Micron® Technical Brief: Comparing HSE 2.0 to HSE 1.9

A STEP FORWARD FOR OPEN SOURCE STORAGE ENGINES

Legacy storage engines from the hard disk drive era are not tailored to optimize for performance when using modern technologies like SSDs and storage class memory (SCM).[1]

Micron Heterogenous-Memory Storage Engine (HSE)[2] is designed from the ground up to accelerate Linux® workloads using flash-based storage and storage-class memory.

HSE 2.0 is built on this core tenet of HSE 1.0 and brings significant improvements to HSE users: ease of deployment, more integration options, a more collaborative development environment, and superior performance.[3]

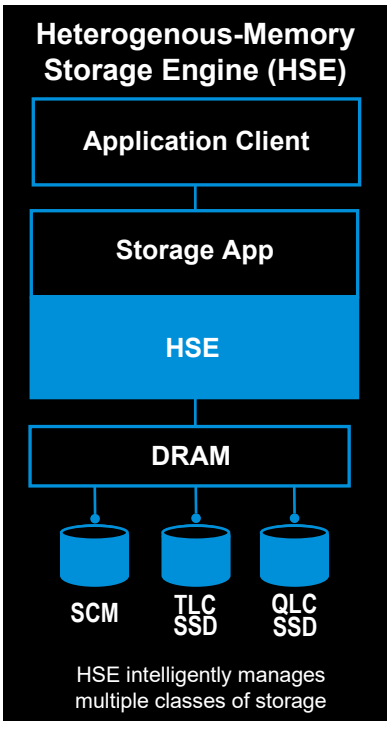

Figure 1: HSE Block Diagram

HSE 2.0 Features

Micron is advancing its next-generation, open-source storage engine – HSE 2.0 – to simplify deployment and offer more integration options, a more collaborative development environment and enhanced performance.

Ease of Deployment

- Runs entirely in user space with no kernel module modifications
- Configuration and logging aligned with best practices for containerized and cloud native apps[5]
- Designed to work with popular Linux 64-bit distributions[6]

More Integration Options

- Designed to run on common Linux file systems[7]
- Easier to integrate with various storage applications
- Native C library with Python™ language bindings

Collaborative Development Environment

- All code changes go through GitHub pull requests (PRs)
- All major changes documented in a dedicated request for comment (RFC) repository
- No contributor licensing agreement (CLA) required
- All new tests published in the project repositories

Enhanced Performance

- Higher throughput for NoSQL workload
- Lower average read latency

1. A type of memory combining the speed of conventional DRAM with a backup power source: https://www.techtarget.com/searchstorage/definition/storage-class-memory
2. Additional operational details are available at www.micron.com/hse
3. In this document, 'performance' means database operations per second, average read latency, or both; 'throughput' means database operations per second. All comparisons statements are relative to HSE 1.9.
4. Workloads tested are similar to workloads A, B, C, and F present on the Github YCSB Core Workloads site: https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads
5. According to *Best practices for operating containers* (https://cloud.google.com/architecture/best-practices-for-operating-containers)
6. For example: RHEL® 8 and Ubuntu 18.04
7. For example: XFS and ext4

Micron® Technical Brief: Comparing HSE 2.0 to HSE 1.9

Overview

This technical brief highlights the superior performance and lower average read latency using HSE 2.0 when compared to HSE 1.9 using three common NoSQL workloads.[4]

Although this document focuses on HSE 2.0 performance and average read latency improvements compared to HSE 1.9, these improvements should be understood in the context of the ease of deployment, additional integration options and the more collaborative development environment features of HSE 2.0.

For each NoSQL workload, HSE 2.0 and HSE 1.9 maximum throughput is compared. Database load rates (and, in-turn, throughput) varies with the number of threads used during the test (#Threads). Because of this relationship, several #Threads values were tested and compared. The highest throughput (peak) results for all tested #Threads is shown. Observed differences are expressed as a multiplier, for example a 0.1X improvement is shown as 1.1X (1.1 times the reference throughput).

Average read latency comparisons also consider throughput. For a desired average read latency range (or maximum value), one would naturally seek the highest throughput available while remaining at or below that average read latency upper limit. To facilitate this analysis, throughput versus average read latency figures have horizontal reference lines added (each marking an average read latency cap).

HSE 1.9

Micron architects, designs and manufactures an extensive portfolio of DRAM and SSDs, and we also have a long history of workload testing, which means we have the expertise and insight and expertise to build a storage engine/key-value store that intelligently manages data placement across disparate memory and storage media types.[7]

HSE 2.0

Micron's most recent HSE release, HSE 2.0, builds on the success of HSE 1.9, with the following additions:

- Simplified deployment
- More integration options
- A more collaborative development environment
- Higher performance

The Micron 7400 SSD With NVMe™

The Micron 7400 SSD is optimized for mainstream NVMe SSD workloads, including NoSQL and SQL databases, block and object stores, VDI and server virtualization, and cloud storage.[8]

Tested NoSQL Workloads

We tested four NoSQL workload results across a wide array of thread counts (#Threads from 4 to 256),[6] showing performance (across measured #Threads), performance at each #Threads and average read latency. We found that HSE 2.0 showed higher peak performance and lower average read latency on every workload tested.[9]

Caching User Profiles

Caching user profiles is 100% read (data does not change). Examples include reading immutable data for user authentication or reading a profile cache (when a user or system profile was created elsewhere).

Tagging Existing Assets

This workload is read-heavy with approximately 95% read storage IO and 5% write. Examples of this workload include adding metadata to existing data (tagging). Most of the tags are read while a few are written (or rewritten).

Users Modifying Records

This is a 50% read/50% write workload that reads a record, modifies it and then writes it back (read-modify-write). This workload models common database and user activity.

Recording User Sessions

This is an update-heavy workload with about 50% of all storage IO being written. Examples of this workload can be seen when user sessions are recorded.

8. For more details on the Micron 7400 SSD, see www.micron.com/7400
9. We tested 4 to 256 threads with each workload to represent a broad variety of database tuning options and activity levels. Actual #threads may vary from those shown.

A Closer Look at HSE 2.0 and HSE 1.9 Performance

A storage engine is a critical software component of a storage application. It manages data on the underlying memory and storage devices. HSE was designed to maximize the capabilities of new storage technologies by intelligently and seamlessly managing data among multiple storage classes. HSE 2.0 offers significantly improved performance and decreased latency.

Caching User Profiles	This workload is 100% read (data does not change). Examples include reading immutable data for user authentication or reading a profile cache (when a user or system profile was created elsewhere).

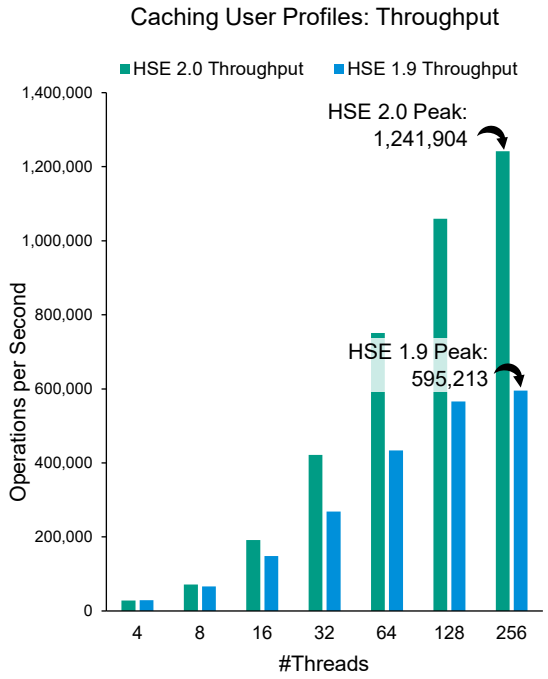

Figure 2a: Caching User Profiles: Throughput

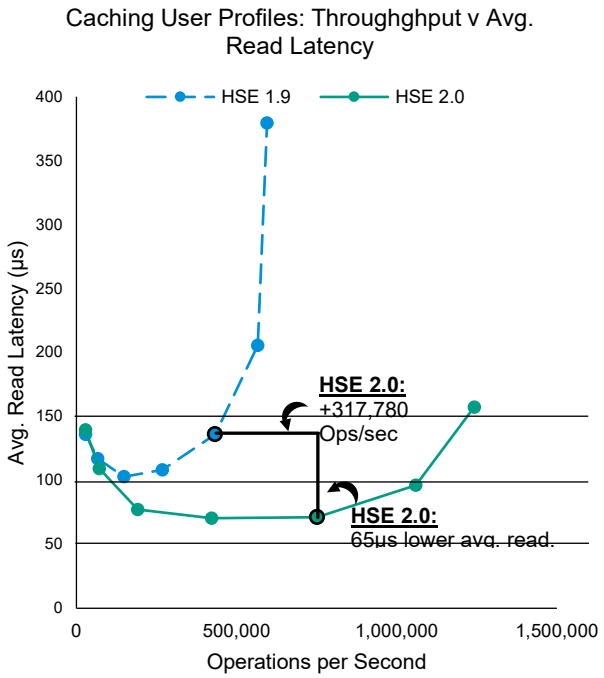

Figure 2b: Tagging Existing Assets: Avg. Read Latency

Caching User Profiles

Throughput

Figure 2a shows that for all tested #Threads, HSE 2.0 demonstrates superior performance compared to HSE 1.9.

The performance difference is low at very small #Threads (<16) and grows as #Threads increase to a maximum of 2.1X. HSE 2.0 reached 1,241,904 peak operations per second, while HSE 1.9 reaches just 595,213 peak operations per second.

Throughput v Avg. Read Latency

How to read this chart: HSE 2.0 and 1.9 data points for #Threads = 64 are shown with a dark outline. The HSE 2.0 data point is 317,780 operations per second farther to the right (higher throughput) and 65µs farther down (lower average read latency). Other point pairs for different #Threads can be similarly compared.

Figure 2b illustrates a striking contrast: HSE 2.0 showed greatly reduced read latency across all data points

HSE 1.9 shows acceptable average read latency at good throughput (reaching 433,179 operations per second with average read latency below 150µs). However, its average read latency reaches past 200µs in the next data point, then climbs to its worst value of 380µs.

Tagging Existing Assets	This workload is read-heavy with approximately 95% read storage IO and 5% written. Examples of this workload include adding metadata to existing data (tagging). Most of the tags are read while a few are written (or rewritten).

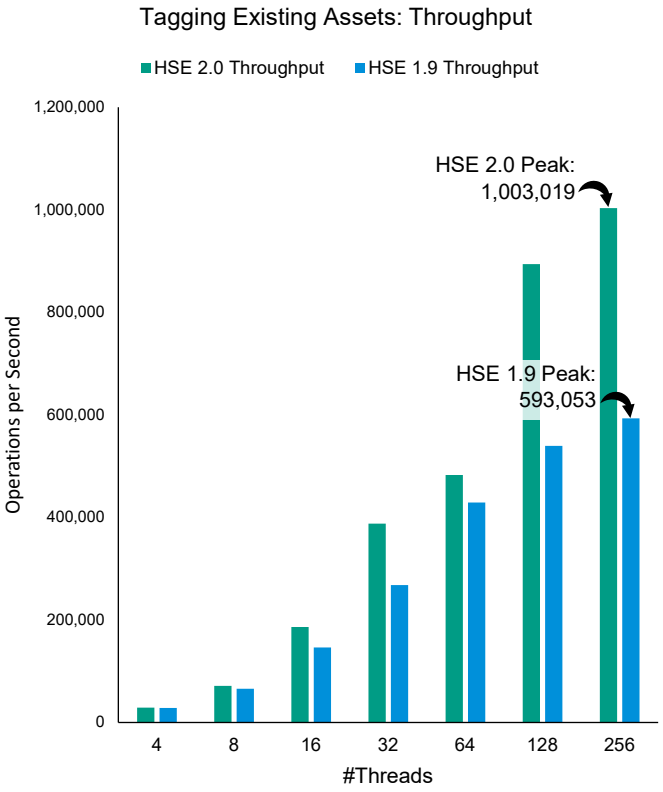

Figure 3a: Tagging Existing Assets: Throughput

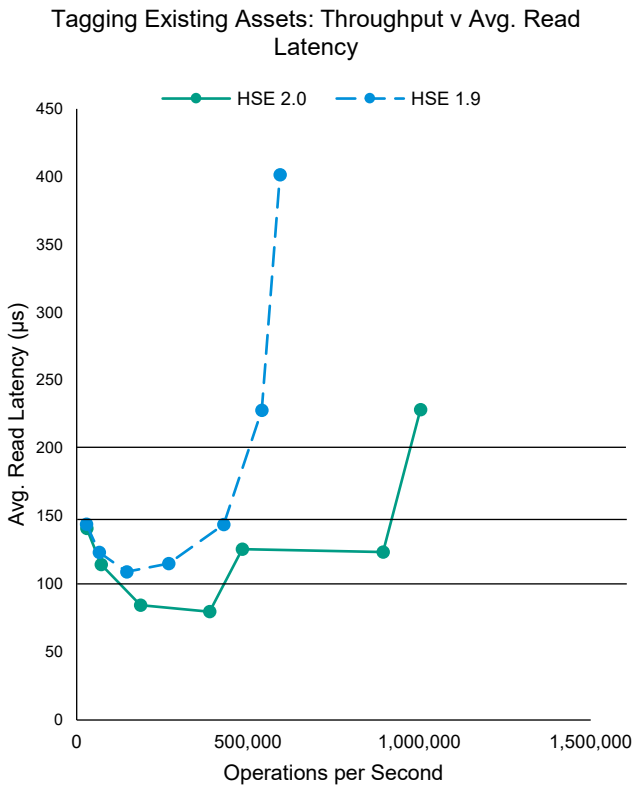

Figure 3b: Tagging Existing Assets: Avg. Read Latency

Tagging Existing Assets

Throughput

Figure 3a shows that for all tested #Threads, HSE 2.0 demonstrates superior performance compared to HSE 1.9.

HSE 1.9 reaches 593,053 peak operations per second, while HSE 2.0 reaches 1,003,019 peak operations per second, for a 1.7X improvement.

Throughput v Avg. Read Latency

Figure 3b shows database operations per second (x-axis) and avg. read latency (y-axis). For any point pair, farther down and to the right is better.

101µs to 150µs: Most of the HSE 2.0 avg. read latency values are in this range, reflecting higher throughput (farther right) at similar or lower avg. read latency than HSE 1.9 (farther down).

151µs to 200µs: HSE 2.0 and 1.9 each have few points in this region. While at similar height (µs), the HSE point in this region shows much higher performance (farther right).

Users Modifying Records	This is a 50% read/ 50% written workload. It reads a record, modifies it and then writes it back (read-modify-write). This workload models common database and user activity.

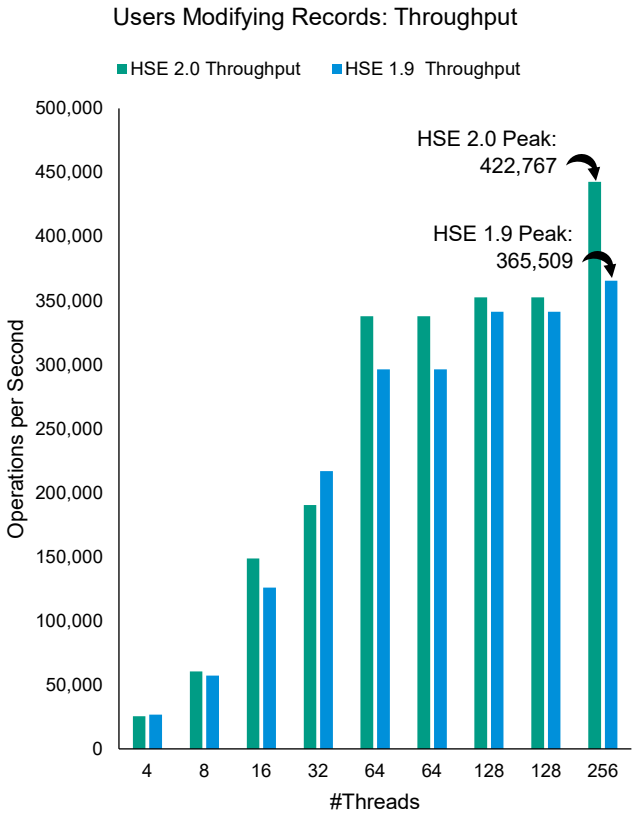

Figure 4a: Read-Modify-Write: Throughput

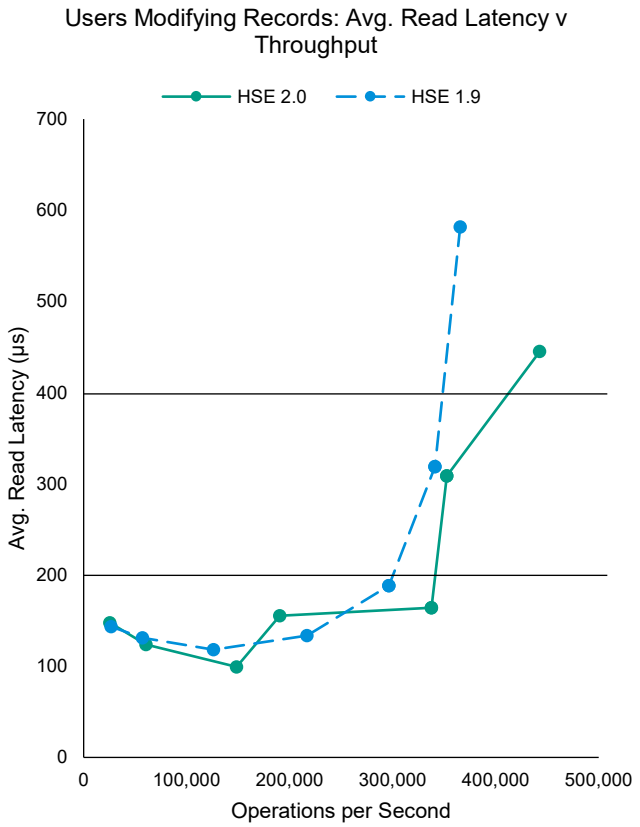

Figure 4b: Read-Modify-Write: Avg. Read Latency

Read/Modify/Write

Throughput

Figure 4a compares HSE 1.9 to 2.0 and shows similar performance for #Threads = 4, 8 and 128, while #Threads = 16, 64 and 256 show that HSE 2.0 offers better performance.

#Threads = 32 is the one anomalous value (HSE 1.9 offers a slight performance advantage). This #Threads value may be the ideal combination for HSE 1.9 under these exact conditions (although HSE 2.0 offers similar-to-better performance at all other #Threads, with a 1.2X advantage in peak performance).

Throughput v Avg. Read Latency

Figure 4b shows database operations per second (x-axis) and avg. read latency (y-axis). Looking at each avg. read latency range, we see

100µs to 200µs: HSE 1.9 reaches a maximum throughput of 296,264 operations per second while HSE 2.0 reaches 337,672 operations per second.

201µs to 400µs: In this average read latency range, HSE 1.9 and HSE 2.0 shows show similar results (341,212 and 352,455 operations per second).

>401µs: HSE 2.0 shows clearly better performance, reaching 442,767 operations per second at just 455µs average read latency while HSE 1.9 was slower at just 365,509 operations per second at a much higher average read latency of 582µs average read latency.

Recording User Sessions This is an update-heavy workload with about 50% of all storage IO being written. Examples of this workload can be seen when user sessions are recorded.

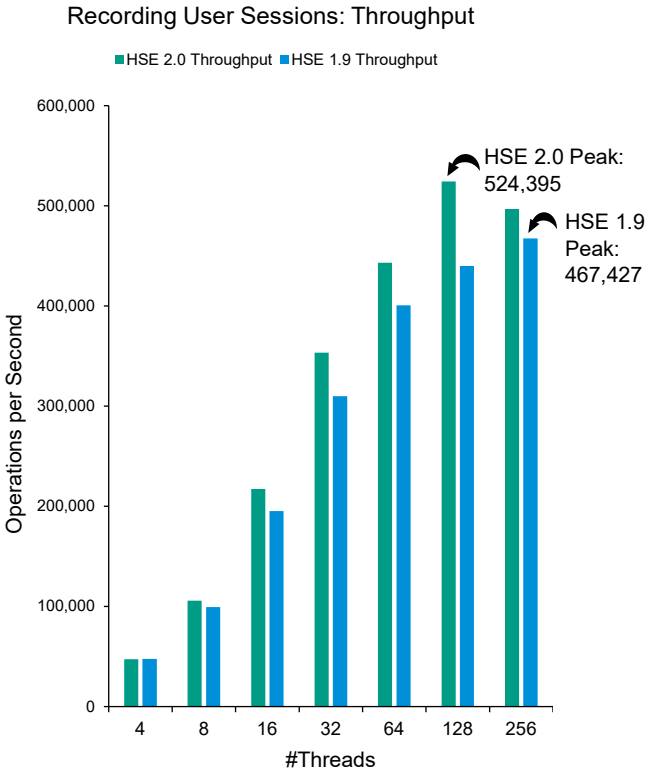

Figure 5a: Recording User Sessions: Throughput

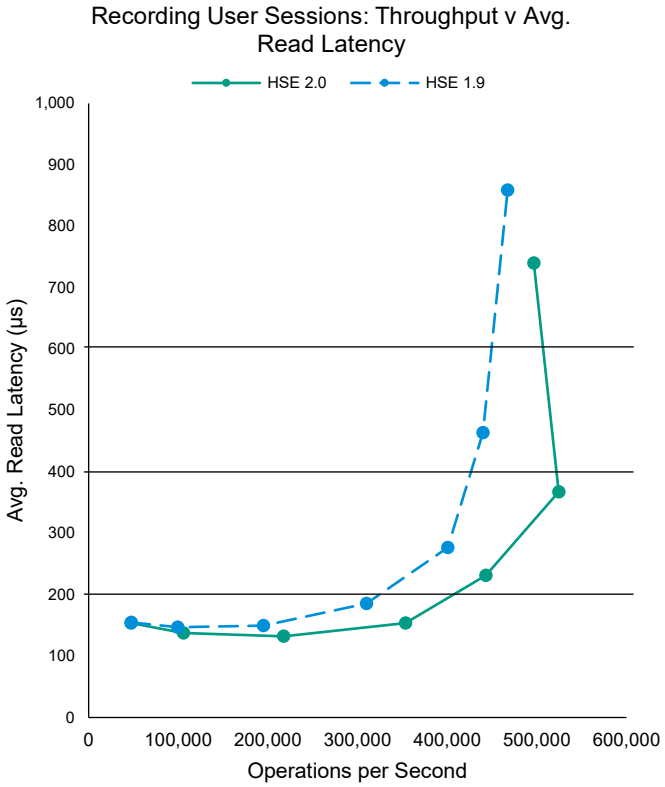

Figure 5b: Recording User Sessions: Avg. Read Latency

Recording User Sessions

Throughput Figure 5a shows that HSE 1.9 reaches 467,427 peak operations per second while HSE 2.0 reaches 524,395 peak operations per second, an improvement of 1.1X.

Throughput v Avg. Read Latency Figure 5b shows database operations per second (x-axis) and avg. read latency (y-axis). Looking at each avg. read latency range, we see:

<200µs: HSE 1.9 reaches a maximum throughput of 300,852 operations per second while HSE 2.0 reaches 353,486 operations per second.

201µs to 400µs: HSE 2.0 shows better performance at lower avg. read latency.

401µs to 600µs: HSE 2.0 shows better performance although its latency is just 317µs.

Conclusion

Micron HSE was designed to maximize the capabilities of new storage technologies by intelligently and seamlessly managing data placement. The result is significantly improved storage performance and decreased latency — even under the burden of massive-scale deployments.

With the growth in cloud comes rapid data growth, scalability and faster time to deployment of applications. HSE 2.0 reduces time spent deploying new instances for scale out by supporting more application integrations and aligning with best practices for containerized and cloud native apps. Applications utilizing HSE 2.0 optimizations, along with SSDs for semi-structured and non-structure data, benefit from higher and more predictable performance, lower latency and lower power consumption.

Get started developing at www.github.com/hse-project.

micron.com/hse

How We Tested

The Micron 7400 is optimized for mainstream NVMe SSD workloads, including SQL and NoSQL databases, block and object stores, VDI and server virtualization, and cloud storage. With its PCIe® Gen4 NVMe interface and vertically integrated architecture, the Micron 7400 SSD brings powerful performance to seven physical form factors with end-to-end validation.

It also offers a high degree of configurability with up to 128 namespaces, multiple-sector-size support and standards-based management with low power and reduced total cost of ownership.

For more details on the Micron 7400 SSD see www.micron.com/7400.

Hardware Configuration	
Server	Dell® PowerEdge R7525
CPU	AMD EPYC™ 7713 64-Core Processor
Memory	512GB Micron DDR4-3200
Server Storage	1x Micron 7400 Pro SSD with NVMe™ (3.84TB)
Boot Drive	Micron mainstream 960GB M.2 SSD with NVMe™
YCSB Version	YCSB 0.17.0 (HSE fork - https://github.com/hse-project/hse-ycsb)
HSE 2.0	YCSB 0.17.0.3.0-hse
HSE 1.9	YCSB 0.17.0
OS	CentOS Linux 8
Kernel	4.18.0-240.22.1.el8_3.x86_64

Table 1: HSE Server Configuration

System/Software Configuration

Micron Heterogeneous Memory Storage Engine: 2.0.0

HSE Profile:

```
# Copyright (C) 2019 Micron Technology.  All rights reserved.
# Workload: Native YCSB, workloads ABCDF, load phase

api_version: 1
kvs:
  mclass_policy: staging_max_capacity
  pfx_len: 7
```

Virtual Memory Tuning:

```
sudo sysctl -w vm.nr_hugepages=256
sudo sysctl -w vm.swappiness=1
sudo sysctl -w vm.dirty_background_ratio=5
sudo sysctl -w vm.dirty_ratio=15
```

The Yahoo Cloud Service Benchmark (YCSB) framework was originally designed to facilitate performance comparisons between various cloud data serving systems for transaction-processing workloads. This paper uses test workload similar to YCSB workloads.

Use Case	IO Type	Ratio
Recording User Sessions	Update heavy	50% Read, 50% Write
Tagging Existing Assets	Read mostly	95% Read, 5% Write
Caching User Profiles	Read only	100% Read, 0% Write
Users Modifying Records	Read-modify-write	50% Read, 50% read-modify-write

Table 2: Workload Overview

Rev. A 02/2022 CCM004-676576390-11595