Practical Near-Data Processing for In-Memory Analytics Frameworks

Mingyu Gao, Grant Ayers, Christos Kozyrakis

Stanford University
http://mast.stanford.edu
Motivating Trends

- End of Dennard scaling $\rightarrow$ systems are energy limited
- Emerging big data workloads
  - Massive datasets, limited temporal locality, irregular access patterns
  - They perform poorly on conventional cache hierarchies
- Need alternatives to improve energy efficiency

Figs: http://oceanaute.blogspot.com/2015/06/how-to-shuffle-sort-mapreduce.html
PIM & NDP

- Improve performance & energy by avoiding data movement
- Processing-In-Memory (1990’s – 2000’s)
  - Same-die integration is too expensive
- Near-Data Processing
  - Enabled by 3D integration
  - Practical technology solution
  - Processing on the logic die

Hybrid Memory Cube (HMC)
High Bandwidth Memory (HBM)

Figs: www.extremetech.com
Base NDP Hardware

- Stacks linked to host multi-core processor
  - Code with temporal locality: runs on host
  - Code without temporal locality: runs on NDP

- 3D memory stack
  - x10 bandwidth, x3-5 power improvement
  - 8-16 vaults per stack
    - Vertical channel
    - Dedicated vault controller
  - NDP cores
    - General-purpose, in-order cores
    - FPU, L1 caches I/D, no L2
    - Multithreaded for latency tolerance
Challenges and Contributions

- NDP for large-scale highly distributed analytics frameworks
  - General coherence maintaining is expensive
    ✓ Scalable and adaptive software-assisted coherence
  - Inefficient communication and synchronization through host processor
    ✓ Pull-based model to directly communicate, remote atomic operations
  - Hardware/software interface
    ✓ A lightweight runtime to hide low-level details to make program easier
  - Processing capability and energy efficiency
    ✓ Balanced and efficient hardware

- A general, efficient, balanced, practical-to-use NDP architecture
Example App: PageRank

- Edge-centric, scatter-gather, graph processing framework
- Other analytics frameworks have similar behaviors
Architecture Design

Memory model, communication, coherence, ...
Lightweight hardware structures and software runtime
Shared Memory Model

- Unified physical address space across stacks
  - Direct access from any NDP/host core to memory in any vault/stack

- In PageRank
  - One thread to access data in a remote graph partition
    - For edges across two partitions

- Implementation
  - Memory ctrl forwards local/remote accesses
  - Shared router in each vault
Virtual Memory Support

- NDP threads access virtual address space
  - Small TLB per core (32 entries)
  - Large pages to minimize TLB misses (2 MB)
  - Sufficient to cover local memory & remote buffers

- In PageRank
  - Each core works on local data, much smaller than the entire dataset
  - 0.25% miss rate for PageRank

- TLB misses served by OS in host
  - Similar to IOMMU misses in conventional systems
Software-Assisted Coherence

- Maintaining general coherence is expensive in NDP systems
  - Highly distributed, multiple stacks

- Analytics frameworks
  - Little data sharing except for communication
  - Data partitioning is coarse-grained

- Only allow data to be cached in one cache
  - Owner cache
  - No need to check other caches

- Page-level coarse-grained
  - Owner cache configurable through PTE
Software-Assisted Coherence

- **Scalable**
  - Avoids directory lookup and storage

- **Adaptive**
  - Data may overflow to other vault
  - Able to cache data from any vault in local cache

- **Flush only when owner cache changes**
  - Rarely happen as dataset partitioning is fixed
Communication

- Pull-based model
  - Producer buffers intermediate/result data **locally** and **separately**
  - Post small message (address, size) to consumer
  - Consumer **pulls** data when it needs with load instructions

---

**Diagram:**
- Task → Cores
- Process
- Buffer
- Pull

---

12
Pull-based model is efficient and scalable
- Sequential accesses to data
- Asynchronous and highly parallel
- Avoids the overheads of extra copies
- Eliminates host processor bottleneck

In PageRank
- Used to communicate the update lists across partitions
Communication

- **HW optimization: remote load buffer (RLBs)**
  - A small buffer per NDP core (a few cachelines)
  - Prefetch and cache remote (sequential) load accesses
    - Remote data are not cache-able in the local cache
    - Do not want owner cache change as it results in cache flush

- **Coherence guarantee with RLBs**
  - Remote stores bypass RLB
    - All writes go to the owner cache
    - Owner cache always has the most up-to-date data
  - Flush RLBs at synchronization point
    - ... at which time new data are guaranteed to be visible to others
    - Cheap as each iteration is long and RLB is small
Synchronization

- Remote atomic operations
  - Fetch-and-add, compare-and-swap, etc.
  - HW support at memory controllers [Ahn et al. HPCA’05]

- Higher-level synchronization primitives
  - Build by remote atomic operations
  - E.g., hierarchical, tree-style barrier implementation
    - Core → vault → stack → global

- In PageRank
  - Build barrier between iterations
Software Runtime

- Hide low-level coherence/communication features
  - Expose simple set of API

- Data partitioning and program launch
  - Optionally specify running core and owner cache close to dataset
  - No need to be perfect, correctness is guaranteed by remote access

- Hybrid workloads
  - Coarsely divide work between host and NDP by programmers
    - Based on temporal locality and parallelism
  - Guarantee no concurrent accesses from host and NDP cores
Evaluation

Three analytics framework: MapReduce, Graph, DNN
Methodology

- Infrastructure
  - zsim
  - McPAT + CACTI + Micron’s DRAM power calculator

- Calibrate with public HMC literatures

- Applications
  - MapReduce: Hist, LinReg, grep
  - Graph: PageRank, SSSP, ALS
  - DNN: ConvNet, MLP, dA
Porting Frameworks

- MapReduce
  - In map phase, input data streamed in
  - Shuffle phase handled by pull-based communication

- Graph
  - Edge-centric
  - Pull remote update lists when gathering

- Deep Neural Networks
  - Convolution/pooling layers handled similar to Graph
  - Fully-connected layers use local combiner before communication

- Once the framework is ported, no changes to the user-level apps
2.9x performance and energy improvement

- Edge-centric version optimize for spatial locality
- Higher utilization for cachelines and DRAM rows
Balance: PageRank

- Performance scales to 4-8 cores per vault
  - Bandwidth saturates

- Final design
  - 4 cores per vault
  - 1.0 GHz
  - 2-threaded
  - Area constrained
Performance scales well up to 16 stacks (256 vaults, 1024 threads)
- Inter-stack links are not heavily used
Final Comparison

- Four systems
  - Conv-DDR3
    - Host processor + 4 DDR3 channels
  - Conv-3D
    - Host processor + 8 HMC stacks
  - Base-NDP
    - Host processor + 8 HMC stacks with NDP cores
    - Communication coordinated by host
  - NDP
    - Similar to Base-NDP
    - With our coherence and communication
Final Comparison

- Conv-3D: improve 20% for Graph (bandwidth-bound), more energy
- Base-NDP: 3.5x faster and 3.4x less energy than Conv-DDR3
- NDP: up to 16x improvement than Conv-DDR3, 2.5x over Base-NDP
Hybrid Workloads

- Use both host processor and NDP cores for processing
- NDP portion: similar speedup
- Host portion: slight slowdown
  - Due to coarse-grained address interleaving

Execution Time Breakdown
Conclusion

- Lightweight hardware structures and software runtime
  - Hides hardware details
  - Scalable and adaptive software-assisted coherence model
  - Efficient communication and synchronization

- Balanced and efficient hardware

- Up to 16x improvement over DDR3 baseline
  - 2.5x improvement over previous NDP systems

- Software optimization
  - 3x improvement from spatial locality
Thanks!

Questions?