Multi-Core Memory Hierarchies
Lecture 3 : NUCA

Tong Dong

http://mprc.pku.edu.cn/~tongdong/MMH
In last lecture

**Definitions and Policies**

- **MIP**: MRU Insertion Policy (traditional approach)
- **DIP**: Selects the best
- **LIP**: LRU Insertion Policy
- **TADIP**: Selects the best for each thread
- **BIP**: Bimodal Insertion Policy
  - Few insertions at head, most at tail
- **RRIP**: Probabilistic insertion near tail
- **PIPP**: Inserts each thread at different positions + probabilistic promotion
- **AGGRESSOR-VT**: Victimizes the aggressor thread with a high probability
- **UCP**: Partitions ways across threads based on marginal utility

**Priority Stack**

Highest priority ← Priority stack of blocks in a set → Lowest priority
Model of cache performance

- For N cores and T thread per core, the miss ratio of the $i$-level cache:

$$m_{Li} = \left( \frac{C_{Li}}{ln(NT)\beta_{Li}} \right)^{1-\alpha_{Li}}$$

- $\alpha$, $\beta$: locality function constant
- $ln(NT)$: interference function
- $C$: cache size in level $i$

- Average Access Time in $i$-level cache:

$$T_{Li} = (1 - m_{Li})T_{Li}^{hit} + m_{Li}T_{Li-1}$$
表 1: Cacti 工具 32nm Cache 各种配置对比结果

<table>
<thead>
<tr>
<th></th>
<th>32KB 4-way 64B Line</th>
<th>32KB 8-way 64B Line</th>
<th>64KB 4-way 64B Line</th>
<th>64KB 8-way 64B Line</th>
<th>128KB 4-way 64B Line</th>
<th>128KB 4-way 32B Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>访问时间 ns</td>
<td>0.6329</td>
<td>0.9790</td>
<td>0.6718</td>
<td>1.0086</td>
<td>0.6920</td>
<td>0.5390</td>
</tr>
<tr>
<td>每次读能量 nJ</td>
<td>0.0816</td>
<td>0.1730</td>
<td>0.1426</td>
<td>0.1779</td>
<td>0.1515</td>
<td>0.0423</td>
</tr>
<tr>
<td>每次读功耗 W</td>
<td>0.3564</td>
<td>0.7947</td>
<td>0.6139</td>
<td>0.7859</td>
<td>0.5871</td>
<td>0.1344</td>
</tr>
<tr>
<td>静态功耗 W</td>
<td>0.0117</td>
<td>0.0196</td>
<td>0.0233</td>
<td>0.0263</td>
<td>0.0377</td>
<td>0.0294</td>
</tr>
<tr>
<td>面积 mm^2</td>
<td>0.3613</td>
<td>0.6415</td>
<td>0.6060</td>
<td>0.7042</td>
<td>0.7500</td>
<td>0.3283</td>
</tr>
</tbody>
</table>
NUCA: Non-Uniform Cache Access

- How to build a large cache for single-core?
- Challenge: Growing wire delay
- A Highly-banked Cache
- Logic Policies
  - Placement/Mapping: the possible location for a data block
  - Search: the mechanisms required to locate a data block.
  - Migration/Movement: the mechanisms required to change a blocks location.

C. Kim and D. Burger, et al. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. ASPLOS-10’02. (804)
Static NUCA, S-NUCA

- Mapping policy
  - Distributing the sets of the cache across banks
  - Co-locating all ways of a set in one bank.

- Search policy: no need
  - Cache index bits in address are enough to locate bank.

- Movement policy: no need
  - Static
S-NUCA vs. D-NUCA

Figure 2: UCA and S-NUCA-1 cache design

(c) S-NUCA-1
32 banks
34 cycles

(d) S-NUCA-2
32 banks
24 cycles

(e) D-NUCA
256 banks
18 cycles
D-NUCA: Mapping policy

(a) Simple Mapping

(b) Fair Mapping

(c) Shared Mapping
D-NUCA achieves 1.5 times the IPC of UCA; outperforms the best static-NUCA scheme by 11%; comes within 13% of an ideal solution
More topics in D-NUCA research

- Placement/Migration policies
  - Dynamically place data close to cores that use it, reducing access latency

- Replication Policies
  - Makes multiple copies of frequently used lines, reducing latency for widely read-shared lines (e.g., hot code), at the expense of some capacity loss.

- OS-Based Page Placement
  - Page Coloring

- Data Management in Private LLC
NUCA LLC: Multi-cores

- Multi-cores will have a large last-level cache
  - Shared vs. Private
  - Centralized vs. Distributed
  - Inclusive vs. Exclusive vs. Non-inclusive

- LLC metrics
  - Miss rate
  - Access time: $\text{Hit Time} = T_{\text{search}} + T_{\text{bank hit time}} + T_{\text{transfer}}$
  - System throughput, bandwidth
  - Fairness
  - QoS
Shared NUCA

Advantages
- No replicated shared blocks (high effective capacity)
- Dynamics allocation of space (high effective capacity)
- Quick traversal through coherence interface (low latency for shared data)
- No tag replication for directory (low area requirement)

Disadvantages
- High Interference between threads (QoS)
- Long wire traversals to detect hit (high hit latency)
- High contention on LLC and NoC (high hit latency for private data)
Multi-Core NUCA: Search Policies

- Challenges:
  - Access Request injected from multiple locations.
  - Higher bandwidth requirement.
  - The on-chip network overheads

- Contributions:
  - Bank regions (one way all set)
  - Block migration between regions

Beckmann and Wood, MICRO’04

- Shared data locate in central regions
- Private data locate in all regions
- Placement: random first
- Migration
  - other local -> other inter -> other center -> my center -> my inter -> my local
- Search: Multicast 6 region first; Multicast 10 region if miss
- Result: no benefits to S-NUCA (40% in central)
- Limitation: lack of robust search mechanism
- Solution: prefetching
S-NUCA: long delay
D-NUCA: search overhead
Contributions:
- Replicated partial tags
- Sharing degree (SD)
- Coherent directory
- Dynamic scheme
Shared vs. Private cache: Hybrid design
- S-NUCA with 2- or 4- sharing provides the best on high performance and low complexity
Lin et al. HPCA’04. OS-based NUCA

NuRAPID and CMP-NuRAPID

- NuRAPID, MICRO’03
  - Centralized entire tag array
  - Sequential tag-data access
  - Decoupling tag and data block placement (pointer)
  - Distance associativity

- CMP-NuRAPID, ISCA’05
CMP-NuRAPID, ISCA’05

- Private L2 caches tag
  - Large private tag array

- Shared L2 cache data: Single Copy
  - Multiple copies in private LLC
  - Frequent L2 coherent misses in private LLC

- Statically partition across cores
  - Capacity Stealing: Large tag array
  - Reverse pointer in data block
  - Controlled replication
Disadvantage of D-NUCA

- Blocks are allowed to move arbitrarily and have no fixed home in the data array
  - Search mechanism: multicast-based method
  - Entire Tag: pointer-base method

- Bloom filter

- S-NUCA with Page Coloring
  - Combining the best properties of S-NUCA and D-NUCA
Replication in shared NUCA

- The problems in S-NUCA and shared D-NUCA
  - Shared block may not be close proximity to cores accessing it
  - Long access latency
  - Maintain multiple copies of a shared block

- Replicas coherence

- Need to combine the favorable properties of shared and private caches
Victim Replication, ISCA’05

- **Victim Cache**
  - Replica: evicted block placed to local L2 bank
  - Simple coherence mechanism

M. Zhang and K. Asanovic. Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. ISCA-32’05. (416)
ASR: Adaptive Selective Replication

- Static replication rule in Victim Replication degrade performance.
- ASR: A dynamic probabilistic system to select the appropriate level of replication
  - Cache replication policies should focus on shared read-only blocks
    - 42~70% of requests
    - 10~21% of capacity
  - Selective Probabilistic Replication (SPR)
  - Improving performance by 12% versus CMP-NuRAPID and Victim Replication.
OS page coloring policy in shared NUCA

- Page Coloring in NUMA
- 16 MB 8-way L2 Cache partitioning into 16 banks
  - Set-interleaving mapping
  - Page-to-Bank mapping

OS page coloring policy, Cho and Jin, MICRO’06

- NUMA: first-touch policy
- Cho and Jin, MICRO’06
  - Page-spreading policy in S-NUCA
  - Map subsequent pages to neighboring bank
- Awasthi, HPCA’09: Memory Copy/Migration
Reactive NUCA (R-NUCA), ISCA’09

- S-NUCA with OS-based page coloring
  - Combining D-NUCA (performance potentials) and S-NUCA (complexity)
  - Key problem: migration overhead

Motivation

- On-chip communication delay favors Private Cache
- Large working size favors Shared Cache
- The cache access patterns of server and scientific workload can be classified into distinct classes
- Each class is amenable to different block placement policies
Reactive NUCA (R-NUCA), ISCA’09

- Page categories:
  - Shared instruction pages ($I = 4$)
  - Private data pages ($P = 1$)
  - Shared data pages ($S = 16$)
Rotational interleaving
- Fixed-center cluster of tiles, logically surround a core
- Rotational ID (RID): consecutive column tiles
  - \( \text{RID} = \text{RID}_{\text{up}} + \log_2(n) \)
- Boolean indexing function: not (A[n-1:0] – RID[n-1:0])
  - Center: 00; Right: 01, Up: 10, Left: 11
- Each slice stores exactly the same \(1/n\)-th of any cluster to which it belongs.

Placement in OS

Page Classification
- CID in TLB
- Private->poisoned->shared
Disadvantage of private LLC
- Poor hit rate because of fixed size of private cache
- Poor hit rate because of data replication
- Overhead because of coherence look-up

Cooperative Caching, ISCA’06
- Cache-to-cache sharing
- Evicting replicated blocks
  - N% probability to evict replicated blocks
- Spilling of evicted blocks
  - M% probability to spill to a sibling private cache
- Challenge: complex central directory structures
Cooperative Caching, ISCA’06

Figure 3. CCE and Directory Memory Structure (8-core CMP with 4-way associative L2 caches)
ASR: Adaptive Selective Replication

- Static replication rule in Victim Replication degrade performance.
- ASR: A dynamic probabilistic system to select the appropriate level of replication
  - Cache replication policies should focus on shared read-only blocks
    - 42~70% of requests
    - 10~21% of capacity
  - Selective Probabilistic Replication (SPR)
  - Improving performance by 12% versus CMP-NuRAPID and Victim Replication.
# Characterizing CMP Working Sets

## Table 1: L2 Cache Request and L2 Cache Capacity Profile

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Single Requestor</th>
<th>Shared Read-Only</th>
<th>Shared Read-Write</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% of Requests</td>
<td>% of Capacity</td>
<td>% of Requests</td>
</tr>
<tr>
<td>apache</td>
<td>11%</td>
<td>51%</td>
<td>44%</td>
</tr>
<tr>
<td>jbb</td>
<td>57</td>
<td>91</td>
<td>42</td>
</tr>
<tr>
<td>oltp</td>
<td>4</td>
<td>51</td>
<td>71</td>
</tr>
<tr>
<td>zeus</td>
<td>20</td>
<td>71</td>
<td>54</td>
</tr>
<tr>
<td>apsi</td>
<td>&gt; 99</td>
<td>&gt; 99</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>art</td>
<td>53</td>
<td>71</td>
<td>46</td>
</tr>
<tr>
<td>barnes</td>
<td>19</td>
<td>93</td>
<td>74</td>
</tr>
<tr>
<td>ocean</td>
<td>94</td>
<td>98</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Figure 4: a) Replication Benefit**

**Figure 4: b) Replication Cost**

**Figure 4: c) Replication Effectiveness**
SPR: Selective Probabilistic Replication

- Benefit of Increasing Replication \((H_C - H_H)\)
  - Next Level Hit Buffers (NLHBs)

- Cost of Increasing Replication \((M_H - M_C)\)

- Benefit of Decreasing Replication \((M_C - M_L)\)
  - Victim Tag Buffers (VTBs)

- Cost of Decreasing Replication \((H_L - H_C)\)

Definitions:

\[
\begin{align*}
\Delta \text{Increase} &= (H_C - H_H) - (M_H - M_C) \\
\Delta \text{Decrease} &= (M_C - M_L) - (H_L - H_C)
\end{align*}
\]
Each private cache designated
- Spiller or Receiver
- Only allow spilling from a spiller to a receiver

Estimation based on Set Dueling
- Always-spill and always-receive sets
- A single counter per cache
- Dynamically select better policy

Reconfigurable (Partitioned) private LLC
- CloudCache, HPCA’11
- StimulusCache, HPCA’10
- MorphCache, HPCA’11
Scale-Out Processor, ISCA-39'12

Partitioned shared NUCA

Jigsaw lets software combine multiple bank partitions into a logical, software-defined cache.

Figure 1. Jigsaw overview: target tiled CMP, tile configuration with microarchitectural changes and additions introduced by Jigsaw, and Share-Bank Translation Buffer (STB).
**Challenge:** LLC consumes 17% energy; data movement over wires consumes 90% energy in a 2MB LLC.

**SLIP (Sub-Level Insertion Policy):** Reducing Wire Energy in Cache Hierarchies.

---

Subhasis Das, Tor M. Aamodt, and William J. Dally. SLIP: reducing wire energy in the memory hierarchy. ISCA-42’15. (13)
Jenga: Software-Defined Cache Hierarchies

Po-An Tsai, N Beckmann, D. Sanchez, ISCA-44 2017

Figure 2: The best cache hierarchy (by EDP) for SPEC CPU2006.

Figure 7: 16-tile Jenga system with distributed on-chip SRAM banks and four DRAM banks (e.g., stacked DRAM vaults).

Figure 9: The virtual hierarchy table (VHT) maps addresses to their unique VL1 and (if present) VL2 locations, storing the bank ids and partition ids (not shown).
Scalable Directory

- Requirements
  - Small area, energy, and latency overheads
  - Accurate sharer information
  - Limited directory-induced invalidations

- Duplicate Tags
  - Area-efficient
  - High associativity -> high power

- Sparse Directory
  - Power-efficient
  - Large capacity -> large area

- Coarse-grain vectors, Hierarchical, etc.
Scalable Directory Solutions

- Tagless Directory, MICRO’09 (140)
  - Bloom Filter
  - SPATL, PACT’11

- Cuckoo Directory, HPCA’11 (112)
  - Sparse Directory
    - Sharer bit vector
  - Multiple Hash Tables
SCD: Scalable Coherence Directory
D. Sanchez, C. Kozyrakis, HPCA’12

- For 1000 cores
- Efficient highly-associative caches
  - ZCache and Cuckoo Directory
- A variable number of directory tags to represent sharer sets
  - Lines with one or few sharers use a single tag,
  - Widely shared lines use additional tags
  - Tags remains small
- Fully characterized using analytical models
  - Negligible directory-induced invalidations
SCD: Scalable Coherence Directory

Figure 1: Example 3-way array organization used.

<table>
<thead>
<tr>
<th>Address, Index</th>
<th>Way 0</th>
<th>Way 1</th>
<th>Way 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(11, 3)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add sharer 64 to address 0x5CA1AB1E:

1. Lookup (0x5CA1AB1E, 0), all pointers are used → switch to multi-line
2. Allocate lines (0x5CA1AB1E, leafNum+1) with leafNum=1,2,8
3. Write leaf bit-vectors
4. Write (0x5CA1AB1E, 0) as a root bit-vector

Figure 2: SCD line formats. Field widths assume a 1024-sharer directory.

<table>
<thead>
<tr>
<th>Line Address</th>
<th>Type</th>
<th>Leaf Bit-Vector</th>
<th>Root Bit-Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>(44b)</td>
<td>(2b)</td>
<td>(32b)</td>
<td>(32b)</td>
</tr>
</tbody>
</table>

- **Type**:
  - 00: Unused (37b)
  - 01: Coherence State (5b), #ptrs (2b), 3x 10-bit sharer pointers (30b)
  - 10: Coherence State (5b), Root bit-vector (32b)
  - 11: Leaf number (5b), Leaf bit-vector (32b)
Why On-Chip Coherence Is Here to Stay

- **Traffic**: Coherence’s interconnection network traffic per miss scales when precisely tracking sharers.
- **Storage**: Hierarchy combined with inclusion enables efficient scaling of the storage cost for exact encoding of sharers.
- **Maintaining inclusion**: Chip architects can design a system with an inclusive shared cache with negligible recall rate, and thus can efficiently embed the tracking state in the shared cache.
- **Latency**: Though misses to actively shared blocks have greater latency than other misses, the latency ratio is tolerated, and the ratio need not grow as the number of cores increases.
- **Energy**: Based on these traffic and storage scalability analyses, we find no reason the energy overheads of coherence must increase with the number of cores.

**Conclusion**: Hierarchy of inclusive caches
Summary

- Wire delay
- Shared (manycore) vs. Private (few cores) LLC
- Multithreading vs. Multiprogramming
  - Instruction
  - Private data
  - Shared data
- Shared distributed cache: long term impact.
  - OS-based policies
  - Reactive NUCA
  - Cache partition -> Private cache
- Cache Coherence Directory