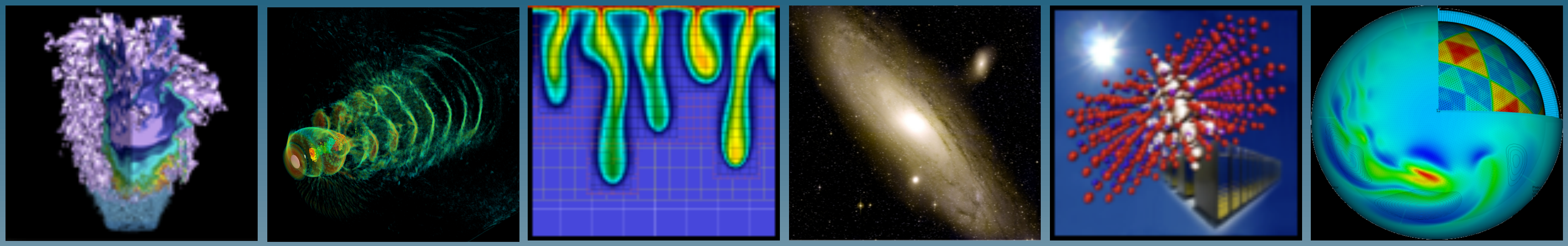


Scaling to Petascale and Beyond: Performance Analysis and Optimization of Applications

R. Glenn Brook, U. Tennessee
Don Frederick, LLNL
Richard Gerber, NERSC / Berkley Lab
Jeff Larkin, Cray, Inc.

http://www.nersc.gov/users/training/nersc-training-events/sc11/s10/

CRAY THE SUPERCOMPUTER COMPANY

NICS

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC National Energy Research Scientific Computing Center

BERKELEY LAB Lawrence Berkeley National Laboratory

Outline

Design Considerations

- 8:30 - Introduction to Petascale Systems – 30 minutes – Richard Gerber
- 9:00 - HPC Communication - 45 minutes - Glenn Brook
- 9:45 - Break
- 10:00 - HPC IO & Lustre File Systems - Richard Gerber, 50 minutes
- 10:50 - HPC Computation - 40 minutes - Jeff Larkin

- **Multi-Core Performance Analysis**
  - 11:30 - Introduction and General Methodology - 30 minutes Don Frederick and Glenn Brook
  - 12:00 - Lunch
  - 1:00 - Profiling with Open SpeedShop - 45 minutes Don Frederick
- **Extreme Scaling on Petascale-class Systems**
  - 1:45 - Cray XT Architecture, System Software, and Scientific Libraries - 15 minutes- Jeff Larkin
  - 2:00 - Cray XT Porting, Scaling, and Optimization - 1 hour - Jeff Larkin
  - 3:00 - Break
  - 3:15 - IBM Blue Gene P Architecture, System Software, and Scientific Libraries – 15 minutes - Don Frederick
  - 3:30 - IBM Blue Gene P Porting, Scaling and Optimization Case Studies – 1 hour -Don Frederick

Introduction to High Performance Computers

Richard Gerber
NERSC User Services

SC11 Tutorial

Scaling to Petascale and Beyond: Performance Analysis and Optimization of Applications

Nov. 13, 2011

U.S. DEPARTMENT OF ENERGY | Office of Science
NERSC National Energy Research Scientific Computing Center
BERKELEY LAB Lawrence Berkeley National Laboratory

Outline

- Background
- CPUs
- Nodes
- Interconnect
- Data Storage
- Operating Systems

For updates to all slides in this day-long tutorial, see

http://www.nersc.gov/users/training/nersc-training-events/sc11/

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
- You want to efficiently configure the way your job runs
- The technology is just cool!

Major Parts of an HPC System

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

A distributed-memory HPC system

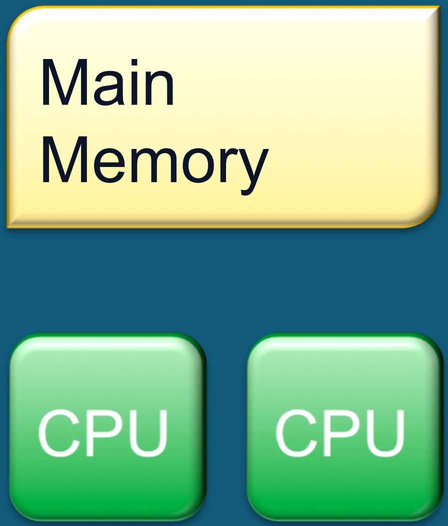
CPU

A distributed-memory HPC system

CPU

CPU

A distributed-memory HPC system

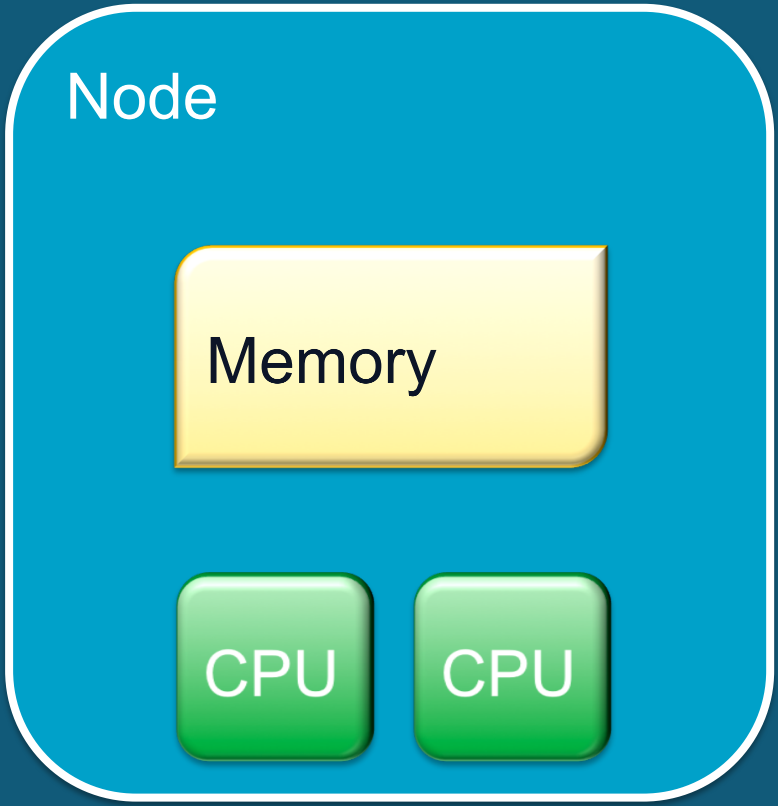


Main Memory

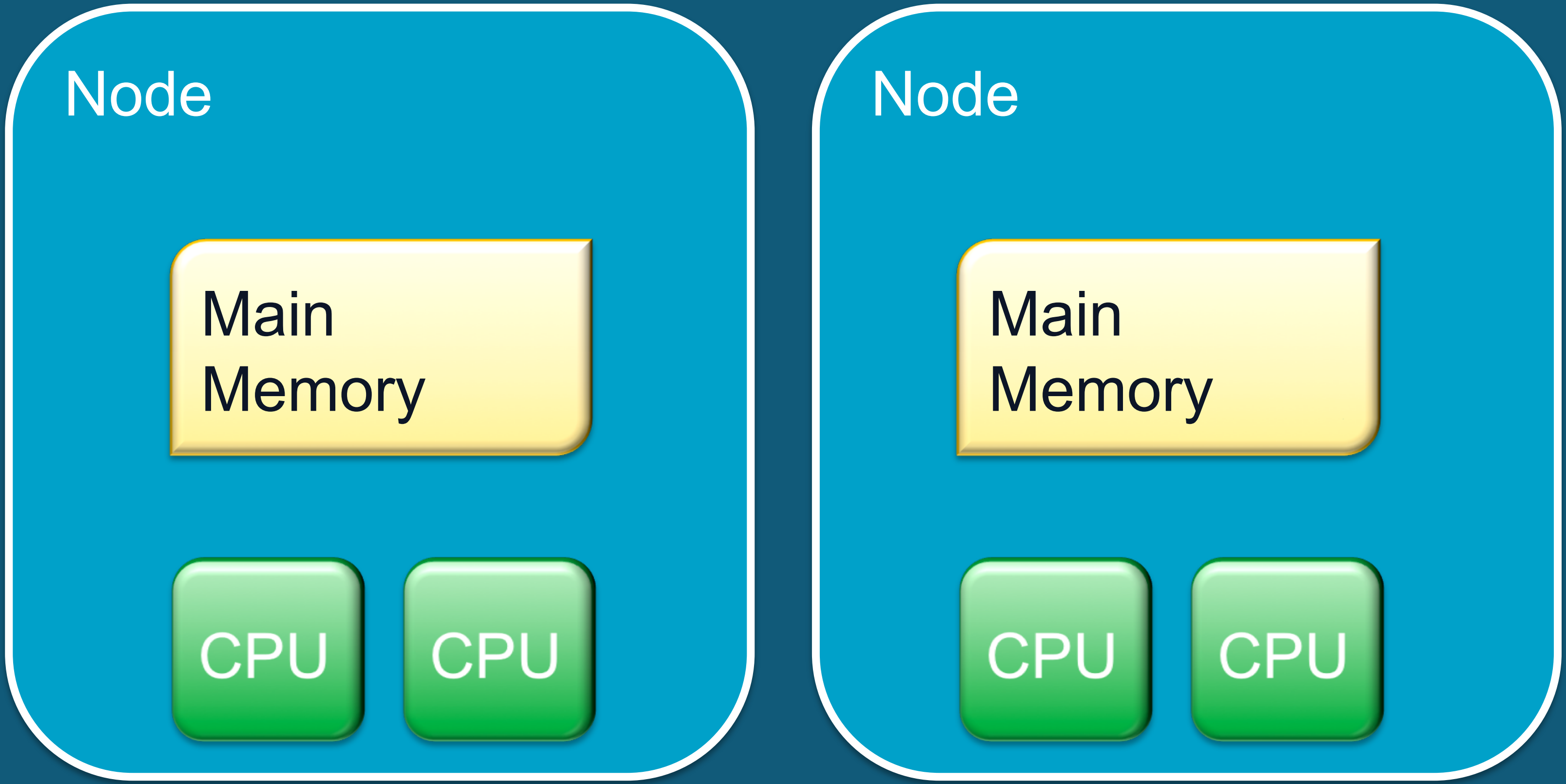
CPU

CPU

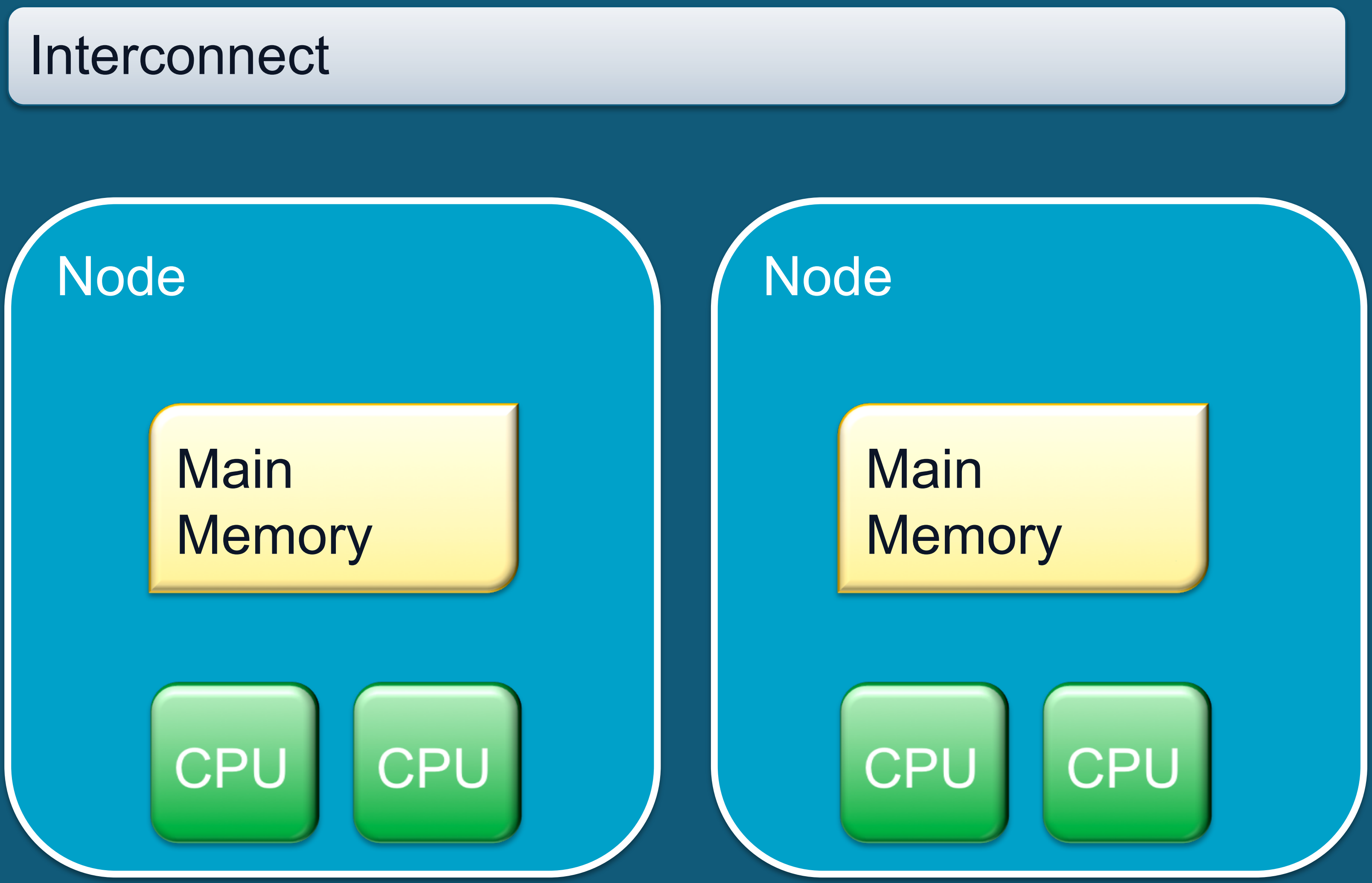
A distributed-memory HPC system



A distributed-memory HPC system



A distributed-memory HPC system



Interconnect

Node

Main Memory

CPU

CPU

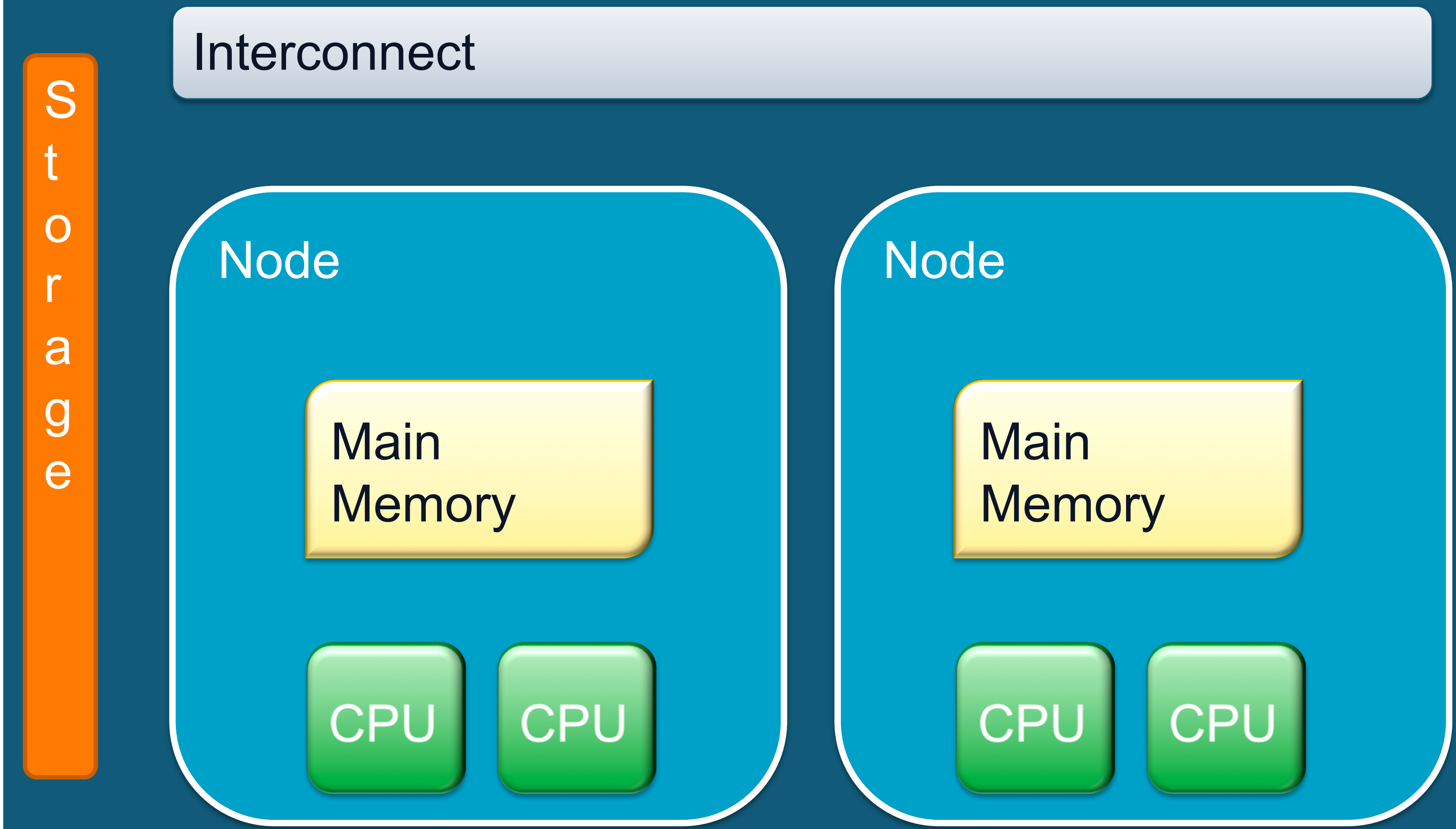
Node

Main Memory

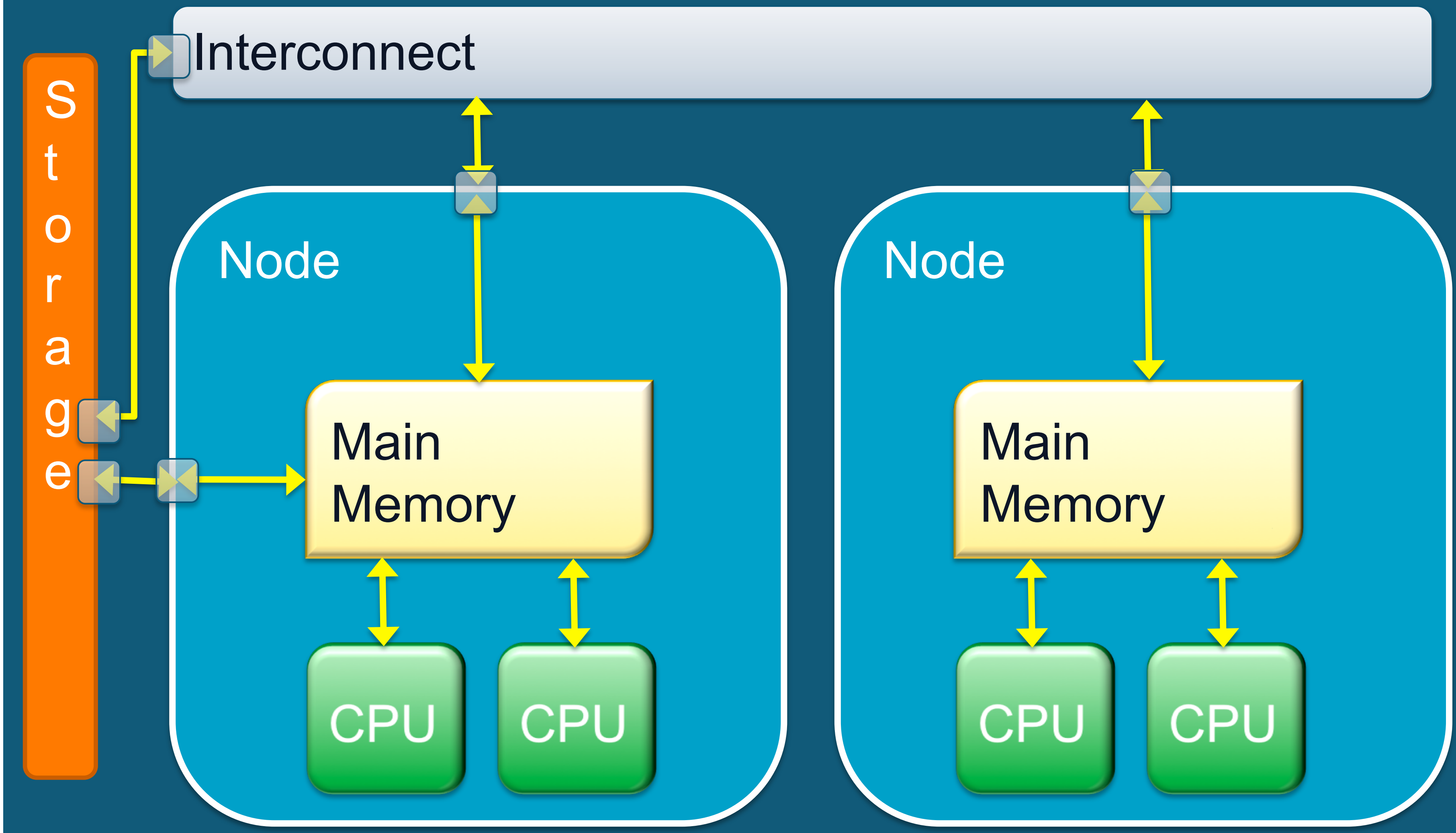
CPU

CPU

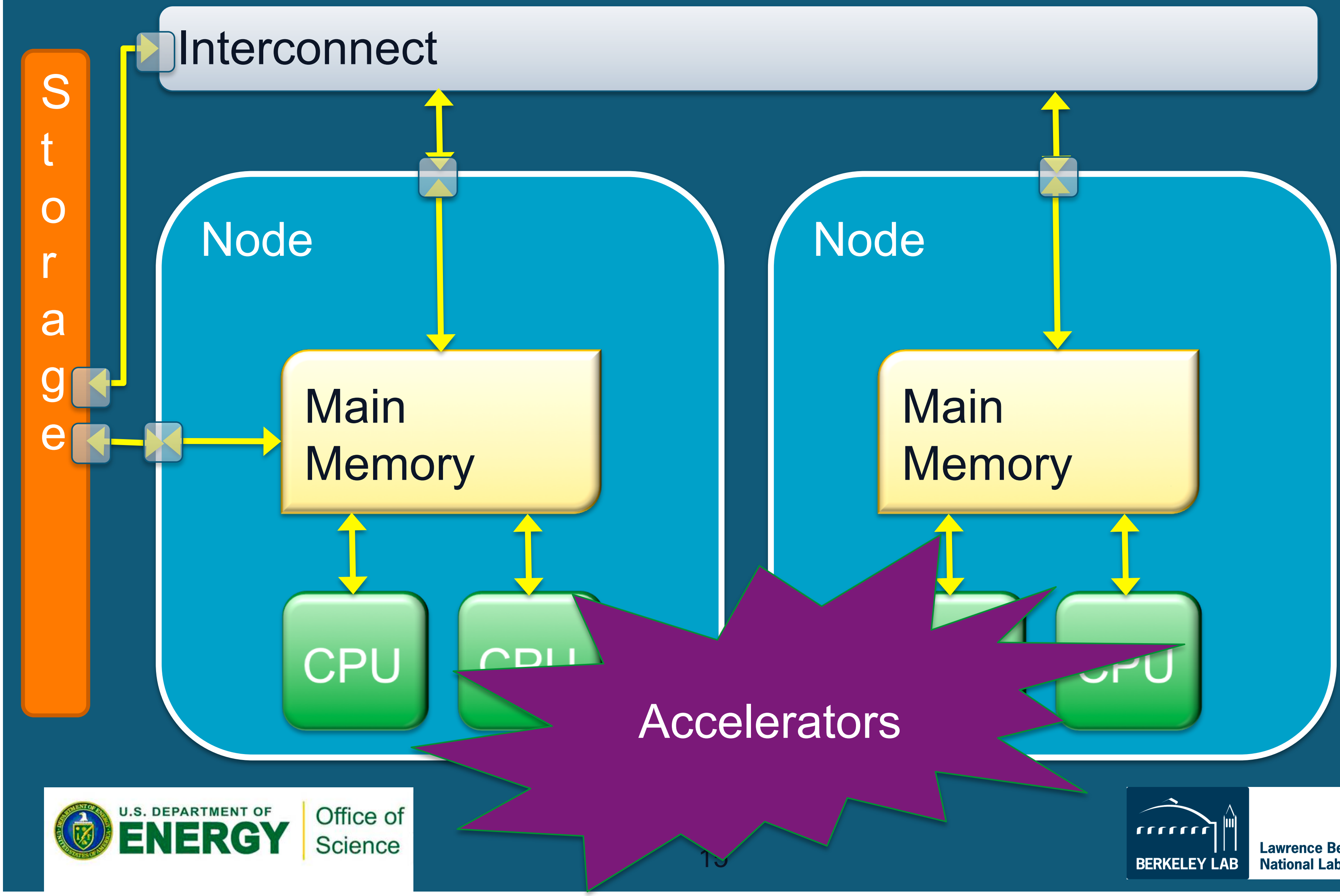
A distributed-memory HPC system



A distributed-memory HPC system



A distributed-memory HPC system

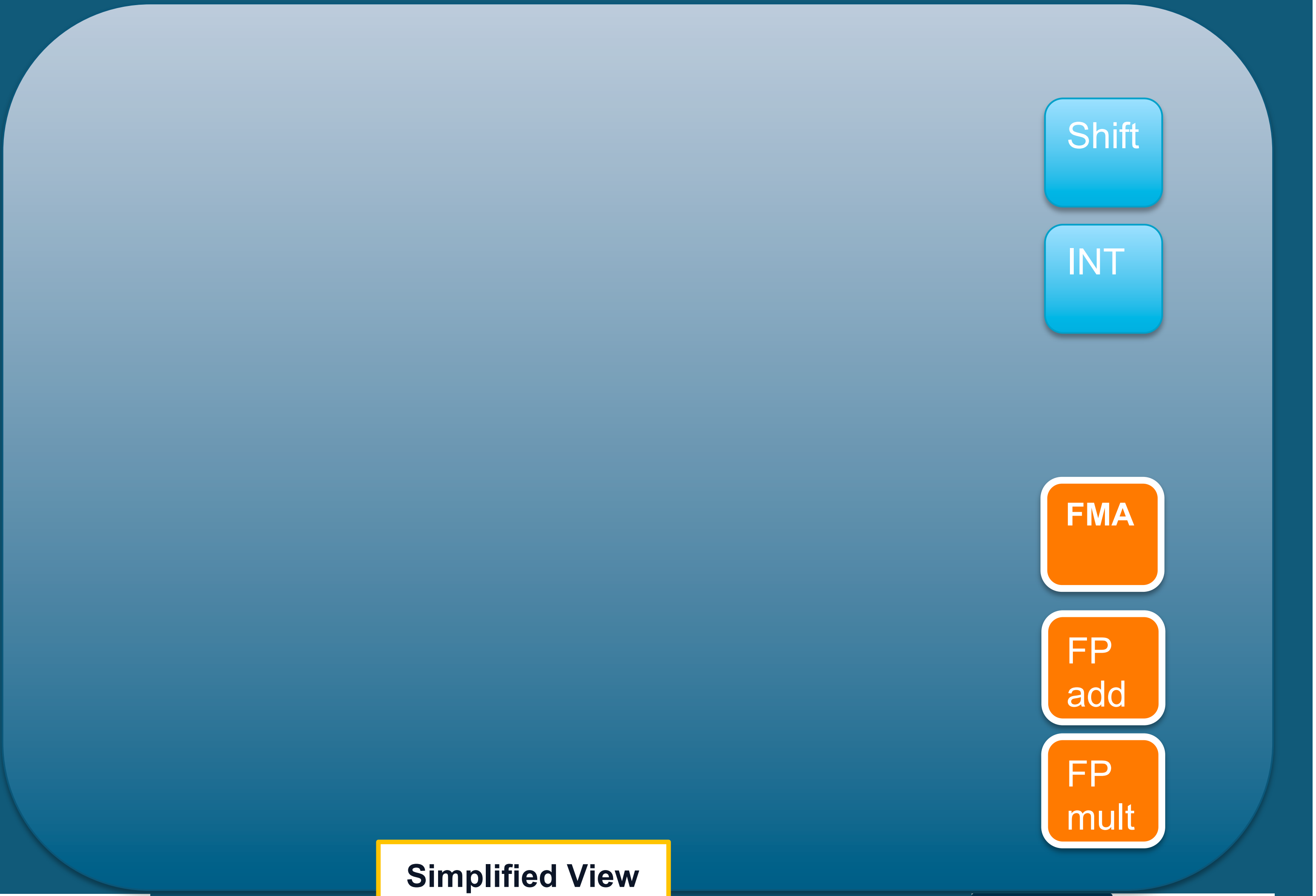


GPUs

- 3 of top 5 systems on the Top 500 are GPU-accelerated
- "Graphics Processing Units" are composed of 100s of simple "cores" that increase 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?????

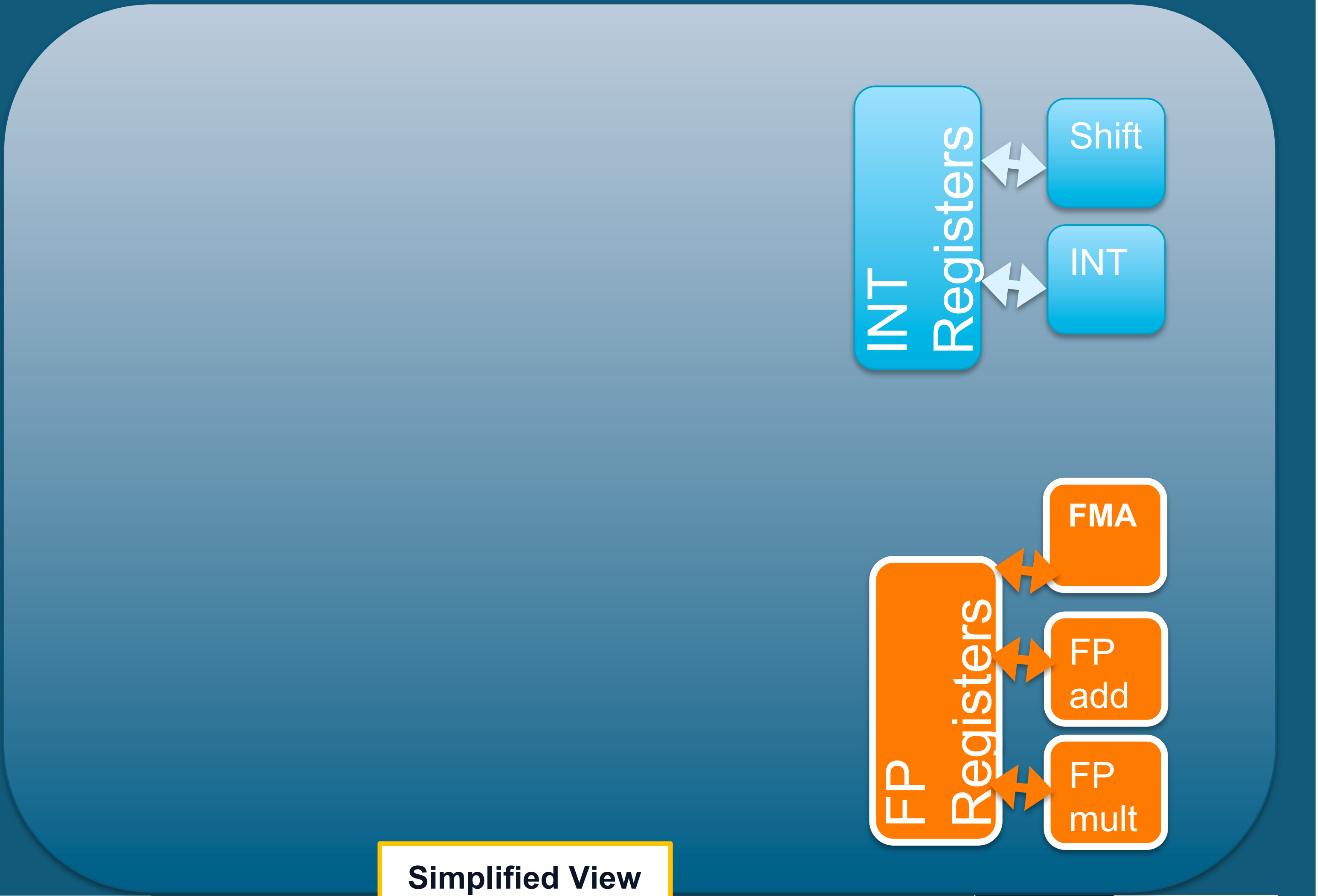
CPUs

CPU (1 core)



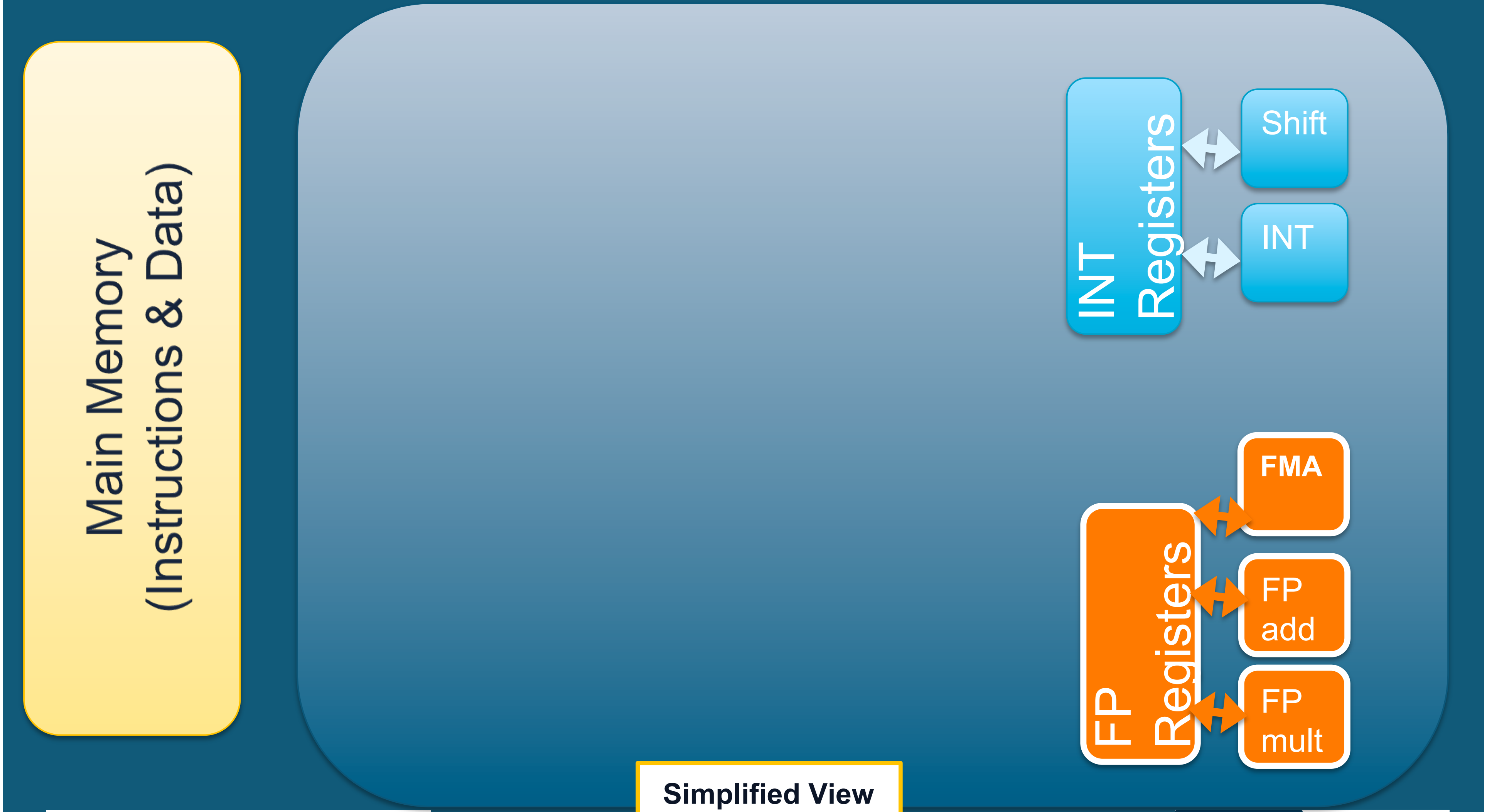
Simplified View

CPU (1 core)



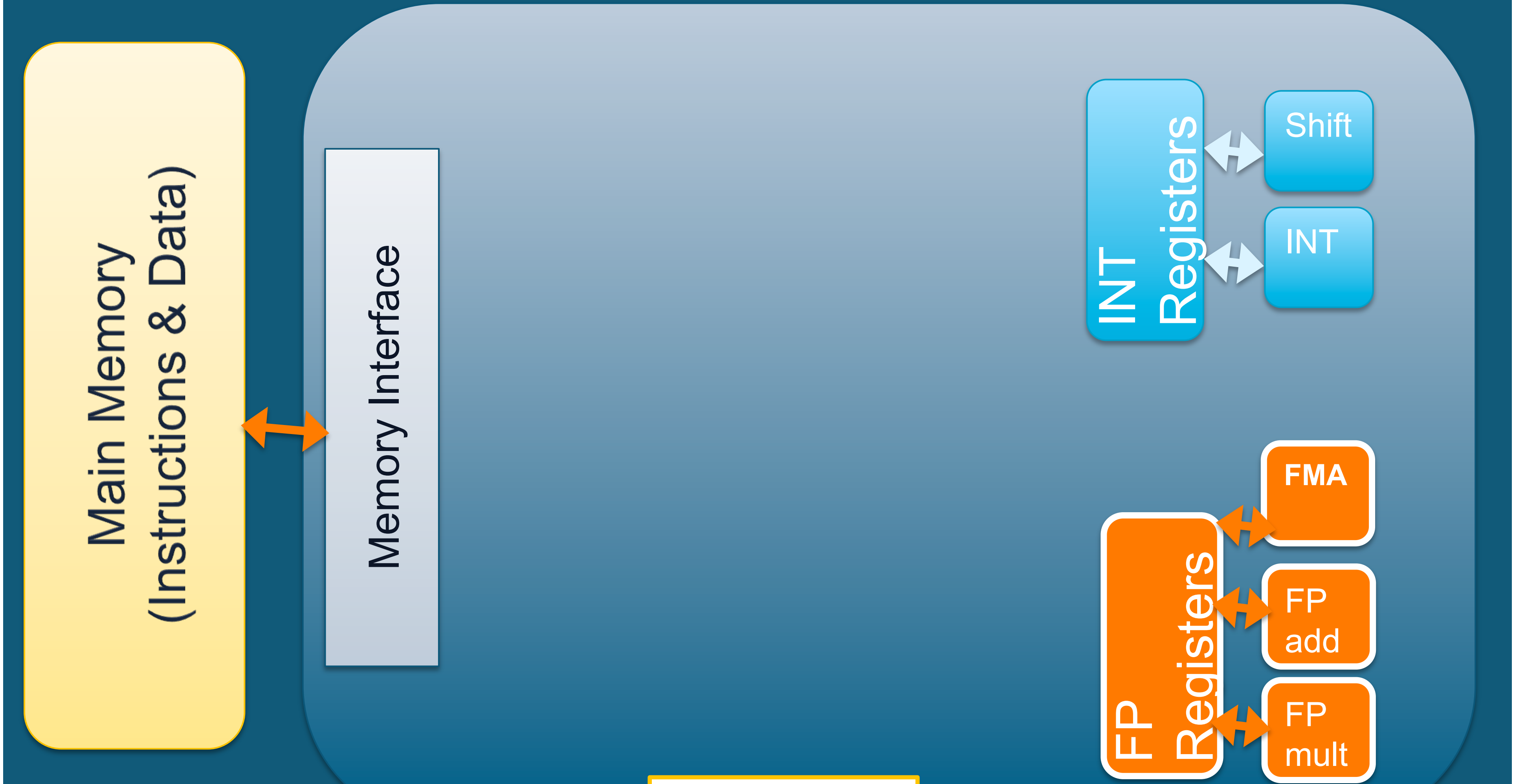
Simplified View

CPU (1 core)



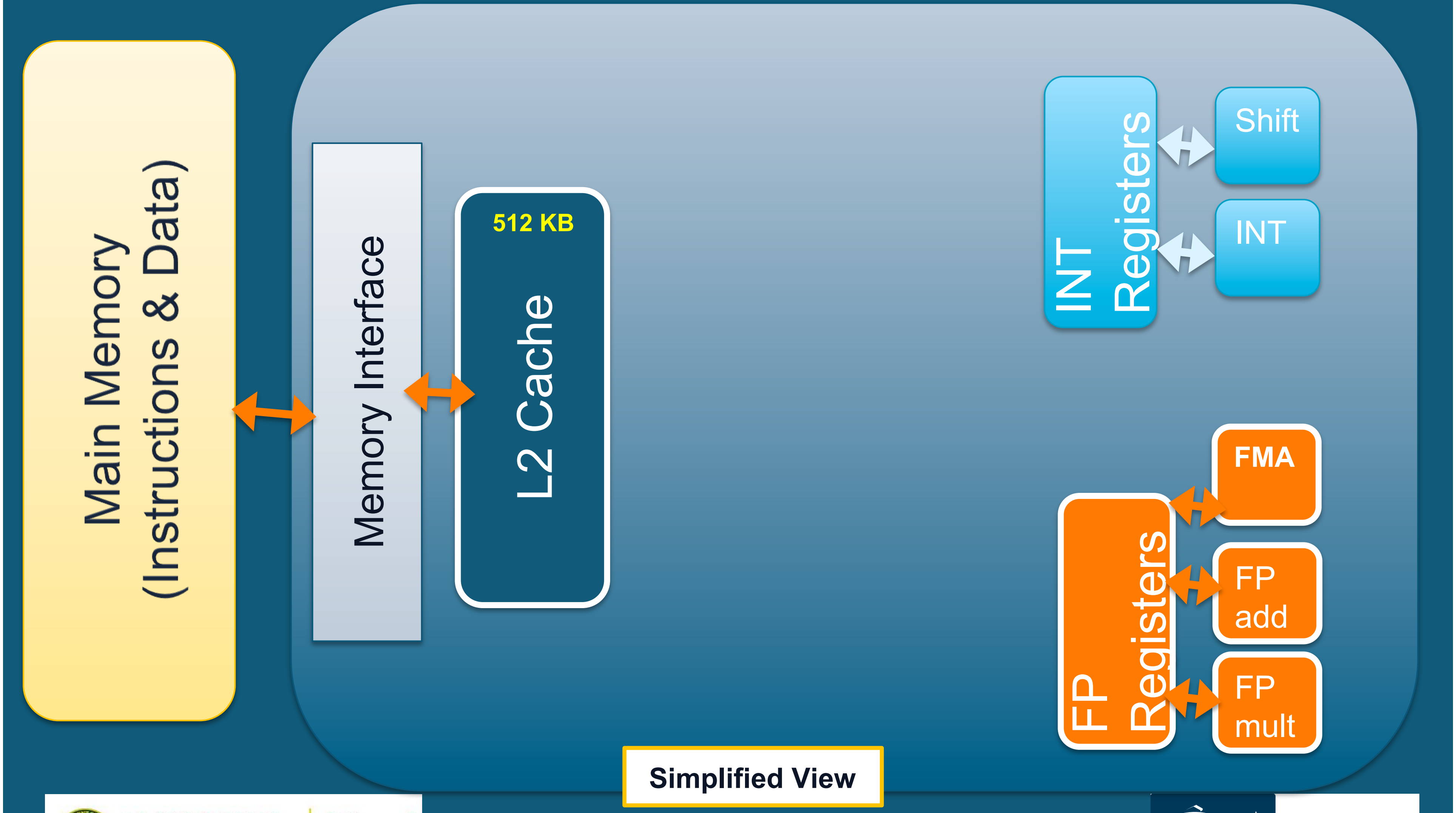
Simplified View

CPU (1 core)



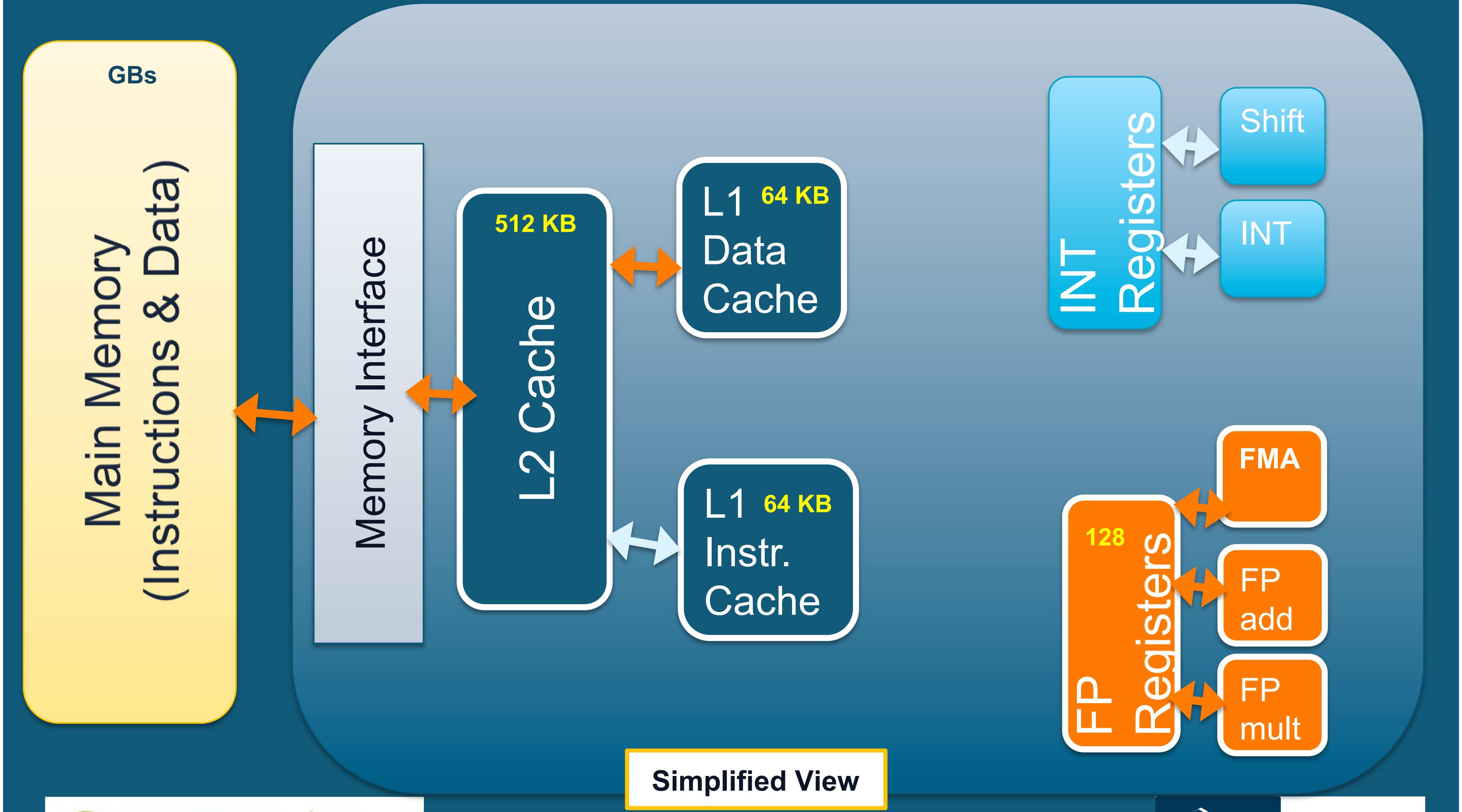
Simplified View

CPU (1 core)



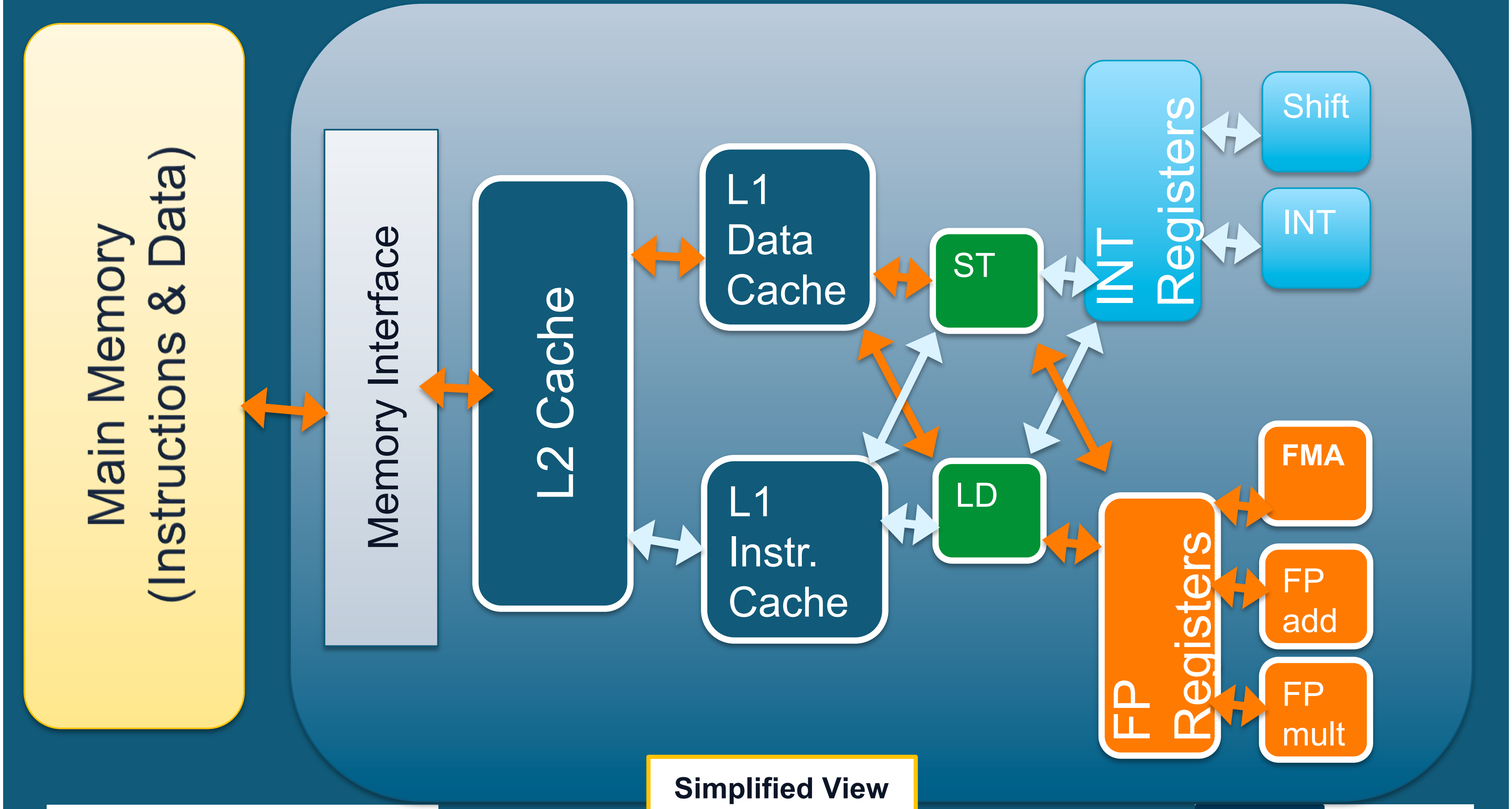
Simplified View

CPU (1 core)



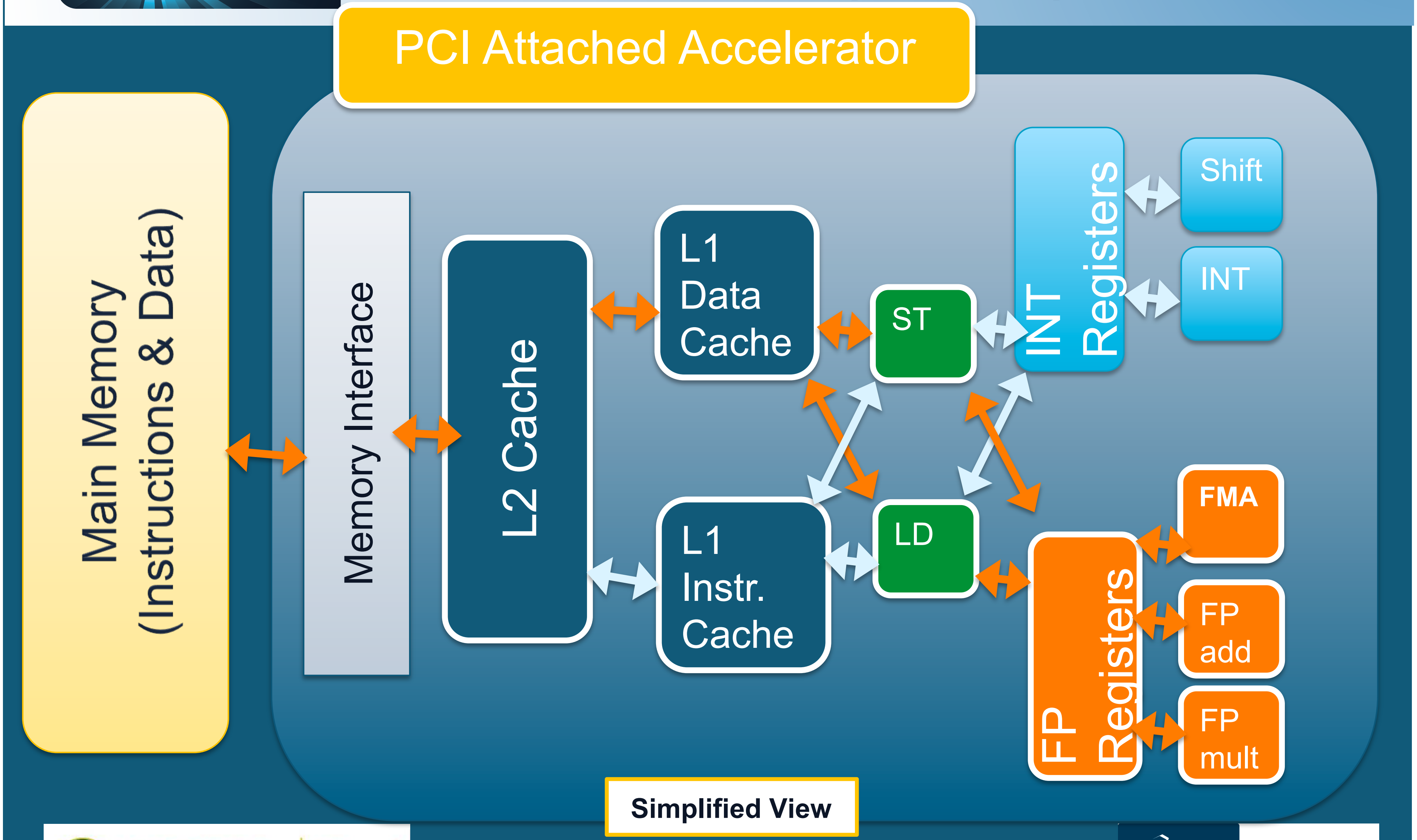
Simplified View

CPU (1 core)



Simplified View

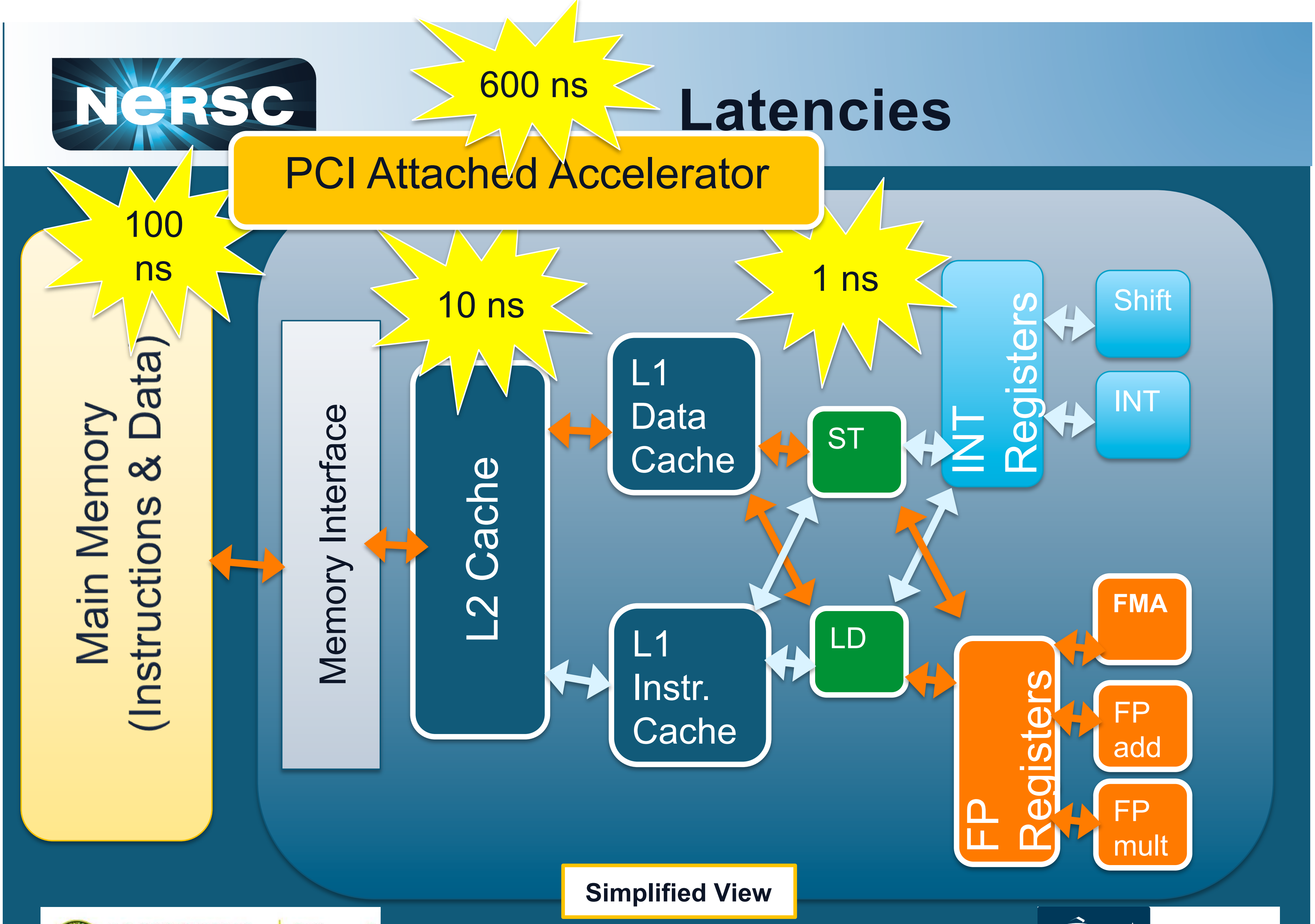
CPU (1 core)


PCI Attached Accelerator
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
  - …

Latencies

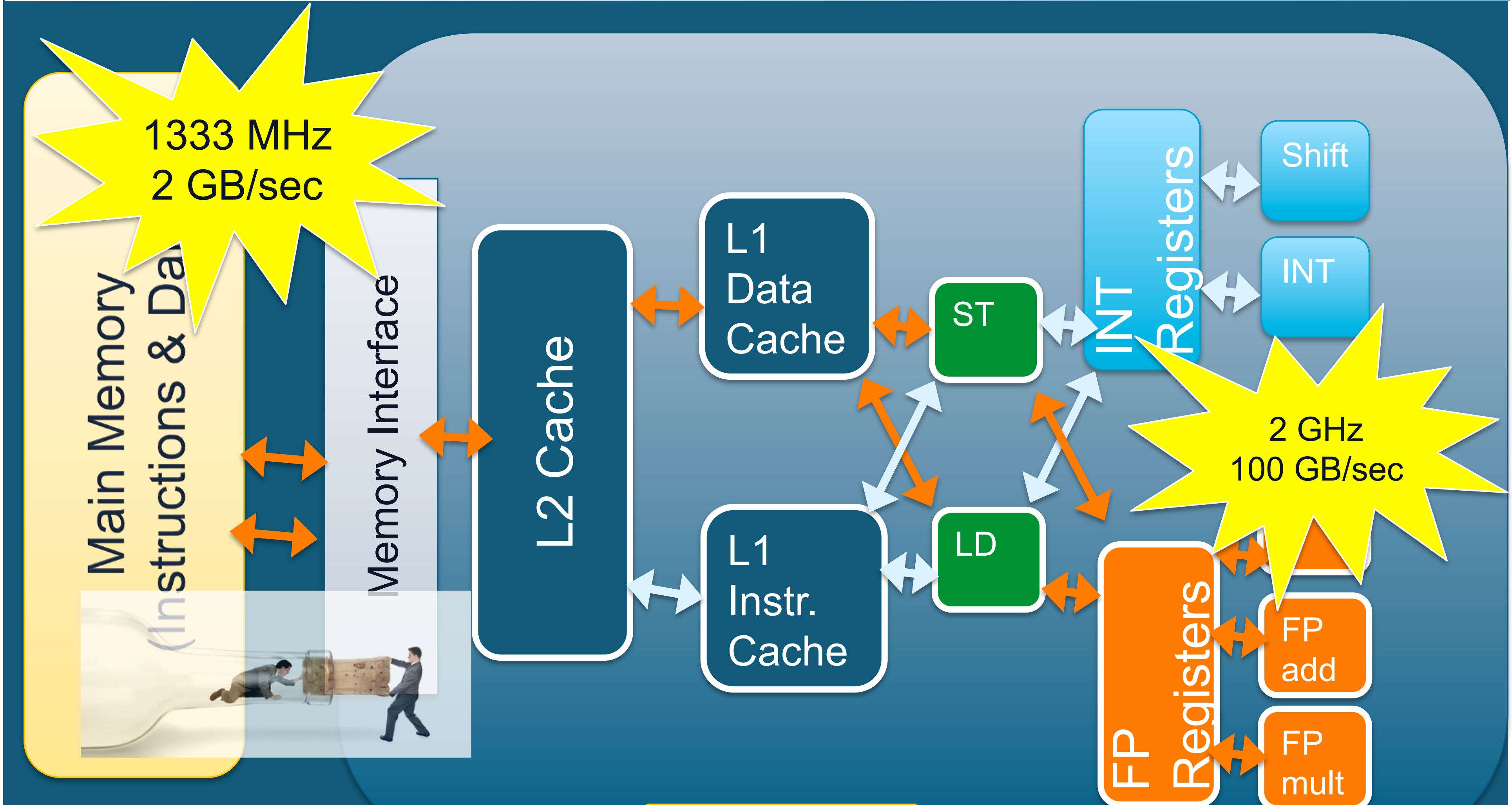


Simplified View

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**

Memory Bottleneck



1333 MHz
2 GB/sec

Main Memory
(Instructions & Da

Memory Interface

L2 Cache

L1
Data
Cache

L1
Instr.
Cache

ST

LD

INT
Registers

Shift

INT

FP
Registers

FP
add

FP
mult

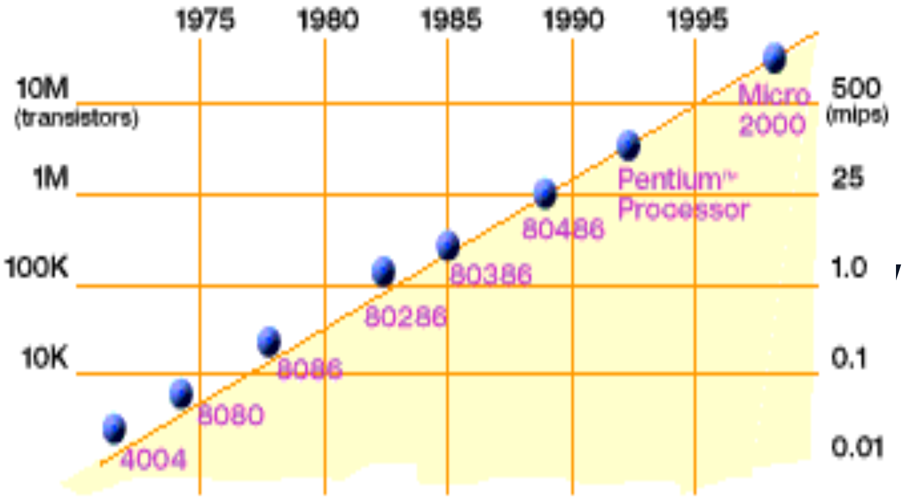
2 GHz
100 GB/sec

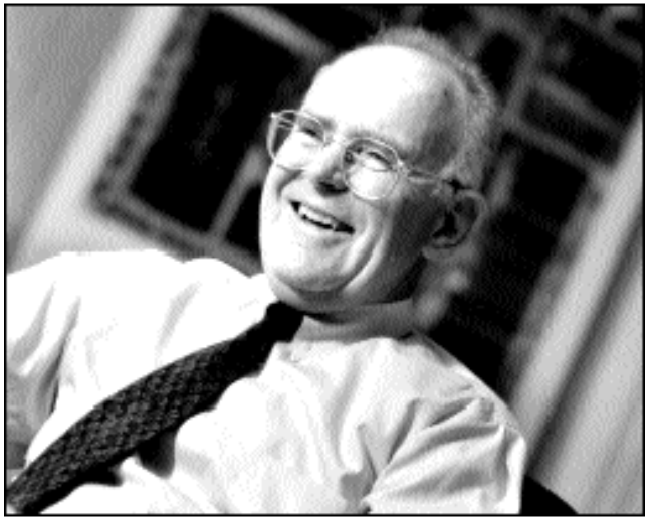
Simplified View

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)
- May not be “symmetric” wrt cores and functional units
- 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.

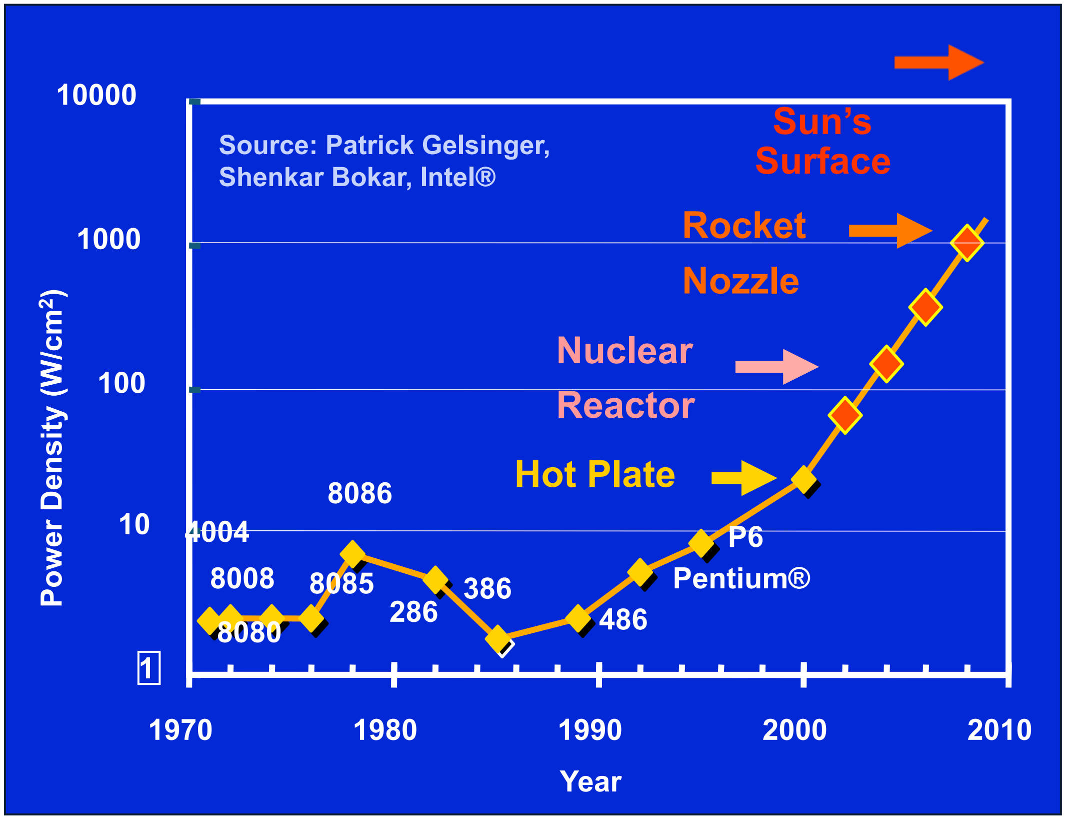
Slide source: Jack Dongarra

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB Lawrence Berkeley National Laboratory

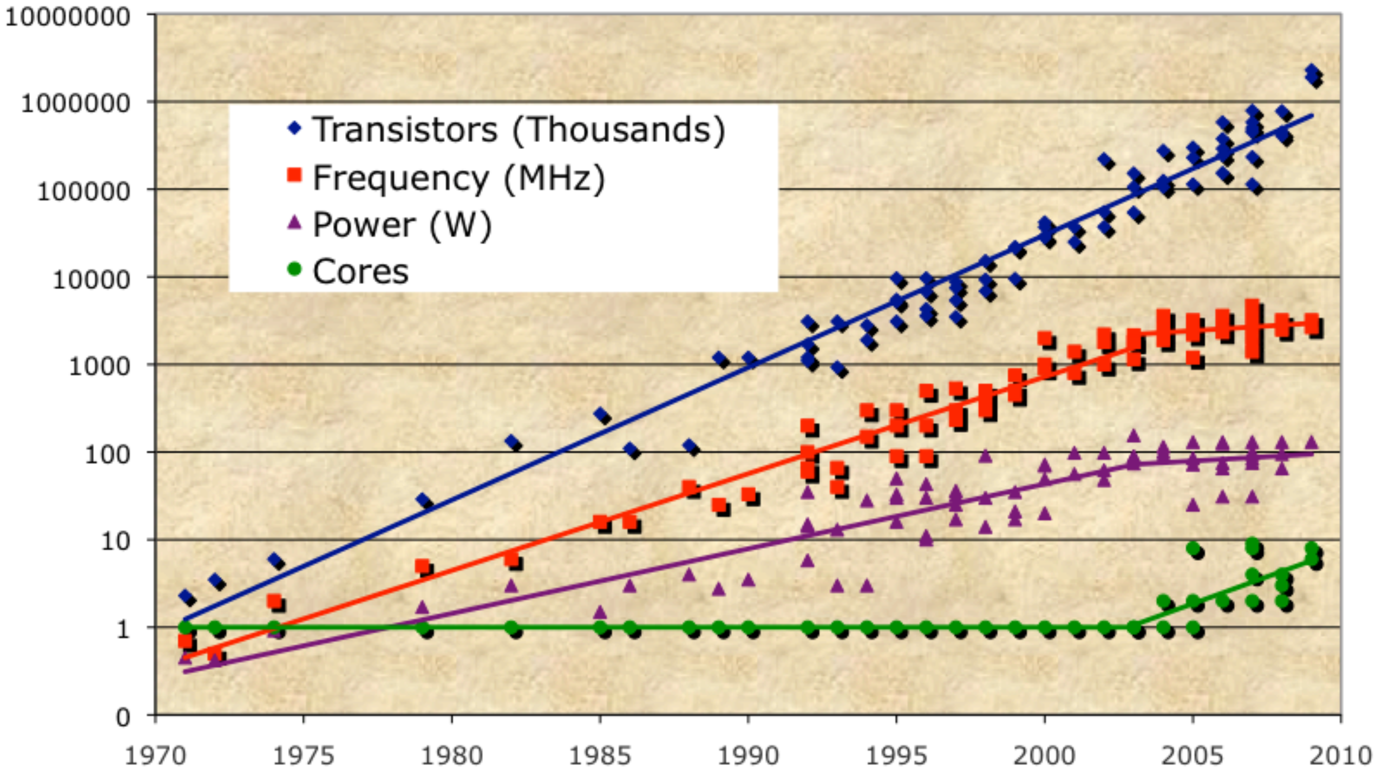
Power Density Limits Serial Performance

- 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 (actually, may not be happen)
- 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

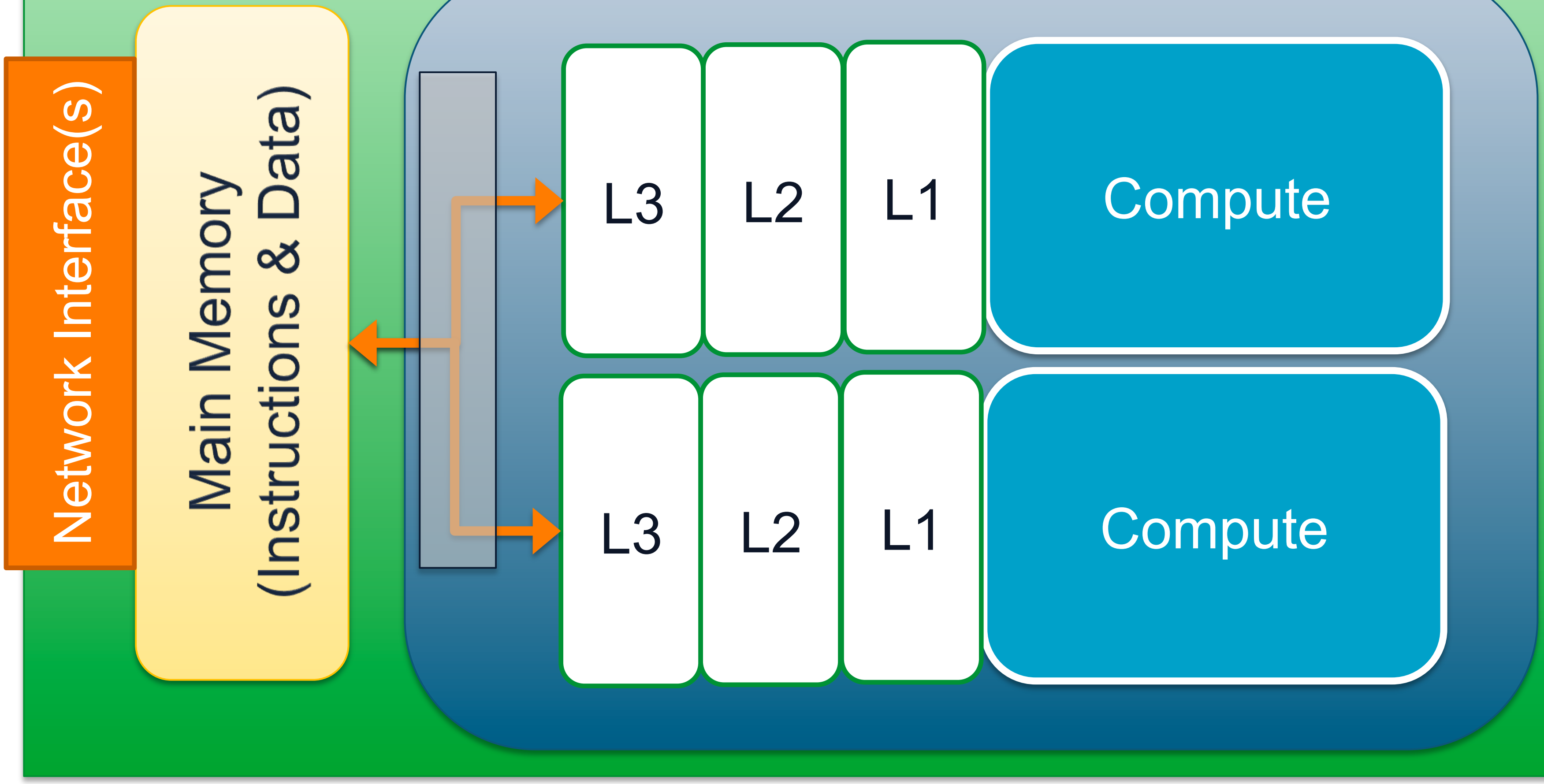
HPC Nodes

HPC Node

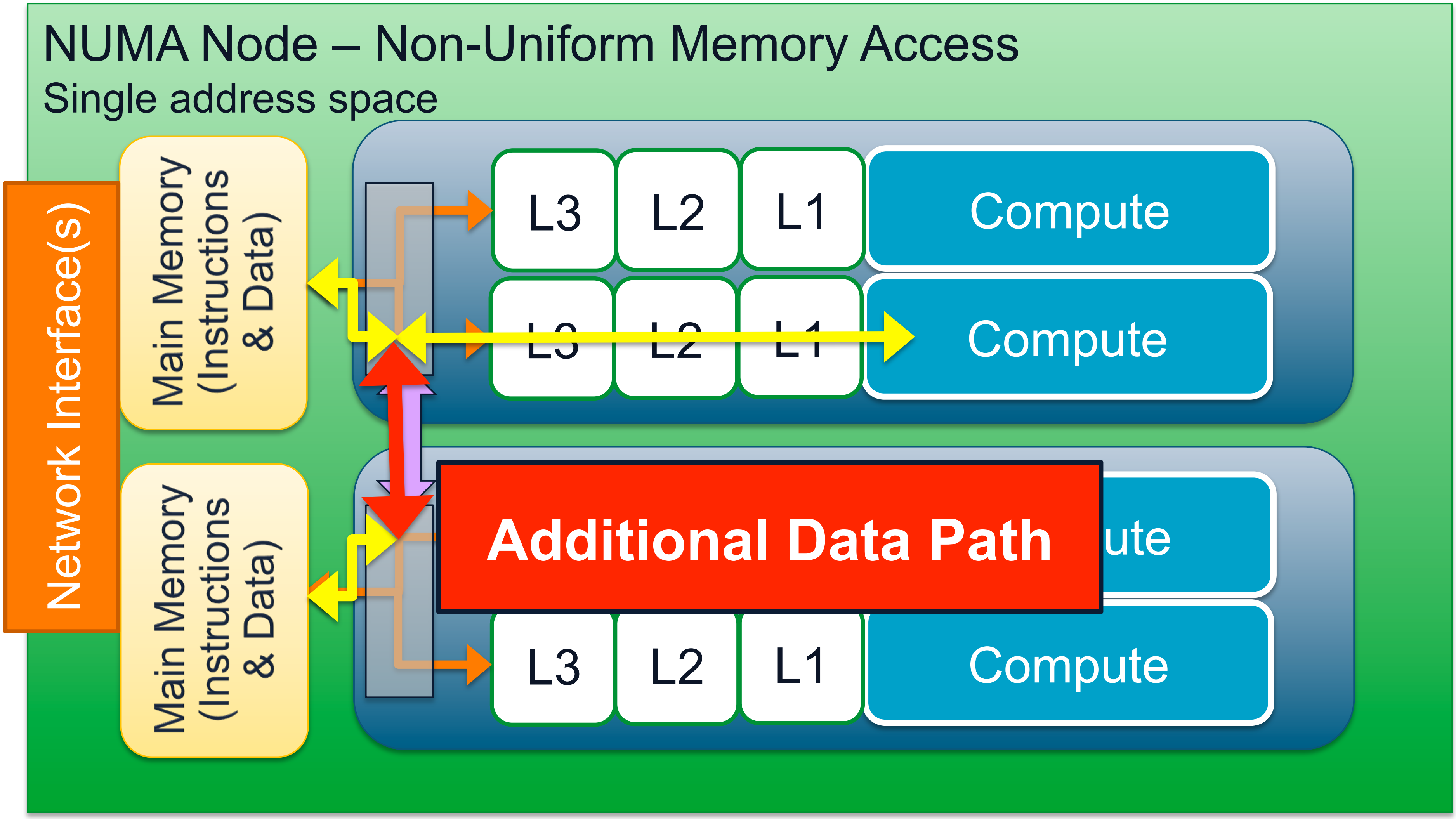
- **A “node” is a (physical) collection of CPUs, memory, and interfaces to other nodes and devices.**
  - Single memory address space
  - Shared memory pool
  - Memory access “on-node” is significantly faster than “off-node” memory access
  - Often called an “SMP node” for “Shared Memory Processing”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

Example SMP Node

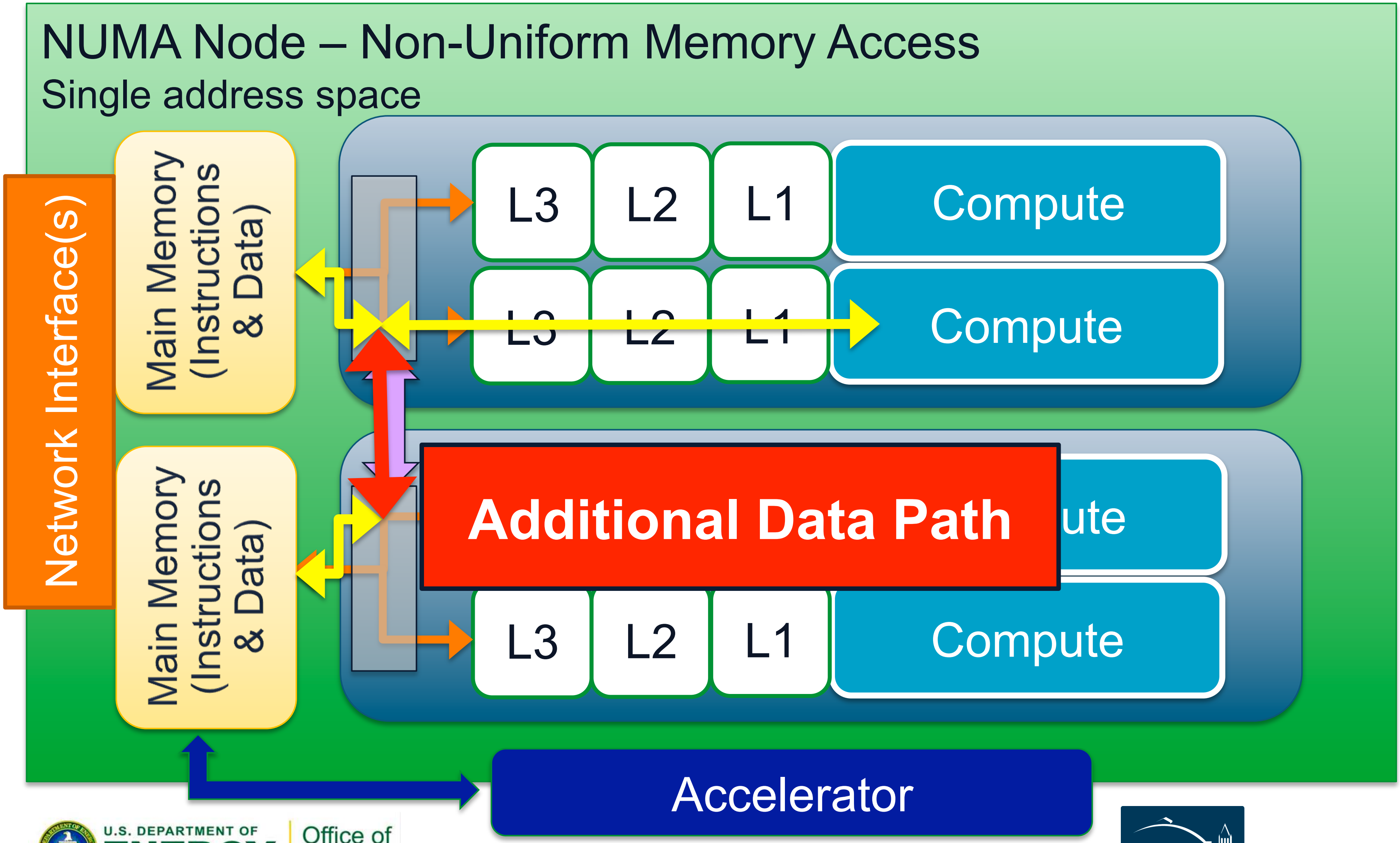
SMP Node – Each core has equal access to memory and cache



Example NUMA Node


NUMA Node – Non-Uniform Memory Access
Single address space
Network Interface(s)
Main Memory (Instructions & Data)
Main Memory (Instructions & Data)
L3
L2
L1
Compute
L3
L2
L1
Compute
ute
L3
L2
L1
Compute
Additional Data Path

Example NUMA Node



Internode Networks

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

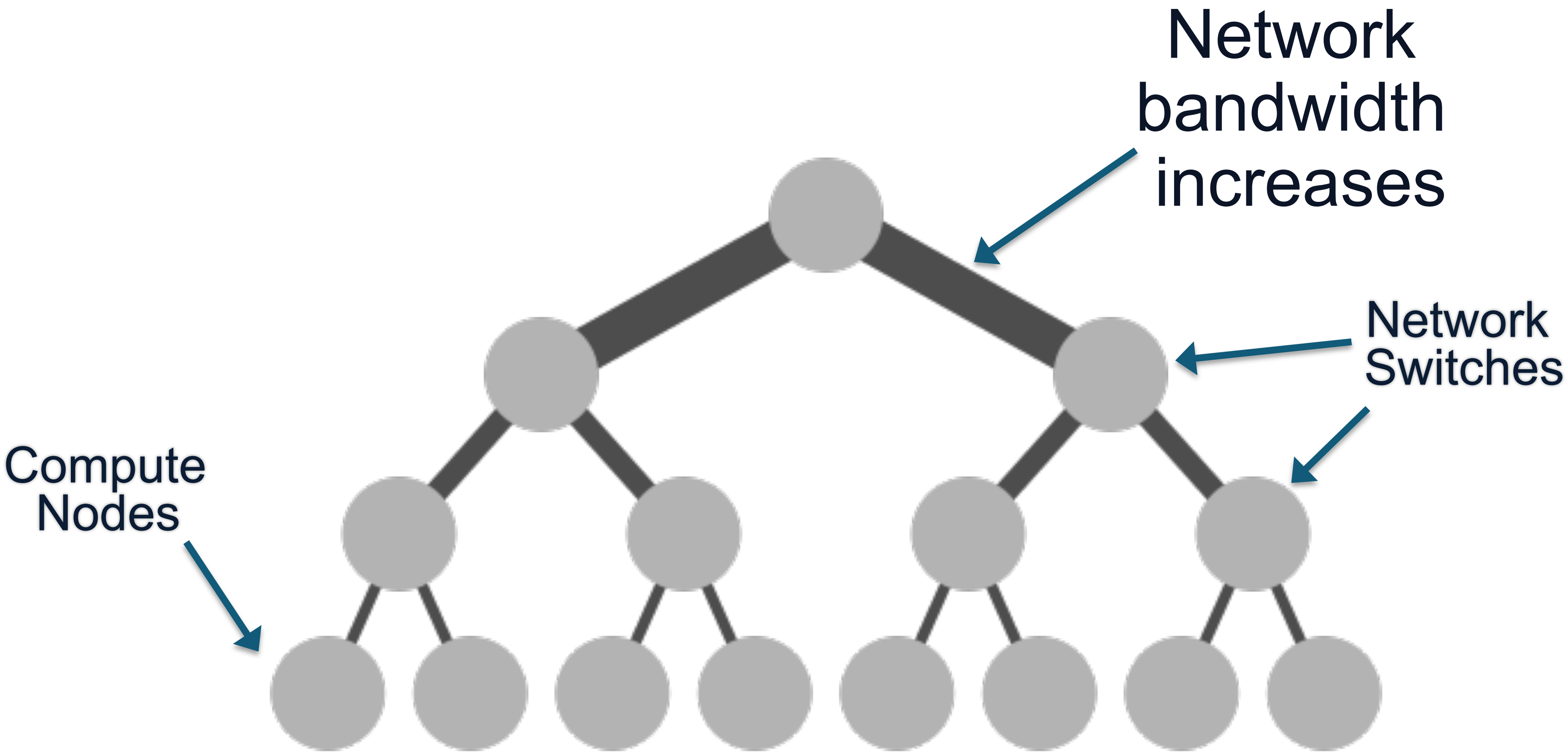
Examples

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)

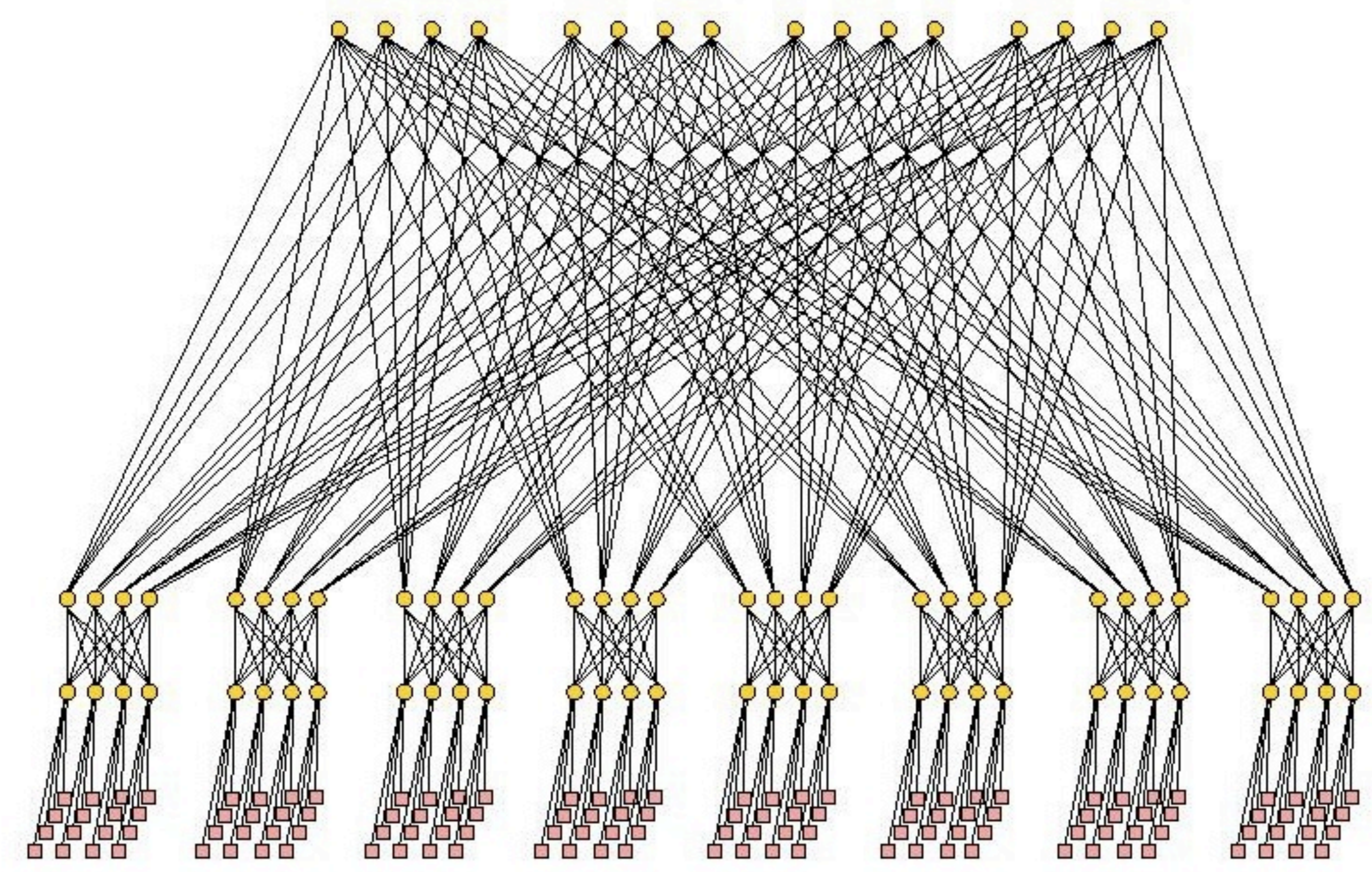
Fat Tree Switched Network


Network bandwidth increases

Network Switches

Compute Nodes

Implementation Can Be Complex

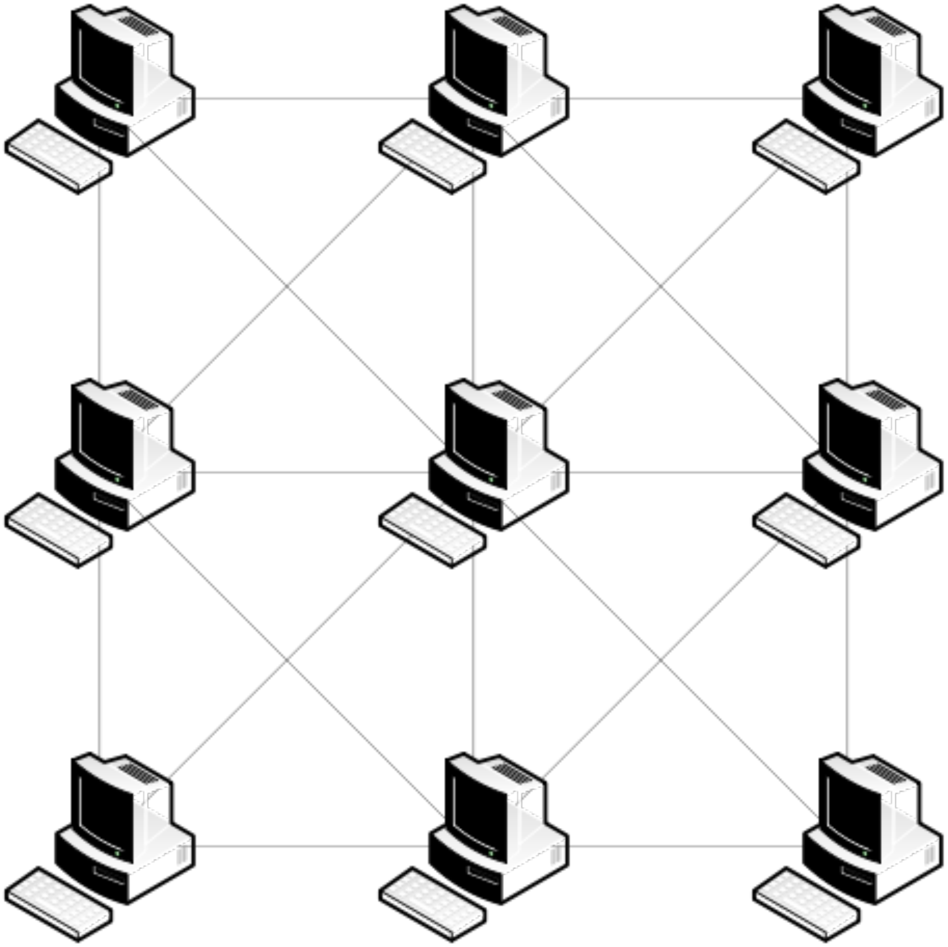


128-way fat tree

Mesh Networks

Mesh network topologies can be complex

Grids
Cubes
Hypercubes
Tori



Disk and Tape Storage

Largest and Slowest Memory

- **File storage is the slowest level in the data memory hierarchy**
  - Not uncommon for checkpoints / memory dumps to be taking a large fraction of total run time (>50%?)
  - NERSC users say they want no more than 10% of time to be IO
- **FLASH**
  - Non-volatile solid-state memory
  - Fast
  - Expensive
  - Some experimental systems with FLASH for fast IO

Archival Storage

- **Large, Permanent Storage**
  - Many PBs
  - Often tape storage fronted by a disk cache
  - HSM
    - Some systems may have Hierarchical Storage Management in place
    - Data automatically migrated to slower, larger storage via some policy
  - Often accessed via ftp, grid tools, and/or custom clients (e.g. hsi for HPSS)

HPC Operating Systems

- **Most HPC OSs are Linux-Based**
  - IBM AIX on POWER (also offers Linux)
- **“Generic” Cluster Systems**
  - Full Linux OS on each node
- **Specialized HPC Systems (e.g., Cray XT series, IBM Blue Gene)**
  - Full Linux OS on login, “services” nodes
  - Lightweight kernel on compute nodes
    - Helps performance
    - May hinder functionality (DLLs, dynamic process creation, some system calls may not be supported.)

Summary: Why Do You Care About Architecture?

- **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**

NERSC

National Energy Research
Scientific Computing Center

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB Lawrence Berkeley National Laboratory