



The Impact of Multicore on Mathematical Software

Jack Dongarra
INNOVATIVE COMPUTING LABORATORY

University of Tennessee
Oak Ridge National Laboratory



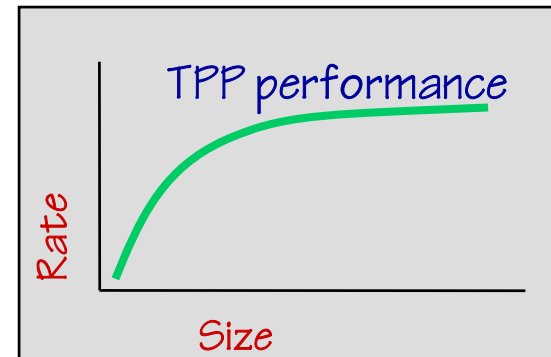
Outline

- **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**

H. Meuer, H. Simon, E. Strohmaier, & JD

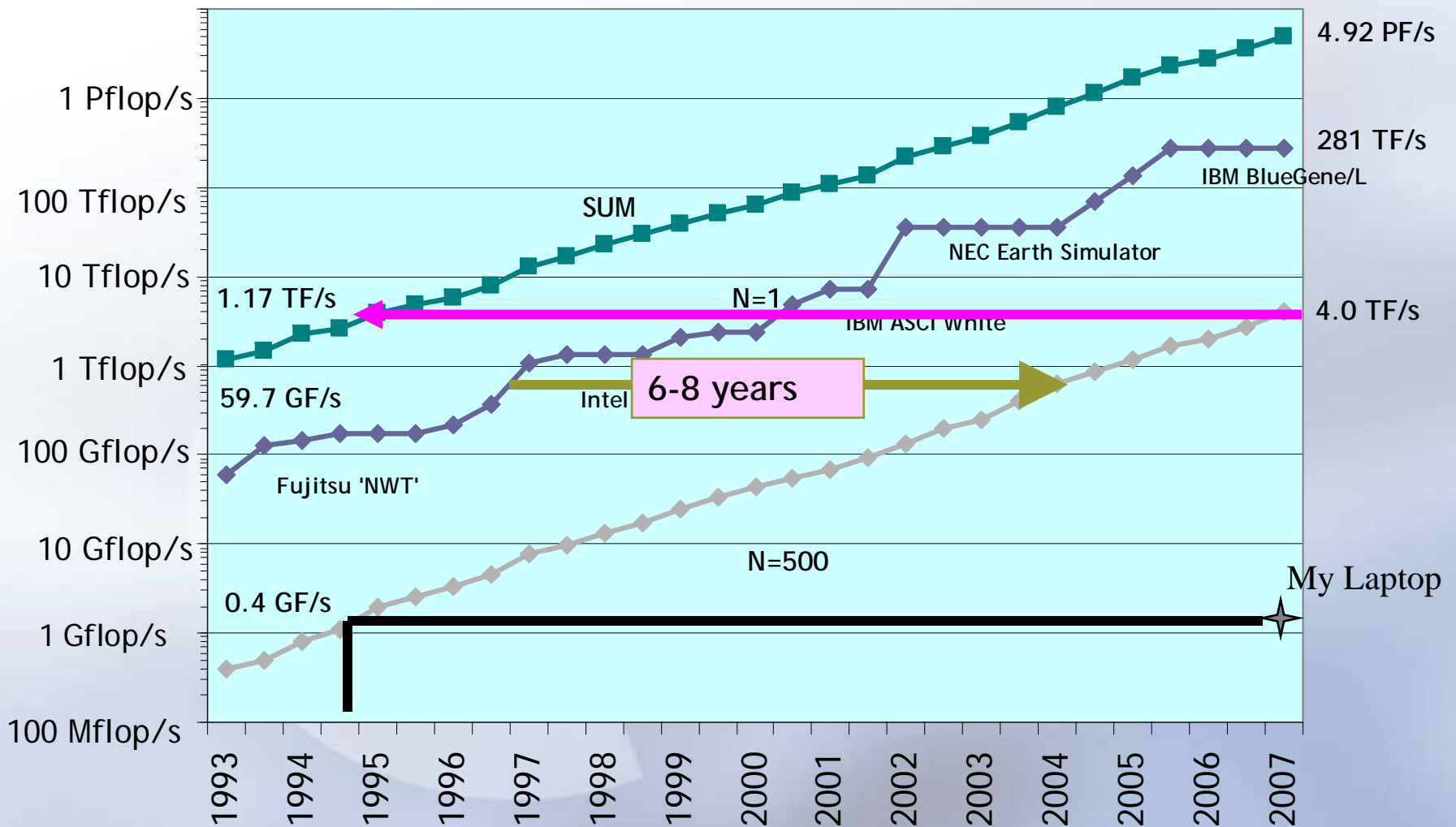
- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

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

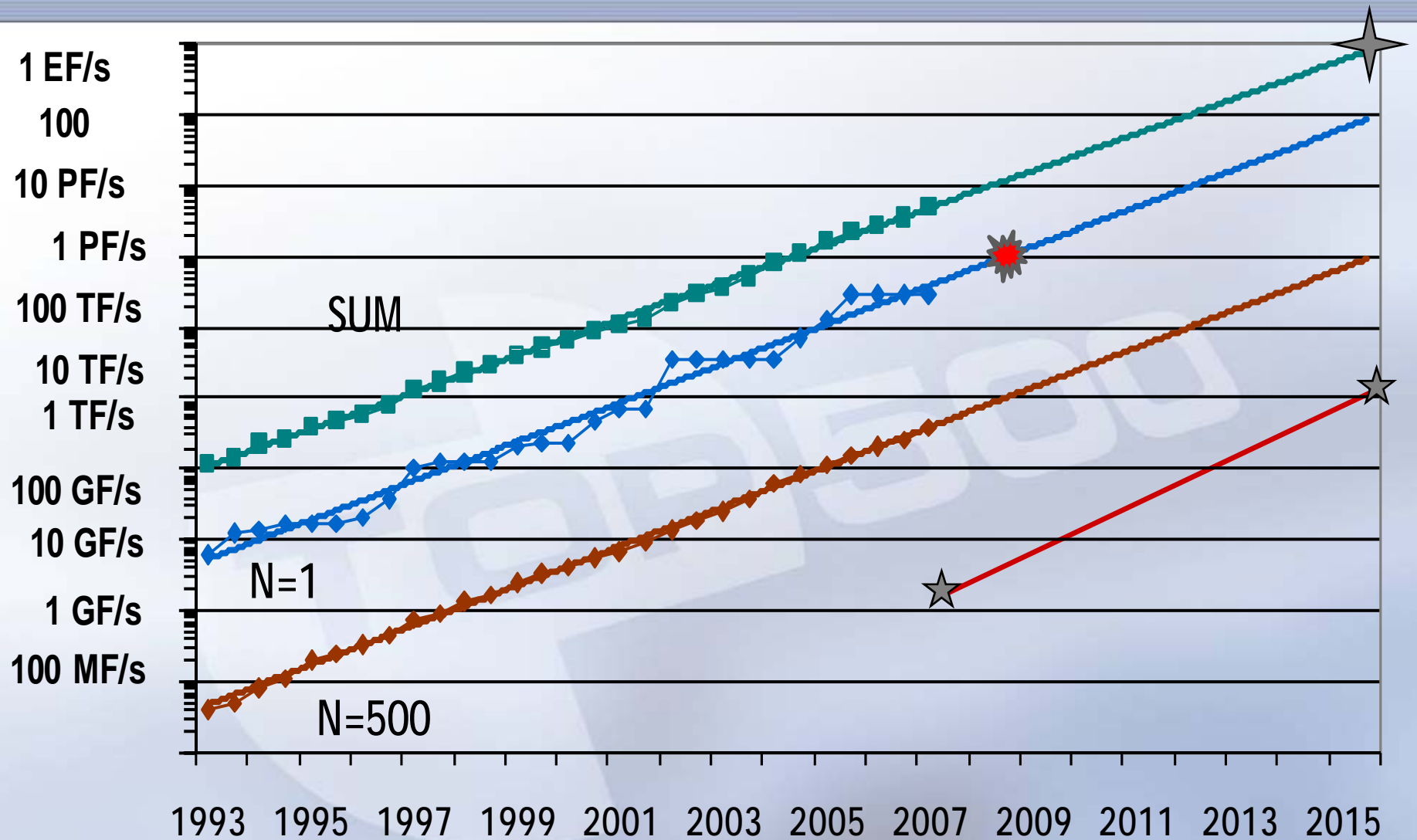


- Updated twice a year
 - SC'xy in the States in November
 - Meeting in Germany in June
- All data available from www.top500.org

Performance Development



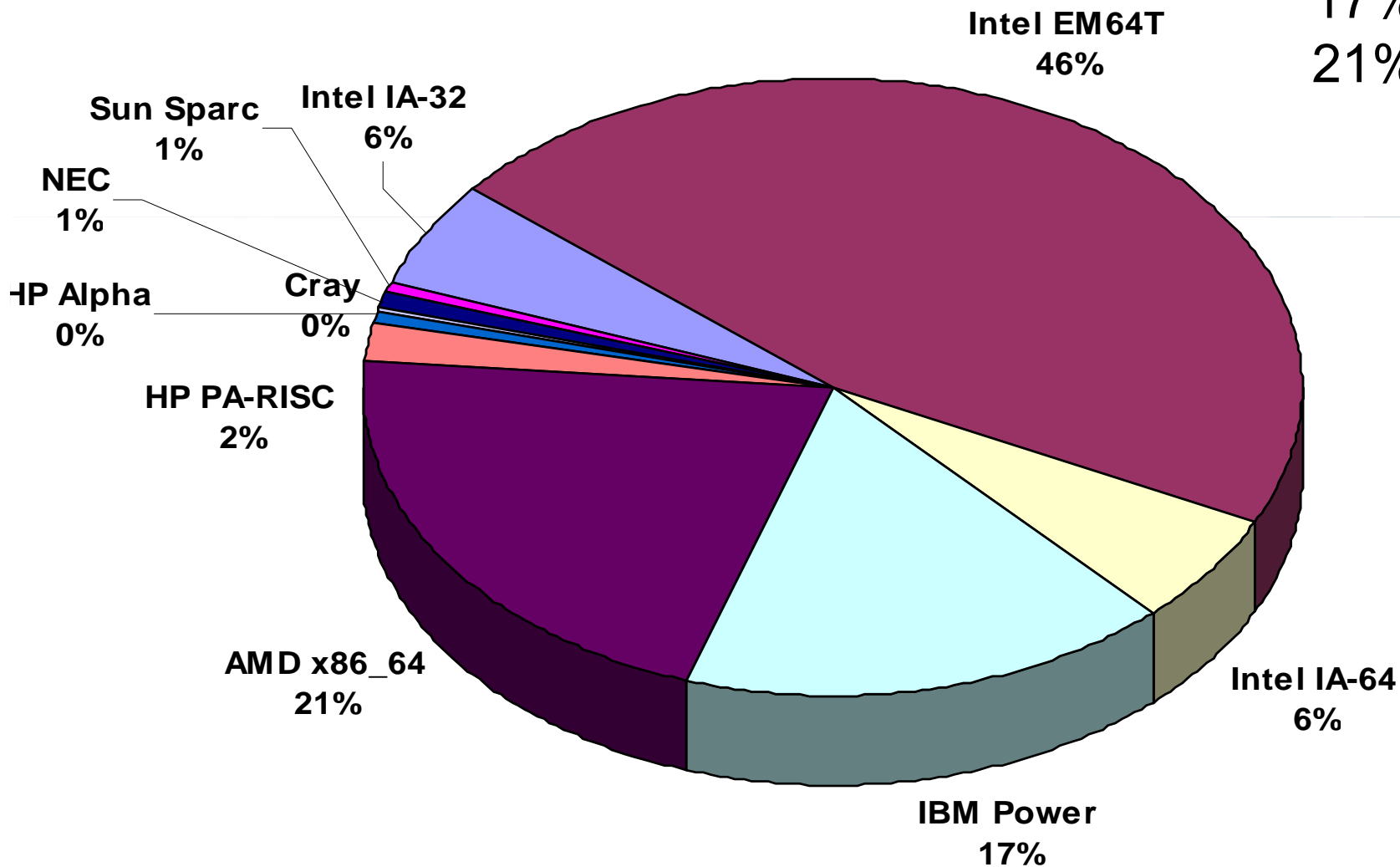
Performance Projection



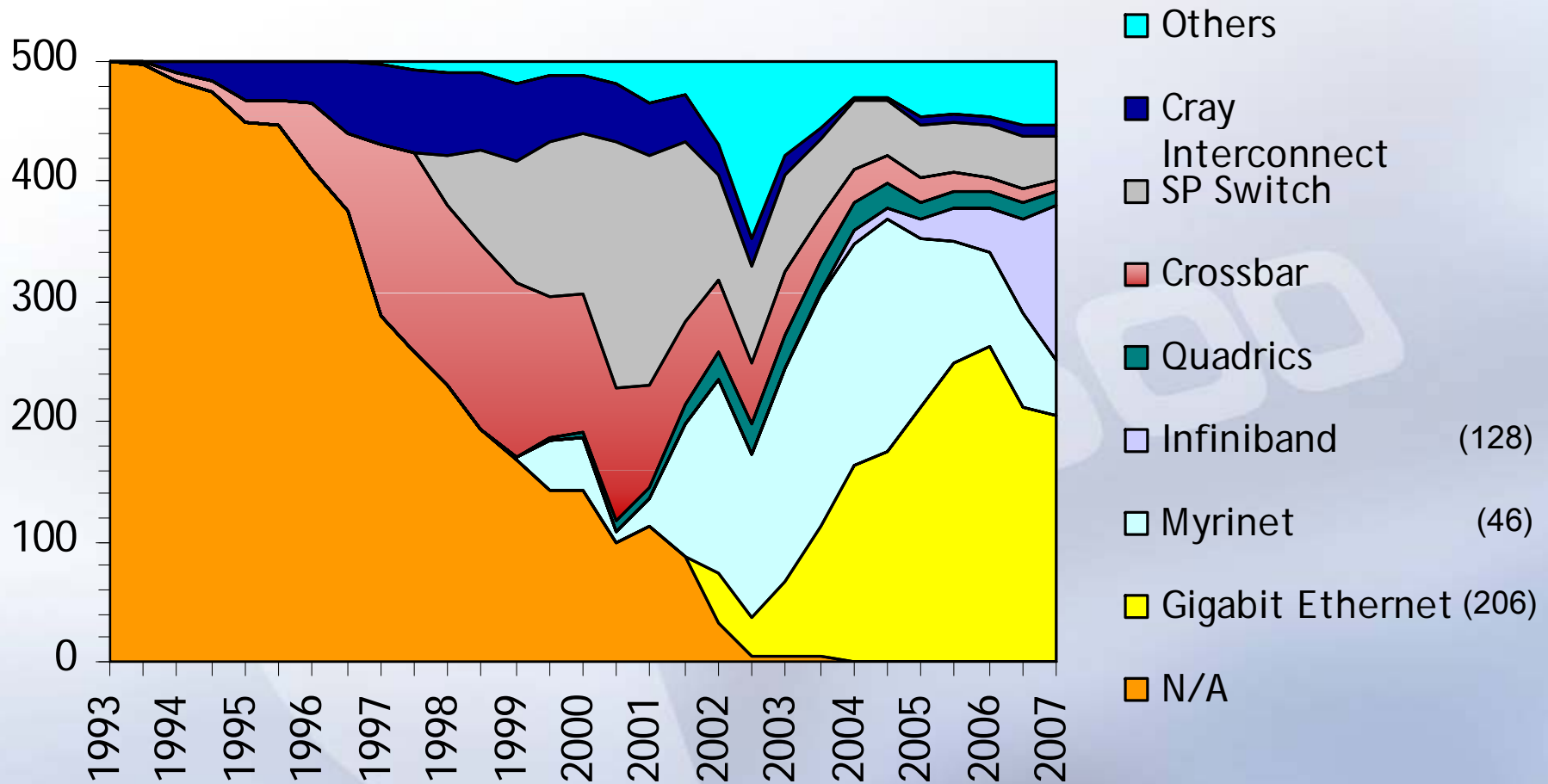


Chips Used in Each of the 500 Systems

96% = 58% Intel
17% IBM
21% AMD



Interconnects / Systems

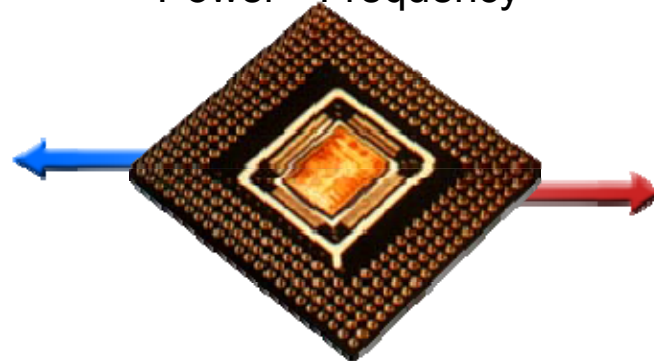


GigE + Infiniband + Myrinet = 76%

ICLU Increasing CPU Performance:

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

$$\text{Power} \propto \text{Frequency}^3$$



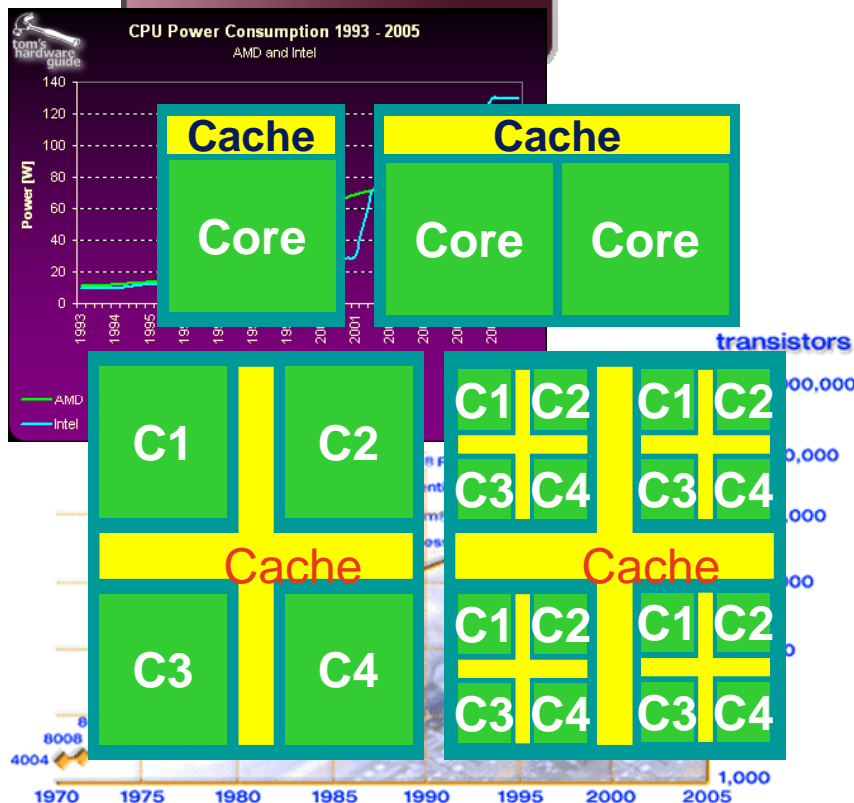
Increase
Clock Rate
& Transistor
Density

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.

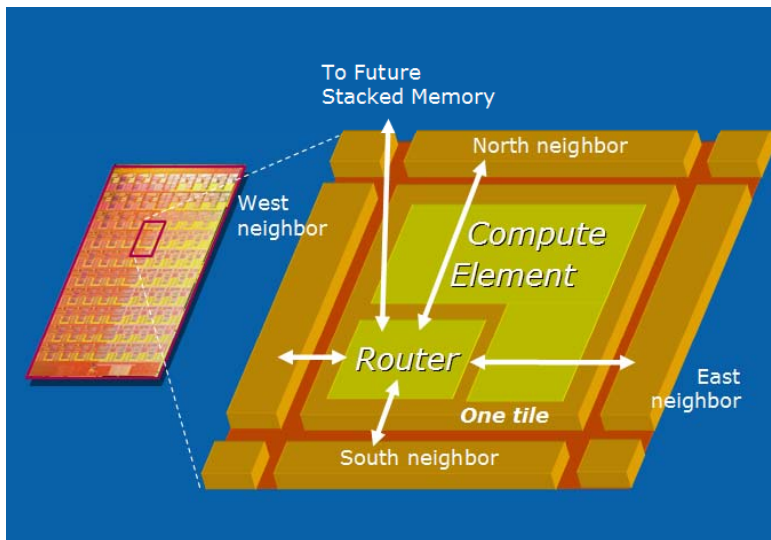
However, the number of gates on a chip will continue to increase.





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.

E-MAIL

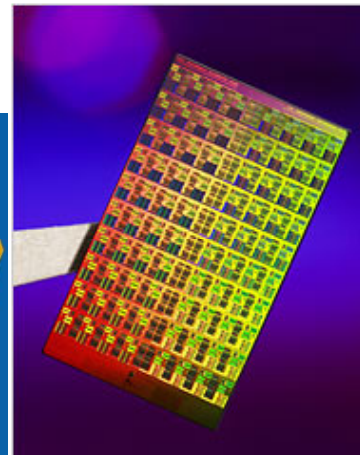
PRINT

REPRINTS

SAVE

SHARE

ARTICLE TOOLS
 SPONSORED BY
FOR YOUR CONSIDERATION
 LITTLE MISS SUNSHINE



The Teraflop Chip has 80 separate processing engines and takes advantage of manufacturing technology that Intel introduced last month.

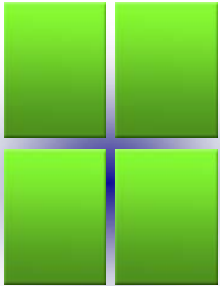
The new processor, which the company first described as a Teraflop Chip at a conference last year, will be detailed in a technical paper to be presented on the opening day of the International Solid States Circuits Conference, beginning here on Monday.

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.

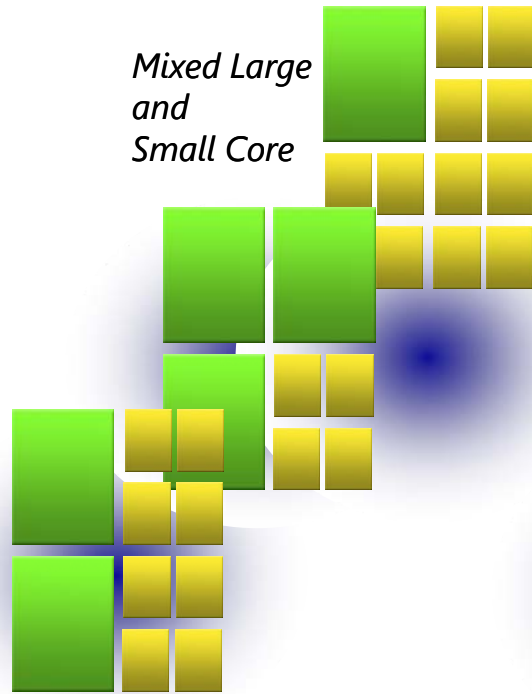


What's Next?

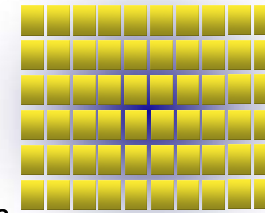
All Large Core



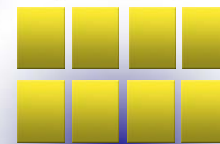
Mixed Large and Small Core



Many Small Cores

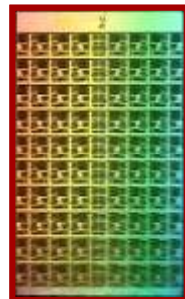


All Small Core

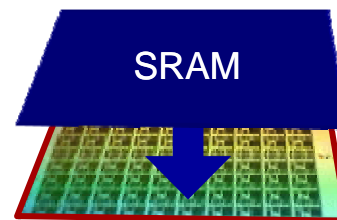


Different Classes of Chips
Home
Games / Graphics
Business
Scientific

Many Floating-Point Cores



+ 3D Stacked Memory



Major Changes to Software

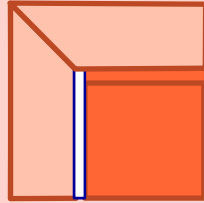
- **Must rethink the design of our software**
 - **Another disruptive technology**
 - Similar to what happened with cluster computing and message passing
 - **Rethink and rewrite the applications, algorithms, and software**
- **Numerical libraries for example will change**
 - **For example, both LAPACK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



A New Generation of Software:

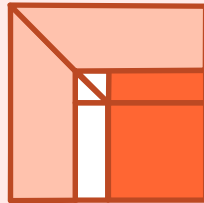
Algorithms follow hardware evolution in time

LINPACK (80's)
(Vector operations)



Rely on
- Level-1 BLAS
operations

LAPACK (90's)
(Blocking, cache
friendly)



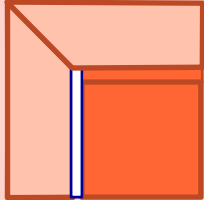
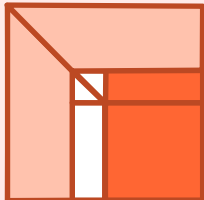
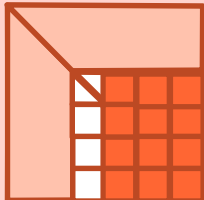
Rely on
- Level-3 BLAS
operations



A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Algorithms follow hardware evolution in time

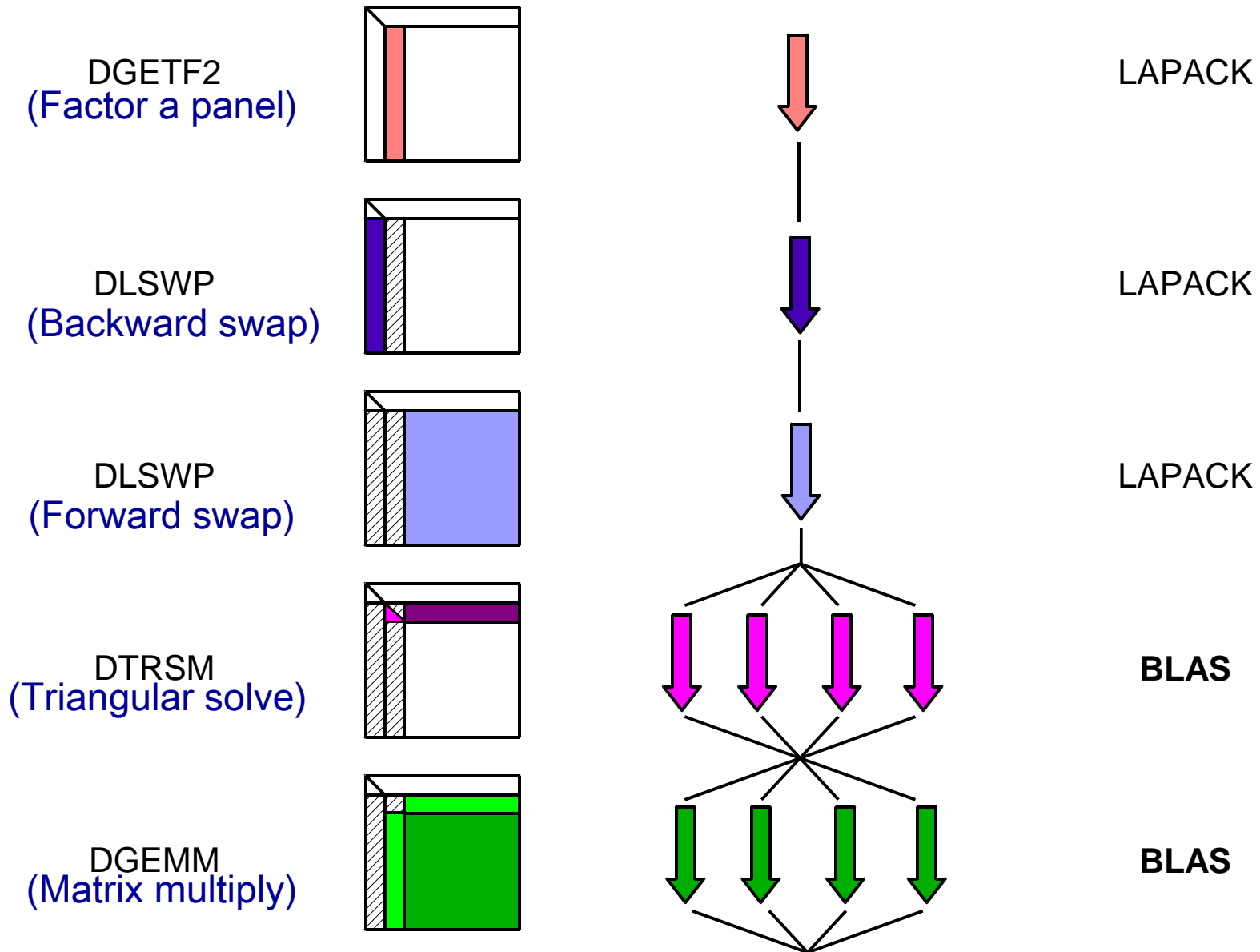
LINPACK (80's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (90's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels

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.

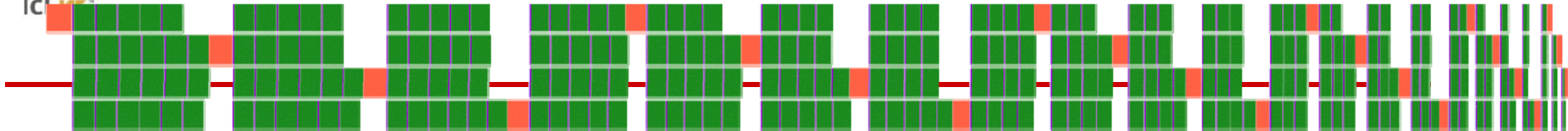
Steps in the LAPACK LU





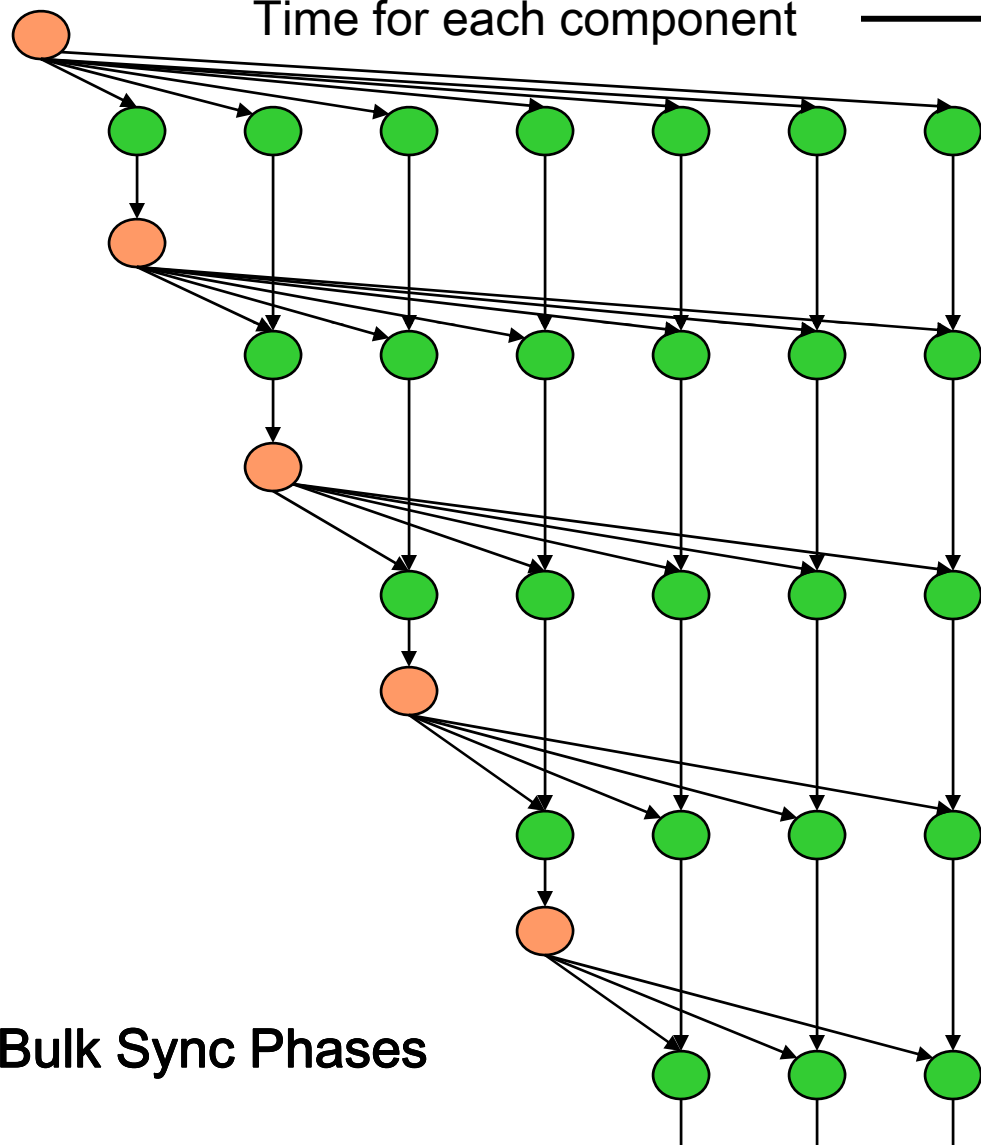
LU Timing Profile (4 processor system)

Threads – no lookahead



Time for each component

1D decomposition and SGI Origin



Bulk Sync Phases

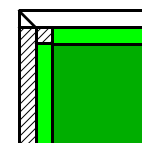
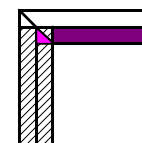
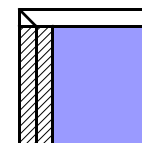
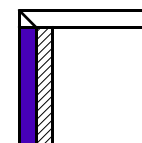
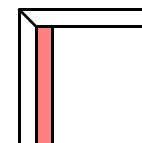
DGETF2






DLSWP

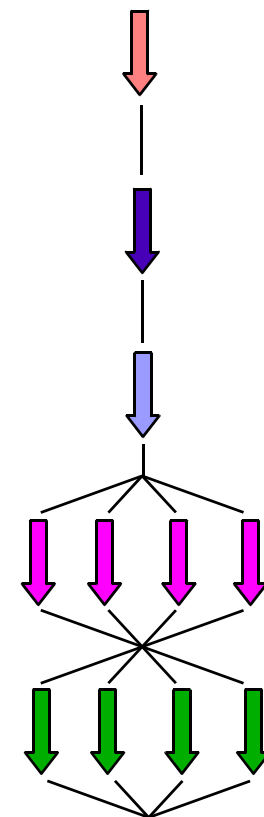
DLSWP

DTRSM

DGEMM

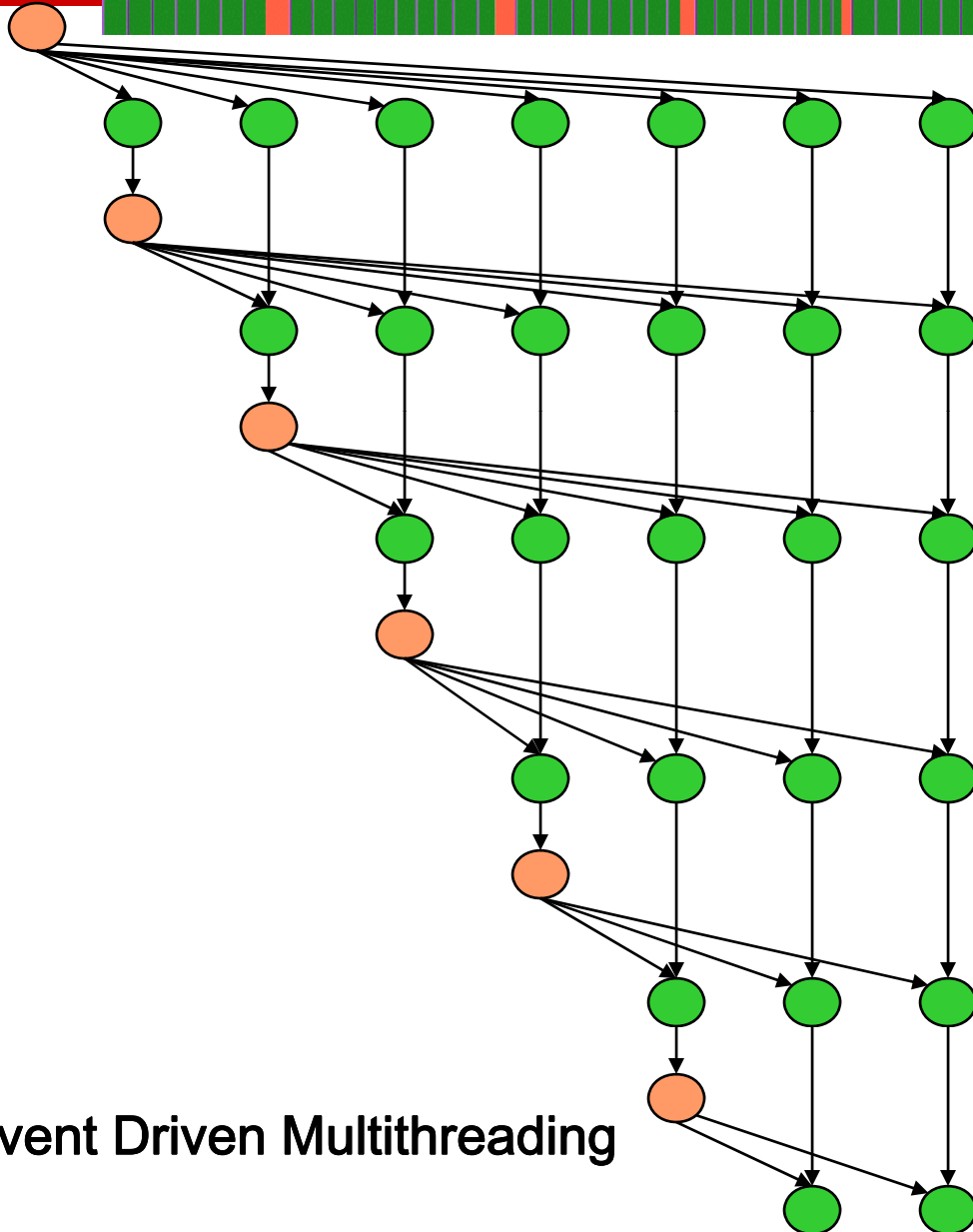
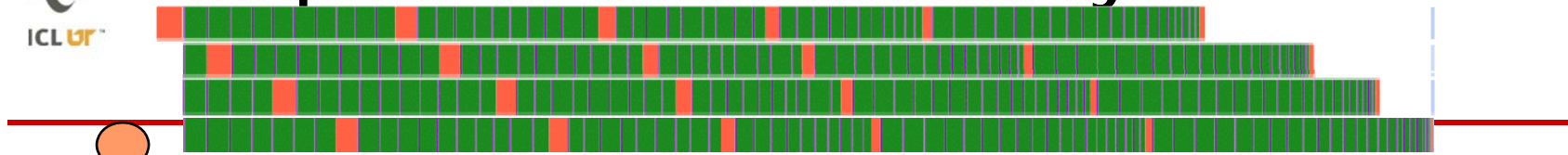


-  DGETF2
-  DLASWP(L)
-  DLASWP(R)
-  DTRSM
-  DGEMM





Adaptive Lookahead - Dynamic



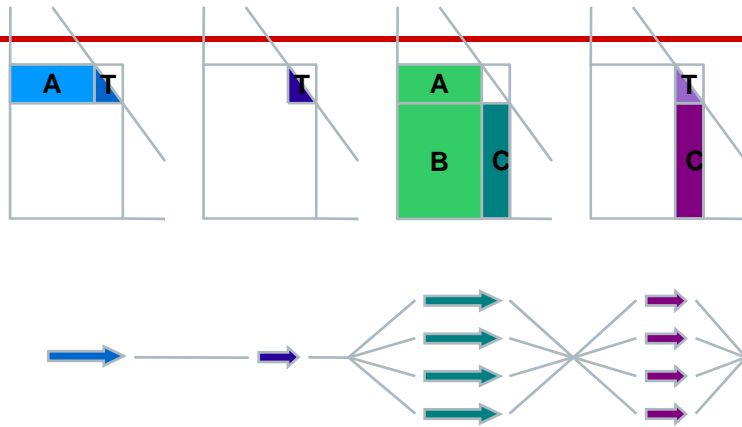
```
while(1)
  fetch_task();
  switch(task.type) {
    case PANEL:
      dgetf2();
      update_progress();
    case COLUMN:
      dlaswp();
      dtrsm();
      dgemm();
      update_progress();
    case END:
      for()
        dlaswp();
      return;
  }
}
```

Reorganizing algorithms to use this approach

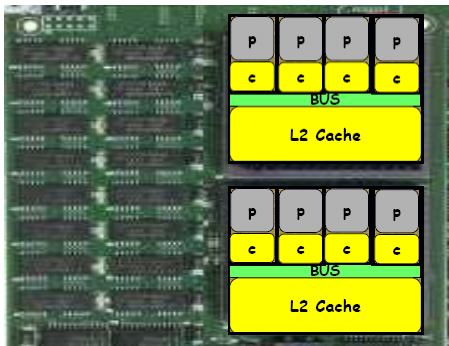
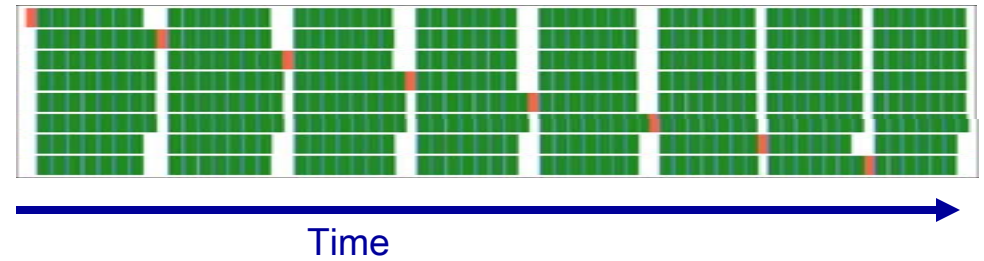
Event Driven Multithreading



Fork-Join vs. Dynamic Execution



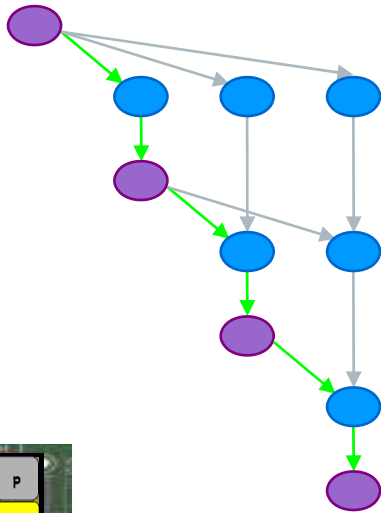
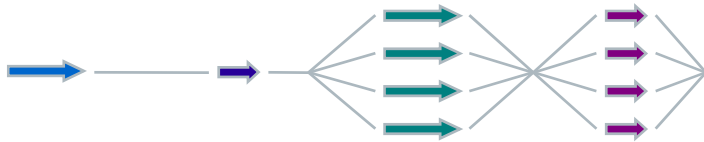
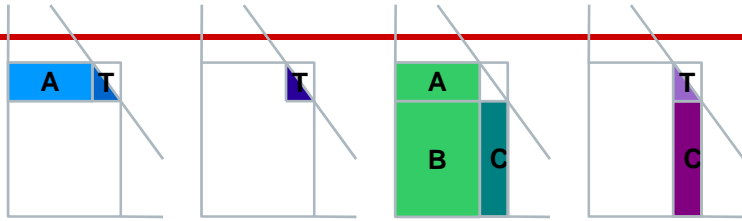
Fork-Join – parallel BLAS



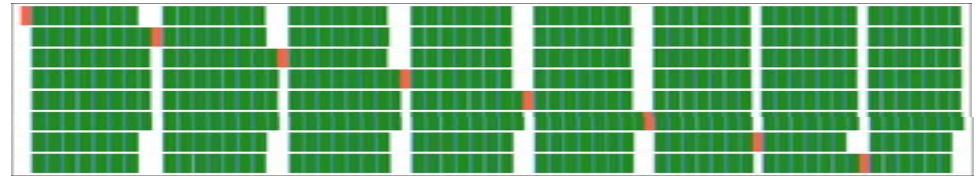
Experiments on
Intel's Quad Core Clovertown
with 2 Sockets w/ 8 Treads



Fork-Join vs. Dynamic Execution

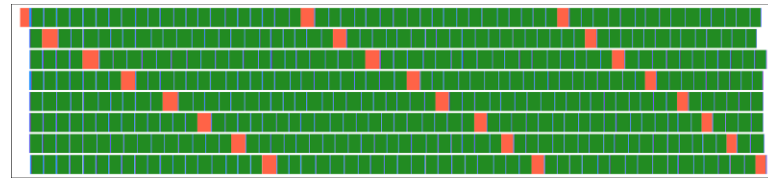


Fork-Join – parallel BLAS

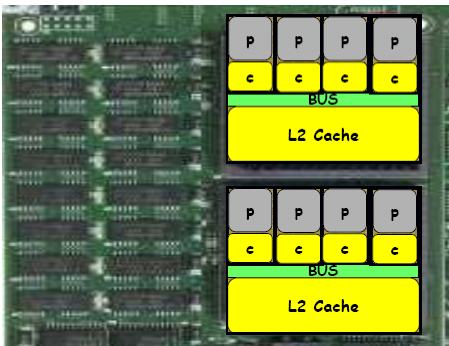


Time

DAG-based – dynamic scheduling



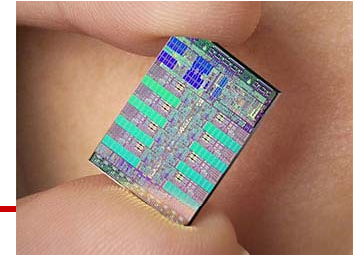
Time saved



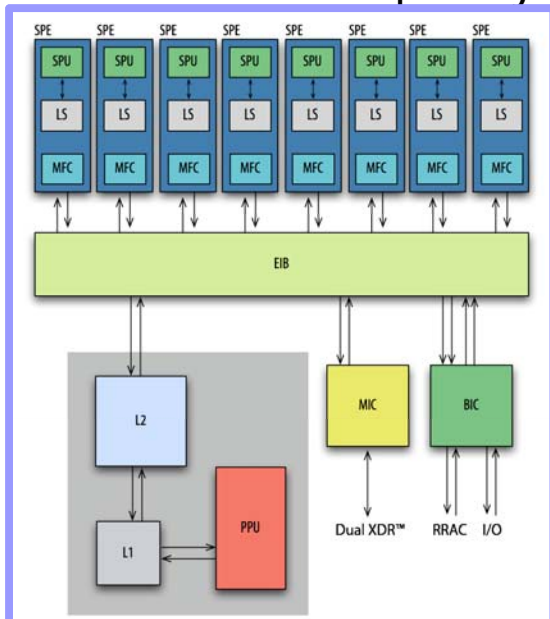
Experiments on
 Intel's Quad Core Clovertown
 with 2 Sockets w/ 8 Treads



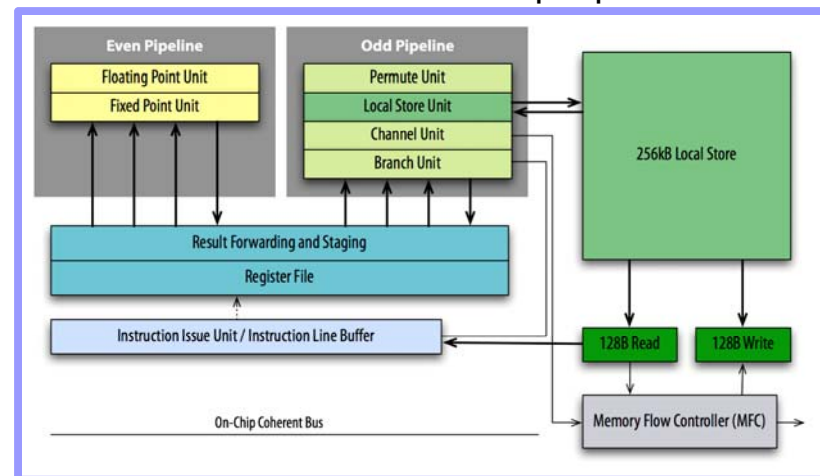
With the Hype on Cell & PS3 We Became Interested



- 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

	Size	SGEMM/ DGEMM	Size	SGEMV/ DGEMV
AMD Opteron 246	3000	2.00	5000	1.70
UltraSparc-Ile	3000	1.64	5000	1.66
Intel PIII Coppermine	3000	2.03	5000	2.09
PowerPC 970	3000	2.04	5000	1.44
Intel Woodcrest	3000	1.81	5000	2.18
Intel XEON	3000	2.04	5000	1.82
Intel Centrino Duo	3000	2.71	5000	2.21

Single precision is faster because:

- Higher parallelism in SSE/vector units
- Reduced data motion
- Higher locality in cache

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, $10^{\pm 38}$)
 - 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.

$L U = lu(A)$	SINGLE	$O(n^3)$
$x = L \setminus (U \setminus b)$	SINGLE	$O(n^2)$
$r = b - Ax$	DOUBLE	$O(n^2)$
WHILE $\ r \ $ not small enough		
$z = L \setminus (U \setminus r)$	SINGLE	$O(n^2)$
$x = x + z$	DOUBLE	$O(n^1)$
$r = b - Ax$	DOUBLE	$O(n^2)$
END		

Mixed-Precision Iterative Refinement

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

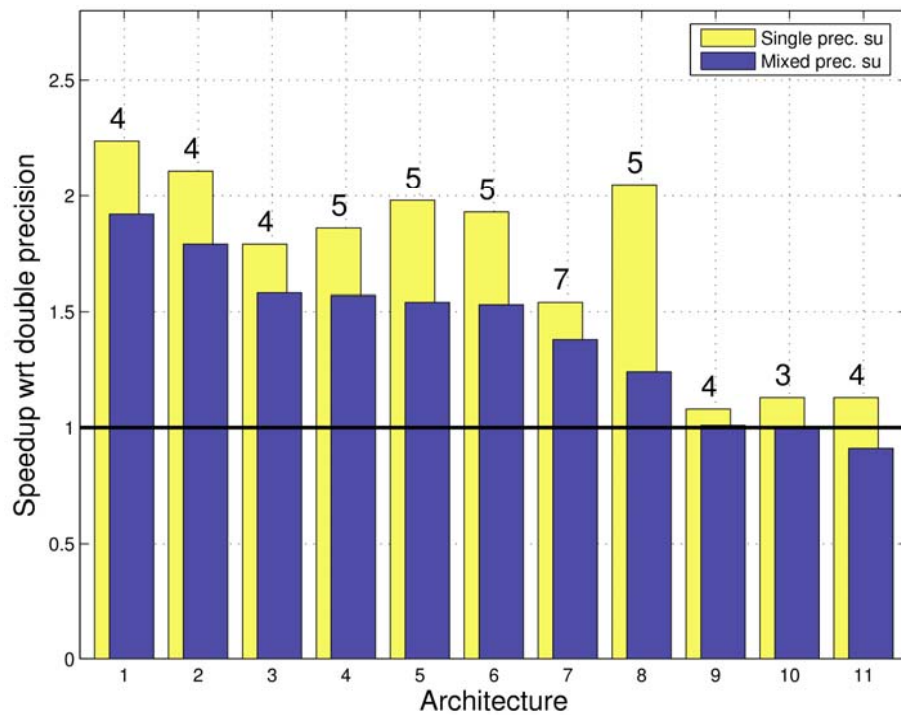
$L U = lu(A)$	SINGLE	$O(n^3)$
$x = L \setminus (U \setminus b)$	SINGLE	$O(n^2)$
$r = b - Ax$	DOUBLE	$O(n^2)$
WHILE $\ r \ $ not small enough		
$z = L \setminus (U \setminus r)$	SINGLE	$O(n^2)$
$x = x + z$	DOUBLE	$O(n^1)$
$r = b - Ax$	DOUBLE	$O(n^2)$
END		

- 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; $O(10^8)$



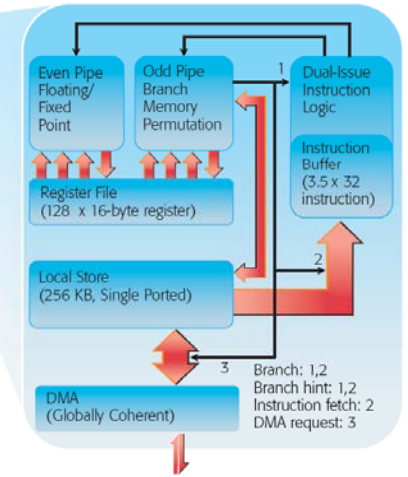
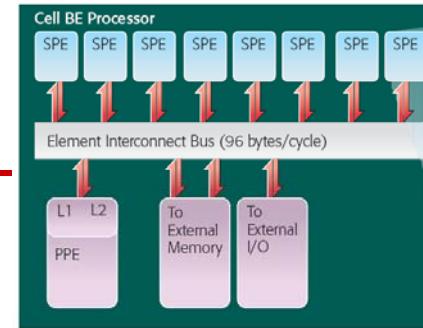
Results for Multiple Precision Iterative Refinement



	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)

New routines in LAPACK that do this for LU and LL^T

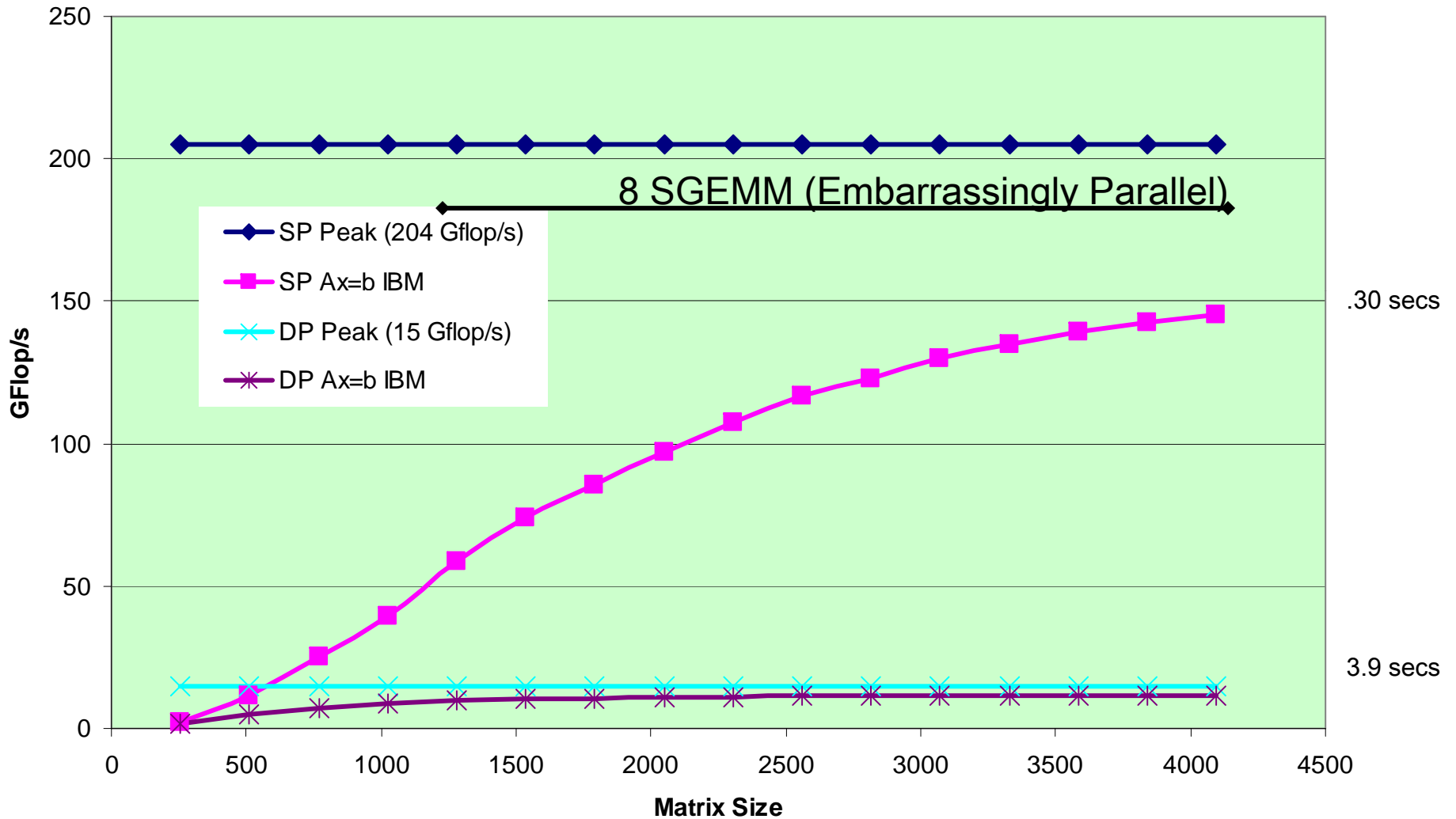
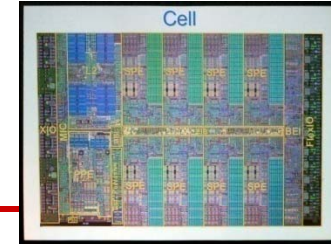
What about the Cell?



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

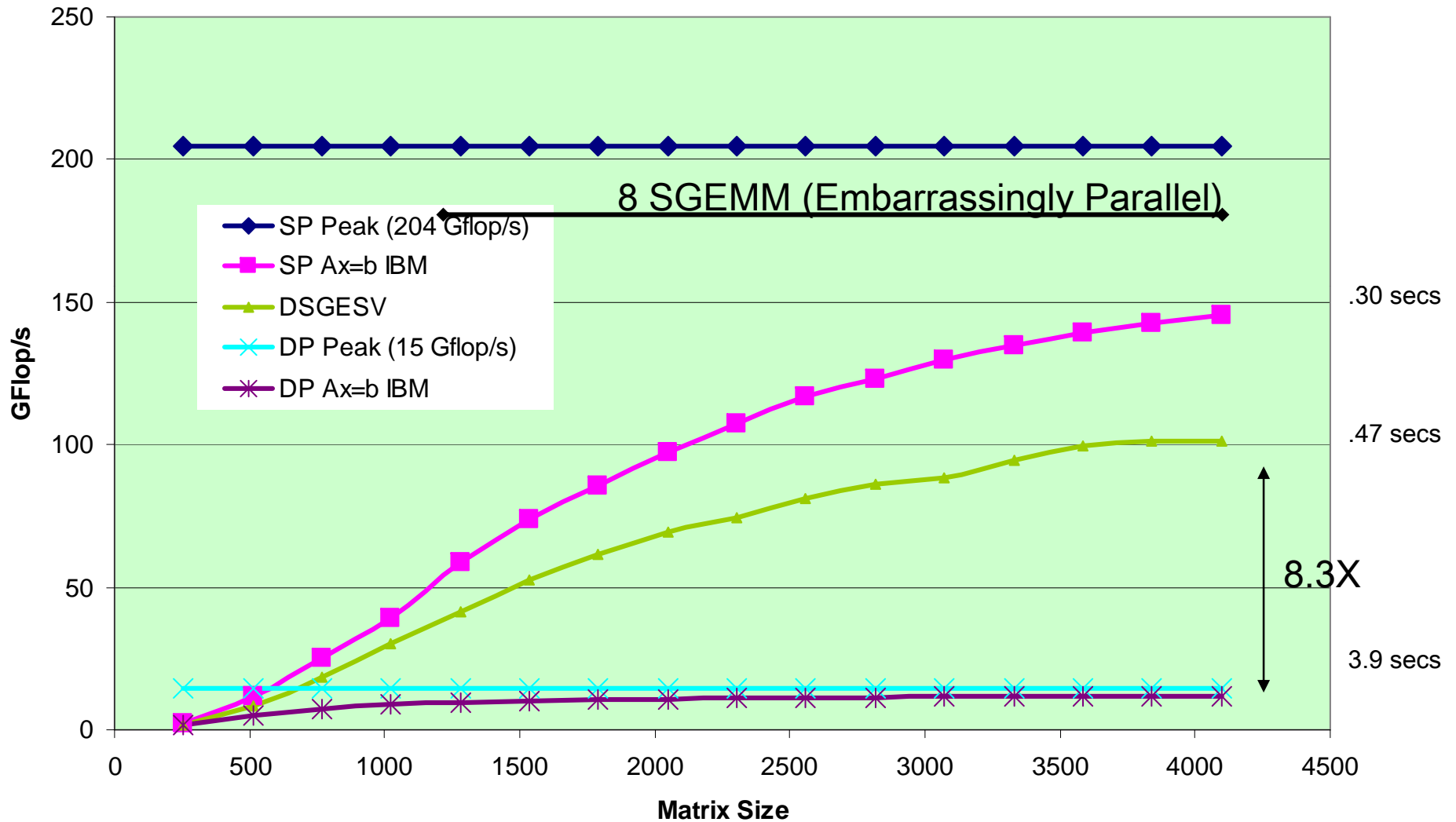
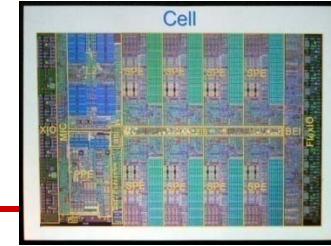


IBM Cell 3.2 GHz, $Ax = b$

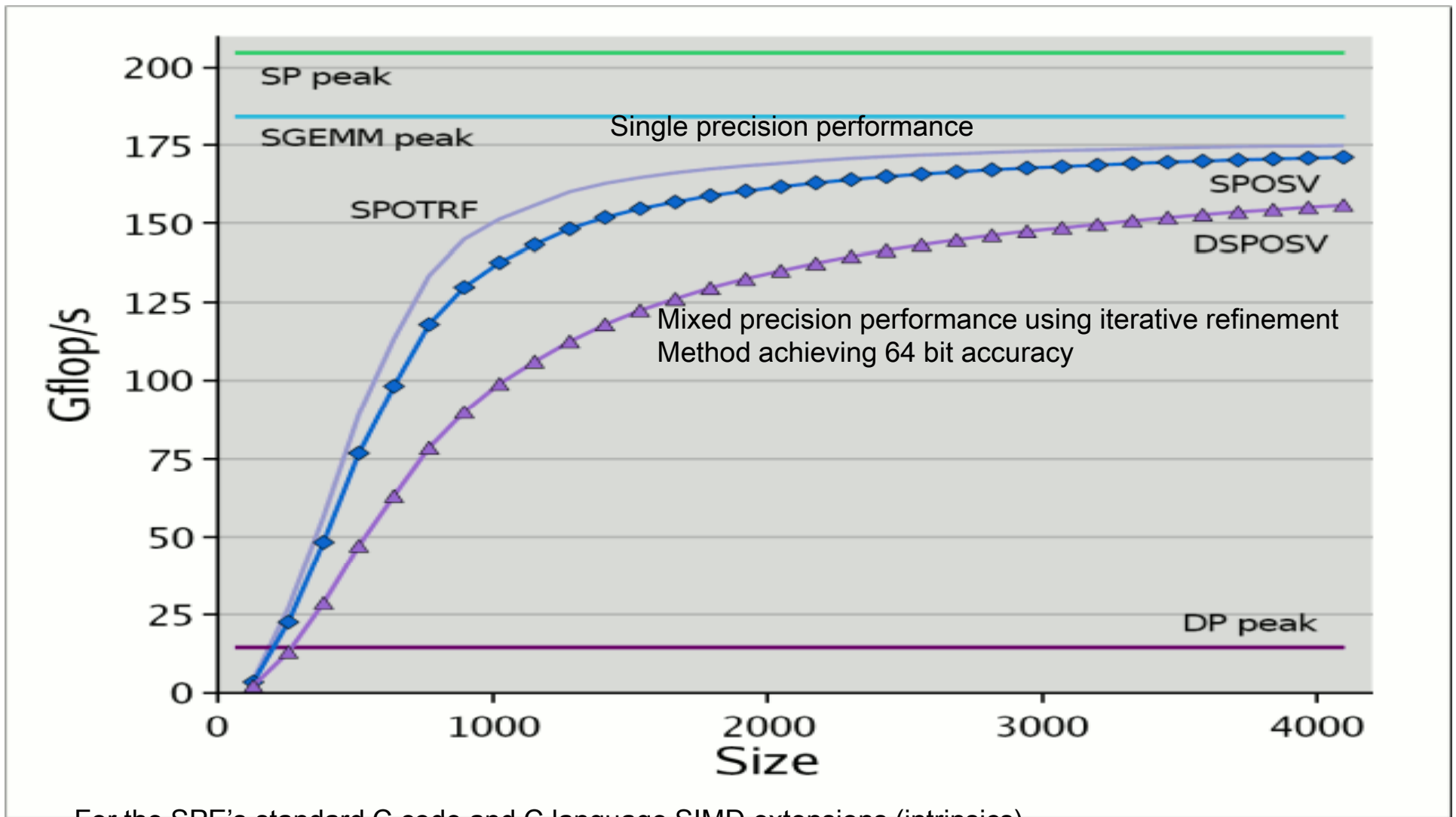




IBM Cell 3.2 GHz, $Ax = b$

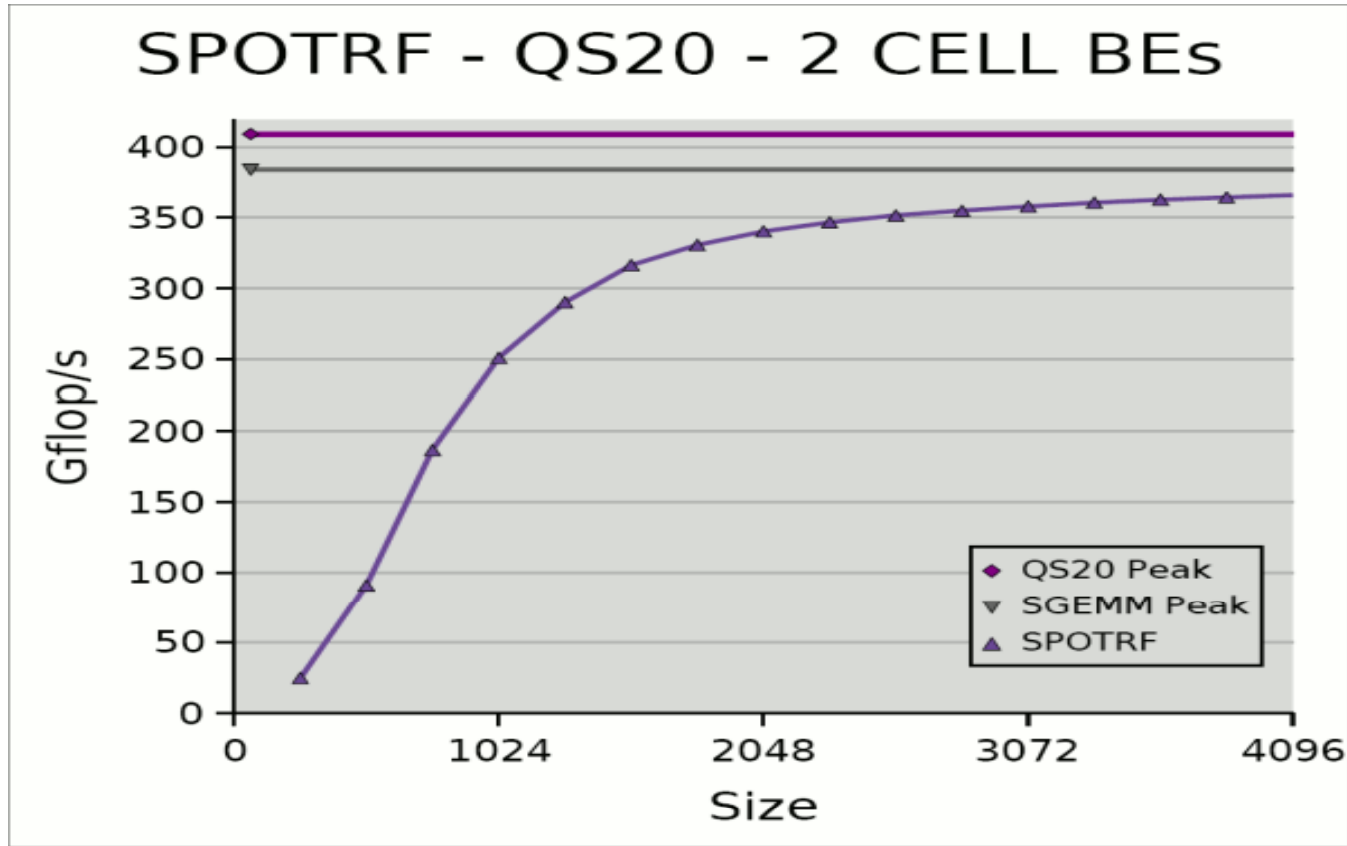


Cholesky on the Cell, $Ax=b$, $A=A^T$, $x^T Ax > 0$



For the SPE's standard C code and C language SIMD extensions (intrinsics)

Cholesky - Using 2 Cell Chips





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.

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

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

Correction = - A\((b - Ax)



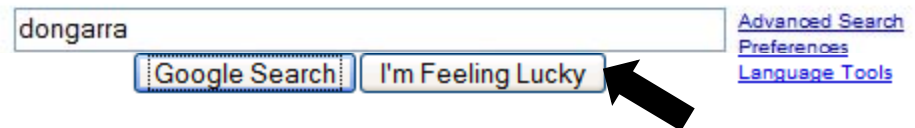
Conclusions

- 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



[Advertising Programs](#) - [Business Solutions](#) - [About Google](#)

©2007 Google