Workshop on
Edge Computing Using New Commodity Architectures (EDGE)
May 23 - 24, 2006
Chapel Hill, North Carolina

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

Jack Dongarra
University of Tennessee
and
Oak Ridge National Laboratory

Overview

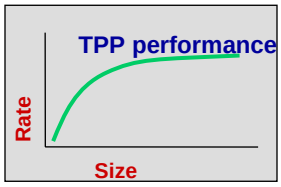
- Look at current state of high performance computing
  - Top500 data for Past and present
- Some of the changes Multicore brings
  - Look at the impact on numerical libraries
- Potential gains by exploiting lower precision devices
  - GPUs, Cell, SSE2, AltaVec

TOP500 supercomputer

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

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

  $Ax=b$, *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**

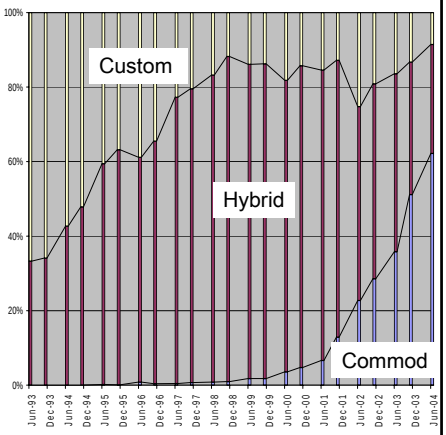
33 3

Current HPC Architecture/Systems

Tightly Coupled

- Custom processor with custom interconnect
  - Cray X1
  - NEC SX-8
  - IBM Regatta
  - IBM Blue Gene/L
- Commodity processor with custom interconnect
  - SGI Altix
    - Intel Itanium 2
  - Cray XT3, XD1
    - AMD Opteron
- Commodity processor with commodity interconnect
  - Clusters
    - Pentium, Itanium, Opteron, Alpha
    - GigE, Infiniband, Myrinet, Quadrics
  - NEC TX7
  - IBM eServer
  - Dawning

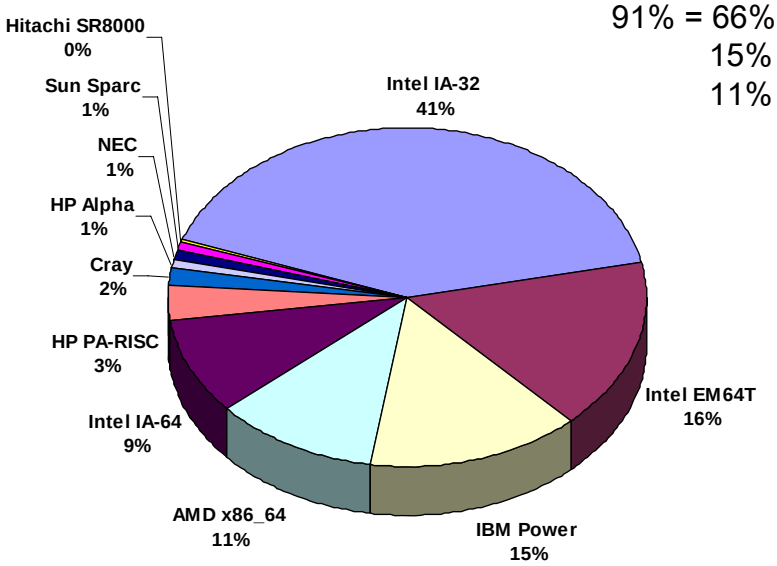
Loosely Coupled



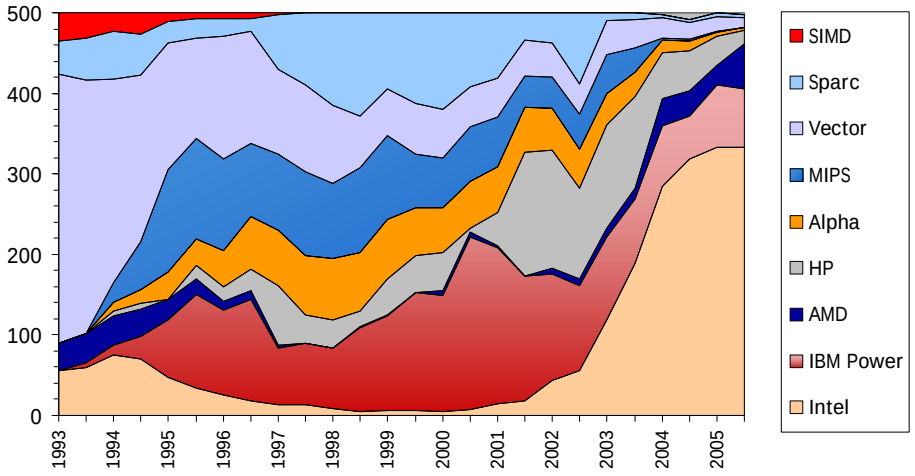
33 4

Processor Type Used in the Top500 Systems

91% = 66% Intel
15% IBM
11% AMD



Processor Types (Top500)

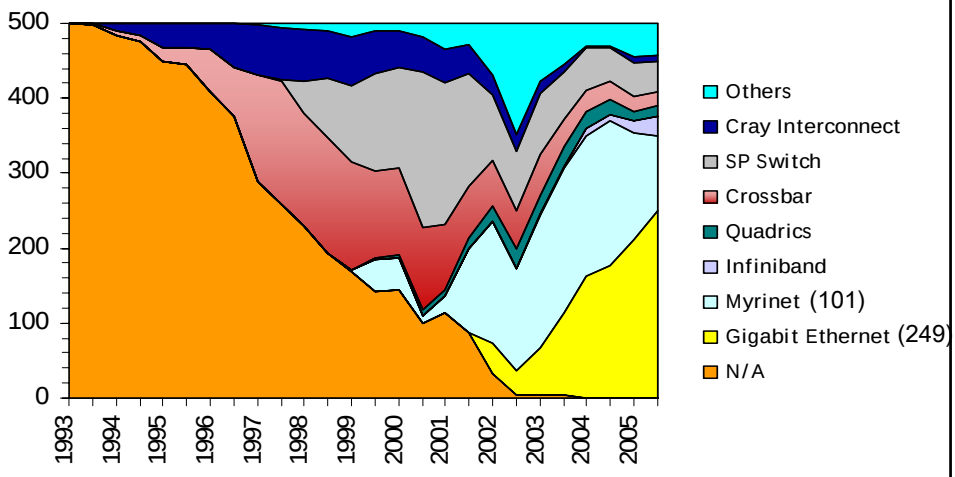


33

Intel + IBM Power PC + AMD = 91%

6

Interconnects / Systems (Top500)

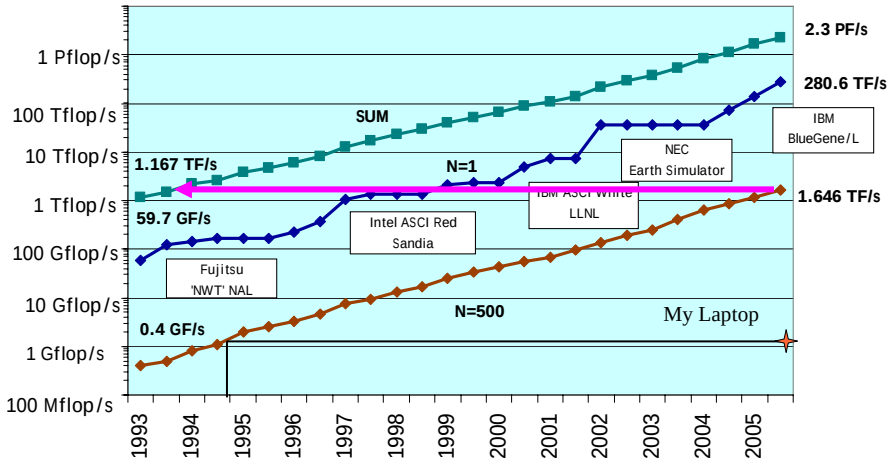


33

GigE + Myrinet = 70%

7

Performance Development (Top500)



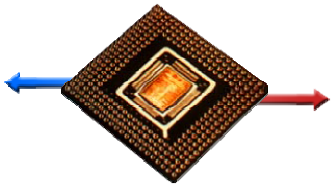
33

8

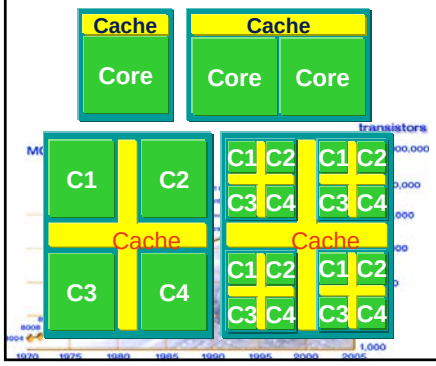
Increasing CPU Performance: A Delicate Balancing Act

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

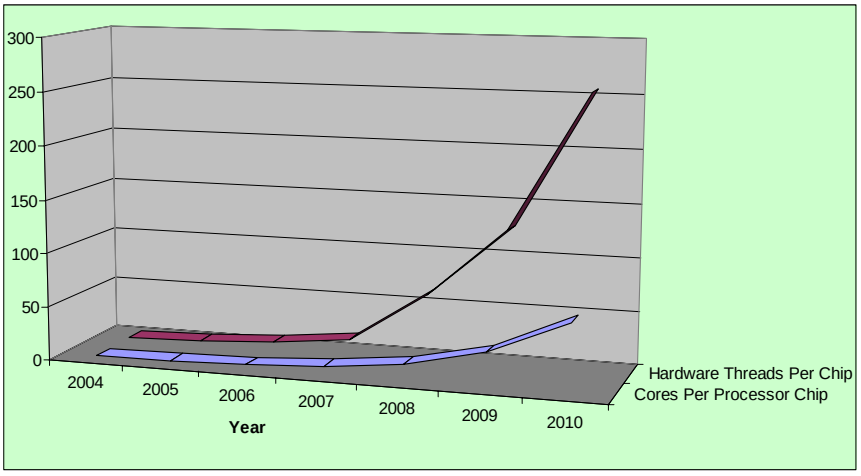
We will not see the dramatic increases in clock speeds in the future.

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

9

CPU Desktop Trends – Change is Coming

- Relative processing power will continue to double every 18 months
- 256 logical processors per chip in late 2010



10

Commodity Processor Trends

Bandwidth/Latency is the Critical Issue, not FLOPS

Got Bandwidth?

	Annual increase	Typical value in 2006
Single-chip floating-point performance	59%	4 GFLOP/s
Front-side bus bandwidth	23%	1 GWord/s = 0.25 word/flop
DRAM latency	(5.5%)	70 ns = 280 FP ops = 70 loads

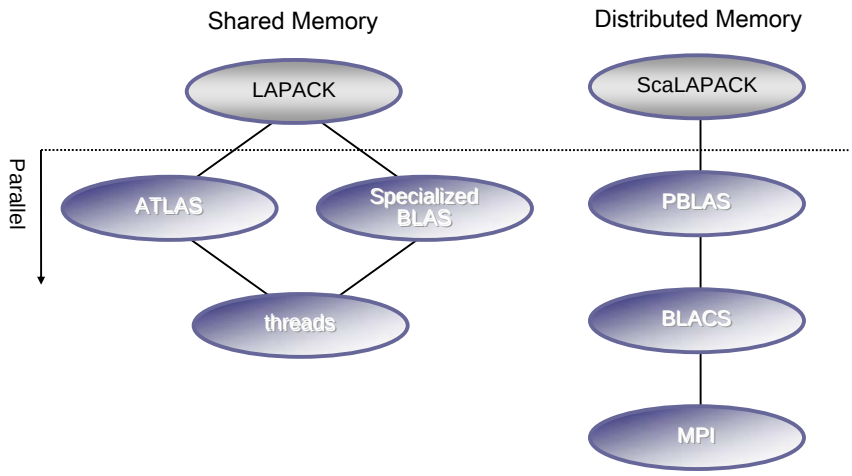
33 Source: *Getting Up to Speed: The Future of Supercomputing*, National Research Council, 222 pages, 2004, National Academies Press, Washington DC, ISBN 0-309-09502-6. 11

That Was the Good News

- **Bad news: the effect of the hardware change on the existing software base**
- **Must rethink the design of our software**
  - **Another disruptive technology**
  - **Rethink and rewrite the applications, algorithms, and software**

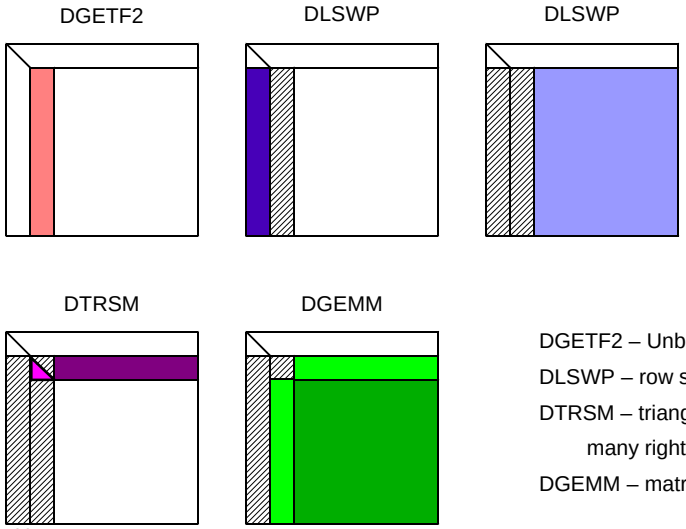
33 12

Parallelism in LAPACK / ScaLAPACK



33

Right-Looking LU factorization (LAPACK)



DGETF2 – Unblocked LU

DLSWP – row swaps

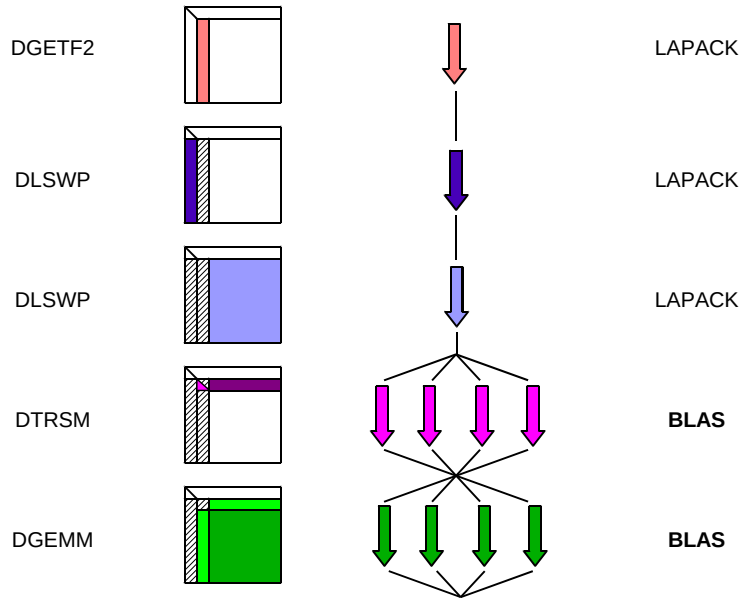
DTRSM – triangular solve with many right-hand sides

DGEMM – matrix-matrix multiply

33

14

Steps in the LAPACK LU



33

15

LU Timing Profile (4 processor system)

LAPACK + BLAS threads

Time for each component

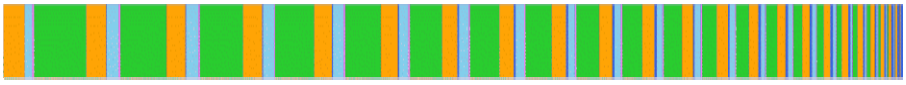
1D decomposition and SGI Origin

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

33

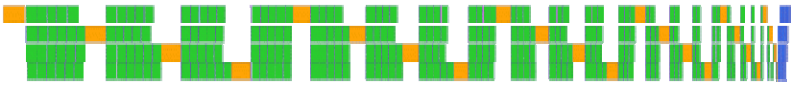
LU Timing Profile (4 processor system)

LAPACK + BLAS threads



Time for each component →

Threads – no lookahead



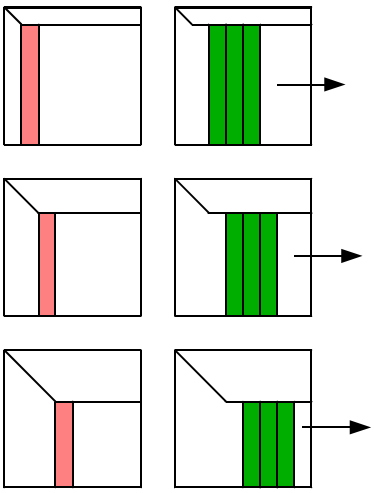
- DGETF2
- DLASWP(L)
- DLASWP(R)
- DTRSM
- DGEMM

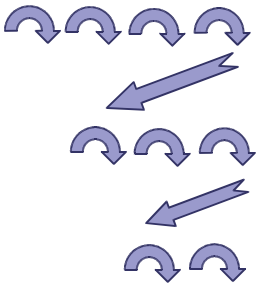
In this case the performance difference comes from parallelizing row exchanges (DLASWP) and threads in the LU algorithm.

1D decomposition and SGI Origin

33

Right-Looking LU Factorization

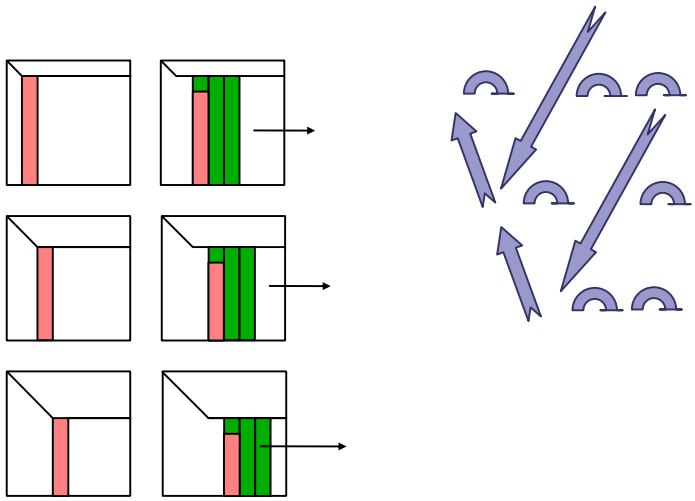




33

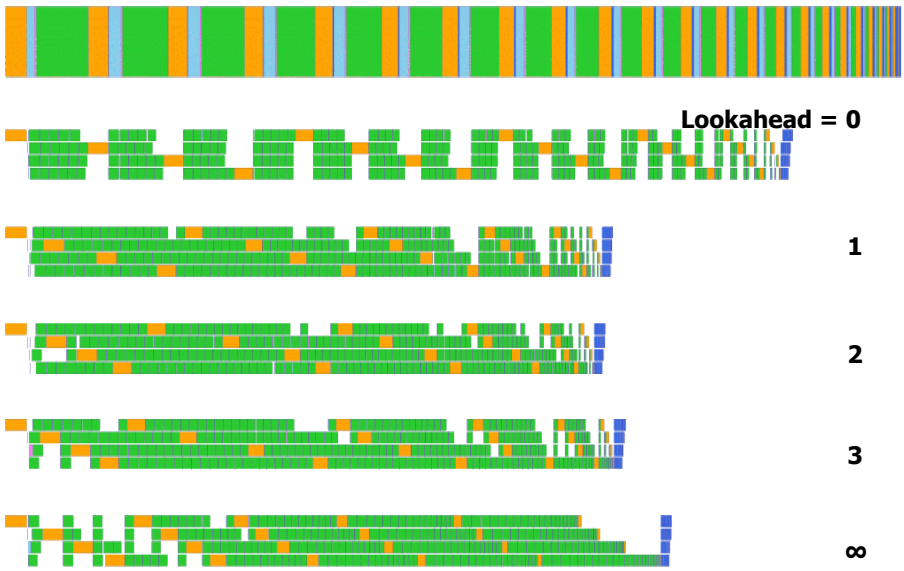
18

Right-Looking LU with a Lookahead



33

Pivot Rearrangement and Lookahead
4 Processor runs

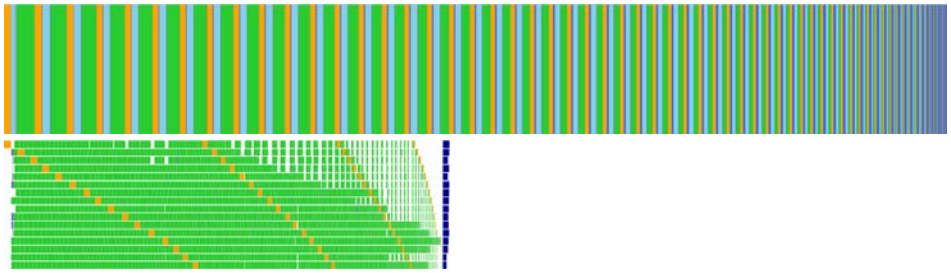


20

Fixed vs Adaptive Lookahead

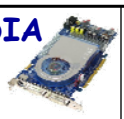
- No look-ahead or shallow look-ahead:
  - Not enough work in the update to the trailing matrix Pipeline stalls "bubbles" at the end of factorization.
- Deep or unlimited lookahead:
  - Attempt to factorization the next panel before the necessary piece of the trailing matrix is available,
  - Pipeline stalls "bubbles" at the beginning of the factorization.
- Solution - adaptive look-ahead:
  - Basically implement left-looking version of the algorithm,
  - Pursue the panels as fast a possible,
  - But continue updating the trailing matrix until sure that calling next panel does not stall.

Pivot Rearrangement and Adaptive Look-ahead (16 SMP runs)



33 22

GPU Performance

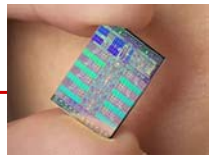
GPU Vendor	NVIDIA 	NVIDIA 	ATI 
Model	6800Ultra	7800GTX	X1900XTX
Release Year	2004	2005	2006
32-bit Performance	60 GFLOPS	200 GFLOPS	400 GFLOPS
64-bit Performance	must be emulated in software		

33

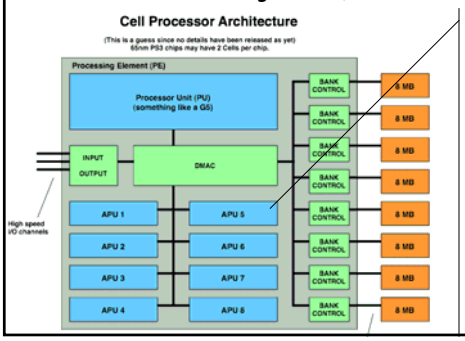
23

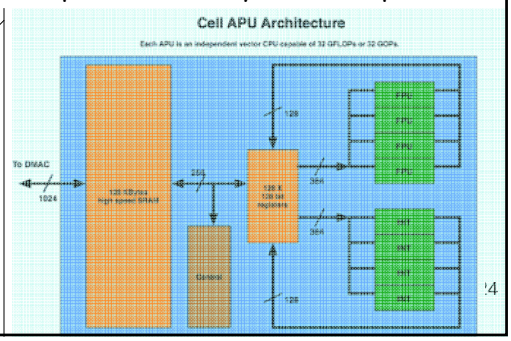
Thanks: Jeremy Meredith, ORNL

Things to Watch: 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 32 Gflop/s (8 Gflop/s each)
  - 256 Gflop/s peak! 32 bit floating point; 64 bit floating point at 25 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.





24

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.

33 25

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.

33 26

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.
  - We can show using this approach that 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; $O(10^8)$

33

Iterative Refinement – What's New?

- **Hasn't been used for speed improvement, only for accuracy improvement.**
- **Most of the theorems on mixed-precision iterative refinement are:**
  - "what is the SINGLE precision accuracy I can get with iterative refinement single/double?"
- **Our problem is :**
  - "what is the DOUBLE precision accuracy I can get using iterative refinement single/double?"

33 28

Additional Benefits

- If non-IEEE 32 bit arithmetic, but 64 bit is IEEE
  - If the floating point is not non-IEEE arithmetic for 32 bit computations and 64 bit computations does IEEE arithmetic, then accuracy should be as good as if IEEE was used.
- Possibility of correcting "errors" in the 32 bit computation.
  - Say a bit flips in the LU factorization and is undetected, then the process will self correct.

33 29

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.

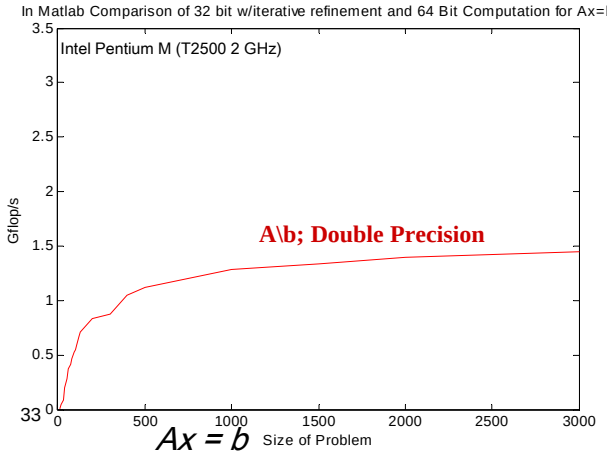
```
sa=single(a); sb=single(b);
[sl,su,sp]=lu(sa);                                          O(n^3)
sx=su\(sl\(sp*sb)); x=double(sx); r=b-a*x;                  O(n^2)
i=0;
while(norm(r)>res1),
    i=i+1;
    sr = single(r);
    sx1=su\(sl\(sp*sr)); x1=double(sx1); x=x1+x; r=b-a*x;   O(n^2)
if (i==30),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

33 30

Another Look at Iterative Refinement

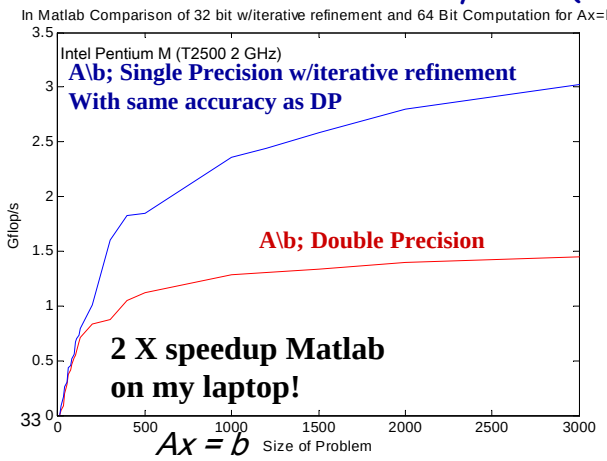
- On a Pentium; using SSE2, single precision can perform 4 floating point operations per cycle and in double precision 2 floating point operations per cycle.
- In addition there is reduced memory traffic (factor on sp data)



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)



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

33 33

Speedups for Ax = b (Ratio of Times)

Architecture (BLAS)	*n*	DGEMM /SGEMM	DP Solve /SP Solve	DP Solve /Iter Ref	# iter
Intel Pentium IV-M Northwood (Goto)	4000	2.02	1.98	1.54	5
Intel Pentium III Katmai (Goto)	3000	2.12	2.11	1.79	4
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

33 34

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	17.81	0.44	40.4
500	34.71	0.69	49.7
600	60.11	1.01	59.0
700	94.95	1.38	68.7
800	141.75	1.83	77.3
900	201.81	2.33	86.3
1000	276.94	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

33 35

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

33 See webpage for tech report which discusses this. 36

Constantly Evolving - Hybrid Design

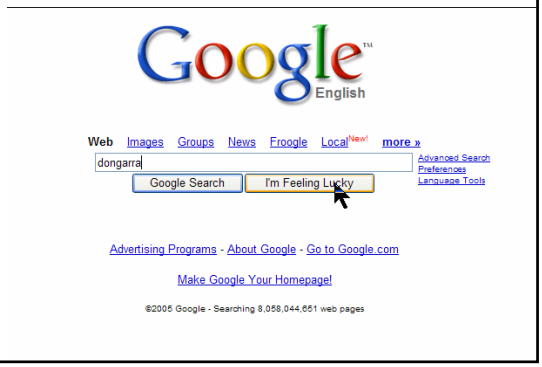
- Cluster of Cluster systems
  - Multicore nodes in a cluster
- Nodes augmented with accelerators
  - ClearSpeed, GPUs, Cell
- Japanese 10 PFlop/s "Life Simulator"
  - Vector+Scalar+Grape:
    - Theoretical peak performance: >1-2 PetaFlops from Vector + Scalar System, ~10 PetaFlops from MD-GRAPE-like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

Summary of Current Unmet Needs

- Performance / Portability
- Fault tolerance
- Memory bandwidth/Latency
- Adaptability: Some degree of autonomy to self optimize, test, or monitor.
  - Able to change mode of operation: static or dynamic
- Better programming models
  - Global shared address space
  - Visible locality
- Maybe coming soon (incremental, yet offering real benefits):
  - Global Address Space (GAS) languages: UPC, Co-Array Fortran, Titanium, X10, Chapel, Fortress)
    - "Minor" extensions to existing languages
    - More convenient than MPI
    - Have performance transparency via explicit remote memory references
- What's needed is a long-term, balanced investment in hardware, software, algorithms and applications.

Collaborators / Support

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



33