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

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 apps⁵
- Designed to work with popular Linux 64-bit distributions⁶

More Integration Options

- Designed to run on common Linux file systems⁷
- 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](https://www.techtarget.com/searchstorage/definition/storage-class-memory)
² Additional operational details are available at [www.micron.com/hse](http://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](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](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
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).

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 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.

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.
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).

Tagging Existing Assets

<table>
<thead>
<tr>
<th>#Threads</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>400,000</td>
</tr>
<tr>
<td>8</td>
<td>800,000</td>
</tr>
<tr>
<td>16</td>
<td>1,600,000</td>
</tr>
<tr>
<td>32</td>
<td>3,200,000</td>
</tr>
<tr>
<td>64</td>
<td>6,400,000</td>
</tr>
<tr>
<td>128</td>
<td>12,800,000</td>
</tr>
<tr>
<td>256</td>
<td>25,600,000</td>
</tr>
</tbody>
</table>

HSE 2.0 Peak: 1,003,019
HSE 1.9 Peak: 593,053

Figure 3a: Tagging Existing Assets: Throughput

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

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

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).
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).

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.

#Read/Modify/Write

Throughput

Throughput v Avg. Read Latency

Figure 4a 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 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.

Recording User Sessions: Throughput

Figure 5a: 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.

Recording User Sessions: Throughput v Avg. Read Latency

Figure 5b: Recording User Sessions: 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](http://www.micron.com/7400).

<table>
<thead>
<tr>
<th>Hardware Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Server Storage</td>
</tr>
<tr>
<td>Boot Drive</td>
</tr>
<tr>
<td>YCSB Version</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>OS</td>
</tr>
<tr>
<td>Kernel</td>
</tr>
</tbody>
</table>

Table 1: HSE Server Configuration

<table>
<thead>
<tr>
<th>System/Software Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micron Heterogeneous Memory Storage Engine</td>
</tr>
<tr>
<td>HSE Profile</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Virtual Memory Tuning

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.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>IO Type</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recording User Sessions</td>
<td>Update heavy</td>
<td>50% Read, 50% Write</td>
</tr>
<tr>
<td>Tagging Existing Assets</td>
<td>Read mostly</td>
<td>95% Read, 5% Write</td>
</tr>
<tr>
<td>Caching User Profiles</td>
<td>Read only</td>
<td>100% Read, 0% Write</td>
</tr>
<tr>
<td>Users Modifying Records</td>
<td>Read-modify-write</td>
<td>50% Read, 50% read-modify-write</td>
</tr>
</tbody>
</table>

Table 2: Workload Overview

©2022 Micron Technology, Inc. All rights reserved. All information herein is provided on an "AS IS" basis without warranties of any kind. Products are warranted only to meet Micron’s production data sheet specifications. Products, programs and specifications are subject to change without notice. Micron Technology, Inc. is not responsible for omissions or errors in typography or photography. Micron, the Micron logo and all other Micron trademarks are the property of Micron Technology, Inc. All other trademarks are the property of their respective owners.

Rev. A 02/2022 CCM004-676576390-11595