It’s a Multicore World

John Urbanic
Pittsburgh Supercomputing Center
Parallel Computing Scientist
Moore's Law abandoned serial programming around 2004
Moore’s Law is not to blame.

Intel process technology capabilities

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Size</td>
<td>90nm</td>
<td>65nm</td>
<td>45nm</td>
<td>32nm</td>
<td>22nm</td>
<td>16nm</td>
<td>11nm</td>
<td>8nm</td>
</tr>
<tr>
<td>Integration Capacity (Billions of Transistors)</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
</tr>
</tbody>
</table>

Transistor for 90nm Process
Source: Intel

Influenza Virus
Source: CDC
At end of day, we keep using all those new transistors.
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
And how to get more performance from more transistors with the same power.

**RULE OF THUMB**

<table>
<thead>
<tr>
<th>Frequency Reduction</th>
<th>Power Reduction</th>
<th>Performance Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td>45%</td>
<td>10%</td>
</tr>
</tbody>
</table>

A 15% Reduction In Voltage Yields

**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

**Frequency Reduction**

15%

**Power Reduction**

45%

**Performance Reduction**

10%
<table>
<thead>
<tr>
<th>Processor</th>
<th>Year</th>
<th>Vector</th>
<th>Bits</th>
<th>SP FLOPs / core / cycle</th>
<th>Cores</th>
<th>FLOPs/cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium III</td>
<td>1999</td>
<td>SSE</td>
<td>128</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Pentium IV</td>
<td>2001</td>
<td>SSE2</td>
<td>128</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Core</td>
<td>2006</td>
<td>SSE3</td>
<td>128</td>
<td>8</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Nehalem</td>
<td>2008</td>
<td>SSE4</td>
<td>128</td>
<td>8</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>Sandybridge</td>
<td>2011</td>
<td>AVX</td>
<td>256</td>
<td>16</td>
<td>12</td>
<td>192</td>
</tr>
<tr>
<td>Haswell</td>
<td>2013</td>
<td>AVX2</td>
<td>256</td>
<td>32</td>
<td>18</td>
<td>576</td>
</tr>
<tr>
<td>KNC</td>
<td>2012</td>
<td>AVX512</td>
<td>512</td>
<td>32</td>
<td>64</td>
<td>2048</td>
</tr>
<tr>
<td>KNL</td>
<td>2016</td>
<td>AVX512</td>
<td>512</td>
<td>64</td>
<td>72</td>
<td>4608</td>
</tr>
<tr>
<td>Skylake</td>
<td>2017</td>
<td>AVX512</td>
<td>512</td>
<td>96</td>
<td>28</td>
<td>2688</td>
</tr>
</tbody>
</table>
Prototypical Application: Serial Weather Model
First Parallel Weather Modeling Algorithm: Richardson in 1917

Courtesy John Burkhardt, Virginia Tech
Four meteorologists in the same room sharing the map.

Fortran:

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

C/C++:

```c
#pragma omp parallel for
for(i=1; i<=n; i++)
a[i] = b[i] + c[i];
```
1 meteorologists coordinating 1000 math savants using tin cans and a string.

```c
#pragma acc kernels
for (i=0; i<N; i++)  {
    double t = (double)((i+0.05)/N);
    pi += 4.0/(1.0+t*t);
}

__global__ void saxpy_kernel( float a, float* x, float* y, int n ){
    int i;
    i = blockIdx.x*blockDim.x + threadIdx.x;
    if( i <= n ) x[i] = a*x[i] + y[i];
}
```
call MPI_Send( numbertosend, 1, MPI_INTEGER, index, 10, MPI_COMM_WORLD, errcode)

50 meteorologists using a telegraph.
The pieces fit like this...
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
- Multi-Core (Threads)
- SMP/Multi-socket
- Accelerators: GPU & MIC
- Clusters
- MPPs

Compiler
(not your problem)

Also Important
- ASIC/FPGA/DSP
- RAID/IO
Cores, Nodes, Processors, PEs?

- The most unambiguous way to refer to the smallest useful computing device is as a Processing Element, or PE.

- This is usually the same as a single core.

- “Processors” usually have more than one core – as per the previous list.

- “Nodes” is commonly used to refer to an actual physical unit, most commonly a circuit board or blade with a network connection. These often have multiple processors.

I will try to use the term PE consistently here, but I may slip up myself. Get used to it as you will quite often hear all of the above terms used interchangeably where they shouldn’t be.
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
GPU Architecture - GK110 Kepler

From a document you should read if you are interested in this:
<table>
<thead>
<tr>
<th>#</th>
<th>Site</th>
<th>Manufacturer</th>
<th>Computer</th>
<th>CPU Interconnect [Accelerator]</th>
<th>Cores</th>
<th>Rmax (Tflops)</th>
<th>Rpeak (Tflops)</th>
<th>Power (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RIKEN Center for Computational Science Japan</td>
<td>Fujitsu</td>
<td>Fugaku</td>
<td>ARM 8.2A+ 48C 2.2GHz Torus Fusion Interconnect</td>
<td>7,299,072</td>
<td>415,530</td>
<td>513,854</td>
<td>28.3</td>
</tr>
<tr>
<td>2</td>
<td>DOE/SC/ORNL United States</td>
<td>IBM</td>
<td>Summit</td>
<td>Power9 22C 3.0 GHz Dual-rail Infiniband EDR NVIDIA V100</td>
<td>2,414,592</td>
<td>148,600</td>
<td>200,794</td>
<td>10.1</td>
</tr>
<tr>
<td>3</td>
<td>DOE/NNSA/LLNL United States</td>
<td>IBM</td>
<td>Sierra</td>
<td>Power9 3.1 GHz 22C Infiniband EDR NVIDIA V100</td>
<td>1,572,480</td>
<td>94,640</td>
<td>125,712</td>
<td>7.4</td>
</tr>
<tr>
<td>4</td>
<td>National Super Computer Center in Wuxi China</td>
<td>NRCPC</td>
<td>Sunway TaihuLight</td>
<td>Sunway SW26010 260C 1.45GHz</td>
<td>10,649,600</td>
<td>93,014</td>
<td>125,435</td>
<td>15.3</td>
</tr>
<tr>
<td>5</td>
<td>National Super Computer Center in Guangzhou China</td>
<td>NUDT</td>
<td>Tianhe-2 (MilkyWay-2)</td>
<td>Intel Xeon E5-2692 2.2 GHz TH Express-2 Intel Xeon Phi 31S1P</td>
<td>4,981,760</td>
<td>61,444</td>
<td>100,678</td>
<td>18.4</td>
</tr>
<tr>
<td>6</td>
<td>Eni S.p.A Italy</td>
<td>Dell</td>
<td>HPC5</td>
<td>Xeon 24C 2.1 GHz Infiniband HDR NVIDIA V100</td>
<td>669,760</td>
<td>35,450</td>
<td>51,720</td>
<td>2.2</td>
</tr>
<tr>
<td>7</td>
<td>Eni S.p.A Italy</td>
<td>NVIDIA</td>
<td>Selene</td>
<td>EPYC 64C 2.25GHz Infiniband HDR NVIDIA A100</td>
<td>272,800</td>
<td>27,580</td>
<td>34,568</td>
<td>1.3</td>
</tr>
<tr>
<td>8</td>
<td>Texas Advanced Computing Center/Univ. of Texas United States</td>
<td>Dell</td>
<td>Frontera</td>
<td>Intel Xeon 8280 28C 2.7 GHz InfiniBand HDR</td>
<td>448,448</td>
<td>23,516</td>
<td>38,745</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Cineca Italy</td>
<td>IBM</td>
<td>Marconi100</td>
<td>Power9 16C 3.0 GHz Infiniband EDR NVIDIA V100</td>
<td>347,776</td>
<td>21,640</td>
<td>29,354</td>
<td>1.5</td>
</tr>
<tr>
<td>10</td>
<td>Swiss National Supercomputing Centre (CSCS) Switzerland</td>
<td>Cray</td>
<td>Piz Daint/Cray XC50</td>
<td>Xeon E5-2690 2.6 GHz Aries NVIDIA P100</td>
<td>387,872</td>
<td>21,230</td>
<td>27,154</td>
<td>2.4</td>
</tr>
</tbody>
</table>
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)
In Conclusion...

OpenMP

OpenACC

MPI