An Open Source Performance Tool Suite for Scientific Computing

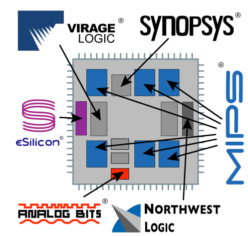
Philip J. Mucci

June 26th, 2007

International Supercomputing Conference

Evolution of HPC Hardware/Software Design

Pre-Linux Era → Linux Era → Neo-Linux Era



Linux

One System, One OS

New software stack for every new system design and workload

One OS, Many Systems

Stack adaptable to many different system designs and workloads

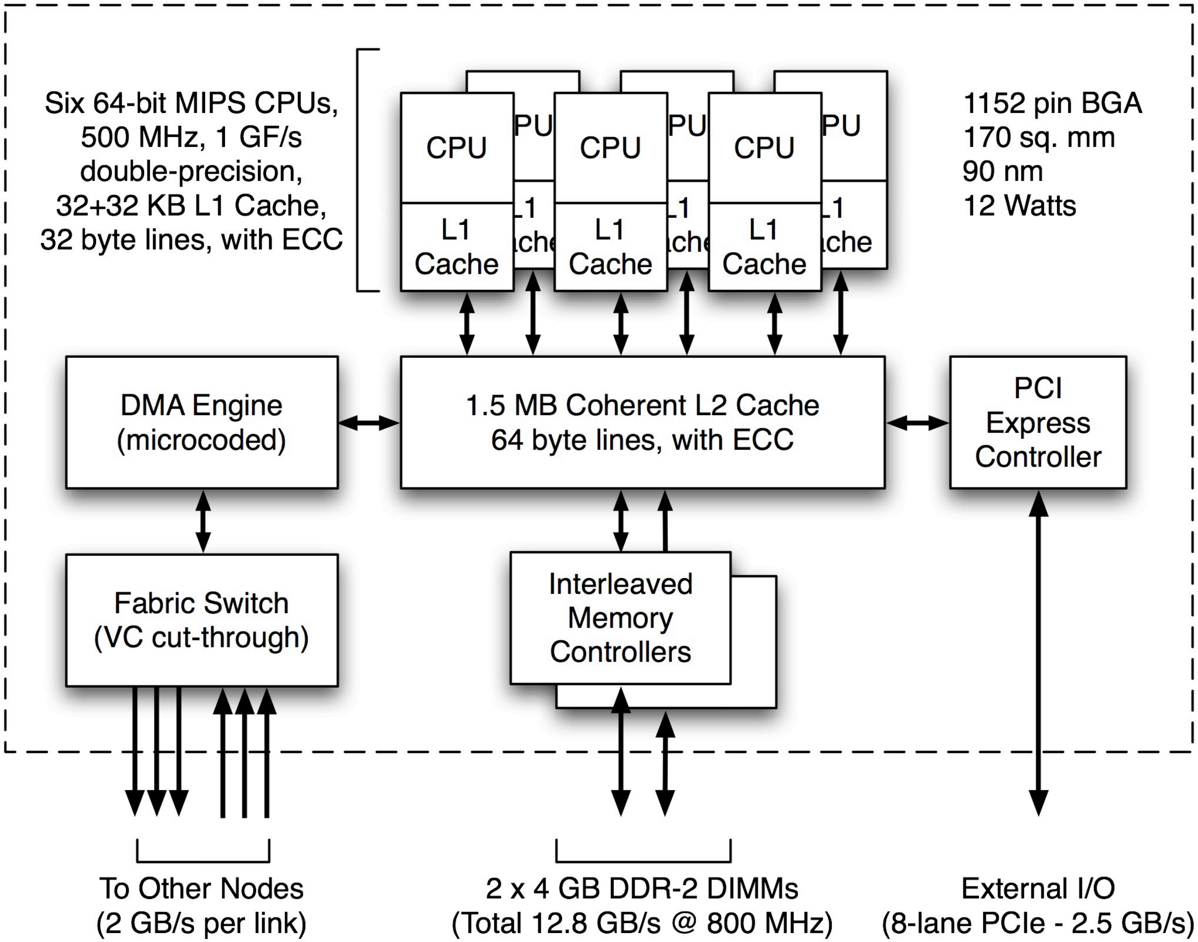
Targeted system design

HPC workload on industry standard OS drives the system design

Guiding Principles

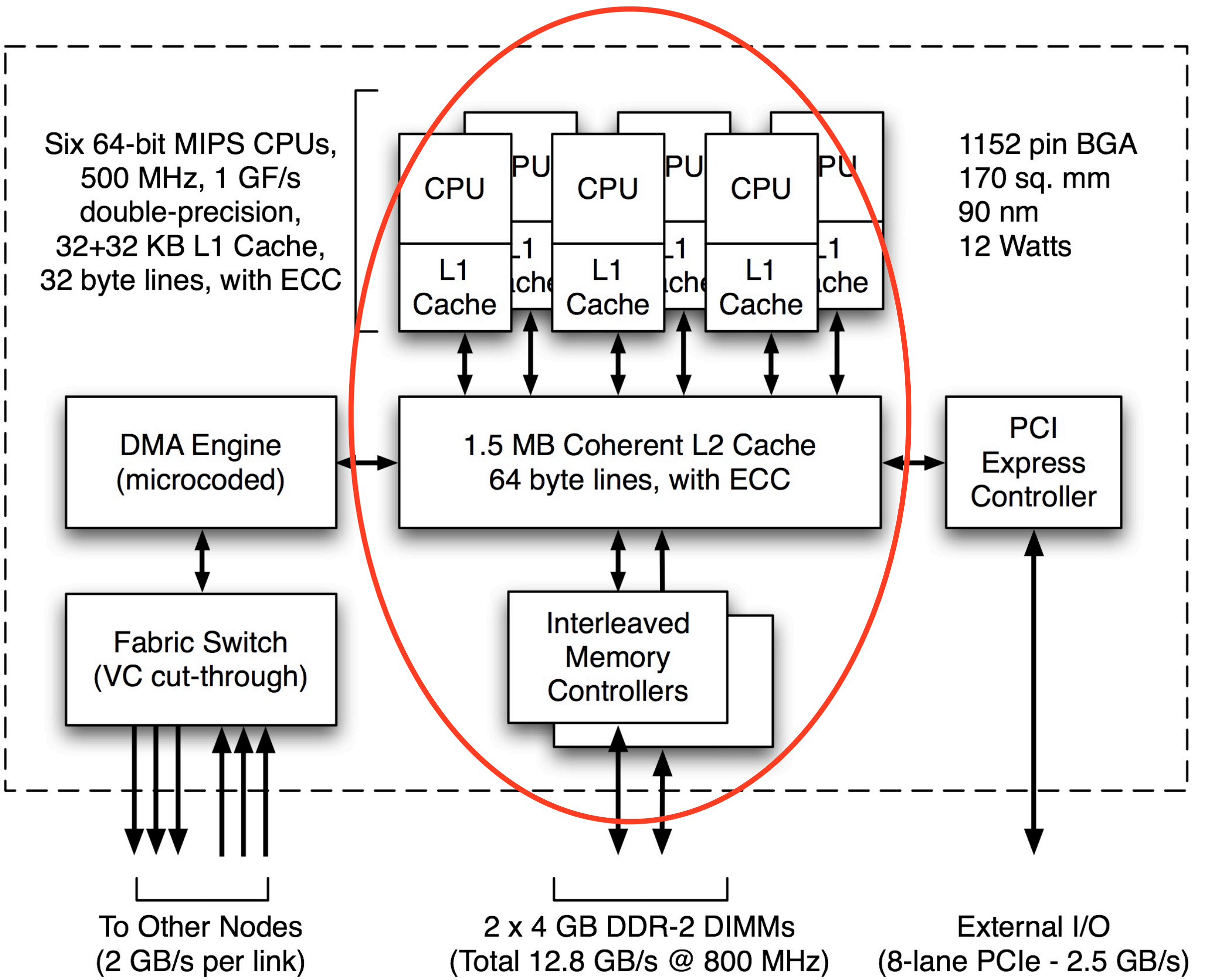
- Hardware
  - Run existing HPC applications faster
    - Per dollar, per watt, per square foot
  - Simplify: If apps don't need it, leave it out!
    - CPU should run fast enough to keep memory busy.
    - Put nodes as close together as possible.
    - Minimize power per node through SOC design.
- Software
  - 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

Single Chip Cluster Node

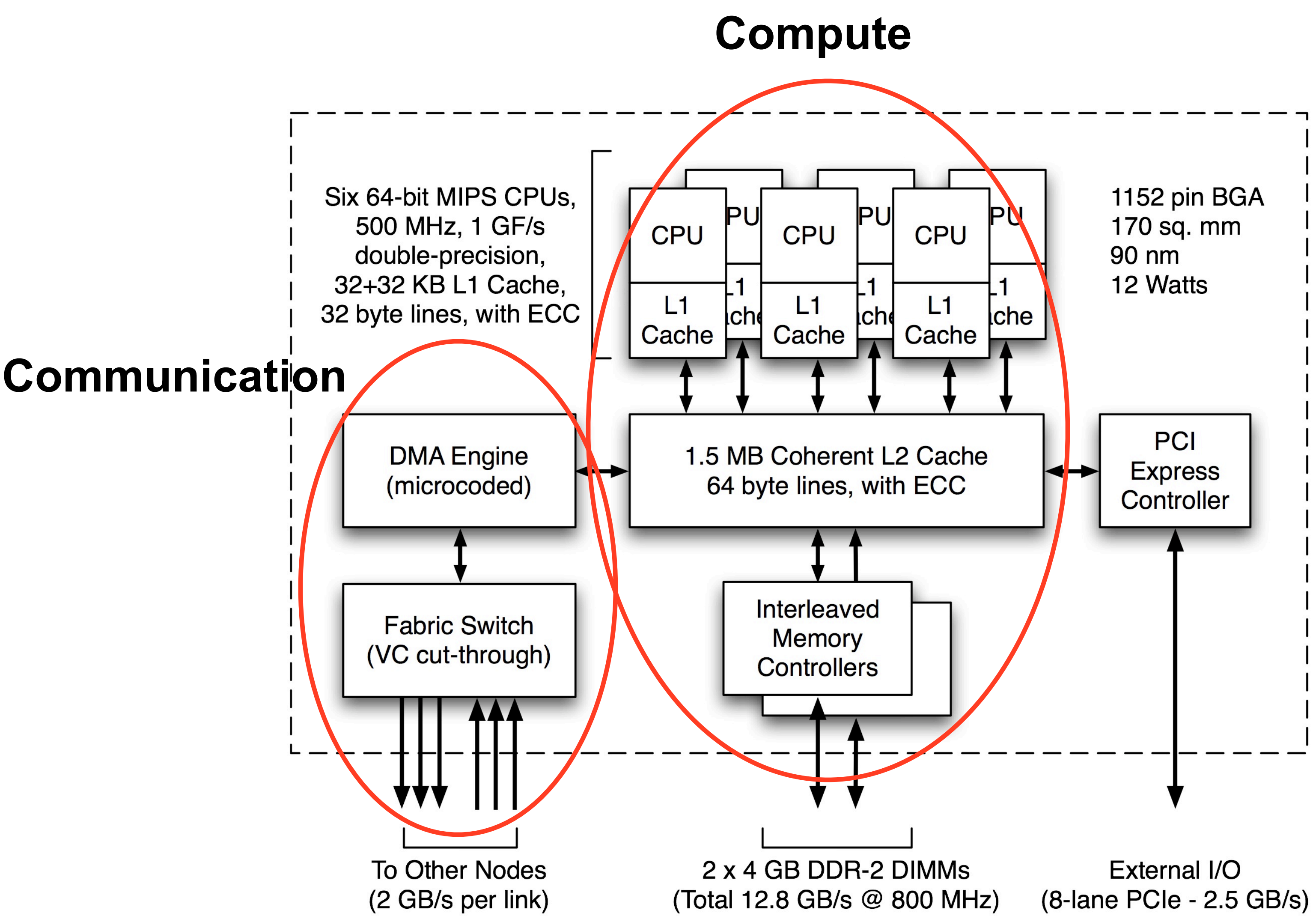


Single Chip Cluster Node

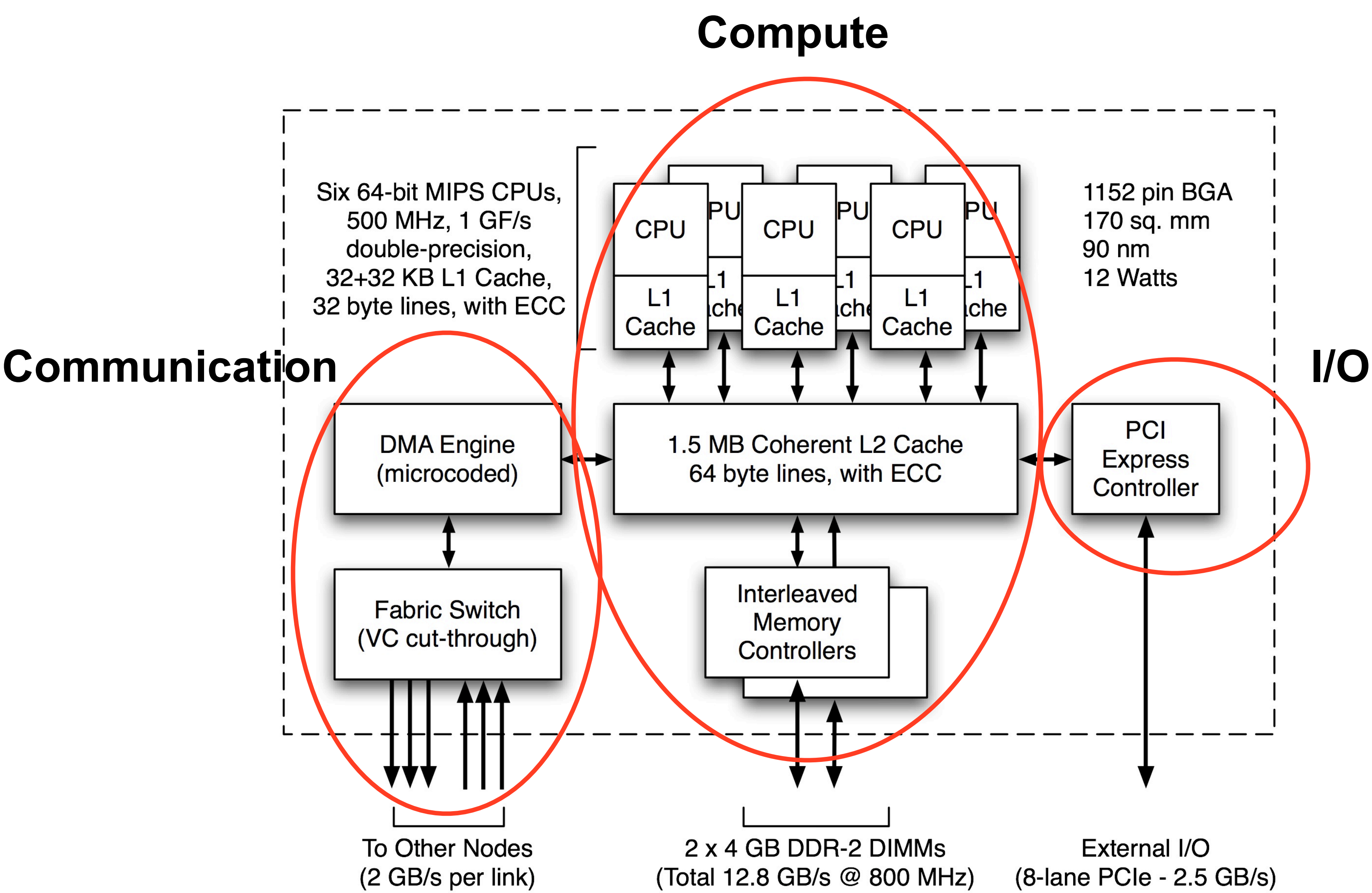
Compute



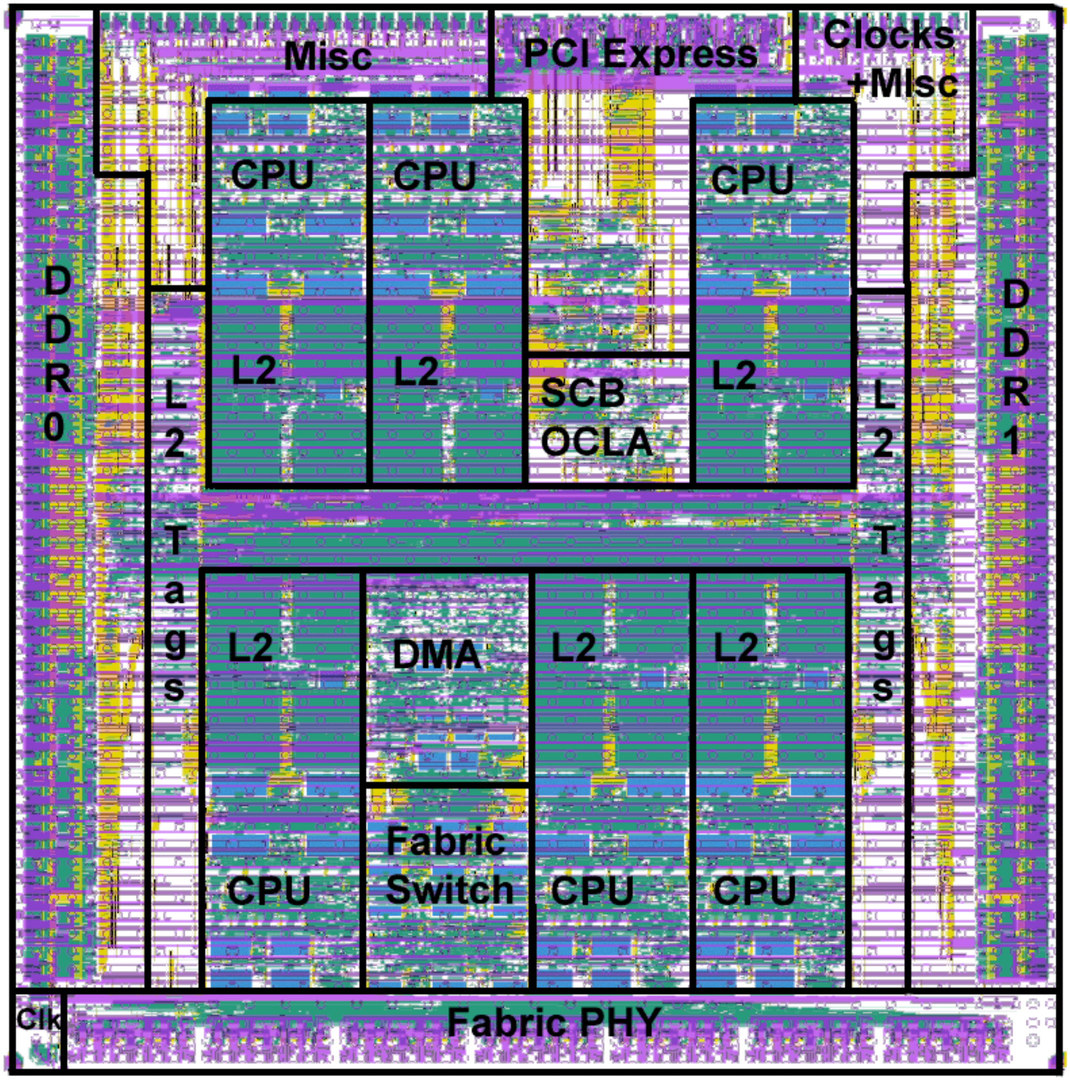
Single Chip Cluster Node



Single Chip Cluster Node

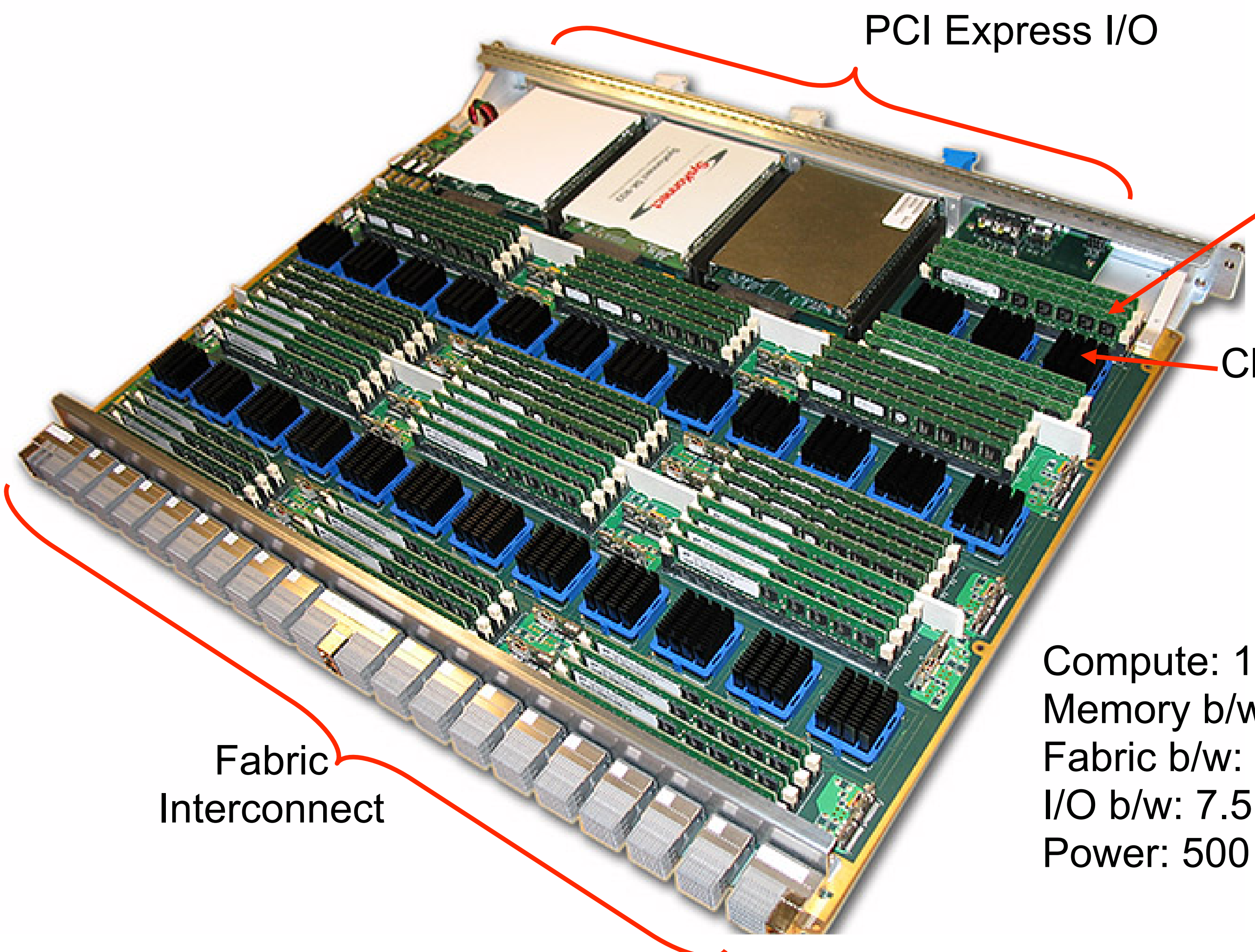


Single Chip Cluster Node



SiCortex

SiCortex CPU Module



Compute: 162 GF/sec
Memory b/w: 345 GB/sec
Fabric b/w: 162 GB/sec
I/O b/w: 7.5 GB/sec
Power: 500 Watts

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

SiCortex

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

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
  - 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

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

MFLIPS	66.51
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 %	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
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 twenty, descending, milliseconds) -
--------------------------------------------------------------
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
```


SiCortex

MPIPEX: Load Balance

```
----------------------------------------
@--- MPI Time (seconds) ---------------
----------------------------------------
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
```


SiCortex

Ioex

- 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

Ioex: 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
```

Hpcex

- 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

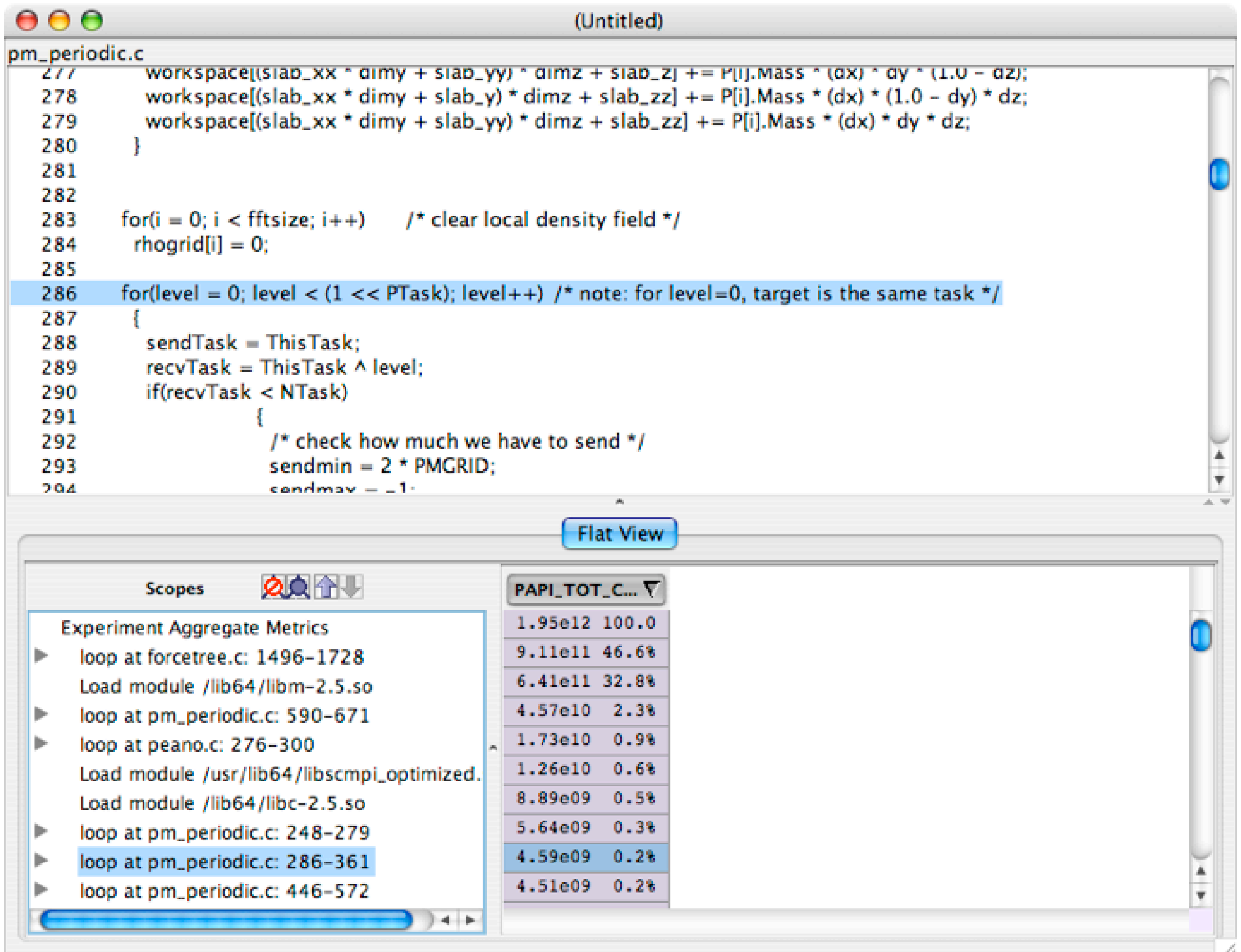
```
 0.8%      for (j = coff; j < nca-coff; j++)
             {
 0.1%        out = 0.0;
 2.5%        for (ki = 0; ki < nrk; ki++)
               {
 6.5%          for (kj = 0; kj < nck; kj++)
                 {
                 // out += a[i+ki][j+kj] * k[ki][kj];
26.1%            out += *(a+(i+ki-roff)*nca + j+kj-coff) * *(k+(ki*nck)+kj);
                 }
               }
             // c[i+roff][j+coff] = out;
 1.0%        *(c+(i)*nca + j) = out;
             }
           }
```

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
```


SiCortex

Hpcviewer: Loop-level profiling



SiCortex

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

Tuning and Analysis Utilities

- 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

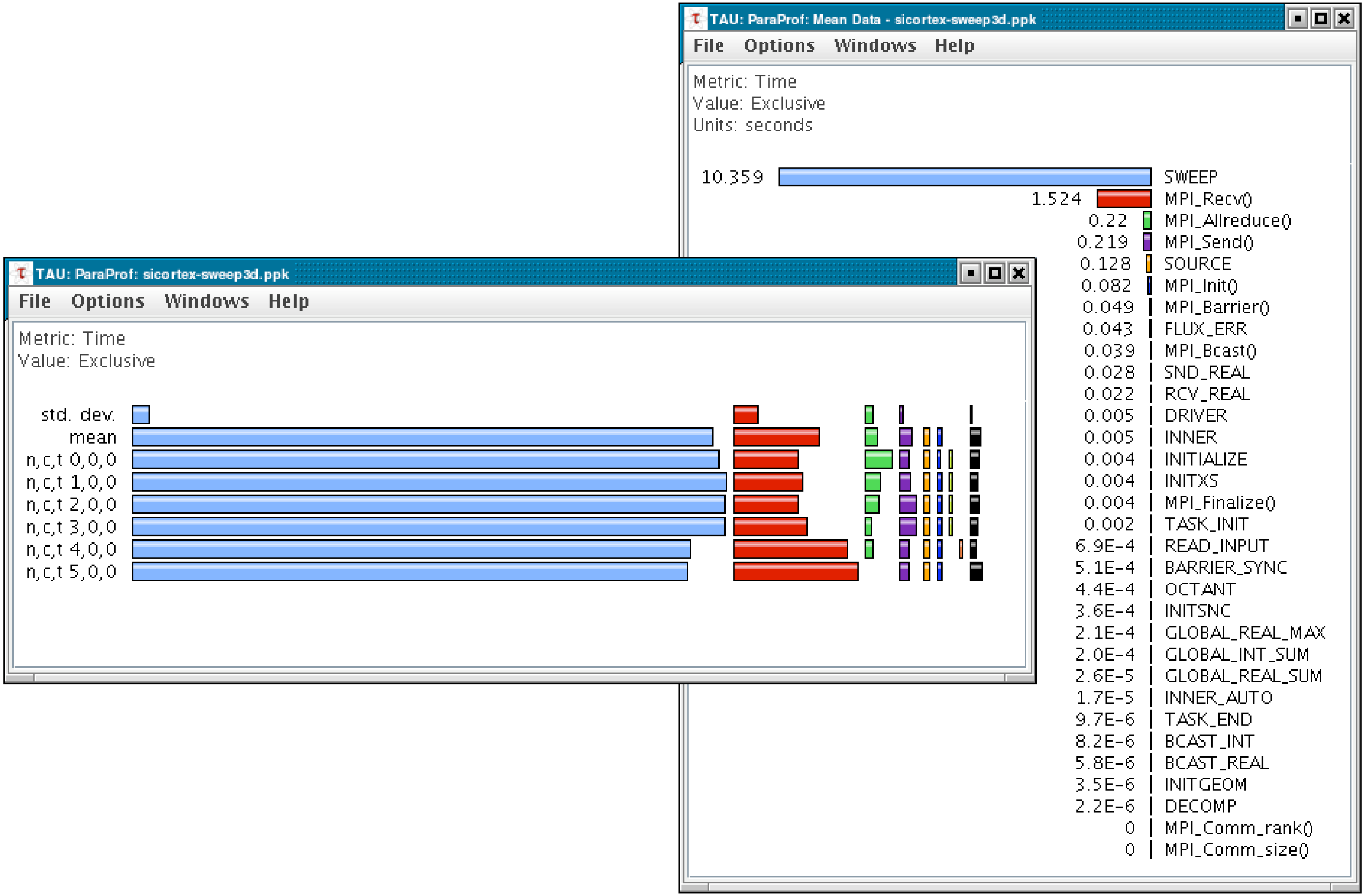
Tuning and Analysis Utilities

- 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 .

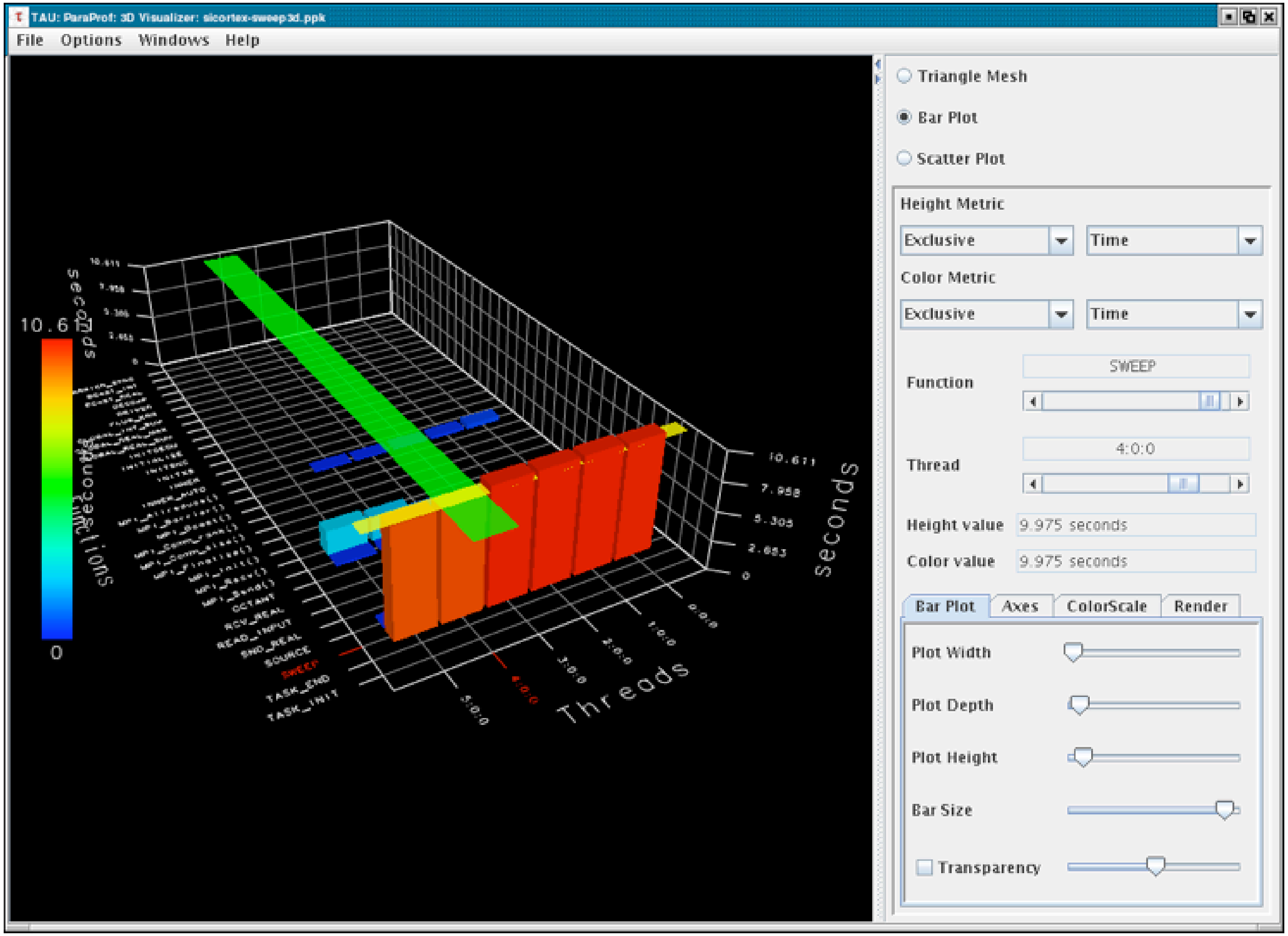
Tauex

- 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

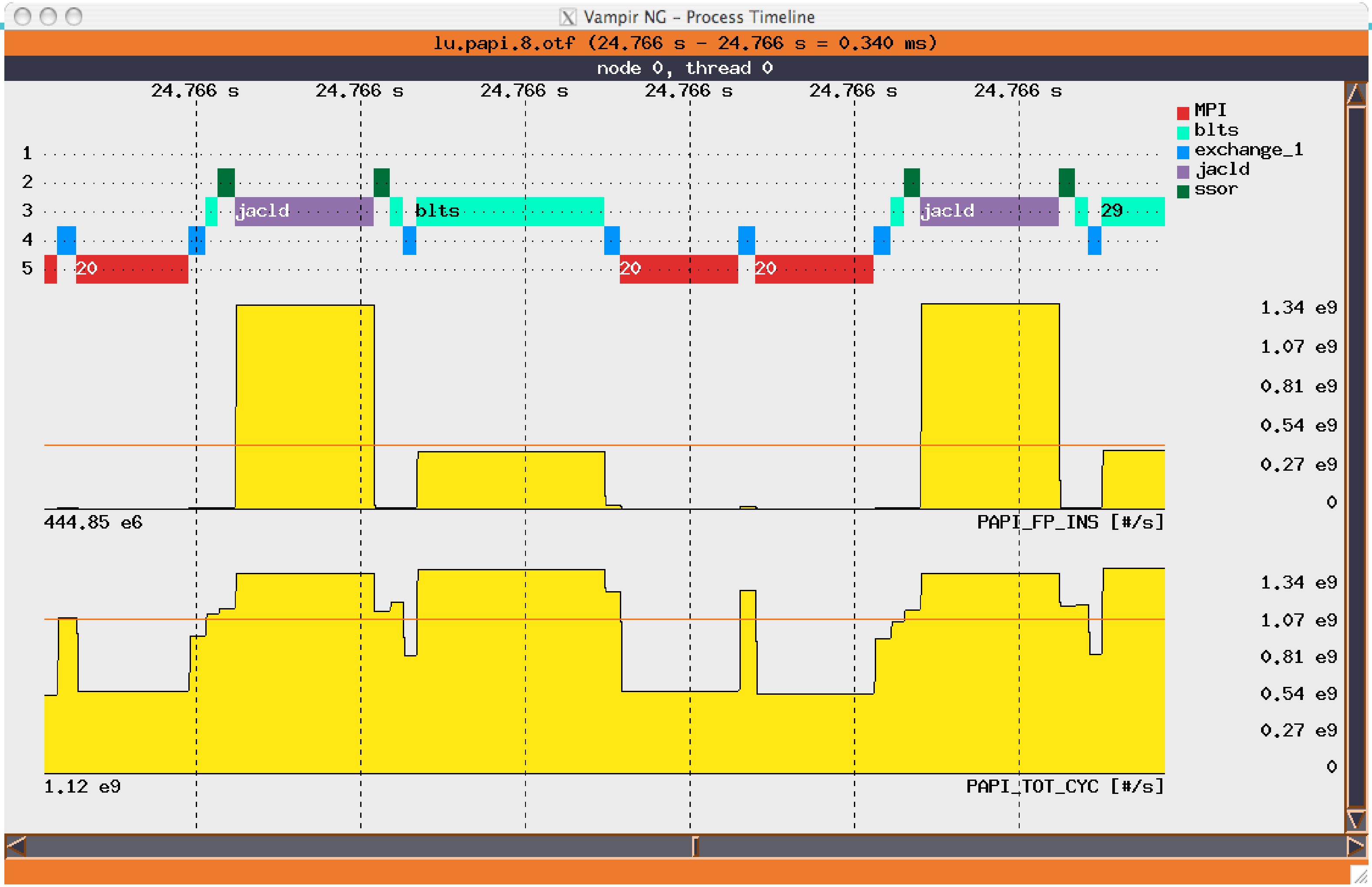


TAUEX: ParaProf 3D Profile



SiCortex

Visualizing TAU Traces with VampirNG



SiCortex

Vampir

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

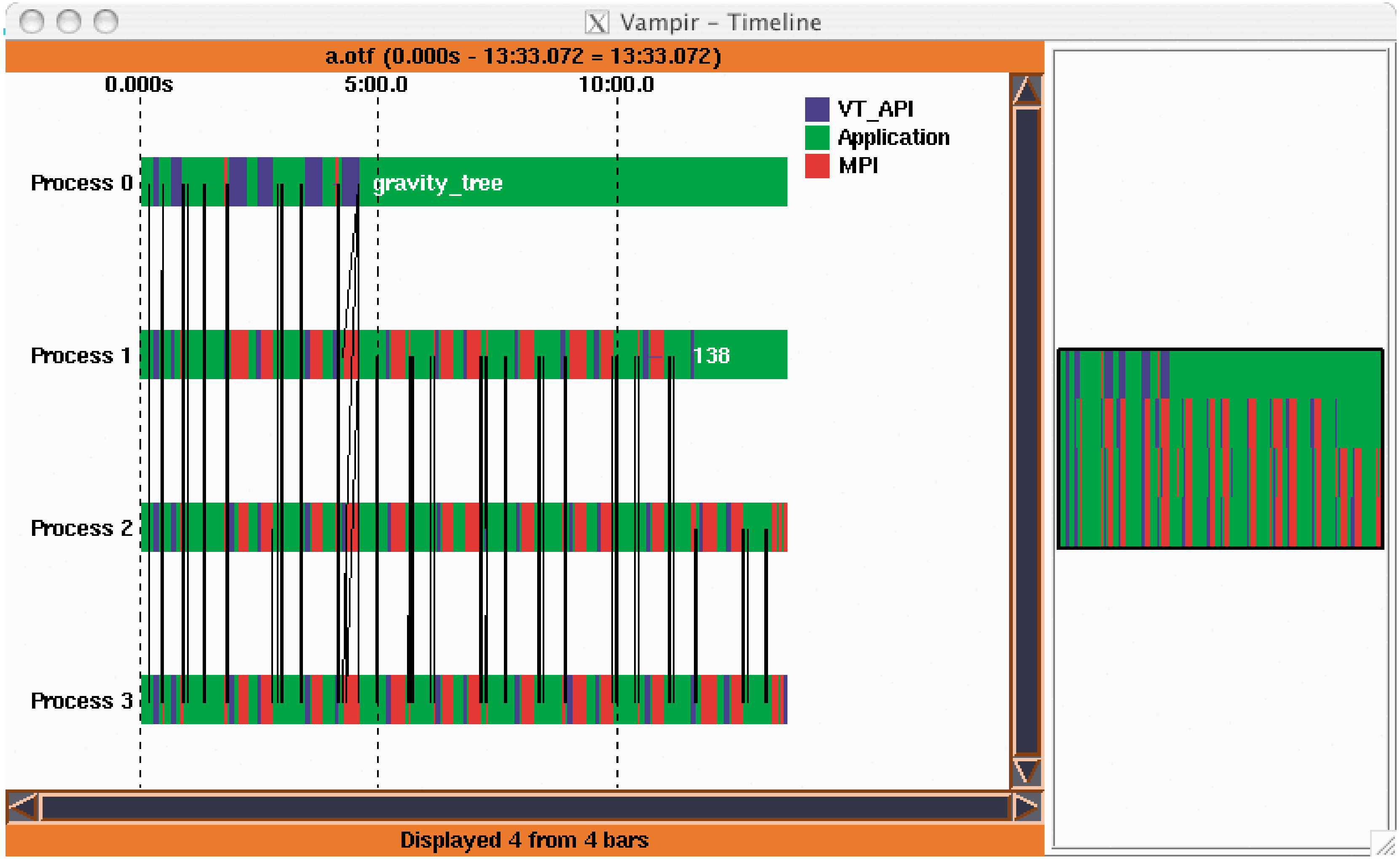
VampirTrace

- 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

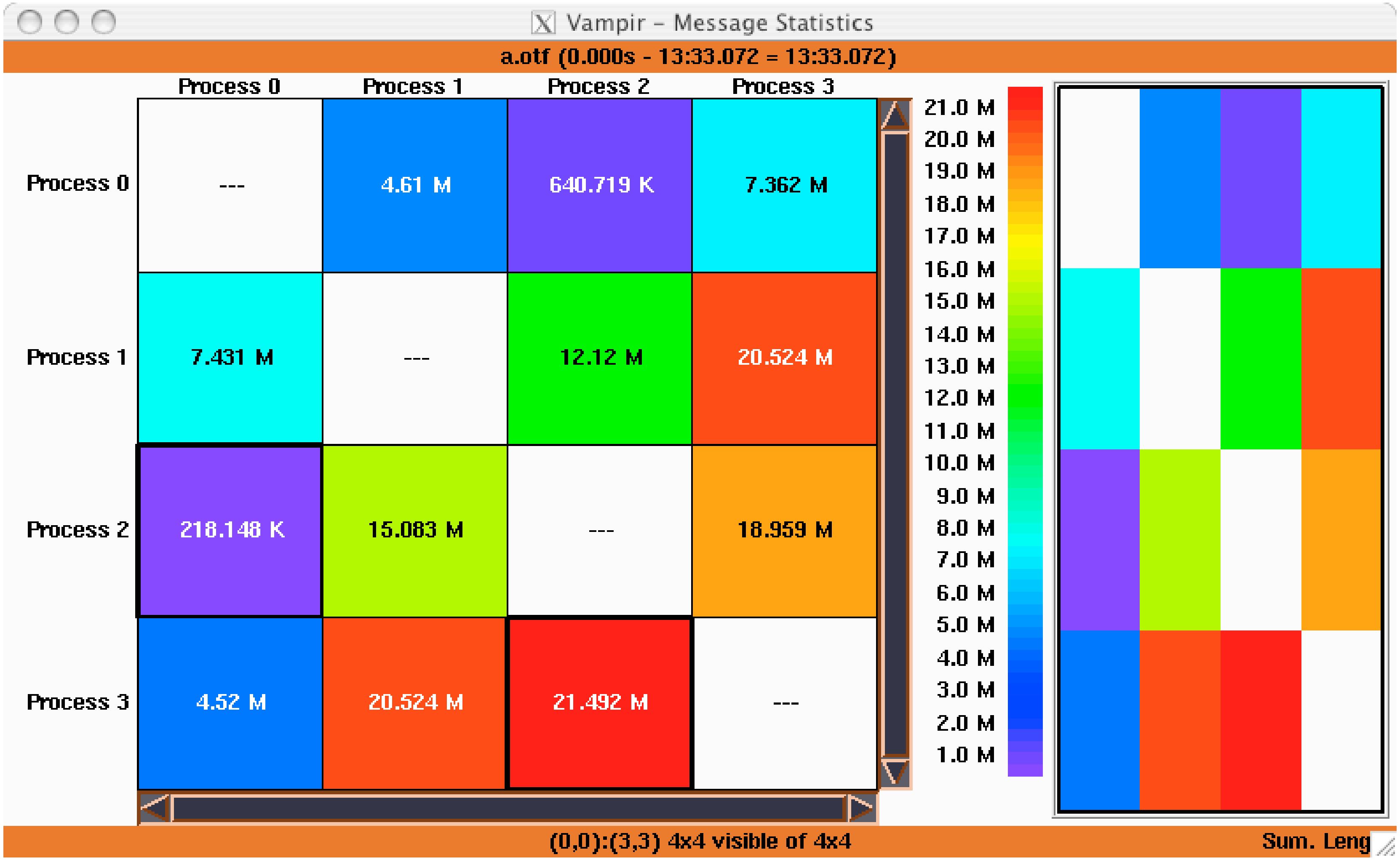
- VampirServer: Distributed high-end performance visualization
  - Client/server architecture
  - Parallel event processing
  - Runs on a (part of a) production environment
  - No need to transfer huge traces, uses parallel I/O
- VampirServer Browser: Lightweight client on local workstation
  - Outer 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



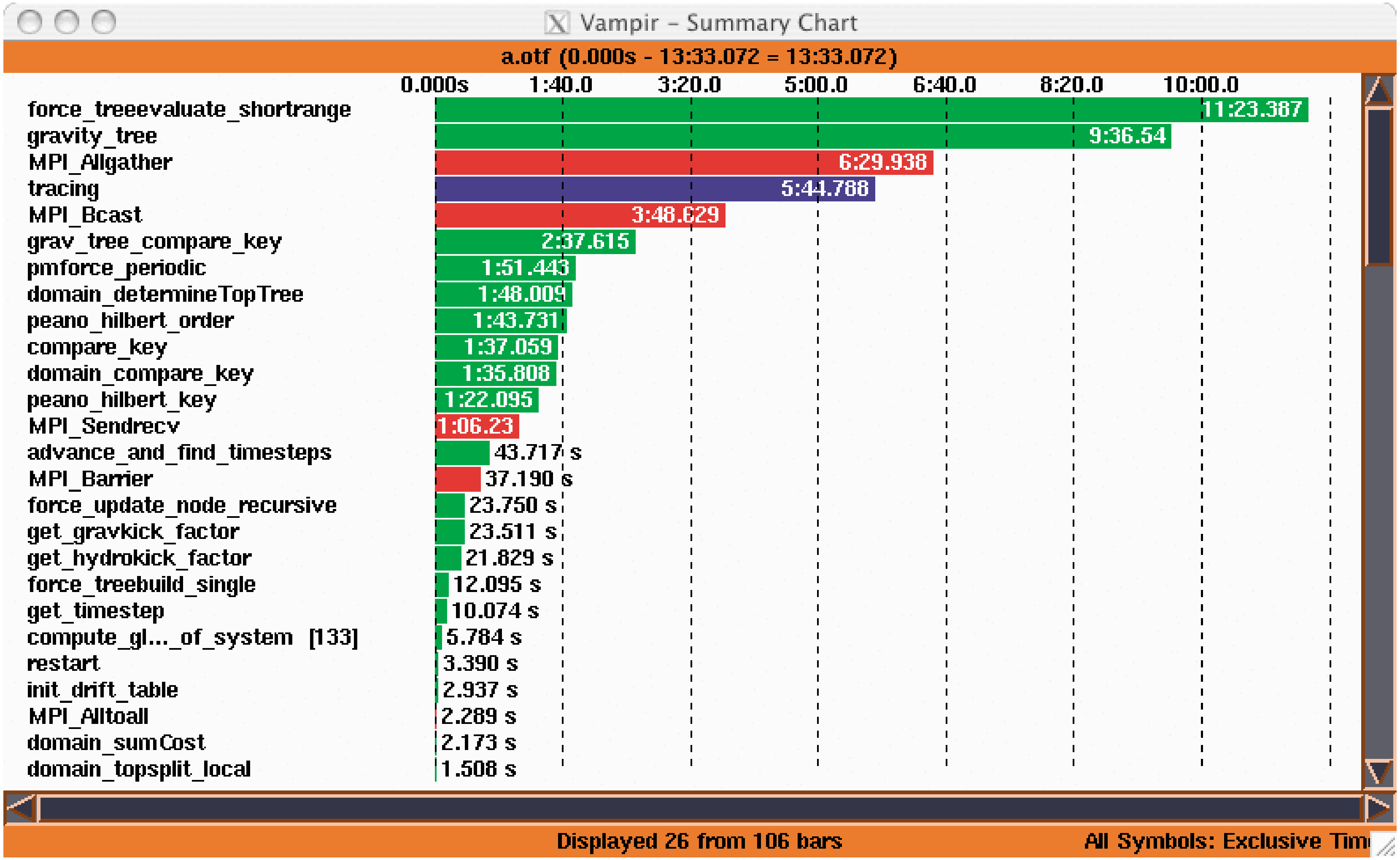
SiCortex

Vampir Message Statistics



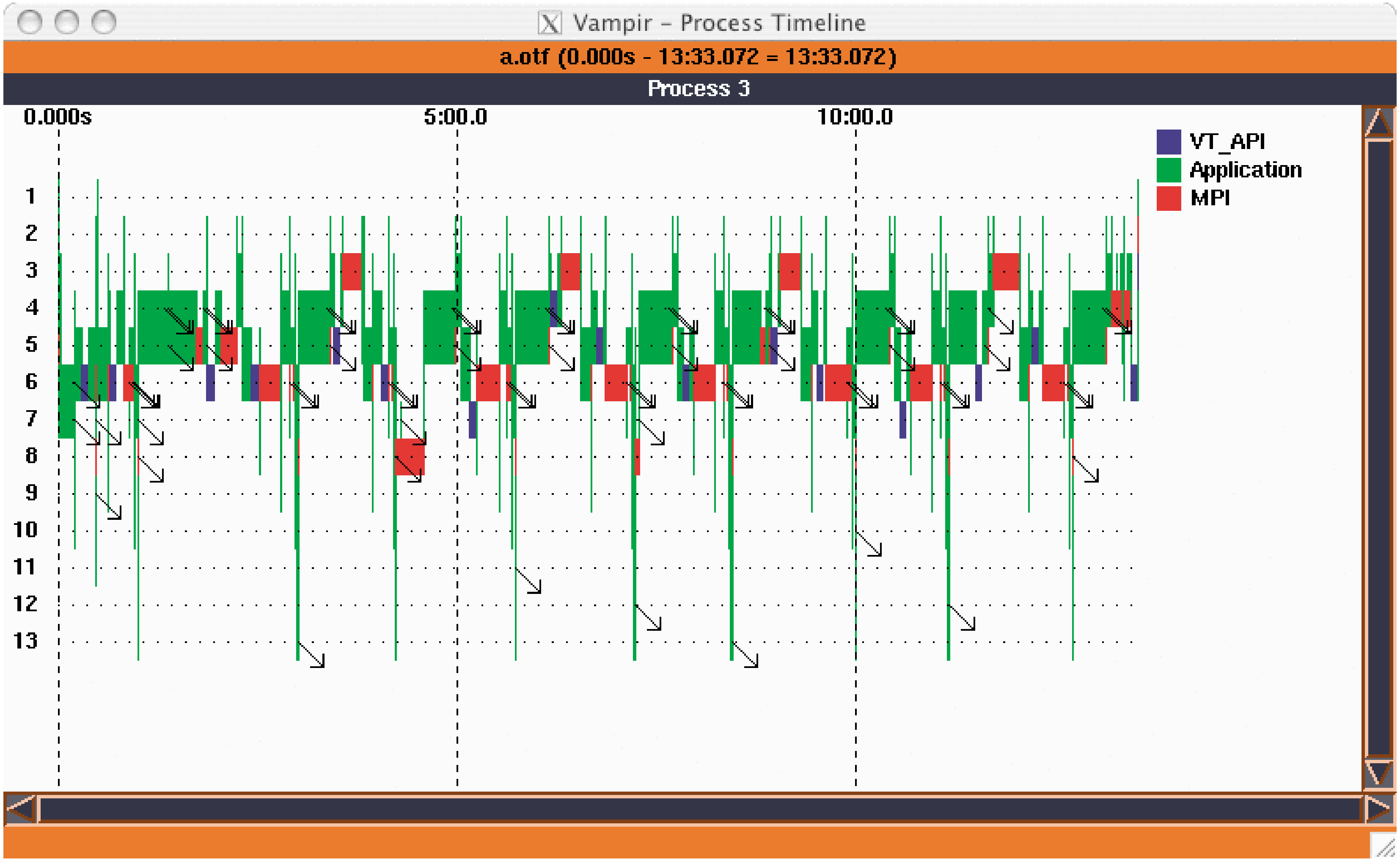
	Process 0	Process 1	Process 2	Process 3
Process 0	---	4.61 M	640.719 K	7.362 M
Process 1	7.431 M	---	12.12 M	20.524 M
Process 2	218.148 K	15.083 M	---	18.959 M
Process 3	4.52 M	20.524 M	21.492 M	---

Vampir Summary Chart



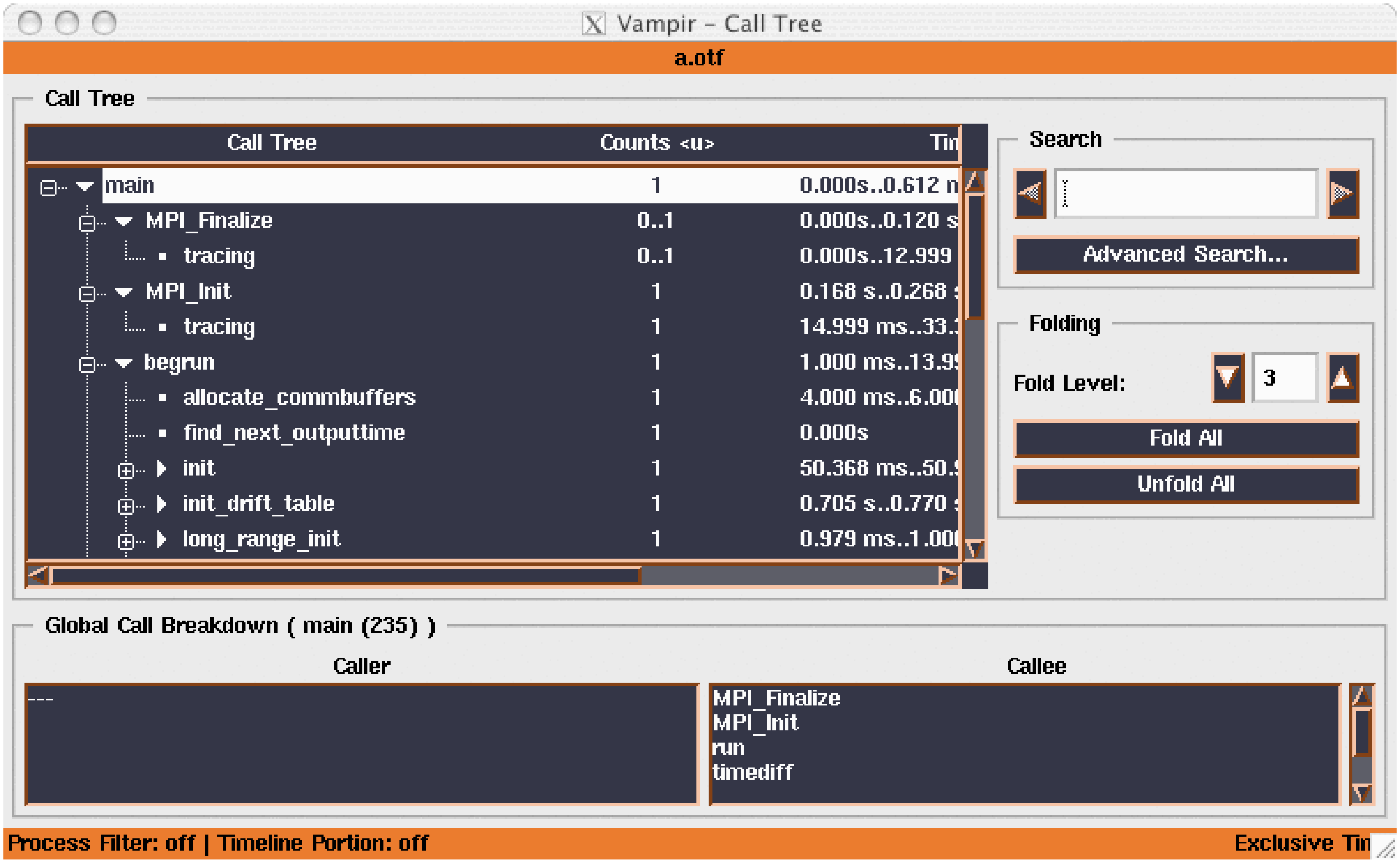
SiCortex

Vampir Process Timeline



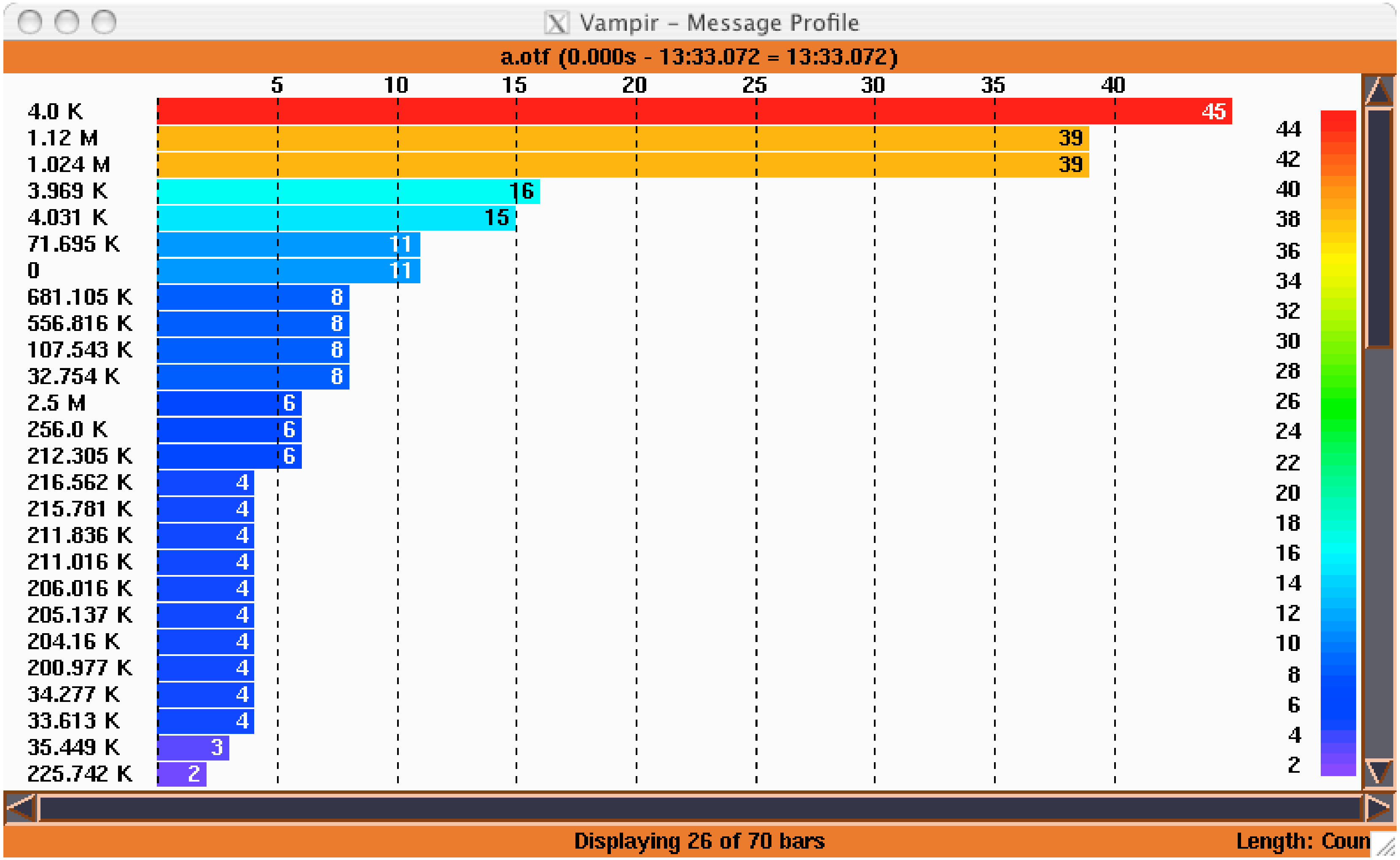
SiCortex

Vampir Call Tree



SiCortex

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

SiCortex

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.