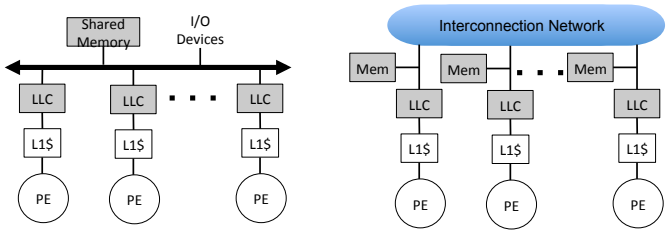
CS6560: Parallel Computer Architecture

Shared Memory Multicore Systems

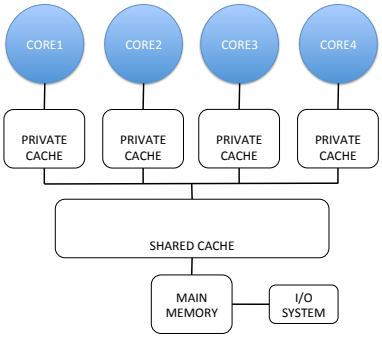
Madhu Mutyam
PACE Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Feb 21-Feb 28, 2018

Shared Memory Multiprocessor Systems



Multicore Caches

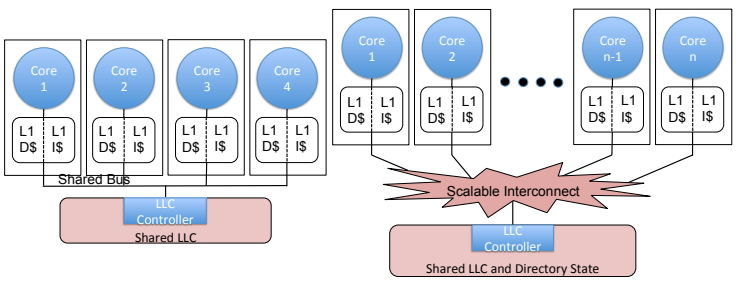


- Reduce the average access time and bandwidth requirement
- Multiple processors can perform load/store operations simultaneously

Managing Large Caches in Multicore

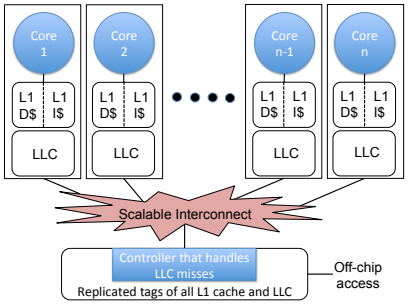
- Shared vs. Private
- Centralized vs. distributed
- Uniform access vs. non-uniform access

Shared Last-Level Cache (LLC) in Multicore



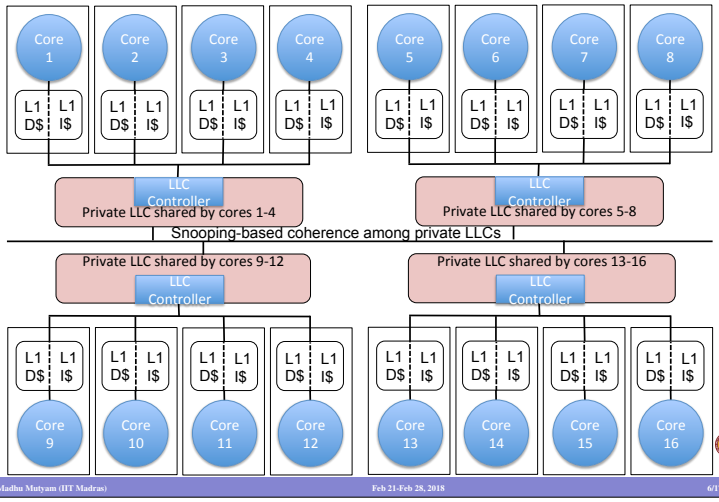
- Dynamic allocation of space among threads/cores
- No replication of shared blocks
- Core-to-core interference
- High contention when accessing shared resources
- High average hit latency

Private LLC in Multicore

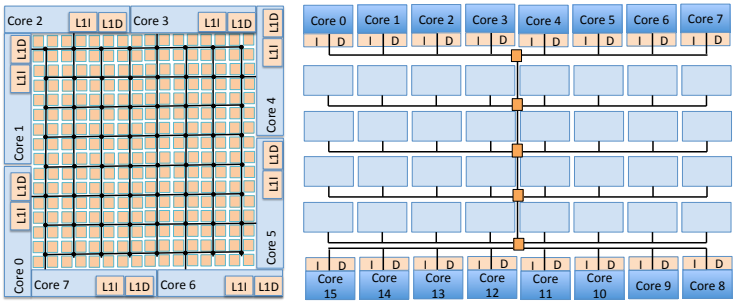


- No core-to-core interference
- No contention when accessing L2 cache
- Low average hit latency
- Replication of shared blocks

Shared-Private LLC in Multicore

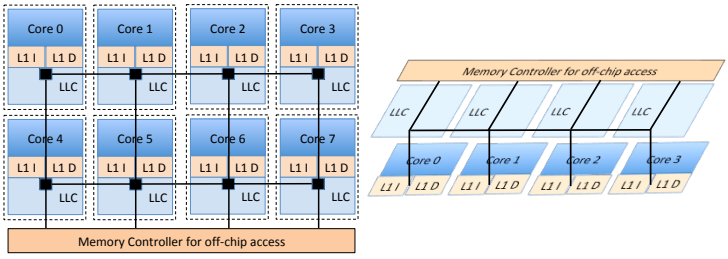


Centralized Shared LLC in Multicore



- Shared LLC is multi-banked and the functionality of cache controller is replicated in each bank
- Not a scalable design

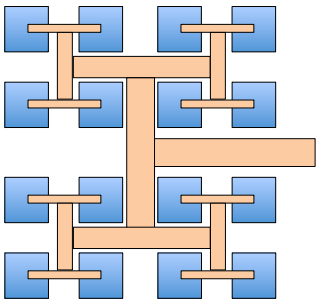
Distributed Shared LLC in Multicore



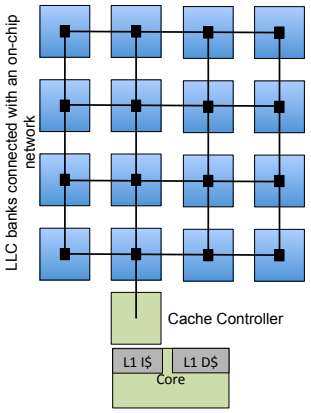
- Tile-based organization provides better scalability and reduced verification cost
- High cost in moving data between L2 cache banks and next level of memory hierarchy

Non-Uniform Cache Access

- *Uniform cache access* – worst-case delay is considered
- In UCA banked cache design, H-tree topology can be used



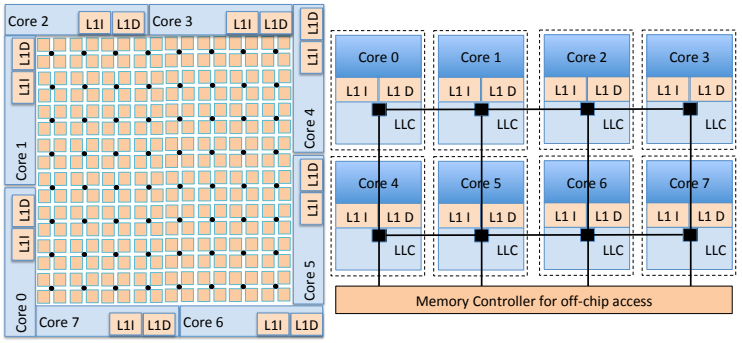
- In NUCA banked cache design, NoC based mechanism is used



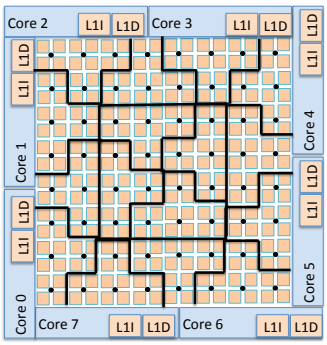
Data Management in Shared NUCA Caches

- Three key issues:
  - *Mapping*: the possible locations for a data block
  - *Searching*: the mechanisms to locate a data block
  - *Movement*: the mechanisms to change the location of a block
- Two variations of NUCA caches:
  - *Static-NUCA* (S-NUCA)
  - *Dynamic-NUCA* (D-NUCA)
- One open problem:
  - Efficient search in a D-NUCA cache

Placement/Migration/Search Policies for D-NUCA



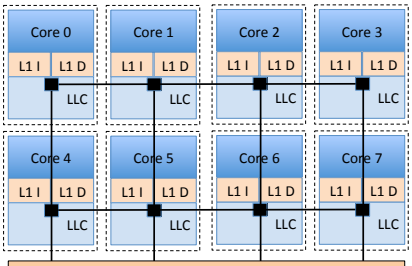
Placement/Migration/Search Policies for D-NUCA



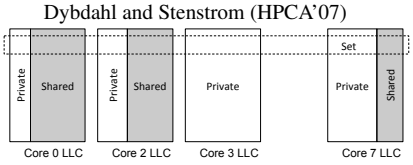
Beckmann and Wood (MICRO'04)

- 256 cache banks into 16 regions
- Local, inter, and center regions
- Each region contains a way for all sets (16-way set-associative cache)
- A block, placed anywhere initially, can be migrated gradually to the local region
- 2-level multicast to search a block
  - Phase #1: local, inter, and center regions
  - Phase #2: remaining 10 regions
- Searching is very expensive

Placement/Migration/Search Policies for D-NUCA (Contd)

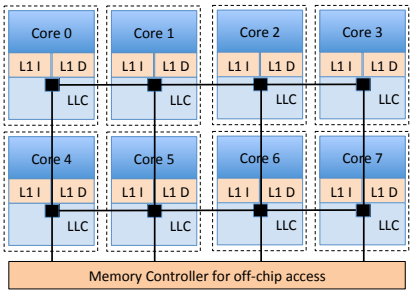


Dybdahl and Stenstrom (HPCA'07)



- The ways of a set are distributed across the tiles
- Some ways in each tile are private to the local core
- Private ways of the core are searched first, followed by all the other ways
- A block evicted from a private way is moved to a shared way
- Private ways are allocated based on run-time statistics
- Replacement policy should aware of private and shared ways

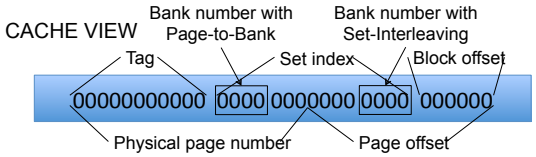
Replication Policies for S-NUCA



Zhang and Asanovic (ISCA'05)

- Each cache block maps to a unique L2 bank
- Evicting a block from the L1 cache of a core
  - Place it in the local L2 bank
  - No change to the sharers list in the coherence directory
- A subsequent write to the block by any core invalidates the replicated copy
- On a miss in the L1 cache, the local L2 bank is searched
- An extra L2 bank lookup on every invalidation

OS-Based Page Placement



Cho and Jin (MICRO'06)

- The bank number bits that represent a subset of the page number bits can be referred to as the *page color*
- When a new page is requested, the OS can determine the optimal bank for that page and then select a free page from the appropriate color list
- *First-touch* page color policy combined with *page-spreading* policy is effective at balancing latency and capacity needs of **private pages**
- Not effective for shared pages or threads that migrate

OS-Based Page Placement for Multi-Threaded Applications

- Awasthi *et. al.* (HPCA'09): *First-touch* + *page-spreading* + *page migration* in L2 banks
- In order to migrate a page from one L2 bank to another, map it to a new physical address
- Shadow address region is used for renaming page addresses
- Reflect the change of physical address only in the TLB, but not in DRAM
- For effectiveness, banks can be classified as *acceptors* (need more cache space) and *donors* (can spare some cache space to other programs) based on cache usage
- Large TLBs are required
- Every page migration requires a flush of the original page's contents from cache as well as TLBs

Thank You