Introduction to High Performance Computers

Richard Gerber

NERSC User Services

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

Why Do You Care About Architecture?

- **To use HPC systems well, you need to understand the basics and conceptual design**
  - Otherwise, too many things are mysterious
- **Programming for HPC systems is hard**
  - To get your code to work properly
  - To make it run efficiently (performance)
- **You want to efficiently configure the way your job runs**
- **The technology is just cool!**

Outline

- Terminology
- 5 main parts of an HPC system
- CPUs
- Nodes
- Interconnect
- Data Storage
- HPC Systems

Definitions & Terminology

- **HPC**
  - High Performance Computing
  - Scientific computing at scale
- **CPU**
  - Central Processing Unit
  - Now ambiguous terminology
  - Generic for “some unit that computes”
  - Context-sensitive meaning
- **Core**
  - Hardware unit that performs arithmetic operations
  - A CPU may have more than one core
- **Die**
  - An integrated circuit manufactured as a unit
  - Many cores may be included on a die
- **Socket**
  - A physical package that connects to a computer board
  - A socket package may be composed of multiple dies

Definitions & Terminology

- **FLOP: Floating Point Operation**
  - e.g., `a+b`, `a*b+c`
  - FLOPs/sec is a common performance metric
- **SMP**
  - Defn: Symmetric Multiprocessing
  - Common usage: Collection of processors that have (approx equal) access to a shared pool of memory in a single memory address space
- **FORTRAN**
  - Programming language popular with scientists, esp. in HPC
- **MPP**
  - Massively parallel processing
- **Interconnect**
  - A high-performance data network that connects nodes to each other and possibly other devices

Definitions & Terminology

- **Memory**
  - Volatile storage of data or computer instructions
- **Bandwidth**
  - The rate at which data is transferred between destinations (typically GB/s)
- **Latency**
  - The time needed to initialize a data transfer (ranges from 10^{-9} to 10^{-6} secs or more)
- **SRAM: Static RAM (random access memory)**
  - Fast
  - 6 transistors store a bit
    - Expensive
    - Limits storage density
- **DRAM: Dynamic RAM (random access memory)**
  - Slower
  - 1 transistor, 1 capacitor stores a bit
    - Higher density, cheaper
  - Capacitor voltage needs to be refreshed
    - Additional power

What are the main parts of a computer?

Boy Scouts of America
Offer a Computers Merit Badge

Merit Badge Requirements

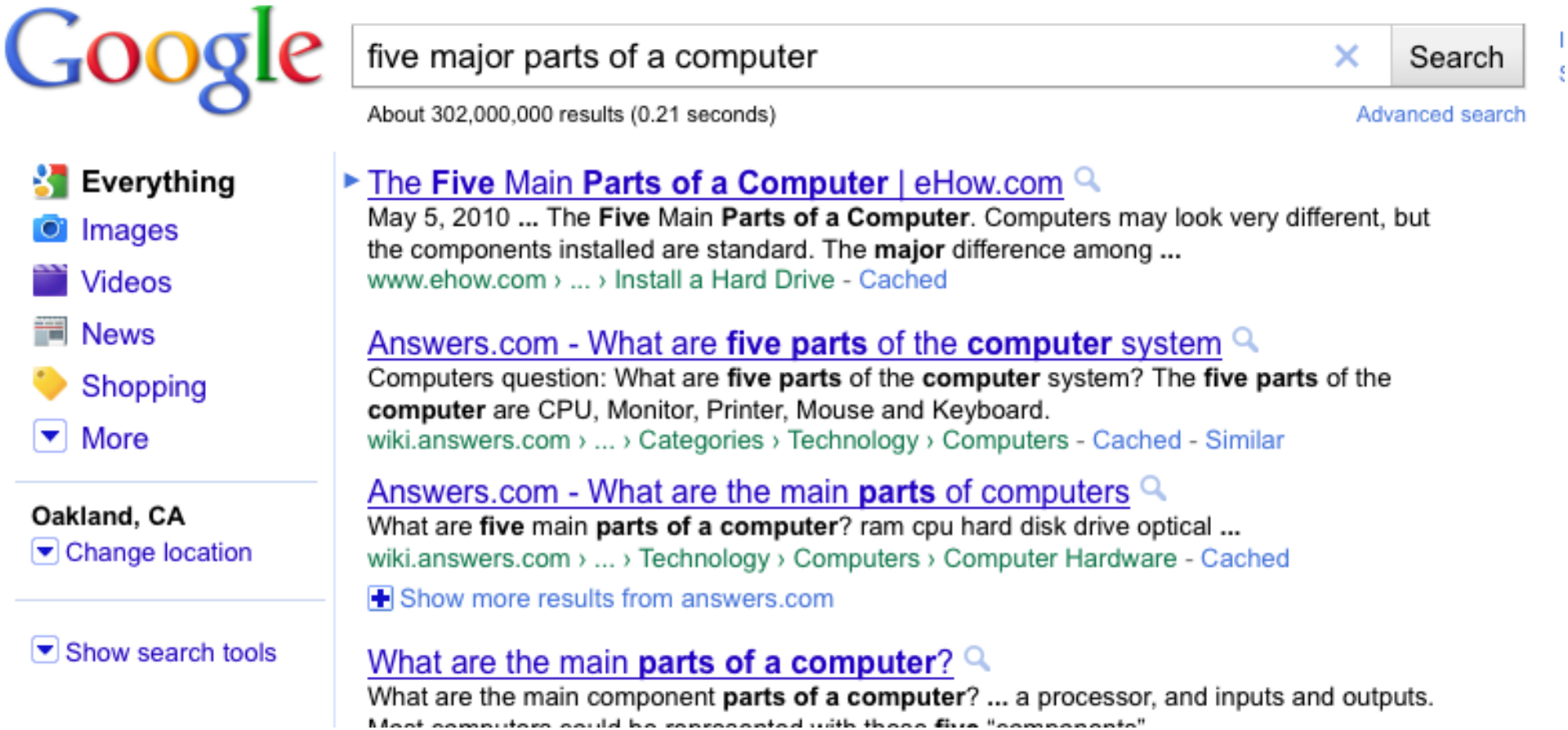
…

4. Explain the following to your counselor:

a. The five major parts of a computer.

…

What are the “5 major parts”?



Five Major Parts

eHow.com	Answers.com	Fluther.com	Yahoo!	Wikipedia
CPU	CPU	CPU	CPU	Motherboard
RAM	Monitor	RAM	RAM	Power Supply
Hard Drive	Printer	Storage	Power Supply	Removable Media
Video Card	Mouse	Keyboard/ Mouse	Video Card	Secondary Storage
		Monitor		
Motherboard	Keyboard	Motherboard	Motherboard	Sound Card
		Case / Power Supply		IO Peripherals

It Depends on Your Perspective

- What is a computer?
  - It depends what you are interested in.
    - CPU, memory, video card, motherboard, ...
    - Monitor, mouse, keyboard, speakers, camera, …
- We'll take the perspective of an application programmer or a scientist running a code on an HPC system
- What features of an HPC system are important for you to know about?

5 Major Parts of an HPC System

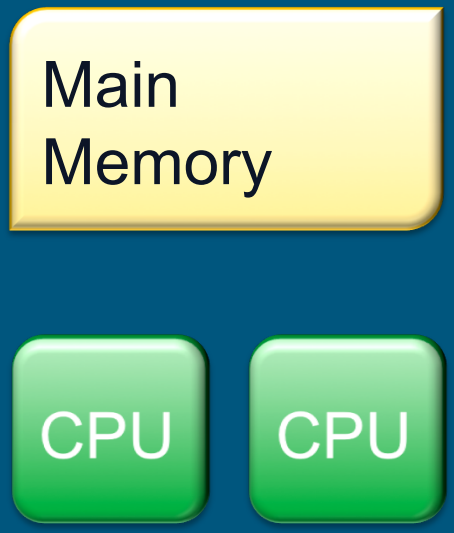
1. CPUs
2. Memory (volatile)
3. Nodes
4. Inter-node network
5. Non-volatile storage (disks, tape)

Lawrence Berkeley
National Laboratory

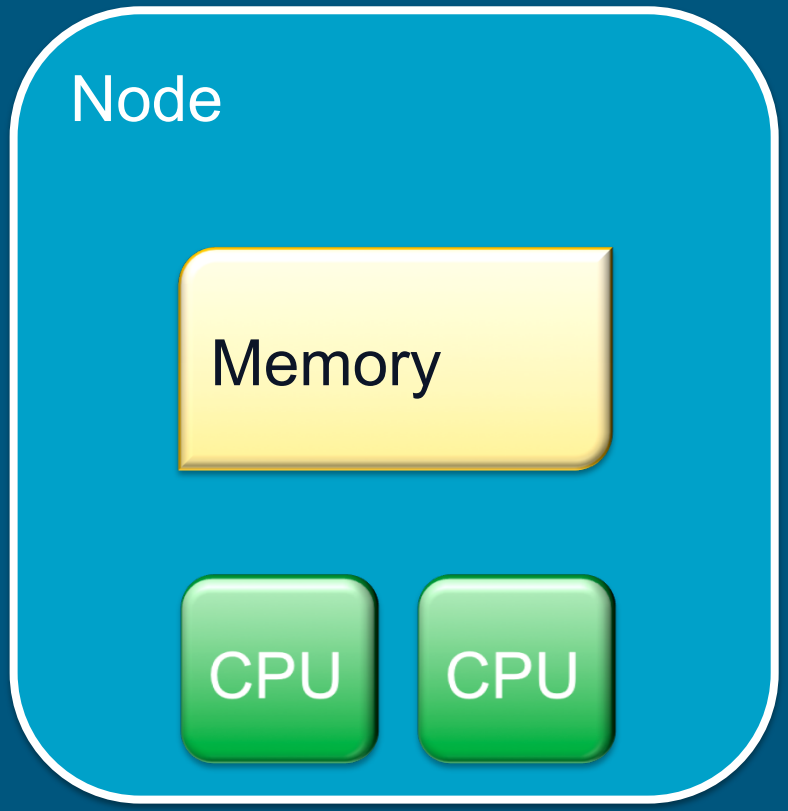
A distributed-memory HPC system

A distributed-memory HPC system

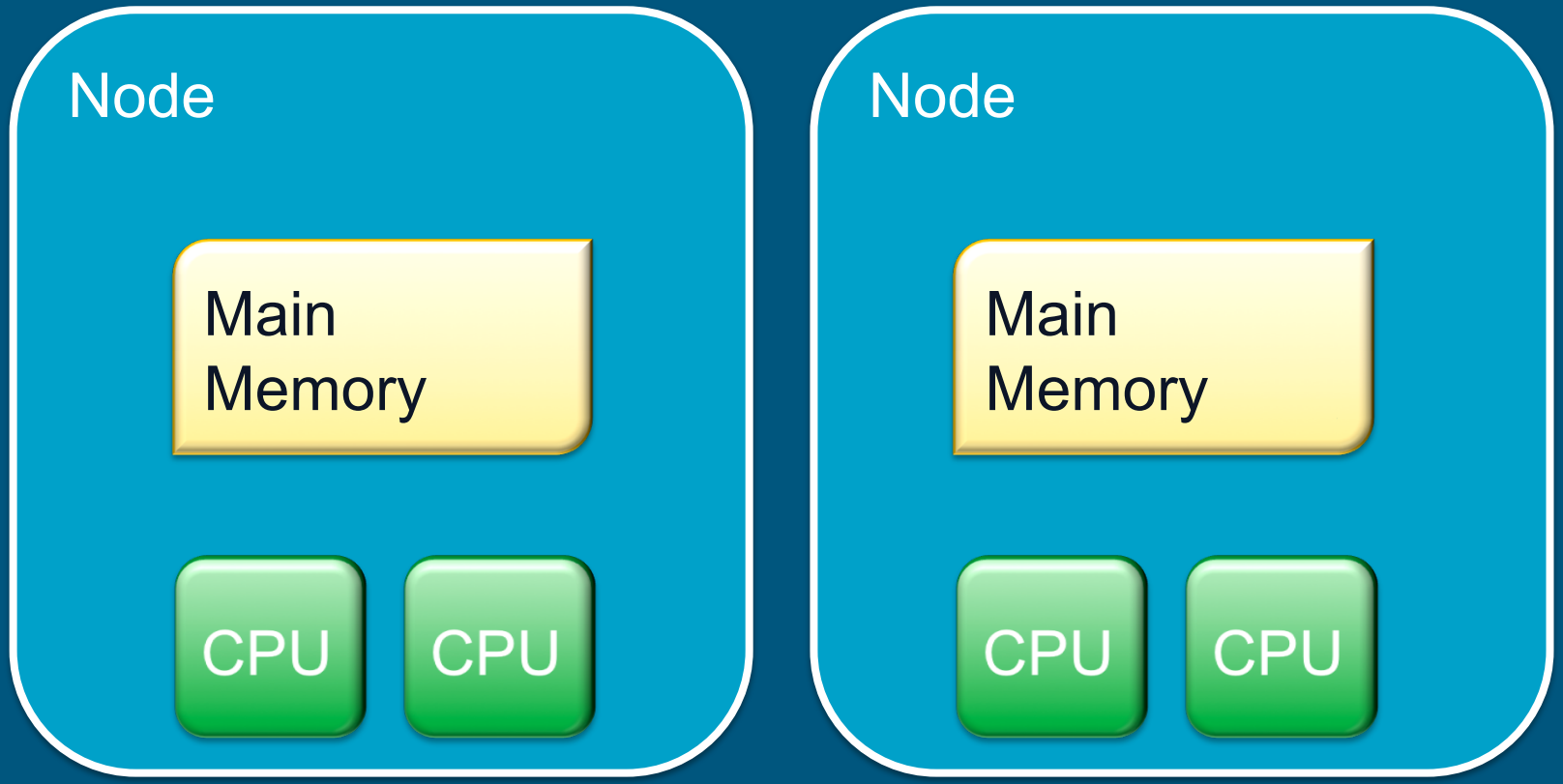
A distributed-memory HPC system



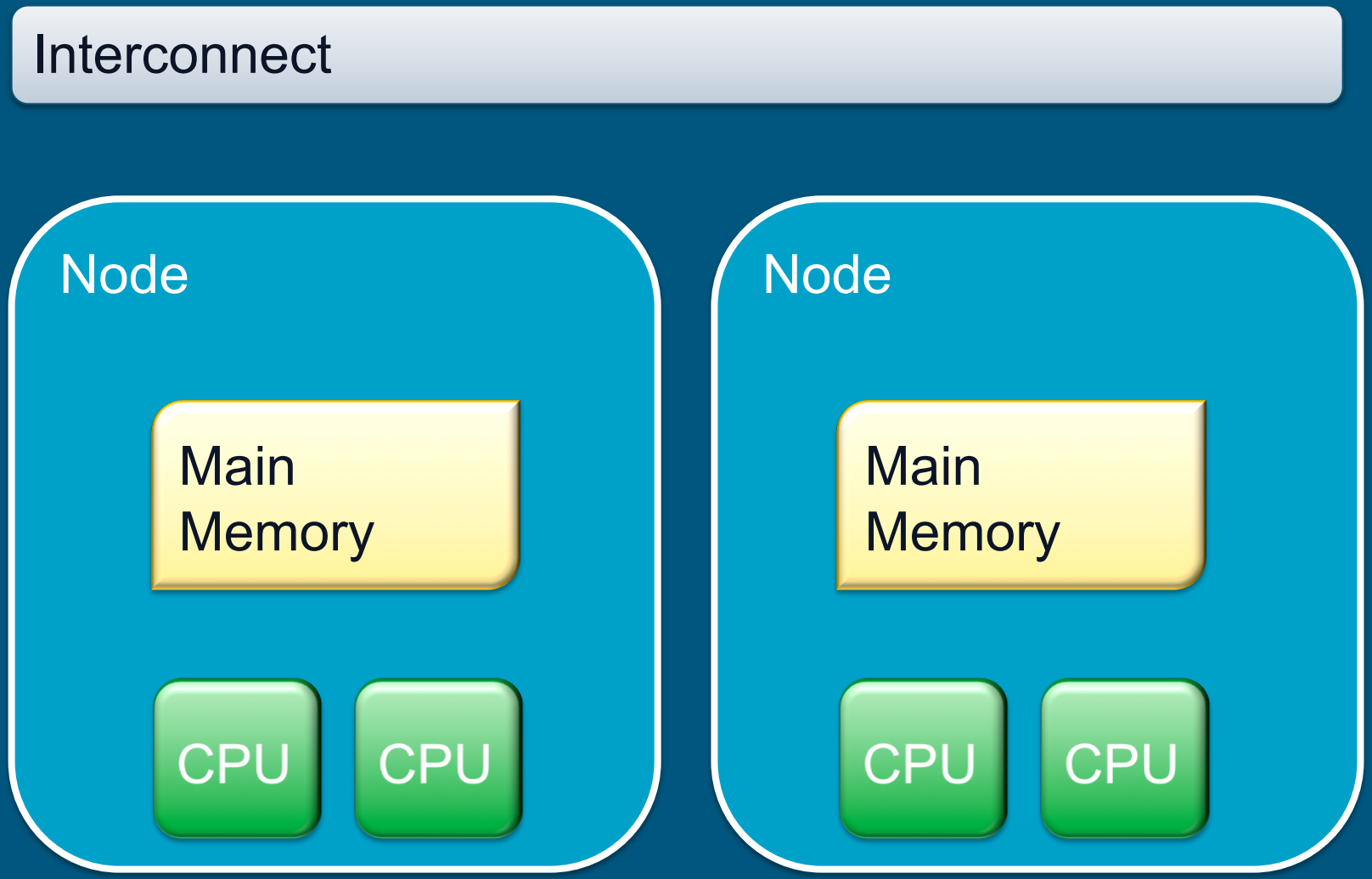
A distributed-memory HPC system



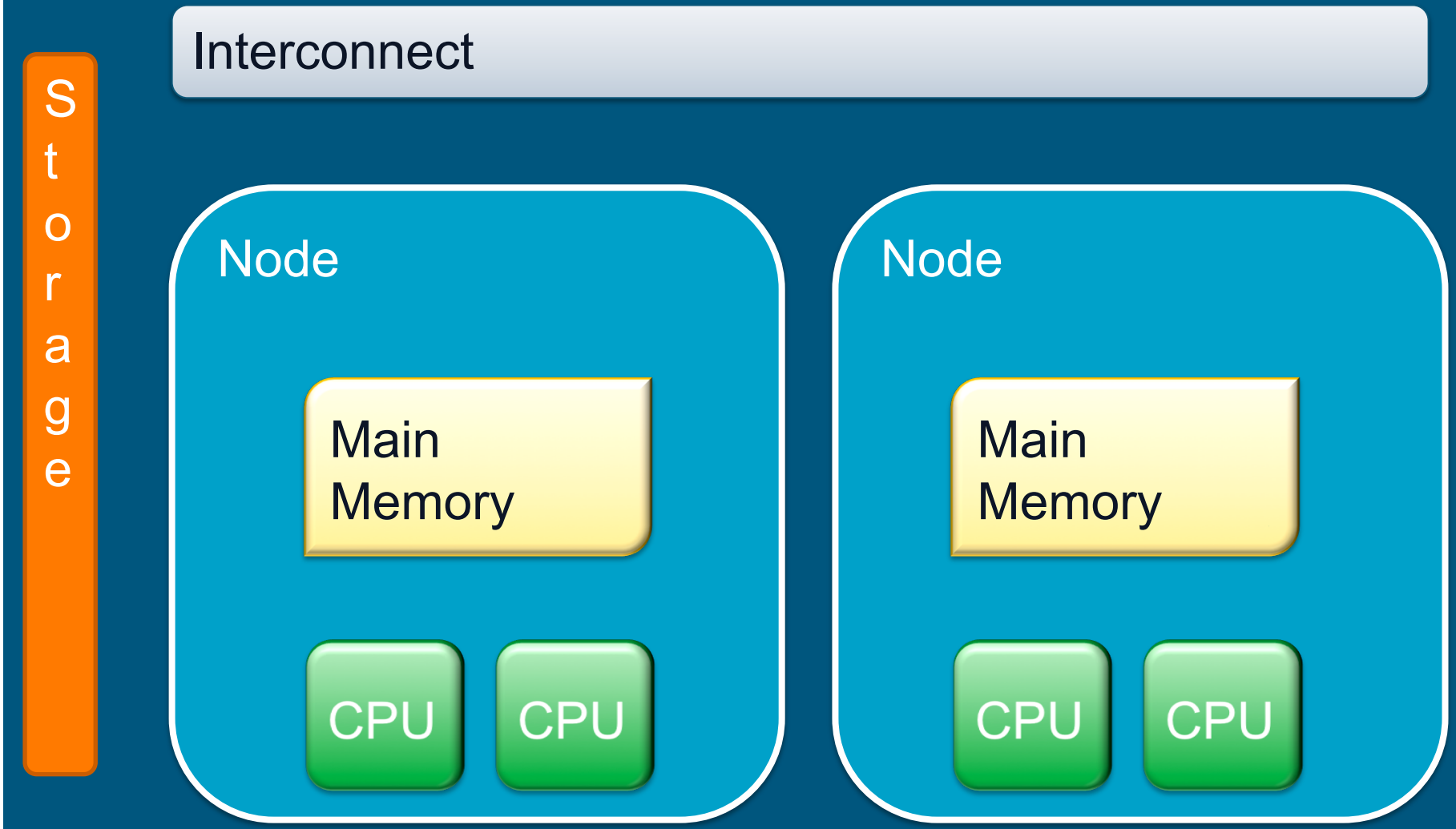
A distributed-memory HPC system



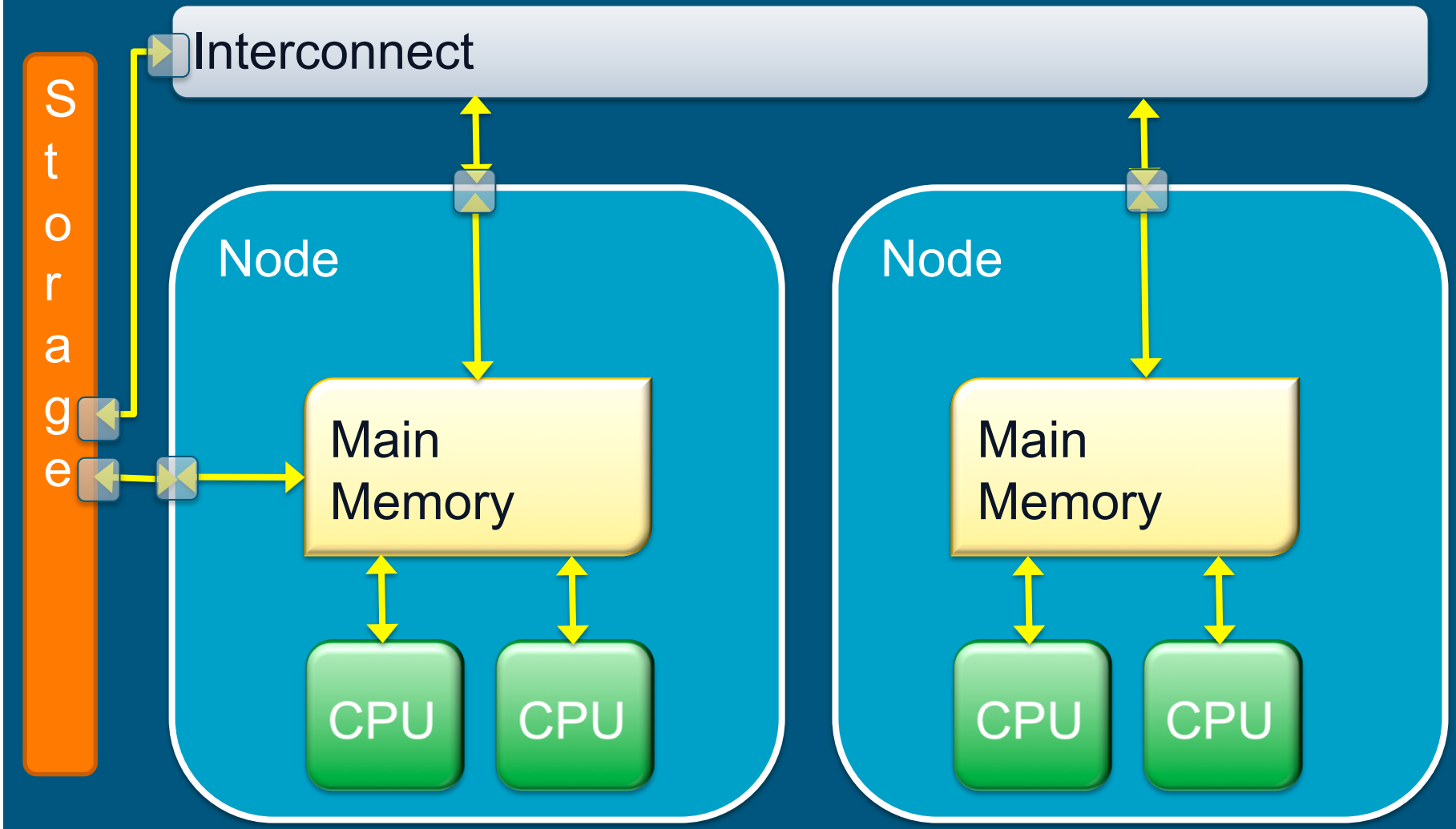
A distributed-memory HPC system



A distributed-memory HPC system



A distributed-memory HPC system

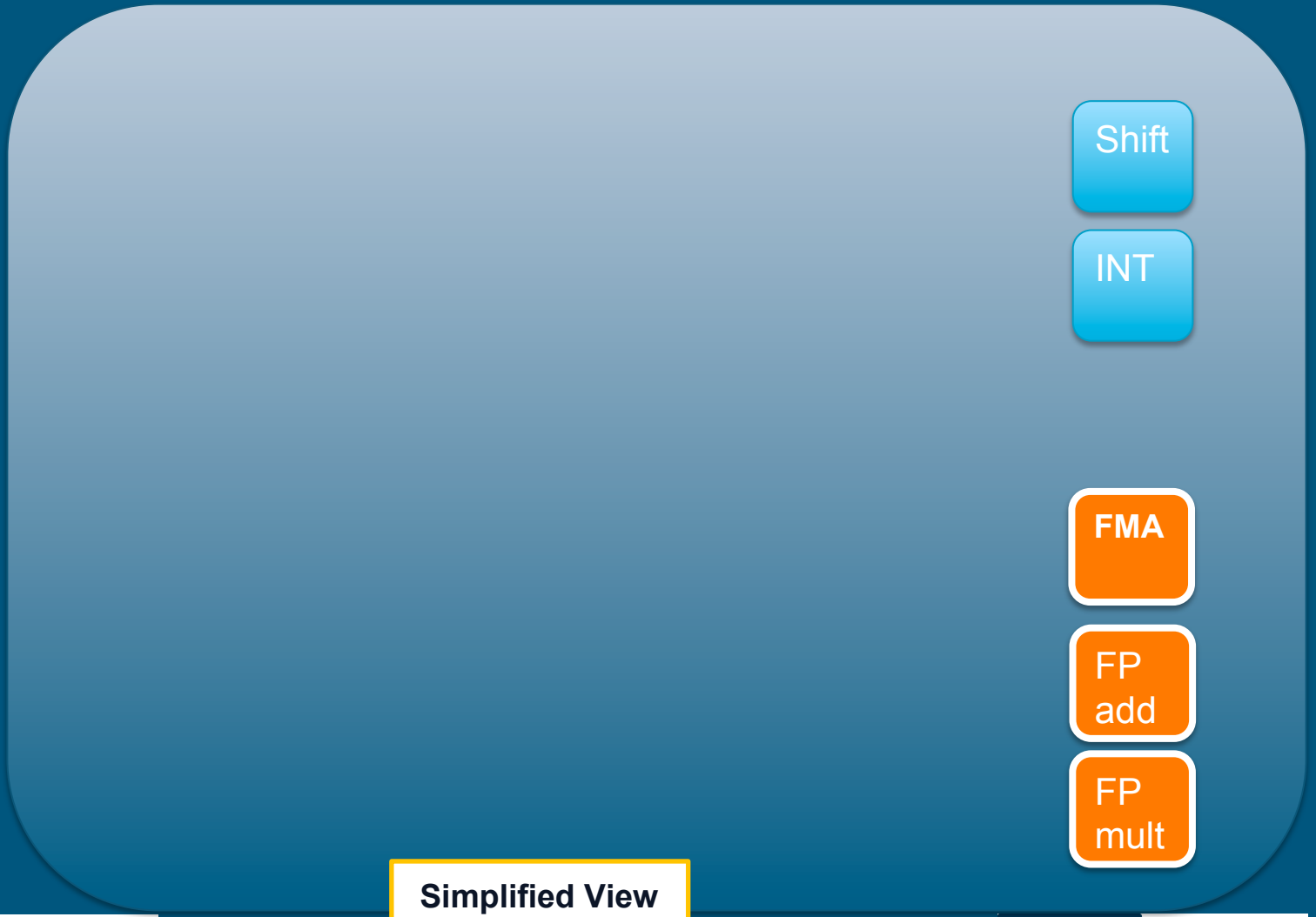


CPUs

Stored Program Computers

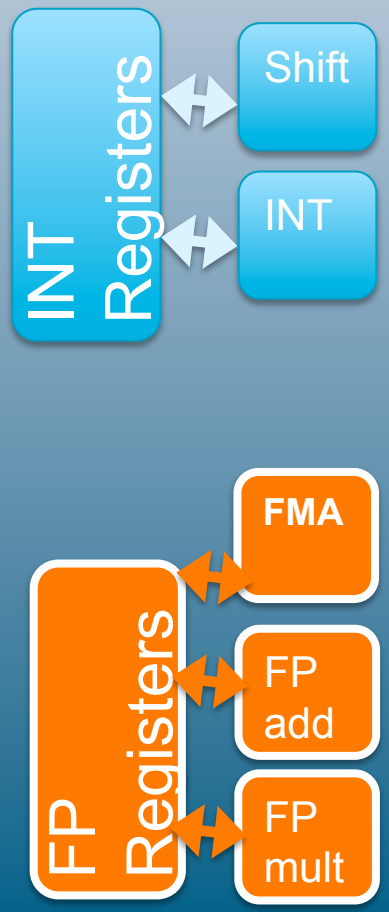
- **Modern computers are "stored program computers"**
  - Conceived by Turing in 1936
  - Implemented in 1949 (EDVAC)
- **Instructions are stored as data in memory**
  - Read and executed by control units
- **Arithmetic and logic**
  - Performed by functional units separate from instruction control units

CPU (1 core)



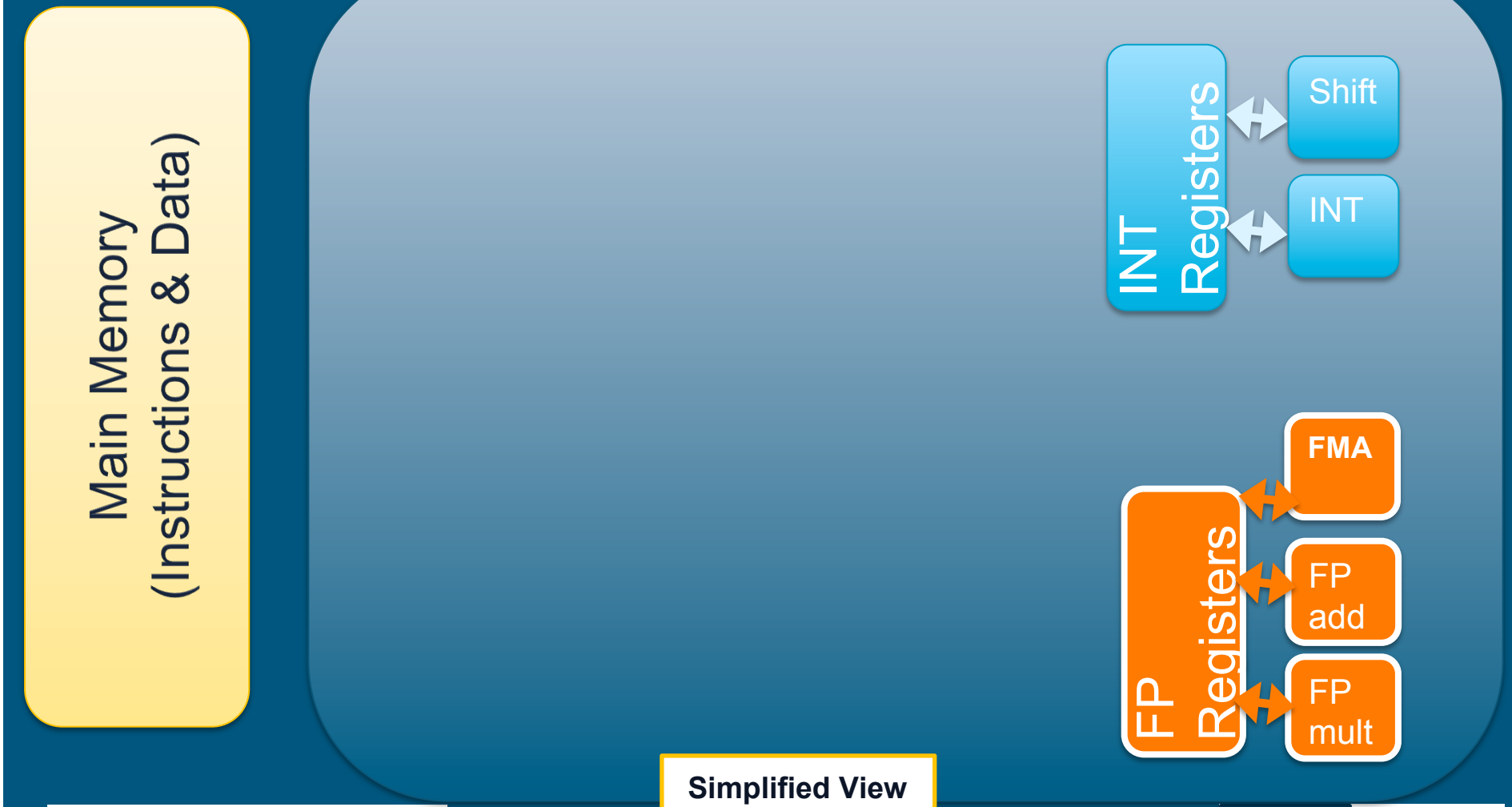
Simplified View

CPU (1 core)



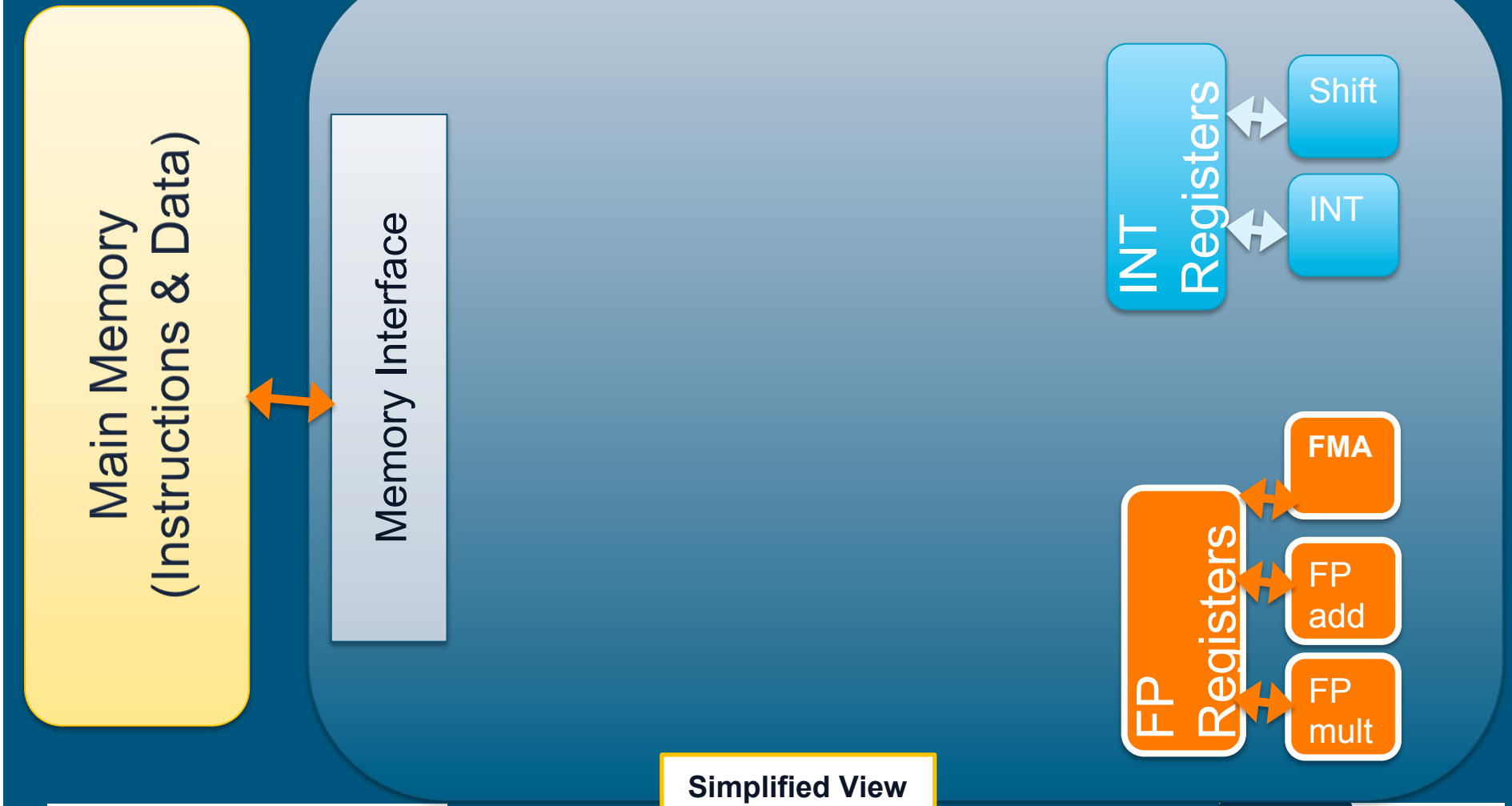
Simplified View

CPU (1 core)



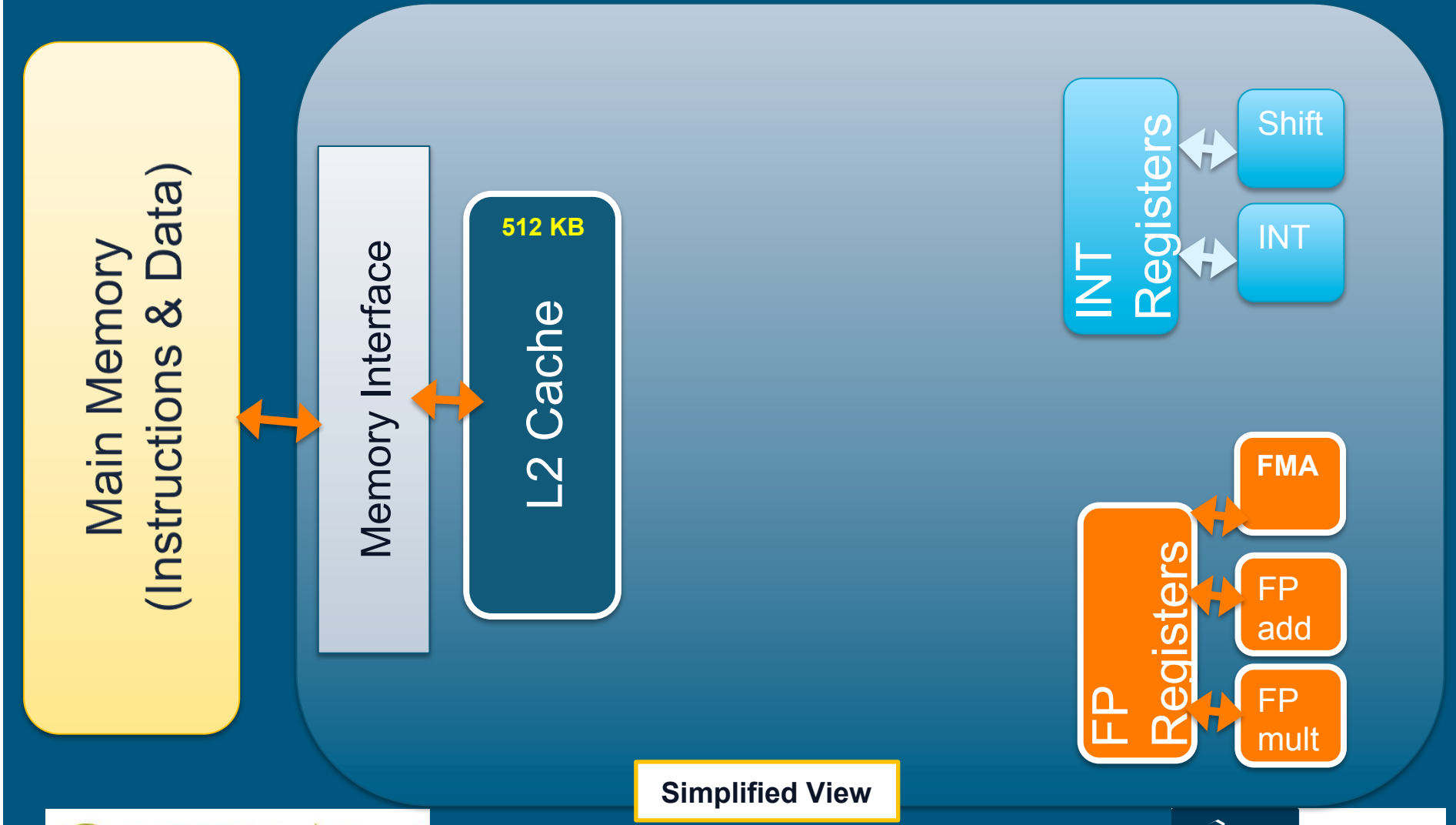
Simplified View

CPU (1 core)



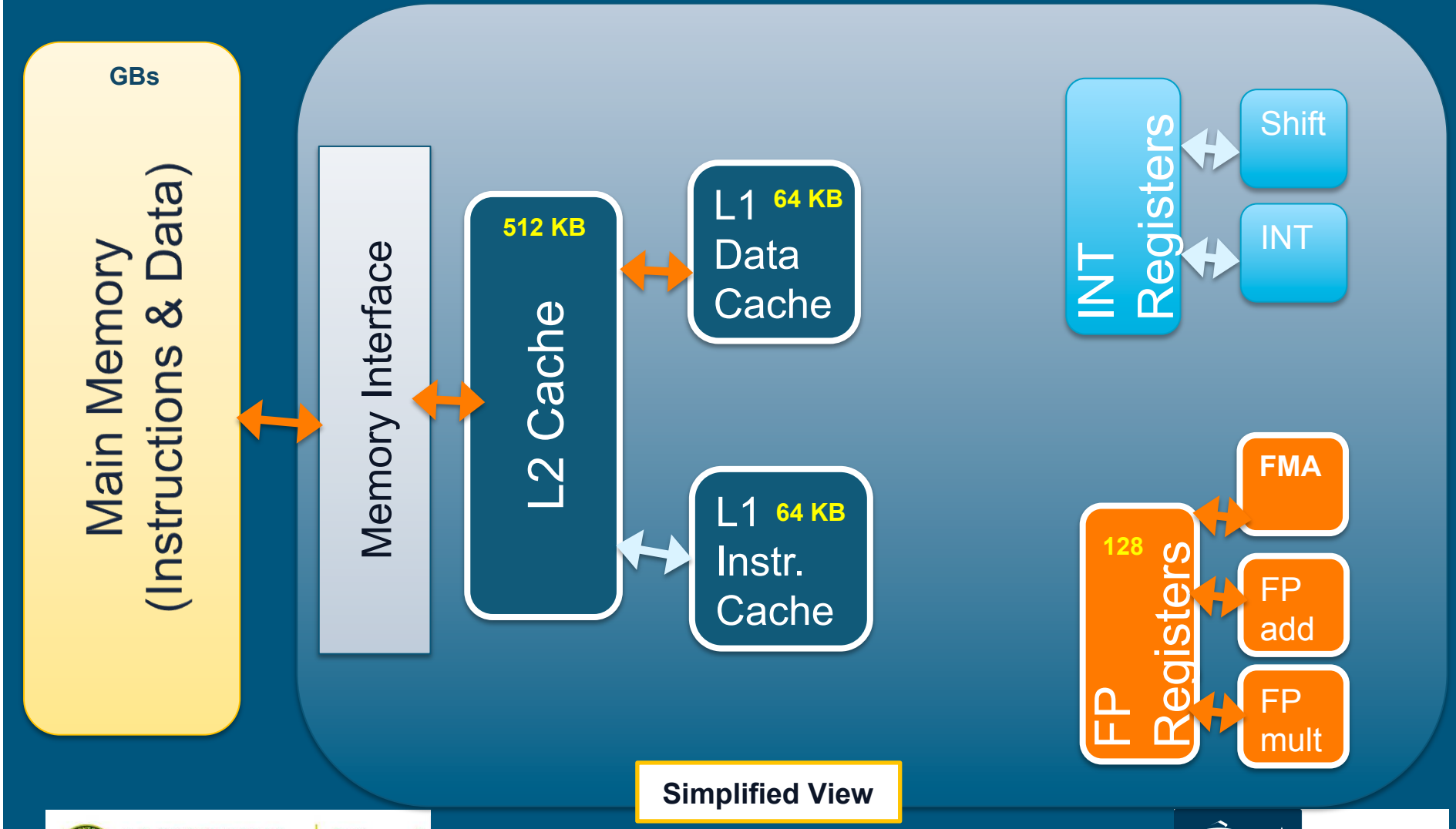
Simplified View

CPU (1 core)

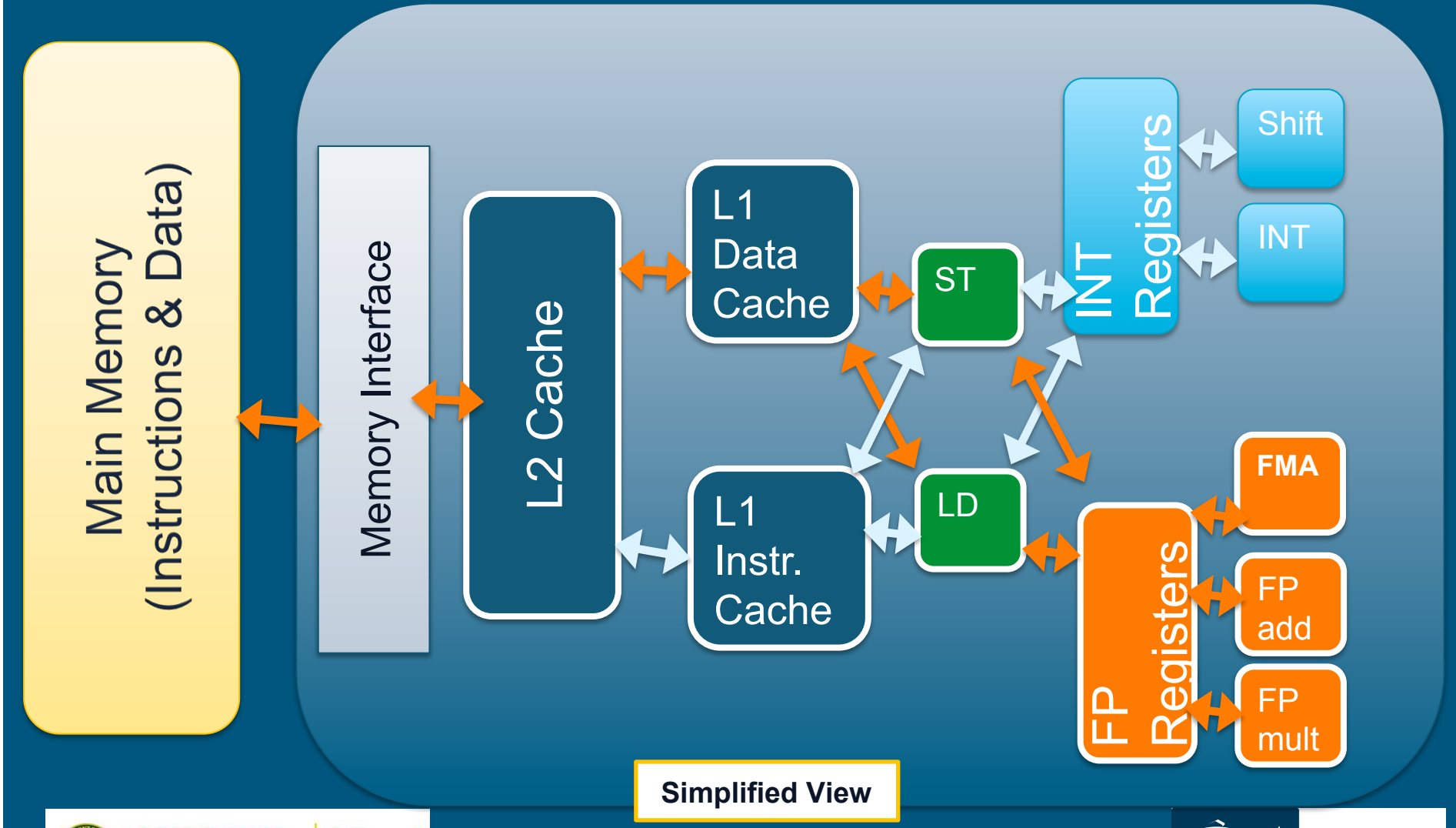


Simplified View

CPU (1 core)


GBs
Main Memory
(Instructions & Data)
Memory Interface
512 KB
L2 Cache
L1 64 KB
Data
Cache
L1 64 KB
Instr.
Cache
INT
Registers
Shift
INT
FP
Registers
128
FMA
FP
add
FP
mult
Simplified View

CPU (1 core)


Main Memory
(Instructions & Data)
Memory Interface
L2 Cache
L1
Data
Cache
L1
Instr.
Cache
ST
LD
INT
Registers
Shift
INT
FP
Registers
FMA
FP
add
FP
mult
Simplified View

Additional Functional Units

- **There's a lot more on the CPU than shown previously, e.g.**
  - L3 cache (~10 MB)
  - SQRT/Divide/Trig FP unit
  - "TLB" to cache memory addresses
  - Instruction decode
  - …

On-Chip Performance Enhancements

- **Chip designers have added lots of complexity to increase performance**
- **Instruction Parallelism**
  - Pipelined functional units (e.g. FPU)
  - Superscalar processors
- **Data Parallelism**
  - SIMD
- **Cache lines**
  - Data brought into cache in contiguous chunks
- **Out of order & speculative execution**

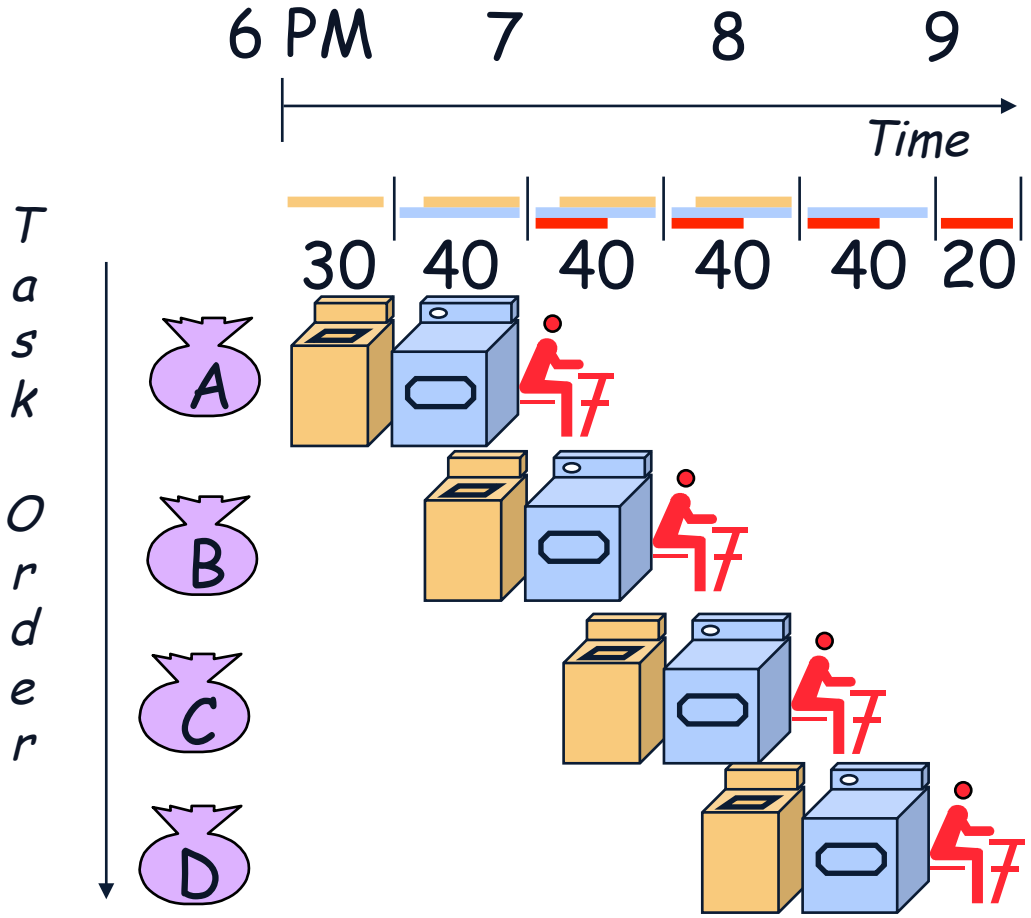
Example: Pipeline Stages in an FPU

- Separate mantissa / exponent
- Multiply mantissas
- Add exponents
- Normalize result
- Insert sign

Pipelining

Dave Patterson's Laundry example: 4 people doing laundry

wash (30 min) + dry (40 min) + fold (20 min) = 90 min



- In this example:
  - Sequential execution takes 4 * 90min = 6 hours
  - Pipelined execution takes 30+4*40+20 = 3.5 hours
- Bandwidth = loads/hour
- BW = 4/6 l/h w/o pipelining
- BW = 4/3.5 l/h w pipelining
- BW <= 1.5 l/h w pipelining, more total loads
- Pipelining helps bandwidth but not latency (90 min)
- Bandwidth limited by slowest pipeline stage
- Potential speedup = Number pipe stages

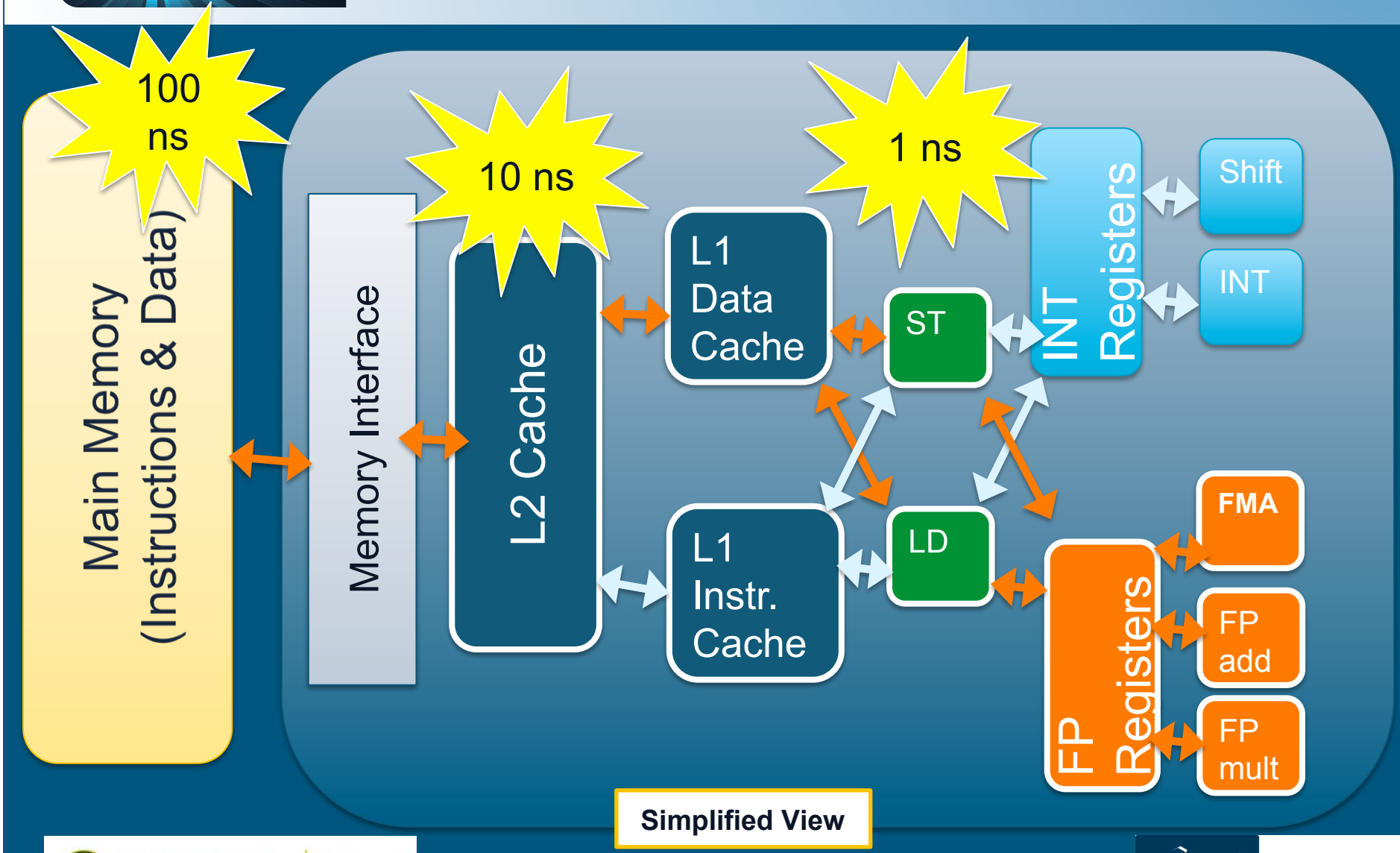
Superscalar Processors

- **Superscalar processors can execute more than one instruction per clock cycle**
- **Example**
  - 2 FMAs
  - 2 integer ops
  - Multiple LOAD & STORE

SIMD

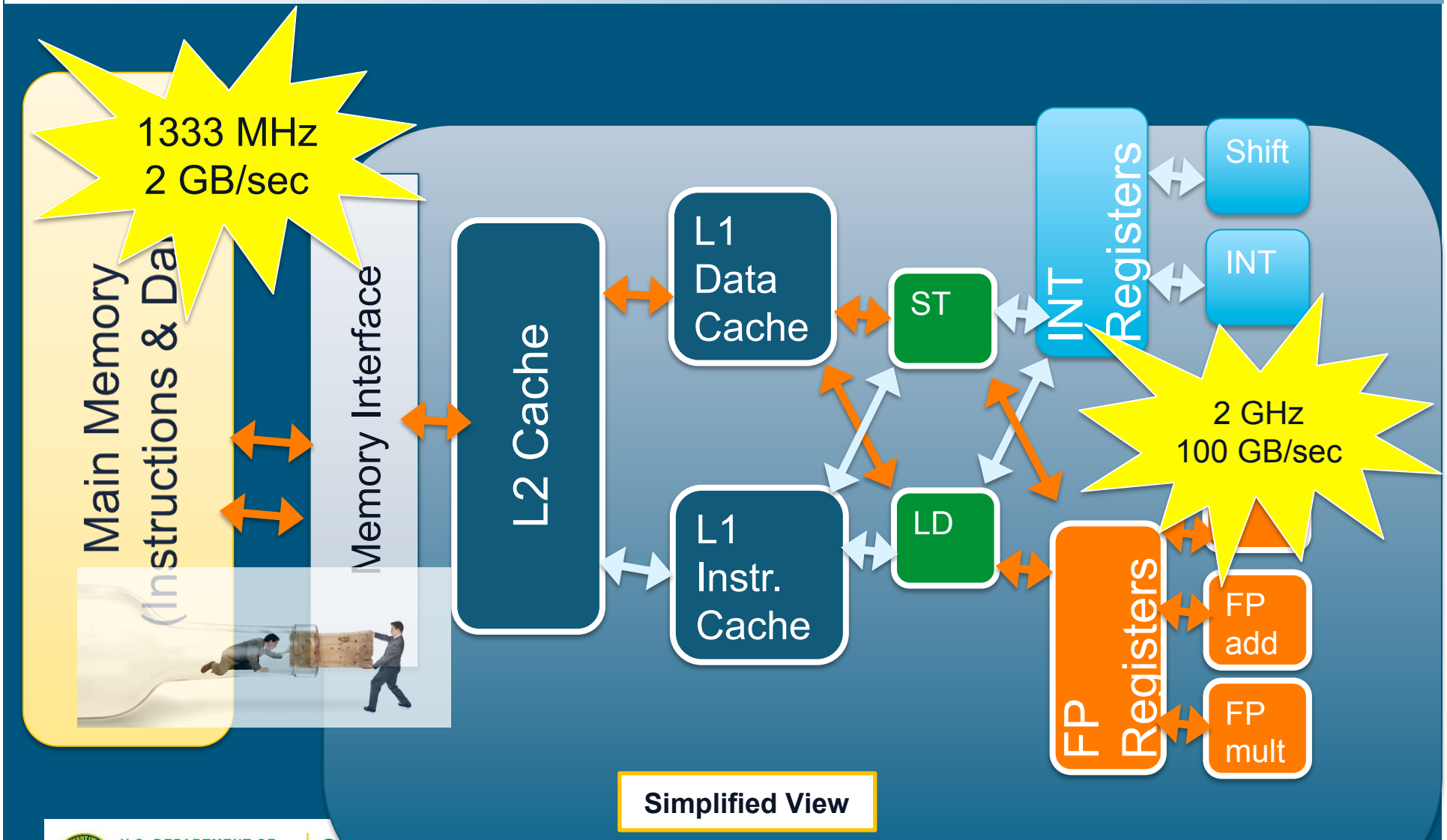
- Special registers can hold multiple words of data
- A single instruction (e.g. floating point multiply) is applied to all the data at once
- "SSE[2-4]" : Streaming SIMD Extension instruction set for x86
- aka "Vectorization"

Latencies



Simplified View

Memory Bottleneck



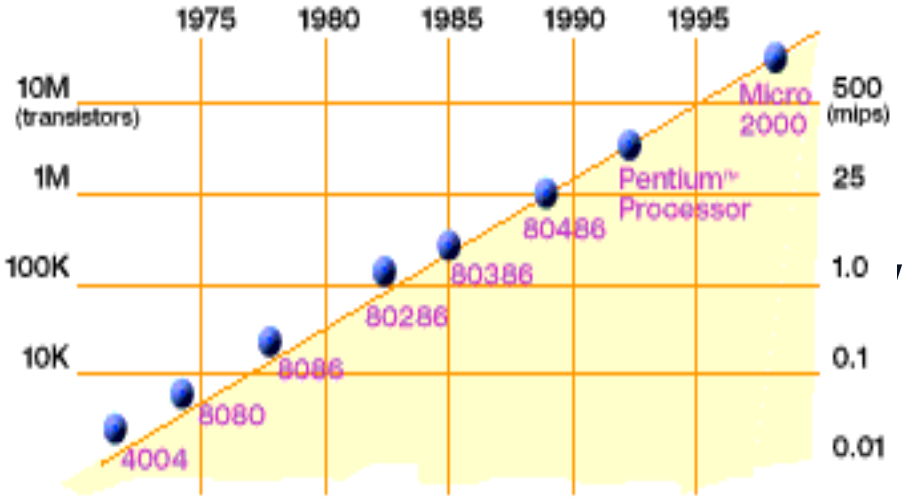
Memory Bottleneck

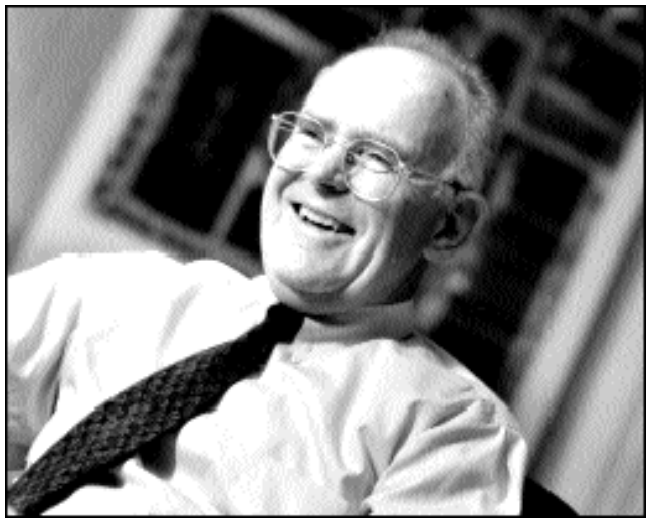
- **Modern HPC CPUs can achieve ~5 Gflop/sec per compute core**
  - 2 8-byte data words per operation
  - 80 GB/sec of data needed to keep CPU busy
- **Memory interfaces provide a few GB/ sec per core from main memory**
- **Memory latency – the startup time to begin fetching data from memory – is even worse**

Multicore Processors

- There are no more single-core CPUs (processors) as just described
- All CPUs (processors) now consist of multiple compute “cores” on a single “chip” or “die” with possibly multiple chips per “socket” (the unit that plugs into the motherboard)
- Increased complexity
- The trend is for ever-more cores per die, but this is for good reasons

Moore's Law





2X transistors/Chip Every 1.5 years

Called "Moore's Law"

Microprocessors have become smaller, denser, and more powerful.

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

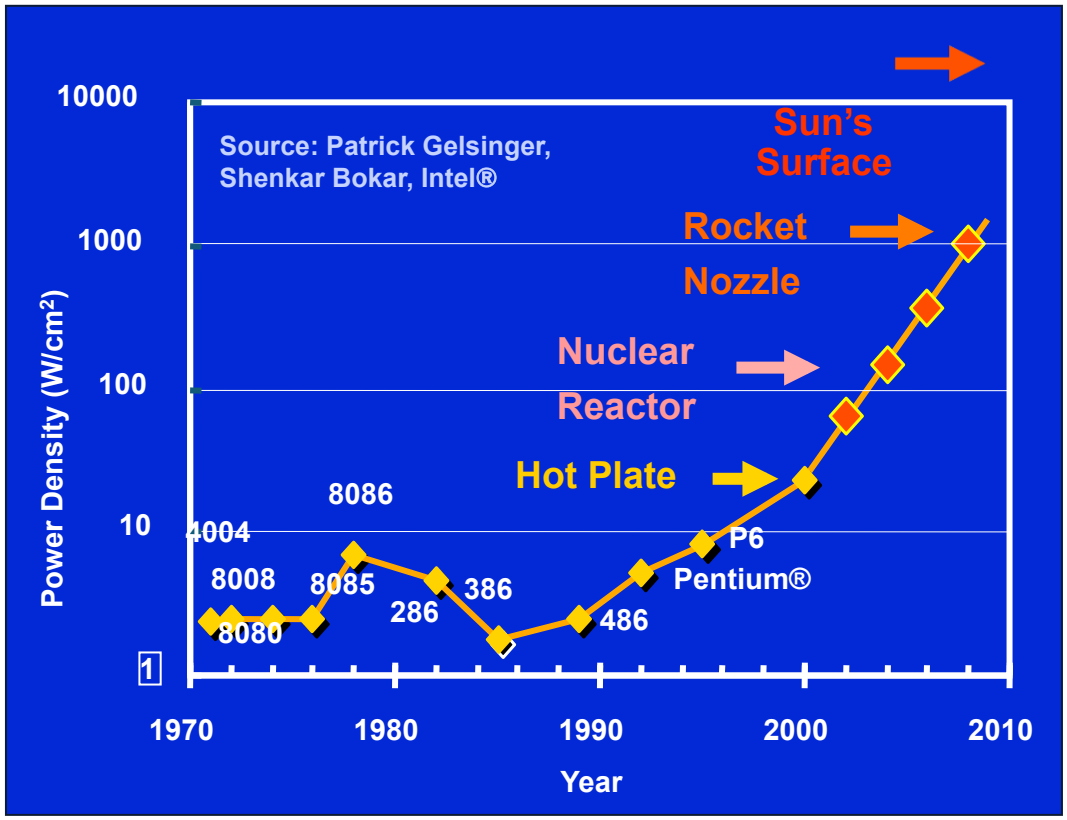
Slide source: Jack Dongarra

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB Lawrence Berkeley National Laboratory

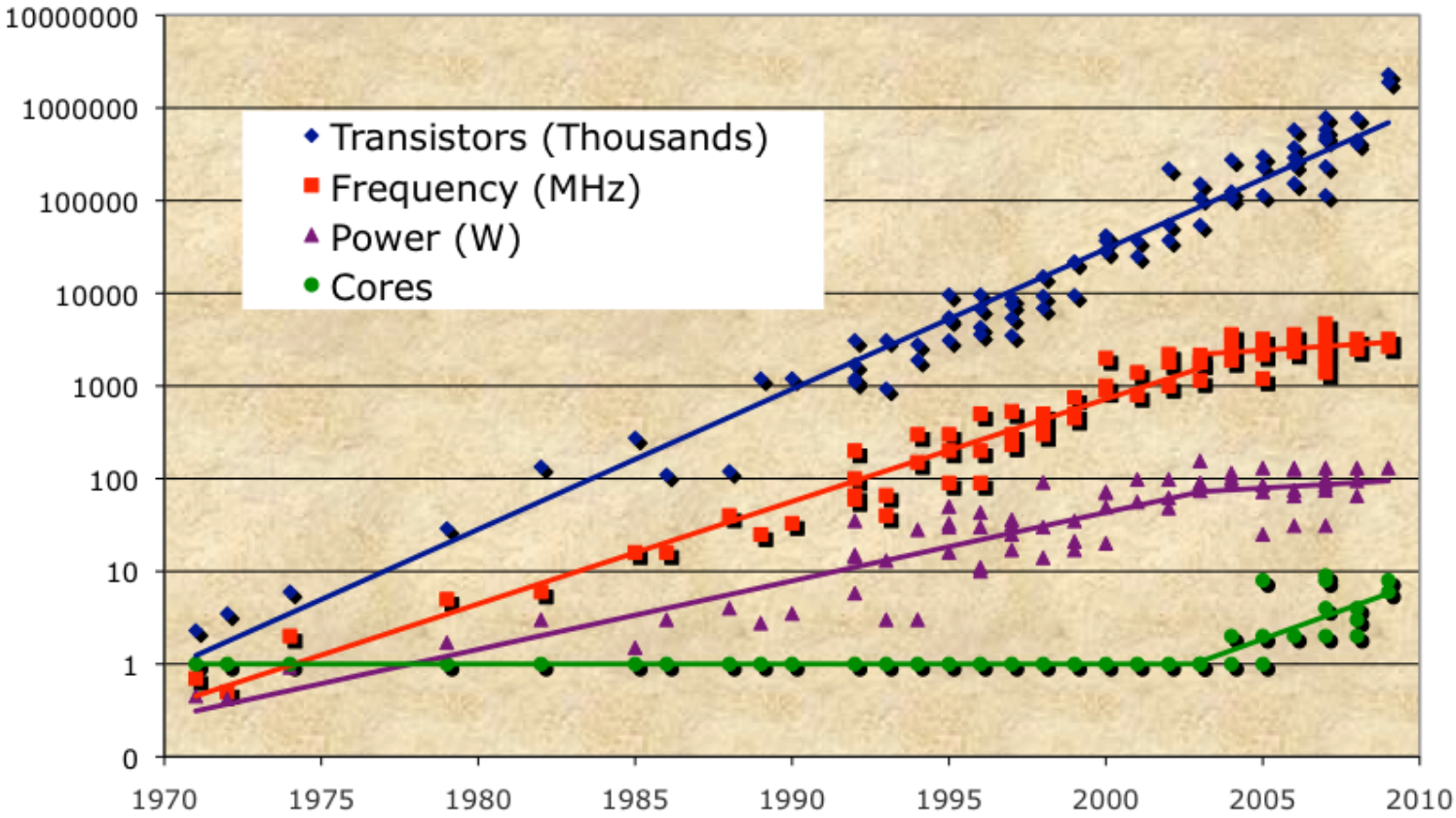
Power Density Limits Serial Performance

- Concurrent systems are more power efficient
  - Dynamic power is proportional to V^2fC
  - Increasing frequency (f) also increases supply voltage (V) → cubic effect
  - Increasing cores increases capacitance (C) but only linearly
  - Save power by lowering clock speed



- High performance serial processors waste power
  - Speculation, dynamic dependence checking, etc. burn power
  - Implicit parallelism discovery
- More transistors, but not faster serial processors

Revolution in Processors

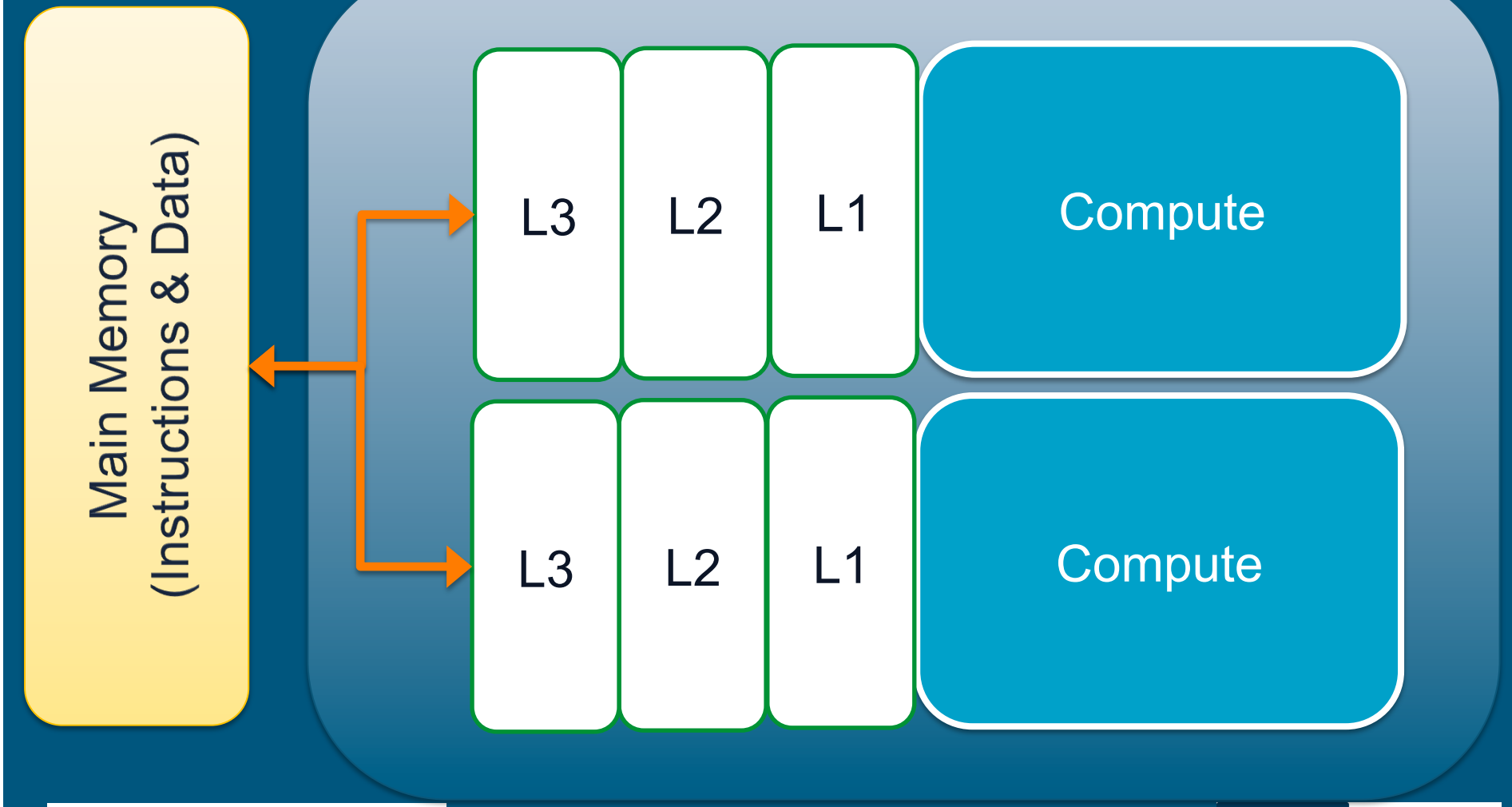


- Chip density is continuing increase ~2x every 2 years
- Clock speed is not
- Number of processor cores may double instead
- Power is under control, no longer growing

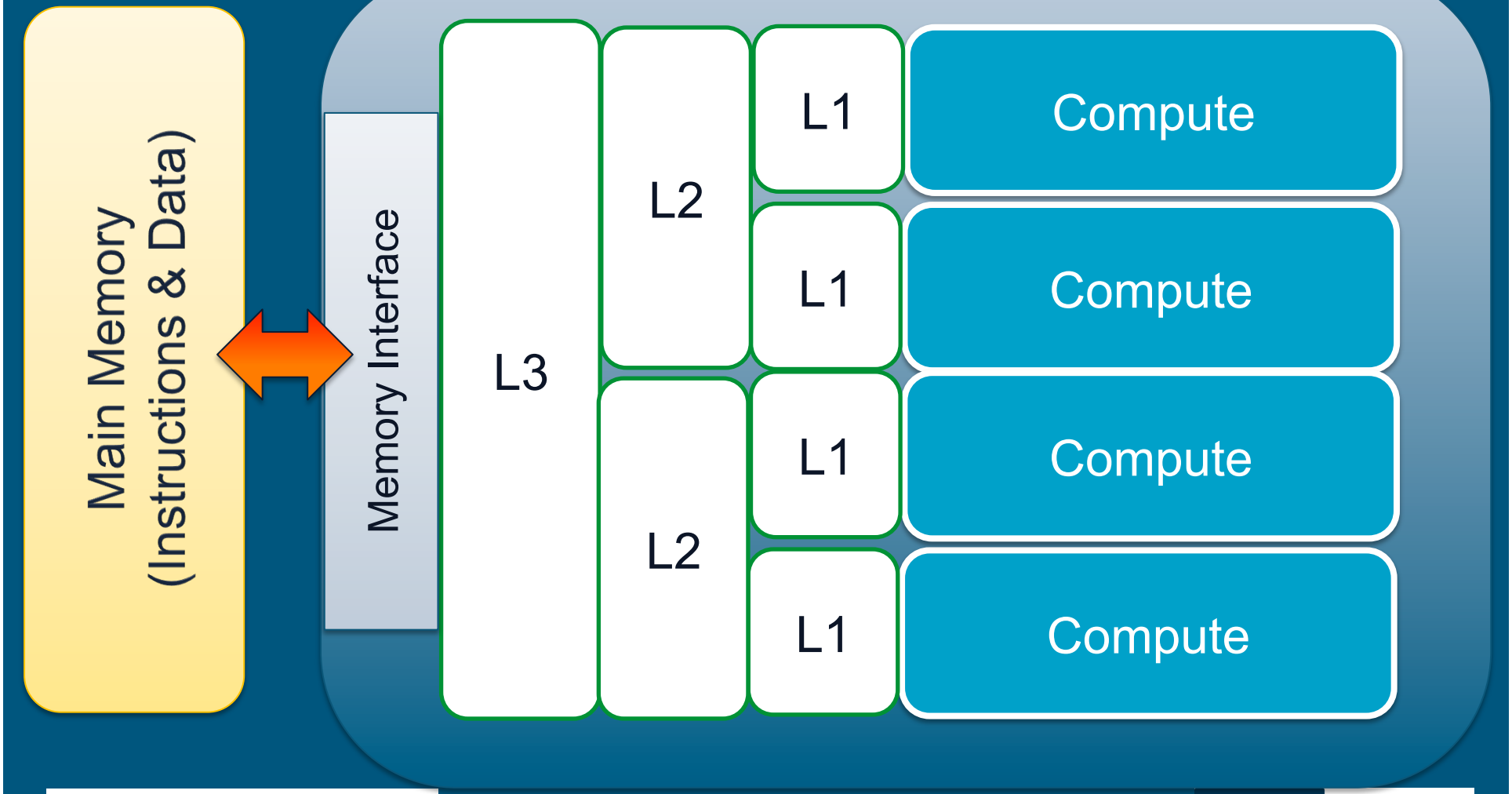
Moore's Law reinterpreted

- Number of cores per chip will double every two years
- Clock speed will not increase (possibly decrease)
- Need to deal with systems with millions of concurrent threads
- Need to deal with inter-chip parallelism as well as intra-chip parallelism

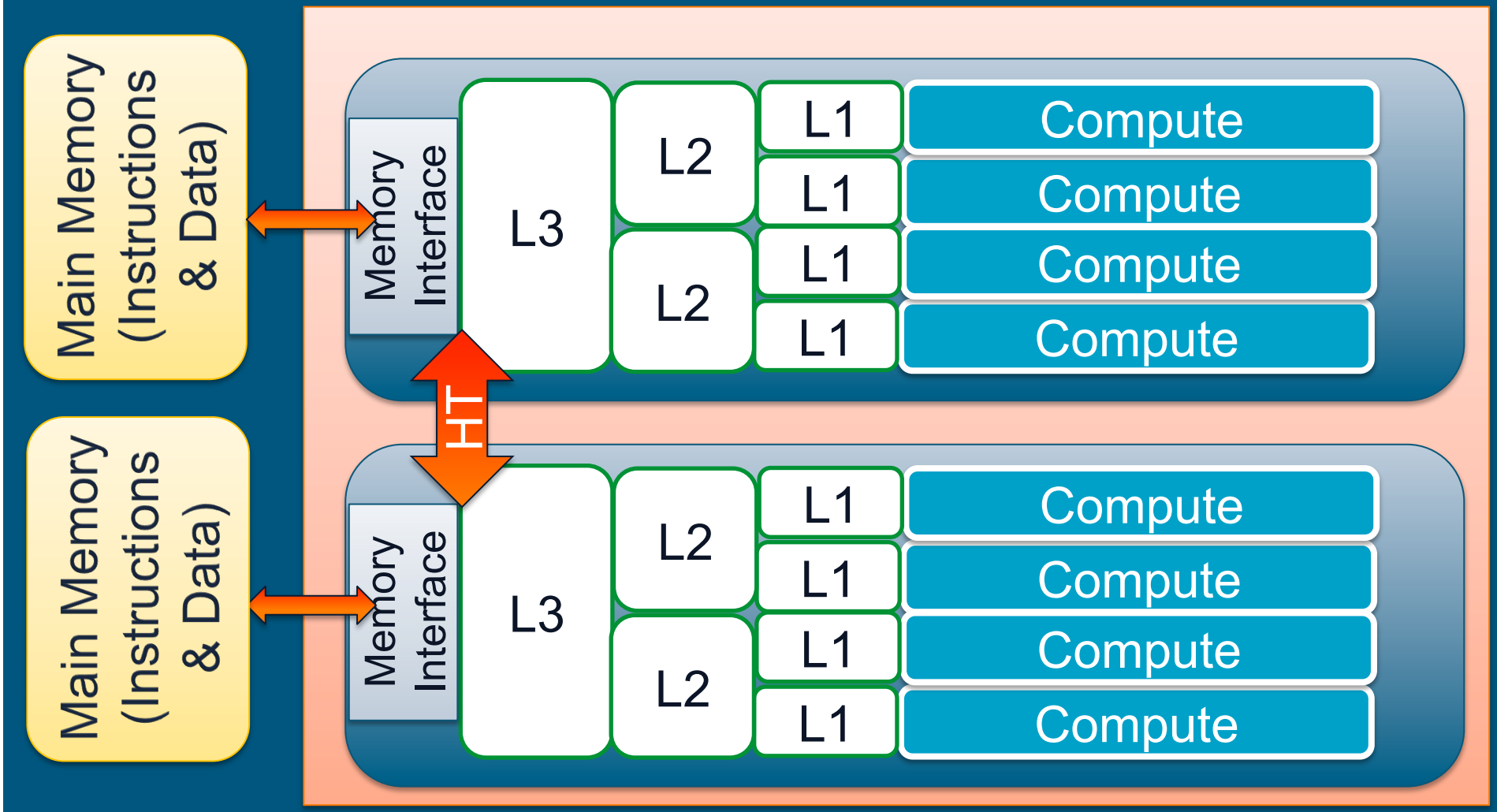
Example HPC CPU (2 core)



Example HPC CPU (4 core)



Hypothetical Socket (8 core)



HPC Nodes

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

HPC Node

- **A "node" is a (physical) collection of CPUs, memory, and interfaces to other nodes and devices.**
  - Single memory address space
  - Memory access "on-node" is significantly faster than "off-node" memory access
  - Often called an "SMP node" for "Shared Memory Processing"
    - Not necessarily "symmetric" memory access as in "Symmetric Multi-Processing"

Example SMP Node

SMP Node – Each core has equal access to memory and cache

Example NUMA Node

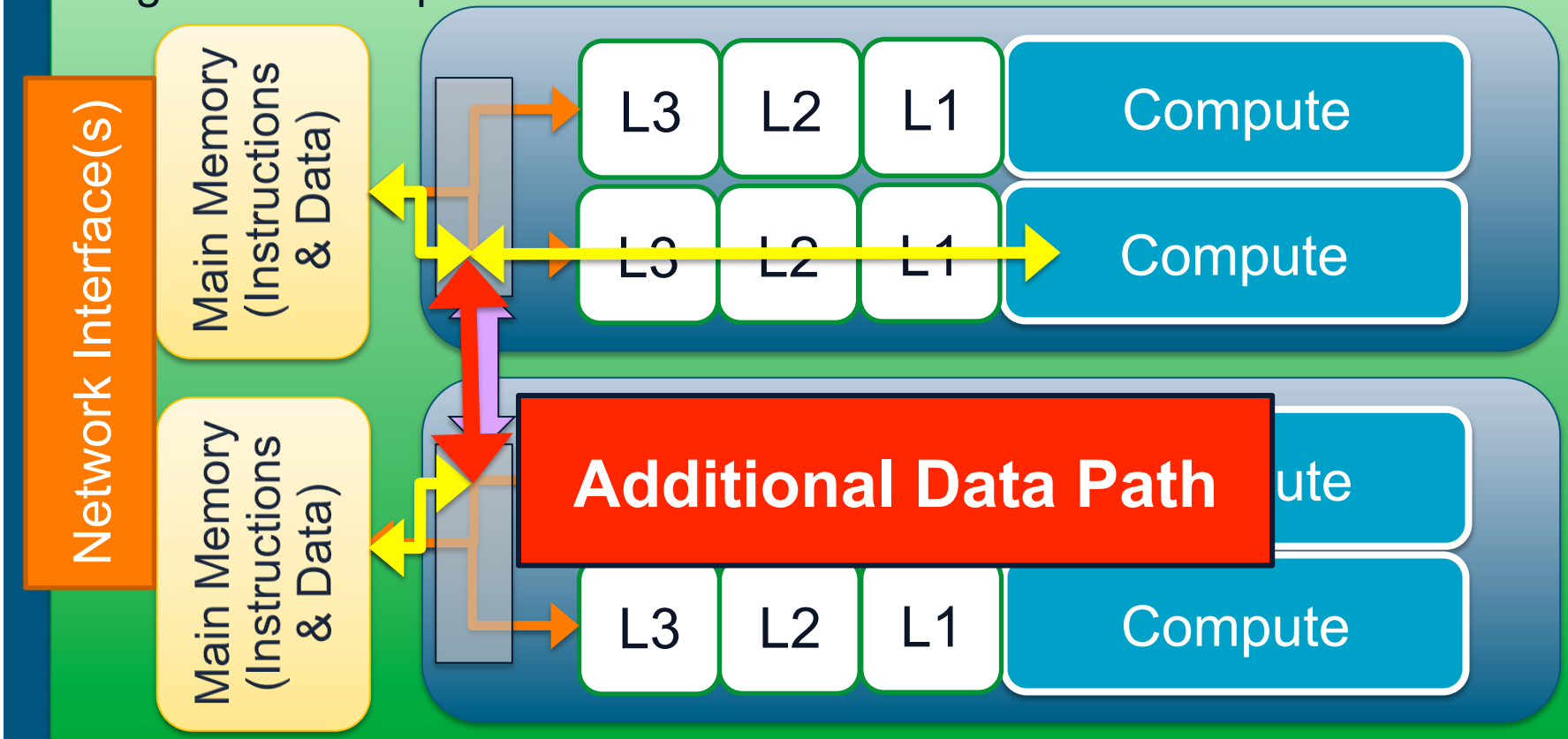
NUMA Node – Non-Uniform Memory Access

Single address space

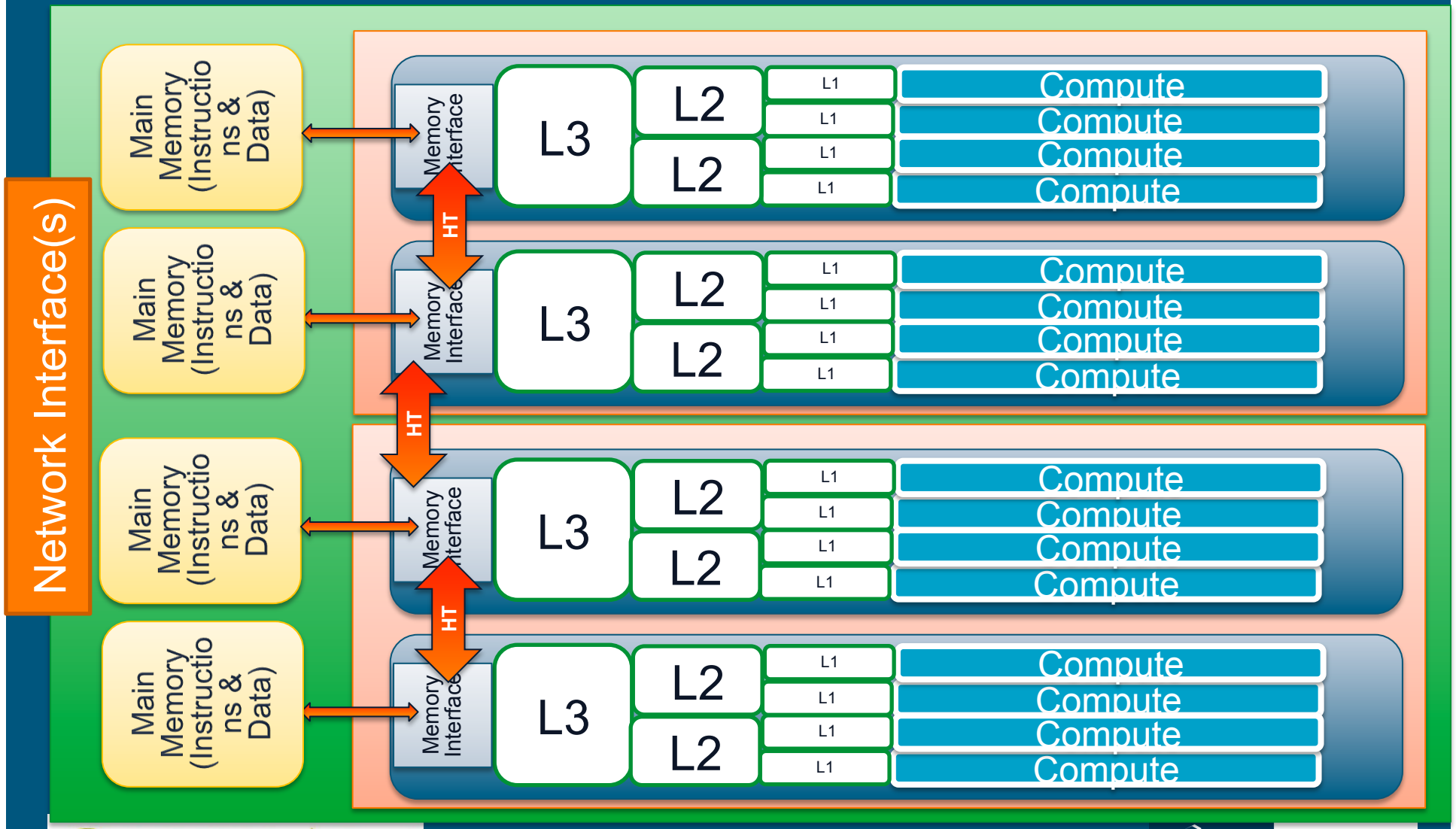
Example NUMA Node

NUMA Node – Non-Uniform Memory Access

Single address space



NUMA node (4 dies, 2 sockets)



Internode Networks

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

Distributed Memory Systems

- **Most HPC systems are "distributed memory"**
  - Many nodes, each with its own local memory and distinct memory space
  - Nodes communicate over a specialized high-speed, low-latency network
  - SPMD (Single Program Multiple Data) is the most common model
    - Multiple copies of a single program (tasks) execute on different processors, but compute with different data
    - Explicit programming methods (MPI) are used to move data among different tasks

Interconnect Characteristics

- **Latency**
  - The startup-time needed to initiate a data transfer between nodes (time to send a zero-byte message)
  - Latencies between different nodes may be different
  - Typically ~ a few µsec
- **Bandwidth**
  - Data transfer rate between nodes
  - May be quoted as uni- or bi-directional
  - Typically ~ a few GB/sec in/out of a node
- **Bisection Bandwidth**
  - If a network is divided into two equal parts, the bandwidth between them is the bisection bandwidth

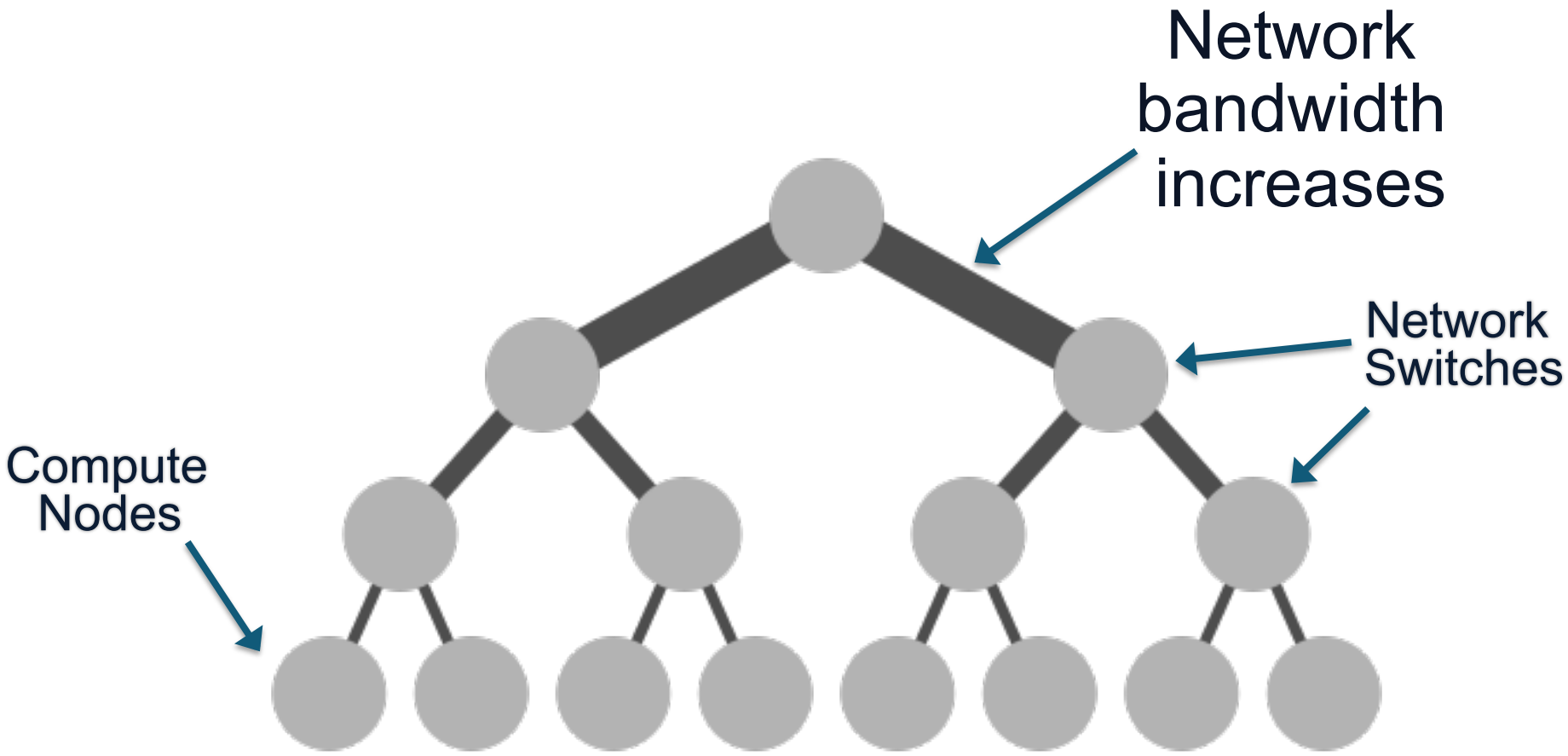
Examples

Network	Bandwidth (GB/s)	Latency (µs)
Arista 10GbE(stated)	1.2	4.0
BLADE 10GbE(measured)	1.0	4.0
Cray SeaStar2+ (measured)	6.0	4.5
Cray Gemini (measured)	6.1	1.0
IBM (Infiniband) (measured)	1.2	4.5
SGI NumaLink 5(measured)	5.9	0.4
Infiniband (measured)	1.3	4.0
Infinipath (measured)	0.9	1.5
Myrinet 10-G (measured)	1.2	2.1

Main Networks Types

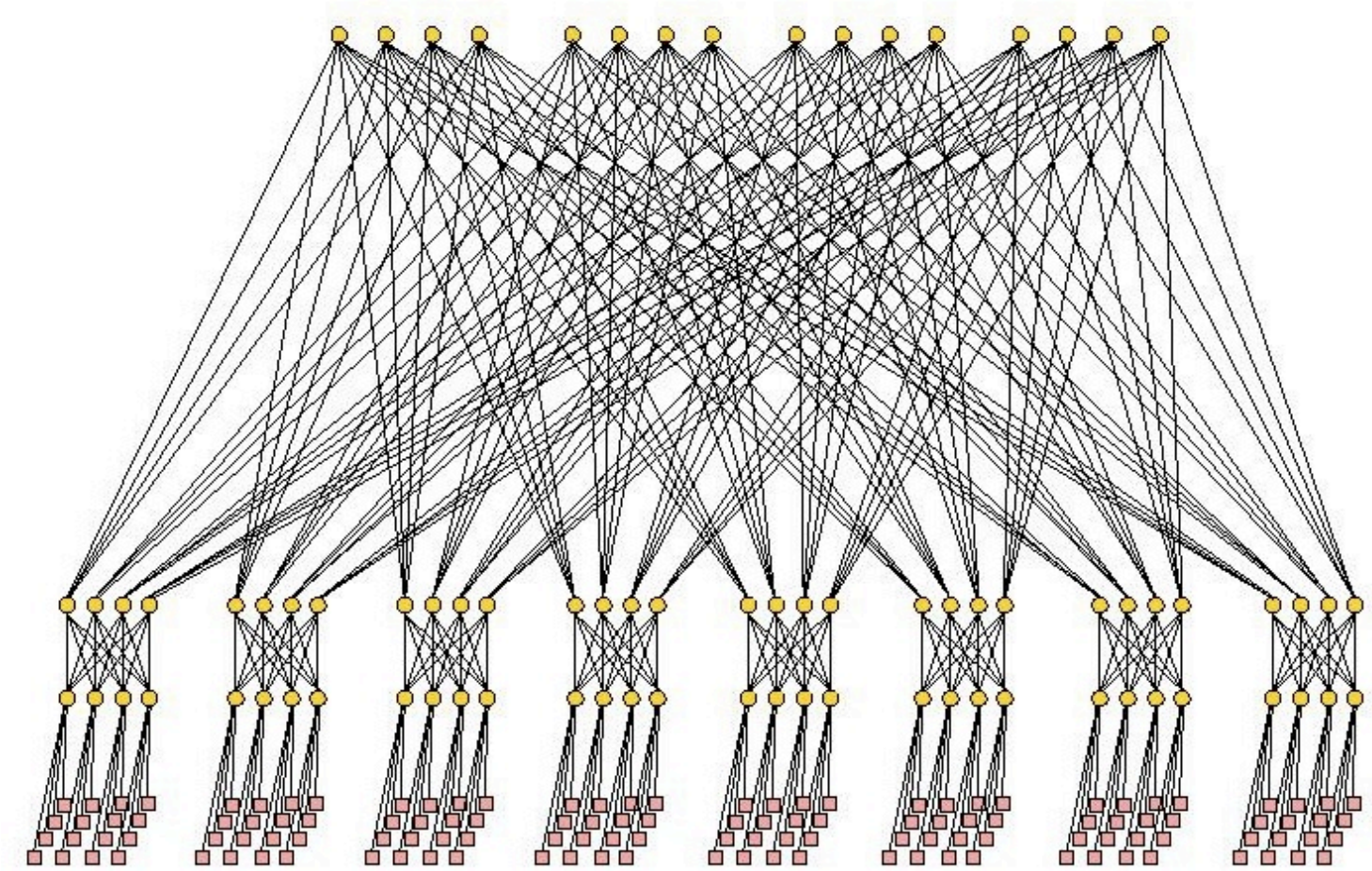
- **Switched**
  - Network switches connect and route network traffic over the interconnect
- **Mesh**
  - Each node sends and receives its own data, and also relays data from other nodes
  - Messages hop from one node to another until they reach their destination (must deal with routing around down nodes)

Fat Tree Switched Network



e.g., Infiniband (IB)

Implementation Can Be Complex

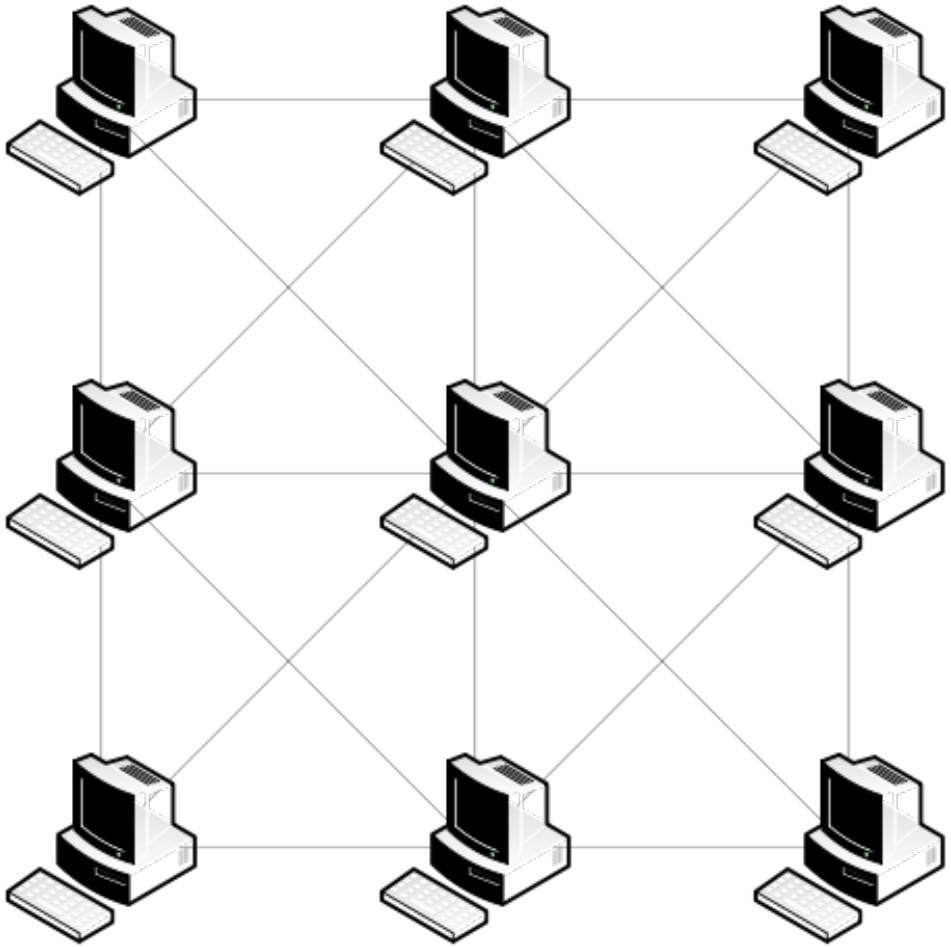


128-way fat tree

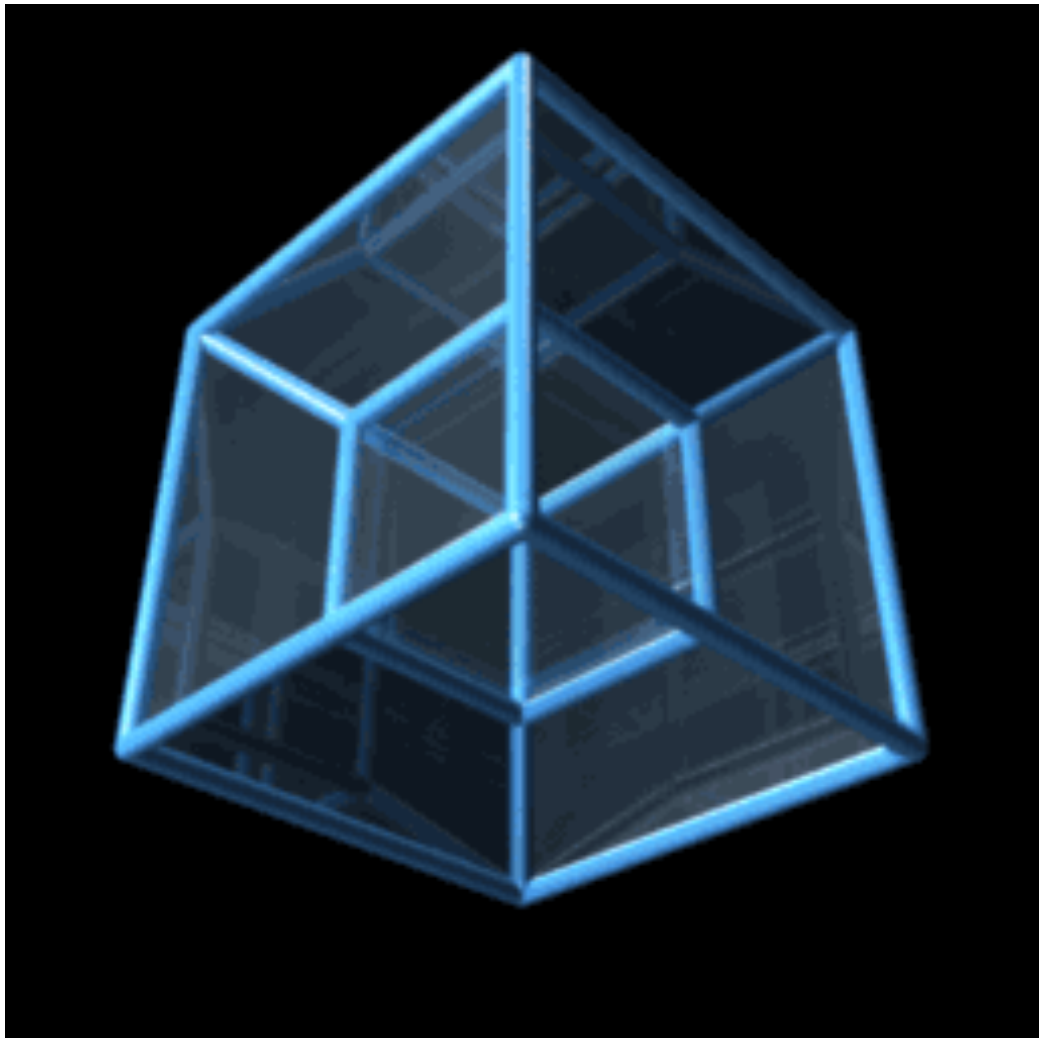
Mesh Networks

Mesh network topologies can be complex

Grids
Cubes
Hypercubes
Tori



4-D Hypercube



Disk and Tape Storage

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

- Local vs. Global File systems
- Connection methods
- Archival storage
- FLASH
- Bandwidths
- Slowest level in memory hierarchy

Why Do You Care?

- **Details of machine are important for performance**
  - Processor and memory system (not just parallelism)
  - What to expect? Use understanding of hardware limits
- **There is parallelism hidden within processors**
  - Pipelining, SIMD, etc
- **Locality is at least as important as computation**
  - Temporal: re-use of data recently used
  - Spatial: using data nearby that recently used
- **Machines have memory hierarchies**
  - 100s of cycles to read from DRAM (main memory)
  - Caches are fast (small) memory that optimize average case
- **Can rearrange code/data to improve locality**

HPC Systems

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

Top 500

- **Listing the 500 most powerful computers in the world**
- **Yardstick: Rmax of Linpack**
  - Solve Ax=b, dense problem, matrix is random
  - Dominated by dense matrix-matrix multiply
- **Update twice a year:**
  - ISC'xy in June in Germany
  - SCxy in November in the U.S.
- **All information available from the TOP500 web site at: www.top500.org**

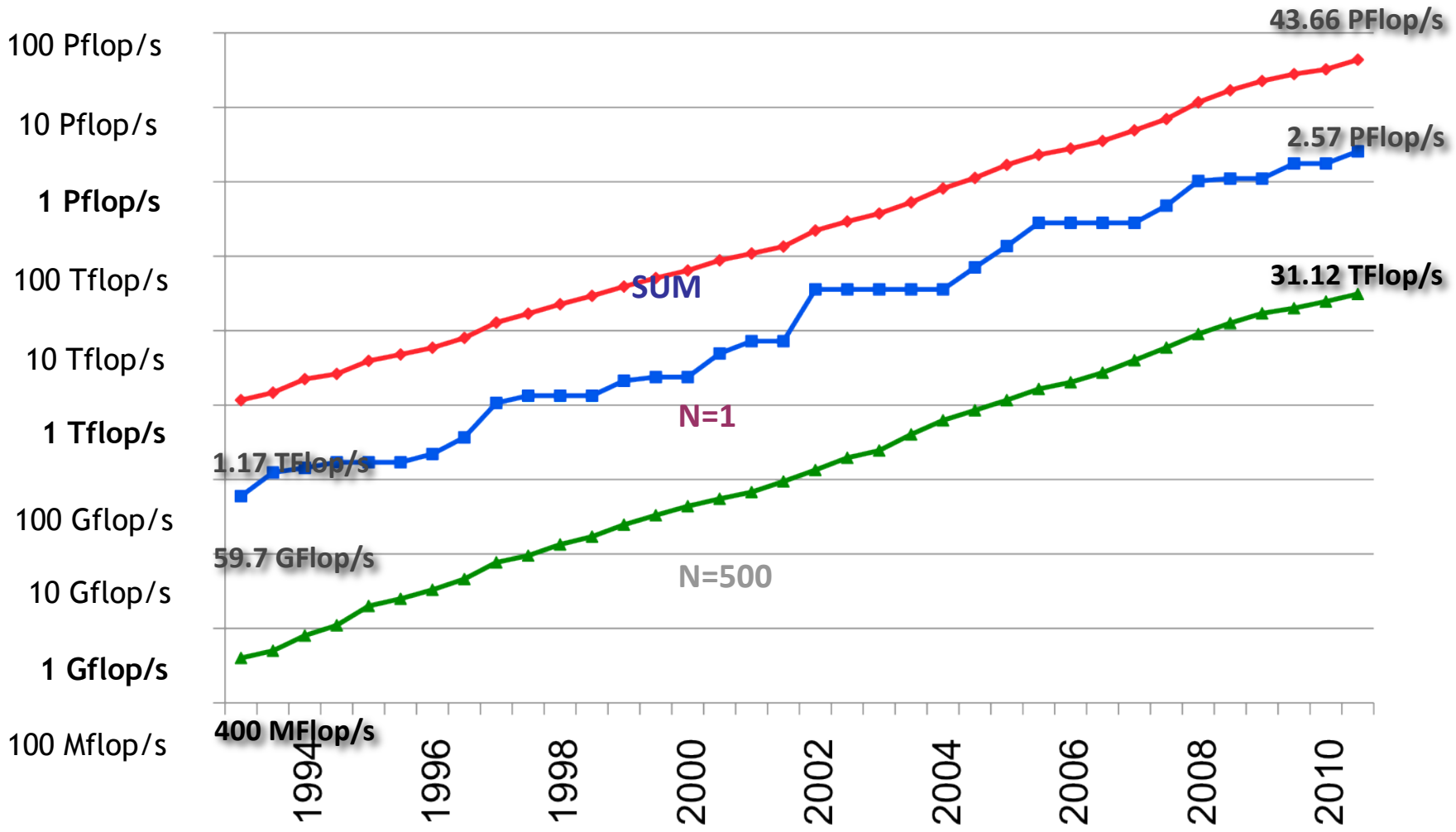
Nov 2010 Top 500 List

Rank	Site	Manufacturer	Computer	Country	Cores	Rmax [Tflops]	Power [MW]
1	National SuperComputer Center in Tianjin	NUDT	Tianhe-1A NUDT TH MPP, Xeon 6C, NVidia, FT-1000 8C	China	186,368	2,566	4.04
2	Oak Ridge National Laboratory	Cray	Jaguar Cray XT5, HC 2.6 GHz	USA	224,162	1,759	6.95
3	National Supercomputing Centre in Shenzhen	Dawning	Nebulae TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU	China	120,640	1,271	2.58
4	GSIC, Tokyo Institute of Technology	NEC/HP	TSUBAME-2 HP ProLiant, Xeon 6C, NVidia, Linux/Windows	Japan	73,278	1,192	1.40
5	DOE/SC/ LBNL/NERSC	Cray	Hopper Cray XE6, 6C 2.1 GHz	USA	153,408	1.054	2.91
6	Commissariat a l'Energie Atomique (CEA)	Bull	Tera 100 Bull bullx super-node S6010/S6030	France	138.368	1,050	4.59
7	DOE/NNSA/LANL	IBM	Roadrunner BladeCenter QS22/LS21	USA	122,400	1,042	2.34
8	University of Tennessee	Cray	Kraken Cray XT5 HC 2.36GHz	USA	98,928	831.7	3.09
9	Forschungszentrum Juelich (FZJ)	IBM	Jugene Blue Gene/P Solution	Germany	294,912	825.5	2.26
10	DOE/NNSA/ LANL/SNL	Cray	Cielo Cray XE6, 6C 2.4 GHz	USA	107,152	816.6	2.95

GPUs

- 3 of top 4 are GPU-accelerated systems
- "Graphics Processing Units" are composed of 100s of simple "cores" that provide data-level on-chip parallelism
- Yet more low-level complexity to consider
  - Another interface with the socket (or on socket?)
  - Limited, private memory (for now?)
- Programmability is currently poor
- Legacy codes may have to be rewritten to minimize data movement
- Not all algorithms map well to GPUs
- What is their future in HPC?????

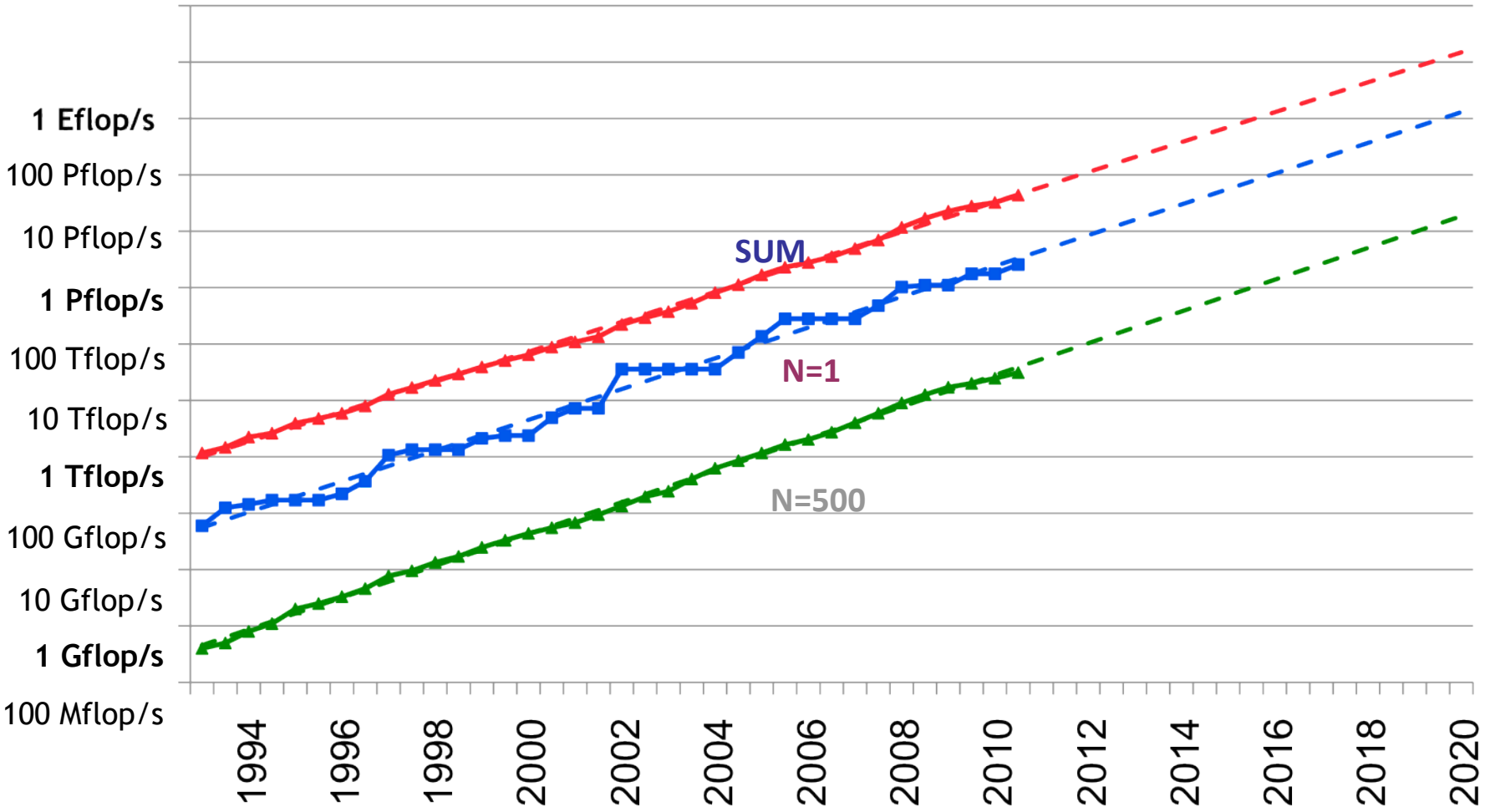
Performance Development (Top 500)



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB Lawrence Berkeley National Laboratory

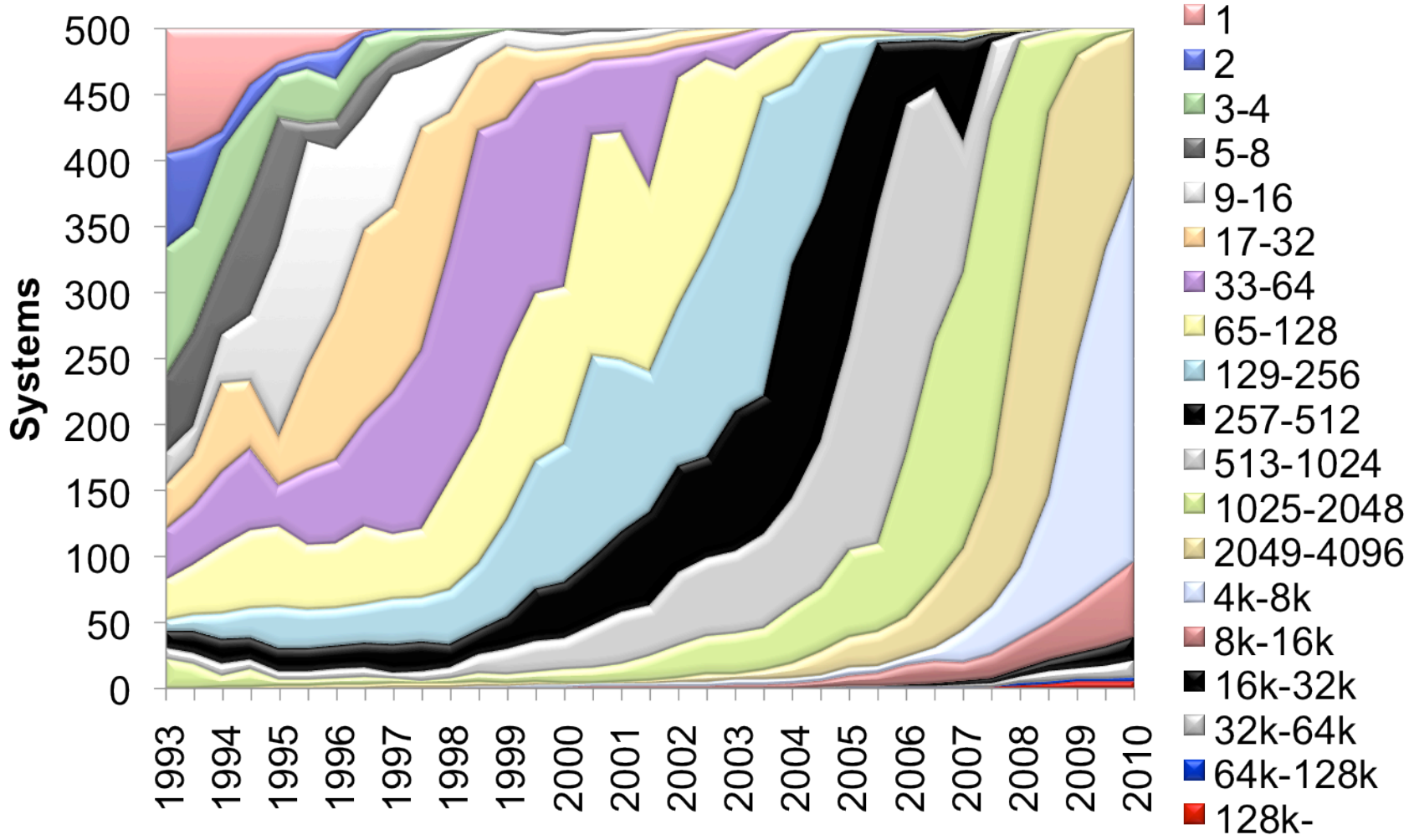
Projected Performance Development



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB | Lawrence Berkeley National Laboratory

Core Count


Systems
500
450
400
350
300
250
200
150
100
50
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
1
2
3-4
5-8
9-16
17-32
33-64
65-128
129-256
257-512
513-1024
1025-2048
2049-4096
4k-8k
8k-16k
16k-32k
32k-64k
64k-128k
128k-

National Energy Research Scientific Computing Center

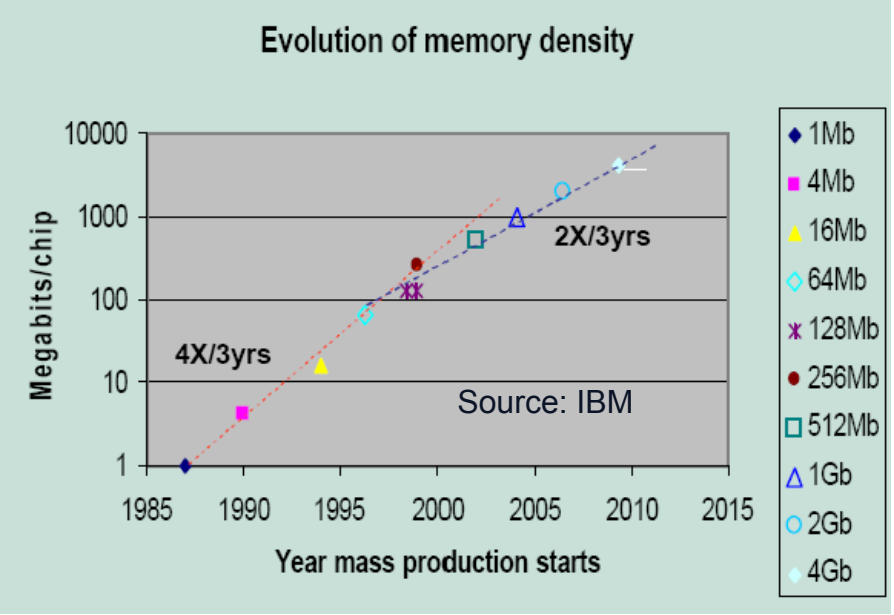
U.S. DEPARTMENT OF ENERGY | Office of Science

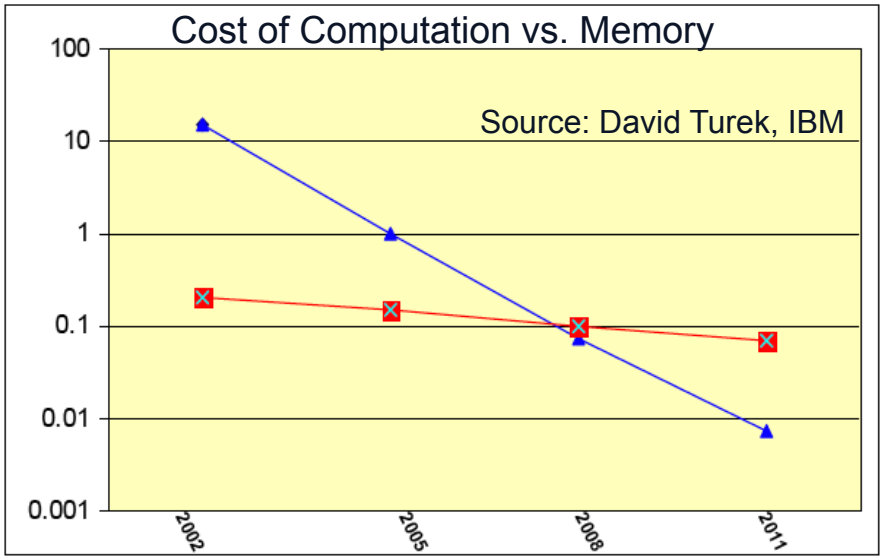
BERKELEY LAB
Lawrence Berkeley National Laboratory

Memory is Not Keeping Pace

Technology trends against a constant or increasing memory per core

- Memory density is doubling every three years; processor logic is every two
- Storage costs (dollars/Mbyte) are dropping gradually compared to logic costs





The cost to sense, collect, generate and calculate data is declining much faster than the cost to access, manage and store it

Question: Can you double concurrency without doubling memory?

- **Strong scaling:** fixed problem size, increase number of processors
- **Weak scaling:** grow problem size proportionally to number of processors

Uniprocessors in the Real World

- Real processors have
  - registers and caches
    - small amounts of fast memory
    - store values of recently used or nearby data
    - different memory ops can have very different costs
  - parallelism
    - multiple "functional units" that can run in parallel
    - different orders, instruction mixes have different costs
  - pipelining
    - a form of parallelism, like an assembly line in a factory
- Why is this your problem?
    - In theory, compilers and hardware "understand" all this and can optimize your program; in practice they don't.
    - They won't know about a different algorithm that might be a much better "match" to the processor