stop accepting writes when the limit is exceeded until a flush completes, and will trigger a flush based on memtable_cleanup_threshold If omitted, Cassandra will set both to 1/4 the size of the heap. Min unit: MiB

Default Value: 2048MiB

memtable_offheap_space

This option is commented out by default. Min unit: MiB

Default Value: 2048MiB

memtable_cleanup_threshold

This option is commented out by default.

memtable_cleanup_threshold is deprecated. The default calculation is the only reasonable choice. See the comments on memtable_flush_writers for more information.

Ratio of occupied non-flushing memtable size to total permitted size that will trigger a flush of the largest memtable. Larger mct will mean larger flushes and hence less compaction, but also less concurrent flush activity which can make it difficult to keep your disks fed under heavy write load.

memtable_cleanup_threshold defaults to 1 / (memtable_flush_writers + 1)

Default Value: 0.11

memtable_allocation_type

Specify the way Cassandra allocates and manages memtable memory. Options are:

heap_buffers on heap nio buffers

offheap_buffers off heap (direct) nio buffers

offheap_objects off heap objects

Default Value: offheap_objects

repair_session_space

This option is commented out by default.

Limit memory usage for Merkle tree calculations during repairs of a certain table and common token range. Repair commands targetting multiple tables or virtual nodes can exceed this limit depending on concurrent_merkle_tree_requests.

The default is 1/16th of the available heap. The main tradeoff is that smaller trees have less resolution, which can lead to over-streaming data. If you see heap pressure during repairs, consider lowering this, but you cannot go below one mebibyte. If you see lots of over-streaming, consider raising this or using subrange repair.

For more details see https://issues.apache.org/jira/browse/CASSANDRA-14096.

Min unit: MiB

concurrent_merkle_tree_requests

This option is commented out by default.

The number of simultaneous Merkle tree requests during repairs that can be performed by a repair command. The size of each validation request is limited by the repair_session_space property, so setting this to 1 will make sure that a repair command doesn't exceed that limit, even if the repair command is repairing multiple tables or multiple virtual nodes.

There isn't a limit by default for backwards compatibility, but this can produce OOM for commands

repairing multiple tables or multiple virtual nodes. A limit of just 1 simultaneous Merkle tree request is generally recommended with no virtual nodes so repair_session_space, and thereof the Merkle tree resolution, can be high. For virtual nodes a value of 1 with the default repair_session_space value will produce higher resolution Merkle trees at the expense of speed. Alternatively, when working with virtual nodes it can make sense to reduce the repair_session_space and increase the value of concurrent_merkle_tree_requests because each range will contain fewer data.

For more details see https://issues.apache.org/jira/browse/CASSANDRA-19336.

A zero value means no limit.

Default Value: 0

repair

This option is commented out by default.

commitlog_total_space

This option is commented out by default. # Configure the retries for each of the repair messages that support it. As of this moment retries use an exponential algorithm where each attempt sleeps longer based off the base_sleep_time and attempt. retries: max_attempts: 10 base_sleep_time: 200ms max_sleep_time: 1s # Increase the timeout of validation responses due to them containing the merkle tree merkle_tree_response: base_sleep_time: 30s max_sleep_time: 1m

Total space to use for commit logs on disk.

If space gets above this value, Cassandra will flush every dirty CF in the oldest segment and remove it. So a small total commitlog space will tend to cause more flush activity on less-active columnfamilies.

The default value is the smaller of 8192, and 1/4 of the total space of the commitlog volume.

Default Value: 8192MiB

memtable_flush_writers

This option is commented out by default.

This sets the number of memtable flush writer threads per disk as well as the total number of memtables that can be flushed concurrently. These are generally a combination of compute and IO bound.

Memtable flushing is more CPU efficient than memtable ingest and a single thread can keep up with the ingest rate of a whole server on a single fast disk until it temporarily becomes IO bound under contention typically with compaction. At that point you need multiple flush threads. At some point in the future it may become CPU bound all the time.

You can tell if flushing is falling behind using the MemtablePool.BlockedOnAllocation metric which should be 0, but will be non-zero if threads are blocked waiting on flushing to free memory.

memtable_flush_writers defaults to two for a single data directory. This means that two memtables can be flushed concurrently to the single data directory. If you have multiple data directories the default is one memtable flushing at a time but the flush will use a thread per data directory so you will get two or more writers.

Two is generally enough to flush on a fast disk [array] mounted as a single data directory. Adding more flush writers will result in smaller more frequent flushes that introduce more compaction overhead.

There is a direct tradeoff between number of memtables that can be flushed concurrently and flush size and frequency. More is not better you just need enough flush writers to never stall waiting for flushing to free memory.

Default Value: 2

cdc_total_space

This option is commented out by default.

Total space to use for change-data-capture logs on disk.

If space gets above this value, Cassandra will throw WriteTimeoutException on Mutations including tables with CDC enabled. A CDCCompactor is responsible for parsing the raw CDC logs and deleting them when parsing is completed.

The default value is the min of 4096 MiB and 1/8th of the total space of the drive where cdc_raw_directory resides. Min unit: MiB

Default Value: 4096MiB

cdc_free_space_check_interval

This option is commented out by default.

When we hit our cdc_raw limit and the CDCCompactor is either running behind or experiencing backpressure, we check at the following interval to see if any new space for cdc-tracked tables has been made available. Default to 250ms Min unit: ms

Default Value: 250ms

trickle_fsync

Whether to, when doing sequential writing, fsync() at intervals in order to force the operating system to flush the dirty buffers. Enable this to avoid sudden dirty buffer flushing from impacting read

latencies. Almost always a good idea on SSDs; not necessarily on platters.

Default Value: true

trickle_fsync_interval

Min unit: KiB

Default Value: 10240KiB

storage_port

TCP port, for commands and data For security reasons, you should not expose this port to the internet. Firewall it if needed.

Default Value: 7000

ssl_storage_port

SSL port, for legacy encrypted communication. This property is unused unless enabled in server_encryption_options (see below). As of cassandra 4.0, this property is deprecated as a single port can be used for either/both secure and insecure connections. For security reasons, you should not expose this port to the internet. Firewall it if needed.

Default Value: 7001

listen_address

Address or interface to bind to and tell other Cassandra nodes to connect to. You *must* change this if you want multiple nodes to be able to communicate!

Set listen_address OR listen_interface, not both.

Leaving it blank leaves it up to InetAddress.getLocalHost(). This will always do the Right Thing *if* the node is properly configured (hostname, name resolution, etc), and the Right Thing is to use the address associated with the hostname (it might not be). If unresolvable it will fall back to InetAddress.getLoopbackAddress(), which is wrong for production systems.

Setting listen_address to 0.0.0.0 is always wrong.

Default Value: localhost

listen_interface

This option is commented out by default.

Set listen_address OR listen_interface, not both. Interfaces must correspond to a single address, IP aliasing is not supported.

Default Value: eth0

listen_interface_prefer_ipv6

This option is commented out by default.

If you choose to specify the interface by name and the interface has an ipv4 and an ipv6 address you can specify which should be chosen using listen_interface_prefer_ipv6. If false the first ipv4 address will be used. If true the first ipv6 address will be used. Defaults to false preferring ipv4. If there is only one address it will be selected regardless of ipv4/ipv6.

Default Value: false

broadcast_address

This option is commented out by default.

Address to broadcast to other Cassandra nodes Leaving this blank will set it to the same value as listen_address

Default Value: 1.2.3.4

listen_on_broadcast_address

This option is commented out by default.

When using multiple physical network interfaces, set this to true to listen on broadcast_address in addition to the listen_address, allowing nodes to communicate in both interfaces. Ignore this property if the network configuration automatically routes between the public and private networks such as EC2.

Default Value: false

internode_authenticator

This option is commented out by default.

Internode authentication backend, implementing IInternodeAuthenticator; used to allow/disallow connections from peer nodes.

start_native_transport

```
 class_name : org.apache.cassandra.auth.AllowAllInternodeAuthenticator
 parameters :
MutualTlsInternodeAuthenticator can be configured using the following configuration.One
can add their own validator
which implements MutualTlsCertificateValidator class and provide logic for extracting
identity out of certificates
and validating certificates.
 class_name : org.apache.cassandra.auth.MutualTlsInternodeAuthenticator
 parameters :
   validator_class_name: org.apache.cassandra.auth.SpiffeCertificateValidator
   trusted_peer_identities: "spiffe1,spiffe2"
   node_identity: "spiffe1"
Whether to start the native transport server.
The address on which the native transport is bound is defined by rpc_address.
```

Default Value: true

native_transport_port

port for the CQL native transport to listen for clients on For security reasons, you should not expose this port to the internet. Firewall it if needed.

Default Value: 9042

native_transport_port_ssl

This option is commented out by default. Enabling native transport encryption in client_encryption_options allows you to either use encryption for the standard port or to use a dedicated, additional port along with the unencrypted standard native_transport_port. Enabling client encryption and keeping native_transport_port_ssl disabled will use encryption for native_transport_port. Setting native_transport_port_ssl to a different value from native_transport_port will use encryption for native_transport_port_ssl while keeping native_transport_port unencrypted. This feature is deprecated since Cassandra 5.0 and will be removed. Please consult deprecation section in NEWS.txt.

Default Value: 9142

native_transport_max_threads

This option is commented out by default. The maximum threads for handling requests (note that idle threads are stopped after 30 seconds so there is not corresponding minimum setting).

Default Value: 128

native_transport_max_auth_threads

This option is commented out by default. The maximum threads for handling auth requests in a separate executor from main request executor. When set to 0, main executor for requests is used.

Default Value: 4

native_transport_max_frame_size

This option is commented out by default.

The maximum size of allowed frame. Frame (requests) larger than this will be rejected as invalid. The default is 16MiB. If you're changing this parameter, you may want to adjust max_value_size accordingly. This should be positive and less than 2048. Min unit: MiB

Default Value: 16MiB

native_transport_max_concurrent_connections

This option is commented out by default.

The maximum number of concurrent client connections. The default is -1, which means unlimited.

Default Value: -1

native_transport_max_concurrent_connections_per_ip

This option is commented out by default.

The maximum number of concurrent client connections per source ip. The default is -1, which means unlimited.

Default Value: -1

native_transport_allow_older_protocols

Controls whether Cassandra honors older, yet currently supported, protocol versions. The default is true, which means all supported protocols will be honored.

Default Value: true

native_transport_idle_timeout

This option is commented out by default.

Controls when idle client connections are closed. Idle connections are ones that had neither reads nor writes for a time period.

Clients may implement heartbeats by sending OPTIONS native protocol message after a timeout, which will reset idle timeout timer on the server side. To close idle client connections, corresponding values for heartbeat intervals have to be set on the client side.

Idle connection timeouts are disabled by default. Min unit: ms

Default Value: 60000ms

native_transport_rate_limiting_enabled

This option is commented out by default.

When enabled, limits the number of native transport requests dispatched for processing per second. Behavior once the limit has been breached depends on the value of THROW_ON_OVERLOAD specified in the STARTUP message sent by the client during connection establishment. (See section "4.1.1. STARTUP" in "CQL BINARY PROTOCOL v5".) With the THROW_ON_OVERLOAD flag enabled, messages that breach the limit are dropped, and an OverloadedException is thrown for the client to handle. When the flag is not enabled, the server will stop consuming messages from the channel/socket, putting backpressure on the client while already dispatched messages are processed.

Default Value: false

native_transport_max_requests_per_second

This option is commented out by default.

Default Value: 1000000

rpc_address

The address or interface to bind the native transport server to.

Set rpc_address OR rpc_interface, not both.

Leaving rpc_address blank has the same effect as on listen_address (i.e. it will be based on the configured hostname of the node).

Note that unlike listen_address, you can specify 0.0.0.0, but you must also set broadcast_rpc_address to a value other than 0.0.0.0.

For security reasons, you should not expose this port to the internet. Firewall it if needed.

Default Value: localhost

rpc_interface

This option is commented out by default.

Set rpc_address OR rpc_interface, not both. Interfaces must correspond to a single address, IP aliasing is not supported.

Default Value: eth1

rpc_interface_prefer_ipv6

This option is commented out by default.

If you choose to specify the interface by name and the interface has an ipv4 and an ipv6 address you can specify which should be chosen using rpc_interface_prefer_ipv6. If false the first ipv4 address will be used. If true the first ipv6 address will be used. Defaults to false preferring ipv4. If there is only one address it will be selected regardless of ipv4/ipv6.

Default Value: false

broadcast_rpc_address

This option is commented out by default.

RPC address to broadcast to drivers and other Cassandra nodes. This cannot be set to 0.0.0.0. If left blank, this will be set to the value of rpc_address. If rpc_address is set to 0.0.0.0, broadcast_rpc_address must be set.

Default Value: 1.2.3.4

rpc_keepalive

enable or disable keepalive on rpc/native connections

Default Value: true

internode_socket_send_buffer_size

This option is commented out by default.

Uncomment to set socket buffer size for internode communication Note that when setting this, the buffer size is limited by net.core.wmem_max and when not setting it it is defined by

net.ipv4.tcp_wmem See also: /proc/sys/net/core/wmem_max /proc/sys/net/core/rmem_max /proc/sys/net/ipv4/tcp_wmem /proc/sys/net/ipv4/tcp_wmem and 'man tcp' Min unit: B

internode_socket_receive_buffer_size

This option is commented out by default.

Uncomment to set socket buffer size for internode communication Note that when setting this, the buffer size is limited by net.core.wmem_max and when not setting it it is defined by net.ipv4.tcp_wmem Min unit: B

incremental_backups

Set to true to have Cassandra create a hard link to each sstable flushed or streamed locally in a backups/ subdirectory of all the keyspace data in this node. Removing these links is the operator's responsibility. The operator can also turn off incremental backups for specified table by setting table parameter incremental_backups to false, which is set to true by default. See CASSANDRA-15402

Default Value: false

snapshot_before_compaction

Whether or not to take a snapshot before each compaction. Be careful using this option, since Cassandra won't clean up the snapshots for you. Mostly useful if you're paranoid when there is a data format change.

Default Value: false

auto_snapshot

Whether or not a snapshot is taken of the data before keyspace truncation or dropping of column families. The STRONGLY advised default of true should be used to provide data safety. If you set this flag to false, you will lose data on truncation or drop.

Default Value: true

auto_snapshot_ttl

This option is commented out by default.

Adds a time-to-live (TTL) to auto snapshots generated by table truncation or drop (when enabled). After the TTL is elapsed, the snapshot is automatically cleared. By default, auto snapshots **do not** have TTL, uncomment the property below to enable TTL on auto snapshots. Accepted units: d (days), h (hours) or m (minutes)

Default Value: 30d

snapshot_links_per_second

The act of creating or clearing a snapshot involves creating or removing potentially tens of thousands of links, which can cause significant performance impact, especially on consumer grade SSDs. A non-zero value here can be used to throttle these links to avoid negative performance impact of taking and clearing snapshots

Default Value: 0

sstable

The sstable formats configuration. SSTable formats implementations are loaded using the service loader mechanism. In this section, one can select the format for created sstables and pass additional parameters for the formats available on the classpath. The default format is "big", the legacy SSTable format in use since Cassandra 3.0. Cassandra versions 5.0 and later also support the trie-indexed "bti" format, which offers better performance.

column_index_size

Granularity of the collation index of rows within a partition. Applies to both BIG and BTI SSTable formats. In both formats, a smaller granularity results in faster lookup of rows within a partition, but a bigger index file size. Using smaller granularities with the BIG format is not recommended because bigger collation indexes cannot be cached efficiently or at all if they become sufficiently large. Further, if large rows, or a very large number of rows per partition are present, it is recommended to increase the index granularity or switch to the BTI SSTable format.

Leave undefined to use a default suitable for the SSTable format in use (64 KiB for BIG, 16KiB for BTI). Min unit: KiB

Default Value: 4KiB

default_compaction

Default compaction strategy, applied when a table's parameters do not specify compaction. The selected compaction strategy will also apply to system tables.

If no value is specified, the default is to use SizeTieredCompactionStrategy, with its default compaction parameters.

concurrent_compactors

Number of simultaneous compactions to allow, NOT including validation "compactions" for anti-entropy repair. Simultaneous compactions can help preserve read performance in a mixed read/write

workload, by mitigating the tendency of small sstables to accumulate during a single long running compactions. The default is usually fine and if you experience problems with compaction running too slowly or too fast, you should look at compaction_throughput first.

concurrent_compactors defaults to the smaller of (number of disks, number of cores), with a minimum of 2 and a maximum of 8.

If your data directories are backed by SSD, you should increase this to the number of cores.

Default Value: 8

concurrent_validations

This option is commented out by default.

Number of simultaneous repair validations to allow. If not set or set to a value less than 1, it defaults to the value of concurrent_compactors. To set a value greeater than concurrent_compactors at startup, the system property cassandra.allow_unlimited_concurrent_validations must be set to true. To dynamically resize to a value > concurrent_compactors on a running node, first call the bypassConcurrentValidatorsLimit method on the org.apache.cassandra.db:type=StorageService mbean

Default Value: 0

concurrent_materialized_view_builders

Number of simultaneous materialized view builder tasks to allow.

Default Value: 1

compaction_throughput

Throttles compaction to the given total throughput across the entire system. The faster you insert data, the faster you need to compact in order to keep the sstable count down, but in general, setting this to 16 to 32 times the rate you are inserting data is more than sufficient. Setting this to 0 disables throttling. Note that this accounts for all types of compaction, including validation compaction (building Merkle trees for repairs).

Default Value: 64MiB/s

sstable_preemptive_open_interval

When compacting, the replacement sstable(s) can be opened before they are completely written, and used in place of the prior sstables for any range that has been written. This helps to smoothly transfer reads between the sstables, reducing page cache churn and keeping hot rows hot Set sstable_preemptive_open_interval to null for disabled which is equivalent to sstable_preemptive_open_interval_in_mb being negative Min unit: MiB

Default Value: 50MiB

uuid_sstable_identifiers_enabled

Starting from 4.1 sstables support UUID based generation identifiers. They are disabled by default because once enabled, there is no easy way to downgrade. When the node is restarted with this option set to true, each newly created sstable will have a UUID based generation identifier and such files are not readable by previous Cassandra versions. At some point, this option will become true by default and eventually get removed from the configuration.

Default Value: true

stream_entire_sstables

When enabled, permits Cassandra to zero-copy stream entire eligible SSTables between nodes, including every component. This speeds up the network transfer significantly subject to throttling specified by entire_sstable_stream_throughput_outbound, and entire_sstable_inter_dc_stream_throughput_outbound for inter-DC transfers. Enabling this will reduce the GC pressure on sending and receiving node. When unset, the default is enabled. While this feature tries to keep the disks balanced, it cannot guarantee it. This feature will be automatically disabled if internode encryption is enabled.

Default Value: true

entire_sstable_stream_throughput_outbound

This option is commented out by default.

Throttles entire SSTable outbound streaming file transfers on this node to the given total throughput in Mbps. Setting this value to 0 it disables throttling. When unset, the default is 200 Mbps or 24 MiB/s.

Default Value: 24MiB/s

entire_sstable_inter_dc_stream_throughput_outbound

This option is commented out by default.

Throttles entire SSTable file streaming between datacenters. Setting this value to 0 disables throttling for entire SSTable inter-DC file streaming. When unset, the default is 200 Mbps or 24 MiB/s.

Default Value: 24MiB/s

stream_throughput_outbound

This option is commented out by default.

Throttles all outbound streaming file transfers on this node to the given total throughput in Mbps. This is necessary because Cassandra does mostly sequential IO when streaming data during bootstrap or repair, which can lead to saturating the network connection and degrading rpc performance. When unset, the default is 200 Mbps or 24 MiB/s.

Default Value: 24MiB/s

inter_dc_stream_throughput_outbound

This option is commented out by default.

Throttles all streaming file transfer between the datacenters, this setting allows users to throttle inter dc stream throughput in addition to throttling all network stream traffic as configured with stream_throughput_outbound_megabits_per_sec When unset, the default is 200 Mbps or 24 MiB/s.

Default Value: 24MiB/s

read_request_timeout

Server side timeouts for requests. The server will return a timeout exception to the client if it can't complete an operation within the corresponding timeout. Those settings are a protection against: 1) having client wait on an operation that might never terminate due to some failures. 2) operations that use too much CPU/read too much data (leading to memory build up) by putting a limit to how long an operation will execute. For this reason, you should avoid putting these settings too high. In other words, if you are timing out requests because of underlying resource constraints then increasing the timeout will just cause more problems. Of course putting them too low is equally ill-advised since clients could get timeouts even for successful operations just because the timeout setting is too tight.

How long the coordinator should wait for read operations to complete. Lowest acceptable value is 10 ms. Min unit: ms

Default Value: 5000ms

range_request_timeout

How long the coordinator should wait for seq or index scans to complete. Lowest acceptable value is 10 ms. Min unit: ms

Default Value: 10000ms

write_request_timeout

How long the coordinator should wait for writes to complete. Lowest acceptable value is 10 ms. Min unit: ms

Default Value: 2000ms

counter_write_request_timeout

How long the coordinator should wait for counter writes to complete. Lowest acceptable value is 10 ms. Min unit: ms

Default Value: 5000ms

cas_contention_timeout

How long a coordinator should continue to retry a CAS operation that contends with other proposals for the same row. Lowest acceptable value is 10 ms. Min unit: ms

Default Value: 1000ms

truncate_request_timeout

How long the coordinator should wait for truncates to complete (This can be much longer, because unless auto_snapshot is disabled we need to flush first so we can snapshot before removing the data.) Lowest acceptable value is 10 ms. Min unit: ms

Default Value: 60000ms

request_timeout

The default timeout for other, miscellaneous operations. Lowest acceptable value is 10 ms. Min unit: ms

Default Value: 10000ms

internode_tcp_connect_timeout

This option is commented out by default.

Defensive settings for protecting Cassandra from true network partitions. See (CASSANDRA-14358) for details.

The amount of time to wait for internode tcp connections to establish. Min unit: ms

Default Value: 2000ms

internode_tcp_user_timeout

This option is commented out by default.

The amount of time unacknowledged data is allowed on a connection before we throw out the connection Note this is only supported on Linux + epoll, and it appears to behave oddly above a setting of 30000 (it takes much longer than 30s) as of Linux 4.12. If you want something that high set this to 0 which picks up the OS default and configure the net.ipv4.tcp_retries2 sysctl to be ~8. Min unit: ms

Default Value: 30000ms

internode_streaming_tcp_user_timeout

This option is commented out by default.

The amount of time unacknowledged data is allowed on a streaming connection. The default is 5 minutes. Increase it or set it to 0 in order to increase the timeout. Min unit: ms

Default Value: 300000ms

internode_application_send_queue_capacity

This option is commented out by default.

Global, per-endpoint and per-connection limits imposed on messages queued for delivery to other nodes and waiting to be processed on arrival from other nodes in the cluster. These limits are applied to the on-wire size of the message being sent or received.

The basic per-link limit is consumed in isolation before any endpoint or global limit is imposed. Each node-pair has three links: urgent, small and large. So any given node may have a maximum of N*3*(internode_application_send_queue_capacity+internode_application_receive_queue_capacity) messages queued without any coordination between them although in practice, with token-aware routing, only RF*tokens nodes should need to communicate with significant bandwidth.

The per-endpoint limit is imposed on all messages exceeding the per-link limit, simultaneously with the global limit, on all links to or from a single node in the cluster. The global limit is imposed on all messages exceeding the per-link limit, simultaneously with the per-endpoint limit, on all links to or from any node in the cluster.

Min unit: B

Default Value: 4MiB

internode_application_send_queue_reserve_endpoint_capacity

This option is commented out by default.

Default Value: 128MiB

internode_application_send_queue_reserve_global_capacity

This option is commented out by default.

Default Value: 512MiB

internode_application_receive_queue_capacity

This option is commented out by default.

Default Value: 4MiB

internode_application_receive_queue_reserve_endpoint_capacity

This option is commented out by default.

Default Value: 128MiB

internode_application_receive_queue_reserve_global_capacity

This option is commented out by default.

Default Value: 512MiB

slow_query_log_timeout

How long before a node logs slow queries. Select queries that take longer than this timeout to execute, will generate an aggregated log message, so that slow queries can be identified. Set this value to zero to disable slow query logging. Min unit: ms

Default Value: 500ms

internode_timeout

This option is commented out by default.

Enable operation timeout information exchange between nodes to accurately measure request timeouts. If disabled, replicas will assume that requests were forwarded to them instantly by the coordinator, which means that under overload conditions we will waste that much extra time processing already-timed-out requests.

Warning: It is generally assumed that users have setup NTP on their clusters, and that clocks are modestly in sync, since this is a requirement for general correctness of last write wins.

Default Value: true

streaming_keep_alive_period

This option is commented out by default.

Set period for idle state control messages for earlier detection of failed streams This node will send a keep-alive message periodically on the streaming's control channel. This ensures that any eventual SocketTimeoutException will occur within 2 keep-alive cycles If the node cannot send, or timeouts sending, the keep-alive message on the netty control channel the stream session is closed. Default value is 300s (5 minutes), which means stalled streams are detected within 10 minutes Specify 0 to disable. Min unit: s

Default Value: 300s

streaming_connections_per_host

This option is commented out by default.

Limit number of connections per host for streaming Increase this when you notice that joins are CPU-bound rather that network bound (for example a few nodes with big files).

Default Value: 1

streaming_state_expires

This option is commented out by default.

Settings for stream stats tracking; used by system_views.streaming table How long before a stream is evicted from tracking; this impacts both historic and currently running streams.

Default Value: 3d

streaming_state_size

This option is commented out by default. How much memory may be used for tracking before evicting session from tracking; once crossed historic and currently running streams maybe impacted.

Default Value: 40MiB

streaming_stats_enabled

This option is commented out by default. Enable/Disable tracking of streaming stats

Default Value: true

partition_denylist_enabled

This option is commented out by default.

Allows denying configurable access (rw/rr) to operations on configured ks, table, and partitions, intended for use by operators to manage cluster health vs application access. See CASSANDRA-12106 and CEP-13 for more details.

Default Value: false

denylist_writes_enabled

This option is commented out by default.

Default Value: true

denylist_reads_enabled

This option is commented out by default.

Default Value: true

denylist_range_reads_enabled

This option is commented out by default.

Default Value: true

denylist_refresh

This option is commented out by default.

The interval at which keys in the cache for denylisting will "expire" and async refresh from the backing DB. Note: this serves only as a fail-safe, as the usage pattern is expected to be "mutate state, refresh cache" on any changes to the underlying denylist entries. See documentation for details. Min unit: s

Default Value: 600s

denylist_initial_load_retry

This option is commented out by default.

In the event of errors on attempting to load the denylist cache, retry on this interval. Min unit: s

Default Value: 5s

denylist_max_keys_per_table

This option is commented out by default.

We cap the number of denylisted keys allowed per table to keep things from growing unbounded. Nodes will warn above this limit while allowing new denylisted keys to be inserted. Denied keys are loaded in natural query / clustering ordering by partition key in case of overflow.

Default Value: 1000

denylist_max_keys_total

This option is commented out by default.

We cap the total number of denylisted keys allowed in the cluster to keep things from growing unbounded. Nodes will warn on initial cache load that there are too many keys and be direct the operator to trim down excess entries to within the configured limits.

Default Value: 10000

denylist_consistency_level

This option is commented out by default.

Since the denylist in many ways serves to protect the health of the cluster from partitions operators have identified as being in a bad state, we usually want more robustness than just CL.ONE on operations to/from these tables to ensure that these safeguards are in place. That said, we allow users to configure this if they're so inclined.

Default Value: QUORUM

phi_convict_threshold

This option is commented out by default.

phi value that must be reached for a host to be marked down. most users should never need to adjust this.

Default Value: 8

endpoint_snitch

endpoint_snitch — Set this to a class that implements IEndpointSnitch. The snitch has two functions:

- it teaches Cassandra enough about your network topology to route requests efficiently
- it allows Cassandra to spread replicas around your cluster to avoid correlated failures. It does this by grouping machines into "datacenters" and "racks." Cassandra will do its best not to have more than one replica on the same "rack" (which may not actually be a physical location)

CASSANDRA WILL NOT ALLOW YOU TO SWITCH TO AN INCOMPATIBLE SNITCH ONCE DATA IS INSERTED INTO THE CLUSTER. This would cause data loss. This means that if you start with the default SimpleSnitch, which locates every node on "rack1" in "datacenter1", your only options if you need to add another datacenter are GossipingPropertyFileSnitch (and the older PFS). From there, if you want to migrate to an incompatible snitch like Ec2Snitch you can do it by adding new nodes under Ec2Snitch (which will locate them in a new "datacenter") and decommissioning the old ones.

Out of the box, Cassandra provides:

SimpleSnitch: Treats Strategy order as proximity. This can improve cache locality when disabling read repair. Only appropriate for single-datacenter deployments.

GossipingPropertyFileSnitch This should be your go-to snitch for production use. The rack and datacenter for the local node are defined in cassandra-rackdc.properties and propagated to other nodes via gossip. If cassandra-topology.properties exists, it is used as a fallback, allowing migration from the PropertyFileSnitch.

PropertyFileSnitch: Proximity is determined by rack and data center, which are explicitly configured in cassandra-topology.properties.

AlibabaCloudSnitch: Snitch for getting dc and rack of a node from metadata service of Alibaba cloud. This snitch that assumes an ECS region is a DC and an ECS availability_zone is a rack.

AzureSnitch: Gets datacenter from 'location' and rack from 'zone' fields of 'compute' object from instance metadata service. If the availability zone is not enabled, it will use the fault domain and get its respective value.

CloudstackSnitch: A snitch that assumes a Cloudstack Zone follows the typical convention country-

location-az and uses a country/location tuple as a datacenter and the availability zone as a rack. WARNING: This snitch is deprecated and it is scheduled to be removed in the next major version of Cassandra.

Ec2Snitch: Appropriate for EC2 deployments in a single Region. Loads Region and Availability Zone information from the EC2 API. The Region is treated as the datacenter, and the Availability Zone as the rack. Only private IPs are used, so this will not work across multiple Regions.

Ec2MultiRegionSnitch: Uses public IPs as broadcast_address to allow cross-region connectivity. (Thus, you should set seed addresses to the public IP as well.) You will need to open the storage_port or ssl_storage_port on the public IP firewall. (For intra-Region traffic, Cassandra will switch to the private IP after establishing a connection.)

GoogleCloudSnitch: Snitch for getting dc and rack of a node from metadata service of Google cloud. This snitch that assumes an GCE region is a DC and an GCE availability_zone is a rack.

RackInferringSnitch: Proximity is determined by rack and data center, which are assumed to correspond to the 3rd and 2nd octet of each node's IP address, respectively. Unless this happens to match your deployment conventions, this is best used as an example of writing a custom Snitch class and is provided in that spirit.

You can use a custom Snitch by setting this to the full class name of the snitch, which will be assumed to be on your classpath.

Default Value: SimpleSnitch

dynamic_snitch_update_interval

controls how often to perform the more expensive part of host score calculation Min unit: ms

Default Value: 100ms

dynamic_snitch_reset_interval

controls how often to reset all host scores, allowing a bad host to possibly recover Min unit: ms

Default Value: 600000ms

dynamic_snitch_badness_threshold

if set greater than zero, this will allow 'pinning' of replicas to hosts in order to increase cache capacity. The badness threshold will control how much worse the pinned host has to be before the dynamic snitch will prefer other replicas over it. This is expressed as a double which represents a percentage. Thus, a value of 0.2 means Cassandra would continue to prefer the static snitch values until the pinned host was 20% worse than the fastest.

Default Value: 1.0

crypto_provider

Configures Java crypto provider. By default, it will use DefaultCryptoProvider which will install Amazon Correto Crypto Provider.

Amazon Correto Crypto Provider works currently for x86_64 and aarch_64 platforms. If this provider fails it will fall back to the default crypto provider in the JRE.

To force failure when the provider was not installed properly, set the property "fail_on_missing_provider" to "true".

To bypass the installation of a crypto provider use class 'org.apache.cassandra.security.JREProvider'

server_encryption_options

Configure server-to-server internode encryption

JVM and netty defaults for supported SSL socket protocols and cipher suites can be replaced using custom encryption options. This is not recommended unless you have policies in place that dictate certain settings, or need to disable vulnerable ciphers or protocols in case the JVM cannot be updated.

FIPS compliant settings can be configured at JVM level and should not involve changing encryption settings here: https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/FIPS.html

NOTE this default configuration is an insecure configuration. If you need to enable server-to-server encryption generate server keystores (and truststores for mutual authentication) per: http://download.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html#CreateKeystore Then perform the following configuration changes:

Step 1: Set internode_encryption=<dc|rack|all> and explicitly set optional=true. Restart all nodes

Step 2: Set optional=false (or remove it) and if you generated truststores and want to use mutual auth set require_client_auth=true. Restart all nodes

Default Value (complex option):

```
      # On outbound connections, determine which type of peers to securely connect to.
      #   The available options are :
      #     none : Do not encrypt outgoing connections
      #     dc   : Encrypt connections to peers in other datacenters but not within
datacenters
      #     rack : Encrypt connections to peers in other racks but not within racks
      #     all  : Always use encrypted connections
      internode_encryption: none
```

```
      # When set to true, encrypted and unencrypted connections are allowed on the
storage_port
      # This should _only be true_ while in unencrypted or transitional operation
      # optional defaults to true if internode_encryption is none
      # optional: true
      # If enabled, will open up an encrypted listening socket on ssl_storage_port.
Should only be used
      # during upgrade to 4.0; otherwise, set to false.
      legacy_ssl_storage_port_enabled: false
      # Set to a valid keystore if internode_encryption is dc, rack or all
      keystore: conf/.keystore
      #keystore_password: cassandra
      # Configure the way Cassandra creates SSL contexts.
      # To use PEM-based key material, see
org.apache.cassandra.security.PEMBasedSslContextFactory
      # ssl_context_factory:
      #     # Must be an instance of org.apache.cassandra.security.ISslContextFactory
      #     class_name: org.apache.cassandra.security.DefaultSslContextFactory
      # During internode mTLS authentication, inbound connections (acting as servers) use
keystore, keystore_password
      # containing server certificate to create SSLContext and
      # outbound connections (acting as clients) use outbound_keystore &
outbound_keystore_password with client certificates
      # to create SSLContext. By default, outbound_keystore is the same as keystore
indicating mTLS is not enabled.
    #  outbound_keystore: conf/.keystore
    #  outbound_keystore_password: cassandra
      # Verify peer server certificates
      require_client_auth: false
      # Set to a valid trustore if require_client_auth is true
      truststore: conf/.truststore
      #truststore_password: cassandra
      # Verify that the host name in the certificate matches the connected host
      require_endpoint_verification: false
      # More advanced defaults:
      # protocol: TLS
      # store_type: JKS
      # cipher_suites: [
      #   TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
      #   TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
      #   TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256,
TLS_RSA_WITH_AES_128_CBC_SHA,
      #   TLS_RSA_WITH_AES_256_CBC_SHA
      # ]
```

client_encryption_options

Configure client-to-server encryption.

NOTE this default configuration is an insecure configuration. If you need to enable client-to-server encryption generate server keystores (and truststores for mutual authentication) per: http://download.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html#CreateKeystore Then perform the following configuration changes:

Step 1: Set enabled=true and explicitly set optional=true. Restart all nodes

Step 2: Set optional=false (or remove it) and if you generated truststores and want to use mutual auth set require_client_auth=true. Restart all nodes

Default Value (complex option):

```
      # Enable client-to-server encryption
      enabled: false
      # When set to true, encrypted and unencrypted connections are allowed on the
native_transport_port
      # This should _only be true_ while in unencrypted or transitional operation
      # optional defaults to true when enabled is false, and false when enabled is true.
      # optional: true
      # Set keystore and keystore_password to valid keystores if enabled is true
      keystore: conf/.keystore
      #keystore_password: cassandra
      # Configure the way Cassandra creates SSL contexts.
      # To use PEM-based key material, see
org.apache.cassandra.security.PEMBasedSslContextFactory
      # ssl_context_factory:
      #     # Must be an instance of org.apache.cassandra.security.ISslContextFactory
      #     class_name: org.apache.cassandra.security.DefaultSslContextFactory
      # Verify client certificates
      require_client_auth: false
      # require_endpoint_verification: false
      # Set trustore and truststore_password if require_client_auth is true
      # truststore: conf/.truststore
      # truststore_password: cassandra
      # More advanced defaults:
      # protocol: TLS
      # store_type: JKS
      # cipher_suites: [
      #   TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
      #   TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
      #   TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256,
TLS_RSA_WITH_AES_128_CBC_SHA,
```

```
      #   TLS_RSA_WITH_AES_256_CBC_SHA
      # ]
```

internode_compression

internode_compression controls whether traffic between nodes is compressed. Can be:

all all traffic is compressed

dc traffic between different datacenters is compressed

none nothing is compressed.

Default Value: dc

inter_dc_tcp_nodelay

Enable or disable tcp_nodelay for inter-dc communication. Disabling it will result in larger (but fewer) network packets being sent, reducing overhead from the TCP protocol itself, at the cost of increasing latency if you block for cross-datacenter responses.

Default Value: false

trace_type_query_ttl

TTL for different trace types used during logging of the repair process. Min unit: s

Default Value: 1d

trace_type_repair_ttl

Min unit: s

Default Value: 7d

user_defined_functions_enabled

If unset, all GC Pauses greater than gc_log_threshold will log at INFO level UDFs (user defined functions) are disabled by default. As of Cassandra 3.0 there is a sandbox in place that should prevent execution of evil code.

Default Value: false

transparent_data_encryption_options

Enables encrypting data at-rest (on disk). Different key providers can be plugged in, but the default reads from a JCE-style keystore. A single keystore can hold multiple keys, but the one referenced by the "key_alias" is the only key that will be used for encrypt opertaions; previously used keys can still (and should!) be in the keystore and will be used on decrypt operations (to handle the case of key rotation).

It is strongly recommended to download and install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files for your version of the JDK. (current link: http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html)

Currently, only the following file types are supported for transparent data encryption, although more are coming in future cassandra releases: commitlog, hints

Default Value (complex option):

```
    enabled: false
    chunk_length_kb: 64
    cipher: AES/CBC/PKCS5Padding
    key_alias: testing:1
    # CBC IV length for AES needs to be 16 bytes (which is also the default size)
    # iv_length: 16
    key_provider:
      - class_name: org.apache.cassandra.security.JKSKeyProvider
        parameters:
          - keystore: conf/.keystore
            keystore_password: cassandra
            store_type: JCEKS
            key_password: cassandra
```

sai_options

This option is commented out by default. Storage Attached Indexing options.

tombstone_warn_threshold

SAFETY THRESHOLDS #

When executing a scan, within or across a partition, we need to keep the tombstones seen in memory so we can return them to the coordinator, which will use them to make sure other replicas also know about the deleted rows. With workloads that generate a lot of tombstones, this can cause performance problems and even exaust the server heap. (http://www.datastax.com/dev/blog/cassandra-anti-patterns-queues-and-queue-like-datasets) Adjust the thresholds here if you understand the dangers and want to scan more tombstones anyway. These thresholds may also be adjusted at runtime using the StorageService mbean.

Default Value: 1000

tombstone_failure_threshold

Default Value: 100000

replica_filtering_protection

Filtering and secondary index queries at read consistency levels above ONE/LOCAL_ONE use a mechanism called replica filtering protection to ensure that results from stale replicas do not violate consistency. (See CASSANDRA-8272 and CASSANDRA-15907 for more details.) This mechanism materializes replica results by partition on-heap at the coordinator. The more possibly stale results returned by the replicas, the more rows materialized during the query.

batch_size_warn_threshold

Log WARN on any multiple-partition batch size exceeding this value. 5KiB per batch by default. Caution should be taken on increasing the size of this threshold as it can lead to node instability. Min unit: KiB

Default Value: 5KiB

batch_size_fail_threshold

Fail any multiple-partition batch exceeding this value. 50KiB (10x warn threshold) by default. Min unit: KiB

Default Value: 50KiB

unlogged_batch_across_partitions_warn_threshold

Log WARN on any batches not of type LOGGED than span across more partitions than this limit

Default Value: 10

gc_log_threshold

This option is commented out by default.

GC Pauses greater than 200 ms will be logged at INFO level This threshold can be adjusted to minimize logging if necessary Min unit: ms

Default Value: 200ms

gc_warn_threshold

This option is commented out by default.

GC Pauses greater than gc_warn_threshold will be logged at WARN level Adjust the threshold based on your application throughput requirement. Setting to 0 will deactivate the feature. Min unit: ms

Default Value: 1000ms

max_value_size

This option is commented out by default.

Maximum size of any value in SSTables. Safety measure to detect SSTable corruption early. Any value size larger than this threshold will result into marking an SSTable as corrupted. This should be positive and less than 2GiB. Min unit: MiB

Default Value: 256MiB

default_keyspace_rf

This option is commented out by default.

- Impact on keyspace creation ** If replication factor is not mentioned as part of keyspace creation, default_keyspace_rf would apply. Changing this configuration would only take effect for keyspaces created after the change, but does not impact existing keyspaces created prior to the change.
- Impact on keyspace alter ** When altering a keyspace from NetworkTopologyStrategy to SimpleStrategy, default_keyspace_rf is applied if rf is not explicitly mentioned.
- Impact on system keyspaces ** This would also apply for any system keyspaces that need replication factor. A further note about system keyspaces - system_traces and system_distributed keyspaces take RF of 2 or default, whichever is higher, and system_auth keyspace takes RF of 1 or default, whichever is higher. Suggested value for use in production: 3

Default Value: 1

ideal_consistency_level

This option is commented out by default.

Track a metric per keyspace indicating whether replication achieved the ideal consistency level for writes without timing out. This is different from the consistency level requested by each write which may be lower in order to facilitate availability.

Default Value: EACH_QUORUM

automatic_sstable_upgrade

This option is commented out by default.

Automatically upgrade sstables after upgrade - if there is no ordinary compaction to do, the oldest non-upgraded sstable will get upgraded to the latest version

Default Value: false

max_concurrent_automatic_sstable_upgrades

This option is commented out by default. Limit the number of concurrent sstable upgrades

Default Value: 1

audit_logging_options

Audit logging - Logs every incoming CQL command request, authentication to a node. See the docs on audit_logging for full details about the various configuration options and production tips.

full_query_logging_options

This option is commented out by default.

default options for full query logging - these can be overridden from command line when executing nodetool enablefullquerylog

corrupted_tombstone_strategy

This option is commented out by default.

validate tombstones on reads and compaction can be either "disabled", "warn" or "exception"

Default Value: disabled

diagnostic_events_enabled

Diagnostic Events # If enabled, diagnostic events can be helpful for troubleshooting operational issues. Emitted events contain details on internal state and temporal relationships across events, accessible by clients via JMX.

Default Value: false

native_transport_flush_in_batches_legacy

This option is commented out by default.

Use native transport TCP message coalescing. If on upgrade to 4.0 you found your throughput decreasing, and in particular you run an old kernel or have very fewer client connections, this option might be worth evaluating.

Default Value: false

repaired_data_tracking_for_range_reads_enabled

Enable tracking of repaired state of data during reads and comparison between replicas Mismatches between the repaired sets of replicas can be characterized as either confirmed or unconfirmed. In this context, unconfirmed indicates that the presence of pending repair sessions, unrepaired partition tombstones, or some other condition means that the disparity cannot be considered conclusive. Confirmed mismatches should be a trigger for investigation as they may be indicative of corruption or data loss. There are separate flags for range vs partition reads as single partition reads are only tracked when CL > 1 and a digest mismatch occurs. Currently, range queries don't use digests so if enabled for range reads, all range reads will include repaired data tracking. As this adds some overhead, operators may wish to disable it whilst still enabling it for partition reads

Default Value: false

repaired_data_tracking_for_partition_reads_enabled

Default Value: false

report_unconfirmed_repaired_data_mismatches

If false, only confirmed mismatches will be reported. If true, a separate metric for unconfirmed mismatches will also be recorded. This is to avoid potential signal:noise issues are unconfirmed mismatches are less actionable than confirmed ones.

Default Value: false

auth_read_consistency_level

This option is commented out by default.

configure the read and write consistency levels for modifications to auth tables

Default Value: LOCAL_QUORUM

auth_write_consistency_level

This option is commented out by default.

Default Value: EACH_QUORUM

auth_cache_warming_enabled

This option is commented out by default.

Delays on auth resolution can lead to a thundering herd problem on reconnects; this option will enable warming of auth caches prior to node completing startup. See CASSANDRA-16958

Default Value: false

dynamic_data_masking_enabled

This option is commented out by default.

If enabled, dynamic data masking allows to attach CQL masking functions to the columns of a table. Users without the UNMASK permission will see an obscured version of the values of the columns with an attached mask. If dynamic data masking is disabled it won't be allowed to create new column masks, although it will still be possible to drop any previously existing masks. Also, any existing mask will be ignored at query time, so all users will see the clear values of the masked columns. Defaults to false to disable dynamic data masking.

Default Value: false

materialized_views_enabled

EXPERIMENTAL FEATURES #

Enables materialized view creation on this node. Materialized views are considered experimental and are not recommended for production use.

Default Value: false

sasi_indexes_enabled

Enables SASI index creation on this node. SASI indexes are considered experimental and are not recommended for production use.

Default Value: false

transient_replication_enabled

Enables creation of transiently replicated keyspaces on this node. Transient replication is experimental and is not recommended for production use.

Default Value: false

drop_compact_storage_enabled

Enables the used of 'ALTER ... DROP COMPACT STORAGE' statements on this node. 'ALTER ... DROP COMPACT STORAGE' is considered experimental and is not recommended for production use.

Default Value: false

use_statements_enabled

This option is commented out by default.

Whether or not USE <keyspace> is allowed. This is enabled by default to avoid failure on upgrade.

Default Value: true

client_error_reporting_exclusions

This option is commented out by default.

When the client triggers a protocol exception or unknown issue (Cassandra bug) we increment a client metric showing this; this logic will exclude specific subnets from updating these metrics

read_thresholds_enabled

This option is commented out by default. subnets: - 127.0.0.1 - 127.0.0.0/31

Enables read thresholds (warn/fail) across all replicas for reporting back to the client. See: CASSANDRA-16850

Default Value: false # scheduled to be set true in 4.2

coordinator_read_size_warn_threshold

This option is commented out by default. When read_thresholds_enabled: true, this tracks the materialized size of a query on the coordinator. If coordinator_read_size_warn_threshold is defined, this will emit a warning to clients with details on what query triggered this as well as the size of the result set; if coordinator_read_size_fail_threshold is defined, this will fail the query after it has exceeded this threshold, returning a read error to the user.

coordinator_read_size_fail_threshold

This option is commented out by default.

local_read_size_warn_threshold

This option is commented out by default. When read_thresholds_enabled: true, this tracks the size of the local read (as defined by heap size), and will warn/fail based off these thresholds; undefined disables these checks.

local_read_size_fail_threshold

This option is commented out by default.

row_index_read_size_warn_threshold

This option is commented out by default. When read_thresholds_enabled: true, this tracks the expected memory size of the RowIndexEntry and will warn/fail based off these thresholds; undefined disables these checks

row_index_read_size_fail_threshold

This option is commented out by default.

keyspaces_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when creating more user keyspaces than threshold. The two thresholds default to -1 to disable.

Default Value: -1

keyspaces_fail_threshold

This option is commented out by default.

Default Value: -1

tables_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when creating more user tables than threshold. The two thresholds default to

-1 to disable.

Default Value: -1

tables_fail_threshold

This option is commented out by default.

Default Value: -1

uncompressed_tables_enabled

This option is commented out by default.

Guardrail to enable or disable the ability to create uncompressed tables

Default Value: true

columns_per_table_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when creating/altering a table with more columns per table than threshold. The two thresholds default to -1 to disable.

Default Value: -1

columns_per_table_fail_threshold

This option is commented out by default.

Default Value: -1

secondary_indexes_per_table_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when creating more secondary indexes per table than threshold. The two thresholds default to -1 to disable.

Default Value: -1

secondary_indexes_per_table_fail_threshold

This option is commented out by default.

Default Value: -1

secondary_indexes_enabled

This option is commented out by default.

Guardrail to enable or disable the creation of secondary indexes

Default Value: true

materialized_views_per_table_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when creating more materialized views per table than threshold. The two thresholds default to -1 to disable.

Default Value: -1

materialized_views_per_table_fail_threshold

This option is commented out by default.

Default Value: -1

table_properties_warned

This option is commented out by default.

Guardrail to warn about, ignore or reject properties when creating tables. By default all properties are allowed.

Default Value: []

table_properties_ignored

This option is commented out by default.

Default Value: []

table_properties_disallowed

This option is commented out by default.

Default Value: []

user_timestamps_enabled

This option is commented out by default.

Guardrail to allow/disallow user-provided timestamps. Defaults to true.

Default Value: true

maximum_timestamp_warn_threshold

This option is commented out by default.

Guardrail to bound user-provided timestamps within a given range. Default is infinite (denoted by null). Accepted values are durations of the form 12h, 24h, etc.

maximum_timestamp_fail_threshold

This option is commented out by default.

minimum_timestamp_warn_threshold

This option is commented out by default.

minimum_timestamp_fail_threshold

This option is commented out by default.

group_by_enabled

This option is commented out by default.

Guardrail to allow/disallow GROUP BY functionality.

Default Value: true

drop_truncate_table_enabled

This option is commented out by default.

Guardrail to allow/disallow TRUNCATE and DROP TABLE statements

Default Value: true

drop_keyspace_enabled

This option is commented out by default.

Guardrail to allow/disallow DROP KEYSPACE statements

Default Value: true

page_size_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when using a page size greater than threshold. The two thresholds default to -1 to disable.

Default Value: -1

page_size_fail_threshold

This option is commented out by default.

Default Value: -1

read_before_write_list_operations_enabled

This option is commented out by default.

Guardrail to allow/disallow list operations that require read before write, i.e. setting list element by index and removing list elements by either index or value. Defaults to true.

Default Value: true

partition_keys_in_select_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when querying with an IN restriction selecting more partition keys than threshold. The two thresholds default to -1 to disable.

Default Value: -1

partition_keys_in_select_fail_threshold

This option is commented out by default.

Default Value: -1

in_select_cartesian_product_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when an IN query creates a cartesian product with a size exceeding threshold, eg. "a in (1,2,...10) and b in (1,2...10)" results in cartesian product of 100. The two thresholds default to -1 to disable.

Default Value: -1

in_select_cartesian_product_fail_threshold

This option is commented out by default.

Default Value: -1

read_consistency_levels_warned

This option is commented out by default.

Guardrail to warn about or reject read consistency levels. By default, all consistency levels are allowed.

Default Value: []

read_consistency_levels_disallowed

This option is commented out by default.

Default Value: []

write_consistency_levels_warned

This option is commented out by default.

Guardrail to warn about or reject write consistency levels. By default, all consistency levels are allowed.

Default Value: []

write_consistency_levels_disallowed

This option is commented out by default.

Default Value: []

partition_size_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when writing partitions larger than threshold, expressed as 100MiB, 1GiB, etc. The guardrail is only checked when writing sstables (flush and compaction), and exceeding the fail threshold on that moment will only log an error message, without interrupting the operation. This operates on a per-sstable basis, so it won't detect a large partition if it is spread across multiple sstables. The warning threshold replaces the deprecated config property compaction_large_partition_warning_threshold. The two thresholds default to null to disable.

partition_size_fail_threshold

This option is commented out by default.

partition_tombstones_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when writing partitions with more tombstones than threshold. The guardrail is only checked when writing sstables (flush and compaction), and exceeding the fail threshold on that moment will only log an error message, without interrupting the operation. This operates on a per-sstable basis, so it won't detect a large partition if it is spread across multiple sstables. The warning threshold replaces the deprecated config property compaction_tombstone_warning_threshold. The two thresholds default to -1 to disable.

Default Value: -1

partition_tombstones_fail_threshold

This option is commented out by default.

Default Value: -1

column_value_size_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when writing column values larger than threshold. This guardrail is only applied to the values of regular columns because both the serialized partitions keys and the values of the components of the clustering key already have a fixed, relatively small size limit of 65535 bytes, which is probably lesser than the thresholds defined here. Deleting individual elements of non-frozen sets and maps involves creating tombstones that contain the value of the deleted element, independently on whether the element existed or not. That tombstone value is also guarded by this guardrail, to prevent the insertion of tombstones over the threshold. The downside is that enabling or

raising this threshold can prevent users from deleting set/map elements that were written when the guardrail was disabled or with a lower value. Deleting the entire column, row or partition is always allowed, since the tombstones created for those operations don't contain the CQL column values. This guardrail is different to max_value_size. max_value_size is checked when deserializing any value to detect sstable corruption, whereas this guardrail is checked on the CQL layer at write time to reject regular user queries inserting too large columns. The two thresholds default to null to disable. Min unit: B

column_value_size_fail_threshold

This option is commented out by default.

collection_size_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when encountering larger size of collection data than threshold. At query time this guardrail is applied only to the collection fragment that is being writen, even though in the case of non-frozen collections there could be unaccounted parts of the collection on the sstables. This is done this way to prevent read-before-write. The guardrail is also checked at sstable write time to detect large non-frozen collections, although in that case exceeding the fail threshold will only log an error message, without interrupting the operation. The two thresholds default to null to disable. Min unit: B

collection_size_fail_threshold

This option is commented out by default. Min unit: B

items_per_collection_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when encountering more elements in collection than threshold. At query time this guardrail is applied only to the collection fragment that is being writen, even though in the case of non-frozen collections there could be unaccounted parts of the collection on the sstables. This is done this way to prevent read-before-write. The guardrail is also checked at sstable write time to detect large non-frozen collections, although in that case exceeding the fail threshold will only log an error message, without interrupting the operation. The two thresholds default to -1 to disable.

Default Value: -1

items_per_collection_fail_threshold

This option is commented out by default.

Default Value: -1

allow_filtering_enabled

This option is commented out by default.

Guardrail to allow/disallow querying with ALLOW FILTERING. Defaults to true. ALLOW FILTERING can potentially visit all the data in the table and have unpredictable performance.

Default Value: true

simplestrategy_enabled

This option is commented out by default.

Guardrail to allow/disallow setting SimpleStrategy via keyspace creation or alteration. Defaults to true.

Default Value: true

fields_per_udt_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when creating a user-defined-type with more fields in than threshold. Default -1 to disable.

Default Value: -1

fields_per_udt_fail_threshold

This option is commented out by default.

Default Value: -1

vector_dimensions_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when creating a vector column with more dimensions than threshold. Default -1 to disable.

Default Value: -1

vector_dimensions_fail_threshold

This option is commented out by default.

Default Value: -1

alter_table_enabled

This option is commented out by default.

Guardrail to indicate whether or not users are allowed to use ALTER TABLE commands to make column changes to tables

Default Value: true

data_disk_usage_percentage_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when local data disk usage percentage exceeds threshold. Valid values are in [1, 100]. This is only used for the disks storing data directories, so it won't count any separate disks used for storing the commitlog, hints nor saved caches. The disk usage is the ratio between the amount of space used by the data directories and the addition of that same space and the remaining free space on disk. The main purpose of this guardrail is rejecting user writes when the disks are over the defined usage percentage, so the writes done by background processes such as compaction and streaming don't fail due to a full disk. The limits should be defined accordingly to the expected data growth due to those background processes, so for example a compaction strategy doubling the size of the data would require to keep the disk usage under 50%. The two thresholds default to -1 to disable.

Default Value: -1

data_disk_usage_percentage_fail_threshold

This option is commented out by default.

Default Value: -1

data_disk_usage_max_disk_size

This option is commented out by default.

Guardrail that allows users to define the max disk size of the data directories when calculating thresholds for disk_usage_percentage_warn_threshold and disk_usage_percentage_fail_threshold, so if this is greater than zero they become percentages of a fixed size on disk instead of percentages of the physically available disk size. This should be useful when we have a large disk and we only want to use a part of it for Cassandra's data directories. Valid values are in [1, max available disk size of all data directories]. Defaults to null to disable and use the physically available disk size of data directories during calculations. Min unit: B

minimum_replication_factor_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when the minimum replication factor is lesser than threshold. This would also apply to system keyspaces. Suggested value for use in production: 2 or higher

Default Value: -1

minimum_replication_factor_fail_threshold

This option is commented out by default.

Default Value: -1

maximum_replication_factor_warn_threshold

This option is commented out by default.

Guardrail to warn or fail when the maximum replication factor is greater than threshold. This would also apply to system keyspaces.

Default Value: -1

maximum_replication_factor_fail_threshold

This option is commented out by default.

Default Value: -1

zero_ttl_on_twcs_enabled

This option is commented out by default.

Guardrail to enable a CREATE or ALTER TABLE statement when default_time_to_live is set to 0 and the table is using TimeWindowCompactionStrategy compaction or a subclass of it. It is suspicious to use default_time_to_live set to 0 with such compaction strategy. Please keep in mind that data will not start to automatically expire after they are older than a respective compaction window unit of a certain size. Please set TTL for your INSERT or UPDATE statements if you expect data to be expired as table settings will not do it. Defaults to true. If set to false, such statements fail and zero_ttl_on_twcs_warned flag is irrelevant.

Default Value: true

zero_ttl_on_twcs_warned

This option is commented out by default. Guardrail to warn a user upon executing CREATE or ALTER TABLE statement when default_time_to_live is set to 0 and the table is using TimeWindowCompactionStrategy compaction or a subclass of it. Defaults to true. if zero_ttl_on_twcs_enabled is set to false, this property is irrelevant as such statements will fail.

Default Value: true

non_partition_restricted_index_query_enabled

This option is commented out by default.

Guardrail enabling secondary index queries that do not restrict on partition key (defaults to true)

Default Value: true

sai_sstable_indexes_per_query_warn_threshold

This option is commented out by default. Maximum number of referenced SAI indexes on a replica when executing a SELECT query before emitting a warning (defaults to 32)

Default Value: 32

sai_sstable_indexes_per_query_fail_threshold

This option is commented out by default. Maximum number of referenced SAI indexes on a replica when executing a SELECT query before emitting a failure (defaults to -1 to disable)

Default Value: -1

default_secondary_index

The default secondary index implementation when CREATE INDEX does not specify one via USING. ex. "legacy_local_table" - (default) legacy secondary index, implemented as a hidden table ex. "sai" - "storage-attched" index, implemented via optimized SSTable/Memtable-attached indexes

Default Value: sai

default_secondary_index_enabled

Whether a default secondary index implementation is allowed. If this is "false", CREATE INDEX must specify an index implementation via USING.

Default Value: true

startup_checks

This option is commented out by default.

Startup Checks are executed as part of Cassandra startup process, not all of them are configurable (so you can disable them) but these which are enumerated bellow. Uncomment the startup checks and configure them appropriately to cover your needs.

Default Value (complex option):

```
    # Verifies correct ownership of attached locations on disk at startup. See CASSANDRA-
16879 for more details.
    #  check_filesystem_ownership:
    #    enabled: false
    #    ownership_token: "sometoken" # (overriden by "CassandraOwnershipToken" system
property)
    #    ownership_filename: ".cassandra_fs_ownership" # (overriden by
"cassandra.fs_ownership_filename")
    # Prevents a node from starting if snitch's data center differs from previous data
center.
    #  check_dc:
    #    enabled: true # (overriden by cassandra.ignore_dc system property)
    # Prevents a node from starting if snitch's rack differs from previous rack.
    #  check_rack:
    #    enabled: true # (overriden by cassandra.ignore_rack system property)
    # Enable this property to fail startup if the node is down for longer than
gc_grace_seconds, to potentially
    # prevent data resurrection on tables with deletes. By default, this will run against
all keyspaces and tables
    # except the ones specified on excluded_keyspaces and excluded_tables.
    #  check_data_resurrection:
    #    enabled: false
    # file where Cassandra periodically writes the last time it was known to run
    #    heartbeat_file: /var/lib/cassandra/data/cassandra-heartbeat
    #    excluded_keyspaces: # comma separated list of keyspaces to exclude from the
check
    #    excluded_tables: # comma separated list of keyspace.table pairs to exclude from
the check
```

storage_compatibility_mode

This property indicates with what Cassandra major version the storage format will be compatible with.

The chosen storage compatibility mode will determine the versions of the written sstables, commitlogs, hints, etc. For example, if we're going to remain compatible with Cassandra 4.x, the value of this property should be 4, which will make us use sstables in the latest N version of the BIG format.

This will also determine if certain features that depend on newer formats are available. For example, extended TTL (up to 2106) depends on the sstable, commit-log, hints, and messaging versions introduced by Cassandra 5.0, so that feature won't be available if this property is set to CASSANDRA_4. See the upgrade guide for more details.

Possible values are:

- CASSANDRA_4: Stays compatible with the 4.x line in features, formats and component versions.
- UPGRADING: The cluster monitors the version of each node during this interim stage. This has a cost but ensures all new features, formats, versions, etc. are enabled safely.
- NONE: Start with all the new features and formats enabled.

A typical upgrade would be:

1. Do a rolling upgrade, starting all nodes in CASSANDRA_X compatibility mode.
2. Once the new binary is rendered stable, do a rolling restart with the UPGRADING mode. The cluster will keep new features disabled until all nodes are started in the UPGRADING mode; when that happens, new features controlled by the storage compatibility mode are enabled.
3. Do a rolling restart with all nodes starting with the NONE mode. This eliminates the cost of checking node versions and ensures stability. If Cassandra was started at the previous version by accident, a node with disabled compatibility mode would no longer toggle behaviors as when it was running in the UPGRADING mode.

Default Value: NONE