TAU Performance System and GPU Performance Measurement

Sameer Shende, Wyatt Spear, Scott Biersdorff, Suzanne Millstein
Performance Research Lab

Allen D. Malony
Department of Computer and Information Science
University of Oregon

Keeneland Tutorial, April 14-15, 2011

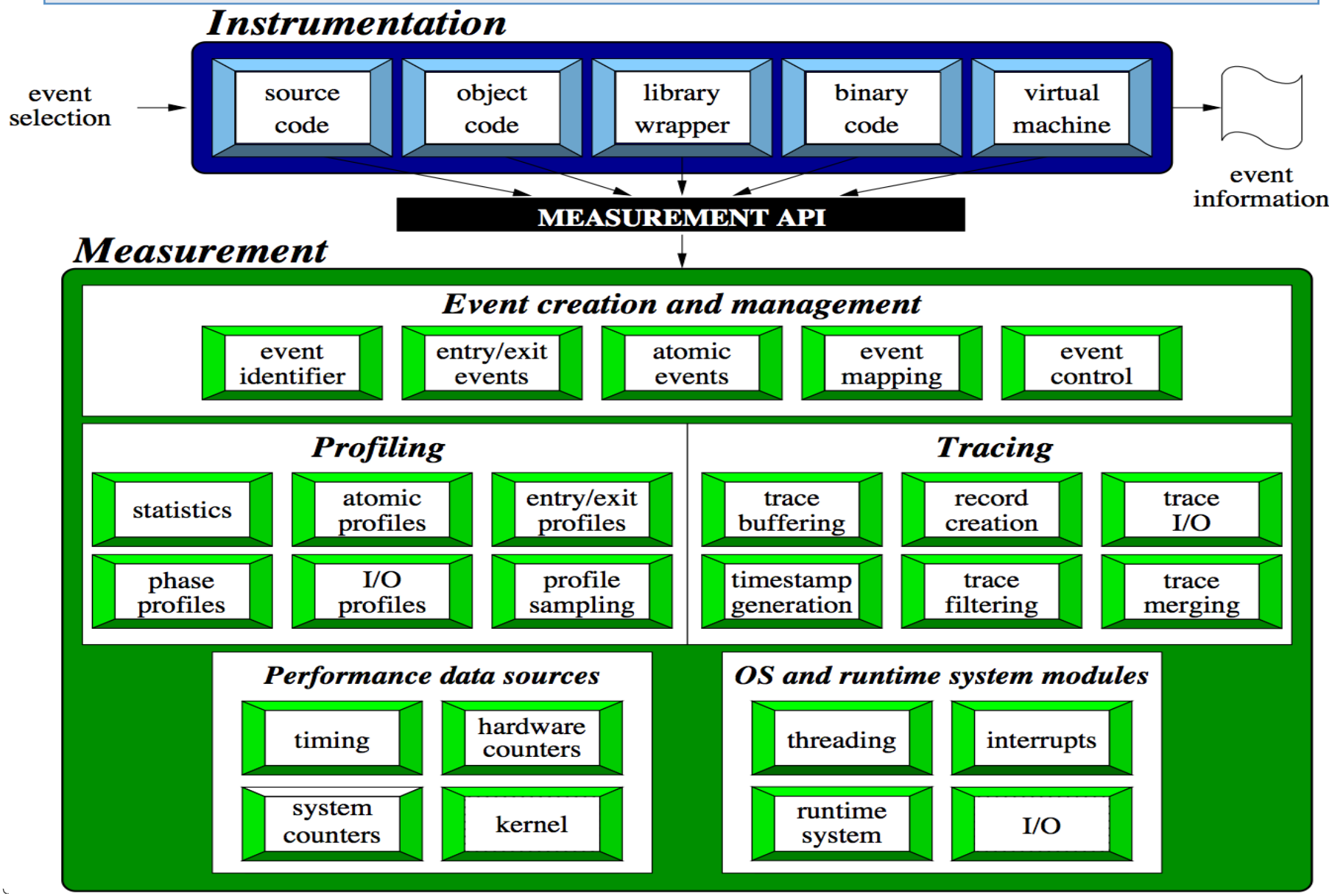
Outline

- Overview of the TAU Performance System
  - TAU is (being) installed on Keeneland (KID)
  - Provide background on what TAU is and how to use
  - Necessary for effectively using TAU with GPU
- Introduction to TAU support for GPU
  - Measurement approaches
  - Implementation mechanisms
  - Examples
- Getting started with TAU on KID
  - Brief discussion of hands-on
- TAU Appendix

TAU Performance System®

- *T*uning and *A*nalysis *U*tilities (18+ year project)
- Performance problem solving framework for HPC
  - Integrated, scalable, flexible, portable
  - Target all parallel programming / execution paradigms
- Integrated performance toolkit (open source)
  - Instrumentation, measurement, analysis, visualization
  - Widely-ported performance profiling / tracing system
  - Performance data management and data mining
- Broad application use (NSF, DOE, DOD, …)

TAU Instrumentation / Measurement



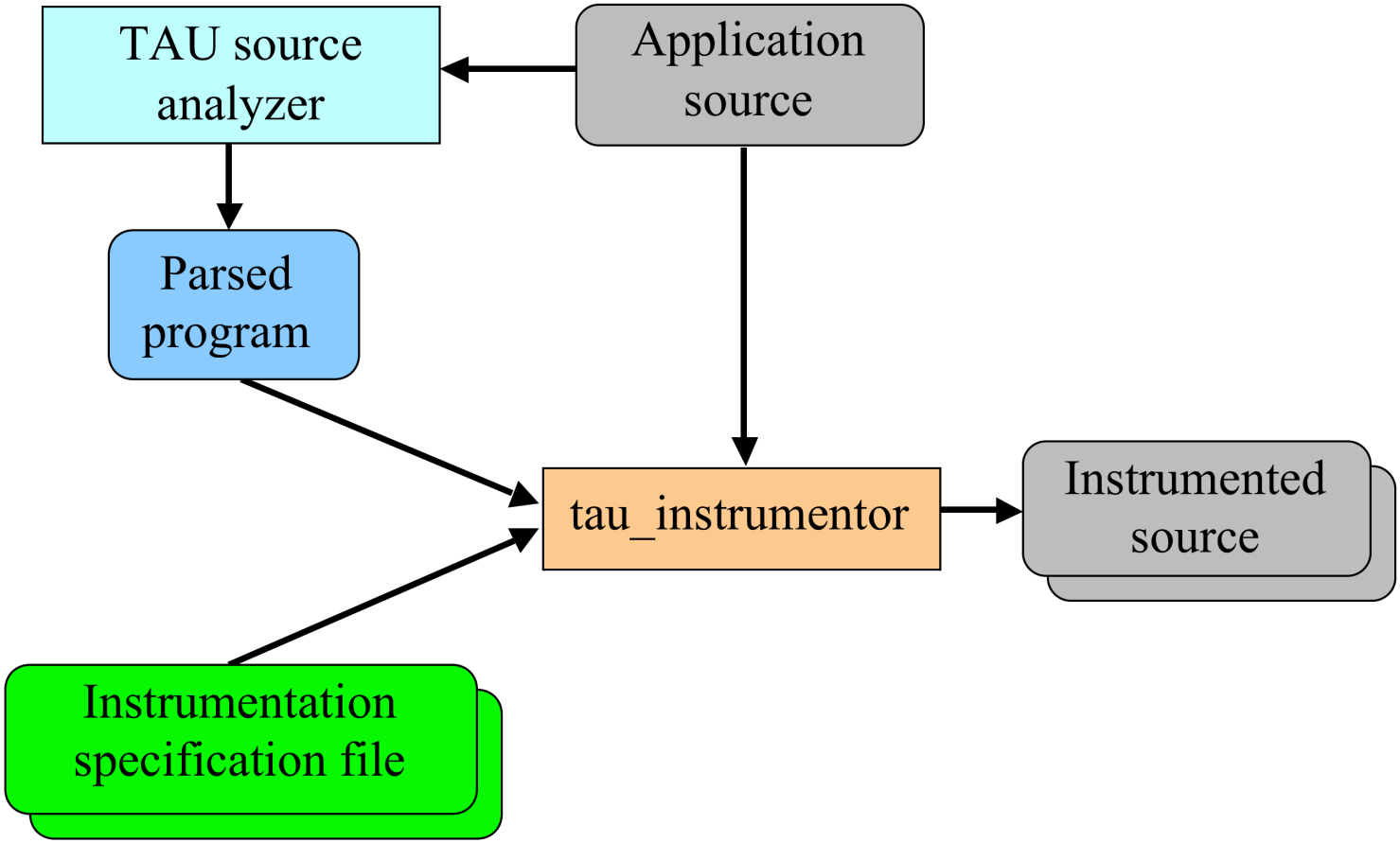
TAU Instrumentation Approach

- Direct instrumentation primarily
  - Indirect support forthcoming
- Support for standard program events
  - Routines, classes and templates
  - Statement-level blocks
  - Begin/End events (Interval events)
- Support for user-defined events
  - Begin/End events specified by user
  - Atomic events (e.g., size of memory allocated/freed)
  - Flexible selection of event statistics
- Provides static events and dynamic events
- Enables "semantic" mapping
- Specification of event groups (aggregation, selection)
- Instrumentation optimization

TAU Instrumentation Mechanisms

- Source code
  - Manual (TAU API, TAU component API)
  - Automatic (robust)
    - C, C++, F77/90/95 (Program Database Toolkit (PDT))
    - OpenMP (directive rewriting (Opari), POMP2 spec)
    - Library header wrapping
- Object code
  - Pre-instrumented libraries (e.g., MPI using PMPI)
  - Statically- and dynamically-linked (with `LD_PRELOAD`)
- Executable code
  - Binary and dynamic instrumentation (Dyninst)
  - Virtual machine instrumentation (e.g., Java using JVMPI)
- TAU_COMPILER to automate instrumentation process

Automatic Source-level Instrumentation



Selective Instrumentation File

- Specify a list of events to exclude or include
- # is a wildcard in a routine name

```
BEGIN_EXCLUDE_LIST
Foo
Bar
D#EMM
END_EXCLUDE_LIST
BEGIN_INCLUDE_LIST
int main(int, char **)
F1
F3
END_INCLUDE_LIST
```

Selective Instrumentation File

- Optionally specify a list of files
- * and ? may be used as wildcard characters

```
BEGIN_FILE_EXCLUDE_LIST
f*.f90
Foo?.cpp
END_FILE_EXCLUDE_LIST
BEGIN_FILE_INCLUDE_LIST
main.cpp
foo.f90
END_FILE_INCLUDE_LIST
```

MPI Wrapper Interposition Library

- Uses standard MPI Profiling Interface
  - Provides name shifted interface
    - MPI_Send = PMPI_Send
    - Weak bindings
- Create TAU instrumented MPI library
  - Interpose between MPI and TAU
  - Done during program link
    - `-lmpi` replaced by `-lTauMpi -lpmpi -lmpi`
  - No change to the source code!
  - Just re-link application to generate performance data

MPI Shared Library Instrumentation

- Interpose the MPI wrapper library for applications that have already been compiled
  - Avoid re-compilation or re-linking
- Requires shared library MPI
  - Uses `LD_PRELOAD` for Linux
  - On AIX use `MPI_EUILIB` / `MPI_EUILIBPATH`
  - Does not work on XT3
- Approach will work with other shared libraries
- Use TAU `tau_exec`
  - `% mpirun -np 4 tau_exec a.out`

TAU Measurement Approach

- Portable and scalable parallel profiling solution
  - Multiple profiling types and options
  - Event selection and control (enabling/disabling, throttling)
  - Online profile access and sampling
  - Online performance profile overhead compensation
- Portable and scalable parallel tracing solution
  - Trace translation to OTF, EPILOG, Paraver, and SLOG2
  - Trace streams (OTF) and hierarchical trace merging
- Robust timing and hardware performance support
- Multiple counters (hardware, user-defined, system)
- Performance measurement of I/O and Linux kernel

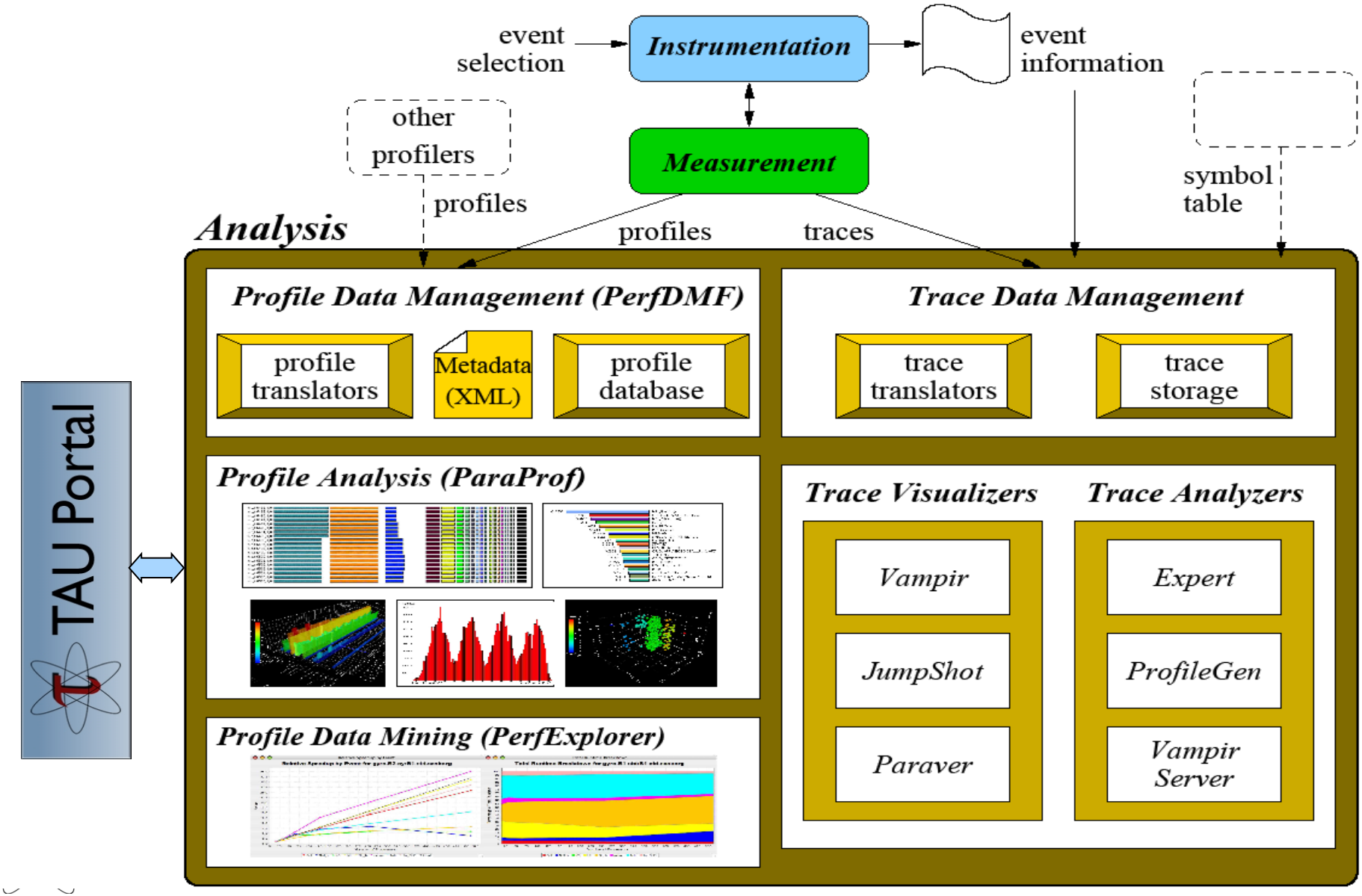
TAU Measurement Mechanisms

- Parallel profiling
  - Function-level, block-level, statement-level
  - Supports user-defined events and mapping events
  - Support for flat, callgraph/callpath, phase profiling
  - Support for parameter and context profiling
  - Support for tracking I/O and memory (library wrappers)
  - Parallel profile stored (dumped, shapshot) during execution
- Tracing
  - All profile-level events
  - Inter-process communication events
  - Inclusion of multiple counter data in traced events

Parallel Performance Profiling

- Flat profiles
  - Metric (e.g., time) spent in an event (callgraph nodes)
  - Exclusive/inclusive, # of calls, child calls
- Callpath profiles (Calldepth profiles)
  - Time spent along a calling path (edges in callgraph)
  - "main=> f1 => f2 => MPI_Send" (event name)
  - TAU_CALLPATH_DEPTH environment variable
- Phase profiles
  - Flat profiles under a phase (nested phases are allowed)
  - Default "main" phase
  - Supports static or dynamic (per-iteration) phases

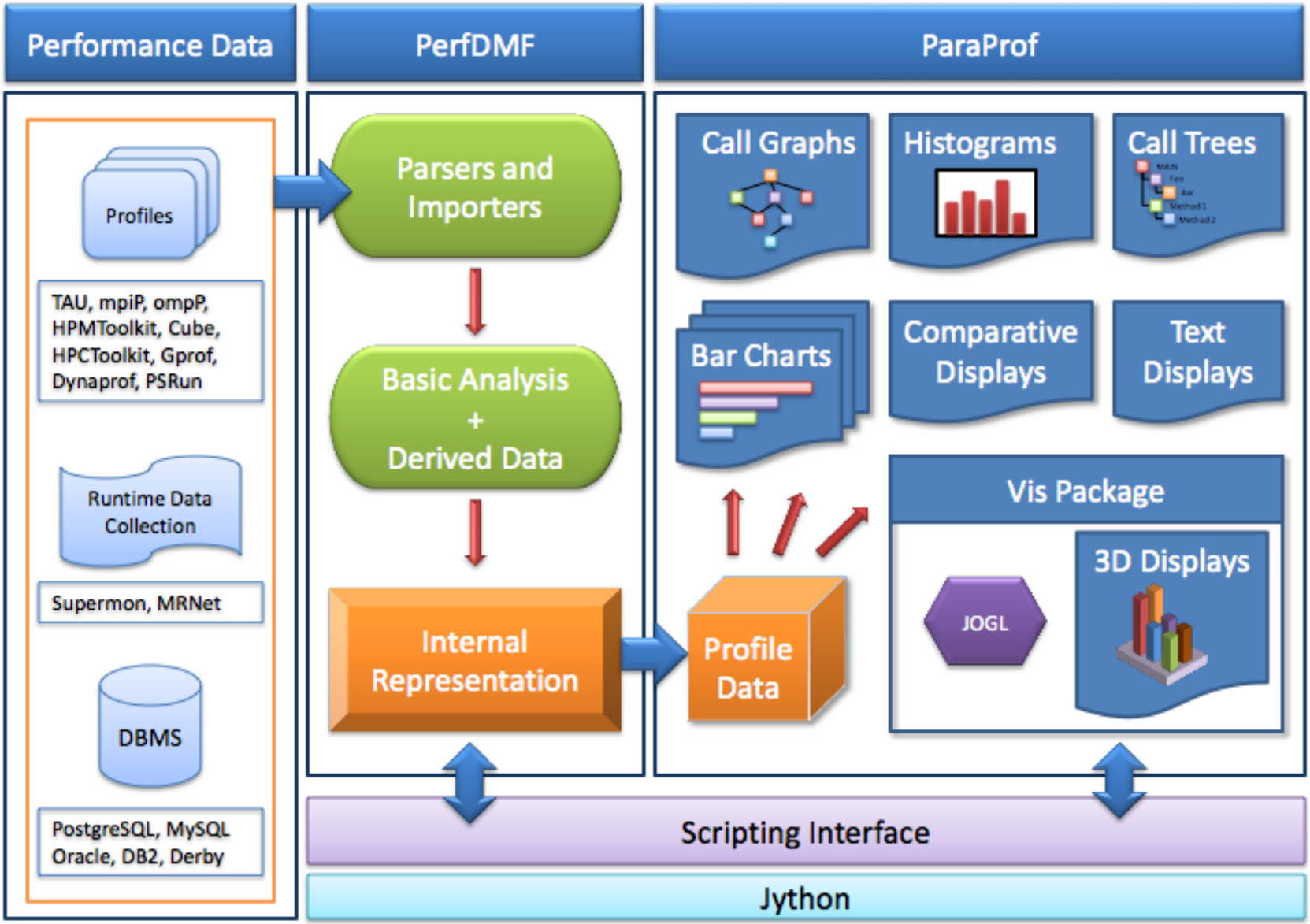
TAU Analysis



Performance Analysis

- Analysis of parallel profile and trace measurement
- Parallel profile analysis (*ParaProf*)
  - Java-based analysis and visualization tool
  - Support for large-scale parallel profiles
- Performance data management framework (*PerfDMF*)
- Performance data mining (*PerfExplorer*)
- Parallel trace analysis
  - Translation to VTF (V3.0), EPILOG, OTF formats
  - Integration with Vampir / Vampir Server (TU Dresden)
  - Profile generation from trace data
- Online parallel analysis and visualization
- Integration with CUBE browser (Scalasca, UTK / FZJ)

ParaProf Profile Analysis Framework



Parallel Profile Analysis – pprof

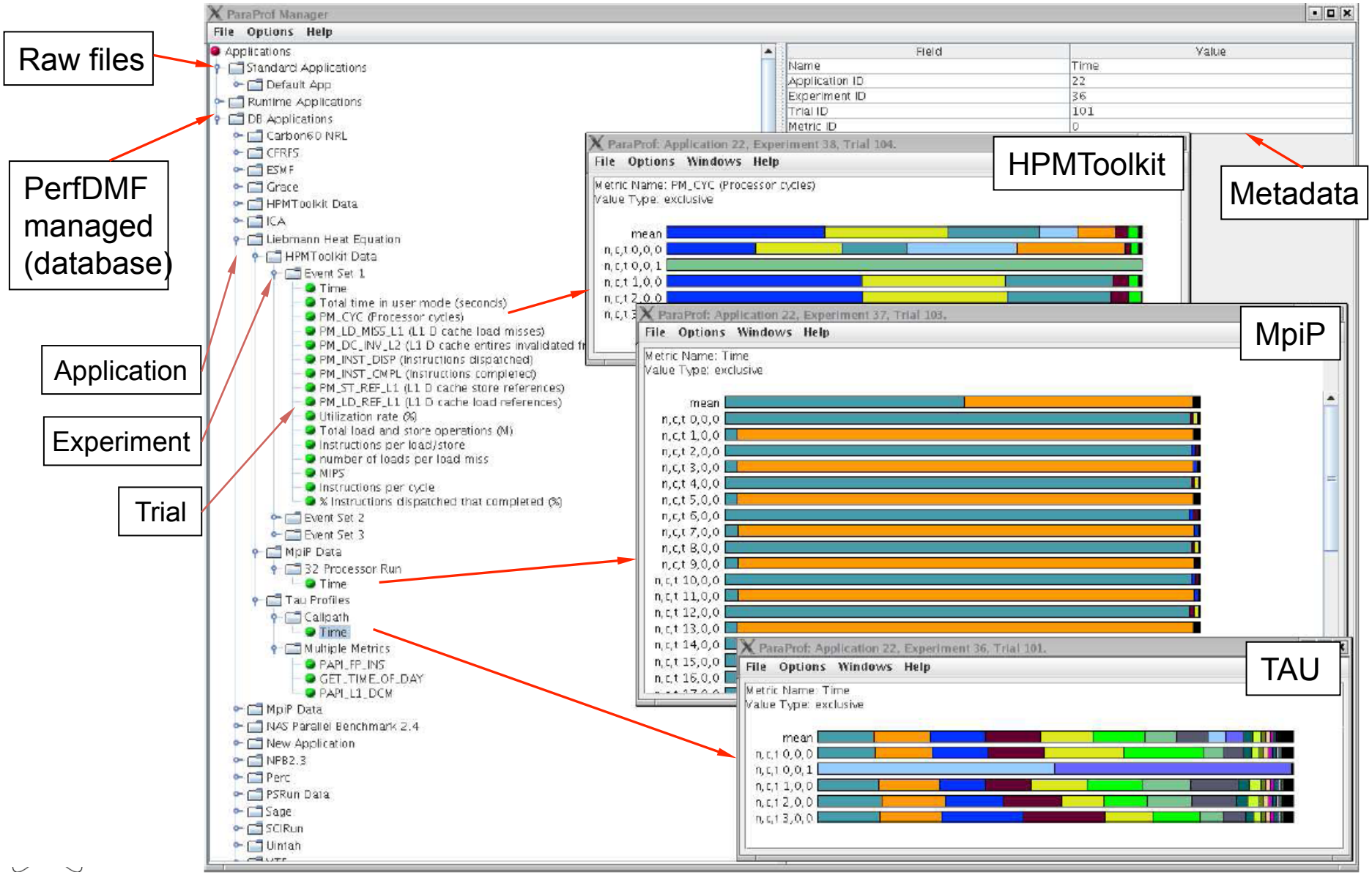
emacs@neutron.cs.uoregon.edu

Buffers Files Tools Edit Search Mule Help

```
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0            1     3:11.293           1          15  191293269 applu
 99.6        3,667     3:10.463           3       37517   63487925 bcast_inputs
 67.1          491     2:08.326       37200       37200       3450 exchange_1
 44.5        6,461     1:25.159        9300       18600       9157 buts
 41.0     1:18.436     1:18.436       18600           0       4217 MPI_Recv()
 29.5        6,778       56,407        9300       18600       6065 blts
 26.2       50,142       50,142       19204           0       2611 MPI_Send()
 16.2       24,451       31,031         301         602     103096 rhs
  3.9        7,501        7,501        9300           0        807 jacld
  3.4          838        6,594         604        1812      10918 exchange_3
  3.4        6,590        6,590        9300           0        709 jacu
  2.6        4,989        4,989         608           0       8206 MPI_Wait()
  0.2         0.44          400           1           4     400081 init_comm
  0.2          398          399           1          39     399634 MPI_Init()
  0.1          140          247           1       47616     247086 setiv
  0.1          131          131       57252           0          2 exact
  0.1           89          103           1           2     103168 erhs
  0.1        0.966           96           1           2      96458 read_input
  0.0           95           95           9           0      10603 MPI_Bcast()
  0.0           26           44           1        7937      44878 error
  0.0           24           24         608           0         40 MPI_Irecv()
  0.0           15           15           1           5      15630 MPI_Finalize()
  0.0            4           12           1        1700      12335 setbv
  0.0            7            8           3           3       2893 l2norm
  0.0            3            3           8           0        491 MPI_Allreduce()
  0.0            1            3           1           6       3874 pintgr
  0.0            1            1           1           0       1007 MPI_Barrier()
  0.0        0.116        0.837           1           4        837 exchange_4
  0.0        0.512        0.512           1           0        512 MPI_Keyval_create()
  0.0        0.121        0.353           1           2        353 exchange_5
  0.0        0.024        0.191           1           2        191 exchange_6
  0.0        0.103        0.103           6           0         17 MPI_Type_contiguous()
--:--  NPB_LU.out        (Fundamental)--L8--Top----------------------------------------
```

Parallel Profile Analysis – ParaProf



Metadata for Each Experiment

TAU: ParaProf Manager

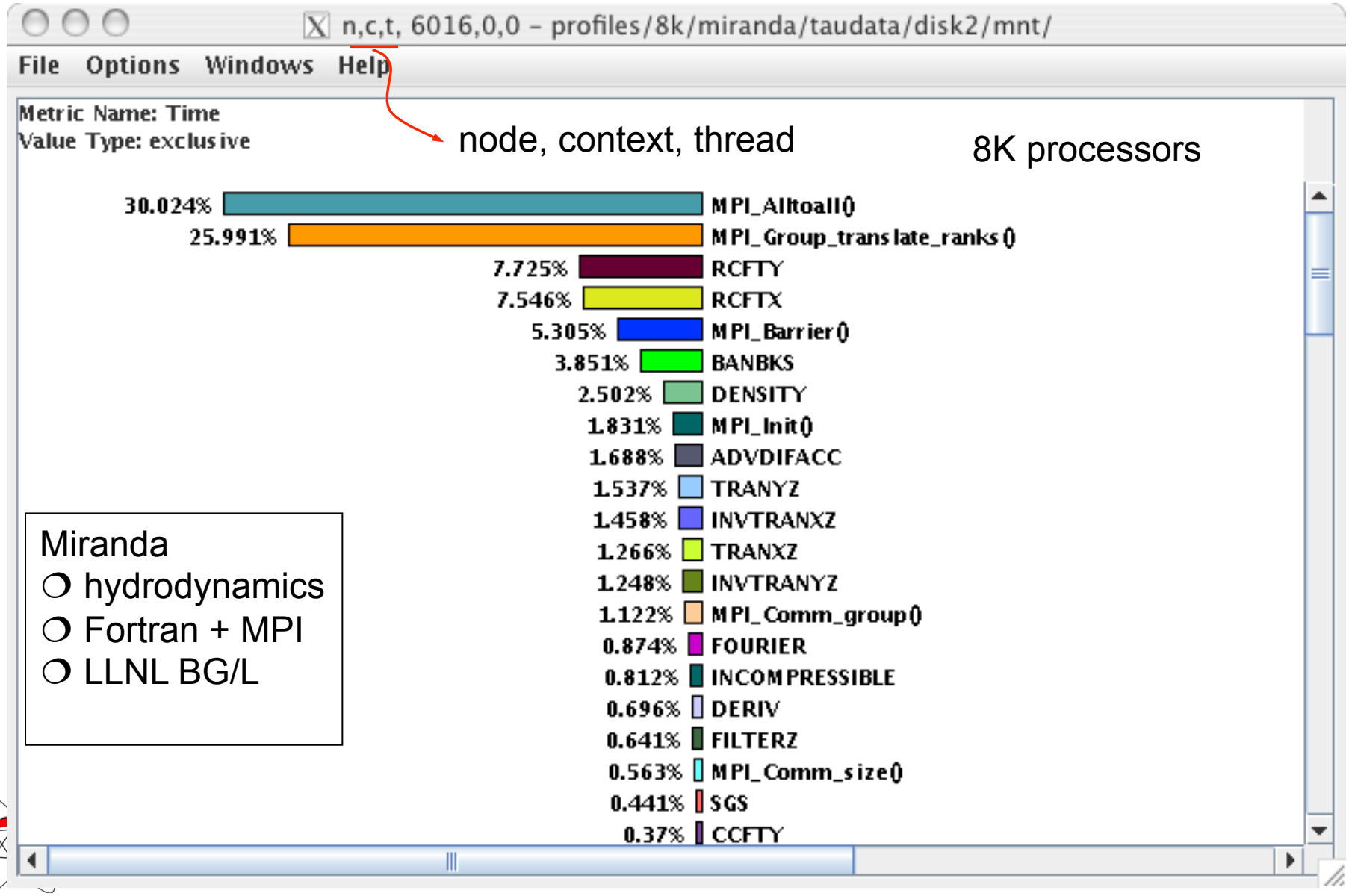
File Options Help

- Applications
  - Standard Applications
    - Default App
      - Default Exp
        - f90/pdt_mpi/examples/tau2/amorris/home/
          - PAPI_FP_OPS
          - GET_TIME_OF_DAY
  - Default (jdbc:postgresql://spaceghost.cs.uoregon.edu:5
  - utonium (jdbc:postgresql://utonium.cs.uoregon.edu:543
  - spaceghost2 (jdbc:postgresql://spaceghost.cs.uoregon.
  - proton_mysql (jdbc:mysql://192.168.1.1:3306/perfdm
  - spaceghost_peri_milc (jdbc:postgresql://spaceghost.cs.u
  - proton_postgresql (jdbc:postgresql://192.168.1.1:543
  - utonium_oracle (jdbc:oracle:thin:@//utonium.cs.uoregon.
  - perictc (jdbc:postgresql://spaceghost.cs.uoregon.edu:5

Multiple PerfDMF DBs

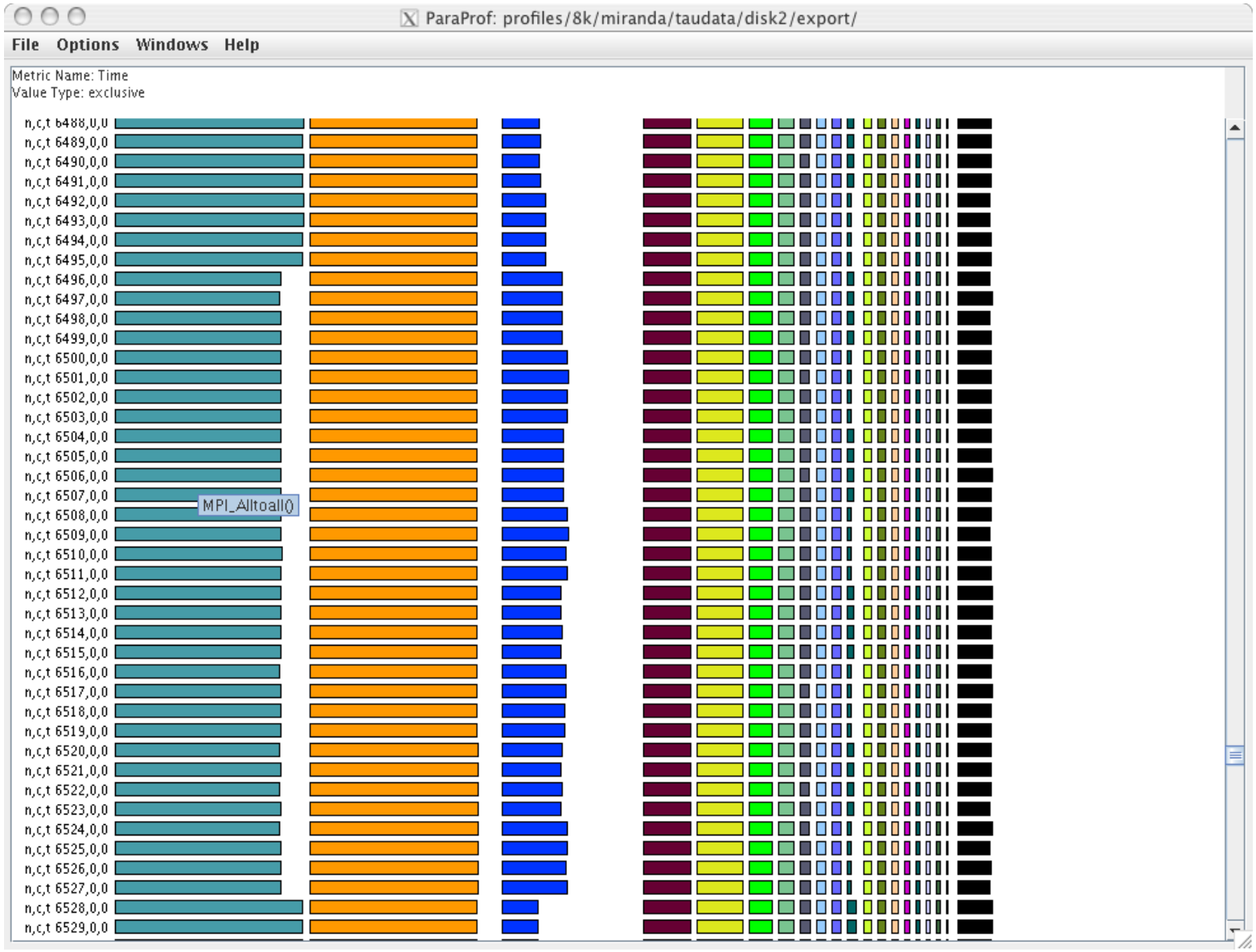
TrialField	Value
Name	f90/pdt_mpi/examples/tau2/amorris/home/
Application ID	0
Experiment ID	0
Trial ID	0
CPU Cores	2
CPU MHz	2992.505
CPU Type	Intel(R) Xeon(R) CPU 5160 @ 3.00GHz
CPU Vendor	GenuineIntel
CWD	/home/amorris/tau2/examples/pdt_mpi/f90
Cache Size	4096 KB
Executable	/home/amorris/tau2/examples/pdt_mpi/f...
Hostname	demon.nic.uoregon.edu
Local Time	2007-07-04T04:21:14-07:00
MPI Processor Name	demon.nic.uoregon.edu
Memory Size	8161240 kB
Node Name	demon.nic.uoregon.edu
OS Machine	x86_64
OS Name	Linux
OS Release	2.6.9-42.0.3.EL.perfctrsmp
OS Version	#1 SMP Fri Nov 3 07:34:13 PST 2006
Starting Timestamp	1183548072220996
TAU Architecture	x86_64
TAU Config	-papi=/usr/local/packages/papi-3.5.0 -M...
Timestamp	1183548074317538
UTC Time	2007-07-04T11:21:14Z
pid	11395
username	amorris

ParaProf – Flat Profile

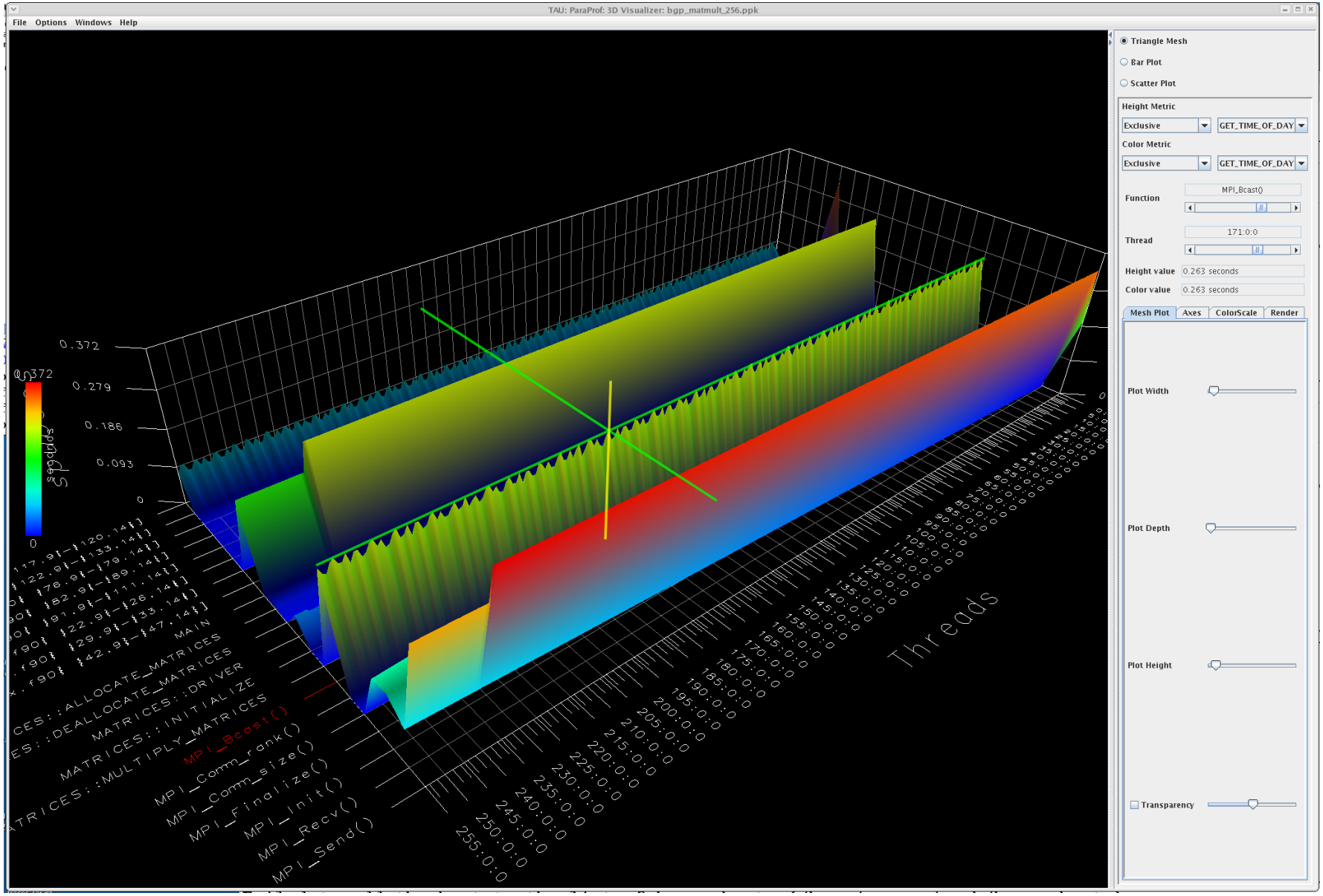


Miranda
- hydrodynamics
- Fortran + MPI
- LLNL BG/L

ParaProf – Stacked View

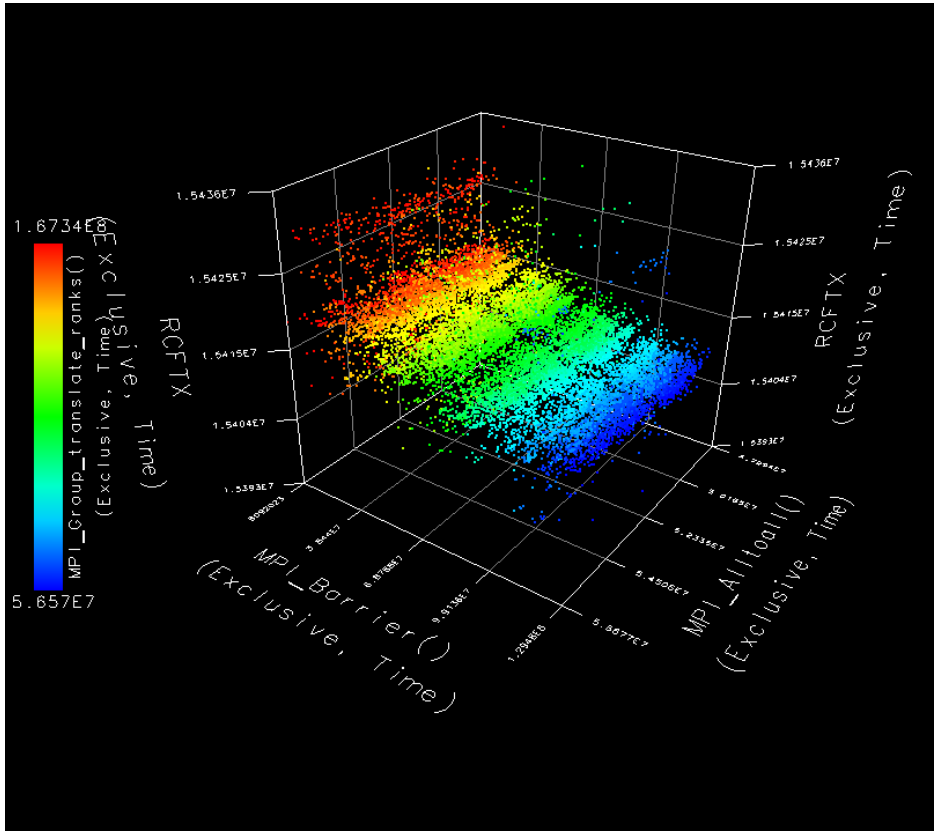


ParaProf – 3D View (Full Profile)



ParaProf – 3D Scatterplot

- Each point is a "thread" of execution
- A total of four metrics shown in relation
- ParaProf's visualization library
  - JOGL
- Miranda



Using TAU

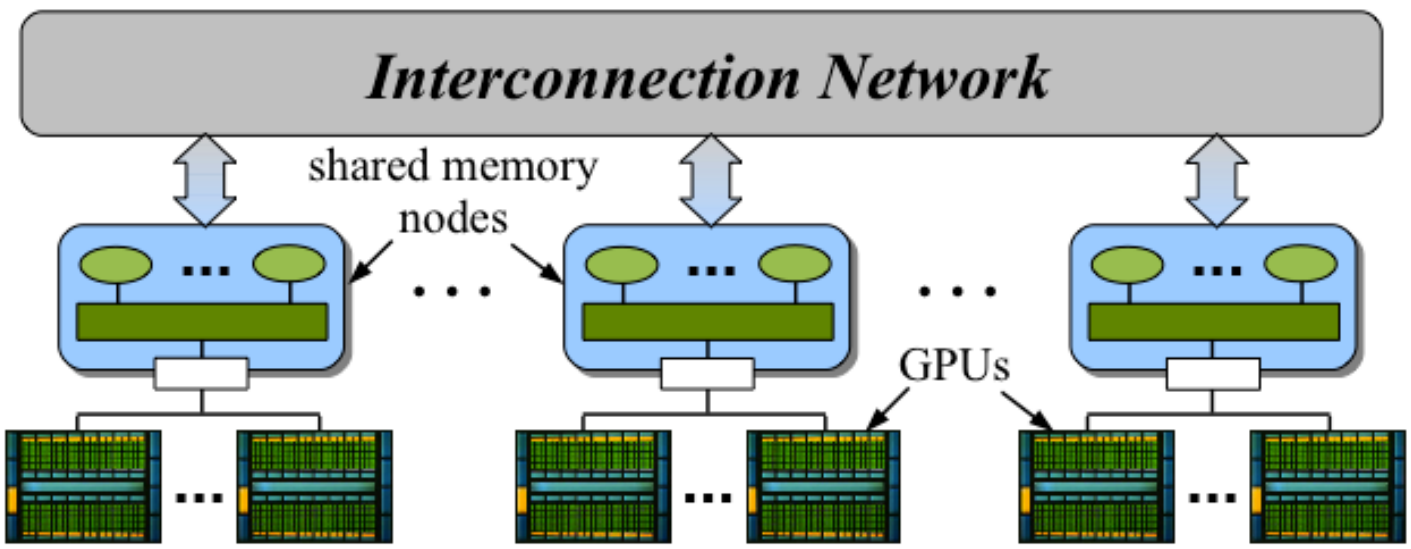
- Install TAU
  - `% configure [options]; make clean install`
- Modify application makefile and choose TAU configuration
  - Select TAU's stub makefile
  - Change name of compiler in makefile
- Set environment variables
  - Directory where profiles/traces are to be stored/counter selection
  - TAU options
- Execute application
  - `% mpirun -np <procs> a.out;`
- Analyze performance data
  - paraprof, vampir, pprof, paraver …

For more information

- TAU Website: http://tau.uoregon.edu
  - Software
  - Release notes
  - Documentation
- TAU LiveDVD: http://tau.uoregon.edu/point.iso
  - Boot up on your laptop or desktop
  - Includes TAU and variety of other packages
  - Include documentation and tutorial slides

Heterogeneous Architecture

- Multi-CPU, multicore shared memory nodes
- GPU accelerators connected by high-BW I/O
- Cluster interconnection network



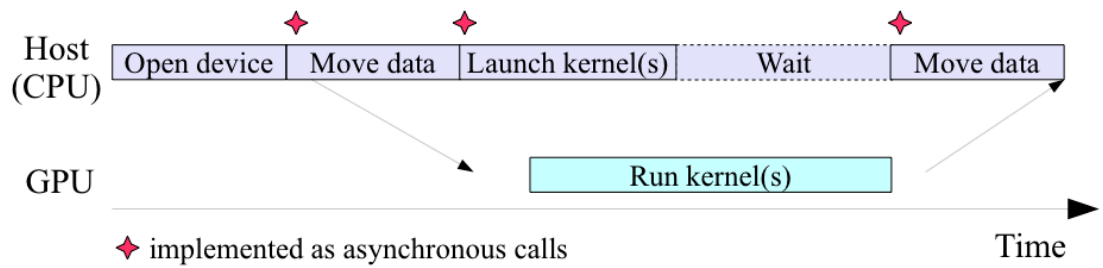
Heterogeneous Performance Measurement

Heterogeneous performance perspectives

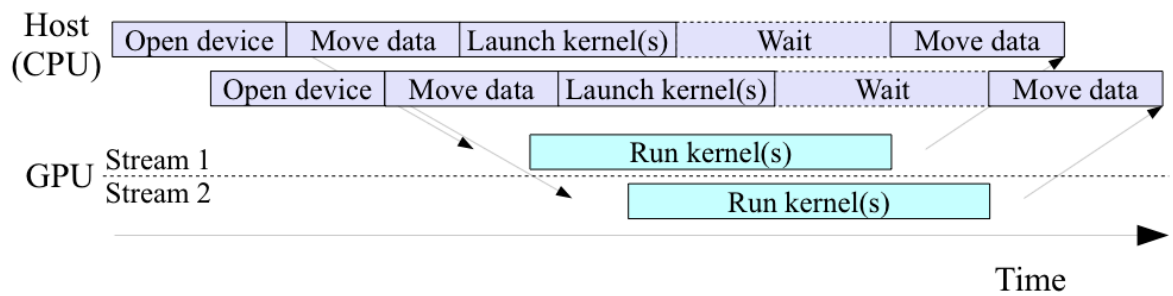
- Inter-node communication
  - Message communication, overhead, synchronization
- Intra-node execution
  - Multicore thread execution and interactions
- Host-GPU interactions
  - Kernel setup, memory transfer, concurrency overlap, synchronization
- GPU kernel execution

Host (CPU) - GPU Scenarios

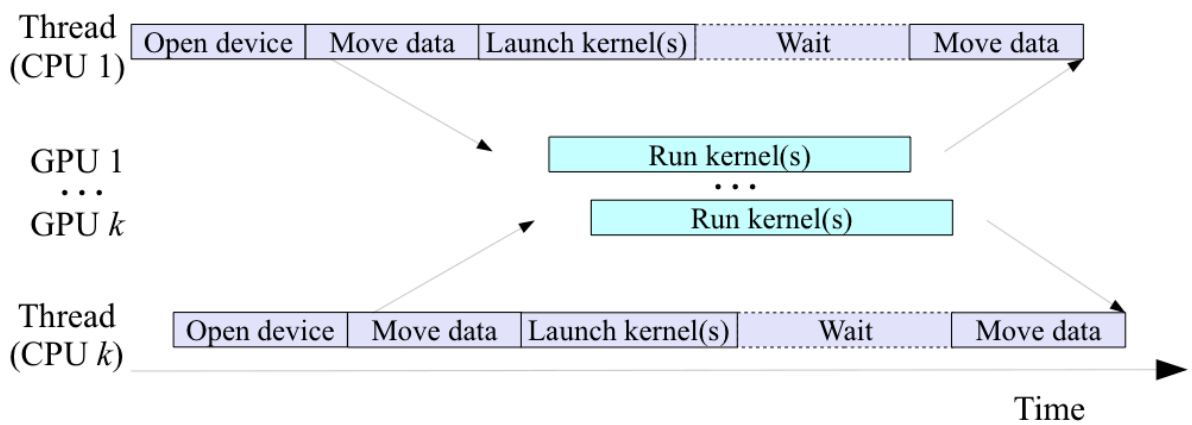
- Single GPU



- Multi-stream

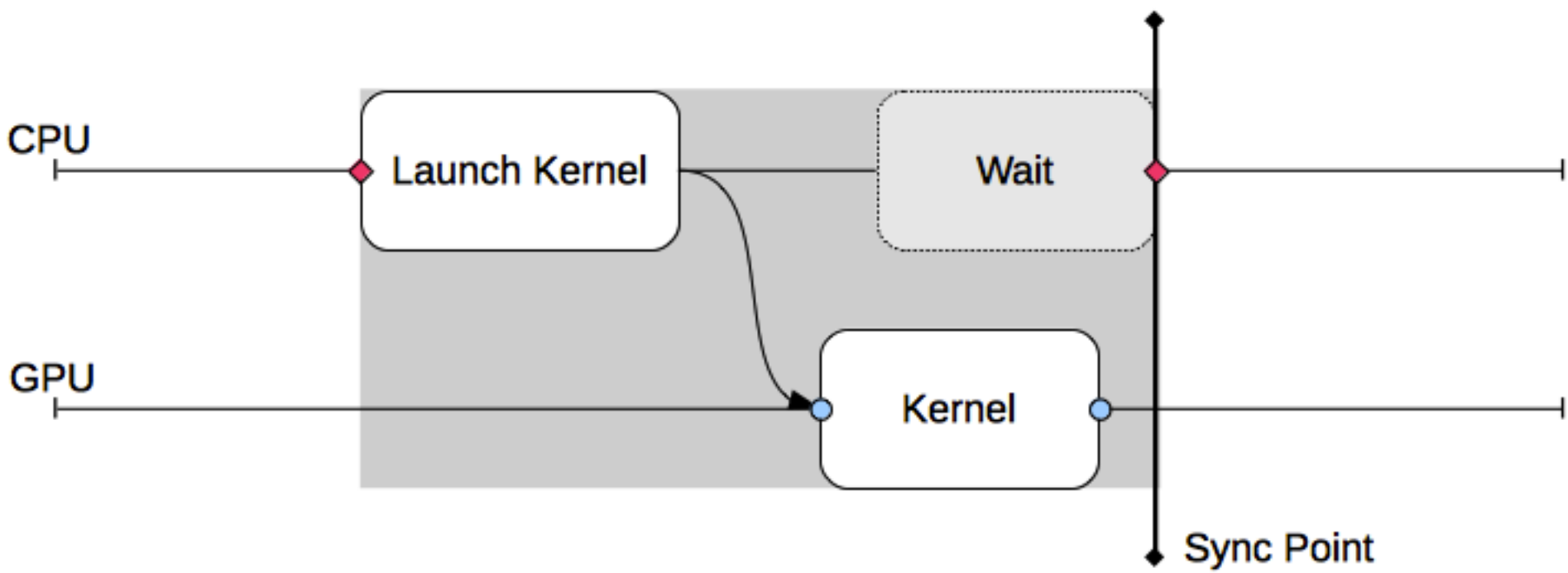


- Multi-CPU, Multi-GPU



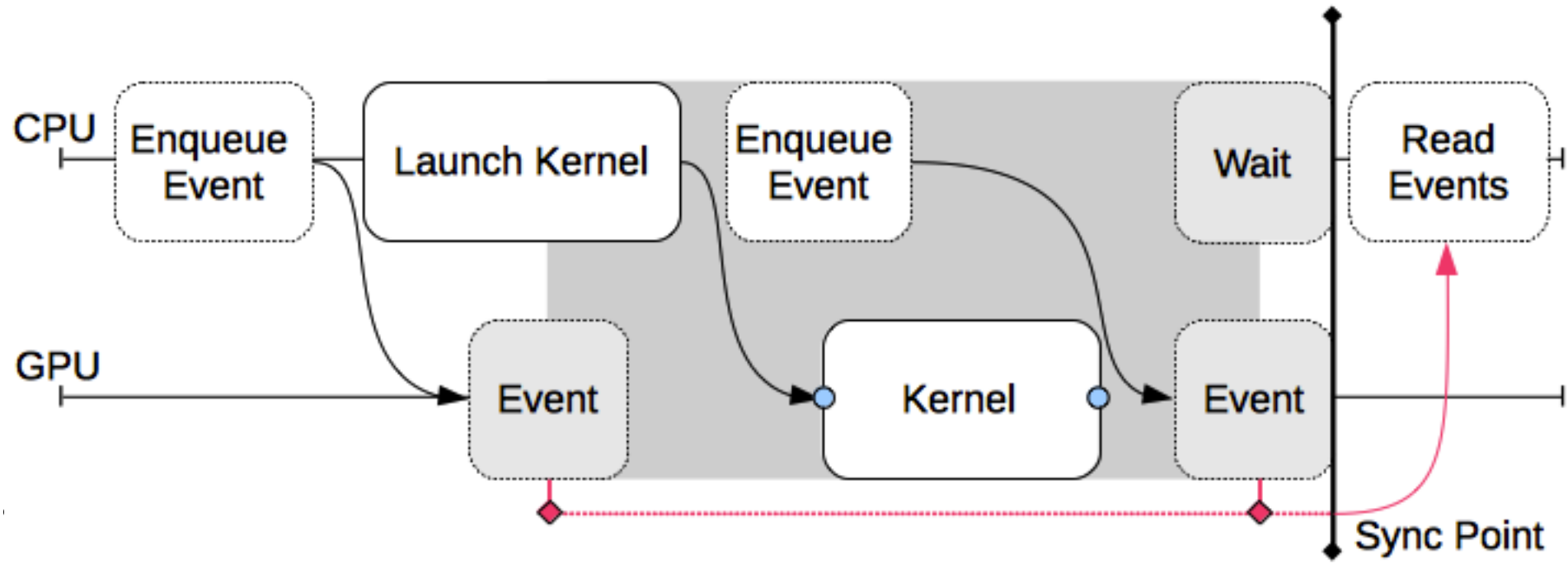
Host-GPU Measurement – Synchronous Method

- Consider three measurement approaches
- Synchronous events on the CPU ◆
- Some inaccuracy with actual kernel start/stop ○



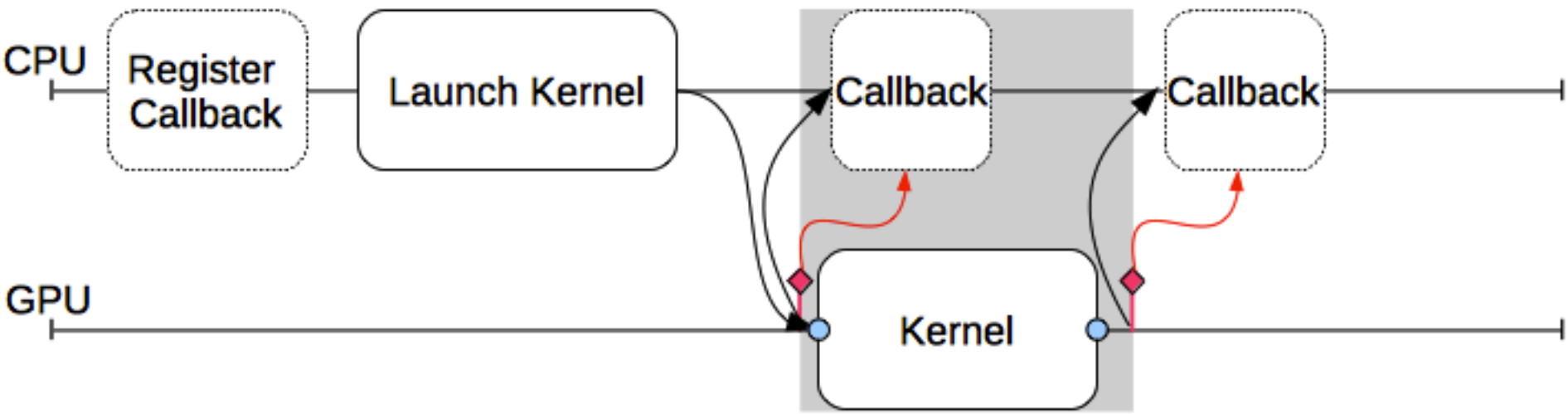
Host-GPU Measurement – Event Queue Method

- Events queued in GPU stream, measured by GPU, and read at sync points on CPU
- Events must be placed around kernel launch



Host-GPU Measurement – Callback Method

- GPU driver libraries provide callbacks for certain routines and captures measurements
- Measurement tool registers the callbacks and processes performance data
- Application code is not modified



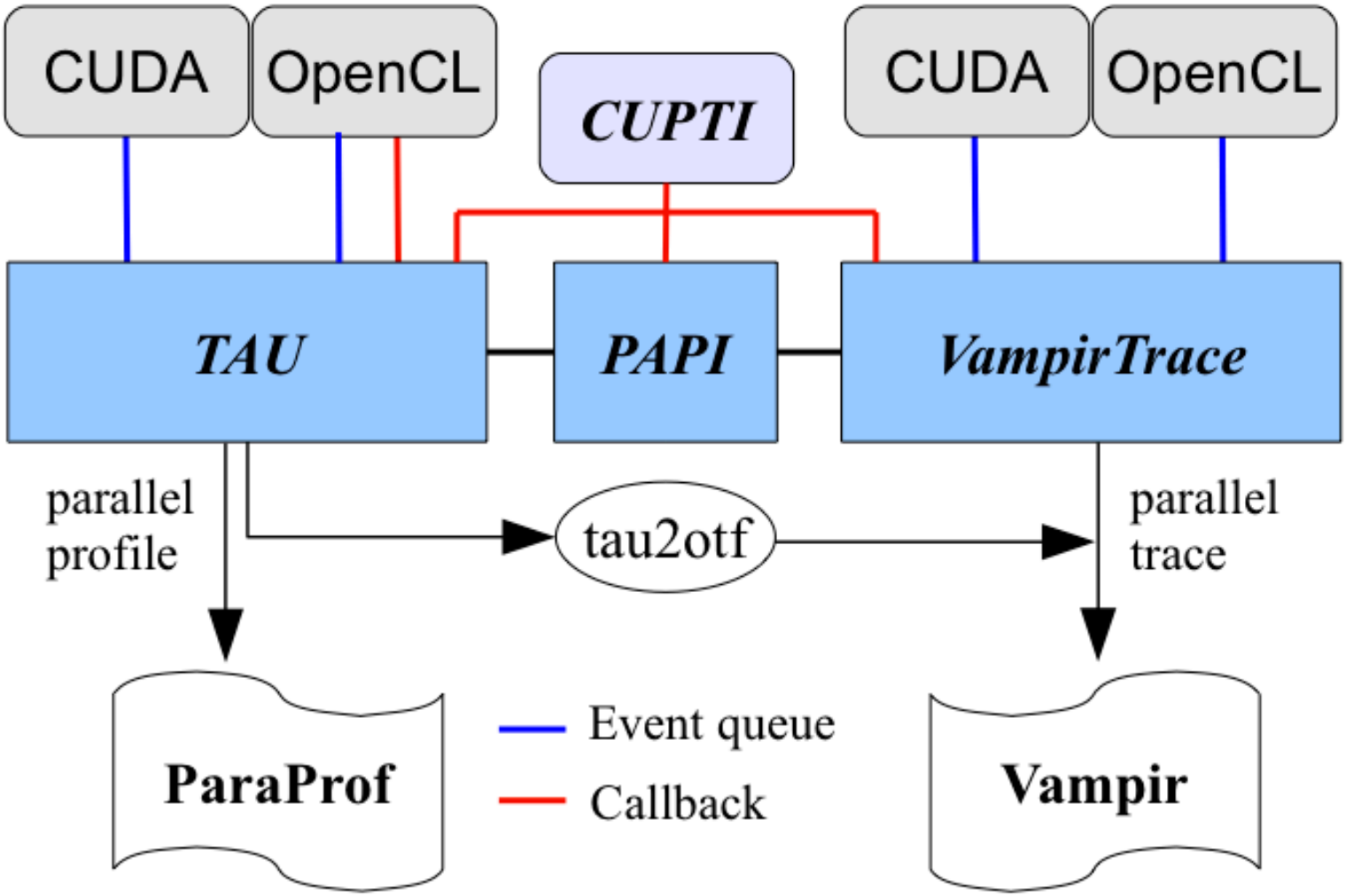
Method Support and Implementation

- Synchronous method
  - Place instrumentation appropriately around GPU calls (kernel launch, library routine, …)
  - Wrap (synchronous) library with performance tool
- Event queue method
  - Utilize CUDA and OpenCL event support
  - Again, need instrumentation to create and insert events in the streams with kernel launch and process events
  - Can be implemented with driver library wrapping
- Callback method
  - Utilize language-level callback support in OpenCL
  - Utilize NVIDIA CUDA Performance Tool Interface (CUPTI)
  - Need to appropriately register callbacks

GPU Performance Measurement Tools

- Support the Host-GPU performance perspective
- Provide integration with existing measurement system to facilitate tool use
- Utilize support in GPU driver library and device
- Tools
  - TAU performance system
  - Vampir
  - PAPI
  - NVIDIA CUPTI

GPU Performance Tool Interoperability



NVIDIA CUPTI

- NVIDIA is developing CUPTI to enable the creation of profiling and tracing tools
- Callback API
  - Interject tool code at the entry and exist to each CUDA runtime and driver API call
- Counter API
  - Query, configure, start, stop, and read the counters on CUDA-enabled devices
- CUPTI is delivered as a dynamic library
- CUPTI is expected to be released with CUDA 4.0

TAU for Heterogeneous Measurement

- Multiple performance perspectives
- Integrate Host-GPU support in TAU measurement framework
  - Enable use of each measurement approach
  - Include use of PAPI and CUPTI
  - Provide profiling and tracing support
- Tutorial
  - Use TAU library wrapping of libraries
  - Use `tau_exec` to work with binaries

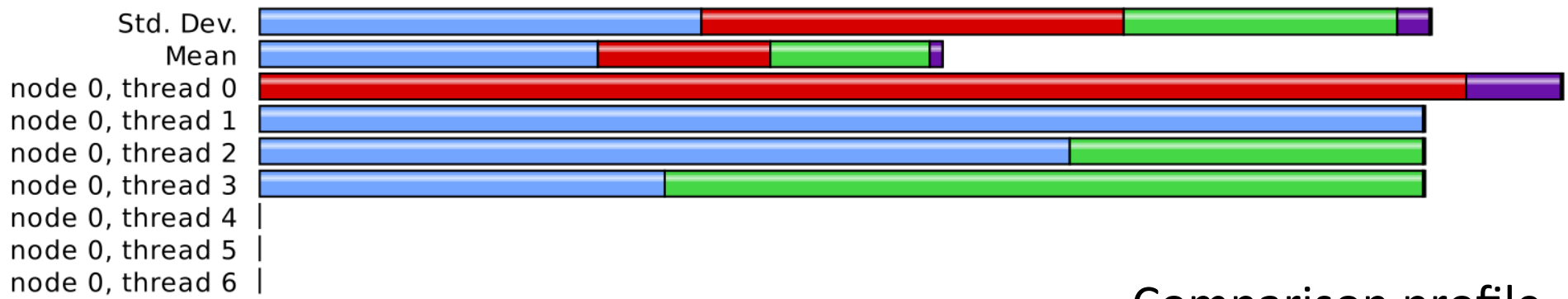
Example: SDK simpleMultiGPU

- Demonstration of multiple GPU device use
- *main* ➡ *solverThread* ➡ *reduceKernel*
- One Keeneland node with three GPUs
- Performance profile for:
  - One *main* thread
  - Three *solverThread* threads
  - Three *reduceKernel* "threads"

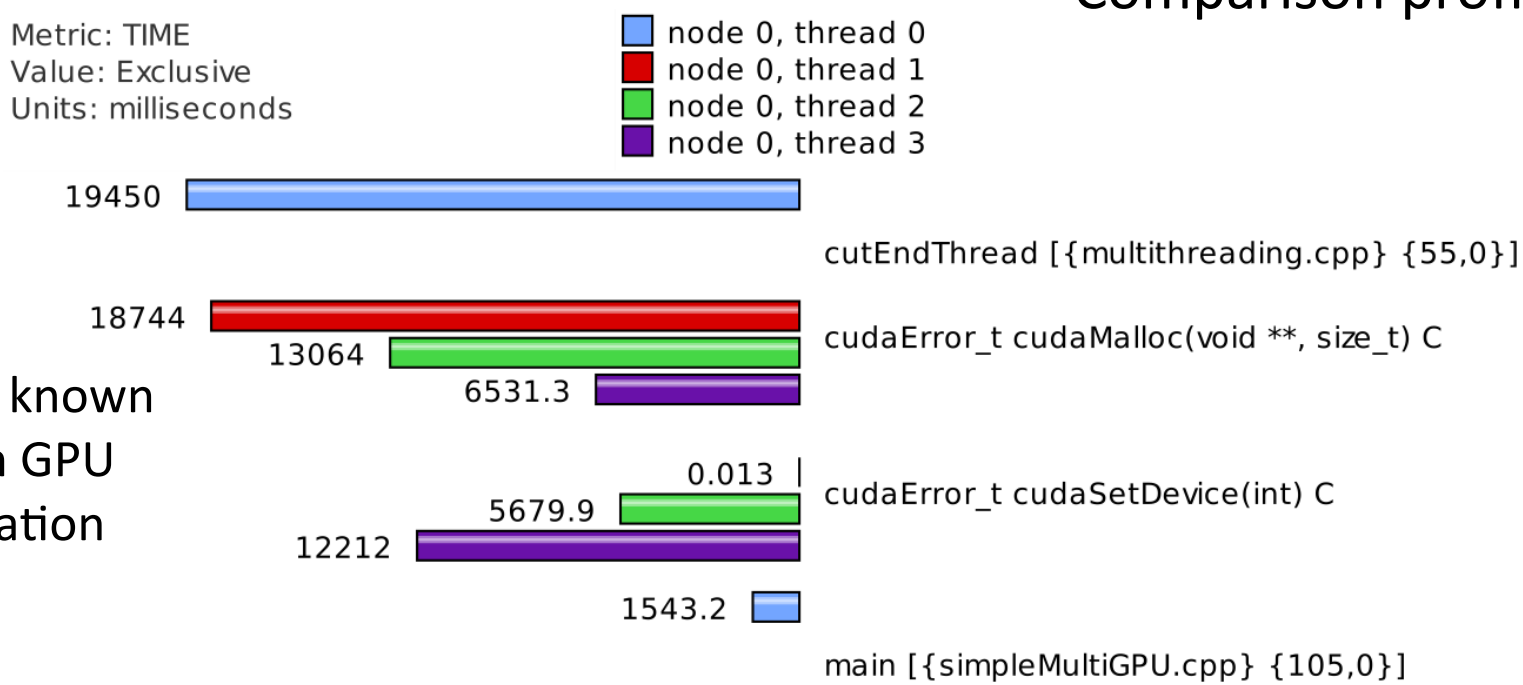
simpleMultiGPU Profile

Metric: TIME
Value: Exclusive

Overall profile



Comparison profile



Identified a known overhead in GPU context creation

simpleMultiGPU Profile

Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child Calls
main [{simpleMultiGPU.cpp} {105,0}]	1,543.18	20,997.786	12	17
cutWaitForThreads [{multithreading.cpp} {65,0}]	0.006	19,450.196	1	3
cutEndThread [{multithreading.cpp} {55,0}]	19,450.19	19,450.19	3	0
cudaError_t cudaGetDeviceCount(int *) C	4.342	4.342	1	0
cutStartThread [{multithreading.cpp} {48,0}]	0.064	0.064	3	0
cudaError_t cudaThreadExit(void) C	0.004	0.004	1	0

Main Thread

Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child Calls
solverThread(TGPUplan*) [{simpleMultiGPU.cpp} {47,0}]	0.103	18,767.447	9	21
cudaError_t cudaMalloc(void **, size_t) C	18,743.914	18,743.914	2	0
cudaError_t cudaMemcpy(void *, const void *, size_t, enum cudaMemcpyKind) C	20.108	20.108	2	0
cudaError_t cudaFree(void *) C	2.007	2.007	2	0
cudaError_t cudaThreadSynchronize(void) C	1.286	1.286	1	0
cudaError_t cudaSetDevice(int) C	0.013	0.013	1	0
cudaError_t cudaLaunch(const char *) C	0.012	0.012	1	0
cudaError_t cudaConfigureCall(dim3, dim3, size_t, cudaStream_t) C	0.002	0.002	1	0
cudaError_t cudaSetupArgument(const void *, size_t, size_t) C	0.002	0.002	3	0

Solver Thread

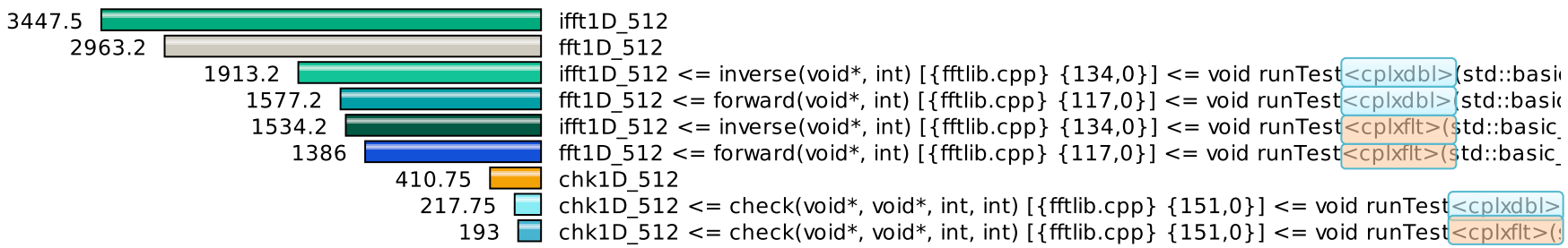
Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child Calls
.TAU application	0	0.865	1	1
reduceKernel	0.865	0.865	1	0

GPU Execution

SHOC FFT Profile with Callsite Info

- TAU is able to associate callsite context information with kernel launch so that different kernel calls can be distinguished

Metric: TAUGPU_TIME
Value: Exclusive
Units: microseconds



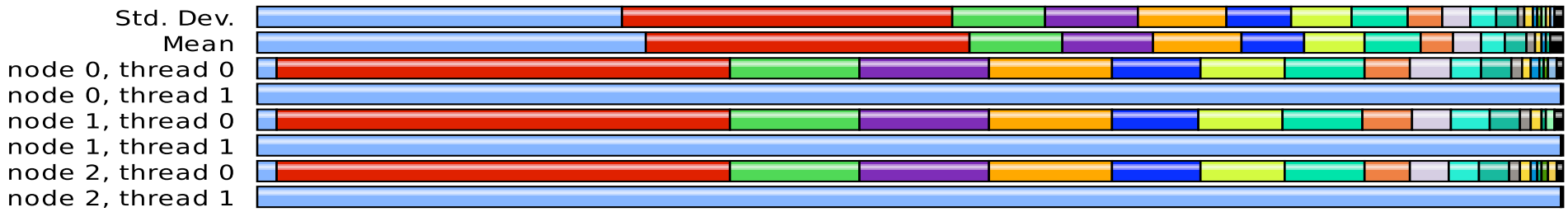
Each kernel (ifft1D_512, fft1D_512 and chk1D_512) is broken down by call-site, either during the single precession or double precession step.

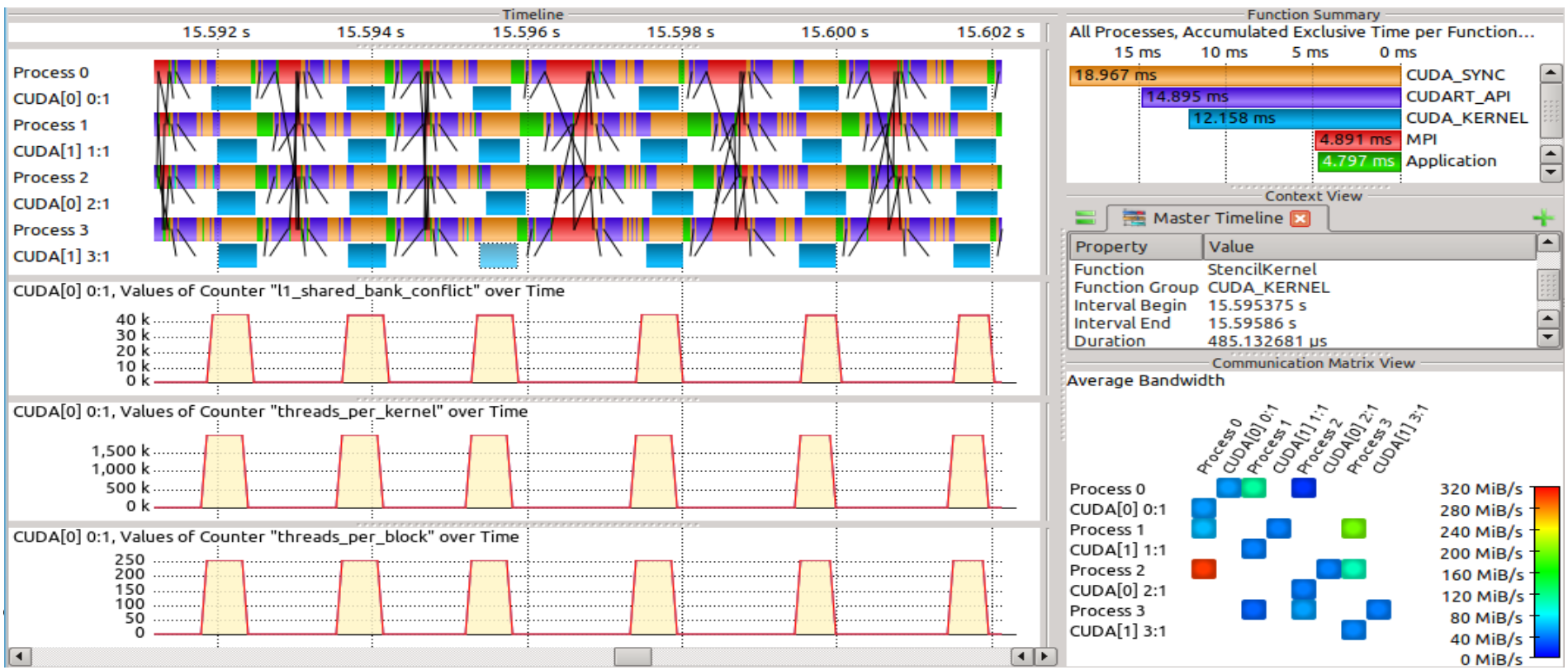
Example: SHOC Stencil2D

- Compute 2D, 9-point stencil
  - Multiple GPUs using MPI
  - CUDA and OpenCL versions
- One Keeneland node with 3 GPUs
- Eight Keeneland nodes with 24 GPUs
- Performance profile and trace
  - Application events
  - Communication events
  - Kernel execution

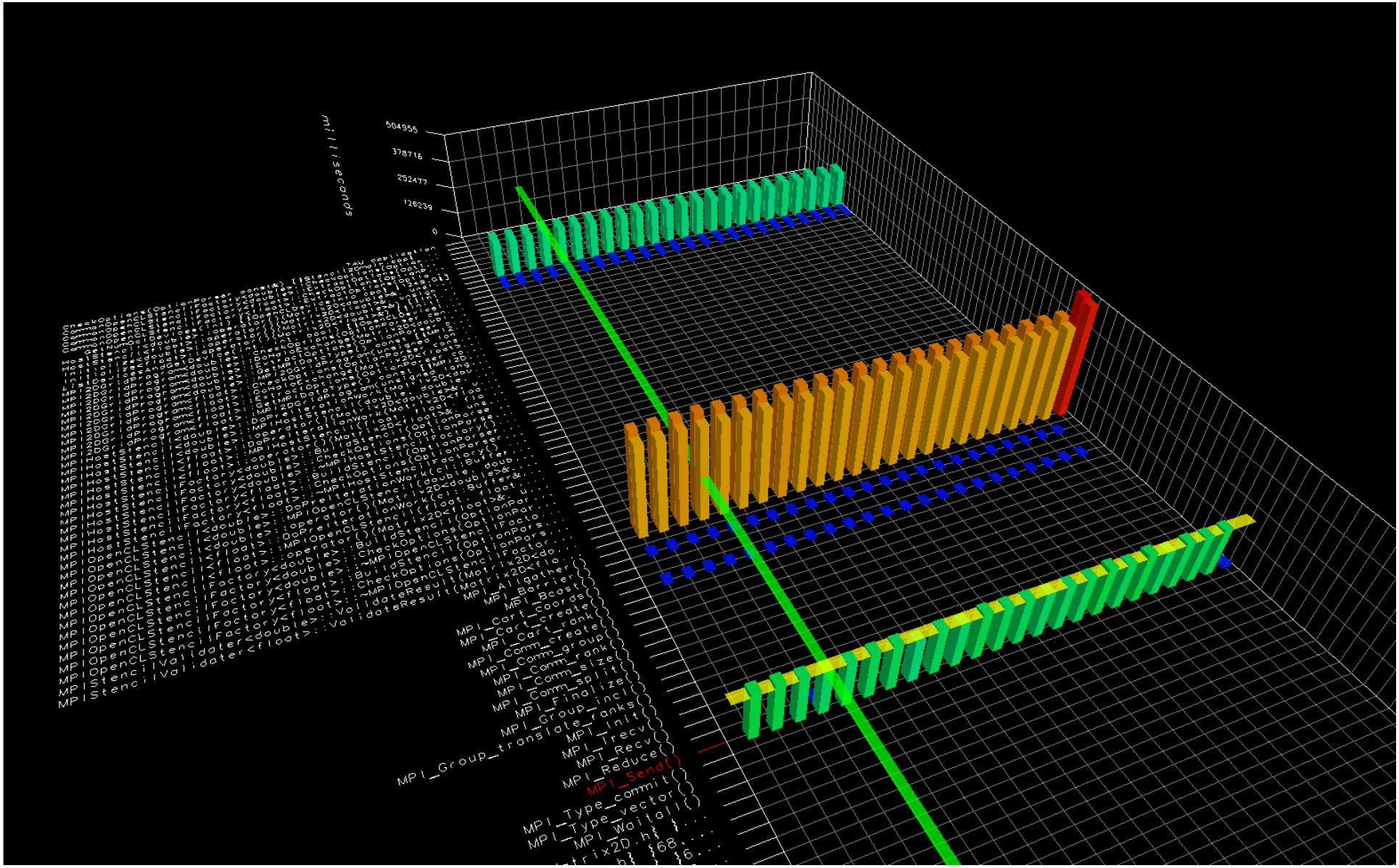
Stencil2D Parallel Profile / Trace

Metric: TAUGPU_TIME
Value: Exclusive



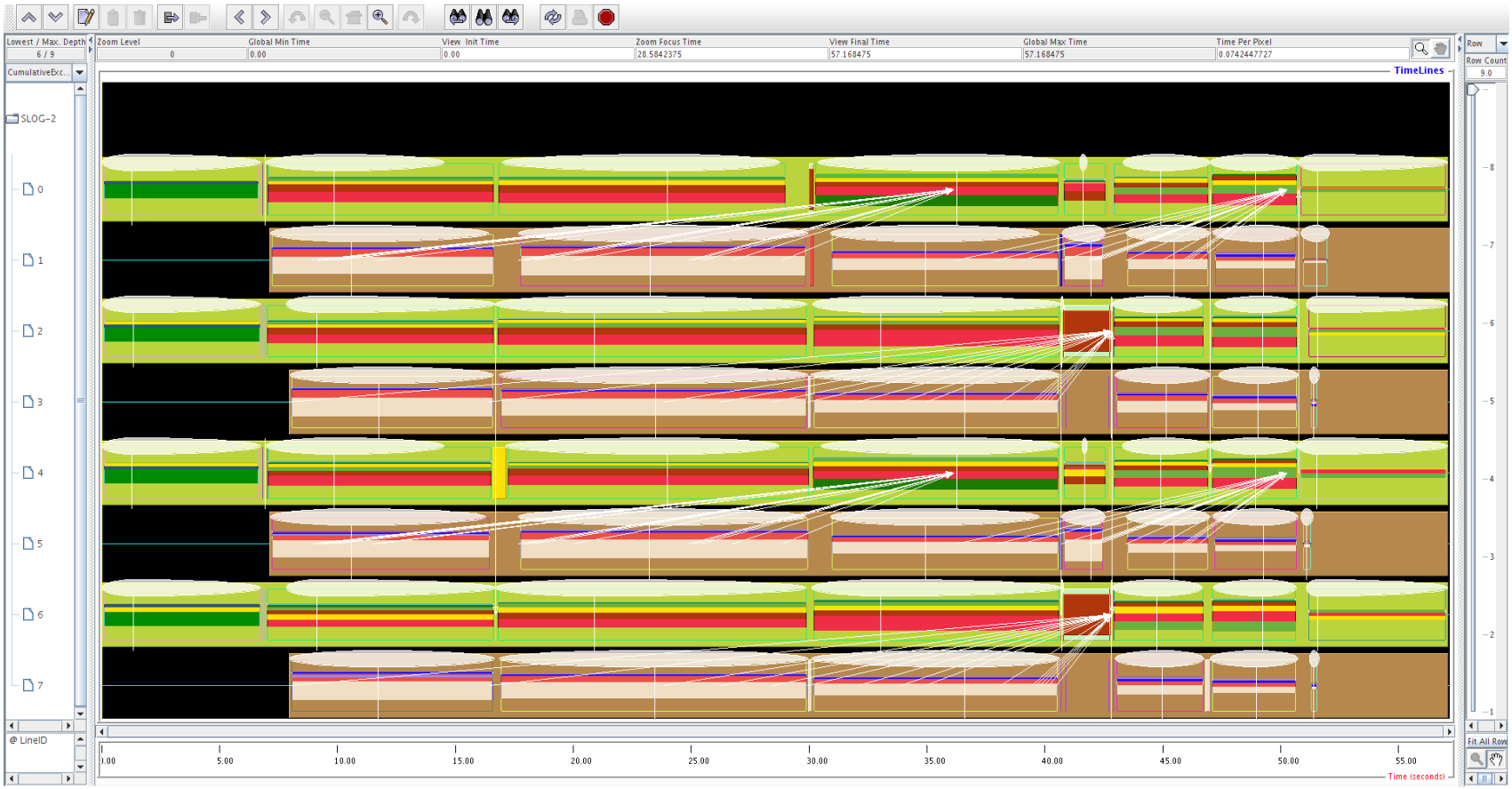


Stencil2D Parallel Profile

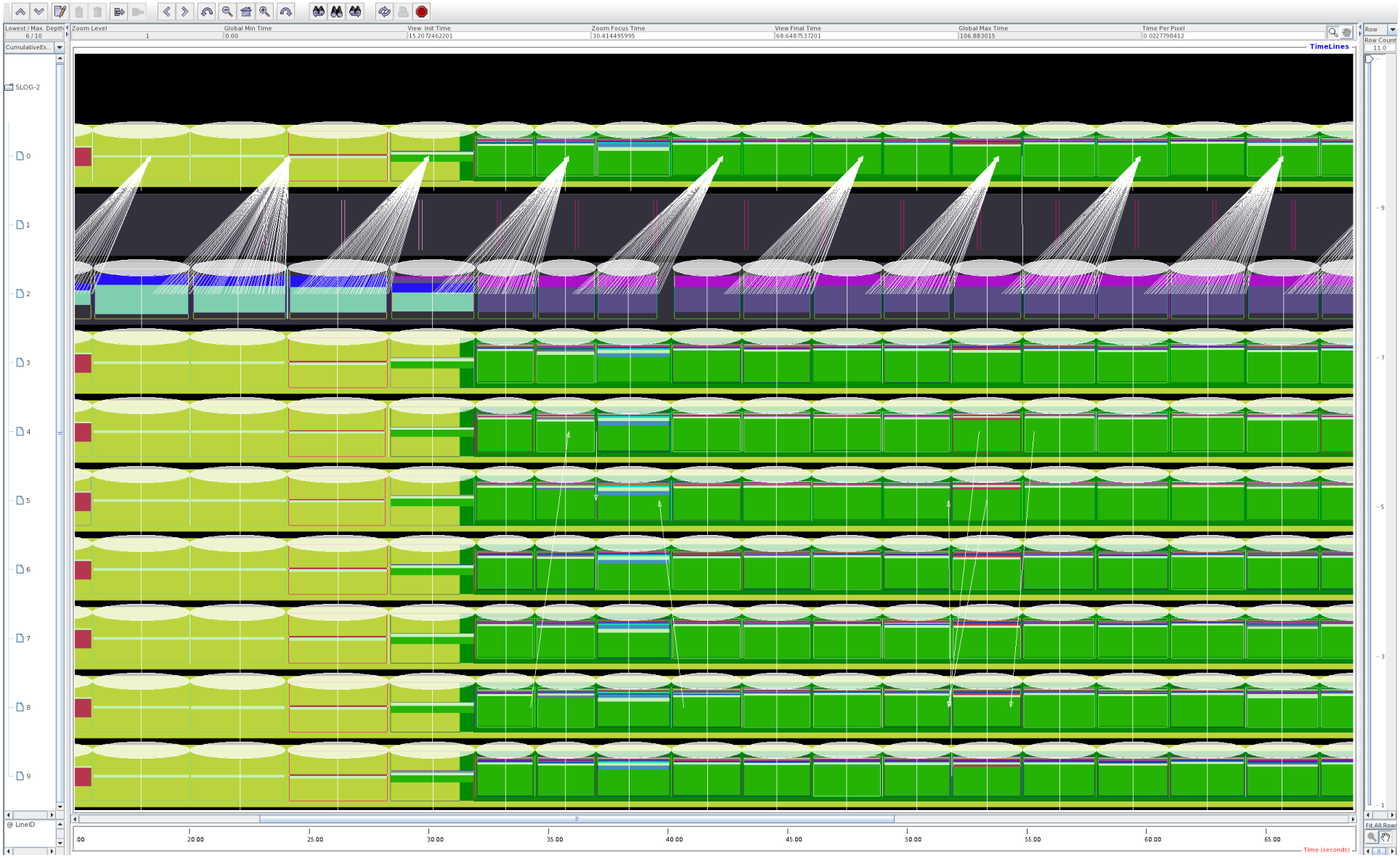


Example: CUDA Linpack

- TAU traces with Jumpshot visualization



Example: NAMD with CUPTI



Profiling PGI Accelerator Primitives

- PGI compiler allows users to annotate source code to identify loops that should be accelerated
- When a program is compiled with TAU, its measurement library intercepts the PGI runtime library layer to measure time spent in the runtime library routines and data transfers
- TAU also captures the arguments:
  - array data dimensions and sizes, strides, upload and download times, variable names, source file names, row and column information, and routines

Example: PGI GPU-accelerated MM

TAU: ParaProf: n,c,t 0,0,0 - /Users/sameer/rs/taudata/mm

Metric: TIME
Value: Exclusive percent

55.367%	__pgi_cu_downloadx multiply_matrices var=a, dims=2, desc.devx=0, desc.devstride=1, desc.hoststride=1, desc.size=3000, desc.extent=
39.329%	__pgi_cu_init multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}{9}]
1.822%	mymatrixmultiply [{mmdriv.f90} {1,0}]
1.648%	__pgi_cu_uploadx multiply_matrices var=c, dims=2, desc.devx=0, desc.devstride=1, desc.hoststride=1, desc.size=3000, elementsize=4 [
1.618%	__pgi_cu_uploadx multiply_matrices var=b, dims=2, desc.devx=0, desc.devstride=1, desc.hoststride=1, desc.size=3000, elementsize=4 |
0.083%	__pgi_cu_free multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}]
0.07%	__pgi_cu_alloc multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}{9}]
0.037%	multiply_matrices [{mm2.f90} {5,0}]
0.007%	__pgi_cu_module multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}{9}]
0.006%	__pgi_cu_launch multiply_matrices (multiply_matrices_11_gpu,gx=188,gy=188,gz=1,bx=16,by=16,bz=1,flag=0) [{/mnt/netapp/home1/sa
0.005%	__pgi_cu_paramset multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}]
0.004%	__pgi_cu_launch multiply_matrices (multiply_matrices_15_gpu,gx=188,gy=188,gz=1,bx=16,by=16,bz=1,flag=0) [{/mnt/netapp/home1/sa
0.002%	__pgi_cu_module_function2 multiply_matrices name=multiply_matrices_11_gpu, argname=(null), argsize=20, varname=(null), varsize=0 [{/r
0.002%	__pgi_cu_module_function2 multiply_matrices name=multiply_matrices_15_gpu, argname=(null), argsize=44, varname=(null), varsize=0 [{/r

TAU: ParaProf: Thread Statistics: n,c,t, 0,0,0 - /Users/sameer/rs/taudata/mm

Exclusive TIME %	Name	Exclusive TIME	Inclusive TIME	Calls	Child Calls
55.4%	__pgi_cu_downloadx multiply_matrices var=a, dims=2, desc.devx=0, desc.devstride=1, desc.hostst	2.721	2.721	5	0
39.3%	__pgi_cu_init multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}{9}]	1.933	1.933	5	0
1.8%	mymatrixmultiply [{mmdriv.f90} {1,0}]	0.09	4.914	1	5
1.6%	__pgi_cu_uploadx multiply_matrices var=c, dims=2, desc.devx=0, desc.devstride=1, desc.hoststrid	0.081	0.081	5	0
1.6%	__pgi_cu_uploadx multiply_matrices var=b, dims=2, desc.devx=0, desc.devstride=1, desc.hoststrid	0.079	0.079	5	0
0.1%	__pgi_cu_free multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}]	0.004	0.004	15	0
0.1%	__pgi_cu_alloc multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}{9}]	0.003	0.003	15	0
0.0%	multiply_matrices [{mm2.f90} {5,0}]	0.002	4.825	5	85
0.0%	__pgi_cu_module multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}{9}]	0	0	5	0
0.0%	__pgi_cu_launch multiply_matrices (multiply_matrices_11_gpu,gx=188,gy=188,gz=1,bx=16,by=16	0	0	5	0
0.0%	__pgi_cu_paramset multiply_matrices [{/mnt/netapp/home1/sameer/mm/mm2.f90}]	0	0	10	0
0.0%	__pgi_cu_launch multiply_matrices (multiply_matrices_15_gpu,gx=188,gy=188,gz=1,bx=16,by=16	0	0	5	0
0.0%	__pgi_cu_module_function2 multiply_matrices name=multiply_matrices_11_gpu, argname=(null), arg:	0	0	5	0

PGI MM Computational Kernel

TAU: ParaProf: Function Data Window: /Users/sameer/rs/taudata/pgiacc/mm/PGI11.2

Name: __pgi_cu_downloadx multiply_matrices var=a, dims=2, desc.devx=0, desc.devstride=1, desc.hoststride=1, desc.size=3000, desc.extent=3000, elementsize=4 [{/mnt/netapp/home1/sameer/mm/mm2.f90}{20}]
Metric Name: TIME
Value: Exclusive
Units: seconds

2.722 node 0

- Show Source Code
- Show Function Bar Chart
- Show Function Histogram
- Assign Function Color
- Reset to Default Color
- Rename

TAU: ParaProf: Source Browser: ./mm2.f90

```
1   ! Simple matmul example
2
3       module mymm
4       contains
5       subroutine multiply_matrices( a, b, c, m )
6        real, dimension(:,:) :: a,b,c
7        i = 0
8
9   !$acc region
10       do j = 1,m
11           do i = 1,m
12            a(i,j) = 0.0
13           enddo
14         do k = 1,m
15           do i = 1,m
16            a(i,j) = a(i,j) + b(i,k) * c(k,j)
17           enddo
18         enddo
19       enddo
20  !$acc end region
21      end subroutine
22      end module
23
```

Getting Started with TAU on KID

- Tutorial archive with example programs, performance data, and instructions
- Instructions
  - ssh Keeneland
  - cp /nics/c/home/biersdor/tau-tutorial-examples.tar.gz .
  - tar xzf tau-tutorial-examples.tar.gz
  - qlogin -I -l nodes=1:ppn=3 -l feature=nvupgrade
  - cd tau-tutorial-examples
  - source env.bash (or) source env.csh
  - cd cuda/binaries
  - tau_exec -T serial -cuda ./MD
  - pprof

Does it work?

- If successful, you will see something like:

```
Reading Profile files in profile.*NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0         50        5,924           1           2    5924844 .TAU application
99.1       5,393        5,874          59         296      99565 main
6.1          361          361           3           0     120463 cudaError_t cudaMallocHost(void **, size_t) C
1.1           64           64           2           0      32290 cudaError_t cudaGetDeviceCount(int *) C
0.7           42           42           1           0      42653 cudaError_t cudaThreadExit(void) C
0.1            3            3           3           0       1329 cudaError_t cudaFreeHost(void *) C
0.1            3            3          13           0        280 cudaError_t cudaEventSynchronize(cudaEvent_t) C
0.0            1            1           3           0        419 cudaError_t cudaMemcpy(void *, const void *, …
0.0        0.991        0.991          34           0         29 cudaError_t cudaEventElapsedTime(float *, …
0.0        0.353        0.353           1           0        353 cudaError_t cudaThreadSynchronize(void) C
0.0        0.336        0.336          22           0         15 cudaError_t cudaEventQuery(cudaEvent_t) C
0.0        0.205        0.331          11          80         30 cudaError_t cudaLaunch(const char *) C
0.0        0.308        0.308           3           0        103 cudaError_t cudaFree(void *) C
0.0        0.119        0.196           9          14         22 EnumerateDevicesAndChoose(int, bool)
```

- For more detailed profiles and traces copy the results directory to your local machine and run paraprof on the .ppk files and jumpshot on the .slog2 files

Downloading TAU to Desk/Laptop

- Window (http://tau.uoregon.edu/tau.exe)
  - Executable is self-extracting
  - Launch ParaProf and Jumpshot from
  - C:\Program Files\Tau 2.20.1 directory
- Mac (http://tau.uoregon.edu/tau.dmg)
  - Mount DMG and drag to copy
  - TAU to /Applications directory
  - Launch ParaProf and Jumpshot from /Application/TAU directory
- Linux (http://tau.uoregon.edu/tau.tgz)
  - Untar and run ./configure from tau-2.20.1 directory
  - Launch ParaProf and Jumpshot from tau-2.20.1/<arch>/bin directory (<arch> likely x86_64 or i386)

Support Acknowledgements

- Department of Energy (DOE)
  - Office of Science
    - MICS, Argonne National Lab
  - ASC/NNSA
    - University of Utah ASC/NNSA Level 1
    - ASC/NNSA, Lawrence Livermore National Lab
- Department of Defense (DoD)
  - HPC Modernization Office (HPCMO)
- NSF Software Development for Cyberinfrastructure (SDCI)
- Research Centre Juelich
- Los Alamos National Laboratory
- TU Dresden
- ParaTools, Inc.