Profiling LArTest with OpenlSpeedshop

Soon Yung Jun and Hans Wenzel (PDS)

LArSoft Coordination Meeting

April 25, 2017

Application: LArTest

- A standalone Geant4 application (developed by H. Wenzel)
  - Cubic (5mx5mx5m) LAr fiducial volume
  - GDML to assign step limits and sensitive detector to volumes
  - Optical (scintillation) photons produced in sensitive detector
- Computing performance monitoring features
  - Event time
  - Memory (IgProf, statm)
  - Statistics for tracks/steps by the particle type
- Goal for profiling LArTests
  - Monitor Geant4 part of computing performance changes with LAr detectors (energy, particle, physics list)
  - Integrate into the Geant4 computing performance task (https://g4cpt.fnal.gov)

Profiler: OpenISpeedshop (OSS)

- Comprehensive performance analysis for sequential, multithreaded, and MPI applications
- Open source (the Krell institute, https://openspeedshop.org) and one of ASCR profiling tools (TAU, HPCToolkit, Jumpshot, …)
- The base functionality includes
  - Sampling experiment (light-weighted)
  - Support call stack analysis
  - Hardware performance (PAPI) counters
  - Multi-threaded, MPI profiling and tracing
  - Memory function tracing, I/O profiling and tracing, and etc.
- Tested on a variety of Linux clusters and supports parallel hardware architectures (Intel MIC, NVIDIA CUDA) as well as HPC systems (Cray, Blue Gene)

OSS: Installation and Performance Measurement

- Installation: a typical build (with the version 2.2)

```
./install-tool --build-krell-root
               --krell-root-prefix  ${install_dir}/krellroot_v2.2
               --with-openmpi /usr/local/openmpi-1.8.1
./install-tool --build-offline
               --openss-prefix ${install_dir}/openspeedshop2.2
               --krell-root-prefix ${install_dir}/krellroot_v2.2
               --with-openmpi /usr/local/openmpi-1.8.1
```

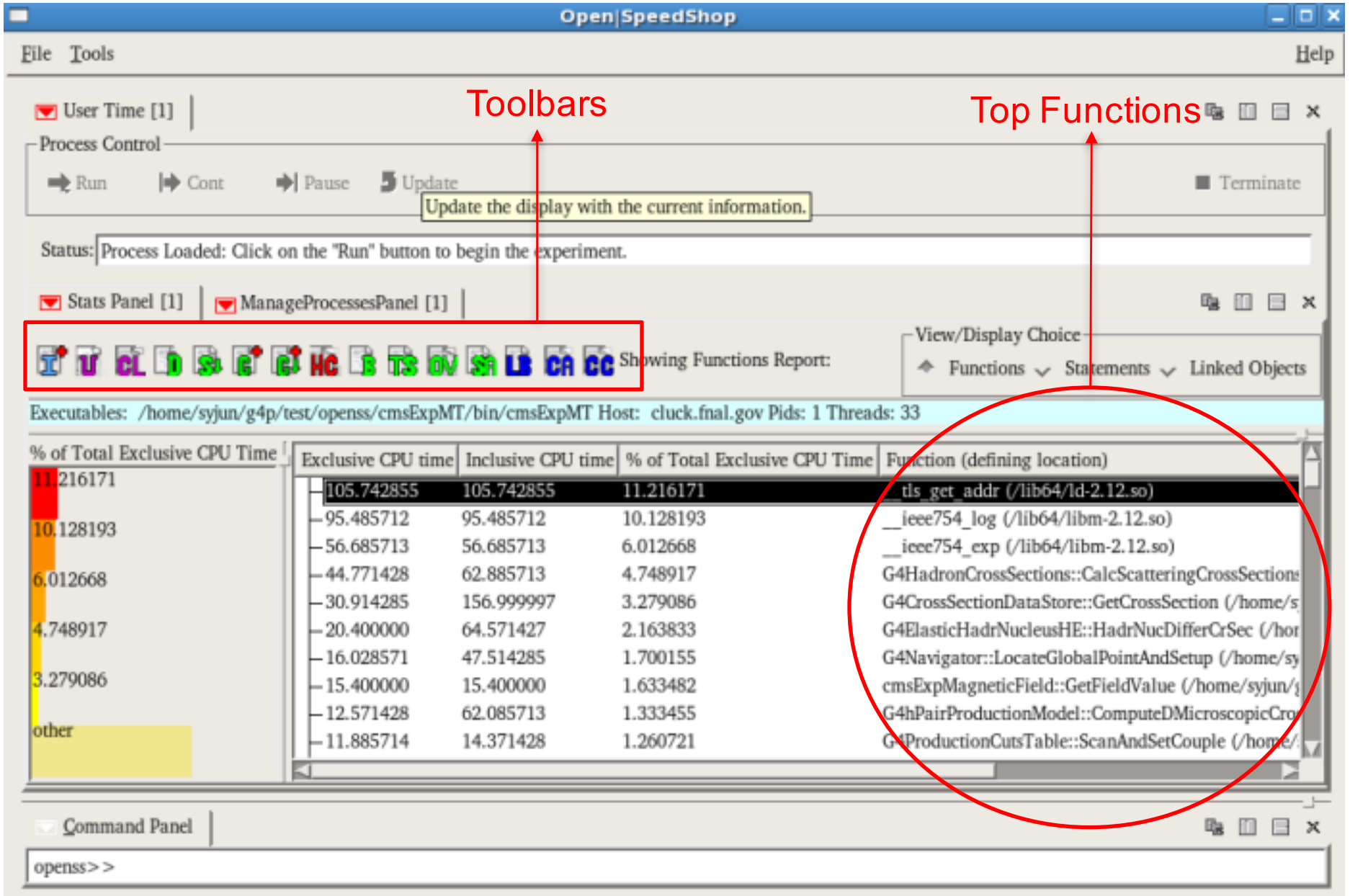
- Running an experiment: unmodified binary instrumentation

```
osspcsamp "lArTest lArBox.gdml profile.pi-5GeV”  [frequency]
```

- Performance analysis: GUI example with the output db file

```
openss -f lArTest-pcsamp.openss
```

OSS (GUI): Default View and Statistical Panel



Sampling Experiments in OSS

- pcsamp (periodic sampling the program counters)
  - low overhead overview of time distribution
- usertime (call path profiling)
  - inclusive and exclusive timing data
  - call paths, caller and callee relationships
- hwcsamp (periodic sampling hardware counters)
  - profile of hardware counter events (PAPI events)
- pthreads (POSIX thread tracing)
- mem (memory tracing)
  - call paths for memory related function call events
  - aggregate and individual rank, thread, or processing timings
- io (I/O tracing)
- Many other useful experiments

OSS: Measurement Overheads and Output Size

- pcsamp: exclusive time - insensitive to sampling frequency (default 100Hz)

Frequency	Time(sec)	OverHead(%)	DB size(MB)
base :	52.20	-	
50 Hz:	52.27	0.13	0.376832
100 Hz:	52.62	0.80	0.486400
200 Hz	52.36	0.31	0.607232
500 Hz	52.98	1.49	0.811008
1000 Hz:	52.65	0.86	0.971776
10000 Hz:	52.76	1.07	1.012736

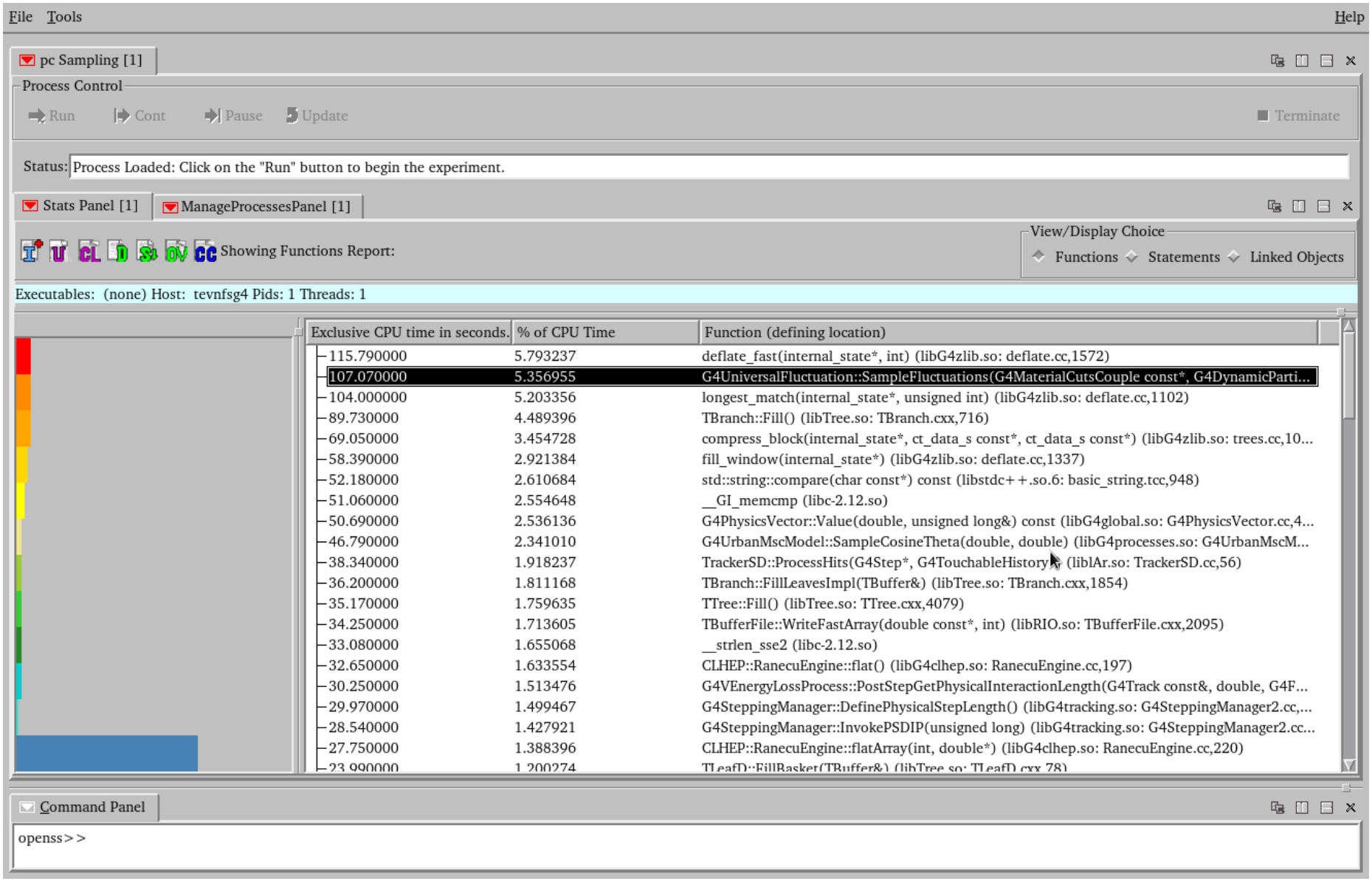
- usertime: inclusive time and call paths – large overhead (default 35): similar overhead for hwcsamp

Frequency	Time(sec)	OverHead(%)	DB size(MB)
base :	52.80	-	
35 Hz:	53.89	2.06	1.087488
50 Hz:	54.33	2.90	1.430528
100 Hz:	56.21	6.46	2.355200
200 Hz	60.25	14.11	4.208640
1000 Hz:	92.84	75.83	18.725888

Preliminary Performance Experiments with LArTest

- LArTest configuration
  - Beam: 5 GeV pi-
  - Step limit: 0.01 cm
  - Physics list: FTFP_BERT + Standard EM
  - 1000 events
- osspcsamp (100 Hz)
  - I/O (digitization) ON
  - Analysis ON
- ossusertime and osshwcsamp (35 Hz)
  - I/O (digitization) OFF
  - Analysis OFF

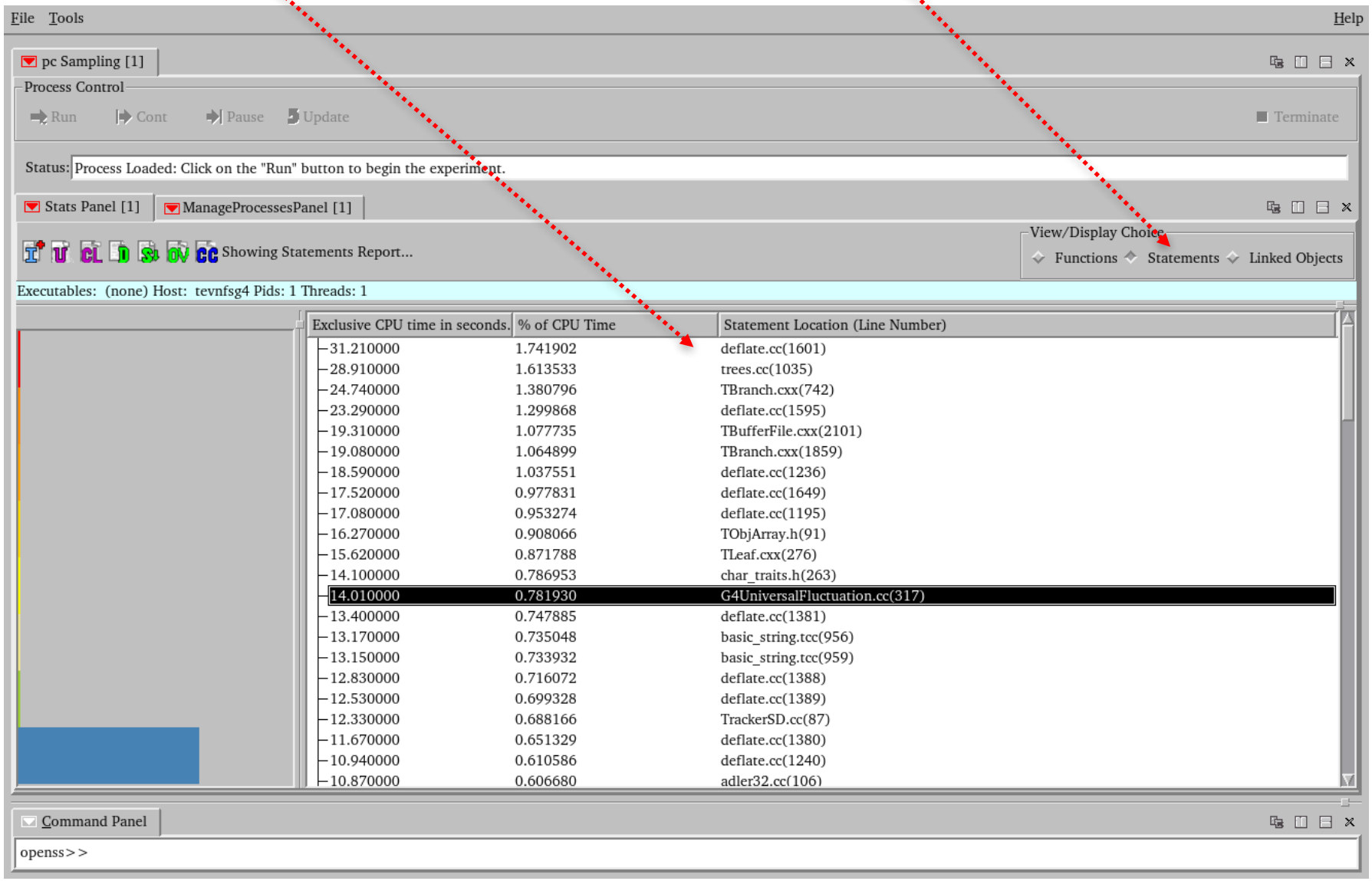
osspcsamp LArTest: Functions (Exclusive CPU Time)



- I/O (zlib/root functions) are leading top functions (> 30%)

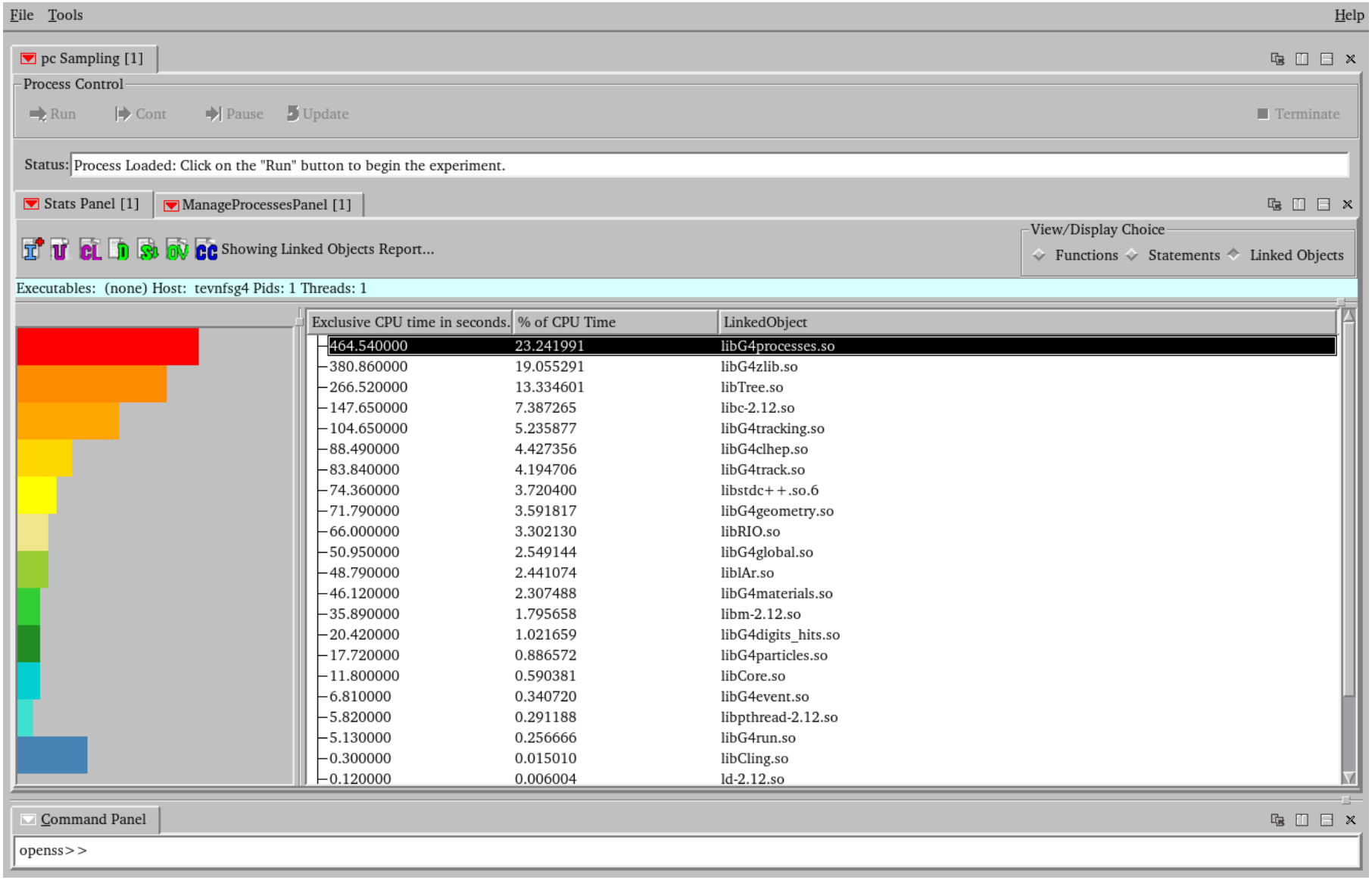
osspcsamp LArTest: Statements (Line Numbers)

- Select statement level granularity
- List line numbers in program that took most of time



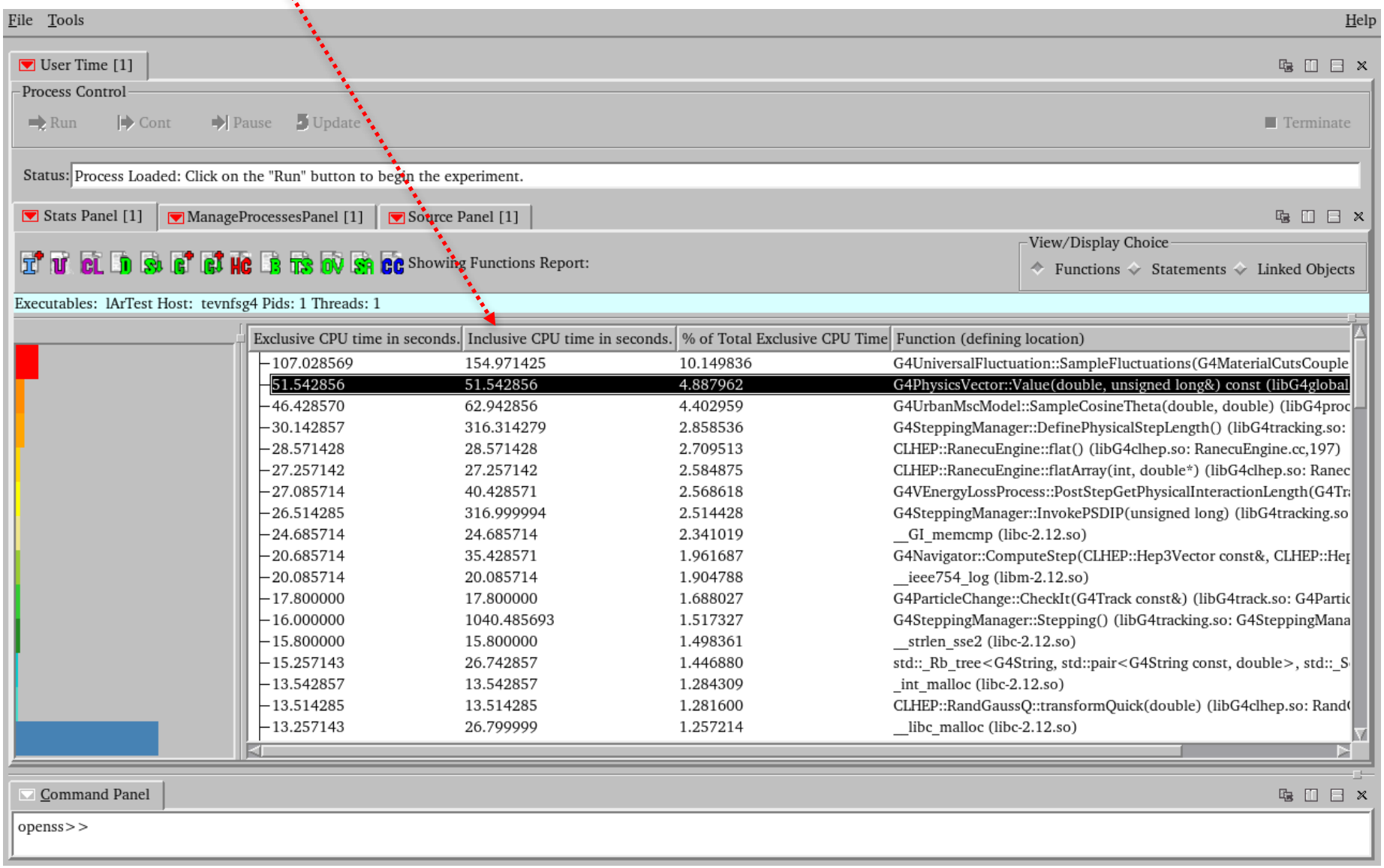
osspcsamp LArTest: Linked Objects

- The library in which the associated function is located (aggregated by shared objects)



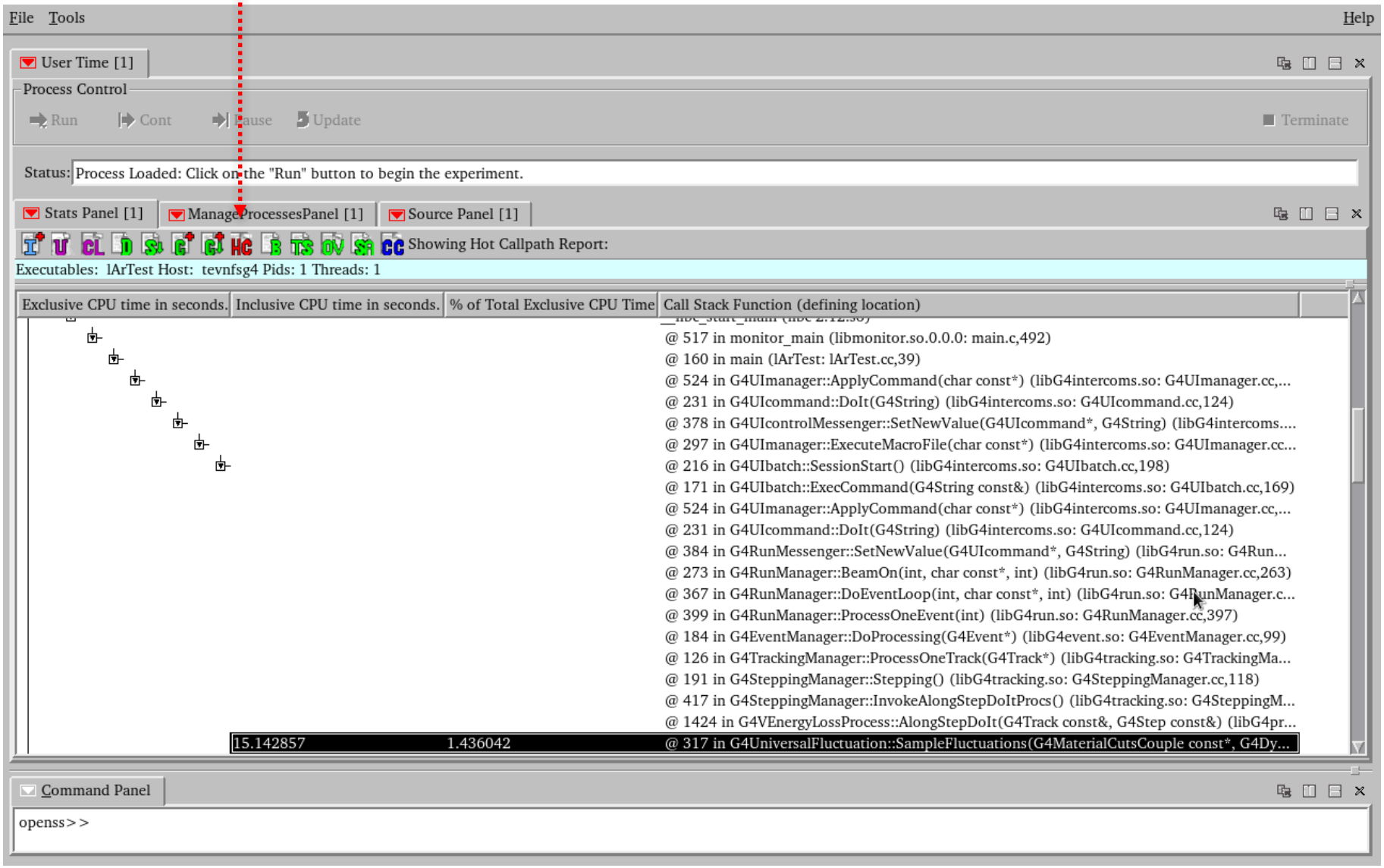
ossusertime LArTest: Call Path (Functions)

- Function calls observed anywhere in the stack
- The inclusive time taken by the function and all its callees



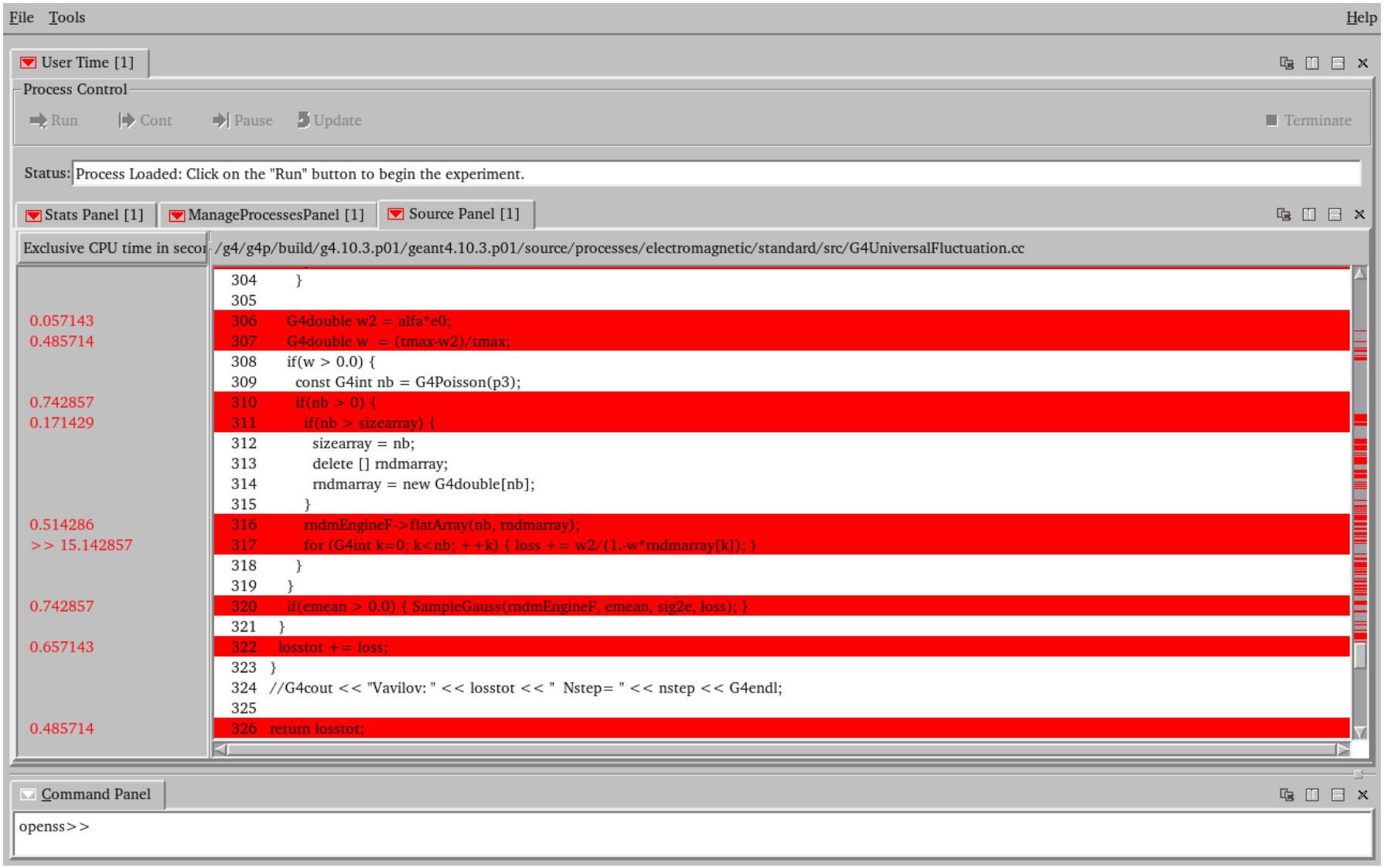
ossusertime LArTest: Hot Call Path

- Relationship between caller and callee
- The paths through the application that take the most time



ossusertime LArTest: Hot Call (Source)

- Exclusive time on highlighted lines that indicate relatively high CPU times



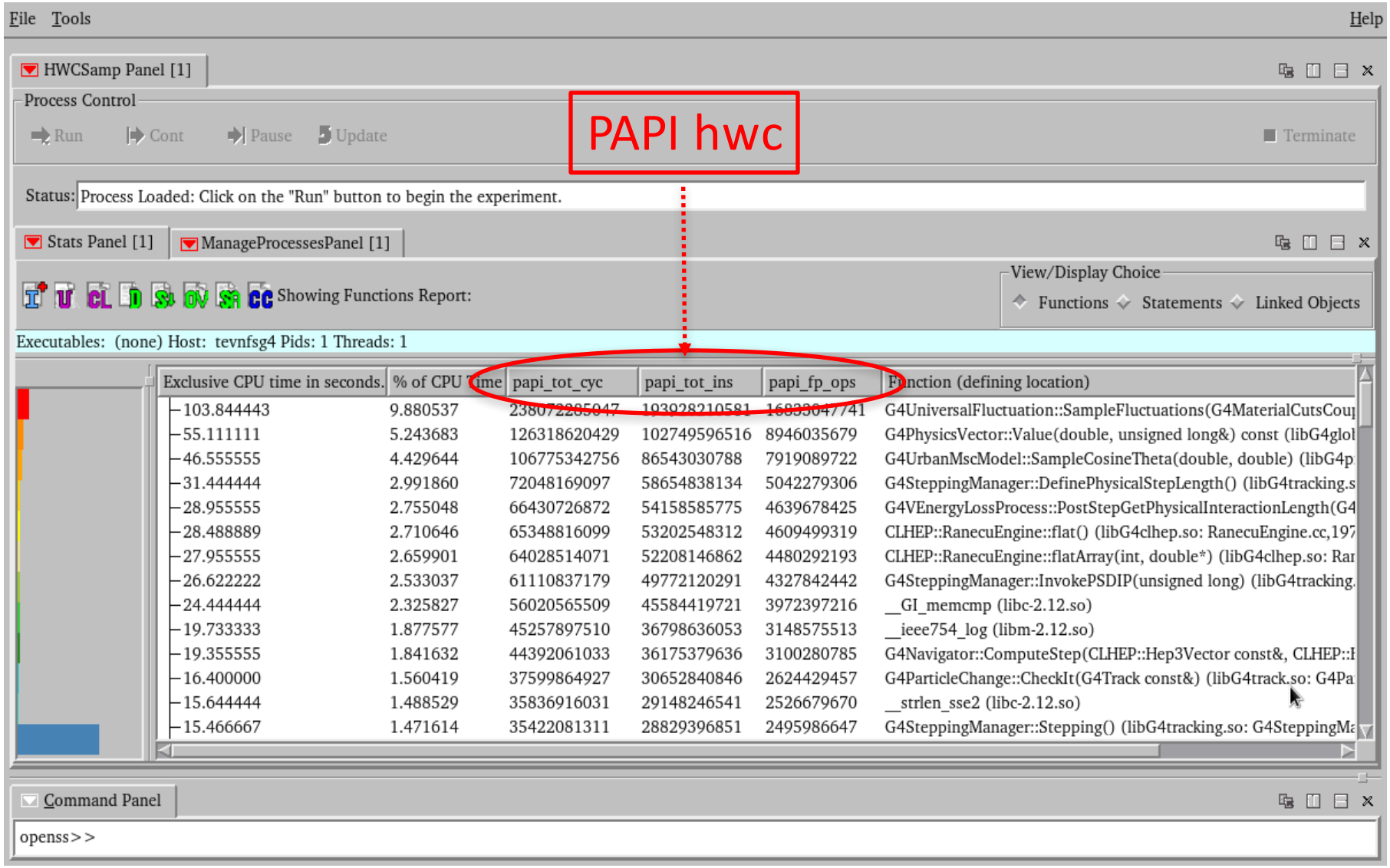
OpenISpeedshop: Experiments with Hardware Counters

- Periodic sampling hardware counters (hwcsamp)
- Supports both derived and non-derived PAPI presets
  - Ex.: Wilson intel12 nodes: 58 available events, 14 are derived
- A list of some possible hardware counter combinations

For **Xeon** processors:	
PAPI_FP_INS, PAPI_LD_INS, PAPI_SR_INS	Load store info, memory bandwidth needs
PAPI_L1_DCM, PAPI_L1_TCA	L1 cache hit/miss ratios
PAPI_L2_DCM, PAPI_L2_TCA	L2 cache hit/miss ratios
LAST_LEVEL_CACHE_MISSES, LAST_LEVEL_CACHE_REFERENCES	L3 cache info
MEM_UNCORE_RETIRED:REMOTE_DRAM, MEM_UNCORE_RETIRED:LOCAL_DRAM	Local/nonlocal memory access
For **Opteron** processors:	
PAPI_FAD_INS, PAPI_FML_INS	Floating point add multiply
PAPI_FDV_INS, PAPI_FSQ_INS	Square root and divisions
PAPI_FP_OPS, PAPI_VEC_INS	Floating point and vector instructions
READ_REQUEST_TO_L3_CACHE:ALL_CORES, L3_CACHE_MISSES:ALL_CORES	L3 cache

osshwcsamp LArTest: Hardware Counter Sampling

- ‘papi_avail -a’ will show available papi events on a system
- Metrics for INS, FLOPS, memory and resource patterns, …



Code Performance by Hardware Counter Metrics

- Derivatives: examples

Hardware Counter Metrics Derivatives	Performance
IPC (Instruction/Cycle)	Large values suggest good balance with minimal stalls.
FPC (FLOPS/Cycle)	Large values for floating point intensive codes suggests efficient CPU utilization
FMO (FLOPS/Memory Ops)	Good data locality, Computational Intensity
LPC (Loads/Cycle)	Useful for calculating FMO, may indicate good stride through arrays.
SPC (Stores/Cycle)	Useful for calculating FMO, may indicate good stride through arrays.

- LArTest (Overall): 5 GeV pi- (Intel Xeon X5650@2.67GHz)
  - IPC = 0.79 (relatively small)
  - FMO = 0.32

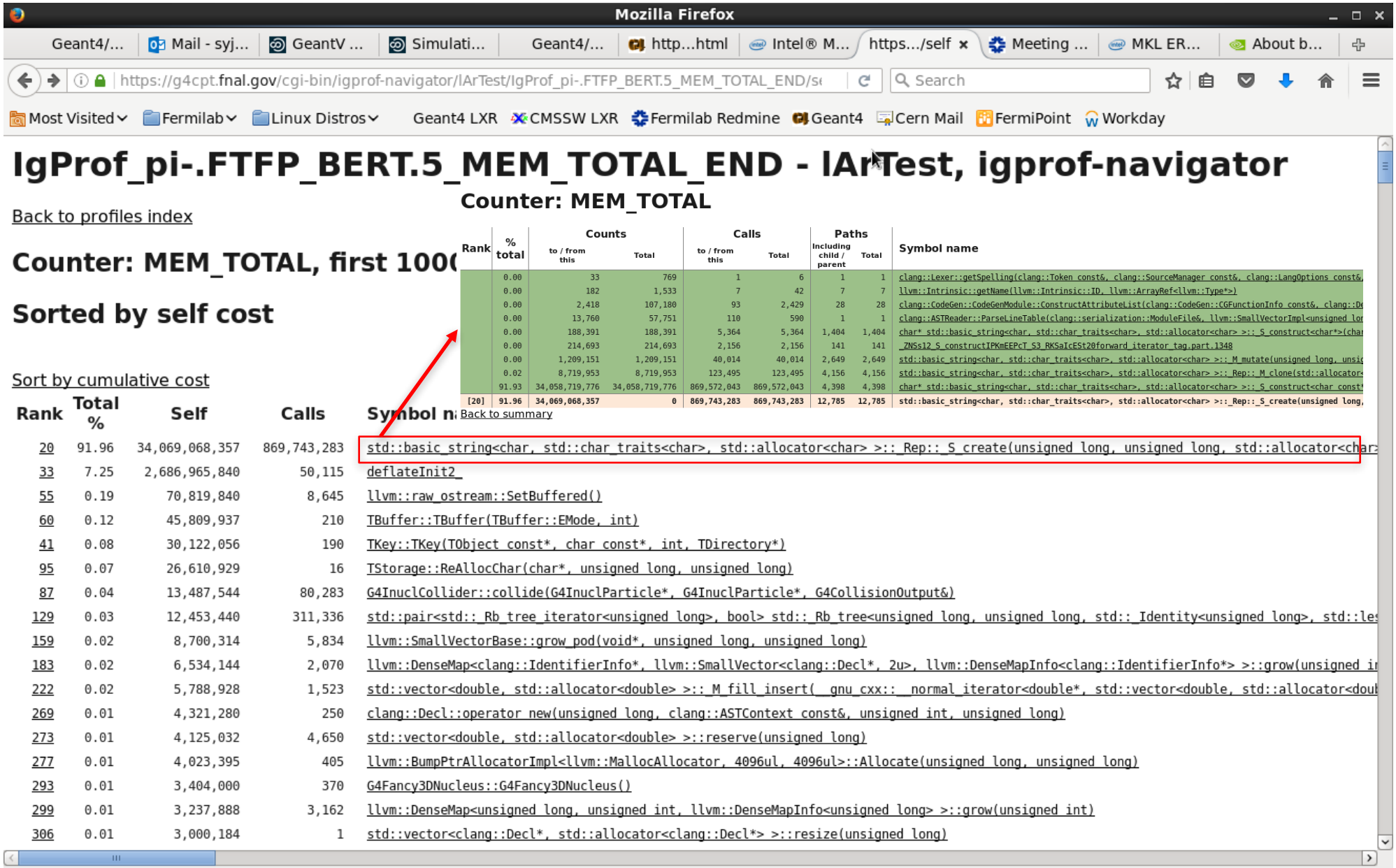
Other useful OSS Features

- Flexible analysis options (GUI, command line, online)
- Export report data in different formats (text, cvs, chart)
- Multi-threading capability
- Compare two experiments (osscompare): examples
  - two releases
  - two experiments with the different numbers of threads
- Call path analysis based on DB
- Experiments for parallel codes (MPI tracing)

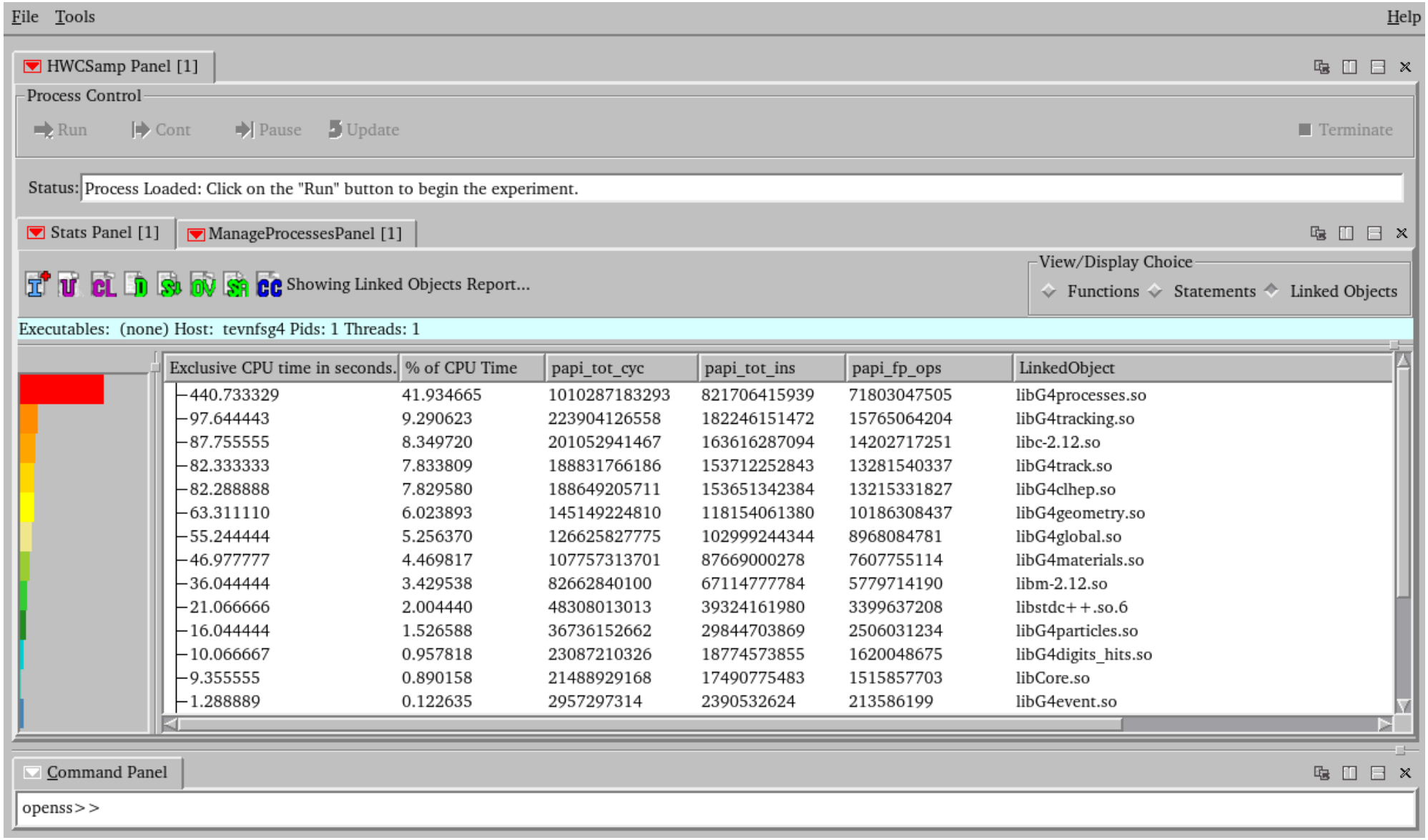
Summary and Plan

- Developed a standalone Geant4 application (LArtest) with a LAr fiducial volume and added functionalities for computing performance measurement and analysis
- Profiled LArTest with OpenlSpeedshop (and lgProf)
- Extend the test with other geometry descriptions
  - protodune.gdml (v3 from Tom Junk)
  - GDML extension for material properties
- Monitor Geant4 part of computing performance changes for LAr-based detectors by
  - Beam energy
  - Particle type
  - Physics list
  - Geant4 (reference) release

LArTest: IgProf (TOTAL MEM)

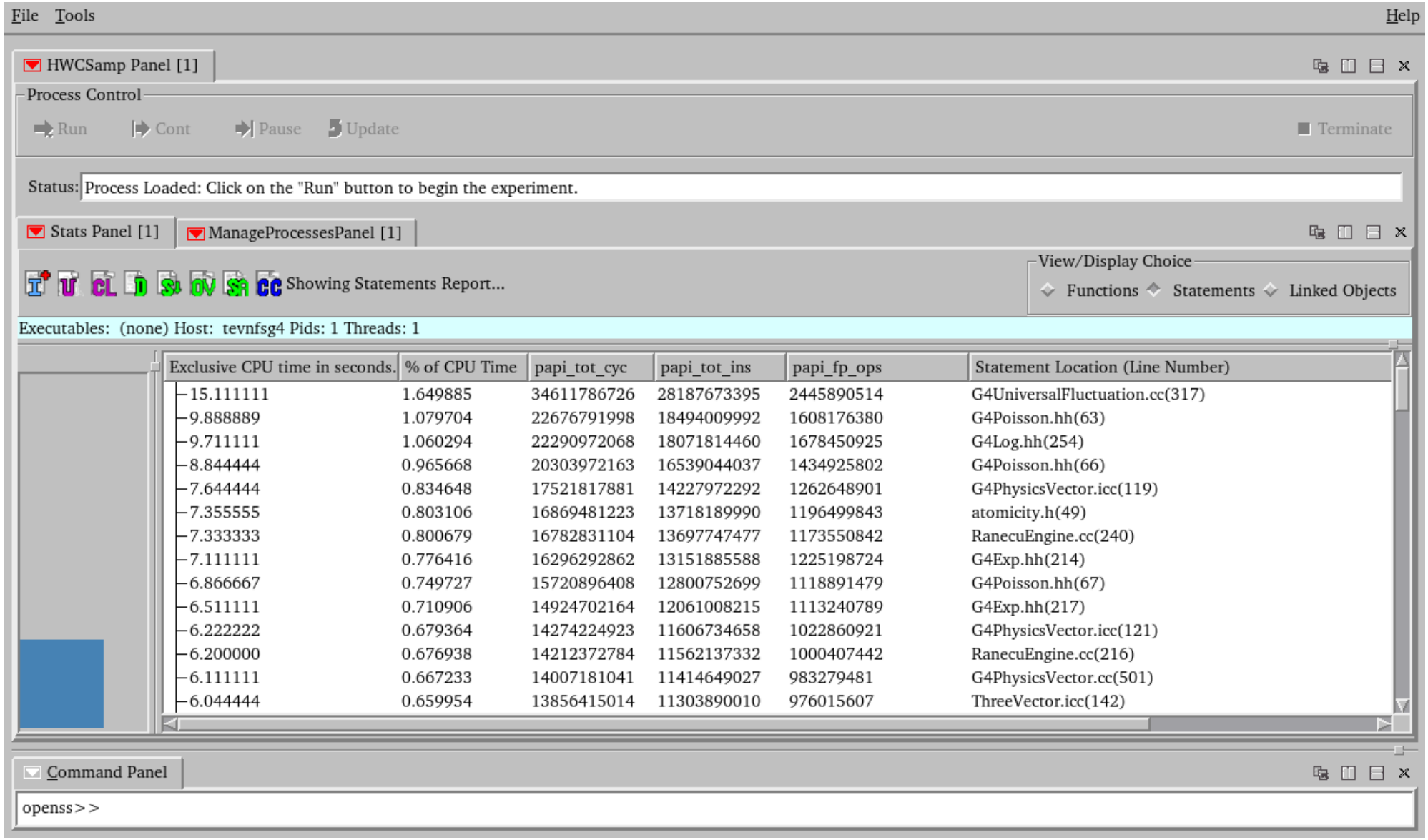


osshwcsamp LArTest: Libraries



Exclusive CPU time in seconds.	% of CPU Time	papi_tot_cyc	papi_tot_ins	papi_fp_ops	LinkedObject
440.733329	41.934665	1010287183293	821706415939	71803047505	libG4processes.so
97.644443	9.290623	223904126558	182246151472	15765064204	libG4tracking.so
87.755555	8.349720	201052941467	163616287094	14202717251	libc-2.12.so
82.333333	7.833809	188831766186	153712252843	13281540337	libG4track.so
82.288888	7.829580	188649205711	153651342384	13215331827	libG4clhep.so
63.311110	6.023893	145149224810	118154061380	10186308437	libG4geometry.so
55.244444	5.256370	126625827775	102999244344	8968084781	libG4global.so
46.977777	4.469817	107757313701	87669000278	7607755114	libG4materials.so
36.044444	3.429538	82662840100	67114777784	5779714190	libm-2.12.so
21.066666	2.004440	48308013013	39324161980	3399637208	libstdc++.so.6
16.044444	1.526588	36736152662	29844703869	2506031234	libG4particles.so
10.066667	0.957818	23087210326	18774573855	1620048675	libG4digits_hits.so
9.355555	0.890158	21488929168	17490775483	1515857703	libCore.so
1.288889	0.122635	2957297314	2390532624	213586199	libG4event.so

osshwcsamp LArTest: Statements



Exclusive CPU time in seconds.	% of CPU Time	papi_tot_cyc	papi_tot_ins	papi_fp_ops	Statement Location (Line Number)
15.111111	1.649885	34611786726	28187673395	2445890514	G4UniversalFluctuation.cc(317)
9.888889	1.079704	22676791998	18494009992	1608176380	G4Poisson.hh(63)
9.711111	1.060294	22290972068	18071814460	1678450925	G4Log.hh(254)
8.844444	0.965668	20303972163	16539044037	1434925802	G4Poisson.hh(66)
7.644444	0.834648	17521817881	14227972292	1262648901	G4PhysicsVector.icc(119)
7.355555	0.803106	16869481223	13718189990	1196499843	atomicity.h(49)
7.333333	0.800679	16782831104	13697747477	1173550842	RanecuEngine.cc(240)
7.111111	0.776416	16296292862	13151885588	1225198724	G4Exp.hh(214)
6.866667	0.749727	15720896408	12800752699	1118891479	G4Poisson.hh(67)
6.511111	0.710906	14924702164	12061008215	1113240789	G4Exp.hh(217)
6.222222	0.679364	14274224923	11606734658	1022860921	G4PhysicsVector.icc(121)
6.200000	0.676938	14212372784	11562137332	1000407442	RanecuEngine.cc(216)
6.111111	0.667233	14007181041	11414649027	983279481	G4PhysicsVector.cc(501)
6.044444	0.659954	13856415014	11303890010	976015607	ThreeVector.icc(142)