FAST Data Accesses in DRAM

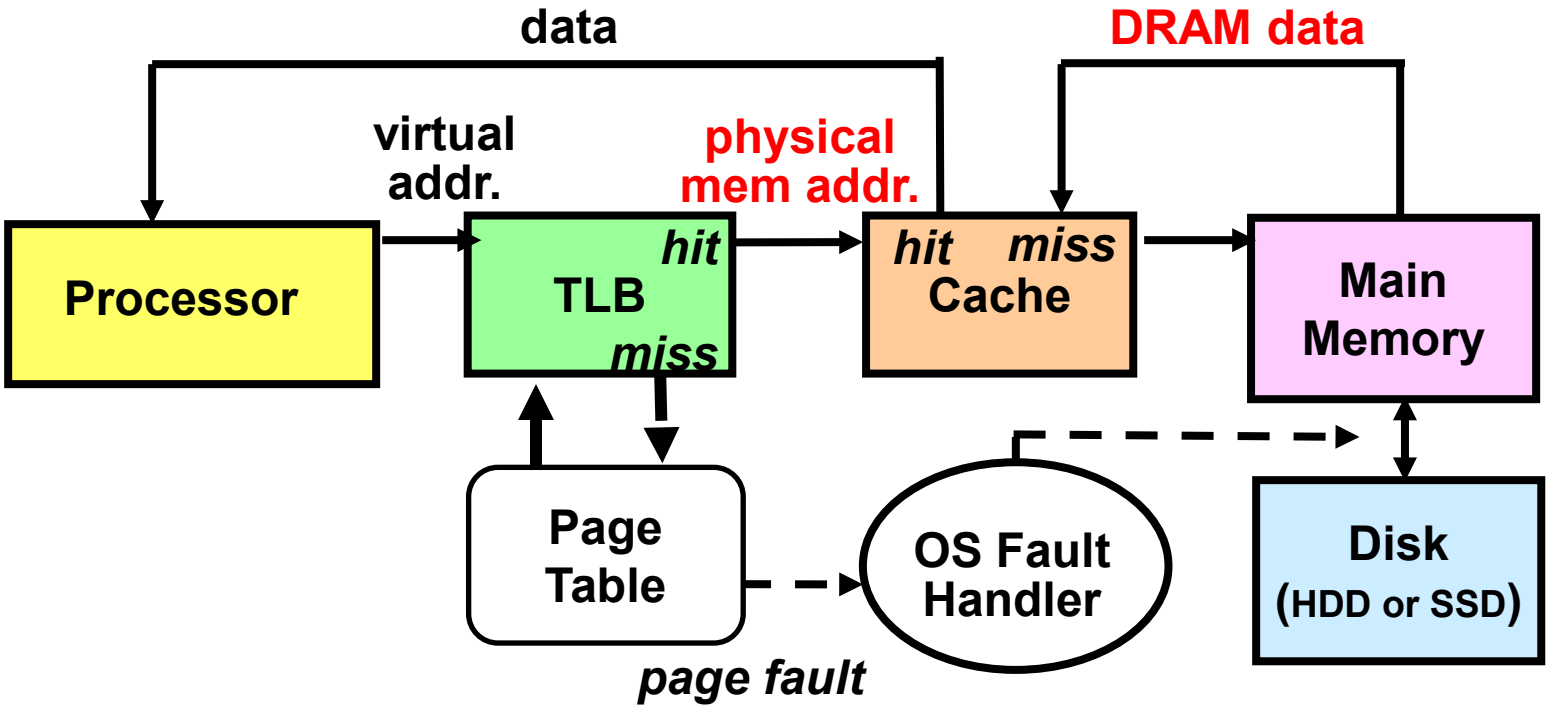
Xiaodong Zhang

CSE3421 Additional notes in Memory Systems

A review of data accesses in caches and memory

What happens if a page/block access misses in the cache?

We need to load page/block from DRAM to cache



About me

- At Ohio State since 2006
- I was the Department Chair from 2006-2018.
- Research in Data Management in Computer Systems
  - Architecture, OS, parallel processing, and databases
- My architecture research focusing on memory systems
- Three results of mine are in general-purpose processors
  - Multicolumn cache (1997), a low latency highly associative cache
  - Permutation interleaving (2000) in all memory controllers
  - Cache partitioning (2008) in multicores in Intel processors

DRAM History and basics

DRAM was patented at IBM by Robert Dennard in 1968

(RAM = random-access memory)

- It is low cost and much cheaper than SRAM (for cache)
  - 1 transistor and 1 capacitor vs 6 transistors for each cell
  - DRAM cell will lose its content in the capacitor timely
  - Each cell is charged periodically, this is why called D=dynamic
  - SRAM is "static", once it is written, it maintains the value

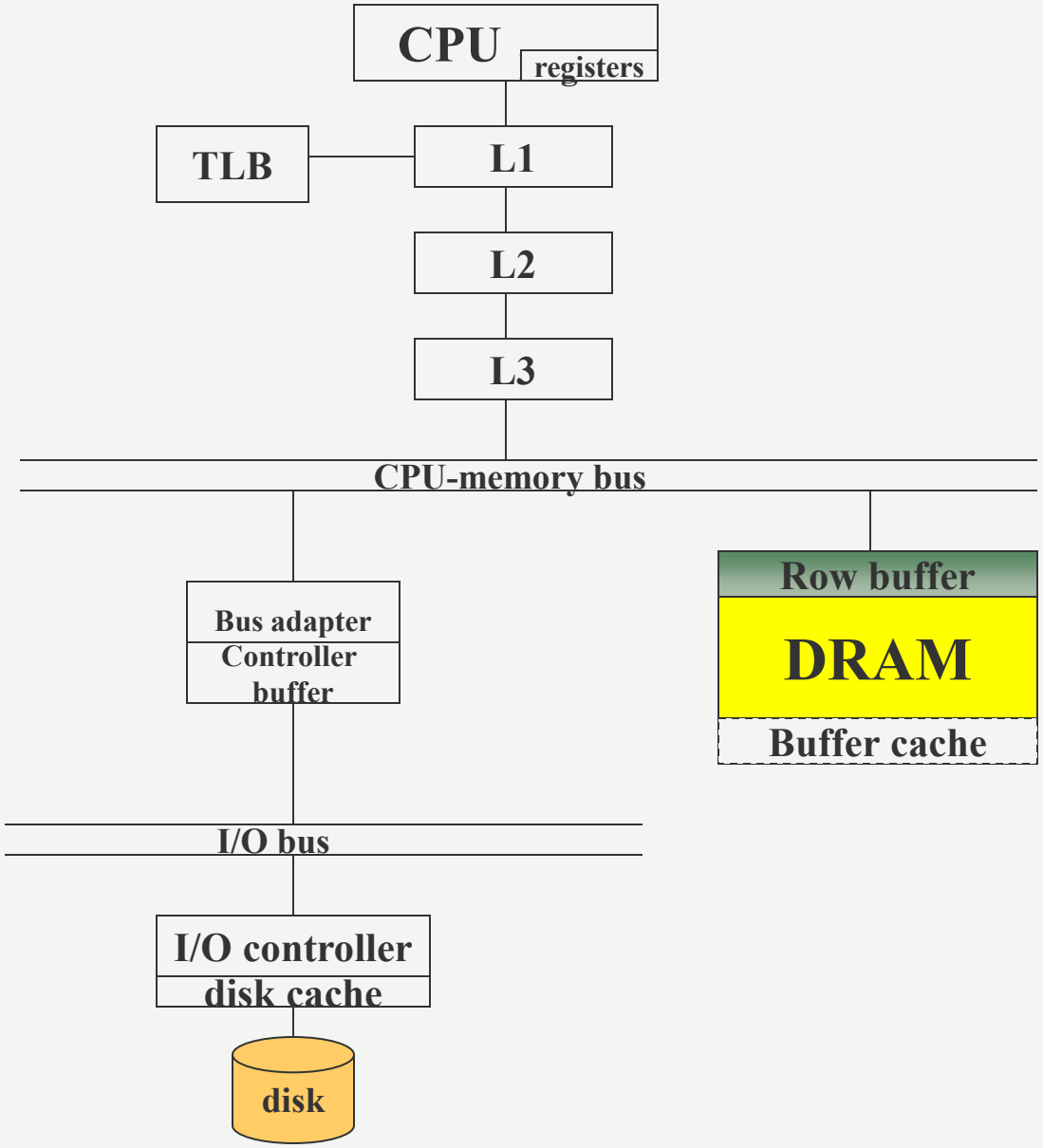
- DRAM is much slower than SRAM
  - SRAM is used for on-chip cache, DRAM is off-chip

Where is Locality in DRAM?

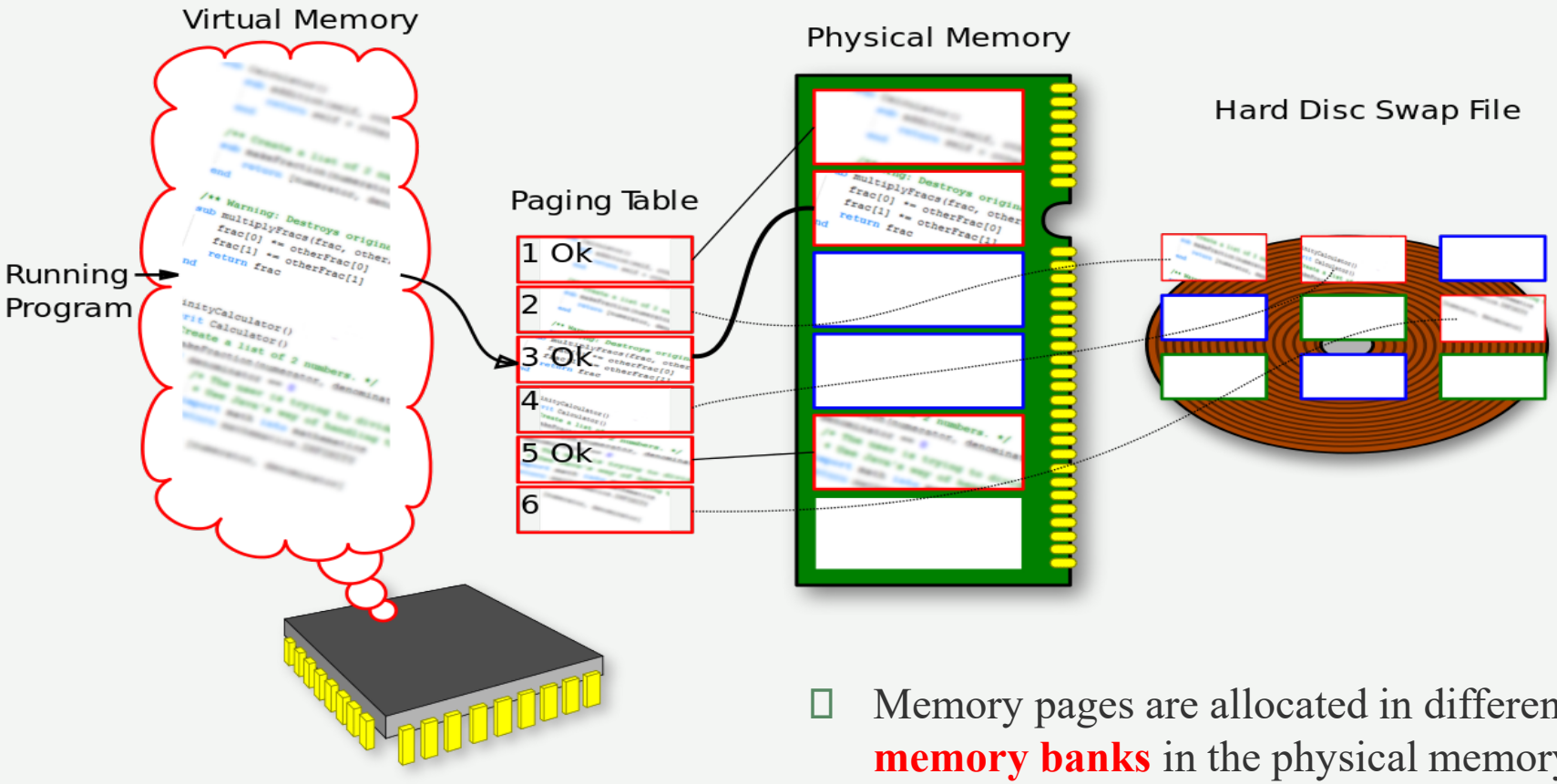
- DRAM is the center of memory hierarchy:
  - High density and high capacity
  - Low-cost fast accesses (relative to disks)
- A cache miss has been considered as a constant delay for long time. This is wrong!
  - Non-uniform access latencies exist within DRAM

- Row-buffer serves as a fast cache in DRAM
  - Its access patterns have been paid little attention.
  - Reusing buffer data minimizes the DRAM latency.

Locality Exploitation in Row Buffer

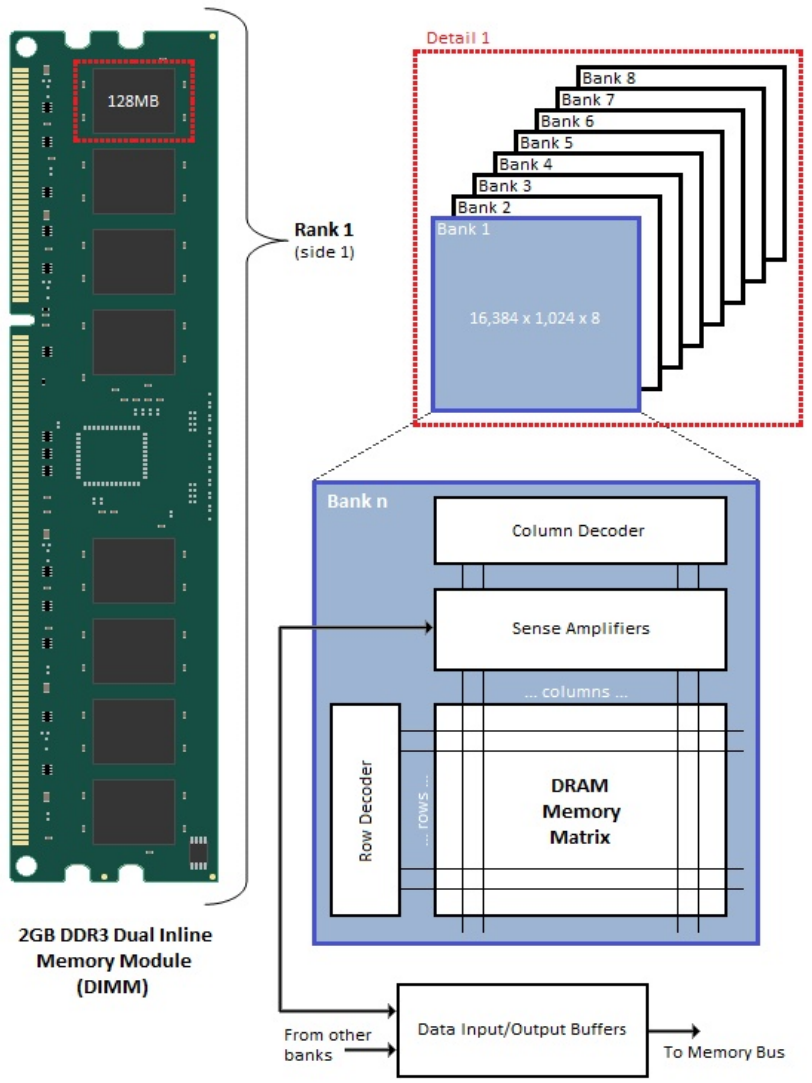


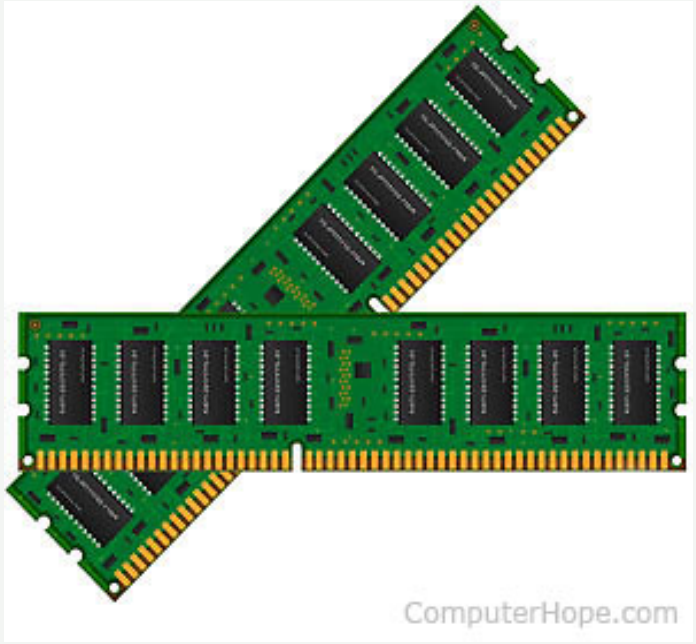
Memory pages are moving around in the hierarchy



- Memory pages are allocated in different **memory banks** in the physical memory

Memory Banks

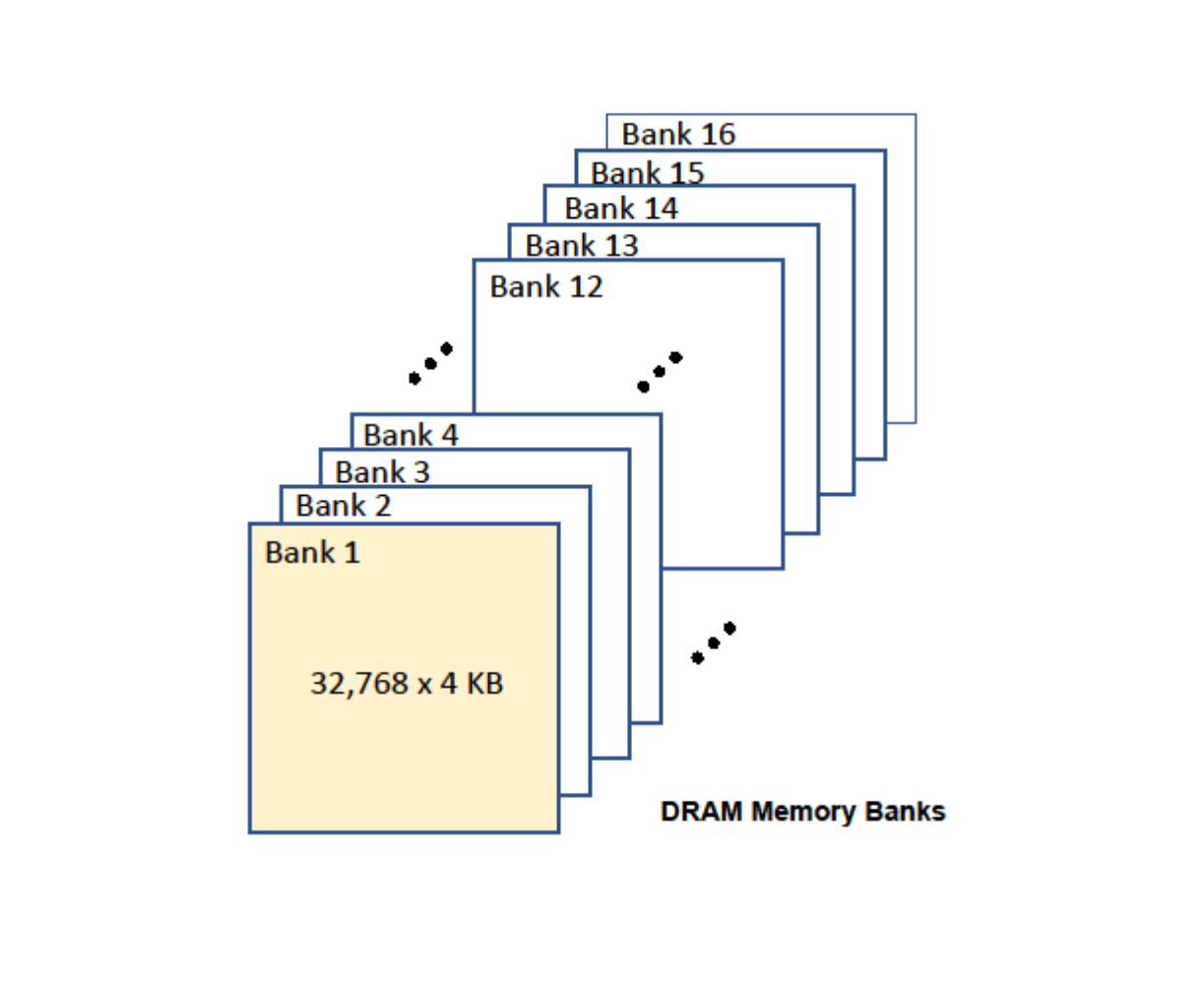




- A set of memory chips
  - DIMM: dual in-line memory module
  - SIMM: single in-line memory module

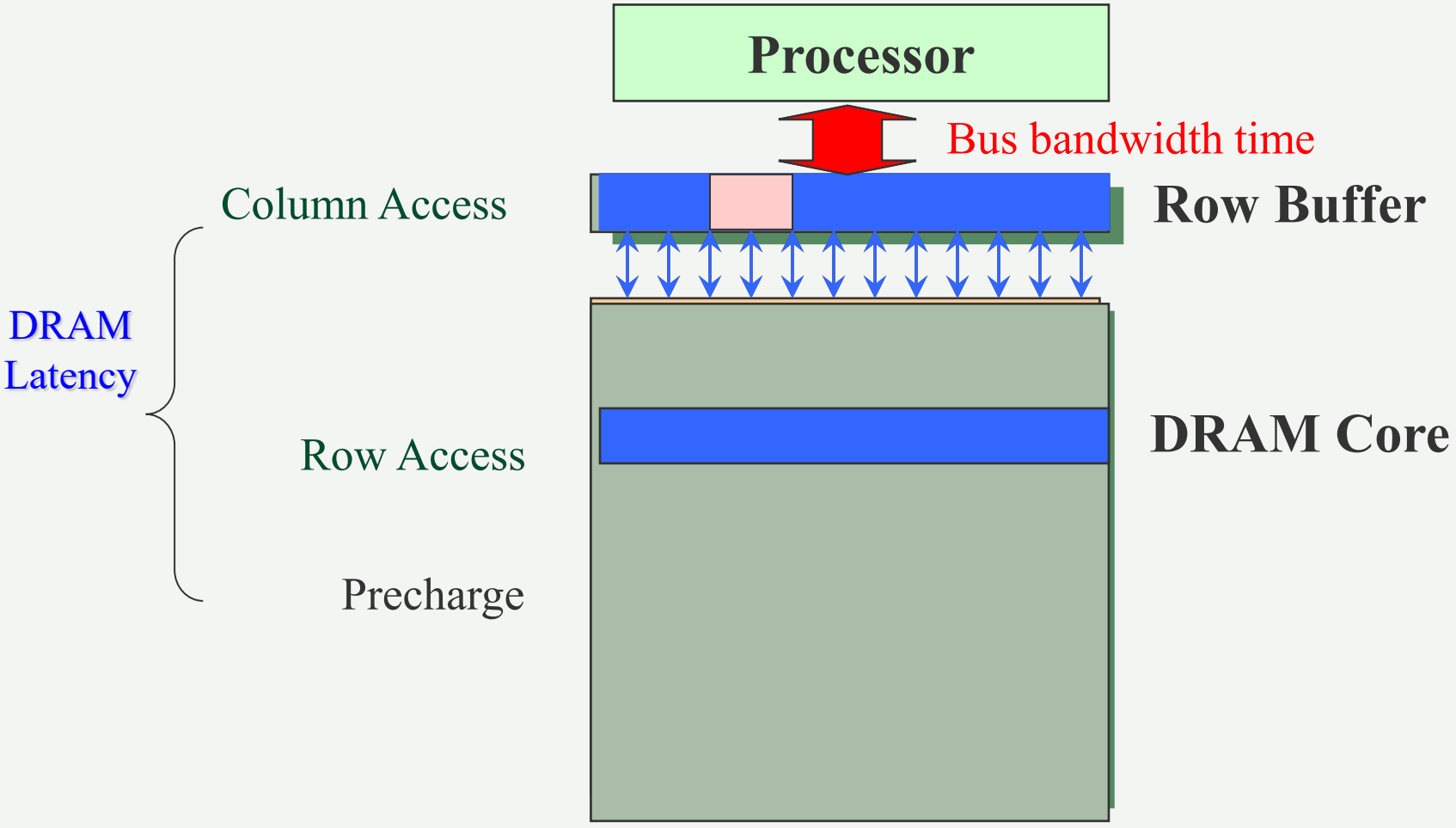
- 2GB DIMM (2 ranks of 8 128MB chips, 64-bit data path, compared with 32-bit in SIMM)
- Each rank has 8 banks, and the main memory consists of many memory banks.
- A Laptop can have as high as 256 GB memory, or 2,048 (2K) memory banks.
- Note: DDR3 = double data rate 3

An Example of Memory Banks



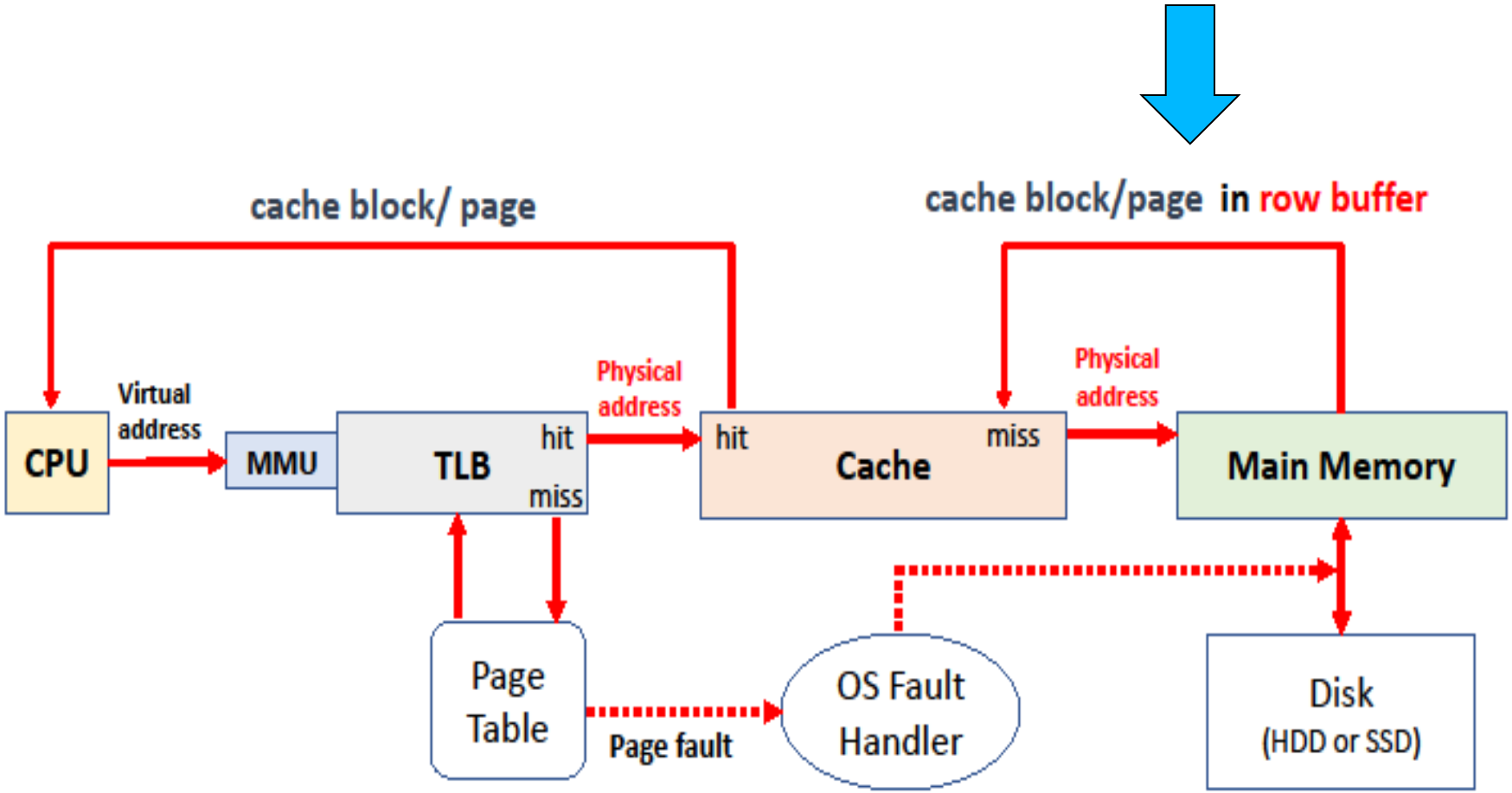
- A 2 GB memory module consisting of 16 banks, 128MB each
- Each memory bank has 32K pages, and the page size is 4KB
- Each memory bank has a row-buffer of 4KB

DRAM Access = Latency + Bandwidth Time



Row buffer misses come from a sequence of accesses to different pages in the same bank.

Access hit/miss & loading block/page from DRAM row buffer



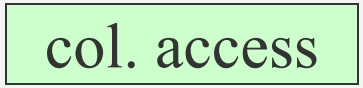
DRAM Pre-charge, Recharge, or Refresh

- Every DRAM cell (a tiny capacity and a transistor for 1 bit) must be precharged every 64 ms
- A row read/write automatically precharges the row
  - Row buffer is frequently and automatically precharged
  - Row buffer in modern DRAM is built by SRAM
- Every precharge operation covers a number of rows

- A precharge is issued by the memory controller
  - in a period of 7.8us on average due to frequent accesses to the DRAM

Nonuniform DRAM Access Latency

- Case 1: Row buffer hit (20+ ns)

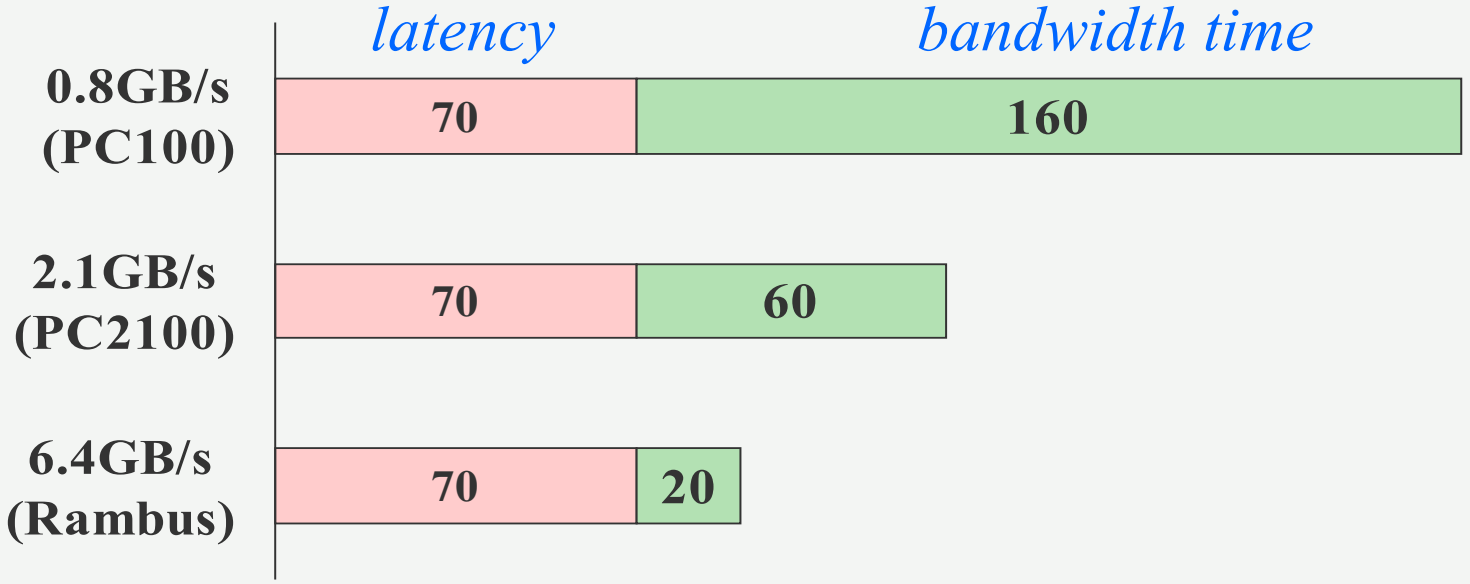


- Case 2: Row buffer miss (row is precharged, 40+ ns)

- Case 3: Row buffer miss (not precharged, ≈ 70 ns)

Amdahl's Law applies in DRAM

- Time (ns) to fetch a 128-byte cache block:



- As the bandwidth improves, DRAM latency will decide cache miss penalty.

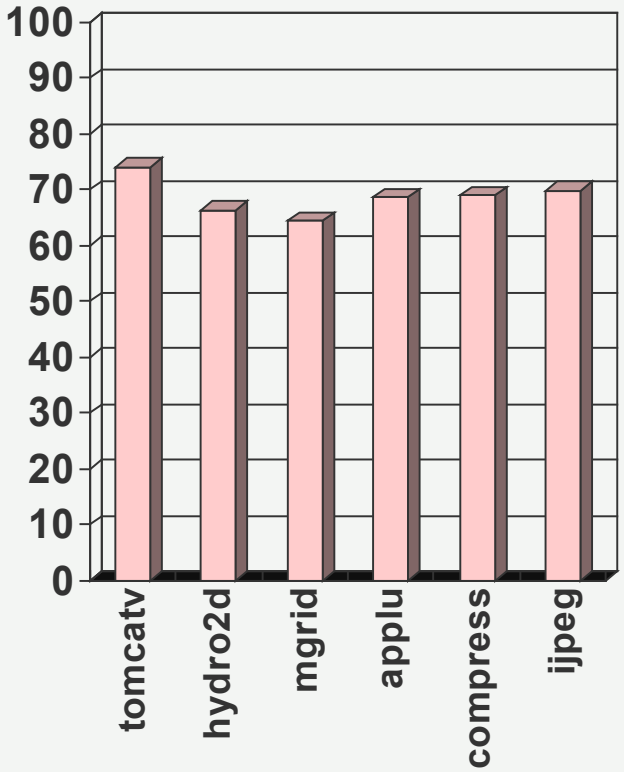
Row Buffer Locality Benefit

$$Latency_{\text{row buffer hit}} < Latency_{\text{row buffer miss}}$$

Reduce latency by up to 67%.

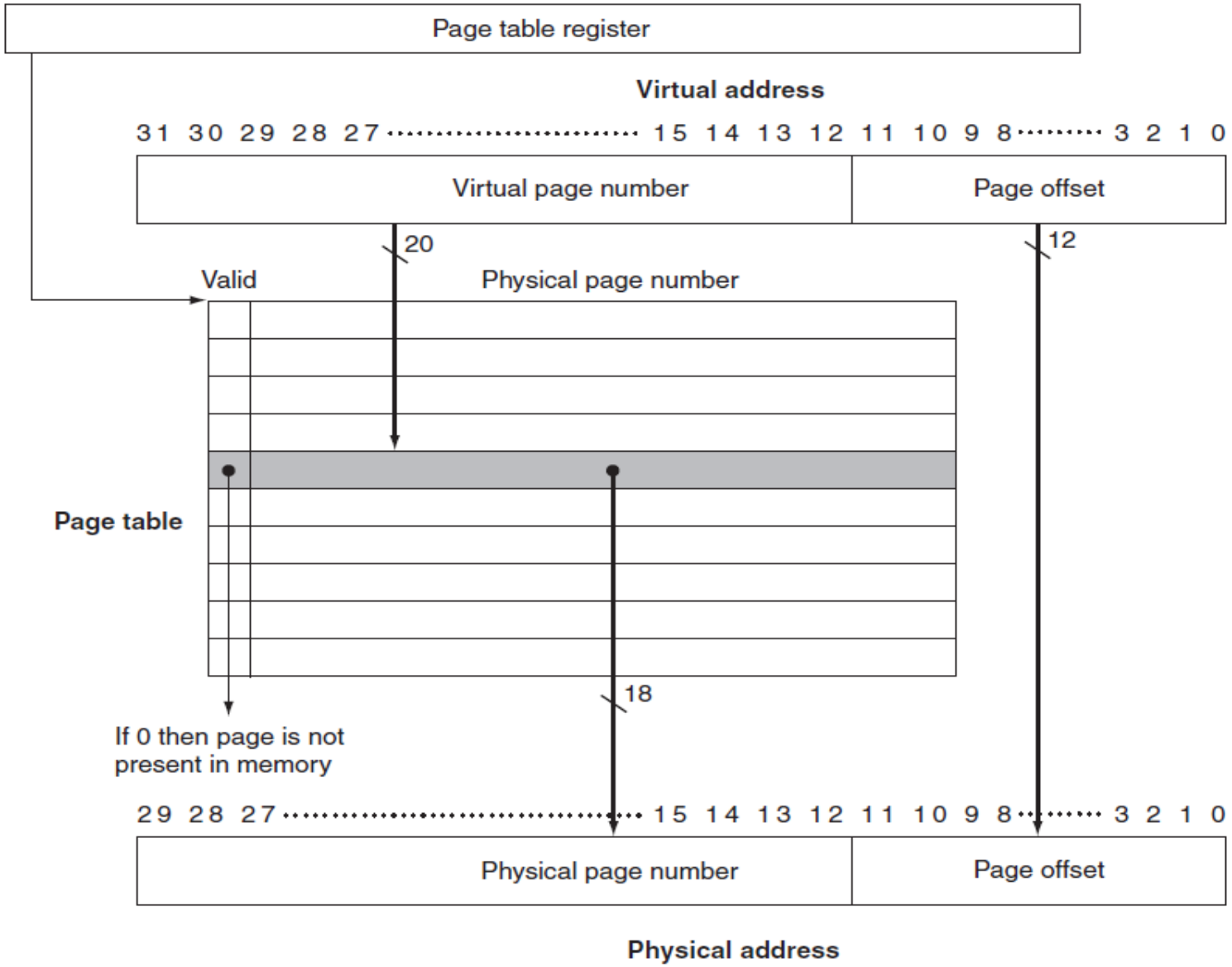
Objective: serve memory requests without accessing the DRAM core as much as possible.

Row Buffer Misses are Surprisingly High

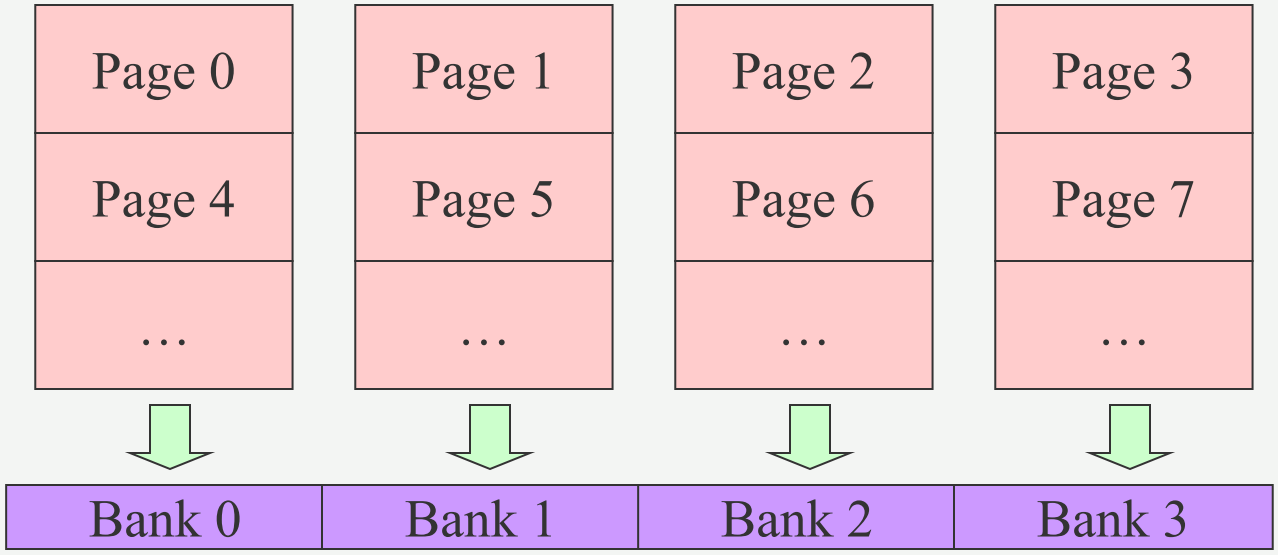


- Standard configuration
  - Conventional cache mapping
  - Page interleaving for DRAM memories
  - 32 DRAM banks, 2KB page size
  - SPEC95 and SPEC2000
- **What is the reason behind this?**

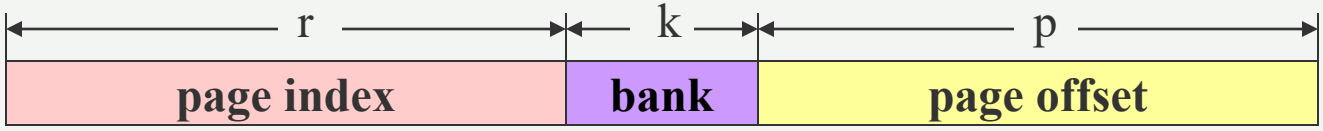
How to map physical pages among multiple memory banks?



Conventional Page Interleaving



Address format

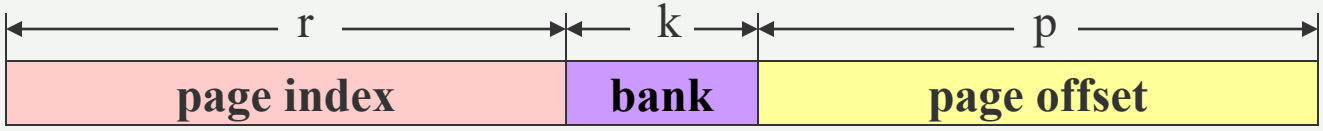


Note: the total # pages are indexed by r+k bits, and k is used which bank to go

Pages mapped to each bank in an interleaving way (see above figure)

Conventional Page Interleaving

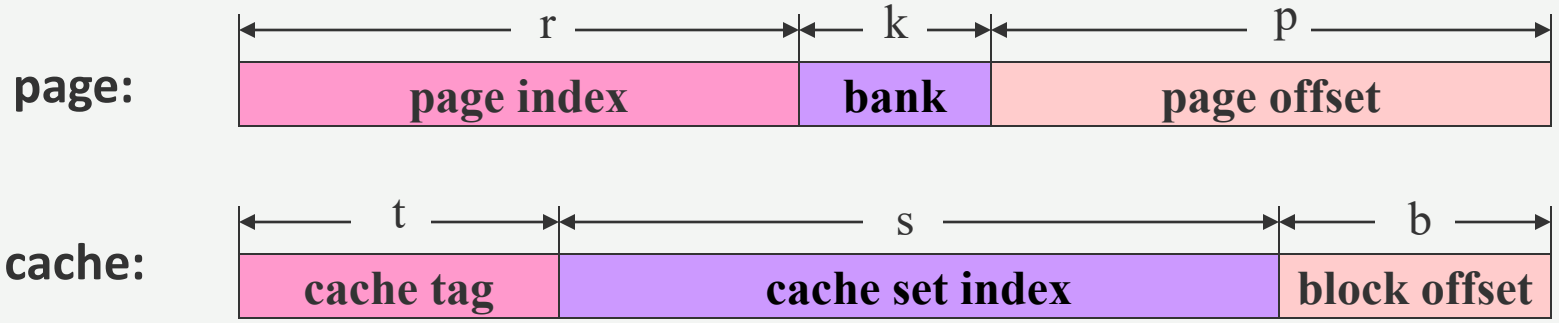
Address format



- Row buffer hits:
  - bank index is the same, page indices keep the same (same page)
  - bank index is different, page indices keep the same (concurrent accesses)

- Row buffer conflicts:
  - bank index is the same, page indices are different to keep replacing the row buffer (access different pages in the same memory bank)
  - bank index is different, page indices are different to keep replacing the buffer in different banks

Conflict Correspondence between Cache/DRAM



- cache-conflict: same cache index, different tags.
- row-buffer conflict: same bank index, different pages.
- address mapping: bank index $\subseteq$ cache set index
- Considering two addresses, x and y, and conventional mapping
- Property: $\forall x \forall y$, x and y conflict on cache $\Rightarrow$ also on row buffer.
- **There is one-to-one correspondence between cache and row buffer hits/conflicts.**

Note: $\forall x \forall y$ => for all x and for all y

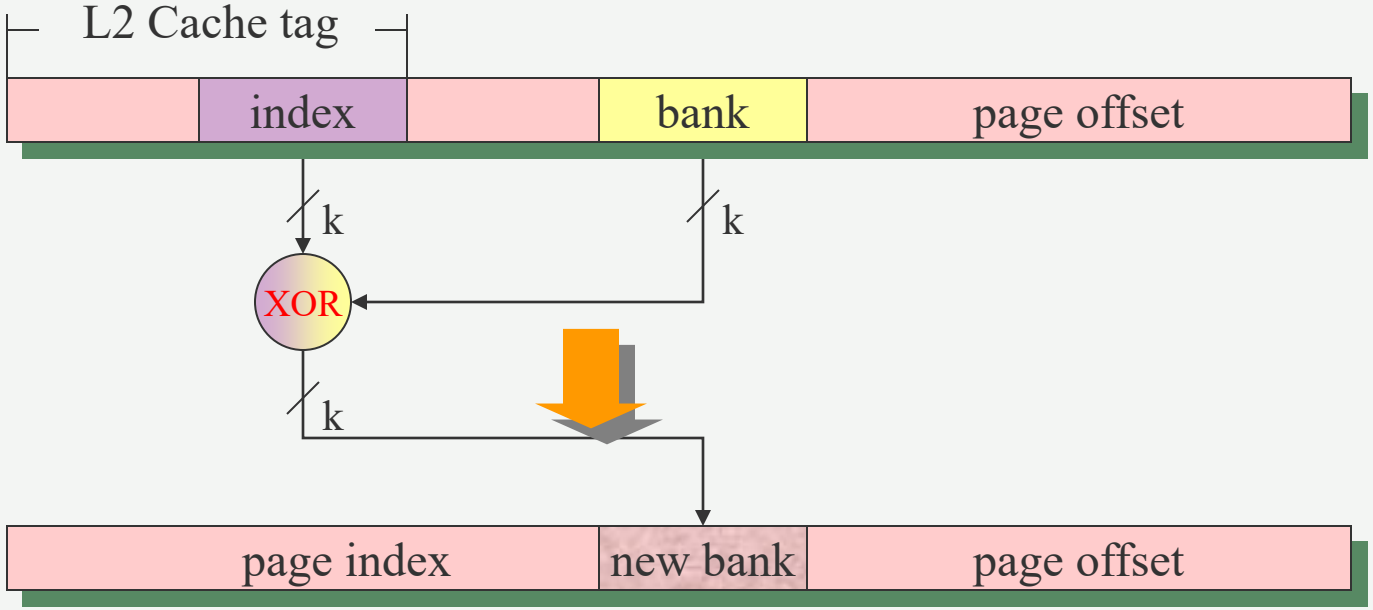
Sources of Misses

- Symmetry: invariance (of conflicts) in results under a transformation (of the conventional memory mapping).
- Address mapping symmetry propagates conflicts from the cache address pace to the memory address space:
  - Cache-conflicting addresses are also row-buffer conflicting addresses
    - Cache conflict misses are also row-buffer misses
  - Cache write-back conflicts
    - A dirty cache line needs to write its DRAM page
    - Memory controller brings that page to row buffer by evicting the existing page causing row-buffer miss on the existing page

Write-Back Conflicts

- When a cache block (dirty bit=1) is evited for cache replacement, CPU will write this cache block to the DRAM memory by
  - bringing its page to the row buffer, and the current page in the row buffer has to be replaced
  - The cache line is then taken from the row buffer for CPU to update and write back
  - The replaced page in the row buffer will have to be loaded again soon for an access, causing long latency

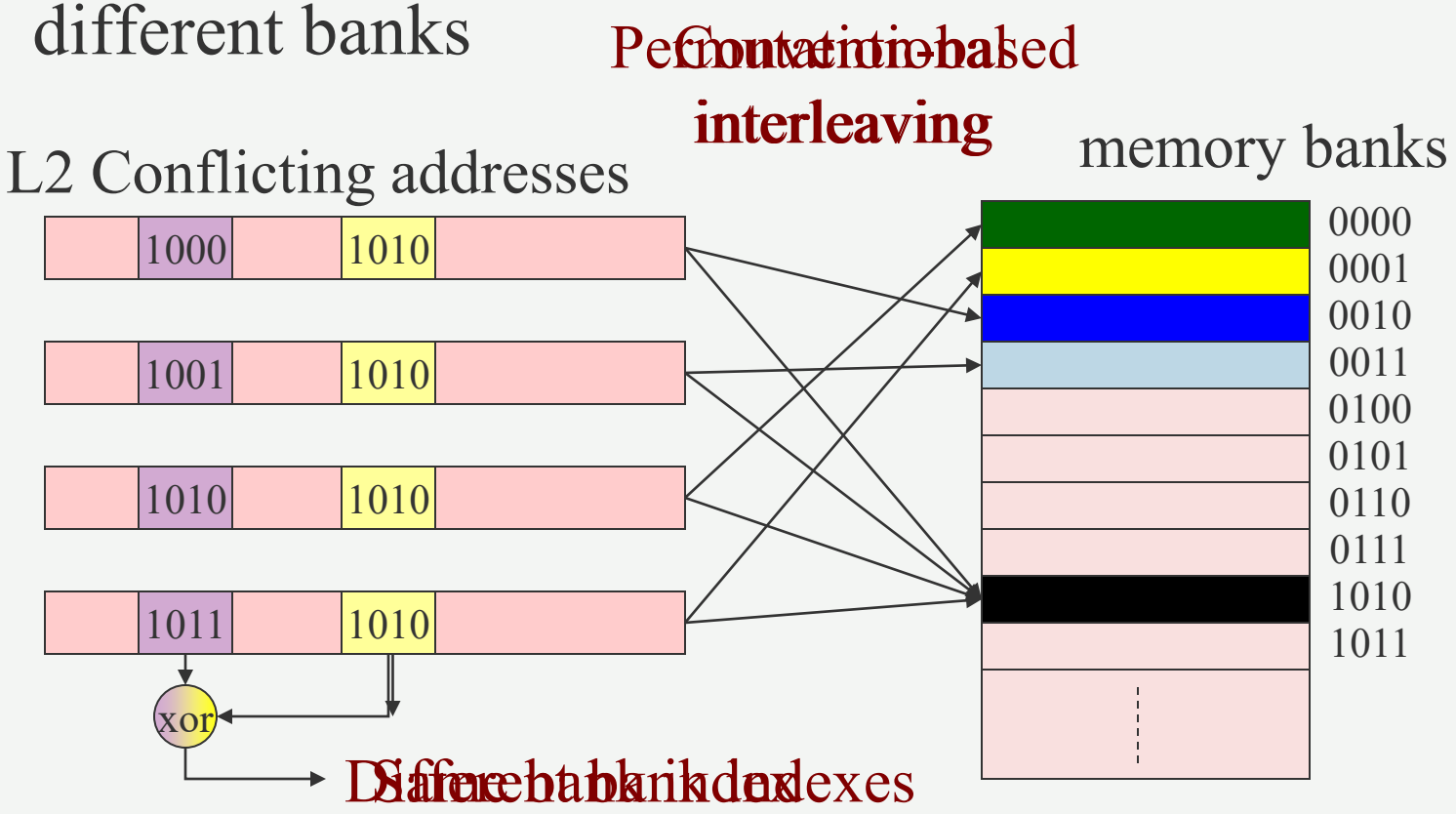
Breaking the Symmetry by Permutation-based Page Interleaving



- The k bits of the bank index xor with last order k bits in the cache tag field
  - starting from the least significant bit
- The k bit xor result forms a new bank index
- Note: memory addresses are not changed, only new bank index is generated

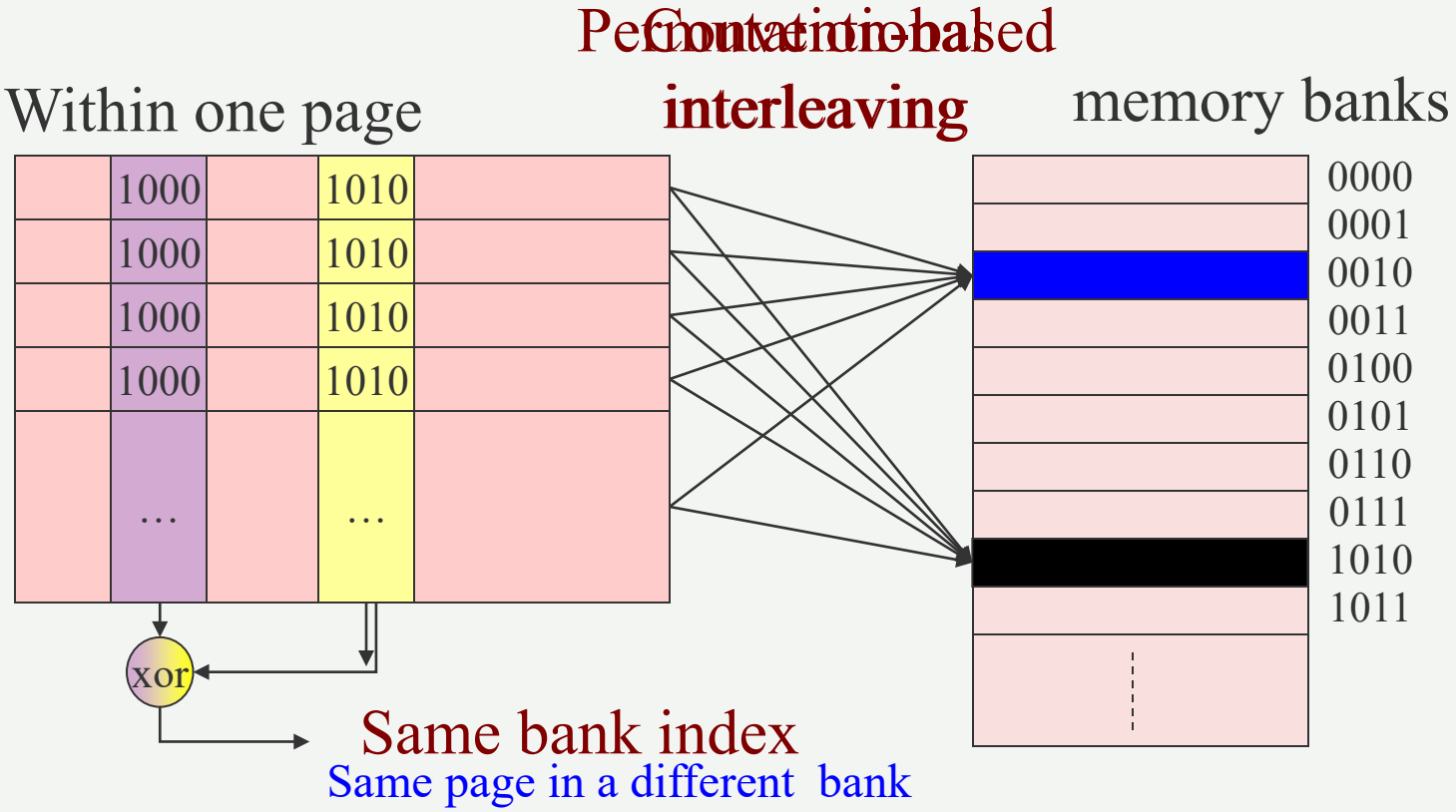
Permutation Property (1)

- Conflicting addresses are distributed onto different banks



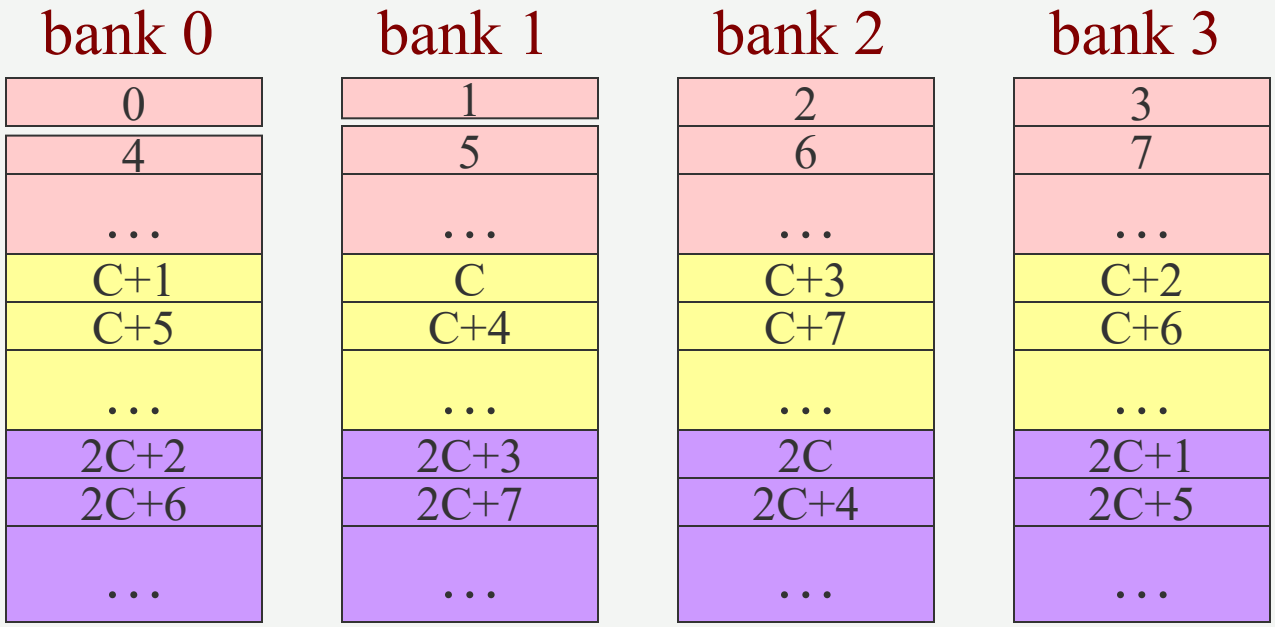
Permutation Property (2)

- Since we only change pages among banks, spatial locality of memory accesses in a page is preserved.



Permutation Property (3)

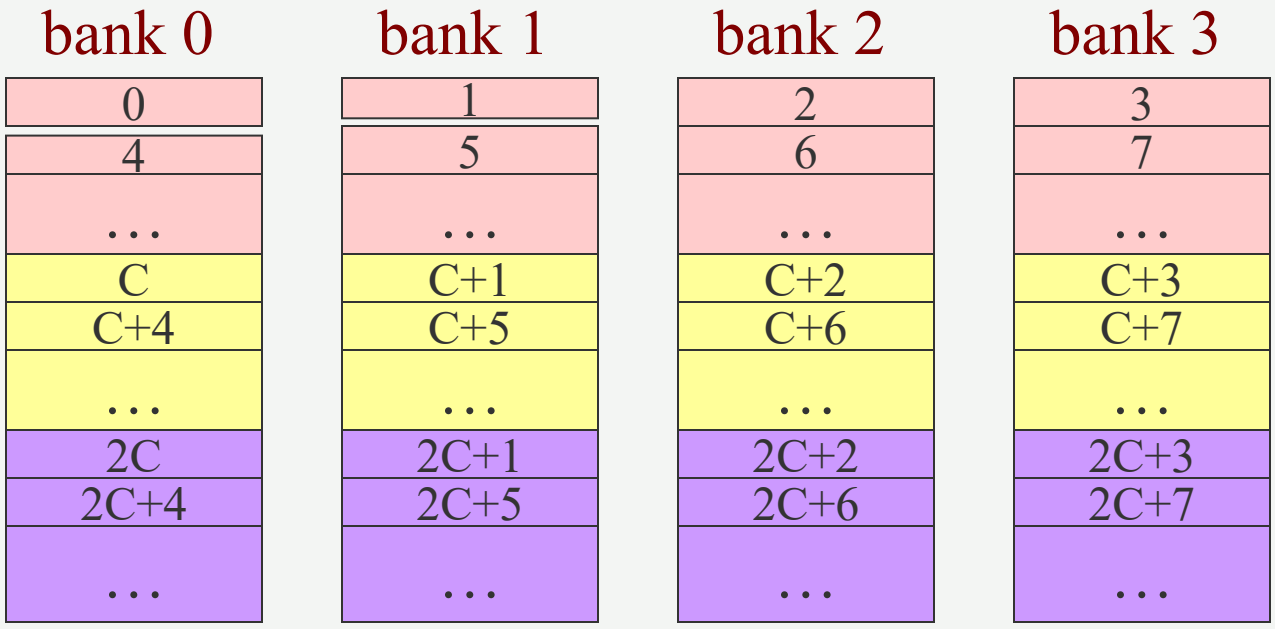
- Pages are uniformly mapped onto ALL memory banks.



- Note: C is the number pages cache can hold, page number order is in each entry.

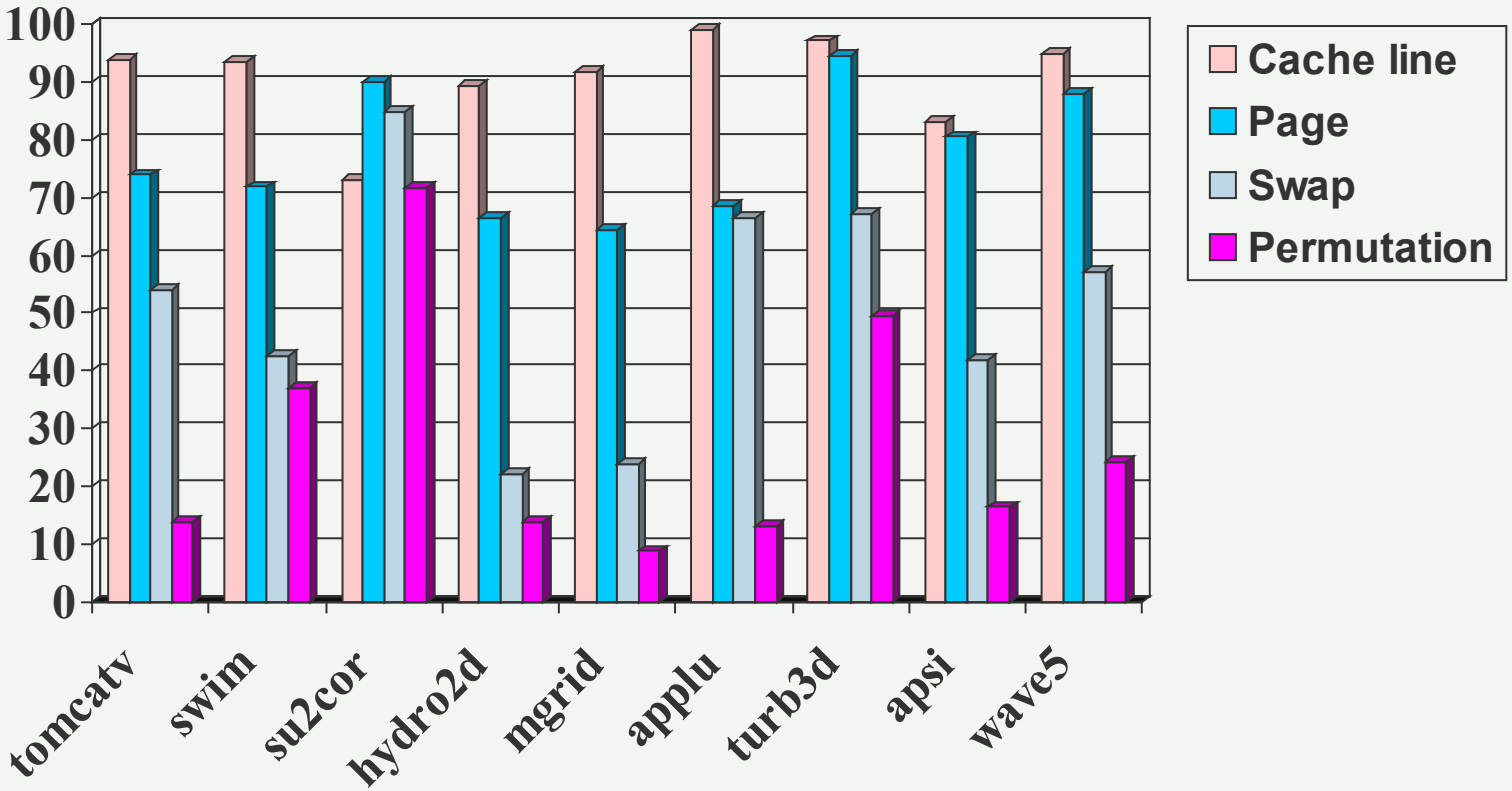
Comparing with Conventional Interleaving

- Pages are uniformly mapped onto ALL memory banks but a lot of row buffer conflicts

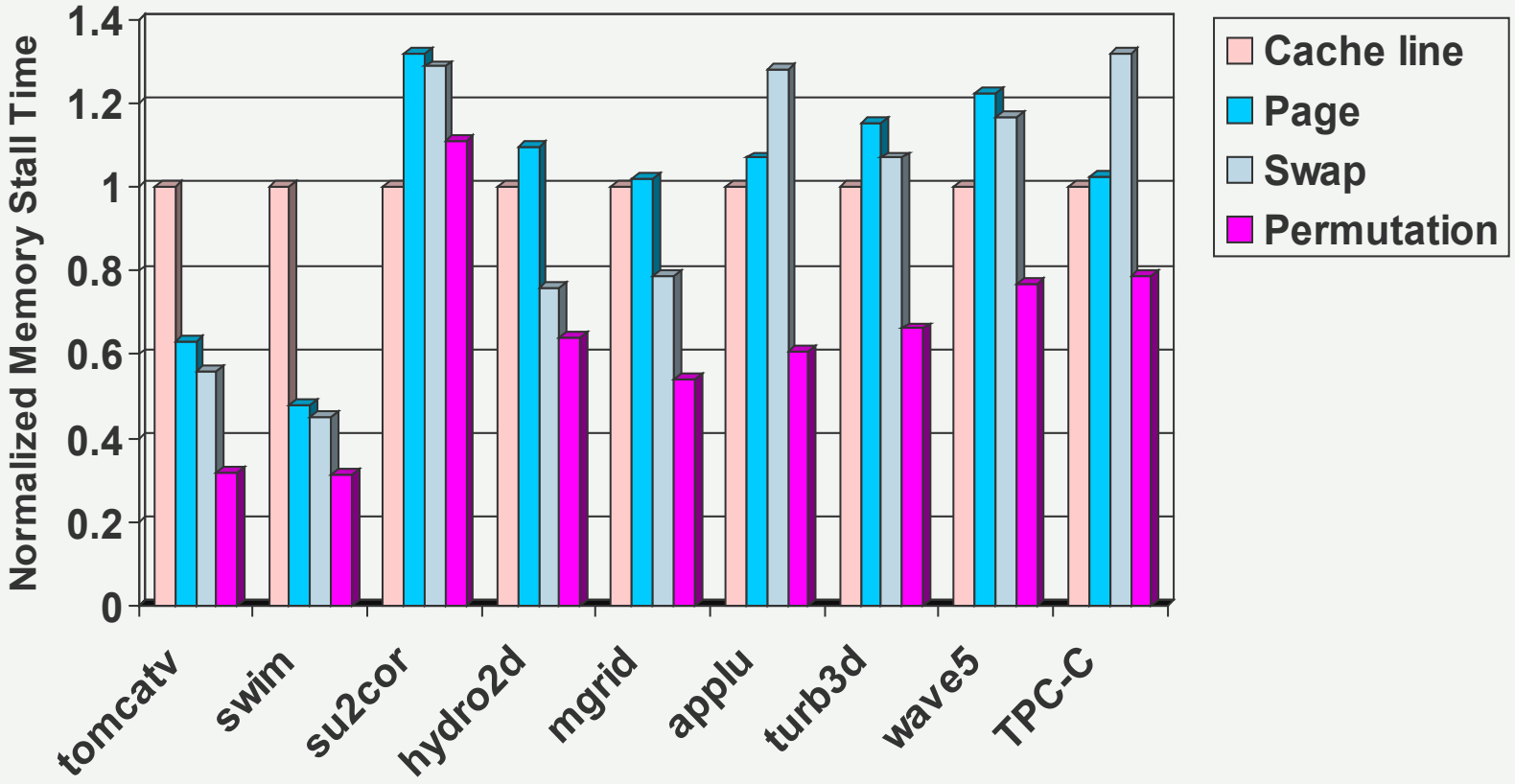


- Note: C is the number pages cache can hold, page number order is in each entry.

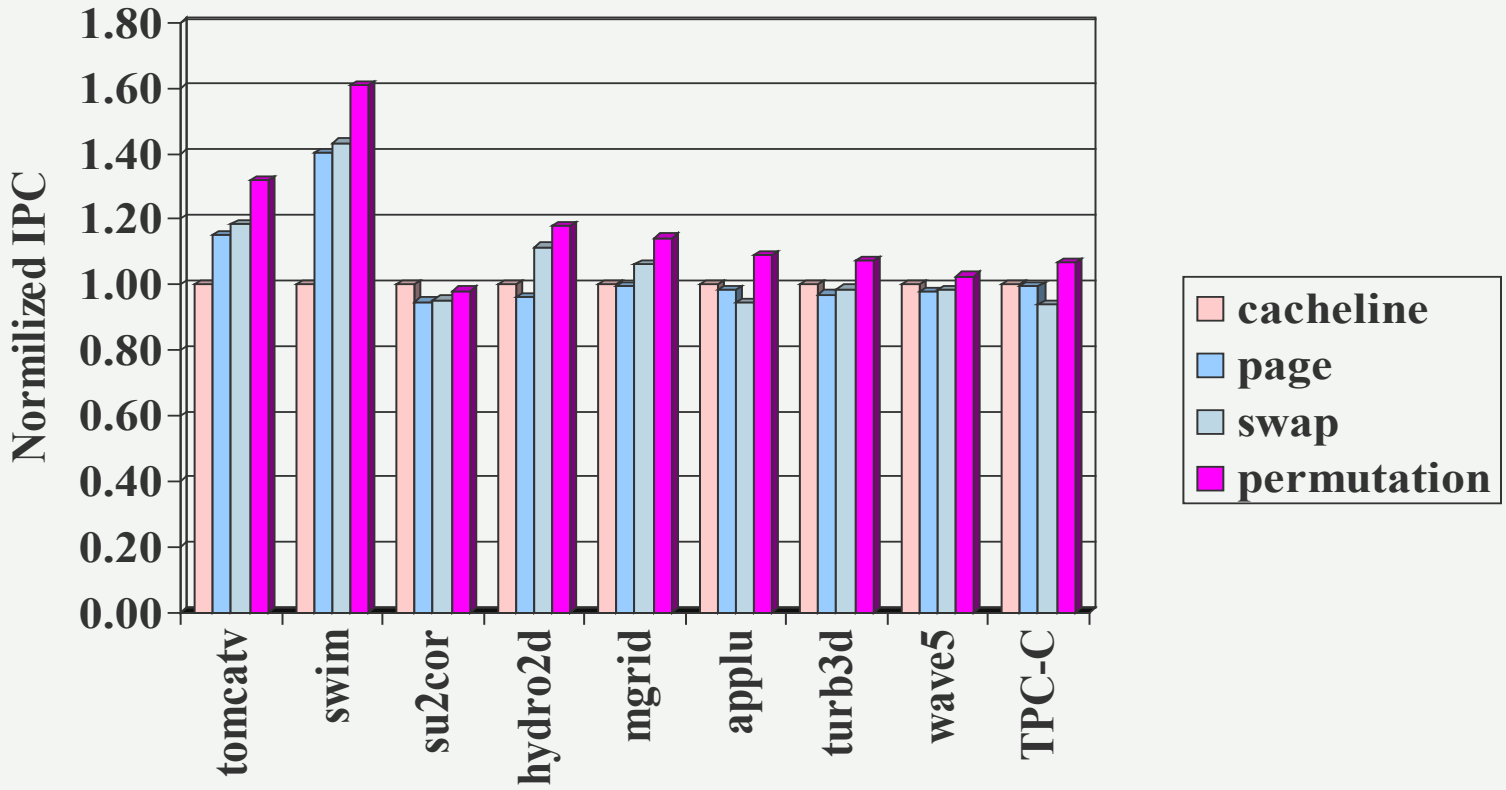
Row-buffer Miss Rates



Comparison of Memory Stall Times



Measuring IPC (#instructions per cycle)



Impact to Commercial Systems

- The method was quickly adopted in Sun UltraSPARC IIIi processor: XOR interleaving, or permutation interleaving
  - Chief architect Kevin Normoyle had intensive discussions with us for this adoption in 2001.
  - The results in the Micro-33 paper on ``conflict propagation'', and ``write-back conflicts'' are quoted in the Sun Ultra SPARC Technical Manuals.
  - Sun Microsystems has formally acknowledged our research contribution to their products.
- It is also used in Sun's Gemini dual-core processor.

What roles does UltraSPARC IIIi Play?

- UltraSPARC IIIi is a flagship chip in Sun products.
  - Up to 1.593 GHz
  - L2 cache: 1 MB on-chip, 4-way associative
  - Multiprocessor: up to 4 processors
- In a wide range of Sun computer products:
  - Sun Fire servers (V210, V240, V250, and V440 Servers)
  - Workstations and Desktops: Sun Blade 1500 series.

Acknowledgement from Sun MicroSystems

Sun Microsystems, Inc.
Mailstop UNWK20-310
7788 Gateway Boulevard, Bldg. 20
Newark, CA 94560

July 15, 2005

Jason P. McDevitt, Ph.D
Director, Technology Transfer Office
College of William and Mary
Corner House, 402 Jamestown Road
P.O. Box 8795
Williamsburg, VA 23187-8795

Dear Mr. McDevitt,

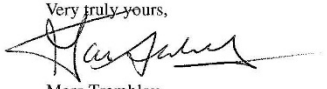
In response to your request, Sun provides the following:

Sun MicroSystems, Inc. has applied the permutation-based memory interleaving technique, called "XOR interleaving" or "permutation interleaving", as proposed by Zhao Zhang (Ph.D.'02), Zhichun Zhu (Ph.D.'03), and Xiaodong Zhang (Lettie Pate Evans Professor of Computer Science and the Department Chair) at the College of William and Mary, in the Sun UltraSPARC® IIIi processor.

A paper about this technique entitled "A permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality" was published in the 33rd Annual IEEE/ACM International Symposium on Microarchitecture (Micro-33, pp. 32-41, Monterey, California, December 10-13, 2000). A chief finding demonstrated in the report by the three researchers was that address mapping conflicts at the cache level, including address conflicts and write-back conflicts, may inevitably propagate to the DRAM memory under a standard memory interleaving method, causing significant memory access delays. The proposed permutation interleaving technique proposed a low cost solution to these conflicts problems.

This statement is an acknowledgment of Sun's use of the technique for Sun's purposes and is neither an endorsement of the technique nor recommendation of its use by others.

Very truly yours,



Marc Tremblay
Sun Fellow, Vice President & Chief Architect

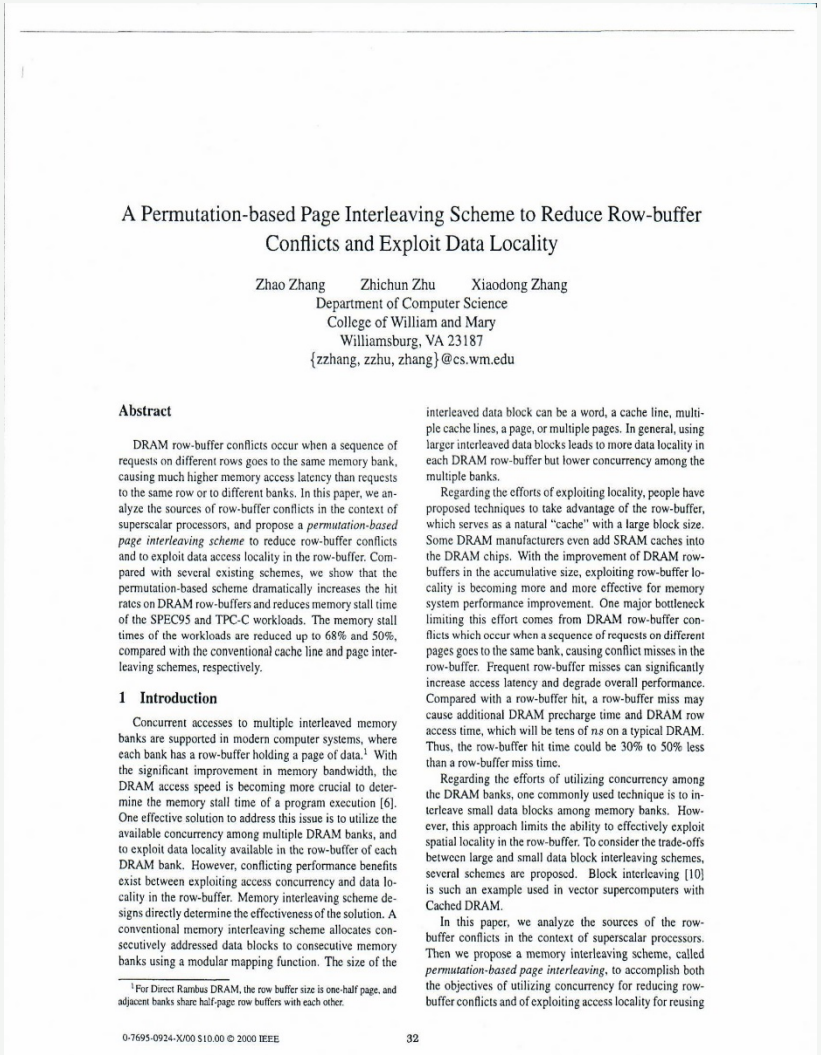
Acknowledgement from Sun MicroSystems

- Sun Microsystems, Inc. has applied the permutation-based memory interleaving technique, called ``XOR interleaving'' or ``permutation interleaving'' as proposed by Zhao Zhang (Ph.D.'02), Zhichun Zhu (Ph.D.'03), and Xiaodong Zhang (Lettie Pate Evans Professor of Computer Science and the Department Chair) at the College of William and Mary, in the Sun UltraSPARC IIIi processors.

- A paper about this technique entitled "A permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality" was published in the 33rd Annual IEEE/ACM International Symposium on Microarchitecture (Micro-33, pp. 32-41, Monterey, California, December 10-13, 2000). A chief finding demonstrated in the report by the three researchers was that address mapping conflicts at the cache level, including address conflicts and write-back conflicts, may inevitably propagate to DRAM memory under a standard memory interleaving method, causing significant memory access delays. The proposed permutation interleaving technique proposed a low cost solution to these conflict problems.

Marc Tremblay, Sun Fellow, Vice President & Chief Architect

Permutation Interleaving has been adopted in many CPU chips



A Permutation-based Page Interleaving Scheme to Reduce Row-buffer Conflicts and Exploit Data Locality

Zhao Zhang Zhichun Zhu Xiaodong Zhang
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187
{zzhang, zzhu, zhang}@cs.wm.edu

Abstract

DRAM row-buffer conflicts occur when a sequence of requests on different rows goes to the same memory bank, causing much higher memory access latency than requests to the same row or to different banks. In this paper, we analyze the sources of row-buffer conflicts in the context of superscalar processors, and propose a *permutation-based page interleaving scheme* to reduce row-buffer conflicts and to exploit data access locality in the row-buffer. Compared with several existing schemes, we show that the permutation-based scheme dramatically increases the hit rates on DRAM row-buffers and reduces memory stall time of the SPEC95 and TPC-C workloads. The memory stall times of the workloads are reduced up to 68% and 50%, compared with the conventional cache line and page interleaving schemes, respectively.

1 Introduction

Concurrent accesses to multiple interleaved memory banks are supported in modern computer systems, where each bank has a row-buffer holding a page of data.[1] With the significant improvement in memory bandwidth, the DRAM access speed is becoming more crucial to determine the memory stall time of a program execution [6]. One effective solution to address this issue is to utilize the available concurrency among multiple DRAM banks, and to exploit data locality available in the row-buffer of each DRAM bank. However, conflicting performance benefits exist between exploiting access concurrency and data locality in the row-buffer. Memory interleaving scheme designs directly determine the effectiveness of the solution. A conventional memory interleaving scheme allocates consecutively addressed data blocks to consecutive memory banks using a modular mapping function. The size of the interleaved data block can be a word, a cache line, multiple cache lines, a page, or multiple pages. In general, using larger interleaved data blocks leads to more data locality in each DRAM row-buffer but lower concurrency among the multiple banks.

Regarding the efforts of exploiting locality, people have proposed techniques to take advantage of the row-buffer, which serves as a natural "cache" with a large block size. Some DRAM manufacturers even add SRAM caches into the DRAM chips. With the improvement of DRAM row-buffers in the accumulative size, exploiting row-buffer locality is becoming more and more effective for memory system performance improvement. One major bottleneck limiting this effort comes from DRAM row-buffer conflicts which occur when a sequence of requests on different pages goes to the same bank, causing conflict misses in the row-buffer. Frequent row-buffer misses can significantly increase access latency and degrade overall performance. Compared with a row-buffer hit, a row-buffer miss may cause additional DRAM precharge time and DRAM row access time, which will be tens of *ns* on a typical DRAM. Thus, the row-buffer hit time could be 30% to 50% less than a row-buffer miss time.

Regarding the efforts of utilizing concurrency among the DRAM banks, one commonly used technique is to interleave small data blocks among memory banks. However, this approach limits the ability to effectively exploit spatial locality in the row-buffer. To consider the trade-offs between large and small data block interleaving schemes, several schemes are proposed. Block interleaving [10] is such an example used in vector supercomputers with Cached DRAM.

In this paper, we analyze the sources of the row-buffer conflicts in the context of superscalar processors. Then we propose a memory interleaving scheme, called *permutation-based page interleaving*, to accomplish both the objectives of utilizing concurrency for reducing row-buffer conflicts and of exploiting access locality for reusing

[1] For Direct Rambus DRAM, the row buffer size is one-half page, and adjacent banks share half-page row buffers with each other.

0-7695-0924-X/00 $10.00 © 2000 IEEE

- No patent has been filed for permutation interleaving, it is free
- Memory controllers in AMD, Intel, NVIDIA, and others have adopted the permutation interleaving
- Permutation interleaving is in architecture textbooks:
  - *Microprocessor Architecture*, J-L Baer
  - *Memory Systems*, B. Jacob et. al.
  - *Memory controller Design*, B. Akesson and K. Goossens
  - …

The permutation method is found everywhere now

- **AMD** Geode GX, LX processors
  - X86-compatible processor, widely used in the embedded computing market
- **Intel** Core i7 processors
  - Widely used in desktops and laptops for billions of users
- **Intel** Mobile 945 Express Chipset Family
  - Widely used in graphics and gaming markets
- **NVIDIA** GPU chipset (GeForce 7025 + nForce 630a)
  - Widely used in graphics and gaming markets

The Permutation Method is a core in all CPU Chips Today

Professor Xiaodong Zhang Receives 2020 ACM Microarchitecture Test of Time Award

Posted: January 19, 2021

CSE is delighted to announce that Professor Xiaodong Zhang's paper entitled "A permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality" (paper link: http://web.cse.ohio-state.edu/hpcs/WWW/HTML/publications/papers/TR-00-7.pdf) has won the ACM/IEEE Micro Test of Time Award. This paper was presented and published in the 33rd International Symposium on Microarchitecture in 2000 and co-authored with Zhao Zhang and Zhichun Zhu, who were then Ph.D. students at the College of William and Mary and are now Professors at University of Illinois at Chicago. According to the ACM SIGMICRO organization, "The Test of Time award is the highest honor an academic paper receives for its impact and recognizes an influential MICRO paper whose influence is still felt 18-22 years after its initial publication."

The three authors (from left to right): Xiaodong Zhang, Zhao Zhang, and Zhichun Zhu, had a brief celebration after receiving the acceptance notice of their Micro paper in Xiaodong's office at the College of William and Mary in September 2000.

This paper demonstrates that conventional memory interleaving method would propagate address-mapping conflicts at cache level to the memory address space, causing DRAM row-buffer misses. This is a serious structural issue in DRAM design, significantly lengthening memory access latency. The permutation-based interleaving method in the paper solves the problem with a trivial microarchitecture cost. Sun MicroSystems adopted the permutation method in 2001. Today, this patent-free method can be found in almost all commercial microprocessors, such as AMD, Intel, and NVIDIA, for embedded systems, laptops, desktops, and enterprise servers. The findings and the method are included in several Computer Architecture textbooks.

The citation for this Test of Time Award reads *"The method proposed in this seminal paper has had a significant impact in modern systems. For example, Sun Microsystems adopted the method, and many mainstream commercial processors use the method or variant of it".*

The Microarchitecture Test of Time Award is presented annually during the International Symposium on Microarchitecture, which is the flagship conference in the field of processor design and implementation. Since its first conference in 1968, only 21 Micro papers have been selected for this award.

"I feel highly honored to receive this award, and I am even happier for my former Ph.D. students to be recognized for their research 20 years ago," said Xiaodong Zhang. He has been an instructor of the undergraduate Computer Architecture class (CSE 3421) for several semesters. Xiaodong makes efforts to integrate research into his teaching. He said, "I introduce the permutation method in this paper when I teach DRAM memory design in the class, aiming to inspire our students to do critical and innovative thinking."