Scalability Improvements in the TAU Performance System

Sameer Shende
Dir., Performance Research Laboratory, University of Oregon

Workshop on Extreme Scale Performance Tools

SC'12, Salt Lake City Convention Center, Room 155-F

10:45am-11am, Friday, Nov. 16, 2012

http://tau.uoregon.edu

UNIVERSITY
OF OREGON

Acknowledgments

- Prof. Allen D. Malony, Professor CIS, and Director NeuroInformatics Center, University of Oregon
- Scott Parker, Argonne National Labs
- Kalyan Kumaran, Argonne National Labs
- Dr. Kevin Huck, U. Oregon
- Wyatt Spear, U. Oregon
- John Linford, ParaTools, Inc.
- Suzanne Millstein, U. Oregon
- Scott Biersdorff, U. Oregon
- Nick Chaimov, U. Oregon
- Dr. Robert Yelle, U. Oregon

Outline

- Introduction to TAU
- Scalable topology displays
- Improving tool usability for extreme scale performance evaluation
- Data reduction techniques in profiling
- Conclusions

UNIVERSITY OF OREGON

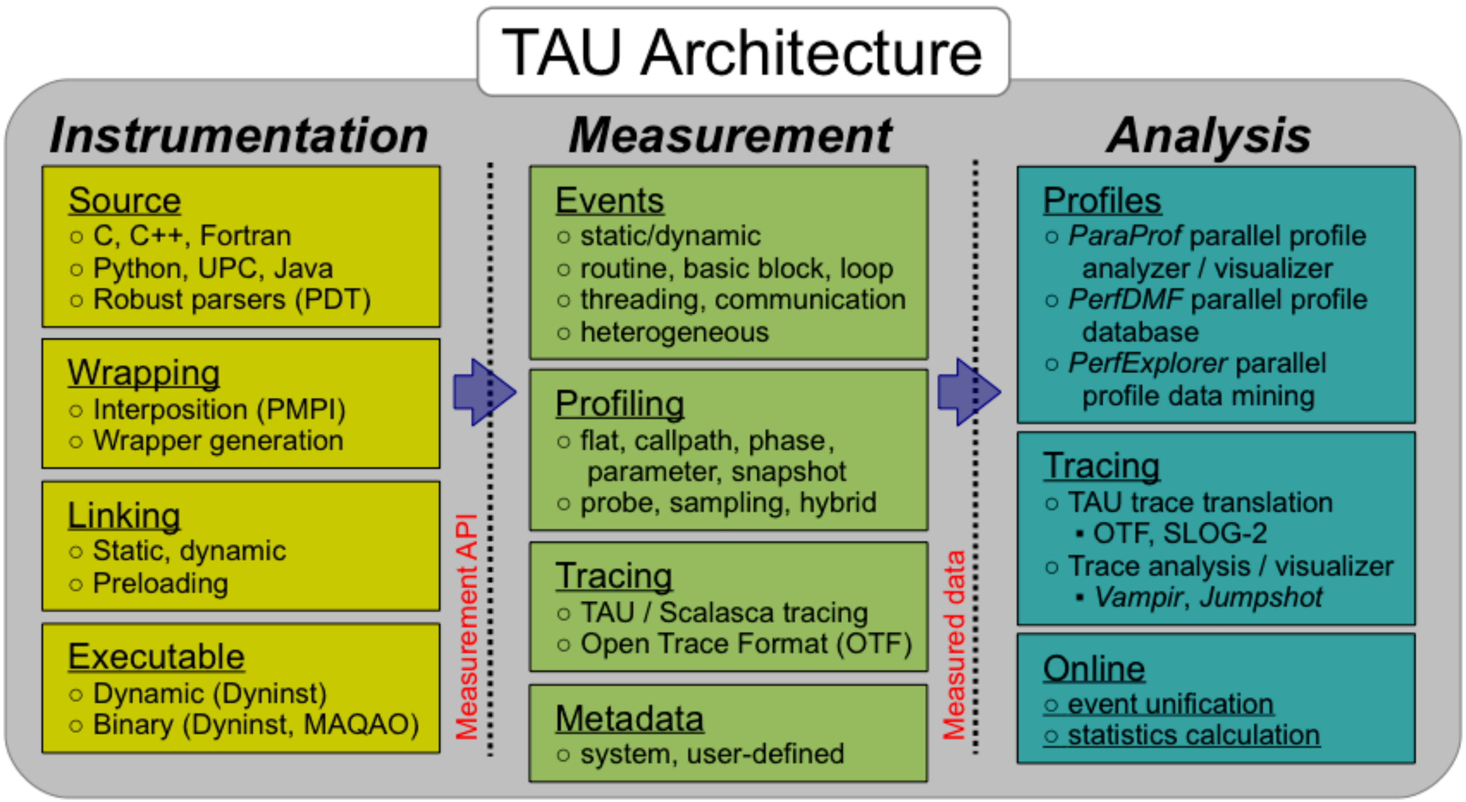
What is TAU?

- TAU is a portable profiling and tracing tool
- Profiling and tracing can measure time as well as hardware performance counters (cache misses, instructions) from your CPU
- TAU can automatically instrument your source code using a package called PDT for routines, loops, I/O, memory, phases, etc.
- TAU runs on most HPC platforms and it is free (BSD style license)
- TAU has instrumentation, measurement, and analysis tools
- TAU interfaces with other tools such as Score-P measurement library, PAPI hardware counter library, Vampir, ParaVer, and Scalasca analysis tools
- It can scale to large core counts
- *http://tau.uoregon.edu*

TAU Performance System®

Parallel performance framework and toolkit

- Goal: to supports all HPC platforms, compilers, and runtime systems
- Provides portable instrumentation, measurement, analysis



Key features of TAU

- **Support for outer-loop level instrumentation using both source (PDT) and binary rewriting capabilities**
- **Support for instrumentation of memory and I/O operations for accurate heap memory usage, memory allocation/de-allocation, and I/O volume and bandwidth computations**
- **Wrapping technology for instrumenting any external library**
- **Performance database technology to store performance data, cross experiment and data mining tool (PerfExplorer)**
- **Support for hybrid sampling and direct measurement**
- **3D profile browser, ParaProf**
- **Automatic performance measurement system on BG/P, Q.**
- **Dynamic event group creation and reassignment in ParaProf**
- **Support for debugging (Callstack, memory leak detection, and *soon* runtime bounds checking)**
- **Cross-platform and cross-language portability**

Early Availability on New Systems

- Intel compilers with Intel MPI on Intel Xeon Phi^{TM} (MIC)
- GPI with Intel Linux x86_64 Infiniband clusters
- IBM BG/Q and Power 7 Linux with IBM XL UPC compilers
- NVIDIA Kepler K20 with CUDA 5.0 with NVCC
- Fujitsu Fortran/C/C++ MPI compilers on the K computer
- PGI compilers with OpenACC support on NVIDIA systems
- Cray CX30 Sandybridge Linux systems with Intel compilers
- Cray CCE compilers with OpenACC support on Cray XK7
- AMD OpenCL libs with GNU on AMD Fusion cluster systems
- MPC compilers on TGCC Curie system (Bull, Linux x86_64)
- GNU compilers on ARM Linux clusters (MontBlanc, BSC)
- Cray CCE compilers with OpenACC on Cray XK6 (K20)
- Microsoft MPI with Mingw compilers under Windows Azure
- LLVM and GNU compilers under Mac OS X

Understanding Application Performance using TAU

- **How much time is spent in each application routine and outer *loops*? Within loops, what is the contribution of each *statement*?**
- **How many instructions are executed in these code regions? Floating point, Level 1 and 2 *data cache misses*, hits, branches taken?**
- **What is the *peak heap memory* usage of the code? When and where is memory allocated/de-allocated? Are there any memory leaks? Where?**
- **How much time does the application spend performing *I/O*? What is the peak read and write *bandwidth* of individual calls, total volume?**
- **What is the contribution of different *phases* of the program? What is the time wasted/spent waiting for collectives, and I/O operations in Initialization, Computation, I/O phases?**
- **How does the application *scale*? What is the efficiency, runtime breakdown of performance across different core counts?**

UNIVERSITY OF OREGON

How does TAU work?

Instrumentation: Adds probes to perform measurements

- Source code instrumentation using pre-processors and compiler scripts
- Wrapping external libraries (I/O, MPI, Memory, CUDA, OpenCL, pthread)
- Rewriting the binary executable using MAQAO (UVSQ, Intel Exascale Labs)

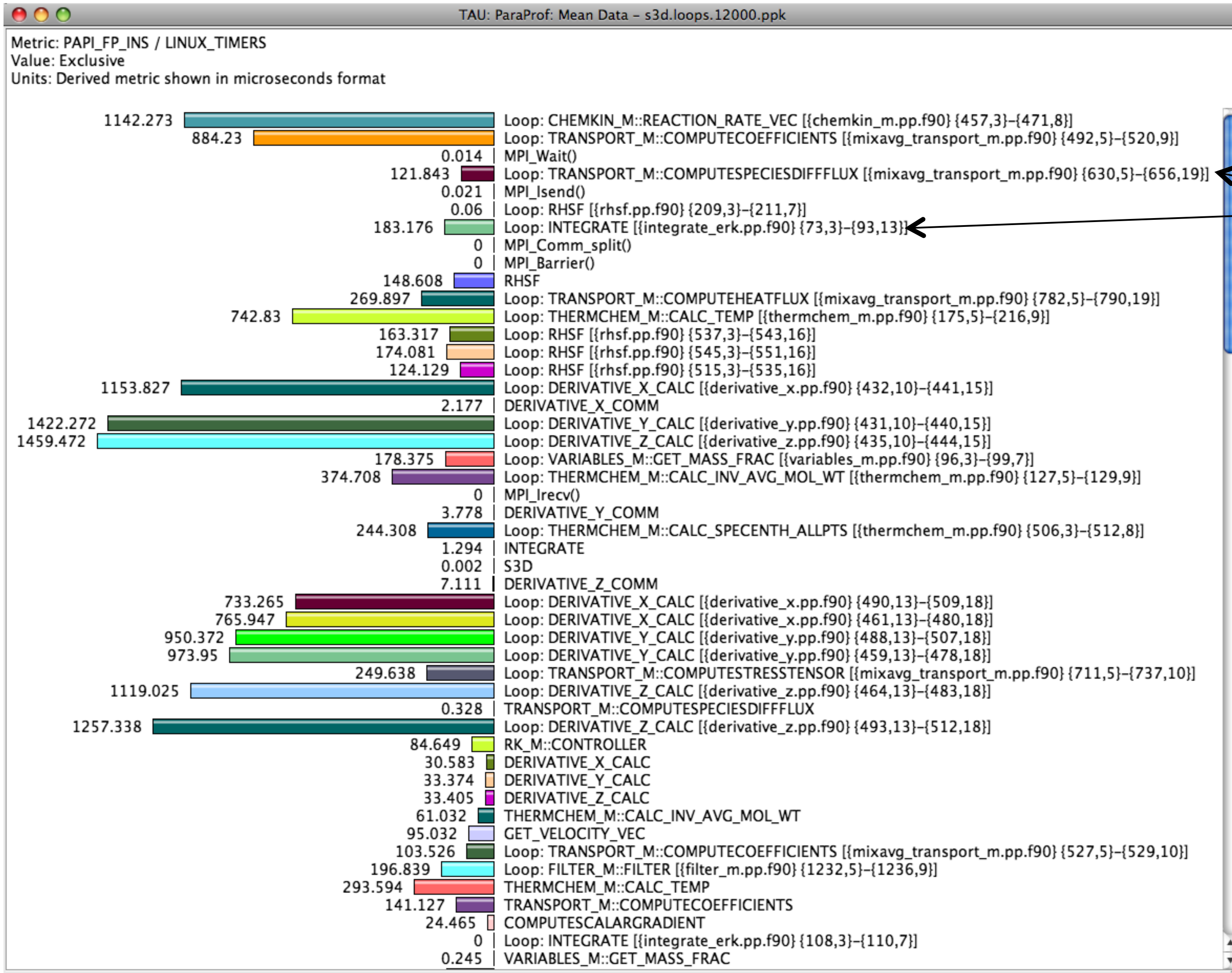
Measurement: Profiling or Tracing using wallclock time or hardware counters

- Direct instrumentation (Interval events measure exclusive or inclusive duration)
- Indirect instrumentation (Sampling measures statement level contribution)
- Throttling and runtime control of low-level events that execute frequently
- Per-thread storage of performance data
- Interface with external packages (e.g. PAPI hw performance counter library)

Analysis: Visualization of profiles and traces

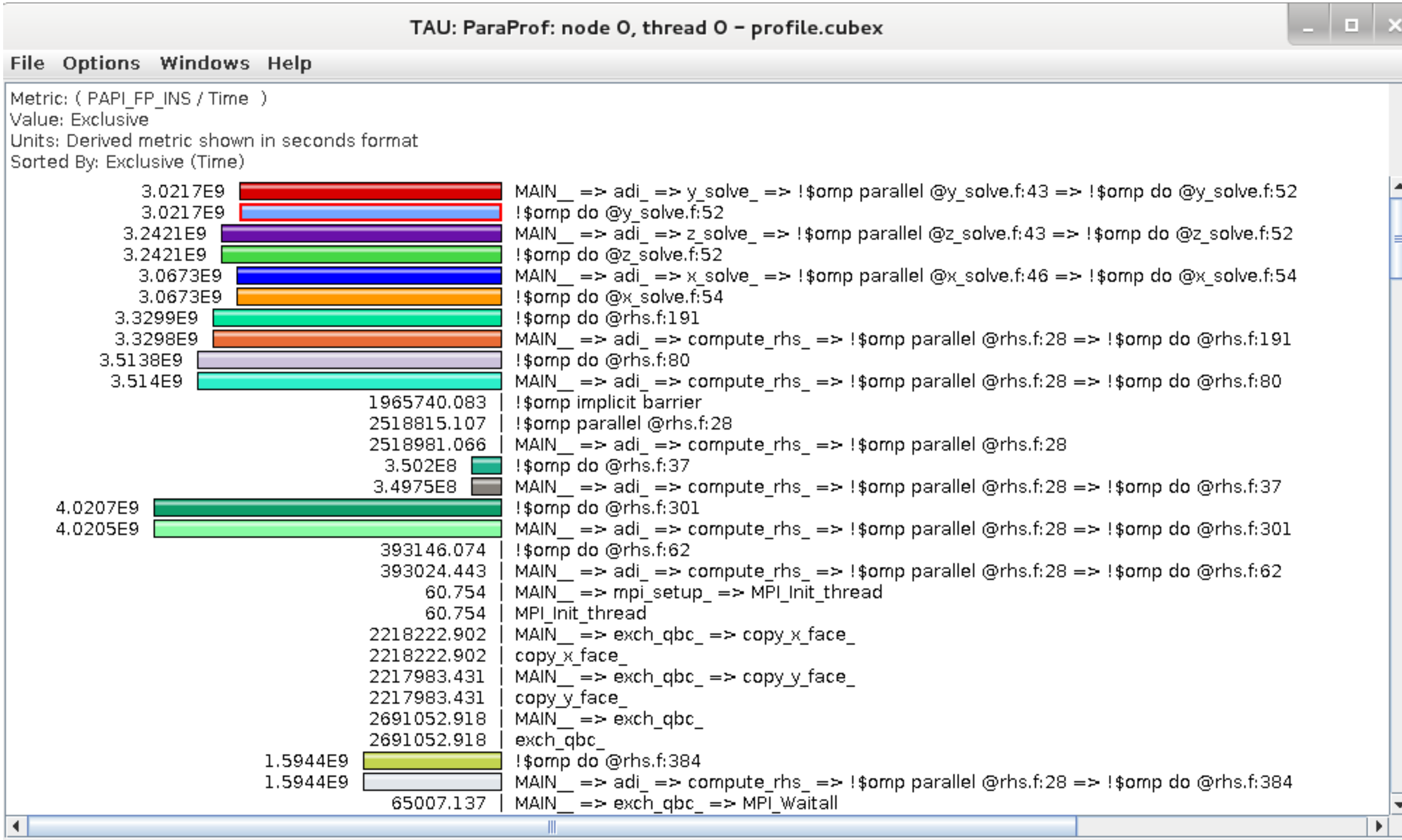
- 3D visualization of profile data in paraprof, perfexplorer tools
- Trace conversion & display in external visualizers (Vampir, Jumpshot, ParaVer)

Identifying Potential Bottlenecks



low mflops in loops?

Sorting Derived Flops Metric by Exclusive Time (Score-P with PAPI)

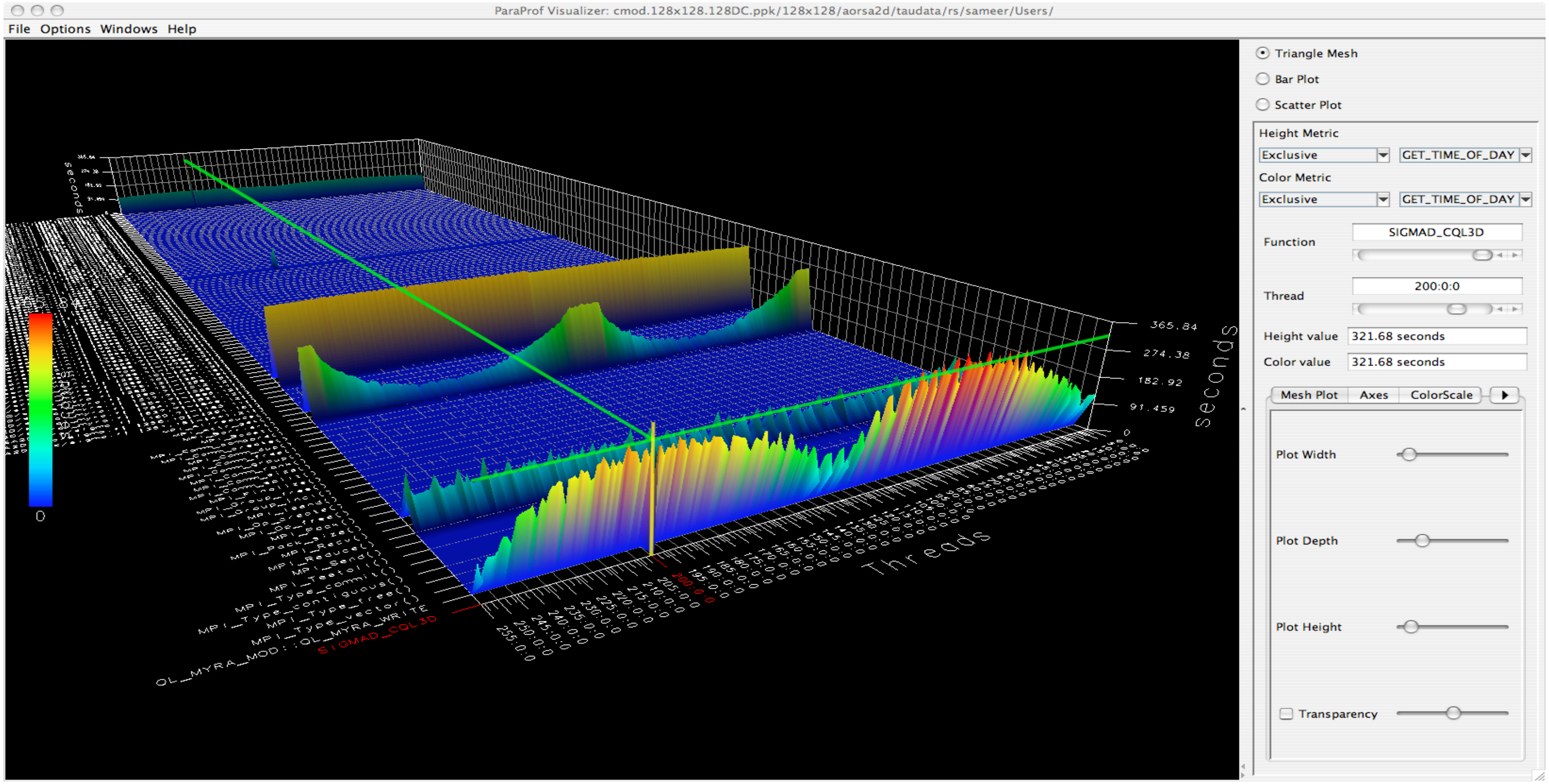


Memory Leaks in MPI

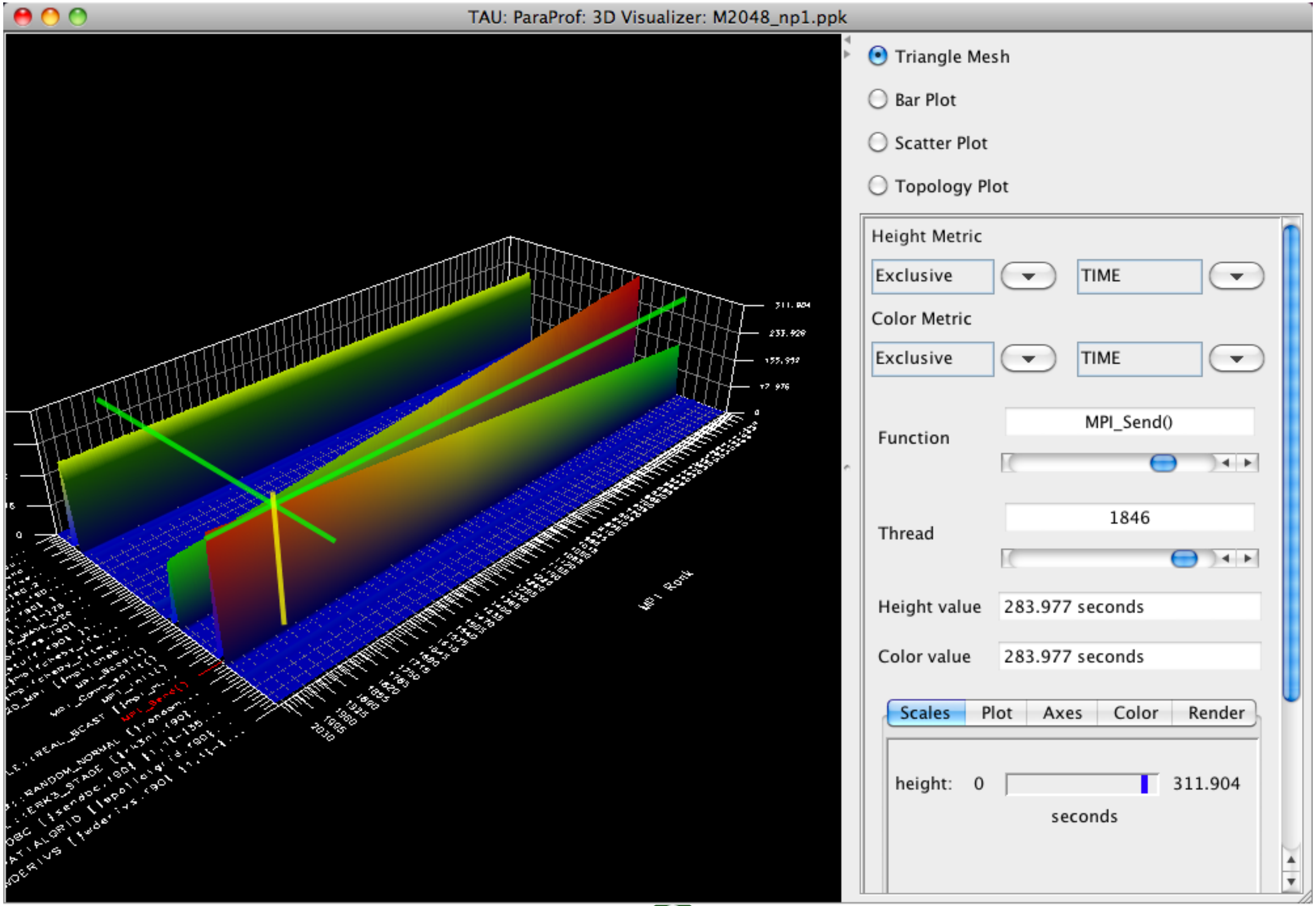
TAU: ParaProf: Context Events for thread: n,c,t, 0,0,0 – samarc_obe_4p_iomem_cp.ppk

Name	Total	MeanValue	NumSamples	MaxValue	MinValue	Std. Dev.
.TAU application						
MPI_Finalize()						
free size	23,901,253	22,719.822	1,052	2,099,200	2	186,920.948
malloc size	5,013,902	65,972.395	76	5,000,000	2	569,732.815
MEMORY LEAK!	5,000,264	500,026.4	10	5,000,000	3	1,499,991.2
read()						
Bytes Read	4	4	1	4	4	0
READ Bandwidth (MB/s) <file="pipe">		0.308	1	0.308	0.308	0
Bytes Read <file="pipe">	4	4	1	4	4	0
READ Bandwidth (MB/s)		0.308	1	0.308	0.308	0
write()						
WRITE Bandwidth (MB/s)		0.635	102	12	0	1.472
Bytes Written <file="/dev/infiniband/rdma_cm">	24	24	1	24	24	0
Bytes Written	1,456	14.275	102	28	4	5.149
WRITE Bandwidth (MB/s) <file="/dev/infiniband/uverbs0">		0.528	97	12	0.089	1.32
Bytes Written <file="pipe">	64	16	4	28	4	12
WRITE Bandwidth (MB/s) <file="/dev/infiniband/rdma_cm">		1.714	1	1.714	1.714	0
Bytes Written <file="/dev/infiniband/uverbs0">	1,368	14.103	97	24	12	4.562
WRITE Bandwidth (MB/s) <file="pipe">		2.967	4	5.6	0	2.644
writev()						
WRITE Bandwidth (MB/s)		4.108	2	7.667	0.549	3.559
Bytes Written	297	148.5	2	230	67	81.5
WRITE Bandwidth (MB/s) <file="socket">		4.108	2	7.667	0.549	3.559
Bytes Written <file="socket">	297	148.5	2	230	67	81.5
readv()						
Bytes Read	112	28	4	36	20	8
READ Bandwidth (MB/s) <file="socket">		25.5	4	36	10	11.079
Bytes Read <file="socket">	112	28	4	36	20	8
READ Bandwidth (MB/s)		25.5	4	36	10	11.079
MPI_Comm_free()						
free size	10,952	195.571	56	1,024	48	255.353
read()						
MPI_Type_free()						
MPI_Init()						
fopen64()						
free size	231,314	263.456	878	568	35	221.272
MEMORY LEAK!	1,105,956	1,868.169	592	7,200	32	3,078.574
malloc size	1,358,286	901.318	1,507	7,200	32	2,087.737
OurMain()						
fclose()						

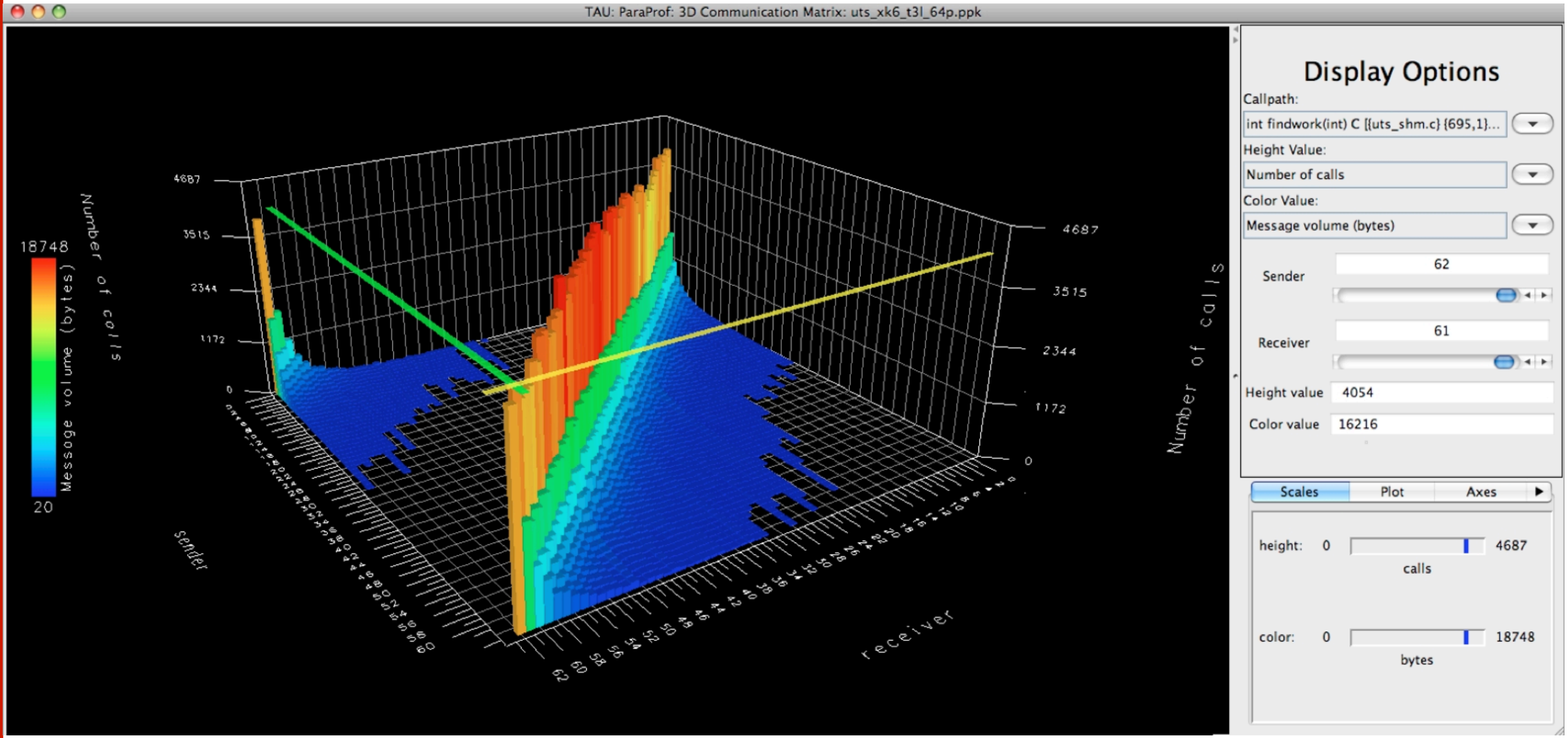
ParaProf 3D Profile Browser



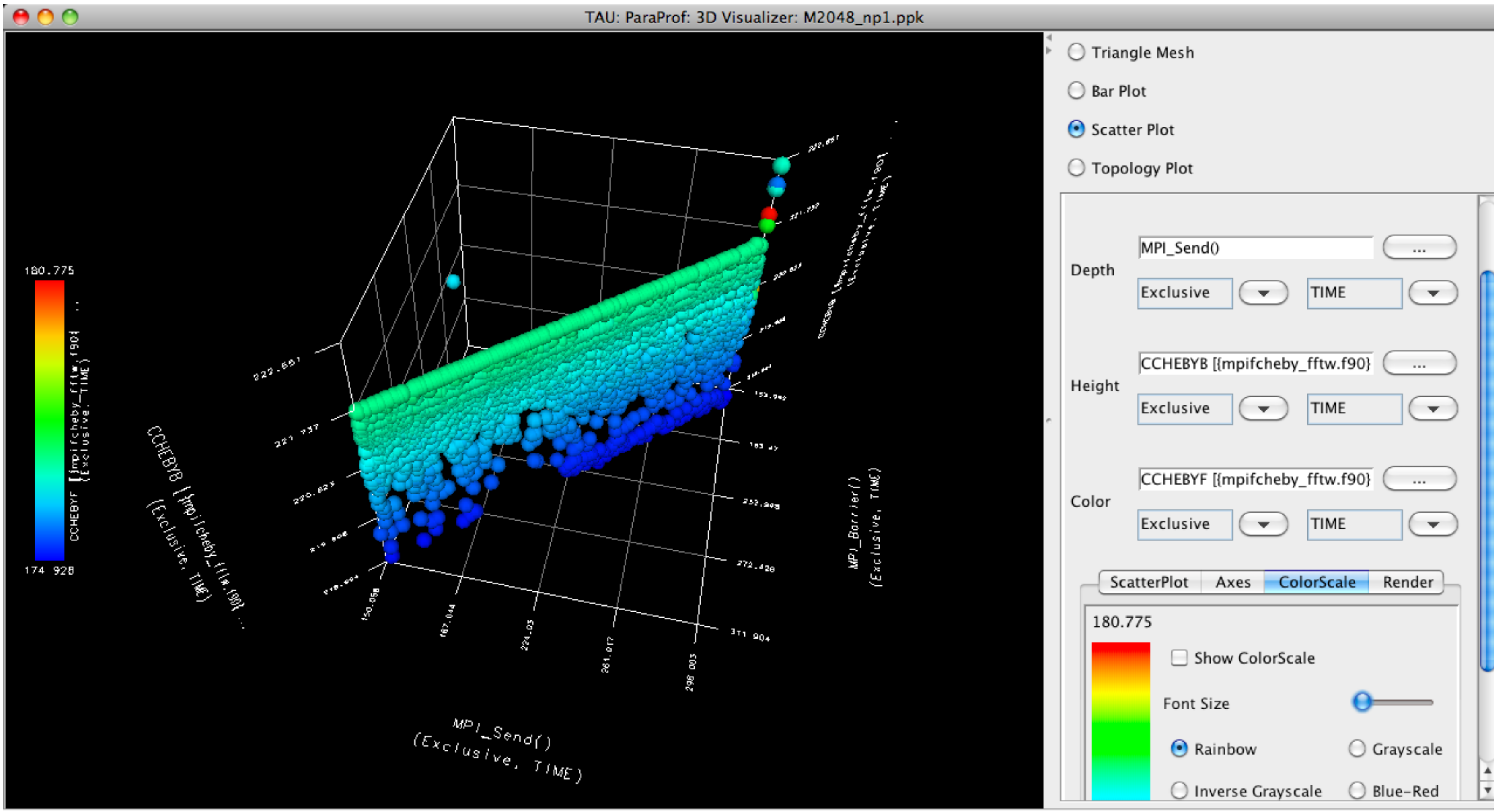
Scalable 3D Visualization in ParaProf



ParaProf: 3D Communication Matrix

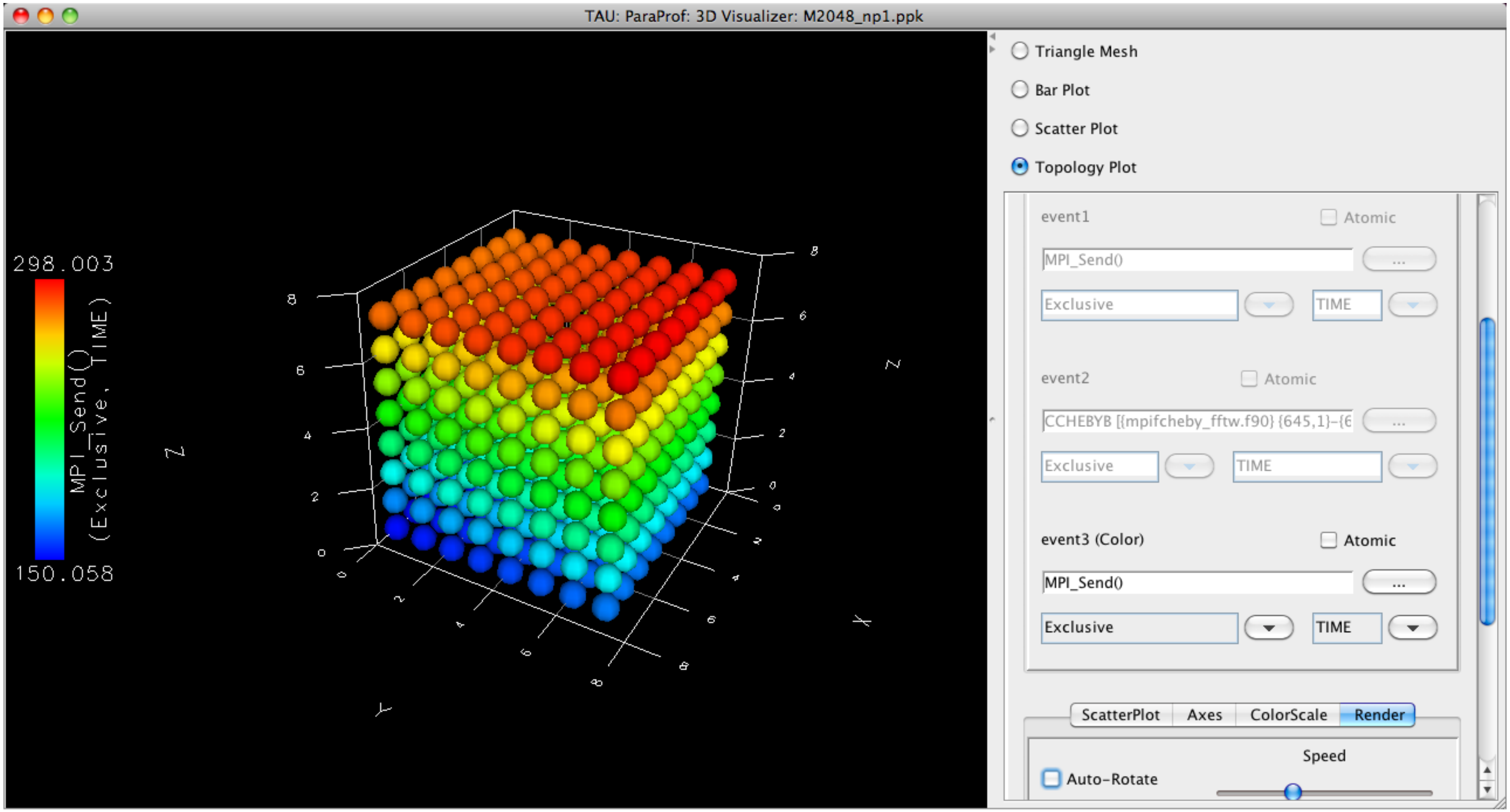


ParaProf: Scatter Plot

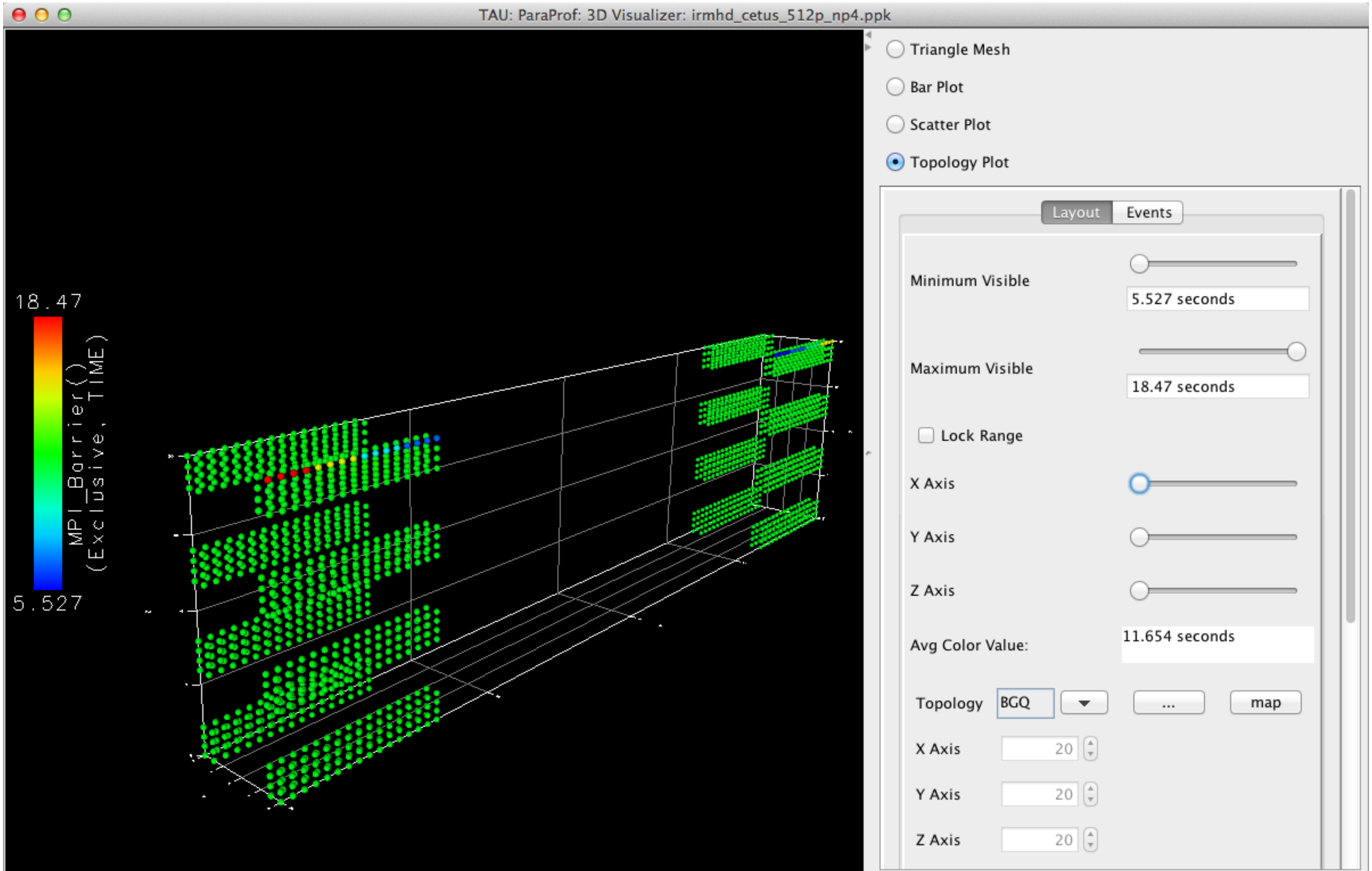


UNIVERSITY
OF OREGON

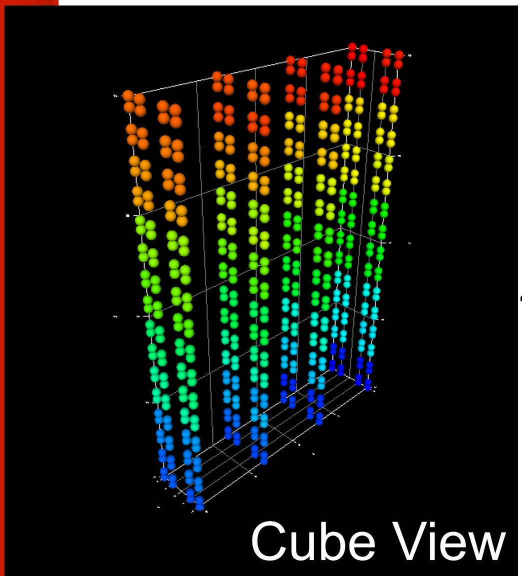
ParaProf: Topology View IBM BG/P



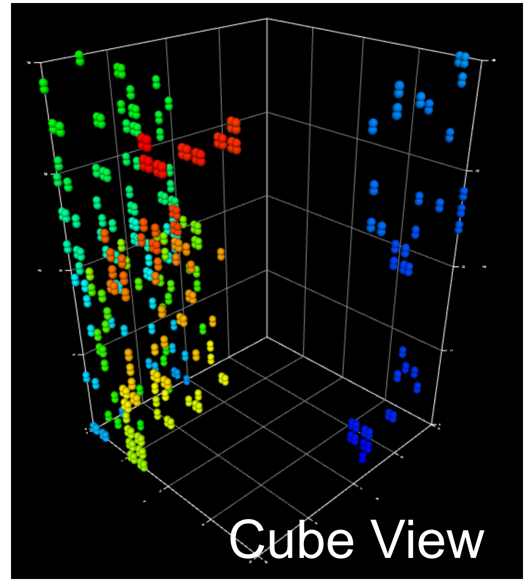
ParaProf:Topology View (6D Torus)



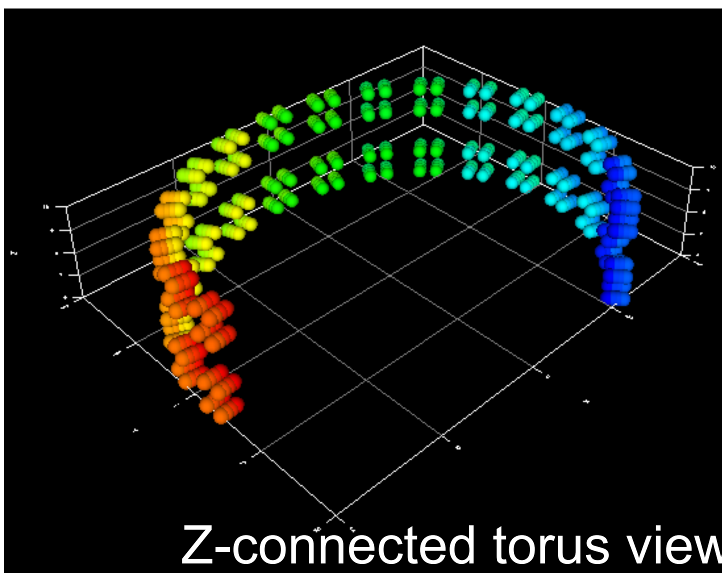
MPI Rank placement in Cray XE6 Topology

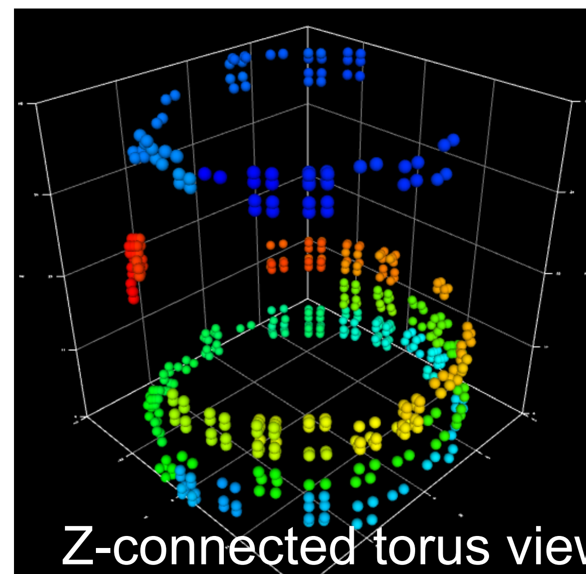


Fastest case CCSM:
4:14:08
(hh:mm:ss)



Slowest case:
5:35:50
(hh:mm:ss)





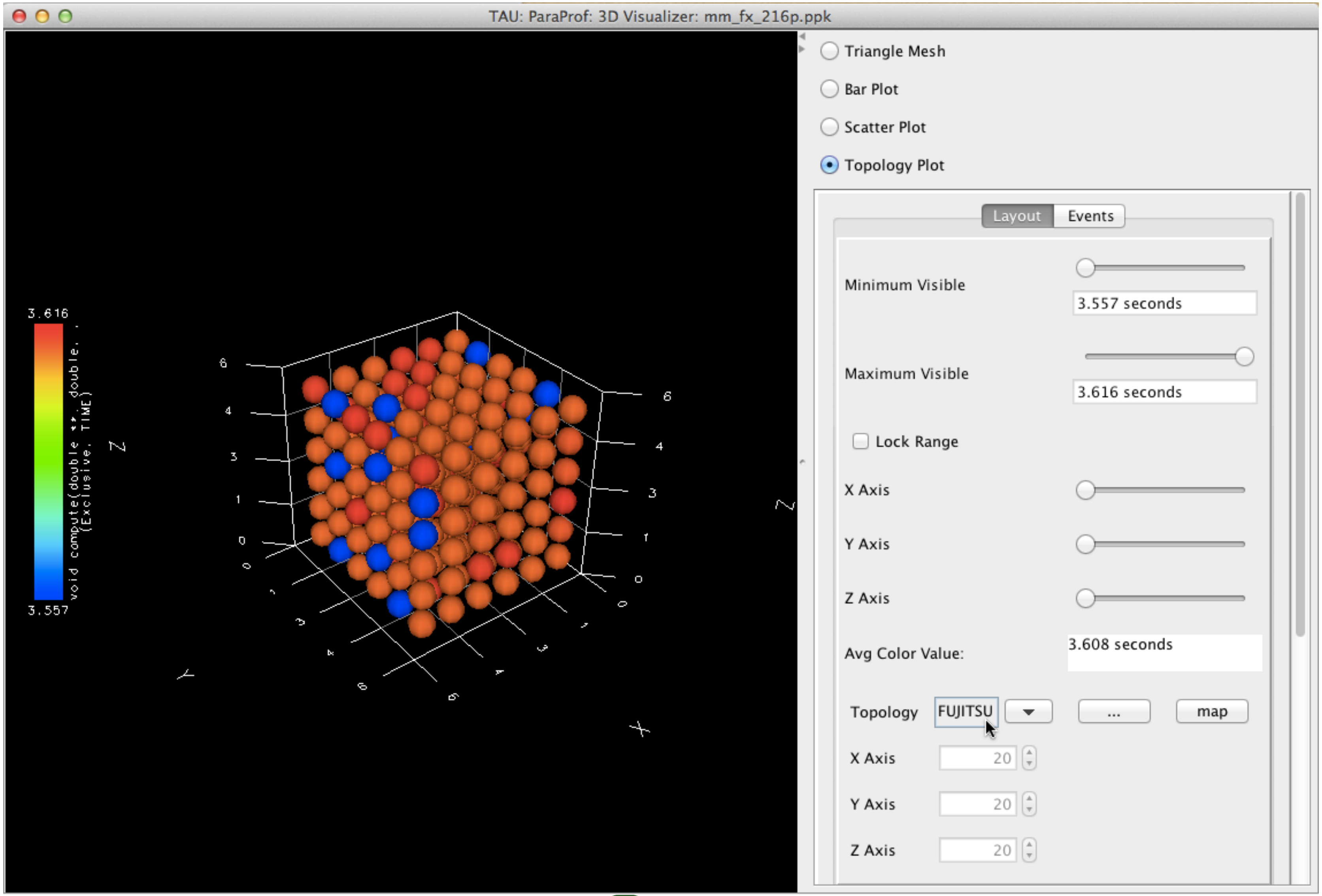
TAU's MetaData Collection on Fujitsu

Metadata for n,c,t 9,0,0

Name	Value
FUJITSU Coords	(3,1,0)
FUJITSU Dimension	3
FUJITSU Size	(6,6,6)
File Type Index	0
File Type Name	ParaProf Packed Profile
Hostname	e09t14226
Local Time	2012-11-12T02:14:16+09:00
MPI Processor Name	e09t14226
Memory Size	32836968 kB
Node Name	e09t14226
OS Machine	s64fx
OS Name	Linux
OS Release	2.6.25.8
OS Version	#1 SMP Tue Sep 11 11:04:02 JST 2...
Starting Timestamp	1352654056461761
TAU Architecture	sparc64fx

% pjsub –interact –L node="6x6x6"
% mpiexec –n 216 ./a.out

Fujitsu FX10 (K computer)



UNIVERSITY OF OREGON

Issues of data reduction

- TAU can store the mean profile in the TAUdb instead of data from each core
- Efficient access for high core count data
- Cross experiment analysis views tailored for mean data
- Dimension reduction can further reduce data to salient events (e.g., exclusive time > 3%)
- Store metadata with performance data

UNIVERSITY OF OREGON

Issues of Scalability – File I/O

- **Each thread generates its own performance data**
- **How can we devise mechanisms to reduce this data?**
- **TAU_PROFILE_FORMAT="merged" uses MPI_Reduce at the end of the application and computes:**
  - Min, max, mean, total, std. deviation
  - Rank 0 writes a single file during MPI_Finalize
- **TAU_SUMMARY=1 generates a single file with data from:**
  - Node 0, node 1, and
  - Mean, max, min, total, std. deviation across all nodes
  - Get the essence of performance
- **TAU_LITE=1 uses a lightweight measurement core**
  - 50% lower overhead than TAU's default core
  - No support for sampling, throttling, tracing, just flat profiles

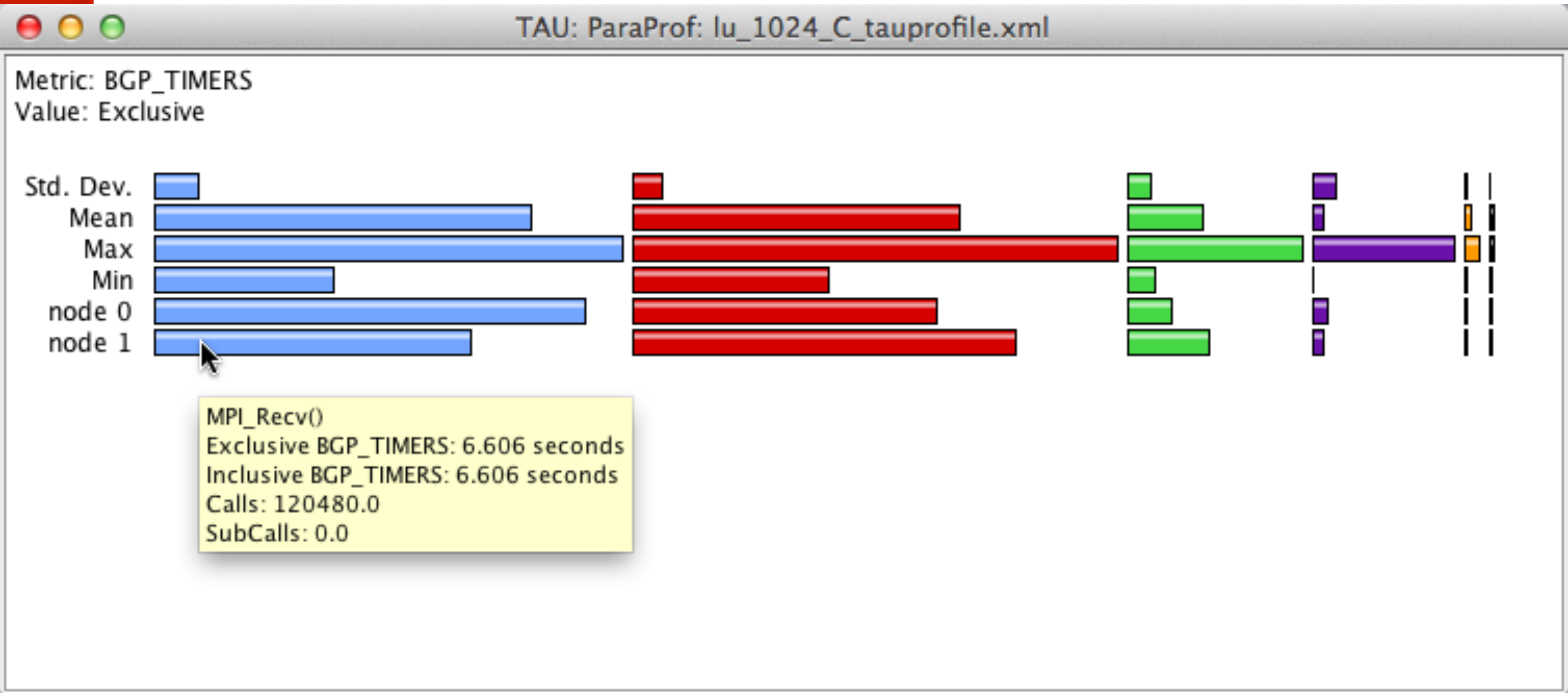
Communication Performance Summary

Calls			Inclusive BGP_TIMERS (Microseconds)			Bytes Transferred			Name
Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	
1.000	1.000	1.000	16669810.12471	16922166.95294	16749888.33824	0.00000	0.00000	0.00000	.TAU application
80320.000	160640.000	155620.000	3756604.90471	9764019.05294	7850486.06824	160.00000	240.00000	199.58333	MPI_Recv()
80828.000	161658.000	156606.309	569994.65765	3645268.14000	1595452.31161	80.00000	77760.00000	611.42134	MPI_Send()
508.000	1018.000	986.309	11133.31529	2986615.32118	252510.39040	80.00000	77760.00000	65587.87739	MPI_Wait()
10.000	10.000	10.000	3881.93412	69391.22588	40817.46760	8.00000	40.00000	24.00000	MPI_Allreduce()
2.000	2.000	2.000	18.84706	35.60706	29.68785	0.00000	0.00000	0.00000	MPI_Barrier()
9.000	9.000	9.000	133.23294	2351.46000	2346.84256	4.00000	40.00000	10.22222	MPI_Bcast()
1.000	1.000	1.000	0.59059	0.74588	0.63449	0.00000	0.00000	0.00000	MPI_Comm_rank()
1.000	2.000	1.001	0.47059	2.31059	0.51387	0.00000	0.00000	0.00000	MPI_Comm_size()
1.000	1.000	1.000	39731.75765	41183.16706	39770.34094	0.00000	0.00000	0.00000	MPI_Finalize()
1.000	1.000	1.000	60347.44588	312700.78706	140419.99977	0.00000	0.00000	0.00000	MPI_Init()
508.000	1018.000	986.309	1861.14000	6877.39176	4489.96484	0.00000	0.00000	0.00000	MPI_Irecv()

Summary data generated from a 1024 core execution on IBM BG/P using TAU_SUMMARY=1 and TAU_PROFILE_FORMAT="merged" based reduction

% export PATH=/soft/apps/tau/tau2/bgp/bin/compilers:$PATH
Build your code as is. TAU provides mpixlf90_r, mpixlc_r, mpixlcxx_r, etc. scripts
No changes to build or source code.

ParaProf Main Window (LU, 1024 cores)



Data Reduction Techniques

- TAU collects data from all native counters on IBM BG/P at MPI_Init and MPI_Finalize
- Computes mean, max, min for each counter at MPI_Finalize
- Stores this as metadata
- Discards per-rank data

UNIVERSITY OF OREGON

Metadata

TAU: ParaProf Manager

- Applications
  - Standard Applications
    - Default App
      - Default Exp
        - lu_1024_C_tau|
          - BGP_TIMERS
  - Default (jdbc:h2:/Users/s
  - perfexplorer_working (jdk

TrialField	Value		
Name	lu_1024_C_tauprofile.xml		
Application ID	0		
Experiment ID	0		
Trial ID	0		
BGP Coords	(0,0,0)		
BGP DDRSize (MB)	2048		
BGP Location	R00-M1-N00-J23		
BGP Node Mode	Virtual		
BGP Processor ID	1		
BGP Size	(8,4,8)		
BGP isTorus	(0,0,0)		
BGP numPsets	256		
BGP psetNum	0		
BGP psetSize	64		
BGP rankInPset	64		
BGP_COL_AC_CH0_VC0_MATURE	0	24	13
BGP_COL_AC_CH0_VC0_MATURE_UM1	0	24	13
BGP_COL_AC_CH0_VC1_MATURE	0	174	6
BGP_COL_AC_CH0_VC1_MATURE_UM1	0	91	9
BGP_COL_AC_CH1_VC0_MATURE	0	24	15
BGP_COL_AC_CH1_VC0_MATURE_UM1	0	24	13
BGP_COL_AC_CH1_VC1_MATURE	0	174	9
BGP_COL_AC_CH1_VC1_MATURE_UM1	0	174	9
BGP_COL_AC_CH2_VC0_MATURE	0	24	19
BGP_COL_AC_CH2_VC0_MATURE_UM1	0	24	19
BGP_COL_AC_CH2_VC1_MATURE	0	174	11
BGP_COL_AC_CH2_VC1_MATURE_UM1	0	91	8
BGP_COL_AC_INJECT_VC0_MATURE	24	24	24
BGP_COL_AC_INJECT_VC0_MATURE_UM1	24	24	24
BGP_COL_AC_INJECT_VC1_MATURE	0	0	0
BGP_COL_AC_INJECT_VC1_MATURE_UM1	0	174	1
BGP_COL_AC_PENDING_REQUESTS	48	14252911	2718122
BGP_COL_AC_WAITING_REQUESTS	48	313	74
BGP_COL_ALC_PACKET_TAKEN	24	24	24
BGP_COL_AR0_BAD_PACKET_MARKER	0	0	0
BGP_COL_AR0_HEADER_PARITY_ERROR	0	0	0
BGP_COL_AR0_IDLE_PACKET	107140	107144	107141
BGP_COL_AR0_PACKET_TAKEN	0	198	19
BGP_COL_AR0_RESYNC	0	0	0
BGP_COL_AR0_UNEXPECTED_HEADER_ERROR	0	0	0
BGP_COL_AR0_VC0_CUT_THROUGH	0	3120	1569
BGP_COL_AR0_VC0_DATA_PACKETS_RECEIVED	0	24	13
BGP_COL_AR0_VC0_EMPTY_PACKET	10499762	10606870	10530378

UNIVERSITY OF OREGON

File Formats for Extreme Scale

- Investigating an indexed profile file format
- Efficient startup of analysis tool
- Summary data presented at the beginning of file
- Hierarchical storage and presentation of profile data
- Score-P measurement library addresses scalability issues for tracing using OTF2

Support Acknowledgments

US Department of Energy (DOE)

- Office of Science contracts
- SciDAC, LBL contracts
- LLNL-LANL-SNL ASC/NNSA contract
- Battelle, PNNL contract
- ANL, ORNL contract

Department of Defense (DoD)

- HPCMO

National Science Foundation (NSF)

- Glassbox, SI-2

Partners:

University of Oregon

ParaTools, Inc.

University of Tennessee, Knoxville

T.U. Dresden, GWT

Juelich Supercomputing Center

Intel Exascale Labs, UVSQ, France

For more information

TAU Website:

http://tau.uoregon.edu

- Software
- Release notes
- Documentation

UNIVERSITY
OF OREGON