Computer Organization

Douglas Comer

Computer Science Department
Purdue University
250 N. University Street
West Lafayette, IN 47907-2066

http://www.cs.purdue.edu/people/comer

XII

Caches And Caching

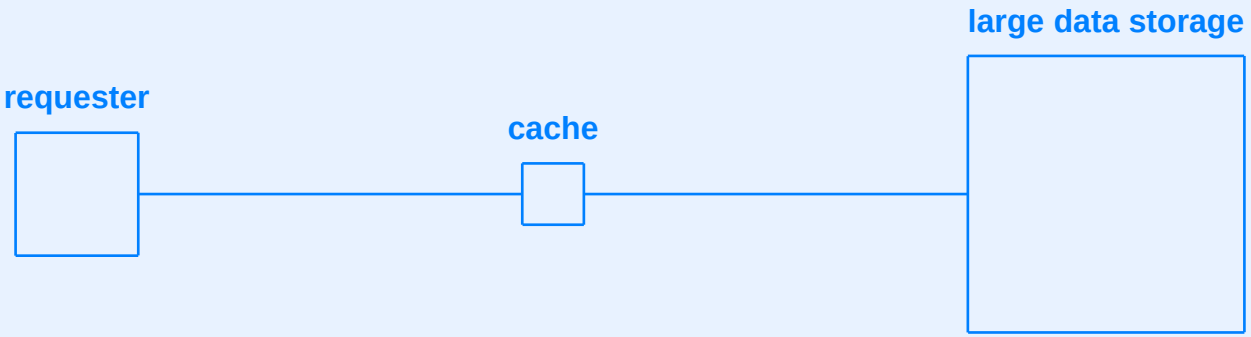
Caching

- Key concept in computing
- Used in hardware and software
- Memory cache can reduce the Von Neumann bottleneck

Cache

- Acts as an intermediary
- Located between source of requests and source of replies
- Contains high-speed, temporary storage
- Keeps a copy of selected items
- Answers requests from local copy when possible

Illustration Of A Cache



Cache Characteristics

- Small (usually much smaller than storage needed for entire set of items)
- Active (makes decisions)
- Transparent (invisible to both requester and data store)
- Automatic (uses sequence of requests; does not receive extra instructions)

Generality Of Caching

- Hardware, software, or combination
- Small or large data items (byte of memory or complete file)
- Generic data items (e.g., disk block)
- Specific data item (e.g., document from a word processor)
- Textual data (e.g., an email message)

Generality Of Caching
(continued)

- Nontextual data (e.g., an image, an audio file, or a video clip)
- A single computer system (e.g., between a processor and a memory)
- Many computer systems (e.g., between a set of desktop computers and a database server)
- Systems that are designed to retrieve data (e.g., the World Wide Web)
- Systems that store as well as retrieve data (e.g., a physical memory)

The Importance Of Caching

Caching is a fundamental optimization technique used throughout most hardware and software systems that retrieve information. Caching is a broad concept; data items kept in a cache are not limited to a specific type, form, or size.

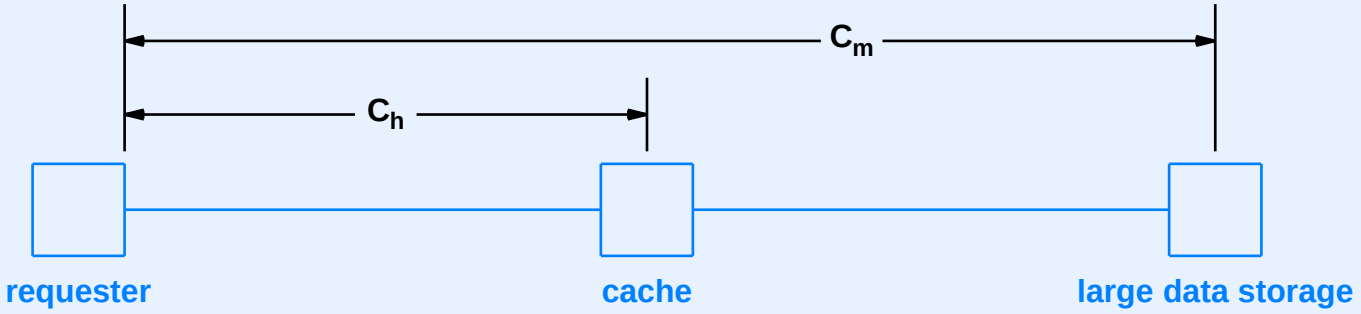
Cache Terminology

- *Cache hit*
  - Request that can be satisfied from cache
  - No need to access data store
- *Cache miss*
  - Request cannot be satisfied from cache
  - Cache retrieves item from data store

Cache Terminology
(continued)

- *Locality of reference*
  - Refers to repetitions of same request
  - High locality means many repetitions
  - Low locality means few repetitions
- Note: cache works well with high locality of reference

Cache Performance



- Cost measured with respect to requester

Worst Case Cache Performance

- C_h is the cost of an item found in the cache
- C_m is the cost of an item not found in the cache
- Average cost per request is C_m
- Worst case for sequence of N requests

$$C_{worst} = N\ C_m$$

Best Case Cache Performance

- Best case for sequence of N requests is

$$C_{best} = C_m + (N - 1)\, C_h$$

- Average cost per request is

$$\frac{C_m + (N - 1)\, C_h}{N} = \frac{C_m}{N} - \frac{C_h}{N} + C_h$$

- As $N \rightarrow \infty$, average cost becomes C_h

Summary Of Costs

If we ignore overhead, in the worst case, the performance of caching is no worse than if the cache were not present. In the best case, the cost per request is approximately equal to the cost of accessing the cache, which is lower than the cost of accessing the data store.

Definition Of Hit and Miss Ratios

- *Hit ratio*
  - Percentage of requests satisfied from cache
  - Given as value between 0 and 1
- *Miss ratio*
  - Percentage of requests not satisfied from cache
  - Equal to 1 minus hit ratio

Cache Performance On A Typical Sequence

- Access cost depends on hit ratio

$$Cost = r\,C_h + (1 - r)\,C_m$$

  where r is the hit ratio

- Note: performance improves if hit ratio increases or cost of access from cache decreases

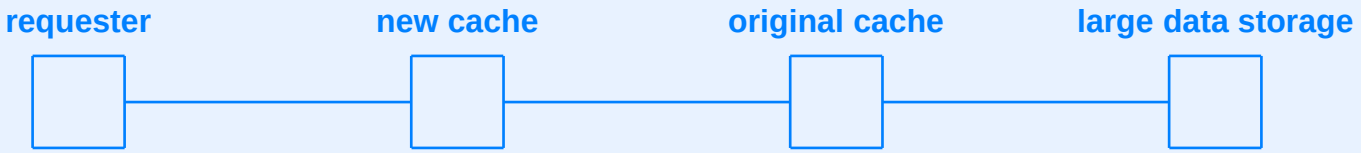
Cache Replacement Policy

- Cache is smaller than data store
- Once cache is full, existing item must be ejected before another can be inserted
- *Replacement policy* chooses items to eject
- Most popular replacement policy known as *Least Recently Used* (*LRU*)
  - Tends to retain items that will be requested again

Multi-level Cache Hierarchy

- Can use multiple caches to improve performance
- Arranged in hierarchy by speed

Illustration Of Two-Level Cache



Analysis Of Two-Level Cache

- Cost is:

$$Cost = r_1 C_{h1} + r_2 C_{h2} + (1 - r_1 - r_2) C_m$$

- r_1 is fraction of hits for the new cache
- r_2 is fraction of hits for the original cache
- C_{h1} is cost of accessing the new cache
- C_{h2} is cost of accessing the original cache

Preloading Caches

- Optimization technique
- Stores items in cache *before* requests arrive
- Works well if data accessed in related groups
- Examples
  - When web page is fetched, web cache preloads images
  - When byte of memory is fetched, memory cache can preload succeeding bytes

Memory Cache

- Several memory mechanisms operate as a cache
  - TLB
  - Demand paging
  - Physical memory cache

Demand Paging Performance

Cache analysis shows that using demand paging on a computer system with a small physical memory can perform almost as well as if the computer had a physical memory large enough for the entire virtual address space.

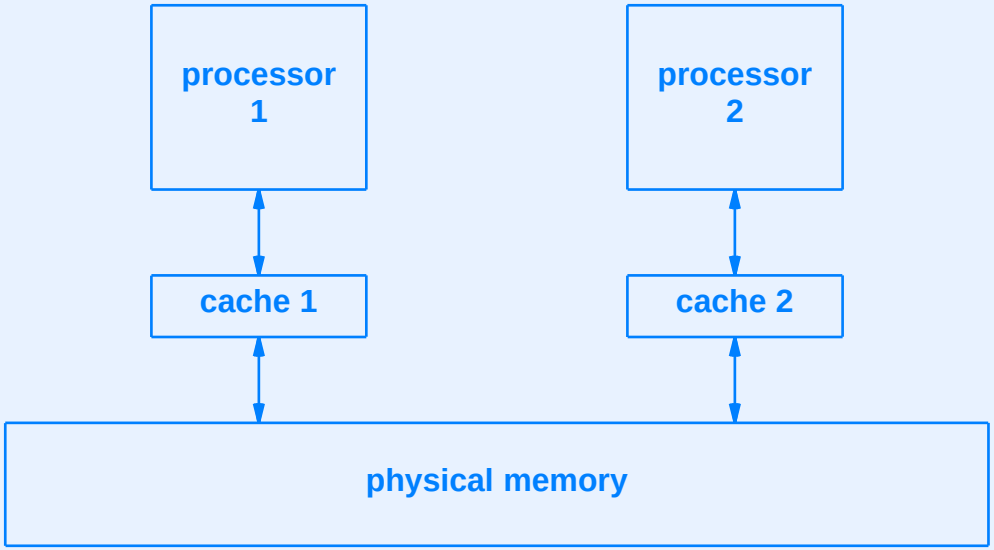
Physical Memory Cache

- Located between processor and physical memory
- Smaller than physical memory
- Note: sophisticated cache hardware operates in parallel to achieve high performance:
  - Search local cache
  - Send request to underlying memory
- If answer found in cache, cancel request to memory

Two Basic Types Of Cache

- Differ in how they handle *write* operation
- *Write-through*
  - Keep copy of item in cache
  - Send *write* request on to physical memory
- *Write-back*
  - Keep copy of item in cache
  - Only write copy to physical memory when necessary
  - Works well for frequent updates (e.g., a loop increments a value)

Illustration Of System With Multiple Caches



- Write-back means each cache can retain copy of item
- *Cache coherence* needed to ensure correctness

Motivation For Multi-Level Memory Cache

- Traditional memory cache was separate from both the memory and the processor
- To access traditional memory cache, a processor used pins that connect the processor chip to the rest of the computer
- Using pins to access external hardware takes much longer than accessing functional units that are internal to the processor chip
- Advances in technology have made it possible to increase the number of transistors per chip, which means a processor chip can contain a larger cache

Multi-Level Memory Cache Terminology

- *Level 1 cache* (*L1 cache*) on the processor chip
- *Level 2 cache* (*L2 cache*) external to the processor
- *Level 3 cache* (*L3 cache*) built into the physical memory

Cost Of Accessing Memory

Computer systems use a multi-level cache hierarchy in which an L1 cache is embedded on the processor chip, an L2 cache is external to the processor, and an L3 cache is built into the physical memory. In the best case, a multi-level cache makes the cost of accessing memory approximately the same as the cost of accessing a register.

Instruction And Data Caches

- Instruction references are typically sequential
  - High locality of reference
- Data references are more random
  - Lower locality of reference
- Question: does performance improve with separate caches for data and instructions?

Instruction And Data Caches
(continued)

- Random references tend to lower cache performance
- Separating instruction and data caches can improve performance
- However: if cache is ‘‘large enough’’, separation doesn’t help

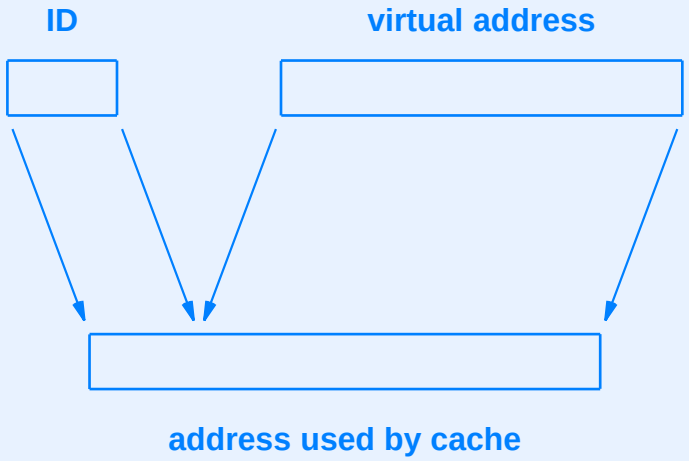
Virtual Memory Caching

- Can choose to cache
  - Physical memory address and contents
  - Virtual memory address and contents
- Notes
  - If MMU is off-chip, L1 cache must use virtual addresses
  - Multiple applications use *same* virtual address space

Caching Virtual Addresses

- OS performs cache *flush* operation when changing applications
- Cache includes disambiguating tag with each entry (e.g., application ID)

Illustration Of ID Register



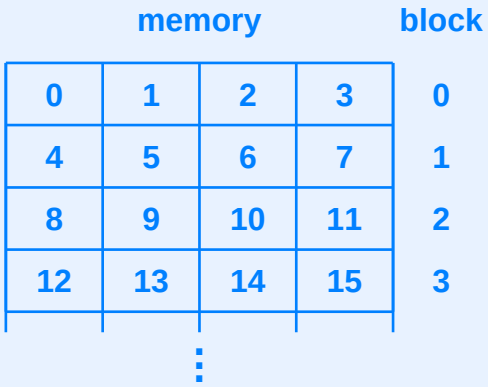
Two Technologies For Memory Caching

- Direct mapping memory cache
- Set associative memory cache

Direct mapping memory cache

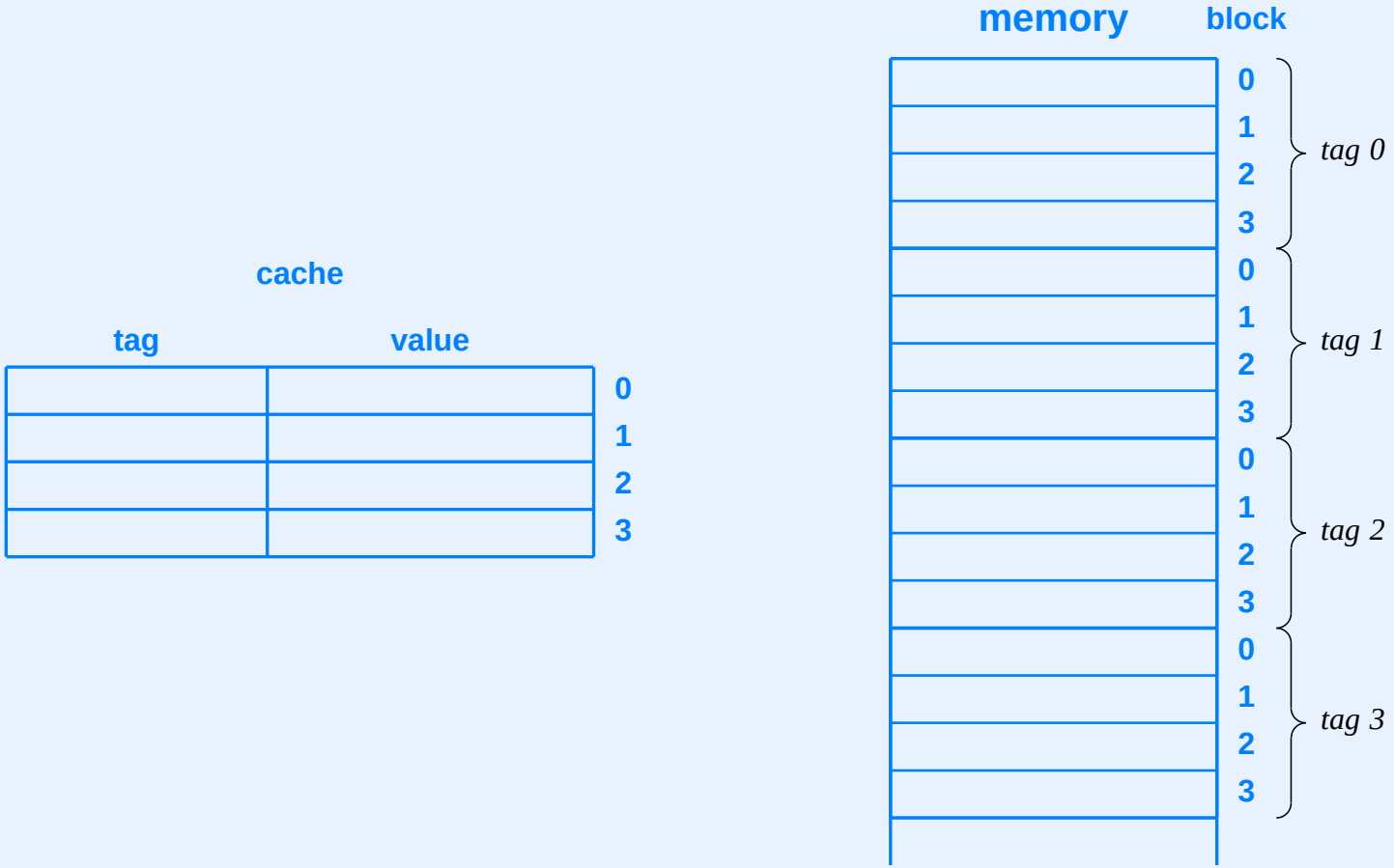
- Divides memory into numbered blocks
- Tag used to distinguish among blocks

Illustration Of Direct Mapping Cache



- Example block size is 4
- Only block numbered *i* can be placed in cache slot *i*

Illustration Of Tags



- Use of tags saves space

Using Powers Of Two

- Using powers of two means bits of address specify *tag*, *block*, and *offset*

Algorithm For Cache Lookup

Given:

A memory address

Find:

The data byte at that address

Method:

Extract the tag number, t, block number, b, and offset, o, from the address.

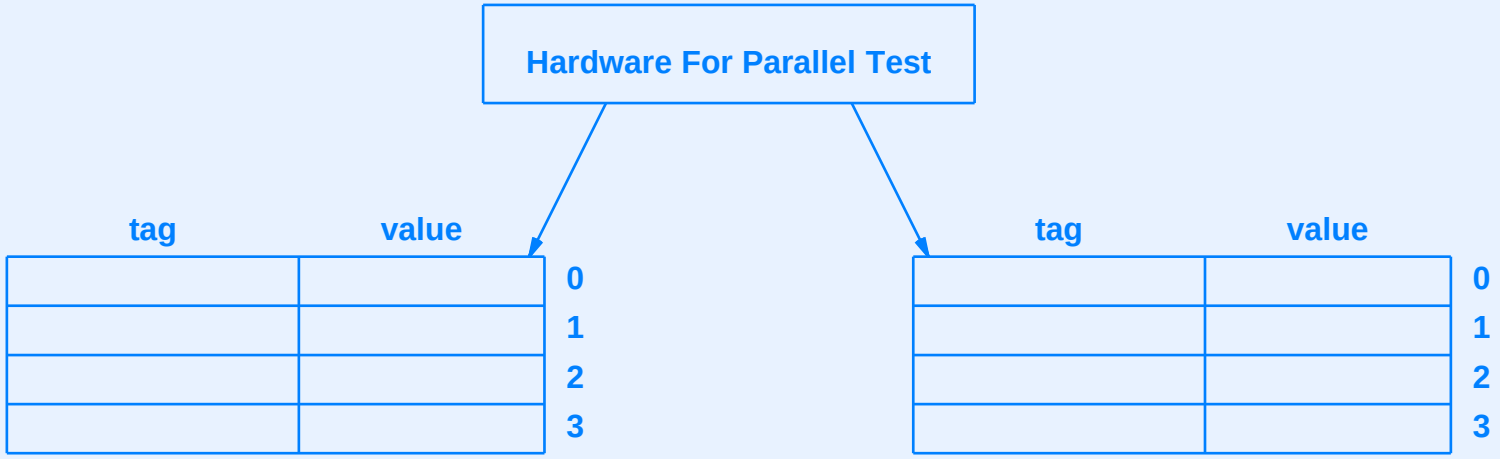
Examine the tag in slot b of the cache. If the tag matches t, extract the value from slot b of the cache.

If the tag in slot b of the cache does not match t, use the memory address to extract the block from memory, place a copy in slot b of the cache, replace the tag with t, and use o to select the appropriate byte from the value.

Set Associative Memory Cache

- Alternative to direct mapping memory cache
- Uses parallel hardware
- Maintains multiple, independent caches

Illustration Of Set Associative Cache



Advantage Of Set Associative Cache

- Assume two memory addresses A_1 and A_2
  - Both have block number zero
  - Have different tags
- In direct mapped cache
  - A_1 and A_2 contend for single slot
  - Only one can be cached at a given time
- In set associative cache
  - A_1 and A_2 can be placed in separate caches
  - Both can be cached at a given time

Fully Associative Cache

- Many parallel caches
- Each cache has exactly one slot
- Slot can hold arbitrary item

Continuum Of Caches

- No parallelism corresponds to direct mapped cache
- Some parallelism corresponds to set associative cache
- Full parallelism corresponds to Content Addressable Memory

Consequences For Programmers

- In many programs caching works well without extra work
- To optimize cache performance perform all operations on one data item before moving to another data item

Summary

- Caching is fundamental optimization technique
- Cache intercepts requests, automatically stores values, and answers requests quickly, whenever possible
- Caching can be used with both physical and virtual memory addresses

Summary
(continued)

- Memory cache uses hierarchy
  - L1 onboard processor
  - L2 between processor and memory
  - L3 built into memory
- Two technologies used for memory cache
  - Direct mapped
  - Set associative


Questions?