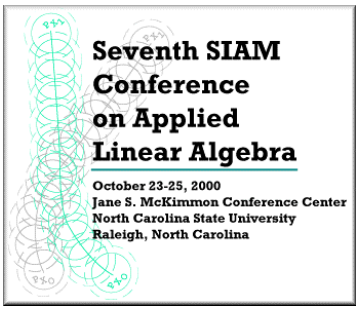
The Impact Of Computer Architectures On Linear Algebra Algorithms



Jack Dongarra
Innovative Computing Laboratory
University of Tennessee

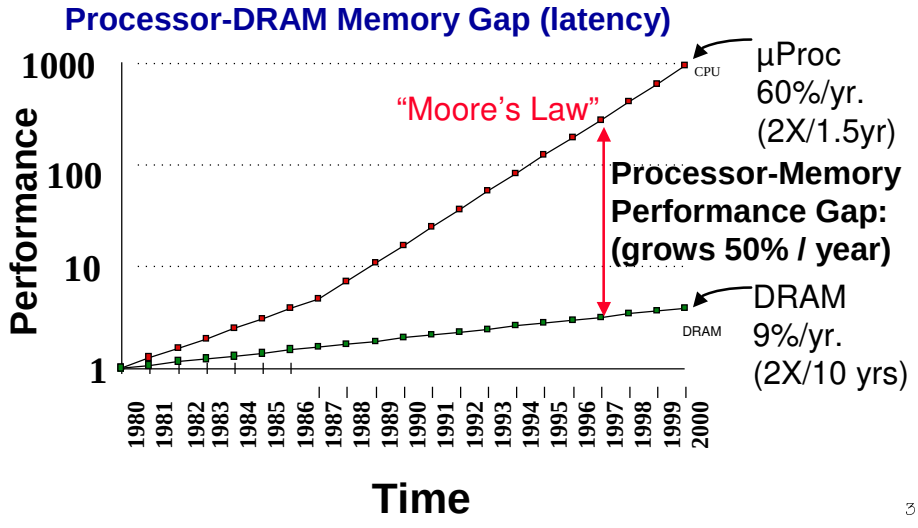
http://www.cs.utk.edu/~dongarra/

Overview

- Performance of some kernels
  - ATLAS
  - Recursive decomposition
    - Direct
    - Sparse
- Tools for monitoring performance
  - PAPI

2

Where Does the Performance Go? or Why Should I Care About the Memory Hierarchy?



3

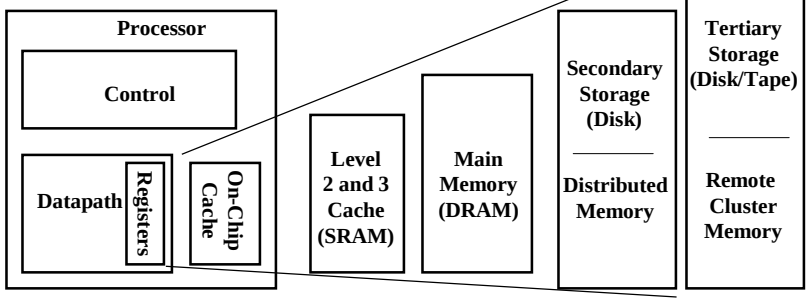
Optimizing Computation and Memory Use

- **Computational optimizations**
  - Theoretical peak:(# fpus)*(flops/cycle) * Mhz
    - PIII: (1 fpu)*(1 flop/cycle)*(650 Mhz) = 650 MFLOP/s
    - Athlon: (2 fpu)*(1flop/cycle)*(600 Mhz) = 1200 MFLOP/s
    - Power3: (2 fpu)*(2 flops/cycle)*(375 Mhz) = 1500 MFLOP/s
- **Operations like:**
  - Dot: 2 operands (16 Bytes) needed for 2 flops
  - Axpy: 3 operands (24 Bytes) needed for 2 flops
- **Memory optimization**
  - Theoretical peak: (bus width) * (bus speed)
    - PIII : (32 bits)*(133 Mhz) = 532 MB/s = 66.5 MW/s
    - Athlon: (64 bits)*(133 Mhz) = 1064 MB/s = 133 MW/s
    - Power3: (128 bits)*(100 Mhz) = 1600 MB/s = 200 MW/s

4

Memory Hierarchy

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.



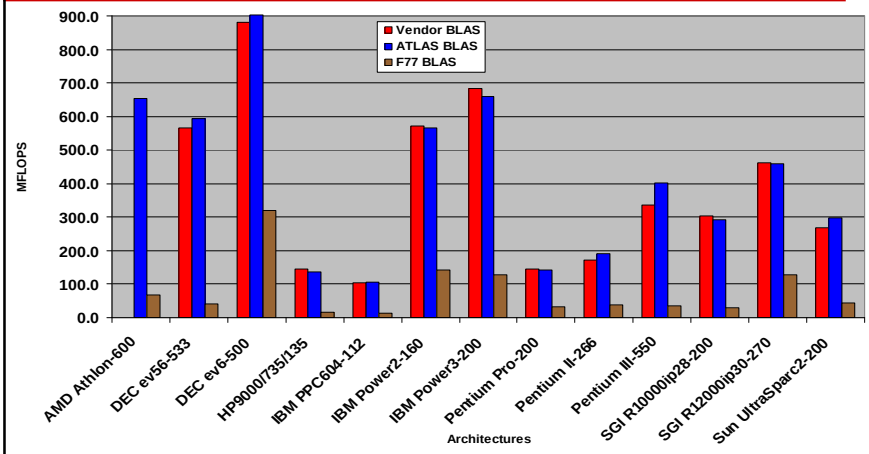
Speed (ns):	1s	10s	100s	10,000,000s (10s ms)	10,000,000,000s (10s sec)
Size (bytes):	100s	Ks	Ms	100,000 s (.1s ms)	10,000,000 s (10s ms)
				Gs	Ts

How To Get Performance From Commodity Processors?

- Today's processors can achieve high-performance, but this requires extensive machine-specific hand tuning.
- Hardware and software have a large design space w/many parameters
  - Blocking sizes, loop nesting permutations, loop unrolling depths, software pipelining strategies, register allocations, and instruction schedules.
  - Complicated interactions with the increasingly sophisticated micro-architectures of new microprocessors.
- Until recently, no tuned BLAS for Pentium for Linux.
- Need for quick/dynamic deployment of optimized routines.
- ATLAS - Automatic Tuned Linear Algebra Software
  - PhiPac from Berkeley
  - FFTW from MIT (http://www.fftw.org)

6

ATLAS (DGEMM n = 500)



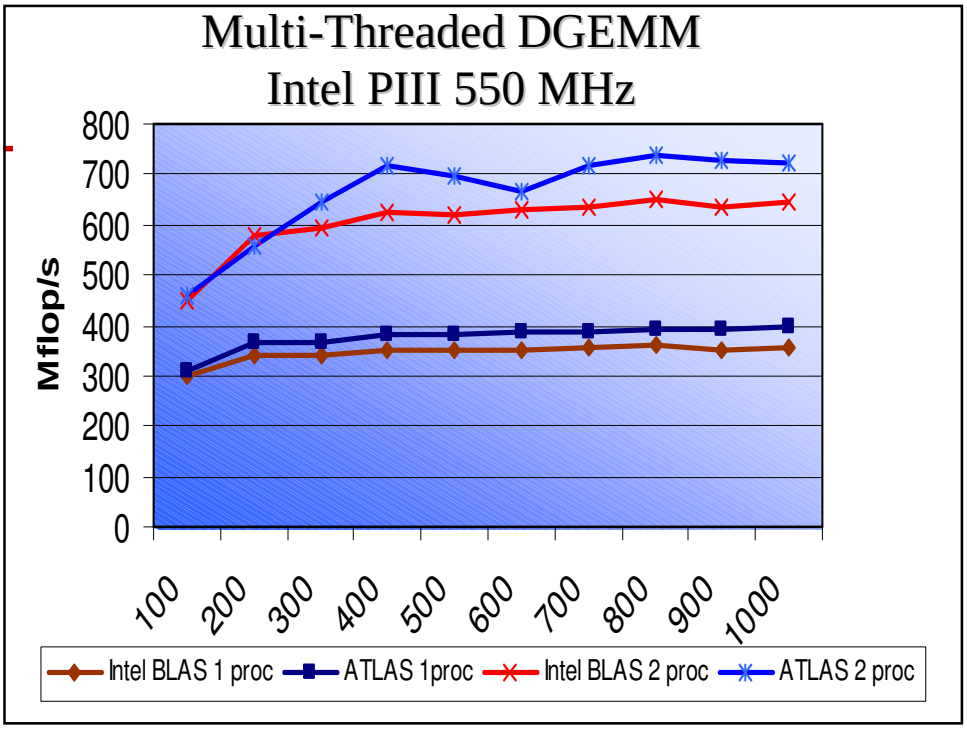
- ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.

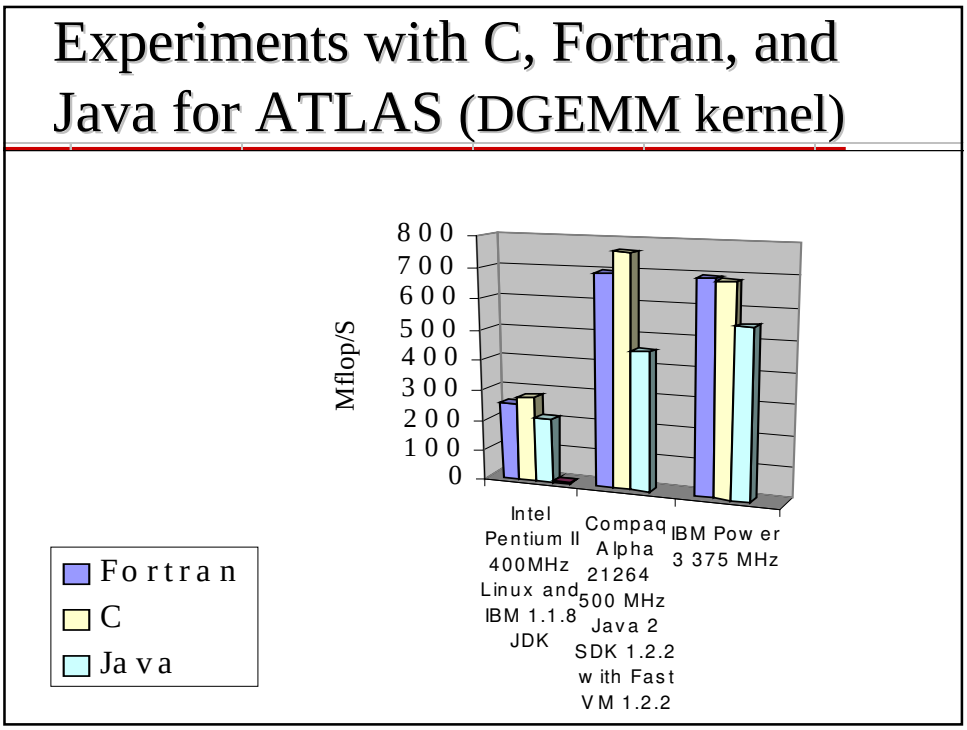
7

Why ATLAS is needed

- Operations like the BLAS require many man-hours / platform
  - Only done if financial incentive is there
    - Many platforms will never have an optimal version
  - Lags behind hardware
  - May not be affordable by everyone
  - Improves vendor code
- Operations may be important, but not general enough for standard
- Allows for portably optimal codes
- New model of high performance programming in which performance critical code is machine generated using parameter optimization.
- Done once to build the library, then used on that machine.

8


Multi-Threaded DGEMM
Intel PIII 550 MHz
Mflop/s
800
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000
Intel BLAS 1 proc
ATLAS 1proc
Intel BLAS 2 proc
ATLAS 2 proc


Experiments with C, Fortran, and Java for ATLAS (DGEMM kernel)
Mflop/S
800
700
600
500
400
300
200
100
0
Intel Pentium II 400MHz Linux and IBM 1.1.8 JDK
Compaq Alpha 21264 500 MHz Java 2 SDK 1.2.2 with Fast VM 1.2.2
IBM Power 3 375 MHz
Fortran
C
Java

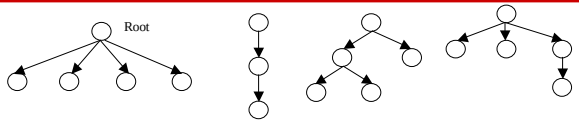
Extend These Ideas

- **Communication libraries**
  - **Optimize for the specifics of one's configuration.**
- **Algorithm layout and implementation**
  - **Look at the different ways to express implementation**

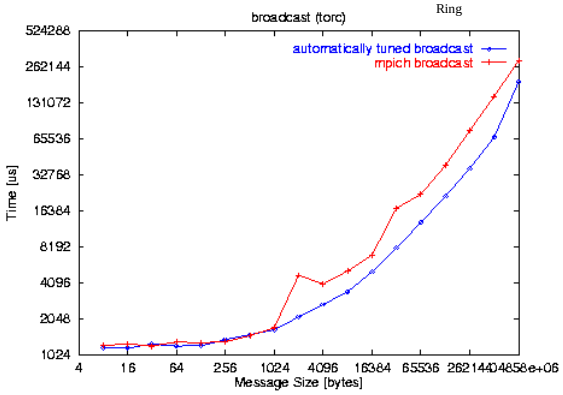
11

Work in Progress: ATLAS Approach Applied to Broadcast

(PII 8 Way Cluster with 100 Mb/s switched network)



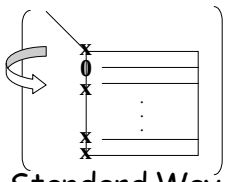
Sequential Ring Binary Binomial



Message Size (bytes)	Optimal algorithm	Buffer Size (bytes)
8	binomial	8
16	binomial	16
32	binary	32
64	binomial	64
128	binomial	128
256	binomial	256
512	binomial	512
1K	sequential	1K
2K	binary	2K
4K	binary	2K
8K	binary	2K
16K	binary	4K
32K	binary	4K
64K	ring	4K
128K	ring	4K
256K	ring	4K
512K	ring	4K
1M	binary	4K

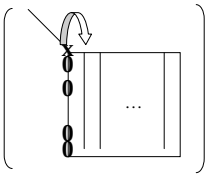
12

Gaussian Elimination



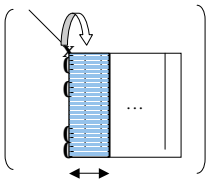
Standard Way

subtract a multiple of a row



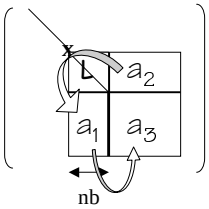
LINPACK

apply sequence to a column



apply sequence to nb

LAPACK



$a_2 = L^{-1} a_2$

$a_3 = a_3 - a_1 * a_2$

then apply nb to rest of matrix

13

Gaussian Elimination via a Recursive Algorithm

F. Gustavson and S. Toledo

LU Algorithm:

1: Split matrix into two rectangles (m x n/2)
if only 1 column, scale by reciprocal of pivot & return

2: Apply LU Algorithm to the left part

3: Apply transformations to right part
(triangular solve $A_{12} = L^{-1} A_{12}$ and
matrix multiplication $A_{22} = A_{22} - A_{21} * A_{12}$)

4: Apply LU Algorithm to right part

Most of the work in the matrix multiply

Matrices of size n/2, n/4, n/8, …

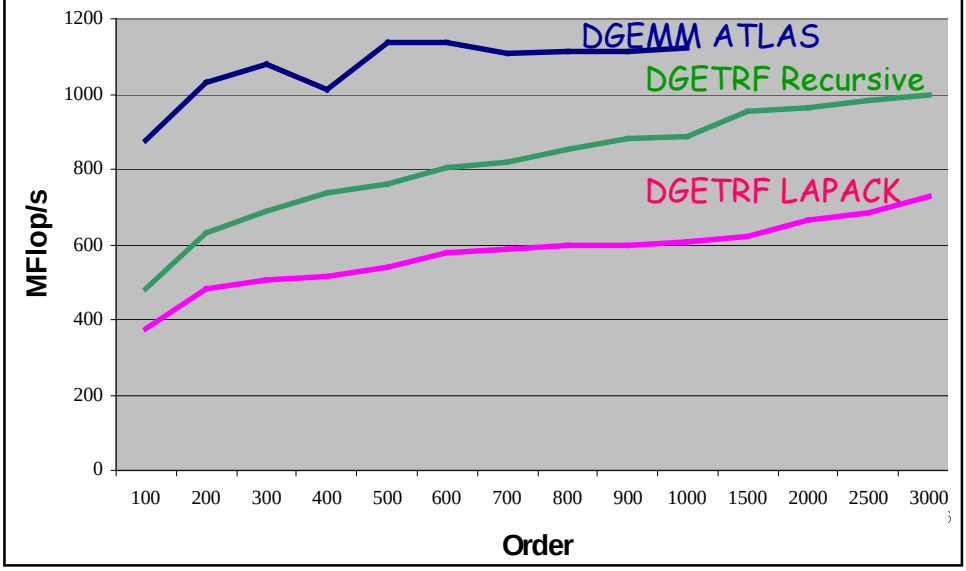
14

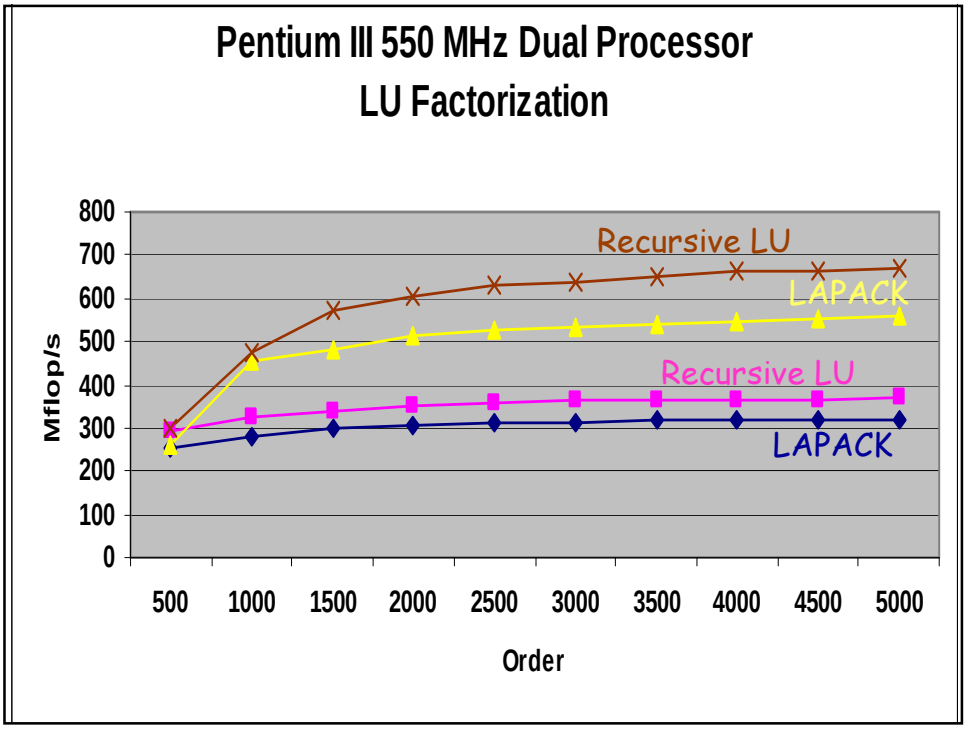
Recursive Factorizations

- Where Crout, Left and Right Looking meet
- Just as Accurate as conventional method
- Automatic Variable Blocking
  - Level 1 and 3 BLAS only !
- Extreme Clarity and Simplicity of Expression
- Highly Efficient
- The recursive formulation is just a rearrangement of the point-wise LINPACK algorithm
- The standard error analysis applies (assuming the matrix operations are computed the "conventional" way).

15

AMD Athlon 1 GHz Processor (~ $1100 System)





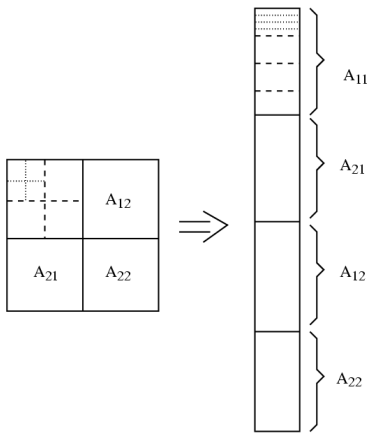
Recursive Factorization Applied to Sparse Direct Methods

Victor Eijkhout, Piotr Luszczek & JD

Also see poster at 6:30 in Rm 1D JD and Raghavan on Sparse Cholesky Factorization

1. Symbolic Factorization
2. Search for blocks that contain non-zeros
3. Conversion to sparse recursive storage
4. Search for embedded blocks
5. Numerical factorization

Layout of sparse recursive matrix



18

Dense recursive factorization

- **The algorithm:**

function rlu(A)

begin

rlu(A_{11});	recursive call
$A_{21} \leftarrow A_{21} \bullet U^{-1}(A_{11})$;	xTRSM() on upper triangular submatrix
$A_{12} \leftarrow L_1^{-1}(A_{11}) \bullet A_{12}$;	xTRSM() on lower triangular submatrix
$A_{22} \leftarrow A_{22} - A_{21} \bullet A_{12}$;	xGEMM()
rlu(A_{22});	recursive call

end.

- **Replace xTRSM and xGEMM with sparse implementations that are themselves recursive**

19

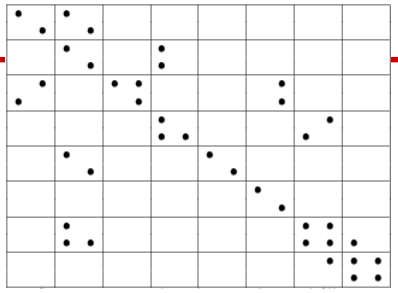
Sparse Recursive Factorization Algorithm

- **Solutions - continued**
  - **fast sparse xGEMM() is two-level algorithm**
    - **recursive operation on sparse data structures**
    - **dense xGEMM() call when recursion reaches single block**
- **Fill reducing ordering can be applied before recursive algorithm**
- **Uses Reverse Cuthill-McKee ordering**
- **No partial pivoting**
  - **use iterative improvement or**
  - **pivot only within blocks**

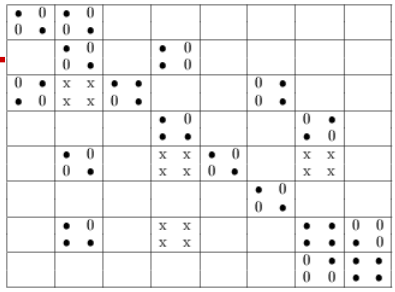
20

Recursive storage conversion steps

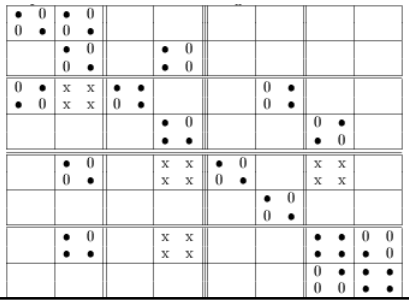
Matrix divided into 2x2 blocks



Matrix with explicit 0's and fill-in



Recursive algorithm division lines



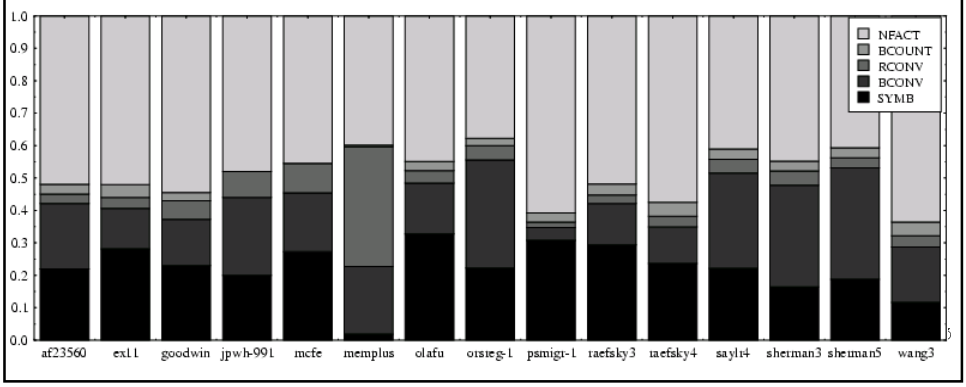
- • - original nonzero value
- 0 - zero value introduced due to blocking
- x - zero value introduced due to fill-in

21

Architecture: Pentium III						
Matrix name	order	nonzeros	SuperLU		LUSR	
			time [s]	FERR	time [s]	FERR
af23560	23560	460598	44.19	5.8e-14	31.34	1.8e-14
ex11	16614	109694	109.67	2.5e-05	55.3	1.3e-06
goodwin	7320	324772	6.49	1.2e-08	6.74	4.6e-06
jpwh_991	991	6027	0.19	2.9e-15	0.25	2.6e-15
mcfe	765	24382	0.07	1.2e-13	0.22	9.1e-13
memplus	17758	126150	0.29	2.1e-12	12.67	6.6e-13
olafu	16146	1015156	26.16	1.1e-06	22.1	3.7e-09
orsreg_1	2205	14133	0.46	1.3e-13	0.45	2.1e-13
psmigr_1	3140	543162	110.79	7.9e-11	88.61	1.2e-05
raefsky3	21200	1488768	62.07	1.4e-09	69.67	4.4e-13
raefsky4	19779	1316789	82.45	2.3e-06	104.29	3.5e-06
saylr4	3564	22316	0.85	3.2e-11	0.95	1.2e-11
sherman3	5005	20033	0.61	6.0e-13	0.67	4.8e-13
sherman5	3312	20793	0.28	1.4e-13	0.32	6.2e-15
wang3	26064	177168	84.14	2.4e-14	79.18	1.6e-14

Where the Time Goes

1. Symbolic Factorization (SYMB)
2. Search for blocks that contain non-zeros (BCONV)
3. Conversion to sparse recursive storage (RCONV)
4. Search for embedded blocks (BCOUNT)
5. Numerical factorization (NFACT)



What's Next?

- Improve the performance on the worst cases
- Look at optimization
- Develop and assess parallel implementation
  - SMP
  - Distributed memory
- Enhance embedded blocking concept
- Speed up symbolic factorization with use of embedded blocking
- Increase the set of test matrices
- Paper with the details on web page

24

Tools for Performance Evaluation

- Timing and performance evaluation has been an art
  - Resolution of the clock
  - Issues about cache effects
  - Different systems
- Situation about to change
  - Today's processors have internal counters

25

Performance Counters

- Almost all high performance processors include hardware performance counters.
- Some are easy to access, others not available to users.
- On most platforms the APIs, if they exist, are not appropriate for a common user, functional or well documented.
- Existing performance counter APIs
  - Cray T3E
  - SGI MIPS R10000
  - IBM Power series
  - DEC Alpha pfm pseudo-device interface
  - Windows 95, NT and Linux

26

Performance Data That May Be Available

- Cycle count
- Floating point instruction count
- Integer instruction count
- Instruction count
- Load/store count
- Branch taken / not taken count
- Branch mispredictions
- Pipeline stalls due to memory subsystem
- Pipeline stalls due to resource conflicts
- I/D cache misses for different levels
- Cache invalidations
- TLB misses
- TLB invalidations

27

Implementation

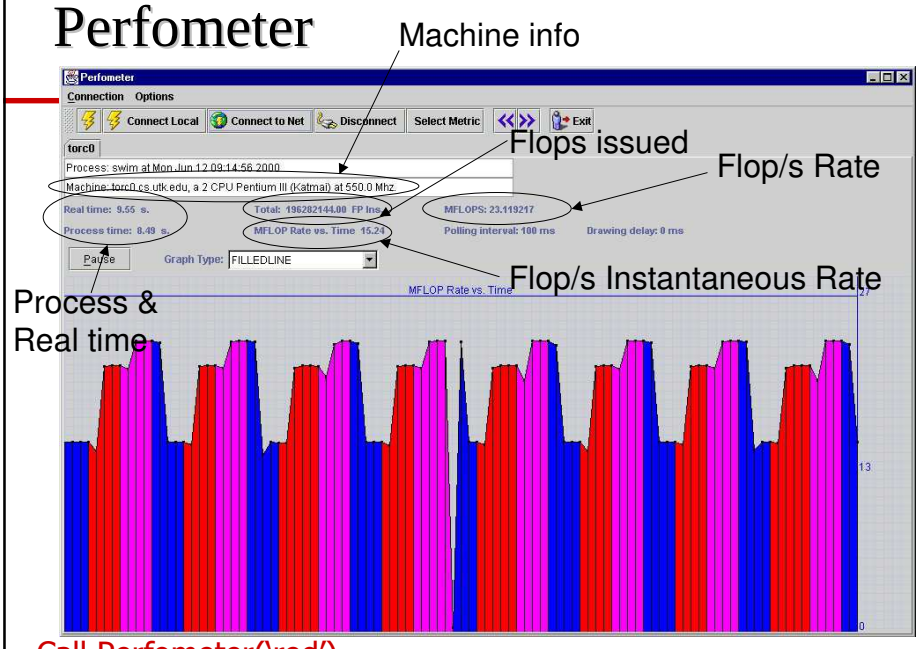
- Counters exist as a small set of registers that count *events.*
- PAPI provides three interfaces to the underlying counter hardware:
  - The low level interface manages hardware events in user defined groups called EventSet.
  - The high level interface simply provides the ability to start, stop and read the counters for a specified list of events.
  - Graphical tools to visualize information.

28

Graphical Tools
Perfometer Usage

- Application is instrumented with PAPI
  - call perfometer()
- Will be layered over the best existing vendor-specific APIs for these platforms
- Sections of code that are of interest are designated with specific colors
  - Using a call to set_perfometer('color')
- Application is started, at the call to perfometer a task is spawned to collect and send the information to a Java applet containing the graphical view.

Perfometer



Call Perfometer('red')

Go To Demo

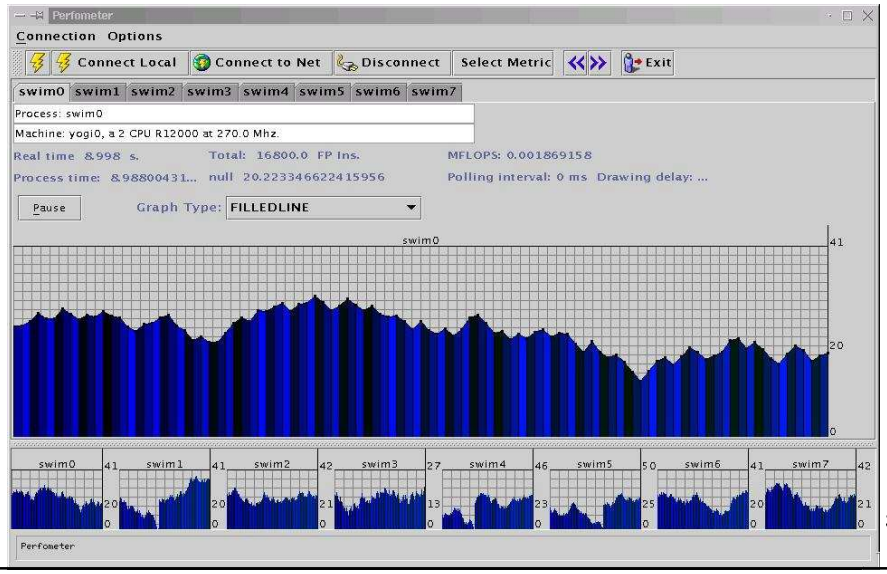
31

PAPI 1.0 Release

- Platforms
  - Linux/x86
  - Solaris/Ultra
  - AIX/Power
  - Tru64/Alpha
  - IRIX/MIPS
- Mailing list
  - send "subscribe ptools-perfapi" to majordomo@ptools.org
  - ptools-perfapi@ptools.org is the reflector
- May require path to kernel
- C and Fortran bindings
- To download software see:
  http://icl.cs.utk.edu/projects/papi/

32

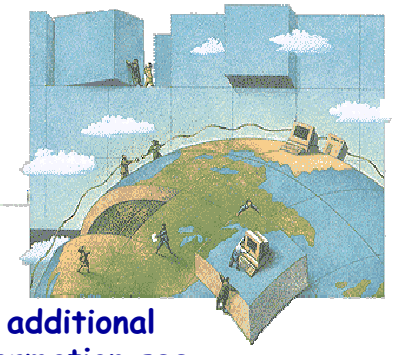
PAPI's Parallel Interface



33

Contributors to These Ideas

- **ATLAS**
  - Antoine Petitet, UTK
  - Clint Whaley, UTK
  - Tatebe Osamu, ETL Japan
- **Recursive Factorization**
  - Victor Eijkhout, UTK
  - Piotr Luszczek, UTK
  - Padma Raghavan, PSU
- **PAPI**
  - Shirley Browne, UTK
  - Kevin London, UTK
  - Phil Mucci, UTK
  - Keith Seymour, UTK



For additional information see…

http://www.netlib.org/atlas/

http://icl.cs.utk.edu/projects/papi/

http://www.cs.utk.edu/~dongarra/

34

Opportunities in the group for visiting and permanent positions