It sounds simple...

Take Xeon Phi CPU, lots of flops in a chip
Add some memory and IO
Put multiple CPU’s on a board
Stick many boards in a cabinet
Wire up a network
Fill warehouse with cabinets
Call it a supercomputer!
EXA is a very big number (from Wikipedia)

Exa- (symbol E) is a prefix in the metric system denoting $10^{18}$ or 1,000,000,000,000,000,000
- Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta,...

Adopted in 1975, it comes from the Greek ἕξ, used as a prefix ἕξα-, meaning six (like hexa-), because it is equal to $1000^6$

Examples:

- 1 exasecond ~ 32 billion years
- 1 exameter ~ 110 light years
- 0.43 Es ≈ the age of the Universe
- 1.6 Em — 172 ± 12.5 light years — Diameter of Omega Centauri (one of the largest known globular clusters, perhaps containing over a million stars)
Supercomputer Performance Growth

Since Cray 1 in 1976, Supercomputer performance has grown by a factor of ~1000 every eleven years

- Almost doubling every year. Roughly twice the rate of general processor performance
- As measured by Linpack benchmark
- Follows history of processor architecture:
  - Vectors, Frequency, RISC, Superscalar, GPU, Multicore, back to vectors, ...

Some factors:

- Linpack performance scales with transistor count
  - New benchmark based on Conjugate Gradient on horizon
- No limit on system growth: From uniprocessor to many thousand processors
  - But Exascale goal comes with power limit of 20 Mega Watt
Points along the way...

Xeon Phi line: KNF, KNC, ...

Roughly:
- Flops per core per cycle (vectors): 16
- Core Frequency: 1 GHz
- Cores per die: 60
- Sockets in the system: 6,400

Possible Evolution:

\[
\begin{align*}
\text{Flops} & \rightarrow 32 & 10^{**2} \\
\text{Frequency} & \rightarrow 2.5 \text{ GHz} & 10^{**9} \\
\text{Core count} & \rightarrow 100 & 10^{**2} \\
\text{Sockets} & \rightarrow 100,000 & 10^{**5}
\end{align*}
\]
Using Common Components

- Processor die
- IO Die
- Proc
- Router
- DRAM DIMM
- Memory link
- Scalability link

Argonne Training Program on Extreme Scale Computing 2013
Chip level Multi Processing, CMP

Not quite Exascale on Die, but a start

Tiles of Cores and Caches in regular interconnect

Along with Memory Controllers and Network Interfaces

Scalability Links

Memory Links

Memory Interface Unit

Core Tile

Scalability Interface Unit

Data Channels

Argonne Training Program on Extreme Scale Computing 2013
Challenges to Exascale

Need Breakthrough Innovations Across the Board
“We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible.”

Memory access time, $T$:

$$T_{avg} = Hit_{rate} \times T_{cache} + Miss_{rate} \times T_{memory}$$

Compulsory, Capacity, Conflict misses

- Clever (or brute force) cache design mainly helps with Capacity and Conflict misses

~ Every fifth instruction reads memory

*When $T_{avg}$ exceeds 5 instruction times, it may signal the end of computer architecture*

Compulsory Misses (alone) will get us to the wall (at least in HPC)

*This is a major reason for CMP!*
What makes memory different?

Intel is a chip integration company

We integrate everything of importance
– Whole Core, FPU, Cache, HT, SSE, Multiple Cores, VPU, Memory Controller, Printer, Network, PCIe, ...

Why not memory?

• No capacity is ever enough
• The reason we want higher performance computers is to process more information
• Ideally, all the World’s information fits in memory so we can access it and use it in calculations
The role of Caches

Caches – the best solution so far:

• 360/85 - 1969 - the first production machine with cache memory
• 16KB Buffer, ¼ latency of main memory (up to 4 MB). Write thru. No allocate. Unified. 64 byte blocks
• 360/85 with cache beat out 360/91 with out of order processing

Caches are wonderful, we keep reducing miss rate, both capacity and conflict

• Bigger, multi-level, more associative, better allocation, sharing, shorter latency, ECC, new functionality, ... the gift that keeps on giving

Compulsory misses call for prefetching and multi threading --- both require more bandwidth, ...

Very large caches remain mysterious
Pushing back Memory Wall

High Bandwidth Memory
- Memory Stacks
- Through Silicon Via’s for high bandwidth
- And Low Power!
  - Short wires
  - Power single page!
- But low capacity

Reduce need for Memory Bandwidth
- Bigger, better caches
- Data compression on memory links
- Sub-block transfers and caching
- Active Memory Operations
Memory Hierarchy

Stacked Memories
- High Bandwidth
- Low Power
- Low capacity

Software challenge to control:
- Fast Malloc
- Malloc
- Slow, but Persistent, Malloc

Traditional DIMMs
- Low Bandwidth
- High Power
- High Capacity

Possible Solutions:
- Near Memory as Cache
- Blocking for Memory Hierarchy
- NUMA-like OS control

Third level Memory
- Low Cost
- Non Volatile

Some day all memory will be fast, large, low power, and non-volatile?
Shrinking Circuits and More Cabinets, what happens to Reliability?

100 thousand sockets, 10 million cores, 40 million threads

Good solutions for memory and networks

Cores are the challenge

Detection and Recovery

- Recovery through check pointing
  - OS, Compiler, Application, HW
  - Micro level retry
- Wish for elegant solution for error detection
  - HW checkers, Self-checking software, Redundancy
- Robust Systems:
  - Co-design of software and hardware
Power: How do we study global warming without causing it?

Miniaturization and Chip level integration saves overall power
- But creates a local power problem
- Limits physical packaging options
  - E.g. memory and network access

Clock gating saves dynamic power
- But does not help leakage power

Power gating uses power to save power

Voltage/frequency scaling still works, but range is shrinking

Need fresh ideas to get to Exascale at 20 MegaWatts
Robert Dennard’s Scaling Rules for MOSFETs


Scaling factor : \( \kappa \sim \sqrt{2} \sim 1.4, \quad 1/\kappa \sim 0.7 \)

Device dimension \( t_{ox}, L, W \): \( 1/\kappa \)

Doping Concentration, \( N_a \): \( \kappa \)

Voltage, \( V \): \( 1/\kappa \)

Current, \( I \): \( 1/\kappa \)

Capacitance, \( \varepsilon A/t \), \( 1/\kappa \)

Delay time/circuit, \( V_C/I \): \( 1/\kappa \)

Power dissipation per circuit, \( V_I \): \( 1/\kappa^2 \)

Power density, \( V_I/A \): \( 1 \)
Moore’s Law will provide transistors

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</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

Argonne Training Program on Extreme Scale Computing 2013
Programming for Extreme Parallelism

Your text here...