Direct to Data caches

By *Ram Prasad, Kyle Sunden and Aarati Kakaraparthy*

Traditional caches

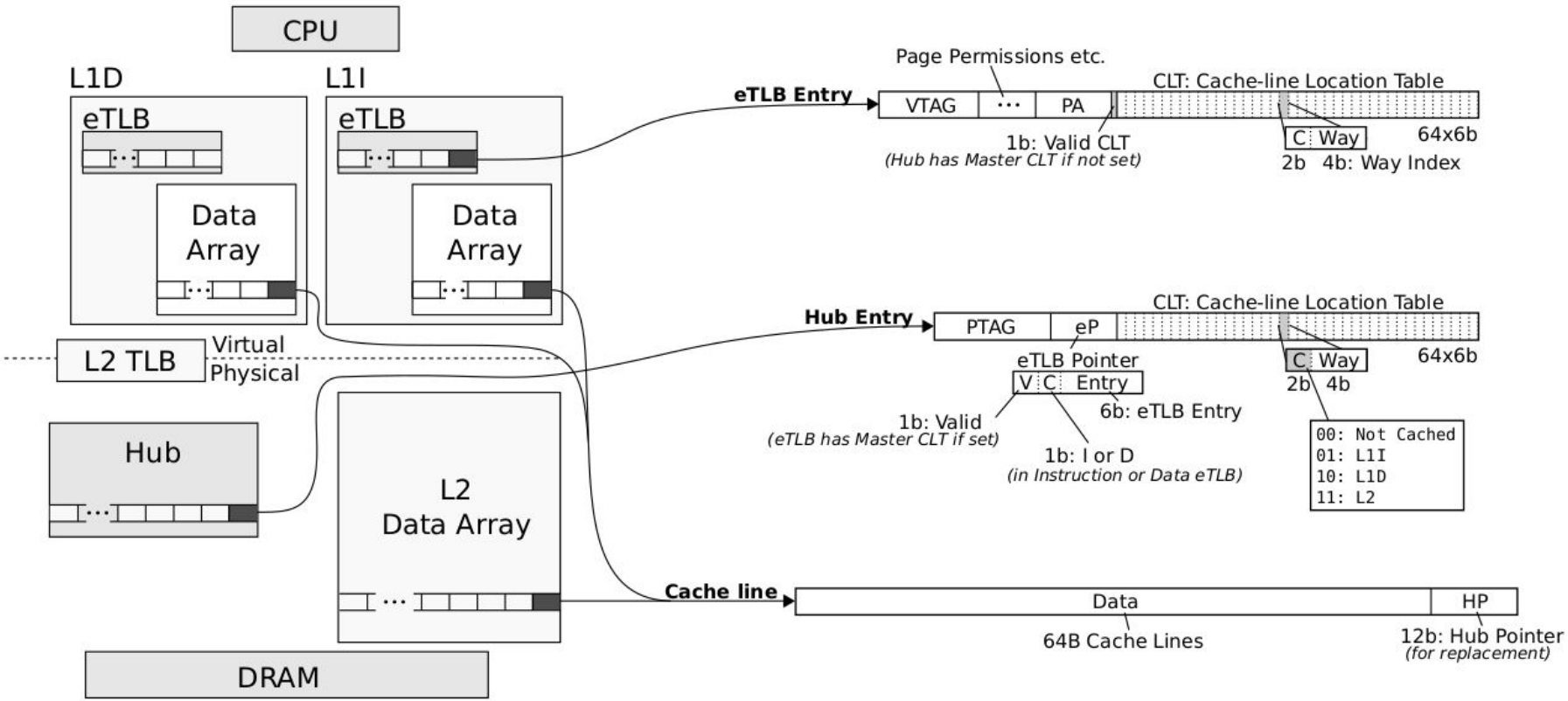
- Multiple cache hierarchy
- Traverses through various levels to find the data
- ***Involves tag check at each level***
- Consumes 12-45% of the core power

Motivation for D2D

- ***What if we can determine the cache level holding the data with a single access?***
- This can eliminate the need for checking tags at each level, and go to the correct cache directly.
- Saves power and cycle time!
- Micro-arch optimizations

The D2D Cache Architecture

Source - *The Direct-to-data cache: Navigating the Cache Hierarchy with a single lookup,* by Sembrant et. al.



Evaluation of D2D Cache Architecture

We have attempted the following two approaches for evaluating D2D caches:

- An implementation of D2D caches in Gem5
- An evaluation of D2D caches through memory traces

Tools used

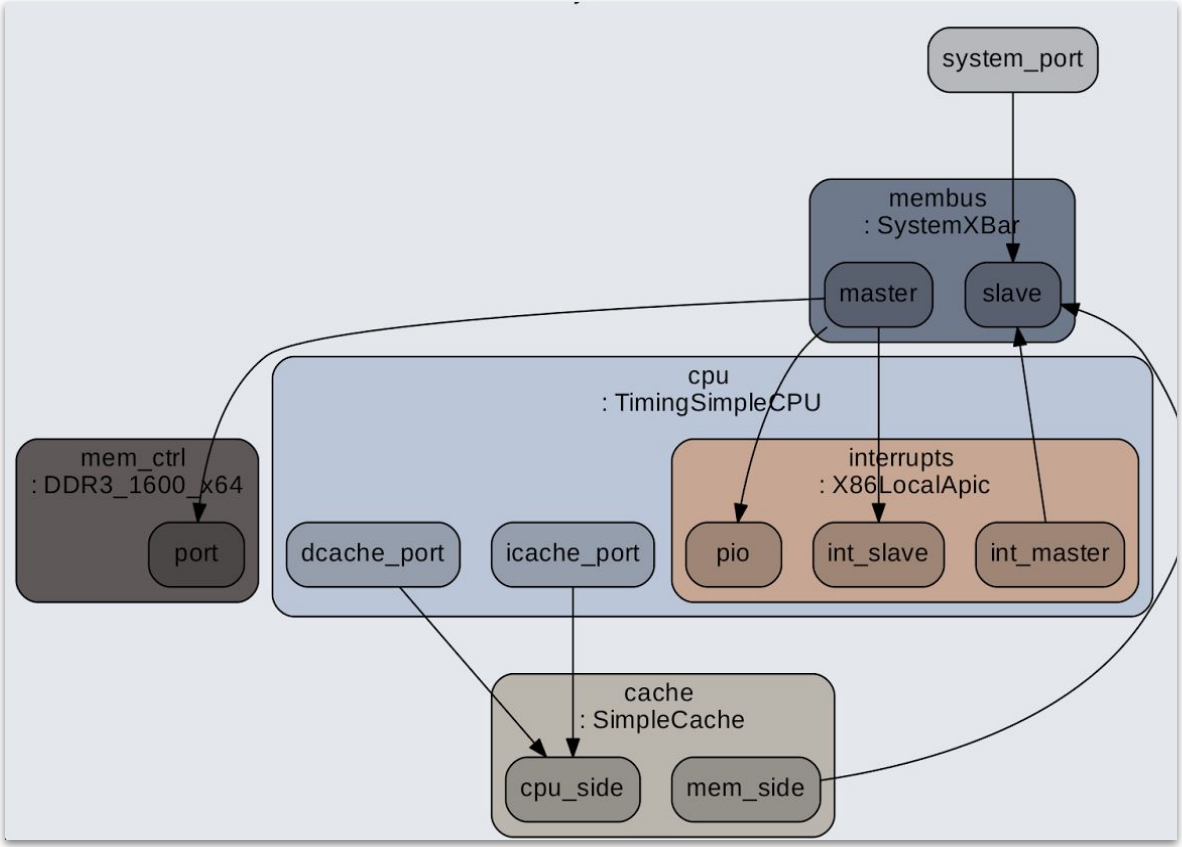
- Gem5
- Cacti
- Pintool

Modelling D2D in Gem5

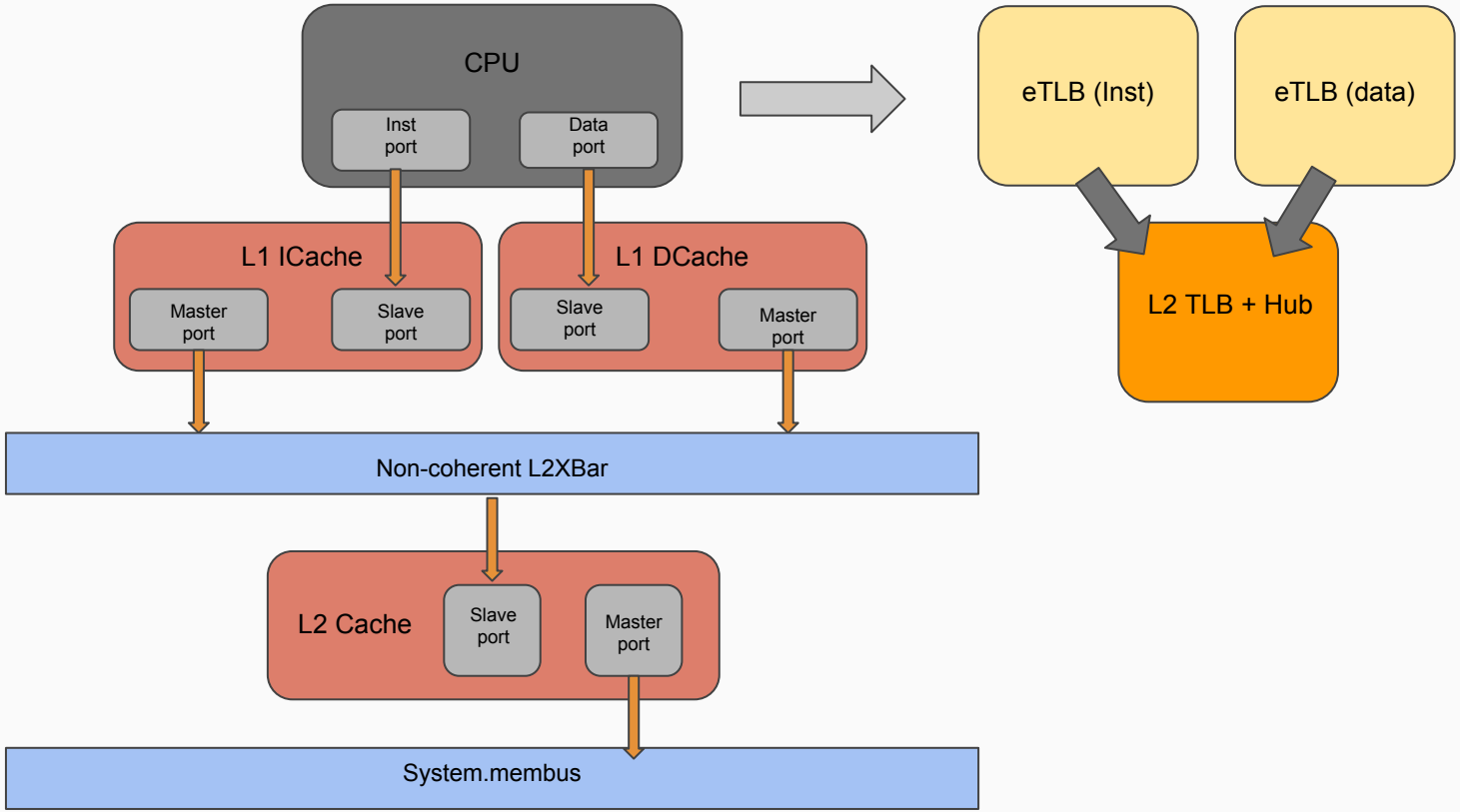
A brief on Gem5 Memory System

- Uses “Memory Objects” (derived classes of *MemObject* class), to model the memory system.
- Different memory objects are connected via ports.
- They communicate through transferring packets
- Event-driven programming

A Simple Gem5 configuration



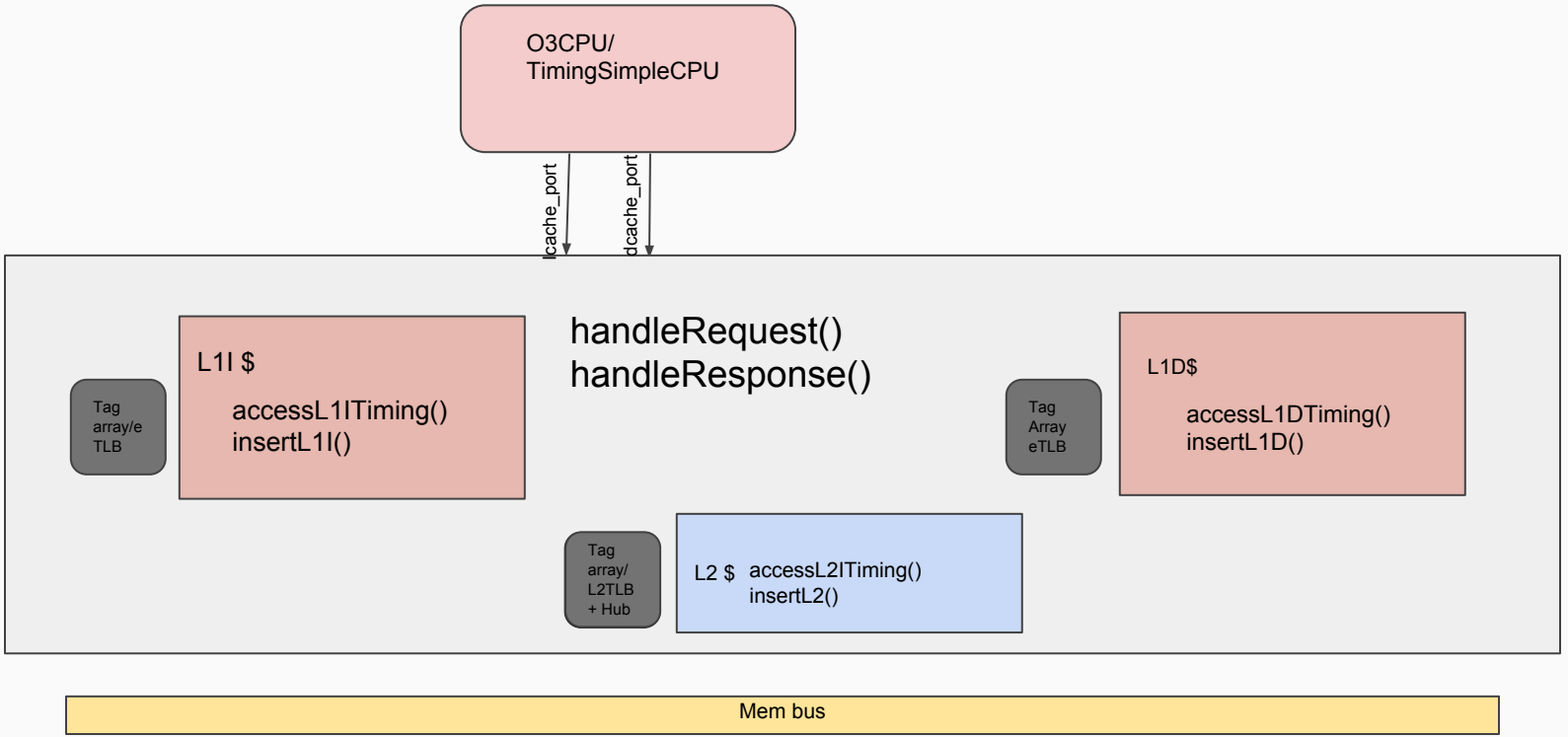
The Initial configuration attempted



Challenges

- The traditional Gem5 caches provided a “mostly exclusive” mode
- We needed to model *fully exclusive caches* for D2D
- Traditional caches also deal with cache coherency
- We have a single CPU in our desired configuration, so cache coherency is not essential

The Configuration Implemented (Base and D2D)



Benchmark results of the base configuration

Benchmark name	Total Energy spent(nJ)	L1I Hit rate	L1D hit rate	L2 hit rate
libquantum	5705767.03	0.989	0.765	0.997
gcc	11496797.27	0.909	0.503	0.992
hmmer	11692466.37	0.907	0.53	0.999
bzip2	14248942.71	0.932	0.51	0.997

Roadblocks

- Complexity of Gem5 traditional cache, for maintaining cache coherency
- Pages containing both instructions and data blocks
- Instruction and data boundary at block level

Evaluating D2D using memory traces

Memory trace Implementation

- Python cache implementation which only uses the memory addresses
- Allows for control over the memory subsystem, without needing the full system.
- “cache” system: Two Level Simple LRU Caches, no virtual to physical translation
- “etlb” system: Two level D2D Caches with eTLB and Hub, offset translation

Source: http://github.com/ksunden/cachesim

Input

```
R 0x93ff60
R 0x7f0ed33ac2f8
R 0x7f0ed33ac338
R 0x7f0ed33ac2f0
W 0x7ffc4e508548
R 0x7f0ed33ac460
R 0x93fea0
R 0x400298
W 0x7f0ed33ac4b4
R 0x4002a0
R 0x40029c
W 0x7f0ed33ac4b8
R 0x4002a4
```

GCC trace had 10^9 lines, was over 15 GB

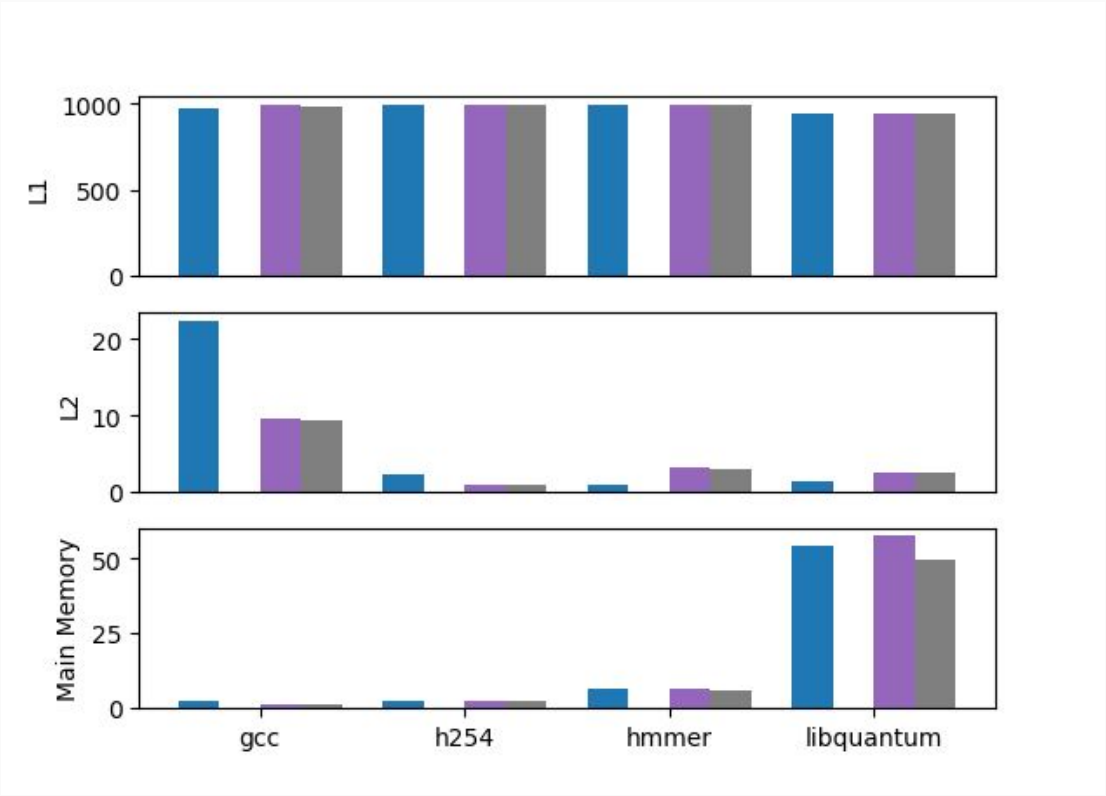
Other Sample programs had fewer lines (200k to 800k)

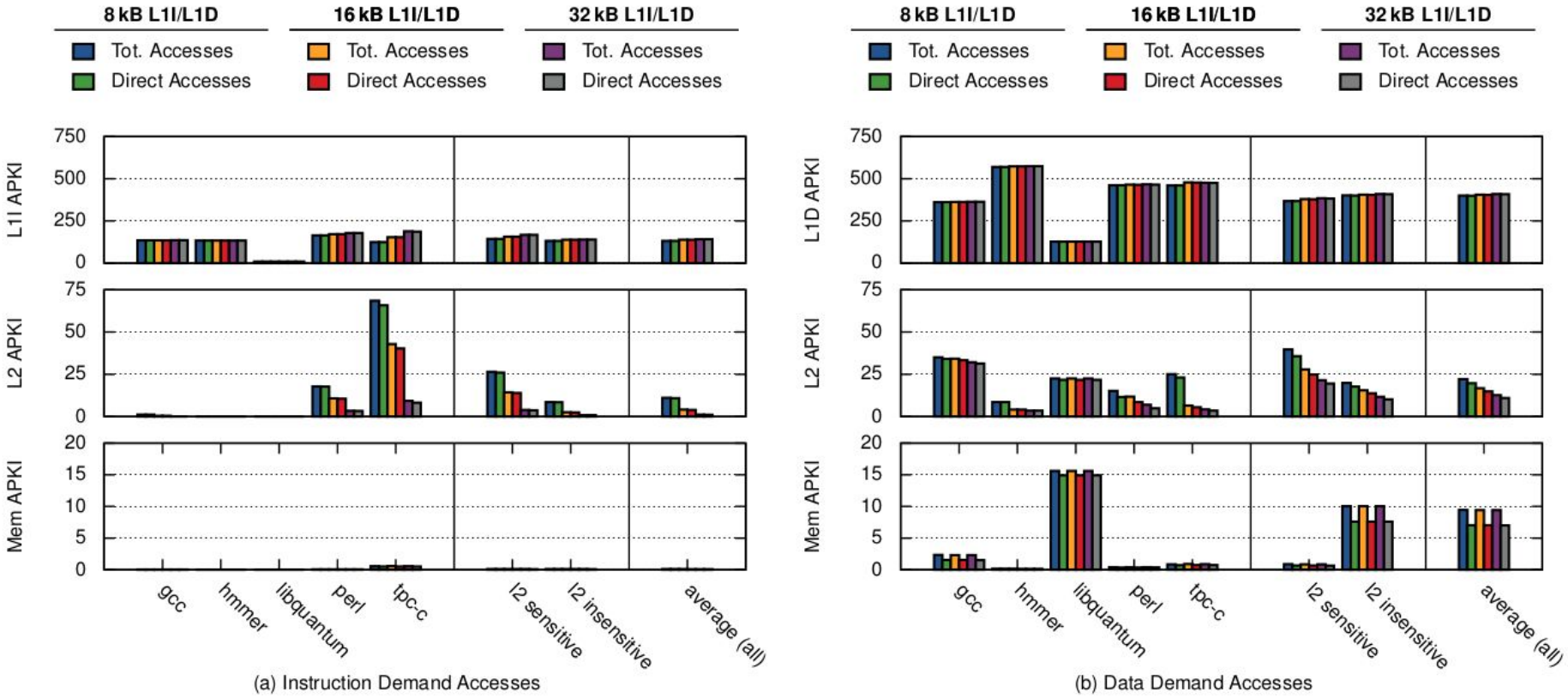
Output

```
N: 1083945089
L1 hit:  1057090378 (97.523)
L1 miss: 26854711 (2.477)
L2 hit:  24358138 (2.247)
L2 miss: 2496573 (0.230)
Time L1: 4335780356, L2: 295401821,
total: 4631182177
Energy L1: 17605873.417, L2:
12438378.831, total: 30044252.248
```

```
N: 1083945089
ETLB Hit, NIC 999191, (0.092181)
ETLB Hit, L1D 1068406787, (98.566505)
ETLB Hit, L2  10211154, (0.942036)
ETLB Miss, 4327957, (0.399278)
Hub Hit, NIC 106862, (0.009859)
Hub Hit, L1  3929152, (0.362486)
Hub Hit, L2  288923, (0.026655)
Hub Miss,  3020, (0.000279)
Time L1: 4273627148, L2: 141972236,
total: 4415599384
Energy L1: 12041636.129, L2:
4428224.749, total: 16469860.878
```

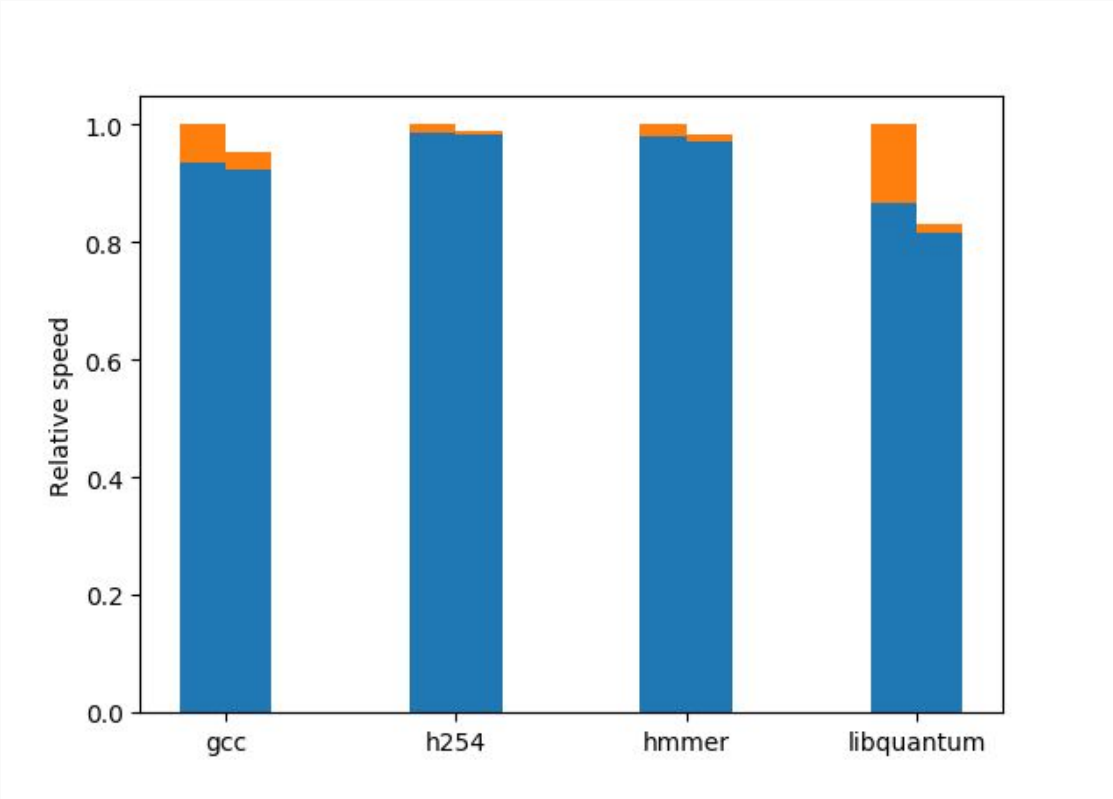
Memory Access Patterns

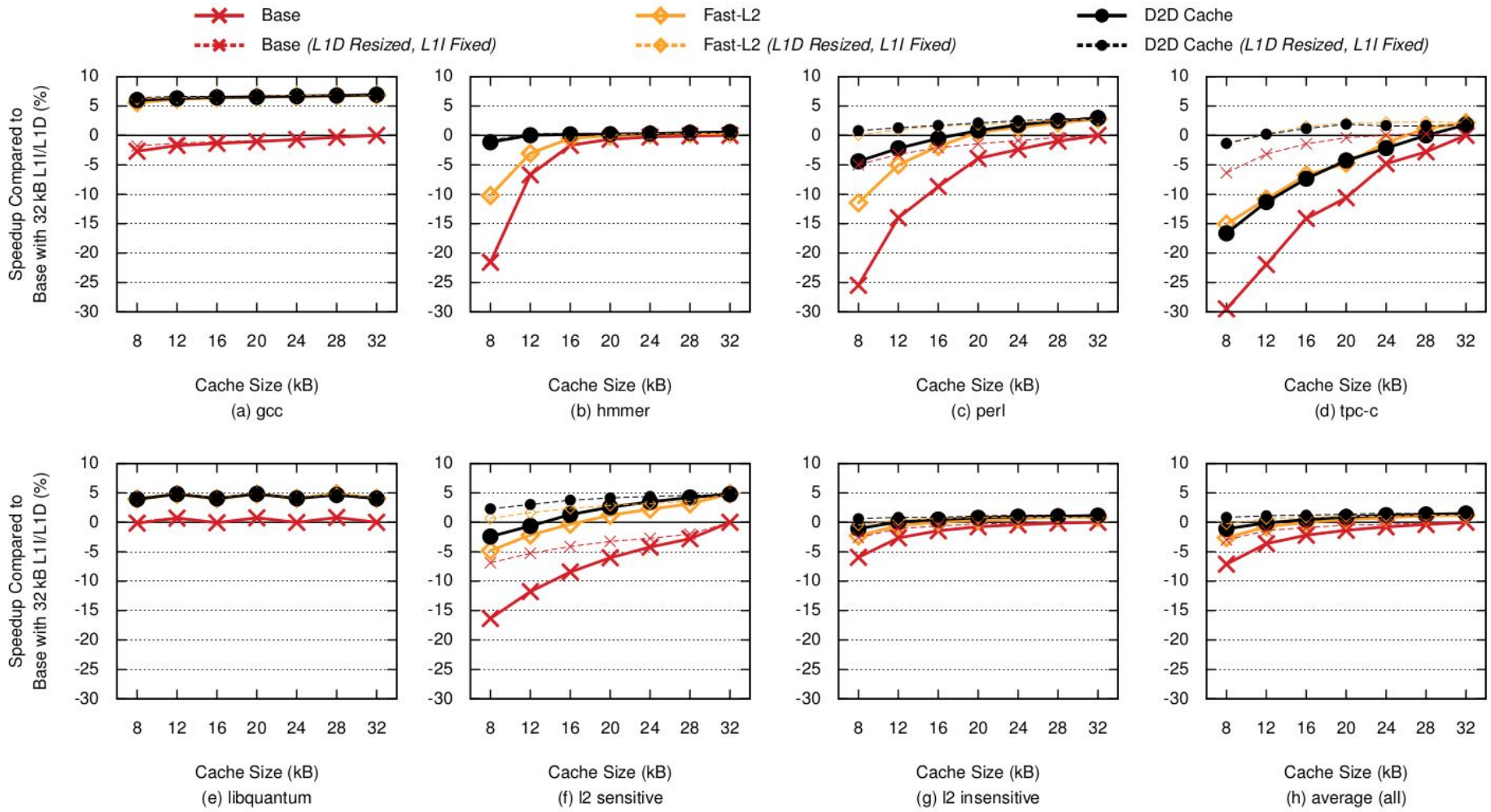




Source: *The Direct-to-Data (D2D) Cache: Navigating the Cache Hierarchy with a Single Lookup,* by Sembrant et. at.

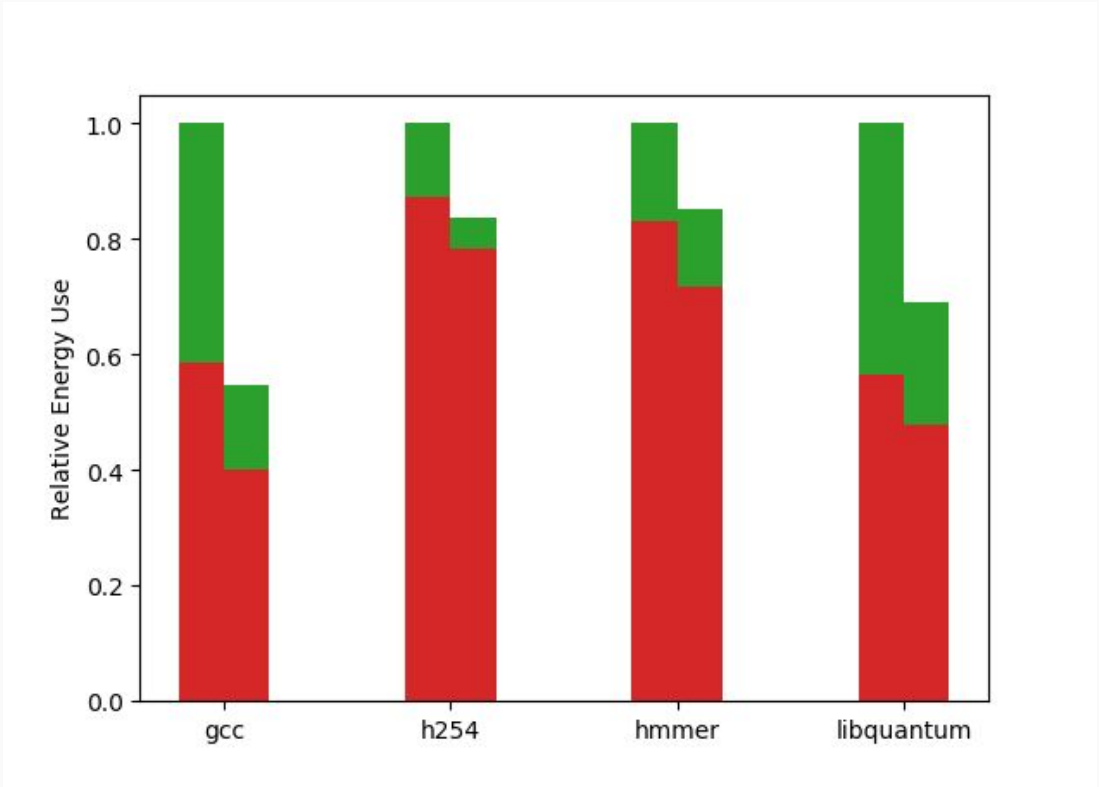
Speed Of Execution

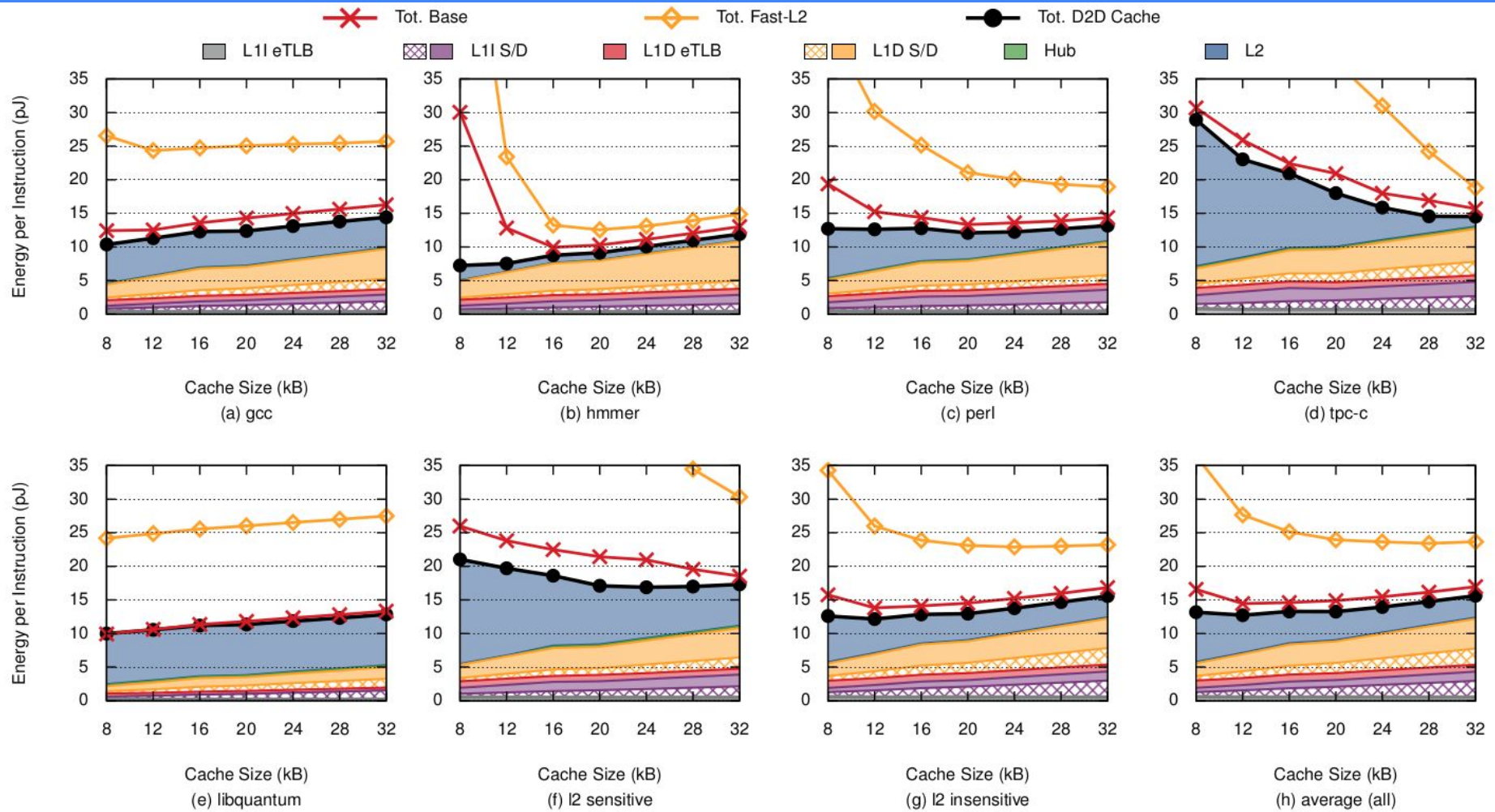




Source: *The Direct-to-Data (D2D) Cache: Navigating the Cache Hierarchy with a Single Lookup,* by Sembrant et. at.

Energy Usage





Source: *The Direct-to-Data (D2D) Cache: Navigating the Cache Hierarchy with a Single Lookup,* by Sembrant et. al

Conclusions

- Most accesses can be made directly (i.e. hit the ETLB)
- D2D Caches Provide modest memory speed improvements
  - Greater Improvement where L2 Access is important
- Significant Improvements in Energy usage
  - Challenging to account for all energy usage

Future work

- Completing the Gem5 implementation
- Adding the L1 stride prefetcher
- Cachesim improvements
  - Separate Instruction and Data Caches
  - Investigate effects of cache size changes (As in Sembrant et al)
  - Ensure energy usage is all counted
  - Support more cache structures

References

The Direct-to-Data (D2D) cache: navigating the cache hierarchy with a single lookup Sembrant,Hagersten, Black-Schaffer; ISCA '14; 10.1145/2678373.2665694

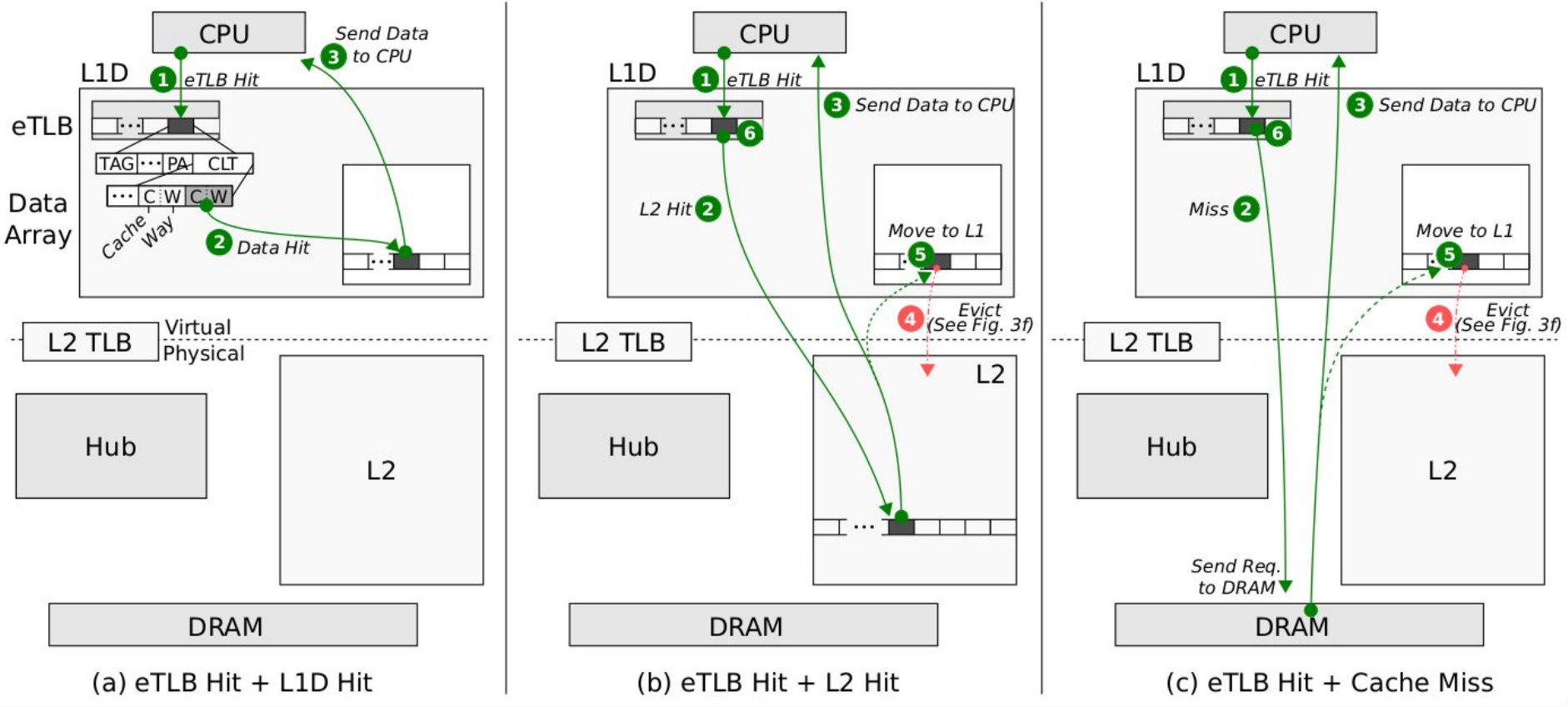
The gem5 Simulator. Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. May 2011, ACM SIGARCH Computer Architecture News.

CACTI 6.5: http://www.hpl.hp.com/research/cacti/

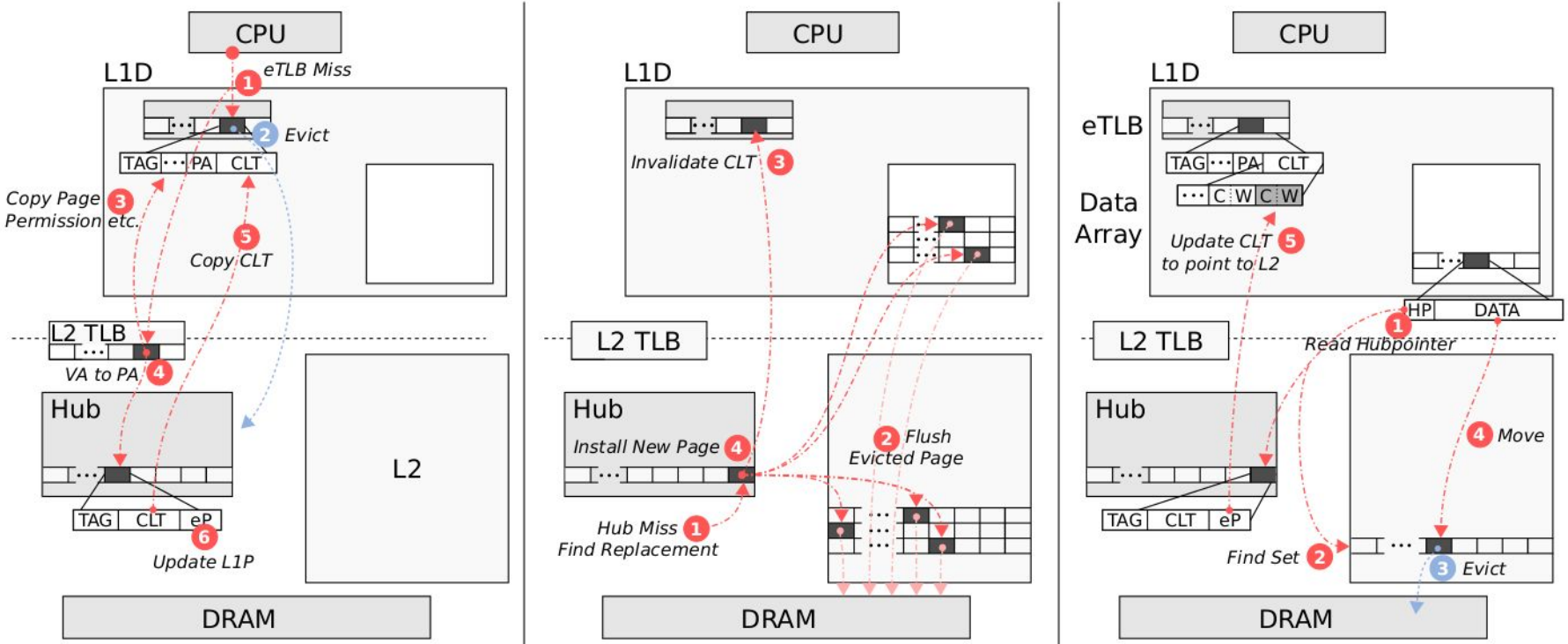
Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, Kim Hazelwood. "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," Programming Language Design and Implementation (PLDI), Chicago, IL, June 2005, pp. 190-200.

Questions?

Appendix



Source: *The Direct-to-Data (D2D) Cache: Navigating the Cache Hierarchy with a Single Lookup,* by Sembrant et. at.



Source: *The Direct-to-Data (D2D) Cache: Navigating the Cache Hierarchy with a Single Lookup,* by Sembrant et. at.

System Configuration from Sembrant et al.

	Base	Fast-L2	D2D
Frequency	2.5 GHz		
Fetch / ROB / IQ / LSQ / Regs.	3 / 32 / 32 / 16 / 128		
L1 Instruction Cache	32kB, 64B, LRU, 4 c		
- Associativity	8-way		16-way
L1 Data Cache	32kB, 64B, LRU, 4 c		
- Associativity	8-way		16-way
L2 Unified Cache	1MB, 64B, 16-way, LRU		
- Latency	+10c	+6c	+6c
L2 Prefetcher	12 stream stride prefetcher		
Memory	2GB LP-DDR3 12-12-12		
L1 TLB / eTLB	64 entries, 4kB Pages, 8-way, LRU		
L2 TLB	512 entries, LRU, 5 c		
Hub			4k, 1kB, 8-way