Advanced Caches

ECE/CS 752 Fall 2017

Prof. Mikko H. Lipasti
University of Wisconsin-Madison

Lecture notes based on notes by John P. Shen and Mark Hill
Updated by Mikko Lipasti

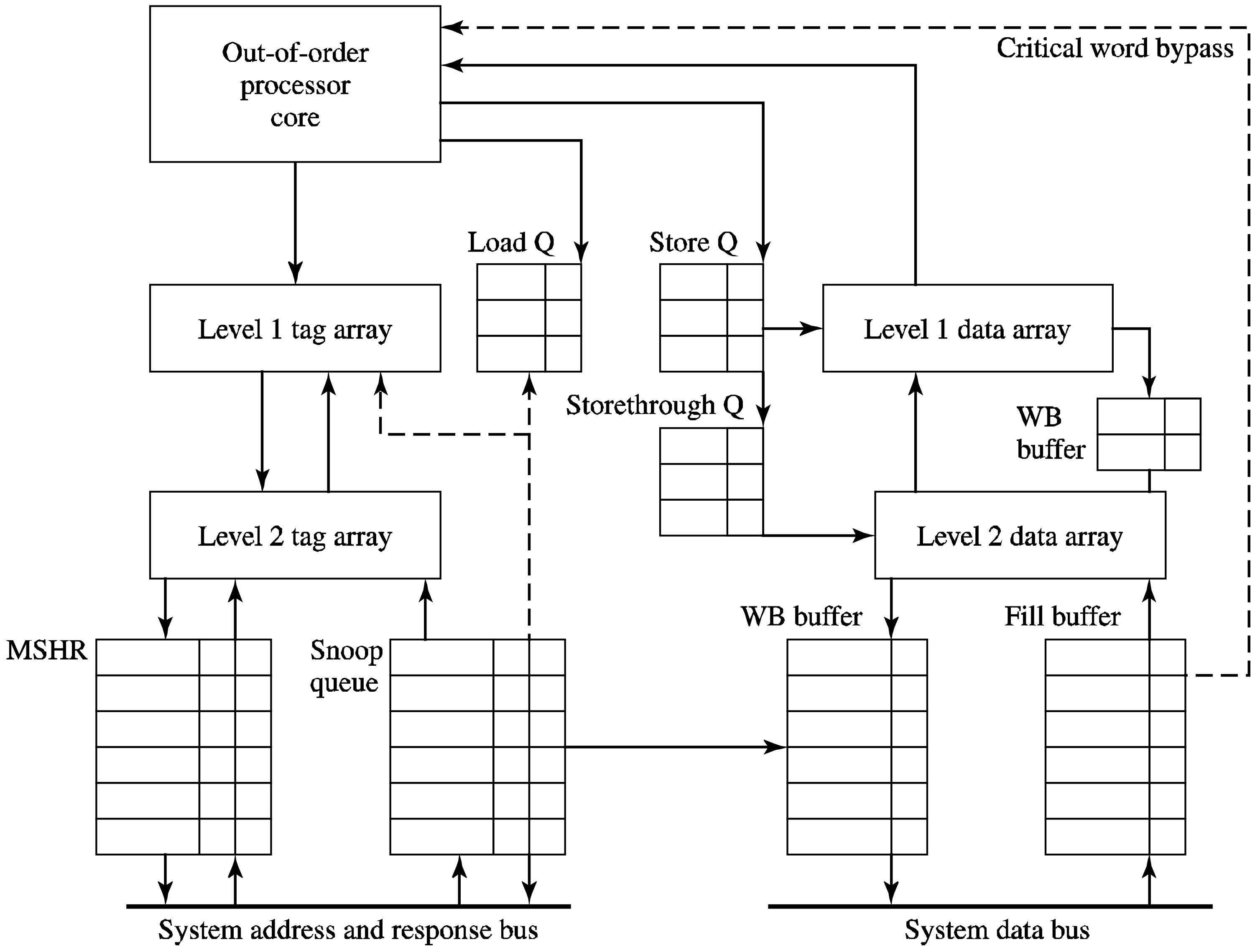
Readings

- Read on your own:
  - Review: Shen & Lipasti Chapter 3
  - W.-H. Wang, J.-L. Baer, and H. M. Levy. “Organization of a two-level virtual-real cache hierarchy,” Proc. 16th ISCA, pp. 140-148, June 1989 (B6) Online PDF
  - Read Sec. 1, skim Sec. 2, read Sec. 3: Bruce Jacob, “The Memory System: You Can't Avoid It, You Can't Ignore It, You Can't Fake It,” Synthesis Lectures on Computer Architecture 2009 4:1, 1-77. Online PDF
- To be discussed in class:
  - Review #1 due 11/1/2017: Andreas Sembrant, Erik Hagersten, David Black-Schaffer, “The Direct-to-Data (D2D) cache: navigating the cache hierarchy with a single lookup,” Proc. ISCA 2014, June 2014.. Online PDF
  - Review #2 due 11/3/2017: Jishen Zhao, Sheng Li, Doe Hyun Yoon, Yuan Xie, and Norman P. Jouppi. 2013. Kiln: closing the performance gap between systems with and without persistence support. In Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-46). ACM, New York, NY, USA, 421-432. Online PDF
  - Review #3 due 11/6/2017: T. Shaw, M. Martin, A. Roth, “NoSQ: Store-Load Communication without a Store Queue,” in Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006. Online PDF

Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- Beyond simple blocks
- Two level caches

Coherent Memory Interface



Coherent Memory Interface

- Load Queue
  - Tracks inflight loads for aliasing, coherence
- Store Queue
  - Defers stores until commit, tracks aliasing
- Storethrough Queue or Write Buffer or Store Buffer
  - Defers stores, coalesces writes, must handle RAW
- MSHR
  - Tracks outstanding misses, enables *lockup-free caches* [Kroft ISCA 91]
- Snoop Queue
  - Buffers, tracks incoming requests from coherent I/O, other processors
- Fill Buffer
  - Works with MSHR to hold incoming partial lines
- Writeback Buffer
  - Defers writeback of evicted line (demand miss handled first)

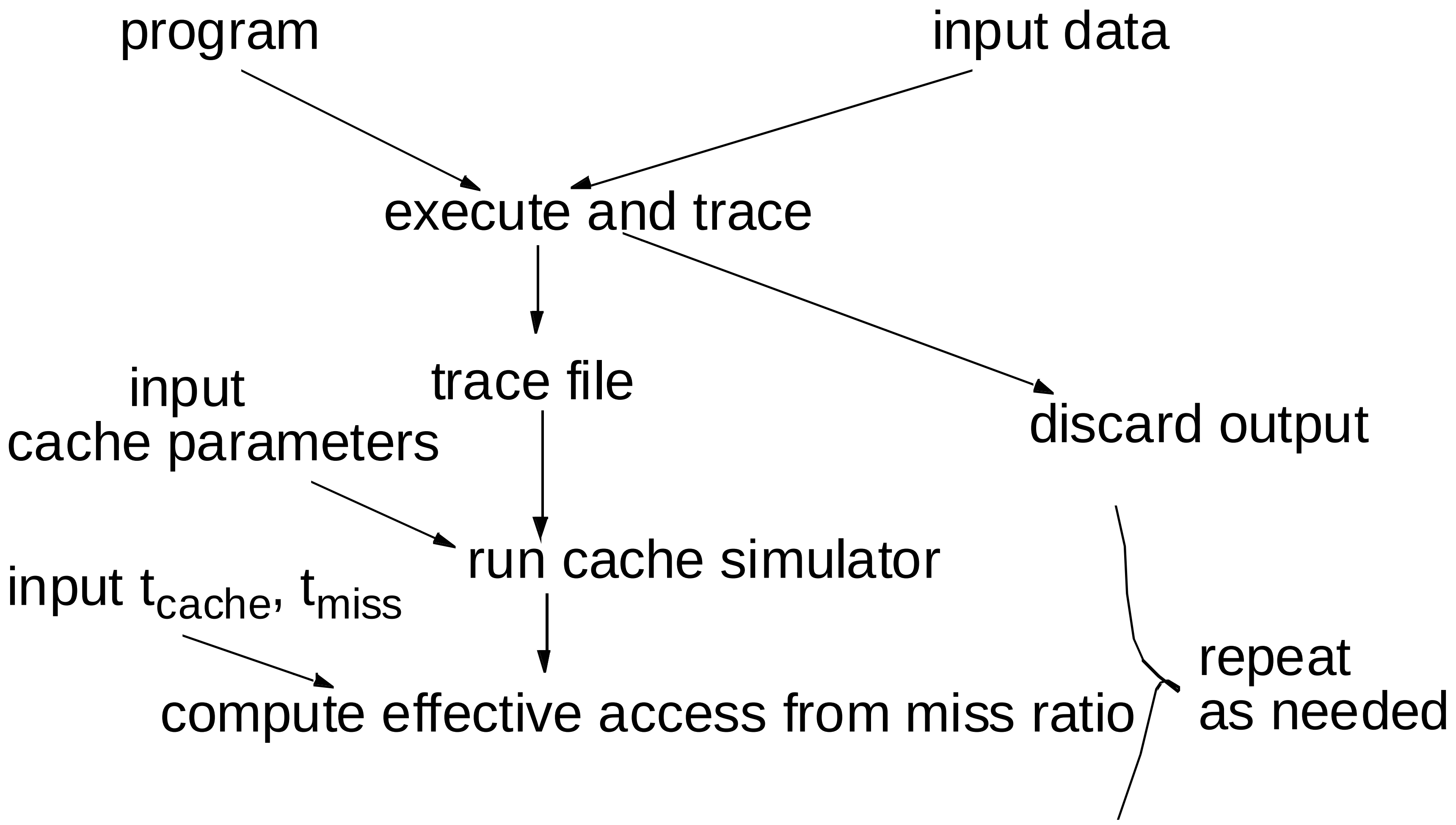
Evaluation Methods - Counters

- Counts hits and misses in hardware
  - see [Clark, TOCS 1983]
  - Intel VTune tool
- Accurate
- Realistic workloads - system, user, everything
- Requires machine to exist
- Hard to vary cache parameters
- Experiments not deterministic

Evaluation Methods - Analytical

- Mathematical expressions
  - Insight - can vary parameters
  - Fast
  - Absolute accuracy suspect for models with few parameters
  - Hard to determine many parameter values
  - Not widely used today

Evaluation: Trace-Driven Simulation



Evaluation: Trace-Driven Simulation

- Experiments repeatable
- Can be accurate
- Much recent progress
- Reasonable traces are very large ~gigabytes
- Simulation can be time consuming
- Hard to say if traces representative
- Don’t model speculative execution

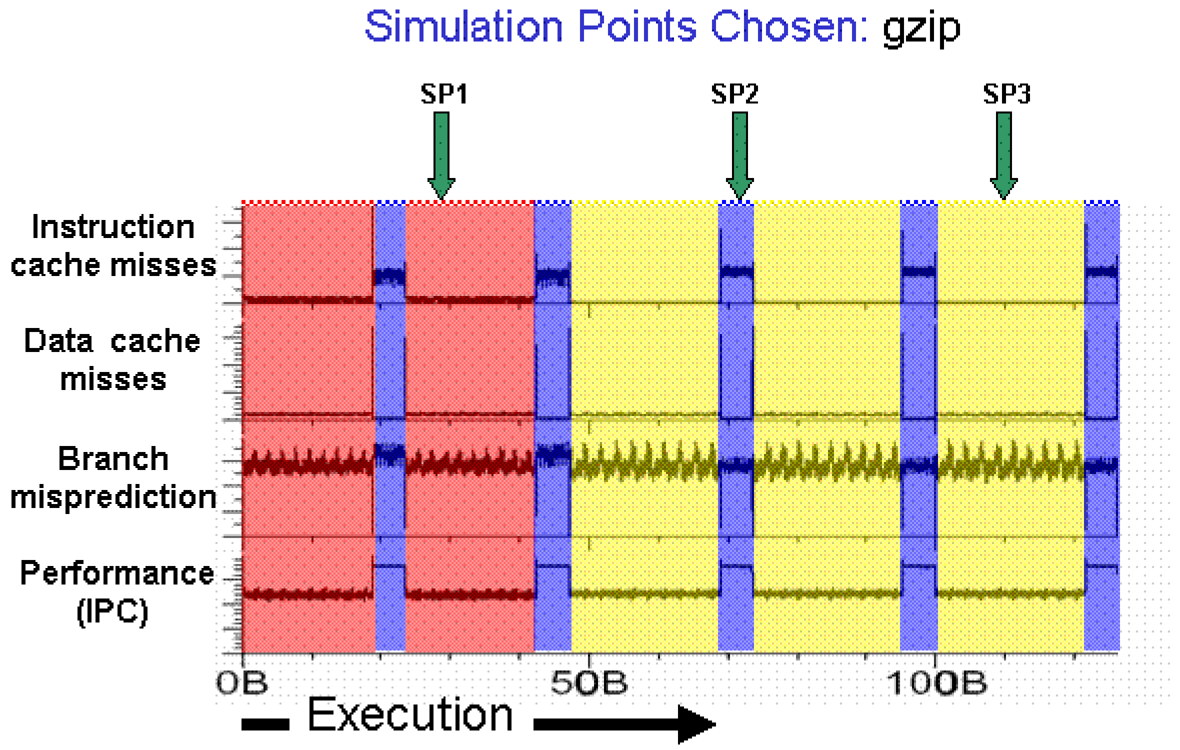
Evaluation: Execution-Driven Simulation

- Do full processor simulation each time
  - Actual performance; with ILP miss rate means nothing
    - Non-blocking caches
    - Prefetches (timeliness)
    - Pollution effects due to speculation
  - No need to store trace
  - Much more complicated simulation model
- Time-consuming - but good programming can help
- Very common today

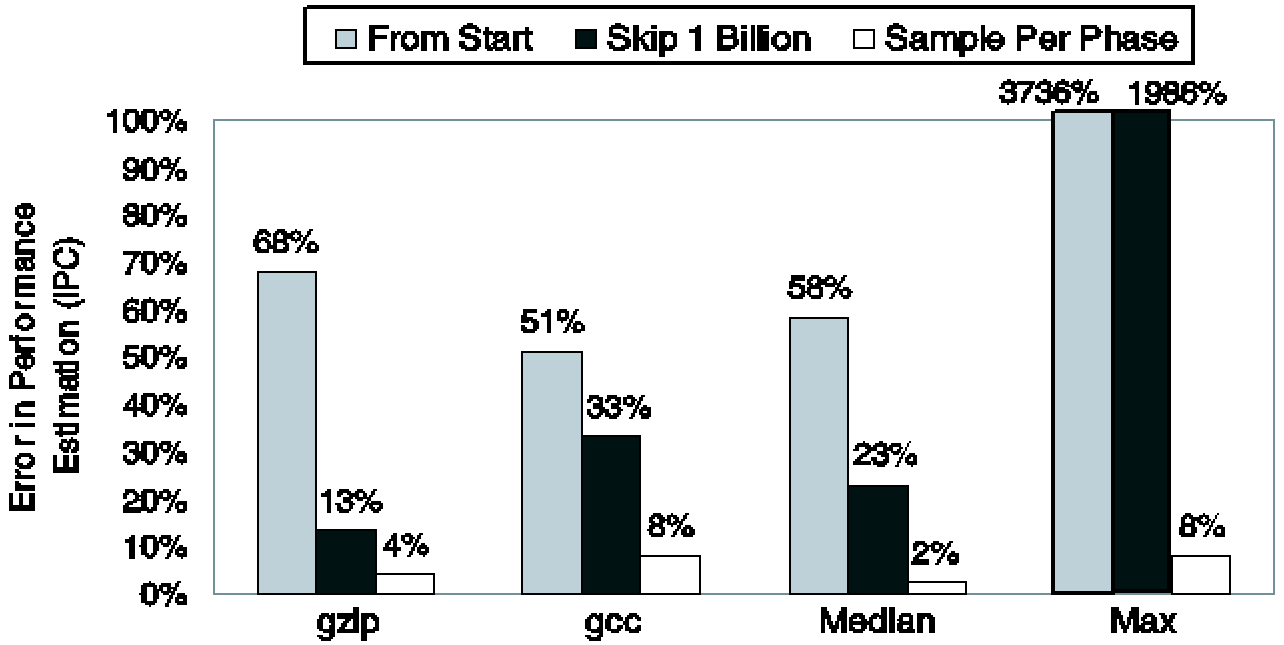
Trace/Execution Sampling

- We always sample:
  - Even entire programs aren't what the end user is running
- Sampling in space
  - Set sampling: monitor only a subset of cache sets
- Sampling in time
  - Cold start concerns
  - Choosing representative interval: where, how long, how many?
- Simpoints [Sherwood et al. ASPLOS 02]
  - Offline phase analysis to choose one or more intervals that are cumulatively representative of the whole program
- SMARTS [Wenisch et al., ISCA 2003]
  - Detailed simulation in short bursts
  - Alternate with fast, functional simulation that keeps caches/predictors warm

Simpoints [Sherwood et al. ASPLOS 02]



Simpoints [Sherwood et al. ASPLOS 02]

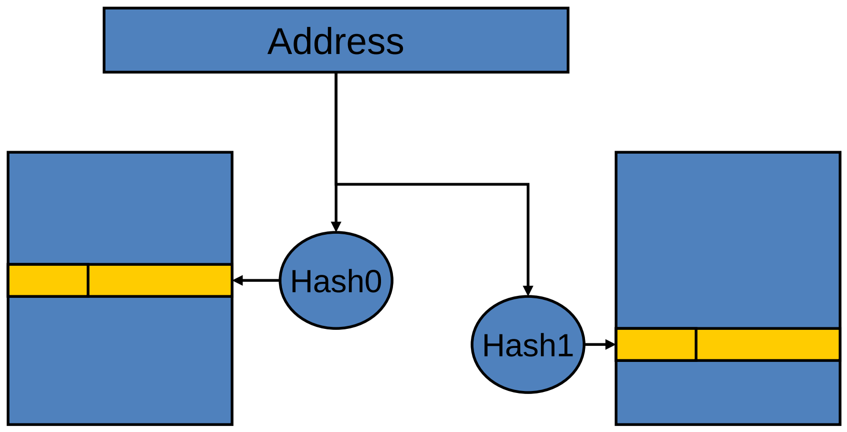


Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- **Better miss rate: skewed associative caches, victim caches**
- Reducing miss costs through software restructuring
- Beyond simple blocks
- Two level caches

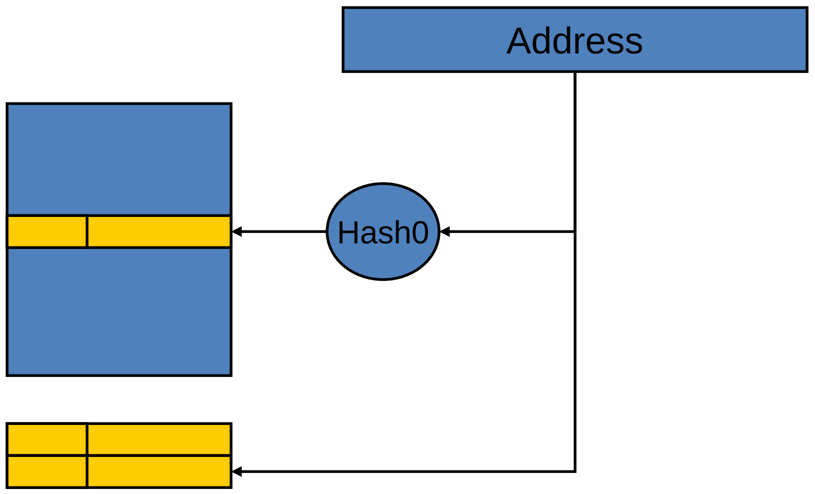
Seznec's Skewed Associative Cache

- Alleviates conflict misses in a conventional set assoc cache
- If two addresses conflict in 1 bank, they conflict in the others too
  - e.g., 3 addresses with same index bits will thrash in 2-way cache
- Solution: use different hash functions for each bank
- Works reasonably well: more robust conflict miss behavior
- But: how do you implement replacement policy?



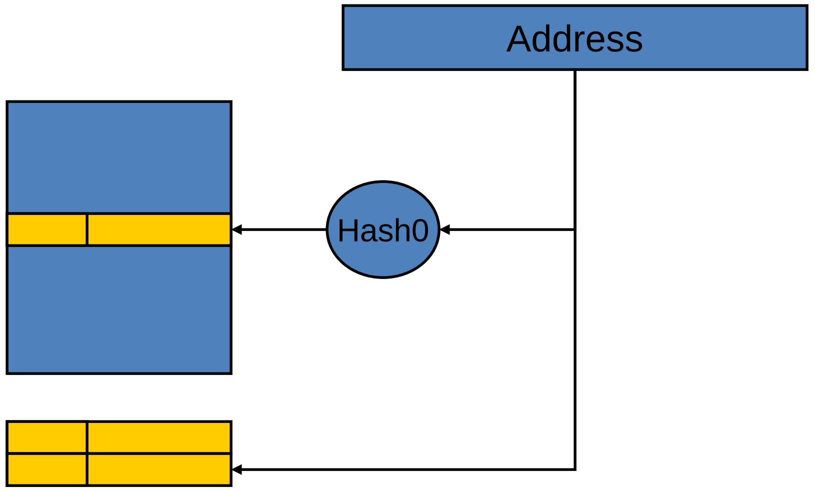
Jouppi's Victim Cache

- Targeted at conflict misses
- Victim cache: a small fully associative cache
  - holds victims replaced in direct-mapped or low-assoc
  - LRU replacement
  - a miss in cache + a hit in victim cache
    - => move line to main cache
- Poor man's associativity
  - Not all sets suffer conflicts; provide limited capacity for conflicts



Jouppi’s Victim Cache

- Removes conflict misses, mostly useful for DM or 2-way
  - Even one entry helps some benchmarks
  - I-cache helped more than D-cache
- Versus cache size
  - Generally, victim cache helps more for smaller caches
- Versus line size
  - helps more with larger line size (why?)
- Used in Pentium Pro (P6) I-cache to handle SMC



Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- **Reducing miss costs through software restructuring**
- Beyond simple blocks
- Two level caches

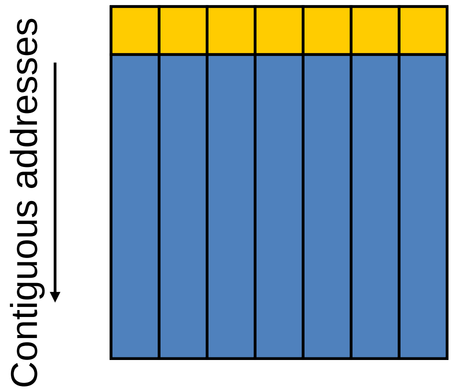
Software Restructuring

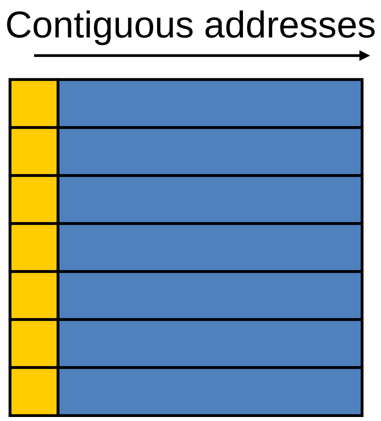
- If column-major (Fortran)
  - x[i+1, j] follows x [i,j] in memory
  - x[i,j+1] long after x[i,j] in memory
- Poor code

```
for i = 1, rows
   for j = 1, columns
      sum = sum + x[i,j]
```

- Conversely, if row-major (C/C++)
- Poor code

```
for j = 1, columns
   for i = 1, rows
      sum = sum + x[i,j]
```



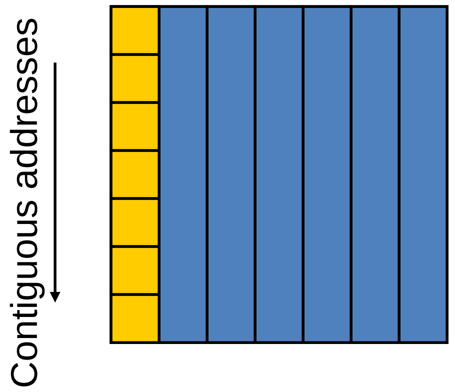


Software Restructuring

- **Better column-major code**

```
for j = 1, columns
  for i = 1, rows
    sum = sum + x[i,j]
```

- **Optimizations - need to check if it is valid to do them**
  - **Loop interchange (used above)**
  - **Blocking**
  - **Etc.**
- **Hard:** pointers, indirection, unknown loop bounds, sparse matrices



Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- **Beyond simple blocks**
- Two level caches

Sublines

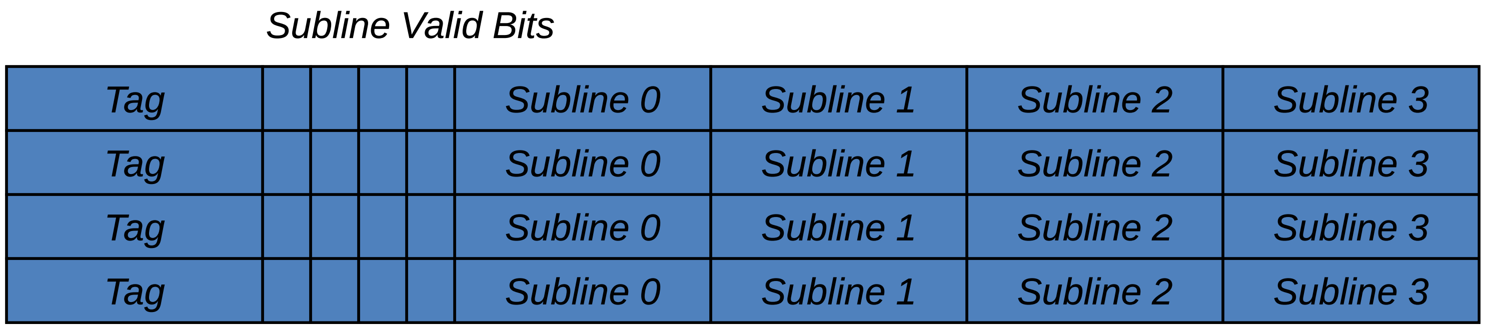
- Break blocks into
  - Address block associated with tag
  - Transfer block to/from memory (subline, sub-block)
- Large address blocks
  - Decrease tag overhead
  - But allow fewer blocks to reside in cache (fixed mapping)

Subline Valid Bits

Tag					*Subline 0*	*Subline 1*	*Subline 2*	*Subline 3*
Tag					*Subline 0*	*Subline 1*	*Subline 2*	*Subline 3*
Tag					*Subline 0*	*Subline 1*	*Subline 2*	*Subline 3*
Tag					*Subline 0*	*Subline 1*	*Subline 2*	*Subline 3*

Sublines

- Larger transfer block
  - Exploit spatial locality
  - Amortize memory latency
  - But take longer to load
  - Replace more data already cached (more conflicts)
  - Cause unnecessary traffic
- Typically used in large L3/L4/DRAM caches
- Sublines tracked by MSHR during pending fill



Latency vs. Bandwidth

- Latency can be handled by
  - Hiding (or tolerating) it - out of order issue, nonblocking cache
  - Reducing it – better caches
- Parallelism helps to hide latency
  - MLP – multiple outstanding cache misses overlapped
- But increases bandwidth demand
- Latency ultimately limited by physics

Latency vs. Bandwidth

- Bandwidth can be handled by “spending” more (hardware cost)
  - Wider buses, interfaces
  - Banking/interleaving, multiporting
- Ignoring cost, a well-designed system should never be bandwidth-limited
  - Can’t ignore cost!
- Bandwidth improvement usually increases latency
  - No free lunch
- Hierarchies decrease bandwidth demand to lower levels
  - Serve as traffic filters: a hit in L1 is filtered from L2
- Parallelism puts more demand on bandwidth
- If average b/w demand is not met => infinite queues
  - Bursts are smoothed by queues
- If burst is much larger than average => long queue
  - Eventually increases delay to unacceptable levels

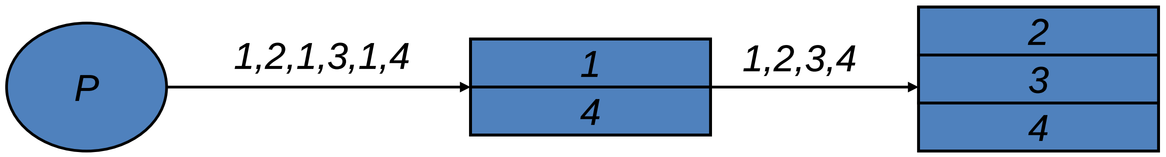
Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- Beyond simple blocks
- **Multilevel caches**

Multilevel Caches

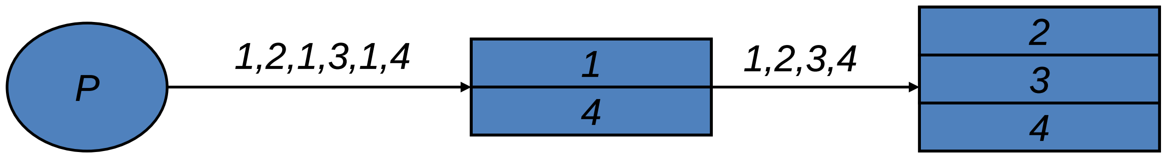
- Ubiquitous in high-performance processors
  - Gap between L1 (core frequency) and main memory too high
  - Level 2 usually on chip, level 3 on or off-chip, level 4 off chip
- Inclusion in multilevel caches
  - Multi-level inclusion holds if L2 cache is superset of L1
  - Can handle virtual address synonyms
  - Filter coherence traffic: if L2 misses, L1 needn’t see snoop
  - Makes L1 writes simpler
    - For both write-through and write-back

Multilevel Inclusion



- Example: local LRU not sufficient to guarantee inclusion
  - Assume L1 holds two and L2 holds three blocks
  - Both use local LRU
- Final state: L1 contains 1, L2 does not
  - Inclusion not maintained
- Different block sizes also complicate inclusion

Multilevel Inclusion



- Inclusion takes effort to maintain
  - Make L2 cache have bits or pointers giving L1 contents
  - Invalidate from L1 before replacing from L2
  - In example, removing 1 from L2 also removes it from L1
- Number of pointers per L2 block
  - L2 blocksize/L1 blocksize
- Reading list: [Wang, Baer, Levy ISCA 1989]

Multilevel Miss Rates

- Miss rates of lower level caches
  - Affected by upper level filtering effect
  - LRU becomes LRM, since “use” is “miss”
  - Can affect miss rates, though usually not important
- Miss rates reported as:
  - Miss per instruction
  - Global miss rate
  - Local miss rate
  - “Solo” miss rate
    - L2 cache sees all references (unfiltered by L1)

Advanced Memory Hierarchy

- Coherent Memory Interface
- Evaluation methods
- Better miss rate: skewed associative caches, victim caches
- Reducing miss costs through software restructuring
- Beyond simple blocks
- Multilevel caches