A Comparison of Library Tracking Methods in High Performance Computing

Computer System Cluster and Networking Summer Institute 2013 Poster Seminar

William Rosenberger (New Mexico Tech), Dennis Trujillo (New Mexico State University), Chris DeJager (Michigan Technological University)
Need for a Library Tracking Utility

• HPC requires the support of many software packages, and multiple versions must be supported at the same time
  o Issues
    ▪ License availability
    ▪ Additional licensing cost
    ▪ Admin support
• No existing methods of tracking user software behavior
• Current user tracking methods don't typically illustrate true use
• Eliminate less utilized software and compiler options
Simulated New Software Version Acceptance

Users

- openmpi-1.5.4
- openmpi-1.6.5
Simulated New Software Version Rejection

Users

- gcc-4.4.4
- gcc-4.4.7
Potential Tracking Solutions

Automatic Library Tracking Database
• Developed at National Institute of Computational Science for use on Cray systems
• Presented at Cray User Group 2010
• Wraps the linker 'ld' and job launcher 'aprun'
• Stores to MySQL database

Linux Auditing Utility (auditd)
• System daemon
• Tracks specified files for access and or writes
• Stores to log files
Linux Auditing Utility (auditd)

- Provides log information on targeted files or directories
  - Object access and use
  - Stores to log files
  - Provides commands for searching log data
- Originally meant for security
- Root access required to add files to be tracked
- Log file is readable by a non-root Linux group
ALTD Operation

Wrapper Scripts

Linker Wrapper
- Gets list of libraries used
- Creates Linkline entry
- Creates Link Tag entry

Job Launch Wrapper
- Pulls reference number to the Link Tag table from the ALTD assembly header
- Creates Jobs table entry

MySQL Database

Linkline Table
Table Data:
- Linkline_id: Reference number to this record
- Unique list of libraries used in a compilation

Link Tag Table
Table Data:
- Tag_id: Reference number to this record
- Reference number to the Linkline entry.
- Username, date linked, build machine

Jobs Table
Table Data:
- Reference number to the Link Tag table
- Run machine, date run, username
ALTD Alterations

- Support added for wrapping mpirun and mvapich2 job launchers, while still allowing aprun
- Detects version of preferred MPI wrapped compiler and MPI executable and wraps accordingly
- Created script to query the database and collect the number of times a library has been compiled and run
- Made ALTD entries generic across servers
ALTD

Pros:
• Data is well organized into structured database
• Theoretical constant low overhead
• Tracks all libraries automatically

Cons:
• Requires a SQL database
• Tracks libraries used at compile time, not necessarily libraries used at run time
auditd

Pros:
• Standard Linux daemon
• Well tested by the Linux community
• Logs to a flat file

Cons:
• Dynamically linked libraries cannot be reliably tracked, the .so files are cached in memory and not read at every run
• Statically linked libraries can’t be tracked at runtime as they are in the executable
• A single make can generate numerous log entries
System Overhead Testing

• Interested in increase in compile and run time of different solutions over a control time
• Linpack was run while varying the problem size and the number of processors working on the problem
• A simple "do nothing" MPI program was also tested to investigate launch time more easily
Additional Runtime for a Simple MPI Program

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.12</td>
</tr>
<tr>
<td>9</td>
<td>0.12</td>
</tr>
<tr>
<td>16</td>
<td>0.12</td>
</tr>
<tr>
<td>25</td>
<td>0.12</td>
</tr>
<tr>
<td>36</td>
<td>0.12</td>
</tr>
<tr>
<td>49</td>
<td>0.12</td>
</tr>
<tr>
<td>64</td>
<td>0.12</td>
</tr>
</tbody>
</table>

- ALTD
- auditd
Linpack Runtime with ALTD

![Graph showing the relationship between Problem Size (N), Processor Grid (P*Q), and Runtime (s). The graph is a 3D plot with Problem Size on the x-axis, Processor Grid on the y-axis, and Runtime on the z-axis. The colors represent different levels of runtime across the problem sizes and processor grids.]
Linpack Runtime with auditd

Processor Grid (P*Q)

Problem Size (N)

Runtime (s)
Additional Runtime for Linpack with ALTD

Processor Grid (P*Q)

Runtime Difference (s)

Problem Size (N)

Average: 0.121 (s)
Additional Runtime for Linpack with auditd

Runtime Difference (s)

Processor Grid (P*Q)

Problem Size (N)

Average: 0.00304 (s)
Summary

• ALTD
  o The Automatic Library Tracking Database has the potential to track libraries well in HPC
  o Performance overhead is minimal and seems to be constant, regardless of job size
  o Additions to the software have provided for simplicity in utilization and data evaluation.

• auditd
  o Outperforms ALTD in some cases but falls behind compiling larger programs
  o auditd is not sufficient to track all types of libraries during compilation and runtime
Thanks!

We would like to thank the following individuals for their support throughout the workshop:

Georgia Pedicini
David Gunter
Dane Gardner
Matt Broomfield
Carol Hogsett
Josephine Olivas
Carol Connor
Gary Grider