Multi-Core Memory Hierarchies

Lecture 8: DRAM Cache

Tong Dong

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

- How to improve LLC effective capacity without increasing physical area?
  - Cache Compression
    - Compression Algorithms
    - Cache structure: fixed- and variable-segment caches
  - Cache Compaction
    - Various spatial locality
  - Cache block indexing
    - Decoupled sectored cache: superblock
    - Indirect indexing cache: pointer-based
    - Region-based indexing: page-based
    - Hash indexing: skewed cache
    - Bitmap indexing: alloy cache
Model of cache performance

- Average Access Time in $i$-level cache:
  - Hit Time (NUCA)
  - Miss Penalty (Prefetching/DRAM Scheduling and Partition)
    \[ t_{Li} = (1 - m_{Li})t_{Li}^{hit} + m_{Li}t_{Li-1} \]

- For N cores and T thread per core, the miss ratio of the $i$-level cache:
  \[ m_{Li} = \left( \frac{C_{Li}}{\beta_{Li}I_{Li}(NT)} \right)^{1-\alpha_{Li}} \]
  - $\alpha$, $\beta$: locality (Replacement/Associativity)
  - $I(NT)$: interference function (Partition/Thread Scheduling)
  - $C$: total cache size (NUCA/Compression Cache/DRAM Cache/NVM Cache)
Memory-Wall & Power-Wall: 3D-Stack

Experimental demonstrations:
(a) 3D RRAM
(b) Efficient heat removal solutions
(c) Monolithic 3D “high-rise chip”

(1) Energy-efficient FETs
- 1D CNTs
- 2D layered nanomaterials

(2) High-density nonvolatile memories
- 3D RRAM: massive storage
- STT-MRAM: quick access

(3) Fine-grained monolithic 3D integration
- Compute + memory elements
- Ultradense connectivity using nanoscale vias

(4) Efficient heat removal

(5) Computation immersed in memory

On-chip nanoconvection/conduction solutions
3D-stacked DRAM

3D-stacked DRAM

2.5D-stacked DRAM
3D Stacking DRAM Memory

- On-chip wire latency
  - “In the forthcoming 65 nm regime, up to 77% of the delay will be attributed to the interconnect”. [ICCAD’04]
  - NUCA

- 3D stacking technology
  - The introduction of 3D circuits in earlier 00’s
  - Wide low-latency buses
**3D D-NUCA, Li et al. ISCA 2006**

- **Novel 3D network**
  - Indeed of 3D router design
  - 2D meshes per die + inter-die dTDMA buses (3D pillars)

**Figure 7. A high-level overview of the modified router of the pillar nodes.**

**Figure 8. A CPU has more cache banks in its vicinity in the 3D architecture.**

CPU and Cache placement
- No two core are stacked in the same vertical plane
- Low thermal emergency
- D-NUCA without migration between dies

Results
- 2D NUCA vs 3D NUCA
  - 37% IPC improvement
- 2D S-NUCA to 3D S-NUCA is better than to 2D D-NUCA
3D die stacking (wafer-to-wafer bounding)
- Significant reduction of interconnect both within/across dies in a system
- Lower power
  - Wire 30%
- Higher bandwidth
- Lower latency
- Thermal impact

Stacking different types of dies
Memory + Logic stacking
- Additional high-density stacked cache, not memory
  - 4MB on-chip SRAM L2
  - 8MB stacked SRAM L2 + 4MB on-chip SRAM L2
  - 32MB stacked DRAM L2 + 2MB on-chip tags
  - 64MB stacked DRAM L2 + 4MB on-chip tags

Reduction
- BW 3x
- CPMA 13%
- Power 66%
- Acceptable

Thermal results

Figure 5. Performance results for 2 threaded RMS benchmarks as cache capacity increases from 4MB to 64MB.
Logic + Logic stacking

Figure 9. Planar floorplan of a deeply pipelined

3D provides 15% added perf and 15% pwr savings at same frequency

<table>
<thead>
<tr>
<th></th>
<th>Pwr</th>
<th>Pwr %</th>
<th>Temp</th>
<th>Perf</th>
<th>Vcc</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>147</td>
<td>100%</td>
<td>99</td>
<td>100%</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Same Pwr</td>
<td>147</td>
<td>100%</td>
<td>127</td>
<td>129%</td>
<td>1</td>
<td>1.18</td>
</tr>
<tr>
<td>Same Freq.</td>
<td>125</td>
<td>85%</td>
<td>113</td>
<td>115%</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Same Temp</td>
<td>97.28</td>
<td>66%</td>
<td>99</td>
<td>108%</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>Same Perf.</td>
<td>68.2</td>
<td>46%</td>
<td>77</td>
<td>100%</td>
<td>0.82</td>
<td>0.82</td>
</tr>
</tbody>
</table>

10. 3D floorplan of the planar
DRAM and Main Memory Controller

L2 Cache → MSHRs → Memory Controller (MC) → MRQ → Arb. → DRAM Modules

One Rank → Bank n-1 → Bank 2 → Bank 1 → Bank 0

Data Bus → Addr Bus

One Bank

Bitcells

Row Decoder

Sense Amps

Row Buffer

Column Select
3D-Stacked Memory Architectures, Loh, ISCA 2008

- 2D: tRAS=36ns, tRCD, tCAS, tWR, tRP=12ns each;
- t(3D) ~ 2/3 * t(2D)

Figure 3. 3D-stacked DRAM on CPU with (a) one rank per layer and (b) ranks split across multiple layers.
3D-Stacked Memory Architectures, Loh, ISCA 2008

- 3D DRAM: 1.75x performance

![Graph showing performance speedup](image)

- Cached DRAM: multiple row buffer
- Increasing MSHR Capacity
- Scalable L2 Miss Handling
  - Direct-mapped MSHR
  - Vector Bloom Filter (VBF) MSHR
  - Additional 17.8% improvement

G. Loh. 3D-Stacked Memory Architectures for Multi-Core Processors. ISCA 2008.
**CHOP**, Jiang et al. HPCA 2010

- **Huge DRAM Cache Challenge**
  - Block granularity DRAM Cache: large tag area (1/10)
  - **Page granularity** DRAM Cache: waste bandwidth

- **CHOP (Caching HOT Pages)**
  - Filter cache: CHOP-FC
  - Memory-based filter cache: CHOP-MFC
  - Adaptive filter cache: CHOP-AFC

- **Hot page**
  - Pages that are heavily accessed (25% pages)
  - hot pages: the topmost accessed pages that contribute to 80% of the total access number.

<table>
<thead>
<tr>
<th>Workload</th>
<th>Hot Page Percentage</th>
<th>Hot Page Min #Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP</td>
<td>24.8%</td>
<td>95</td>
</tr>
<tr>
<td>SPECjApps</td>
<td>38.4%</td>
<td>65</td>
</tr>
<tr>
<td>SPECjbb</td>
<td>30.6%</td>
<td>93</td>
</tr>
<tr>
<td>TPCC</td>
<td>7.2%</td>
<td>64</td>
</tr>
<tr>
<td>Avg</td>
<td>25.2%</td>
<td>79</td>
</tr>
</tbody>
</table>
CHOP, Jiang et al. HPCA 2010

Results
- 30% performance improvement
- Several magnitudes lower area overhead
- Significantly lower memory bandwidth consumption
3D-stacked memory architecture with a vertical L2 fetch/write-back network using a large array of TSVs.

Figure 3: SMART-3D Memory Hierarchy: Implementing a 64B-Wide Bus with TSVs directly on top of Each L2 Subbank

LH-Cache, Loh and Hill, MICRO 2011

- 1GB DRAM Cache requires 96MB tag storage
- Contribution
  - Combining the tags and data in the DRAM row
  - Make hits faster: scheduling the tag and data accesses as a compound access to exploit row-buffer hit.
  - Make misses faster: using MissMap to eschew stacked DRAM access on all misses.
- Results
  - 92.9% of performance benefit of an ideal 1GB DRAM Cache with 96MB tag storage
LH-Cache, Loh and Hill, MICRO 2011

Figure 2: Mapping the tags and data of a cache set to a single DRAM row.

Figure 5: (a) MissMap entry covering a 1KB memory segment. (b) Setting a MissMap bit when installing a line in the DRAM cache, and (c) clearing a MissMap bit when evicting a line from the DRAM cache.
Alloy Cache, Qureshi and Loh, MICRO 2012

- First for latency, and then for hit rate
  - Direct-mapped cache

- Hit latency
  - Streaming tag and data together in a single burst

- Miss penalty
  - Memory Access Predictor: off-chip DRAM access without need to wait for a cache miss detection

- Results
  - 8.7% performance improvement than LH-Cache
Alloy Cache, Qureshi and Loh, MICRO 2012

Figure 3: Latency breakdown for two classes of isolated accesses X and Y. X has good row buffer locality and Y needs to activate the memory row to get serviced. The latency incurred in an activity is marked as [N] processor cycles.
Figure 5: Architecture and Operation of Alloy Cache that integrates Tag and Data (TAD) into a single entity called TAD. The size of data transfers is determined by a 16-byte wide data-bus, hence minimum transfer of 80 bytes for obtaining one TAD.

Figure 7: Cache Access Models: Serial vs Parallel

Block-based vs. page-based 3D-stacked DRAM Cache

Figure 2. A DRAM die stacked on top of the logic die, used as (a) a block-based cache or as (b) a page-based cache. For the block-based design, one tag entry corresponds to one data block. For the page-based design, only the useful blocks (accessed by the cores) are shown in the figure, and one tag entry corresponds to one page.
Prefetching Based on Spatial Locality

- Spatial Memory Streaming
  - In commercial applications, memory accesses within a region can be sparse, un-strided.
  - But, they are repeatable and then predictable.

**FIGURE 3. Pattern History Table and prediction process.** Upon a trigger access that matches in the PHT, the region base address and spatial pattern are transferred to a prediction register, beginning the streaming process.
Server Workload: low temporal locality

Footprint Cache
- Decoupling cache allocation unit from the fetch unit
- Allocating large pages but fetching only those blocks that will be used

Figure 3. Footprint Cache tag array and Footprint History Table (FHT).

ATCache, PACT’14

- A small SRAM tag cache
  - Maintain the tags in DRAM -> critical path
  - Maintain the tags in SRAM -> area cost

Bi-Modal, MICRO’14.
Trade-offs in architecting stacked DARM
- either as part of main memory
- or as a hardware-managed cache

CAMEO
- Retains recently accessed data lines in stacked DRAM
- Swaps out the victim line to off-chip memory.

Line Location Table (LLT)
- To tracks the physical location of all data lines

Line Location Predictor (LLP)
- To avoid the serialization of the LLT look-up and access
CAMEO, Micro’14

CAMEO, Micro’14

(a) Hardware-Managed Cache

(b) Two Level Memory

(c) CAMEO

Alloy Cache + Footprint Cache
- Tag overhead, Hit latency, Hit ratio
- Set Associative
- Way Prediction
- Footprint Predictor

Figure 3. DRAM row organization in the Unison Cache design.
Tag Tables, \textit{HPCA’15}

- Support for very large caches with fine-grained block sizes with low overhead
- A compact “base-plus-offset” encoding
- Forward page table for hot pages
TDC: Tagless DRAM Cache, ISCA’15

- TLB and cache tag array overhead
- cTLB, cache-map TLB, which store virtual-to-cache address mappings.
- GIPT, Global Inverted Page Table
- Victim Cache

- F-TDC, HPCA’16
  - Footprint+TDC

DICE: Compression DRAM Cache

- **DICE (Dynamic Indexing Cache Compression)**
  - Spatial Indexing if compressible, otherwise
  - Traditional Set Indexing (TSI)
  - Cache Index Predictors (CIP). 94%

Figure 8: Design of DICE. DICE is implemented by deciding index policy on write, and predicting index policy on read.

Figure 9: History-based Cache Index Predictor. CIP tracks history at page granularity.

V. Young, et al. DICE: Compressing DRAM Caches for Bandwidth and Capacity. ISCA-44, 2017. (8)
Banshee: HW/SW Cooperating DRAM Caching

- Bandwidth Efficiency: First Design Constrain
  - In-Package and Off-Package DRAM same latency
- Tag: TLB and Page Table Entry
- Bandwidth-aware frequency-based replacement policy

<table>
<thead>
<tr>
<th>Scheme</th>
<th>DRAM Cache Hit</th>
<th>DRAM Cache Miss</th>
<th>Replacement Traffic</th>
<th>Replacement Decision</th>
<th>Large Page Caching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unison [32]</td>
<td>In-package traffic: 128 B (data + tag read and update) Latency: ~1x</td>
<td>In-package traffic: 96 B (spec. data + tag read) Latency: ~2x</td>
<td>On every miss Footprint size [31]</td>
<td>Hardware managed, set-associative, LRU</td>
<td>Yes</td>
</tr>
<tr>
<td>Alloy [50]</td>
<td>In-package traffic: 96 B (data + tag read) Latency: ~1x</td>
<td>In-package traffic: 96 B (spec. data + tag read) Latency: ~2x</td>
<td>On some misses Cacheline size (64 B)</td>
<td>Hardware managed, direct-mapped, stochastic [20]</td>
<td>Yes</td>
</tr>
<tr>
<td>TDC [38]</td>
<td>In-package traffic: 64 B Latency: ~1x TLB coherence</td>
<td>In-package traffic: 0 B Latency: ~1x TLB coherence</td>
<td>On every miss Footprint size [28]</td>
<td>Hardware managed, fully-associative, FIFO</td>
<td>No</td>
</tr>
<tr>
<td>HMA [44]</td>
<td>In-package traffic: 64 B Latency: ~1x</td>
<td>In-package traffic: 0 B Latency: ~1x</td>
<td>Software managed, high replacement cost</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Banshee (This work)</td>
<td>In-package traffic: 64 B Latency: ~1x</td>
<td>In-package traffic: 0 B Latency: ~1x</td>
<td>Only for hot pages Page size (4 KB)</td>
<td>Hardware managed, set-associative, frequency based</td>
<td>Yes</td>
</tr>
</tbody>
</table>

ACCORD ISCA 2018

- Way Prediction (Union Cache): per-set storage, causing multi-megabyte storage overheads
- ACCORD: Associativity via Coordinated Way-Install and Way-Prediction
  - Steers an incoming line to a “preferred way” based on the line address and
  - Uses the preferred way as the default way prediction.
  - Probabilistic Way-Steering (PWS)
  - Ganged Way-Steering (GWS): spatially contiguous region
  - Skewed Way-Steering (SWS): highly-associative cache
eDRAM Cache

POWER8 Processor

Technology
- 22nm SOI, eDRAM, 15 ML 650mm2

Caches
- 512 KB SRAM L2 / core
- 96 MB eDRAM shared L3
- Up to 128 MB eDRAM L4 (off-chip)

Memory
- Up to 230 GB/s sustained bandwidth

Bus Interfaces
- Durable open memory attach interface
- Integrated PCIe Gen3
- SMP Interconnect
- CAPI (Coherent Accelerator Processor Interface)

Cores
- 12 cores (SMT8)
- 8 dispatch, 10 issue, 16 exec pipe
- 2X internal data flows/queue
- Enhanced prefetching
- 64K data cache, 32K instruction cache

Accelerators
- Crypto & memory expansion
- Transactional Memory
- VMM assist
- Data Move / VM Mobility

Energy Management
- On-chip Power Management Micro-controller
- Integrated Per-core VRM
- Critical Path Monitors

© 2013 International Business Machines Corporation
## eDRAM Cache

### Table 1: Comparison of various memory technologies for on-die caches.

<table>
<thead>
<tr>
<th></th>
<th>(A) SRAM</th>
<th>(B) STT-RAM</th>
<th>eDRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(C) ITTC</td>
</tr>
<tr>
<td>Cell schematic</td>
<td><img src="image" alt="Cell schematic" /></td>
<td><img src="image" alt="Cell schematic" /></td>
<td><img src="image" alt="Cell schematic" /></td>
</tr>
<tr>
<td>Process</td>
<td>CMOS</td>
<td>CMOS + MTJ</td>
<td>CMOS + Cap</td>
</tr>
<tr>
<td>Cell size ($F^2$)</td>
<td>120 - 200</td>
<td>6 - 50</td>
<td>20 - 50</td>
</tr>
<tr>
<td>Data storage</td>
<td>Latch</td>
<td>Magnetization</td>
<td>Capacitor</td>
</tr>
<tr>
<td>Read time</td>
<td>Short</td>
<td>Short</td>
<td>Short</td>
</tr>
<tr>
<td>Write time</td>
<td>Short</td>
<td>Long</td>
<td>Short</td>
</tr>
<tr>
<td>Read energy</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Write energy</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Leakage</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Endurance</td>
<td>$10^{16}$</td>
<td>$&gt; 10^{15}$</td>
<td>$10^{16}$</td>
</tr>
<tr>
<td>Retention time</td>
<td>-</td>
<td>-</td>
<td>&lt; 100 us *</td>
</tr>
<tr>
<td>Features</td>
<td>(+) Fast</td>
<td>(+) Non-volatile</td>
<td>(+) Low leakage</td>
</tr>
<tr>
<td></td>
<td>(-) Large area</td>
<td>(+) Potential to scale</td>
<td>(+) Small area</td>
</tr>
<tr>
<td></td>
<td>(-) Leakage</td>
<td>(+) Extra process</td>
<td>(-) Extra process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-) Long write time</td>
<td>(-) Destructive read</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-) High write energy</td>
<td>(-) Refresh</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* *32 nm technology node
eDRAM Cache

- Optimized Refreshing: Dead-line prediction

- Refrint, Agrawal and Torrellas, HPCA’13
  - Cold line vs. Hot lines
Summary

- How to design a very large cache?

- 3D-stacked DRAM Cache
  - 5x high bandwidth
  - 1/3 low latency? -> Same latency
  - Tag storage
  - Access Latency

- eDRAM
  - Refresh management