Performance Analysis of Scylla as a Write Heavy Database

[1]Ashutosh Utpal Gandhi, [2]G Jaya Prasad Reddy, [3]Dr. Nagaraja G.S.

[1, 2]Undergraduate Student,[3]Professor
[1, 2, 3]Computer Science and Engineering,
[1, 2, 3,]R.V. College of Engineering, Bengaluru, India

Abstract : In the past decade, a lot of technologies have emerged to solve the problem of handling of humungous data that is being generated every day. Among these technologies, NoSQL databases have shown who rule the world in the field today. We will take a look at Scylla and how it performs as a write heavy database. The setup consists of a cluster with 6 nodes spread across 2 datacenters. The component is installed on machines running Custom Linux OS. Cassandra-stress will be used to perform tests on the cluster. The cluster reached 600k ops with mere 6 nodes and can scale really well as the nodes are increased. The latencies achieved were impressive with 99th percentile as 50ms and 95th percentile as 10ms.

***Index Terms* - NoSQL, Write-Heavy, Scylla, Cassandra-stress.**

I. INTRODUCTION

Scylla is an open-source NoSQL distributed data store. It is one of the NoSQL databases which offers really high throughput at sub millisecond latencies. The important thing is that it accomplishes this at a fraction of the cost of a modern NoSQL database.

ScyllaDB is almost similar to Cassandra in terms of functionality except that it is completely written in C++. But saying it's a mere C++ port would be an understatement. It has a lot of changes compared to Cassandra in terms of the design and implementation under the hood which are not visible to the user but they lead to a huge performance improvement.

1.1. Scylla Architecture

Scylla is developed using C++ as it provides very precise control over everything a database does, alongside deliberations that empower database designers to make code both complex and manageable. Using C++ allows Scylla to fully optimize low-level operations for the available hardware.

Scylla supports complete compatibility with Cassandra. Scylla's Cassandra compatibility encompasses:

- Write Protocol: Scylla provides supports for Thrift, CQL, and the full polyglot of languages.
- Monitoring: Scylla also provides support for the JMX protocol. Scylla adds a JVM-proxy daemon which transparently translates JMX into it's RESTful API.
- Underlying File Format and Algorithms: Scylla uses Cassandras SSTable format and supports all the compaction strategies supported by Cassandra.
- Configuration File: Scylla is uses the same configuration file while Cassandra uses, cassandra.yaml, allowing for a seamless migration path. Scylla always provides maximum parallelism.
- Command Line Interface: Scylla uses the same command line tool (nodetool) as Cassandra. The processes from backup to repair of a Scylla node are identical to Cassandra's.

Scylla Uses a shared-Nothing architecture. There are two levels of shardng in Scylla. In the first level, the entire dataset of the cluster is sharded into among the individual nodes. The second level, which is transparent to users, is within each node. The nodes token range is automatically sharded across any available CPU cores. Each shard-percore forms an independent unit, which is fully responsible for its own dataset.

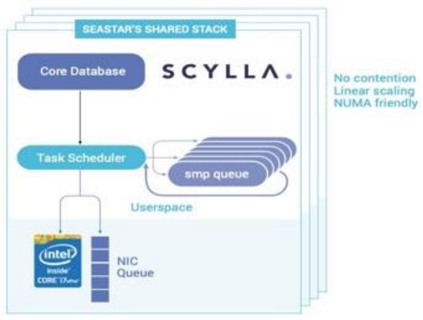


Figure 1. Scylla Architecture

Every shard issues its very own I/O, either to the disk or to the NIC straightforwardly. Administrative tasks, for example, compaction, repair, and streaming are additionally overseen freely by every shard. To manage data on the disk Scylla uses an algorithm called Log Structured Merge Tree (LSM tree). LSM trees create immutable files during data insertion with sequential I/O yielding great initial throughput. Scylla dynamically adjusts priority across tasks in user space, enabling self optimizing operations.

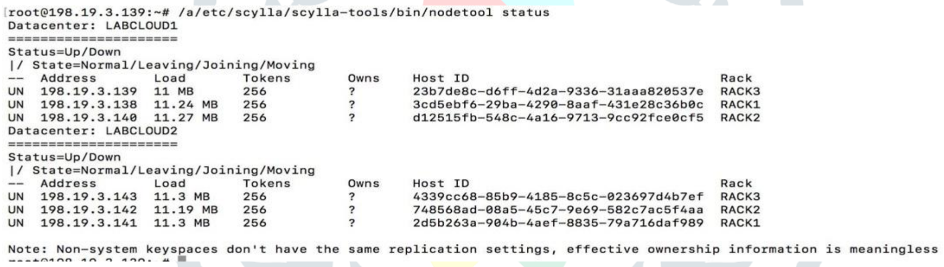
II. STATE OF THE ART

This section looks at the survey made on Scylla. The next few paragraphs discuss previously used approaches. [1] surveys the scalability of Scylla as compared to Cassandra. The parameters checked include OS level metrics such as disk utilization and cache-miss rates. Scylla provides a much better read performance as compared to Cassandra.[2] explores the various databases such as Cassandra, MongoDB, CouchBase and MS SQL server and how they scale to write heavy operations. It is concluded that Cassandra performs the best of the three NoSQL databases by a factor of 4. With a large amount of available NoSQL solutions, it is important to be able to distinguish the fittest database solution for a particular system with respect to its distinctive characteristics. All the continuous development and evolution of non-relational technology, over the past years, has contributed to the constant interest in evaluating NoSQL databases. Moreover, until now, all the available studies were highly focused on the performance testing using standard benchmarks [2]. Although those evaluations provide a basic knowledge of the database behavior, there is no performance guarantee while working in a real enterprise environment, where data and interaction are much more random, unpredictable and hard to model [2].

In recent NoSQL evaluations, the authors focus on different possibilities of adapting NoSQL solutions and using those databases along with other domain-specific technologies, such as clinical decision support systems [4]. In these studies, the authors evaluated different possibilities of integrating NoSQL databases in existing systems where flexibility or big data handling capabilities were needed. They concluded that, according to each system characteristic, NoSQL, in fact, could be a good possibility for data management. In particular, its flexibility and scalability were seen as fitting for the needs of each of these works. One of the drawbacks of NoSQL databases is the learning curve. Another recent trend has been the development of different approaches in terms of data queries. While most developers and DBAs are comfortable and accustomed with SQL, the querying and management of non-relational databases requires more time to learn. More than that, NoSQL technology is known for the non- existence of a querying standard, with different databases having different querying languages. Further reviewing the literature, one can gather that although there have been many evaluations, with a specific focus on synthetic data, there are few evaluations focused on write-heavy datasets.

III. EXPERIMENTAL SETUP

For the testing purpose we setup the cluster with 6 nodes spread across 2 datacenters with each datacenter consisting of 3 nodes each. For each node the physical disk was partitioned to 2 logical disks. The first disk for system data in ext4 filesystem. The second one for Scylla data with RAID0 and XFS file system to enable faster IO operations.



```
root@198.19.3.139:~# /a/etc/scylla/scylla-tools/bin/nodetool status
Datacenter: LABCLOUD1
=====================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address        Load       Tokens       Owns    Host ID                               Rack
UN  198.19.3.139  11 MB      256          ?       23b7de8c-d6ff-4d2a-9336-31aaa820537e  RACK3
UN  198.19.3.138  11.24 MB   256          ?       3cd5ebf6-29ba-4290-8aaf-431e28c36b0c  RACK1
UN  198.19.3.140  11.27 MB   256          ?       d12515fb-548c-4a16-9713-9cc92fce0cf5  RACK2
Datacenter: LABCLOUD2
=====================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address        Load       Tokens       Owns    Host ID                               Rack
UN  198.19.3.143  11.3 MB    256          ?       4339cc68-85b9-4185-8c5c-023697d4b7ef  RACK3
UN  198.19.3.142  11.19 MB   256          ?       748568ad-08a5-45c7-9e69-582c7ac5f4aa  RACK2
UN  198.19.3.141  11.3 MB    256          ?       2d5b263a-904b-4aef-8835-79a716daf989  RACK1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
```

Figure 2. Scylla Cluster state for 6 nodes running on 2 datacenters

IV. EXPERIMENTAL RESULTS

The chapter lists the results of the experiment conducted and the inferences that were made from the testing. The evaluation metrics have been listed and the results have been accordingly listed out in this chapter. The results were obtained from a combination of the information acquired through initial process discovery as well as statistics gathered from multiple runs of in the system.

4.1. Evaluation Metric

Evaluating NoSQL databases is always a tricky task. Evaluating Scylla is no different. On the contrary: while Scylla, by using the Seastar framework, does a brilliant job in isolating itself from most of OS and third-party components performance influences, that same ability may cause existing setups not to use Scylla to its full potential. The way Scylla is configured plays a significant role while evaluating/benchmarking.
The primary evaluation metrics that were used to measure the performance of Scylla are:

- IOPS
- 95th percentile read/write latency
- 99th percentile read/write latency
- Mean read/write latency
- Max read/write latency
- CPU utilization
- Cache hits and misses
- Disk usage

4.2. Performance Analysis

The open source cassandra-stress tool was used for evaluating and load testing. cassandra-stress is a Java-based tool that provides a built-in method to populate test data and perform a variety of workload stress tests. Two key database operations were tested: read operations and write operations. The evaluation metrics mentioned earlier were measured for every test run. Throughput is measured as the number of read (or write) operations per second aka IOPS. Latency is measured by mean, median, max,95th and 99th latency in milliseconds(ms). The tests were run multiple times for both read and write operations, and averages were calculated. Tests were also run using a mix of both read operations (20%) and write operations (80%).
The configuration used for Cassandra-stress writes is as follows:

- The write tests were run with the uniform model. The read and mixed operation tests were run with the Gaussian model 10,000/5,000/1,000.
- 15 CPUs pinned for the single process.
- Continuous evaluation for 4 days(96 hours).
- Consistency level of LOCAL QUORUM.
- Thread count of 20000.
- Throttle of 25000/s
- Replication factor of 3 for each keyspace.

All the tests were run on a cluster with single DC and with multiple DCs as well. Throughput and latency were kept as high priority while performing multi DC operations.

Even within the same client machine, with all else being equal, we saw better results after aggregating more than one process instead of just bumping the number of threads in a single process. That is specially true for cassandra-stress, that will keep a fixed number of socket connections ongoing even at higher thread counts. Scylla will work better with more connections that will allow it to better distribute its load.

Multiple clients were used to perform the benchmarking and they pushed to their limits to test the performance of the cluster. Unsurprisingly, the cluster performed really well with very satisfying throughput and latency. The load distribution and the CPU utilization among the nodes were optimal.

Scylla is designed to use the Seastar framework, which uses the Data Plane Development Kit (DPDK) to drive NIC hardware directly, instead of relying on the kernels network stack. This provides an enormous performance boost for Scylla. Scylla and DPDK also rely on the Linux hugepages feature to minimize overhead on memory allocations. DPDK is supported on a variety of high-performance network devices.

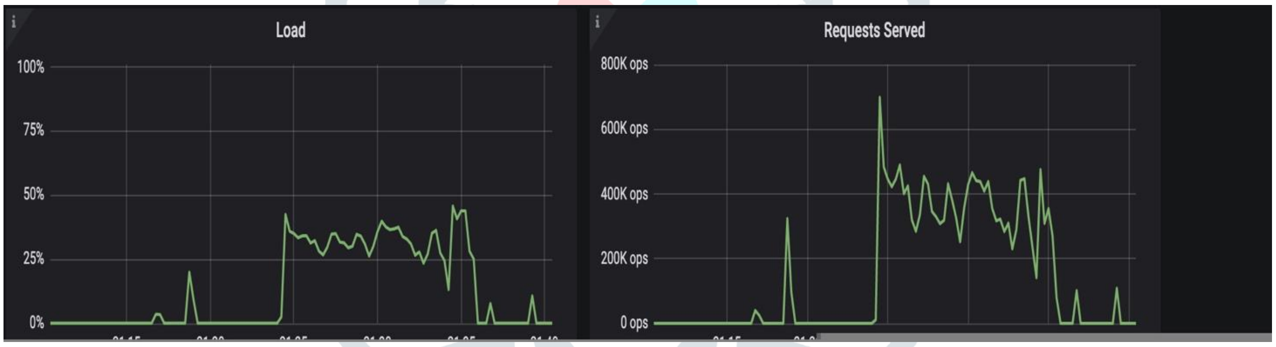


Figure 3. Requests served with 6 nodes

V. Conclusion

We all know that in the past decade, a lot of technologies have emerged to solve the problem of handling of humungous data that is being generated every day. Among these technologies, NoSQL databases have shown who rules the world in the field today. When we look at write heavy databases, we have seen that Scylla has performed really well. It is able to take the heavy loads with ease and with zero down time.

Scylla has met the requirements of achieving 600k operations per second with mere 6 machines. The 99th percentile latency is around 50ms which shows how fast it is considering it is a multi DC cluster.

The only issue we encountered was a drop in throughput at times and it drastically increased the latency for that period of time. But the silver lining was none of the nodes in the cluster went down at any point of time which meant that the application was available at all points of time. It is clear from the graphs that Scylla pushes the CPU utilization to maximum to obtain best performance. It also optimizes the disk usage using the Size-tiered Compaction Strategy (STCS). This compaction strategy makes sure the reads and writes are performed efficiently and doesn't overwhelm the disk.

REFERENCES

[1] Ashraf Mahgoub, Sachandhan Ganesh, Folker Meyer, Ananth Grama, and Somali Chaterji. ”Suitability of NoSQL systemsCassandra and ScyllaDBfor IoT workloads”, 9th International Conference on Commu-nication Systems and Networks (COMSNETS), pp. 476-479, January 2017.

[2] J. R. Lourenco et al.,” NoSQL in Practice: A Write-Heavy Enterprise Application”, IEEE international congress on Big Data, pp. 584-591, 2015.

[3] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, Benchmarking cloud serving systems with ycsb, in Proceedings of the 1st ACM symposium on Cloud computing. ACM, 2010, pp. 143154.

[4] S. K. Gajendran, A Survey on NoSQL databases, University of Illinois, 2012.

[5] Datastax, Benchmarking Top NoSQL databases: A performance comparison for architects and it managers. White Paper.

[6] T. Zhong, K. Doshi, X. Tang, T. Lou, Z. Lu, and H. Li, Big data work-loads drawn from real-time analytics scenarios across three deployed solutions, in Advancing Big Data Benchmarks. Springer, 2014, pp. 97104.

[7] M. Bach and A. Werner, Standardization of NoSQL database languages, in Beyond Databases, Architectures, and Structures. Springer, 2014, pp. 5060.

[8] Scylla,” Building the Real-Time Big Data Database: Seven Design Principles behind Scylla”, White Paper.