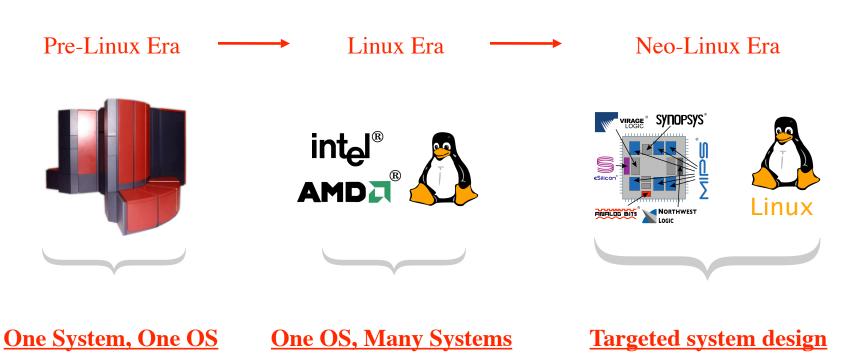


An Open Source Performance Tool Suite for Scientific Computing

Philip J. Mucci

June 26th, 2007 International Supercomputing Conference

Evolution of HPC Hardware/Software Design



New software stack for every new system design and workload

Stack adaptable to many different system designs and workloads

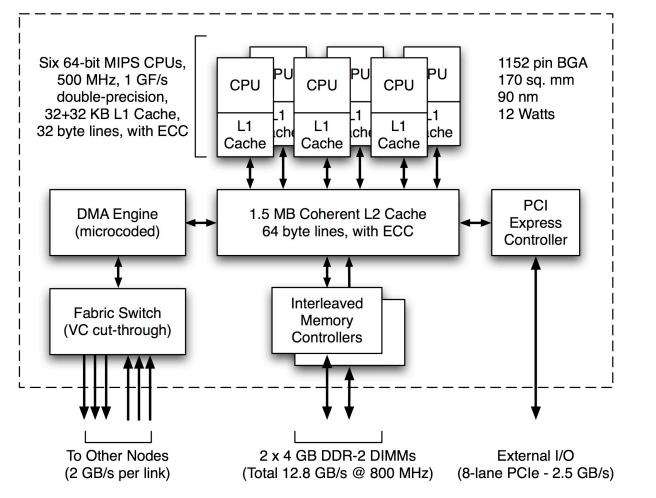
HPC workload on industry standard OS drives the system design



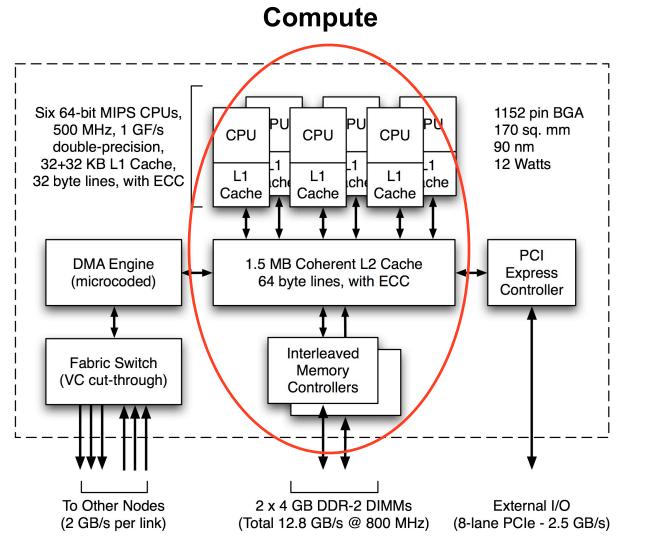
Guiding Principles

- Hardware
 - Run existing HPC applications faster
 - Per dollar, per watt, per square foot
 - Simplify: If apps don't need it, leave it out!
 - CPU should run fast enough to keep memory busy.
 - Put nodes as close together as possible.
 - Minimize power per node through SOC design.
- Software
 - Everything is Open Source
 - Offer a choice of support models, binary, custom.
 - Fully integrated operating environment
 - Modern version of Linux operating system and utilities.
 - Fast communication interface from MPI on down.
 - POSIX compliant parallel file system.
 - Compilers, libraries, debuggers, performance tools
 - Monitoring, configuration, resource management, updates

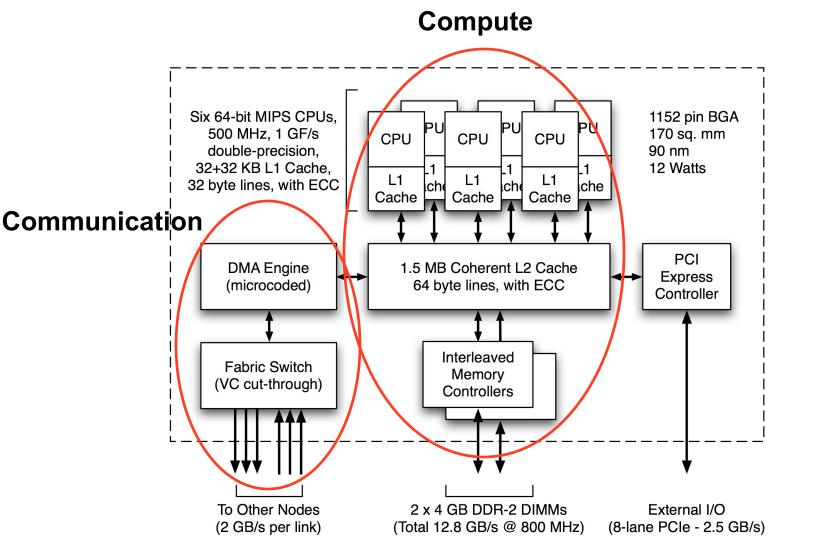






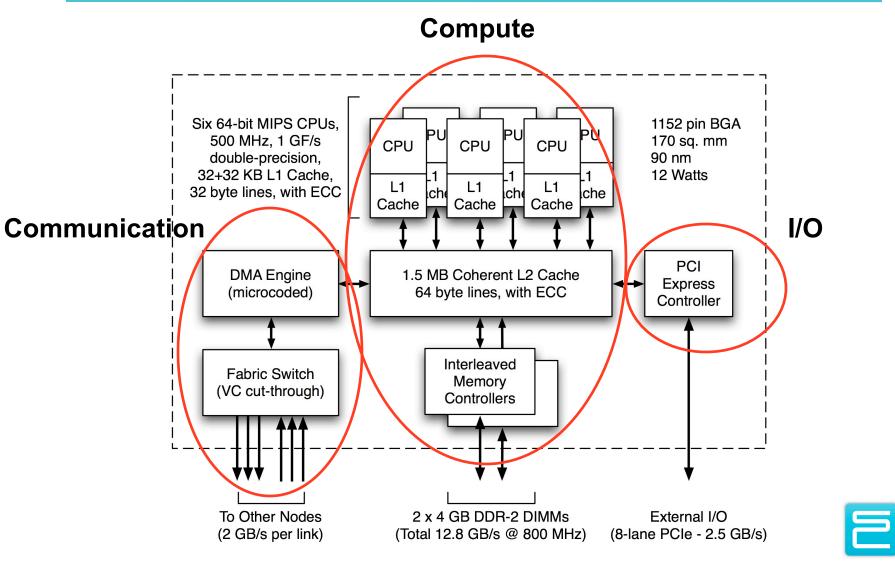




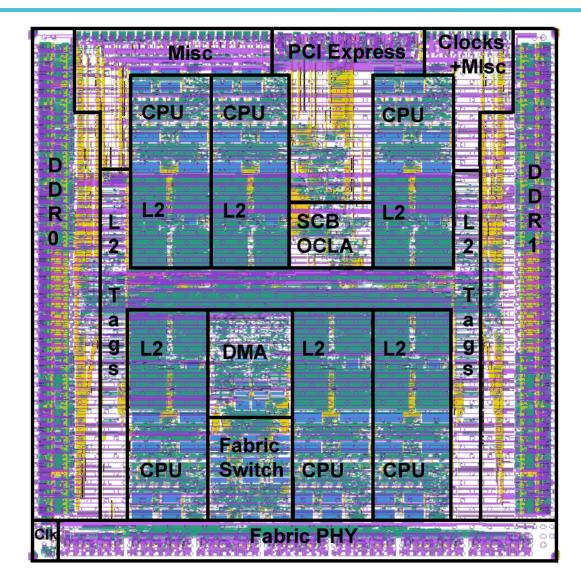


SiCortex

2

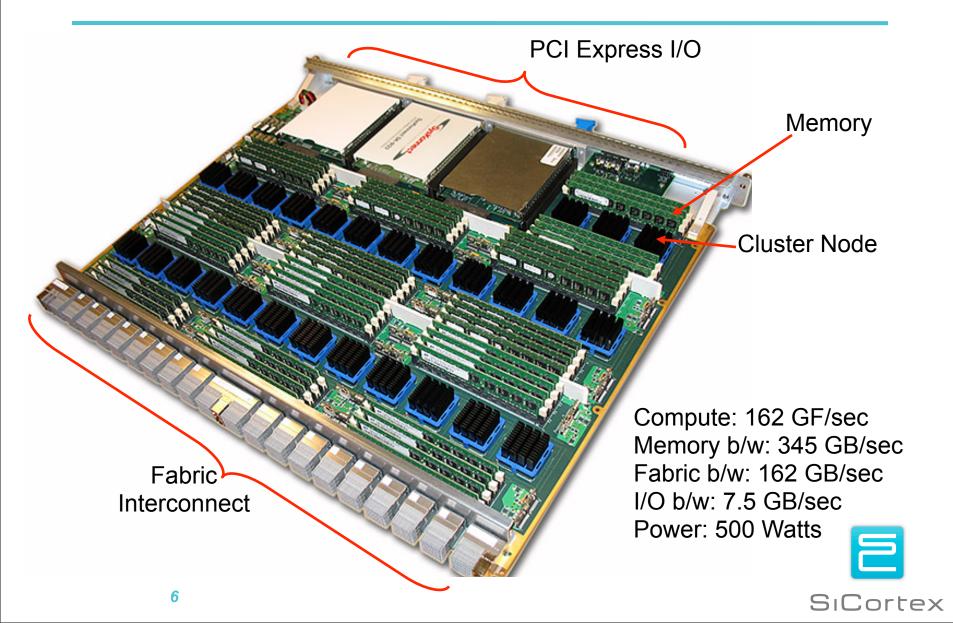


SICortex





SiCortex CPU Module



SC5832

- 5.8 Teraflops
- 7.7 Terabytes
- 500 GByte/s bisection bandwidth
 - 1 µs MPI latency
- 108 8-lane PCI Express
 - 18 KW (208v 3Ø 60A)
 - 1 Cabinet





Pervasive Monitoring across the Chip

- In-core
 - Instructions, cache hits/misses, ...
 - Stalls due to resources and conflicts (both for program tuning, and for next-generation architectural data)
- Off-core
 - L2 cache
 - DMA
 - Packets, memory transactions, mi
 - DDR
 - Transactions, bank hits, power downs, ...
 - Fabric Switch
 - Packets, stalls due to congestion, microcode activity, ...
 - PCI
 - Transactions



SiCortex Performance Monitoring Hardware

- MIPS64 architected PMU
 - 2 32-bit counters per core, 1 32-bit RTC
 - 4 counting domains (user, kernel, supervisor, interrupt)
 - Interrupt on overflow
 - 4 supplemental registers that get incremented by the SCB: 2 program counter, 2 effective address
- Off core: Serial Control Bus
 - 256 32-bit counters organized into 128 buckets of 2 counters.
 - Round-robin sampling of all buckets or direct measurement of 2.
 - Interrupt on overflow
 - Thresholding (true > n cycles, increment)
 - Pairwise conditional counting (IF-AND, IF-AND-NOT)



State of Linux Performance Tools

- Linux kernel does not contain any code to support profiling in production environments.*
 - Despite highly stable kernel patches being available for > 10 years on some platforms.
- No major commercial Linux distribution contains anything beyond OProfile and Gprof.*
- Vendors have developed some tools, but kept the code private.
 - Will the N.I.H. disease ever die?
 - Installation is complicated by support agreement with Linux distribution vendor regarding running an unpatched kernel.



* except IA64, which has kernel support and pfmon

'Productizing' Open Source

- What is the model really good at?
 - Innovation
 - Evolution
 - Distribution
 - Support*
 - Standardization*
- The bad news?
 - Specialization
 - Documentation
 - Verification
 - Integration

Focus resources where needed, drive that 'last mile'.

SICortex

Tools Strategy

- Leverage best-of-breed Open Source tools.
 - Foster relationships with original authors.
 - Propagate changes back to public source trees.
- Provide a 'drill-down' hierarchy
 - Follow a Unix-like philosophy. (Needs drive tools)
- Uniform user interface and semantics.
 - Observe linux standards. (LSB and beyond)
- Develop value added extensions and test engines.
- Guarantee full interoperability.
- Contract expertise where appropriate.



Evaluation of Workloads

- Characterization
 - Overall evaluation of performance
 - Isolate specific components for focus.
- Analysis and Optimization
 - Establish baseline performance data
 - Focus experimentation and optimization passes.
- Performance Development
 - Integration of robust performance evaluation
 - Regular performance regression testing



Selection Criteria for the Tools Suite

- Work on unmodified codes
- Quick and easy characterization of:
 - Hardware utilization (on and off-core)
 - Memory

14

- I/O
- Communication
- Thread/Task load balance
- Detailed analysis using sampling
- Simple instrumentation
- Adv. instrumentation and tracing
- Trace-based visualization
- Expert access to PMU and perfmon2



SICortex



The Perfmon2 Kernel Subsystem

- Lightweight:
 - Efficient code structure.
 - Lazy updates.
 - Buffered interrupts with sampling.
- Feature-rich:
 - System wide and per-thread counting.
 - First-person and third-person (attach) operation.
 - Kernel mode PMU multiplexing.
 - Flexible event sampling interface.
- Being considered for adoption (see LKML).
- Vendor supported.



Libpfm

- Portable, low-level library to perform counter setup.
 - Enforces register and event dependencies
 - Performs register allocation
 - Result is set of PMU control values that can be passed to the kernel.
- Not tied to Perfmon2
- Interface is only appropriate for tool designers, too low level for use in applications.
- PAPI uses this for counter setup where possible.



PAPI

- Ad-hoc standard library for the implementation of application performance analysis tools.
- 2 level API, high-level (apps) and low-level (tools)
- Provides first and third person semantics for 'thread-centric' counting and sampling based on PMU events.
- Handles the 'gory details' and allows one to focus on tool development.
- Portable: write once, run anywhere.



Monitor

- Library infrastructure to insert instrumentation at runtime on unmodified executables.
 - Uses library preloading and function overloading, does not edit the object on disk or in memory.
 - Provides callbacks to tools for relevant events, thread creation, destruction, library loading, fork/exec, etc...
- Code based on that originally developed by Rice University as part of HPCToolkit.
- Used in all the **Ex**periment tools
 - Command line tools that require no modifications to the source code



Performance **Ex**periment Tools

• Set of commands that provide the interface to the underlying performance monitoring tools.

- All are based on Monitor and PAPI

- papiex, mpipex, ioex, hpcex, gptlex, tauex
 - Easy to use as /bin/time
 - Generate concise text output where appropriate.
 - Take the same arguments, except for tool-specific options.
 - Provide standard and HTML man pages and documentation.



Papiex

- Used to obtain summary information about an application using PAPI and other metrics.
- Represents the first pass of application performance evaluation.
- It provides:
 - Memory footprint
 - Percent of time in I/O
 - Percent of time in MPI
 - PAPI, native and derived metrics
 - Provides per-thread, per-task and per-job summaries
 - Very basic instrumentation API.



Papiex: Workload Characterization

IPC 0.40 CPU Utilization 0.96 % Memory Instructions 39.02 % FP Instructions 33.38 % Branch Instructions 18.87 % Integer Instructions 66.62 Loads/Stores Ratio 18.14 L1 D-cache Hit % 97.22 L1 I-cache Hit % 100.00 D-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 11.41 Est. Stall % 7.79 Est. L1 D-cache Miss Stall % 0.02 Est. I-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
% Memory Instructions 39.02 % FP Instructions 33.38 % Branch Instructions 18.87 % Integer Instructions 66.62 Loads/Stores Ratio 18.14 L1 D-cache Hit % 97.22 L1 I-cache Hit % 100.00 D-TLB Hit % 87.43 I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 11.41 Est. Stall % 7.79 Est. L1 D-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
% FP Instructions 33.38 % Branch Instructions 18.87 % Integer Instructions 66.62 Loads/Stores Ratio 18.14 L1 D-cache Hit % 97.22 L1 I-cache Hit % 100.00 D-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 11.41 Est. Stall % 7.79 Est. L1 D-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
% Branch Instructions 18.87 % Integer Instructions 66.62 Loads/Stores Ratio 18.14 L1 D-cache Hit % 97.22 L1 I-cache Hit % 100.00 D-TLB Hit % 87.43 I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 11.41 Est. Stall % 7.79 Est. L1 D-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
% Integer Instructions 66.62 Loads/Stores Ratio 18.14 L1 D-cache Hit % 97.22 L1 I-cache Hit % 100.00 D-TLB Hit % 87.43 I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 11.41 Est. Stall % 7.79 Est. L1 D-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
Loads/Stores Ratio 18.14 L1 D-cache Hit % 97.22 L1 I-cache Hit % 100.00 D-TLB Hit % 87.43 I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. L1 I-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
L1 D-cache Hit % 97.22 L1 I-cache Hit % 100.00 D-TLB Hit % 87.43 I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
L1 I-cache Hit % 100.00 D-TLB Hit % 87.43 I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 14.47 Dual Issue % 11.41 Est. Stall % 7.79 Est. L1 D-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
D-TLB Hit % 87.43 I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 14.47 Dual Issue % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. L1 I-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
I-TLB Hit % 99.97 FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 14.47 Dual Issue % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
FP ins. per D-cache Miss 30.72 Computational Intensity 0.86 Branch Misprediction % 14.47 Dual Issue % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. L1 I-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
Computational Intensity 0.86 Branch Misprediction % 14.47 Dual Issue % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. L1 I-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
Branch Misprediction % 14.47 Dual Issue % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. L1 I-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
Dual Issue % 11.41 Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. L1 I-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
Est. Stall % 17.06 Est. L1 D-cache Miss Stall % 7.79 Est. L1 I-cache Miss Stall % 0.02 Est. D-TLB Miss Stall % 3.91 Est. I-TLB Miss Stall % 0.03	
Est. L1 D-cache Miss Stall %	
Est. L1 I-cache Miss Stall %	
Est. D-TLB Miss Stall %	
Est. I-TLB Miss Stall % 0.03	
Est. TLB Trap Stall % 0.00	
Est. Mispred. Branch Stall % 1.09	
Dependency Stall % 4.22	
T: Actual/Ideal Cycles 3.77	
T: Ideal (max dual) MFLIPS 250.55	
P: Actual/Ideal Cycles 2.83	
P: Ideal (curr dual) MFLIPS 188.41	
% MPI Cycles 18.49	
% I/O Cycles 0.02	



Mpipex

- Used to characterize the MPI performance of an application.
 - Uses mpiP from LLNL.
- It provides:
 - MPI load balance
 - MPI function profile
 - Message size distribution
 - Call site information: file, function and line



MPIPEX: Aggregate MPI Profile

@ Aggregate Time	(top tv	venty, desce	nding, r	nilliseco	nds) -
Call	Site	Time	App%	MPI%	cov
Barrier	29	9.65e+05	4.96	30.20	0.00
Barrier	18	6.1e+05	3.14	19.10	0.21
Allgather	12	3.68e+05	1.89	11.51	0.47
Barrier	43	3.25e+05	1.67	10.18	0.43
Sendrecv	78	2.2e+05	1.13	6.88	2.19
Sendrecv	21	1.57e+05	0.81	4.92	0.51



MPIPEX: Load Balance

				· —
@	MPI Time (se	conds)		-
Task	AppTime	MPITime	MPI%	
0	1.06e+03	79.8	7.53	
1	1.06e+03	89.9	8.47	
2	1.06e+03	85.2	8.03	
3	1.06e+03	85.8	8.09	
4	1.06e+03	85.1	8.03	
5	1.06e+03	111	10.42	
6	1.06e+03	144	13.54	
7	1.06e+03	142	13.37	
8	1.06e+03	139	13.12	
9	1.06e+03	147	13.85	
10	1.06e+03	140	13.16	
11	1.06e+03	141	13.33	
12	1.06e+03	143	13.47	
13	1.06e+03	138	13.03	
14	1.06e+03	144	13.55	
15	1.06e+03	182	17.19	
*	1.7e+04	2e+03	11.76	



loex

- Used to characterize the I/O performance of an application.
 - Based on concepts from IOtrack written at PDC/KTH.
- Per-file statistics:
 - Flags
 - Access type
 - Bandwidth
 - Chunk size
 - Time spent



loex: Per-file profile

File: /dev/zero	
open64	
calls	: 1
read	
calls	: 10
usecs	: 587
usecs/call	: 58
bytes	: 10485760
bytes/call	: 1048576
MB/s	: 17863
File: /home/out	
open64	
calls	: 1
flags	: O_WRONLY O_CREAT O_TRUNC
write	
calls	: 10
usecs	: 157444
usecs/call	: 15744
bytes	: 10485760
bytes/call	: 1048576
MB/s	: 66



- Used to produce statistical profiles without instrumentation.
 - Based on HPCToolkit from Rice University.
- Take interrupts when a counter overflows a certain threshold.
 - i.e. every 10000 cache misses, interrupt/sample the PC.
 - Supports multiple simultaneous profiles
- Data is viewed with hpcprof (text) and hpcviewer (Java GUI)
 - Advanced source code correlation and visualization through bloop (a binary analyzer) and hpcviewer.
- Profile by load module, file, function, line and even instruction.



Hpcprof: Hotspot analyses

Columns correspond to the following events [event:period (events/sample)] PAPI_TOT_CYC:999999 - Total cycles (2553 samples)

Load Module Summary: 65.5% testconv2d 34.5% /lib64/libc-2.5.so

File Summary:

- 36.9% <<testconv2d>>/home/phil/ISC/new/convolution/simplest_conv.c
- 34.5% <</lib64/libc-2.5.so>><unknown>
- 10.0% <<testconv2d>>/home/phil/ISC/new/convolution/support.c
- 9.8% <<testconv2d>>/home/phil/ISC/new/convolution/testconv2d.c
- 8.8% <<testconv2d>>/home/phil/ISC/new/convolution/convCore.c

Function Summary:

- 36.9% <<testconv2d>>conv2d_simple
- 17.0% <</lib64/libc-2.5.so>>random
- 12.9% <</lib64/libc-2.5.so>>random_r
- 10.0% <<testconv2d>>makeRandomDouble
- 9.8% <<testconv2d>>main
- 8.8% <<testconv2d>>conv2dBy3TileZero
- 4.6% <</lib64/libc-2.5.so>>rand

Line Summary:

- 34.5% <</lib64/libc-2.5.so>><unknown>:0
- 26.1% <<testconv2d>>/home/phil/ISC/new/convolution/simplest_conv.c:27
- 6.5% <<testconv2d>>/home/phil/ISC/new/convolution/simplest_conv.c:24



Hpcprof: Source code annotation

```
19
     0.8%
              for (j = coff; j < nca-coff; j++)
20
                 Ł
21
                out = 0.0;
     0.1%
22
     2.5%
                for (ki = 0; ki < nrk; ki++)
23
24
     6.5%
                   for (kj = 0; kj < nck; kj++)
25
                     {
26
                     // out += a[i+ki][j+kj] * k[ki][kj];
                    out += *(a+(i+ki-roff)*nca + j+kj-coff) * *(k+(ki*nck)+kj);
27
    26.1%
28
                     }
29
                   }
30
                // c[i+roff][j+coff] = out;
31
     1.0%
                *(c+(i)*nca + j) = out;
32
                 }
33
              }
```



Hpcprof: Assembly annotation

0x1200068c0:	0.01%	move	v0,v1
0x1200068c4:	0.06%	daddu	a0,a2,v0
0x1200068c8:	0.60%	dsll	a1,a0,0x3
0x1200068cc:	5.48%	ld	v0,48(s8)
0x1200068d0:	0.01%	daddu	v1,a1,v0
0x1200068d4:	4.18%	ldc1	\$f0,0(v1)
0x1200068d8:		mul.d	\$f2,\$f3,\$f0
0x1200068dc:	0.03%	ldc1	\$f1,8(s8)
0x1200068e0:		add.d	\$f0,\$f1,\$f2
0x1200068e4:	0.04%	sdc1	\$f0,8(s8)
0x1200068e8:	5.04%	lw	v0,16(s8)
0x1200068ec:	0.01%	addiu	v1,v0,1
0x1200068f0:	6.60%	SW	v1,16(s8)
0x1200068f4:	7.80%	lw	v0,16(s8)
0x1200068f8:	0.02%	lw	v1,60(s8)
0x1200068fc:	0.03%	slt	a0,v0,v1
0x120006900:	0.02%	bnez	a0,0x12000683c



Hpcviewer: Loop-level profiling

n_periodic.c			
		D_yy) * dimz + slab_zj += P[i].Mass * (dx) * dy * (1.0 - dz); D_y) * dimz + slab_zz] += P[i].Mass * (dx) * (1.0 - dy) * dz;	
		$p_{y} = dinz + slab_{zz} + = P[i].Mass + (dx) + (1.0 - dy) - dz,$ $p_{y} = dinz + slab_{zz} + = P[i].Mass + (dx) + dy + dz;$	
280 }	inspace[[slab_XX dilly + slab	$(a_{x}) = (a_{x})^{-1} (a_{x}$	
281			•
282			•
283 for(i	= 0; i < fftsize; i++) /* clear	r local density field */	
284 rhog	grid[i] = 0;		
285			
	evel = 0; level < (1 << PTask); le	evel++) /* note: for level=0, target is the same task */	
287 {	int i with with		
	ndTask = ThisTask;		
	cvTask = ThisTask ^ level; recvTask < NTask)		
290 11(1	recviask < Niask)		
292	/* check how much v	we have to send */	
293	sendmin = 2 * PMGRI		
294	condmax1	,	Ψ.
7 444			
7.44			A 4
7.04		Flat View	
	opes 😥 🔐	Flat View	
Sco			
Sco Experiment	Aggregate Metrics	PAPI_TOT_C V	
Sco Experiment	Aggregate Metrics rcetree.c: 1496–1728	PAPI_TOT_C V 1.95e12 100.0	
Sco Experiment / Ioop at for Load mode	Aggregate Metrics rcetree.c: 1496-1728 ule /lib64/libm-2.5.so	PAPI_TOT_C ▼ 1.95e12 100.0 9.11e11 46.6% 6.41e11 32.8%	
Sco Experiment / Ioop at for Load mode Ioop at pr	Aggregate Metrics rcetree.c: 1496-1728 ule /lib64/libm-2.5.so n_periodic.c: 590-671	PAPI_TOT_C ▼ 1.95e12 100.0 9.11e11 46.6% 6.41e11 32.8% 4.57e10 2.3%	
Sco Experiment A loop at for Load mode loop at pm loop at pe	Aggregate Metrics rcetree.c: 1496–1728 ule /lib64/libm-2.5.so n_periodic.c: 590–671 eano.c: 276–300	PAPI_TOT_C▼ 1.95e12 100.0 9.11e11 46.6% 6.41e11 32.8% 4.57e10 2.3% 1.73e10 0.9%	
Sco Experiment / loop at for Load mode loop at pr loop at pe Load mode	Aggregate Metrics rcetree.c: 1496–1728 ule /lib64/libm–2.5.so n_periodic.c: 590–671 eano.c: 276–300 ule /usr/lib64/libscmpi_optimized	PAPI_TOT_C ▼ 1.95e12 100.0 9.11e11 46.6% 6.41e11 32.8% 4.57e10 2.3% 1.73e10 0.9% 1.26e10 0.6%	
Sco Experiment loop at for Load mode loop at pr loop at pe Load mode Load mode	Aggregate Metrics rcetree.c: 1496-1728 ule /lib64/libm-2.5.so n_periodic.c: 590-671 eano.c: 276-300 ule /usr/lib64/libscmpi_optimized ule /lib64/libc-2.5.so	PAPI_TOT_C V 1.95e12 100.0 9.11e11 46.68 6.41e11 32.88 4.57e10 2.38 1.73e10 0.98 1.26e10 0.68 8.89e09 0.58	
Sco Experiment / loop at for Load modi loop at pre Load modi Load modi loop at pr	Aggregate Metrics rcetree.c: 1496-1728 ule /lib64/libm-2.5.so n_periodic.c: 590-671 ano.c: 276-300 ule /usr/lib64/libscmpi_optimized ule /lib64/libc-2.5.so n_periodic.c: 248-279	PAPI_TOT_C V 1.95e12 100.0 9.11e11 46.6% 6.41e11 32.8% 4.57e10 2.3% 1.73e10 0.9% 1.26e10 0.6% 8.89e09 0.5% 5.64e09 0.3%	
Sco Experiment / loop at for Load modi loop at pre Load modi Load modi loop at pr	Aggregate Metrics rcetree.c: 1496-1728 ule /lib64/libm-2.5.so n_periodic.c: 590-671 eano.c: 276-300 ule /usr/lib64/libscmpi_optimized ule /lib64/libc-2.5.so	PAPI_TOT_C V 1.95e12 100.0 9.11e11 46.68 6.41e11 32.88 4.57e10 2.38 1.73e10 0.98 1.26e10 0.68 8.89e09 0.58	

SICortex

Ξ

Pfmon

- Used to perform highly focused instrumentation and/or advanced sampling.
 - Uses libpfm and the Perfmon2 kernel subsystem directly.
- Per-thread, per-CPU, system-wide sampling and counting.
- Allows one to attach to a running code.
- Limited but highly accurate instrumentation with software breakpoints.

- Works with static binaries.



Gptlex

- Used to control the behavior of GPTL performance system on instrumented and uninstrumented executables.
- Previously, GPTL options were hard-coded in the instrumentation.
 - Now, all options can be changed at run-time.
- Adds support for automatic compiler instrumentation using hooks in the GCC and Pathscale compilers.



GPTL

- Used to easily instrument applications for the generation of performance data.
 - Developed at NCAR for inclusion into their applications.
- Optimized for usability.
- Provides access to timers as well as PAPI events.
- Thread-safe and per-thread statistics.
- Provides estimates of overhead.
- Call-tree generation.
- Preserves parent/child relationships.



TAU Parallel Performance System



- Parallel Performance Evaluation Tool for Fortran, C, C++, Python and Java
- Used for in-depth performance studies of an application throughout its lifecycle.
- Supports Parallel Profiling
 - Flat, callpath, and phase based profiling
 - PerfDMF performance database and PerfExplorer cross experiment analysis tool
 - PAPI counters (one or more), wallclock time, CPU time
- Supports Event Tracing
 - Generates TAU binary traces in OTF (Open Trace Format, VampirTrace) or Epilog(KOJAK).
 - Supports Memory and PAPI counters in trace files with synchronized time stamps.

TAU Parallel Performance System



- Multi-level instrumentation
 - Source code (manual), pre-processor (Program Database Toolkit, PDT), MPI library
 - Memory, I/O instrumentation in Fortran and C/C++
 - Supports runtime throttling, selective instrumentation at routine and loop level.
- Widely-ported parallel performance profiling system.
 - All HPC systems, compilers, MPI-1 and 2 implementations, OpenMP and pthreads.



Tauex

- Used to control the behavior of the TAU performance system on instrumented and uninstrumented executables.
- Previously, TAU required extensive setup and relinking when options changed.
 - Now, all TAU options can be changed at run-time.

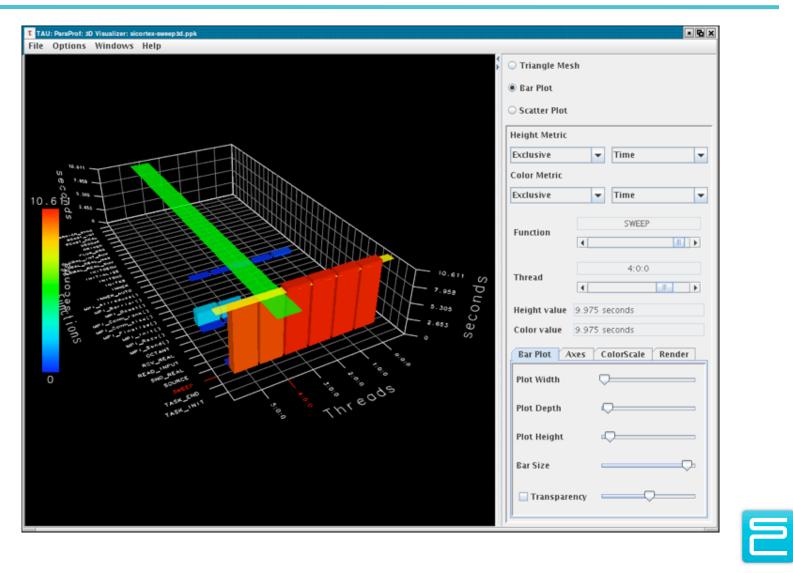


TAUEX: Paraprof Function Profile

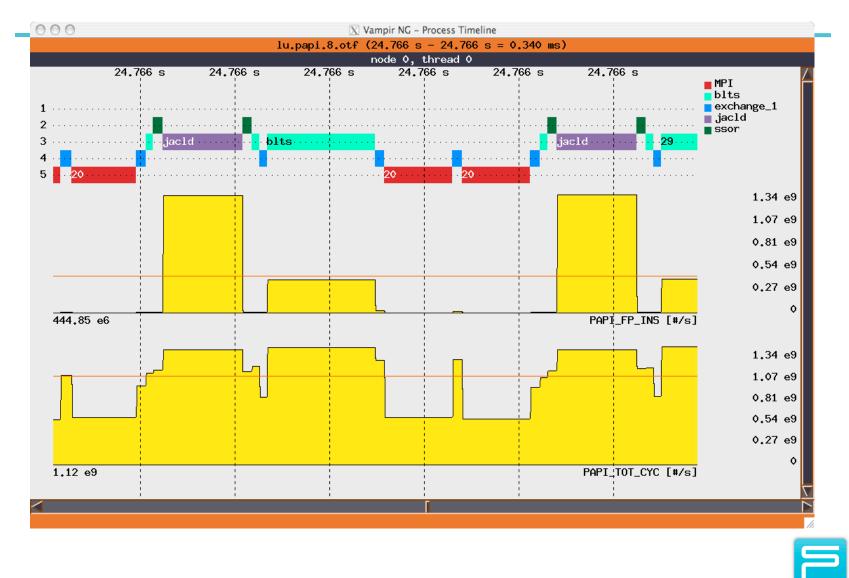
	🔨 TAU: ParaProf: Mean Data - sicortex-sweep3d.ppk
	File Options Windows Help
	Metric: Time Value: Exclusive Units: seconds
	10.359 SWEEP 1.524 MPI_Recv() 0.22 MPI_Alreduce() 0.24 MPI_Alreduce()
TAU: ParaProf: sicortex-sweep3d.ppk	0.219 MPI_Send() 0.128 SOURCE
TAU: ParaPror: sicorrex-sweep3d.ppk File Options Windows Help	0.082 MPI_Init() 0.049 MPI_Barrier()
letric: Time alue: Exclusive std. dev. mean n,c,t 0,0,0 n,c,t 1,0,0 n,c,t 2,0,0 n,c,t 3,0,0 n,c,t 3,0,0 n,c,t 5,0,0	0.043 FLUX_ERR 0.039 MPI_Bcast() 0.028 SND_REAL 0.022 RCV_REAL 0.005 DRIVER 0.005 INNER 0.004 INITIALIZE 0.004 INITIALIZE 0.004 INITXS 0.004 INITXS 0.004 INITXS 0.004 INITXS 0.004 INITXS 0.002 TASK_INIT 6.9E4 READ_INPUT 5.1E-4 BARRIER_SYNC 4.4E-4 OCTANT 3.6E-4 INITSNC 2.1E-4 GLOBAL_REAL_MAX 2.0E-4 GLOBAL_REAL_SUM
	1.7E-5 INNER_AUTO 9.7E-6 TASK_END 8.2E-6 BCAST_INT 5.8E-6 BCAST_REAL 3.5E-6 INITGEOM
	2.2E-6 DECOMP 2.2E-6 DECOMP 0 MPI_Comm_rank() 0 MPI_Comm_size()



TAUEX: ParaProf 3D Profile



Visualizing TAU Traces with VampirNG





Vampir

- Used to visualize temporal performance data (traces)
- 3 Components
 - VampirTrace, can be invoked from TAU or directly
 - VampirServer
 - VampirServer Browser



VampirTrace

- Recorded events
 - Function entry/exit if compiler instrumentation is used.
 - MPI and OpenMP events
 - Hardware/software performance counters (e.g. PAPI)
 - OS events: Process creation, resource management
- Collected event properties
 - Time stamp
 - Location (process / thread / MPI)
 - MPI specifics like message size etc.
- Generates data in Open Trace Format (OTF)
 - Human readable
 - Fast searching and indexing
 - On-the-fly compression

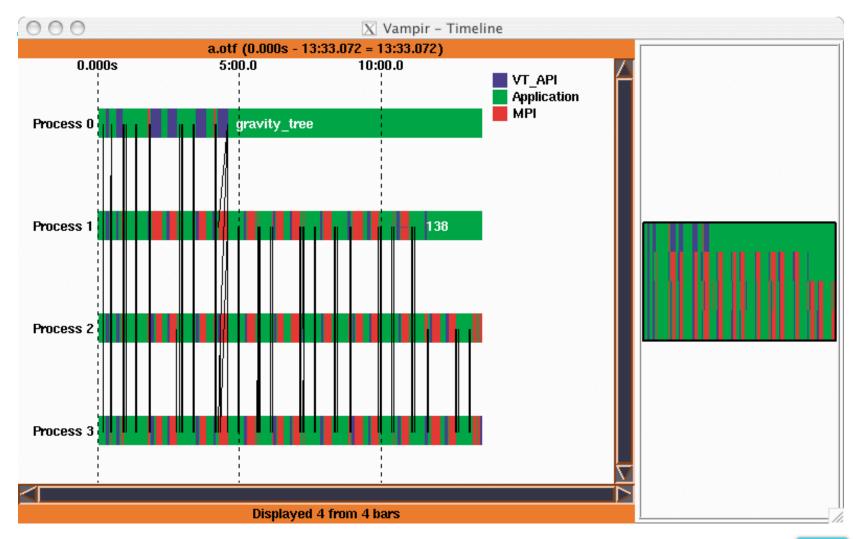


VampirServer

- VampirServer: Distributed high-end performance visualization
 - Client/server architecture
 - Parallel event processing
 - Runs on a (part of a) production environment
 - No need to transfer huge traces, uses parallel I/O
- VampirServer Browser: Lightweight client on local workstation
 - Outer appearance identical to Vampir
 - Highly scalable display engine
 - Statistics, profiles and summary charts
 - Message traffic and timelines
 - Receives visual content only
 - Already adapted to display resolution (but no images)
 - Moderate network bandwidth and latency requirements
 - Scales to trace data volumes > 40GB

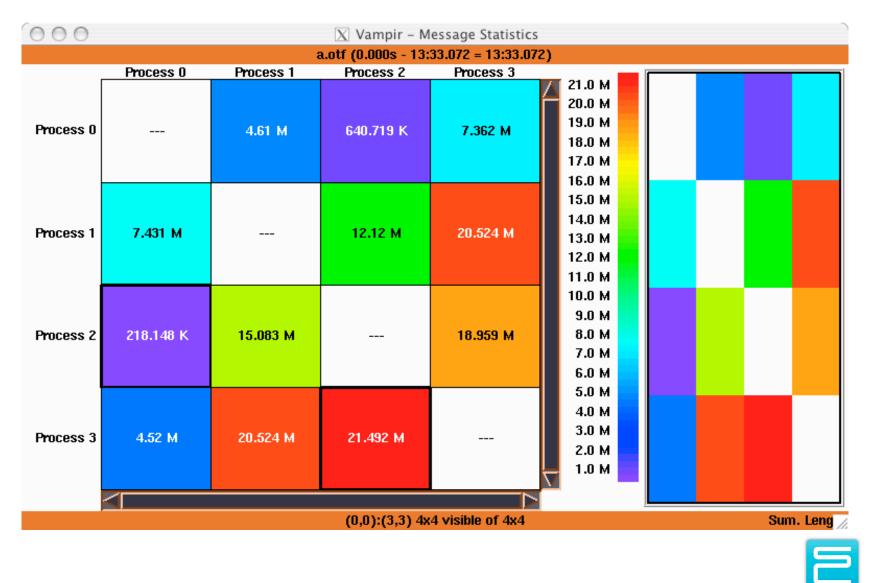


Vampir Timeline





Vampir Message Statistics

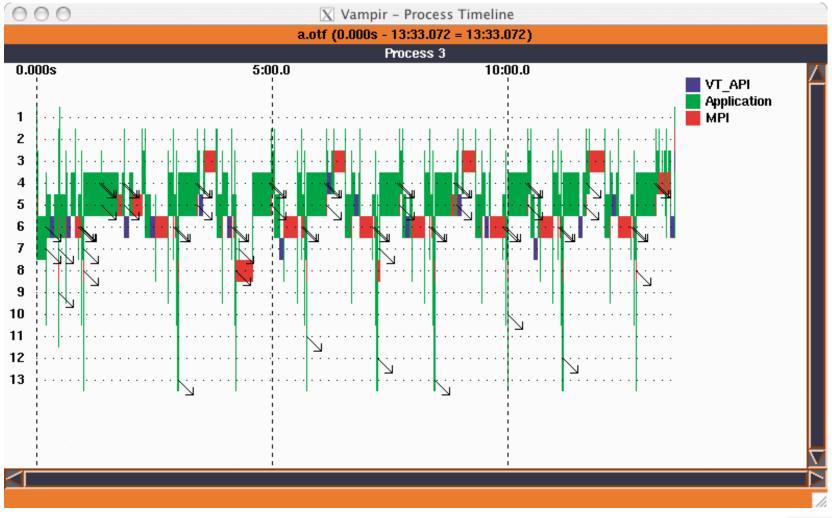




Vampir Summary Chart

MPI_Bcast 3:48.629 grav_tree_compare_key 2:37.615 pmforce_periodic 1:51.443 domain_determineTopTree 1:48.009 peano_hilbert_order 1:43.731 compare_key 1:37.059 domain_compare_key 1:35.808 peano_hilbert_exey 1:35.808 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 21.829 s get_timestep 10.074 s compute_glof_system [133] 5.784 s mort_table 2.937 s MPI_Altoall 2.289 s		0.000s	1:40.0	<mark>s - 13:33.07:</mark> 3:20.0	5:00.0	6:40.0	8:20.0	10:00.0	
MPI_Aligather 6:29.938 tracing 5:44.768 MPI_Bcast 5:44.768 grav_tree_compare_key 2:37.615 pmforce_periodic 1:51.443 domain_determineTopTree 1:48.005 peano_hilbert_order 1:48.005 compare_key 1:37.059 domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 domain_compare_key 1:32.095 domain_find_timesteps 43.717 s MPI_Santrer 53.190 s get_gravkick_factor 23.750 s get_fixed_ractor 23.511 s get_fixed_factor 21.829 s force_update_node_recursive 12.095 s get_imestep 10.074 s compute glof system [133] 5.784 s sign 3.390 s init_drit_table 2.937 s MPI_Alitoal 2.269 s domain_sumCost 2.173 s domain_topsplit_local 1.506 s									-
tracing 5:44.788 MPI_Bcast 2:37.515 grav_tree_compare_key 2:37.515 penforce_periodic 1:51.443 domain_determineTopTree 1:48.009 peano_hilbert_order 1:348.009 grav_tree_compare_key 1:35.008 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 21.829 s get_gravkick_factor 21.829 s force_treebuild_single 12.095 s get_imestep 10.074 s compute_glof_system [133] 5.784 s mestart 3.390 s init_drift_table 2.937 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s							9:36	.54	
MPI_Bcast 3:48.629 grav_tree_compare_key 2:37.615 pmforce_periodic 1:51.443 domain_determineTopTree 1:48.004 peano_hilbert_order 1:48.004 compare_key 1:37.059 domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s' get_gravkick_factor 23.511 s' get_invglokick factor 21.829 s get_invglokick factor 21.829 s compute_glof_system [133] 5.784 s gravkick_fact 2.937 s MPI_Altoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s						9.938			
grav_tree_compare_key 2:37.615 pmforce_periodic 1:51.443 domain_determineTopTree 1:48.004 peano_hilbert_order 1:43.731 compare_key 1:37.059 domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_hydrokick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s int_drift_table 2.937 s MPI_Alltoal 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s					5:44.788				
pmforce_periodic 1:51.443 domain_determineTopTree 1:48.005 peano_hilbert_order 1:43.731 compare_key 1:37.059 domain_compare_key 1:35.806 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 21.829 s force_treebuild_single 12.095 s get_finestep 10.074 s compute_glof_system [133] 5.784 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									
domain_determineTopTree 1:48.00\$ peano_hilbert_order 1:43.731 compare_key 1:37.059 domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_gravkick_factor 23.511 s get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.397 s MPI_Alttoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s				5 - A - A - A - A - A - A - A - A - A -					1.1
peano_hilbert_order 1:43.731 compare_key 1:37.053 domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_fravkick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.937 s MPI_Alitoall 2.269 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									
compare_key 1:37.059 domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_hydrokick_factor 21.829 s force_treebuild_single 112.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.937 s MPI_Alitoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									- j -
domain_compare_key 1:35.808 peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_hydrokick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s init_drift_table 2.937 s MPI_Altoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									- j -
peano_hilbert_key 1:22.095 MPI_Sendrecv 1:06.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_hydrokick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									ų.
MPI_Sendrecv 106.23 advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_hydrokick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									1.
advance_and_find_timesteps 43.717 s MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_hydrokick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.937 s MPI_Alltoall 2.269 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									۰H .
MPI_Barrier 37.190 s force_update_node_recursive 23.750 s get_gravkick_factor 23.511 s get_hydrokick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									
force_update_node_recursive23.750 sget_gravkick_factor23.511 sget_hydrokick_factor21.829 sforce_treebuild_single12.095 sget_timestep10.074 scompute_glof_system [133]5.784 srestart3.390 sinit_drift_table2.937 sMPI_Alltoall2.269 sdomain_sumCost2.173 sdomain_topsplit_local1.508 s									1
get_gravkick_factor23.511 sget_hydrokick_factor21.829 sforce_treebuild_single12.095 sget_timestep10.074 scompute_glof_system [133]5.784 srestart3.390 sinit_drift_table2.937 sMPI_Alltoall2.289 sdomain_sumCost2.173 sdomain_topsplit_local1.508 s									1
get_hydrokick_factor 21.829 s force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									÷ į –
force_treebuild_single 12.095 s get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									. j 1
get_timestep 10.074 s compute_glof_system [133] 5.784 s restart 3.390 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									1
compute_glof_system [133]5.784 srestart3.390 sinit_drift_table2.937 sMPI_Alitoall2.289 sdomain_sumCost2.173 sdomain_topsplit_local1.508 s									
init_drift_table 3.390 s init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									
init_drift_table 2.937 s MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									
MPI_Alltoall 2.289 s domain_sumCost 2.173 s domain_topsplit_local 1.508 s									-
domain_sumCost 2.173 s									1 i -
domain_topsplit_local 1.508 s ' ' ' ' ' ' ' ' ' '									1 j -
Displayed 26 from 106 bars All Symbols: Evolusive Ti	domain_topsplit_local	1.508	s '		· · · ·	, . .			1
Displayed 26 from 106 bars All Symbols: Evolusive Ti									
			Displa	ved 26 from	106 bars		AIL S	vmhols: Exclusive	e Ti

Vampir Process Timeline



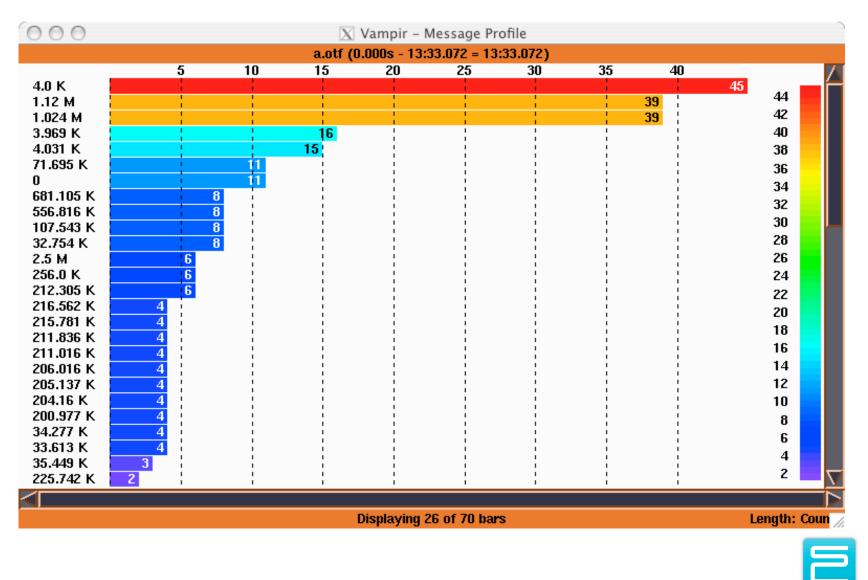


Vampir Call Tree

Call Tree	Counts <u></u>	Tin	Search
👻 main	1	0.000s0.612 n	Ĭ
🖕 🔻 MPI_Finalize	01	0.000s0.120 s	Å.
📖 🛛 tracing	01	0.000s12.999	Advanced Search
	1	0.168 s0.268 :	
• tracing	1	14.999 ms33.	Folding
🖕 🛨 begrun	1	1.000 ms13.9	Fold Level: 73
• allocate_commbuffers	1	4.000 ms6.00	Fold Level: Marcola 3
···· • find_next_outputtime	1	0.000s	Fold All
init ⊕… ▶ init	1	50.368 ms50.	Unfold All
	1	0.705 s0.770 :	
⊕… ▶ long_range_init	1	0.979 ms1.00	
ilobal Call Breakdown (main (235)) ———			
Caller			Callee
	Mi	위_Finalize 위_Init n nediff	



Vampir Message Profile





Additional Software

- EPILOG
 - Trace library from the KOJAK suite
- OProfile
 - Ported to use the Perfmon2 kernel infrastructure
- Other quality software not included:
 - OpenSpeedShop: LANL
 - PerfSuite: NCSA
 - ParaVer: BSC
 - EXPERT/CUBE from KOJAK: Juelich
 - DynInstAPI: Wisconsin



Summary

- ~1.5 man-years of effort has produced a leading tool suite where none existed.
 - Open Source can truly mean standing on the shoulders of giants.
- Continued success and R.O.I gained by following through on the strategy.
 - Integration and cooperation lowers support cost



Acknowledgements

- Center for Information Services and HPC, Technische Universität Dresden, Germany.
- ParaTools, Inc.
- Innovative Computing Laboratory, University of Tennessee, Knoxville.
- Lawrence Livermore National Laboratory.
- HiPerSoft, Rice University.
- National Center for Atmospheric Research.
- Stefane Eranian of HP Laboratories.
- Tushar Mohan, Jim Rosinski, Peter Watkins of SiCortex.

