Open | SpeedShop™

COMPONENT BASED TOOL FRAMEWORK: CBTF

Status of Krell Tools Built using Dyninst/MRNet

Paradyn Week 2013

Madison, Wisconsin

April 30, 2013

LLNL-PRES-503431

Presenters

- **Jim Galarowicz, Krell**
- **Don Maghrak, Krell**

- **Larger team**
  - William Hachfeld, Dave Whitney, Dane Gardner: Krell
  - Martin Schulz, Matt Legendre, Chris Chambreau: LLNL
  - Jennifer Green, David Montoya, Mike Mason, Phil Romero: LANL
  - Mahesh Rajan, Anthony Agelastos: SNLs
  - Dyninst group:
    - Bart Miller, UW and team
    - Jeff Hollingsworth, UMD and team
  - Phil Roth, Michael Brim: ORNL

Outline

- ❖ **Welcome**
- ① **Open|SpeedShop overview and status**
- ② **Component Based Tool Framework overview and status**
- ③ **SWAT (Scalable Targeted Debugger for Scientific and Commercial Computing) DOE STTR Project Status**
- ④ **GPU Support DOE SBIR Project Status**
- ⑤ **Cache Memory Analysis DOE STTR Project Status**
- ⑥ **Parallel GUI Tool Framework DOE SBIR Project Status**
- ❖ **Questions**

Open | SpeedShop™

COMPONENT BASED TOOL FRAMEWORK: CBTF

Open|SpeedShop

(www.openspeedshop.org)

Paradyn Week 2013

April 20, 2013

Project Overview: What is Open|SpeedShop?

❖ What is Open|SpeedShop?

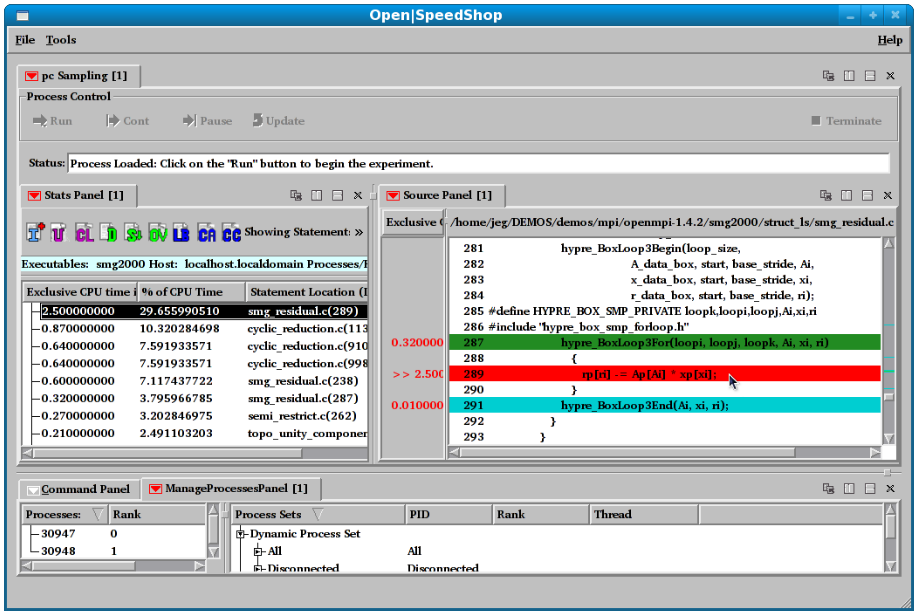
- HPC Linux, platform independent application performance tool
- Linux clusters, Cray, Blue Gene platforms supported

❖ What can Open|SpeedShop do for the user?

- **pcsamp**: Give lightweight overview of where program spends time
- **usertime**: Find hot call paths in user program and libraries
- **hwc,hwctime,hwcsamp**: Give access to hardware counter event information
- **io,iot**: Record calls to POSIX I/O functions, give timing, call paths, and optional info like: bytes read, file names...
- **mpi,mpit**: Record calls to MPI functions. give timing, call paths, and optional info like: source, destination ranks, ...
- **fpe**: Help pinpoint numerical problem areas by tracking FPE

Project Overview: What is Open|SpeedShop?

❖ **Maps the performance information back to the source and displays source annotated with the performance information.**




```
>openss -cli -f smg2000-pcsamp.openss
openss>>Welcome to OpenSpeedShop 2.0.2
openss>>expview
```

Exclusive CPU time in seconds.	% of CPU Time	Function (defining location)
3.630000000	43.060498221	hypre_SMGResidual
2.860000000	33.926453144	hypre_CyclicReduction
0.280000000	3.321470937	hypre_SemiRestrict
0.210000000	2.491103203	hypre_SemiInterp
0.150000000	1.779359431	opal_progress

❖ **osspcsamp** "How you run your application outside of O|SS"

❖ **openss –f** smg2000-pcsamp.openss for GUI

❖ **openss –cli –f** smg2000-pcsamp.openss for CLI (command line)

❖ Update on status of Open|SpeedShop

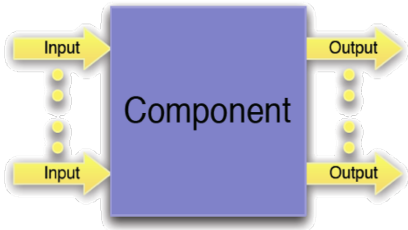
- Continued to focus more on CBTF the past year
- Completed port to Blue Gene Q
  - Static executables using osslink
  - Dynamic (shared) executable using osspcsamp, ossusertime, etc.
- Added functionality to Open|SpeedShop
  - Added MPI File I/O support to MPI experiment.
  - Keeping up with components like: libunwind, papi, dyninst, libmonitor...
  - Derived metric support: arithmetic on gathered performance metrics
  - More platforms, users & application exposure -> more robust
- New CBTF component instrumentor for data collection
  - Leverages lightweight MRNet for scalable data gathering and filtering.
  - Uses CBTF collectors and runtimes
  - Passes data up the transport mechanism, based on MRNet
  - Provides basic filtering capabilities currently

Future Experiments by End of 2013

- **New Open|SpeedShop experiments under construction**
  - **Lightweight I/O experiment (iop)**
    - Profile I/O functions by recording individual call paths
      - Rather than every individual event with the event call path, (**io** and **iot**).
      - More opportunity for aggregation and smaller database files
    - Map performance information back to the application source code.
  - **Memory analysis experiment (mem)**
    - Record and track memory consumption information
      - How much memory was used – high water mark
      - Map performance information back to the application source code
  - **Threading analysis experiment (thread)**
    - Report statistics about pthread wait times
    - Report OpenMP (OMP) blocking times
    - Attribute gathered performance information to proper threads
    - Thread identification improvements
      - Use a simple integer alias for POSIX thread identifier
    - Report synchronization overhead mapped to proper thread
    - Map performance information back to the application source code

Scaling Open|SpeedShop

- Open|SpeedShop designed for traditional clusters
  - Tested and works well up to 1,000-10,000 cores
  - Scalability concerns on machines with 100,000+ cores
  - Target: ASC capability machines like LLNL's Sequoia (20 Pflop/s BG/Q)
- Component Based Tool Framework (CBTF)
  - http://ft.ornl.gov/doku/cbtfw/start
  - Based on tree based communication infrastructure
  - Porting O|SS on top of CBTF



- Improvements:
  - Direct streaming of performance data to tool without writing temporary raw data I/O files
  - Data will be filtered (reduced or combined) on the fly
  - Emphasis on scalable analysis techniques
- Initial prototype exists, working version: Mid-2013
  - Little changes for users of Open|SpeedShop
  - CBTF can be used to quickly create new tools
  - Additional option: use of CBTF in applications to collect data

UW/UMD Software - Open|SpeedShop

- **What UW/UMD software is used in Open|SpeedShop?**
  - **symtabAPI**
    - For symbol resolution on all platforms
  - **instructionAPI, parseAPI**
    - For loop recognition and details
      - This work is in progress
  - **dyninstAPI**
    - For dynamic instrumentation and binary rewriting
      - Includes the subcomponents that comprise "Dyninst".
      - Inserts performance info gathering collectors and runtimes into the application.
  - **MRNet –**
    - Transfer data from application level to the tool client level.
    - Filtering of performance data on the way up the tree.
- **Keeping up with the releases and pre-release testing**
  - At release level 8.1.1

Open | SpeedShop™

COMPONENT BASED TOOL FRAMEWORK: CBTF

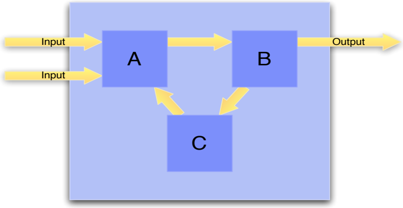
Component Based Tool Framework (CBTF)

Paradyn Week 2013

April 20, 2013

CBTF

- **What is CBTF?**
  - A Framework for writing Tools that are Based on Components.
  - Consists of:
    - Libraries that support the creation of reusable components, component networks (single node and distributed) and support connection of the networks.
    - Tool building libraries (decomposed from O|SS)



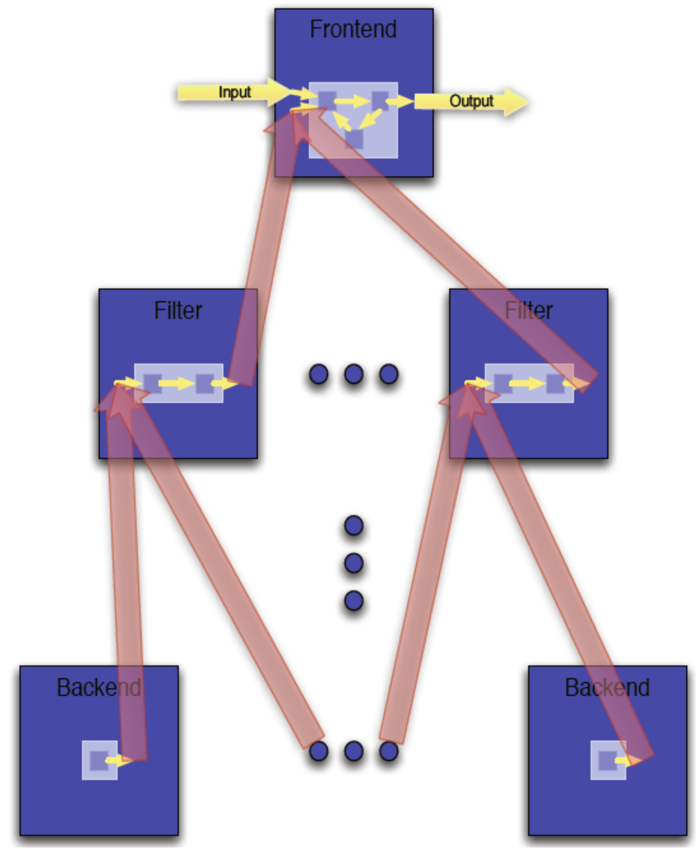
- **Benefits of CBTF**
  - Components are reusable and easily added to new tools.
  - With a large component repository new tools can be written quickly with little code.
  - Create scalable tools by virtue of the distributed network based on MRNet.
  - Components can be shared with other projects

MRNet

- **CBTF uses a transport mechanism to handle all of its communications.**
- **CBTF uses MRNet as its transport mechanism**
  - Multicast/Reduction Network
  - Scalable tree structure
  - Hierarchical on-line data aggregation
- **CBTF views MRNet as "just" another component.**

CBTF Networks

- **Three Networks where components can be connected**
  - Frontend, Backend, multiple Filter levels
  - Every level is homogeneous
- **Each Network also has some number of inputs and outputs.**
- **Any component network can be run on any level, but logically**
  - Frontend component network
    - Interact with or Display info to the user
  - Filter component network
    - Filter or Aggregate info from below
    - Make decisions about what is sent up or down the tree
  - Backend component network
    - Real work of the tool (extracting information)



Using CBTF Beyond O|SS

- **What can this framework be used for?**
- **CBTF is flexible and general enough**
  - To be used for any tool that needs to "do something" on a large number of nodes and filter or collect the results.
- **Sysadmin Tools**
  - Poll information on a large number of nodes
  - Run commands or manipulate files on the backends
  - Make decisions at the filter level to reduce output or interaction
- **Performance Analysis Tools**
  - Massively parallel applications need scalable tools
  - Have components running along side the application
- **Debugging Tools**
  - Use cluster analysis to reduce thousands (or more) processes into a small number of groups

CBTF Related and Next Steps

- **Tool startup investigations (Libi, launchmon)**
- **Continuing porting to Cray and Blue Gene**
  - Cray
    - Working, but needs some further automation for node allocation
  - Blue Gene
    - Delayed, because lightweight MRNet does not currently work on BG/Q
    - Investigation with Matt Legendre, LLNL, on an alternative way to transfer performance information from the application to the CBTF/OSS tool.
- **Add more advanced data reduction filters**
  - Cluster analysis
  - Data matching techniques: keep a representative rank/thread
- **Full Open|SpeedShop integration**
- **Completed Phase I DOE SBIR to research and add performance analysis support for GPU/Accelerators**

Open | SpeedShop™

COMPONENT BASED TOOL FRAMEWORK: CBTF

Scalable Targeted Debugger for Scientific and Commercial Computing (SWAT) STTR Project

Paradyn Week 2013

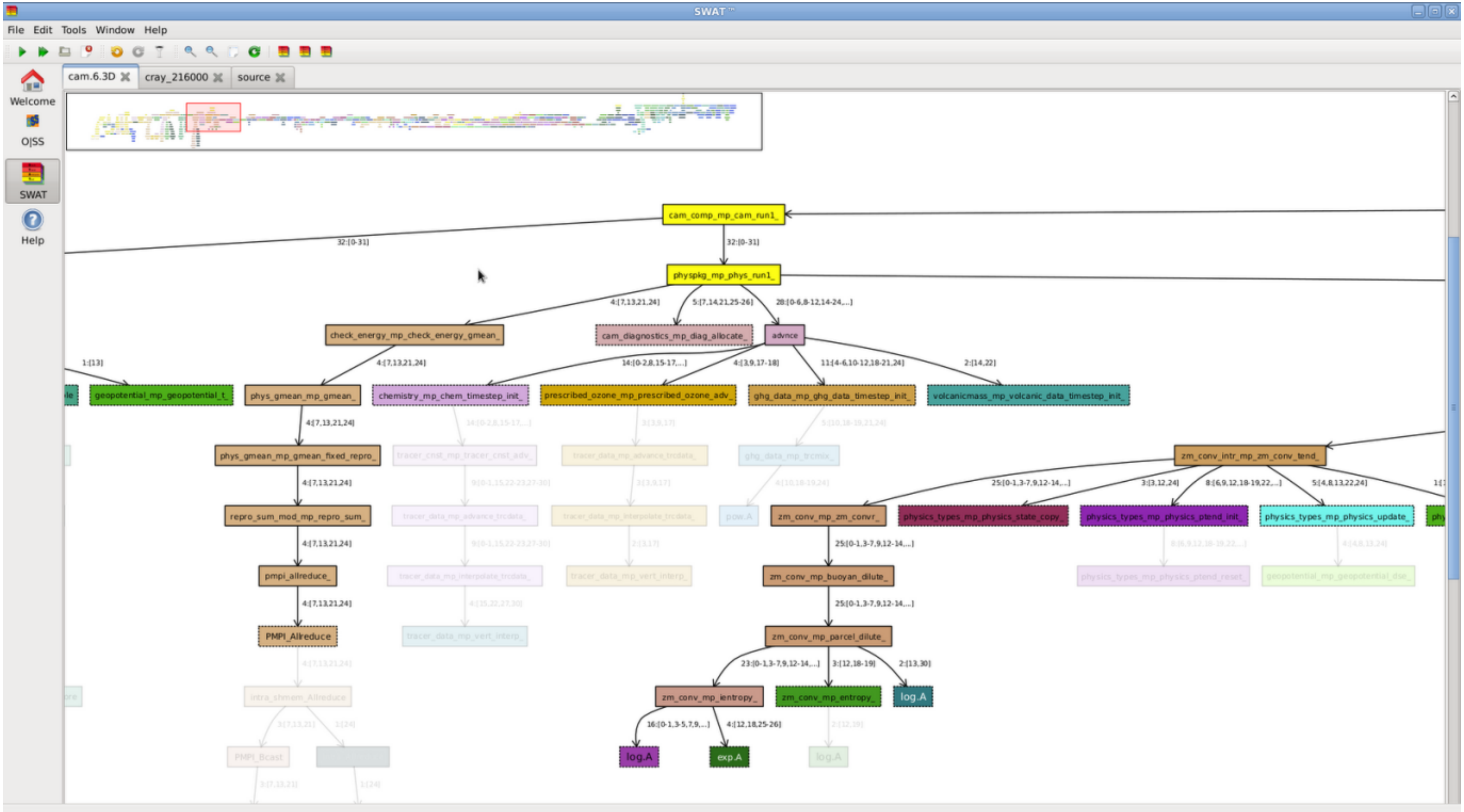
April 20, 2013

SWAT

- **What is SWAT?**
  - A Commercialized version of the STAT debugger primarily developed by LLNL/UW
  - Attach to a hung job, find all call paths and expose the outliers.
- **UW and Argo Navis* teamed together on STTR to:**
  - Port SWAT to more platforms
  - Test and extend the stack walking component used by SWAT, the StackwalkerAPI to work with more compilers, platforms, …
    - This was done
  - Enhance the GUI so that it is portable, robust, and easy to use.
    - New GUI was written based on the Parallel Tools GUI Framework (PTGF)
  - Develop more advanced call tree reduction algorithms
  - Improve SWAT's ability to display complex stack trees
- **Uses StackWalkerAPI and MRNet**
- **Looking for new funding and marketing opportunities for SWAT.**

***Commercial entity associated with Krell**

SWAT



SWAT™
File Edit Tools Window Help
cam.6.3D
cray_216000
source
Welcome
O|SS
SWAT
Help
cam_comp_mp_cam_run1_
32:[0-31]
32:[0-31]
physpkg_mp_phys_run1_
4:[7,13,21,24]
5:[7,14,21,25-26]
28:[0-6,8-12,14-24,...]
check_energy_mp_check_energy_gmean_
cam_diagnostics_mp_diag_allocate_
advnce
1:[13]
4:[7,13,21,24]
14:[0-2,8,15-17,...]
4:[3,9,17-18]
11:[4-6,10-12,18-21,24]
2:[14,22]
geopotential_mp_geopotential_t_
phys_gmean_mp_gmean_
chemistry_mp_chem_timestep_init_
prescribed_ozone_mp_prescribed_ozone_adv_
ghg_data_mp_ghg_data_timestep_init_
volcanicmass_mp_volcanic_data_timestep_init_
4:[7,13,21,24]
phys_gmean_mp_gmean_fixed_repro_
zm_conv_intr_mp_zm_conv_tend_
4:[7,13,21,24]
25:[0-1,3-7,9,12-14,...]
3:[3,12,24]
8:[6,9,12,18-19,22,...]
5:[4,8,13,22,24]
repro_sum_mod_mp_repro_sum_
zm_conv_mp_zm_convr_
physics_types_mp_physics_state_copy_
physics_types_mp_physics_ptend_init_
physics_types_mp_physics_update_
4:[7,13,21,24]
25:[0-1,3-7,9,12-14,...]
pmpi_allreduce_
zm_conv_mp_buoyan_dilute_
4:[7,13,21,24]
25:[0-1,3-7,9,12-14,...]
PMPI_Allreduce
zm_conv_mp_parcel_dilute_
23:[0-1,3-7,9,12-14,...]
3:[12,18-19]
2:[13,30]
zm_conv_mp_ientropy_
zm_conv_mp_entropy_
log.A
16:[0-1,3-5,7,9,...]
4:[12,18,25-26]
log.A
exp.A

Open | SpeedShop™

COMPONENT BASED TOOL FRAMEWORK: CBTF

Open|SpeedShop Support GPU SBIR Phase I Project

Paradyn Week 2013

April 20, 2013

GPU support: CBTF & OpenSpeedShop

- **Argo Navis* GPU DOE SBIR phase I**
  - Prototype application profiling support for GPUs into OpenSpeedShop
- **Using the CUDA and PAPI Cupti interfaces**
- **These were the goals we proposed for the GPU SBIR:**
  - Report the time spent in the GPU device (when exited - when entered).
    - Completed
  - Report the cost and size of data transferred to and from the GPU.
    - Completed
  - Report information to help the user understand the balance of CPU versus GPU utilization.
    - Close to completion
  - Report information to help the user understand the balance between
    - The transfer of data between the host and device memory and the execution of computational kernels.
    - Have info to derive this, need to create the views.
  - Report information to help the user understand the performance of the internal computational kernel code running on the GPU device.
    - Close to completion

***Commercial entity associated with Krell**

GPU support: CBTF & OpenSpeedShop

- **Because transitioning Open|SpeedShop to use CBTF to collect performance data.**
  - GPU collection capabilities were added to the CBTF collector set. Makes the functionality available in CBTF as well.
- **Rudimentary views are available.**
  - Info external to GPU displays based on I/O tracing collector view
  - Info internal to GPU displays based on the hwc sampling collector view
- **Current status:**
  - Collection of external GPU kernel statistics is completed
  - Working on gathering information about the GPU kernels themselves.
  - Looking for new funding opportunities for further GPU related development, as we did not win phase II funding.
    - CLI and GUI view work needed.

***Commercial entity associated with Krell**

Open | SpeedShop™

COMPONENT BASED TOOL FRAMEWORK: CBTF

Cache Memory Analysis STTR Phase I Project (active)

Paradyn Week 2013

April 20, 2013

Automated Cache Performance Analysis and Optimization in Open|SpeedShop

- **Teamed with Kathryn Mohror and Barry Roundtree, LLNL**
- **Use Precise Event-Based Sampling (PEBS) counters**
- **With the newest iteration of PEBS technology**
  - Cache events can be tied to a tuple of:
    - Instruction pointer
    - Target address (for both loads and stores)
    - Memory hierarchy and observed latency
- **With this information we can analyze Cache usage for:**
  - Efficiency of regions of code
  - How these regions interact with particular data structures
  - How these interactions evolve over time.
- **Short term, research focus:**
  - Performance analysis: understanding and optimizing the behavior of application codes related to their memory hierarchy.
- **Long term, research focus:** Automation

Open | SpeedShop™

COMPONENT BASED TOOL FRAMEWORK: CBTF

Parallel Tools GUI Framework (PTGF) Phase I Project (active)

Paradyn Week 2013

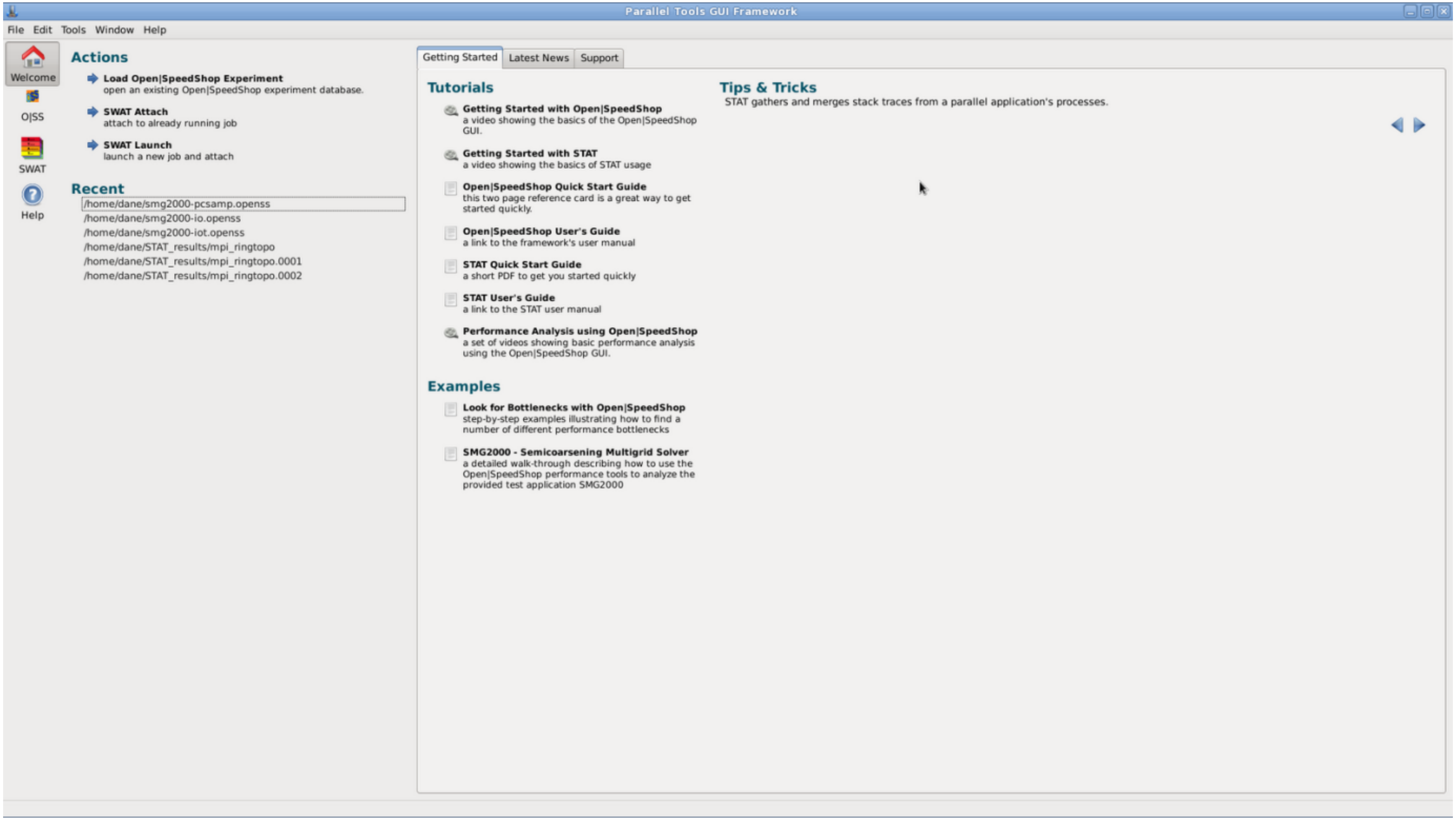
April 20, 2013

Parallel Tools GUI Framework DOE SBIR

Parallel Tools GUI Framework Goals:

- **Facilitate the rapid development of cross-platform user interfaces for new and existing parallel tools.**
- **Target a stable version of Qt4 which is currently available on many existing clusters. It is forward compatible with Qt5.**
- **Provide abstracted visualizations for easy inclusion in multiple parallel tools. These abstracted visualizations will accept a simple dataset.**
  - The visualization plugins will also act as dynamic libraries, which can be easily extended by tool developers looking to specialize a particular view.
- **Provide a scalable design/model which will allow tools with very large datasets to be used effectively within the PTGF.**
- **Provide a standardized interface such that users will find enough similarities between tools to make learning additional ones easier.**
- **Provide facilities for user learning of a new parallel tool from within PTGF, and the ability to link to online resources.**

Parallel Tools GUI Framework DOE SBIR



Questions

- **Jim Galarowicz**
  - jeg@krellinst.org
- **Don Maghrak**
  - dpm@krellinst.org
- **Questions about Open|SpeedShop or CBTF**
  - oss-questions@openspeedshop.org