Lecture 1: Introduction and Overview

NMNV565: High-Performance Computing for Computational Science

About me

- https://www.karlin.mff.cuni.cz/~carson/
- Research area: numerical linear algebra, parallel algorithms, HPC

- Brief C.V.
  - Ph.D. in Computer Science from U.C. Berkeley, 2015
  - Courant Instructor/Assistant Professor, Courant Institute at NYU, 2015-2018
  - Since 2018, at MFF...

About this course

- Goals:
  - Introduction to concepts and tools used in HPC
  - Understanding of modern machine architectures, parallel programming models, and models of parallel performance
  - Develop intuition about how to map an algorithm to a parallel implementation

Overview of the course (not in order)

- Basics of computer architecture, memory hierarchies, performance
- Parallel Programming Models and Machines (plus some architecture, e.g., caches)

Algorithm/machine model	Language / Library skills
Shared memory	OpenMP
Distributed memory	MPI
Data parallel	SPARK
	CUDA

- Parallelization Strategies for the "Motifs" of Scientific Computing (and Data)

Dense Linear Algebra
Sparse Linear Algebra
Particle Methods
Structured Grids
Unstructured Grids

- Performance models: Roofline, α-β (latency/bandwidth), LogP
- Cross-cutting: Communication-avoiding, load balancing, hierarchical algorithms, autotuning, Moore's Law, Amdahl's Law, Little's Law

Course syllabus

- Syllabus available on course Moodle site (please sign up and self-enroll if you haven't been added)
  - Link in syllabus

Today's Outline

- Why ~~high performance~~ **all** machines are parallel
  - Including your laptops and phones
- Some of the World's Fastest (Parallel) Computers
  - Top500 List highlights
- Why science needs high performance computing (HPC)
  - Simulation and data analysis
- Why writing (fast) parallel programs is hard
- Measuring, understanding, and reporting performance

- If time: Brief preview of other HPC topics for the rest of this class

Units of Measure

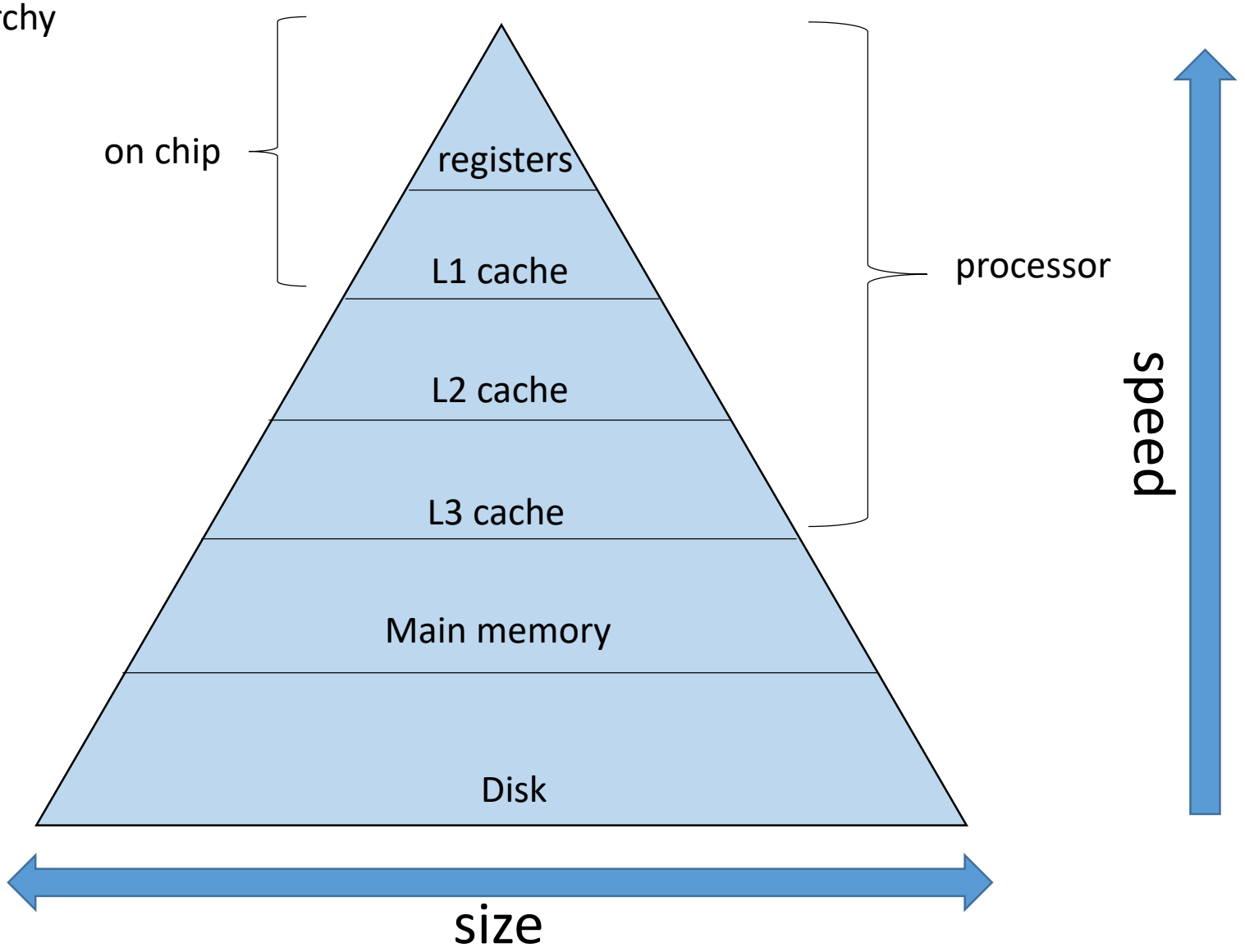
- High Performance Computing (HPC) units are:
  - Flop: floating point operation, usually double precision unless noted
  - Flop/s: floating point operations per second
  - Bytes: size of data (a double precision floating point number is 8 bytes)
- Typical sizes are millions, billions, trillions...

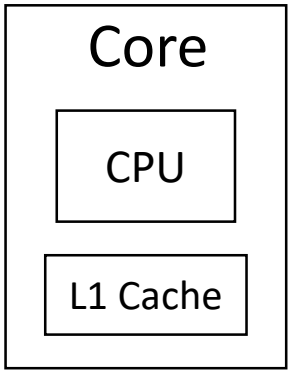
Mega	Mflop/s = 10^6 flop/sec	Mbyte = 2^{20} = 1048576 ~ 10^6 bytes
Giga	Gflop/s = 10^9 flop/sec	Gbyte = 2^{30} ~ 10^9 bytes
Tera	Tflop/s = 10^{12} flop/sec	Tbyte = 2^{40} ~ 10^{12} bytes
Peta	Pflop/s = 10^{15} flop/sec	Pbyte = 2^{50} ~ 10^{15} bytes
Exa	Eflop/s = 10^{18} flop/sec	Ebyte = 2^{60} ~ 10^{18} bytes
Zetta	Zflop/s = 10^{21} flop/sec	Zbyte = 2^{70} ~ 10^{21} bytes
Yotta	Yflop/s = 10^{24} flop/sec	Ybyte = 2^{80} ~ 10^{24} bytes

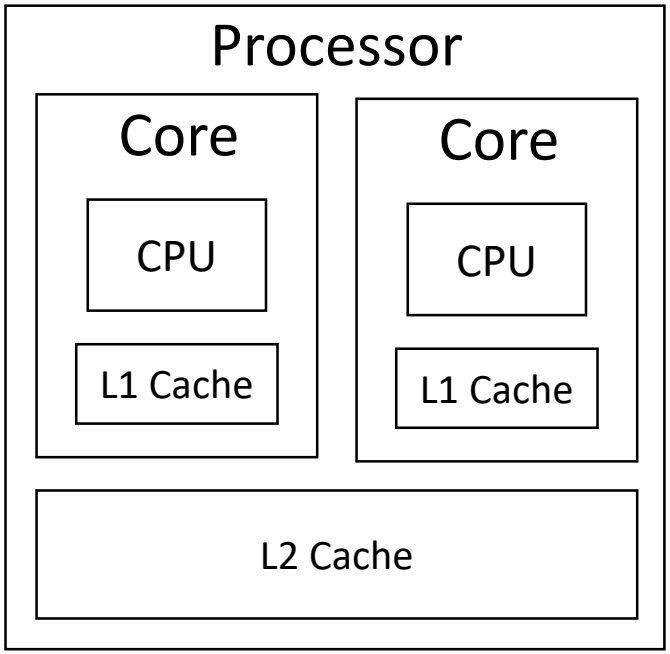
- Current fastest (public) machine >1 Eflop/s
  - Up-to-date list at www.top500.org

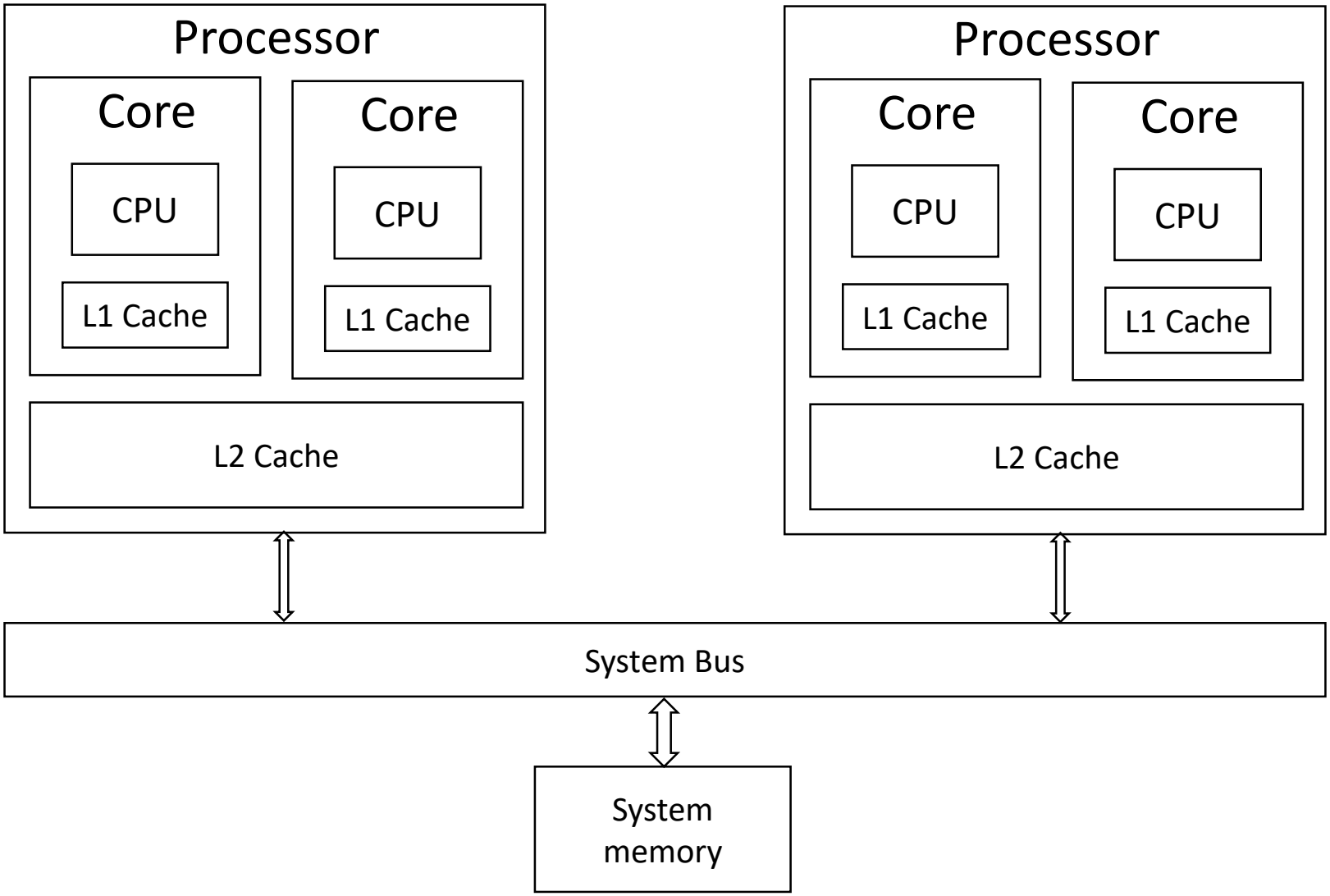
Basics of Computer Architecture

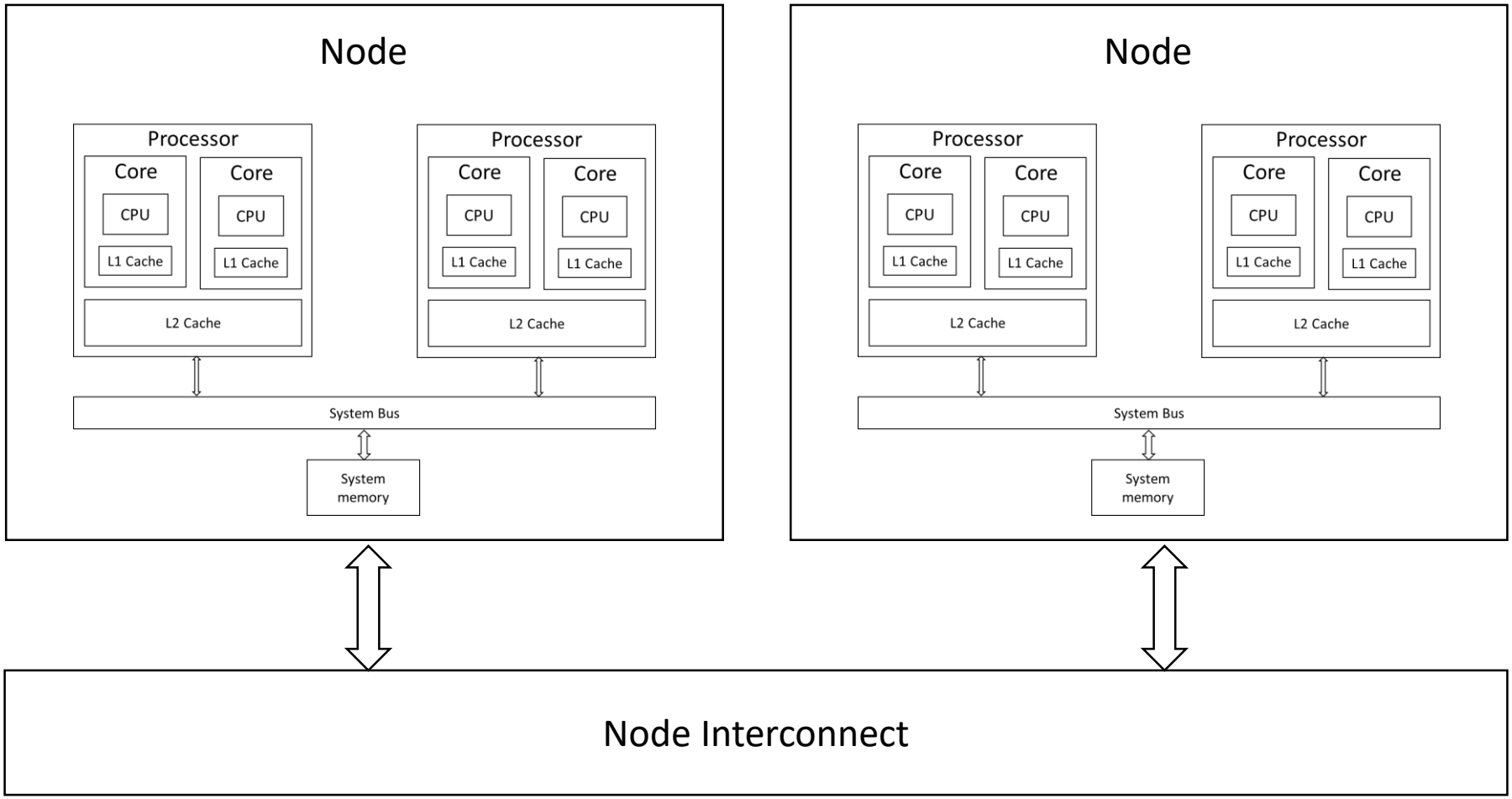
Memory Hierarchy

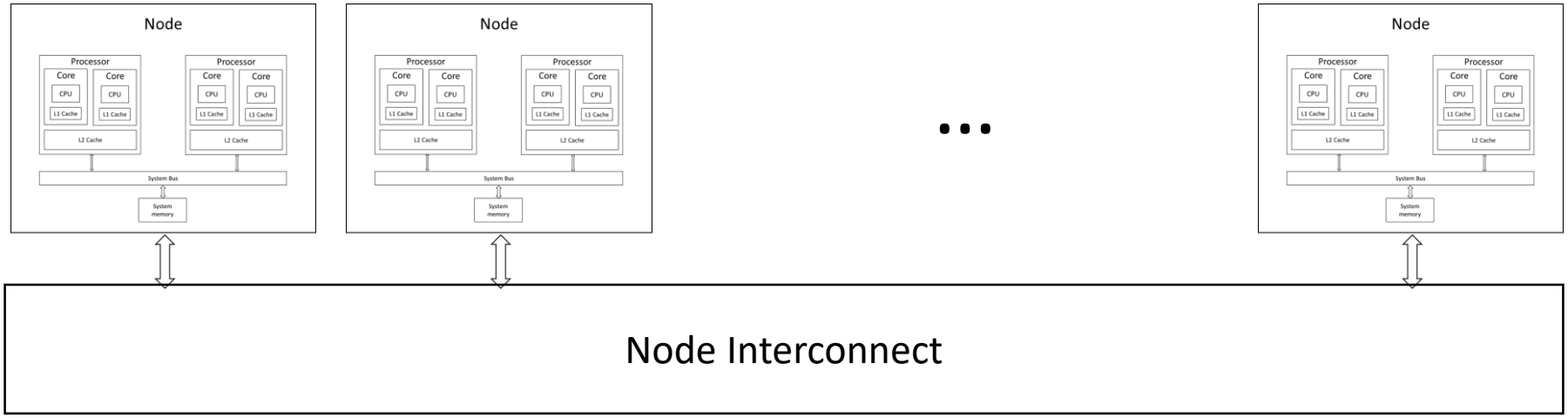



Core
CPU
L1 Cache


Processor
Core
CPU
L1 Cache
Core
CPU
L1 Cache
L2 Cache


Processor
Core
CPU
L1 Cache
Core
CPU
L1 Cache
L2 Cache
Processor
Core
CPU
L1 Cache
Core
CPU
L1 Cache
L2 Cache
System Bus
System memory


Node
Processor
Core
CPU
L1 Cache
Core
CPU
L1 Cache
L2 Cache
Processor
Core
CPU
L1 Cache
Core
CPU
L1 Cache
L2 Cache
System Bus
System memory
Node
Processor
Core
CPU
L1 Cache
Core
CPU
L1 Cache
L2 Cache
Processor
Core
CPU
L1 Cache
Core
CPU
L1 Cache
L2 Cache
System Bus
System memory
Node Interconnect


Node
Processor
Core
CPU
L1 Cache
L2 Cache
System Bus
System memory
Node
Processor
Core
CPU
L1 Cache
L2 Cache
System Bus
System memory
...
Node
Processor
Core
CPU
L1 Cache
L2 Cache
System Bus
System memory
Node Interconnect

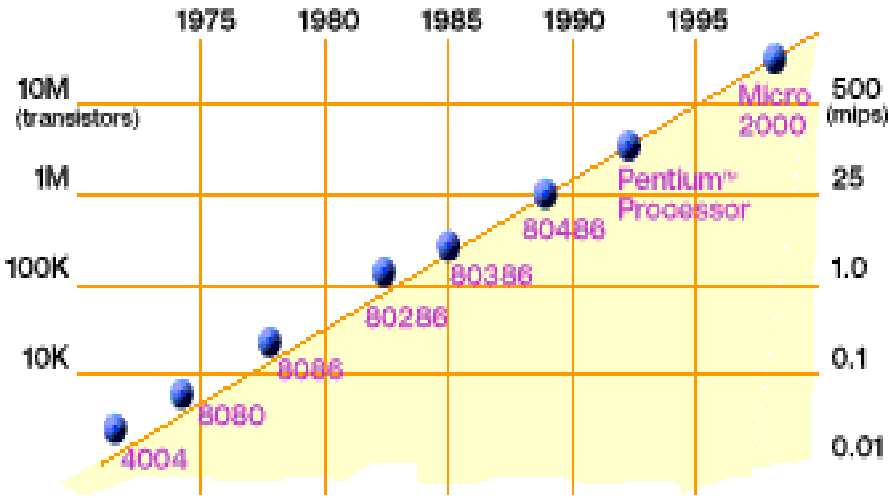
Why ~~high-performance~~ ALL machines are parallel

Tunnel Vision by Experts

- "I think there is a world market for maybe five computers."
  - Thomas Watson, chairman of IBM, 1943.
- "There is no reason for any individual to have a computer in their home"
  - Ken Olson, president and founder of Digital Equipment Corporation, 1977.
- "640K [of memory] ought to be enough for anybody."
  - Bill Gates, chairman of Microsoft,1981.
- "On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it."
  - Ken Kennedy, CRPC Directory, 1994

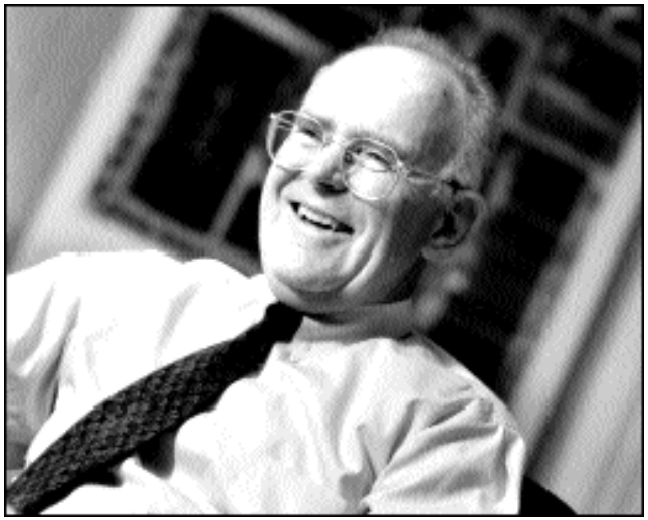
Slide source: Warfield et al.

Technology Trends: Microprocessor Capacity



2X transistors/Chip Every 1.5 years
Called "Moore's Law"

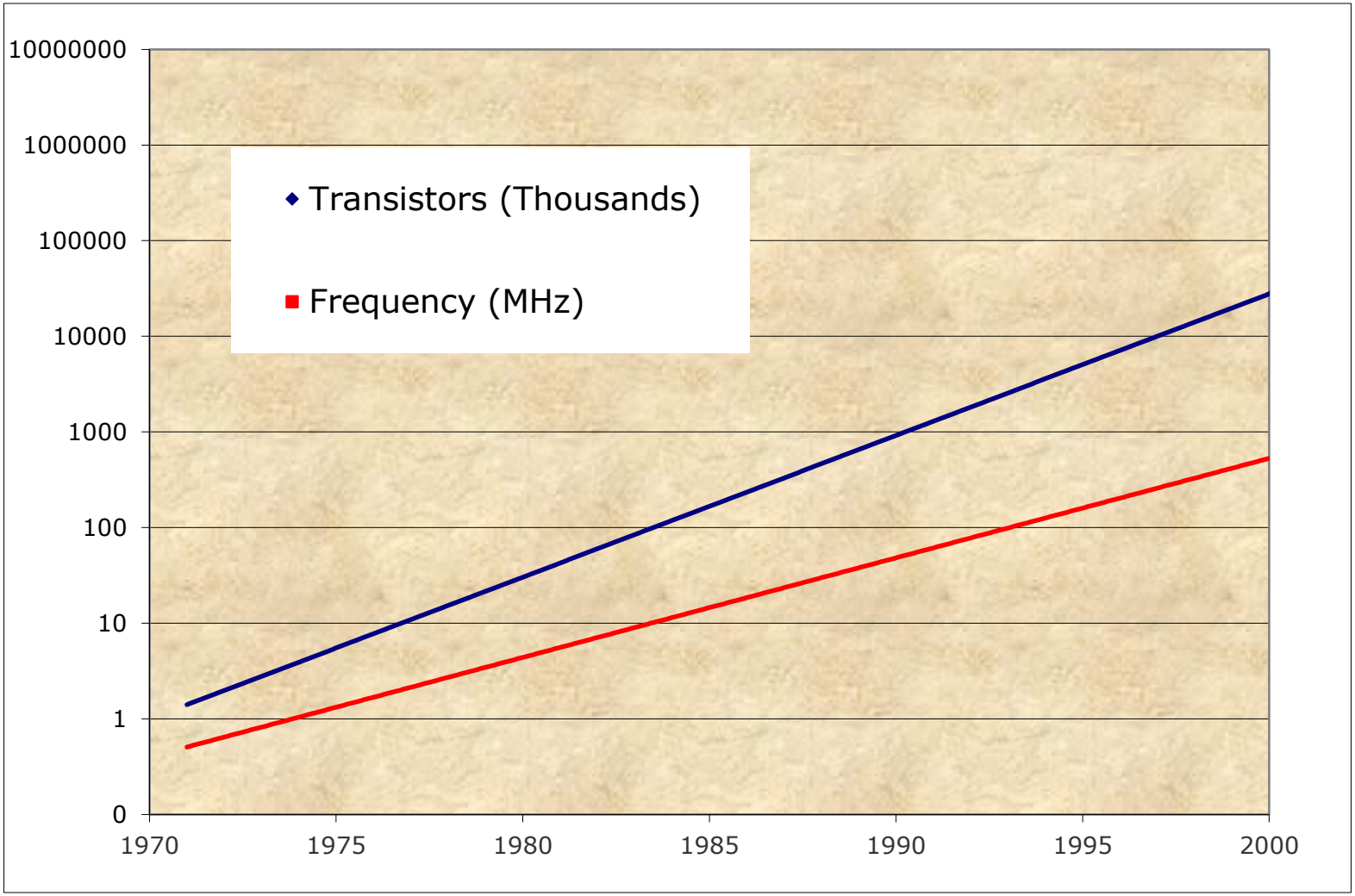
Microprocessors have become smaller, denser, and more powerful.



Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Slide source: Jack Dongarra

Microprocessor Transistors / Clock (1970-2000)

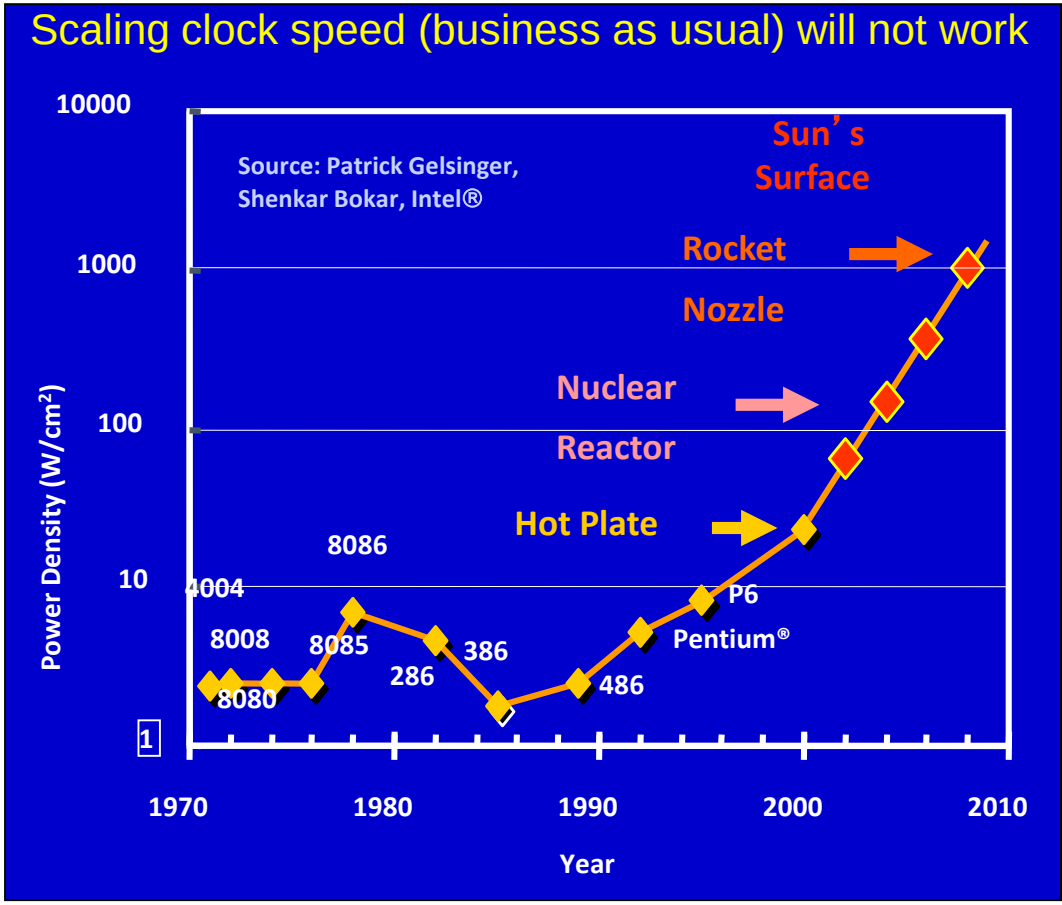


Historical Impact of Device Shrinkage

- What happens when the feature size (transistor size) shrinks by a factor of x ?
- Clock rate goes up by x because wires are shorter
  - actually less than x, because of power consumption
- Transistors per unit area goes up by x^2
- Die size has also increased
  - typically another factor of ~x
- Raw computing power of the chip goes up by ~ x^4 !
  - typically x^3 is devoted to either on-chip
    - parallelism: hidden parallelism such as ILP
    - locality: caches
- So most programs x^3 times faster, without changing them

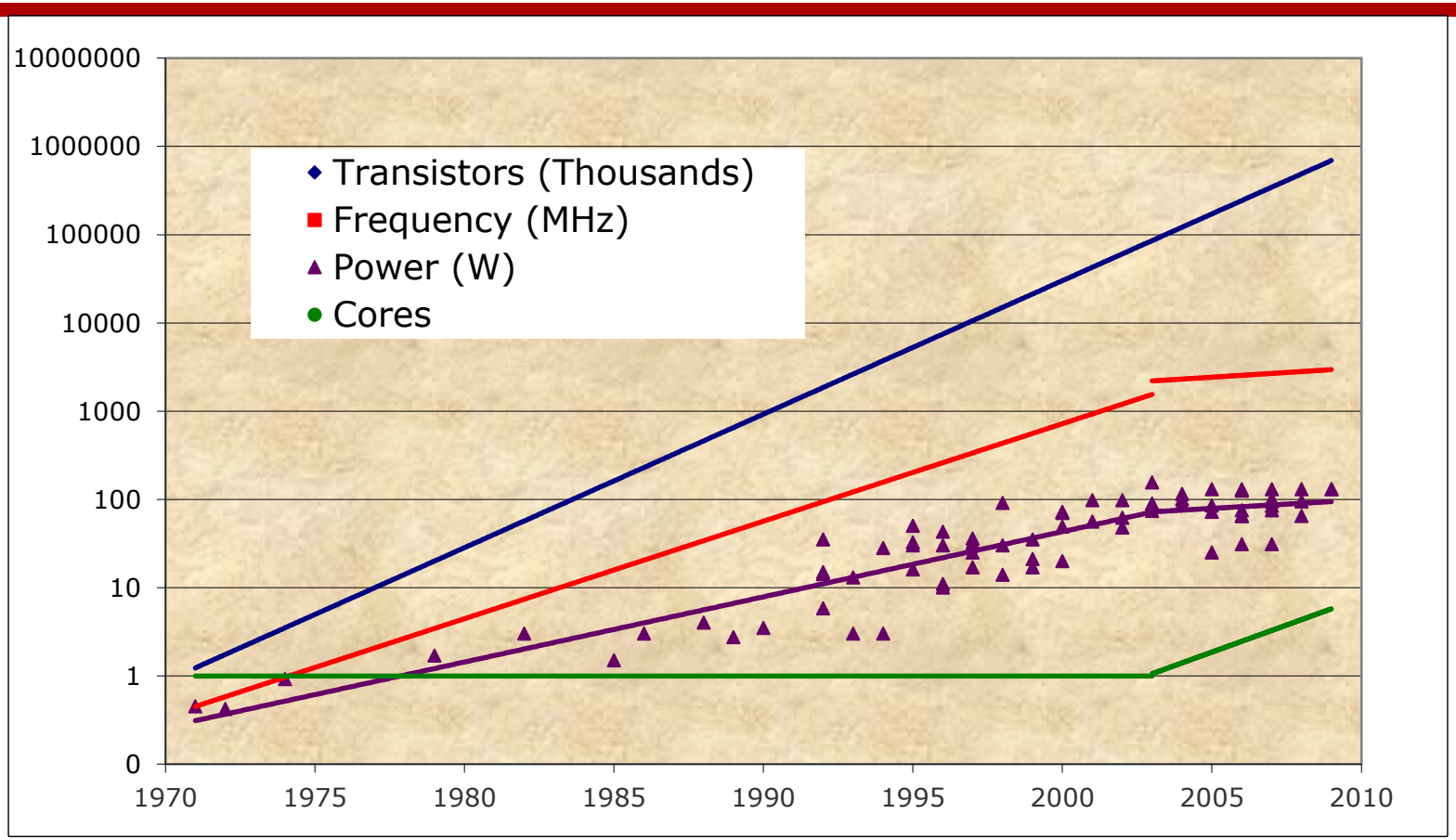
Power Density Limits Serial Performance

- Concurrent systems are more power efficient
  - Dynamic power is proportional to V^2fC
  - Increasing frequency (f) also increases supply voltage (V) → cubic effect
  - Increasing cores increases capacitance (C) but only linearly
  - Save power by lowering clock speed



- High performance serial processors waste power
  - Speculation, dynamic dependence checking, etc. burn power
  - Implicit parallelism discovery
- More transistors, but not faster serial processors

Revolution in Processors



- Chip density is continuing increase ~2x every 2 years
- Clock speed is not
- Number of processor cores may double instead
- Power is under control, no longer growing

Moore's Law reinterpreted

- Number of cores per chip can double every two years
- Clock speed will not increase (possibly decrease)
- Need to deal with systems with millions of concurrent threads
- Need to deal with inter-chip parallelism as well as intra-chip parallelism
- But Moore's Law is not forever... industry consortium predicts end in 2020-2030

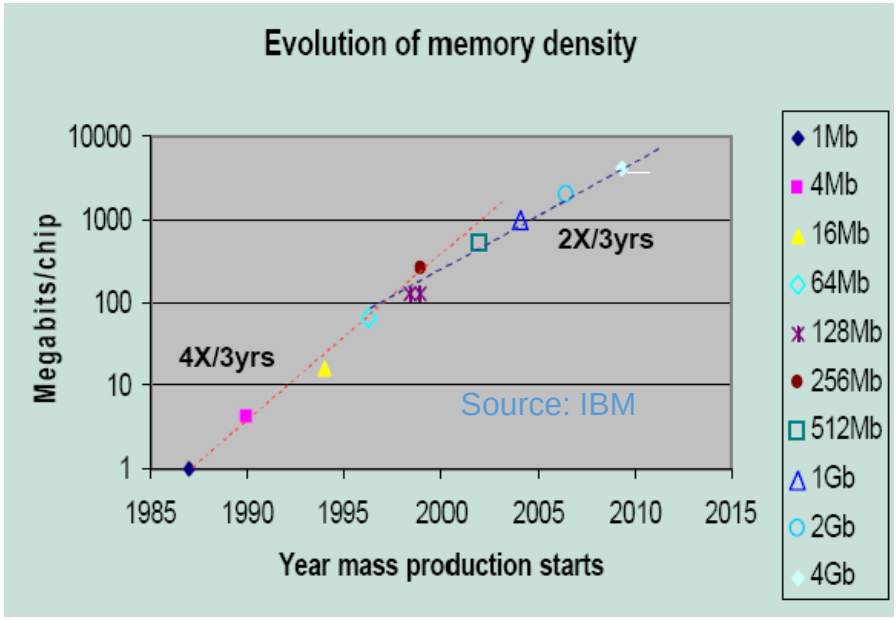
Parallelism today?

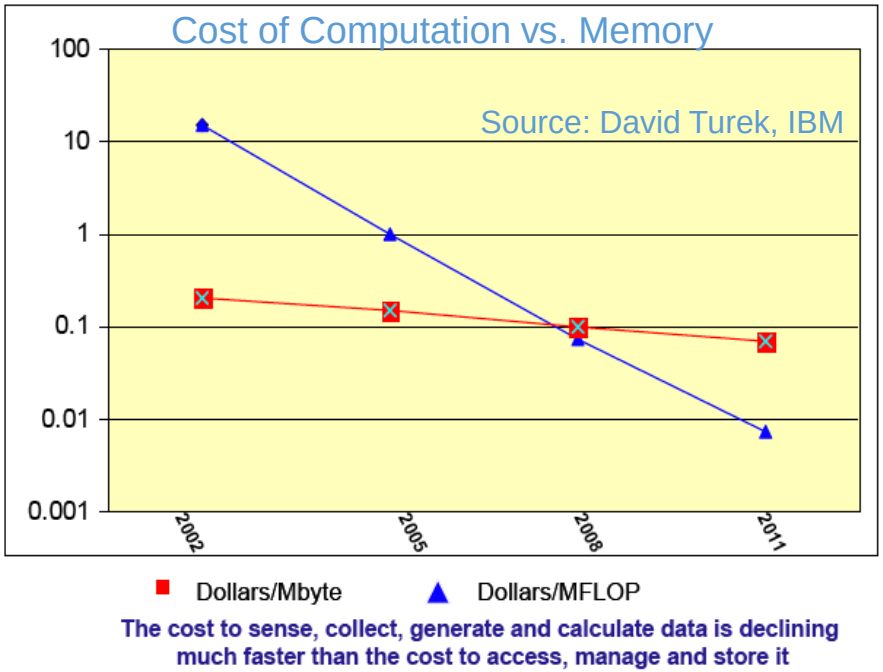
- These arguments are no longer theoretical
- All major processor vendors are producing *multicore* chips
  - Every machine will soon be a parallel machine
  - To keep doubling performance, parallelism must double
- Which applications can use this parallelism?
  - Do they have to be rewritten from scratch?
- Will all programmers have to be parallel programmers?
  - New software model needed
  - Try to hide complexity from most programmers – eventually
  - In the meantime, need to understand it
- Computer industry betting on this big change, but does not have all the answers

Memory is Not Keeping Pace

Technology trends against a constant or increasing memory per core

- Memory density is doubling every three years; processor logic is every two
- Storage costs (dollars/Mbyte) are dropping gradually compared to logic costs





Question: Can you double concurrency without doubling memory?

- **Strong scaling:** fixed problem size, increase number of processors
- **Weak scaling:** grow problem size proportionally to number of processors

Supercomputers

What is a supercomputer?

- A computer with a high level of computing performance
- Performance is measured in floating-point operations per second (FLOPS)
- First produced by Cray in the 60s

- play an important role in the field of computational science: quantum mechanics, weather forecasting, climate research, oil and gas exploration, molecular modeling, physical simulations, cryptanalysis

Supercomputing History

- First supercomputer: CDC 6600, 1964
- First installed at CERN
- Performance: 3 Mflop/s
  - (iphone 6 does about 1 Gflop/s)
- 1 CPU, a few "peripheral" processors
- 80s: ~8 processors
- 90s: ~1000 processors

- Today: millions of cores, augmented with computational accelerators (like GPUs); high-speed, low-latency interconnects; storage area networks for persistent data storage with local disks used only for temporary files

Top500 June 2023

Rank (previous)	Rmax Rpeak (PetaFLOPS)	Name	Model	CPU cores	Accelerator (e.g. GPU) cores	Interconnect	Manufacturer	Site country	Year	Operating system
1 —	1,194.00 1,679.82	Frontier	HPE Cray EX235a	561,664 (8,776 × 64-core Optimized 3rd Generation EPYC 64C @2.0 GHz)	36,992 × 220 AMD Instinct MI250X	Slingshot-11	HPE	Oak Ridge National Laboratory United States	2023	Linux (HPE Cray OS-SUSE)
2 —	442.010 537.212	Fugaku	Supercomputer Fugaku	7,630,848 (158,976 × 48-core Fujitsu A64FX @2.2 GHz)	158,976 x Fujitsu A64FX	Tofu interconnect D	Fujitsu	RIKEN Center for Computational Science Japan	2020	Linux (RHEL)
3 —	309.10 428.70	LUMI	HPE Cray EX235a	150,528 (2,352 × 64-core Optimized 3rd Generation EPYC 64C @2.0 GHz)	9,408 × 220 AMD Instinct MI250X	Slingshot-11	HPE	EuroHPC JU European Union, Kajaani, Finland	2022	Linux (HPE Cray OS-SUSE)
4 —	238.70 304.47	Leonardo	BullSequana XH2000	110,592 (3,456 × 32-core Xeon Platinum 8358 @2.6 GHz)	15,872 × 108 Nvidia Ampere A100	Nvidia HDR100 Infiniband	Atos	EuroHPC JU European Union, Bologna, Italy	2023	Linux
5 —	148.600 200.795	Summit	IBM Power System AC922	202,752 (9,216 × 22-core IBM POWER9 @3.07 GHz)	27,648 × 80 Nvidia Tesla V100	InfiniBand EDR	IBM	Oak Ridge National Laboratory United States	2018	Linux (RHEL 7.4)
6 —	94.640 125.712	Sierra	IBM Power System S922LC	190,080 (8,640 × 22-core IBM POWER9 @3.1 GHz)	17,280 × 80 Nvidia Tesla V100	InfiniBand EDR	IBM	Lawrence Livermore National Laboratory United States	2018	Linux (RHEL)

Rmax: performance on LINPACK benchmark, Rpeak: theoretical peak of the machine

Czech Republic in Top500

June 2023:

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
95	**Karolina, GPU partition** - Apollo 6500, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Infiniband HDR200, HPE IT4Innovations National Supercomputing Center, VSB-Technical University of Ostrava Czechia	71,424	6.75	9.08	311
264	**Karolina, CPU partition** - Apollo 2000, AMD EPYC 7H12 64C 2.6GHz, InfiniBand HDR 100, HPE IT4Innovations National Supercomputing Center, VSB-Technical University of Ostrava Czechia	92,160	2.84	3.83	503

The Linpack Benchmark

- Benchmark involves solving nonsingular Ax=b using Gaussian elimination with partial pivoting (GEPP) ($2/3n^3 + 2n^2$ flops)

- Details about the benchmark
  - http://www.netlib.org/benchmark/hpl/
  - http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html

- Used as the Benchmark for supercomputers since the start of the Top500 in 1993 (for better or for worse...)
  - Is this indicative of typical scientific applications?
    - Compute-bound kernel...
  - Are supercomputer architectures *designed* to do well on the LINPACK benchmark?

Graph500 (Since 2010)

- Two rankings based on breadth-first search (BFS) and single source shortest path (SSSP) algorithms; designed to model data-intensive workloads
- performance measured in GTEPS (giga-traversed edges per second)
- http://graph500.org/

June 2021 SSSP:

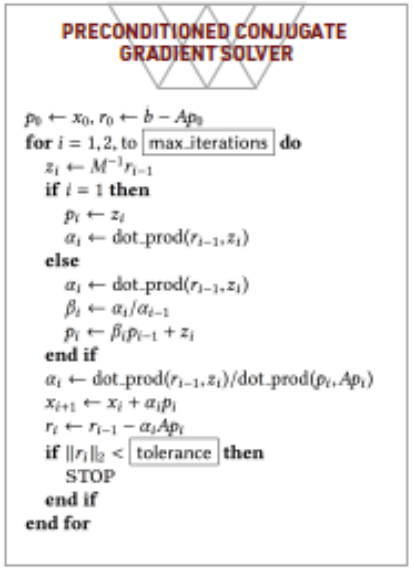
RANK	MACHINE	VENDOR	INSTALLATION SITE	LOCATION	COUNTRY	YEAR	NUMBER OF NODES	NUMBER OF CORES	SCALE	GTEPS
1	Supercomputer Fugaku	Fujitsu	RIKEN Center for Computational Science (R-CCS)	Kobe Hyogo	Japan	2020	158976	7630848	41	102955
2	Sunway TaihuLight	NRCPC	National Supercomputing Center in Wuxi	Wuxi	China	2015	40768	10599680	40	23755.7
3	Wisteria/BDEC-01 (Odyssey)	Fujitsu	Information Technology Center The University of Tokyo	Kashiwa Chiba	Japan	2021	7680	368640	37	16118
4	TOKI-SORA	Fujitsu	Japan Aerospace eXploration Agency (JAXA)	Tokyo	Japan	2020	5760	276480	36	10813
5	LUMI-C	HPE	EuroHPC/CSC	Kajaani	Finland	2021	1492	190976	38	8467.71

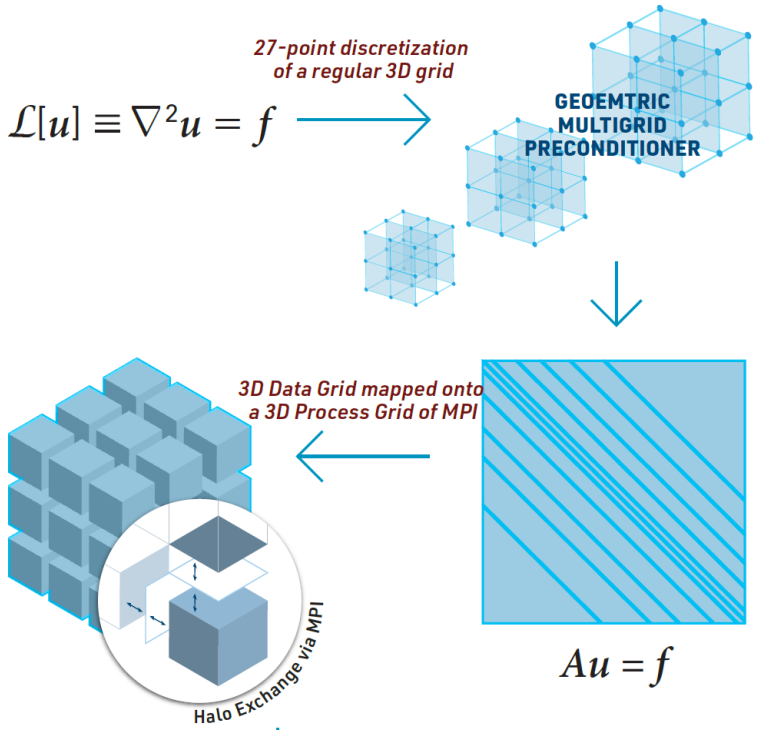
HPCG Benchmark (Since 2015)

- Intended to model the data access patterns of real-world applications such as sparse matrix calculations; test limitations of the memory subsystem and internal interconnect of the supercomputer on its computing performance
- http://www.hpcg-benchmark.org/

The HPC Conjugate Gradient (HPCG) benchmark uses a Preconditioned Conjugate Gradient (PCG) algorithm to measure the performance of HPC platforms with respect to frequently observed, and yet challenging, patterns of execution, memory access, and global communication.

The PCG implementation uses a regular 27-point stencil discretization in 3 dimensions of an elliptic Partial Differential Equation (PDE). The 3D domain is scaled to fill a 3D virtual process grid of all available MPI process ranks. The CG iteration includes a local and symmetric Gauss-Seidel preconditioner, which computes a forward and a back solve with a triangular matrix. All of these features combined allow HPCG to deliver a more accurate performance metric for modern HPC hardware architectures.





HPCG Benchmark

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	HPCG (TFlop/s)
1	2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	16004.50
2	4	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	2925.75
3	3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	1935.73
4	7	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70.87	1905.44
5	5	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94.64	1795.67

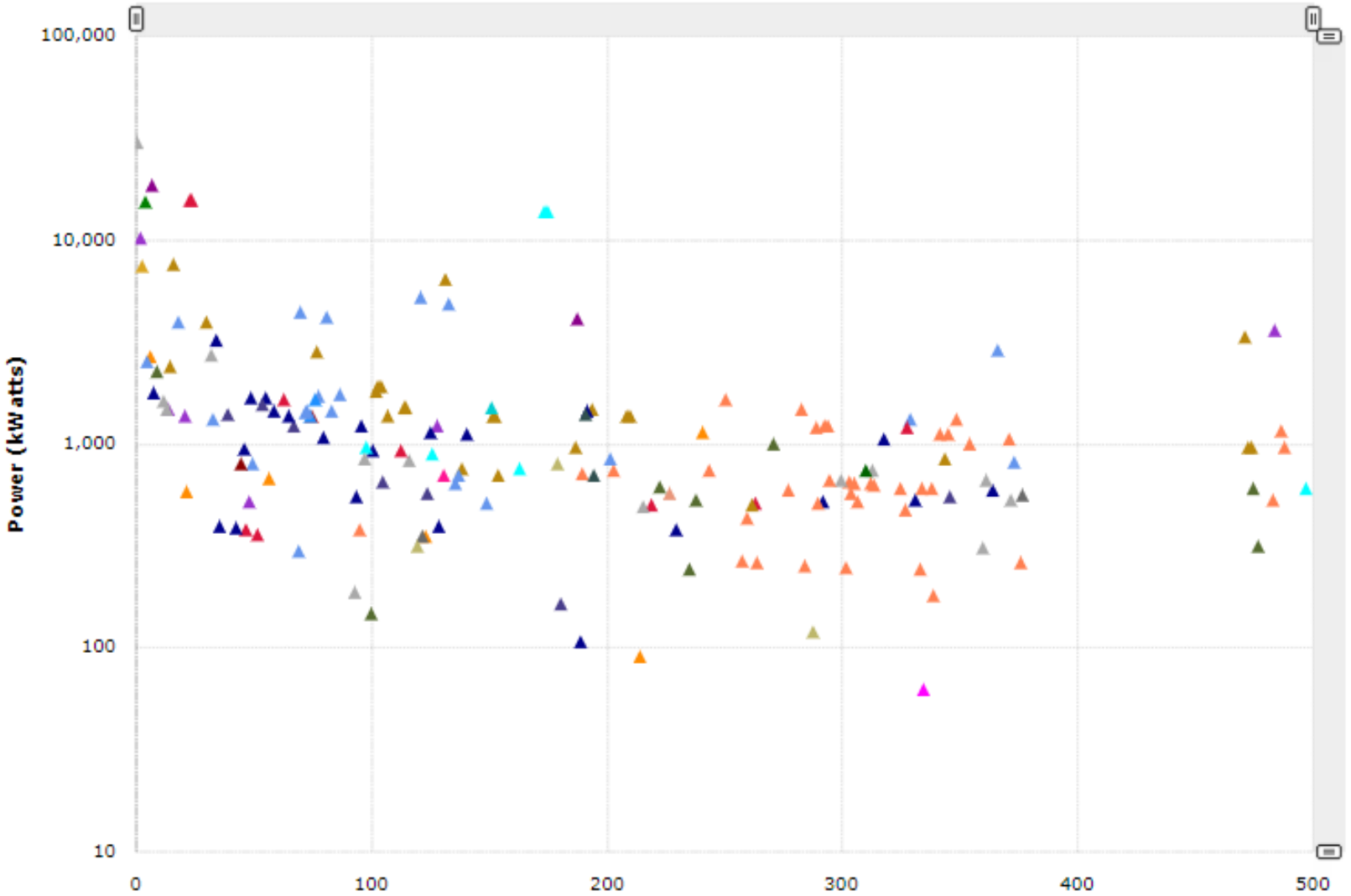
Rpeak for Fugaku:
537,210 Tflops/s

HPL gets about 80% of theoretical peak.

HPCG only gets about 3% of theoretical peak!

Sparse computations ⇒ communication-bound performance

Power, Top500 June 2021



Green500

- Since 2007
- ranks the top 500 supercomputers in the world by energy efficiency
- https://www.top500.org/green500/

About the Green500 List

The Green500 list ranks the top 500 supercomputers in the world by energy efficiency. The focus of performance-at-any-cost computer operations has led to the emergence of supercomputers that consume vast amounts of electrical power and produce so much heat that large cooling facilities must be constructed to ensure proper performance. To address this trend, the Green500 list puts a premium on energy-efficient performance for sustainable supercomputing.

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	Power (kW)	Energy Efficiency (GFlops/watts)
1	29	**Frontier TDS** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	120,832	19.20	309	62.684
2	1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	21,100	52.227
3	3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	2,942	51.629
4	10	**Adastra** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Grand Equipement National de Calcul Intensif - Centre Informatique National de l'Enseignement Suprieur (GENCI-CINES) France	319,072	46.10	921	50.028

Summit (#5 machine) System Overview

System Performance

- Peak performance of 200 petaflops for modeling & simulation
- Peak of 3.3 ExaOps for data analytics and artificial intelligence

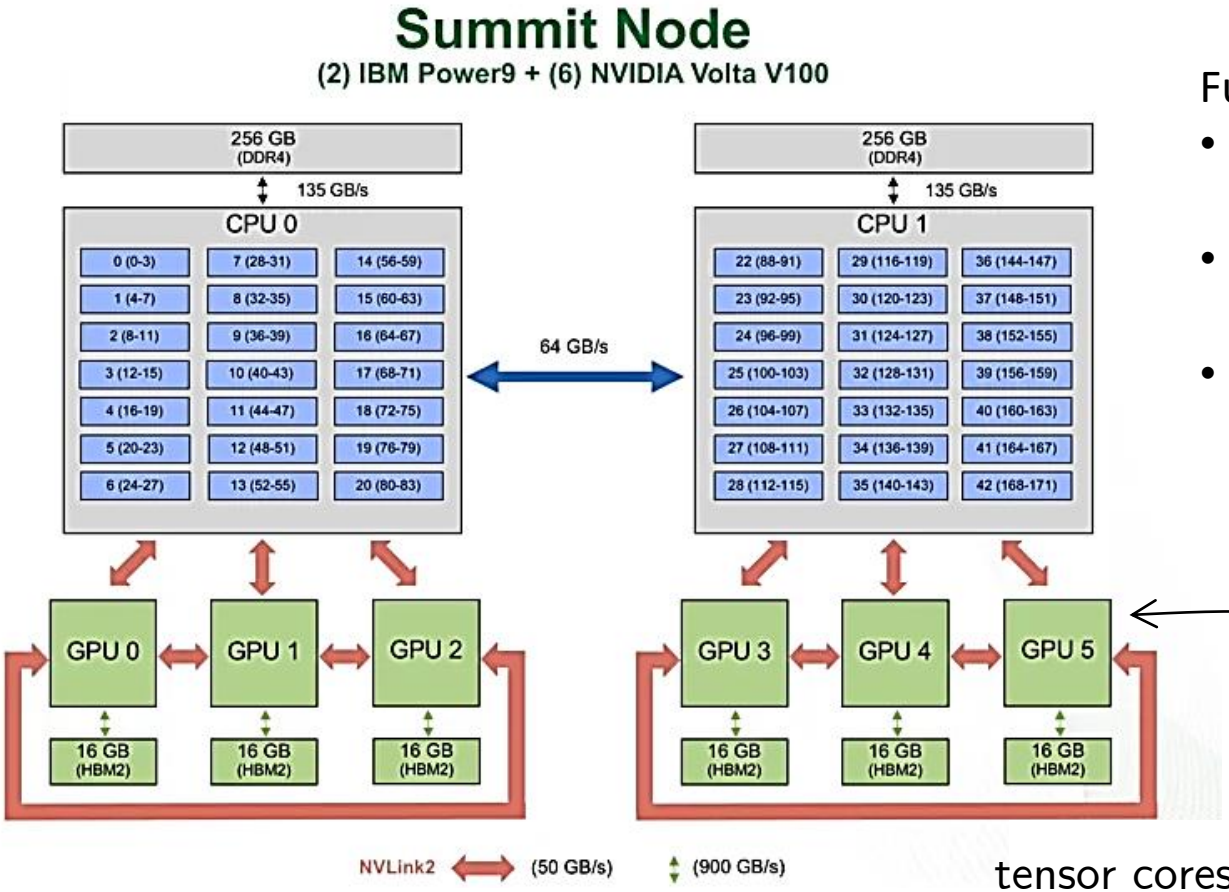
Each node has

- 2 IBM POWER9 processors
- 6 NVIDIA Tesla V100 GPUs
- 608 GB of fast memory
- 1.6 TB of NVMe memory

The system includes

- 4608 nodes
- Dual-rail Mellanox EDR InfiniBand network
- 250 PB IBM Spectrum Scale file system transferring data at 2.5 TB/s

HPC Architectures Today



https://www.olcf.ornl.gov/for-users/system-user-guides/summit/summit-user-guide

Future architectures:

- non-uniform memory access
- **GPUS/FPGAs, other specialized accelerators**
- multiple hardware precisions

tensor cores for half precision:

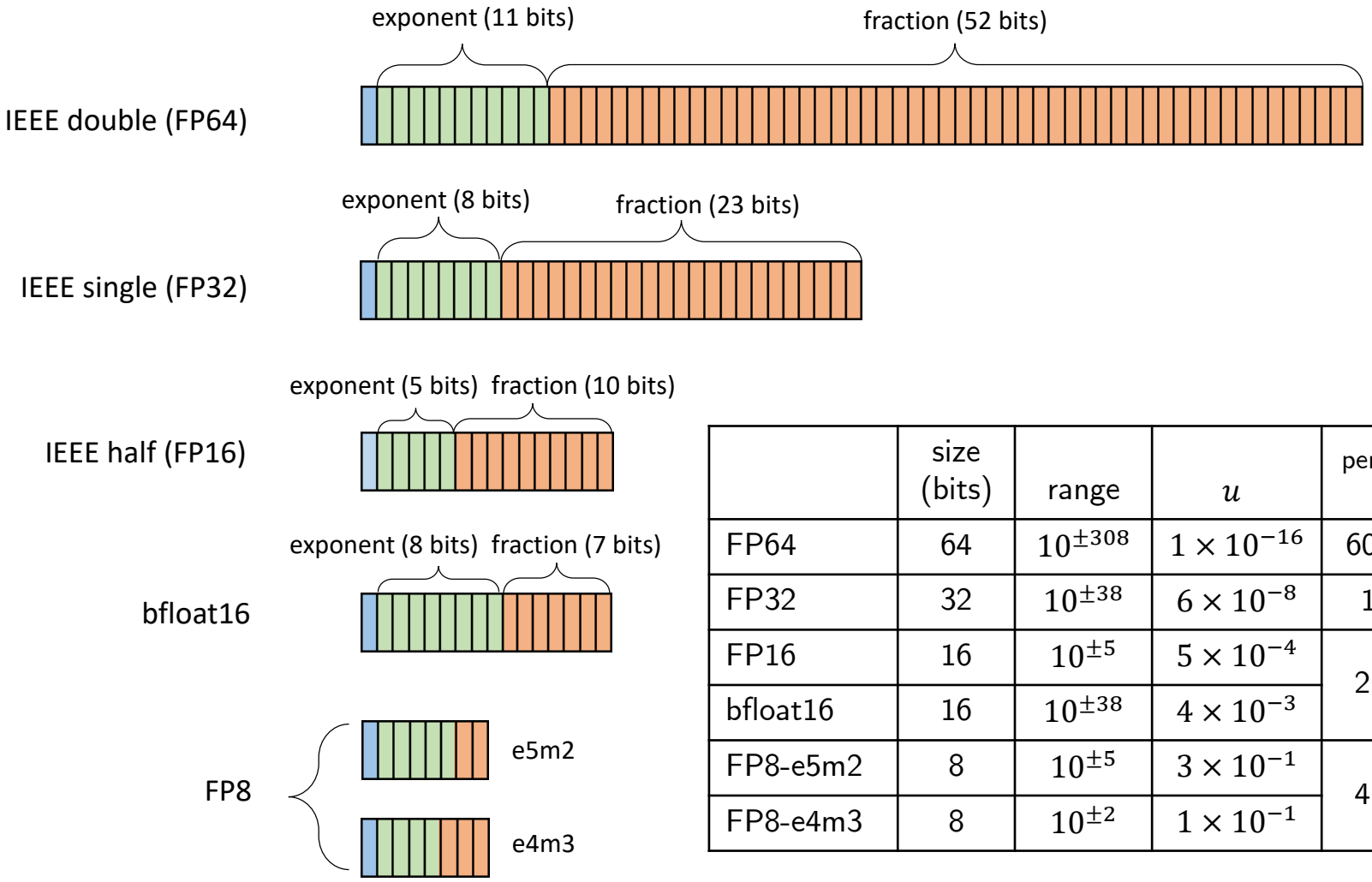
4x4 matrix multiply in *one clock cycle*

double: 7 TFLOPS,

half+tensor: 112 TFLOPS (16x!)

Floating Point Formats

$$(-1)^{\text{sign}} \times 2^{(\text{exponent}-\text{offset})} \times 1.\text{fraction}$$



	size (bits)	range	u	perf. (NVIDIA H100)
FP64	64	$10^{\pm 308}$	1×10^{-16}	60 Tflops/s
FP32	32	$10^{\pm 38}$	6×10^{-8}	1 Pflop/s
FP16	16	$10^{\pm 5}$	5×10^{-4}	2 Pflops/s
bfloat16	16	$10^{\pm 38}$	4×10^{-3}	
FP8-e5m2	8	$10^{\pm 5}$	3×10^{-1}	4 Pflops/s
FP8-e4m3	8	$10^{\pm 2}$	1×10^{-1}	

Hardware Support for Multiprecision Computation

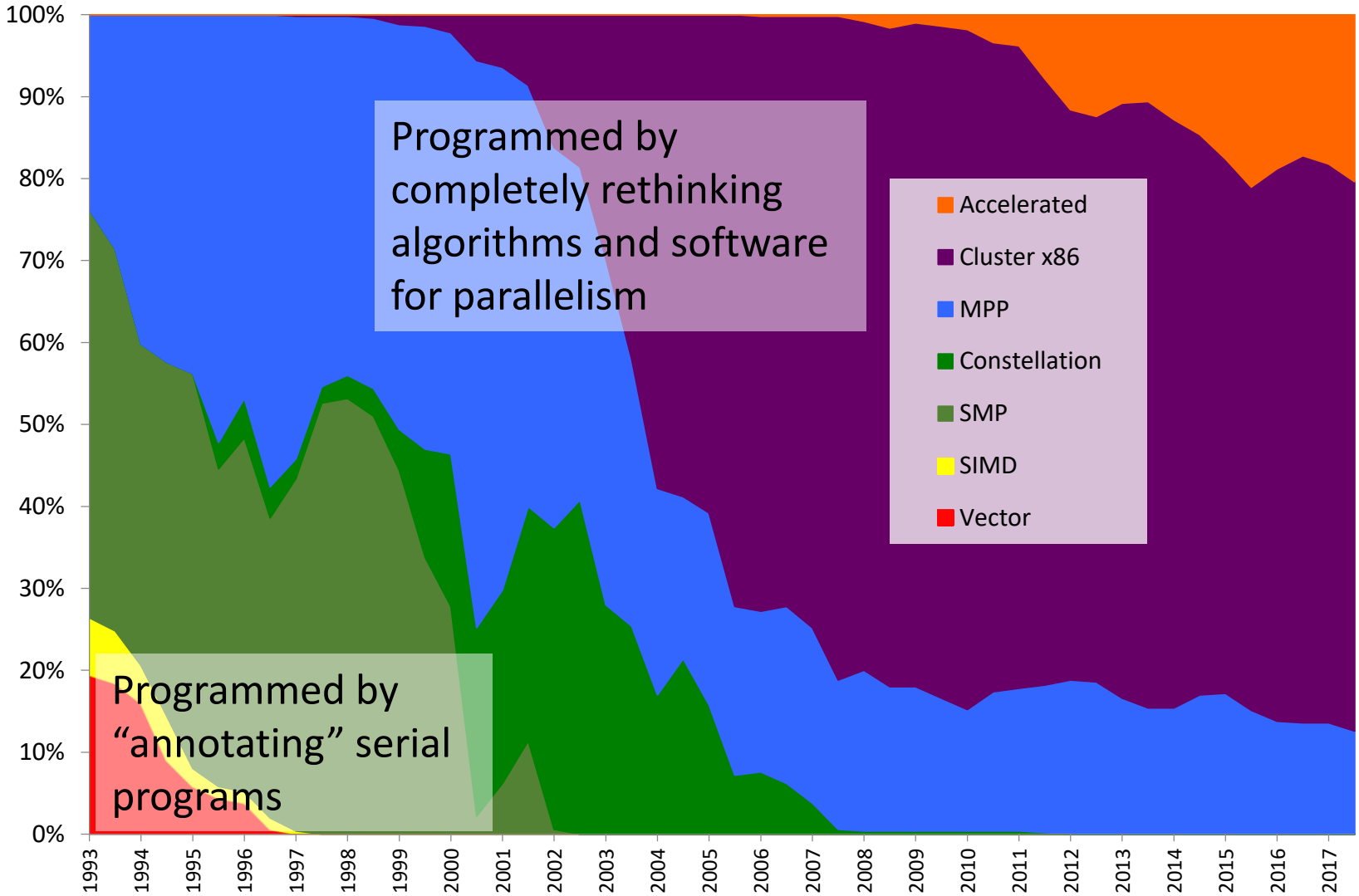
Use of low precision in machine learning has driven emergence of low-precision capabilities in hardware:

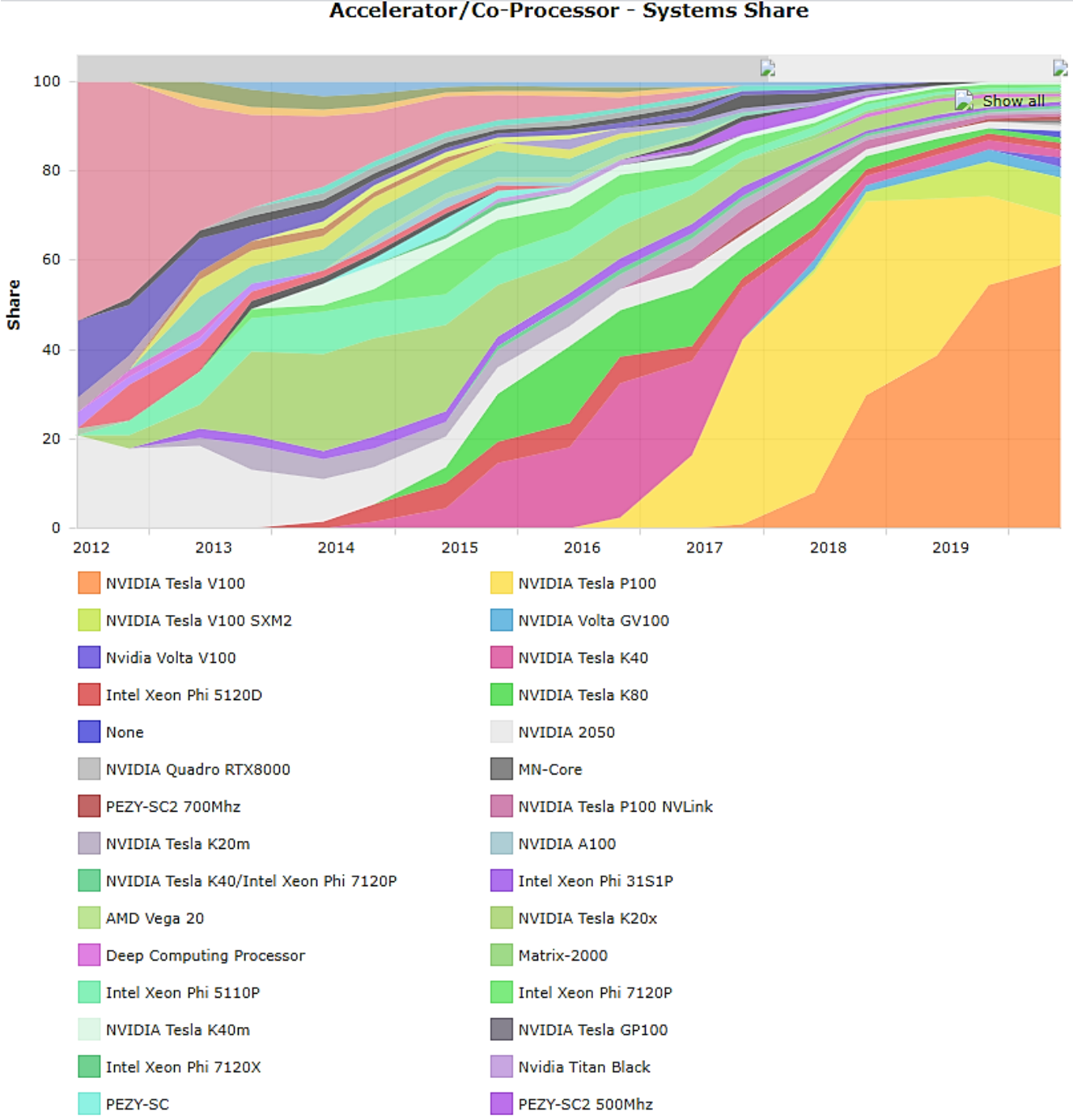
- Half precision (FP16) defined as storage format in 2008 IEEE standard
- ARM NEON: SIMD architecture, instructions for 8x16-bit, 4x32-bit, 2x64-bit
- AMD Radeon Instinct MI25 GPU, 2017:
  - single: 12.3 TFLOPS, half: 24.6 TFLOPS
- NVIDIA Tesla P100, 2016: native ISA support for 16-bit FP arithmetic
- NVIDIA Tesla V100, 2017: tensor cores for half precision;
    4x4 matrix multiply in one clock cycle
  - double: 7 TFLOPS, half+tensor: 112 TFLOPS (**16x!**)
- Google's Tensor processing unit (TPU)
- NVIDIA A100, 2020: tensor cores with multiple supported precisions: FP16, FP64, Binary, INT4, INT8, bfloat16
- NVIDIA H100, 2022: now with quarter-precision (FP8) tensor cores
- Exascale supercomputers: Expected extensive support for reduced-precision arithmetic (Frontier: FP64, FP32, FP16, bfloat16, INT8, INT4)

The rise of accelerators

1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
6	**HPC5** - PowerEdge C4140, Xeon Gold 6252 24C 2.1GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, Dell EMC Eni S.p.A. Italy	669,760	35,450.0	51,720.8	2,252
7	**Selene** - DGX A100 SuperPOD, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	272,800	27,580.0	34,568.6	1,344

From Vector Supercomputers to Massively Parallel Accelerator Systems




Accelerator/Co-Processor - Systems Share
Share
100
80
60
40
20
0
2012
2013
2014
2015
2016
2017
2018
2019
Show all
NVIDIA Tesla V100
NVIDIA Tesla P100
NVIDIA Tesla V100 SXM2
NVIDIA Volta GV100
Nvidia Volta V100
NVIDIA Tesla K40
Intel Xeon Phi 5120D
NVIDIA Tesla K80
None
NVIDIA 2050
NVIDIA Quadro RTX8000
MN-Core
PEZY-SC2 700Mhz
NVIDIA Tesla P100 NVLink
NVIDIA Tesla K20m
NVIDIA A100
NVIDIA Tesla K40/Intel Xeon Phi 7120P
Intel Xeon Phi 31S1P
AMD Vega 20
NVIDIA Tesla K20x
Deep Computing Processor
Matrix-2000
Intel Xeon Phi 5110P
Intel Xeon Phi 7120P
NVIDIA Tesla K40m
NVIDIA Tesla GP100
Intel Xeon Phi 7120X
Nvidia Titan Black
PEZY-SC
PEZY-SC2 500Mhz

Mixed precision in NLA

- BLAS: cuBLAS, MAGMA, [Agullo et al. 2009], [Abdelfattah et al., 2019], [Haidar et al., 2018]
- Iterative refinement:
  - Long history: [Wilkinson, 1963], [Moler, 1967], [Stewart, 1973], ...
  - More recently: [Langou et al., 2006], [C., Higham, 2017], [C., Higham, 2018], [C., Higham, Pranesh, 2020], [Amestoy et al., 2021]
- Matrix factorizations: [Haidar et al., 2017], [Haidar et al., 2018], [Haidar et al., 2020], [Abdelfattah et al., 2020]
- Eigenvalue problems: [Dongarra, 1982], [Dongarra, 1983], [Tisseur, 2001], [Davies et al., 2001], [Petschow et al., 2014], [Alvermann et al., 2019]
- Sparse direct solvers: [Buttari et al., 2008]
- Orthogonalization: [Yamazaki et al., 2015]
- Multigrid: [Tamstorf et al., 2020], [Richter et al., 2014], [Sumiyoshi et al., 2014], [Ljungkvist, Kronbichler, 2017, 2019]
- (Preconditioned) Krylov subspace methods: [Emans, van der Meer, 2012], [Yamagishi, Matsumura, 2016], [C., Gergelits, Yamazaki, 2021], [Clark, 2019], [Anzt et al., 2019], [Clark et al., 2010], [Gratton et al., 2020], [Arioli, Duff, 2009], [Hogg, Scott, 2010]

For survey and references, see [Abdelfattah et al., IJHPC, 2021], [Higham and Mary, 2022]

HPL-MxP Benchmark https://hpl-mxp.org/

HPL-MXP MIXED-PRECISION BENCHMARK

The HPL-MxP benchmark seeks to highlight the emerging convergence of high-performance computing (HPC) and artificial intelligence (AI) workloads. While traditional HPC focused on simulation runs for modeling phenomena in physics, chemistry, biology, and so on, the mathematical models that drive these computations require, for the most part, 64-bit accuracy. On the other hand, the machine learning methods that fuel advances in AI achieve desired results at 32-bit and even lower floating-point precision formats. This lesser demand for accuracy fueled a resurgence of interest in new hardware platforms that deliver a mix of unprecedented performance levels and energy savings to achieve the classification and recognition fidelity afforded by higher-accuracy formats.

HPL-MxP strives to unite these two realms by delivering a blend of modern algorithms and contemporary hardware while simultaneously connecting to the solver formulation of the decades-old HPL framework of benchmarking the largest supercomputing installations in the world. The solver method of choice is a combination of LU factorization and iterative refinement performed afterwards to bring the solution back to 64-bit accuracy. The innovation of HPL-MxP lies in dropping the requirement of 64-bit computation throughout the entire solution process and instead opting for low-precision (likely 16-bit) accuracy for LU, and a sophisticated iteration to recover the accuracy lost in factorization. The iterative method guaranteed to be numerically stable is the generalized minimal residual method (GMRES), which uses application of the L and U factors to serve as a preconditioner. The combination of these algorithms is demonstrably sufficient for high accuracy and may be implemented in a way that takes advantage of the current and upcoming devices for accelerating AI workloads.

HPL-MxP Benchmark

- Supercomputers traditionally ranked by performance on high-performance LINPACK (HPL) benchmark
  - Solves dense $Ax = b$ via Gaussian elimination with partial pivoting

- HPL-MxP: Like HPL, solves dense $Ax = b$, results still to double precision accuracy
  - But achieves this via mixed-precision iterative refinement

HPL-MxP Benchmark

November 2022

Rank	Site	Computer	Cores	HPL-AI (Eflop/s)	TOP500 Rank	HPL Rmax (Eflop/s)	Speedup
1	DOE/SC/ORNL	Frontier	8,730,112	7.942	1	1.1020	7.2
2	EuroHPC/CSC	LUMI	2,174,976	2.168	3	0.3091	7.0
3	RIKEN	Fugaku	7,630,848	2.000	2	0.4420	4.5
4	EuroHPC/CINECA	Leonardo	1,463,616	1.842	4	0.1682	11.0
5	DOE/SC/ORNL	Summit	2,414,592	1.411	5	0.1486	9.5
6	NVIDIA	Selene	555,520	0.630	9	0.0630	9.9
7	DOE/SC/LBNL	Perlmutter	761,856	0.590	8	0.0709	8.3
8	FZJ	JUWELS BM	449,280	0.470	12	0.0440	10.0
9	GENCI-CINES	Adastra	319,072	0.303	11	0.0461	6.6
10	Pawsey Supercomputing Centre	Setonix - GPU	181,248	0.175	15	0.0272	6.4

HPL-AI Benchmark

November 2022

Rank	Site	Computer	Cores	HPL-AI (Eflop/s)	TOP500 Rank	HPL Rmax (Eflop/s)	Speedup
1	DOE/SC/ORNL	Frontier	8,730,112	7.942	1	1.1020	7.2
2	EuroHPC/CSC	LUMI	2,174,976	2.168	3	0.3091	7.0
3	RIKEN	Fugaku	7,630,848	2.000	2	0.4420	4.5
4	EuroHPC/CINECA	Leonardo	1,463,616	1.842	4	0.1682	11.0
5	DOE/SC/ORNL	Summit	2,414,592	1.411	5	0.1486	9.5
6	NVIDIA	Selene	555,520	0.630	9	0.0630	9.9
7	DOE/SC/LBNL	Perlmutter	761,856	0.590	8	0.0709	8.3
8	FZJ	JUWELS BM	449,280	0.470	12	0.0440	10.0
9	GENCI-CINES	Adastra	319,072	0.303	11	0.0461	6.6
10	Pawsey Supercomputing Centre	Setonix - GPU	181,248	0.175	15	0.0272	6.4

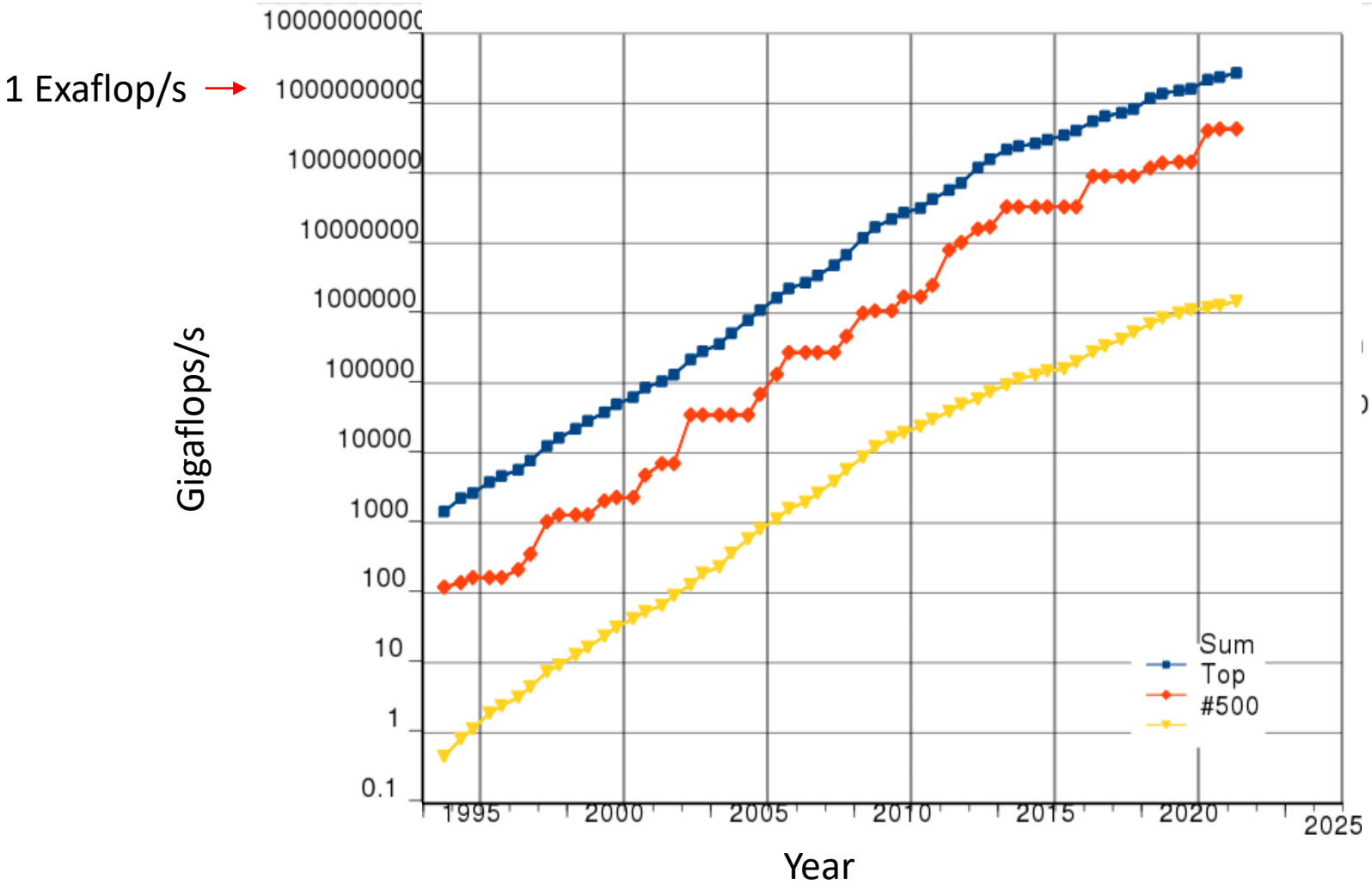
HPL-AI Benchmark

November 2022

Rank	Site	Computer	Cores	HPL-AI (Eflop/s)	TOP500 Rank	HPL Rmax (Eflop/s)	Speedup
1	DOE/SC/ORNL	Frontier	8,730,112	7.942	1	1.1020	7.2
2	EuroHPC/CSC	LUMI	2,174,976	2.168	3	0.3091	7.0
3	RIKEN	Fugaku	7,630,848	2.000	2	0.4420	4.5
4	EuroHPC/CINECA	Leonardo	1,463,616	1.842	4	0.1682	11.0
5	DOE/SC/ORNL	Summit	2,414,592	1.411	5	0.1486	9.5
6	NVIDIA	Selene	555,520	0.630	9	0.0630	9.9
7	DOE/SC/LBNL	Perlmutter	761,856	0.590	8	0.0709	8.3
8	FZJ	JUWELS BM	449,280	0.470	12	0.0440	10.0
9	GENCI-CINES	Adastra	319,072	0.303	11	0.0461	6.6
10	Pawsey Supercomputing Centre	Setonix - GPU	181,248	0.175	15	0.0272	6.4

Performance Development

Exascale Achieved in June 2022



Distribution of supercomputers in the TOP500 list by country (as of November 2022)[36]

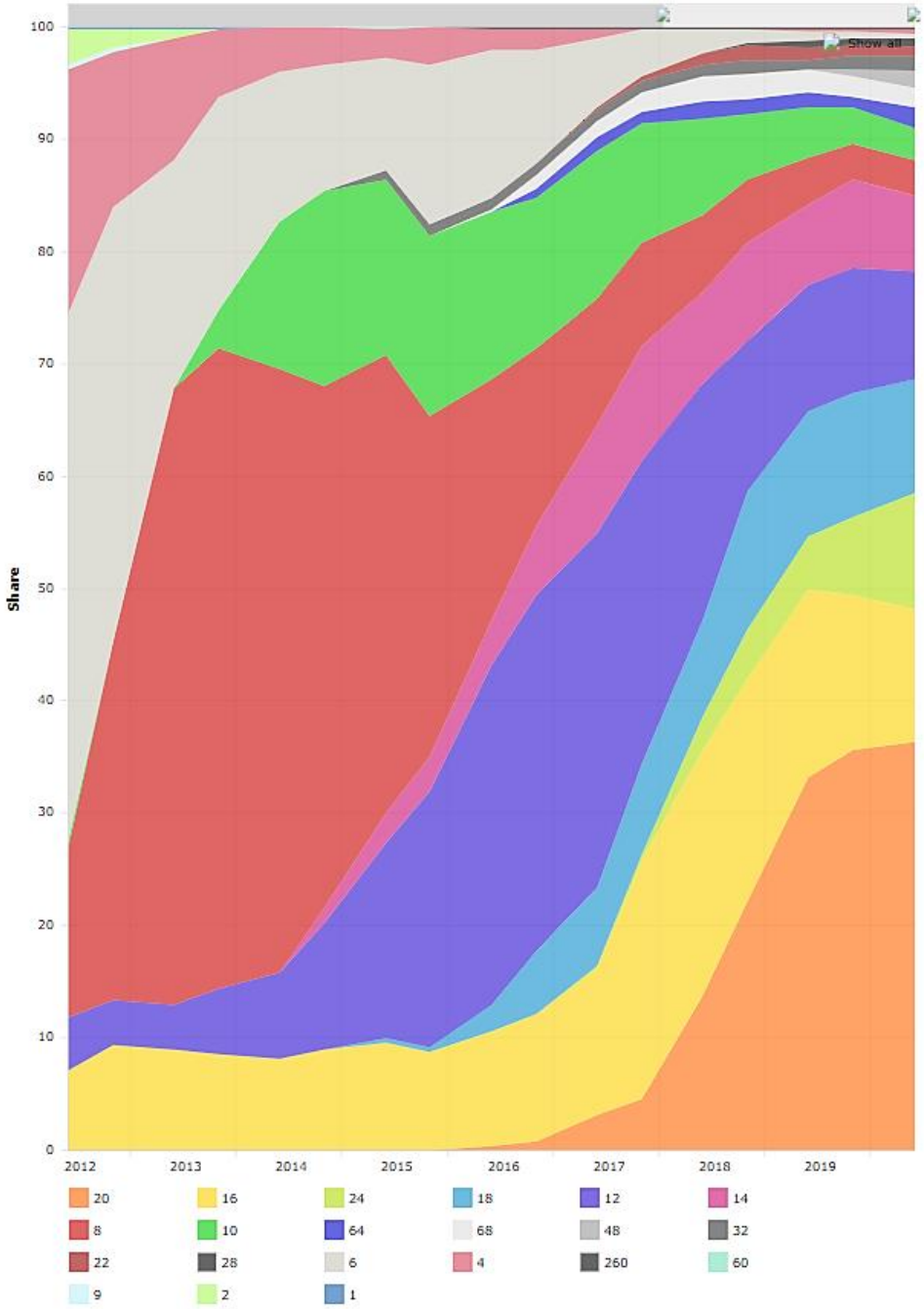
Country or Territory	Systems
China	162
United States	127
Germany	34
Japan	31
France	24
United Kingdom	15
Canada	10
South Korea	8
Brazil	8
Netherlands	8
Italy	7
Russia	7
Saudi Arabia	6
Sweden	6
Australia	5
Ireland	5
Switzerland	4
Singapore	3
Norway	3
India	3
Finland	3
Poland	3
Taiwan	2
Czechia	2
Luxembourg	2
United Arab Emirates	2
Austria	2
Slovenia	2
Spain	1

EuroHPC

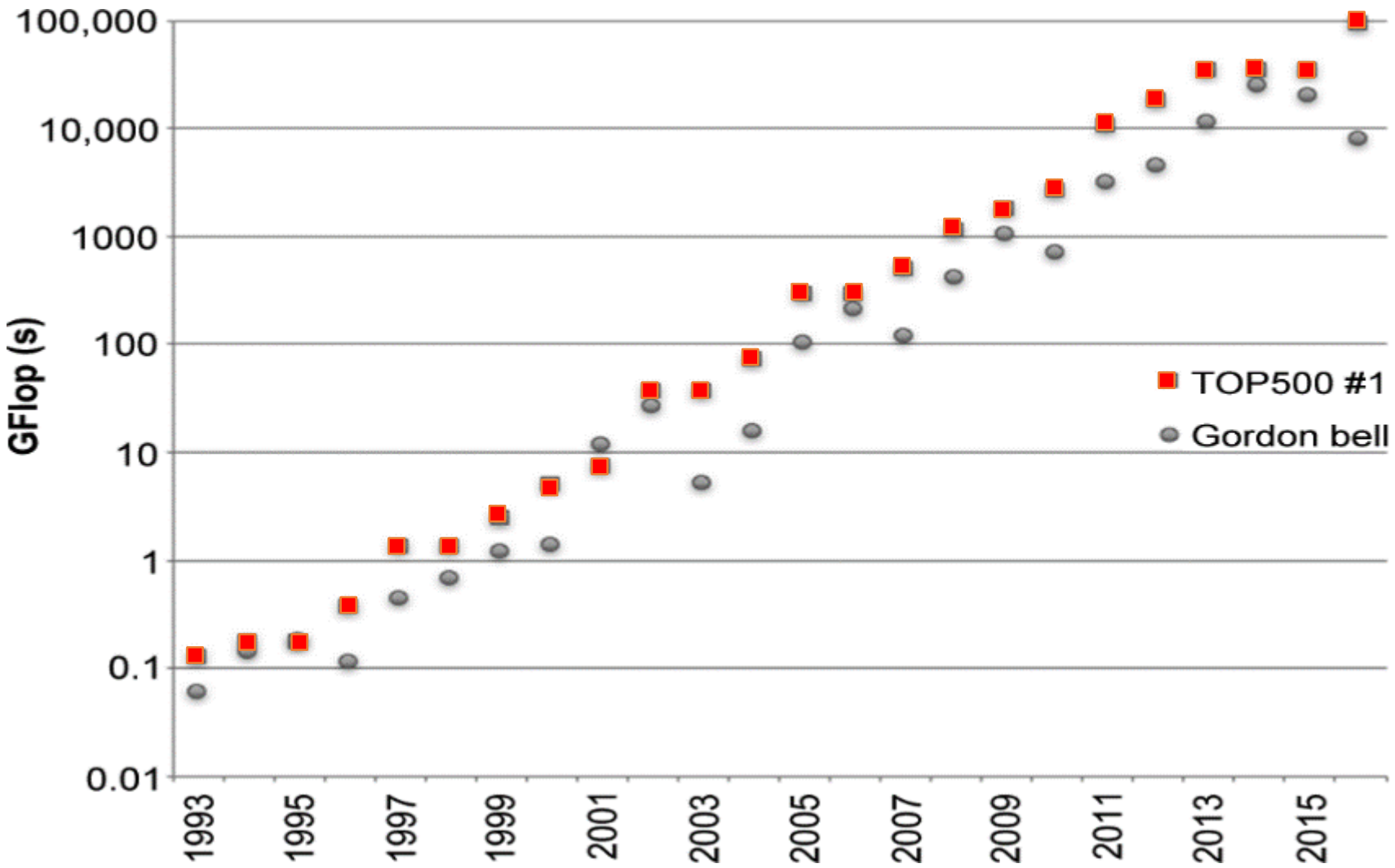
- EU Declaration on HPC in 2017; project formalized in 2018
- 3 Billion EUR project
- Aims to pool European resources to develop top-of-the-range exascale supercomputers, developing a pan-European supercomputing infrastructure
- Providing a world-class **petascale and pre-exascale** supercomputing and data infrastructure for Europe's scientific, industrial and public users, matching their demanding application requirements **in 2020**
- Acquisition in **2023 of two exascale systems**, one post-exascale system, networking and coordination of HPC Competence Centres, support for the first hybrid HPC / Quantum computing infrastructure in Europe
- https://eurohpc-ju.europa.eu/index.html

Ostrava, ČR is the site of one new machine

Cores per Socket - Systems Share



Science: Gordon Bell Prizes vs Top 500



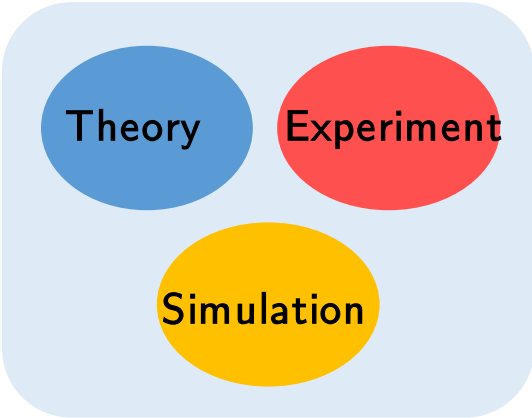
Science using Supercomputers

The "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 or galactic evolution
- Too dangerous—weapons, drug design, climate experimentation

Computational science and engineering 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 manually

Some Particularly Challenging Computations

- Science
  - Global climate modeling
  - Biology: genomics; protein folding; drug design
  - Astrophysical modeling
  - Computational Chemistry
  - Computational Material Sciences and Nanosciences
- Engineering
  - Semiconductor design
  - Earthquake and structural modeling
  - Computation fluid dynamics (airplane design)
  - Combustion (engine design)
  - Crash simulation
- Business
  - Financial and economic modeling
  - Transaction processing, web services and search engines
- Defense
  - Nuclear weapons -- test by simulations
  - Cryptography

Global Climate Modeling Problem

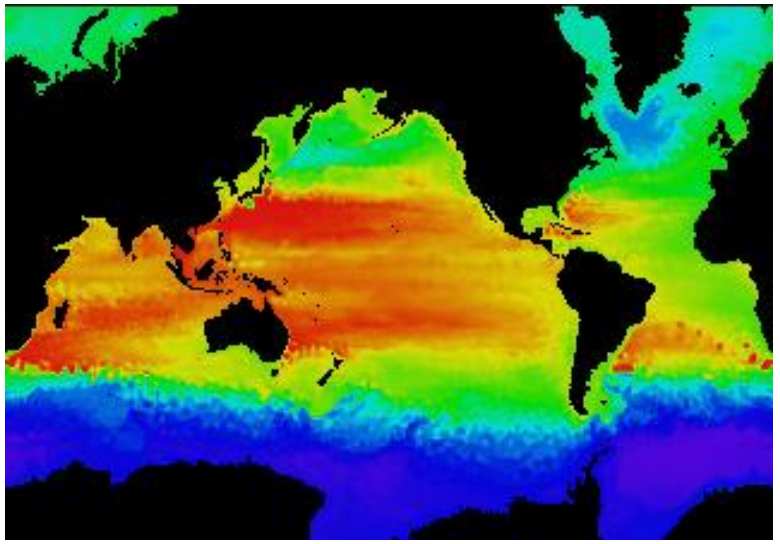
- Problem is to compute:

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

  (temperature, pressure, humidity, wind velocity)

- Approach:
  - *Discretize* the domain, e.g., a measurement point every 10 km
  - Devise an algorithm to predict weather at time t+δt given t

- Uses:
  - Predict major events, e.g., El Nino
  - Use in setting air emissions standards
  - Evaluate global warming scenarios



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

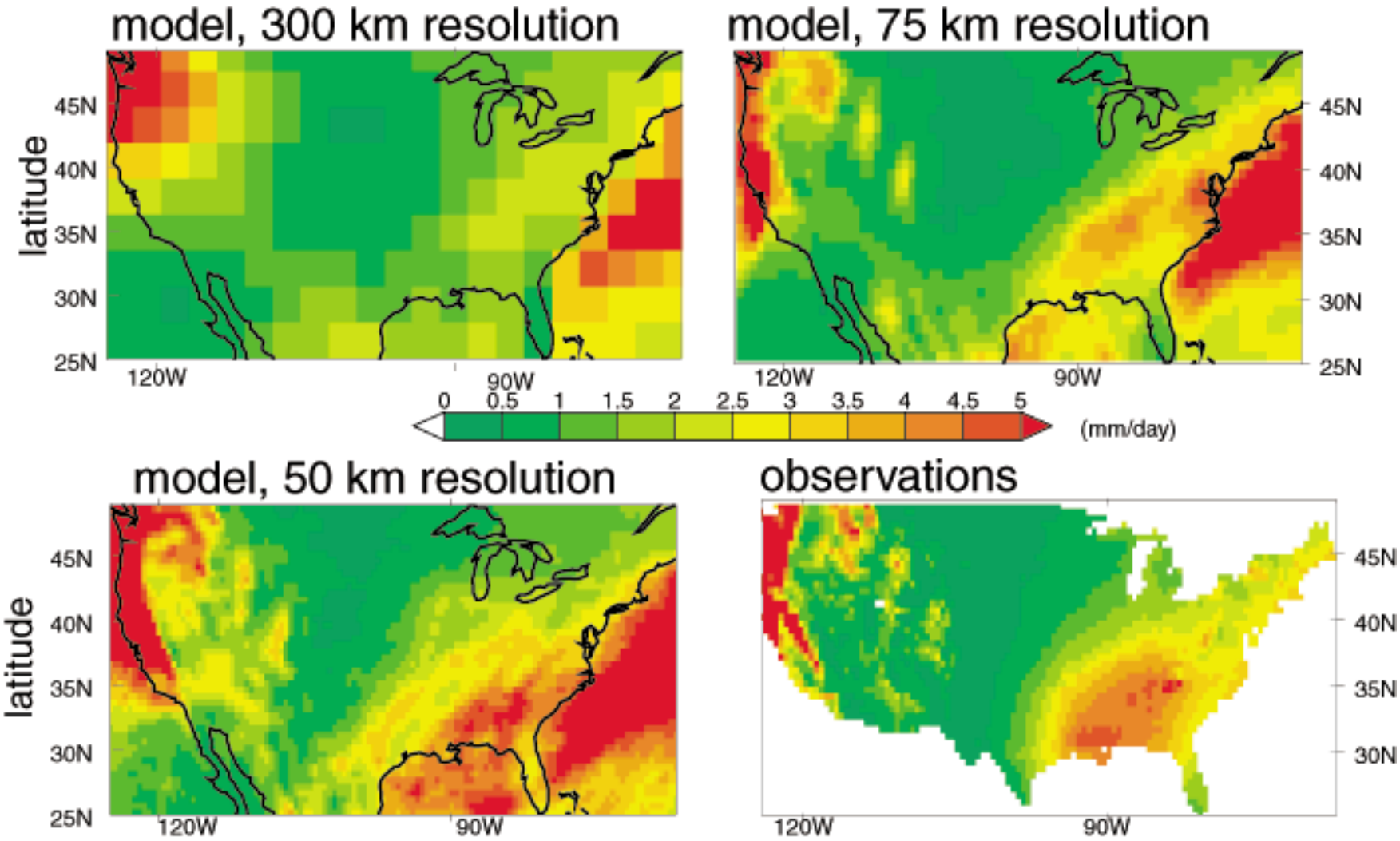
Global Climate Modeling Computation

- One piece is modeling the fluid flow in the atmosphere
  - Solve Navier-Stokes equations
    - Roughly 100 Flops per grid point with 1 minute timestep
- Computational requirements:
  - 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
- To double the grid resolution, computation is 8x to 16x
- State of the art models require integration of atmosphere, clouds, ocean, sea-ice, land models, plus possibly carbon cycle, geochemistry and more
- Current models are coarser than this

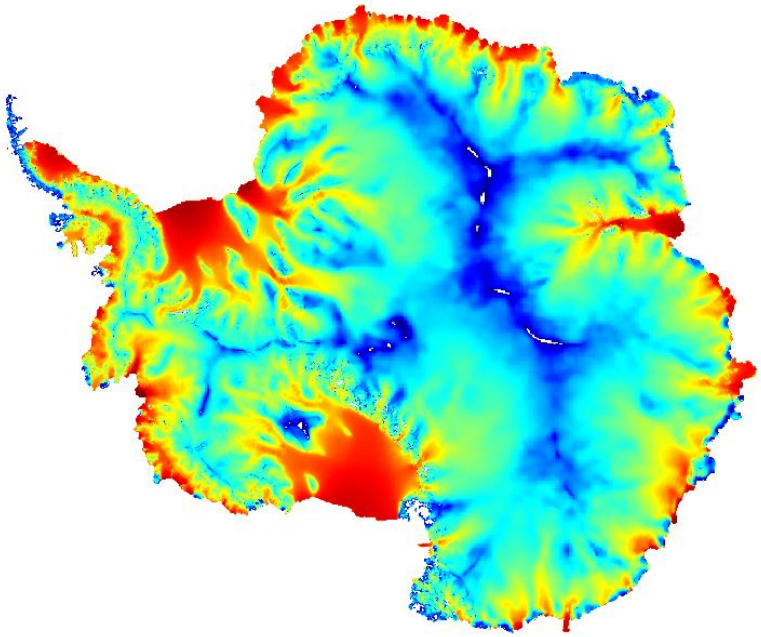
High Resolution Climate Modeling on NERSC-3 – P. Duffy, et al., LLNL

Wintertime Precipitation (millimeters/day)

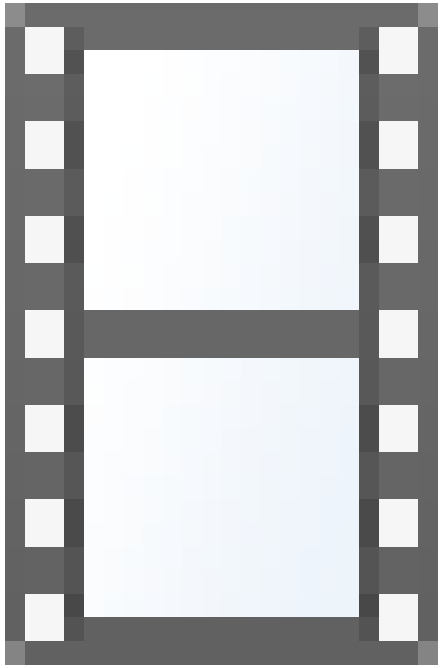
As model resolution becomes finer, results converge towards observations



Extreme Scale Climate Science



Melting of West Antarctic Ice Sheet using Adaptive Mesh Refinement (AMR)
Dan Martin, LBNL (BISICLES/E3SM)

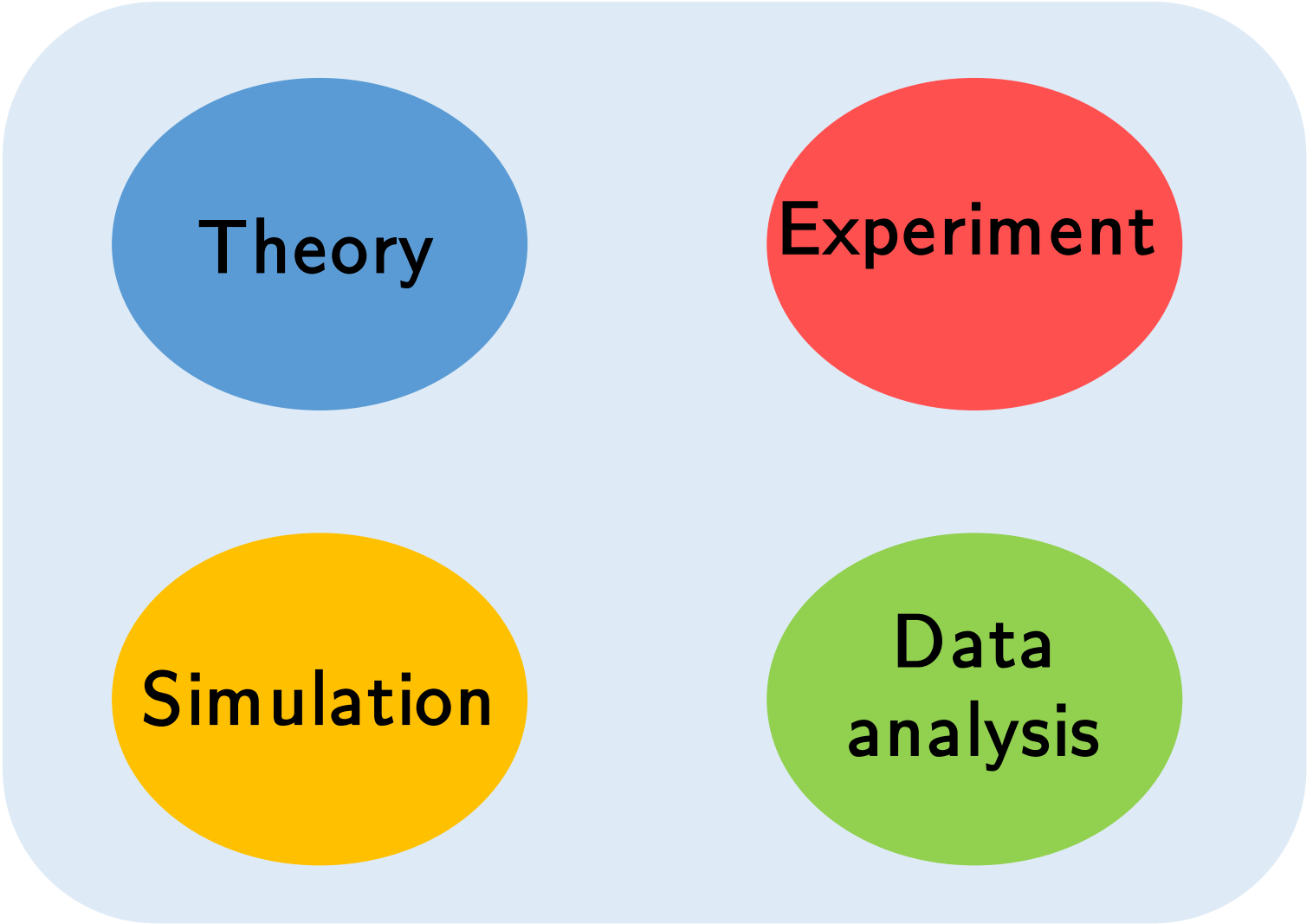


Mathematical model for clouds
David Romps et al, UCB

Exascale is needed to simulate climate and analyze impacts

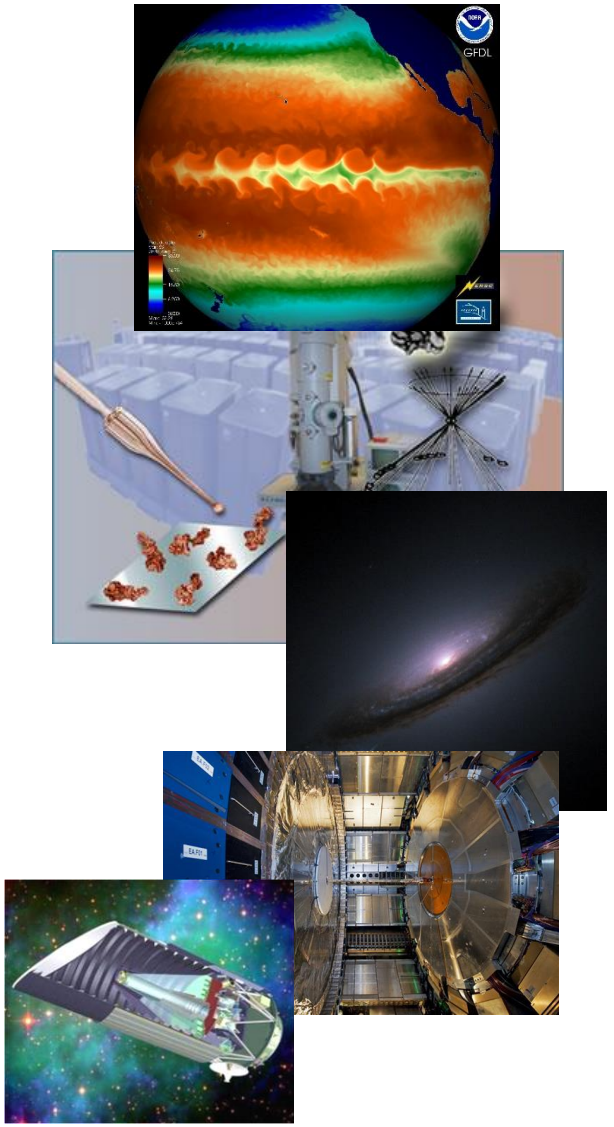
Resolve clouds, predict sea level rise, quantify extreme events and model precipitation and ground water levels

The "Fourth Paradigm" of Science



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 will soon be common:
  - *Climate modeling:* estimates of the next IPCC data is in 10s of petabytes
  - *Genome:* JGI alone will have .5 petabyte of data this year and double each year
  - *Particle physics*: LHC is projected to produce 16 petabytes of data per year
  - *Astrophysics*: LSST and others will produce 5 petabytes/year (via 3.2 Gigapixel camera)
- Create scientific communities with "Science Gateways" to data



Data analytics in science and engineering

High Performance Data Analytics (HPDA) is used for data sets that are:

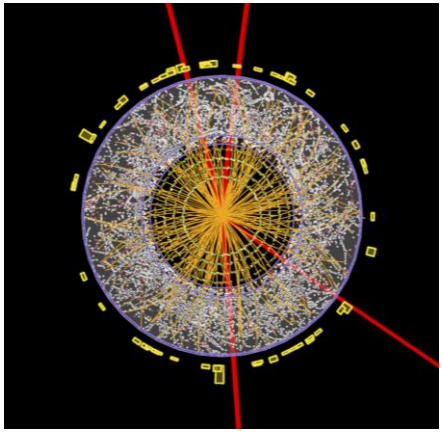
- too big
- too complex
- too fast (streaming)
- too noisy
- too heterogeneous

for measurement alone

Images from telescopes

Wayne Joubert, Dan Jacobson et al, Gordon Bell Prize (1 of 2) at SC18

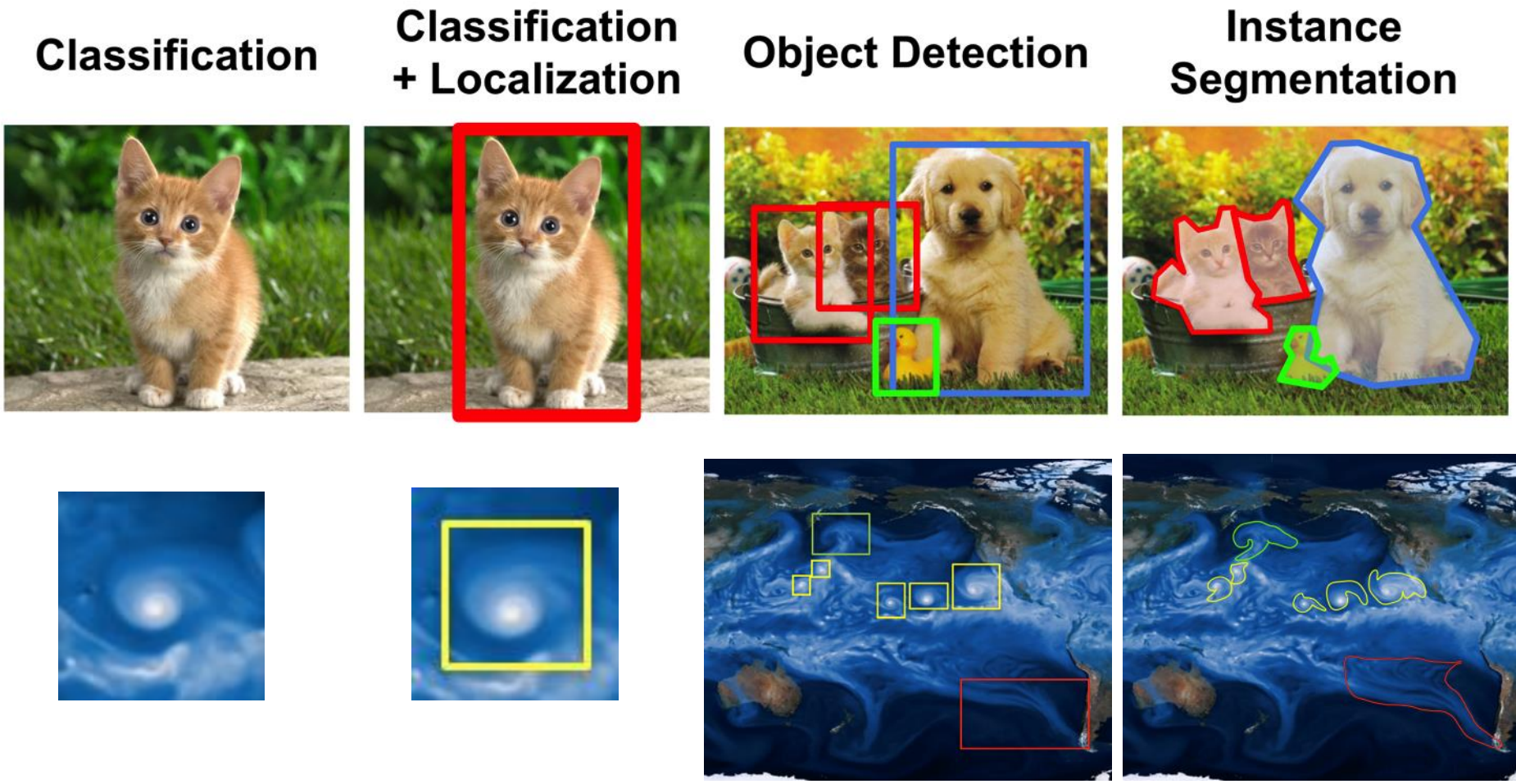
Genomes from sequencers



Particle from detectors

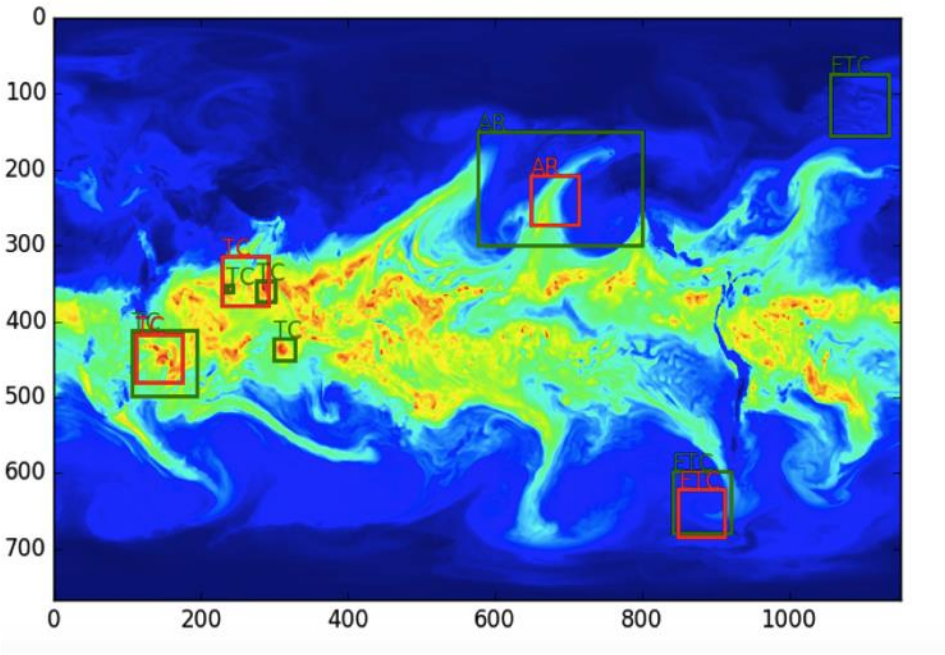
Sensor data

Machine Learning in Climate Data

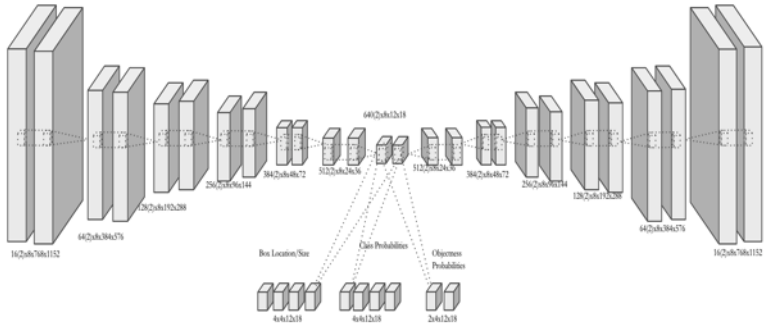


Contributors: Prabhat, Thorsten Kurth, Jian Yang, Ioannis Mitliagkas, Chris Pal, Nadathur Satish, Narayanan Sundaram, Amir Khosrowshahi, Michael Wehner, Bill Collins.

Deep Learning for Extreme Weather Events



Ground Truth vs Prediction



Use of deep learning (CNNs)

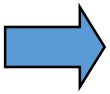
- Supervised and semi-supervised learning on CAM5 data
- 85-99% accuracy at identifying extreme climate events
- 1 ExaOp (16-bit) on Summit at ORNL; trained in 100 minutes

Thorsten Kurth et al

Why writing (fast) parallel programs is hard

Parallel Programming Challenges

- Finding enough parallelism (Amdahl's Law)
- Granularity – how big should each parallel task be
- Locality – moving data costs more than arithmetic
- Load balance – don't want 1K processors to wait for one slow one
- Coordination and synchronization – sharing data safely
- Performance modeling/debugging/tuning



All of these things makes parallel programming even harder than sequential programming.

"Automatic" Parallelism in Modern Machines

- Bit level parallelism
  - within floating point operations, etc.
- Instruction level parallelism (ILP)
  - multiple instructions execute per clock cycle
- Memory system parallelism
  - overlap of memory operations with computation
- OS parallelism
  - multiple jobs run in parallel on commodity SMPs

Limits to all of these -- for very high performance, need user to identify, schedule and coordinate parallel tasks

Finding Enough Parallelism: Amdahl's Law

- Suppose only part of an application is parallel
- Amdahl's law
  - **s** = fraction of work done sequentially (Amdahl fraction), so (1-s) is fraction parallelizable
  - **P** = number of processors

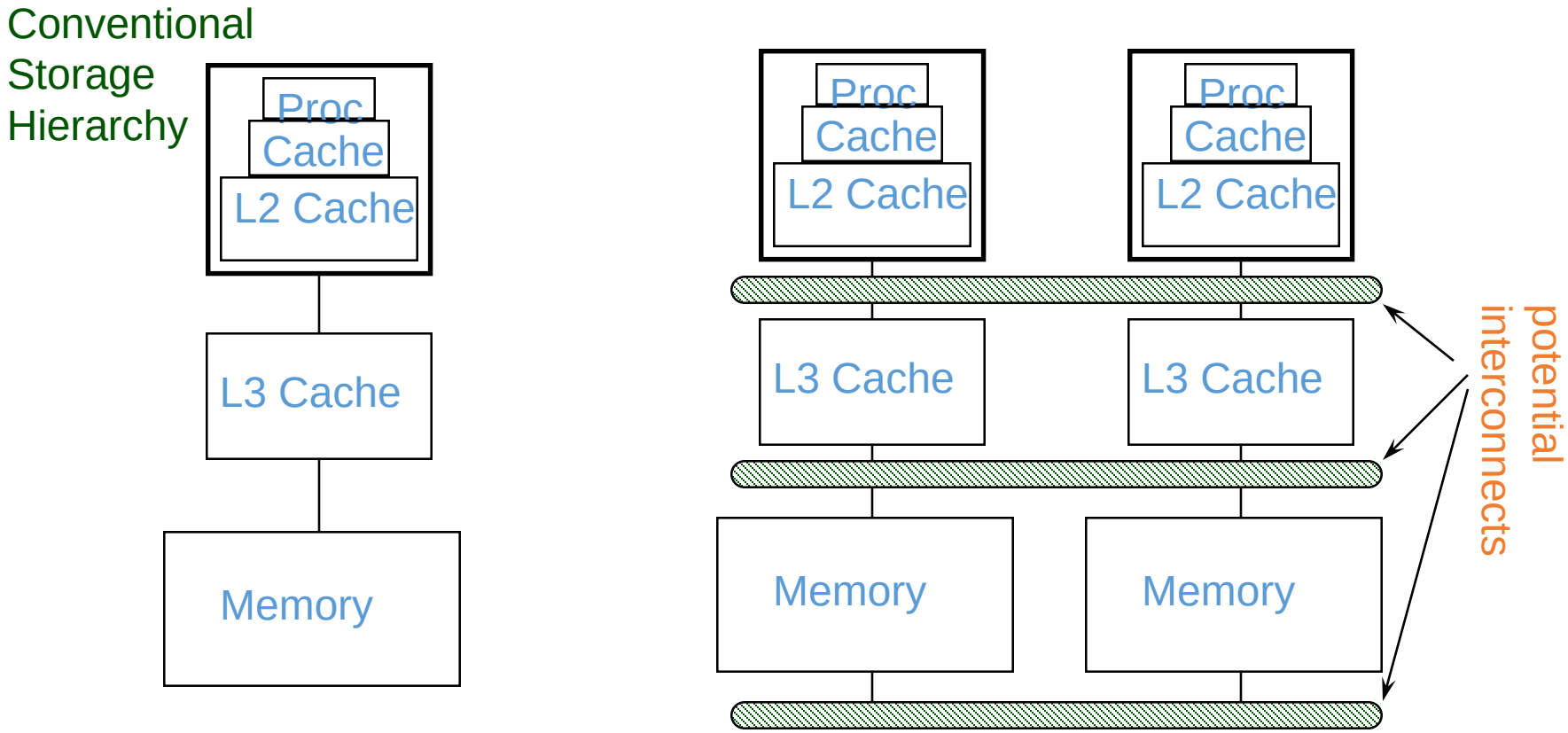
$$\begin{aligned} \text{Speedup}(P) &= \text{Time}(1)/\text{Time}(P) \\ &<= 1/(s + (1\text{-}s)/P) \\ &<= 1/s \end{aligned}$$

- Even if the parallel part speeds up perfectly performance is limited by the sequential part
- $1/10^{th}$ of your code's runtime is serial → max speedup is 10x

Overhead of Parallelism

- Given enough parallel work, this is the biggest barrier to getting desired speedup
- Parallelism overheads include:
  - cost of creating parallelism (starting a thread/process)
  - cost of communicating shared data
  - cost of synchronizing
  - extra (redundant) computation
- Overheads can be in milliseconds (which is millions of flops)
- Tradeoff:
  - Algorithm needs sufficiently large units of work to run fast in parallel (i.e. large granularity),
  - But not so large that there is not enough parallel work

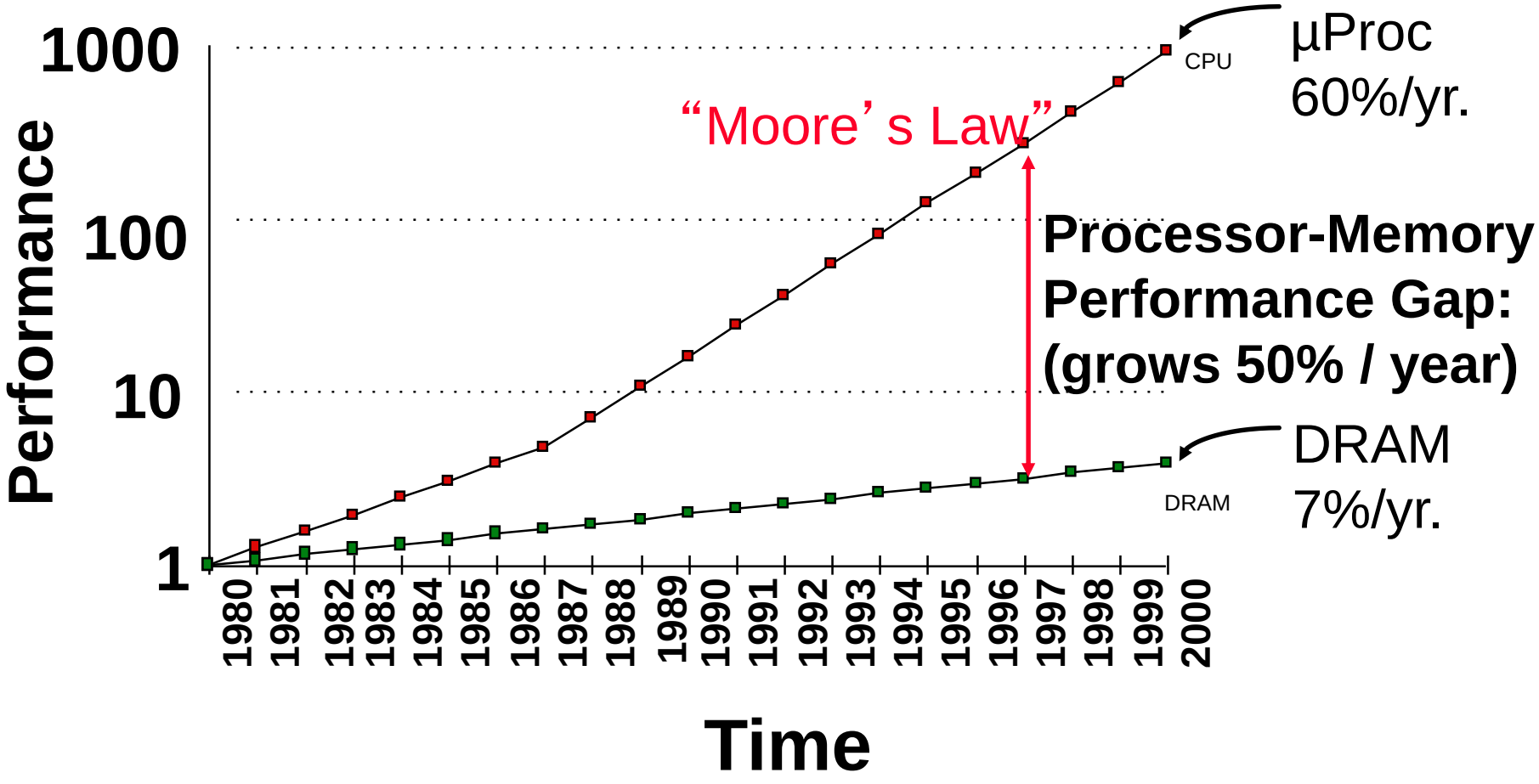
Locality and Parallelism



- Large memories are slow, fast memories are small
- Storage hierarchies are large and fast on average
- Parallel processors, collectively, have large, fast cache
  - the slow accesses to "remote" data we call "communication"
- Algorithm should do most work on local data

Processor-DRAM Gap (latency)

Goal: find algorithms that minimize communication, not necessarily arithmetic



Load Imbalance

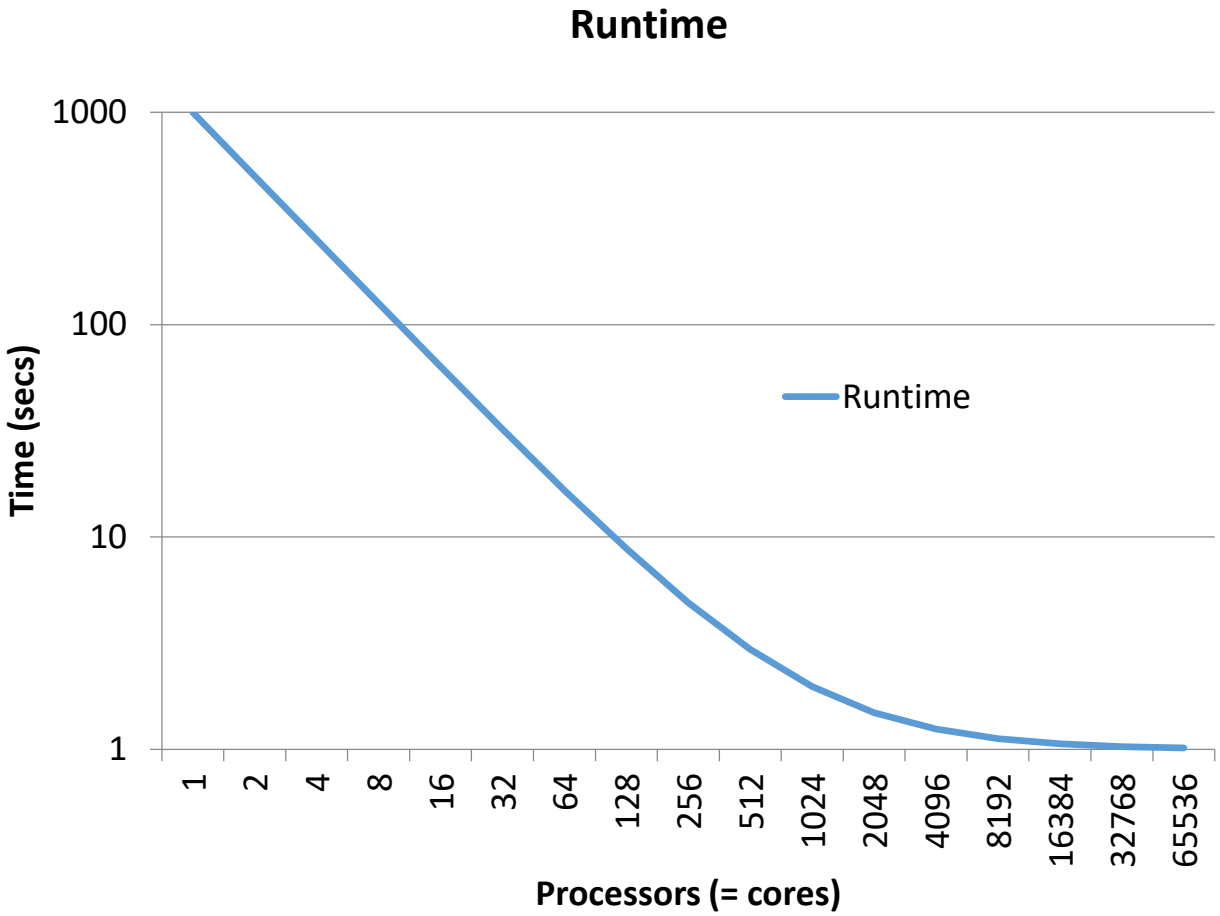
- Load imbalance is the time that some processors in the system are idle due to
  - insufficient parallelism (during that phase)
  - unequal size tasks
- Examples of the latter
  - adapting to “interesting parts of a domain”
  - tree-structured computations
  - fundamentally unstructured problems
- Algorithm needs to balance load
  - Sometimes can determine work load, divide up evenly, before starting
    - "Static Load Balancing"
  - Sometimes work load changes dynamically, need to rebalance dynamically
    - "Dynamic Load Balancing", e.g. work-stealing

Parallel Software ... Eventually

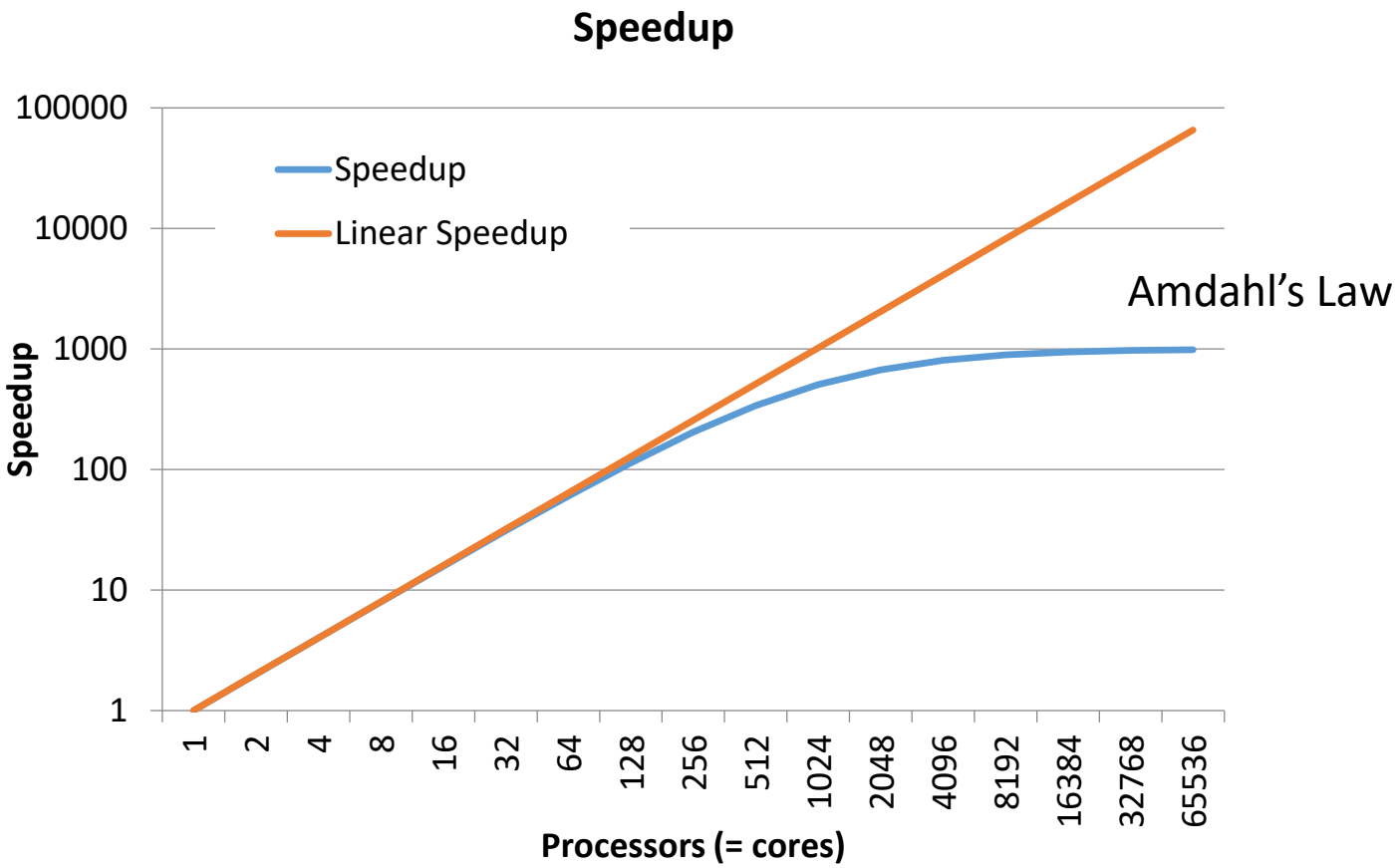
- 2 types of programmers ⇒ 2 layers of software
- **Efficiency Layer** (10% of programmers)
  - Expert programmers build Libraries implementing kernels, "Frameworks", OS, ...
  - Highest fraction of peak performance possible
- **Productivity Layer** (90% of programmers)
  - Domain experts / Non-expert programmers productively build parallel applications by composing frameworks & libraries
  - Hide as many details of machine, parallelism as possible
  - Willing to sacrifice some performance for productive programming
- I expect that you may want to work at either level...
  - In the meantime, we all need to understand enough of the efficiency layer to use parallelism effectively

Measuring Performance

Goal of Parallelism: Decrease Running Time



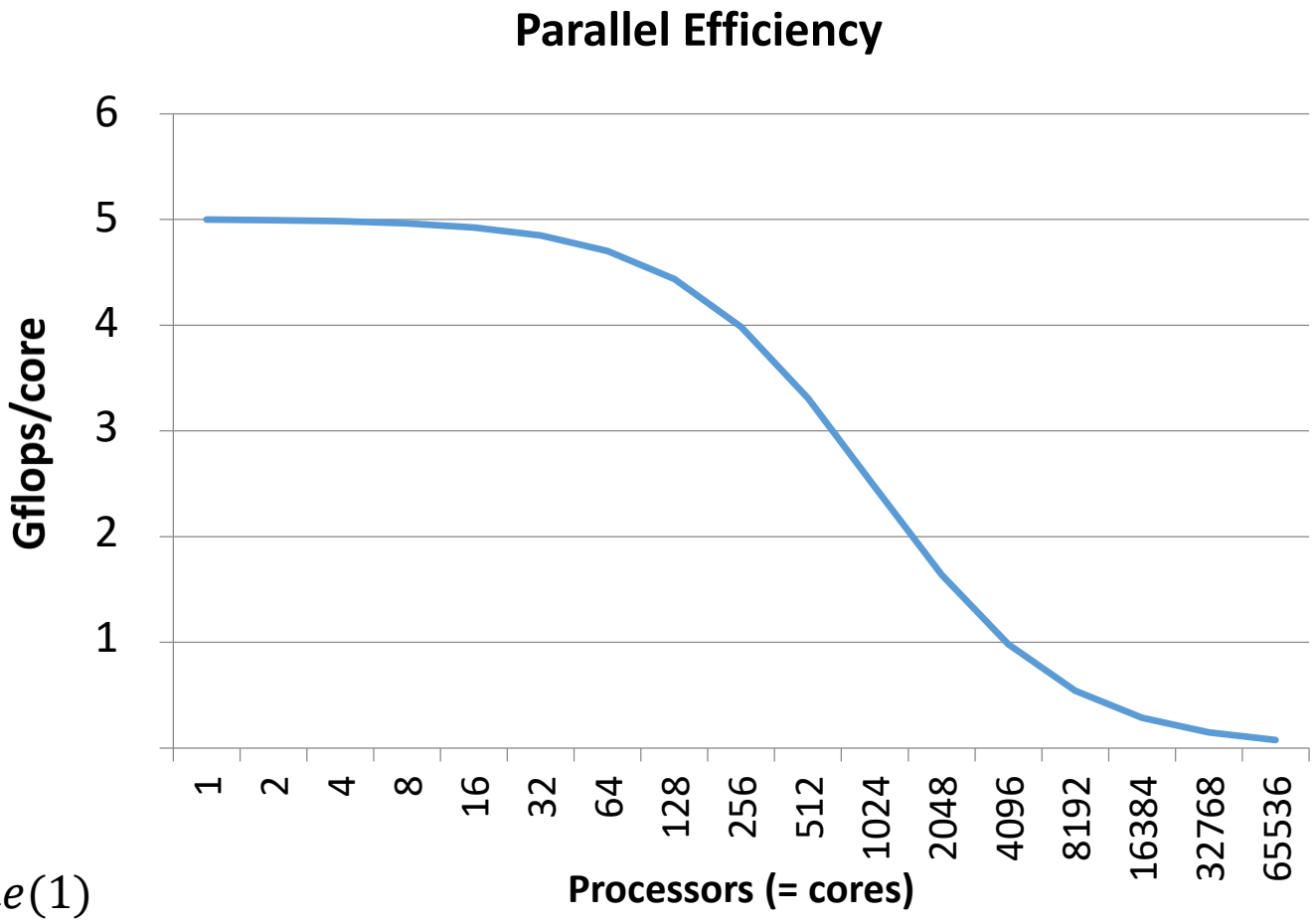
Reporting Speedup (Strong Scaling)



Speedup(P) = Time(1)/Time(P)

This is a **strong scaling** plot: fixed problem size, vary number of processors

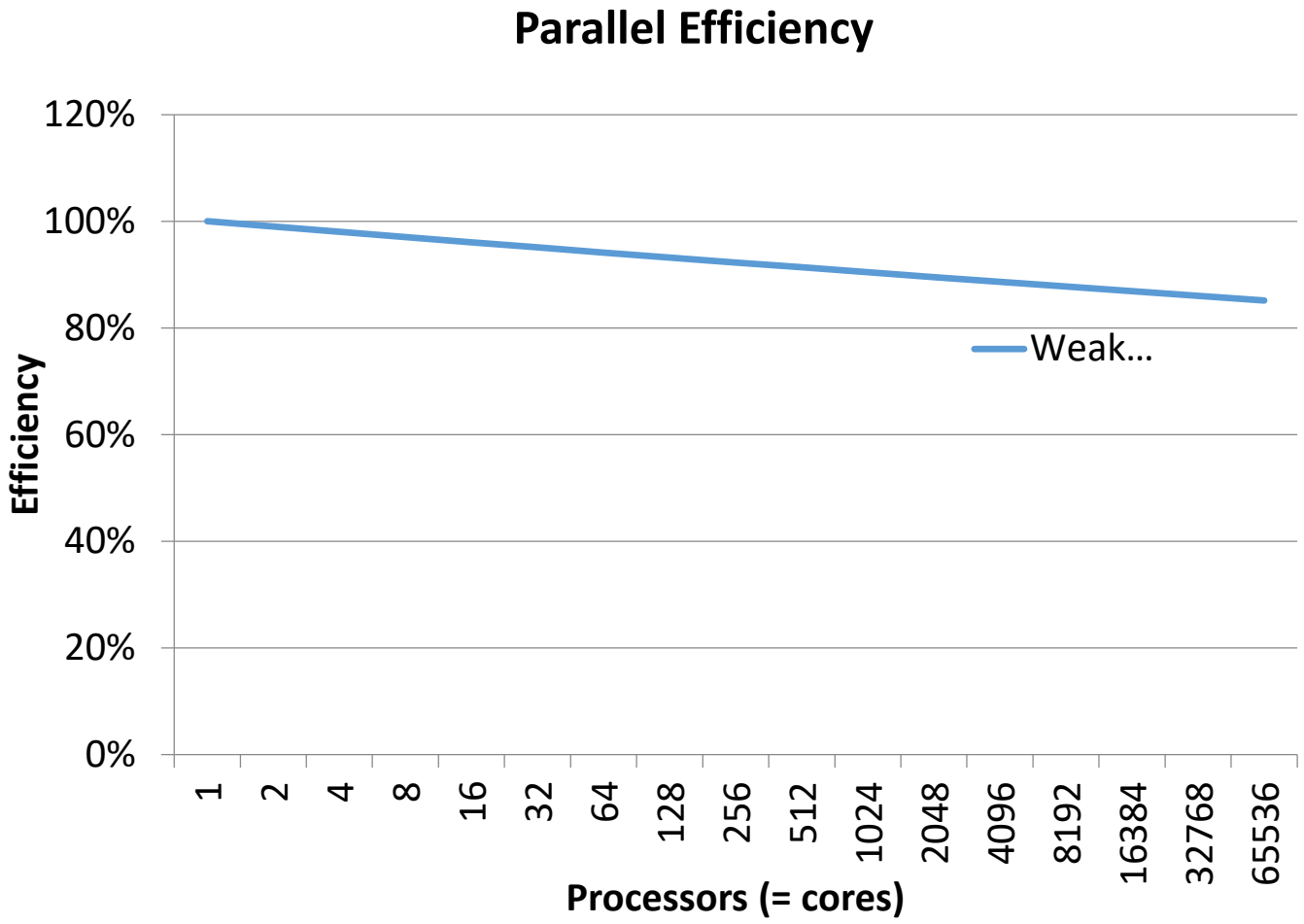
Parallel Efficiency



$$E = \frac{Time(1)}{Time(P) \times P}$$

Same **strong scaling** results shown as efficiency (ideally flat)

Parallel Efficiency (Weak Scaling)



Weak scaling uses a fixed problem size **per processor.** Can report as:

- Flop/s (or other rate) per processor; Efficiency based on rate per processor
- Time (if algorithm is linear in data size)

Summary: Strong Scaling versus Weak Scaling

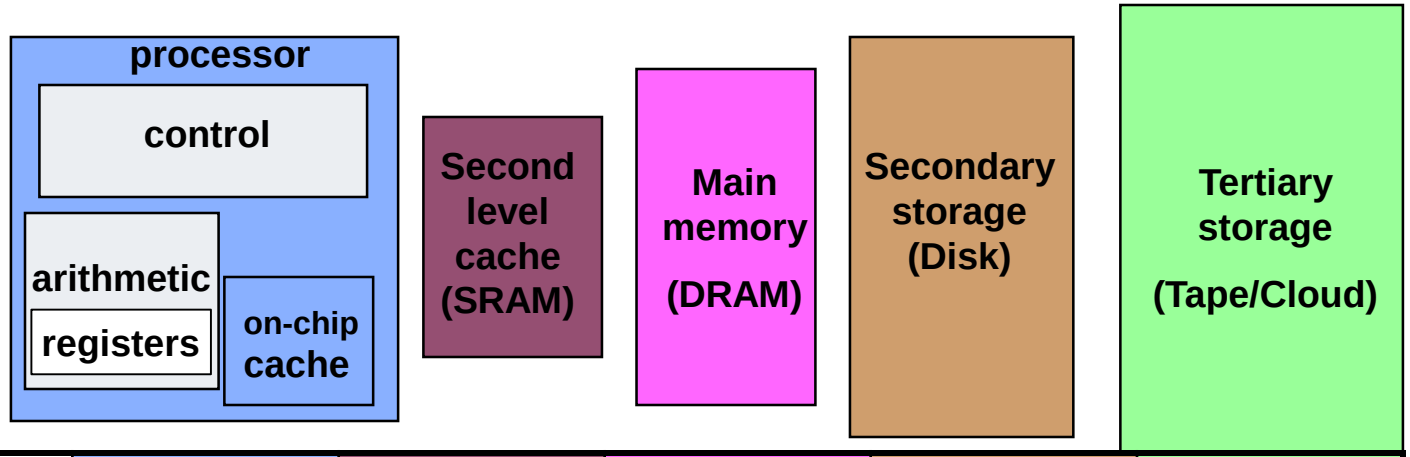
- Strong scaling: concerns the speedup for a **fixed problem size** with respect to the number of processors (fixed total problem size)
  - governed by Amdahl's law

- Weak scaling: concerns the speedup for a **scaled problem size** with respect to the number of processors (fixed problem size per processor)
  - speedup depends on amount of serial work remaining constant or increasing slowly as the size of the problem grows
  - also depends on how amount of communication between processors is grows

Limits to Performance Start at Home

(Single Processor Memory Hierarchy)

- Large memories are slow; fast memories are small
- Most programs have a things nearby previous accesses
  - **temporal locality:** reusing an item that was previously accessed
- Memory hierarchy use this to improve *average case*



Speed	1ns	10ns	100ns	10ms	10sec
Size	KB	MB	GB	TB	PB

First Assignment: Create Cluster Account

- Go to http://cluster.karlin.mff.cuni.cz/jak-se-stat-uzivatelem/
- Follow instructions there

Jak se stát uživatelem

Je to vlastně snadné...

Napište mailem na `clusteradmin@karlin.mff.cuni.cz` a uveďte následující položky:

- Jméno a příjmení:.....
- Login:..... (*pokud již máte účet na pracovních stanicích v Karlíně [hill.karlin.mff.cuni.cz] uveďte tento*)
- e-mail:.....
- Doporučující osoba:.....
- Požadavky na Software:.....

Pokud nejste zaměstnanec MFF z matematické sekce, uveďte jméno doporučující osoby (vedoucí diplomové práce, školitel, spolupracovník z MFF z mat. sekce, se kterým v dané oblasti spolupracujete)