Samsung Z-SSD™ and ScyllaDB: Delivering Low Latency and Multi-Terabyte Capacity in a Persistent Database

A comparative study of Scylla performance on Samsung Z-SSD and DRAM

January 2018

SAMSUNG

SCYLLA.

Samsung Z-SSD™ and ScyllaDB: Delivering Low Latency and Multi-Terabyte Capacity in a Persistent Database

Table of Contents

1. Executive Summary

New analysis of Samsung's latency-shattering Z-SSD shows major performance gains when the recently-introduced drive is integrated with the Scylla data store. In this whitepaper, Samsung has closely examined the performance of Scylla [1] — an open-source, high-performance NoSQL data store — on Z-SSD in comparisons with in-memory based architectures. The study also looked at the cost-effectiveness of one solution over the other.

With several vertical optimizations in its embedded NAND, SSD controller and SSD memory, Z-SSD reduces read latency more than 5x compared to today's leading NVMe SSDs.

In our tests, we used Scylla, with its finely crafted C++ implementation, as a drop-in replacement for Cassandra. We compared the performance of Scylla when serving requests from memory and when handling them from Samsung's Z-SSD. To this end, we have characterized system performance as well as overall throughput and latency, using the well-known Cassandra-stress tool [3]. This whitepaper reports our findings, and makes the following observations:

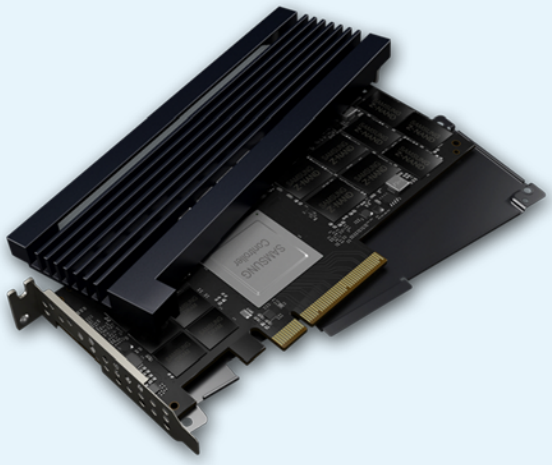
- Samsung Z-SSD has shrunk the performance gap between serving data from memory and having the data served from the SSD.
- When considering maximum throughput with a constraint of sub-millisecond, 95 percentile latency, and with 50% of requests served from Z-SSD, we have found that the gap between Z-SSD and an in-memory run is only 44% for a read-only workload, and only 40% for a mixed workload (75% read, 25% write).
- While Scylla provides a high performance NoSQL service, the overheads of the software stack are considerable – between 6x to 7x compared to the raw device latencies measured with FIO. Further optimizations in the software layer are expected to decrease the gap between SSDs and memory to an even more advantageous level.
- Our cost model for performance-per-dollar provides insights into the cost-efficiency of storage systems and highlights the most noteworthy area where Z-SSD could significantly contribute to the building of server systems that are especially cost-efficient.

2. Introduction

Scylla is a Cassandra-compatible NoSQL data store that has been optimized for integrating with modern hardware in order to deliver higher performance. While supporting the same API as Cassandra to enable drop-in replacement, Scylla takes a radically different design approach [4]. Some of the key features are:

- **Direct access to resources:** Scylla is implemented in C++ to avoid having an external management software layer (JVM) and the ensuing problems. Conversely, Cassandra is implemented in Java and runs inside the JVM. Such a design can suffer from sudden latency hiccups, expensive locking, and low throughput due to low processor utilization.
- **Shared-nothing model:** Scylla is built on top of the Seastar framework [5] for extreme performance in multi-cores. Since locking and coordination among cores will considerably slow down most modern servers, Seastar uses a lock-less share-nothing model to avoid locks for cross-CPU communications.
- **High-performance network I/O:** Scylla efficiently utilizes a DPDK driver framework [6], so that multiple TCP/IP instances can run on each core. By running the network stack in the user space, DPDK eliminates the need for network system calls, costly context switches, and data copying. As the default setting, Scylla uses the Linux TCP/IP stack, which is the setting that we used in our tests.

Z-SSD™



Samsung SZ985 Z-SSD	
Form Factor	HHHL
Interface	PCIe Gen3 x4
NAND	Z-NAND™ Technology
Port	Single
Data Transfer Rate (128KB data size, QD = 32)	
Seq. R/W (GB/s)	3.2 / 3.0
Data I/O Speed (4KB data size, sustained, QD = 32)	
Ran. R/W (IOPs)	750K / 170K
Latency (sustained random workload, QD = 1)	
Ran. R (Typical)	20µs
Ran. R (Best)	12µs
Ran. W (Typical)	16µs
DWPD	30
Capacity	800GB

The Samsung SZ985 Z-SSD [7] is a new ultra-low latency flash storage device that embraces the fundamental structure of Samsung's V-NAND – the industry's leading 3D flash production technology. It offers a unique circuit design and has its own controller, which together serve to maximize performance. The SZ985 can provide 5.5 times lower latency than today's leading NVMe SSDs. Available in an 800GB capacity, the drive has been designed with proven NAND technology for improved reliability, exceptional scalability and greater cost-efficiency. Today's pioneering generation of Z-SSDs can easily be considered the optimal storage solution for latency-sensitive, I/O-intensive applications. Indeed, our results indicate that SZ985 Z-SSD combined with Scylla's high-performance software implementations can deliver competitive performance to an in-memory NoSQL setup.

3. Methodology

We evaluated the performance of Scylla in two scenarios:

- **Scenario 1: when Scylla serves all requests from the memory layer**
- **Scenario 2: when Scylla serves a portion of the requests from the memory**

In the second scenario, we varied the percentage of requests that originated from the memory in a range between 25% and 75%. Under these conditions, the rest of the requests are served from the Z-SSD. As an example, when 25% of the requests are served from memory, the other 75% are served from the Z-SSD. We considered two workloads: read-only and mixed (75% read, 25% write).

Our performance metric aimed to achieve maximum throughput under a given tail latency constraint. In one scenario, we chose our latency target to be sub-millisecond for the 95 percentile latency. In a second scenario, we set the 99 percentile latency to be under two milliseconds.

For testing the corresponding performance of Scylla, we used three Supermicro servers. Each server was equipped with two Intel Xeon Gold 6154 CPUs running at 3GHz (36 CPU cores per node), 128GB of DDR4 memory, and two SZ985 Z-SSD drives.

The servers ran Ubuntu 16.04 with Linux Kernel 4.12 and Scylla version 2.0. We deployed nine client machines to generate the workloads and connected all nodes via a 10GbE network. Our hardware and software settings are summarized in Table 1.

Table 1: Description of the hardware/software stack

Cluster Size	3 nodes
Server	Supermicro 1U (dual socket)
Processors	2 x Intel Xeon Gold 6154 @ 3.00GHz • Overall: 36 cores
Memory	128GB ECC DDR4
Network	10 Gigabit Ethernet
NVMe Storage	2 x Samsung Z-SSD™ (SZ985, 800GB)
Operating System	Ubuntu 16.04.1
Linux Kernel Version	4.12.0
Scylla Version	2.0

More information about the Cassandra-stress command and the related settings is available in the Appendix section.

4. Performance Results

In this portion of our testing, we compared Scylla in-memory with Scylla in the SZ985 Z-SSD. In Section 4.1, we used a read-only workload, and in Section 4.2, we used a mixed workload for our comparisons. In Section 4.3, we identified the amount of overhead created by the software stack (operating system and application).

4.1 Results for the "Read-only" Workload

Figure 1(a) illustrates the throughput while 95-percentile latency is below 1 millisecond and compares these cases with an in-memory configuration (dashed blue line). When only 25% of requests are served from memory (75% are served from Z-SSD), the gap is only 58%. Note that this is an extreme case and that in most real-world scenarios more that 50% of requests are expected to be served from memory. When 50% of the requests are served from memory, the gap shrinks to 44%. When 75% of requests are served from memory, the gap is further reduced to only 23%.

Figure 1(b) illustrates the result of a second use-case where the 99-percentile latency is kept below 2 milliseconds. The gap in this case is even smaller, varying from 51% to 24%.

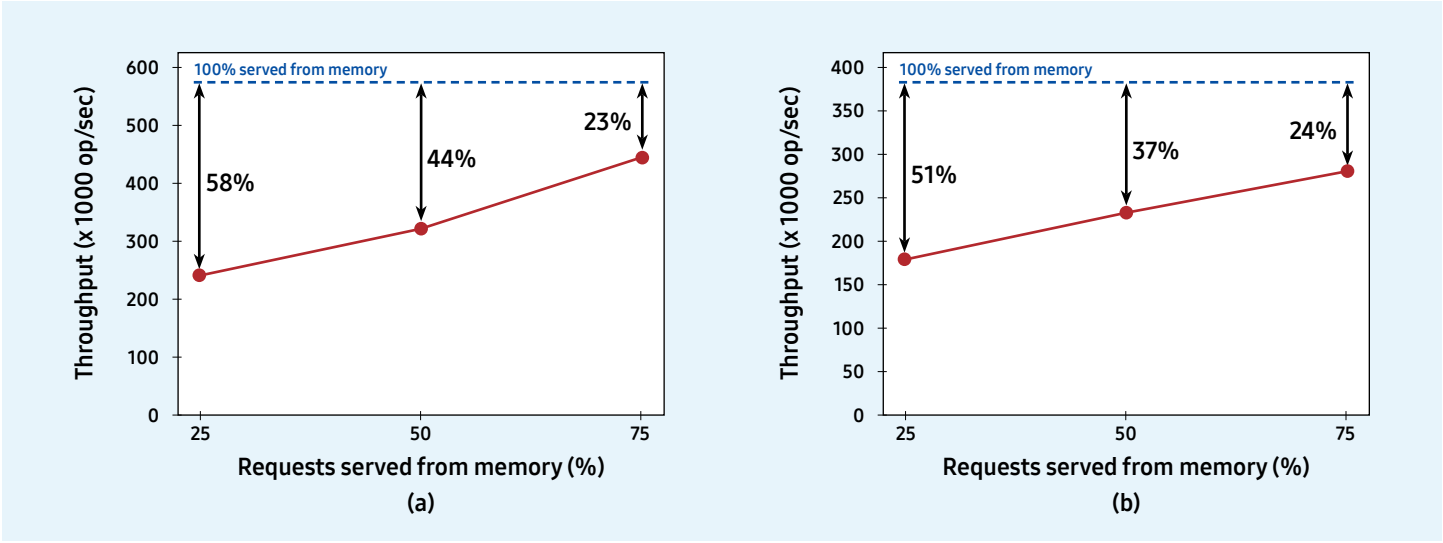


Figure 1: Maximum throughput for read-only workload, serving 25%, 50%, and 75% of requests from memory compared to all in-memory in two cases: (a) Sub millisecond, 95 percentile latency and (b) sub two-millisecond, 99 percentile latency

4.2 Results for the "Mixed Workload" (75% read/ 25% write)

In this section, we used a mixed workload benchmark when measuring Scylla performance. Figure 2(a) depicts the throughput while the 95-percentile latency is below one millisecond and compares these cases with an in-memory case (dashed blue line). When 25% of the requests are served from the memory, the performance gap is 46%. When 50% of the requests are served from memory, the gap shrinks to only 40%, and falls to 29% when 75% of the requests are served from the memory.

Figure 2(b) depicts the result of another noteworthy experiment where the 99-percentile latency is kept below two milliseconds and the maximum attained throughput is monitored. The gap in this case is even smaller and varies from 38% to only 16%. Here, the gap between in-memory and in Z-SSD performance is surprisingly narrow.

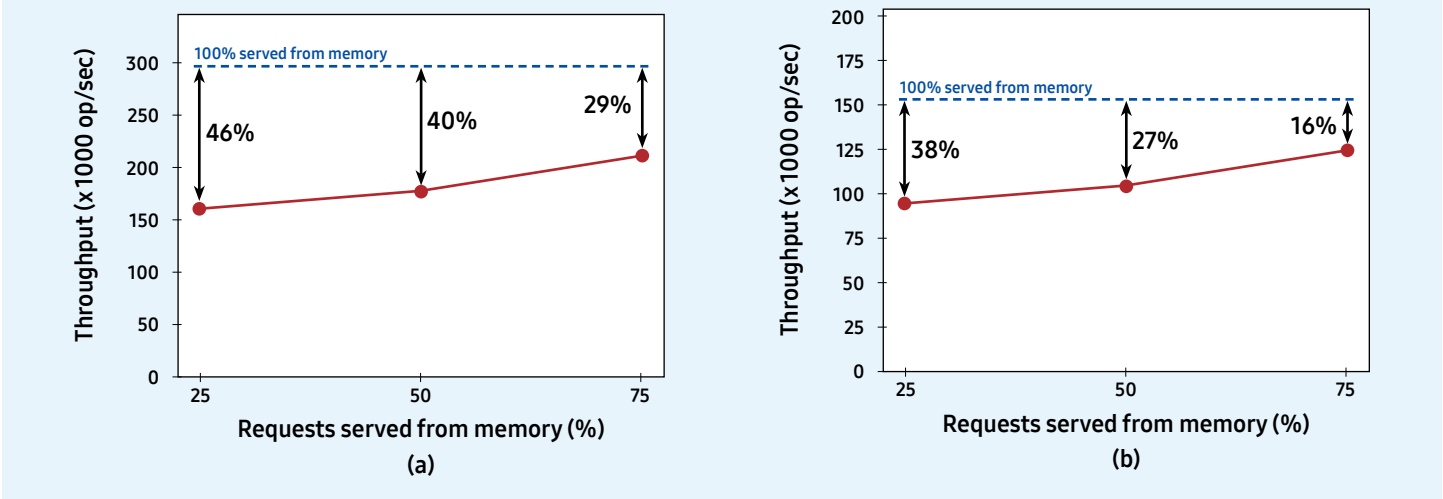


Figure 2: Maximum throughput for a mix workload (75% read/25% write), serving 25%, 50%, and 75% of requests from memory compared to all in-memory in two cases: (a) Sub millisecond, 95 percentile latency and (b) sub two-millisecond, 99 percentile latency

4.3 Software Overheads Imposed by the Software Stack

We also conducted a set of tests to understand the overheads imposed by the software stack. In one test, we ran the FIO (Flexible I/O tester) [8] with queue depths that varied between one and 32.

We measured the throughput, average, and 95-percentile latency [result in Figure 3(a)]. The tail latency ranged from 15 microseconds under a light load to 76 microseconds when the device was fully loaded. In the second test, we fixed the throughput and looked at the 95-percentile latency from the viewpoint of the Scylla servers. This is the read latency when the Scylla server issued requests to the underlying layers [result in Figure 3(b)]. The difference between the two tests revealed the overheads of the software layer. In this case, the overhead varied from 6x to 7.1x depending on the number of requests served from the memory. The overhead imposed by the software stack remains comparatively high; improvements in the software layer will further shrink the gap between Z-SSD and in-memory performance.

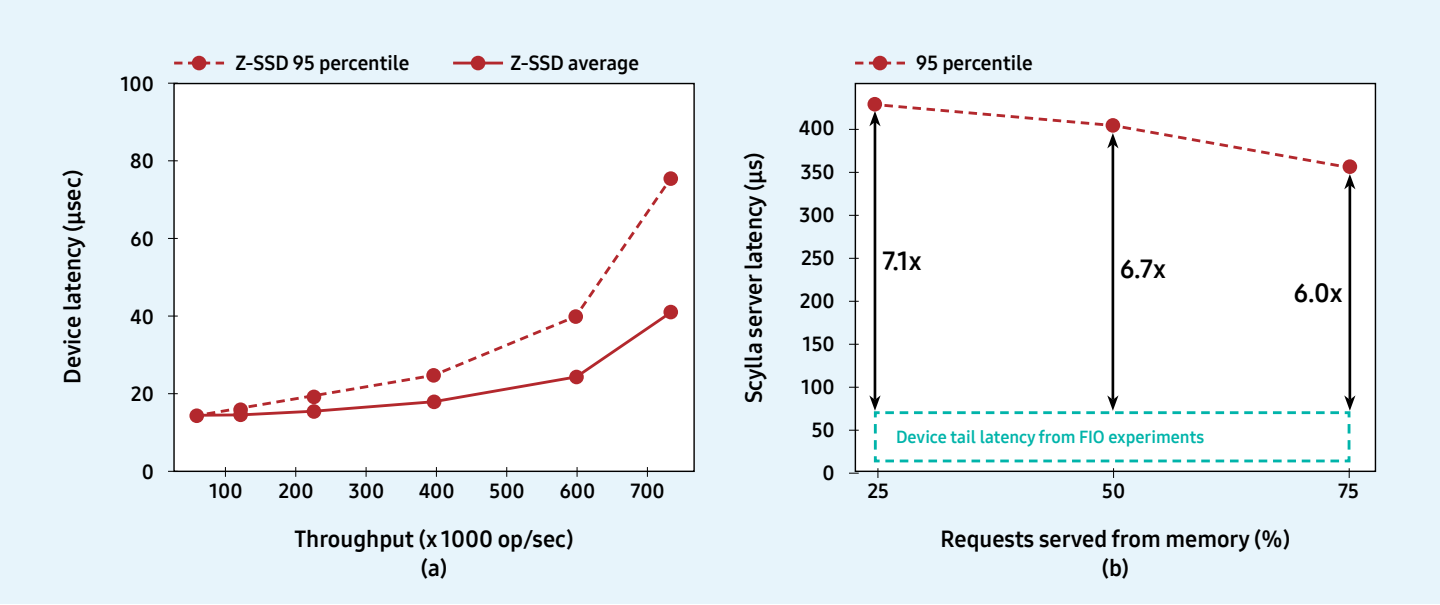


Figure 3: (a) The latency (average and 95-percentile) results for Z-SSD with FIO experiments on a read-only workload, and (b) the 95-percentile latency when the Scylla server accommodates a read-only workload, with the raw device tail highlighted in green.

5. Cost Model

We also developed a cost model to answer the following question: For a given use-case requirement (throughput and database size), what is the most cost-efficient cluster configuration?

In our model, we optimized performance and cost for different use-cases. For each combination of throughput and database capacity requirements, we calculated the cost of three clusters that would meet the requirement. We examined these configurations: an in-memory cluster, a Z-SSD based cluster, and an NVMe-SSD based cluster. We use the performance numbers from Figure 1(a) (all in-memory with 570K op/sec, 50% in-memory for Z-SSDs with 320K op/sec). To derive NVMe-SSD performance numbers, we repeated the test using Samsung PM963 [9] (under the same sub-millisecond constraint for the 95 percentile tail latency, it provided 160K op/sec). Figure 4 shows the configuration that minimizes the cost of the system, for every requirement pair. As one might expect, for a small database with a very high throughput requirement, memory provides the most cost-efficient option (orange area). For a very large database with a low throughput requirement, an NVMe SSD cluster is the preferable choice (green area). The area highlighted with blue corresponds to cases where Z-SSD is the most cost-efficient option. The Z-SSD provides new options to system architects looking to build more cost-efficient systems.

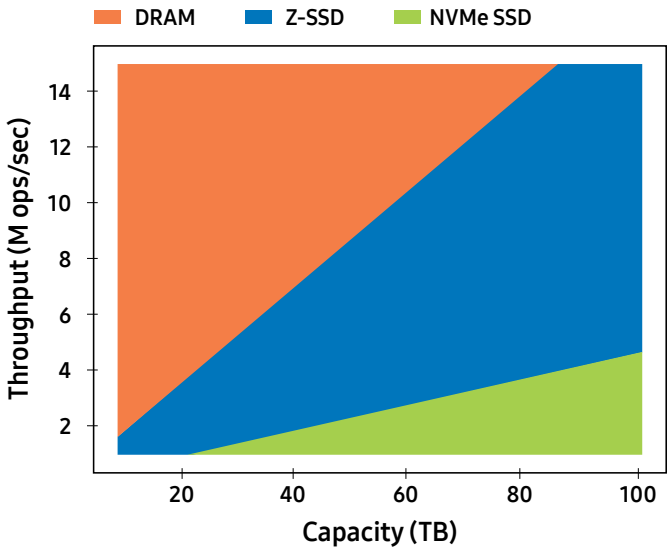


Figure 4: Cost model for different devices

6. Conclusions

In this whitepaper, we evaluated the performance of Scylla — a high-performance NoSQL data store — on a Samsung ultra-low latency Z-SSD. We showed that the Z-SSD sharply shrinks the performance gap between serving data from memory and serving data from an SSD. Considering throughput under latency constraints, we showed that the gap between in-memory Scylla and Z-SSD Scylla is only 44% for a read-only workload and only 40% for a mixed workload. Improvements to the software stack are expected to further shrink this gap, as software layer overheads currently increase latencies by 6x to 7x compared to raw device capabilities.

Finally, we developed a cost model to compare the cost-efficiency of different database system configurations: in-memory, NVMe SSD and Z-SSD. Our cost model provides a great deal of insight into the cost effectiveness of Z-SSD and the significant role that it can now play in building high performance database systems.

Samsung Z-SSD™ and ScyllaDB: Delivering Low Latency and Multi-Terabyte Capacity in a Persistent Database

About Samsung

Samsung inspires the world and shapes the future with transformative ideas and technologies. The company is redefining the worlds of TVs, smartphones, wearable devices, tablets, digital appliances, network systems, and memory, system LSI, foundry and LED solutions. Samsung has been leading efforts to enable the creation of smaller-footprint servers that will decrease data center operational expenses, with Samsung low-latency Z-SSDs, high-density DRAM, and extremely fast NVMe SSD storage drives.

About Scylla

ScyllaDB is the world's fastest NoSQL database. Fully compatible with Apache Cassandra, Scylla embraces a shared-nothing approach that increases throughput and storage capacity to 10X that of Cassandra. AppNexus, Samsung, Mogujie, Outbrain, Kenshoo, Olacabs, Investing.com, Eniro, IBM's Compose and many more leading companies have adopted Scylla to realize order-of-magnitude performance improvements and reduce hardware costs. ScyllaDB was founded by the team responsible for the KVM hypervisor and is backed by Bessemer Venture Partners, Innovation Endeavors, Wing Venture Capital, Qualcomm Ventures, Magma Venture Partners, Western Digital Capital and Samsung Ventures.

For more information, contact:

SCYLLA.
info@scylladb.com

SAMSUNG
msl-inquiry@ssi.samsung.com

7. Appendix

A sample Cassandra-stress command for the read-only workload from our experiments is provided below:

```
cassandra-stress user profile=setting.yaml no-warmup ops\(getdata=1\)
duration=${duration} cl=one -mode native cql3 -pop
dist=gaussian\(${range},${mean},${stdev}\) -node ${server_list} -rate
threads=300 limit=${limit}
```

The content of the setting file *(setting.yaml)* is presented below. The value for *duration* is the length of the experiment (10 minutes). The *cl* argument specifies the consistency level. The *dist* argument sets the Gaussian distribution with a range (minimum and maximum), mean and standard deviation. The node parameter specifies the list of Scylla servers and the *limit* value is used to cap the throughput per client. The values that we used for the statistical distribution parameter and the throughput cap were calculated using empirical search based on the hit ratio observed in the experiments. The actual values depend on the hardware configuration of each system (i.e., CPU core count and frequency, amount of memory, I/O throughput and latency) and need to be tuned per system.

```
#
#  Keyspace info
#
keyspace: sizebase
#
#  The CQL for creating a keyspace (optional if it already exists)
#
keyspace_definition: |
   CREATE KEYSPACE sizebase WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
#
#  Table info
#
table: mytable
#
#  The CQL for creating a table you wish to stress (optional if it already exists)
table_definition: |
   CREATE TABLE mytable (
     col1 text,
     col2 blob,
     PRIMARY KEY ( col1 ))
columnspec:
   - name: col1
     size: fixed(64)
   - name: col2
     size: fixed(1024)     # When you change the column size, you'll be generating less/more data in the system
insert:
   partitions: fixed(1)   # number of unique partitions to update in a single operation
                          # if batchcount > 1, multiple batches will be used but all partitions will
                          # occur in all batches (unless they finish early); only the row counts will vary
   batchtype: UNLOGGED    # type of batch to use
   select: fixed(1)/1     # uniform chance any single generated CQL row will be visited in a partition;
                          # generated for each partition independently, each time we visit it
#
# A list of queries you wish to run against the schema
#
queries:
   getdata:
     cql: select * from mytable where col1 = ?
     fields: samerow      # samerow or multirow (select arguments from the same row, or
                          # randomly from all rows in the partition
```

8. References

[1] "Scylla," Scylla, [Online]. Available: http://www.scylladb.com/

[2] "ScyllaDB and Samsung NVMe SSDs Accelerate NoSQL Database Performance," http://www.samsung.com/semiconductor/insights/tech-leadership/scylladb-and-samsung-nvme-ssds-accelerate-nosql-database-performance/

[3] The cassandra-stress tool, https://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsCStress_t.html

[4] "Architecture of Scylla," [Online]. Available: http://www.scylladb.com/product/technology/

[5] "Seastar," [Online]. Available: http://www.seastar-project.org/

[6] "Data Plane Development Kit (DPDK)," Intel, [Online]. Available: http://www.intel.com/content/www/us/en/communications/data-plane-development-kit.html

[7] "Ultra-Low Latency with Samsung Z-NAND SSD," http://www.samsung.com/us/labs/pdfs/collateral/Samsung_Z-NAND_Technology_Brief_v5.pdf

[8] "FIO: Flexible I/O Tester," [Online]. Available: https://github.com/axboe/fio

[9] "Samsung PM963," https://www.samsungforpartners.com/pdf/products/PM963_Whitepaper.pdf