Krell Institute related tools (OSS, CBTF, SWAT)
Implementation Details, Issues, and Status

CsCADS 2012
Snowbird, Utah
June 26, 2012
Presenters

- Jim Galarowicz, Krell

Larger team

- Don Maghrak, William Hachfeld, Dave Whitney, Dane Gardner: Krell Institute
- Martin Schulz, Matt Legendre, Chris Chambreaau: LLNL
- David Montoya, TJ Machado, Mike Mason, Jennifer Green, Phil Romero: LANL
- Mahesh Rajan: SNLs
- Dyninst group:
  - Bart Miller, UW and team
  - Jeff Hollingsworth, UMD and team
- Phil Roth: ORNL
Outline

❖ Introduction

① Open|SpeedShop overview and status
② PTGF (Parallel Tool GUI Framework)
③ SWAT (Scalable Targeted Debugger for Scientific and Commercial Computing)
④ DOE SBIR Heterogeneous Processor support
⑤ Component Based Tool Framework (overview)
⑥ Component Based Tool Framework (tools)
⑦ Next generation: CBTF

❖ Questions
Open | SpeedShop

Component Based Tool Framework: CBTF

Open | SpeedShop
(www.openspeedshop.org)

CsCADS WorkShop 2012

June 26, 2012
Project Overview: What is Open|SpeedShop?

- **What is Open|SpeedShop?**
  - HPC Linux, platform independent application performance tool
  - Works on dynamic and static executables

- **What can Open|SpeedShop do for the user?**
  - pcsamp: Give lightweight overview of where program spends time
  - userstime: 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

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

- **osspcsamp** “How you run your application outside of O|SS”
Update on status of Open|SpeedShop

- Port Open|SpeedShop to Blue Gene Q
  - Available at ANL on vesta, LLNL on rzuseq, and on IBM Rochester xxx
  - Issues with unwinding. Thanks to Matt Legendre for patches to libunwind.

- More focus on CBTF the past year but, added functionality to O|SS
  - Support for Cray dynamic executables
    - Execute similar to cluster: osspcsamp “how you run your application”
  - Blue Gene P/Q personality support
  - Added more options to compare script (osscompare)
  - Support derived metrics in the CLI through doing arithmetic on perf. data

- Starting DOE SBIR to research and add performance analysis support for GPU/Accelerators
  - Talk more about this in the DOE SBIR portion of the talk

- Adding a CBTF component instrumentor for data collection that leverages Lightweight MRNet for scalable data gathering and filtering.
  - Talk more about this in the CBTF portion of talk
PTGF (Parallel Tools GUI Framework)

CsCADS WorkShop 2012

June 26, 2012
What is PTGF?
- A parallel tools GUI framework developed by Argo Navis* as part of NASA SBIR program written in QT4 toolkit.
- Integrates parallel tools into common interface
  - Multiple tools already exist or are in-the-works

Highlights:
- Support for adding tool plugins
  - Initial ‘welcome’ page
    - Register external web links (tutorials, videos, etc.)
    - Register links to internal help files
    - Register RSS feeds for updated news
  - Register help files for integration into a common interface
  - Register menu actions
- Data views can be used by all tools
- Supports 2D and 3D visualizations
- Annotatable source code viewer
- Client/server for remote GUI operation

*Commercial entity associated with Krell
DOE Small Business Technology Transfer (STTR)

SWAT

CsCADS WorkShop 2012

June 26, 2012
What is SWAT?
- A commercialized version of the STAT debugger primarily developed by LLNL/UW (https://computing.llnl.gov/code/STAT)
- Identify groups of processes in a parallel application that exhibit similar behavior

Components used: StackWalkerAPI, MRNet, and PTGF

UW and Argo Navis* teaming together on STTR to:
- Port SWAT to more platforms
- Improve infrastructure
  - Test and extend StackWalkerAPI to work with more compilers, platforms
  - Develop more advanced call tree reduction algorithms
- Improve interface
  - Enhance the GUI to be more portable, robust, and easy to use
  - Add more support for simplified modes of use
  - Improve SWAT’s ability to display complex stack trees

*Commercial entity associated with Krell
SWAT Early PGTF based GUI View
SWAT Early PGTF based GUI View

*Data used for this rendering courtesy of Greg Lee, LLNL*
DOE Small Business Innovation Research (SBIR)
Heterogeneous Processor support

CsCADS WorkShop 2012
June 26, 2012
DOE SBIR Heterogeneous Processor

- **Project goals**
  - Phase I: Investigate and provide proof of concept for adding heterogeneous processor support into Open|SpeedShop
  - Phase II (if awarded):
    - Argo Navis* commercialize GPU support and OpenSpeedShop
    - Finish proof of concept features into a finished product state

- **Research areas:**
  - Tool to identify loops that might be good GPU kernel candidates
  - Reporting time spent in the GPU device (when exited - when entered)
  - Reporting cost and size of data transferred to and from the GPU
  - Reporting information to help the user understand
    - The balance of CPU versus GPU utilization.
    - The balance the transfer of data between the host and device memory with the execution of computational kernels
    - The performance of the internal computational kernel code running on the GPU device
  - Combining other Open|SpeedShop experiment information with GPU info
Implementation:

- Because we are transitioning to a new scalable tool back-end, we will do all of the accelerator data collection in our Component Based Tool Framework (CBTF) source tree.

Discussions this week:

- Interested in talking with people about accelerator issues.
  - MIC, NVIDIA, other forms of accelerators and gathering performance information
  - Loop analysis techniques for detecting GPU kernel candidate loops
Component Based Tool Framework (CBTF)

CsCADS WorkShop 2012

June 26, 2012
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)
    - Support connection of the networks.
  - Tool building libraries (partially derived 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 a distributed network based on MRNet.
- Components can be shared with other projects
CBTF: Base CBTF Libraries

- Create components, component networks, distributed component networks

Diagram:
- cbtf
  - framework
    - libcbtf
    - libcbtf-xml
    - libcbtf-mrnet
  - tools
    - core
    - messages
    - services
    - examples
      - pcsampDemo
        - Using LW MRNet BE
      - daemonToolDemo
        - Using MRNet
  - contrib
Main concepts: Components

- **Data-Flow Model**
  - Accepts Inputs
  - Performs Processing
  - Emits Outputs

- **Reusable objects with 0-N inputs and 0-M outputs.**

- **Designed to be connected together**
  - Connections defined in C++ or XML file.

- **Components are written in C++**
  - Components can do anything your C++ code can do.
  - Run system commands, open files, do calculations.
CBTF: Component Networks

- **Components**
  - Specific Versions

- **Connections**
  - Matching Input/Output Data Types

- **Arbitrary Component Topology**
  - Pipelines
  - Graphs with cycles
  - ....

- **Recursive**
  - CBTF Component Network is itself a component.

- **XML-Specified Connections**
  - Declare the component connections defining a component network
  - Component version
  - Input/Output types and names
Simple way to build the CBTF networks and connect the components.

- No need to recompile (if the available components provide the capabilities needed).

```xml
<Network xmlns=http://www.krellinst.org/CBTF/Network>
  <Type>TestXML</Type>
  <Version>1.2.3</Version>
  <Plugin>plugin-xml.so</Plugin>

  <Component>
    <Name>Stage1</Name>
    <Type>Doubler</Type>
  </Component>

  <Component>
    <Name>Stage2</Name>
    <Type>Incrementer</Type>
  </Component>

  <Component>
    <Name>Stage3</Name>
    <Type>Doubler</Type>
    <Version min="0.0.1" max="0.0.5"/>
  </Component>

  <Input>
    <Name>in</Name>
    <To>
      <Name>Stage1</Name>
      <Input>in</Input>
    </To>
  </Input>

  <Connection>
    <From>
      <Name>Stage1</Name>
      <Output>out</Output>
    </From>
    <To>
      <Name>Stage2</Name>
      <Input>in</Input>
    </To>
  </Connection>

  <Connection>
    <From>
      <Name>Stage2</Name>
      <Output>out</Output>
    </From>
    <To>
      <Name>Stage3</Name>
      <Input>in</Input>
    </To>
  </Connection>

  <Output>
    <Name>out</Name>
    <From>
      <Name>Stage3</Name>
      <Output>out</Output>
    </From>
  </Output>

</Network>
```
CBTF uses a transport mechanism to handle all of its communications.

Currently that transport mechanism is MRNet
- Multicast/Reduction Network
- Scalable tree structure
- Hierarchical on-line data aggregation

CBTF views MRNet as just another component.
- In the future it could be swapped with some other transport mechanism, if desired.
CBTF Networks

- Three Networks where components can be connected
  - Frontend, Backend, Multiple communication process 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
  - Communication Process 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)
CBTF: Tool Building Support

- To enable tool builders to get started
  - Add a tool building side to CBTF (tools subdirectory under cbtf)

Note:
- daemonToolDemo doesn't rely on any service, message, or core “tools” code
- collectionTool is a test tool that can be used to test the collectors
CBTF: Software Stack (Tools)

- **Open|SpeedShop**
  - Using Services, Messages, Core built using CBTF infrastructure
  - Create CBTF instrumentor class in O|SS to interface with CBTF
  - Full fledged multipurpose performance tool
  - Scalability.....
    - Allow filtering of performance data as it moves from the application to the client tool
    - Eliminates the current method which writes temporary files to disk.
      - New method does not write files

- **Customized Tools**
  - Use the CBTF infrastructure, not necessarily any support from the tools support sub-directories
  - If tool creator sees a useful service in tools, they can choose to use it (along with any message and/or core library)
  - Aimed at specific tool needs determined by application code teams
Using CBTF Beyond O|SS

- **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 and Debugger Tools**
  - Massively parallel applications need scalable tools
  - Have components running along side the application
  - Use cluster analysis to reduce thousands (or more) processes into a small number of groups

- **Customized Tools in development:**
  - **LANL**
    - Sysadmin Tools
      - psTool – run commands on BE processes and transmit, filter on way to client
      - Tbon-FS – perform group file operations
    - Memory analysis tool – displays memory usage information
    - Debugging – stack trace gathering and like trace grouping
    - GPU double bit monitoring tool
  - **ORNL**
    - GPU monitoring tool
  - **UMD**
    - Active Harmony integration
CBTF Next Steps

Next steps related to CBTF development

- **Support**: Active Harmony integration
- **Support**: Other CBTF tool developments
- **Provide**: More detailed documentation of examples, demo tools
- **Investigate and integrate**: tool start up (launchmon, libi, ...)
  - Variations dependent on platform type (BG, Cray)
- **Continue**: Adding tool services, messages, component creation to support more types of collection
  - Not all O|SS collectors have been converted to cbtf/tools
- **Continue**: Porting to Cray and Blue Gene platforms
- **Develop**: More filtering components for MRNet communication node deployment
- **Continue**: The effort to integrate/connect CBTF to O|SS
CBTF Related Discussion Items

Discussions this week:

- Tool start-up and node allocation issues related to running with MRNet
  - Need extra nodes to not impact the application execution
    - Or co-locate communication processes with application
  - Can we get support from the system administrators when users request node allocations?
  - Automatic topology generation

- Installation of needed packages for tools. Can we as a community create a development tools root that would contain packages we all could use for building our tools?

- If any interest, discussions with people about using CBTF to create tools
CBTF Information

- Where to find information
  - CBTF wiki: [http://ft.ornl.gov/doku/cbtfw/start](http://ft.ornl.gov/doku/cbtfw/start)

- Source Access
  - Source hosted at ORNL git repository
  - In process of opening up source – need to move repository to do that

- CBTF Tutorial, Step by Step Instructional Info on CBTF wiki

- LANL: CBTF user guide (in progress)

- Always looking to collaborate with others, please contact us
Questions

- Jim Galarowicz
  - jeg@krellinst.org

- Don Maghrak
  - dpm@krellinst.org

- Questions about Open|SpeedShop or CBTF
  - oss-questions@openspeedshop.org