1. Problem: Traditional Caches

Traditional cache hierarchies waste time and energy by sequentially searching for data.

2. Solution: Go Directly to Data

D2D extends the TLB with per cache line level and way info. One lookup in the eTLB tells us where the data is located.

3. D2D: Accessing Data Directly

The cache-level bits in the eTLB entry indicate that the data is in the L1 cache. The L1 data entry is then accessed with the set index from the virtual address and the way index stored in the eTLB's L1 entry. (Note that the L1 cache is an array directly and 32-byte aligned. Each cache line is stored in a single array element.) The data in the correct cache line in the L2 is then read from the data in the eTLB entry. (Note that only a few bits of the TLB (8 bits) are needed to find the correct way.) Moreover, the physical address in the eTLB is not needed since the correct way is determined by the TLB entry. The L1 data array is read sequentially and the data is found. This wastes energy by probing levels that do not contain the data, and increases access latency for every level examined.

4. D2D: Extra Cache Management

A. eTLB Miss
   - Fetch CLT from Hub
B. Hub Eviction
   - Flush cache lines (no tags)
C. Cache Line Replacement
   - Move data and update eTLB

Off Critical Path

These events occur in the L1 cache. When the CPU requests data that is not in the L1 cache, the data is accessed directly using the eTLB. This allows some extra cache management for better and more energy-efficient cache access.

5. Summary

1. Skip levels in the cache hierarchy by determining the correct level from the TLB
2. Eliminate extra data-array reads by determining the correct way from the TLB
3. Eliminate the tag-arrays by avoiding tag comparisons

40% Lower L2 latency
21% Lower Cache Energy

6. eTLB Effectiveness

The cache-level bits in the eTLB entry indicate that the data is in any of the caches. D2D then reads the virtual address from the eTLB, and sends a memory request directly to the DRAM controller. The data is fetched from memory, and D2D sends the data to the CPU. The data is then stored into the eTLB, and the data is then sent to the CPU. The eTLB is updated to point to the new location in the L1 cache.

7. Use Case 1: L1 Cache Size

Trade off D2D's lower L2 latency for energy by using a smaller L1 cache size (without hurting performance).

8. Use Case 2: ROB Size

D2D is effective at accessing the data level. D2D reduces the overall L2, L1, L2 and L1 ROB sizes and increases the D2D performance.

25% Smaller ROB