Cassandra Performance Tuning

Apache Cassandra:
Environment Tuning

Learning objectives

- **Discuss environment tuning outside of Cassandra**
- Identify tools and tuning strategies for the JVM
- Discuss working with page cache
- Identify tools to monitor the CPU

Question

What are the most common bottlenecks in Cassandra?

Common Bottlenecks

- Inadequate hardware
- Insufficient or incorrect memory cache tuning
- Poorly configured JVM parameters
- High CPU utilization

Question

What is the best way to cope with inadequate node hardware in a Cassandra cluster?

How do you cope with inadequate node hardware?

- Upgrade the hardware
- Analyze your operating system

Question

What are some of the technologies upon which a Cassandra node depends?

Java technologies and tools

- Java—an object-oriented, current and class-based programming language.
- JVM—a process virtual machine that can operate Java bytecode.
- JNA—(Java Native Access) an API for allowing Java programs access to shared libraries.
- Heap—an area of memory used for dynamic memory allocation.
- JMX—(Java Management Extensions) supplies tools for managing and monitoring resources.

Learning objectives

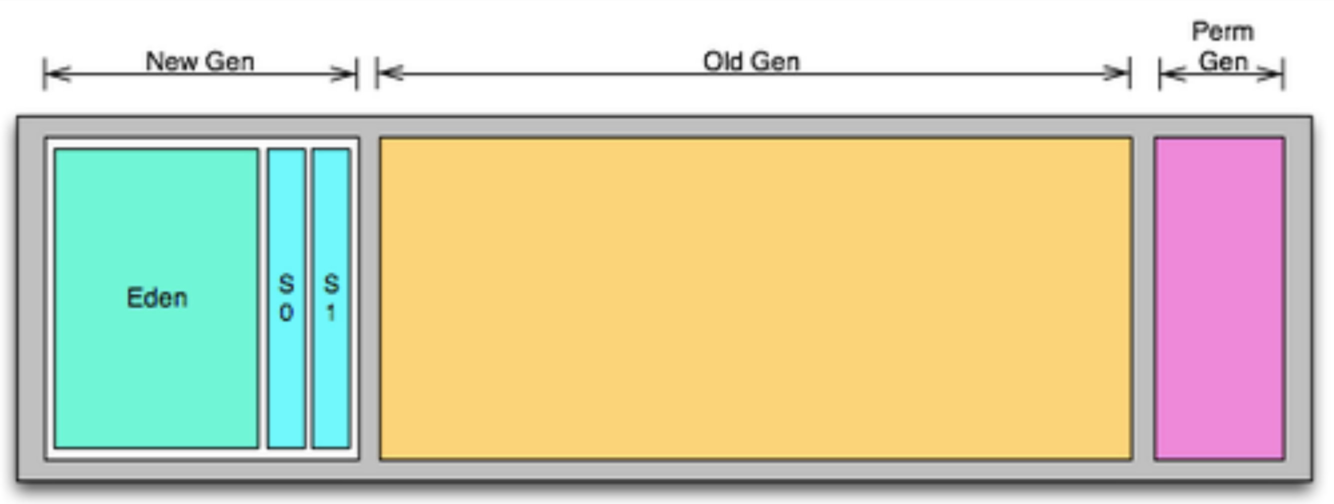
- Discuss environment tuning outside of Cassandra
- **Identify tools and tuning strategies for the JVM**
- Discuss working with page cache
- Identify tools to monitor the CPU

JVM and Garbage Collection (GC)

- Recall that 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 larger "holes" 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

JVM available memory

- New generation—broken up into eden and survivor spaces
  - When eden fills with new object, minor garbage collection occurs
  - Application pauses when ParNew collector is run
- Old generation
  - Contains objects that have survived long enough to not be collected by gc
  - When a certain percentage (75% default) is full, CMS collector is run



Graphic by Blake Eggleston

JVM heap options and Cassandra performance

- Setting *MAX_HEAP_SIZE* to not more than 8GB
  - Large heaps can introduce GC pauses that lead to latency
  - On the other hand, different workloads can justify different settings
    - SHIFT, for instance, found that using 10GB improved their performance
- Setting to *HEAP_NEWSIZE* to 100MB per core
  - 8 cores would mean 800MB
  - But again, different workloads may react differently
- Interesting to set these values to low settings and high settings and compare the results
  - In the lab for this section, you will do just that, so you can see for yourself.

Using *nodetool tpstats* to discover JVM issues

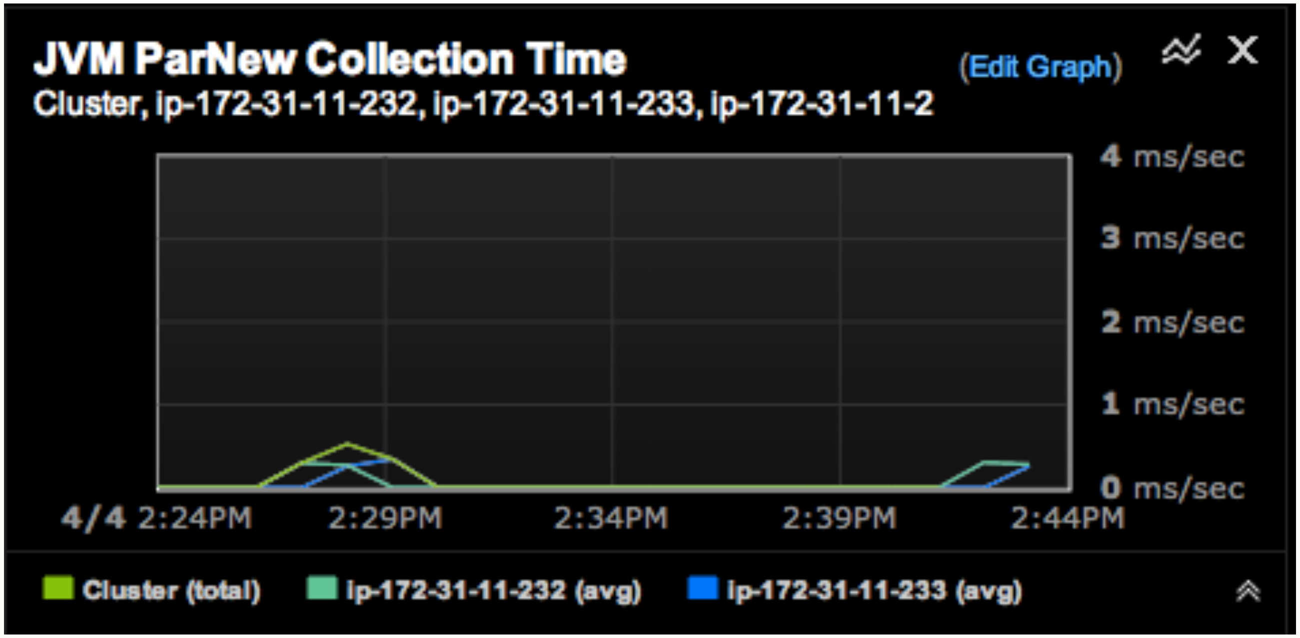
- Displays the number of active, pending, and completed tasks for each of the thread pool that Cassandra uses for stages of operations

```
ubuntu@ip-172-31-8-219:~$ nodetool -h node0 tpstats
Pool Name                    Active   Pending      Completed   Blocked  All time blocked
ReadStage                         0         0        4681361         0                 0
RequestResponseStage              0         2       11269852         0                 0
MutationStage                     0         0       18235083         0                 0
ReadRepairStage                   0         0         111456         0                 0
ReplicateOnWriteStage             0         0              0         0                 0
GossipStage                       0         0        4775637         0                 0
AntiEntropyStage                  0         0         187096         0                 0
MigrationStage                    0         0             70         0                 0
MemoryMeter                       0         0            342         0                 0
MemtablePostFlusher               0         0          82753         0                 0
FlushWriter                       0         0          17698         0                 1
MiscStage                         0         0          53456         0                 0
PendingRangeCalculator            0         0              3         0                 0
commitlog_archiver                0         0              0         0                 0
AntiEntropySessions               0         0           3341         0                 0
InternalResponseStage             0         0          53459         0                 0
HintedHandoff                     0         0             20         0                 0

Message type           Dropped
RANGE_SLICE                  0
READ_REPAIR                  0
PAGED_RANGE                  0
BINARY                       0
READ                         0
MUTATION                     0
_TRACE                 2690912
REQUEST_RESPONSE             0
COUNTER_MUTATION             0
```

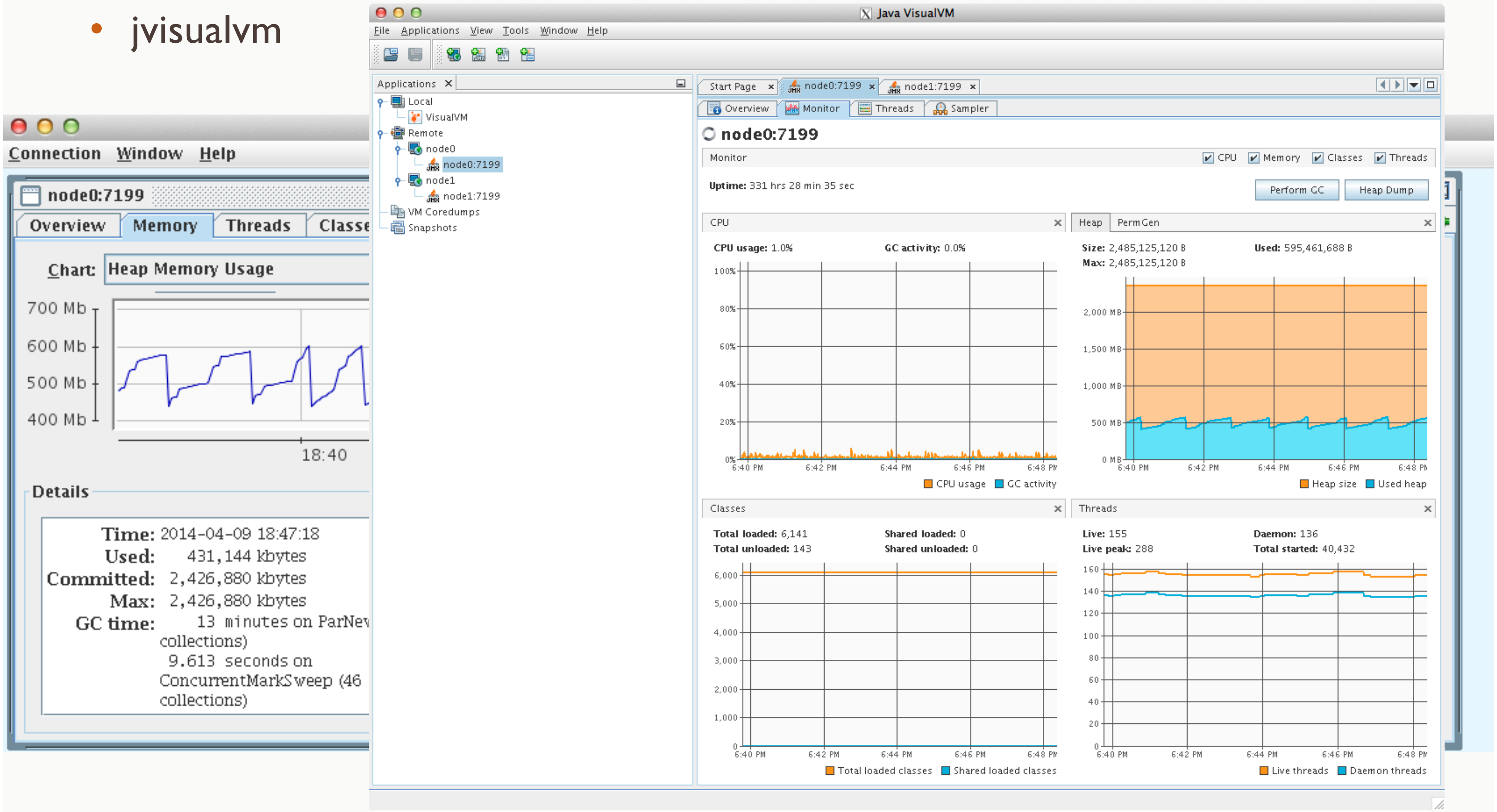
Using *OpsCenter* to explore JVM issues

- *OpsCenter* can provide visual insight into the JVM performance with the following graphs:
  - JVM Collection Count for both ParNew and CMS
  - JVM Collection Time for both ParNew and CMS
  - Heap Max, Heap Used, Heap Committed
  - Number of pending tasks – drill down for more information



Using JMX clients to explore JVM issues

- A couple of visual tools that can be used to "watch" heap action
  - jconsole
  - jvisualvm



Demo 1: Viewing JVM heap memory using jconsole and OpsCenter

Exercise 1: Tuning JVM heap

Learning objectives

- Discuss environment tuning outside of Cassandra
- Identify tools and tuning strategies for the JVM
- **Discuss working with page cache**
- Identify tools to monitor the CPU

What is page caching?

- Accelerates access to files on non-volatile storage
  - When reading and writing to hard disk, Linux also stores data in unused areas of memory.
  - These act as a cache, and allow that data to be quickly read from memory if they are accessed again.
- Page cache is sometimes called Buffer cache
  - At one time, page cache and buffer cache were separate, but now they have been combined.
- The command *free –m* can be used to discover how much RAM is used for page cache

```
ubuntu@ip-172-31-8-219:~$ free -m
             total       used       free     shared    buffers     cached
Mem:          7450       7125        324          0        125       6583
-/+ buffers/cache:        417       7033
Swap:            0          0          0
```

How does Cassandra utilize page caching?

- Linux will cache most recently used disk pages.
- The more RAM you have, the more data that will be held in memory.
  - Once memory is filled up, the most stale data will be overwritten.
- mmap makes file access very efficient for C*
  - Using file mapping, the kernel maps your program's virtual pages directly onto the page cache.
- Page cache makes writes more efficient
  - OS flushes page cache to disk.
- Page cache makes reads more efficient.
  - OS can access reads from memory more quickly than from disk.
- How can page caching improve performance?
  - More RAM, more page cache that is available.

How do you triage root cause for out of memory (OOM) errors?

- Writing to partitions with inadequate timeout
  - Writes back up in memory if the throughput is too slow.
  - Increase the disk throughput with local disk and/or SSDs.
- Reading from partitions with lots of deletes, including TTLs
  - Reads require ALL the data, including the deleted data, to be read into memory to get the queried data.
  - If you must use frequent TTLs and deletes, consider whether or not you can cluster these to fewer rows – deleting a row can solve many of the issues associated with this.
- Client-side joins
  - Too much data must be resident on memory to accomplish the joins.
  - Try to eliminate client-side joins as much as possible – consider redesigning your schemas.

Exercise 2: Working with the Linux page cache

Learning objectives

- Discuss environment tuning outside of Cassandra
- Identify tools and tuning strategies for the JVM
- Discuss working with page cache
- **Identify tools to monitor the CPU**

What operations benefit from better CPUs?

- Garbage collection is CPU-intensive
  - Add more CPUs or faster CPUs, and garbage collection will run faster.
- Compression is also CPU-intensive
  - Just like GC, compression stresses the CPUs.

How do you triage root cause for CPU errors?

- Check CPU usage with *dstat*
- Check CPU usage with *OpsCenter*
  - CPU graphs can be added to monitor CPU usage
- What to do?
  - Add nodes
  - Use nodes that have more and faster CPUs

Demo 2: Using command line tools to monitor CPU

Review Questions

- How can insufficient hardware affect performance?
- What impact does garbage collection have on performance?
- Where can the page cache be configured?
- What Cassandra operations are CPU intensive?
- What tools are available for monitoring the CPU?