Performance Optimization for Cluster Computing

Jack Dongarra
Innovative Computing Laboratory
Computer Science Department
University of Tennessee

Overview

• Self Adapting Numerical Software (SANS) Effort
• LAPACK for Clusters
• PAPI

• Work sponsored by ...

Next Generation Software (NGS)
Cluster Sublist

This is an official ranking. Please visit the site for more information and benchmarks.

Table of results: 

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site Description</th>
<th>Site Name</th>
<th>Integrator</th>
<th>Node Number</th>
<th>Total Processors</th>
<th>Total Peak Performance</th>
<th>Interconnect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Law School</td>
<td>Law School Computing Center</td>
<td>IBM</td>
<td>995</td>
<td>1320</td>
<td>3729.00</td>
<td>Fast Ethernet</td>
</tr>
<tr>
<td>2</td>
<td>State University</td>
<td>State University Computing Center</td>
<td>IBM</td>
<td>1224</td>
<td>1728</td>
<td>2932.00</td>
<td>Myrinet 2000</td>
</tr>
<tr>
<td>3</td>
<td>Penn State</td>
<td>Penn State Computing Center</td>
<td>NVIDIA</td>
<td>616</td>
<td>1000</td>
<td>1307.90</td>
<td>Fast Ethernet</td>
</tr>
<tr>
<td>4</td>
<td>National Laboratory</td>
<td>National Laboratory Computing Facility</td>
<td>IBM</td>
<td>776</td>
<td>1400</td>
<td>1092.00</td>
<td>Fast Ethernet</td>
</tr>
<tr>
<td>5</td>
<td>University Ltd.</td>
<td>University Ltd. Computing Facility</td>
<td>IBM</td>
<td>608</td>
<td>1200</td>
<td>3961.00</td>
<td>Fast Ethernet</td>
</tr>
<tr>
<td>6</td>
<td>Shell Technology</td>
<td>Shell Technology Computing Center</td>
<td>IBM</td>
<td>1000</td>
<td>1000</td>
<td>3057.80</td>
<td>Infiniband</td>
</tr>
<tr>
<td>7</td>
<td>GECA</td>
<td>GECA Computing Center</td>
<td>IBM</td>
<td>145</td>
<td>1000</td>
<td>1052.00</td>
<td>Myrinet 2000</td>
</tr>
<tr>
<td>8</td>
<td>AT&amp;T – A</td>
<td>AT&amp;T Supercomputer Center</td>
<td>IBM</td>
<td>620</td>
<td>1040</td>
<td>967.20</td>
<td>Myrinet 2000</td>
</tr>
<tr>
<td>9</td>
<td>Japan</td>
<td>Japan Supercomputer Center</td>
<td>IBM</td>
<td>612</td>
<td>1024</td>
<td>835.70</td>
<td>Myrinet 2000</td>
</tr>
<tr>
<td>10</td>
<td>University of</td>
<td>University of Illinois Supercomputing Facility</td>
<td>IBM</td>
<td>100</td>
<td>4096</td>
<td>614.90</td>
<td>Fast Ethernet</td>
</tr>
<tr>
<td>11</td>
<td>Infomata Ltd.</td>
<td>Infomata Ltd. Computing Facility</td>
<td>IBM</td>
<td>100</td>
<td>4096</td>
<td>614.90</td>
<td>Fast Ethernet</td>
</tr>
<tr>
<td>12</td>
<td>Lawrence Livermore National Laboratory</td>
<td>Lawrence Livermore National Laboratory</td>
<td>IBM</td>
<td>224</td>
<td>480</td>
<td>763.60</td>
<td>Ethernet</td>
</tr>
<tr>
<td>13</td>
<td>IBM Research</td>
<td>IBM Research Computing Facility</td>
<td>IBM</td>
<td>160</td>
<td>480</td>
<td>256.00</td>
<td>Myrinet 2000</td>
</tr>
<tr>
<td>14</td>
<td>Sony</td>
<td>Sony Computing Facility</td>
<td>IBM</td>
<td>128</td>
<td>640</td>
<td>628.00</td>
<td>Myrinet</td>
</tr>
<tr>
<td>15</td>
<td>OGI Technology Corporation</td>
<td>OGI Technology Computing Facility</td>
<td>IBM</td>
<td>565</td>
<td>768</td>
<td>620.22</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>16</td>
<td>Dell</td>
<td>Dell Advanced Computing Facility</td>
<td>IBM</td>
<td>100</td>
<td>768</td>
<td>463.94</td>
<td></td>
</tr>
</tbody>
</table>

- Peak performance
- Interconnection
- http://clusters.top500.org
- Of the top100 clusters 36 use Myrinet

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

Processor-DRAM Memory Gap (latency)

“Moore’s Law”

μProc 60%/yr. (2X/1.5yr)

DRAM 9%/yr. (2X/10 yrs)
Where Does the Performance Go? or Why Should I Care About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)

- Processor-Memory Performance Gap: (grows 50% / year)
- Processor 60%/yr. (2X/1.5yr)
- DRAM 9%/yr. (2X/10 yrs)

Optimizing Computation and Memory Use

• Computational optimizations
  - Theoretical peak: (# fpu) * (flops/cycle) * Mhz
    - Pentium III: (1 fpu) * (1 flop/cycle) * (850 Mhz) = 850 MFLOP/s
    - Pentium 4: (1 fpu) * (2 flops/cycle) * (2.53 Ghz) = 5060 MFLOP/s
    - Athlon: (2 fpu) * (1flop/cycle) * (600 Mhz) = 1200 MFLOP/s
    - Power3: (2 fpu) * (2 flops/cycle) * (375 Mhz) = 1500 MFLOP/s
Optimizing Computation and Memory Use

**Computational optimizations**

- Theoretical peak: (# feps) * (flops/cycle) * Mhz
  - Pentium III: (1 fpu) * (1 flop/cycle) * (850 Mhz) = 850 MFLOP/s
  - Pentium 4: (1 fpu) * (2 fops/cycle) * (2.53 GHz) = 5060 MFLOP/s
  - Athlon: (2 fpu) * (1 flop/cycle) * (600 Mhz) = 1200 MFLOP/s
  - Power3: (2 fpu) * (2 fops/cycle) * (375 Mhz) = 1500 MFLOP/s

- Operations like:
  - \( \alpha = x^t \gamma \) : 2 operands (16 Bytes) needed for 2 flops; at 850 Mflop/s will requires 1700 MW/s bandwidth
  - \( y = \alpha x + y \) : 3 operands (24 Bytes) needed for 2 flops; at 850 Mflop/s will requires 2550 MW/s bandwidth

**Memory optimization**

- Theoretical peak: (bus width) * (bus speed)
  - Pentium III: (32 bits) * (133 MHz) = 532 MB/s = 66.5 MW/s
  - Pentium 4: (32 bits) * (533 MHz) = 2132 MB/s = 266 MW/s
  - Athlon: (64 bits) * (133 MHz) = 1064 MB/s = 133 MW/s
  - Power3: (128 bits) * (100 MHz) = 1600 MB/s = 200 MW/s
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.

![Diagram showing the memory hierarchy]

- Processor
  - Control
  - Datapath
    - Registers
    - On-Chip Cache
  - Level 2 and 3 Cache (SRAM)
  - Main Memory (DRAM)
  - Secondary Storage (Disk)
  - Tertiary Storage (Disk/Tape)
  - Remote Cluster Memory

<table>
<thead>
<tr>
<th></th>
<th>Speed (ns)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Memory</td>
<td>10 s</td>
<td>Ms</td>
</tr>
<tr>
<td>Secondary</td>
<td>100 s</td>
<td>Gs</td>
</tr>
<tr>
<td>Tertiary</td>
<td>10,000,000s</td>
<td>Ts</td>
</tr>
<tr>
<td>Remote Cluster</td>
<td>10,000,000s</td>
<td></td>
</tr>
</tbody>
</table>

Motivation Self Adapting Numerical Software (SANS) Effort

• Optimizing software to exploit the features of a given processor has historically been an exercise in hand customization.
  — Time consuming and tedious
  — Hard to predict performance from source code
  — Growing list of kernels to tune
  — Must be redone for every architecture and compiler
    - Compiler technology often lags architecture
    - Best algorithm may depend on input, so some tuning may be needed at run-time.
    - Not all algorithms semantically or mathematically equivalent
    — Need for quick/dynamic deployment of optimized routines.
What is Self Adapting Performance Tuning of Software?

• Two steps:
  1. Identify and generate a space of algorithm/software, with various
     - Instruction mixes and orders
     - Memory Access Patterns
     - Data structures
     - Mathematical Formulations
  2. Search for the fastest one, by running them

• When do we search?
  — Once per kernel and architecture
  — At compile time
  — At run time
  — All of the above

• Many examples
  — PHiPAC, ATLAS, Sparsity, FFTW, Spiral,…

Self Adapting Numerical Software - SANS Effort

• Provide software technology to aid in high performance on commodity processors, clusters, and grids.
• Pre-run time (library building stage) and run time optimization.
• Integrated performance modeling and analysis
• Automatic algorithm selection - polyalgorithmic functions
• Automated installation process
• Can be expanded to areas such as communication software and selection of numerical algorithms
Self-Adapting Numerical Software (SANS) Effort

• The complexities of modern processors or clusters makes it difficult to analytically predict or model by hand the performance.

• Operations as simple as 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 microarchitectures of new microprocessors.

• Need for quick/dynamic deployment of optimized routines.

Software Generation Strategy - ATLAS BLAS

• 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, generates Level 1, 2, & 3 BLAS

• “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 used within various ASCI and SciDAC activities and by Matlab, Mathematica, Octave, Maple, Debian, Scyld Beowulf, SuSE,...
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.

Pentium 4 - SSE2
Today’s “Sweet Spot” in Price/Performance

- 2.53 GHz, 533 MHz front side bus, 8K (data) L1 & 512K L2 Cache, theoretical peak of 2.53 Gflop/s (w/o SSE2), high power consumption (59.3 Watts)

- 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
  - ATLAS was trained to probe and detect SSE2
### Table 1: Performance in Solving a System of Linear Equations

<table>
<thead>
<tr>
<th>Computer</th>
<th>&quot;LINPACK Benchmark&quot;</th>
<th>&quot;TPP&quot;</th>
<th>&quot;Theoretical Peak&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n=100$</td>
<td>$n=1000$, Mflop/s</td>
<td>$n=1000$, Mflop/s</td>
</tr>
<tr>
<td>Intel P4 2.53 GHz</td>
<td><code>-O3 -2W -ipo -4p -align</code> 1190</td>
<td>14400</td>
<td>5960</td>
</tr>
<tr>
<td>NEC SX-6/8 (8proc. 2.0 ms)</td>
<td>41520</td>
<td>64000</td>
<td></td>
</tr>
<tr>
<td>NEC SX-6/4 (4proc. 2.0 ms)</td>
<td>23680</td>
<td>32000</td>
<td></td>
</tr>
<tr>
<td>NEC SX-6/2 (2proc. 2.0 ms)</td>
<td>13530</td>
<td>16000</td>
<td></td>
</tr>
<tr>
<td>NEC SX-6/1 (1proc. 2.0 ms)</td>
<td>7675</td>
<td>8000</td>
<td></td>
</tr>
<tr>
<td>Fujitsu VPP5000/1 (1/4 proc. 333mHz)</td>
<td>R12.1 -pi -Wf -prob,use 1156</td>
<td>8784</td>
<td>9600</td>
</tr>
<tr>
<td>Cray T912 (32 proc. 2.2 ms)</td>
<td><code>-Wv,-rt28,-Of,-KA32</code> 1156</td>
<td>23660</td>
<td>57900</td>
</tr>
<tr>
<td>Cray T924 (28 proc. 2.2 ms)</td>
<td>23840</td>
<td>50400</td>
<td></td>
</tr>
<tr>
<td>Cray T924 (24 proc. 2.2 ms)</td>
<td>26170</td>
<td>43200</td>
<td></td>
</tr>
<tr>
<td>Cray T916 (16 proc. 2.2 ms)</td>
<td>19860</td>
<td>28800</td>
<td></td>
</tr>
<tr>
<td>Cray T916 (8 proc. 2.2 ms)</td>
<td>10680</td>
<td>14400</td>
<td></td>
</tr>
<tr>
<td>Cray T94 (4 proc. 2.2 ms)</td>
<td><code>-00,-2collie2</code> 1129</td>
<td>5735</td>
<td>7200</td>
</tr>
<tr>
<td>IBM eServer pSeries 690 Turbo 16 proc (1300 MHz)</td>
<td>28800</td>
<td>82200</td>
<td></td>
</tr>
<tr>
<td>IBM eServer pSeries 690 Turbo 8 proc (1300 MHz)</td>
<td>18290</td>
<td>41600</td>
<td></td>
</tr>
<tr>
<td>IBM eServer pSeries 690 Turbo 1 proc (1300 MHz)</td>
<td><code>-G3 -qarch=pwr4 -qtime=pwr4</code> 1674</td>
<td>2864</td>
<td>5200</td>
</tr>
</tbody>
</table>

~$2000 for system => $0.50/Mflops!
Related Tuning Projects

- **UHFFT**
  - tuning parallel FFT algorithms
  - rodin.cs.uh.edu/~mirkovic/fft/parfft.htm

- **FFTW Fastest Fourier Transform in the West**
  - www.fftw.org

- **PHIPAC**
  - Portable High Performance ANSI C
  - www.icsi.berkeley.edu/~bilmes/phinac initial automatic GEMM
  - generation project

- **SPIRAL**
  - Signal Processing Algorithms Implementation Research for Adaptable
  - Libraries maps DSP algorithms to architectures

- **Sparsity**
  - Tunes code to sparsity structure of matrix
  - Sparse-matrix-vector and Sparse-matrix-matrix multiplication
    - University of California, Berkeley
    - http://www.cs.berkeley.edu/~yelick/sparisty/
    - University of Tennessee

---

Machine-Assisted Application Development and Adaptation

- **Communication libraries**
  - Optimize for the specifics of one’s configuration.
  - A specific MPI collective communication algorithm may not give best results on all platforms.
  - Choose collective communication parameters that give best results for the system.

- **Algorithm layout and implementation**
  - Look at the different ways to express implementation
Work in Progress:
SANS Approach Applied to Broadcast

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

<table>
<thead>
<tr>
<th>Message Size (bytes)</th>
<th>Optimal algorithm Buffer Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>binomial</td>
</tr>
<tr>
<td>16</td>
<td>binomial</td>
</tr>
<tr>
<td>32</td>
<td>binary</td>
</tr>
<tr>
<td>64</td>
<td>binomial</td>
</tr>
<tr>
<td>128</td>
<td>binomial</td>
</tr>
<tr>
<td>256</td>
<td>binomial</td>
</tr>
<tr>
<td>512</td>
<td>binomial</td>
</tr>
<tr>
<td>1K</td>
<td>sequential</td>
</tr>
<tr>
<td>2K</td>
<td>binary</td>
</tr>
<tr>
<td>4K</td>
<td>binary</td>
</tr>
<tr>
<td>8K</td>
<td>binary</td>
</tr>
<tr>
<td>16K</td>
<td>binary</td>
</tr>
<tr>
<td>32K</td>
<td>binary</td>
</tr>
<tr>
<td>64K</td>
<td>ring</td>
</tr>
<tr>
<td>128K</td>
<td>ring</td>
</tr>
<tr>
<td>256K</td>
<td>ring</td>
</tr>
<tr>
<td>512K</td>
<td>ring</td>
</tr>
<tr>
<td>1M</td>
<td>binary</td>
</tr>
</tbody>
</table>
CG Variants by Dynamic Selection at Run Time

- Variants combine inner products to reduce communication bottleneck at the expense of more scalar ops.
- Same number of iterations, no advantage on a sequential processor.
- With a large number of processor and a high-latency network may be advantages.
- Improvements can range from 15% to 50% depending on size.

<table>
<thead>
<tr>
<th>Classical</th>
<th>Solved/Minorant</th>
<th>Choorespeed/Gear</th>
<th>Ejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix calculation: $|error = \sqrt{|r}|$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preconditioner application: $z = M^{-1}r$</td>
<td>$x \leftarrow z - u|g$</td>
<td>$x \leftarrow M^{-1}r$</td>
<td>id</td>
</tr>
<tr>
<td>Matrix-vector product: $uz \leftarrow A \times x$</td>
<td></td>
<td>id</td>
<td></td>
</tr>
<tr>
<td>Inner products 1:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a \leftarrow \rho |x$</td>
<td>$\pi \leftarrow \rho |x$</td>
<td>residual update: $r \leftarrow r - a|x$</td>
<td>3 separate inner products</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\beta = \mu / \|x\| \\
\text{Search direction update:} \quad \mu = z + \beta \|x\| \\
\text{Matrix-vector product:} \quad up \leftarrow A \times p \\
\text{Preconditioner application:} \quad y = M^{-1}up \\
\text{Inner products 2:} \quad \pi \leftarrow \rho \|y\| \\
\text{Residual update:} \quad r \leftarrow r - a\|x$ |
LAPACK For Clusters

- Developing middleware which couples cluster system information with the specifics of a user problem to launch cluster based applications on the “best” set of resource available.

Sample computing environment...

- Using ScaLAPACK as the prototype software

ScaLAPACK

- ScaLAPACK is a portable distributed memory numerical library
- Complete numerical library for dense matrix computations
- Designed for distributed parallel computing (MPP & Clusters) using MPI
- One of the first math software packages to do this
- Numerical software that will work on a heterogeneous platform
- In use today by various ASCI and SciDAC efforts, IBM, HP-Convex, Fujitsu, NEC, Sun, SGI, Cray, NAG, IMSL, ...
  — Tailor performance & provide support
How ScaLAPACK Works

- To use ScaLAPACK a user must:
  - Download the package and auxiliary packages (like PBLAS, BLAS, BLACS, & MPI) to the machines.
  - If heterogeneous collection of machines, make sure proper versions available.
  - Write a SPMD program which
    - Sets up the logical 2-D process grid
    - Places the data on the logical process grid
    - Calls the library routine in a SPMD fashion
    - Collects the solution after the library routine finishes
  - The user must allocate the processors and decide the number of processes the application will run on
  - The user must commit to a certain # of processors then start the application
    - "mpirun -np N user_app"
      Note: the number of processors is fixed by the user before the run
  - Upon completion, return the processors to the pool of resources

LAPACK For Clusters

- Idea to make it easy to use your cluster to solve dense matrix problems.
- As simple as a conventional call to LAPACK
- Make decisions on which machines to use based on the user’s problem and the state of the system
  - Determine machines that can be used
  - Optimize for the best time to solution
  - Distribute the data on the processors and collections of results
  - Start the SPMD library routine on all the platforms
Big Picture...

- User has problem to solve (e.g. $Ax = b$)
- Natural Data ($A, b$)
- Natural Answer ($x$)
- Middleware
- Structured Data ($A', b'$)
- Application Library (e.g. LAPACK, ScALAPACK, PETSc,…)
- Structured Answer ($x'$)

File System -based

- User
- Stage data to disk
- $A, b$
File System -based

Can use Grid infrastructure, i.e. Globus/NWS, but doesn’t have to.

Resource Selector

- Uses Rich Wolski’s (UCSB) NWS to build an array of values for the machines that are available for the user.
  - 2 matrices (bw,lat) 3 arrays (load, cpu, memory available)

- Generated dynamically by library routine

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Latency</th>
<th>Load</th>
<th>Memory</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>X X . . X</td>
<td>X X . . X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X . . X</td>
<td>. . . . .</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>. . . . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X . . X</td>
<td>X X . . X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Page 17
Ax = b
Cluster of 8 Pentium III 933 MHz

<table>
<thead>
<tr>
<th>Size</th>
<th>Time to Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.1</td>
</tr>
<tr>
<td>1024</td>
<td>1</td>
</tr>
<tr>
<td>2048</td>
<td>10</td>
</tr>
<tr>
<td>4096</td>
<td>100</td>
</tr>
<tr>
<td>8192</td>
<td>1000</td>
</tr>
<tr>
<td>10240</td>
<td>10000</td>
</tr>
</tbody>
</table>

LAPACK For Clusters (LFC)

- **LFC** will automate much of the decisions in the Cluster environment to provide best time to solution.
  - Adaptivity to the dynamic environment.
  - As the complexities of the Clusters and Grid increase need to develop strategies for self adaptability.
  - Handcrafted developed leading to an automated design.
- Developing a basic infrastructure for computational science applications and software in the Cluster and Grid environment.
  - Lack of tools is hampering development today.
- Plan to do suite: LU, Cholesky, QR, Symmetric eigenvalue, and Nonsymmetric eigenvalue
- Model for more general framework
**TORC Cluster**

- Torc is a cluster of 8 Dual PIII 550 MHz machines
  - each containing 512MB of Ram.
- **Fast Ethernet**
  - 3Com Fast Etherlink 905TX 10/100BaseT NIC (integrated)
  - 16 Port Fast Ethernet Bay Networks Model 350T
- **Myrinet**
  - Myricom PCI64 Lanai 7.2 NIC
  - Myricom 8 Port m2m-dual-sw8 switch
- **Gigabit Ethernet**
  - 3Com Model 3C996 Copper Gigabit NICS
  - Dell PowerConnect 5012 Gig Copper Switch 10 Ports

---

**MPI bytes broadcast vs. time, TORC**

![Graph showing MPI bytes broadcast vs. time for different networks.](image)

- IP, GigCable(Cu)
- Fast Ethernet
- Myrinet
- IP, Myrinet
BLACS broadcast results, TORC
[p,q]=[1,16]

Bytes Broadcast

Ax=b using HPL
16 Pentium III 550 MHz TORC

Problem Size (N)
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

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

PAPI - Supported Processors

- Intel Pentium, II, III, Itanium, (P 4 in alpha testing now)
  - Linux 2.4, 2.2, 2.0 and perf kernel patch
- IBM Power 3, 604, 604e (Power 4 coming)
  - For AIX 4.3 and pmtoolkit (in 4.3.4 available)
  - (jaderose@us.ibm.com)
- Sun UltraSparc I, II, & III
  - Solaris 2.8
- SGI IRIX/MIPS
- AMD Athlon
  - Linux 2.4 and perf kernel patch
- Cray T3E, SV1, SV2
- Windows 2K and XP
- To download software see:
  http://icl.cs.utk.edu/papi/
  Work in progress on Compaq Alpha Fortran, C, and MATLAB bindings
Early Users of PAPI

- DEEP/PAPI (Pacific Sierra)
  [http://www.psrv.com/deep_papi_top.html](http://www.psrv.com/deep_papi_top.html)
- TAU (Allen Mallony, U of Oregon)
  [http://www.cs.uoregon.edu/research/paracomp/tau/](http://www.cs.uoregon.edu/research/paracomp/tau/)
- SvPablo (Dan Reed, U of Illinois)
  [http://vibes.cs.uiuc.edu/Software/SvPablo/svPablo.htm](http://vibes.cs.uiuc.edu/Software/SvPablo/svPablo.htm)
- Cactus (Ed Seidel, Max Plank/U of Illinois)
  [http://www.aei-potsdam.mpg.de](http://www.aei-potsdam.mpg.de)
- Vprof (Curtis Janssen, Sandia Livermore Lab)
- Cluster Tools (Al Geist, ORNL)
- DynaProf

What is DynaProf?

- A portable tool to dynamically instrument a running executable with Probes that monitor application performance.
- Simple command line interface.
- Java based GUI interface.
- Open Source Software.
- Built on and in collaboration with Bart Miller and Jeff Hollingsworth Paradyn project at U. Wisconsin and Dyninst project at U. Maryland
  - [http://www.paradyn.org/](http://www.paradyn.org/)
- A work in progress...
Dynamic Instrumentation:

- Operates on a running executable.
- Identifies instrumentation points where code can be inserted.
- Inserts code snippets at selected points.
- Snippets can collect and monitor performance information.
- Snippets can be removed and reinserted dynamically.
- Source code not required, just executable
Next Version of Perfometer Implementation

PAPI’s Parallel Interface
Futures for Numerical Algorithms and Software on Clusters and Grids

- Retargetable Libraries - Numerical software will be adaptive, exploratory, and intelligent
- Determinism in numerical computing will be gone.
  - After all, it’s not reasonable to ask for exactness in numerical computations.
  - Auditability of the computation, reproducibility at a cost
- Importance of floating point arithmetic will be undiminished.
  - 16, 32, 64, 128 bits and beyond.
- Reproducibility, fault tolerance, and auditability
- Adaptivity is a key so applications can effectively use the resources.

Collaborators

- **ATLAS**
  - Antoine Petitet, Sun
  - Clint Whaley, FSU
- **Sparse Ops**
  - Victor Eijkhout, UTK
- **Optimizing communication**
  - Sathish Vadhiyar, UTK
- **LFC**
  - Jeffrey Chen, UTK
  - Piotr Luszczek, UTK
  - Kenny Roche, UTK
- **PAPI**
  - Kevin London, UTK
  - Shirley Moore, UTK
  - Phil Mucci, UTK
- **DynaProf**
  - Jeff Hollingsworth, UMaryland
  - Bart Miller, UWisconsin
  - Dan Terpstra, UTK
  - Haihang You, UTK

Software Availability

- **ATLAS**
- **LFC**
  - 5 drivers from ScaLAPACK by the end of summer
- **PAPI**
- **DynaProf**
  - tarball coming soon