CXLMemSim A pure software simulated CXL.mem for performance characterization

Yiwei Yang, Pooneh Safayenikoo, Jiacheng Ma*, Tanvir Ahmed Khan*, Andrew Quinn

Yarch' 23

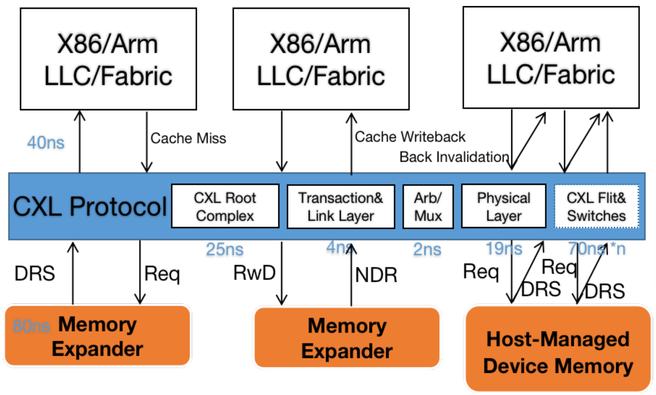
UC SANTA CRUZ

UNIVERSITY OF MICHIGAN *

Problem

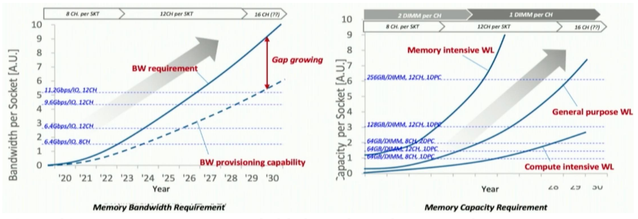
Goal: Provide a cheap and more available to do CXL.mem performance caracherization.

- Trace based Full System simulator like Gem5[3] is too slow, Using NUMA for simulation lacks bandwidth and topology details[7]. Sometimes we only care about the memory latency and bandwidth for performance characterization without knowing the arch details.
- We provide a wide range of memory latency and bandwidth choice and different memory expander and pool topology compared to NUMA simulator. We can support the CXL.mem 2.0 type3 semantic from perspective of CPU, we will future support all the CXL. mem device semantic of back invalidation.



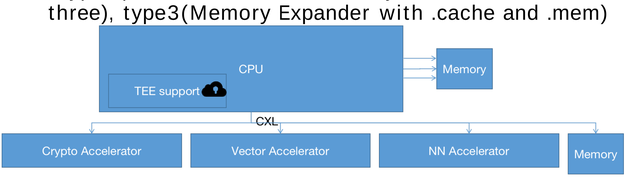
Background

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



CXL (Compute Express Link) is a low latency interconnect technology designed to enable communication between various computing components, including processors, memory, and accelerators.

- CPU venders like Intel empower the PCIe attached devices coherency protocol.
  - Persistent memory wastes memory channels
  - Make Hardware Software Co-Design prosperous
- CXL consists of three protocols, CXL.io, CXL.cache, and **CXL.mem** and have type1(Acclerator with .io and .cache) , type2(Accelerators with Memory like GPU with all three), type3(Memory Expander with .cache and .mem)

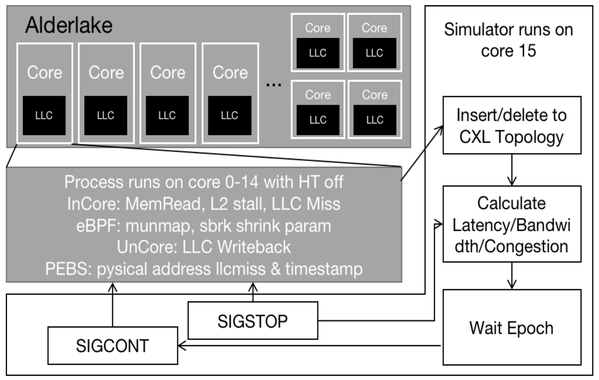


- CXL 2.0 supports Multi-Level Devices by CXL Switches. CXL 3.0 supports memory pooling across data center.

Our approach

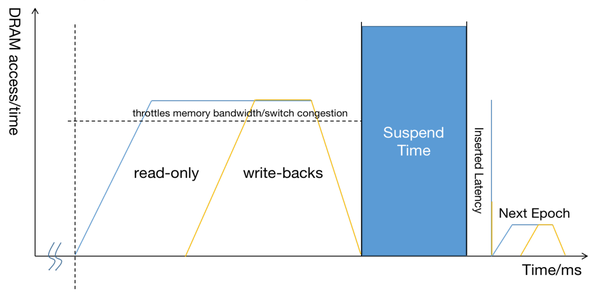
Design Diagram

- Epoch Based Sending SIGSTOP to userspace prgram and observe the PMU/PEBS[1,8]/eBPF[5] result from and stored access history in topology map.
  - Every 5ms stop, the calculation of Linear Regression bandwidth and latency panelty based on the PMU we observed takes 100ms and we back insert the latency and resume.
- Append latency panelty to program and send SIGCOUT to the program



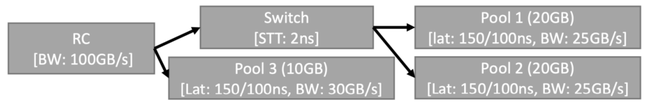
Epoch Graph

- CXLMemSim divides the execution of the attached user program into epochs and sets up an epoch timer that periodically interrupts the attached program. When the program is interrupted, CXLMemSim obtains memory access information from the memory event trace and uses the allocation trace to determine the corresponding memory pool of each memory access.



- CXLMemSim divides the execution of the attached user program into epochs and sets up an epoch timer that periodically interrupts the attached program. When the program is interrupted, CXLMemSim obtains memory access information from the memory event trace and uses the allocation trace to determine the corresponding memory pool of each memory access.
- While the program is paused, CXLMemSim uses the memory trace to calculate three types of timing delaysthat should be added to the execution time of each epoch: 1) latency delay, 2) congestion delay, and 3) bandwidth delay. CXLMemSim calculates the latency delay by multiplying the number of memory operations to each memory pool by the difference between the latency of the target memory pool and the latency of local DRAM. Then, CXLMemSim calculates the congestion delay by iteratingover the memory trace to find events that use the same switchwithin a smaller interval than the switchs serial transmission time(STT); once such events are found, CXLMemSim injects the necessary delays. Finally, CXLMemSim determines the bandwidth delays.

Input Topology



- 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
- Observe the program deallocate memory by eBPF
  - Find & Delete the entry in the endpoint

PMU and PEBS events we collected

Performance events of CPU counters	
$L2_{stalls}$	CYCLE_ACTIVITY:STALLS_L2_PENDING
LLC_{hit}	MEM_LOAD_UOPS_L3_HIT_RETIRED: XSNP_NONE
LLC_{miss}, LLC_{miss,cpu_i}	MEM_LOAD_UOPS_L3_MISS_RETIRED: LOCAL_MEM
LLC_{miss,PF_i} + LLC_{miss,cpu_i}	OFFCORE_RESPONSE_0 (offcore rsp: 0x3FB84003F7)
Performance events of CBo (LLC controller) counters	
WB	LLC_VICTIMS.M_STATE

Algorithm

- We uses performance linear regression model from MES[6], other models can be found in Quartz[9] and LEEP[10]. The LLC misses are divided into Writeback and Readonly.

Algorithm 1 CXLDelayedAlgorithm(epoch)

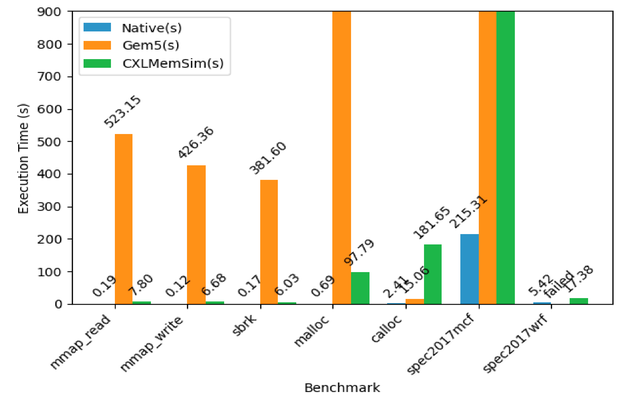
```
LLCMissEvent=PEBS()
PMUResult=PMU()
DeallocateEvent=eBPF()
Controller=InterLeavingPolicy(LLCMissEvent)
for EndPoint in Controller do
    if EndPoint is Expander then
        EndPoint.CXL_writeback=(α*PMUResult.LLC_WB_misses)/(PMUResult.LLC_miss*α+PMUResult.LLC_hit) *PMUResult.L2_stalls
        EndPoint.CXL_writeback_delay=CXL_writeback*(EndPoint.CXL_write_latency-DRAMLatency)
        EndPoint.CXL_readonly=(α*(PMUResult.LLC_misses-PMUResult.LLC_WB_misses))/(PMUResult.LLC_miss*α+PMUResult.LLC_hit) *PMUResult.L2_stalls
        EndPoint.CXL_readonly_delay=CXL_readonly*(EndPoint.CXL_write_latency-DRAMLatency)
        if EndPoint.CXL_PEBS_miss_times*Granularity/epoch <EndPoint.CXL_bandwidth then
            EndPoint.CXL_bandwidth_latency=(EndPoint.CXL_PEBS_miss_times*Granularity)/(EndPoint.CXL_bandwidth) -epoch
        end if
    else
        for all Endpoint1 in Endpoint do
            for all Endpoint2 in Endpoint do
                if Endpoint1.LLC_miss_timestamp-Endpoint2.LLC_Miss_Timestamp<2ns then
                    Endpoint.CXL_switch_delay+=Endpoint.STT_time
                end if
            end for
        end for
    end if
    Controller.CXL_delay+=EndPoint.CXL_writeback_delay+EndPoint.CXL_readonly_delay+
        EndPoint.CXL_bandwidth_latency+EndPoint.CXL_switch_delay
end for
```

Usage

- Online Migration in Page Level/ Cacheline Level
  - For page, we can say we just need to update the page table to the memory address of somewhere in the memory expander.
    - like numa mode CXL memory expander
  - For cacheline, we can say we make another mapping under physical address that
    - ASPLOS'22 Software-defined Address Mapping: a Case on 3D Memory
    - CPU don ' t know the real data ' s address ' location, so need to first ask local then CXL Controller
    - take 2 magic bit from ECC and mark valid and location like memory mode PM.
    - Compared with RDMA based ideas, we can leverage the cacheline state. so-called shared memory semantic.
- Side Channel Attack characterization
  - You can get clear traces and set up side channel models over the simulator.

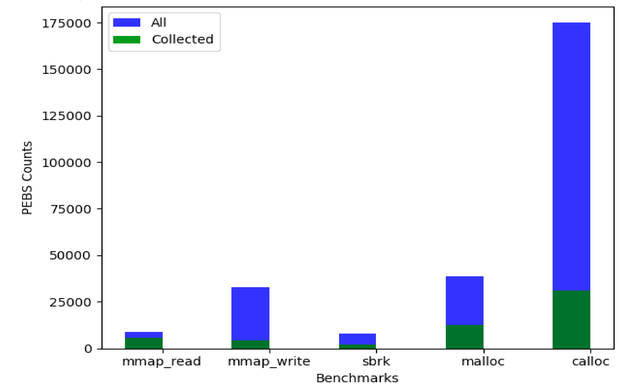
Preliminary Results

Performance Evaluation



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

Acuracy Evaluation



- We think our latency calculation linear regression design of the simulator are dependent to the percentage of sample based buffer of PEBS inside kernel.

References

- **Project page**: https://github.com/SlugLab/SlugAllocator/

[1] D. Bakhvalov. **Advanced profiling topics. pebs and lbr**, 2018.
[2] N. Binkert, B. Beckmann, et al. The **gem5** simulator. ACM SIGARCH computer architecture news, 39(2):1 – 7, 2011.
[3] fadedzipper. **Gem5 cxl version**, 2022. URL https://github.com/fadedzipper/gem5-cxl/compare/stable...cxl.mem-dev.
[4] D. Gouk, S. Lee, M. Kwon, and M. Jung. Direct access,{High-Performance} memory disaggregation with {**DirectCXL**}. In 2022 USENIX Annual Technical Conference (USENIX ATC 22), pages 287 – 294, 2022.
[5] B. Gregg. **BPF Performance Tools**. Addison-Wesley Professional, 2019.
[6] A. Koshiba, T. Hirofuchi, R. Takano, and M. Namiki. **MES**: A software-based nvm emulator supporting read/write asymmetric latencies. IEICE TRANSACTIONS on Information and Systems, 102(12):2377 – 2388, 2019.
[7] H. Li, D. S. Berger, L. Hsu, D. Ernst, P. Zardoshti, S. Novakovic, M. Shah, S. Rajadnya, S. Lee, I. Agarwal, M. D. Hill, M. Fontoura, and R. Bianchini. **Pond**: Cxl-based memory pooling systems for cloud platforms. 2023.
[8] A. Raybuck, T. Stamler, W. Zhang, M. Erez, and S. Peter. **Hemem**: Scalable tiered memory management for big data applications and real nvm. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, SOSP 21.
[9] H. Volos, G. Magalhaes, L. Cherkasova, and J. Li. **Quartz**: A lightweight performance emulator for persistent memory software. In Proceedings of the 16th Annual Middleware Conference, pages 37 – 49, 2015.
[10] G. Zhu, K. Lu, X. Wang, X. Zhou, and Z. Shi. **LEEP:** Building emulation framework for non-volatile memory. IEEE Access, 5:21574 – 21584, 2017.
[11] Neugebauer, R., Antichi, G., Zazo, J. F., Audzevich, Y., López-Buedo, S. , & Moore, A. W. (2018, August). Understanding PCIe performance for end host networking. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (pp. 327-341).