nodetool tpstats

Provides usage statistics of thread pools.

Synopsis

```
$ nodetool <options> tpstats
```

Cassandra tarball installations:

```
installation_location/cassandra/bin
```

Options

Short	Long	Description
-h	--host	Hostname or IP address.
-p	--port	Port number.
-pwf	--password-file	Password file path.
-pw	--password	Password.
-u	--username	Remote JMX agent username.
--	Separates an option from an argument that could be mistaken for a option.	

Description

Cassandra is based on a Staged Event Driven Architecture (SEDA). Cassandra separates different tasks into stages connected by a messaging service. Each stage has a queue and a thread pool. Although some stages skip the messaging service and queue tasks immediately on a different stage when it exists on the same node. Cassandra can back up a queue if the next stage is too busy and lead to a performance bottlenecks.

The `nodetool tpstats` command reports on each stage of Cassandra operations by thread pool:

- The number of `Active` threads
- The number of `Pending` requests waiting to be executed by this thread pool
- The number of tasks `Completed` by this thread pool
- The number of requests that are currently `Blocked` because the thread pool for the next step in the service is full
- The total number of `All-Time Blocked` requests, which are all requests blocked in this thread pool up to now.

Reports are updated when SSTables change through compaction or flushing.

Run `nodetool tpstats` on a local node to get statistics for the thread pool used by the Cassandra instance running on that node.

Run `nodetool tpstats` with the appropriate options to check the thread pool statistics for a remote node. For setup instructions, see Secure JMX Authentication.

nodetool tpstats pool names and tasks

This table describes the Cassandra task or property associated with each pool name reported in the `nodetool tpstats` output:

Pool Name	Associated tasks	Related information
AntiEntropyStage	Processing repair messages and streaming	For details, see Nodetool repair.
CacheCleanupExecutor	Clearing the cache	
CommitlogArchiver	Copying or archiving commitlog files for recovery	
CompactionExecutor	Running compaction	

Pool Name	Associated tasks	Related information
CounterMutationStage	Processing local counter changes	Will back up if the write rate exceeds the mutation rate. A high pending count will be seen if consistency level is set to ONE and there is a high counter increment workload.
GossipStage	Distributing node information via Gossip	Out of sync schemas can cause issues. You may have to sync using `nodetool resetlocalschema` .
HintedHandoff	Sending missed mutations to other nodes	Usually symptom of a problem elsewhere. Use `nodetool disablehandoff` and run repair.
InternalResponseStage	Responding to non-client initiated messages, including bootstrapping and schema checking	
MemtableFlushWriter	Writing memtable contents to disk	May back up if the queue is overruns the disk I/O, or because of sorting processes. **WARNING:** `nodetool tpstats` no longer reports blocked threads in the MemtableFlushWriter pool. Check the Pending Flushes metric reported by `nodetool tblestats`.
MemtablePostFlush	Cleaning up after after flushing the memtable (discarding commit logs and secondary indexes as needed)	
MemtableReclaimMemory	Making unused memory available	
MigrationStage	Processing schema changes	
MiscStage	Snapshotting, replicating data after node remove completed.	
MutationStage	Performing local inserts/updates, schema merges, commit log replays or hints in progress	A high number of `Pending` write requests indicates the node is having a problem handling them. Fix this by adding a node, tuning hardware and configuration, and/or updating data models.
Native-Transport-Requests	Processing CQL requests to the server	
PendingRangeCalculator	Calculating pending ranges per bootstraps and departed nodes	Reporting by this tool is not useful — see Developer notes
ReadRepairStage	Performing read repairs	Usually fast, if there is good connectivity between replicas. If `Pending` grows too large, attempt to lower the rate for high-read tables by altering the table to use a smaller `read_repair_chance` value, like 0.11.
ReadStage	Performing local reads	Also includes deserializing data from row cache. Pending values can cause increased read latency. Generally resolved by adding nodes or tuning the system.
RequestResponseStage	Handling responses from other nodes	
ValidationExecutor	Validating schema	

nodetool tpstats droppable messages

Cassandra generates the messages listed below, but discards them after a timeout. The `nodetool tpstats` command reports the number of messages of each type that have been dropped. You can view the messages themselves using a JMX client.

Message Type	Stage	Notes
BINARY	n/a	Deprecated

Message Type	Stage	Notes
_TRACE	n/a (special)	Used for recording traces (nodetool settraceprobability) Has a special executor (1 thread, 1000 queue depth) that throws away messages on insertion instead of within the execute
MUTATION	MutationStage	If a write message is processed after its timeout (write_request_timeout_in_ms) it either sent a failure to the client or it met its requested consistency level and will relay on hinted handoff and read repairs to do the mutation if it succeeded.
COUNTER_MUTATION	MutationStage	If a write message is processed after its timeout (write_request_timeout_in_ms) it either sent a failure to the client or it met its requested consistency level and will relay on hinted handoff and read repairs to do the mutation if it succeeded.
READ_REPAIR	MutationStage	Times out after write_request_timeout_in_ms
READ	ReadStage	Times out after read_request_timeout_in_ms. No point in servicing reads after that point since it would of returned error to client
RANGE_SLICE	ReadStage	Times out after range_request_timeout_in_ms.
PAGED_RANGE	ReadStage	Times out after request_timeout_in_ms.
REQUEST_RESPONSE	RequestResponseStage	Times out after request_timeout_in_ms. Response was completed and sent back but not before the timeout

Example

Running `nodetool tpstats` on the host `labcluster`:

```
$ nodetool -h labcluster tpstats
```

Example output is:

Pool Name	Active	Pending	Completed	Blocked	All time blocked
CounterMutationStage	0	0	0	0	0
ReadStage	0	0	103	0	0
RequestResponseStage	0	0	0	0	0
MutationStage	0	0	13234794	0	0
ReadRepairStage	0	0	0	0	0
GossipStage	0	0	0	0	0
CacheCleanupExecutor	0	0	0	0	0
AntiEntropyStage	0	0	0	0	0
MigrationStage	0	0	11	0	0
ValidationExecutor	0	0	0	0	0
CommitLogArchiver	0	0	0	0	0
MiscStage	0	0	0	0	0
MemtableFlushWriter	0	0	126	0	0
MemtableReclaimMemory	0	0	126	0	0
PendingRangeCalculator	0	0	1	0	0
MemtablePostFlush	0	0	1468	0	0
CompactionExecutor	0	0	254	0	0
InternalResponseStage	0	0	1	0	0
HintedHandoff	0	0	0		

Message type	Dropped