UniHeap : Managing Persistent Objects Across Managed Runtimes for Non-Volatile Memory

Daixuan Li    Benjamin Reidys    Jinghan Sun
Thomas Shull  Josep Torrellas  Jian Huang
Non-Volatile Memory: Opportunities & Challenges

Performance & Persistency

Byte-Addressable

Data Durability

Memory Persistency

Programmability
Programmability Challenge of NVM

- Object
- Volatile Cache
- Extra Programming Effort
  - clwb & sfence instructions
- Persist-Object
- NVM Device
- Performance Bugs
- Correctness
Leveraging Managed Runtime to Manage NVM

Popular Programming Models

Hardware Complexity

Managed Data Objects
AutoPersist: An Easy-to-Use NVM Framework

Programmability Improvement

Cross Managed Runtime?
Managing Persistent Object Across Runtime is Desirable

Web Service

Shared Libraries

Data Analytics

A Unified Interface to Access Persistent Object is Needed.
State-of-the-Art Object Sharing Approaches

• Thrift/Protobuf:
  - Unified Interface Definition
  - Compiler
  - Java
  - Python
  - RPC Protocol
  - Serialization Overhead

• Shared Memory:
  - Java
  - Python
  - Shared Memory
  - Does not support NVM
UniHeap: Managing Persistent Objects Across Runtimes

- Java
- Python
- JavaScript

Unified Persistence Layer

Shared NVM Heap
Challenges of Persistent Object Management Across Runtimes

- Unified Object Model
- Persistent and Crash-Safe Implementation
- Efficient and Correct GC
Unified Object Model and Type System

<table>
<thead>
<tr>
<th>Uniheap</th>
<th>char</th>
<th>short</th>
<th>int</th>
<th>long</th>
<th>float</th>
<th>double</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>boolean, byte</td>
<td>char</td>
<td>int</td>
<td>long</td>
<td>float</td>
<td>double</td>
<td>reference, array</td>
</tr>
<tr>
<td>Python</td>
<td>-</td>
<td>-</td>
<td>int</td>
<td>long</td>
<td>float</td>
<td>-</td>
<td>list, dict, tuple</td>
</tr>
<tr>
<td>JavaScript</td>
<td>boolean</td>
<td>-</td>
<td>num</td>
<td>num</td>
<td>num</td>
<td>num</td>
<td>array</td>
</tr>
</tbody>
</table>

- Two kinds of built-in types: **numeral type** and **reference type**
Managing Persistent Objects in A Log-Structured Manner

Reduce Logging Overhead with Out-of-Place Update
Enforce Memory Persistency with Simple API

Atomic Region

atomic_begin

Persist Objects

atomic_end

Durable Root

set_root

get_root

UniHeap

Crash Consistency

Failure Atomic
Coordinated GC Across Managed Runtimes

- Marking phase
- Relocation phase
- Compaction phase
- Clean-up phase
Put It All Together

Java Program

Python Program

......

JavaScript Program

Java Module

Python Module

JavaScript Module

UniHeap Shared Library

Runtime

Unified Interface

Direct Memory Access

Shared NVM Heap
Experiment Setup

- CPU: 24-core Intel 2nd Xeon
- NVM: 128GB Intel Optane

Evaluation Benchmarks

- Java: YCSB
- Python: N Queens, N body
- JavaScript: Matrix multiplication
Performance of Persistent Object Sharing

UniHeap outperforms existing approach by 1.2x - 3.4x
UniHeap outperforms existing approach by 1.2x - 3.4x
Scalability of UniHeap

UniHeap can scale to support multiple managed runtimes.
UniHeap Summary

Unified Persistence Layer

Shared NVM Heap
<table>
<thead>
<tr>
<th>Daixuan Li</th>
<th>Benjamin Reidys</th>
<th>Jinghan Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thomas Shull</td>
<td>Josep Torrellas</td>
<td>Jian Huang</td>
</tr>
</tbody>
</table>