Introduction to High Performance Computers

Richard Gerber
NERSC User Services
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
• **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
• **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.

…
The Five Main Parts of a Computer | eHow.com
May 5, 2010 ... The Five Main Parts of a Computer. Computers may look very different, but the components installed are standard. The major difference among ...
www.ehow.com › ... › Install a Hard Drive - Cached

Answers.com - What are five parts of the computer system
Computers question: What are five parts of the computer system? The five parts of the computer are CPU, Monitor, Printer, Mouse and Keyboard.
wiki.answers.com › ... › Technology › Computers - Cached - Similar

Answers.com - What are the main parts of computers
What are five main parts of a computer? ram cpu hard disk drive optical ...
wiki.answers.com › ... › Technology › Computers › Computer Hardware - Cached

What are the main parts of a computer?
What are the main component parts of a computer? ... a processor, and inputs and outputs. Most computers could be represented with these five “components”.
<table>
<thead>
<tr>
<th>eHow.com</th>
<th>Answers.com</th>
<th>Fluther.com</th>
<th>Yahoo!</th>
<th>Wikipedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
<td>CPU</td>
<td>Motherboard</td>
</tr>
<tr>
<td>RAM</td>
<td>Monitor</td>
<td>RAM</td>
<td>RAM</td>
<td>Power Supply</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>Printer</td>
<td>Storage</td>
<td>Power Supply</td>
<td>Removable Media</td>
</tr>
<tr>
<td>Video Card</td>
<td>Mouse</td>
<td>Keyboard/Mouse</td>
<td>Video Card</td>
<td>Secondary Storage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Monitor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motherboard</td>
<td>Keyboard</td>
<td>Motherboard</td>
<td>Motherboard</td>
<td>Sound Card</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Case / Power Supply</td>
<td></td>
<td>IO Peripherals</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• 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?
1. CPUs
2. Memory (volatile)
3. Nodes
4. Inter-node network
5. Non-volatile storage (disks, tape)
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

Interconnect

Node

Main Memory

CPU

Node

Main Memory

CPU
A distributed-memory HPC system

Interconnect

Node
- Main Memory
- CPU
- CPU

Node
- Main Memory
- CPU
- CPU
A distributed-memory HPC system

Interconnect

Node

Main Memory

CPU

CPU

Node

Main Memory

CPU

CPU
CPUs
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)
CPU (1 core)

Simplified View

FP Registers → FMA
FP Registers → FP add
FP Registers → FP mult

INT Registers ← Shift
INT Registers ← INT
CPU (1 core)

Main Memory (Instructions & Data)

FP Registers
- FP add
- FP mult

INT Registers
- INT

Shift

Simplified View
CPU (1 core)

Main Memory (Instructions & Data)

Memory Interface

L2 Cache

512 KB

INT Registers

FP Registers

FP add

FP mult

FMA

Shift

INT

Simplified View
CPU (1 core)

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

Simplified View
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
- …
• 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

<table>
<thead>
<tr>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 PM</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

- In this example:
  - Sequential execution takes $4 \times 90\text{ min} = 6\text{ hours}$
  - Pipelined execution takes $30 + 4 \times 40 + 20 = 3.5\text{ hours}$

- Bandwidth = loads/hour
  - $BW = \frac{4}{6}\text{ l/h w/o pipelining}$
  - $BW = \frac{4}{3.5}\text{ l/h w pipelining}$
  - $BW \leq 1.5\text{ l/h w pipelining}$, 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
• 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

100 ns

Main Memory
(Instructions & Data)

Memory Interface

L2 Cache

L1 Data Cache

L1 Instr. Cache

L1 St

FP Registers

ST

FD

INT Registers

FMA

FP add

FP mult

1 ns

10 ns

1 ns

Simplified View
Memory Bottleneck

**Simplified View**

- Main Memory (Instructions & Data)
- Memory Interface
- L2 Cache
- L1 Data Cache
- L1 Instr. Cache
- ST
- LD
- INT Registers
- FP Registers
- Shift
- INT
- 1333 MHz 2 GB/sec
- 2 GHz 100 GB/sec

**FP Operations**
- FP add
- FP mult

**INT Operations**
- INT

**Other Details**
- U.S. Department of Energy
- Office of Science
- Energy Sciences Computing Laboratory
• 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
• 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
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)

Main Memory (Instructions & Data)

L3 L2 L1 Compute

L3 L2 L1 Compute
Example HPC CPU (4 core)
Hypothetical Socket (8 core)
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

Network Interface(s)

Main Memory (Instructions & Data)

L3 L2 L1
Compute

L3 L2 L1
Compute

L3 L2 L1
Compute

L3 L2 L1
Compute
Example NUMA Node

NUMA Node – Non-Uniform Memory Access
Single address space

Main Memory (Instructions & Data)

Network Interface(s)

L3 L2 L1
Compute

L3 L2 L1
Compute

L3 L2 L1
Compute

L3 L2 L1
Compute

NUMA Node
Example NUMA Node

NUMA Node – Non-Uniform Memory Access
Single address space

Main Memory (Instructions & Data)
Main Memory (Instructions & Data)

Additional Data Path

Network Interface(s)

L3 L2 L1
L3 L2 L1
L3 L2 L1
L3 L2 L1

Compute
Compute
Compute
Compute
NUMA node (4 dies, 2 sockets)
Internode Networks
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
• **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
<table>
<thead>
<tr>
<th>Network</th>
<th>Bandwidth (GB/s)</th>
<th>Latency (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arista 10GbE (stated)</td>
<td>1.2</td>
<td>4.0</td>
</tr>
<tr>
<td>BLADE 10GbE (measured)</td>
<td>1.0</td>
<td>4.0</td>
</tr>
<tr>
<td>Cray SeaStar2+ (measured)</td>
<td>6.0</td>
<td>4.5</td>
</tr>
<tr>
<td>Cray Gemini (measured)</td>
<td>6.1</td>
<td>1.0</td>
</tr>
<tr>
<td>IBM (Infiniband) (measured)</td>
<td>1.2</td>
<td>4.5</td>
</tr>
<tr>
<td>SGI NumaLink 5 (measured)</td>
<td>5.9</td>
<td>0.4</td>
</tr>
<tr>
<td>Infiniband (measured)</td>
<td>1.3</td>
<td>4.0</td>
</tr>
<tr>
<td>Infinipath (measured)</td>
<td>0.9</td>
<td>1.5</td>
</tr>
<tr>
<td>Myrinet 10-G (measured)</td>
<td>1.2</td>
<td>2.1</td>
</tr>
</tbody>
</table>
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

e.g., Infiniband (IB)
Implementation Can Be Complex

128-way fat tree
Mesh networks can be complex.

- Grids
- Cubes
- Hypercubes
- Tori
4-D Hypercube
Disk and Tape Storage
• 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
• 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
<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Manufacturer</th>
<th>Computer</th>
<th>Country</th>
<th>Cores</th>
<th>Rmax [Tflops]</th>
<th>Power [MW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National SuperComputer Center in Tianjin</td>
<td>NUDT</td>
<td>Tianhe-1A</td>
<td>China</td>
<td>186,368</td>
<td>2,566</td>
<td>4.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NUDT TH MPP, Xeon 6C, Nvidia, FT-1000 8C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Oak Ridge National Laboratory</td>
<td>Cray</td>
<td>Jaguar</td>
<td>USA</td>
<td>224,162</td>
<td>1,759</td>
<td>6.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XT5, HC 2.6 GHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>National Supercomputing Centre in Shenzhen</td>
<td>Dawning</td>
<td>Nebulae</td>
<td>China</td>
<td>120,640</td>
<td>1,271</td>
<td>2.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GSIC, Tokyo Institute of Technology</td>
<td>NEC/HP</td>
<td>TSUBAME-2</td>
<td>Japan</td>
<td>73,278</td>
<td>1,192</td>
<td>1.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HP ProLiant, Xeon 6C, NVidia, Linux/Windows</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/LBNL/NERSC</td>
<td>Cray</td>
<td>Hopper</td>
<td>USA</td>
<td>153,408</td>
<td>1.054</td>
<td>2.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XE6, 6C 2.1 GHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Commissariat l'Energie Atomique (CEA)</td>
<td>Bull</td>
<td>Tera 100</td>
<td>France</td>
<td>138,368</td>
<td>1,050</td>
<td>4.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bull bullx super-node S6010/S6030</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>DOE/NNSA/LANL</td>
<td>IBM</td>
<td>Roadrunner</td>
<td>USA</td>
<td>122,400</td>
<td>1,042</td>
<td>2.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BladeCenter QS22/LS21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>University of Tennessee</td>
<td>Cray</td>
<td>Kraken</td>
<td>USA</td>
<td>98,928</td>
<td>831.7</td>
<td>3.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XT5 HC 2.36GHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Forschungszentrum Juelich (FZJ)</td>
<td>IBM</td>
<td>Jugene</td>
<td>Germany</td>
<td>294,912</td>
<td>825.5</td>
<td>2.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Blue Gene/P Solution</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>DOE/NNSA/LANL/SNL</td>
<td>Cray</td>
<td>Cielo</td>
<td>USA</td>
<td>107,152</td>
<td>816.6</td>
<td>2.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XE6, 6C 2.4 GHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• 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??????
Projected Performance Development

- 1 Gflop/s
- 1 Ttroll/s
- 100 Mflop/s
- 100 Gflop/s
- 100 Tflop/s
- 100 Pflop/s
- 1 Pflop/s
- 1 Eflop/s

Graph showing projected performance development from 1994 to 2020 with different performance metrics.
National Energy Research Scientific Computing Center
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

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
• 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