Cassandra JP Upgrade Runbook

ToDo

- [ ] Type your task here, using "@" to assign to a user and "//" to select a due date

Overview

To ensure we have the most recent bug fixes and performance improvements we intend to upgrade from Cassandra 1.2.19 to 2.0.14.

The general approach is:

1. Prepare the cluster.
2. To each node in turn:
    a. Shutdown then node.
    b. Upgrade the installation.
    c. Restart the node.
3. Upgrade on disk files.
4. Cleanup.

Changes

Tickets

The following tickets (extracted from CHANGES.txt) are expected to benefit our installation:

Ticket	Description
CASSANDRA-8550	Use more efficient slice size for querying internal secondary index tables.

cassandra.yaml

The following configuration settings in `cassandra.yaml` have been deprecated, removed or modified:

Setting	Change
reduce_cache_sizes_at	Setting has been removed.

---

reduce_cache_capacity_to	Setting has been removed.
flush_largest_memtables_at	Setting has been removed.
authority	Has been replaced with `authorizer`.
index_interval	Is now a Table property, Cassandra will pick up the value defined in cassanda.yaml as the default for existing Tables.
native_transport_min_threads	Setting has been removed.
read_request_timeout_in_ms	Default is now 5000.
write_request_timeout_in_ms	Default is now 2000.
truncate_request_timeout_in_ms	Default is now 60000.
memtable_total_space_in_mb	Default has changed from 1/3 of the heap to 1/4.

cassandra-env.sh

The following configuration settings in `cassandra-env.sh` have been deprecated, removed or modified:

Setting	Change

Schema Changes

The following schema settings have been deprecated, removed or modified:

Setting	Change
read_repair_chance	The default has changed from 0.1 to 0.0.
local_read_repair_chance	The default has changed from 0.0 to 0.1.

News

The following new features or incompatibilities (extracted from NEWS.txt) are expected to benefit or impact our installation:

Change	Impact	Actions
Java 7 is now required.	Cassandra will only start using Java version 7 (or above).	Ensure the latest release of Oracle Java 1.7 is installed on the nodes. **Recommendation:** We recommend upgrading Java before performing this Cassandra upgrade.
Upgrading is ONLY supported from Cassandra 1.2.9 or later. This goes for sstable compatibility as well as network. When upgrading from an earlier release, upgrade to 1.2.9 first and run upgradesstables before proceeding to 2.0.	The upgrade will fail if incompatible versions of SSTables are found, or incompatible wire formats are encountered.	See Assumptions.
Inputting blobs as string constants is now fully deprecated in favour of blob constants. Make sure to update your applications to use the new syntax while you are still on 1.2 (which supports both string and blob constants for blob input) before upgrading to 2.0.	Client code using the older format for blobs will fail.	See Assumptions.
VNodes are enabled by default in `cassandra.yaml`. `initial_token` for non-vnode deployments has been removed from the example yaml, but is still respected if specified.	No impact on existing or upgraded clusters. New clusters will use VNodes by default with `num_tokens` set to 256.	None.
Disabling auto compactions by setting min/max compaction threshold to 0 has been deprecated, instead, use the `nodetool` commands 'disableautocompaction' and 'enableautocompaction' or set the compaction strategy option `enabled = false`.	None.	Change operational run books to use `nodetool` commands.
After performance testing for CASSANDRA-5727, the default LCS file size has been changed from 5MB to 160MB.	No impact on existing Tables. New tables created using `LeveledCompactionStrategy` will be created using the new default for `sstable_size_in_mb` of 160.	Existing tables using `LeveledCompactionStrategy` should be modified to `sstable_size_in_mb` to 160. The SSTables will be gradually re-compacted using the new size, or it can be forced. See Clean Up section below.
CQL2 has been deprecated, and will be removed entirely in 2.2. See CASSANDRA-5918 for details.	No change on existing applications.	Applications using CQL 2 should be upgraded to use CQL 3.

Speculative retry defaults to 99th percentile (See blog post at http://www.datastax.com/dev/blog/rapid-read-protection-in-cassandra-2-0-2)	Read requests will make additional requests to unused replicas once the latency for a request has passed the 99th percentile request latency for the Table. This feature will smooth out request latency when nodes are being restarted or experiencing pauses.	Speculative Retry may result in additional, local, read operations. The number of time Speculative Retry has fired can be monitored using the `org.apache.cassandra.metrics.ColumnFamily.$KEYSPAC.$TABLE.SpeculativeRetries` metric. If the client side socket timeout is lower than the server side `read_request_timeout_in_ms` Speculative Retry may run after the client has closed the connection. This will result in the cluster performing unnecessary reading operations whose result will be ignored. **Recommendation:** We recommend disabling Speculative Retry or reducing the `read_request_timeout_in_ms` to the desired timeout and disabling the client side socket timeout.
Configurable metrics reporting (see conf/metrics-reporter-config-sample.yaml)	No impact on existing systems.	Existing monitoring systems should be upgraded to take advantage of the rich metrics now available. See the `metrics-reporter-config-sample.yaml` configuration file for information.

Current Situation

The current output from nodetool status is:

```
$ nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns (effective) Host ID Rack
UN 10.193.33.19 309.86 GB 1 37.5% ee862b5c-0897-4154-80f4-11d2cb9c5e5c rack1
UN 10.193.33.32 293.73 GB 1 37.5% f61495e2-5932-4d83-bf2b-9d38f2aba4c7 rack1
UN 10.193.33.18 296 GB 1 37.5% 660db5c9-1e73-49db-9e54-4eabcff3b7cd rack1
UN 10.193.33.33 303.75 GB 1 37.5% 6bbf072d-f47f-4359-b284-71c80981b0c5 rack1
UN 10.193.33.34 293.27 GB 1 37.5% 5e166bb0-e88d-42fd-b8b9-7e90867b2115 rack1
UN 10.193.33.17 307.56 GB 1 37.5% 82c580f0-c41a-4994-b6a2-56e24b78af0b rack1
UN 10.193.33.35 301.72 GB 1 37.5% ec334063-cc93-4d4c-ba62-4479bb479e18 rack1
UN 10.193.33.16 296.47 GB 1 37.5% d3d46bc8-904d-44a7-9136-2f51a2f778f6 rack1
Datacenter: GQ1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns (effective) Host ID Rack
UN 10.193.121.15 488.6 GB 1 37.5% 95677836-bbf3-4a41-ad45-a635a65dc357 rack1
UN 10.193.121.14 317.17 GB 1 37.5% ca6c4be5-3a10-44a9-be32-83b4a18652a7 rack1
UN 10.193.121.13 433.5 GB 1 37.5% 9e81e349-41c1-4614-940b-dff7ce66844d rack1
UN 10.193.121.12 507.88 GB 1 37.5% 47b732d0-d5bd-43d1-bd31-f947a244a948 rack1
UN 10.193.121.86 536.62 GB 1 37.5% 0419f1bd-c76b-4249-8c0b-8d9b5cb3ae43 rack1
UN 10.193.121.87 455.65 GB 1 37.5% 015f9e64-f9b2-4ec5-814e-59592dc6a946 rack1
UN 10.193.121.84 296.63 GB 1 37.5% 89cf85f8-0b9e-4d5d-8deb-8730a454dbe3 rack1
UN 10.193.121.85 686.85 GB 1 37.5% ae5031be-281a-4790-9963-8bac2e5956c6 rack1

$ date
Wed Apr 20 15:25:54 EST 2015
```

The current cassandra version is:

```
$ date
Wed Apr 20 15:25:54 EST 2015
$ nodetool version
ReleaseVersion: 1.2.19
```

The current gossip state is:

```
$ nodetool gossipinfo
localhost/127.0.0.1
  generation:1429501656
  heartbeat:1748
  NET_VERSION:7
  LOAD:61714.0
  HOST_ID:0afebaa3-ea16-4530-9826-5e4a200ca6a4
  DC:datacenter1
  STATUS:NORMAL,-1056300706838771218
  RACK:rack1
  SCHEMA:2bd33b31-5f77-3161-8e39-e9d53915f3bd
  SEVERITY:2.220446049250313E-16
  RELEASE_VERSION:2.0.11
  RPC_ADDRESS:127.0.0.1
```

Assumptions

Assumptions are to be checked before the starting the work, they provide a form of smoke test.

Name	Description	Checked
Disk Space	We assume each node has at least 50% free disk space. Additional disk space may be needed when performing repairs, taking snapshots, and rewriting SSTables to the new on-disk format.	

No Errors	No ERROR messages have been logged on any node in the last 72 hours. Messages at the ERROR level in the Cassandra logs should be treated as error conditions that need to be resolved.	
Gossip Stable	All entries in the output from `nodetool gossipinfo` have the gossip state `STATUS:NORMAL`. Nodes listed in Gossip that are not in the NORMAL state may have recently left the cluster, unsuccessfully left the cluster, be joining the cluster, or participating as an observer. The history of all non- NORMAL nodes should be investigated before proceeding to ensure the upgrade can complete. Unknown or ghost nodes may be removed via JMX using the `unsafeAssassinateEndpoint` method of the org.apache.cassandra.net:type=Gossiper MBean. NOTE: as the name suggests this method is not intended for day to day use, we recommend observing the cluster closely when using it.	
No Dropped Messages	No "Dropped Message" messages have been logged on any node in the last 72 hours. Dropped messages indicate the cluster is overloaded. Making changes to an overloaded cluster may reduce performance further. We recommend addressing the root cause of the dropped messages (such as JVM GC pauses or blocked Flush Writers) before upgrading. If the root cause cannot be identified, or it may be resolved by upgrading, the upgrade may proceed.	
Backups Disabled	Automatic backups have been disabled. Automatic backups may interfere with, or be interfered with, the upgrade process. On smaller clusters we recommend disabling them until the backup is complete. For larger clusters, or longer upgrade processes, we recommend testing and monitoring to ensure the backup process works as expected.	
Repair Disabled	Automatic Repairs have been disabled. Cassandra can only repair across nodes when all SSTables are in the current file format, for version 2.0.1 and above this means `-jb-` is in the file name. Upgrading major versions will typically result in a file format change which will cause repairs to fail. Additionally repair requires that all replicas for a token range be available before it can start. Once the repair is started, restarting a node will cause the process is fail. As the upgrade requires a rolling restart any scheduled repairs will most likely fail.	

On Disk Format	All on disk SSTables use the `-ic-` format.
	
	Cassandra 2.0 can only be upgraded to from Cassandra 1.2.9 and above, which uses the `-ic-` on disk format. For example `Keyspace1-Standard1-ic-10-Data.db`. The file formats supported by Cassandra 2.0 can be found in Descriptor.java
	
	SSTables that do not use the `-ic-` version can be identified using:
	
	`find /var/lib/cassandra/data/ -type f | grep -v "\-ic\-"`
	
	If a Table contains older format SSTables they can be upgraded using:
	
	`sstableupgrade <keyspace> <cf>`
	
	Note: If secondary indexes are used they must be upgraded individually using the following for each index:
	
	`sstableupgrade <keyspace> <index_name>`	

Minimum Version	All nodes are running Cassandra version 1.2.9 or above. When upgrading to Cassandra 2.0 all nodes must be on at least Cassandra 1.2.9 to ensure wire format compatibility. While 1.2.9 is the minimum we recommend upgrading from the most recent version of 1.2.	
Monitoring	Monitoring is in place for all nodes, for the cluster as a whole, and for the application. Upgrading may result in temporary reduction in performance, as it simulates a series of temporary node failures. Understanding how this impacts the performance of the system is crucial when working through the process. After the upgrade has completed it is also important to understand how the upgrade has affected cluster performance.	
Blob Constants	All client code is using the correct format for blob constants. See News section above.	
Availability	Areas of the application that require Strong Consistency are using the QUORUM Consistency Level and a Replication Factor of 3. When a QUORUM is used with a Replication Factor below 3 all nodes must be available for requests to start. A rolling restart using this configuration will result in full or partial unavailability while a node is down.	

Risks

The following risks have been identified:

Name	Description	Impact	Monitoring and Mitigation
Latency	Read and write latency may be affected during the migration process as nodes are stopped and started and rejoin with a cold page cache and cold JVM.	Increased read and write latency until the nodes warm up and the node availability stabilises.	This risk can be mitigated by softly shutting down nodes using the process described below. Adding a delay of 3 to 5 minutes between starting a new node and shutting down the next old node will allow the page cache to heat up. The risk can be monitored by tracking cluster level read and write latency at the 95% percentile for the duration of the work. If the latency rises to an unacceptably high level the migration can be stopped until it returns to the previous level.
Availability	When a node is down during the rolling migration process the cluster will not have redundancy. The loss of second node will result in a partial or complete outage of data in the cluster.	Loss of cluster availability.	This risk can be mitigated by ensuring that all nodes are `UP` before migrating a node, reducing the time nodes are `DOWN` for, and monitoring the state of cluster while migrating a node. This can be done using `nodetool status` or the current monitoring tools.
Incompatibility	Unexpected issues in driver compatibility or server functionality may occur after upgrading.	Loss of function or availability.	This risk can be mitigated by testing the upgrade in QA or upgrading one node first, checking that it is processing client traffic, and ensuring the application is free from errors.
Disk Space	Additional disk space used doing repairs, taking snapshots, and upgrading SSTables may result in low disk free space.	Loss of compaction functionality due to lack of space for data files. Loss of write functionality does to lack of space for commit log or data files.	The risk can be mitigated by ensuring at least 50% free disk space before starting the upgrade. It can be monitored by tracking free disk space per node, and alerting when it is below 50%. If disk space runs low remove snapshots and any Java Heap dumps.

Phase 0. Check Assumptions

Assumptions must be checked to ensure the decisions based on them are correct. We do this by working through the list in the Assumptions section.

Step 1

Check the assumptions in the Assumptions section and update the table with the outcome.

End of Phase

At the end of this phase:

1. Run Book assumptions will have been verified.

Preparing the cluster will create rollback points incase we encounter issues. We do this by creating Cassandra snapshots.

Phase 1. Prepare the cluster

Preparing the cluster will put it into the correct state for starting the upgrade.

Step 1

Create a snapshot on each node using:

```
nodetool snapshot -t pre-2.0
```

This will create a snapshot called `pre-2.0`

Step 2

Update the packaging system on all nodes to point to the 2.0 repository.

Step 3

We recommend disabling Hinted Handoff on clusters with a very high throughput using:

```
nodetool disablehandoff
```

During the rolling restart the remaining nodes will store hints for the `DOWN` node. Storing a lot of Hints in version 2.0 may result in excessive memory usage due to CASSANDRA-7546. If hints are disabled a repair must be run after the upgrade has been completed.

End of Phase

At the end of this phase:

1. Run Book assumptions will have been verified.
2. All nodes will have a snapshot to serve as a rollback point.
3. All nodes will have the correct repository branch for version 2.0.

Phase 2. Upgrade First Node to 2.0.14

With the cluster ready we can now upgrade to version 2.0.14. We do this using a rolling upgrade process to have zero down time.

Do the following to each node in the cluster in turn.

Step 1

Ensure all nodes are `UP` using:

```
nodetool status
```

Shutdown the node cleanly using:

```
nodetool disablethrift
nodetool disablebinary
sleep 10
nodetool disablegossip
nodetool drain
sudo service cassandra stop
```

Stopping the client API's will allow existing requests to complete reducing the impact on clients.

Step 2

Clear the commit log to avoid replaying on restart:

```
sudo mkdir /var/lib/cassandra/commitlog-pre-2.0.14
sudo mv /var/lib/cassandra/commitlog/*.log /var/lib/cassandra/commitlog-pre-2.0.14/
ls -lah /var/lib/cassandra/commitlog/
```

Step 3

Upgrade the package to 2.0.14, if prompted the current (modified) configuration files should be left in place.

Recommendation: We recommend installing a specified version ("pinning") to avoid accidental automatic upgrades, for example:

```
sudo apt-get update
sudo apt-get install cassandra=2.0.14
```

Step 4

Modify `cassandra.yaml` to remove the following deprecated options:

```
reduce_cache_sizes_at
reduce_cache_capacity_to
flush_largest_memtables_at
native_transport_min_threads
index_interval
```

If a non default `index_interval` has been set the property should not be removed from `cassandra.yaml` until after the upgrade. This will allow the schema to be updated with the configured value.

Step 5

Start the node using:

```
sudo service cassandra start
```

Monitor `/var/log/cassandra/log/system.log` on startup for errors.

After upgrading one node ensure other nodes see the upgraded node as `UP`, and the upgraded node sees all other nodes as `UP`. This can be done using:

```
nodetool status
```

Ensure the upgraded node is processing read and write traffic, as well as requests from clients. This can be done by watching for "Completed" tasks using:

```
watch -d nodetool tpstats
```

Specifically:

- `ReadStage` tasks are local reads.
- `MutationStage` tasks are local writes.
- `RequestResponseStage` tasks are used to process the responses from replicas when acting as a coordinator.

Step 6

After the first node has been upgraded confirm applications are functioning correctly before upgrading the remaining nodes.

If a rollback is determined is necessary, stop the upgraded node using:

```
sudo service cassandra stop
```

Copy any new SSTables to a backup location using:

```
mkdir -p /var/lib/cassandra/2.0-files/$KEYSPACE/$TABLE
cp /var/lib/cassandra/data/$KEYSPACE/$TABLE/*-jb-* /var/lib/cassandra/2.0-files/$KEYSPACE/$TABLE
```

Backing up the new SSTables is a precaution, see the Note below.

Delete the live SSTables (both old and new) from each Table using:

```
rm /var/lib/cassandra/data/$KEYSPACE/$TABLE/*
```

Restore the snapshot files for each Table using:

```
cp /var/lib/cassandra/data/$KEYSPACE/$TABLE/snapshots/pre-2.0/* /var/lib/cassandra/data/$KEYSPACE/$TABLE/
```

This node will experience a data loss of any data written to it while it was running with 2.0. Data written to the cluster during this time will (in theory) be in an Eventually Consistent state. As we can no longer confirm how many nodes committed the data disk disk. A repair should be run on the node to repair consistency using:

```
nodetool repair
```

(The `-pr` node should **not** be used.)

The backup may be removed once the repair has been completed. If the repair cannot be completed, or data was written using Consistency Level `ONE`, the backup files should be restored. This is done by converting the SSTables to JSON format using `sstable2json` and back to the correct format using `json2sstable`.

End of Phase

At the end of this phase:

1. One node has been upgraded to version 2.0.14.

Phase 3. Upgrade Remaining Nodes to 2.0.14

After the first node has successfully been upgraded we can upgrade the remaining nodes.

Step 1

Migrate the remaining nodes to 2.0.14 one at a time using Steps 1 to 5 from Phase 1.

Migrate the remaining nodes to 2.0.10.

End of Phase

At the end of this phase:

1. All nodes have been upgraded to version 2.0.14.
2. On disk data may be in the 1.2 format.

Phase 4. Cleanup

If adjustments were made to configuration settings before the upgrade these can now be reversed.

Step 1

Enable Hinted Handoff if it was previously disabled using:

```
nodetool enablehandoff
```

End of Phase

At the end of this phase:

1. All nodes have been upgraded to version 2.0.14.
2. All configuration settings have been restored.
3. On disk data may be in the 1.2 format.

Phase 5. Upgrade SSTables

The SSTables must be upgraded as the on disk format has changed between version 1.2.19 and 2.0.14. Version 1.2.19 used the format specifier `-ic-` while 2.0.14 uses `-jb-`. Repair cannot be run until all SSTables have the current version specifier. We can upgrade the SSTables using `nodetool`.

Upgrading the SSTables will re-write each SSTable in turn using the new on disk format.

> Upgrading the SStables is a throttled process governed by the `compaction_throughput_mb_per_sec` setting in `cassandra.yaml` and `nodetool setcompactionthroughput`
> It may be run in parallel on multiple nodes, however we recommend allowing the first node to complete so that the duration and impact can be determined.

Step 1

Remove the rollback snapshot using:

```
nodetool clearsnapshot -t pre-2.0
```

Upgrading the SSTables will re-write all the SSTables which will require significant disk space. If the snapshot is left in place the amount of space used by Cassandra may double. The snapshot may be left in place and removed after `upgradesstables` has run if there is sufficient space.

Step 2

Update the SSTables on each node in turn using:

```
nodetool upgradesstables
```

Upgrading SSTables is performed through the compaction subsystem, and may be monitored using:

```
nodetool compactionstats
```

At the end of this task all SSTables (not in a snapshot) should use the `-jb-` format specifier. This can be confirmed using:

```
find /var/lib/cassandra/data/ -type f | grep -v "\-ic\-"
```

End of Phase

At the end of this phase:

1. All nodes will be running version 2.0.14.
2. All data will be in the 2.0.14 format.

Phase 6. Restore Support Services

The cluster has now been fully upgraded, any support services can now be restored.

Step 1

Enable scheduled repair operations.

Step 2

Enable scheduled backup operations.

End of Phase

At the end of this phase:

1. The cluster will be fully upgraded.