

DS210

DataStax Enterprise Operations with Apache Cassandra™

Introduction

Let's get to know each other

Tell us:

- Your name
- Where you are from
- Where you work
- Goals for using Cassandra
- Prior experience
- Personal trivia

What are we going to learn?

We'll explore:

- Configuring Clusters
- Cluster Performance Estimation and Monitoring
- Adding/Removing Nodes
- Care and Feeding of a Cassandra Cluster
- Backup, Restore and Moving Data
- JVM Tuning Options
- Security Considerations
- OpsCenter and LifeCycleManager

Configuring Clusters

YAML File

Configuring Clusters

Cassandra.yaml – the main configuration file

- Cassandra nodes read this file on start-up
 - Remember, restart the node for the changes to take effect!
- Located in the following directories:
 - Cassandra package installations: /etc/dse/cassandra
 - Cassandra tarball installations:
`<INSTALL_LOCATION>/resources/cassandra/conf`

Configuring Clusters

Cassandra.yaml quick start – minimal properties

- cluster_name (Default: “Test Cluster”)
- listen_address (Default: localhost)
- native_transport_address (Default: localhost)
- seeds (Default: "127.0.0.1")

Configuring Clusters

Cassandra.yaml – cluster_name

```
# Cassandra storage config YAML
# NOTE:
# See http://wiki.apache.org/cassandra/StorageConfiguration for
# full explanations of configuration directives
# /NOTE
# The name of the cluster. This is mainly used to
# prevent machines in one logical cluster from joining another.
cluster_name: 'Test Cluster'
```

Configuring Clusters

Cassandra.yaml – 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. Interfaces must correspond  
# to a single address, IP aliasing is not supported.  
#  
# 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 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.  
listen_address: localhost  
# listen_interface: eth0  
# listen_interface_prefer_ipv6: false
```

Configuring Clusters

Cassandra.yaml – native_transport_address

```
# The address or interface to bind the native transport server to.  
#  
# Set native_transport_address OR native_transport_interface, not both.  
#  
# Leaving native_transport_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 native_transport_broadcast_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.  
native_transport_address: localhost
```

Configuring Clusters

Cassandra.yaml – “seeds”

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

Configuring Clusters

Cassandra.yaml – example cluster settings

Setting	Node 1	Node 2	Node 3
IP Address (external)	54.67.60.144	18.144.66.96	13.57.37.68
cluster_name	KillrVideo	KillrVideo	KillrVideo
listen_address	172.31.15.137	172.31.12.88	172.31.12.167
native_transport_address	172.31.15.137	172.31.12.88	172.31.12.167
seeds	“172.31.15.137”	“172.31.15.137”	“172.31.15.137”

Commonly Used YAML Settings

Here are some of the commonly used configuration settings

- endpoint_snitch (Default: SimpleSnitch)
- initial_token (Default: disabled) num_token (Default: 128)
- commitlog_directory (Default: /var/lib/cassandra/commitlog)
- data_file_directories (Default: /var/lib/cassandra/data)
- hints_directory (Default: /var/lib/cassandra/hints)
- saved_caches_directory (Default: /var/lib/cassandra/saved_caches)

Notable YAML Settings

- hinted_handoff_enabled (Default: true)
- max_hint_window_in_ms (Default: 10800000 milliseconds [3 hours])
- row_cache_size_in_mb (Default: 0 - disabled)
- file_cache_size_in_mb (Default: 4096 - disabled)
- memtable_heap_space_in_mb/memtable_offheap_space_in_mb (Default: 2048 - disabled)

Exercise – Setting up your environment

Cluster Performance Estimation and Verification

Cluster Sizing

Cluster Sizing

This is an estimation process

- Estimates are only a rough-order of magnitude due to metadata
- Things to consider when estimating cluster size:
 - Throughput - How much data per second?
 - Growth Rate - How fast does capacity increase?
 - Latency - How quickly must the cluster respond?

Cluster Sizing

Throughput

- Measure throughput in data movement per time period (e.g. GB/S)
- Consider reading and writing separately
- A function of:
 - Operation generators (e.g. users)
 - Rate of operation generation (e.g. 3 clicks per minute)
 - Size of the operations (number of rows X row width)
 - Operation mix (read/write ratio)

Cluster Sizing

Example: Write Throughput from Posting Comments to KillrVideo

- 2 million users
- Each user comments 5 times/day
- Yields ~100 comments/second = $(2M * 5) / (24 * 60 * 60)$
- Each comment inserts 1 row
- Each row is ~1,000 bytes
- Result is writing 100KB/S = $100 * 1,000$

Cluster Sizing

Example: Read Throughput from Displaying Comments for KillrVideo

- 2 million users
- Viewing 10 video summaries/day
- Yields \sim 250 queries/second = $(2M * 10) / (24 * 60 * 60)$
- 4 comments per video = 4 rows/query
- \sim 1,000 bytes per comment
- Results in reading \sim 1MB/S = $250 * 4 * 1,000$

Cluster Sizing

Growth Rate

- How big must the cluster be just to hold the data?
- Given the write throughput, we can calculate growth
 - What is the new/update ratio?
 - What is the replication factor?
 - Additional headroom for operations

Cluster Sizing

Growth Rate Example

- Throughput assumptions:
 - 100KB/S write throughput
 - 20% write updates
 - Yields effective growth of 80KB/S = $100\text{KB/S} * (1 - .2)$

Cluster Sizing

Growth Rate Example (continued)

- Replication assumptions:
 - Replication factor of 6 (3 in each of 2 DCs)
 - Yields effective growth rate of $480\text{KB/S} = 80\text{KB/S} * 6$

Cluster Sizing

Growth Rate Example (continued)

- Headroom requirements for anti-entropy operations
- No more than 50% loading
- Yields an effective growth rate of $960\text{KB/S} = 480\text{KB/S} / .5$

Cluster Sizing

Growth Rate Example (continued)

- Node capacity assumption: 2TB max using SSD (1TB with HDD)
- Fills a node about every month = $(960K * 60 * 60 * 24 * 30) / 2T$

Cluster Sizing

Latency

- Calculating cluster capacity is not enough
- Understand your SLAs
 - In terms of latency
 - In terms of throughput

Cluster Sizing

Latency

- Relevant factors:
 - IO Rate
 - Workload shape
 - Access patterns
 - Table width
 - Node profile (i.e., cores, memory, storage, network)
- Improve estimates with benchmarking

Cluster Sizing

Future Capacity

- Validate your assumptions often
- Monitor for changes over time
- Plan for increasing cluster size before you need it
- Be ready to draw down if needed

Cassandra-stress

cassandra-stress

Stress-testing utility for benchmarking/load testing a cluster

- Simulates a user-defined workload
- Use the cassandra-stress to:
 - Determine schema performance
 - Understand how your database scales
 - Optimize your data model and settings
 - Determine production capacity
- **Try out your database *before* you go into production**

Cassandra-stress

Use YAML to configure cassandra-stress

- The YAML file lets you:
 - Define your schema
 - Specify any compaction strategy
 - Create a characteristic workload
 - Without writing a custom tool

Cassandra-stress

Use YAML to configure cassandra-stress

- The YAML file is split into a few sections:
 - Schema Description — defines the keyspace
 - Column Descriptions — outlines how to create the simulated data
 - Batch Descriptions — defines the data insertion pattern
 - Query Descriptions — defines the possible queries for test performs

Cassandra-stress

Schema description

- Defines the keyspace and table information
- If the schema is not yet defined the test will create it
- If the schema already exists, only defines the keyspace and table names

Cassandra-stress

Schema description example

```
# Keyspace Name
keyspace: stresscql

# The CQL for creating a keyspace (optional if it already exists)
keyspace_definition: |
    CREATE KEYSPACE stresscql WITH replication = {'class': 'SimpleStrategy',
'replication_factor': 1};

# Table name
table: blogposts

# The CQL for creating a table you wish to stress (optional if it already exists)
table_definition: |
    CREATE TABLE blogposts (
        domain text, published_date timeuuid, url text, author text, title text, body text,
        PRIMARY KEY(domain, published_date) )
        WITH CLUSTERING ORDER BY (published_date DESC) AND
        compaction = { 'class':'LeveledCompactionStrategy' } AND
        comment='A table to hold blog posts'
```

Cassandra-stress

Column descriptions

- Describes how to generate the data for each column
- The data values are meaningless, but simulate the patterns in terms of size and frequency
- Generated values follow a specified distribution such as Uniform, Exponential, Gaussian...
- Parameters include:
 - Data size — how many characters are in the data value
 - Value population — how often values re-occur
 - Cluster distribution — the number of values for the column appearing in a single partition (cluster columns only)

Cassandra-stress

Column descriptions

- possible distributions are:
 - EXP(min..max) — An exponential distribution over the range [min..max]
 - EXTREME(min..max,shape) — An extreme value (Weibull) distribution over the range [min..max]
 - GAUSSIAN(min..max,stdvrng) — A gaussian/normal distribution, where mean=(min+max)/2, and stdev is (mean-min)/stdvrng
 - GAUSSIAN(min..max,mean,stdev) — A gaussian/normal distribution, with explicitly defined mean and stdev
 - UNIFORM(min..max) — A uniform distribution over the range [min, max]
 - FIXED(val) — A fixed distribution, always returning the same value

Cassandra-stress

Column description example

columnspec:

```
- name: domain
  size: gaussian(5..100)          #domain names are relatively short
  population: uniform(1..10M)      #10M possible domains to pick from

- name: published_date
  cluster: fixed(1000)           #under each domain we have max 1000 posts (1000 rows in the partition)

- name: url
  size: uniform(30..300)

- name: title                  #titles shouldn't go beyond 200 chars
  size: gaussian(10..200)

- name: author
  size: uniform(5..20)           #author names should be short

- name: body
  size: gaussian(100..5000)       #the body of the blog post can be long
```

Cassandra-stress

Batch descriptions

- Specifies how the test inserts data
- For each insert operation, specifies the following distributions/ratios:
 - Partition distribution — Number of partitions to update per batch (default FIXED(1))
 - Select distribution ratio — portion of rows from a partition included in particular batch (default FIXED(1)/1)
 - Batch type — The type of CQL batch to use. Either LOGGED/UNLOGGED (default LOGGED)

Cassandra-stress

Batch description example

insert:

```
partitions: fixed(1)      # Partition key is the domain - insert 1/batch
```

```
select: fixed(1)/1000    # 1000 posts/domain so 1/1000 allows 1 post/batch
```

```
batchtype: UNLOGGED # Unlogged batches
```

Cassandra-stress

Query descriptions

- You can specify any CQL query on the table by naming them under the queries section
- *fields* specifies if the bind variables should be from the same row or across all rows in the partition

Cassandra-stress

Query description example

queries:

singlepost:

cql: select * from blogposts where domain = ? LIMIT 1

fields: samerow

timeline:

cql: select url, title, published_date from blogposts where domain = ? LIMIT 10

fields: samerow

Cassandra-stress

Test Inserts Example

```
cassandra-stress user profile=blogpost.yaml ops\insert=1\)
```

- Without any other options, stress will run our inserts starting with 4 threads and increasing them until it reaches a limit
- All inserts are done with the native transport (CQL as opposed to Thrift) and prepared statements

Cassandra-stress

Test Queries Example

```
cassandra-stress user profile=blogpost.yaml ops\(singlepost=1\)
```

```
cassandra-stress user profile=blogpost.yaml ops\.timeline=1\)
```

Cassandra-stress

Mixed Inserts and Queries Example

```
cassandra-stress user profile=blogpost.yaml ops\(singlepost=2,timeline=1,insert=1\)
```

- We can also run many types of queries and inserts at once
- This syntax sends three queries for every one insert

Exercise - Configure and Run Cassandra Stress

Cluster Performance Monitoring

Linux Tools for Performance Analysis

Linux top Command

Linux Tools for Performance Analysis

Command: top

- The first tool linux sys-admins reach for
 - Gives a brief summary of the system resources
 - Also shows resource utilization and info on a per-process basis
- Use top to make sure the Cassandra process is the main consumer of resources

Linux Tools for Performance Analysis

Command: top Example

Summary

```
top - 21:50:11 up 2 days, 1:00, 2 users, load average: 0.00, 0.01, 0.34
Tasks: 77 total, 2 running, 75 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.7 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st
KiB Mem: 3838800 total, 3282784 used, 556016 free, 118144 buffers
KiB Swap: 0 total, 0 used, 0 free. 1413044 cached Mem
```

Per Process

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7719	root	20	0	413216	2172	1348	S	0.7	0.1	4:52.92	collectd
11800	root	20	0	3074524	1.438g	58340	S	0.7	39.3	93:52.83	java
27	root	20	0	0	0	0	S	0.3	0.0	0:11.36	kworker/0:1
1	root	20	0	33364	2684	1492	S	0.0	0.1	0:01.51	init

Linux Tools for Performance Analysis

Command: top Example – Summary Section (Line 1 - top)

Current

How long the system has been running

Average number of users

CPU average load
1 minute, 5 minutes, 15 minutes

```
top - 21:50:11 up 2 days, 1:00, 2 users, load average: 0.00, 0.01, 0.34
Tasks: 77 total, 2 running, 75 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.7 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st
KiB Mem: 3838800 total, 3282784 used, 556016 free, 118144 buffers
KiB Swap: 0 total, 0 used, 0 free. 1413044 cached Mem
```

Linux Tools for Performance Analysis

Command: top Example – Summary Section (Line 2 - Tasks)

Total number of processes	Count of Running Processes	Count of Sleeping Processes	Count of Stopped Processes	Count of Zombie Processes
top - 21:30:11 up 2 days, 1:08 2 users, 1 load average: 0.00, 0.01, 0.34	Tasks: 77 total, 2 running, 75 sleeping, 0 stopped, 0 zombie	%Cpu(s): 0.0 us, 0.7 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st	KiB Mem: 3838800 total, 3282784 used, 556016 free, 118144 buffers	KiB Swap: 0 total, 0 used, 0 free. 1413044 cached Mem

Linux Tools for Performance Analysis

Command: top Example – Summary Section (Line 3 - % Cpu)

User CPU U	System Spac CPU Utilizatio	Low Priorit CPU Ut	Idle Perce	CPU Wa	Software Interrupt Percentage CPU Utilization	PU
top - 21:51 11 up, 11 days, 00:00, 2 users, load average: 0.00, 1, 0.4	Tasks: 77 total, 2 running, 75 sleeping, 0 stopped, 0 zombie	%Cpu(s): 0.0 us, 0.7 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st	KiB Mem: 3838800 total, 3282784 used, 556016 free, 118144 buffers	KiB Swap: 0 total, 0 used, 0 free. 1413044 cached Mem		

Linux Tools for Performance Analysis

Command: top Example – Summary Section (Line 4 – KiB Mem)

Total Memory (KB)	Count of Used Memory(KB)	Amount of Unallocated Memory (KB)	Amount of Memory the System Uses for File Metadata (KB)
top - 21:50: up 2 days, 00, 2 users, load average: 0.0 .01, 0.34 Tasks: 77 total, 2 running, 5 sleeping, 0 stopped, 0 zombie %Cpu(s): 0.0 us, 0.7 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st KiB Mem: 3838800 total, 3282784 used, 556016 free, 118144 buffers KiB Swap: 0 total, 0 used, 0 free. 1413044 cached Mem			

Linux Tools for Performance Analysis

Command: top Example – Summary Section (Line 5 – KiB swap)

Total Alloc Swap Space	Used Swap Space (KB)	Unused Swap Space (KB)	System Space Used for Caching File Data (KB)
top - 21:50 up 2 days, 1:02, 2 users, load average: 0.00, 0.01, 0.34 Tasks: 77 total, 2 running, 0 sleeping, 0 stopped, 0 zombie %Cpu(s): 0.0 us, 0.7 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st KiB Mem: 3888800 total, 3282184 used, 556016 free, 118144 buffers KiB Swap: 0 total, 0 used, 0 free. 1413044 cached Mem			

Linux Tools for Performance Analysis

Command: top Example – Per Process Section

Process ID	Priority	Amount of Memory	Utilization	Length of Time Process has been running	The Process Command
PID	USER PR NI	VIRT RES SHR S %CPU %MEM TIME+	COMMAND		
7719	root 20 0	413216 2172 1348 S 0.7 0.1	4:52.92	collectd	
11800	root 20 0	3074524 1.438g 58340 S 0.7 39.3	93:52.83	java	
27	root 20 0	0 0 0 S 0.3 0.0	0:11.36	kworker/0:1	
1	root 20 0	33364 2684 1492 S 0.0 0.1	0:01.51	init	

Linux dstat Command

Linux Tools for Performance Analysis

Command: dstat

- Shows all system resources in a single table
- One-stop tool includes stats like:
 - iostat
 - vmstat
 - ifstat
- Configurable output (see man page)
- Default shows CPU, Disk IO, Network, page and system stats

Linux Tools for Performance Analysis

Command: dstat – Example Usage

```
dstat -am
```

- The -a option gives all the default stats
- So -am gives all the defaults plus memory stats

Linux Tools for Performance Analysis

Command: dstat – Reading the Output

-a Option

-m Option

```
dstat -am
----total-cpu-usage---- -dsk/total- -net/total- -paging- --system-- ----memory-usage-----
usr sys idl wai hiq siq| read writ| recv send| in out | int csw | used buff cach free
 74 25 0 0 0 1| 0 0 | 456k 102k| 0 0 | 1819 5742 |1712M 114M 1631M 291M
 92 7 0 0 0 1| 0 0 | 97k 27k| 0 0 | 753 2377 |1712M 114M 1632M 291M
 97 3 0 0 0 0| 0 0 | 12k 10k| 0 0 | 272 278 |1712M 114M 1633M 290M
 76 24 0 0 0 0| 0 2008k| 414k 84k| 0 0 | 1931 7739 |1713M 114M 1633M 289M
```

Linux Tools for Performance Analysis

Command: dstat – CPU Stats

Percent of Time Processing Software Interrupts

Percent of Time Processing Hardware Interrupts

Percent of Time CPU Bound

Percent of Time CPU Idle

Percent of Time CPU Waiting for I/O

Percent of Time CPU System (idle)

Percent of Time CPU User

--total--cpu--usage--						
	usr	sys	idl	wai	hiq	siq
	74	25	0	0	0	1
	92	7	0	0	0	1
	97	3	0	0	0	0
	76	24	0	0	0	0

Linux Tools for Performance Analysis

Command: dstat – Disk Stats

The Amount of
Data Read

```
-dsk/total-
    read  writ
    0      0
    0      0
    0      0
    0  2008k
```

The Amount of
Data Written

- To get a feel for disk utilization, know disk transfer rates
 - HDDs can transfer tens of MBS
 - SSDs can transfer hundreds of MBS

Linux Tools for Performance Analysis

Command: dstat – Network Stats

The Amount of Network Data Received

```
-net/total-
recv  send
456k 102k
 97k  27k
 12k  10k
414k 84k
```

The Amount of Network Data Sent

- Network speeds vary, but Gigabit ethernet is about 100 MBS

Linux Tools for Performance Analysis

Command: dstat – Paging and System Stats

Amount of Memory
being Paged

The Number of
Interrupts Service

The Number of
Context Switches

paging-		--system--	
in	out	int	csw
0	0	1819	5742
0	0	753	2377
0	0	272	278
0	0	1931	7739

If the paging stats
are not zero, you
have a memory
problem that you
need to investigate

System stats can
be an indication of
process
contention

Linux Tools for Performance Analysis

Command: dstat – CPU Stats

The Amount of Memory by the OS for File Memory

The Amount of Memory by the OS for File Data

The Amount of Unallocated Memory

```
-----memory usage-----  
used   buff   cach   free  
1712M  114M  1631M  291M  
1712M  114M  1632M  291M  
1712M  114M  1633M  290M  
1713M  114M  1633M  289M
```

nodetool for Performance Analysis

nodetool for Performance Analysis

nodetool is Cassandra's Swiss Army Knife

- Consider these sub-commands:
 - info
 - compactionhistory
 - gcstats
 - gossipinfo
 - ring
 - tablestats
 - tablehistograms
 - tpstats

nodetool for Performance Analysis

nodetool info

- Node-specific info
- Output is self-explanatory
- Performance points of interest include:
 - Load - amount of space used for SSTables
 - Heap Memory/Off Heap Memory - amount of RAM used by Cassandra for memtables, etc.
 - Key Cache - check out the hit rate

nodetool for Performance Analysis

nodetool info – Example Output (first 11 rows)

```
ID : 57679f16-da27-4804-95bb-76c6bd36da72
Gossip active : true
Native Transport active: true
Load : 48.98 MB
Generation No : 1482181943
Uptime (seconds) : 245146
Heap Memory (MB) : 594.93 / 1024.00
Off Heap Memory (MB) : 1.15
Data Center : Cassandra
Rack : rack1
Exceptions : 0
```

nodetool for Performance Analysis

nodetool info – Example Output (last four rows)

Key Cache : entries 148994, size 12.23 MB, capacity 51 MB, 2080730 hits, 2541295 requests, 0.819 recent hit rate, 14400 save period in seconds

Row Cache : entries 0, size 0 bytes, capacity 0 bytes, 0 hits, 0 requests, NaN recent hit rate, 0 save period in seconds

Counter Cache : entries 0, size 0 bytes, capacity 25 MB, 0 hits, 0 requests, NaN recent hit rate, 7200 save period in seconds

Chunk Cache : entries 1796, size 49.5 MiB, capacity 2.38 GiB, 2318 misses, 4522462 requests, 0.999 recent hit rate, 203.887 microseconds miss latency

Percent Repaired : 100.0%

Token : (invoke with -T/--tokens to see all 8 tokens)

nodetool for Performance Analysis

nodetool compactionhistory

- Node-specific compaction history
- Lists and describes the compaction actions that have taken place
- The columns explain
 - What was compacted (first 3 columns)
 - When the compaction took place
 - The input and output size of the SSTables
 - The number of tables/rows compacted into a new SSTable: {# tables: # rows}

nodetool for Performance Analysis

nodetool compactionhistory – Example Output (first three columns)

Compaction History:

Id	keyspace_name	columnfamily_name
716cecc0-c76f-11e6-8610-09423bb027f3	system	size_estimates
aa88e810-c693-11e6-8610-09423bb027f3	system	size_estimates
d8ca3a70-c63f-11e6-8610-09423bb027f3	system	local
...		

nodetool for Performance Analysis

nodetool compactionhistory – Example Output (last four columns)

Compaction History:

compacted_at	bytes_in	bytes_out	rows_merged
2018-01-08T16:37:06.437	3530	553	{4:8}
2018-01-08T15:36:53.259	3546	533	{4:8}
2018-01-08T10:37:06.435	3192	2898	{4:1}
...			

Other nodetool Functionality

nodetool for Performance Analysis

nodetool gcstats

- Java's garbage collection can impact performance - gcstats tells about the impact
- Node-specific info on the JVM garbage collection
- Measurements accumulate since previous invocation of gcstats
- Reports:
 - Length of time for measurements
 - Amount of time spent garbage collecting
 - Amount of memory collected
 - Number of collection sessions

nodetool for Performance Analysis

nodetool gcstats

Example output (first 4 columns)

Interval (ms)	Max GC Elapsed (ms)	Total GC Elapsed (ms)	Stdev GC Elapsed (ms)
207123	289	1059	14

Example output (last 3 columns)

GC Reclaimed (MB)	Collections	Direct Memory Bytes
1887436296	4	-1

nodetool for Performance Analysis

nodetool gossipinfo

- Cluster-level info
- Shows the current state of gossip entries according to the target node
- Entries have:
 - Label for the entry
 - Version number (except generation and heartbeat)
 - The value for the entry

nodetool for Performance Analysis

nodetool gossipinfo – Example Output

```
/172.31.5.229
generation:1516134195
heartbeat:184181
STATUS:66:NORMAL,-1354985388059245132
LOAD:184141:3.124671E7
SCHEMA:96153:c45cfa42-744d-32ba-8b67-4d689cf94590
DC:77:dc1
RACK:17:rack1
RELEASE_VERSION:4:4.0.0.1988
INTERNAL_IP:13:172.31.5.229
NATIVE_TRANSPORT_ADDRESS:3:172.31.5.229
X_11_PADDING:92121>{"dse_version":"6.0.0","workloads":"Cassandra","workload":"Cassandra","active":true,"server_id":"06-EA-6A-17-E1-7E","graph":false,"health":0.9}
NET_VERSION:1:256
...
...
```

nodetool for Performance Analysis

nodetool ring

- Shows info about tokens in the ring
- Cluster-level, table-level from the perspective of the queried node
- Use to determine the balance of tokens in the cluster
- Compare the load column to determine the balance

nodetool for Performance Analysis

nodetool ring – Example Execution

```
root@node0:~# node/bin/nodetool ring -- killr_video
```

Datacenter: Cassandra

=====

Address	Rack	Status	State	Load	Owns	Token
172.31.11.213	rack1	Up	Normal	51.21 MB	100.00%	9154262618015766378
172.31.11.213	rack1	Up	Normal	51.21 MB	100.00%	-8848252007366263074
172.31.11.213	rack1	Up	Normal	51.21 MB	100.00%	-8838326952028015898
...						

nodetool for Performance Analysis

nodetool tablestats

- Shows info about keyspaces/tables flushed to disk
- Node/Keyspace-level info
- Use the -H option to get human-readable output
- Use this command to get a sense of resources used by tables

nodetool for Performance Analysis

nodetool tablestats – Example Execution

```
root@node0:~# node/bin/nodetool tablestats -H -- killr_video
Total number of tables: 46
```

```
-----
Keyspace: killr_video
    Read Count: 1519021
    Read Latency: 1.7196091792016042 ms.
    Write Count: 540301
    Write Latency: 0.35517111017747516 ms.
    Pending Flushes: 0
        Table: user_by_email
        SSTable count: 2
        Space used (live): 29.38 MB
        Space used (total): 29.38 MB
        Space used by snapshots (total): 0 bytes
        Off heap memory used (total): 476.69 KB
```

...
(Much more output - not shown on this slide)

nodetool for Performance Analysis

nodetool tablehistograms

- Provides statistics about a specific table
- Table-focused command (must supply keyspace and table)
- Use this command to understand table latencies

nodetool for Performance Analysis

nodetool tablehistograms – Example Execution

```
root@node0:~# /root/node/bin/nodetool tablehistograms killr_video user_by_email
killr_video/user_by_email histograms
Percentile    SSTables    Write Latency    Read Latency    Partition Size    Cell Count
                                         (micros)          (micros)          (bytes)
50%           0.00           17.08           219.34            86                2
75%           0.00           17.08           219.34            103               2
95%           0.00           20.50          20924.30            103               2
98%           0.00           29.52          223875.79            103               2
99%           0.00           88.15          464228.84            103               2
Min           0.00           8.24            17.09              51                2
Max           0.00          1155149.91         3449259.15            124               2
```

nodetool for Performance Analysis

nodetool tpstats

- Gives info about thread pools
- Thread pools handle requests for various activities
 - Each row represents a class of activities
- Accepted requests can be active, pending or completed
- If the queue is full, the request is blocked
 - blocked requests indicate a resource problem
- Requests can time out - these messages are dropped

nodetool for Performance Analysis

nodetool tpstats

- An explanation of pool request states:
 - Active - the pool is working on the request
 - Pending - the request is queued for the pool
 - Completed - the request is finished
 - Blocked - the queue was full so the request was rejected

nodetool for Performance Analysis

nodetool tpstats – Example Output

Pool Name	Active	Pending	Completed	Blocked	All time blocked
CompactionExecutor	9	0	13352418	0	0
MemtableFlushWriter	0	0	0	0	0
MemtablePostFlush	0	0	8754502	0	0
TPC/all/EXECUTE_STATEMENT	0	N/A	15	N/A	N/A
... (more stats not shown on this slide)					

Message type	Dropped	Latency waiting in queue (micros)			
		50%	95%	99%	Max
RANGE_SLICE	0	N/A	N/A	N/A	N/A
SNAPSHOT	0	N/A	N/A	N/A	N/A
HINT	0	N/A	N/A	N/A	N/A
... (more stats not shown on this slide)					

System & Output Logs

System and Output Logs

Sometimes nodetool is not enough...

- For investigating in-depth problems, use logging
 - Cassandra's system.log for monitoring/diagnosing node behavior
 - Java's GC logging for tracking down garbage collection related issues

System and Output Logs

Cassandra's Logging

- By default, the log file is in `/var/log/cassandra/system.log`
- Also check `debug.log` in the same directory
- `system.log` logs INFO messages and above
- `debug.log` logs all messages
- Change the location by adding the following line to `/etc/dse/cassandra/jvm.options`:
`-Dcassandra.logdir=<LOG_DIRECTORY_Goes_HERE>`

System and Output Logs

Cassandra's Logging

- There are 7 logging levels:
 - OFF
 - ERROR
 - WARN
 - INFO (Default)
 - DEBUG
 - TRACE
 - ALL

System and Output Logs

Cassandra's Logging

- More is not always better - too much logging may NOT be helpful
 - Increased logging affects system performance
 - Lots of log entries may be overwhelming to understand
- Keep logging manageable; different levels for different packages/classes

System and Output Logs

Cassandra's Logging

- Two ways to configure logging:
 - Using the configuration file `logback.xml`
 - Using `nodetool setlogginglevel` (only sets logging levels temporarily – until restart)

System and Output Logs

Cassandra's Logging

- Logging configuration files
 - In the same directory as `cassandra.yaml`
 - Looks for `logback.xml`
 - Restart node for changes to the file to take affect

System and Output Logs

Cassandra's Logging - Configure log mgmnt (rolling) output pattern

Example logback.xml:

```
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${cassandra.logdir}/system.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <fileNamePattern>${cassandra.logdir}/system.log.%i.zip</fileNamePattern>
        <minIndex>1</minIndex>
        <maxIndex>20</maxIndex>
    </rollingPolicy>

    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <maxFileSize>20MB</maxFileSize>
    </triggeringPolicy>
    <encoder>
        <pattern>%-5level [%thread] %date{ISO8601} %F:%L - %msg%n</pattern>
    </encoder>
</appender>
...
```

System and Output Logs

Cassandra's Logging - Enable logging levels for packages/classes

Example logback.xml:

```
...
<!--Enable for debugging of Lease problems -->
<logger name="com.datastax.bdp.leasemanager" level="TRACE"/>

<!--Enable for debugging of StorageProxy problems -->
<logger name="org.apache.cassandra.service.StorageProxy" level="DEBUG"/>
...
```

System and Output Logs

Cassandra's Logging

- `nodetool setlogginglevel` talks directly to the process
 - No need to restart the node for changes to take affect
 - Changes will not persist if node is restarted

Example:

```
nodetool setlogginglevel org.apache.cassandra.service.StorageProxy DEBUG
```

System and Output Logs

Cassandra's Logging – Example: get current logging levels

```
root@node0:~# node/bin/nodetool getlogginglevels
```

Logger Name	Log Level
ROOT	INFO
DroppedAuditEventLogger	INFO
SLF4JAuditWriter	INFO
SolrValidationErrorHandler	ERROR
com.cryptsoft	OFF
org.apache.cassandra	DEBUG
...	

JVM Garbage Collection Logging

System and Output Logs

JVM Garbage Collection Logging

- GC logging helps you know exactly what the JVM is doing
- Find out:
 - When GC occurred
 - How much memory GC reclaimed
 - How much memory is available in the heap

System and Output Logs

JVM Garbage Collection Logging

- Turn on GC logging
 - Statically by editing `jvm.options`
 - Dynamically by using `jinfo`
- In either case, edit `/etc/dse/cassandra/jvm.options` to specify the GC log file

```
-Xloggc:<LOG_FILE_NAME_Goes_Here>"
```

System and Output Logs

JVM Garbage Collection Logging

- For static configuration:
- Edit /etc/dse/cassandra/jvm.options
- You can use these options:
 - -XX:+PrintGC - Simple, prints a line for every GC and every full GC
 - -XX:+PrintGCDetails - Detailed, young generation as well as old and perm gen
 - -XX:+PrintGCTimeStamps - Adds time to a simple or detailed GC log
 - -XX:+PrintGCDateStamps - Adds date information to a simple or detailed GC log

System and Output Logs

JVM Garbage Collection Logging

- For dynamic configuration, use `jinfo`

```
jinfo -flag +PrintGC <NODE_PROCESS_ID_Goes_HERE>
```

```
jinfo -flag +PrintGCTimeStamps <NODE_PROCESS_ID_Goes_HERE>
```

```
jinfo -flag +PrintGCDateStamps <NODE_PROCESS_ID_Goes_HERE>
```

System and Output Logs

JVM Garbage Collection Logging

Example output:

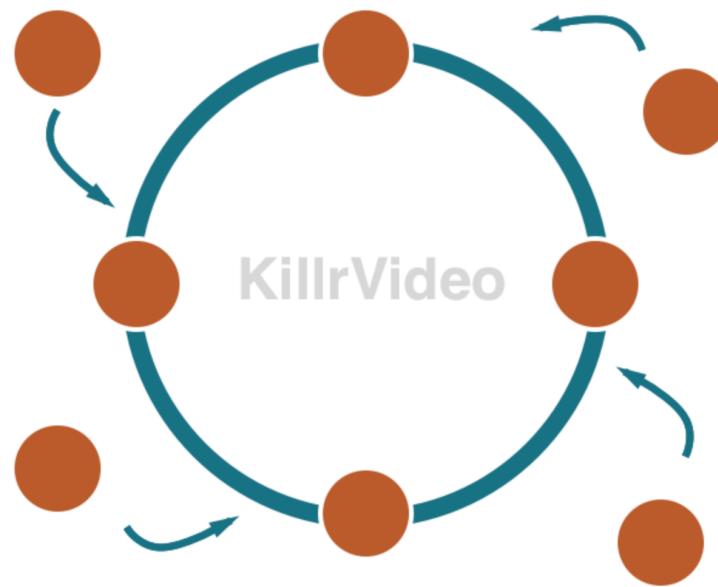
```
8109.128: [GC [PSYoungGen: 109884K->14201K(139904K)]  
691015K->595332K(1119040K), 0.0454530 secs]
```

Exercise - Monitor, Measure and Diagnose Cluster Performance

Adding/Removing Nodes

Adding Nodes

Why would we want to add more nodes to our cluster?



Adding Nodes

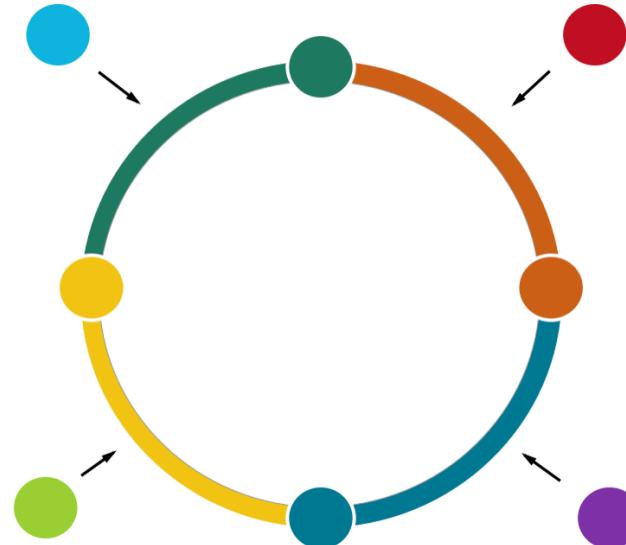
You might want to consider adding a new node if you have:

- Reached data capacity problem
 - Your data has outgrown the node's hardware capacity
- Reached traffic capacity
 - Your application needs more rapid response with less latency
- To increase operational headroom
 - Need more resources for node repair, compaction, and other resource intensive operations

Adding Nodes: Best Practices

Single-token nodes

- Double the size of your cluster (single-token nodes)



Adding Nodes: Best Practices

VNodes

- With VNode clusters, we can add nodes incrementally



Adding Nodes: Best Practices

- Adding a single node at a time will:
 - Result in more data movement
 - Will have a gradual impact on cluster performance
 - Will take longer to grow cluster
- Adding multiple nodes at the same time:
 - Is possible
 - Use extreme caution
 - Only attempt with deep understanding of DSE internals (i.e., support)

Bootstrapping

Bootstrapping

Bootstrapping nodes

- Bootstrapping is the process of a new node joining a cluster:
 - The joining node contacts a seed node
 - The seed node communicates cluster info, including token ranges, to the joining node
 - Cluster nodes prepare to stream necessary SSTables
 - Cluster nodes stream SSTables to the joining node (can be time consuming)
 - Existing cluster nodes continue to satisfy writes, but also forward write to joining node
 - When streaming is complete, joining node changes to normal state and handles read/write requests

Bootstrapping

Bootstrapping nodes

- To bootstrap a node:
 - Set up the node's configuration files (`cassandra.yaml`, etc.)
 - Four main parameters:
 - `cluster_name`
 - `native_transport_address`
 - `listen_address`
 - - seeds
 - Start up the node normally

Bootstrapping

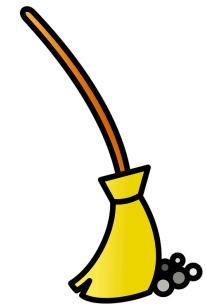
When bootstrapping fails: two scenarios:

- Bootstrapping node could not connect to cluster
 - Examine the log file to understand what's going on
 - Change config and try again
- Streaming portion fails
 - Node exists in cluster in joining state
 - First, try restarting the node
 - If restarting fails, try deleting data directories and rebooting
 - Or, worst case, remove the node from the cluster and try again

Bootstrapping

nodetool cleanup

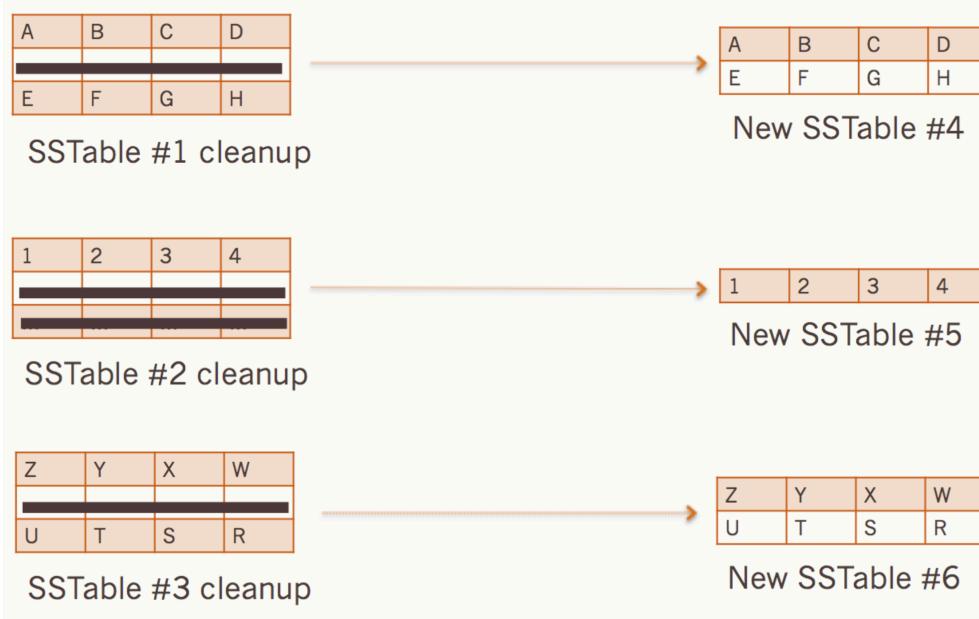
- Perform cleanup after a bootstrap on the OTHER nodes
- Reads all SSTables to make sure there is no token out of range for that particular node
- If the SSTable is not out of range, cleanup just does a copy
- There are options for running these operations in parallel. See the docs for details.



Bootstrapping

nodetool cleanup – What does it look like?

Cleanup is basically a compaction!



Bootstrapping

nodetool cleanup – How to run it

```
bin/nodetool [options] cleanup -- <keyspace> (<table>)
```

- The nodetool cleanup command cleans up all data in a keyspace and table(s) that are specified
- Use flags to specify:
 - -h [host] | [IP address]
 - -p port
 - -pw password
 - -u username
 - -j <sstable count>
- nodetool cleanup command will clean all keyspaces if no keyspace is specified

Removing a Node

Removing a Node from the Cluster

Two very different scenarios:

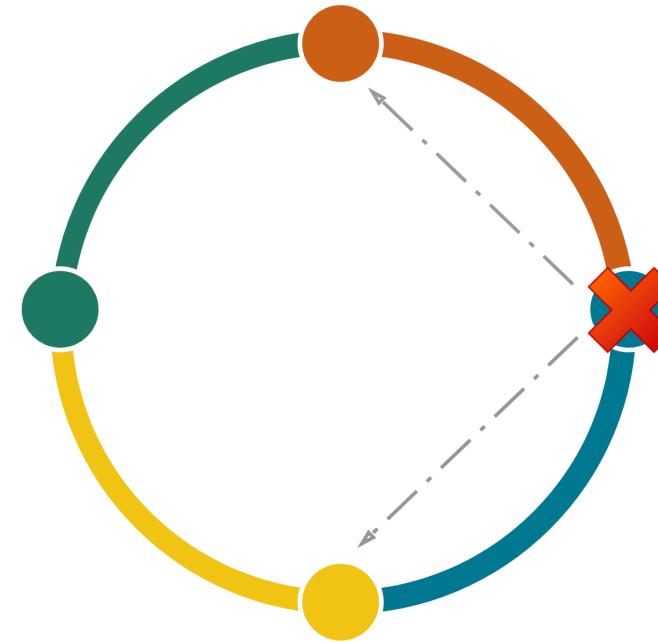
- You're going to reduce capacity, need to decommission (due to some sort of operational requirement)
- The node is offline and will never come back online



Removing a Node from the Cluster

What has to happen to remove a node?

- Other nodes need to pickup the removed-node's data
- The cluster needs to know the node is gone



Removing a Node from the Cluster

Three options for dealing with the data

- Redistribute the data from the node that is going away
 - nodetool decommission
- Redistribute the data from replicas
 - nodetool removenode
- Don't redistribute the data, just make the node go away
 - nodetool assassinate

Removing a Node from the Cluster

Decommissioning a node

- Do this if you want to decrease the size of the cluster
- The node must still be active
- Decommissioning will transfer the data from the decommissioned node to other active nodes in the cluster
 - With VNodes, the rebalance happens automatically
 - With Single-token nodes, you will need to manually rebalance the token ranges on the remaining nodes

Removing a Node from the Cluster

Decommissioning a node

- After running the nodetool decommission command:
 - The node is offline
 - The JVM process is still running (use dse cassandra-stop to kill the process)
 - The data is not deleted from the decommissioned node
 - If you want to add the node back to the cluster, delete the data first!
 - Not deleting the data may cause data resurrection issues

Removing a Node from the Cluster

Decommissioning a node

```
nodetool [options] decommission
```

Flags:

- -h [host]||[IP address]
- -p port
- -pw password
- -u username

Monitor progress with nodetool netstats and nodetool status

Example: `nodetool -u cassandra -pw cassandra decommission`

Removing a Node from the Cluster

Removing a node

- Do this if the node is offline and never coming back
- Obviously, you run this command from a different node than the one you are removing
- `nodetool removenode` will:
 - Make the remaining nodes in the cluster aware that the node is gone
 - Copy data from online nodes to the appropriate replicas to satisfy the replication factor

Removing a Node from the Cluster

Removing a node

```
nodetool [options] removenode <host id>
```

Flags:

- -h [host] | [IP address]
- -p port
- -pw password
- -u username

Additional arguments:

- status
- force

Example: `nodetool -u cassandra -pw cassandra removenode 172.31.12.88`

Removing a Node from the Cluster

Assassinating a node

- Do this **as a last resort** if the node is offline and never coming back
- `nodetool assassinate` will:
 - Make the remaining nodes in the cluster aware that the node is gone
 - **NOT** copy any data
- You should use `nodetool repair` on the remaining nodes to fix the data replication

Removing a Node from the Cluster

Assassinating a node

```
nodetool [options] assassinate <ip_address>
```

Flags:

- -h [host] | [IP address]
- -p port
- -pw password
- -u username

Example: `nodetool -u cassandra -pw cassandra assassinate 172.31.12.88`

Replace Downed Nodes

Replace Downed Nodes

Benefits of replacing a downed node

- You don't have to move the data twice
- Backup for a node will work for a replaced node, because same tokens are used to bring replaced node into cluster
- Best option is to replace rather than remove and add



Replace Downed Nodes

Replacing a downed node using nodetool

```
root@node1:~# nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
| / State=Normal/Leaving/Joining/Moving
--  Address      Load    Tokens  Owns      Host ID
UN  172.17.0.3   804.63 KB  256      ?        e20bf7ef-3ac1-424f-8190-804a71dfe789  rack1
UN  172.17.0.2   817.57 KB  256      ?        66a0073e-94a5-4664-8430-749f6bccfb5e  rack1
DN  172.17.0.4   846.65 KB  256      ?        e40fcebd-718b-4ee2-bd76-08c77ea77a63  rack1
```

- Configure a new node for the cluster normally with one additional step:
- In `jvm.options` add a `replace_address` JVM option
- This option should use the IP address of the node you are replacing

Replace Downed Nodes

Setting replace_address

- In `jvm.options` add this line near the bottom:
`-Dcassandra.replace_address=<IP_ADDRESS_OF_DEAD_NODE>"`
- This line tells the new node to use the same tokens as the old node
- Once you have configured the node, start the node in the cluster
- Monitor the bootstrapping process using `nodetool netstats`

Replace Downed Nodes

What if the node was also a seed node?

- The process is similar except:
 - Make sure the old IP address (of the downed node) does not appear in seeds list in `cassandra.yaml`
 - Also, make sure the new IP address (the node replacing the downed node) is not in the seeds list in `cassandra.yaml`
 - Perform a rolling restart on all nodes so the nodes are aware of the changes to the seeds list
 - Start the replacement node using `replace_address` in the `jvm.options` file
 - Once the replacement node is fully up:
 - Remove `replace_address` from `jvm.options`
 - Add the replacement node's IP to the seeds lists in all the nodes' `cassandra.yaml`

Exercise – Add/Remove a Node to/from the Cluster

Care and Feeding of a Cassandra Cluster

Level Compaction

Leveled Compaction

The Algorithm

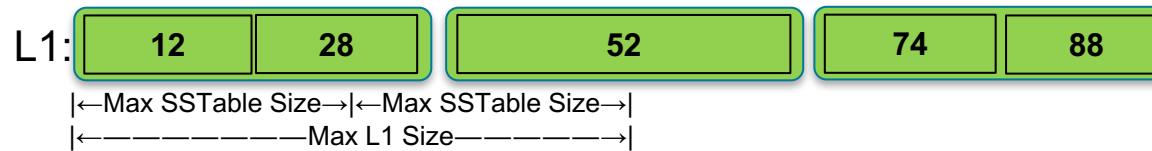


|←Max SSTable Size→|

Leveled Compaction

The Algorithm

L0:



Leveled Compaction

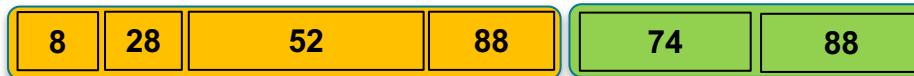
The Algorithm



Leveled Compaction

The Algorithm

L0:



L1:



L2:



Leveled Compaction

The Algorithm

L0:



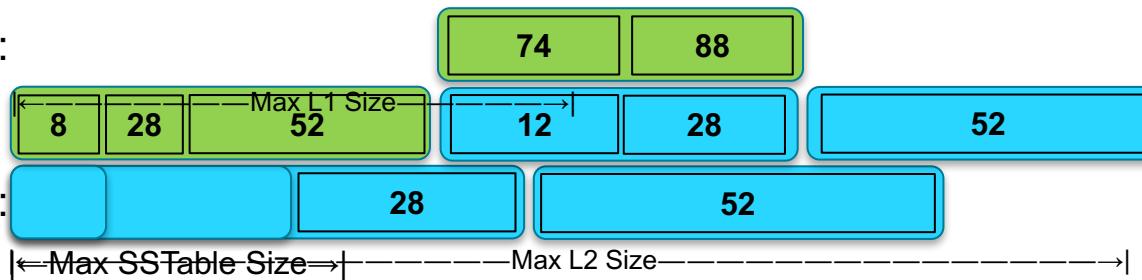
Leveled Compaction

The Algorithm

L0:

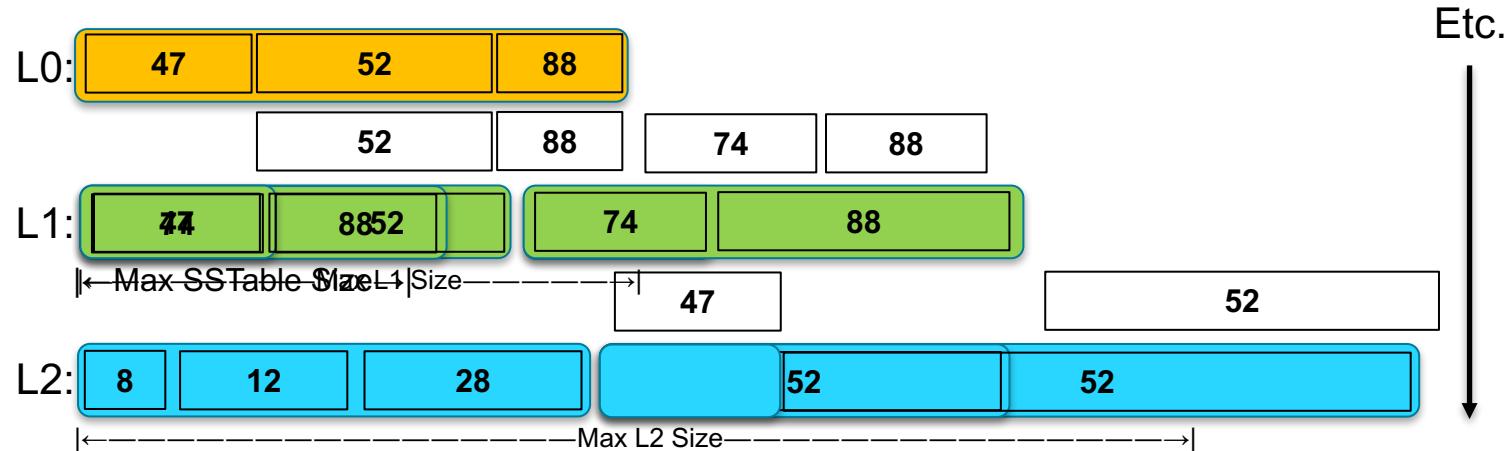
L1:

L2:



Leveled Compaction

The Algorithm



Leveled Compaction

Actual Implementation

- We used a multiplier of two for our example
- Leveled compaction uses a multiplier of 10 per level
- SSTable max size is 160MB (`sstable_size_in_mb`)
- SSTables exceed this amount to ensure the last partition written is complete

Leveled Compaction

Actual Implementation

- Our example data model had extremely large partitions
- The more granular your partitions, the closer to 160MB the SSTables will be
- Hence, more uniform

Leveled Compaction

Reads

- Leveled compaction is best for read-heavy workload
 - Occasional writes but high reads
- Each partition resides in only one SSTable per level (max)
- Generally reads handled by just a few SSTables
 - Partitions group together in a handful of levels as they compact down
 - 90% of the data resides in the lowest level (due to 10x rule)
 - Unless the lowest level is not yet full

Leveled Compaction

Reads

Example:

- L1: 1,600MB (1.6GB)
- L2: 16,000MB (16GB)
- L3: 160,000MB (160GB)
- L4: 1,600,000MB (1.6TB)
- $L1 + L2 + L3 = 177,600\text{GB}$
- $177,600\text{GB} / 1.6\text{TB} \approx 10\%$

Leveled Compaction

Disk Usage

- In general, an SSTable in one level overlaps 10(ish) SSTables in the level below
- Therefore, compaction requires 11x SSTable max size to compact
- One for the SSTable in the higher level
- 10 for the overlapped SSTables in the next level

Leveled Compaction

Disk Usage

- Leveled compaction wastes less disk space
- Obsolete records compact out quickly
 - A single partition's records group as they compact down
 - Updated records merge with older records due to this grouping

Leveled Compaction

Disadvantages

- IO intensive
- Compacts many more SSTables at once over size tiered compaction
- Compacts more frequently than size tiered
- Can't ingest data at high insert speeds

Leveled Compaction

Lagging Behind

- Leveled compaction switches to size-tiered compaction at level 0 when compaction is behind
- Creates larger L0 SSTables to compact with lower levels
- More optimal to compact a larger L0 SSTable to lower levels
- Reduces number of SSTables required for a read

Size Tiered Compaction

Size Tiered Compaction

Consider a Perfect World

HDD

100 mb 100 mb 100 mb 100 mb

400 mb



Size Tiered Compaction

Consider a Perfect World

HDD



400 mb



400 mb

Size Tiered Compaction

Consider a Perfect World

HDD

100 mb 100 mb 100 mb 100 mb

400 mb

400 mb



400 mb

400 mb

Size Tiered Compaction

Consider a Perfect World

HDD



400 mb

400 mb

1600 mb

400 mb

400 mb



compactor

Size Tiered Compaction

Consider a Perfect World

HDD

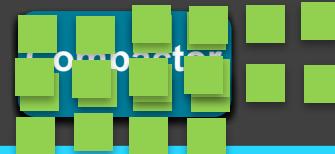


400 mb

400 mb

1600 mb

1600 mb

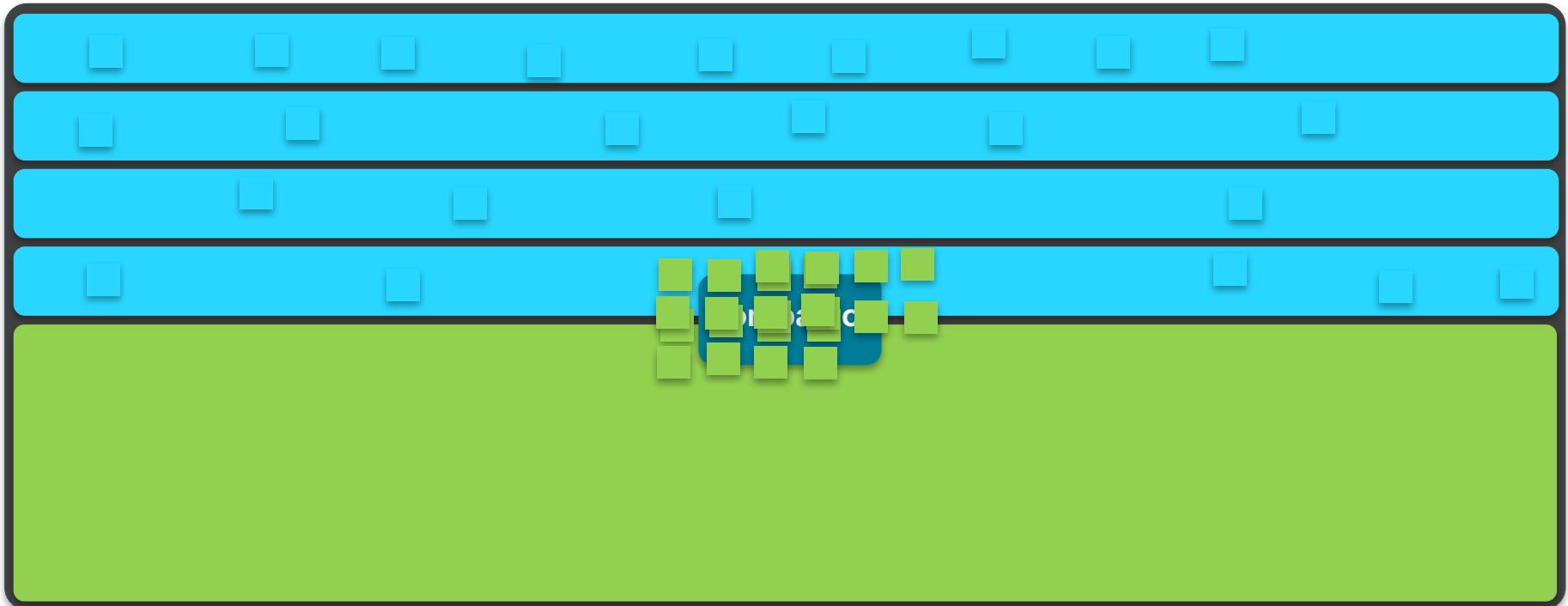


400 mb

400 mb

Size Tiered Compaction

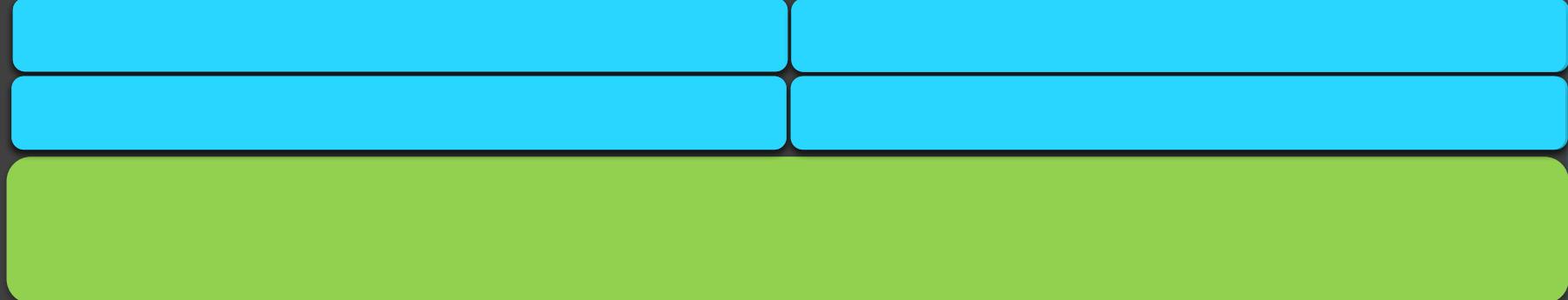
Requires 50% Hard Drive Space



Size Tiered Compaction

Partition Data Can Possibly be Scattered Amongst Several SSTables

HDD



Begin animation slides – copy and paste the following together to create the previous animation

Size Tiered Compaction

Partition Data Can Possibly be Scattered Amongst Several SSTables

HDD

TX	1	Joe
TX	2	Jim
TX	3	Jan
TX	4	Jon

TX 1 Joe

TX 2 Jim

TX 3 Jan

TX 4 Jon

TX	5	Job
----	---	-----

TX	6	Jep
----	---	-----

TX	7	Jed
----	---	-----

TX	8	Jax
----	---	-----

TX 5 Job

TX 6 Jep

TX 7 Jed

TX 8 Jax



TX 9 Jeb

TX 10 Jan

TX 11 Joi

TX 12 Lea

TX 9 Jeb

TX 10 Jan

TX 11 Joi

TX 12 Lea



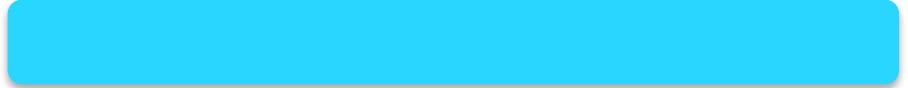
TX | 13 | Lyn

TX | 14 | Lon

TX | 15 | Lee

TX | 16 | Lex

TX | 13 | Lyn **TX** | 14 | Lon **TX** | 15 | Lee **TX** | 16 | Lex



Disappear Blue Names groups here – after prev

TX 1 Joe	TX 2 Jim	TX 3 Jan	TX 4 Jon	TX 5 Job	TX 6 Jep	TX 7 Jed	TX 8 Jax
TX 9 Jeb	TX 10 Jan	TX 11 Joi	TX 12 Lea	TX 13 Lyn	TX 14 Lon	TX 15 Lee	TX 16 Lex

TX	17	Sal
----	----	-----

TX	18	Sid
----	----	-----

TX	19	Alf
----	----	-----

TX	20	Ali
----	----	-----

TX	17	Sal
----	----	-----

TX	18	Sid
----	----	-----

TX	19	Alf
----	----	-----

TX	20	Ali
----	----	-----

TX 21 Ira

TX 22 Isa

TX 23 Ivy

TX 24 Iza

TX 21 Ira

TX 22 Isa

TX 23 Ivy

TX 24 Iza



TX 25 Eli

TX 26 Ely

TX 27 Eva

TX 28 Edd

TX 25 Eli

TX 26 Ely

TX 27 Eva

TX 28 Edd



TX 29 Kia

TX 30 Kim

TX 31 Kya

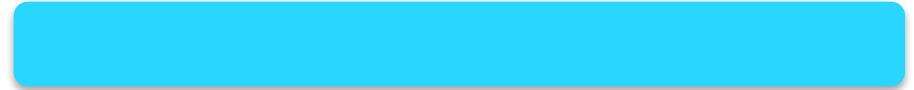
TX 32 Kai

TX 29 Kia

TX 30 Kim

TX 31 Kya

TX 32 Kai



TX 33 Tai

TX 34 Tea

TX 35 Tia

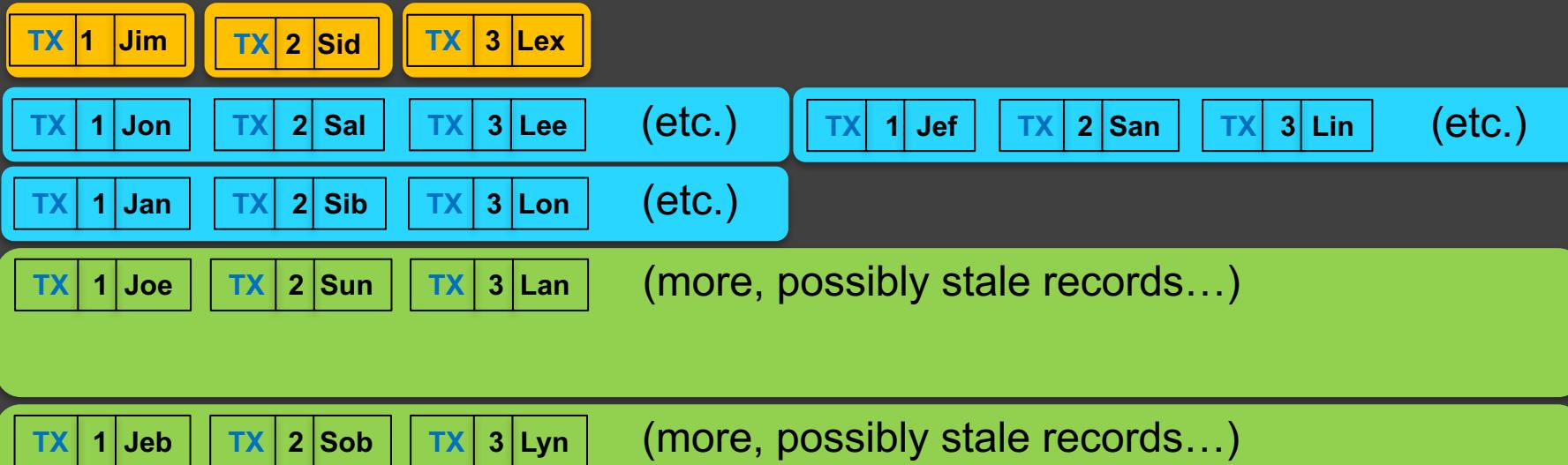
TX 36 Toi

End animation slides

Stale Data

Stale Records in Larger SSTables Take Unnecessary Space

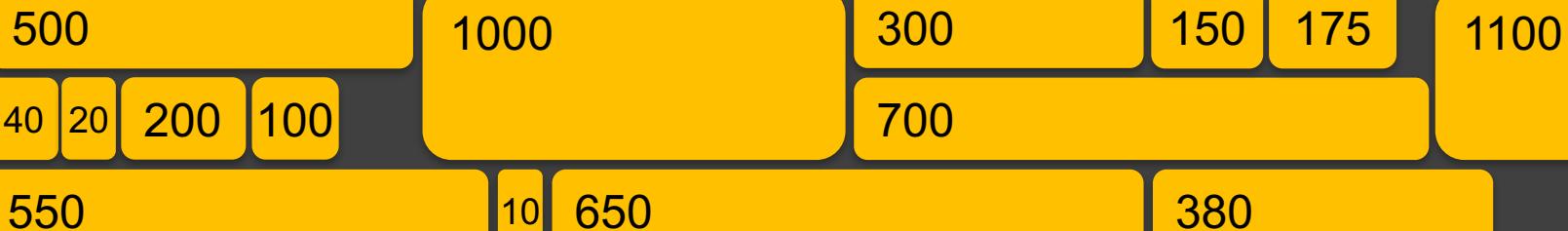
HDD



Realistic Scenario

SSTable sizes will vary

HDD



Realistic Scenario

SSTable sizes will vary

100

50 100 150

Realistic Scenario

SSTable sizes will vary



Realistic Scenario

SSTable sizes will vary



Realistic Scenario

SSTable sizes will vary

490

50 100 245 490 735 500 1000 1500

100

1000

Realistic Scenario

SSTable sizes will vary

20

<50

50 100 150

100

245 490 735

490

500 1000 1500

1000

Realistic Scenario

SSTable sizes will vary

300

<50

20

50 100 150

100

243 495 593

490

500 1000 1500

1000

Realistic Scenario

SSTable sizes will vary

140

<50

20

60 100 180

100

197 395 592

490

300

500 1000 1500

1000

Realistic Scenario

SSTable sizes will vary

<50

20

60 120 180

100

140

197 395 592

490

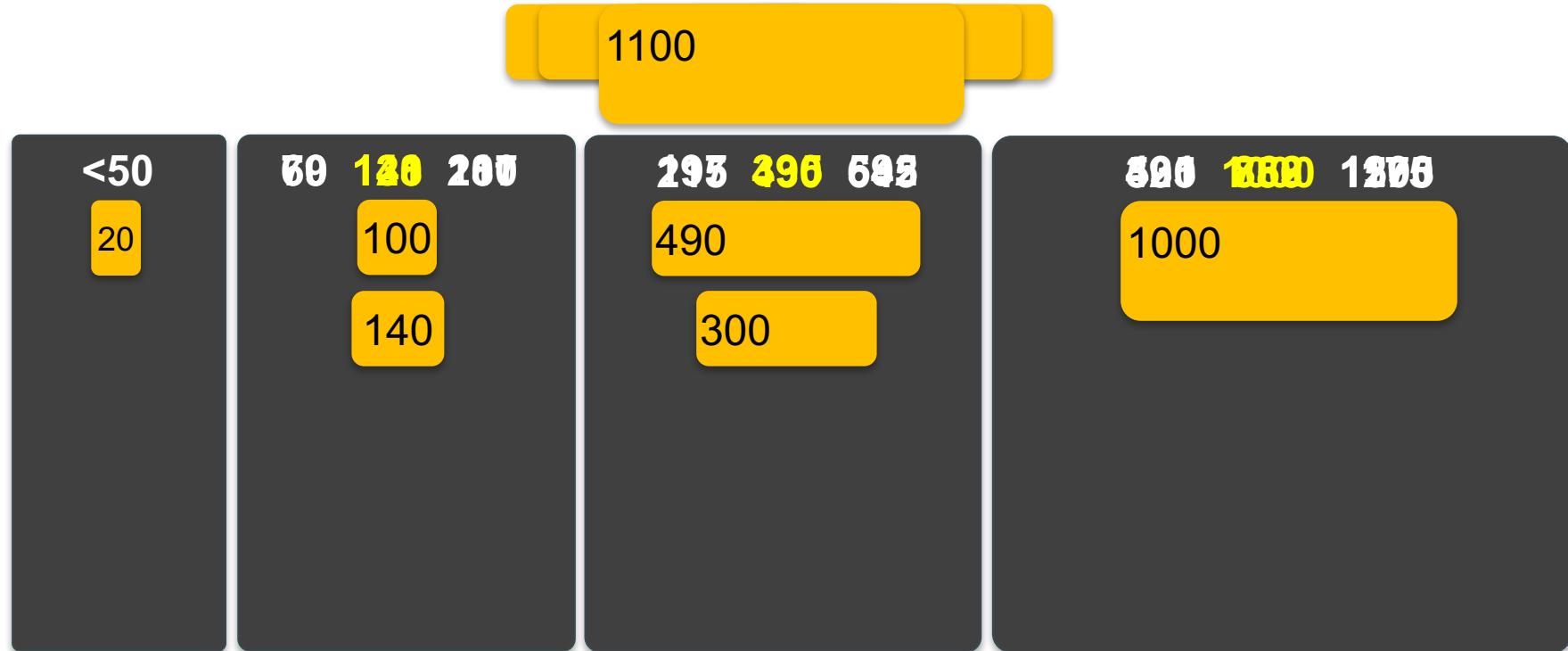
300

500 1000 1500

1000

Realistic Scenario

SSTable sizes will vary



Things to Note

- Groups similarly sized tables together
- Tiers with less than `min_threshold` (four) SSTables are not considered for compaction
- The smaller the SSTables, the "thinner" the distance between `min_threshold` and `max_threshold`
- SSTables qualifying for more than one tier distribute randomly amongst buckets
- Buckets with more than `max_threshold` SSTables are trimmed to just that many SSTables
 - 32 by default
 - Coldest SSTables dropped

Hotness

- Size tiered compaction chooses the hottest tier first to compact
- SSTable hotness determined by number of reads per second per partition key



Similar Sized Tables

- Similar sized SSTables compact together better
- SSTables of similar size will have a fair amount of overlap
- Minimizes write amplification (rewriting large amounts of data simply to copy it)
- Ex: Compacting a 1MB file with a 1TB file (not ideal)

Concurrency

- Cassandra compacts several tiers concurrently
- `concurrent_compactors`
 - Default to smaller of number of disks or number of cores, with a minimum of 2 and a maximum of 8 per CPU core
 - Tables concurrently compacting are not considered for new tiers

Triggering a Compaction

- Compaction starts every time a MemTable flushes to an SSTable
- MemTable too large, commit log too large, or manual flush
- Or when the cluster streams SSTable segments to the node
 - Bootstrap, rebuild, repair
- Compaction continues until there are no more tiers with at least `min_threshold` tables in it

Tombstones

- If no eligible buckets, size tiered compaction compacts a single SSTable
- This eliminates expired tombstones
- The number of expired tombstones must be above 20%
- Largest SSTable chosen first
- Table must be at least one day old before considered
 - `tombstone_compaction_interval`
- Compaction ensures that tombstones DO NOT overlap old records in other SSTables



Size Tiered Compaction

- As with everything, there are trade offs to using size tiered Compaction
- Size tiered compaction is the default
- Absorbs high write-heavy workloads by procrastinating compaction as long as possible
- Other compaction strategies don't handle ingesting data as well as size tiered
- `compaction_throughput_mb_per_sec` controls the compaction IO load on a node

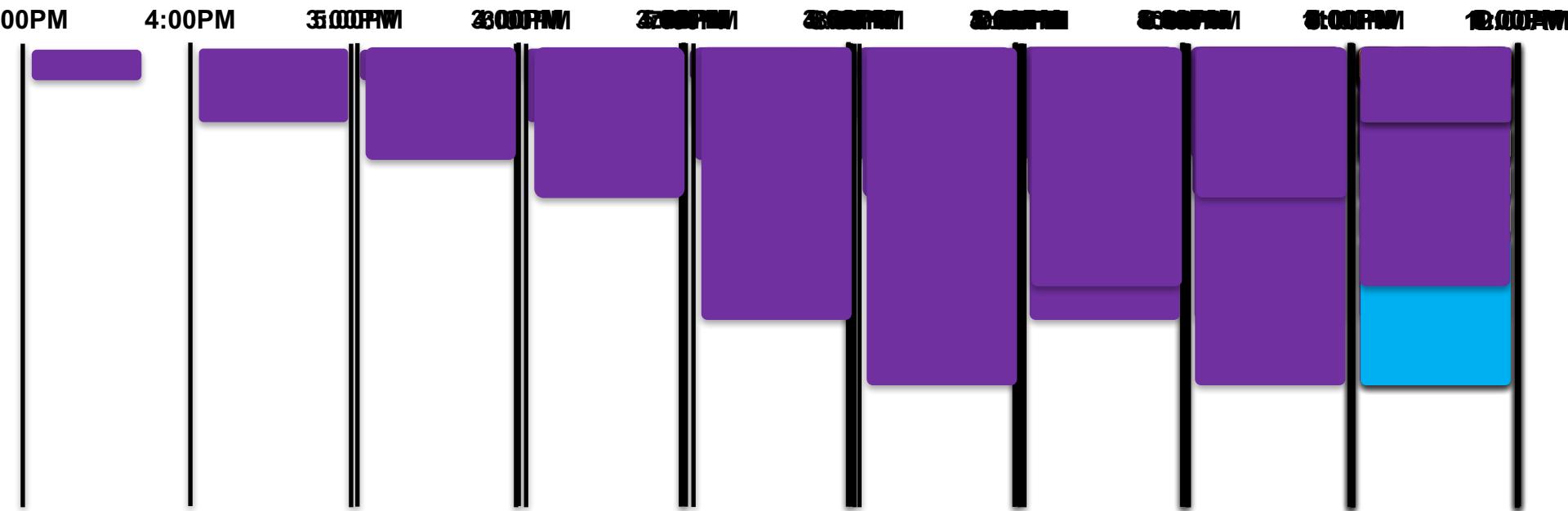
Major Compactions

- You can issue a major compaction via nodetool
- Compacts all SSTables into a single SSTable
- New monolithic SSTable will qualify for the largest tier
- Future updates/deletes will fall into smaller tiers
- Data in largest tier will become obsolete yet still hog a lot of disk space
- Takes a long time for changes to propagate up to large tier
- Major compactions not recommended

Time Window Compaction

Time Windowed Compaction

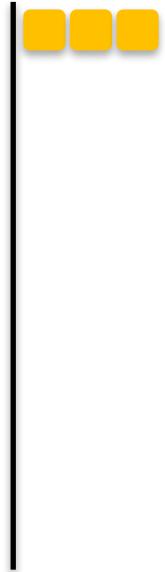
Built for time series data



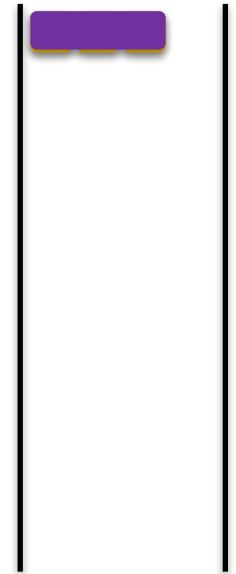
Time Windowed Compaction

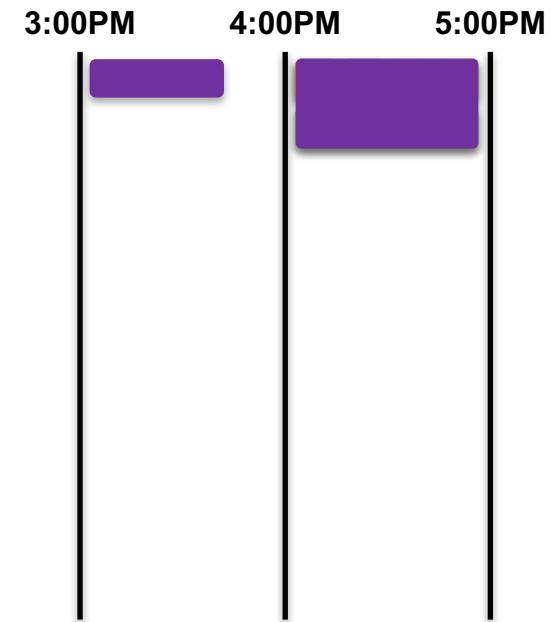
Built for time series data

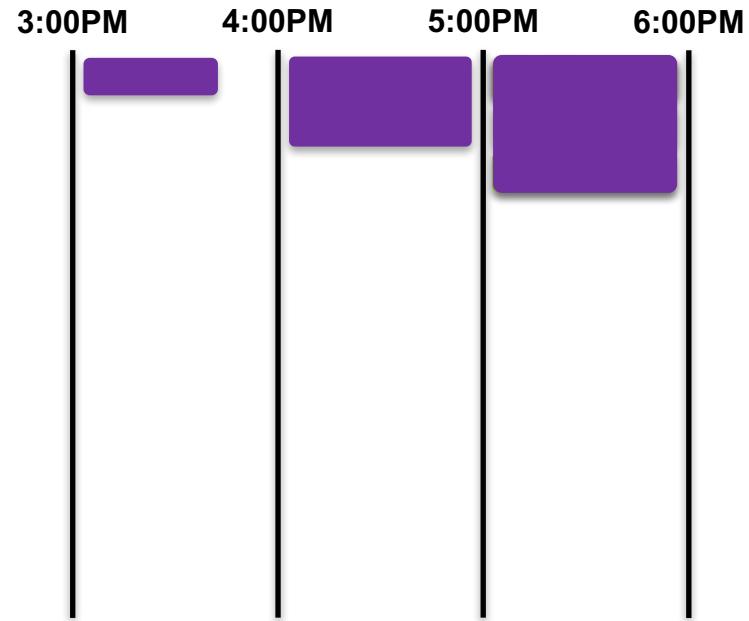
3:00PM

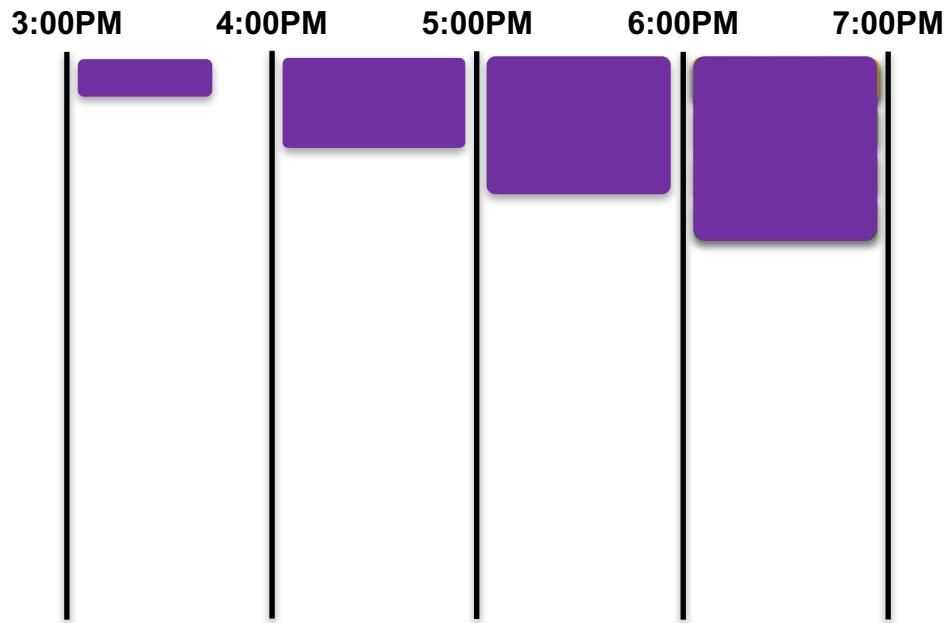


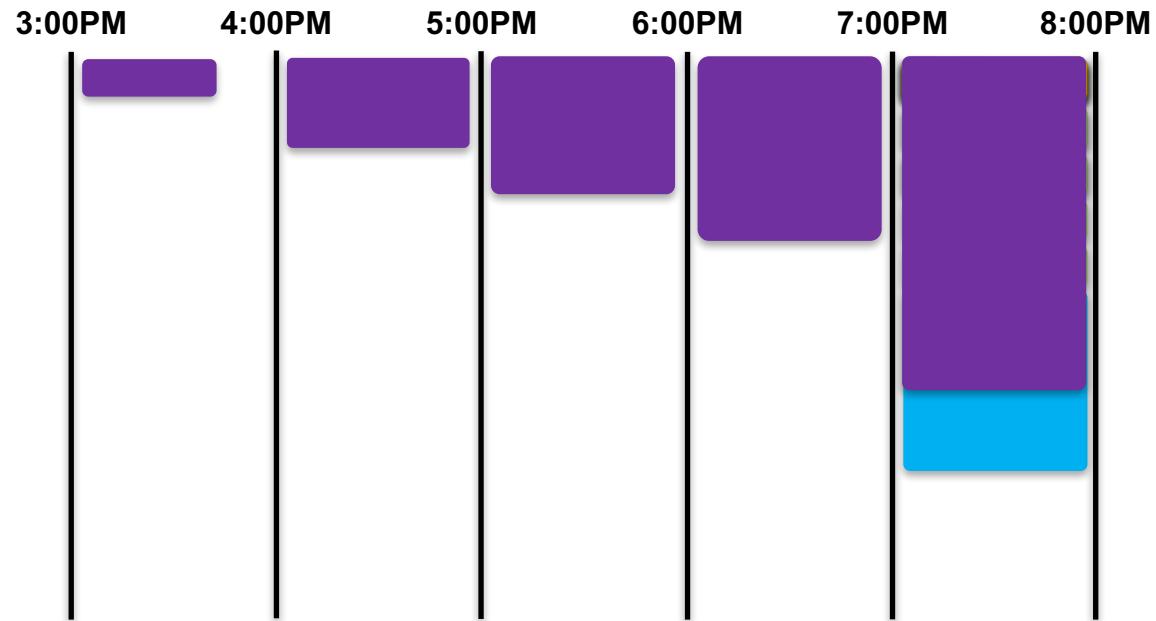
3:00PM 4:00PM

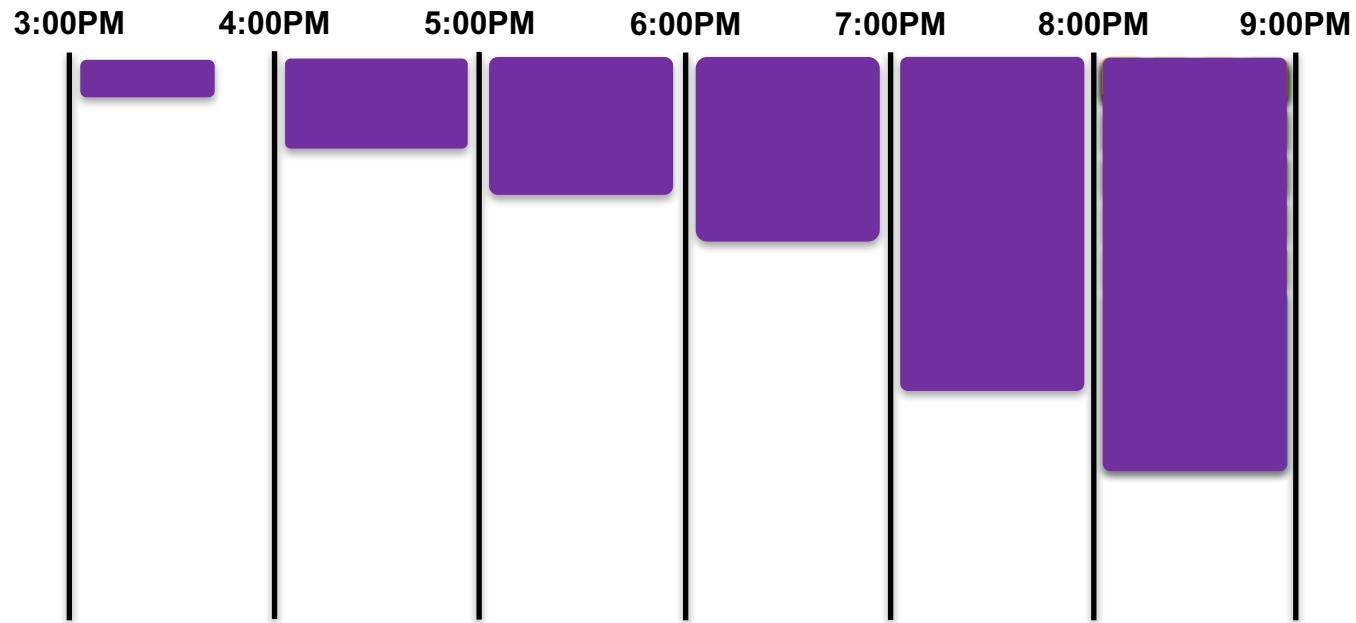


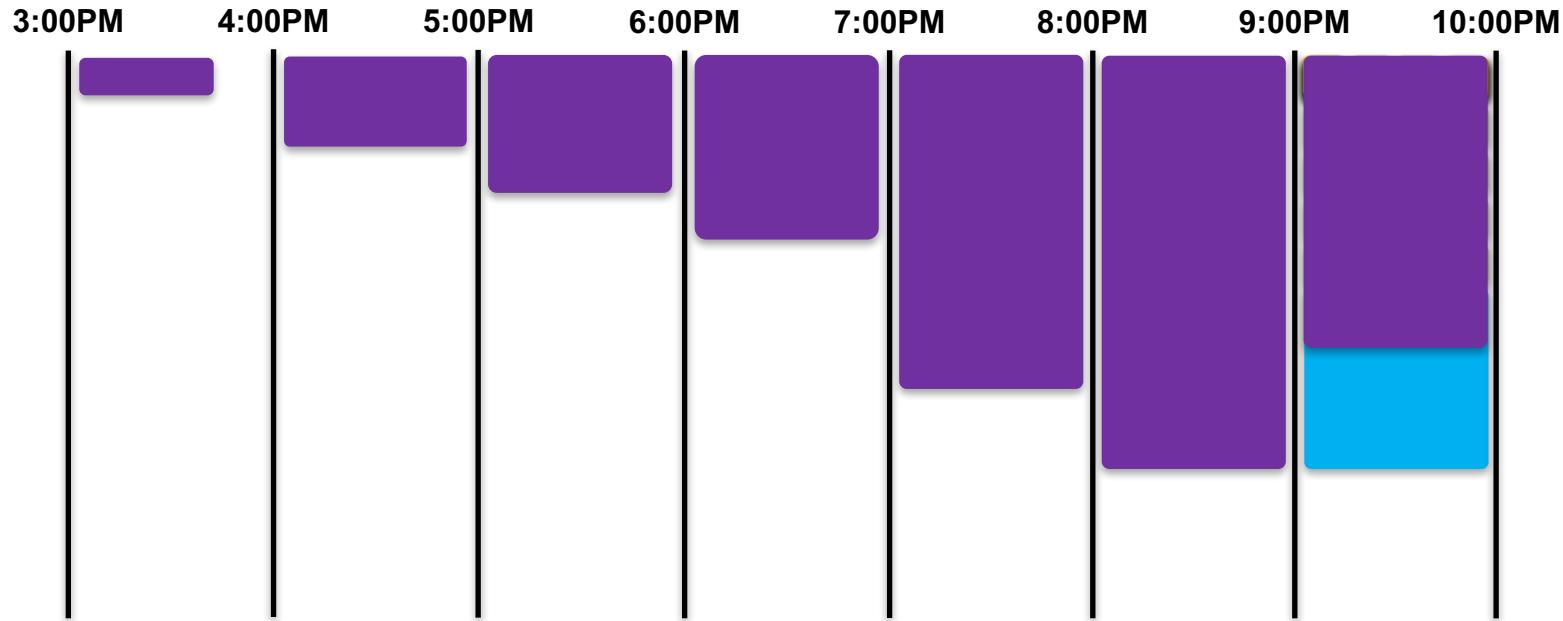


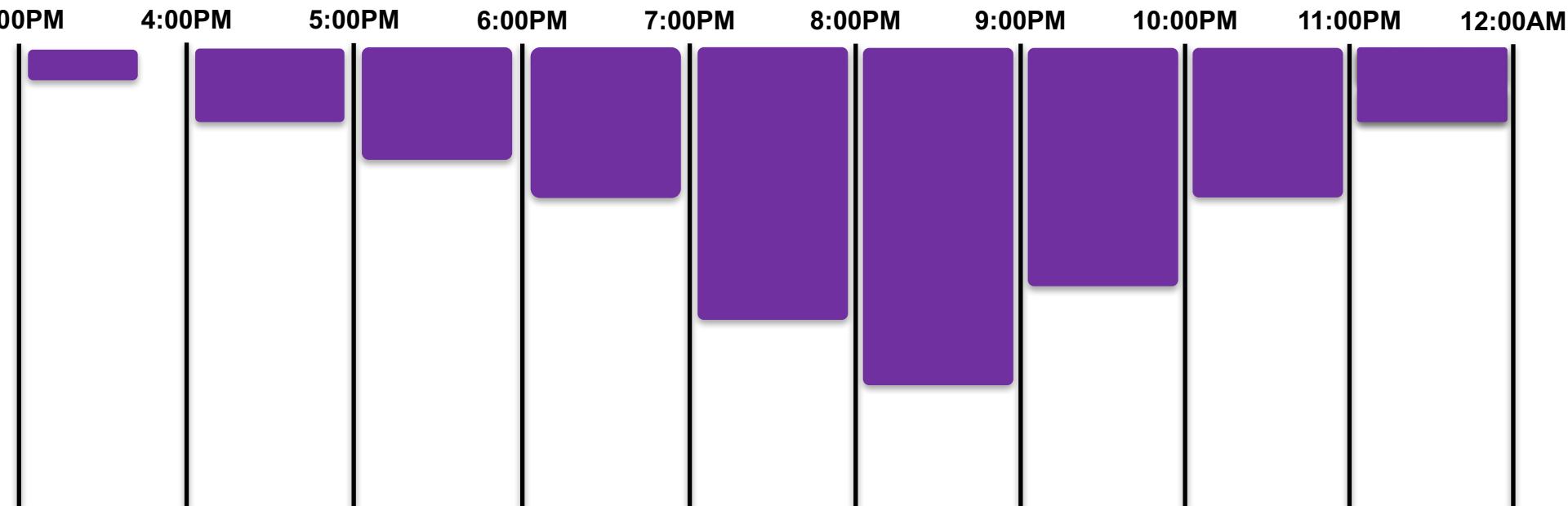












Time Window Details

- An SSTable spanning two windows simply falls into the second window
- Good practice to aim for 50ish max SSTables on disk
 - 20ish for active window
 - 30ish for all past windows combined
- For example: one month of data would have window of a day

Tuning

Simply set the window size

- `compaction_window_unit`
 - minutes
 - hours
 - days
- `compaction_window_size`
- Number of units in each window
- Ex: 15 days, 10 minutes, 20 hours, etc.

Tuning

- `expired_sstable_check_frequency_seconds` determines how often to check for fully expired (tombstoned) SSTables
- Good to tune when using a TTL

Repair

Repair

What is Repair?

- Think of repair as synchronizing replicas
- Repair ensures that all replicas have identical copies of a given partition
- Repair occurs:
 - If necessary when detected by reads (e.g. CL=QUORUM)
 - Randomly with non-quorum reads (table property `read_repair_chance` or `dclocal_read-repair_chance`)
 - Manually using `nodetool repair`

Repair

Why is Repair Necessary?

- Nodes may go down for a period of time and miss writes
 - Especially if down for more than `max_hint_window_in_ms`
 - If nodes become overloaded and drop writes

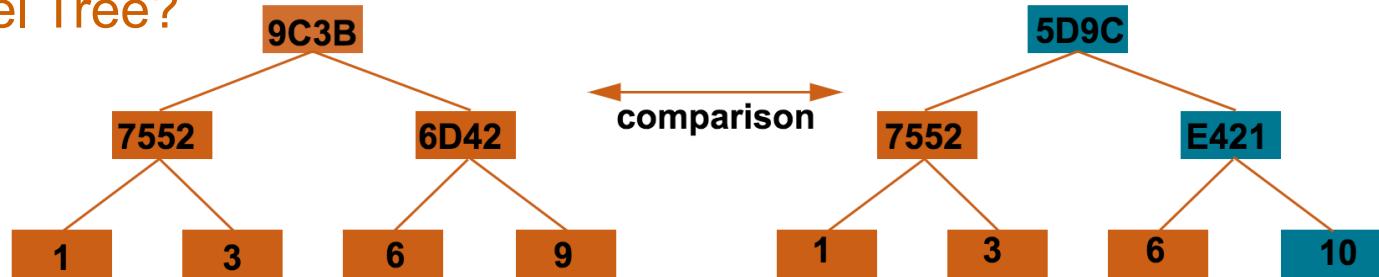
Repair

How Does Repair Work?

- 1) Nodes build Merkel trees from partitions to represent how current the data values are
- 2) Nodes exchange the Merkel trees
- 3) Nodes compare the Merkel trees to identify specific values that need synchronization
- 4) Nodes exchange data values and update their data

Repair

What is a Merkel Tree?



- A binary tree of hash values
- The leaves of the tree represent hashes of the values in the partition
- Each non-leaf tree node is a hash of its children's hash values
- When tree-nodes hashes are the same, the sub-trees are the same

Repair

When to Perform a Repair?

- If a node has been down for a while
- On a regular basis:
 - Once every `gc_grace_seconds`
 - Make sure the repair can complete within the `gc_grace_seconds` window
 - Schedule for lower utilization periods

Repair

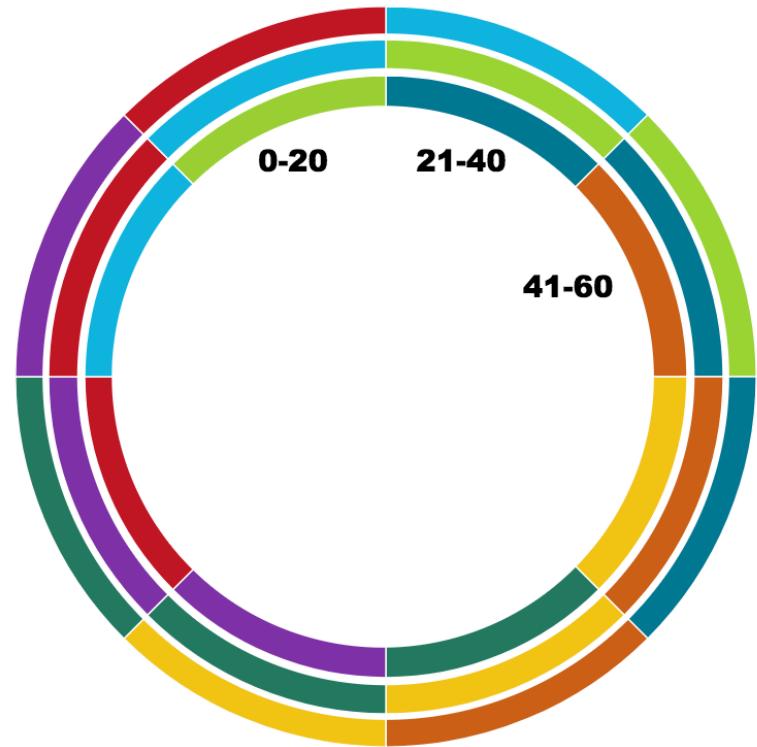
Is Repair a Lot of Work for the Node?

- A full repair *can* be a lot of work
- But there are ways to mitigate the work:
 - Primary range repair
 - Sub-range repair

Repair

What is a Primary Range Repair?

- The primary range is the set of tokens the node is assigned
- Repairing only the node's primary range will make sure that data is synchronized for that range
- Repairing only the node's primary range will eliminate redundant repairs



Repair

What is Sub-Range Repair?

- Repairs can consume significant resources depending on how much data is under consideration
- Targeting sub-ranges of the table will reduce the amount of work done by a single repair

Repair

How do I Use Repair?

```
nodetool <options> repair
```

Options:

- `--dc <dc_name>` identify data centers
- `--et <end_token>` used when repairing a subrange
- `--local` repairs only in the local data center
- `--par (parallel repair)`
- `--pr (partitioner-range)` repairs only primary range
- `--st <start_token>` used when repairing a subrange
- `--<keyspace> <table>`

NodeSync

NodeSync

What is NodeSync?

- Behaves like continuous background repair that delivers:
 - Low overhead
 - Consistent performance
 - Ease of use
- DSE only feature

NodeSync

Distributed system synchronization evolution

- Traditional repair has high operational and overhead costs
- Repair Service is easy to use but can have significant performance implications
- NodeSync
 - captures the benefits of previous approaches while managing and minimizing performance impact
 - Keeps data synchronized with low, predictable overhead

NodeSync

Benefits

- Few edge-cases to worry about
- Predictable synchronization overhead
- Seamless to upgrade to NodeSync
- Manage NodeSync via existing CLI and OpsCenter interfaces
- Overall easier operations

NodeSync

Use cases

- Set-and-forget background synchronization
- One-time manual synchronization for targeted use

NodeSync

How to use it

- Create a cluster with at least two nodes
- Create a keyspace with replication factor ≥ 2 :

```
CREATE KEYSPACE MyKeyspace
    WITH replication={‘class’: ‘SimpleStrategy’,
                      ‘replication_factor’: 2};
```

- Create a table within the keyspace with NodeSync enabled:

```
CREATE TABLE MyTable (k int PRIMARY KEY)
    WITH nodesync={‘enabled’: ‘true’};
```

- NodeSync will now automatically make sure the table data is synchronized

NodeSync

Tuning options – per node basis

- Rate
 - Maximum bytes/second NodeSync will validate
 - Used to reduce the impact of synchronization
 - May also increase the synchronization window
- Target
 - Aspirational time between two validations of the same data
 - Should be less than `gc_grace_seconds`
 - Default value is `max(gc_grace_seconds, 4h)`
- Note that target and rate may conflict – rate wins

NodeSync

Rate configuration

- Via `cassandra.yaml`:

```
# NodeSync settings.  
nodesync:  
    # The (maximum) rate (in kilobytes per second) for data validation.  
    rate_in_kb: 1024
```

- Via `nodetool` (at runtime):

```
nodetool nodesyncservice setrate <value_in_kb_per_sec>  
nodetool nodesyncservice getrate
```

NodeSync

Target configuration

- Set the target for each table

```
ALTER TABLE MyTable WITH nodesync={‘enabled’: ‘true’,  
                         ‘deadline_target_sec’: <value>};
```

NodeSync

CLI tools

- nodetool nodesyncservice
 - Performs various operations on a single node
 - Enable/disable/status
 - Set/get rate
 - Rate simulation
- nodesync
 - Performs cluster-wide operations
 - Enable/disable NodeSync on table(s)
 - Submit/cancel user triggered validations (advanced tool)
 - Tracing

sstablesplit

When Would We Need `sstablesplit`?

- You did a `nodetool compact` (major compaction)
- Maybe you used `SizeTieredCompactionStrategy` for a major compaction
- This would result in a excessively large SSTable
- Good idea to split the table because won't get compacted again until the next huge compaction
- Using size tiered compaction, we may have gotten some really large files over time
- May find yourself with a 200GB file that you need to split up
- It's an anti-compaction in a way



sstablesplit

Usage

Stop Cassandra before you use this tool!

```
$ sudo service cassandra stop
```

- You do this online and it will be bad!!



sstablesplit

Usage

```
$ sstablesplit [options] <filename> [<filename>]*
```

Flag	Option	Description
	--debug	Displays stack traces
-h	--help	Displays help
	--no-snapshot	Do not snapshot the SSTable before splitting
-s	--size <size>	Max size (in MB) for the output SSTables (default 50)
-v	--verbose	Verbose output

sstablesplit

Example

```
$ sstablesplit -s 40 /var/lib/cassandra/data/data/killrvideo/users/*
```

Take all my files in the killrvideo keyspace and make all of them 40MB

Multi Datacenter Concepts

Nodes, Racks and Datacenters

A cluster of nodes can be logically grouped as racks and data centers

- Node— the virtual or physical host of a single Cassandra instance
- Rack— a logical grouping of physically related nodes
- Data Center— a logical grouping of a set of racks
- Enables geographically aware read and write request routing
- Each node belongs to one rack in one data center
- The identity of each node's rack and data center may be configured in its **conf/cassandra-rackdc.properties** file

Multi Datacenter

Adding a second data center

- Ensures continuous availability of your data and application
- Live backup
- Improved performance
- Analytics

Multi Datacenter

How clusters operate between datacenters

- A data center is a grouping of nodes configured together for replication purposes
- Data replicates across data centers automatically and transparently
- Consistency level can be specified at LOCAL level for read/write operations
- Consistency level can also be specified as EACH

Multi Datacenter

What if one datacenter goes down?

- Failure of a datacenter will likely go unnoticed by users
- If node(s) fail, they will stop communicating via gossip
- Recovery can be accomplished with a rolling repair to all nodes in failed datacenter

Multi Datacenter

Implementing a multiple datacenter cluster

- Use the NetworkTopologyStrategy rather than SimpleStrategy
- Use LOCAL_* consistency level for read/write operations to limit latency
- Specify the snitch

Multi Datacenter

After starting Cassandra on the new nodes

- Change the keyspace properties to specify the desired replication factor for the new datacenter
- For example, set strategy options to DC1:2, DC2:2:

```
ALTER KEYSPACE killrvideo WITH replication =  
  {'class': 'NetworkTopologyStrategy', 'DC1' : 1, 'DC2' : 2}
```

Multi Datacenter

Implementing a multiple datacenter cluster

```
$ nodetool rebuild -- name_of_existing_data_center
```

- Run nodetool rebuild specifying the existing datacenter on all nodes in the new datacenter
- Otherwise, requests to the new datacenter with LOCAL_ONE or ONE consistency levels may fail if the existing datacenters are not completely in-sync.

Exercise - Stand-up a Second DC

Backup, Restore and Moving Data

CQL Copy

CQL Copy

Imports and exports delimited data to and from Cassandra

```
COPY table_name ( column, ... )
    FROM ( 'file_name1', 'file_name2', ... | STDIN )
    WITH option = 'value' AND ...
```

```
COPY table_name ( column , ... )
    TO ( 'file_name1', 'file_name2', ... | STDOUT )
    WITH option = 'value' AND ...
```

- COPY uses an argument of one or more comma-separated file names

CQL Copy

COPY FROM a delimited file

Why would we want to do this?

- In a production database, inserting columns and column values programmatically is more practical than using cqlsh
- But often testing queries using this SQL-like shell is very convenient
- A delimited file is useful if several records need inserting
- While not strictly an INSERT command, it is a common method for inserting data

CQL Copy

Some rules

- Cassandra expects every row in the delimited input to contain the same number of columns
- The number of columns in the delimited input is the same as the number of columns in the Cassandra table
- Empty data for a column is assumed by default as NULL value
- COPY FROM is intended for importing small datasets (a few million rows or less) into Cassandra
- For importing larger datasets, use DSBulk

CQL Copy

Procedure

- Locate your delimited file and check options to use

```
---  
lastname|firstname|email|created_date  
Jones|Bob|bob@example.com|2016-02-03  
Walsh|Gary|gary@example.com|2016-02-04  
Nash|Sue|sue@example.com|2016-02-23  
---
```

- To insert the data, using the COPY command with delimited data

```
$ COPY killrvideo.users FROM 'users.csv'  
    WITH DELIMITER='|' AND HEADER=TRUE
```

CQL Copy

What do some of these options mean?

Examples:

- DELIMITER - Set character that separates fields having newline characters in the file (default is comma (,))
- HEADER - Set true to indicate that first row of the file is a header (default is false)
- CHUNKSIZE - Set the size of chunks passed to worker processes (default value is 1000)
- SKIPROWS - The number of rows to skip (default value is 0)

CQL Copy

COPY TO a delimited file

- Assuming you have this user table in CQL:

```
cqlsh> SELECT * FROM users.killrvideo;
```

- After inserting data into the table, you can copy the data to a delimited file in another order by specifying the column names in parentheses after the table name:

lastname	firstname	email	created_date
Jones	Bob	bob@example.com	2016-02-03

CQL Copy

COPY TO a delimited file

- Copy the data to a delimited file in another order by specifying the column names in parentheses after the table name:

```
COPY users (firstname, lastname, created_date, email) TO 'users.csv'
```

CQL Copy

Specifying the source or destination files

- Specify the source file of the delimited input or the destination file of the delimited output by a file path.
- You may also use the **STDIN** or **STDOUT** keywords to import from standard input and export to standard output.
- When using **stdin**, signal the end of the delimited data with a backslash and period ("\\.") on a separate line.

sstabledump

sstabledump

Allows you to see the raw data of a SSTable in a text format

- Dumps the contents of the specified SSTable in the JSON format
- Maybe you're curious about things like timestamps
- Only way you can actually see the inner workings of an SSTable
- You may wish to flush the table to disk (using nodetool flush) before dumping its contents



sstabledump



Cassandra 3.0 sstabledump output

```
$ tools/bin/sstabledump data/data/killrvideo/users-b75e90f0226811e6a172cf782e39035/ma-1-big-Data.db
[
  {
    "partition" : {
      "key" : [ "14c532ac-f5ae-479a-9d0a-36604732e01d" ],
      "position" : 0
    },
    "rows" : [
      {
        "type" : "row",
        "position" : 30,
        "liveness_info" : { "tstamp" : 1464175401324444 },
        "cells" : [
          { "name" : "created_date", "value" : "2016-05-11 00:00-0230" },
          { "name" : "email", "value" : "jane@example.com" },
          { "name" : "firstname", "value" : "Jane" },
          { "name" : "lastname", "value" : "Doe" }
        ]
      }
    ]
  }
]
```

sstabledump

Internal Representation Format

```
$ sstabledump data/data/killrvideo/users-b75e90f0226811e6a172cf782e39035/ma-1-big-Data.db -d  
[14c532ac-f5ae-479a-9d0a-36604732e01d]@0 Row[info=[ts=1464175401324444] ]: |  
[created_date=2016-05-11 00:00-0230 ts=1464175401324444], [email=jane@example.com  
ts=1464175401324444], [firstname=Jane ts=1464175401324444], [lastname=Doe ts=1464175401324444]
```

sstableloader (Cassandra bulk loader)

sstableloader

- Provides the ability to:
 - Bulk load external data into a cluster
 - Load pre-existing SSTables into
 - an existing cluster or new cluster
 - a cluster with the same number of nodes or a different number of nodes
 - a cluster with a different replication strategy or partitioner
- Very flexible

sstableloader

- The sstableloader streams a set of SSTable data files to a live cluster; it does not simply copy the set of SSTables to every node, but transfers the relevant part of the data to each node, conforming to the replication strategy of the cluster
- The table into which the data is loaded does not need to be empty
- If tables are repaired in a different cluster, after being loaded, the tables are not repaired



sstableloader

How does sstableloader work?

- When it runs, it actually reads every single sstable and streams that data back into the cluster
- It repartitions the data. The snitches get involved so that the data will fall into the correct place
- Great for switching between non-like clusters, as it can re-stream the data and repartition it (going from a 30 to 40 node cluster, for example)
- Example of usage: To bulk load the files, specify the path to killrvideo/users/ in the target cluster:

```
$ sstableloader -d 110.82.155.1 /var/lib/cassandra/data/killrvideo/users/
```

sstableloader

Prerequisites

- Since sstableloader uses Cassandra gossip to figure out the topology of the cluster, make sure of the following:
 - The `cassandra.yaml` is in the classpath and properly configured
 - At least one node in the cluster is configured as seed
 - The following properties are properly configured in `cassandra.yaml` for cluster that you are importing into:
 - `cluster_name`
 - `listen_address`
 - `storage_port`
 - `rpc_address`
 - `rpc_port`

Spark for Data Loading

Spark For Data Loading

Spark provides convenient functionality for loading large external datasets into Cassandra tables in parallel

Ingesting files in CSV, TSV, JSON, XML, and other formats

If file is stored in Cassandra File System

- File blocks are ingested concurrently (64 MBs per block)
- May still need to re-partition to optimize

If file is stored in a local file system, to achieve a desired level of parallelism

- Use Spark to repartition the dataset or
- Use Spark's partitionBy to pre-partition the dataset by a Cassandra table partition key

Spark For Data Loading

```
case class User (userid: java.util.UUID,  
    firstname: String,  
    lastname: String,  
    email: String,  
    created_date: java.util.Date)  
val users = sc.textFile("dsefs:.../users.csv")  
    .repartition(2 * sc.defaultParallelism)  
    .map( line => line.split(",")  
        match {  
            case Array(id, firstname, lastname, email, created_date)  
                => User(java.util.UUID.fromString(id),  
                    firstname,  
                    lastname,  
                    email,  
                    new java.text.SimpleDateFormat("yyyy-mm-dd").parse(created_date))  
        }  
    )  
    users.saveToCassandra("killrvideo", "users")
```

Spark For Data Loading

Loading a CSV file into a Cassandra table with validation

```
val beforeCount = sc.cassandraTable("killrvideo", "users").count

val users =
    sc.textFile("file:///home/student/users.csv")
    .repartition(2 * sc.defaultParallelism)
    .cache // The RDD is used in two actions

val loadCount = users.count
users.map(...).saveToCassandra("killrvideo", "users")
val afterCount = sc.cassandraTable("killrvideo", "users").count
if (loadCount - (afterCount - beforeCount) > 0)
    println ("Errors or upserts - further validation required")
```

DSE DSBulk

DSBulk

What is it?

- Moves Cassandra data to/from files in the file system
- Uses both CSV or JSON formats
- Command-line interface
- DSE Customer First feature

DSBulk

Why DSBulk?

- Loading a lot of data into DSE DB has been difficult for a long time
- Unloading was needed too
- Previous tools were not ideal:
 - CQLSH COPY FROM is non-performant and not robust
 - SSTableLoader requires data to be in SSTable format
 - cassandra-loader is not formally supported

DSBulk

Use cases

- Loading data from a pile of files
- One-time load or part of production flow
- Initial developer experience (load an existing familiar DB)
- Unload data for backup
- Migration from DSE to DSE (due to data model changes)

DSBulk

Command syntax

```
dsbulk (<workflow>|help) <options>
    <workflow> = load | unload
```

Some example options (see `dsbulk help` for complete list):

`-f <string>`

Load settings from the given file rather than `conf/application.conf`.

`-c,--connector.name <string>`

The name of the connector to use. Defaults to `csv`.

`-k,--schema.keyspace <string>`

The keyspace to connect to. Optional. If not specified, then `schema.query` must be specified.

`-t,--schema.table <string>`

The destination table. Optional. If not specified, then `schema.query` must be specified.

DSBulk

Example invocation

- `dsbulk load -url file1.csv -k ks1 -t table1`
- Loads CSV data from `file1.csv` to `ks1.table1`
 - `$DSBULK_HOME/conf/application.conf` has mapping specifications
 - Mapping specifications are in HOCON format
 - <https://en.wikipedia.org/wiki/HOCON>
 - See documentation for mapping file details

Backup Fundamentals

Why Should We Backup Our Data?

"There are two types of users: those who have lost data and those who are about to lose data."

- Programmatic accidental deletion or overwriting of data
- For single node failure, recovery can be from a live replica
- So that we may recover from catastrophic data center failure

Why Snapshots

- We don't do backup like traditional databases where we copy out all the data
- It's a distributed system; every server or node has a portion of the data
- SSTables are immutable, which is great! Makes them easy to back up

Why Snapshots

- Snapshots create hardlinks on the file system as opposed to copying data
 - This is DIFFERENT than copying actual data files offline (takes less disk space)
 - Therefore very fast! (Not a lot of data in flight)
 - Represents the state of the data files at a particular point in time
 - Can consist of a single table, single keyspace, or multiple keyspaces

What Is A Snapshot?

- Represents the state of the data files at a particular point in time
- Snapshot directory is created (this has pointers)
- Then you can either leave it there or copy it offline to an NFS mount or copy to S3 etc.

Backup Details

How Do Incremental Backups Work?

- Incremental backups create a hard link to every SSTable upon flush
 - User must manually delete them after creating a new snapshot
- Incremental backups are disabled by default
 - Configured in the **cassandra.yaml** file by setting **incremental_backups** to **true**
- Need a snapshot before taking incremental backups
- Snapshot information is stored in a **snapshots** directory under each table directory
 - Snapshot need only be stored once offsite

How Do Incremental Backups Work?

- Incremental backups are all stored in a backups directory under the keyspace data directory
 - Enables storing incremental backups offsite more easily
- Both snapshot and incrementals are needed to restore data
- Incremental backup files are not automatically cleared
 - Clear when a new snapshot is created

Where Are Snapshots Stored?

- Snapshots and incremental backups are stored on each Cassandra node
 - Handy, if a simple restore is needed
 - Not so good if there is a hardware failure
- Commonly, files are copied to an off-node location
 - Open source program, tablesnap, is useful for backing up to S3
 - Scripts can be used to automate backing up files to another machine
 - cron + bash script, rsync, etc.

Auto Snapshot

CRITICAL safety factor!

- A configuration setting in `cassandra.yaml` that indicates whether or not a snapshot is taken of data before tables are truncated and tables and keyspaces are dropped
- STRONGLY advise using the default setting of `true`

How Do We Use nodetool to Create Snapshots?

```
bin/nodetool [options] snapshot (-cf <table> | -t <tag> -- keyspace)
```

- We can specify to take a snapshot of:
 - one or more keyspaces
 - a table specified to backup data
- Flags
 - -h [host] | [IP address]
 - -p port
 - -pw password
 - -u username
- parallel ssh tool to snapshot entire cluster

How Do We Remove Snapshot?

```
nodetool clearsnapshot
```

- The nodetool clearsnapshot command removes snapshots
- Same options as nodetool command
- Specify the snapshot file and keyspace
- Not specifying a snapshot name removes all snapshots
- Remember to remove old snapshots before taking new ones --previous snapshots are not automatically deleted
- To clear snapshots on all nodes at once, use a parallel ssh utility

How Do We Restore Snapshots?

"You get 1 point for a backup, you get 99 for a restore"

- Most common method is to delete the current data files and copy the snapshot and incremental files to the appropriate data directories
 - If using incremental backups, copy the contents of the backups directory to each table directory
 - Table schema must already be present in order to use this method
 - Restart and repair the node after the file copying is done
- Another method is to use the sstableloader
 - Great if you're loading it into a different size cluster
 - Must be careful about its use as it can add significant load to cluster while loading

Performing Cluster-wide Backup and Restore

- OpsCenter
- SSH programs
 - pssh
 - clusterssh is another tool that can be used to make changes on multiple servers at the same time
- Honorable mention —tablesnap and tablerestore
 - For Cassandra backup to AWS S3
- Recovery

Exercise - Backup/Restore

JVM Tuning Options

JVM Settings

JVM Settings

The Cassandra Technology Stack

- Cassandra is a program written in Java
- Java programs execute on a Java Virtual Machine (JVM)
- The JVM runs on top of the computer's operating system
- The operating system is the software interface to the computer's hardware

JVM Settings

Performance Tuning

- We can adjust parameters at each level of Cassandra's technology stack
 - Cassandra settings (in `cassandra.yaml`)
 - JVM settings (in `jvm.options`)
 - Operating system settings (kernel configuration)
 - Hardware (adding or upgrading)

JVM Settings

JVM Settings - Memory Management

- The most significant settings in the JVM deal with memory management
- JVM memory has several designated areas, including code, stack and heap
- The heap is where Java programs allocate and deallocate transient memory
- Garbage collection refers to when the JVM reclaims the deallocated memory in the heap

JVM Settings

jvm.options

- We usually configure JVM settings when we launch Java programs
 - As command line options
- `cassandra-env.sh` is a shell script that launches the Cassandra server
- `cassandra-env.sh` includes `jvm.options` which has our desired settings

Warning: Only adjust these settings when you understand why you are doing it. The default settings are there for a reason.

JVM Settings

Heap Sizing Options

- `MAX_HEAP_SIZE` set to maximum of 8GB
 - Large heaps can introduce GC pauses that lead to latency
 - Different workloads can justify different settings
- `HEAP_NEWSIZE` set to 100MB per core
 - Example: 8 cores would mean 800MB
 - The larger this is, the longer GC pause times will be. The shorter it is, the more frequently GC will run
 - Different workloads may react differently

JVM Settings

Tuning the Java heap

System Memory	Heap Size
Less than 2GB	$\frac{1}{2}$ of system memory
2GB to 4GB	1GB
4GB or greater with CMS	$\frac{1}{4}$ of system memory, but no more than 14GB
4GB or greater with G1	$\frac{1}{4}$ of system memory

JVM Settings

Don't make the Java heap size too large!

- Capability of Java to gracefully handle garbage collection above 8GB quickly diminishes
- May interfere with operating system's ability to maintain OS page cache for frequently accessed data

JVM Settings

Tuning Java Garbage Collection

- As of Java 9, the default collector is the G1 garbage collector
- G1 garbage collector is better than CMS for large heaps
 - Works on regions of heap with most garbage first
 - Compacts the heap on-the-go

JVM Settings

JMX Options

- JMX is a Java technology
- Supplies hooks for monitoring Java applications and services
- JMX also lets you set some JVM options at runtime

JVM Settings

JMX Options

- `com.sun.management.jmxremote.port`
 - The port on which Cassandra listens from JMX connections.
- `com.sun.management.jmxremote.ssl`
 - Enable/disable SSL for JMX.
- `com.sun.management.jmxremote.authenticate`
 - Enable/disable remote authentication for JMX.
- `-Djava.rmi.server.hostname`
 - Sets the interface hostname or IP that JMX should use to connect.
Uncomment and set if you are having trouble connecting.

Garbage Collection

Garbage Collection

- Garbage collection is a process that a JVM is undergoing all the time to clean out any processes that are not live
- Objects are moved and deleted to free up memory
- GC should happen often enough to create large sections of free memory, but not so often that the CPU is churning on GC all the time
- Since Cassandra runs in a JVM, the pauses to do garbage collection affect Cassandra's performance
- Sizing the JVM is important to performance
- The number of CPUs can also affect performance
- DSE 6.0 keeps one core available for GC and other maintenance activities

What to Consider When Tuning Garbage Collection

- Pause time
 - length of time the collector stops the application while it frees up memory
- Throughput
 - Determined by how often the garbage collection runs, and pauses the application
 - More often the collector runs, the lower the throughput
- We want to minimize length of pauses as well as frequency of collection

JVM Available Memory

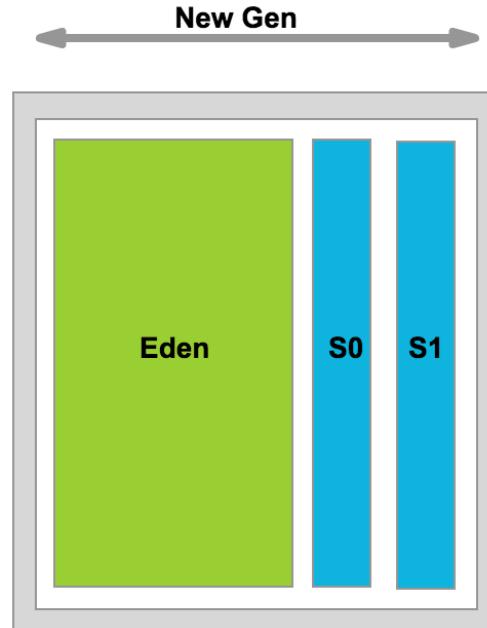


- Permanent generation
- New generation (ParNew)
- Old generation (CMS)

G1 does essentially the same thing,
but with many smaller memory sections

The New Generation

- Eden
- 2 survivor spaces



The New Generation

- Once Eden fills up with new objects, JVM triggers a minor GC
- A minor GC stops execution, iterates over the objects in Eden, copies any objects that are not (yet) garbage to the active survivor space, and clears Eden
- If the minor GC fills up the active survivor space, it performs the same process on the survivor space
- Objects that are still active are moved to the other survivor space, and the JVM clears the old survivor space

The New Generation

ParNew collection of the new generation:

- It's a stop-the-world algorithm
- Fast:
 - Finding and removing garbage
- Slow:
 - Moving active objects from eden to the survivor spaces
 - Moving active objects from the survivor spaces to the old gen

The Old Generation



- Contains objects that have survived long enough to not be collected by a minor GC
- the CMS collector runs when 75% full

Full GC

- Multi-second GC pauses = Major collections happening
- If the old gen fills up before the CMS collector can finish, the application is paused while a full gc runs
- Full GC checks everything: new gen, old gen, and perm gen,
- Significant (multi-second) pauses

Heap Dump

Heap Dump

- Useful when troubleshooting high memory utilization or OutOfMemoryErrors
- Show exactly which objects are consuming most of the heap
- Cassandra starts Java with the option -XX:+HeapDumpOnOutOfMemoryError

Heap Dump

- Perhaps you are getting a message similar to the following:

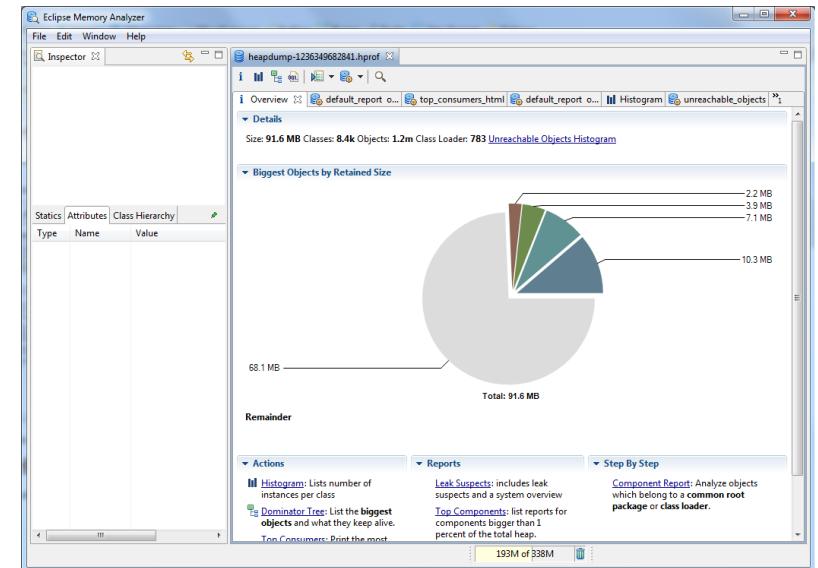
```
WARN [ScheduledTasks:1] 2012-10-22 12:14:49,889 GCInspector.java (line 145)
Heap is 0.9941227313009479 full. You may need to reduce memtable and/or
cache sizes. Cassandra will now flush up to the two largest memtables to
free up memory. Adjust flush_largest_memtables_at threshold in
cassandra.yaml if you don't want Cassandra to do this automatically
```

Heap Dump

- By default, Cassandra puts the file in a subdirectory of the working, root directory when running as a service
- BUT!
 - If Cassandra does not have write permission to the root directory, the heap dump fails
 - If the root directory is too small to accommodate the heap dump, the server crashes

Analyzing Heap Dumps

- Eclipse Memory Analyzer Tool (MAT)



Exercise – JVM Tuning

Tuning the Kernel

Time Sync

Time Sync



- Cassandra nodes identify valid data using timestamps
 - All nodes within a Cassandra cluster need to have synchronized clocks
- Time Stamp Counter (TSC) is a simple register within the CPU that counts the number of clock cycles
 - Over time, TSC will drift because the clock cycles may vary between CPUs
- Network Time Protocol (NTP) is a way to synchronize CPU clocks
 - Nodes communicate with a hierarchy of time servers to adjust their clock
 - Adjustments occur every 1-20 minutes

Kernel Configuration

Kernel Configuration

User resource limits

- Operating systems limit resources to prevent processes from hogging resources
- To view current limits **ulimit -a**

Kernel Configuration

User resource limits

- Since Cassandra nodes don't need to share resources, these limits are not helpful
- Turn them off globally by editing limits.conf
- Limits take effect when you login
- For Ubuntu, use root instead of *

```
* - nofile 1048576
* - memlock unlimited
* - fsize unlimited
* - data unlimited
* - rss unlimited
* - stack unlimited
* - cpu unlimited
* - nproc unlimited
* - as unlimited
* - locks unlimited
* - sigpending unlimited
* - msgqueue unlimited
```

Kernel Configuration

swap

- For Cassandra, swapping is a very bad event
- You are better having a node go down than limp along swapping
- To thoroughly disable swap:
 - Turn off swap for the current kernel process
 - Remove swap entries from fstab
 - Change the swappiness setting

Kernel Configuration

Turning off swap for the current kernel image

- You can check the current list of swap devices `swapon -s`
- You can turn off swap without rebooting
- This command will not persist (i.e., will not survive a reboot):

```
swapoff -a
```

Kernel Configuration

Disabling swap in fstab

- This will permanently disable the swap devices after a reboot
- In /etc/fstab, comment out any rows that are swap devices
- Here's a sed script to do this:

```
sed -i 's/^.*swap\)/#\1/' /etc/fstab
```

Kernel Configuration

Changing the swappiness setting

- To look at the current swappiness setting `cat /proc/sys/vm/swappiness`
- This value has a range of 0-200, (0 is low-swap and 200 is high-swap)
- To make sure your kernel disables swapping after a reboot,
edit `/etc/sysctl.conf`
- Change or add a line to set `vm.swappiness`
- Use `sysctl -p` to get the kernel to reload the changes made
to `/etc/sysctl.conf`

vm.swappiness = 0

Kernel Configuration

Changing network kernel settings

- Here are some recommended settings to tune network performance
- Add or change these settings in /etc/sysctl.conf
- Remember, sysctl -p will cause the kernel to load the settings from /etc/sysctl.conf

```
net.ipv4.ip_local_port_range = 10000 65535
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.netdev_max_backlog = 2500
net.core.somaxconn = 65000
```

Hardware Considerations

Hardware Selection

Hardware Selection

CHOOSE the right balance of the following

- Persistent storage type
- Memory
- CPU
- Number of nodes
- Network

Hardware Selection

Persistent Storage

- Avoid:
 - SAN storage
 - NAS devices
 - NFS

Hardware Selection

Persistent Storage

- SSD is absolutely the preferred solution for every workload
- Provides extremely low-latency response times for random reads while supplying ample sequential write performance for compaction operations
- Has no moving parts to fail

Hardware Selection

Persistent Storage

- HDD is by far the cheapest storage available
- They still show up in new systems for a few reasons:
 - Hardware vendors mark SSDs up by 100-2000%
 - 7200RPM SATA is around \$0.05/GB while a Samsung 1TB MLC is \$0.50
 - High capacity is still easier to come by

Hardware Selection

Persistent Storage

- When do HDDs make sense?
 - Usually never
- Exceptions might be:
 - Mostly-write workloads using DateTieredCompactionStrategy (i.e., limited seeking)
- If you have to use HDD, get as much RAM as possible to increase caching

Hardware Selection

Memory

- For both dedicated hardware and virtual environments:
 - Production: 16GB to 64GB; the minimum is 8GB.
 - Development in non-loading testing environments: no less than 4GB.
- More memory means
 - Better read performance due to caching
 - Memtables hold more recently written data

Hardware Selection

CPU

- Cassandra to DataStax Enterprise is highly concurrent and uses as many CPU cores as available
- Production environments:
 - For dedicated hardware, 16-core CPU processors are the current price-performance sweet spot
- Development in non-loading testing environments:
 - For dedicated hardware, 2 to 4-core CPU processors
 - For virtual environments, 2 to 4-core CPU processors

Hardware Selection

Network

- You should bind your OS interfaces to separate Network Interface Cards (NIC).
- Recommended bandwidth is 1000 Mbit/s (gigabit) or greater.
- Native protocols use the `native_transport_address`.
- Cassandra's internal storage protocol uses the `listen_address`.

Cloud

Provisioning

- There are easy one click deployment patterns for Cassandra and DSE development and test deployments in most of the major public clouds
- For production use OpsCenter LCM in 6.0 + for provisioning and configuration management. You will have to self provision the instances, OpsCenter does installation and config
- If you don't have access to OpsCenter there are Chef / Ansible recipes / playbooks for DSE out there

Selecting Your Instance Type

Disks

- Ephemeral is fastest
- The world is moving toward elastic / non ephemeral volumes
- If an ephemeral volume fails, terminate and get a fresh box
 - Tell-tale sign of bitrot / corrupted sstables

Selecting Your Instance Type

- AWS
 - GP2 or IO2 EBS if you can't afford ephemerals
 - Get big ones, IO latencies get better as the disk volumes increase up to 4TB
 - Get at least 3 TB's on the GP2's
- Google
 - Their elastic volumes always have the same tight latencies
 - Some customer have observed tests that indicate pd-ssd is actually faster than local-ssd
- Azure
 - Today mostly ephemeral is used
 - Premium storage is rolling out pretty fast

Selecting Your Instance Type

CPU Considerations

- Hyperthreads
 - You don't get a real CPU most of the time, tune accordingly
- CPU steal / noisy neighbors
 - If you see CPU steal, terminate your box and get a new one, it means you have a noisy neighbor

Selecting Your Instance Type

Networking considerations

Multi DC options include

- AWS
 - Multi-regions can't communicate via internal IP, you need to figure out networking yourself or use limited number of Security rules
- Azure
 - Vnet 2 vnet — slow, good for small workloads
 - Azure has options for fatter pipes one of which can even go to your physical DC for hybrid deployments
 - Use public IPs
 - Do not use VPN gateways
 - Express Route might be viable
- Google
 - Flat network - No config necessary, it's great

Selecting Your Instance Type

- Enhanced networking
 - AWS thing you need to turn-on on some instance types to get performance

Selecting Your Instance Type

Security

- AWS
 - Volume encryption is available for EBS
- Google
 - Largely secure by default. Everything goes through their multifactor auth

Storage

Storage

SAN

- Just say no!
- Not recommended for on premises deployments
- Difficult and PRICEY architecture to use with distributed databases



Storage

- SAN ROI does not scale along with that of Cassandra
- SAN generally introduces a bottleneck and single point of failure
- Added latency for all operations
- Heap pressure is increased
- Potential network saturation and network availability problems



Storage

A few Cassandra specific performance issues (with shared storage)

- Atrocious read performance
- Potential write performance issues
- System instability (nodes appear to go offline and/or are "flapping")
- Client side failures on read and/or write operations
- Flushwriters are blocked
- Compactions are backed up
- Nodes won't start
- Repair/Streaming won't complete

Storage

Time	DEV	tps	rd_sec/s	wr_sec/s	avgrq-sz	avgqu-sz	await	svctm	%util
2:50:01 AM	dev253-2	108.24	2346.71	766.6	28.76	335.8	3102.31	13.52	146.3
3:00:06 AM	dev253-2	4.68	1020.93	3.15	218.8	5.66	1197.68	399.21	186.85
3:10:42 AM	dev253-2	8.48	1474.21	4.5	174.36	12.35	787.49	214.57	181.97
3:20:14 AM	dev253-2	3.52	829.41	0.66	235.56	4.23	2959.08	462.27	162.89
5:00:13 AM	dev253-2	167.01	1571.72	1256.24	16.93	511.98	3040.23	20.28	338.7
5:10:29 AM	dev253-2	126.06	808.5	972.98	14.13	3655.31	28974.72	31.2	393.36
5:20:02 AM	dev253-2	205.52	6431.41	1419.87	38.2	53.73	294.44	5.89	121.12
5:30:08 AM	dev253-2	23.39	1027.56	150.49	50.36	23.38	1126.18	94.39	220.8

Storage

NAS

- Storing SSTables on a network attached storage (NAS) device is not recommended
- Network related bottlenecks resulting from high levels of I/O wait time on both reads and writes
- These are due to:
 - Router latency
 - The Network Interface Card (NIC) in the node
 - The NIC in the NAS device

Storage NFS

- Has exhibited inconsistent behavior with its abilities to delete and move files
- This configuration is not supported in Cassandra and it is not recommended to use

Hadoop Style Nodes

Hadoop Style Nodes

= Fat Nodes

- An expensive way to make your Cassandra clusters inefficient
- Hadoop style nodes are bias heavily towards large and slow storage and low memory
- Optimized for slow analytic workloads and not fast transactions



Security Considerations

Authentication and Authorization

Authentication and Authorization

Disclaimer

- Full Cassandra security is an in-depth topic
- This is an introductory section
 - Uses a simplified approach
 - Familiarizes you with the security concepts and concerns
- Best practices is a broad subject beyond the scope of this course
 - See the DS410 course for an in-depth treatment
 - Also, consult with your DataStax Solutions Engineer (SE)

Authentication and Authorization

What do these terms mean?

- Authentication controls who can access the cluster
- Authorization controls what they can do on the cluster

Authentication

Basics

- By default, authentication is disabled
- When enabled, client programs must supply a username and password
 - Example: `cqlsh -u cassandra -p cassandra`
 - Driver also supports authentication
- Enable authentication in `dse.yaml`

Authentication

DseAuthenticator options

- Configure authentication in dse.yaml
- Three Schemes
 - Internal
 - LDAP*
 - Kerberos*

dse.yaml

```
authentication_options:  
  enabled: false  
  default_scheme: internal  
  allow_digest_with_kerberos: true  
  plain_text_without_ssl: warn  
  transitional_mode: disabled  
  other_schemes:  
    scheme_permissions: false
```

(* covered in DS410)

Authentication

Internal scheme

- Enable in dse.yaml
- Restart the node(s)
- Login as `cassandra` with a password of `cassandra`
- The `cassandra` user is a superuser - has all permissions
 - Change the default `cassandra` password
 - `ALTER USER cassandra WITH PASSWORD 'newpassword';`
- Cassandra stores the credentials in the `sys_auth` keyspace
 - Losing data in this keyspace could be disastrous!
 - `ALTER KEYSPACE "system_auth" WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'dc1' : 2};`

Authentication

Cassandra users

```
CREATE TABLE system_auth.roles (
    role text PRIMARY KEY,
    can_login boolean,
    is_superuser boolean,
    member_of set<text>,
    salted_hash text
)
```

Authentication

Cassandra users

- Note the fields in the `roles` table:
 - `role` - the role name
 - `can_login` - this is what separates a user from a true role
 - `is_superuser` - full permissions
 - `member_of` - roles this role has been granted
 - `salted_hash` - password hash

Authentication

Role operations

```
CREATE ROLE fred WITH PASSWORD = 'Yabadabado' AND LOGIN = true;  
LIST ROLES;  
DROP ROLE fred;
```

Authentication

Authentication Best Practices

- Create a second superuser role
- Change the default cassandra password and forget it
- Be sure to replicate the system_auth keyspace

Authorization

Grant permissions to roles

```
GRANT SELECT ON killr_video.user_by_email TO fred;
```

This allows user fred to perform SELECT queries on killr_video.user_by_email

Authorization

Permissions

- **ALTER** - ALTER KEYSPACE, ALTER TABLE, CREATE INDEX, DROP INDEX
- **AUTHORIZE** - GRANT, REVOKE
- **CREATE** - CREATE KEYSPACE, CREATE TABLE
- **DROP** - DROP KEYSPACE, DROP TABLE
- **MODIFY** - INSERT, DELETE, UPDATE, TRUNCATE
- **SELECT** - SELECT

Authorization

Grant roles to roles

```
CREATE ROLE db_accessors;  
GRANT SELECT ON killr_video.user_by_email TO db_accessors;  
GRANT db_accessors TO fred;
```

db_accessors becomes a true role that may have complex permissions

Authorization

Revoke permissions

```
REVOKE SELECT FROM db_accessors;
```

This will revoke this permission from the role and all those who are granted this role

Exercise – Authentication and Authorization

SSL

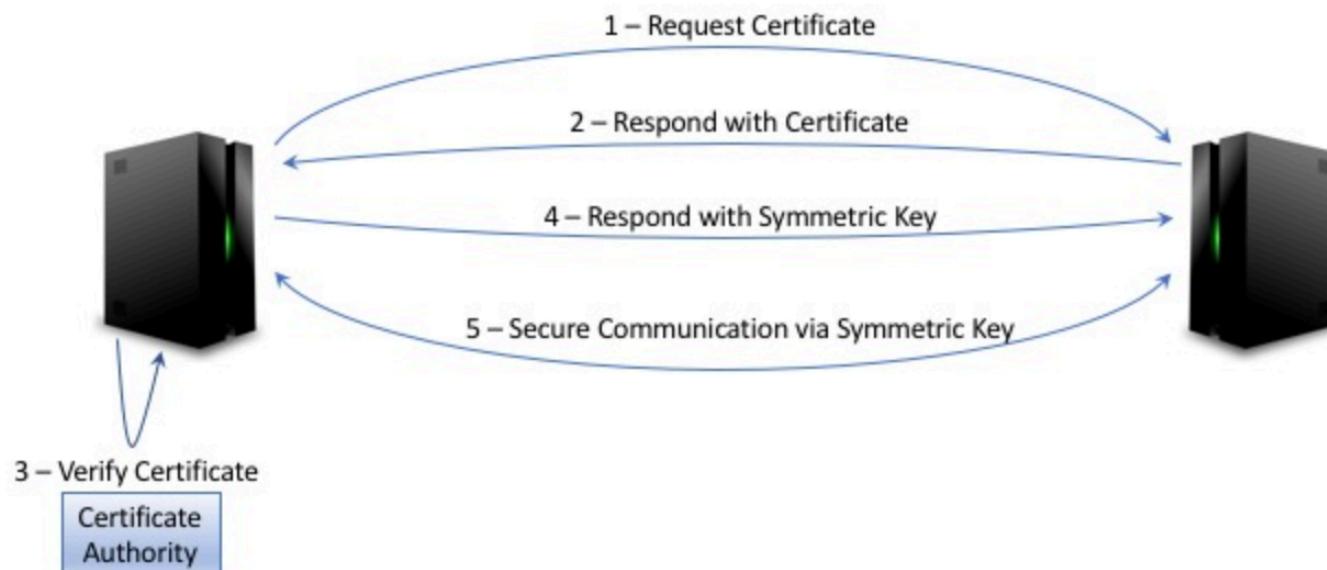
SSL

Using SSL, Cassandra encrypts cluster communication

- Between nodes in the cluster
- Also between clients (like cqlsh) and nodes in the cluster
- Again, consult DS410 and your DataStax Solutions Engineer (SE) for comprehensive SSL coverage

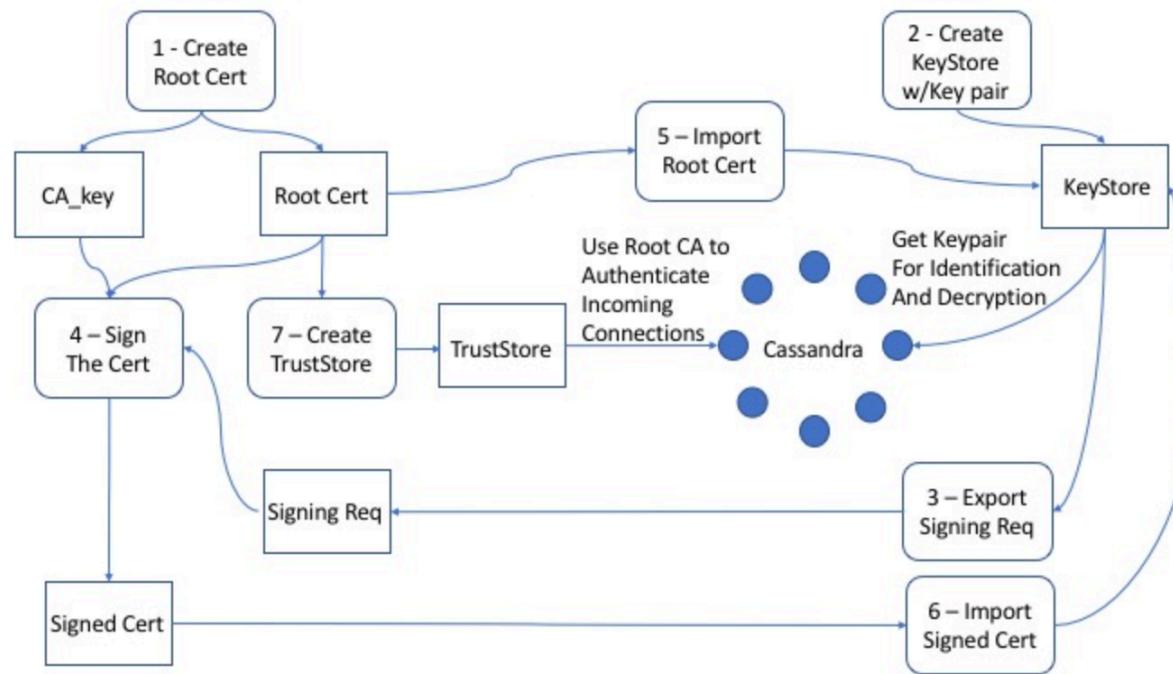
SSL

How does SSL work?



SSL

8 Steps for SSL set-up (node-to-node)



SSL

SSL considerations

- Secure the keystore - since this contains your private key
- Set up the firewall so that only nodes can talk to each other on the `listen_port`
- Protect the root CA private key - this should NOT be distributed to the nodes
- Enable `require_client_auth`, otherwise programs could spoof being a node

Exercise - SSL

OpsCenter and LifeCycleManager

OpsCenter

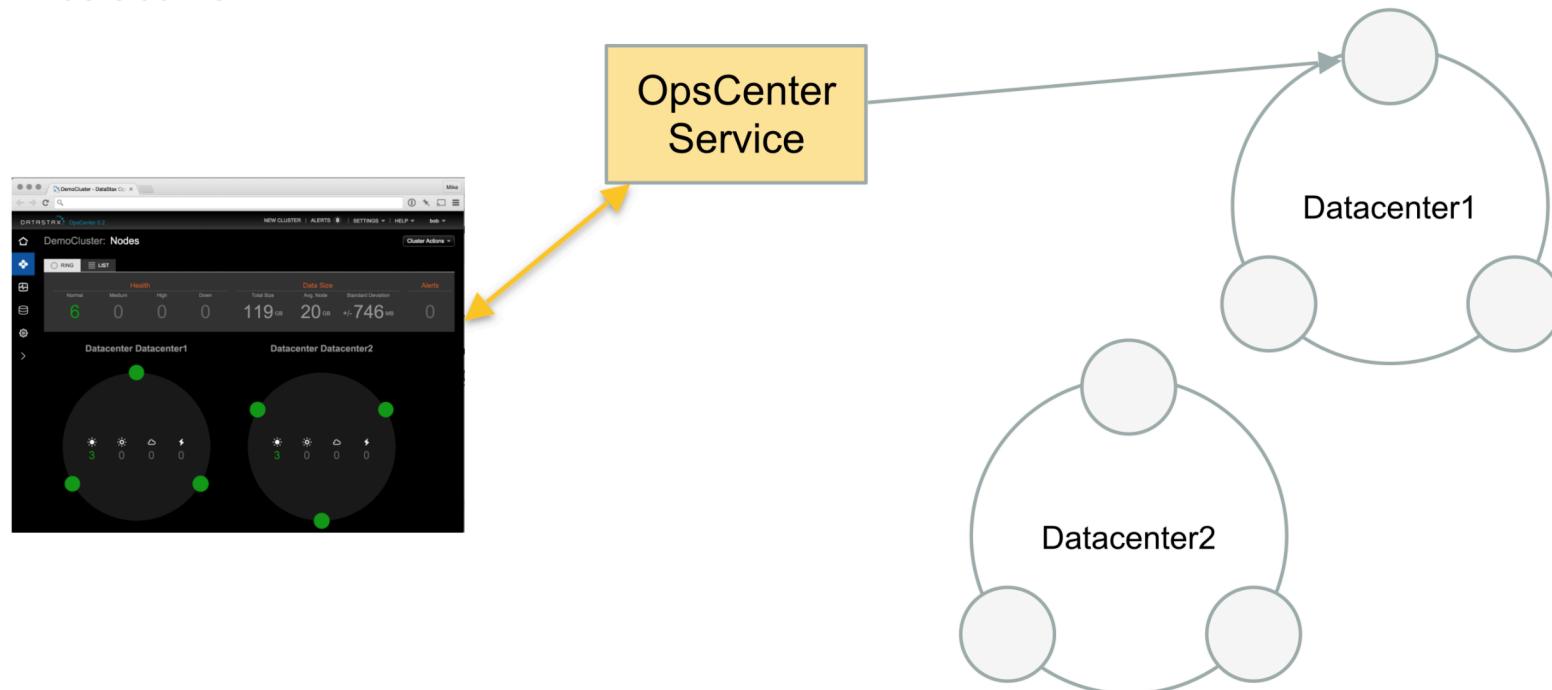
OpsCenter

What is it?

- Browser-based DSE cluster tool for:
 - Configuring
 - Monitoring
 - Managing
- Two major components tied together:
 - Life Cycle Manager (LCM) – mostly configuration and deployment
 - OpsCenter Monitoring – Monitoring and management

OpsCenter

Architecture



OpsCenter

Configuration tasks

- Configure and deploy clusters
- Expand/Shrink clusters to adjust capacity or performance
- Perform software upgrades
- Configure security



The screenshot displays the DataStax OpsCenter web interface, which is a central management tool for DataStax clusters. It includes several key components:

- Job Overview:** Shows a summary of a recent job named "TestCluster" with ID 59e14002-6310-4dd3-bf67-5f46c0d2fb0b. It details the creation and modification times, event types (Milestone - END and Milestone - START), and descriptions (job complete and job started...).
- Nodes Table:** A table showing the status of three nodes: dc210-node1-1, dc210-node1-2, and dc210-node1-3, all of which are marked as successful. The table includes columns for Status, Node, Seed, Rack, and Datacenter.
- Monitoring Dashboard:** A grid of six charts providing real-time data on cluster health, storage capacity, write requests, latency, and OS load across different data centers.
- Keyspace Management:** A sidebar showing the current state of keyspace management, including 5 Keyspaces and 16 Tables. It lists keyspaces like dse_leases, dse_perf, dse_system_local, OpsCenter, and solr_admin, each with their respective table counts.
- Cluster Map:** A circular map showing the physical locations of the DataStax clusters.

OpsCenter

Configuration with Life Cycle Manager



Lifecycle Manager: Clusters

Clusters	Datacenters	Nodes
TestCluster	DC1	ds210-node1-1 ds210-node1-2 ds210-node1-3

DC1

Datacenter Details

ID	ba29d215-541d-4742-88d1-d664c3290037
Workload	Cassandra
Last Job Status	Success
Config Profile	Defaults to the cluster
SSH Credentials	Defaults to the cluster
Repository	Defaults to the cluster
SSH Port	Defaults to the cluster
Created On	3/7/2018, 7:25PM UTC
Modified On	3/7/2018, 7:25PM UTC

OpsCenter

Cluster Monitoring

- Dashboard
- Metrics
- Alerts
- Trend/capacity analysis
- Performance service



OpsCenter

Monitoring with the Node view

OpsCenter 6.5.0.EAP1

TestCluster: Nodes 0 Alerts

DataStax Enterprise Server 6.0.0 Cluster Actions

Health

- Normal: 3
- Medium: 0
- High: 0
- Down: 0
- Unknown: 0

Data Size

Total Size	420 KB
Avg. Node	140 KB
	+/- 11 KB

Services

Repair: Not running

Backup: No Scheduled Backups

Recent: 0 successful
0 failed

Performance: Table Metrics, Thread Pool Stats, Slow Query Log

Best Practices: Rules passed: 0, Rules failed: 0

Agents

NAME **DATACENTER** **RACK** **HTTP** **STOMP** **JMX** **STORAGE DSE** **MONITORED DSE** **LAST KNOWN VERSION** **LAST INSTALL STATUS**

Healthy (3 Agents)

Agents are healthy.

The following agents are installed and connected.

ds210-node1	DC1	Rack1	✓	✓	✓	✓	✓	✓	✓	✓	6.5.0.EAP1	Error
ds210-node1	DC1	Rack1	✓	✓	✓	✓	✓	✓	✓	✓	6.5.0.EAP1	Error
ds210-node1	DC1	Rack1	✓	✓	✓	✓	✓	✓	✓	✓	6.5.0.EAP1	Error

OpsCenter

Monitoring with the dashboard



OpsCenter

Management Services

- Backup & Restore Service
- NodeSync Service
- Repair Service
- Best Practices Service
- Capacity Service
- Performance Service

SERVICE	DESCRIPTION
Backup Service	Schedule and restore backups (local or cloud). Also supports commit log backup for point in time restore.
NodeSync Service	NodeSync is a data validation and syncing service managed by DataStax Enterprise. NodeSync must be configured in DSE. NodeSync status can be viewed in OpsCenter after enabling keyspaces and tables.
Repair Service	Manage automatic repairs of all clusters; keeps data consistent on all nodes without impacting performance.
Best Practice Service	Automatically scans clusters using a set of pre-defined best practice rules and provides expert advice on resolving issues.
Capacity Service	Automatically collects statistical data for trend analysis and forecasts future trends.
Performance Service	Combines OpsCenter metrics with CQL-based diagnostic tables populated by the DSE Performance Service to help understand, tune, and optimize cluster performance. Visually enable the performance objects and analyze the results within OpsCenter. Performance Service is available in DataStax Enterprise version 4.6 and higher.

Backup Service

Visual backup management

- Automatic replication doesn't preclude backups
- Backup & Restore on distributed systems is hard
- Enterprise-class visual backup service guards against data loss on managed database clusters
- Removes need for building custom scripts and tools

Backup Service

Features

- Simple central UI-based
- Schedule or ad hoc
- Destinations include:
 - Local server
 - Amazon S3
 - Custom location
- Compresses data
- Configurable retention policies
- Includes alerts and reporting

Create Backup

Type: Run Now Schedule

Keyspaces: *

Alert on Failure: True False

Current Data Size: 81 MB

Location:

Staging Directory: /var/lib/datastax-e

This directory will be created. Learn how to manage documentation.

▶ Advanced Options

Add Location

Location:

S3 Bucket: *

AWS Key: *

AWS Secret: *

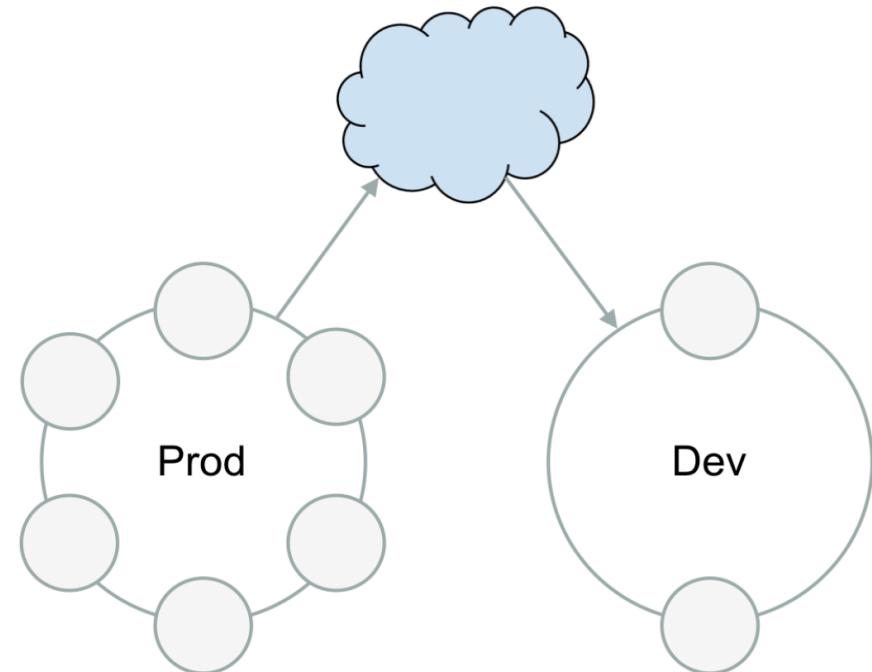
Options:
 Throttle S3 transfer rate at MB/s
 Enable compression
 Enable S3 server-side encryption

Note: Remote locations store only one copy of SSTables. [Read more ↗](#)

Backup Service

Restore

- From:
 - Local file system
 - Cloud provider
- All or specified Keyspaces/tables
- To original cluster or different one
- Point in Time (PIT) Restore



NodeSync

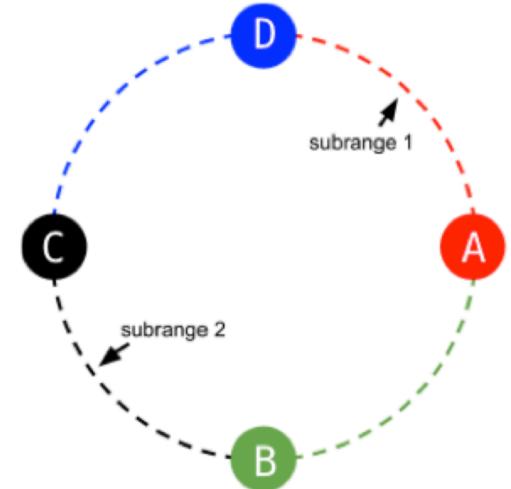
Automatic consistency management

- NodeSync is configured on keyspaces and tables
- OpsCenter lets you monitor the NodeSync performance
- Preferred over Repair Service, but these are not exclusive

Repair Service

Features

- Runs in the background
- Works on small chunks to limit performance impact
- Continuously cycles within a specified time period
- Can work on sub-ranges or incremental
- Can run in parallel



Best Practice Service

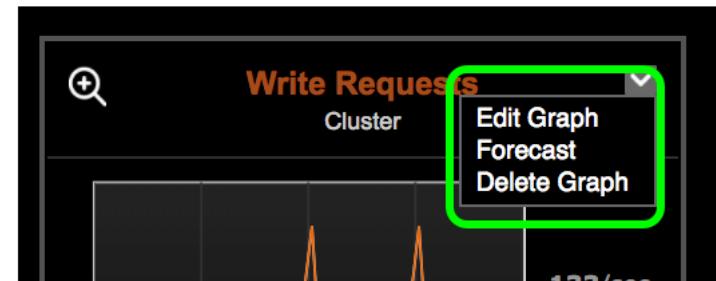
- Codifies DataStax expertise in best practices
- Periodically audits your configuration and behaviors
 - Detects violations of best practices
 - Suggests how to fix violations and improve your system
- Finds threats to:
 - Security
 - Performance
 - Availability
- Very easy to use

cluster0: Services / Best Practice			Cluster Actions
Expand All Collapse All			PASS FAIL 1 2
Replication Advisor	INFO	✓ Passed	1 PASS 1 FAIL 2 OF 3 ACTIVE
Check High RF	INFO	X Failed	Every Minute Configure Turn rule off
SimpleSnitch usage found	INFO		Every Minute Configure Turn rule off
SimpleStrategy keyspace usage found	MEDIUM		Turn rule on
Config Advisor			0 PASS 1 FAIL 1 OF 4 ACTIVE
Backup Advisor			0 PASS 0 FAIL 0 OF 1 ACTIVE
Security Advisor			0 PASS 0 FAIL 0 OF 4 ACTIVE
Search Advisor			0 PASS 0 FAIL 0 OF 7 ACTIVE
OS Advisor			0 PASS 0 FAIL 0 OF 3 ACTIVE
Analytics Advisor			0 PASS 0 FAIL 0 OF 1 ACTIVE
Performance Advisor			0 PASS 0 FAIL 0 OF 3 ACTIVE
Network Advisor			0 PASS 0 FAIL 0 OF 1 ACTIVE
OpsCenter Config Advisor			0 PASS 0 FAIL 0 OF 1 ACTIVE

OpsCenter

Capacity Service

- Used to forecast metrics
- Works for most of the graphs on the dashboard
- Easy to enable
 - In the top-right corner of the graph
 - Click the dropdown
 - Select Forecast
 - Choose a range from the pop-up and see the trend



Performance Service

Overview

- Collects key performance metrics to quickly troubleshoot a database cluster's performance
- Makes recommendations for optimizing cluster performance
- Analyzes Query, Table & Cluster specific metrics
- Correlates different metrics and provides custom recommendations to address specific issues
- Eliminates the need for custom scripting and scheduling to detect problem nodes

Performance Service

Slow queries

- Identify Slow Queries
- Custom recommendations
- Display contextual alerts
- Visual Query tracing

Slow Queries			
DURATION	SOURCE	TABLES	COMMANDS
1498 ms	msrinivasan-rmbp15.local	test.wide	select * from test.wide where key in ('1','2');
4 ms	msrinivasan-rmbp15.local	amazon.clicks	SELECT * FROM amazon.clicks WHERE soir_query='{"q":"asin:","fq":"+(geofilt pt=[-37.546270,-121.988572] shield=location d=1115)" limit 100000;
4 ms	msrinivasan-rmbp15.local	test.wide	select * from test.wide where key='1' and key2='300009';

Slow Queries Alerts	
2 minutes	Local Read Latency at 1,273.1 ms/op for test.wide -

Recommendations	
Use prepared statements	
Use prepared statements for your queries.	


```
select * from test.wide where key in ('1','2');
```

Consistency: One Keyspace: test Coordinator: Auto

TIME	SOURCE	DESCRIPTION	ELAPSED (μs)	THREAD
1:47:12.212	msrinivasan-rmbp15.local	Parsing select * from test.wide where key in ('1','2');	52	SharedPool-Worker-2
1:47:12.212	msrinivasan-rmbp15.local	Preparing statement	120	SharedPool-Worker-2
1:47:12.212	msrinivasan-rmbp15.local	Executing single-partition query on wide	599	SharedPool-Worker-1
1:47:12.212	msrinivasan-rmbp15.local	Acquiring sstable references	626	SharedPool-Worker-1

© 2018 DataStax. Use only with permission. • DS210: DataStax Enterprise Operations with Apache Cassandra™



Performance Service

Table stats

- Diagnose Table performance
- Custom recommendations
- Cluster level & node drill down

OVERVIEW | SETTINGS

← Back to Overview

Table Metrics

Table Details For: dse leases leases

KEYSPACE	TABLE	WRITE LATENCY	READ LATENCY
dse_perf	write_latency_histograms_global	0.06 ms/op	0.00 ms/op
dse_perf	write_latency_histograms_ks	0.06 ms/op	0.00 ms/op
dse_perf	range_latency_histograms_global	0.06 ms/op	0.00 ms/op
dse_perf	thread_pool_messages	0.04 ms/op	0.00 ms/op
dse_perf	read_latency_histograms	0.04 ms/op	0.00 ms/op
dse_perf	partition_size_histograms_summary	0.02 ms/op	0.00 ms/op
dse_perf	partition_size_histograms	0.02 ms/op	0.00 ms/op
dse_perf	net_stats	0.02 ms/op	0.00 ms/op
dse_perf	read_latency_histograms_summary	0.02 ms/op	0.00 ms/op
dse_perf	write_latency_histograms_summary	0.02 ms/op	0.00 ms/op
dse_perf	sstables_per_read_histograms_summary	0.02 ms/op	0.00 ms/op
dse_perf	cell_count_histograms_summary	0.02 ms/op	0.00 ms/op

Table Metrics Alerts
 No active alert events. The cluster is operating within the configured thresholds for all alert metrics.

Recommendations
Compaction Strategy
 Choose the compaction strategy that best fits your data and environment.
http://docs.datastax.com/en/scroll#tabProp_more

Table Details

2018-01-11 11:35 AM | to 2018-01-11 11:55 AM | [Update](#) | [Current](#)

Graph Scale: [20m](#) | [Hour](#) | [Day](#) | [Week](#) | [Month](#)

Nodes

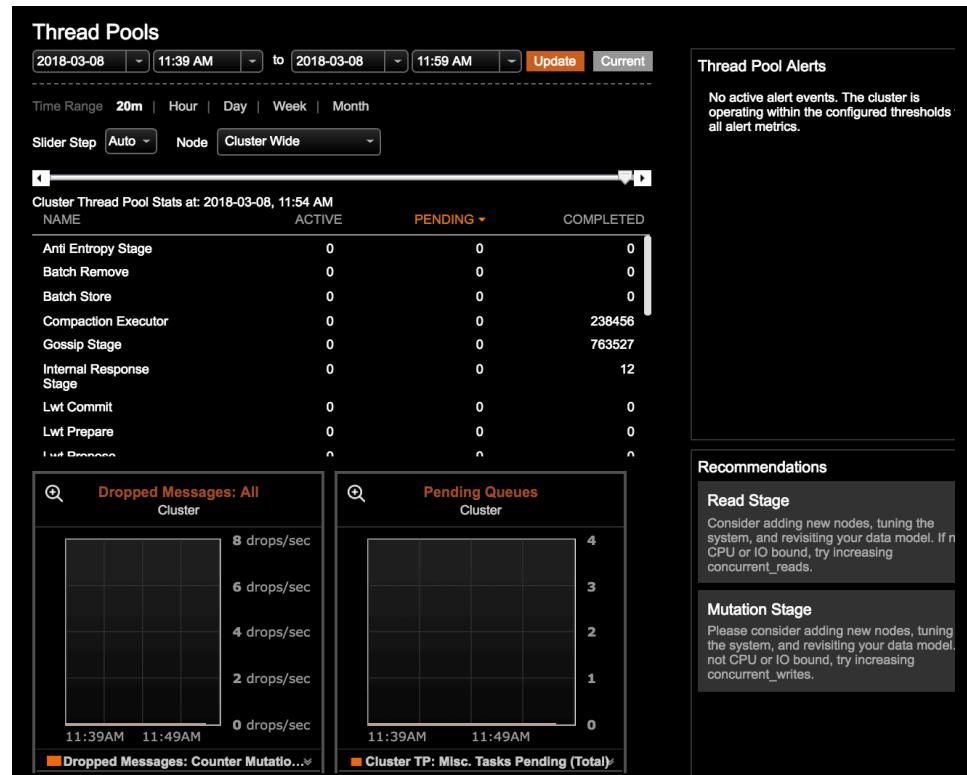
Nodes	TBL: Local Read...	TBL: Local Write...	TBL: Total Disk ...	TBL: Cell Count...
■ Select All	Cluster	Cluster	Cluster	Cluster
	15 ms...	15 ms...	4 by...	4 cells
	7 ms...	7 ms...	3 by...	3 cells
	3 ms...	3 ms...	2 by...	2 cells
	1 ms...	1 ms...	1 by...	1 cells
	0 ms...	0 ms...	0 by...	0 cells
	11:3...11:4...	11:3...11:4...	11:3...11:4...	11:3...11:4...

[Cluster dse leases...](#) | [TBL: Read...](#) | [TBL: Write...](#) | [TBL: Total Disk...](#) | [TBL: Cell Count...](#)

Performance Service

ThreadPool stats

- Investigate Thread Pools metrics
- Historical Tracking
- Eliminates custom scripts



Cluster Monitoring

Alerts

- OpsCenter can send alerts based on OpsCenter daemon activity events
- Alerts can be
 - Emails – sent using and SMTP server
 - HTTP POSTs – to a specified URL
 - SNMP – to an enterprise monitoring system (e.g., Nagios)

Exercise - OpsCenter