

Parallel Programming Methodologies

John Urbanic

Parallel Computing Scientist
Pittsburgh Supercomputing Center

Why this talk?

This is a version of a presentation I give at the *beginning* of the *International HPC Summer School* track.

Warning

This version
contains blunt editorializing.

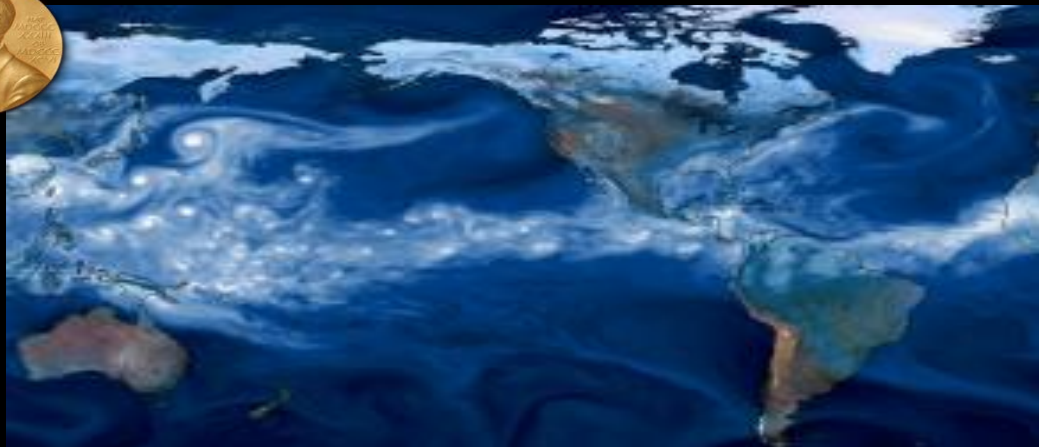
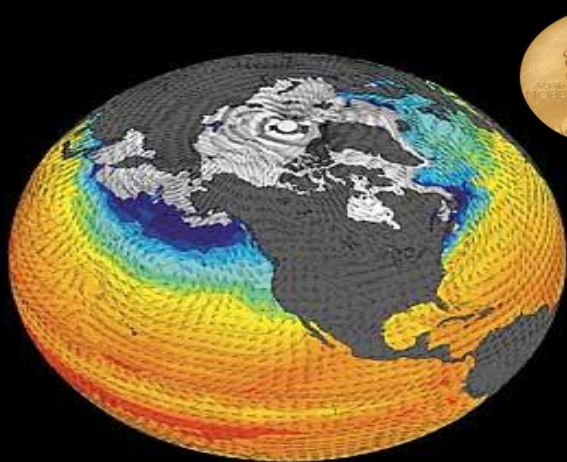


And help them see how the techniques they have learned best apply to their own applications.



at *Camp* to
the techniques

FLOPS we need: Climate change analysis



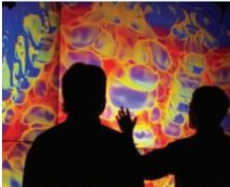
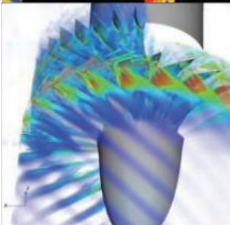

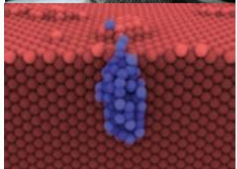
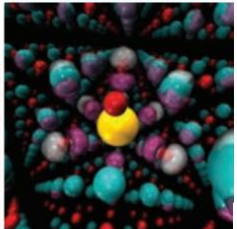



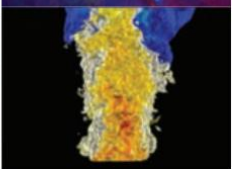
Simulations

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

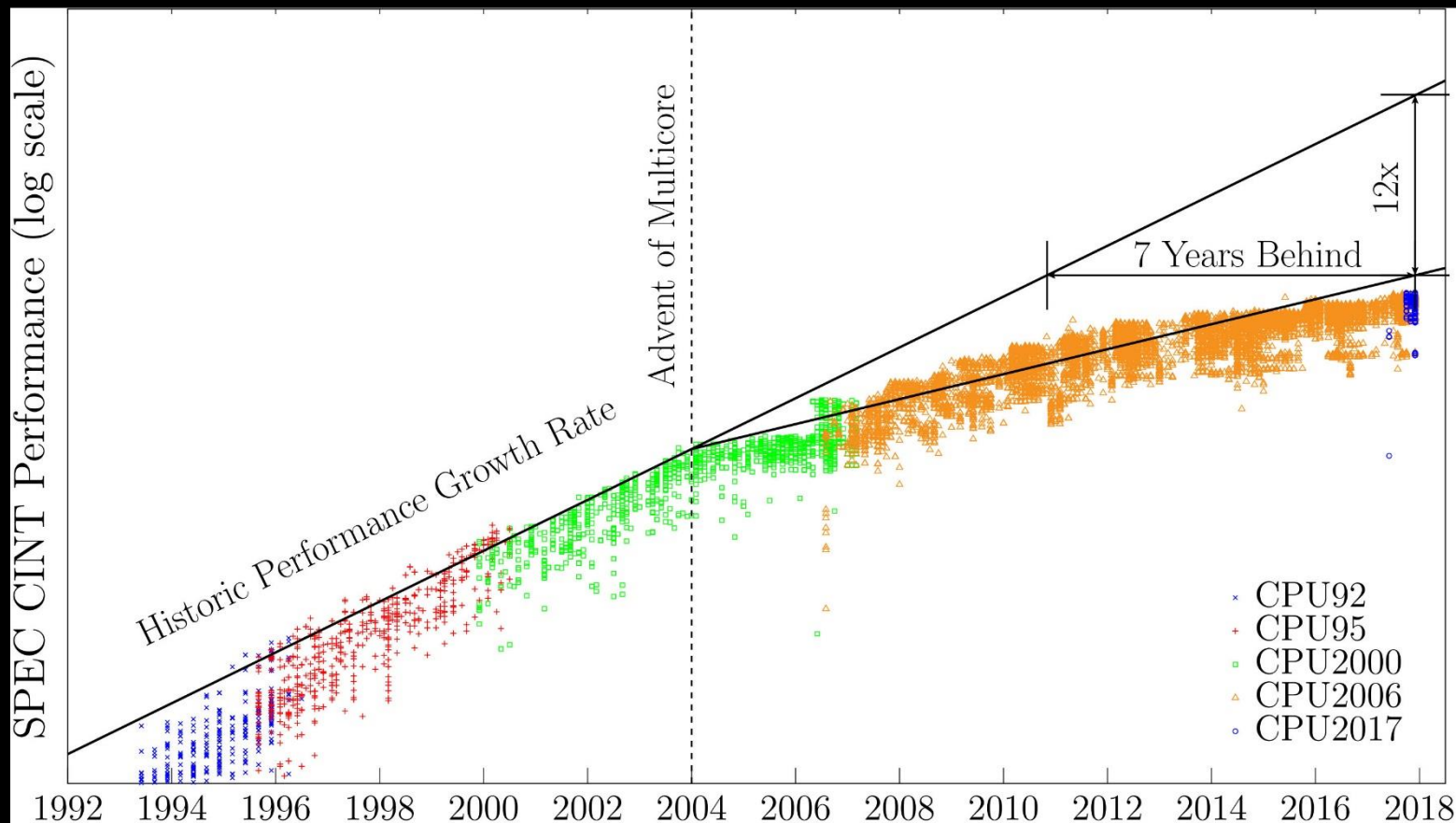
Extreme data

- “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
-

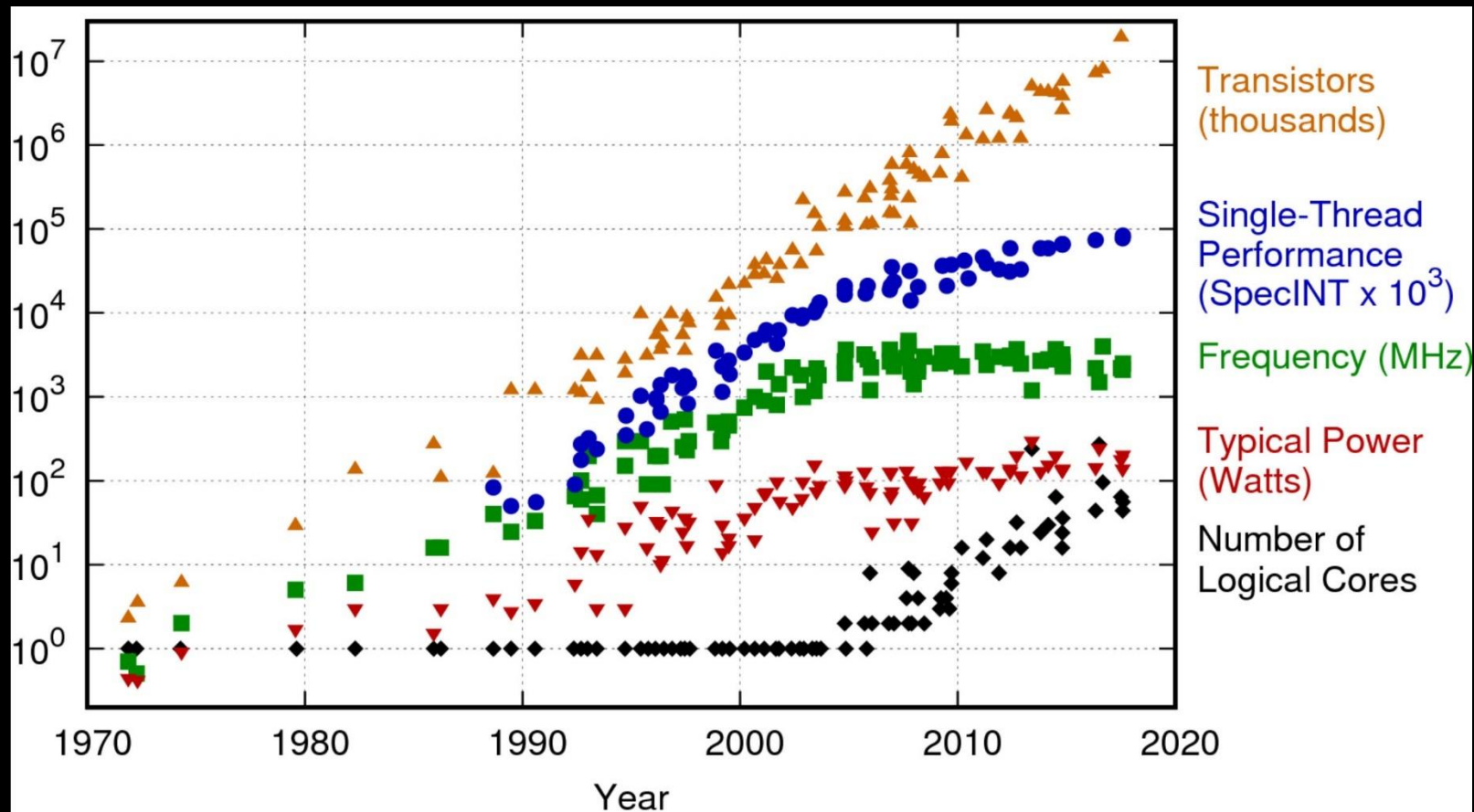
ECP application domains.

National security	Energy security	Economic security	Scientific discovery	Earth system	Health care
<p>Stockpile stewardship</p> <p>Next generation simulation tools for assessing nuclear weapons performance</p> <p>Response to hostile threat environments and reentry conditions</p>	<p>Turbine wind plant efficiency</p> <p>High-efficiency, low-emission combustion engine and gas turbine design</p> <p>Materials design for extreme environments of nuclear fission and fusion reactors</p> <p>Design and commercialization of Small Modular Reactors</p>	<p>Additive manufacturing of qualifiable metal parts</p> <p>Reliable and efficient planning of the power grid</p> <p>Seismic hazard risk assessment</p>	<p>Find, predict, and control materials and properties</p> <p>Cosmological probe of the standard model of particle physics</p> <p>Validate fundamental laws of nature</p> <p>Demystify origin of chemical elements</p> <p>Light source-enabled analysis of protein and molecular structure and design</p>	<p>Accurate regional impact assessments in Earth system models</p> <p>Stress-resistant crop analysis and catalytic conversion of biomass-derived alcohols</p> <p>Metagenomics for analysis of biogeochemical cycles, climate change, environmental remediation</p>	<p>Accelerate and translate cancer research</p>
 	<p>Subsurface use for carbon capture, petroleum extraction, waste disposal</p> <p>Scale-up of clean fossil fuel combustion</p> <p>Biofuel catalyst design</p>	 	<p>Whole-device model of magnetically confined fusion plasmas</p> 	 	 

Moore's Law abandoned serial programming around 2004



Come back tomorrow....



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

Single Socket Parallelism: On your desktop

Processor	Year	Vector	Bits	SP FLOPs / core / cycle	Cores	FLOPs/cycle
Pentium III	1999	SSE	128	3	1	3
Pentium IV	2001	SSE2	128	4	1	4
Core	2006	SSE3	128	8	2	16
Nehalem	2008	SSE4	128	8	10	80
Sandybridge	2011	AVX	256	16	12	192
Haswell	2013	AVX2	256	32	18	576
KNC	2012	AVX512	512	32	64	2048
KNL	2016	AVX512	512	64	72	4608
Skylake	2017	AVX512	512	96	28	2688

MPPs (Massively Parallel Processors)

Distributed memory at largest scale. Shared memory at lower level.

Summit (ORNL)

- 122 PFlops Rmax and 187 PFlops Rpeak
- IBM Power 9, 22 core, 3GHz CPUs
- 2,282,544 cores
- NVIDIA Volta GPUs
- EDR Infiniband



Sunway TaihuLight (NSC, China)

- 93 PFlops Rmax and 125 PFlops Rpeak
- Sunway SW26010 260 core, 1.45GHz CPU
- 10,649,600 cores
- Sunway interconnect



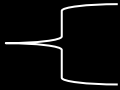
Many Levels and Types of Parallelism

- Vector (SIMD)
- Instruction Level (ILP)
 - Instruction pipelining
 - Superscaler (multiple instruction units)
 - Out-of-order
 - Register renaming
 - Speculative execution
 - Branch prediction

Compiler
(not your problem)

OpenMP 4/5
can help!

OpenMP



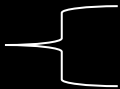
- Multi-Core (Threads)
- SMP/Multi-socket

OpenACC



- Accelerators: GPU & MIC

MPI



- Clusters
- MPPs

Also Important

- ASIC/FPGA/DSP
- RAID/IO

Parallel Computing

One woman can make a baby in 9 months.

Can 9 women make a baby in 1 month?

But 9 women can make 9 babies in 9 months.

Need to write some scalable code?

First Choice:

**Pick a language - or maybe a library, or paradigm
(whatever that is)?**

Languages: Pick One *(Hint: MPI + ?)*

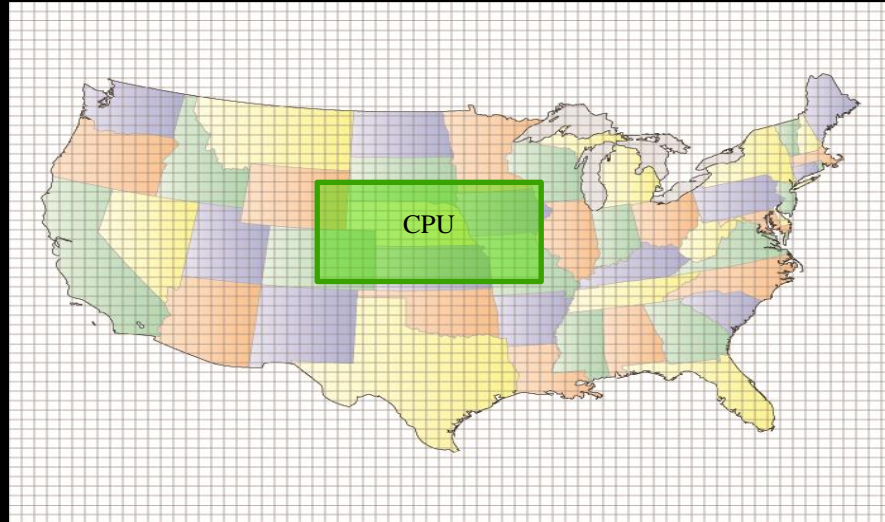
Parallel Programming environments since the 90's

ABCPL	CORRELATE	GLU	Mentat	Parafrase2	pC++
ACE	CPS	GUARD	Legion	Paralation	SCHEDULE
ACT++	CRL	HASL	Meta Chaos	Parallel-C++	SciTL
Active messages	CSP	Haskell	Midway	Parallaxis	POET
Adl	Cthreads	HPC++	Millipede	ParC	POEDA
Adsmith	CUMULVS	JAVAR	CparPar	ParLib++	SDDA
ADDAP					SUMEM
AFAPI					
ALWAN					
AM					
AMDC					
AppLeS					smalltalk
Amoeba					
ARTS					
Athapascan-0b					
Aurora					
Automap					
bb_threads					
Blaze					
BSP					
BlockComm					
C*					
"C*" in C					
C**					
CarlOS					
Cashmere					
C4					
CC++					
Chu					
Charlotte					
Charm					-NUS
Charm++					eads
Cid					
Cilk					
CM-Fortran					
Converse	EX	Lparx	Papers	Quick Threads	Quick Threads
Code	GA	Lucid	AFAPI	Sage++	XENOOPS
COOL	GAMMA	Maisie	Para++	SCANDAL	XPC
	Glenda	Manifold	Paradigm	SAM	Zounds
					ZPL

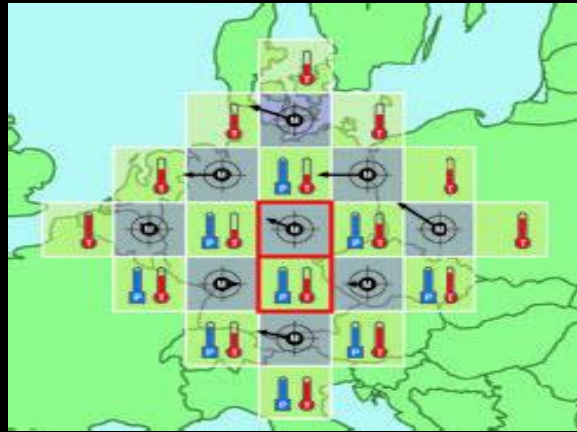
Alternative Approach

“When all you have is a hammer,
everything looks like a nail.”

Prototypical Application: Serial Weather Model

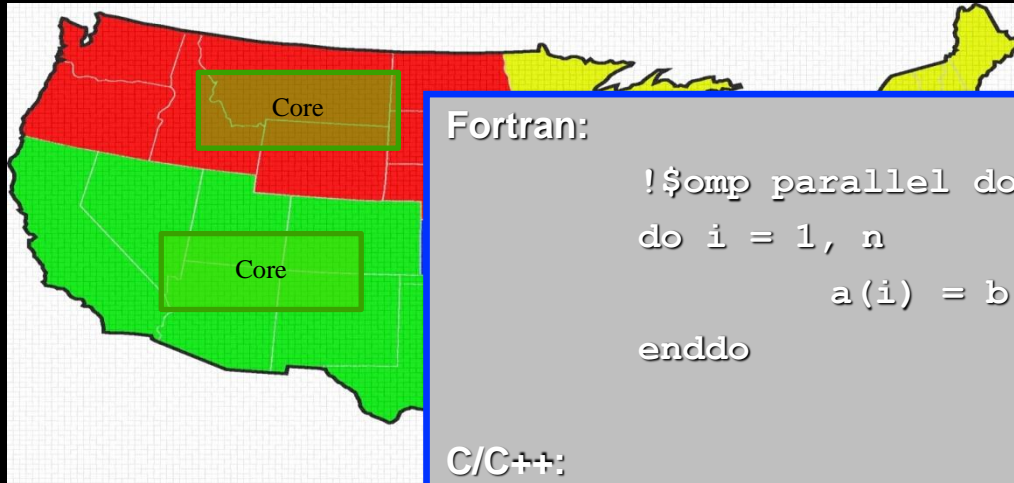


First parallel Weather Modeling algorithm: Richardson in 1917



Courtesy John Burkhardt, Virginia Tech

Weather Model: Shared Memory (OpenMP)



Fortran:

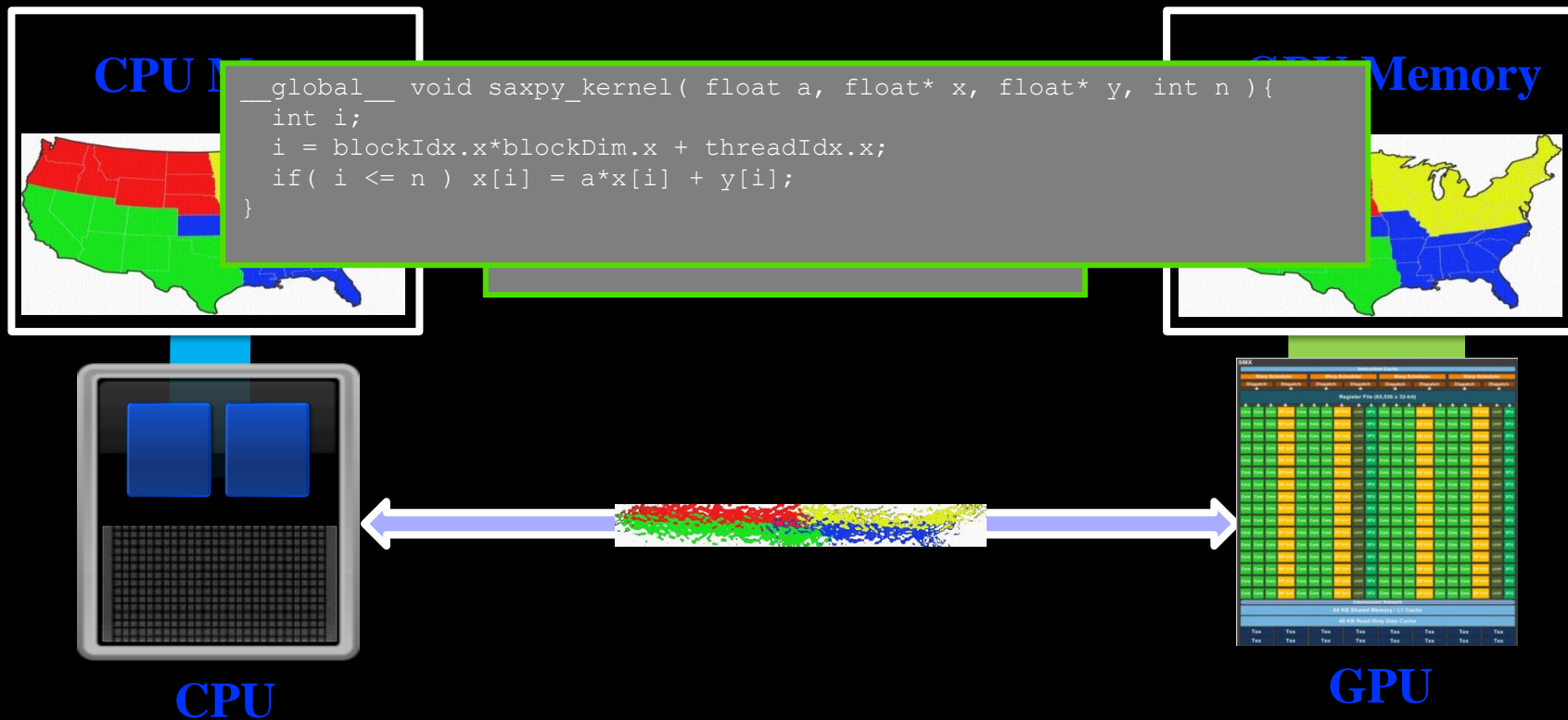
```
!$omp parallel do
do i = 1, n
        a(i) = b(i) + c(i)
enddo
```

C/C++:

```
#pragma omp parallel for
for(i=1; i<=n; i++)
        a[i] = b[i] + c[i];
```

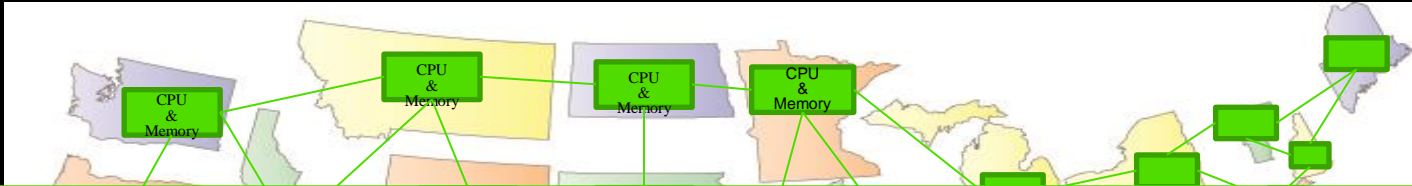
Four meteorologists in the

Weather Model: Accelerator (OpenACC)



1 meteorologists coordinating 1000 math savants using tin cans and a string.

Weather Model: Distributed Memory (MPI)



```
call MPI_Send( numbertosend, 1, MPI_INTEGER, index, 10, MPI_COMM_WORLD, errcode)
```

·
·

```
call MPI_Recv( numbertoreceive, 1, MPI_INTEGER, 0, 10, MPI_COMM_WORLD, status, errcode)
```

·
·
·

```
call MPI_Barrier(MPI_COMM_WORLD, errcode)
```

·



50 meteorologists using a telegraph.

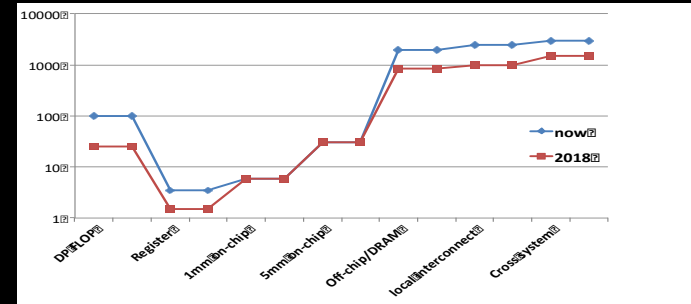
MPI as an answer to an emerging problem ?!

We are at a historic crossover where the cost of even on-chip data movement is surpassing the cost of computation.

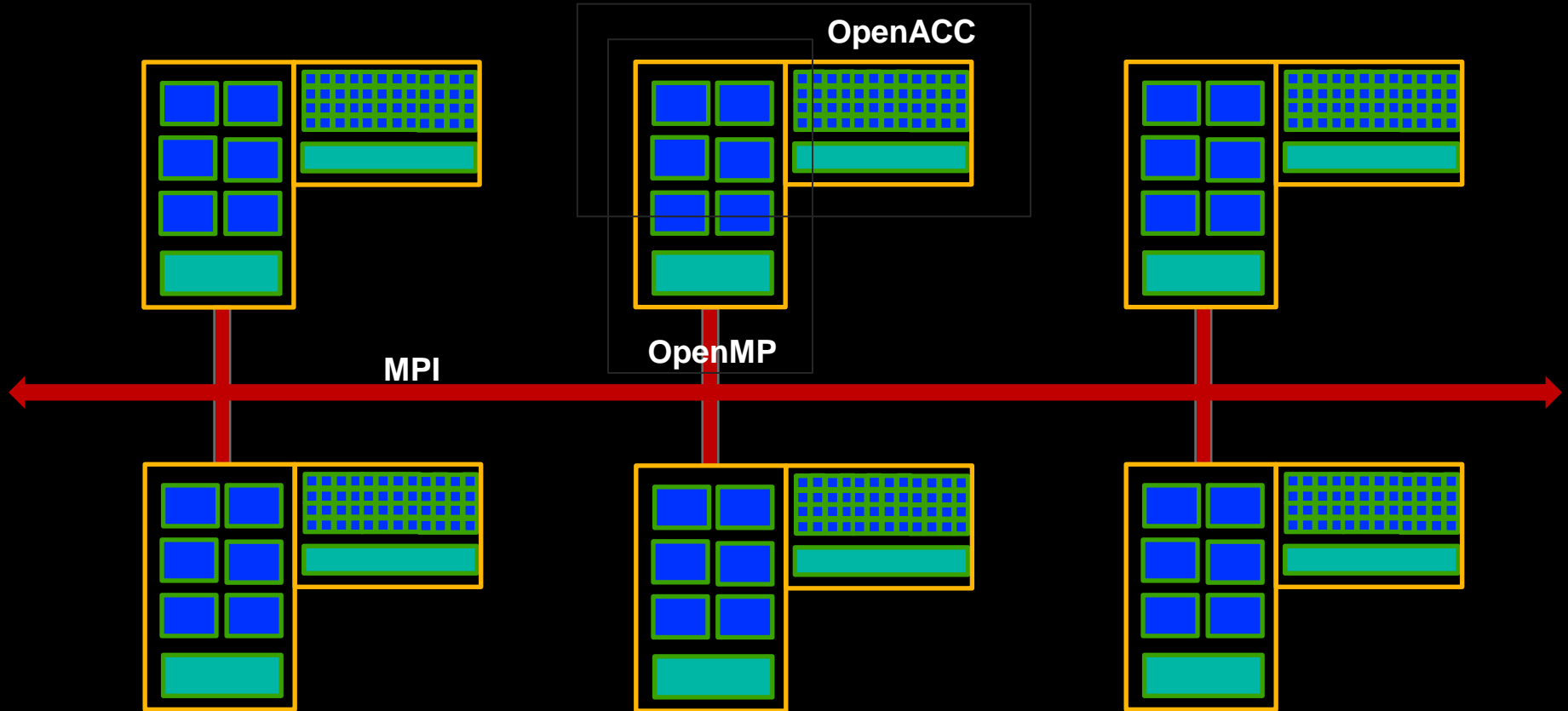
MPI codes explicitly acknowledge and manage data locality and movement (communication).

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!



The pieces fit like this...



#	Site	Manufacturer	Computer	CPU Interconnect [Accelerator]	Cores	Rmax (Tflops)	Rpeak (Tflops)	Power (MW)
1	RIKEN Center for Computational Science Japan	Fujitsu	Fugaku	ARM 8.2A+ 48C 2.2GHz Torus Fusion Interconnect	7,299,072	415,530	513,854	28.3
2	DOE/SC/ORNL United States	IBM	Summit	Power9 22C 3.0 GHz Dual-rail Infiniband EDR NVIDIA V100	2,414,592	148,600	200,794	10.1
3	DOE/NNSA/LLNL United States	IBM	Sierra	Power9 3.1 GHz 22C Infiniband EDR NVIDIA V100	1,572,480	94,640	125,712	7.4
4	National Super Computer Center in Wuxi China	NRCPC	Sunway TaihuLight	Sunway SW26010 260C 1.45GHz	10,649,600	93,014	125,435	15.3
5	National Super Computer Center in Guangzhou China	NUDT	Tianhe-2 (MilkyWay-2)	Intel Xeon E5-2692 2.2 GHz TH Express-2 Intel Xeon Phi 31S1P	4,981,760	61,444	100,678	18.4
6	Eni S.p.A Italy	Dell	HPc5	Xeon 24C 2.1 GHz Infiniband HDR NVIDIA V100	669,760	35,450	51,720	2.2
7	Eni S.p.A Italy	NVIDIA	Selene	EPYC 64C 2.25GHz Infiniband HDR NVIDIA A100	272,800	27,580	34,568	1.3
8	Texas Advanced Computing Center/Univ. of Texas United States	Dell	Frontera	Intel Xeon 8280 28C 2.7 GHz InfiniBand HDR	448,448	23,516	38,745	
9	Cineca Italy	IBM	Marconi100	Power9 16C 3.0 GHz Infiniband EDR NVIDIA V100	347,776	21,640	29,354	1.5
10	Swiss National Supercomputing Centre (CSCS) Switzerland	Cray	Piz Daint Cray XC50	Xeon E5-2690 2.6 GHz Aries NVIDIA P100	387,872	21,230	27,154	2.4

Other Paradigms?

- ✓ ● **Message Passing**
 - MPI
- ✓ ● **Threads**
 - OpenMP, OpenACC, CUDA
- ✓ ● **Hybrid**
 - MPI + OpenMP
- **Data Parallel**
 - Fortran90
- **PGAS (Partitioned Global Address Space)**
 - UPC, Coarray Fortran (Fortran 2008)
- **Frameworks**
 - Charm++

Data Parallel – Fortran90

Computation in FORTRAN 90

P	P	P	P
P	P	P	P
P	P	P	P
P	P	P	P

```
Real A(4,4), B(4,4), C(4,4)
```

```
A=2.0
```

```
FORALL (I=1:4, J=1:4)
```

```
  B(I, J)=I+J
```

```
C=A+B
```

P = Processor

C=

A

+

B

4	5	6	7
5	6	7	8
6	7	8	9
7	8	9	10

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

2	3	4	5
3	4	5	6
4	5	6	7
5	6	7	8

Data Parallel

Communication in FORTRAN 90

P	P	P	P
P	P	P	P
P	P	P	P
P	P	P	P

P = Processor

```
Real A(4,4), B(4,4)
```

```
FORALL (I=1:4, J=1:4)
```

```
  B(I,J)=I+J
```

```
A=CSHIFT (B, DIM=2,1)
```

A=

3	4	5	6
4	5	6	7
5	6	7	8
2	3	4	5

CSHIFT (B,2,1)

2	3	4	5
3	4	5	6
4	5	6	7
5	6	7	8

Data Parallel

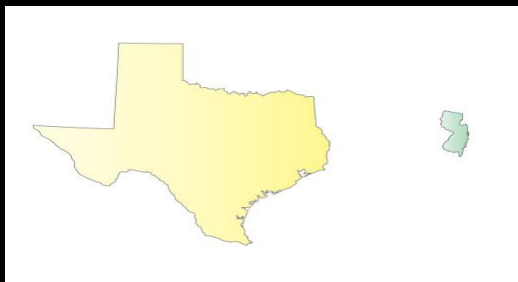
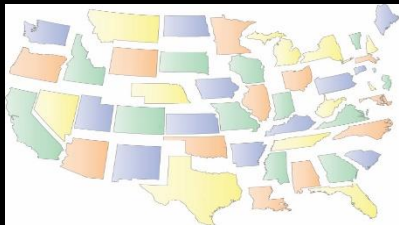
Pros

- So simple you just learned some of it
- ...or already knew it from using Fortran
- Easy to debug

Cons

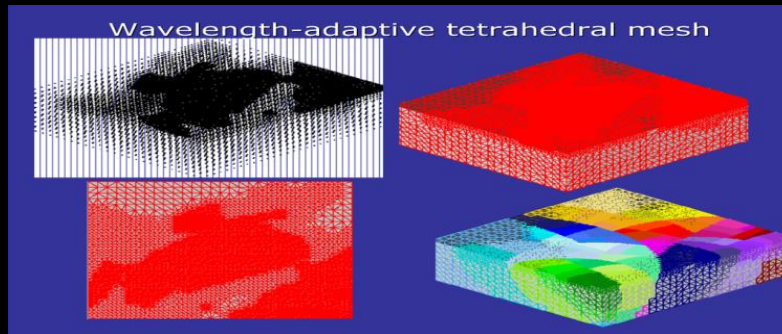
- If your code doesn't totally, completely express itself as nice array operations, you are left without a flexible alternative.
 - Image processing: Great
 - Irregular meshes: Not so great

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.



PGAS with Co-Array Fortran (now Fortran 2008)

Co-array synchronization is at the heart of the typical Co-Array Fortran program. Here is how to exchange an array with your north and south neighbors:

```
COMMON/XCTILB4/ B(N,4) [*]  
SAVE /XCTILB4/  
  
CALL SYNC_ALL( WAIT=(/IMG_S,IMG_N/) )  
B(:,3) = B(:,1) [IMG_S]  
B(:,4) = B(:,2) [IMG_N]  
CALL SYNC_ALL( WAIT=(/IMG_S,IMG_N/) )
```

Lots more examples at co-array.org.

Partitioned Global Address Space: (PGAS)

Multiple threads share at least a part of a global address space.

Can access local and remote data with same mechanisms.

Can distinguish between local and remote data with some sort of typing.

Variants:

Co-Array Fortran (CAF)

Fortran 2008

Unified Parallel C (UPC)

Pros:

1. Looks like SMP on a distributed memory machine.
2. Currently translates code into an underlying message passing version for efficiency.

Cons:

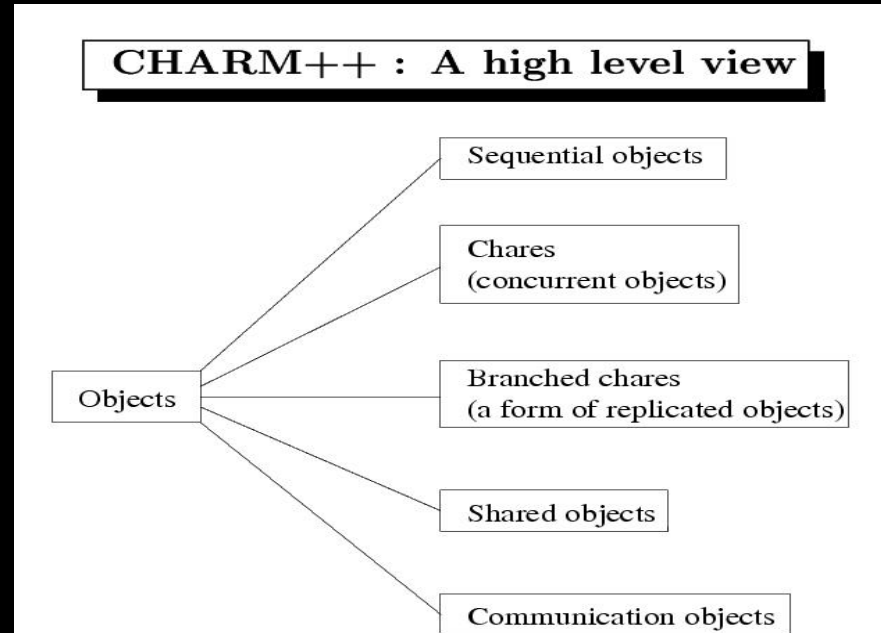
1. Depends on (2) to be efficient.
2. Can easily write lots of expensive remote memory access without paying attention.
3. Currently immature.

Frameworks

One of the more experimental approaches that was gaining some traction was to use a parallel framework that handle the load balancing and messaging while you “fill in” the science. Charm++ is the most popular example:

Charm++

- Object-oriented parallel extension to C++
- Run-time engine allows work to be “scheduled” on the computer.
- Highly-dynamic, extreme load-balancing capabilities.
- Completely asynchronous.
- NAMD, a very popular MD simulation engine is written in Charm++



Frameworks (Newsflash!)

- After a long time with no positive reports in this space, I can definitely say that the Machine Learning (Artificial Intelligence) community has embraced this in an effective manner.
- The most popular frameworks/toolkits/packages used for deep learning (aka Neural Nets) are very much in this philosophy.
- Caffe, TensorFlow and others use a high level descriptive approach to arrange other components, often themselves a higher level layer in Python or whatnot, to invoke libraries written in C++ (and actually Fortran is hidden in there more often than those groups would believe in the form of BLAS type libraries).
- These frameworks use threads, GPUs and distributed nodes very heavily.
- You could say that the math library nature of this work makes this unique, but the innovation in arranging these codeflows is not at all rote.

Some Alternatives

- **OpenCL (Khronos Group)**
 - Everyone supports, but not as a primary focus
 - Intel – OpenMP
 - NVIDIA – CUDA, OpenACC
 - AMD – now HIP
 - Khronos has now brought out SYCL
- **Fortran 2008+ threads (sophisticated but not consistently implemented)**
- **C++11 threads are basic (no loops) but better than POSIX**
 - C++17 brings parallel STL
 - C++20 atomic smart pointers, futures, latches and barriers, coroutines, transactional memory, task blocks
- **Python threads are fake (due to Global Interpreter Lock) but multiprocessing is pretty easy.**
- **DirectCompute (Microsoft) is not HPC oriented**
- **C++ AMP (MS/AMD)**
- **TBB (Intel C++ template library)**
- **Cilk (Intel, now in a gcc branch)**
- **Intel oneAPI (Includes DPC++ and extends SYCL)**
- **Kokkos**

How parallel is a code?

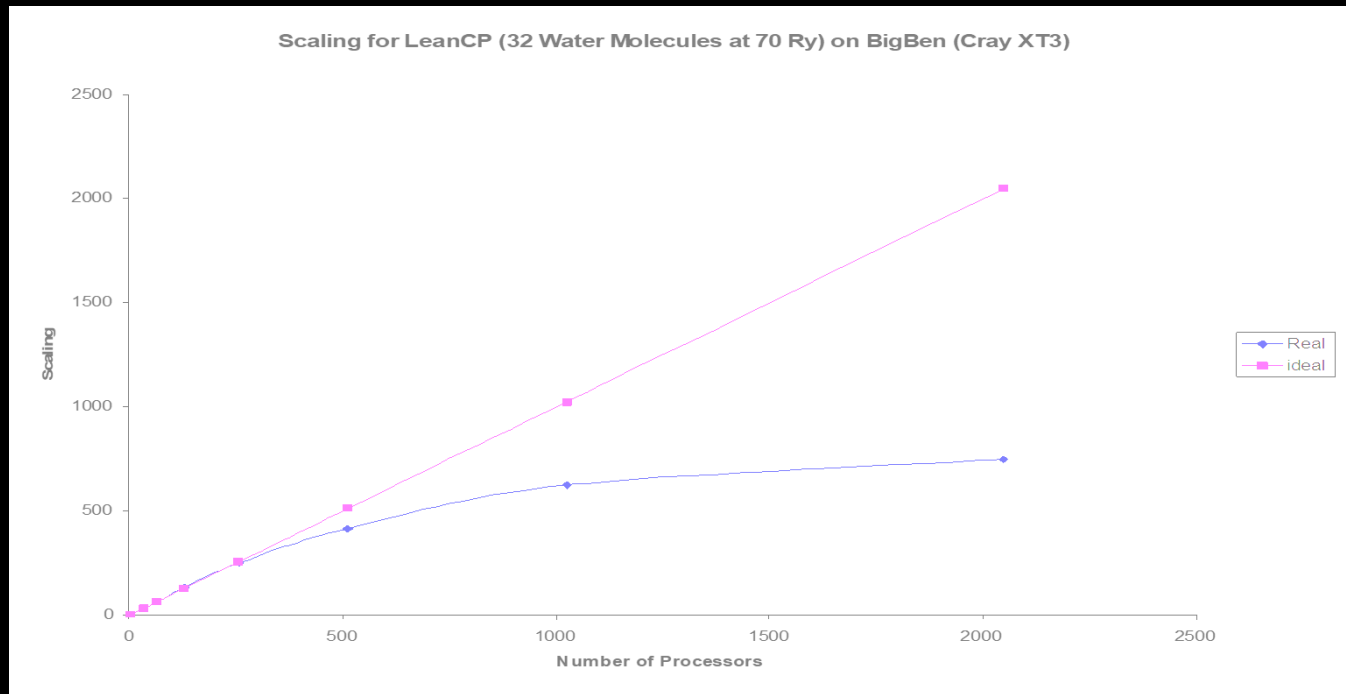
- Parallel performance is defined in terms of scalability

Strong Scalability

Can we get faster for a given problem size?

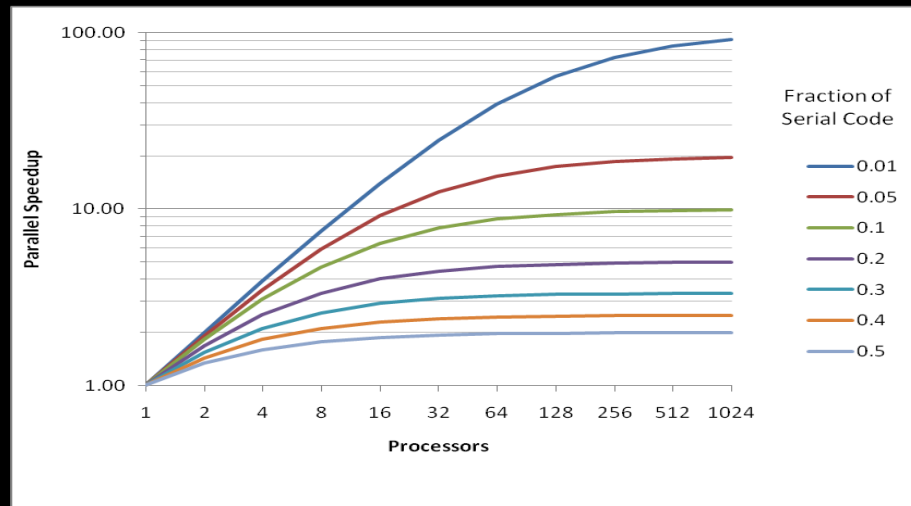
Weak Scalability

Can we maintain runtime as we scale up the problem?



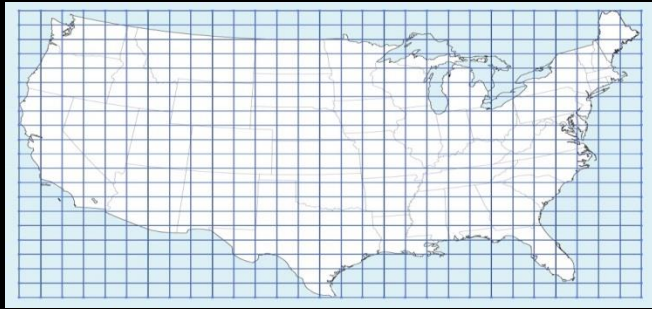
Amdahl's Law

- 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 .



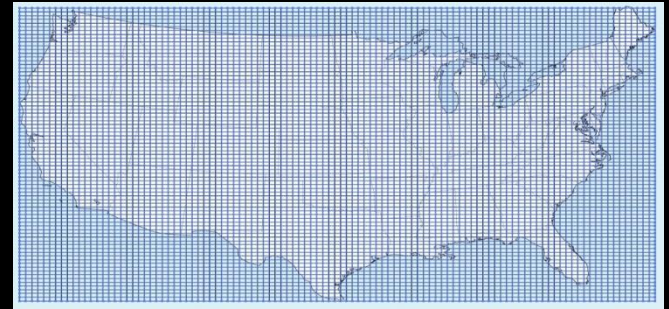
- Amdahl's law used to be cited by the knowledgeable as a limitation.
- These days it is mostly raised by the uninformed.
- Massive scaling is commonplace:
 - Science Literature
 - Web (map reduce everywhere)
 - Data Centers (Spark, etc.)
 - Machine Learning (GPUs and others)

Weak vs. Strong scaling

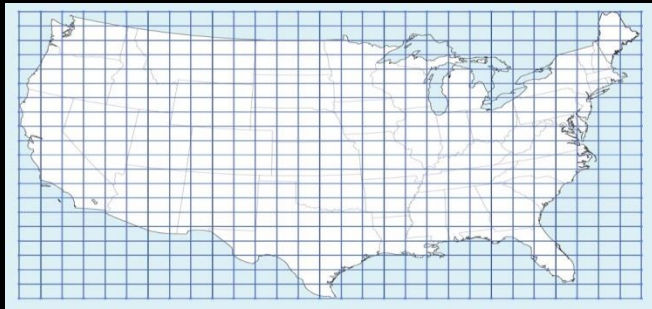


More
Processors

Weak Scaling

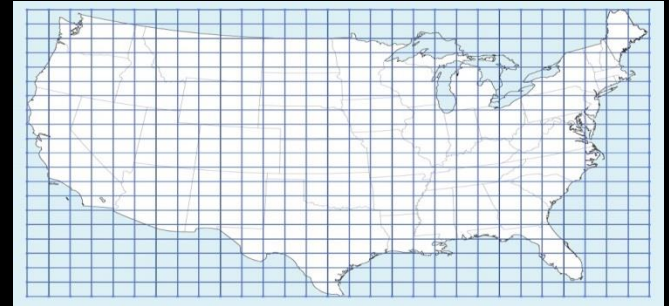


More accurate results



More
Processors

Strong Scaling

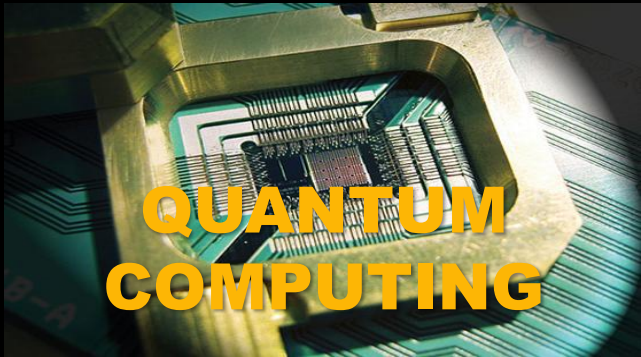


Faster results
(Tornado on way!)



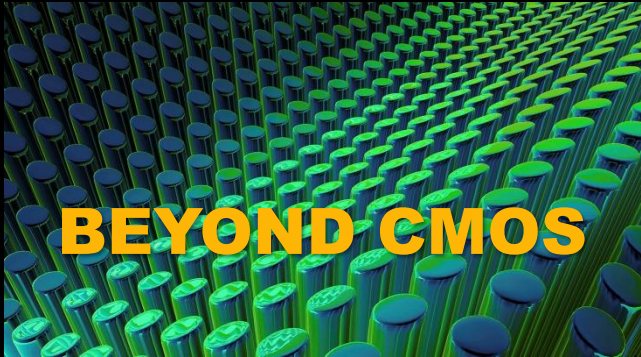
End of Moore's Law Will Lead to New Architectures

Non-von
Neumann



ARCHITECTURE

von Neumann



CMOS

Beyond CMOS

TECHNOLOGY

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)

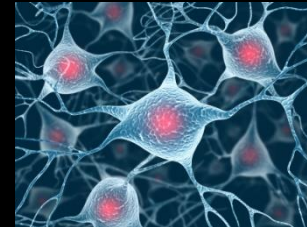
CAF (now in Fortran 2008!). **UPC**

What about Big Data?



Deep Learning?

A Different Talk!
Perhaps next year?



Again...

