BLOOM FILTERS

WHAT ARE BLOOM FILTERS :

Cassandra combines data from RAM (in memtables) and disk (in SSTables) in the read path. Cassandra uses a data structure called a bloom filter to avoid having to go through each and every SSTable data file to find the partition that is being requested.

Cassandra can decide one of two probable states using bloom filters, a probabilistic data structure: either the data absolutely does not exist in the provided file, or the data probably does.

Bloom filters can be made more accurate by allowing them to use more RAM, even when they cannot ensure that the data resides in a particular SSTable. Administrators can fine-tune this behavior for each table by setting the bloom\_filter\_fp\_chance to a float value between 0 and 1.

For tables that use the LeveledCompactionStrategy, the default value for bloom\_filter\_fp\_chance is 0.1; in all other situations, it is 0.01.

Operators should not take bloom filters into account when choosing the maximum heap size because bloom filters are stored offheap but in RAM. Memory usage grows non-linearly with precision (i.e., as bloom\_filter\_fp\_chance approaches 0). For example, a bloom filter with bloom\_filter\_fp\_chance = 0.01 will require around three times as much memory as a table with bloom\_filter\_fp\_chance = 0.1.

Bloom\_filter\_fp\_chance typically has values between 0.01 (1%) and 0.1 (10%) false-positive chance, which is the possibility that Cassandra will search an SSTable for a row but discover that it is not present on the disk. The use case should guide the parameter tuning:

Setting the bloom\_filter\_fp\_chance to a numerically lower value (like 0.01) may help users with slower disks and more RAM avoid unnecessary IO operations.

A greater bloom\_filter\_fp\_chance may be tolerated by users with less RAM, denser nodes, or highly fast disks in order to conserve RAM at the expense of unnecessary IO operations.

Bloom filter fp chance can be set to a considerably larger amount in workloads that don't read much or just read by scanning the full data set (like analytics workloads).

The bloom filter false positive chance is visible in the DESCRIBE TABLE output as the field bloom\_filter\_fp\_chance. Operators can change the value with an ALTER TABLE statement: :

ALTER TABLE keyspace.table WITH bloom\_filter\_fp\_chance=0.01

Operators should be aware that this is a gradual shift, as the bloom filter is computed at the time the file is written and is stored on disk as part of the SSTable's Filter component. An operator can use nodetool scrub or nodetool upgradesstables -a to force an SSTable rewrite in order to make changes to bloom\_filter\_fp\_chance take effect. This will rebuild the sstables on disk and regenerate the bloom filters while doing so. New files on disk will be written with the new bloom\_filter\_fp\_chance upon issuing an ALTER TABLE statement, but existing sstables will not be modified until they are compacted.

BULK LOADING :

Several programs facilitate loading Apache Cassandra data in bulk. SSTables are the required format for the data to be bulk loaded. No, Cassandra does not support directly loading data in other formats, like CSV, JSON, and XML. The cqlsh COPY command is not a viable choice for large volumes of data, even though it can load CSV data. Using bulk loading enables:

* Restore snapshots and incremental backups. SSTables are already the format for backups and snapshots.
* Move the current SSTables to a different cluster. The data may have a different replication mechanism or number of nodes.
* Add outside data to a cluster.

TOOLS USED :

Two commands or tools are available in Cassandra for bulk loading data:

sstableloader, another name for Cassandra Bulk loader

The import command for nodetool

If the Cassandra installation bin directory is present in the PATH environment variable, you can use the sstableloader and nodetool import. Alternatively, these might be viewed straight from the bin directory. The keyspaces and tables established in Backups are used in the examples.

SSTABLE LOADER :

The primary tool for uploading data in bulk is the sstableloader. SSTable data files are streamed to an operational cluster by sstableloader in accordance with the replication factor and replication method. The table to which data uploads is not required to be empty.

Runtime requirements for sstableloader are as follows:

To connect to and obtain ring information, first hosts should be separated by one or more commas.

a directory path that loads the SSTables

sstableloader <dir\_path> [options]

The SSTables located in the directory <dir\_path> are bulk loaded into the configured cluster by Sstableloader. The intended keyspace or table name is <dir\_path>. For instance, the files Standard1-g-1-Data.db and Standard1-g-1-Index.db must be in the directory /path/to/Keyspace1/Standard1/ in order to load an SSTable with the name Standard1-g-1-Data.db into Keyspace1/Standard1.

ACCEPTING TARGER KEYSPACE NAME :

Some Cassandra DBAs store a whole data directory, usually as part of a backup plan. When data corruption is discovered, it is typical to restore the data using an alternative keyspace name within the same cluster (big clusters with 200 nodes).

Keyspace name is currently derived by sstableloader from the folder structure. Version 4.0 adds support for the --target-keyspace option (CASSANDRA-13884) as a way to configure the target keyspace name as part of sstableloader.

The -d,--nodes <initial hosts> option is necessary for the following options to be supported:

| **Short option** | **Long option** | **Description** |
| --- | --- | --- |
| -alg | --ssl-alg <ALGORITHM> | Client SSL algorithm (default: SunX509). |
| -ap | --auth-provider <auth provider class name> | Allows the use of a third party auth provider. Can be combined with -u <username> and -pw <password> if the auth provider supports plain text credentials. |
| -ciphers | --ssl-ciphers <CIPHER-SUITES> | Client SSL. Comma-separated list of encryption suites. |
| -cph | --connections-per-host <connectionsPerHost> | Number of concurrent connections-per-host. |
| -d | --nodes <initial\_hosts> | **Required**. Connect to a list of (comma separated) hosts for initial cluster information. |
| -f | --conf-path <path\_to\_config\_file> | Path to the cassandra.yaml path for streaming throughput and client/server SSL. |
| -h | --help | Display help. |
| -i | --ignore <NODES> | Do not stream to this comma separated list of nodes. |
| -ks | --keystore <KEYSTORE> | Client SSL. Full path to the keystore. |
| -kspw | --keystore-password <KEYSTORE-PASSWORD> | Client SSL. Password for the keystore.Overrides the client\_encryption\_options option in [cassandra.yaml](https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/configuration/configCassandra_yaml.html#configCassandra_yaml__client_encryption_options) |
|  | --no-progress | Do not display progress. |
| -p | --port <rpc port> | RPC port (default: 9160 [Thrift]). |
| -prtcl | --ssl-protocol <PROTOCOL> | Client SSL. Connections protocol to use (default: TLS).Overrides the server\_encryption\_options option in [cassandra.yaml](https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/configuration/configCassandra_yaml.html#configCassandra_yaml__server_encryption_options) |
| -pw | --password <password> | Password for Cassandra authentication. |
| -st | --store-type <STORE-TYPE> | Client SSL. Type of store. |
| -t | --throttle <throttle> | Throttle speed in Mbits (default: unlimited).Overrides the stream\_throughput\_outbound\_megabits\_per\_sec option in [cassandra.yaml](https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/configuration/configCassandra_yaml.html#configCassandra_yaml__stream_throughput_outbound_megabits_per_sec) |
| -tf | --transport-factory <transport factory> | Fully-qualified ITransportFactory class name for creating a connection to Cassandra. |
| -ts | --truststore <TRUSTSTORE> | Client SSL. Full path to truststore. |
| -tspw | --truststore-password <TRUSTSTORE-PASSWORD> | Client SSL. Password of the truststore. |
| -u | --username <username> | User name for Cassandra authentication. |
| -v | --verbose | Verbose output. |

You may set up streaming throughput, client and server encryption options, and more by running the cassandra.yaml file on the command line with the -f option. The only parameters that are read from the cassandra.yaml file are stream\_throughput\_outbound\_megabits\_per\_sec, server\_encryption\_options, and client\_encryption\_options. Option read from cassandra.yaml can be overridden using the relevant command line options.