HCDS 2025

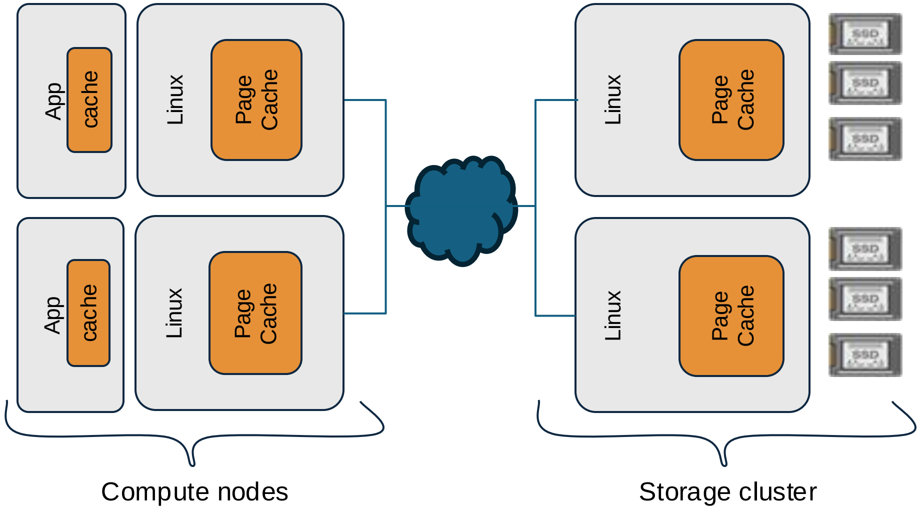
Storage cluster for persistency, CXL pools for caching

Karim Manaouil*, Ji Zhang, Yang Zhe, Zhou Xing Wang,
Shai Bergman, Antonio Barbalace*

*

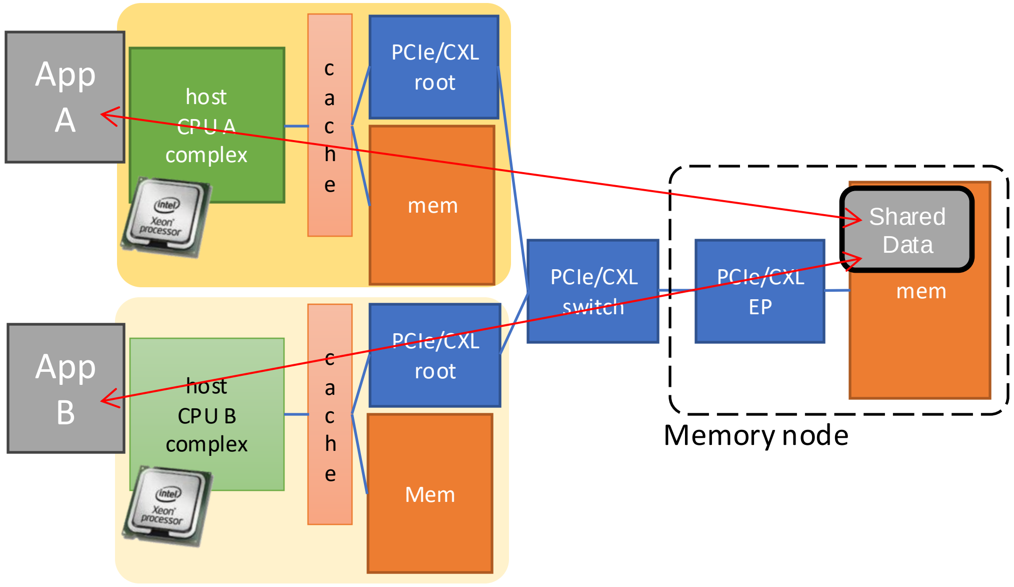
Background: applications and storage

- Modern applications requires **efficient** storage data access
  - High bandwidth, Low latency
- Hyperscalers **separate** compute nodes from storage nodes
  - Data loading overhead
- Software caches play a **key role**
  - Require additional server (cost, power)
  - Data duplication at multiple different levels (wasted memory)
  - Additional data transfers (overheads)
  - Increases latencies (time)
  - Etc.



Background: emerging CXL hardware

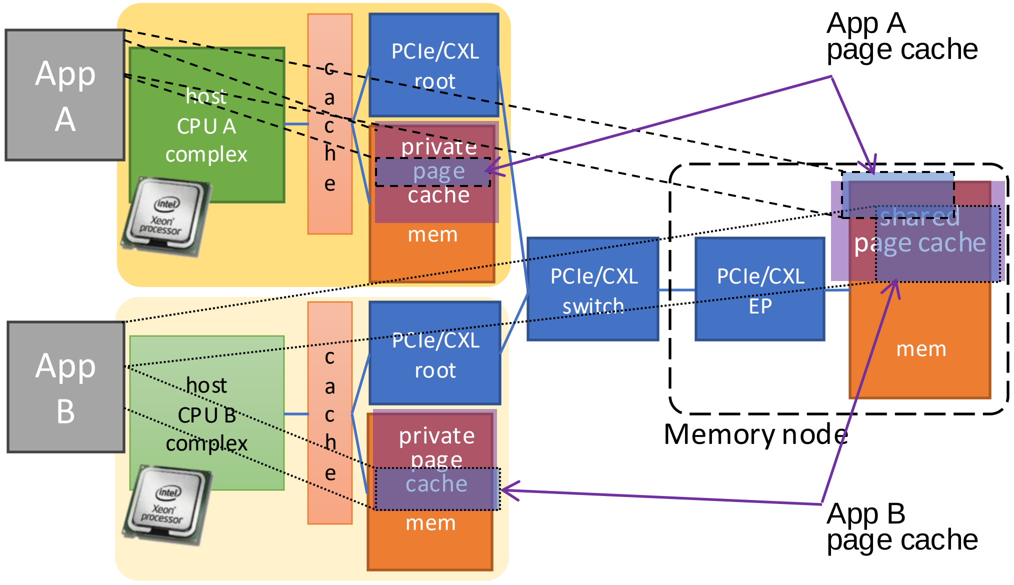
- CXL built on PCIe
- Enables disaggregated memory in data centers
- Enables **inter-machine memory sharing**
    - **HW/SW coherent**
- Byte-addressable access with latencies comparable to remote NUMA memory



Idea: Storage Cache on CXL Shared Memory

Investigate CXL shared memory pools **as a data cache tier** in data center clusters

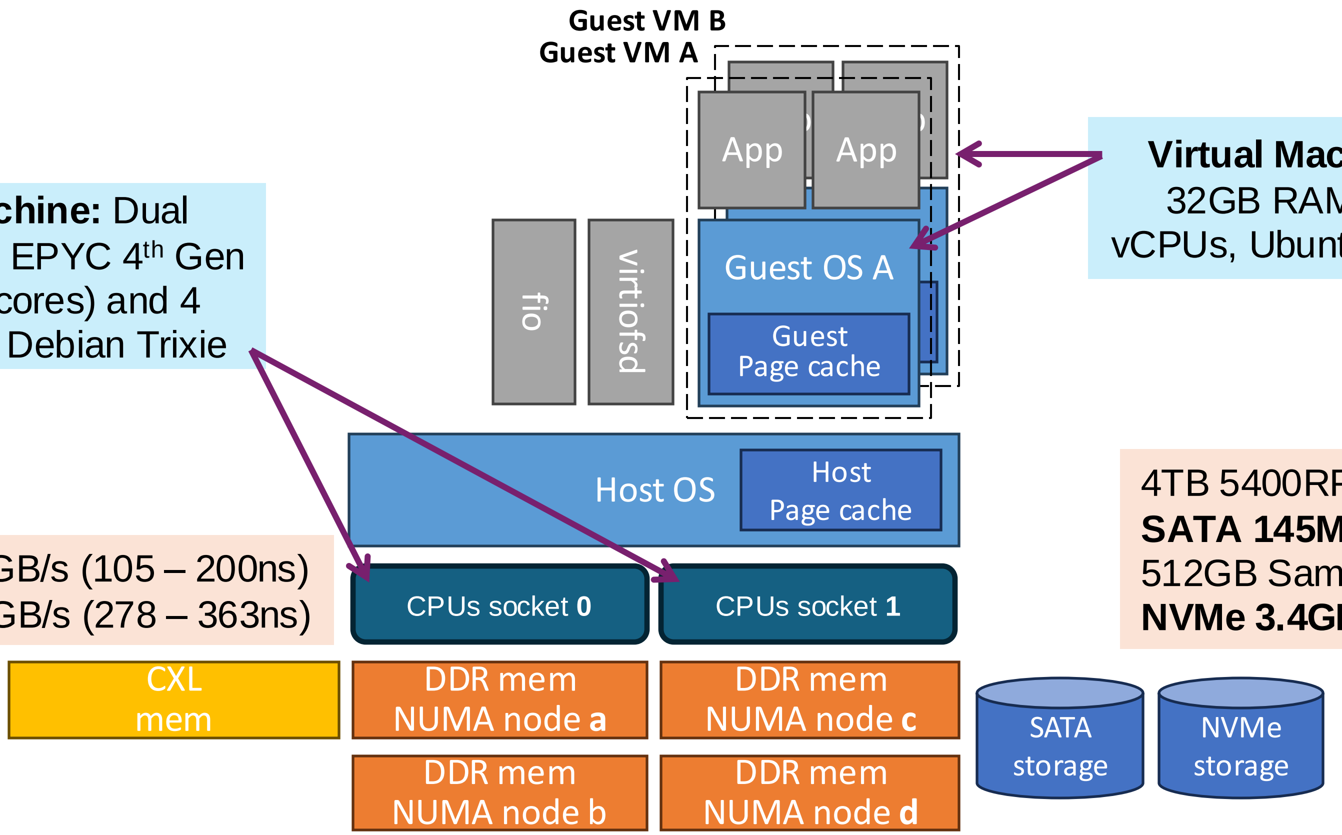
- Eliminate caching servers
- Reduce data replication/duplication
- Minimize data transfers
- Reduce latency
- Etc.



Is it worth using CXL to cache storage data? Any **problem** doing that?

… but we don’t have any CXL switch …

Prototype: multiple NUMA + CXL + KVM + virtiofsd



Host Machine: Dual Socket AMD EPYC 4th Gen 9224 (48 cores) and 4 NUMA (2!), Debian Trixie

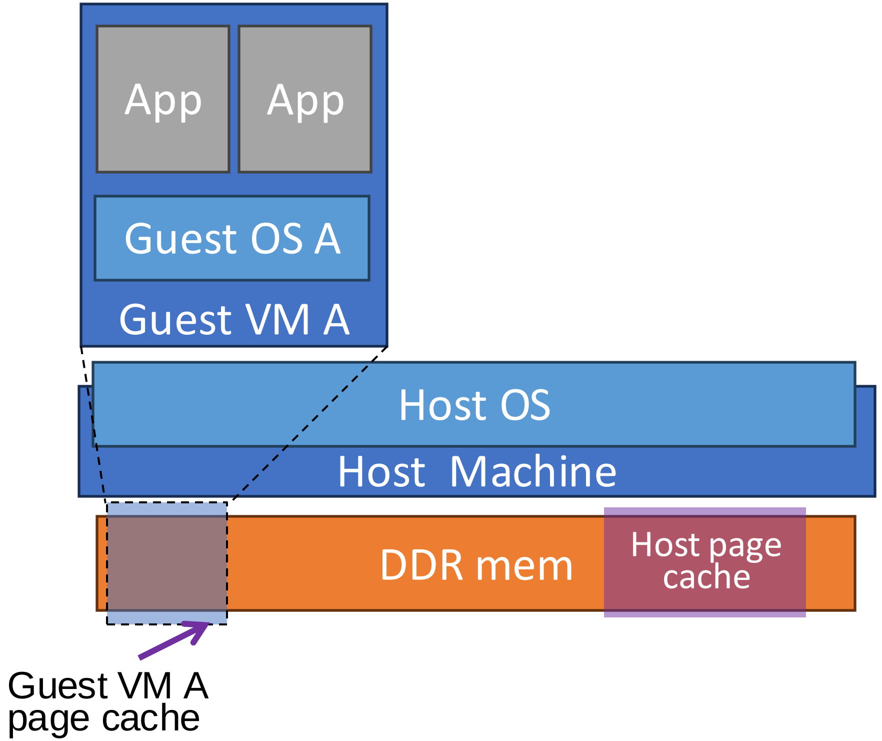
Virtual Machine: 32GB RAM, 12 vCPUs, Ubuntu 22.04

768GB DDR 104GB/s (105 – 200ns)
128GB CXL 17.5GB/s (278 – 363ns)

4TB 5400RPM Seagate
SATA 145MB/s
512GB Samsung 970P
NVMe 3.4GB/s

Benefits of Caching

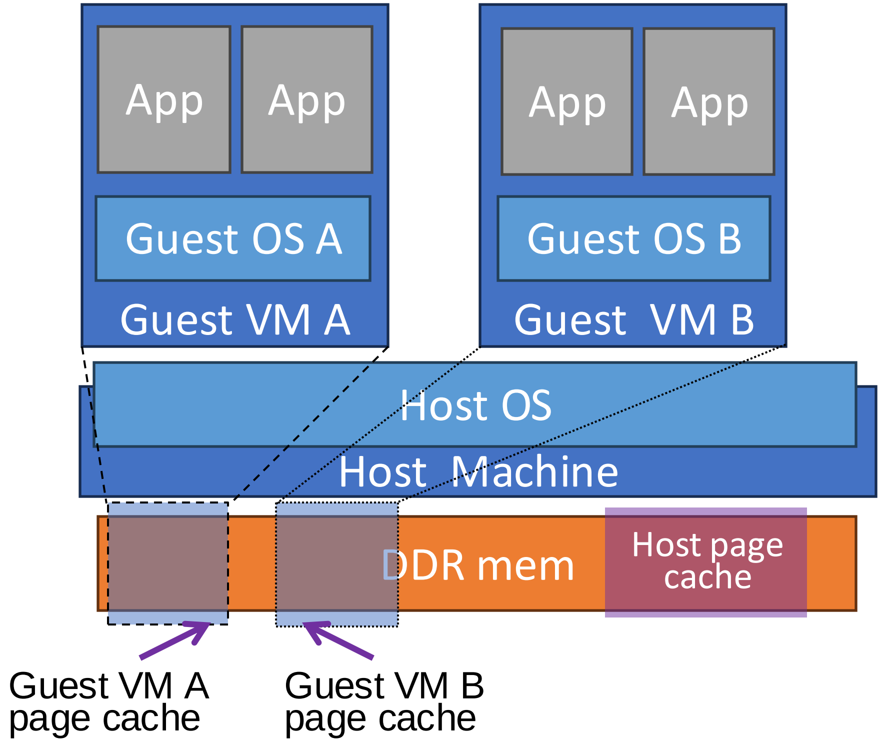
- `virtiofsd`
- SATA
  - Without caching 84.4MB/s
  - With caching 38GB/s
- NVMe
  - Without caching 2.45GB/s
  - With caching 38GB/s
- **Takeaway**
  - Caching matters (as expected)



Benefits of a Caching with File Sharing

- `virtiofsd`
- SATA
  - Without caching 145MB/s
  - With caching 70GB/s
- NVMe
  - Without caching 3.4GB/s
  - With caching 70GB/s

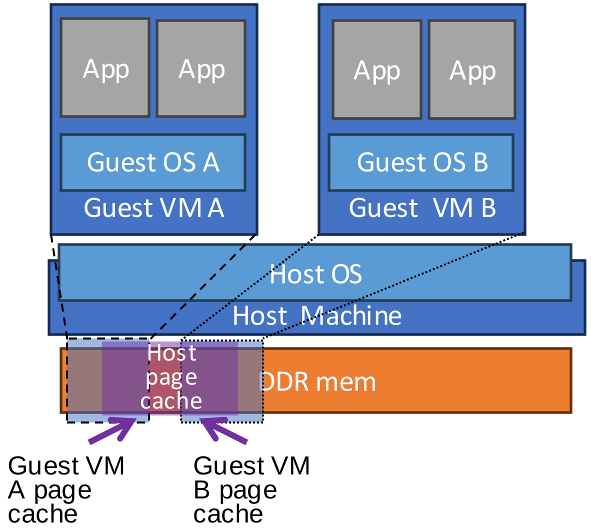
- **Takeaway**
  - File sharing further improves the achievable bandwidth

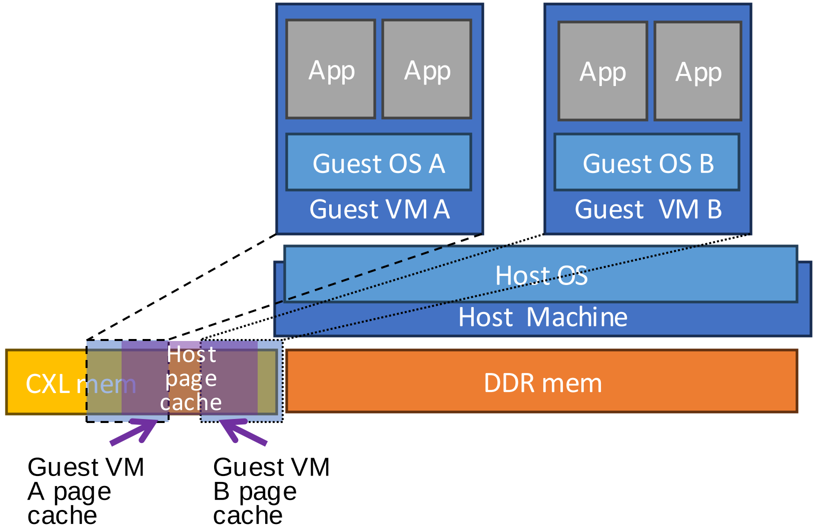


Caching on DDR vs CXL

- `virtiofsd` DAX (shared host page cache)
- SATA
  - Caching on DDR 70GB/s
  - Caching on CXL 17.4GB/s
- NVMe
  - Caching on DDR 70GB/s
  - Caching on CXL 17.4GB/s

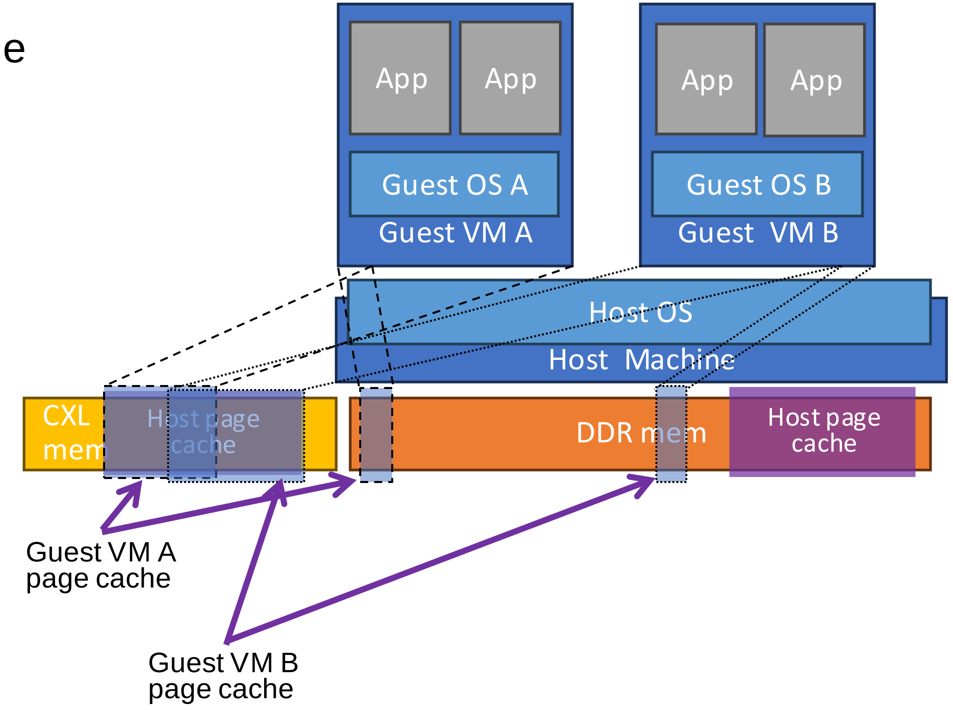
- **Takeaway**
  - The bandwidth (and latency) are constrained by the CXL device





Idea Solution

- **Hybrid** DDR-CXL shared page cache
- **Dynamically**
  - **Promote** frequently accesses chunks to DDR
  - **Demote** less frequently accessed chunks to CXL
- Kernel page cache extension

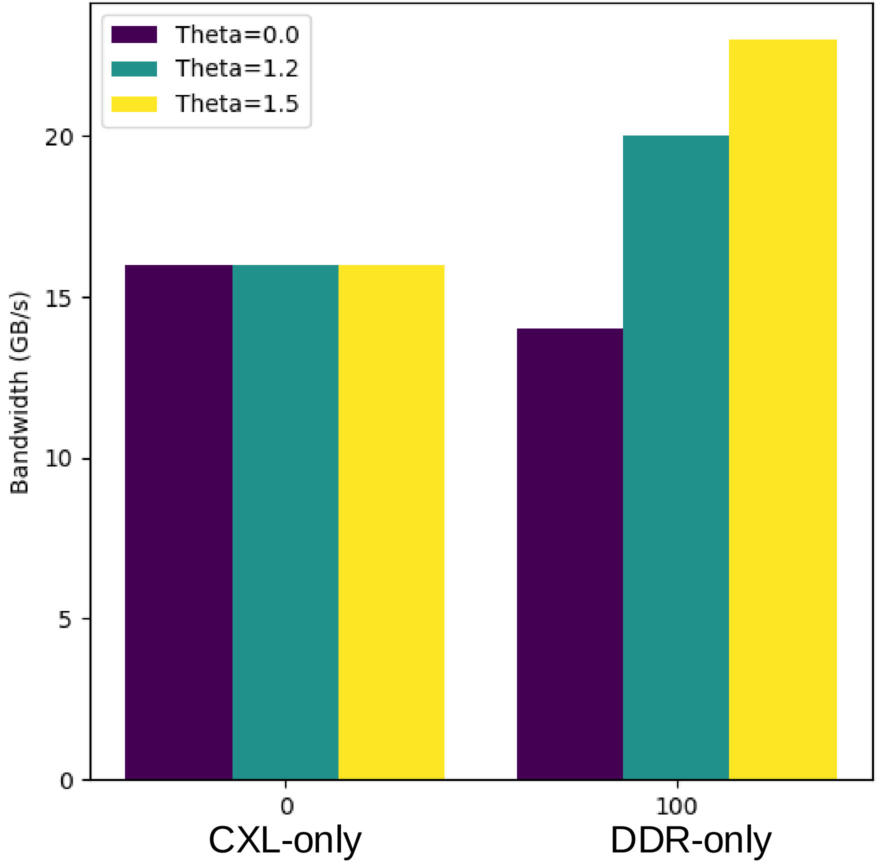


Implementation/Evaluation

- **Implement** Linux kernel patch that allocates files either
  - CXL memory
    - “Shared” files – a single copy
  - DDR memory
    - “Private” files – one copy per VM
- Set of **Python script** to control experiments

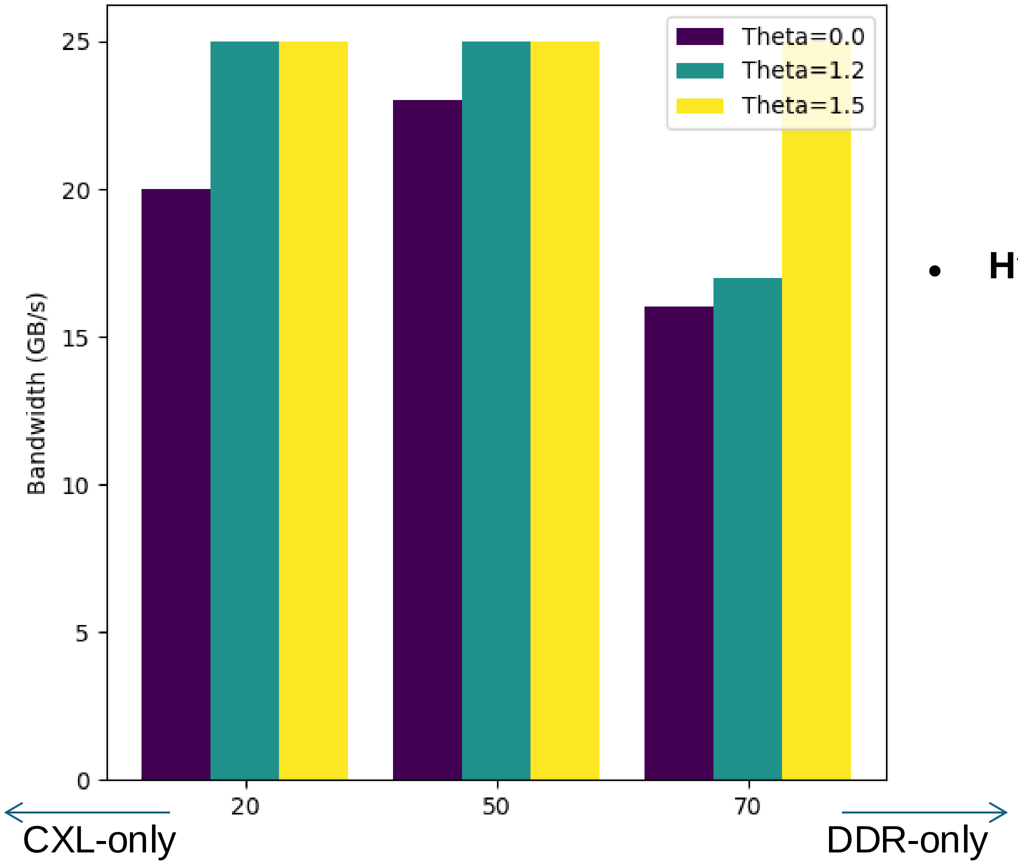
- **Evaluate** Hybrid CXL+DDR
  - Hot data in DDR memory
    - “Private” files simulate promoted data
  - Cold/shared data in CXL memory
    - Sharerd simulated demoted data
- **Simulate** Dynamic
  - Vary the amount of shared vs private files
  - Vary the access frequency of each file (Theta)
  - Theta=0.0 no skew, uniform access

Results: CXL-only vs DDR-only page-cache allocation



- **CXL-only**
  - BW capped by CXL expander
- **DDR-only**
  - Theta=0.0 low BW because of page reclaim
  - Increasing Theta, increases hit, reduces reclaim

Results: Varying CXL vs DDR page-cache allocation



- **Hybrid caching** achieves higher BW (up to 25GB/s)
  - **CXL** memory
  - Avoids OS page reclaim

  - **DDR** memory
  - Faster access

Summary

Thank you!

- We explore **CXL shared memory pool** as a storage data cache
  - Using virtualization (KVM), `virtiofsd`, NUMA, a CXL memory expander
- We show that it is a **viable solution**
  - But a naïve approach of just moving the page cache to CXL may affect performance
- We proposed a **dynamic hybrid approach** DDR+CXL page cache
  - We tested hybrid, for multiple scenarios

- Several open research questions
  - Do results hold with real hardware? How to automatize page cache placement, promotion and demotion? Consistency with shared and private copies? How to exploit CXL3.0 HW CC? How to integrate with cluster storage packages? Etc.

karim.manaouil@ed.ac.uk, antonio.barbalace@ed.ac.uk, shai.aviram.bergman@huawei.com

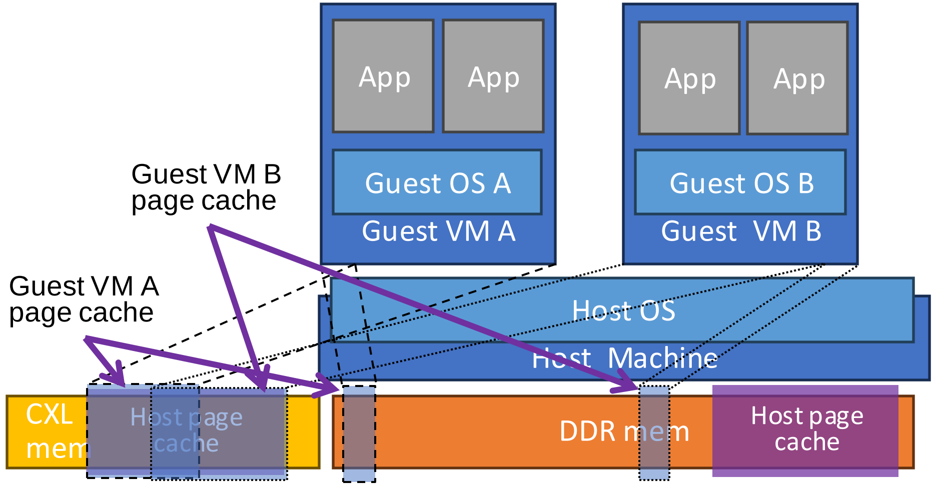
End

Idea Solution

- CXL is better than attached-storage or network attached-cache
  - Doesn't require an additional full-fledged server
- Cannot match the performance of memory
- But local memory is limited in size (and costly) and cannot be shared

- Summary: for performance, just moving the page cache to CXL is not sufficient

- Solution:
  - In-kernel page cache extension
  - Dynamic mechanism for caching data promotion and demotion at runtime
    - Promote frequently accesses chunks from CXL to local memory
    - Demote less frequently accessed chunks to CXL
- Research questions:
  - How to reengineer Linux and similar
  - How to make this dynamic?

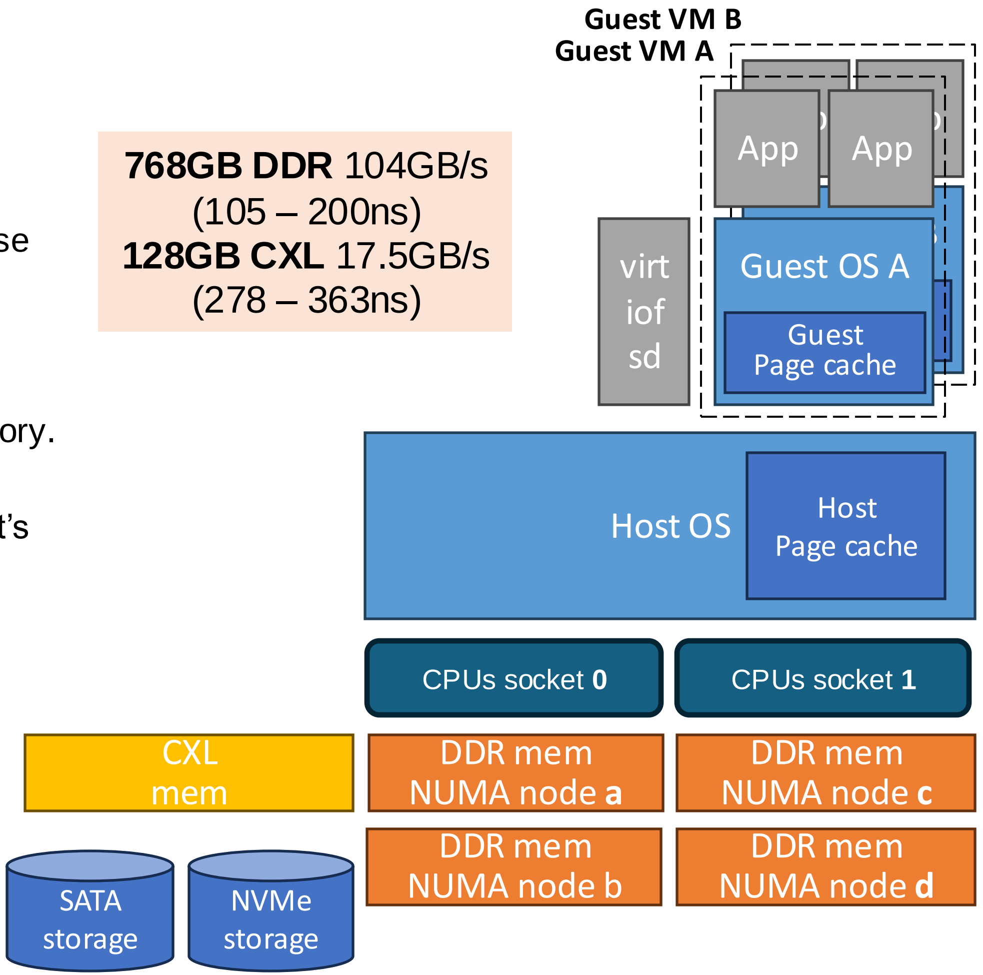


Prototype

- However, CXL switches aren't available to us, we use
  - A multi-NUMA machine with a CXL memory expander
  - VMs running on different NUMA nodes
  - Shared page cache is allocated on CXL memory.

- We use virtiofsd which allows VMs to share the host's page cache by mapping it directly into guests.

768GB DDR 104GB/s
(105 – 200ns)
128GB CXL 17.5GB/s
(278 – 363ns)



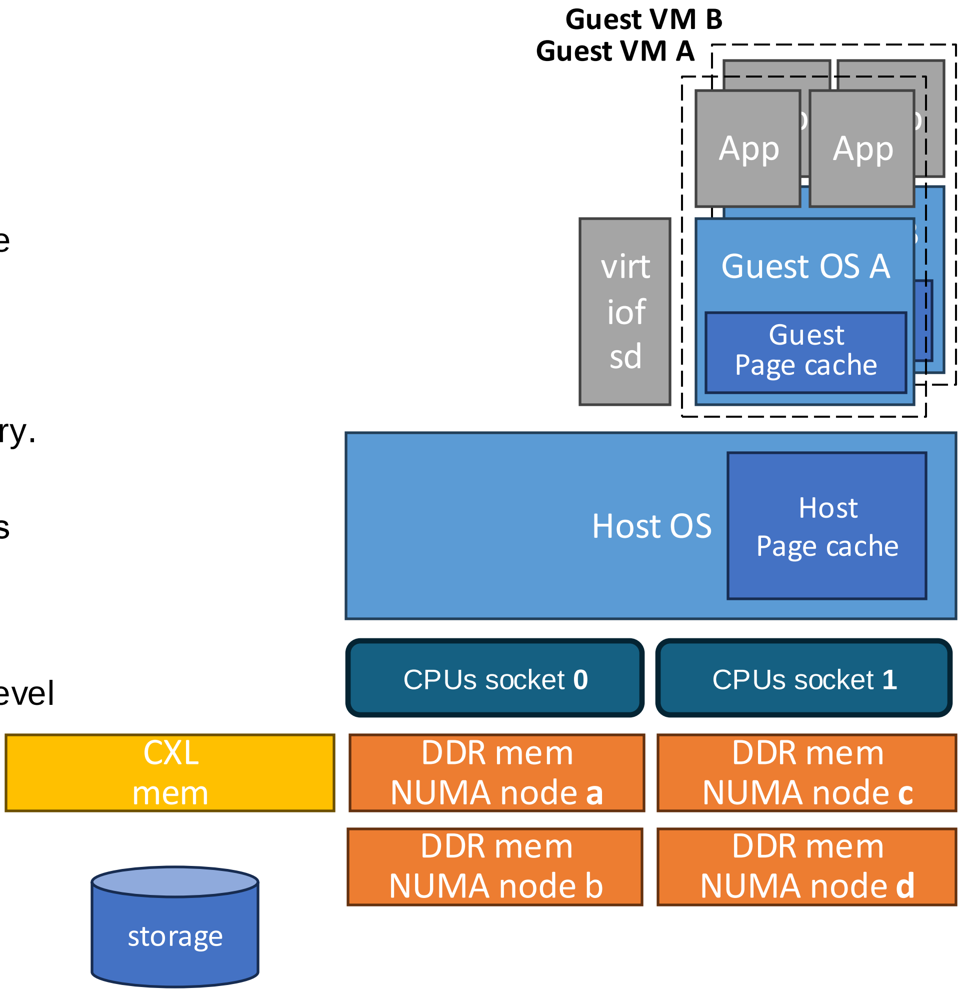
4TB 5400RPM Seagate **SATA**
512GB Samsung 970P **NVMe**

Prototype

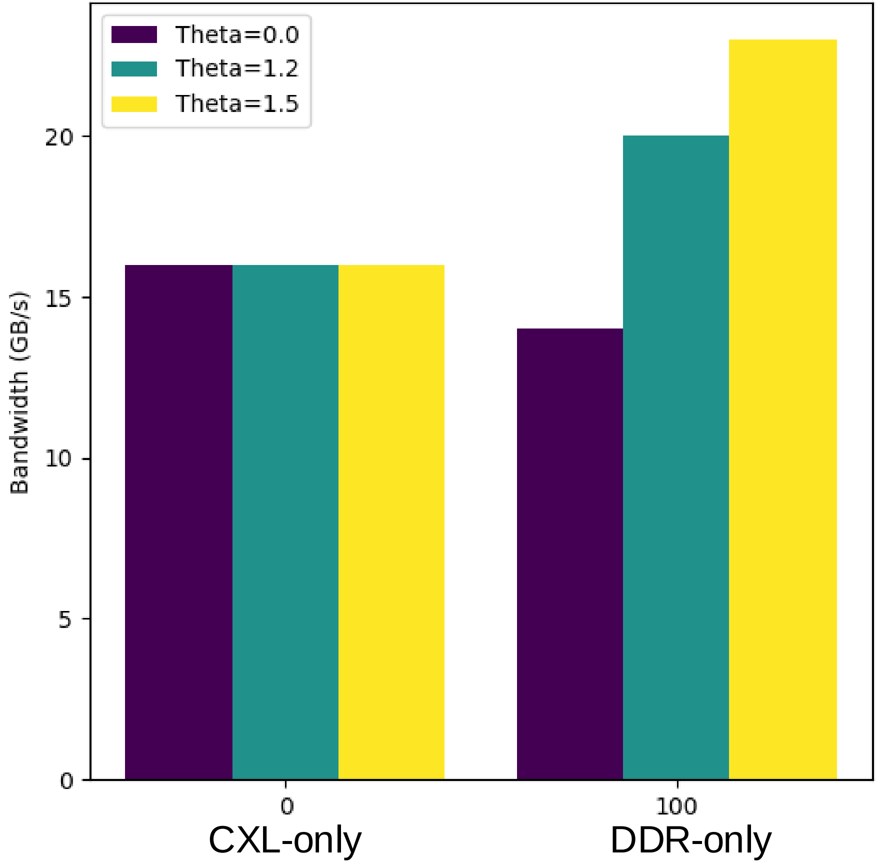
- However, CXL switches aren't available to us, we use
  - A multi-NUMA machine with a CXL memory expander
  - VMs running on different NUMA nodes
  - Shared page cache is allocated on CXL memory.

- We use virtiofsd which allows VMs to share the host's page cache by mapping it directly into guests.

- Our goal: use CXL memory to host a shared kernel-level page cache across kernel instances.



Results: CXL-only vs DDR-only page-cache allocation



- CXL-only
  - Bandwidth is capped by CXL expander at 17GB/s
- DDR-only
  - With a uniform distribution, BW is low because of reclaim activity
  - The more skewed the access distribution, the higher the BW (more page cache hits)
- To avoid reclaim and achieve acceptable BW, we must dynamically size the shared and private page caches

Solution is not dynamic, we
only simulate different
dynamic conditions

Implementation/Evaluation

- So far, relayed on virtiofsd
- Developed Linux kernel patch that allocates files either
  - CXL memory
    - “Shared” files – a single copy
  - DDR memory
    - “Private” files – one copy per VM
- This doesn’t simulate dynamic behaviour

We focus on the hybrid part here

- Evaluate Idea Solution
  - Hot data in DDR memory
    - “Private” files simulate promoted data
  - Cold/shared data in CXL memory
    - Sharerd simulated demoted data
- Vary the amount of shared vs private files
- Vary the access frequency of each file (Theta)
- Theta=0.0 no skew, uniform access