Managing Apache Cassandra

Brooke Thorley, VP Technical Operations, Instaclustr

April 2017

Agenda

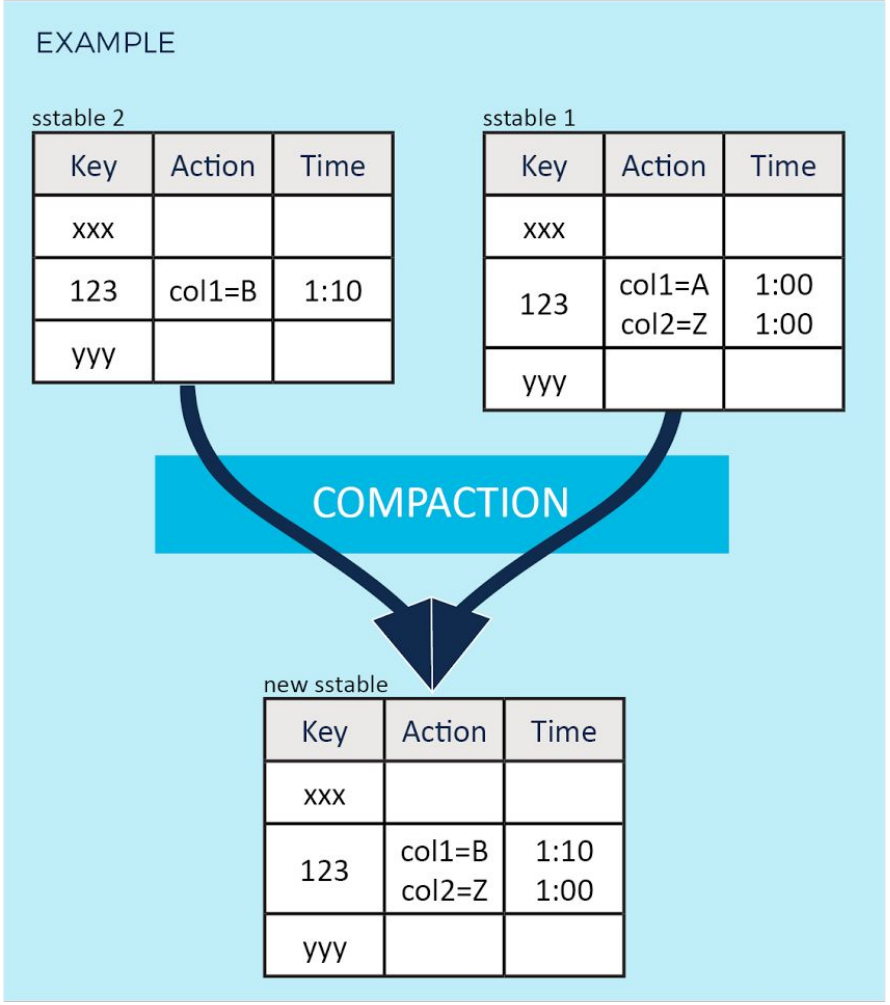
instaclustr

1. Introduction to important concepts
2. Diagnosing Problems
3. Managing Compactions
4. Cluster Mutations
5. Topology design for easier maintenance
6. Final Tips
7. How Instaclustr can help

Compaction Intro

instaclustr

- SSTables are “immutable” - never updated once written to disk.
- Instead all inserts and updates are essentially (logically) written as transaction logs that are reconstituted when read
- Compaction is the process of consolidating transaction logs to simplify reads
- It’s an ongoing background process in Cassandra
- Compaction ≠ Compression



Tombstones

instaclustr

- Data is not immediately purged when deleted. It is marked with a *tombstone* to be purged at a later time.
- Tombstones are ***removed after gc_grace_seconds in the next compaction of the SSTable in which they are stored.***
  - *A tombstoned cell must be propagated to all replica nodes before gc_grace_seconds in order to prevent resurrection of data (zombies). Generally repairs are the only way to ensure this consistency (and is why the default gc_grace_seconds is 10 days as generally repairs will run within that time period).*
- ***Tombstones can remain in the cluster well past gc_grace_seconds.***
  - *E.g In a table using LeveledCompactionStrategy it can be a very long time before the SSTables containing the deleted data move to the next level and are compacted. Furthermore, the nodes will need sufficient free space for these compactions to complete.*
- An alternative is to use **TTL** (time to live) on insert. Once the TTL expires the cell is treated as deleted on all replicas. There is no need to propagate tombstones because each replica will already have it (by way of the expired TTL).
- http://thelastpickle.com/blog/2016/07/27/about-deletes-and-tombstones.html

Eventual Consistency

instaclustr

- **Replication Factor (RF)** - defines how many copies (replicas) of a row should be stored in the cluster.
- **Consistency level (CL)** - How many acknowledgements/responses from replicas before a query is considered a success.

- Inconsistency means that not *all* replicas have a copy of the data, and this can happen for a few reasons:
  - Application uses a low consistency level for writes (eg LOCAL_ONE)
  - Nodes have dropped mutation messages under load
  - Nodes have been DOWN for longer than hinted handoff window (3 hours)
- **Repairs** are how Cassandra fixes inconsistencies and ensures that *all* replicas hold a copy of the data.

Monitoring Cassandra (Metrics + Alerting)

instaclustr

Metric	Description	Frequency
**Node Status	Nodes DOWN should be investigated immediately. `org.apache.cassandra.net:type=FailureDetector`	*Continuous, with alerting*
**Client read latency	Latency per read query over your threshold `org.apache.cassandra.metrics:type=ClientRequest,scope=Read`	*Continuous, with alerting*
**Client write latency	Latency per write query over your threshold `org.apache.cassandra.metrics:type=ClientRequest,scope=Write`	*Continuous, with alerting*
CF read latency	Local CF read latency per read, useful if some CF are particularly latency sensitive.	*Continuous if required*
Tombstones per read	A large number of tombstones per read indicates possible performance problems, and compactions not keeping up or may require tuning	*Weekly checks*
SSTables per read	High number (>5) indicates data is spread across too many SSTables	*Weekly checks*
**Pending compactions	Sustained pending compactions (>20) indicates compactions are not keeping up. This will have a performance impact. `org.apache.cassandra.metrics:type=Compaction,name=PendingTasks`	*Continuous, with alerting*

*Items marked ** give an overall indication of cluster performance and availability*

Diagnosing: Cluster Status

instaclustr

nodetool status – overview of all nodes in the cluster

```
Datacenter: us-west
===========================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address        Load       Tokens       Owns    Host ID                               Rack
UN  10.65.XX.XXX   108.77 GB  256          ?       e462bc9f-9df7-4342-b987-52a86d29c7f4  1a
UN  10.65.XX.XXX   116.28 GB  256          ?       93530c86-3cb3-4d4e-a005-9f02ed4c0b3a  1c
UN  10.65.XX.XXX   109.17 GB  256          ?       ab779176-1513-4849-8531-6ff39037e078  1a
UN  10.65.XX.XXX   103.1 GB   256          ?       cd112339-3224-4b8f-9be0-de26edb3a0d1  1a
UN  10.65.XX.XXX   111.45 GB  256          ?       3bfa406f-63f6-47e7-8798-6f650726ba23  1c
UN  10.65.XX.XXX   110.09 GB  256          ?       5b39c8c2-4896-48b5-940d-d48b12157acf  1a
UN  10.65.XX.XXX   105.18 GB  256          ?       467e03e4-0cdd-4088-b122-6b0e6848f7ed  1c
UN  10.65.XX.XXX   112.22 GB  256          ?       a48b999f-4473-4e85-83b2-1208fa63223c  1a
UN  10.65.XX.XXX   107.69 GB  256          ?       9e48a874-57ca-40df-8053-dfb141389c09  1a
UN  10.65.XX.XXX   109.21 GB  256          ?       cb20eaa4-ba95-452f-9ac0-5ff41010b702  1c
UN  10.65.XX.XXX   119.29 GB  256          ?       3cf1cd91-26ed-4057-b09b-9092c01e03ec  1c
UN  10.65.XX.XXX   109.08 GB  256          ?       d7aff1c4-0ace-46c2-b7db-a18f285fcdc4  1c
```

All nodes should be UN (UP, NORMAL) in all DCs. Investigate any DN (Down) nodes.

Diagnosing - Internals (thread pools)

instaclustr

nodetool tpstats – threadpool statistics (since last Cassandra restart on this node)

Pool Name	Active	Pending	Completed	Blocked	All time blocked
ReadStage	0	0	1073	0	0
MiscStage	0	0	0	0	0
CompactionExecutor	2	2	1759	0	0
MutationStage	128	615267	118435822	0	0
MemtableReclaimMemory	0	0	210	0	0
PendingRangeCalculator	0	0	45	0	0
GossipStage	0	0	12390	0	0
SecondaryIndexManagement	0	0	0	0	0
HintsDispatcher	1	22	10	0	0
RequestResponseStage	1	5	519510274	0	0
Native-Transport-Requests	1	0	38354372	0	21184990
ReadRepairStage	0	0	1	0	0
CounterMutationStage	0	0	0	0	0
MigrationStage	0	0	65	0	0
MemtablePostFlush	1	1	231	0	0
PerDiskMemtableFlushWriter_0	1	1	210	0	0
ValidationExecutor	0	0	0	0	0
Sampler	0	0	0	0	0
MemtableFlushWriter	1	1	210	0	0
InternalResponseStage	0	0	2817415	0	0
ViewMutationStage	0	0	0	0	0
AntiEntropyStage	0	0	0	0	0
CacheCleanupExecutor	0	0	0	0	0

Dropped Messages

instaclustr

- Nodetool tpstats – message statistics
  - Second part of this output shows dropped messages since last Cassandra restart.
  - Dropped messages indicates a problem and possibly repair required.

Message type	**Dropped**
RANGE_SLICE	0
READ_REPAIR	23
PAGED_RANGE	0
BINARY	0
READ	**10434**
MUTATION	**4948**
_TRACE	0
REQUEST_RESPONSE	6
COUNTER_MUTATION	0

Cassandra logs

instaclustr

What indicates a problem?

ERRORS:

```
[ReadStage:497] ERROR org.apache.cassandra.db.filter.SliceQueryFilter Scanned over 200000 tombstones in system.schema_columns; query aborted (see
tombstone_failure_threshold)
[FlushWriter:193] ERROR org.apache.cassandra.service.CassandraDaemon Exception in thread Thread[FlushWriter:193,5,RMI Runtime]
```

Large Partition warnings:

```
INFO [CompactionExecutor:37] 2017-04-02 22:09:42,075 CompactionController.java (line 196) Compacting large row
Keyspace/Table:sub-c-868487ce-a5ce-11e2-88ac-123130f22c9a!info-user-status-887!14884128000000000 (539806586 bytes) incrementally
```

Log GC Pauses:

```
Nov 04 12:34:44 [ScheduledTasks:1]  INFO org.apache.cassandra.service.GCInspector GC for ConcurrentMarkSweep: 13624 ms for 2 collections, 3206968456 used; max is
3858759680
```

Batch warnings:

```
Jul 30 03:58:06 UTC: WARN o.a.c.cql3.statements.BatchStatement Unlogged batch covering 30 partitions detected against table [data.cf]. You should use a logged batch
for atomicity, or asynchronous writes for performance.
```

Preventative Maintenance: Health Checks

instaclustr

Other things to monitor on a regular basis

- Disk usage on all nodes. Keep it under 70% to allow for compactions and data growth.
- Tombstones per read
- SSTables per read
- Large partitions
- Backup status. Make sure they are working!
- Repair Status

Managing Compactions

instaclustr

- Recall that compactions are ongoing and are an integral part of any healthy cluster.
- Can have a significant disk, memory (GC), cpu, IO overhead.
- Are often the cause of “unexplained” latency or IO issues in the cluster.
- Compactions need sufficient headroom to complete (at least the size of the largest SSTable included).
- Compactions can fall behind because of excessive write load or heap pressure. Heap pressure will cause frequent flushing of Memtables to disk.

Heap pressure => flushing memtables => many small SSTables => many compactions

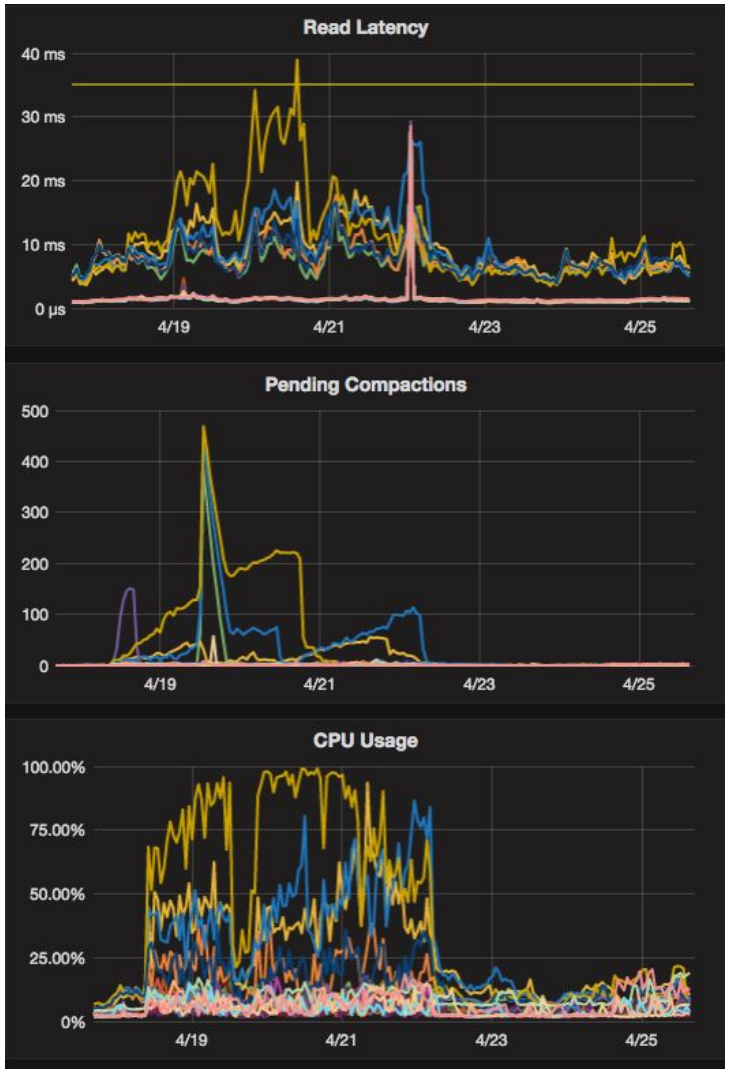
Monitoring Compactions

instaclustr

- Monitor with nodetool compactionstats

```
~ $ nodetool compactionstats -H
pending tasks: 518
   compaction type   keyspace   table   completed       total    unit   progress
        Compaction       data      cf    18.71 MB   111.16 MB   bytes     16.83%
Active compaction remaining time :   0h00m05s
```

- A single node doing compactions can cause latency issues **<u>across the whole cluster</u>**, as it will become slow to respond to queries.



Managing Compactions

instaclustr

What to do if compactions are causing issues?

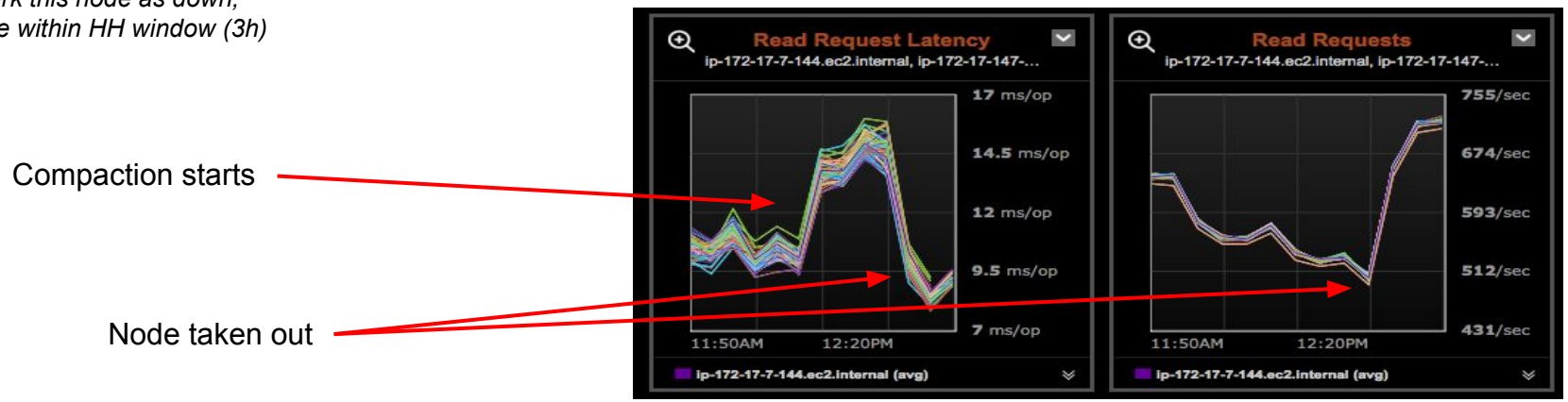
Throttle `nodetool setcompactionthroughput 16` ← *Set until C* is restarted. On 2.1 applies to NEW compactions, on 2.2.5+ applies instantly*

Stop and disable `nodetool stop COMPACTION` ← *Case is important! Stops currently active compactions only.*

If that doesn't help - take the node out of the cluster

```
nodetool disablebinary && nodetool disablegossip && nodetool disablethrift && nodetool setcompactionthroughput 0
```

Other nodes will mark this node as down,
So need to complete within HH window (3h)



Tip: Removing data

instaclustr

- **DELETE** - creates tombstones which will not be purged by compactions until after gc_grace_seconds
  - Default is 10 days, but you can ALTER it and it is effective immediately.
  - Make sure all nodes are UP before changing gc_grace.

- **TRUNCATE or DROP** – only creates a snapshot as a backup before removing all the data.
  - The disk space is released as soon as the snapshot is cleared
  - Preferred where possible.

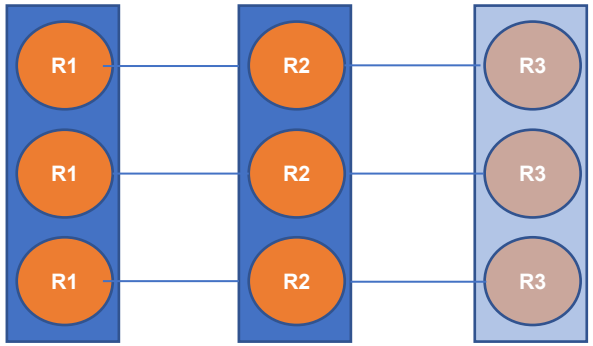
Topology for availability and maintenance

instaclustr

For production we recommend **3 nodes in 3 racks with RF3.**

- Use Cassandra logical racks and map to physical racks.
  - Ideally, make racks a multiple of RF ⟶ Each rack will contain a full copy of the data
  - Can survive the loss of a full rack without losing QUORUM (strong consistency)
  - Always NetworkTopologyStrategy. It's not just for multi-DC, but is also "rack aware".

```
ALTER KEYSPACE <keyspace> WITH replication = {'class': 'NetworkTopologyStrategy','DC1': '3'}
```



Getting this right upfront will make management of the cluster much easier in the future.

Topology for availability and maintenance

instaclustr

What does this topology mean for manageability?

1. Cluster maintenance operations (eg upgrades) can be done *by rack*, significantly cutting down the work and service interruption
2. You only need to run repair on *one rack* in order to repair the whole data set.

Cluster Mutations

instaclustr

Including:

- Adding and removing nodes
- Replacing dead nodes
- Adding new data centers

Ensure the cluster is 100% healthy and stable before making ANY changes.

Cluster Mutations

instaclustr

Ensure the cluster is 100% healthy and stable before making ANY changes.

Adding Nodes

instaclustr

- **How do you know when to add nodes?**
  - When disks are becoming >70% full.
  - When CPU/OS load is consistently high during peak times.
- **Tips for adding new nodes:**
  - If using logical racks, add one node to every rack (keep distribution even)
  - Add one node at a time.
  - During the joining process, the new node will stream data from the existing node.
  - A joining node will accept writes but not reads.
  - Unthrottle compactions on the JOINING node “nodetool setcompactionthroughput 0”
    - *But throttle again once node is joined.*
  - Monitor joining status with “nodetool netstats” (see next page)
  - After the node has streamed and joined it will have a backlog of compactions to get through.
  - Versions <2.2.x Cassandra will lose level info (LCS) during streaming and have to recompact *all* sstables again.

Replacing Nodes

instaclustr

- Replacing a dead node is similar to adding a new one, but add this line in the cassandra-env.sh *before* bootstrapping:

```
-Dcassandra.replace_address_first_boot=<dead_node_ip>
```

- This tells Cassandra to stream data from the other replicas.
  - *Note this can take quite a long time depending on data size*
  - *Monitor with nodetool netstats*
- If on >2.2.8 and replacing with a different IP address, the node will receive all the writes while joining.
- Otherwise, you should run repair.
  - If the replacement process takes longer than `max_hint_window_in_ms` you **should** run repair to make the replaced node consistent again, since it missed ongoing writes during bootstrapping (streaming).

Adding DCs

instaclustr

Why will you need to do this?

- Distributing workload across data centers or regions.
- Major topology changes.
- Cluster migrations.

A data center is a logical grouping of nodes.

Adding another DC

instaclustr

- Ensure all keyspaces are using **NetworkTopologyStrategy**
- All queries using `LOCAL_*` consistency. This ensures queries will not check for replicas in the new DC that will be empty until this process is complete.
- All client connections are restricted to connecting only to nodes in the original DC. Use a data center aware load balancing policy such as **DCAwareRoundRobinPolicy**.
- Bring up the new DC as a stand alone cluster.
    - Provision nodes and configure Cassandra:
    - `cluster_name` in yaml must be the SAME as the original DC.
    - DC name in `cassandra-rackdc.properties` must be UNIQUE in the cluster.
    - Include seed nodes from the other DC.
- Join the new DC to the old one:
    - Start cassandra
    - Change replication on keyspaces
    - Execute `nodetool rebuild <from existing dc>` on 1-3 nodes at a time.

Final tips

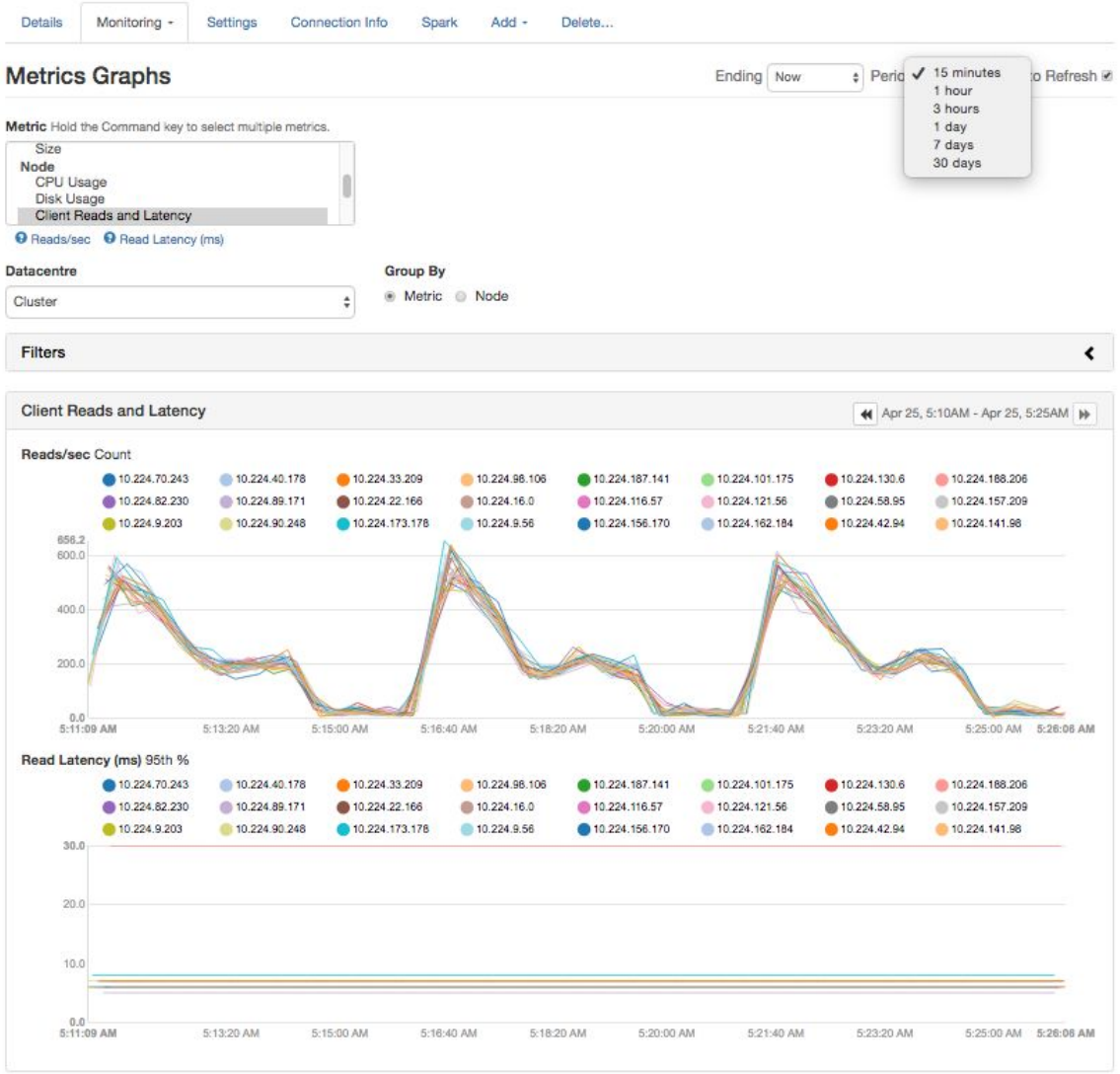
instaclustr

- When making major changes to the cluster (expanding, migrating, decommissioning), **GO SLOW.**
  - It takes longer to recover from errors than just doing it right the first time.
  - Things I've seen customers do:
    - Rebuild 16 nodes in a new DC concurrently
    - Decommission multiple nodes at once
    - Unthrottled data loads
- Don't overload your cluster. It is possible to get your cluster into a state from which you are unable to recover without significant downtime or data loss.
- Keep C* up to date, but not too up to date.
  - Currently investigating segfaults with MV in 3.7+
- Read the source code.
  - It is the most thorough and up to date documentation.

How Instaclustr can help

instaclustr

- Managed Service
  - Gives you a proven, best practice configured Cassandra cluster in < ½ hour
  - AWS, Azure, GCP and SoftLayer
  - Security configuration with tick of a check box
  - Cluster health page for automated best practice checks
  - Customer monitoring UI & ongoing monitoring & response by our Ops Team
- Consulting
  - Cluster health reviews
  - Data model & Cassandra application design assistance
- Enterprise Support
  - Support from our Managed Service tech-ops team where you run your own cluster



Resources

instaclustr

- Article by TLP explaining Tombstones

  http://thelastpickle.com/blog/2016/07/27/about-deletes-and-tombstones.html

- Instaclustr tech blog:
  https://www.instaclustr.com/blog

- Contact us at any time:
  support@instaclustr.com

instaclustr

Brooke Thorley
VP of Technical Operations
brooke@instaclustr.com