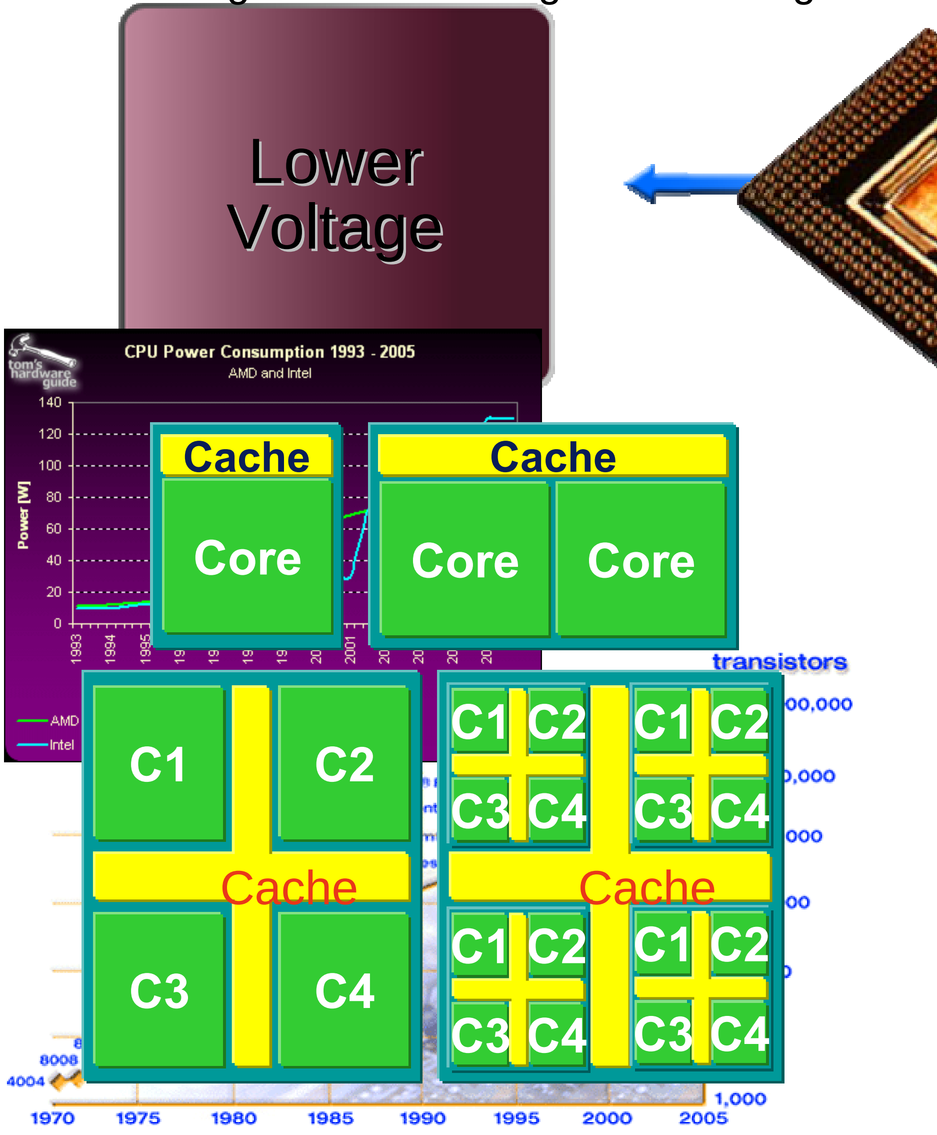
LACSI SYMPOSIUM 2006

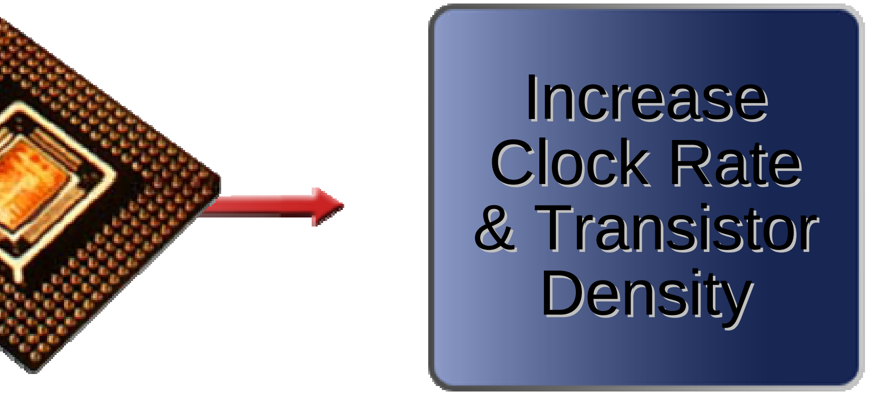
The Impact of Multicore on Math Software and Exploiting Single Precision in Obtaining Double Precision

Jack Dongarra
University of Tennessee
and
Oak Ridge National Laboratory

Increasing CPU Performance: A Delicate Balancing Act

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


Lower Voltage

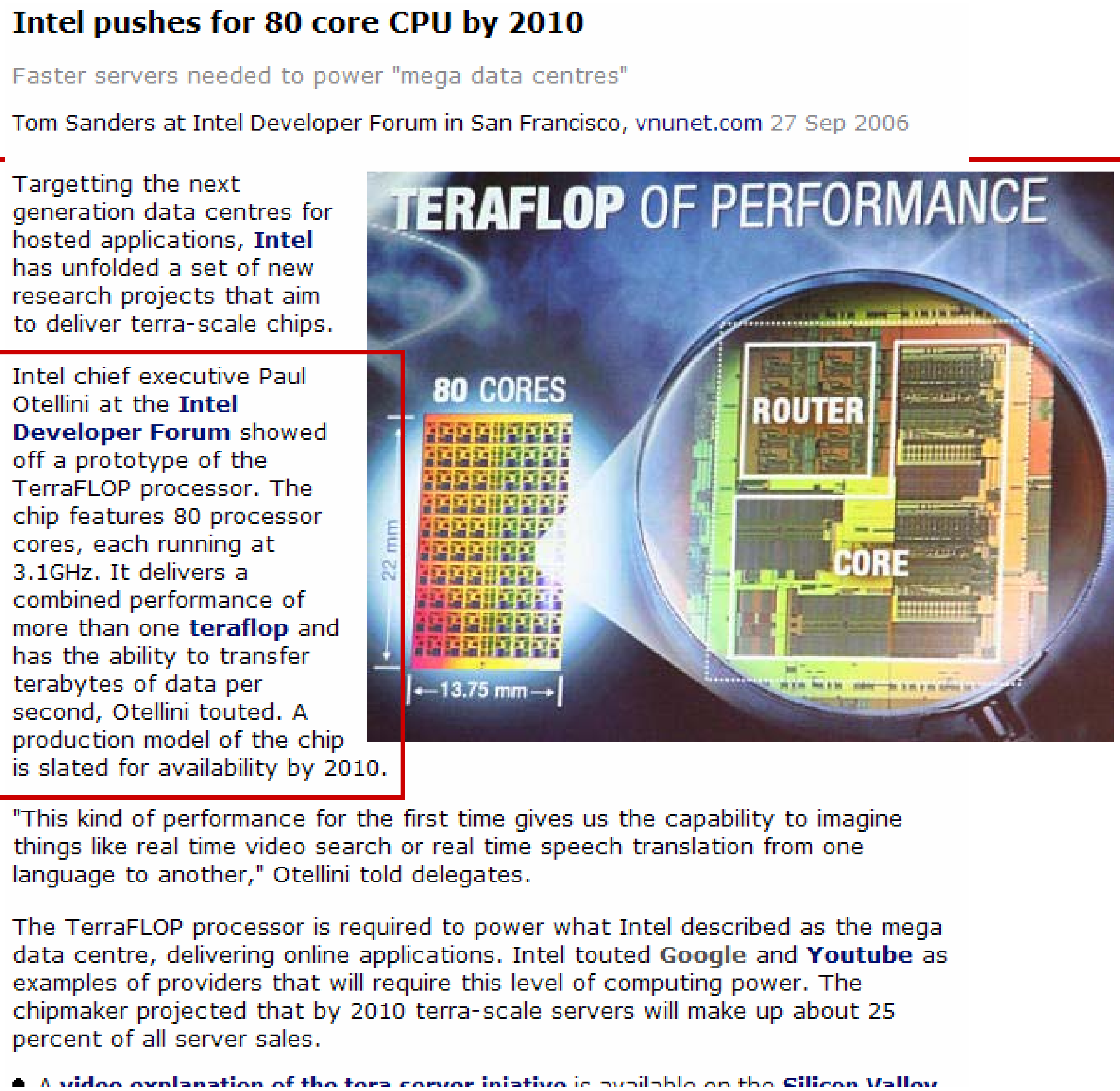

Increase Clock Rate & Transistor Density

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.



Intel pushes for 80 core CPU by 2010

Faster servers needed to power "mega data centres"

Tom Sanders at Intel Developer Forum in San Francisco, vnunet.com 27 Sep 2006

Targetting the next generation data centres for hosted applications, **Intel** has unfolded a set of new research projects that aim to deliver terra-scale chips.

Intel chief executive Paul Otellini at the **Intel Developer Forum** showed off a prototype of the TerraFLOP processor. The chip features 80 processor cores, each running at 3.1GHz. It delivers a combined performance of more than one **teraflop** and has the ability to transfer terabytes of data per second, Otellini touted. A production model of the chip is slated for availability by 2010.

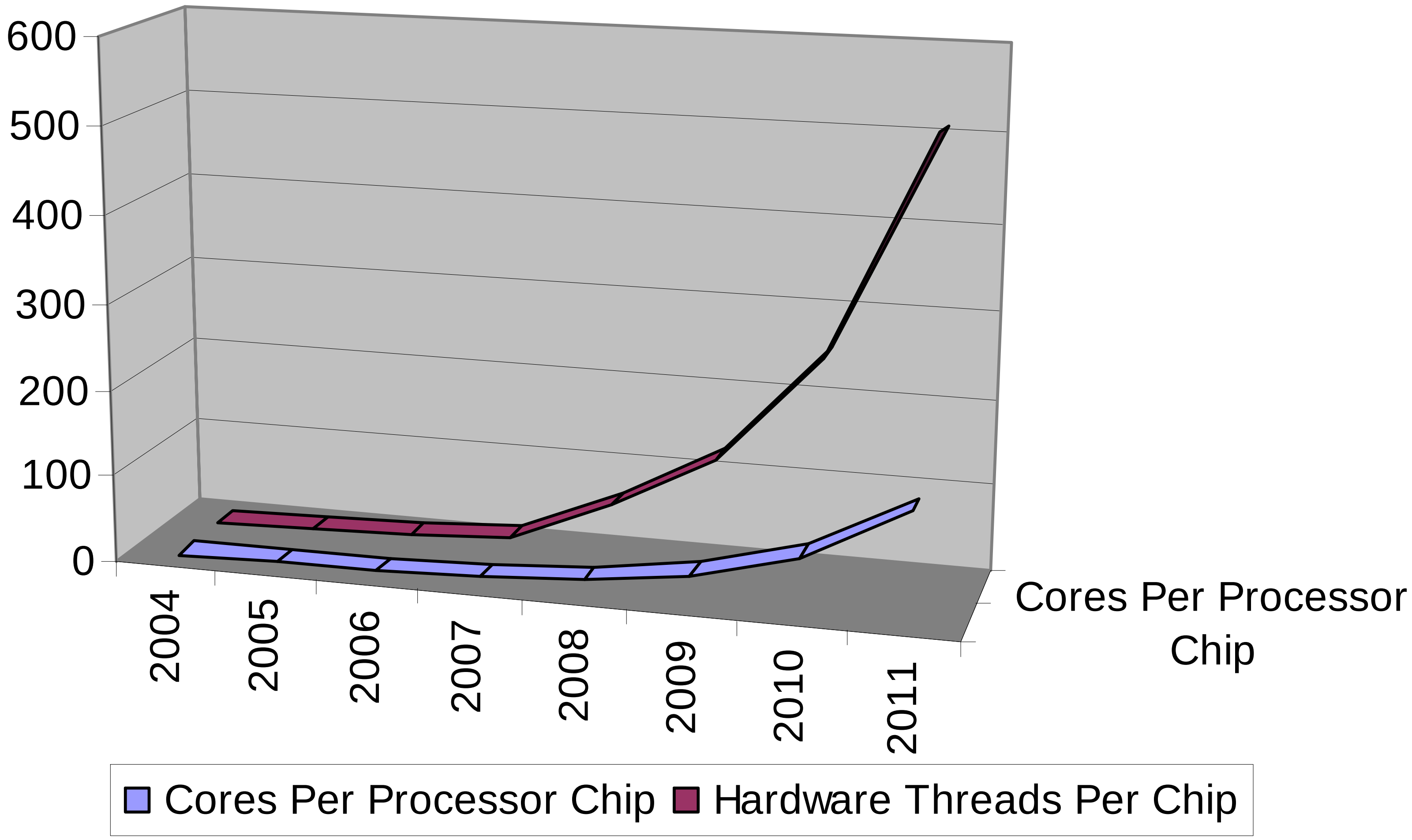
"This kind of performance for the first time gives us the capability to imagine things like real time video search or real time speech translation from one language to another," Otellini told delegates.

The TerraFLOP processor is required to power what Intel described as the mega data centre, delivering online applications. Intel touted **Google** and **Youtube** as examples of providers that will require this level of computing power. The chipmaker projected that by 2010 terra-scale servers will make up about 25 percent of all server sales.

- A **video explanation of the tera server iniative** is available on the **Silicon Valley**

CPU Desktop Trends 2004-2011

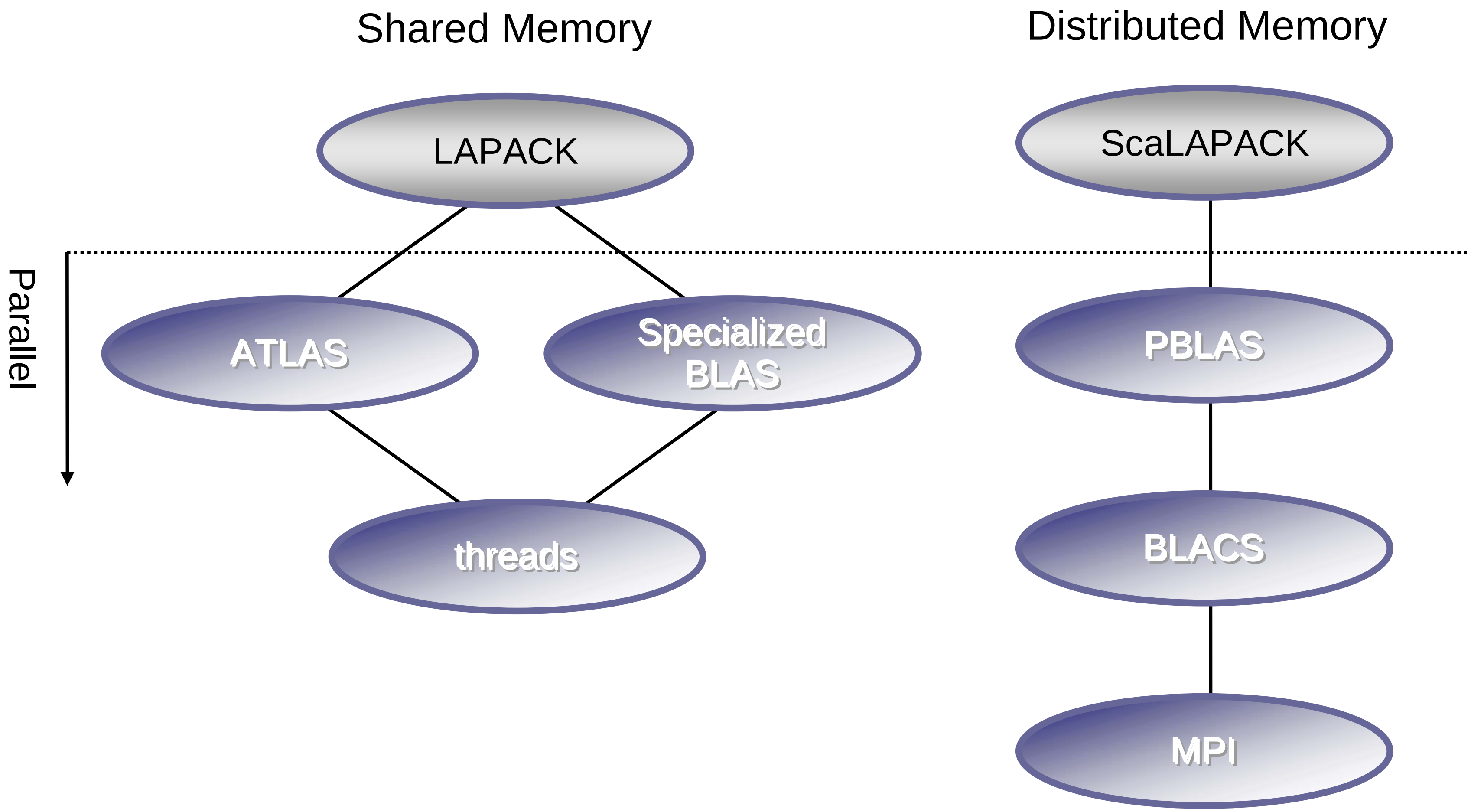
- **Relative processing power will continue to double every 18 months**
- **5 years from now: 128 cores/chip w/512 logical processes per chip**



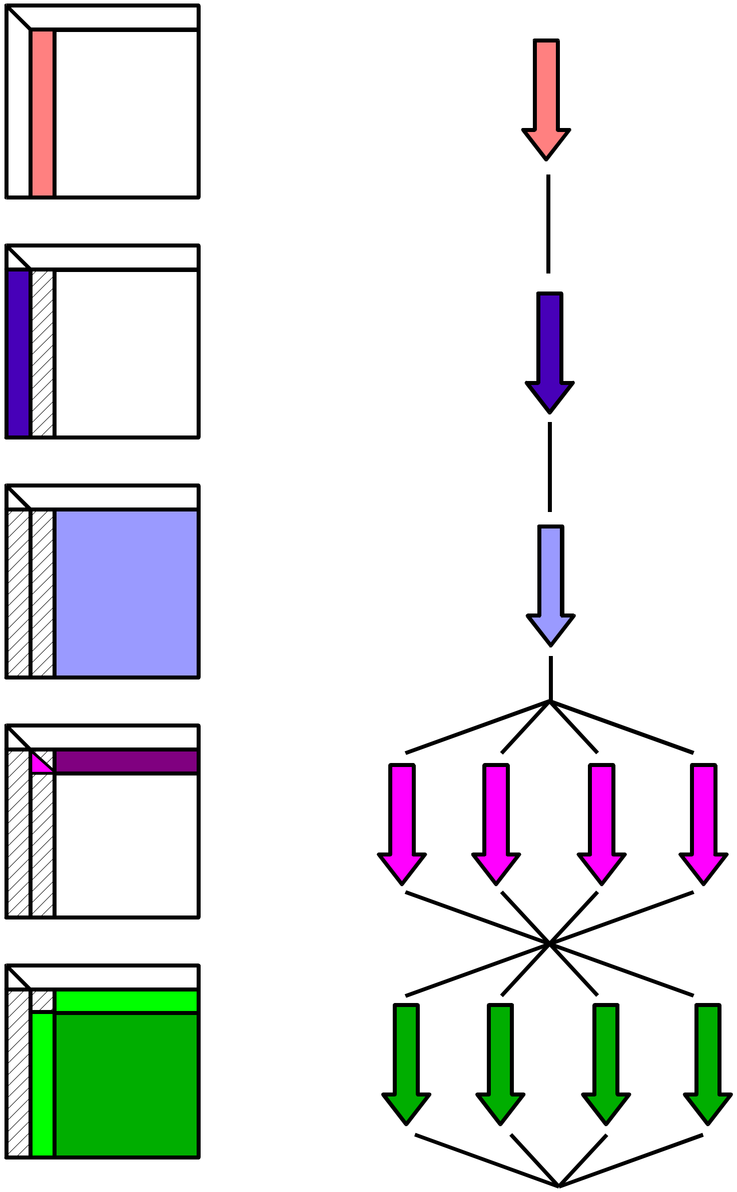
Major Changes to Software

- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with 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

Parallelism in LAPACK / ScaLAPACK



Steps in the LAPACK LU



DGETF2
(Factor a panel)

LAPACK

DLSWP
(Backward swap)

LAPACK

DLSWP
(Forward swap)

LAPACK

DTRSM
(Triangular solve)

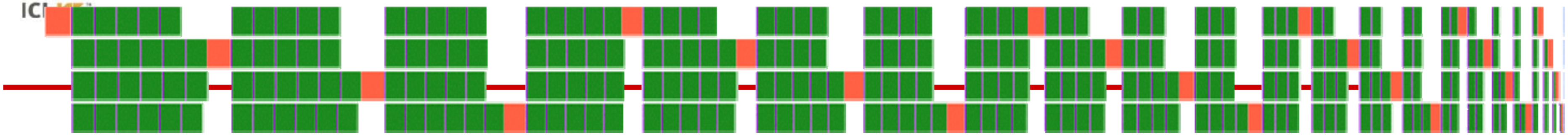
BLAS

DGEMM
(Matrix multiply)

BLAS

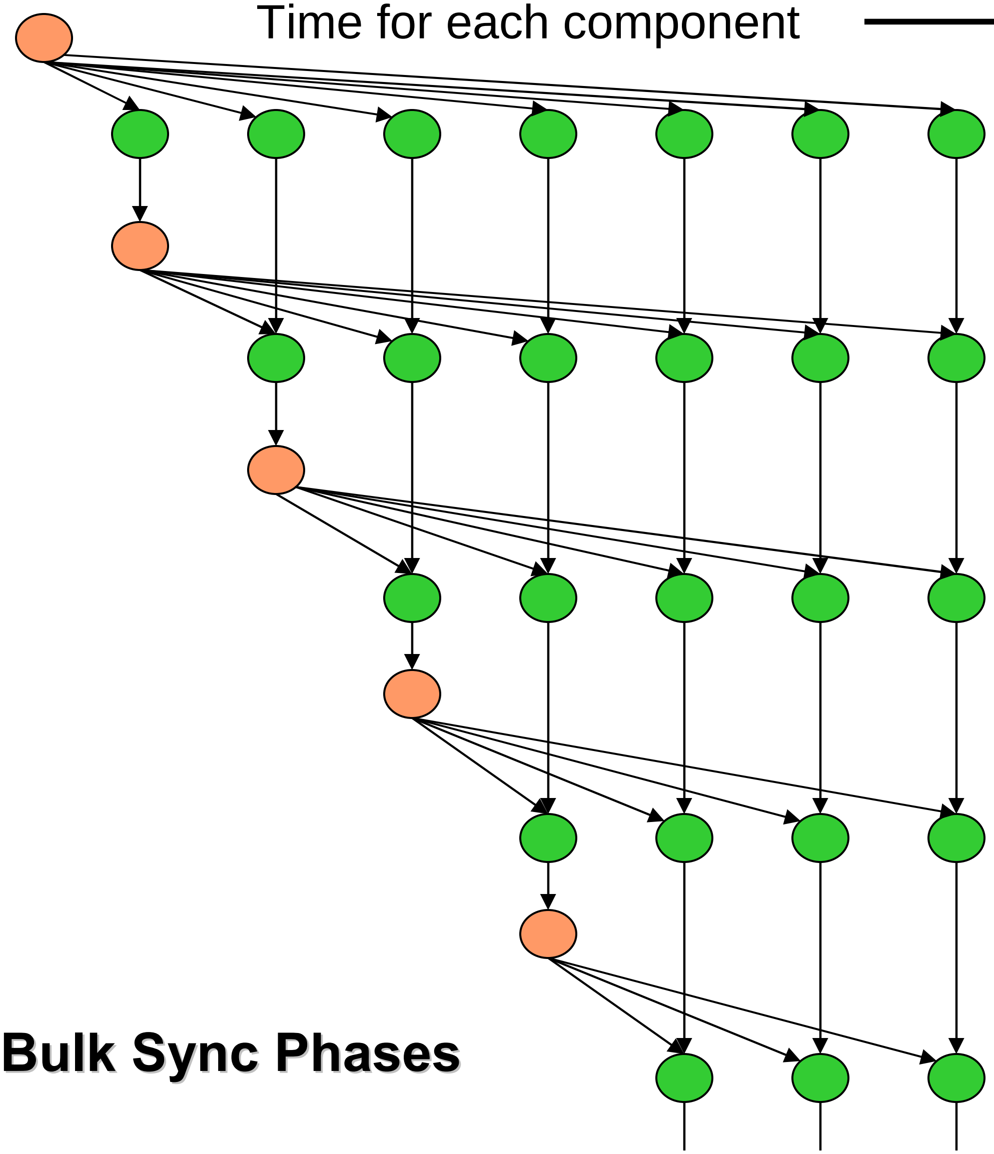
LU Timing Profile (4 processor system)

Threads – no lookahead

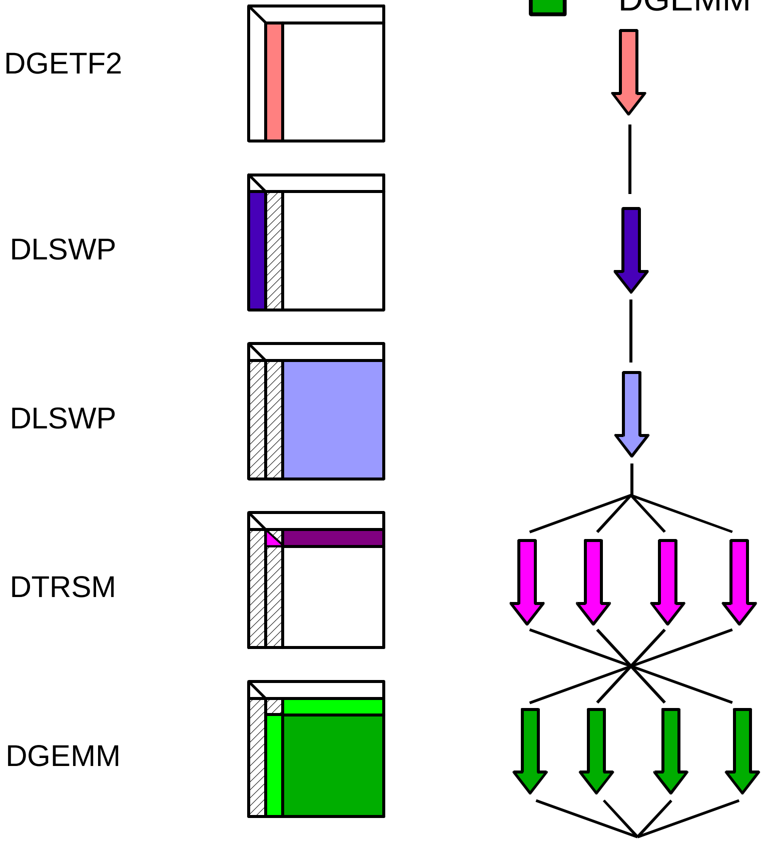


Time for each component → 1D decomposition and SGI Origir

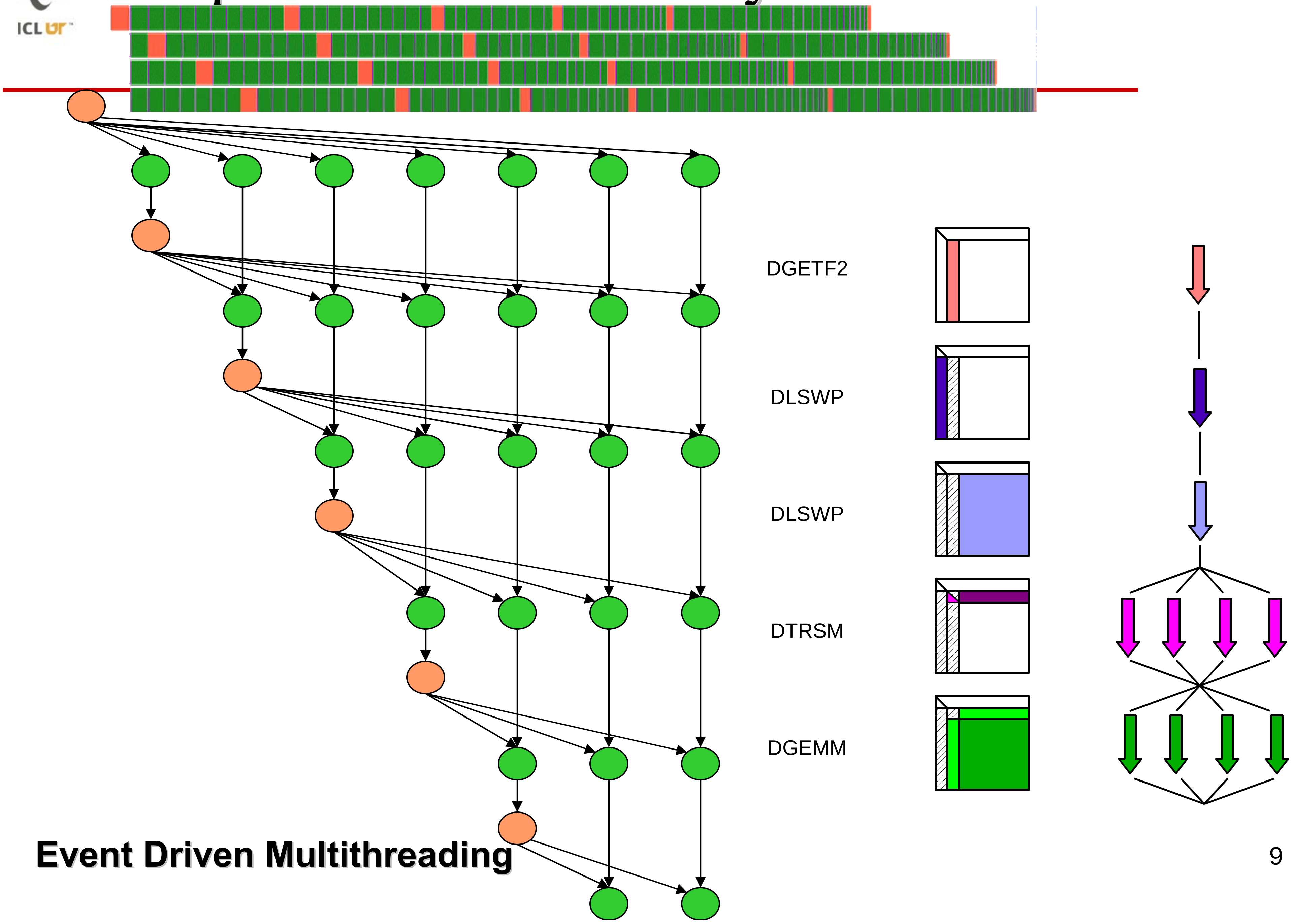
- 🟧 DGETF2
- 🟦 DLASWP(L)
- 🟦 DLASWP(R)
- 🟪 DTRSM
- 🟩 DGEMM



Bulk Sync Phases



Adaptive Lookahead - Dynamic



DGETF2

DLSWP

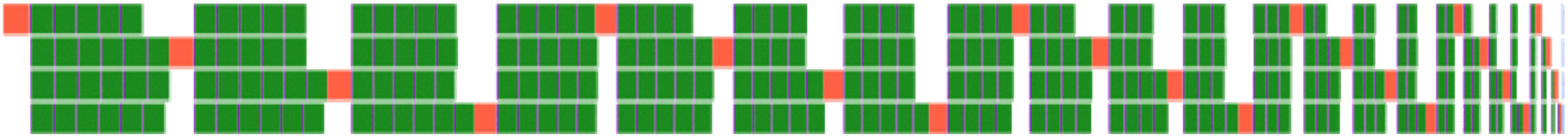
DLSWP

DTRSM

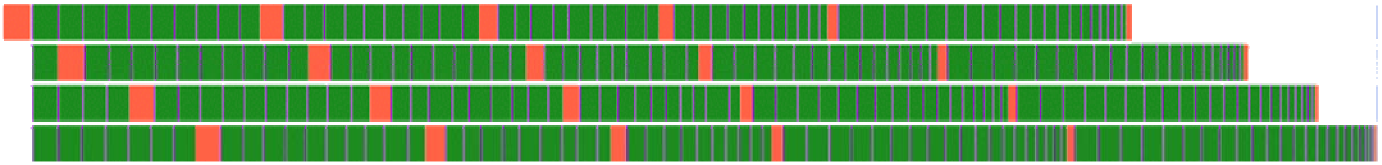
DGEMM

Event Driven Multithreading

LU – Fixed Lookahead – 4 processors



Original LAPACK Code



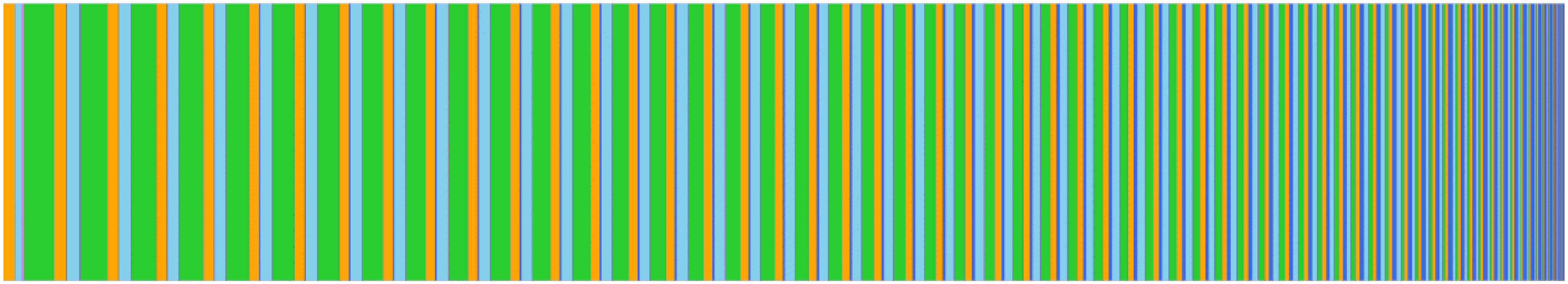
Data Flow Code

Time

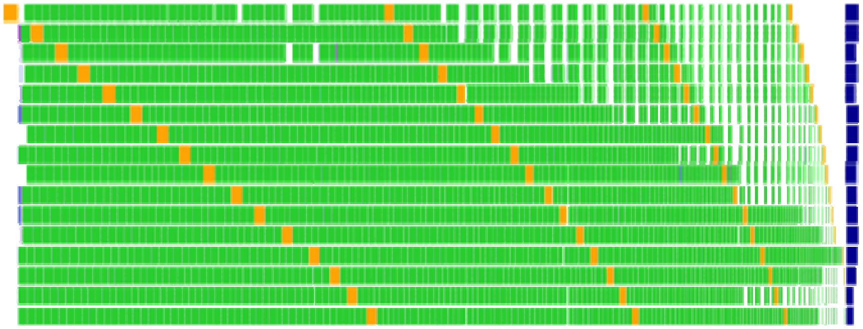
LU - BLAS Threads vs. Dynamic Lookahead

SGI Origin 3000 / 16 MIPS R14000 500 Mhz

BLAS Threads (LAPACK)



Dynamic Lookahead

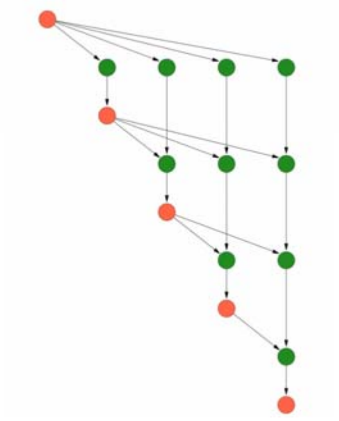


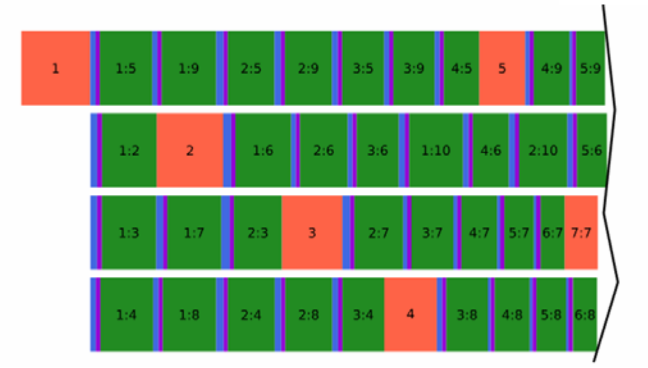
Problem Size N = 4000

Time

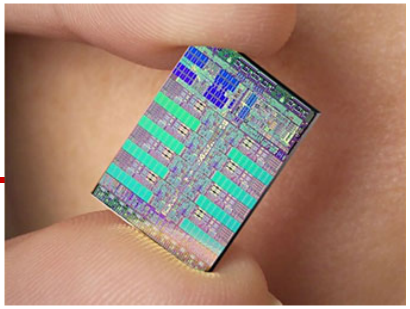
Event Driven Multithreading

```
while(1)
    fetch_task();
    switch(task.type) {
        case PANEL:                  // reduce the panel
            dgetf2();
            update_progress();
        case COLUMN:                 // update a block-column
            dlaswp();                // of the trailing submatrix
            dtrsm();
            dgemm();
            update_progress();
        case END:                    // perform left swap and return
            for()
                dlaswp();
            return;
    }
}
```

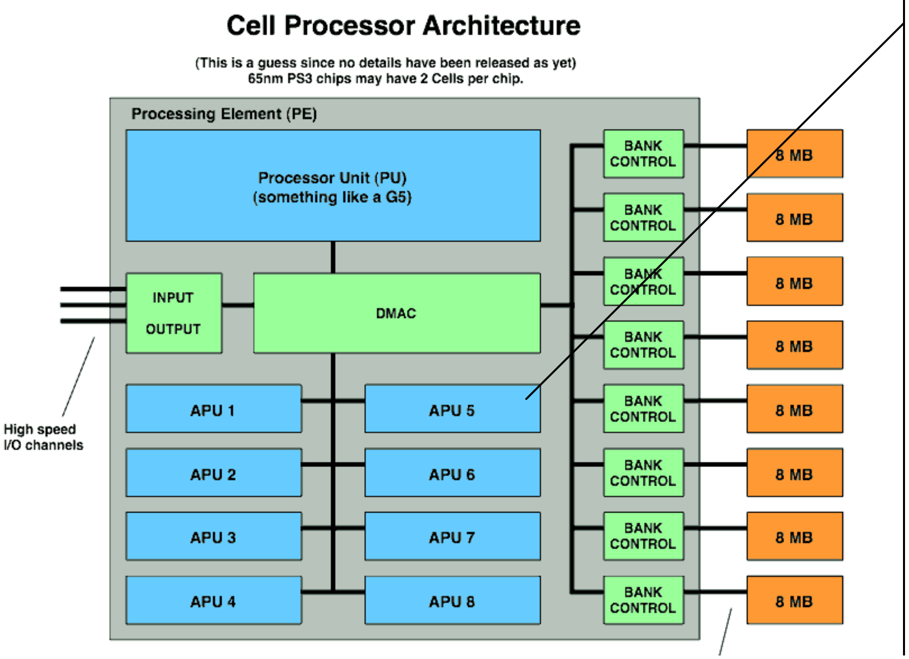




And Along Came the PlayStation 3

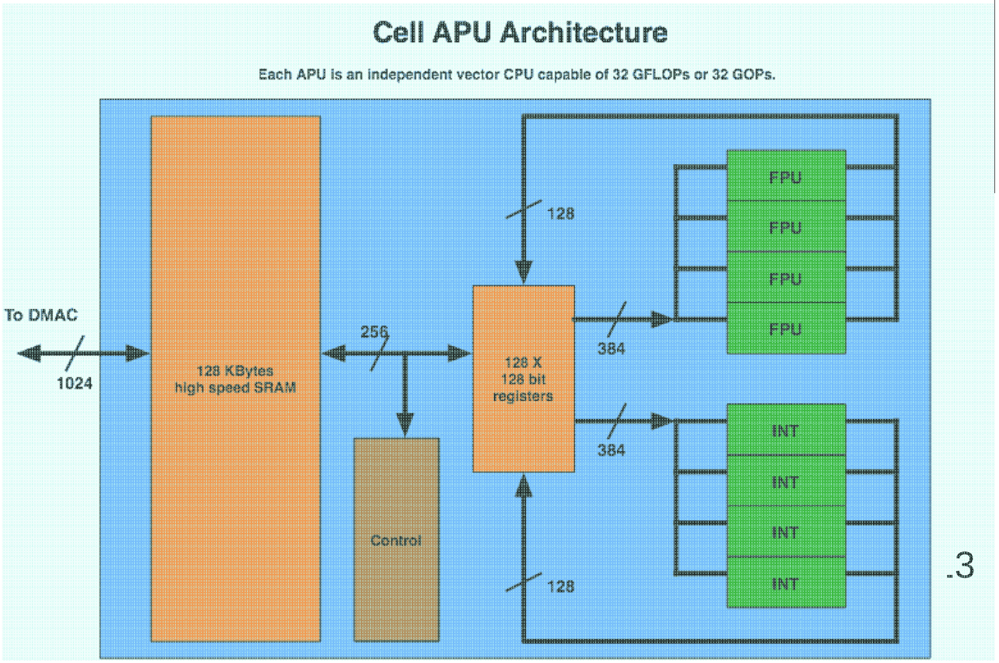


- The PlayStation 3's CPU based on a chip codenamed "Cell"
- Each Cell contains 8 APUs.
  - An APU is a self contained vector processor which acts independently from the others.
  - 4 floating point units capable of a total of 25 Gflop/s (5 Gflop/s each @ 3.2 GHz)
  - 204 Gflop/s peak! 32 bit floating point; 64 bit floating point at 15 Gflop/s.
  - IEEE format, but only rounds toward zero in 32 bit, overflow set to largest
    - According to IBM, the SPE's double precision unit is fully IEEE854 compliant.



Cell Processor Architecture

(This is a guess since no details have been released as yet)
65nm PS3 chips may have 2 Cells per chip.



Cell APU Architecture

Each APU is an independent vector CPU capable of 32 GFLOPs or 32 GOPs.

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 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.

32 and 64 Bit Floating Point Arithmetic

- **Iterative refinement for dense systems can work this way.**

Solve $Ax = b$ in lower precision,
save the factorization ($L*U = A*P$); $O(n^3)$

Compute in higher precision $r = b - A*x$; $O(n^2)$

Requires the original data A (stored in high precision)

Solve $Az = r$; using the lower precision factorization; $O(n^2)$

Update solution $x_+ = x + z$ using high precision; $O(n)$

Iterate until converged.

- 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; $O(10^8)$

In Matlab on My Laptop!

- Matlab has the ability to perform 32 bit floating point for some computations
  - Matlab uses LAPACK and MKL BLAS underneath.

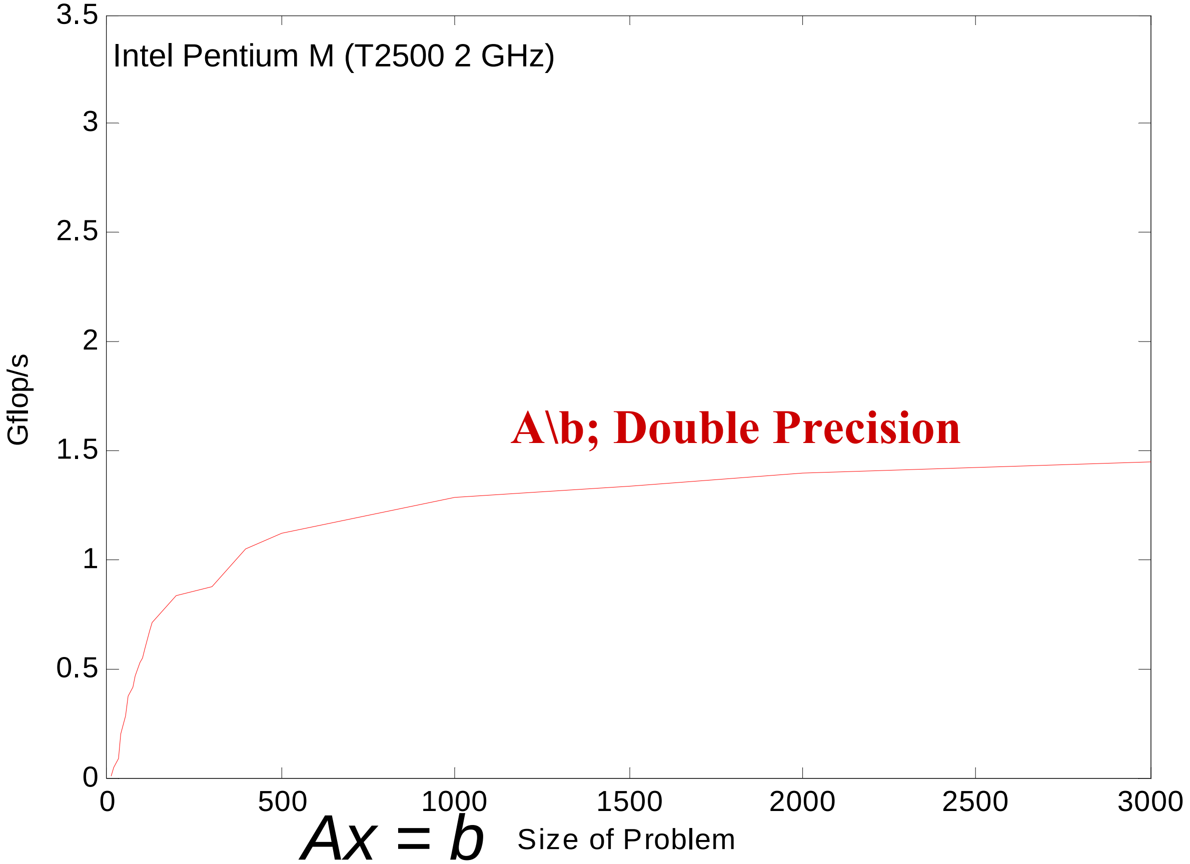
```
sa=single(a); sb=single(b);
[sl,su,sp]=lu(sa);                                  Most of the work: O(n^3)
sx=su\(sl\(sp*sb)); x=double(sx); r=b-a*x;                            O(n^2)
i=0;
while(norm(r)>res1),
   i=i+1;
   sr = single(r);
   sx1=su\(sl\(sp*sr)); x1=double(sx1); x=x1+x; r=b-a*x;              O(n^2)
if (i==30), break; end;
```

- Bulk of work, $O(n^3)$, in “single” precision
- Refinement, $O(n^2)$, in “double” precision
  - Computing the correction to the SP results in DP and adding it to the SP results in DP.

Another Look at Iterative Refinement

- **On a Pentium; using SSE2, single precision can perform 4 floating point operations per cycle and in double precision 2 floating point operations per cycle.**
- **In addition there is reduced memory traffic (factor on sp data)**

In Matlab Comparison of 32 bit w/iterative refinement and 64 Bit Computation for Ax=b

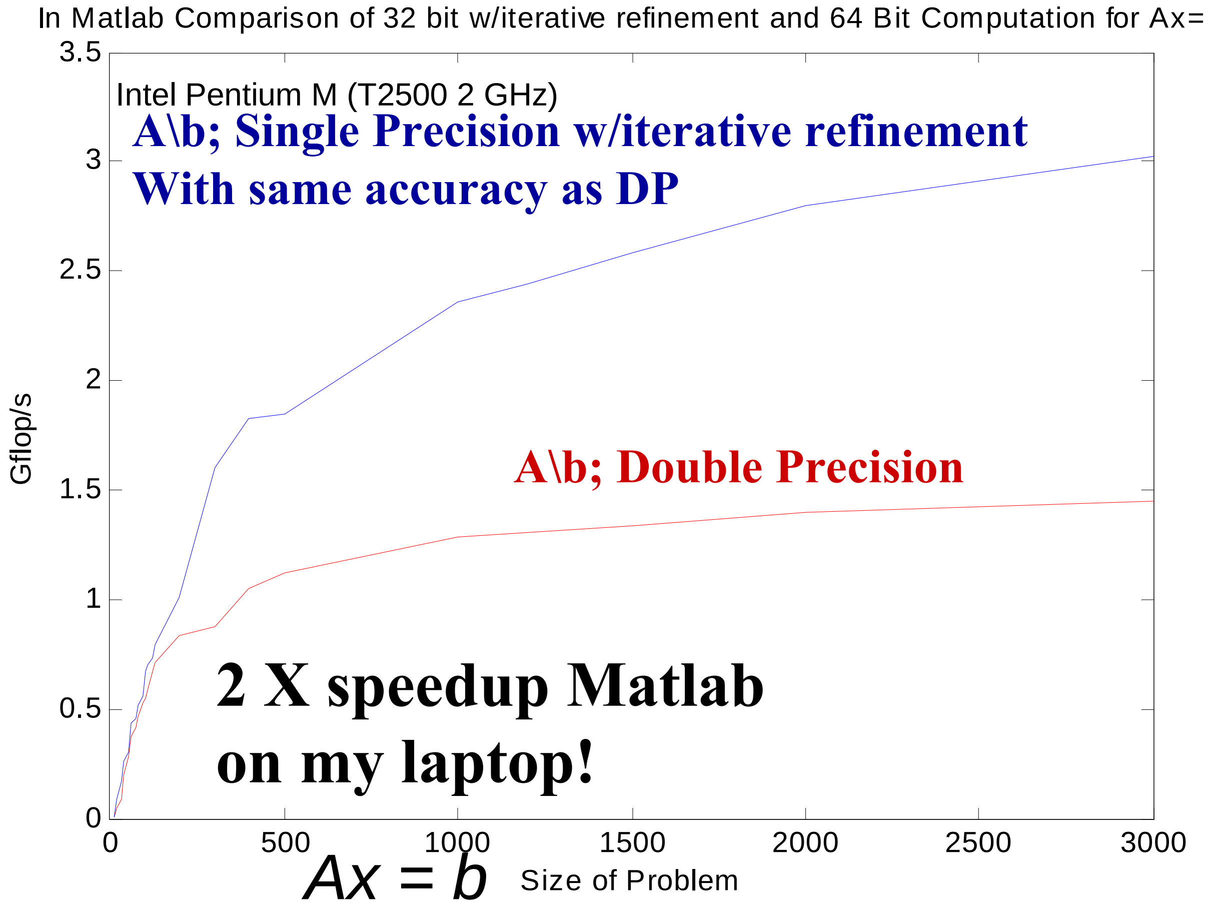


$Ax = b$

1.4 GFlop/s!

Another Look at Iterative Refinement

- On a Pentium; using SSE2, single precision can perform 4 floating point operations per cycle and in double precision 2 floating point operations per cycle.
- In addition there is reduced memory traffic (factor on sp data)



3 GFlop/s!!

On the Way to Understanding How to Use the Cell Something Else Happened …

- 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

Processor and BLAS Library	SGEMM (GFlop/s)	DGEMM (GFlop/s)	Speedup SP/DP
Pentium III Katmai (0.6GHz) Goto BLAS	0.98	0.46	2.13
Pentium III CopperMine (0.9GHz) Goto BLAS	1.59	0.79	2.01
Pentium Xeon Northwood (2.4GHz) Goto BLAS	7.68	3.88	1.98
Pentium Xeon Prescott (3.2GHz) Goto BLAS	10.54	5.15	2.05
Pentium IV Prescott (3.4GHz) Goto BLAS	11.09	5.61	1.98
AMD Opteron 240 (1.4GHz) Goto BLAS	4.89	2.48	1.97
PowerPC G5 (2.7GHz) AltiVec	18.28	9.98	1.83

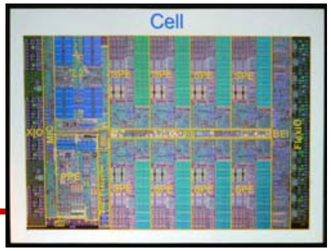
Performance of single precision and double precision matrix multiply (SGEMM and DGEMM) with n=m=k=1000

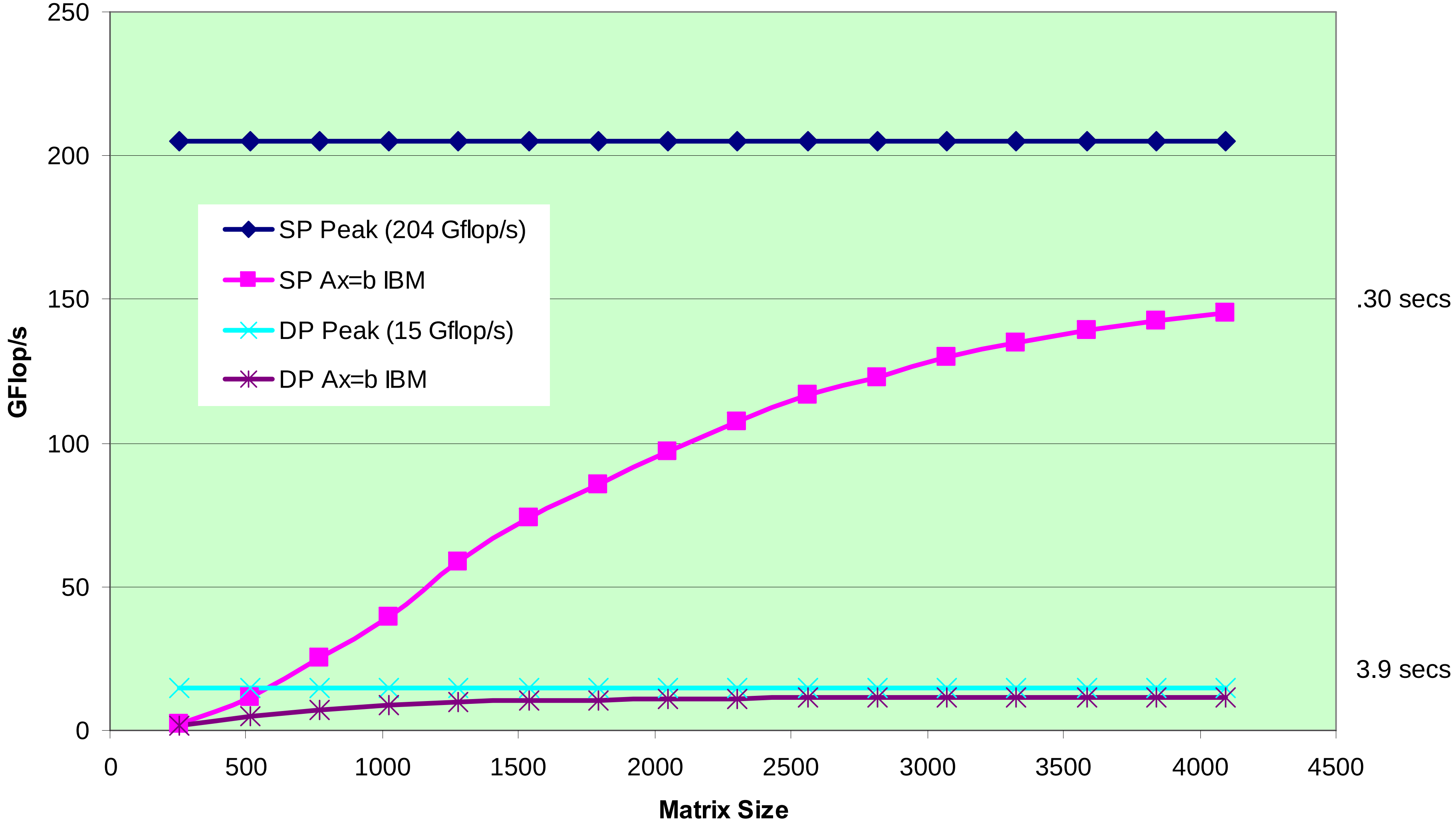
Speedups for Ax = b (Ratio of Times)

Architecture (BLAS)	*n*	DGEMM /SGEMM	DP Solve /SP Solve	DP Solve /Iter Ref	# iter
Intel Pentium III Coppermine (Goto)	3500	2.10	2.24	1.92	4
Intel Pentium IV Prescott (Goto)	4000	2.00	1.86	1.57	5
AMD Opteron (Goto)	4000	1.98	1.93	1.53	5
Sun UltraSPARC IIe (Sunperf)	3000	1.45	1.79	1.58	4
IBM Power PC G5 (2.7 GHz) (VecLib)	5000	2.29	2.05	1.24	5
Cray X1 (libsci)	4000	1.68	1.57	1.32	7
Compaq Alpha EV6 (CXML)	3000	0.99	1.08	1.01	4
IBM SP Power3 (ESSL)	3000	1.03	1.13	1.00	3
SGI Octane (ATLAS)	2000	1.08	1.13	0.91	4

Architecture (BLAS-MPI)	# procs	*n*	DP Solve /SP Solve	DP Solve /Iter Ref	# iter
AMD Opteron (Goto – OpenMPI MX)	32	22627	1.85	1.79	6
AMD Opteron (Goto – OpenMPI MX)	64	32000	1.90	1.83	6

IBM Cell 3.2 GHz, Ax = b

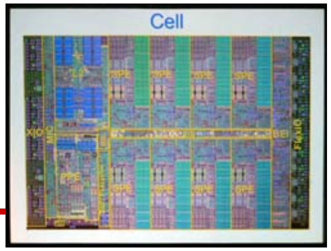


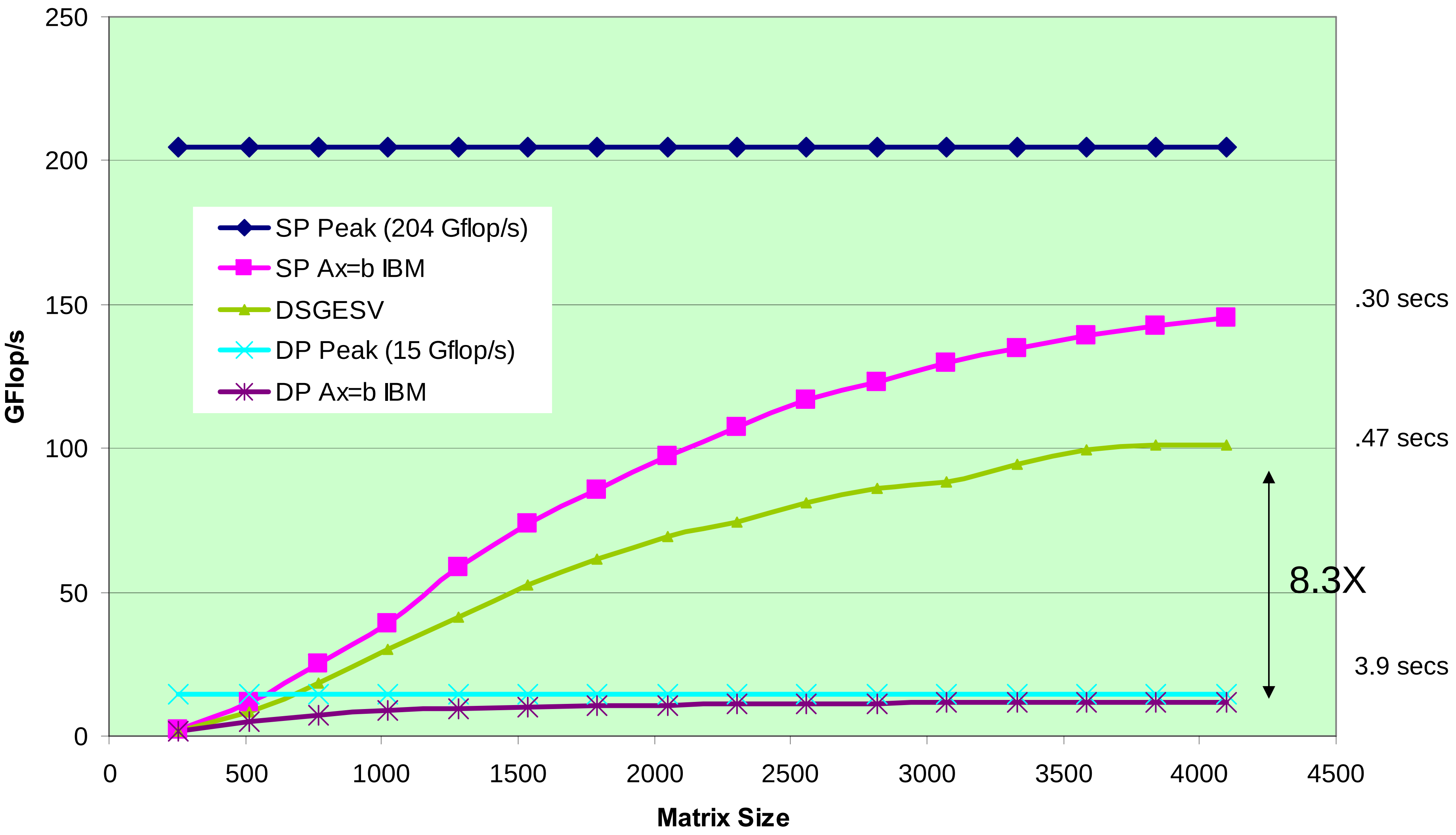


.30 secs

3.9 secs

IBM Cell 3.2 GHz, Ax = b





Quadruple Precision

n	Quad Precision Ax = b	Iter. Refine. DP to QP	
	time (s)	time (s)	Speedup
100	0.29	0.03	9.5
200	2.27	0.10	20.9
300	7.61	0.24	30.5
400	17.8	0.44	40.4
500	34.7	0.69	49.7
600	60.1	1.01	59.0
700	94.9	1.38	68.7
800	141.	1.83	77.3
900	201.	2.33	86.3
1000	276.	2.92	94.8

Intel Xeon 3.2 GHz

Reference implementation of the quad precision BLAS

Accuracy: 10^{-32}

No more than 3 steps of iterative refinement are needed.

- **Variable precision factorization (with say < 32 bit precision) plus 64 bit refinement produces 64 bit accuracy**

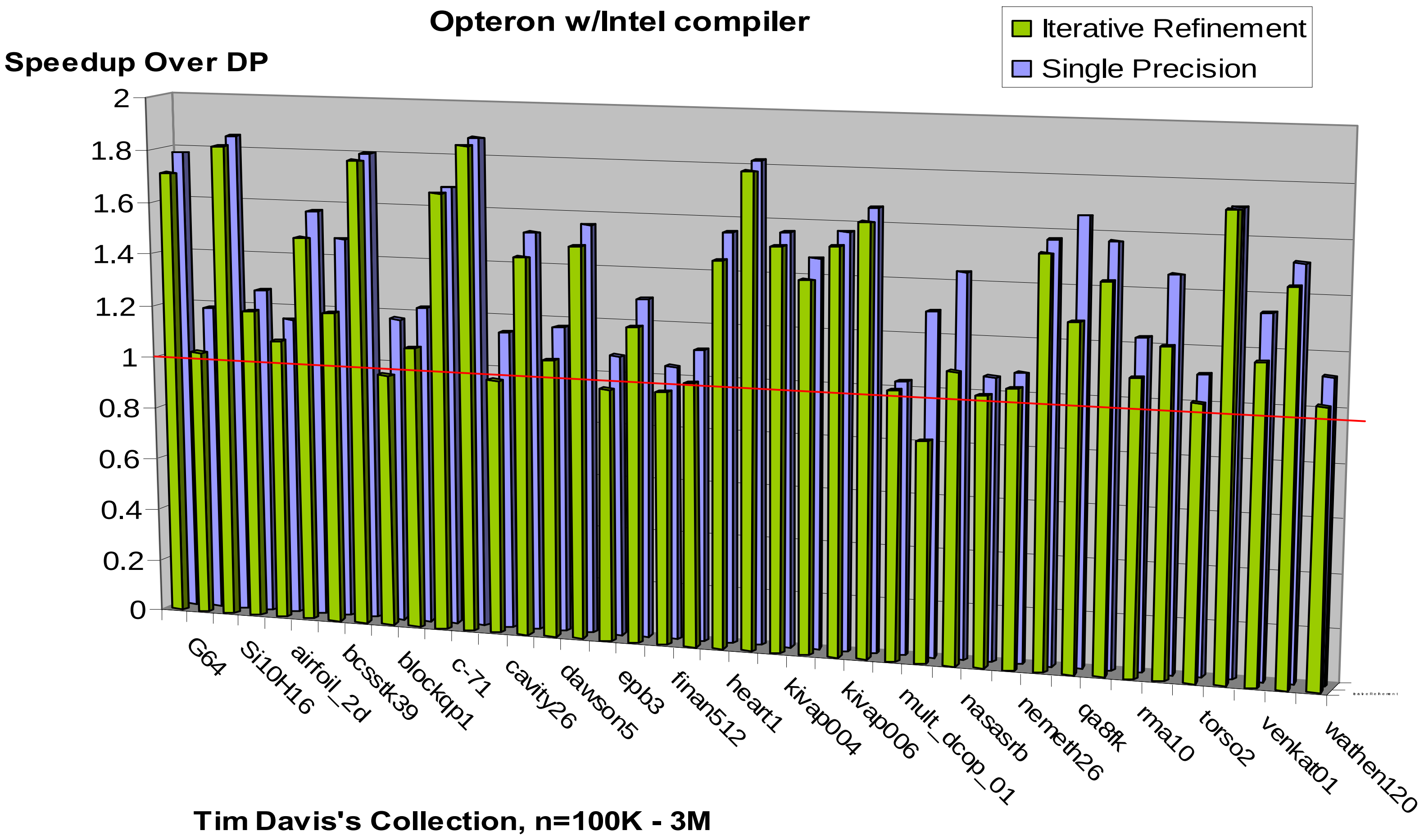
Refinement Technique Using Single/Double Precision

- Linear Systems
  - LU (dense and sparse)
  - Cholesky
  - QR Factorization
- Eigenvalue
  - Symmetric eigenvalue problem
  - SVD
  - Same idea as with dense systems,
    - Reduce to tridiagonal/bi-diagonal in lower precision, retain original data and improve with iterative technique using the lower precision to solve systems and use higher precision to calculate residual with original data.
    - $O(n^2)$ per value/vector
- Iterative Linear System
  - Relaxed GMRES
  - Inner/outer iteration scheme

See webpage for tech report which discusses this.

MUMPS and Iterative Refinement

Sparse direct solver based on multifrontal approach which generates dense matrix multiplies

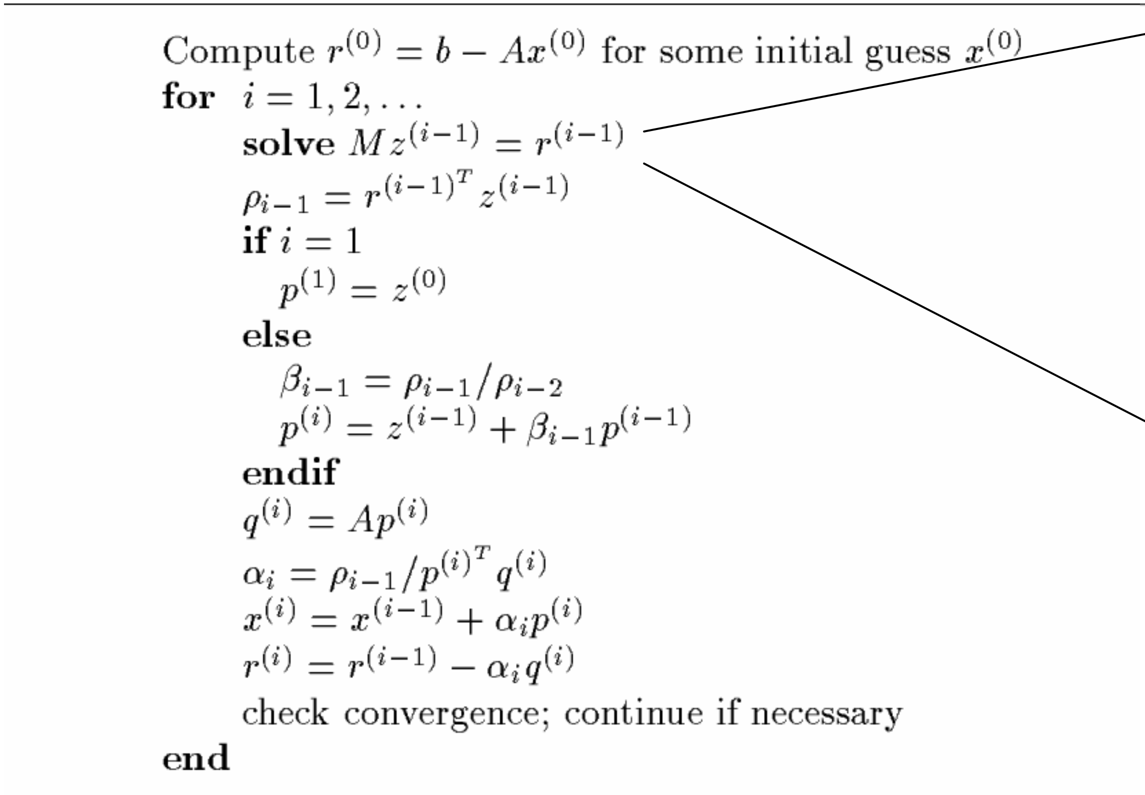


Sparse Iterative Methods (PCG)

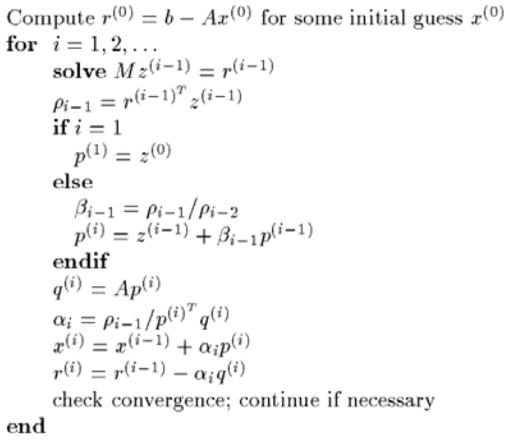
- **Outer/Inner Iteration**

Outer iterations using 64 bit floating point

Inner iteration:
In 32 bit floating point



```
Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for $i = 1, 2, \ldots$
    solve $Mz^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
    if $i = 1$
        $p^{(1)} = z^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    endif
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1} / p^{(i)^T} q^{(i)}$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
    check convergence; continue if necessary
end
```

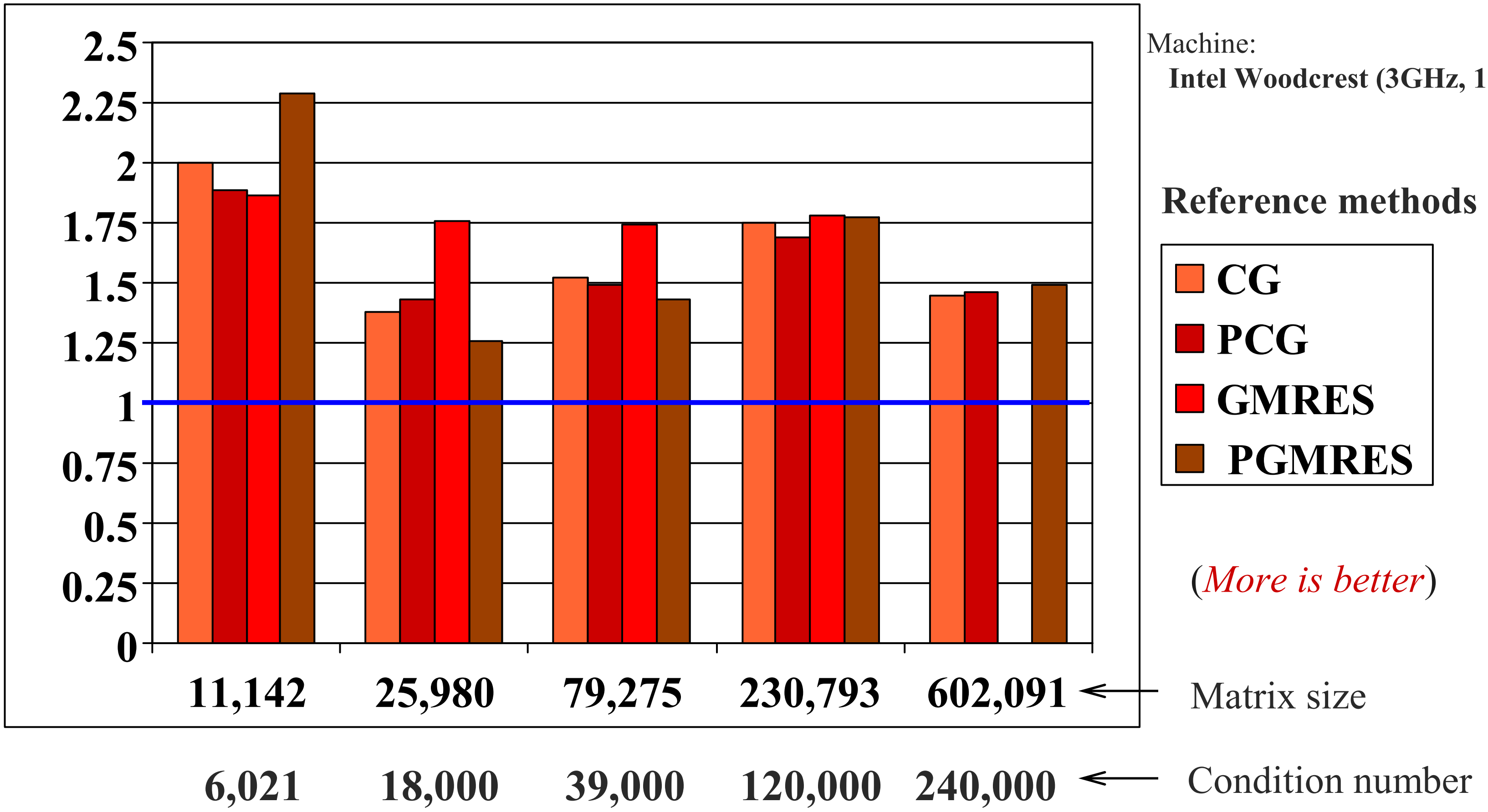


```
Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for $i = 1, 2, \ldots$
    solve $Mz^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
    if $i = 1$
        $p^{(1)} = z^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    endif
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1} / p^{(i)^T} q^{(i)}$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
    check convergence; continue if necessary
end
```

- **Outer iteration in 64 bit floating point and inner iteration in 32 bit floating point**

Mixed Precision Computations for Sparse Inner/Outer-type Iterative Solvers

Time speedups for mixed precision Inner SP/Outer DP (SP/DP) iter. methods *vs* DP/DP
(CG, GMRES, PCG, and PGMRES with diagonal preconditioners)



Machine:
Intel Woodcrest (3GHz, 1333MHz bus)

(*More is better*)

Future Large Systems, Say in 5 Years

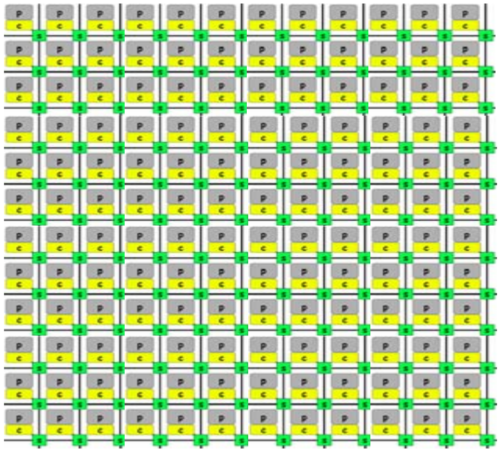
- 128 cores per socket

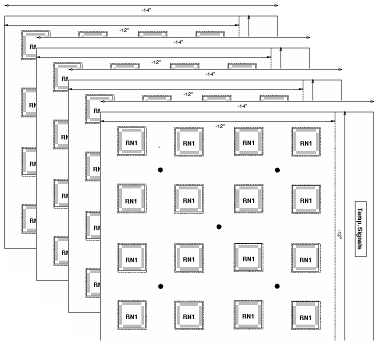
- 32 sockets per node

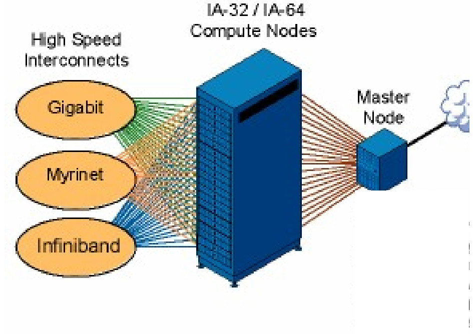
- 128 nodes per system

- System = 128*32*128
  = 524,288 Cores!

- And by the way, its 4 threads of exec per core
- That's about 2M threads to manage







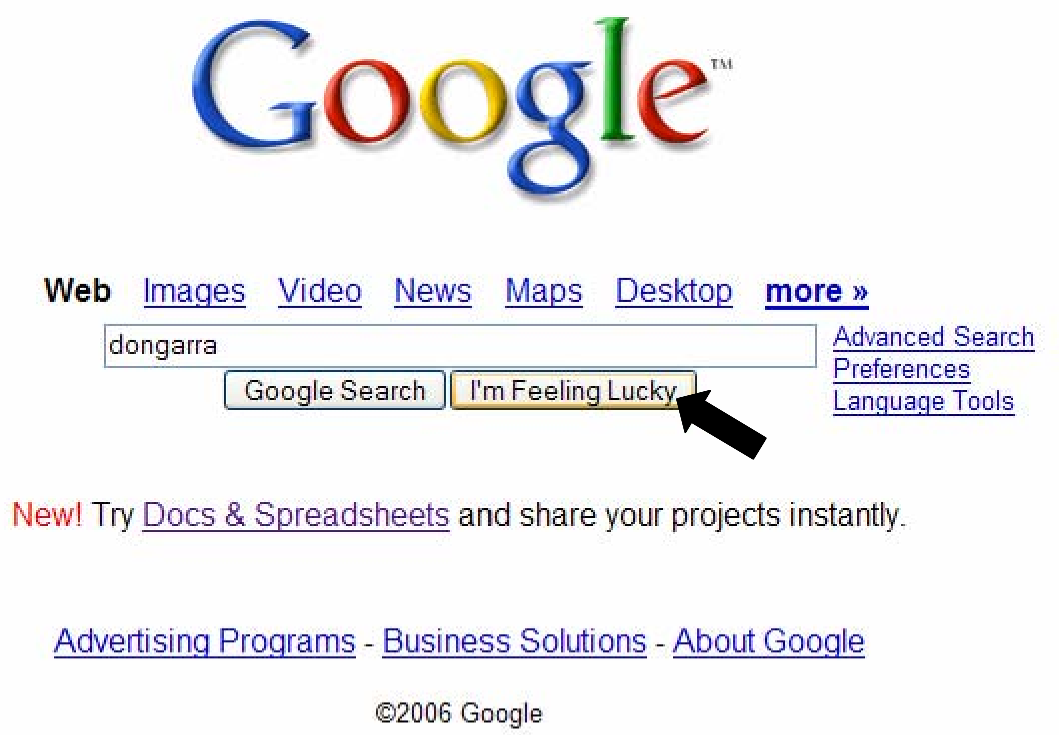
Real Crisis With HPC Is With The Software

- Programming is stuck
  - Arguably hasn't changed since the 60's
- It's time for a change
  - Complexity is rising dramatically
    - highly parallel and distributed systems
      - From 10 to 100 to 1000 to 10000 to 100000 of processors!!
    - multidisciplinary applications
- A supercomputer application and software are usually much more long-lived than a hardware
  - Hardware life typically five years at most.
  - Fortran and C are the main programming models
- Software is a major cost component of modern technologies.
  - The tradition in HPC system procurement is to assume that the software is free.

Collaborators / Support

- U Tennessee, Knoxville
  - Alfredo Buttari, Julien Langou, Julie Langou, Piotr Luszcze Jakub Kurzak Stan Tomov

Office of Science
U.S. DEPARTMENT OF ENERGY



dongarra

New! Try Docs & Spreadsheets and share your projects instantly.

Advertising Programs - Business Solutions - About Google

©2006 Google