A Study of Pointer-Chasing Performance on Shared-Memory Processor-FPGA Systems

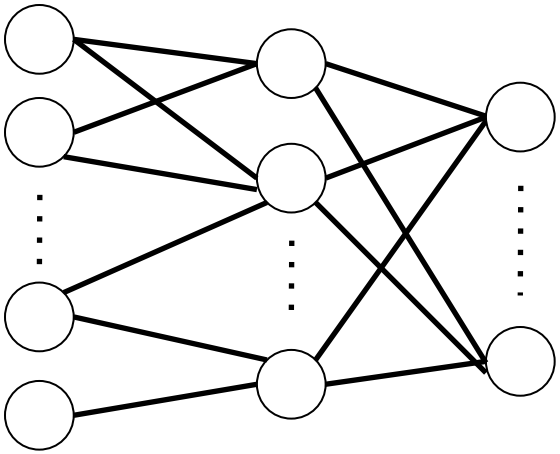
Gabriel Weisz,[1] Joseph Melber, Yu Wang, Kermin Fleming,[2] Eriko Nurvitadhi[2] and James C. Hoe

[1]Now at USC/ISI

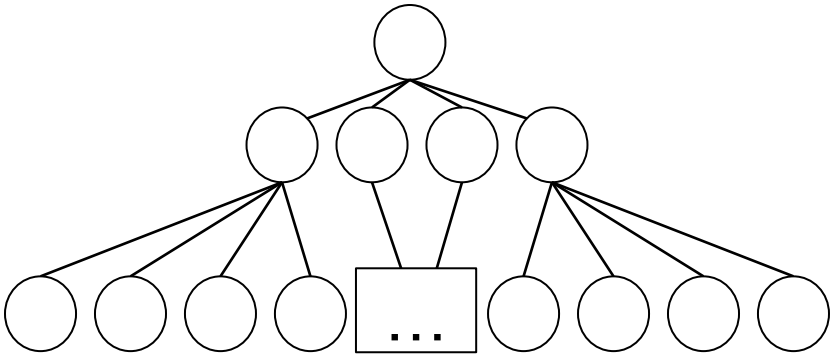
[2]Intel Corporation

Computer Architecture Lab at Carnegie Mellon

Pointer Chasing FPGA Applications



Machine Learning - Graphs



Databases - Trees

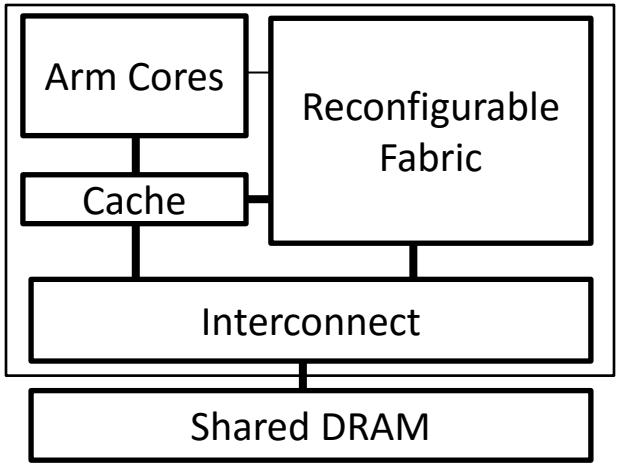
Graph and database applications on FPGAs:

- [Ovtcharov+ HotChips 2015, Chalamalasetti+ FPGA 2013, Choi+ FPPL 2012, Dennl+ FCCM 2012, Betkaoui+ FPT 2011]
- At least 5 graph and neural net papers at FPGA 2016

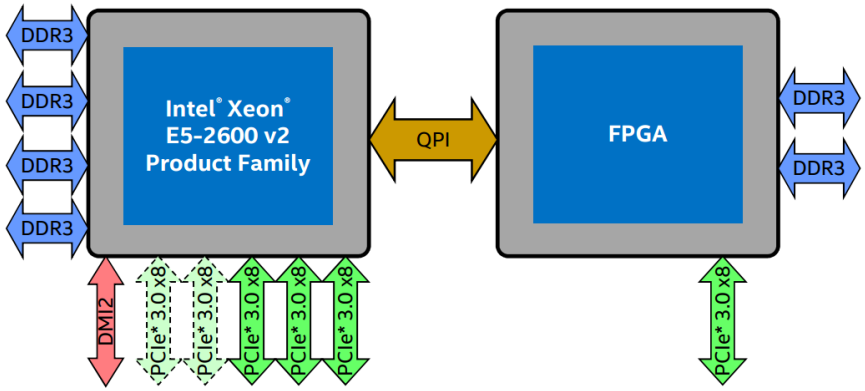
How should we handle pointer-based data structures on shared memory CPU-FPGA machines?

Shared Memory CPU-FPGA Systems

Package-level integration:
Xilinx Zynq and Altera SoC



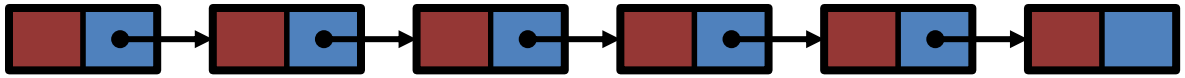
System-level integration:
Intel QPI and IBM CAPI



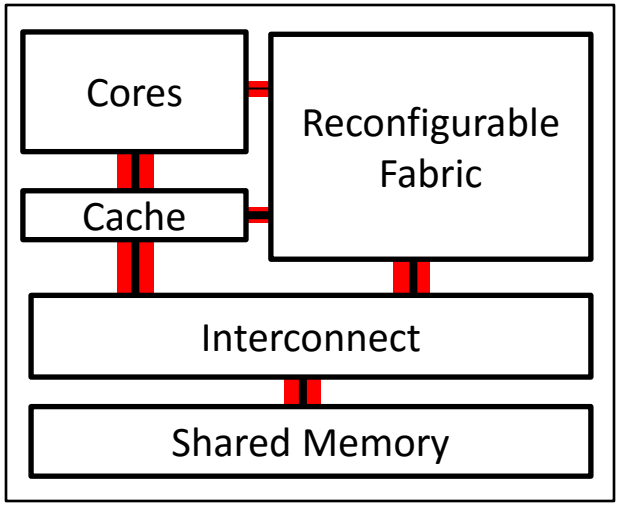
[Figure source: Intel slides from CARL 2015]

Can the processor accelerate pointer chasing memory accesses for the FPGA?

Executive Summary

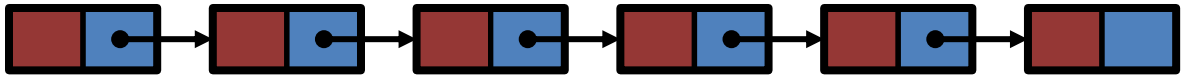


We reduce pointer chasing to linked list traversal in order to explore the core memory accesses

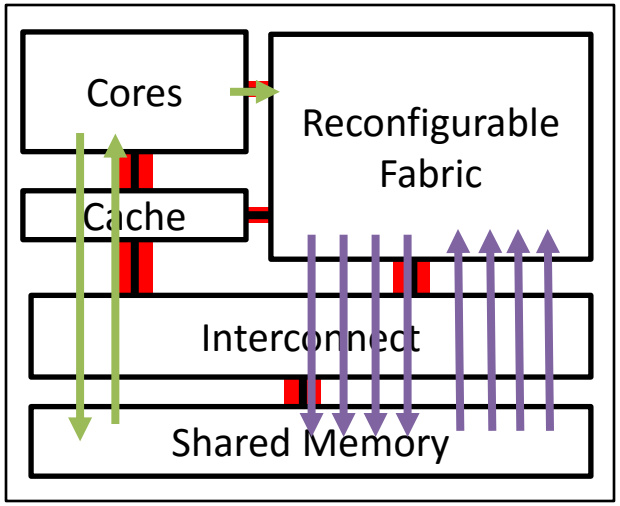


We investigate different pointer chasing mechanisms using all of the interfaces available on shared memory CPU-FPGA systems

Executive Summary



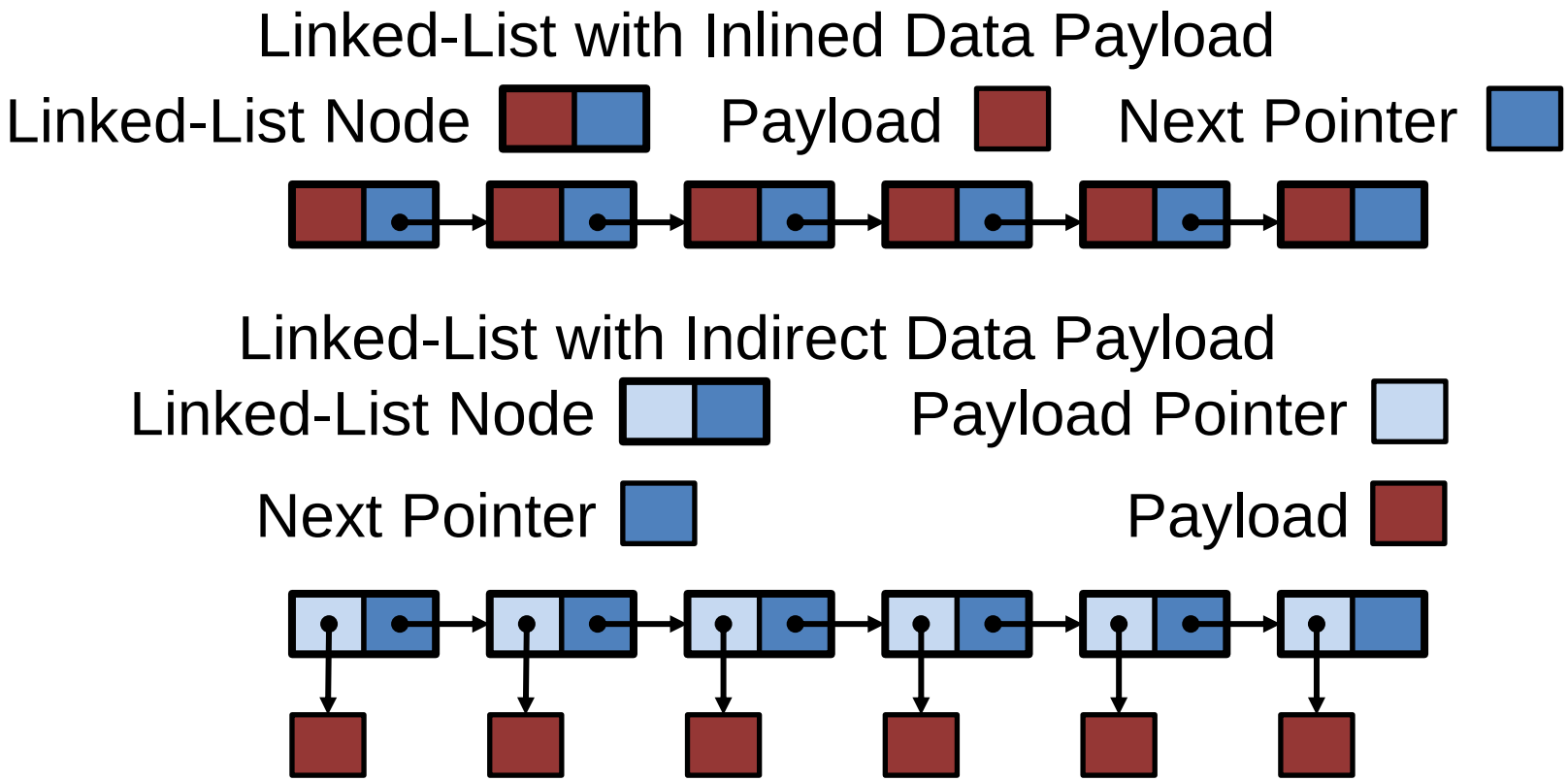
We reduce pointer chasing to linked list traversal in order to explore the core memory accesses



We investigate different pointer chasing mechanisms using all of the interfaces available on shared memory CPU-FPGA systems

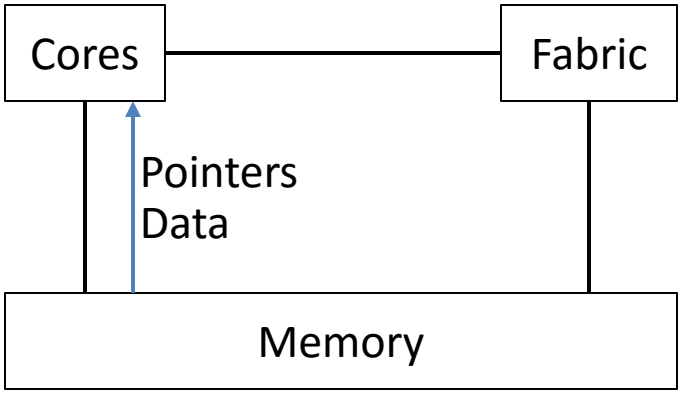
We find that processor assistance and parallel traversal mechanisms can improve performance on inefficient memory access patterns

Case Study: Traverse Linked List Sending Payload to FPGA Fabric

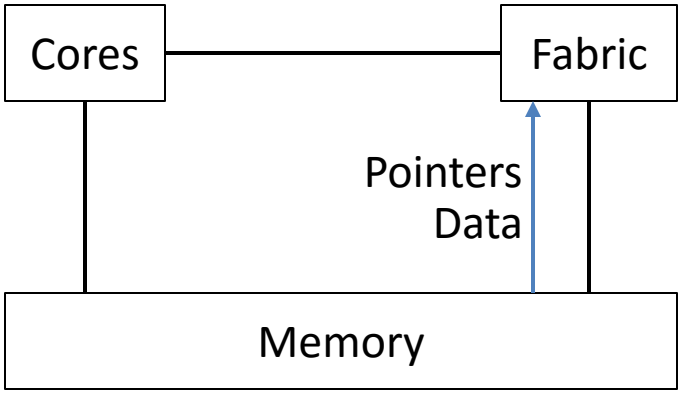


We test best-case (sequential) and worst case (strided) data layouts

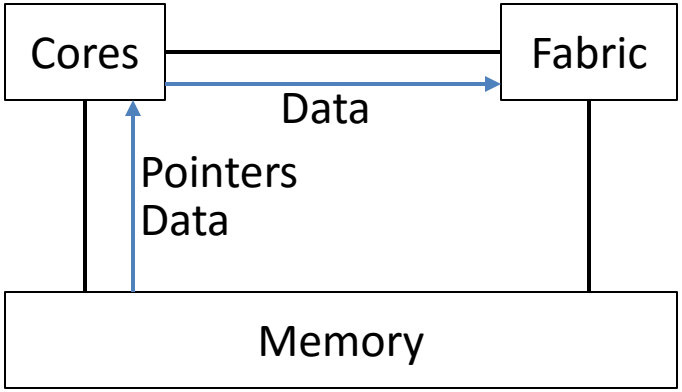
Traversal Approaches



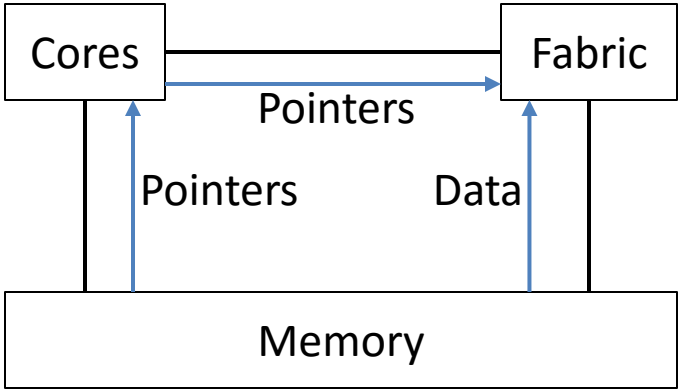
Software-only (reference)



Fabric-only



Hybrid-push

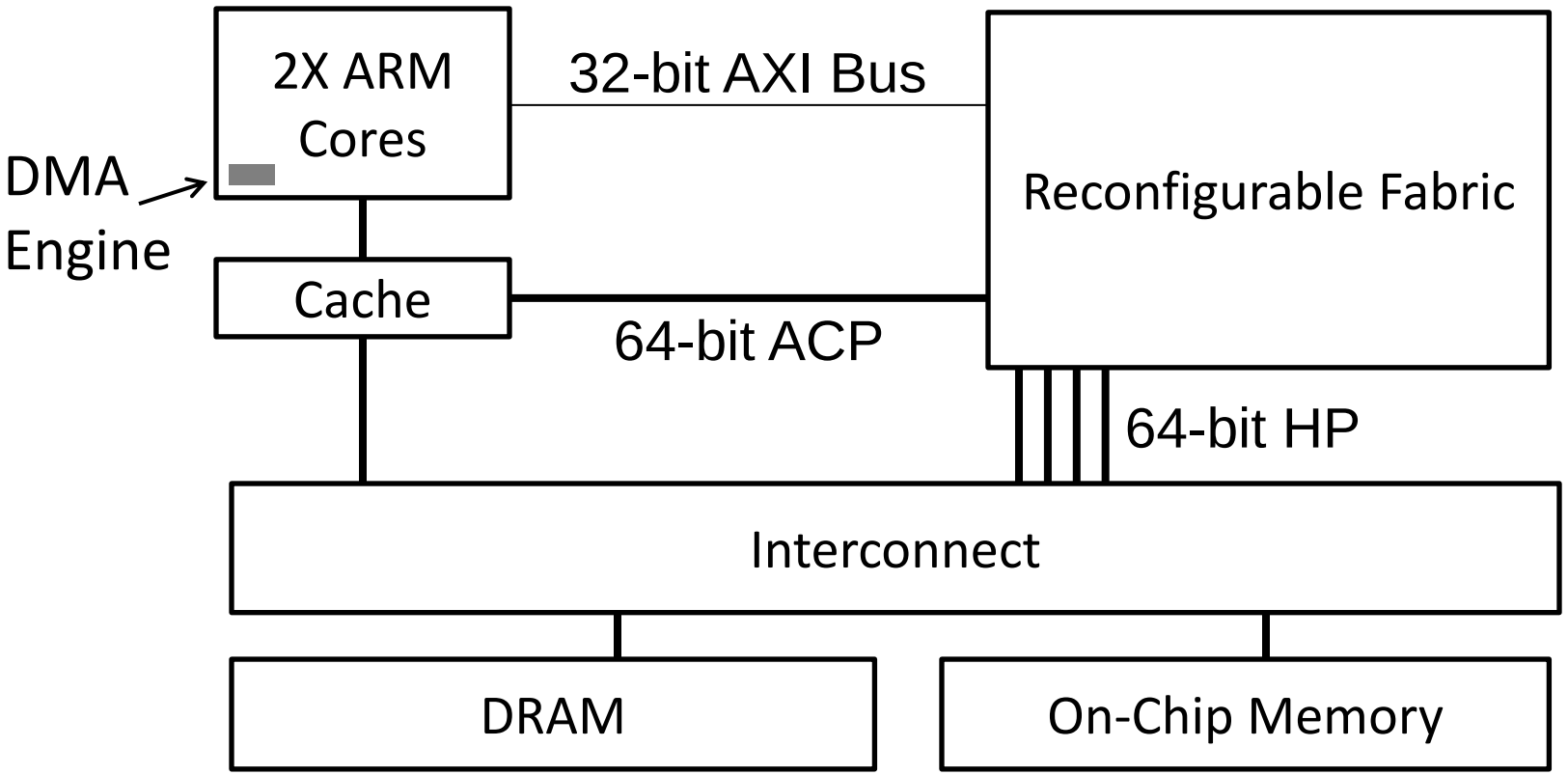


Hybrid-pull
(Valid when traversal is not dependent on payload data)

Experimental Parameters

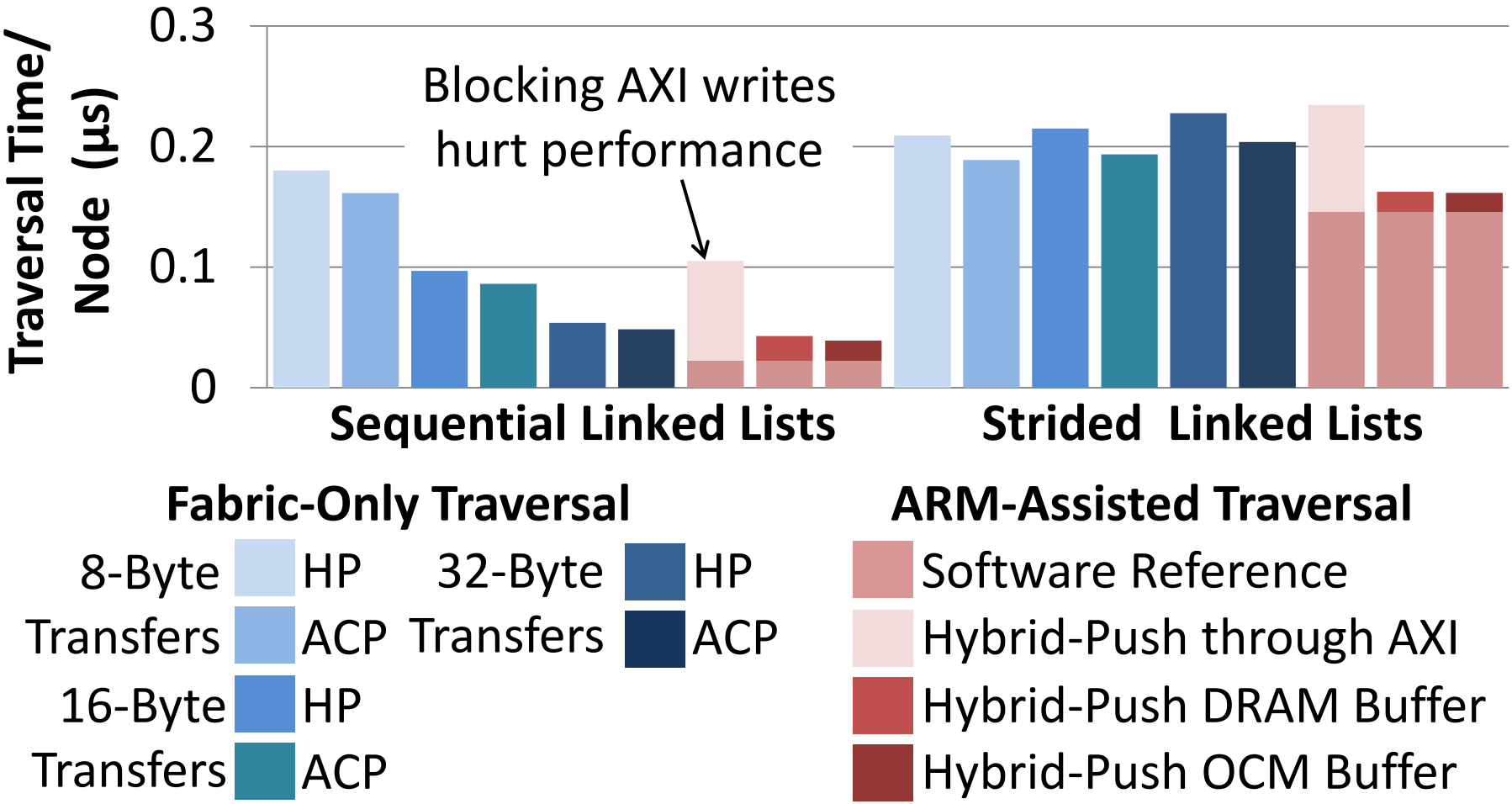
Option	Choices
List length	Fixed at 16k items
Memory layout	Packed or 16k Strides
Payload location	Inline or Indirect
Node/payload Size	8 bytes – 1 kilobyte
Traversal approach	Software-only (reference), fabric-only, hybrid-pull, hybrid-push
Fabric-only parallelism	1-128 ways
Fabric-only block size	8, 16, 32 Bytes (Zynq) 64 Bytes (Intel)
Device-specific parameters	Which memory interface Processor-fabric data transfer mechanism

Zynq Datapath

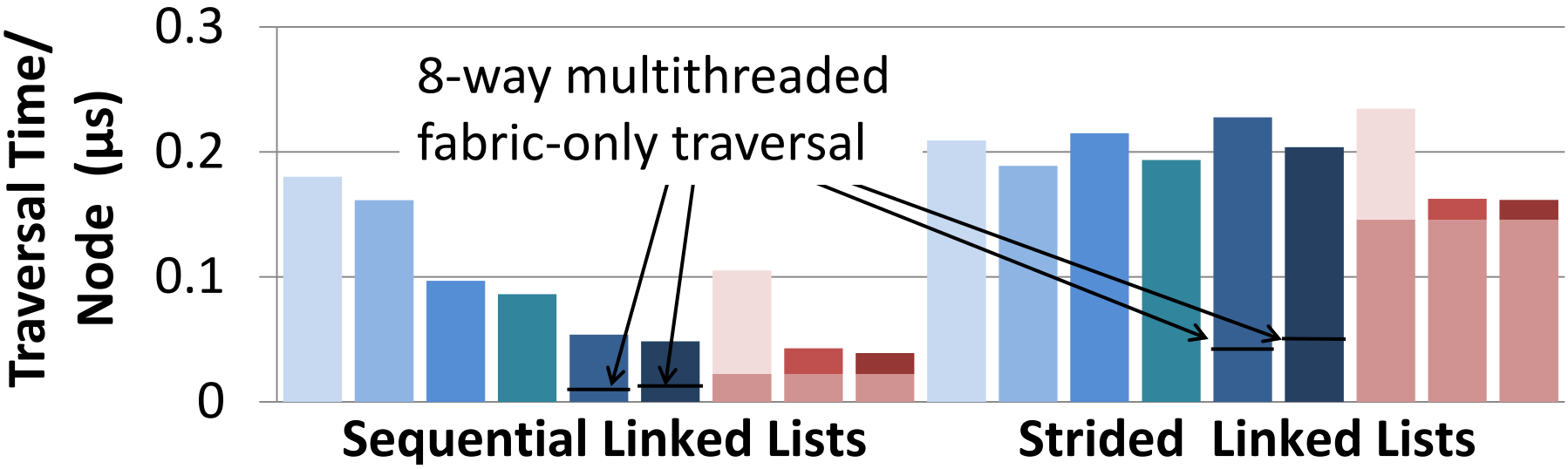


Buffer data in DRAM or On-Chip Memory when using DMA

8-bytes Nodes with Inline Payload



8-bytes Nodes with Inline Payload



Fabric-Only Traversal

8-Byte Transfers: HP, ACP

16-Byte Transfers: HP, ACP

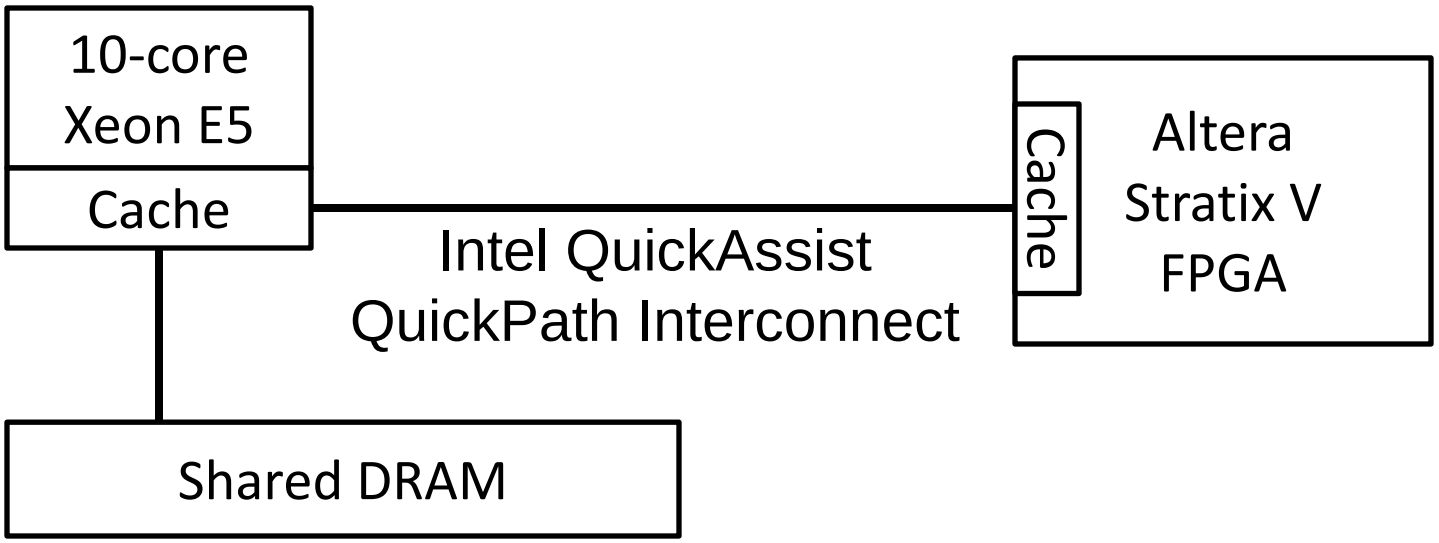
32-Byte Transfers: HP, ACP

ARM-Assisted Traversal

- Software Reference
- Hybrid-Push through AXI
- Hybrid-Push DRAM Buffer
- Hybrid-Push OCM Buffer

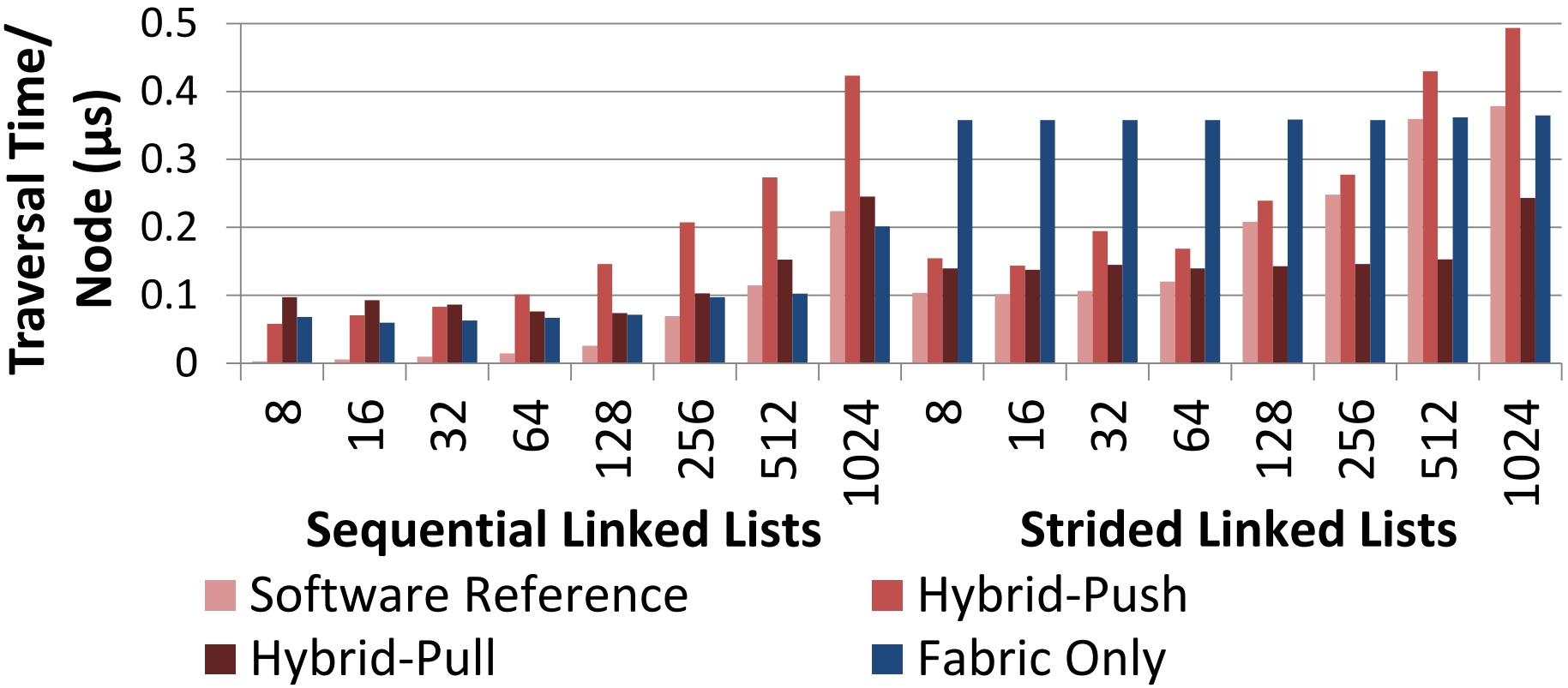
Fabric-only and hybrid-pull were most effective for lists with larger and indirect payload

Intel Heterogeneous Architecture Research Platform



Note: This is pre-production hardware

Indirect Traversals on Intel QPI



- Parallel fabric-only traversal engines improved per node performance on strided lists to ~1.9 µs
- Multi-threading hybrid traversals only improved performance slightly due to pthreads overhead

Conclusions

- Processors can accelerate pointer-chasing memory accesses while simplifying programming
- Parallel traversal engines can improve performance
- Wish list for future FPGAs:
  - Hard-logic memory controllers
  - Non-blocking writes from processors to fabric

CMU authors were supported by NSF CCF-1320725.

Thank you Altera, Xilinx, Intel, and Bluespec for hardware and tools donated to CMU