Open Source Performance Analysis for Linux SSI and Clusters

Paradyn Conference 2007
Jim Galarowicz, Krell Institute
Martin Schulz, LLNL
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
Agenda

- What is Open | SpeedShop?
- Project Highlights/Overview
- Features
- Dyninst Usage in Open | SpeedShop
- Current Status
- Future Plans
- Questions
What is Open|SpeedShop?

- Comprehensive Open Source Parallel Performance Analysis Framework
  - **Goal**: *One tool for all performance analysis needs*
  - Targets **Users and Tool Developers**

- **Funding**
  - DOE/NNSA as part of ASC PathForward
  - Initial phase co-funded by SGI

- **Status**
  - Version 1.0 available as source and RPMs
  - Source code is GPL/LGPL
Partners

• Krell Institute
  – Hosts Development

• DOE/NNSA Tri-Labs
  – Lawrence Livermore
  – Los Alamos
  – Sandia

• University of Wisconsin & University of Maryland
  – DynInst & Infrastructure
Overview / Highlights

• Open Source Performance Analysis Tool
  – Most common performance analysis steps in *all in one tool*
  – *Extensible* by using plugins for data collection and viewing
  – GPL/LGPL license

• 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*
  – *Batch*
  – *Python Scripting*
Overview / Highlights

- Large Range of Platforms
  - *Linux Clusters* with x86, IA-64, Opteron, and EM64T CPUs
  - *SGI 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](http://www.openspeedshop.org)

- Linux versions
  - Tested on typical Linux distributions
    (including *SLES, RHEL, Fedora Core, Suse ....*)
Typical Workflow

Application
“Experiment”

Consists of one or more data “Collectors”

Results can displayed using several “Views”

Process Management Panel

Run

Open | SpeedShop™

Results

Stored in SQL database
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
Features:

- **Performance Experiments**
  - Gather new performance data
  - Analyze existing data from previous runs

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

- 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.
Features:

- Performance Experiments
- Aggregated Inclusive/Exclusive Time from 64 process MPI job

Graphical display with basic charts

Program output
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
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)
Dyninst in Open|SpeedShop

- 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, ...
Current Status: Open\SpeedShop

- Project being funded by NNSA/DOE
  - Two full-time developers and one part time
  - Developing new features
  - Bug fixing and support
- Project is on sourceforge
  - Can download source and rpms
  - Submit bug reports, comments, requests
  - Version 1.00 Released last November (SC07)
- ASC labs (LLNL, LANL, Sandia) are main users
- Several other users are in contact with team
Future plans: Open|SpeedShop

• **Target: Peta-Scale machines**
  – Data Collection and Transport
  – Result storage, aggregation, and analysis

• **Offline Collectors**
  – Execute experiments without tool backend
  – Targets microkernel architectures

• **Fully disconnect GUI from framework**
  – Remote execution with local GUI
  – Built on Command Line Interface (CLI)

• **Long Term Vision**
  – Performance “Cookbooks”
  – Help users plan experiments
Questions?

Jim Galarowicz
jeg@krellinst.org

Krell Institute
http://www.krellinst.org