Apache Cassandra* with Intel® In-Memory Analytics Accelerator

Apache Cassandra* with Intel® In-Memory Analytics Accelerator (Intel® IAA) achieves 22% higher performance compared with the Zstandard (zstd) compression algorithm and 24% higher storage capacity compared with the LZ4 compression algorithm.

Authors

Binuraj Ravindran
Principal Engineer

Shylaja Kokoori
Cloud Software Development Engineer

Kalyan Jee
Cloud Solution Architect

Nandhita Narendra Babu
Cloud Software Development Engineer

Sevanthi Suvarna Reddy
Cloud Software Development Engineer

Tony Ruiz
Cloud Software Development Engineer

In the realm of data-driven decision-making, organizations and businesses increasingly rely on robust data-analytics frameworks to extract valuable insights from vast datasets. Intel® IAA[1,2] is one of the Intel® Analytics Engines in 4th generation Intel® Xeon® Scalable processors[3] and provides a range of benefits including increased application performance, reduced costs, and improved power efficiency for data-analytics workloads.

This paper highlights performance improvements and cost savings that Intel IAA can provide when integrated with Apache Cassandra, an open source NoSQL distributed database.

Cassandra with Intel IAA achieves a 22% higher throughput compared to zstd-based software solutions for 80:20 read/write with 16 KB block sizes and maintains p99 latency (99th percentile of latency) in the same range or lower. Compared with a zstd-based software solution at the same throughput, the cost-of-compute can be reduced. The compression and decompression operations can be offloaded to Intel IAA to save CPU cores.

In contrast to a LZ4-based software solution, Cassandra with Intel IAA demonstrates the ability to increase the storage capacity by 24% while maintaining similar throughput and p99 latency.

Data Analytics with Intel IAA

Data analytics involves using a database to uncover valuable information that addresses an organization's business requirements and requires extensive scanning of large datasets, efficient data storage, and seamless data movement across memory tiers to attain timely and cost-effective query results. Achieving these outcomes demand faster data-scan (query processing) and compact data-storage.

The 4th generation Intel Xeon Scalable processor integrates multiple accelerators to deliver enhanced performance across various established and emerging workloads. Depending on the CPU model, the number of accelerators and instances may vary, which offers cost-effective matching of diverse workloads at data centers.

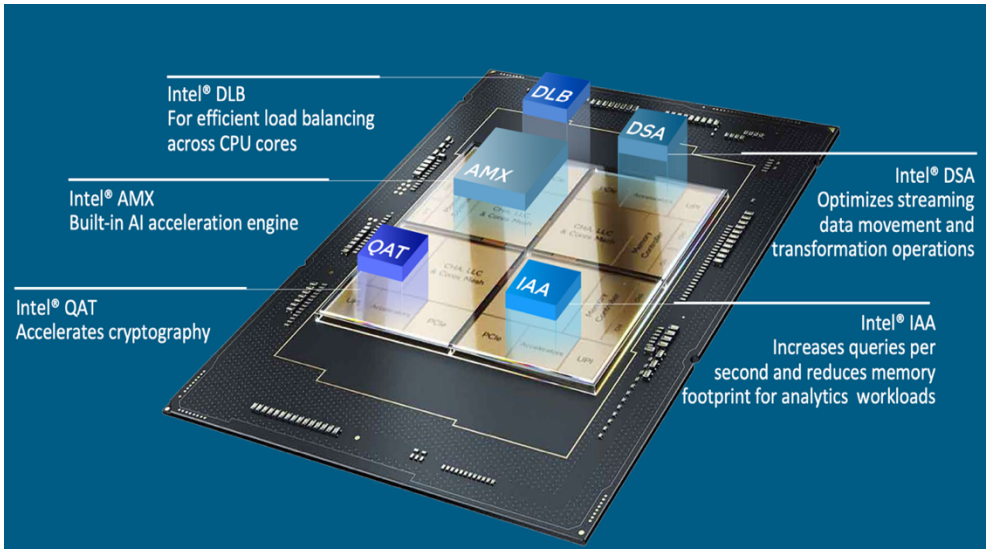

Figure 1. Intel Accelerator Engines in 4th generation Intel Xeon Scalable processors

Figure 1 illustrates the various analytics engines in the Intel Xeon Scalable processor including Intel IAA, which can offload the following operations from the CPU:

- Query and scan function
- Compression and decompression

The rest of the paper describes how Intel IAA can be integrated into databases like Cassandra to achieve performance improvements and cost savings.

Cassandra Architecture

Cassandra is a row-oriented database, designed to handle large amounts of unstructured data, with high availability and fault-tolerance. The storage engine in Cassandra is based on a log-structured merge-tree (LSM) model and enables faster writes.

Figure 2 illustrates the write and read path architecture in Cassandra.

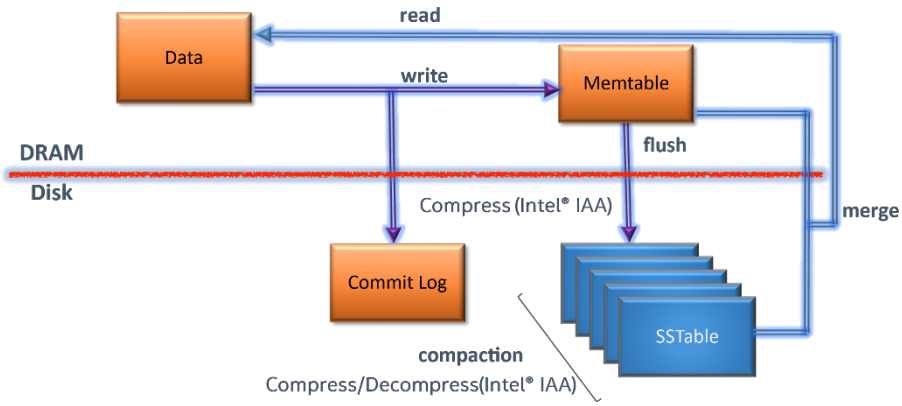

Figure 2. Apache Cassandra high-level architecture

When a client issues a write request, an incoming request is first appended to a commit log for durability and added to an in-memory data structure, memtable. As memtables fill up, they are flushed to storage and persisted in files, called Sorted String Tables (SSTables). SSTables typically store data compressed with a configurable block size (the current default is 16 KB). These SSTables are immutable files and are regularly compacted to clean up redundant updates and deletions. The compaction process involves decompressing data from multiple source files, compacting them, and recompressing them into new SSTables.

For a read request, if the row cache in Cassandra is enabled, data is returned immediately from the cache. Otherwise, data from the memtable and potentially multiple SSTables (if data updates are present) are retrieved, decompressed, and merged before providing a response back to the client.

Through the ICompressor interface[4] in Cassandra, Intel has added a new compressor class, QPLCompressor[5] that allows offloading of compression and decompression operations to Intel IAA. Clients can create tables in Cassandra using QPLCompressor for flush and compaction similar to other software compressors like LZ4 and zstd.

QPLCompressor for the Cassandra Storage Engine

QPLCompressor enables Cassandra to perform compress and decompress operations using Intel IAA. This is achieved using:

Intel® Query Processing Library (Intel® QPL)[6]
This is a low-level library and software API that supports the capabilities of Intel IAA. It provides extremely high throughput compression and decompression combined with primitive analytic functions, and highly optimized software fallback for other Intel CPUs. Intel QPL primarily targets applications such as big-data and in-memory analytic databases.

Java* Native Interface Binding for Intel® Query Processing Library (JNI binding for Intel® QPL)
This library allows Java* applications to communicate with Intel QPL for using Intel IAA hardware. It improves the application performance by accelerating operations like compression, decompression, and analytics.

As illustrated in Figure *3*, Intel QPL sits on top of the system drivers. Applications use Intel QPL to interface with Intel IAA and offload analytics operations to the Intel IAA device. Source code for the new compressor implementation,[5] JNI binding [8] and low-level library[6] are all available publicly. For databases that are developed in C++, unlike Cassandra, developers can call Intel QPL APIs directly to offload compression and decompression operations.[7]

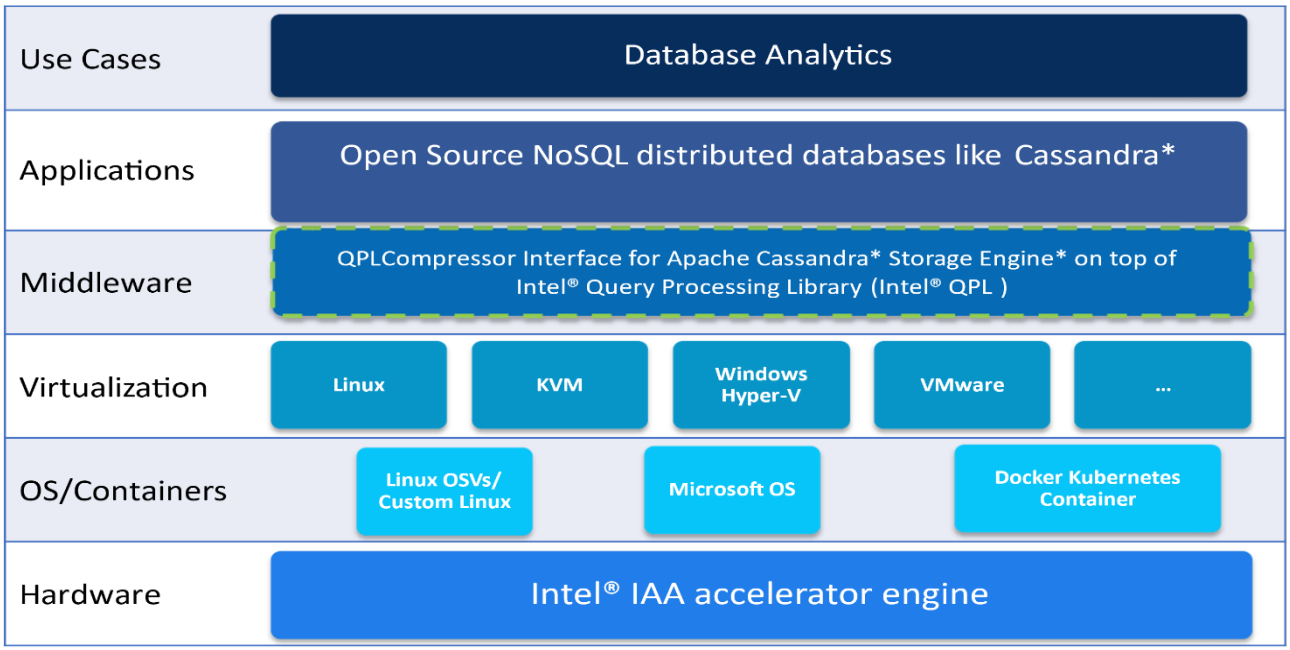

Figure 3. Intel QPL unlocks Intel IAA and applications.

Performance Capture Methodology

For performance measurements, Intel used NoSQLBench,[9] a performance testing tool for NoSQL frameworks. NoSQLBench includes many different parameters to customize the workload conditions and Cassandra options.

Settings for NoSQLBench

Benchmarks:

- 80:20 read/write
- The database is populated using a NoSQLBench cql-timeseries workload.
- The read/write workload exercises both compression and decompression paths

NoSQLBench Instances:

- Read/write:
  - 8 instances
  - Increase the client load (NoSQLBench thread count per instance) till the 99th latency percentile is 20 ms
  - Block size 16 KB
  - Data size 1.1 billion entries per database size varies between different compression types
  - Row cache disabled; no uncompressed data is cached.

Compressor

- zstd, LZ4, Intel IAA
  - zstd: v1.5.4. zstd is a reference to compare Intel IAA performance.
  - LZ4: 1.8.0. LZ4 is provided to show another popular speed versus compression ratio tradeoff.
  - Intel QPL for Intel IAA v1.3.1

The tests were run on a dual-socket 60-core system with Intel® Hyper-Threading Technology (Intel® HT Technology) enabled. The NoSQLBench benchmarks were run on a separate client system and all eight Cassandra server instances were run on the server with four Cassandra instances bound to one socket and the other four bound to the second socket on a two-socket system. All four Intel IAA devices available on each socket were enabled with one SWQ (shared work queue) setup per device. A client/server configuration is provided in Figure 4.
For additional hardware and software configurations, see Configuration 1.

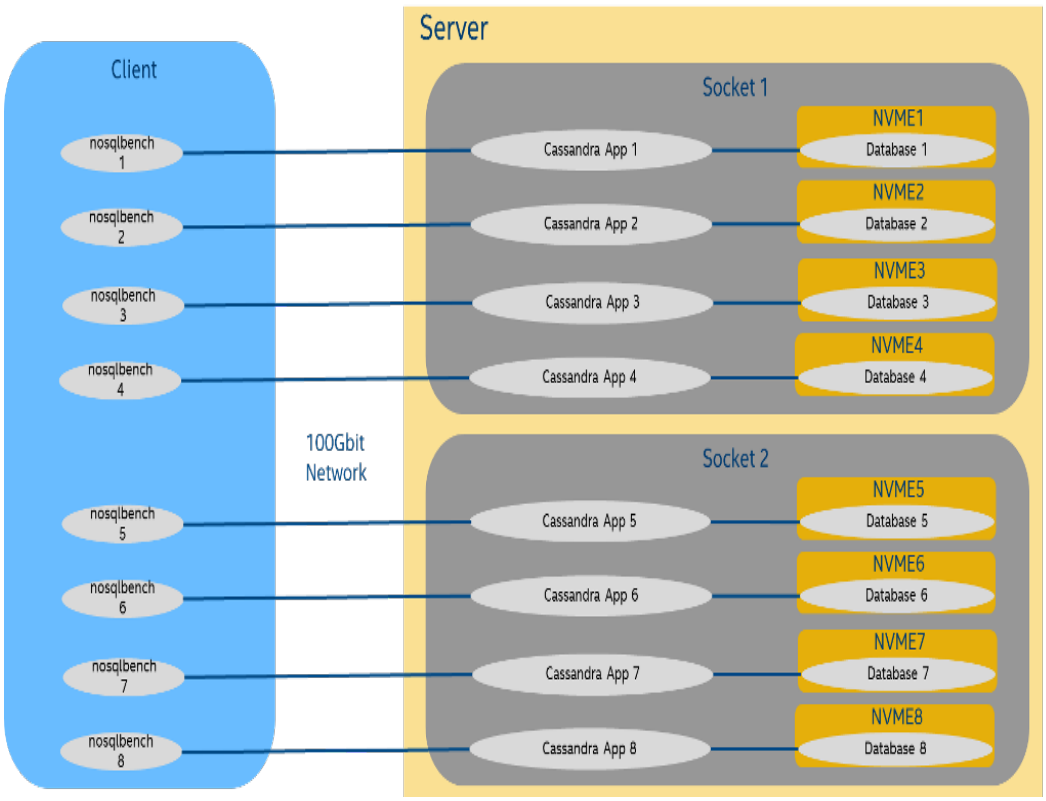

Figure 4. Client/Server configuration

Performance Improvement and Storage Size Reduction with Intel IAA

Cassandra with Intel IAA can achieve higher throughput compared to zstd-based software compression for 80:20 read/write scenario.

- **Throughput improvement:** 22%
- **Compressed-data-size tradeoff:** 9%

Figure 5 shows the relative improvement in throughput versus zstd across different compressor types. Compared to zstd, the higher throughput with Intel IAA compression allows increased application performance or reduced cluster sizes. This leads to improved resource use and cost savings.

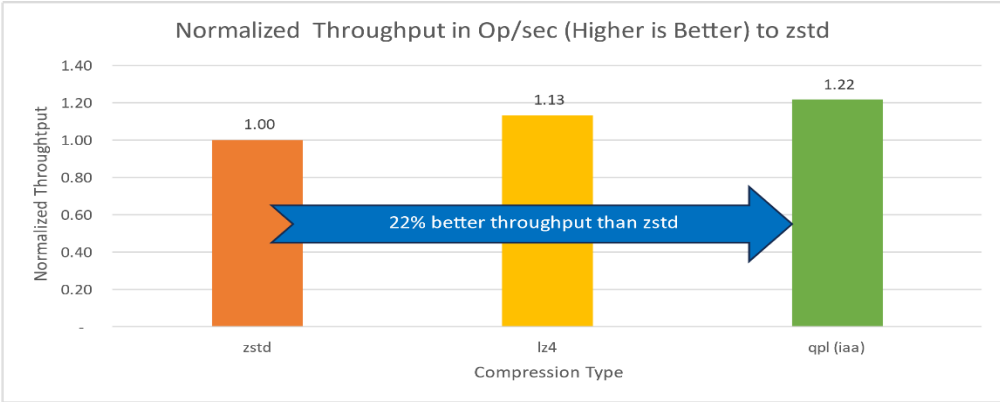

Figure 5. Relative throughout across compressors compared with zstd

The relative decrease in storage capacity enabled with Intel IAA is depicted in Figure *6*. Compared to the LZ4 compressor, Apache Cassandra with Intel IAA can achieve:

- **Database size reduction:** 24%
- **Throughput improvement:**7%

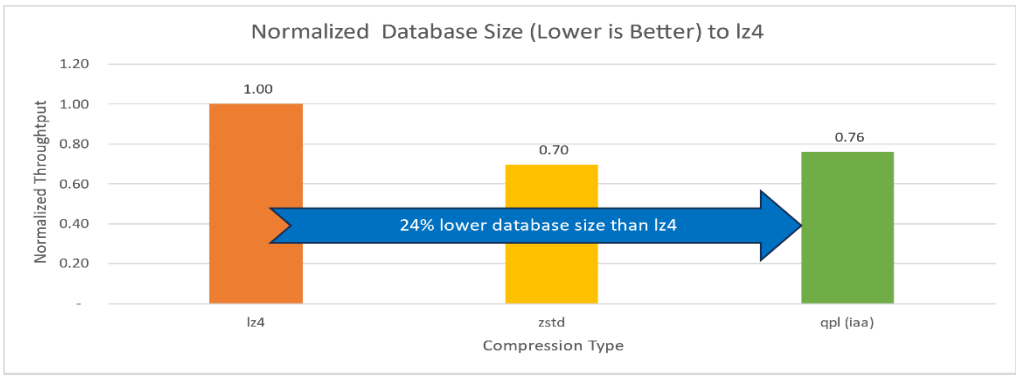

Figure 6. Relative database size across compressors compared with LZ4

Reduction in data size for the similar throughput can significantly increase the storage capacity, especially for scenarios with storage capacity bottlenecks.

There exists a tradeoff between speed and size for any compression algorithms. Intel IAA offers a good compromise between zstd and LZ4, where a higher throughput can be achieved compared with zstd and a much smaller database size can be obtained compared with LZ4, depending on whether speed or size is more important in the design.

When compression/decompression operations are offloaded to Intel IAA, less CPU resources are needed to achieve the same performance. The freed-up CPU resources can be used either for reducing the overall cost of a Cassandra server instance or for running other tasks.

Performance Scaling

The scenarios covered in the previous sections are for scenarios when all the CPUs in the socket were used. Figure 7 shows the results from a performance scaling case study across the number of cores with zstd/LZ4 software compression and across number of Intel IAA devices (one and four Intel IAA devices).

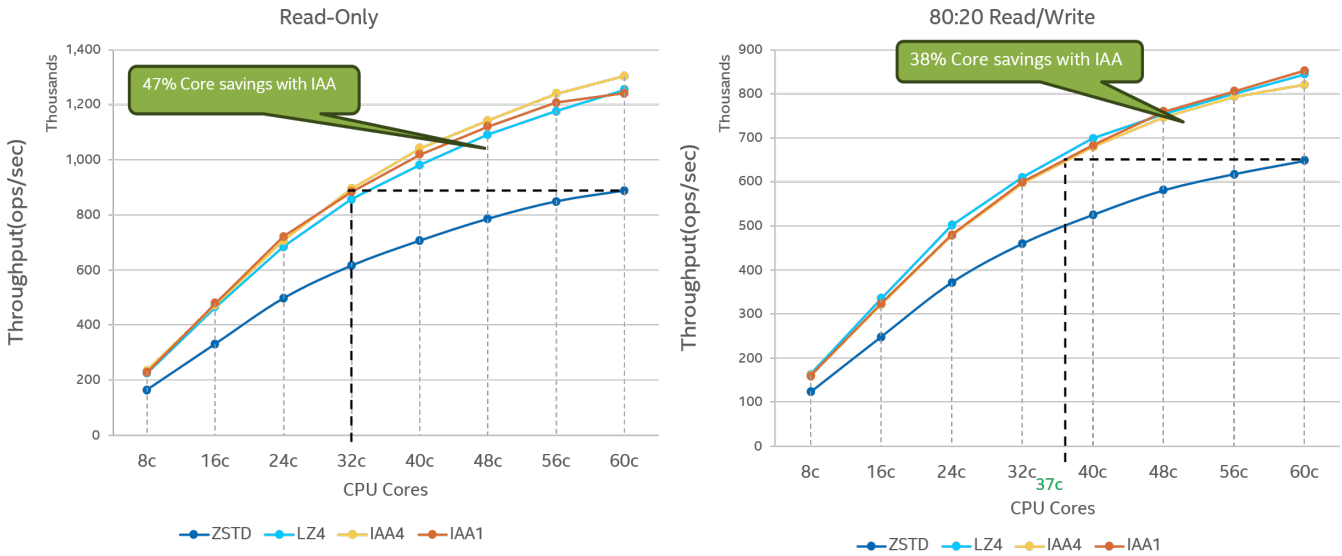

Figure 7. Performance scaling across CPU cores and Intel IAA devices

The horizontal dotted line represents the maximum throughput achieved with zstd. The vertical dotted line can be used to estimate the number of CPU cores that can be saved by offloading compression/decompression operations to Intel IAA while achieving the maximum zstd throughput. Intel IAA can reduce the number of CPUs needed to reach the same throughput as zstd at a lower latency. In this example Cassandra with Intel IAA can achieve the same throughput as that of zstd-based Cassandra with 47% reduction in CPU cores in a 100% read scenario and 38% reduction in CPU cores for an 80:20 read/write scenario.

As the percentage of the time spent in compression/decompression operation is less in Cassandra, increasing the number of Intel IAA devices beyond two may not increase the throughput much. However, there are ways to improve the throughput by breaking down the larger block sizes to smaller ones, thus parallelizing the operations, which is under investigation.

This case study uses a scaled-down setup to eliminate the network overhead as in Configuration 1. Cassandra Server is running on Socket 0 and NoSQLBench clients are running Socket 1 of a two-socket system. More details of the hardware and software configurations are in Configuration 2. The idea is to showcase how the performance scales with the number of CPU cores and the potential CPU savings with Intel IAA rather than a full deployment scenario. Full deployment scenario is covered in a previous section using a much larger data size and with across the network configurations.

Software configuration details for this simplified case study are as follows:

- **Benchmarks:** Read-only and 80:20 read/write:
  - The database is populated using a NoSQLBench cql-timeseries workload.
  - Read-only test exercises for only decompression.
  - Read/write workload exercises for both compression and decompression.
- **NoSQLBench instances:**
  - **Read only:** Four instances; 200 threads per instance
  - **Read/Write:** Four instances; 200 threads per instance plus background flush/compaction threads.
- **Block size:** 16 KB
- **Data size**: 10 million entries per database, 10 G uncompressed size per database
- **Row cache disabled:** No uncompressed data is cached.
- **Compression:** zstd, LZ4, Intel IAA.
  - **zstd:** v1.5.4
  - **LZ4:** 1.8.0
  - **Intel QPL for Intel IAA:** v1.1.0

Conclusion

Intel Analytics Engines represent a significant leap in innovation within the 4th gen Intel Xeon Scalable processors.[3] Among these accelerators, the Intel In-Memory Analytics Accelerator (Intel IAA) is designed to deliver exceptional throughput and low-latency compression/decompression capabilities alongside primitive analytical functions.

Intel IAA enhances the performance of a Cassandra NoSQL database, resulting in increased application speed, reduced costs, and lower power. Compared with zstd-based software solutions, Cassandra with Intel IAA achieves a 22% higher throughput for 80:20 read/write with 16 KB block sizes, maintaining p99 latency (99th percentile of latency) in the same range or lower. The compression and decompression operations are offloaded to Intel IAA, thereby saving CPU cores and power.

Furthermore, Cassandra with Intel IAA demonstrates the ability to increase the storage capacity by 24%, compared with LZ4-based software solution at the same time maintaining similar throughput and p99 latency.

Additionally, widely used databases such as Cassandra, Kafka*, and OpenSearch* can take full advantage of the capabilities of the qpl-java library. Due to the seamless interfaces provided by Intel QPL and qpl-java from Intel, these applications can offload compression, decompression, and analytics operations to the Intel IAA device, thereby supercharging application performance and improving overall power efficiency.

References

1. Intel Corporation, "Intel In-Memory Analytics Accelerator (Intel IAA) Architecture Specification," 2023. [Online]. Available: https://www.intel.com/content/www/us/en/content-details/721858/intel-in-memory-analytics-accelerator-architecture-specification.html.
2. Intel Corporation, "Intel In-Memory Analytics Accelerator (Intel IAA) User Guide," 2023. [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/780887.
3. Intel Corporation, “Drive More Data Value Through Acceleration," 2023. [Online]. Available https://www.intel.com/content/www/us/en/products/docs/processors/xeon-accelerated/analytics-accelerators-product-brief.html.
4. “Cassandra Documentation,” 2024. [Online]. Available: https://cassandra.apache.org/doc/4.1/cassandra/operating/compression.html#advanced-use.
5. Intel Corporation, "Cassandra for Intel QPL Compressor," 2022. [Online]. Available: https://github.com/intel-staging/cassandra/tree/QPLCompressor.
6. Intel Corporation, "Welcome to Intel QPL Documentation! — Intel QPL v1.1.0 Documentation.," 2023. [Online]. Available: https://intel.github.io/qpl/.
7. Intel Corporation, "Intel® In-Memory Analytics Accelerator Plugin for RocksDB* Storage Engine (Intel® IAA Plugin for RocksDB* Storage Engine)," 2023. [Online]. Available: https://cdrdv2-public.intel.com/788607/357191EN-Intel-IAA-Plugin-RocksDB.pdf
8. Intel Corporation, "Java Native Interface Binding for Intel Query Processing Library," 2022. [Online]. Available: https://github.com/intel/qpl-java.
9. “No SQL Benchmark Tool,” 2023. [Online]. Available: https://docs.nosqlbench.io/.
10. TWiki @ Cern, “CostEst < Main < TWiki.” 2017. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/Main/CostEst

Configurations

Configuration 1

Server: 1-node, 2x Intel® Xeon Platinum 8490H CPU, 60 cores, Intel® HT Technology on, turbo on, non-uniform memory access (NUMA) 4, integrated accelerators available [used 0]: Intel® Dynamic Load Balancer (Intel® DLB) 8 [0], Intel® Data Streaming Accelerator (Intel® DSA) 8 [0], Intel® In-Memory Analytics Accelerator (Intel® IAA) 8 [8], Intel® Quick Assist Technology (Intel® QAT) 8 [0], total memory 512 GB (16x32 GB DDR5 4800 MT/s [4800 MT/s]), BIOS EGSDCRB1.SYS.0102.D37.2305081420, microcode 0x2b0004d0, 1x Ethernet controller I225-LM, 4x Ethernet controller X710/X557-AT 10GBASE-T, 2x Ethernet controller E810-C for QSFP, 1x 1.5T Intel® Solid State Drive (Intel® SSD) SC2BB01, 1x 447.1G Intel SSD SC2BW48, 1x 1.7T Intel SSD SC2KB01, 8x 1.5T Intel SSD PF21Q016TB, Ubuntu* 22.04.3 LTS, 5.19.17-051917-generic, workload and version, compiler, libraries, other software, score=?UNITS.

Client: 1-node, 2x Intel Xeon Platinum 8380 CPU at 2.30 GHz, 40 cores, Intel HT Technology on, turbo on, NUMA 4, integrated accelerators available (used): integrated accelerators available [used 0]: Intel® Dynamic Load Balancer (Intel® DLB) 0 [0], Intel® Data Streaming Accelerator (Intel® DSA) 0 [0], Intel® In-Memory Analytics Accelerator (Intel® IAA) 0 [0], Intel® Quick Assist Technology (Intel® QAT) 0 [0], total memory 512 GB (16x32 GB DDR4 3200 MT/s [3200 MT/s]), BIOS SE5C620.86B.01.01.0005.2202160810, microcode 0xd0003a5, 2x Ethernet Controller X710 for 10GBASE-T, 1x 223.6G KINGSTON SA400M8240G, 4x 3.5T Intel SSD PF2KX038TZ, Ubuntu 22.04.2 LTS, 5.15.0-88-generic

Software configuration: Cassandra 4.1.1, OpenJDK version 11.0.20.1 2023-08-24, Intel QPL 1.3.1 , QPL-java 1.0.0 , NoSQLBench 4.15.102

Test by Intel as of 02/29/24.

Configuration 2

1-node, 2 sockets 4th Intel® Xeon Scalable processor (60-cores, 4x Intel IAA devices) preproduction platform with 512 GB (16x32 GB DDR5-4800MT/s) total memory, Intel HT Technology on, turbo on. BIOS: EGSDCRB1.86B.9409.P15.2301131123.OS: CentOS* Stream 8, Linux* kernel: 6.1.12

Software configuration: GCC* version 8.5.0 20210514 (Red Hat* 8.5.0-13), NoSQLBench version 4.15.101, OpenJDK version 11.0.18 2023-01-17 LTS, Intel QPL 1.1.0 synchronous API. Used loopback for network and dataset fits completely in memory. Server running on socket 0 and client on socket 1.

Testing by Intel as of March 23,2023.

Notices & Disclaimers