Ménage à Trois: Hybrid Profiling, Performance Visualization, and Kernel Measurement

Allen D. Malony, Chee Wai Lee, Wyatt Spear, Will Voorhees
Sameer Shende, Scott Biersdorff, Suzanne Millstein

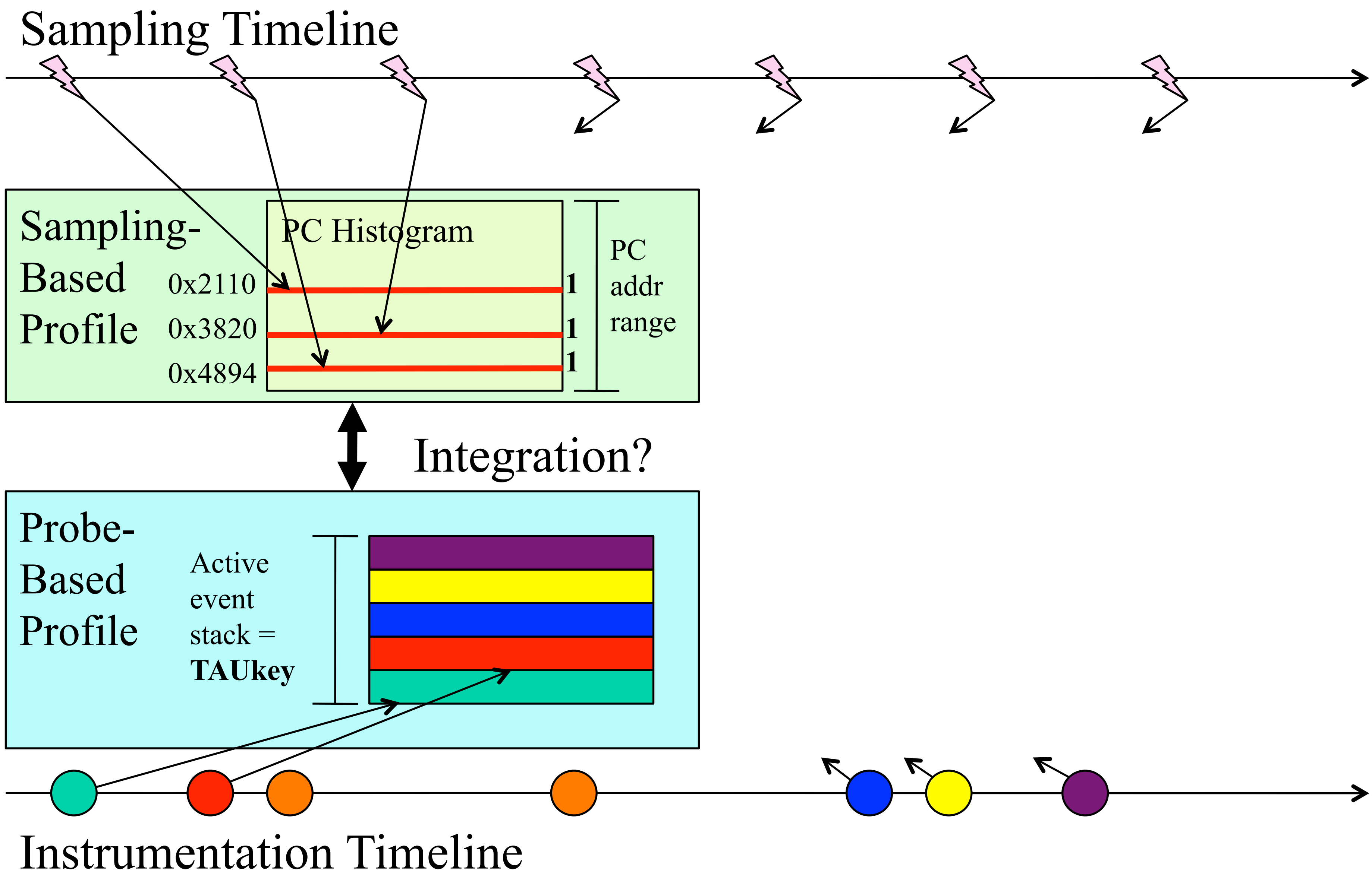
Dept. Computer and Information Science
Performance Research Laboratory
University of Oregon

UN

Hybrid Profiling – Motivation

- Different approaches for observing parallel performance
- ***Sampling-based measurement***
  - Event-based / instruction-based sampling (EBS / IBS)
  - Examples: PerfSuite, HPCToolkit, ...
- ***Probe-based measurement*** (PBM)
  - Instrumentation of program code
  - Example: TAU, Scalasca, ...
- Combine the two to exploits advantages of probe-based instrumentation with advantages of sampling
- TAUebs
  - TAU for probe-based instrumentation and measurement
  - Event-based sampling measurement (with callstack unwinding)

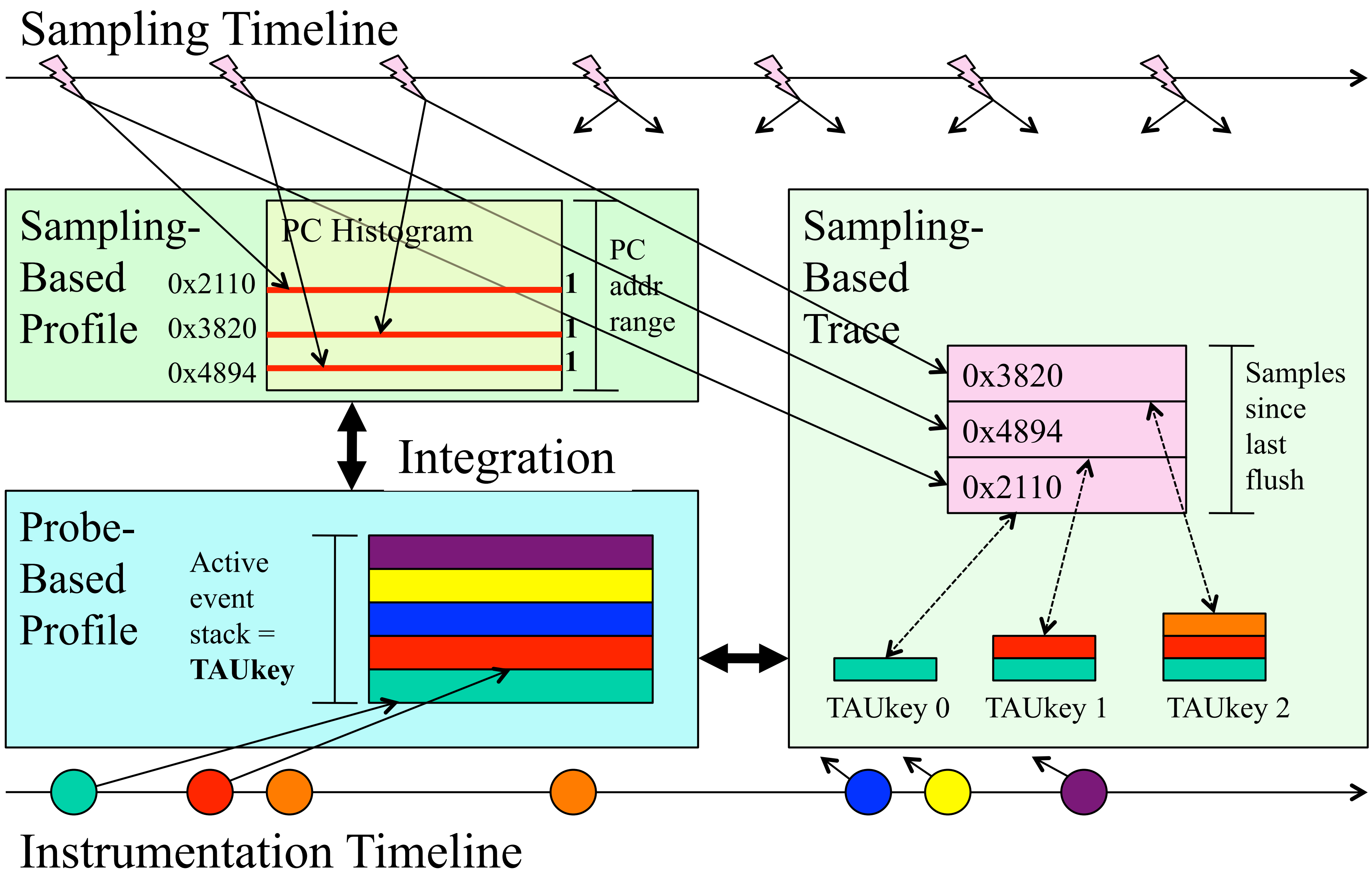
Integrated Probe + EBS Measurement Design (1)



Hmm, seems like we have seen this before …

- Previously, TAUebs:
  - Captured a trace of EBS samples
  - Post-processed the trace to recover symbol information
  - Merged sample traces with generated profiles offline
  - Paper in ICPP 2010 and discussed briefly at CScADS

Integrated Probe + EBS Measurement Design (2)

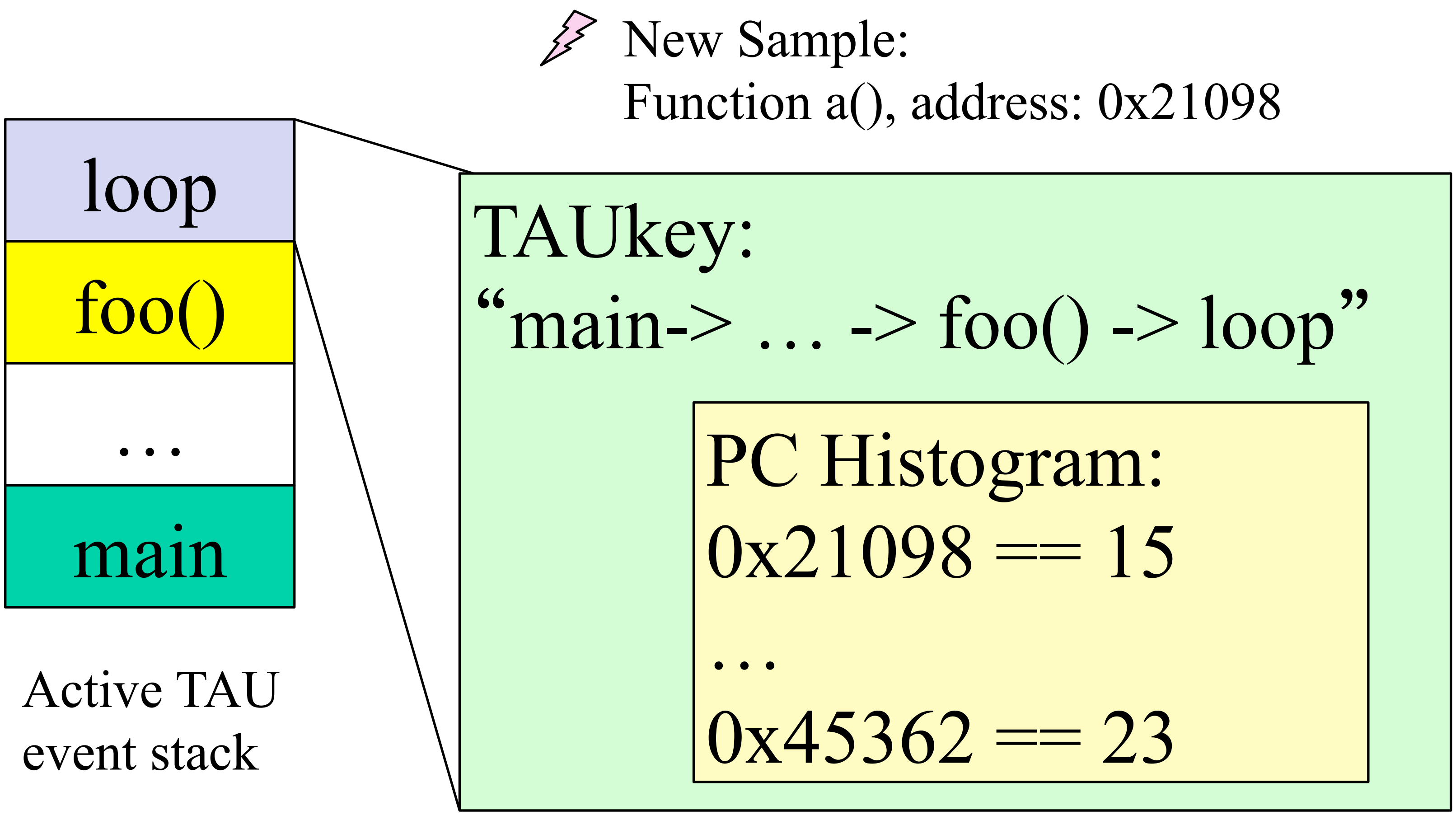


So what's new?

- Previously, TAUebs:
  - Captured a trace of EBS samples
  - Post-processed the trace to recover symbol information
  - Merged sample traces with generated profiles offline
  - Paper in ICPP 2010 and discussed briefly at CScADS
- Now, TAUebs:
  - Captures EBS sample histograms at runtime (profiling)
  - Sample histograms are associated with TAU event context
  - TAU profile output now incorporates sampled histograms
  - It is still possible to generate EBS traces

TAUebs Hybrid Profiling

- Instance of new sample contextualized by TAUkey and integrated into TAU profile structures at runtime

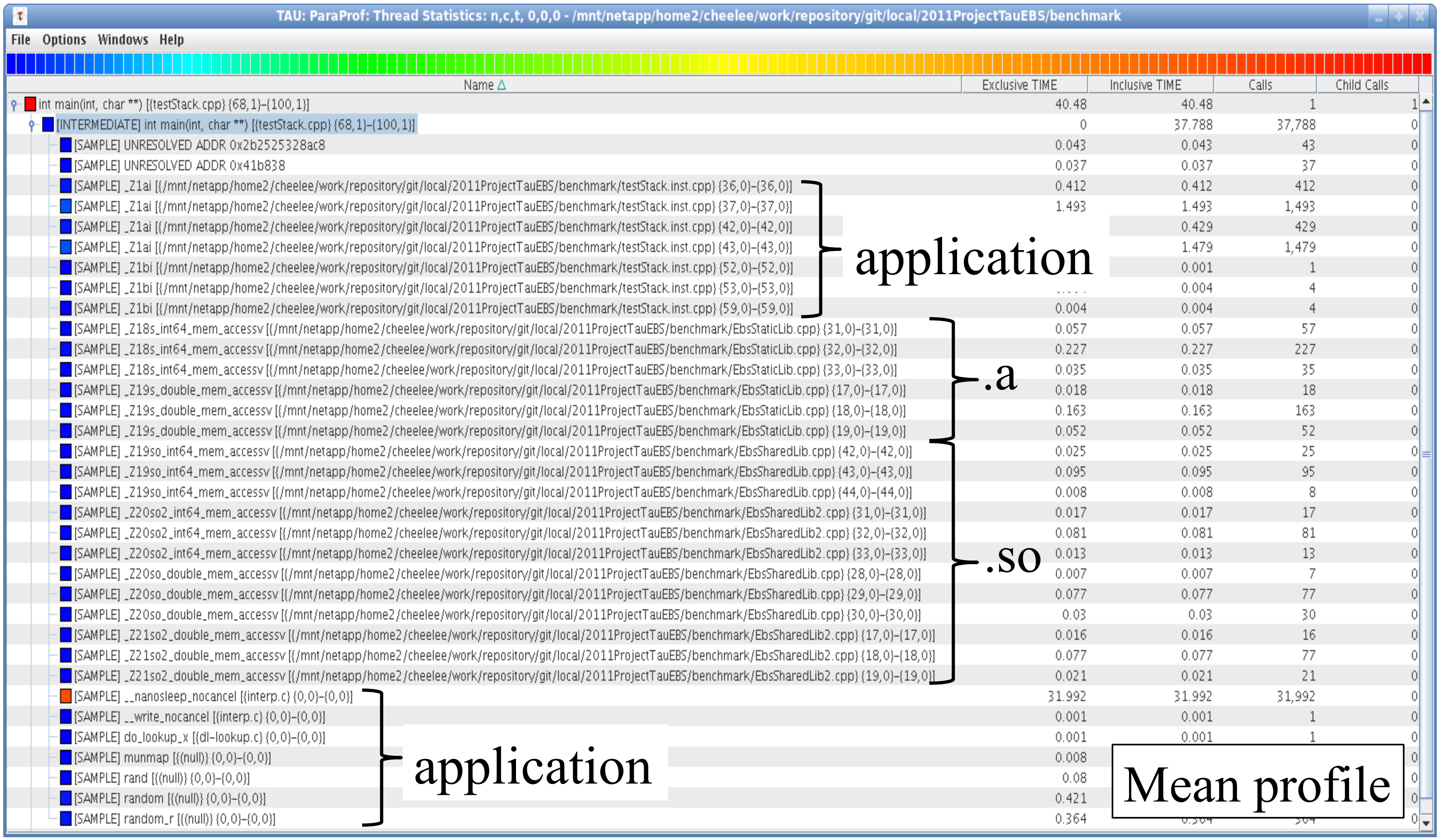


Hybrid Profiling Implementation

- Uses existing timer-interrupt framework to trigger samples
- With each sample:
  - Query active TAU event context to determine TAUkey
  - Create/update PC address histogram for the active TAU event context represented by its key
- Addresses are resolved to meaningful symbol information via BFD at the end of the run
- TAU event context can be controlled by the *event path depth*
- Caveats
  - Current implementation does not unwind the callstack
    - flat sample profile for each TAU event context
  - Works only with single-threaded processes presently

Examples: Simple Benchmark – Flat Profile

- Pure sampling – only main() is instrumented



Name △	Exclusive TIME	Inclusive TIME	Calls	Child Calls
int main(int, char **) [{testStack.cpp} {68,1}-{100,1}]	40.48	40.48	1	1
[INTERMEDIATE] int main(int, char **) [{testStack.cpp} {68,1}-{100,1}]	0	37.788	37,788	0
[SAMPLE] UNRESOLVED ADDR 0x2b2525328ac8	0.043	0.043	43	0
[SAMPLE] UNRESOLVED ADDR 0x41b838	0.037	0.037	37	0
[SAMPLE] _Z1ai [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/testStack.inst.cpp} {36,0}-{36,0}]	0.412	0.412	412	0
[SAMPLE] _Z1ai [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/testStack.inst.cpp} {37,0}-{37,0}]	1.493	1.493	1,493	0
[SAMPLE] _Z1ai [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/testStack.inst.cpp} {42,0}-{42,0}]	[illegible]	0.429	429	0
[SAMPLE] _Z1ai [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/testStack.inst.cpp} {43,0}-{43,0}]	[illegible]	1.479	1,479	0
[SAMPLE] _Z1bi [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/testStack.inst.cpp} {52,0}-{52,0}]	[illegible]	0.001	1	0
[SAMPLE] _Z1bi [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/testStack.inst.cpp} {53,0}-{53,0}]	[illegible]	0.004	4	0
[SAMPLE] _Z1bi [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/testStack.inst.cpp} {59,0}-{59,0}]	0.004	0.004	4	0
[SAMPLE] _Z18s_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsStaticLib.cpp} {31,0}-{31,0}]	0.057	0.057	57	0
[SAMPLE] _Z18s_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsStaticLib.cpp} {32,0}-{32,0}]	0.227	0.227	227	0
[SAMPLE] _Z18s_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsStaticLib.cpp} {33,0}-{33,0}]	0.035	0.035	35	0
[SAMPLE] _Z19s_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsStaticLib.cpp} {17,0}-{17,0}]	0.018	0.018	18	0
[SAMPLE] _Z19s_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsStaticLib.cpp} {18,0}-{18,0}]	0.163	0.163	163	0
[SAMPLE] _Z19s_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsStaticLib.cpp} {19,0}-{19,0}]	0.052	0.052	52	0
[SAMPLE] _Z19so_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib.cpp} {42,0}-{42,0}]	0.025	0.025	25	0
[SAMPLE] _Z19so_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib.cpp} {43,0}-{43,0}]	0.095	0.095	95	0
[SAMPLE] _Z19so_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib.cpp} {44,0}-{44,0}]	0.008	0.008	8	0
[SAMPLE] _Z20so2_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib2.cpp} {31,0}-{31,0}]	0.017	0.017	17	0
[SAMPLE] _Z20so2_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib2.cpp} {32,0}-{32,0}]	0.081	0.081	81	0
[SAMPLE] _Z20so2_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib2.cpp} {33,0}-{33,0}]	0.013	0.013	13	0
[SAMPLE] _Z20so_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib.cpp} {28,0}-{28,0}]	0.007	0.007	7	0
[SAMPLE] _Z20so_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib.cpp} {29,0}-{29,0}]	0.077	0.077	77	0
[SAMPLE] _Z20so_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib.cpp} {30,0}-{30,0}]	0.03	0.03	30	0
[SAMPLE] _Z21so2_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib2.cpp} {17,0}-{17,0}]	0.016	0.016	16	0
[SAMPLE] _Z21so2_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib2.cpp} {18,0}-{18,0}]	0.077	0.077	77	0
[SAMPLE] _Z21so2_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/EbsSharedLib2.cpp} {19,0}-{19,0}]	0.021	0.021	21	0
[SAMPLE] __nanosleep_nocancel [{interp.c} {0,0}-{0,0}]	31.992	31.992	31,992	0
[SAMPLE] __write_nocancel [{interp.c} {0,0}-{0,0}]	0.001	0.001	1	0
[SAMPLE] do_lookup_x [{dl-lookup.c} {0,0}-{0,0}]	0.001	0.001	1	0
[SAMPLE] munmap [{(null)} {0,0}-{0,0}]	0.008	[illegible]	[illegible]	0
[SAMPLE] rand [{(null)} {0,0}-{0,0}]	0.08	[illegible]	[illegible]	0
[SAMPLE] random [{(null)} {0,0}-{0,0}]	0.421	[illegible]	[illegible]	0
[SAMPLE] random_r [{(null)} {0,0}-{0,0}]	0.364	[illegible]	[illegible]	0

Examples: Simple Benchmark – Hybrid Profile

- Mix of sampling and probed-based instrumentation

TAU: ParaProf: Thread Statistics: n,c,t, 0,0,0 - /mnt/netapp/home2/cheelee/work/repository/git/local/2011ProjectTauEBS/benchmark/Mix

File Options Windows Help

Name △	Exclusive TIME	Inclusive TIME	Calls	Child Calls
int main(int, char **) [{testStack.cpp} {68,1}-{100,1}]	27.546	34.078	1	4
[INTERMEDIATE] int main(int, char **) [{testStack.cpp} {68,1}-{100,1}]	0	27.498	27,498	0
[SAMPLE] UNRESOLVED ADDR 0x2b32c567aac8	0.049	0.049	49	0
[SAMPLE] UNRESOLVED ADDR 0x41b880	0.02	0.02	20	0
[SAMPLE] _Z18s_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repositc	0.026	0.026	26	0
[SAMPLE] _Z18s_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repositc	0.082	0.082	82	0
[SAMPLE] _Z18s_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repositc	0.012	0.012	12	0
[SAMPLE] _Z19s_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/reposi	0.012	0.012	12	0
[SAMPLE] _Z19s_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/reposi	0.081	0.081	81	0
[SAMPLE] _Z19s_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/reposi	0.034	0.034	34	0
[SAMPLE] _Z19so_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/reposit	0.019	0.019	19	0
[SAMPLE] _Z19so_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/reposit	0.073	0.073	73	0
[SAMPLE] _Z19so_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/reposit	0.014	0.014	14	0
[SAMPLE] _Z20so2_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repos	[illegible]	0.016	16	0
[SAMPLE] _Z20so2_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repos	[illegible]	0.088	88	0
[SAMPLE] _Z20so2_int64_mem_accessv [{/mnt/netapp/home2/cheelee/work/repos	[illegible]	0.019	19	0
[SAMPLE] _Z20so_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repo	0.01	0.01	10	0
[SAMPLE] _Z20so_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repo	0.082	0.082	82	0
[SAMPLE] _Z20so_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/repo	0.03	0.03	30	0
[SAMPLE] _Z21so2_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/rep	0.014	0.014	14	0
[SAMPLE] _Z21so2_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/rep	0.085	0.085	85	0
[SAMPLE] _Z21so2_double_mem_accessv [{/mnt/netapp/home2/cheelee/work/rep	0.029	0.029	29	0
[SAMPLE] __nanosleep_nocancel [{interp.c} {0,0}-{0,0}]	26.086	26.086	26,086	0
[SAMPLE] munmap [{(null)} {0,0}-{0,0}]	0.006	0.006	6	0
[SAMPLE] puts [{(null)} {0,0}-{0,0}]	0.001	0.001	1	0
[SAMPLE] rand [{(null)} {0,0}-{0,0}]	0.05	0.05	50	0
[SAMPLE] random [{(null)} {0,0}-{0,0}]	0.318	0.318	318	0
[SAMPLE] random_r [{(null)} {0,0}-{0,0}]	0.242	0.242	242	0
double a(int) [{testStack.cpp} {30,1}-{44,1}]	0.009	3.022	1	1
[INTERMEDIATE] double a(int) [{testStack.cpp} {30,1}-{44,1}]	[illegible]	0.007	7	0
[SAMPLE] _Z1ai [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011I	[illegible]	0.003	3	0
[SAMPLE] _Z1ai [{/mnt/netapp/home2/cheelee/work/repository/git/local/2011I	0.004	0.004	4	0
void main_foo() [{testStack.cpp} {23,1}-{28,1}]	[illegible]	[illegible]	1	1
[INTERMEDIATE] void main_foo() [{testStack.cpp} {23,1}-{28,1}]	[illegible]	[illegible]	2,009	0
[SAMPLE] __nanosleep_nocancel [{interp.c} {0,0}-{0,0}]	[illegible]	[illegible]	2,009	0
void main_bar() [{testStack.cpp} {18,1}-{21,1}]	1.004	1.004	1	0
[INTERMEDIATE] void main_bar() [{testStack.cpp} {18,1}-{21,1}]	[illegible]	[illegible]	[illegible]	0
[SAMPLE] __nanosleep_nocancel [{interp.c} {0,0}-{0,0}]	[illegible]	[illegible]	[illegible]	0
double b(int) [{testStack.cpp} {46,1}-{60,1}]	0.009	3.021	1	1
int main_divByZero() [{testStack.cpp} {10,1}-{16,1}]	0	0	1	0
void main_inlined_computeNest() [{testStack.cpp} {62,1}-{66,1}]	0.488	0.488	1	0

C: main

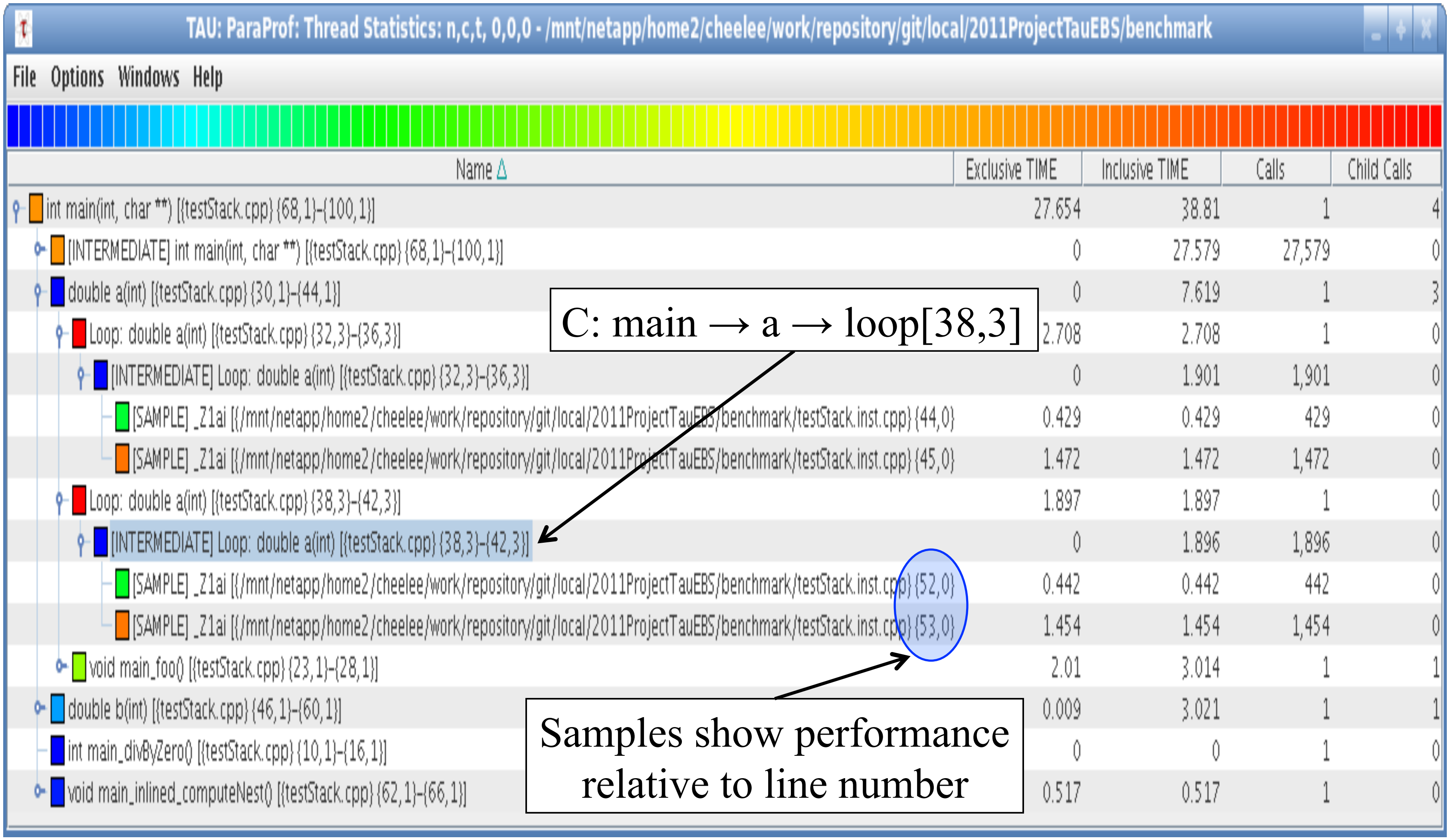
C: main → a

C: main → a → main_foo

C: main → a → main_foo → main_bar

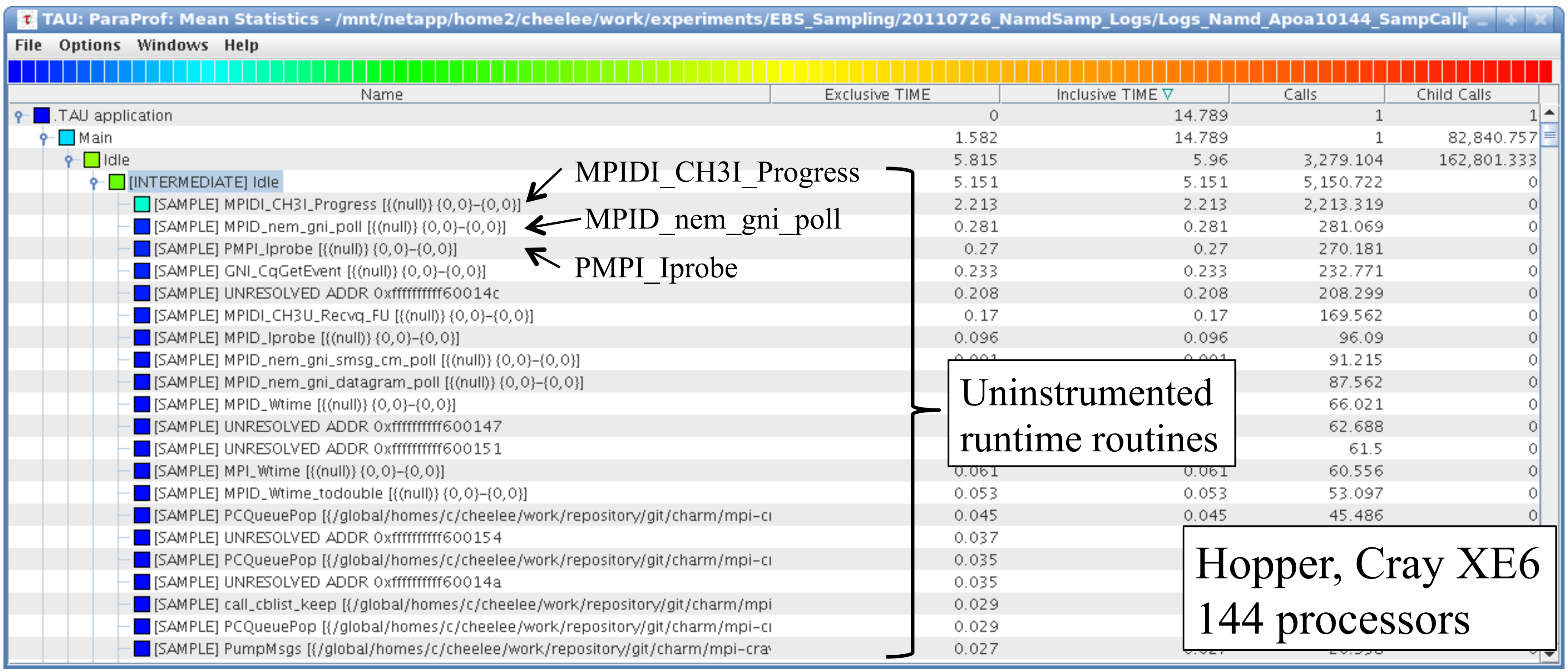
Examples: Simple Benchmark – Loops

- TAU events not restricted to routines (e.g., blocks, loops, …)



Examples: NAMD – Runtime System Contexts

- ❒ NAMD is implemented with Charm++ programming model
- ❒ Charm++ exposes its programming and runtime constructs
  - ❍ Callback system for TAU for probed-based measurement
- ❒ Hybrid profiling reveals action within Charm++ “idle” state



Examples: FLASH4 – Event Paths + Samples

TAU: ParaProf: Mean Statistics - /home/cheelee/Logs_Flash4_Cellular0144_SampCallpath50

File Options Windows Help

Name △	Exclusive TIME	Inclusive TIME	Calls	Child Calls
FLASH [{Flash.F90} {40,1}-{56,17}]	0	211.551	1	4
DRIVER_EVOLVEFLASH [{Driver_evolveFlash.F90} {40,1}-{481,33}]	0.066	209.722	1	50,027.889
COSMOLOGY_GETREDSHIFT [{Cosmology_getRedshift.F90} {28,1}-{36,36}]	0.002	0.002	2,000	0
COSMOLOGY_REDSHIFTHYDRO [{Cosmology_redshiftHydro.F90} {26,1}-{33,38}]	0.003	0.003	4,000	0
COSMOLOGY_SOLVEFRIEDMANNEQN [{Cosmology_solveFriedmannEqn.F90} {27,1}-{35,42}]	0.003	0.003	4,000	0
DIFFUSE [{Diffuse.F90} {57,1}-{66,22}]	0.003	0.003	4,000	0
DRIVER_COMPUTEDT [{Driver_computeDt.F90} {57,1}-{561,31}]	0.063	7.288	2,000	137,056.736
DRIVER_GETELAPSEDWCTIME [{Driver_getElapsedWCTime.F90} {32,1}-{46,38}]	0.002	0.002	2,000	0
DRIVER_SOURCETERMS [{Driver_sourceTerms.F90} {31,1}-{60,33}]	0.023	56.479	4,000	32,000
DR_SHORTENLASTDT [{dr_shortenLastDt.F90} {32,1}-{58,31}]	0.002	0.002	2,000	0
GRAVITY_POTENTIALLISTOFBLOCKS [{Gravity_potentialListOfBlocks.F90} {54,1}-{64,44}]	0.005	0.005	4,000	0
GRID_GETLISTOFBLOCKS [{Grid_getListOfBlocks.F90} {73,1}-{196,35}]	0.002	0.002	2,000	0
GRID_GETLOCALNUMBLKS [{Grid_getLocalNumBlks.F90} {19,1}-{30,35}]	0.003	0.003	2,000	0
GRID_UPDATEREFINEMENT [{Grid_updateRefinement.F90} {42,1}-{106,36}]	0.012	26.146	2,000	5,002
HYDRO [{Hydro.F90} {47,1}-{208,20}]	0.014	119.684	4,000	12,000
GRID_MOVECUSTOMREGION [{Grid_moveCustomRegion.F90} {26,1}-{182,36}]	0.008	0.008	4,000	0
HY_PPM_SWEEP [{hy_ppm_sweep.F90} {66,1}-{661,27}]	1.573	119.662	8,000	644,084.333
DIFFUSE_THERM [{Diffuse_therm.F90} {106,1}-{147,28}]	0.055	0.055	52,339.778	0
DIFFUSE_VISC [{Diffuse_visc.F90} {94,1}-{135,27}]	0.032	0.032	52,339.778	0
EOS_WRAPPED [{Eos_wrapped.F90} {93,1}-{210,26}]	1.274	31.851	157,019.333	1,046,795.5...
EOS [{Eos.F90} {209,1}-{304,18}]	1.3	30.577	1,046,795.5...	1,046,795.5...
EOS_HELMHOLTZ [{eos_helmholtz.F90} {228,1}-{689,28}]	28.42	29.277	1,046,795.5...	219,766.472
Loop: EOS_HELMHOLTZ [{eos_helmholtz.F90} {314,3}-{334,7}]	0.398	0.398	73,154.306	0
[INTERMEDIATE] Loop: EOS_HELMHOLTZ [{eos_helmholtz.F90} {314,3}-{334,7}]	0	0.354	353.896	0
[SAMPLE] RtsLayer::myThread() [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/RtsThread.cpp} {100,0}-{100,0}]	0	0	0.042	0
[SAMPLE] Tau_get_tid [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/TauCAPI.cpp} {1506,0}-{1506,0}]	0	0	0.014	0
[SAMPLE] Tau_start_timer [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/TauCAPI.cpp} {330,0}-{330,0}]	0	0	0.083	0
[SAMPLE] Tau_stop_timer [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/TauCAPI.cpp} {343,0}-{343,0}]	0	0	0.097	0
[SAMPLE] Tau_stop_timer [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/TauCAPI.cpp} {344,0}-{344,0}]	0	0	0.042	0
[SAMPLE] __multispecies_data_MOD_ms_iszero [{(null)} {0,0}-{0,0}]	0.03	0.03	29.694	0
[SAMPLE] eos_helmholtz_ [{(null)} {0,0}-{0,0}]	0.015	0.015	15.292	0
[SAMPLE] ms_mapmsindex_ [{(null)} {0,0}-{0,0}]	0.021	0.021	20.514	0
[SAMPLE] multispecies_getproperty_ [{(null)} {0,0}-{0,0}]	0.099	0.099	99.292	0
[SAMPLE] multispecies_getsumfrac_ [{(null)} {0,0}-{0,0}]	0.117	0.117	116.694	0
[SAMPLE] multispecies_getsuminv_ [{(null)} {0,0}-{0,0}]	0.072	0.072	71.917	0
[SAMPLE] tau_profile_stop_ [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/TauFAPI.cpp} {618,0}-{618,0}]	0	0	0.111	0
[SAMPLE] tau_profile_stop_ [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/TauFAPI.cpp} {619,0}-{619,0}]	0	0	0.063	0
[SAMPLE] tau_profile_stop_ [{/global/homes/c/cheelee/work/repository/git/tau2/src/Profile/TauFAPI.cpp} {621,0}-{621,0}]	[illegible]	[illegible]	[illegible]	0
Loop: EOS_HELMHOLTZ [{eos_helmholtz.F90} {381,6}-{407,10}]	[illegible]	[illegible]	[illegible]	0
Loop: EOS_HELMHOLTZ [{eos_helmholtz.F90} {410,6}-{461,11}]	[illegible]	[illegible]	[illegible]	0
[INTERMEDIATE] EOS_HELMHOLTZ [{eos_helmholtz.F90} {228,1}-{689,28}]	[illegible]	[illegible]	[illegible]	0
[INTERMEDIATE] EOS [{Eos.F90} {209,1}-{304,18}]	[illegible]	[illegible]	[illegible]	0
[INTERMEDIATE] EOS_WRAPPED [{Eos_wrapped.F90} {93,1}-{210,26}]	0	0.895	895.201	0

Follow probed event path to end and pick up samples to highlight significant code points

Hopper, Cray XE6
144 processors

Example: FLASH4 – Reverse Event/Call Path

- Aggregate events / samples performance data

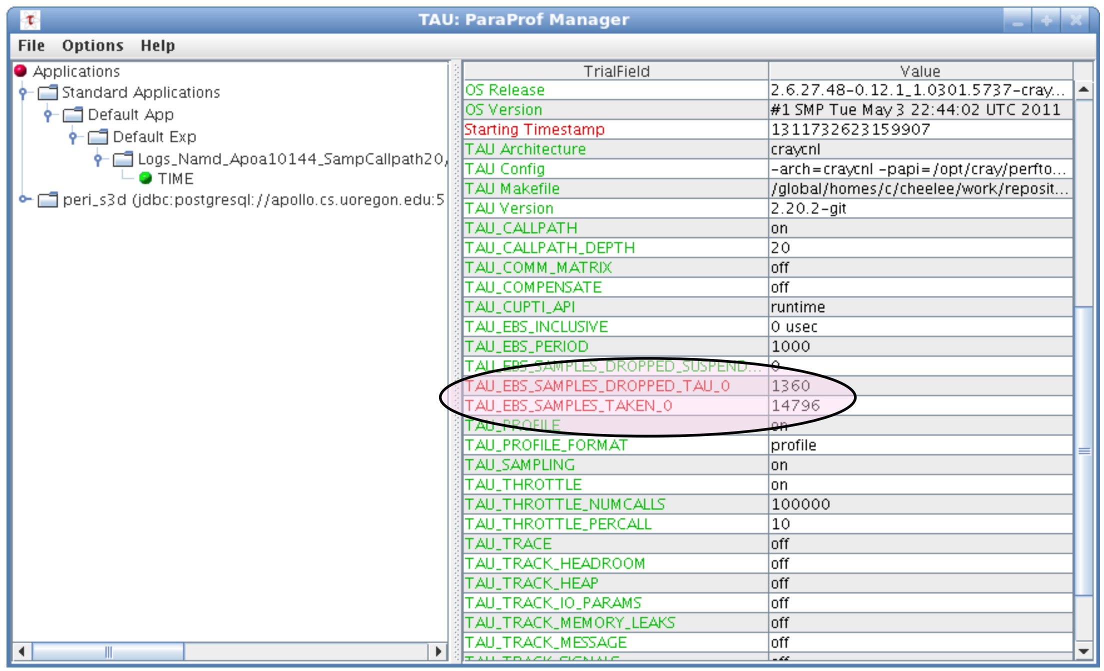
TAU: ParaProf: Mean Statistics – flash4-cellular-sampRev.hopper.0144.ppk

Name	Exclusive TIME ▽	Inclusive TIME	Calls	Child Calls
▶ HY_PPM_BLOCK [{hy_ppm_block.F90} {114,1}-{844,27}]	3.642	27.069	52,339.778	523,398.778
▶ AMR_1BLK_GUARDCELL_SRL [{amr_1blk_guardcell_srl.F90} {10,7}-{856,43}]	2.907	2.92	77,447.889	404.75
▶ [SAMPLE] amr_1blk_cc_cp_remote_ [{(null)} {0,0}-{0,0}]	2.737	2.737	2,736.556	0
▶ MPI_Waitall()	2.457	2.457	47,013.688	0
▶ [SAMPLE] interp_ [{(null)} {0,0}-{0,0}]	2.426	2.426	2,426.354	0
▼ [SAMPLE] memcpy [{/usr/src/packages/BUILD/glibc-2.9/string/../sysdeps/x86_64/memcpy.S} {267	2.408	2.408	2,408.014	0
▶ [INTERMEDIATE] MPI_Ssend()	1.727	1.727	1,726.59	0
▶ [INTERMEDIATE] MPI_AMR_READ_RESTRICT_COMM [{mpi_amr_store_comm_info.F90} {565,7}-{6	0.229	0.229	228.535	0
▶ [INTERMEDIATE] MPI_Waitall()	0.172	0.172	172.465	0
▶ [INTERMEDIATE] MPI_AMR_READ_FLUX_COMM [{mpi_amr_store_comm_info.F90} {399,7}-{477,4	0.13	0.13	130.285	0
▶ [INTERMEDIATE] MPI_AMR_READ_GUARD_COMM [{mpi_amr_store_comm_info.F90} {88,7}-{162,4	0.093	0.093	93.41	0
▶ [INTERMEDIATE] AMR_CHECK_DEREFINE [{mpi_amr_derefine_blocks.F90} {448,7}-{1176,39}]	0.023	0.023	22.562	0
▶ [INTERMEDIATE] AMR_SORT_MORTON [{mpi_amr_morton.F90} {481,7}-{631,36}]	[illegible]	[illegible]	9.174	0
▶ [INTERMEDIATE] MPI_Bcast()	[illegible]	[illegible]	4.462	0
▶ [INTERMEDIATE] MPI_AMR_WRITE_GUARD_COMM [{mpi_amr_store_comm_info.F90} {18,7}-{82,4	[illegible]	[illegible]	4.458	0
▶ [INTERMEDIATE] MPI_AMR_WRITE_RESTRICT_COMM [{mpi_amr_store_comm_info.F90} {485,7}-{5	[illegible]	[illegible]	3.923	0
▶ [INTERMEDIATE] MPI_AMR_WRITE_FLUX_COMM [{mpi_amr_store_comm_info.F90} {325,7}-{393,4	[illegible]	[illegible]	3.879	0
▶ [INTERMEDIATE] MPI_AMR_WRITE_PROL_COMM [{mpi_amr_store_comm_info.F90} {168,7}-{234,4	0.004	0.004	3.705	0
▶ [INTERMEDIATE] MPI_Allgather()	0.003	0.003	3.133	0
▶ [INTERMEDIATE] MPI_AMR_READ_PROL_COMM [{mpi_amr_store_comm_info.F90} {240,7}-{318,4	0.003	0.003	2.617	0
▶ [INTERMEDIATE] MPI_Barrier()	0.002	0.002	1.9	0
▶ [INTERMEDIATE] MPI_Irecv()	0.001	0.001	1	0
▶ [INTERMEDIATE] MPI_XCHANGE_BLOCKS [{mpi_lib.F90} {1211,7}-{1381,39}]	0.001	0.001	1	0
▶ [SAMPLE] multispecies_getsumfrac_ [{(null)} {0,0}-{0,0}]	2.389	2.389	2,389.194	0
▶ BN_NETINTEGRATE [{bn_netIntegrate.F90} {86,1}-{294,30}]	2.238	24.789	1,572,443.778	1,572,443.778
▶ [SAMPLE] rieman_ [{(null)} {0,0}-{0,0}]	2.193	2.193	2,193	0
▶ [SAMPLE] __pow [{/usr/src/packages/BUILD/glibc-2.9/math/../sysdeps/x86_64/fpu/w_pow.c} {765	2.168	2.168	2,167.75	0
▶ [SAMPLE] multispecies_getproperty_ [{(null)} {0,0}-{0,0}]	2.023	2.023	2,022.896	0
▶ EOS_WRAPPED [{Eos_wrapped.F90} {93,1}-{210,26}]	1.891	46.015	215,811.486	1,451,586.944
▶ [SAMPLE] states_ [{(null)} {0,0}-{0,0}]	1.822	1.822	1,822.16	0
▶ EOS [{Eos.F90} {209,1}-{304,18}]	1.795	44.124	1,451,798.5	1,451,798.5
▶ [SAMPLE] monot_ [{(null)} {0,0}-{0,0}]	1.73	1.73	1,730.194	0
▶ AMR_GUARDCELL [{mpi_amr_guardcell.F90} {88,7}-{485,34}]	1.713	23.099	9,006	127,343.806
▶ [SAMPLE] amr_guardcell_ [{(null)} {0,0}-{0,0}]	1.645	1.645	1,644.826	0
▶ HY_PPM_SWEEP [{hy_ppm_sweep.F90} {66,1}-{661,27}]	1.578	119.099	8,000	644,084.333
▶ [SAMPLE] hydro_1d_ [{(null)} {0,0}-{0,0}]	1.527	1.527	1,527.403	0
▶ [SAMPLE] eigensystem.1622.clone.1 [{interp_char.inst.F90} {0,0}-{0,0}]	1.5	1.5	1,499.903	0
▶ [SAMPLE] multispecies_getsuminv_ [{(null)} {0,0}-{0,0}]	1.473	1.473	1,473.479	0
▶ BURN [{Burn.F90} {51,1}-{256,19}]	1.396	55.819	4,000	1,597,228.667
▶ MPI_PACK_BLOCKS [{mpi_pack_blocks.F90} {15,7}-{281,36}]	1.273	1.273	22,272	0

TAU events within which this memcpy sample occurs

Issues – Signal Safety and Dropped Samples

- Signal safety
  - Strictly, signal handlers cannot attempt to acquire lock
  - TAUebs drops a sample if inside TAU operations for safety
  - EBS sample handler must avoid memory allocation
    - currently, use own C++ allocator and minimize calls to malloc
- Signal safety handling will drop samples
  - Keep track of # dropped samples in metadata
- Rules can be loosened to drop sample in a more intelligent way (only if unsafe)



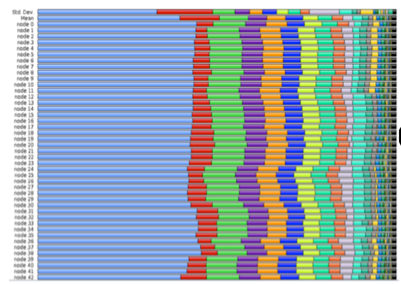
TAUebs Future Work

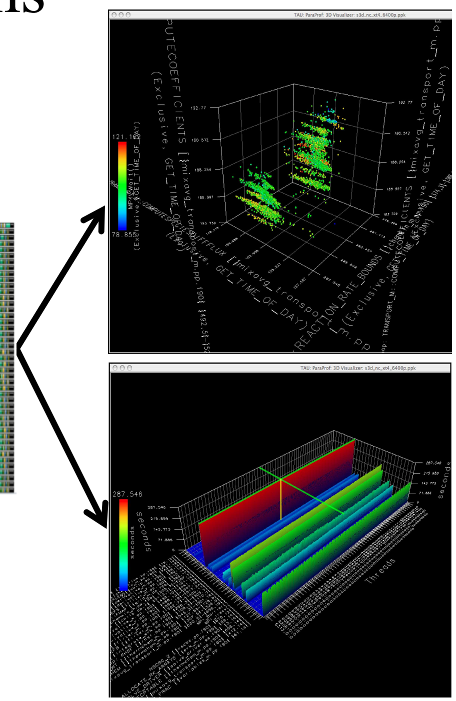
- Add callstack unwinding when sampling
  - Incorporate robust module for this
  - Control of unwinding depth with event/routine matching
  - Selective unwinding determined by event context
  - Mix of flat and structured sample blocks per context
- Rational interpretation of code structures for samples relative to non-function TAU contexts (e.g., loops)
- Improvements in data management
- Handling symbol resolution for sampled addresses at epoch transitions due to dynamic library loading and unloading
- More accurate sample timing
  - Measuring time elapsed between a sample and last probed event
- Sampling with other metrics and state (e.g., GPU counters)

DEUX

Performance Visualization – Motivation

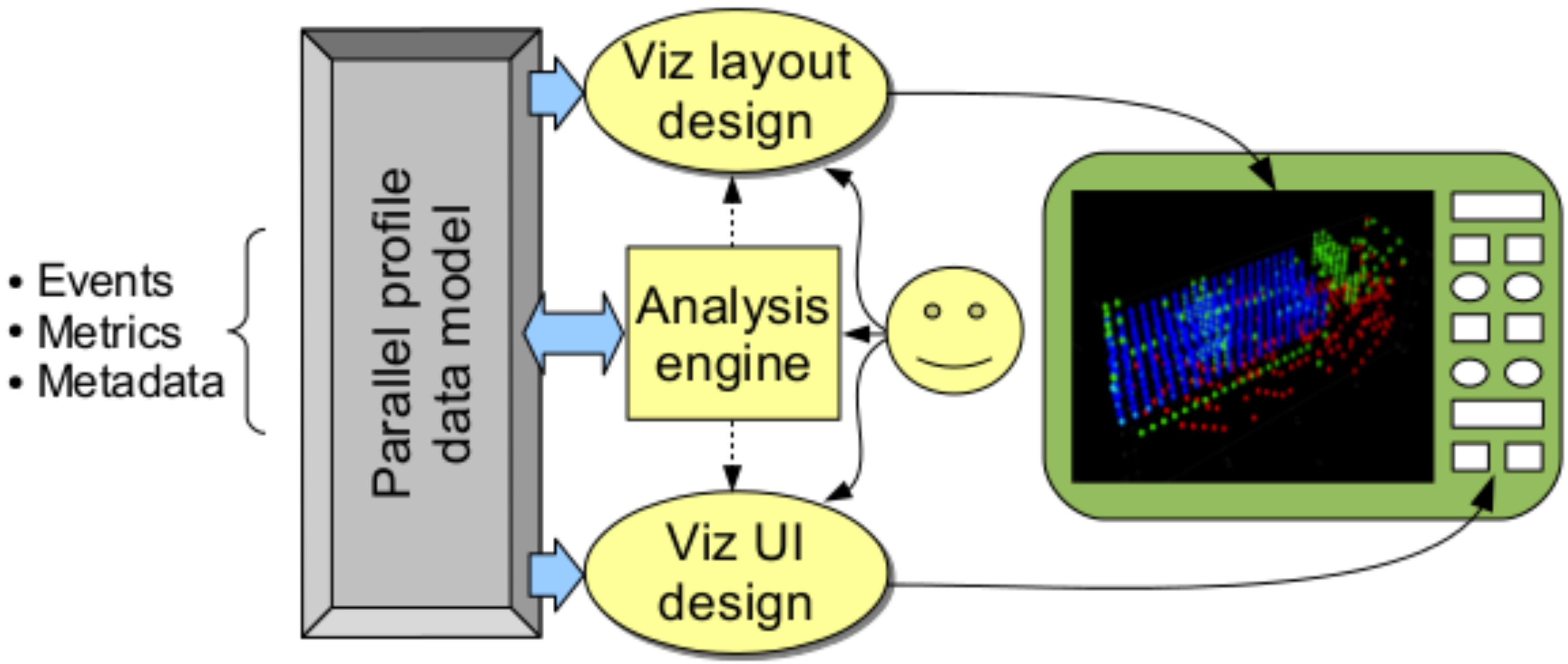
- Large performance data presents interpretation challenges
- Visualization aids in data exploration and pattern analysis
  - 3D visualization can help in identifying relations between events/metrics
- Existing tools provide "canned" views
  - TAU provides a few
    2D: bargraph, histogram
    3D: *full profile, correlation*





- Developing new visualizations is a challenge
  - Strategy 1: Create new view for each problem
  - Strategy 2: Use external visualization environment
- Provide high-level support to use within existing framework

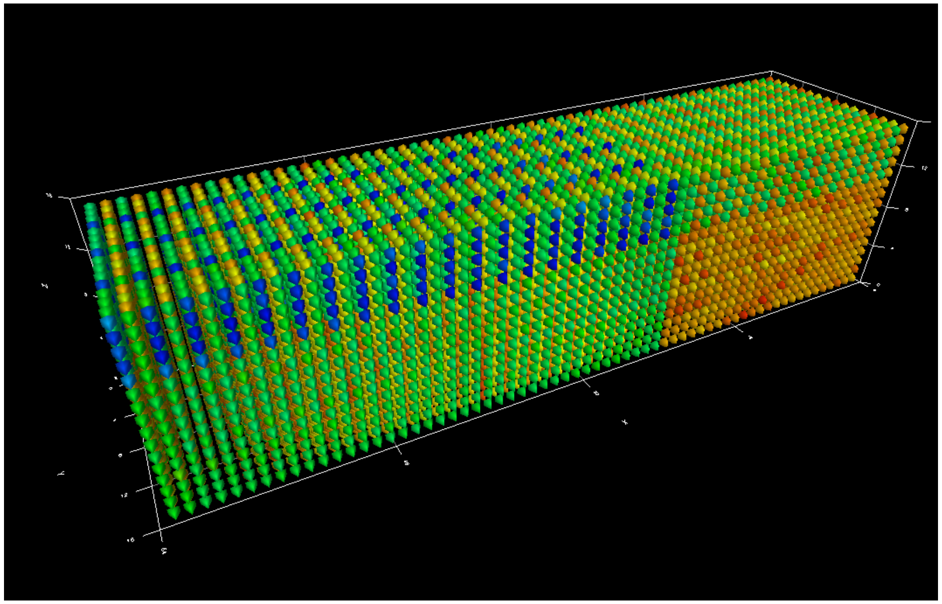
Extending Visualization Support in Profile Tool



- ❒ User defines visualization based on performance data model
- ❒ Specifies layout based on events, metrics, and metadata
- ❒ UI provides control of data binding and visualization

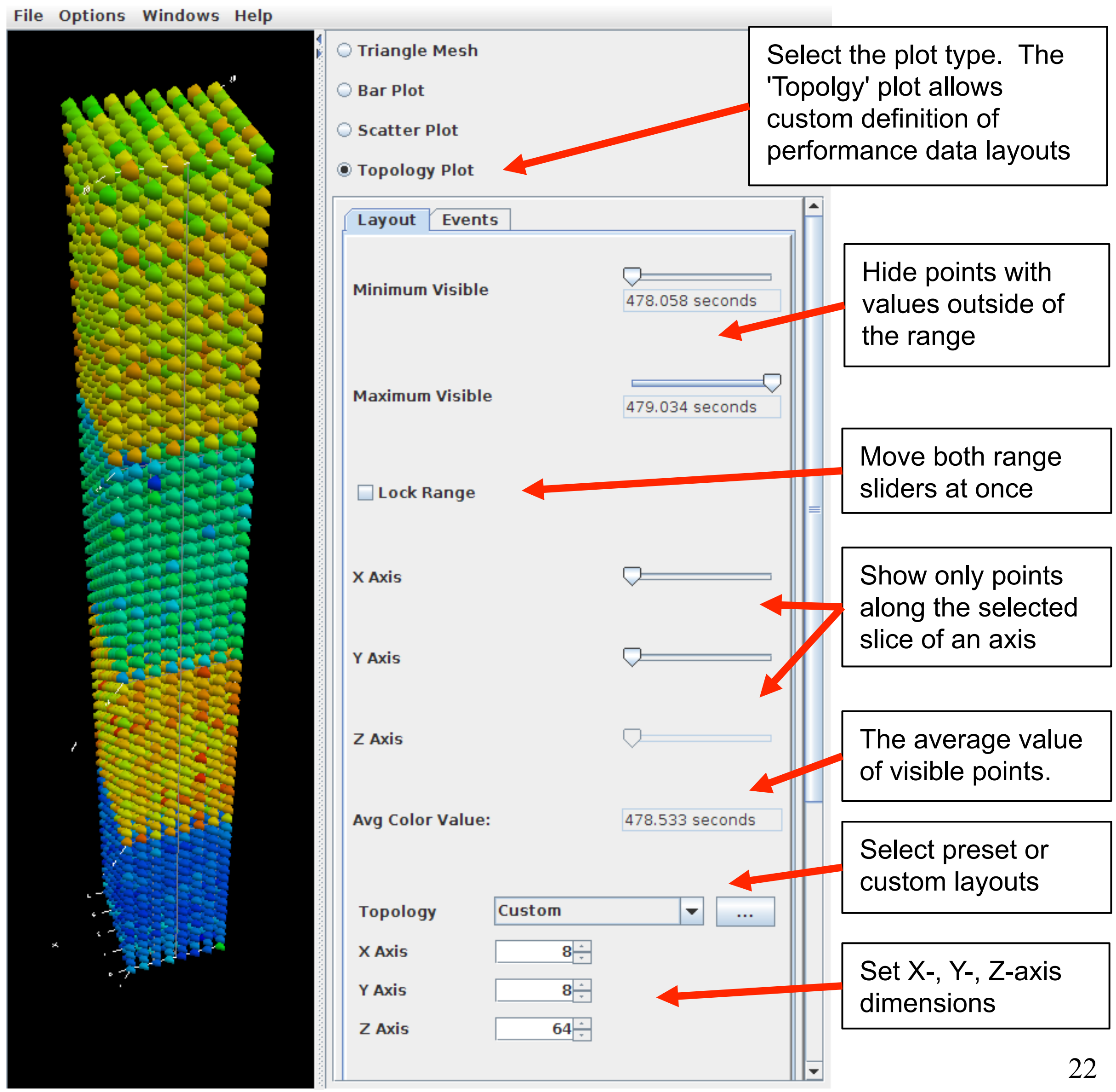
Using Process Topology Metadata

- Inspired by the CUBE topology display for BG/P
- Each point represents a thread of execution (MPI process)
  - Positioned according to the Cartesian (x,y,z,t) coordinates
- Color is determined by selected event/metric value
- Topology information can be recorded in TAU metadata
- ParaProf reads metadata to determine topology and create layout
- Sweep3D 16K run on BG/L
  - Color is exclusive time in the "sweep" function



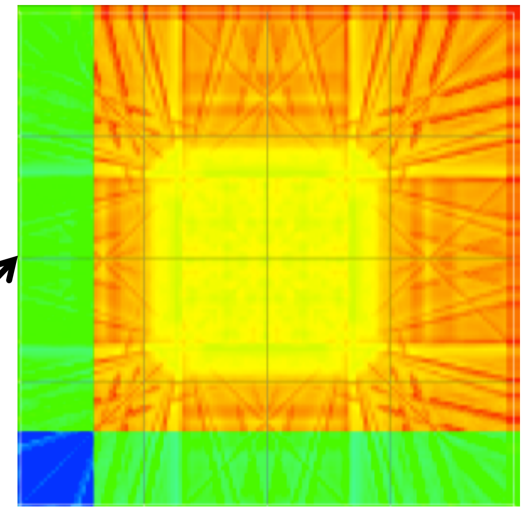
Topology Control UI

- *Layout* tab allows customization of the position and visibility of data points
- Performance event/metric data used to define color and position is selected in the *Event* tab
- Additional rendering options, such as color scale and point size are available
- 4k-core S3D run on BG/P

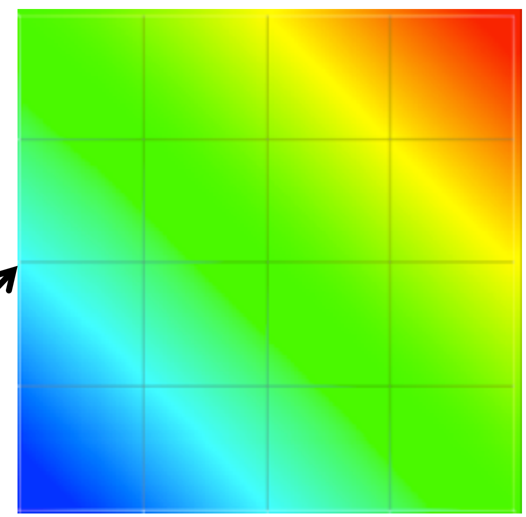


Alternate Topologies

- Certain views may hide deeper inter-process behavior
- Spatially dependent performance issues may be revealed by manipulating topology
- Sweep3D profile with alternative Cartesian mapping exposes distribution of computational effort

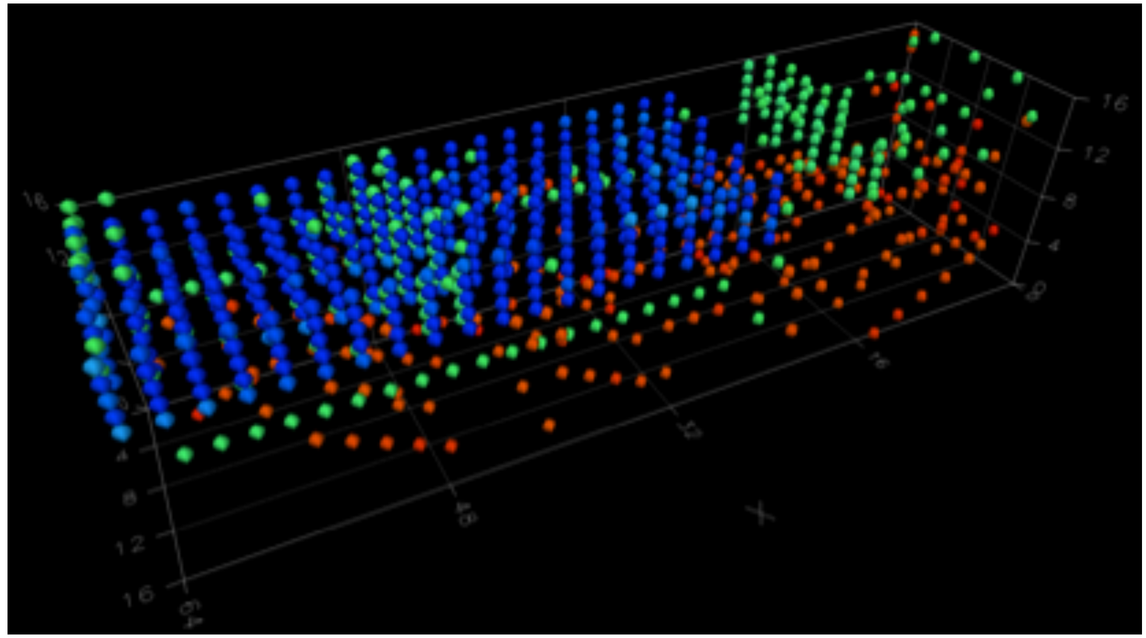


- Topology has direct effect on communication
- Visualization mapped to hardware topologies can suggest better node/rank mapping
- *MPI_AllReduce()* values for Sweep3D highlights waiting distribution from rank 0 (lower left) to the most distant rank (upper right)



Viewing Internal Structure

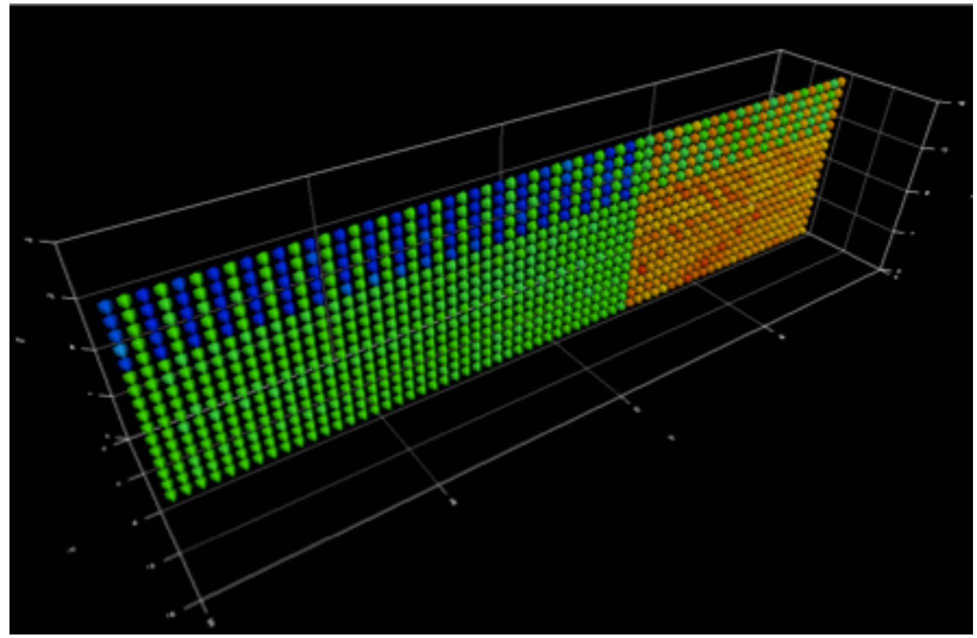
- Dense topologies can hide internal structure
- Restrict visibility by color value to expose performance patterns



- ParaProf visualization UI now allows for range filtering
  - Mid-level values can be excluded
  - Remaining points are:
    - high outliers (hotspots)
    - low outliers (underutilized nodes)

Slicing to Reduce Dimensionality

- Restrict visibility to slices along the spatial axes
- Multiple axis controls allow selection of planes, lines, or an individual point



- ParaProf visualization UI provides filtering behavior
  - Averaging the color value for all points in the selected area

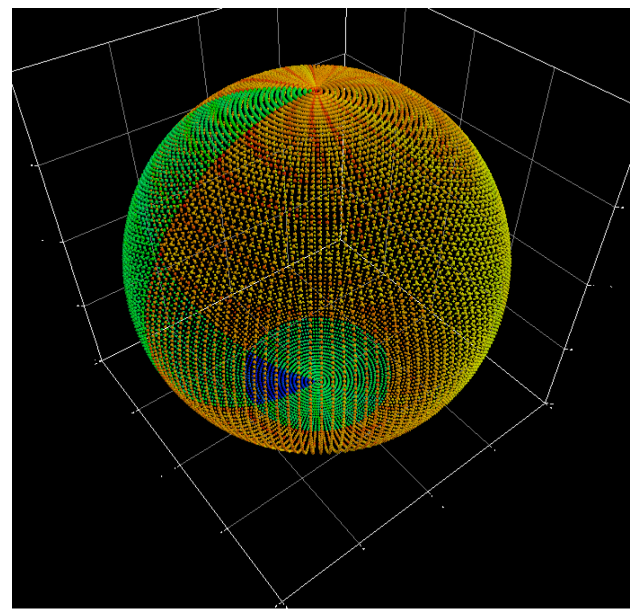
Visual Layout Specification

- Want to allow creation of explicit layouts
- Define a specification “language” that allows mathematical expressions to describe features of performance display
  - Equations define X, Y, Z coordinates and color per process
  - Event and metrics are seen as variables
    - *eventX.val* : value for *X*th specified event and metric
    - *eventX.{min,max,mean}* : global aggregate values
    - *atomicY* : *Y*th atomic event value
  - Intermediate variables can be used in the calculation
  - Defined global variables (e.g., max rank) are provided
- Specifications are loaded and processed by ParaProf
  - Use the MESP expression parser

Sphere Layout Specification

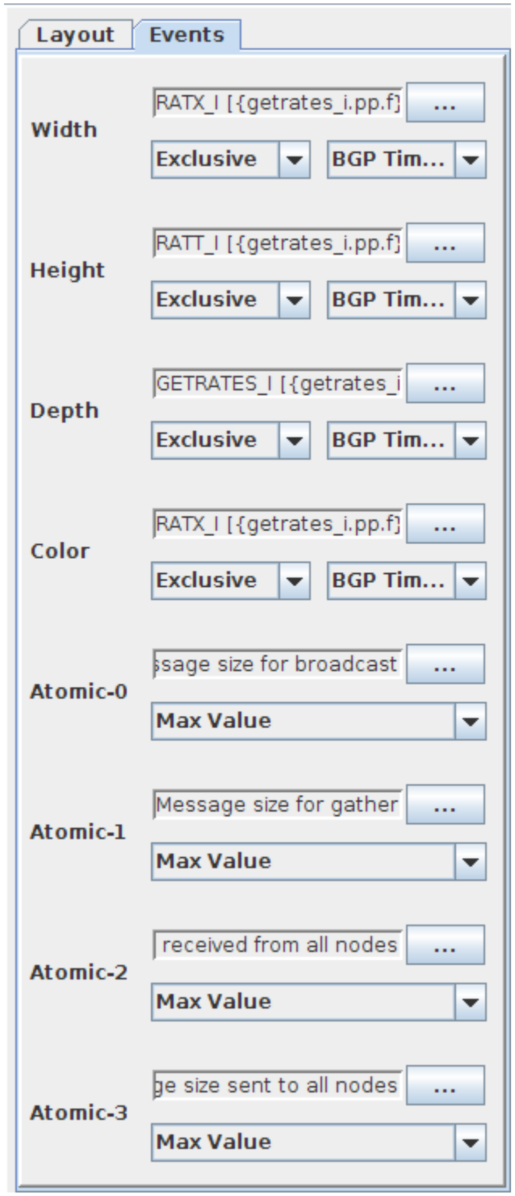
- Spatially mediated performance behavior may not be represented directly in topology metadata
  - Applications allocate resources with respect to a data-driven model
- The position of each point can be defined by custom equations in terms of event/metric, aggregate, atomic event and metadata
- Sweep3D profile mapped to a sphere

```
BEGIN_VIZ=Sphere
rootRanks=sqrt(maxRank)
theta=2*pi()/rootRanks*mod(rank,rootRanks)
phi=pi()/rootRanks*(ceil(rank/rootRanks))
x=cos(theta)*sin(phi)*100
y=sin(theta)*sin(phi)*100
z=cos(phi)*100
END_VIZ
```



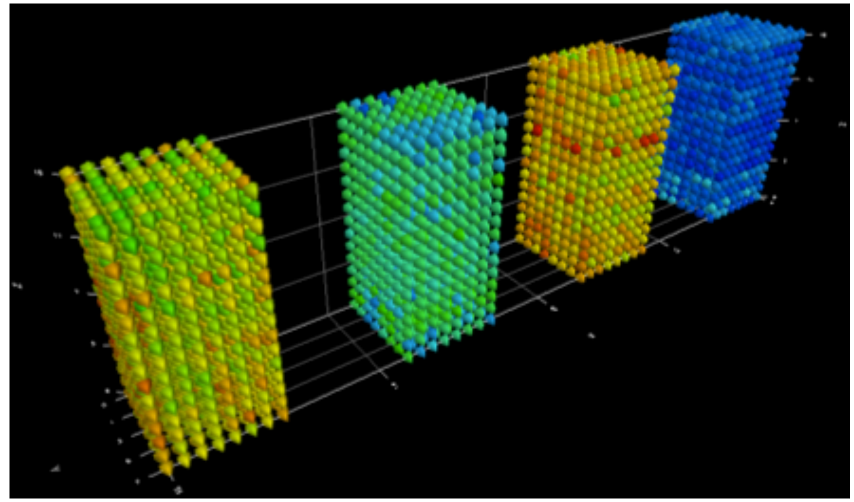
ParaProf Events Panel

- Events / metrics get bound in ParaProf UI
- Example:
  - *event0* is the FLOP count for function *foo*
  - *event1* is the time value for function *foo*
  - To set the X coordinate for each process point to the FLOPS for event *foo*:
    $x = event0.val / event1.val$
  - To set the Y coordinate for each process point to the global average FLOPS for event *foo*:
    $y = event0.mean / event1.mean$



Adding Dimensionality

- Topologies can involve more than three dimensions (e.g., intranode)
- Mirror actual machine layout to capture communication structure and cores
- Custom layouts allow specification of multiple points from a single process/rank
- 4K-core S3D run on BG/P
- Default topology only covers X, Y, Z coordinates
- A custom topology divides each *n*th core into its own block

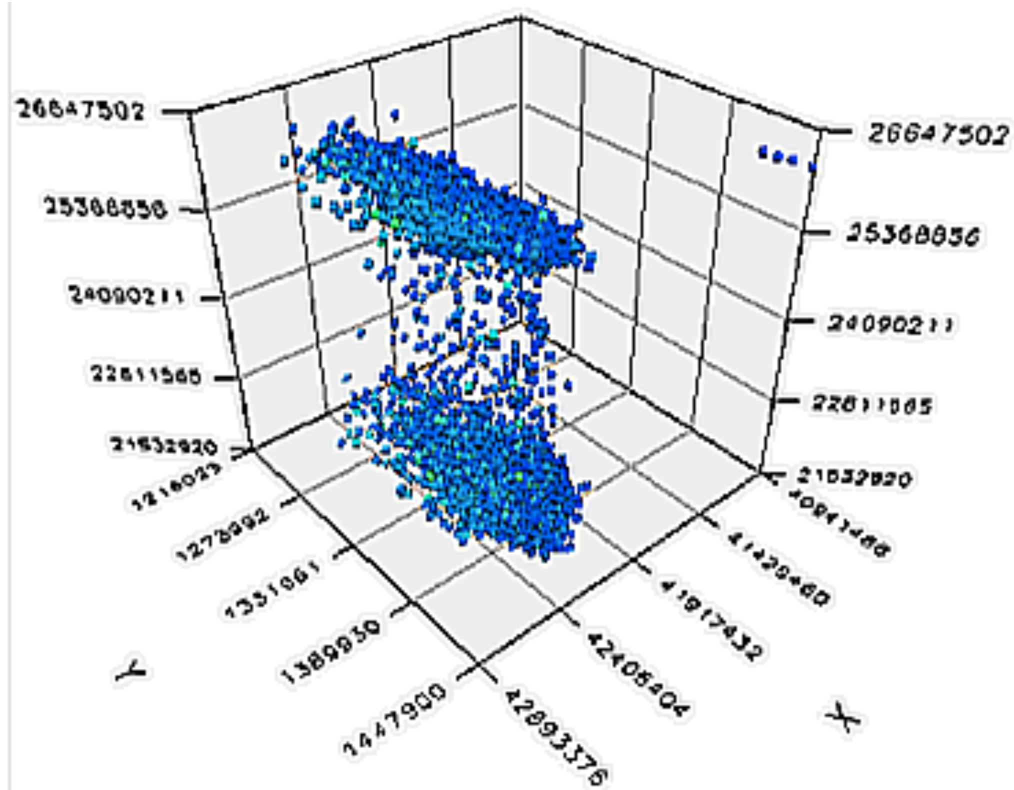


```
BEGIN_VIZ=4K_8x8x16Block
xdim=8
ydim=8
zdim=16

x=mod(rank,xdim)+16*floor(rank/
1024)
y=mod(floor(rank/xdim),ydim)
z=mod(floor(rank/xdim/ydim),zdim)
END_VIZ
```

Non-Spatial Relationships

- Positioning of points needs not be with respect to physical or data topology
- Correlation of metrics within the same events or events between processes can indicate relevant performance effects
- Partitioning or clustering of different processes based on selected performance criteria
- 3D scatterplot for 10240 core run of GCRM/ZGrd application
- Correlates four selected events, one for each spatial axis plus color




```
BEGIN_VIZ=ScatterTest
restrictDim=1
x=event0.val
y=event1.val
z=event2.val
END_VIZ
```

Visualizaton Next Steps

- Collect topology data from additional platforms (e.g. Cray)
- Expand UI for more general access to performance data model
- Allow independent manipulation of unconnected segments
- Improve presentation of data values, ranks, and metrics
- Better functionality for automatic higher-dimensional layouts
- Add representation of communication channels

TROIS

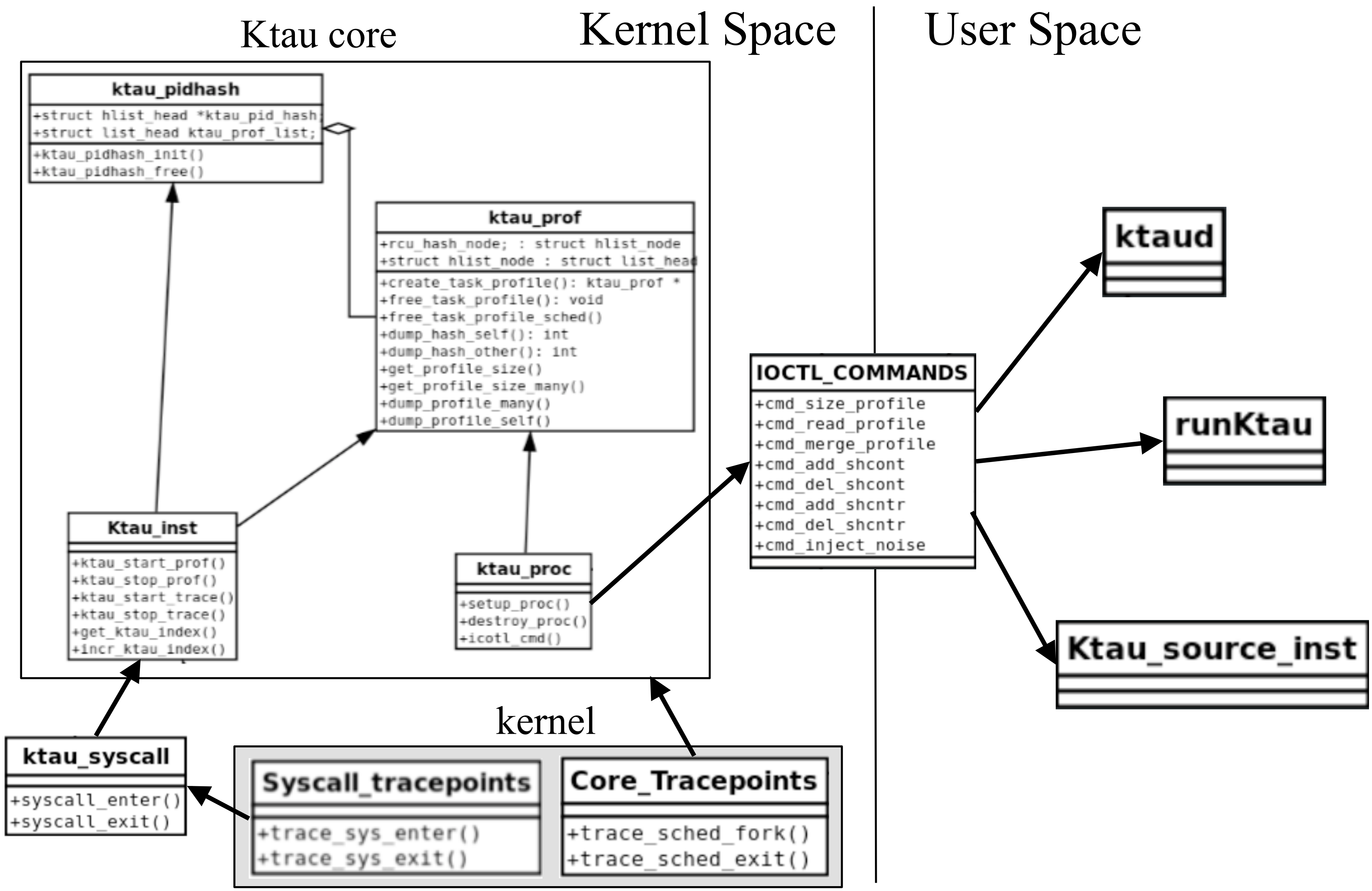
Kernel Measuremnt (KTAU) – Motivation

- Observe kernel performance and integrated with application performance measurements
- Earlier development of KTAU (Kernel TAU)
  - Profiling and tracing measurement of kernel events
  - Done via source instrumentation of Linux kernel
- Need more viable solution going forward
  - No patches or source modification
  - Easy to use and install
- Objective
  - Re-implement original KTAU features
  - Leverage work in the kernel instrumentation community

Approach

- Utilizes kernel infastructure for tracing
  - *tracepoints* and *kprobes*
- Simple user application with loadable kernel module
  - Similar to Unix “time”
- Efficient memory mapping between user and kernel space
- Minimal instructions required to record performance data
- Support for both profiling and tracing

Major Components of KTAU



KTAU and Other Projects

- What about Oprofile, LTTNG and SystemTap?
- These provide similar data from the kernel
- Way in which data is used and displayed is different perhaps
- KTAU focus is on application developers
- Comparing to LTTNG
  - Both use tracepoints, kprobes, RCU locking, kernel timestamp
  - KTAU
    - only requires kernel headers to build, root to install module
    - new development with targeted instrumentation
    - works with TAU to produce profiles and traces
  - LTTNG
    - mature with lots of instrumentation points
    - requires kernel patches
    - very basic user space instrumentation
    - no profile support

KTAU Status and Future

- Just finished initial prototype
- Undergoing more robust testing and evaluation
- Re-engineering of profile/trace merging tools

- Investigate interactions with LTTNG
  - Some movement towards including LTTNG in kernel
- Develop more efficient mechanism for accessing KTAU data
  - Use shared memory regions between kernel and user