Enabling HPC software productivity with the TAU performance system Jean-Baptiste BESNARD <jbbesnard@paratools.fr>

PDC Summer School August 2023, KTH, Stockholm, Sweden.



Instrumentation

Various means of capturing program's state



Direct Observation

Direct Performance Observation

Execution *actions* exposed as *events*

- In general, actions reflect some execution state
 - presence at a code location or change in data
 - occurrence in parallelism context (thread of execution)
- Events encode actions for observation

Observation is *direct*

- Direct instrumentation of program code (probes)
- Instrumentation invokes performance measurement
- Event measurement = performance data + context

Performance experiment

• Actual events + performance measurements

Instrumentation



Measurement code is inserted such that every event of interest is captured directly

Can be done in various ways

Advantage:

Much more detailed information

Disadvantage:

- Processing of source-code / executable necessary
- Large relative overheads for small functions

```
int main()
{
    int i;
    TAU_START(``main");
    for (i=0; i < 3; i++)
        foo(i);
    TAU_STOP(``main");
    return 0;
}
void foo(int i)
{
    TAU_START(``foo");
    if (i > 0)
        foo(i - 1);
    TAU_STOP(``foo");
}
```



Three Instrumentation Techniques for Wrapping External Libraries

Pre-processor based substitution by re-defining a call (e.g., read) Preloading a library at runtime Linker based substitution

Preprocessor based substitution

Pre-processor based substitution by re-defining a call

Compiler replaces read() with tau_read() in the body of the source code Advantages:

- Simple to instrument
 - Preprocessor based replacement
 - A header file redefines the calls
 - No special linker or runtime flags required

Disadvantages

- Only works for C & C++ for replacing calls in the body of the code.

Incomplete instrumentation: fails to capture calls in uninstrumented libraries (e.g., libhdf5.a)

Preloading a wrapper library

Preloading a library at runtime

- Tool defines read(), gets address of global read() symbol (dlsym), internally calls timing calls around call to global read
- *tau_exec* tool uses this mechanism to intercept library calls Advantages
 - Drop-in replacement library implemented using LD_PRELOAD environment variable under Linux, Cray CNL, IBM BG/P CNK, Solaris...
 - No need to re-compile or re-link the application source code

Disadvantages

- Only works with dynamic executables. Default compilation mode under Cray XE6 and IBM BG/ P is to use static executables
- Not all operating systems support preloading of dynamic shared objects (DSOs)





Linker based substitution

Linker based substitution

- Advantages
 - Tool can intercept all references to a given call
 - Works with static as well as dynamic executables
 - the tool wrapper library

Disadvantages

- arguments. It is better to store these arguments in a file and pass the file to the linker
- Approach does not work with un-instrumented binaries



• Wrapper library defines wrap read which calls real read and linker is passed -WI,-wrap, read

No need to recompile the application source code, just re-link the application objects and libraries with

Wrapping an entire library can lengthen the linker command line with multiple –WI,-wrap,<func>

Indirect Performance Observation

Program code instrumentation is not used Performance is observed indirectly

- Execution is interrupted
 - can be triggered by different events
- •Execution state is queried (sampled)
 - different performance data measured
- Event-based sampling (EBS)
- Performance attribution is inferred
 - Determined by execution context (state)
 - Observation resolution determined by interrupt period
 - Performance data associated with context for period

Indirect Observation









foo(2)

Measurement

Addresses are mapped to routines using symbol table information

Statistical inference of program behavior

- Not very detailed information on highly volatile metrics
- Requires long-running applications

Works with unmodified executables

Time

```
int main()
  int i;
  for (i=0; i < 3; i++)
    foo(i);
  return 0;
void foo(int i)
  if (i > 0)
foo(i - 1);
```



Performance Data Measurement

Call START('potential') // code Call STOP('potential')

- Exact measurement
- Fine-grain control
- Calls inserted into code



- No code modification
- Minimal effort
- Relies on debug symbols (-g)



Profiling and Tracing

TAU: ParaProf: node 0 - fun3d_d19.ppk		
Metric: TIME Value: Exclusive Units: seconds		
Units: seconds 43.649 36.231 27.473 19.688 16.273 13.381 10.059 9.954 8.04 7.319 3.78 3.562 3.384 1.309 1.086 0.977 0.668 0.667 0.624 0.586 0.565 0.524	Loop: POINT_SOLVER::POINT_SOLVE_5 [{point_solver.F90} {2757,5}-{2917,19}] Loop: JACOBIAN_VISCOUS::EDGEJP [{jacobian_viscous.f90} {440,7}-{1584,22}] Loop: FLUX_PERFGAS::ROE_FLUX [[flux_perfgas.f90] {397,5}-{597,20}] FILL_JACOBIANS::FILL_JACOBIAN [[fill_jacobians.f90} {19,3}-{341,30}] PUNS3D_IO_C2N::PUNS3D_READ_VGRID_C2N_SM [{puns3d_io_c2n.f90} {1641,14}] GRADIENT_DRIVER::GRAD_VARIABLE [{gradient_driver.f90} {110,3}-{508,30}] MPI_Waitall0 MPI_Reduce() FLUX_PERFGAS::VISRHS_TET [[flux_perfgas.f90} {3023,14}] TURB_SPALART::SA_RESID [{turb_spalart.f90} {219,14}] FLUX_TURB::TURBULENT_GRADIENTS [{flux_turb.f90} {796,3}-{899,36}] TIMESTEP::DELTAT2 [{timestep.f90} {20,3}-{31,24}] POINT_SOLVER::POINT_SOLVE_1 [{point_solver.F90} {1083,3}-{1206,30}] TURB_SPALART::EDGE_ASSEMBLY_RES_CONV_DIFF [{turb_spalart.f90} {5048,14}] MPI_Bcast0 TURB_SPALART::EDGE_ASSEMBLY_RES_CONV_DIFF [{turb_spalart.f90} {5209,14}] PPARTY_MIXED_ELEMENT::EDGE_POINTER_DRIVER [{pparty_mixed_element.f90} {74,3}-{502,36}] UPDATE_MEAN::UPDATE_Q [{update_mean.F90} {723,3}-{803,25}] POINT_LU_DDQ::LU_5 [{point_lu_ddq.f90} {1566,3}-{1636,21}] SOLUTION::WREST_INTERLEAF [{solution.F90} {224,14}] DUAL_NUMBERINC::FMATRIX_TO_GRID [{dual_numbering.f90} {122,3}-{143,32}] GRID_METRICS::DUALMETRIC [{purb_spalart.f90} {757,14}]	
0.524	TURB_SPALART::SA_JACOB [{turb_spalart.f90} {757,14}]	
0.49	MPI_Barrier()	
0.479 0.41 0.371	DUAL_NUMBERING::FGRID_TO_MATRIX [{dual_numbering.f90} {94,3}-{115,32}] LMPI_APP::SINGLE_START_MATRIX_XFER [{Impi_app.F90} {7907,3}-{8132,41}]	
0.36	TURB_SPALART::JACOBIAN_SA [{turb_spalart.f90} {88,14}]	
0.352	TIMEACC::TIME_DIAG_NC [{timeacc.f90} {1067,3}-{1330,29}]	
0.336	MPI_Init()	
0.285	LMPI_APP::SINGLE_MATRIX_COMPLETE_XFER [{Impi_app.F90} {11520,3}-{11626,44}]	
0.283	PPARTY_METIS::MY_METIS [{pparty_metis.F90} {116,3}-{545,24}]	
0.275	PPARTY::NODE_CELL_CHOPPER [{pparty.f90} {41,3}-{453,33}]	
0.269	FLUX_UTIL::L2NORM [{flux_util.f90} {31,3}-{249,23}]	

Profiling

Profiling shows you **how much** (total) time was spent in each routine **Tracing** shows you **when** the events take place on a timeline



Tracing

How much data do you want?

Limited Loop Callpath Profile Profile Profile





Inclusive vs. Exclusive values

- Inclusive: Information of all subelements aggregated into single value
- Exclusive: Information cannot be subdivided further





Inclusive Measurements

• • •		140.1
Metric: TIME		
Value: Inclusiv	e	
Units: seconds	5	
221.305		TAU applicatio
221 304		NODET [{main
197 989		FLOW ITERAT
105 577		
105 560		DELAY STEAD
195.509	61.027	
	61.32	
	61.28	
	61.273	
	61.258	
	59.068	GCR_SOLVE::G
	57.635	GCR_SOLVE_U
	57.152	POINT_SOLVER
	56.882	UPDATE_MEAN
	54.402	RELAX_MEAN::
	53.103	LINEARSOLVE_
	52.867	UPDATE_MEAN
	52.866	FUN3D_RES_FL
	52.756	FLUX::RESIDUA
	52.747	POINT_SOLVER
	52.744	Loop: POINT_S
	36.232	JACOBIAN_VIS
	36.231	JACOBIAN_VIS
	36.231	Loop: JACOBIA
	27.474	FLUX_PERFGAS
	27.474	FLUX_PERFGAS
	27.473	Loop: FLUX_PE
	22.707	FLOW::INITIAL
	22.694	FLOW::INITIAL
	20.916	PPARTY_PREPR
	16.726	PPARTY_PREPR
	16.726	PUNS3D_IO_C
	16.657	PUNS3D_IO_C
	14.159	GRADIENT_DR
	13.852	UPDATE_TURB



TAU: ParaProf: node 0 - fun3d_d19.ppk

on .f90} {4,1}-{35,17}] E [{flow.F90} {1692,14}] OLVER [{flow.F90} {1845,14}] Y::RELAX [{relax_steady.f90} {30,3}-{307,22}] N::UPDATE_LINEAR_SYSTEM_MEAN [{update_mean.F90} {195,3}-{275,42}] N::UPDATE_JACOBIAN_DRIVER_MEAN [{update_mean.F90} {460,3}-{505,44}] N::UPDATE_JACOBIAN [{update_mean.F90} {513,3}-{588,32}] NS::FILL_JACOBIAN [{fill_jacobians.f90} {19,3}-{341,30}] GCR_SOLVER_QSET [{gcr_solve.f90} {47,3}-{415,32}] JTIL::GCR_PRECONDITIONER_QSET [{gcr_solve_util.f90} {40,3}-{131,40}] R::POINT_SOLVE [{point_solver.F90} {31,3}-{214,28}] N::UPDATE_RHS_MEAN [{update_mean.F90} {102,3}-{185,32}] :RELAX [{relax_mean.f90} {22,3}-{84,22}] _NODIVCHECK::NODIVCHECK_RELAX_Q [{linearsolve_nodivcheck.F90} {56,14}] N::RESIDUAL_S [{update_mean.F90} {42,3}-{94,27}] LOW::RES_FLOW [{fun3d_res_flow.f90} {27,3}-{279,25}] AL_COMPRESSIBLE [{flux.f90} {25,3}-{592,38}] R::POINT_SOLVE_5 [{point_solver.F90} {2700,3}-{2921,30}] SOLVER::POINT_SOLVE_5 [{point_solver.F90} {2757,5}-{2917,19}] COUS::VISCOUS_JACOBIAN [{jacobian_viscous.f90} {20,14}] COUS::EDGEJP [{jacobian_viscous.f90} {324,14}] AN_VISCOUS::EDGEJP [{jacobian_viscous.f90} {440,7}-{1584,22}] S::INVISCID_FLUX_DRIVER [{flux_perfgas.f90} {37,14}] S::ROE_FLUX [{flux_perfgas.f90} {236,14}] ERFGAS::ROE_FLUX [{flux_perfgas.f90} {397,5}-{597,20}] IZE_DATA [{flow.F90} {465,14}] .IZE_DATA2 [{flow.F90} {663,14}] ROCESSOR::PPARTY_PREPROCESS [{pparty_preprocessor.f90} {28,14}] ROCESSOR::PPARTY_READ_GRID [{pparty_preprocessor.f90} {735,14}] 2N::PUNS3D_READ_VGRID_C2N [{puns3d_io_c2n.f90} {1543,14}] 2N::PUNS3D_READ_VGRID_C2N_SM [{puns3d_io_c2n.f90} {1641,14}] RIVER::GRAD_VARIABLE [{gradient_driver.f90} {110,3}-{508,30}] S::UPDATE_RHS_TURB [{update_turb.f90} {742,3}-{845,32}]

Exclusive Time

	TAU: ParaProf:
Metric: TIME	
Value: Exclusive	
Units: seconds	
43.649	Loop: POINT SOLVER::
36.231	Loop: IACOBIAN VISCO
27.473	Loop: FLUX PERFGAS::
19.688	FILL IACOBIANS::FILL
16.273	PUNS3D IO C2N::PUNS
13.381	GRADIENT DRIVER :: GR
10.059	MPI Waitall()
9.954	MPI_Reduce()
8.04	FLUX_PERFGAS::VISRHS
7.319	TURB_SPALART::SA_RES
3.78	FLUX_TURB::TURBULEN
3.562	TIMESTEP::DELTAT2 [{t
3.384	POINT_SOLVER::POINT
1.309	TURB_SPALART::EDGE_
1.086	MPI_Bcast()
0.977	TURB_SPALART::EDGE_
0.668	PPARTY_MIXED_ELEMEN
0.66	UPDATE_MEAN::UPDAT
0.637	POINT_LU_DDQ::LU_5
0.624	SOLUTION::WREST_INT
0.586	DUAL_NUMBERING::FM
0.565	GRID_METRICS::DUALM
0.524	TURB_SPALART::SA_JAC
0.49	MPI_Barrier()
0.479	UPDATE_TURB::UPDATE
0.41	DUAL_NUMBERING::FG
0.371	LMPI_APP::SINGLE_STA
0.36	TURB_SPALART::JACOB
0.352	TIMEACC::TIME_DIAG_I
0.336	MPI_Init()
0.285	LMPI_APP::SINGLE_MAT
0.283	PPARTY_METIS::MY_ME
0.275	PPARTY::NODE_CELL_C
0.269	FLUX_UTIL::L2NORM [{

:POINT_SOLVE_5 [{point_solver.F90} {2757,5}-{2917,19}] OUS::EDGEJP [{jacobian_viscous.f90} {440,7}-{1584,22}] :ROE_FLUX [{flux_perfgas.f90} {397,5}-{597,20}] _JACOBIAN [{fill_jacobians.f90} {19,3}-{341,30}] S3D_READ_VGRID_C2N_SM [{puns3d_io_c2n.f90} {1641,14}] RAD_VARIABLE [{gradient_driver.f90} {110,3}-{508,30}]

S_TET [{flux_perfgas.f90} {3023,14}] ESID [{turb_spalart.f90} {219,14}] NT_GRADIENTS [{flux_turb.f90} {796,3}-{899,36}] timestep.f90} {20,3}-{331,24}] F_SOLVE_1 [{point_solver.F90} {1083,3}-{1206,30}] _ASSEMBLY_RES_CONV_DIFF [{turb_spalart.f90} {5048,14}]

_ASSEMBLY_JAC_CONV_DIFF [{turb_spalart.f90} {5209,14}] ENT::EDGE_POINTER_DRIVER [{pparty_mixed_element.f90} {74,3}-{502,36}] TE_Q [{update_mean.F90} {723,3}-{803,25}] 5 [{point_lu_ddq.f90} {1566,3}-{1636,21}] TERLEAF [{solution.F90} {3214,14}] MATRIX_TO_GRID [{dual_numbering.f90} {122,3}-{143,32}] METRIC [{grid_metrics.f90} {2289,3}-{2791,27}] ACOB [{turb_spalart.f90} {757,14}]

E_VALUES_TURB [{update_turb.f90} {854,3}-{877,35}] GRID_TO_MATRIX [{dual_numbering.f90} {94,3}-{115,32}] ART_MATRIX_XFER [{Impi_app.F90} {7907,3}-{8132,41}] BIAN_SA [{turb_spalart.f90} {88,14}] NC [{timeacc.f90} {1067,3}-{1330,29}]

TRIX_COMPLETE_XFER [{lmpi_app.F90} {11520,3}-{11626,44}] ETIS [{pparty_metis.F90} {116,3}-{545,24}] CHOPPER [{pparty.f90} {41,3}-{453,33}] {flux_util.f90} {31,3}-{249,23}]