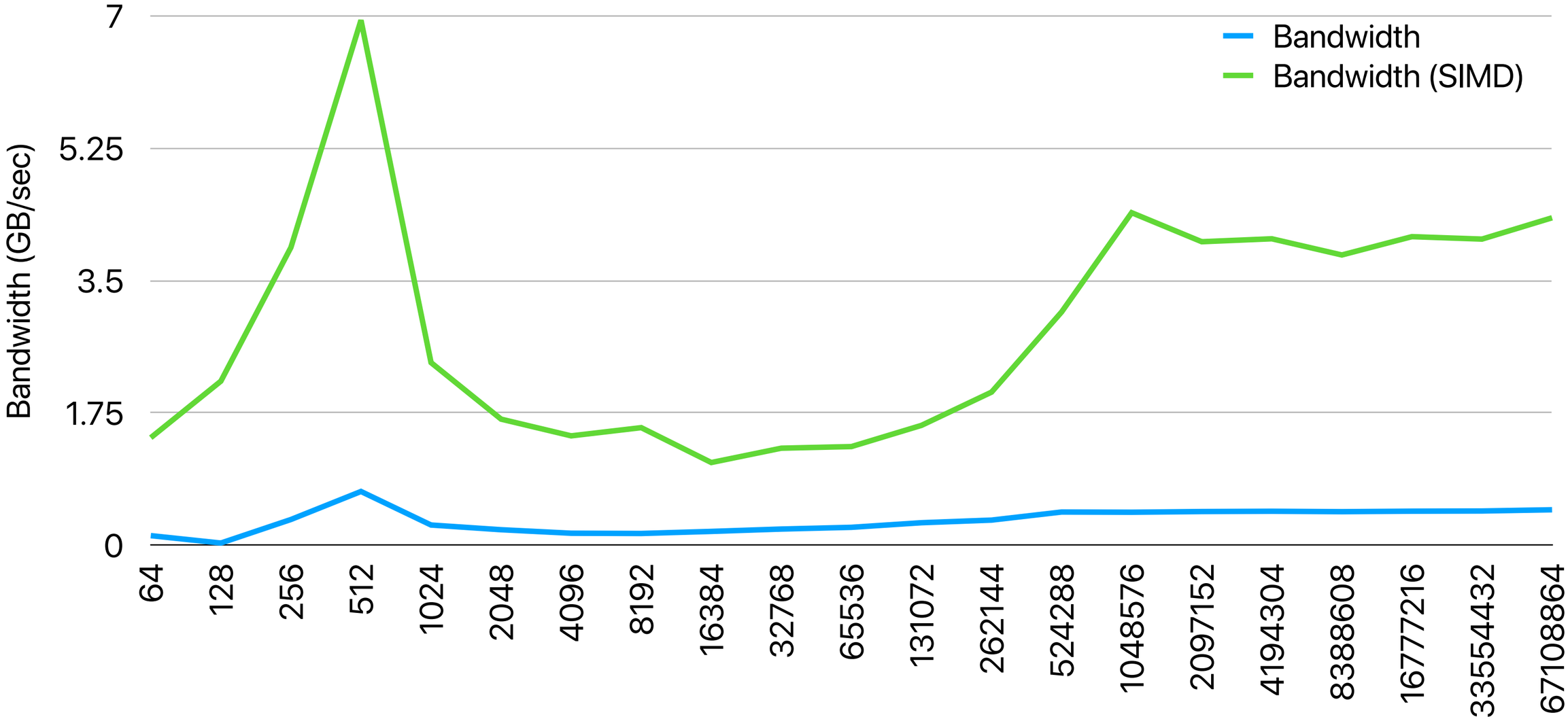
Basics on System Peripherals and Storage Devices

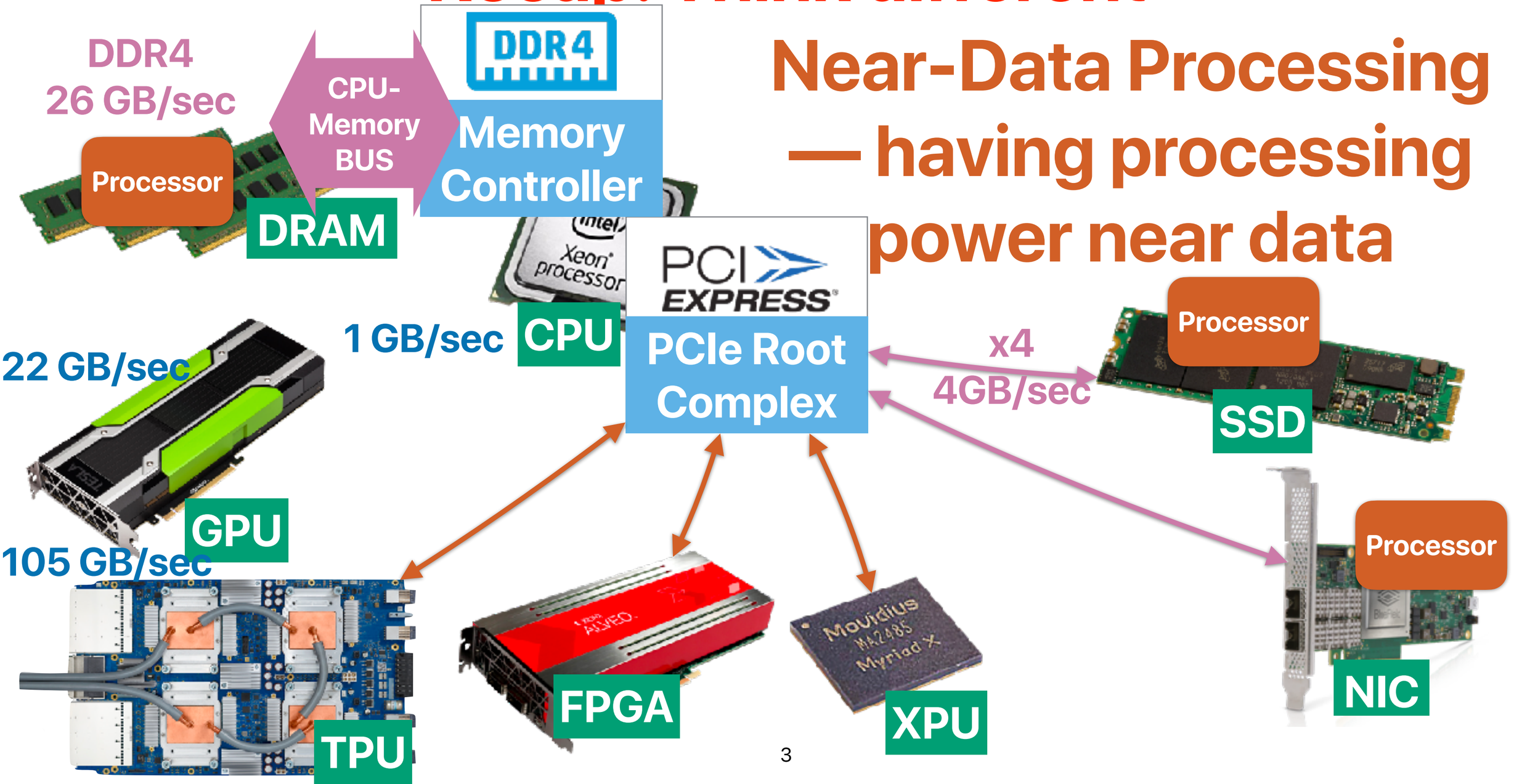
Hung-Wei Tseng

Recap: Memory copy bandwidth using SIMD

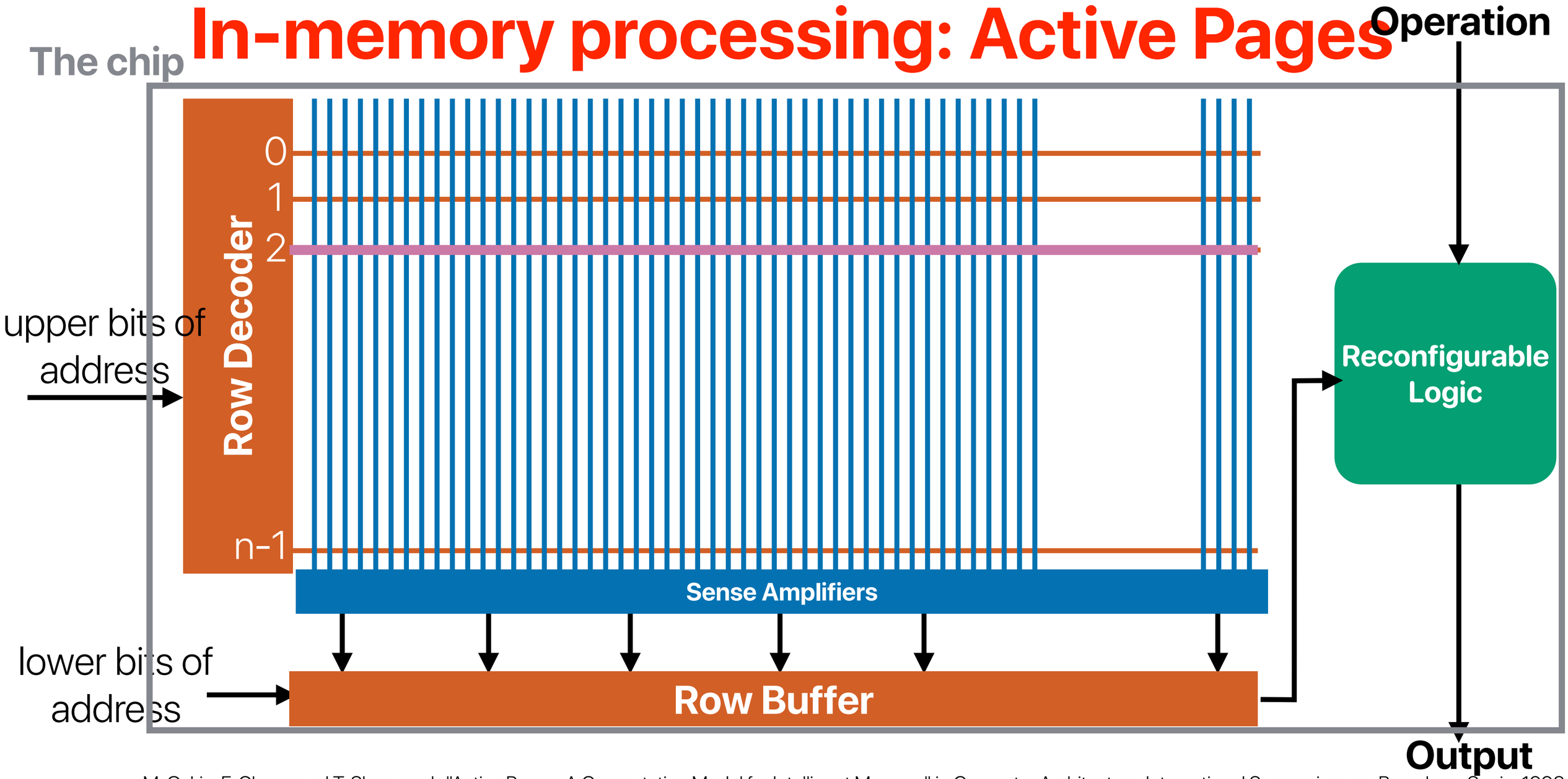


Recap: Think different

Near-Data Processing — having processing power near data

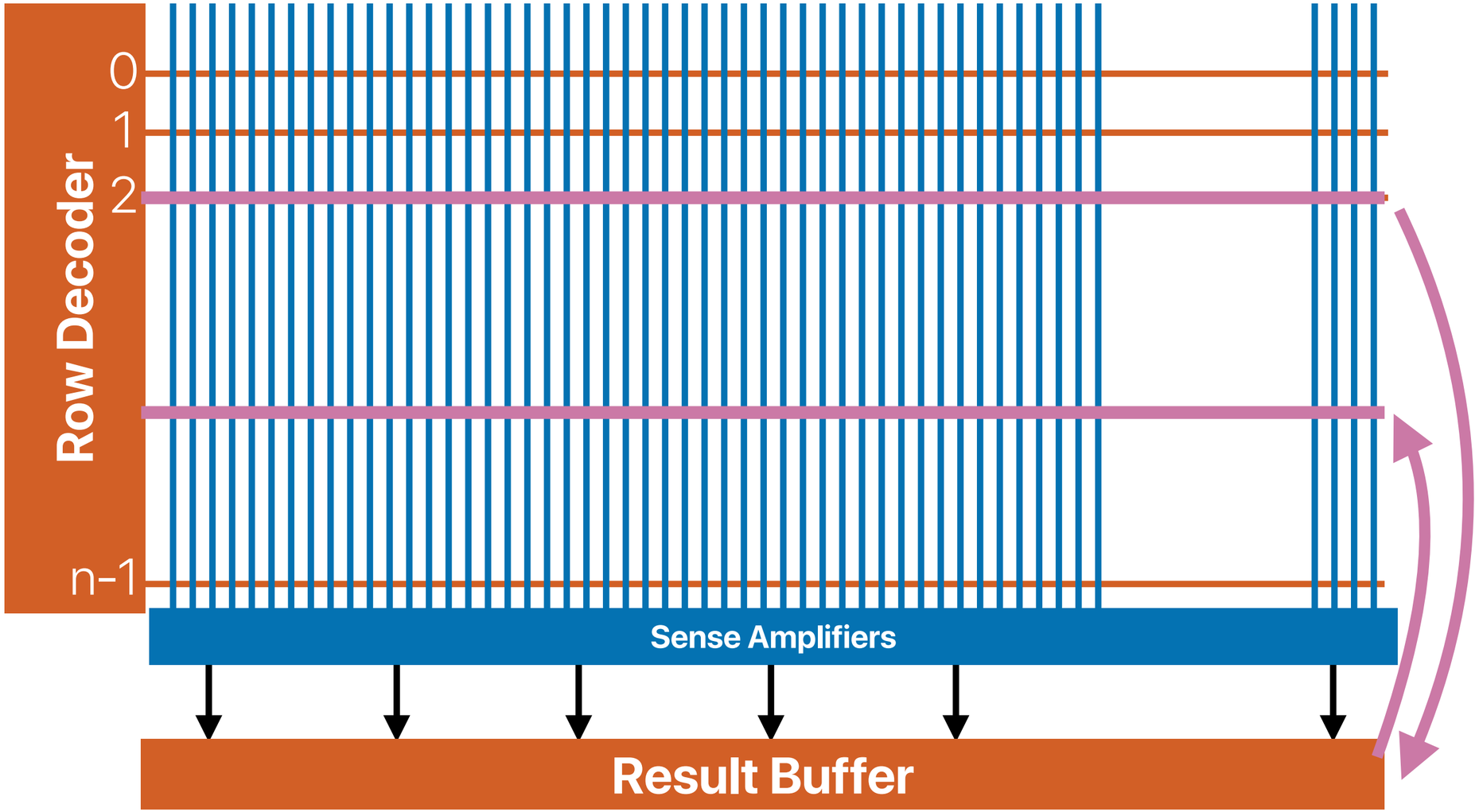


In-memory processing: Active Pages



M. Oskin, F. Chong and T. Sherwood, "Active Pages: A Computation Model for Intelligent Memory," in Computer Architecture, International Symposium on, Barcelona, Spain, 1998

Recap: Or we just clone/move?

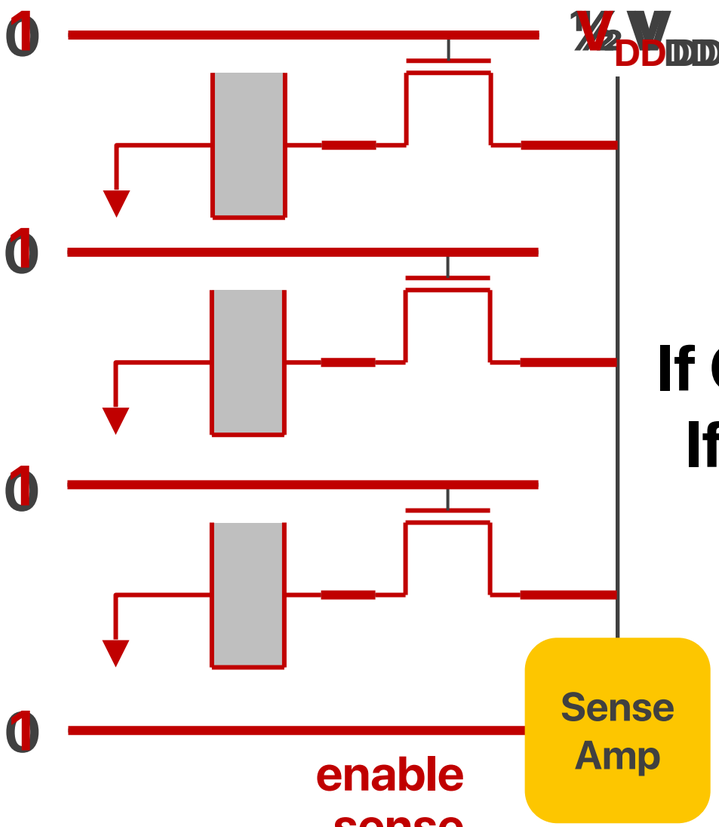


memmove & memcpy: 5% cycles in Google's datacenter

Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks.Profiling a warehouse-scale computer ISCA '15

Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. RowClone: fast and energy-efficient in-DRAM bulk data copy and initialization. In MICRO-46.

Recap: Activate Three Rows



AB + BC + AC =
C(A+B) + C′AB

If C is 0, we can compute AND
If C is 1, we can compute OR

Input			Output
A	B	C	
0	0	0	**0**
0	1	0	**0**
1	0	0	**0**
1	1	0	**1**
0	0	1	**0**
0	1	1	**1**
1	0	1	**1**
1	1	1	**1**

		A′B′	A′B	AB	AB′
	Out(A, B)	0,0	0,1	1,1	1,0
C′	0	0	0	1	0
C	1	0	1	1	1

BC AB AC

Recap: Ambit

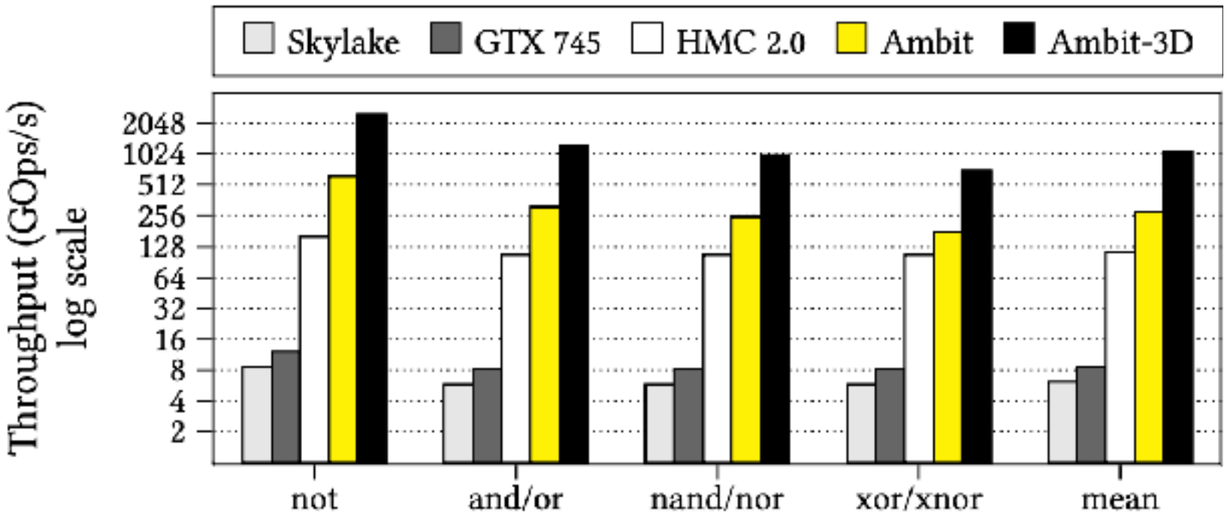


Figure 9: Throughput of bulk bitwise operations.

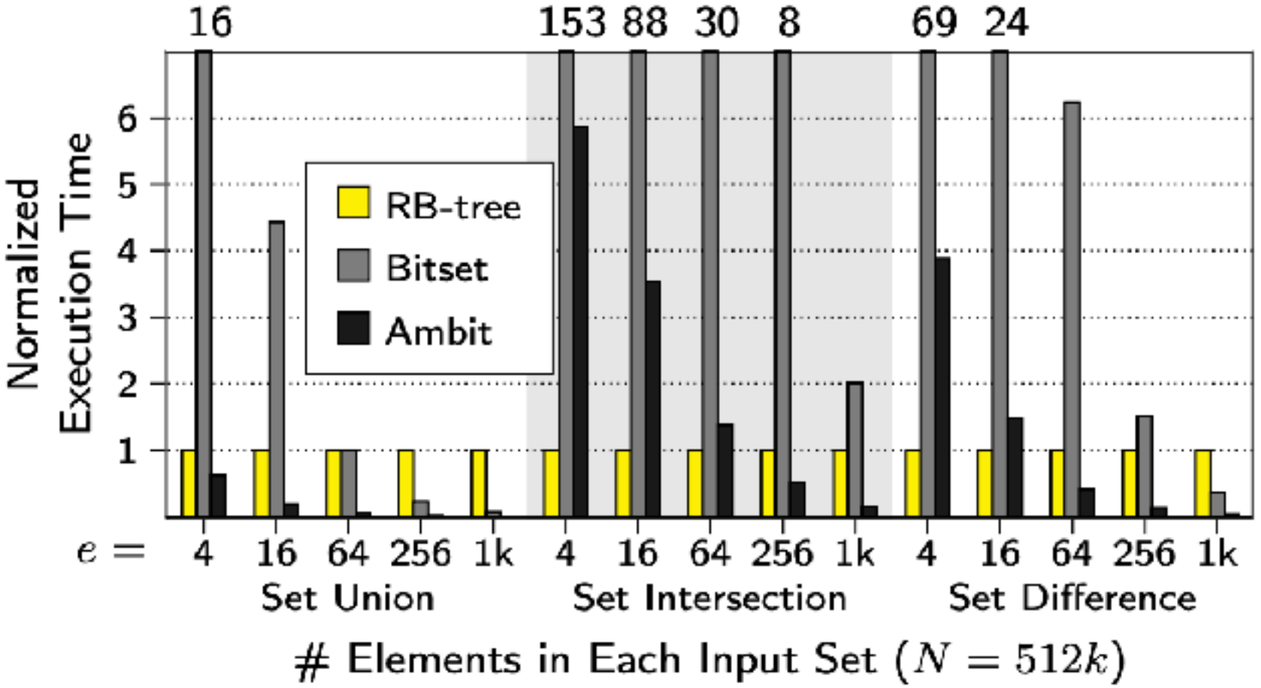
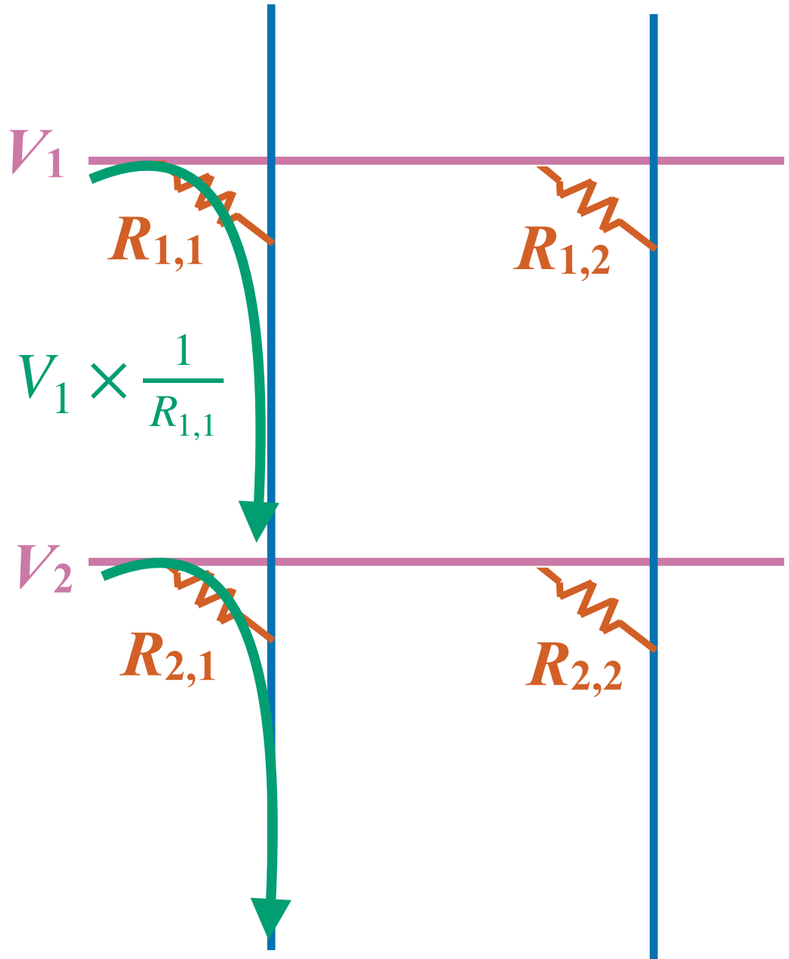


Figure 12: Performance of set operations

Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry. Ambit: in-memory accelerator for bulk bitwise operations using commodity DRAM technology. MICRO-50.

Recap: ReRAM



$$I_{1,1} = V_1 \times \frac{1}{R_{1,1}}$$

$$I_1 = V_1 \times \frac{1}{R_{1,1}} + V_2 \times \frac{1}{R_{2,1}} \qquad I_2 = V_1 \times \frac{1}{R_{2,1}} + V_2 \times \frac{1}{R_{2,2}}$$

V_1
V_2
V_3
V_4

$R_{1,1}^{-1}$	$R_{1,2}^{-1}$	$R_{1,3}^{-1}$	$R_{1,4}^{-1}$
$R_{2,1}^{-1}$	$R_{2,2}^{-1}$	$R_{2,3}^{-1}$	$R_{2,4}^{-1}$
$R_{3,1}^{-1}$	$R_{3,2}^{-1}$	$R_{3,3}^{-1}$	$R_{3,4}^{-1}$
$R_{4,1}^{-1}$	$R_{4,2}^{-1}$	$R_{4,3}^{-1}$	$R_{4,4}^{-1}$

Performance of ReRAM

TABLE III. EFFECTIVE READ LATENCY FOR DIFFERENT R_{sense}.

R_{sense}	T_{sense}	P(retry)	Effective T_{rd} (incl T_{PRE}, T_{BUS})
10KΩ	69 ns	0.0%	80 ns
8KΩ	55 ns	0.0%	66 ns
7.5KΩ	51 ns	0.0%	62 ns
7KΩ	**48 ns**	**0.5%**	**60 ns**
6.5KΩ	44 ns	40%	90 ns

If each ReRAM's array dimension is 4K — the total OPs the ReRAM can perform per cycle is $4096 \times 4096 \times 2 = 8380416$

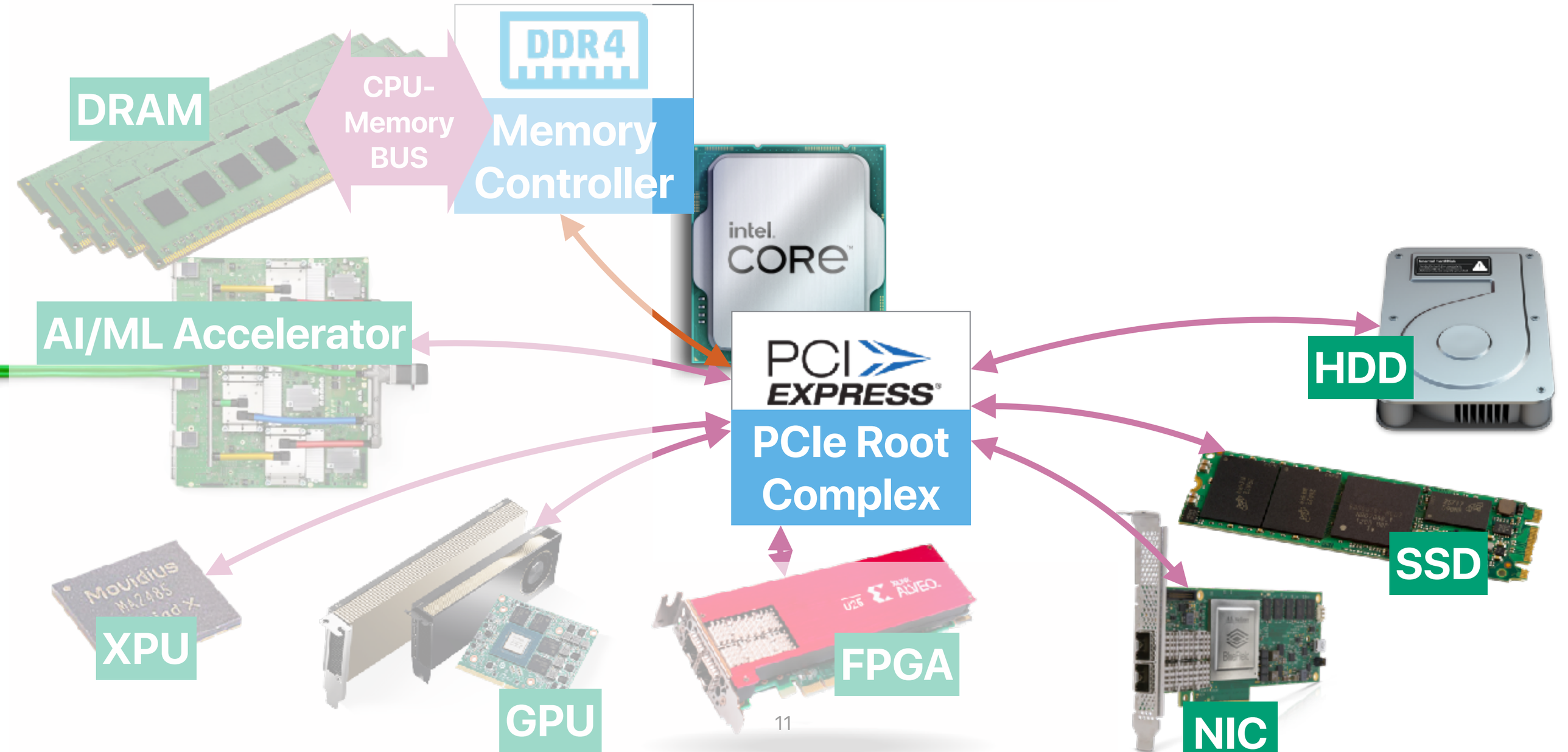
The throughput is $\frac{8380416}{60ns} = 139TOPS$

P. J. Nair, C. Chou, B. Rajendran and M. K. Qureshi. Reducing read latency of phase change memory via early read and Turbo Read. 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)

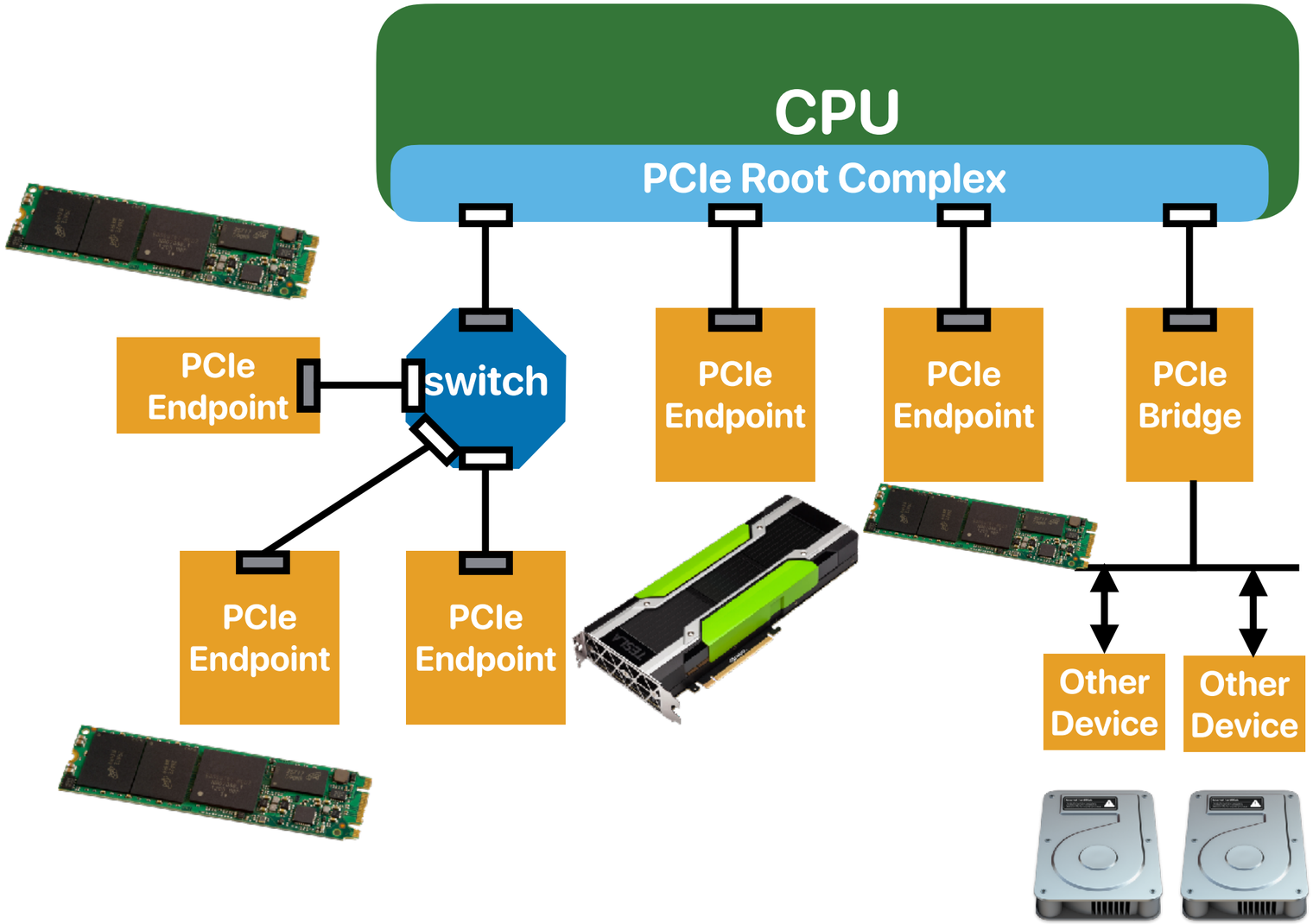
Outline

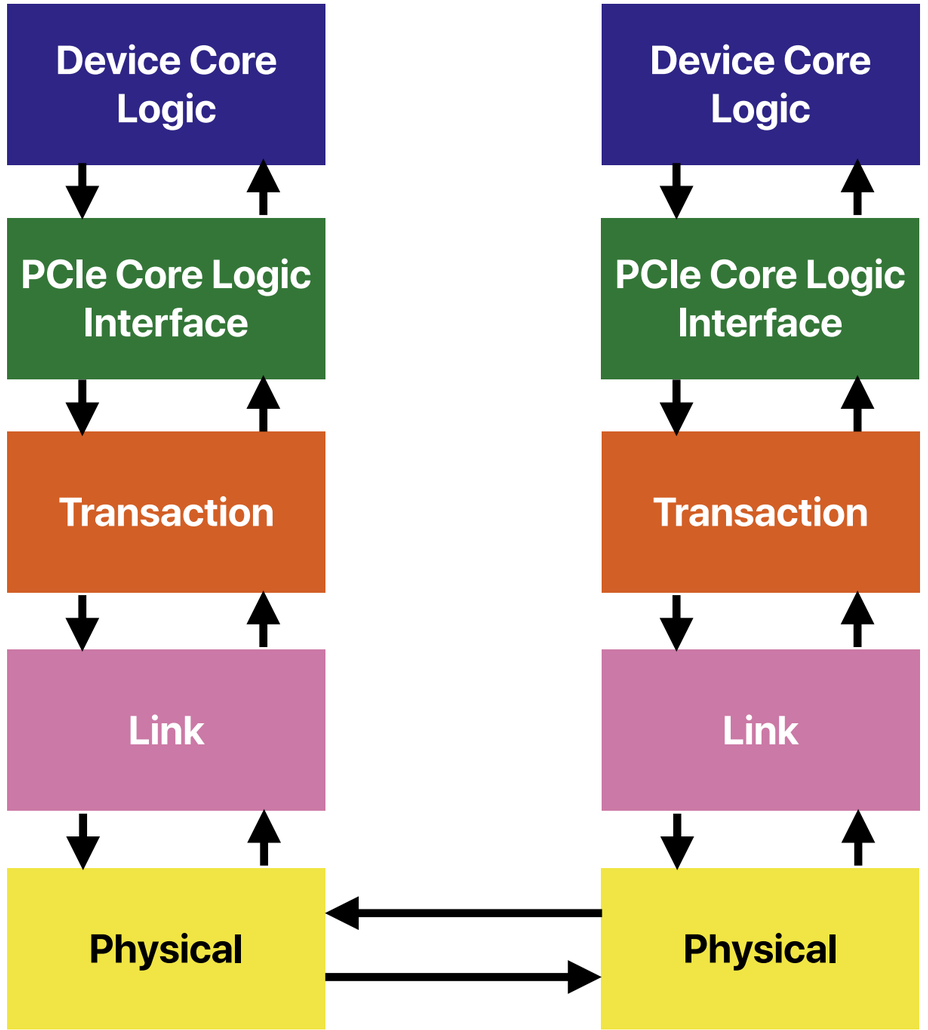
- How to interact with an I/O device
- Storage devices

Recap: The landscape of modern computers



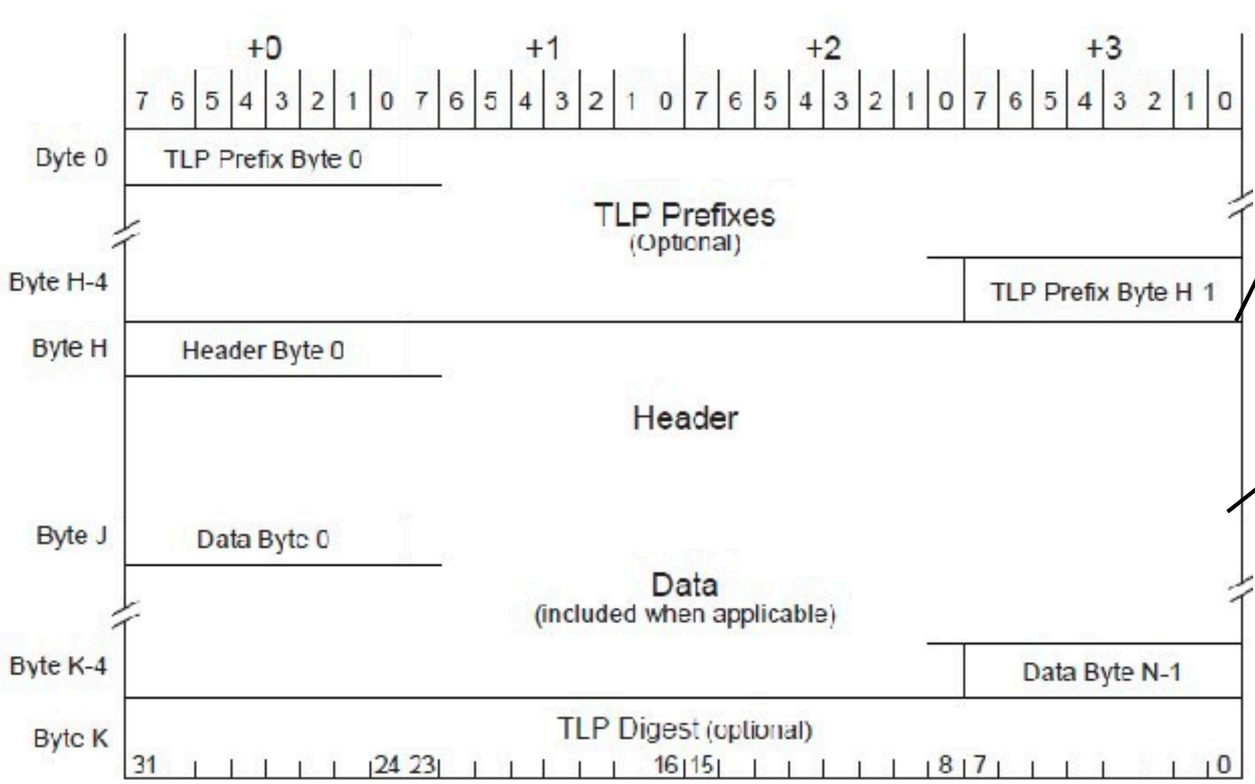
PCIe "Interconnect"

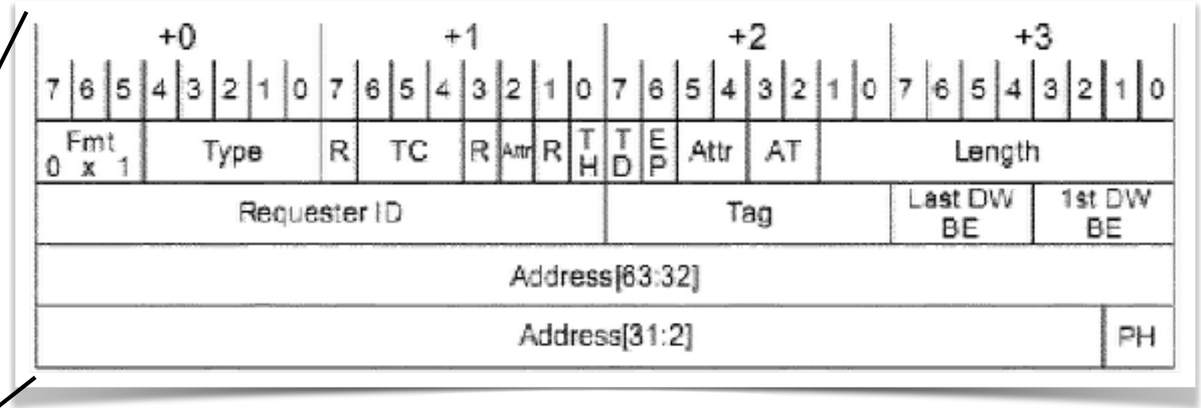




PCIe interconnect

- Very similar to computer networks
- Use “memory addresses” as the identifier for routing



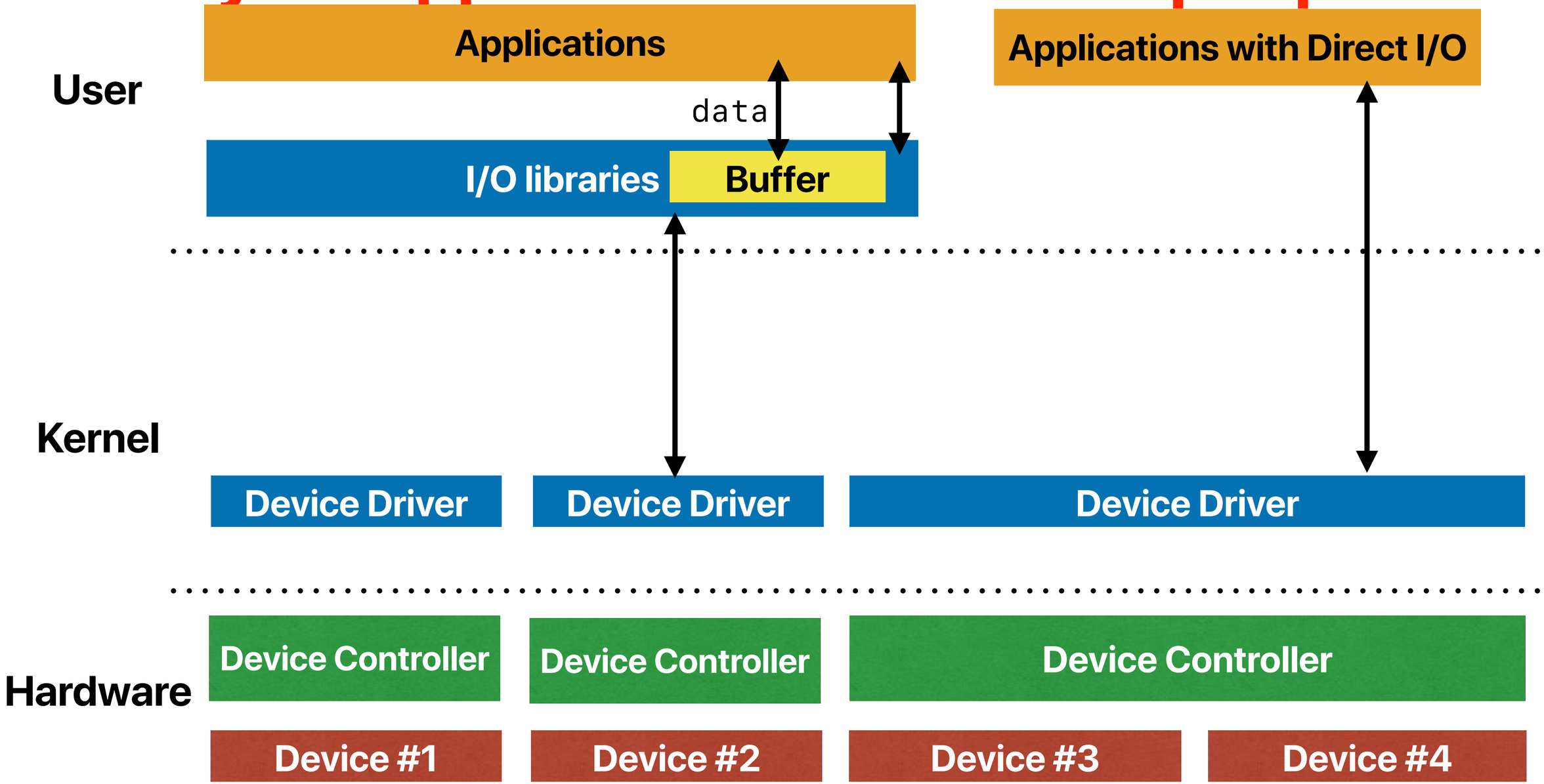


3:00

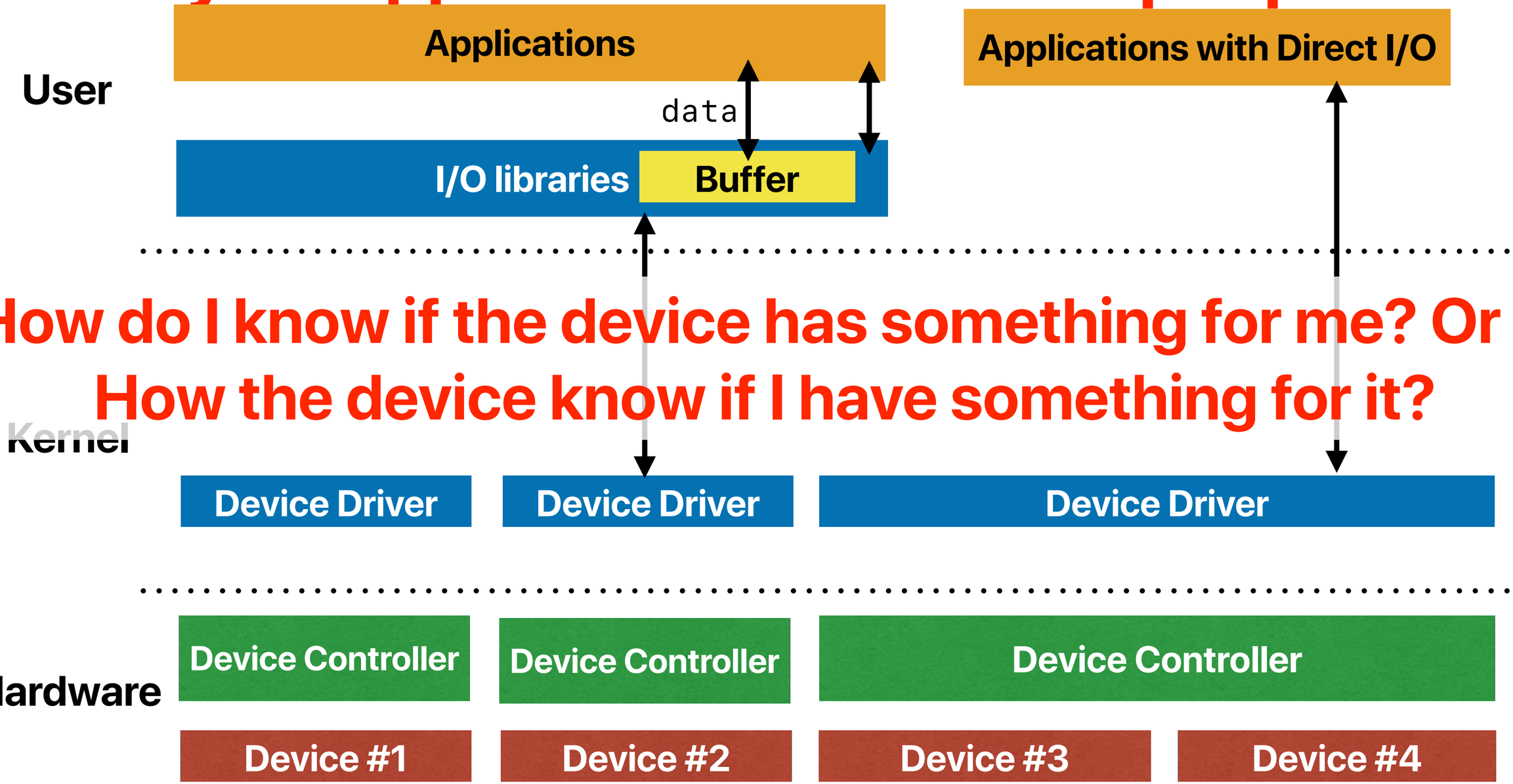
How does your program talk to SSDs, HDDs & NICs?

How does your program talk to devices?

How your application interact with peripherals



How your application interact with peripherals

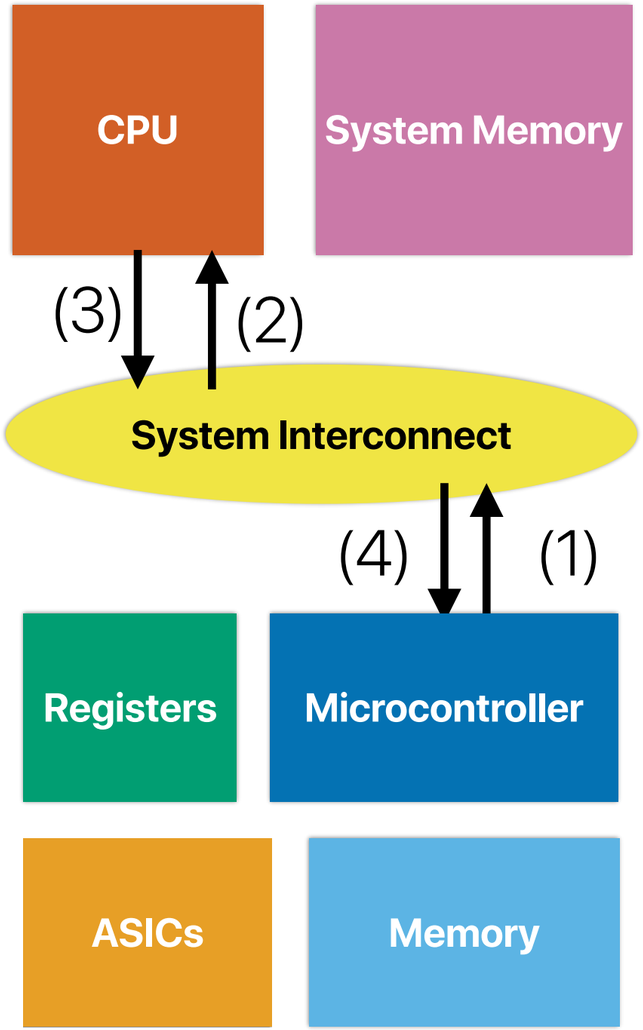


How your CPU/OS interact with I/O devices

- The mechanism of knowing if there is an event
  - Interrupt
  - Polling
- The mechanism of handling the request
  - PIO
  - DMA

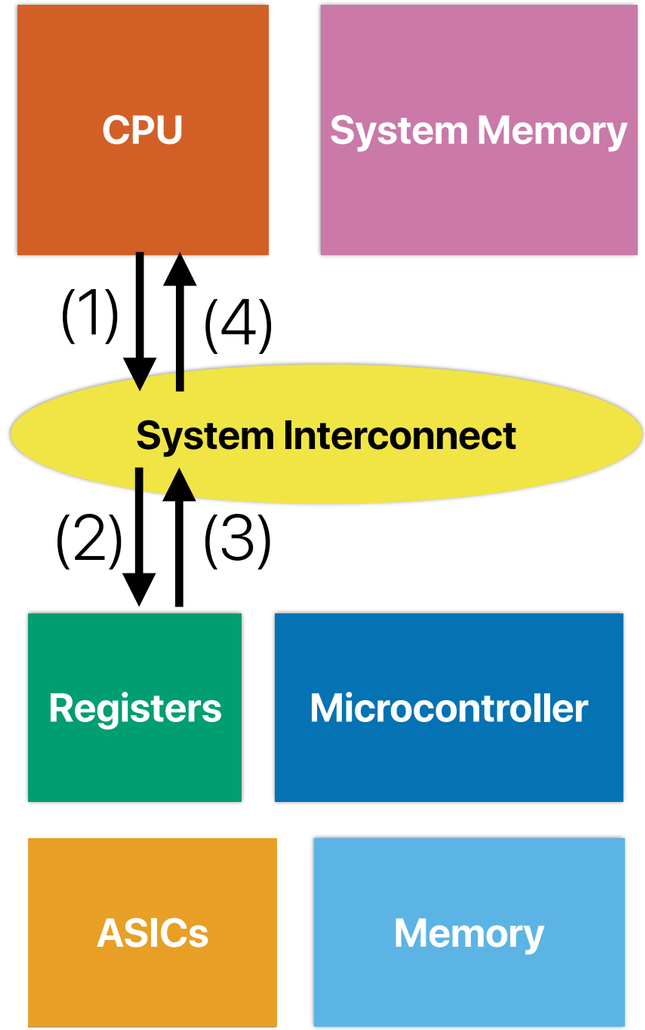
Interrupt

- The device signals the processor only when the device requires the processor/OS handle some tasks/data
- The processor only signals the device when necessary



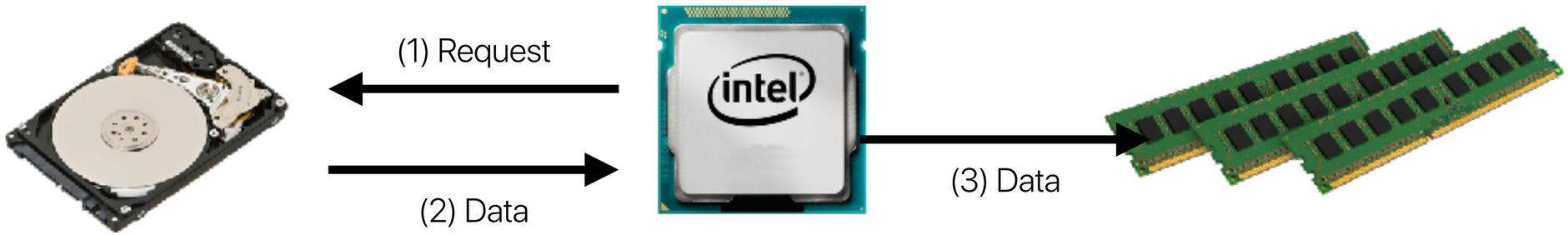
Polling

- The processor/OS constantly asks if the device (e.g. examine the status register of the device) is ready to or requires the processor/OS handle some tasks/data
- The OS/processor executes corresponding handler if the device can handle demand tasks/data or has tasks/data ready



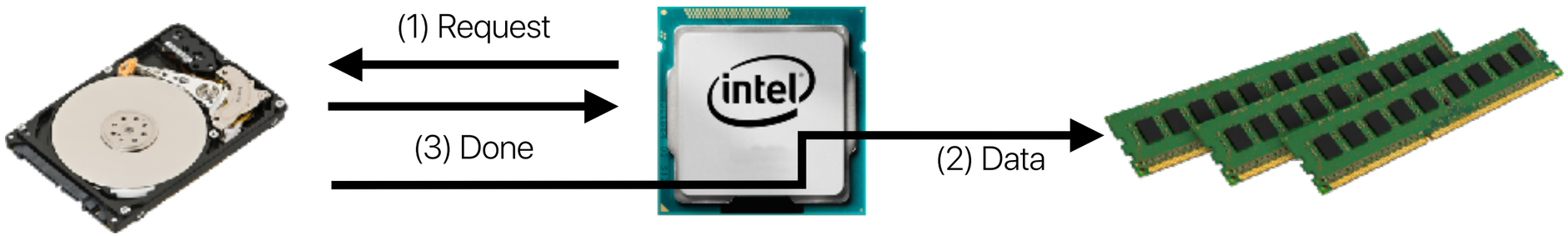
Programmed I/O (PIO)

- The processor/OS executes code and issues commands to handle the data movements between the peripheral and the main memory
  - For example, the in/out instruction in x86



Direct Memory Access (DMA)

- The OS allocates a buffer to exchange data with the peripheral
- The peripheral device can directly access the allocated buffer



3:00

Which model is the best?

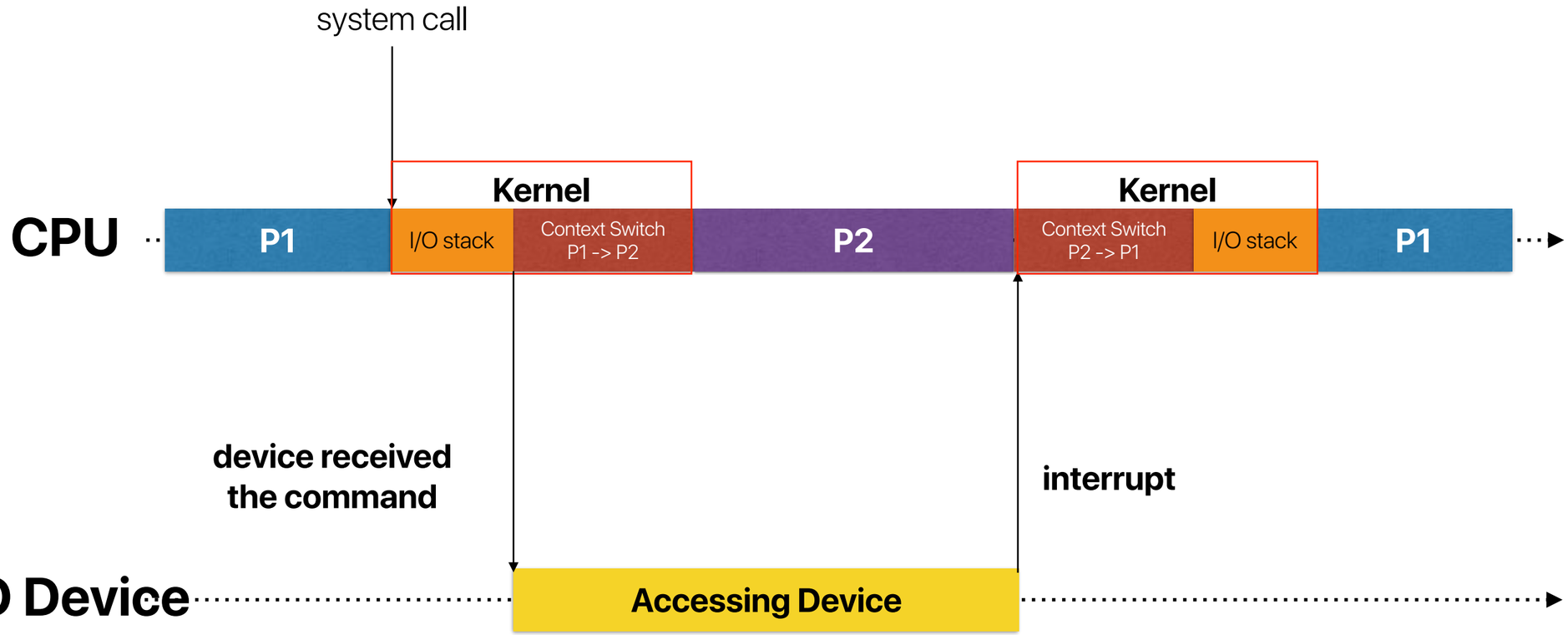
Polling/Interrupt/DMA/PIO?

	Polling	Interrupt
DMA		
PIO		

Recap: What happens during context switch

- Load architectural states from process control block (somewhere in the main memory, potentially a cache miss, TLB miss) — takes several microseconds if everything is in the physical memory
- Set processor registers according to the loaded architectural states
  - **Set the CR3 (page table base register in x86) register to identify the root page table node in the hierarchical page table**
  - Set the RIP (program counter in x86) to the previous execution
- **Restore virtual memory address**
  - **You must load the root page table node to the main memory at least.**
  - **TLB flush**
    - Invalidate all entries in the TLB
    - Most TLBs are not tagged, so you've to do this
- You DO NOT have to load every page content back from disk — remember that we have demand paging!

To switch or not to switch that's the question.



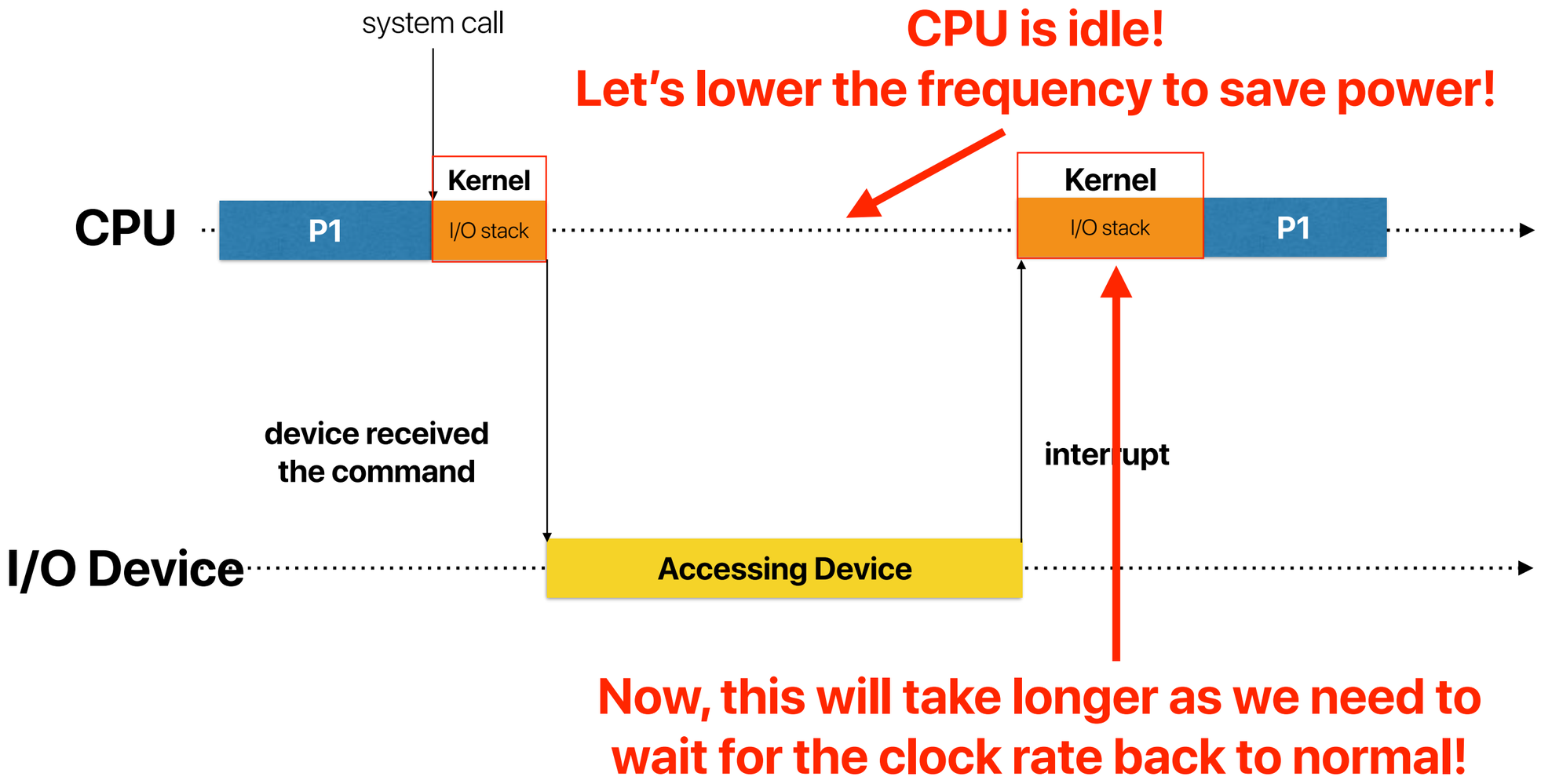
If $T_{\text{Context switch P1->P2}} + T_{\text{Context switch P2->P1}} < T_{\text{Accessing peripherals}}$

makes sense to context switch

But context switch overhead is not the only thing

- Cache warm up cost when you switch back
- TLB warm up cost

What if we don't switch?



Polling/Interrupt/DMA/PIO?

	Polling	Interrupt
DMA	CPU OPs more than IRQ Shorter latency High throughput	Lowest CPU Operations Longer latency High throughput
PIO	Most CPU operations Shorter latency Low throughput	CPU OPs more than IRQ Longer latency Low throughput

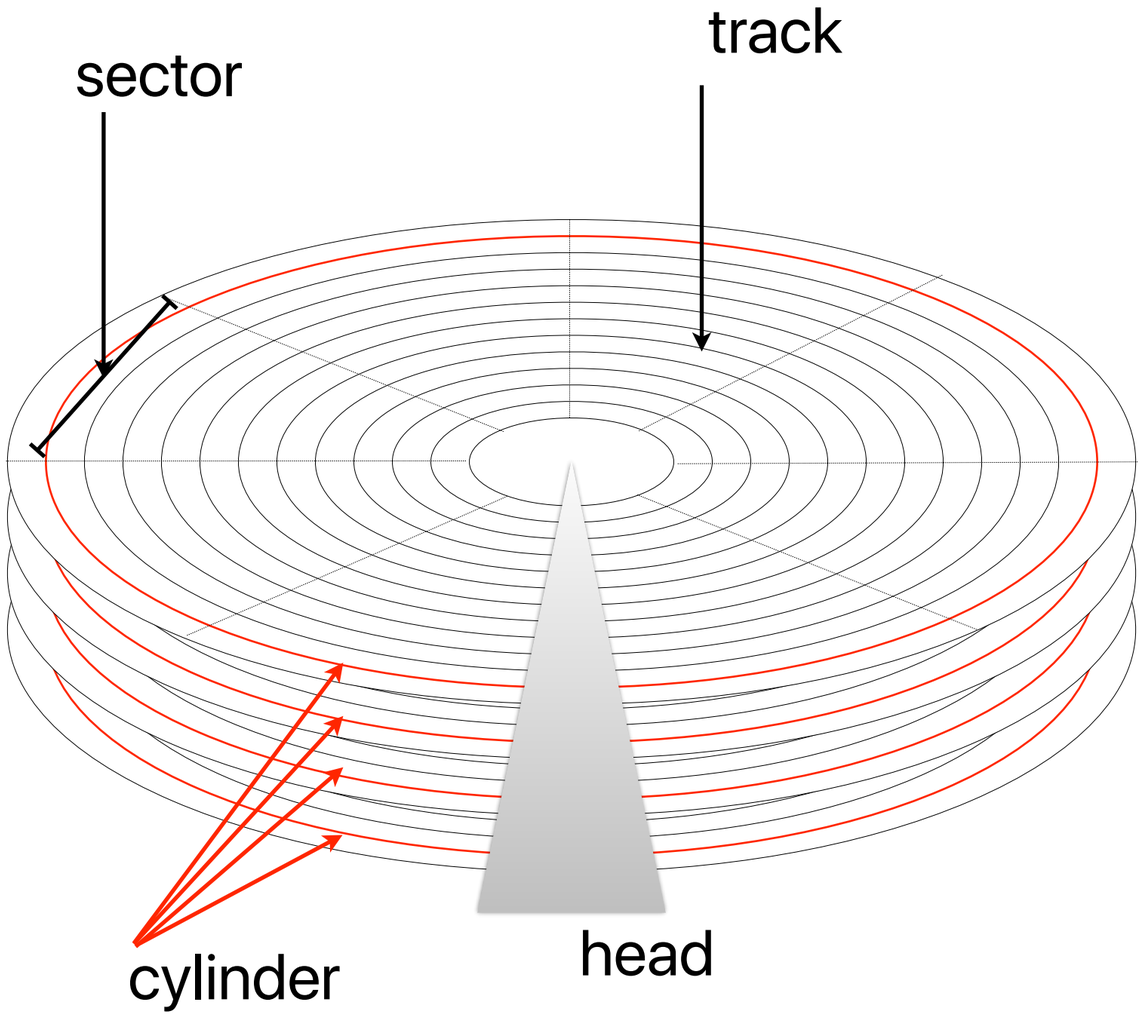
When should we poll? When should we interrupt

- Interrupt is only a good option if the benefit from context switching or energy saving is larger than waiting for the I/O to finish
- In general, applying polling on faster devices
  - DRAM
  - Non-volatile memory (e.g., flash, PCM)

Hard Disk Drives

Hard Disk Drive

Each sector is identified, locate by an "block address"



- Position the head to proper track (seek time)
- Rotate to desired sector. (rotational delay)
- Read or write data from/to disk to in the unit of sectors (e.g. 512B)
- Takes at least 5ms for each access

SATA Standard

		Speed
SATA Express	16 Gbit/s	1.97 GB/s[e]
SATA revision 3.0	6 Gbit/s	600 MB/
SATA revision 2.0	3 Gbit/s	300 MB/s
SATA revision 1.0	1.5 Gbit/s	150 MB/

Latency Numbers Every Programmer Should Know (2020 Version)

Operations	Latency (ns)	Latency (us)	Latency (ms)	
L1 cache reference	0.5 ns			~ 1 CPU cycle
Branch mispredict	3 ns			
L2 cache reference	4 ns			14x L1 cache
Mutex lock/unlock	17 ns			
Send 2K bytes over network	44 ns			
Main memory reference	100 ns			20x L2 cache, 200x L1 cache
Compress 1K bytes with Zippy	2,000 ns	2 us		
Read 1 MB sequentially from memory	3,000 ns	3 us		
Read 4K randomly from SSD*	16,000 ns	16 us		
Read 1 MB sequentially from SSD*	49,000 ns	49 us		
Round trip within same datacenter	500,000 ns	500 us		
Read 1 MB sequentially from disk	825,000 ns	825 us		
Disk seek	2,000,000 ns	2,000 us	2 ms	4x datacenter roundtrip
Send packet CA-Netherlands-CA	150,000,000 ns	150,000 us	150 ms	

https://colin-scott.github.io/personal_website/research/interactive_latency.html

Seagate Barracuda 12

- SATA II (300MB/s in theory), 7200 R.P.M., seek time around 8 ms. Assume the controller overhead is 0.2ms. What's the **latency** and **bandwidth** of accessing a **512B** sector?

Latency = seek time + rotational delay + transfer time + controller overhead

$$8\ ms \quad +\frac{1}{2}\times\frac{1}{\frac{7200}{60}} \quad +\frac{\frac{0.5}{1024}}{300} \quad +0.2\ ms$$

$$= 8\ ms + 4.17\ ms + 0.00167\ us + 0.2\ ms = 12.36\ ms$$

Bandwidth = volume_of_data over period_of_time

$$= \frac{0.5KB}{12.36ms} = 40.45KB/sec$$

Seagate Barracuda 12

- SATA II (300MB/s in theory), 7200 R.P.M., seek time around 8 ms. Assume the controller overhead is 0.2ms. What's the **latency** and **bandwidth** of accessing consecutive **4MB** data?

Latency = seek time + rotational delay + transfer time + controller overhead

$$8\ ms \quad +\frac{1}{2}\times\frac{1}{\frac{7200}{60}} \qquad +\frac{4}{300} \quad +0.2\ ms$$

$$= 8\ ms + 4.17\ ms + 13.33\ ms + 0.2\ ms = 25.69\ ms$$

Bandwidth = volume_of_data over period_of_time

$$= \frac{4MB}{25.69ms} = 155.7\ MB/sec$$

Trading latencies with bandwidth

Electrical
Computer Science
Engineering

277

つづく