CXLMemSim: A pure software simulated CXL.mem for performance characterization

Yiwei Yang
Background

- Increase in SoC core counts requires continued increase in memory bandwidth and capacity, but the gap between such requirements and platform provisioning capability is growing.
  - Guided Testcases: Meta's Metaverse

Source: SK Hynix | Adding New Value to Memory Subsystems through CXL
**Background**

- **CXL**(Compute eXpress Link) is a Serial Low latency coherency protocol based on PCIe5.0+.
  - CPU vendors like Intel empower the PCIe attached devices coherency protocol
    - Persistent memory is wasting Memory Channel Resources
  - CXL.io is pure PCIe, CXL.cache supports cacheline access from device, CXL.mem is a memory window that is cacheable in LLC.
    - Type 1 **Accelerator** support CXL.io and CXL.cache, Type 2 **GPU/FPGA** support all three protocol, Type 3 **Memory Expander** support CXL.io and CXL.mem
  - CXL.mem Support 150-260ns access and constant bandwidth. CXL2.0 & 3.0 is extending the topology accross datacenter with low scarification.
Motivation

• Full System simulator like Gem5 is too slow if we don’t require all microarch metrics.

• Can provide a variety of memory latency and bandwidth choice and diverse memory expander, pool topology and different types of data request.
Design Diagram

- Epoch Based Sending SIGSTOP to userspace program and observe the PMU/PEBS/eBPF result from and stored access history in topology map.
- Calculate and append latency penalty to program and send SIGCOUT to the program

```
Simulator runs on core 15

Insert/delete to CXL Topology

Calculate Latency/Bandwidth/Congestion

Wait Epoch

Alderlake

Core LLC Core LLC Core LLC Core LLC ...

Process runs on core 0-14 with HT off
InCore: MemRead, L2 stall, LLC Miss
eBPF: munmap, sbrk shrink param
UnCore: LLC Writeback
PEBS: physical address LLCmiss & timestamp
```

SIGSTOP

SIGCONT
Design Diagram

- One Epoch DRAM Access like below
- Clearly see how LLC miss, L2 stalls and bandwidth throttles affect latency.

![Design Diagram](image-url)
Design Diagram

• User input Memory Topology, Bandwidth, Capacity.
  • Construct the Topology Map, each with access history map capped by capacity.
  • Default topology map insertion policy based on NUMA Policy
    • Use LLC misses PEBS event to insert to the endpoint’s access history map.
    • Use linear regression of DRAM latency and L2 stalls+LLC misses PMU metrics to targeted latency.
    • Based on timestamp to calculate migration timing and Switch STT time.
  • Observe the program deallocate memory by eBPF
    • Find & Delete the entry in the endpoint
Performance Evaluation

• i9-12900K@5.0GHz processor with 96 GB of DDR5 4800MHz memory
• Average 4.41x speed up for real world application