Kobold: Simplified Cache Coherence for Cache-Attached Accelerators

Carnegie Mellon University

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

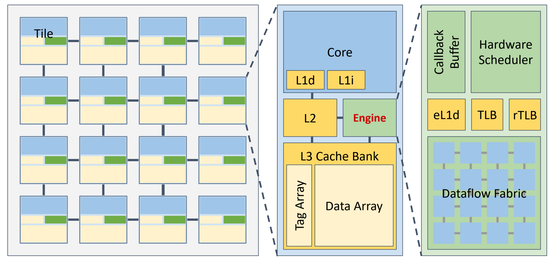
Jennifer Brana (University of Portland), Brian Schwedock (CMU), Yatin Manerkar (University of Michigan), Nathan Beckmann (CMU)

MICRO 2022
Undergraduate SRC

Motivation

Rising cost of data movement → Move compute to where data resides

Cache-attached accelerators move accelerators to ***within*** the cache hierarchy

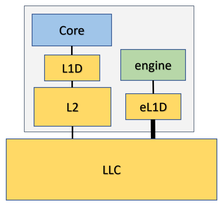


täkō [1] (above) is a representative system that augments each tile of a CMP with an engine and engine cache. This allows the engine to:

1. Accelerate key computations
2. Use low-latency & fine-grained communication with the processor

To fully benefit, the engine's cache must maintain coherence with the system

Naïve Design



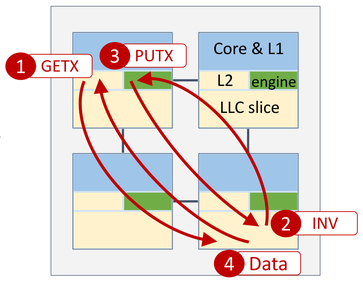
Treat eL1d as additional LLC sharer

eL1D uses baseline protocol

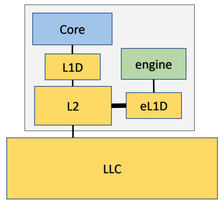
Results in excessive writebacks to the LLC

Core and engine must transfer data through the LLC

LLC banks are often far from cores → *wasteful & unnecessary data movement*



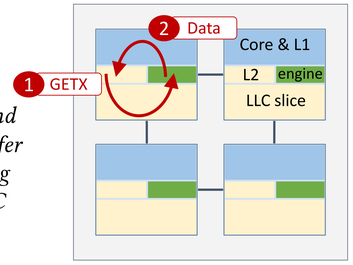
Kobold Design



Goal 1: reduce writebacks to the LLC

Attach the eL1D to the L2 cache → **Requires new coherence protocol**

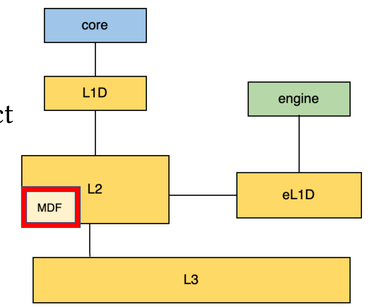
Now, the core and engine can transfer data w/o writing back to the LLC



Challenge: Verification of new coherence protocols can be extremely costly

Insight: restricting the complexity of the accelerator to ***within*** a tile allows the LLC protocol to remain unchanged

Goal 2: restrict complexity of engine to within a tile



Add directory to L2: Mis-direction Filter (MDF) tracks the state of the eL1D

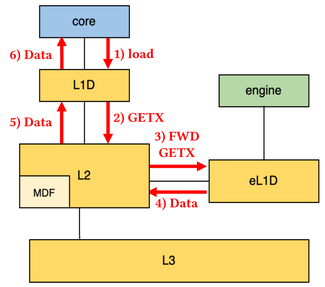
Intra-tile coherence is maintained using MDF & new intra-tile communication

→

Intra-tile locality enables fast, local communication

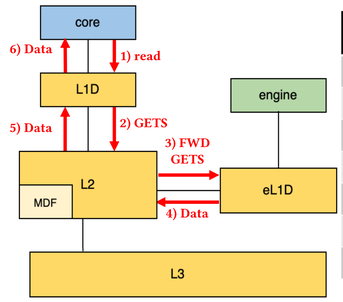
Coherence Protocols

1) Tile caches can transfer ownership without sending requests to the LLC



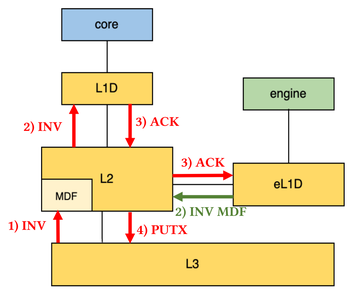
Step	L1D	eL1d	L2 / MDF	L3
Init)	I	M	I / M	M
1)	I	M	I / M	M
2)	I	M	I / M	M
3)	I	I	I / M	M
4)	I	I	M / I	M
5)	M	I	M / I	M
6)	M	I	M / I	M

2) All tile caches can share data that is tracked as exclusive in the LLC directory



Step	L1D	eL1d	L2 / MDF	L3
Init)	I	M	I/M	M
1)	I	M	I/M	M
2)	I	M	I/M	M
3)	I	S	I/M	M
4)	I	S	S/M	M
5)	S	S	S/M	M
6)	S	S	S/M	M

3) Caches coordinate responses to LLC requests so there is only one responder



Step	L1D	eL1 d	L2 / MDF	L3
Init)	S	S	S/M	M
1)	S	S	S/M	M
2)	I	I	M/I	M
3)	I	I	M/I	M
4)	I	I	I/I	M
5)	I	I	I/I	I

Kobold is not traditional hierarchal cache coherence *(HCC)*

Typical HCC adds inclusive intermediate caches to maintain cluster coherence, adding hierarchical indirection and increasing storage overhead

To avoid this, Kobold implements intra-tile coherence using the MDF and intra-tile communication, allowing the eL1D and L2 to maintain coherence between themselves and preserve baseline performance

Performance Considerations

Goal 3: prevent L2 cache pollution. → L2 cache is non-inclusive of the eL1D.

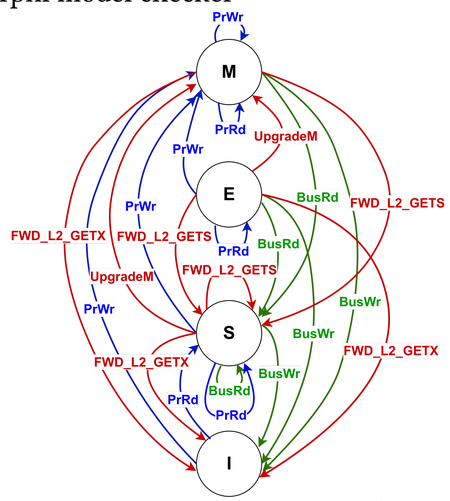
Optional eL1D optimization: with a minor modification to the LLC protocol, we allow speculative eL1D loads

Evaluation

We evaluate a system with a 128KB L2, 8KB eL1D, and 512KB LLC per tile

Estimated MDF overhead of only 0.09% of baseline (L2+LLC) area using CACTI

Verified stable state protocols using the Murphi model checker



Stable state protocol for eL1D cache controller. Red arrows represent new intra-tile messages.

Next Steps

Generate fully concurrent protocols using the HieraGen toolset

Implement and test in the täkō system

References
[1] täkō: A Polymorphic Cache Hierarchy for General-Purpose Optimization of Data Movement. B. Schwedock, et al. ISCA 2022.