

Dr. Christopher S. Simmons (csim@ices.utexas.edu)

September 1, 2015

Architecture

CSE380: Tools and Techniques
of Computational Science

Lecture Outline

- ❖ Parallel Computer Architectures
- ❖ Components of a Cluster
- ❖ Basic Anatomy of a Server / Desktop / Laptop / Cluster-node
 - ❖ Memory Hierarchy
 - ❖ Structure, Size, Speed, Line Size, Associativity
 - ❖ Latencies and Bandwidths
 - ❖ Memory Architecture
 - ❖ Point-to-Point Communications between platforms
- ❖ Node Communication in Clusters (Interconnects)

Top 500

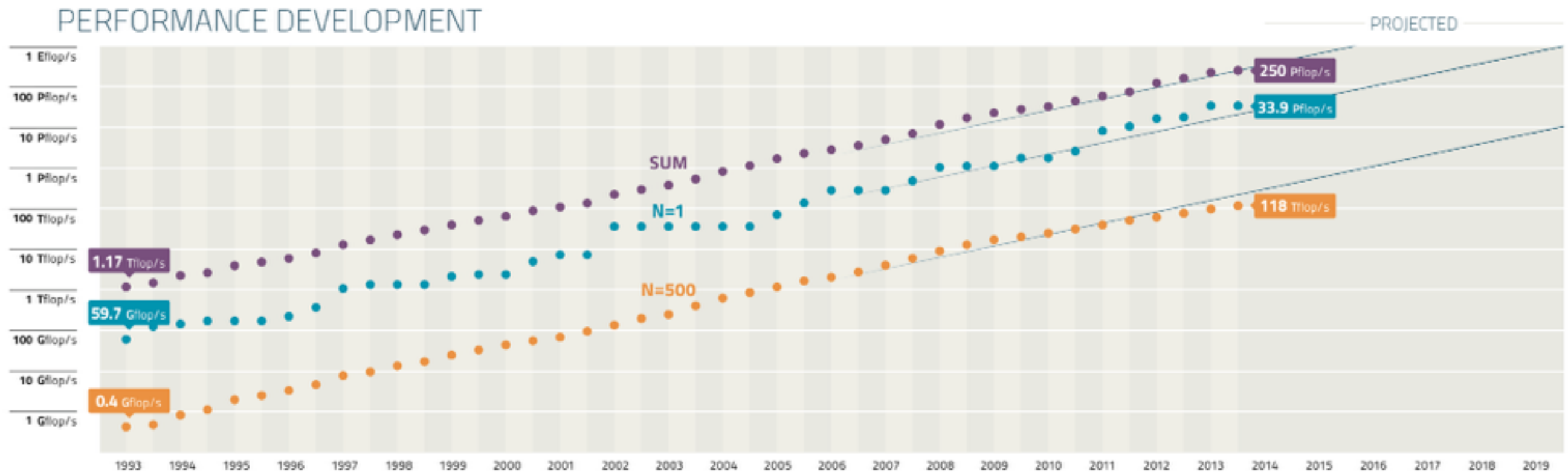
	NAME	SPECS	SITE	COUNTRY	CORES	R _{MAX} PFLOP/S	POWER MW
1	Tianhe-2 (Milkyway-2)	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NSCC Guangzhou	China	3,120,000	33.9	17.8
2	Titan	Cray XK7, Operon 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	17.6	8.2
3	Sequoia	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	17.2	7.9
4	K computer	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	10.5	12.7
5	Mira	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	8.59	3.95

List	Rank	System	Vendors	Cores	Rmax (GFlop/s)	Rpeak (GFlop/s)
11/2013	7	Stampede - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P	Dell	462,462	5,168,110	8,520,111.6
11/2013	153	Lonestar 4 - Dell PowerEdge M610 Cluster, Xeon 5680 3.3Ghz, Infiniband QDR	Dell	22,656	251,800	301,777

Top 500 list

- ❖ High-Performance LINPACK
- ❖ Dense linear system solve with LU factorization
- ❖ # of floating point operations = $\frac{2}{3} n^3 + \text{lower order terms}$
- ❖ Measure: MFlops
- ❖ <http://www.netlib.org/benchmark/hpl/>
- ❖ The problem size can be chosen
 - ❖ fiddle with it until you find n to get the best performance
 - ❖ report n , maximum performance, and theoretical peak performance
- ❖ <http://www.top500.org>

Performance Development



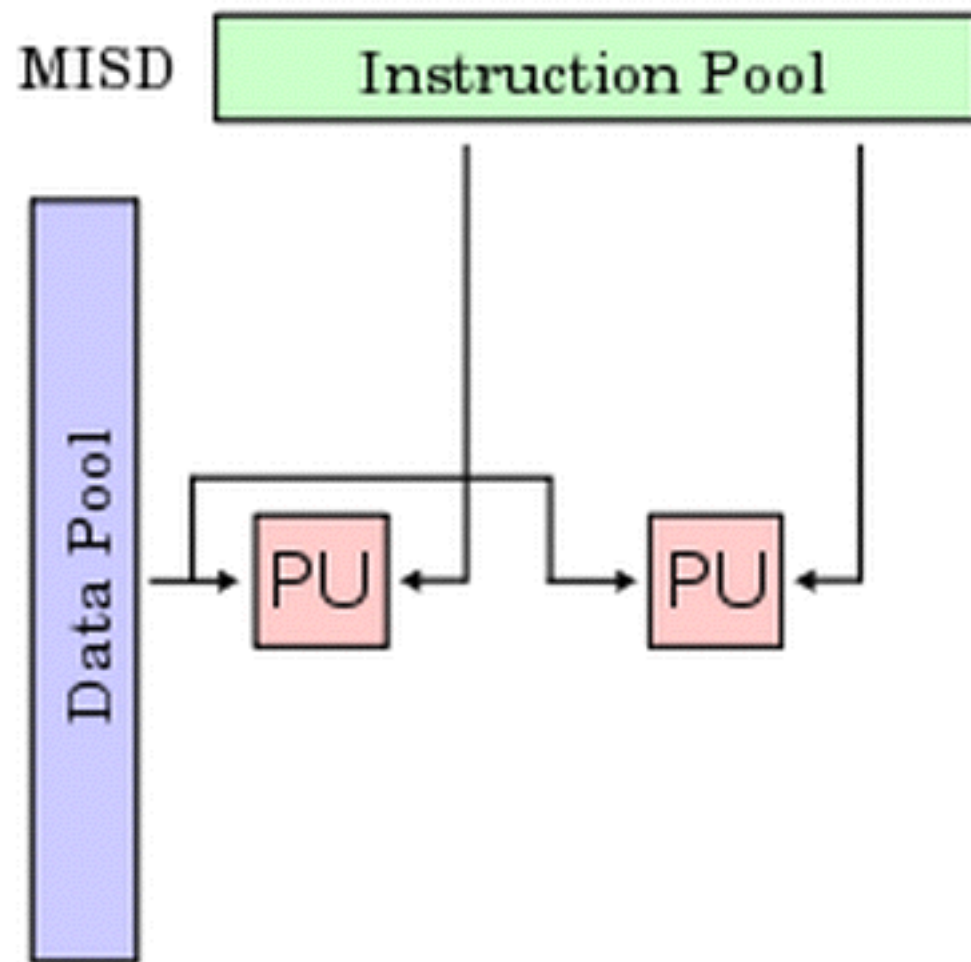
Parallel Computer Architectures

- ❖ Parallel computing means using multiple processors, possibly composed of multiple computers
- ❖ Until recently, Flynn's (1966) taxonomy was commonly used to classify parallel computers into one of four types:
 - ❖ SISD — Single instruction, single data
 - ❖ Your desktop/laptop from a few years ago
 - ❖ SIMD — Single instruction, multiple data:
 - ❖ Thinking machines CM-2
 - ❖ Cray 1, and other vector machines (things become blurry quickly)
 - ❖ Parts of modern GPUs
 - ❖ MISD — Multiple instruction, single data
 - ❖ Special purpose machines
 - ❖ No commercials, general purpose machines
 - ❖ MIMD — Multiple instruction, multiple data
 - ❖ Nearly all of today's parallel machines

Formal Taxonomy

Flynn's taxonomy

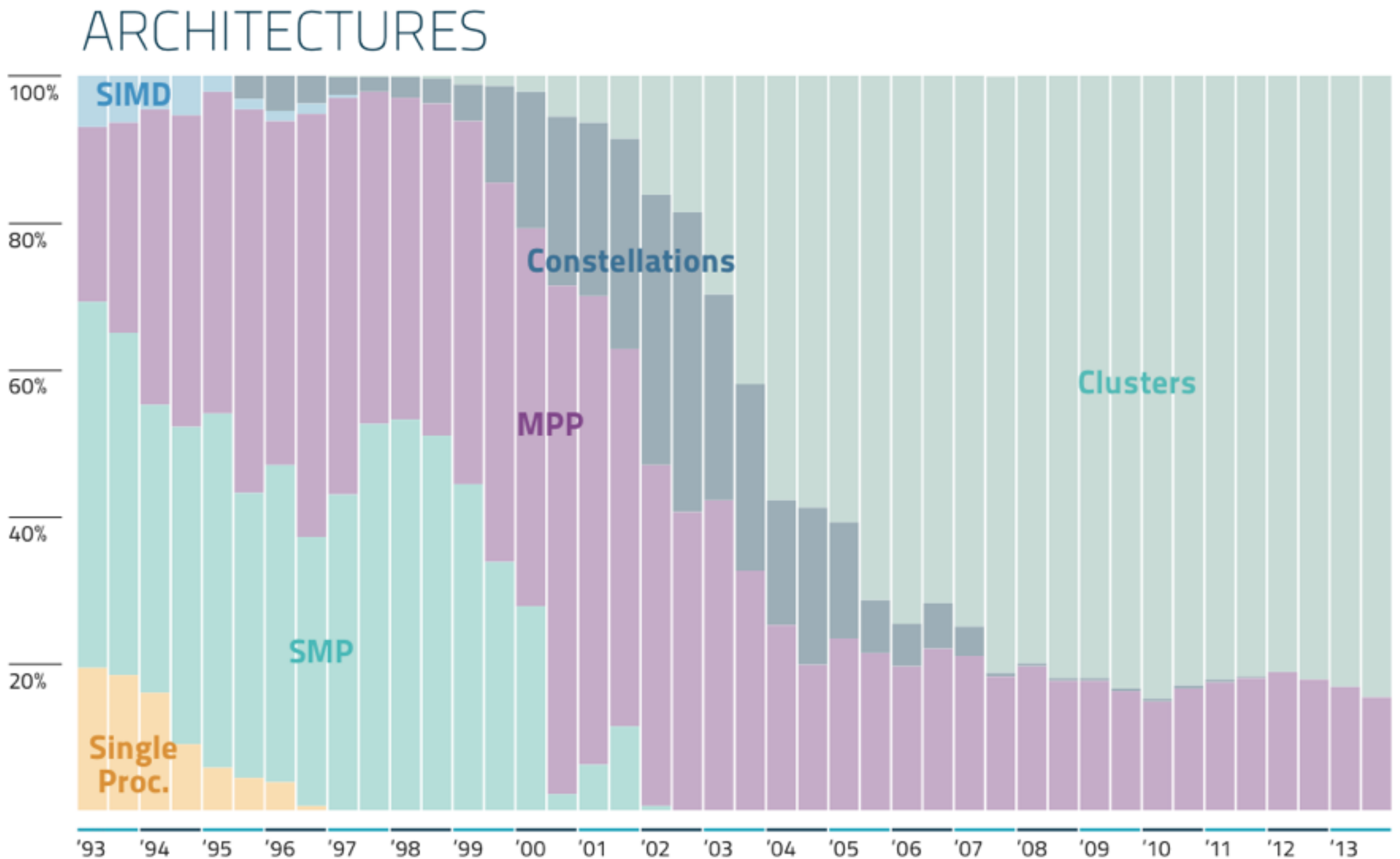
	Single instruction	Multiple instruction
Single data	SISD	MISD
Multiple data	SIMD	MIMD



Examples of MISD

- ❖ Fault-tolerance computers
 - ❖ Mainframes (UT Define!)
- ❖ Space shuttle flight computers

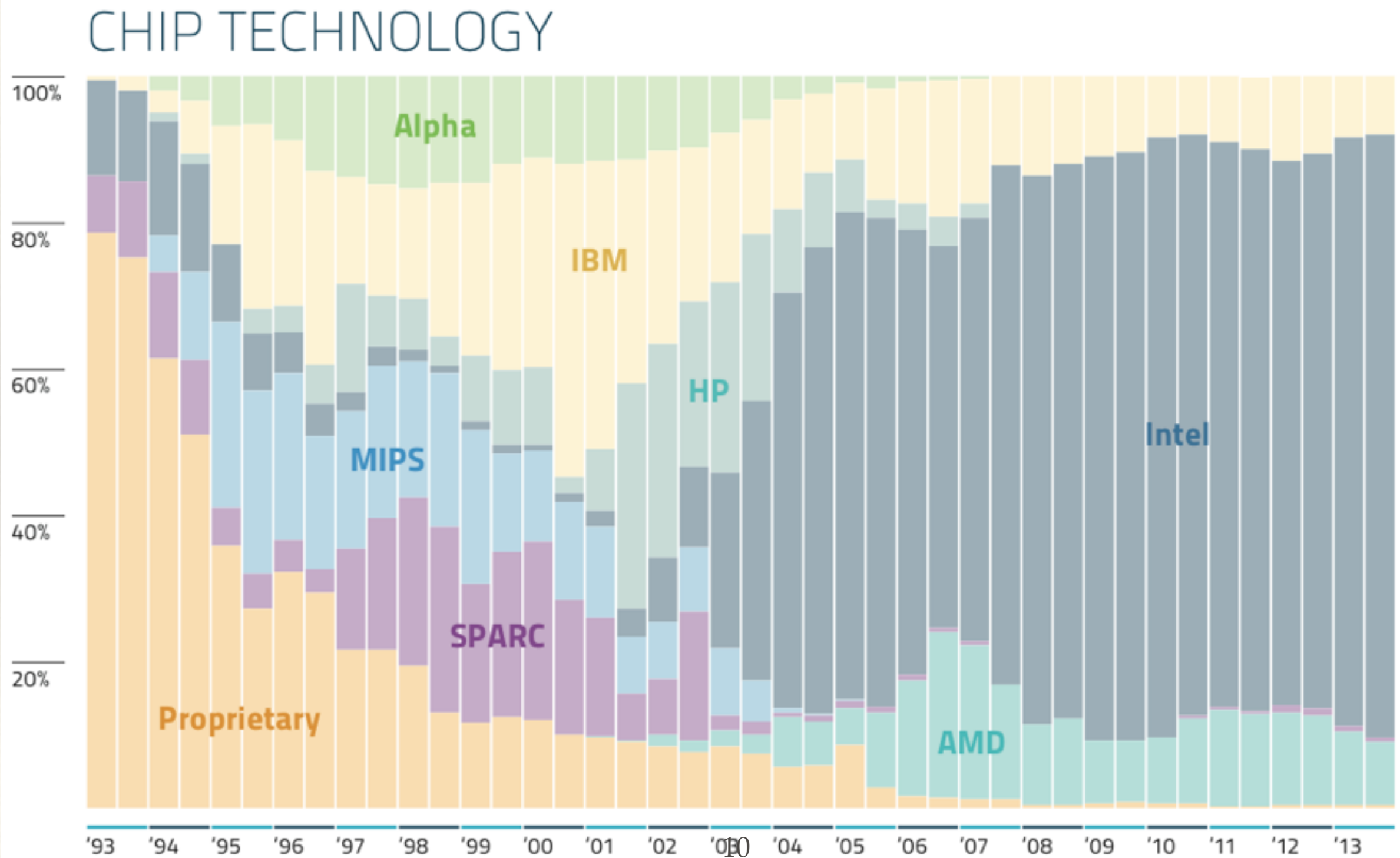
Architectures



Vector Machines

- ❖ Based on single processor with:
 - ❖ Multiple functional units
 - ❖ Each performing the same operation
- ❖ Dominated early parallel market
 - ❖ Over taken in the 90s by MPP et al.
- ❖ Making a comeback (sort of)
 - ❖ clusters / constellations of vector machines
 - ❖ Earth Simulator (NEC SX6) Cray (X1 / X1E)
 - ❖ modern micros have vector instructions MMX, SSE, AVX
 - ❖ GPUs

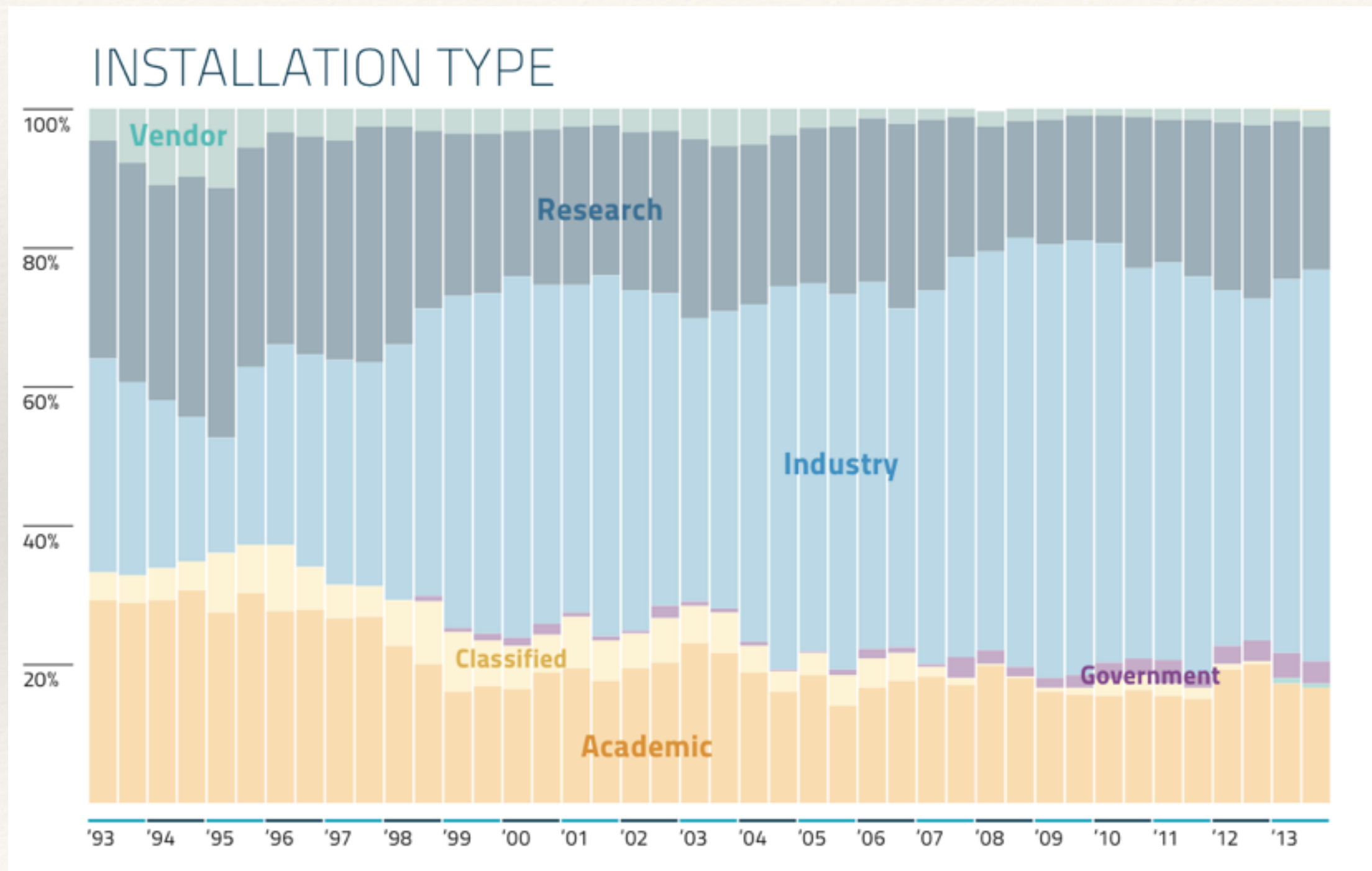
Chip Technology



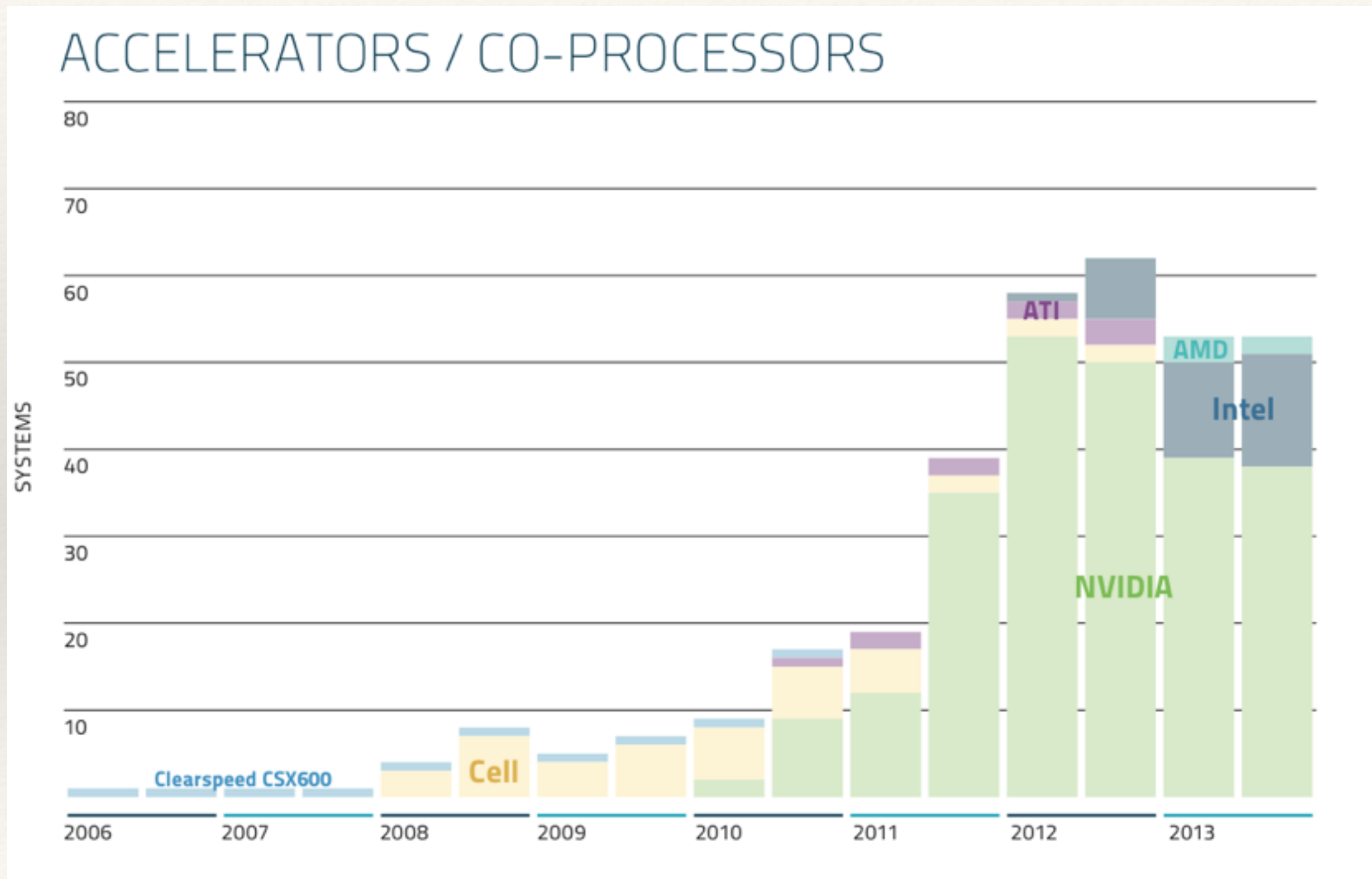
Parallel Computer Architectures

- ❖ Top500 List now dominated by MPPs, Constellations and Clusters
- ❖ The MIMD model “won”.
- ❖ A much more useful way to classification is by memory model
 - ❖ shared memory
 - ❖ distributed memory
- ❖ Note that the distinction is (mostly) logical, not physical: distributed memory systems could still be single systems (e.g. Cray XT5) or a set of computers (e.g. clusters)

Installation Type



Accelerators/Co-Processors



Economics

Table 1. Video Game Market Revenue, Worldwide, 2012-2015 (Millions of Dollars)

Segment	2012	2013	2014	2015
Video Game Console	37,400	44,288	49,375	55,049
Handheld Video Games	17,756	18,064	15,079	12,399
Mobile Games	9,280	13,208	17,146	22,009
PC Games	14,437	17,722	20,015	21,601
Total Video Game Market	78,872	93,282	101,615	111,057

The global economy in HPC is growing again:

- 2010 grew by 10%, to reach \$9.5 billion
- 2011 grew by 8.4% to reach \$10.3 billion
- HPC revenue for 2012 exceeded \$11B

Future projection

January 21, 2014

Report Estimates HPC Market to be Worth \$33.43 Billion by 2018

DALLAS, Tex., Jan. 21 – The report “High Performance Computing Market [Server, Storage, Networking, Software (Middleware, Cluster & Fabric Management, Performance Optimization), Professional Services, Deployment Models, Price Bands] – Global Market Forecasts & Analysis (2013 – 2018),” segments the global market into various sub segments with in-depth analysis of the ecosystem. It also identifies the drivers and restraints of this market with insights into trends, opportunities, and challenges.

Clusters and Constellations

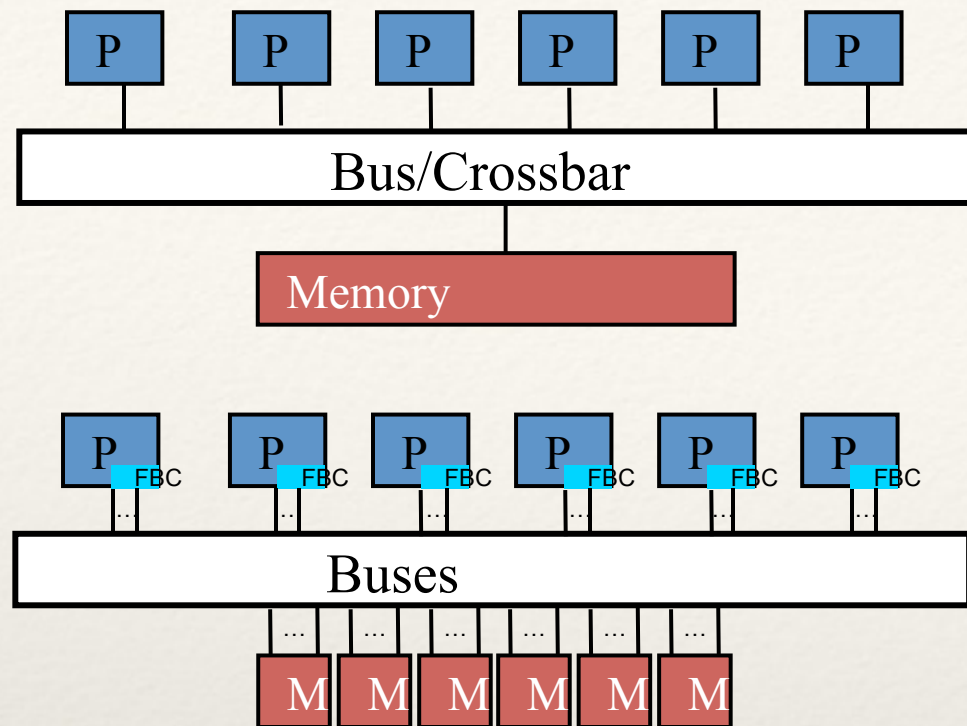
- ❖ (Commodity) Clusters

- ❖ collection of independent nodes connected with a communications network
- ❖ each node a stand-alone computer
- ❖ both nodes and interconnects available on the open market
- ❖ each node may have more than one processor (i.e., be an SMP)

- ❖ Constellations

- ❖ clusters where there are more processors within the node than there are nodes interconnected
- ❖ not very many of these any more

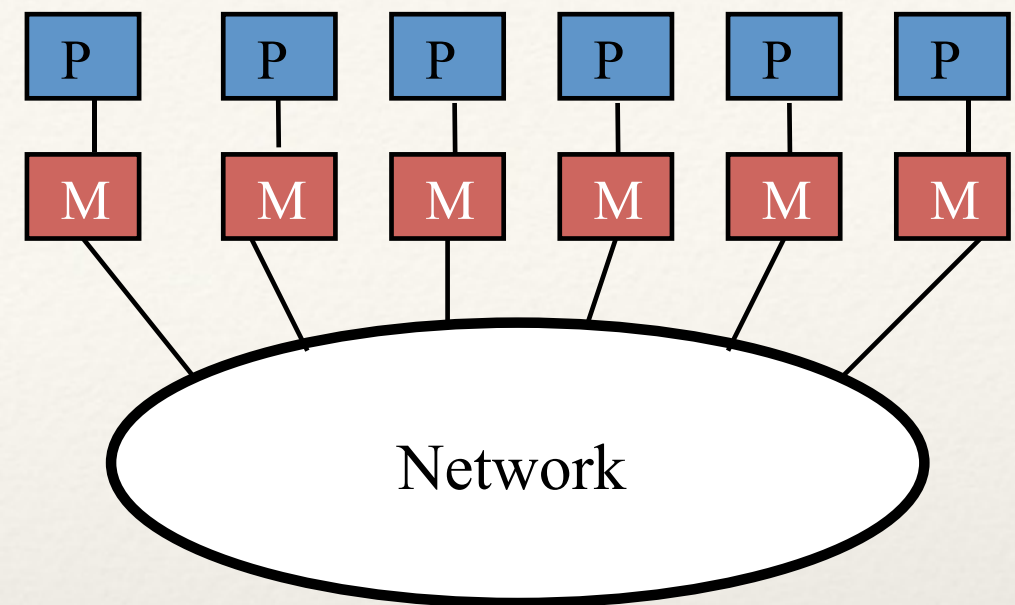
Shared and Distributed Memory



Shared memory — all processors have access to a pool of shared memory. (e.g., Single Cluster node (2-way, 4-way, ...), IBM Power6 node, Cray X1E)

Methods of memory access :

- ❖ Bus
- ❖ Distributed Switch
- ❖ Crossbar



Distributed memory — each processor has its own local memory. Must do message passing to exchange data between processors. (examples: Linux Clusters, Cray XT5)

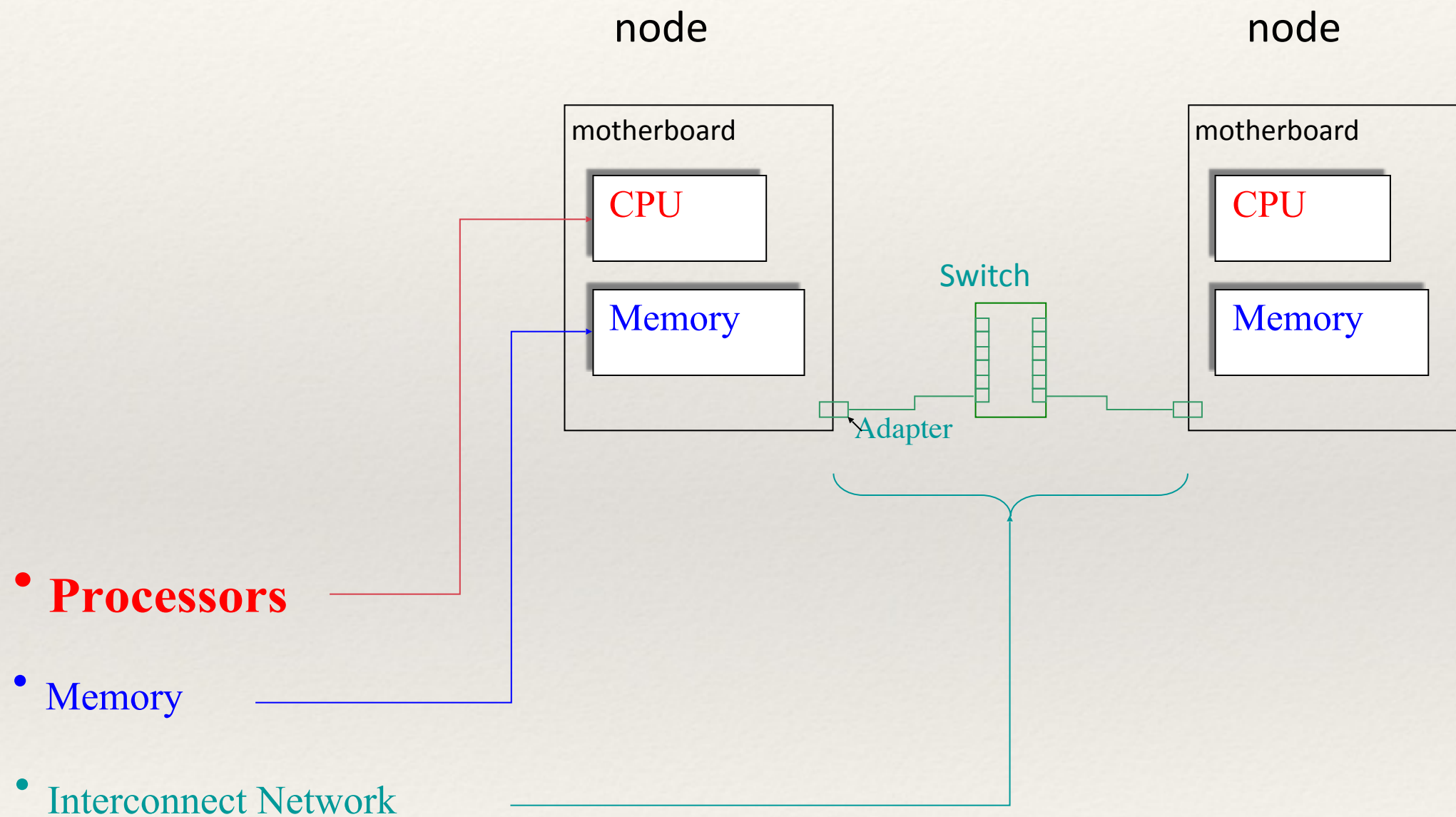
Methods of memory access :

- ❖ single switch
- ❖ switch hierarchy with many topology options

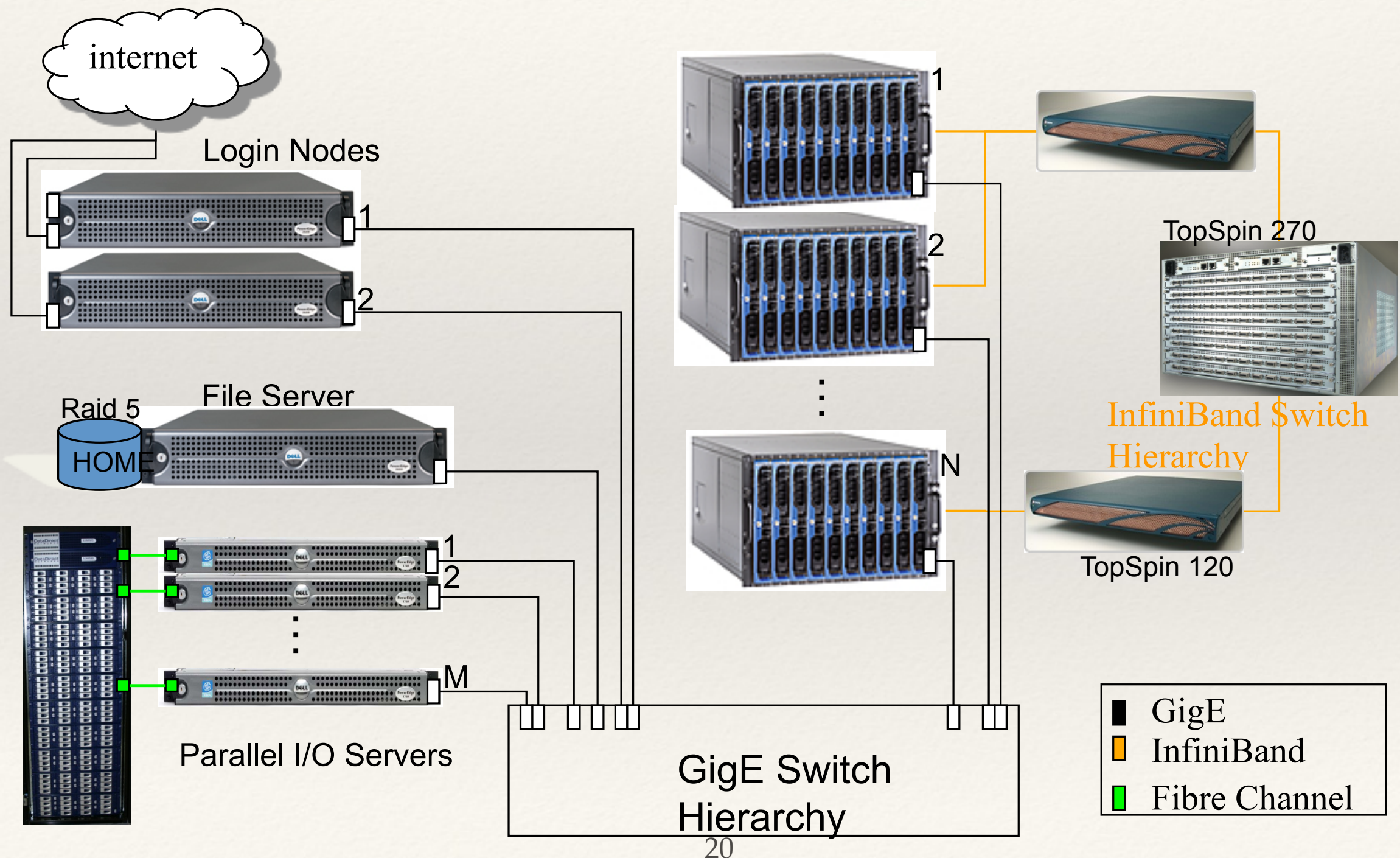
Memory Access Problems

- ❖ SMP systems do not scale well
 - ❖ bus-based systems can become saturated
 - ❖ large, fast (high bandwidth, low latency) crossbars are expensive
 - ❖ cache-coherency is hard to maintain at scale (we'll get to what this means in a minute)
- ❖ Distributed systems scale well, but:
 - ❖ they are harder to program (message passing)
 - ❖ interconnects have higher latency
 - ❖ makes parallel algorithm development and programming harder

Basic Anatomy of a Server/Desktop/Laptop/Cluster-node



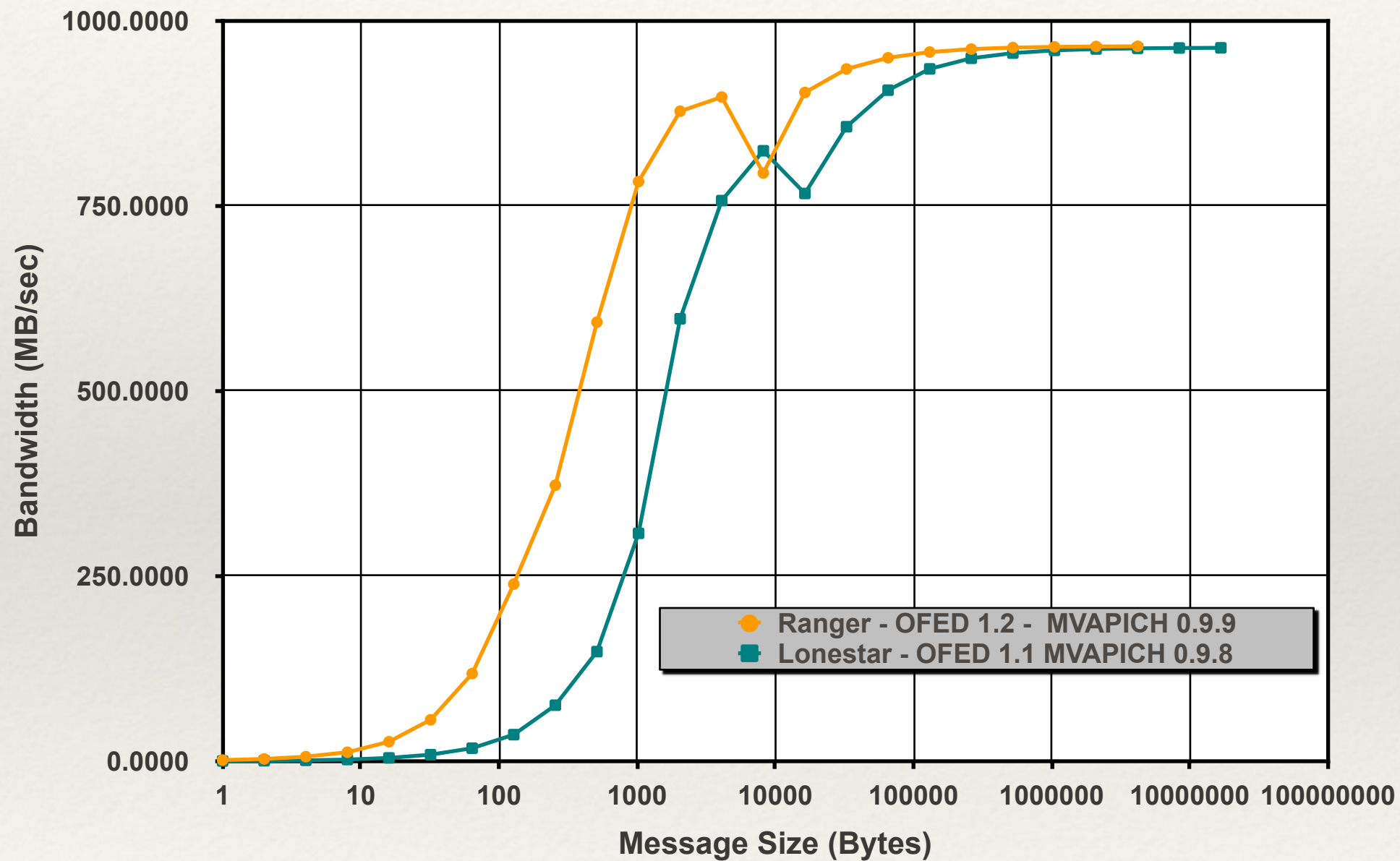
Lonestar @ TACC



Interconnects

- ❖ Started with FastEthernet (Beowulf @ NASA)
 - ❖ 100 Mb / s, 100 μ s latency
 - ❖ quickly transitioned to higher bandwidth, lower latency solutions
- ❖ Now
 - ❖ Ethernet / IP network for administrative work
 - ❖ InfiniBand, Myrinet, Quadrics for message passing

Interconnect Performance



RAID

- ❖ Was: Redundant Array of Inexpensive Disks
- ❖ Now: Redundant Array of Independent Disks
- ❖ Multiple disk drives working together to:
 - ❖ increase capacity of a single logical volume
 - ❖ increase performance
 - ❖ improve reliability / add fault tolerance
- ❖ 1 Server with RAIDed disks can provide disk access to multiple nodes with NFS

Parallel Filesystems

- ❖ Use multiple servers together to aggregate disks
 - ❖ utilizes RAIDed disks
 - ❖ improved performance
 - ❖ even higher capacities
 - ❖ may use high-performance network
- ❖ Vendors/Products
 - ❖ Oracle/Lustre
 - ❖ IBM/GPFS
 - ❖ Panassas
 - ❖ RedHat/GFS/ Gluster
 - ❖ lots of “experimental” file systems

Microarchitecture

- ❖ Memory hierarchies
- ❖ Commodity CPUs
 - ❖ theoretical performance
 - ❖ pipelining
 - ❖ superscaling
- ❖ Interconnects
 - ❖ Different topologies
 - ❖ Performance

Byte Prefixes

Prefixes for multiples of bits (b) or bytes (B)							
Decimal				Binary			
Value		Metric		Value	JEDEC		IEC
1000	k	kilo		1024	K	kilo	Ki kibi
1000 ²	M	mega		1024 ²	M	mega	Mi mebi
1000 ³	G	giga		1024 ³	G	giga	Gi gibi
1000 ⁴	T	tera		1024 ⁴	-	-	Ti tebi
1000 ⁵	P	peta		1024 ⁵	-	-	Pi pebi
1000 ⁶	E	exa		1024 ⁶	-	-	Ei exbi
1000 ⁷	Z	zetta		1024 ⁷	-	-	Zi zebi
1000 ⁸	Y	yotta		1024 ⁸	-	-	Yi yobi

Bits, Bytes and Words

- ❖ Bit – Binary digit 0, 1
- ❖ Byte – 8 bits
- ❖ Word
 - ❖ 4 bytes (32 bit architecture)
 - ❖ 8 bytes (64 bit architecture)
- ❖ With 8 bits in a byte we can represent 256 values ranging from 0 to 255
 - ❖ 0 = 00000000
 - ❖ 1 = 00000001
 - ❖ 2 = 00000010
 - ❖ ...
 - ❖ 254 = 11111110
 - ❖ 255 = 11111111

32-bit vs 64-bit

- ❖ How much is addressable?
- ❖ 32 bit architecture
 - ❖ Each bit can have two possible values [0,1]
 - ❖ $2^{32} = 4,294,967,296$ different values
 - ❖ $2^{30} = 1$ Gibabyte
 - ❖ ~ 4 GB available to address

32-bit vs 64-bit

- ❖ How much is addressable?
- ❖ 64 bit architecture
 - ❖ $2^{64} = 18,446,744,073,709,551,616$
 - ❖ That's approximately
 - ❖ 17.2 billion Gigabytes
 - ❖ 16.8 million Terabytes
 - ❖ or 16 Exabytes.

64-bit Limitations

- ❖ Most 64-bit CPUs have an artificial limit on the amount of memory they can address.
- ❖ AMD's 64-bit architecture allows 48-bits
- ❖ That is still 2^{48} or 256 TB
- ❖ Architecture definition allows this to be raised to the full 2^{64} or 16 EB in future implementations

Memory Hierarchies

- ❖ Due primarily to cost, memory is divided into different levels:
 - ❖ Registers
 - ❖ Caches
 - ❖ Main Memory
- ❖ Memory is accessed through the hierarchy
 - ❖ registers where possible
 - ❖ ... then the caches
 - ❖ ... then main memory

Memory Relativity

SPEED

SIZE

Cost (\$/bit)



CPU
Registers

L1 cache
(SRAM)

L2 cache
(SRAM)

MEMORY
(DRAM)

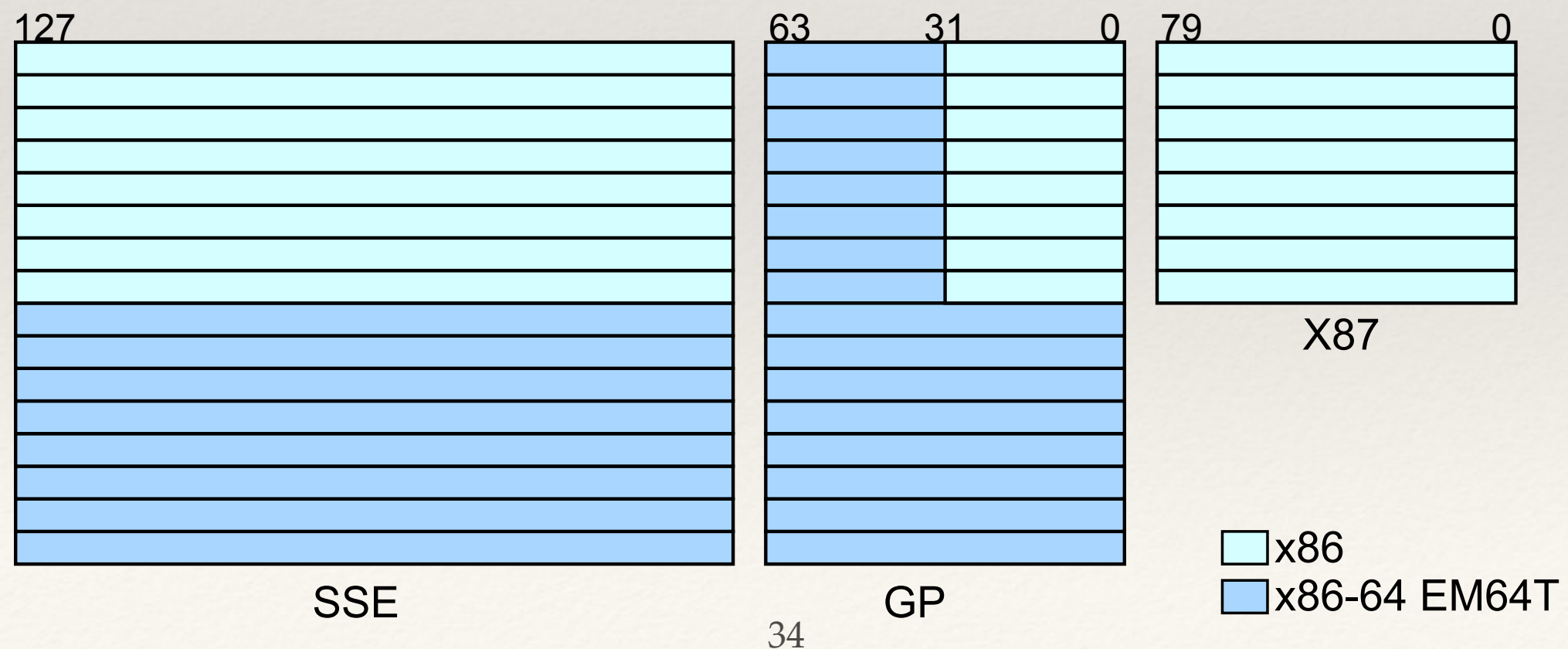


Latency and Bandwidth

- ❖ The two most important terms related to performance for memory subsystems and for networks are:
 - ❖ Latency
 - ❖ How long does it take to retrieve a word of memory?
 - ❖ Units are generally nanoseconds or clock periods (CP).
 - ❖ Bandwidth
 - ❖ What data rate can be sustained once the message is started?
 - ❖ Units are B/sec (MB/sec, GB/sec, etc.)

Registers

- ❖ Highest bandwidth, lowest latency memory that a modern processor can access
- ❖ built into the CPU
 - ❖ often a scarce resource
 - ❖ not RAM
- ❖ AMD x86-64 and Intel EM64T Registers



Registers

- ❖ Processors instructions operate on registers directly
 - ❖ have assembly language names like:
 - ❖ `eax, ebx, ecx, etc.`
 - ❖ sample instruction:
 - ❖ `addl %eax, %edx`
- ❖ Separate instructions and registers for floating-point operations

Data Caches

- ❖ Between the CPU Registers and main memory
- ❖ L1 Cache: Data cache closest to registers
- ❖ L2 Cache: Secondary data cache, stores both data and instructions
 - ❖ Data from L2 has to go through L1 to registers
 - ❖ L2 is 10 to 100 times larger than L1
 - ❖ Some systems have an L3 cache, ~10x larger than L2
- ❖ Cache line
 - ❖ The smallest unit of data transferred between main memory and the caches (or between levels of cache)
 - ❖ N sequentially-stored, multi-byte words (usually N=8 or 16).

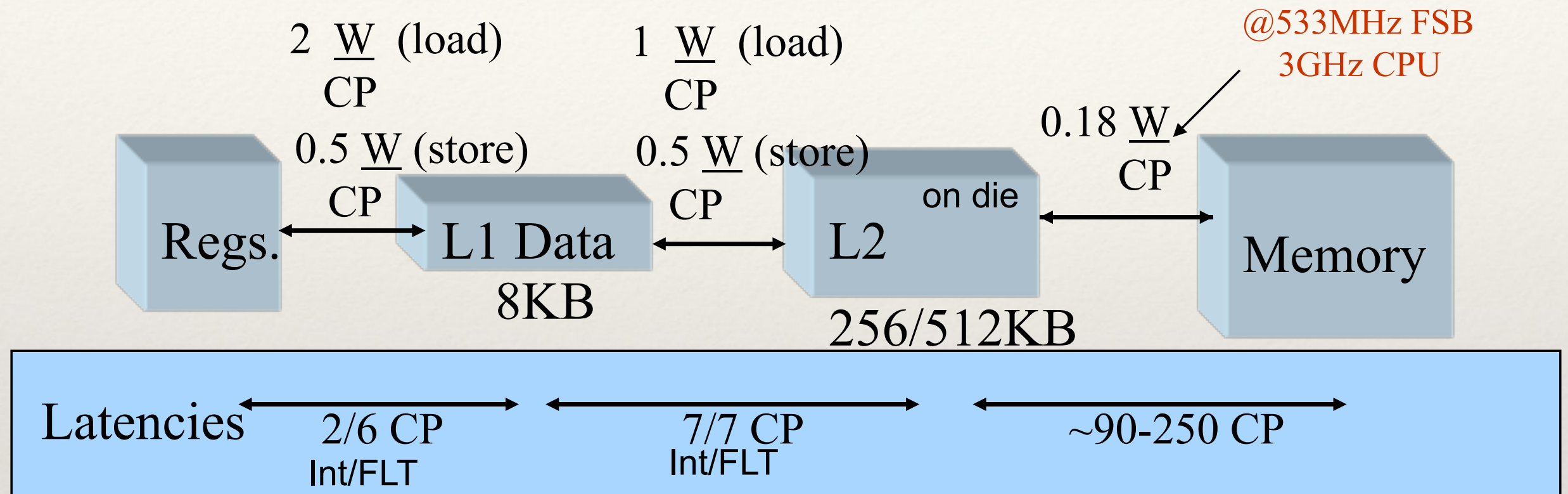
Main Memory

- ❖ Cheapest form of RAM
- ❖ Also the slowest
 - ❖ lowest bandwidth
 - ❖ highest latency
- ❖ Unfortunately most of our data lives out here

Approximate Latencies and Bandwidths in a Memory Hierarchy

	Latency	Bandwidth
Registers		
L1	$\sim 5 \text{ CP}$	$\sim 2 \text{ W} / \text{CP}$
L2	$\sim 15 \text{ CP}$	$\sim 1 \text{ W} / \text{CP}$
Memory	$\sim 300 \text{ CP}$	$\sim 0.25 \text{ W} / \text{CP}$
Dist. Mem	$\sim 10000 \text{ CP}$	$\sim 0.01 \text{ W} / \text{CP}$

Example: Pentium 4



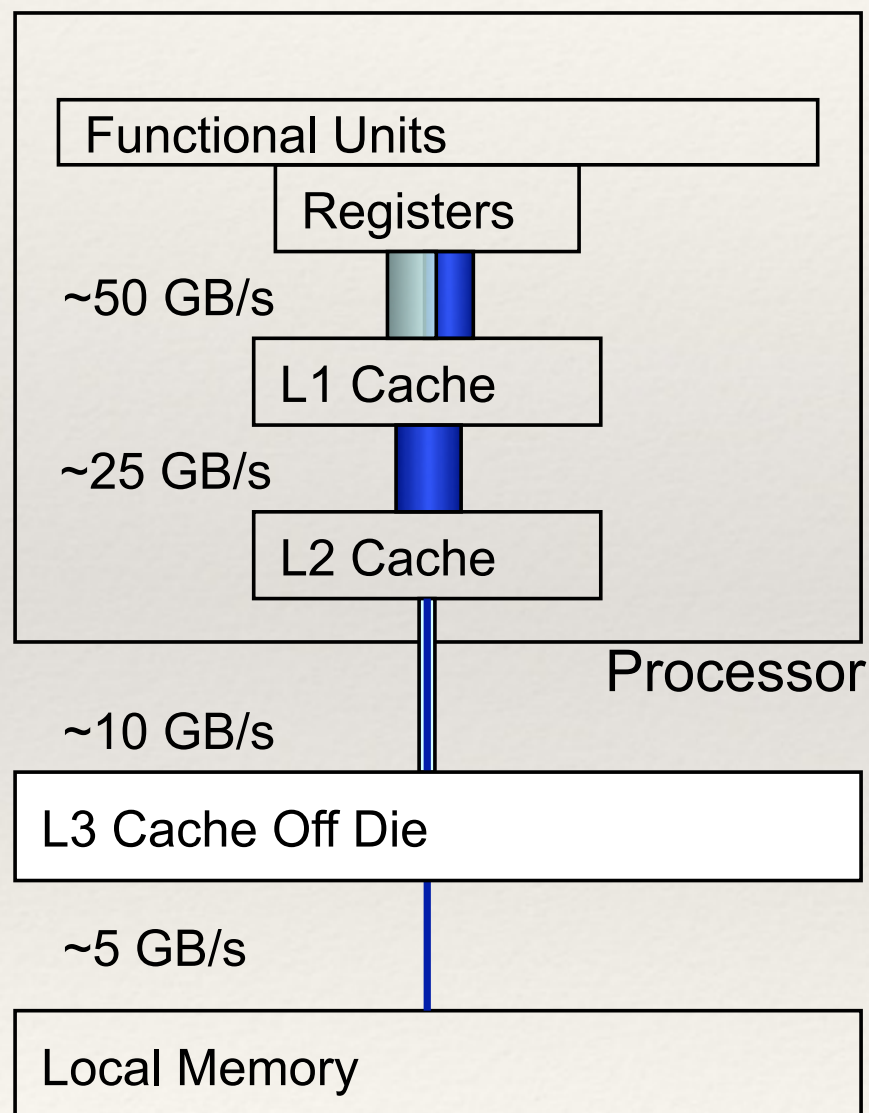
Homework 0.1

Line size L1 / L2 = 8W / 16W

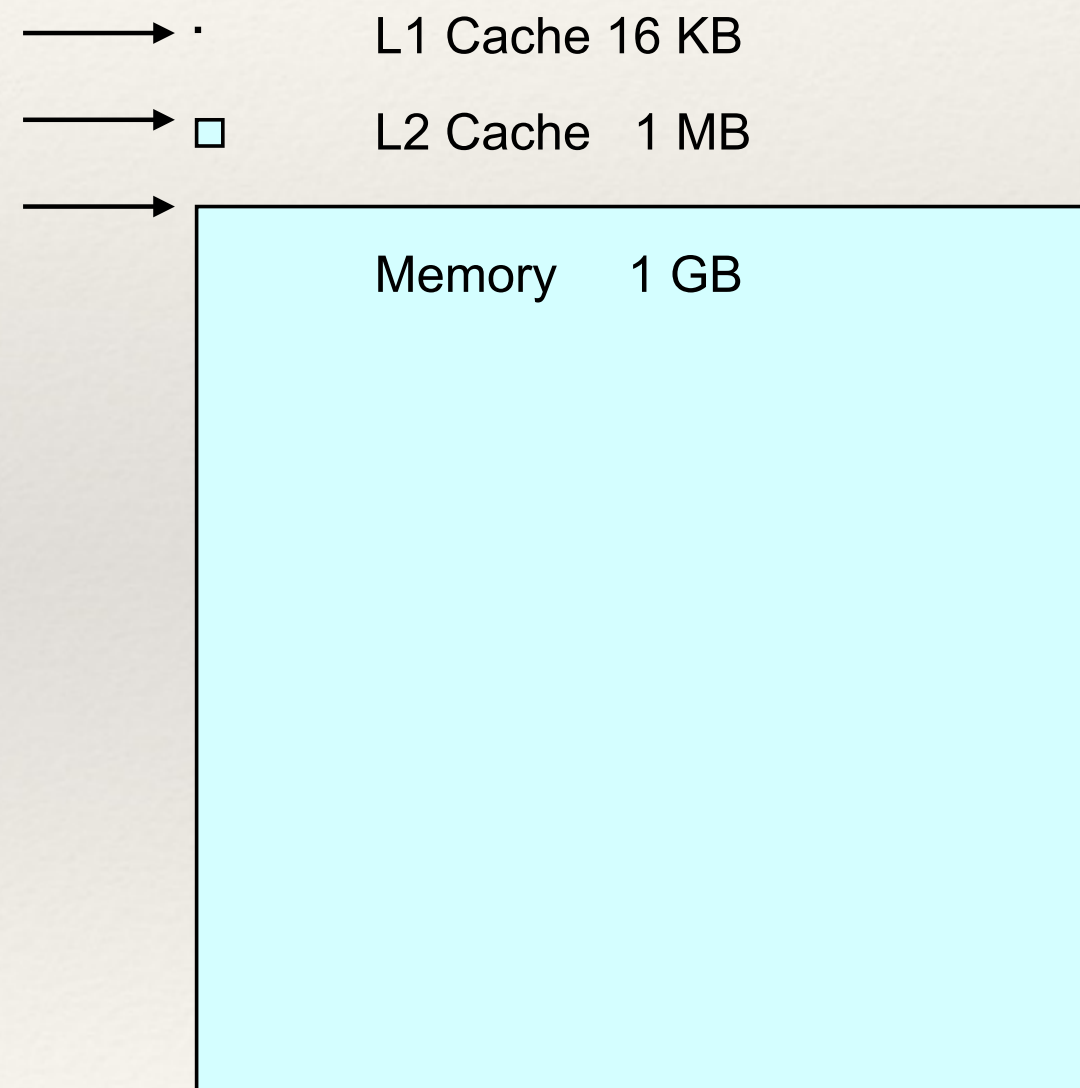
- ❖ Using the TACC website and the interwebs, construct a similar diagram for stampede
- ❖ Why is stampede faster if the clock speed is slower?
- ❖ Again, you have a choice as to what program you like for now. You will be assimilated eventually.

Memory Bandwidth and Size Diagram

Relative Memory Bandwidths



Relative Memory Sizes



Why Caches?

- ❖ Since registers are expensive
- ❖ ... and main memory slow
- ❖ Caches provide a buffer between the two
- ❖ Access is transparent
 - ❖ either it's in a register or...
 - ❖ it's in a memory location
 - ❖ processor / cache controller / MMU hides cache access from the programmer

Hits, Misses, Thrashing

- ❖ Cache hit
 - ❖ location referenced is found in the cache
- ❖ Cache miss
 - ❖ location referenced is not found in cache
 - ❖ triggers access to the next higher cache or memory
- ❖ Cache thrashing
 - ❖ a thrashed cache line (TCL) must be repeatedly recalled in the process of accessing its elements
 - ❖ caused when other cache lines, assigned to the same location, are simultaneous accessing data/instructions that replace the TCL with their content.

Design Considerations

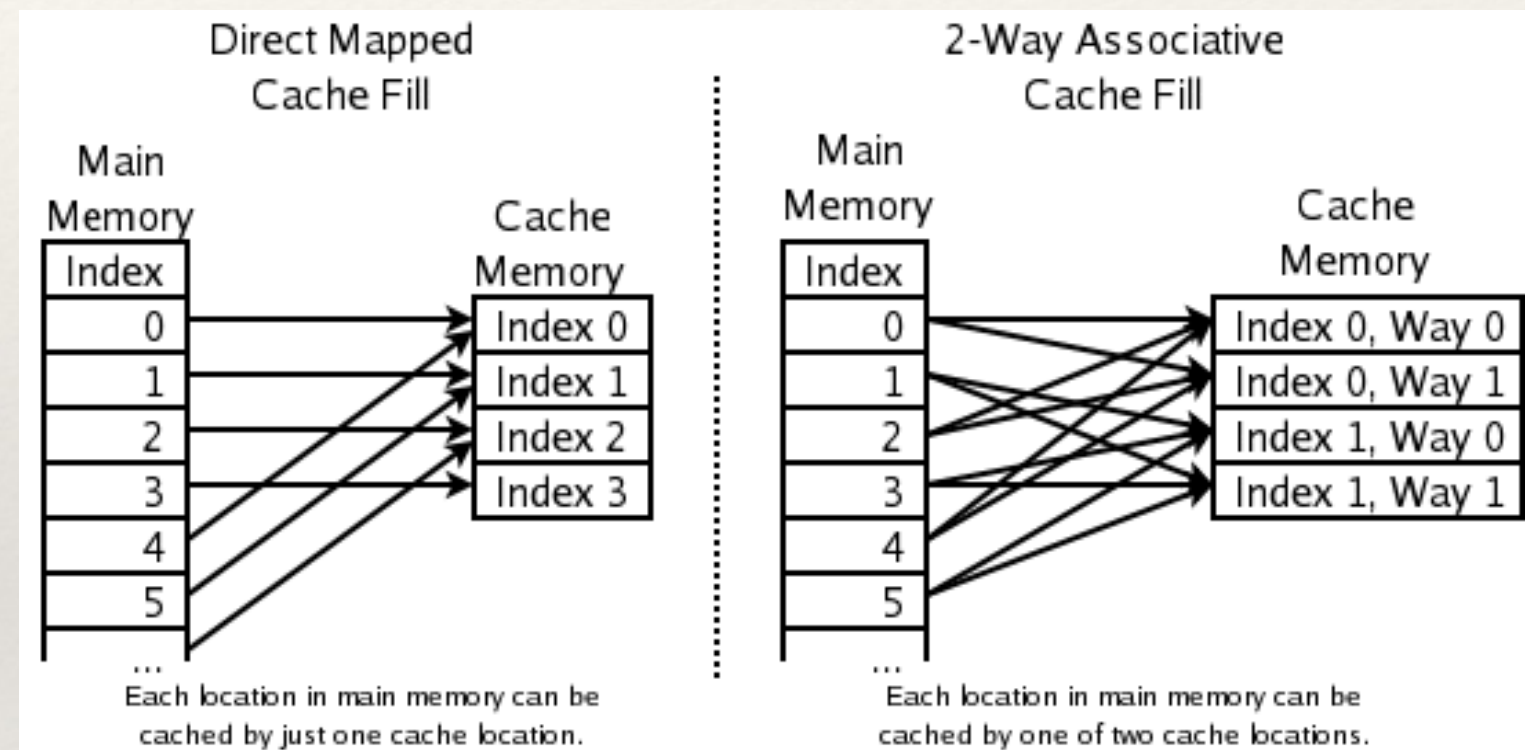
- ❖ Data cache designed with two key concepts in mind
- ❖ Spatial Locality
 - ❖ when an element is referenced, its neighbors will be referenced, too
 - ❖ all items in the cache line are fetched together
 - ❖ work on consecutive data elements in the same cache line gives performance boost
- ❖ Temporal Locality
 - ❖ when an element is referenced, it will be referenced again soon
 - ❖ arrange code so that data in cache is reused as often as possible

Cache Line Size vs Access Mode

Access Line-size	Sequential data	Random data
Short Line	more fetches (overhead)	Best—low “latency”
Long Line	Best--Fewer fetches, but higher probability for cache thrashing.	Longer “latency”, effectively smaller cache

Intel Woodcrest Caches

- ❖ L1
 - ❖ 32 KB
 - ❖ 8-way set associative
 - ❖ 64 byte line size
- ❖ L2
 - ❖ 4 MB
 - ❖ 8-way set associative
 - ❖ 64 byte line size



Theoretical Performance for Woodcrest

- ❖ How many operations per clock cycle?
 - ❖ Intel Woodcrest: 4 Flop / cp
- ❖ Clock rate?
 - ❖ 2.66 GHz
- ❖ $4 \text{ Flop / cp} * 2.66 \text{ Gcp / s} = 10.64 \text{ GFlops}$
- ❖ 2 W / cp (loaded from L1 cache) for reads
 - ❖ $2 * 2.66 * 8 = 42.56 \text{ GB / s}$
- ❖ $\sim 0.25 \text{ W / cp}$ (from main memory)
 - ❖ $0.25 * 2.66 * 8 = 5.32 \text{ GB / s}$

CPU	FLOPS / cp
Intel Xeon E5-2600 “Sandy Bridge”	8
Intel Xeon E3-1200 “Ivy Bridge”	8
AMD Opteron 6200 “Bulldozer”	4
AMD Opterton 6300 “Piledriver”	4

Cluster performance

- ❖ Theoretical Peak Performance

- ❖ $\text{GFLOPS} = \text{node} * (\text{sockets} / \text{node}) * (\text{cores} / \text{socket}) * \text{GHz} * \text{FLOPS}$

- ❖ Efficiency

- ❖ $\text{Actual Performance GFLOPS} / \text{Theoretical Peak Performance} * 100$

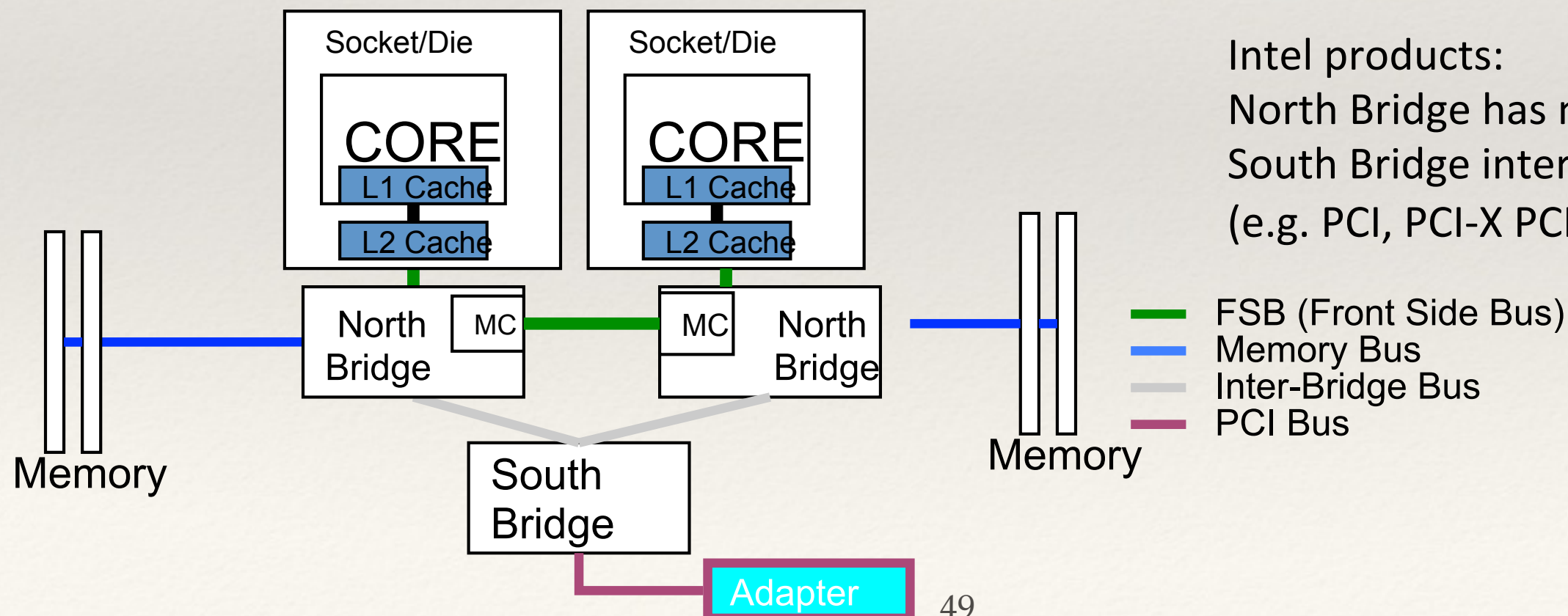
Homework 0.2

- ❖ Using the TACC homepage and the interwebs, cook up similar information for stampede

Intel System Architecture

❖ Basic Components of a Compute Node

- ❖ Function Units: perform operations (e.g. Floating Point/Integer ops, load, stores, etc.)
- ❖ Cores: (1-4) contains a set of functional units that function as an independent processor.
- ❖ Caches: on-die
- ❖ Bridges: Interconnects two different busses (may contain a “controller”)



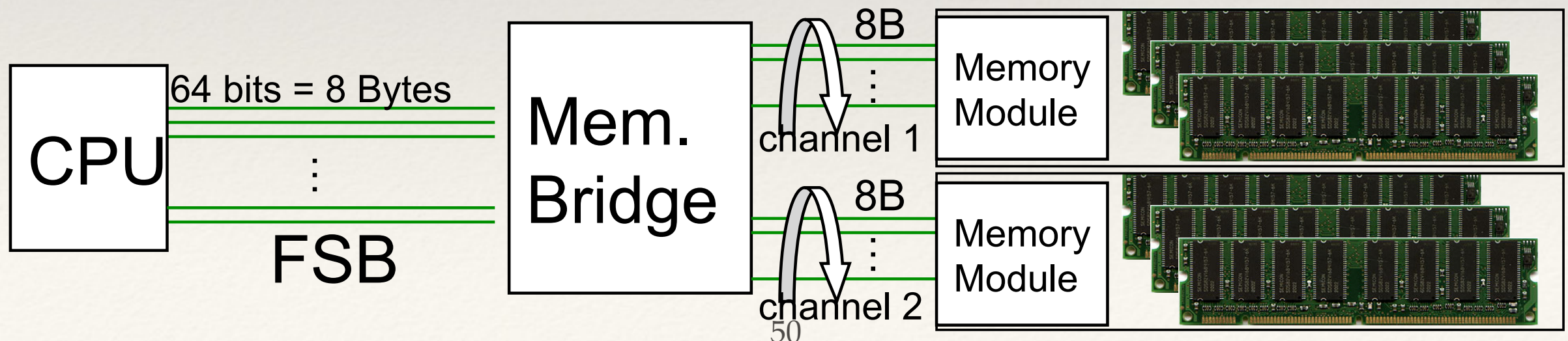
Memory Bandwidth

- ❖ Component Bandwidth (BW):

- ❖ The bus between the CPU and the Memory controller is known as the front-side bus (FSB). Multiply the frequency times the bus width to obtain bandwidth.
- ❖ The bus between the Memory Controller and the DIMMS determines the “Memory” speed. Multiply the frequency, bus width and number of channels to obtain an “aggregate” bandwidth.

- ❖ $BW \text{ (memory)} = 533 \text{ MHz (1/s)} * 8B \text{ (W)} * 2 = 8.5 \text{ GB/s}$

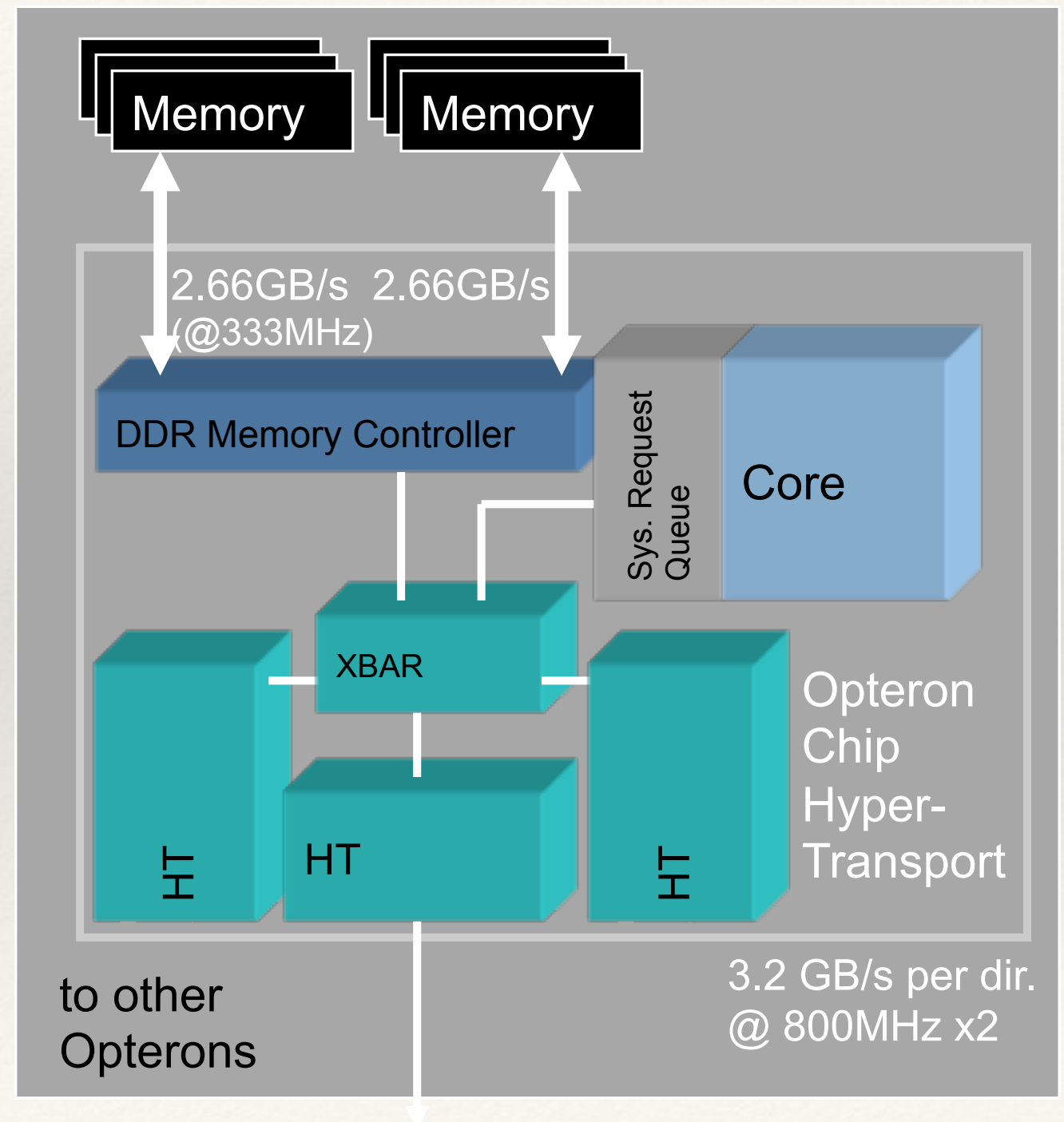
- ❖ $BW \text{ (FSB)} = 1.33 \text{ GHz (1/s)} * 8B \text{ (W)} = 10.6 \text{ GB/s}$



AMD System Architecture

HyperTransport: New technology & protocol for data transfer—point-to-point links (chip-to-chip)

Crossbar switches between memory and HyperTransport (effectively, the Front Side Bus)



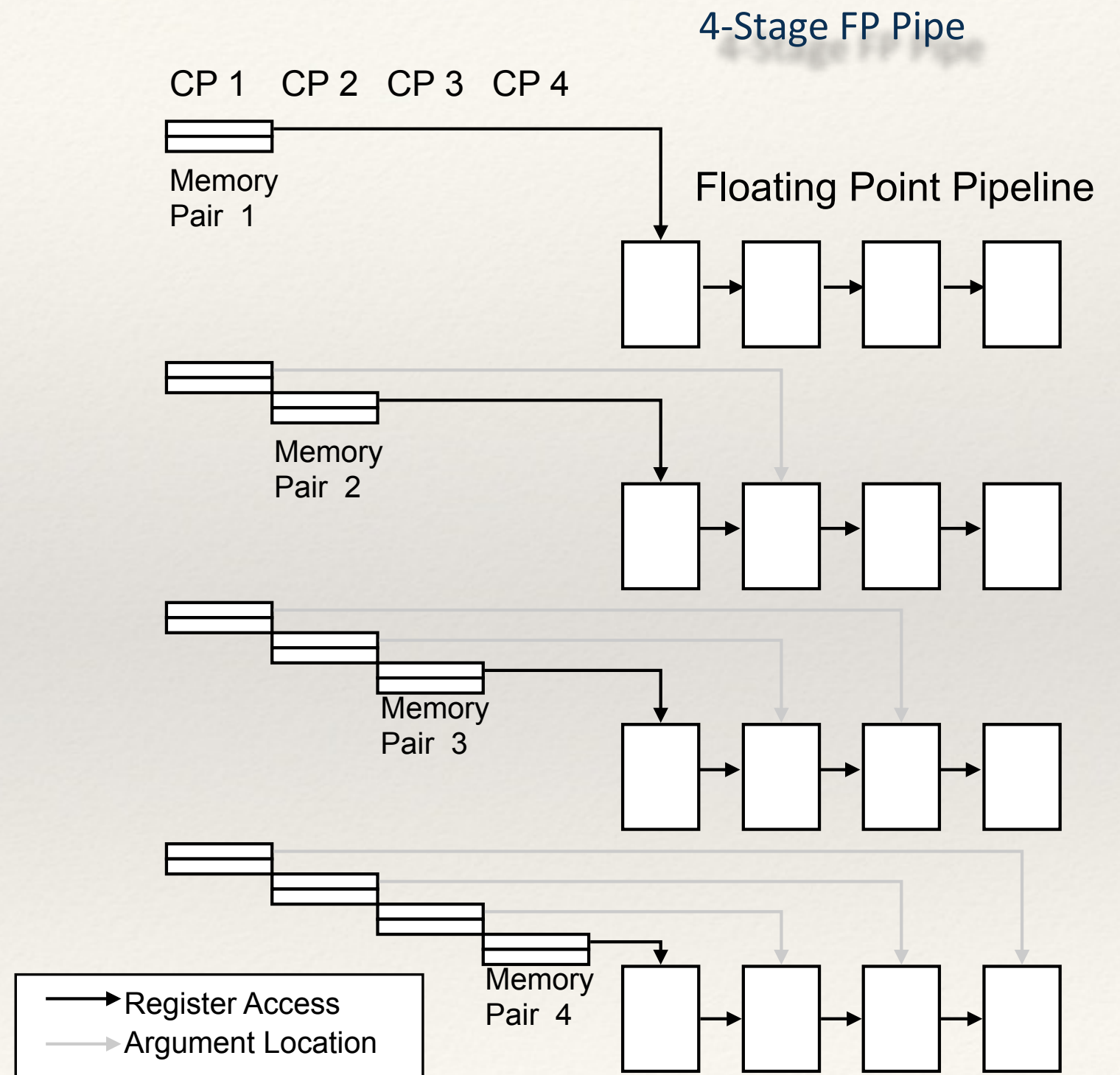
Pipeline

A serial multistage functional unit.
Each stage can work on different
sets of independent operands
simultaneously.

After execution in the final stage,
first result is available.

Latency = # of stages * CP/stage

CP/stage is the same for
each stage and usually 1.



Branch Prediction

- ❖ The “instruction pipeline” is all of the processing steps (also called segments) that an instruction must pass through to be “executed”
- ❖ Higher frequency machines have a larger number of segments
- ❖ Branches are points in the instruction stream where the execution may jump to another location, instead of executing the next instruction
- ❖ For repeated branch points (within loops), instead of waiting for the loop to branch route outcome, it is predicted.

Pentium III processor pipeline

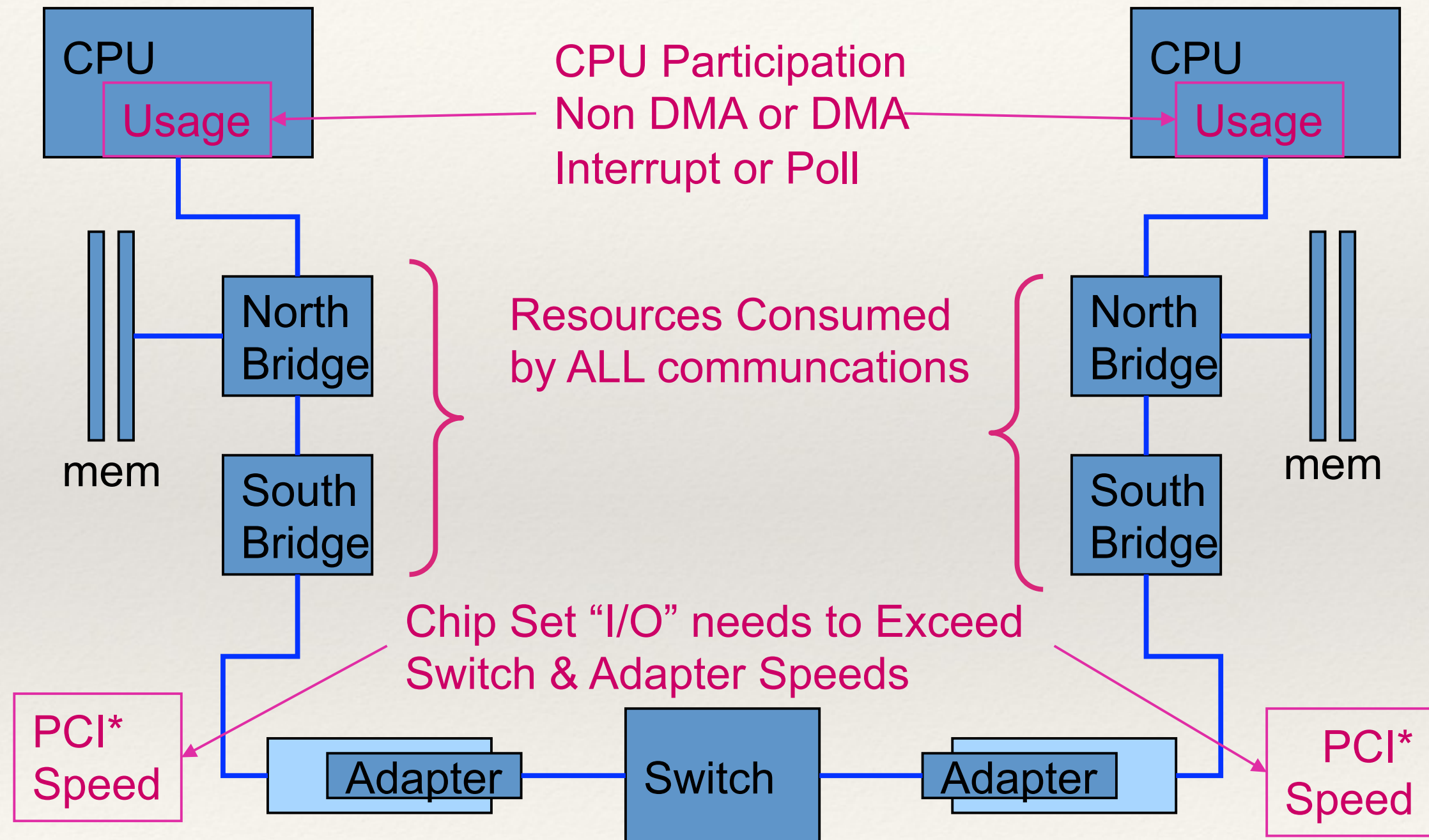
1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Pentium 4 processor pipeline

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

Misprediction is more “expensive” on Pentium 4’s.

Hardware View of Communication (Intel)



Node Communication in Clusters

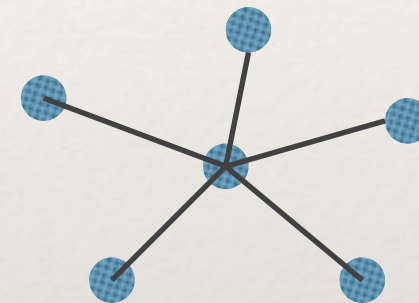
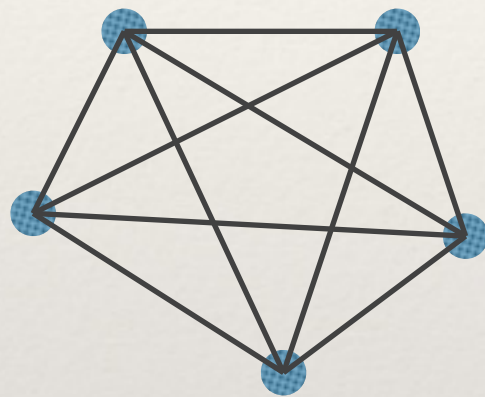
- ❖ Latency : How long does it take to start sending a "message"?
 - ❖ Units are generally microseconds or milliseconds.
- ❖ Bandwidth : What data rate can be sustained once the message is started?
 - ❖ Units are Mbytes / sec or Gbytes / sec.
- ❖ Topology: What is the actual 'shape' of the interconnect?
 - ❖ Are the nodes connect by a 2D mesh?
 - ❖ A ring?
 - ❖ Something more elaborate?

Node Communication in Clusters

- ❖ Processors can be connected by a variety of interconnects
- ❖ Static/Direct
 - ❖ point-to-point, processor-to-processor
 - ❖ no switch
- ❖ Dynamic
 - ❖ processors connect to switches
 - ❖ major types are crossbar and fat tree
- ❖ Major types / topologies for Static/Direct topologies
 - ❖ completely connected
 - ❖ star
 - ❖ linear array
 - ❖ ring
 - ❖ n-d mesh
 - ❖ n-d torus
 - ❖ n-d hyper cube

Completely Connected and Star Networks

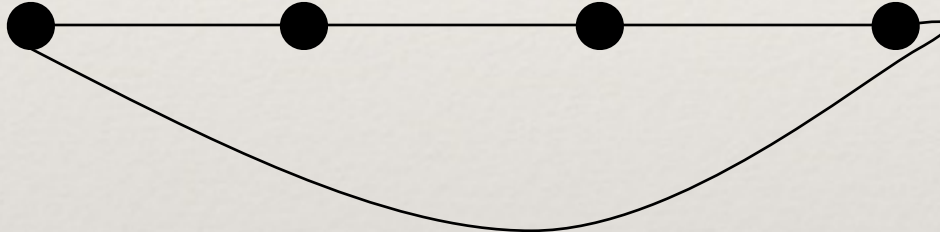
- ❖ Completely Connected : Each processor has direct communication link to every other processor



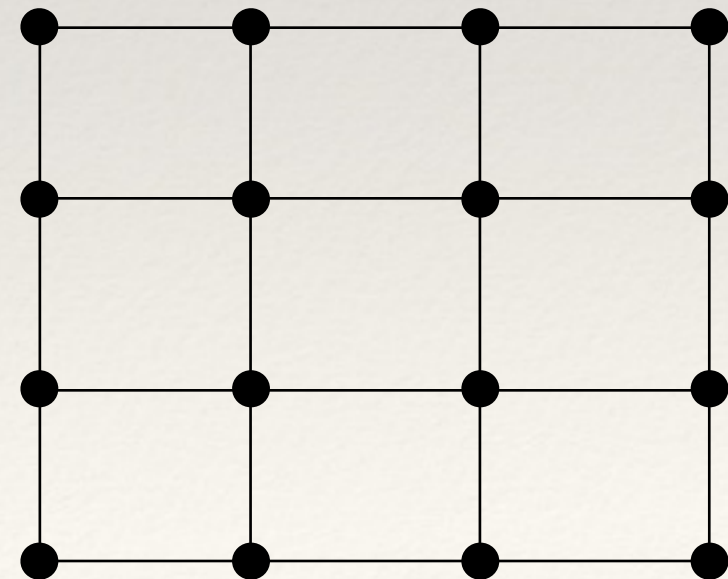
- ❖ Star Connected Network : The middle processor is the central processor. Every other processor is connected to it.
- ❖ Counterpart of Cross Bar switch in Dynamic interconnect.

Arrays and Rings

- Linear Array : 

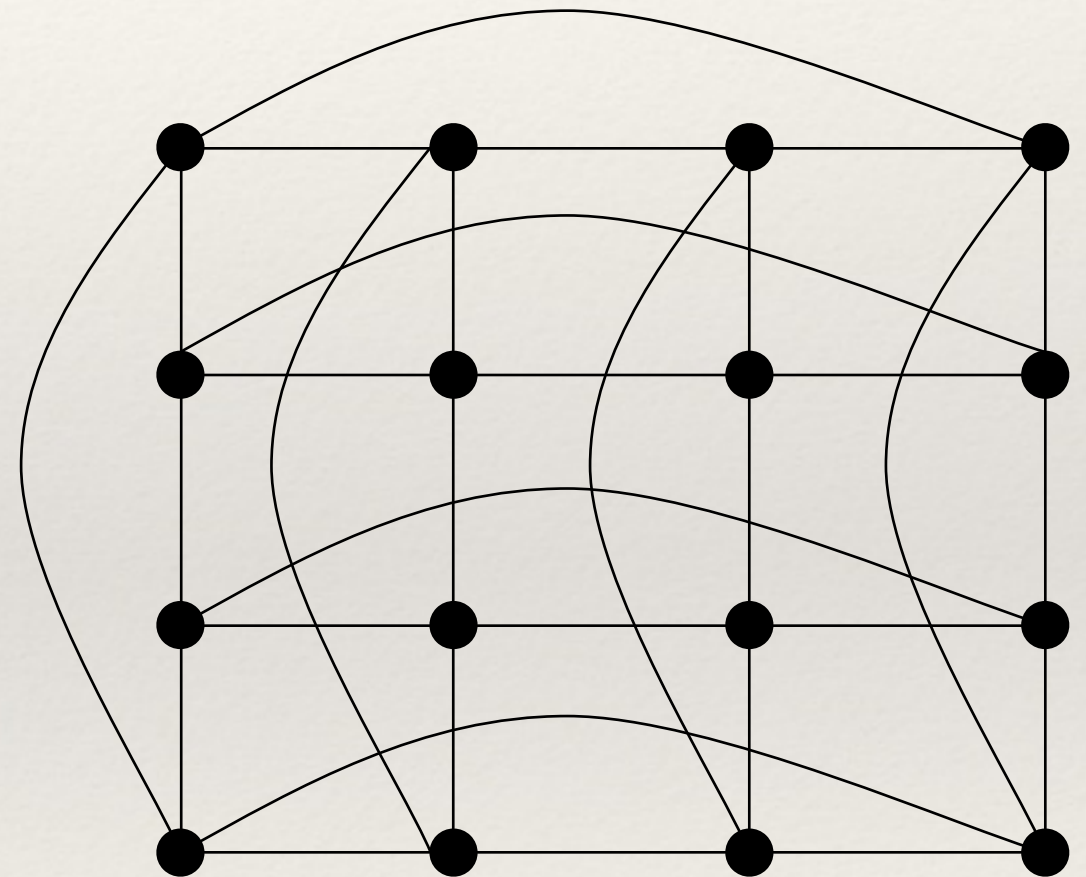
- Ring : 

- Mesh Network (e.g. 2D-array)



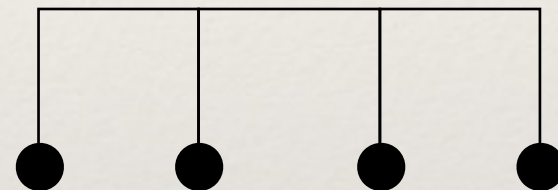
Torus

2-d Torus (2-d version of the ring)

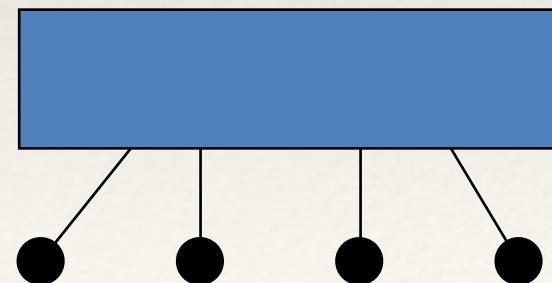


Busses/Hubs and Crossbars

Hub / Bus: Every processor shares the communication links

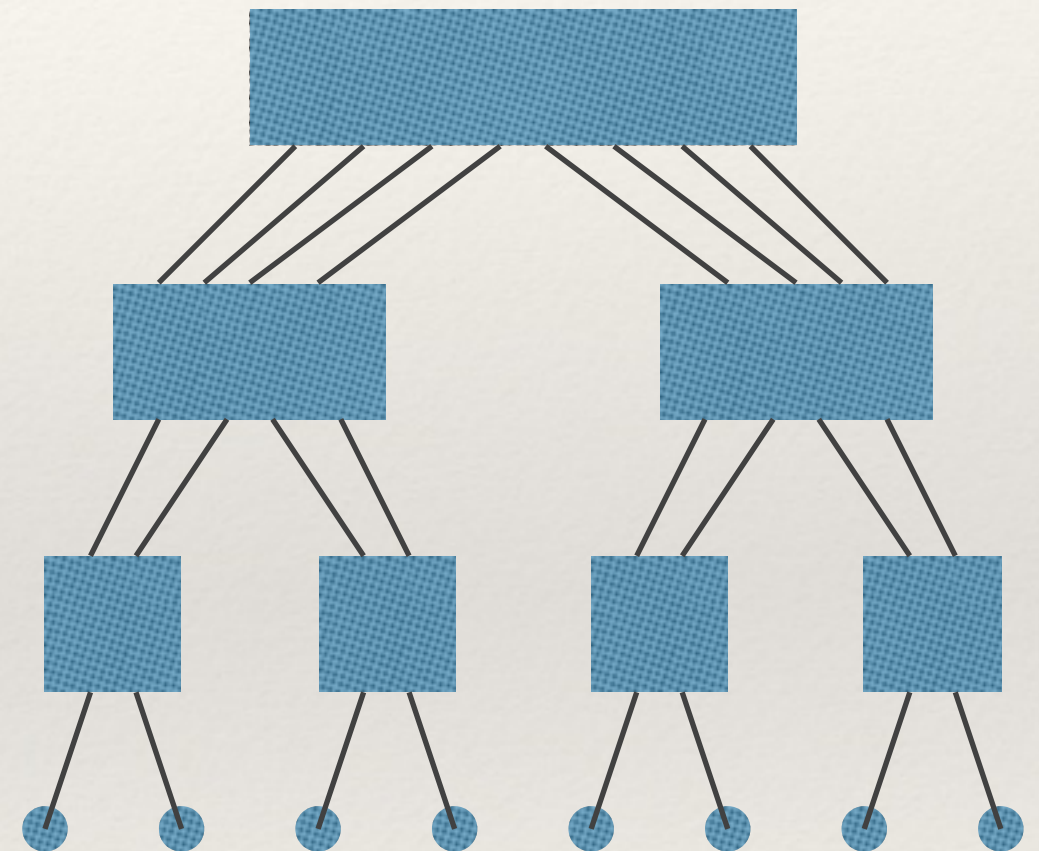


Crossbar Switches: Every processor connects to the switch which routes communications to their destinations



Fat Trees

- ❖ Multiple switches
- ❖ Each level has the same number of links in as out
- ❖ Increasing number of links at each level
- ❖ Gives full bandwidth between the links
- ❖ Added latency the higher you go



Interconnects

- ❖ Diameter

- ❖ maximum distance between any two processors in the network.
- ❖ The distance between two processors is defined as the shortest path, in terms of links, between them.
- ❖ completely connected network is 1, for star network is 2, for ring is $p/2$ (for p even processors)

- ❖ Connectivity

- ❖ measure of the multiplicity of paths between any two processors (# arcs that must be removed to break the connection).
- ❖ high connectivity is desired since it lowers contention for communication resources.
- ❖ 1 for linear array, 1 for star, 2 for ring, 2 for mesh, 4 for torus
- ❖ technically 1 for traditional fat trees, but there is redundancy in the switch infrastructure

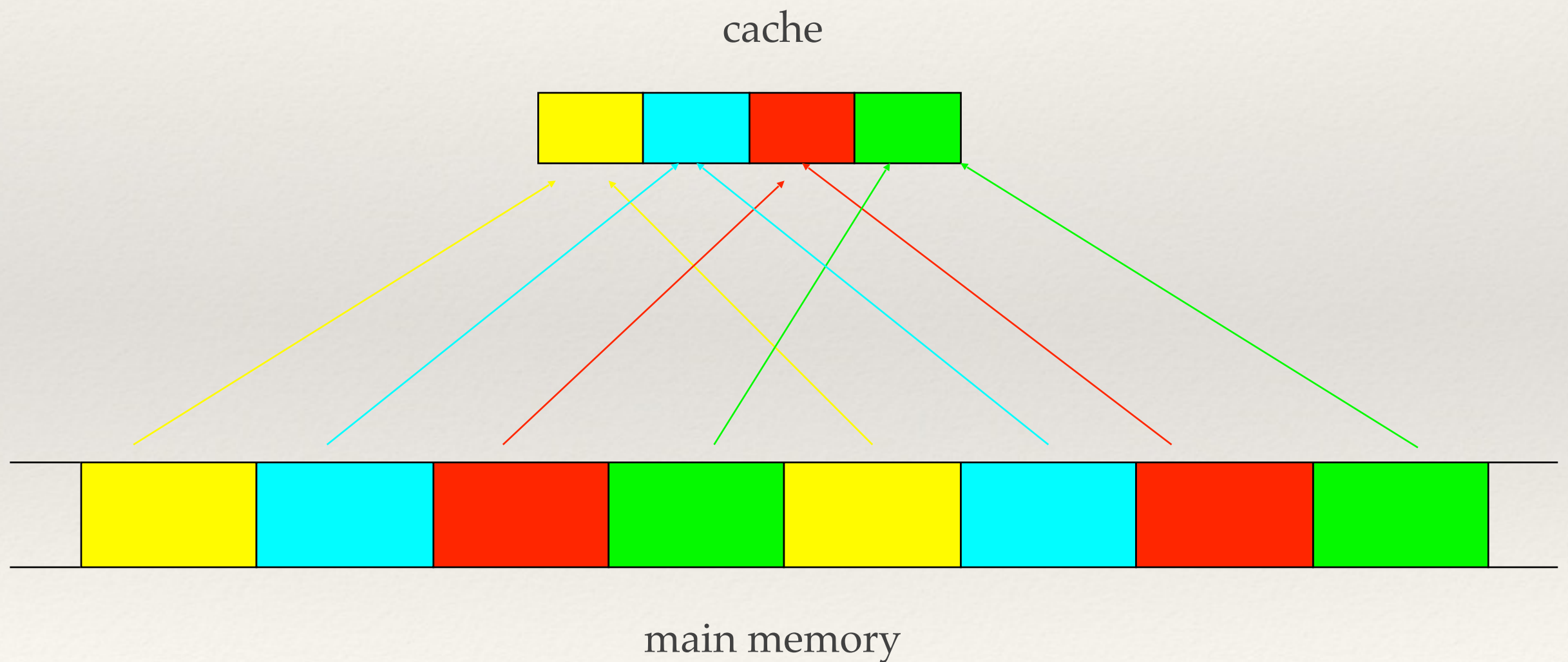
Interconnects

- ❖ Bisection width
 - ❖ Minimum # of communication links that have to be removed to partition the network into two equal halves. Bisection width is
 - ❖ 2 for ring, $\text{sq. root}(p)$ for mesh with p (even) processors, $p/2$ for hypercube, $(p*p)/4$ for completely connected (p even).
- ❖ Channel width
 - ❖ # of physical wires in each communication link
- ❖ Channel rate
 - ❖ peak rate at which a single physical wire link can deliver bits
- ❖ Channel BW
 - ❖ peak rate at which data can be communicated between the ends of a communication link
 - ❖ $= (\text{channel width}) * (\text{channel rate})$
- ❖ Bisection BW
 - ❖ minimum volume of communication found between any 2 halves of the network with equal # of procs
 - ❖ $= (\text{bisection width}) * (\text{channel BW})$

Cache Mapping

- ❖ Because each memory subsystem is smaller than the next-closer level, data must be mapped
- ❖ Types of mapping
 - ❖ Direct
 - ❖ Set associative
 - ❖ Fully associative

Direct Mapped Caches

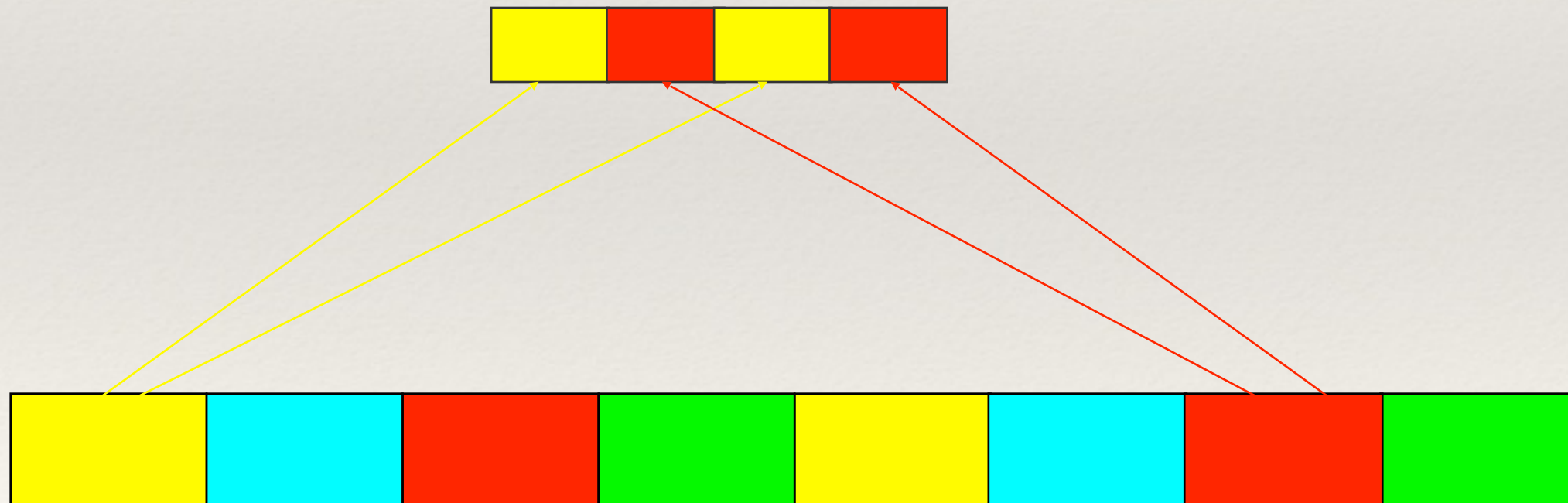


Direct Mapped Caches

- ❖ If the cache size is N_c and it is divided into k lines, then each cache line is N_c/k in size
- ❖ If the main memory size is N_m , memory is then divided into $N_m / (N_c/k)$ blocks that are mapped into each of the k cache lines
- ❖ Means that each cache line is associated with particular regions of memory

Set Associative Caches

2-way set-associative cache

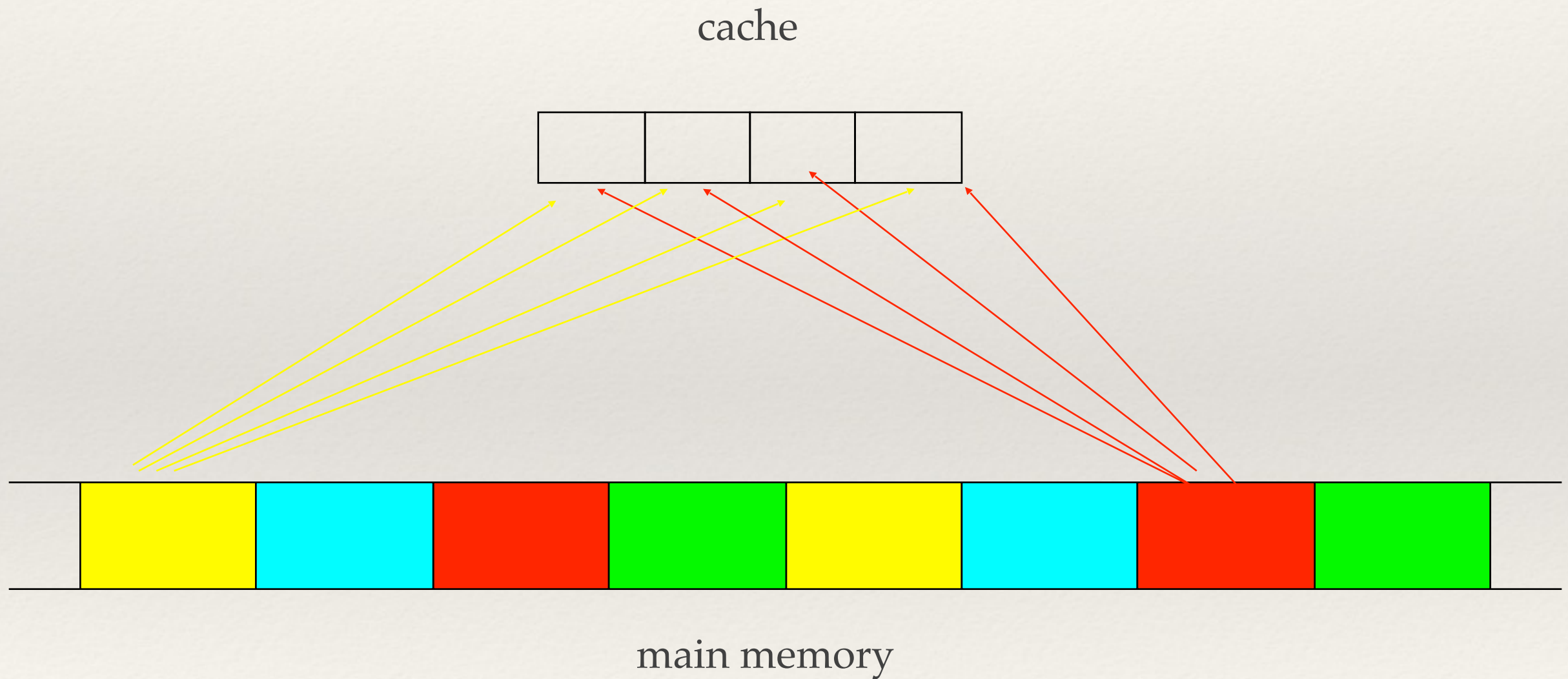


main memory

Set Associative Caches

- ❖ Direct-mapped caches are 1-way set-associative caches
- ❖ For a k -way set-associative cache, each memory region can be associated with k cache lines

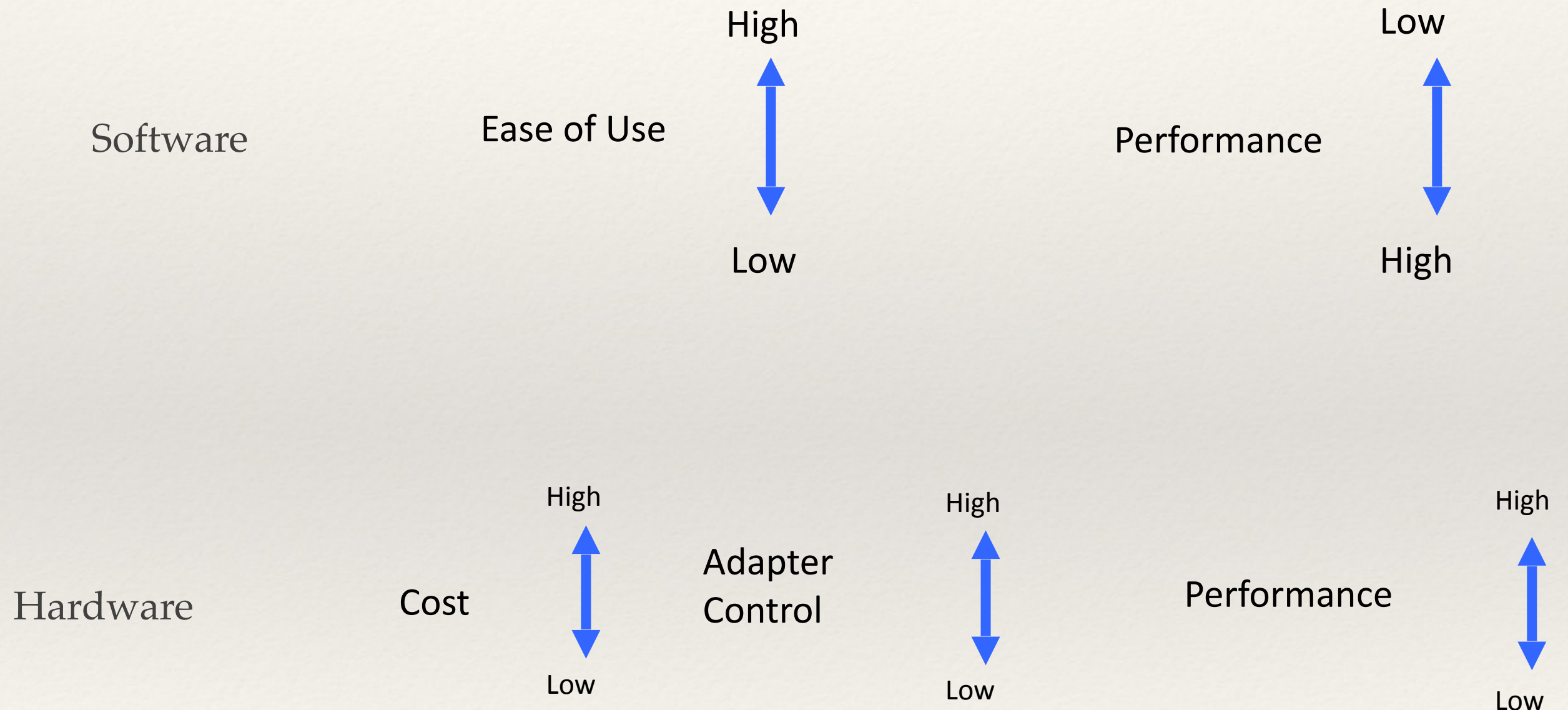
Fully Associative Caches



Fully Associative Caches

- ❖ Ideal situation
- ❖ Any memory location can be associated with any cache line
- ❖ Cost prohibitive

Performance Summary



Summary

- ❖ Why so much parallel talk?
 - ❖ Every computer is a parallel computer now
 - ❖ Good serial computing skills are central to good parallel computing
 - ❖ Cluster and MPP nodes appear largely like desktops and laptops
 - ❖ Processing units: CPUs, FPUs, GPUs
 - ❖ Memory hierarchies: Registers, Caches, Main memory
 - ❖ Internal Interconnect: Buses and Switch-based networks
 - ❖ Clusters and MPPs built via fancy connections.