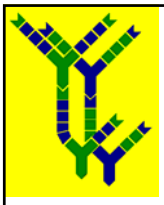


INTERNATIONAL ADVANCED RESEARCH WORKSHOP
ON
HIGH PERFORMANCE COMPUTING AND GRIDS
Cetraro (Italy), July 3-6, 2006

The Challenges of Multicore and Specialized Accelerators

Jack Dongarra
University of Tennessee
and
Oak Ridge National Laboratory

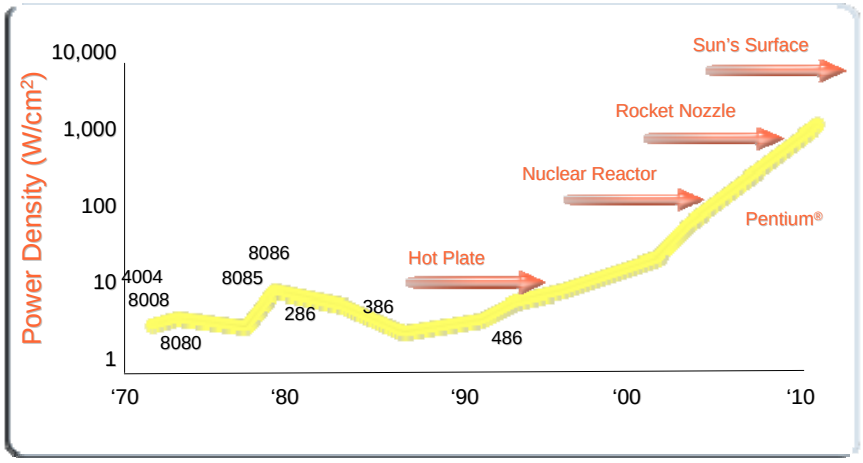
Overview

- **Some of the changes Multicore brings**
  - **Look at the impact on numerical libraries**
- **Potential gains by exploiting lower precision devices**
  - **IBM Cell, SSE2, GPUs, AltaVec**

Today's CPU Architecture:

Heat becoming an unmanageable problem

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



Intel Developer Forum, Spring 2004 - Pat Gelsinger
(Pentium at 90 W)

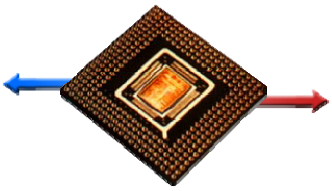
Cube relationship between the cycle time and power.3

Increasing CPU Performance:

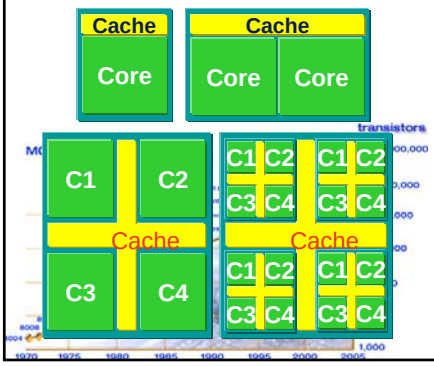
A Delicate Balancing Act

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

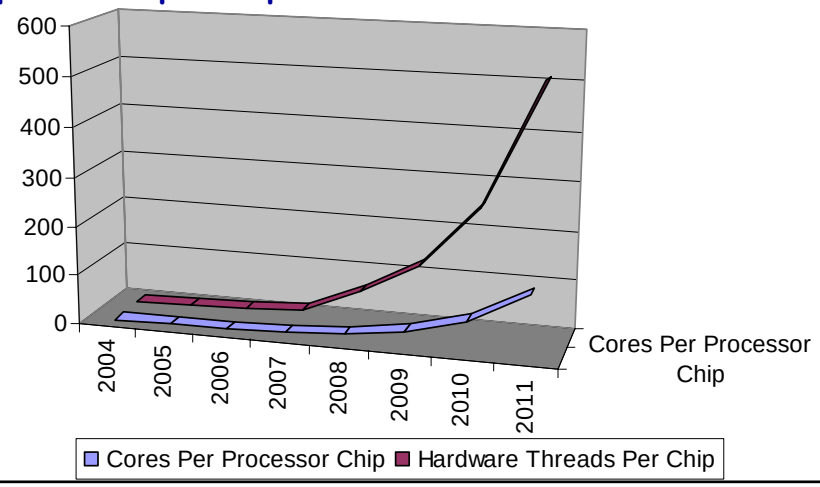
We will not see the dramatic increases in clock speeds in the future.

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

4

CPU Desktop Trends 2004-2011

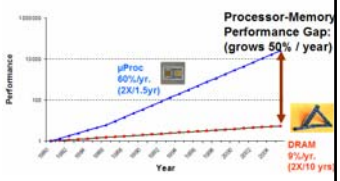
- Relative processing power will continue to double every 18 months
- 5 years from now: 128 cores/chip w/512 logical processes per chip



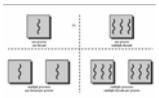
5

Challenges Resulting From Multicore

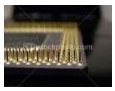
- Aggravates memory wall
  - Memory bandwidth
    - to get data out of memory banks
    - to get data into multi-core processors
  - Memory latency
  - Fragments L3 cache



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



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



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

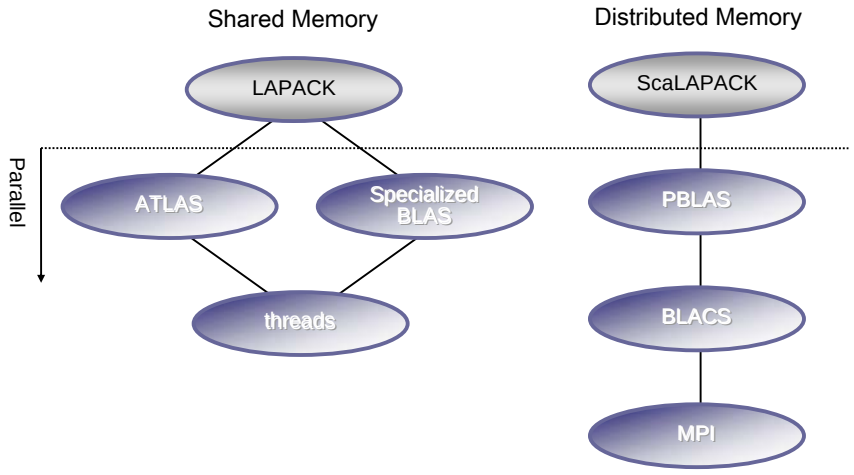
6

Major Changes to Software

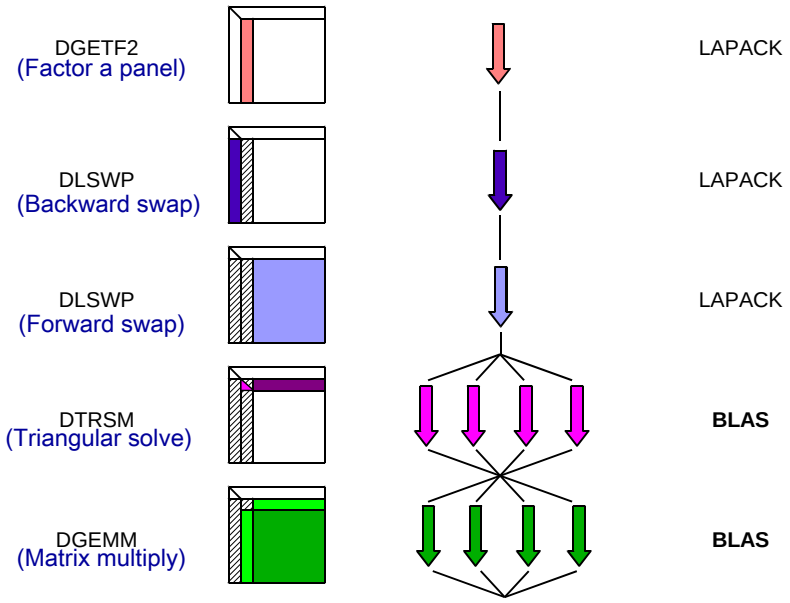
- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with message passing
  - Rethink and rewrite the applications, algorithms, and software
- Numerical libraries for example will change
  - For example, both LAPACK and ScaLAPACK will undergo major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

7

Parallelism in LAPACK / ScaLAPACK



Steps in the LAPACK LU



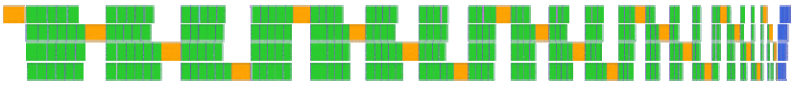
9

LU Timing Profile (4 processor system)

LAPACK + BLAS threads

Time for each component

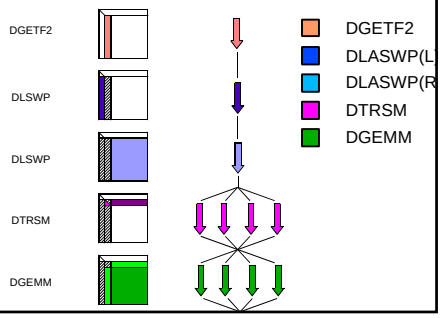
Threads – no lookahead



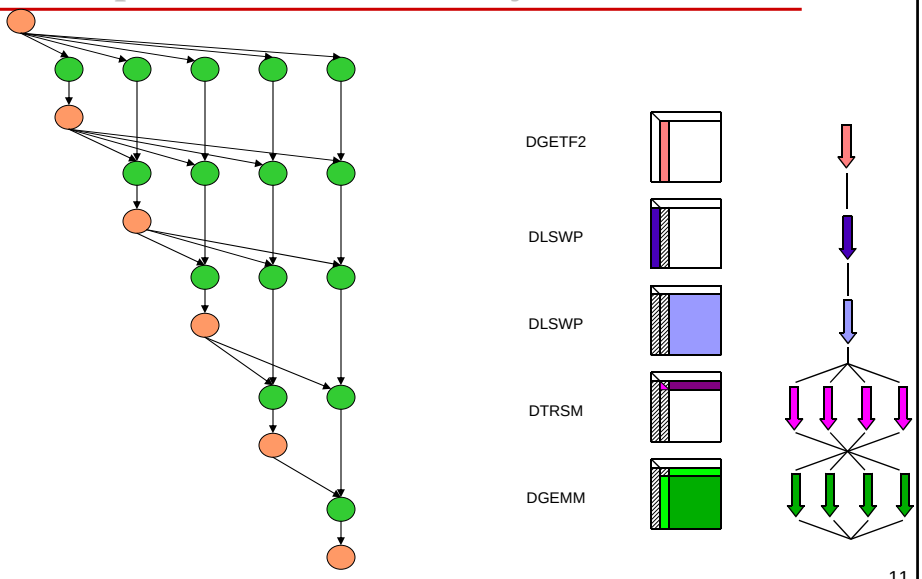
In this case the performance difference comes from

parallelizing row exchanges (DLASWP) and threads in the LU algorithm.

1D decomposition and SGI Origin

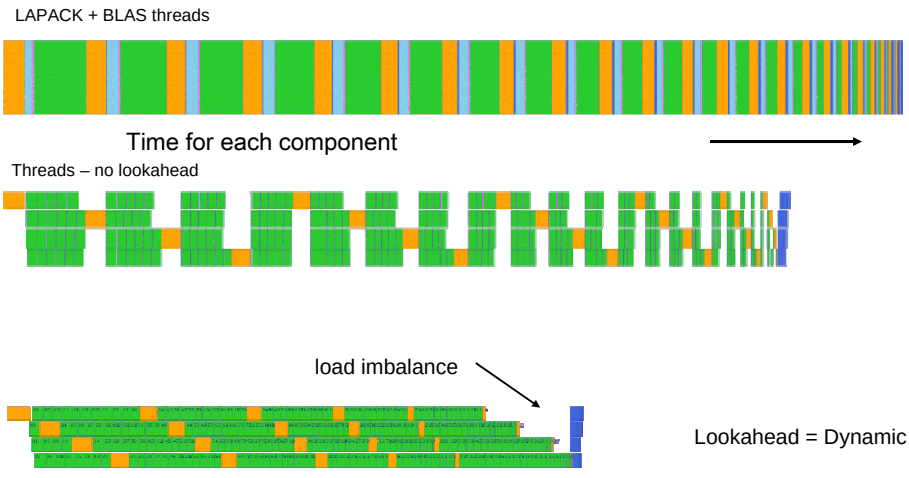


Adaptive Lookahead - Dynamic



11

Dynamic Lookahead

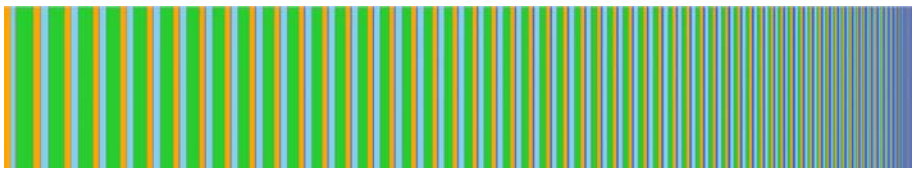


12

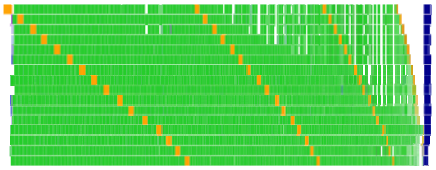
LU - BLAS Threads vs. Dynamic Lookahead

SGI Origin 3000 / 16 MIPS R14000 500 Mhz

BLAS Threads (LAPACK)



Dynamic Lookahead



Problem Size N = 4000

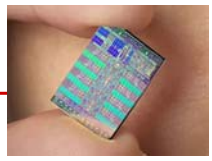
13

Pipelined Cholesky Factorization

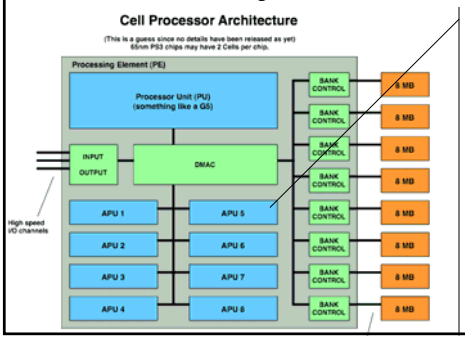
- Check dependencies,
- **Perform task**,
- Update progress,
- Make transition.

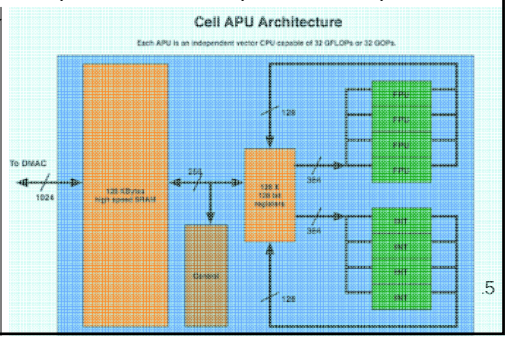
```
while (1)
{
    switch (task)
    {
        case DSYRK:
        {
            check_dependencies();
            dsyrk();
            update_progress();
            make_transition();
        }

        case DPOTF2:
        {
            check_dependencies();
            dpotf2();
            update_progress();
            make_transition();
        }

        case DGEMM:
        {
            check_dependencies();
            dgemm();
            update_progress();
            make_transition();
        }

        case DTRSM:
        {
            check_dependencies();
            dtrsm();
            update_progress();
            make_transition();
        }
    }
}
```

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.





.5

GPU Performance

Fastest Intel processor: Woodcrest is at 12 Gflop/s peak (64 bit)

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		

16

Thanks: Jeremy Meredith, ORNL

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.

17

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.

18

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; $O(10^8)$

Additional Benefits

- **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.
- **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.

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.

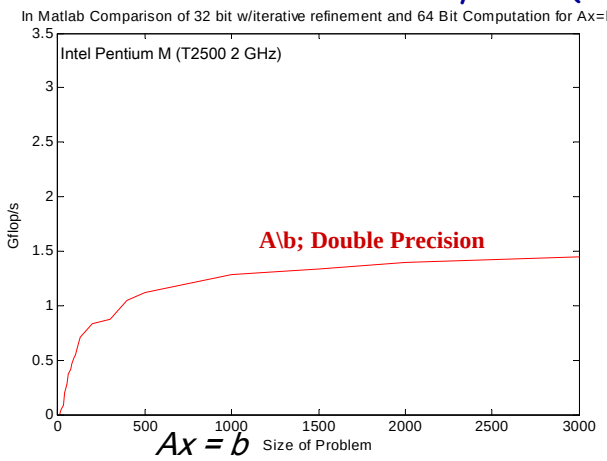
```
sa=single(a); sb=single(b);
[sl,su,sp]=lu(sa);                                   Most of the work: O(n^3)
sx=su\(sl\(sp*sb)); x=double(sx); r=b-a*x;                          O(n^2)
i=0;
while(norm(r)>res1),
    i=i+1;
    sr = single(r);
    sx1=su\(sl\(sp*sr)); x1=double(sx1); x=x1+x; r=b-a*x; 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

21

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)

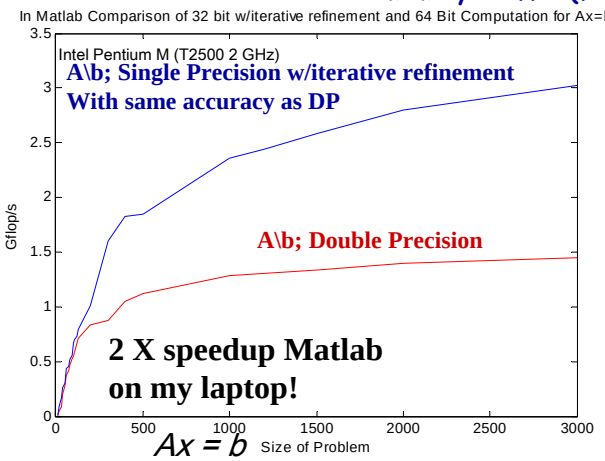


1.4 GFlop/s!

22

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!!

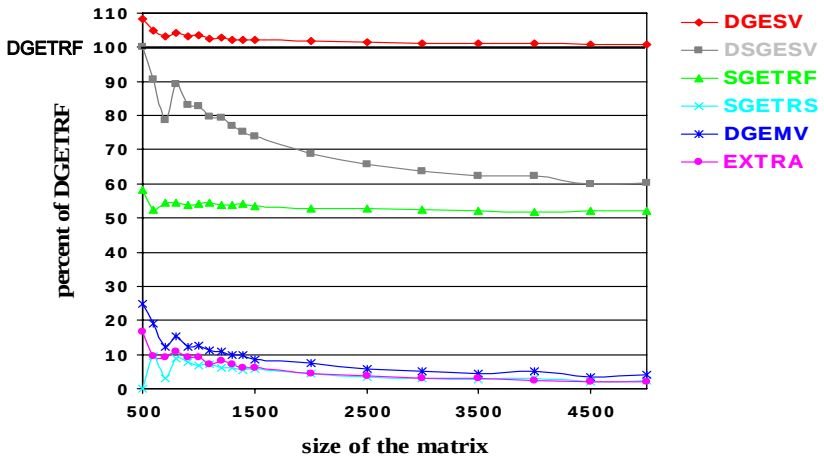
On the Way to Understanding How to Use the Cell Something Else Happened …

- 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

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



25

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

26

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

27

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.

28

Collaborators / Support

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