Cassandra Performance Tuning

Apache Cassandra:
Understanding Your Environment

Learning Objectives

- **Examine cluster and node health and tuning**
- Discuss table design and tuning

Question

DATASTAX

What activities in a Cassandra cluster happen between nodes?

Cassandra Cluster-Level Activities

DATASTAX

- coordinator
- gossip
- replication
- repair
- read repair
- bootstrapping
- node removal
- node decommissioning

Question

DATASTAX

What does a Cassandra node *do*?

What does a Cassandra Node *DO?*

DATASTAX

Read

Write

Maintain Consistency

Monitor

Participate in cluster

Question

DATASTAX

How does Cassandra organize all of that work?

How does Cassandra organize all that work?

DATASTAX

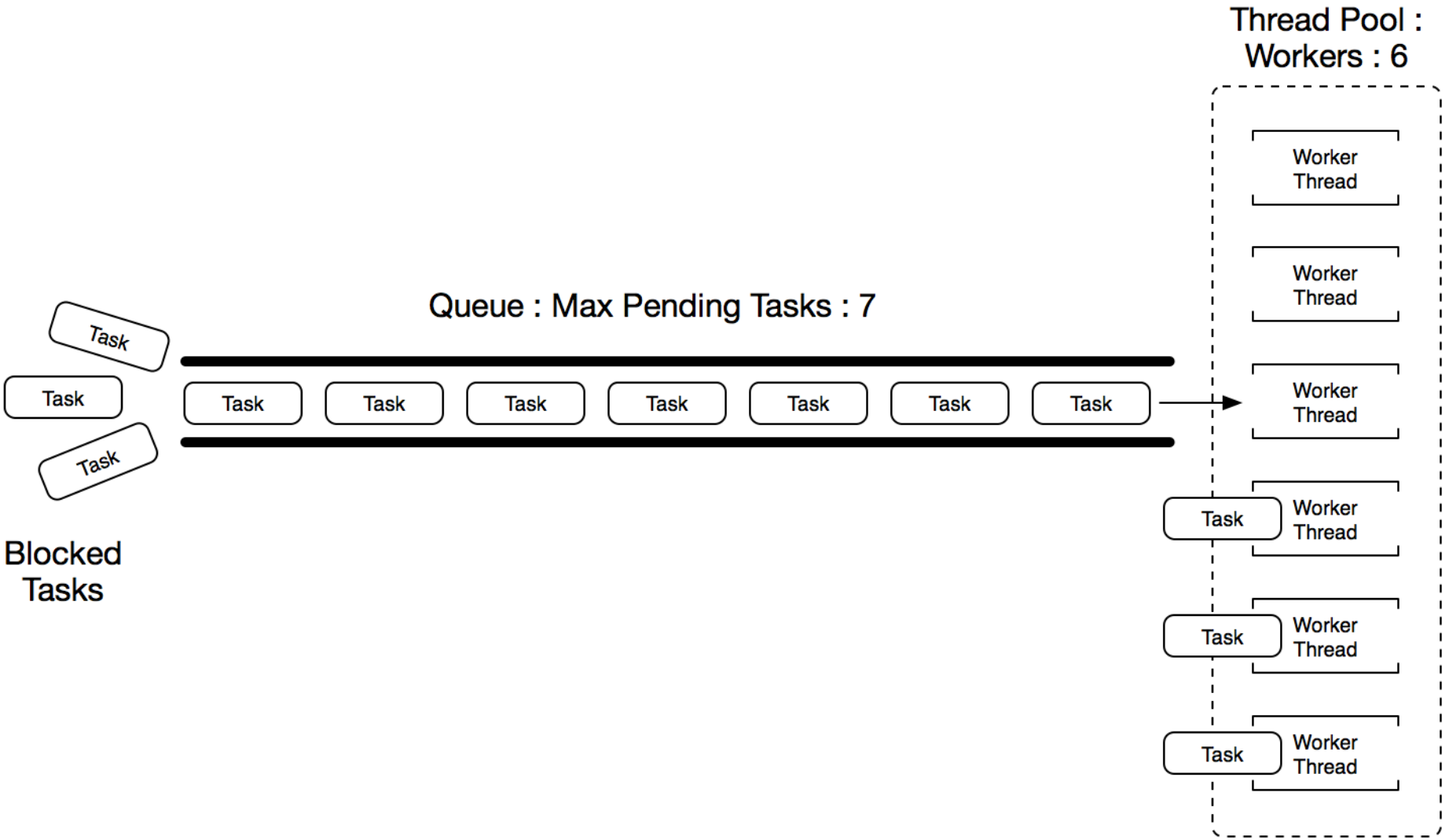
- Staged Event Driven Architecture (SEDA)
  - Separates different tasks into stages that are connected by a messaging service.
  - Each like task is grouped into a stage having a queue and thread pool.

Question

DATASTAX

What is a thread pool?

Tasks (aka Messages), a Queue and a Thread Pool



Question

DATASTAX

Why do blocked messages matter?

Why do blocked messages matter?

DATASTAX

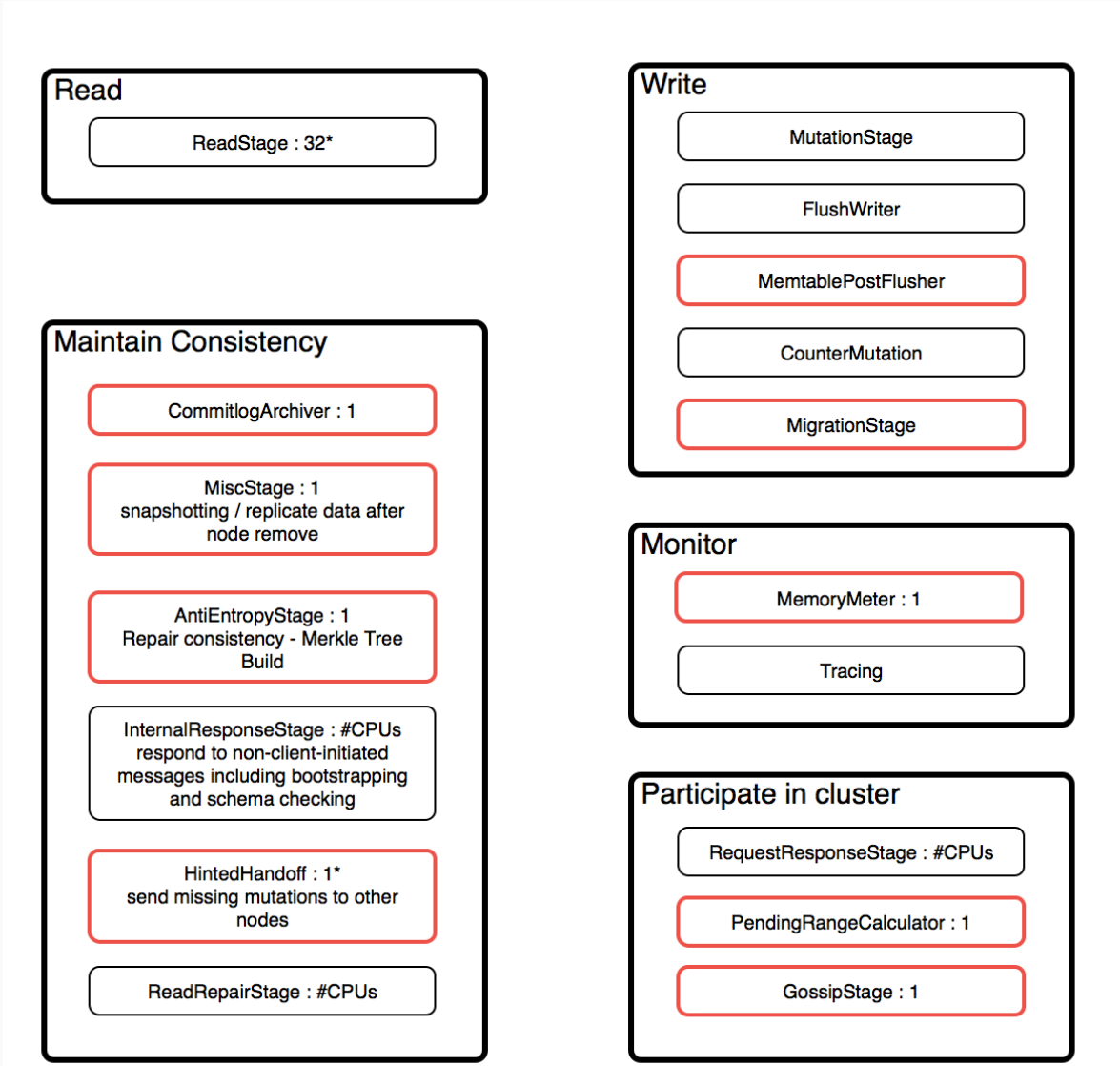
- In normal operations, code adds messages to the queue and keeps going.
- When queue fills up, this process stops until space opens up.
- Once this block occurs, other code waiting for that code blocks as well.
- Blocking is bad!

Question

DATASTAX

What are Cassandra's thread pools?

Cassandra Thread Pools

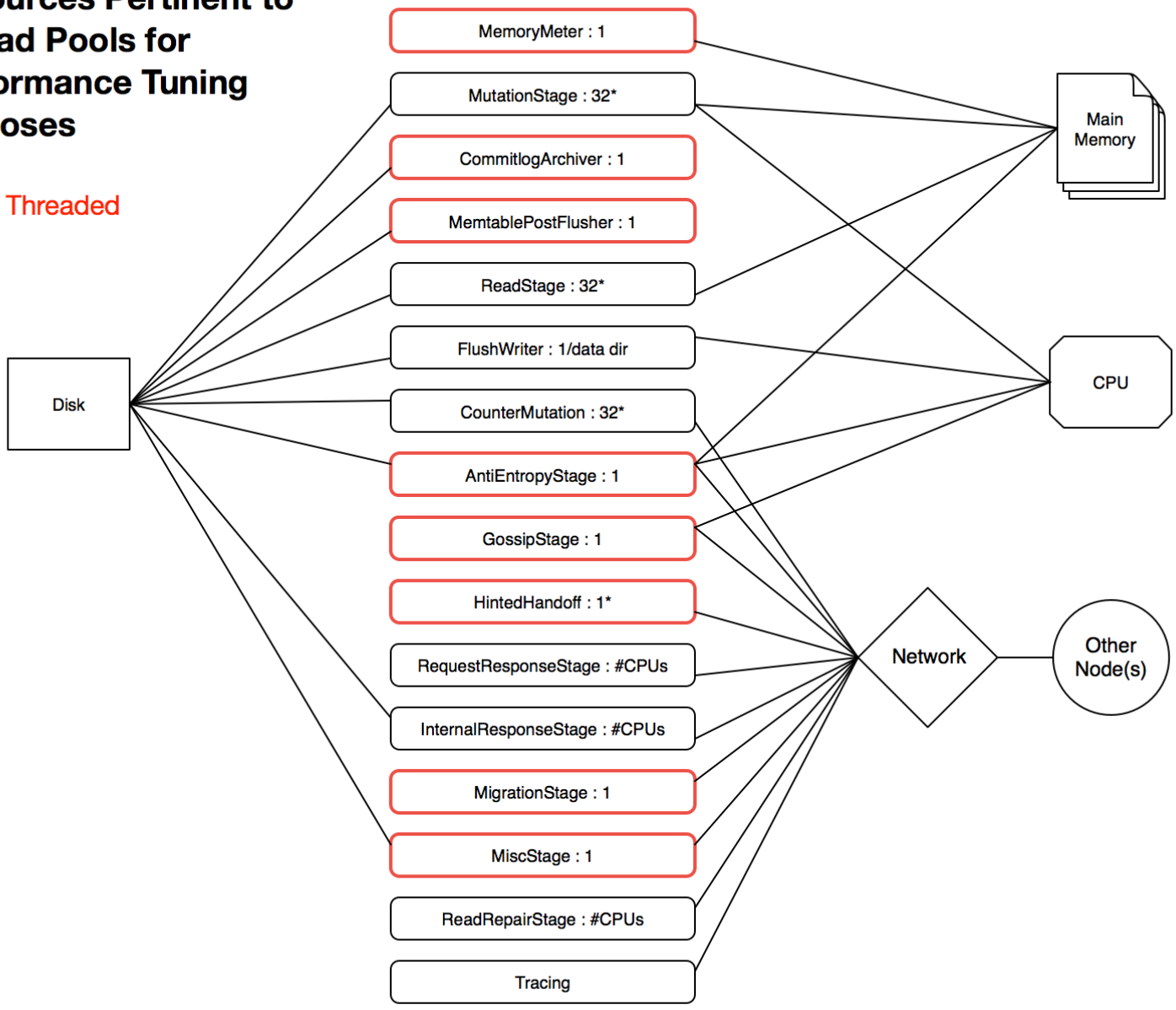


TPStats Headers

- Active—number of messages pulled off the queue, currently being processed by a thread.
- Pending—number of messages in queue waiting for a thread.
- Completed—number of messages completed.
- Blocked—when a pool reaches its max thread count it will begin queuing until the max size is reached. When this is reached it will block until there is room in the queue.
- Total Blocked/All Time Blocked—total number of messages that have been blocked.

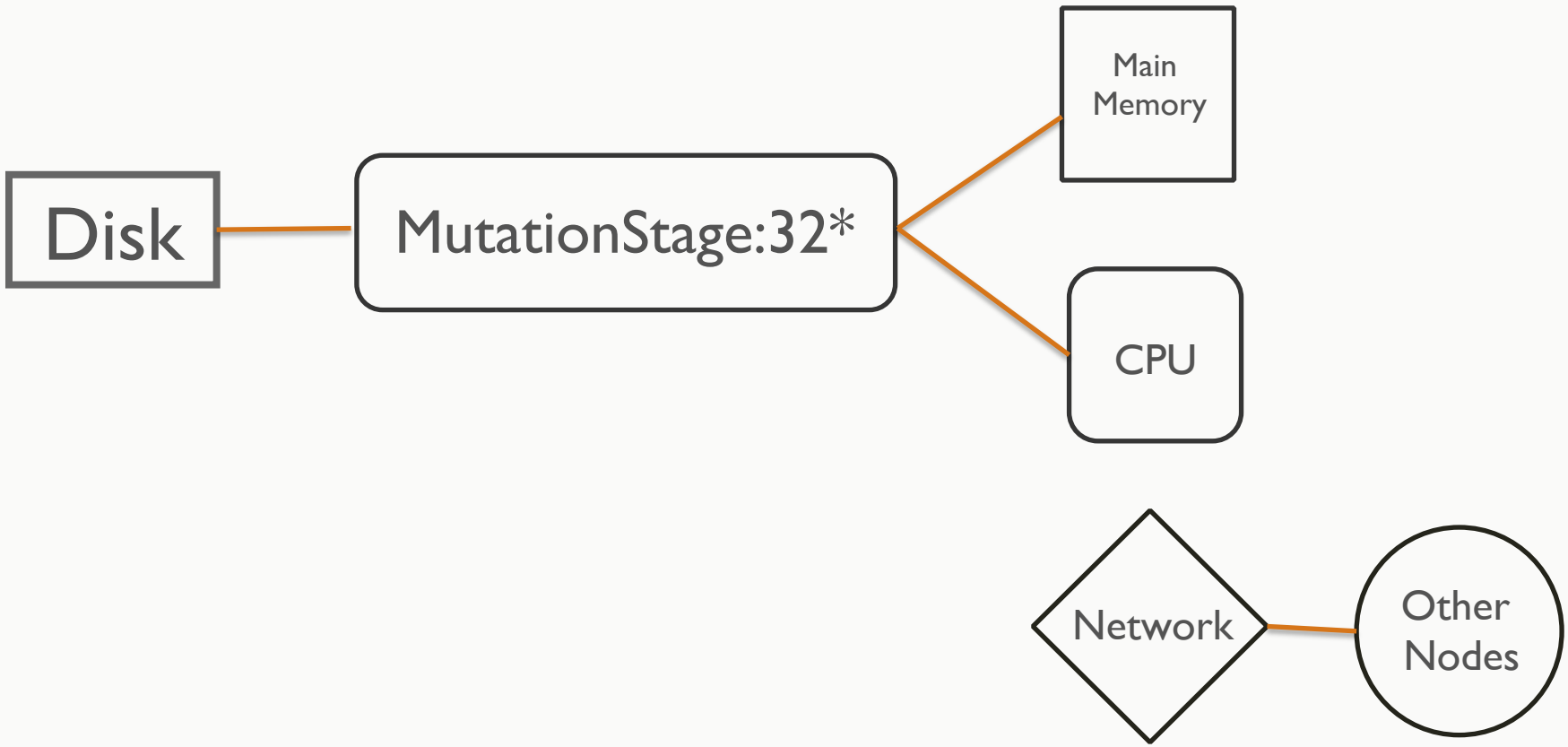
Resources Pertinent to Thread Pools for Performance Tuning Purposes

Single Threaded

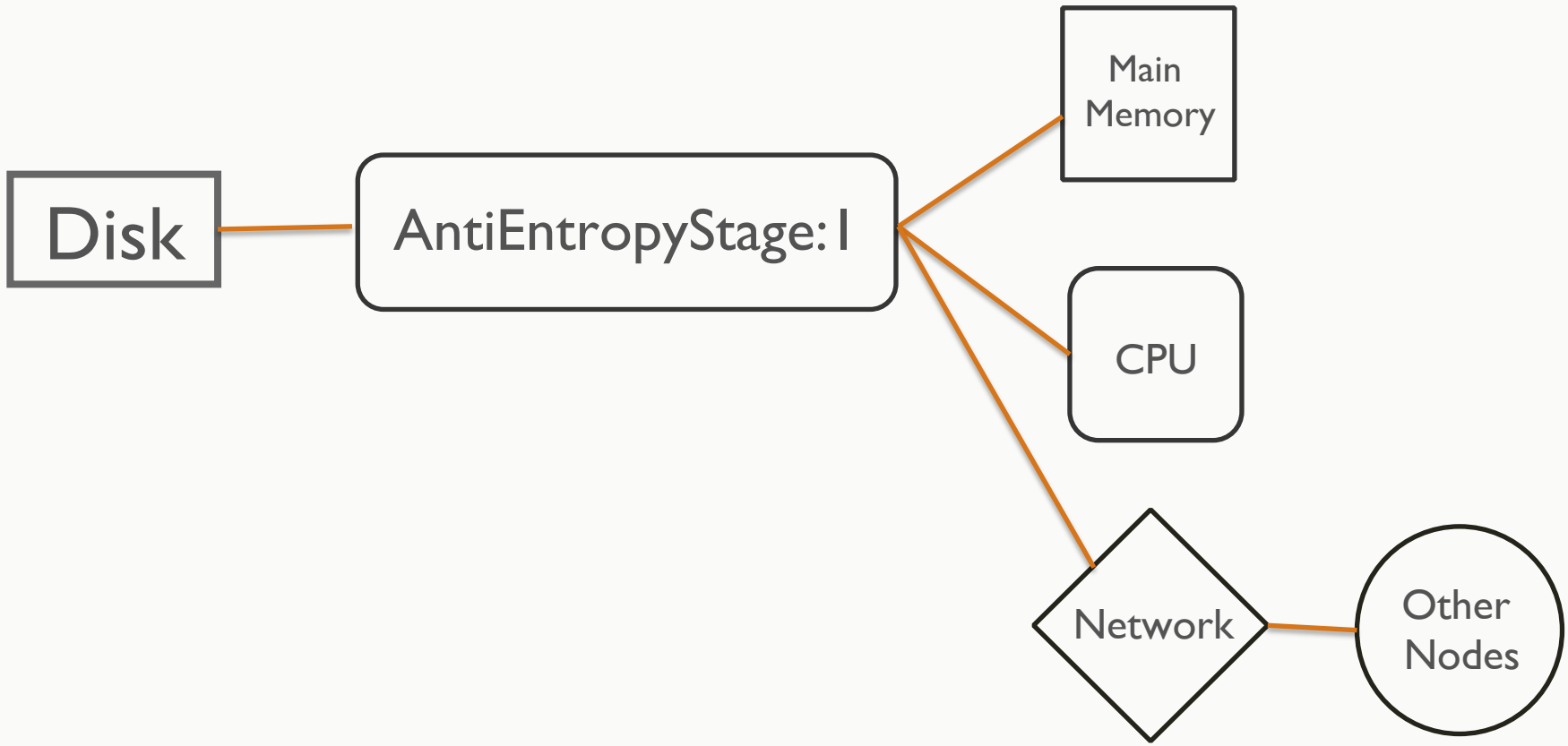


* configurable # threads

MutationStage Thread Pool (multi-threaded)



AntiEntropyStage Thread Pool (single-threaded)



Cassandra Thread Pools

DATASTAX

Multi-Threaded Stages

- **ReadStage** (affected by main memory, disk)—perform local reads
- **MutationStage** (affected by CPU, main memory, disk)—perform local insert/update, schema merge, commit log replay, hints in progress.
- **RequestResponseStage** (affected by network, other nodes)—When a response to a request is received this is the stage used to execute any callbacks that were created with the original request.
- **FlushWriter** (affected by CPU, disk)—Sort and write memtables to disk.
- **HintedHandoff** (one thread per host being sent hints, affected by disk, network, other nodes)—Sends missing mutations to other nodes.
- **MemoryMeter** (several separate threads)—Measure memory usage and live ratio of a memtable.
- **ReadRepairStage** (affected by network, other nodes)—Perform read repair
- **CounterMutation** (Formerly ReplicateOnWriteStage)—Performs counter writes on non-coordinator nodes and replicates after a local write.
- **InternalResponseStage**—Responds to non-client initiated messages, including bootstrapping and schema checking.

Cassandra Thread Pools (Continued)

DATASTAX

Single-Threaded Stages

- **GossipStage** (affected by network)—Gossip communication.
- **AntiEntropyStage** (affected by network, other nodes)—Build Merkle tree and repair consistency.
- **MigrationStage** (affected by network, other nodes)—Make schema changes.
- **MiscStage** (affected by disk, network, other nodes)—Snapshotting; replicating data after node remove.
- **MemtablePostFlusher** (affected by disk)—Operations after flushing the memtable. Discard commit log files that have all data in them persisted to sstables. Flush non-column family backed secondary indexes.
- **Tracing**—for query tracing.
- **CommitLogArchiver** (formally commitlog_archiver)—back up or restore a commit log.

Message Types

DATASTAX

Handled by the ReadStage thread pool:

- READ: read data from cache or disk
- RANGE_SLICE: read a range of data
- PAGED_RANGE: read part of a range of data

Handled by the MutationStage thread pool:

- MUTATION: write (insert or update) data
- COUNTER_MUTATION: changes counter columns
- READ_REPAIR: update out-of-sync data discovered during a read

Handled by ReadResponseStage:

- REQUEST_RESPONSE: respond to a coordinator
- _TRACE: used to trace a query (enable tracing or every (trace probability) queries)
- BINARY: deprecated

Question

DATASTAX

Why are some message types “droppable”?

Why are some message types "droppable"?

DATASTAX

- Some messages can be dropped to prioritize activities in Cassandra when high resource contention occurs.
- The number of these messages dropped is in the nodetool *tpstats* output.
- If you see dropped messages, you should investigate.

What is the state of your cluster?

DATASTAX

What is the state of your cluster?

- Use nodetool to gather information on your cluster
- Use OpsCenter to gather information on your cluster

DATASTAX

What can Nodetool do?

- nodetool -h node0 tpstats
  - Statistics that indicate active, pending and completed tasks for each threadpools
- nodetool -h node0 compactionstats
  - Statistics that indicate as it increases that compactions are falling behind
- nodetool -h node0 netstats
  - Statistics that display network information, such as status of streaming operations and number of active, pending, and completed commands and responses

Exercise 1: Examine cluster health with nodetool

DATASTAX

What is OpsCenter?

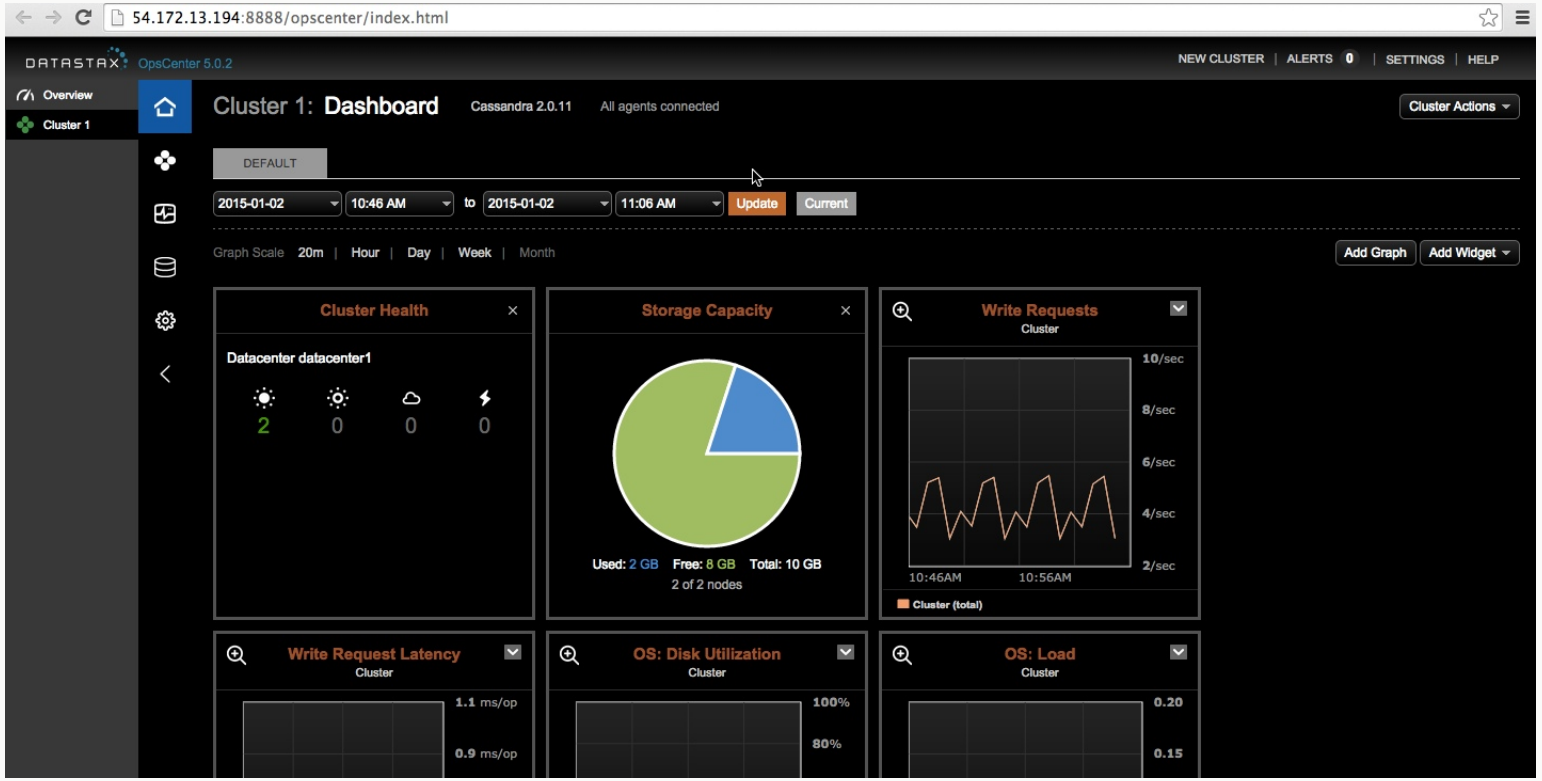
- Web-based dashboard to monitor and manage complex workloads
- Can be used with Apache Cassandra or DataStax Enterprise (DSE)
- Provides ability to:
  - Create new clusters
  - Add, edit and remove nodes
  - Rebalance clusters
  - Run repairs
  - Creates charts, graphs and reports for performance analysis

What can you learn from *OpsCenter* output?

- JVM characteristics can be monitored with graphs.
- Hardware stats on disk, CPU, memory can all be displayed.
- Read/write statistics for individual column families, as well clusterwide.
- All Cassandra metrics can be monitored – compactions, repairs, flushes, gossip, hinted handoffs.
- The visual displays can be customized and saved, to quickly pull up specialized sets of data when troubleshooting is required.

What performance tools are available with OpsCenter?

- Ability to create several charts/graphs with performance metrics
- Can compare graphs by overlaying results in a single graph



Exercise 2: Examine cluster health with OpsCenter

DATASTAX

Changing logging levels with nodetool setlogginglevel

- nodetool getlogginglevels
  - Used to get the current runtime logging levels
- nodetool setlogginglevel
  - Used to set logging level for a service
  - Can be used instead of modifying the logback.xml file
- Possible levels:
  - ALL
  - TRACE
  - DEBUG
  - INFO
  - WARN
  - ERROR
  - OFF

Exercise 3: Use nodetool setlogginglevel to analyze logs

Cassandra Specific Tuning Objectives

DATASTAX

- Examine cluster and node health and tuning
- **Discuss table design and tuning**

What can you learn with *nodetool cfstats* output?

- Look for patterns!
- Read/write latency for each keyspace
- SSTable count and space used per table
- Memtable cell count and data size per table
- Local read and write latency per table
- Bloom filter false ratio, false positives, space used per table
- Compaction information table
- Tombstone information per table

```
Keyspace: Keyspace1
        Read Count: 100
        Read Latency: 18.459229999999998 ms.
        Write Count: 0
        Write Latency: NaN ms.
        Pending Tasks: 0
Table: Standard1
SSTable count: 23
Space used (live), bytes: 405696340
Space used (total), bytes: 405939016
SSTable Compression Ratio: 0.0
Number of keys (estimate): 1349888
Memtable cell count: 0
Memtable data size, bytes: 0
Memtable switch count: 0
Local read count: 100
Local read latency: NaN ms
Local write count: 0
Local write latency: NaN ms
Pending tasks: 0
Bloom filter false positives: 0
Bloom filter false ratio: 0.00000
Bloom filter space used, bytes: 2580048
Compacted partition minimum bytes: 259
Compacted partition maximum bytes: 310
Compacted partition mean bytes: 310
Average live cells per slice (last five minutes): 5.0
Average tombstones per slice (last five minutes): 0.0
```

DATASTAX

What can you learn with *nodetool cfhistograms* output?

- Number of SSTables per read – how many SSTables are involved in reads
- Write latency – how many write operations took a given number of microseconds
- Read latency – how many read operations took a given number of microseconds
- Partition Size – how many partitions are a given number of bytes in size
- Cell Count Size –how many partitions have a given number of cells

Exercise 4: Use nodetool cfstats to examine table health

Exercise 5: Use nodetool cfhistograms command

What is CQL tracing?

- In cqlsh, enable TRACING
- Once tracing is on, operations like an INSERT will create a trace of the operations executed to complete the command
- SHOW SESSION <tracing session id> will show the trace for any session

```
Tracing session: 1abe23f0-b767-11e3-80e6-a56f583037a6

 activity                                                                                                  | timestamp    | source         | source_elapsed
-----------------------------------------------------------------------------------------------------------+--------------+----------------+----------------
                                                                                        execute_cql3_query | 17:25:20,943 | 172.31.11.234 |              0
                                          Parsing select * from rollups300 limit 10; | 17:25:20,943 | 172.31.11.234 |             75
                                                                        Preparing statement | 17:25:20,943 | 172.31.11.234 |            139
                                                                   Determining replicas to query | 17:25:20,943 | 172.31.11.234 |            219
                                                  Enqueuing request to /172.31.11.232 | 17:25:20,943 | 172.31.11.234 |            357
                                                    Sending message to /172.31.11.232 | 17:25:20,944 | 172.31.11.234 |            485
                                                 Message received from /172.31.11.234 | 17:25:20,947 | 172.31.11.232 |             92
 Executing seq scan across 2 sstables for [min(-9223372036854775808), max(-3074457345618258603)] | 17:25:20,950 | 172.31.11.232 |           2564
                                               Seeking to partition beginning in data file | 17:25:20,951 | 172.31.11.232 |           3747
                                               Seeking to partition beginning in data file | 17:25:20,973 | 172.31.11.232 |          25950
                                                 Message received from /172.31.11.232 | 17:25:21,006 | 172.31.11.234 |          63271
                                             Processing response from /172.31.11.232 | 17:25:21,006 | 172.31.11.234 |          63360
                                                 Read 11 live and 0 tombstoned cells | 17:25:21,008 | 172.31.11.232 |          60257
                                               Seeking to partition beginning in data file | 17:25:21,008 | 172.31.11.232 |          60331
                                                 Read 1 live and 0 tombstoned cells | 17:25:21,008 | 172.31.11.232 |          60392
                                                       Scanned 1 rows and matched 1 | 17:25:21,008 | 172.31.11.232 |          60616
                                                 Enqueuing response to /172.31.11.234 | 17:25:21,008 | 172.31.11.232 |          60683
                                                   Sending message to /172.31.11.234 | 17:25:21,008 | 172.31.11.232 |          61059
                                                                        Request complete | 17:25:21,006 | 172.31.11.234 |          63561
```

CQL tracing with *nodetool settraceprobability*

DATASTAX

- nodetool –h <host> settraceprobability 1/0
- 1 enables tracing, 0 disables
- Turn on this function, check trace output, make sure to disable tracing when you're done!
  - One trace equals almost 10 writes, so trace data can add up quickly
  - Table *system_traces.sessions* holds trace data
- Traces all queries for the node until it is disabled, whereas TRACING on in cqlsh only traces queries in that shell
  - TRACING ON in cqlsh is essentially setting the probability to 1
- Can be used to constantly trace data at low level
  - Set the probability to 0.01 (1%) to collect data at a slow trickle
  - Useful for troubleshooting – collects data before a problem occurs

Exercise 6: Use nodetool cqltracing to collect trace output

Exercise 7: Examine table health with OpsCenter

Summary

- Performance problems are found by collecting information about your cluster, nodes, hardware, operating system and data model
- Nodetool and OpsCenter are tools used to collect data about your environment
- Poorly designed data model can affect performance by causing hotspots or too many tombstones
- cfstats and cfhistograms are tools that can be used to collect column family metrics

DATASTAX