Computational and Data Science Challenges

45’ minutes of jargon to shine in society

Matthieu Lefebvre
Princeton University

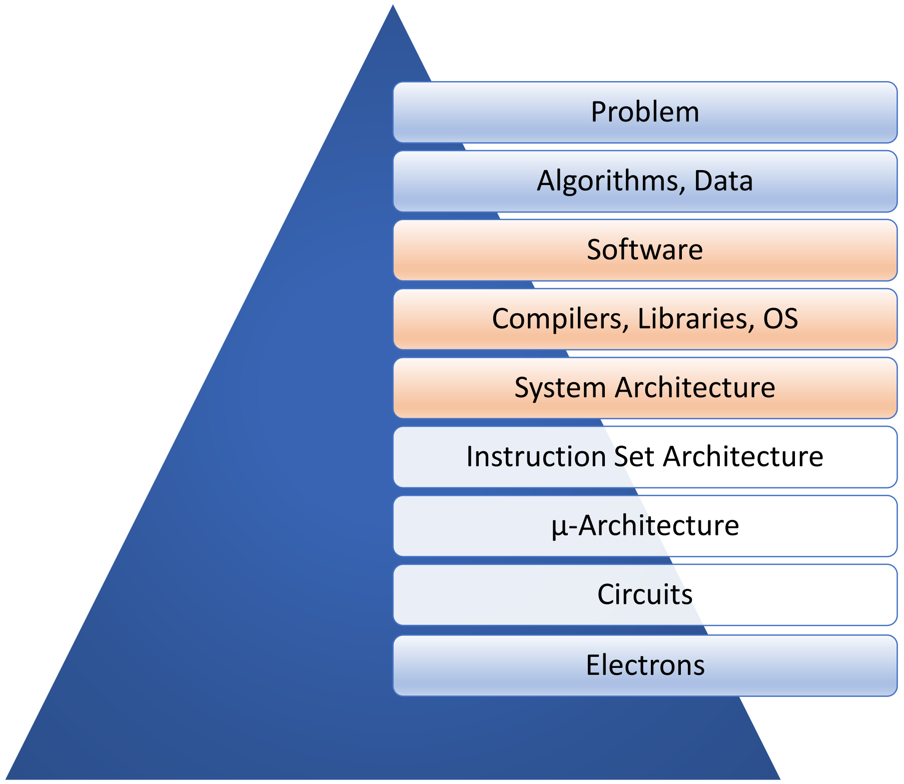
Monday, 10 July 2017

First Computational and Data Science school for HEP (CoDaS-HEP)

Outline

- Motivation
- Performance Metrics
- Architecture
  - General
  - Compute
  - Memory
- Emerging Architectures
- Supercomputers & Cloud Computing

Ecosystem

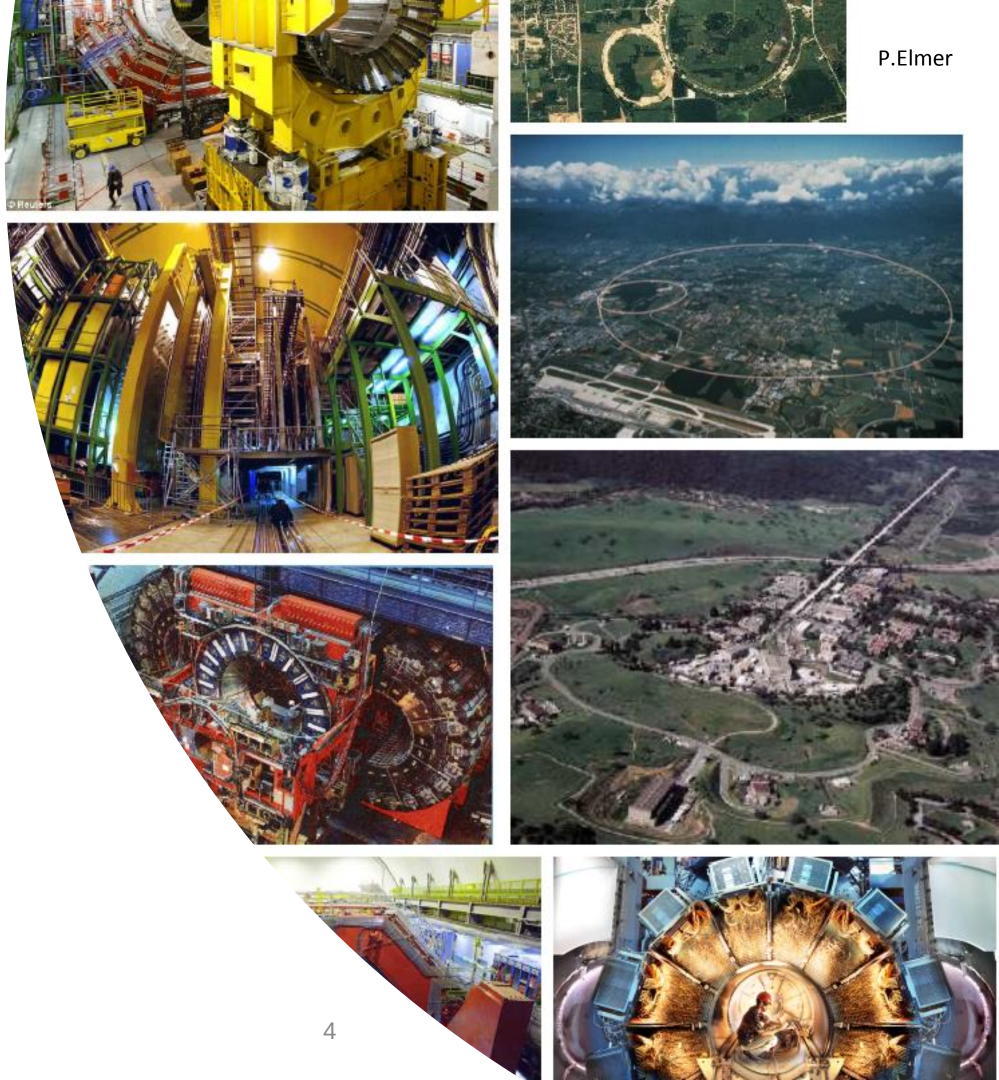


Y.Patt / S.Jarp

High-Energy Physics: The Problem

- Embarrassingly parallel:
  - Each *event* can be computed completely independently
  - No communications between events
  - Can be launched in separated processes

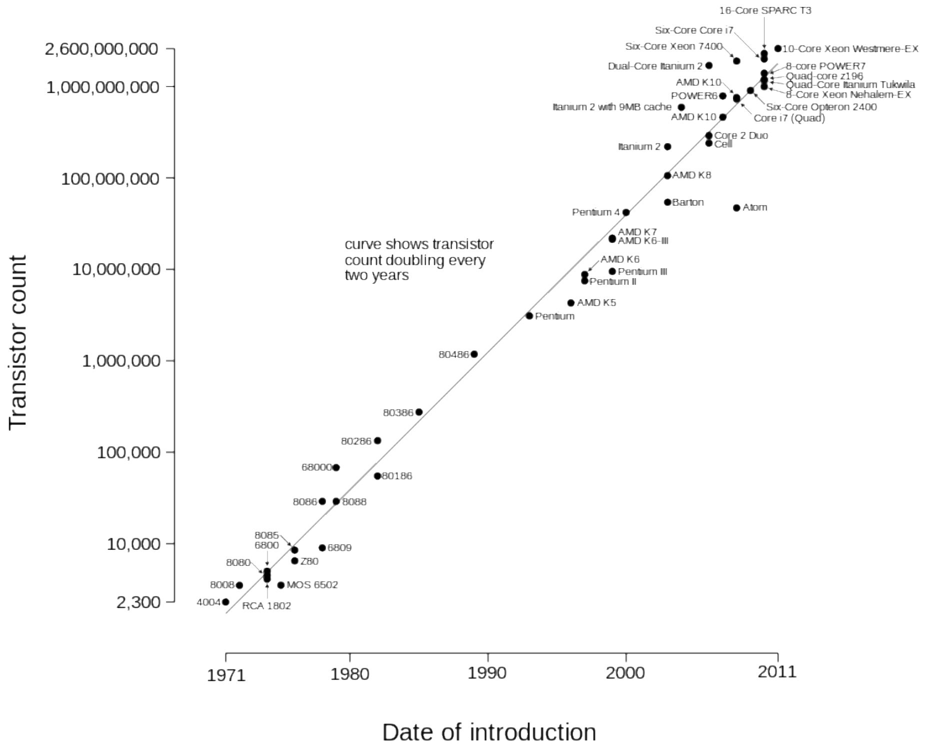
- Why are you here then?
  - Is there additional sources of parallelism to be found *inside* events processing?
  - Why do we need more parallelism?



Moore’s Law

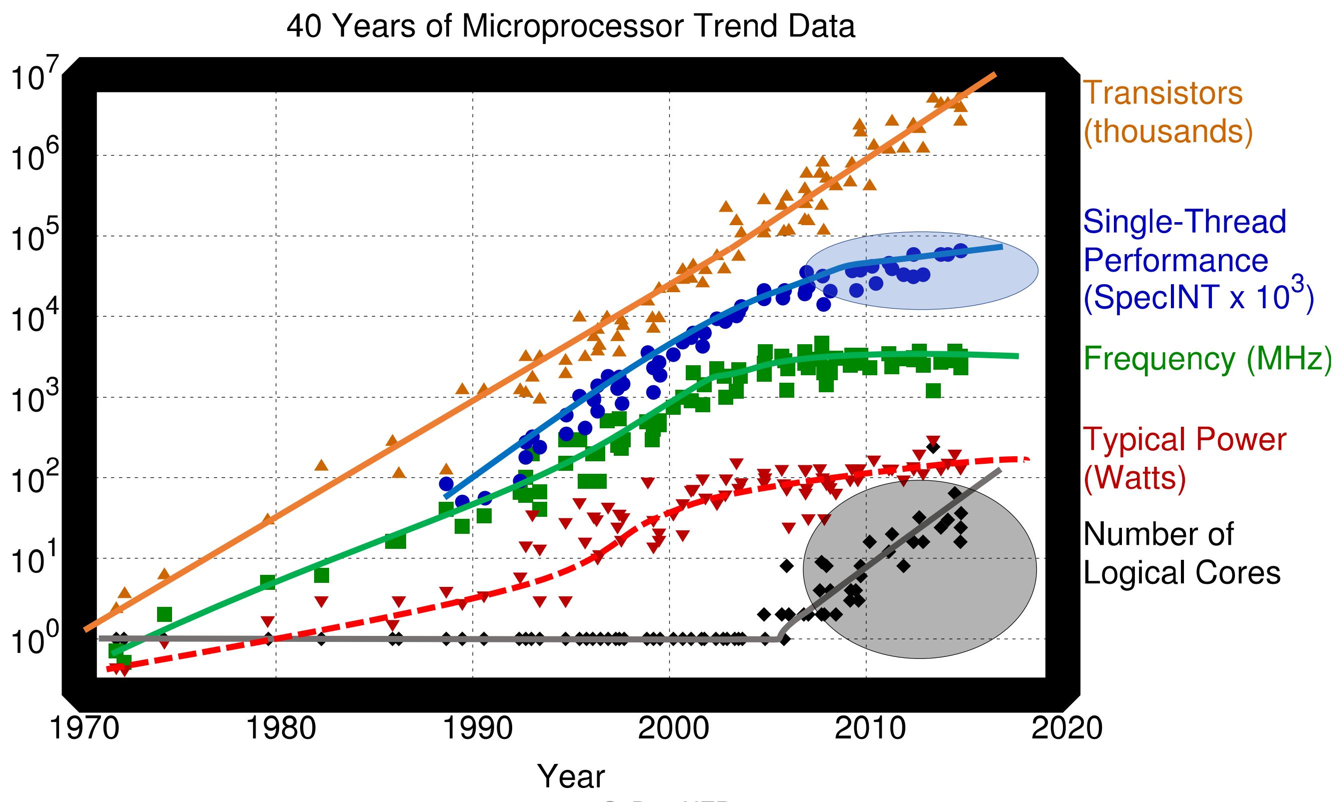
- Cost of transistors drop overtime
  - Number of transistors double every 18 months / 24 months
- More transistors == better performance?
- How these transistors are deployed to achieve ever greater (exponential) growth in application performance?

Microprocessor Transistor Counts 1971-2011 & Moore's Law



☞ https://en.wikipedia.org/wiki/Moore%27s_law

"The Free Lunch is Over"



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Performance Metrics

- FLOP = Floating-point Operation
- FLOPS = Floating-point Operations Per Second

- FLOPS/$
- FLOPS/Watt
- Bandwidth: GB/s

- HEP Specific: Events/s, Event/Watt, Event/$?
- Pick a metric relevant to what you are trying to achieve

UNSPEAKABLE HORRORS FROM OUTER SPACE PARALYZE THE LIVING AND RESURRECT THE DEAD!
PLAN 9 FROM OUTER SPACE
with
BELA LUGOSI
VAMPIRA
LYLE TALBOT
A J. Edward Reynolds Production
Produced and Directed by Edward D. Wood, Jr.
DCA

Comparing Performance

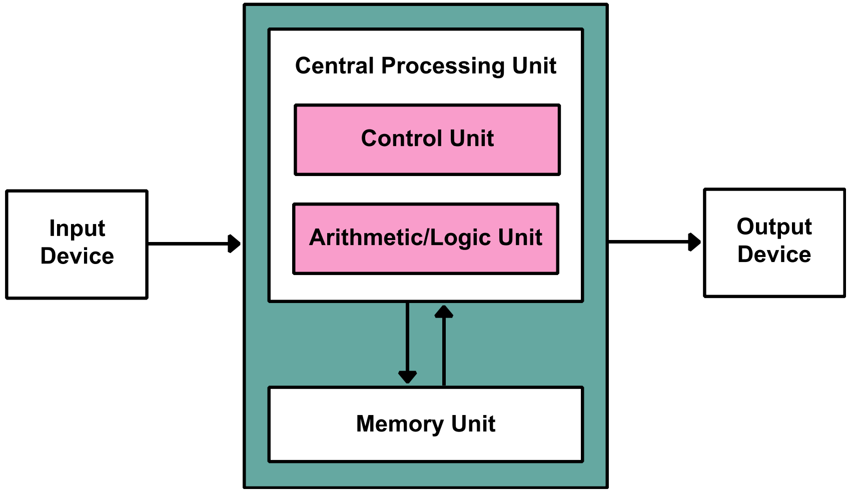
- *David H. Bailey. "Highly parallel perspective**: Twelve ways to fool the masses when giving performance results on parallel computers**". Supercomputing Review, 4(8):54-55, August, 1991. ISSN: 1048-6836. Also appears as NASA Ames RNR Technical Report RNR-91-020.*
- Adapted to GPU computing
  - https://www.hpcwire.com/2011/12/13/ten_ways_to_fool_the_masses_when_giving_performance_results_on_gpus/

  1. Quote performance results only with 32-bit floating-point arithmetic, not 64-bit arithmetic.
  2. **Don't time data movement or kernel-invocation overhead.**
  3. Quote GPU cost and ubiquity for low-end parts. Measure performance on high-end parts.
  4. Quote memory bandwidth only to/from on-board GPU memory, not to/from main memory
  5. Disable ECC checks on memory
  6. **Compare full (or even multiple) GPU performance to a single CPU core.**
  7. **Compare heavily optimized GPU code to unoptimized CPU code**
  8. Scale the problem size to fit within GPU memory.
  9. Sacrifice meaningful numerics for GPU performance.
  10. Select algorithms that favor GPUs.

- 100x speedups are suspicious…

Von Neumann Architecture

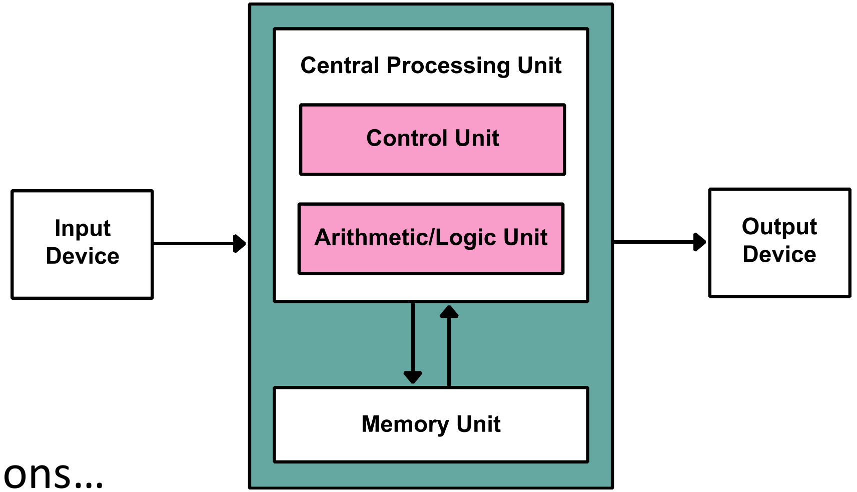
- Separate Memory and Processing Units
- Memory unit holds Data **AND** Instructions
- Input and Output are the interaction with the “outside-world”



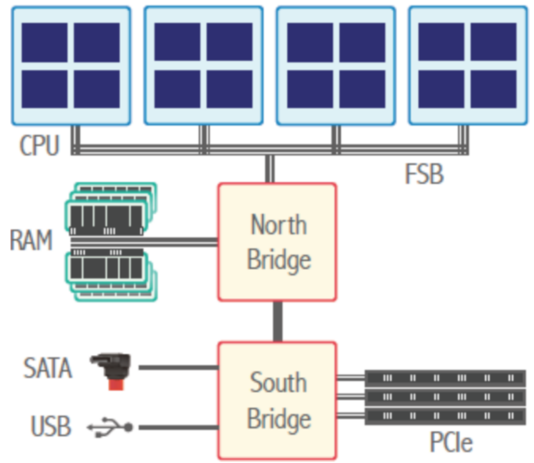
☞ https://en.wikipedia.org/wiki/Von_Neumann_architecture

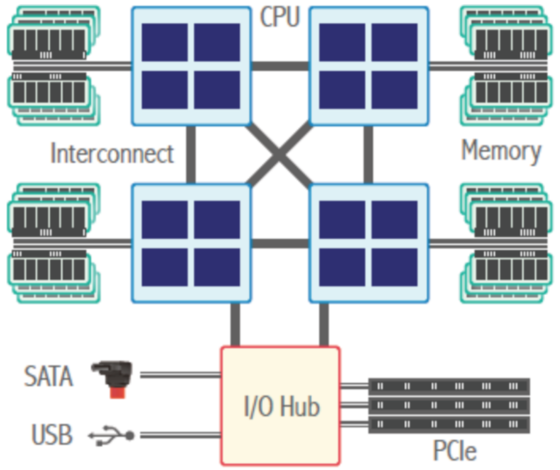
Von Neumann Bottleneck

- Instructions and data share the same bus between memory and processing
  - Bandwidth issues
  - Latency issues
  - ”Tricks”
    - => Caches
    - => Branch Prediction
    - => Stacks
- Additional issue:
  - Modifying data can modify instructions…



Motherboard Layouts



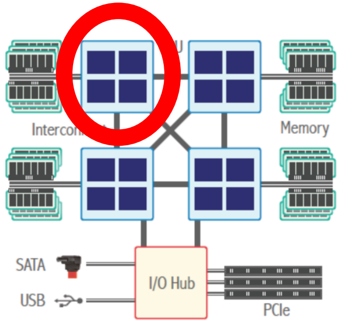


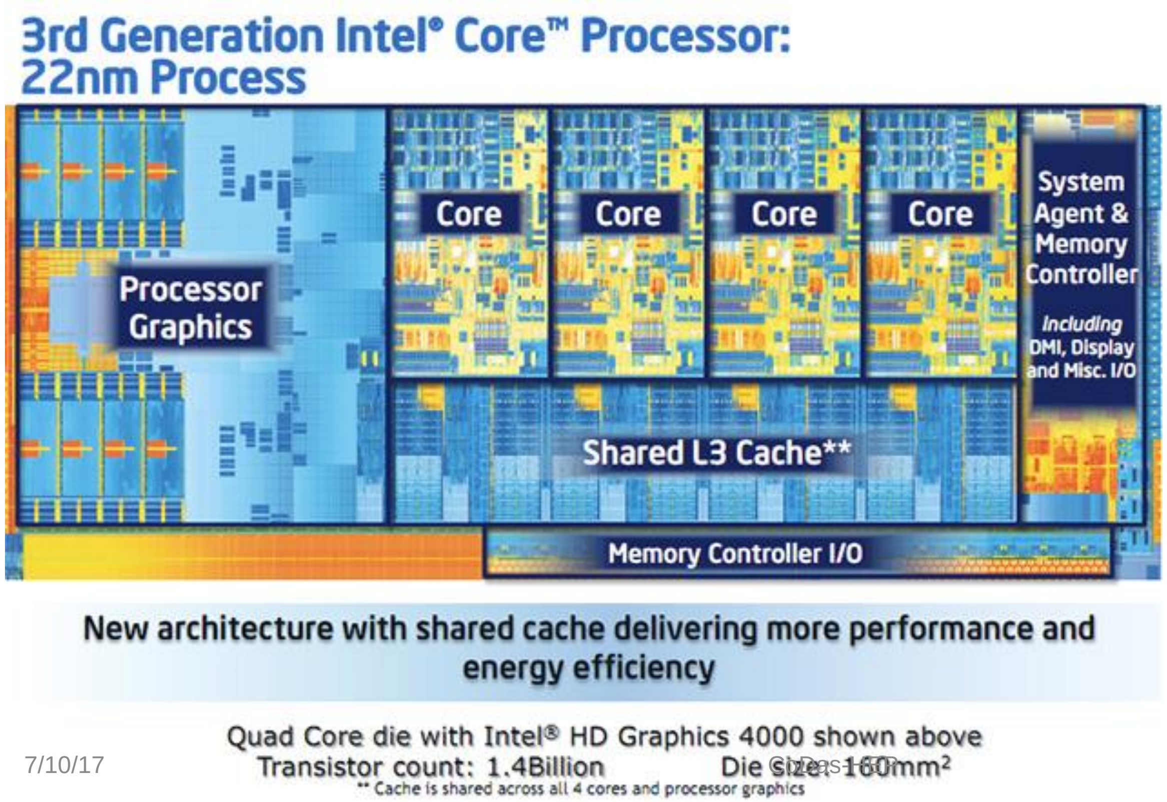
← Intel QPI, AMD HT

Symmetric MultiProcessing (SMP)
front side bus (FSB) bottlenecks in SMP systems, device-to-memory bandwidth via north bridge, north bridge to RAM bandwidth limitations.

Non-Uniform Memory Access (**NUMA**)
physical memory partitioned per CPU, fast interconnect to link CPUs to each other and to I/O.
Remove bottleneck but memory is no longer uniform – 30-50% overhead to accessing remote memory, up to 50x in obscure multi-hop systems.

Example: SandyBridge CPU





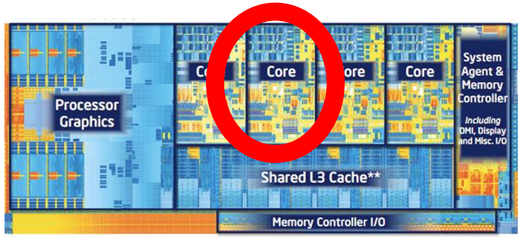
3rd Generation Intel® Core™ Processor:
22nm Process
Processor Graphics
Core
Core
Core
Core
System Agent & Memory Controller
Including DMI, Display and Misc. I/O
Shared L3 Cache**
Memory Controller I/O

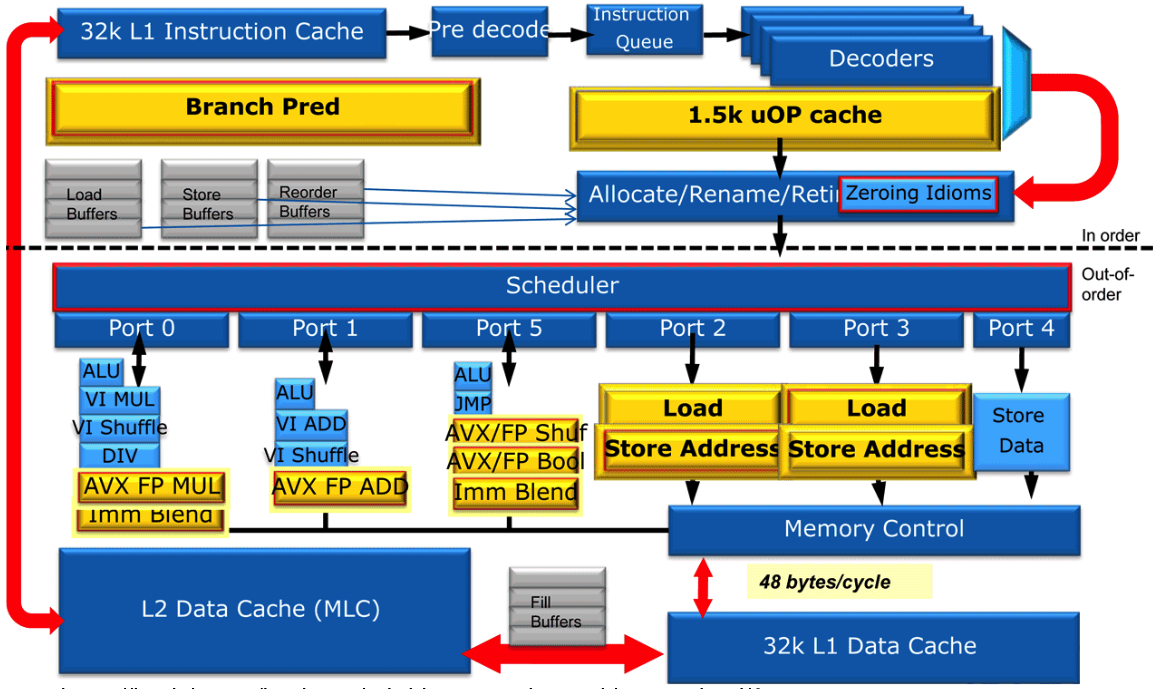
New architecture with shared cache delivering more performance and energy efficiency

Quad Core die with Intel® HD Graphics 4000 shown above
Transistor count: 1.4Billion Die size: 160mm²
** Cache is shared across all 4 cores and processor graphics

Core Definition (Jarp):
"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."

SNB Core Architecture





https://hw-lab.com/intel-sandy-bridge-cpu-microarchitecture.html/6

Key Components:
Control logic

Register file

Functional Units
- ALU (arithmetic and logic unit)
- FPU (floating point unit)

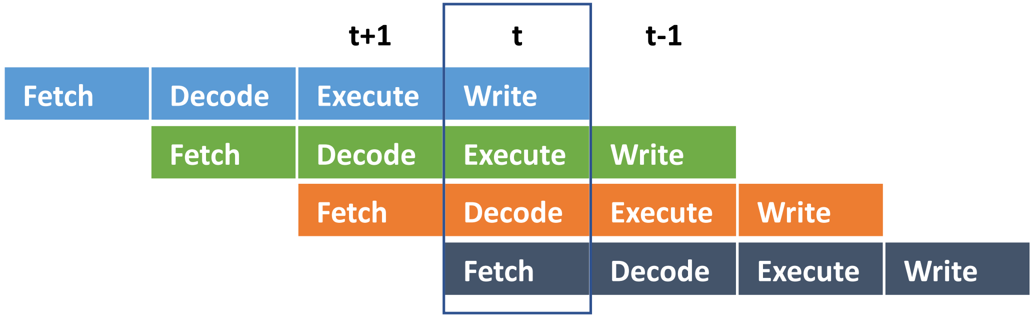
Data Transfer
- Load / Store

Instruction Pipelining

- Instructions are split into stages
  - Number of stages depends on the architecture (SNB: 14 (16 with fetch/retire))

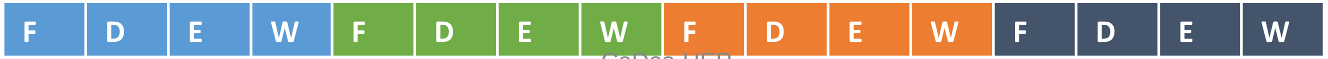
- Instruction level parallelism (ILP)
  (one of the technique to exploit it)
- Increase instruction throughput
- Potential issues: bubbles

With a one-cycle latency (in-order) pipeline:



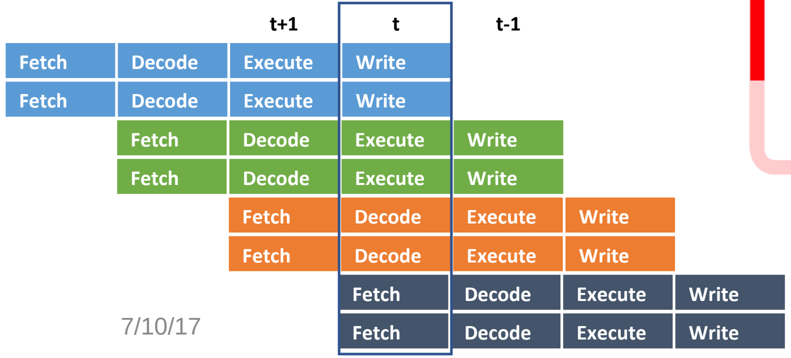
Write ≡ Write-back

Without pipelining:

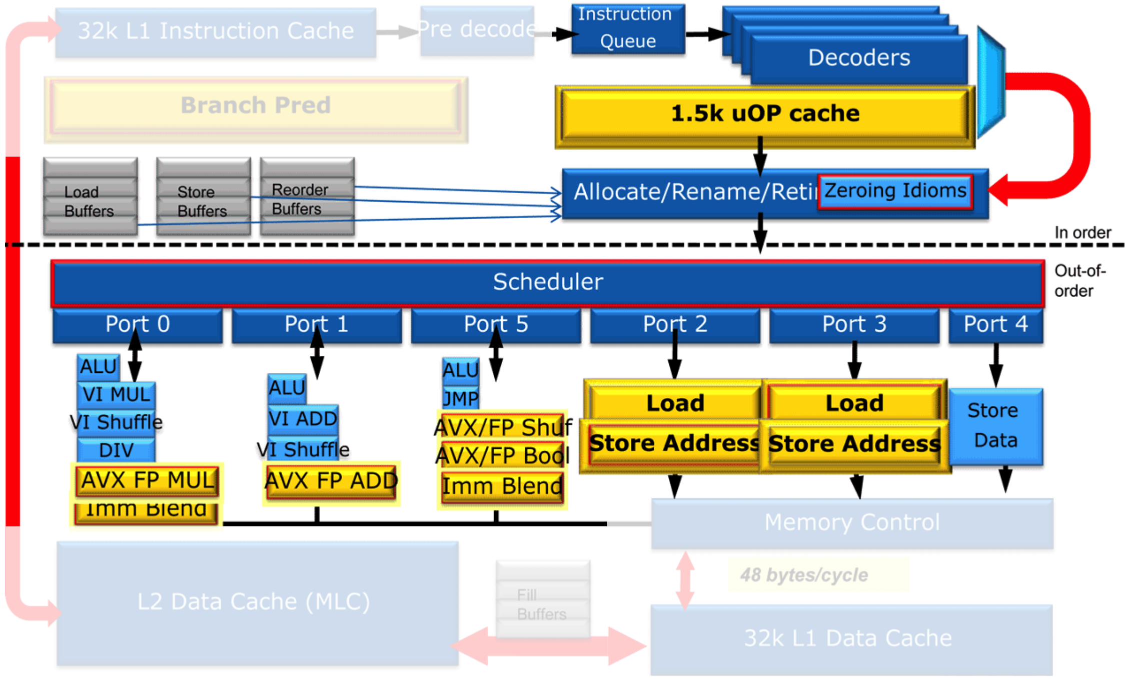


Superscalar Architecture

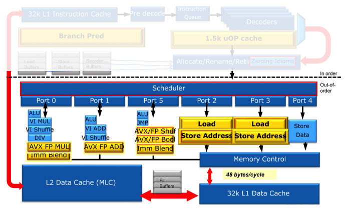
- “Multiple pipelines”
- Increase the ILP
- Limited by:
  - Degree of parallelism (dependencies)
  - Branching

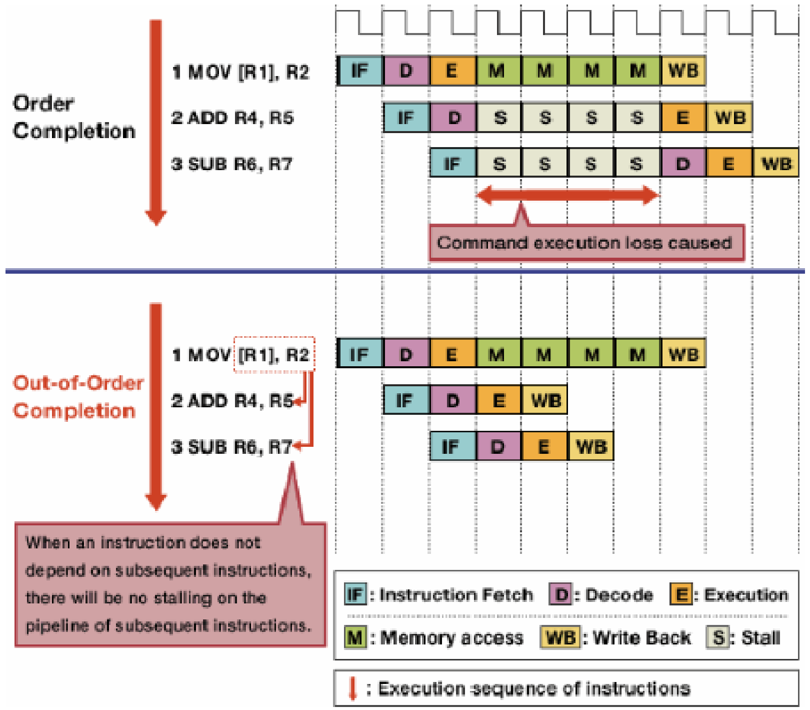


		t+1	t	t-1		
Fetch	Decode	Execute	Write			
Fetch	Decode	Execute	Write			
	Fetch	Decode	Execute	Write		
	Fetch	Decode	Execute	Write		
		Fetch	Decode	Execute	Write	
		Fetch	Decode	Execute	Write	
			Fetch	Decode	Execute	Write
			Fetch	Decode	Execute	Write



Out-of-Order Execution

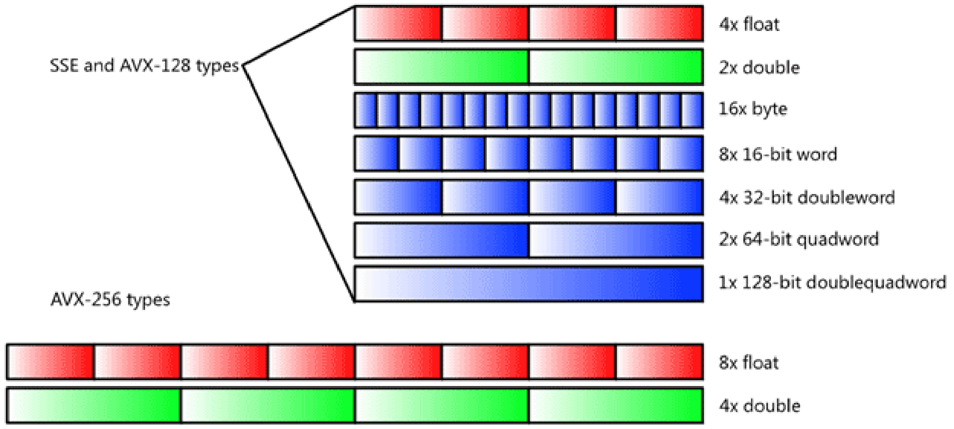




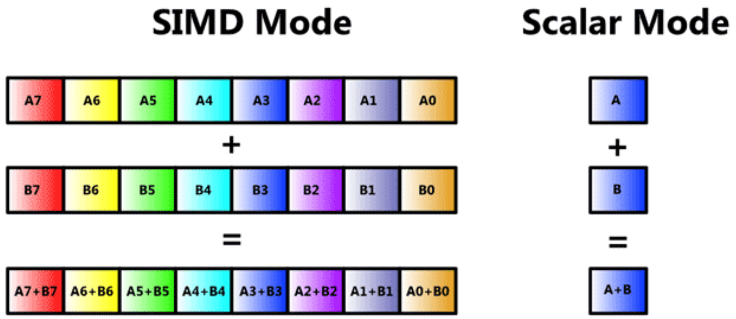
- Avoid pipeline bubbles
- Out-of-order execution (OoOE)
  - Execution can be scheduled to compensate for unavailable functional units or while waiting for data
- Speculative execution of next independent instruction
  - Some instructions might have to be unrolled
  - Also used in branch-prediction

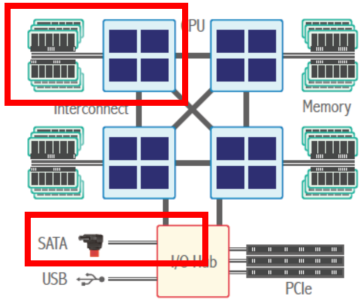
Vector Unit

- AVX: Advanced Vector Extension
- SSE: Streaming SIMD Extensions
- SIMD: Single Instructions Multiple Data
  - ☞ Flynn classification

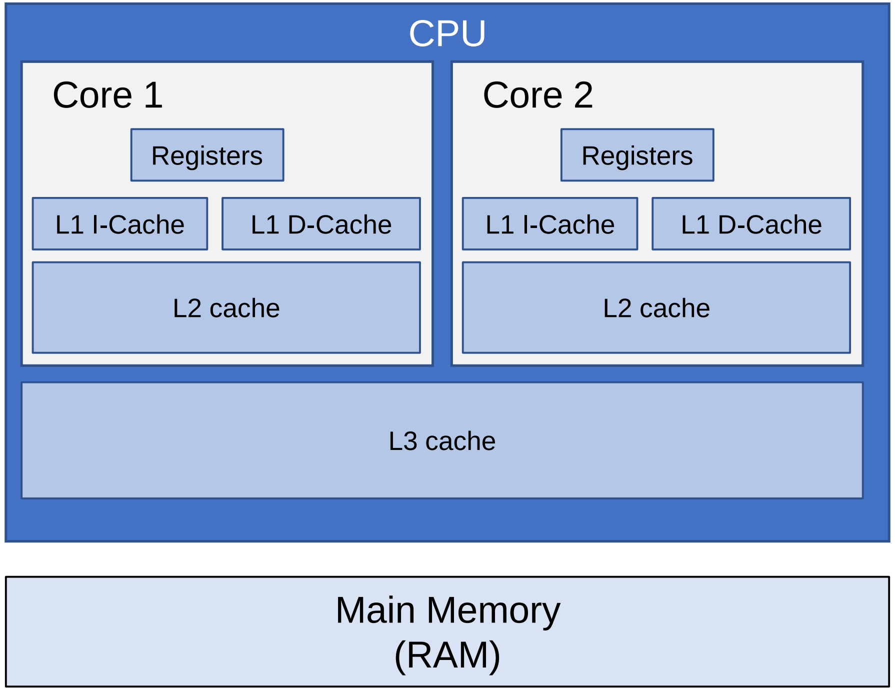


https://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions





Memory Hierarchy

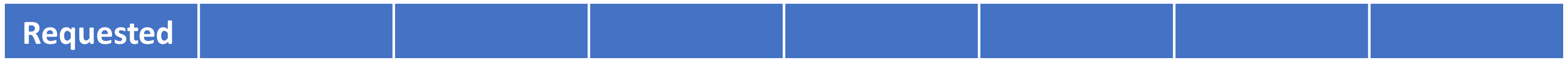


	Registers	L1 Cache	L2 Cache
Speed	1 cycle	~4 cycles	~10 cycles
Size	< KB	~32KB	~256KB

	L3 Cache	DRAM	Disk
Speed	~30 Cycle	~200 cycles	10 ms
Size	~35MB	10-100 GB	TB

Cache Lines (1)

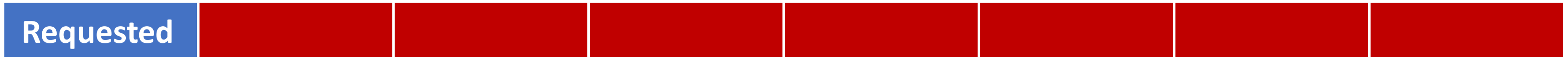
- **When a data element or an instruction is requested by the processor, a cache line is ALWAYS moved (as the minimum quantity), usually to Level-1**



- **A cache line is a contiguous section of memory, typically 64B in size (8 * double) and 64B aligned**
  - A 32KB Level-1 cache can hold 512 lines
- **When cache lines have to be moved come from memory**
  - Latency is long (>200 cycles)
    - It is even longer if the memory is remote
  - Memory controller stays busy (~8 cycles)

Cache Lines (2)

- **Good utilization is vital**
  - When only one element (4B or 8B) element is used inside the cache line: A lot of bandwidth is wasted!



- **Multidimensional C arrays should be accessed with the last index changing fastest:**

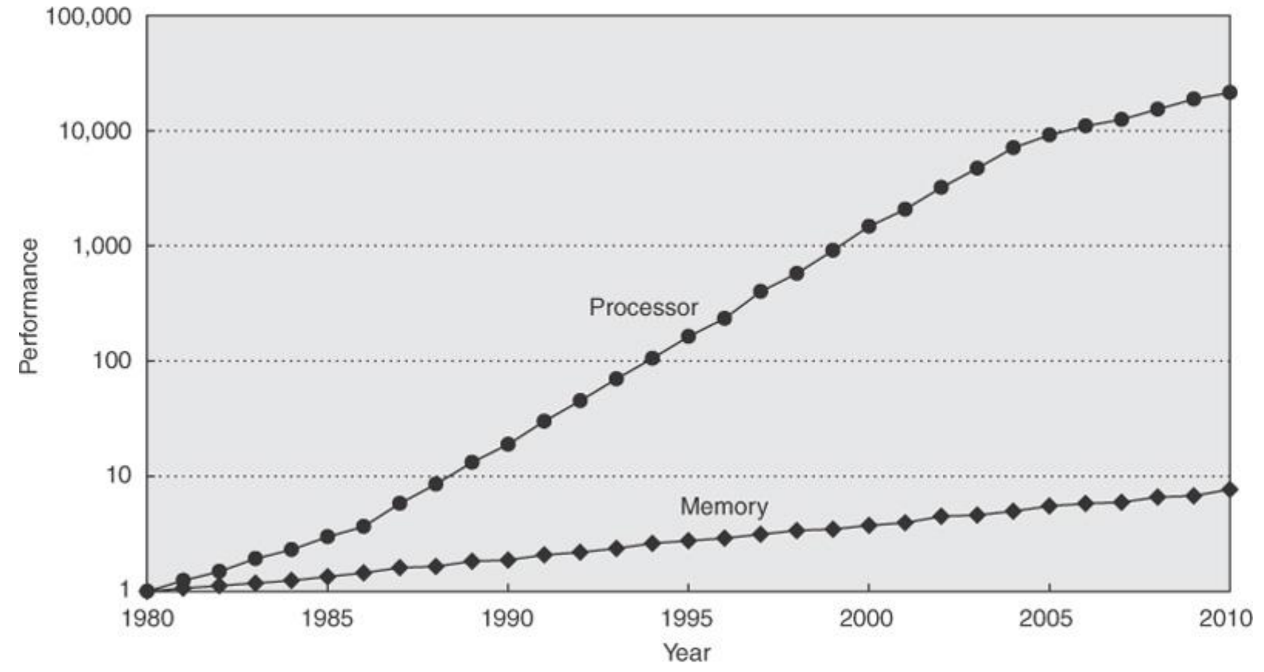
```
for (int j = 0; j < M; ++j)
    for (int i = 0; j < N; ++j)
        A[j][i] = B[j][i] + C[j][i];
```

- **Pointer chasing (in linked lists) can easily lead to "cache thrashing" (too much memory traffic)**

Cache Lines (3)

- Prefetching:
  - Fetch a cache line before it is requested Hiding latency
  - Normally done by the hardware / compiler
    - Especially if processor executes Out-of-order
  - Also done by software instructions
    - Especially when In-order (IA-64, Xeon Phi, etc.)
- Locality is vital:
  - Spatial locality – Use all elements in the line
  - Temporal locality – Complete the execution whilst the elements are certain to be in the cache
- False Sharing:
  - Two threads using data from the same cache line can lead to ”false-sharing”
  - Only happen if the caches are coherent
    - Cache coherency is a mechanism to ensure data accesses are made on up-to-date cache lines.

Does it matter?



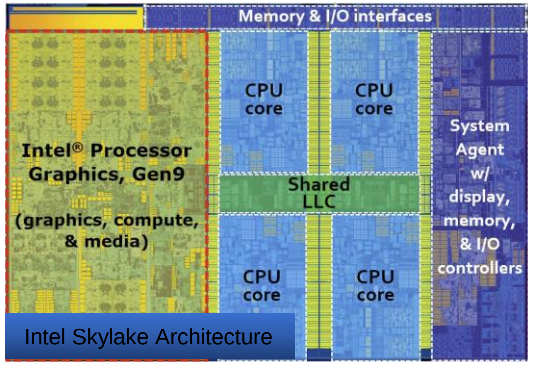
http://web.sfc.keio.ac.jp/~rdv/keio/sfc/teaching/architecture/architecture-2008/hennessy-patterson/Ch5-fig02.jpg

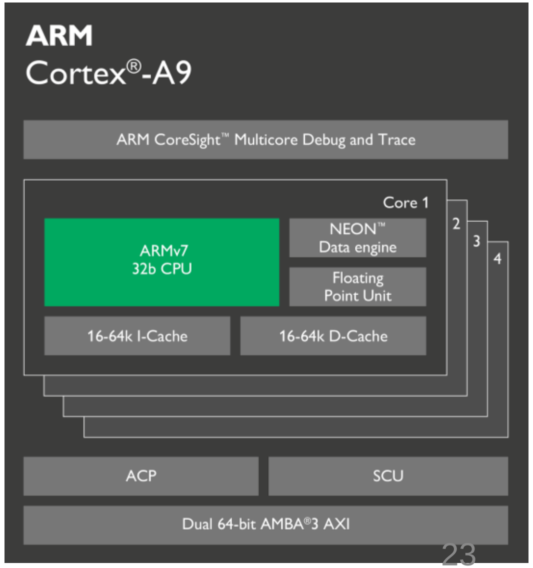
Multicore Processors

- (A lot of) modern processors are multicores
  - Intel Xeon
  - IBM Power
  - AMD Opteron
  - ARM processors
  - Mobile processors (Snapdragon, …)

- Most have vector units
- Most have multiple cache levels
- Require special care to program (multi-threading, vectorization, …)

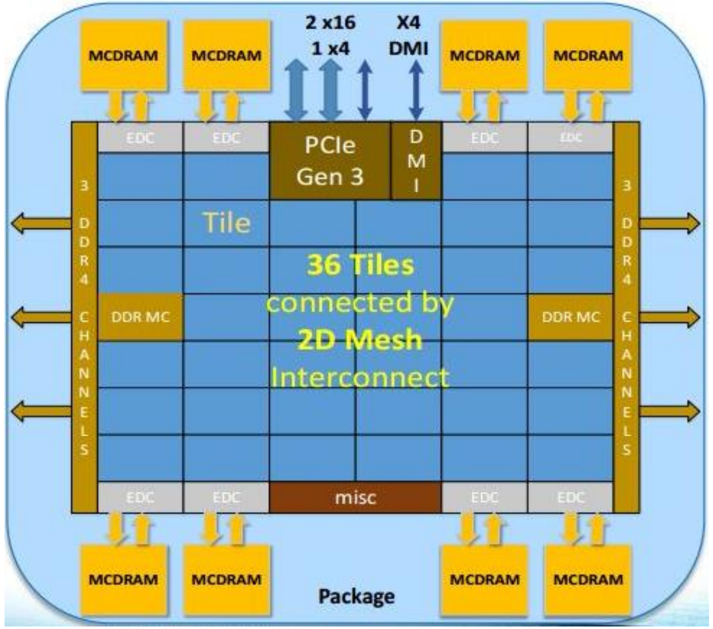
- Many-core Processors
  - Tilera
  - SW26010 (Sunway TaihuLight's processors)
  - Intel: Mic / Xeon Phi / KNL / …
  - GPUs


Memory & I/O interfaces
CPU core
CPU core
Intel® Processor Graphics, Gen9
(graphics, compute, & media)
Shared LLC
CPU core
CPU core
System Agent w/ display, memory, & I/O controllers
Intel Skylake Architecture



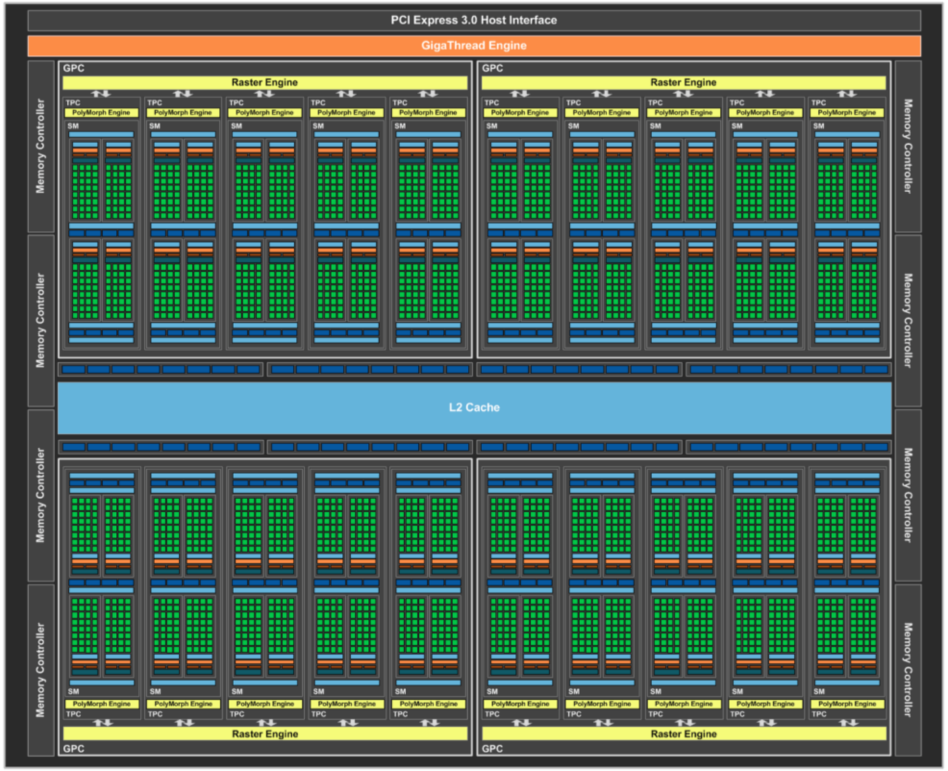
Intel Xeon Phi

- x86-compatible multiprocessor architecture
- Mostly used as a co-processor
- Programmable using
  - C, C++, Fortran
  - MPI, OpenMP, OpenCL, …



NVIDIA Pascal

- Graphic Processing Unit (GPU)
- Exclusively a co-processor
- Not compatible with x86 library
- Programmable using:
  - Cuda, OpenCL
  - OpenAcc, OpenMP



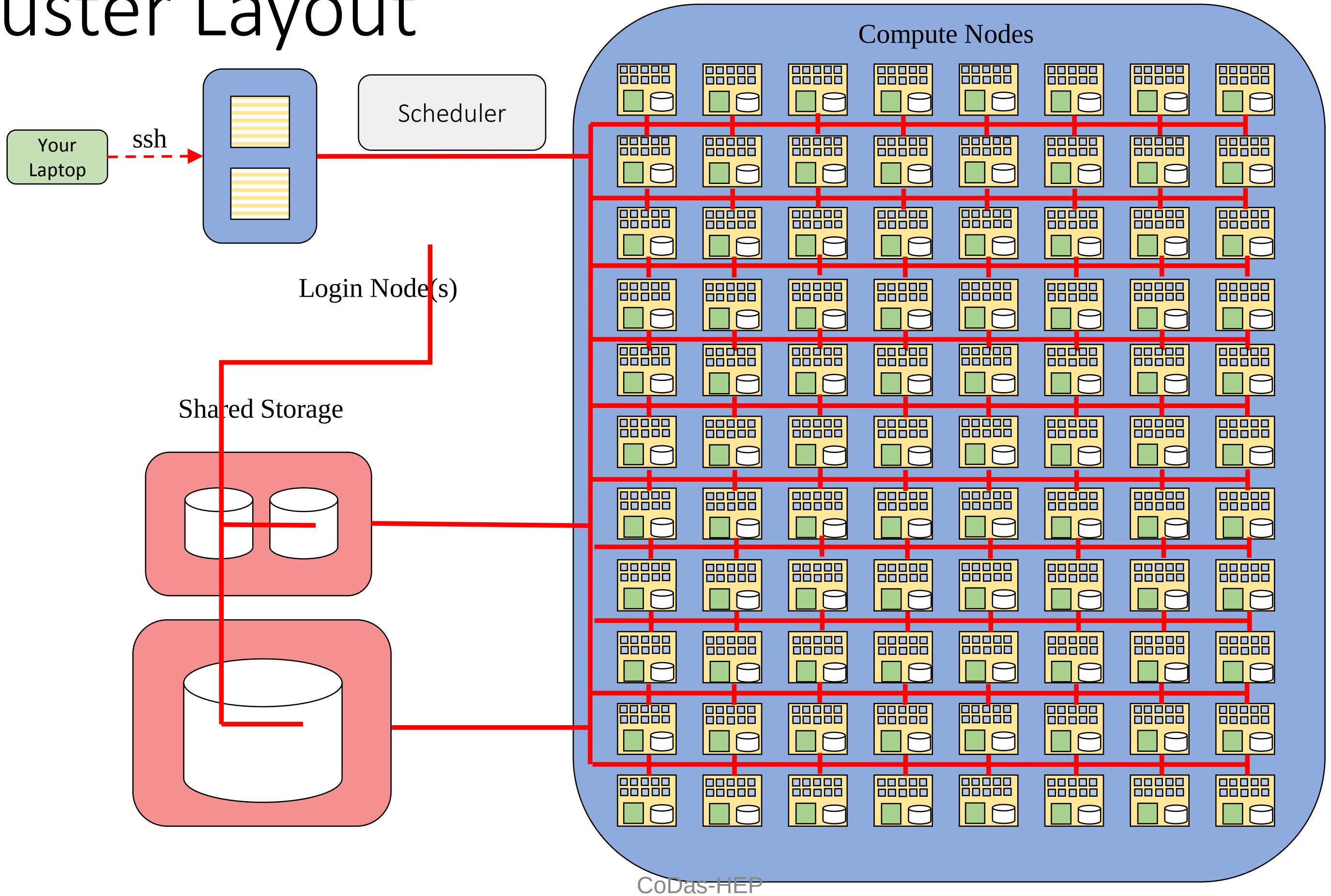
Comparison

	Xeon E5-2680 v4	**Xeon Phi 7120P**	**NVIDIA P100**
Cores	14 x 2	72	56
Logical Cores	28 x 2	288	1792(SP)/ 3584(DP)
Clock rate	2.4 GHz	1.7 GHz	1480 MHz
Theoretical GFLOPS (double)	268 x 2	3456	5304
SIMD width	64 bytes	128 bytes	Warp of 32 threads
Memory	~32-1540GB x 2	16 GB MCDRAM 384 GB DDR4	16 GB
Memory B/W	76.8 GB/s x 2	352 GB/s	288 GB/s
~ Launching price	$1700 x 2	$6703	$12,500

What is a Cluster?

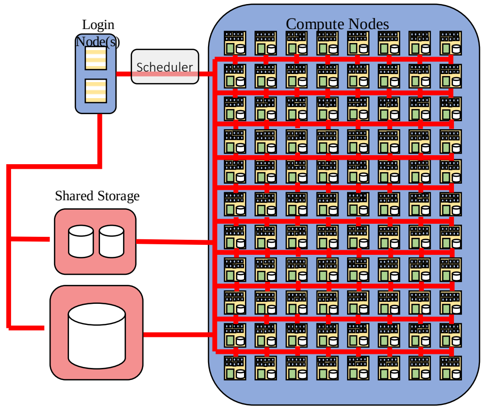
- ("Beowulf") Cluster
  - A collection of commodity computers connected to form a single entity
    - Each is essentially a self contained computer
    - OS with CPU, RAM, Hard drive, etc.
    - Perfect for MPI
  - Stored in racks in dedicated machine rooms
  - Connected together via (low latency) interconnects
  - Connected to storage
  - Vast majority of HPC clusters
- SMP Cluster
  - Symmetric Multi-Processing
  - CPUs all share memory – essentially one machine
  - Expensive and serve unique purpose
    - Huge memory and huge OpenMP jobs
  - Princeton Tigress: Hecate
- Vector supercomputers
  - Not anymore, too expensive

Basic Cluster Layout

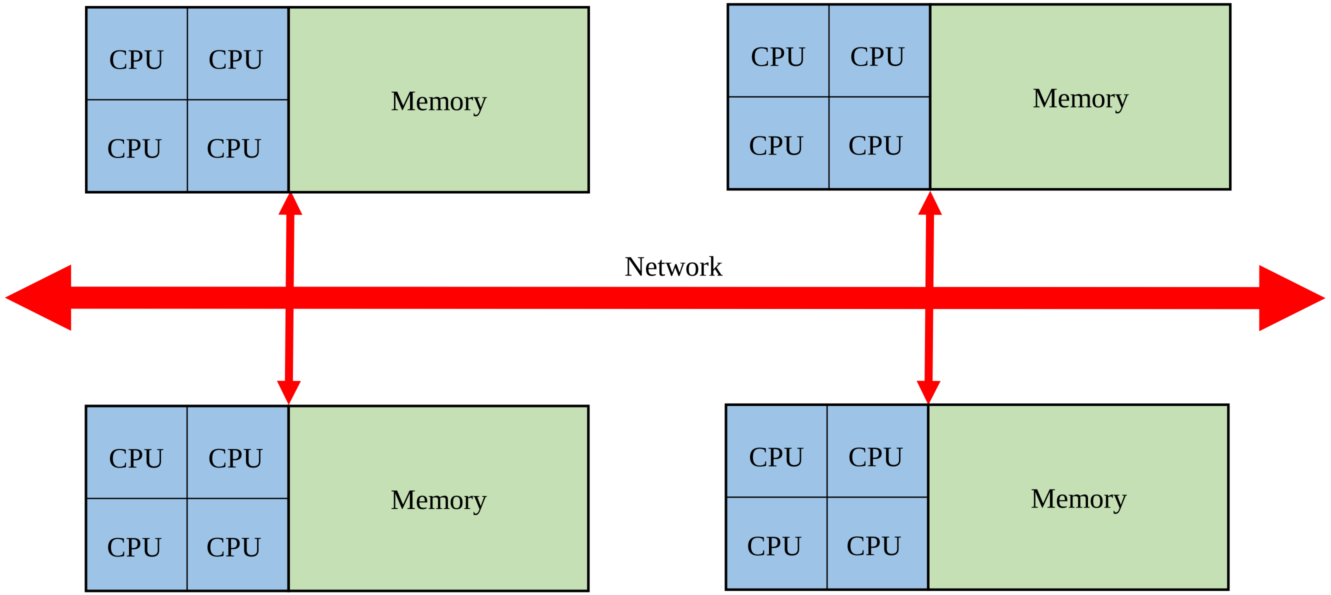


Cluster Decomposed

- Login Node(s)
  - Edit, debug, compile, and interact with scheduler
  - Not for long running jobs!
- Scheduler
  - You tell the scheduler what resources you need
    - # of cpus, #of nodes, GB of memory, # of hours, etc
    - Then what to do: “run this program”
  - Scheduler then assigns hardware exclusive for your job
  - SLURM on our machines
- Storage
  - /home directories NFS mounted everywhere
  - Each compute node has local storage
  - Some clusters have more storage options
    - Parallel, backed up long term storage
    - See researchcomputing.princeton.edu

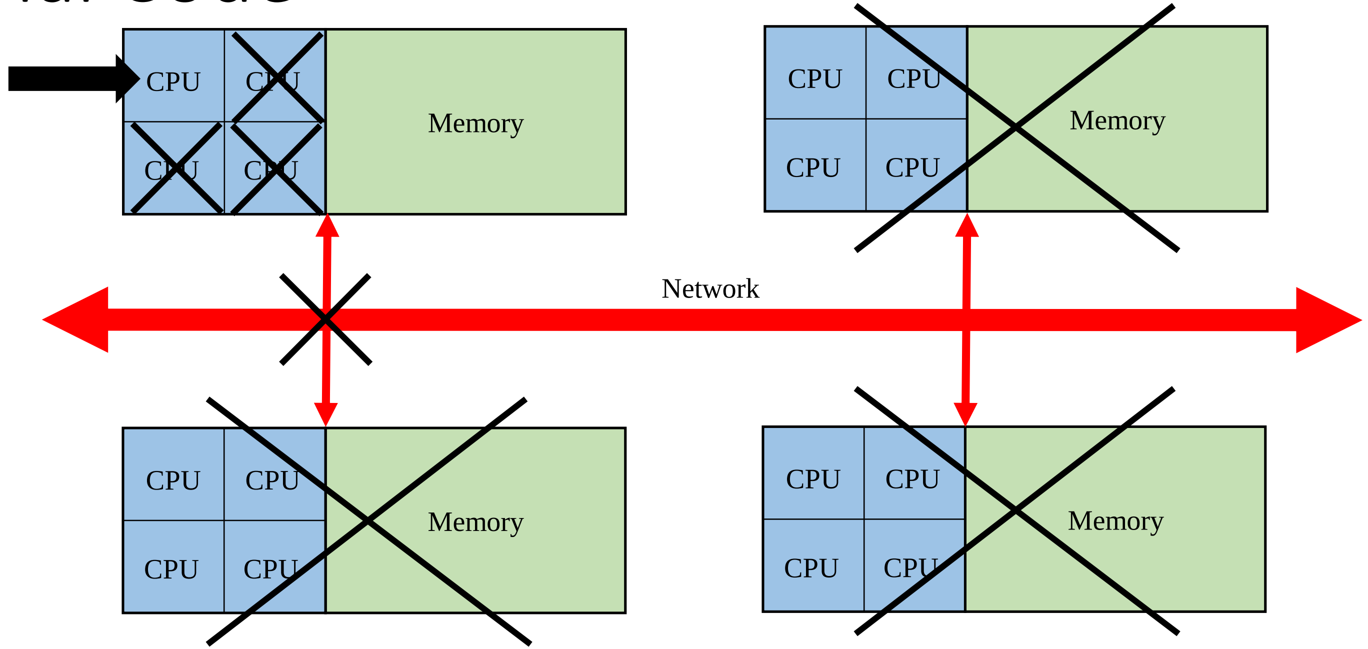


Modern HPC "Beowulf" Cluster

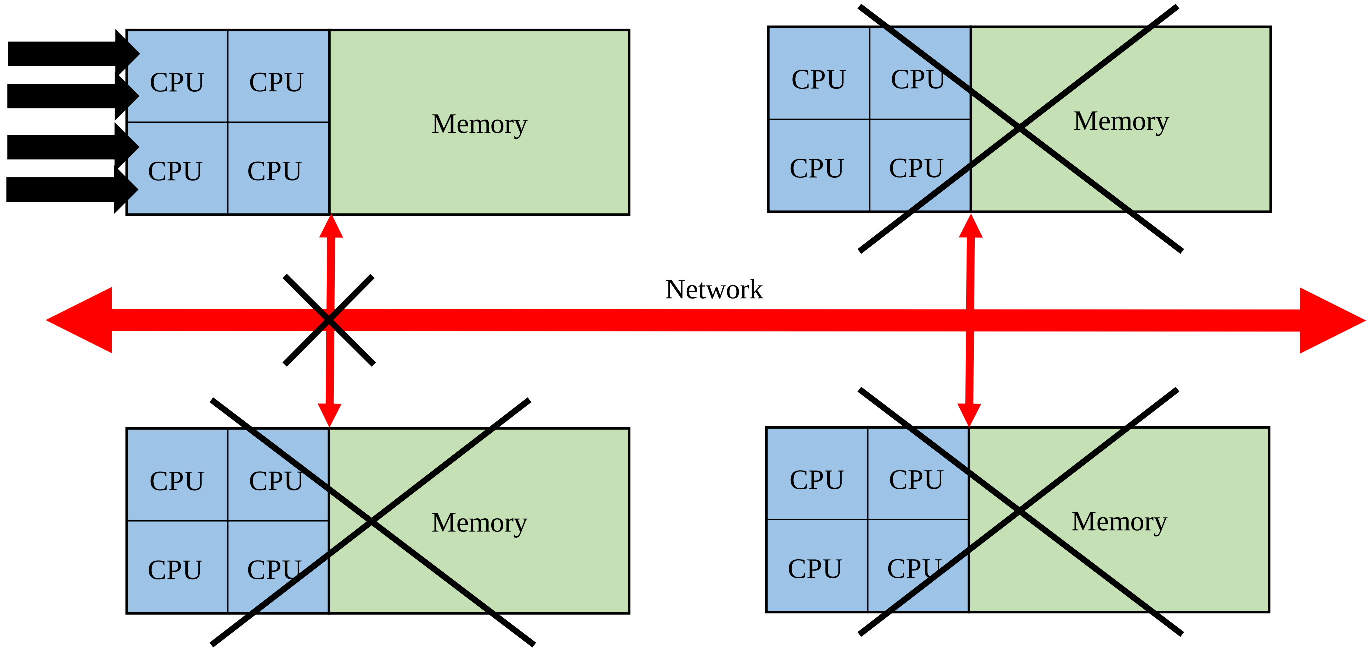


Tiger has 16 cores per node and 644 nodes
For a total of more than 10,000 cores!

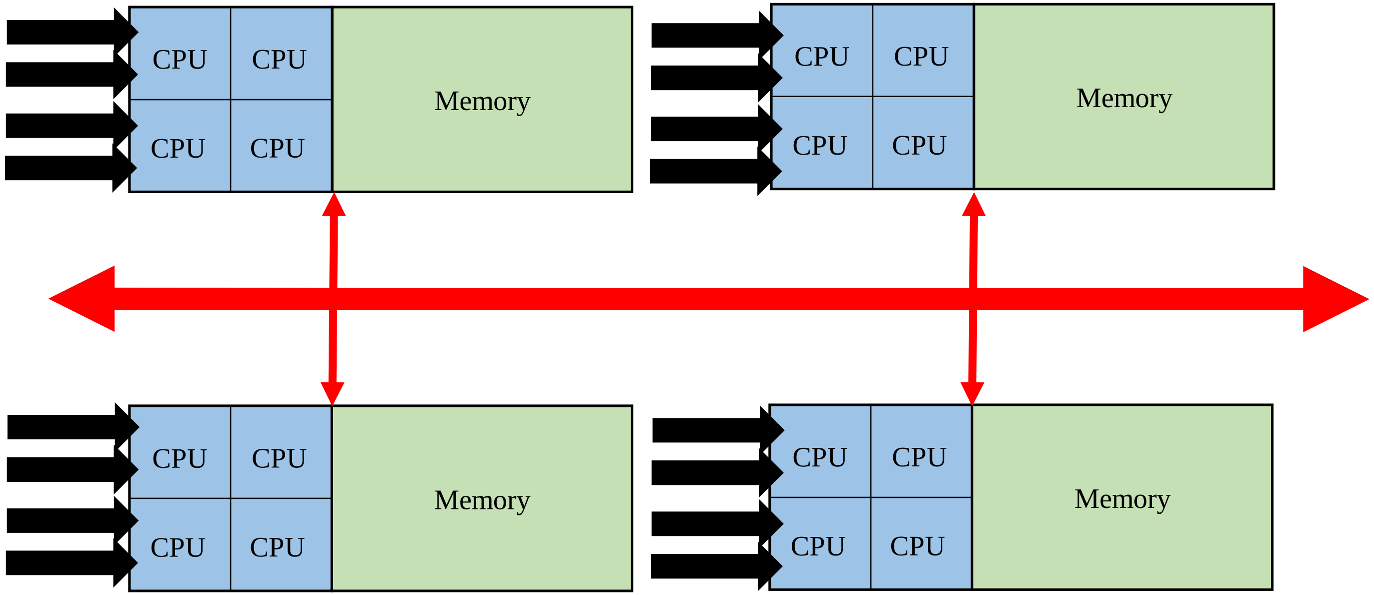
Serial Code


CPU
CPU
CPU
CPU
Memory
CPU
CPU
CPU
CPU
Memory
Network
CPU
CPU
CPU
CPU
Memory
CPU
CPU
CPU
CPU
Memory

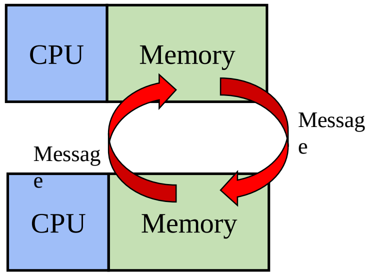
Multithreading (OpenMP)



Message Passing Interface (MPI)

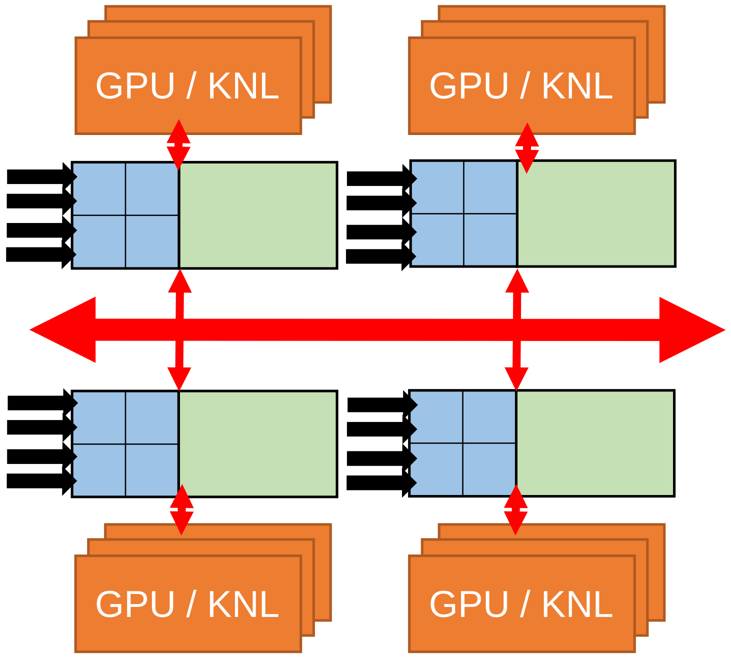


MPI + OpenMP



- MPI – Designed for distributed memory
  - In the old days this was it
  - Think “messages”
- OpenMP – Designed for shared memory
  - More cores & bigger memory = win!
  - Think “sharing”
- C, C++, and Fortran
- There are other options!
  - Interpreted languages with multithreading
    - Python, R, matlab (have OpenMP & MPI underneath)
  - CUDA, OpenACC (GPUs)
  - Pthreads, Intel Cilk Plus (multithreading)
  - OpenCL, Chapel, Co-array Fortran, Unified Parallel C (UPC)

Clusters with accelerators



- Accelerators: GPUs or Xeon Phi (KNC, KNL)
- Programmable with MPI + x
- x = OpenMP, OpenAcc, CUDA, ….
- Or x = OpenMP + CUDA, ….

- HPC center are constrained by the amount of power they can drain
- To increase computational power, increase FLOPS / Watt

- Top500.org has seen a shift toward systems with accelerators (less true in the June 2017 list)

Demystifying the Cloud

- ”Regular” computers, just somewhere else
- Provide users with remote virtual machines or containers
- Can be used for anything:
  - Mobile-services, Hosting websites, Business Application, …
  - **Data Analysis, High Performance Computing**
- Providers
  - Major players: Amazon, Google, Microsoft, HP, IBM, Salesforce.com, …
  - Lots of others

Cloud Computing

- Advantages:
  - Potentially lower cost:
    - Pay as you go
  - Potentially lower cost:
    - Save on sysadmins and infrastructure
  - Potentially lower cost:
    - Economy of scale: providers
  - Scaling up or down as needed
    - Can be used to run overflow from a regular data center
  - Access to a wide range of hardware
- Additional challenges
  - Data movement
    - Expensive and time consuming
  - Security, privacy, …

What we didn’t talk about

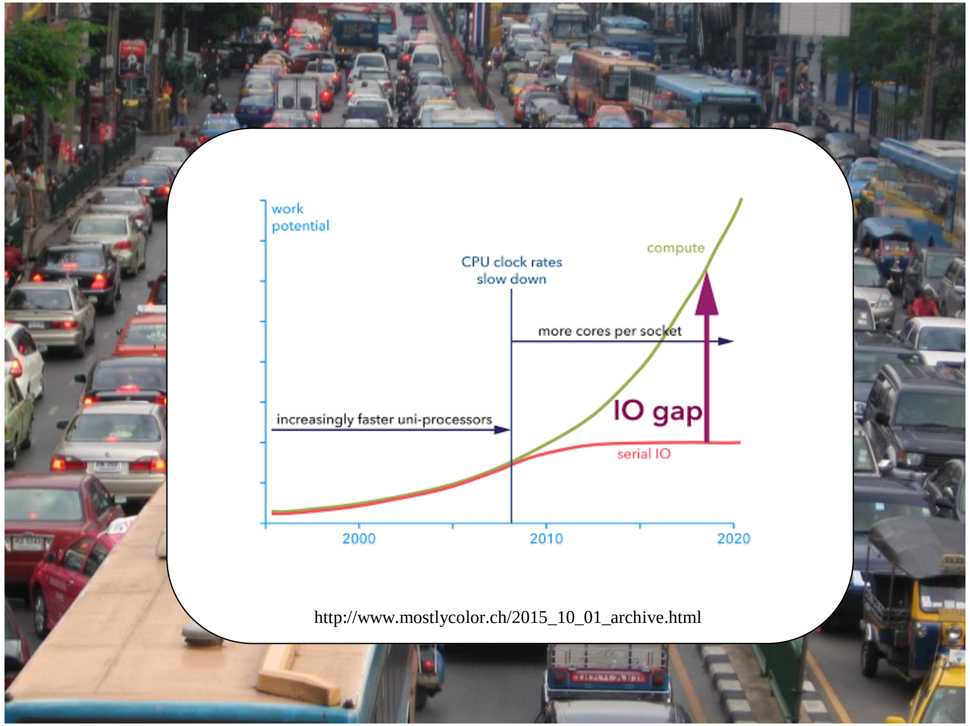
- Operating Systems
- Compiler/Library
- Functional units details (ALU, FPU, …)
- Cache associativity
- Virtual Memory
- I/O
- .... + a lot of other things

References

- J. Hennessy, D. Patterson, Computer Architecture: A Quantitative Approach, 5th edition (2011), ISBN 978-0-12-383872-8
- U. Drepper, What Every Programmer Should Know About Memory, http://people.redhat.com/drepper/cpumemory.pdf

- Glossary:
  - https://cvw.cac.cornell.edu/main/glossary

File I/O?



https://en.wikipedia.org/wiki/Transport_in_Bangkok