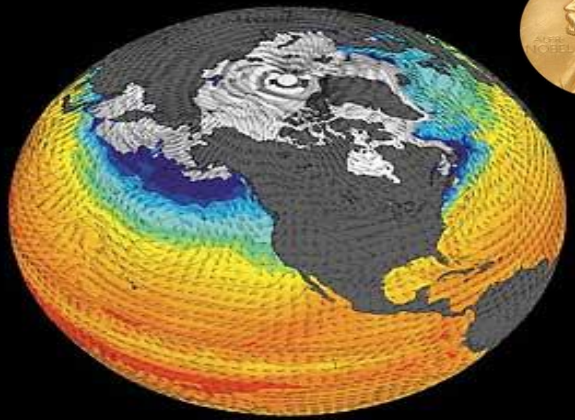
Intro To Parallel Computing

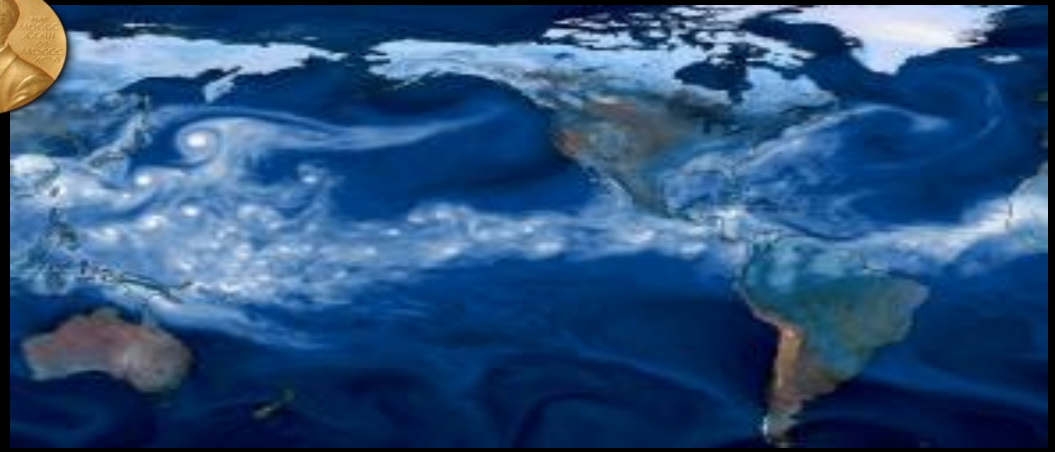
John Urbanic
Parallel Computing Scientist
Pittsburgh Supercomputing Center

Purpose of this talk

- This is the 50,000 ft. view of the parallel computing landscape. We want to orient you a bit before parachuting you down into the trenches to deal with MPI.
- This talk bookends our technical content along with the Outro to Parallel Computing talk. The Intro has a strong emphasis on hardware, as this dictates the reasons that the software has the form and function that it has. Hopefully our programming constraints will seem less arbitrary.
- The Outro talk can discuss alternative software approaches in a meaningful way because you will then have one base of knowledge against which we can compare and contrast.
- The plan is that you walk away with a knowledge of not just MPI, etc. but where it fits into the world of High Performance Computing.

FLOPS we need: Climate change analysis



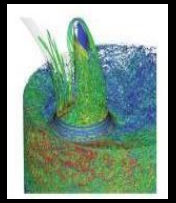
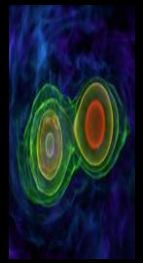
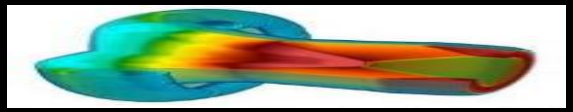
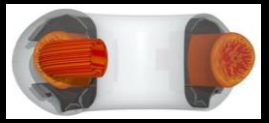
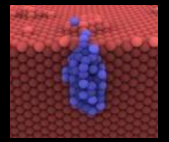
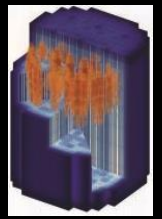
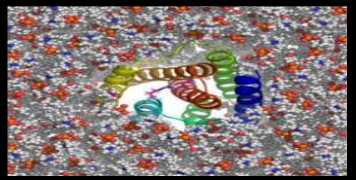
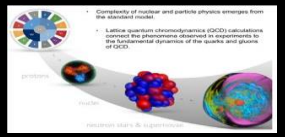


Simulations	Extreme data
• Cloud resolution, quantifying uncertainty, understanding tipping points, etc., will drive climate to exascale platforms • New math, models, and systems support will be needed	• "Reanalysis" projects need 100× more computing to analyze observations • Machine learning and other analytics are needed today for petabyte data sets • Combined simulation/observation will empower policy makers and scientists

Courtesy Horst Simon, LBNL

The list is long, and growing.

- Molecular-scale Processes: atmospheric aerosol simulations
- AI-Enhanced Science: predicting disruptions in tokomak fusion reactors
- Hypersonic Flight
- Modeling Thermonuclear X-ray Bursts: 3D simulations of a neutron star surface or supernovae
- Quantum Materials Engineering: electrical conductivity photovoltaic and plasmonic devices
- Physics of Fundamental Particles: mass estimates of the bottom quark
- Digital Cells

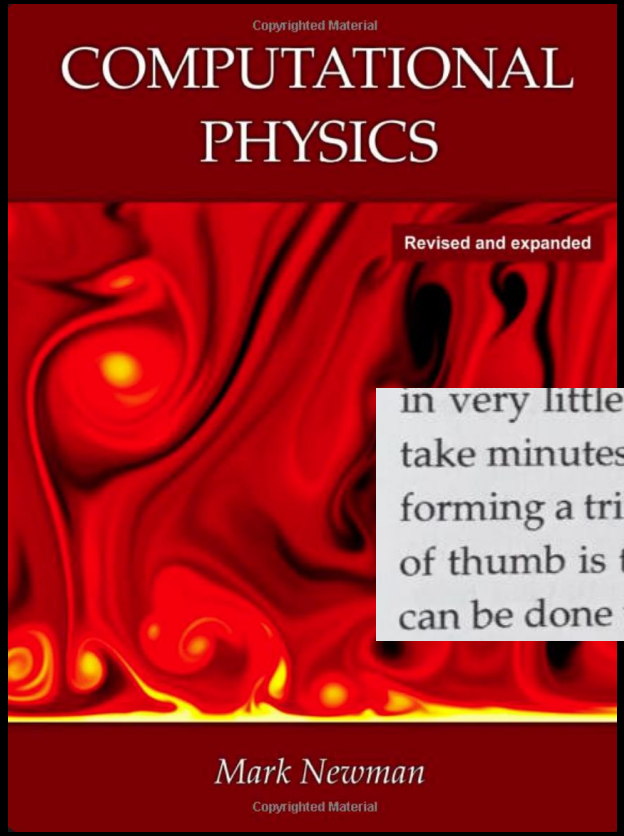
These and others are in an appendix at the end of our Outro To Parallel Computing talk.

And many of you doubtless brought your own immediate research concerns. Great!

'Nuff Said

There is an appendix with many more important exascale challenge applications at the end of our Outro To Parallel Computing talk.

And, many of you doubtless brough your own immediate research concerns. Great!



in very little time. Performing a billion operations, on the other hand, could take minutes or hours, though it's still possible provided you are patient. Performing a trillion operations, however, will basically take forever. So a fair rule of thumb is that the calculations we can perform on a computer are ones that can be done with *about a billion operations or less.*

Welcome to The Year of Exascale!

exa = 10^{18} = 1,000,000,000,000,000,000 = quintillion

64-bit precision floating point operations per second

23,800
Cray Red Storms
2004 (42 Tflops)

133,33
NVIDIA V100
(7.5 Tflops)

Where are those 10 or 12 orders of magnitude?

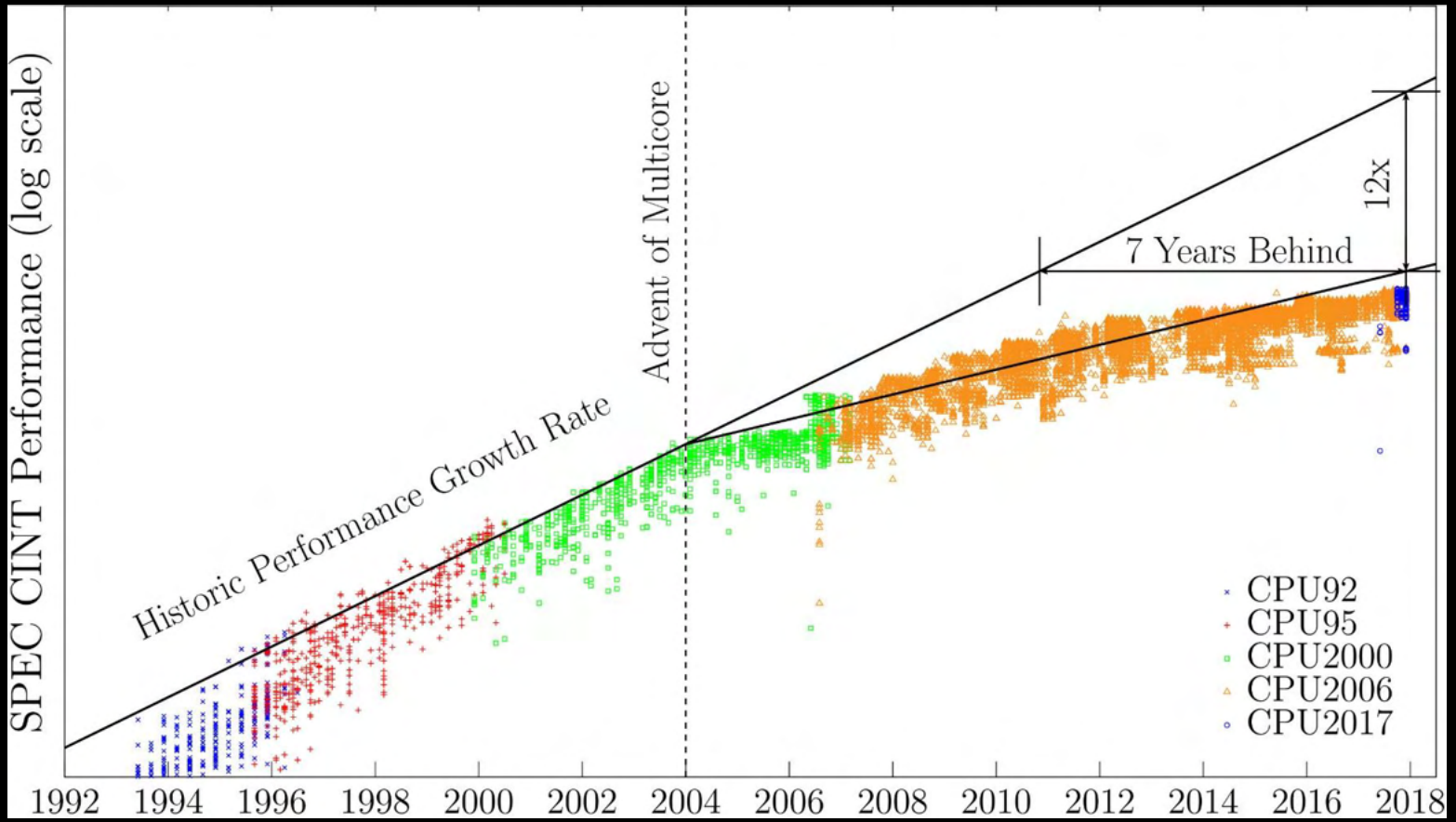
How do we get there from here?

BTW, that's a bigger gap than

VS.

IBM 709
12 kiloflops

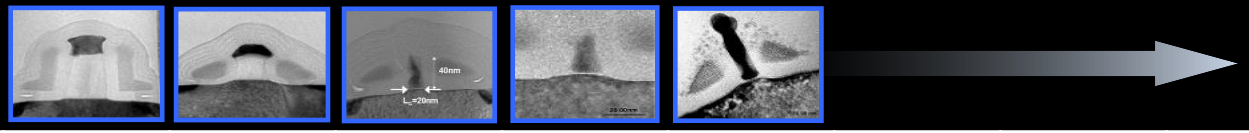
Moore's Law abandoned serial programming around 2004



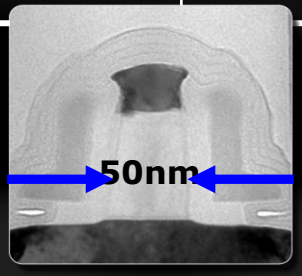
Courtesy Liberty Computer Architecture Research Group

But Moore's Law is only beginning to stumble now.

Intel process technology capabilities

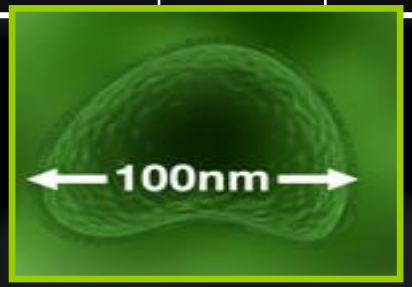


High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2018	2021
Feature Size	90nm	65nm	45nm	32nm	22nm	14nm	10nm	7nm
Integration Capacity (Billions of Transistors)	2	4	8	16	32	64	128	256



Transistor for 90nm Process

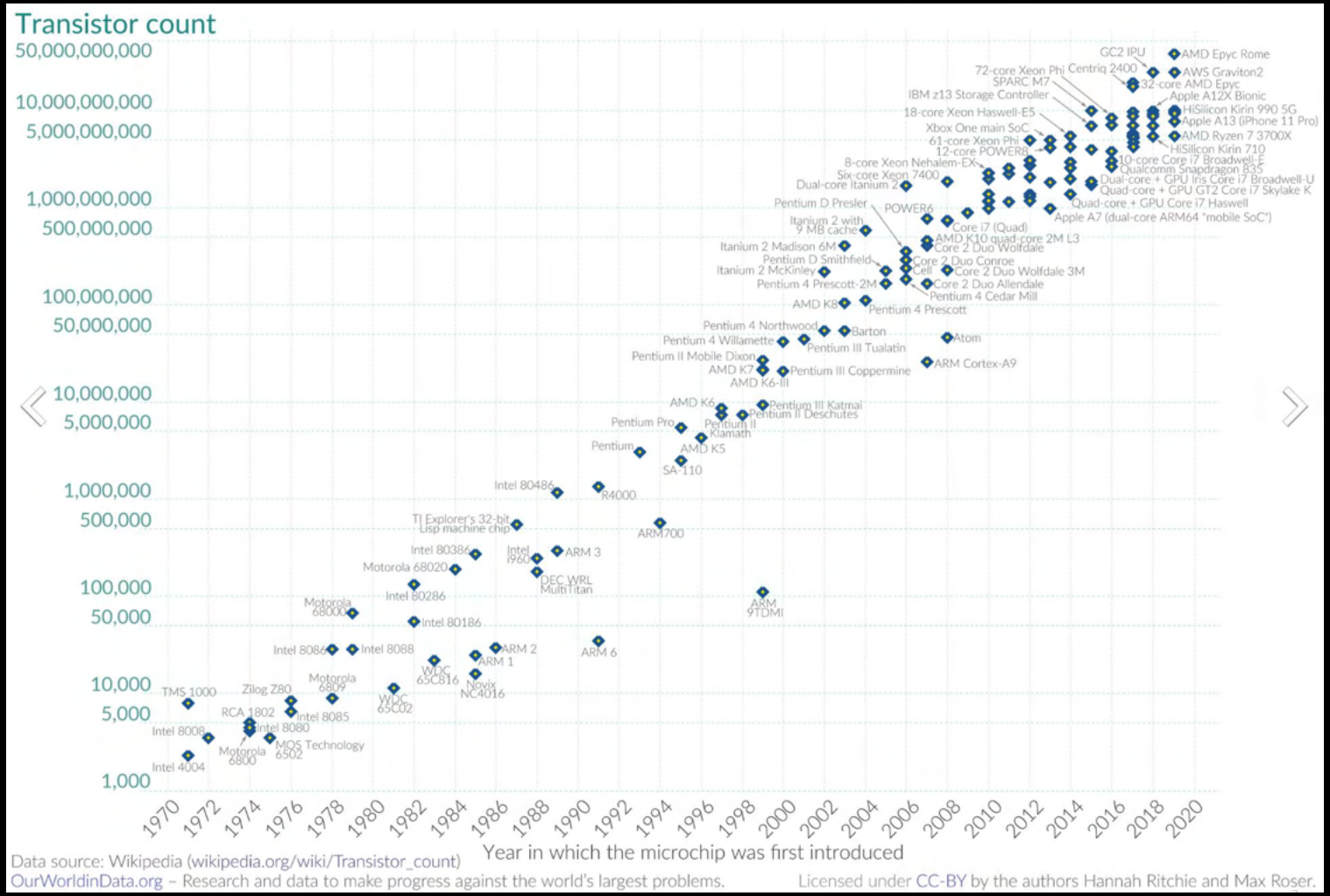
Source: Intel



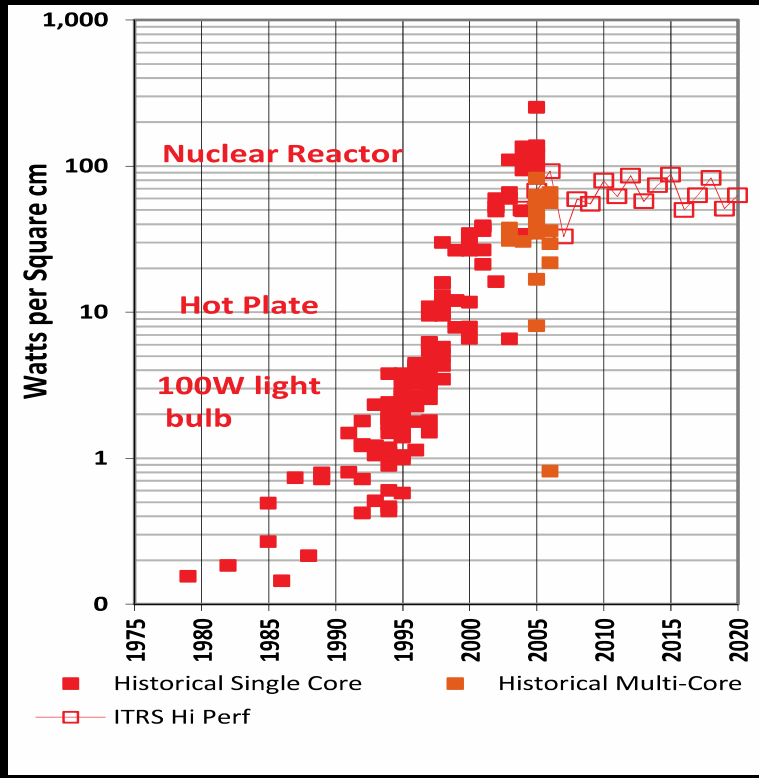
Influenza Virus

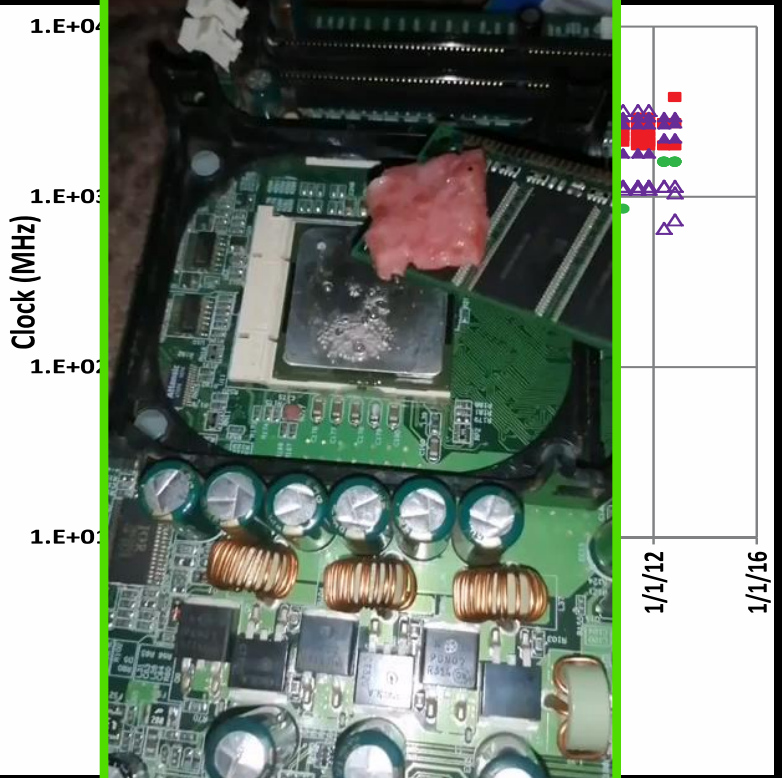
Source: CDC

And at end of day we keep using getting more transistors.



That Power and Clock Inflection Point in 2004… didn't get better.

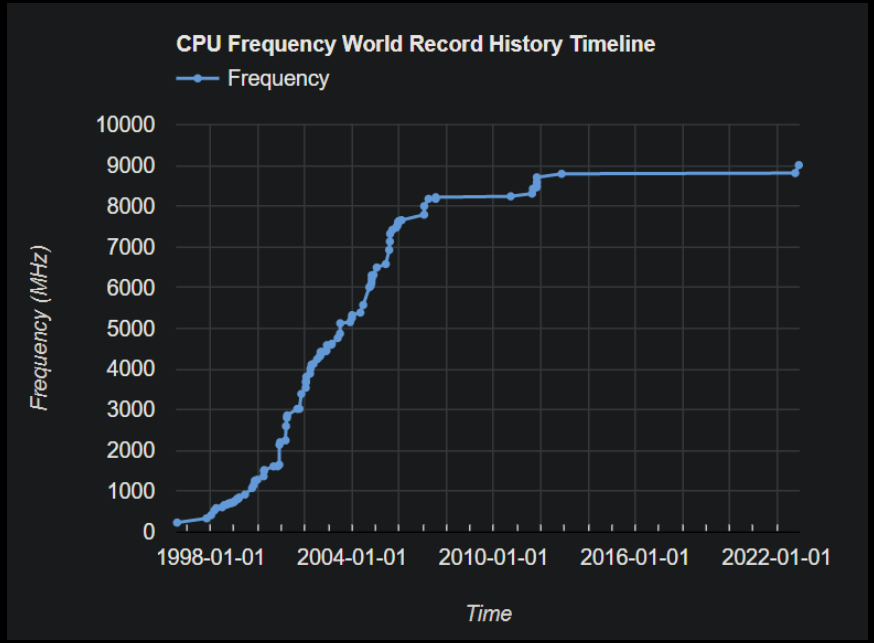




Fun fact: At 100+ Watts and <1V, currents are beginning to exceed 100A at the point of load.

Courtesy Horst Simon, LBNL

Even when you go extreme...

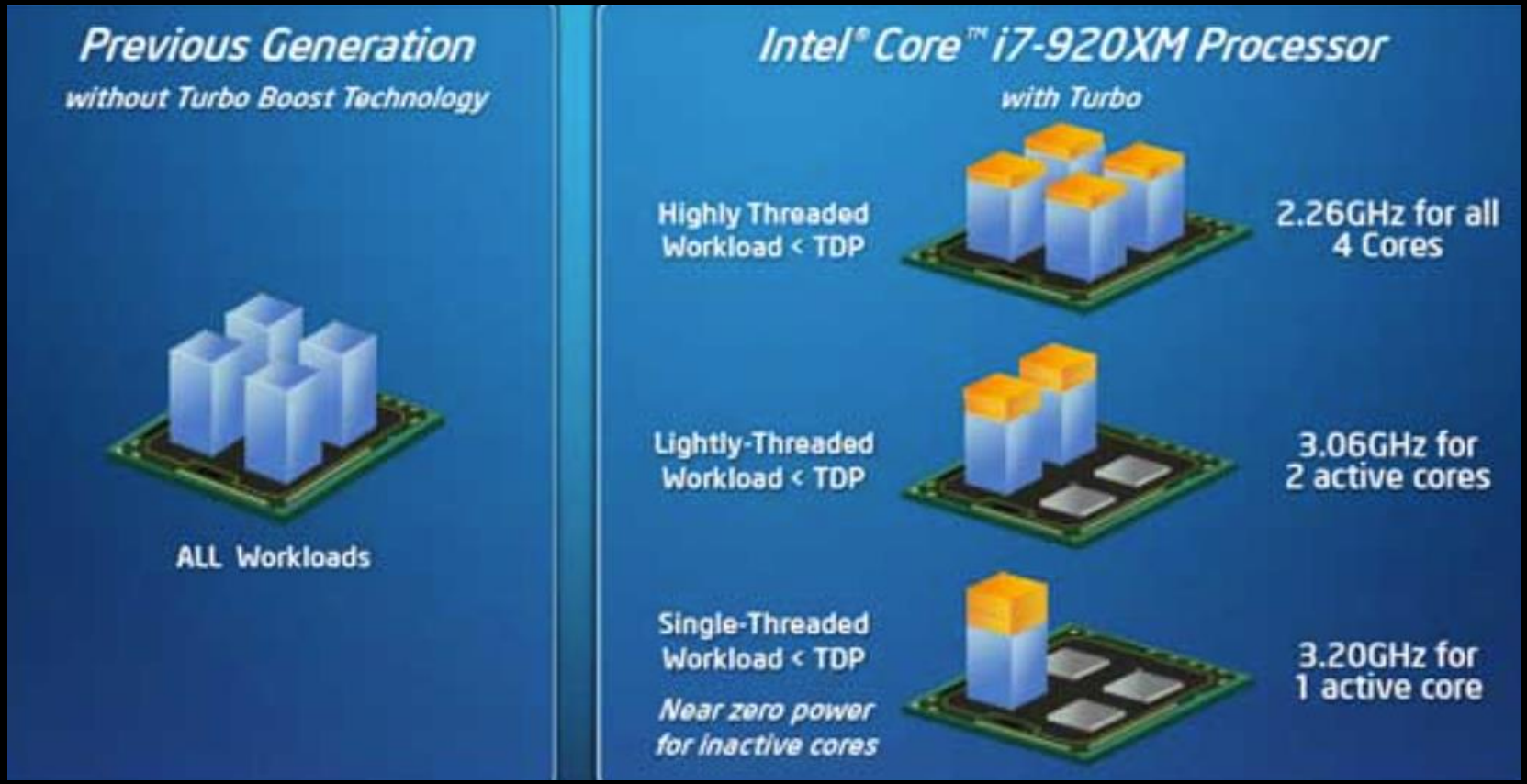


These are CPUs you can buy.

https://hwbot.org/benchmark/cpu_frequency/halloffame

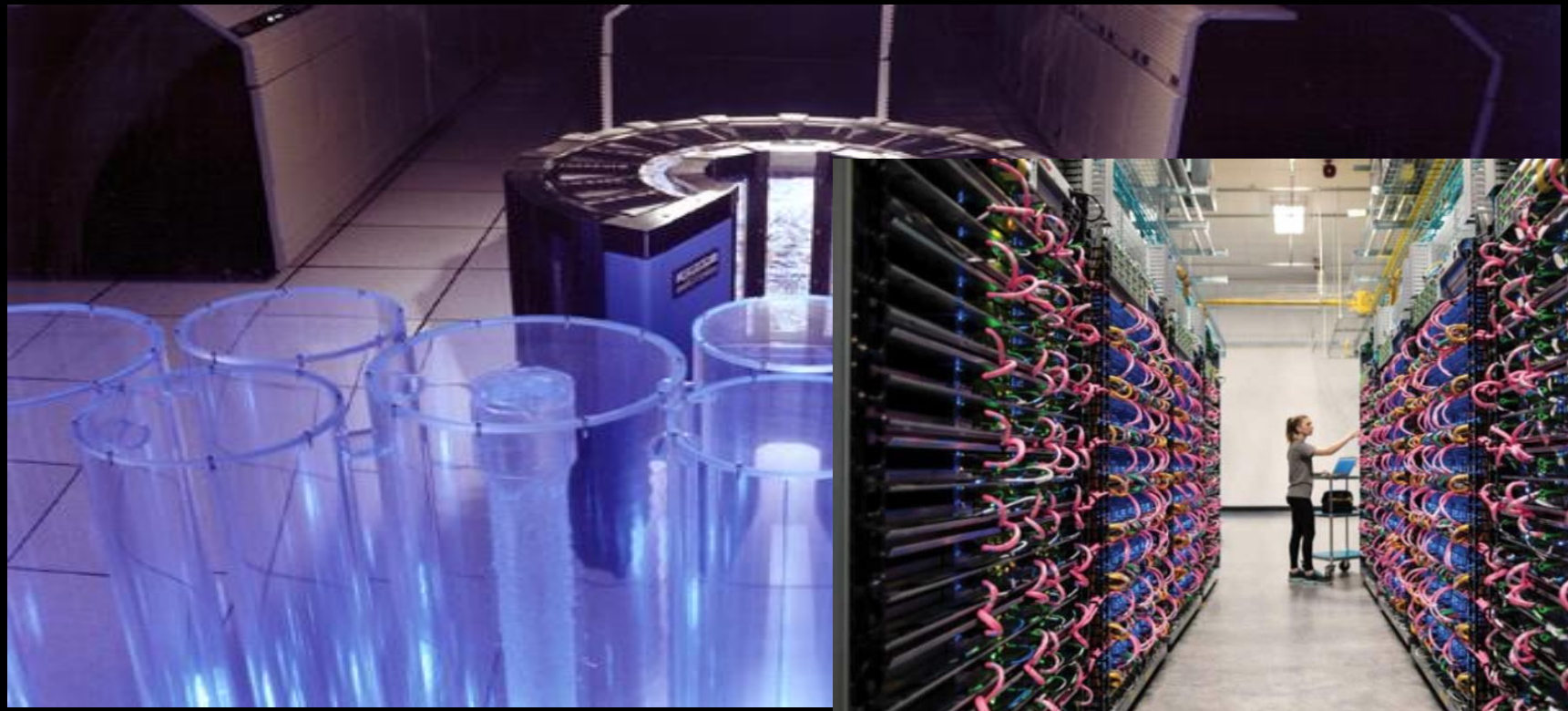
Complex liquid cooling on a consumer GPU.

For those of you thinking, "Well, at least *my* CPU runs at 4+ GHz."



Maybe sometimes.

Not a new problem…just ubiquitous.



Cray-2 with cooling tower i

Google's liquid cooled TPU v2 servers deployed in racks. *Source: Google.*

Starting to see 200KW per cabinet in datacenters.

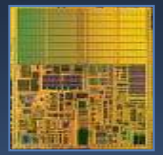
And how to get more performance from more transistors with the same power.

A 15% Reduction In Voltage Yields

RULE OF THUMB

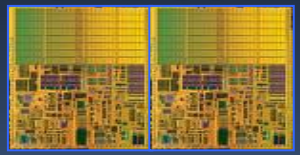
Frequency Reduction	Power Reduction	Performance Reduction
15%	45%	10%

SINGLE CORE



Area = 1
Voltage = 1
Freq = 1
Power = 1
Perf = 1

DUAL CORE

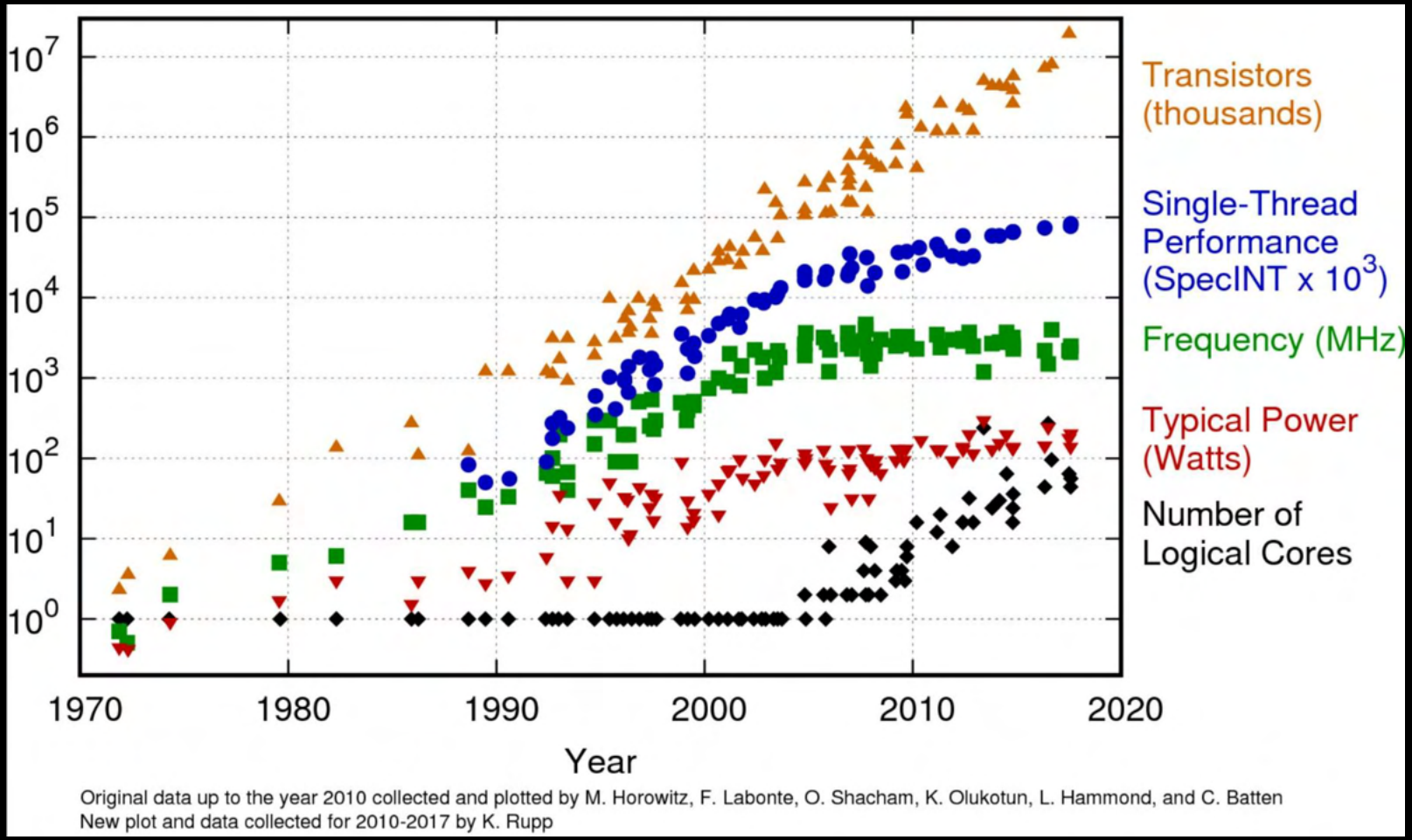


Area = 2
Voltage = 0.85
Freq = 0.85
Power = 1
Perf = ~1.8

Single Socket Parallelism

Processor	Year	Vector	Bits	SP FLOPs / core / cycle	Cores	FLOPs/cycle
Pentium III	1999	SSE	128	3	1	3
Pentium IV	2001	SSE2	128	4	1	4
Core	2006	SSE3	128	8	2	16
Nehalem	2008	SSE4	128	8	10	80
Sandybridge	2011	AVX	256	16	12	192
Haswell	2013	AVX2	256	32	18	576
KNC	2012	AVX512	512	32	64	2048
KNL	2016	AVX512	512	64	72	4608
Skylake	2017	AVX512	512	96	28	2688

Putting It All Together



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 2010-2017 by K. Rupp

Parallel Computing

One woman can make a baby in 9 months.

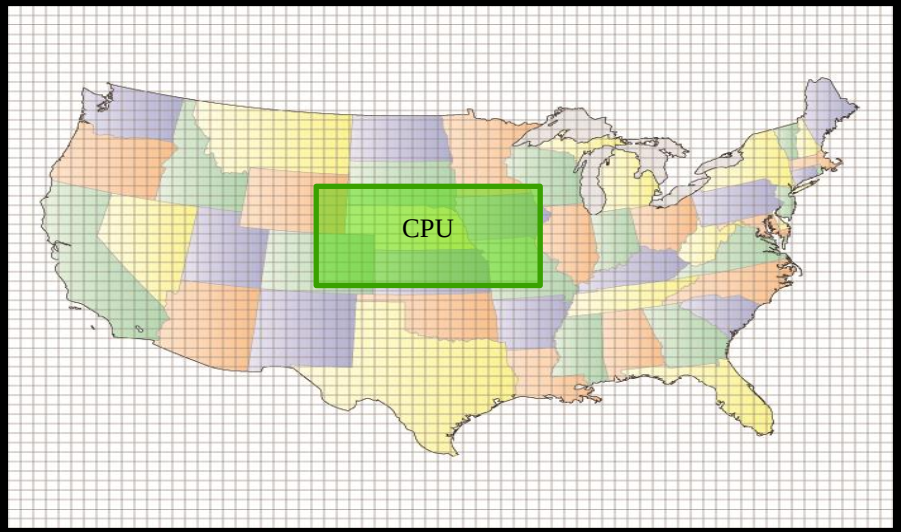
Can 9 women make a baby in 1 month?

But 9 women can make 9 babies in 9 months.

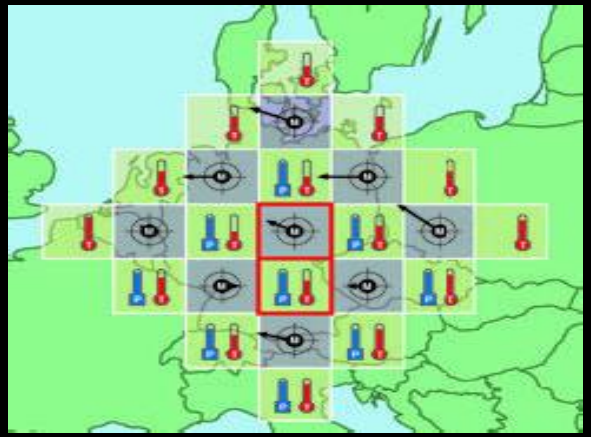
First two bullets are Brook's Law. From *The Mythical Man-Month.*

A must-read for serious project programmers that includes many other classics such as:
"What one programmer can do in one month, two programmers can do in two months."

Prototypical Application: Serial Weather Model

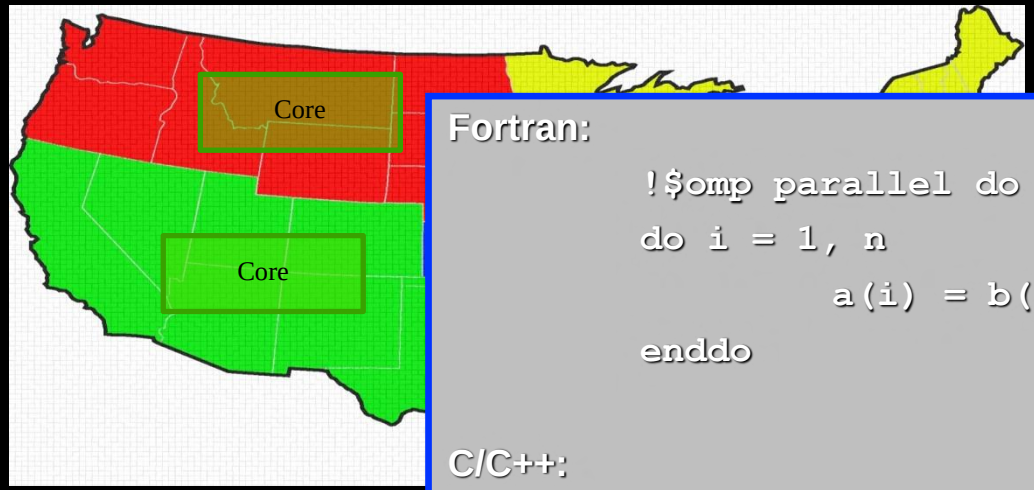


First Parallel Weather Modeling Algorithm: Richardson in 1917



Courtesy John Burkhardt, Virginia Tech

Weather Model: Shared Memory (OpenMP)



Four meteorologists in t

Fortran:

```
!$omp parallel do
do i = 1, n
        a(i) = b(i) + c(i)
enddo
```

C/C++:

```
#pragma omp parallel for
for(i=1; i<=n; i++)
        a[i] = b[i] + c[i];
```

V100 GPU and SM

Volta GV100 GPU with 85 Streaming Multiprocessor (SM) units

Volta GV100 SM

Rapid evolution continues with:

Turing
Ampere
Hopper

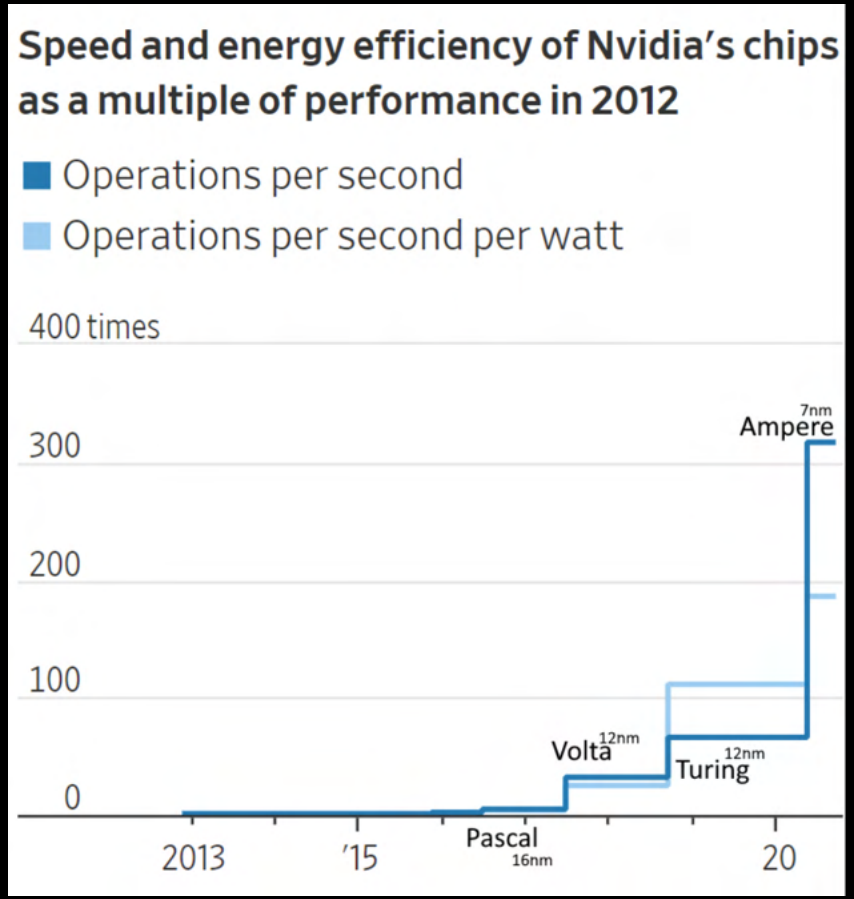
Huang's Law

An observation/claim made by Jensen Huang, CEO of Nvidia, at its 2018 GPU Technology Conference.

He observed that Nvidia's GPUs were "25 times faster than five years ago" whereas Moore's law would have expected only a ten-fold increase.

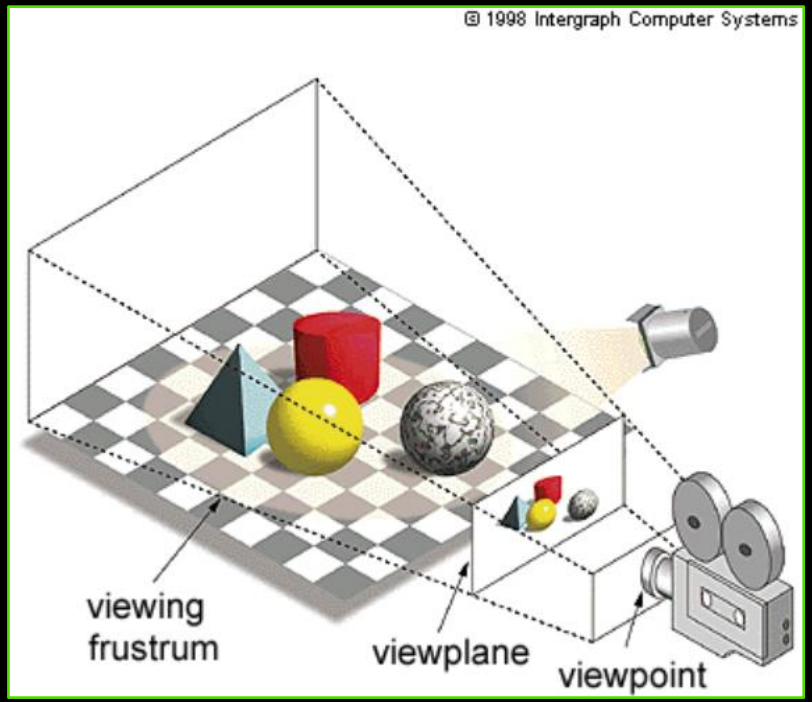
In 2006 Nvidia's GPU had a 4x performance advantage over other CPUs. In 2018 the Nvidia GPU was 20 times faster than a comparable CPU node: the GPUs were 1.7x faster each year. Moore's law would predict a doubling every two years, however Nvidia's GPU performance was more than tripled every two years fulfilling Huang's law.

It is a little premature, and there are confounding factors at play, so most people haven't yet elevated this to the status of Moore's Law.



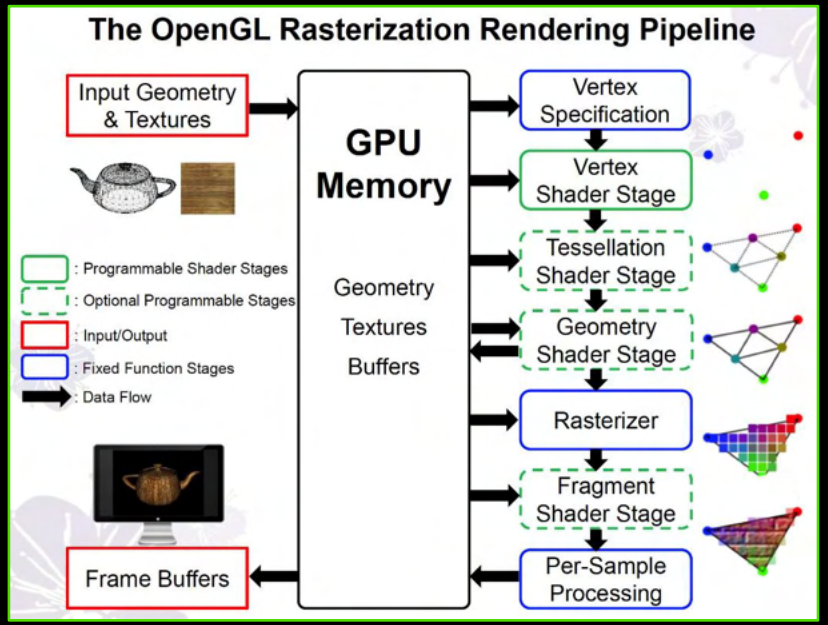
Source: NVIDIA

Why Video Gaming Cards?



By the turn of the century, the video gaming market has already standardized around a few APIs for rendering 3D video games in real-time.

None of these looked anything like scientific computing.



Heroic Efforts

An API in 2004 first demonstrated the potential use of this latent floating point ability.

By 2007 NVIDIA supported a dedicated API for their own hardware.

Note that these early devices were not at all engineered for scientific computing and lacked several very fundamental capabilities. In particular EEC and double precision.

Brook for GPUs: Stream Computing on Graphics Hardware

Ian Buck Tim Foley Daniel Horn Jeremy Sugerman Kayvon Fatahalian Mike Houston Pat Hanrahan
Stanford University

Abstract

In this paper, we present Brook for GPUs, a system for general-purpose computation on programmable graphics hardware. Brook extends C to include simple data-parallel constructs, enabling the use of the GPU as a streaming co-processor. We present a compiler and runtime system that abstracts and virtualizes many aspects of graphics hardware. In addition, we present an analysis of the effectiveness of the GPU as a compute engine compared to the CPU, to determine when the GPU can outperform the CPU for a particular algorithm. We evaluate our system with five applications, the SAXPY and SGEMV BLAS operators, image segmentation, FFT, and ray tracing. For these applications, we demonstrate that our Brook implementations perform comparably to hand-written GPU code and up to seven times faster than their CPU counterparts.

CR Categories: I.3.1 [Computer Graphics]: Hardware Architecture—Graphics processors D.3.2 [Programming Languages]: Language Classifications—Parallel Languages

Keywords: Programmable Graphics Hardware, Data Parallel Computing, Stream Computing, GPU Computing, Brook

1 Introduction

In recent years, commodity graphics hardware has rapidly evolved from being a fixed-function pipeline into having programmable vertex and fragment processors. While this new modern hardware. In addition, the user is forced to express their algorithm in terms of graphics primitives, such as textures and triangles. As a result, general-purpose GPU computing is limited to only the most advanced graphics developers.

This paper presents *Brook*, a programming environment that provides developers with a view of the GPU as a streaming coprocessor. The main contributions of this paper are:

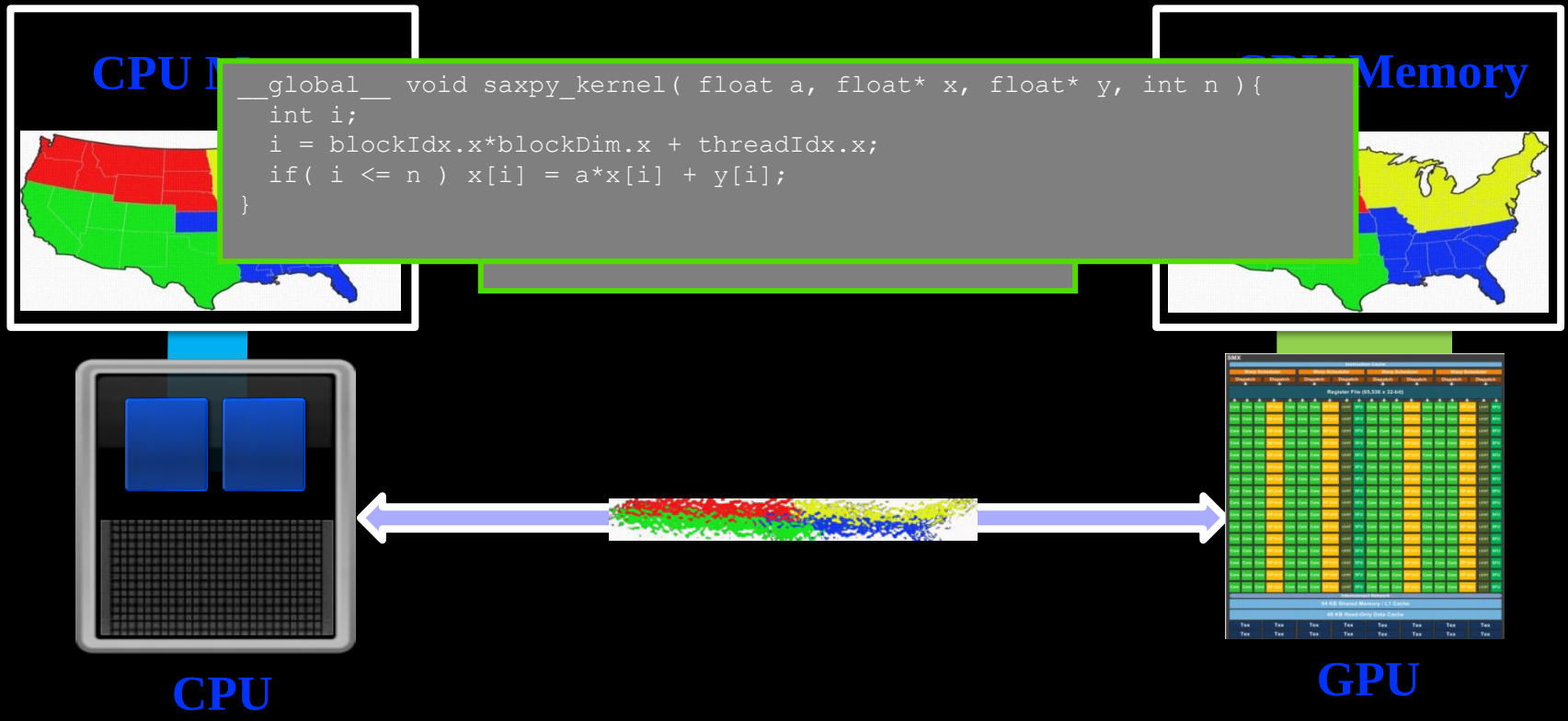
- The presentation of the Brook stream programming model for general-purpose GPU computing. Through the use of streams, kernels and reduction operators, Brook abstracts the GPU as a streaming processor.
- The demonstration of how various GPU hardware limitations can be virtualized or extended using our compiler and runtime system; specifically, the GPU memory system, the number of supported shader outputs, and support for user-defined data structures.
- The presentation of a cost model for comparing GPU vs. CPU performance tradeoffs to better understand under what circumstances the GPU outperforms the CPU.

2 Background

2.1 Evolution of Streaming Hardware

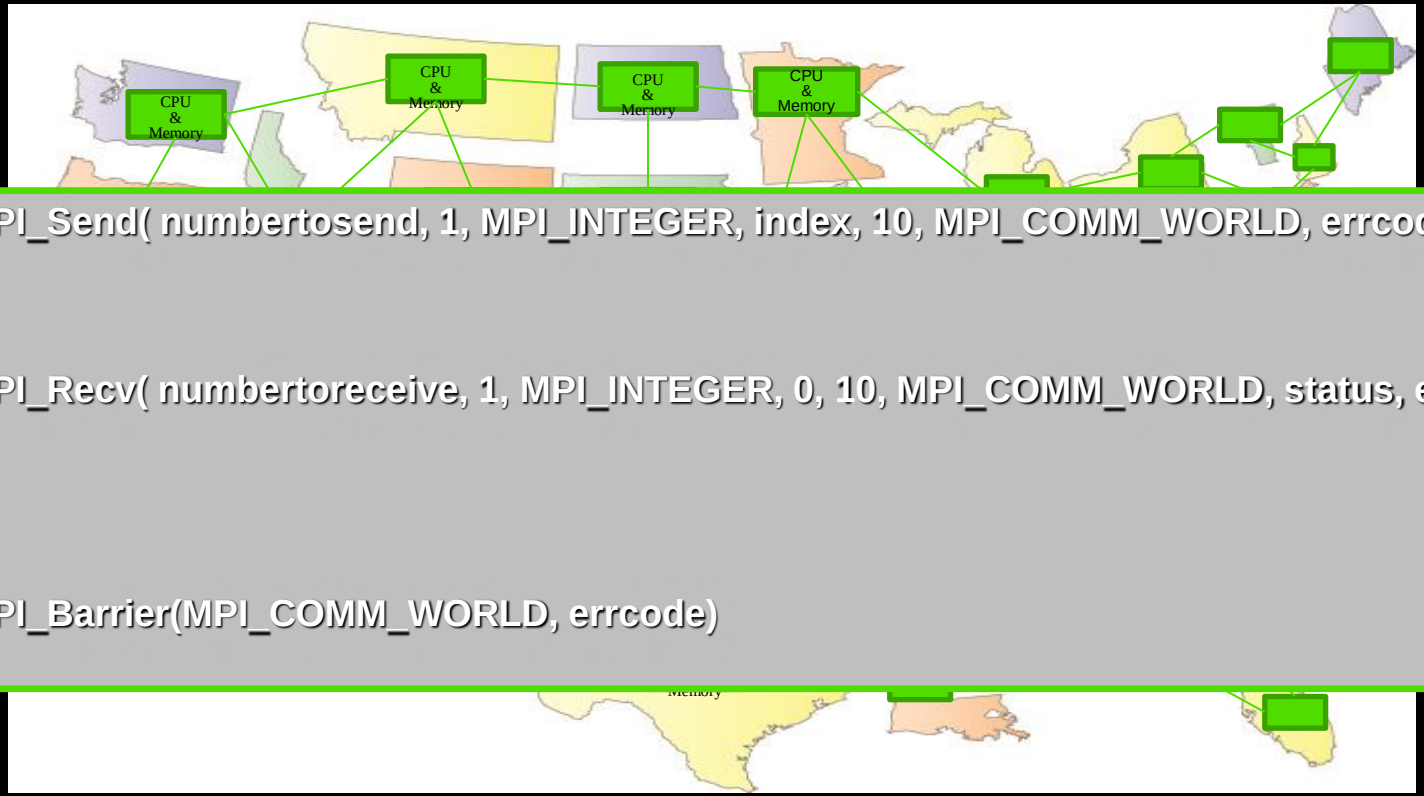
Programmable graphics hardware dates back to the original programmable framebuffer architectures [England 1986].

Weather Model: Accelerator (OpenACC)



1 meteorologists coordinating 1000 math savants using tin cans and a string.

Weather Model: Distributed Memory (MPI)




```
call MPI_Send( numbertosend, 1, MPI_INTEGER, index, 10, MPI_COMM_WORLD, errcode)
.
.
call MPI_Recv( numbertoreceive, 1, MPI_INTEGER, 0, 10, MPI_COMM_WORLD, status, errcode)
.
.
.
call MPI_Barrier(MPI_COMM_WORLD, errcode)
.
```

50 meteorologists using a telegraph.

MPPs (Massively Parallel Processors)

Distributed memory at largest scale. Shared memory at lower level.

Summit (ORNL)

- 122 PFlops Rmax and 187 PFlops Rpeak
- IBM Power 9, 22 core, 3GHz CPUs
- 2,282,544 cores
- NVIDIA Volta GPUs
- EDR Infiniband

Sunway TaihuLight (NSC, China)

- 93 PFlops Rmax and 125 PFlops Rpeak
- Sunway SW26010 260 core, 1.45GHz CPU
- 10,649,600 cores
- Sunway interconnect

Many Levels and Types of Parallelism

- Vector (SIMD)
- Instruction Level (ILP)
  - Instruction pipelining
  - Superscaler (multiple instruction units)
  - Out-of-order
  - Register renaming
  - Speculative execution
  - Branch prediction

Compiler
(not your problem)

OpenMP 4/5 can help!

OpenMP

- Multi-Core (Threads)
- SMP/Multi-socket

OpenACC

- Accelerators: GPU & MIC

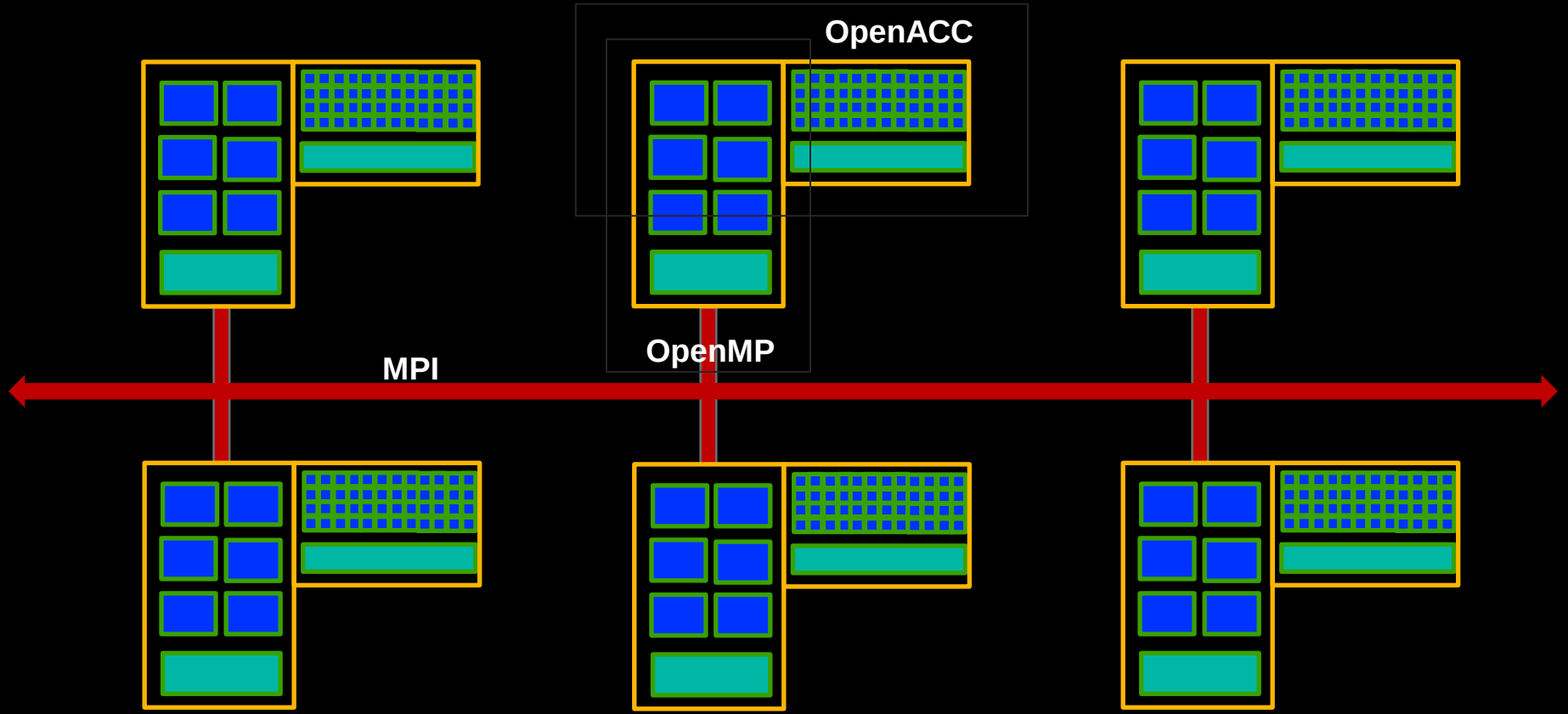
MPI

- Clusters
- MPPs

Also Important

- ASIC/FPGA/DSP
- RAID/IO

The pieces fit like this…



Cores, Nodes, Processors, PEs?

- A "core" can run an independent thread of code. Hence the temptation to refer to it as a processor.
- “Processors” refer to a physical chip. Today these almost always have more than one core.
- “Nodes” is used to refer to an actual physical unit with a network connection; usually a circuit board or "blade" in a cabinet. These often have multiple processors.
- To avoid ambiguity, it is precise to refer to the smallest useful computing device as a Processing Element, or PE. On normal processors this corresponds to a core.

I will try to use the term PE consistently myself, but I may slip up. Get used to it as you will quite often hear all of the above terms used interchangeably where they shouldn’t be. Context usually makes it clear.

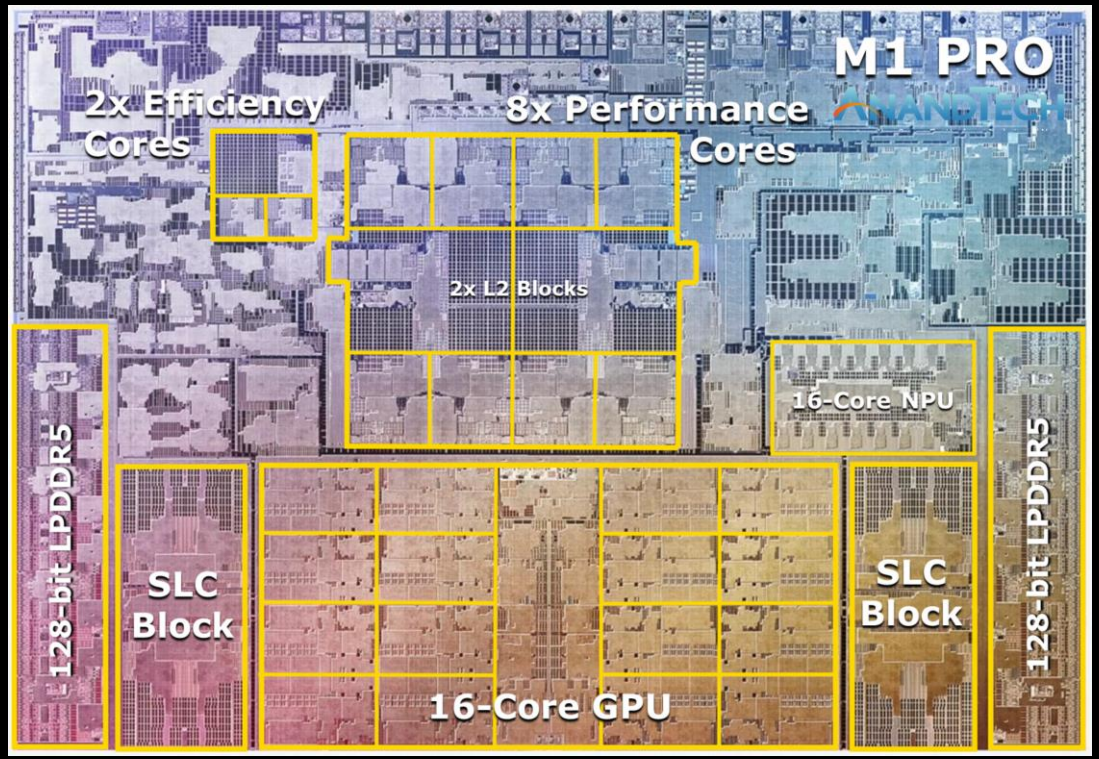
Top 10 Systems as of June 2023

#	Computer	Site	Manufacturer	CPU Interconnect [*Accelerator*]	Cores	Rmax (Pflops)	Rpeak (Pflops)	Power (MW)
1	**Frontier**	Oak Ridge National Laboratory **United States**	HPE	AMD EPYC 64C 2GHz Slingshot-11 AMD Instinct MI250X	**8,699,904**	**1194**	**1692**	**22.7**
2	**Fugaku**	RIKEN Center for Computational Science **Japan**	Fujitsu	ARM 8.2A+ 48C 2.2GHz Torus Fusion Interconnect	**7,630,072**	**442**	**537**	**29.9**
3	**LUMI**	EuroHPC **Finland**	HPE	AMD EPYC 64C 2GHz Slingshot-11 AMD Instinct MI250X	**2,220,288**	**309**	**428**	**6.0**
4	**Leonardo**	EuroHPC **Italy**	Atos	Intel Xeon 8358 32C 2.6GHz Infiniband HDR NVIDIA A100	**1,824,768**	**238**	**304**	**7.4**
5	**Summit**	Oak Ridge National Laboratory **United States**	IBM	Power9 22C 3.0 GHz Dual-rail Infiniband EDR *NVIDIA V100*	**2,414,592**	**148**	**200**	**10.1**
6	**Sierra**	Lawrence Livermore National Laboratory **United States**	IBM	Power9 3.1 GHz 22C Infiniband EDR *NVIDIA V100*	**1,572,480**	**95**	**125**	**7.4**
7	**Sunway TaihuLight**	National Super Computer Center in Wuxi **China**	NRCPC	Sunway SW26010 260C 1.45GHz Sunway Interconnect	**10,649,600**	**93**	**125**	**15.3**
8	**Perlmutter**	NERSC **United States**	HPE	EPYC 64C 2.45 GHz Slingshot-10 *NVIDIA A100*	**761,304**	**70**	**93**	**2.6**
9	**Selene**						**79**	**2.6**
10	**Tiahne-2A**						**101**	**18.4**

500	Inspur TS10000, Xeon Gold 6130 16C 2.1GHz, NVIDIA Tesla V100, 25G Ethernet, Inspur Internet Service P China	40,320	1.87	3.52

The word is *Heterogeneous*

And it's not just supercomputers. It's on your desk, and in your phone.



AnandTech

How much of this can you program?

Networks

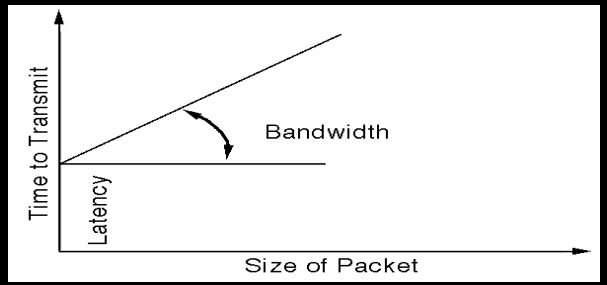
3 characteristics sum up the network:

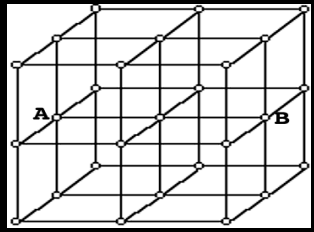
- **Latency**

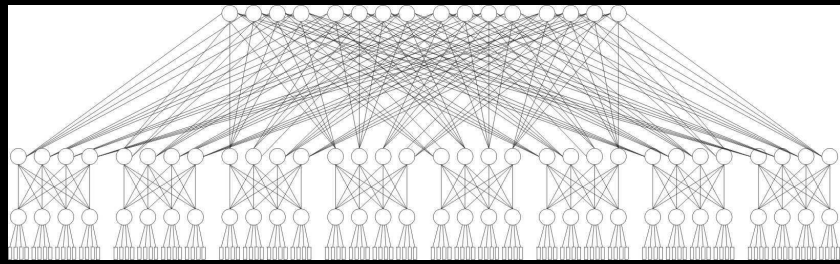
  The time to send a 0 byte packet of data on the network

- **Bandwidth**

  The rate at which a very large packet of information can be sent



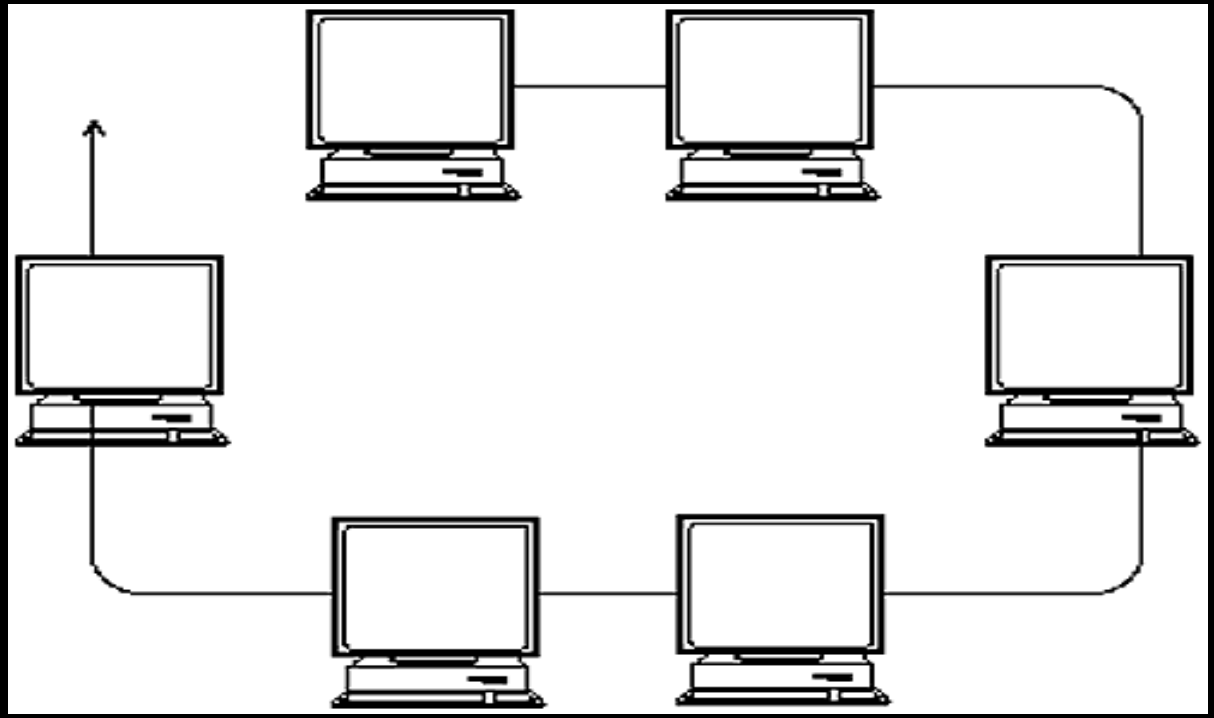




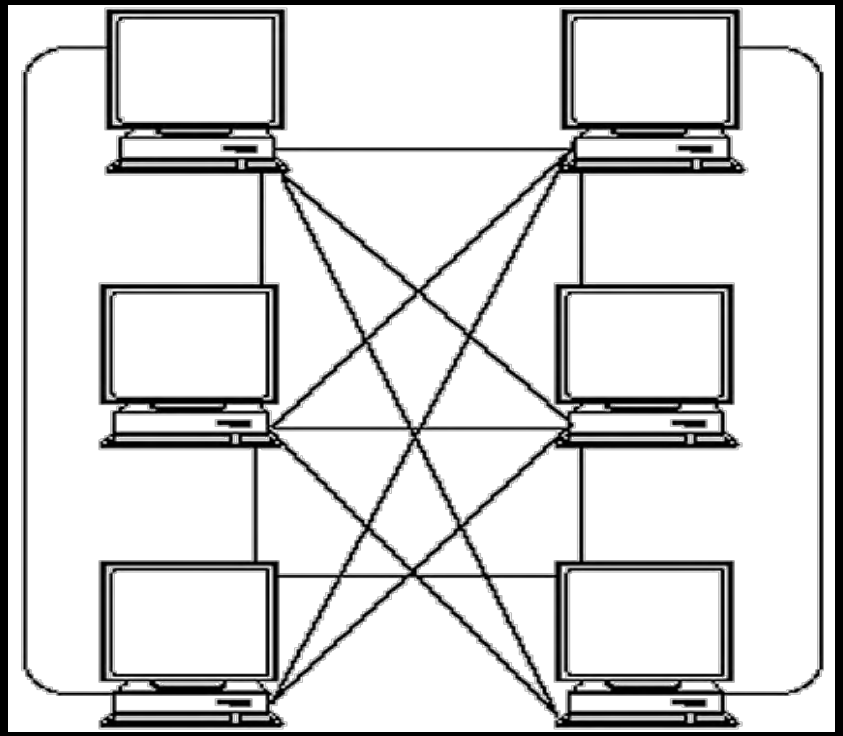
- **Topology**

  The configuration of the network that determines how processing units are directly connected.

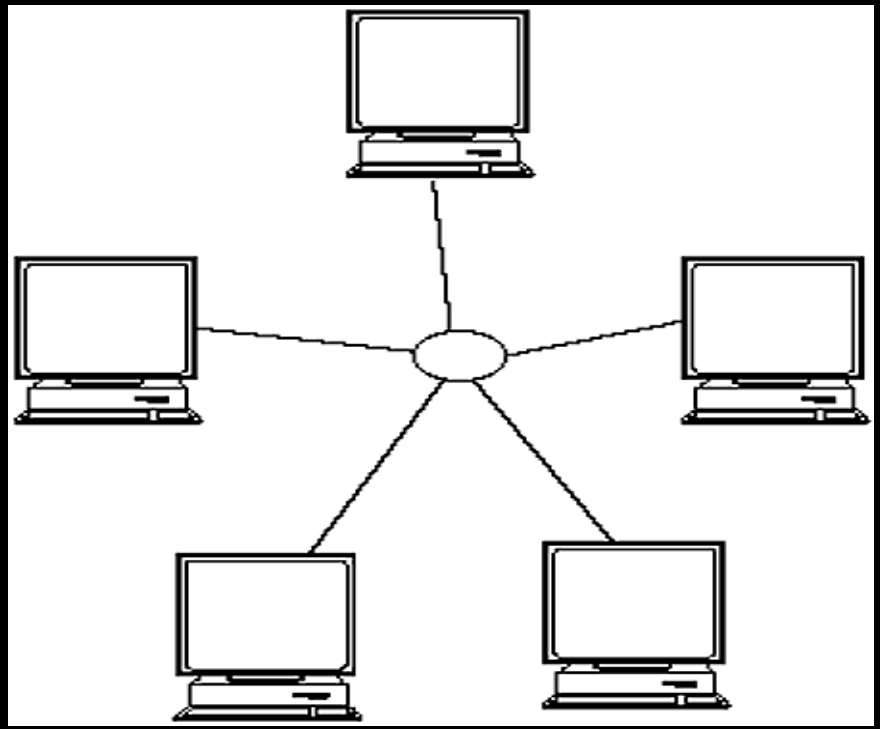
Ethernet with Workstations



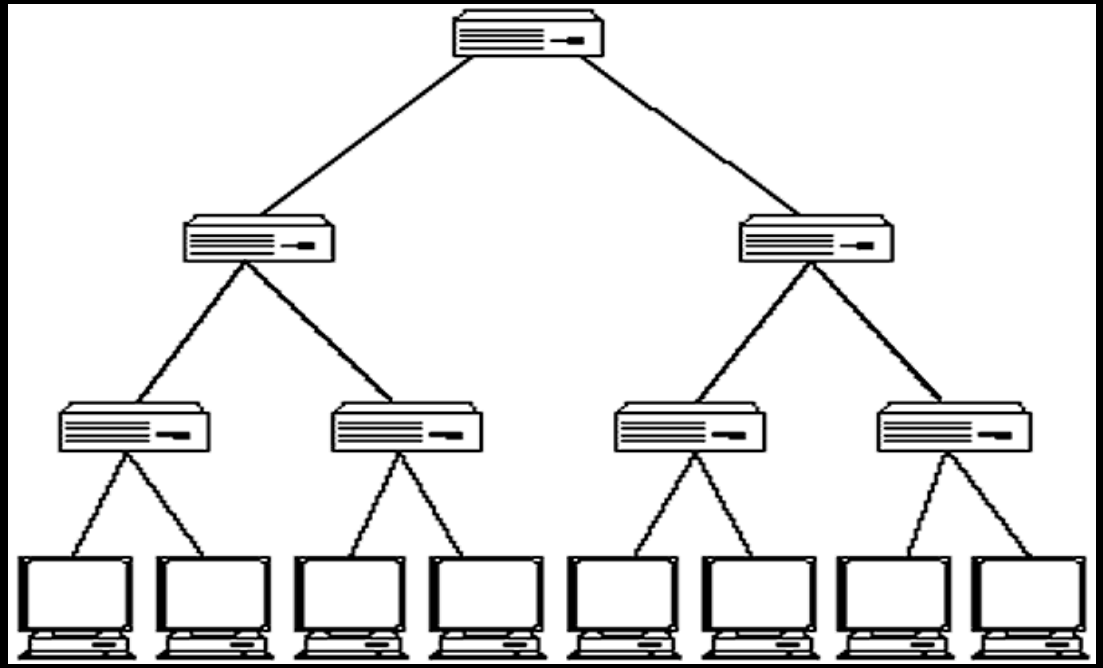
Complete Connectivity



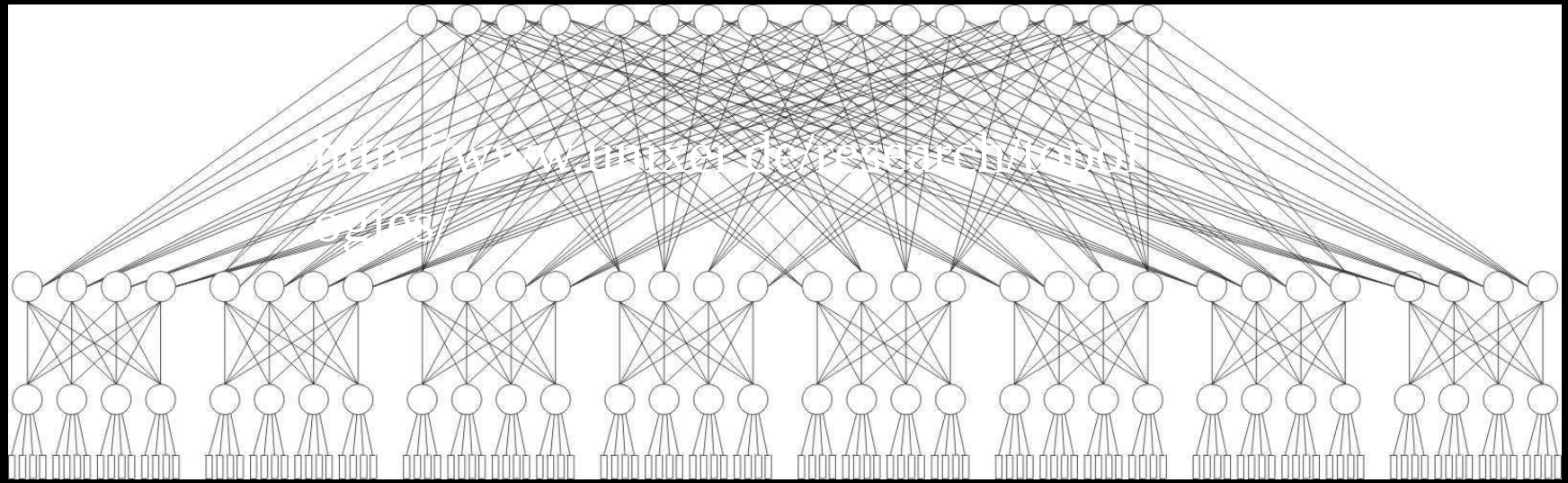
Crossbar



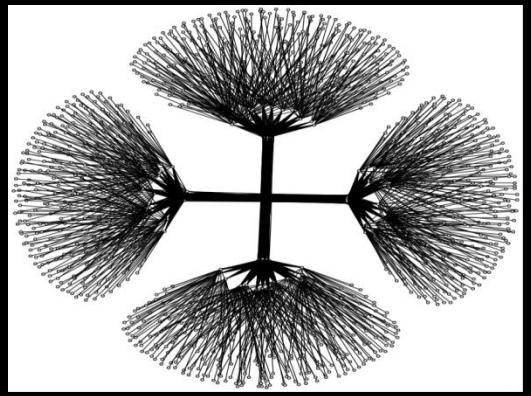
Binary Tree



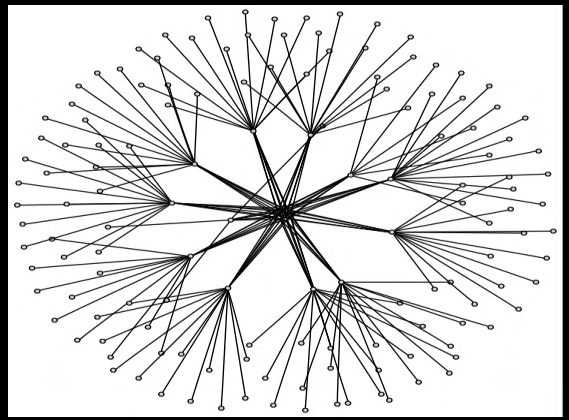
Fat Tree



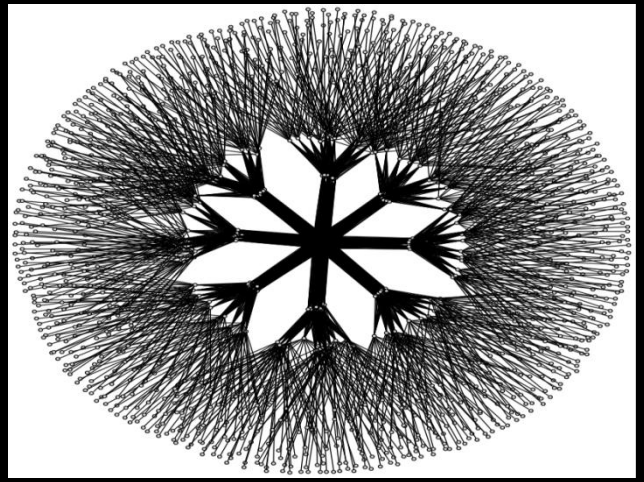
Other Fat Trees



Big Red @ IU

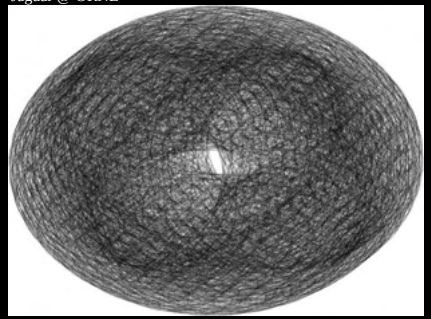


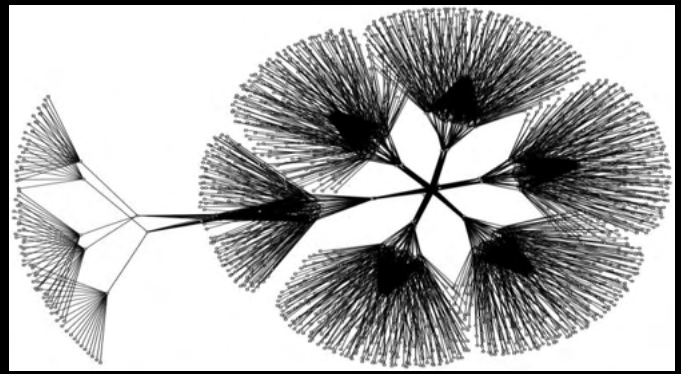
Odin @ IU



Atlas @ LLNL

Jaguar @ ORNL





Tsubame @ Tokyo Inst. of Tech

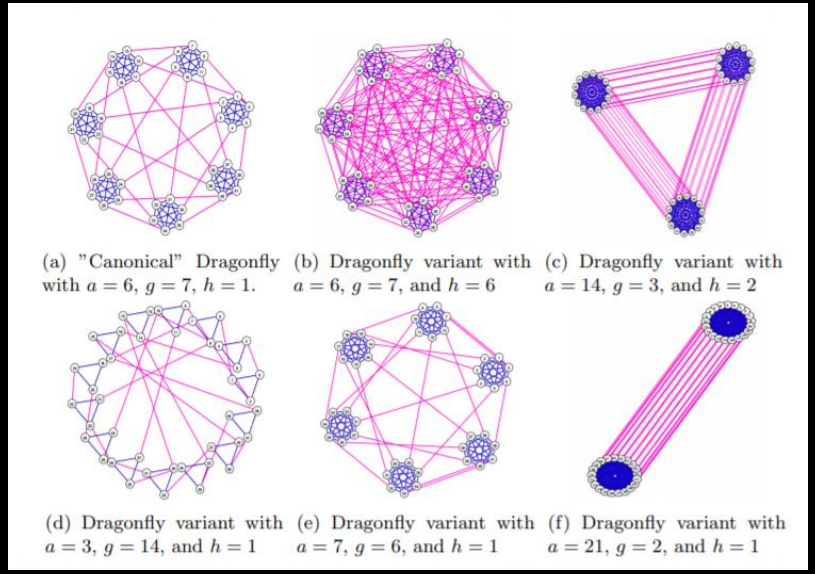
From Torsten Hoefler's Network Topology Repository at
http://www.unixer.de/research/topologies/

Dragonfly

A newer innovation in network design is the dragonfly topology, which benefits from advanced hardware capabilities like:

- **High-Radix Switches**
- **Adaptive Routing**
- **Optical Links**

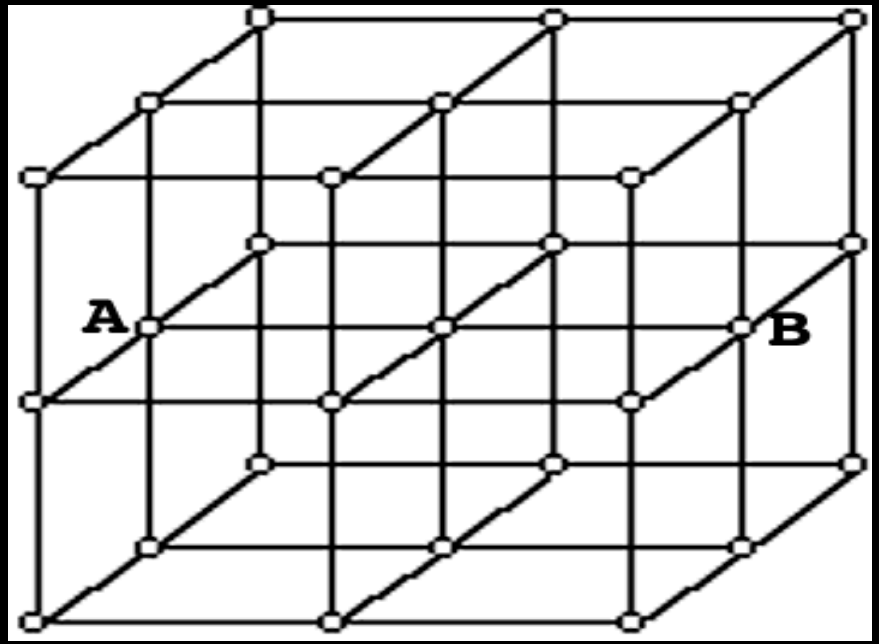
Various 42 node Dragonfly configurations.



Purple links are optical, and blue are electrical.

Graphic from the excellent paper *Design space exploration of the Dragonfly topology* by Yee, Wilke, Bergman and Rumley.

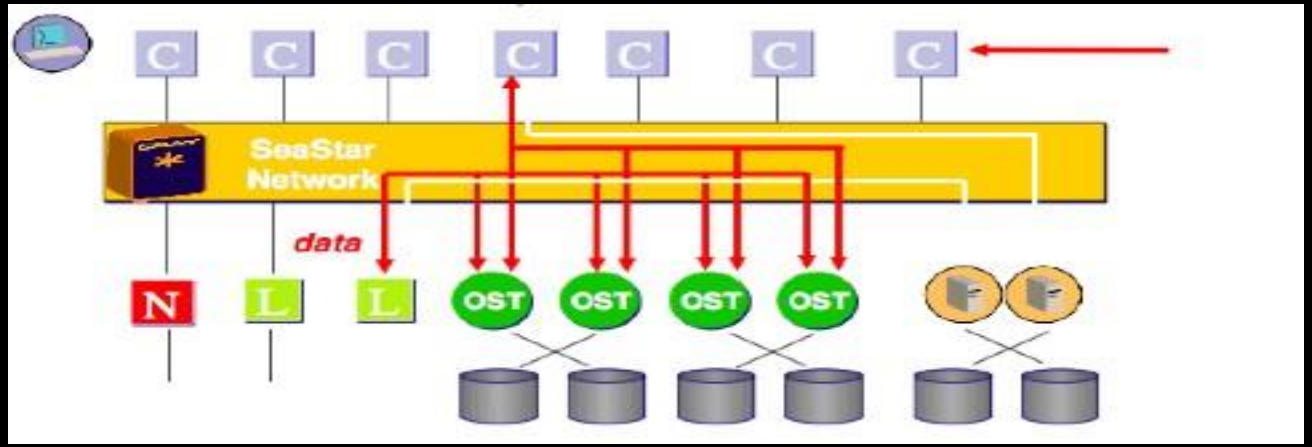
3-D Torus



Torus simply means that “ends” are connected. This means A is really connected to B and the cube has no real boundary.

Parallel IO (RAID…)

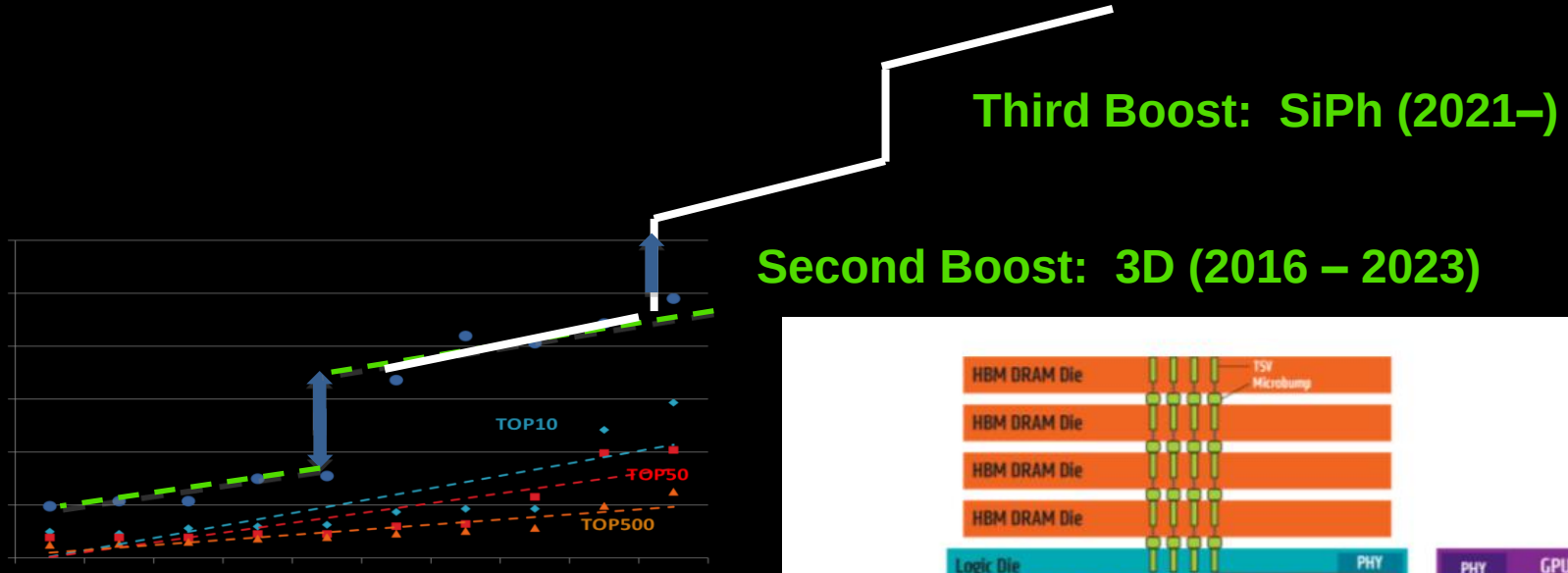
- There are increasing numbers of applications for which many PB of data need to be written.
- Checkpointing is also becoming very important due to MTBF issues (a whole ‘nother talk).
- Build a large, fast, reliable filesystem from a collection of smaller drives.
- Supposed to be transparent to the programmer.
- Increasingly mixing in SSD.



Today

- Pflops computing fully established with more than 500 machines
- The field is thriving
- Interest in supercomputing is now worldwide, and growing in many new markets
- Exascale projects in many countries and regions

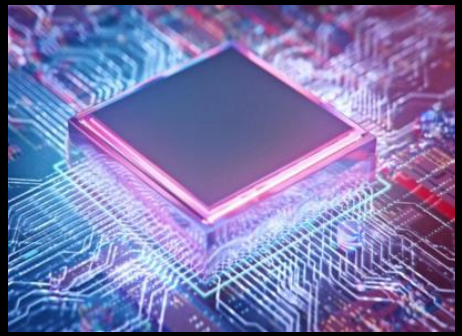
The path to Exascale has not been incremental.



Is Silicon Photonics a game changer?

Electrically switched networks can operate in “packet switching” mode to lower the effective latency and utilize all the available link bandwidth. The alternative to this mode is “circuit-switching” and it was abandoned by the electronic community long ago. Without practical means to buffer light, process photon headers in-flight, or reverting to switches with expensive optical-electrical-optical conversions, we would have to resort to circuit-switching with all the inherent deficiencies:

- **complex traffic steering calculations**
- **switching delays**
- **latency increase due to lack of available paths**
- **under-utilization of links**



Photonics is often cited as an enabler for extensive memory disaggregation, but this yields another challenge, specifically the speed of light. Photons travel at a maximum speed of 3.3 ns/m in fibers. This is equivalent to a level-2 cache access of a modern CPU, not including the disaggregation overhead (such as from the protocol, switching, or optical-electrical conversions at the endpoints). At 3–4 m distance, the photon travel time alone exceeds the first-word access latency of modern DDR memory.

A great dive into these topics can be found in *Myths and Legends in High-Performance Computing*, Matsuoka, Domke, et. al.

It is not just “exaflops” – we are changing the whole computational model

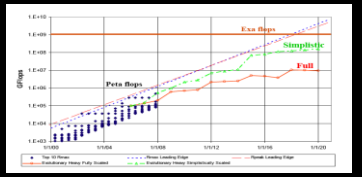
Current programming systems have WRONG optimization targets

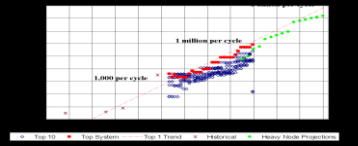
Old Constraints

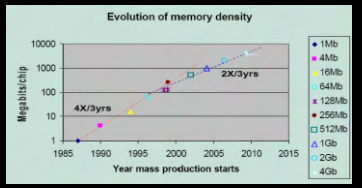
- **Peak clock frequency *as primary limiter for performance improvement***
- **Cost: *FLOPs* are biggest cost for system: *optimize for compute***
- **Concurrency: Modest growth of parallelism by adding nodes**
- **Memory scaling*: maintain byte per flop capacity and bandwidth***
- **Locality*: MPI+X model (uniform costs within node & between nodes)***
- **Uniformity: *Assume uniform system performance***
- **Reliability*: It’s the hardware’s problem***

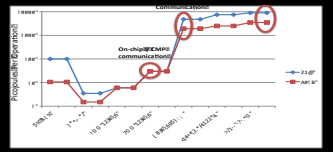
New Constraints

- **Power** *is primary design constraint for future HPC system design*
- **Cost:** *Data movement dominates: optimize to minimize data movement*
- **Concurrency:** *Exponential growth of parallelism within chips*
- **Memory Scaling:** *Compute growing 2x faster than capacity or bandwidth*
- **Locality***: must reason about data locality and possibly topology*
- **Heterogeneity**: *Architectural and performance non-uniformity increase*
- **Reliability**: *Cannot count on hardware protection alone*









Fundamentally breaks our current programming paradigm and computing ecosystem

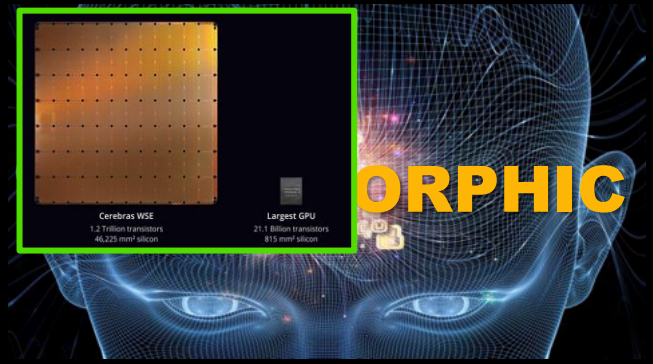
Adapted from John Shalf

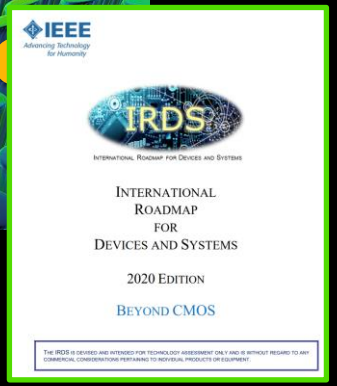
End of Moore's Law Will Lead to New Architectures

Non-von Neumann

ARCHITECTURE

von Neumann



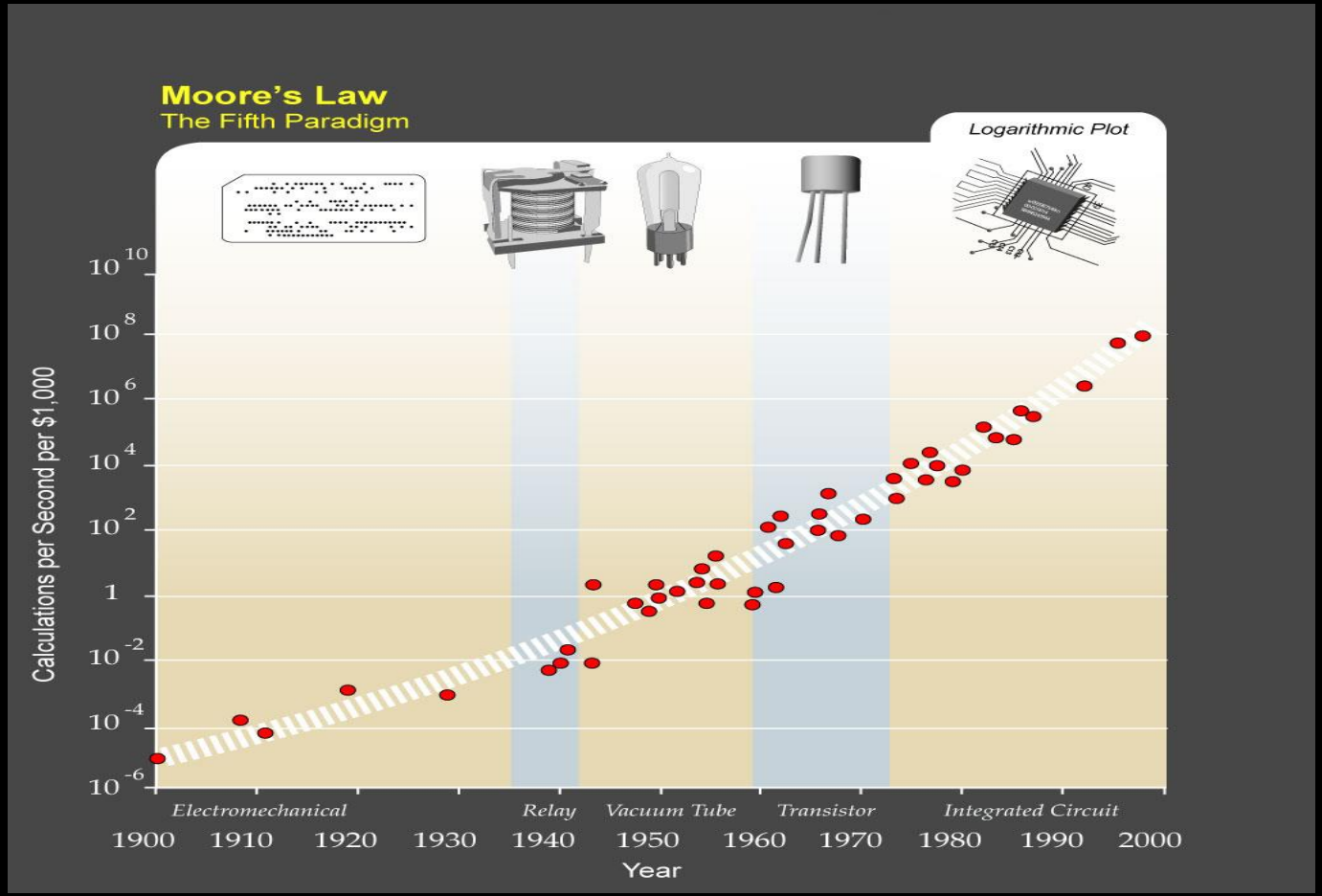


CMOS

Beyond CMOS

TECHNOLOGY

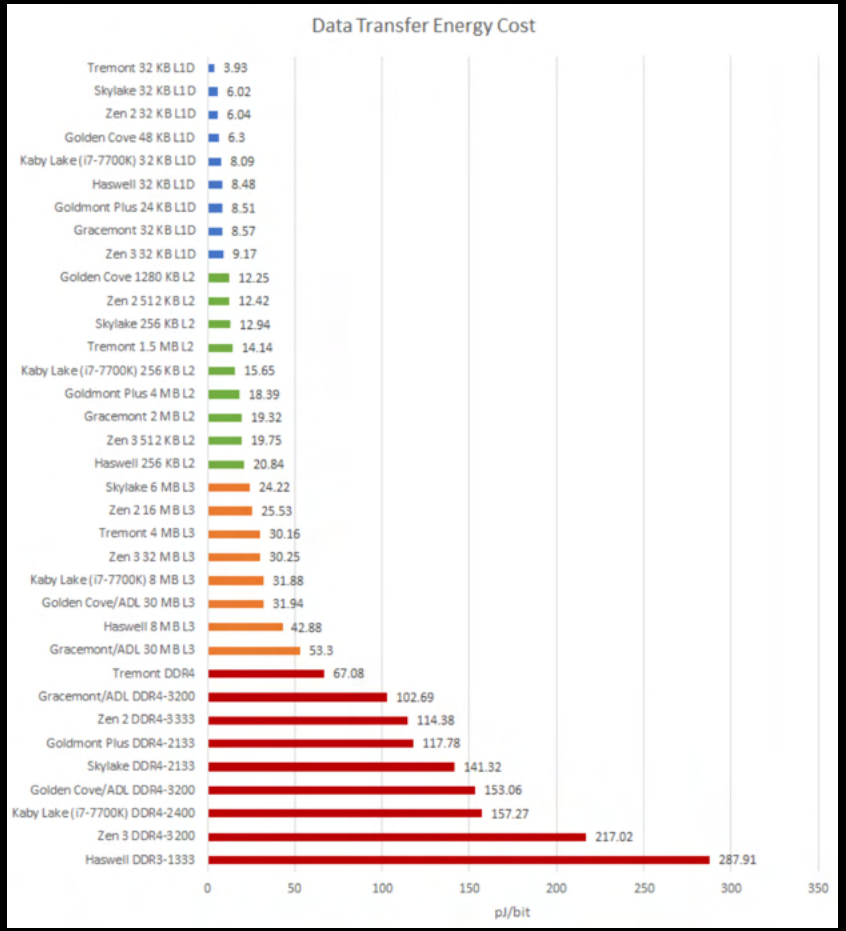
It would only be the 6th paradigm.



We can do better. We have a role model.

- We hope to "simulate" a human brain in real time on one of these Exascale platforms with about 1 - 10 Exaflop/s and 4 PB of memory
- These newest Exascale computers use 20+ MW
- The human brain runs at 20W
- Our brain is a million times more power efficient!

On a related note.



- Laughlin was the first to provide explicit quantities for the energetic cost of processing sensory information. Their findings in blowflies revealed that for visual sensory data, the cost of transmitting one bit of information is around 50 fJ (5×10^{-14} Joules), or equivalently 104 ATP molecules.
- The units on this graph are pJ, 1000X larger. Thus, neural processing efficiency is still far from Landauer's limit of kTln(2) J, but still considerably more efficient than a modern computer’s near memory. For far (MPI network, or further) accesses it is a huge difference.

Why you should be (extra) motivated.

- This parallel computing thing is no fad.
- The laws of physics are drawing this roadmap.
- If you get on board (the right bus), you can ride this trend for a long, exciting trip.

Let’s learn how to use these things!

In Conclusion…