PRISM: Optimizing Key-Value Store for Modern Heterogeneous Storage Devices

Yongju Song, Wook-Hee Kim, Sumit Kumar Monga,
Changwoo Min, and Young Ik Eom
Heterogeneous Storage Systems

A method for assigning different categories of data to *various types of storage media* to *reduce overall storage costs.*
Heterogeneous Storage Systems

A method for assigning different categories of data to various types of storage media to reduce overall storage costs.
Heterogeneous Storage Systems

A method for assigning different categories of data to *various types of storage media* to *reduce overall storage costs*.
Heterogeneous Storage Systems

A method for assigning different categories of data to *various types of storage media* to *reduce overall storage costs*.
## Evolution of Storage Heterogeneity

<table>
<thead>
<tr>
<th>Specification</th>
<th>Model</th>
<th>Capacity</th>
<th>Cost $/TB</th>
<th>Read Latency (usec)</th>
<th>Write Latency (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAM</td>
<td>SK Hynix DRAM w/DDR4</td>
<td>16</td>
<td>5,427</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>NVM</td>
<td>Intel Optane DCPMM w/DDR-T</td>
<td>128</td>
<td>4,096</td>
<td>0.30</td>
<td>0.09</td>
</tr>
<tr>
<td>NVM SSD</td>
<td>Intel Optane 905P w/PCIe 3</td>
<td>960</td>
<td>1,024</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Flash SSD</td>
<td>Samsung 980 Pro w/PCIe 4</td>
<td>1024</td>
<td>150</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>Flash SSD</td>
<td>Samsung 980 w/PCIe 3</td>
<td>1024</td>
<td>100</td>
<td>60</td>
<td>20</td>
</tr>
</tbody>
</table>
## Evolution of Storage Heterogeneity

<table>
<thead>
<tr>
<th>Type</th>
<th>Model</th>
<th>Capacity</th>
<th>Cost ($/TB)</th>
<th>Read Latency (usec)</th>
<th>Write Latency (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM</td>
<td>SK Hynix DRAM w/DDR4</td>
<td>16</td>
<td>5,427</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>NVM</td>
<td>Intel Optane DCPMM w/DDR-T</td>
<td>128</td>
<td>4,096</td>
<td>0.30</td>
<td>0.09</td>
</tr>
<tr>
<td>NVM SSD</td>
<td>Intel Optane 905P w/PCIe 3</td>
<td>960</td>
<td>1,024</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Flash SSD</td>
<td>Samsung 980 Pro w/PCIe 4</td>
<td>1024</td>
<td>150</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>Flash SSD</td>
<td>Samsung 980 w/PCIe 3</td>
<td>1024</td>
<td>100</td>
<td>60</td>
<td>20</td>
</tr>
</tbody>
</table>
# Evolution of Storage Heterogeneity

<table>
<thead>
<tr>
<th>Type</th>
<th>Model</th>
<th>Capacity</th>
<th>Cost</th>
<th>GB/$/TB</th>
<th>Read Latency (usec)</th>
<th>Write Latency (usec)</th>
<th>Read BW (GB/s)</th>
<th>Write BW (GB/s)</th>
<th>Warranty (PBW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM</td>
<td>SK Hynix DRAM w/DDR4</td>
<td>16</td>
<td>5,427</td>
<td></td>
<td>0.08</td>
<td>0.08</td>
<td>15</td>
<td>15</td>
<td>∞</td>
</tr>
<tr>
<td>NVM</td>
<td>Intel Optane DCPMM w/DDR-T</td>
<td>128</td>
<td>4,096</td>
<td></td>
<td>0.30</td>
<td>0.09</td>
<td>6.8</td>
<td>1.9</td>
<td>292</td>
</tr>
<tr>
<td>NVM SSD</td>
<td>Intel Optane 905P w/PCIe 3</td>
<td>960</td>
<td>1,024</td>
<td></td>
<td>10</td>
<td>10</td>
<td>2.6</td>
<td>2.2</td>
<td>17.52</td>
</tr>
<tr>
<td>Flash SSD</td>
<td>Samsung 980 Pro w/PCIe 4</td>
<td>1024</td>
<td>150</td>
<td></td>
<td>50</td>
<td>20</td>
<td>7</td>
<td>5</td>
<td>0.6</td>
</tr>
<tr>
<td>Flash SSD</td>
<td>Samsung 980 w/PCIe 3</td>
<td>1024</td>
<td>100</td>
<td></td>
<td>60</td>
<td>20</td>
<td>3.5</td>
<td>3</td>
<td>0.6</td>
</tr>
</tbody>
</table>
There is **No Clear Separation** between performance-/capacity-devices.

“The Storage Hierarchy is **Becoming a Jungle.**” [CIDR’21, Dong Xie]

“The Storage Hierarchy is **Not a Hierarchy.**” [FAST’21, Remzi H. Arpaci-Dusseau]
Today’s Storage Hierarchy
Today’s Storage Hierarchy

Placing hot data on NVM

- System can leverage the low latency of NVM but suffer from its limited bandwidth.
Today’s Storage Hierarchy

Placing hot data on NVM

- System can leverage the low latency of NVM but suffer from its limited bandwidth.

Traversing data layer by layer for handling read requests

- Inefficient traversal leads to wasting CPU cycles.
- Overall performance may be bounded to the device with the lowest performance.
Today’s Storage Hierarchy

Placing hot data on NVM
- System can leverage the low latency of NVM but suffer from its limited bandwidth.

Traversing data layer by layer for handling read requests
- Inefficient traversal leads to wasting CPU cycles.
- Overall performance may be bounded to the device with the lowest performance.

How should we design a **Heterogeneous Storage System** in the **Modern Storage Landscape**?
Design Goals of PRISM

**Drawing the full potential of heterogeneous storage devices.**

**Minimizing the overhead of software stack for scalability**

**Providing a high level of crash consistency & concurrency**
Overview of PRISM

Persistent Key Index

Heterogeneous Storage Index Table (HSIT)

Persistent Write Buffer (PWB)

Value Storage

Scan-aware Value Cache (SVC)
Overview of PRISM: Insert (k1, v1)

1. Persistent Write Buffer (PWB)
2. Heterogeneous Storage Index Table (HSIT)
3. Persistent Key Index
4. Value Storage

Diagram:
- **DRAM**
- **NVM**
- **SSD**
Overview of PRISM: Insert (k1, v1)

Background Reclamation of PWB
- Preventing application threads from blocking

Asynchronous I/O batching
- Achieving high bandwidth of SSD

Asynchronous Bandwidth-Optimized WRITE
Asynchronous Bandwidth-Optimized WRITE

Persistent Write Buffer (PWB)

Watermark (50%)

Application Thread (from Index to PWB)

Per-value Metadata

Background Reclamation (from PWB to Value Storage)

Value Storage

Free Chunk N (512KB)

Backward ptr

Value size

Append-only writes
Asynchronous Bandwidth-Optimized WRITE

Persistent Write Buffer (PWB)

Watermark (50%)

Application Thread (from Index to PWB)

Background Reclamation (from PWB to Value Storage)

Value Storage

Free Chunk N (512KB)

Append-only writes
Design Overview of \textit{PRISM}: Lookup(k4)

- Persistent Key Index
- Heterogeneous Storage Index Table (HSIT)
- Persistent Write Buffer (PWB)
- Value Storage
- Scan-aware Value Cache (SVC)
Design Overview of PRISM: Lookup(k4)

Adjust the IO batch size for SSD reads according to thread concurrency

Combine reads from multiple threads to a single read operation

- Aggressively utilizing the bandwidth and hide latency of SSD
Opportunistic Thread Combining for READ

Thread Combining Queue (TCQ)

Value Storage (io_uring)
Opportunistic Thread Combining for READ

Thread Combining Queue (TCQ)

Thread 1
- READ 'v1'

Thread 2
- READ 'v2'

Thread N
- READ 'v4'

Leader

Atomic swap

TCQ tail

Coalesce and submit requests

I/O Completion thread

Value Storage (io_uring)

R: 'v1'

R: 'v2'

Submission Queue (SQ)

Notify request completion

Completion Queue (CQ)
Cross-media Crash Consistency

Persistent Key Index

Heterogeneous Storage Index Table (HSIT)

Persistent Write Buffer (PWB)

Value Storage

Scan-aware Value Cache (SVC)

NVM
SSD
DRAM
Cross-media Crash Consistency

Components are scattered across multiple heterogeneous devices
• Lightweight crash consistency with Forward & Backward pointers

Heterogeneous Storage Index Table (HSIT)

Cross-media Crash Consistency
Crash Consistent Update of Values with HSIT

```
INSERT {k2, v2}
```

- Leaf node of Persistent Key Index
- HSIT entry
- Forward pointer
- Backward pointer
- Obsolete value
- Updated value

Diagram:
- kN
- INSERT {k2, v2}
- PWB ptr.
- VS ptr.
- SVC ptr.
- PWB #0
- PWB #1
- v2
- v2

1. Forward pointer
2. Backward pointer
Crash Consistent Update of Values with HSIT

INSERT \{k2, v2\}

Leaf node of Persistent Key Index

Forward pointer

Backward pointer

Obsolete value

Updated value
Crash Consistent Update of Values with HSIT

UPDATE \{k2, v2\} to \{k2, v2'\}

UPDATE \{k3, v3\} to \{k3, v3'\}

Leaf node of Persistent Key Index

Forward pointer

Backward pointer

Obsolete value

Updated value
Experimental Setup

Hardware environment

• Two-socket Intel Xeon machine
• Each socket: 20 CPU cores, Six 128GB Intel Optane DIMMSs, and 96GB DRAM
• Eight Samsung 980 PRO 1TB SSDs with two NVMe RAID Controllers HighPoint SSD7103

Competitors

• KVell: DRAM-SSD with up to 64 batched I/Os [SOSP’21]
• MatrixKV: DRAM-NVM-SSD [ATC’20]
• Allocated their hardware resources at the same cost levels
Performance Comparison on YCSB

WRITE: Does not require level-compaction & Per-thread write buffer
READ: No need for traversing multiple levels & Efficient KV item caching
Opportunistic Thread Combining

Prism opportunistically adjust the IO batch size for read operations according to thread concurrency.
In the paper...

Performance under other workloads

Performance impact of..

- Number of SSDs
- Size of PWB/SVC
- Write amplification
- Garbage collection in Value Storage
- Individual techniques

Size of NVM space

Recovery
Conclusion

We answered the question:

*How should we design a Heterogeneous Storage System in the Modern Storage Landscape?*

- Synergistic Five Components
- Asynchronous Bandwidth-Optimized WRITE
- Opportunistic Thread Combining
- Cross-media Crash Consistency & Concurrency Control using Forward & Backward Pointers
**PRISM**: Optimizing Key-Value Store for Modern Heterogeneous Storage Devices

Yongju Song, Wook-Hee Kim, Sumit Kumar Monga, Changwoo Min, and Young Ik Eom

Paper