



中山大學
SUN YAT-SEN UNIVERSITY



国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

Advanced Computer Architecture

高级计算机体系结构

第5讲：Memory (2)

张献伟

xianweiz.github.io

DCS5367, 10/26/2021

作业 – HW1

- <https://xianweiz.github.io/teach/dcs5637/hws/hw1.pdf>
- 截止时间：10.31, 23:59
- 提交方式：超算习堂
 - 注册（<https://easyhpc.net/>）
 - 加入课程（<https://easyhpc.net/course/133>）
 - 作业列表：HW1

John L. Hennessy | David A. Patterson
COMPUTER
ARCHITECTURE

A Quantitative Approach

DCS5637 - 高级计算机体系结构（周二）



0人 其他

Advanced Computer Architecture, Fall 2021 课程主页：<https://xianweiz.github.io/teach/dcs5637/>

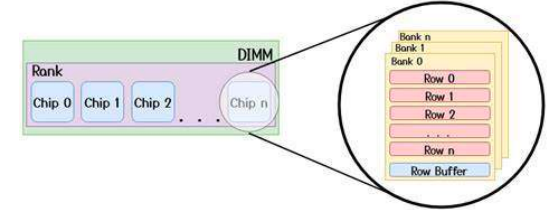
加入课程

Review Questions

- What is 'tag' in cache access?
Part of address to be used to decide the access is hit/miss.
- Cache associativity?
Cache is organized as sets, each of which contains multi blocks.
- Disadvantages of higher associativity?
Larger tags, and higher overhead on tag comparator and data mux.
- Write back and write through?
Write back first writes into cache, and then got updated into Memory when being evicted; write through directly updates mem.
- Steps of address translation?
MMU → page table lookup → page fault → retry
- Why we say DRAM is 'dynamic'?
Data is leaking, and thus dynamic refreshes are needed.

Page Mode[页模式]

- A “DRAM row” is also called a “DRAM page”
 - Usually larger than the OS page, e.g., 8KB vs. 4KB
- Row buffers act as a cache within DRAM
- Open page
 - Row buffer hit: ~20 ns access time (must only move data from row buffer to pins)
 - Row buffer conflict: ~60 ns (must first precharge the bitlines, then read new row, then move data to pins)
- Closed page
 - Empty row buffer access: ~40 ns (must first read arrays, then move data from row buffer to pins)
 - Steps
 - Activate command opens row (placed into row buffer)
 - Read/write command reads/writes column in the row buffer
 - Precharge command closes the row and prepares the bank for next access



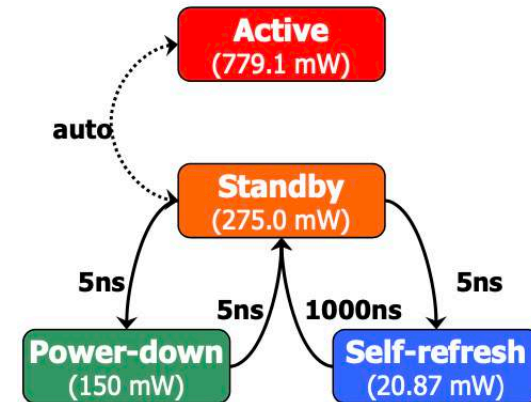
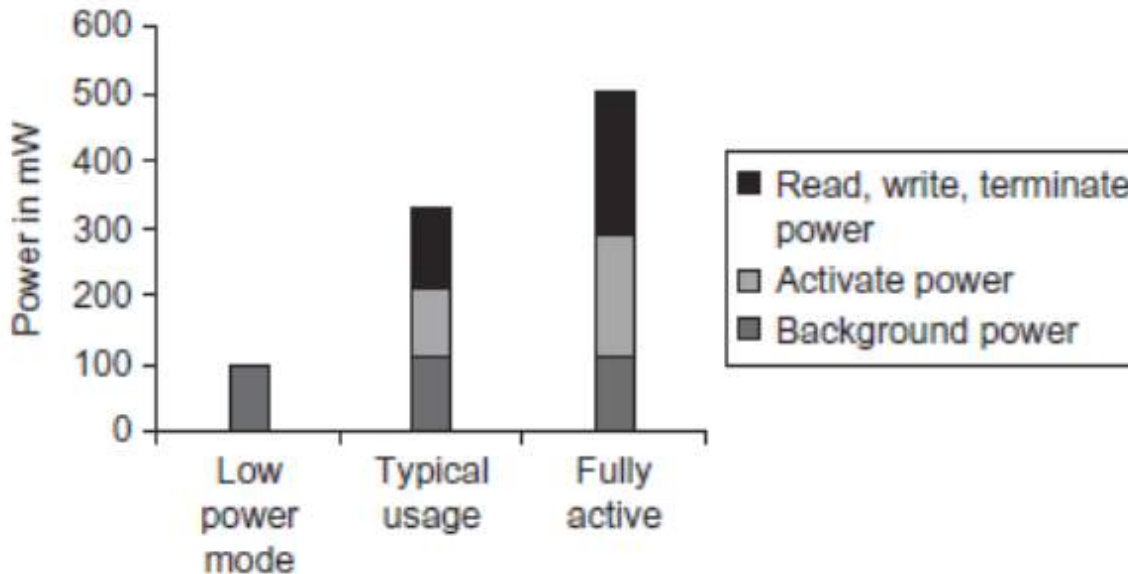
DRAM Bandwidth[带宽]

- Reading from a cell in the core array is a **very slow** process
 - DDR: Core speed = $\frac{1}{2}$ interface speed
 - DDR2/GDDR3: Core speed = $\frac{1}{4}$ interface speed
 - DDR3/GDDR4: Core speed = $\frac{1}{8}$ interface speed
 - ... likely to be worse in the future
- Calculation: *transfer_rate* * *interface_width*
 - Example: 266 MT/s * 64b = 2128 MB/s

Standard	I/O clock rate	M transfers/s	DRAM name	MiB/s/DIMM	DIMM name
DDR1	133	266	DDR266	2128	PC2100
DDR1	150	300	DDR300	2400	PC2400
DDR1	200	400	DDR400	3200	PC3200
DDR2	266	533	DDR2-533	4264	PC4300
DDR2	333	667	DDR2-667	5336	PC5300
DDR2	400	800	DDR2-800	6400	PC6400
DDR3	533	1066	DDR3-1066	8528	PC8500
DDR3	666	1333	DDR3-1333	10,664	PC10700
DDR3	800	1600	DDR3-1600	12,800	PC12800
DDR4	1333	2666	DDR4-2666	21,300	PC21300

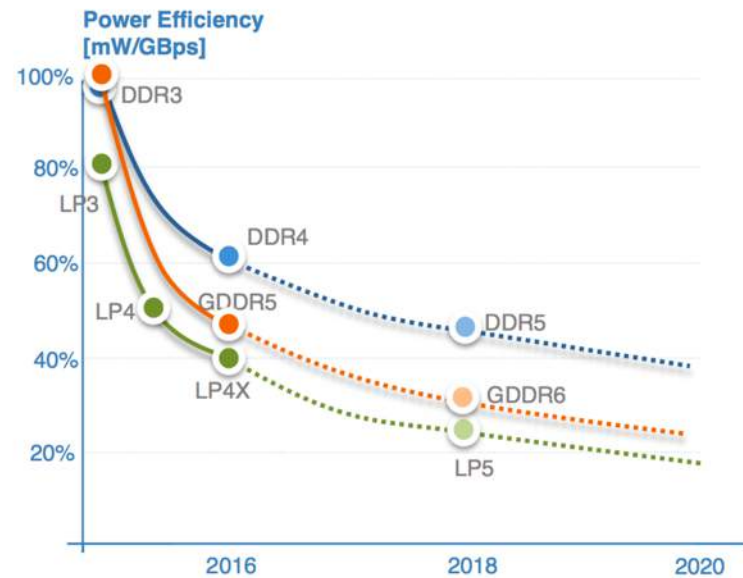
Memory Power[功耗]

- Dynamic + static[动态和静态]
 - read/write + standby
- Reduce power[降低功耗]
 - Drop operating voltage
 - Power-down mode: disable the memory, except internal automatic refresh



DRAM Variants[变种]

- DDR
 - DDR3: 1.5V, 800MHz, 64b \rightarrow 1.6G*64b = 12.8GB/s
- GDDR: graphics memory for GPUs
 - GDDR5: based on DDR3, 8Gb/s, 32b \rightarrow 8G*32b = 32GB/s
- LPDDR: low power DRAM, a.k.a., mobile memory
 - Lower voltage, narrower channel, optimized refresh



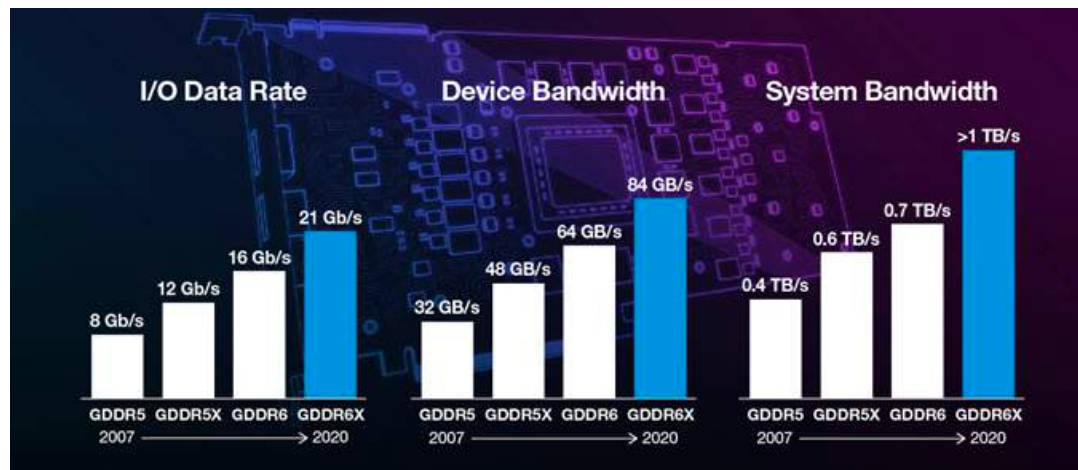
DDR5 & GDDR6

- DDR

- DDR4: 1-1.2V, 1333MHz, 64b \rightarrow 21.3GB/s x 4 = 85.2GB/s
- DDR5: 1.1V, 6.4Gbps, 64b \rightarrow 51.2GB/s x 4 = 204.8GB/s

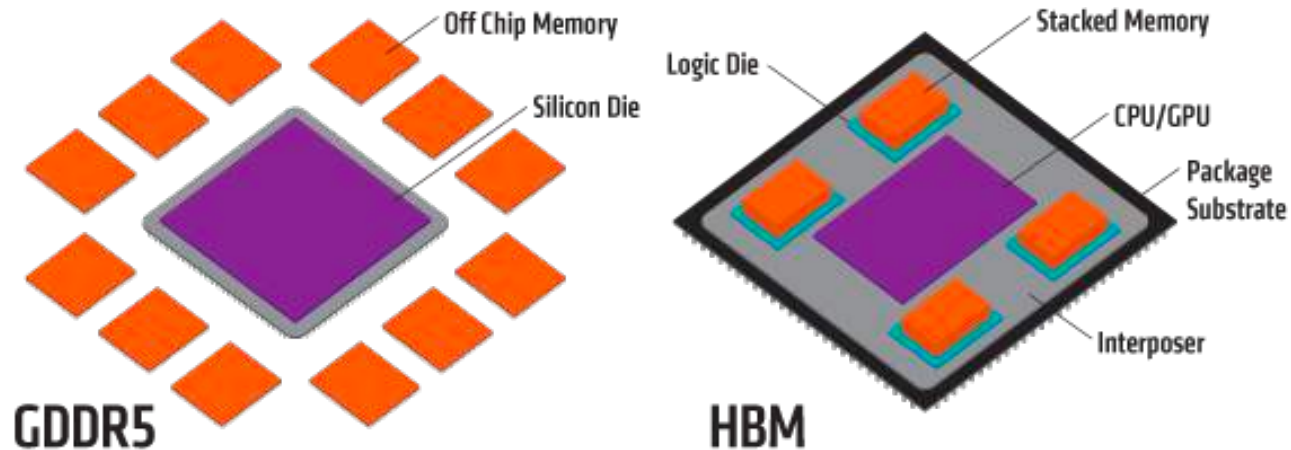
- GDDR

- GDDR5: 8Gb/s(~7), 256b, 224GB/s, 12GB, [GTX 980](#)
- GDDR5X: 12Gb/s(~10), 256b, 320GB/s, 8GB, [GTX 1080](#)
- GDDR6: 16Gb/s(~14), 256b, 448GB/s, 10GB, [RTX 2080](#)
- GDDR6X: 21Gb/s (~19), 320b, 760GB/s, 10GB, [RTX 3080](#)



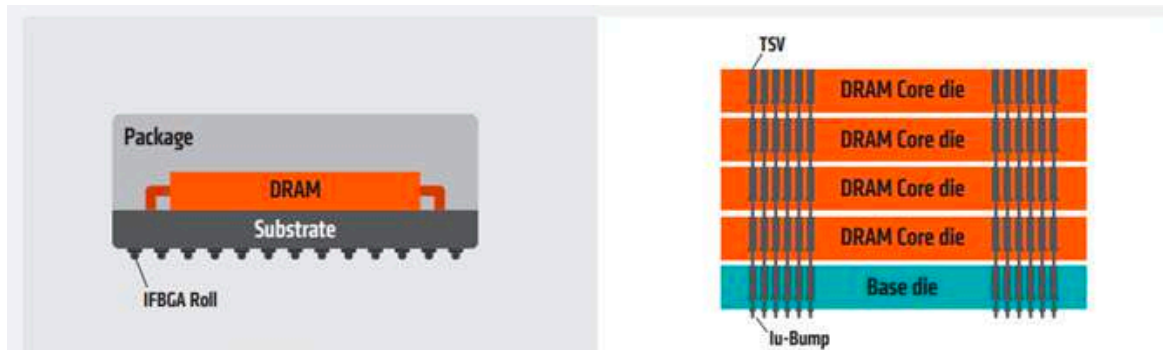
Stacked DRAMs[堆叠]

- Stacked DRAMs in same package as processor
 - High Bandwidth Memory (HBM)
- HBM consumes less power and still maintains significantly higher bandwidth in a small form factor
 - To keep the TDP target low, HBM's clock speed is limited to 1GBPs but, it makes up for it with its 4096 bits of the memory bus



HBM[高帶寬內存]

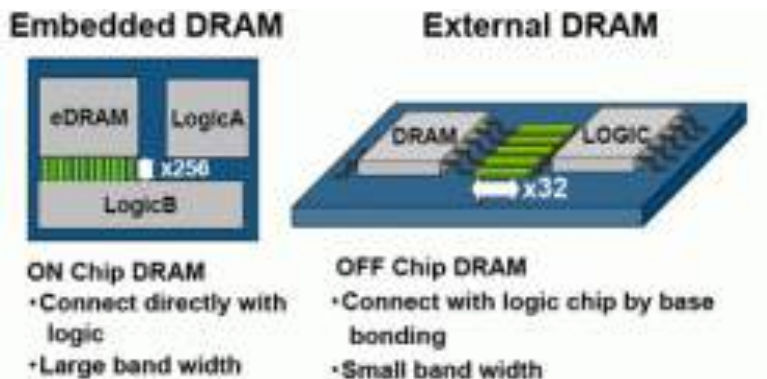
- A normal stack consist of four 4 DRAM dies on a base die and has two 128-bit channels per DRAM die
 - Making 8 channels in total which results in a 1024-bit interface
 - 4 HBM stacks gives a width of $4 * 1024 = 4096b$, 1Gb/s
 - Bandwidth: $4096b * 1Gb/s = 512GB/s$
- [Nvidia Tesla P100](#): HBM2, 4096b, 16GB, 732.2GB/s
- [Nvidia Tesla A100](#): HBM2e, 5120b, 40GB, 1555GB/s



GDDR5	Per Package	HBM
32-bit	Bus Width	1024-bit
Up to 1750MHz (7GBps)	Clock Speed	Up to 500MHz (1GBps)
Up to 28GB/s per chip	Bandwidth	>100GB/s per stack
1.5V	Voltage	1.3V

eDRAM[嵌入式]

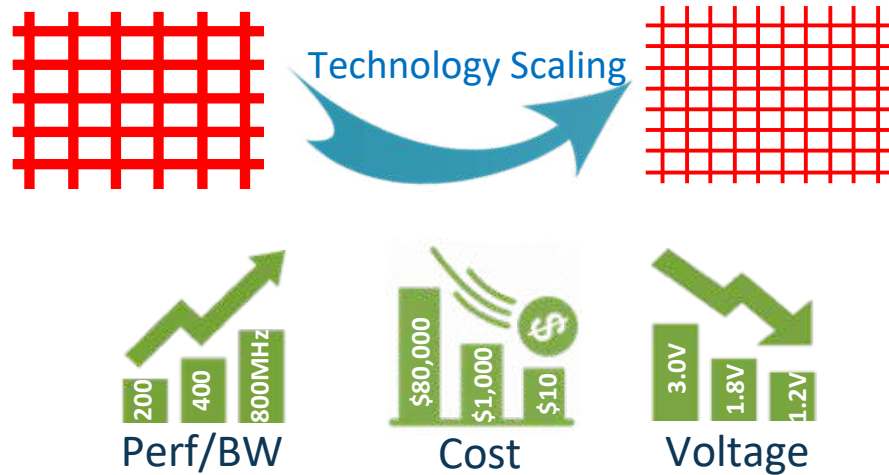
- eDRAM: embedded DRAM
 - DRAM integrated on the same die with ASIC/logic
- No pin limitations
 - Can access using a wide on-chip buses
- System power savings
 - Avoids off-chip I/O transfers



Use of eDRAM in various products

Product name	Amount of eDRAM
IBM z15	256+ MB
IBM's System Controller (SC) SCM, with L4 cache for the z15	960 MB
Intel Haswell , Iris Pro Graphics 5200 (GT3e)	128 MB
Intel Broadwell , Iris Pro Graphics 6200 (GT3e)	128 MB
Intel Skylake , Iris Graphics 540 and 550 (GT3e)	64 MB
Intel Skylake, Iris Pro Graphics 580 (GT4e)	64 or 128 MB
Intel Coffee Lake , Iris Plus Graphics 655 (GT3e)	128 MB
PlayStation 2	4 MB
PlayStation Portable	4 MB
Xbox 360	10 MB
Wii U	32 MB

DRAM Scaling[缩放]



Production year	Chip size	DRAM type	Best case access time (no precharge)			Precharge needed
			RAS time (ns)	CAS time (ns)	Total (ns)	Total (ns)
2000	256M bit	DDR1	21	21	42	63
2002	512M bit	DDR1	15	15	30	45
2004	1G bit	DDR2	15	15	30	45
2006	2G bit	DDR2	10	10	20	30
2010	4G bit	DDR3	13	13	26	39
2016	8G bit	DDR4	13	13	26	39

Scaling Issues[问题]

- DRAM cells are more leaky[数据流失]
 - More frequent refreshes
- Slower access[访问时延]
 - Longer sensing and restoring time
- Decreased reliability[可靠性]
 - Cross-talking noise, enlarged process variations



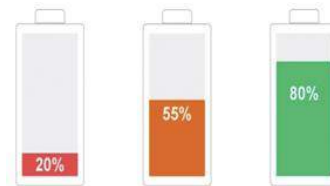
Less charge
higher leakage current

More Leaky



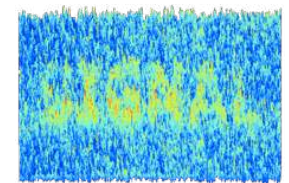
Larger resistance
Weaker signal

Longer Sensing



Larger resistance
Lower voltage

Prolonged Restore

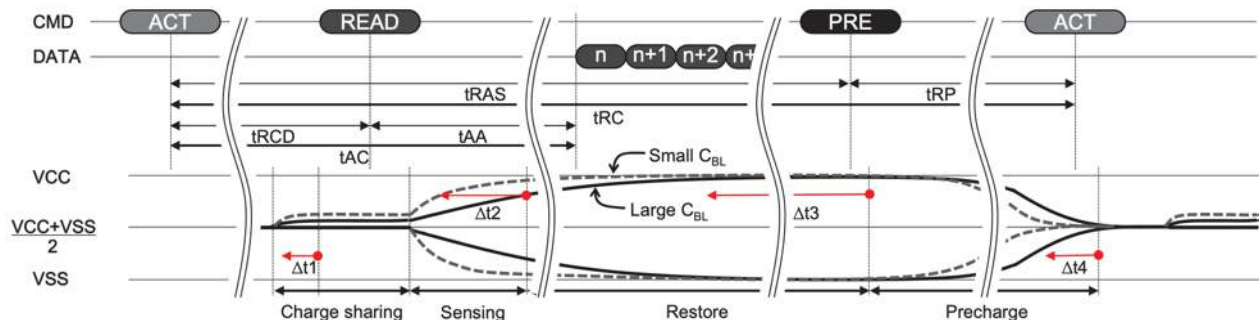


Nearer cells
Process variations

Severer Noise

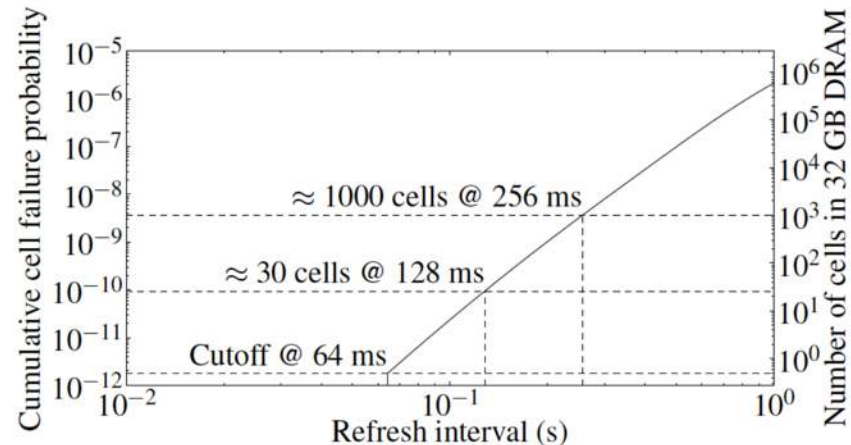
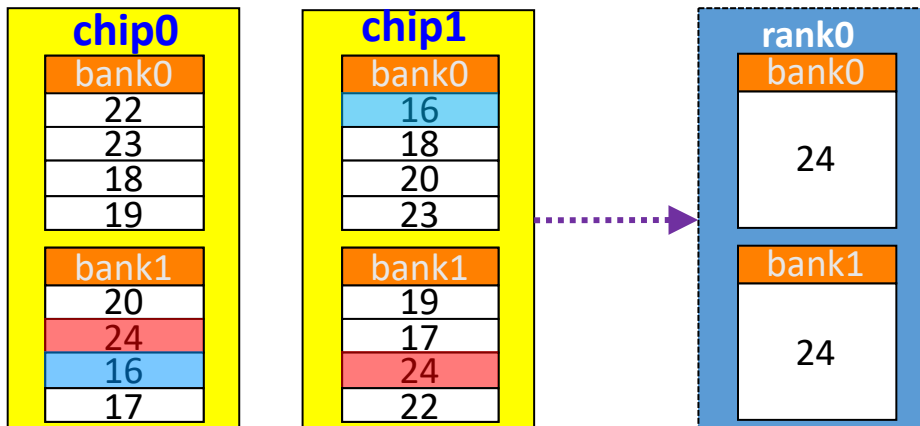
DRAM Researches[前沿研究]

- Sharing/sensing timing reduction[读取时延]
 - Optimize DRAM internal structures [CHARM'ISCA13, TL-DRAM'HPCA13, etc]
 - Utilize existing timing margins [NUAT'HPCA14, AL-DRAM'HPCA15, etc]
- DRAM restore studies[恢复时延]
 - Identify the restore scaling issue [Co-arch'MEM14, tWR'Patent15, etc]
 - Reduce restore timings [AL-DRAM'HPCA15, MCR'ISCA15, RT'HPCA16]
- Memory-based approximate computing[近似计算]
 - Skip DRAM refresh [Flicker'ASPLOS11, Alloc'CASES15, etc]
 - Restore [DrMP'PACT17]



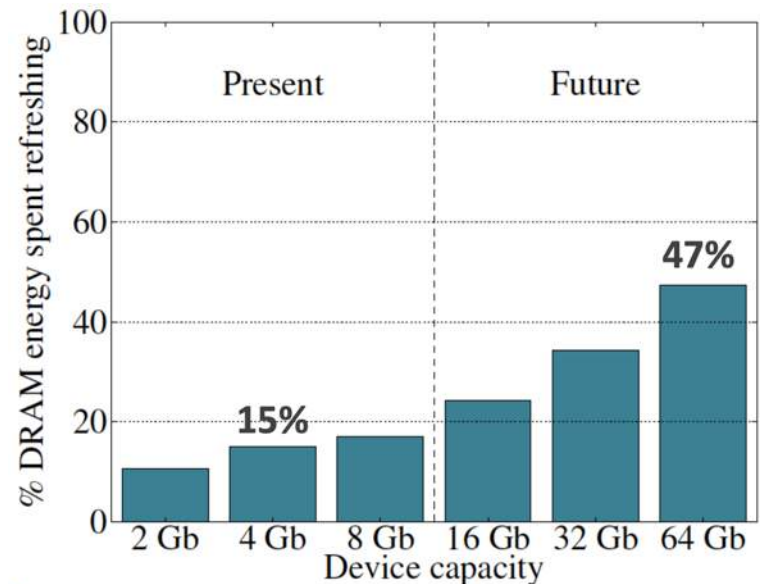
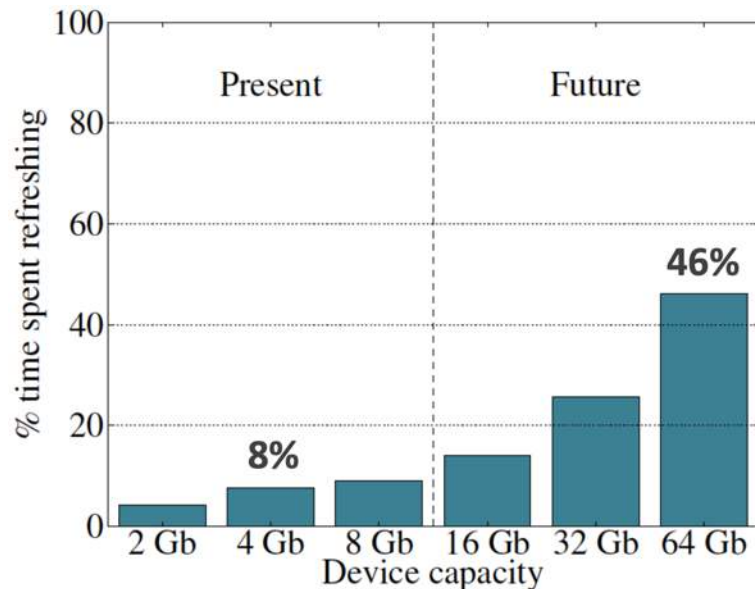
DRAM Researches (cont'd)

- Nowadays DRAMs are **worst-case determined**
- Examples:
 - Refresh: only very few rows need to be refreshed at the worst-case rate
 - Timings: overall timing constraints are determined by the worst one
- Idea: use **common-case** instead



Refresh Issues[刷新问题]

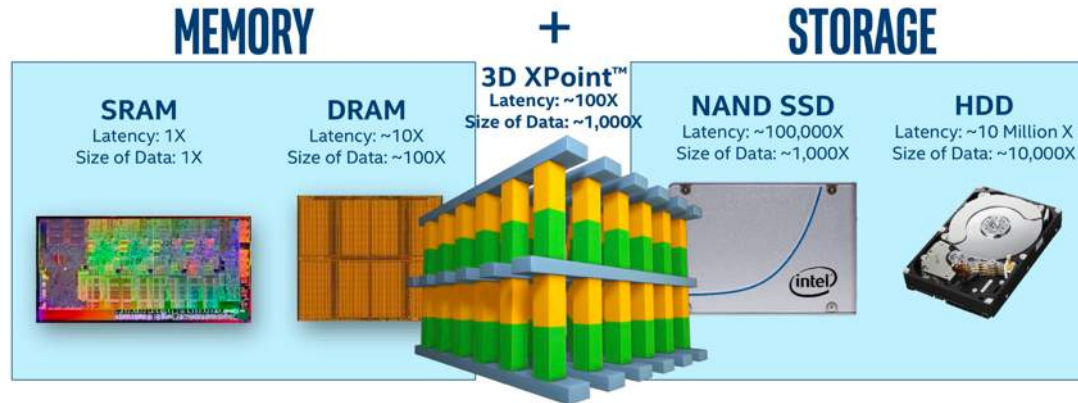
- With higher DRAM capacity, more time will be spent on refresh operations, greatly **blocking** normal reads/writes
- With further scaled DRAMs, more cells need to be refreshed at likely **higher rates** than today
- Overheads on both performance and energy



Emerging Memory[新型存储]

3D XPOINT™ MEMORY MEDIA

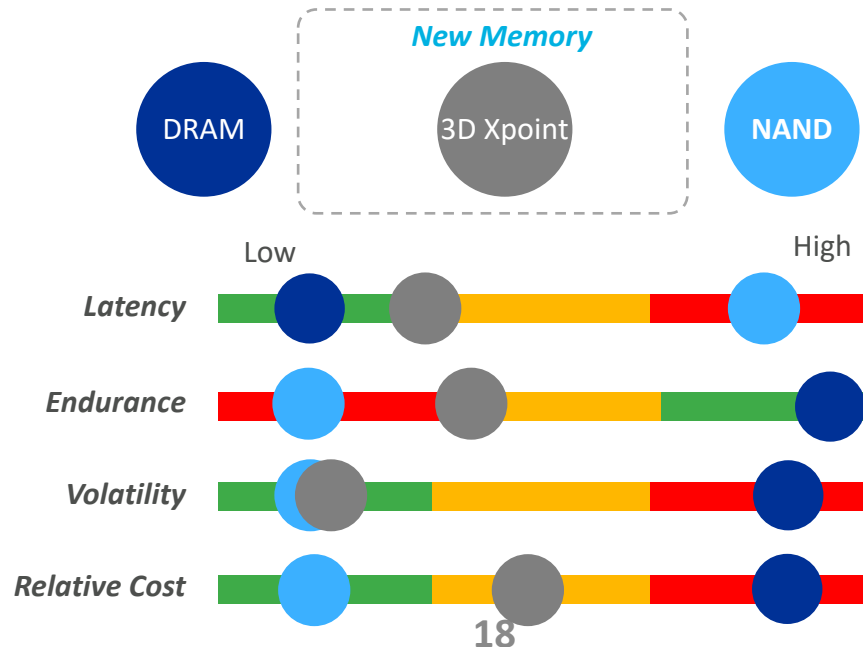
Breaks the memory/storage barrier



	Hard disk drive (HDD)	Dynamic RAM (DRAM)	NAND single-level cell (SLC) flash	Phase change RAM (PCRAM) SLC	Spin-torque transfer RAM (STT-RAM)	Resistive RAM (ReRAM)
Data retention	Y	N	Y	Y	Y	Y
Cell size (F = feature size)	N/A	6 to 10F ²	4 to 6F ²	4 to 12F ²	6 to 50F ²	4 to 10F ²
Access granularity (Bytes)	512	64	4,192	64	64	64
Endurance (writes)	>10 ¹⁵	>10 ¹⁵	10 ⁴ to 10 ⁵	10 ⁸ to 10 ⁹	>10 ¹⁵	10 ¹¹
Read latency	5 ms	50 ns	25 us	50 ns	10 ns	10 ns
Write latency	5 ms	50 ns	500 us	500 ns	50 ns	50 ns
Standby power	Disk access mechanisms	Refresh	N	N	N	N

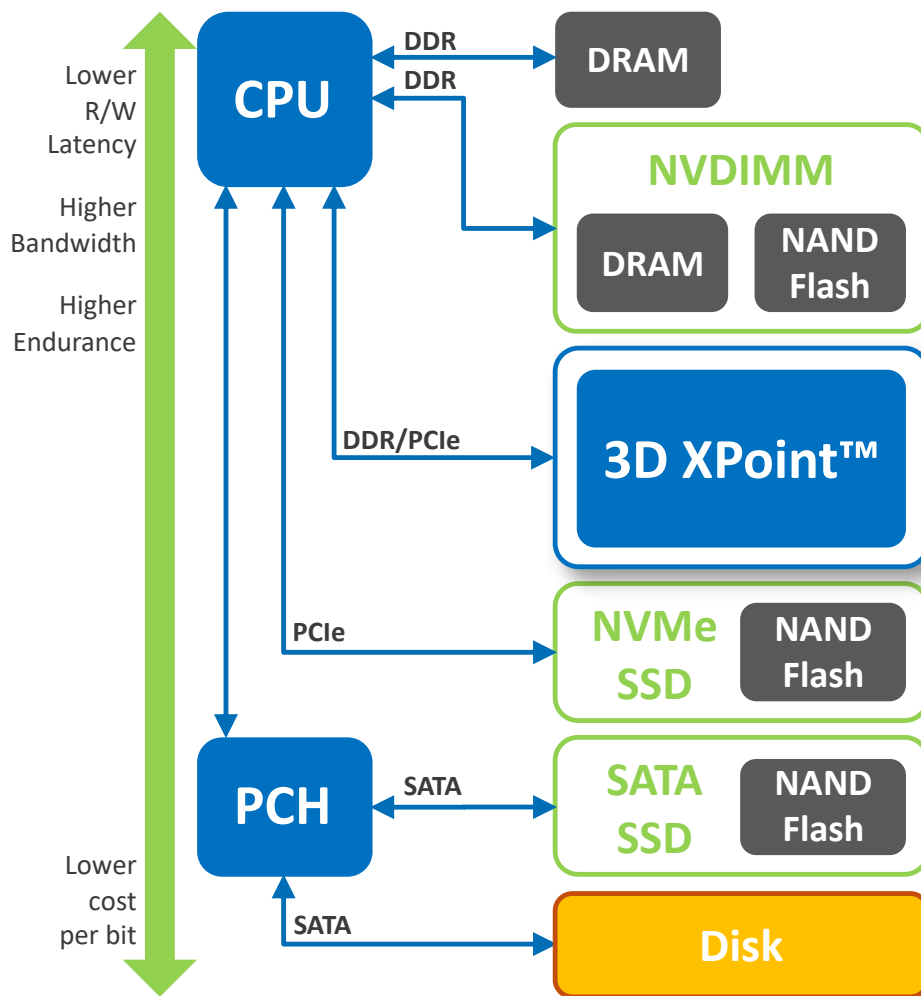
NVM[非易失性存储]

- Numerous emerging memory candidates
 - Many fall between NAND and DRAM
- Pros and cons
 - Non-volatility with fraction of DRAM cost/bit
 - Ideal for large memory systems
 - Slower access and limited lifetime



Future Memory System[未来存储系统]

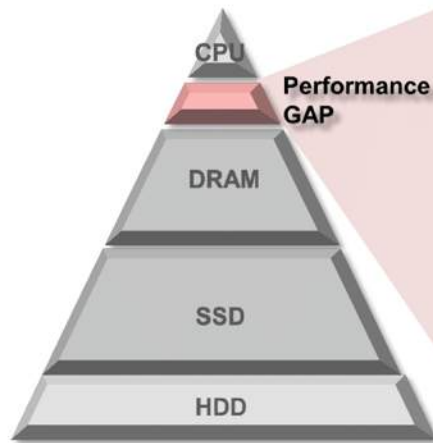
- Demands[需求]
 - Low latency
 - Large size
 - High bandwidth
 - Low power/energy
- Hybrid memory[混合]
 - DRAM + emerging
- Abstracted interface[抽象]
 - Hide device characteristics
- Changing processor-memory relationship[存算]
 - Processor-centric to memory-centric



NDP/PIM[近内存/存内计算]

<https://cseweb.ucsd.edu/~swanson/papers/IEEEMicro2014WONDP.pdf>

- Near data processing
 - Minimize data movement by computing at the most appropriate location in the hierarchy
 - In NDP, computation can be performed right at the data's home, either in caches, main memory, or persistent storage
- Processing-in-memory
 - Do computation inside the memory

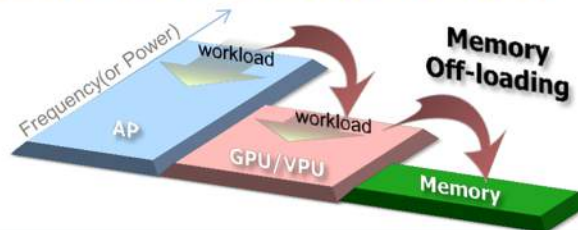


Processing In-Memory

Better parallelism and lower bus traffic

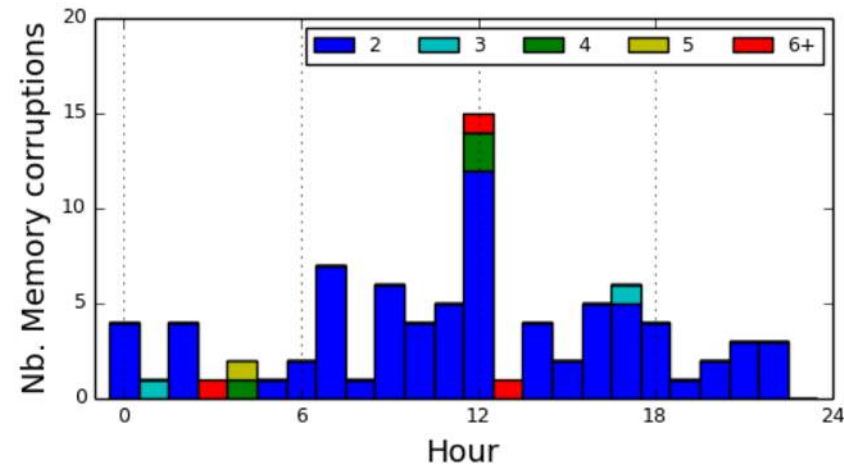


Memory off-loading for lower frequency and power



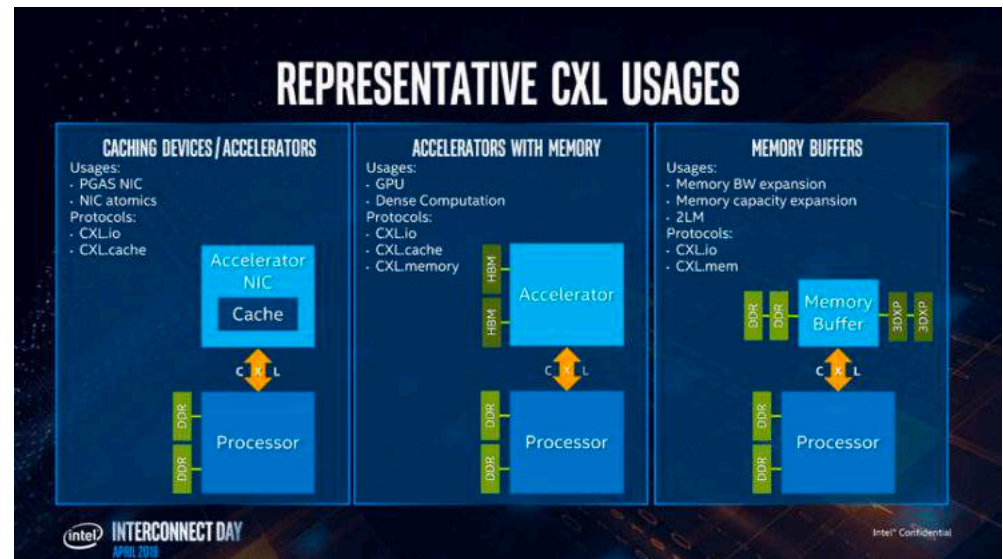
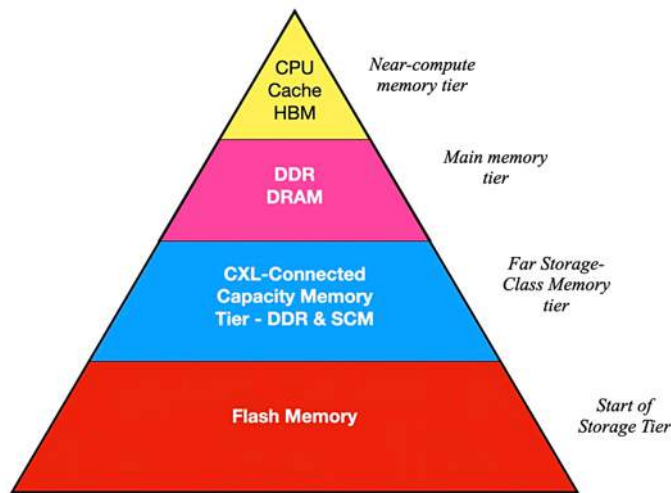
Memory Dependability[可靠性]

- Memory is susceptible to cosmic rays
- *Soft errors*: dynamic/transient errors
 - Detected and fixed by error correcting codes (ECC)
- *Hard errors*: permanent errors
 - Use spare rows to replace defective rows
- Chip-level errors
 - Chipkill: a RAID-like error recovery technique
- *Stuck-at errors*
 - May use data-dependent sparing
- Endurance problems
- Cross-talk (bit-line & word-line)
- Read/write disturbance



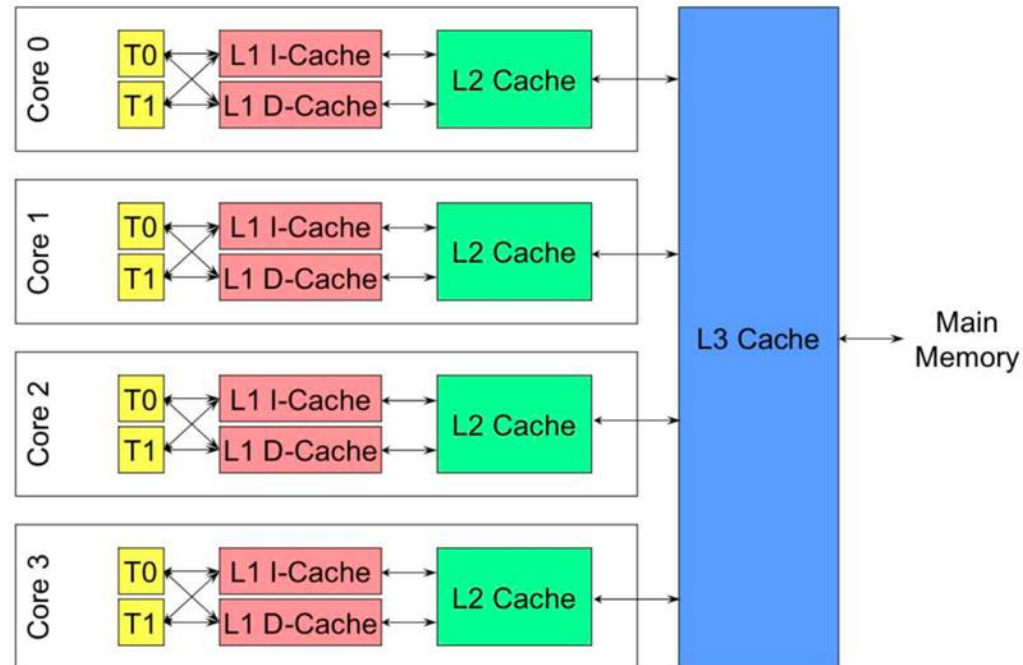
Storage Class Memory (SCM)

- An era of very big, PB-level memory pools
- The big memory pooling is made possible by the compute express link (CXL)
- CXL is a standard for linking memory bus devices together: CPUs, GPUs, and memory (and a few other more exotic things like TPUs and DPUs).

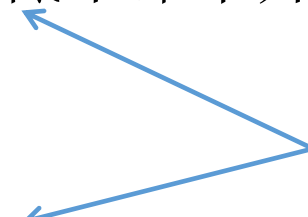


To Further Optimize Cache[优化缓存]

- Average memory access time (AMAT) = (hit-rate * hit-latency) + (miss-rate * miss-latency)
- Basic requirements
 - Hit latency
 - Miss rate
 - Miss penalty
- Two more requirements
 - Cache bandwidth
 - Power consumption

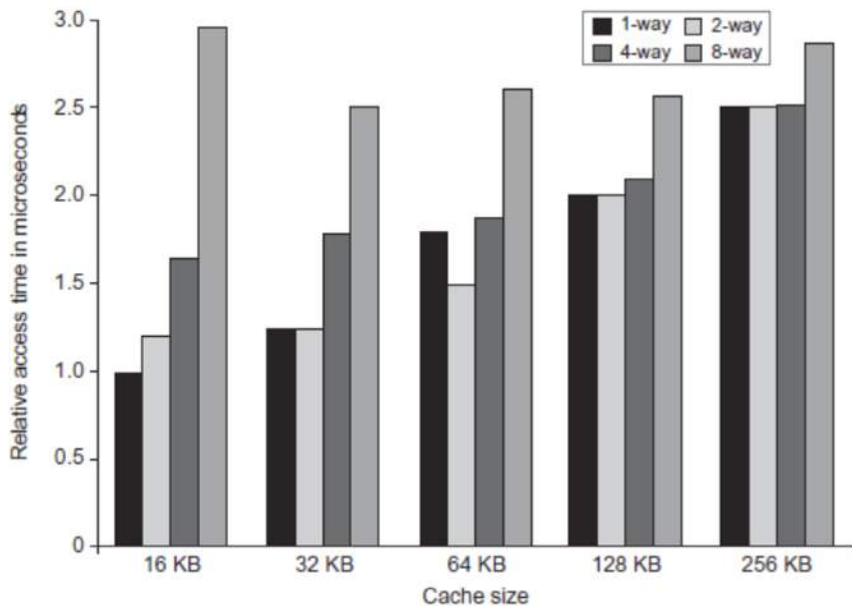


Advanced Cache Optimizations[优化]

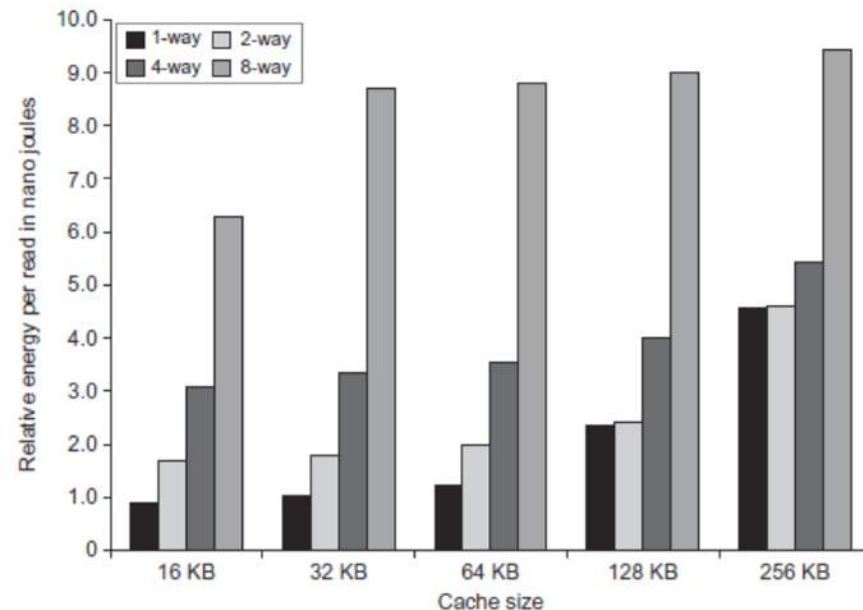
- Reducing the hit time[缩短命中时延]
 - Small and simple first-level caches
 - Way prediction
 - Increasing cache bandwidth[提高缓存带宽]
 - Pipelined caches
 - Multibanked caches
 - Non-blocking caches
 - Reducing miss penalty[降低不命中开销]
 - Critical word first
 - Merging write buffers
 - Reducing miss rate[降低不命中率]
 - Compiler optimizations
- parallelism
- 

#1: Small & Simple 1st-level Cache[小]

- To reduce hit time and power
- The L1 cache size has recently increased either slightly or not at all
 - Limited size: pressure of both a fast clock cycle and power limitations encourages small sizes
 - Lower level of associativity: reduce both hit time and power

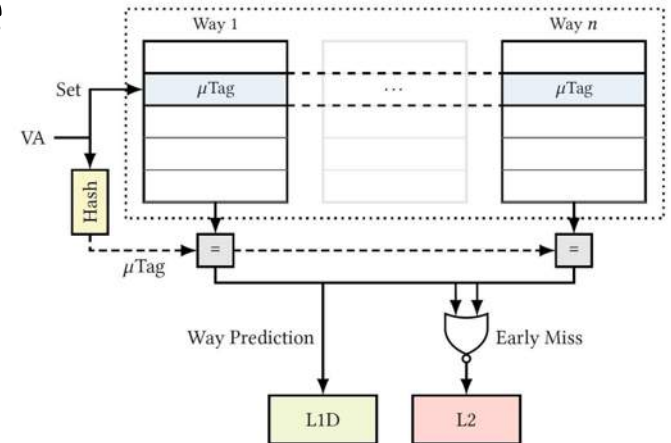


25



#2: Way Prediction[预测]

- To reduce hit time
 - Add extra bits in the cache to predict the way of the next cache access
 - Block predictor bits
 - Multiplexor is set early to select the desired block
 - And in that clock cycle, only a single tag comparison is performed in parallel with reading the cache data
 - A miss results in checking the other blocks for matches in the next clock cycle
- Miss-prediction gives longer hit time
 - Prediction accuracy
 - > 90% for two-way
 - > 80% for four-way
 - I-cache has better accuracy than D-cache
 - First used on MIPS R10000 in mid-90s, now used on ARM Cortex-A8

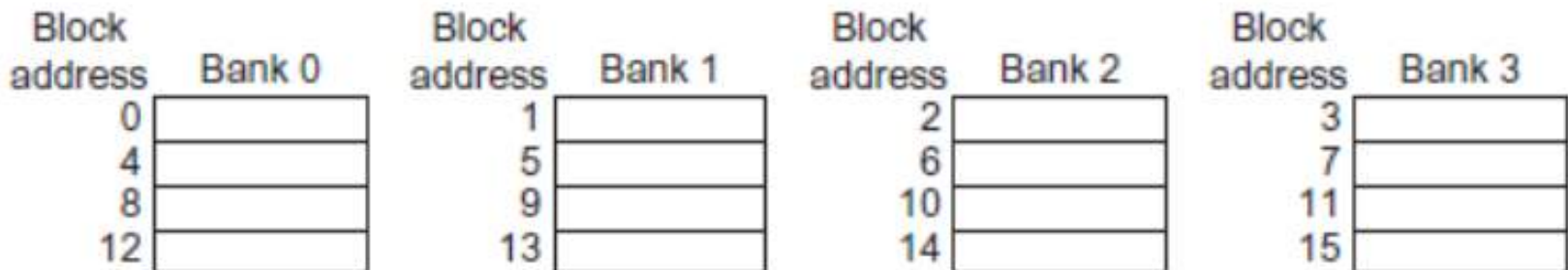


#3: Pipelined[流水线]

- To increase bandwidth
 - Primarily target at L1, where access bandwidth constrains instruction throughput
 - Multibanks are also used in L2/L3, but mainly for power
- Pipelining L1
 - Stages
 - Address calculation
 - disambiguation (decoder)
 - cache access (parallel tag and data)
 - result drive (aligner)
 - Allows a higher clock cycle, at the cost of increased latency
 - Examples
 - Pentium: 1 cycle, Pentium Pro – III: 2, Pentium 4 – Core i7: 4 cycles

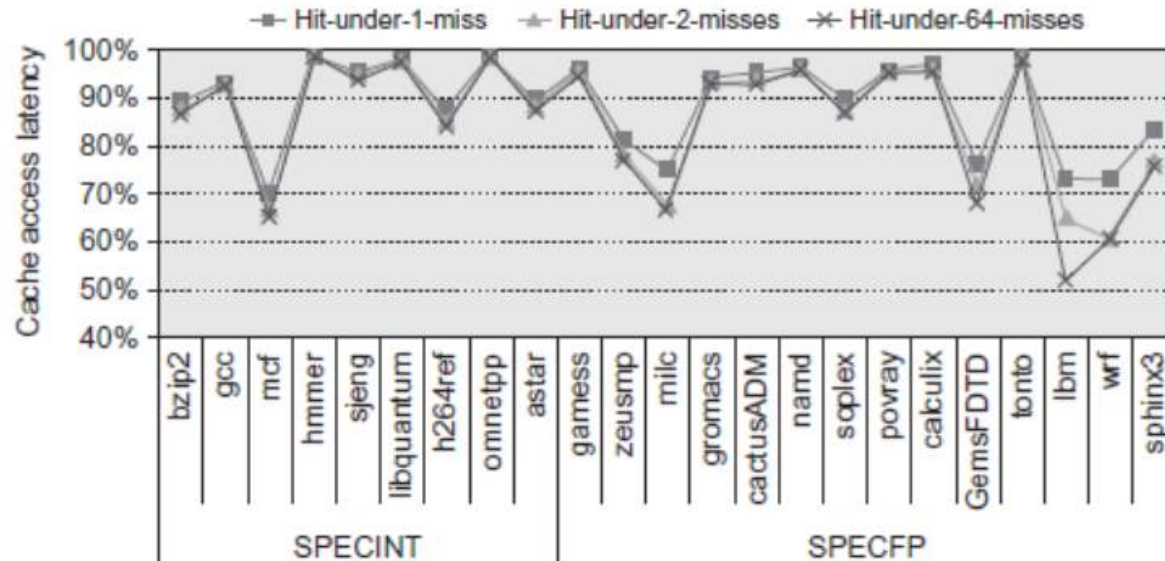
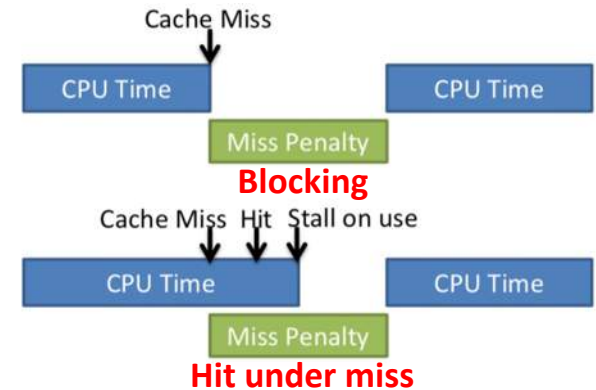
#3: Multibanked[多单元]

- Organize cache as independent banks to support simultaneous access
 - ARM Cortex-A8 supports 1-4 banks for L2
 - Intel i7 supports 4 banks for L1 and 8 banks for L2
- Interleave banks according to block address
 - Banking works best when the accesses naturally spread across banks
- Multiple banks also are a way to reduce power consumption in both caches and DRAM



#4: Nonblocking Caches[非阻塞]

- To increase cache bandwidth
- Allow hits before previous misses complete
 - “Hit under miss”
 - “Hit under multiple miss”
- Nontrivial to implement the nonblocking
 - Arbitrating contention between hits and misses; tracking outstanding misses
 - Miss Status Handling Registers (MSHRs)



#5: Critical Word First & Early Restart

- To reduce miss penalty
- Processor normally needs just one word of the block at a time
 - Don't wait for the full block to be loaded before sending the requested word and restarting the processor
- Critical word first[关键字优先]
 - Request missed word from memory first
 - Send it to the processor as soon as it arrives
- Early restart[提早重启]
 - Request words in normal order
 - Send missed work to the processor as soon as it arrives
- Effectiveness depends on block size and likelihood of another access to the portion of the block that has not yet been fetched

#6: Merging Write Buffers[写缓冲合并]

- To reduce miss penalty
- When storing to a block that is already pending in the write buffer, update write buffer
- Advantages
 - Multiword writes are usually faster than writes one word a time
 - Reduces stalls due to full write buffer
- Do not apply to I/O addresses[I/O设备]

Write address	V	V	V	V		
100	1	Mem[100]	0	0	0	0
108	1	Mem[108]	0	0	0	0
116	1	Mem[116]	0	0	0	0
124	1	Mem[124]	0	0	0	0

No write buffering

Write address	V	V	V	V				
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

Write buffering

#7: Compiler Optimizations[编译]

- To reduce miss rate, without any hardware changes
- Loop interchange
 - Swap nested loops to access memory in sequential order
 - Improving spatial locality
 - Maximizes use of data in a cache block before they are discarded

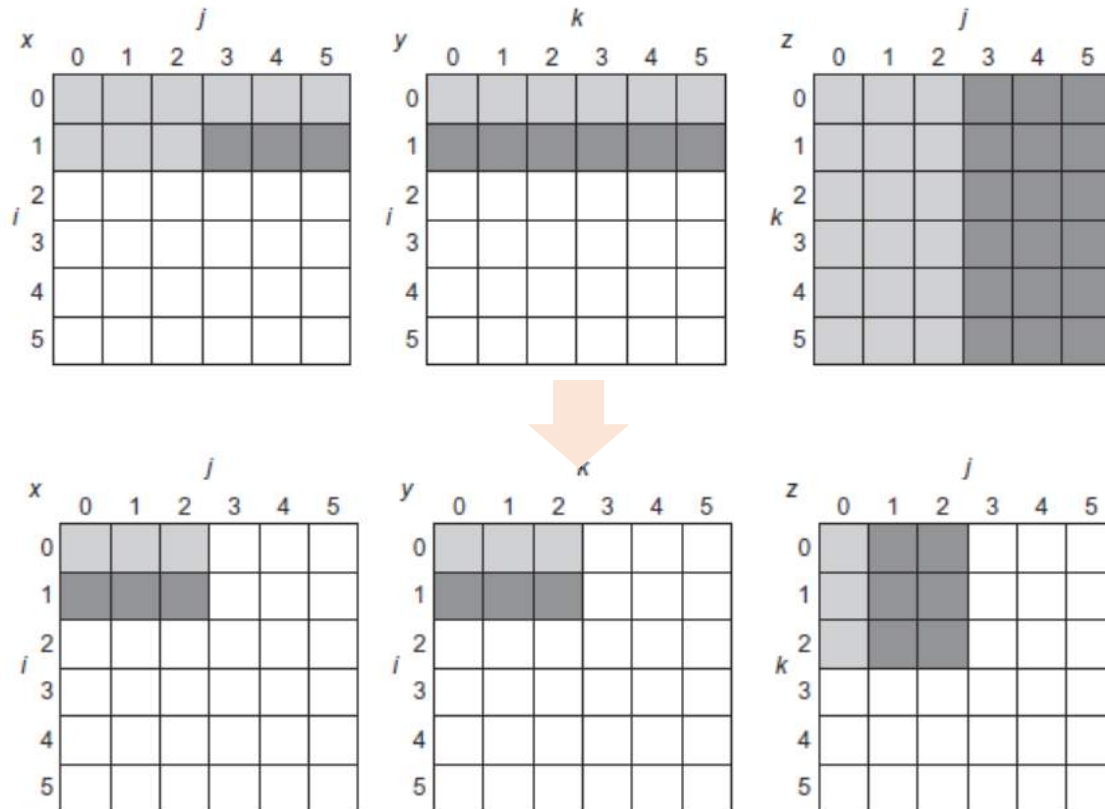
```
/* Before */  
for ( j = 0; j < 100; j = j + 1 )  
    for ( i = 0; i < 5000; i = i + 1 )  
        x[i][j] = 2 * x[i][j];
```



```
/* After */  
for ( i = 0; i < 5000; i = i + 1 )  
    for ( j = 0; j < 100; j = j + 1 )  
        x[i][j] = 2 * x[i][j];
```

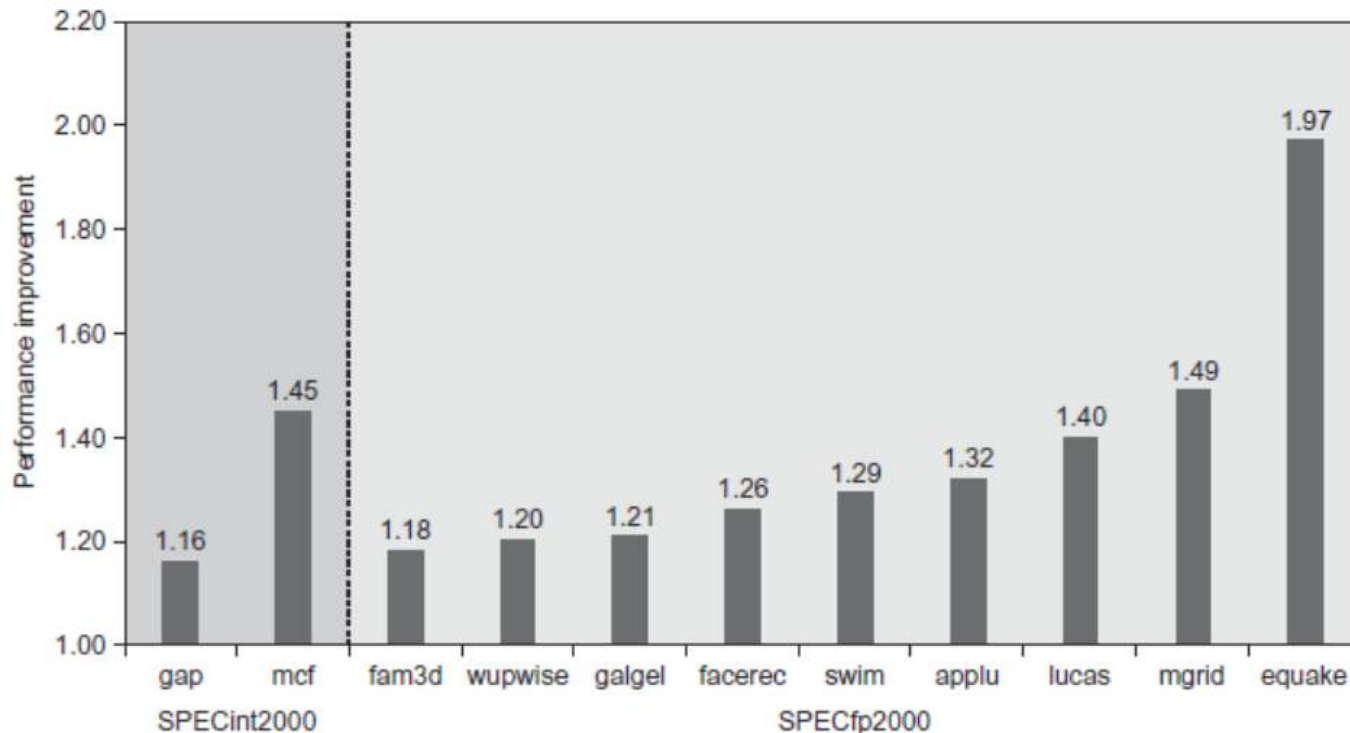

#7: Compiler Optimizations (cont'd)

- Blocking to reduce cache misses
 - Instead of accessing entire rows or columns, subdivide matrices into blocks
 - Exploits a combination of spatial and temporal locality, and can even help register allocation



#8: Hardware Prefetching[硬件预取]

- To reduce miss penalty or miss rate
- Prefetch items before the processor requests them
 - Instruction: fetches two blocks on miss, the requested and the next consecutive
 - Data: prefetch predicted blocks



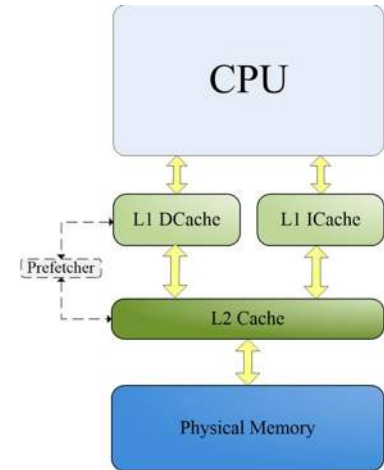
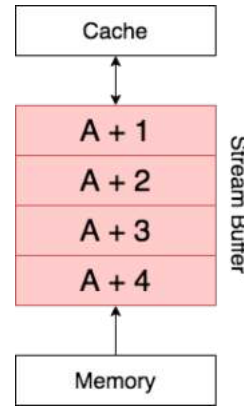
#8: Hardware Prefetching (cont'd)

- What to prefetch? (prefetch useful data)

- Next sequential
- Stride
- General pattern

- Where to place?

- Directly into caches
- External buffers



- When to prefetch?

- Prefetched data should be timely provided

- Prefetching relies on extra memory bandwidth

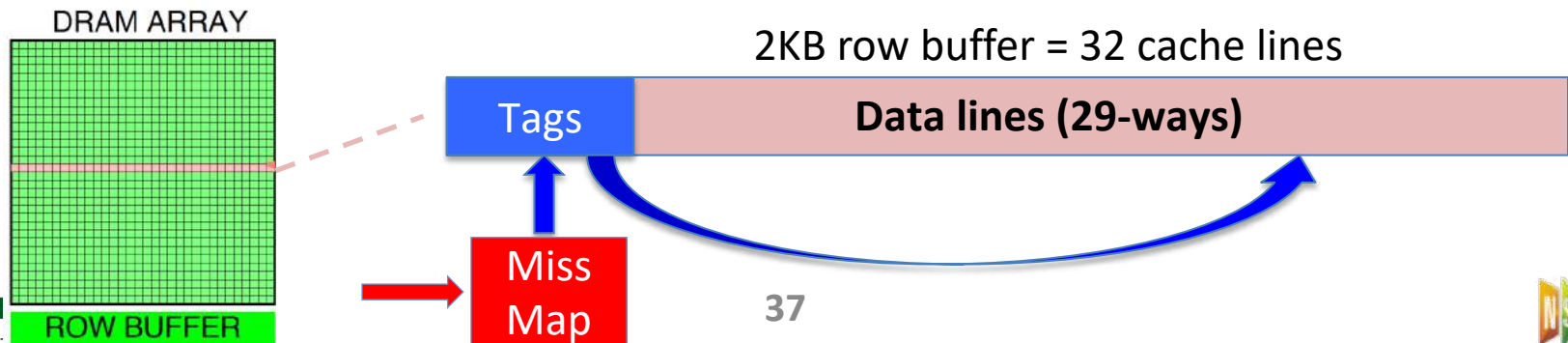
- Should not interfere much with demand accesses
- Otherwise it hurts performance

#9: Compiler-controlled Prefetching

- To reduce miss penalty or miss rate
- Compiler inserts prefetch instructions to request data before the processor needs it
- Two flavors
 - Register prefetch: loads the value into a register
 - Cache prefetch: loads data into the cache
- Typically *nonfaulting* prefetches
 - Simply turns into no-ops if they would normally result in an exception
- Compilers must take care to gain performance
 - Issuing prefetch instructions incurs an instruction overhead

#10: Use HBM[高带宽内存]

- Use HBM to build massive L4 caches, size of 128MB - 1GB
- Tags of HBM cache
 - 64B block: 1GB L4 requires 94MB of tags
 - Issue: cannot place in on-chip caches
 - 4KB block: 1GB L4 requires <1MB tag
 - Issues: inefficient use of huge blocks, and high transfer overhead
- One approach (L-H, MICRO'2011):
 - Each SDRAM row is a block index
 - Each row contains set of tags and 29 data segments
 - 29-set associative



Summary

Technique	Hit time	Band-width	Miss penalty	Miss rate	Power consumption	Hardware cost/complexity	Comment
Small and simple caches	+			-	+	0	Trivial; widely used
Way-predicting caches	+				+	1	Used in Pentium 4
Pipelined & banked caches	-	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
Critical word first and early restart			+			2	Widely used
Merging write buffer			+			1	Widely used with write through
Compiler techniques to reduce cache misses				+		0	Software is a challenge, but many compilers handle common linear algebra calculations
Hardware prefetching of instructions and data			+	+	-	2 instr., 3 data	Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware
Compiler-controlled prefetching			+	+		3	Needs nonblocking cache; possible instruction overhead; in many CPUs
HBM as additional level of cache		+/-	-	+	+	3	Depends on new packaging technology. Effects depend heavily on hit rate improvements