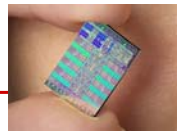
Exploiting the Performance of 32-bit Floating-Point Arithmetic in Obtaining 64-bit Accuracy (Computing on Games)

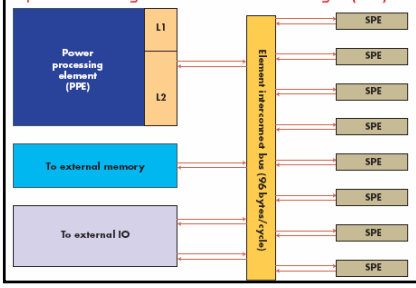
Jack Dongarra
University of Tennessee
and
Oak Ridge National Laboratory

With All the Hype on the PS3 We Became Interested

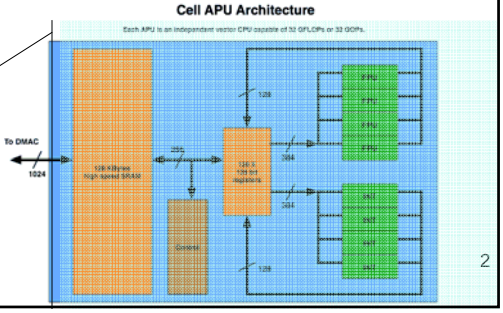


- The PlayStation 3's CPU based on a "Cell" processor
- Each Cell contains a Power PC processor and 8 SPEs. (SPE is processing unit, SPE: SPU + DMA engine)
  - An SPE is a self contained vector processor which acts independently from the others.
    - 4 way SIMD floating point units capable of a total 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

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



Cell APU Architecture



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.

3

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.

4

32 and 64 Bit Floating Point Arithmetic

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

```
L U = lu(A)                    O(n^3)
x = L\(U\b)                    O(n^2)
r = b – Ax                     O(n^2)
WHILE || r || not small enough
    z = L\(U\r)                O(n^2)
    x = x + z                  O(n^1)
    r = b – Ax                 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; $O(10^8)$

5

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.

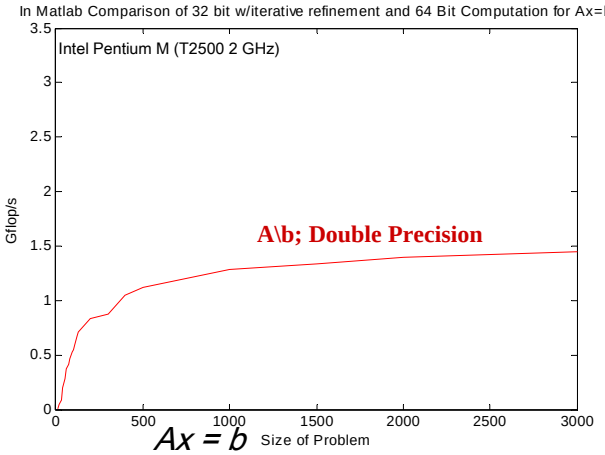
```
sa=single(a); sb=single(b);
[sl,su,sp]=lu(sa);                                  Most of the work: O(n^3)
sx=su\(sl\(sp*sb)); x=double(sx); r=b-a*x;                            O(n^2)
i=0;
while(norm(r)>res1),
    i=i+1;
    sr = single(r);
    sx1=su\(sl\(sp*sr)); x1=double(sx1); x=x1+x; r=b-a*x;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

6

Another Look at Iterative Refinement

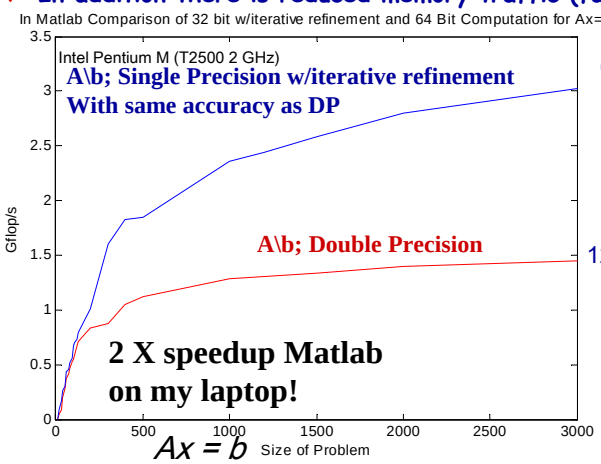
- On a Pentium; using SSE2, single precision can perform 4 floating point operations per cycle and in double precision 2 floating point operations per cycle.
- In addition there is reduced memory traffic (factor on sp data)



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)



6.1 sec

3 GFlop/s!!

12.8 sec

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

9

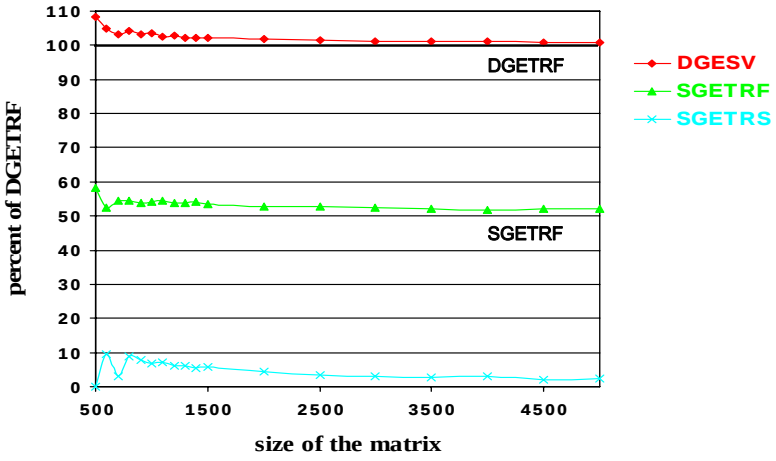
Speedups for Ax = b (Ratio of Times)

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

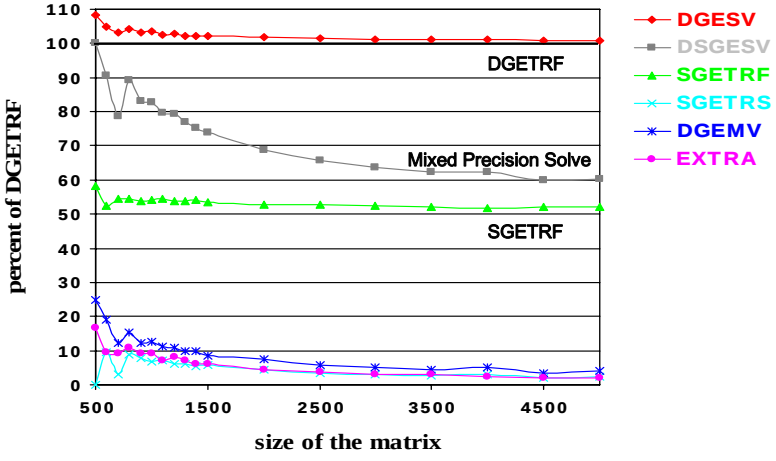
10

AMD Opteron Processor 240 (1.4GHz), Goto BLAS (1 thread)



11

AMD Opteron Processor 240 (1.4GHz), Goto BLAS (1 thread)



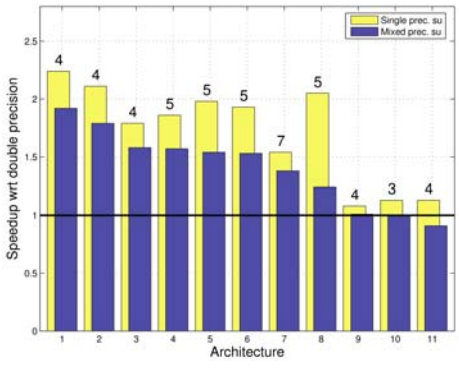
12

Bottom Line

- **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**

	Size	SGEMM/ DGEMM	Size	SGEMV/ DGEMV
AMD Opteron 246	3000	2.00	5000	1.70
Sun UltraSparc-IIe	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

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)

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

15

Refinement Technique Using Single/Double Precision

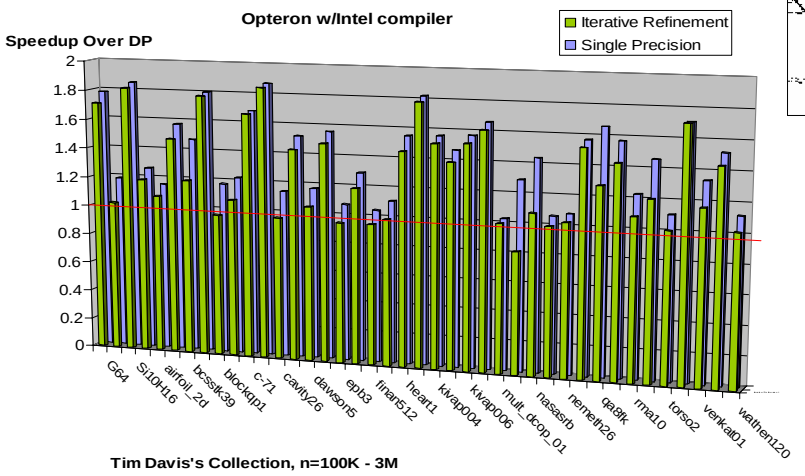
- Linear Systems
  - LU (dense and sparse)
  - Cholesky
  - QR Factorization
- Eigenvalue
  - Symmetric eigenvalue problem
  - SVD
  - Same idea as with dense systems,
    - Reduce to tridiagonal/bi-diagonal in lower precision, retain original data and improve with iterative technique using the lower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

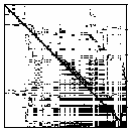
See webpage for tech report which discusses this.

16

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**

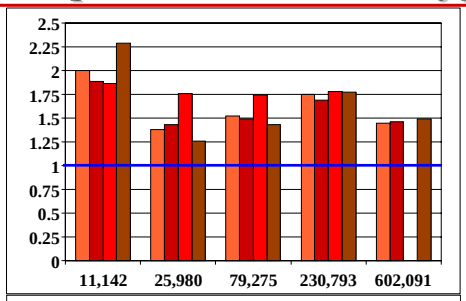
  Outer iterations using 64 bit floating point

Inner iteration:
In 32 bit floating point

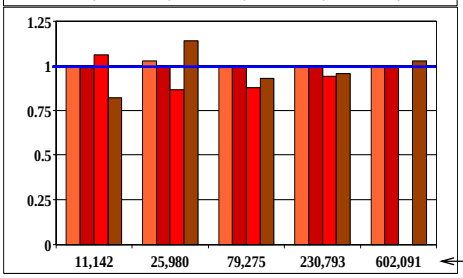
Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for $i = 1, 2, \ldots$
  solve $Mz^{(i-1)} = r^{(i-1)}$
  $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
  if $i = 1$
    $p^{(1)} = z^{(0)}$
  else
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
  endif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

- **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



Speedups for mixed precision
Inner SP/Outer DP (SP/DP) iter. methods *vs* DP/DP
(CG[2], GMRES[2], PCG[2], and PGMRES[2] with diagonal prec.)
(***Higher is better***)



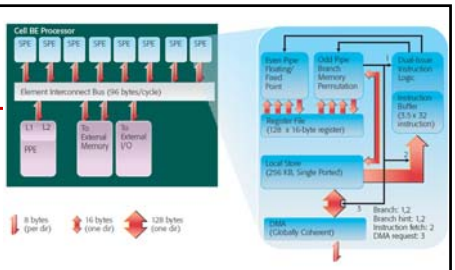
Iterations for mixed precision
SP/DP iterative methods *vs* DP/DP
(***Lower is better***)

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

Stopping criteria:
Relative to r_0 residual reduction (10^{-12})

19

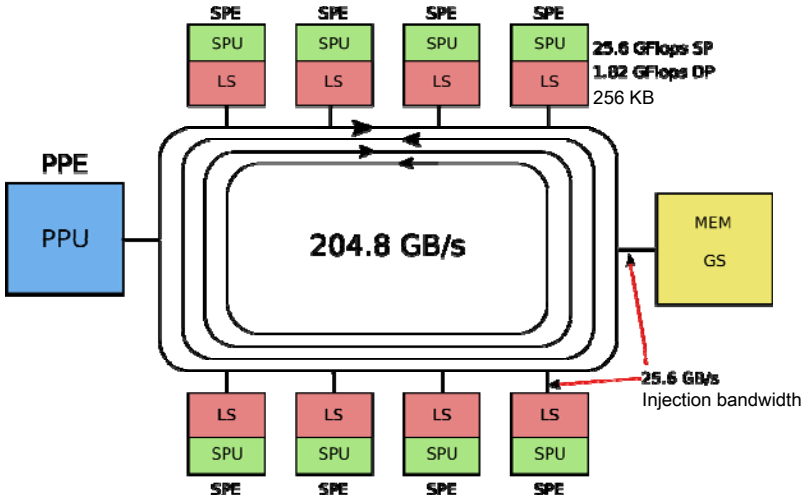
What about the Cell?



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

20

Moving Data Around on the Cell



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)

Cell Software for Iterative Refinement

- LAPACK FORTRAN 77 DSGESV at the top
  - LINPACK-SP (from IBM)
    - SGETRF
    - SGETRS
- Additional SPE-parallel code
  - Conversion from standard to block layout
  - Conversion from single to double precision
  - DLANGE – matrix norm (DP)
  - DGEMM – matrix multiply (DP)
- PPU auxiliary Level 1 BLAS (DAXPY, DLACPY, DNRM2)
- Block data layout (64x64 SP, 32x32 DP)

22

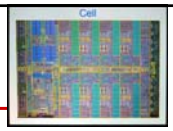
32 and 64 Bit Floating Point Arithmetic

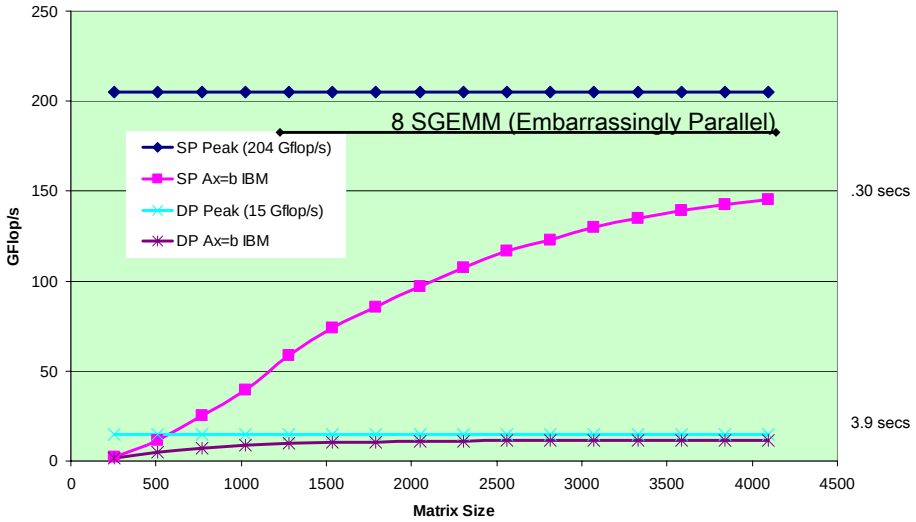
- **Iterative refinement for dense systems, $Ax = b$.**

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		

3

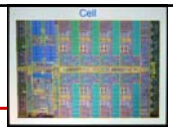
IBM Cell 3.2 GHz, Ax = b

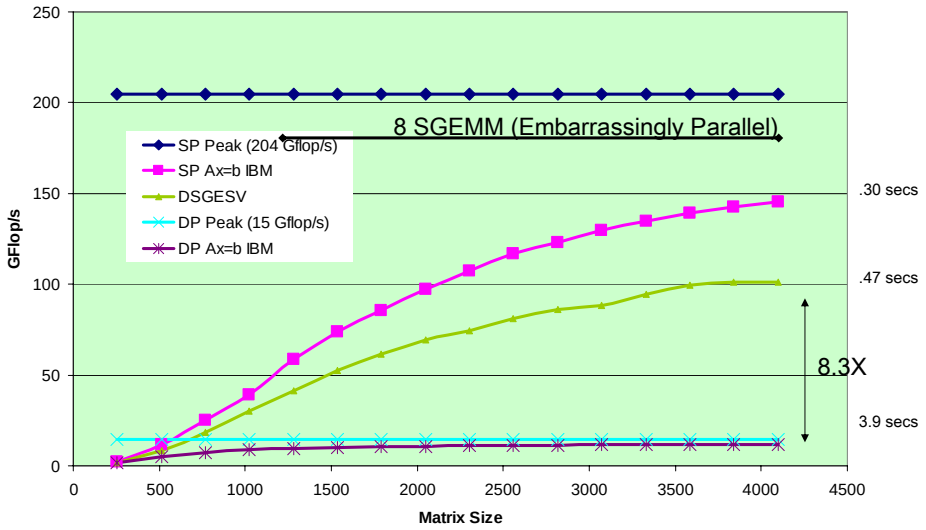




24

IBM Cell 3.2 GHz, Ax = b





25

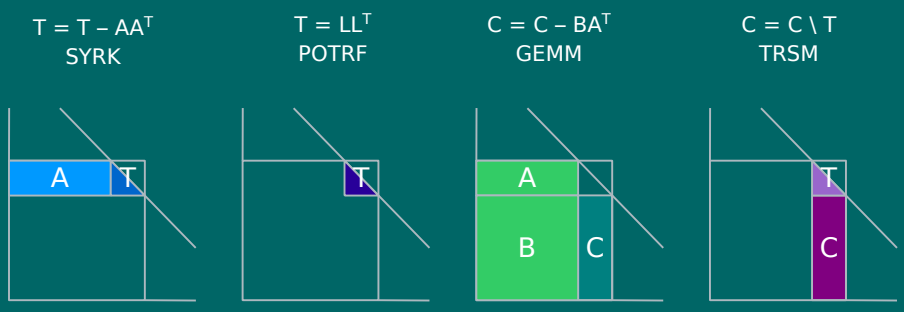
LINPACK Benchmark

Potential Realized

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 ClearSpeed boards each 250 MHz) (frontend HP ProLiant DL380 G5, dual node Intel Xeon 5100 dual core, 3 GHz)	6	93.3	45000		240

26

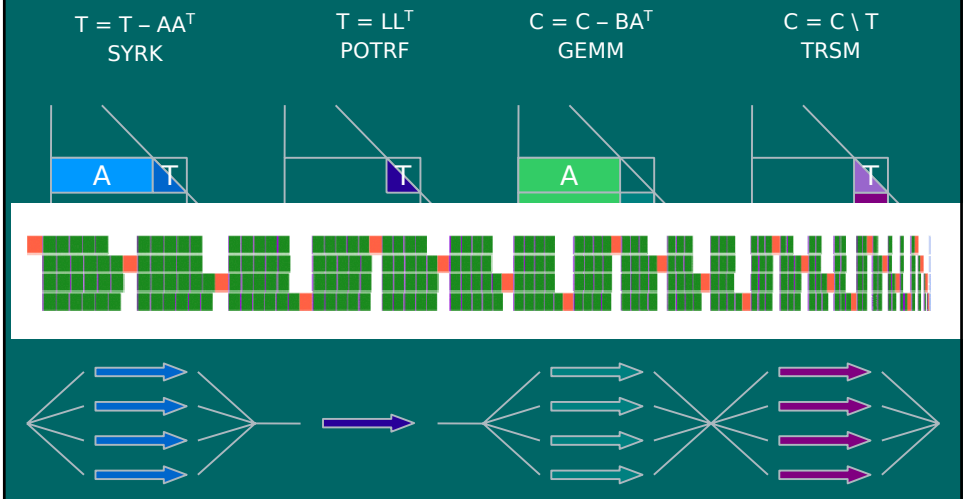
LAPACK Cholesky Factorization



SPE Parallelization:

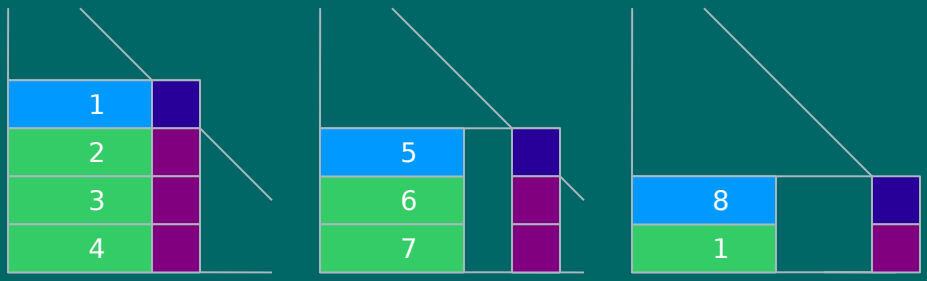
- Every operation chopped into 64x64 tiles,
  - 1, 2 or 3 tiles on input side,
  - 1 tile on output side.

Cholesky Factorization



Poor performance on many-core architectures because of sequential bottleneck and fork join parallelism

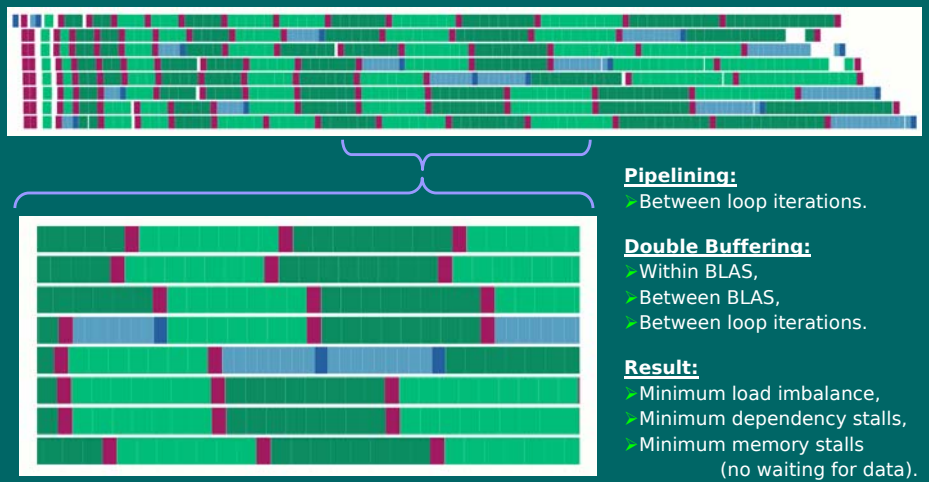
Pipelining Loop Iterations



1D Work Partitioning
- Facilitates data reuse,
- Prevents bus saturation.

2D Dependency Tracking
- Facilitates loop pipelining,
- Eliminates load imbalance.

Pipelining & Double Buffering

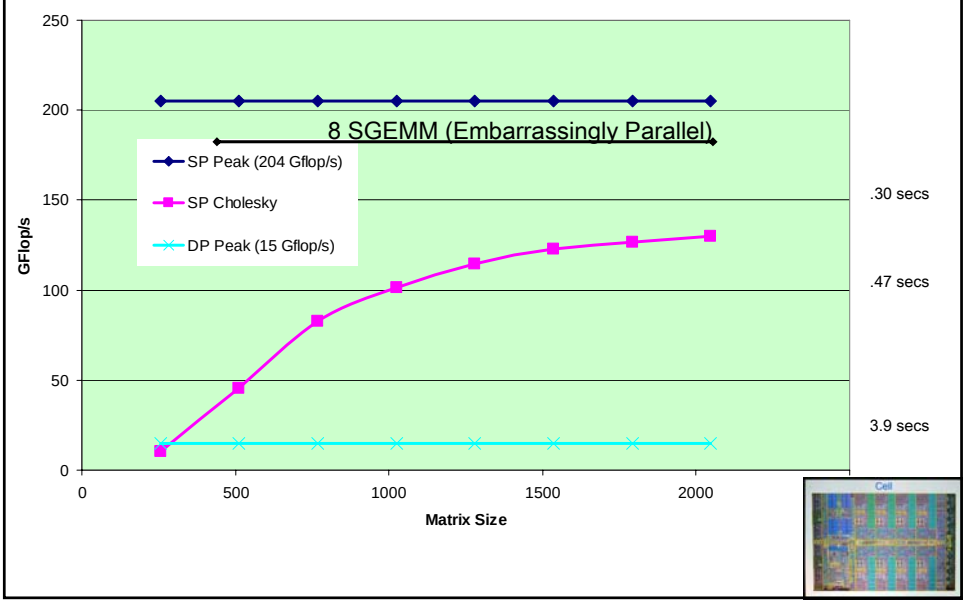


Pipelining:
- Between loop iterations.

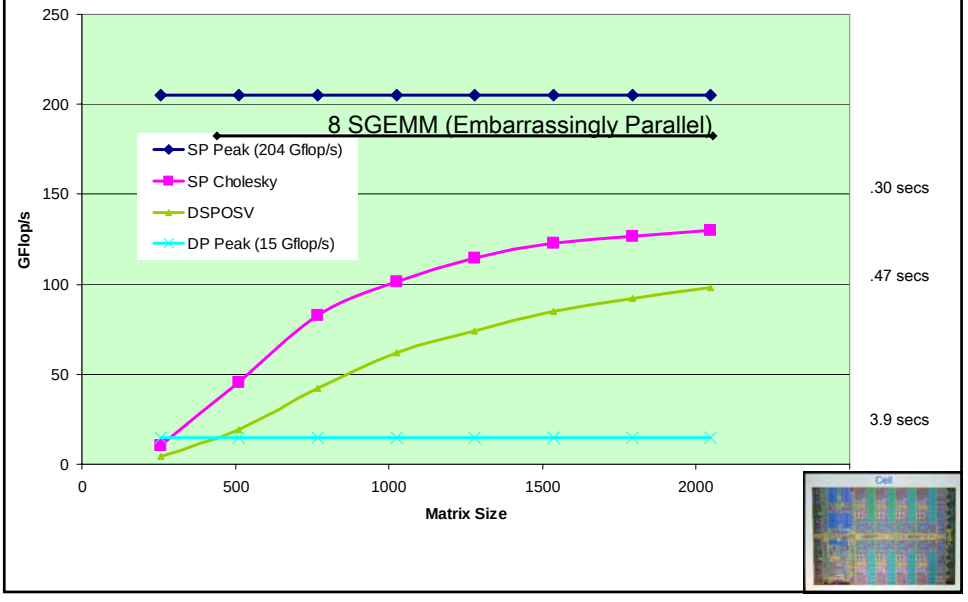
Double Buffering:
- Within BLAS,
- Between BLAS,
- Between loop iterations.

Result:
- Minimum load imbalance,
- Minimum dependency stalls,
- Minimum memory stalls (no waiting for data).

IBM Cell 3.2 GHz, Ax = b, A Sym Pos Def



IBM Cell 3.2 GHz, Ax = b, A Sym Pos Def



What About the PS3?

- Sony Playstation 3, release:
  - *November 11th, 2006* (Japan),
  - *November 17th, 2006* (North America) *and*
  - *March 2007* (Europe).
- The main elements to the Playstation 3 are a 7 SPE version of IBM's Cell processor and nVidia's Reality Synthesizer GPU.
  - Note that the Cell processor actually contains 8 SPEs, but for yield reasons Sony have decided to disable one of the cores.
- Each SPE has 256KB of local memory
- Seven SPEs, of which one is dedicated to OS tasks (the remaining 6 can be used as floating point units).
  - Now we are down to 6 SPE's
- PS3 connects to 256MB of Rambus XDR memory clocked at 3.2Ghz, giving a memory bandwidth of 25.6 GB/s.
- The PPE features 64KB L1 cache, 512KB L2 cache and also features Symmetric Multithreading (i.e. two threads can run concurrently rather like Intel's Hyperthreading).
- The PS3 additionally supports a removable hard disk, which will be available in either 60GB (Premium model) or 20GB (Basic model) sizes - although any size drive can be inserted

33

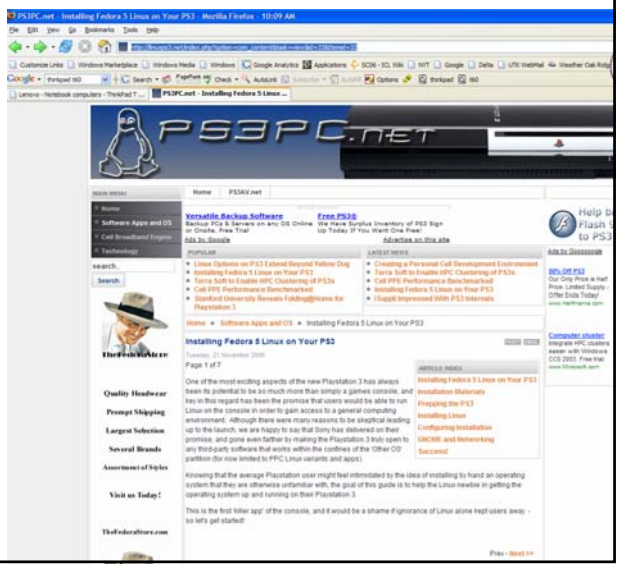
Sony Playstation 3

- **$600 + a monitor for HDTV output**

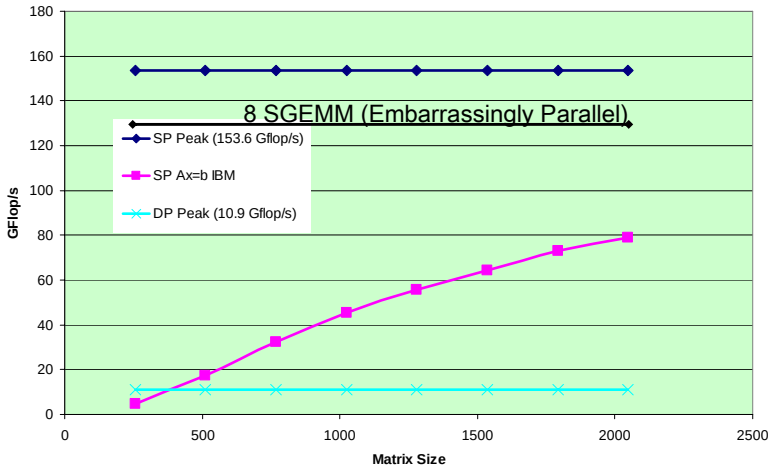
34

Price Comparison

- From IBM or Mercury
  - 2 Cell chip
    - Each w/8 SPEs
  - 512 MB/Cell
  - ~$17K
  - Some SW
- From WAL*MART PS3
  - 1 Cell chip
    - w/6 SPEs
  - 256 MB/PS3
  - $600
  - Download SW
  - Dual boot

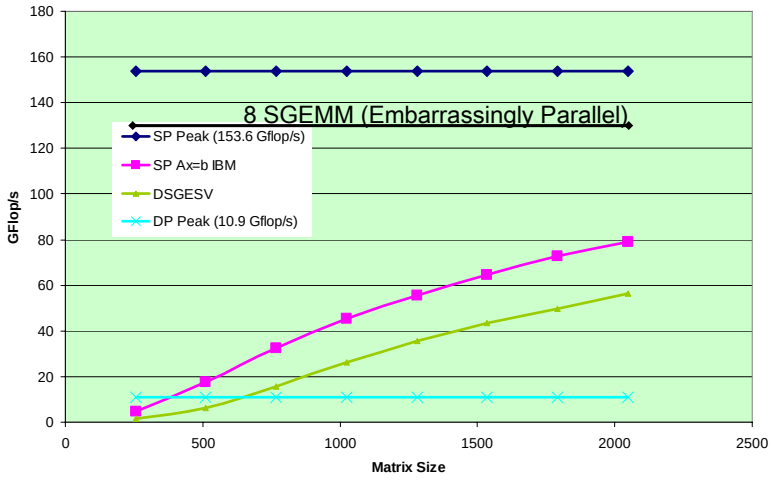


PlayStation 3 LU Codes



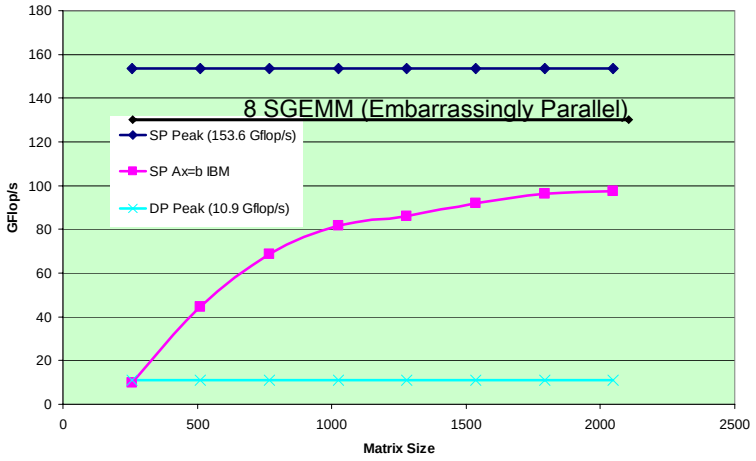
36

PlayStation 3 LU Codes



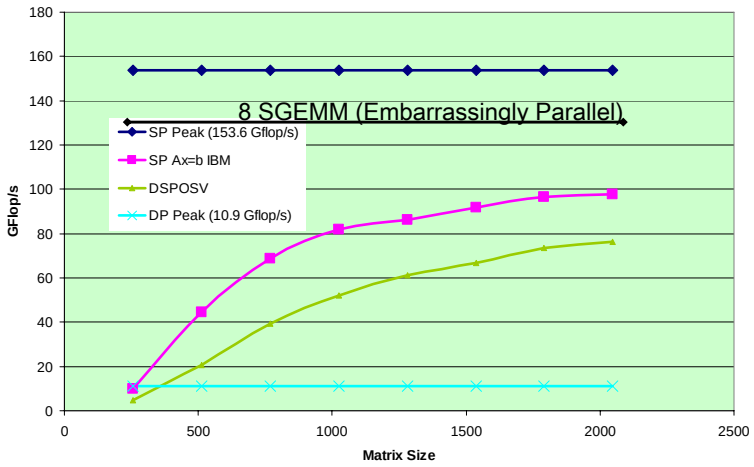
37

PlayStation 3 Cholesky Codes



38

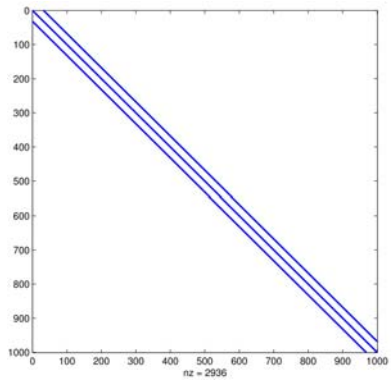
PlayStation 3 Cholesky Codes



39

A Sparse Matrix on the Cell

One lucky case:



The Good:
- stride-1 access on the source vector
- easy vectorization
- regular memory access pattern
- big chunks of data may be fetched at once

The Bad:
- still no surface to volume as in matrix multiply

For Performance: Upper bound is bus speed if no reuse.

PCG on the Cell: Grouping Operations (ops/data movement)

$r_0 = b - Ax_0$
$z_1 = rM^{-1}$
$\rho_1 = \langle r, z \rangle$
$q_1 = Az_1$
$\alpha_1 = \rho_1 / \langle z, q \rangle$
$x_1 = x_0 + \alpha_1 z_1$
$normx = \|x_1\|$
$r_1 = r_0 - \alpha_1 q_1$
$normr = \|r_1\|$
check convergence
for i = 2,...
$z_i = r_{i-1} M^{-1}$
$\rho_i = \langle r_{i-1}, z_i \rangle$
$\beta = \rho_i / \rho_{i-1}$
$p_i = z_i + \beta p_i$
$q_i = Ap_i$
$\alpha_i = \rho_i / \langle p_i, q_i \rangle$
$x_i = x_0 + \alpha_i z_i$
$normx = \|x_i\|$
$r_i = r_0 - \alpha_i q_i$
$normr = \|r_i\|$
check convergence
end

9n/7n=1.28 5.842 Gflops = 18 GB/s

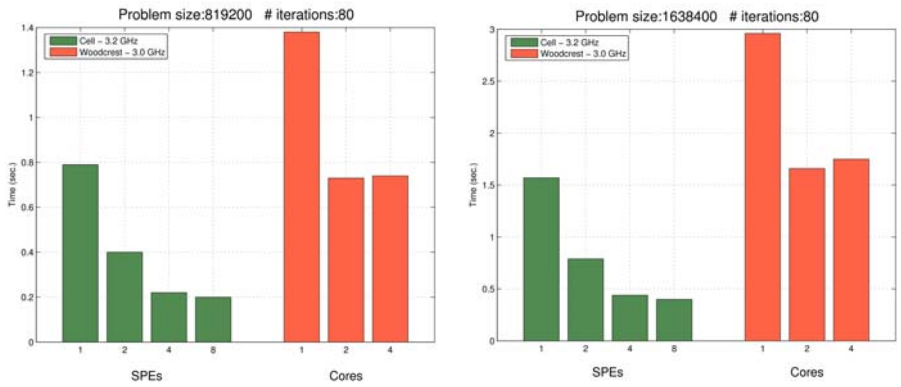
8n/4n=2.00 10.653 Gflops = 20 GB/s

4n/3n=1.33 7.004 Gflops = 21 GB/s

3n/3n=1.00 5.250 Gflops = 21 GB/s

2n/3n=0.66 3.503 Gflops = 21 GB/s

PCG on the Cell: results



Code on the Woodcrest (2 dual core) is blocked, unrolled, vectorized and OpenMP parallelized.

Lower is better

In LAPACK Today

- LAPACK 3.1.1 has a General Solver GE (DSGESV)
- For the next release:
  - GB: General Band Matrix
  - PB: Symmetric Positive Definite Band Matrix
  - PO: Symmetric Positive Definite Matrix (Full Storage)
  - SY: Symmetric Matrix (Full Storage)
- After Symmetric packed matrices:
  - PP: Symmetric Positive Definite Matrix (Packed Storage)
  - SP: Symmetric Matrix (Packed Storage)
- Probably not worth doing (*O(n)* ops for factor and solve)
  - GT: General Tridiagonal Matrix
  - PT: Symmetric Positive Definite Tridiagonal Matrix

43

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

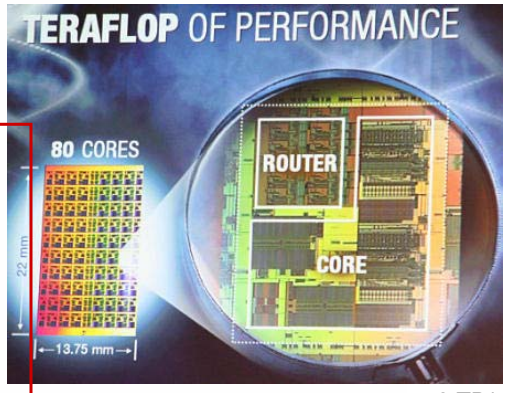
44

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.

1.2 TB/s memory BW

"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.

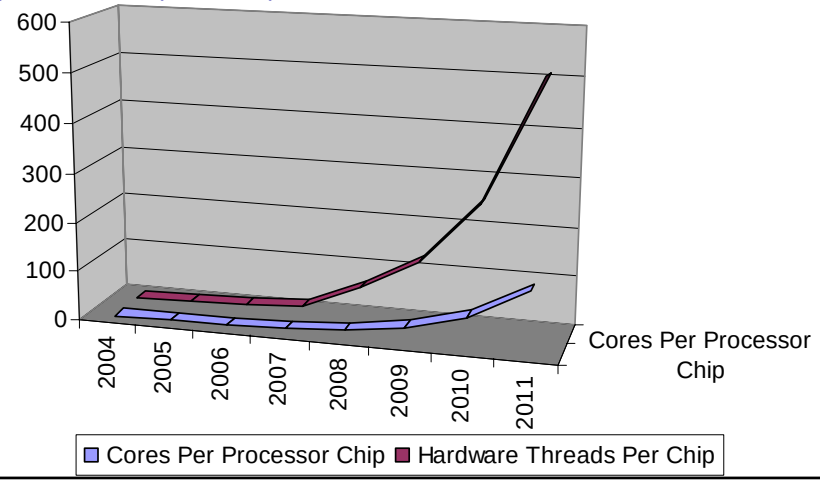
The TerraFLOP processor is required to power what Intel described as the mega data centre, delivering online applications. Intel touted **Google** and **Youtube** as examples of providers that will require this level of computing power. The chipmaker projected that by 2010 terra-scale servers will make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

45

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

CPU Desktop Trends 2004-2011

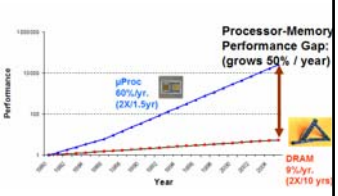
- **Relative processing power will continue to double every 18 months**
- **5 years from now: 128 cores/chip w/512 logical processes per chip**



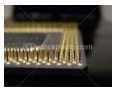
46

Challenges Resulting From Multicore

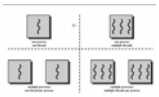
- Aggravated memory wall
  - Memory bandwidth
    - to get data out of memory banks
    - to get data into multi-core processors
  - Memory latency
  - Fragments L3 cache



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



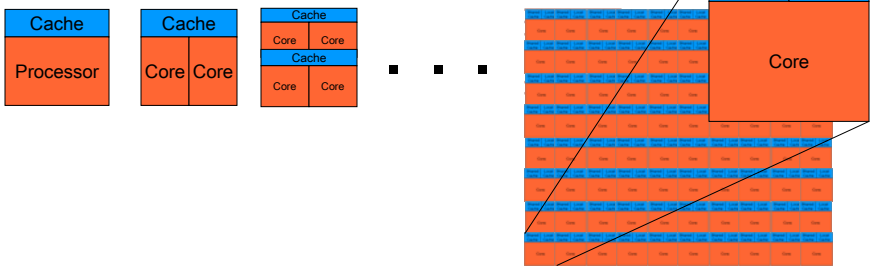
- Relies on effective exploitation of multiple-thread parallelism
  - Need for parallel computing model and parallel programming model



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

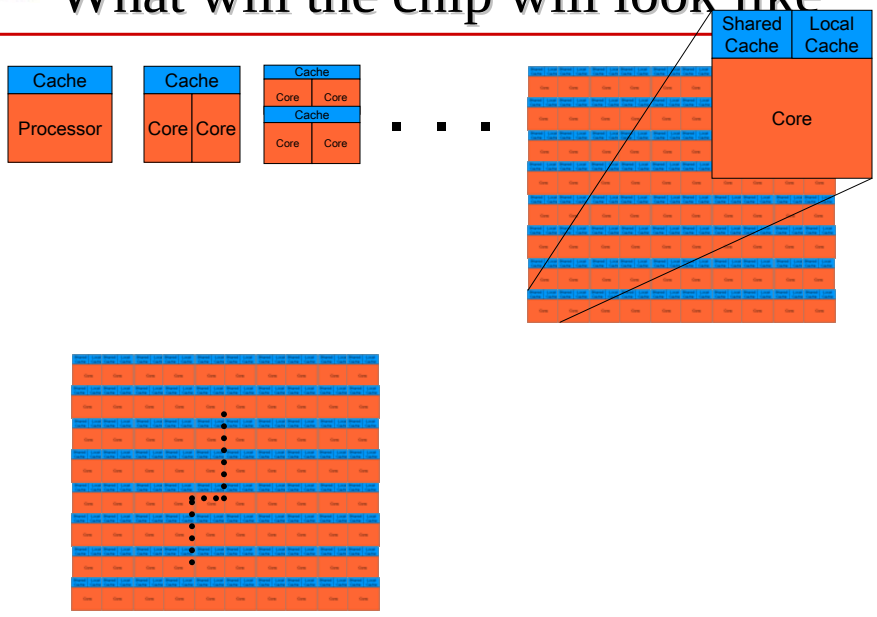
47

What will the chip will look like?



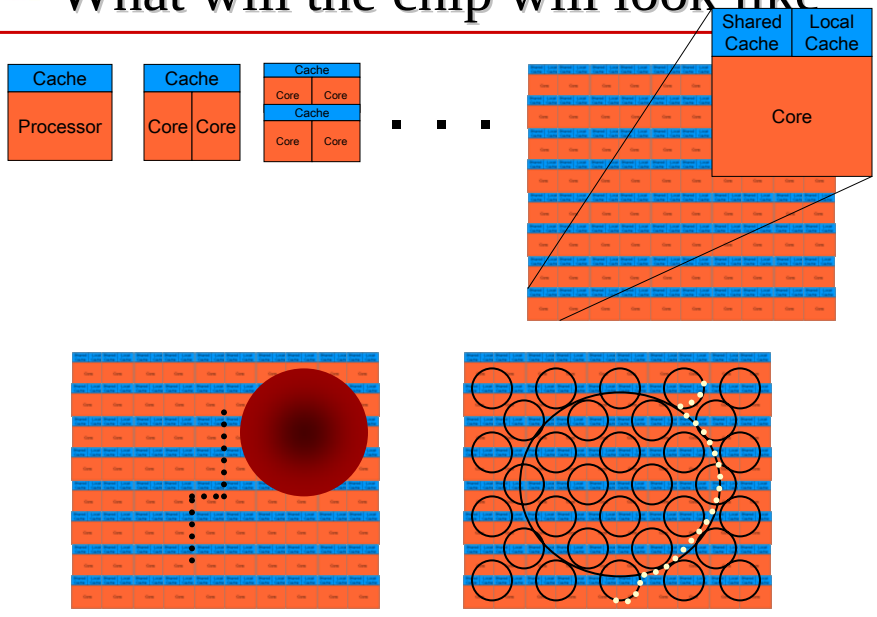
48

What will the chip will look like



49

What will the chip will look like



50

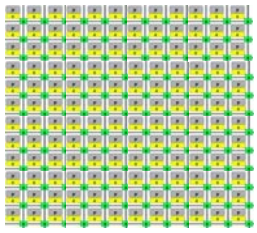
Major Changes to Software

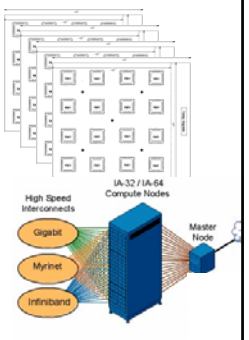
- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with cluster computing and message passing
  - Rethink and rewrite the applications, algorithms, and software
- Numerical libraries for example will change
  - For example, both LAPACK and ScaLAPACK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

51

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





Constantly Evolving - Hybrid Design

- More and more High Performance Computers will be built on a Hybrid Desing
- Cluster of Cluster systems
  - Multicore nodes in a cluster
- Nodes augmented with accelerators
  - ClearSpeed, GPUs, Cell
- Japanese 10 PFlop/s "Life Simulator"
  - Vector+Scalar+Grape:
    - Theoretical peak performance: >1-2 PetaFlops from Vector + Scalar System, ~10 PetaFlops from MD-GRAPE-like System
- LANL's Roadrunner
  - Multicore + specialized accelerator boards

53

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.

54

Collaborators / Support

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

Software and papers available: http://icl.cs.utk.edu/iter-ref/