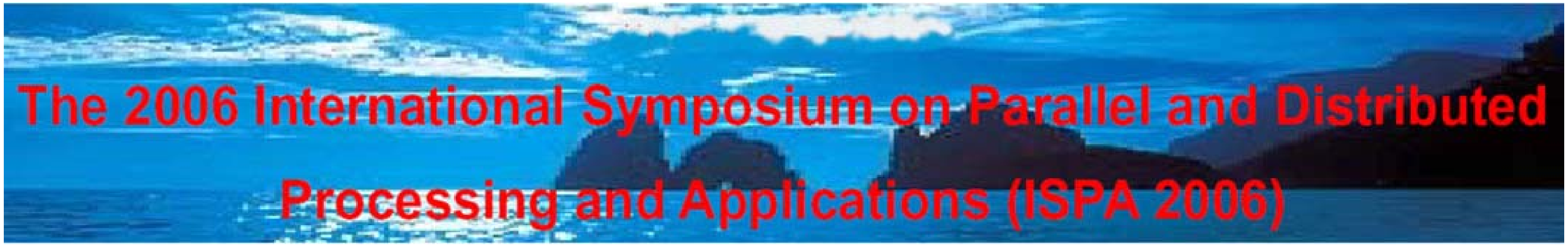


The 2006 International Symposium on Parallel and Distributed Processing and Applications (ISPA 2006)

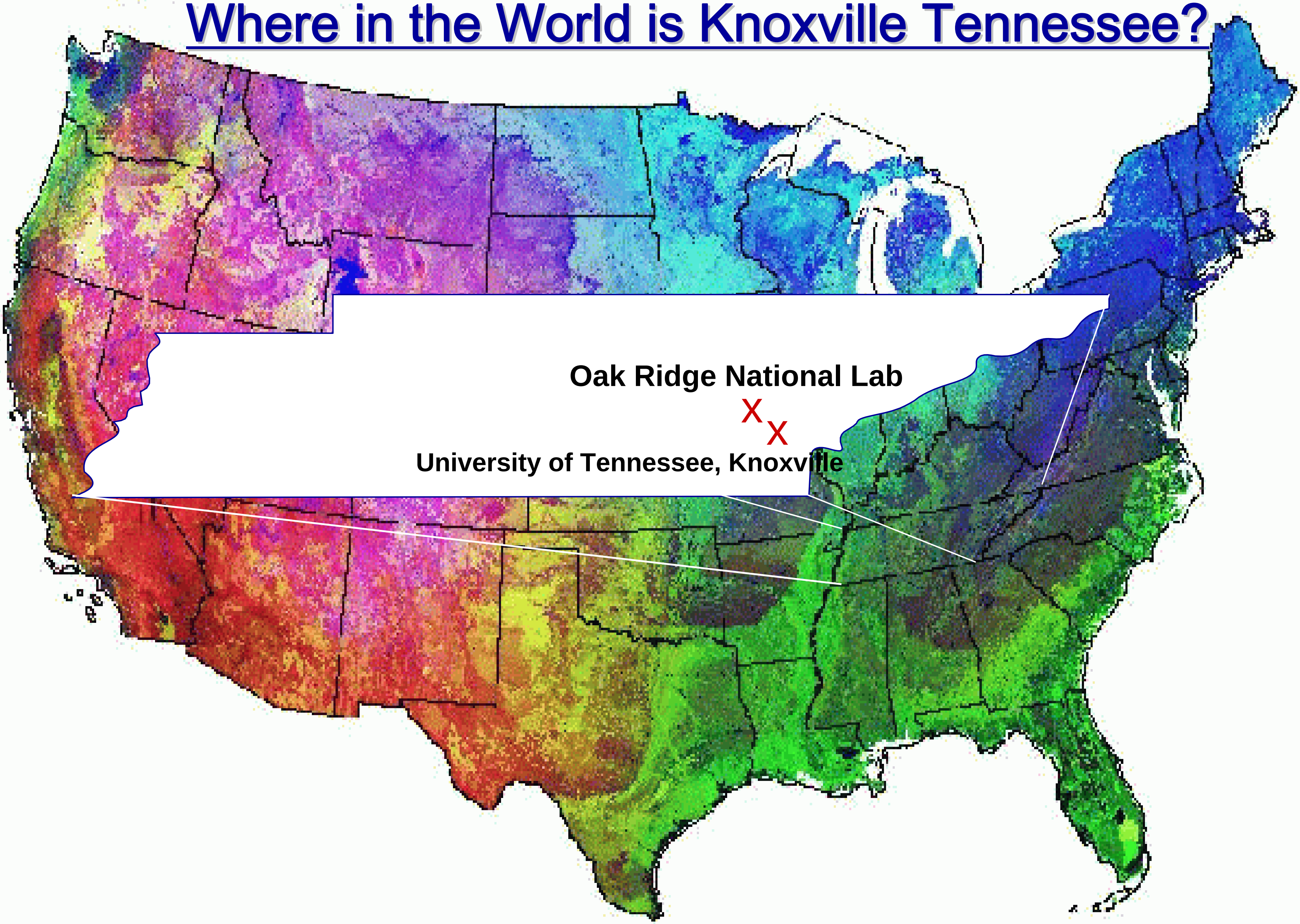
The Impact of Multicore on Math Software and Exploiting Single Precision Computing to Obtain Double Precision Results

Jack Dongarra

Innovative Computer Laboratory

University of Tennessee
Oak Ridge National Laboratory

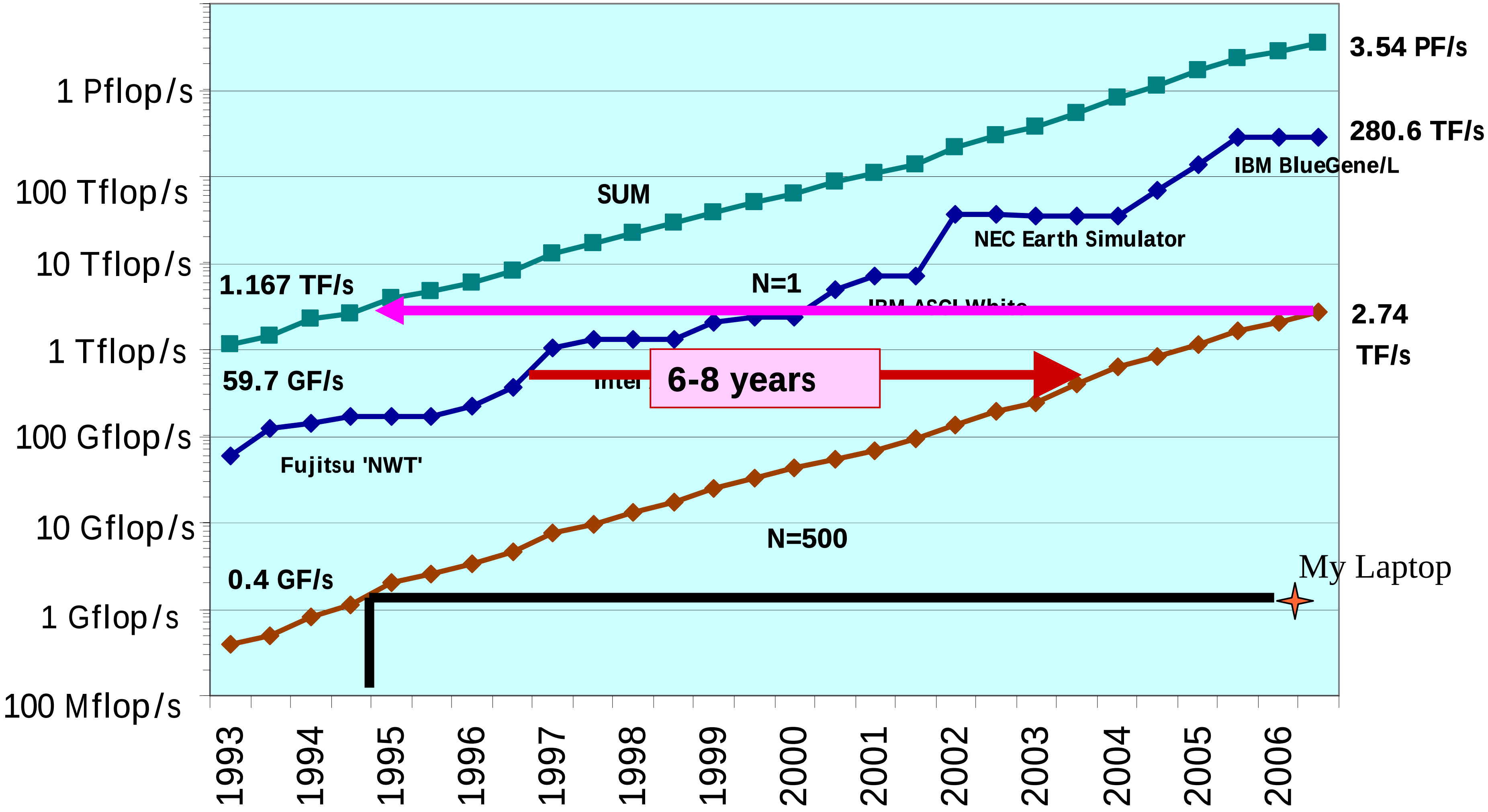
Where in the World is Knoxville Tennessee?



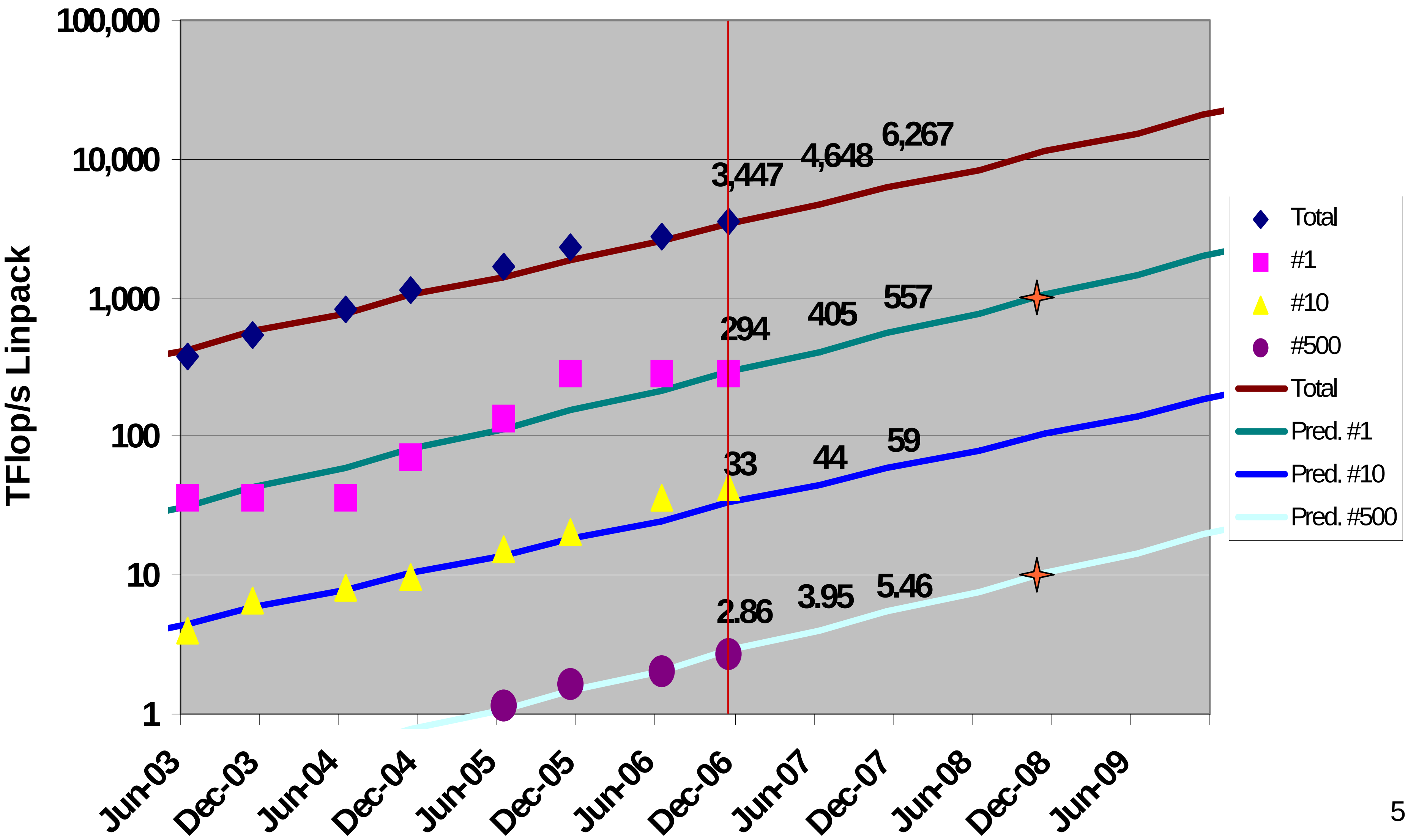
Outline

- Top500
  - A quick look
- Multicore
  - Software changes
- IBM Cell processor
  - Early experiments

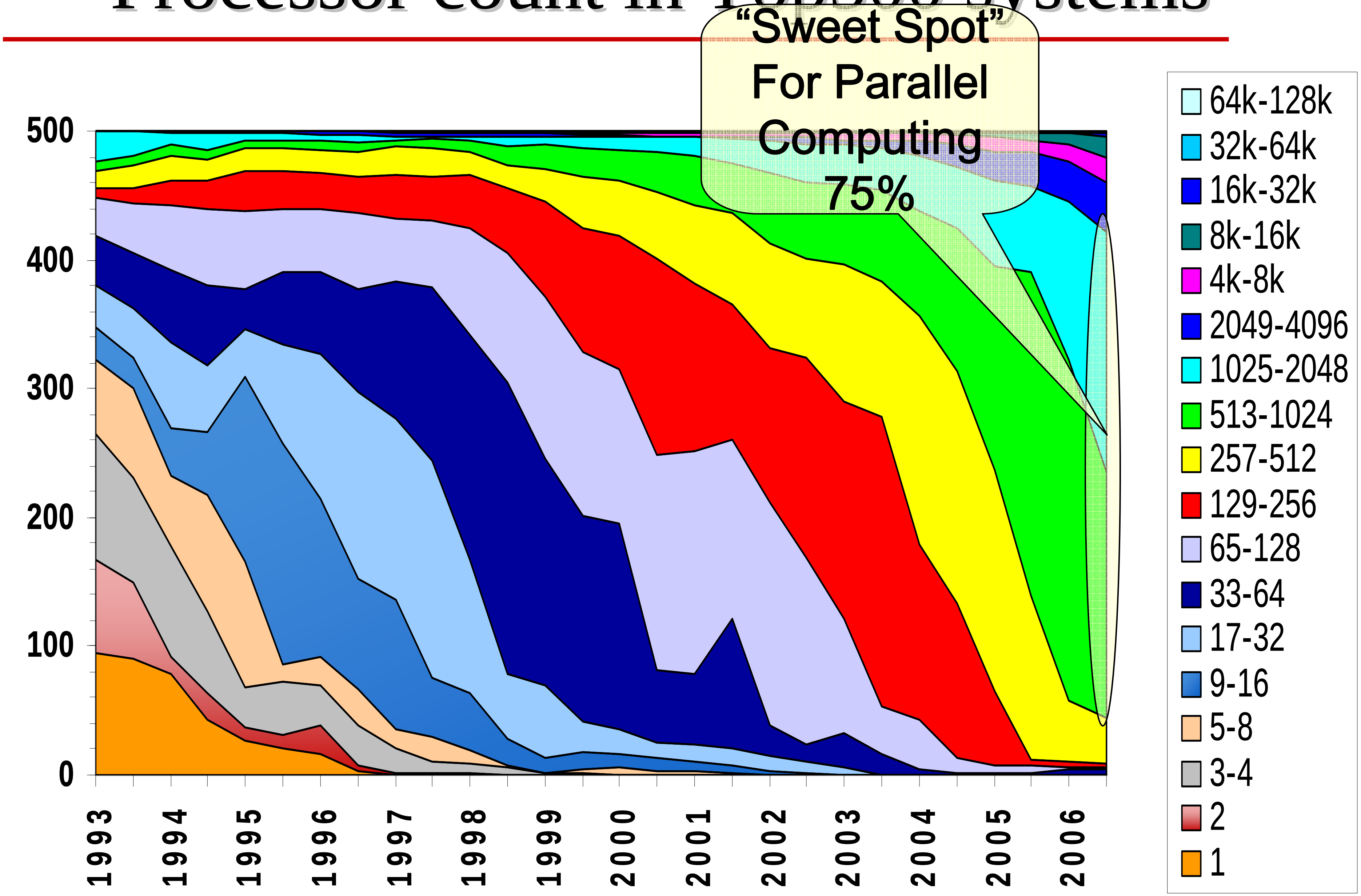
Performance Development; Top500



Predicted Performance Levels



Processor count in Top500 systems

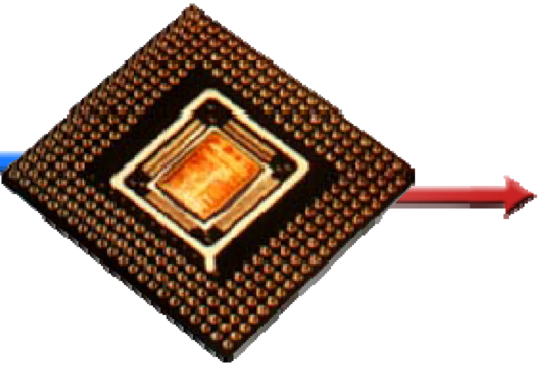


“Sweet Spot” For Parallel Computing 75%

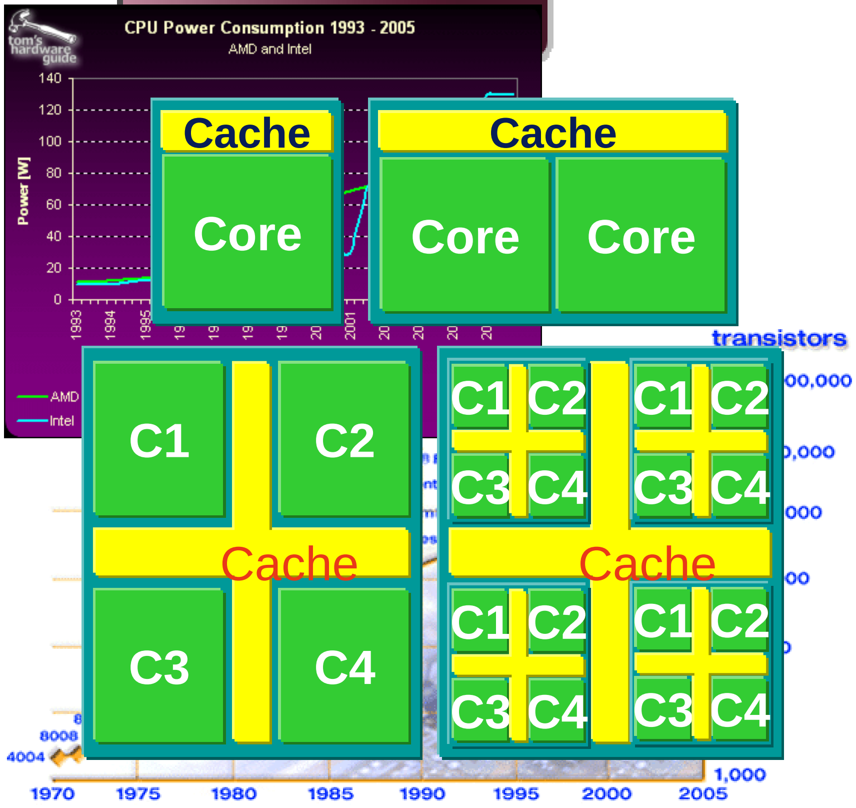
Increasing CPU Performance: A Delicate Balancing Act

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.

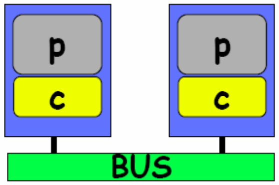
However, the number of gates on a chip will continue to increase.

What is Multicore?

- Multiple, externally visible processors on a single die
- Processors have independent control-flow, separate internal state and no critical resource sharing.

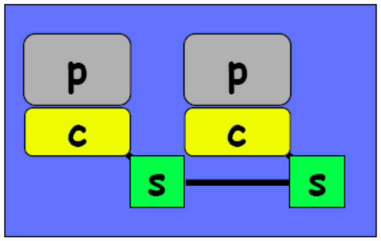
- Its not just SMP on a chip
  - Cores on the wrong side of the pins
- Highly sensitive to temporal locality
- Memory wall is getting worse

- Discrete chips

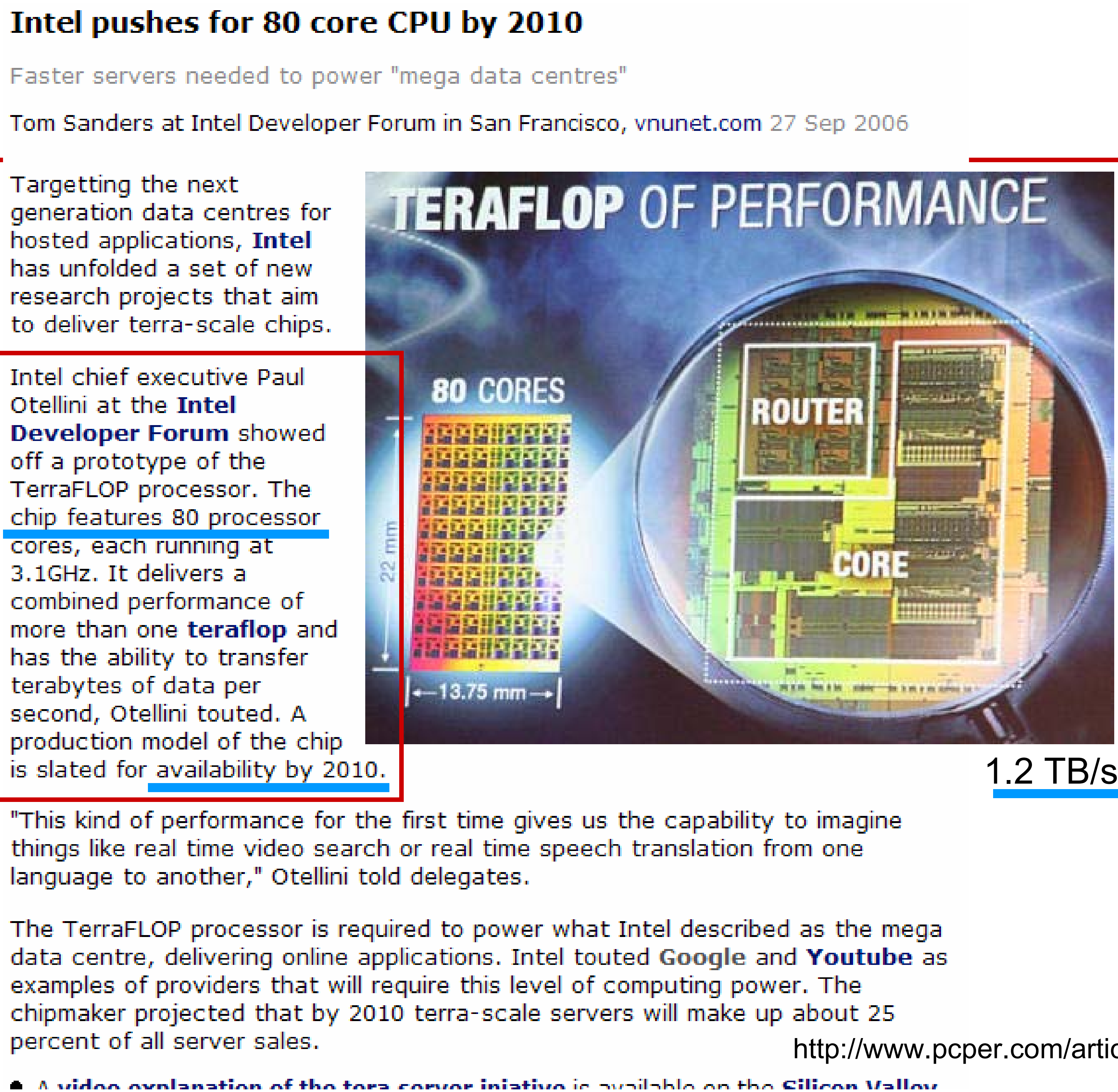


  - Bandwidth 2 GBps
  - Latency 60 ns

- Multicore



  - Bandwidth > 20 GBps
  - Latency < 3ns



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.

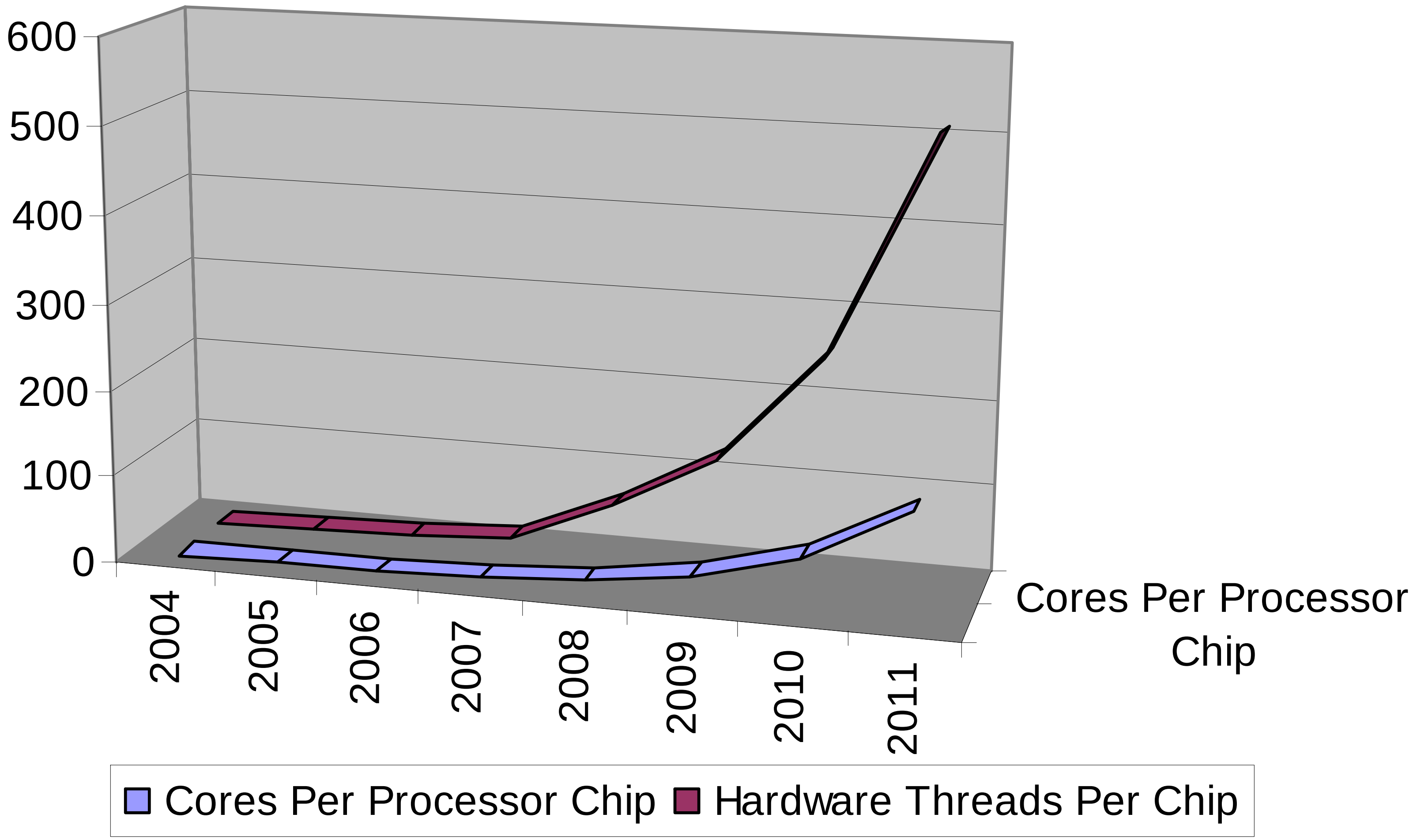
- A **video explanation of the tera server iniative** is available on the **Silicon Valley**

1.2 TB/s memory BW

http://www.pcper.com/article.php?aid=302

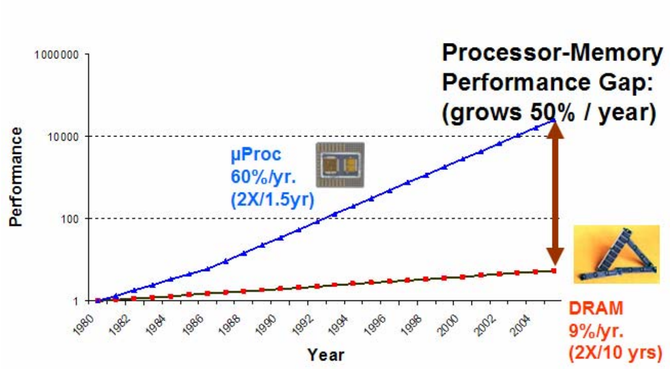
CPU Desktop Trends 2004-2011

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



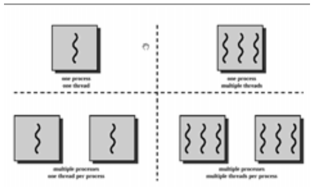
Challenges Resulting From Multicore

- **Aggravated memory wall**
  - Memory bandwidth
    - to get data out of memory banks
    - to get data into multi-core processors
  - Memory latency
  - Fragments L3 cache



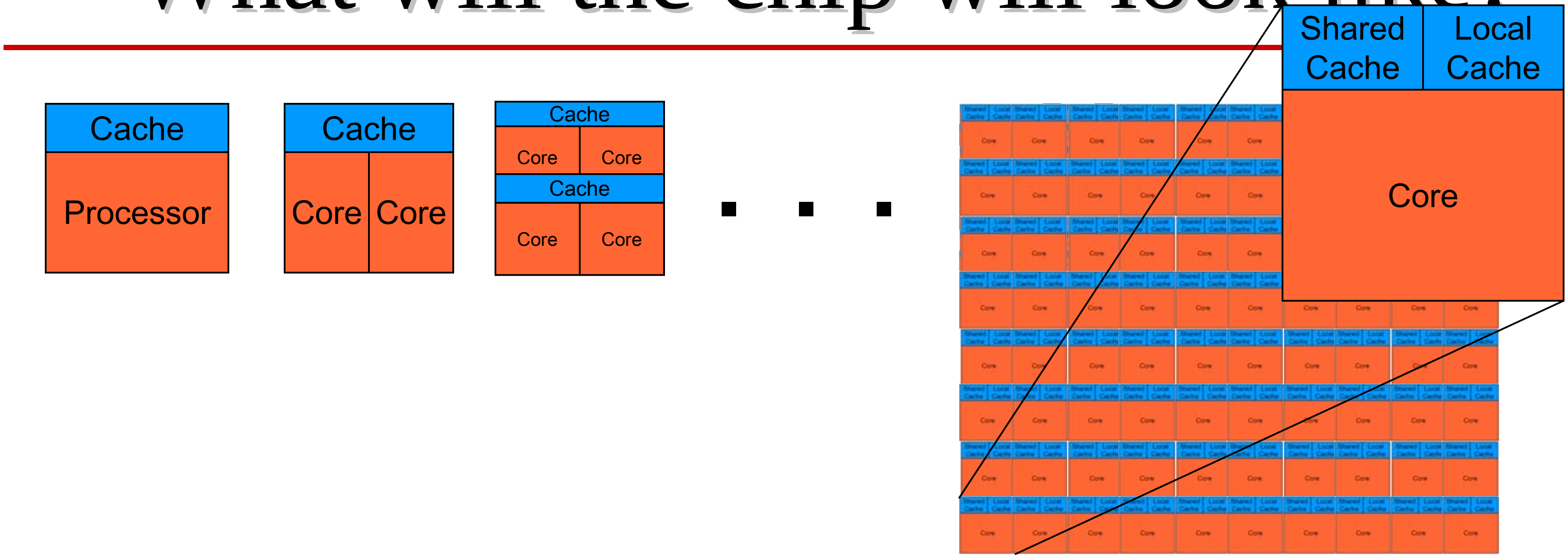
- **Pins become strangle point**
  - Rate of pin growth projected to slow and flatten
  - Rate of bandwidth per pin projected to grow slowly

- **Relies on effective exploitation of multiple-thread parallelism**
  - Need for parallel computing model and parallel programming model



- **Requires mechanisms for efficient inter-processor coordination**
  - Synchronization
  - Mutual exclusion
  - Context switching

What will the chip will look like?



Cache

Processor

Cache

Core Core

Cache

Core Core

Cache

Core Core

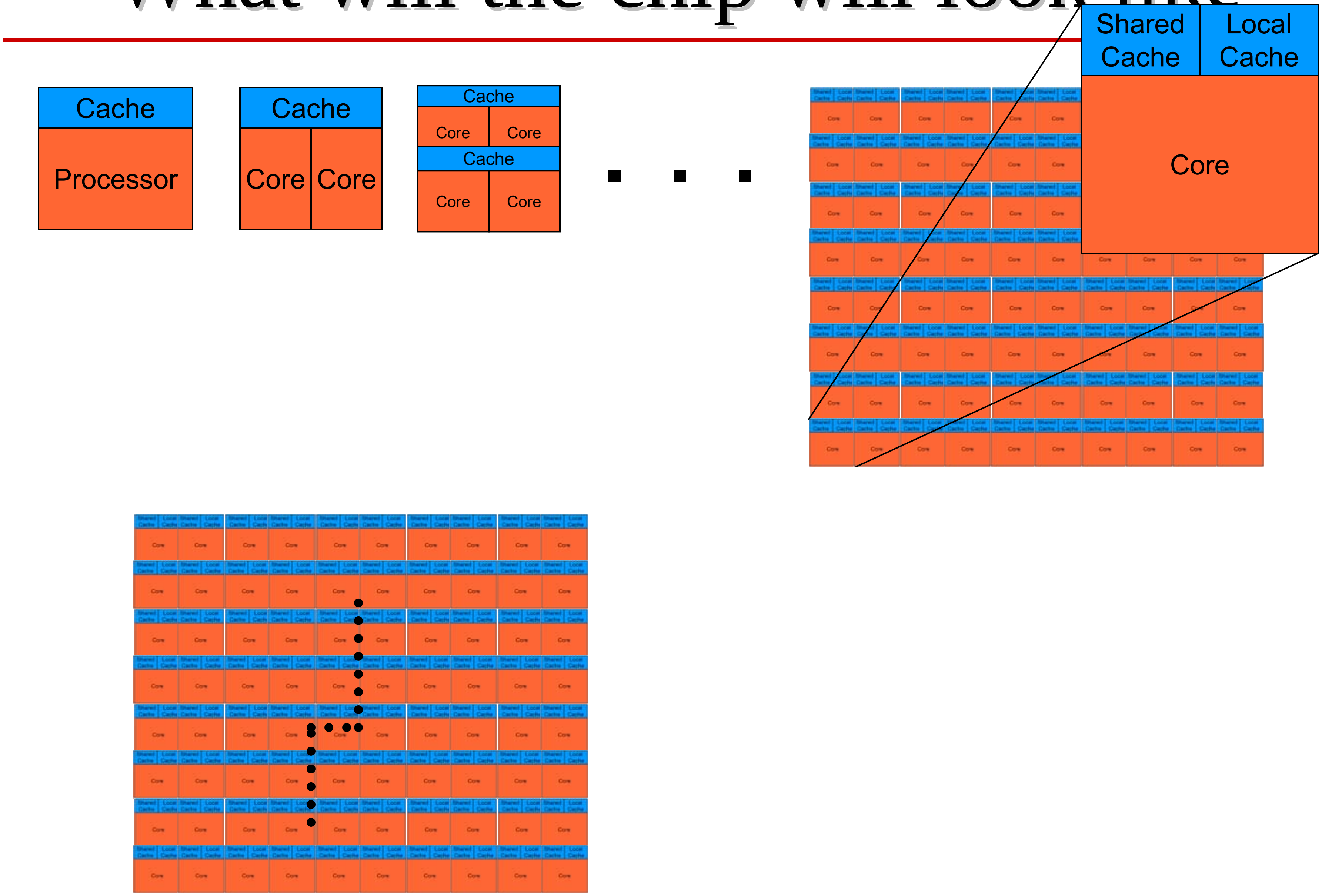
. . .

Shared Cache

Local Cache

Core

What will the chip will look like



Cache

Processor

Cache

Core Core

Cache

Core Core

Cache

Core Core

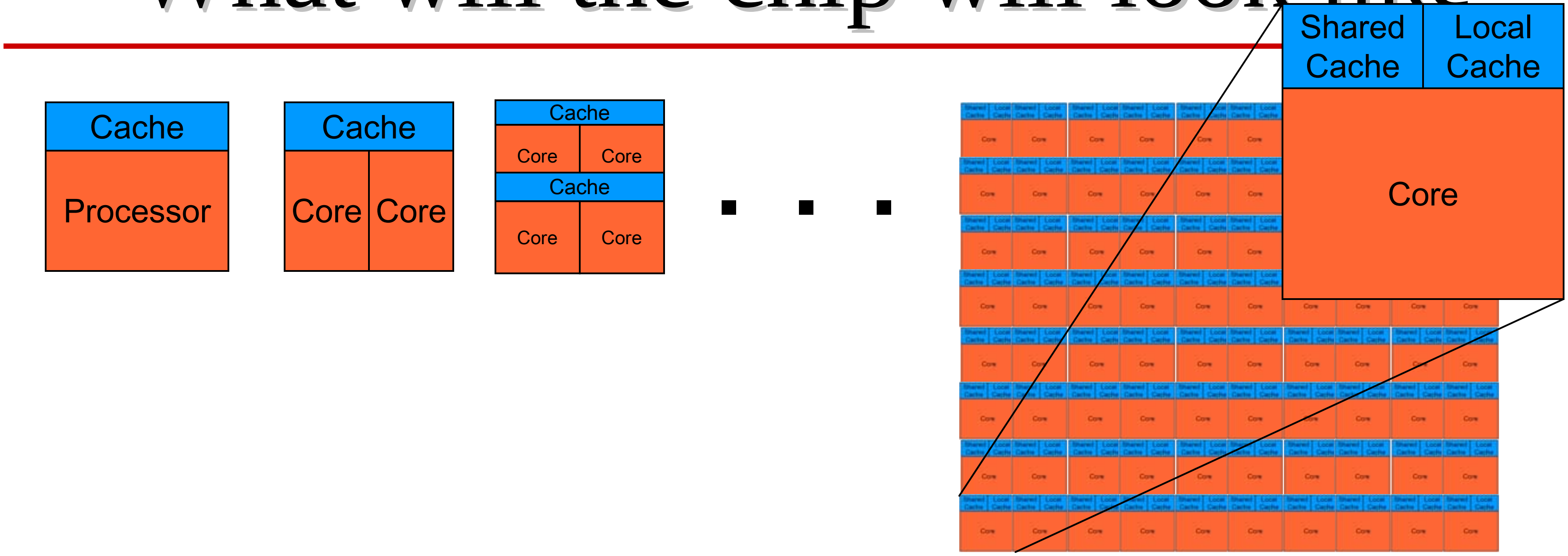
. . .

Shared Cache

Local Cache

Core

What will the chip will look like



Cache

Processor

Cache

Core Core

Cache

Core Core

Cache

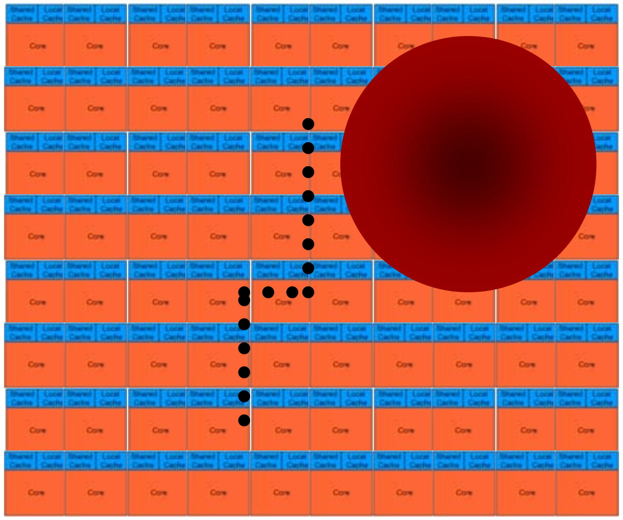
Core Core

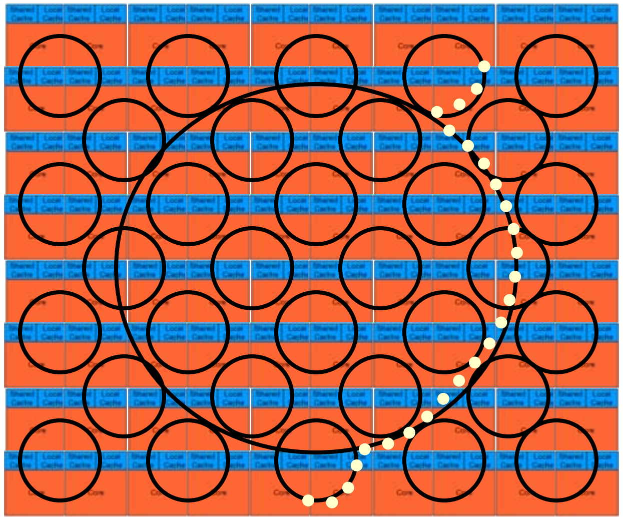
. . .

Shared Cache

Local Cache

Core

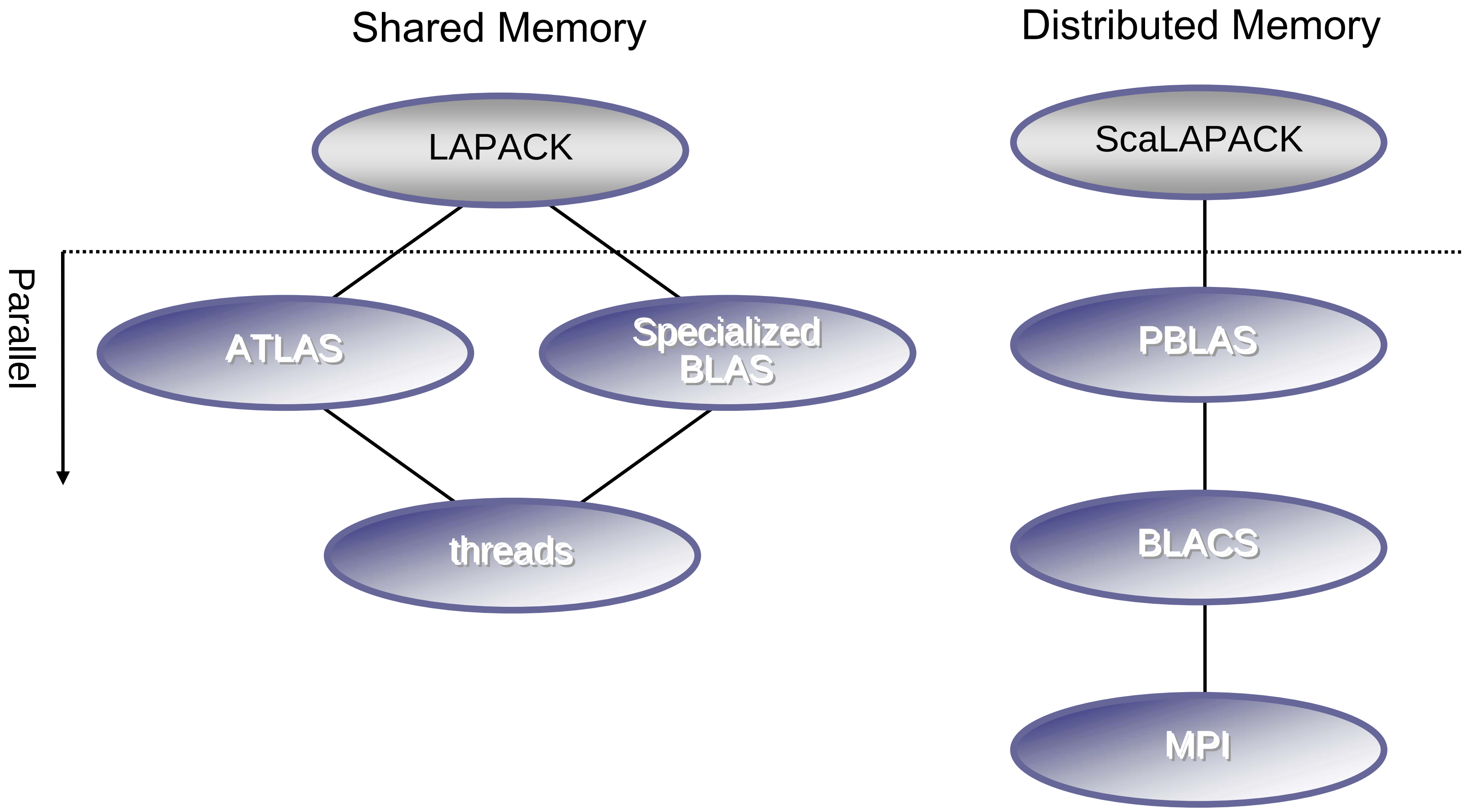




Major Changes to Software

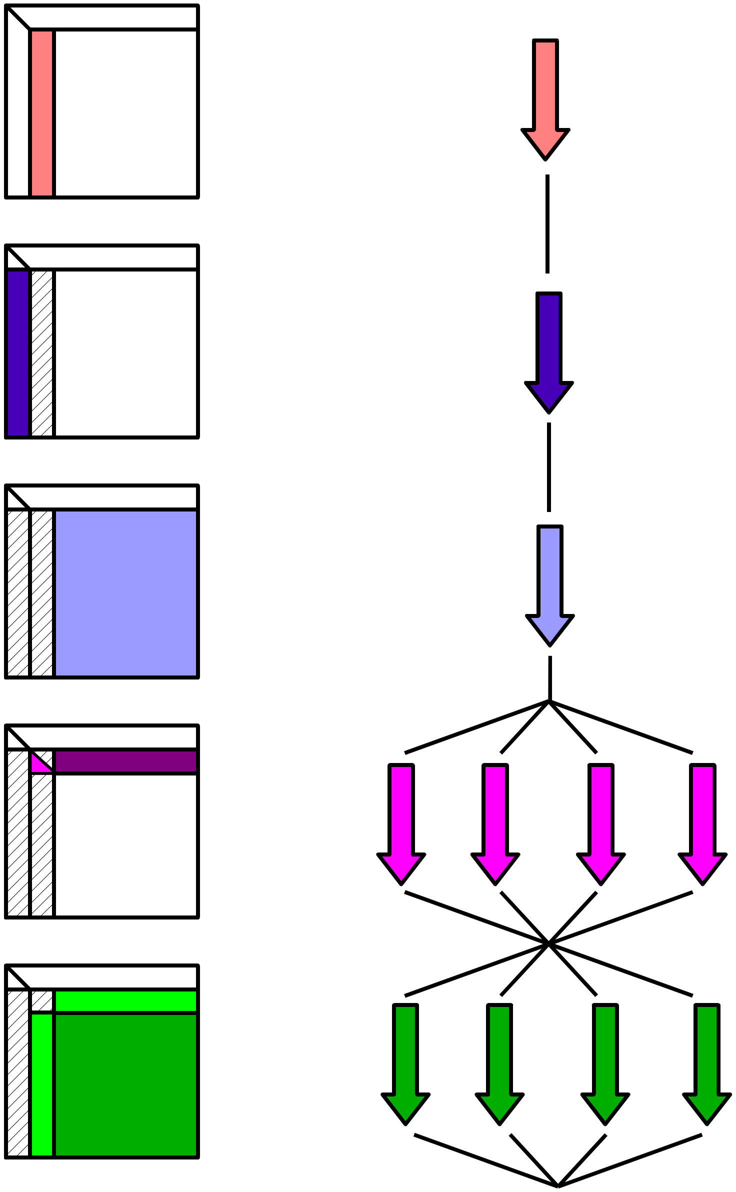
- Must rethink the design of our software
    - Another disruptive technology
        - Similar to what happened with cluster computing and message passing
    - Rethink and rewrite the applications, algorithms, and software
- Numerical libraries for example will change
    - For example, both LAPACK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

Parallelism in LAPACK / ScaLAPACK



Two well known open source software efforts for dense matrix problems.

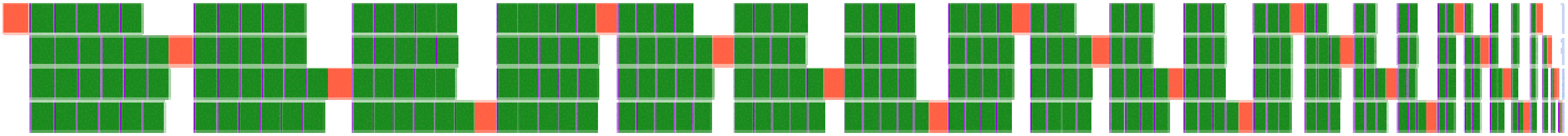
Steps in the LAPACK LU



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

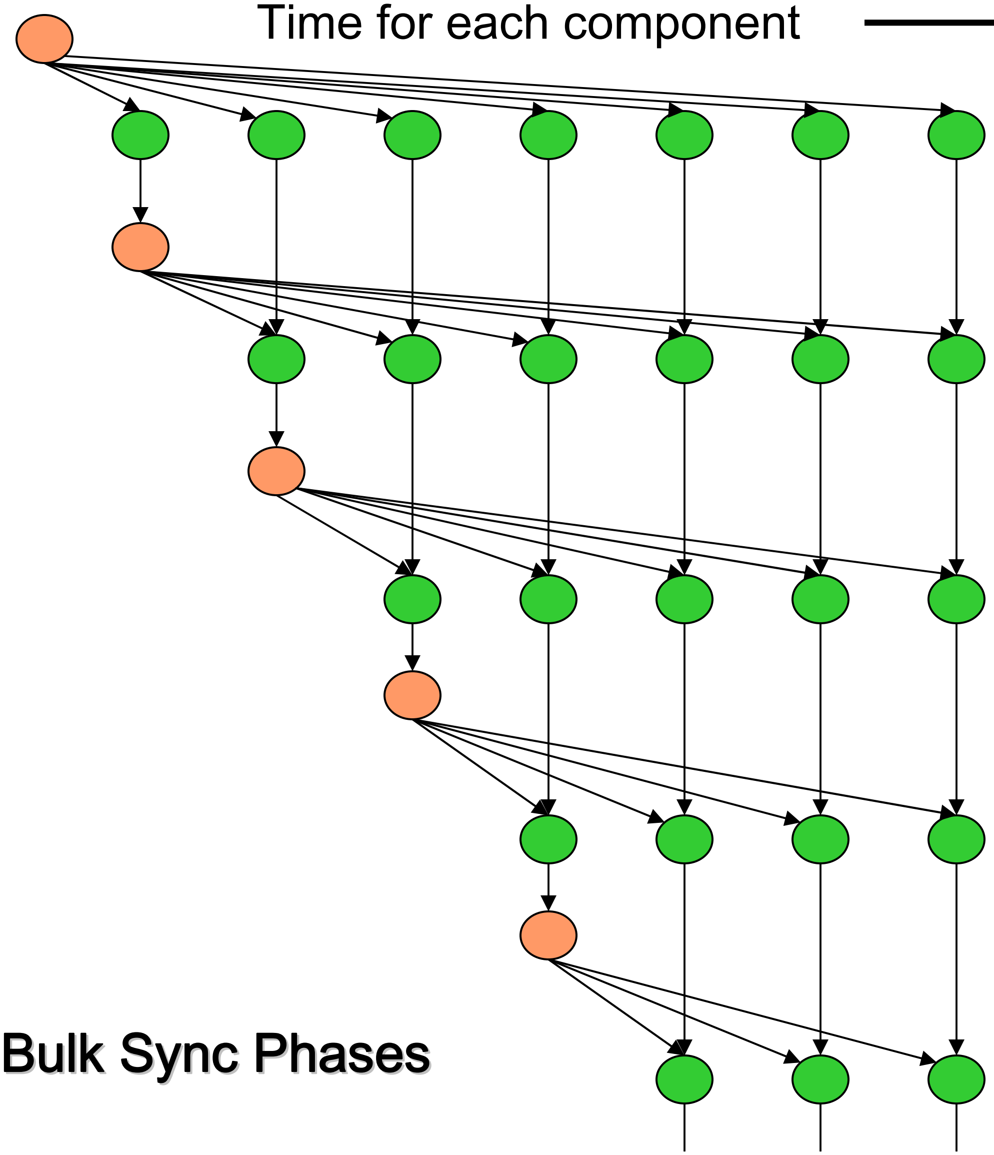
LU Timing Profile (4 processor system)

Threads – no lookahead

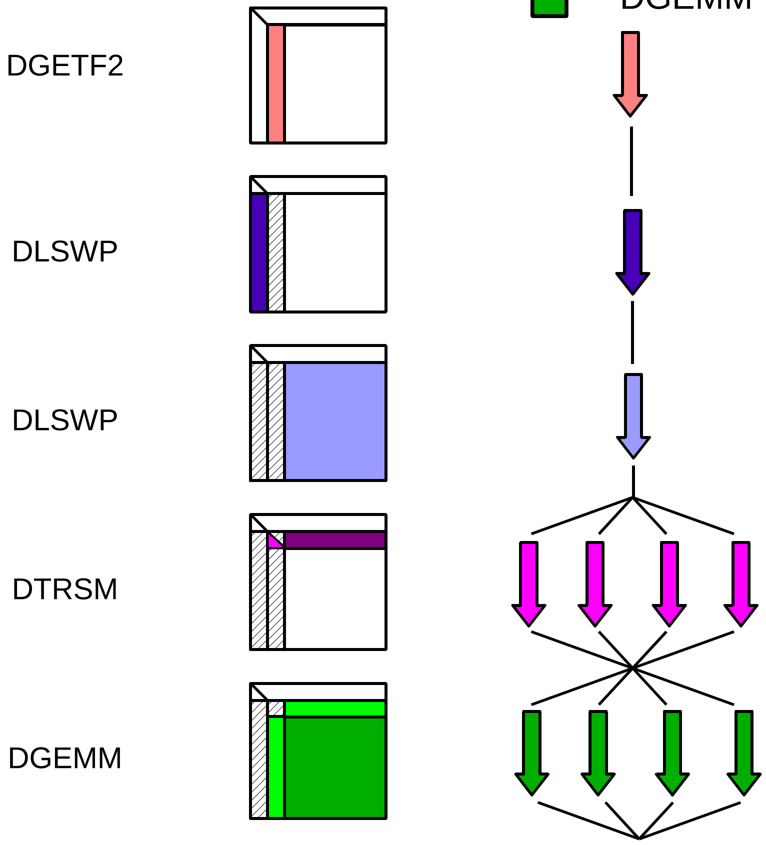


Time for each component → 1D decomposition and SGI Origir

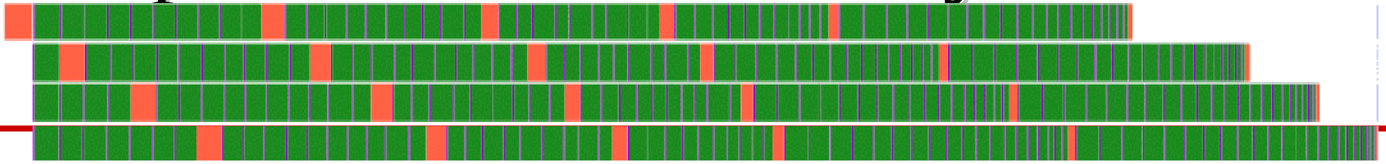
- 🟧 DGETF2
- 🟦 DLASWP(L)
- 🟦 DLASWP(R)
- 🟪 DTRSM
- 🟩 DGEMM

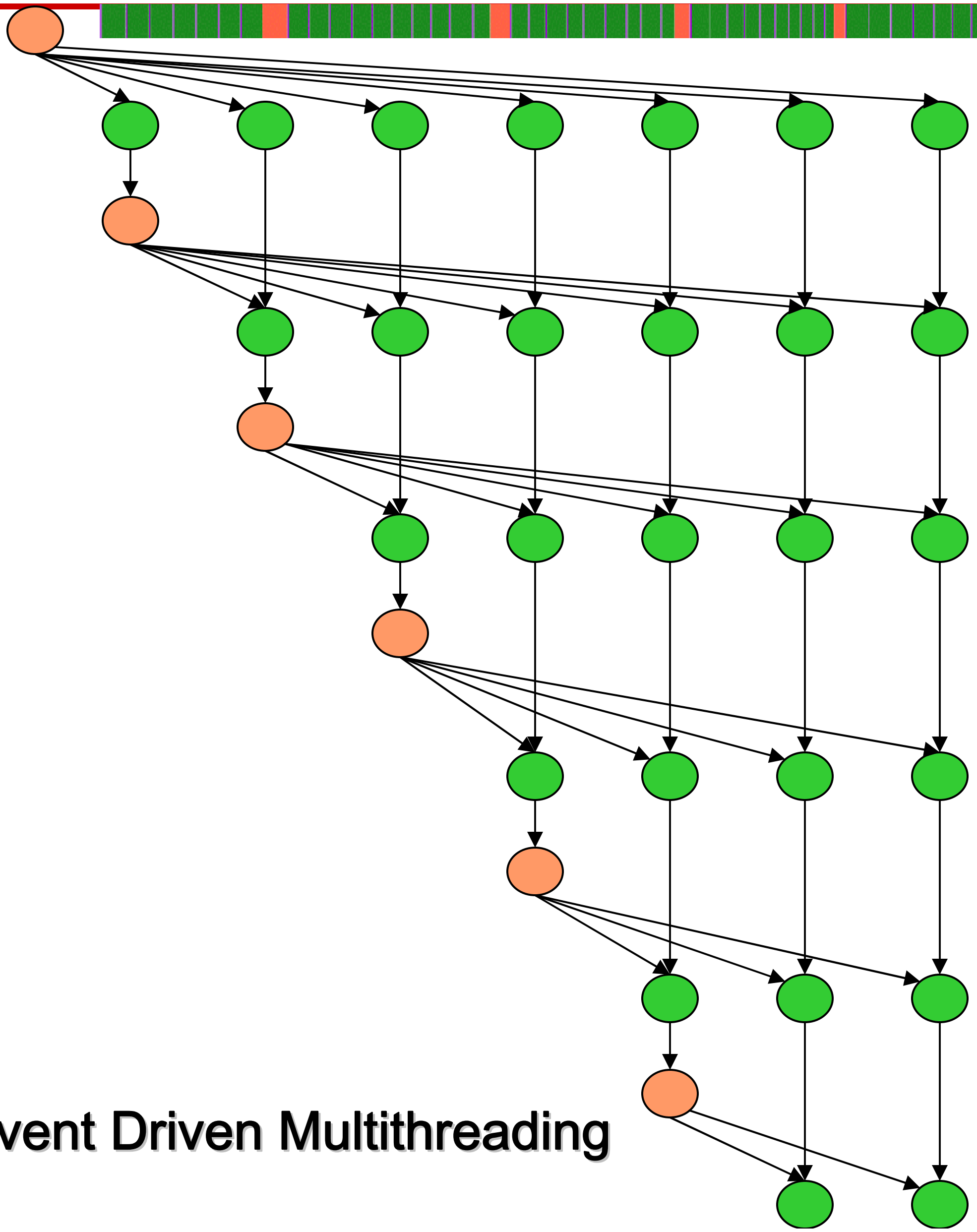


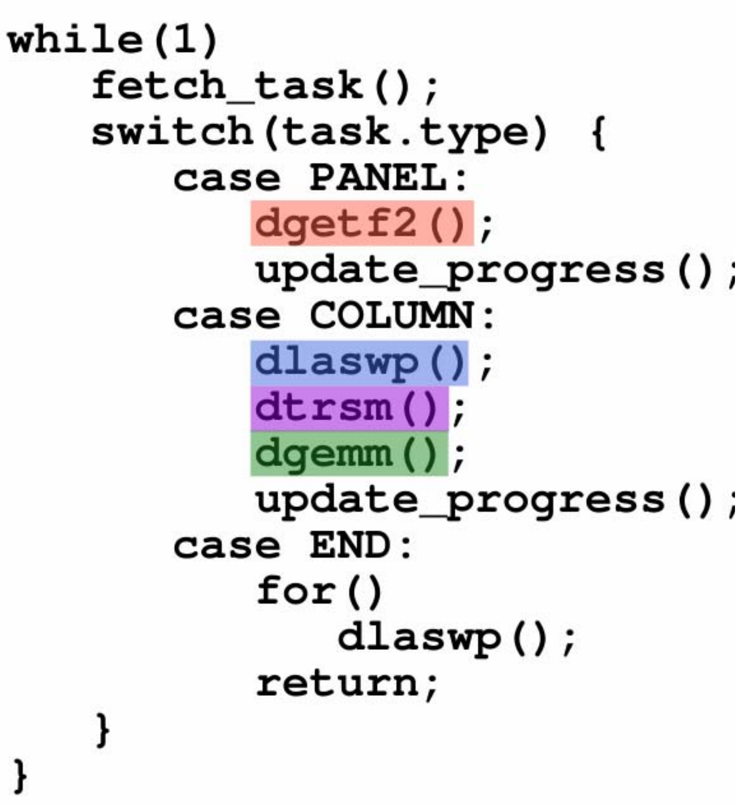
Bulk Sync Phases



Adaptive Lookahead - Dynamic





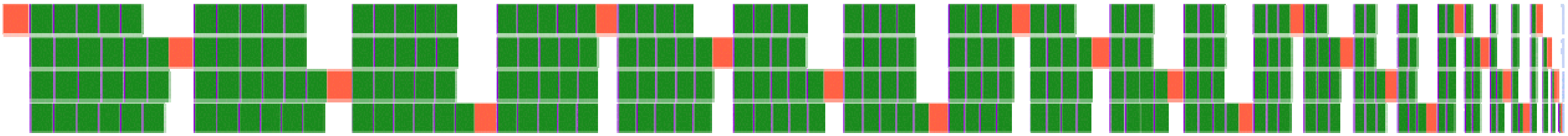


```
while(1)
    fetch_task();
    switch(task.type) {
        case PANEL:
            dgetf2();
            update_progress();
        case COLUMN:
            dlaswp();
            dtrsm();
            dgemm();
            update_progress();
        case END:
            for()
                dlaswp();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

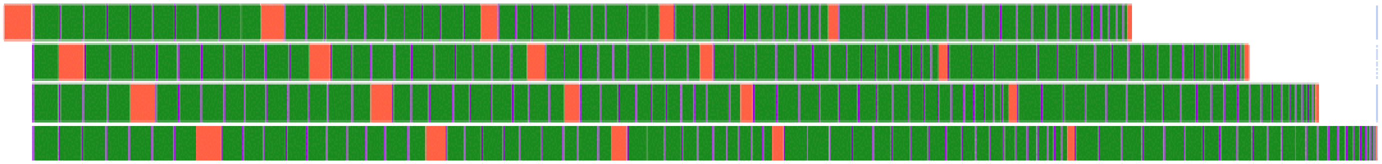
Event Driven Multithreading

Reorganizing algorithms to use this approach

LU – Fixed Lookahead – 4 processors



Original LAPACK Code



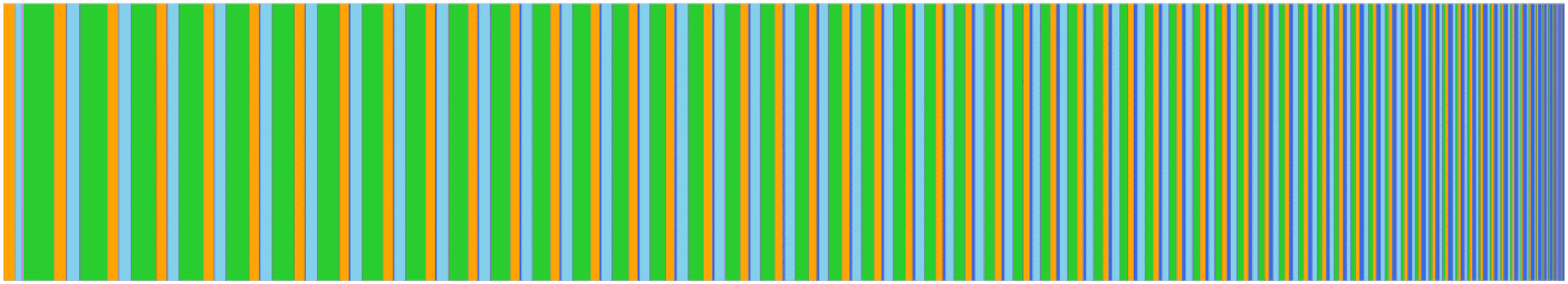
Data Flow Code

Time

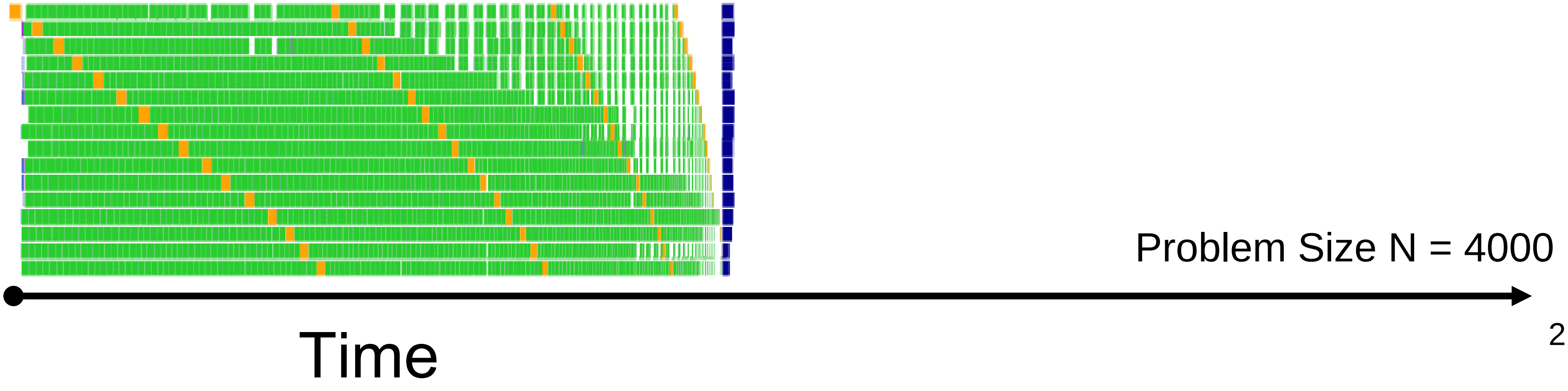
LU - BLAS Threads vs. Dynamic Lookahead

SGI Origin 3000 / 16 MIPS R14000 500 Mhz

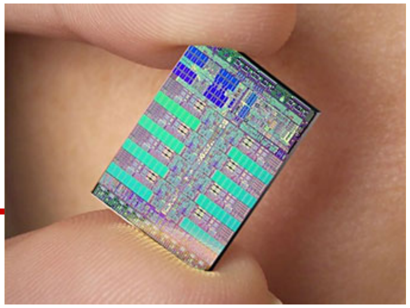
BLAS Threads (LAPACK)



Dynamic Lookahead

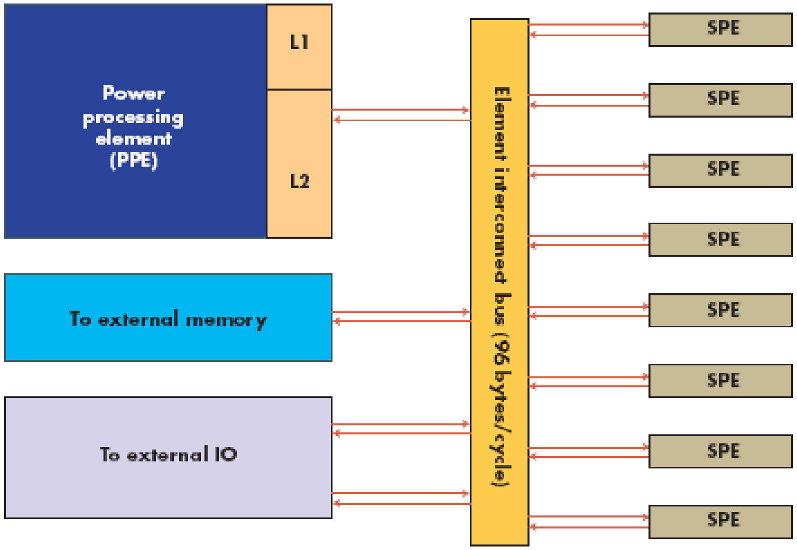


Taking a Look at the PlayStation 3



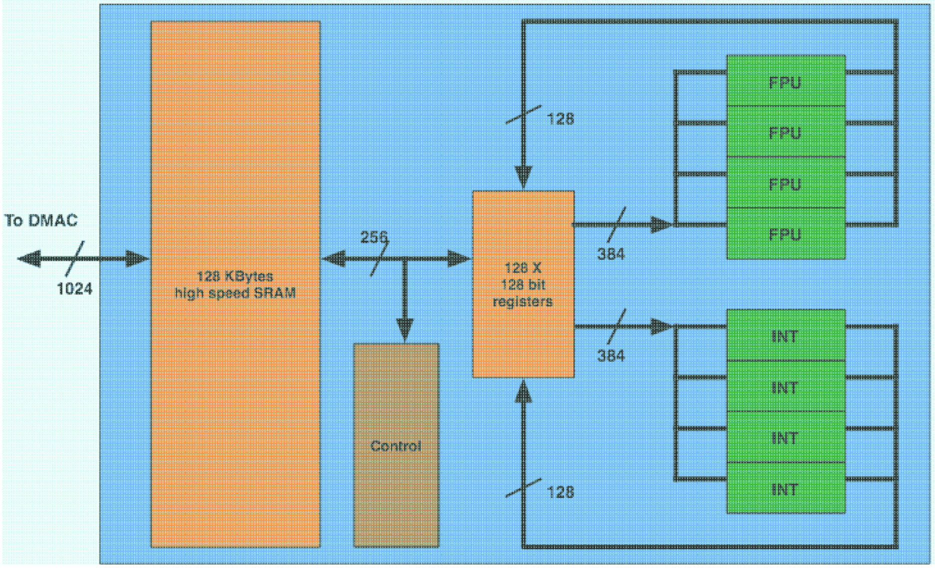
- The PlayStation 3's CPU based on a "Cell“ processor
- Each Cell contains 8 APUs.
  - An SPE is a self contained vector processor which acts independently from the others.
  - 4 floating point units capable of a total of 25.6 Gflop/s (6.4 Gflop/s each @ 3.2 GHz)
  - 204.8 Gflop/s peak! 32 bit floating 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.

Top-level block diagram of the Cell Broadband Engine (CBE)


Power processing element (PPE)
L1
L2
To external memory
To external IO
Element interconnect bus (96 bytes/cycle)
SPE
SPE
SPE
SPE
SPE
SPE
SPE
SPE

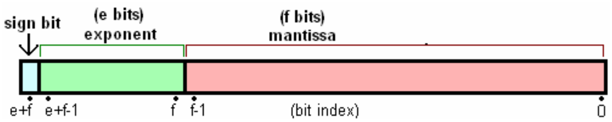
Cell APU Architecture

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.

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 AltaVec
  - 8 flops/cycle SP
  - 4 flops/cycle DP
    - No DP on AltaVec

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) AltaVec	18.28	9.98	1.83

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

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.

Mixed-Precision Iterative Refinement

Solve: $Ax = b$

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

32 and 64 Bit Floating Point Arithmetic

- Wilkinson, Moler, Stewart, & Higham provide error bound for LU Decomposition 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!

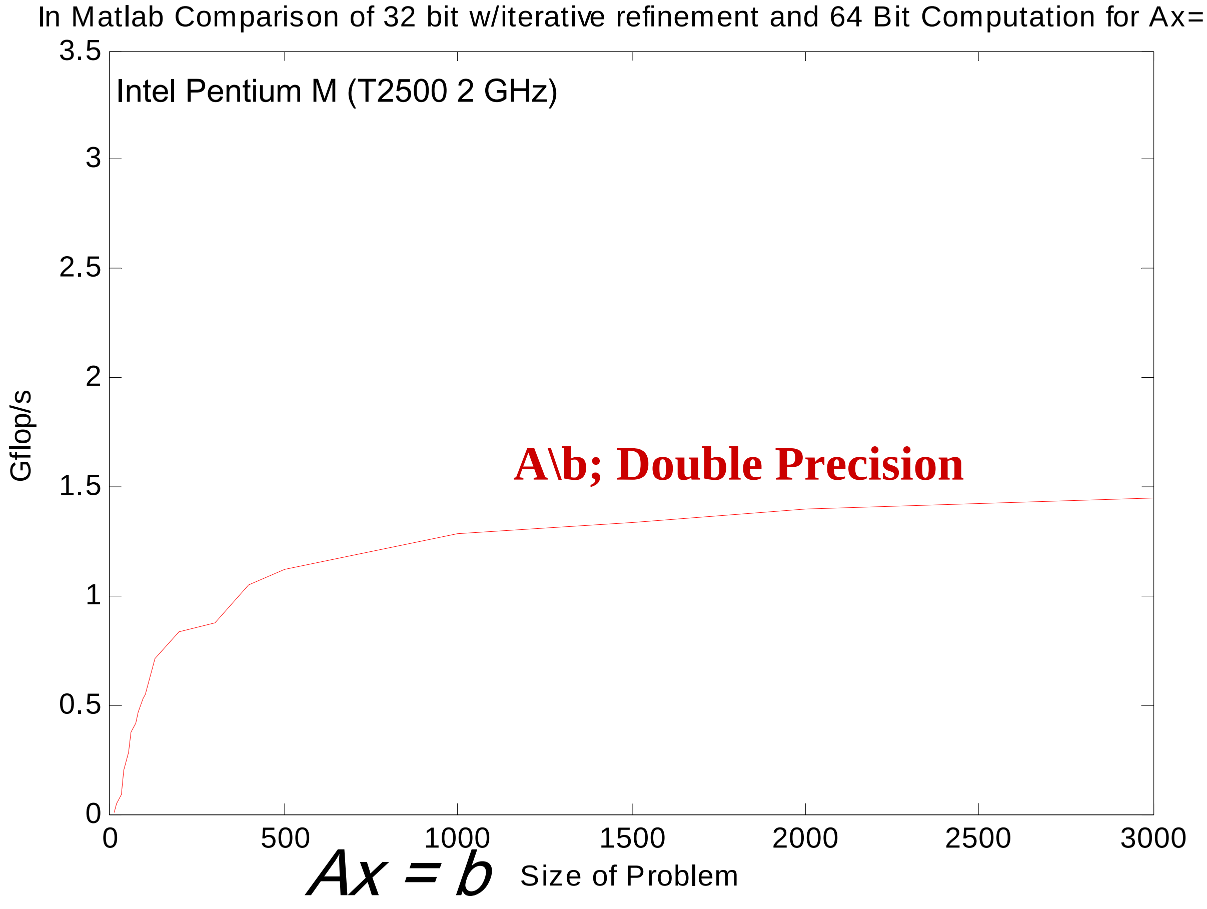
- Matlab has the ability to perform 32 bit floating point for some computations
  - Matlab uses LAPACK and MKL BLAS underneath.

```
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

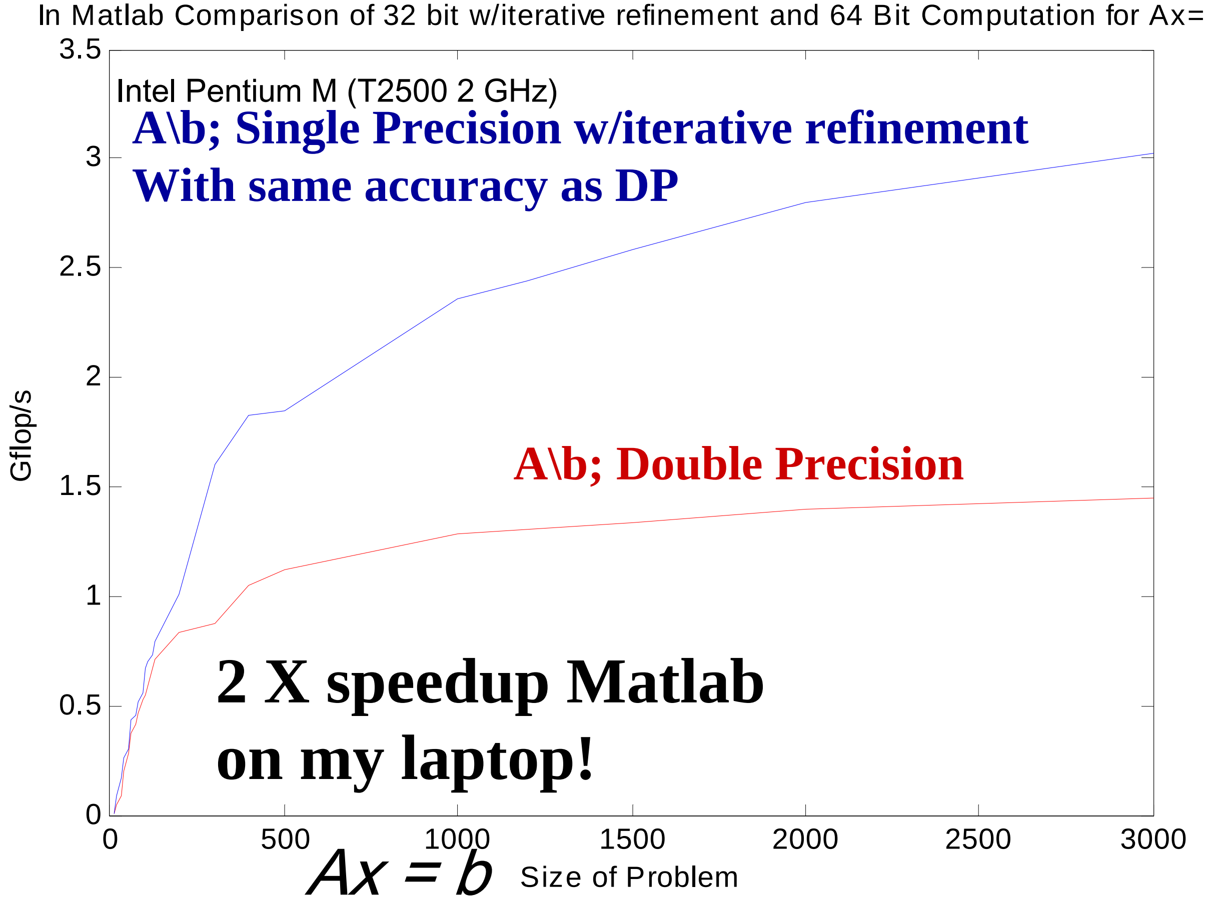
- On a Pentium; using SSE2, single precision can perform 4 floating point operations per cycle and in double precision 2 floating point operations per cycle.
- In addition there is reduced memory traffic (factor on sp data)



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)



2 X speedup Matlab on my laptop!

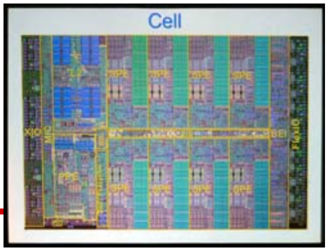
3 GFlop/s!!

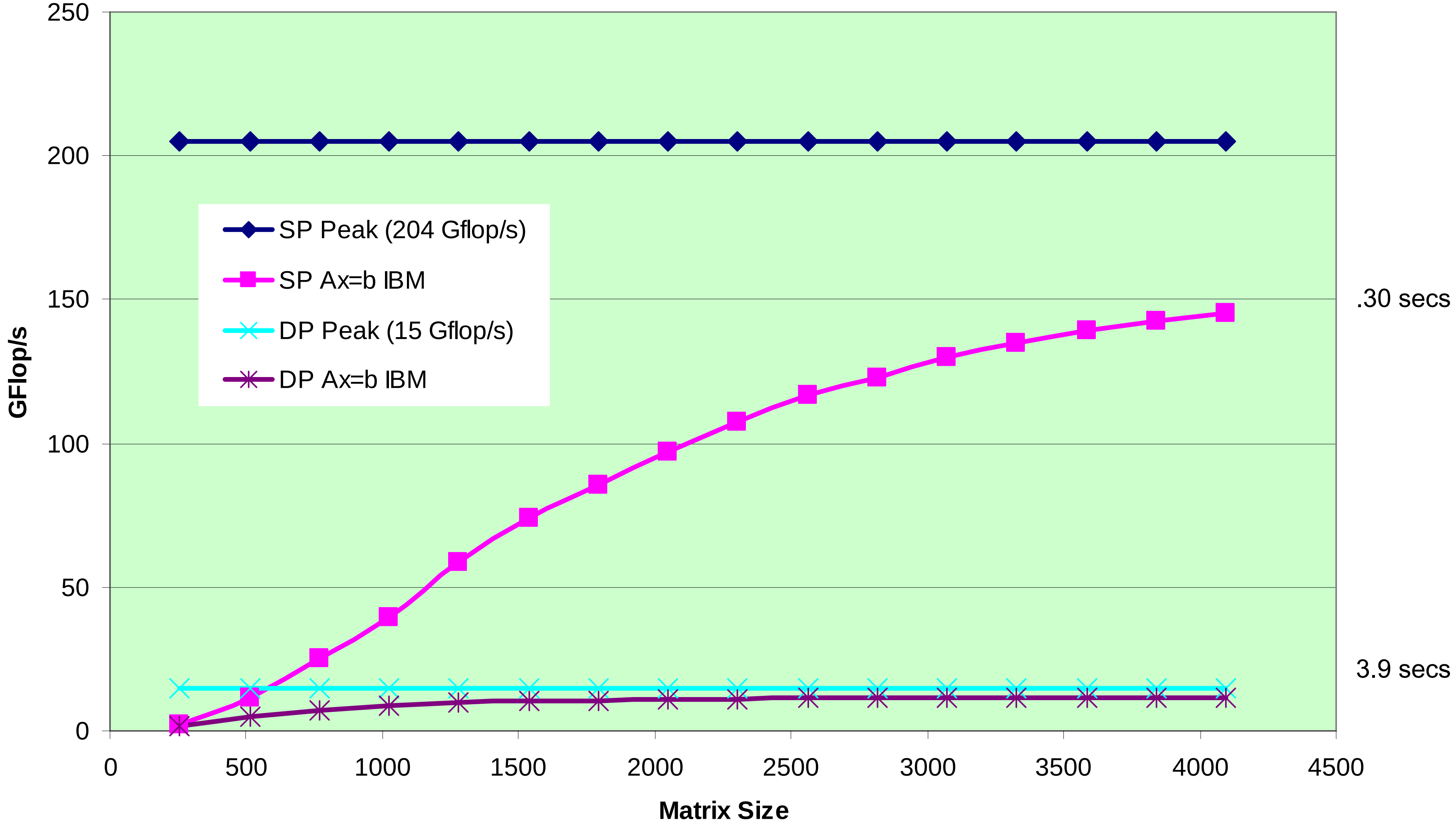
Speedups for Ax = b (Ratio of Times)

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

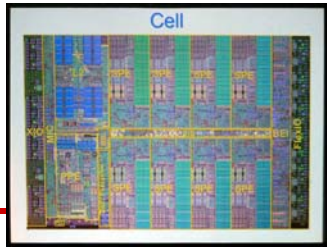
Architecture (BLAS-MPI)	# procs	*n*	DP Solve /SP Solve	DP Solve /Iter Ref	# iter
AMD Opteron (Goto – OpenMPI MX)	32	22627	1.85	1.79	6
AMD Opteron (Goto – OpenMPI MX)	64	32000	1.90	1.83	6

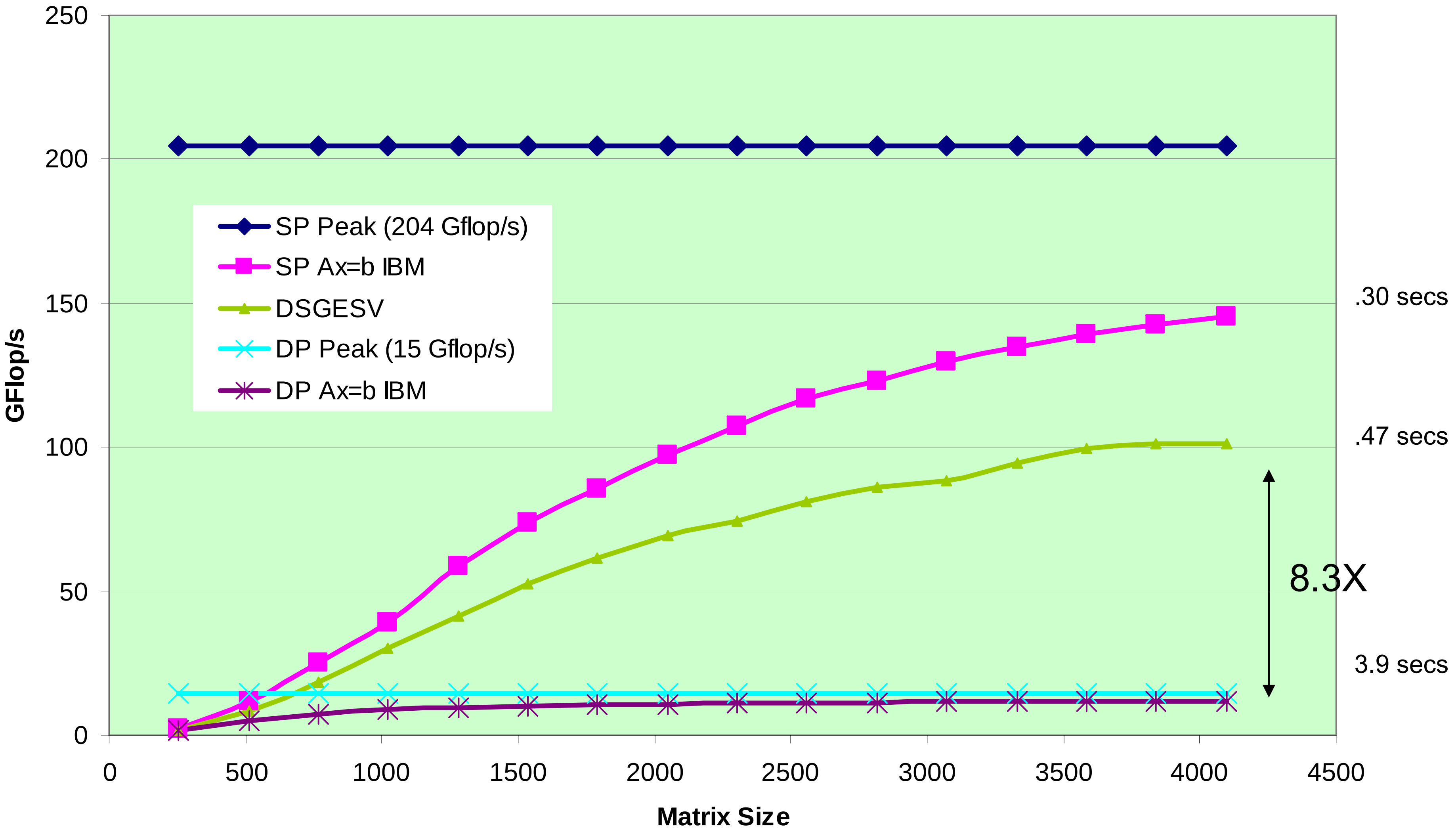
IBM Cell 3.2 GHz, Ax = b





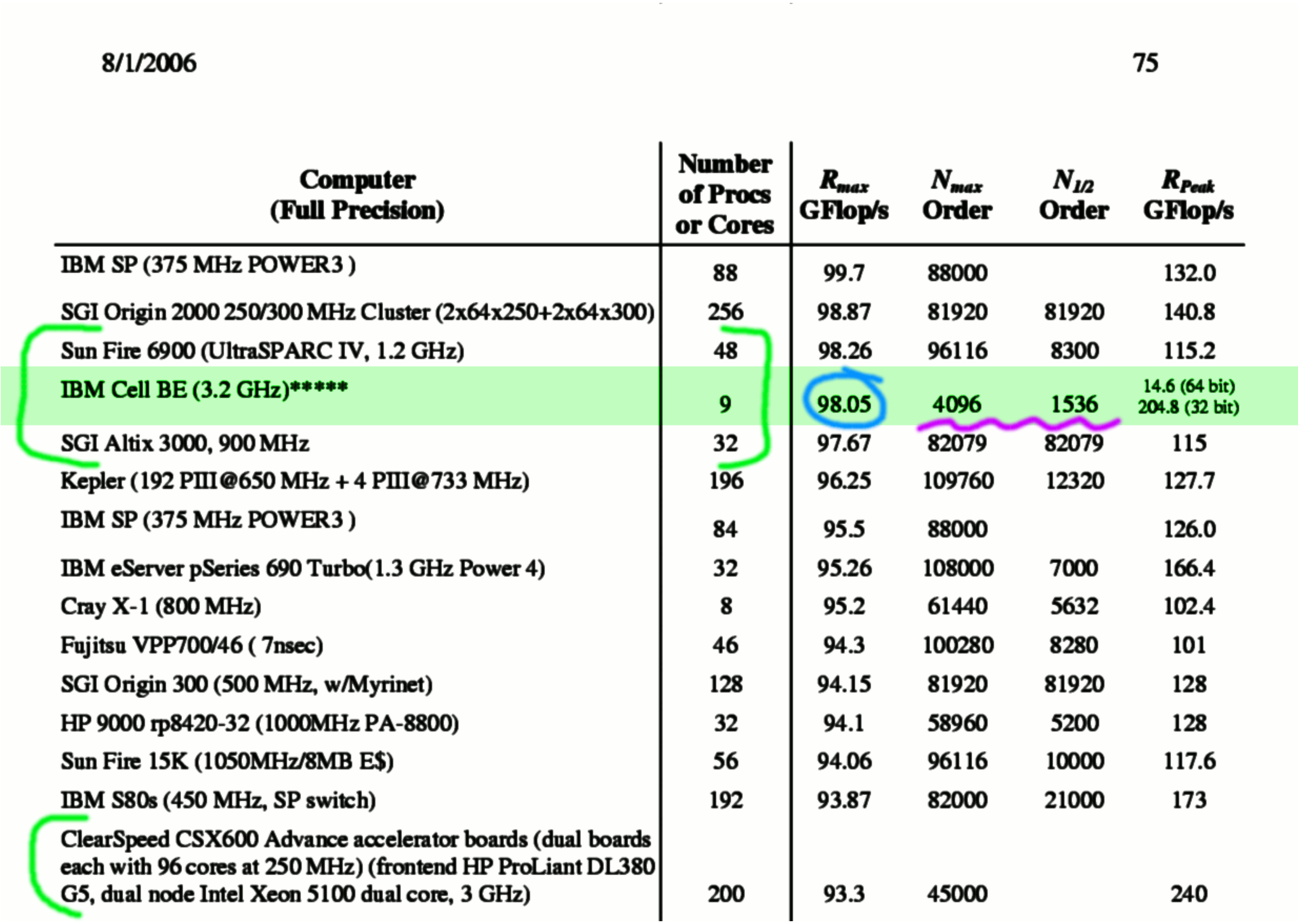
IBM Cell 3.2 GHz, Ax = b





LINPACK Benchmark

Potential Realized



8/1/2006 75

Computer (Full Precision)	Number of Procs or Cores	R_{max} GFlop/s	N_{max} Order	$N_{1/2}$ Order	R_{Peak} GFlop/s
IBM SP (375 MHz POWER3)	88	99.7	88000		132.0
SGI Origin 2000 250/300 MHz Cluster (2x64x250+2x64x300)	256	98.87	81920	81920	140.8
Sun Fire 6900 (UltraSPARC IV, 1.2 GHz)	48	98.26	96116	8300	115.2
IBM Cell BE (3.2 GHz)*****	9	98.05	4096	1536	14.6 (64 bit) 204.8 (32 bit)
SGI Altix 3000, 900 MHz	32	97.67	82079	82079	115
Kepler (192 PIII@650 MHz + 4 PIII@733 MHz)	196	96.25	109760	12320	127.7
IBM SP (375 MHz POWER3)	84	95.5	88000		126.0
IBM eServer pSeries 690 Turbo(1.3 GHz Power 4)	32	95.26	108000	7000	166.4
Cray X-1 (800 MHz)	8	95.2	61440	5632	102.4
Fujitsu VPP700/46 (7nsec)	46	94.3	100280	8280	101
SGI Origin 300 (500 MHz, w/Myrinet)	128	94.15	81920	81920	128
HP 9000 rp8420-32 (1000MHz PA-8800)	32	94.1	58960	5200	128
Sun Fire 15K (1050MHz/8MB E$)	56	94.06	96116	10000	117.6
IBM S80s (450 MHz, SP switch)	192	93.87	82000	21000	173
ClearSpeed CSX600 Advance accelerator boards (dual boards each with 96 cores at 250 MHz) (frontend HP ProLiant DL380 G5, dual node Intel Xeon 5100 dual core, 3 GHz)	200	93.3	45000		240

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

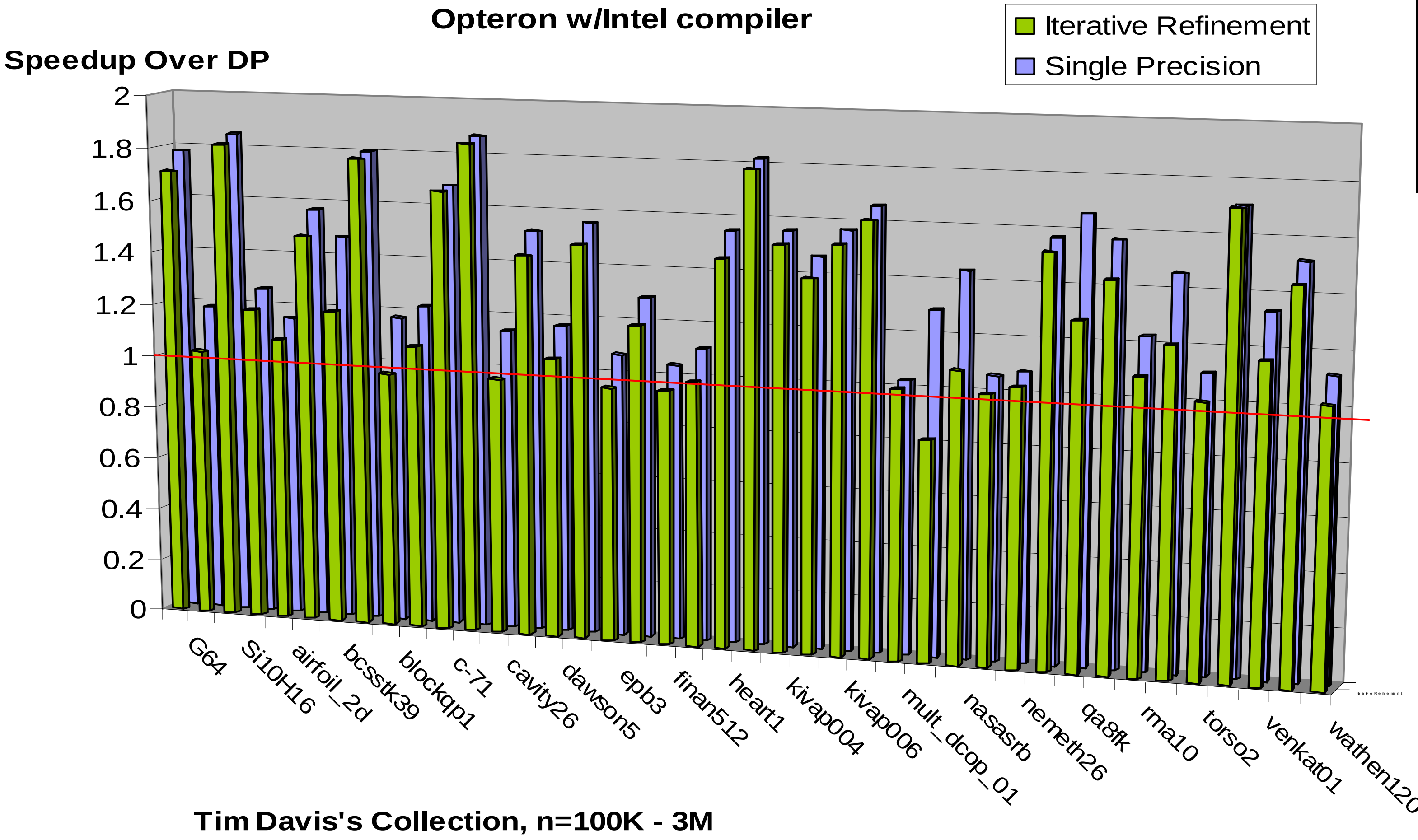
Refinement Technique Using Single/Double Precision

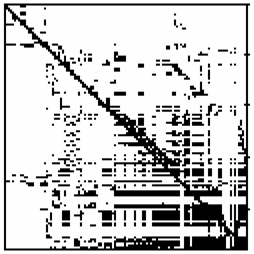
- Linear Systems
  - LU dense (in current release of LAPACK) and sparse
  - Cholesky
  - QR Factorization
- Eigenvalue
  - Symmetric eigenvalue problem
  - SVD
  - Same idea as with dense systems,
    - Reduce to tridiagonal/bi-diagonal in lower precision, retain original data and improve with iterative 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.

Sparse Direct Solver and Iterative Refinement

MUMPS package based on multifrontal approach which generates small dense matrix multiplies



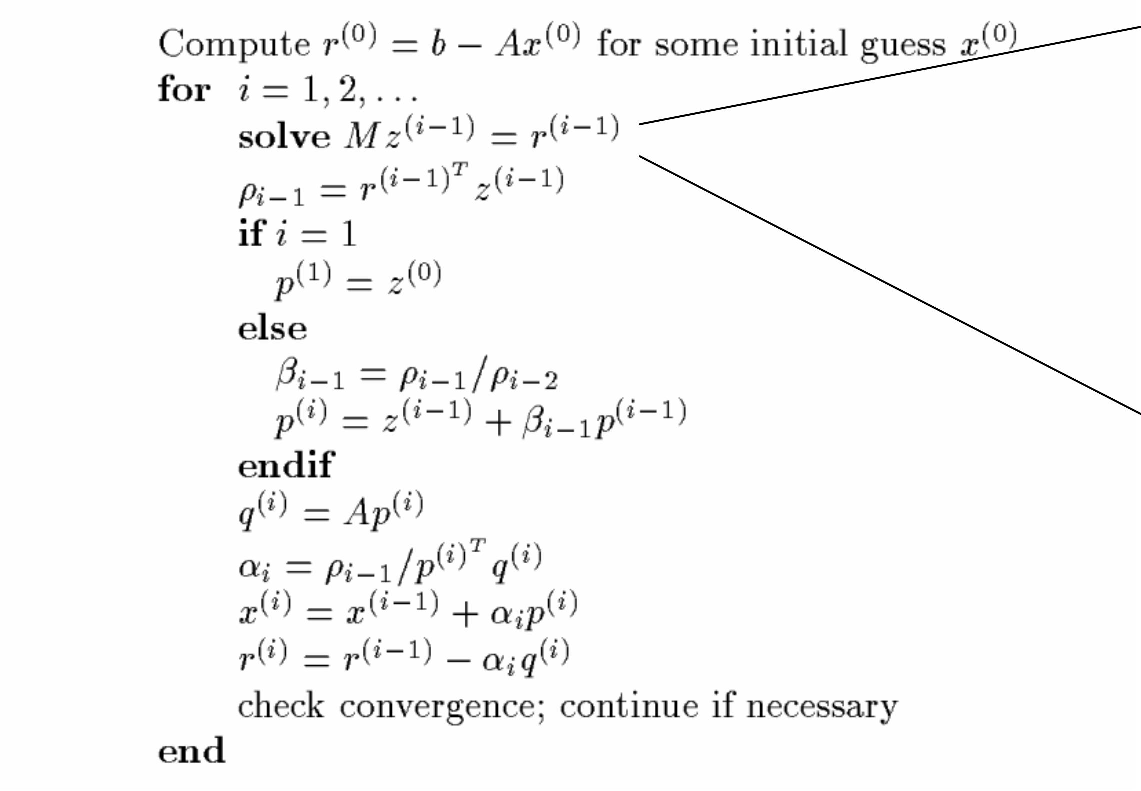


Sparse Iterative Methods (PCG)

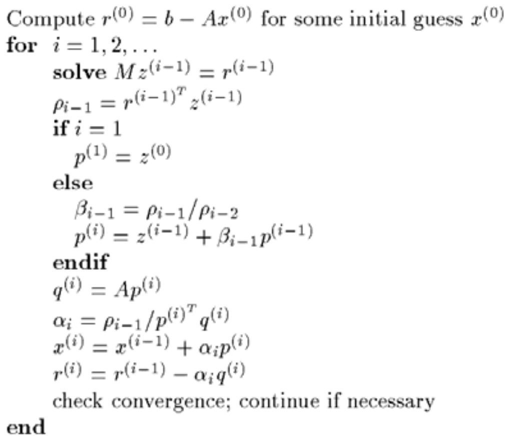
- **Outer/Inner Iteration**

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
```

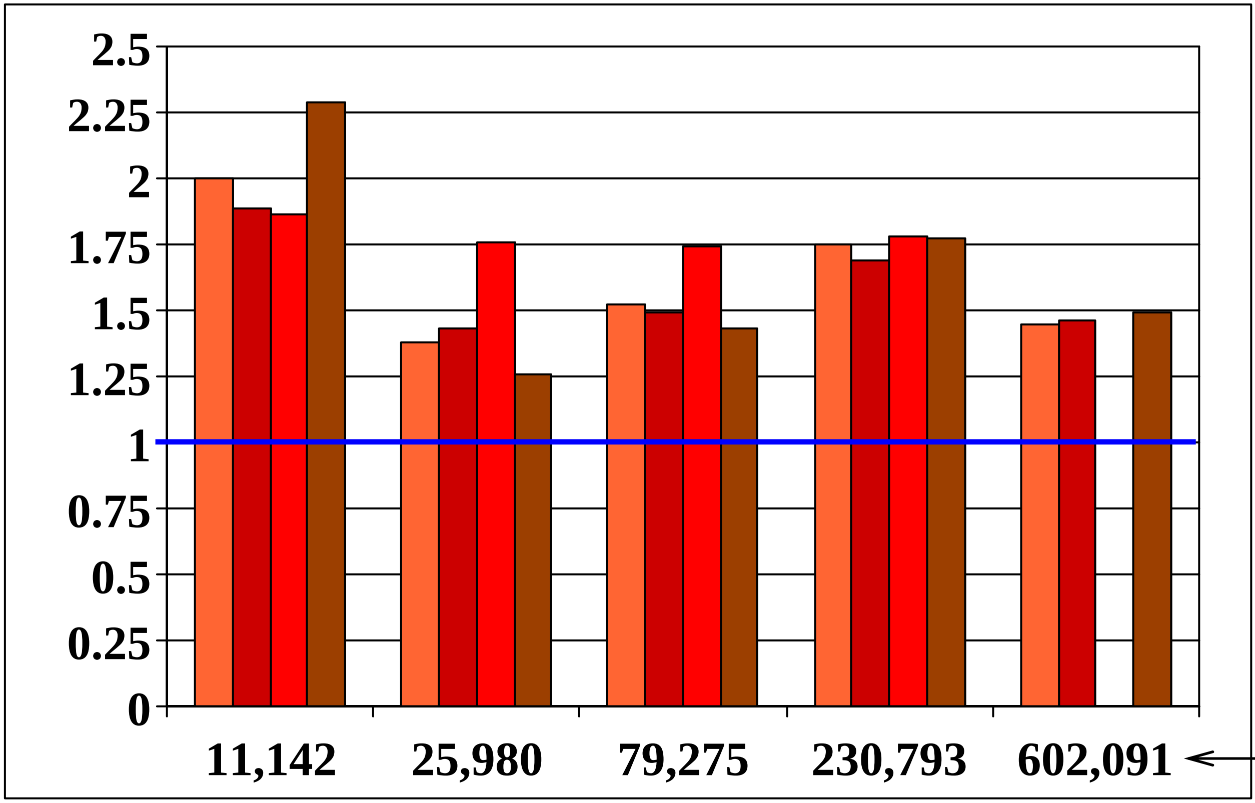


```
Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for $i = 1, 2, \ldots$
    solve $Mz^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
    if $i = 1$
        $p^{(1)} = z^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    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 fixed number of 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)

Reference methods

- 🟧 **CG**
- 🟥 **PCG**
- 🟥 **GMRES**
- 🟫 **PGMRES**

(*More is better*)

← Matrix size

6,021 18,000 39,000 120,000 240,000 ← Condition number

Data movement the main source of improvement

Intriguing Potential

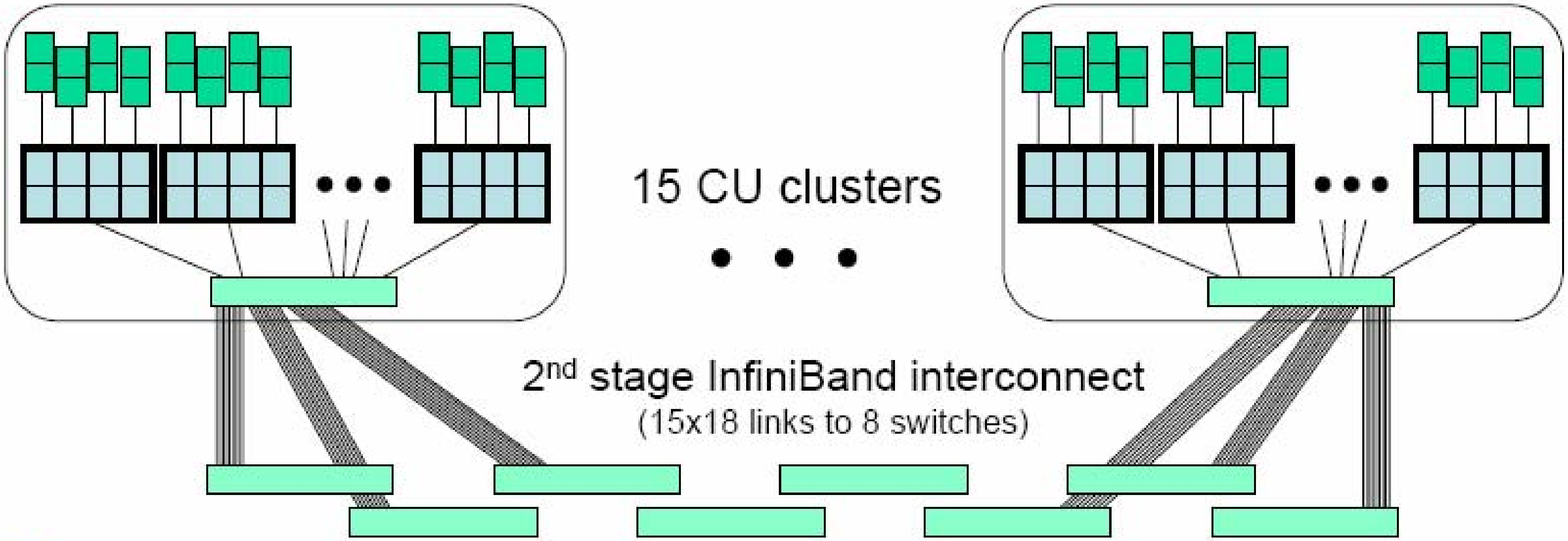
- Exploit lower precision as much as possible
  - Payoff in performance
    - Faster floating point
    - Less data to move
- Automatically switch between SP and DP to match the desired accuracy
  - Compute solution in SP and then a correction to the solution in DP
- Potential for GPU, FPGA, special purpose processors
  - What about 16 bit floating point?
  - 128 bit floating point?
- Linear systems and Eigenvalue problems

Accelerated Roadrunner

Hybrid Design

“Connected Unit” cluster
144 quad-socket
dual-core nodes
(138 w/ 4 dual-Cell blades)
InfiniBand interconnects

In aggregate:
8,640 dual-core Opterons + 16,560 eDP Cell chips
76 TeraFlops Opteron + ~1.7 PetaFlops Cell
75,600 cores

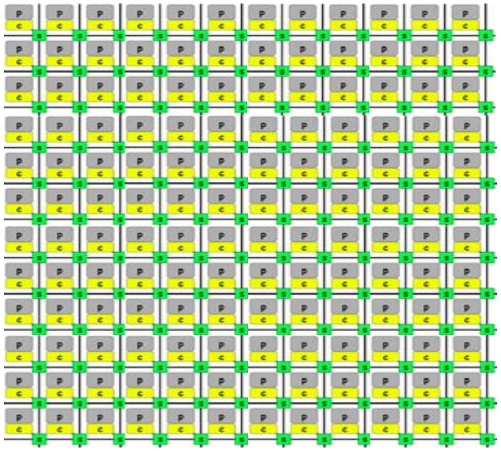


15 CU clusters

2nd stage InfiniBand interconnect
(15x18 links to 8 switches)

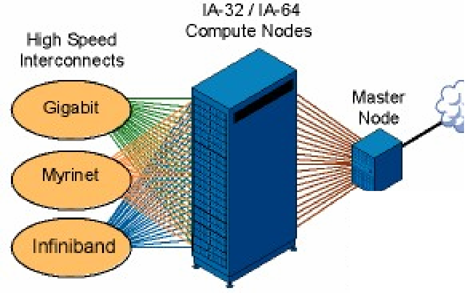
1152 AMD cores / cluster each core with a Cell processor

Future Large Systems, Say in 5 Years

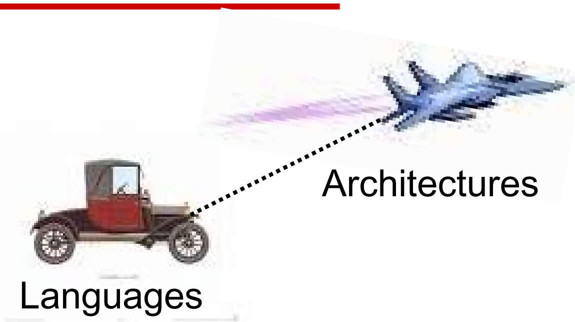


- 128 cores per socket
- 32 sockets per node
- 128 nodes per system

- 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
  - Languages out of touch with architecture
- 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.
- High Performance Ecosystem
  - Hardware, OS, Compilers, Software, Algorithms, Applications
    - No Moore's Law for software, algorithms and applications

Collaborators / Support

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

See webpage for tech reports.

Office of Science
U.S. DEPARTMENT OF ENERGY

Google™ Italia
Web Immagini Gruppi News Desktop altro »
Ricerca avanzata
Preferenze
Strumenti per le ling
Cerca con Google
Mi sento fortunato
Pubblicità - Soluzioni Aziendali - Tutto su Google - Google.com in English
©2006 Google

dongarra

Cerca: ● il Web ○ pagine in Italiano ○ pagine provenienti da: Italia

Many opportunities for students and post-docs in my group at UTK