Myrinet User's Group Conference
12-14 May 2002
Vienna, Austria

Performance Optimization for Cluster Computing

Jack Dongarra
Innovative Computing Laboratory
Computer Science Department
University of Tennessee

1

Overview

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

- Work sponsored by …

LACSI

OFFICE OF SCIENCE
U.S. DEPARTMENT of ENERGY

Scientific Discovery through Advanced Computing (SciDAC)

CISE NSF

TOPS
Terascale Optimal PDE Simulations

Next Generation Software (NGS)

GrADS
Grid Application Development Software Project

PERC

2

@ TOP500 ! Mon May 6

Cluster Sublist

This is **no official ranking**. Please read here to learn more about the results and the benchmarks.

Number of results: 100

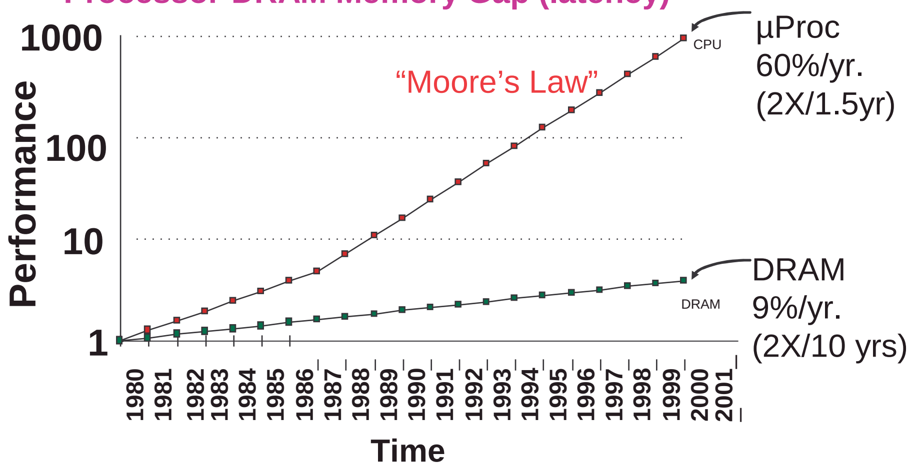
Go back to form

#	Site	Country	System Name	Integrator	Node Number	Total Processors	Total Peak Performance	Interconnect
1	Locus Discovery	USA	Locus Supercluster	Self, Western Scientific, VA L	960	1920	1920.00	Fast Ethernet
2	IWR, University Heidelberg	Germany	HELICS	MEGWARE, IWR	256	512	1433.60	Myrinet 2000
3	Mississippi State University	USA	EMPIRE	IBM	519	1038	1157.40	Fast Ethernet
4	Brookhaven National Laboratory	USA	RHIC Computing Facility	VA Linux and IBM	706	1412	1082.80	Fast Ethernet
5	Inpharmatica Ltd.	United Kingdom	Biopendium	In house	800	1220	1061.00	Fast Ethernet
6	Shell Technology Exploration and Production	Netherlands	Genesis Machine	IBM	1030	1038	1037.10	Gigabit Ethernet
7	NCSA	USA	Platinum	IBM	516	1032	1032.00	Myrinet 2000
8	AIST - Computational Biology Research Center	Japan	CBRC Magi system	NEC	520	1040	967.20	Myrinet 2000
9	Real World Computing Partnership	Japan	RWC SCore Cluster III	Self-made	512	1024	955.40	Myrinet 2000
10	University of Utah, Center for High Performance Computing	USA	ICE Box	Self Made	303	406	914.90	Fast Ethernet
11	Inphamatica Ltd.	United Kingdom	Biopendium II	Northgate Information Solution	880	880	856.00	Fast Ethernet
12	Lawrence Livermore National Laboratory	USA	Lawrence Livermore National Laboratory	Linux NetworX	224	448	761.60	Ethernet
13	Incyte Genomics	USA	Incyte Genomics	In house	767	1511	754.00	Gigabit Ethernet
14	Sandia National Lab	USA	CPlant Siberia	Self-made	628	628	628.00	Myrinet
15	GX Technology Corporation	USA	HGSC	GX Technology	585	754	620.22	Gigabit Ethernet
16	Sun	USA	HPC 4500 Cluster	Sun	90	720	483.84	

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

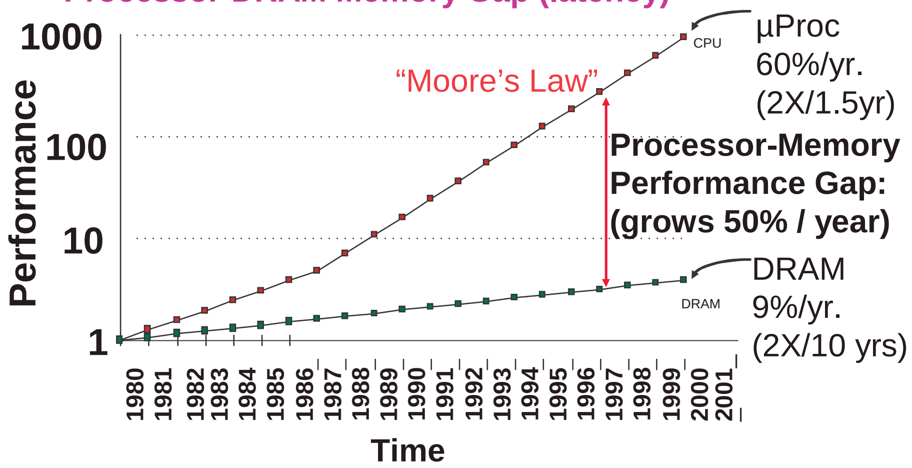
Where Does the Performance Go? or Why Should I Care About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)



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

Processor-DRAM Memory Gap (latency)



5

Optimizing Computation and Memory Use

- Computational optimizations
  - Theoretical peak:(# fpus)*(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

6

Optimizing Computation and Memory Use

- **Computational optimizations**
  - Theoretical peak:(# fpus)*(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
- **Operations like:**
  - $\alpha = x^T y$: 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

7

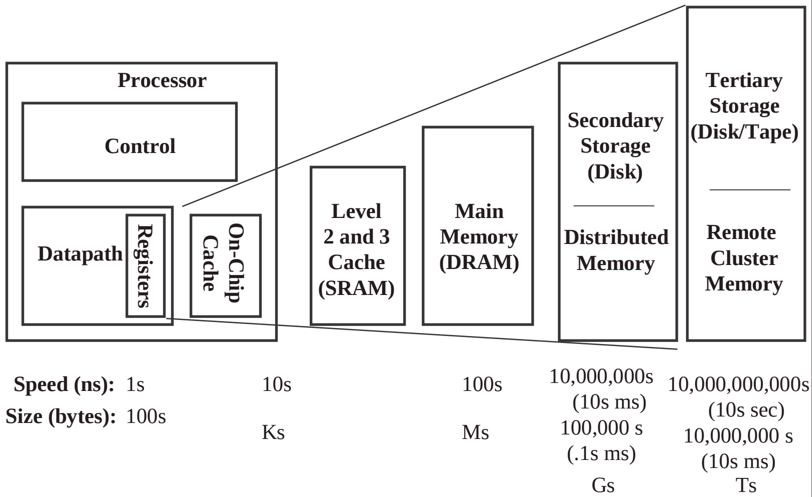
Optimizing Computation and Memory Use

- **Computational optimizations**
  - Theoretical peak:(# fpus)*(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
- **Operations like:**
  - $\alpha = x^T y$: 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

8

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.



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.

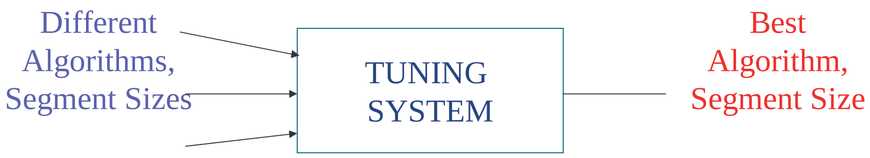
What is Self Adapting Performance Tuning of Software?

- **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,…

11

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



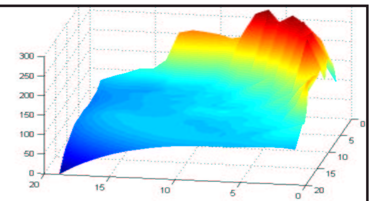
12

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 micro-architectures of new microprocessors.
- Need for quick/dynamic deployment of optimized routines.

13

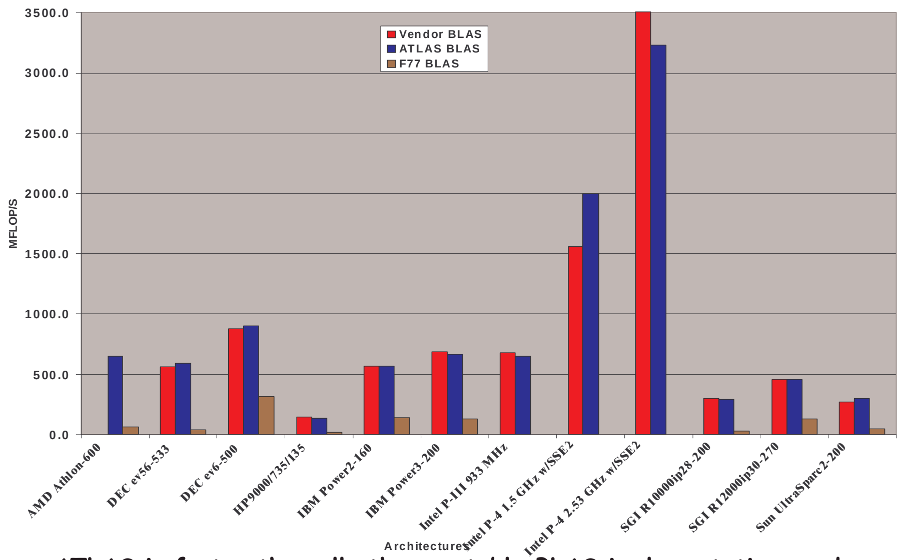
Software Generation Strategy - ATLAS BLAS



- 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,…

14

ATLAS (DGEMM n = 500)



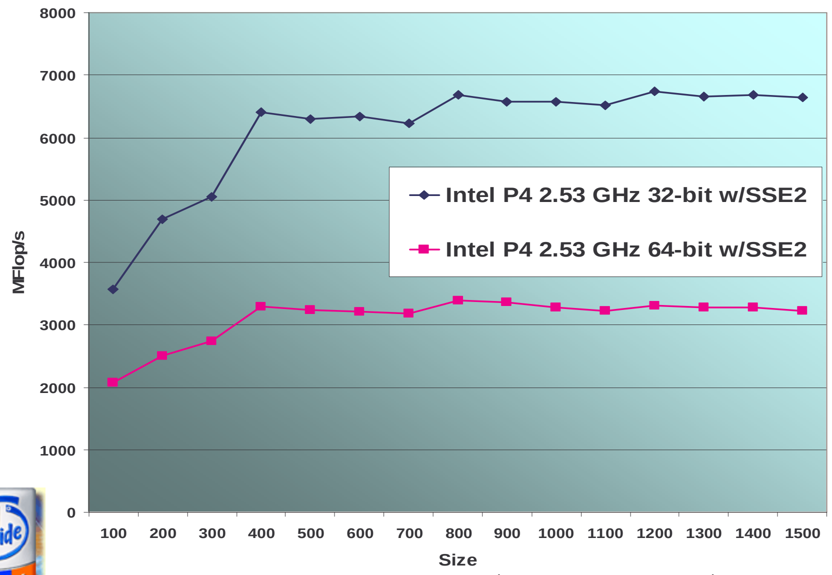
- **ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.**

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

16

ATLAS Matrix Multiply
Intel Pentium 4 at 2.53 GHz – using SSE2



~$2000 for system => $0.50/Mflops !!

17

Table 1: **Performance in Solving a System of Linear Equations**

Computer	"LINPACK Benchmark" n = 100 OS/Compiler	Mflop/s	"TPP" Best Effort n=1000, Mflop/s	"Theoretical Peak" Mflop/s
Intel P4 2.53 Ghz	ifc -O3 -xW -ipo -ip -align	1190	2355	5060
NEC SX-6/8 (8proc. 2.0 ns)			41520	64000
NEC SX-6/4 (4proc. 2.0 ns)			23680	32000
NEC SX-6/2 (2proc. 2.0 ns)			13350	16000
NEC SX-6/1 (1proc. 2.0 ns)	R12.1 -pi -Wf"-prob_use"	1161	7575	8000
Fujitsu VPP5000/1(1 proc.3.33ns)	frt -Wv,-r128 -Of -KA32	1156	8784	9600
Cray T932 (32 proc. 2.2 ns)			29360	57600
Cray T928 (28 proc. 2.2 ns)			28340	50400
Cray T924 (24 proc. 2.2 ns)			26170	43200
Cray T916 (16 proc. 2.2 ns)			19980	28800
Cray T916 (8 proc. 2.2 ns)			10880	14400
Cray T94 (4 proc. 2.2 ns)	f90 -O3,inline2	1129	5735	7200
IBM eServer pSeries 690 Turbo 16 proc(1300 MHz)			28080	83200
IBM eServer pSeries 690 Turbo 8 proc(1300 MHz)			18290	41600
IBM eServer pSeries 690 Turbo 1 proc(1300 MHz)	-O3 -qarch=pwr4 -qtune=pwr4 -Pv -Wp,-ea478,-g1	1074	2894	5200

18

Related Tuning Projects

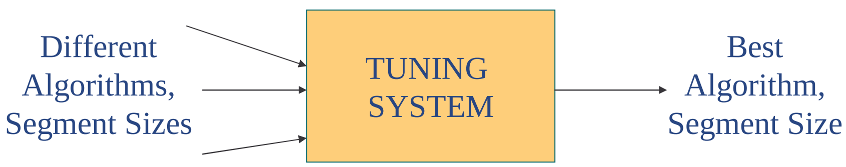
- UHFFT
  - tuning parallel FFT algorithms
  - rodin.cs.uh.edu/~mirkovic/fft/parfft.htm
- FFTW Fastest Fourier Transform in the West
  - www.fftw.org
- PHiPAC
  - Portable High Performance ANSI C www.icsi.berkeley.edu/~bilmes/phipac initial automatic GEMM generation project
- SPIRAL
  - Signal Processing 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/sparsity/
    - University of Tennessee

19

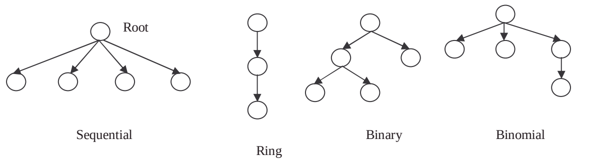
Machine-Assisted Application Development and Adaptation

- 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



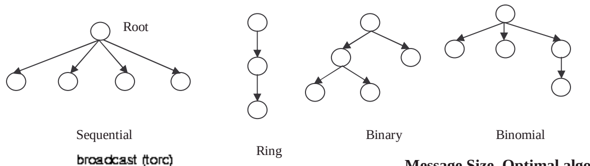
20

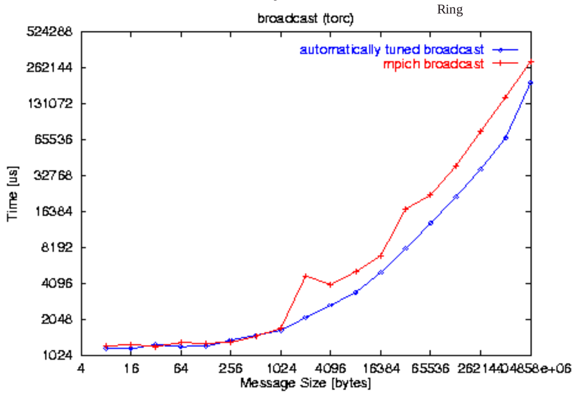
Work in Progress:
SANS Approach Applied to Broadcast
(PII 8 Way Cluster with 100 Mb/s switched network)



21

Work in Progress:
SANS Approach Applied to Broadcast
(PII 8 Way Cluster with 100 Mb/s switched network)





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

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.

Classical
Norm calculation:
$\boxed{error = \sqrt{r^t r}}$
Preconditioner application:
$z \leftarrow M^{-1} r$
Matrix-vector product:
Inner products 1:
$\boxed{\rho \leftarrow z^t r}$
$\beta \leftarrow \rho / \rho_{\text{old}}$
Search direction update:
$p \leftarrow z + \beta p$
Matrix-vector product:
$ap \leftarrow A \times p$
Preconditioner application:
Inner products 2:
$\boxed{\pi \leftarrow p^t ap}$
$\alpha = \rho / \pi$
Residual update:
$r \leftarrow r - \alpha Ap$
3 separate inner products

23

CG Variants by Dynamic Selection at Run Time

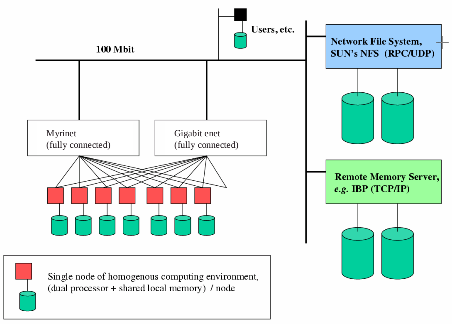
- Variants combine inner products to reduce communication bottleneck at the expense of more scalar ops.
- Same number of iterations, no advantage on a sequential processor
- With a large number of processor and a high-latency network may be advantages.
- Improvements can range from 15% to 50% depending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

Classical	Saad/Meurant	Chronopoulos/Gear	Eijkhout
Norm calculation:			
$\boxed{error = \sqrt{r^t r}}$			
Preconditioner application:			
$z \leftarrow M^{-1} r$	$z \leftarrow z - \alpha q$	$z \leftarrow M^{-1} r$	id
Matrix-vector product:			
		$az \leftarrow A \times z$	id
Inner products 1:			
$\boxed{\rho \leftarrow z^t r}$	$\rho_{\text{predict}} \leftarrow -\rho_{\text{true}} + \alpha^2 \mu$	$error = \sqrt{r^t r}$ $\rho \leftarrow z^t r$ $\zeta \leftarrow z^t az$ (boxed)	$error = \sqrt{r^t r}$ $\rho \leftarrow z^t r$ $\zeta \leftarrow z^t az$ $\epsilon \leftarrow (M^{-1} r)^t (Ap)$ (boxed)
$\beta \leftarrow \rho / \rho_{\text{old}}$	$\beta = \rho_{\text{predict}} / \rho_{\text{old}}$	$\beta \leftarrow \rho / \rho_{\text{old}}$	id
Search direction update:			
$p \leftarrow z + \beta p$	id	id	id
Matrix-vector product:			
$ap \leftarrow A \times p$	id	$ap \leftarrow az + \beta ap$	id
Preconditioner application:			
	$q \leftarrow M^{-1} ap$		
Inner products 2:			
$\boxed{\pi \leftarrow p^t ap}$	$\pi \leftarrow p^t ap$ $\mu \leftarrow ap^t q$ $error = \sqrt{r^t r}$ $\rho_{\text{true}} = z^t r$ (boxed)	$\pi \leftarrow \zeta - \beta^2 \pi$	$\pi \leftarrow \zeta + \beta \epsilon$
$\alpha = \rho / \pi$	$\cdots \rho_{\text{true}} \cdots$	$\alpha = \rho / \pi$	
Residual update:			
$r \leftarrow r - \alpha Ap$	id	id	id
3 separate inner products	4 combined	3 combined	4 combined
	1 extra vector update	id	id

24

LAPACK For Clusters

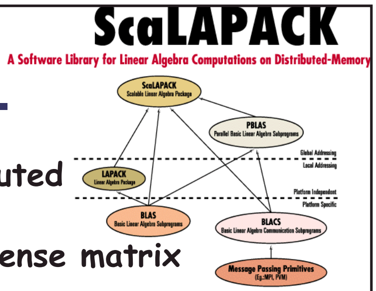
- Developing middleware which couples cluster system information with the specifics of a user problem to launch cluster based applications on the "best" set of resource available.



- Using ScaLAPACK as the prototype software

25

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

26

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

hetero

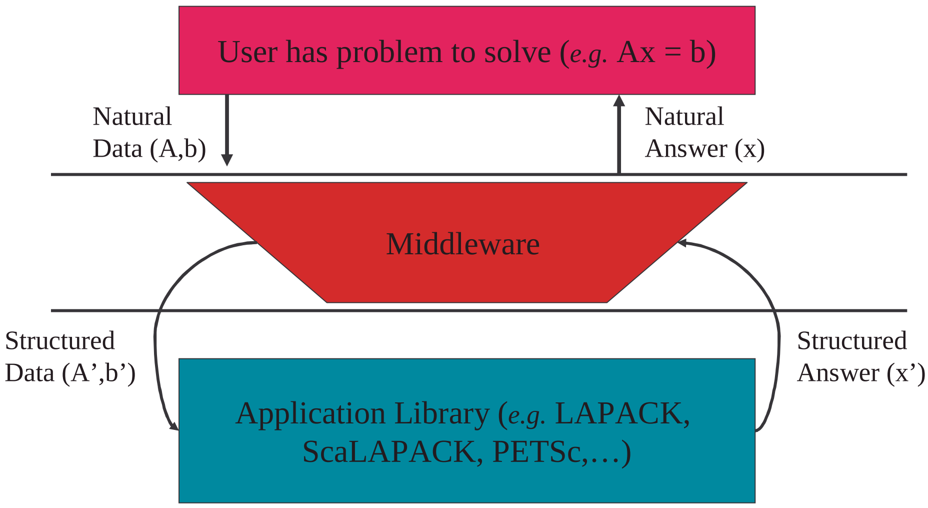
27

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
  - Determinate 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

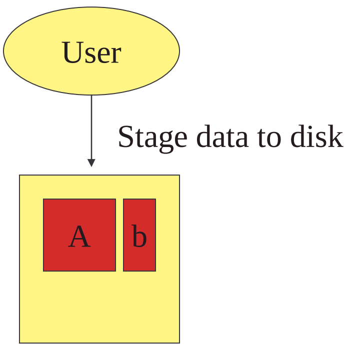
28

Big Picture…



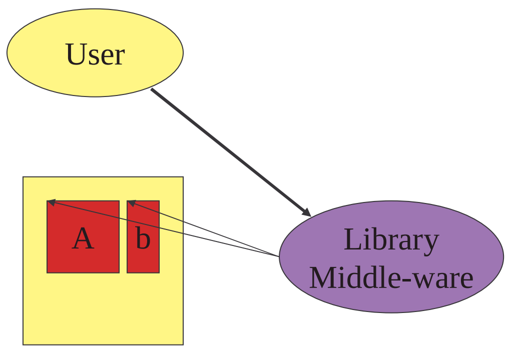
29

File System -based



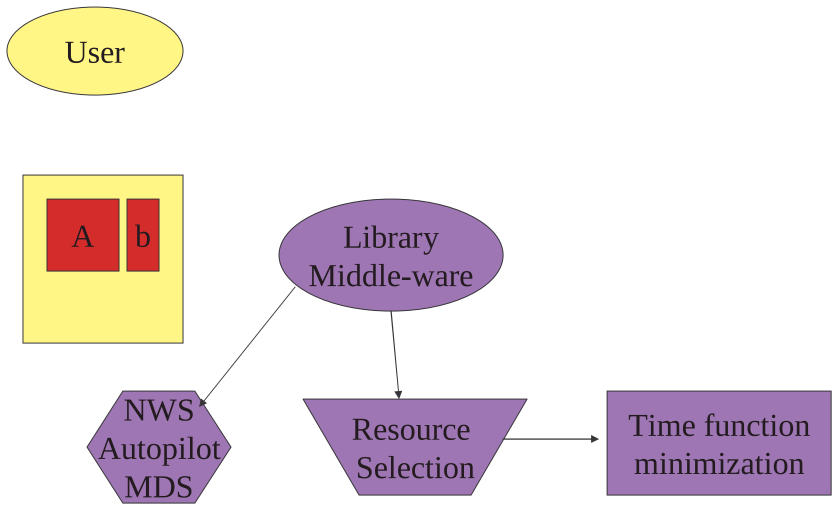
30

File System -based



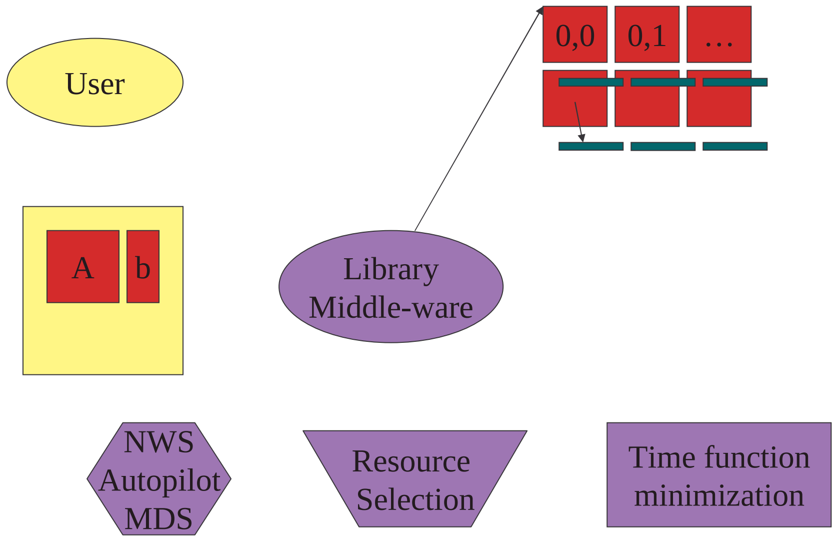
31

File System -based



32

File System -based



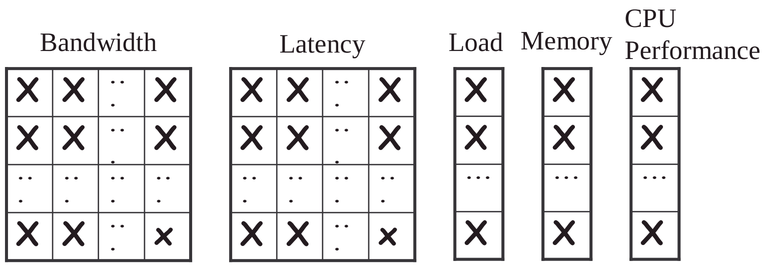
Can use Grid infrastructure, i.e.Globus/NWS, but doesn't have to.

33

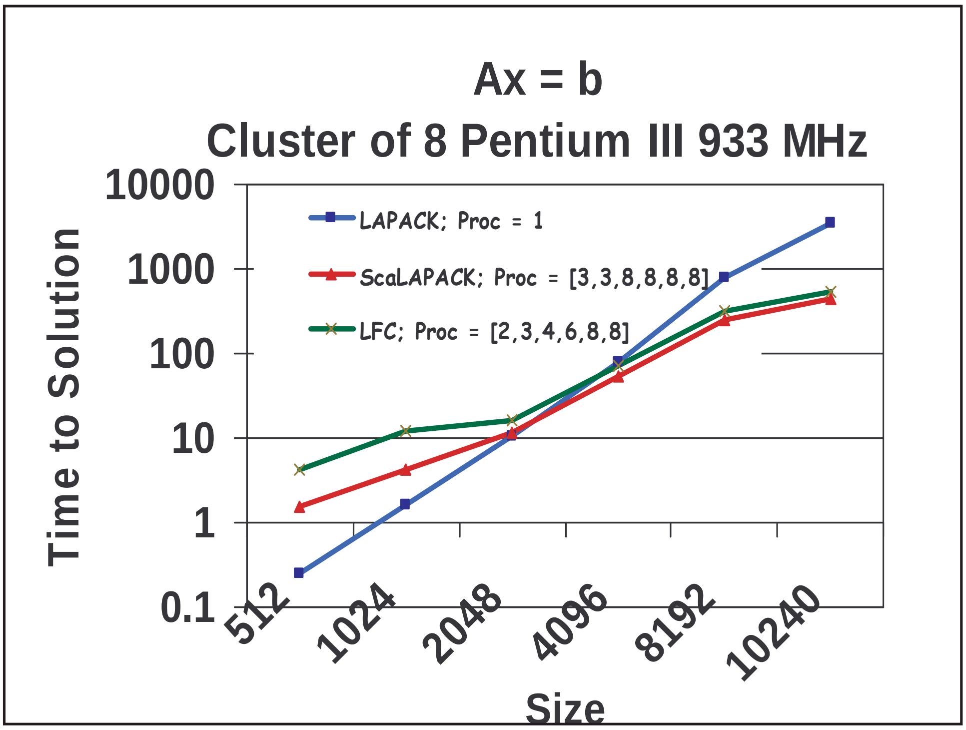
Resource Selector

NETWORK WEATHER SERVICE

- 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



34



LAPACK For Clusters (LFC)

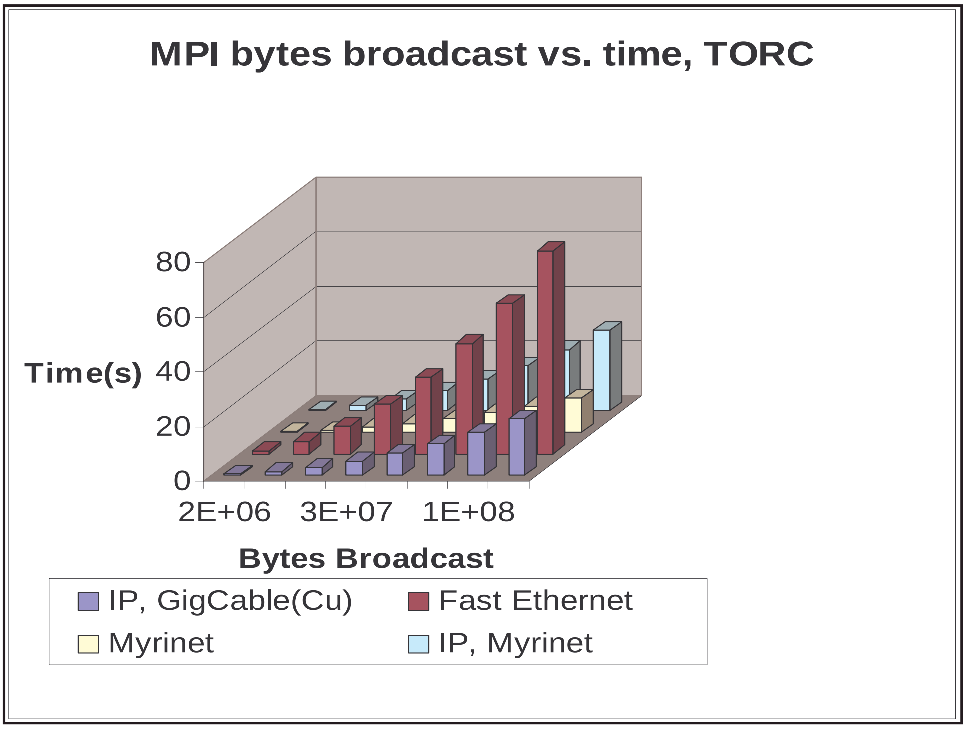
- LFC will automate much of the decisions in the Cluster environment to provide best time to solution.
  - Adaptivity to the dynamic environment.
  - As the complexities of the Clusters and Grid increase need to develop strategies for self adaptability.
  - Handcrafted developed leading to an automated 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

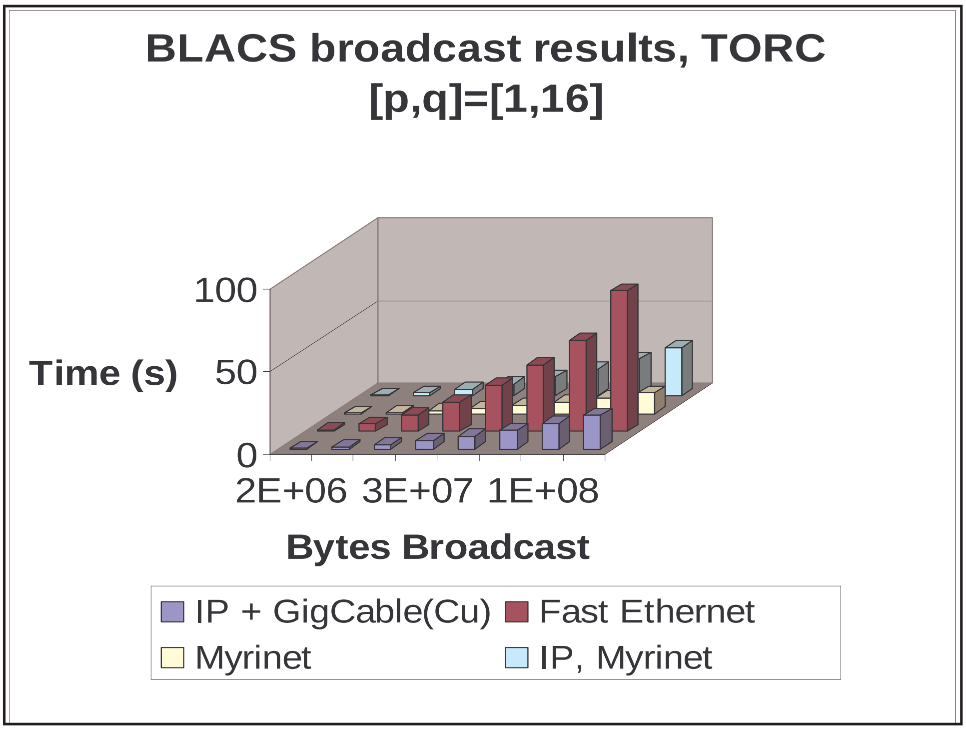
36

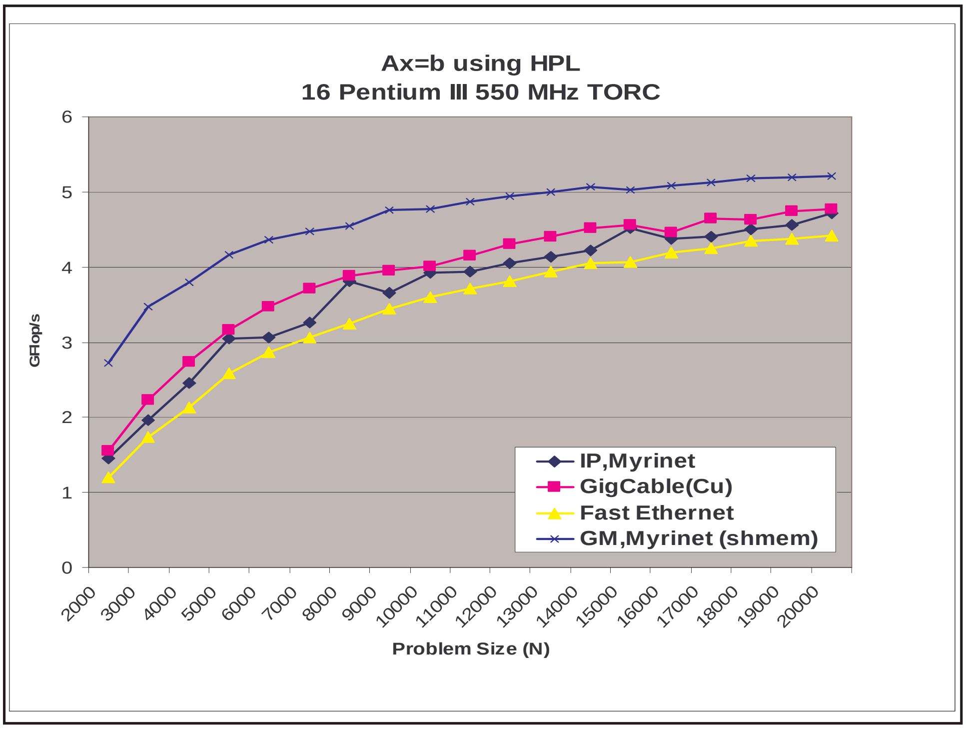
TORC Cluster

- 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

37




BLACS broadcast results, TORC
[p,q]=[1,16]
100
Time (s)
50
0
2E+06 3E+07 1E+08
Bytes Broadcast
IP + GigCable(Cu)
Fast Ethernet
Myrinet
IP, Myrinet


Ax=b using HPL
16 Pentium III 550 MHz TORC
GFlop/s
0
1
2
3
4
5
6
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
Problem Size (N)
IP,Myrinet
GigCable(Cu)
Fast Ethernet
GM,Myrinet (shmem)

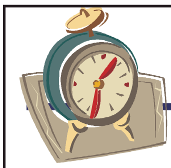
Tools for
Performance Evaluation

- 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

41

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

43

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)
  - (laderose@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

PAPI

44

Early Users of PAPI

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

45

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.dyninst.org/
- A work in progress…

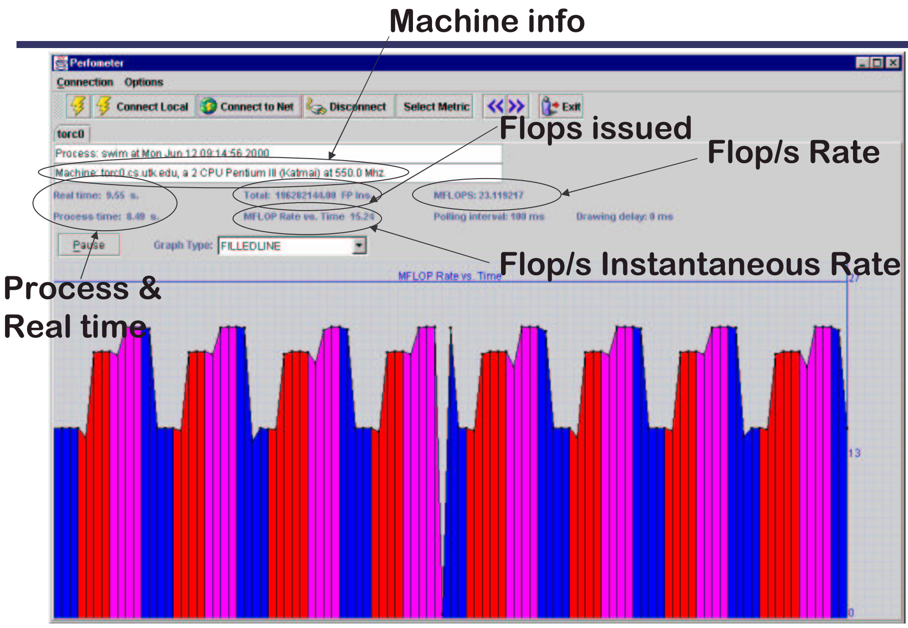
46

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

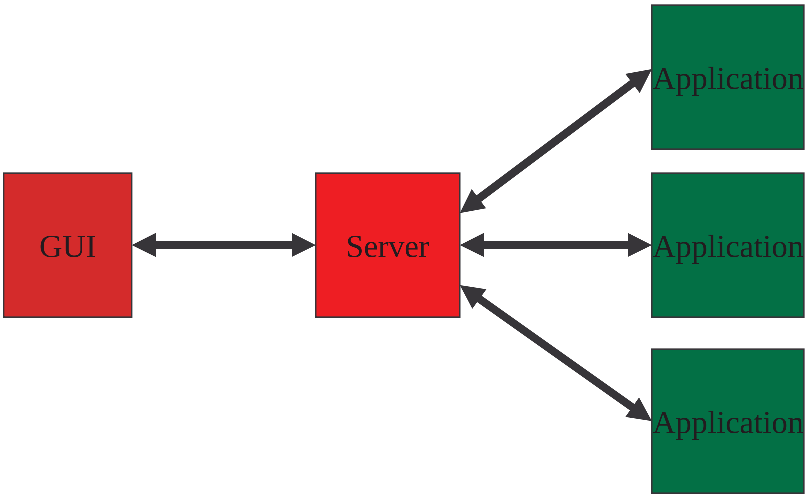
47

Perfometer/ DynaProf



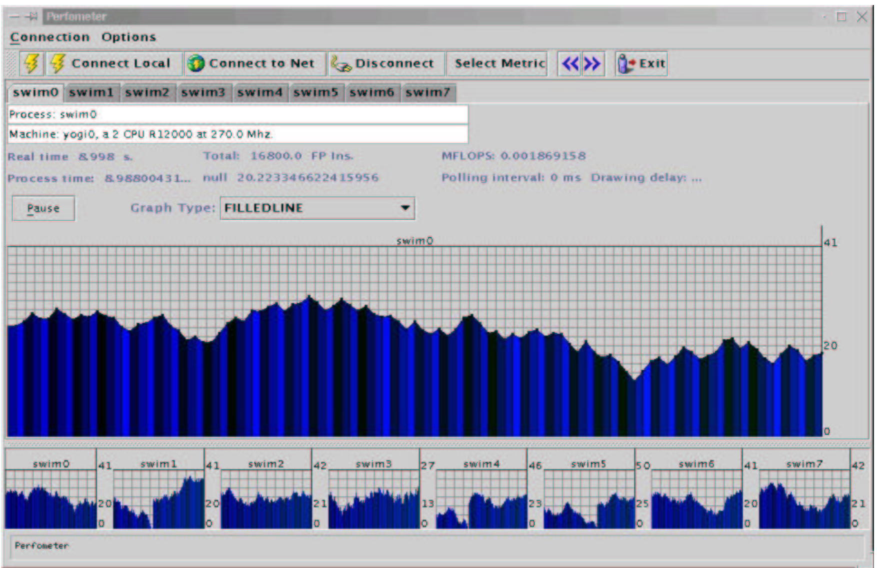
48

Next Version of Perfometer Implementation



49

PAPI's Parallel Interface



50

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, its 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.

51

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
    - http://icl.cs.utk.edu/atlas/
  - LFC
    - 5 drivers from ScaLAPACK by the end of summer
  - PAPI
    - http://icl.cs.utk.edu/papi/
  - DynaProf
    - tarball coming soon

52