Status of Krell Tools Built using Dyninst/MRNet

Paradyn Week 2012
College Park, MD
March 28, 2012
Presenters

- Jim Galarowicz, Krell
- Don Maghrak, Krell

Larger team

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

① Open|SpeedShop overview and status
② SWAT (Scalable Targeted Debugger for Scientific and Commercial Computing)
③ Component Based Tool Framework (overview)
④ Component Based Tool Framework (tools)
⑤ Next generation: CBTF

Questions
Open SpeedShop™

Component Based Tool Framework: CBTF

Open SpeedShop
(www.openspeedshop.org)

Paradyn Week 2012

March 28, 2012
Project Overview: What is Open|SpeedShop?

- What is Open|SpeedShop?
  - HPC Linux, platform independent application performance tool

- What can Open|SpeedShop do for the user?
  - Give lightweight overview of where program spends time
  - Find hot call paths in user program and libraries
  - Give access to hardware counter event information
  - Record calls to POSIX I/O functions, give timing, call paths, and optional info like: bytes read, file names...
  - Record calls to MPI functions. give timing, call paths, and optional info like: source, destination ranks, ..... 
  - 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

- More focus on CBTF the past year, but...
- Added functionality to Open|SpeedShop
  - Improved installation of Cray and Blue Gene versions
  - Support for Cray dynamic executables
    - Execute similar to cluster: osspcsamp “how you run your application”
  - Blue Gene P static executable support
  - Added more options to compare script (osscompare)
  - Support for doing arithmetic on the columns of information to create derived metrics

- Starting DOE SBIR to research and add performance analysis support for GPU/Accelerators
- Porting to Blue Gene Q
- 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
Currently:

- **What UW/UMD software is used in Open|SpeedShop?**
  - `symtabAPI` – for symbol resolution on the Blue Gene/Q
  - `dyninstAPI` – for dynamic instrumentation and binary rewriting
    - Includes the subcomponents that comprise “Dyninst”.
    - Inserts performance info gathering collectors and runtimes into the application.
  - `MRNet` – for transfer mechanism

Future:

- **What UW/UMD software will we use in Open|SpeedShop?**
  - Will be using much of the same software but in a different context through CBTF (code in CBTF components and component networks)
SWAT

Paradyn Week 2012

March 28, 2012
What is SWAT?
- A Commercialized version of the STAT debugger primarily developed by LLNL/UW

UW and Argo Navis* teaming together on SBIR to:
- Port SWAT to more platforms
- Test and extend the stack walking component used by SWAT, the StackwalkerAPI to work with more compilers, platforms, ...
- Enhance the GUI so that it is portable, robust, and easy to use. It will support simplified modes of use.
- Develop more advanced call tree reduction algorithms
- Improve SWAT’s ability to display complex stack trees

Uses StackWalkerAPI and MRNet

*Commercial entity associated with Krell
Component Based Tool Framework (CBTF)

Paradyn Week 2012

March 28, 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) 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
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

- 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.

![Component Diagram](image)
CBTF: Component Networks

- **Components**
  - Specific Versions

- **Connections**
  - Matching Types

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

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

- **XML-Specified**
Simple way to build the CBTF networks and connect the components.

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

```xml
<Backend>
  <Network>
    <Type>TestNetwork_Backend</Type>
    <Version>1.0.0</Version>
    <SearchPath>/users/mmason/mycode-rr/psToolGui</SearchPath>
    <Plugin>psPlugin</Plugin>
    <Plugin>mrnetPlugin</Plugin>

    <Component>
      <Name>psCmdComp</Name>
      <Type>psCmd</Type>
    </Component>

    ...<Input>
      <Name>Backend_In</Name>
      <To>
        <Name>PacketToString</Name>
        <Input>in1</Input>
      </To>
    </Input>

    <Connection>
      <From>
        <Name>psCmd</Name>
        <Output>out</Output>
      </From>
      <To>
        <Name>StringToPacket</Name>
        <Input>in2</Input>
      </To>
    </Connection>

    <IncomingDownstream>
      <Name>DownwardStream</Name>
      <To><Input>Backend_In</Input></To>
    </IncomingDownstream>

    <OutgoingUpstream>
      <Name>UpwardStream</Name>
      <From><Output>Backend_Output</Output></From>
    </OutgoingUpstream>

  </Network>
</Backend>
```
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 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)
CBTF: Tool Building Support

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

- Note:
  - The directory structure is subject to change
  - daemonToolDemo doesn’t rely on any service, message, or core “tools” code
CBTF: Software Stack (Future Tools)

- **Open|SpeedShop**
  - Using Services, Messages, Core built using CBTF infrastructure
  - Full fledged multipurpose performance tool

- **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
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
Tool Example: Simple Memory Tool

- memTool (tools/contrib/memTool)
  - Displays memory info for each node and total memory used for each MPI process.
Tool Example: Sysadmin

- **PS Tool (tools/contrib/psTool)**
  - Run `ps` on all backends, use filters to identify the procs running on all nodes and what procs are unique to each node.
CBTF based TBON-FS implementation (tools/contrib/tbonFS)
- Perform group file operations on the backend local file system.
- Run top on all backends.
- Run grep, tail, read or write files on all backends.
- Copy a file from NFS to the local /tmp on all backends.
Stack trace grouping (tools/contrib/stack)

- Run gstack on each pid for a parallel application and group similar stack traces together.
Why do we want to connect CBTF to O|SS?

- Scalability.....
  - Allow filtering of performance data as it moves from the application to the client tool.
  - Eliminates the current “fileio” method which writes temporary files to disk. New method does not write files.

- CBTF hides many details of MRNet coding by providing components that handle the actual MRNet implementation when using MRnet as a transport mechanism

- For O|SS we will use: MRNet, symtabAPI initially, then dyninstAPI (and components) for “online/dynamic” version.
CBTF Next Steps

- Support Active Harmony integration with CBTF
- More detailed documentation of examples, demo tools
- Tool start up investigation/implementation (launchmon, libi, ...)
  - Variations dependent on platform type (BG, Cray)
- Tool services, messages, component creation to support more types of collection (don’t have all O|SS in cbtf/tools)
- Continue porting to Cray and Blue Gene platforms
- Add filtering components for MRNet communication node deployment
- Continue the connection of CBTF to Open|SpeedShop
CBTF Information

❖ Where to find information
  ➢ CBTF wiki: http://ft.ornl.gov/doku/cbtfw/start

❖ Source Access
  ➢ Currently -> Friendly access available through request
    • Probably will be completely open shortly
  ➢ Source hosted at ORNL git repository

❖ CBTF Tutorial, Step by Step Instructional Info on CBTF wiki.

❖ Technical paper is needed…

❖ 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