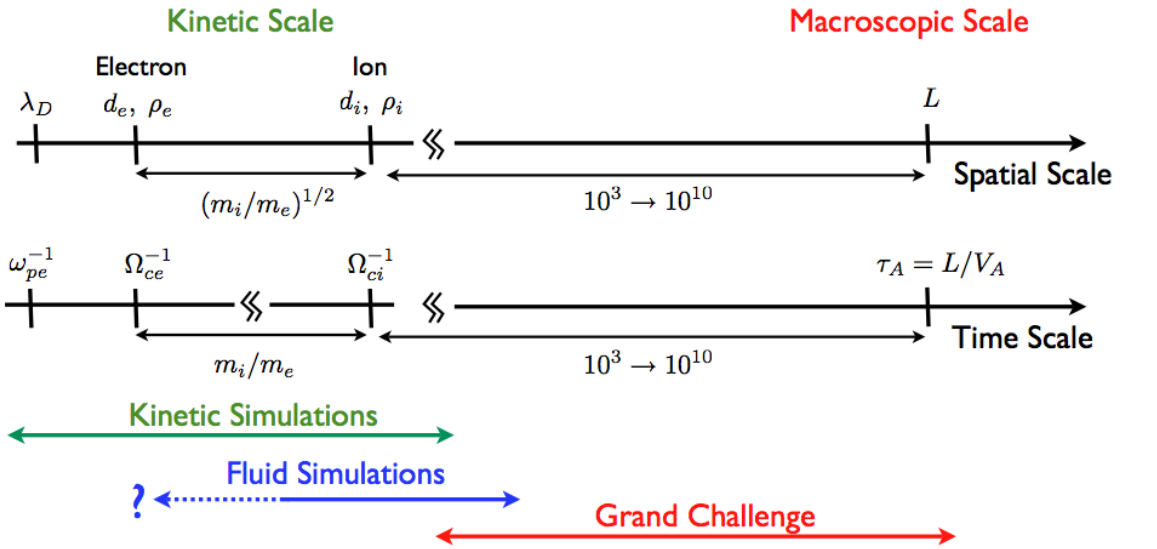
Plasma physics is truly multiscale!



Parameters/size	physics	Cost, CPU-hrs
2D: M/m=100; Ω_{pe}/Ω_{ce}=2 1024 x 2048 cells	Basic physics of the diffusion region, role of the external drive	1.5 x 10^4
2D: M/m=1836; Ω_{pe}/Ω_{ce}=2 5300 x 10000 cells	Realistic influence of binary collisions, realistic kinetic physics of trapping, etc	6.5 x 10^6
3D: M/m=300; Ω_{pe}/Ω_{ce}=2 2000 x 4000x4000 cells	Influence of current-aligned instabilities	9 x 10^8
3D: M/m=1836; Ω_{pe}/Ω_{ce}=80 $(2x10^5)$ x $(4x10^5)$ x $(4x10^5)$ cells	Realistic physical parameters	9 x 10^{18}

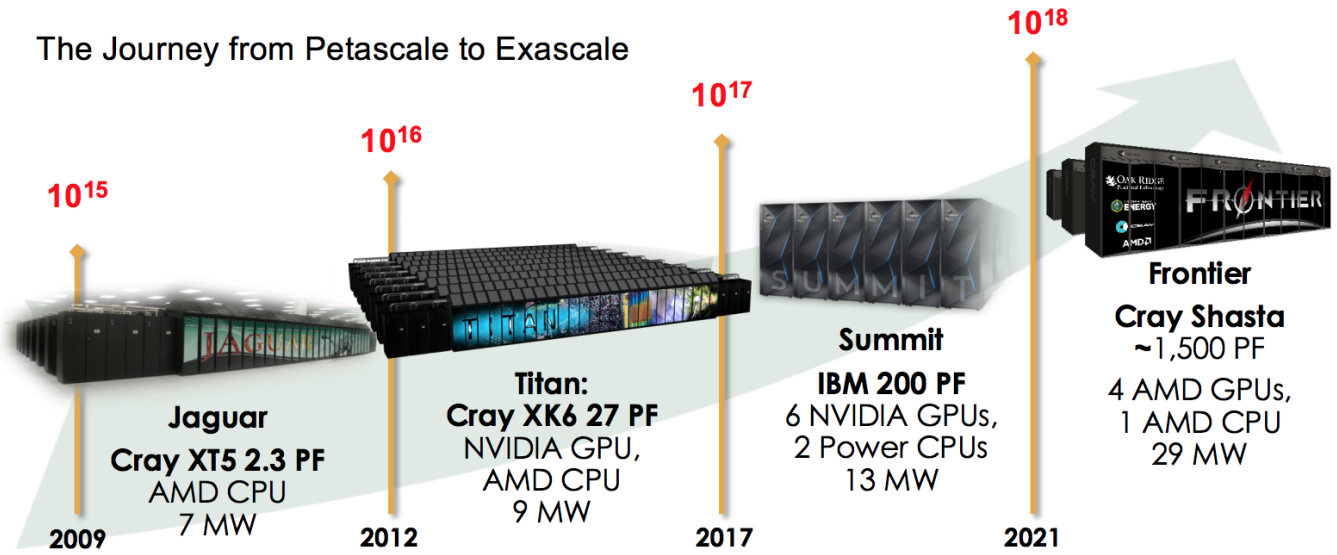
Computers then and now

Cray X-MP/48 (1986), 800 MFlops

Road to Exascale

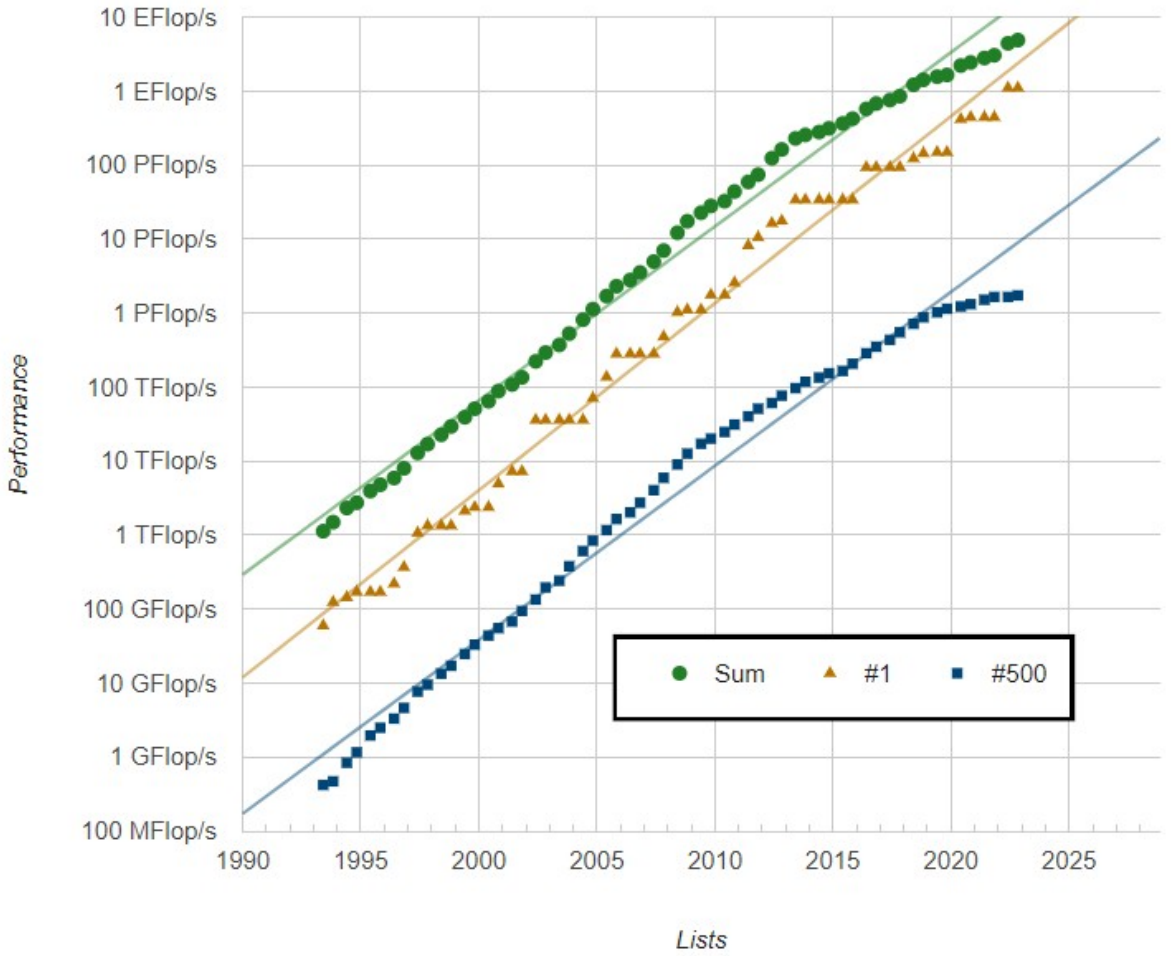
Oak Ridge Leadership Computing Facility
Roadmap to Exascale

Mission: Providing world-class computational resources and specialized services for the most computationally intensive global challenges for researchers around the world.

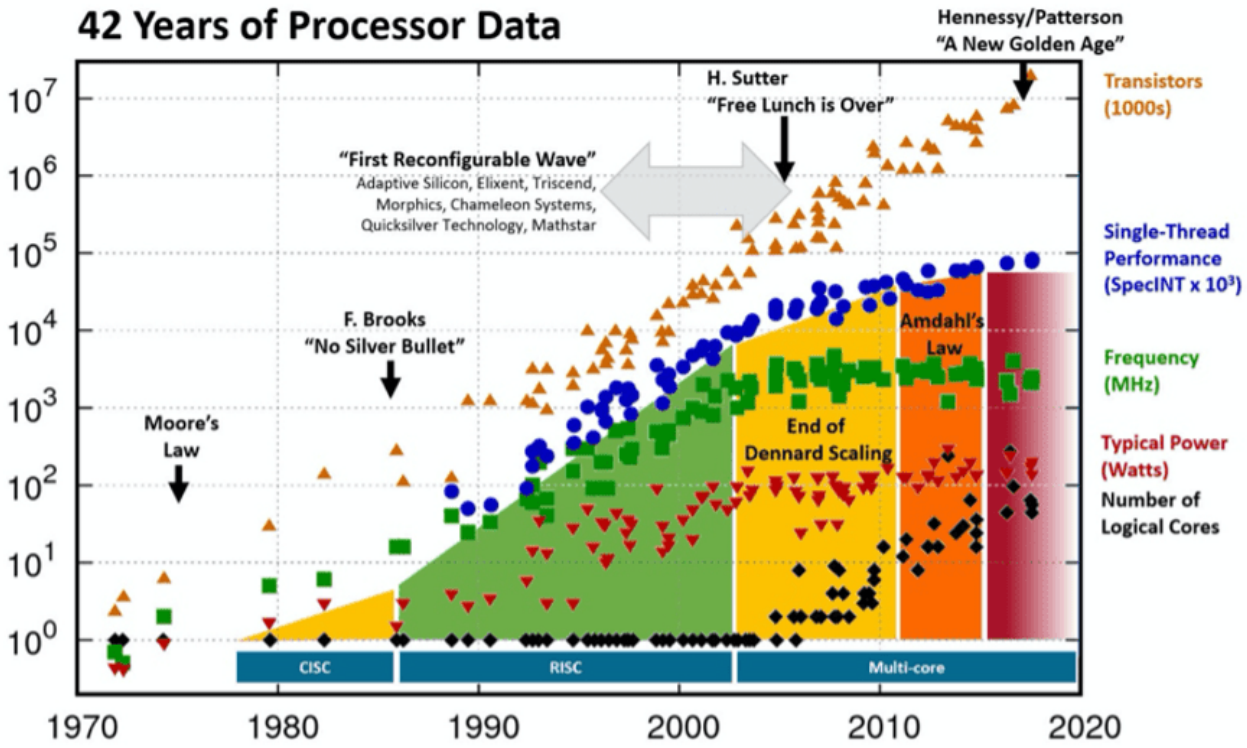


(from OLCF)

TOP 500 list of supercomputers



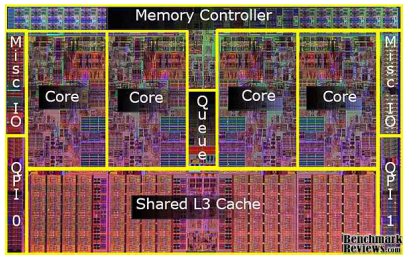
Processor Performance



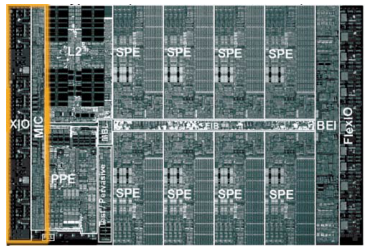
Hennessy and Patterson, Turing Lecture 2018, overlaid over "42 Years of Processors Data"
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/;"First Wave" added by Les Wilson, Frank Schirrmeister
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

https://semiengineering.com/chip-design-shifts-as-fundamental-laws-run-out-of-steam/

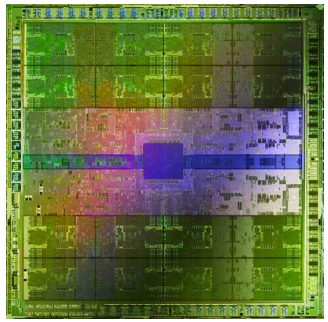
Heterogeneous computing



multi-core



Cell



GPU

	multi-core	Cell	GPU
OS runs on	any core	PPU	host CPU
main code on	any core	PPU	host CPU
comput. kernels on	any core	SPUs	GPU MP
memory architecture	main memory caches	main memory per-SPU local store	main memory GPU memory shared memory
data movement	transparent	DMA for moving data	explicit
SIMD	explicit	explicit	automatic
threads	few	no (?)	many
efficient programming	hard	hard	hard

Heterogeneous computing

Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers
David H. Bailey
June 11, 1991
Ref: Supercomputing Review, Aug. 1991, pg. 54--55

6. Compare your results against scalar, unoptimized code on Crays.

It really impresses the audience when you can state that your code runs several times faster than a Cray, currently the world's dominant supercomputer. Unfortunately, with a little tuning many applications run quite fast on Crays. Therefore you must be careful not to do any tuning on the Cray code. Do not insert vectorization directives, and if you find any, remove them. In extreme cases it may be necessary to disable all vectorization with a command line flag. Also, Crays often run much slower with bank conflicts, so be sure that your Cray code accesses data with large, power-of-two strides whenever possible. It is also important to avoid multitasking and autotasking on Crays --- imply in your paper that the one processor Cray performance rates you are comparing against represent the full potential of a $25 million Cray system.