TAU Parallel Performance System

PDC Summer School in HPC

Part 3: TAU Applications and Developments

Tutorial Outline – Part 3

TAU Applications and Developments

- Selected Applications
  - PETSc, EVH1, SAMRAI, Stommel
  - mpiJava, Blitz++, SMARTS
  - C-SAFE/Uintah
  - HYCOM, AVUS
- Current developments
  - PerfDMF
  - Online performance analysis
  - ParaVis
- Integrated performance evaluation environment

Case Study: PETSc v2.1.3 (ANL)

- Portable, Extensible Toolkit for Scientific Computation
- Scalable (parallel) PDE framework
  - Suite of data structures and routines (374,458 code lines)
  - Solution of scientific applications modeled by PDEs
- Parallel implementation
  - MPI used for inter-process communication
- TAU instrumentation
  - PDT for C/C++ source instrumentation (100%, no manual)
  - MPI wrapper interposition library instrumentation
- Example
  - Linear system of equations (Ax=b) (SLES) (ex2 test case)
  - Non-linear system of equations (SNES) (ex19 test case)

PETSc ex2 (Profile - wallclock time)

Mean Total Stat Window: /home/users/sameer/petsc/src/sles/examples/tutorials/pprof.dat

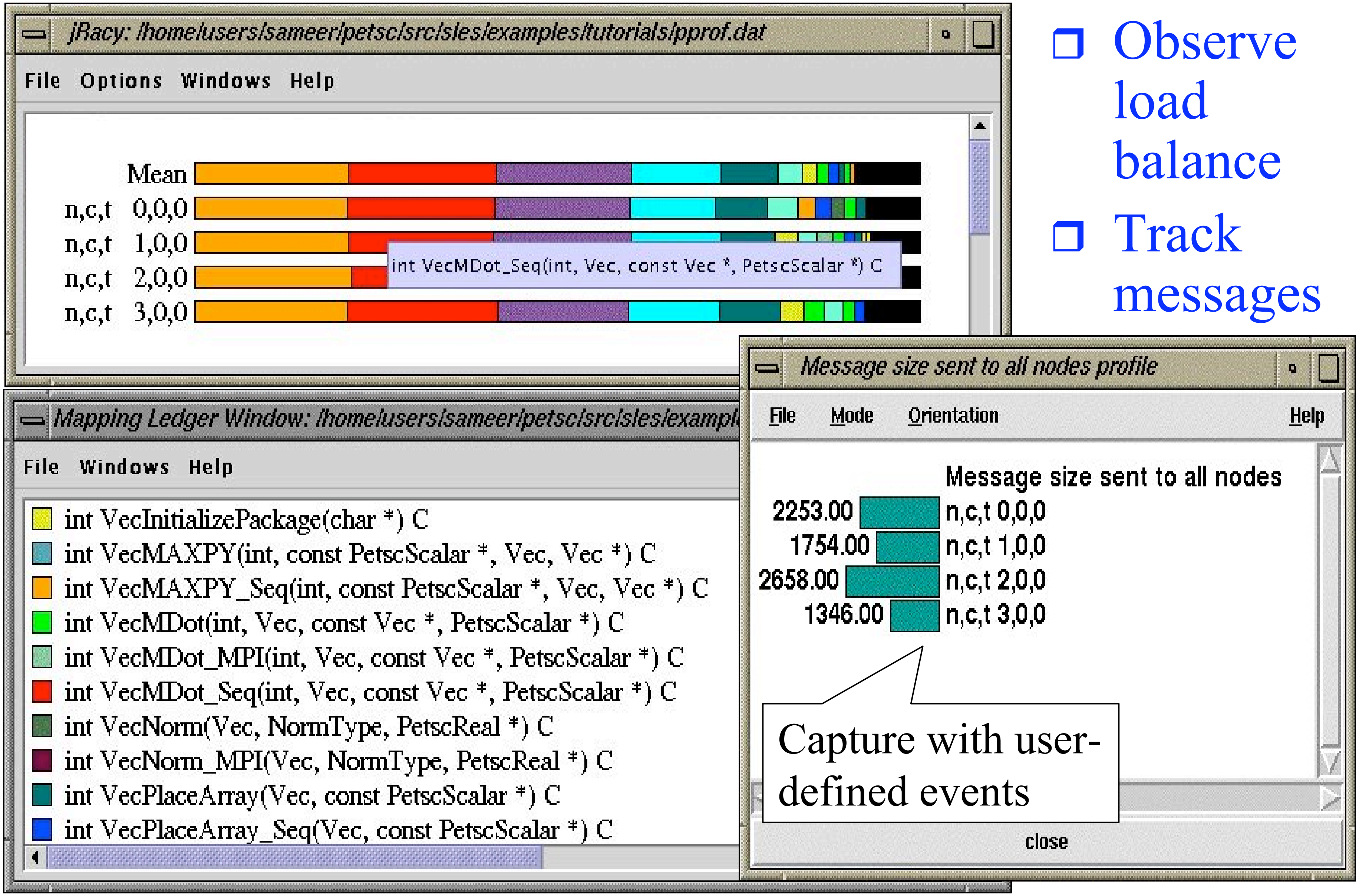
File Options Windows Help

Sorted with respect to exclusive time

%time	msec	total msec	#call	#subrs	usec/call	name
21.3	7,794	7,794	407	0	19152	int VecMAXPY_Seq(int, const PetscScalar *, Vec, V
20.6	7,534	7,534	393	0	19172	int VecMDot_Seq(int, Vec, const Vec *, PetscScala
18.8	6,886	6,908	407	1628	16973	int MatSolve_SeqAIJ_NaturalOrdering(Mat, Vec, Vec
12.5	4,548	4,599	407	1628	11302	int MatMult_SeqAIJ(Mat, Vec, Vec) C
7.9	2,877	2,877	1353.75	0	2126	MPI_Recv()
4.9	1,282	1,801	49800	49800	36	int MatSetValues(Mat, int, int *, int, int *, Pet
100.0	785	36,651	1	49832	36651451	int main(int, char **) C
1.7	618	627	407	1628	1543	int MatMultAdd_SeqAIJ_Inode(Mat, Vec, Vec, Vec) C
1.4	519	519	49800	0	10	int MatSetValues_MPIAIJ(Mat, int, int *, int, int
0.9	337	337	1	35	337700	MPI_Init()
1.1	328	394	3142	15205	126	int PetscOptionsFindPair_Private(const char *, co
0.7	233	240	182	649	1320	int PetscFListGetPathAndFunction(const char *, ch
1.3	219	463	153	1110	3032	int PetscFListAdd(PetscFList *, const char *, con
0.6	215	215	1526.25	0	141	int PetscStackCopy(PetscStack *, PetscStack *) C

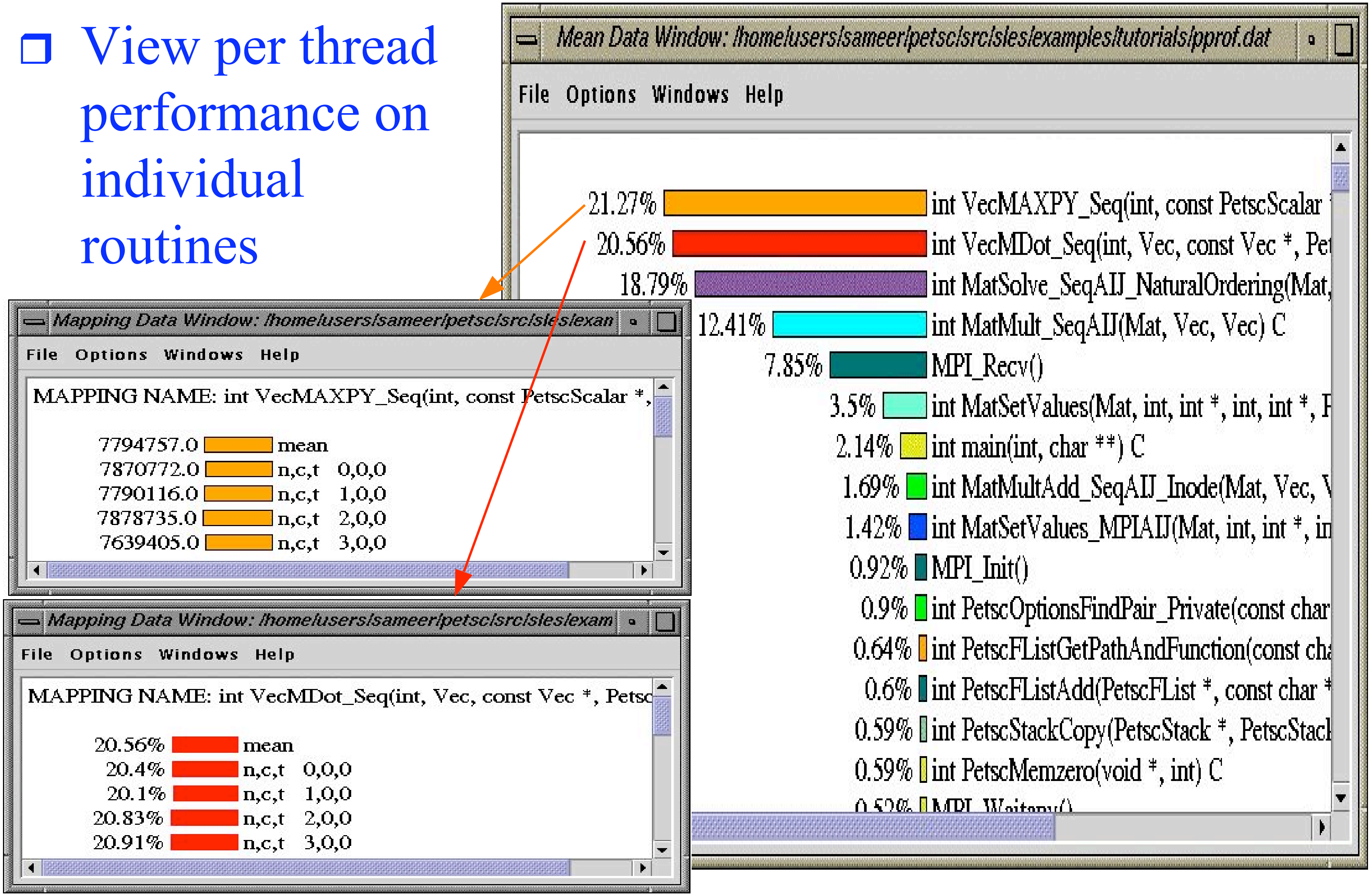
PETSc ex2(Profile - overall and message counts)

- Observe load balance
- Track messages

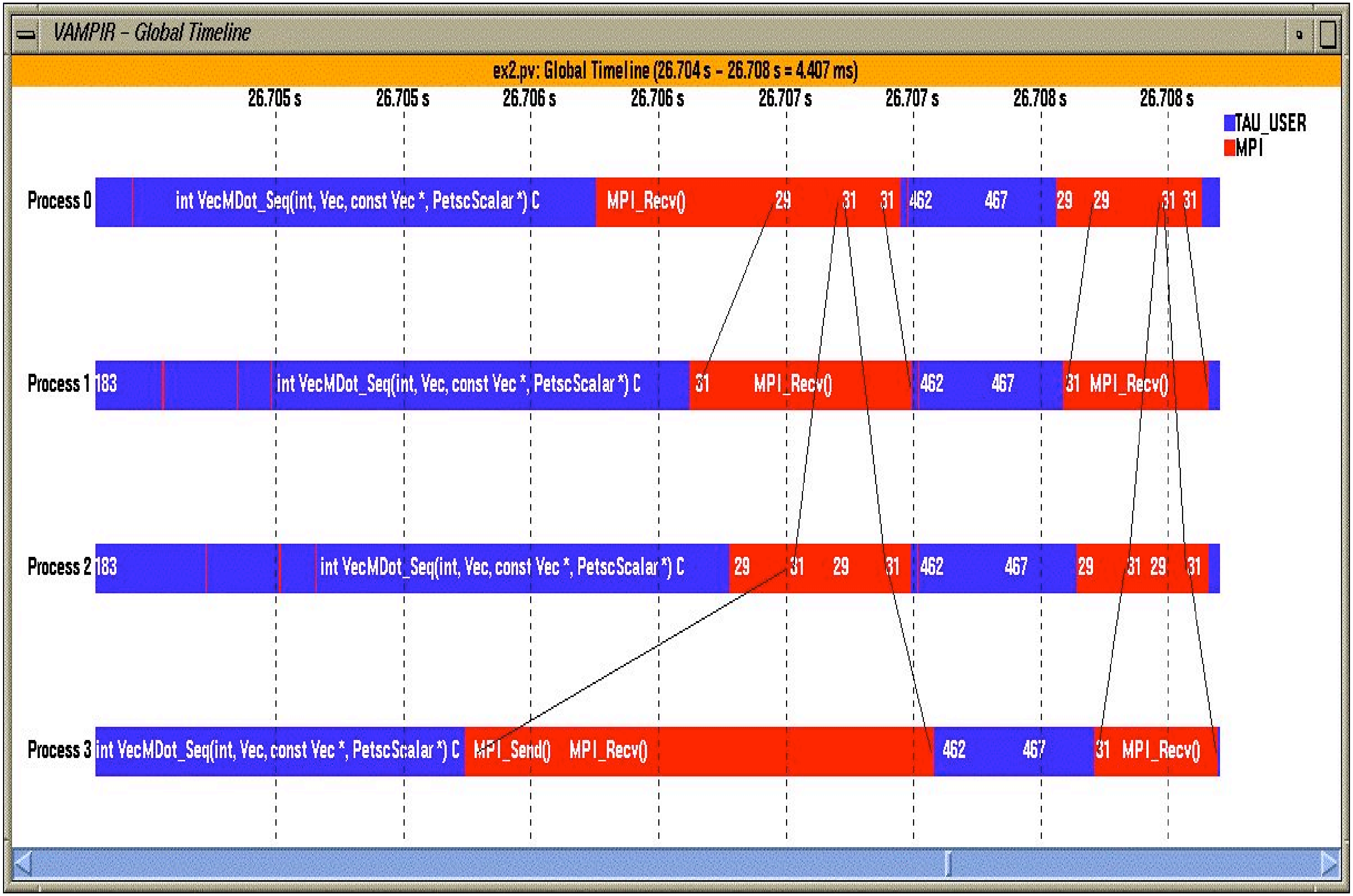


PETSc ex2 (Profile - percentages and time)

- View per thread performance on individual routines



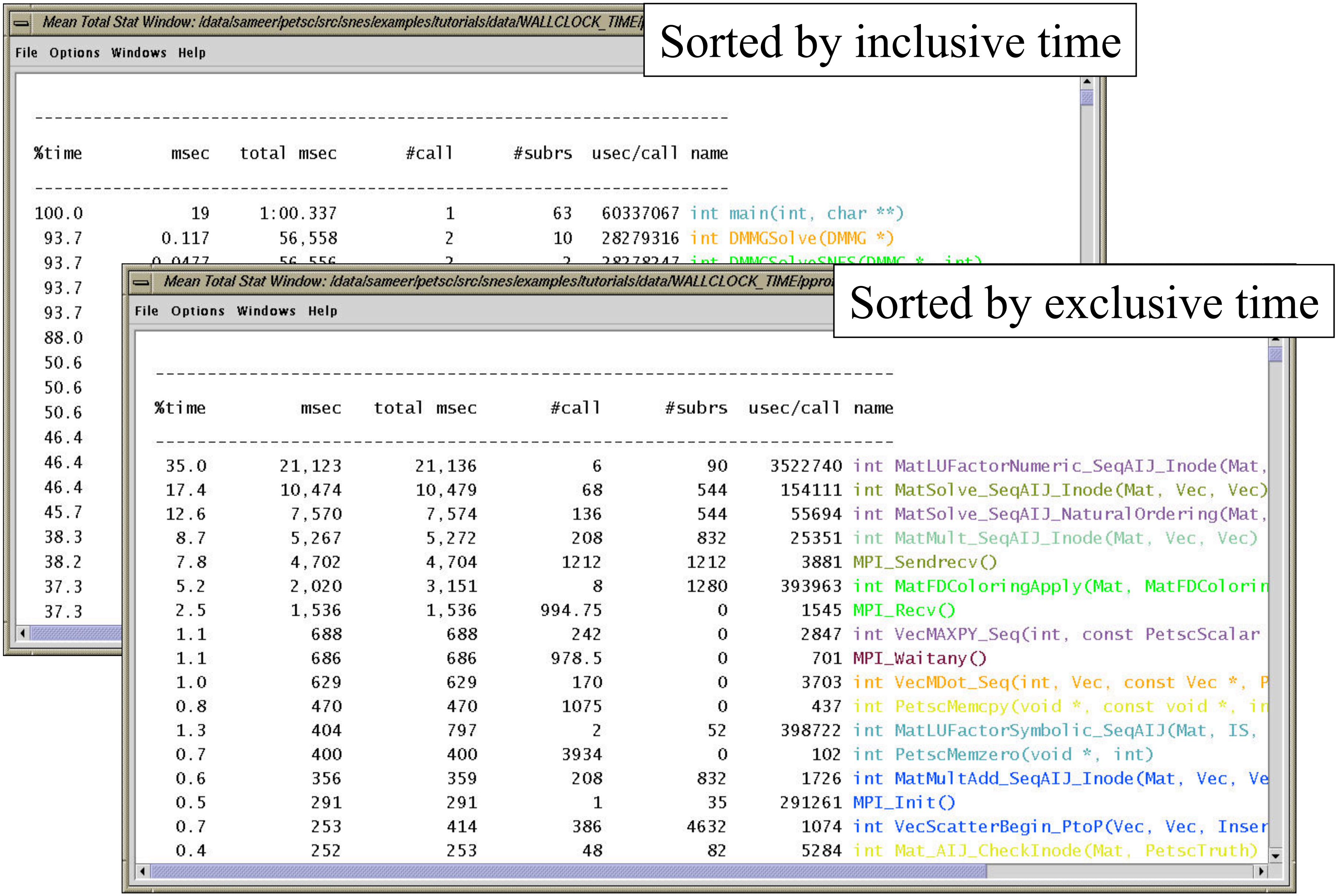
PETSc ex2 (Trace)



PETSc ex19

- Non-linear solver (SNES)
  - 2-D driven cavity code
  - Uses velocity-vorticity formulation
  - Finite difference discretization on a structured grid
- Problem size and measurements
  - 56x56 mesh size on quad Pentium III (550 Mhz, Linux)
  - Executes for approximately one minute
  - MPI wrapper interposition library
  - PDT (*tau_instrumentor*)
  - Selective instrumentation (*tau_reduce*)
    - three routines identified with high instrumentation overhead

PETSc ex19 (Profile - wallclock time)



Mean Total Stat Window: /data/sameer/petsc/src/snes/examples/tutorials/data/WALLCLOCK_TIME/

File Options Windows Help

%time	msec	total msec	#call	#subrs	usec/call	name
100.0	19	1:00.337	1	63	60337067	int main(int, char **)
93.7	0.117	56,558	2	10	28279316	int DMMGSolve(DMMG *)
93.7	0.0477	56,556	2	2	28278247	int DMMGSolveSNES(DMMG *, int)
93.7						
93.7						
88.0						
50.6						
50.6						
50.6						
46.4						
46.4						
46.4						
45.7						
38.3						
38.2						
37.3						
37.3						

Sorted by inclusive time

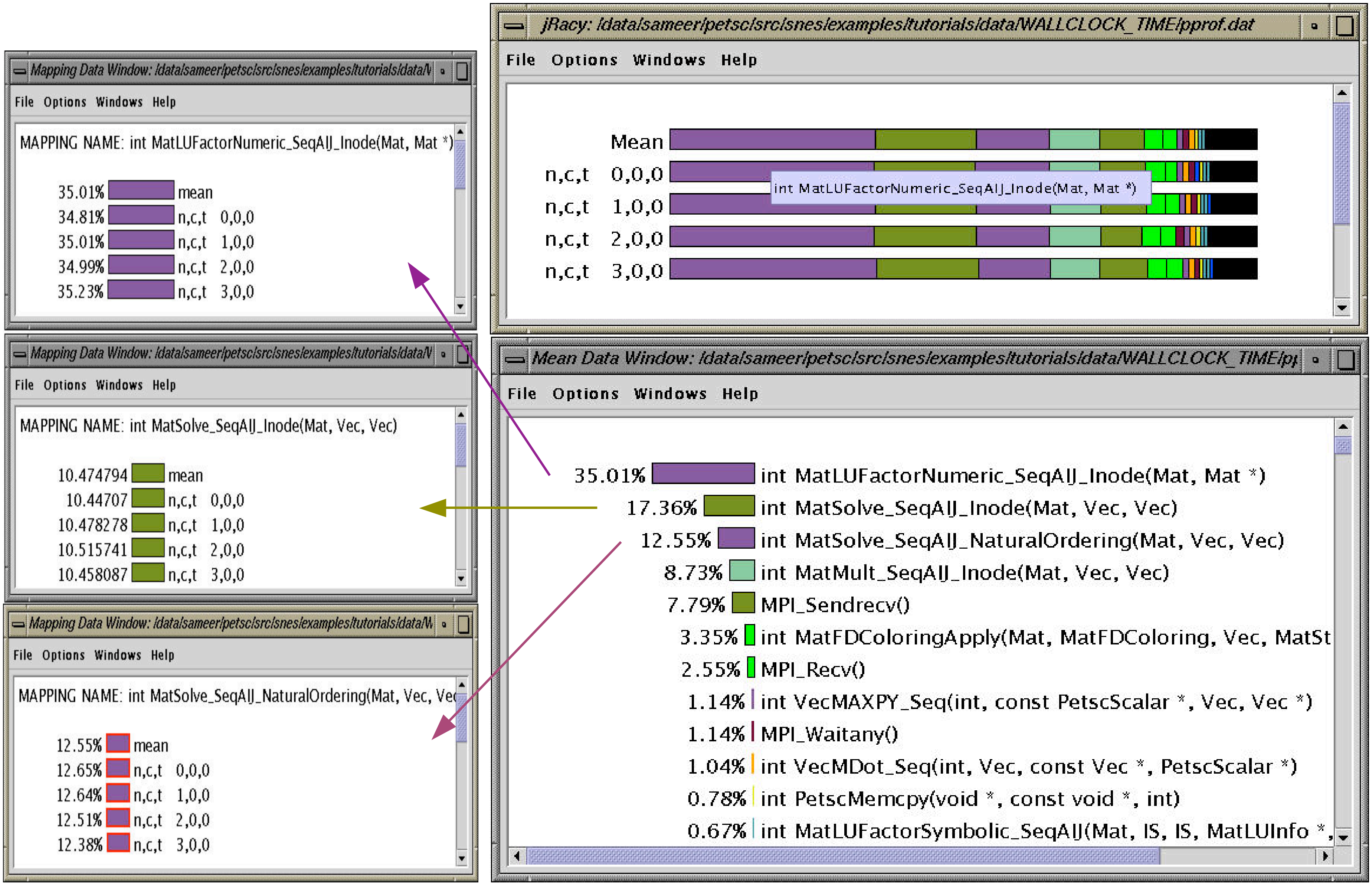
Mean Total Stat Window: /data/sameer/petsc/src/snes/examples/tutorials/data/WALLCLOCK_TIME/ppro

File Options Windows Help

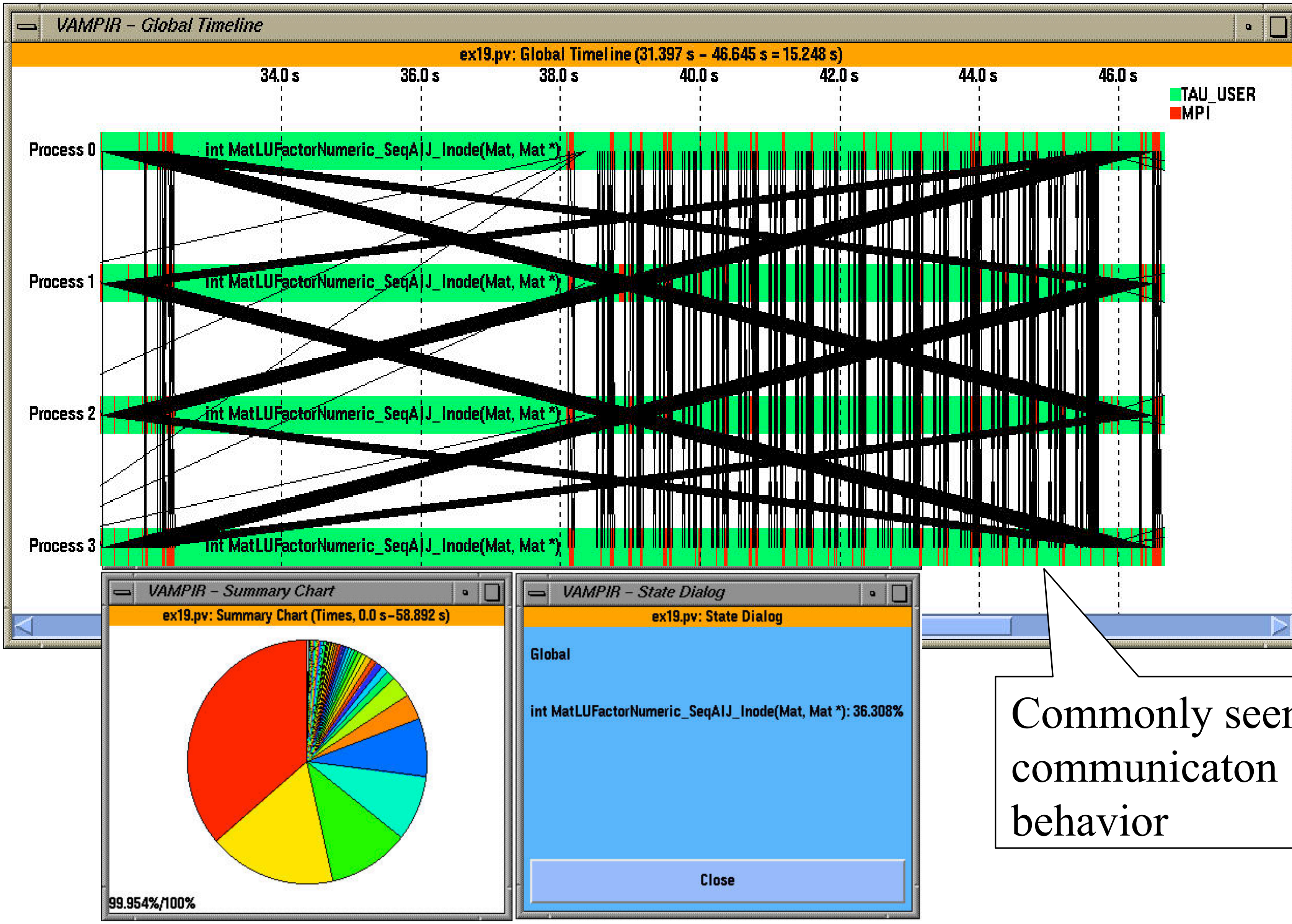
%time	msec	total msec	#call	#subrs	usec/call	name
35.0	21,123	21,136	6	90	3522740	int MatLUFactorNumeric_SeqAIJ_Inode(Mat,
17.4	10,474	10,479	68	544	154111	int MatSolve_SeqAIJ_Inode(Mat, Vec, Vec)
12.6	7,570	7,574	136	544	55694	int MatSolve_SeqAIJ_NaturalOrdering(Mat,
8.7	5,267	5,272	208	832	25351	int MatMult_SeqAIJ_Inode(Mat, Vec, Vec)
7.8	4,702	4,704	1212	1212	3881	MPI_Sendrecv()
5.2	2,020	3,151	8	1280	393963	int MatFDColoringApply(Mat, MatFDColorin
2.5	1,536	1,536	994.75	0	1545	MPI_Recv()
1.1	688	688	242	0	2847	int VecMAXPY_Seq(int, const PetscScalar
1.1	686	686	978.5	0	701	MPI_Waitany()
1.0	629	629	170	0	3703	int VecMDot_Seq(int, Vec, const Vec *, P
0.8	470	470	1075	0	437	int PetscMemcpy(void *, const void *, in
1.3	404	797	2	52	398722	int MatLUFactorSymbolic_SeqAIJ(Mat, IS,
0.7	400	400	3934	0	102	int PetscMemzero(void *, int)
0.6	356	359	208	832	1726	int MatMultAdd_SeqAIJ_Inode(Mat, Vec, Ve
0.5	291	291	1	35	291261	MPI_Init()
0.7	253	414	386	4632	1074	int VecScatterBegin_PtoP(Vec, Vec, Inser
0.4	252	253	48	82	5284	int Mat_AIJ_CheckInode(Mat, PetscTruth)

Sorted by exclusive time

PETSc ex19 (Profile - overall and percentages)

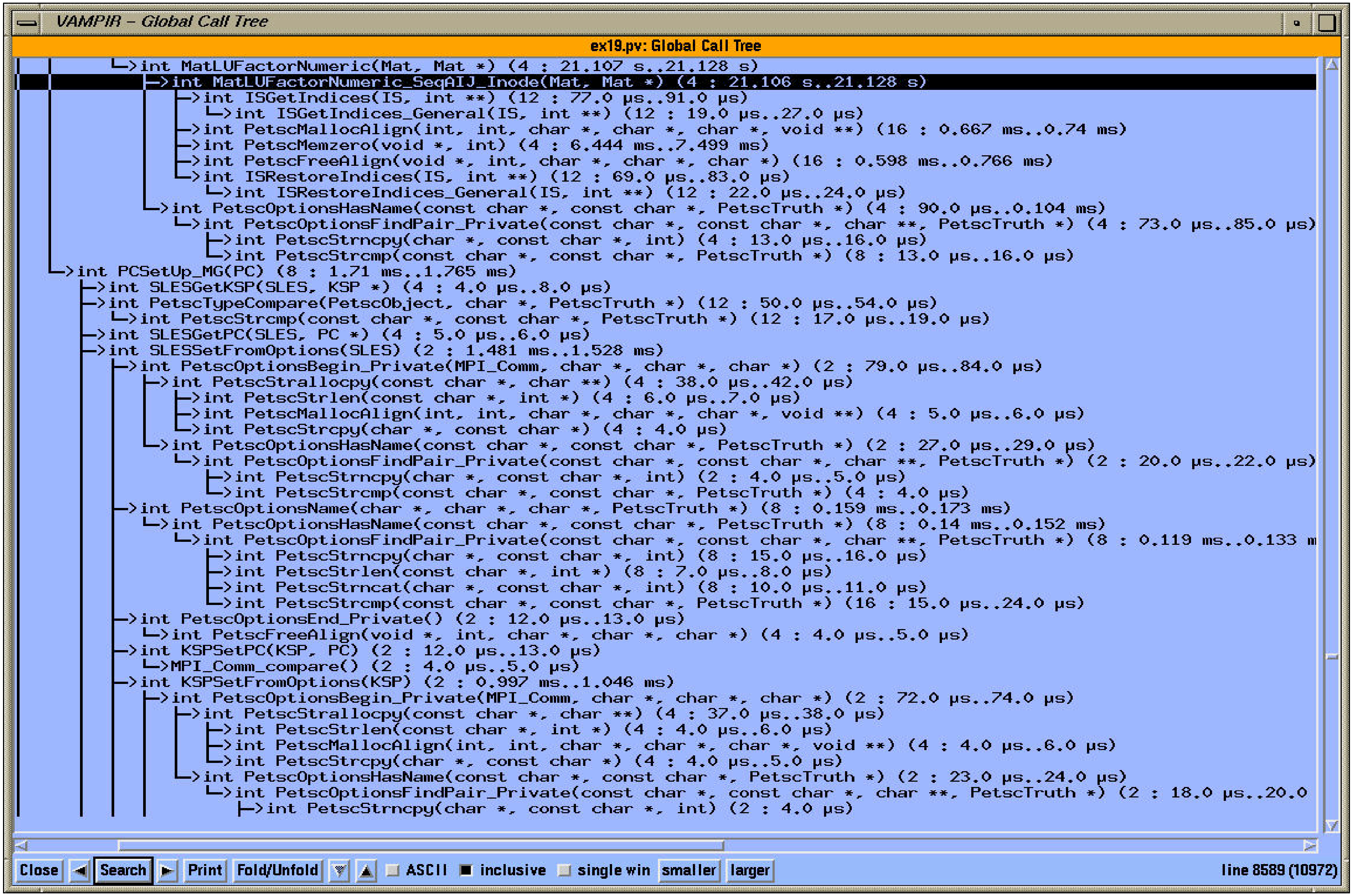


PETSc ex19 (Tracing)

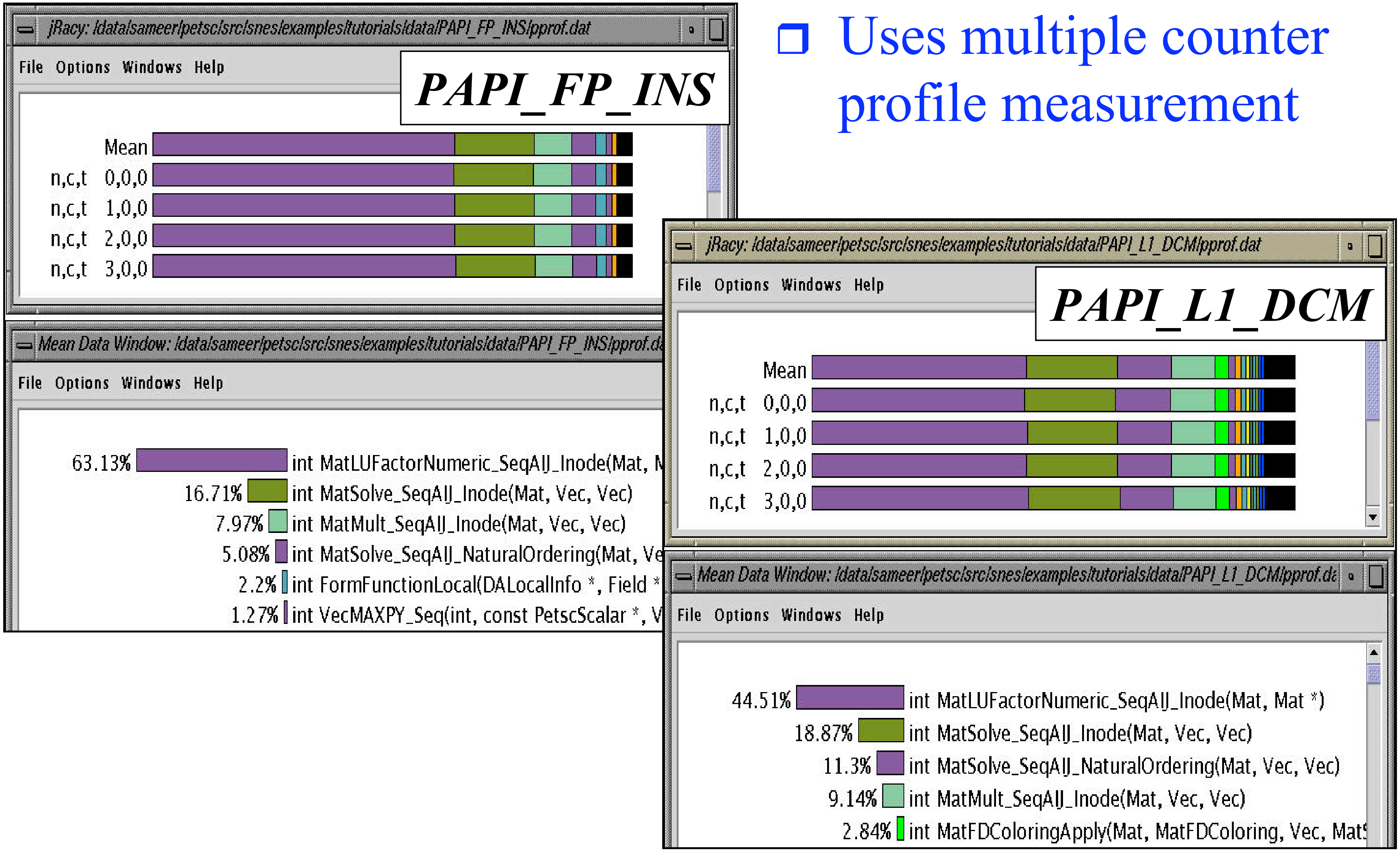


Commonly seen communicaton behavior

PETSc ex19 (Tracing - callgraph)



PETSc ex19 (PAPI_FP_INS, PAPI_L1_DCM)



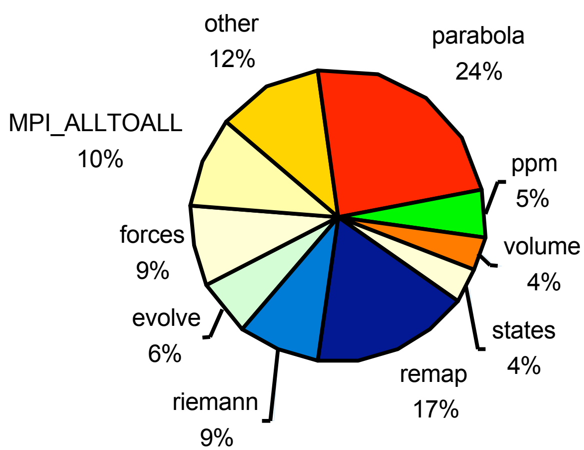
- Uses multiple counter profile measurement

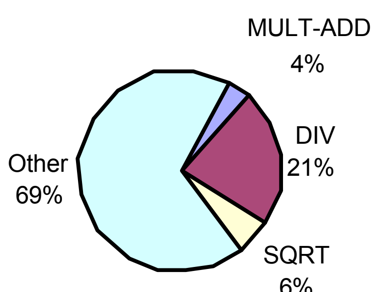
EVH1 – High Energy and Nuclear Physics

- Enhanced Virginia Hydrodynamics #1 (EVH1)
  - "TeraScale Simulations of Neutrino-Driven Supernovae and Their Nucleosynthesis" SciDAC project
  - Configured to run a simulation of the Sedov-Taylor blast wave solution in 2D spherical geometry
- EVH1 communication bound for > 64 processors
  - Predominant routine (>50% of execution time) at this scale is MPI_ALLTOALL
  - Used in matrix transpose-like operations
  - Current implementation uses 1D matrix decomposition
- PERC benchmark code

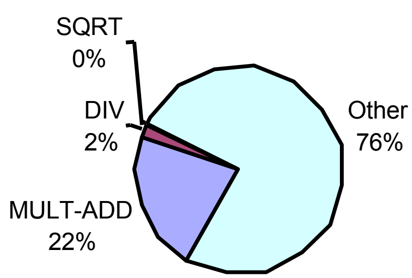
EVH1 Aggregate Performance

- Aggregate performance measures over all tasks for .1 second simulation
  - Using PAPI on IBM SP

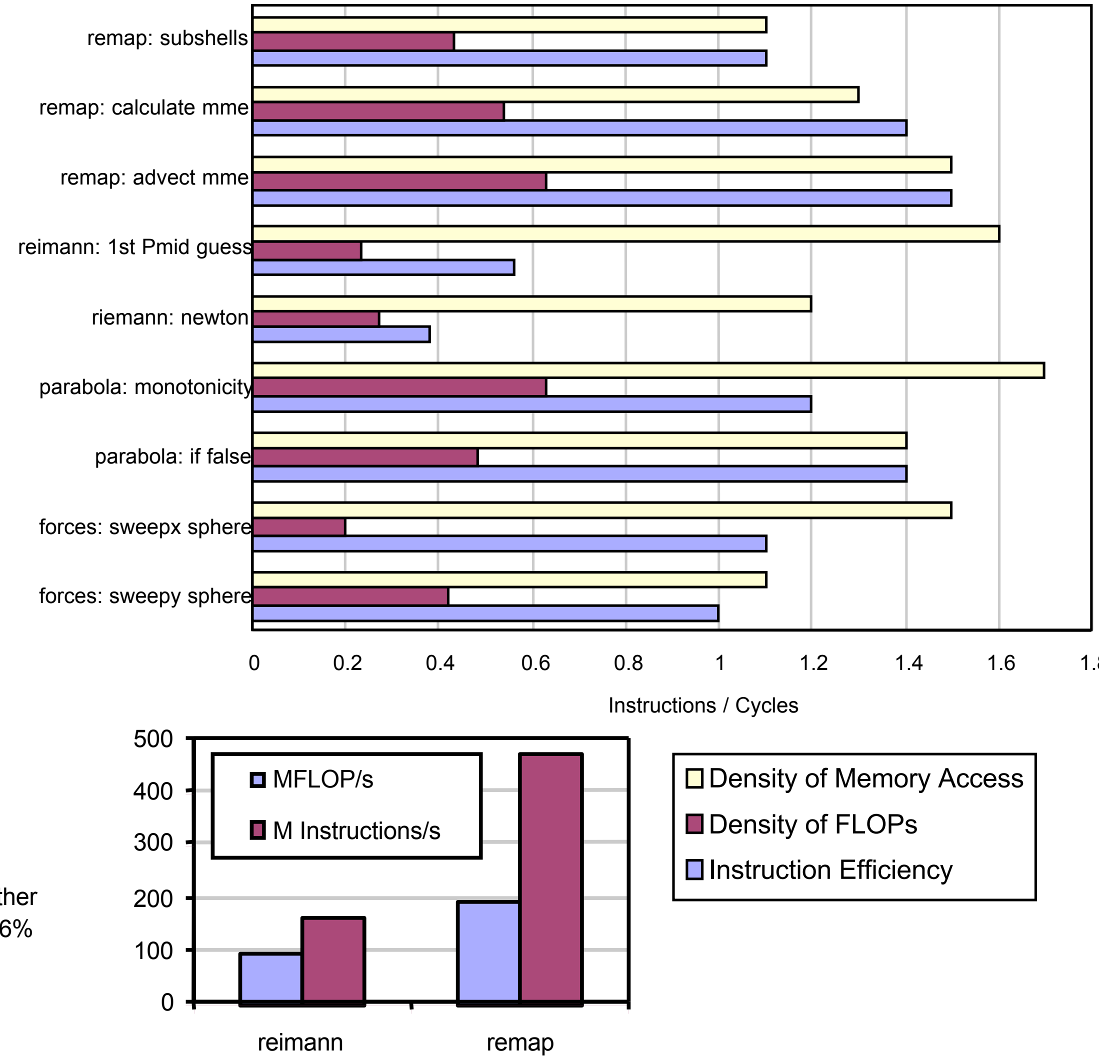




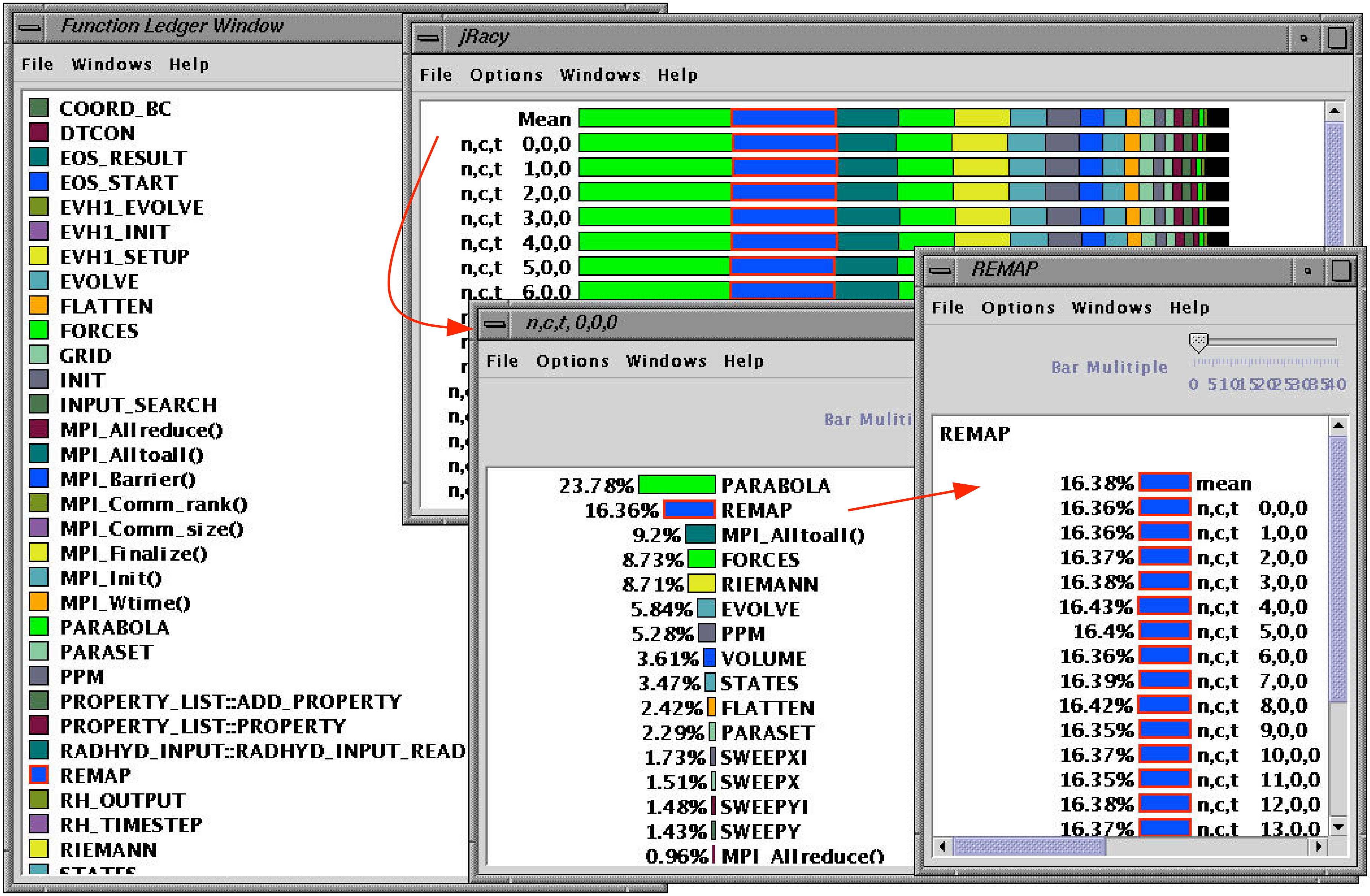
RIEMANN
Floating Point



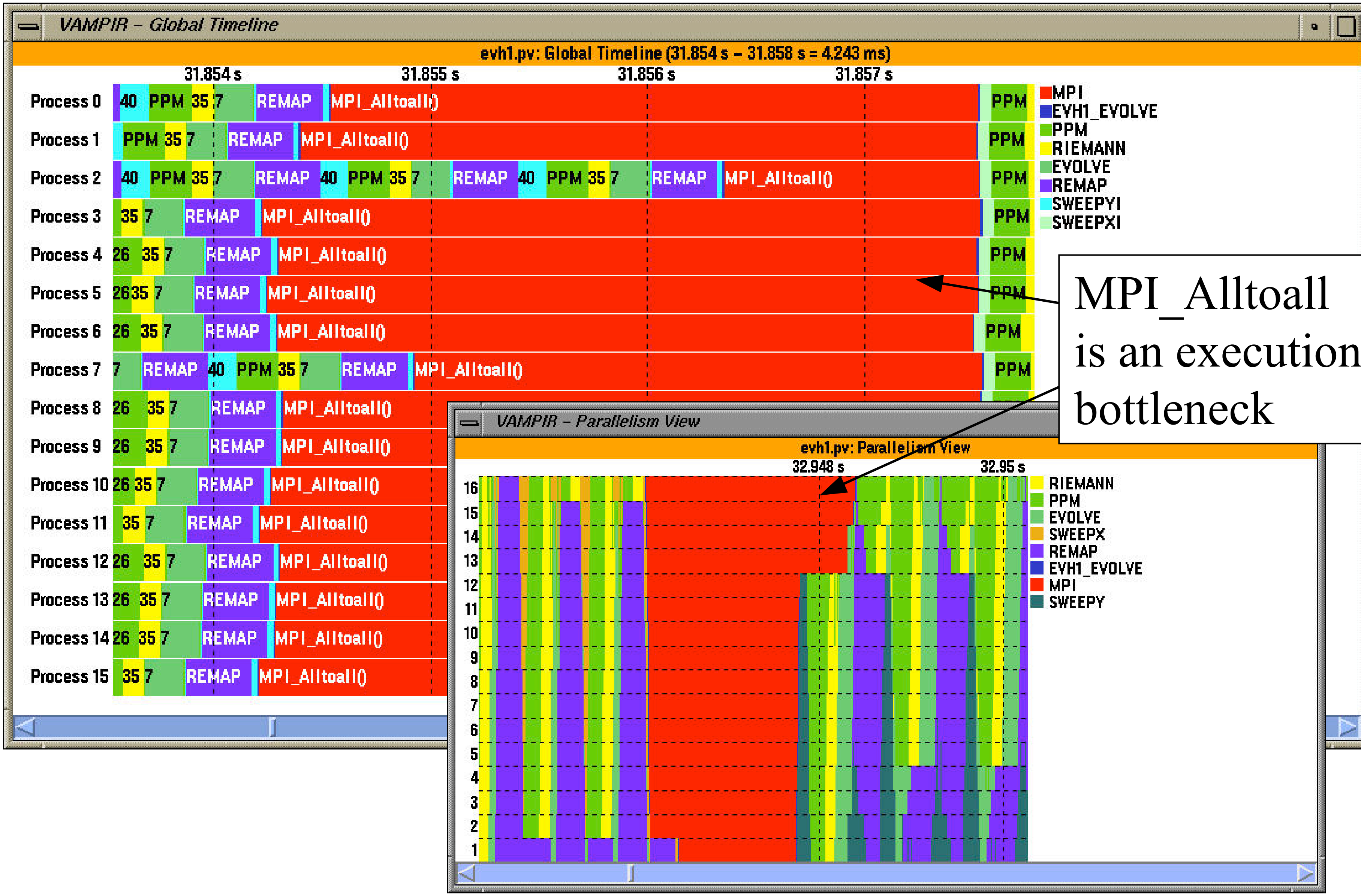
REMAP
Floating Point



EVH1 Execution Profile



EVH1 Execution Trace



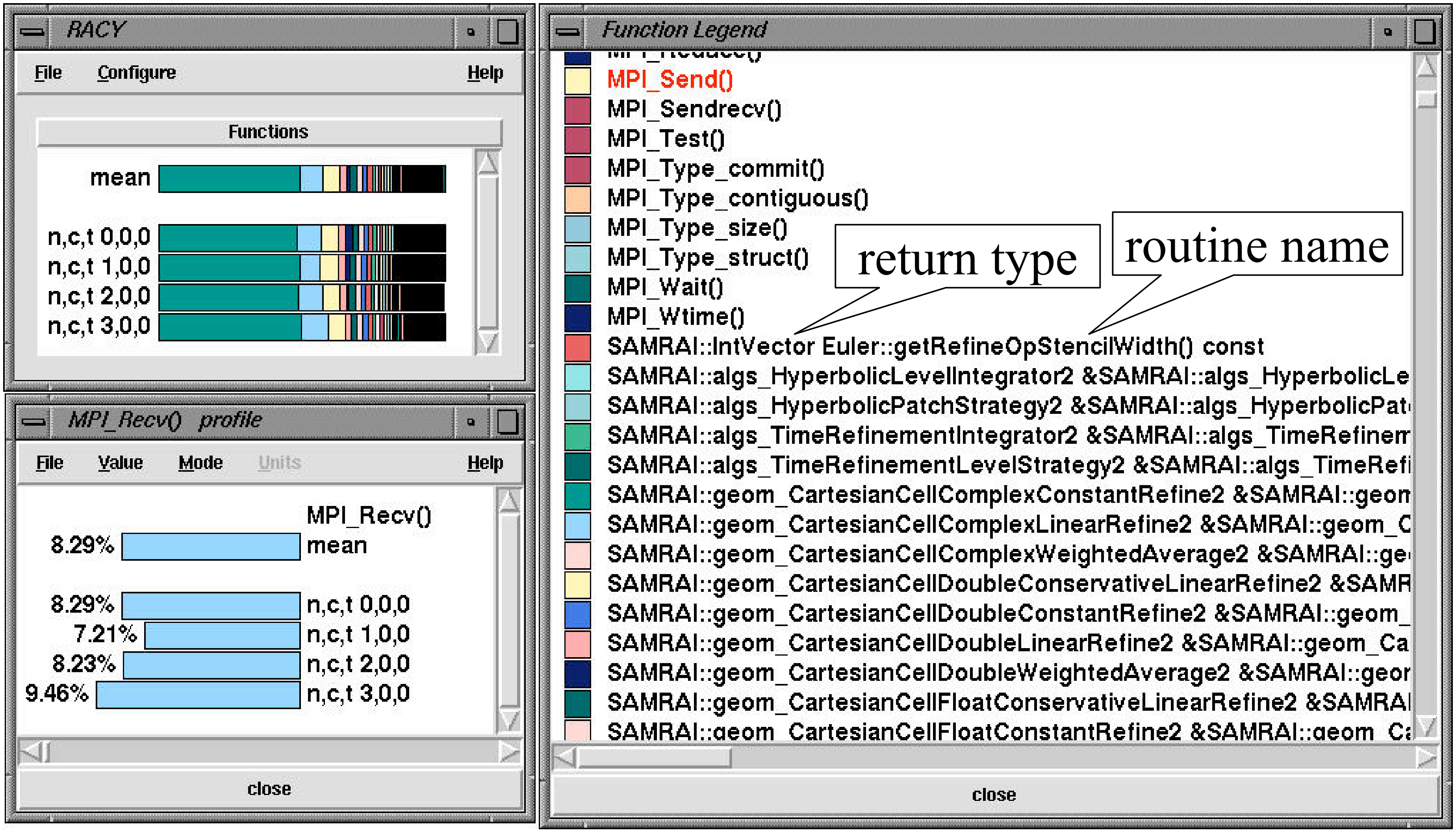
MPI_Alltoall is an execution bottleneck

SAMRAI

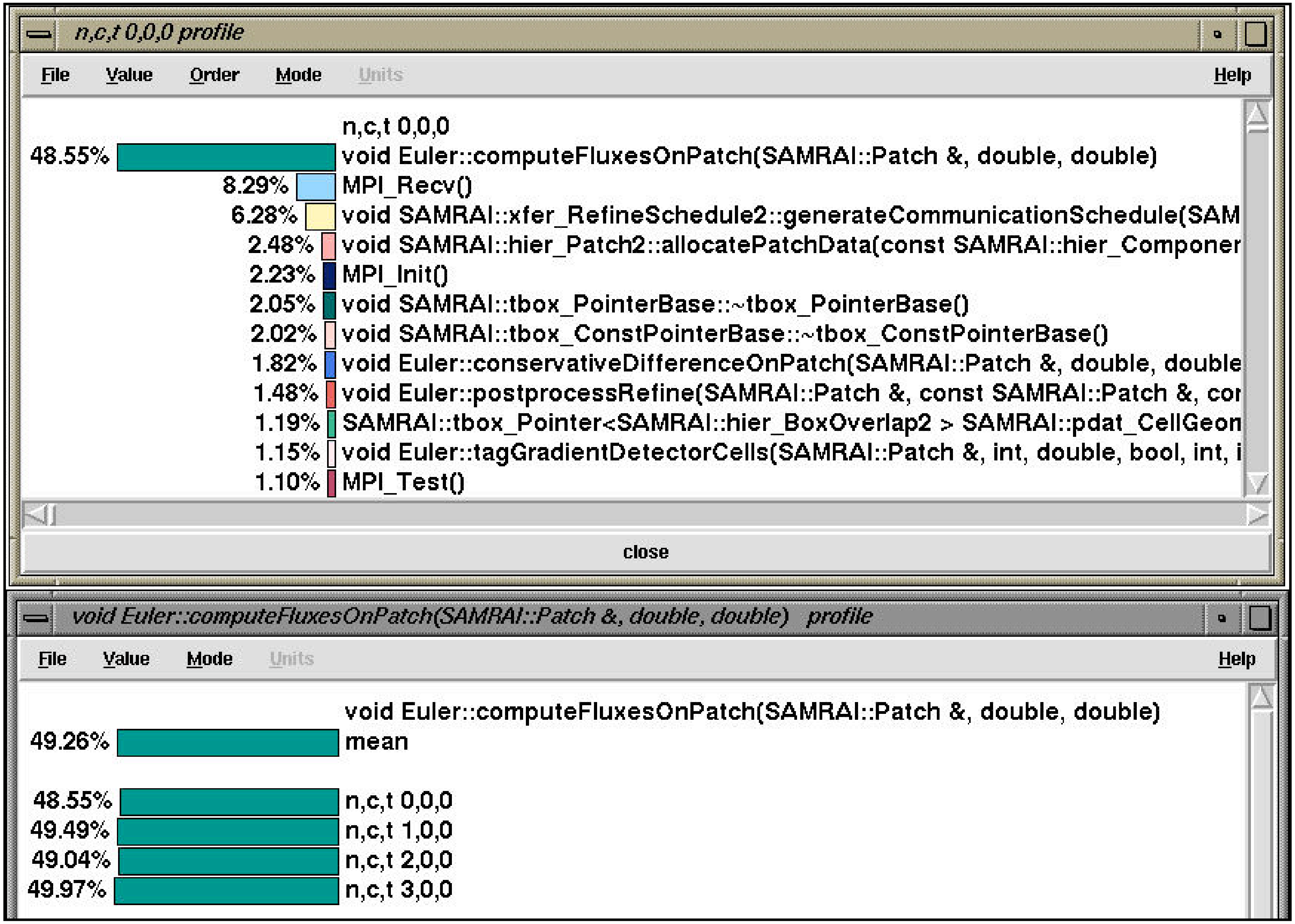
- Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI)
  - Andy Wissink (LLNL)
- Programming
  - C++ and MPI
  - SPMD
- Instrumentation
  - PDT for automatic instrumentation of routines
  - MPI interposition wrappers
  - SAMRAI timers for interesting code segments
    - timers classified in groups (*apps*, *mesh*, …)
    - timer groups are managed by TAU groups

SAMRAI Execution Profile

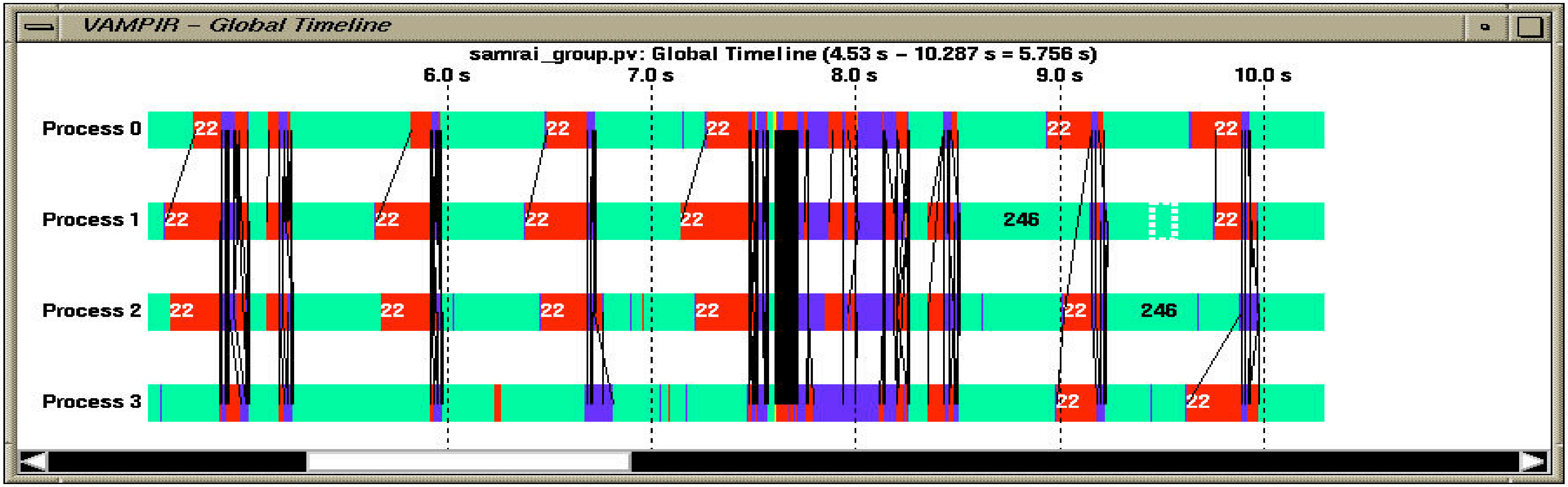
- Euler (2D)

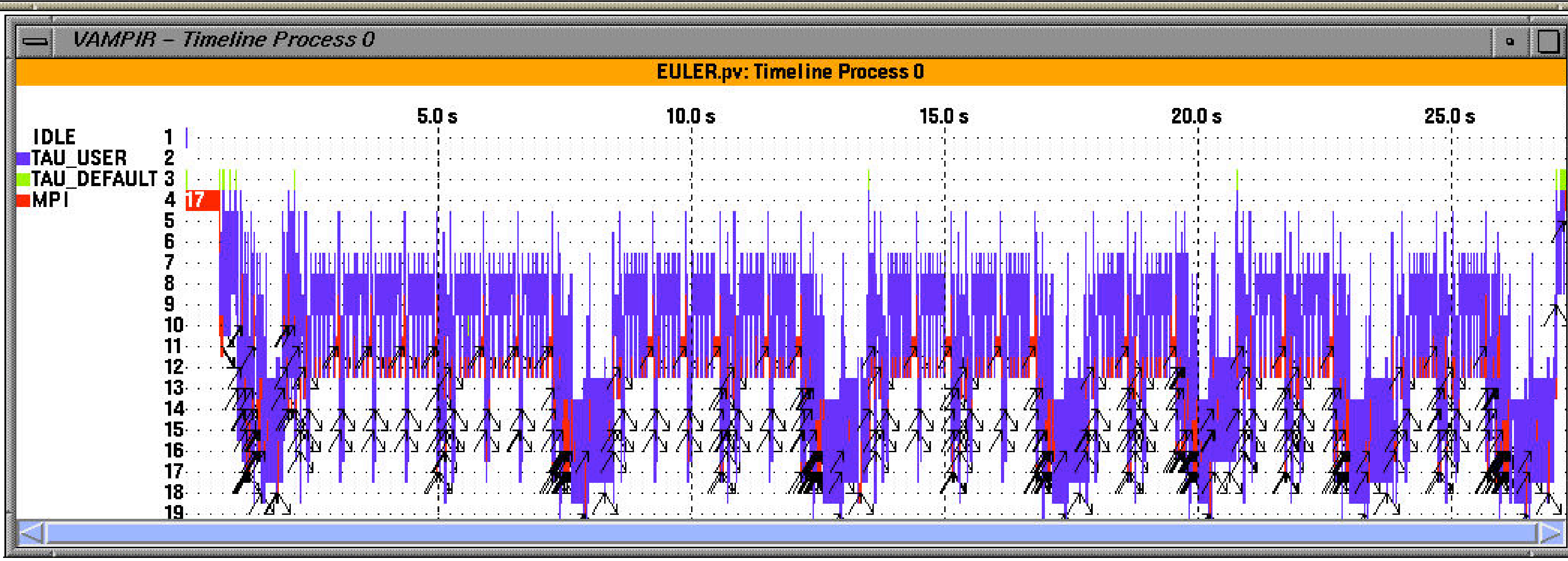


SAMRAI Euler Profile

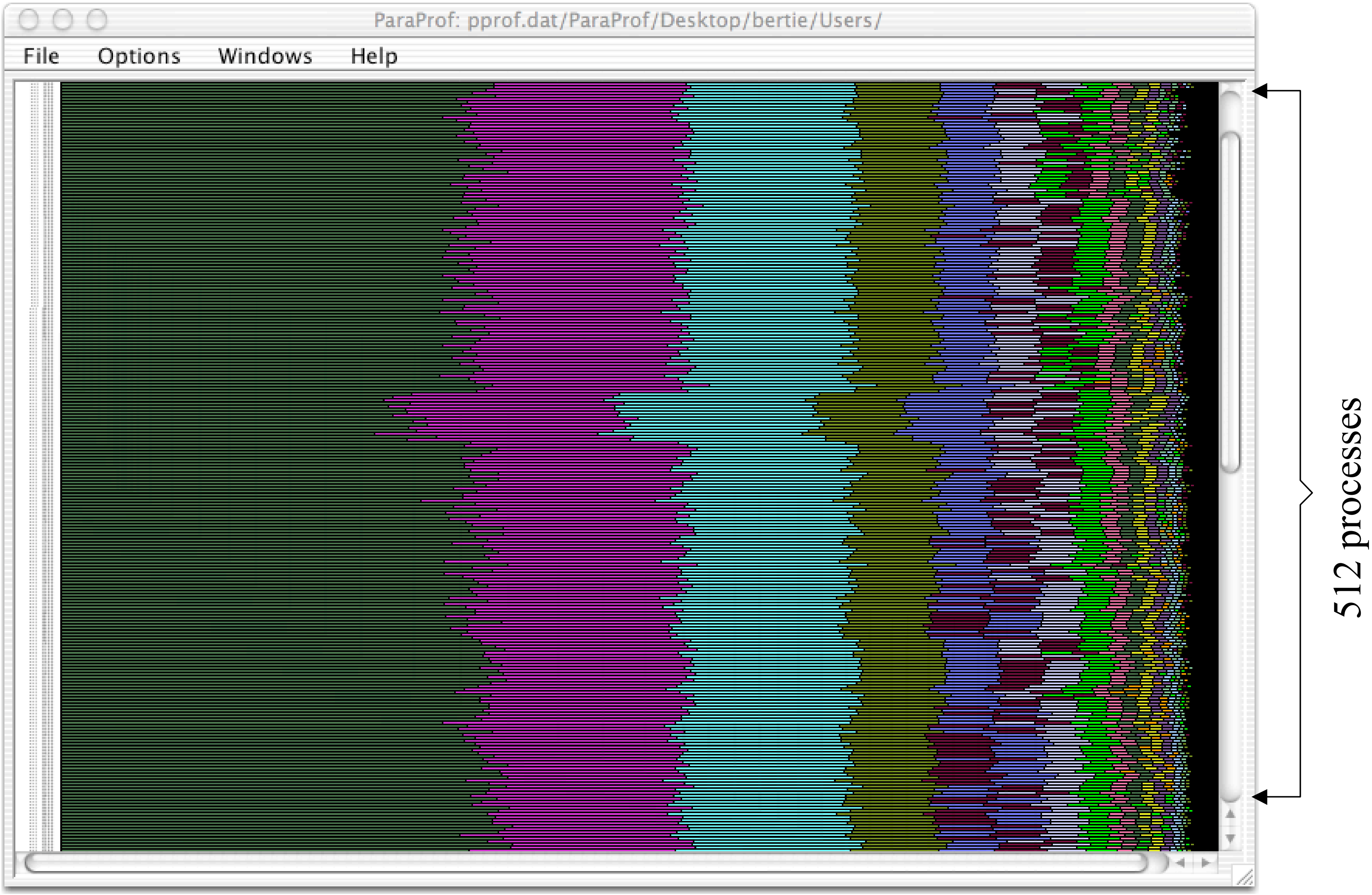


SAMRAI Euler Trace

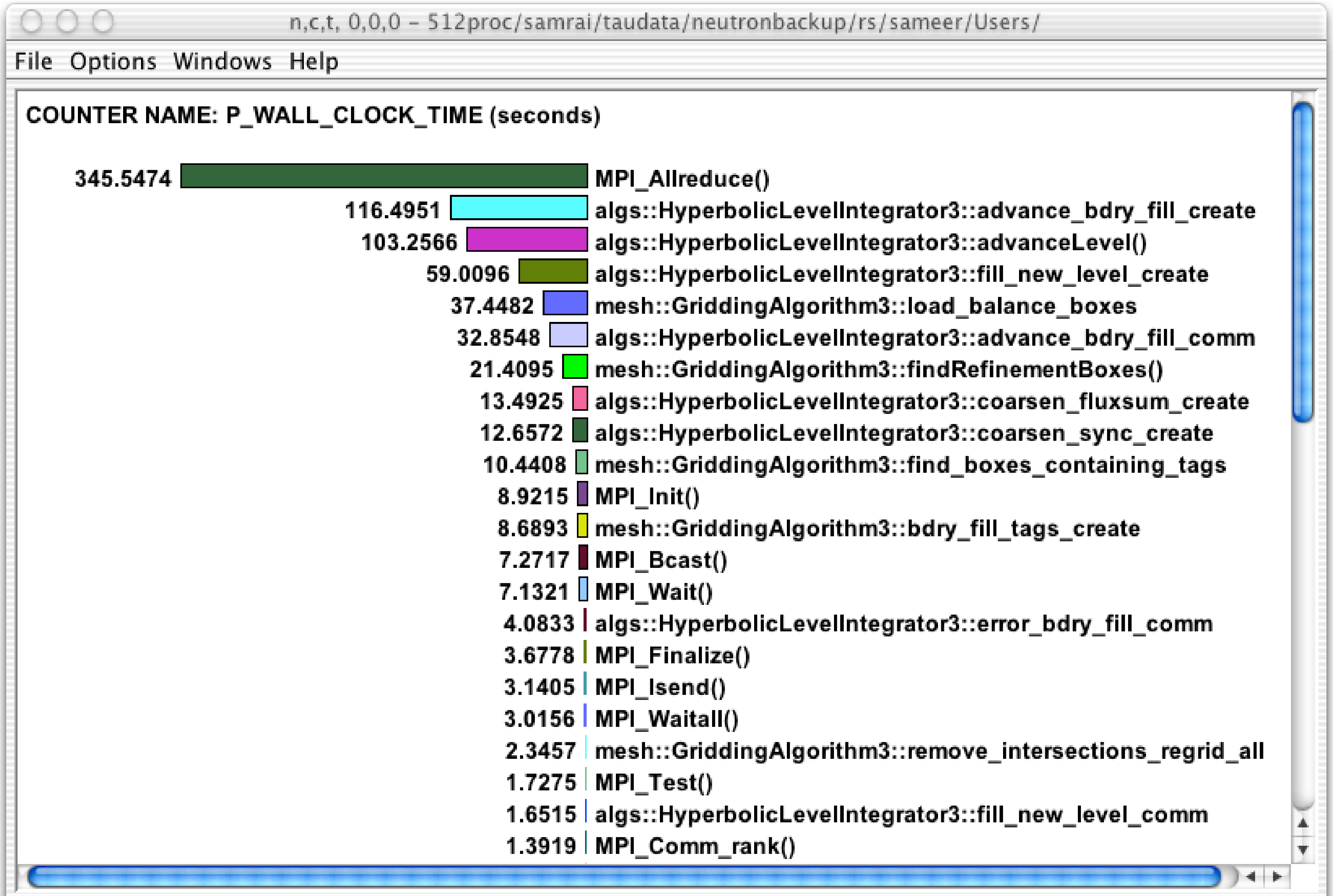




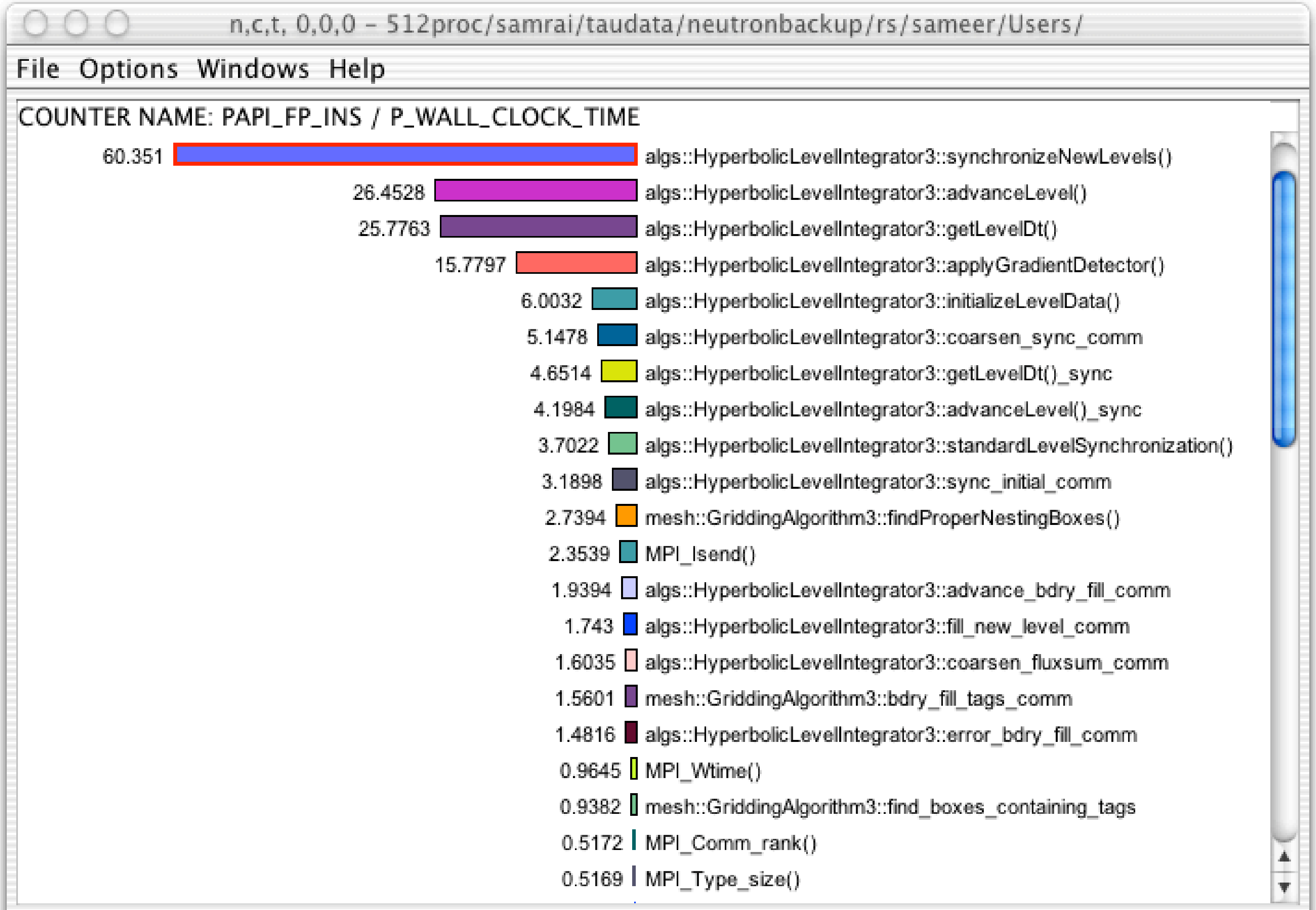
Full Profile Window (512 Processors)



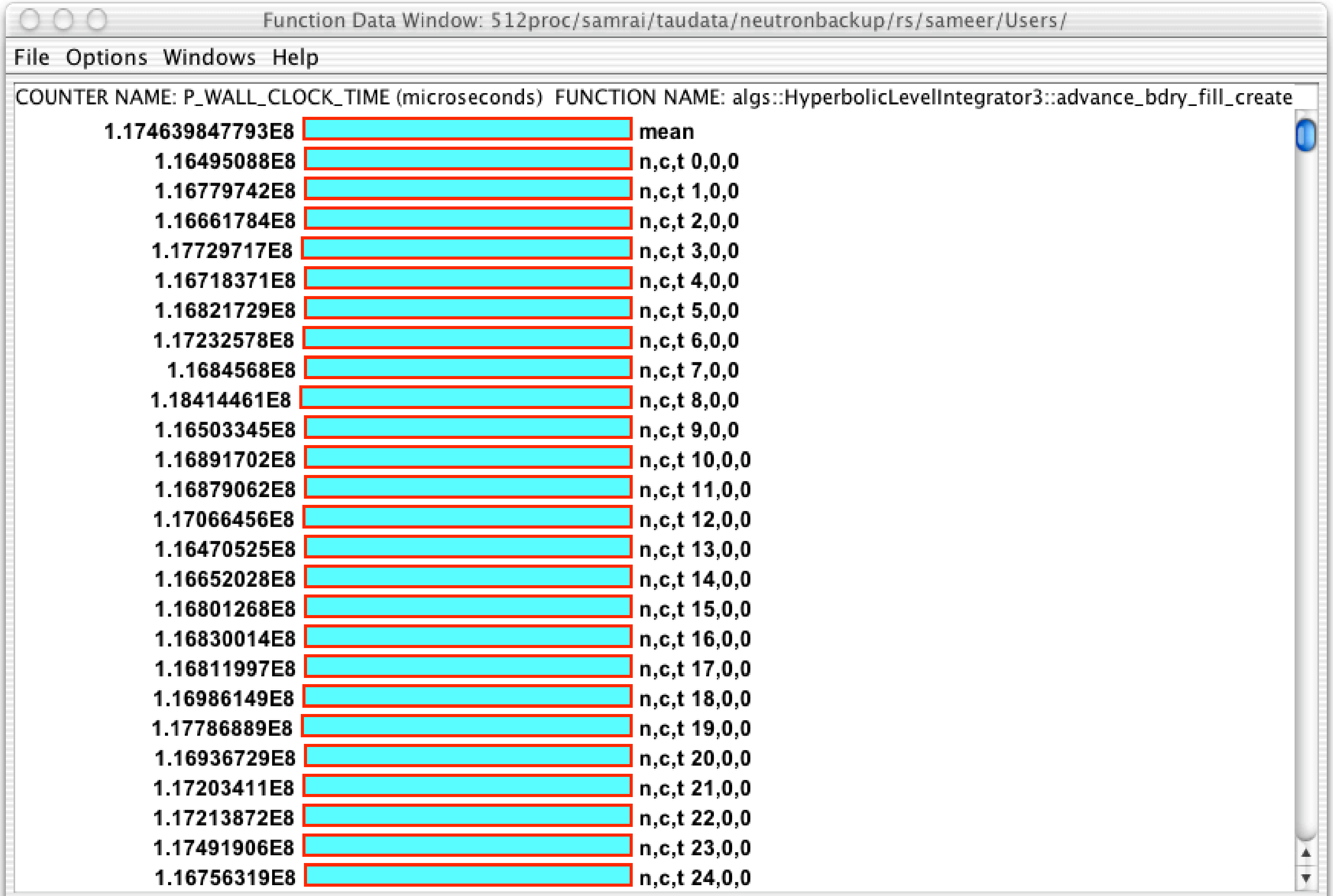
Node / Context / Thread Profile Window



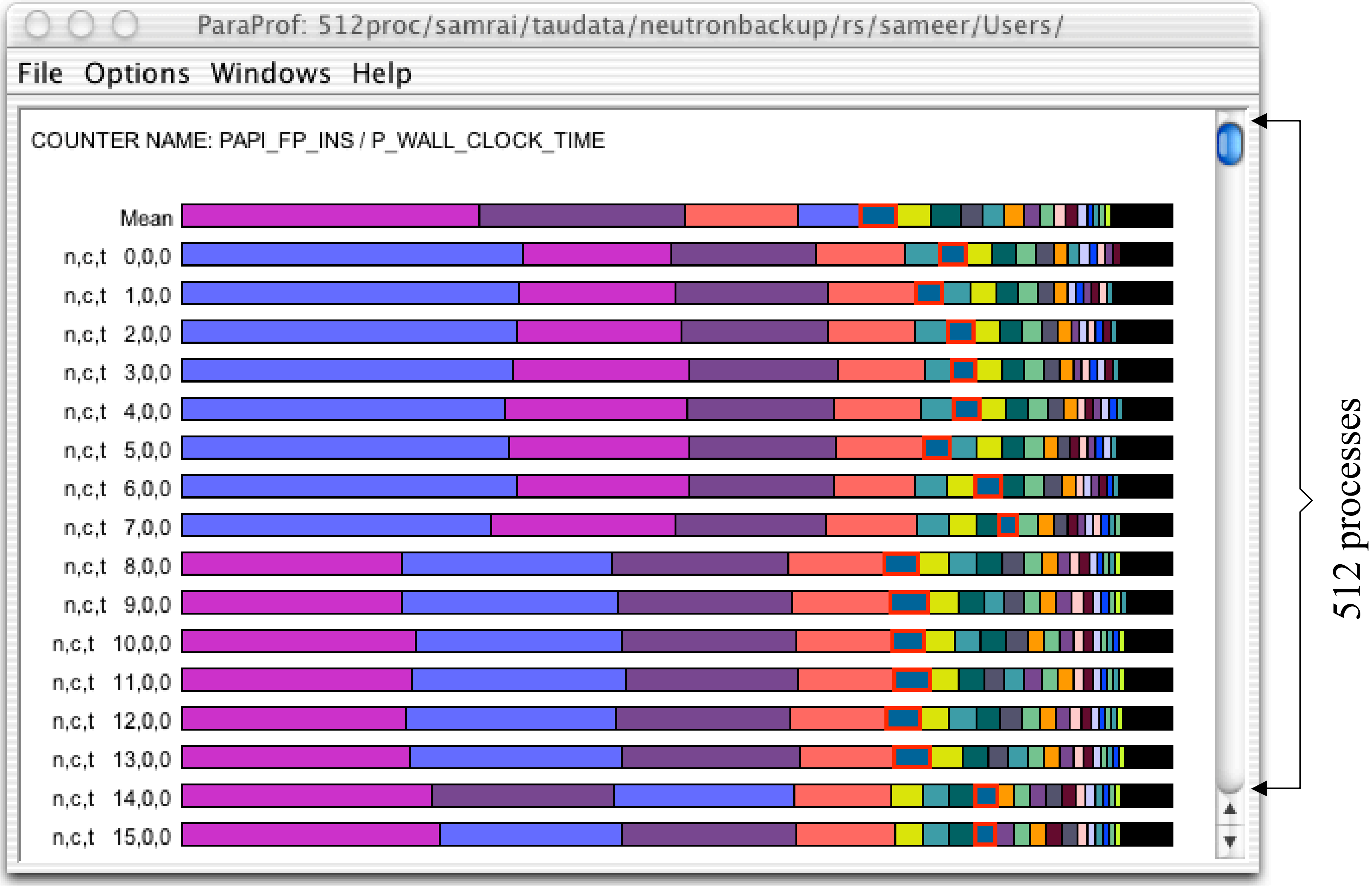
Derived Metrics



Paraprof Profile Browser Routine Window



Full Profile Window (Metric-specific)



Mixed-mode Parallel Programs (OpenMPI + MPI)

- Portable mixed-mode parallel programming
  - Multi-threaded shared memory programming
  - Inter-node message passing
- Performance measurement
  - Access to RTS and communication events
  - Associate communication and application events
- 2D Stommel model of ocean circulation
  - OpenMP for shared memory parallel programming
  - MPI for cross-box message-based parallelism
  - Jacobi iteration, 5-point stencil
  - Timothy Kaiser (San Diego Supercomputing Center)

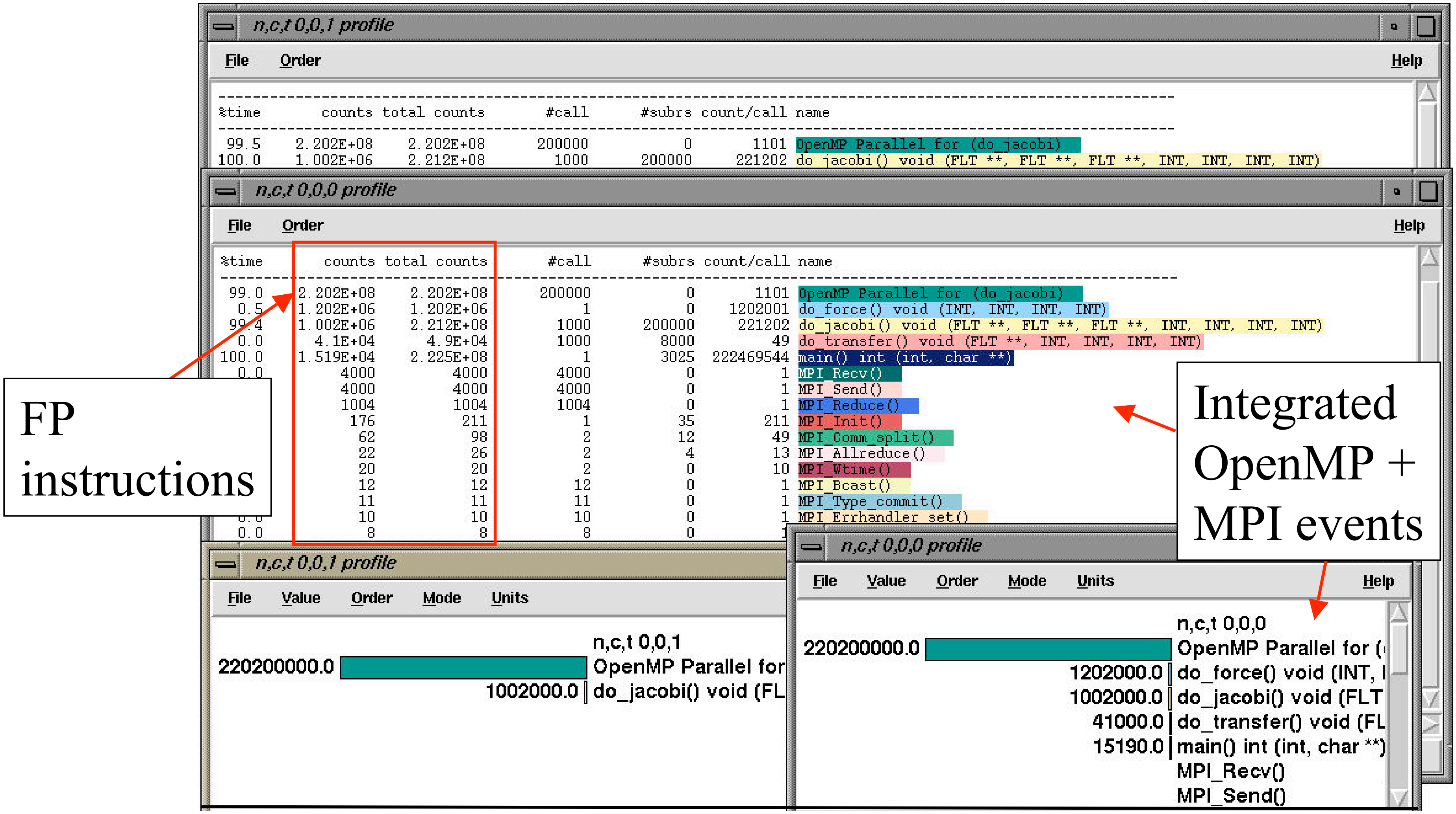
Stommel Instrumentation

- OpenMP directive instrumentation

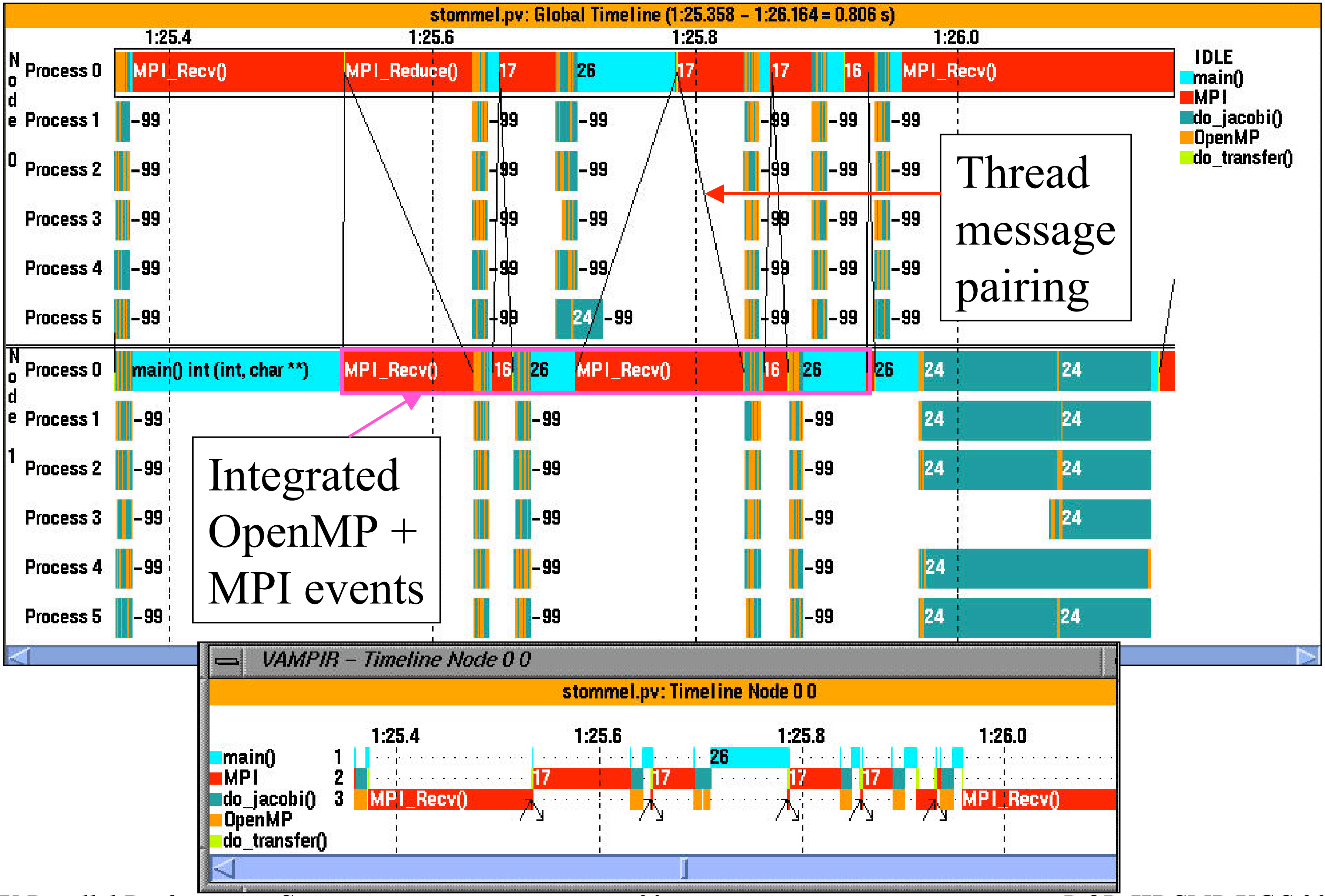
```
pomp_for_enter(&omp_rd_2);
#line 252 "stommel.c"
#pragma omp for schedule(static) reduction(+: diff) private(j)
  firstprivate (a1,a2,a3,a4,a5) nowait
for( i=i1;i<=i2;i++) {
  for(j=j1;j<=j2;j++){
    new_psi[i][j]=a1*psi[i+1][j] + a2*psi[i-1][j] + a3*psi[i][j+1]
      + a4*psi[i][j-1] - a5*the_for[i][j];
    diff=diff+fabs(new_psi[i][j]-psi[i][j]);
  }
}
pomp_barrier_enter(&omp_rd_2);
#pragma omp barrier
pomp_barrier_exit(&omp_rd_2);
pomp_for_exit(&omp_rd_2);
#line 261 "stommel.c"
```

OpenMP + MPI Ocean Modeling (HW Profile)

```
% configure -papi=../packages/papi -openmp -c++=pgCC -cc=pgcc
  -mpiinc=../packages/mpich/include -mpilib=../packages/mpich/libo
```



OpenMP + MPI Ocean Modeling (Trace)



HYbrid Coordinate Ocean Model (HYCOM)

- Climate, weather and ocean (CWO) based application
  - Primitive equation ocean circulation model (MICOM)
  - Improved vertical coordinate scheme that remains isopycnic in the open, stratified ocean
- Transitions smoothly
  - To *z-level* coordinates in the weakly-stratified upper-ocean mixed layer, to sigma coordinates in shallow water conditions and back to *z-level* coordinates in very shallow water conditions
- User has control over the model domain
  - Generating the forcing field
- Dr. Avi Purkayastha

Getting started with HYCOM

- For generation of the serial or parallel executables
  - Make.com => ../config/$(ARCH)_$(TYPE)
  - Contains all Makefile macro definitions for the preprocessor, fortran, C, parser, and instrumentor compile and link options and associated libraries
- Serial runs
  - 2.00 degree Atlantic Ocean regional grid was used for input, without any change to domain or resolution
- Parallel runs
  - Global ocean (GLBA0.24) was the domain
  - Performance analysis was carried out only for the MPI version of HYCOM

Gprof Profile Data on HYCOM

- Gprof profile data conclusions
  - Prime candidates for optimization
    - ***momtum***
    - ***tsadvc***
    - ***mx****
    - ***cnuity***
  - Mathematical functions also contribute significantly
    - ***atan*** and ***sqrt***

TAU Profile Analysis of HYCOM

- Exclusive time shows the relative largest time consuming functions
  - Including, surprisingly, MPI_Waitall
- Exclusive time spent can be used as an indicator for measuring efficiency of these functions
  - For example, obtaining MFLOP rates

Tau profile data excerpt with the highest time-consuming functions

%Time	Exclusive msec	Inclusive total msec	#call	#subrtns	Inclusive usec/call	Name
100.0	8:25.60	49:01.37	1	54686.1	2941369933	HYCOM
34.3	16:47.70	16:47.70	54191.4	0	18595	MPI_Waitall
32.1	10:41.98:6	15:43.07	192	113178	4911808	TSADVC
19.7	:11.67	9:40.60	192	31488	3023979	MOMTUM
9.3	2:12.77	4:33.82	192	20601.6	1426132	CNUITY

TAU Profile Analysis of HYCOM

- Low and high end of the time variance spent by the MPI_Waitall call in some of the processors
- Additional investigation is then required from the tracefiles for better understanding overall communication model

Tau profile data excerpt highlighting load imbalance on MPI_Waitall

Proc #	%Time	Exclusive msec	Inclusive total msec	#call	Inclusive usec/call
13	4.7	2:17.33	2:17.33	54182	2535
14	5.6	2:43.63	2:43.63	54182	3020
18	80.6	39:31.07	39:31.07	54229	43723
23	75.3	36:54.28	36:54.28	54182	40867
24	81.1	39:45.41	39:45.41	54229	43988

PAPI profile analysis of HYCOM

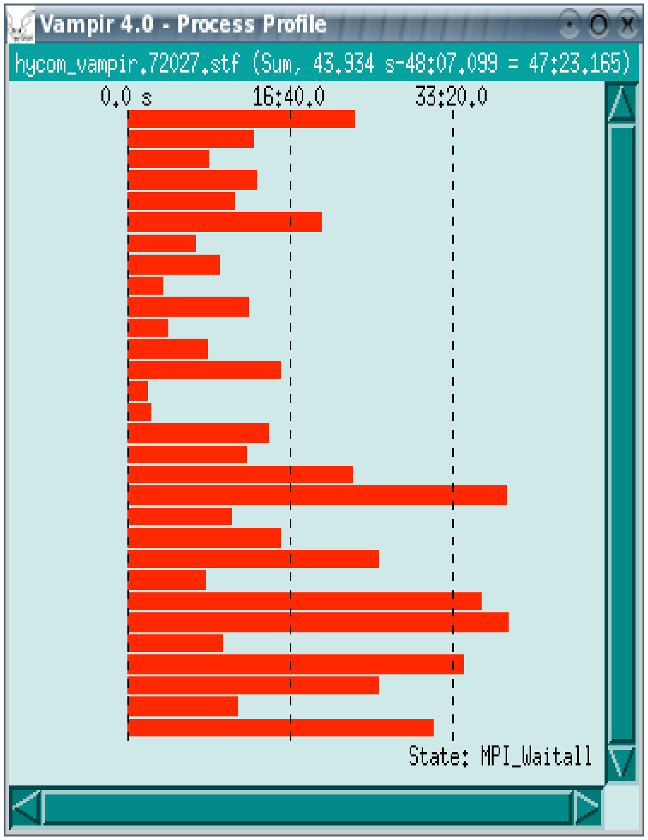
- PAPI profiling (obtained with Tau) exclusive operation count can show performance of individual functions
  - TSADVC
    - 2.07e+11/10:41.98(=642s) = 323 Mflops (~6% of peak)

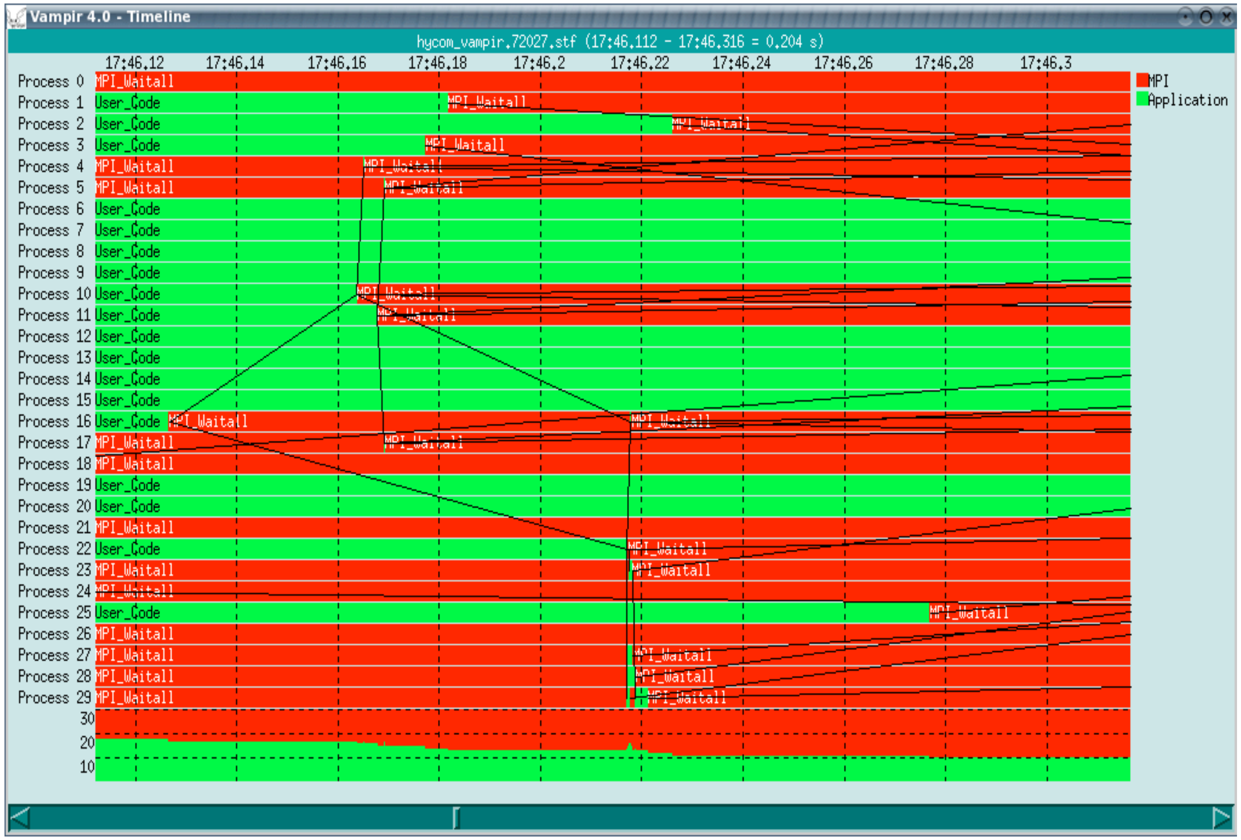
Papi profile data excerpt with the highest time-consuming functions

%Time	Exclusive counts	Inclusive total counts	#call	#subrtns	Inclusive usec/call	Name
100.0	1.56e+11	5.25e+11	1	54686.1	52543238273	HYCOM
39.3	2.07e+11	2.08e+11	192	113178	1085499128	TSADVC
22.2	1.15e+11	1.16e+11	192	31488	606724005	MOMTUM
8.3	4.32e+10	4.37e+10	192	20601.6	227820049	CNUITY
0.5	2.57e+09	2.57e+09	54191.4	0	47497	MPI_Waitall

Vampir Trace analysis of HYCOM

- ***MPI_Waitall*** time per process shows the wide disparity
  - Time spent for blocking call to return => load imbalance
- Small snapshot of global timeline indicates ***MPI_Waitall*** waiting on non-blocking receive operations to complete



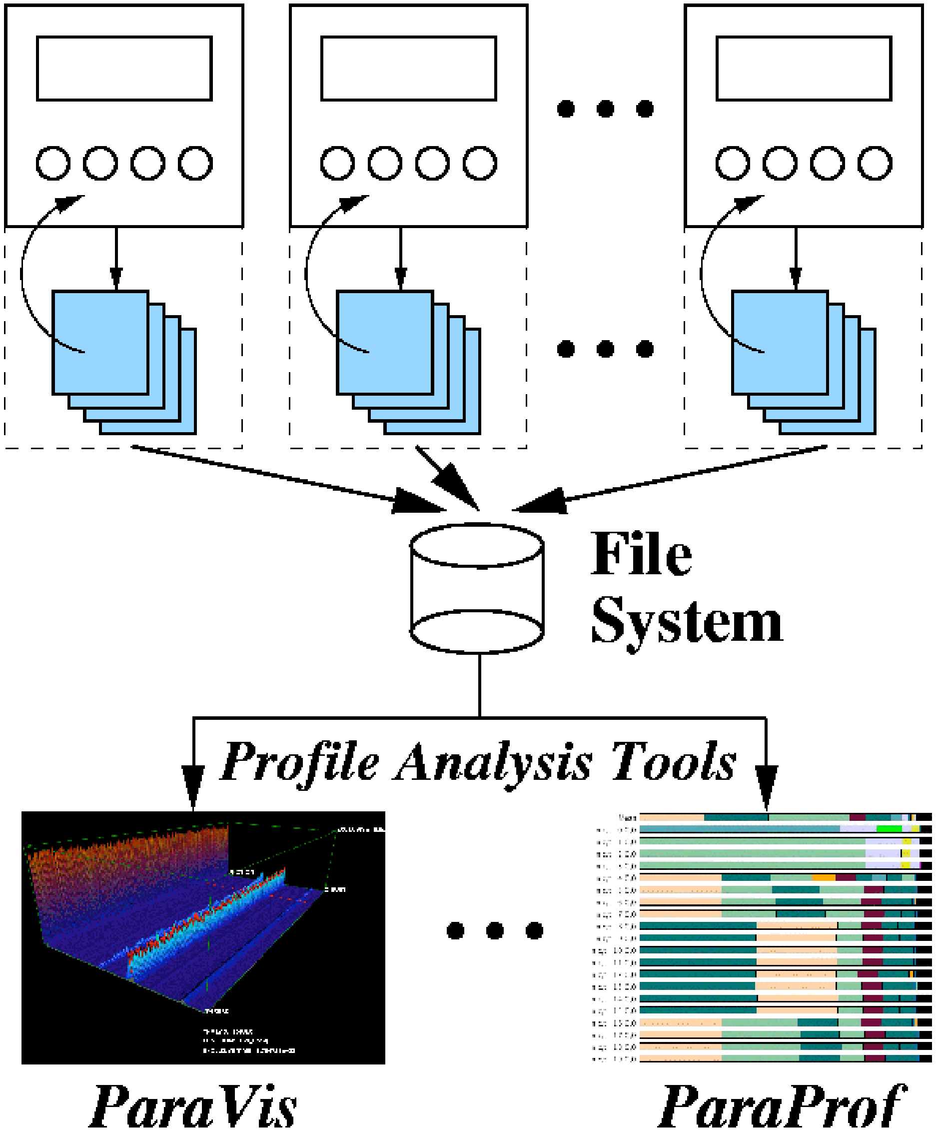


Vampir Trace analysis of HYCOM

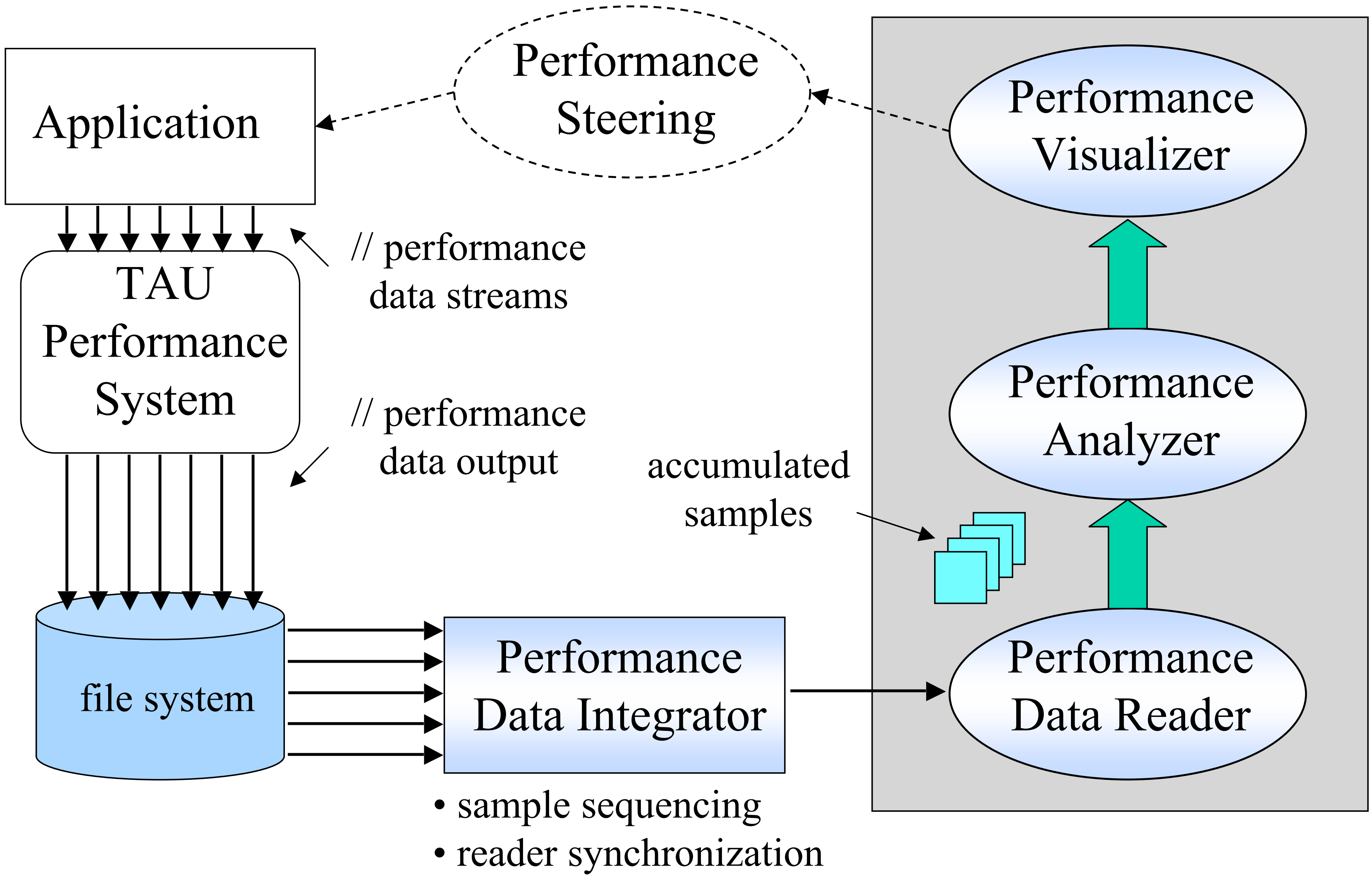
- This kind of load imbalance is a generic problem for structured-grid ocean models caused by variations in amount of ocean per tile
- HYCOM avoids all calculations over land, so the load imbalance leads to long intervals spent in MPI_Waitall on processors that "own" little ocean
- A different MPI strategy is perhaps necessary to reduce the MPI overhead on those processors with the most computational overhead which is the main contributing factors for those processors waiting for non-blocking receive operations

Online Profile Measurement and Analysis in TAU

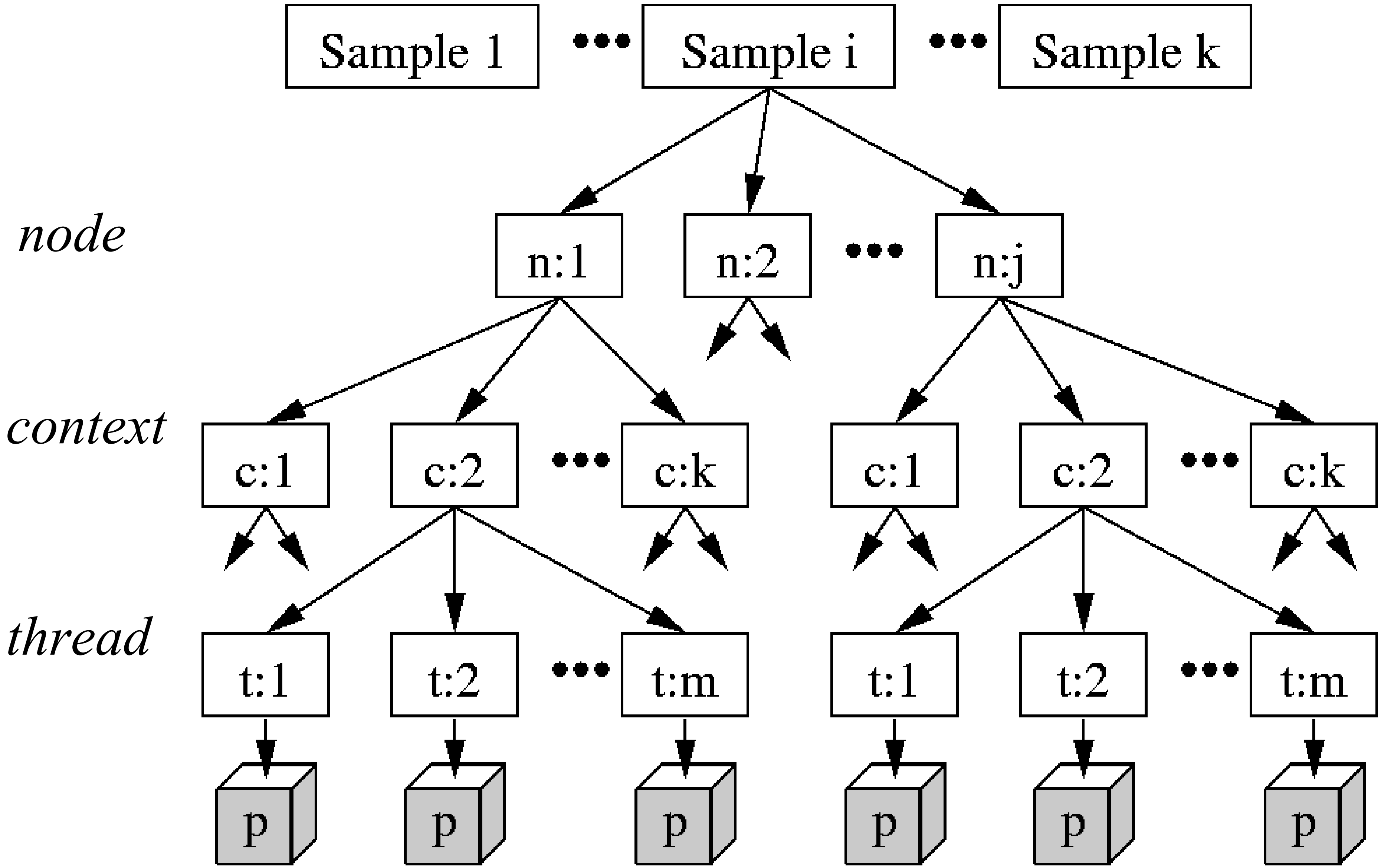
- Standard TAU profiling
  - Per node/context/thread
- Profile "dump" routine
  - Context-level
  - Profile file per each thread in context
  - Appends to profile file
  - Selective event dumping
- Analysis tools access files through shared file system
- Application-level profile "access" routine



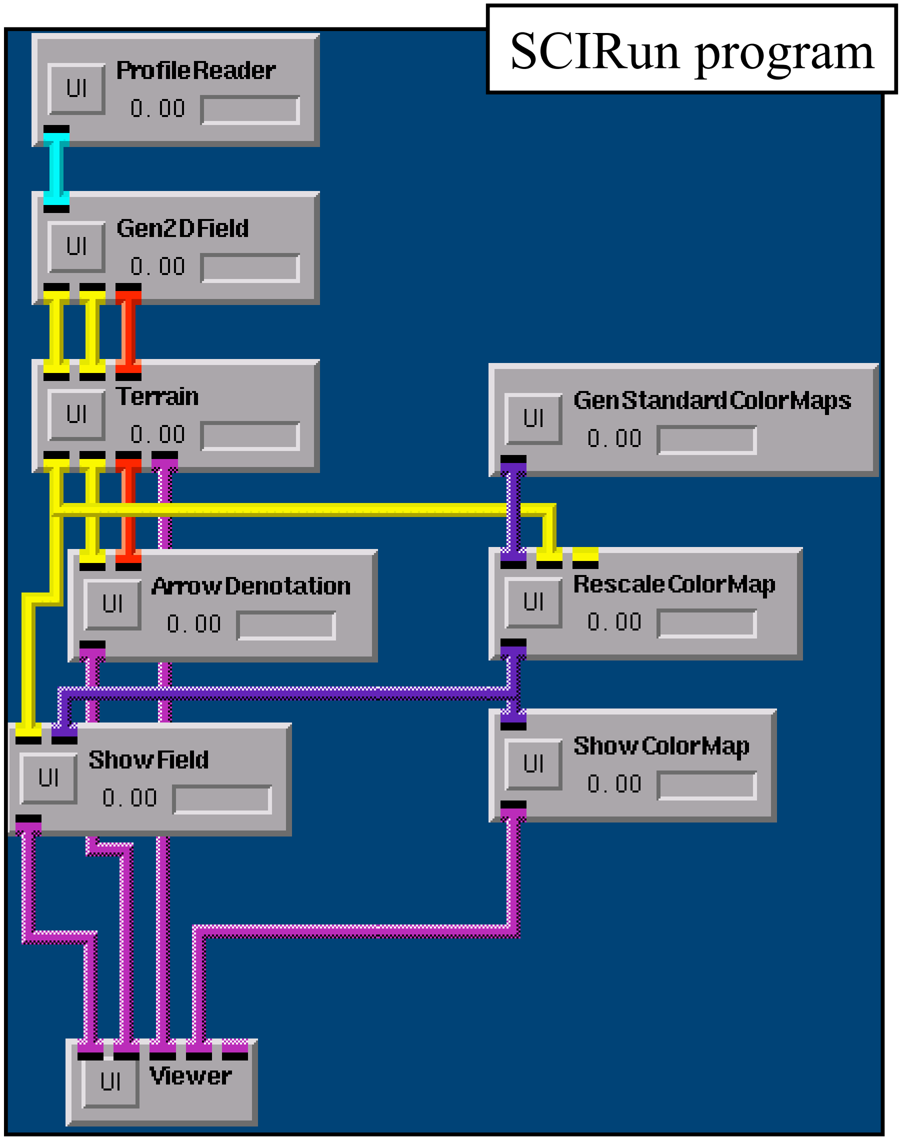
Online Performance Analysis and Visualization

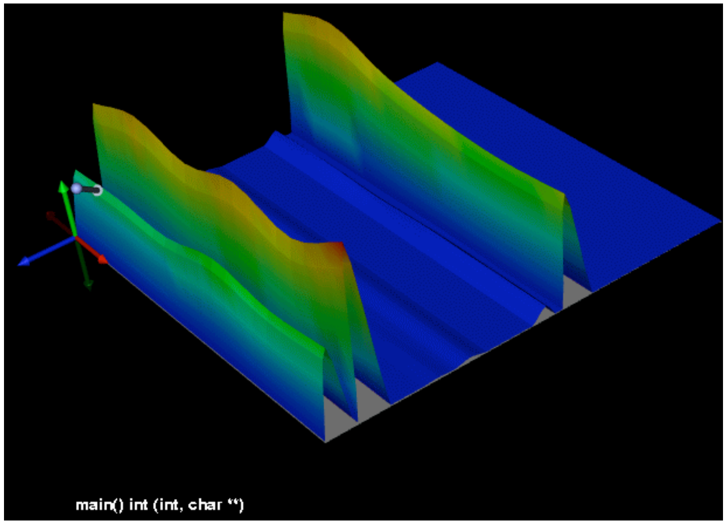


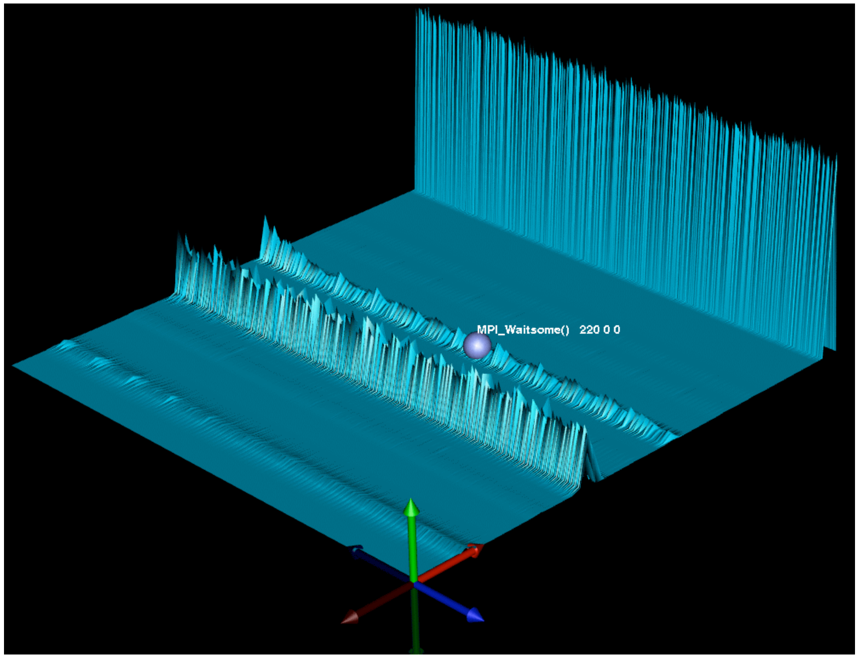
Profile Sample Data Structure



Performance Analysis/Visualization in SCIRun

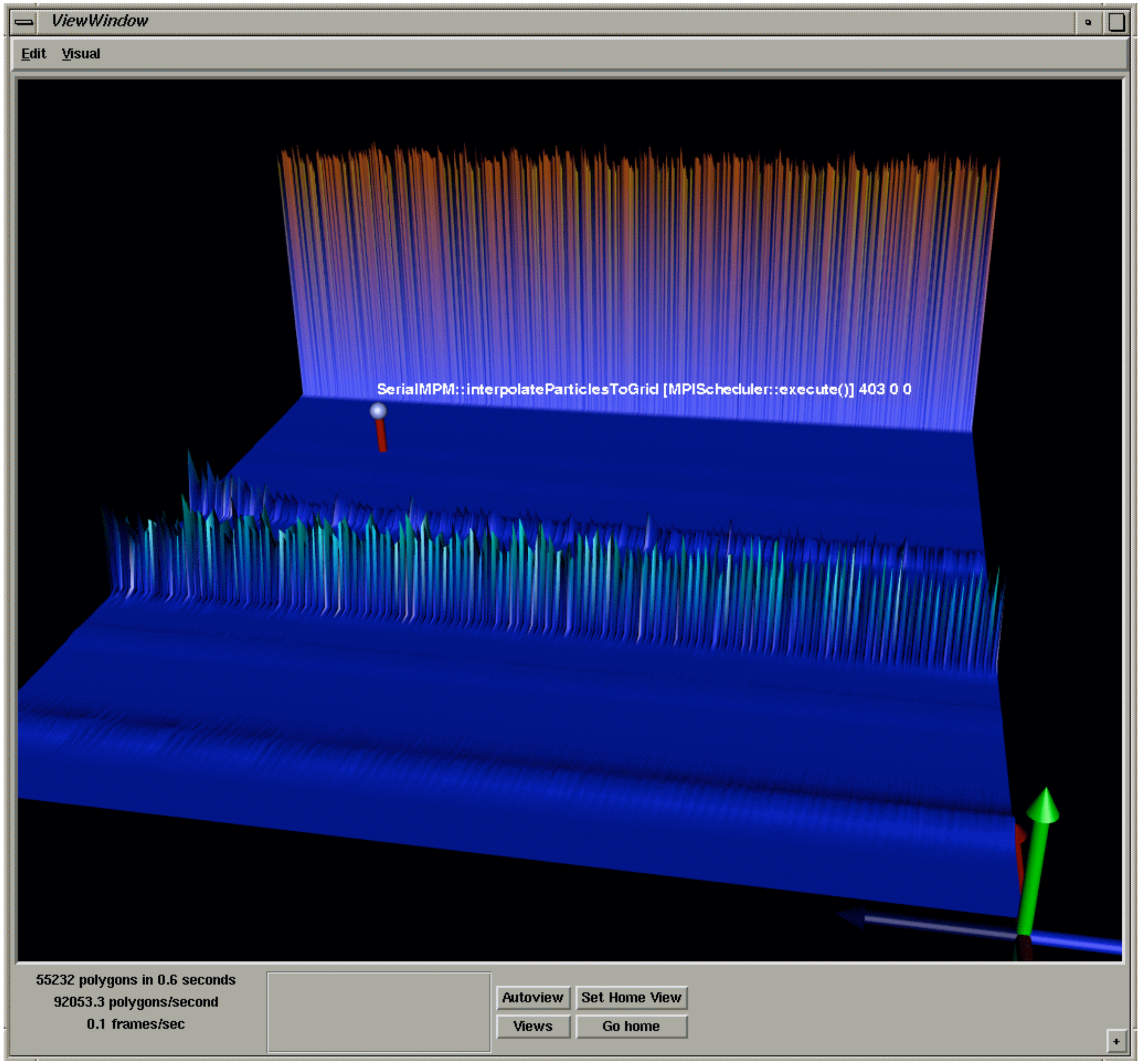




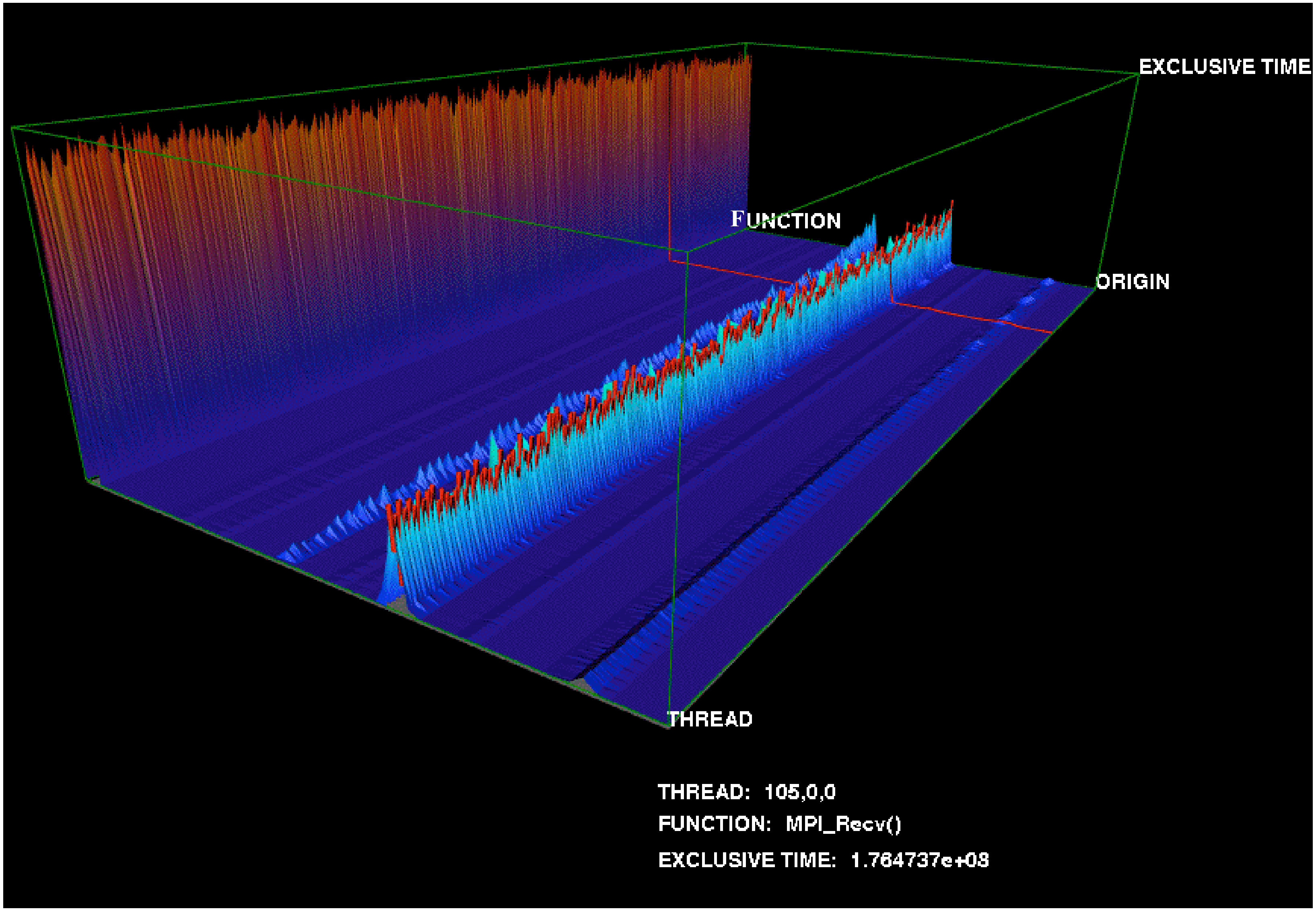


Uintah Computational Framework (UCF)

- University of Utah
- UCF analysis
  - Scheduling
  - MPI library
  - Components
- 500 processes
- Use for online and offline visualization
- Apply SCIRun steering

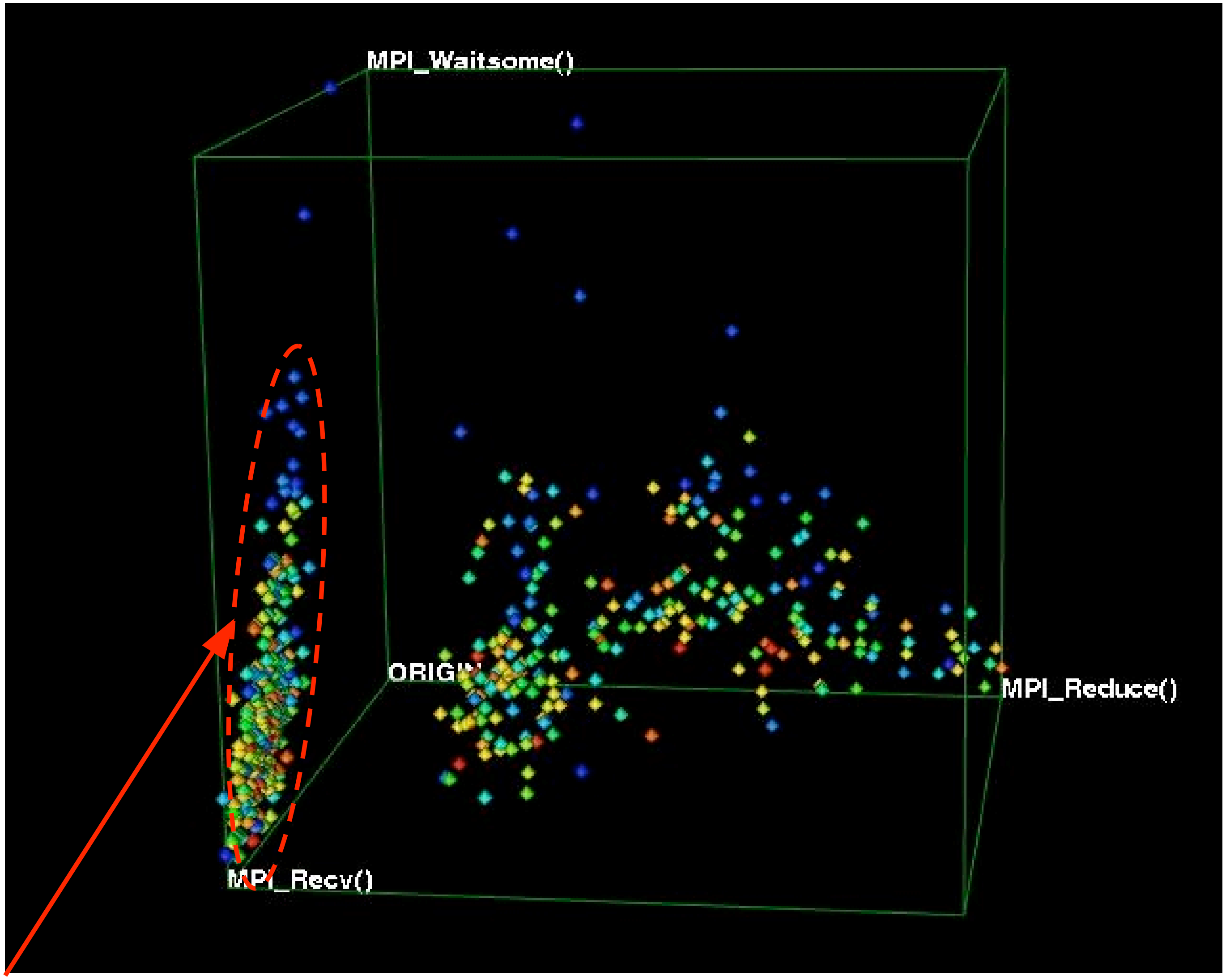


"Terrain" Performance Visualization



Scatterplot Displays

- Each point coordinate determined by three values:
  - *MPI_Reduce*
  - *MPI_Recv*
  - *MPI_Waitsome*
- Min/Max value range
- Effective for cluster analysis

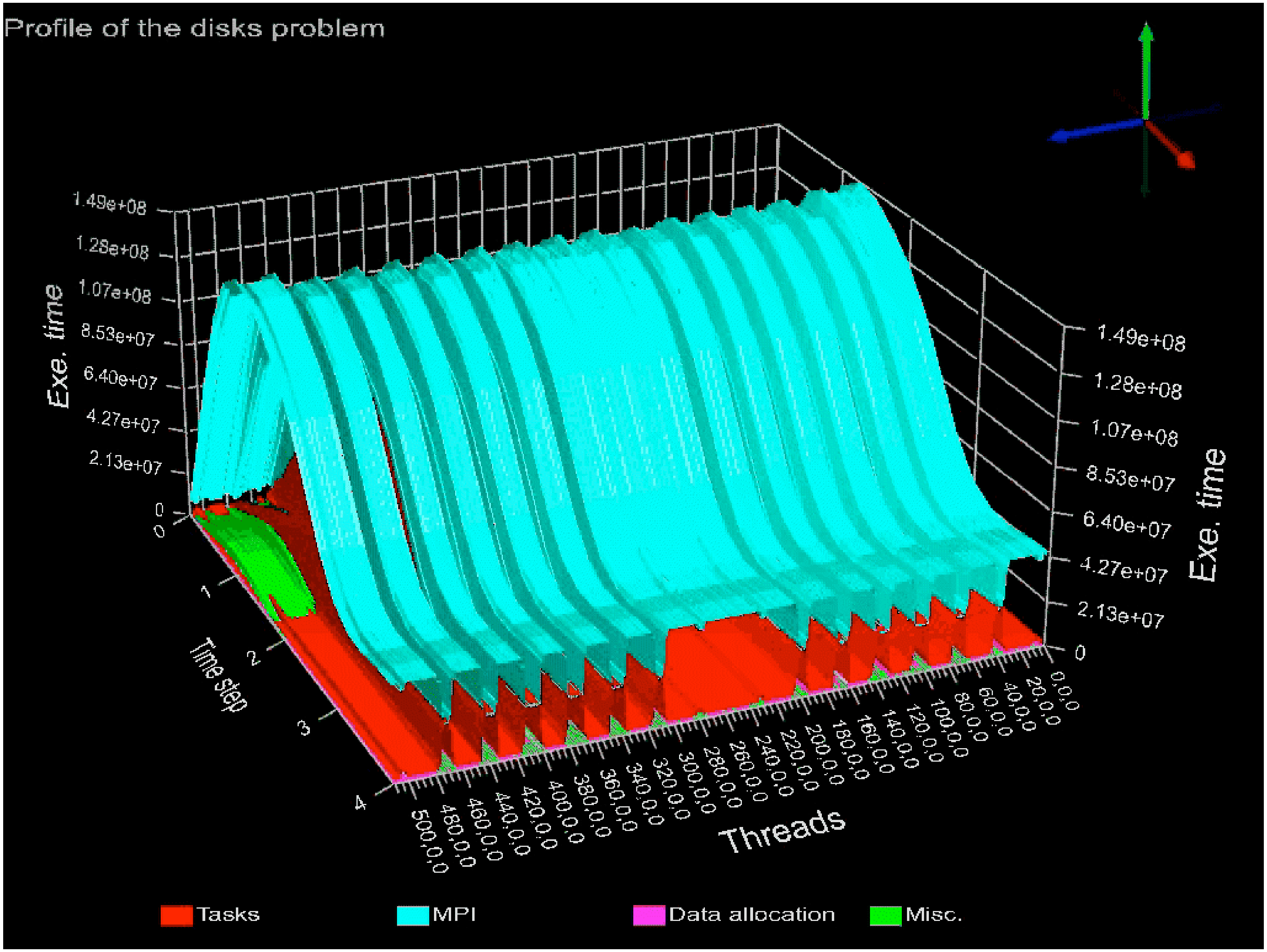


  - Relation between *MPI_Recv* and *MPI_Waitsome*

Online Unitah Performance Profiling

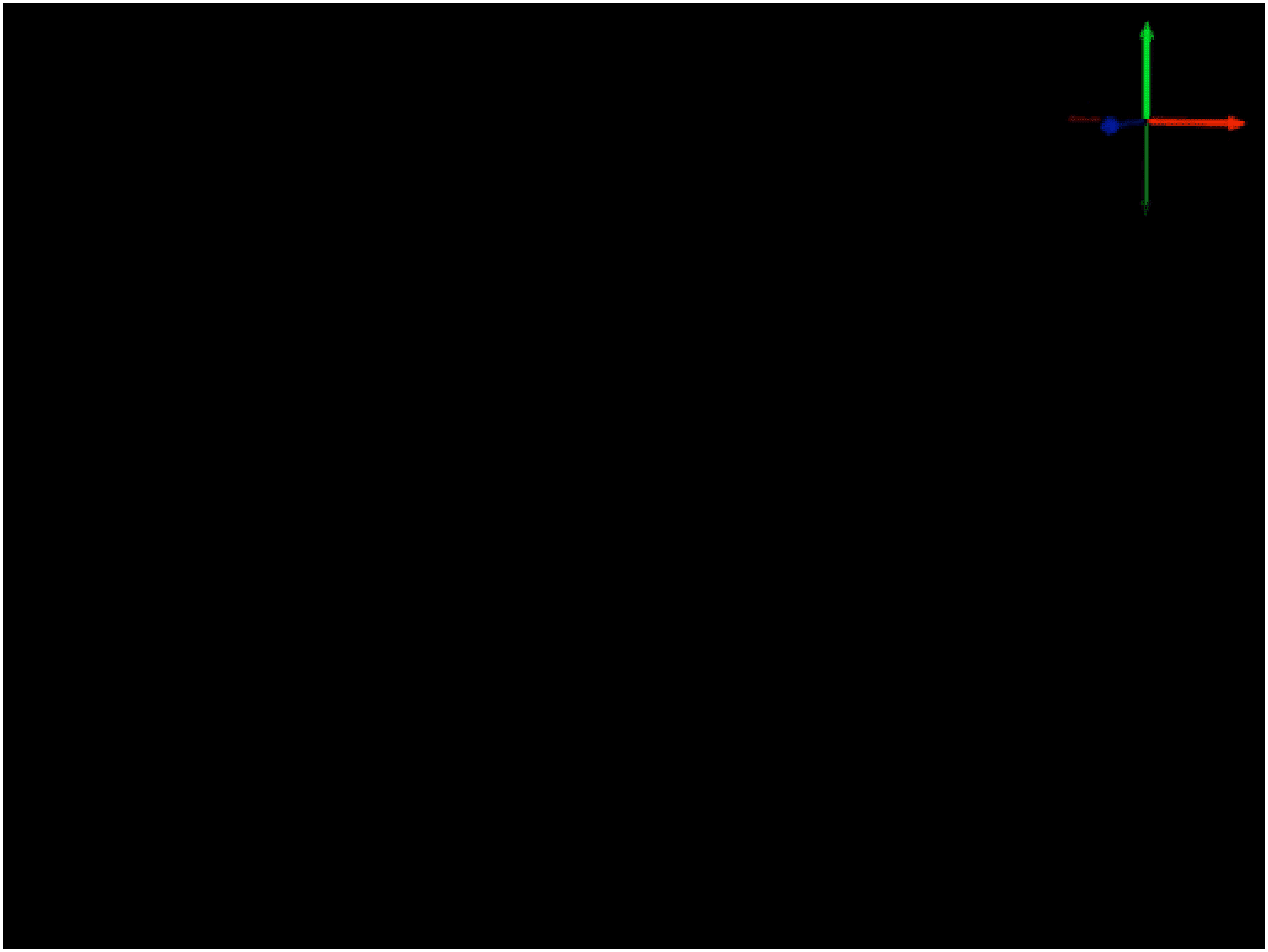
- Demonstration of online profiling capability
- Colliding elastic disks
  - Test material point method (MPM) code
  - Executed on 512 processors ASCI Blue Pacific at LLNL
- Example 1 (Terrain visualization)
  - Exclusive execution time across event groups
  - Multiple time steps
- Example 2 (Bargraph visualization)
  - MPI execution time and performance mapping
- Example 3 (Domain visualization)
  - Task time allocation to “patches”

Example 1 (Event Groups)



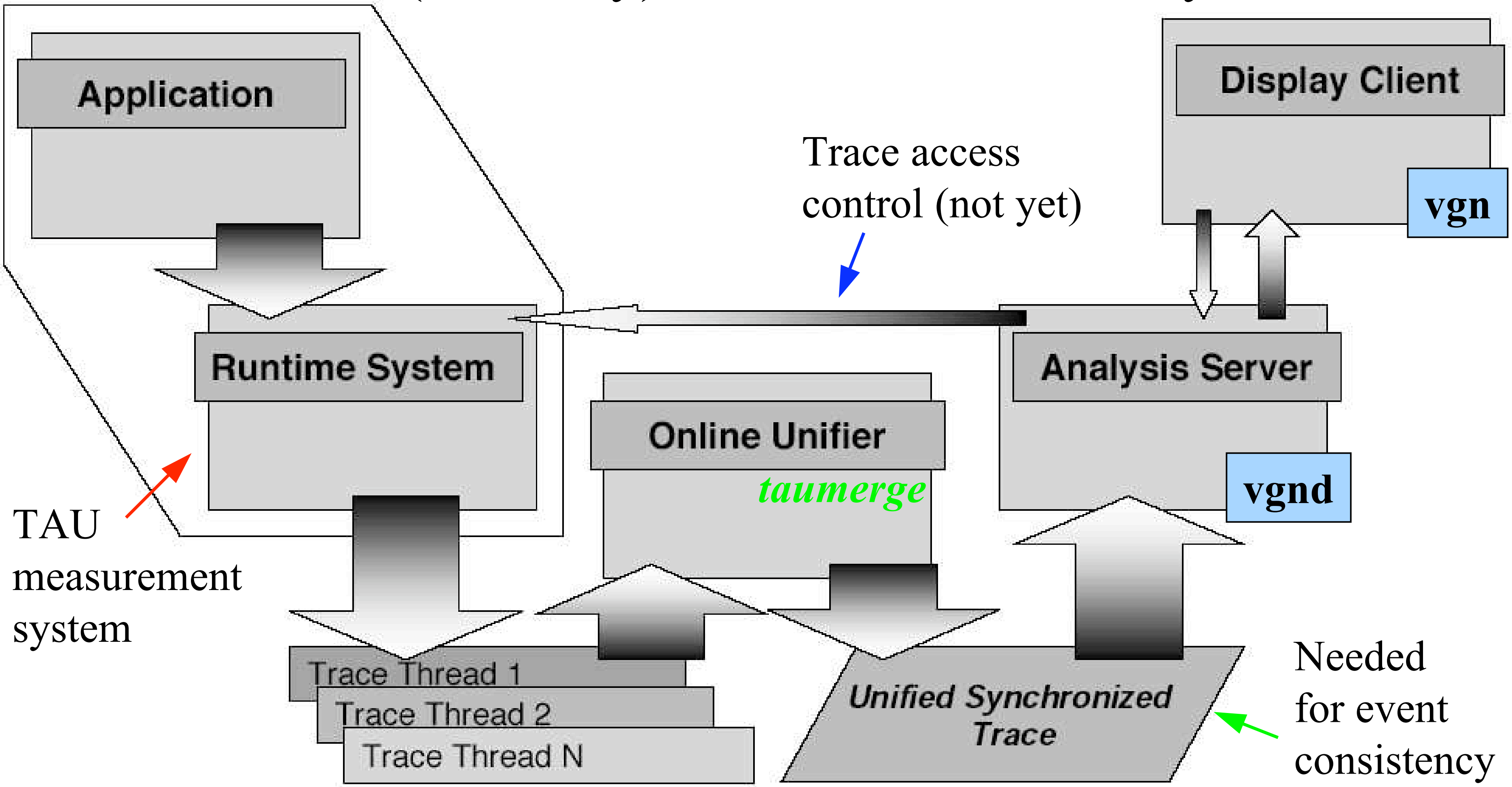
Example 2 (MPI Performance)

Example 3 (Domain-Specific Visualization)



Online Trace Analysis with TAU and VNG

- TAU measurement of application to generate traces
- Write traces (currently) to NFS files and unify



Integrated Performance Evaluation Environment