Past, Present, and Future of High Performance Computing

Jack Dongarra

University of Tennessee
Oak Ridge National Laboratory
University of Manchester
Looking at the Gordon Bell Prize
(Recognize outstanding achievement in high-performance computing applications and encourage development of parallel processing)

- 1 GFlop/s; 1988; Cray Y-MP; 8 Processors
  - Static finite element analysis

- 1 TFlop/s; 1998; Cray T3E; 1024 Processors
  - Modeling of metallic magnet atoms, using a variation of the locally self-consistent multiple scattering method.

- 1 PFlop/s; 2008; Cray XT5; $1.5 \times 10^5$ Processors
  - Superconductive materials

- 1 EFlop/s; ~2018; ?; $1 \times 10^7$ Processors ($10^9$ threads)
- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

$$Ax = b, \text{ dense problem}$$

- Updated twice a year
  SC‘xy in the States in November
  Meeting in Germany in June

- All data available from www.top500.org
<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Computer</th>
<th>Country</th>
<th>Cores</th>
<th>Rmax [Tflops]</th>
<th>% of Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nat. SuperComputer Center in Tianjin</td>
<td>NUDT YH Cluster, X5670 2.93Ghz 6C, NVIDIA GPU</td>
<td>China</td>
<td>186,368</td>
<td>2.57</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>DOE / OS Oak Ridge Nat Lab</td>
<td>Jaguar / Cray Cray XT5 sixCore 2.6 GHz</td>
<td>USA</td>
<td>224,162</td>
<td>1.76</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>Nat. Supercomputer Center in Shenzhen</td>
<td>Nebulea / Dawning / TC3600 Blade, Intel X5650, Nvidia C2050 GPU</td>
<td>China</td>
<td>120,640</td>
<td>1.27</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>GSIC Center, Tokyo Institute of Technology</td>
<td>Tusbame 2.0 HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU</td>
<td>Japan</td>
<td>73,278</td>
<td>1.19</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/LBNL/NERSC</td>
<td>Hopper, Cray XE6 12-core 2.1 GHz</td>
<td>USA</td>
<td>153,408</td>
<td>1.054</td>
<td>82</td>
</tr>
<tr>
<td>6</td>
<td>Commissariat a l'Energie Atomique (CEA)</td>
<td>Tera-100 Bull bullx supernode S6010/S6030</td>
<td>France</td>
<td>138,368</td>
<td>1.050</td>
<td>84</td>
</tr>
<tr>
<td>7</td>
<td>DOE / NNSA Los Alamos Nat Lab</td>
<td>Roadrunner / IBM BladeCenter QS22/LS21</td>
<td>USA</td>
<td>122,400</td>
<td>1.04</td>
<td>76</td>
</tr>
<tr>
<td>8</td>
<td>NSF / NICS / U of Tennessee</td>
<td>Jaguar / Cray Cray XT5 sixCore 2.6 GHz</td>
<td>USA</td>
<td>98,928</td>
<td>.831</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>Forschungszentrum Juelich (FZJ)</td>
<td>Jugene / IBM Blue Gene/P Solution</td>
<td>Germany</td>
<td>294,912</td>
<td>.825</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>DOE/ NNSA / Los Alamos Nat Lab</td>
<td>Cray XE6 8-core 2.4 GHz</td>
<td>USA</td>
<td>107,152</td>
<td>.817</td>
<td>79</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------</td>
<td>---------------</td>
<td>-----------</td>
</tr>
<tr>
<td>1</td>
<td>Nat. SuperComputer Center in Tianjin</td>
<td>NUDT YH Cluster, X5670 2.93Ghz 6C, NVIDIA GPU</td>
<td>China</td>
<td>186,368</td>
<td>2.57</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>DOE / OS Oak Ridge Nat Lab</td>
<td>Jaguar / Cray Cray XT5 sixCore 2.6 GHz</td>
<td>USA</td>
<td>224,162</td>
<td>1.76</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>Nat. Supercomputer Center in Shenzhen</td>
<td>Nebulea / Dawning / TC3600 Blade, Intel X5650, Nvidia C2050 GPU</td>
<td>China</td>
<td>120,640</td>
<td>1.27</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>GSIC Center, Tokyo Institute of Technology</td>
<td>Tusbame 2.0 HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU</td>
<td>Japan</td>
<td>73,278</td>
<td>1.19</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/LBNL/NERSC</td>
<td>Hopper, Cray XE6 12-core 2.1 GHz</td>
<td>USA</td>
<td>153,408</td>
<td>1.054</td>
<td>82</td>
</tr>
<tr>
<td>6</td>
<td>Commissariat a l'Energie Atomique (CEA)</td>
<td>Tera-100 Bull bullx supernode S6010/S6030</td>
<td>France</td>
<td>138,368</td>
<td>1.050</td>
<td>84</td>
</tr>
<tr>
<td>7</td>
<td>DOE / NNSA Los Alamos Nat Lab</td>
<td>Roadrunner / IBM BladeCenter QS22/LS21</td>
<td>USA</td>
<td>122,400</td>
<td>1.04</td>
<td>76</td>
</tr>
<tr>
<td>8</td>
<td>NSF / NICS / U of Tennessee</td>
<td>Jaguar / Cray Cray XT5 sixCore 2.6 GHz</td>
<td>USA</td>
<td>98,928</td>
<td>.831</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>Forschungszentrum Juelich (FZJ)</td>
<td>Jugene / IBM Blue Gene/P Solution</td>
<td>Germany</td>
<td>294,912</td>
<td>.825</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>DOE/ NNSA / Los Alamos Nat Lab</td>
<td>Cray XE6 8-core 2.4 GHz</td>
<td>USA</td>
<td>107,152</td>
<td>.817</td>
<td>79</td>
</tr>
</tbody>
</table>
# Pflop/s Club (11 systems; Peak)

<table>
<thead>
<tr>
<th>Name</th>
<th>Peak Pflop/s</th>
<th>“Linpack” Pflop/s</th>
<th>Country</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tianhe-1A</td>
<td>4.70</td>
<td>2.57</td>
<td>China</td>
<td>NUDT: Hybrid Intel/Nvidia/Self</td>
</tr>
<tr>
<td>Nebula</td>
<td>2.98</td>
<td>1.27</td>
<td>China</td>
<td>Dawning: Hybrid Intel/Nvidia/IB</td>
</tr>
<tr>
<td>Jaguar</td>
<td>2.33</td>
<td>1.76</td>
<td>US</td>
<td>Cray: AMD/Self</td>
</tr>
<tr>
<td>Tsubame 2.0</td>
<td>2.29</td>
<td>1.19</td>
<td>Japan</td>
<td>HP: Hybrid Intel/Nvidia/IB</td>
</tr>
<tr>
<td>RoadRunner</td>
<td>1.38</td>
<td>1.04</td>
<td>US</td>
<td>IBM: Hybrid AMD/Cell/IB</td>
</tr>
<tr>
<td>Hopper</td>
<td>1.29</td>
<td>1.054</td>
<td>US</td>
<td>Cray: AMD/Self</td>
</tr>
<tr>
<td>Tera-100</td>
<td>1.25</td>
<td>1.050</td>
<td>France</td>
<td>Bull: Intel/IB</td>
</tr>
<tr>
<td>Mole-8.5</td>
<td>1.14</td>
<td>.207</td>
<td>China</td>
<td>CAS: Hybrid Intel/Nvidia/IB</td>
</tr>
<tr>
<td>Kraken</td>
<td>1.02</td>
<td>.831</td>
<td>US</td>
<td>Cray: AMD/Self</td>
</tr>
<tr>
<td>Cielo</td>
<td>1.02</td>
<td>.817</td>
<td>US</td>
<td>Cray: AMD/Self</td>
</tr>
<tr>
<td>JuGene</td>
<td>1.00</td>
<td>.825</td>
<td>Germany</td>
<td>IBM: BG-P/Self</td>
</tr>
</tbody>
</table>
Performance of Countries

- US
- EU
- Japan
Performance of Countries

- US
- EU
- Japan
- China

Teraflop/sec

Town Hall Meetings April–June 2007

Scientific Grand Challenges Workshops November 2008 – October 2009
- Climate Science (11/08),
- High Energy Physics (12/08),
- Nuclear Physics (1/09),
- Fusion Energy (3/09),
- Nuclear Energy (5/09),
- Biology (8/09),
- Material Science and Chemistry (8/09),
- National Security (10/09) (with NNSA)

Cross-cutting workshops
- Architecture and Technology (12/09)
- Architecture, Applied Math and CS (2/10)

Meetings with industry (8/09, 11/09)

External Panels
- ASCAC Exascale Charge (FACA)
- Trivelpiece Panel

“The key finding of the Panel is that there are compelling needs for exascale computing capability to support the DOE’s missions in energy, national security, fundamental sciences, and the environment. The DOE has the necessary assets to initiate a program that would accelerate the development of such capability to meet its own needs and by so doing benefit other national interests. Failure to initiate an exascale program could lead to a loss of U.S. competitiveness in several critical technologies.”

Trivelpiece Panel Report, January, 2010
## Potential System Architectures

<table>
<thead>
<tr>
<th>Systems</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>2 Pflop/s</td>
</tr>
<tr>
<td>System memory</td>
<td>0.3 PB</td>
</tr>
<tr>
<td>Node performance</td>
<td>125 Gflop/s</td>
</tr>
<tr>
<td>Node memory BW</td>
<td>25 GB/s</td>
</tr>
<tr>
<td>Node concurrency</td>
<td>12</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>18,700</td>
</tr>
<tr>
<td>Total concurrency</td>
<td>225,000</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
</tr>
<tr>
<td>IO</td>
<td>0.2 TB/s</td>
</tr>
<tr>
<td>MTTI</td>
<td>days</td>
</tr>
<tr>
<td>Power</td>
<td>7 MW</td>
</tr>
</tbody>
</table>
Exascale \( (10^{18} \text{ Flop/s}) \) Systems:
Two possible paths

- **Light weight processors (think BG/P)**
  - \( \sim 1 \) GHz processor \( (10^9) \)
  - \( \sim 1 \) Kilo cores/socket \( (10^3) \)
  - \( \sim 1 \) Mega sockets/system \( (10^6) \)

- **Hybrid system (think GPU based)**
  - \( \sim 1 \) GHz processor \( (10^9) \)
  - \( \sim 10 \) Kilo FPUs/socket \( (10^4) \)
  - \( \sim 100 \) Kilo sockets/system \( (10^5) \)
Factors that Necessitate Redesign of Our Software

- Steepness of the ascent from terascale to petascale to exascale
- Extreme parallelism and hybrid design
  - Preparing for million/billion way parallelism
- Tightening memory/bandwidth bottleneck
  - Limits on power/clock speed implication on multicore
  - Reducing communication will become much more intense
  - Memory per core changes, byte-to-flop ratio will change
- Necessary Fault Tolerance
  - MTTF will drop
  - Checkpoint/restart has limitations

Software infrastructure does not exist today
Commodity plus Accelerators

**Commodity**
- Intel Xeon
- 8 cores
- 3 GHz
- 8x4 ops/cycle
- 96 Gflop/s (DP)

**Accelerator (GPU)**
- Nvidia C2050 “Fermi”
- 448 “Cuda cores”
- 1.15 GHz
- 448 ops/cycle
- 515 Gflop/s (DP)

![Diagram showing the connection between the X86 Host, Host Memory, and Device Memory using Interconnect PCI Express with a bandwidth of 512 MB/s to 32GB/s or 8 MW – 512 MW.]
Major Changes to Software

- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with cluster computing and message passing
  - Rethink and rewrite the applications, algorithms, and software
- Numerical libraries for example will change
  - For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this
Five Important Software Features to Consider When Computing at Scale

1. Effective Use of Many-Core and Hybrid architectures
   - Break fork-join parallelism
   - Dynamic Data Driven Execution
   - Block Data Layout

2. Exploiting Mixed Precision in the Algorithms
   - Single Precision is 2X faster than Double Precision
   - With GP-GPUs 10x
   - Power saving issues

3. Self Adapting / Auto Tuning of Software
   - Too hard to do by hand

4. Fault Tolerant Algorithms
   - With 1,000,000’s of cores things will fail

5. Communication Reducing Algorithms
   - For dense computations from $O(n \log p)$ to $O(\log p)$ communications
   - Asynchronous iterations
   - GMRES k-step compute ($x, Ax, A^2x, \ldots A^kx$)


- Fork-join, bulk synchronous processing
Parallel Tasks in LU/LLᵀ/QR

- Break into smaller tasks and remove dependencies

* LU does block pair wise pivoting
PLASMA: Parallel Linear Algebra s/w for Multicore Architectures

• Objectives
  ▪ High utilization of each core
  ▪ Scaling to large number of cores
  ▪ Shared or distributed memory

• Methodology
  ▪ Dynamic DAG scheduling
  ▪ Explicit parallelism
  ▪ Implicit communication
  ▪ Fine granularity / block data layout

• Arbitrary DAG with dynamic scheduling

Diagram showing DAG scheduling with different parallelism options.
Communication Avoiding Algorithms

- **Goal:** Algorithms that communicate as little as possible
- **Jim Demmel and company have been working on algorithms that obtain a provable minimum communication.**
- **Direct methods (BLAS, LU, QR, SVD, other decompositions)**
  - Communication lower bounds for *all* these problems
  - Algorithms that attain them (*all* dense linear algebra, some sparse)
    - Mostly not in LAPACK or ScaLAPACK (yet)
- **Iterative methods - Krylov subspace methods for Ax=b, Ax=λx**
  - Communication lower bounds, and algorithms that attain them (depending on sparsity structure)
    - Not in any libraries (yet)
- **For QR Factorization they can show:**

<table>
<thead>
<tr>
<th># flops</th>
<th>$\Theta(mn^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td># words</td>
<td>$\Theta(\frac{mn^2}{W})$</td>
</tr>
<tr>
<td># messages</td>
<td>$\Theta(\frac{mn^2}{W^{3/2}})$</td>
</tr>
</tbody>
</table>
Hybrid Computing

- Match algorithmic requirements to architectural strengths of the hybrid components
  - **Multicore**: small tasks/tiles
  - **Accelerator**: large data parallel tasks

- e.g. split the computation into tasks; define critical path that “clears” the way for other large data parallel tasks; proper schedule the tasks execution

- Design algorithms with well defined “search space” to facilitate auto-tuning
What’s Next?

- All Large Core
- Mixed Large and Small Core
- Many Small Cores
- All Small Core
- Many Floating-Point Cores

Different Classes of Chips
- Home
- Games / Graphics
- Business
- Scientific

+ 3D Stacked Memory

3D memory layers

photonic NoC

multi-core processor layer
Parallel Performance of the hybrid SPOTRF (4 Opteron 1.8GHz and 4 GPU TESLA C1060 1.44GHz)

- 1CPU-1GPU
- 2CPUs-2GPUs
- 3CPUs-3GPUs
- 4CPUs-4GPUs

Matrix sizes
Exploiting Mixed Precision Computations

- Single precision is faster than DP because:
  - **Higher parallelism within floating point units**
    - 4 ops/cycle (usually) instead of 2 ops/cycle
  - **Reduced data motion**
    - 32 bit data instead of 64 bit data
  - **Higher locality in cache**
    - More data items in cache
Idea Goes Something Like This...

- Exploit 32 bit floating point as much as possible.
  - Especially for the bulk of the computation
- Correct or update the solution with selective use of 64 bit floating point to provide a refined results
- Intuitively:
  - Compute a 32 bit result,
  - Calculate a correction to 32 bit result using selected higher precision and,
  - Perform the update of the 32 bit results with the correction using high precision.
Iterative refinement for dense systems, $Ax = b$, can work this way.

\[
\begin{align*}
L U & = lu(A) \\
x & = L\backslash(U\backslash b) \\
r & = b - Ax \\
\text{WHILE} \quad ||r|| \text{ not small enough} \\
z & = L\backslash(U\backslash r) \\
x & = x + z \\
r & = b - Ax \\
\text{END}
\end{align*}
\]

- Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.
Iterative refinement for dense systems, \( Ax = b \), can work this way.

\[
L \ U = \text{lu}(A) \\
\text{SINGLE} \quad O(n^3) \\
x = L\backslash(U\backslash b) \\
\text{SINGLE} \quad O(n^2) \\
r = b - Ax \\
\text{DOUBLE} \quad O(n^2) \\
\text{WHILE} \ |r| \text{ not small enough} \\
z = L\backslash(U\backslash r) \\
\text{SINGLE} \quad O(n^2) \\
x = x + z \\
\text{DOUBLE} \quad O(n^1) \\
r = b - Ax \\
\text{DOUBLE} \quad O(n^2) \\
\text{END}
\]

- Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.
- It can be shown that using this approach we can compute the solution to 64-bit floating point precision.

- Requires extra storage, total is 1.5 times normal;
- \( O(n^3) \) work is done in lower precision
- \( O(n^2) \) work is done in high precision
- Problems if the matrix is ill-conditioned in sp; \( O(10^8) \)
$Ax = b$

---

**Single Precision**

**Double Precision**

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>Single Precision</th>
<th>Double Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>960</td>
<td>960</td>
<td>960</td>
</tr>
<tr>
<td>3200</td>
<td>3200</td>
<td>3200</td>
</tr>
<tr>
<td>5120</td>
<td>5120</td>
<td>5120</td>
</tr>
<tr>
<td>7040</td>
<td>7040</td>
<td>7040</td>
</tr>
<tr>
<td>8960</td>
<td>8960</td>
<td>8960</td>
</tr>
<tr>
<td>11200</td>
<td>11200</td>
<td>11200</td>
</tr>
<tr>
<td>13120</td>
<td>13120</td>
<td>13120</td>
</tr>
</tbody>
</table>

Tesla C2050, 448 CUDA cores (14 multiprocessors x 32) @ 1.15 GHz,
3 GB memory, connected through PCIe to a quad-core Intel @2.5 GHz.
Ax = b

Tesla C2050, 448 CUDA cores (14 multiprocessors x 32) @ 1.15 GHz,
3 GB memory, connected through PCIe to a quad-core Intel @2.5 GHz.
Automatic Performance Tuning

• Writing high performance software is hard
• Ideal: get high fraction of peak performance from one algorithm
• Reality: Best algorithm (and its implementation) can depend strongly on the problem, computer architecture, compiler,…
  ▪ Best choice can depend on knowing a lot of applied mathematics and computer science
  ▪ Changes with each new hardware, compiler release
• Automatic performance tuning
  ▪ Use machine time in place of human time for tuning
  ▪ Search over possible implementations
  ▪ Use performance models to restrict search space
  ▪ Past successes: ATLAS, FFTW, Spiral, Open-MPI
How to Deal with Complexity?

- Many parameters in the code need to be optimized.
- Software adaptivity is the key for applications to effectively use available resources whose complexity is exponentially increasing.
Auto-Tuning

- Best algorithm implementation can depend strongly on the problem, computer architecture, compiler,...

- There are 2 main approaches
  - Model-driven optimization
    - Analytical models for various parameters;
    - Heavily used in the compilers community;
    - May not give optimal results
  - Empirical optimization
    - Generate large number of code versions and runs them on a given platform to determine the best performing one;
    - Effectiveness depends on the chosen parameters to optimize and the search heuristics used

- Natural approach is to combine them in a hybrid approach
  - 1st model-driven to limit the search space for a 2nd empirical part
  - Another aspect is adaptivity - to treat cases where tuning can not be restricted to optimizations at design, installation, or compile time
# PLASMA 2.3 for Multicore Systems

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear systems and least squares</td>
<td>LU, Cholesky, QR &amp; LQ</td>
</tr>
<tr>
<td>Mixed-precision linear systems</td>
<td>LU, Cholesky, QR</td>
</tr>
<tr>
<td><em>Tall and skinny</em> factorization</td>
<td>QR</td>
</tr>
<tr>
<td>Generation of the Q matrix</td>
<td>QR, LQ, tall and skinny QR</td>
</tr>
<tr>
<td>Explicit matrix inversion</td>
<td>Cholesky</td>
</tr>
<tr>
<td>Level 3 BLAS</td>
<td>GEMM, HEMM, HER2K, HERK, SYMM, SYR2K, SYRK, TRMM, TRSM (complete set)</td>
</tr>
<tr>
<td>In-place layout translations</td>
<td>CM, RM, CCRB, CRRB, RCRB, RRRB (all combinations)</td>
</tr>
</tbody>
</table>

## Features

- Covering four precisions: Z, C, D, S (and mixed-precision: ZC, DS)
- Static scheduling and dynamic scheduling with QUARK
- Support for Linux, MS Windows, Mac OS and AIX
# MAGMA 1.0 for Hybrid Systems

## Functionality

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear systems and least squares</td>
<td>LU, Cholesky, QR &amp; LQ</td>
</tr>
<tr>
<td>Mixed-precision linear systems</td>
<td>LU, Cholesky, QR</td>
</tr>
<tr>
<td><strong>Eigenvalue and singular value problems</strong></td>
<td>Reductions to upper Hessenberg, bidiagonal, and tridiagonal forms</td>
</tr>
<tr>
<td>Generation of the Q matrix</td>
<td>QR, LQ, Hessenberg, bidiagonalization, and tridiagonalization</td>
</tr>
<tr>
<td>MAGMA BLAS</td>
<td>Subset of BLAS, critical for MAGMA performance for Tesla and Fermi</td>
</tr>
</tbody>
</table>

## Features

- Covering four precisions: Z, C, D, S (and mixed-precision: ZC, DS)
- Support for multicore and one NVIDIA GPU
- CPU and GPU interfaces
- Support for Linux and Mac OS
Major Challenges are ahead for extreme computing

- Parallelism
- Hybrid
- Fault Tolerance
- Power
- ... and many others not discussed here

We will need completely new approaches and technologies to reach the Exascale level

This opens up many new opportunities for applied mathematicians and computer scientists
A Call to Action

- Hardware has changed dramatically while software ecosystem has remained stagnant
- Need to exploit new hardware trends (e.g., manycore, heterogeneity) that cannot be handled by existing software stack, memory per socket trends
- Emerging software technologies exist, but have not been fully integrated with system software, e.g., UPC, Cilk, CUDA, HPCS
- Community codes unprepared for sea change in architectures
- No global evaluation of key missing components
Exascale is a Global Challenge

- Formed in 2008
- Goal to engage international computer science community to address common software challenges for Exascale
- Focus on open source systems software that would enable multiple platforms
- Shared risk and investment
- Leverage international talent base
International Exascale Software Program

Improve the world’s simulation and modeling capability by improving the coordination and development of the HPC software environment

Workshops:

Build an international plan for coordinating research for the next generation open source software for scientific high-performance computing

www.exascale.org
4.1 Systems Software
4.1.1 Operating systems
4.1.2 Runtime Systems
4.1.2 I/O systems
4.1.3 External Environments
4.1.4 Systems Management

4.2 Development Environments
4.2.1 Programming Models
4.2.2 Frameworks
4.2.3 Compilers
4.2.4 Numerical Libraries
4.2.5 Debugging tools

4.3 Applications
4.3.1 Application Element: Algorithms
4.3.2 Application Support: Data Analysis and Visualization
4.3.3 Application Support: Scientific Data Management

4.4 Crosscutting Dimensions
4.4.1 Resilience
4.4.2 Power Management
4.4.3 Performance Optimization
4.4.4 Programmability
Where We Are Today:

- Ken Kennedy - Petascale Software Project (2006)  
  SC08 (Austin TX) meeting to generate interest  
  Funding from DOE’s Office of Science & NSF Office of Cyberinfrastructure and sponsorship by Europeans and Asians
- US meeting (Santa Fe, NM) April 6-8, 2009   
  65 people
- European meeting (Paris, France) June 28-29, 2009  
  Outline Report
- Asian meeting (Tsukuba Japan) October 18-20, 2009  
  Draft roadmap and refine report
- SC09 (Portland OR) BOF to inform others  
  Public Comment; Draft Report presented
- European meeting (Oxford, UK) April 13-14, 2010  
  Refine and prioritize roadmap; look at management models
- Maui Meeting October 18-19, 2010
- SC10 (New Orleans) BOF to inform others (Wed 5:30, Room 389)
- Kyoto Meeting - April 6-7, 2011

Nov 2008
Apr 2009
Jun 2009
Oct 2009
Nov 2009
Apr 2010
Oct 2010
Nov 2010
Apr 2011
www.exascale.org
Conclusions

• For the last decade or more, the research investment strategy has been overwhelmingly biased in favor of hardware.

• This strategy needs to be rebalanced - barriers to progress are increasingly on the software side.

• Moreover, the return on investment is more favorable to software.
  ▪ Hardware has a half-life measured in years, while software has a half-life measured in decades.

• High Performance Ecosystem out of balance
  ▪ Hardware, OS, Compilers, Software, Algorithms, Applications
    • No Moore’s Law for software, algorithms and applications
“We can only see a short distance ahead, but we can see plenty there that needs to be done.”

- Alan Turing (1912 –1954)

• www.exascale.org