Future Directions in High Performance Computing

Jack Dongarra
INNOVATIVE COMPUTING LABORATORY

University of Tennessee
Oak Ridge National Laboratory
University of Manchester
Outline

- Top500 Results
- Four Important Concepts that Will Effect Math Software
  - Effective Use of Many-Core
  - Exploiting Mixed Precision in Our Numerical Computations
  - Self Adapting / Auto Tuning of Software
  - Fault Tolerant Algorithms
- 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
Performance Development

- NEC Earth Simulator: 6.96 PF/s
- IBM BlueGene/L: 1.17 TF/s
- Intel ASCI Red: 5.9 TF/s
- IBM ASCI White: 10 Tflop/s
- Fujitsu ‘NWT’: 100 Gflop/s
- My Laptop: 0.4 GF/s

6-8 years

6.96 PF/s
478 TF/s
5.9 TF/s

1 Ptflop/s
100 Ttflop/s
10 Ttflop/s
1 Ttflop/s
10 Gflop/s
1 Gflop/s
100 Mflop/s

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Computer</th>
<th>Rmax [TF/s]</th>
<th>Installation Site</th>
<th>Country</th>
<th>Year</th>
<th>#Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>Blue Gene/L eServer Blue Gene Dual Core .7 GHz</td>
<td>478</td>
<td>DOE</td>
<td>USA</td>
<td>2007 Custom</td>
<td>212,992</td>
</tr>
<tr>
<td>IBM</td>
<td>Blue Gene/P Quad Core .85 GHz</td>
<td>167</td>
<td>Forschungszentrum Jülich</td>
<td>Germany</td>
<td>2007 Custom</td>
<td>65,536</td>
</tr>
<tr>
<td>SGI</td>
<td>Altix ICE 8200 Xeon Quad Core 3 GHz</td>
<td>127</td>
<td>SGI/New Mexico Computing Applications Center</td>
<td>USA</td>
<td>2007 Hybrid</td>
<td>14,336</td>
</tr>
<tr>
<td>HP</td>
<td>Cluster Platform Xeon Dual Core 3 GHz</td>
<td>118</td>
<td>Computational Research Laboratories, TATA SONS</td>
<td>India</td>
<td>2007 Commod</td>
<td>14,240</td>
</tr>
<tr>
<td>HP</td>
<td>Cluster Platform Dual Core 2.66 GHz</td>
<td>102.8</td>
<td>Government Agency</td>
<td>Sweden</td>
<td>2007 Commod</td>
<td>13,728</td>
</tr>
<tr>
<td>Cray</td>
<td>Opteron Dual Core 2.4 GHz</td>
<td>102.2</td>
<td>DOE</td>
<td>USA</td>
<td>2007 Hybrid</td>
<td>26,569</td>
</tr>
<tr>
<td>Cray</td>
<td>Opteron Dual Core 2.6 GHz</td>
<td>101.7</td>
<td>DOE</td>
<td>USA</td>
<td>2006 Hybrid</td>
<td>23,016</td>
</tr>
<tr>
<td>IBM</td>
<td>eServer Blue Gene/L Dual Core .7 GHz</td>
<td>91.2</td>
<td>IBM Thomas J. Watson Research Center</td>
<td>USA</td>
<td>2005 Custom</td>
<td>40,960</td>
</tr>
<tr>
<td>Cray</td>
<td>Opteron Dual Core 2.6 GHz</td>
<td>85.4</td>
<td>DOE</td>
<td>USA</td>
<td>2006 Hybrid</td>
<td>19,320</td>
</tr>
<tr>
<td>IBM</td>
<td>eServer Blue Gene/L Dual Core .7 GHz</td>
<td>82.1</td>
<td>Stony Brook/BNL, NY Center for Computational Sciences</td>
<td>USA</td>
<td>2006 Custom</td>
<td>36,864</td>
</tr>
</tbody>
</table>
IBM BlueGene/L #1 212,992 Cores

The compute node ASICs include all networking and processor functionality. Each compute ASIC includes two 32-bit superscalar PowerPC 440 embedded cores (note that L1 cache coherence is not maintained between these cores). (20.7K sec about 5.7 hours; n=2.5M)
7 systems > 100 Tflop/s
21 systems > 50 Tflop/s
149 systems > 10 Tflop/s

Top500 Systems November 2007
Chips Used in Each of the 500 Systems

- 72% Intel
- 12% IBM
- 16% AMD

- Intel IA-32: 3%
- Intel IA-64: 4%
- Intel EM64T: 65%
- AMD x86_64: 16%
- IBM Power: 12%
- HP PA-RISC: 0%
- HP Alpha: 0%
- NEC: 0%
- Sun Sparc: 0%
- Cray: 0%
Interconnect Analysis

Interconnect Efficiency (1993 - 2007)

- Others
- Cray Interconnect
- SP Switch
- Crossbar
- Quadrics
- Infiniband
- Myrinet
- Gigabit Ethernet

GigE + Infiniband + Myrinet = 82%
Countries / Performance (Nov 2007)

- United States: 60%
- Germany: 7.7%
- United Kingdom: 7.4%
- China: 7.4%
- Taiwan: 3.2%
- Sweden: 2.7%
- Spain: 2.8%
- India: 4.2%
- France: 2.7%
- Others: 2.7%
Power is an Industry Wide Problem

Google facilities
- leveraging hydroelectric power
- old aluminum plants
- >500,000 servers worldwide

“Hiding in Plain Sight, Google Seeks More Power”, by John Markoff, June 14, 2006

New Google Plant in The Dulles, Oregon, from NYT, June 14, 2006
Gflop/KWatt in the Top 20
<table>
<thead>
<tr>
<th>Green500 Rank</th>
<th>MFLOPS/W</th>
<th>Site*</th>
<th>Computer*</th>
<th>Total Power (kW)</th>
<th>TOP500 Rank*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>357.23</td>
<td>Science and Technology Facilities Council - Daresbury Laboratory</td>
<td>Blue Gene/P Solution</td>
<td>31.10</td>
<td>121</td>
</tr>
<tr>
<td>2</td>
<td>352.25</td>
<td>Max-Planck-Gesellschaft MPI/IPP</td>
<td>Blue Gene/P Solution</td>
<td>62.20</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>346.95</td>
<td>IBM - Rochester</td>
<td>Blue Gene/P Solution</td>
<td>124.40</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>336.21</td>
<td>Forschungszentrum Juelich (FZJ)</td>
<td>Blue Gene/P Solution</td>
<td>497.60</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>310.93</td>
<td>Oak Ridge National Laboratory</td>
<td>Blue Gene/P Solution</td>
<td>70.47</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>210.56</td>
<td>Harvard University</td>
<td>eServer Blue Gene Solution</td>
<td>44.80</td>
<td>170</td>
</tr>
<tr>
<td>7</td>
<td>210.56</td>
<td>High Energy Accelerator Research Organization / KEK</td>
<td>eServer Blue Gene Solution</td>
<td>44.80</td>
<td>171</td>
</tr>
<tr>
<td>8</td>
<td>210.56</td>
<td>IBM - Almaden Research Center</td>
<td>eServer Blue Gene Solution</td>
<td>44.80</td>
<td>172</td>
</tr>
<tr>
<td>9</td>
<td>210.56</td>
<td>IBM Research</td>
<td>eServer Blue Gene Solution</td>
<td>44.80</td>
<td>173</td>
</tr>
<tr>
<td>10</td>
<td>210.56</td>
<td>IBM Thomas J. Watson Research Center</td>
<td>eServer Blue Gene Solution</td>
<td>44.80</td>
<td>174</td>
</tr>
</tbody>
</table>
Performance Projection

- 1 Eflop/s
- 1 Pflop/s
- 10 Pflop/s
- 100 Pflop/s
- SUM 1 Tflop/s
- 100 Tflop/s
- 10 Tflop/s
- 1 Tflop/s
- 100 Gflop/s
- 10 Gflop/s
- 1 Gflop/s
- 100 Mflop/s

- 1993 1995 1997 1999 2001 2003 2005 2007 2009 2011 2013 2015

- N=1 6-8 years
- N=500 8-10 years

www.top500.org
Los Alamos Roadrunner
A Petascale System in 2008

“Connected Unit” cluster
192 Opteron nodes
(180 w/ 2 dual-Cell blades
connected w/ 4 PCIe x8 links)

≈ 13,000 Cell HPC chips
• ≈ 1.33 PetaFlop/s (from Cell)
≈ 7,000 dual-core Opterons

2nd stage InfiniBand 4x DDR interconnect
(18 sets of 12 links to 8 switches)

2nd stage InfiniBand interconnect (8 switches)

Based on the 100 Gflop/s (DP) Cell chip

Approval by DOE 12/07
First CU being built today
Expect a May Pflop/s run
Full system to LANL in December 2008
Increasing CPU Performance: A Delicate Balancing Act

Increasing the number of gates into a tight knot and decreasing the cycle time of the processor

We have seen increasing number of gates on a chip and increasing clock speed.

Heat becoming an unmanageable problem, Intel Processors > 100 Watts

We will not see the dramatic increases in clock speeds in the future.

However, the number of gates on a chip will continue to increase.
## Power Cost of Frequency

- Power $\propto$ Voltage$^2 \times$ Frequency (V$^2$F)
- Frequency $\propto$ Voltage
- Power $\propto$ Frequency$^3$

<table>
<thead>
<tr>
<th></th>
<th>Cores</th>
<th>V</th>
<th>Freq</th>
<th>Perf</th>
<th>Power</th>
<th>PE (Bops/watt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superscalar</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>“New” Superscalar</td>
<td>1X</td>
<td>1.5X</td>
<td>1.5X</td>
<td>1.5X</td>
<td>3.3X</td>
<td>0.45X</td>
</tr>
</tbody>
</table>
Power Cost of Frequency

- Power $\propto$ Voltage$^2 \times$ Frequency (V$^2$F)
- Frequency $\propto$ Voltage
- Power $\propto$ Frequency$^3$

<table>
<thead>
<tr>
<th></th>
<th>Cores</th>
<th>V</th>
<th>Freq</th>
<th>Perf</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superscalar</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&quot;New&quot; Superscalar</td>
<td>1X</td>
<td>1.5X</td>
<td>1.5X</td>
<td>1.5X</td>
<td>3.3X</td>
</tr>
<tr>
<td>Multicore</td>
<td>2X</td>
<td>0.75X</td>
<td>0.75X</td>
<td>1.5X</td>
<td>0.8X</td>
</tr>
</tbody>
</table>

PE (Bops/watt)

- Superscalar: 1
- "New" Superscalar: 0.45X
- Multicore: 1.88X

(Bigger # is better)

50% more performance with 20% less power
Preferable to use multiple slower devices, than one superfast device
What’s Next?

- All Large Core
- Mixed Large and Small Core
- Many Small Cores
- Many Floating-Point Cores
- All Small Core
- + 3D Stacked Memory

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

SRAM
80 Core

- Intel’s 80 Core chip
  - 1 Tflop/s
  - 62 Watts
  - 1.2 TB/s internal BW

Intel Prototype May Herald a New Age of Processing

By JOHN MARKOFF
Published, February 12, 2007

SAN FRANCISCO, Feb. 11 — Intel will demonstrate on Monday an experimental computer chip with 80 separate processing engines, or cores, that company executives say provides a model for commercial chips that will be used widely in standard desktop, laptop and server computers within five years.

The new processor, which the company first described as a Teraflop Chip at a conference last year, will be detailed in a technical paper to be presented on the opening day of the International Solid States Circuits Conference, beginning here on Monday.

While the chip is not compatible with Intel’s current chips, the company said it had already begun design work on a commercial version that would essentially have dozens or even hundreds of Intel-compatible microprocessors laid out in a tiled pattern on a single chip.
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
A New Generation of Software:
Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

| Algorithms follow hardware evolution in time | LINPACK (70’s)  
(Vector operations) | LAPACK (80’s)  
(Blocking, cache friendly) | ScaLAPACK (90’s)  
(Distributed Memory) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rely on</td>
<td>Level-1 BLAS operations</td>
<td>Level-3 BLAS operations</td>
<td>PBLAS Mess Passing</td>
</tr>
</tbody>
</table>

Those new algorithms need new kernels and rely on efficient scheduling algorithms.
# A New Generation of Software:
Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Algorithms follow hardware evolution in time

<table>
<thead>
<tr>
<th>LINPACK (70’s) (Vector operations)</th>
<th>Rely on Level-1 BLAS operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK (80’s) (Blocking, cache friendly)</td>
<td>Rely on Level-3 BLAS operations</td>
</tr>
<tr>
<td>ScALAPACK (90’s) (Distributed Memory)</td>
<td>Rely on PBLAS Mess Passing</td>
</tr>
<tr>
<td>PLASMA (00’s) New Algorithms (many-core friendly)</td>
<td>Rely on a DAG/scheduler, block data layout, some extra kernels</td>
</tr>
</tbody>
</table>

Those new algorithms
- have a very low granularity, they scale very well (multicore, petascale computing, …)
- removes a lots of dependencies among the tasks, (multicore, distributed computing)
- avoid latency (distributed computing, out-of-core)
- rely on fast kernels

Those new algorithms need new kernels and rely on efficient scheduling algorithms.
Steps in the LAPACK LU

- **DGETF2** (Factor a panel)
- **DLSWP** (Backward swap)
- **DLSWP** (Forward swap)
- **DTRSM** (Triangular solve)
- **DGEMM** (Matrix multiply)

**LAPACK**

**BLAS**
LU Timing Profile (4 processor system)

Threads – no lookahead

Time for each component

1D decomposition and SGI Origin

Bulk Sync Phases

- DGETF2
- DLASWP(L)
- DLASWP(R)
- DTRSM
- DGEMM
Adaptive Lookahead - Dynamic

Event Driven Multithreading

while(1)
  fetch_task();
  switch(task.type) {
    case PANEL:
      dgetf2();
      update_progress();
    case COLUMN:
      dlaswp();
      dtrsm();
      dgemm();
      update_progress();
    case END:
      for()
        dlaswp();
      return;
  }

Reorganizing algorithms to use this approach
Fork-Join vs. Dynamic Execution

Experiments on Intel’s Quad Core Clovertown with 2 Sockets w/ 8 Treads
Fork-Join vs. Dynamic Execution

Fork-Join – parallel BLAS

DAG-based – dynamic scheduling

Experiments on Intel’s Quad Core Clovertown with 2 Sockets w/ 8 Treads
With the Hype on Cell & PS3 We Became Interested

- The PlayStation 3's CPU based on a "Cell" processor
- Each Cell contains a Power PC processor and 8 SPEs. (SPE is processing unit, SPE: SPU + DMA engine)
  - An SPE is a self contained vector processor which acts independently from the others.
    - 4 way SIMD floating point units capable of a total of 25.6 Gflop/s @ 3.2 GHZ
  - 204.8 Gflop/s peak!
  - The catch is that this is for 32 bit floating point; (Single Precision SP)
  - And 64 bit floating point runs at 14.6 Gflop/s total for all 8 SPEs!!
    - Divide SP peak by 14; factor of 2 because of DP and 7 because of latency issues

SPE ~ 25 Gflop/s peak
Performance of Single Precision on Conventional Processors

- Realized have the similar situation on our commodity processors.
  - That is, SP is 2X as fast as DP on many systems

- The Intel Pentium and AMD Opteron have SSE2
  - 2 flops/cycle DP
  - 4 flops/cycle SP

- IBM PowerPC has AltiVec
  - 8 flops/cycle SP
  - 4 flops/cycle DP
  - No DP on AltiVec

Single precision is faster because:
- Higher parallelism in SSE/vector units
- Reduced data motion
- Higher locality in cache

<table>
<thead>
<tr>
<th>Processor</th>
<th>Size</th>
<th>SGEMM/DGEMM</th>
<th>Size</th>
<th>SGEMV/DGEMV</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Opteron 246</td>
<td>3000</td>
<td>2.00</td>
<td>5000</td>
<td>1.70</td>
</tr>
<tr>
<td>UltraSparc-IIe</td>
<td>3000</td>
<td>1.64</td>
<td>5000</td>
<td>1.66</td>
</tr>
<tr>
<td>Intel PIII Coppermine</td>
<td>3000</td>
<td>2.03</td>
<td>5000</td>
<td>2.09</td>
</tr>
<tr>
<td>PowerPC 970</td>
<td>3000</td>
<td>2.04</td>
<td>5000</td>
<td>1.44</td>
</tr>
<tr>
<td>Intel Woodcrest</td>
<td>3000</td>
<td>1.81</td>
<td>5000</td>
<td>2.18</td>
</tr>
<tr>
<td>Intel XEON</td>
<td>3000</td>
<td>2.04</td>
<td>5000</td>
<td>1.82</td>
</tr>
<tr>
<td>Intel Centrino Duo</td>
<td>3000</td>
<td>2.71</td>
<td>5000</td>
<td>2.21</td>
</tr>
</tbody>
</table>
32 or 64 bit Floating Point Precision?

- **A long time ago 32 bit floating point was used**
  - Still used in scientific apps but limited
- **Most apps use 64 bit floating point**
  - Accumulation of round off error
    - A 10 TFlop/s computer running for 4 hours performs > 1 Exaflop ($10^{18}$) ops.
  - Ill conditioned problems
  - IEEE SP exponent bits too few (8 bits, $10^{±38}$)
  - Critical sections need higher precision
    - Sometimes need extended precision (128 bit fl pt)
  - However some can get by with 32 bit fl pt in some parts
- **Mixed precision a possibility**
  - Approximate in lower precision and then refine or improve solution to high precision.
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.
**Mixed-Precision Iterative Refinement**

- **Iterative refinement for dense systems,** $Ax = b$, can work this way.

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

  END

- Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.
Mixed-Precision Iterative Refinement

- Iterative refinement for dense systems, $Ax = b$, can work this way.

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

- 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; $O(10^8)$
Results for Mixed Precision Iterative Refinement for Dense $Ax = b$

- Single precision is faster than DP because:
  - Higher parallelism within vector 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
Results for Mixed Precision Iterative Refinement for Dense $Ax = b$

Architecture (BLAS)

1. Intel Pentium III Coppermine (Goto)
2. Intel Pentium III Katmai (Goto)
3. Sun UltraSPARC IIe (Sunperf)
4. Intel Pentium IV Prescott (Goto)
5. Intel Pentium IV-M Northwood (Goto)
6. AMD Opteron (Goto)
7. Cray X1 (libsci)
8. IBM Power PC G5 (2.7 GHz) (VecLib)
9. Compaq Alpha EV6 (CXML)
10. IBM SP Power3 (ESSL)
11. SGI Octane (ATLAS)

<table>
<thead>
<tr>
<th>Architecture (BLAS-MPI)</th>
<th># proc</th>
<th>$n$</th>
<th>DP Solve /SP Solve</th>
<th>DP Solve /Iter Ref</th>
<th># iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Opteron (Goto – OpenMPI MX)</td>
<td>32</td>
<td>22627</td>
<td>1.85</td>
<td>1.79</td>
<td>6</td>
</tr>
<tr>
<td>AMD Opteron (Goto – OpenMPI MX)</td>
<td>64</td>
<td>32000</td>
<td>1.90</td>
<td>1.83</td>
<td>6</td>
</tr>
</tbody>
</table>

- Single precision is faster than DP because:
  - Higher parallelism within vector 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
What about the Cell?

- **Power PC at 3.2 GHz**
  - DGEMM at 5 Gflop/s
  - Altivec peak at 25.6 Gflop/s
    - Achieved 10 Gflop/s SGEMM

- **8 SPUs**
  - 204.8 Gflop/s peak!
  - The catch is that this is for 32 bit floating point; (Single Precision SP)
  - And 64 bit floating point runs at 14.6 Gflop/s total for all 8 SPEs!!
    - Divide SP peak by 14; factor of 2 because of DP and 7 because of latency issues
Worst case memory bound operations (no reuse of data)
3 data movements (2 in and 1 out) with 2 ops (SAXPY)
For the cell would be 4.6 Gflop/s (25.6 GB/s*2ops/12B)
IBM Cell 3.2 GHz, $Ax = b$

8 SGEMM (Embarrassingly Parallel)

SP Peak (204 Gflop/s)
SP $Ax = b$ IBM
DP Peak (15 Gflop/s)
DP $Ax = b$ IBM

Matrix Size
IBM Cell 3.2 GHz, $Ax = b$

- **SP Peak (204 Gflop/s)**
- **SP Ax=b IBM**
- **DSGESV**
- **DP Peak (15 Gflop/s)**
- **DP Ax=b IBM**

The diagram shows a comparison of different operations in terms of matrix size and GFlop/s. The line labeled "8 SGEMM (Embarrassingly Parallel)" indicates a significant performance improvement. The chart illustrates that as the matrix size increases, the performance of the operations improves, with a notable 8.3X speedup compared to the baseline.

Key observations:
- GFlop/s increases with matrix size.
- SP Ax=b IBM operation shows a steady increase in performance.
- DSGESV operation shows a more linear improvement.
- DP Ax=b IBM operation shows a slight improvement compared to the baseline.

- **3.9 secs** for SP Ax=b IBM with a matrix size of 0.
- **.47 secs** for SP Ax=b IBM with a matrix size of 4500.
- **.30 secs** for DSGESV with a matrix size of 0.
- **.47 secs** for DSGESV with a matrix size of 4500.

The chart highlights the efficiency gains as the matrix size grows, especially for the SP Ax=b IBM operation.
Cholesky - Using 2 Cell Chips

![Graph showing performance of SPOTRF on QS20 with 2 cell BEs]
Intriguing Potential

- Exploit lower precision as much as possible
  - Payoff in performance
    - Faster floating point
    - Less data to move
- Automatically switch between SP and DP to match the desired accuracy
  - Compute solution in SP and then a correction to the solution in DP
- Potential for GPU, FPGA, special purpose processors
  - What about 16 bit floating point?
    - Use as little you can get away with and improve the accuracy
- Applies to sparse direct and iterative linear systems and Eigenvalue, optimization problems, where Newton’s method is used.

\[ x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \]

Correction = \[-A\|b - Ax\]
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
Collaborators / Support

Alfredo Buttari, UTK  
Julien Langou, UColorado  
Julie Langou, UTK  
Piotr Luszczek, MathWorks  
Jakub Kurzak, UTK  
Stan Tomov, UTK