What Every Computational Physicist Should Know About Computer Architecture*

*Plus some other useful technology

Ian Cosden, Princeton University

Steve Lantz, Cornell University

CoDaS-HEP Summer School, August 1, 2022
Outline

• Motivation
• HPC Cluster
• Compute
• Memory
• Disk
• Other Architectures
Why Care About the Hardware?

• Get the most out of this week
• Be able to ask better questions and understand the answers
• Architecture changes can force new design and implementation strategies
  – Prevent programming mistakes
  – Make educated decisions
• Exploit the hardware to your advantage (performance)
Hardware is the Foundation

User
- Application

Program
- Programming Language
- Programming Models
- Compiler

System
- Runtime Libraries
- OS support for resource management (threading, memory, scheduling)

Hardware
- Shared Memory
- CPU
- CPU
- Disk
- Accelerators
- Network
Why Care About HPC Hardware?

• High-Performance Computing (HPC)
  – Aggregate computing power in order to deliver far more capacity than a single computer
  – Supercomputers

• Some problems demand it from the onset; many problems evolve to need it
  – When you outgrow your laptop, desktop, departmental server
  – Ask: do you need greater capability, or just capacity?
  – Either way, HPC clusters run lots of High-Throughput Computing (HTC) jobs
Moore’s Law

- Number of transistors doubles every 18-24 months
- More transistors = better, right?
The Free Lunch is Over

40 Years of Microprocessor Trend Data

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Chukutuk, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp
https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/
Top 500 Supercomputers

https://www.nextplatform.com/2016/06/20/china-topples-united-states-top-supercomputer-user/

https://www.top500.org
Why Talk About Intel Architecture?

https://en.wikipedia.org/wiki/TOP500
Outline

• Motivation
• HPC Cluster
• Compute
• Memory
• Disk
• Other Architectures
• A cluster is just connected collection of computers called nodes
• The high-speed network that connects the cluster is called the interconnect
• A single, unconnected node is usually just called a server
• Entire clusters, as well as stand-alone servers, are sometimes referred to as machines
HPC Cluster: the vast majority of HPC systems
- A collection of servers connected to form a single entity
  - Each is essentially a self-contained computer
  - OS with CPU, RAM, hard drive, etc.
- Stored in racks in a dedicated machine room
- Networked together via low-latency interconnect
  - Ethernet < InfiniBand, Omni-Path (Intel)
- Interconnect extends to shared storage

SMP System: Symmetric Multi-Processing
- CPUs all share memory – essentially one computer
- Expensive, likely needed for a unique purpose
  - Huge memory, huge OpenMP jobs
Nodes Are Discrete Systems

• Memory address space
  – Range of unique addresses
• Separate for each node
• Can’t just access all memory in cluster

Each node has a separate address space
Clusters With Accelerators

- “Heterogeneous” system architecture
- Accelerators: GPUs, typically
  - Or FPGAs, etc.
  - Node can have multiple accelerators
- Programmable with MPI + x
  - Where x = OpenMP, OpenACC, CUDA, ...
- Increases computational power
  - Increases FLOPS / Watt
- Trending in Top500.org
  - Strong shift toward systems with accelerators started ~10 years ago
Outline

• Motivation
• HPC Cluster
  • Compute
  • Memory
  • Disk
• Other Architectures
Multicore Processors

• Nearly all modern processors are multicores
  – In servers: Intel Xeon, AMD EPYC, IBM Power
  – In laptops: Intel, AMD, ARM
  – In GPUs, phones, and mobile devices as well
• Most have vector units, multiple cache levels
• Require special care to program
  – Multi-threading, vectorization, ...
  – The free lunch is over!
• Instruction sets can vary
  – Intel and AMD: x86_64 base, not identical
  – IBM Power, ARM, NVIDIA, etc. all differ
• Typical nodes have 1, 2, or 4 processors
  – 1: laptop/desktops/servers
  – 2: servers/clusters
  – 4 (or more): high end special use
• Each processor is attached to a socket
• Each processor has multiple cores
• CPU can refer to a core... or a processor
• Processors connected via interconnect
• Memory attached to processor/socket
  – Programmer sees single node memory address space
Symmetric MultiProcessing (SMP) – Memory access is uniform, as all traffic goes via north bridge, but is bottlenecked by front side bus (FSB) and north-bridge-to-RAM bandwidth limitations. Current motherboards have no north bridge; the south bridge is now referred to as the I/O hub or just the chipset.

Non-Uniform Memory Access (NUMA) – Physical memory is partitioned per CPU, with a fast interconnect to link CPUs to each other and to I/O. This removes bottlenecks but memory is no longer uniform – there may be 30-50% extra latency to access remote memory, more in multi-hop systems.
“System on a Chip” (SoC)

- Core: “A complete ensemble of execution logic, and cache storage as well as register files plus instruction counter (IC) for executing a software process or thread.” (Jarp)

UPI – Intel Ultra Path Interconnect
PCIe – Peripheral Component Interconnect Express
DDR – Double Data Rate

https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)#High_Core_Count._28HCC.29
Skylake Core Microarchitecture

Key Components:

- Control logic
- Register file
- Functional Units
  - ALU (arithmetic and logic unit)
  - FPU (floating point unit)
- Data Transfer
- Load / Store

Too much!
Vectorization Terminology

• New generations of CPUs add capabilities and instructions
  – Each core can have one or more vector processing units (VPUs) and vector registers
  – Machine (assembly) commands are often added to take advantage of new hardware

• Vector instructions are also called SIMD = Single Instruction Multiple Data
  – x86_64 has Streaming SIMD Extensions (SSE), Advanced Vector Extensions (AVX)

```
for (int i=0; i<N; i++) {
    c[i]=a[i]+b[i];
}
```

Scalar (SISD)

At inner-most loop level

Vector (SIMD)

One operation
One result

One operation
Multiple results
HPC Performance Metrics

- **FLOP** = Floating-point **Operation**
  - FLOPs = Floating-point **Operations**
  - FLOPS = Floating-point **Operations Per Second**
  - Often write FLOP/s or FLOP/sec to avoid ambiguity

- **Calculation for single processor**
  - Typically Double Precision (64-bit)
  - FLOPS = (Clock Speed)*(Cores)*(FLOPs/cycle)

- **FLOPS/$**, **FLOPS/Watt** are useful metrics too

- **Bandwidth metrics** (“giga” usually means $10^9$)
  - GB/s = gigabytes/sec
  - Gb/s = gigabits/sec

---

N.B. - not to be confused with a Hollywood flop!
Intel Xeon (Server) Architecture Codenames

- Number of FLOP/s depends on width, number, and abilities of VPUs in a core
- Per-core rates, double precision (64-bit doubles – twice this for 32-bit floats)
  - Nehalem/Westmere (SSE):
    - 4 DP FLOPs/cycle: 128-bit addition + 128-bit multiplication
  - Sandy Bridge/Ivy Bridge (AVX)
    - 8 DP FLOPs/cycle: 256-bit addition + 256-bit multiplication
  - Haswell/Broadwell (AVX2)
    - 16 DP FLOPs/cycle: two, 256-bit FMA, fused multiply-add
  - KNL/Skylake (AVX-512)
    - 32 DP FLOPs/cycle: two*, 512-bit FMA
  - Cascade Lake (AVX-512 VNNI with INT8)
    - 32 DP FLOPs/cycle: two, 512-bit FMA

*Some Skylake-SPs can do only one 512-bit FMA = (a × b + c), per core
Outline

• Motivation
• HPC Cluster
• Compute
  • Memory
    – Layout
    – Cache
    – Performance
• Disk
• Other Architectures
**Memory Hierarchy**

### Dual Socket Intel Xeon CPU

- **CPU 1**
  - Cores: Core 1, Core 2, Core n
  - Caches: L1, L2, L3
  - Main Memory (DRAM)

- **CPU 2**
  - Cores: Core 1, Core 2, Core n
  - Caches: L1, L2, L3
  - Main Memory (DRAM)

- **UPI (Uniform Memory Access)**

### Table: Memory Hierarchy

<table>
<thead>
<tr>
<th></th>
<th>Registers</th>
<th>L1 Cache</th>
<th>L2 Cache</th>
<th>L3 Cache</th>
<th>DRAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speed</strong></td>
<td>1 cycle</td>
<td>~4 cycles</td>
<td>~10 cycles</td>
<td>~30 cycles</td>
<td>~200 cycles</td>
<td>10ms</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>&lt; KB per core</td>
<td>~32 KB per core</td>
<td>~256 KB per core</td>
<td>~35 MB per socket</td>
<td>~100 GB per socket</td>
<td>TB</td>
</tr>
</tbody>
</table>
Growth in Memory Performance Lags CPU

If your data is served by RAM and not caches it doesn’t matter if you have vectorization.

To increase performance, try to minimize memory footprint.

http://web.sfc.keio.ac.jp/~rdv/keio/sfc/teaching/architecture/architecture-2008/hennessy-patterson/Ch5-fig02.jpg
• Motivation
• HPC Cluster
• Compute
• Memory
• Disk
  – Types
  – Filesystems
• Other Architectures
“A supercomputer is a device for turning compute-bound problems into I/O-bound problems.”

-- Ken Batcher, Emeritus Professor of Computer Science at Kent State University
Types of Disk

• Hard Disk Drive (HDD)
  – Traditional Spinning Disk
• Solid State Drives (SSD)
  – ~5x faster than HDD
• Non-Volatile Memory Express (NVMe)
  – ~5x faster than SSD
Types of Filesystems - NFS

• NFS – Network File System
  – Simple, cheap, and common
  – Single filesystem across network
  – Not well suited for large throughput

![Diagram showing a network file system setup with a head node, several compute nodes, and storage devices](image)

≤ 8 storage devices
Parallel Filesystems

• GPFS (General Parallel Filesystem – IBM)
  – Designed for parallel read/writes
  – Large files spread over multiple storage devices
  – Allows concurrent access
  – Significantly increase throughput

• Lustre
  – Similar idea
  – Different implementation

• Parallel I/O library in software
  – Necessary for performance realization
Outline

• Motivation
• HPC Cluster
• Compute
• Memory
• Disk
• Other Architectures
  – GPUs
  – Xeon Phi
  – Cloud
Heterogeneous Systems on the Rise

• The computing landscape changes rapidly
• 60% of TOP500 systems now have accelerators (June 2022)
• Graphics processing units (GPUs) are by far the most common type of accelerator

https://www.top500.org/statistics
General Purpose GPUs

- General Purpose GPU (GPGPU)
  - NVIDIA Pascal/Volta/Ampere...
  - AMD Radeon Instinct, others
- Always attached to a CPU host
  - Data travels to device over PCIe
  - NVIDIA NVLink has higher speed
- Not compatible with x86
- Programmable sort-of-like CPUs
  - NVIDIA CUDA: extensions to C++
  - OpenCL, OpenACC, OpenMP, ...
  - Need compatible compiler, libraries
NVIDIA Tesla V100 (V for “Volta”)

- 80 Streaming Multiprocessors (SMs)
  - Analogous to CPU cores
  - Execute only SIMD-type instructions
- One SM holds many “CUDA cores”
  - Analogous to vector lanes
  - 64 FP32 CUDA cores
  - 64 INT32 CUDA cores
  - 32 FP64 CUDA cores
- One SM also has 8 tensor cores
  - Fast 4x4 matrix multiplications in FP16
- Device memory: 32 GB of HBM2
NVIDIA Execution Model

- **SIMT = Single Instruction Multiple Thread**
  - Operations apply to “warps” of 32 threads
  - A warp of threads is like a CPU vector
  - Not all that different from SIMD

- **CUDA core = scalar (thread) processor**
  - Instructions only go to groups of 32
  - A thread block must be split into warps of 32
  - Thus, an SM is like 1 or 2 vector units in a CPU

- One thread block maps to one SM

- A grid of thread blocks can be spread over the entire device

[Image of NVIDIA Execution Model diagram]
R.I.P. Intel Xeon Phi, a.k.a. Intel MIC

• MIC: Many Integrated Cores
  – x86-compatible multiprocessor architecture
• Programmable just CPUs
  – MPI, OpenMP, OpenCL, ...
  – Same compilers as CPUs
• But many cores, fast MCDRAM like GPUs
  – Intel’s “answer” to the GPU challenge
• Started out as a coprocessor
  – Data travels over PCIe
  – Became main processor in 2nd generation
• Eventually merged into Xeon line
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores</td>
<td>4</td>
<td>28</td>
<td>72</td>
<td>84 SMs</td>
</tr>
<tr>
<td>Logical Cores</td>
<td>8</td>
<td>56</td>
<td>288</td>
<td>5120 cores</td>
</tr>
<tr>
<td>Clock rate</td>
<td>3.6 – 3.7 GHz</td>
<td>2.5 – 3.8 GHz</td>
<td>1.5-1.7 GHz</td>
<td>1530 MHz</td>
</tr>
<tr>
<td>Theoretical GFLOPS (double)</td>
<td>460.8</td>
<td>2240</td>
<td>3456</td>
<td>7450</td>
</tr>
<tr>
<td>SIMD width</td>
<td>512 bit</td>
<td>512 bit</td>
<td>512 bit</td>
<td>Warp of 32 threads</td>
</tr>
<tr>
<td>Memory</td>
<td>--</td>
<td>--</td>
<td>16 GB MCDRAM</td>
<td>32 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>384 GB DDR4</td>
<td></td>
</tr>
<tr>
<td>Memory B/W</td>
<td>127.8 GB/s</td>
<td>127.8 GB/s</td>
<td>400+ GB/s MCDRAM</td>
<td>900 GB/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>115.2 GB/s RAM</td>
<td></td>
</tr>
<tr>
<td>Approx. Unit Price</td>
<td>$1,200</td>
<td>$12,000</td>
<td>$3,400</td>
<td>$9,000</td>
</tr>
</tbody>
</table>
• “Regular” computers, just somewhere else
• Provide users with remote virtual machines or containers
• Can be used for anything:
  – Mobile services, website hosting, business applications, ...
  – Data analysis, high performance computing
• Providers
  – Amazon Web Services (AWS)
  – Microsoft Azure
  – Google Cloud Platform (GCP)
  – Oracle Cloud
  – Lots of others
Cloud Computing Pros and Cons

• Advantages:
  – Potentially lower cost
    • Pay as you go
    • Save on sysadmins and infrastructure
    • Economy of scale
  – Scaling up or down as needed
    • Can be used to run overflow from a regular data center
  – Access to a wide range of hardware

• Challenges:
  – Data movement
    • Expensive and time consuming
  – Security, privacy, ...
What We Didn’t Talk About

• Microarchitecture details (ALU, FPU, pipelining...)
• Memory bandwidth and latency
• Cache lines and cache coherence
• IBM (Power) or AMD processors
• FPGAs, Google TPUs, NPUs, ...
• Very nice glossary: https://cvw.cac.cornell.edu/main/glossary
• NVIDIA Volta Architecture Whitepaper https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf