AMS 250: An Introduction to High Performance Computing

Overview

Shawfeng Dong

shaw@ucsc.edu

(831) 502-7743

Applied Mathematics & Statistics

University of California, Santa Cruz

Outline

- Course Overview
    - What is AMS 250
    - What is expected of you
    - What will you learn in AMS 250
- High Performance Computing (HPC)
    - What is HPC
    - What motivates HPC
    - Trends that shape the field
    - Large-scale problems and high-performance computing
    - Parallel architecture types
    - Scalable parallel computing and performance

What is AMS 250

- Successor to *AMS 290B: An Introduction to Parallel Computing and Large Computational Fluid Dynamics Codes*:
  - https://classes.soe.ucsc.edu/ams290b/Winter08/
- AMS 250 is a graduate course that introduces students to the modern world of cutting-edge supercomputing
- AMS 250 was inaugurated by Prof. Nic Brummell in Spring 2015:
  - https://courses.soe.ucsc.edu/courses/ams250/Spring15/01
- My lectures are also heavily influenced by the *Parallel Computing* course at University of Oregon:
  - http://ipcc.cs.uoregon.edu/curriculum.html

What is expected of you

- Fledgling Computational Scientists
- Computer Scientists and Engineers can benefit from this course as well
- Have taken *AMS 209: Foundation of Scientific Computing*; or equivalent
  https://courses.soe.ucsc.edu/courses/ams209
- Reasonably proficient in any, preferably all, of the following languages:
  - C/C++
  - Modern Fortran
  - Python, particularly NumPy
  - Java

Course Web Sites

- Drupal Site:
  https://ams250-spring18-01.courses.soe.ucsc.edu/
- Google Classroom:
  http://classroom.google.com/c/MTIwMDgwNTI1MDBa
  Sign in with your Google Apps for Education account (*@ucsc.edu*)
  You should have all got invitations to join in the Classroom
  You can also Join in with the code *avp55nh*
- AMS 250 on GitHub:
  https://github.com/shawfdong/ams250

Tentative Syllabus

- PART A: CONCEPTS
  - Parallel Computer Architectures
  - Parallel programming models
  - Parallel Programming Patterns & Algorithms
- PART B: TOOLS
  - Shared Memory Programming with OpenMP
  - Distributed Memory Programming with MPI
  - Debugging & Performance Optimization
  - Manycore Computing (GPU, MIC, FPGA)
- PART C: Advanced Topics
  - Parallel Math Libraries
  - Parallel IO
  - Distributed Machine Learning
  - Advanced MPI
  - PGAS

Course Materials

- Major reading materials are lectures notes and references therein
- Supplemental textbooks:
  - *Programming on Parallel Machines*, *by* Norm Matloff, UC Davis

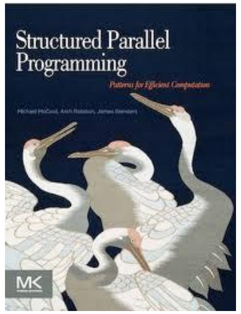
    Open Textbook: http://heather.cs.ucdavis.edu/parprocbook

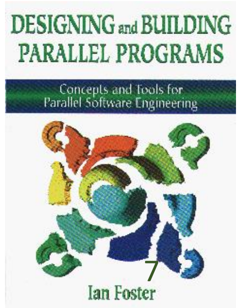
  - *Structured Parallel Programming: Patterns for Efficient Computation*, *by* Michael McCool, Arch Robinson, & James Reinders, Morgan Kaufmann, 2012

    PDF: http://www.sciencedirect.com/science/book/9780124159938

  - *Designing and Building Parallel Programs*, *by* Ian Foster, Addison Wesley, 1995

    http://www.mcs.anl.gov/~itf/dbpp/text/book.html

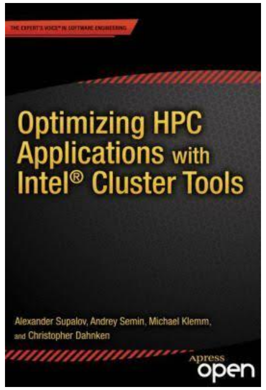




Course Materials

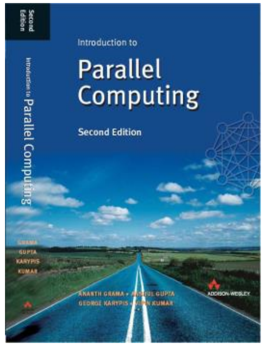
- Supplemental textbooks (cont'd):
  - *Optimizing HPC Applications with Intel Cluster Tools*, *by* Alexander Supalov, Andrey Semin, Michael Klemm, & Christopher Dahnken, Apress, 2014

    Free eBook: http://www.apress.com/9781430264965



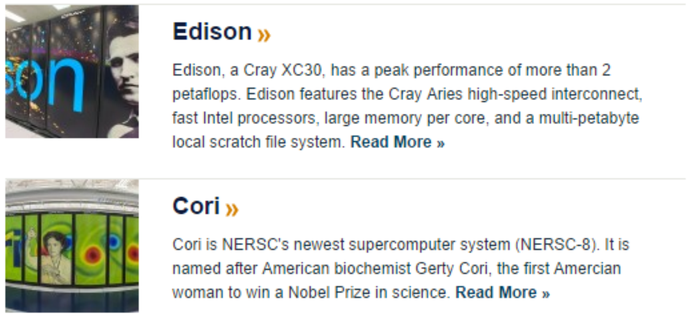
  - *Introduction to Parallel Computing*, *by* Ananth Grama, Anshul Gupta, George Karypis, & Vipin Kumar, Addison Wesley, 2nd Ed., 2003

    http://www-users.cs.umn.edu/~karypis/parbook/



Computing Resources

- NERSC Supercomputers
  - We've received an Education Allocation Award from NERSC
  - You'll hone your skills in HPC on petascale supercomputers at NERSC
- GPU Box
  - PRP has provided us with a GPU box, which is equipped with 4 Nvidia GeForce GTX 1080 Ti GPUs
- Your Own Laptops or Workstations
  - All modern computers are parallel machines



Grading Policy

- Homework (60%)
  - 4 simple programming assignments to help you understand the course materials
  - Homework will be assigned every 2 weeks on Tuesdays, starting from the 1st week
  - Homework will be due 2 weeks from the assignment date
  - Homework will be submitted to Google Classroom site
  - Penalty for late homework submission
    - You are going to receive a maximum of 80% if late by less than 1 day
    - 50% if late by more than a day
- Final Project (40%)

Parallel Programming Final Project

- Major programming project for the course
  - Non-trivial parallel application
  - Include performance analysis
  - Use NERSC supercomputers, or the GPU box
- Project teams
  - Up to 2 persons per team
  - Try to balance skills
- Project dates
  - Proposal due end of 4^{th} week
  - Project presentation during the final week
  - Project report due at the end of the quarter

What will you get out of AMS 250

- In-depth understanding of parallel computer design
- Knowledge of how to program parallel computer systems
- Understanding of pattern-based parallel programming
- Exposure to different forms parallel algorithms
- Practical experience using a parallel computer
- Background on parallel performance modeling
- Techniques for debugging, performance analysis and tuning

What is High Performance Computing

- We mostly use the following terms interchangeably:
    - *Parallel Computing*
    - *High Performance Computing*
    - *Supercomputing*
- *Parallel Computing* is all about *High Performance*
- A *parallel computer* is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem
- *Parallel processing* includes techniques and technologies that make it possible to compute in parallel
    - Hardware, networks, operating systems, parallel libraries, languages, compilers, algorithms, tools, …
- Parallel computing is an evolution of serial computing
    - Parallelism is natural
    - Computing problems differ in level / type of parallelism

Concurrency

- Consider multiple tasks to be executed in a computer
- Tasks are concurrent with respect to each other if
  - They *can* execute at the same time (*concurrent execution*)
  - Implies that there are no dependencies between the tasks
- Dependencies
  - If a task requires results produced by other tasks in order to execute correctly, the task's execution is *dependent*
  - If two tasks are dependent, they are not concurrent
  - Some form of synchronization must be used to enforce (satisfy) dependencies
- Concurrency is fundamental to computer science
  - Operating systems, databases, networking, …

Concurrency and Parallelism

- Concurrent is not the same as parallel! Why?
  [Rob Pike – 'Concurrency is Not Parallelism']
- Parallel execution
  - Concurrent tasks *actually* execute at the same time
  - Multiple (processing) resources have to be available
- **Parallelism = concurrency + parallel hardware**
  - Both are required
  - Find concurrent execution opportunities
  - Develop application to execute in parallel
  - Run application on parallel hardware
- Is a parallel application a concurrent application?
- Is a parallel application run with one processor parallel? Why or why not?

Parallelism

- There are granularities of parallelism (parallel execution) in programs
  - Processes, threads, routines, statements, instructions, …
  - Think about what are the software elements that execute concurrently
- These must be supported by hardware resources
  - Processors, cores, … (execution of instructions)
  - Memory, DMA, networks, … (other associated operations)
  - All aspects of computer architecture offer opportunities for parallel hardware execution
- Concurrency is a necessary condition for parallelism
  - Where can you find concurrency?
  - How is concurrency expressed to exploit parallel systems?

Why use parallel processing?

- Two primary reasons (both performance related)
  - Faster time to solution (response time)
  - Solve bigger computing problems (in same amount of time)
- Other factors motivate parallel processing
  - Effective use of machine resources
  - Cost efficiencies
  - Overcoming memory constraints
- Serial machines have inherent limitations
  - Processor speed, memory bottlenecks, …
- Parallelism has become the mainstream of computing
- Performance is still the driving concern
- **Parallelism = concurrency + parallel hardware = performance**

Perspectives on Parallel Processing

- Parallel computer architecture
    - Hardware needed for parallel execution
    - Computer system design
- (Parallel) Operating system
    - How to manage systems aspects in a parallel computer
- Parallel programming
    - Libraries (low-level, high-level)
    - Languages
    - Software development environments
- Parallel algorithms
- Parallel performance evaluation
- Parallel tools
    - Performance, debugging, analytics, visualization, …

Why study parallel computing today?

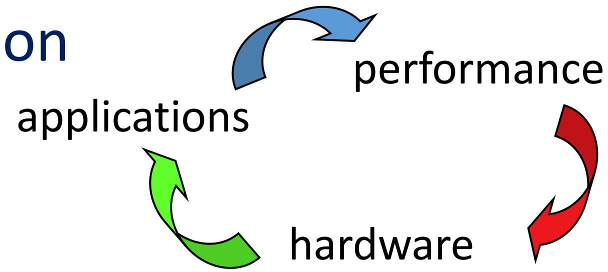
- Computing architecture
  - Innovations often drive to novel programming models
- Technological convergence
  - The “killer micro” is ubiquitous
  - Laptops and supercomputers are fundamentally similar!
  - Trends cause diverse approaches to converge
- Technological trends make parallel computing inevitable
  - Multi-core processors are here to stay!
  - Practically every computing system is operating in parallel
- Understand fundamental principles and design tradeoffs
  - Programming, systems support, communication, memory, …
  - Performance
- Parallelism is the mainstream and future of computing

Inevitability of Parallel Computing

- Application demands
  - Insatiable need for computing cycles
- Technology trends
  - Processor and memory
- Architecture trends
- Economics
- Current trends:
  - Today’s microprocessors have multiprocessor support
  - Servers and workstations available as multiprocessors
  - Tomorrow’s microprocessors are multiprocessors
  - Multi-core is here to stay and #cores/processor is growing
  - Accelerators (GPUs, gaming systems)

Application Characteristics

- Application performance demands hardware advances
- Hardware advances generate new applications
- New applications have greater performance demands
  - Exponential increase in microprocessor performance
  - Innovations in parallel architecture and integration



- Range of performance requirements
  - System performance must also improve as a whole
  - Performance requirements demand computer engineering
  - Costs addressed through technology advancements

Broad Parallel Architecture Issues

- Resource allocation
  - How many processing elements?
  - How powerful are the elements?
  - How much memory?
- Data access, communication, and synchronization
  - How do the elements cooperate and communicate?
  - How are data transmitted between processors?
  - What are the abstractions and primitives for cooperation?
- Performance and scalability
  - How does it all translate into performance?
  - How does it scale?

Moore’s Law

Gordon E Moore, Intel Cofounder

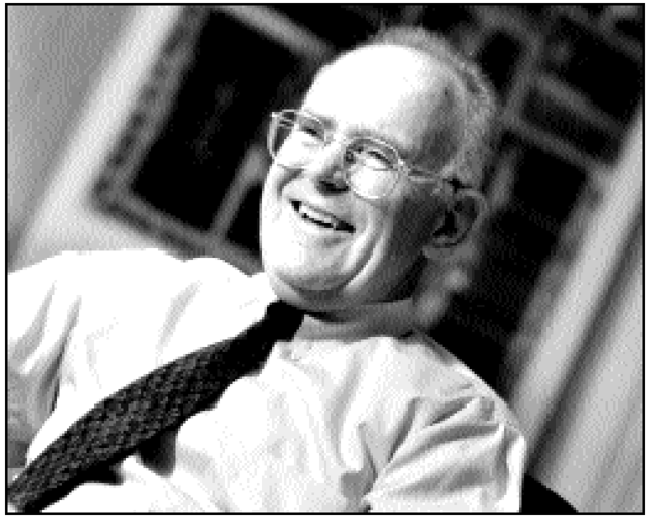
Electronics, 35th anniversary issue, 1965

> *“The complexity for minimum component costs has increased at a rate of* ***roughly a factor of two per year****. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years.”*

1975 revision

> *“The number of transistors than can be* ***cheaply*** *placed on integrated circuit board will* ***double every two years****.”*

≈ Chip performance **doubles every 18 months**



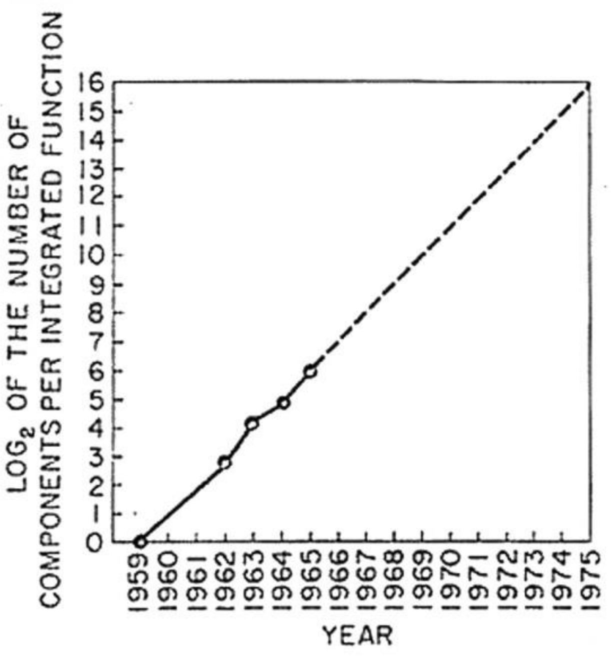
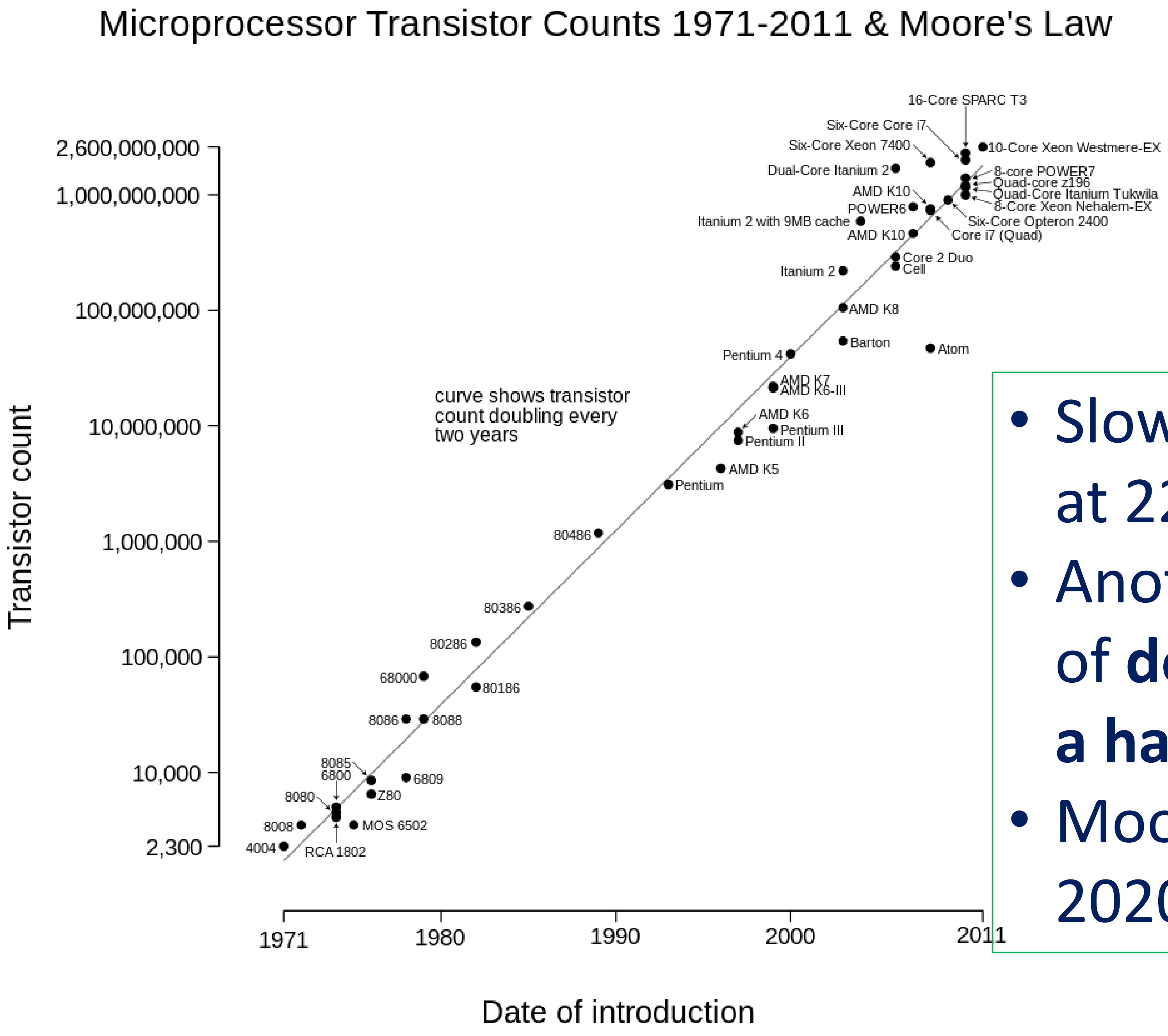


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

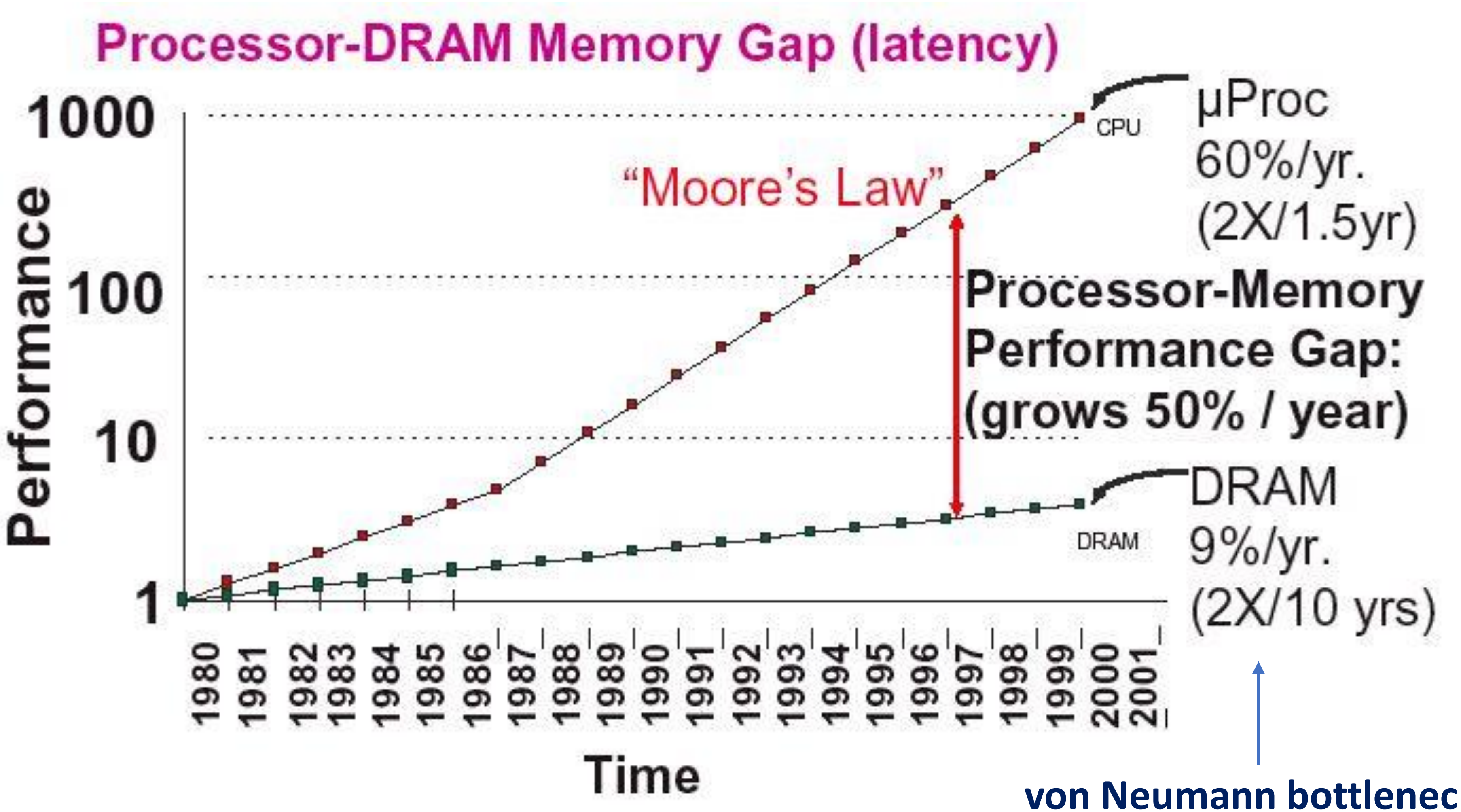


- Slowing down since 2012 at 22nm feature width
- Another revision to a rate of **doubling every two and a half years**?
- Moore's law may live on till 2020s

Leveraging Moore’s Law

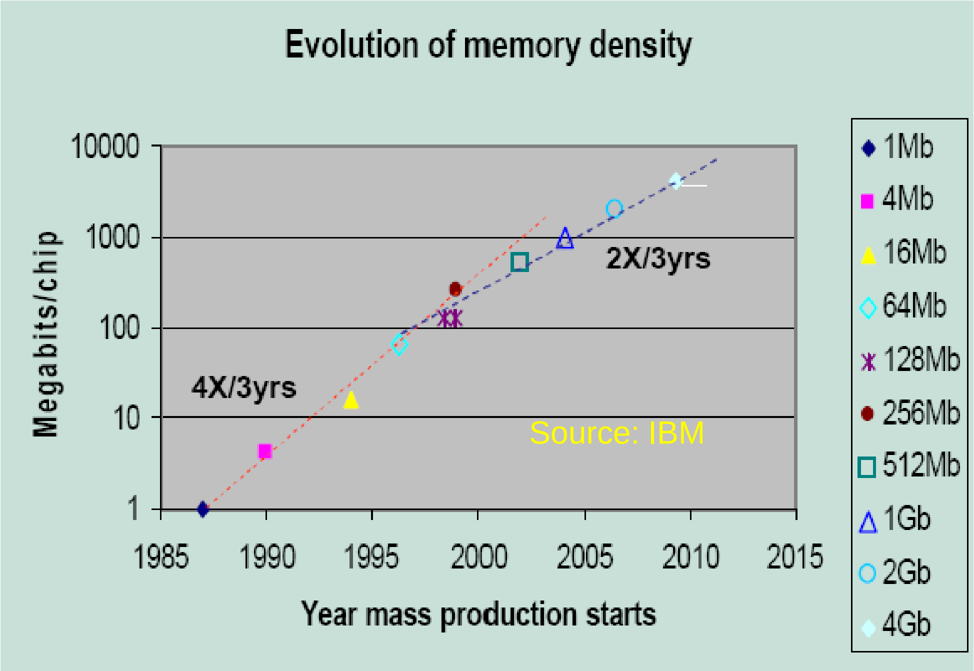
- More transistors = more parallelism opportunities
- Microprocessors
  - Implicit parallelism
    - pipelining
    - multiple functional units
    - superscalar
  - Explicit parallelism
    - SIMD instructions
    - long instruction works

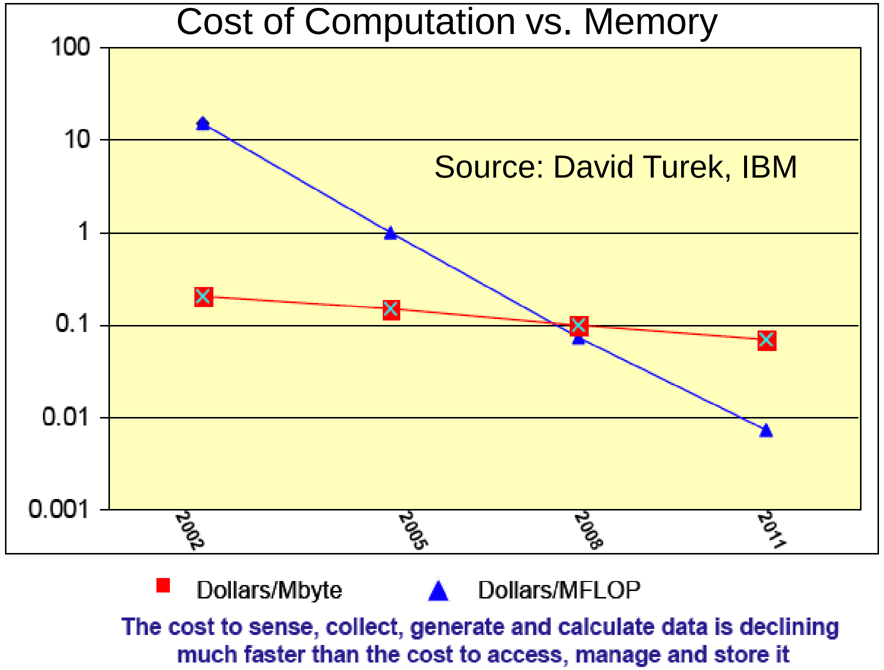
What's Driving Parallel Computing Architecture?



von Neumann bottleneck!
(memory wall)

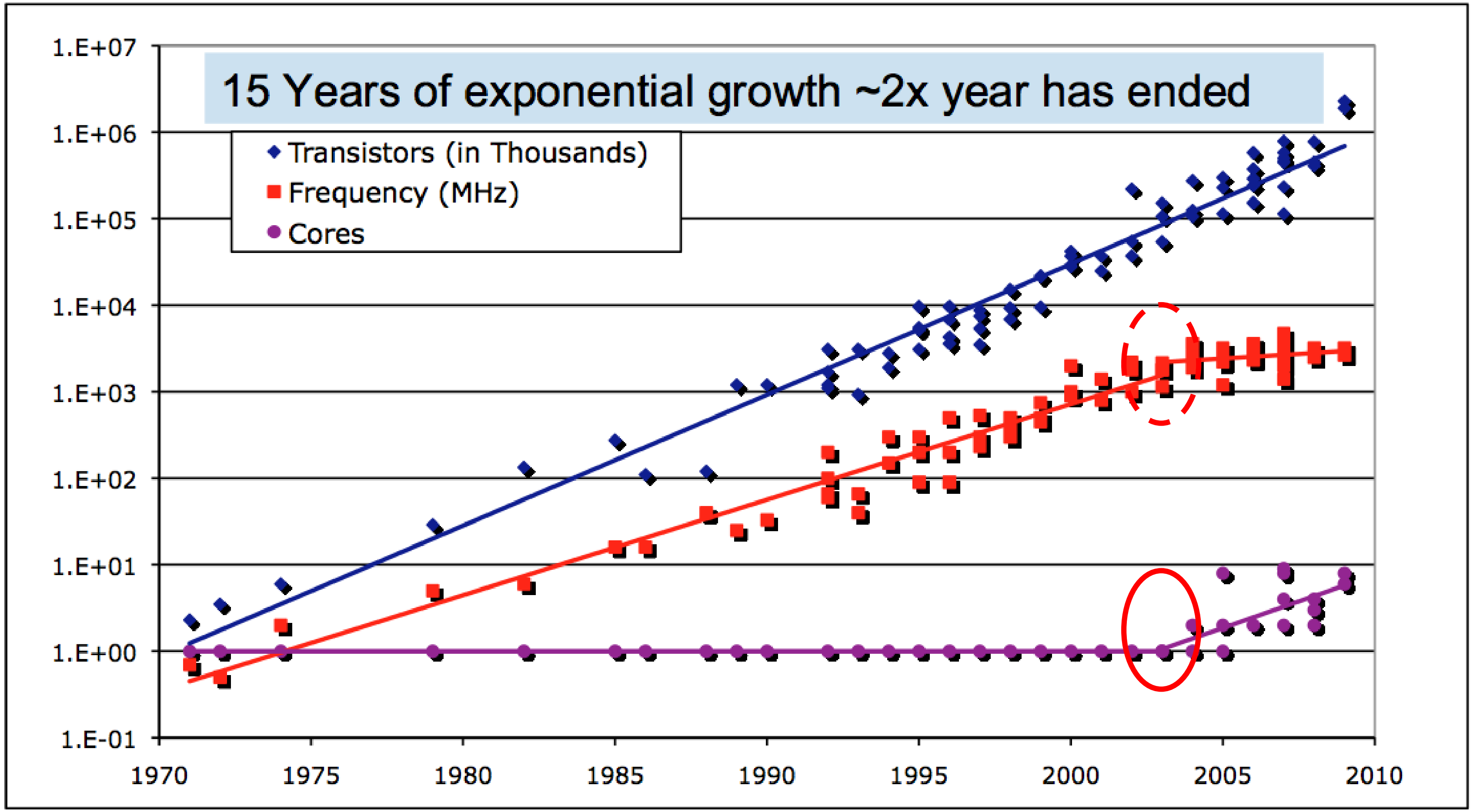
Memory Wall





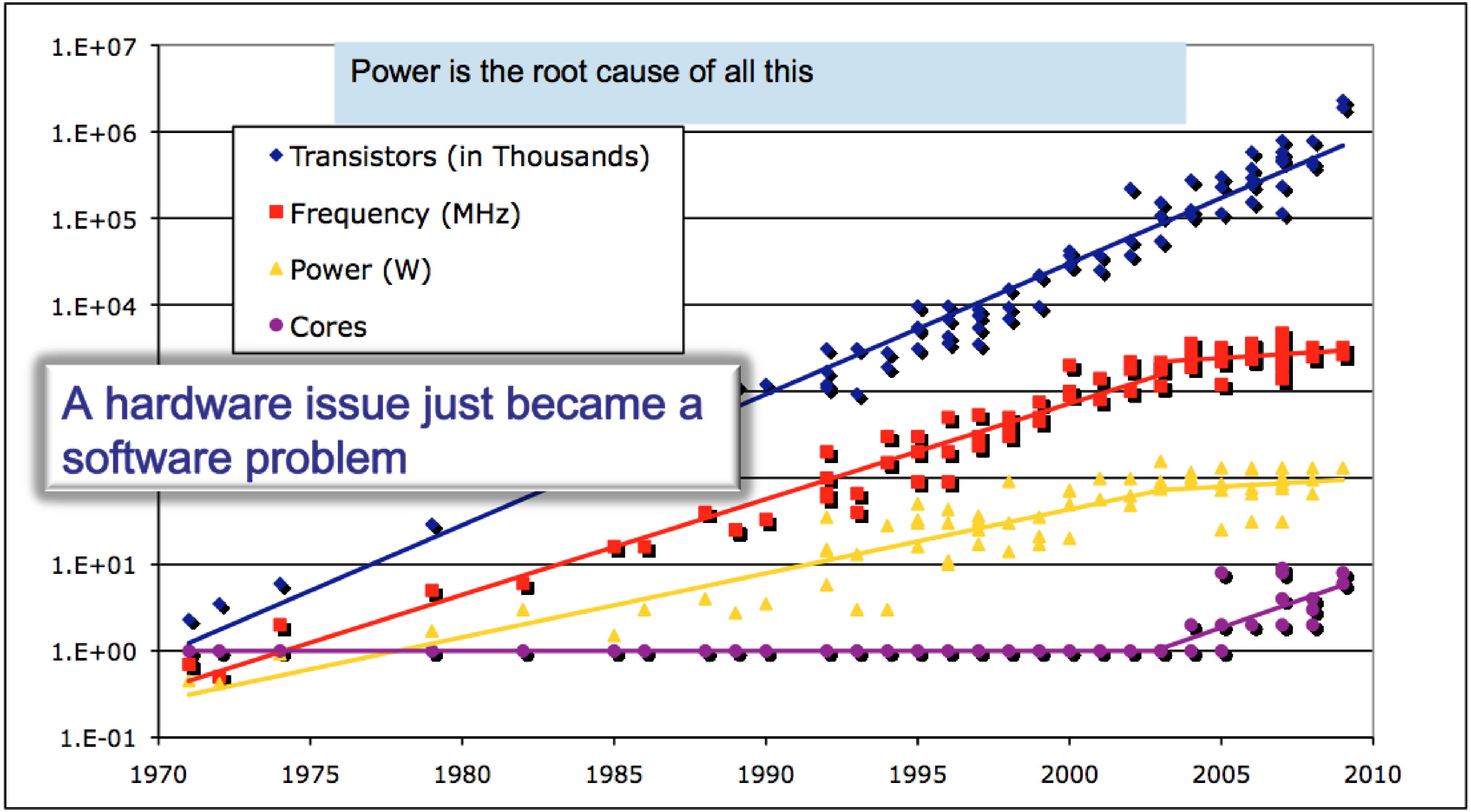
- Memory density is doubling every three years
- Processor logic (computation) is doubling every two years
- Memory are gradually getting more expensive, relative to computation
- Can we double concurrency without doubling memory?

What's Driving Parallel Computing Architecture?

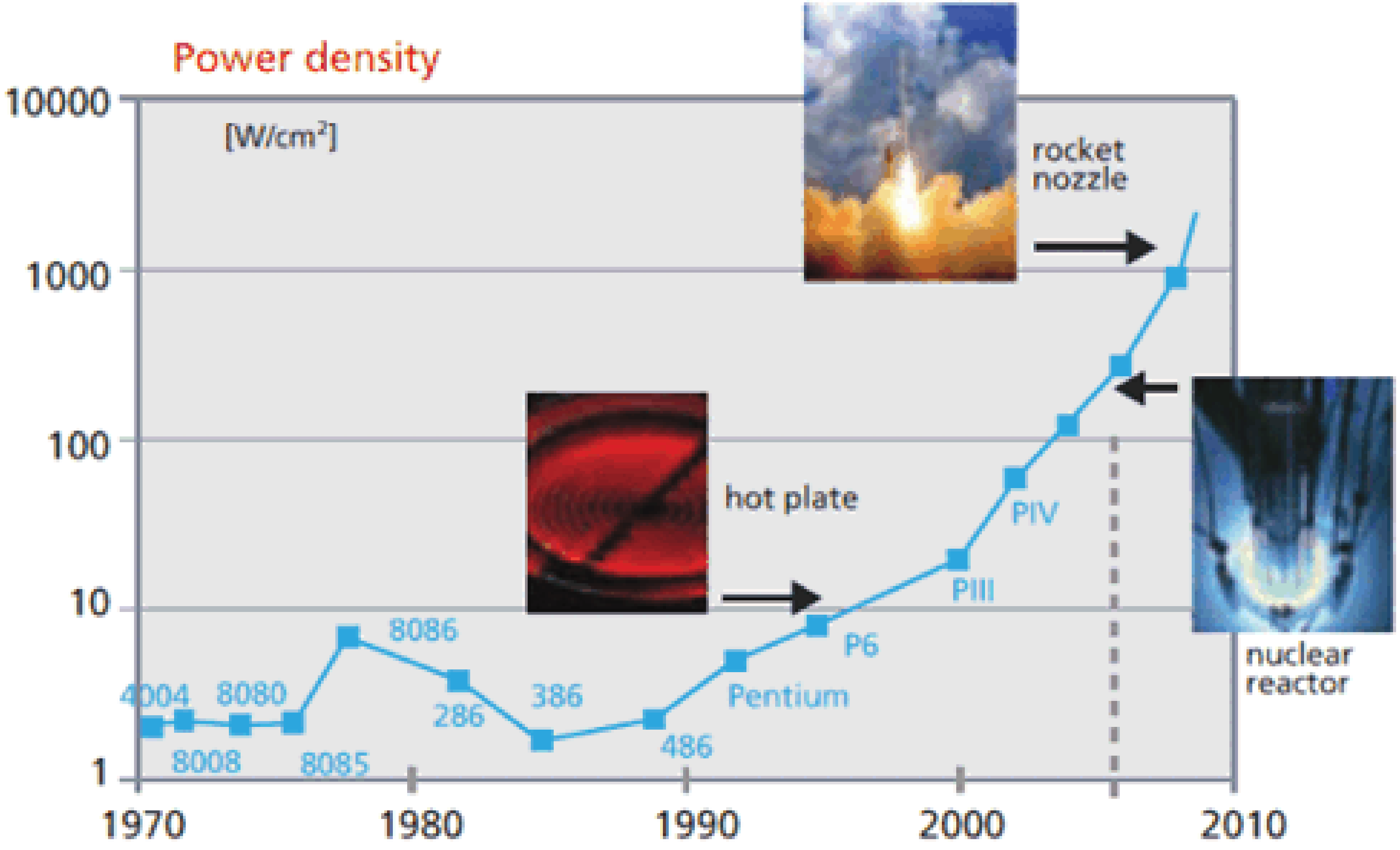


Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanoviç
Slide from Kathy Yelick

What's Driving Parallel Computing Architecture?



Power Density Growth



Power Wall

- Processing chip manufacturers had increased processor performance by increasing CPU clock frequency
- Until the chips got too hot!

$$P = CV^2 f$$

P is dynamic power consumed by a CPU, C is capacitance, V is voltage, f is frequency

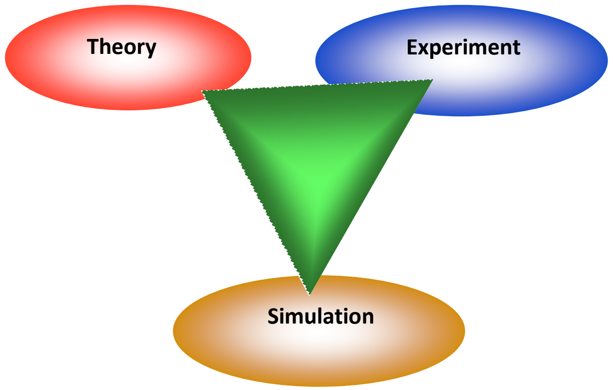
- Then they add more and more cores to increase performance
  - Keep clock frequency same or reduced
  - Keep lid on power requirements

What does the Technology Enable?

- Continued exponential increase in computational power
  - Simulation is becoming third pillar of science, complementing theory and experiment

- Continued exponential increase in experimental data
  - Techniques and technology in data analysis, visualization, analytics, networking, and collaboration tools are becoming essential in all data rich scientific applications

Third Pillar of Science

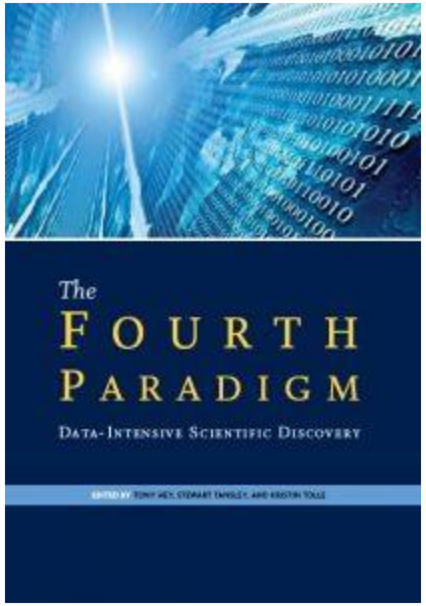


- Traditional scientific and engineering method:
  - (1) Do theory or paper design
  - (2) Perform experiments or build system
- Limitations:
  - Too difficult—build large wind tunnels
  - Too expensive—build a throw-away passenger jet
  - Too slow—wait for climate change or galactic evolution
  - Too dangerous—weapons, drug design, climate experimentation
- Computational Science and Engineering (CSE) paradigm:
  - (3) Use computers to simulate and analyze the phenomenon
  - Based on known physical laws and efficient numerical methods
  - Analyze simulation results with computational tools and methods beyond what is possible experimentally

The Fourth Paradigm

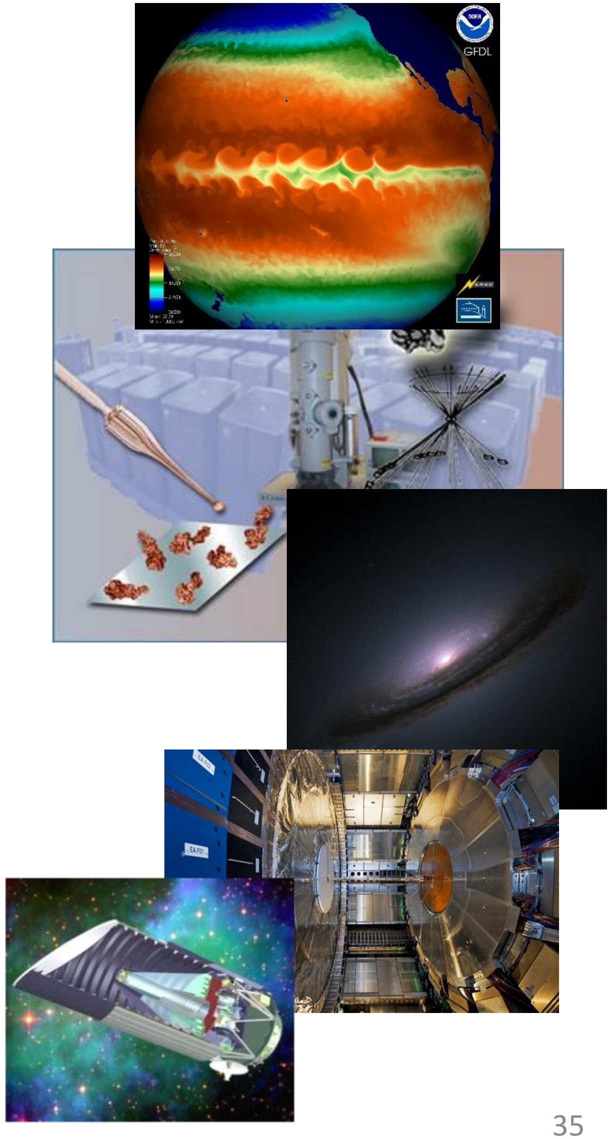
- Increasingly, scientific breakthroughs will be powered by advanced computing capabilities that help researchers manipulate and explore *massive datasets*
- Book:

  The Fourth Paradigm: Data-Intensive Scientific Discovery



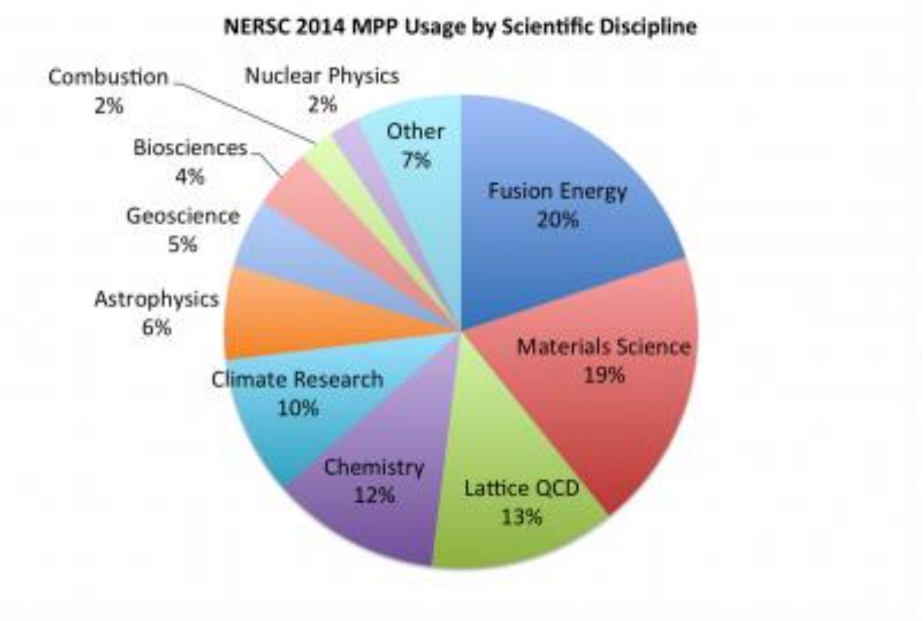
Data-Driven Science

- Scientific data sets are growing exponentially
  - Ability to generate data is exceeding our ability to store and analyze
  - Simulation systems and some observational devices grow in capability with Moore's Law
- Petabyte (PB) data sets are common:
  - ***Climate modeling:*** estimate of the next IPCC (Intergovernmental Panel on Climate Change) data is in 10s of petabytes
  - ***Genome:*** JGI (Joint Genome Institute) alone generated .5 petabyte of data in 2015 and doubles each year
  - ***Particle physics***: LHC (Large Hadron Collider) has generated more than 200 petabytes of data
  - ***Astrophysics***: LSST (Large Synoptic Survey Telescope) will produce 15 terabytes of raw scientific image data per night (via 3.2 Gigapixel camera)



Particularly Challenging Problems

- Science
  - Weather prediction, Global climate modeling
  - Biology: genomics, protein folding, drug design, etc.
  - Astrophysical modeling
  - Computational Chemistry
  - Computational Material Sciences and Nanoscience
- Engineering
  - Semiconductor design
  - Earthquake and structural modeling
  - Computation fluid dynamics (aircraft design)
  - Combustion (engine design)
  - Crash simulation
- Business
  - Financial and economic modeling
  - Transaction processing, web services and search engines
- Defense
  - Nuclear weapons
  - Cryptography



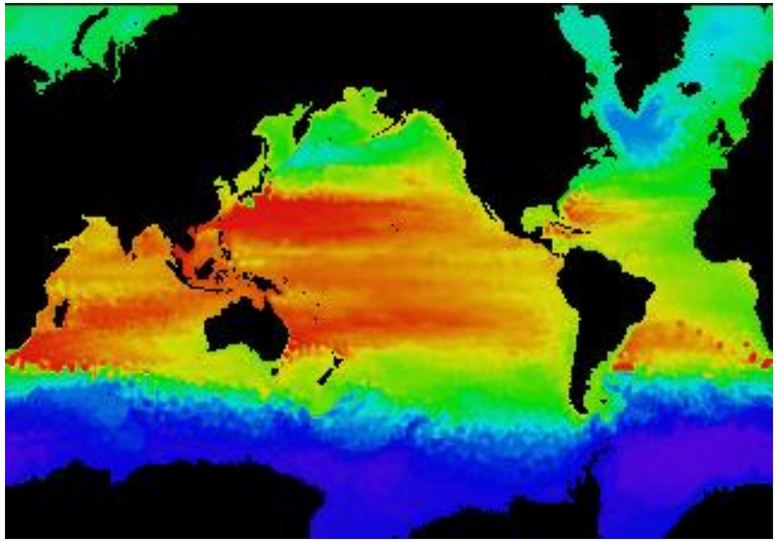
Example: Climate Modeling

- Problem is to compute:

  f(latitude, longitude, elevation, time) → "weather" =

  (temperature, pressure, humidity, wind velocity)

- Approach:
  - *Discretize* the domain - a measurement point every 10 km (0.1 deg)?
  - Devise an algorithm to predict weather at time *t+dt* given *t*
- Importance:
  - Predict major events, e.g., El Nino, hurricanes
  - Evaluate global warming scenarios



Ref: http://www.epm.ornl.gov/chammp/chammp.html

Example: Climate Modeling

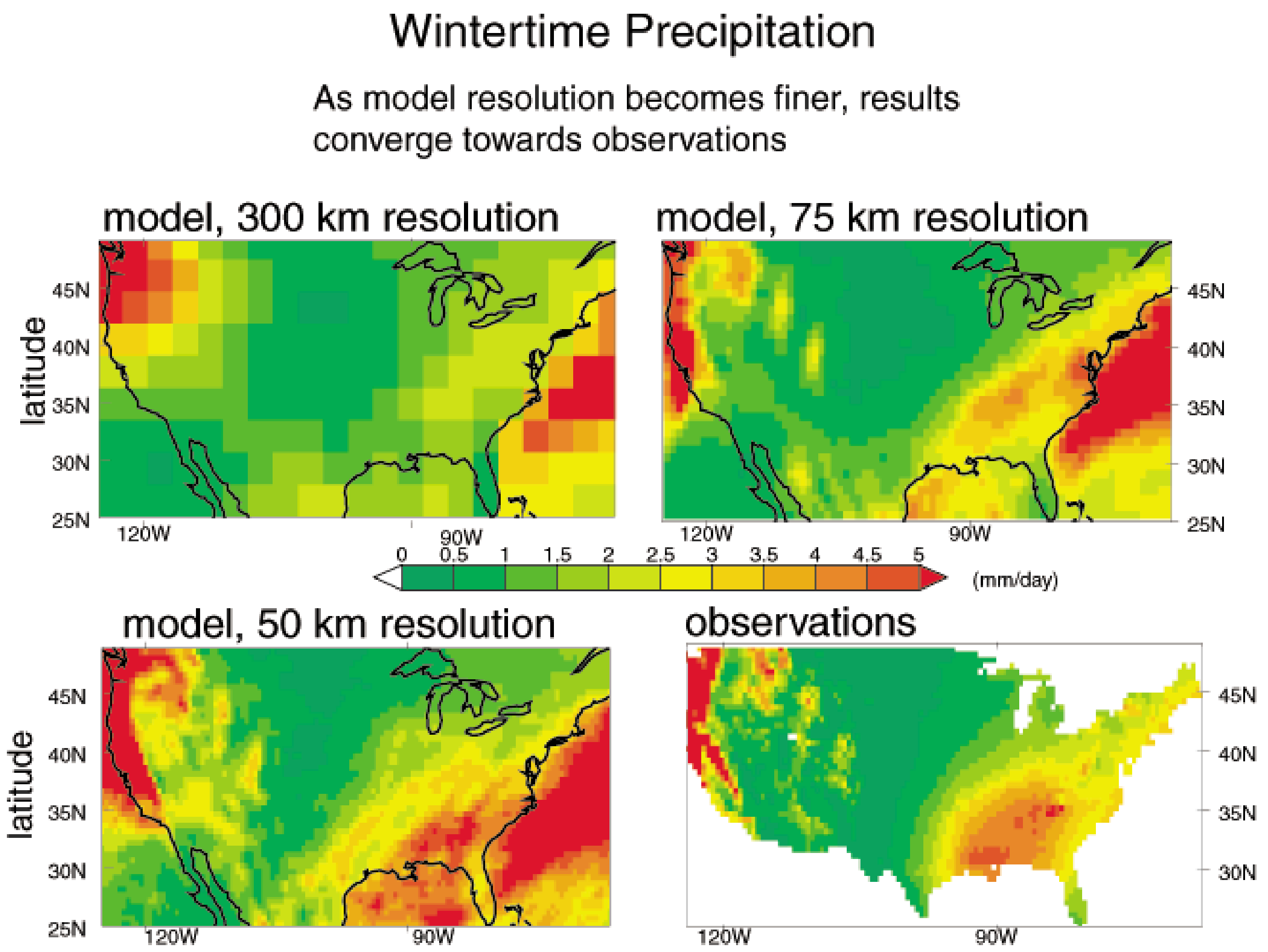
- <u>State of the art models</u> require integration of atmosphere, ocean, clouds, sea-ice, land models, plus possibly carbon cycle, geochemistry and more
- <u>One piece</u> is modeling the fluid flow in the atmosphere by solving the Navier-Stokes equations
  - Takes roughly 100 flops per grid point with 1-minute timestep
  - # points = Area/resolution * #height_levels = 4*pi*$(6000km/10km)^2$ * 1000 ~ 5 x 10^9

Example: Climate Modeling

- Computational requirements:
  - **Speed:** ~ $5 \times 10^9 \times 100$ flops → 5×10^{11} flops/timestep (min)
  - To match real-time, need 5×10^{11} flops in 60 seconds → 8 Gflop/s
  - Weather prediction (7 days in 24 hours) → 56 Gflop/s
  - Climate prediction (50 years in 30 days) → 4.8 Tflop/s
  - To use in policy negotiations (50 years in 12 hours) → 288 Tflop/s
  - **Data:**
    - **Per timestep (min):** 5×10^9 (points) x 8 bytes (double precision) x 5 (variables) → 200 GB
    - **Per sim hour**: 200 GB x 60 (mins) → 12 Terabytes
    - **Per climate prediction**: 12 TB x 50 (years) x 365 x 24 → 5 Exabytes
- To double the grid resolution, computation is 8x to 16x !!

Example: Climate Modeling

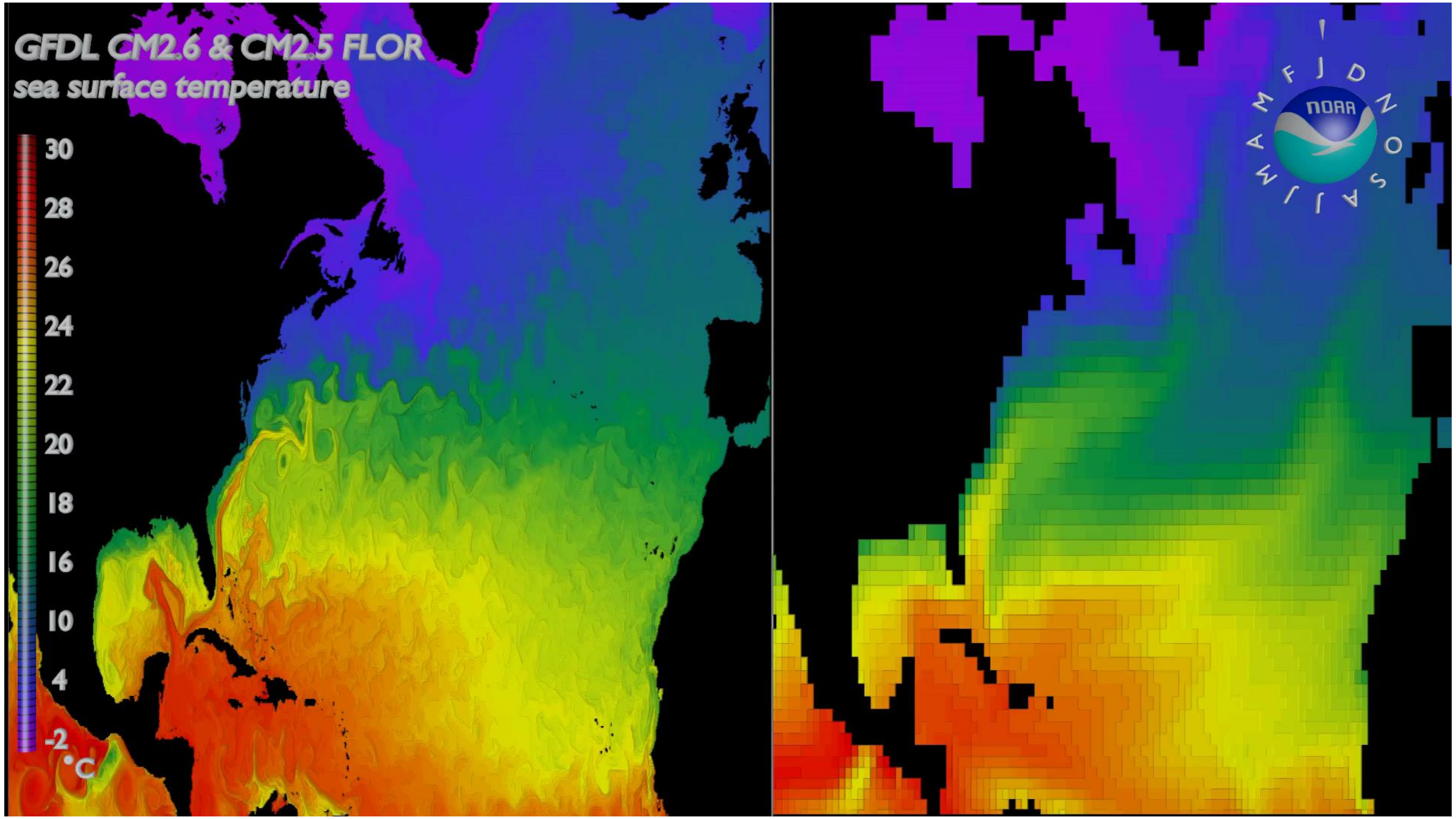
Effect of resolution:



Ref: P. Duffy et al, LLNL

Example: Climate Modeling

Effect of resolution:



Ref: NOAA GFDL

Classifying Parallel Systems – Flynn's Taxonomy

- Distinguishes multi-processor computer architectures along the two independent dimensions
  - ***Instruction*** and ***Data***
  - Each dimension can have one state: ***Single*** or ***Multiple***
- SISD: Single Instruction, Single Data
  - Serial (non-parallel) machine
- SIMD: Single Instruction, Multiple Data
  - Processor arrays and vector machines
  - SIMT (*T: threads*) for GPUs
- MISD: Multiple Instruction, Single Data (weird)
- MIMD: Multiple Instruction, Multiple Data
  - Most common parallel computer systems
  - SPMD & MPMD (*P: program*)

Parallel Architecture Types

- Instruction-Level Parallelism
  - Parallelism captured in instruction processing
- Vector processors
  - Operations on multiple data stored in vector registers
- Shared-memory Multiprocessor (SMP)
  - Multiple processors sharing memory
  - Symmetric Multiprocessor (SMP)
- Multicomputer
  - Multiple computers connect via network
  - Distributed-memory cluster
- Massively Parallel Processor (MPP)

Phases of Supercomputing (Parallel) Architecture

- Phase 1 (1950s): sequential instruction execution
- Phase 2 (1960s): sequential instruction issue
  - Pipeline execution, reservations stations
  - Instruction Level Parallelism (ILP)
- Phase 3 (1970s): vector processors
  - Pipelined arithmetic units
  - Registers, multi-bank (parallel) memory systems
- Phase 4 (1980s): SIMD and SMPs
- Phase 5 (1990s): MPPs and clusters
  - Communicating sequential processors
- Phase 6 (>2000): many cores, accelerators, scale, …

Performance Expectations

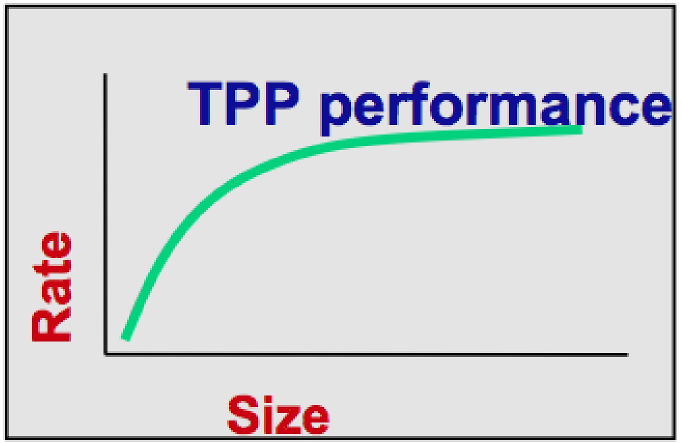
- If each processor is rated at k MFLOPS and there are p processors, we should expect to see k*p MFLOPS performance?
- If it takes 100 seconds on 1 processor, it should take 10 seconds on 10 processors?
- Several causes affect performance
  - Each must be understood separately
  - But they interact with each other in complex ways
    - solution to one problem may create another
    - one problem may mask another
- Scaling (system, problem size) can change conditions
- Need to understand performance space

Scalability

- A program can scale up to use many processors
  - What does that mean?
- How do you evaluate scalability?
- How do you evaluate scalability goodness?
- Comparative evaluation
  - If double the number of processors, what to expect?
  - Is scalability linear?
- Use parallel efficiency measure
  - Is efficiency retained as problem size increases?
- Apply performance metrics

Top 500 Benchmarking Methodology

- http://top500.org/
- Ranks and details of 500 fastest supercomputers in the world
- HPL (High Performance Linpack) benchmark
  - Solving dense linear system of equations (Ax = b)
- Data listed
  - R_{max} : maximal performance
  - R_{peak} : theoretical peak performance
  - N_{max} : problem size needed to achieve R_{max}
  - $N_{1/2}$: problem size needed to achieve 1/2 of R_{max}
  - Manufacturer and computer type
  - Installation site, location, and year
- Updated twice a year at ISC and SC conferences



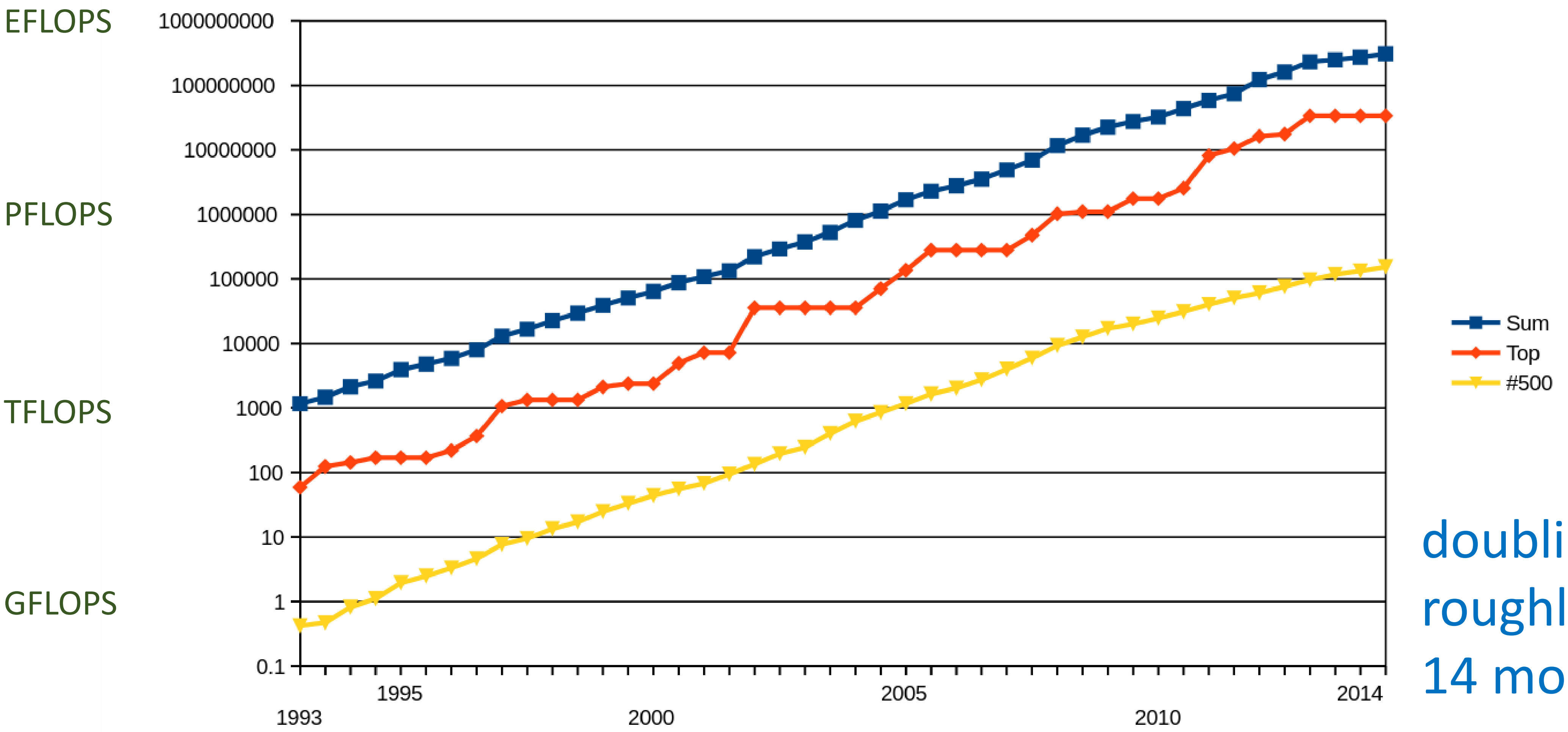
Top 10 positions of the 50th TOP500 in November 2017[15]

Rank	Rmax Rpeak (PFLOPS)	Name	Model	Processor	Interconnect	Vendor	Site country, year	Operating system
1	93.015 125.436	*Sunway TaihuLight*	Sunway MPP	SW26010	Sunway[16]	NRCPC	National Supercomputing Center in Wuxi China, 2016[16]	Linux (Raise)
2	33.863 54.902	*Tianhe-2*	TH-IVB-FEP	Xeon E5–2692, Xeon Phi 31S1P	TH Express-2	NUDT	National Supercomputing Center in Guangzhou China, 2013	Linux (Kylin)
3	19.590 25.326	*Piz Daint*	Cray XC50	Xeon E5-2690v3, Tesla P100	Aries	Cray	Swiss National Supercomputing Centre Switzerland, 2016	Linux (CLE)
4	19.136 28.192	*Gyoukou*	ZettaScaler-2.2 HPC system	Xeon D-1571, PEZY-SC2	Infiniband EDR	ExaScaler	Japan Agency for Marine-Earth Science and Technology Japan, 2017	Linux (CentOS)
5	17.590 27.113	*Titan*	Cray XK7	Opteron 6274, Tesla K20X	Gemini	Cray	Oak Ridge National Laboratory United States, 2012	Linux (CLE, SLES based)
6	17.173 20.133	*Sequoia*	Blue Gene/Q	A2	Custom	IBM	Lawrence Livermore National Laboratory United States, 2013	Linux (RHEL and CNK)
7	14.137 43.902	*Trinity*	Cray XC40	Xeon E5–2698v3, Xeon Phi	Aries	Cray	Los Alamos National Laboratory United States, 2015	Linux (CLE)
8	14.015 27.881	*Cori*	Cray XC40	Xeon Phi 7250	Aries	Cray	National Energy Research Scientific Computing Center United States, 2016	Linux (CLE)
9	13.555 24.914	*Oakforest-PACS*	Fujitsu	Xeon Phi 7250	Intel Omni-Path	Fujitsu	Kashiwa, Joint Center for Advanced High Performance Computing Japan, 2016	Linux
10	10.510 11.280	*K computer*	Fujitsu	SPARC64 VIIIfx	Tofu	Fujitsu	Riken, Advanced Institute for Computational Science (AICS) Japan, 2011	Linux

Tops of the Top 500

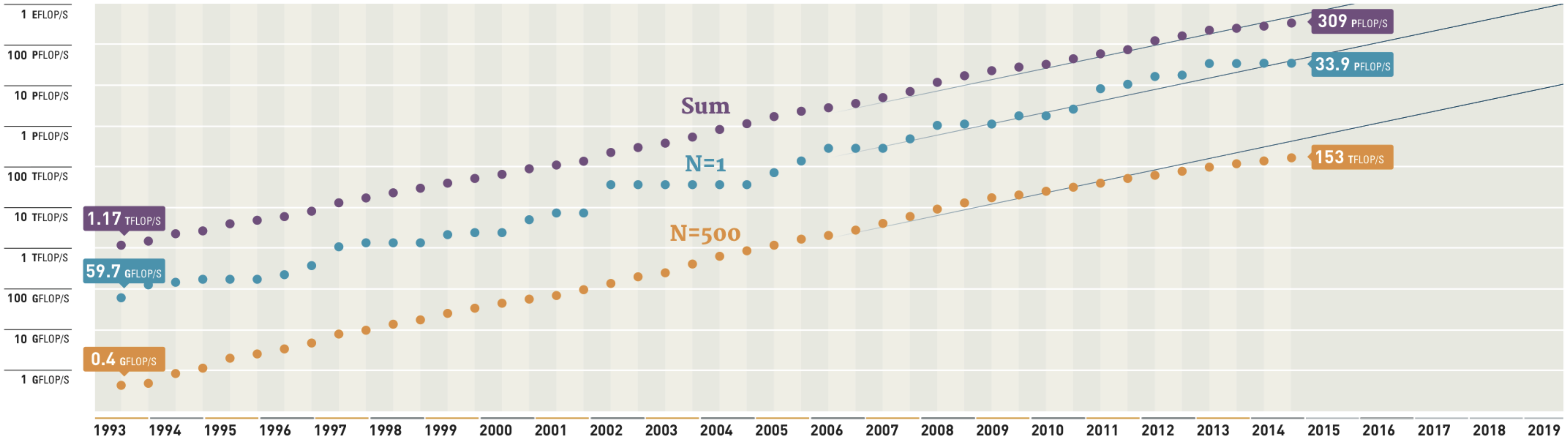
Year	Supercomputer	Peak speed (Rmax)	Location
1993	Fujitsu Numerical Wind Tunnel	124.50 GFLOPS	National Aerospace Laboratory, Tokyo, Japan
1993	Intel Paragon XP/S 140	143.40 GFLOPS	DoE-Sandia National Laboratories, New Mexico, USA
1994	Fujitsu Numerical Wind Tunnel	170.40 GFLOPS	National Aerospace Laboratory, Tokyo, Japan
1996	Hitachi SR2201/1024	220.4 GFLOPS	University of Tokyo, Japan
	Hitachi CP-PACS/2048	368.2 GFLOPS	University of Tsukuba, Tsukuba, Japan
1997	Intel ASCI Red/9152	1.338 TFLOPS	DoE-Sandia National Laboratories, New Mexico, USA
1999	Intel ASCI Red/9632	2.3796 TFLOPS	
2000	IBM ASCI White	7.226 TFLOPS	DoE-Lawrence Livermore National Laboratory, California, USA
2002	NEC Earth Simulator	35.86 TFLOPS	Earth Simulator Center, Yokohama, Japan
2004	IBM Blue Gene/L	70.72 TFLOPS	DoE/IBM Rochester, Minnesota, USA
2005		136.8 TFLOPS	DoE/U.S. National Nuclear Security Administration, Lawrence Livermore National Laboratory, California, USA
		280.6 TFLOPS	
2007		478.2 TFLOPS	
2008	IBM Roadrunner	1.026 PFLOPS	DoE-Los Alamos National Laboratory, New Mexico, USA
		1.105 PFLOPS	
2009	Cray Jaguar	1.759 PFLOPS	DoE-Oak Ridge National Laboratory, Tennessee, USA
2010	Tianhe-IA	2.566 PFLOPS	National Supercomputing Center, Tianjin, China
2011	Fujitsu K computer	10.51 PFLOPS	RIKEN, Kobe, Japan
2012	IBM Sequoia	16.32 PFLOPS	Lawrence Livermore National Laboratory, California, USA
2012	Cray Titan	17.59 PFLOPS	Oak Ridge National Laboratory, Tennessee, USA
2013	NUDT Tianhe-2	33.86 PFLOPS	Guangzhou, China

Top 500 Performance Development

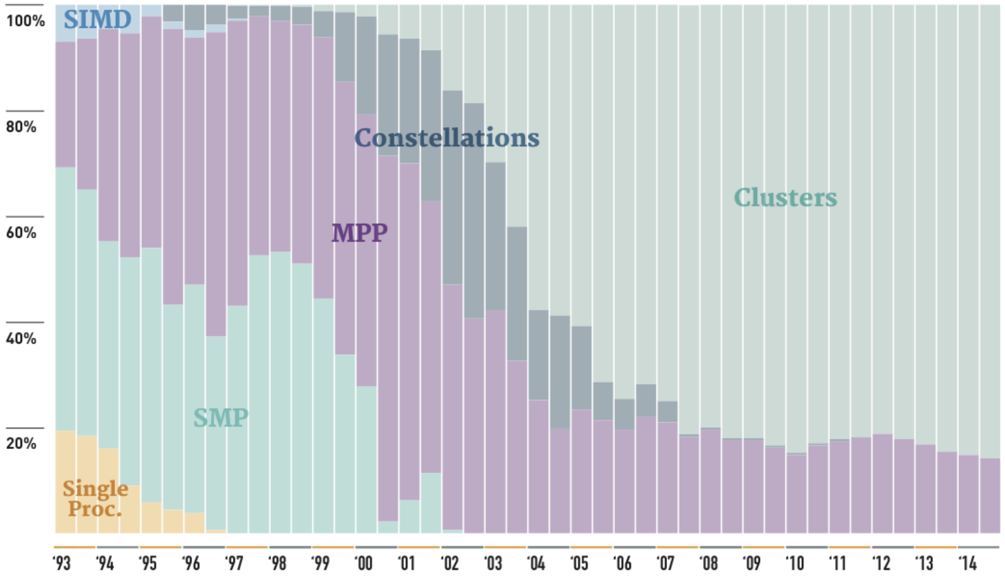


doubling roughly every 14 months

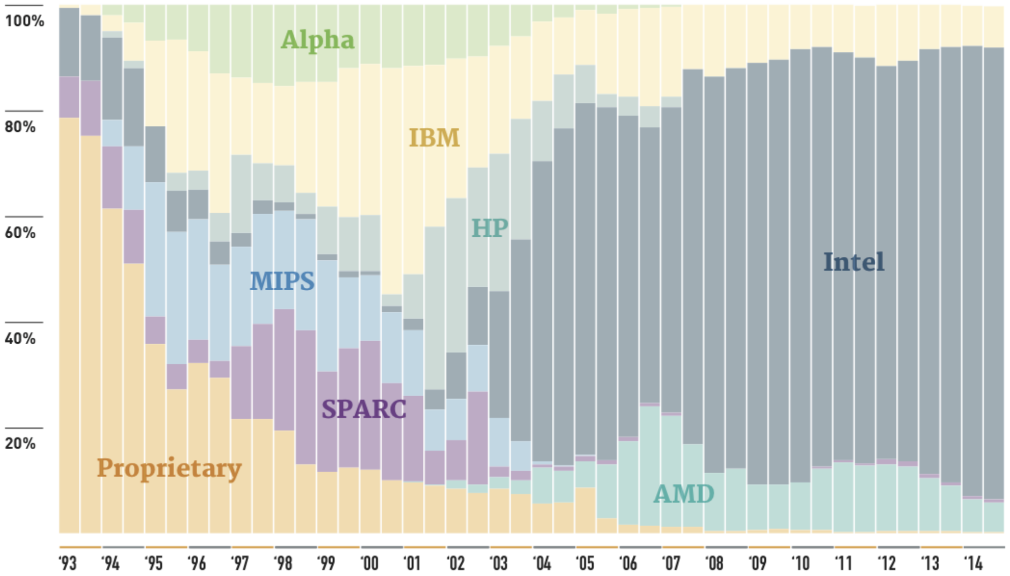
PERFORMANCE DEVELOPMENT



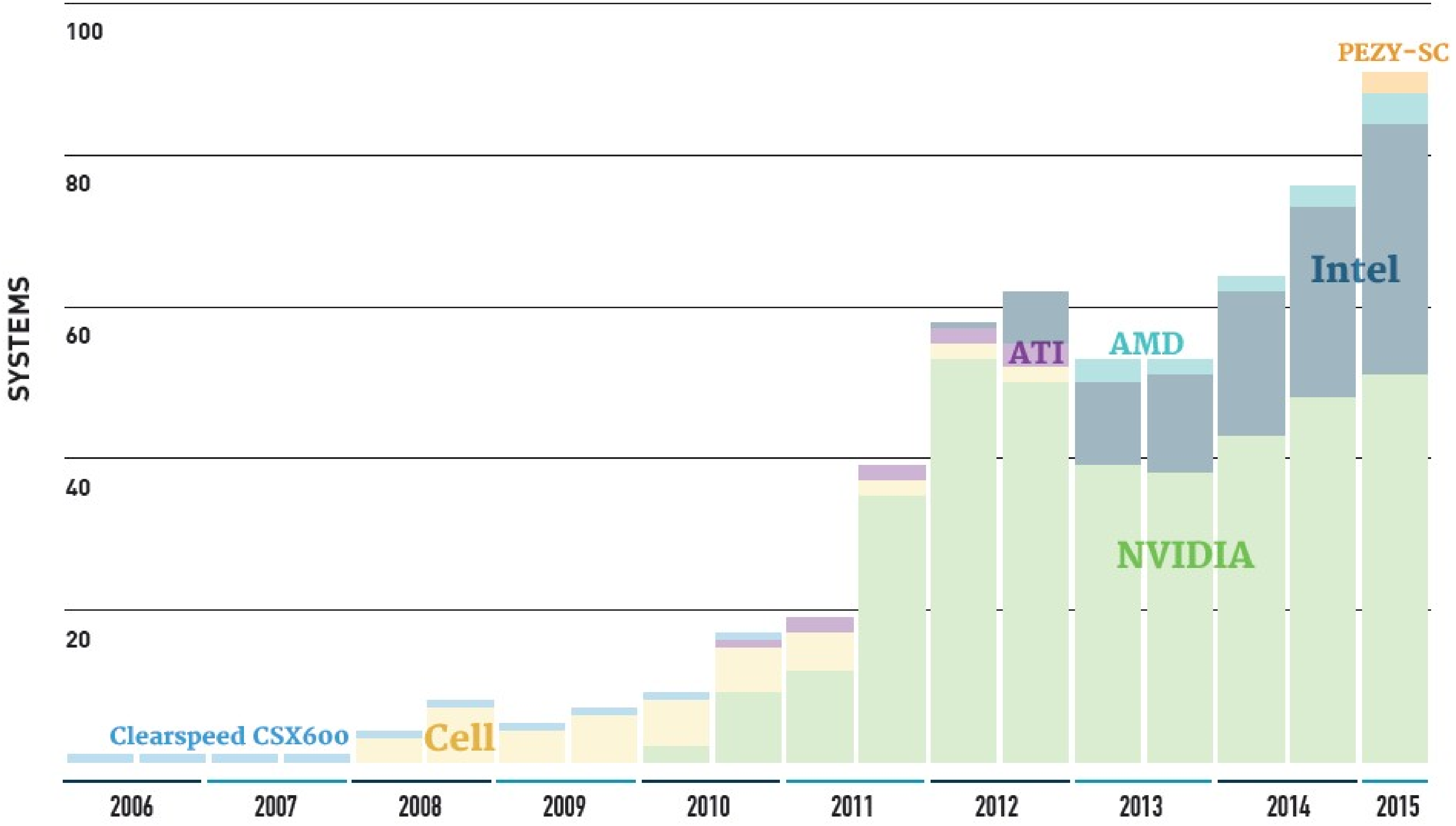
ARCHITECTURES



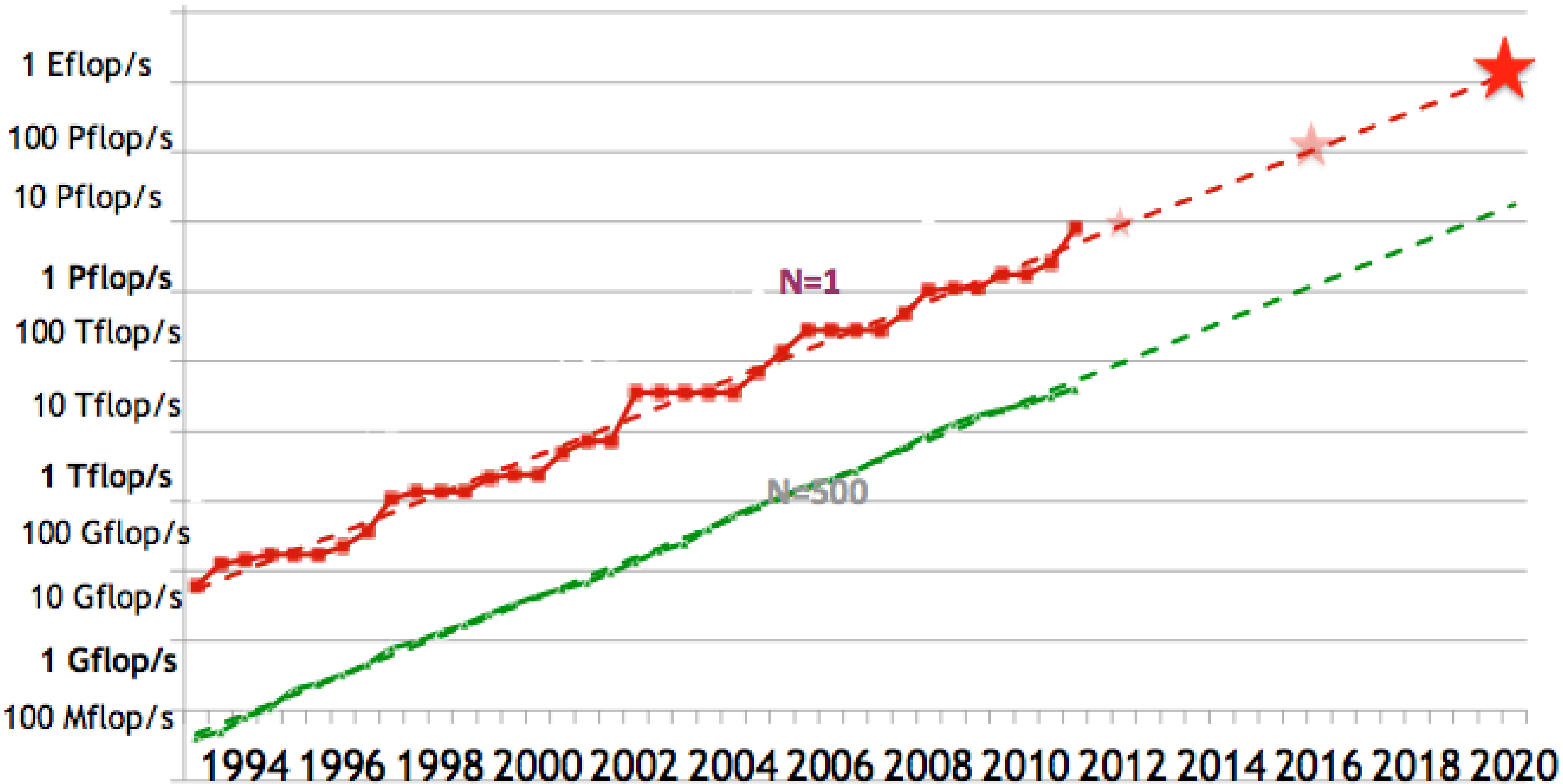
CHIP TECHNOLOGY



ACCELERATORS/CO-PROCESSORS



Top 500 Performance Development



http://www.netlib.org/utk/people/JackDongarra/SLIDES/korea-2011.pdf

Exascale Initiative

- Exascale machines are targeted for 2020
- What are the potential differences and problems?

Systems	2011 K Computer	2019	Difference Today & 2019
System peak	8.7 Pflop/s	1 Eflop/s	O(100)
Power	10 MW	~20 MW	
System memory	1.6 PB	32 - 64 PB	O(10)
Node performance	128 GF	1,2 or 15TF	O(10) - O(100)
Node memory BW	64 GB/s	2 - 4TB/s	O(100)
Node concurrency	8	O(1k) or 10k	O(100) - O(1000)
Total Node Interconnect BW	20 GB/s	200-400GB/s	O(10)
System size (nodes)	68,544	O(100,000) or O(1M)	O(10) - O(100)
Total concurrency	548,352	O(billion)	O(1,000)
MTTI	days	O(1 day)	- O(10)

Table 1. Computational science platform requirements for the OLCF

	2012	2017	2020	2024
Peak flops	10–20 PF	100–200 PF	500–2000 PF	2000–4000 PF
Memory	0.5–1 PB	5–10 PB	32–64 PB	50–100 PB
Burst storage bandwidth	NA	5 TB/s	32 TB/s	50 TB/s
Burst capacity (cache)	NA	500 TB	3 PB	5 PB
Mid-tier capacity (disk)	20 PB	100 PB	1 EB	5 EB
Bottom-tier capacity (tape)	100 PB	1 EB	10 EB	50 EB
I/O servers	400	500	600	700

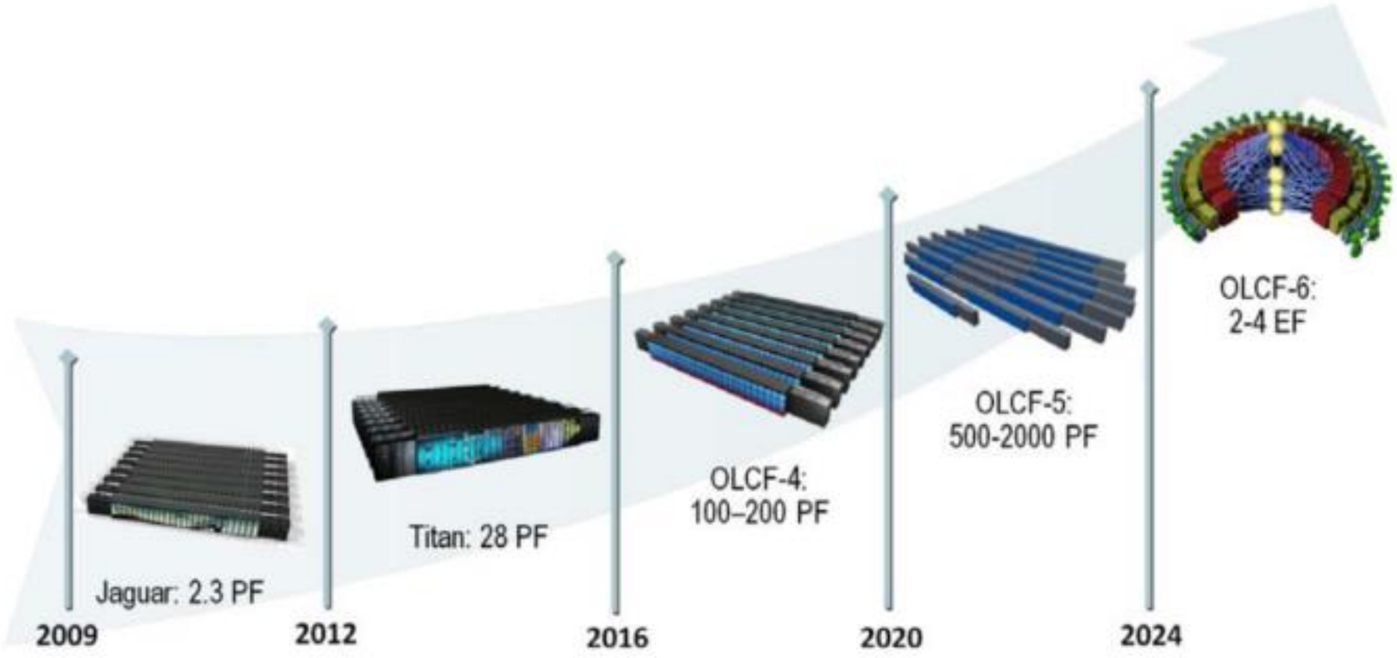


Figure 1. OLCF 2024 roadmap.

https://www.olcf.ornl.gov/wp-content/uploads/2013/01/OLCF_Requirements_TM_2013_Final.pdf

Major Changes to Software and Algorithms

- Must rethink the design for exascale
  - Data movement is expensive
    - Moving 1-bit data: ~1 pj/mm (today & 2020)
    - Moving 192-bit data: ~200 pj/mm (today & 2020)
  - Flops per second are cheap
    - ~50 pj/FLOP today
    - 5 – 10 pj/FLOP achievable 2020
    - Koomey's law: the number of computations per joule of energy dissipated has been doubling approximately every 1.57 years
- Need to reduce communication and synchronization
- Need to develop fault-resilient algorithms
- How do with deal with massive parallelism?
- Software must adapt to the hardware (autotuning)

HPCG Benchmark

- http://www.hpcg-benchmark.org/
- Complemental to **HPL**, which tests solution of *dense* linear systems
- **HPCG** (High Performance Conjugate Gradient) is designed to model the computational and data access patterns of real-world applications
- Generates and solves a synthetic 3D *sparse* linear system using a local symmetric Gauss-Seidel preconditioned conjugate gradient method
- Reference implementation is written in C++ with MPI and OpenMP support

  https://github.com/hpcg-benchmark/hpcg/
- Paper:

  http://www.sandia.gov/~maherou/docs/HPCG-Benchmark.pdf

November 2017 HPCG Results

Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak
1	RIKEN Advanced Institute for Computational Science Japan	**K computer** – , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10.510	10	0.603	5.3%
2	NSCC / Guangzhou China	**Tianhe-2 (MilkyWay-2)** – TH-IVB-FEP Cluster, Intel Xeon 12C 2.2GHz, TH Express 2, Intel Xeon Phi 31S1P 57-core NUDT	3,120,000	33.863	2	0.580	1.1%
3	DOE/NNSA/LANL/SNL USA	**Trinity** – Cray XC40, Intel Xeon E5-2698 v3 300160C 2.3GHz, Aries Cray	979,072	14.137	7	0.546	1.8%
4	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** – Cray XC50, Intel Xeon E5-2690v3 12C 2.6GHz, Cray Aries, NVIDIA Tesla P100 16GB Cray	361,760	19.590	3	0.486	1.9%
5	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** – Sunway MPP, SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93.015	1	0.481	0.4%
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** – PRIMERGY CX600 M1, Intel Xeon Phi Processor 7250 68C 1.4GHz, Intel Omni-Path Architecture Fujitsu	557,056	13.555	9	0.385	1.5%
7	DOE/SC/LBNL/NERSC USA	**Cori** – XC40, Intel Xeon Phi 7250 68C 1.4GHz, Cray Aries Cray	632,400	13.832	8	0.355	1.3%
8	DOE/NNSA/LLNL USA	**Sequoia** – IBM BlueGene/Q, PowerPC A2 1.6 GHz 16-core, 5D Torus IBM	1,572,864	17.173	6	0.330	1.6%
9	DOE/SC/Oak Ridge Nat Lab USA	**Titan** – Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray	560,640	17.590	5	0.322	1.2%
10	GSIC Center, Tokyo Institute of Technology Japan	**TSUBAME3.0** – SGI ICE XA (HPE SGI 8600), IP139-SXM2, Intel Xeon E5-2680 v4 15120C 2.9GHz, Intel Omni-Path Architecture, NVIDIA TESLA P100 SXM2 with NVLink HPE	136,080	8.125	13	0.189	1.6%

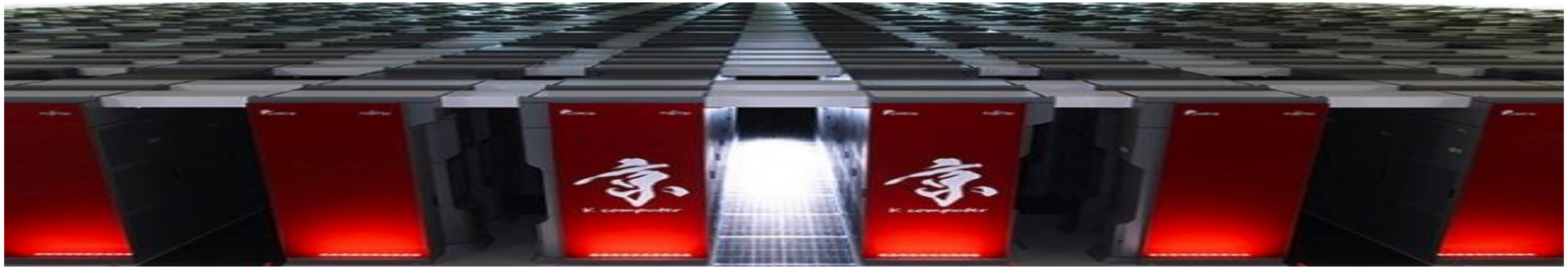
Graph 500

- http://www.graph500.org/
- Rating of supercomputers, focused on ***data intensive loads***
- Graph 500 benchmark
  - breadth-first search in a large undirected graph (model of Kronecker graph with average degree of 16)
- 6 problem classes defined by their input size:
  - **toy** : 17 GB (2^{26} vertices, scale 26; 10^{10} bytes, level 10)
  - **mini** : 140 GB (2^{29} vertices, scale 29; 10^{11} bytes, level 11)
  - **small** : 1 TB (2^{32} vertices, scale 32; 10^{13} bytes, level 13)
  - **medium** : 17 TB (2^{36} vertices, scale 36; 10^{14} bytes, level 14)
  - **large** : 140 TB (2^{39} vertices, scale 39; 10^{15} bytes, level 15)
  - **huge** : 1.1 PB (2^{42} vertices, scale 42; 10^{11} bytes, level 16)
- The main performance metric is ***GTEPS*** (10^9 traversed edges per second)

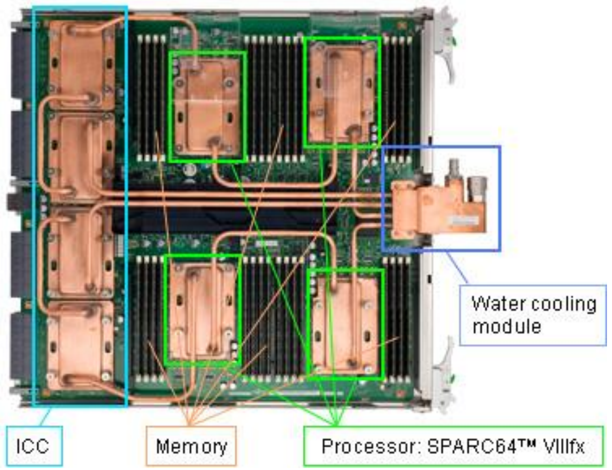
Graph 500 Top 10 (June 2016)

Rank	Site	Machine (Architecture)	Number of nodes	Number of cores	Problem scale	GTEPS
1	RIKEN Advanced Institute for Computational Science	K computer (Fujitsu custom)	82944	663552	40	38621.4
2	National Supercomputing Center in Wuxi	Sunway TaihuLight (NRCPC - Sunway MPP)	40768	10599680	40	23755.7
3	Lawrence Livermore National Laboratory	IBM Sequoia (Blue Gene/Q)	98304	1572864	41	23751
4	Argonne National Laboratory	IBM Mira (Blue Gene/Q)	49152	786432	40	14982
5	Forschungszentrum Jülich	JUQUEEN (Blue Gene/Q)	16384	262144	38	5848
6	CINECA	Fermi (Blue Gene/Q)	8192	131072	37	2567
7	Changsha, China	Tianhe-2 (NUDT custom)	8192	196608	36	2061.48
8	CNRS/IDRIS-GENCI	Turing (Blue Gene/Q)	4096	65536	36	1427
8	Science and Technology Facilities Council - Daresbury Laboratory	Blue Joule (Blue Gene/Q)	4096	65536	36	1427
8	University of Edinburgh	DIRAC (Blue Gene/Q)	4096	65536	36	1427
8	EDF R&D	Zumbrota (Blue Gene/Q)	4096	65536	36	1427
8	Victorian Life Sciences Computation Initiative	Avoca (Blue Gene/Q)	4096	65536	36	1427

RIKEN K Computer



- 82,944 (96/cabinets x 864 cabinets) Compute Nodes, each with:
  - One 8-core SPARC64 VIIIfx @ 2.0 GHz
  - 16 GB of memory
- 5,184 (6/cabinets x 864 cabinets) I/O Nodes
- 6-dimensional torus interconnect (*Tofu*)
- Fujitsu Exabyte File System (*FEFS*), based on Lustre
- #1 in June 2011



R_{peak} = 11.280 PFLOPS
R_{max} = 10.510 PFLOPS
Power = 12.6 MW
Cost > 100 billion Yen ($1.25b)

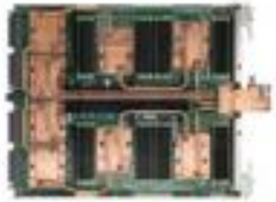
K computer Specifications

CPU (SPARC64 VIIIfx)	Cores/Node	8 cores (@2GHz)
	Performance	128GFlops
	Architecture	SPARC V9 + HPC extension
	Cache	L1(I/D) Cache : 32KB/32KB L2 Cache : 6MB
	Power	58W (typ. 30 C)
	Mem. bandwidth	64GB/s.
Node	Configuration	1 CPU / Node
	Memory capacity	16GB (2GB/core)
System board(SB)	No. of nodes	4 nodes /SB
Rack	No. of SB	24 SBs/rack
System	Nodes/system	> 80,000

Inter-connect	Topology	6D Mesh/Torus
	Performance	5GB/s. for each link
	No. of link	10 links/ node
	Additional feature	H/W barrier, reduction
	Architecture	Routing chip structure (no outside switch box)
Cooling	CPU, ICC*	Direct water cooling
	Other parts	Air cooling

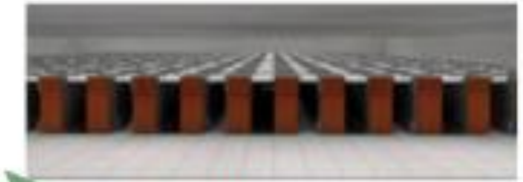
CPU
128GFlops
SPARC64™ VIIIfx
8 Cores@2.0GHz

Node
128 GFlops
16GB Memory
64GB/s Memory band width



System Board
512 GFlops
64 GB memory

Rack
12.3 TFlops
15TB memory



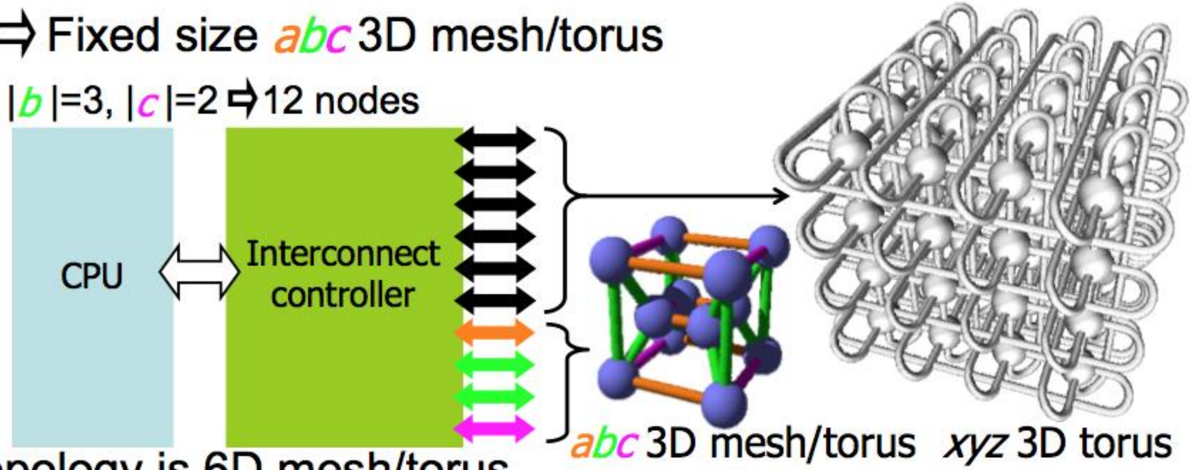
System
LINPACK 10 PFlops
over 1PB mem.
800 racks
80,000 CPUs
640,000 cores

* ICC : Interconnect Chip

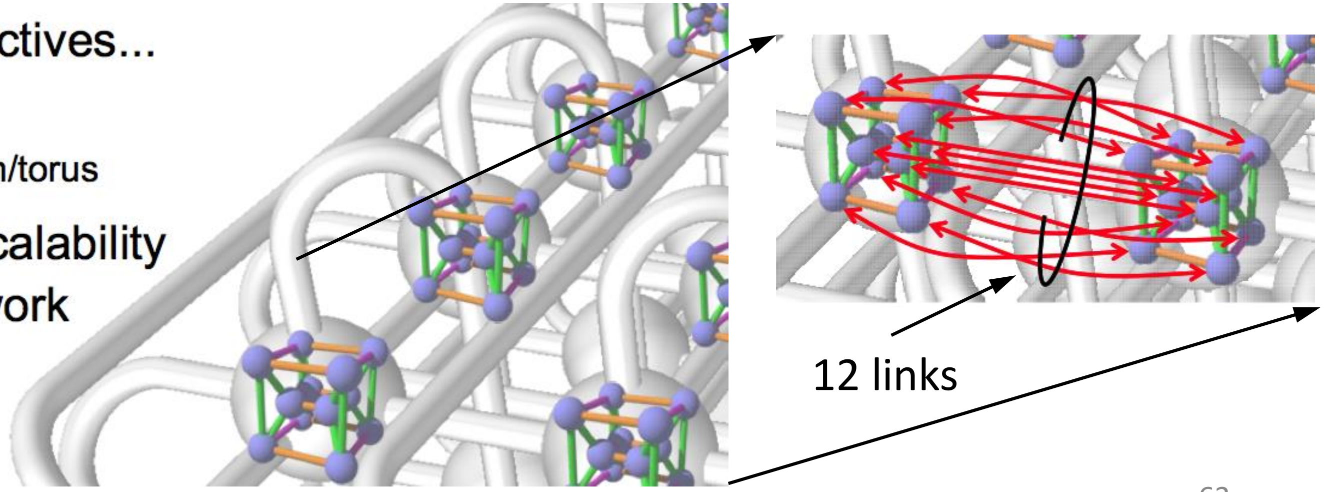
New Linpack run with 705,024 cores at 10.51 Pflop/s (88,128 CPUs)

K Computer – Interconnect

- 6 links ⇨ Scalable *xyz* 3D torus
- 4 links ⇨ Fixed size *abc* 3D mesh/torus
  - |*a*|=2, |*b*|=3, |*c*|=2 ⇨ 12 nodes



- Total topology is 6D mesh/torus
  - Cartesian product of *xyz* and *abc* mesh/torus
- From the other perspectives...
  - Overlaid twelve *xyz* torus
  - X x Y x Z array of *abc* mesh/torus
- Twelve times higher scalability than the 3D torus network



Scalable Parallel Computing

- Scalability in parallel architecture
  - Processor numbers
  - Memory architecture
  - Interconnection network
  - Avoid critical architecture bottlenecks
- Scalability in computational problem
  - Problem size
  - Computational algorithms
    - Computation to memory access ratio
    - Computation to communication ration
- Parallel programming models and tools
- Performance scalability