Accelerating I/O-Intensive Applications Through Multi-Tiered Buffering with Hermes

Luke Logan, Xian-He Sun, Anthony Kougkas
lllogan@hawk.iit.edu, {sun, akougkas}@iit.edu

1. Multi-Tiered Storage
* Many HPC sites have non-volatile burst buffers between memory and disk
* Deep Memory and Storage Hierarchy (DMSH)

2. Current Situation
* A need for automatic and intelligent data placement

3. Overview

<table>
<thead>
<tr>
<th>Applications</th>
<th>Hermes Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Placement Engine</th>
<th>Prefetcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata Manager</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buffer Organizer</th>
<th>Buffer Organizer</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RAM</th>
<th>NVMe</th>
<th>I/O Clients</th>
<th>Burst Buffers</th>
<th>File System</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DMSH Hardware</th>
<th>SSD (i.e., Burst Buffers)</th>
<th>HDD (i.e., Parallel File System)</th>
</tr>
</thead>
</table>

4. Hermes APIs

* Hermex exposes a Put / Get API to store data "blobs"
* Various adapters transparently convert I/O into blobs
* Supports HDF5, POSIX, STDIO, MPI-I0
* No application changes

5. Data Placement Engine
* Decide where to initially place data
* Can be used to improve write performance
* Three policies currently implemented
* Custom policies can be built using buffer schema

6. Buffer Organizer
Adjusts the position of blobs in the hierarchy asynchronously based on the blob's score

```
Compute  I/O  Compute
Burst     Buffer
```

An example of flushing blobs during compute

* Promote
* Demote

7. Buffer Organizer Blob Scoring

<table>
<thead>
<tr>
<th>Access Frequency</th>
<th>Access Recency</th>
<th>Blob Size</th>
<th>Blob Tags</th>
</tr>
</thead>
</table>

8. Prefetcher
Changes the scores of blobs depending on their expected next access

```
Node
```

* Many workloads are predictable in their I/O patterns (e.g., deep learning randomness seeds)
* Prefetcher thread is periodically called to update blob scores
* Hermes I/O events are stored in a log, which the prefetchers can analyze

8. Metadata Manager
* Adapter-specific information (e.g., what files should Hermes flush data to before exiting?)
* Internal metadata (e.g., map blobs to hardware locations)

9. Data Staging

* Should transparently manage data placement for each independent tier

10. Testbed

<table>
<thead>
<tr>
<th>CPU</th>
<th>Nodes</th>
<th>Network</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2GHz Xeon Scalable Silver 4114</td>
<td>16 nodes</td>
<td>40GbE Ethernet with RoCE support</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>48 cores per node</th>
<th>16 nodes</th>
<th>400GB Ethernet with RoCE support</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RAM</th>
<th>NVMe</th>
<th>I/O Clients</th>
</tr>
</thead>
</table>

11. Scientific Simulation Workflow

* VPIC: particle-in-cell simulation code for modeling 3D kinetic plasmas (write-only)
* BD-CATS: particle clustering algorithm (read-only)

12. Conclusion

* Currently working with application domain scientists to evaluate Hermes for more workloads
* Large-scale evaluations
* Identify opportunities for workload-specific optimization

13. Ongoing

* Designed / implemented Hermes, an intelligent and transparent I/O buffering system
* Demonstrated the importance of intelligent buffering on scientific workflows

Setup

* Adding RAM + NVMe 30-50x faster than using only HDD
* Data effectively buffered
* Can utilize burst buffers to optimize data-intensive workflow stages

Analysis

* VPIC writes data
* Data kept in Hermes
* BD-CATS then runs clustering
* 128GB of data per checkpoint
* 8 checkpoints
* 16 nodes, 768 processes
* MaxBandwidth DPE