To exascale and beyond: challenges and opportunities for the most powerful computers ever created

Jeff Hammond
PhD in Chemistry (2009)
Principal Architect at NVIDIA

Abstract

Some time between now and 2023, the world's first exascale supercomputer will be deployed, with a mission to deliver scientific breakthroughs in everything ranging biochemistry to cosmology, as well as applied use cases in mechanical and nuclear engineering. While this is just another mark on the exponential growth in computing power over the past 50 years, exascale is different in that we are simultaneously reaching the limits of nanoscale engineering of semiconductors and the cost ceiling for power consumption of such systems. I will talk about the scientific breakthroughs enabled by really big computers and what programming methods are used to build the software behind these breakthroughs.

Outline

Supercomputers: what are they good for?

Battle of the exponentials

Programming models for next-generation HPC

Acknowledgements:

Peter Boyle (QCD)

David Hardy, John Stone, Julio Maia, Peng Wang (NAMD)

Sotiris Xantheas, Edo Apra (NWChem)

Content:

Salman Habib and the HACC team (HACC)

Jed Brown

Supercomputers: what are they good for?

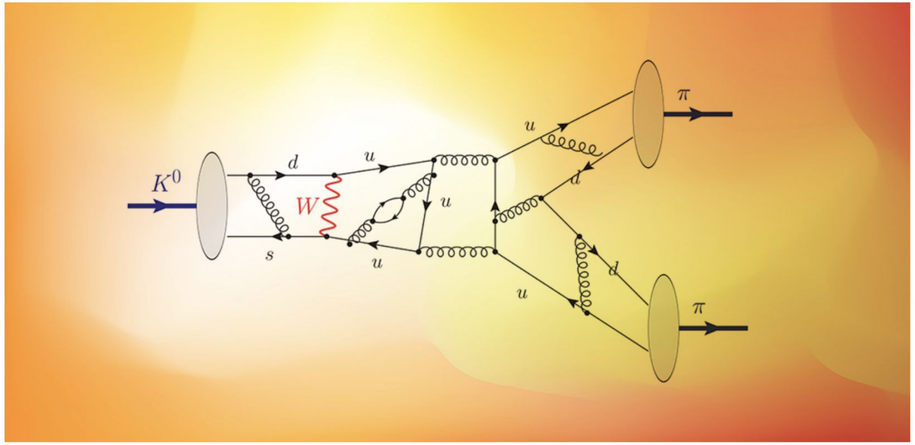
From the smallest things...

Lattice QCD is the class of models used to simulate subatomic particles using Markov Chain Monte Carlo methods and Feynman path integrals.

New Calculation Refines Comparison of Matter with Antimatter

Theorists publish improved prediction for the tiny difference in kaon decays observed by experiments

September 17, 2020

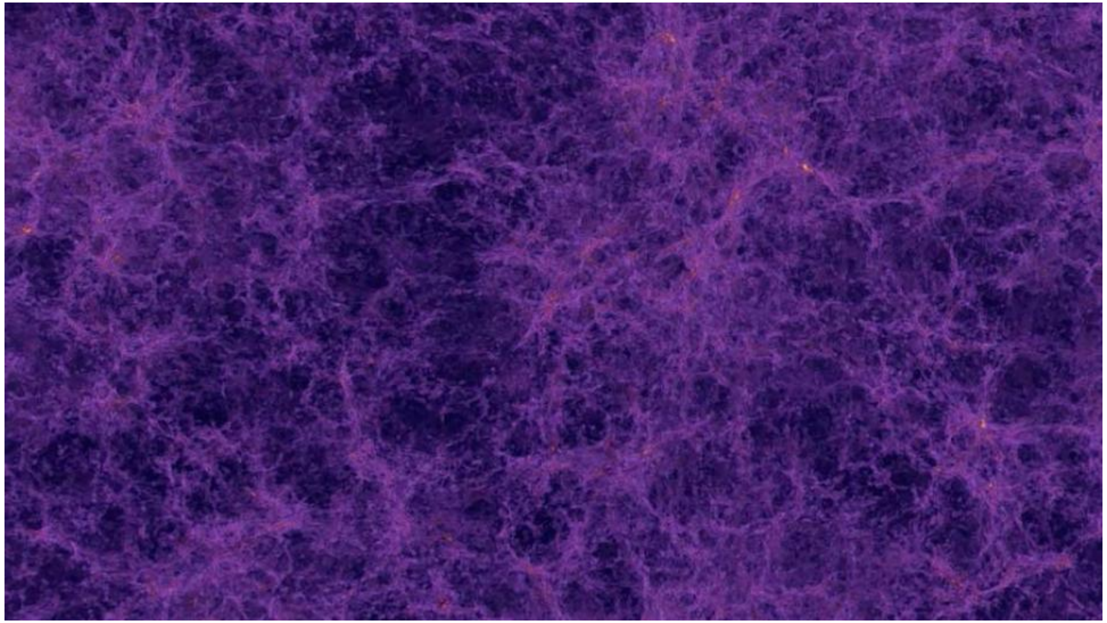


https://gtc21.event.nvidia.com/media/Grid%3A%20A%20High-Performance%20and%20Portable%20Code%20for%20Quantum%20Chromodynamics%20%5BE32016%5D/1_r97ya588

https://www.bnl.gov/newsroom/news.php?a=117325

...to the biggest things

Argonne scientists (Salman Habib and coworkers) are simulating the cosmos using all the biggest HPC systems by computing the interactions between trillions of particles.



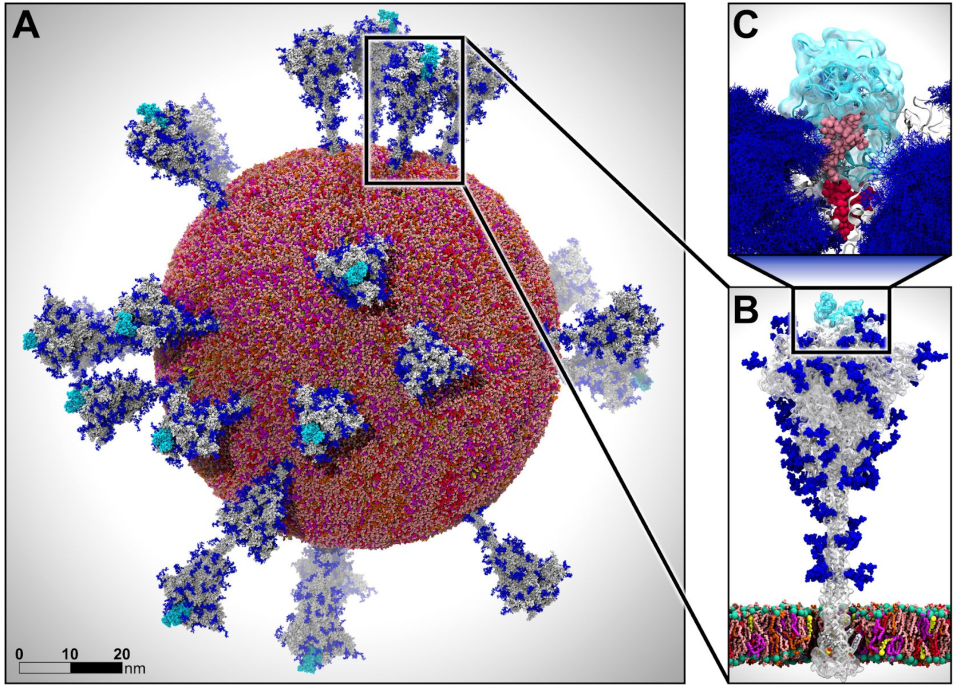
https://cacm.acm.org/magazines/2017/1/211098-hacc/fulltext

https://www.alcf.anl.gov/science/projects/computing-dark-universe

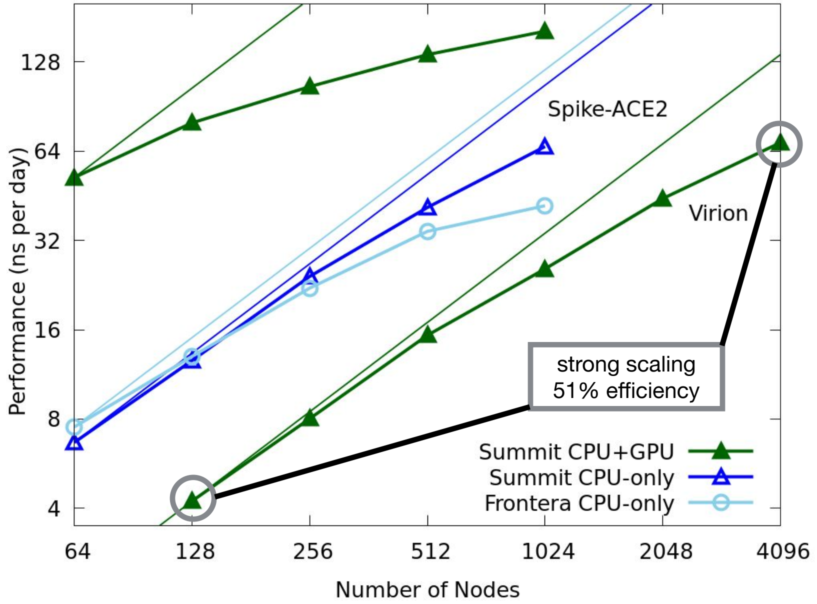
NAMD Simulating SARS-CoV-2 on Frontera and Summit

Collaboration with Amaro Lab at UCSD, images rendered by VMD
Winner of Gordon Bell Special Prize at SC20, project involved overall 1.13 Zettaflops of NAMD simulation

(A) Virion, (B) Spike, (C) Glycan shield conformations



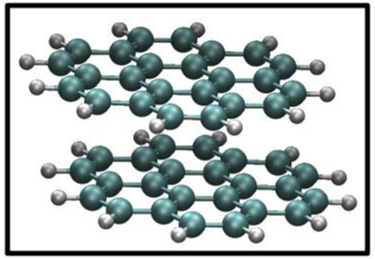
Scaling performance:
- ~305M atom virion
- ~8.5M atom spike

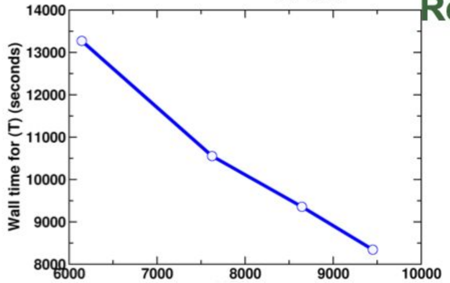


https://gtc21.event.nvidia.com/media/Molecular%20Dynamics%20Simulations%20on%20GPU-Dense%20Architectures%20with%20NAMD%20%5BS31529%5D/1_znsuv1wc

Casalino, et al. *bioRxiv* (2020)
https://doi.org/10.1101/2020.11.19.390187

Scaling of the SPEC CCSD(T) Library on the Full Partition of the KNL Nodes of the Cori Supercomputer at NERSC


Coronene Dimer $(C_{24}H_{12})_2$



Wall time for (T) vs. number of KNL nodes

Scientific Achievement

Calculation of the binding energy of the coronene dimer, an archetypal system for graphene

Significance and Impact

Ability to obtain accurate interaction energies of large systems; largest to date CCSD(T) calculation (9.14 PFLOPs) used 538,650 Knight's Landing (KNL) cores (9,450 nodes; 57/68 cores per node)

Research Details

- **216 electrons / 1,776 basis functions** (cc-pVTZ basis set)
- OpenMP for multi-threading in CCSD and CCSD(T)
- Checkpoint restart capability in CCSD
- Improved inter-node parallelization of the (T) correction on the KNL nodes using the Global Arrays (xGA) tool

Cori Supercomputer @ NERSC

# of KNL nodes/cores	(T) kernel	
	Wall time (sec)	PFLOPs
7,624 / 434,568	10,553	7.65
8,644 / 492,708	9,357	8.13
9,450 / 538,650 (97.5% of full partition)	**8,344**	**9.14**

Team: Aprà, Hammond (Intel), Daily, Palmer, Xantheas
Support from Intel's Parallel Computing Centers (IPCC)

Work was performed at Pacific Northwest National Laboratory under a NERSC Initiative for Scientific Exploration (NISE) and BES allocation awards

U.S. DEPARTMENT OF ENERGY | Office of Science

SPEC Scalable Predictive methods for Excitations and Correlated phenomena

Pacific Northwest NATIONAL LABORATORY

What are the features of a supercomputer?

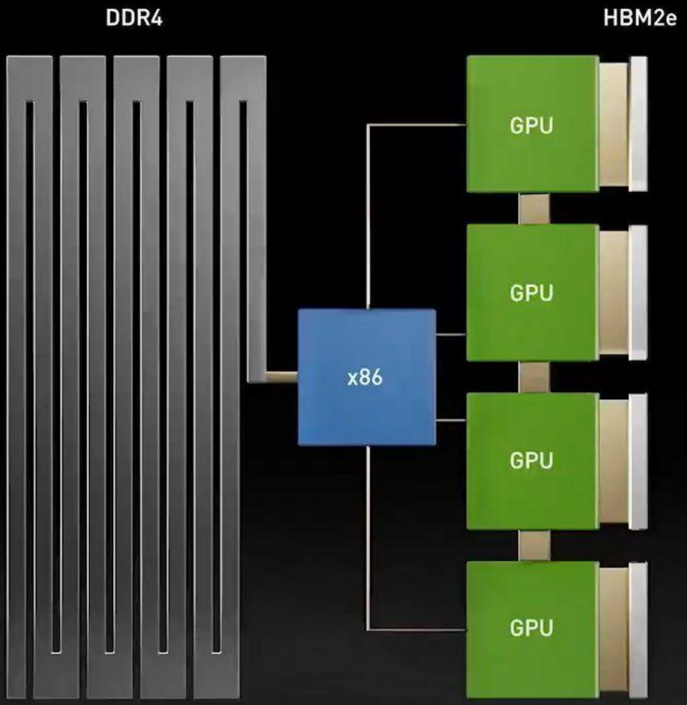
1. Lots and lots of components *working together*
    a. Production computing jobs often use 20-80% of the system for a single simulation
    b. Thousands of nodes with many cores (50+) and/or multiple (4+) GPUs per node
    c. Many terabytes or even petabytes of memory and storage
    d. Virtual all-to-all connectivity of processors, memory and storage.
2. Specialized components
    a. High-bandwidth memory: HBM is faster than DRAM, better than GDDR
    b. High-bandwidth interconnects (between nodes): 1 us latency and 10+ GB/s per link
    c. High-bandwidth interconnects (within node): >2x of PCIe BW with much lower latency and better support for HPC software
    d. More reliable components - individual component failures multiplied by system scale

GTC 21

DATA-COMPUTE DEMAND GROWING FASTER THAN SYSTEM BANDWIDTH

GPU Starved by CPU Memory and PCIE Bandwidth

GPU	8,000	GB/sec
CPU	200	GB/sec
PCIE Gen 4	16	GB/sec
Mem-to-GPU	64	GB/sec



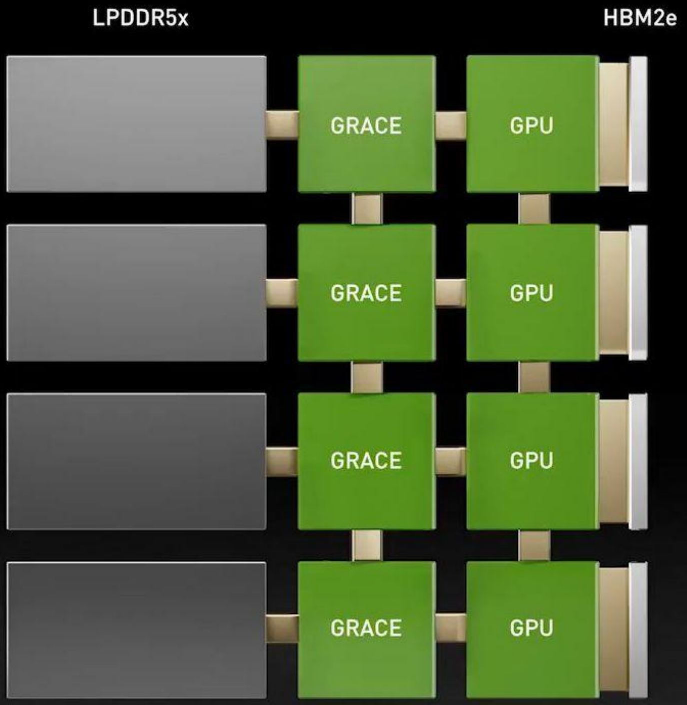
https://youtu.be/eAn_oiZwUXA

GTC 21

A NEW COMPUTING ARCHITECTURE FOR AI AND DATA SCIENCE

30X Increase System Memory to GPU

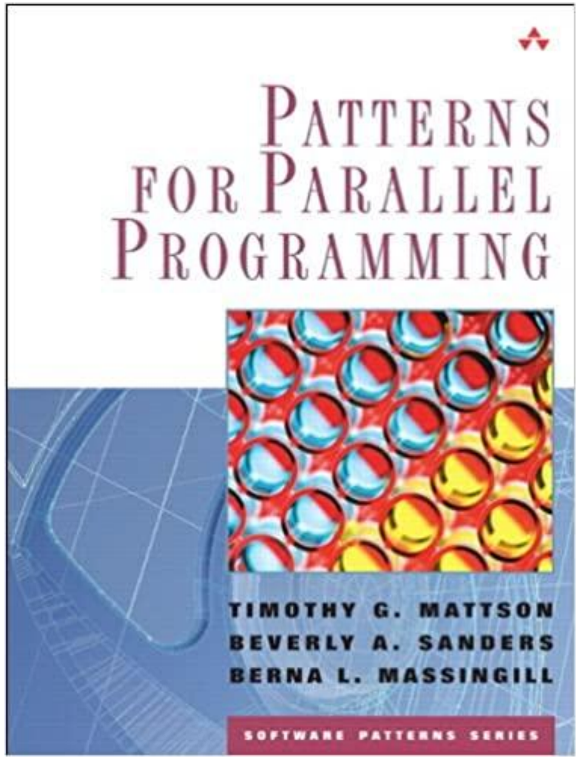
GPU	8,000	GB/sec	
CPU	500	GB/sec	
NVLINK	500	GB/sec	
Mem-to-GPU	2,000	GB/sec	30X



https://youtu.be/eAn_oiZwUXA

How do we program supercomputers?

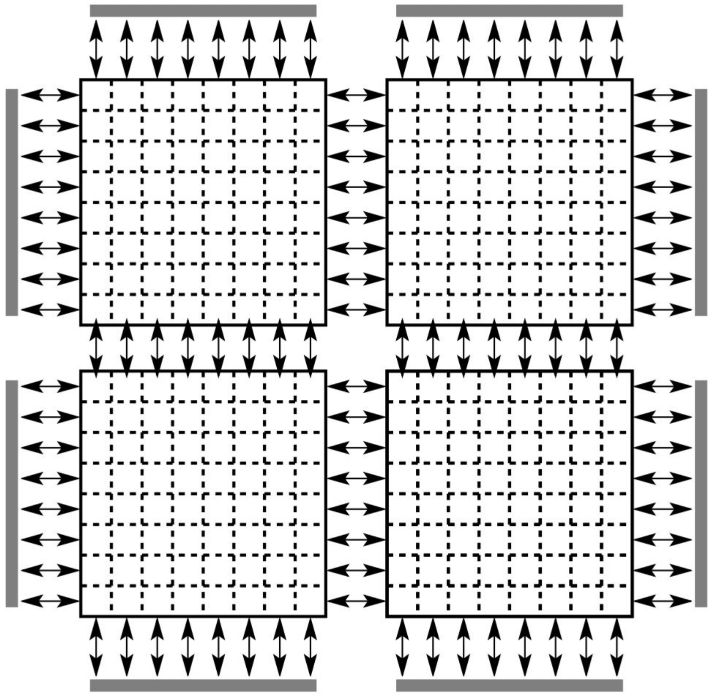
1. Find lots of computation that can be done concurrently (at the same time)
2. Figure out the input and output dependencies of those computations
3. Map the compute and data-dependency graphs to well-known patterns



Domain decomposition pattern

Lots of physics and engineering problems can be parallelized using domain decomposition, where a grid of points/cells is divided up like a checkerboard.

The groups of points need to exchange data at their boundaries (halos).

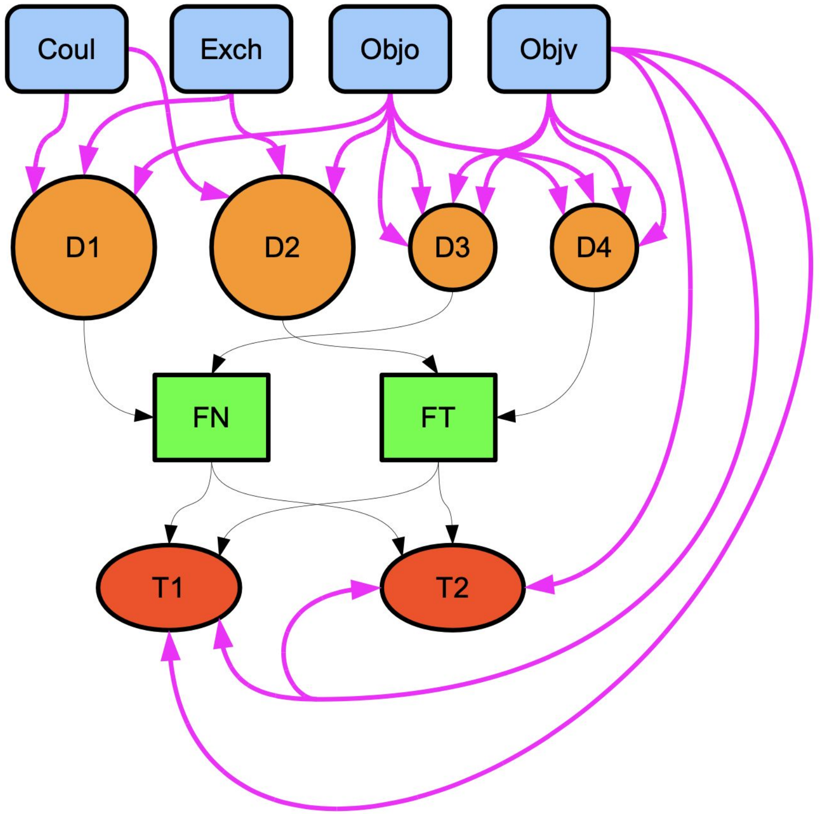


Grid QCD

Task parallelism

Task parallelism involves finding a number of tasks that work on their own data and assigning them to different processing units.

Tasks often produce data that is consumed by other tasks, or combined to produce a final result, which creates dependencies, and thus partial orderings, between tasks.

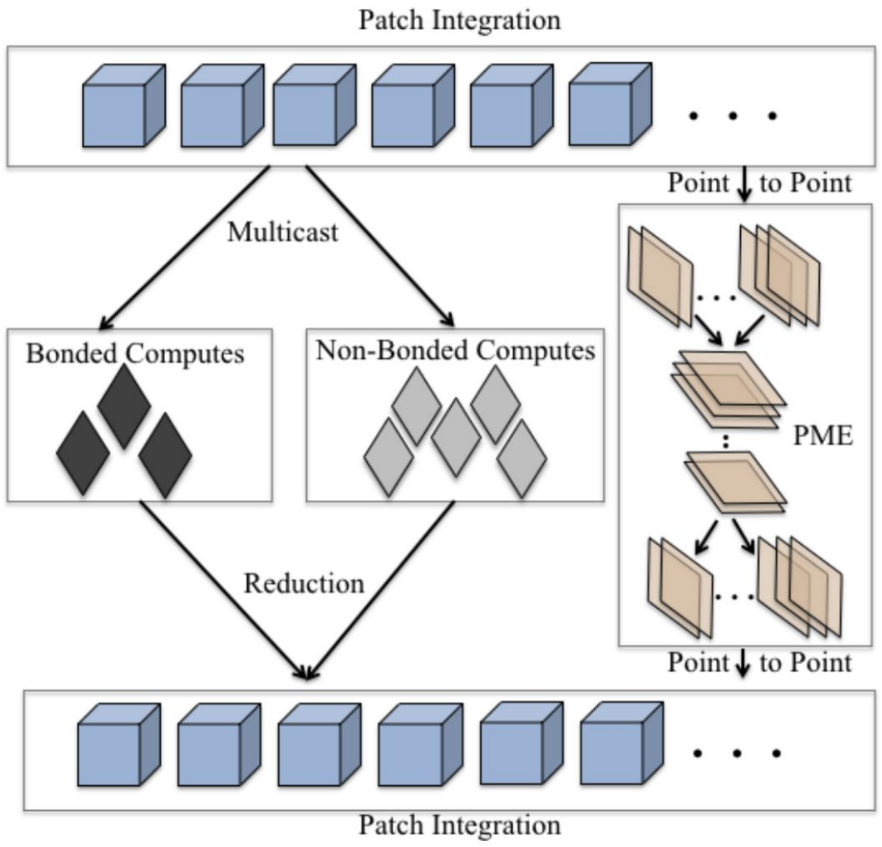


NWChem CCSD(T)

Mixed parallelism

The most successful parallel codes combine all available forms of parallelism, and use the best known strategies for each.

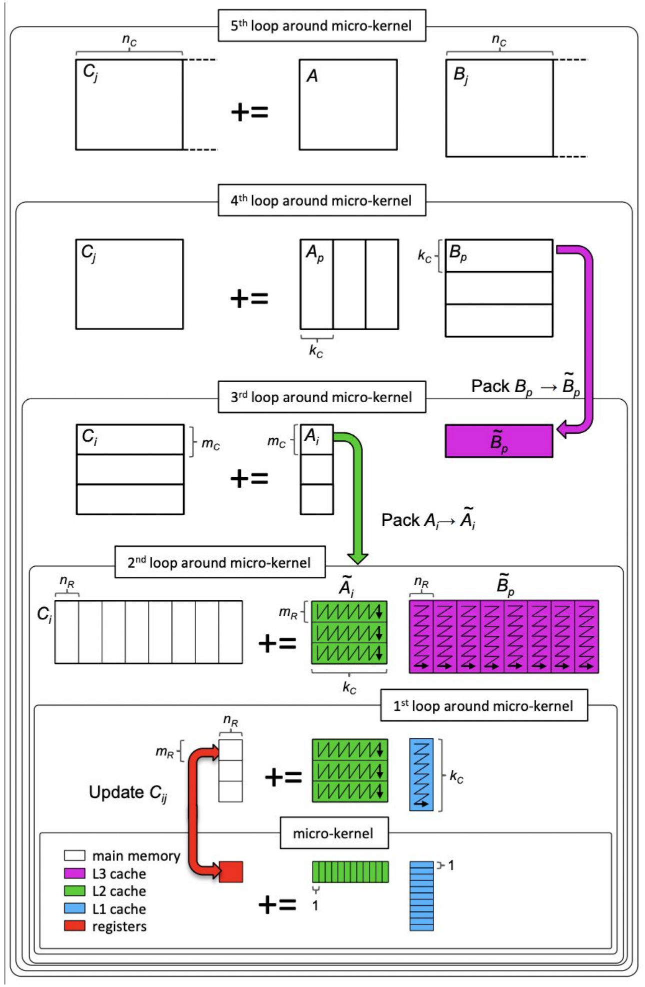
The bookkeeping associated with many forms of parallelism is challenging for programmers, hence the use of specialized systems like Charm++.



NAMD

It's parallelism all the way down...

```
Loop 5   for $j_c = 0 : n-1$ steps of $n_c$
           $\mathcal{J}_c = j_c : j_c + n_c - 1$
Loop 4     for $p_c = 0 : k-1$ steps of $k_c$
             $\mathcal{P}_c = p_c : p_c + k_c - 1$
             $B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \widetilde{B}_p$
Loop 3       for $i_c = 0 : m-1$ steps of $m_c$
               $\mathcal{I}_c = i_c : i_c + m_c - 1$
               $A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \widetilde{A}_i$
               // macro-kernel
Loop 2         for $j_r = 0 : n_c - 1$ steps of $n_r$
                 $\mathcal{J}_r = j_r : j_r + n_r - 1$
Loop 1           for $i_r = 0 : m_c - 1$ steps of $m_r$
                   $\mathcal{I}_r = i_r : i_r + m_r - 1$
                   //micro-kernel
Loop 0             for $p_r = 0 : p_c - 1$ steps of 1
                     $C_c(\mathcal{I}_r, \mathcal{J}_r) \mathrel{+}= \alpha \widetilde{A}_i(\mathcal{I}_r, p_r)\, \widetilde{B}_p(p_r, \mathcal{J}_r)$
                   endfor
                 endfor
               endfor
             endfor
           endfor
         endfor
```

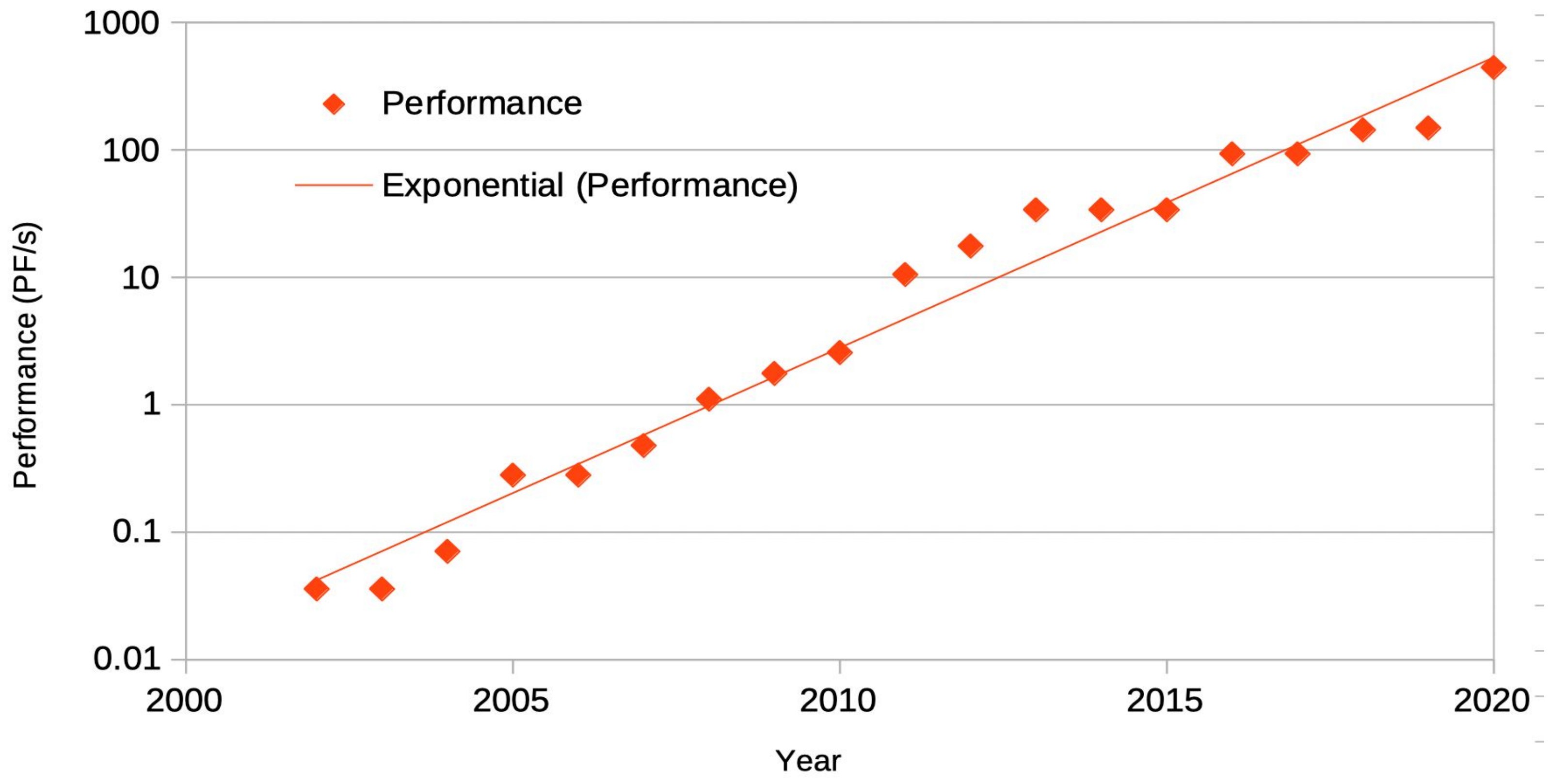

5th loop around micro-kernel
n_C
C_j
+=
A
B_j
4th loop around micro-kernel
C_j
A_p
k_C
B_p
Pack $B_p \rightarrow \widetilde{B}_p$
3rd loop around micro-kernel
C_i
m_C
A_i
$\widetilde{B}_p$
Pack $A_i \rightarrow \widetilde{A}_i$
2nd loop around micro-kernel
n_R
$\widetilde{A}_i$
m_R
$\widetilde{B}_p$
1st loop around micro-kernel
Update C_{ij}
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
1

https://www.cs.utexas.edu/users/flame/pubs/blis3_ipdps14.pdf

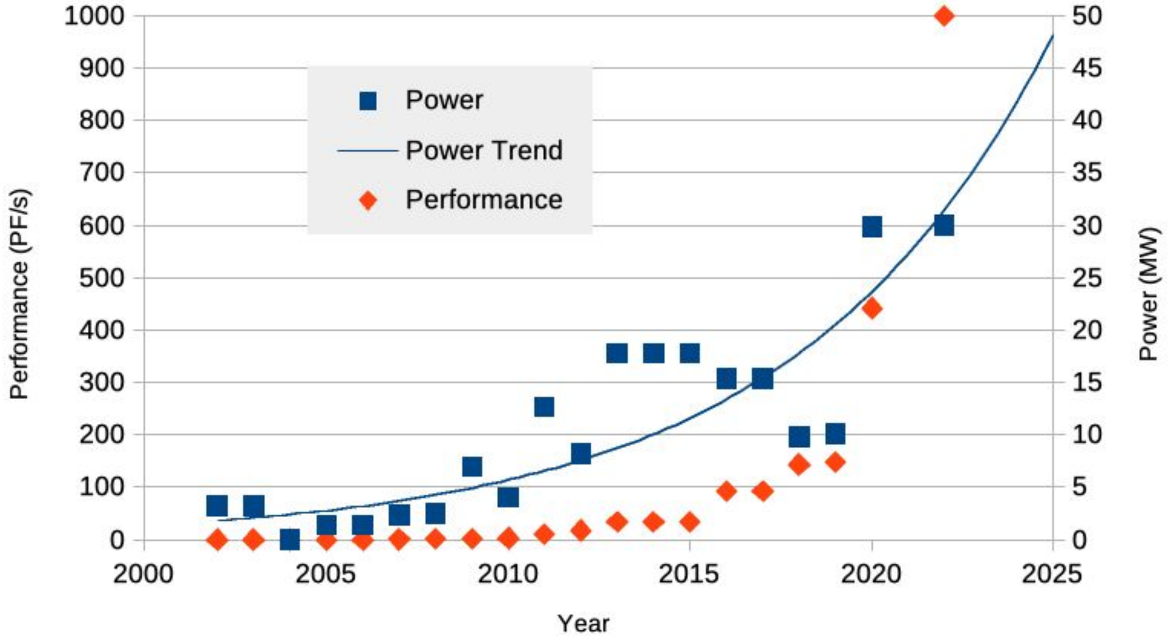
Battle of the exponentials

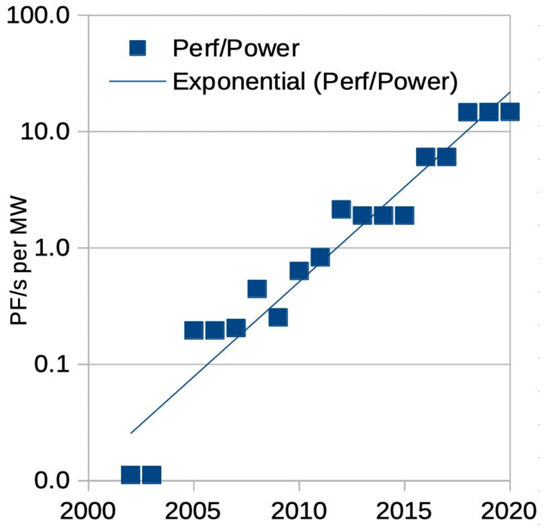
https://top500.org/

Top500 #1 performance (Rmax) is exponential...

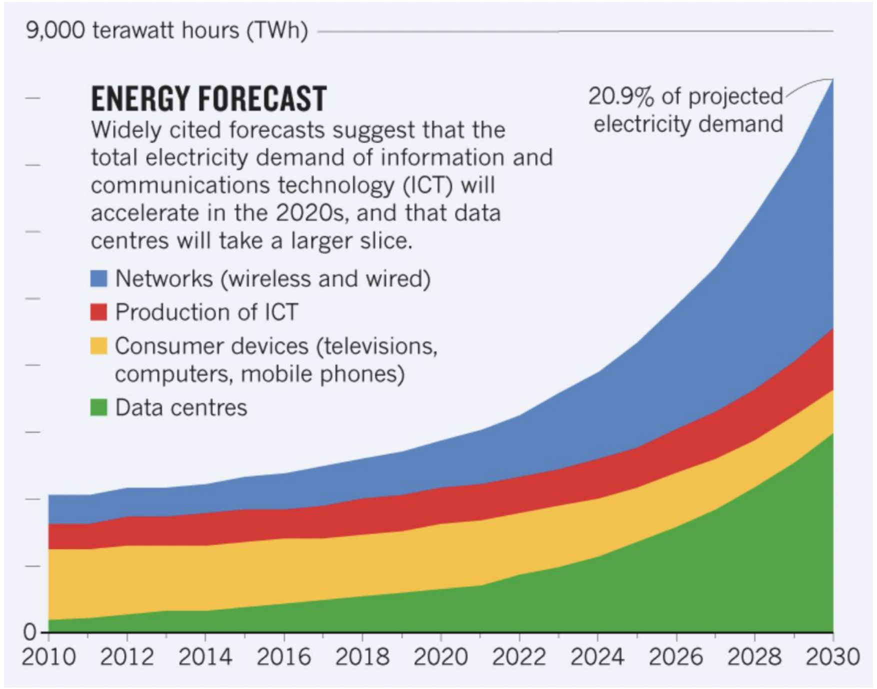


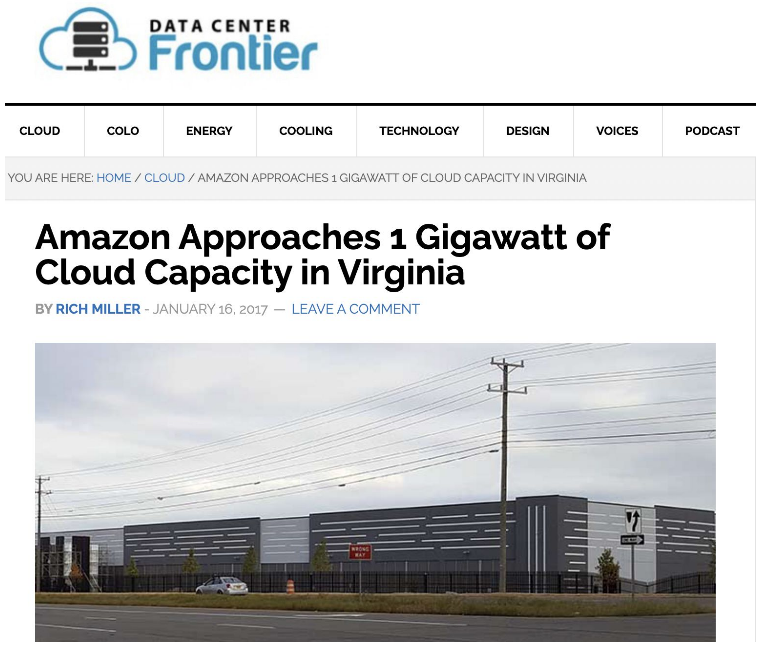
...but is fighting against another exponential: power





Computing will be limited by power





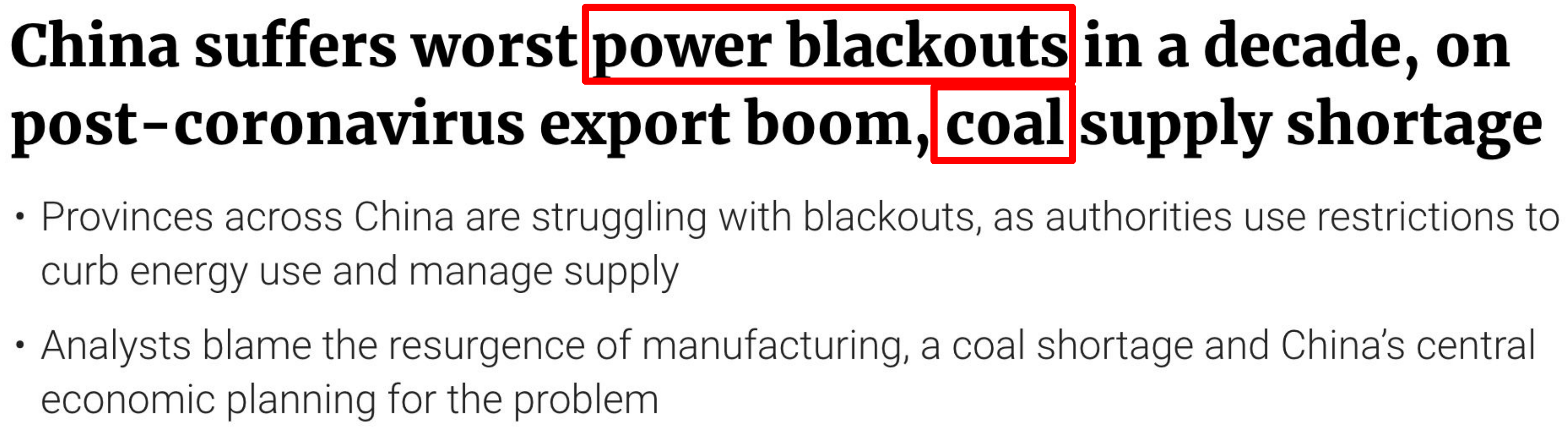
YOU ARE HERE: HOME / CLOUD / AMAZON APPROACHES 1 GIGAWATT OF CLOUD CAPACITY IN VIRGINIA

Amazon Approaches 1 Gigawatt of Cloud Capacity in Virginia

BY RICH MILLER - JANUARY 16, 2017 — LEAVE A COMMENT

https://www.nature.com/articles/d41586-018-06610-y

https://datacenterfrontier.com/amazon-approaches-1-gigawatt-of-cloud-capacity-in-virginia/



China suffers worst power blackouts in a decade, on post-coronavirus export boom, coal supply shortage

- Provinces across China are struggling with blackouts, as authorities use restrictions to curb energy use and manage supply
- Analysts blame the resurgence of manufacturing, a coal shortage and China’s central economic planning for the problem

If power supply is fixed, computers compete with homes and factories.

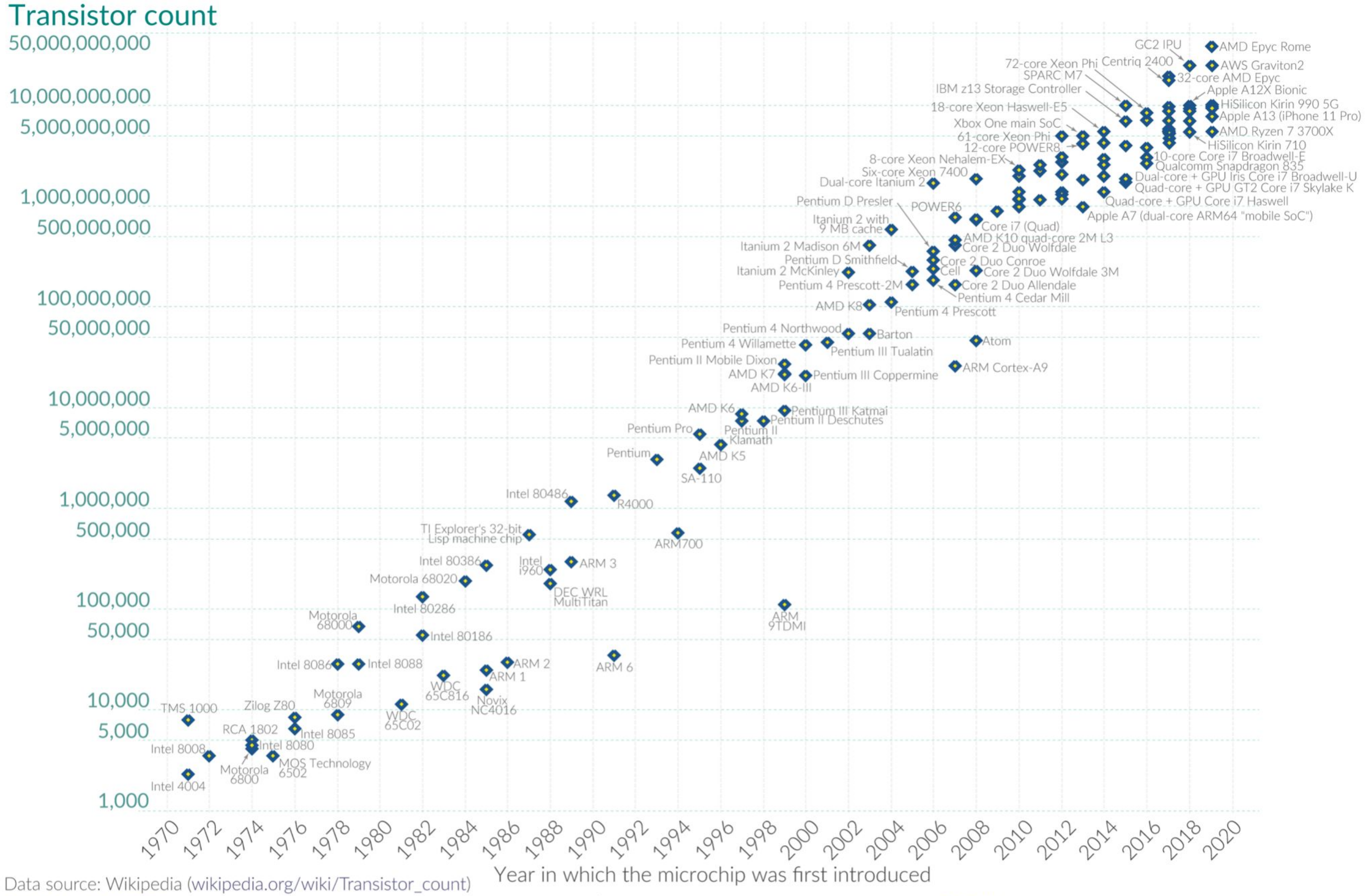
Increasing power supply has environmental consequences.

Power-intensive computing means the value of computing results will be judged more critically than in the past.

https://www.scmp.com/economy/china-economy/article/3115119/china-suffers-worst-power-blackouts-decade-post-coronavirus

Moore’s Law: The number of transistors on microchips doubles every two years

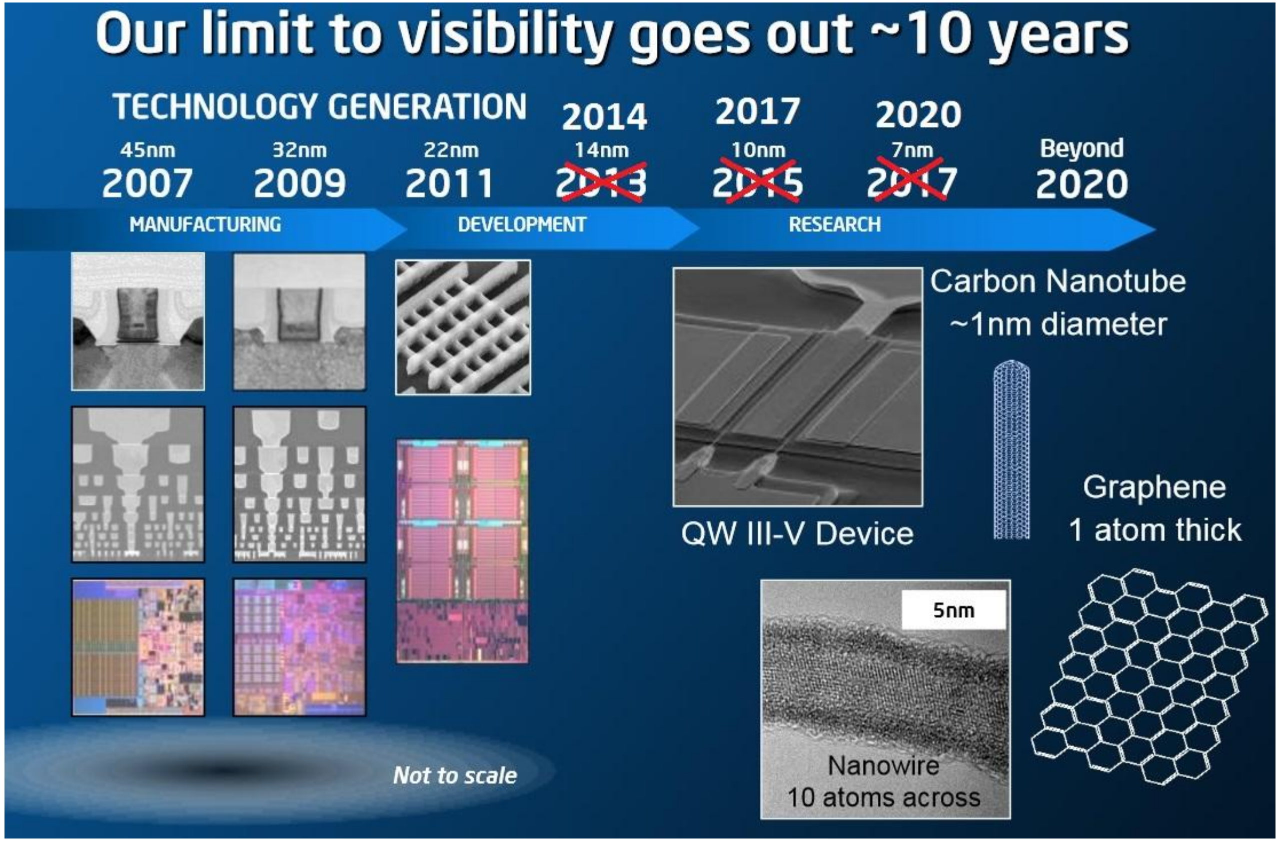
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world’s largest problems.

Moore’s Law (exponential scaling) isn’t a sure thing...



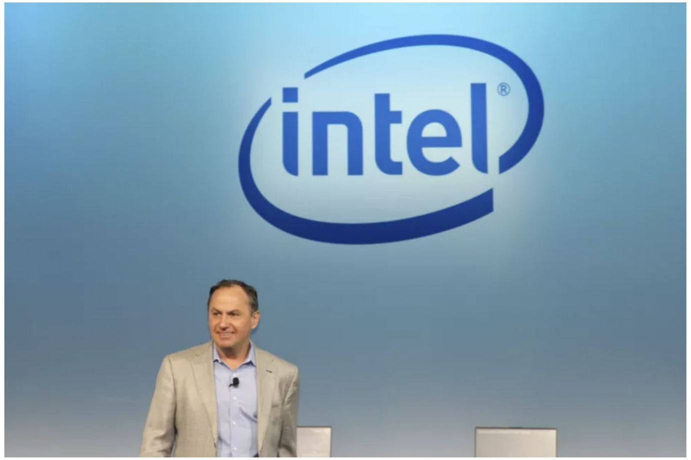
Moore’s Law has already faltered...

Intel's 7nm is Broken, Company Announces Delay Until 2022, 2023

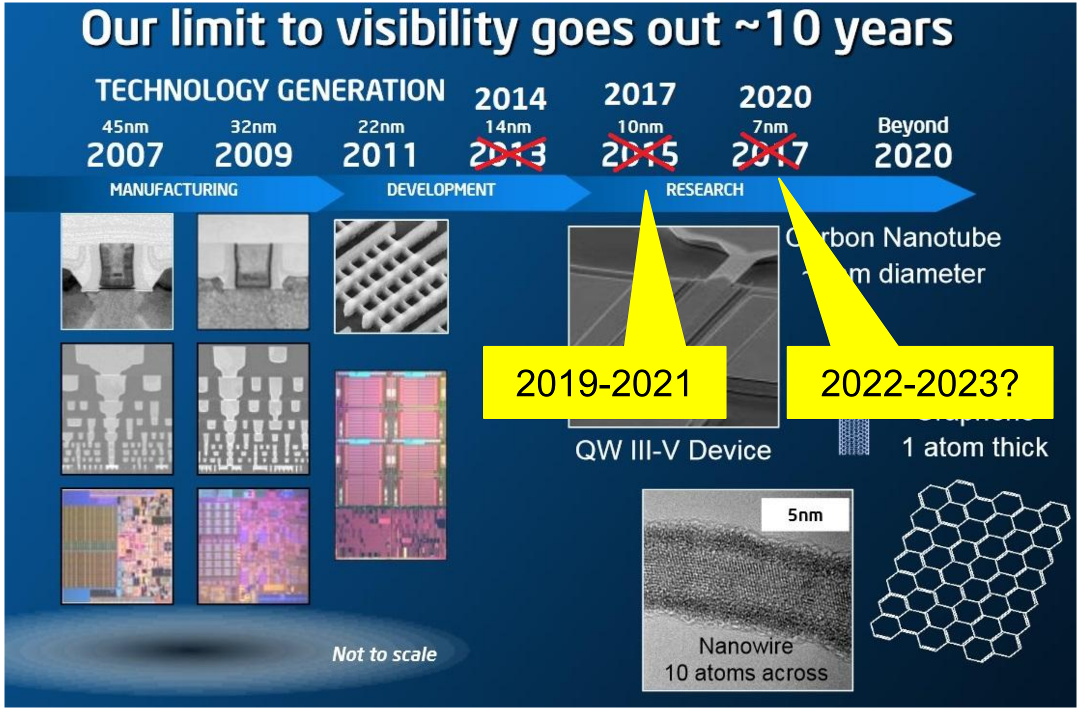
By Paul Alcorn July 23, 2020

From bad to worse

Comments (250)

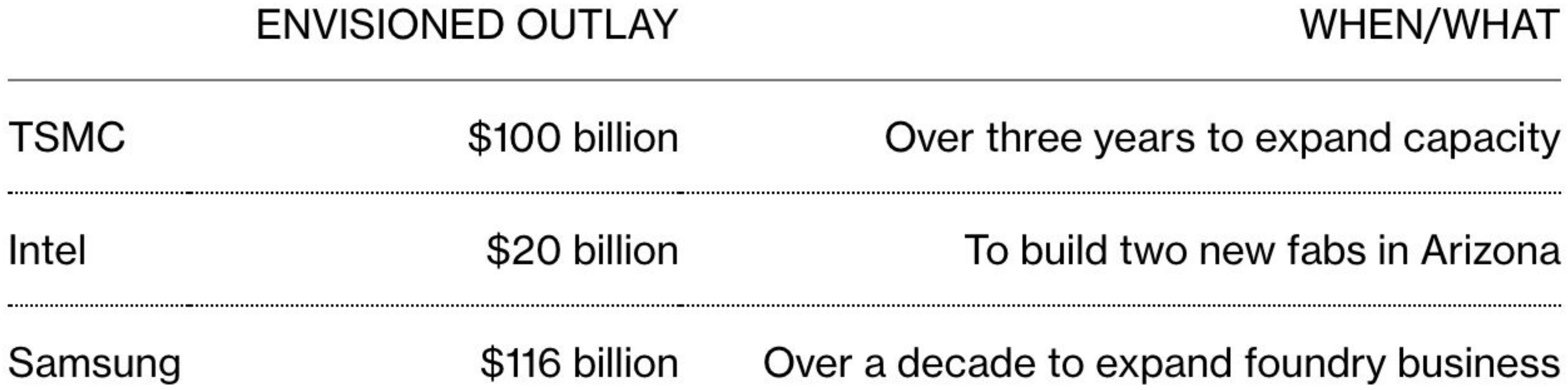


(Image credit: Tom's Hardware)



https://www.tomshardware.com/news/intel-announces-delay-to-7nm-processors-now-one-year-behind-expectations

Manufacturing costs are also growing exponentially (Rock’s Law or Moore’s Second Law)



	ENVISIONED OUTLAY	WHEN/WHAT
TSMC	$100 billion	Over three years to expand capacity
Intel	$20 billion	To build two new fabs in Arizona
Samsung	$116 billion	Over a decade to expand foundry business

https://www.bloomberg.com/news/articles/2021-04-01/tsmc-to-invest-100-billion-over-three-years-to-grow-capacity

Summary

- No more frequency scaling: ~all performance comes from parallelism
- Power efficiency growth is not keeping up with compute demand (Dennard)
- The manufacturing exponential means that more transistors cost more money

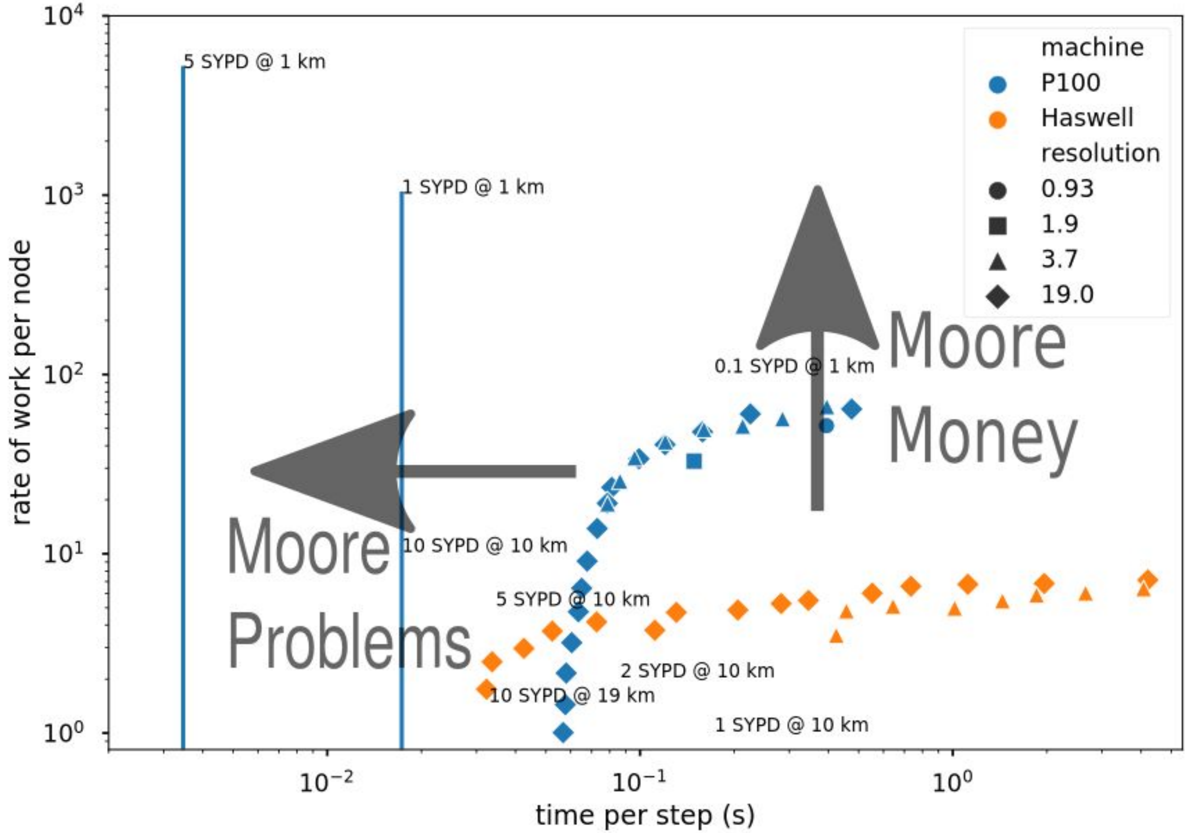
Compute demand outpaces power efficiency growth and transistor manufacturing, so we computing usage will be prioritized by who can afford the power bill and the transistors.

When we talk about “democratizing computing”, we don’t mean the Citizens United form of democracy...

More Moore is not necessarily better

“Bandwidth is money, latency is physics”

HPC applications are hitting the latency wall, particularly in critical scientific domains like weather and climate modeling.



https://jedbrown.org/files/20190822-Latsis.pdf

Programming models for next-generation HPC

HPC vs the Internet

Orders of magnitude difference in latency sensitivity differentiates software:

- Internet computation consumed by humans on the scale of **milliseconds**.
- Internet computation consumed by computers on the scale of seconds?
- HPC computation consumed by computers on the scale of **microseconds**.
- HPC computation consumed by humans on the scale of minutes.

To achieve microsecond latencies, HPC hard-wires the network routing, which changes the reliability model significantly. Internet workloads are highly resilient, whereas most HPC codes crash as soon as the hardware exposes a single fault (server hardware hides correctable hardware errors).

Distributed computing

Cloud

- https://en.wikipedia.org/wiki/Internet_protocol_suite
- Standard since the 1970s, new features added at the top of the stack.

HPC

- MPI: Standard since the 1990s, still changing.
- New features added at the bottom of the stack (expose more HW/perf).

Computing within the node

CPU programming is relatively consistent since the 1970s:

- Fortran 77, 90, 95, 2003, 2008, 2018, … 2100
- C++ 98, 03, 11, 14, 17, 20, 23, 26, 29, ...
- New languages percolate in from outside of HPC, e.g. Python
- Few successes born within the HPC community: Matlab and Julia

Computing within the node

GPU computing for HPC began is relatively new:

2002 contorting physics onto graphics programming

2003 Brook (C with streams)

2006 CUDA introduced by NVIDIA

2011 OpenACC 1.0 (directives for GPUs)

2013 OpenMP 4.0 (more directives for GPUs)

2020 ISO language standard parallelism for GPUs

https://www.nvidia.com/content/gtc-2010/pdfs/2275_gtc2010.pdf

Normalizing GPU computing

GPU computing has become progressively easier since 2002 but easier GPU computing is not sufficient. We need parallel computing to be GPU computing.

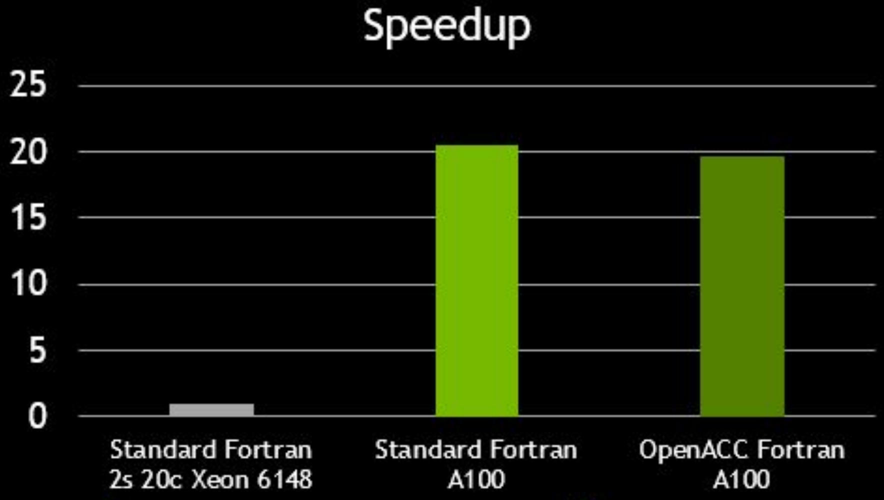
- ISO Fortran 2008 standard parallelism runs on GPUs now
- ISO C++17 standard parallelism runs on GPUs now
- Python supports GPUs with standard tools like Numba

These developments require GPU hardware features like independent forward progress of threads and unified memory. NVIDIA Volta is the first GPU that can run the same software as CPUs (assuming it uses ISO standard features).

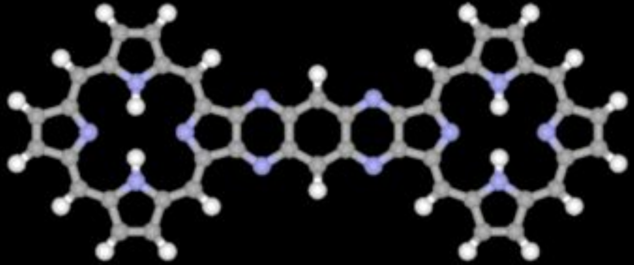
https://developer.nvidia.com/blog/accelerating-fortran-do-concurrent-with-gpus-and-the-nvidia-hpc-sdk/
https://developer.nvidia.com/blog/accelerating-standard-c-with-gpus-using-stdpar/
https://youtu.be/KhZvrF_w1ak
https://developer.nvidia.com/blog/numba-python-cuda-acceleration/
https://youtu.be/75LcDvlEIYw
https://developer.nvidia.com/blog/unified-memory-cuda-beginners/

NWChem TCE CCSD(T) Kernel

Computational Chemistry with Fortran Standard Parallelism



Same Standard Fortran Code



- NWChem provides a massively parallel implementation of the "gold standard" CCSD(T) method that scales to hundreds of thousands of CPU cores.
- The compute bottleneck is a set of 27 loop-driven tensor contractions, which are part of the >100k LOC TCE module.

https://github.com/jeffhammond/nwchem-tce-triples-kernels

The End