An Update of a Couple of Tools: ATLAS and PAPI

Jack Dongarra
Innovative Computing Laboratory
University of Tennessee

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

1

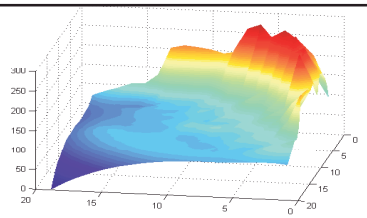
Self-Adapting Numerical Software (SANS)

- Today's processors can achieve high-performance, but this requires extensive machine-specific hand tuning.
- Operations like the BLAS require many man-hours / platform
  - Software lags far behind hardware introduction
  - Only done if financial incentive is there
- Hardware, compilers, 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.
- Need for quick/dynamic deployment of optimized routines.
- ATLAS - Automatic Tuned Linear Algebra Software

2

Software Generation Strategy

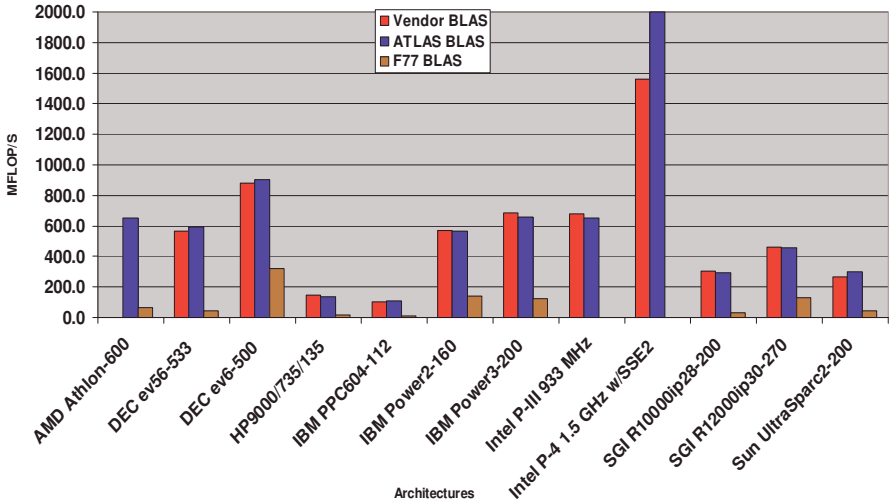
- Parameter study of the hw
- Generate multiple versions of code, w/difference values of key performance parameters
- Run and measure the performance for various versions
- Pick best and generate library
- Level 1 cache multiply optimizes for:
  - TLB access
  - L1 cache reuse
  - FP unit usage
  - Memory fetch
  - Register reuse
  - Loop overhead minimization



- Takes ~ 20 minutes to run.
- "New" model of high performance programming where critical code is machine generated using parameter optimization.
- Designed for RISC arch
  - Super Scalar
  - Need reasonable C compiler
- Today ATLAS in use by Matlab, Mathematica, Octave, Maple, Debian, Scyld Beowulf, SuSE, …

3

ATLAS (DGEMM n = 500)



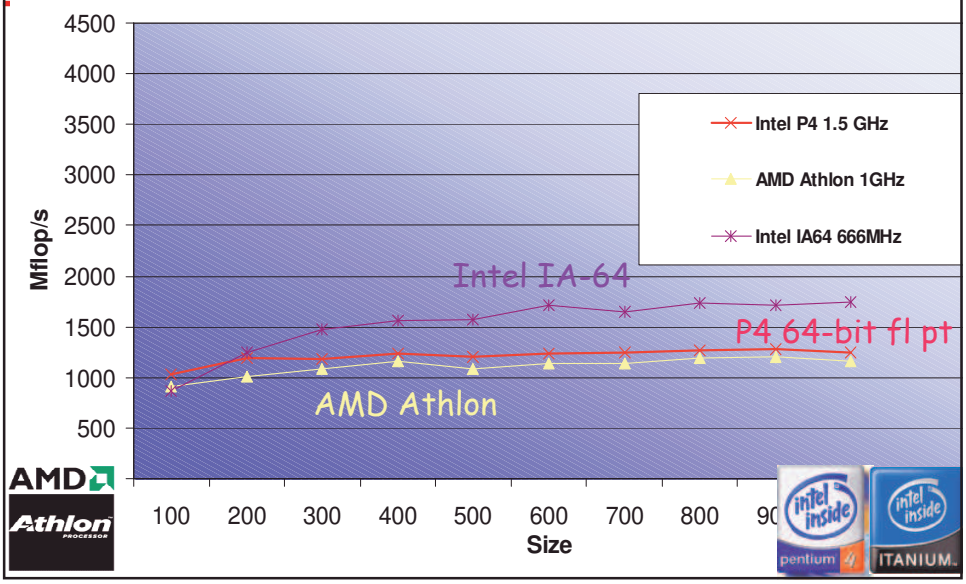
- ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.

Related Tuning Projects

- PHiPAC
  - Portable High Performance ANSI C www.icsi.berkeley.edu/~bilmes/phipac initial automatic GEMM generation project
- FFTW Fastest Fourier Transform in the West
  - www.fftw.org
- UHFFT
  - tune sequential and parallel FFT algorithms
  - rodin.cs.uh.edu/~mirkovic/fft/parfft.htm
- SPIRAL
  - Signal Processing Algorithms Implementation Research for Adaptable Libraries maps DSP algorithms to architectures
- Sparsity
  - Sparse-matrix-vector and Sparse-matrix-matrix multiplication www.cs.berkeley.edu/~ejim/publication/ tunes code to sparsity structure of matrix more later in this tutorial
  - University of Tennessee

5

ATLAS Matrix Multiply (64-bit floating point results)

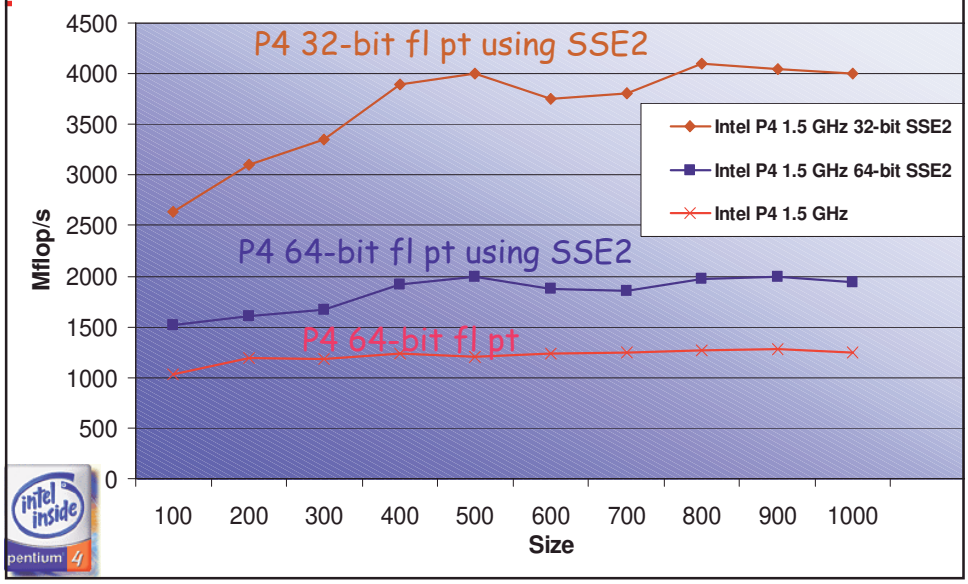


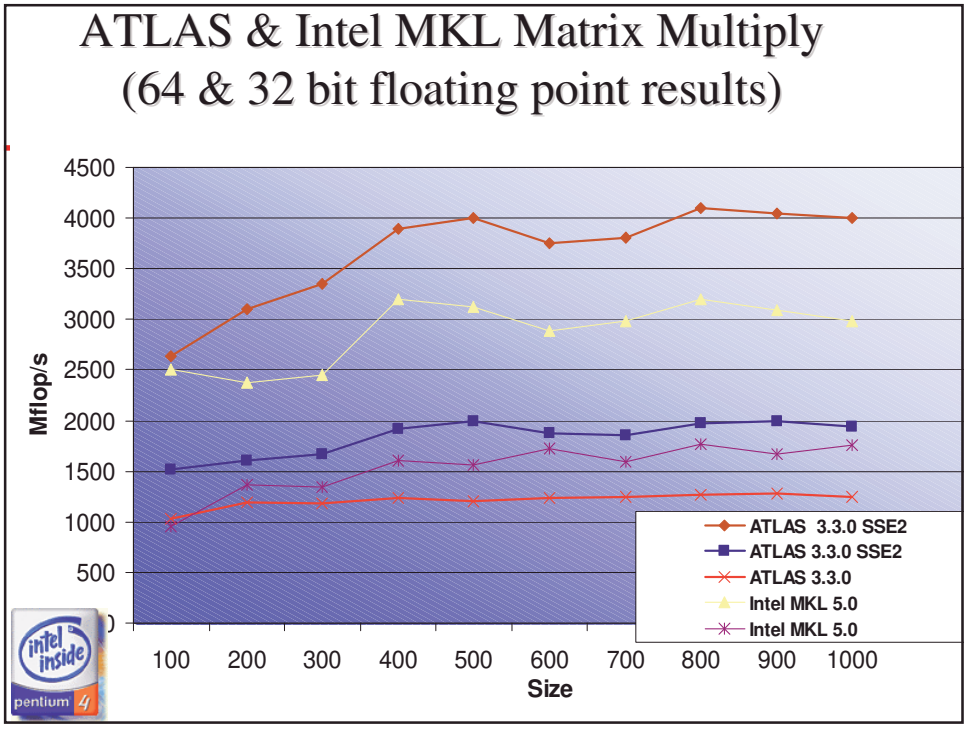
Pentium 4 - SSE2

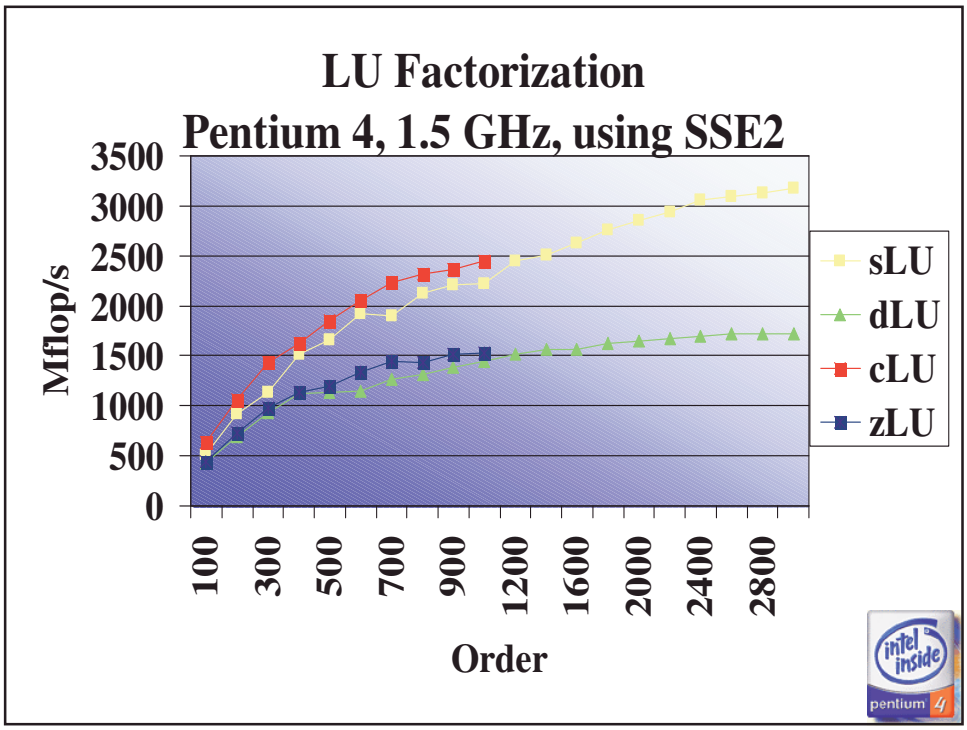
- 1.5 GHz, 400 MHz system bus, 256K L2 Cache, theoretical peak of 1.5 Gflop/s
- Streaming SIMD Extensions 2 (SSE2)
  - which consists of 144 new instructions
  - includes SIMD IEEE double precision floating point
    - Peak for 64 bit floating point 2X
    - Peak for 32 bit floating point 4X
  - SIMD 128-bit integer
  - new cache and memory management instructions.
  - Intel's compiler supports these instructions today

7

ATLAS Matrix Multiply
(64 & 32 bit floating point results)




ATLAS & Intel MKL Matrix Multiply
(64 & 32 bit floating point results)
Mflop/s
4500
4000
3500
3000
2500
2000
1500
1000
500
100
200
300
400
500
600
700
800
900
1000
Size
ATLAS 3.3.0 SSE2
ATLAS 3.3.0 SSE2
ATLAS 3.3.0
Intel MKL 5.0
Intel MKL 5.0
intel inside
pentium 4


LU Factorization
Pentium 4, 1.5 GHz, using SSE2
Mflop/s
3500
3000
2500
2000
1500
1000
500
0
100
300
500
700
900
1200
1600
2000
2400
2800
Order
sLU
dLU
cLU
zLU
intel inside
pentium 4

Tools for Performance Evaluation

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

11

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 the end user or well documented.
- Existing performance counter APIs
  - Compaq Alpha EV 6 & 6/7
  - SGI MIPS R10000
  - IBM Power Series
  - CRAY T3E
  - Sun Solaris
  - Pentium Linux and Windows
  - IA-64
  - HP-PA RISC
  - Hitachi
  - Fujitsu
  - NEC

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

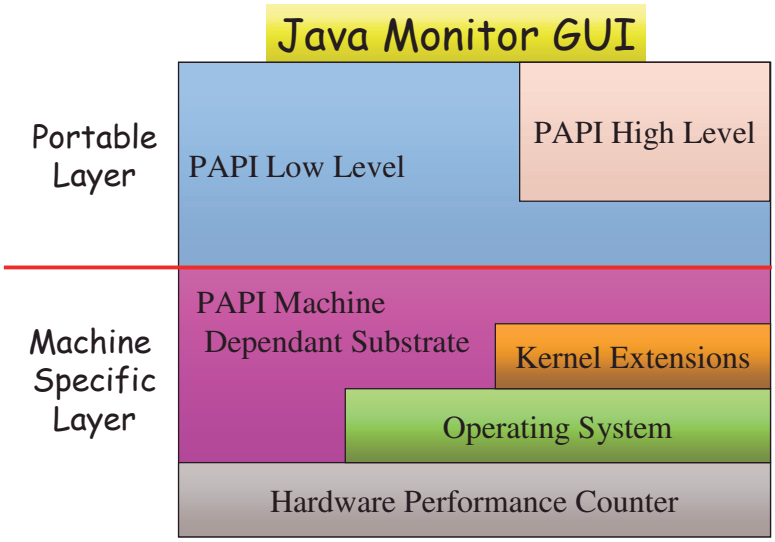
13

Overview of **PAPI**

- Performance Application Programming Interface
- The purpose of the PAPI project is to design, standardize and implement a portable and efficient API to access the hardware performance monitor counters found on most modern microprocessors

14

PAPI Implementation



15

PAPI - Supported Processors

- Pentium,Pro,II,III,4
  - Linux 2.4, 2.2, 2.0 and perf kernel patch
- Power 3,604,604e
  - For AIX 4.3 and pmtoolkit (in 4.3.4 available)
  - (laderose@us.ibm.com)
- UltraSparc I, II, & III
  - Solaris 2.8
- MIPS R10K, R12K
- AMD Athlon
  - Linux 2.4 and perf kernel patch
- Cray T3E, SV1, SV2
- Soon: Alpha EV6, EV67

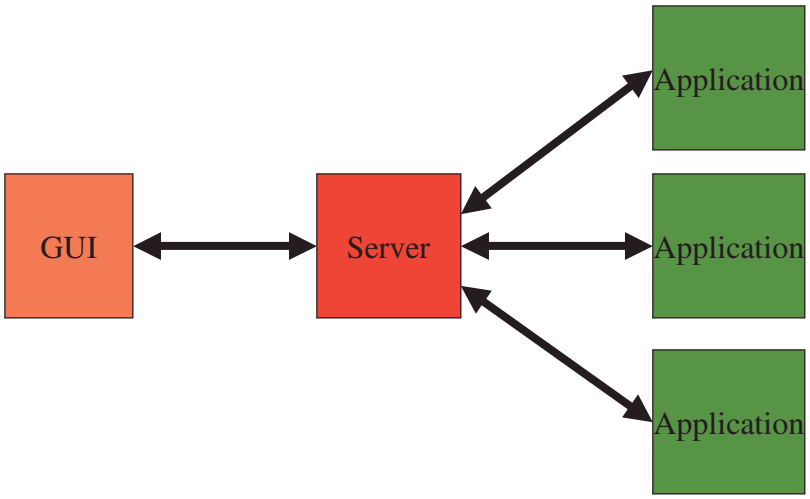
16

Graphical Tools
Perfometer Usage

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

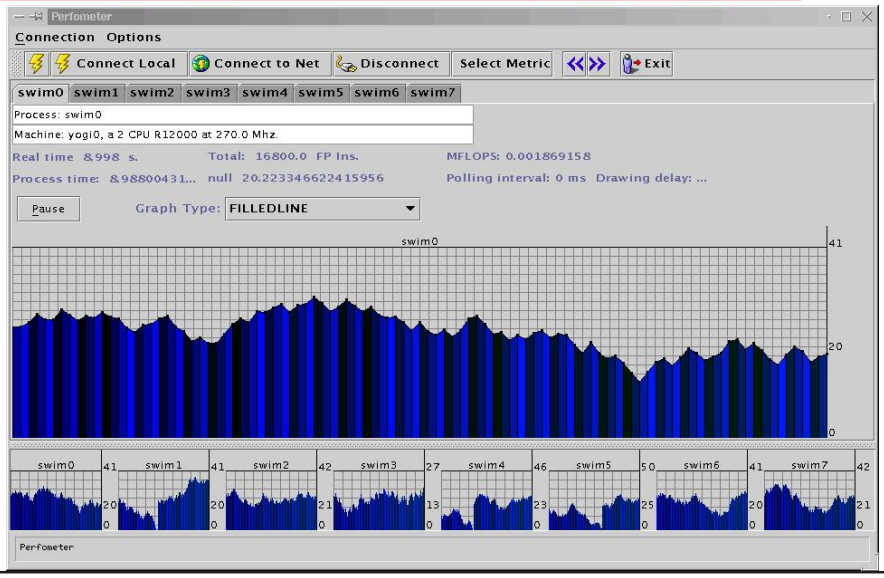
Go To Demo

Next Version of Perfometer Implementation



19

PAPI's Parallel Interface



20

For additional information/software

- http://icl.cs.utk.edu/atlas/
  - Antoine Petitet, Sun France
  - Clint Whaley, UTK
- http://icl.cs.utk.edu/papi
  - Shirley Browne, UTK
  - Kevin London, UTK
  - Phil Mucci, UTK
  - Keith Seymour, UTK
  - Dan Terpstra, UTK

21