KRELL

institute

Open | SpeedShop™

Open Source Performance Analysis for Large Scale Systems

Open|SpeedShop Capabilities and Internal Structure: Current to Petascale

CScADS Workshop, July 16-20, 2007

Jim Galarowicz, Krell Institute

Trademark Acknowledgements

- *Intel, Intel Inside (logos) and Itanium are trademarks of Intel Corporation in the United States, other countries, or both.*
- *Linux is a trademark of Linus Torvalds in the United States, other countries or both.*
- *Qt and the Qt logo are trademarks of Trolltech in Norway, the United States and other countries.*
- *SGI SpeedShop, IRIX, SGI and SGI Altix are trademarks of Silicon Graphics Inc.*
- *IBM is a registered trademarks of International Business Machines Corporation in the United States, other countries, or both.*
- *All other trademarks mentioned herein are the property of their respective owners*

Talk Outline

- Open|SpeedShop – What is it?
- Capabilities and Feature Overview
- Internal Components and Interaction
- Petascale Computing Support
- Questions

Open|SpeedShop
What is it?

- Comprehensive Parallel Performance Analysis Framework
  - **Goal:** Most common performance analysis steps in one tool
  - **Targets Users *and* Tool Developers**
  - **Set of performance analysis tools built on flexible framework**
- **Funding**
  - **DOE/NNSA as part of ASC PathForward**
  - **Initial phase co-funded by SGI**
- **Status**
  - **Version 1.0 available as source and RPMs**
  - **Development version available through cvs**
  - **Open Source: code is GPL/LGPL**

Partners

- Krell Institute
  - Hosts Development
- ASC Tri-Laboratories
  - Lawrence Livermore
  - Los Alamos
  - Sandia
- University of Wisconsin & University of Maryland
  - DynInst & Infrastructure

Acknowledgements

- Open|SpeedShop Team Members
  - Scott Cranford, Sandia National Labs
  - Jim Galarowicz, Krell Institute
  - Bill Hachfeld, Krell Institute
  - Don Maghrak, Krell Institute
  - Dave Montoya, Los Alamos National Labs
  - Martin Schulz, Lawrence Livermore National Labs

- Dyninst Team Members
  - Bart Miller
  - Matt Legendre
  - Drew Bernat

Overview / Highlights

- Open Source Performance Analysis Tool
  - *Extensible* by using plugins for data collection and viewing
  - Emphasis on *usability* from the start - usability studies
- **Instrumentation at Runtime**
  - **Use of** *unmodified application binaries*
  - *Attach/Detach to/from* running executables/applications
  - *Load and Start* executables/applications into tool
- **Flexible and Easy to use user interfaces**
  - *GUI* with wizards to guide users through creation of experiment
  - *Command Line* uses dbx/gdb like commands
  - *Batch* executes commands file or simple create, run view preset
  - *Python Scripting* uses API that feeds into command line interface

Overview / Highlights

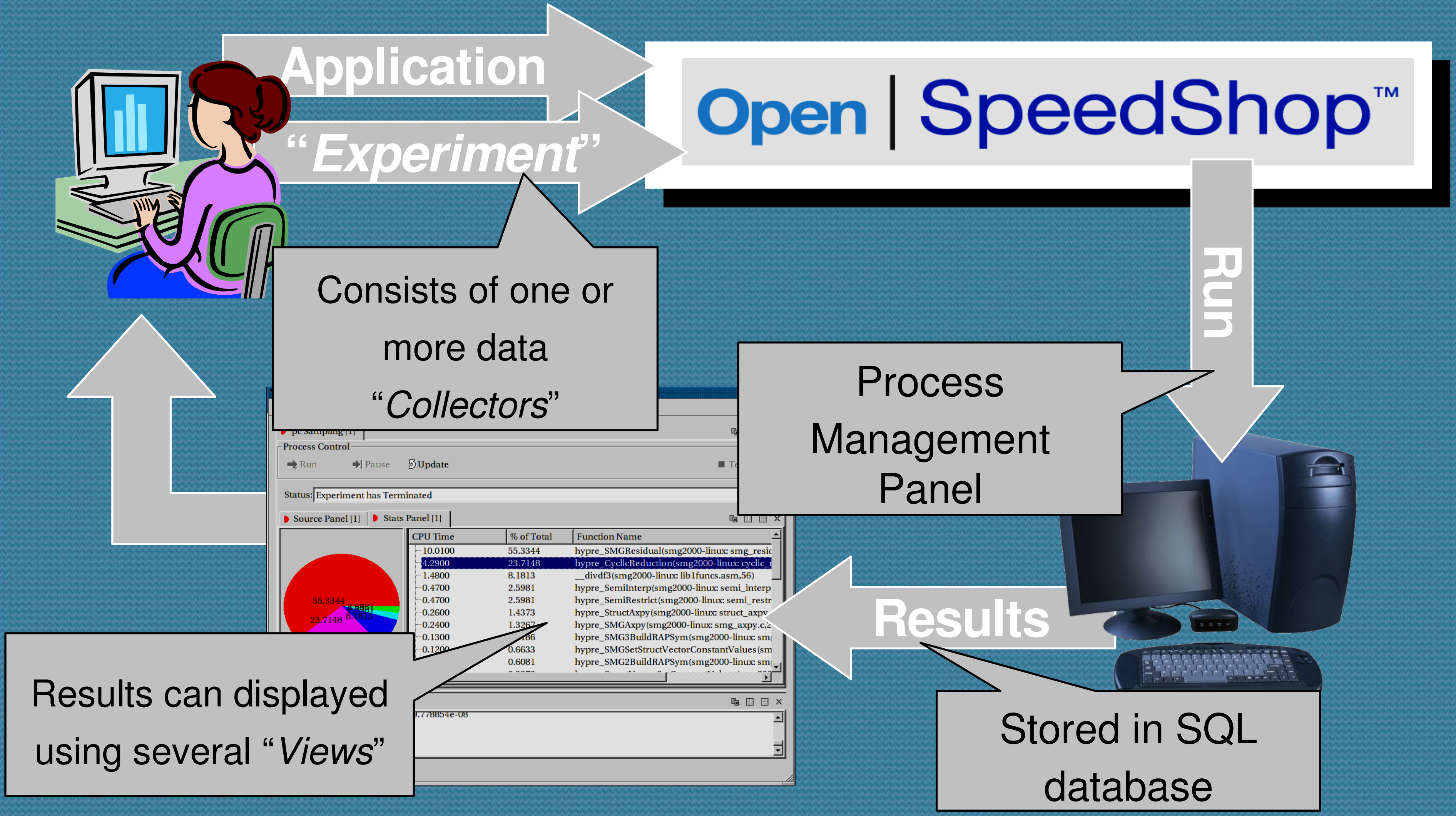
- **Large Range of Platforms**
  - *Linux Clusters* **with x86, IA-64, Opteron, and EM64T CPUs**
  - *SSI* **systems**
  - **Designed with** *portability* **in mind**
- **Availability**
  - **Used at** *all three ASC labs* **with lab-size applications**
  - **Source and RPM versions available**
  - *www.openspeedshop.org*
- Linux versions
  - Tested on typical Linux distributions
    (including *SLES, RHEL, Fedora Core, Suse ....*)

Features:

Performance Experiments

- Available Now:
  - PC sampling (*pcsamp*)
  - User time (*usertime*)
  - Hardware counter (*hwc, hwctime*)
  - MPI call tracing (*mpi, mpit*)
  - I/O call tracing (*io, iot*)
  - Floating Point Exception (FPE) tracing (*fpe*)
- Extensible
  - Plugin concept for collectors and views
  - Well defined/documented APIs – Plugin Guide

Typical Workflow



Consists of one or more data “*Collectors*”

Process Management Panel

Stored in SQL database

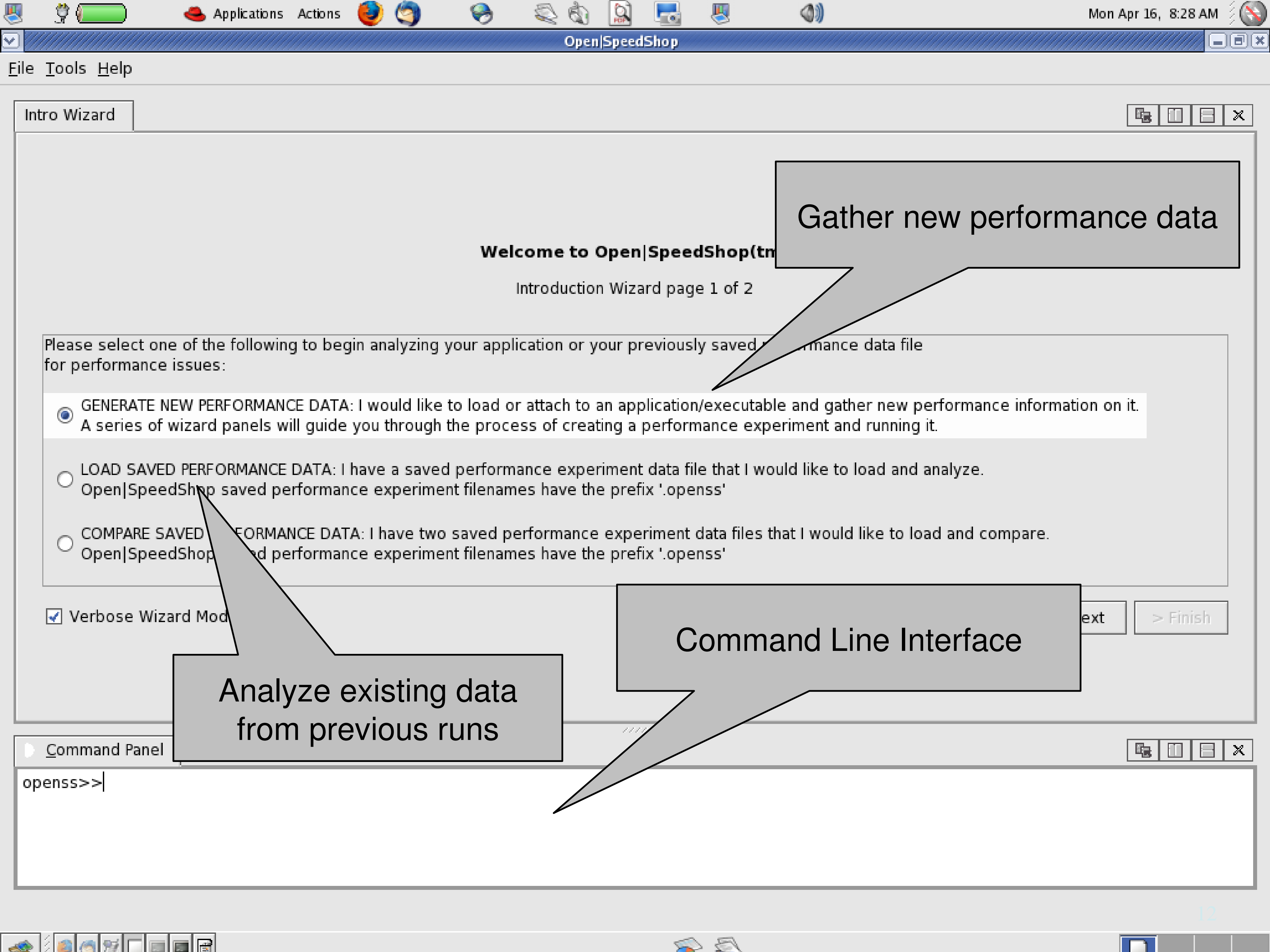
Results can displayed using several “*Views*”

Process Control

Status: Experiment has Terminated

CPU Time	% of Total	Function Name
10.0100	55.3344	hypre_SMGResidual(smg2000-linux: smg_resic
4.2900	23.7148	hypre_CyclicReduction(smg2000-linux: cyclic_r
1.4800	8.1813	__divdf3(smg2000-linux: lib1funcs.asm,56)
0.4700	2.5981	hypre_SemiInterp(smg2000-linux: semi_interp
0.4700	2.5981	hypre_SemiRestrict(smg2000-linux: semi_restr
0.2600	1.4373	hypre_StructAxpy(smg2000-linux: struct_axpy
0.2400	1.3267	hypre_SMGAxpy(smg2000-linux: smg_axpy.c,2
0.1300	186	hypre_SMG3BuildRAPSym(smg2000-linux: sm
0.1200	0.6633	hypre_SMGSetStructVectorConstantValues(sm
	0.6081	hypre_SMG2BuildRAPSym(smg2000-linux: sm

.778854e-08



Welcome to Open|SpeedShop(tm

Introduction Wizard page 1 of 2

Please select one of the following to begin analyzing your application or your previously saved rmance data file for performance issues:

● GENERATE NEW PERFORMANCE DATA: I would like to load or attach to an application/executable and gather new performance information on it. A series of wizard panels will guide you through the process of creating a performance experiment and running it.

○ LOAD SAVED PERFORMANCE DATA: I have a saved performance experiment data file that I would like to load and analyze. Open|SpeedSh p saved performance experiment filenames have the prefix '.openss'

○ COMPARE SAVED FORMANCE DATA: I have two saved performance experiment data files that I would like to load and compare. Open|SpeedShop d performance experiment filenames have the prefix '.openss'

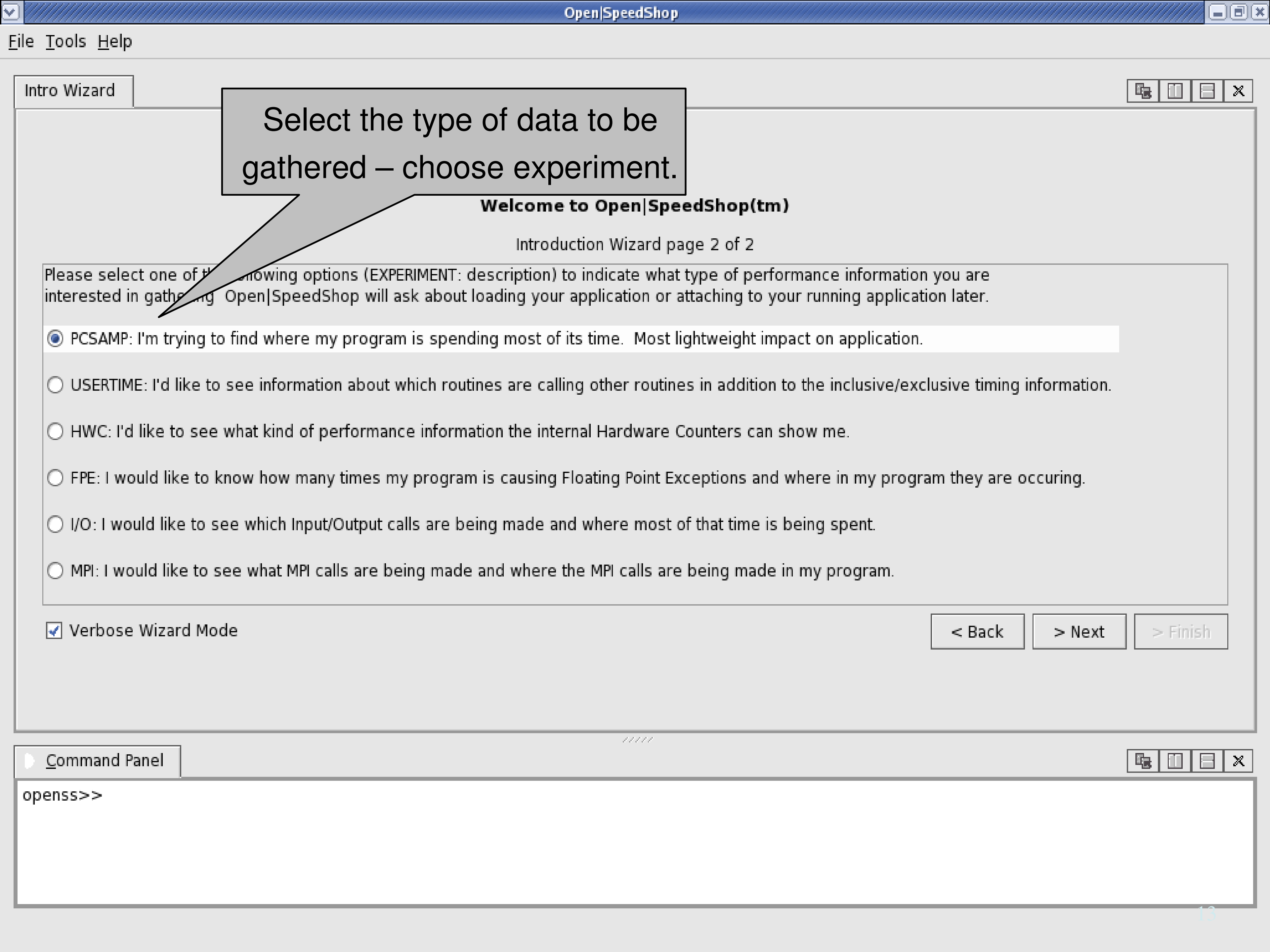
☑ Verbose Wizard Mod

```
openss>>
```

Gather new performance data

Analyze existing data from previous runs

Command Line Interface



Select the type of data to be gathered – choose experiment.

Welcome to Open|SpeedShop(tm)

Introduction Wizard page 2 of 2

Please select one of the following options (EXPERIMENT: description) to indicate what type of performance information you are interested in gathering. Open|SpeedShop will ask about loading your application or attaching to your running application later.

● PCSAMP: I'm trying to find where my program is spending most of its time. Most lightweight impact on application.

○ USERTIME: I'd like to see information about which routines are calling other routines in addition to the inclusive/exclusive timing information.

○ HWC: I'd like to see what kind of performance information the internal Hardware Counters can show me.

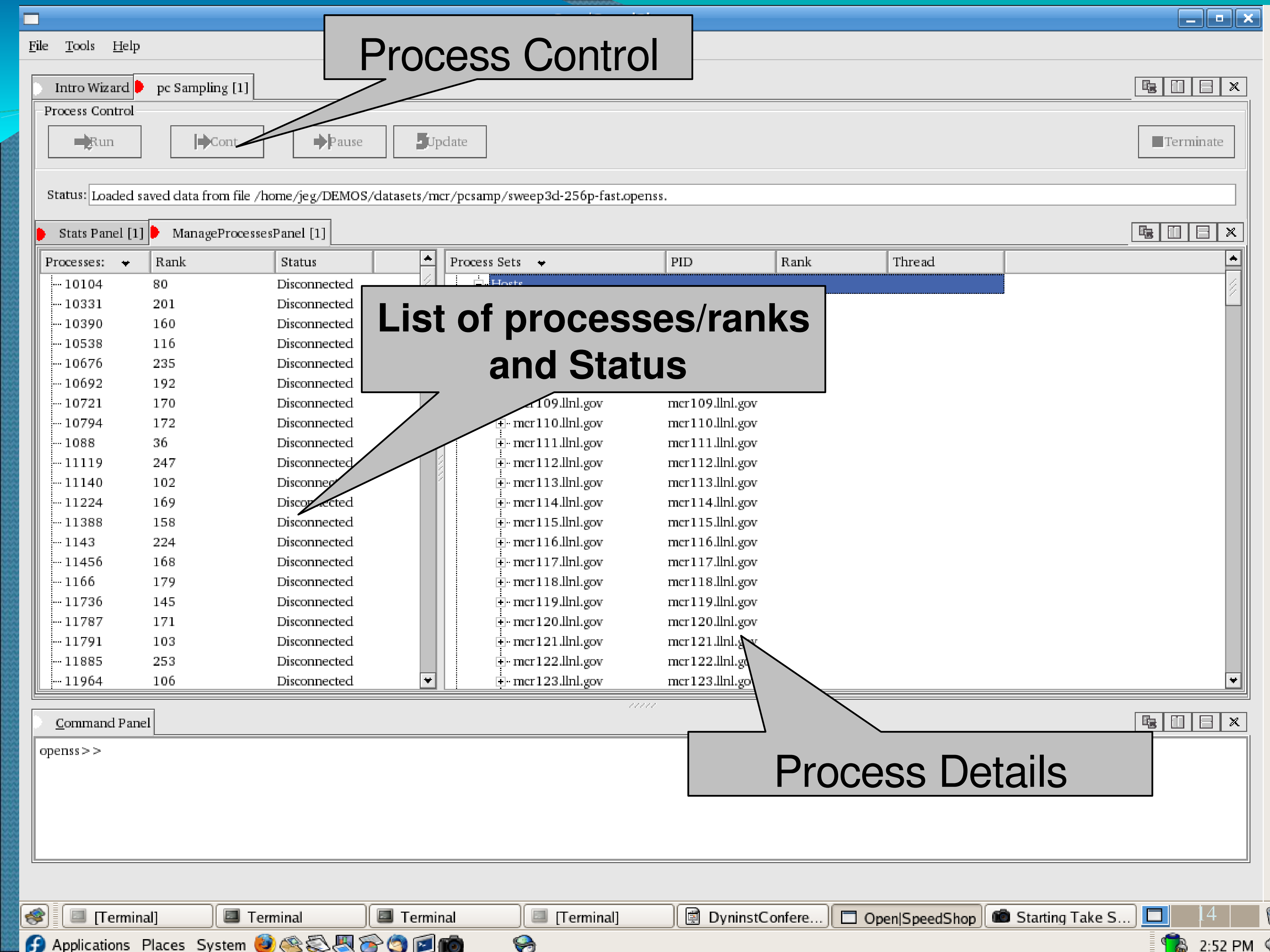
○ FPE: I would like to know how many times my program is causing Floating Point Exceptions and where in my program they are occuring.

○ I/O: I would like to see which Input/Output calls are being made and where most of that time is being spent.

○ MPI: I would like to see what MPI calls are being made and where the MPI calls are being made in my program.

☑ Verbose Wizard Mode

```
openss>>
```



Process Control

List of processes/ranks and Status

Process Details

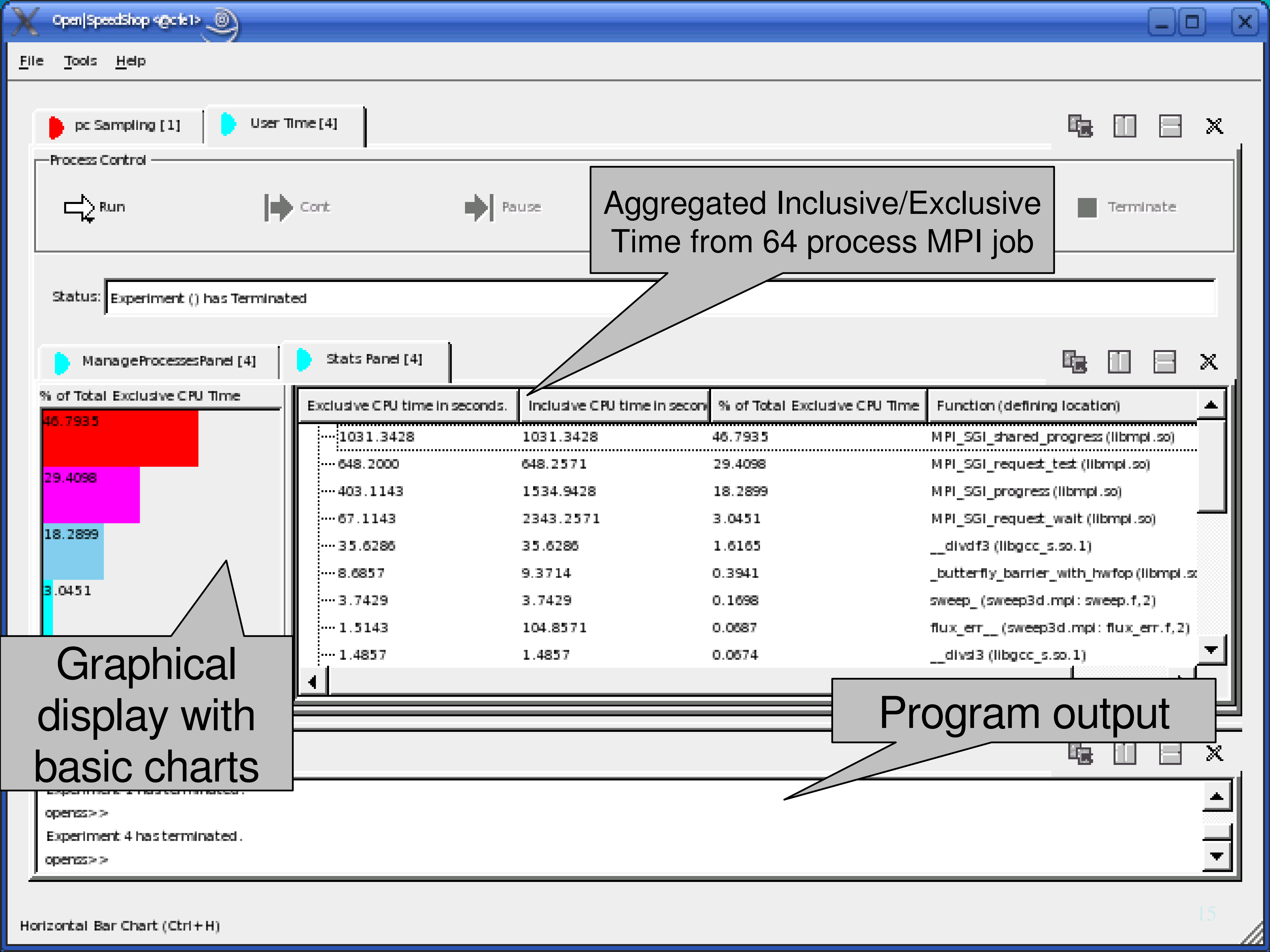
Process Control

Status: Loaded saved data from file /home/jeg/DEMOS/datasets/mcr/pcsamp/sweep3d-256p-fast.openss.

Processes:	Rank	Status
10104	80	Disconnected
10331	201	Disconnected
10390	160	Disconnected
10538	116	Disconnected
10676	235	Disconnected
10692	192	Disconnected
10721	170	Disconnected
10794	172	Disconnected
1088	36	Disconnected
11119	247	Disconnected
11140	102	Disconnected
11224	169	Disconnected
11388	158	Disconnected
1143	224	Disconnected
11456	168	Disconnected
1166	179	Disconnected
11736	145	Disconnected
11787	171	Disconnected
11791	103	Disconnected
11885	253	Disconnected
11964	106	Disconnected

Process Sets	PID	Rank	Thread
Hosts			
109.llnl.gov	mcr109.llnl.gov		
mcr110.llnl.gov	mcr110.llnl.gov		
mcr111.llnl.gov	mcr111.llnl.gov		
mcr112.llnl.gov	mcr112.llnl.gov		
mcr113.llnl.gov	mcr113.llnl.gov		
mcr114.llnl.gov	mcr114.llnl.gov		
mcr115.llnl.gov	mcr115.llnl.gov		
mcr116.llnl.gov	mcr116.llnl.gov		
mcr117.llnl.gov	mcr117.llnl.gov		
mcr118.llnl.gov	mcr118.llnl.gov		
mcr119.llnl.gov	mcr119.llnl.gov		
mcr120.llnl.gov	mcr120.llnl.gov		
mcr121.llnl.gov	mcr121.llnl.g		
mcr122.llnl.gov	mcr122.llnl.g		
mcr123.llnl.gov	mcr123.llnl.go		

```
openss>>
```



Status: Experiment () has Terminated

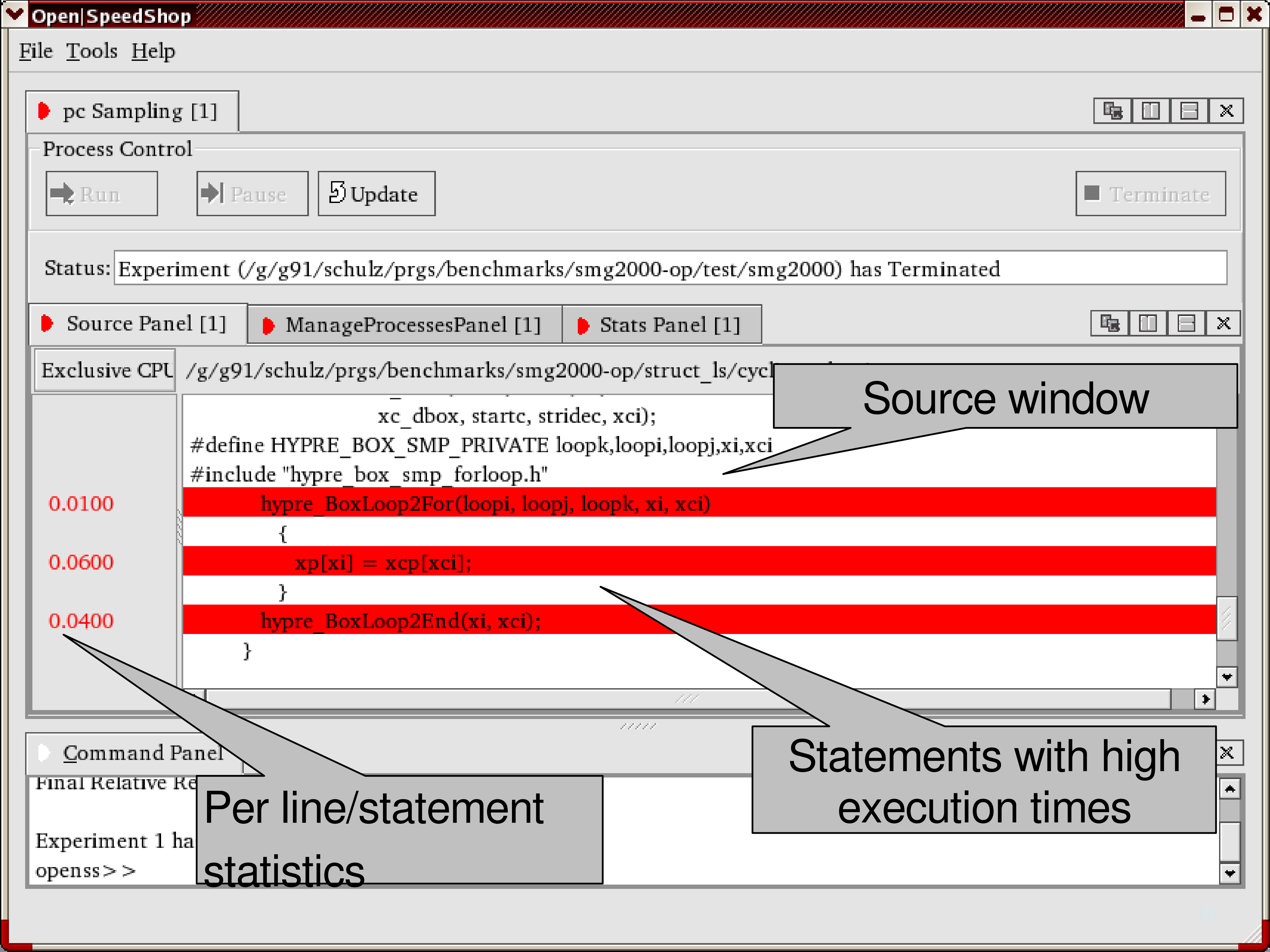
Exclusive CPU time in seconds.	Inclusive CPU time in secon	% of Total Exclusive CPU Time	Function (defining location)
1031.3428	1031.3428	46.7935	MPI_SGI_shared_progress (libmpi.so)
648.2000	648.2571	29.4098	MPI_SGI_request_test (libmpi.so)
403.1143	1534.9428	18.2899	MPI_SGI_progress (libmpi.so)
67.1143	2343.2571	3.0451	MPI_SGI_request_wait (libmpi.so)
35.6286	35.6286	1.6165	__divdf3 (libgcc_s.so.1)
8.6857	9.3714	0.3941	_butterfly_barrier_with_hwfop (libmpi.sc
3.7429	3.7429	0.1698	sweep_ (sweep3d.mpi: sweep.f,2)
1.5143	104.8571	0.0687	flux_err__ (sweep3d.mpi: flux_err.f,2)
1.4857	1.4857	0.0674	__divsi3 (libgcc_s.so.1)

```
Experiment 1 has terminated.
openss>>
Experiment 4 has terminated.
openss>>
```

Aggregated Inclusive/Exclusive Time from 64 process MPI job

Graphical display with basic charts

Program output



Status: Experiment (/g/g91/schulz/prgs/benchmarks/smg2000-op/test/smg2000) has Terminated

```
                         xc_dbox, startc, stridec, xci);
         #define HYPRE_BOX_SMP_PRIVATE loopk,loopi,loopj,xi,xci
         #include "hypre_box_smp_forloop.h"
0.0100         hypre_BoxLoop2For(loopi, loopj, loopk, xi, xci)
                 {
0.0600             xp[xi] = xcp[xci];
                 }
0.0400         hypre_BoxLoop2End(xi, xci);
             }
```

```
Final Relative Re

Experiment 1 ha
openss>>
```

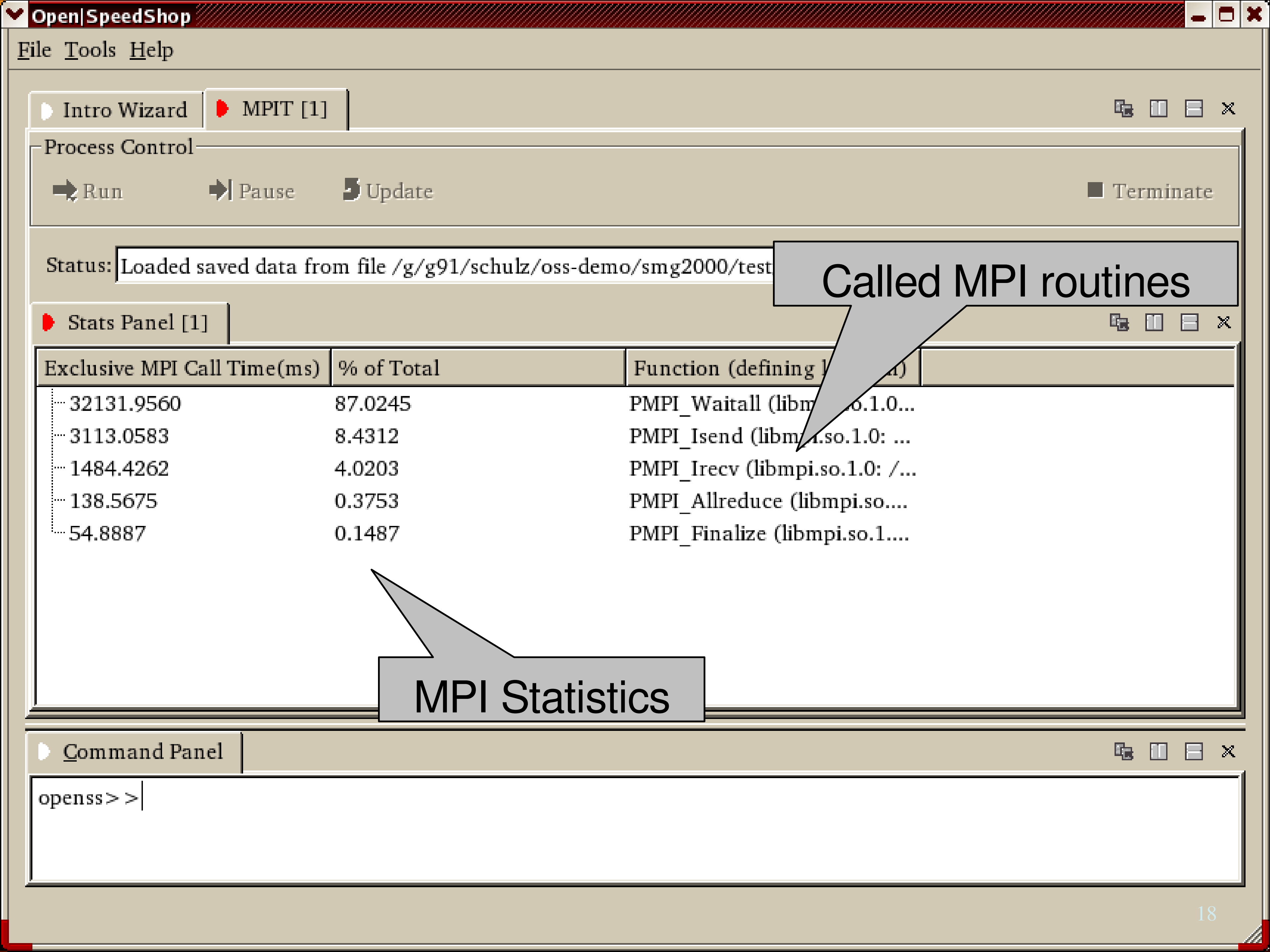
Source window

Statements with high execution times

Per line/statement statistics

Parallel Performance Analysis

- Open|SpeedShop supports MPI and Multithreading
  - MPI Process control using MPIR interface
  - Works with multiple MPI implementations
  - Currently: *mpich, openmpi, lampi, lam, slurm, mpt*
  - Attach to running appl. or create appl. within O|SS
- Parallel Experiments
  - Apply sequential collectors to all nodes
  - Specialized MPI tracing experiments
- Results
  - By default results are aggregated
  - Optional: select individual processes
  - Compare or group ranks



Status: Loaded saved data from file /g/g91/schulz/oss-demo/smg2000/test

Called MPI routines

Exclusive MPI Call Time(ms)	% of Total	Function (defining l n)
32131.9560	87.0245	PMPI_Waitall (libm o.1.0...
3113.0583	8.4312	PMPI_Isend (libm i.so.1.0: ...
1484.4262	4.0203	PMPI_Irecv (libmpi.so.1.0: /...
138.5675	0.3753	PMPI_Allreduce (libmpi.so....
54.8887	0.1487	PMPI_Finalize (libmpi.so.1....

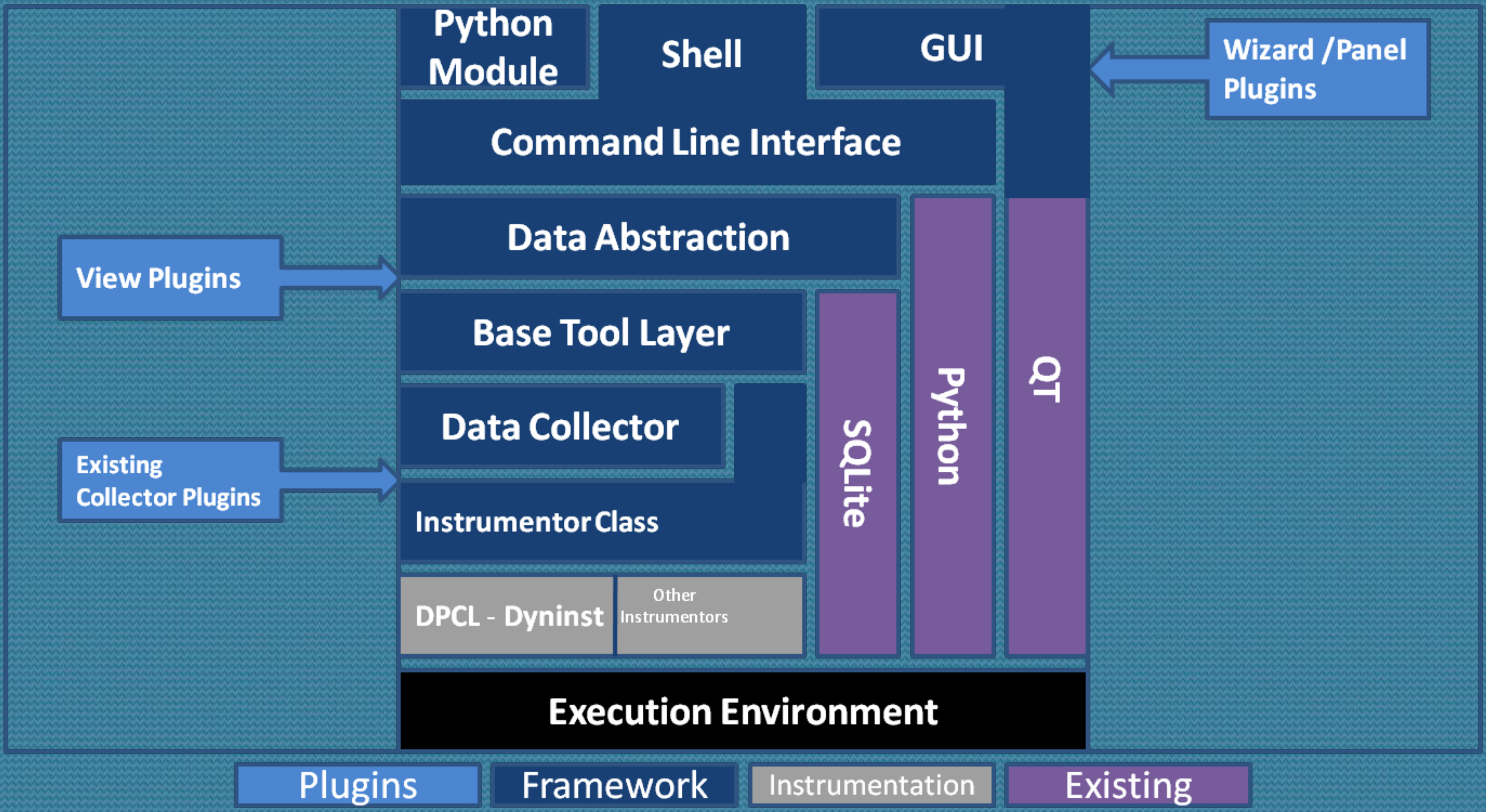
MPI Statistics

```
openss> >
```

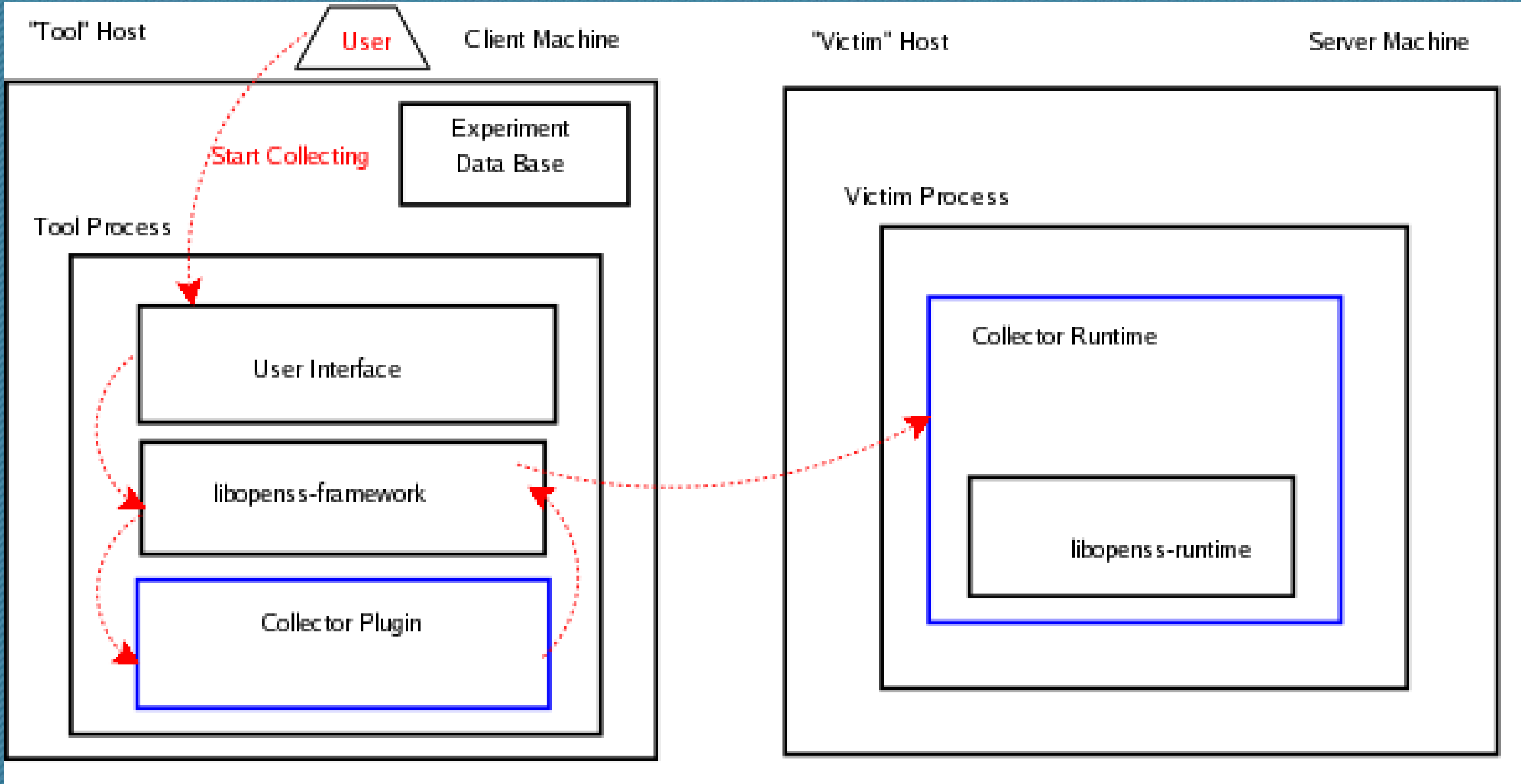
Advanced Capabilities

- Stack trace views
  - Included in tracing and user time experiments
  - Visualize as call-tree and trace-back
- Experiment and Rank/Process/Thread Comparisons
- View results by Time segments
- Multi-rank analysis
  - Restrict results to task sets
  - Compare tasks or task sets
  - Cluster Analysis (grouping similar processes)

Open|SpeedShop Architecture



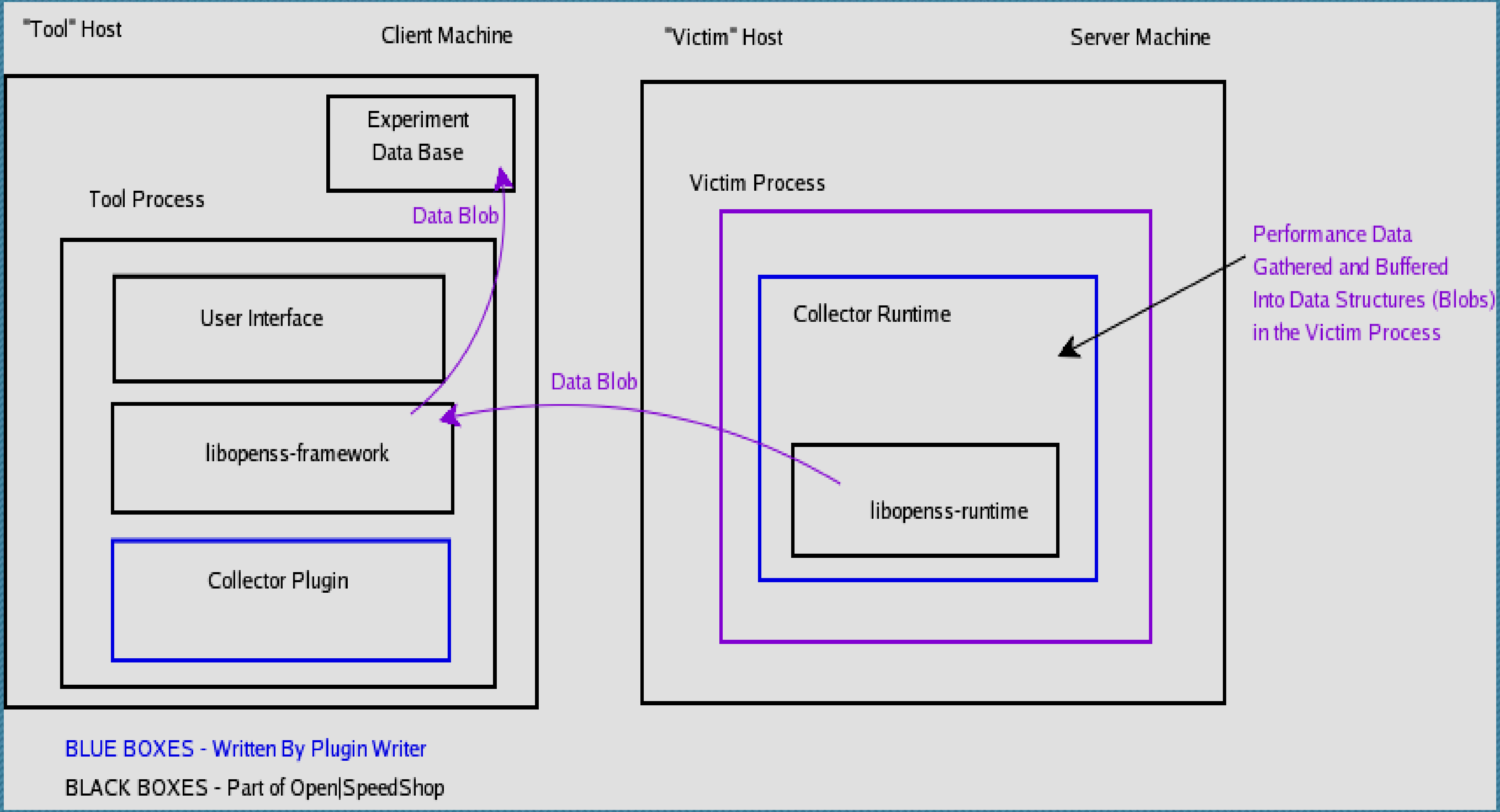
Open|SpeedShop High Level Interactions



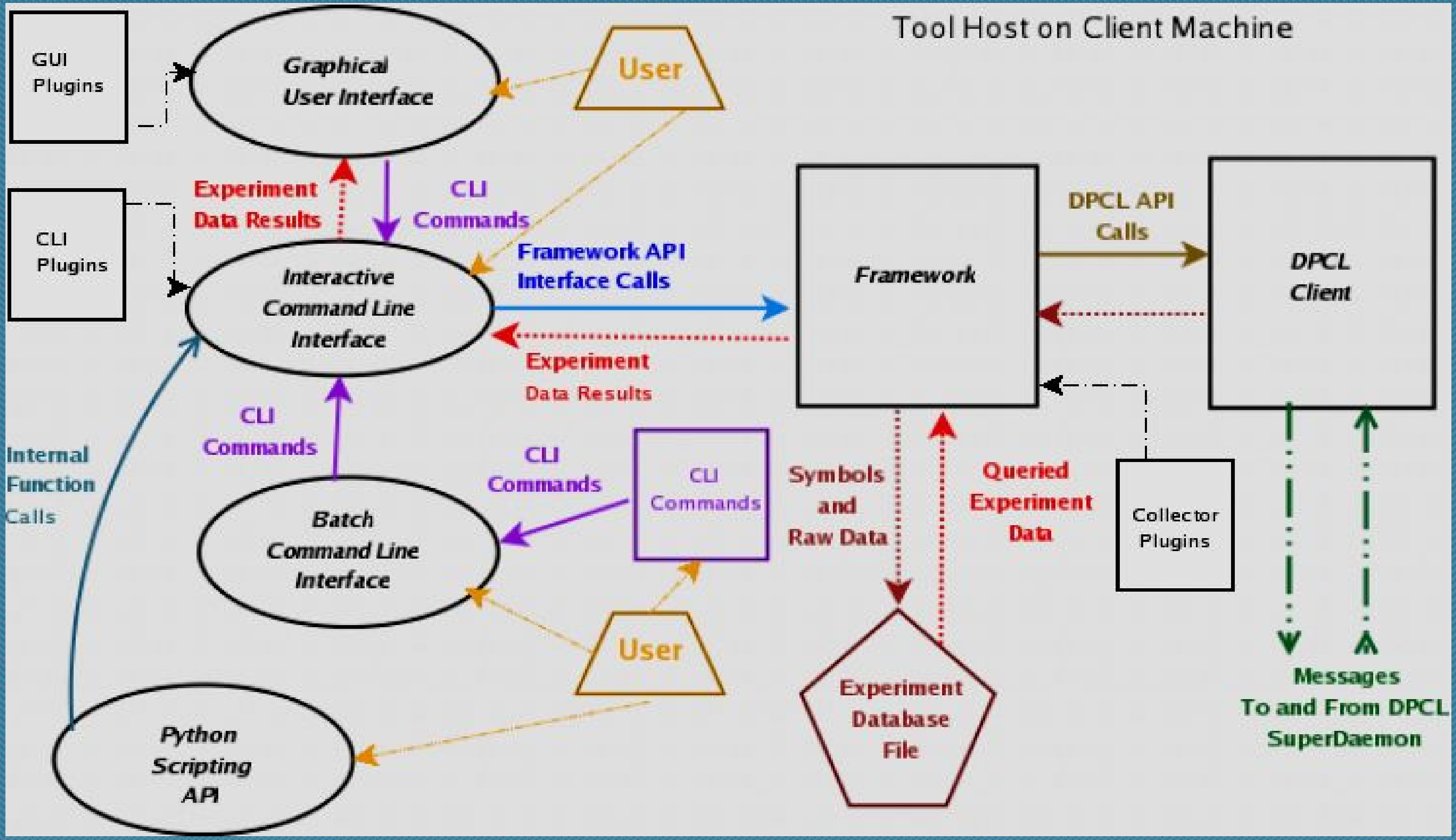
BLUE BOXES - Written By Plugin Writer

BLACK BOXES - Part of Open|SpeedShop

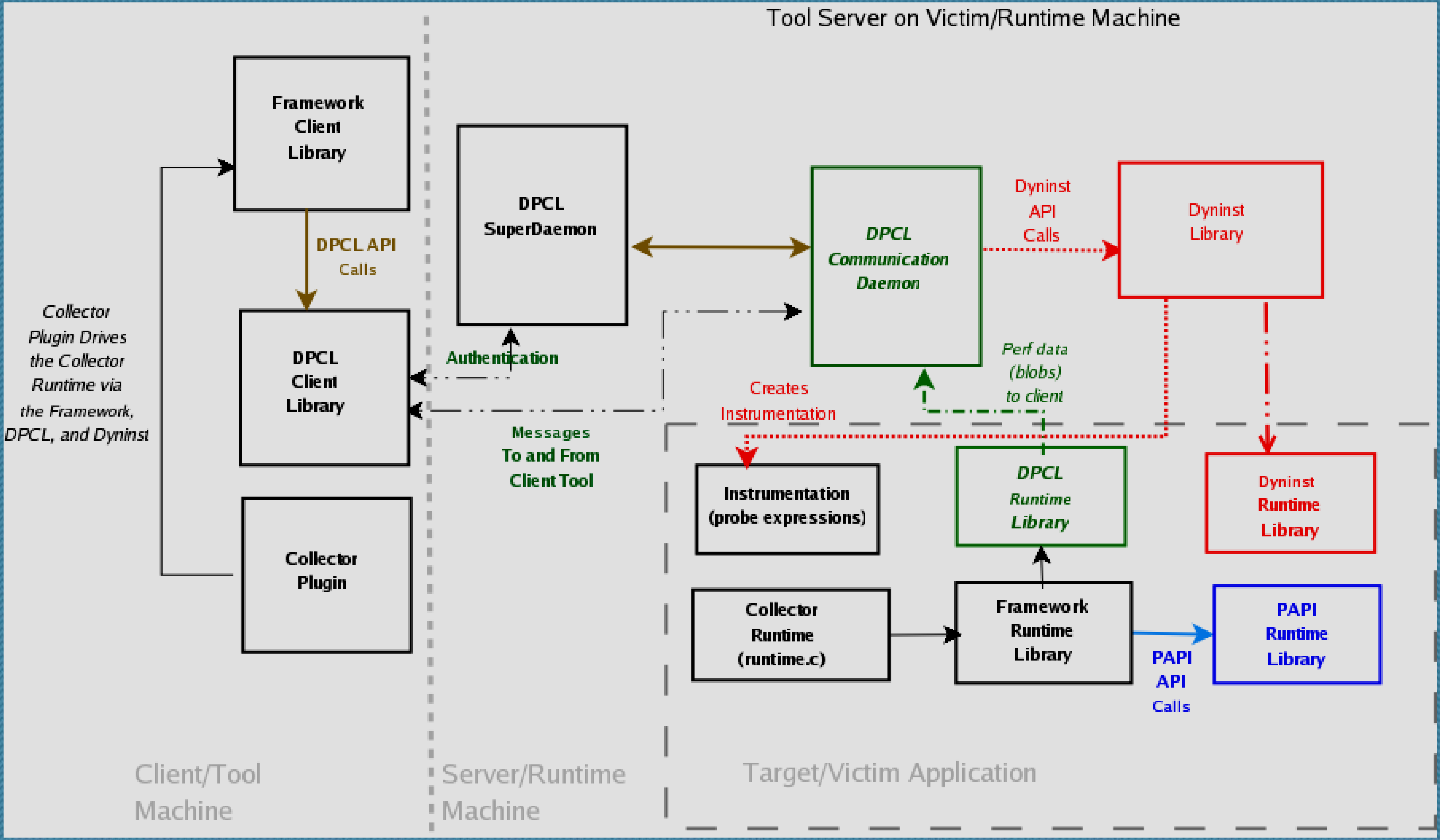
Open|SpeedShop High Level Interactions



Open|SpeedShop Client Architecture



Open|SpeedShop Server Architecture


Tool Server on Victim/Runtime Machine
Framework Client Library
DPCL API Calls
DPCL Client Library
Collector Plugin
Collector Plugin Drives the Collector Runtime via the Framework, DPCL, and Dyninst
DPCL SuperDaemon
Authentication
Messages To and From Client Tool
DPCL Communication Daemon
Dyninst API Calls
Dyninst Library
Perf data (blobs) to client
Creates Instrumentation
Instrumentation (probe expressions)
DPCL Runtime Library
Dyninst Runtime Library
Collector Runtime (runtime.c)
Framework Runtime Library
PAPI API Calls
PAPI Runtime Library
Client/Tool Machine
Server/Runtime Machine
Target/Victim Application

Dyninst Component in Open|SpeedShop

At the node level:

- Obtain and Process Application Symbols
- Attach to a running process
- Insert Code into Application Dynamically
  - Execute at Entry and Exit
  - Execute Now
  - Execute In Place of
- Control the Process/Application (start, stop, …)
- Offline collectors will use symtabAPI component

DPCL Component

Across nodes:

- Connect to application on each node
- Execute Dyninst functions on each remote node
- Use DPCL daemons to return gathered data to the client

Framework Component

Key Component for Open|SpeedShop

- Multi-threaded to support server/client requests
- Interface with the Instrumentor (DPCL, MRNet, other)
    - Insert instrumentation, start/stop collecting
    - Retrieve and store application symbol table information
- Receive performance data from runtime
- Create and manage Open|SpeedShop database
- Provide User Interface with data for display

Plugin Components

- Types of Plugins
  - View, GUI panels, Collector
- All default experiments use plugin mechanism
- Collector Plugins
  - Client and Runtime plugin for each collector
  - Runtime: what performance data to gather
  - Runtime: inserted into application for gathering
  - Client: how to view the data, start/stop gathering

Plugin Components

- GUI plugins use CLI commands to interface
  - All commands go through a single interface
    - Including Python Scripting Interface
  - Ensures equal functionality and robustness
  - Enables easier debugging
  - Have GUI history by using command history tracking in the CLI
  - Key functionality that will enable GUI separation, if desired

Other External Components

- xdr
  - Encode data for transfer between runtime and client.
  - Takes care of endianness issues.
- python
  - Scripting API language
- SQLite
  - Performance database storage, queries
- MPIR interface
  - Retrieving the list of MPI ranked processes
- libmonitor for offline collectors
  - Trap dsos, start gathering, stop gathering, callbacks

Peta-Scale support

Data Collection and Transport

- Replace DPCL with MRNet for distributed communication, control, and monitoring
- Change the existing Instrumentor API to be process group (thread group) centric
- Create MRNet instrumentor
- Define Tool/Daemon Protocol (tool via MRNet to application on nodes)

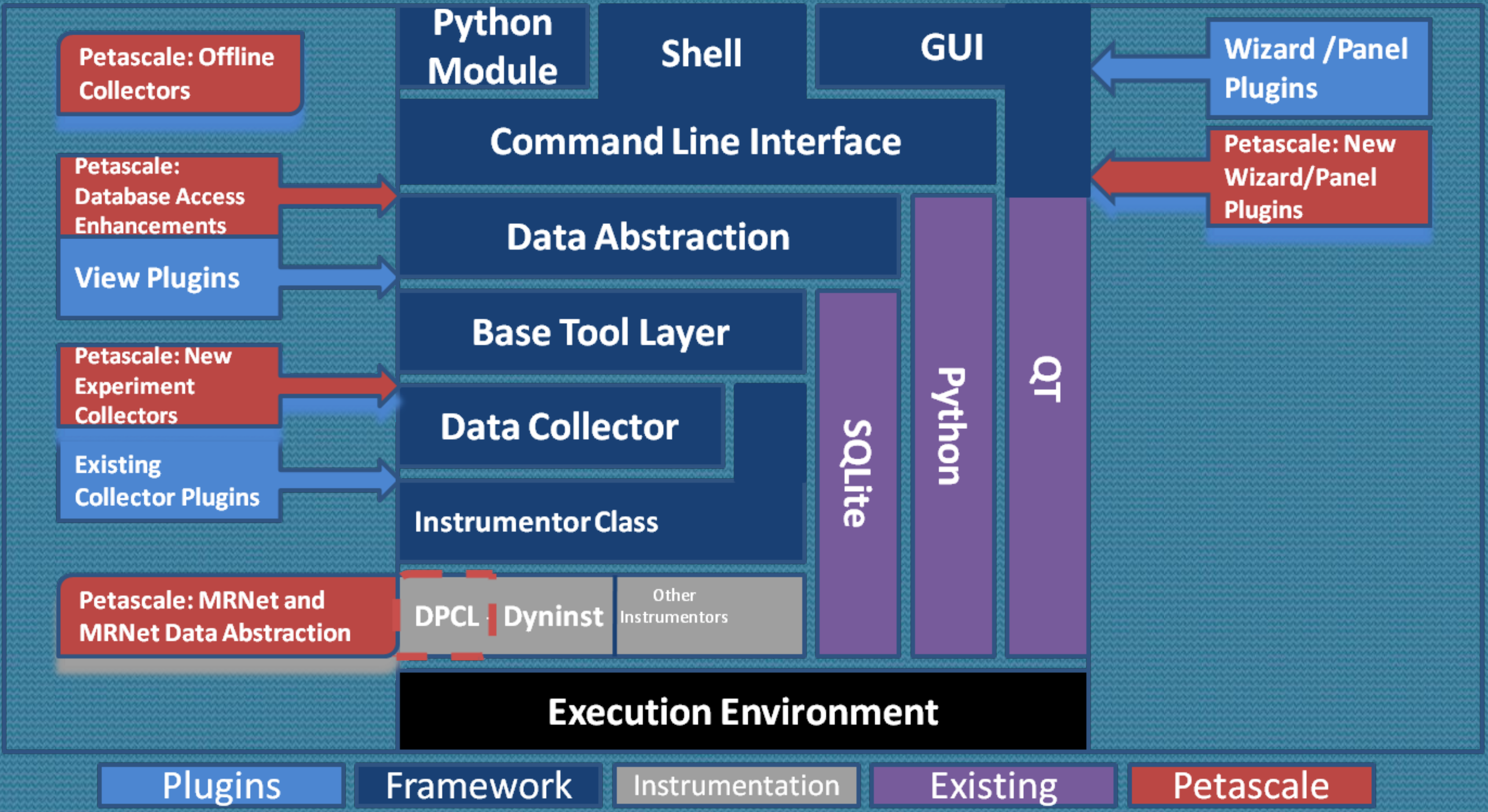
Peta-Scale support

- Result storage, aggregation, and analysis
  - Use of MRNet to gather and analyze perf data
    - Filter data, use intermediate database based on bandwidth available
    - Use extended cluster analysis techniques, apply to database to reduce amount of data stored
    - Create additional wizards to guide user

    Use filter plugins to aggregate data

- Offline Collectors
  - Execute experiments without tool backend
  - Target for microkernel architectures

Open|SpeedShop
Petascale Architecture



MRNet Component

PetaScale Open|SpeedShop

Across nodes:

- Execute Dyninst functions on each remote node
- Use tree structure to return gathered data to the client
- Use filters within the tree structure reduce the gathered data on it's way to the client

Offline Collectors

PetaScale Open|SpeedShop

Alternative method of gathering performance data:

- Targets micro kernel architecture
  - Available in general, but targets platforms where Dyninst support is not available.
- Static application support
  - Requires relinking application with static collector runtime libraries
- Dynamic application support
  - Use LD_PRELOAD to link runtime library to application
  - Leverage libmonitor for dynamic support

Offline Collectors

PetaScale Open|SpeedShop

Alternative method of gathering perf data:

- Offline data written in simple “raw” format
  - Separate tool to convert into native database file format for standard viewing/storage.
  - Eventually Open|SpeedShop client will also do conversion
- Reuse existing collector runtimes where possible.
  - Have run VampirTrace as part of mpiotf offline collector
  - Same collector shared by Open|SpeedShop base tool

Other Future Plans

- Port Open|SpeedShop to other platforms
- Usability improvements from previous usability studies
- New experiments
  - Code coverage plugin – Javelina
  - mpiP
  - Memory tracing

Summary

- Support for wide range of experiments
  - Sampling (timing and hardware counters)
  - Tracing (MPI, I/O, FPE)
- Easy and flexible user access
  - GUI with Wizards
  - Scripting and batch processing
- Plugin infrastructure to extend functionality
- Set of Performance Tools with a flexible framework for additional tool creation

Availability and Contact

Open|SpeedShop website:

http://www.openspeedshop.org/

Feedback

- Bug tracking available from website
- Contact information on website
- Email: oss-questions@openspeedshop.org

Questions?

Jim Galarowicz

jeg@krellinst.org

Krell Institute

http://www.krellinst.org