Part II. Memory Systems

UNIVERSITY OF MARYLAND

18 56

Today's Story

- **Cache**
  (lots to be done in content management, especially at software/app level)

- **DRAM**
  (the design space is huge, sparsely explored, poorly understood)

- **Disk & Flash**
  (flash overtaking disk, very little has been published)

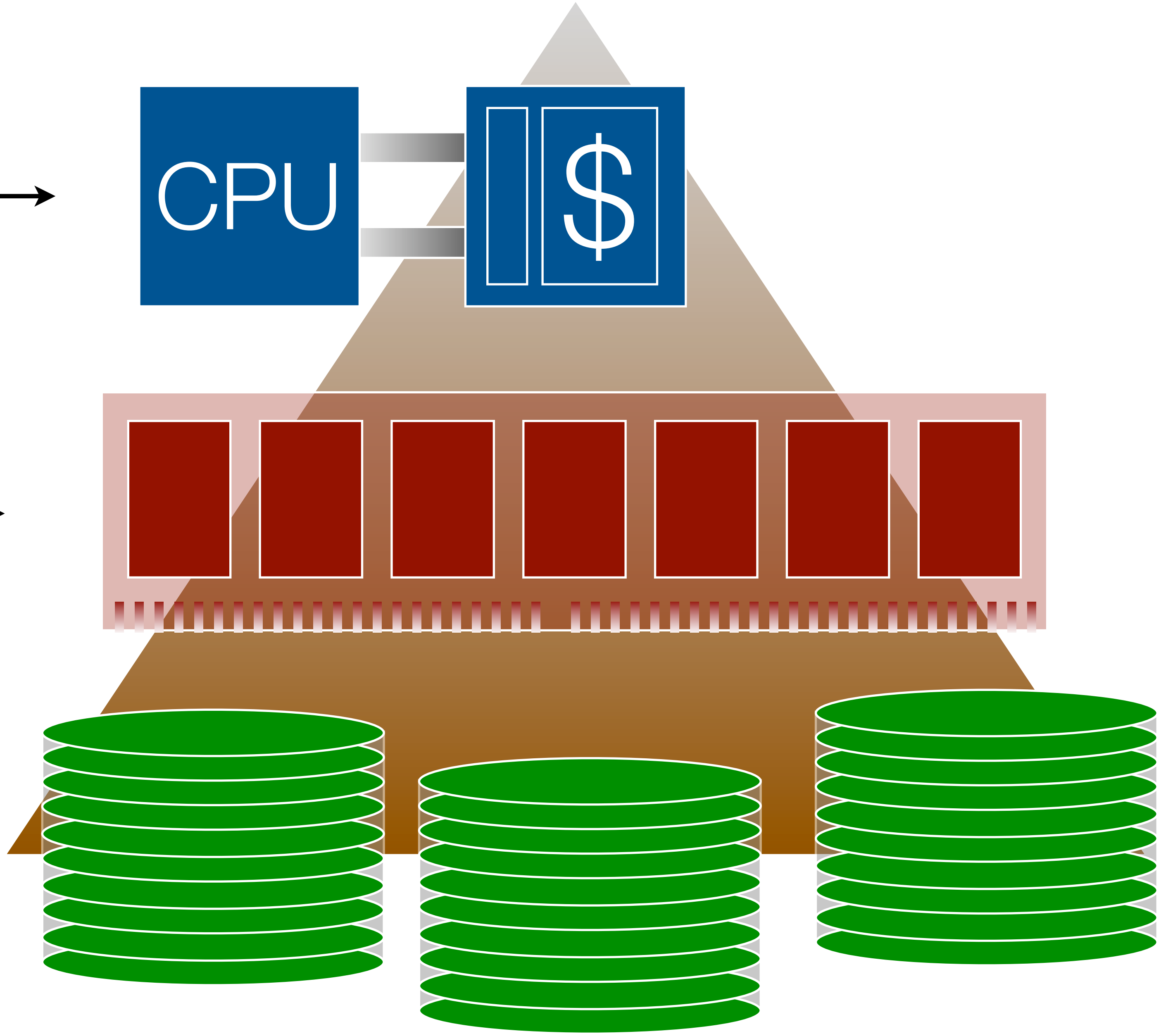
- **For each, a quick look at some of the non-obvious issues**

Perspective: Performance

~10 IPC →

~0.001 IPC →

~0.0000001 IPC→



~0.1 IPC

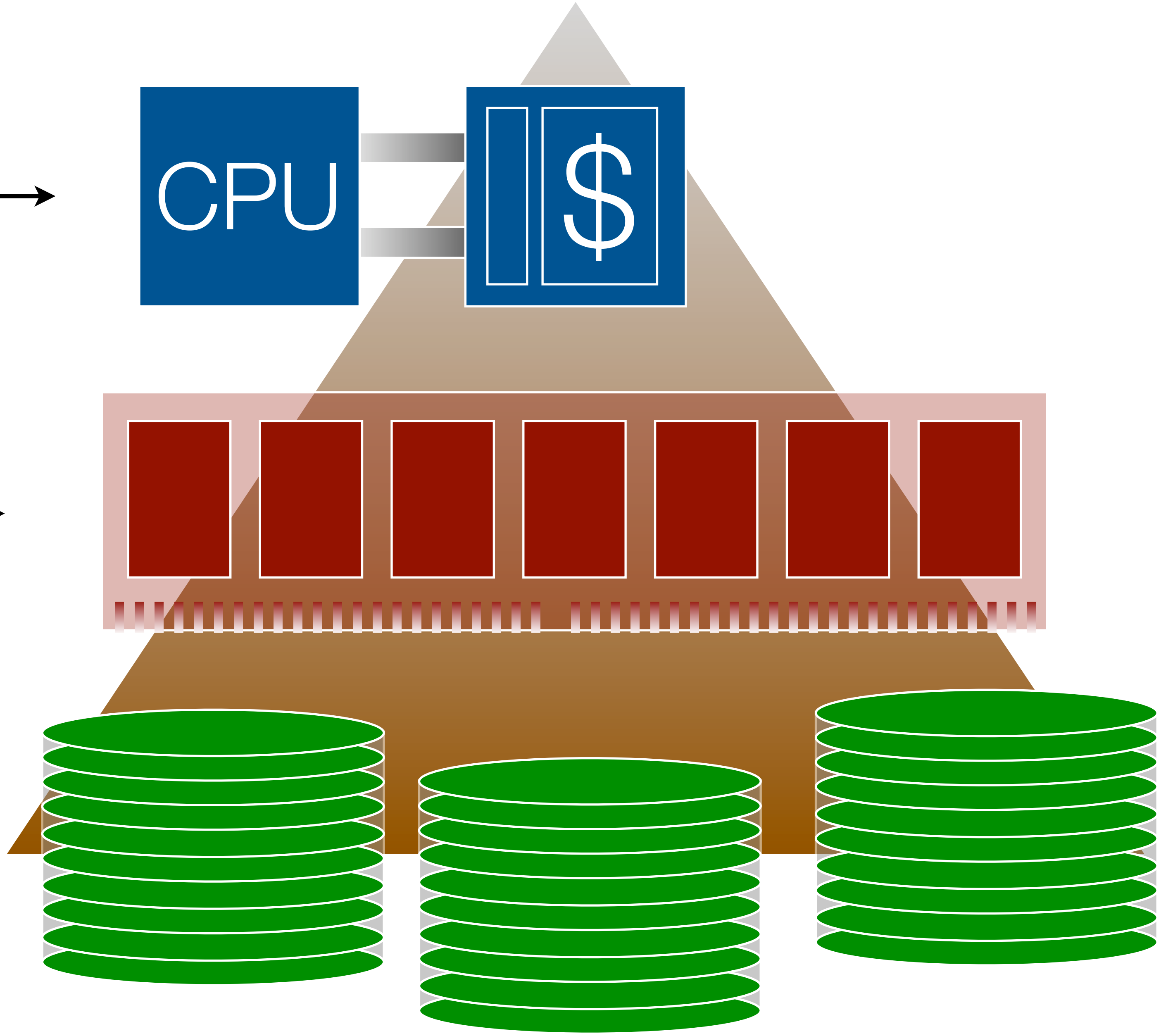
Perspective: Power

~100 W →

~10 W per DIMM →

~10 W per Disk →

~100–400W

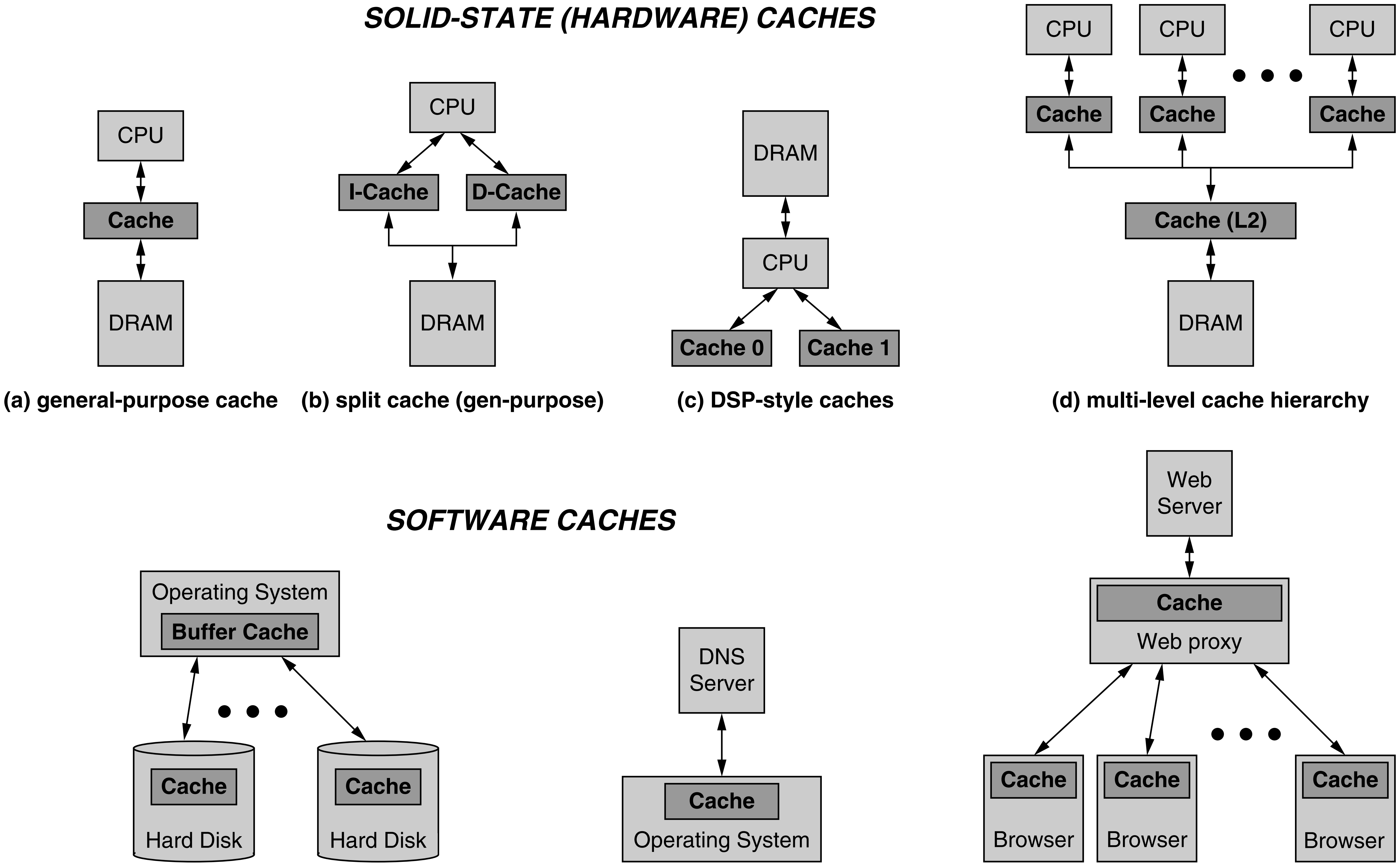


CACHE

Caches: A Small Sample

SOLID-STATE (HARDWARE) CACHES

SOFTWARE CACHES



(a) general-purpose cache

(b) split cache (gen-purpose)

(c) DSP-style caches

(d) multi-level cache hierarchy

(e) buffer caches (on both sides of interface)

(f) IP-address translation caches

(g) web-document caches

Locality Principles

- Spatial
- Temporal
- Algorithmic

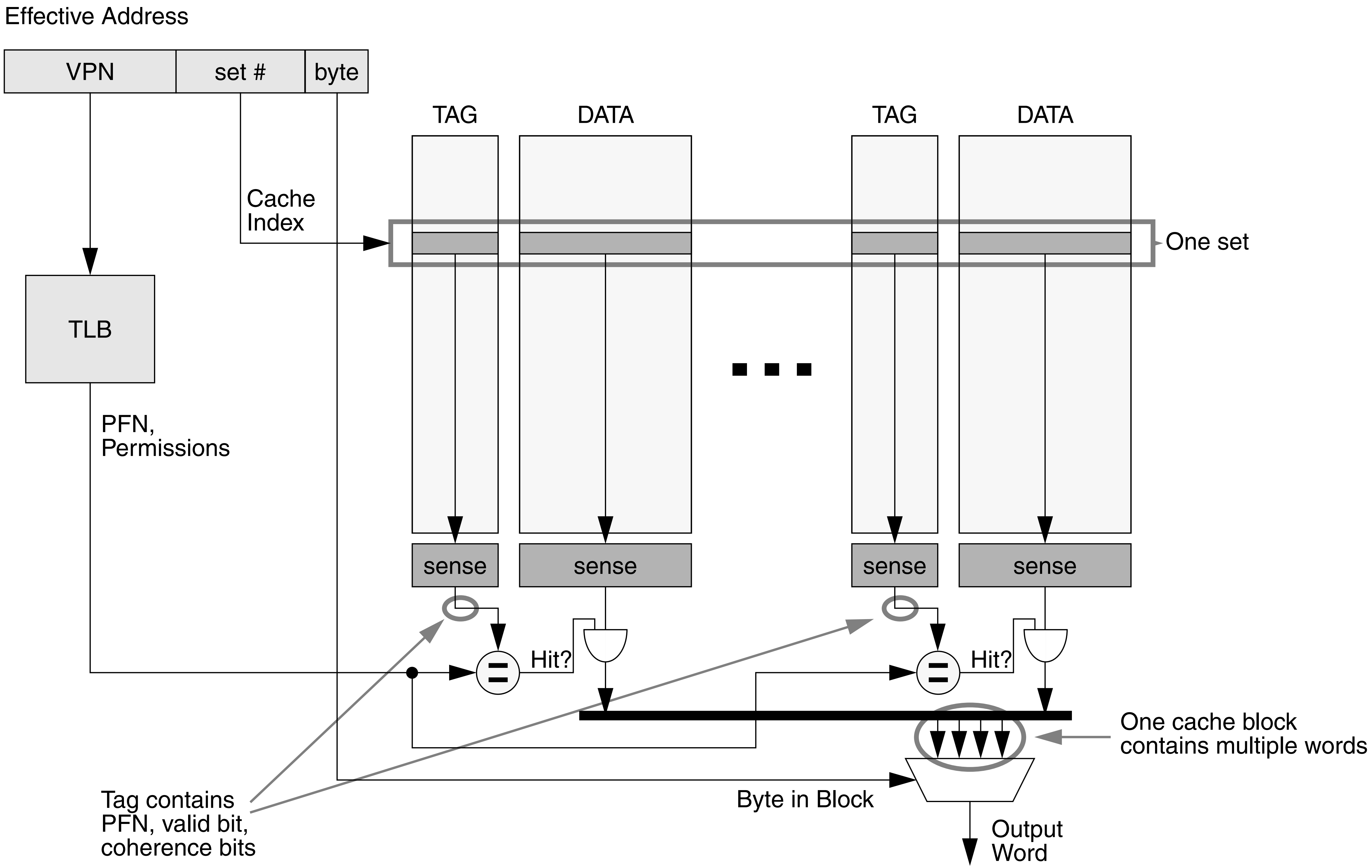
```
for each polygon P in scene {
 for each pixel p in P {
  if (depth of p at i,j < image[i,j].depth) {
   image[i,j].color = p.color
   image[i,j].depth = depth of p at i,j
  }
 }
}
```

Locality Principles

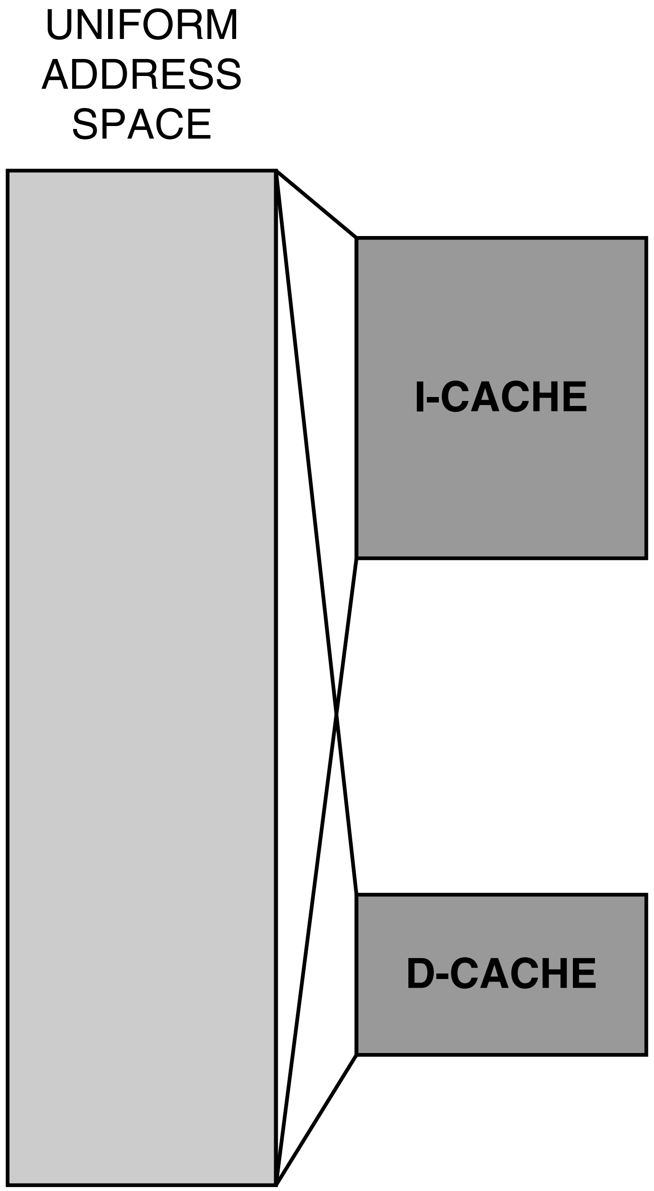
- Spatial
- Temporal
- Algorithmic

```
for each polygon P in scene {
 prefetch P.next
 for each pixel p in P {
  if (depth of p at i,j < image[i,j].depth) {
   image[i,j].color = p.color
   image[i,j].depth = depth of p at i,j
  }
 }
}
```

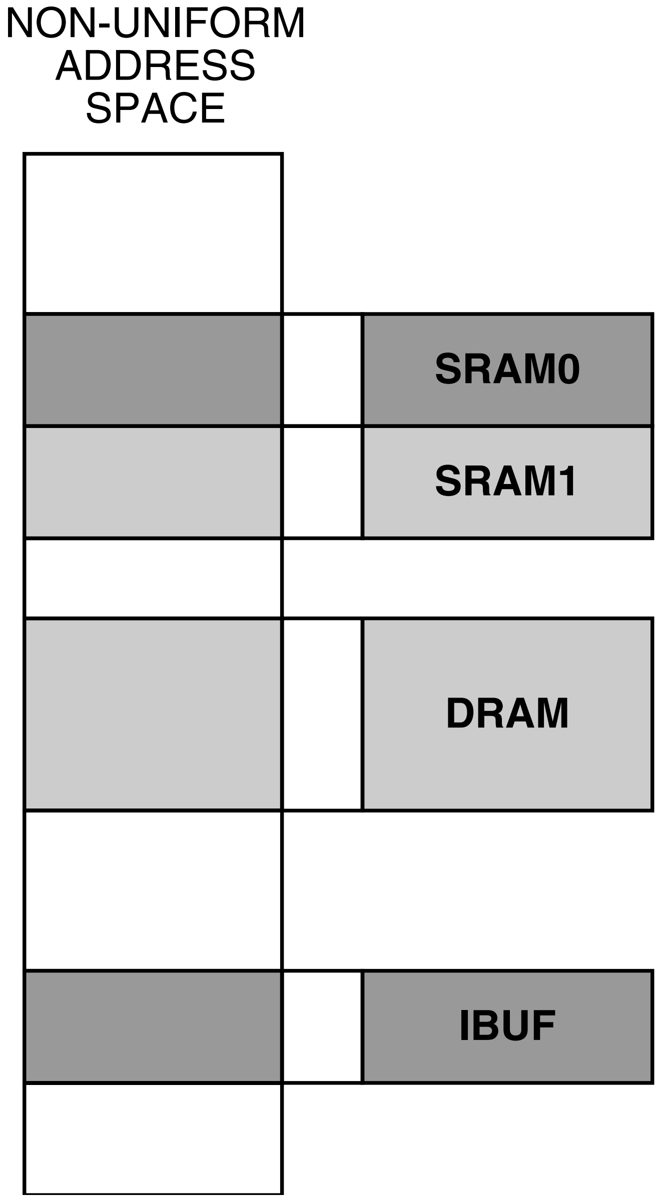
Logical Organization: Who and Where?



Logical Organization



transparent

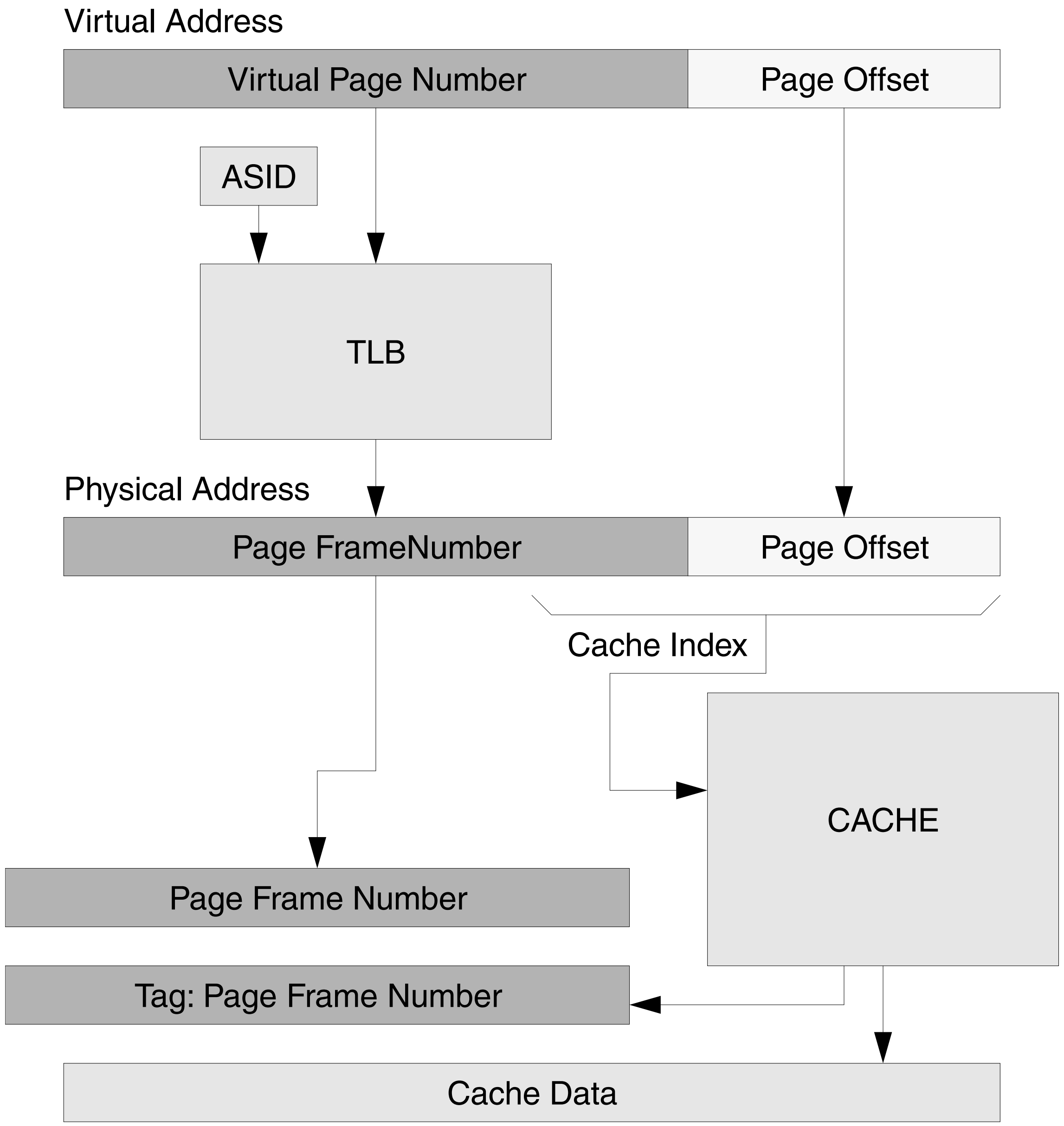


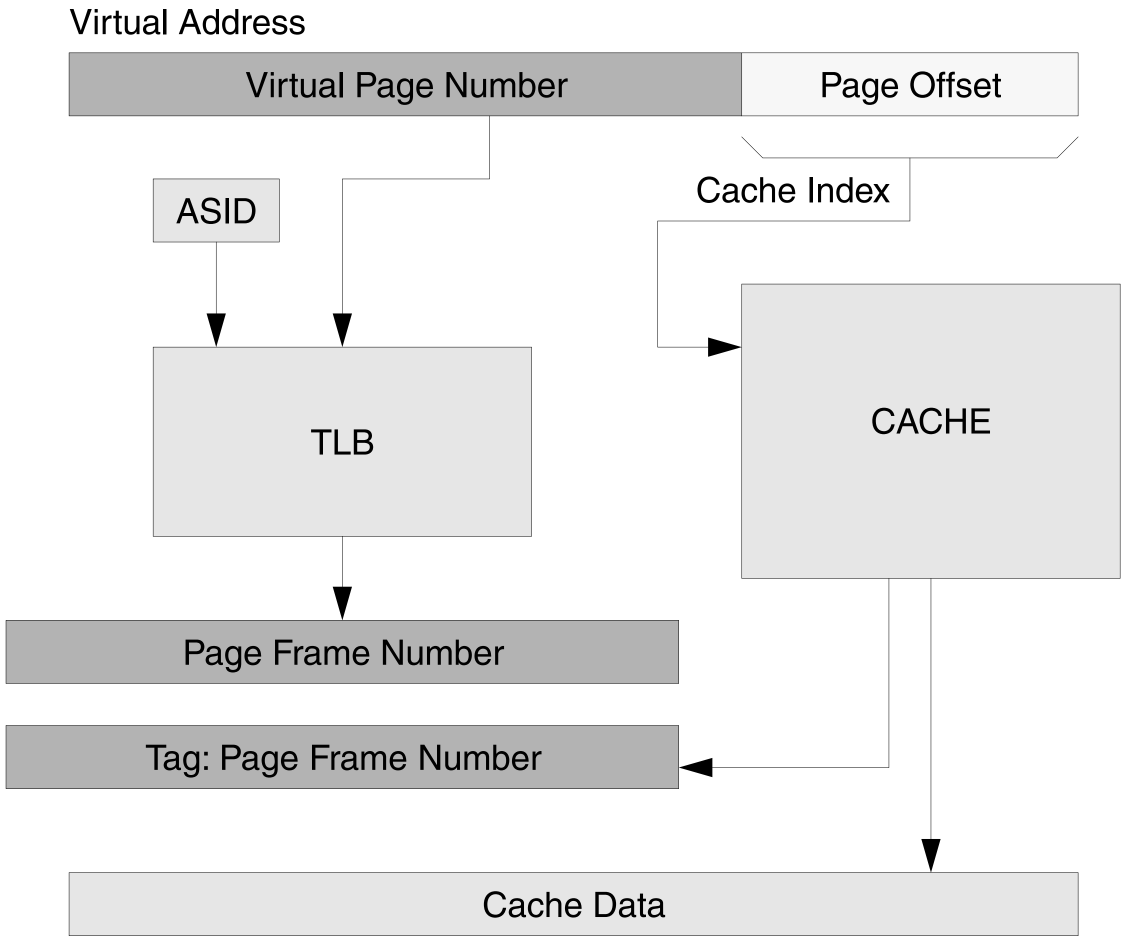
non-transparent

- Major differentiator between caches: is it part of the explicitly addressable space?

- Secondly, who manages the movement of data to/from backing store? (cache itself, or app?)

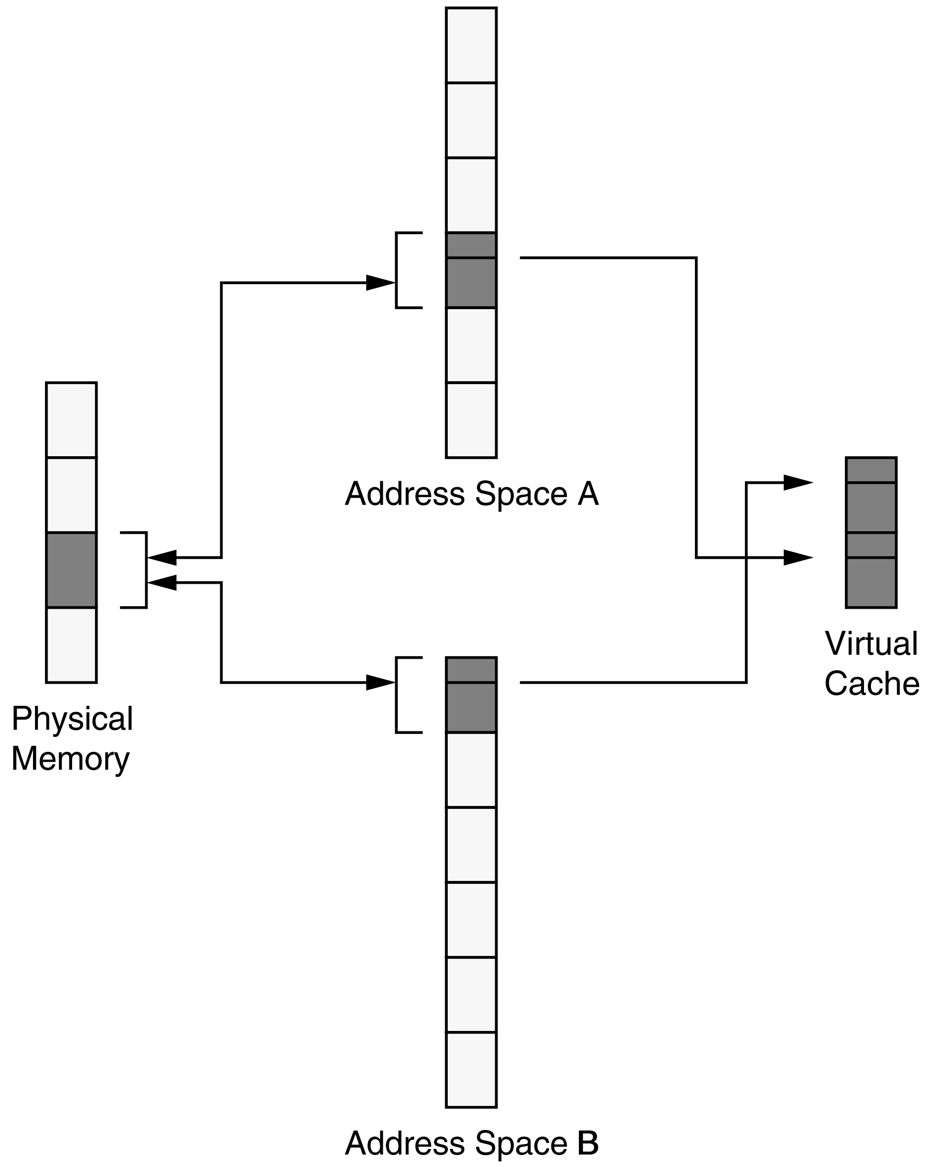
Logical Organization: Issues



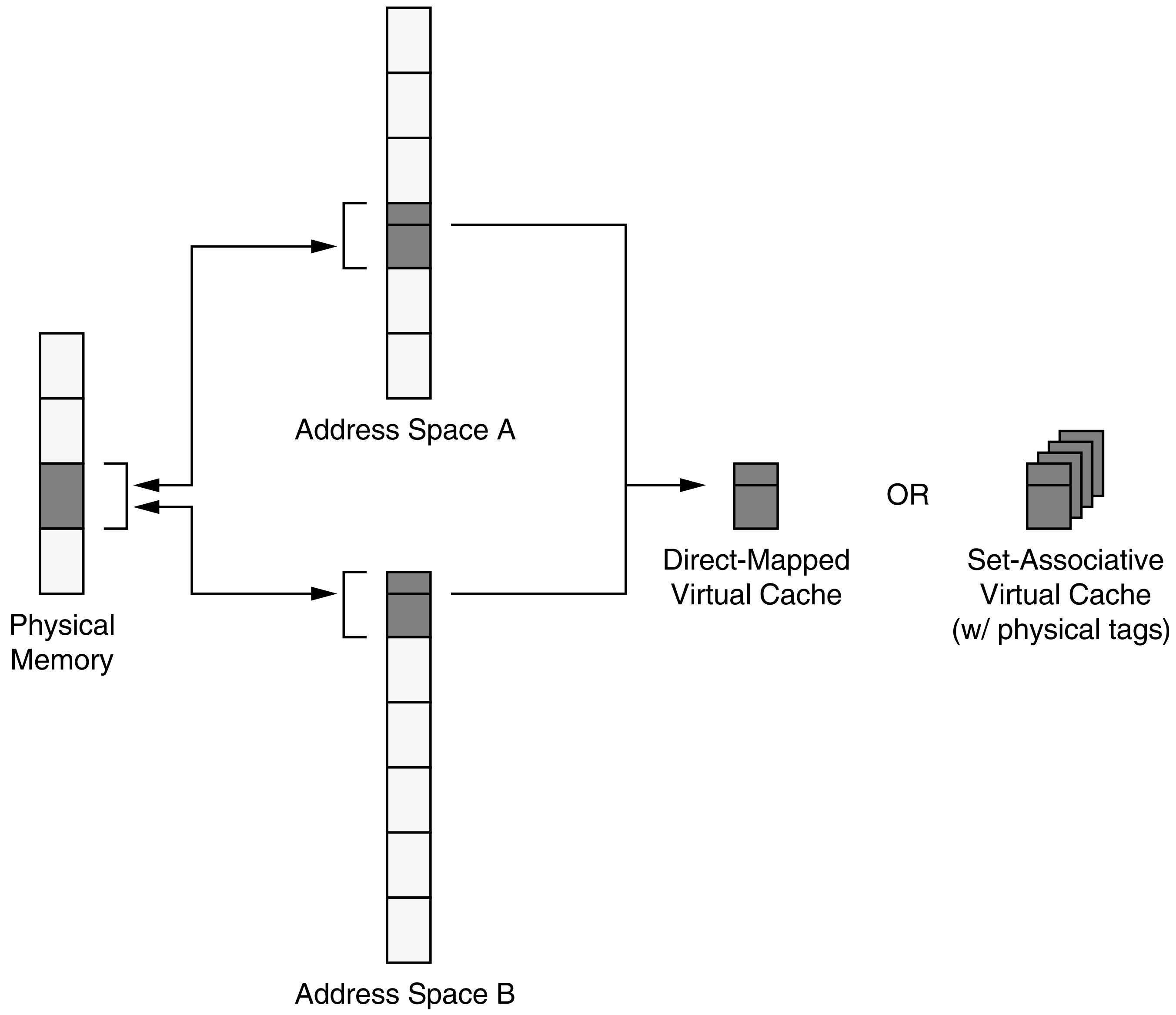


Windows assumes physical cache (left) to solve aliasing problem.

Logical Organization: Issues



the aliasing problem


Address Space A
Physical Memory
Direct-Mapped Virtual Cache
OR
Set-Associative Virtual Cache (w/ physical tags)
Address Space B

common solutions

Content Management: What and When?

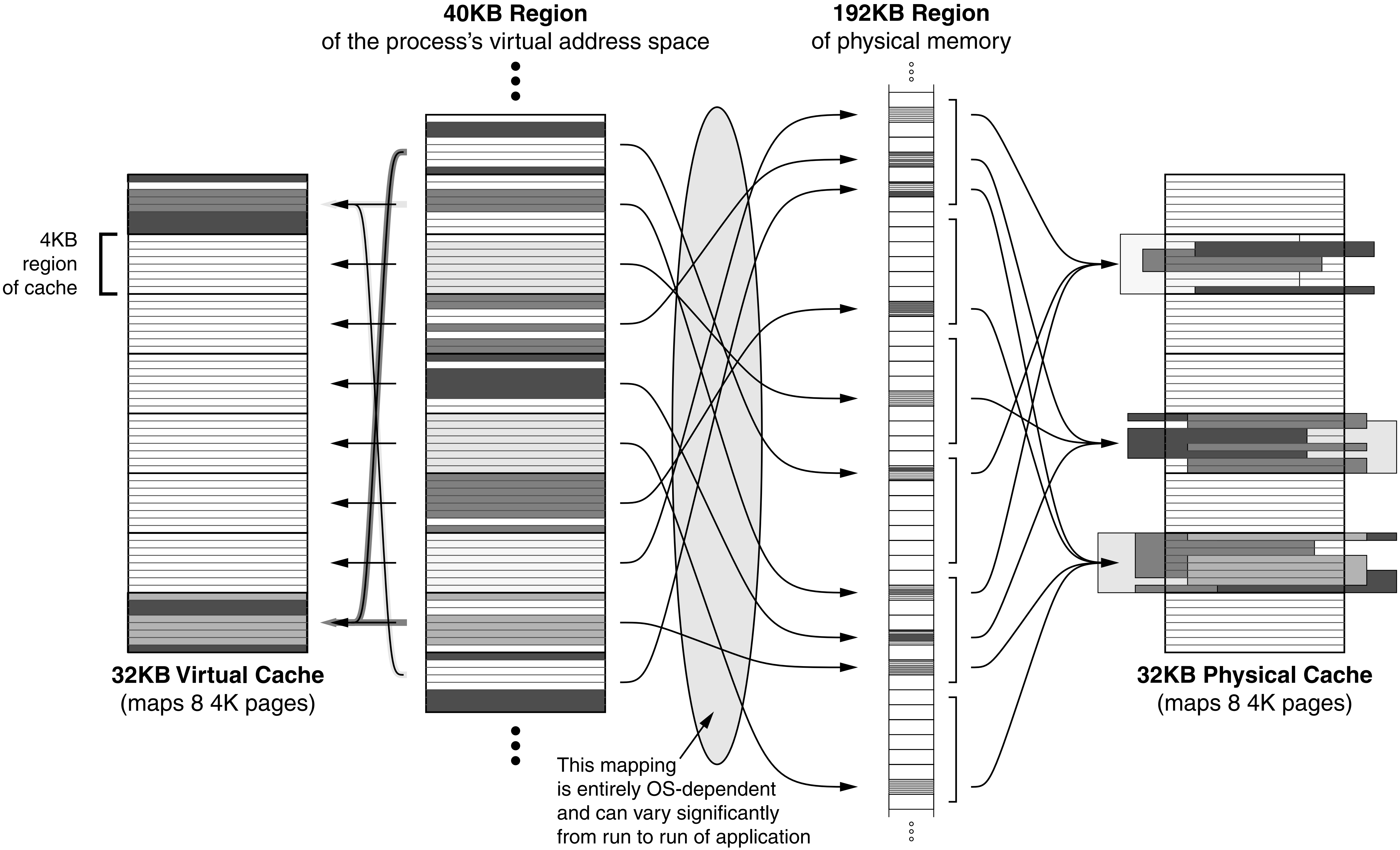
- On-line Heuristics
  (operate at run time)
- Off-line Heuristics
  (operate at design/compile time)
- Combined Heuristics
  (both: e.g., profile-directed)

- Partitioning Heuristics
  (what to cache & not to cache)
- Prefetching Heuristics
  (when to cache it, perhaps early)
- Locality Optimizations
  (rearranging of code & data)

Content Management: Some Examples

- On-line Partitioning: replacement strategies, victim caches
- Off-line Partitioning: scratch-pad management, sleep-mode analysis
- On-line Prefetching: stream buffers, dynamic predictors
- Off-line Prefetching: software prefetching, jump pointers
- On-line Locality: garbage collection, page coloring, dynamic compression
- Off-line Locality: code- and data-packing algorithms, tiling/blocking

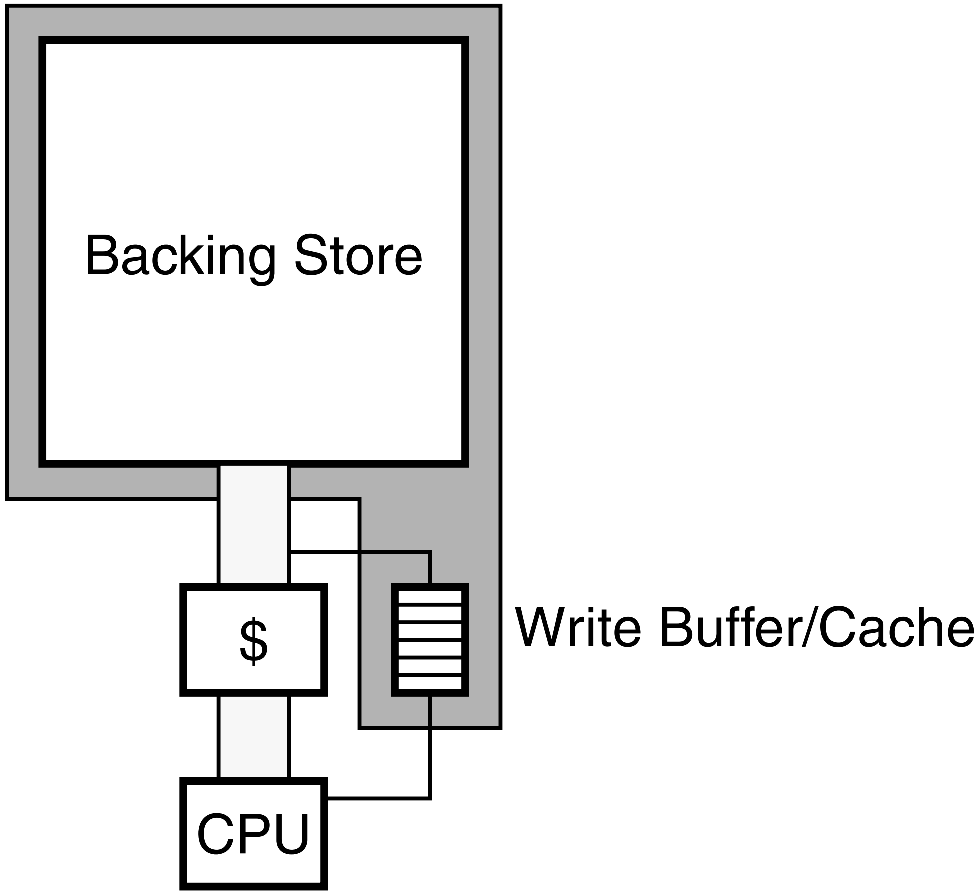
Content Management: Issues



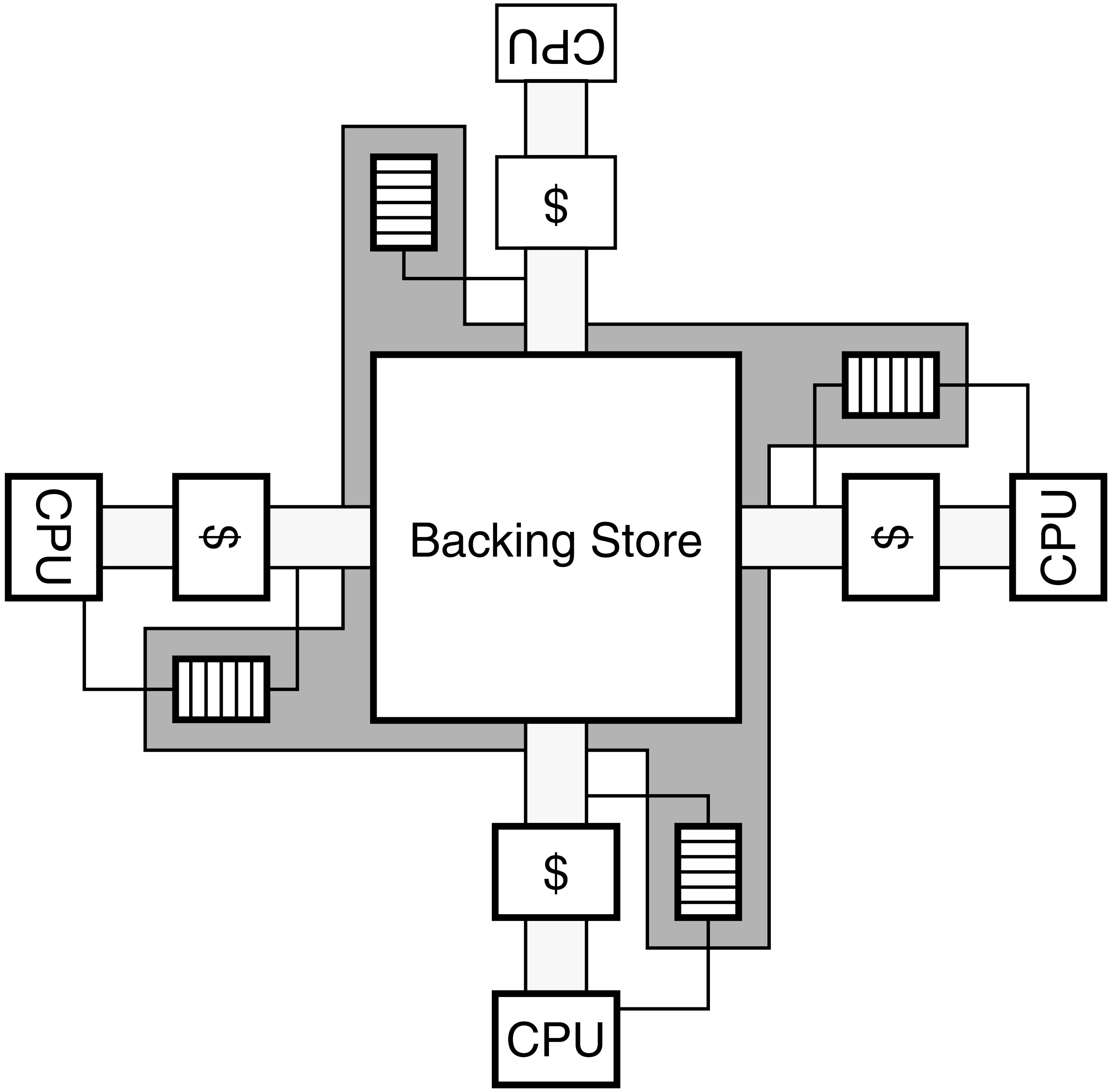
Consistency Management

- Consistency with BACKING STORE
- Consistency with SELF
- Consistency with OTHERS

Consistency Management: BACKING STORE

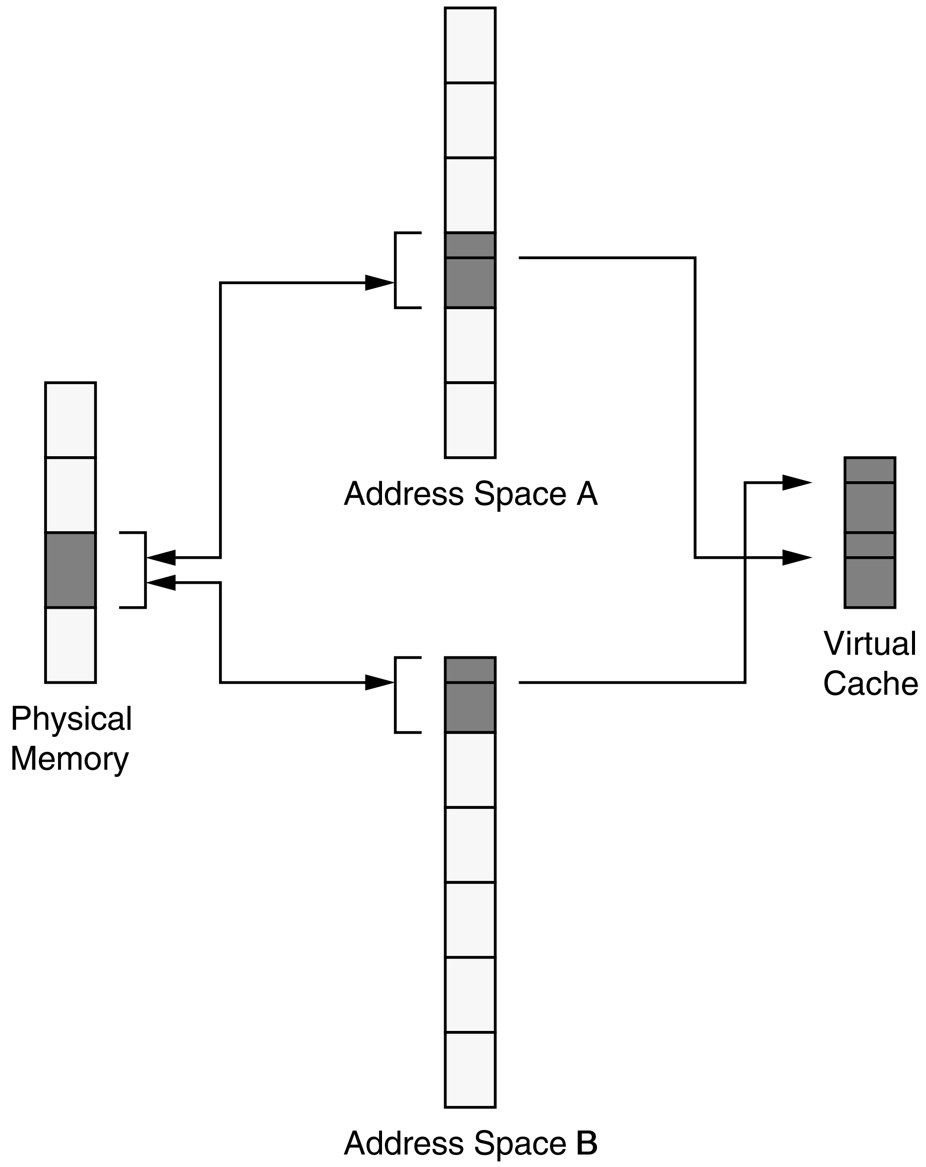


write buffer “in”
backing store



implications for systems
with multiple clients

Consistency Management: SELF



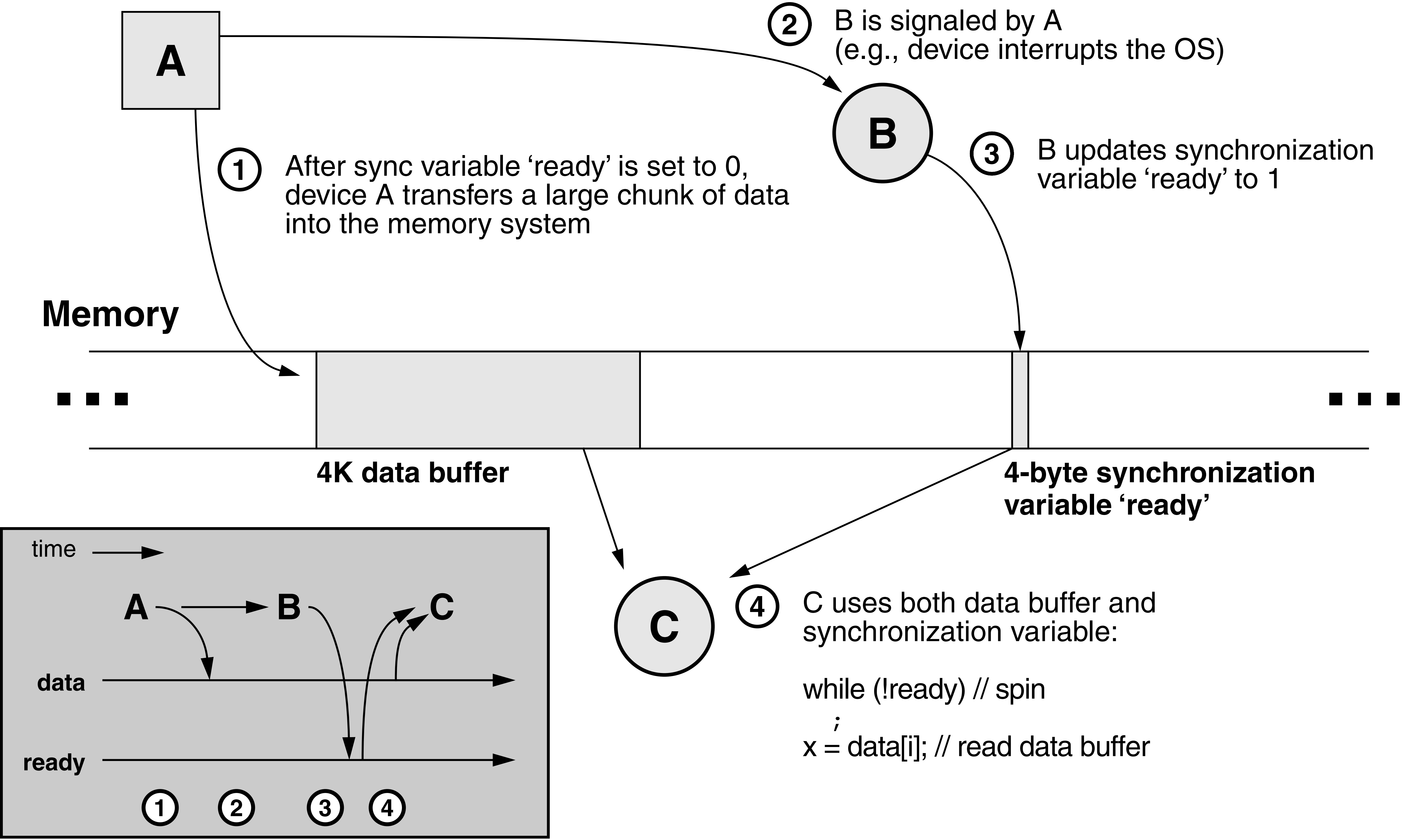
the aliasing problem

There is also an issue with ASIDs, which are used to distinguish the content that belongs to different processes. In general

# procs >> # ASIDS

… ergo constant remapping and reuse. Implication: the OS needs to be careful.

Consistency Management: OTHERS



1. After sync variable 'ready' is set to 0, device A transfers a large chunk of data into the memory system

2. B is signaled by A
(e.g., device interrupts the OS)

3. B updates synchronization variable 'ready' to 1

4. C uses both data buffer and synchronization variable:

```
while (!ready) // spin
     ;
x = data[i]; // read data buffer
```

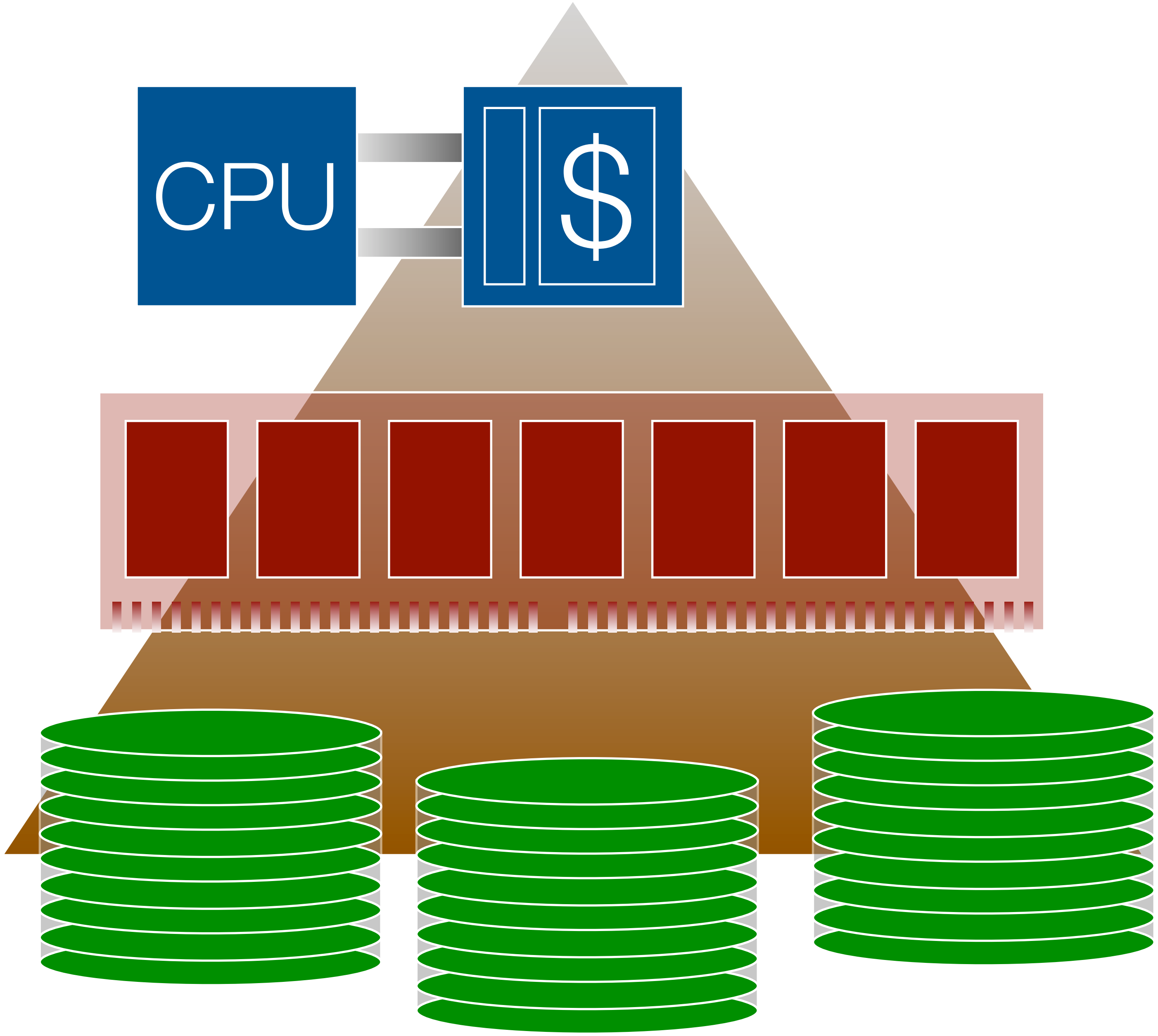
DRAM

Perspective

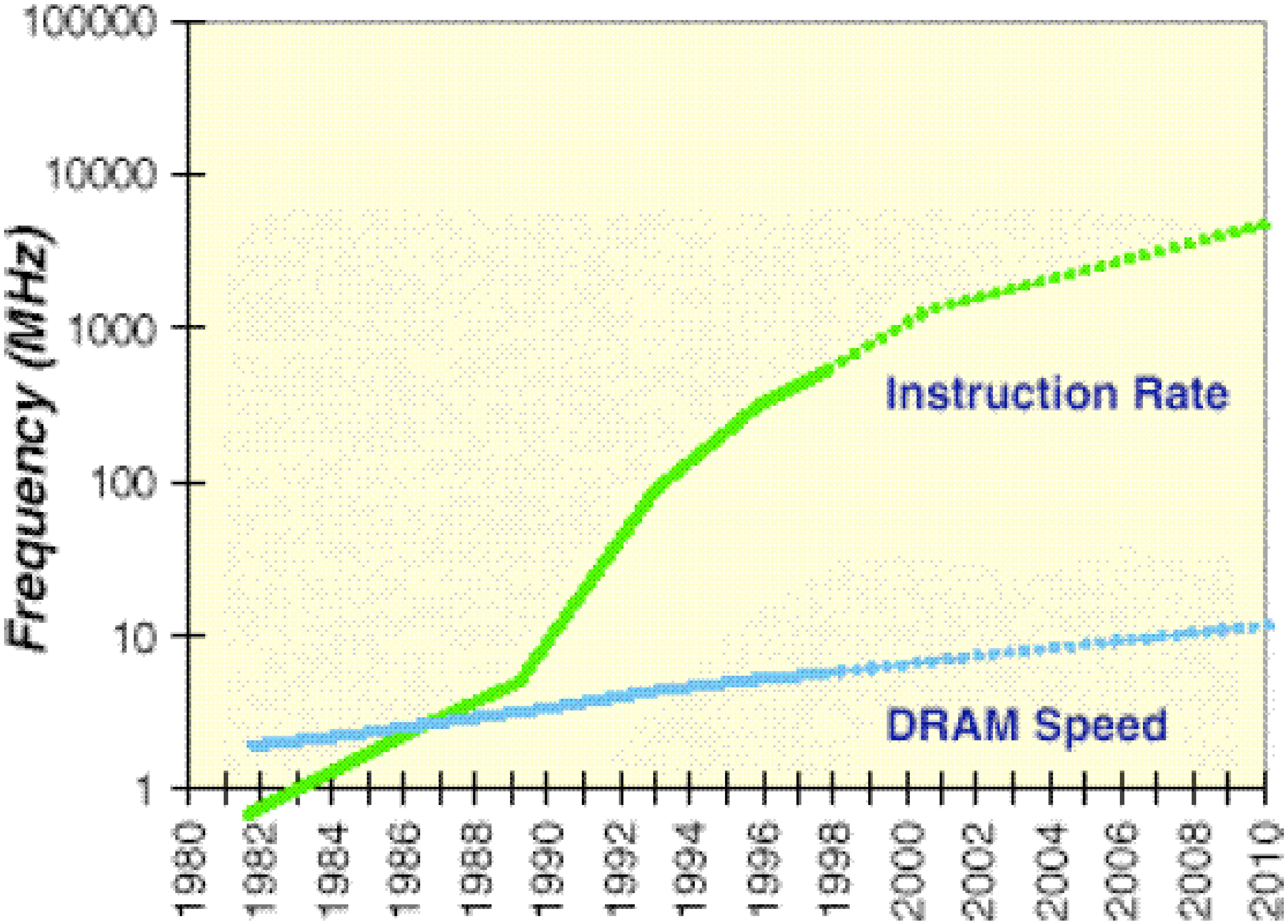
DDRx@800Mbps = 6.4GB/s
(x4 DRAM part: 400MB/s,
100mA, 200mW)

Entry system: 2x 3GHz CPU
(2MB cache each), 1GB DRAM,
80GB disk (7.2K)

CPU = $300
DIMM = $30
DRAM = $3



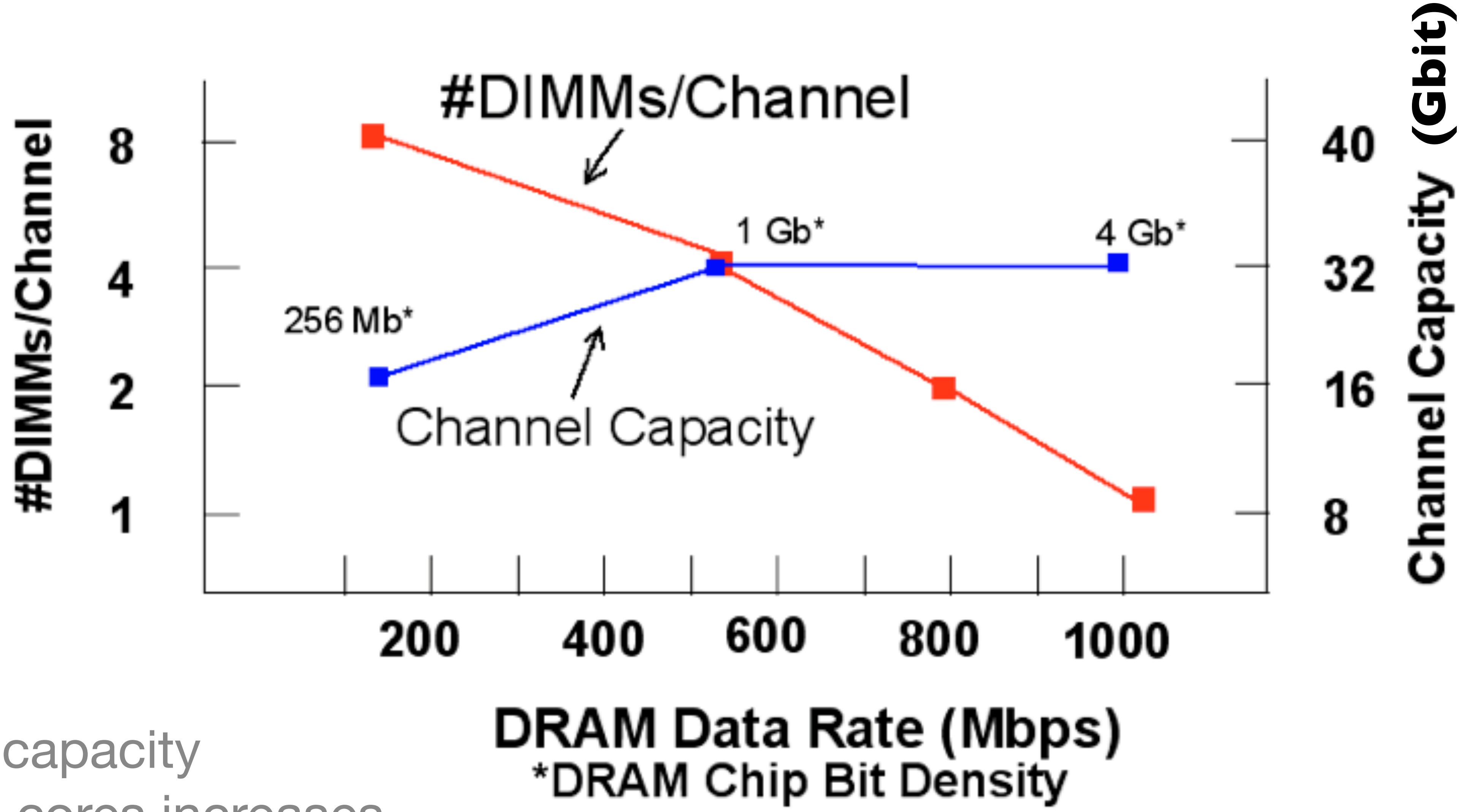
Some Trends



Jean-Luc Gaudiot: *Area and System Clock Effects on SMT/CMP Processors*, 2002.

Some Trends

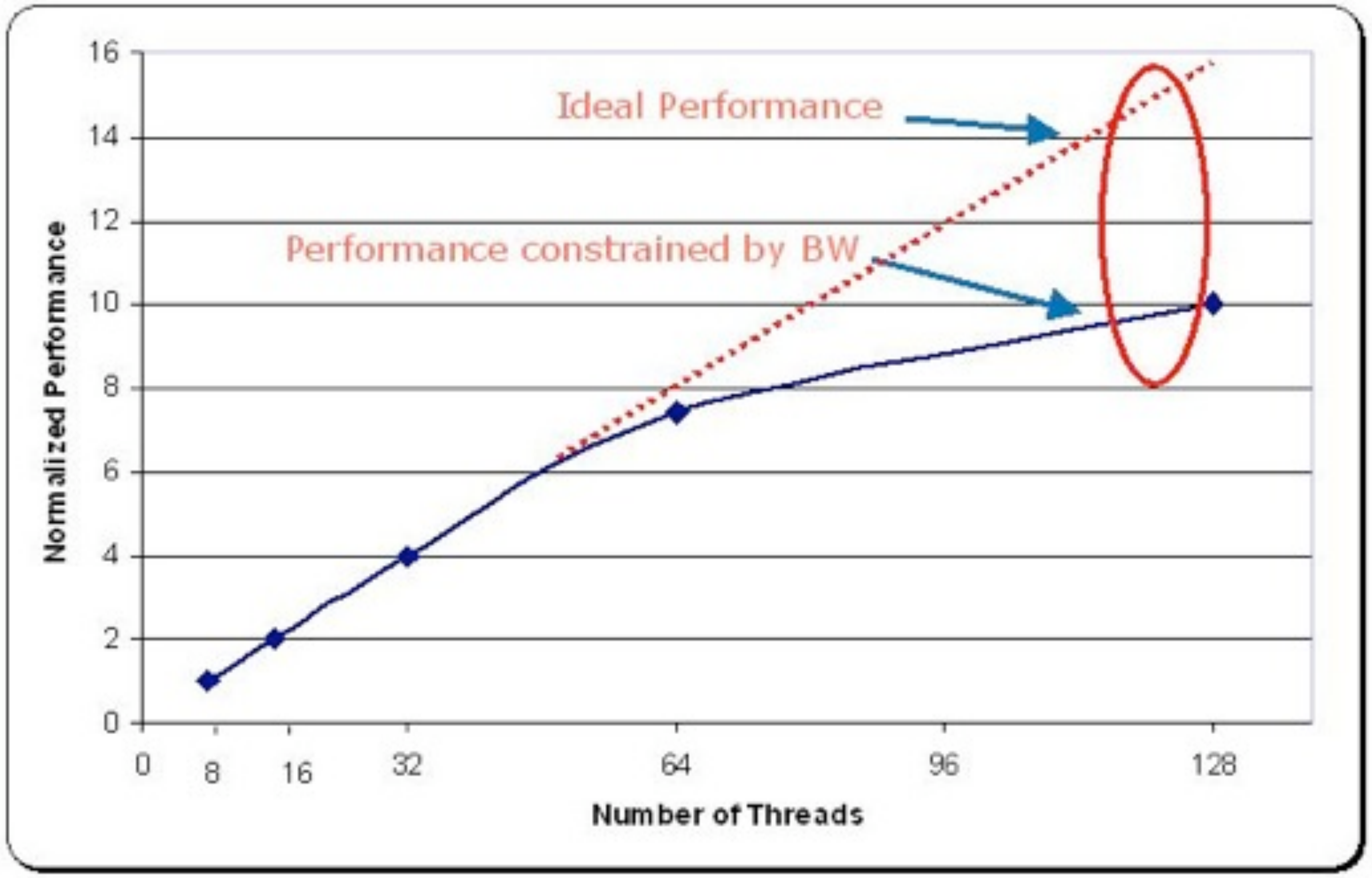
- Storage per CPU socket has been relatively flat for a while



- Note: per-core capacity decreases as # cores increases

Some Trends

- **Required BW per core is roughly 1 GB/s**
- Thread-based load (SPECjbb), memory set to 52GB/s sustained
- Saturates around 64 cores/threads (~1GB/s per core)
- cf. 32-core Sun Niagara: saturates at 25.6 GB/s



Some Trends

Commodity Systems:

- Low double-digit GB per CPU socket
- $10–100 per DIMM

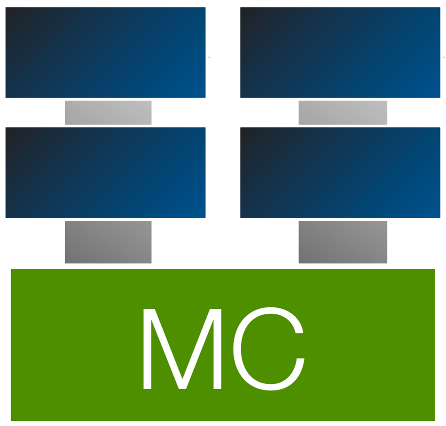
High End:

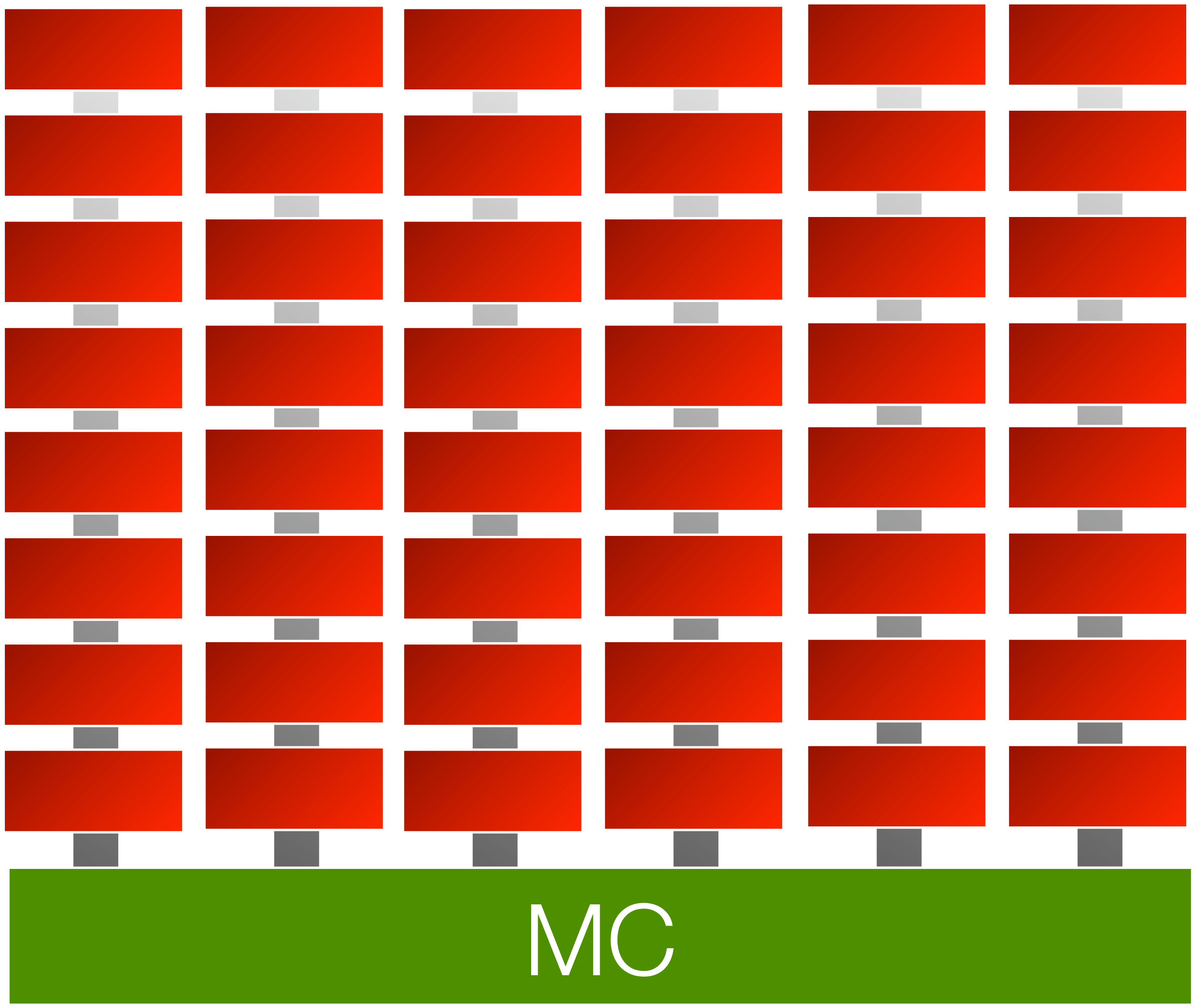
- Higher (but still not *high*)
  double-digit GB per CPU socket
- ~ $1000 per DIMM

Fully-Buffered DIMM:

- (largely failed) attempt to bridge the gap …

Fully Buffered DIMM

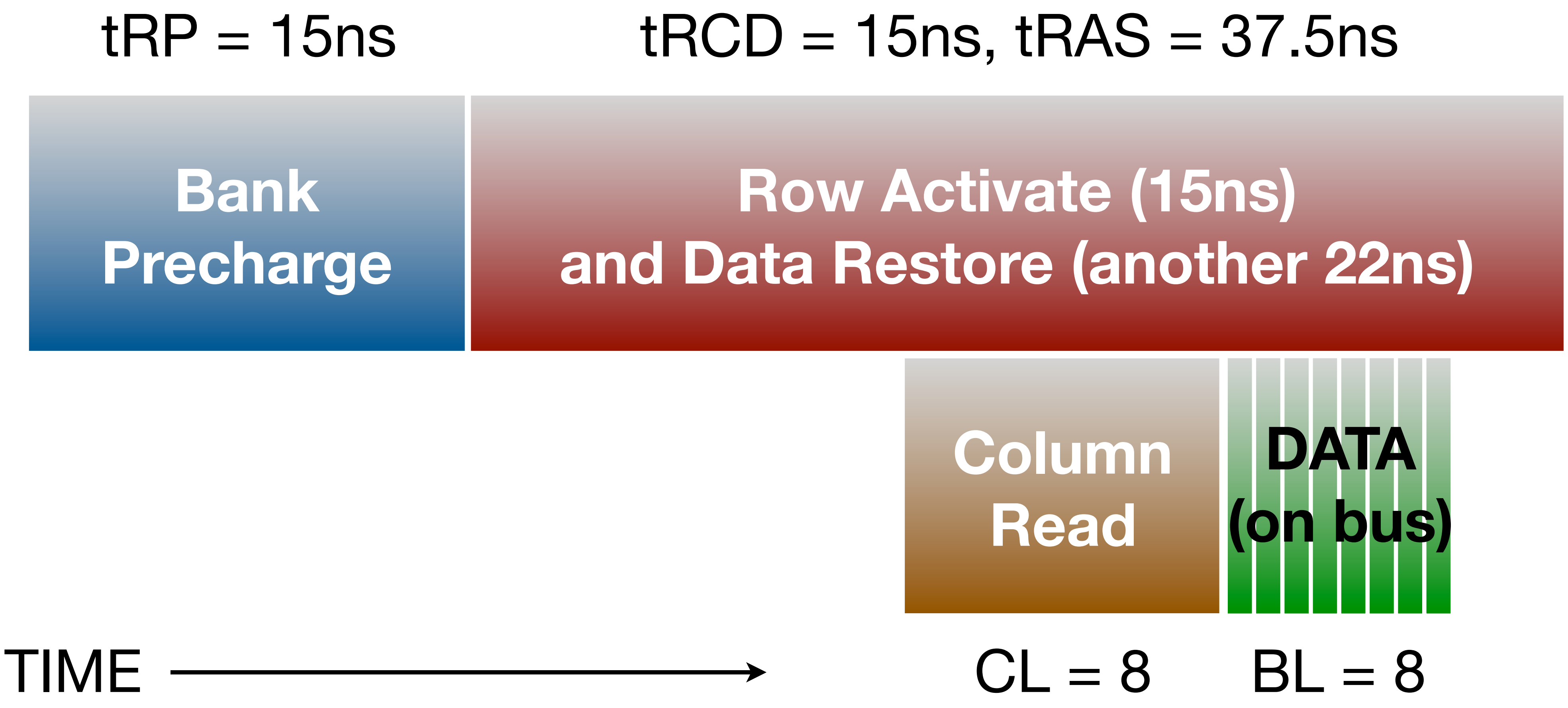




JEDEC DDRx
~10W/DIMM, 20 total

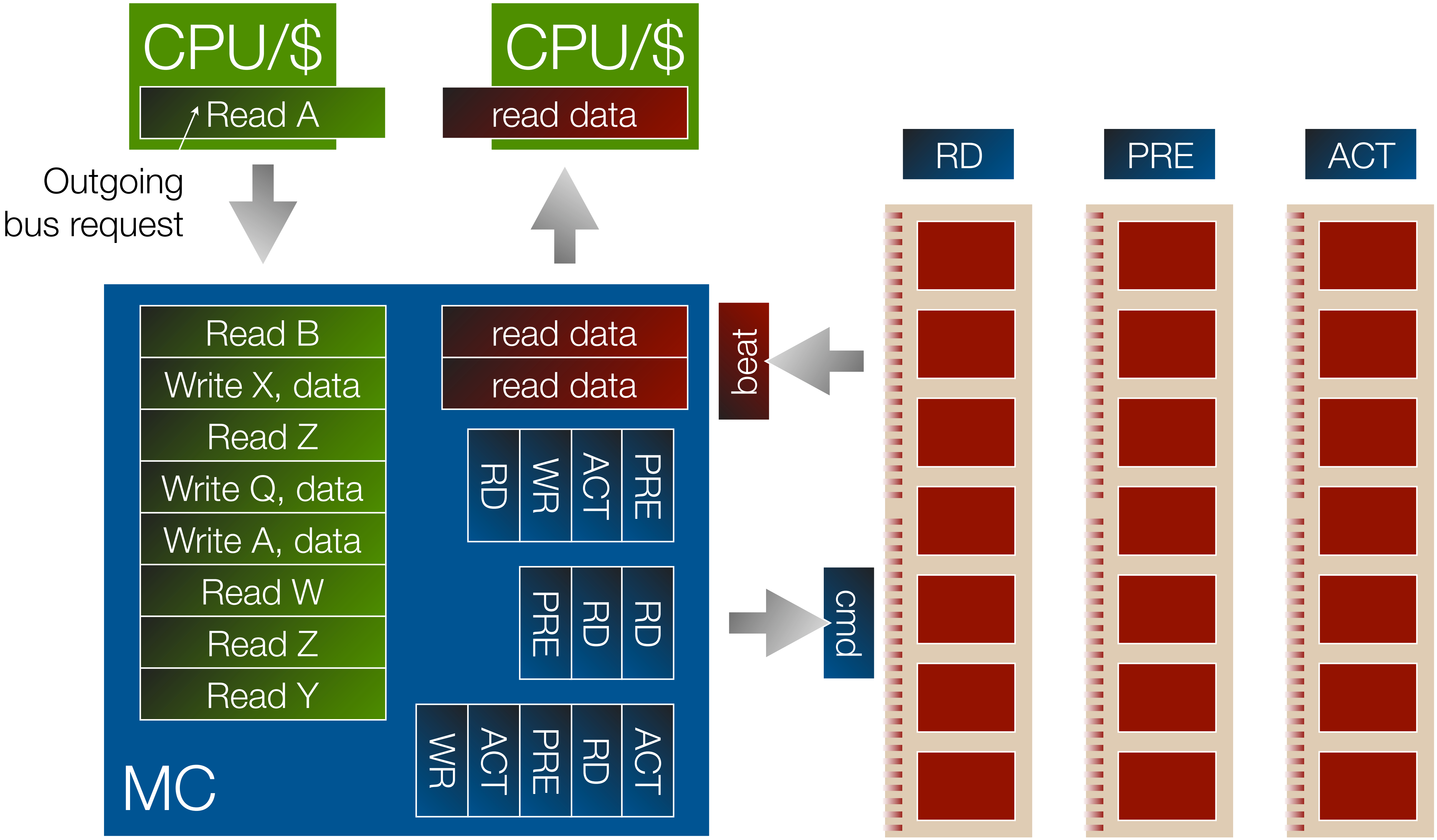
FB-DIMM
~10W/DIMM, ~400W total

The Root of the Problem


tRP = 15ns
tRCD = 15ns, tRAS = 37.5ns
Bank Precharge
Row Activate (15ns) and Data Restore (another 22ns)
Column Read
DATA (on bus)
TIME
CL = 8
BL = 8

Cost of access is high; requires **significant effort** to amortize this over the (increasingly short) payoff.

“Significant Effort”



CPU/$

Read A

Outgoing
bus request

CPU/$

read data

RD

PRE

ACT

Read B

Write X, data

Read Z

Write Q, data

Write A, data

Read W

Read Z

Read Y

read data

read data

RD WR ACT PRE

PRE RD RD

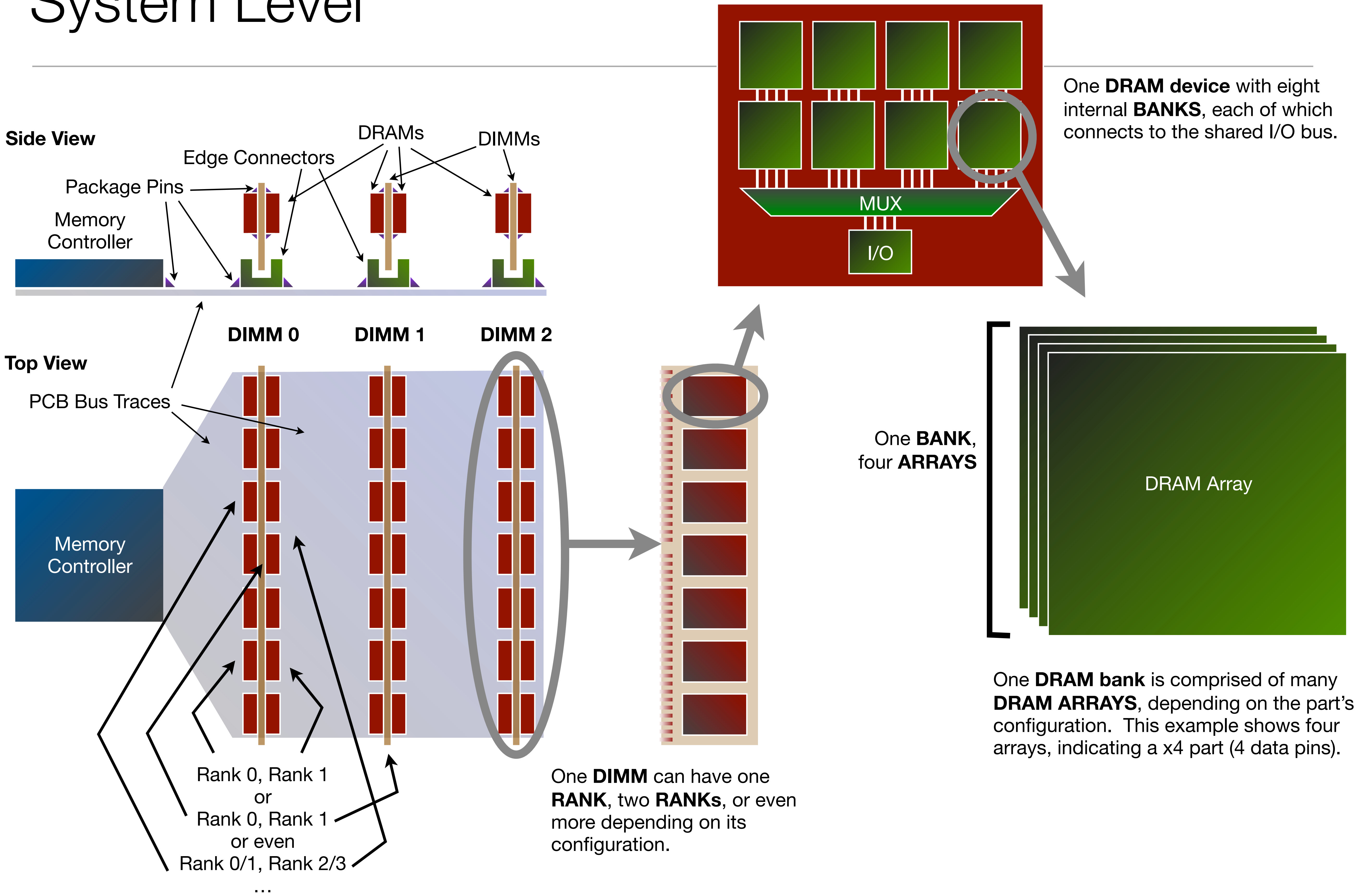
WR ACT PRE RD ACT

MC

beat

cmd

System Level

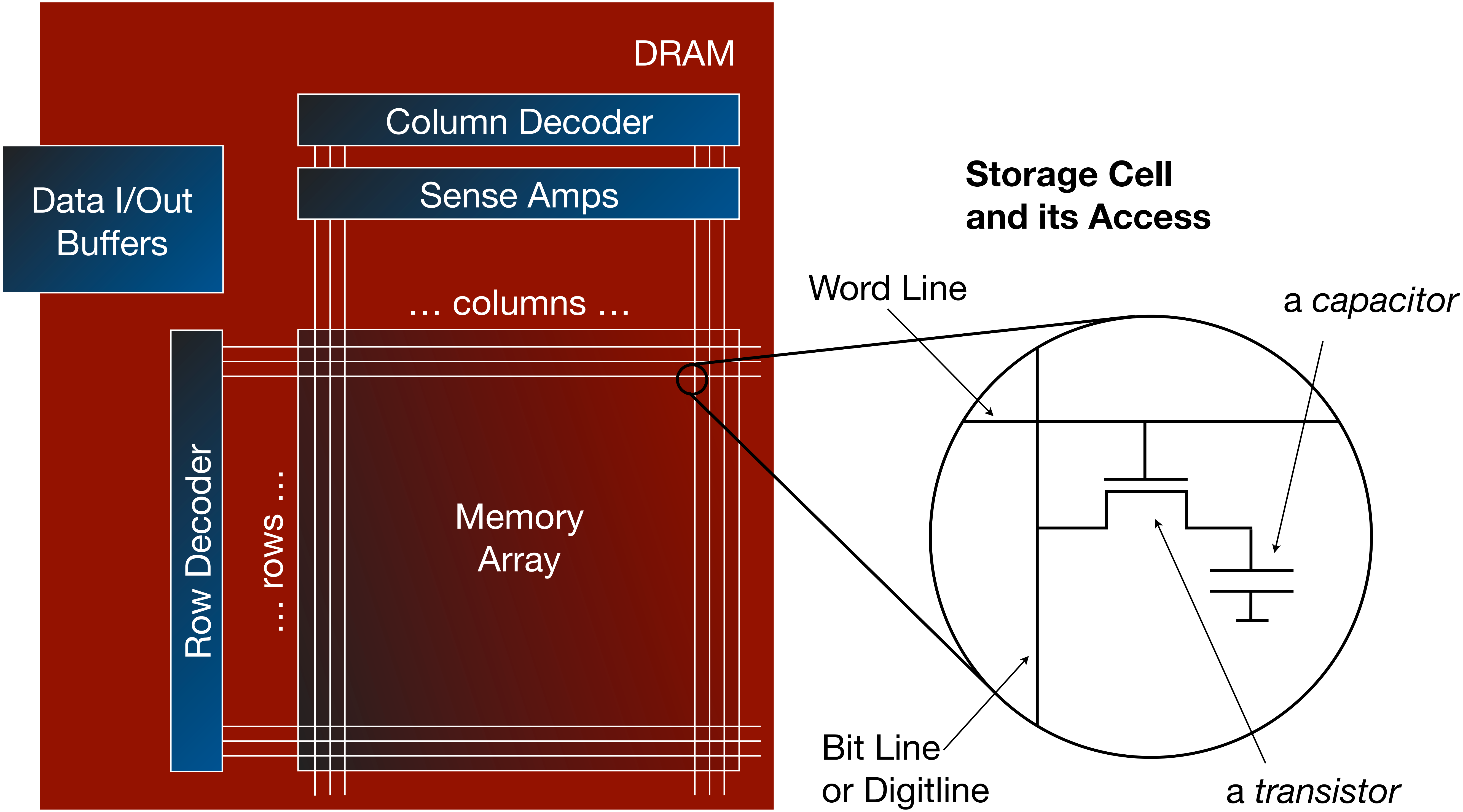


One **DRAM device** with eight internal **BANKS**, each of which connects to the shared I/O bus.

One **DIMM** can have one **RANK**, two **RANKs**, or even more depending on its configuration.

One **DRAM bank** is comprised of many **DRAM ARRAYS**, depending on the part’s configuration. This example shows four arrays, indicating a x4 part (4 data pins).

Device Level



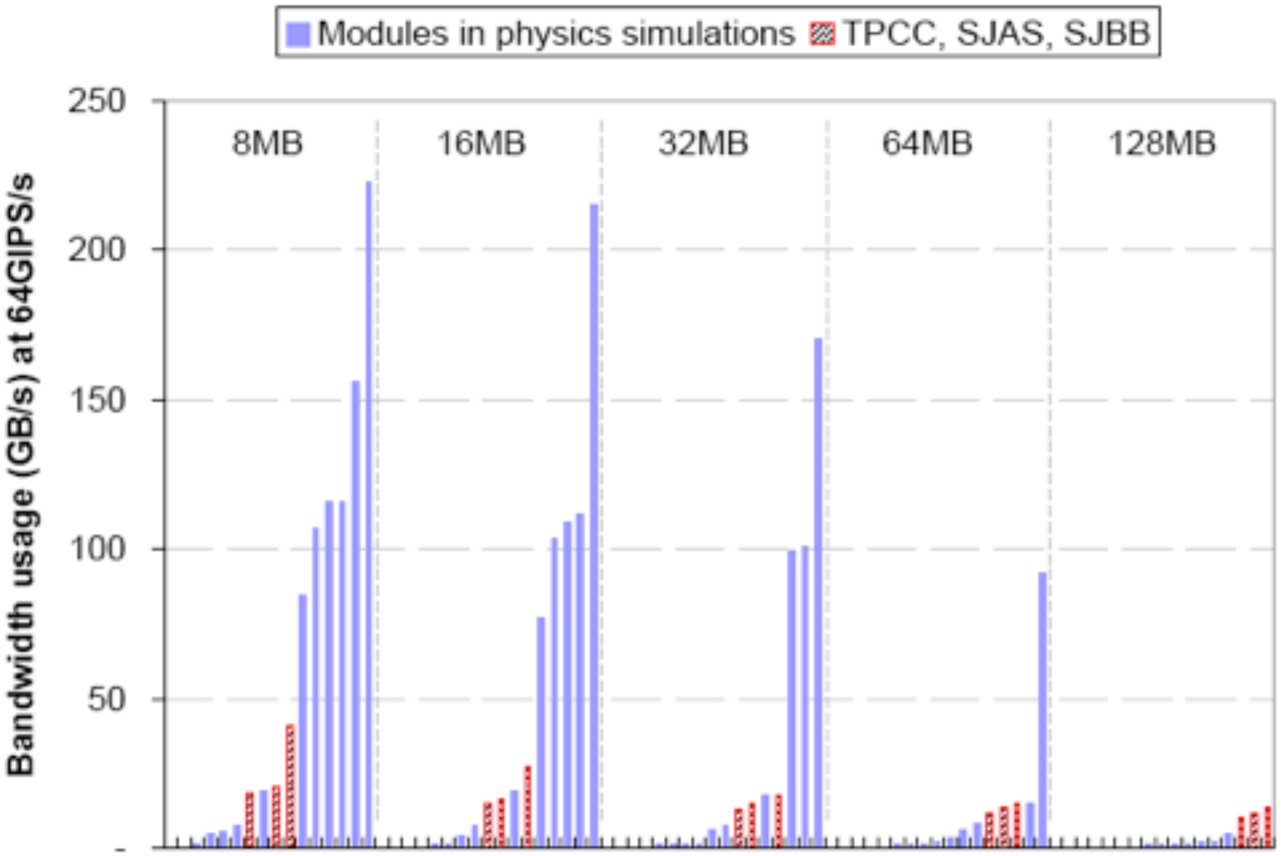
Issues: Palm HD

- 1920 x 1080 x 36b
  x 60fps = 560MB/s (~1GB/s incl. ovhd)

- 3 x4 DDR800 = 1.2GB/s, 600mW

- Power budget = 500mW total (DRAM 10–20%)

Issues



Intel Technology Journal:11(3), August 2007

Cache-Bound ≤ **10M***
Much SPECint (not all), etc.
Embedded: mp3 playback

DRAM-Bound ≤ **10G***
SpecJBB, SPECfp, SAP, etc.
Embedded: HD video

Disk-Bound ≥ **10G***
TPCC, Google

* Desktop; scale down for embedded

Issues: Cost is Primary Limiter

- CPUs: die area (& power)
  Systems: **pins & power**
  (desktop: power is *cost*
  embedded: power is *limit*)

- FB-DIMM (Intel's solution to the capacity problem) observed former at cost of latter … *R.I.P. FBD*

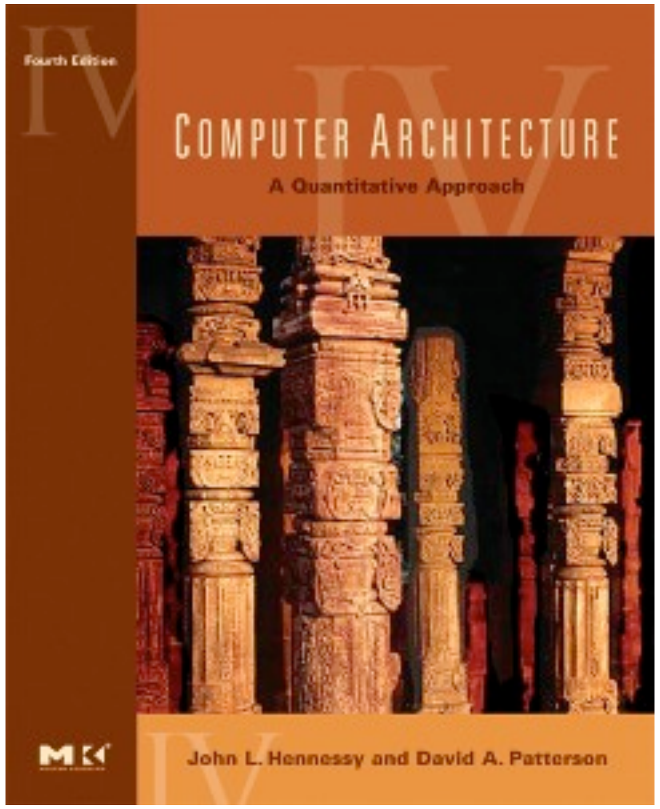
- Whither PERFORMANCE w/o limits? **10x at least**

Issues: Education

```
if (L1(addr) != HIT) {
    if (L2(addr) != HIT) {

        sim += DRAM_LATENCY;

    }
}
```



- Because modeling the memory system is hard,
  few people do it;
  because few do it,
  few understand it

- Memory-system analysis domain of architecture (not circuits)

- Computer designers are enamored w/ CPU
  … *R.I.P. [insert company]*

How It Is Represented

```
if (cache_miss(addr)) {
   cycle_count += DRAM_LATENCY;
}
```

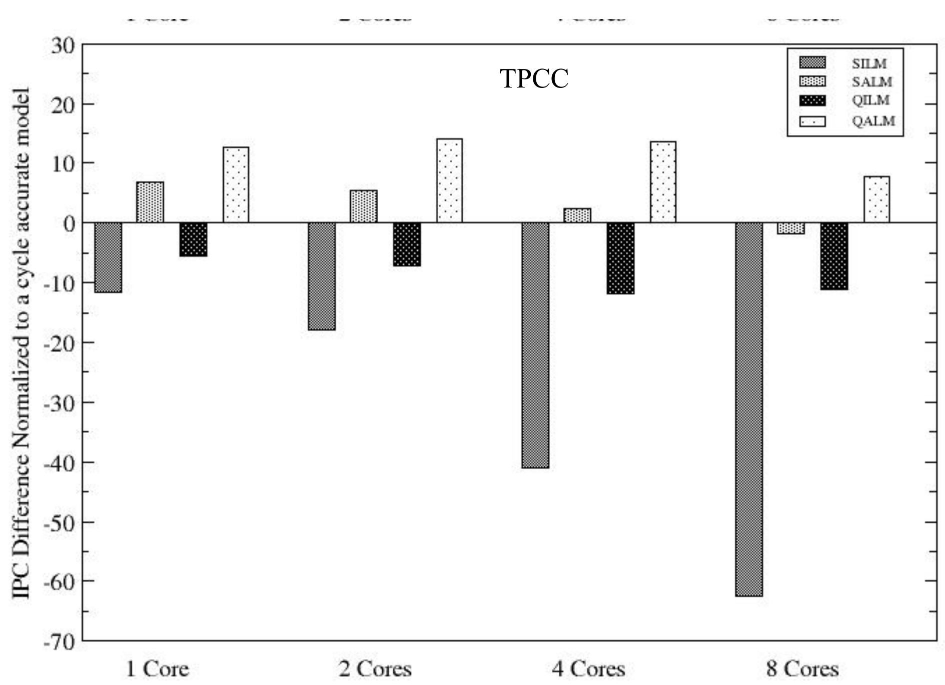
… even in simulators with “cycle accurate” memory systems—no lie

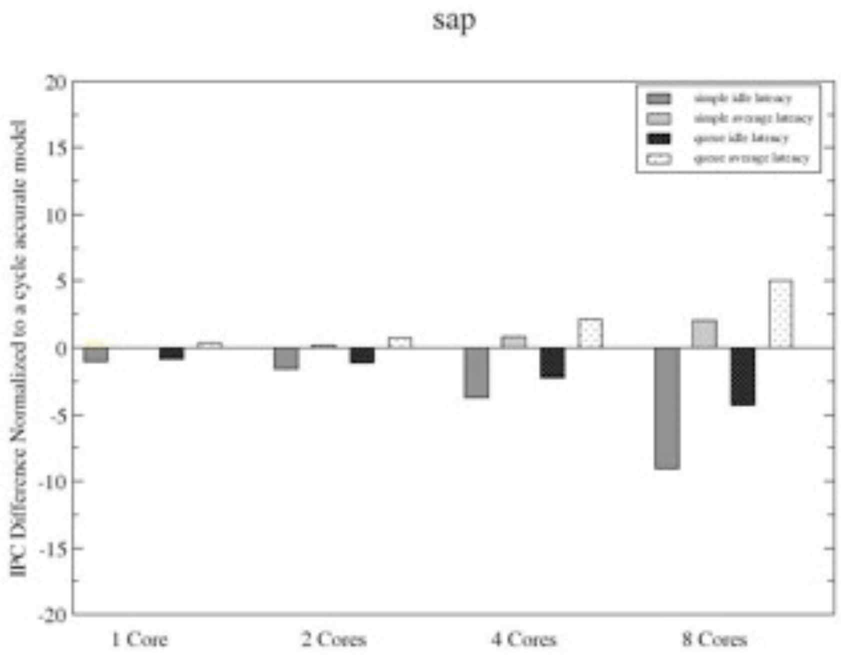
Issues: Accuracy

- Graphs compare
  - fixed latency
  - queueing model (from industry)
  - “real” model

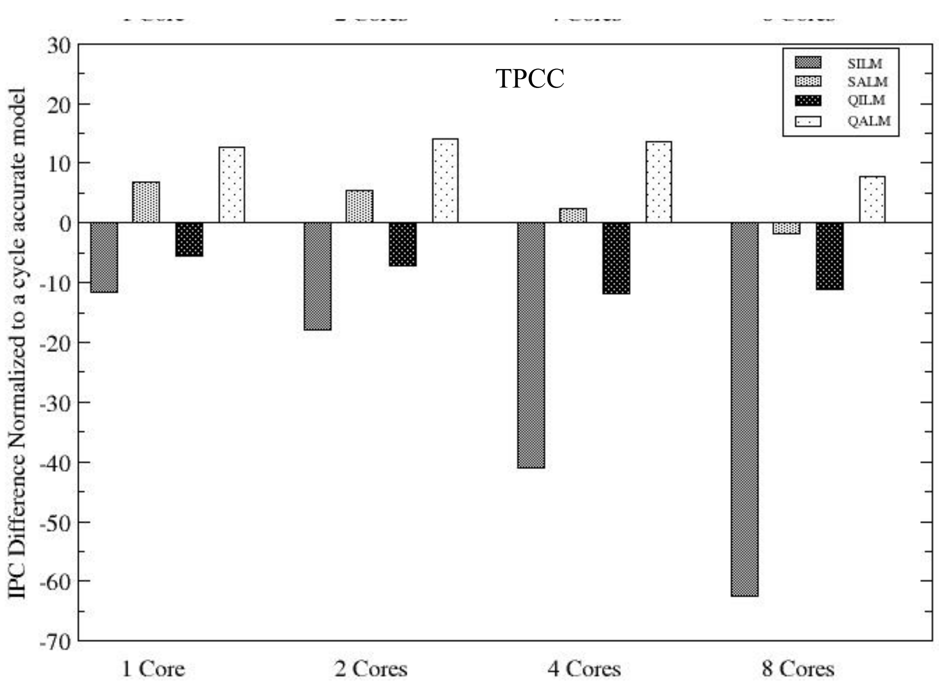
- Using simple models gives inaccurate insights, leads to poor design

- Inaccuracies scale with workload (this is bad)

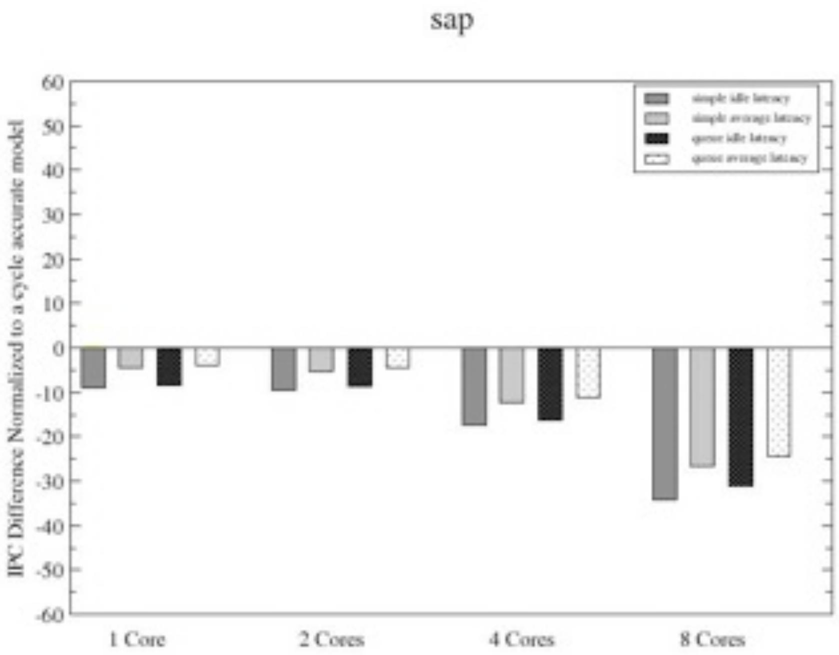


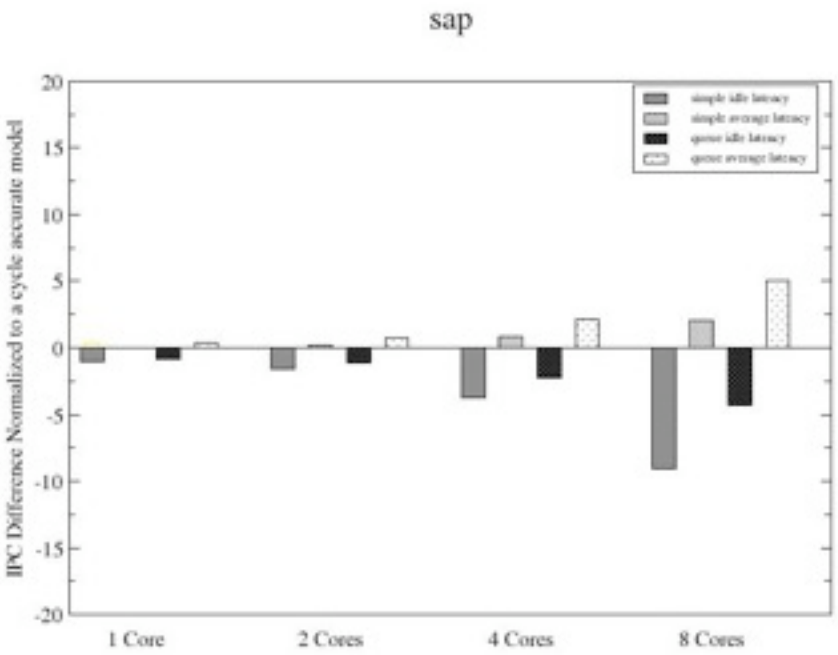


Issues: Accuracy

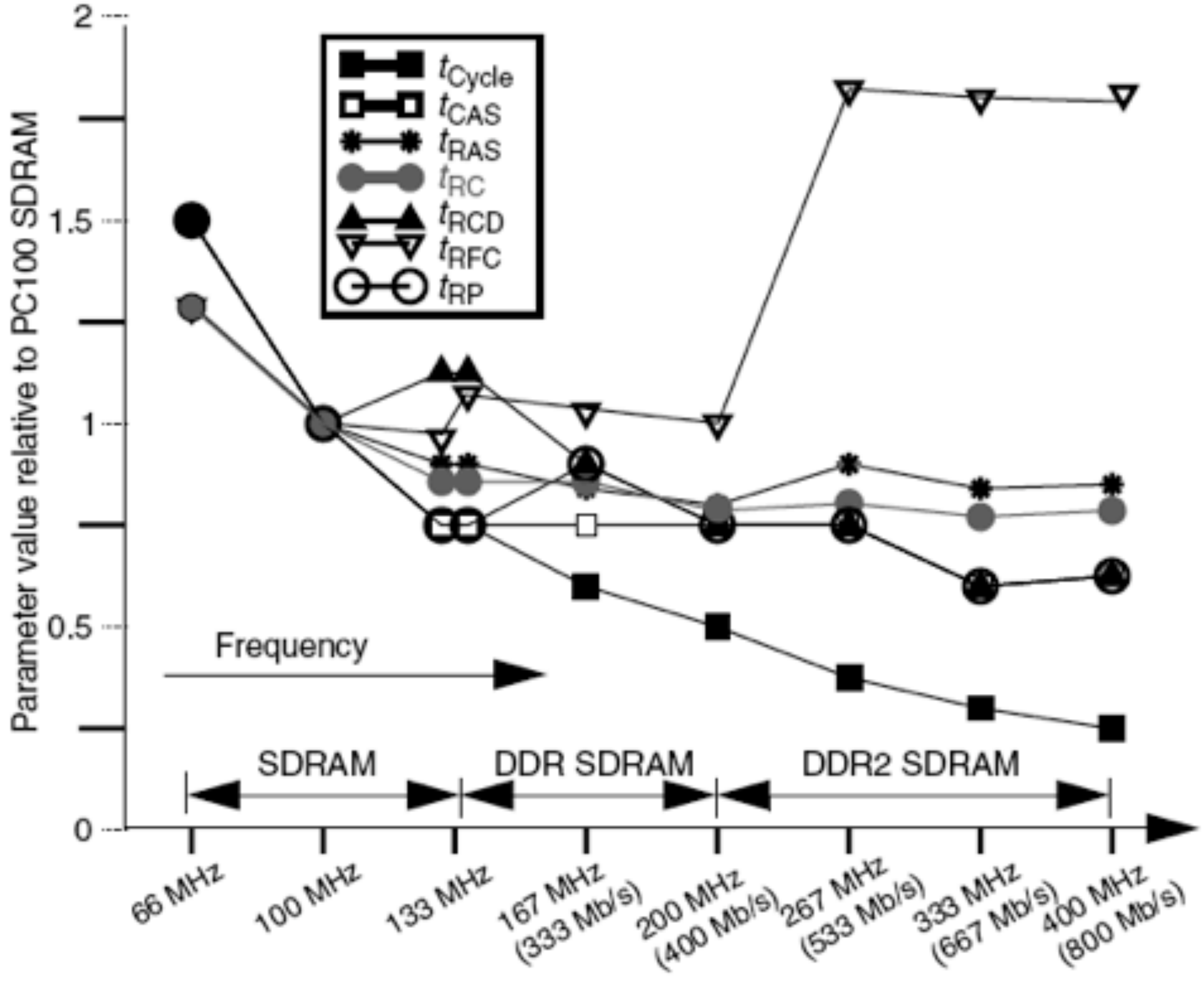


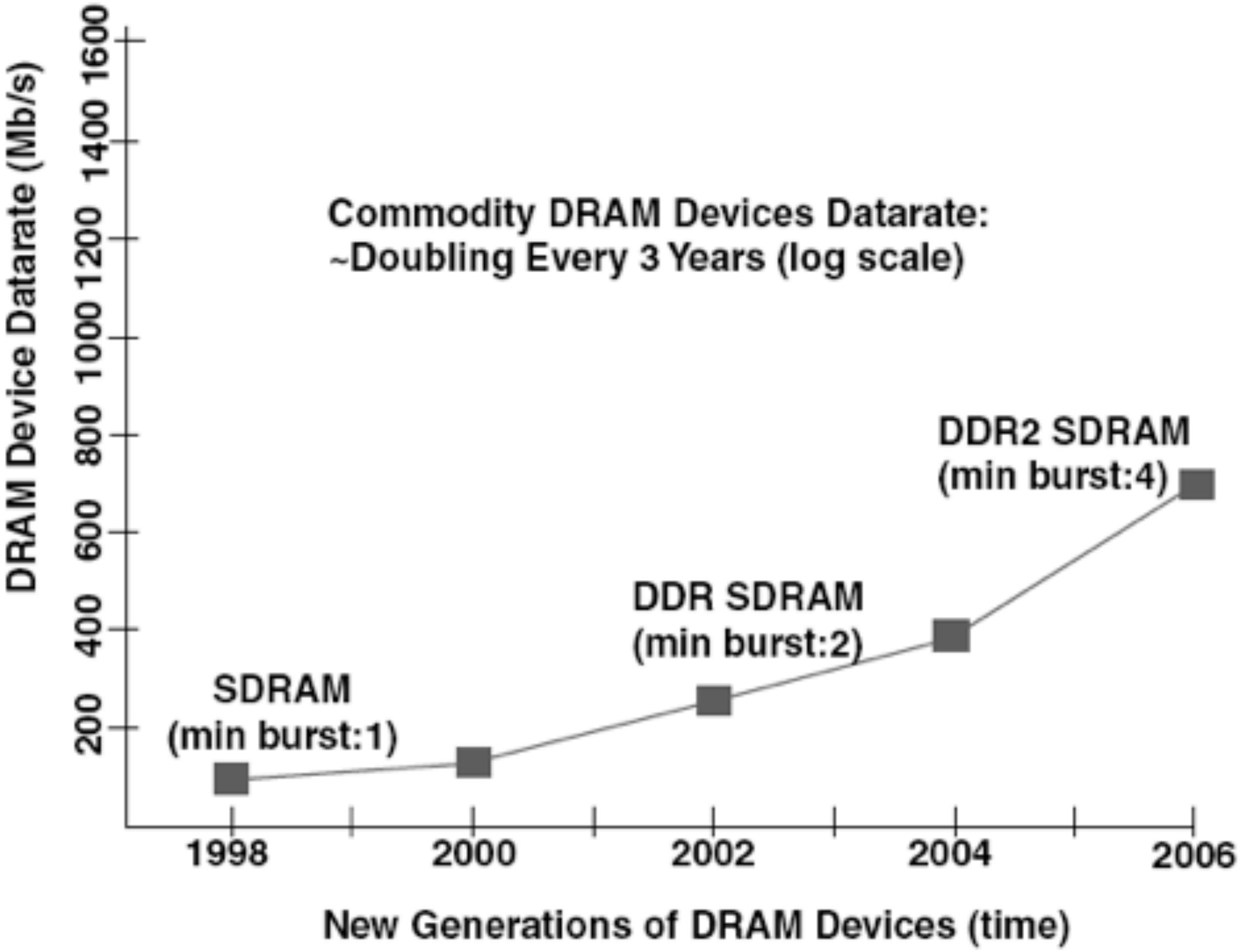
SAP w/ prefetching





Trends …

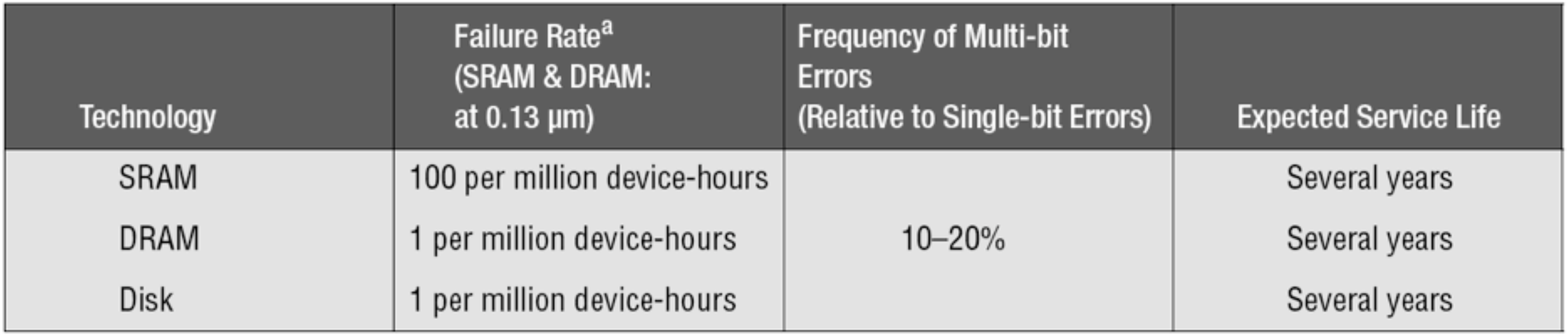




Jacob, Ng, & Wang: Memory Systems, 2007.

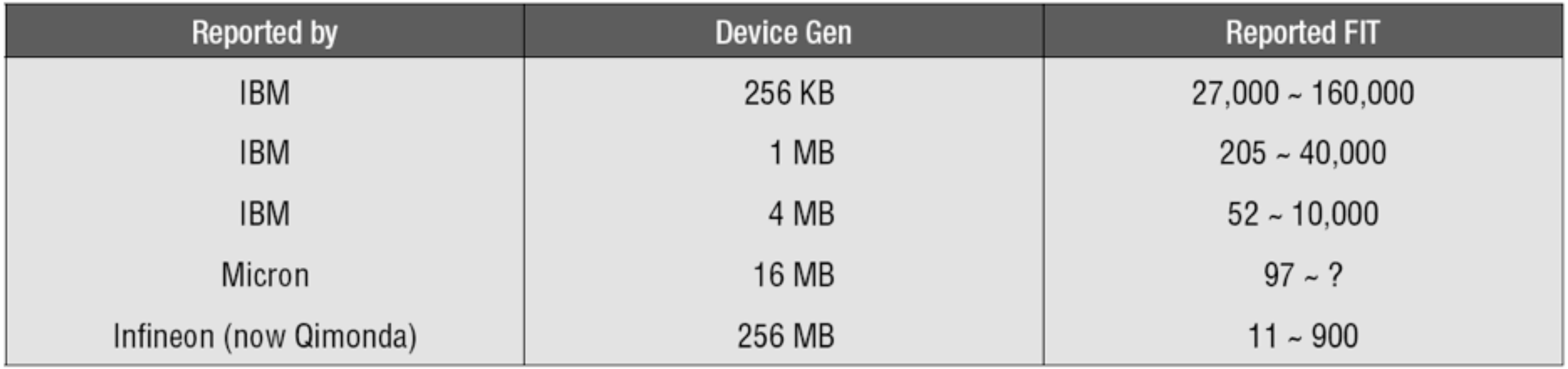
Trends …

TABLE Ov.4 Cross-comparison of failure rates for SRAM, DRAM, and disk



Technology	Failure Rate[a] (SRAM & DRAM: at 0.13 μm)	Frequency of Multi-bit Errors (Relative to Single-bit Errors)	Expected Service Life
SRAM	100 per million device-hours	10–20%	Several years
DRAM	1 per million device-hours		Several years
Disk	1 per million device-hours		Several years

TABLE 30.2 Reported SER (for DRAMs)



Reported by	Device Gen	Reported FIT
IBM	256 KB	27,000 ~ 160,000
IBM	1 MB	205 ~ 40,000
IBM	4 MB	52 ~ 10,000
Micron	16 MB	97 ~ ?
Infineon (now Qimonda)	256 MB	11 ~ 900

Jacob, Ng, & Wang: Memory Systems, 2007.

Trends …

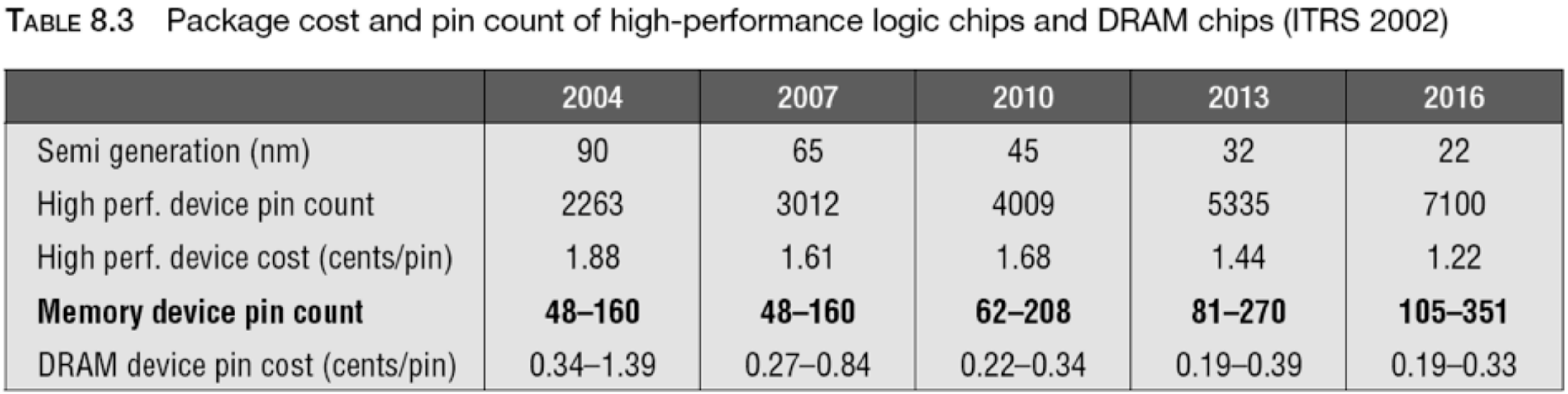


Table 8.3 Package cost and pin count of high-performance logic chips and DRAM chips (ITRS 2002)

	2004	2007	2010	2013	2016
Semi generation (nm)	90	65	45	32	22
High perf. device pin count	2263	3012	4009	5335	7100
High perf. device cost (cents/pin)	1.88	1.61	1.68	1.44	1.22
Memory device pin count	**48–160**	**48–160**	**62–208**	**81–270**	**105–351**
DRAM device pin cost (cents/pin)	0.34–1.39	0.27–0.84	0.22–0.34	0.19–0.39	0.19–0.33

Jacob, Ng, & Wang: Memory Systems, 2007.

Trends …

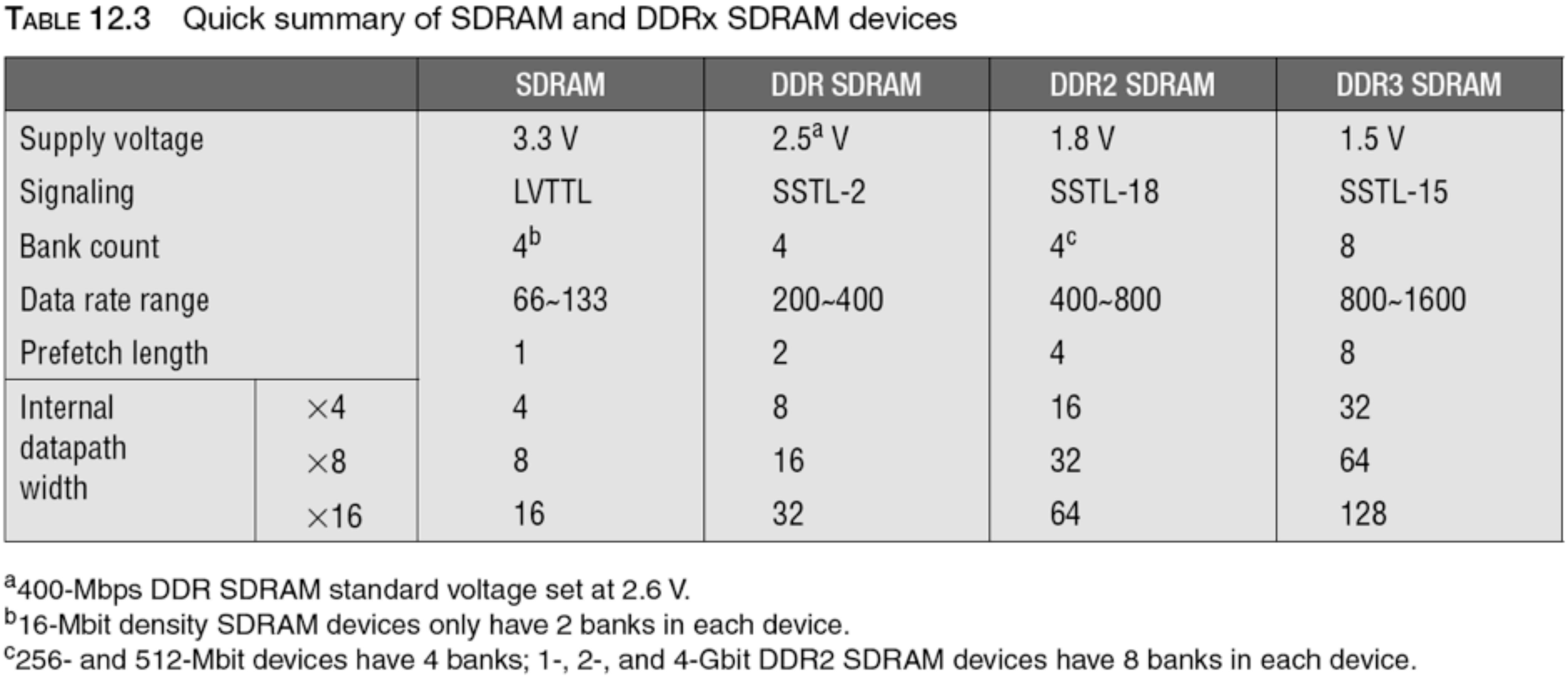


TABLE 12.3 Quick summary of SDRAM and DDRx SDRAM devices

		SDRAM	DDR SDRAM	DDR2 SDRAM	DDR3 SDRAM
Supply voltage		3.3 V	2.5[a] V	1.8 V	1.5 V
Signaling		LVTTL	SSTL-2	SSTL-18	SSTL-15
Bank count		4[b]	4	4[c]	8
Data rate range		66~133	200~400	400~800	800~1600
Prefetch length		1	2	4	8
Internal datapath width	×4	4	8	16	32
	×8	8	16	32	64
	×16	16	32	64	128

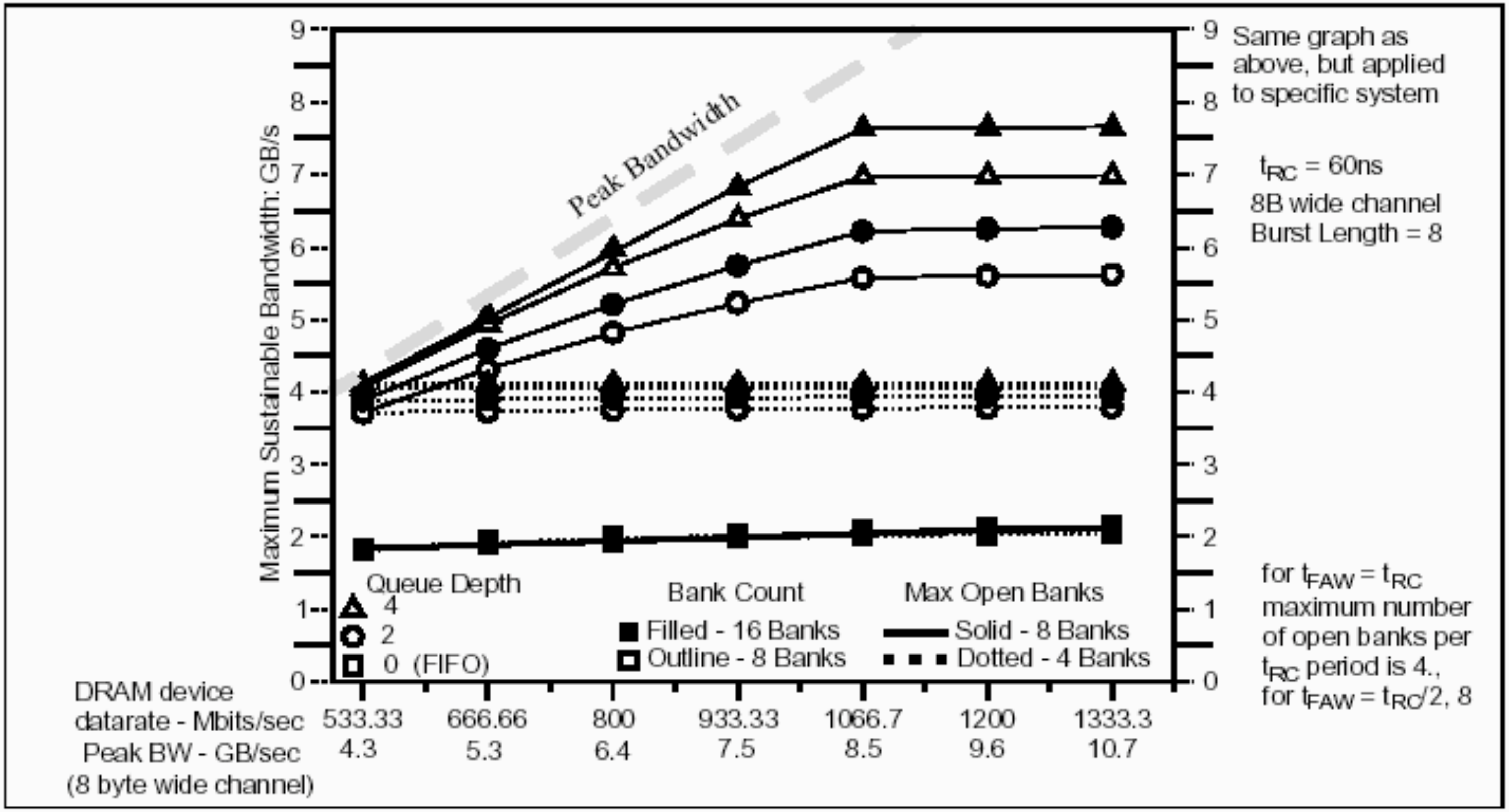
[a] 400-Mbps DDR SDRAM standard voltage set at 2.6 V.

[b] 16-Mbit density SDRAM devices only have 2 banks in each device.

[c] 256- and 512-Mbit devices have 4 banks; 1-, 2-, and 4-Gbit DDR2 SDRAM devices have 8 banks in each device.

Jacob, Ng, & Wang: Memory Systems, 2007.

Trends …



Same graph as above, but applied to specific system

t_{RC} = 60ns
8B wide channel
Burst Length = 8

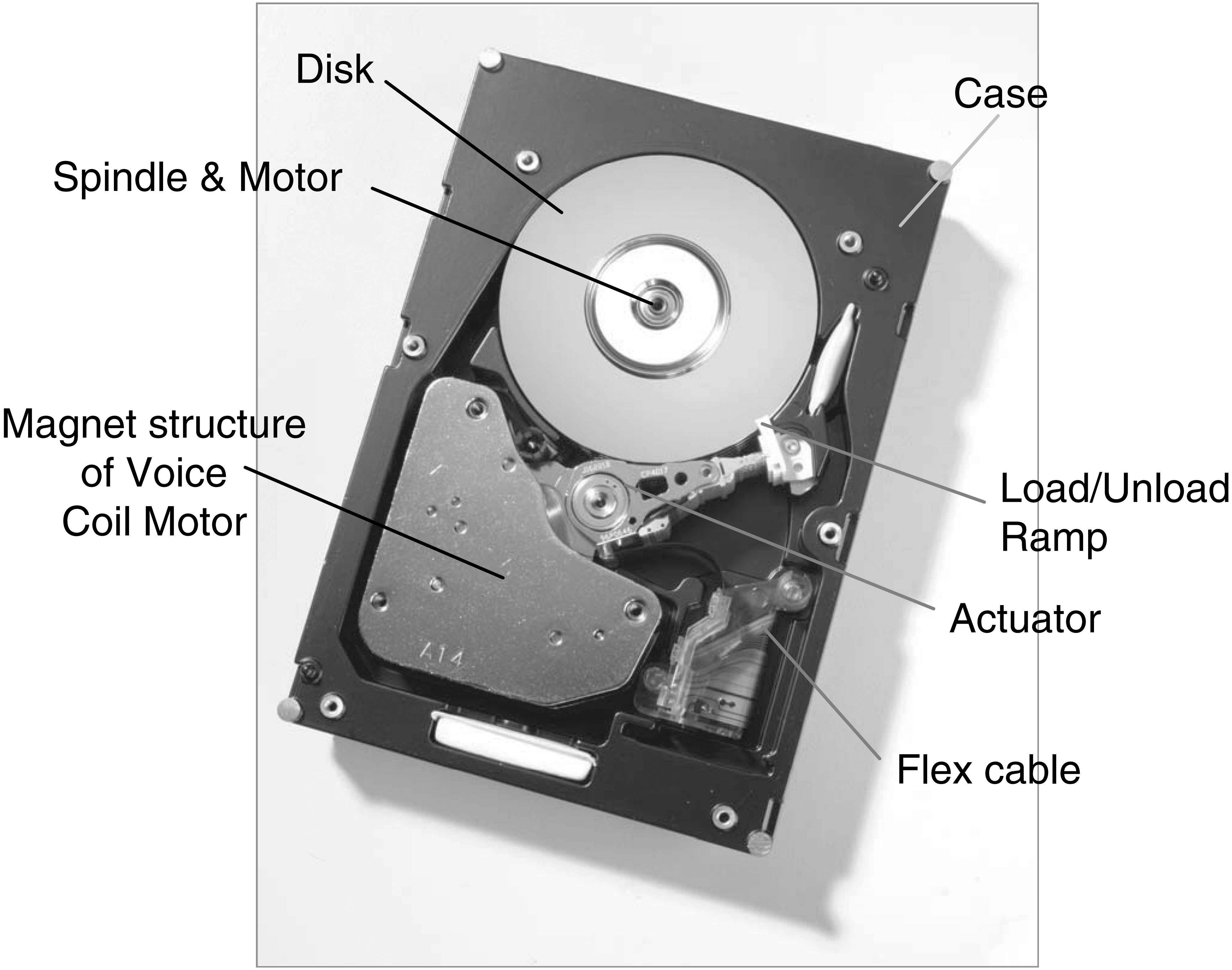
for $t_{FAW} = t_{RC}$ maximum number of open banks per t_{RC} period is 4., for $t_{FAW} = t_{RC}/2$, 8

Figure 7.3: 164.gzip maximum sustainable bandwidth: close-page.

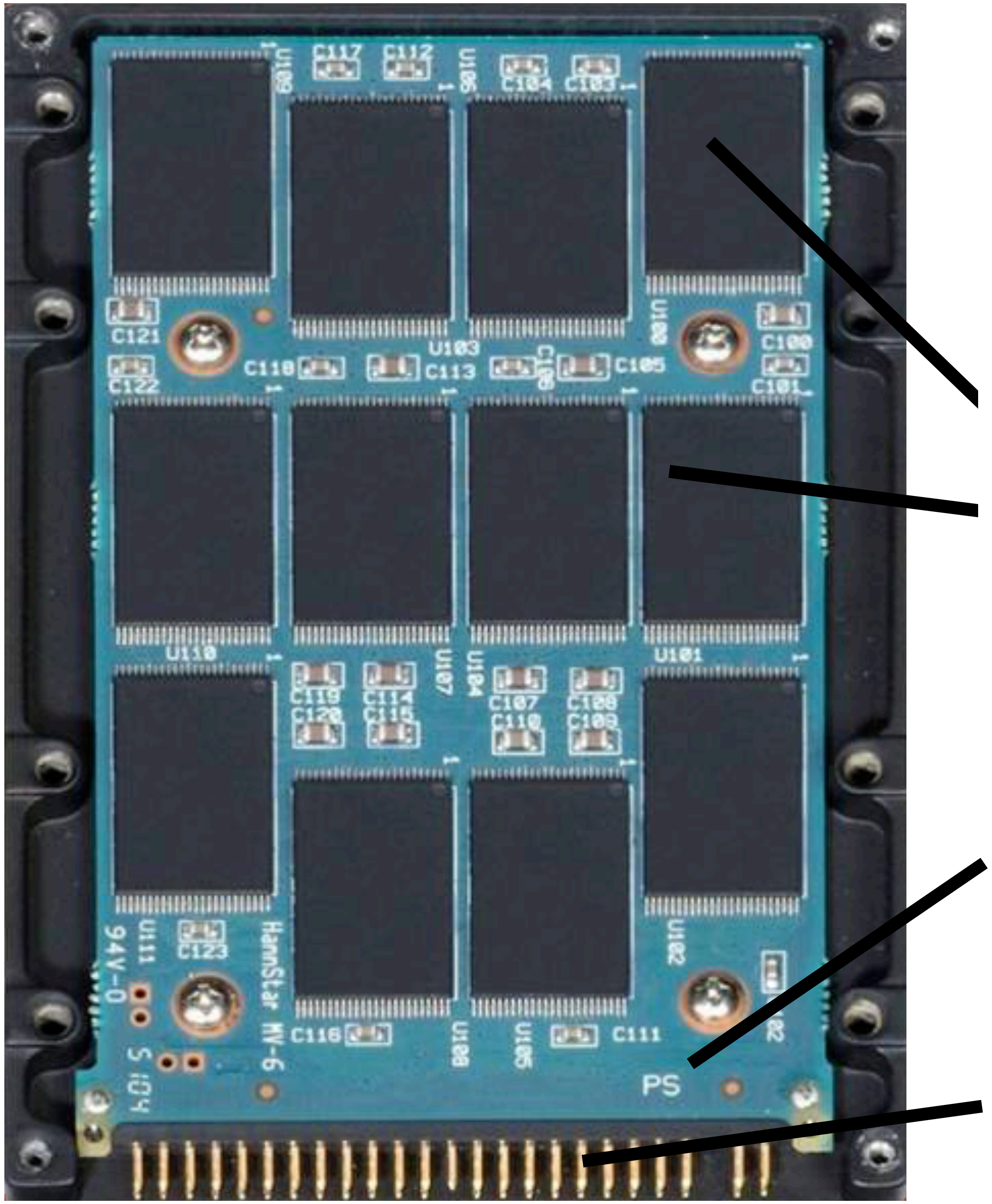
t_{FAW} (& t_{RRD} & t_{DQS}) vs. bandwidth (Dave Wang's thesis)

DISK & FLASH

Disk


Disk
Case
Spindle & Motor
Magnet structure of Voice Coil Motor
Load/Unload Ramp
Actuator
Flex cable

Flash SSD



Flash memory arrays

Circuit board

ATA Interface

Disk Issues

- Keeping ahead of Flash in price-per-GB is difficult (and expensive)
- Dealing with timing in a polar-coordinate system is non-trivial
    - OS schedules disk requests to optimize both linear & rotational latencies; ideally, OS should not have to become involved at that level
- Tolerating long-latency operations creates fun problems
    - E.g., block-fill not atomic; must reserve buffer for duration; Belady's MIN designed for disks & thus does not consider incoming block in analysis
- Internal cache & prefetch mechanisms are slightly behind the times

Flash SSD Issues

- Flash does not allow in-place update of data (must block-erase first); implication is significant amount of garbage collection & storage management

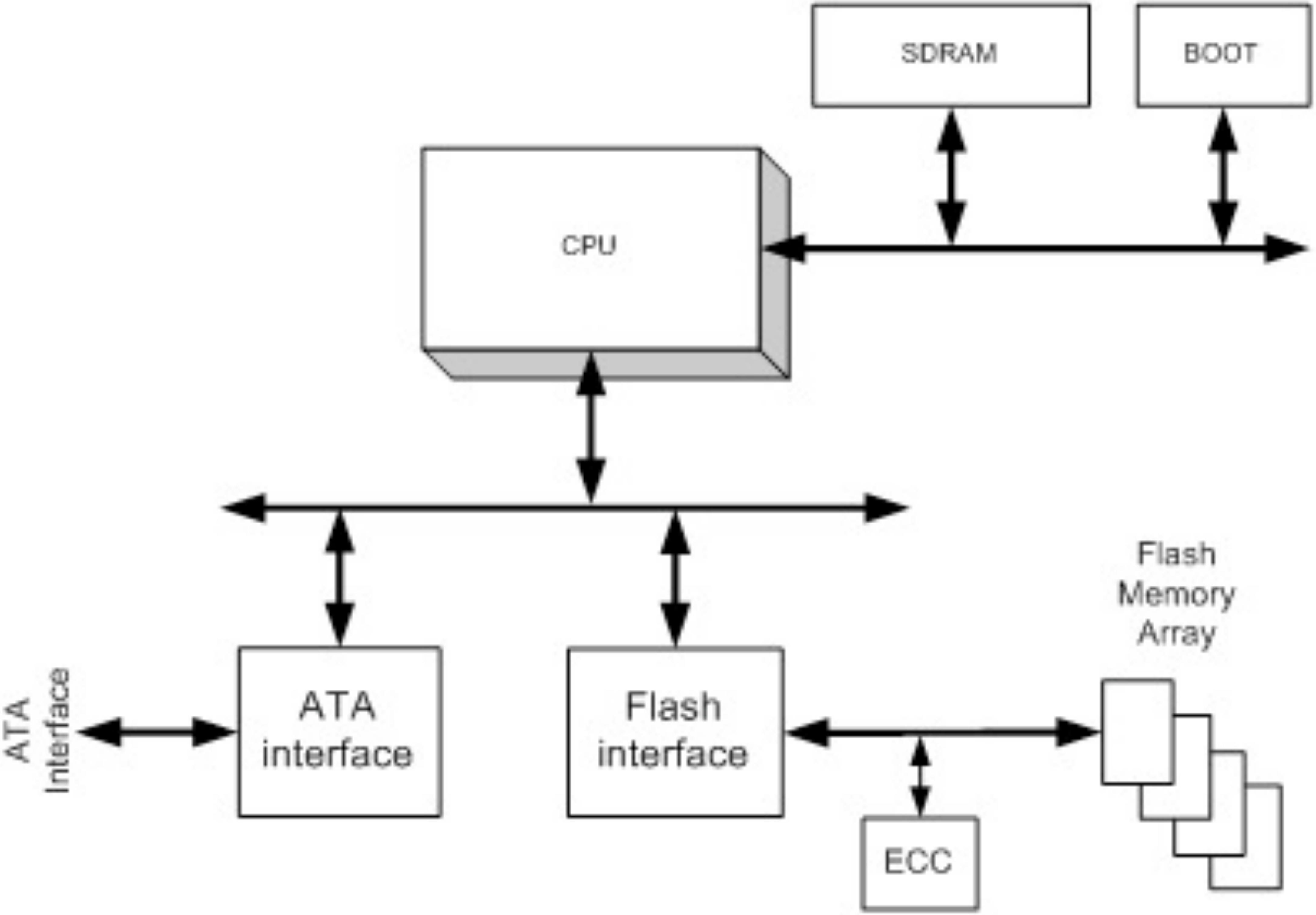
- Asymmetric read [1x] & program times [10x] (plus erase time [100x])

- Proprietary firmware (heavily IP-oriented, not public, little published)

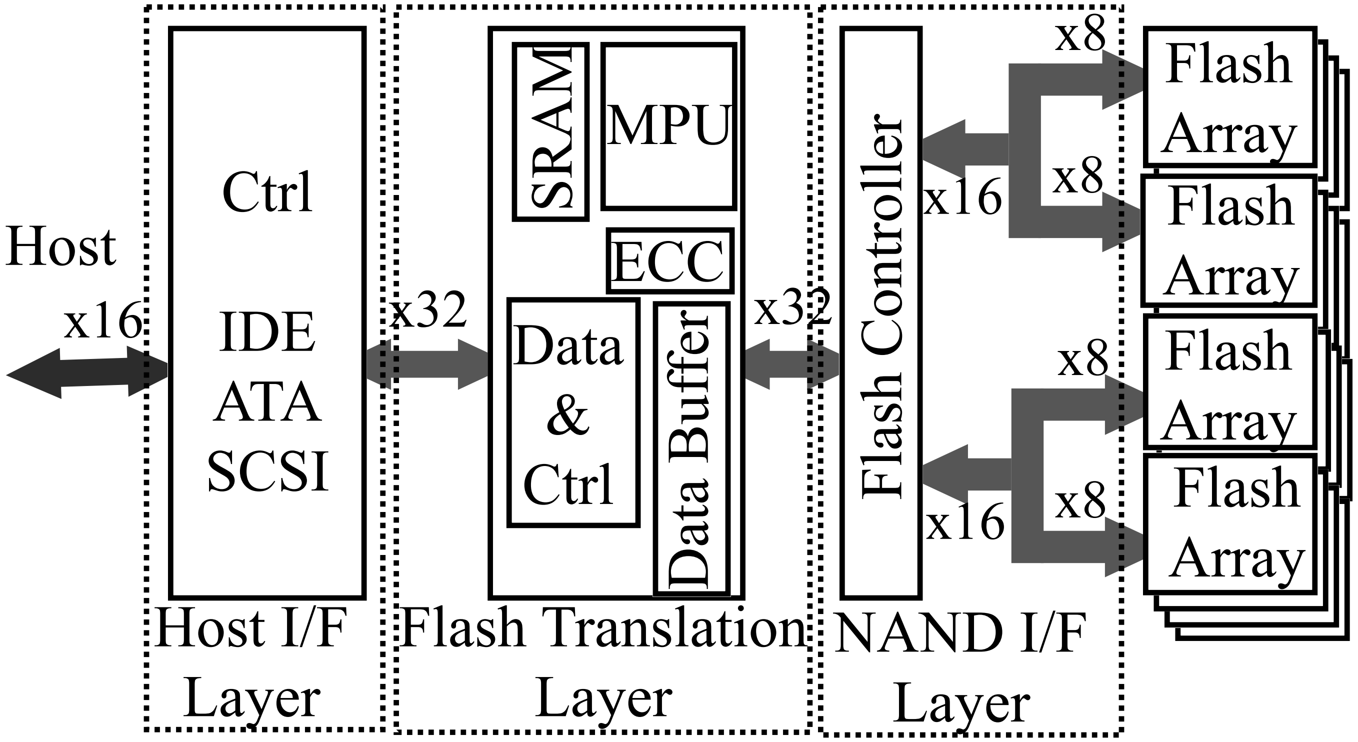
  - Lack of models: timing/performance & power, notably
    Flash Translation Layer is a black box (both good & bad)
    Ditto with garbage collection heuristics, wear leveling, ECC, etc.

  - Result: poorly researched (potentially?)
    E.g., heuristics? how to best organize concurrency? etc.

SanDisk SSD Ultra ATA 2.5” Block Diagram

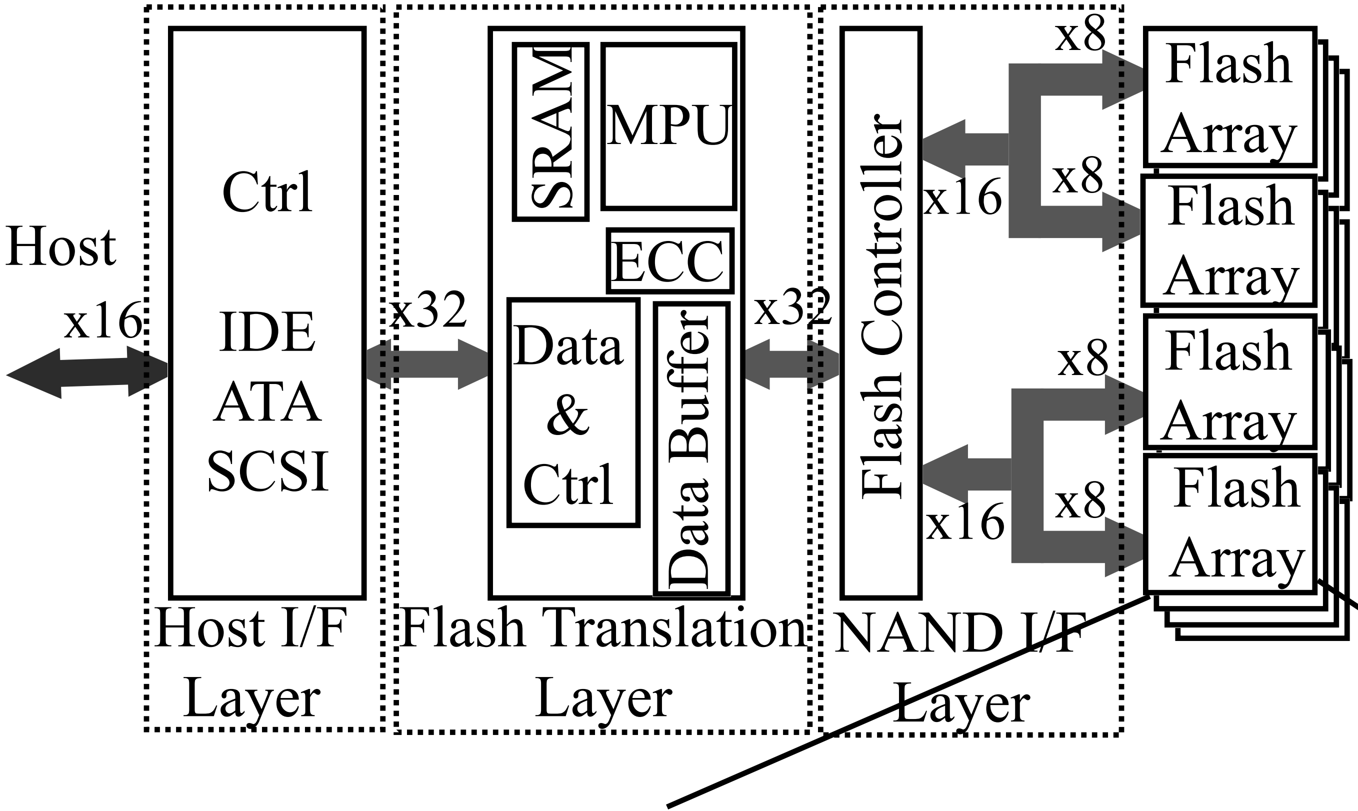


Flash SSD Organization & Operation

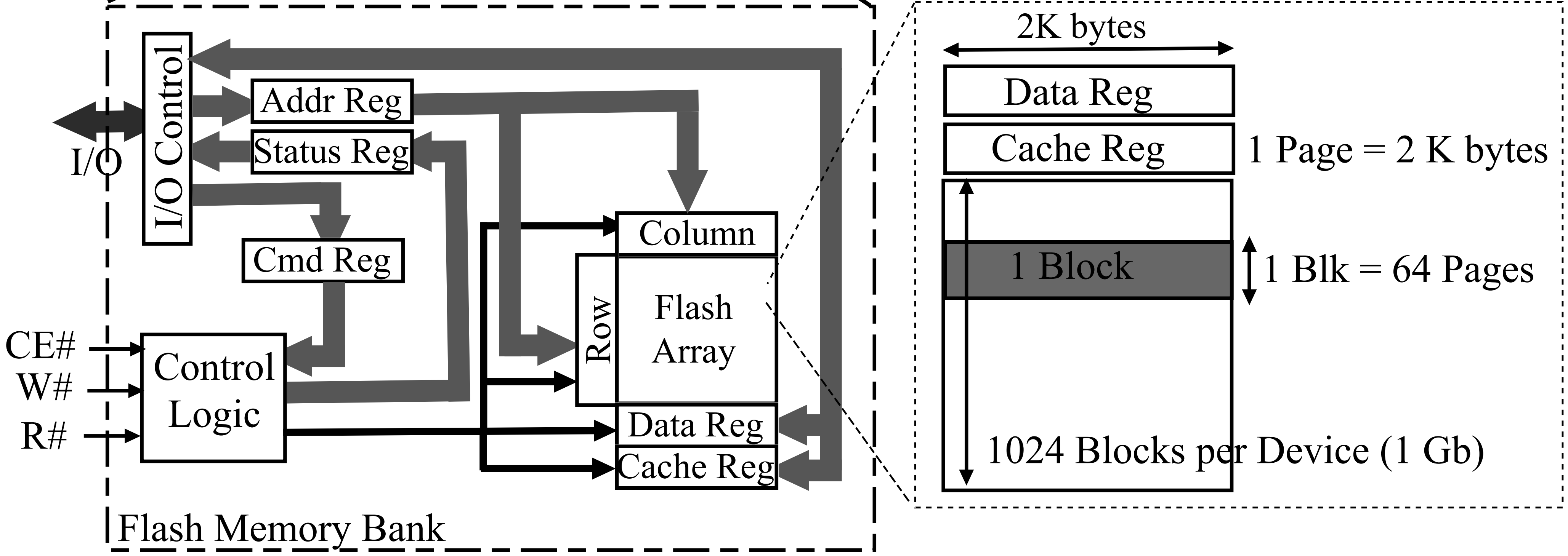


- Numerous Flash arrays
- Arrays controlled externally (controller rel. simple, but can stripe or interleave requests)
- Ganging is device-specific
- FTL manages mapping (VM), ECC, scheduling, wear leveling, data movement
- Host interface emulates HDD

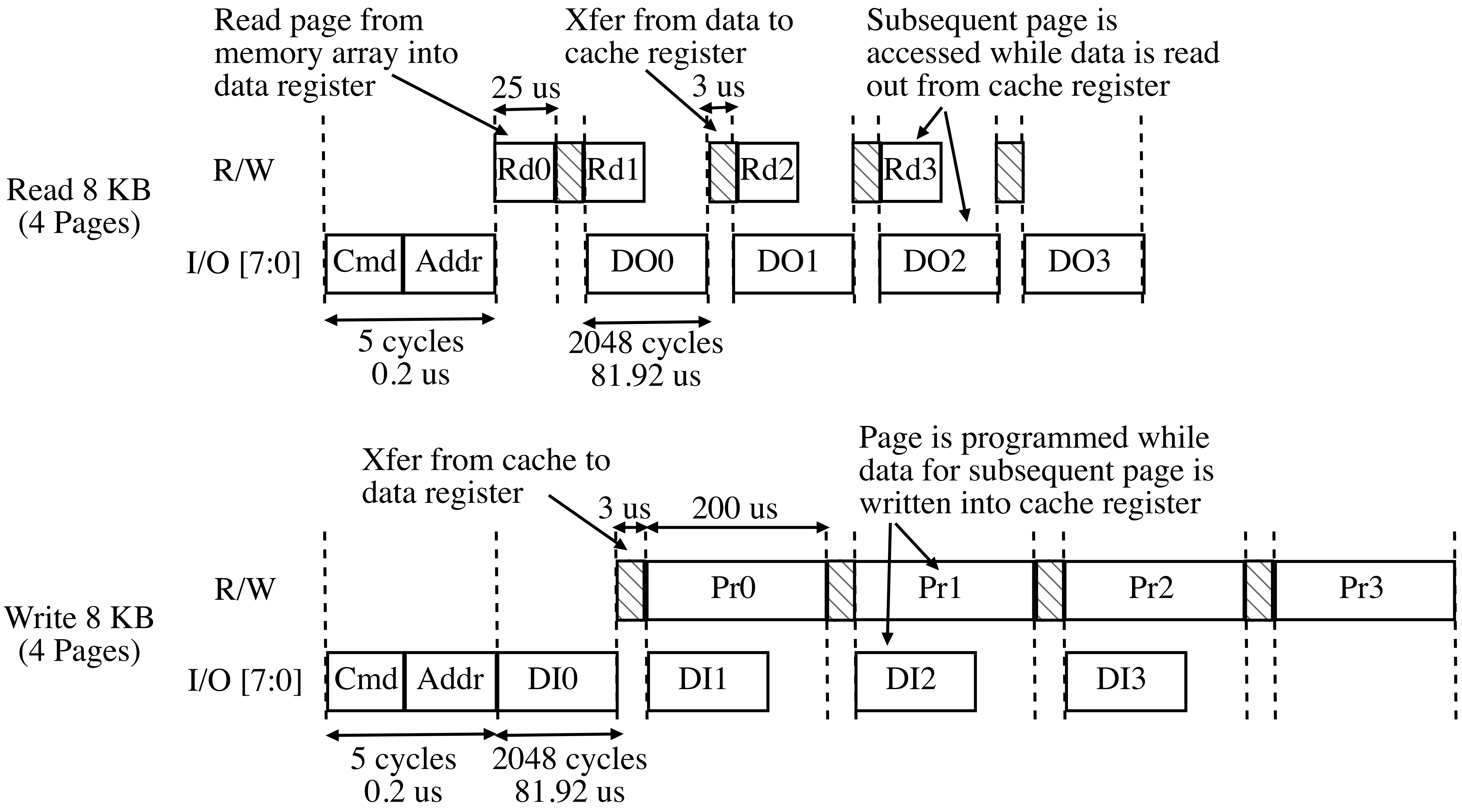
Flash SSD Organization & Operation



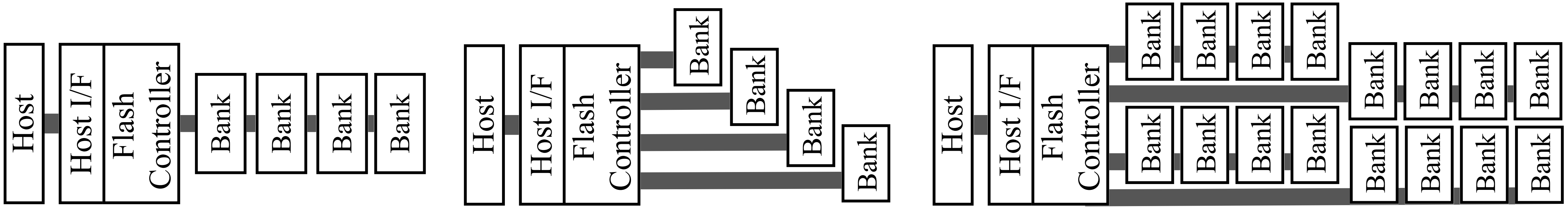
- 2 KB Page
- 128 KB Block
- 2 µs page read
- 200 µs page program
- 3 ms block erase
- 32 GB total storage



Flash SSD Timing


Read 8 KB (4 Pages)
R/W
I/O [7:0]
Read page from memory array into data register
25 us
Xfer from data to cache register
3 us
Subsequent page is accessed while data is read out from cache register
Rd0
Rd1
Rd2
Rd3
Cmd
Addr
DO0
DO1
DO2
DO3
5 cycles
0.2 us
2048 cycles
81.92 us
Write 8 KB (4 Pages)
R/W
I/O [7:0]
Xfer from cache to data register
3 us
200 us
Page is programmed while data for subsequent page is written into cache register
Pr0
Pr1
Pr2
Pr3
Cmd
Addr
DI0
DI1
DI2
DI3
5 cycles
0.2 us
2048 cycles
81.92 us

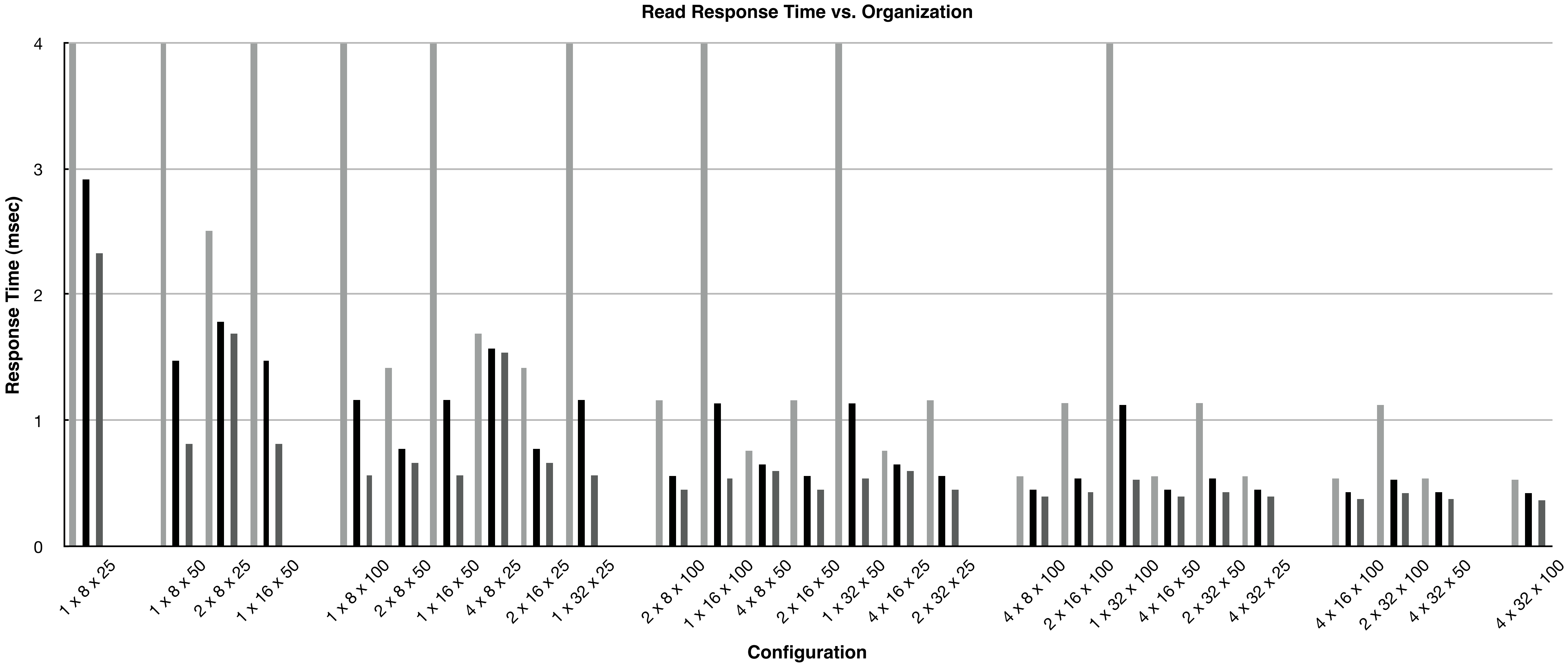
Some Performance Studies



(a) Single channel

(b) Dedicated channel for each bank

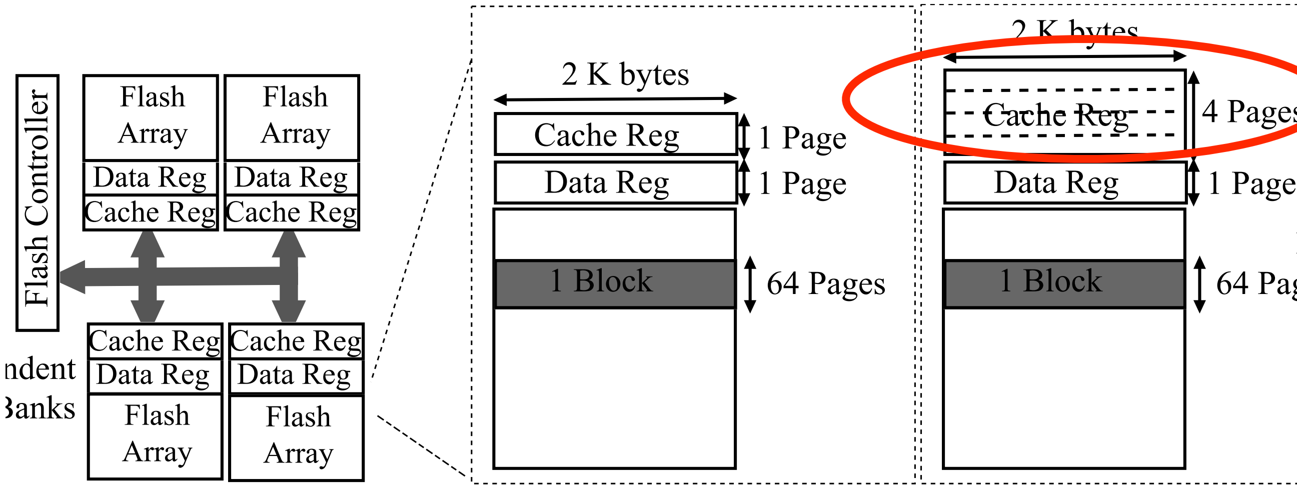
(c) Multiple shared channels

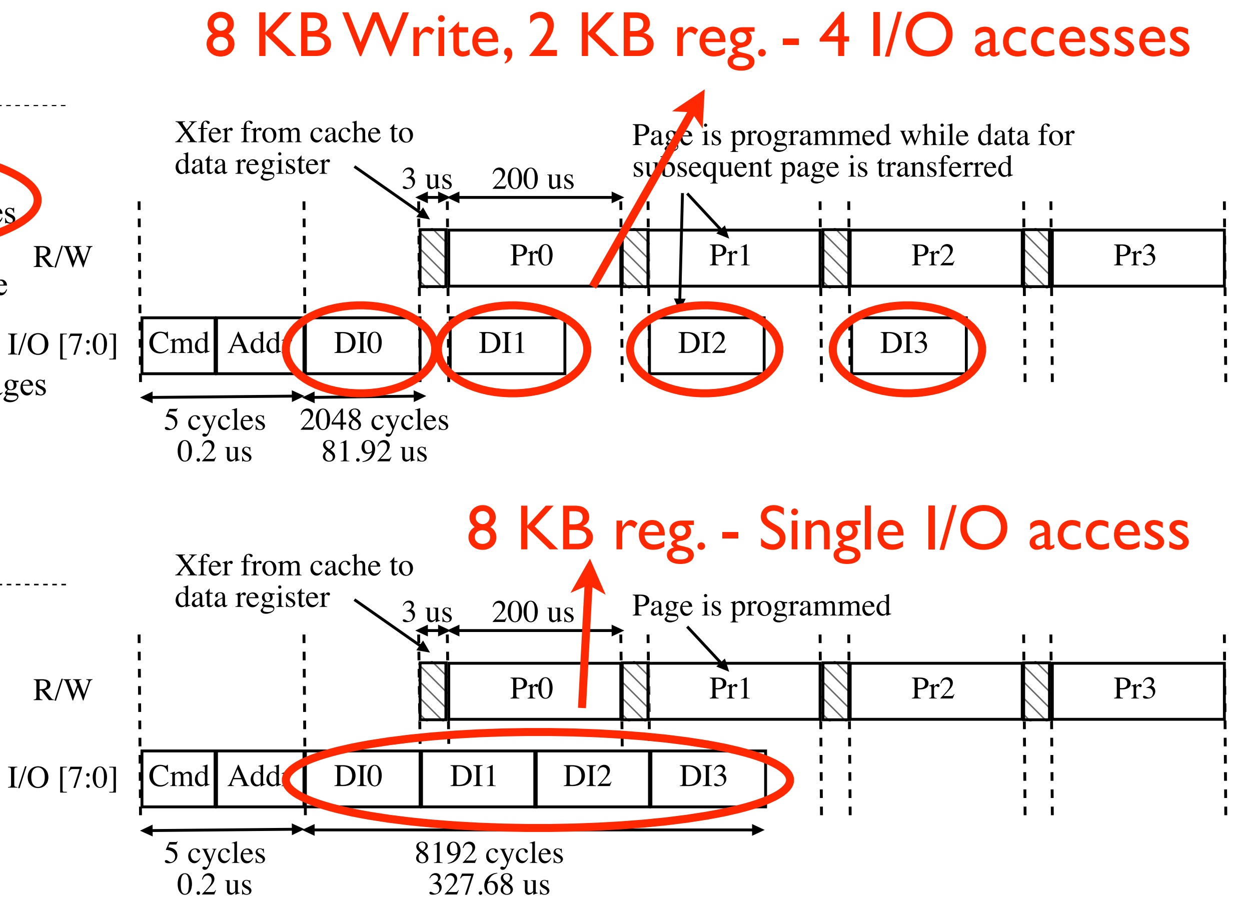


I/O Access Optimization

- Access time increasing with level of banking on single channel

- Increase cache register size




8 KB Write, 2 KB reg. - 4 I/O accesses

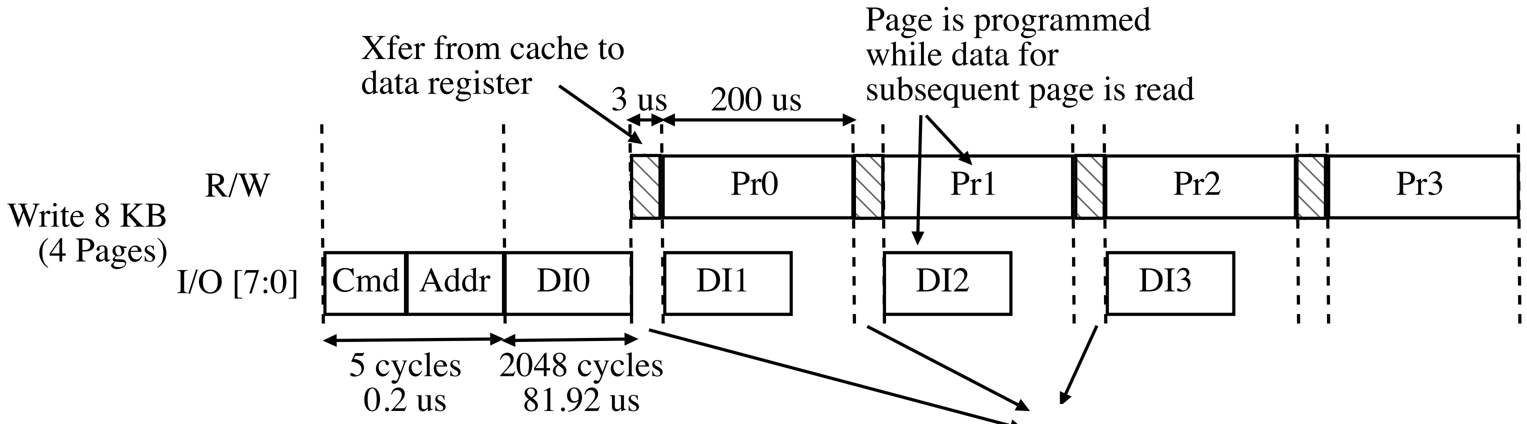
Xfer from cache to data register
3 us
200 us
Page is programmed while data for subsequent page is transferred
R/W
Pr0
Pr1
Pr2
Pr3
I/O [7:0]
Cmd
Add
DI0
DI1
DI2
DI3
5 cycles
0.2 us
2048 cycles
81.92 us

8 KB reg. - Single I/O access

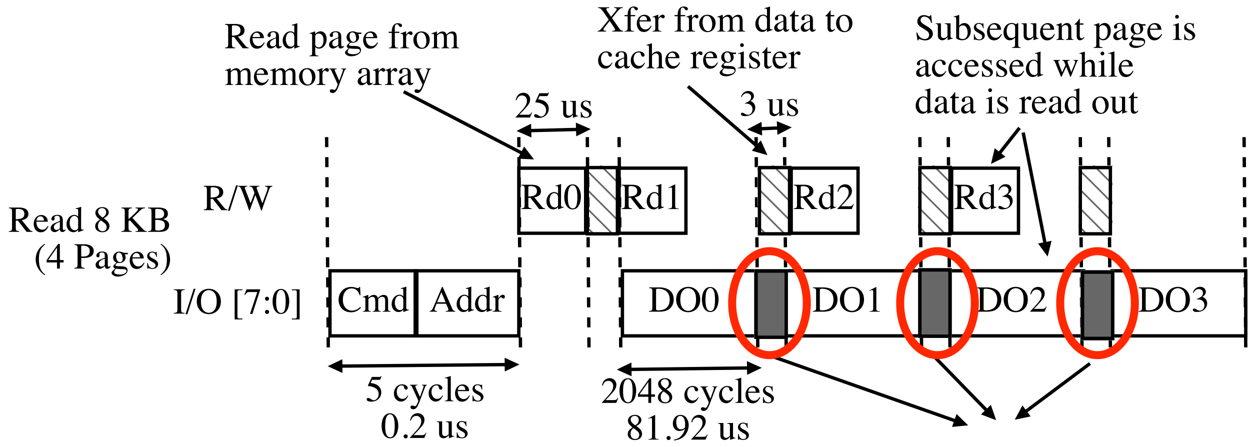
- Reduce # of I/O access requests

I/O Access Optimization

- Implement different bus-access policies for reads and writes



Writes do not need I/O access as frequently as reads


Read page from memory array
25 us
Xfer from data to cache register
3 us
Subsequent page is accessed while data is read out
R/W
Read 8 KB (4 Pages)
I/O [7:0]
Rd0
Rd1
Rd2
Rd3
Cmd
Addr
DO0
DO1
DO2
DO3
5 cycles
0.2 us
2048 cycles
81.92 us

Reads: Hold I/O bus between data bursts