S Cloud DataStax Evaluation Questions

# Summary

This document responds to the S Cloud “Internal Test Results Feedback – DataStax” presentation and “AWS Instances Estimation – DataStax” spreadsheet. Please find here a summary of the DataStax response. Supporting details follow the summary.

## YCSB Testing Feedback

There are three areas of feedback from the DataStax team.

* For the purposes of this quick benchmark, default settings were used by both Samsung and DataStax. There are no other specific recommendations other than to use the default DataStax AMI.
* DataStax Cassandra is highly tunable. We expect to be able to tune your production Cassandra for higher performance for your production use case than you saw in this benchmark.
* Because it is focused on a narrow set of variables (TPS) under non-production conditions, the YCSB benchmark is not well suited for understanding long term production-at-scale database performance. Additional factors must be considered.
  + <http://www.datastax.com/dev/blog/how-not-to-benchmark-DSE>
  + <http://www.datastax.com/dev/blog/how-not-to-benchmark-DSE-a-case-study>

## Node Failure Analysis

Consider the following key points regarding the node failure experienced by Samsung during your tests.

* Cassandra node failure can occur from a number of root causes. The information provided does not enable the DataStax team to diagnose why your node went down. Please see below our request for more information to allow us to correctly diagnose your node failure.
* In a properly configured cluster, a Cassandra node failure should not prevent the test from completing. Given the limited information the DataStax team has, we suspect there is missing information or that your cluster was misconfigured during your test.

## Thread Count Analysis

As you discovered in your testing, DataStax Cassandra provides low latency and high throughput on a mixed workload at lower thread counts. Putting high focus on thread count may not give the best performance increase. Other tuning variables are more important. See the details in section (1) below for more information on thread count.

## Optimal Cluster Sizing and Throughput

The Samsung team commented that the throughput was rather low and did not seem to be network bound. Consider the following points:

* DataStax Cassandra throughput is highly dependent on the hardware stack, data model, driver configuration, and client application access patterns, and many more variables that can be tuned by use case.
* Cassandra is often IO bound, meaning that payload size is a key determinant of throughput. Reduced payload size, along with tuning many other variables that affect storage layer access, will greatly increase TPS.
* YCSB is a poor indicator of true production-at-scale throughput capabilities of Cassandra. See the links above.

# Preface

Throughput is one small aspect of deploying a distributed database solution. The throughput we aim to estimate is not peak throughput, or benchmark type throughput, but rather a real production level throughput that DataStax can stand behind. Of course each application, in reality, is going to have an optimal throughput level, but our estimates are aimed to provide a “close as possible” estimate for what you can expect when you really start to run this.

Once we have a standard throughput, we need to build systems that are always available, scale endlessly, and have the same operational complexity at 100 nodes as they do at 1000 nodes. Our solution’s cross-region replication, peer to peer architecture, and workload isolation is what provides a holistic platform for big data development that makes enterprises comfortable in their ever-growing deployments. For this reason, in many cases, we optimize for various needs that may not focus on throughput.

For example, if we need to keep less data on disk in order to guarantee very fast node bootstraps, we will add more nodes than we absolutely need to meet my throughput, so that each node holds less total data. While this means each node will probably be serving less than it’s peak throughput, it allows me the flexibility to quickly add nodes when necessary, which improves my ability to react to changes in system demand.

Given the stated goals from the SCloud team during your US visit, we think the holistic picture is what you are looking for, and we wanted to re-iterate our position that the entire data lifecycle management in an enterprise is the target market for DataStax, and that we believe we are in the best position to deliver a successful deployment for organizations aligned with these goals.

# 1) YCSB Parameters

## Parameters and test

Your parameters are fine, and the test results that you have achieved are in line with what would be a reasonable estimate of Cassandra performance under this specific use case on the given hardware stack. We have included at the bottom of this document a section titled “Estimating Cluster Throughput” that explains how we arrive at rough estimates for both max throughput and production-level recommended throughput for clusters. We hope this highlights why the throughput you see in your mixed workload tests is in line with expectations.

## Differences between workloads

A read-only workload only has to do random reads from disk. There are very limited sequential writes and reads, which result from compaction. In this case, since the initial data load was completed, there is a lot less compaction going on, limiting the amount of disk access. For this reason, reads will scale very well, despite a large number of threads. At a certain point this will cross your SLA boundaries as the node will be overloaded and the pure random reads would be too much for the disk. But you did not hit the point in your testing where the node became I/O bound.

A write-only workload does sequential writes and sequential reads, but no random reads. For this reason, it will similarly perform very well as you increase threads, again up until a certain point.

We have found that a high thread count in YCSB results in more contention and higher latencies. We did not see the same behavior with a cassandra-stress test run (the stress tool that ships with DSE), although we expect the max thread count even with cassandra-stress to be relatively low for a 10KB payload size and a mixed workload. Beyond a certain point, there is too much load thrown at the cluster to achieve consistent behavior. Like we discuss below, the maximum throughput you saw with 8 threads using YCSB is consistent with the max throughput we would expect.

In your test you had a failure that resulted in an inability to complete the test. However there is not enough detailed information provided about the node failure for the DataStax team to accurately diagnose why the failure occurred. Can you please provide the sysem.log files containing the node failure events? We are not sure if this was a process crash, a server crash, disk space filling up, out of memory condition, etc.

A 50/50 read/write workload does sequential writes, sequential reads, and random reads. The number of total ops/s you can expect from a 50/50 workload will be less than what you can expect from a read-only workload, and will also be less than a write-only workload (although read-only workload will heavily overstate throughput, even more so than write-only, since the only disk access is reads and there is no contention)

## Analysis of results

What you see in your test results is in line with expected throughput given the hardware for your cluster. We have a very rough equation to estimate cluster throughput. While it can’t be relied upon as a scientific, it is a very good sanity check to make sure we are getting enough throughput from our cluster. The equation for PEAK throughput (much different from recommended production throughput) is:

(K \* #physicalCPUCores \* #nodes) / replication\_factor

\* where K is a constant the is determined by key variables of the system and workload

For a 1KB payload size and a workload of blended writes and reads, this constant is around K=4000. The payload size affects the constant (4000). We have found through testing that a 10KB payload size gives us about 1/6 the throughput of a 1KB payload size. Again, these are rough ballpark numbers. This makes our new equation:

(666 \* #physicalCPUCores \* #nodes) / replication\_factor

This means that, for your hardware and cluster size we should expect:

(666 \* 16 \* 5) / 3 = 18K ops/s. This is in line with the numbers you got, around 21K ops/s. We also noticed that at 21K, the latency was a little higher than what we would be comfortable with. We would expect that at 18K ops/s, you would see latencies that are more in line with what the equation is based on (for very low latency workloads).

# 2) DSE Configuration

Your settings are fine, and we used default settings in our tests as well. The only difference is that we spun up our cluster using the DataStax AMI. The AMI sets the IP addresses in the yaml file, goes with default heap settings (8GB), and mounts the instance storage as a single raid0 volume presented to the OS. It also changes from the default of virtual nodes to manual token assignment. We reverted this change for our tests to simulate an actual default cluster, so this will be similar to what you did. In any case, looking at your settings, the throughput you can expect will not be significantly different than what you would expect from the testing we ran.

# 3) Node failure on write test

It is difficult to hypothesize why a node might fail. Also, failure of a single node should not affect the test, as the other nodes should be available to accept writes. At the worst, you can expect 4/5 of the throughput if a single node fails. Can you provide some detail about the nature of this failure? Is it a process failure or a server failure? Was there any heap pressure (“GCInspector” messages in logs)?

# 4) Required memory space

Cassandra only needs enough memory for the JVM on each node. On a production cluster with the G1 garbage collector we would likely size you at a 32GB heap. Default for the YCSB testing was 8GB heap, but for your workload we’d tune that. This means that your minimum amount of memory, given our 150 node recommendation, is 4800GB of memory. You can go as high as you like on main memory, however, and anything you don’t use for heap will be used for page cache. So if you had 256GB per machine, you can very much take advantage of all of that memory with OS cache to improve your read performance by reducing disk access. This will also improve write performance, as more caching makes more i/o available for write and read requests.

# 5) Explanation of sizing

You mentioned in your message that you noticed the throughput was not network bound. Cassandra workloads are rarely network bound, and are generally either CPU bound or disk bound. In any case, we can use our equation above to give us the expected throughput for our cluster. Let us revisit our equation above, given our recommended hardware of i2.4xlarge, with 8 physical CPUs (16vCPU):

(666 \* 8 \* 150) / 3 = 266,000

You mentioned that you needed 200K ops/s. So if 150 nodes can give you 266,000 ops/s, why are we recommending 150 nodes for only 200K ops? The reason is that peak throughput, which the equation measures, is much different from our recommended steady state throughput. We typically recommend 25% overhead to start on throughput, which means that if a cluster can handle 266K ops/s, we recommend you run 75% of that, or 200K.

I wanted to compare this to the actual throughput we got from our longevity test to illustrate that while not exact, the formula does give us a trustworthy way to estimate cluster throughput. If you run the numbers on our longevity test you’ll find that peak throughput should be:

(666 \* 8 \* 5) / 3 = 8800

And if we make room for our 25% overhead:

8800 \* .75 = 6600

If you look at our test results, 6600 is right in line with the throughput we got for our longevity test for our cluster, showing that this rough equation does work. For this reason, we feel that the throughput defined of 1,333 per node is very much in line with our expected results, mainly because of replication and our payload size of 10KB. Please let me know if this makes sense, we are happy to help explain this further if necessary.

# 6) High water mark for memory

Cassandra, again, only really needs the amount of memory you set for the heap. But your cluster will use all memory available unless you limit available main memory through the OS. Any memory not used by the heap will be used by the page cache. So you should generally see very little free memory on your cluster. Again, Cassandra only really needs the heap space, and you can safely limit main memory usage if you don’t want the page cache to take up all your memory. Less memory means less cache space, which generally means more disk access for random workloads.