CXL and the Art of Hierarchical Memories

Their Management and Use

Andy Banta, Storage Janitor, Magnition Inc.
Kin-Yip Liu, Senior Director Solutions Architecture, AMD
Andy Banta

Magnition.io (Consultant)
SolidFire (VMware development, acq. by NetApp)
DataGravity (Container exploitation lead)
VMware (iSCSI Tech Lead, IPO)
Sun Microsystems (Initial Fibre Channel development)
Patent, early distributed network projects, data acquisition
@andybanta
Kin-Yip Liu

AMD (Sr. Director, Solutions Architecture; Networking & Storage)
Intel (Sr. Director, Architecture; Persistent Memories)
Marvell/Cavium Networks (Sr. Director, Solutions Architecture;
Networking, Security, 3G/LTE/5G Infrastructure, Telco NFV)
Intel (Architect, Designer; Server/Network/Mobile Processors)
kin-yip.liu@amd.com
NUMA overview

- Non-Uniform Memory Architecture
- Not all memory access is created equal

```
# numactl -H
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
node 0 size: 64057 MB
node 0 free: 48756 MB
node 1 cpus: 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
node 1 size: 64003 MB
node 1 free: 50473 MB
node 2 cpus:
node 2 size: 255921 MB
node 2 free: 255918 MB
node 3 cpus:
node 3 size: 255623 MB
node 3 free: 255631 MB
node distances:
0: 10 21 17 28
1: 21 10 28 17
2: 17 28 10 28
3: 28 17 28 10
```
CXL Roadmap Drives Memory Hierarchy Innovation

CXL 1.1
Coherent memory expansion
*Boost capacity and/or bandwidth*

CXL 2.0
Pooling, CXL switches
*Improve overall TCO and memory utilization*

CXL 3.0
Coherent memory sharing
*New and fast sharing of data*

* Only a subset of CXL features and benefits are highlighted here.
CXL Enables More Memory and Hierarchy Options

- **Cache hierarchy**
  - DRAM, HBM
    - (local, same/remote socket)
  - 2nd tier memory
    - (local same/remote socket and/or pooled)
  - Storage class / Persistent memories
    - (local same/remote socket and/or pooled)
  - Storage hierarchy
    - (hot/warm/cold tiers)

- **Memory pools enable dynamic resource allocation among compute nodes with exclusive and/or shared data accesses**

- **Advantages**:
  - Shorter latencies
  - Byte-addressable
  - Finer access granularity
  - Lower per byte cost
  - Larger capacity
  - Disaggregation
  - Non-volatile
Workload Performance Tuning Considerations

### Memory Performance
- Characteristics: data rate, latency, read vs. write performance, access granularity, persistency. CXL adds variety, abstraction
- Memory channels, DIMMs per channel, module slots
- Capacity vs. Bandwidth boost. Interleaving options

### NUMA, Affinity, Latency Optimization
- NUMA within socket, across socket, beyond compute node
- Compute and memory bandwidth allocation per NUMA node
- Scheduling processes to NUMA nodes. Dynamic realignment

### Data Management
- Efficient tracking of hot/cold data, and migration among tiers
- Telemetries. Workload profiles
- Accelerator, compute-in-storage
Methods of dealing with latencies

- Understand that not all memory access is equal
- Developers need to understand and deal with differences
  - At the risk of inconsistent performance results
- Segregation
- Tiering
- Dynamic re-alignment
Segregation

- **System level assignment**
  - Assign VMs memory from a specific NUMA node
  - Spread VMs across NUMA nodes and assign memory
  - Assign processes memory from specific NUMA nodes

- **Application-based selection**
  - Use libraries inside applications to tier memory
Assigning NUMA affinity in VMware

- Advanced settings on a VM configuration
- Without setting the affinity, VMware chooses memory, leading to unpredictable performance

Figure courtesy of Fortinet
Configuring VMs across NUMA Nodes

- Large VMs can split across NUMA nodes
- Memory affinity for each virtual CPU stays on node

Figure courtesy of frankdennemann.nl
Node-picking in Linux

- numactl(8)
  - Process level
  - --cpunodebind
  - --membind
  - --localalloc
  - --preferrednode
  - --interleave
Application-level segregation

- **SNIA PMDK**
  - For persistent memory
- **VMware presents pmem resources to VMs**

```c
if (pmem2_config_set_required_store_granularity(cfg, PMEM2_GRANULARITY_PAGE)) {
    pmem2_perror("pmem2_config_set_required_store_granularity");
    exit(1);
}

if (pmem2_map_new(&map, cfg, src)) {
    pmem2_perror("pmem2_map_new");
    exit(1);
}

char *addr = pmem2_map_get_address(map);
size_t size = pmem2_map_get_size(map);
strcpy(addr, "hello, persistent memory");
persist = pmem2_get_persist_fn(map);
persist(addr, size);
```
Caching

- Hit-n-miss
- Promotion and demotion
- Complexity of tiering
  - How rapidly this becomes an unmanageable problem
Caching

- **Hit or miss**
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload
Caching

- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload
Caching

- **Hit or miss**
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload
Caching

- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload
Caching

- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload
Caching

- Promotion and demotion
  - Less frequently used
  - Uses less space
  - Requires more data movement

- SDXI (Smart Data Accelerator Interface)
  - Rapid memory to memory data mover
  - SNIA working group
  - “Most Innovative” at Flash Memory Summit in June 2023
  - Tuesday’s SDXI talk at SDC
Built-in caching

- VMware automatic memory tiering
- Linux numactl(8) and allows built-in tiering
  - Promote/demote
  - Based on node distance
Multi-level caching

- Currently used in Content Delivery Networks
- Useful for new HPC and large-scale hosts for main memory
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts
Multi-level caching within CXL hosts

7 different NUMA nodes plus contention
Modeling and optimizing

- Workload dependent
- Static vs. dynamic reallocation
Workloads matter

- No artificial workloads
- Content delivery
- Inference
- Learning
- File serving

```c
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1) = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1) = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
```
Static Analysis with Simulations

- Determine initial configurations
- Build behavioral simulators
- Mix pre-built components and custom as needed
Engineering Means Simulation
Engineering Means Simulation
Engineering Means Simulation
Engineering Means Simulation
Engineering Means Simulation
Engineering Means Simulation
Engineering Means Simulation

- Cheaper, faster, more flexible than system building
Engineering Means Simulation

- Cheaper, faster, more flexible than system building
- Engineering design uses simulations, why not software?
Code as System Simulation

- Cheaper, faster, more flexible than system building
- Engineering design uses simulations. Why not software?
Details, details

- Each component can be modeled
- Variables are easy to introduce
Results are easy to compare

- Run millions of runs
- More variables = more options
Dynamic reconfiguration in running environment

- Useful for running varied workloads
- Make the most of existing hardware
- Limited to software and sizing changes
Working at enterprise scale

- 100+ cores
- 100+ PCIe lanes
  - CXL capabilities
  - Network capabilities
  - Not limited to memory or IO bound loads
- Clusters of numerous nodes
What the future holds

7 different NUMA nodes plus contention
Today I Learned

- CXL offers a lot of flexibility
Today I Learned

- CXL offers a lot of flexibility and complexity
Today I Learned

- CXL offers a lot of flexibility and complexity
- OS vendors are helping
Today I Learned

- CXL offers a lot of flexibility and complexity
- OS vendors are helping
- More flexibility requires more system design
Today I Learned

- CXL offers a lot of flexibility and complexity
- OS vendors are helping
- More flexibility requires more system design
- Optimized system design requires simulations
As an example, a current customer has achieved the following measurable outcomes with Magnition:

Experiments per day per engineer
- Without Magnition: 2
- With Magnition: 50,000+

Parameter variations tested before prod release
- Without Magnition: 50
- With Magnition: 1,000,000+

Workload performance improvement using our products to find optimal out-of-the-box settings: 10-50%+
AMD Summary

- CXL is a high-performance interconnect standard which has strong industry support and roadmap for driving system architecture and memory hierarchy innovation.

- AMD is a Board of Director of the CXL Consortium, and supports CXL in current and roadmap products.

- AMD works with a rich set of CXL eco-system partners to drive innovative solutions for a variety of applications especially in storage segment.
Please take a moment to rate this session.

Your feedback is important to us.
Section Title

Section Subtitle
Light Slide Title

- Bullets 1
  - Bullets 2
    - Bullets 3
      - Bullets 4
        - Bullets 5
Dark Slide Title

- Bullets 1
  - Bullets 2
    - Bullets 3
      - Bullets 4
      - Bullets 5
Considerations for Hierarchy Options

- Performance, read vs. write, granularity
- Latencies
- TCO
- Software tier management
- Dynamic changes flexibility
- Mix of different parts with different performance characteristics
- Interleaving
- Optimization focus, capacity vs. bandwidth
- Persistency
- NUMA