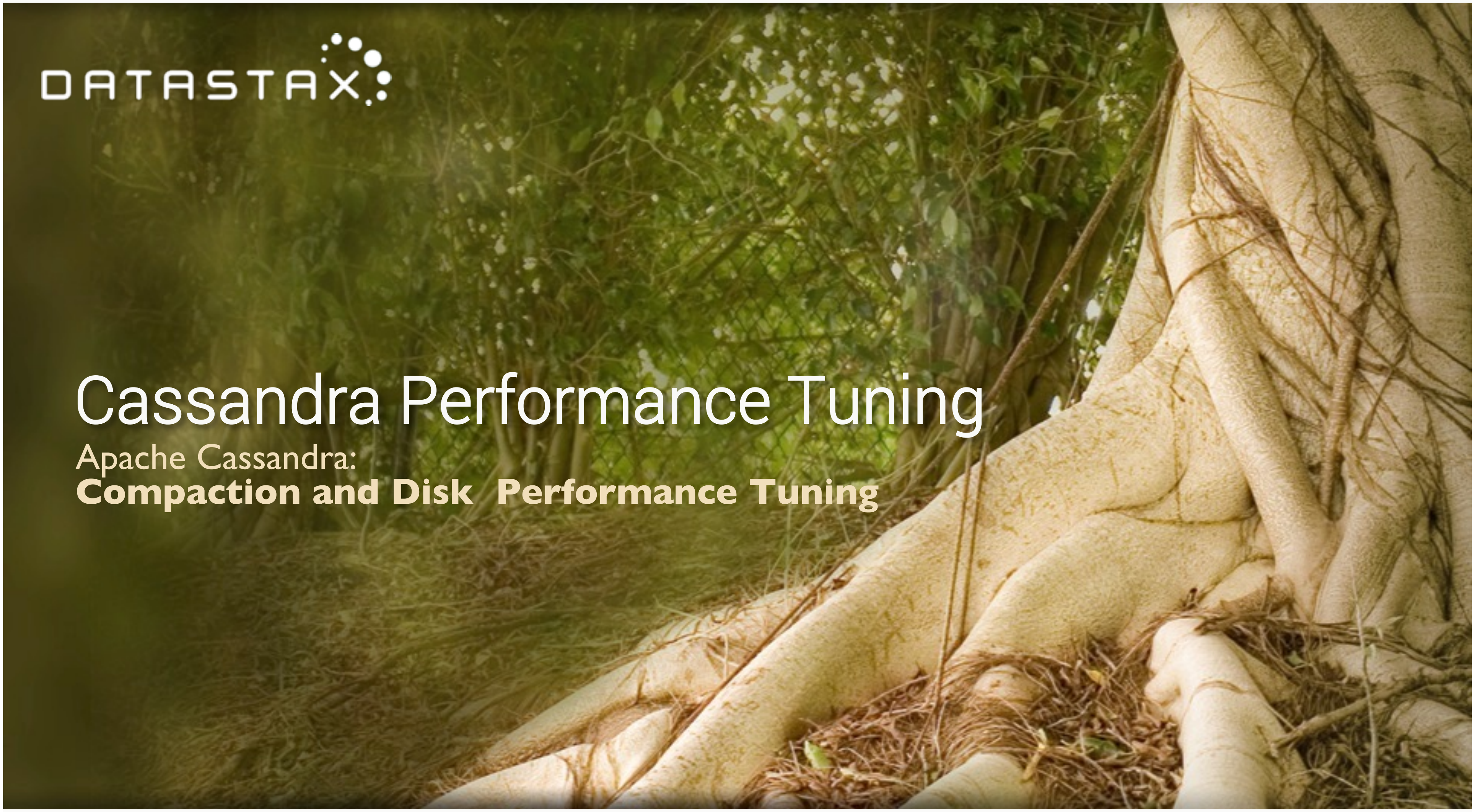

Cassandra Performance Tuning

Apache Cassandra:
Compaction and Disk Performance Tuning

Objectives: Disk and Compaction Tuning

- Discuss disk tuning strategies
- Identify compaction tuning strategies

Question

How do disk considerations affect performance?

Differences between SSDs and spinning disks

- Spinning, or rotating, disks must move a read/write mechanical head to the portion of the disk that is being written to or read
  - Much more efficient if sequential reads and writes are done, as the head can move less.
  - Data can be overwritten without reformatting.
- Solid state drives use NAND-based flash memory and no moving parts
  - However, one drawback is that flash memory must be reformatted to reclaim space from deleted or modified files.
  - Garbage collection must be used to reclaim space.
  - But random reads and writes will be faster, since no moving head must seek a location and move for the operation to be completed.

Configuring disks in the *cassandra.yaml* file

- *disk_failure_policy* – What should occur if a data disk fails?
  - *stop* – shut down gossip and Thrift, node will be considered dead, can still be inspected via JMX
  - *best_effort* – stop using failed disk and respond using remaining sstables on other disks – obsolete data can be returned if the consistency level is one
  - *ignore* – ignore fatal errors and let requests fail
- *commit_failure_policy* – What should occur if a commit log disk fails?
  - *stop* – same as above
  - *stop_commit* – shutdown commit log, let writes collect but continue to service reads
  - *ignore* – ignore fatal errors and let batches fail
- *concurrent_reads* - typically set to 16 * number of drives
- *trickle_fsync* – good to enable on SSDs

Using Linux *sysstat* tools to discover disk statistics

- System Activity Reporter (*sar*) – can get information about system buffer activity, system calls, block device, overall paging, semaphore and memory allocation, and CPU utilization
  - Flags identify the item to check: *sar –d* for disk, *sar –r* for memory, *sar –S* for swap space used, *sar –b* for overall I/O activities
  - Explore *sar* yourself to see what you can do with it.
- dstat – a Leatherman™ tool for Linux – versatile replacement for sysstat (vmstat, iostat, netstat, nfsstat, and ifstat) with colors
  - Flags identify the item to check: *dstat –d* for disk, *dstat –m* for memory, etc.
  - Can also string flags to get multiple stats: *dstat -dmnrs*

```
ubuntu@ip-172-31-8-219:/etc$ dstat -dmnrs
-dsk/total- ------memory-usage----- -net/total- --io/total- ----swap---
 read  writ| used  buff  cach  free| recv  send| read  writ| used  free
  63k   66k| 256M 75.0M 6869M  250M|   0     0 |1.53  1.58 |   0     0
   0     0 | 256M 75.0M 6869M  250M|  52B  818B|   0     0 |   0     0
   0     0 | 256M 75.0M 6869M  250M| 104B  708B|   0     0 |   0     0
   0     0 | 256M 75.0M 6869M  250M|  52B  354B|   0     0 |   0     0
   0     0 | 256M 75.0M 6869M  250M|  52B  354B|   0     0 |   0     0
   0     0 | 256M 75.0M 6869M  250M|  52B  354B|   0     0 |   0     0
   0     0 | 256M 75.0M 6869M  250M|  52B  354B|   0     0 |   0     0
   0     0 | 256M 75.0M 6869M  250M|  52B  354B|   0     0 |   0     0
   0     0 | 256M 75.0M 6869M  250M|  52B  354B|   0     0 |   0     0
```

How do you use Linux scripts to monitor disk statistics?

- *cron* jobs can run scripts that generate *sar/sysstat* output
  - Example: Just run *dstat* and pipe it to a CSV file

    dstat –drsmn --output /var/log/dstat.txt 5 3 > /dev/null
  - Example: Run a bash script that emails results of *dstat* to user

    #!/bin/bash

    Dstat –output /tmp/dstat.csv –CDN 30 360

    Mutt –a /tmp/dstat.csv –s "dstat report" me@me.com < /dev/null
- Output can be displayed on webpage for monitoring
  - Generate a file for display
- Output could be piped into graphical programs, like *Gnumeric, Gnuplot*, and *Excel* for visual displays

Using *nodetool cfhistograms* to discover disk issues

- Data is collected per read, per write, on every SSTable flush, and when compaction occurs.
- There are patterns in the histograms – and that's what you want to look for.
  - The tighter the bump pattern, the better – otherwise, you are experiencing wide variability.
  - Check the number of SSTables that must be read – reduce the seeks for read improvement.
  - If the partition size is large, try tuning some parameters, if you are using SSDs.
    - lower the *index_interval*
    - lower the *column_index_size_in_kb*
    - increase *concurrent_reads*

Using *nodetool cfhistograms* to discover disk issues

- Read latencies may have two "bumps"
  - Can mean the reads are going well until compaction starts (top bump), and point to a disk latency issue (lower bump).
  - Can be looking at filesystem cache latency (top bump) and rotating disk access (lower bump).
- How to fix it?
  - Throttle down compaction – if compaction is taking place too often, reduce the *compaction_throughput_mb_per_sec.*
    - A script could be used to set compaction throughput to a different level during the day and night.
  - Tune the disk(s)
    - Disk I/O is the place to start – almost always the issue.
    - SSDs – Might want to change the block device read-ahead.
  - Ignore it
    - If the lower bump is still showing a reasonable read latency, why bother changing things?

Using *nodetool cfhistograms* to discover disk issues

- Read latencies may create multiple small bumps.
  - Most of the rows are in main memory for processing.
  - Might have really wide rows.
- Generally, this is OK.

Using *nodetool proxyhistograms* to discover disk issues

- *nodetool proxyhistograms* will show the full request latency recorded by the coordinator.
- Whereas *nodetool cfhistograms* shows the local read latency without network or wait times.
- Comparison of these two can point to network latencies versus disk latencies.

Using TRACING to distinguish between slow disk response and slow query

- Slow disk response will be evident in how long it takes to access each drive.
- If your queries need to look for SSTables on too many partitions to complete, you will see this in the trace.
- These issues will have different patterns.

	timestamp	source	source_elapsed
execute_cql3_query	23:19:42,610	172.31.11.234	0
_by_tickerday limit 100;	23:19:42,610	172.31.11.234	22
Preparing statement	23:19:42,610	172.31.11.234	91
nining replicas to query	23:19:42,611	172.31.11.234	190
equest to /172.31.11.232	23:19:42,611	172.31.11.234	306
essage to /172.31.11.232	23:19:42,611	172.31.11.234	367
ived from /172.31.11.234	23:19:42,618	172.31.11.232	7
((-3074457345618258603)]	23:19:42,619	172.31.11.232	592
beginning in data file	23:19:42,672	172.31.11.232	53615
beginning in data file	23:19:42,672	172.31.11.232	53855
and 0 tombstoned cells	23:19:42,675	172.31.11.232	57264
beginning in data file	23:19:42,675	172.31.11.232	57311
ived from /172.31.11.232	23:19:42,678	172.31.11.234	67443
onse from /172.31.11.232	23:19:42,678	172.31.11.234	67499
and 0 tombstoned cells	23:19:42,682	172.31.11.232	64277
ied 1 rows and matched 1	23:19:42,682	172.31.11.232	64488
sponse to /172.31.11.234	23:19:42,682	172.31.11.232	64502
essage to /172.31.11.234	23:19:42,683	172.31.11.232	64624
Request complete	23:19:42,678	172.31.11.234	68639

- *source_elapsed* reports cumulative execution time on a specific node
- *timestamp* is reported in hour:minute:second, millisecond

Where is tracing information stored?

- Keyspace
  - *system_traces*
- Tables
  - *sessions*
  - *events*

```
cqlsh> use system_traces;
cqlsh:system_traces> select * from sessions limit 1;

 session_id                           | coordinator   | duration | parameters                                      | request            | started_at
--------------------------------------+---------------+----------+-------------------------------------------------+--------------------+--------------------------
 a7bd8b50-bb7d-11e3-80e6-a56f583037a6 | 172.31.11.234 |    42169 | {'query': 'select * from rollups300 limit 10;'} | execute_cql3_query | 2014-04-03 22:16:51+0000

(1 rows)

cqlsh:system_traces> select * from events limit 1;

 session_id                           | event_id                             | activity                                | source        | source_elapsed | thread
--------------------------------------+--------------------------------------+-----------------------------------------+---------------+----------------+------------
 a7bd8b50-bb7d-11e3-80e6-a56f583037a6 | a7bd8b51-bb7d-11e3-80e6-a56f583037a6 | Parsing select * from rollups300 limit 10; | 172.31.11.234 |             24 | Thrift:188
```

What role does disk readahead play in performance?

- Readahead is the file prefetching technology used in Linux.
  - Loads a file's contents into the page cache, so that when the file is read, it can be read faster.
- Setting the readahead lower for SSDs is a good idea with Cassandra.
  - The default is a setting for rotational disks, and the readahead is too high.
  - Use a readahead value of 8 for SSDs.
    - *blockdev –setra 8 <device>*

Exercise 1: Fixing Disk Issues

Question

How does compaction impact performance?

How does compaction impact performance?

- Understand compaction strategies and how they impact performance.
- Adjust compaction throttling as necessary.
- Look at the impact of min/max SSTable thresholds.
- Understand the connection between number of SSTables and compaction as it affects performance.
- See what options are available for compaction to improve performance.

How do tombstones affect compaction?

- Compaction does two things – evicts tombstones and removes deleted data – and then consolidates multiple SSTables into one
  - More tombstones means more time spent during compaction of SSTables.
- Once a column is marked with a tombstone, it will continue to exist until compaction permanently deletes the column
  - Note that if a node is down longer than the grace period for compaction, it may miss deleting the column.
  - Can result in replication of deleted data – zombie!
- To prevent issues, node repair must be done on every node on a regular basis.
  - By default, nodes are repaired every 10 days.

How data modeling affects tombstones

- If a data model requires a lot of deletes within a row of data, then a lot of tombstones are created.
  - Tombstones identify stale data awaiting deletion – that data will have to be read until it is removed by compaction.
- More effective data modeling will alleviate this issue .
  - Ensure that your data model is more likely to delete whole rows, rather than columns from a row.
- The data model has a significant impact on performance.
  - Careful data modeling will avoid the pitfalls of rampant tombstones that affect read performance.
- Tombstones will not affect write performance.

Exercise 2: Fixing Tombstone Issues

Using *nodetool compactionstats* to investigate issues

- *nodetool compactionstats* can be used to discover compaction statistics while compaction occurs.
  - Reports how much still needs compacting and the total amount of data getting compacted.

Using CQL tracing to investigate issues

- While executing a SELECT query, if TRACING is turned on
  - Can see how many nodes and partitions are accessed
  - The number of tombstones will be shown.
  - The read access time can be observed as decreasing after a compaction is complete.
    - It can also be seen to take longer while a compaction is in progress.

Why is a durable queue an anti-pattern that can cause compaction issues?

- Durable queues are constantly deleting column values.
  - This creates more tombstones than live cells in a row of data.
  - All the columns that are tombstoned still have to be traversed in order to read a live column of data.
  - Column will be read until (1) the specified limit of live columns is read, (2) all columns in the row have been read, or (3) a column beyond the finish column has been read.
- Tremendously large read latency becomes a problem.
  - A 1-millisecond read can become 300-milliseconds.
- Expiring columns have a similar effect.
- How can you fix this issue? Change your data model

How do disk choices affect compaction issues?

- Need to have enough disk space – compaction temporarily doubles disk space usage.
- Compaction is disk I/O intensive, although no random disk I/O takes place.
  - SSTables are sorted, so the merge is efficient, as no random seek is required.
- Write survey mode can be used to test new compaction strategies.
  - Adds a node to the cluster which accepts all write traffic as if it were part of the cluster, note: it doesn't join the ring
  - The node can be joined to the cluster with *nodetool join*, and the read operations benchmarked.
    - Use *nodetool cfhistograms* to look at the read performance.

Exercise 3: Fixing Compaction Issues