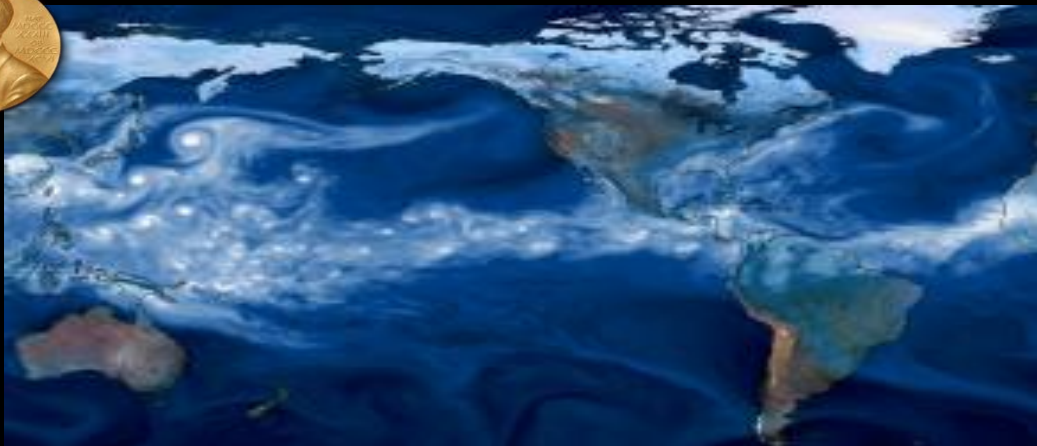
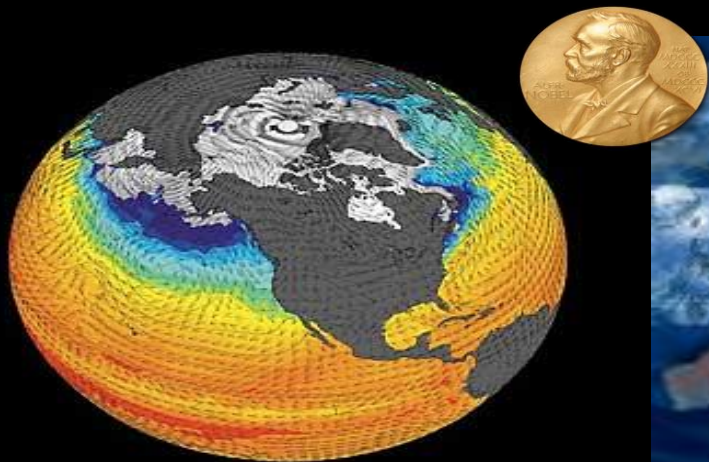


# Programming Methodologies

**John Urbanic**

Parallel Computing Scientist  
Pittsburgh Supercomputing Center

# FLOPS we need: Climate change analysis



---

## Simulations

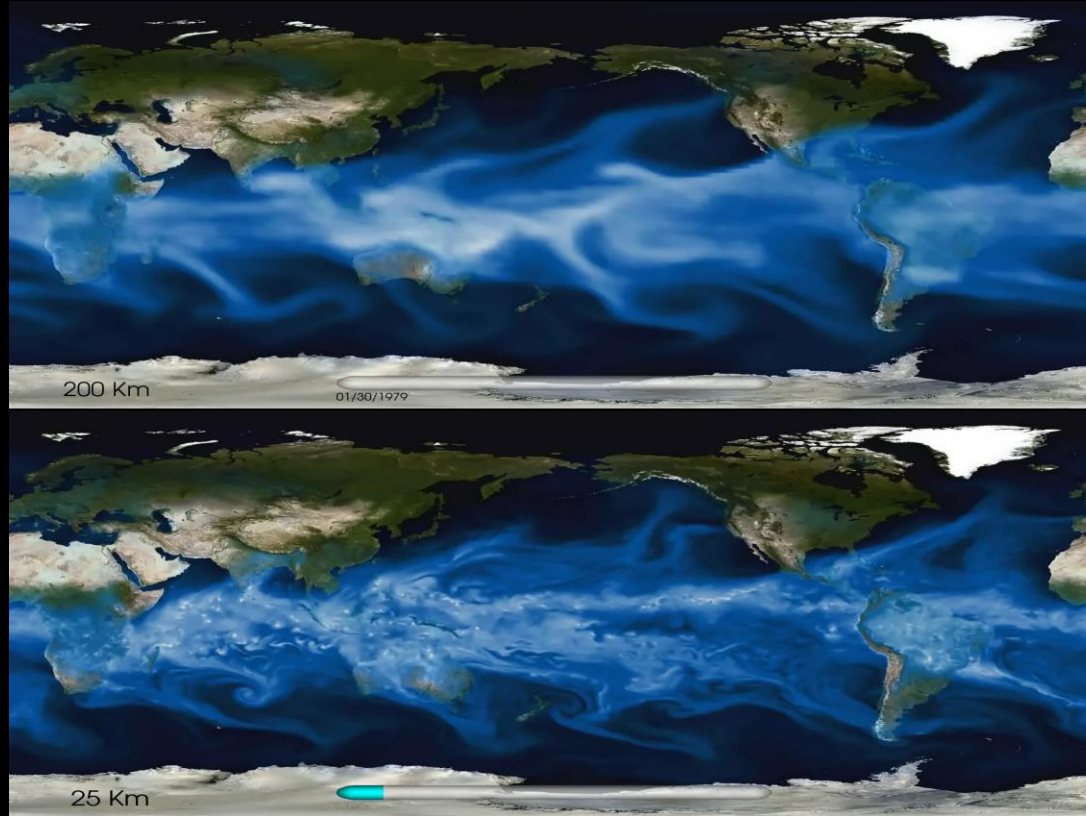
- Cloud resolution, quantifying uncertainty, understanding tipping points, etc., will drive climate to exascale platforms
- New math, models, and systems support will be needed

---

## Extreme data

- “Reanalysis” projects need 100× more computing to analyze observations
  - Machine learning and other analytics are needed today for petabyte data sets
  - Combined simulation/observation will empower policy makers and scientists
-

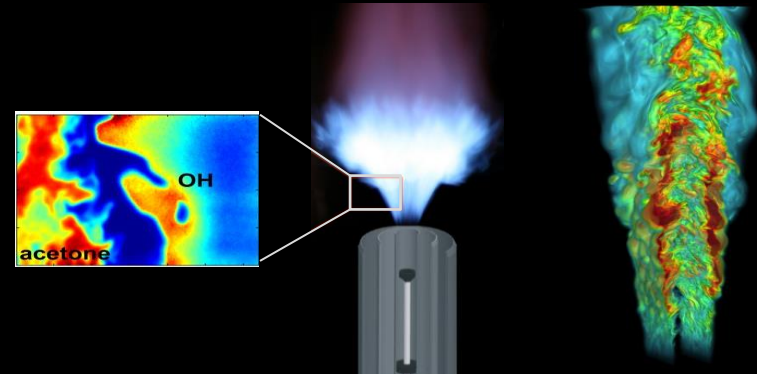
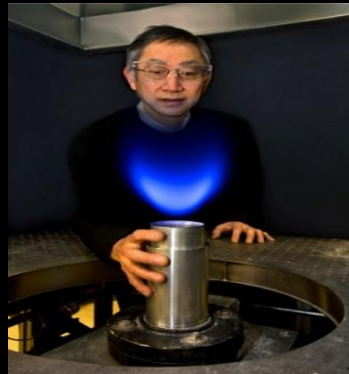
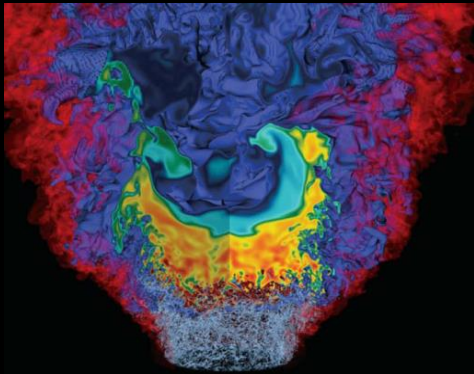
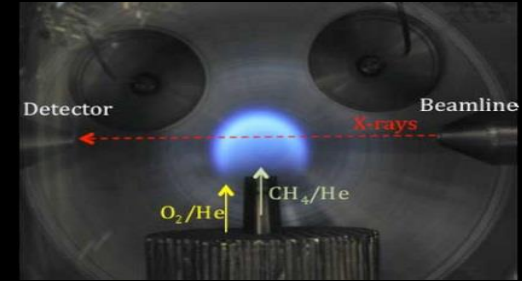
# Qualitative Improvement of Simulation with Higher Resolution (2011)



Michael Wehner, Prabhat, Chris Algeri, Fuyu Li, Bill Collins, Lawrence Berkeley National Laboratory; Kevin Reed, University of Michigan; Andrew Gettelman, Julio Bacmeister, Richard Neale, National Center for Atmospheric Research

# Exascale combustion simulations

- Goal: 50% improvement in engine efficiency
- Center for Exascale Simulation of Combustion in Turbulence (ExaCT)
  - Combines M&S and experimentation
  - Uses new algorithms, programming models, and computer science











# Modha Group at IBM Almaden



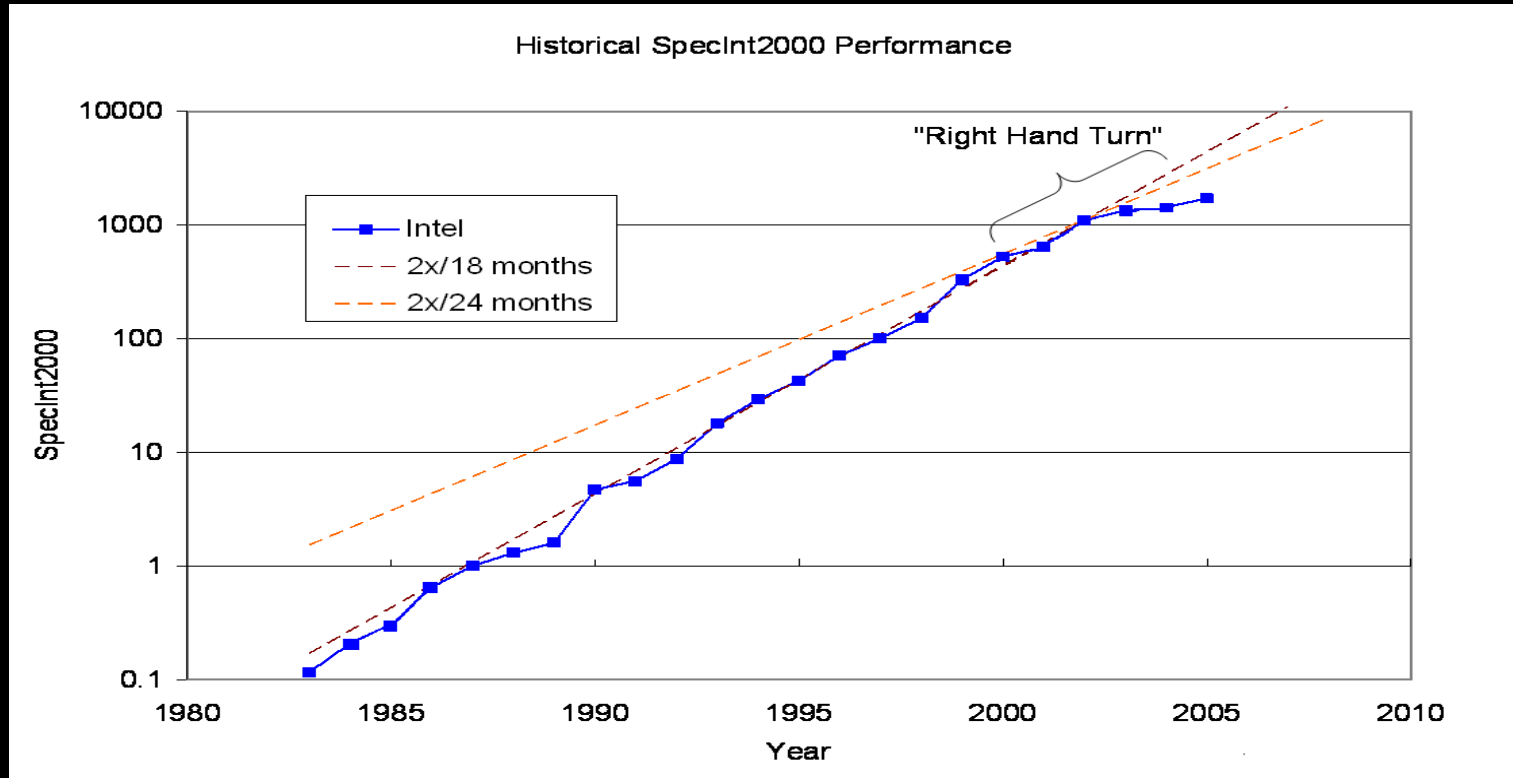
Mouse	Rat	Cat	Monkey	Human
N: $16 \times 10^6$	$56 \times 10^6$	$763 \times 10^6$	$2 \times 10^9$	$22 \times 10^9$
S: $128 \times 10^9$	$448 \times 10^9$	$6.1 \times 10^{12}$	$20 \times 10^{12}$	$220 \times 10^{12}$



				
Almaden	Watson	WatsonShaheen	LLNL Dawn	LLNL Sequoia
BG/L	BG/L	BG/P	BG/P	BG/Q
December, 2006	April, 2007	March, 2009	May, 2009	June, 2012

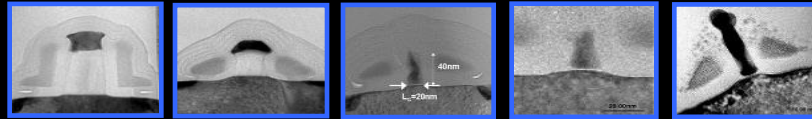
Recent simulations achieve  
unprecedented scale of  
 $65 \times 10^9$  neurons and  $16 \times 10^{12}$  synapses

# Waiting for Moore's Law to save your serial code start getting bleak in 2004

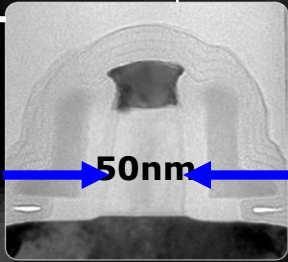


# Moore's Law is not at all dead...

## Intel process technology capabilities

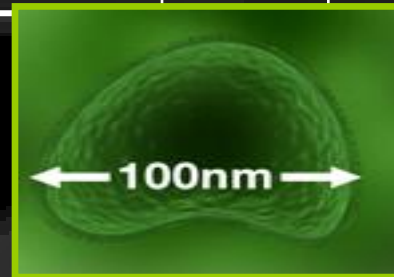


High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Feature Size	90nm	65nm	45nm	32nm	22nm	16nm	11nm	8nm
Integration Capacity (Billions of Transistors)	2	4	8	16	32	64	128	256



**Transistor for  
90nm Process**

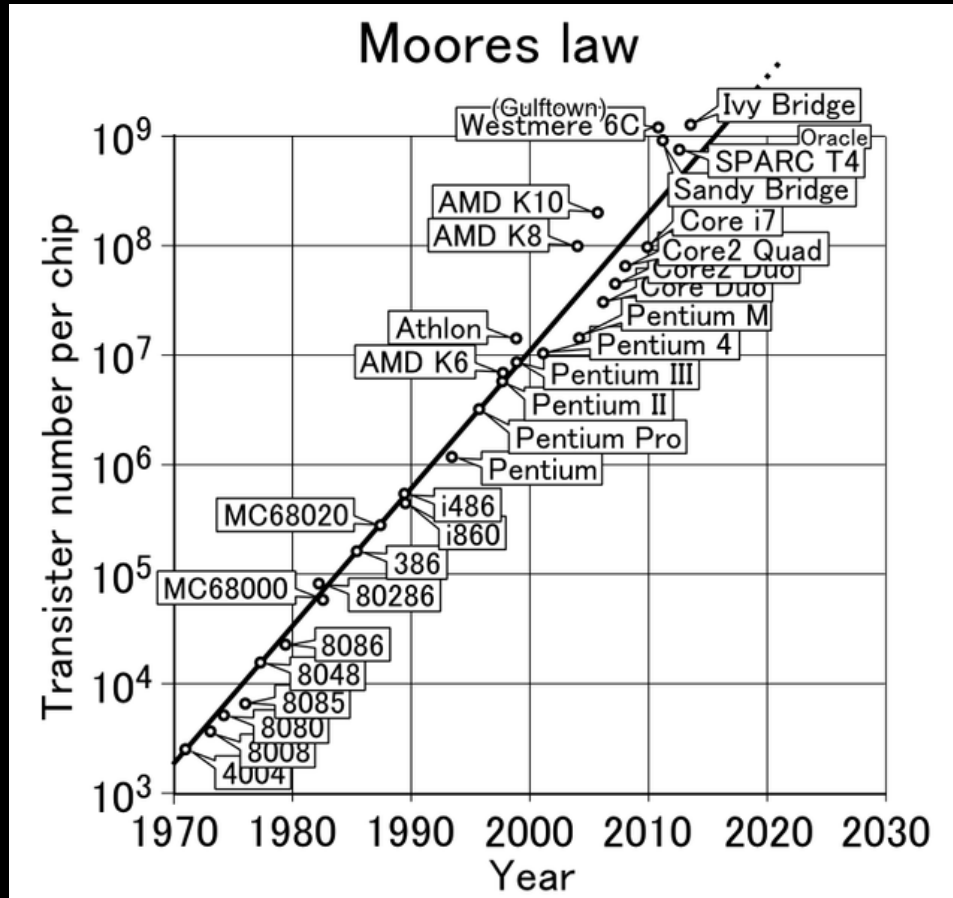
Source: Intel



**Influenza Virus**

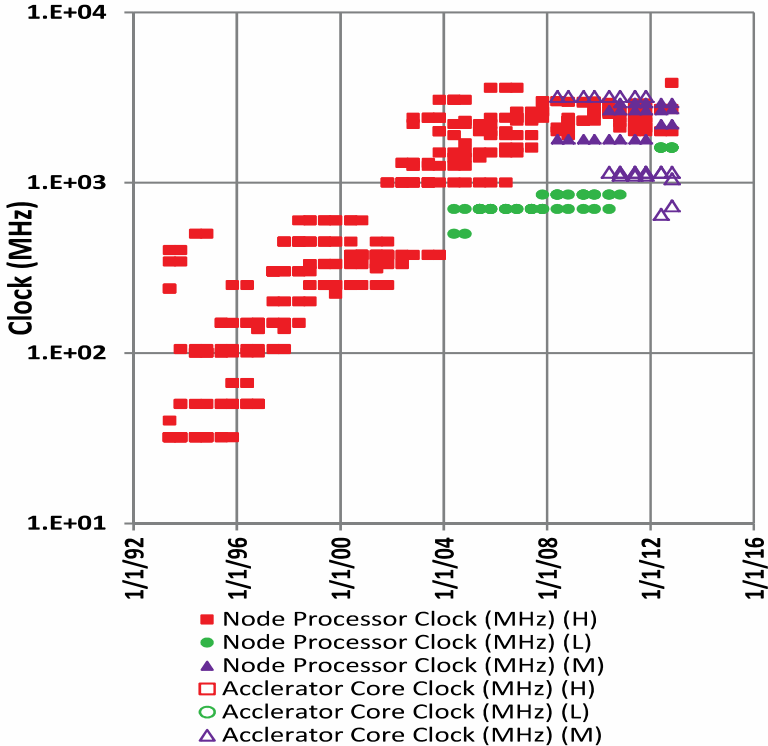
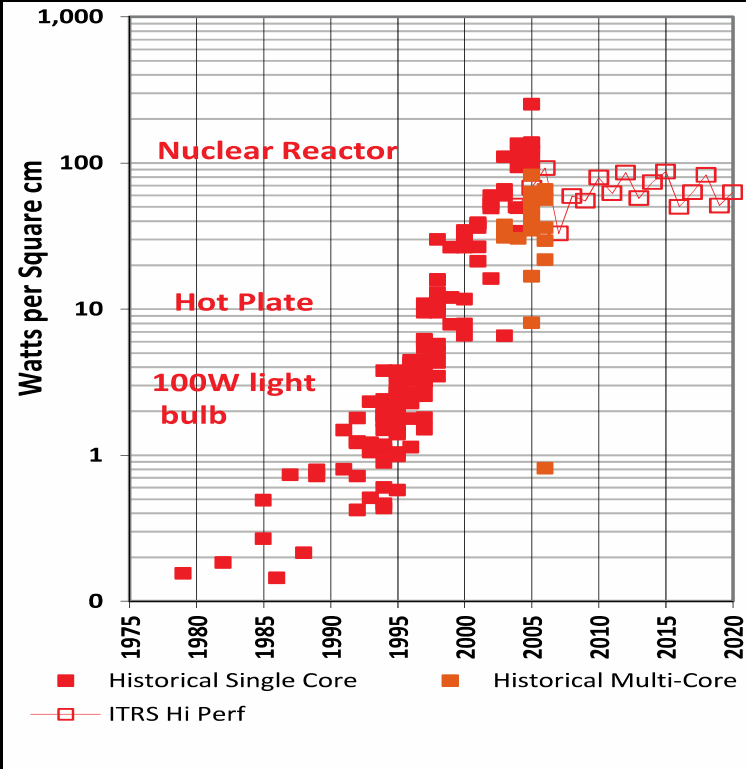
Source: CDC

At end of day, we keep using all those new transistors.



Courtesy Horst Simon, LBNL

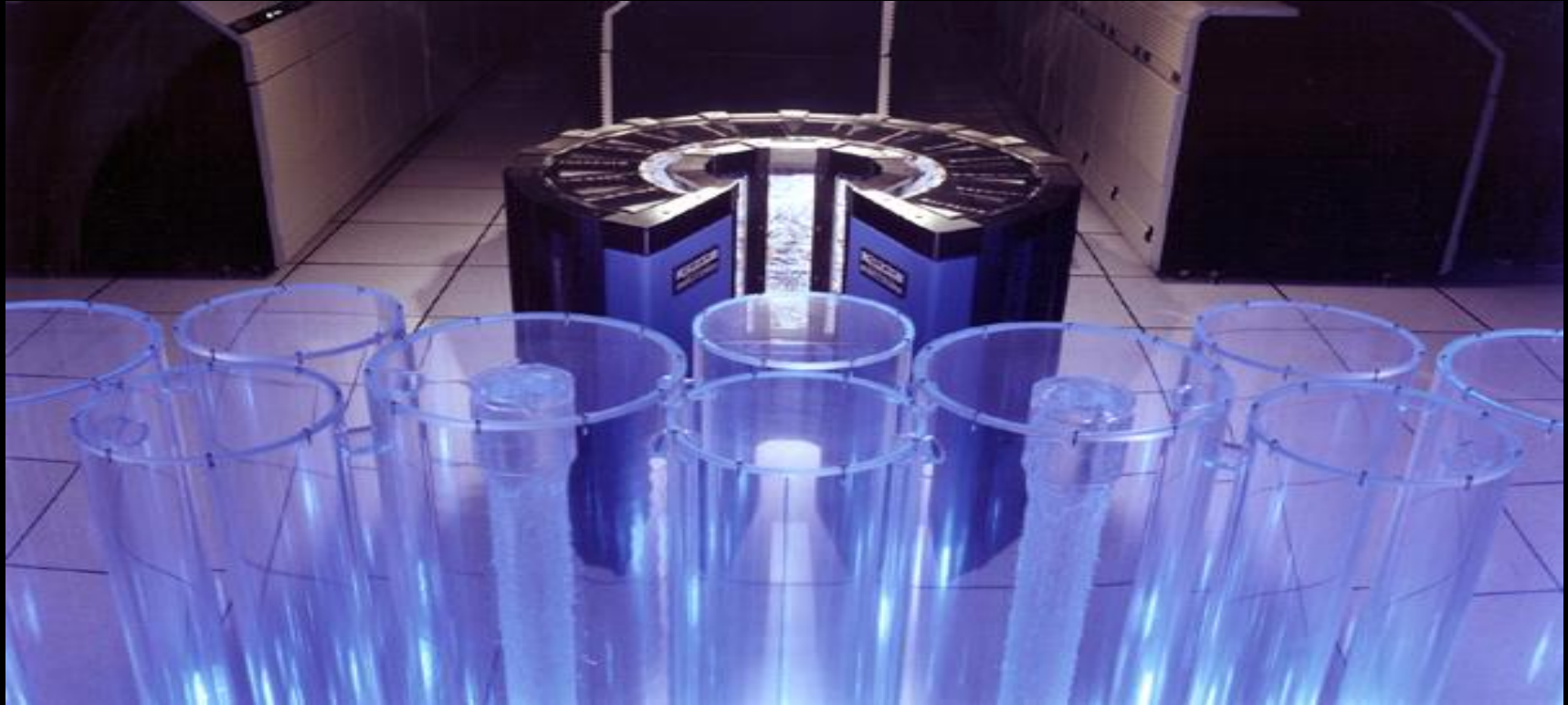
# That Power and Clock Inflection Point in 2004... didn't get better.



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

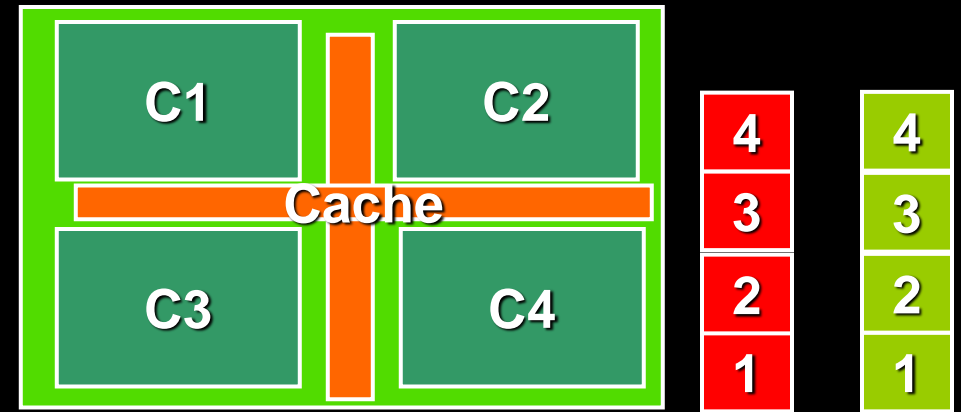
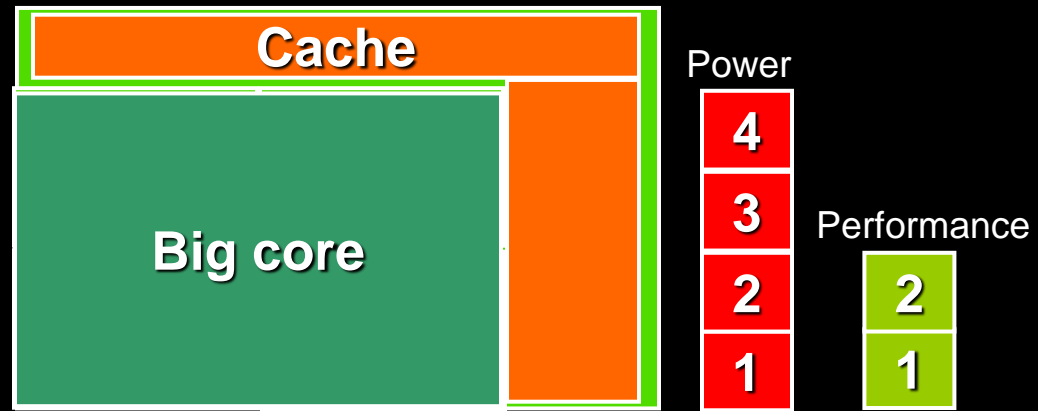


# Not a new problem, just a new scale...



Cray-2 with cooling tower in foreground, circa 1985

# How to get same number of transistors to give us more performance without cranking up power?



Key is that  
Performance  $\approx \sqrt{\text{area}}$

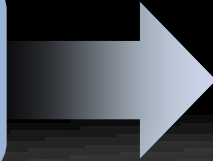


Power =  $\frac{1}{4}$   
Performance =  $\frac{1}{2}$

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

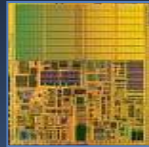
## **RULE OF THUMB**

**A 15%  
Reduction  
In Voltage  
Yields**



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

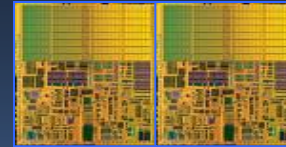
### **SINGLE CORE**



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

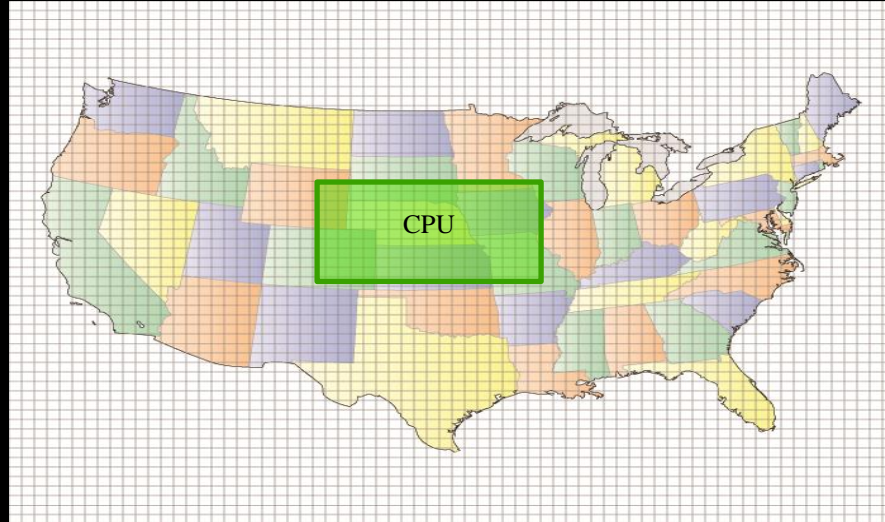


### **DUAL CORE**

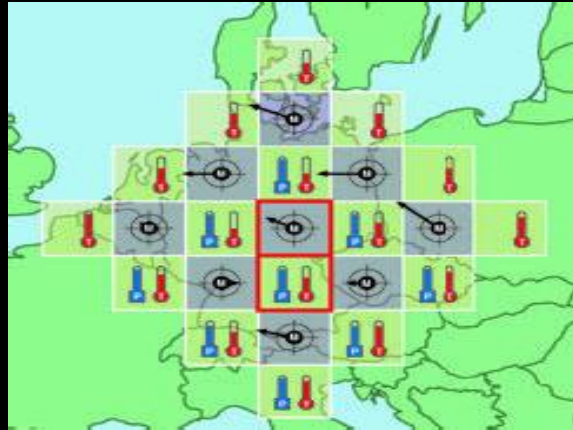


**Area = 2**  
**Voltage = 0.85**  
**Freq = 0.85**  
**Power = 1**  
**Perf = ~1.8**

# Prototypical Application: Serial Weather Model



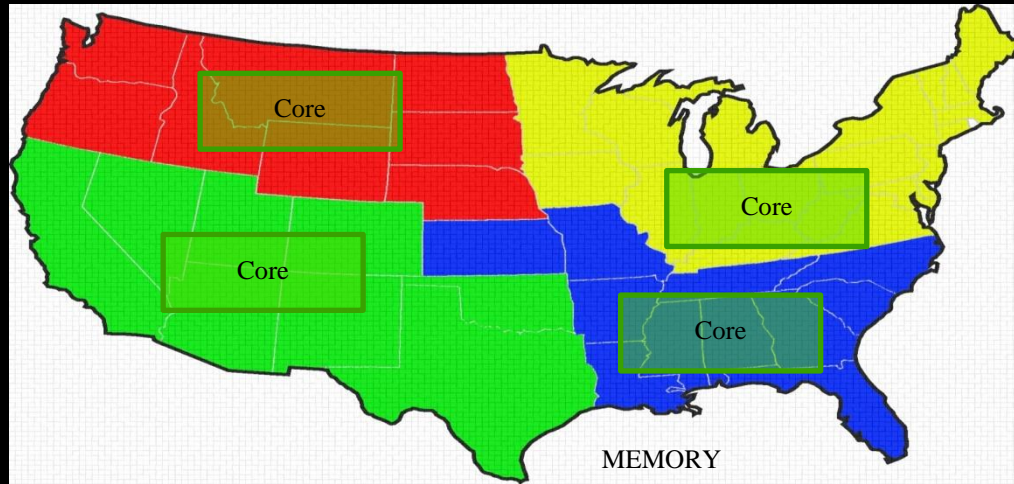
# First parallel Weather Modeling algorithm: Richardson in 1917



*Courtesy John Burkhardt, Virginia Tech*

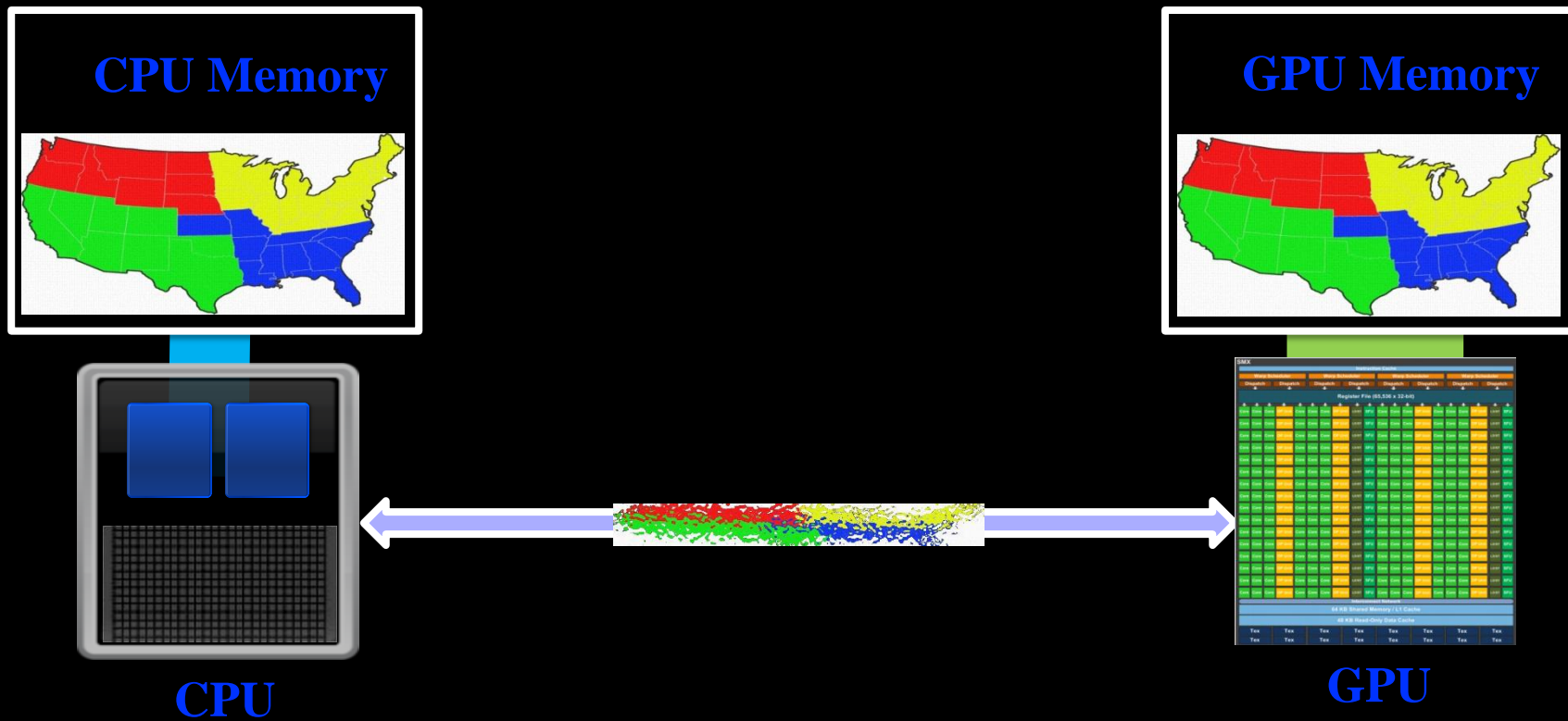


# Weather Model: Shared Memory (OpenMP)



*Four guys in the same room sharing the map.*

# Weather Model: Accelerator (OpenACC)



*1 guy coordinating 1000 using tin cans and a string.*

# Directives: an elegant and common approach.

## OpenMP

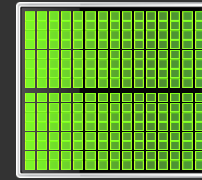
### CPU



```
main() {  
    double pi = 0.0; long i;  
  
    #pragma omp parallel for reduction(+:pi)  
    for (i=0; i<N; i++)  
    {  
        double t = (double)((i+0.05)/N);  
        pi += 4.0/(1.0+t*t);  
    }  
  
    printf("pi = %f\n", pi/N);  
}
```

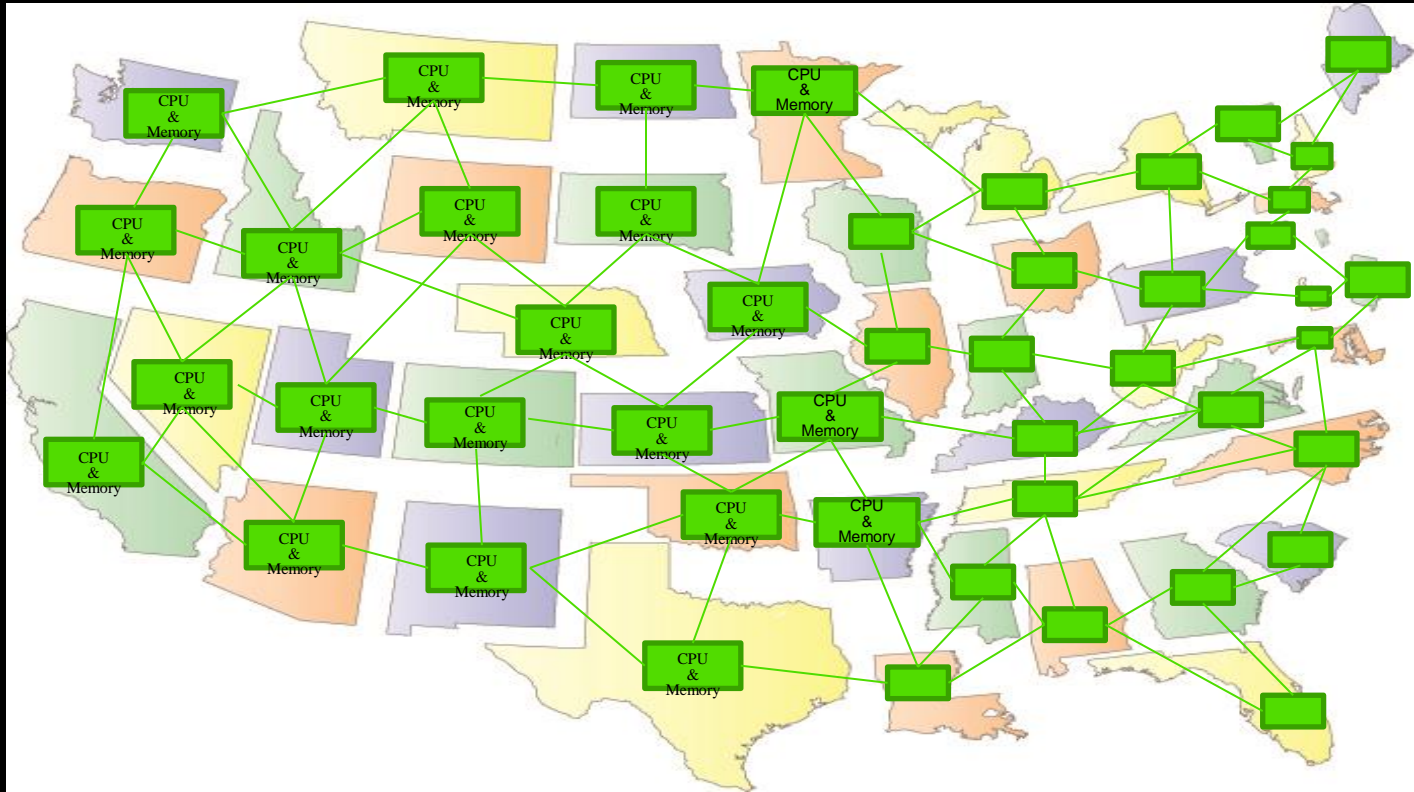
## OpenACC

### GPU



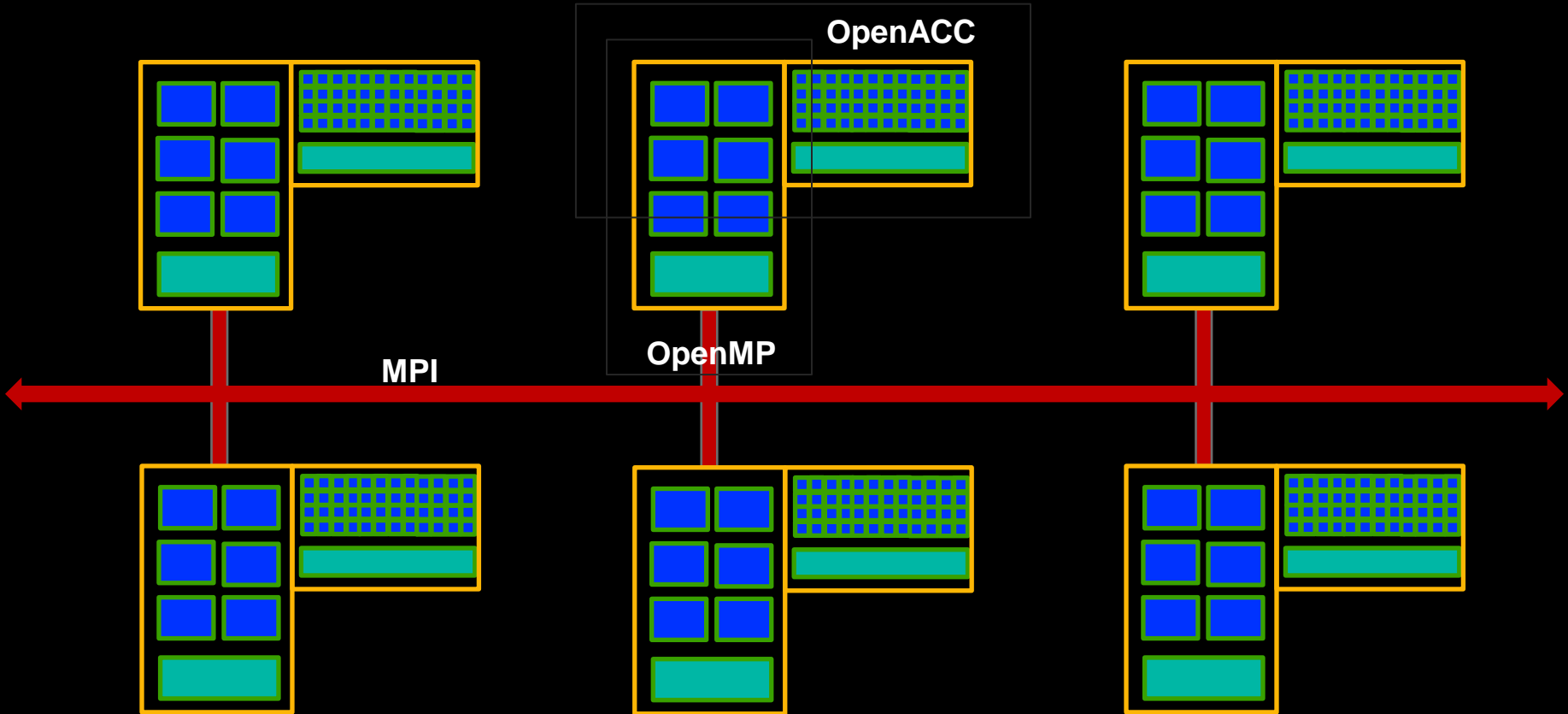
```
main() {  
    double pi = 0.0; long i;  
  
    #pragma acc kernels  
    for (i=0; i<N; i++)  
    {  
        double t = (double)((i+0.05)/N);  
        pi += 4.0/(1.0+t*t);  
    }  
  
    printf("pi = %f\n", pi/N);  
}
```

# Weather Model: Distributed Memory (MPI)



*50 guys using a telegraph.*

# The pieces fit like this...





# Other Paradigms?

- ✓ • Message Passing
  - MPI
- ✓ • Data Parallel
  - Fortran90
- ✓ • Threads
  - OpenMP, OpenACC, CUDA
- PGAS
  - UPC, Coarray Fortran
- Frameworks
  - Charm++
- ✓ • Hybrid
  - MPI + OpenMP

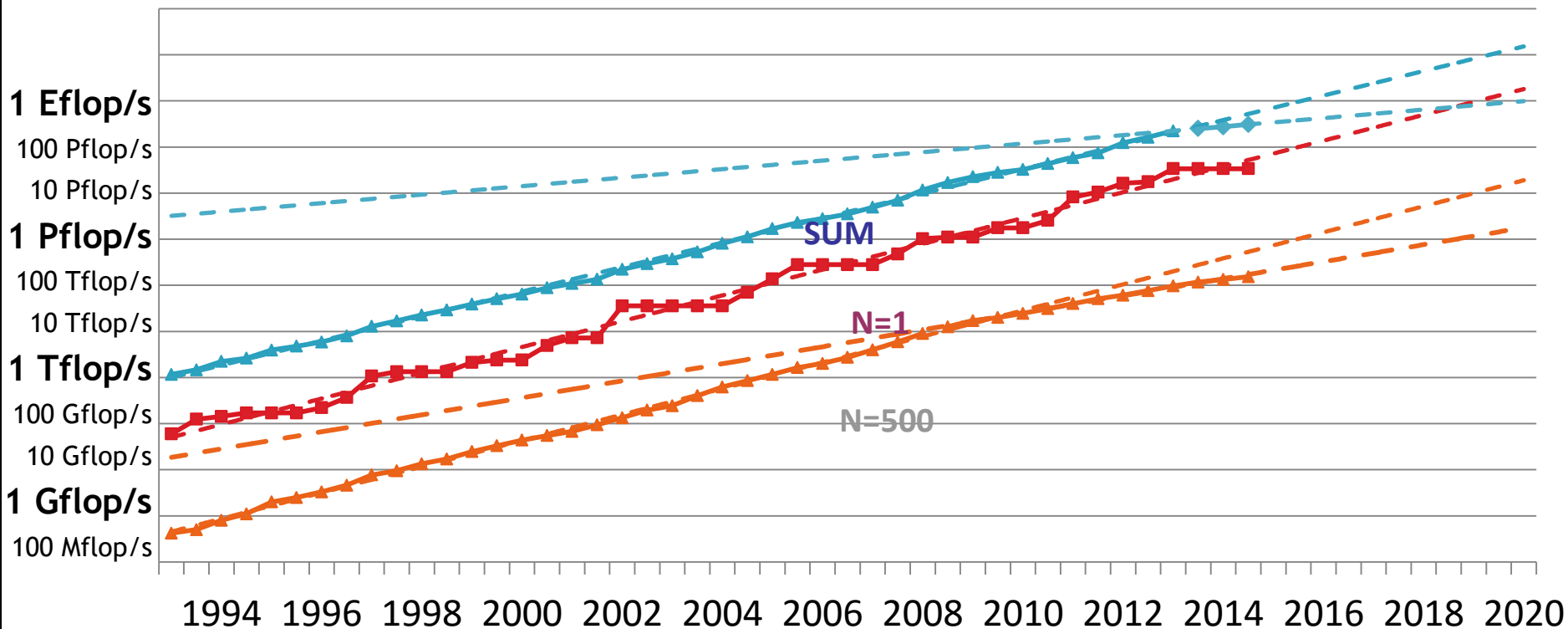
Top 10 Systems as of June 2016								
#	Site	Manufacturer	Computer	CPU Interconnect [Accelerator]	Cores	Rmax (Tflops)	Rpeak (Tflops)	Power (MW)
1	National Super Computer Center in Guangzhou China	NRCPC	Sunway TaihuLight	Sunway SW26010 260C 1.45GHz	10,649,600	93,014	125,435	15.3
2	National Super Computer Center in Guangzhou China	NUDT	Tianhe-2 (MilkyWay)	Intel Xeon E5-2692 2.2 GHz TH Express-2 Intel Xeon Phi 31S1P	3,120,000	33,862	54,902	17.8
3	DOE/SC/Oak Ridge National Laboratory United States	Cray	Titan Cray XK7	Opteron 6274 2.2 GHz Gemini NVIDIA K20x	560,640	17,590	27,112	8.2
4	DOE/NNSA/LLNL United States	IBM	Sequoia BlueGene/Q	Power BQC 1.6 GHz Custom	1,572,864	17,173	20,132	7.8
5	RIKEN Advanced Institute for Computational Science (AICS) Japan	Fujitsu	K Computer	SPARC64 VIIIfx 2.0 GHz Tofu	705,024	10,510	11,280	12.6
6	DOE/SC/Argonne National Laboratory United States	IBM	Mira BlueGene/Q	Power BQC 1.6 GHz Custom	786,432	8,586	10,066	3.9
7	DOE/NNSA/LANL/SNL United States	Cray	Trinity Cray XC40	Xeon E5-2698v3 2.3 GHz Aries	301,056	8,100	11,078	
8	Swiss National Supercomputing Centre (CSCS) Switzerland	Cray	Piz Daint Cray XC30	Xeon E5-2670 2.6 GHz Aries NVIDIA K20x	115,984	6,271	7,788	2.3
9	HLRS Germany	Cray	Hazel Hen Cray XC40	Xeon E5-2680 2.5 GHz Aries	185,088	5,640	7,403	
10	King Abdullah University of Science and Technology	Cray	Shaheen II Cray XC40	Xeon E5-2698v3 2.3 GHz Aries	196,608	5,537	7,235	2.8

# Today

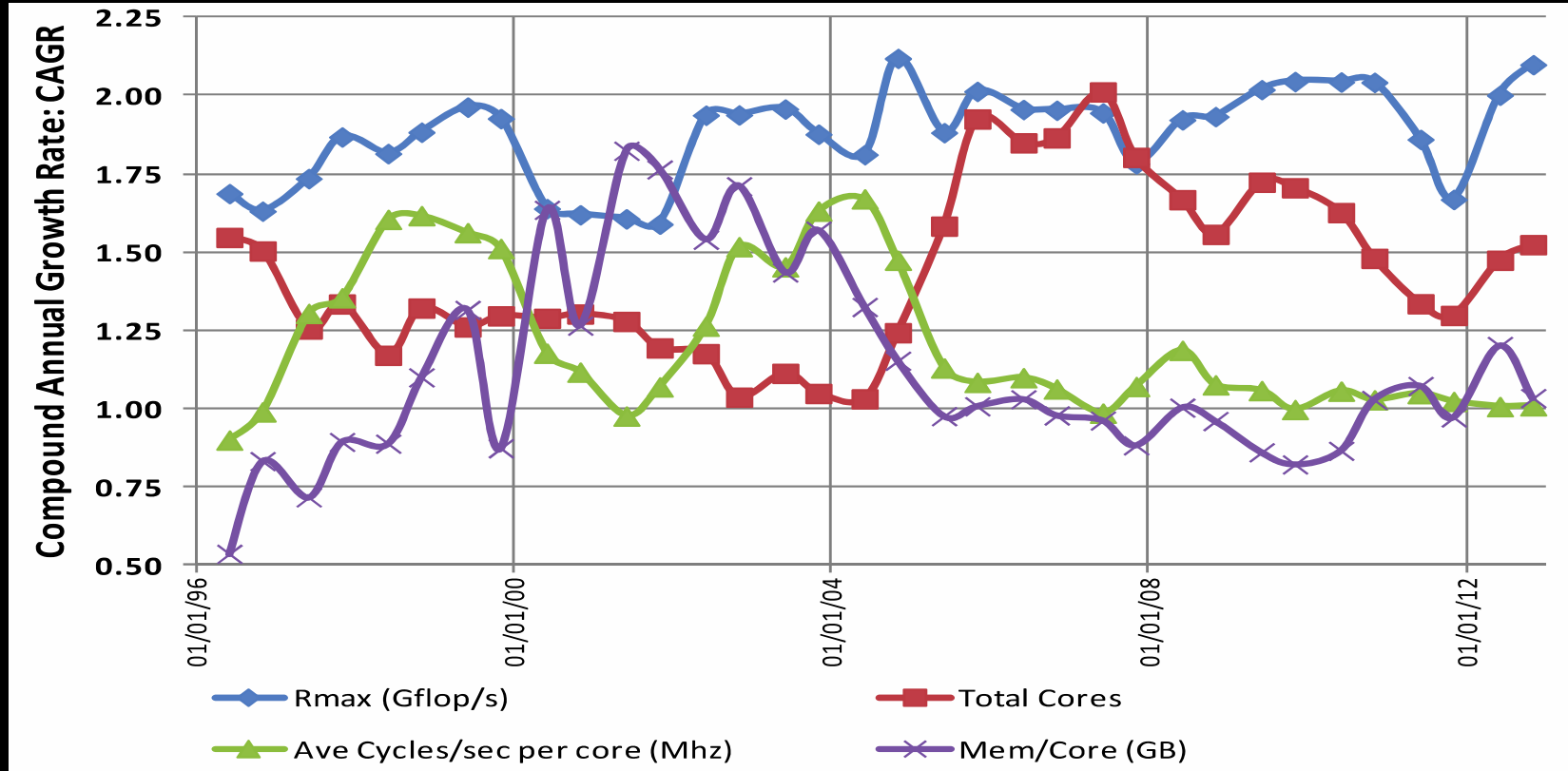
- Pflops computing fully established with more than 50 machines
- The field is thriving
- Interest in supercomputing is now worldwide, and growing in many new markets
- Exascale projects in many countries and regions



# Projected Performance Development



# Trends with ends.





# Two Additional Boosts to Improve Flops/Watt and Reach Exascale Target

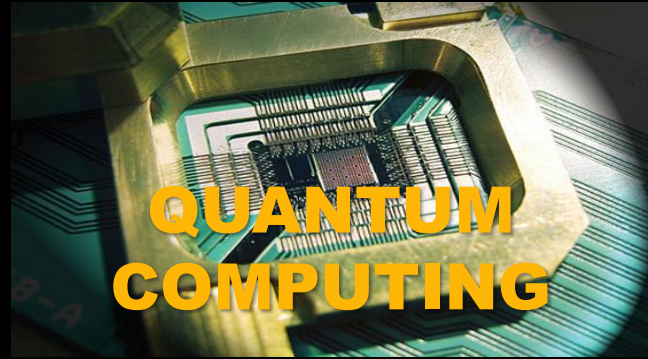


# End of Moore's Law Will Lead to New Architectures

Non-von  
Neumann



ARCHITECTURE



von Neumann



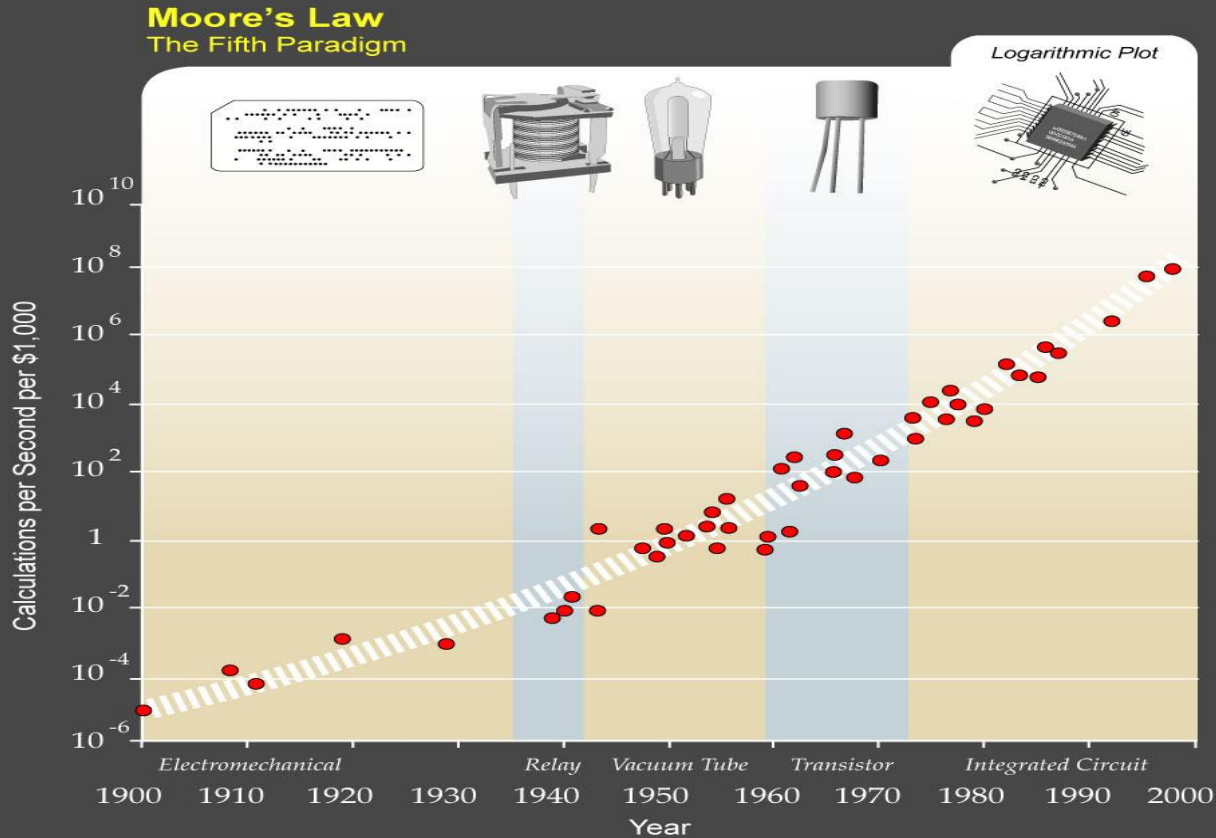
CMOS



Beyond CMOS

TECHNOLOGY

# It would only be the 6<sup>th</sup> paradigm.



# **We can do better. We have a role model.**

- **Straight forward extrapolation results in a real time human brain scale simulation at about 1 - 10 Exaflop/s with 4 PB of memory**
- **Current predictions envision Exascale computers in 2022+ with a power consumption of at best 20 - 30 MW**
- **The human brain takes 20W**
- **Even under best assumptions in 2020 our brain will still be a million times more power efficient**





# The Future and where you fit.

While the need is great, there is only a short list of serious contenders for 2020 exascale computing usability.

**MPI 3.0 +X** (MPI 3.0 specifically addresses exascale computing issues)

**PGAS** (partitioned global address space)

CAF (now in Fortran 2008!), UPC

**APGAS**

**X10, Chapel**

**Frameworks**

**Charm++**

**Functional**

**Haskell**

# Appendix

**Slides I had to ditch in the interest of time. However they are worthy of further discussion in this gathering. If some topic here catches your eye, or your ire, there are people all around you with related knowledge. I am also now an identified target.**

# Credits

- Horst Simon of LBNL
  - His many beautiful graphics are a result of his insightful perspectives
  - He puts his money where his mouth is: \$2000 bet in 2012 that Exascale machine would not exist by end of decade
- Intel
  - Many datapoints flirting with NDA territory
- Top500.org
  - Data and tools
- Supporting cast:

Erich Strohmaier (LBNL)

Jack Dongarra (UTK)

Rob Leland (Sandia)

John Shalf (LBNL)

Scott Aronson (MIT)

Bob Lucas (USC-ISI)

John Kubiatawicz (UC Berkeley)

Dharmendra Modha and team(IBM)

Karlheinz Meier (Univ. Heidelberg)



# Exascale?

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

23,800 X



Cray Red Storm  
2004  
42 Tflops

or

833,000 X



NVIDIA K40  
1.2 Tflops

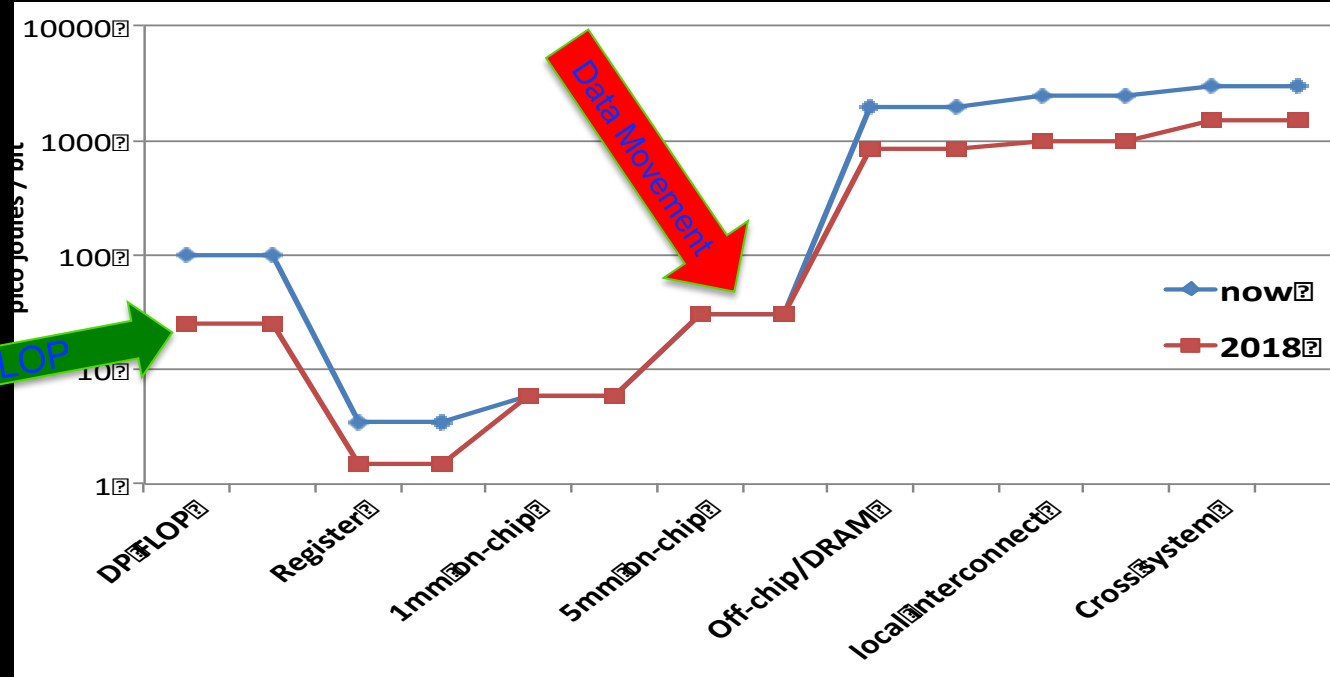
# Obstacles?

One of the many groups established to enable this outcome (the Advanced Scientific Computing Advisory Committee) puts forward this list of 10 technical challenges.

- Energy efficient circuit, power and cooling technologies.
- High performance interconnect technologies.
- Advanced memory technologies to dramatically improve capacity and bandwidth.
- Scalable system software that is power and resilience aware.
- Data management software that can handle the volume, velocity and diversity of data-storage
- Programming environments to express massive parallelism, data locality, and resilience.
- Reformulating science problems and refactoring solution algorithms for exascale.
- Ensuring correctness in the face of faults, reproducibility, and algorithm verification.
- Mathematical optimization and uncertainty quantification for discovery, design, and decision.
- Software engineering and supporting structures to enable scientific productivity.

# Power Issues by 2018

FLOPs will cost less than on-chip data movement!



# Flops are free?

At exascale, >99% of power is consumed by moving operands across machine.

Does it make sense to focus on flops, or should we optimize around data movement?

To those that say the future will simply be Big Data:

*“All science is either physics or stamp collecting.”*

*- Ernest Rutherford*

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

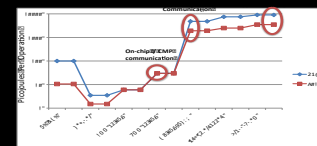
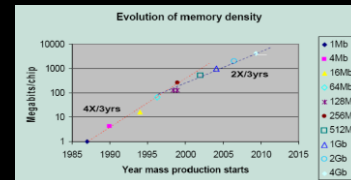
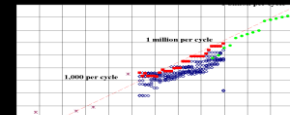
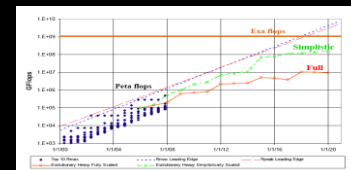
*Current programming systems have WRONG optimization targets*

## Old Constraints

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

## New Constraints

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



*Fundamentally breaks our current programming paradigm and computing ecosystem*

# As a last resort, we ~~could~~ will learn to program again.

It has become a mantra of contemporary programming philosophy that developer hours are so much more valuable than hardware, that the best design compromise is to throw more hardware at slower code.

This might well be valid for some Java dashboard app used twice week by the CEO. But this has spread and results in...

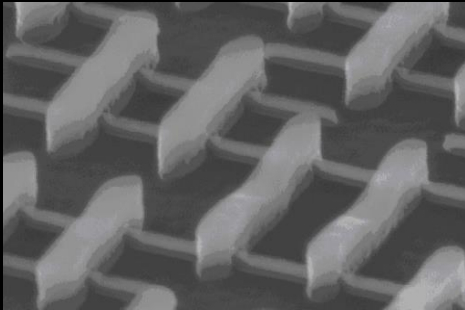
The common observation that a modern PC (or phone) seems to be more laggy than one from a few generations ago that had literally 1 thousandth the processing power.

Moore's Law has been the biggest enabler (or more accurately rationalization) for this trend. If Moore's Law does indeed end, then progress will require good programming.

No more garbage collecting, script languages. I am looking at you, Java, Python, Matlab,

# ...but our metrics are less clear.

After a while, “there was no one design rule that people could point to and say, ‘That defines the node name’ ... The minimum dimensions are getting smaller, but I’m the first to admit that I can’t point to the one dimension that’s 32 nm or 22 nm or 14 nm. Some dimensions are smaller than the stated node name, and others are larger.”



**Mark Bohr, Senior fellow at Intel.**

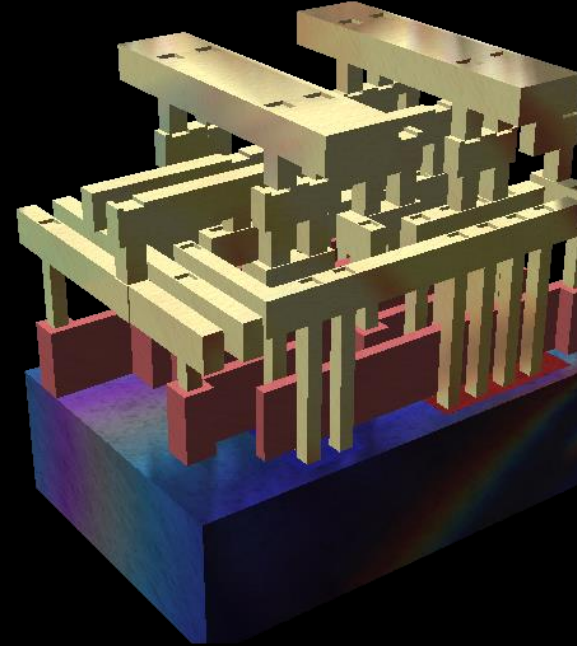
*From The Status of Moore's Law: It's Complicated (IEEE Spectrum)*

# For a while things were better than they appeared.

Intel's 0.13- $\mu\text{m}$  chips, which debuted in 2001, had transistor gates that were actually just 70 nm long. Nevertheless, Intel called them 0.13- $\mu\text{m}$  chips because they were the next in line.

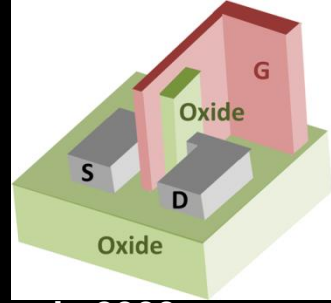
Manufacturers continued to pack the devices closer and closer together, assigning each successive chip generation a number about 70 percent that of the previous one.

A 30 percent reduction in both the x and y dimensions corresponds to a 50 percent reduction in the area occupied by a transistor, and therefore the potential to double transistor density on the chip.





# Then new technologies carried the load.



After years of aggressive gate trimming, simple transistor scaling reached a limit in the early 2000s: Making a transistor smaller no longer meant it would be faster or less power hungry. So Intel, followed by others, introduced new technologies to help boost transistor performance.

- Strain engineering, adding impurities to silicon to alter the crystal, which had the effect of boosting speed without changing the physical dimensions of the transistor.
- New insulating and gate materials.
- And most recently, they rejiggered the transistor structure to create the more efficient FinFET, with a current-carrying channel that juts out of the plane of the chip.

The switch to FinFETs has made the situation even more complex. Intel's 22-nm chips, the current state of the art, have FinFET transistors with gates that are 35 nm long but fins that are just 8 nm wide.

# Now tradeoffs are stealing these gains.

The density and power levels on a state-of-the-art chip have forced designers to compensate by adding:

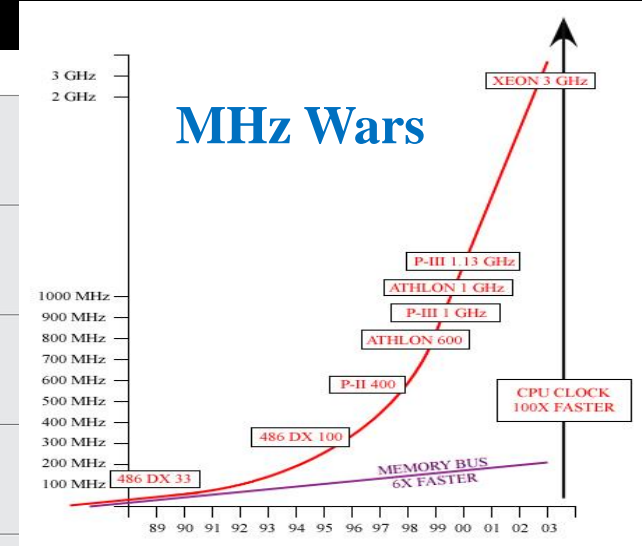
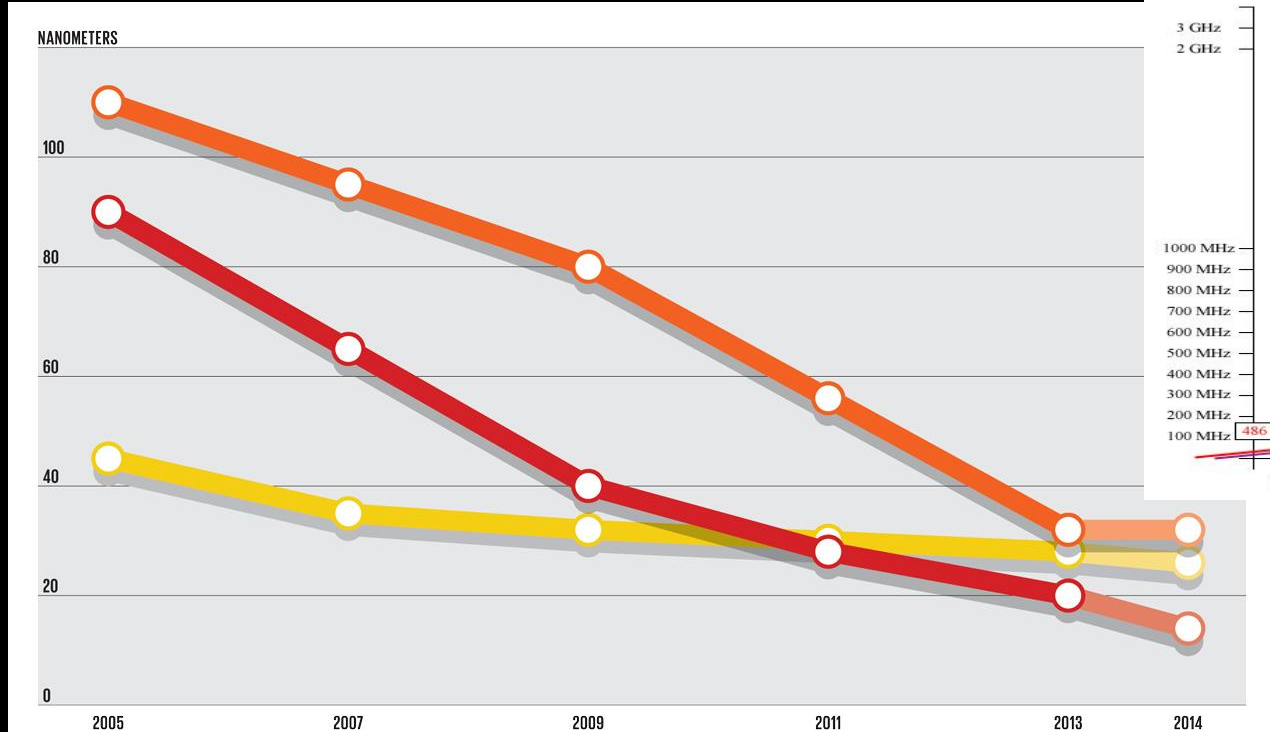
- error-correction circuitry
- redundancy
- read- and write-boosting circuitry for failing static RAM cells
- circuits to track and adapt to performance variations
- complicated memory hierarchies to handle multicore architectures.

All of those extra circuits add area. Some analysts have concluded that when you factor those circuits in, chips are no longer twice as dense from generation to generation. One such analysis suggests, the density improvement over the past three generations, from 2007 on, has been closer to 1.6 than 2.

And cost per transistor has gone up for the first time ever:

- 2012 20M 28nm transistors/dollar
- 2015 19M 16nm transistors/dollar

# Maybe they are even becoming “marketing”.



Global Foundries planned 2013 14nm chip introduction.

*As reported by IEEE Spectrum*

# How parallel is a code?

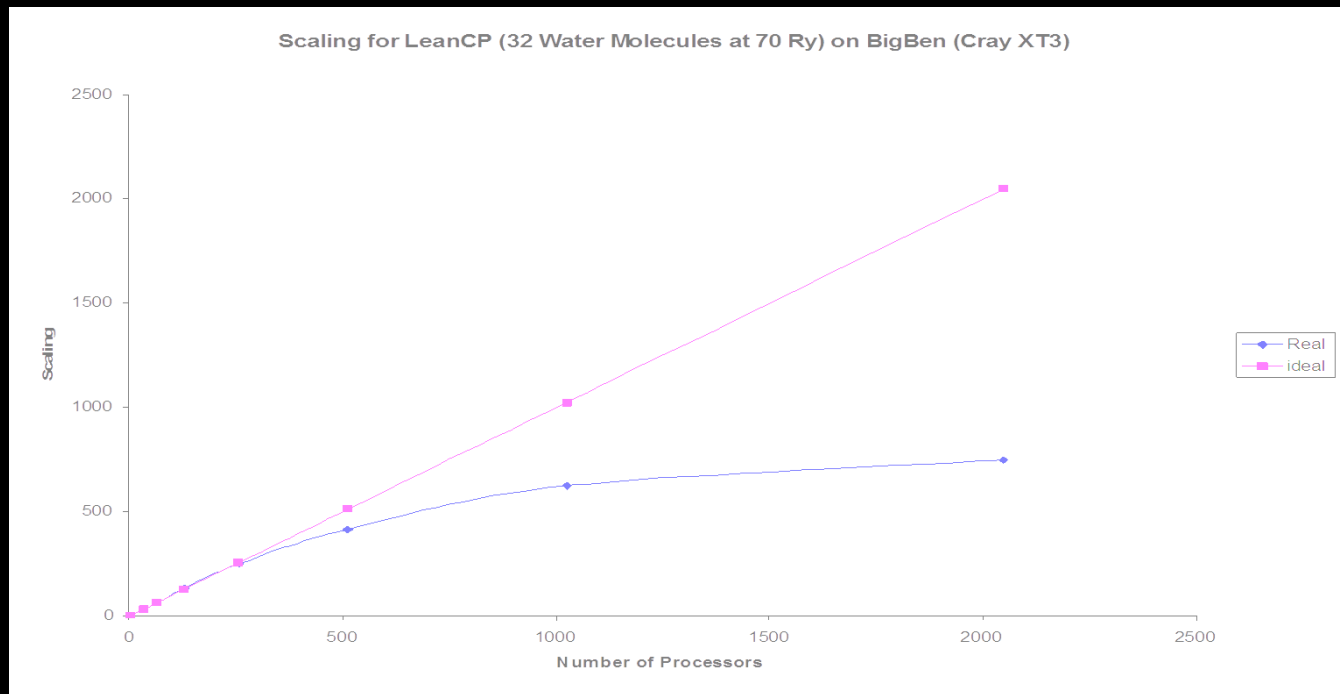
- Parallel performance is defined in terms of scalability

## Strong Scalability

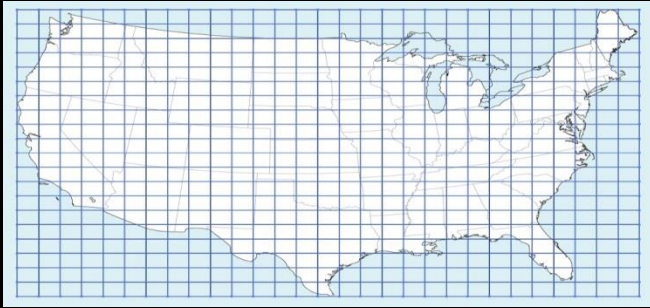
Can we get faster for a given problem size?

## Weak Scalability

Can we maintain runtime as we scale up the problem?

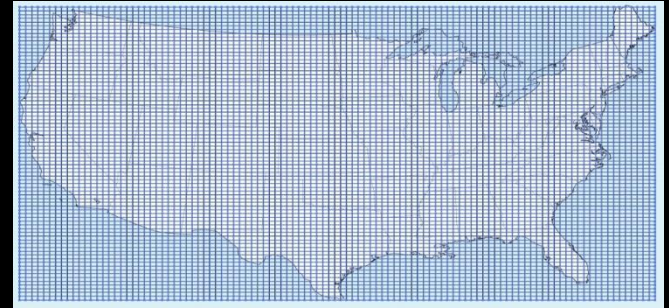


# Weak vs. Strong scaling

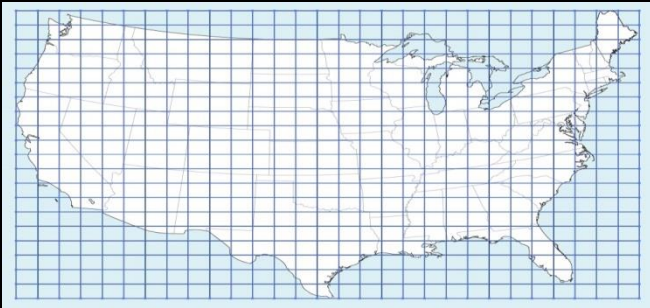


More  
Processors

Weak Scaling

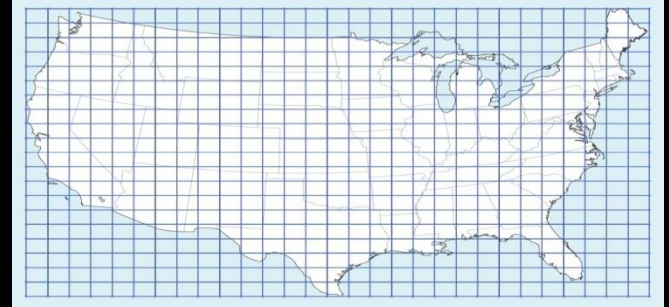


More accurate results



More  
Processors

Strong Scaling

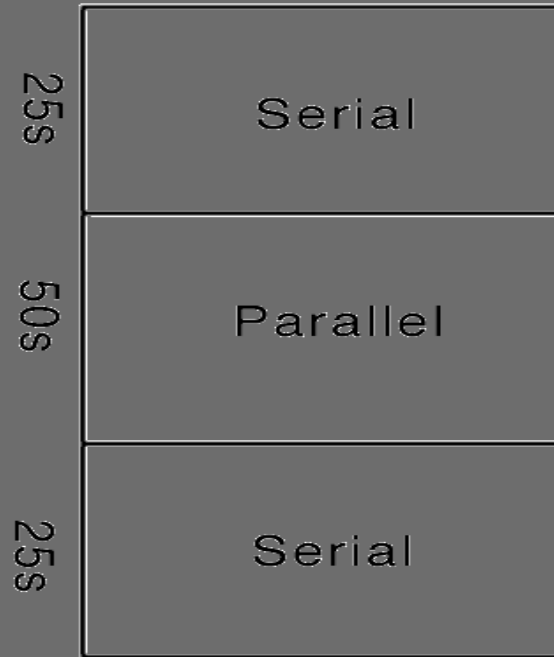


Faster results  
(Tornado on way!)

# Your Scaling Enemy: Amdahl's Law

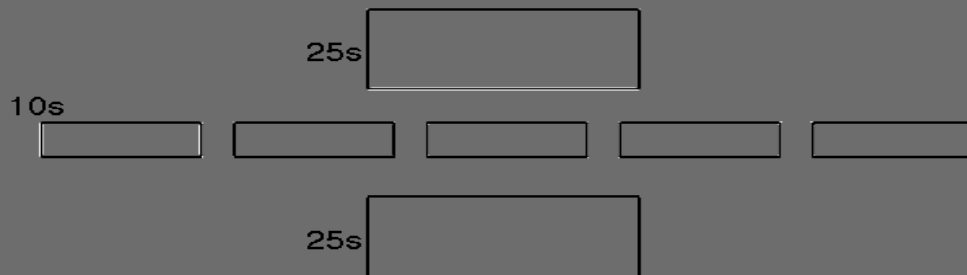
How many processors can we really use?

Let's say we have a legacy code such that it is only feasible to convert half of the heavily used routines to parallel:



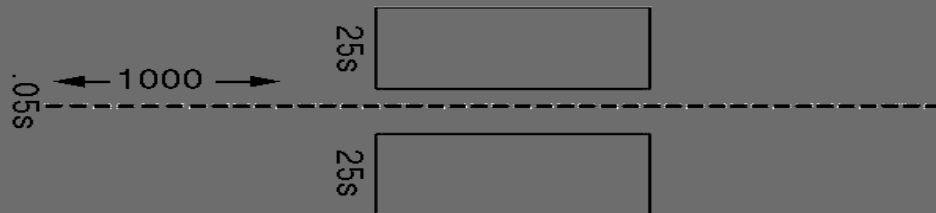
# Amdahl's Law

If we run this on a parallel machine with five processors:



Our code now takes about 60s.  
We have sped it up by about 40%.

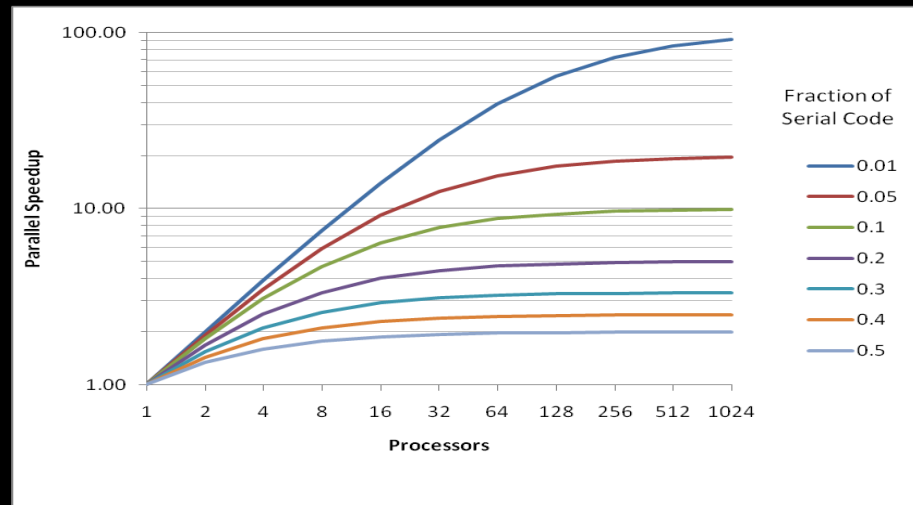
Let's say we use a thousand processors:



We have now sped our code by about a factor of two. Is this a big enough win?

# Amdahl's Law

- If there is  $x\%$  of serial component, speedup cannot be better than  $100/x$ .
- If you decompose a problem into many parts, then the parallel time cannot be less than the largest of the parts.
- If the critical path through a computation is  $T$ , you cannot complete in less time than  $T$ , no matter how many processors you use.





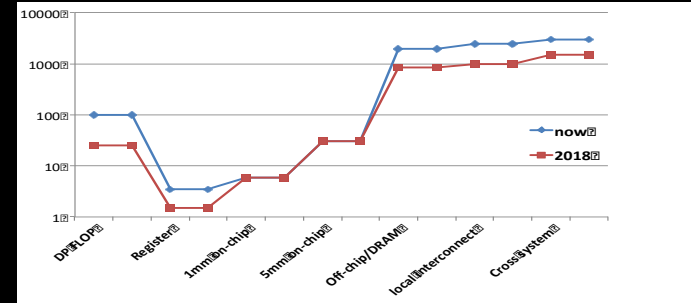
# MPI as an answer to an emerging problem ?!

In the Intro we mentioned that we are at a historic crossover where the cost of even on-chip data movement is surpassing the cost of computation.

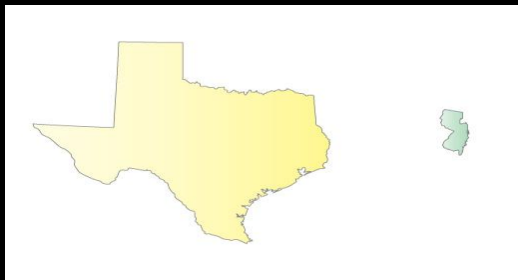
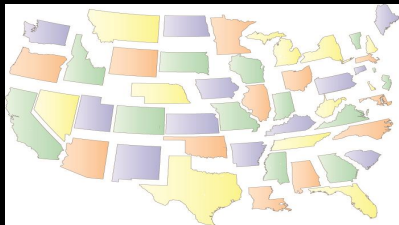
MPI codes explicitly acknowledge and manage data locality and movement (communication).

Both this paradigm, and quite possible outright MPI, offer the only immediate solution. You and your programs may find a comfortable future in the world of massive multi-core.

This is a somewhat recent realization in the most *avant-garde* HPC circles. Amaze your friends with your insightful observations!



# Domain Decomposition Done Well: Load Balanced



Is Texas vs. New Jersey a good idea?

- A parallel algorithm can only be as fast as the slowest chunk.
  - Might be dynamic (hurricane moving up coast)
- Communication will take time
  - Usually orders of magnitude difference between registers, cache, memory, network/remote memory, disk
  - Data locality and “neighborly-ness” matters very much.

