User Environment Tracking and Problem Detection with XALT

Mark R. Fahey
Kapil Agrawal
Robert McLay
Doug James
Acknowledgment

- This work was supported by the NSF award 1339690 entitled “Collaborative Research: SI2-SSE: XALT: Understanding the Software Needs of High End Computer Users.”

- This material is based upon work performed using computational resources provided by the University of Tennessee’s Joint Institute for Computational Sciences and the Texas Advanced Computing Center (TACC) at the University of Texas at Austin.
Introduction

Most every computing center needs or wants

- How many users and projects use a particular library or executable
- If a library they maintain is used (and how often)
- If center provided packages are used more or less than user-installed packages
- After the fact identify users and code that used a buggy library
- Provide information on how an executable was built (provenance data)
- Catch compile time/run time differences
- Identify applications that are using deprecated libraries or just identify old binaries
Goal is a census of libraries and applications and automatic filtering of user issues

- Can answer all the questions on the previous slide
- What additional user problems can we detect and report (perhaps correct) automatically?
- Plan to add tracking of function calls as well

Collecting job-level and link-time level data and subsequent analytics

- Working on subsequent analytics
NSF funded project

Want to balance the need for portability with support for site-specific capabilities

Have a fairly straightforward process system administrators use to install, configure, and manage

Attempting to build a community around analytics of software usage

potentially one of many tools

Making it available to the community

Sourceforge and github

Eventually an optional interface to XDMod/SUPREMME
**Design**

- **Linker**
  - The linker (ld) wrapper intercepts the user link line
  - Generate assembly code (put a small bit of assembly in user code)
  - Generate link text (tracemap output from ld)
  - Generate link data (refine tracemap output)
  - Transmit collected data in `.json` format

- **Code launcher**
  - Launching a parallel job on compute nodes is often done via a batch system (like PBS, Slurm, or LoadLeveler) through a parallel job launcher such as aprun, mpirun, mpiexec, or ibrun.
  - Find executable (parse command line to identify exe)
  - Get actual launcher and command line options
  - Collect link time, job, and shared libraries information
  - Transmit data

- **Database transmission is generic as we support 3 methods**

---

```
.section .xalt
.asciz "XALT_Link_Info"

.byte 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
.asciz "<XALT_Version>%%%%"
.asciz "<Build.Syshost>%%darter%%
.asciz "<Build.compiler>%%driver.cc%"
.asciz"<Build.OS>%%Linux_%%_3.0.101-0.29-default%
.asciz "<Build.User>%%kagrawa1%
.asciz "<Build.UUID>%%bd97b98b-2169-416e-85c1-762be8846dd2%
.asciz "<Build.Year>%%2014%
.asciz "<Build.date>%%Fri_%%_Jul_%%_%%_%%_4_%%_13:37:01_%%_%%_%%_2014%
.asciz "<Build.Epoch>%%1406914621.1%
.byte 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
.asciz "XALT_Link_Info_End"
```

XALT assembly code

---

HUST'14 Workshop | November 21, 2014
Portability

👩‍💻 Support several job launchers and batch systems
👩‍💻 Targeted several architectures
👩‍💻 Expect subsequent releases will include additional architectures as time permits and/or the community shares them.
👩‍💻 New architectures may require supporting additional linkers and code launchers

👩‍💻 Information transmission to database possibilities:
👩‍💻 Files: This is the default for XALT - all information is dumped into ‘.json’ files (one each for compile time and runtime); asynchronously a script parses these files and uploads the data to the XALT database.
👩‍💻 SYSLOG: Data captured is written to SYSLOG; data is asynchronously parsed by a script which then writes it into the XALT database.
👩‍💻 Direct Database Interaction: All the linkage and execution information is directly inserted into the XALT database in real time when a user compiles or executes a code
Requirements

- Avoid impacting the user experience
- Must work seamlessly on any cluster, workstation or high-end computer
- Must support both static and dynamic libraries
- Lightweight solution
- Alert (if possible) users and support staff of software configuration issues
Database schema
Applications of XALT

There are many analyses that can be done with XALT data

- Most/least and trends reports for:
  - Libraries
  - Modulefiles
  - Applications
  - Based on user or project or time used
- Last time a library was used
- Providing usage statistics to developers and vendors
- Restoring the program environment where user applications were built (reproducibility)
mysql> SELECT xalt_object.module_name, count(xalt_run.date) AS Jobs, 
ROUND(SUM(run_time*num_cores)/3600) as TotalSUs FROM xalt_run, xalt_link, join_link_object, xalt_object WHERE xalt_run.syshost='darter' AND xalt_object.module_name is NOT NULL AND xalt_run.uuid = xalt_link.uuid AND xalt_link.link_id = join_link_object.link_id AND join_link_object.obj_id = xalt_object.obj_id AND xalt_run.date >= '2014-11-01' AND xalt_run.date <= '2014-11-09' GROUP BY xalt_object.module_name ORDER BY Jobs DESC;
<table>
<thead>
<tr>
<th>module_name</th>
<th>Jobs</th>
<th>TotalSUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alps/5.2.1-2.0502.8712.10.32.ari</td>
<td>26458</td>
<td>258684</td>
</tr>
<tr>
<td>cray-mpich/7.0.3</td>
<td>26456</td>
<td>259040</td>
</tr>
<tr>
<td>wlm_detect/1.0-1.0502.51217.1.1.ari</td>
<td>13229</td>
<td>129342</td>
</tr>
<tr>
<td>udreg/2.3.2-1.0502.8763.1.11.ari</td>
<td>13229</td>
<td>129342</td>
</tr>
<tr>
<td>xpmem/0.1-2.0502.51169.1.11.ari</td>
<td>13229</td>
<td>129342</td>
</tr>
<tr>
<td>ugni/5.0-1.0502.9037.7.26.ari</td>
<td>13229</td>
<td>129342</td>
</tr>
<tr>
<td>pmi/5.0.5-1.0000.10300.134.8.ari</td>
<td>13227</td>
<td>129341</td>
</tr>
<tr>
<td>gcc/4.8.1</td>
<td>10868</td>
<td>59680</td>
</tr>
<tr>
<td>rca/1.0.0-2.0502.51491.3.92.ari</td>
<td>10852</td>
<td>59675</td>
</tr>
<tr>
<td>dmapp/7.0.1-1.0502.9080.9.32.ari</td>
<td>10852</td>
<td>59675</td>
</tr>
<tr>
<td>fftw/3.3.4.0</td>
<td>3123</td>
<td>1482</td>
</tr>
<tr>
<td>cray-libsci/13.0.1</td>
<td>2357</td>
<td>69848</td>
</tr>
<tr>
<td>craype-intel-knc</td>
<td>1758</td>
<td>522</td>
</tr>
<tr>
<td>hdf4/4.2.9</td>
<td>1180</td>
<td>667</td>
</tr>
<tr>
<td>cray-hdf5/1.8.12</td>
<td>586</td>
<td>174</td>
</tr>
<tr>
<td>cray-netcdf/4.3.1</td>
<td>586</td>
<td>174</td>
</tr>
<tr>
<td>szip/2.1</td>
<td>295</td>
<td>167</td>
</tr>
<tr>
<td>fftw/3.3.0.4</td>
<td>274</td>
<td>78293</td>
</tr>
<tr>
<td>gcc/4.8.2</td>
<td>156</td>
<td>44739</td>
</tr>
</tbody>
</table>
Program usage

```
mysql> select link_program, build_syshost, count(*) from xalt_link group by link_program, build_syshost;
```

<table>
<thead>
<tr>
<th>link_program</th>
<th>build_syshost</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure</td>
<td>darter</td>
<td>7</td>
</tr>
<tr>
<td>driver.CC</td>
<td>darter</td>
<td>8</td>
</tr>
<tr>
<td>driver.cc</td>
<td>darter</td>
<td>180</td>
</tr>
<tr>
<td>ftn_driver</td>
<td>darter</td>
<td>173</td>
</tr>
<tr>
<td>g++</td>
<td>darter</td>
<td>2396</td>
</tr>
<tr>
<td>gcc</td>
<td>darter</td>
<td>6190</td>
</tr>
<tr>
<td>gfortran</td>
<td>darter</td>
<td>959</td>
</tr>
<tr>
<td>icc</td>
<td>darter</td>
<td>1890</td>
</tr>
<tr>
<td>icpc</td>
<td>darter</td>
<td>562</td>
</tr>
<tr>
<td>ifort</td>
<td>darter</td>
<td>915</td>
</tr>
<tr>
<td>make</td>
<td>darter</td>
<td>123</td>
</tr>
</tbody>
</table>

11 rows in set (0.02 sec)
A critical bug was identified in FFTW version 3.3.0.2, affecting code correctness

Find which users have linked this library

```sql
mysql> select distinct build_user from xalt_link,xalt_object
where xalt_object.object_path like '%fftw/3.3.0.2/%';
```

```
+----------+
| username |
+----------+
| user1    |
| user2    |
| user3    |
| user4    |
| user5    |
+----------+
5 rows in set (1.33 sec)
```

• Querying the database reveals that several users have applications linked to the buggy library
Was the buggy library used?

And it’s confirmed that user “user4” has run the application linked to the buggy library.

It’s now up to the user services group to contact the user and recommend relinking their applications against the newer version of FFTW, which has fixed the bug.
How did I build my program 2 months ago?

```sql
mysql> select xalt_link.* from xalt_link where build_user like '%faheymr%' AND exec_path like '%hyperslab%';

+---------+---------------------+--------------+------------+---------------+---------------------+---------------------+--------------+------------+---------------+------------------------------+---------------------+--------------+------------+---------------+------------------------------+---------------------+--------------+------------+---------------+------------------------------+
<table>
<thead>
<tr>
<th>link_id</th>
<th>date</th>
<th>link_program</th>
<th>build_user</th>
<th>build_syshost</th>
<th>build_epoch</th>
<th>exit_code</th>
<th>exec_path</th>
<th>/nics/d/home/faheymr/examples/hdf5/hyperslab</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2014-09-23 14:17:29</td>
<td>ftn_driver</td>
<td>faheymr</td>
<td>darter</td>
<td>1411496249.58</td>
<td>0</td>
<td>/nics/d/home/faheymr/examples/hdf5/hyperslab</td>
<td></td>
</tr>
</tbody>
</table>

3 rows in set (0.01 sec)
```
How did I build my program 2 months ago? (cont)

```sql
mysql> select object_path, timestamp from xalt_object, join_link_object where 
  join_link_object.link_id="4" AND join_link_object.obj_id=xalt_object.obj_id;
+-------------------------------------------------------------------+---------------------+
| object_path             | timestamp            |
+-------------------------------------------------------------------+---------------------+
| //usr/lib64/libc.a      | 2014-09-26 15:49:53  |
| //usr/lib64/libdl.a     | 2014-09-26 15:49:53  |
| //usr/lib64/libhugetlbfs.a | 2014-09-26 15:49:53 |
| //usr/lib64/libm.a      | 2014-09-26 15:49:53  |
| //usr/lib64/libpthread.a | 2014-09-26 15:49:53 |
| //usr/lib64/libz.a      | 2014-09-26 15:49:53  |
| /opt/cray/atp/1.7.2/lib/libAtpSigHCommData.a | 2014-09-26 15:49:53 |
| /opt/cray/atp/1.7.2/lib/libAtpSigHandler.a   | 2014-09-26 15:49:53 |
| /opt/cray/cce/8.2.5/craylibs/x86-64/libcsup.a | 2014-09-26 15:49:53 |
| /opt/cray/cce/8.2.5/craylibs/x86-64/libf.a    | 2014-09-26 15:49:53 |
| /opt/cray/cce/8.2.5/craylibs/x86-64/libfi.a   | 2014-09-26 15:49:53 |
| /opt/cray/cce/8.2.5/craylibs/x86-64/libtcmalloc_minimal.a | 2014-09-26 15:49:53 |
| /opt/cray/cce/8.2.5/craylibs/x86-64/libu.a     | 2014-09-26 15:49:53 |
| /opt/cray/cce/8.2.5/craylibs/x86-64/no_mmap.o  | 2014-09-26 15:49:53 |
| /opt/cray/hdf5/1.8.12/CRAY/81/lib/libhdf5_fortran_cray.a  | 2014-09-26 15:49:53 |
| /tmp/pe_14932/hyperslab_1.o                        | 2014-09-26 15:49:53 |
| /usr/lib64/crt1.o                                    | 2014-09-26 15:49:53 |
| /usr/lib64/crti.o                                     | 2014-09-26 15:49:53 |
| /usr/lib64/crtn.o                                     | 2014-09-26 15:49:53 |
+-------------------------------------------------------------------+---------------------+
27 rows in set (0.00 sec)
```
Automating the process of alerting?

- Ideally, user support specialists would be alerted automatically to “situations of interest”
  - Users running applications linked to legacy, less-performant, or buggy libraries
  - Users running legacy versions of applications
  - Users building code with legacy compilers
  - Users making use of their own libs or apps, when more optimized versions are available centrally
TACC_Stats (and SUPReMM)

- Job-level transparent performance monitoring from HPC compute nodes
  - CPU performance counters
  - IB statistics
  - Lustre statistics
  - Scheduler job statistics
  - Host data
  - OS statistics

- Analyses integrate available Lariat data (and will be XALT in the near future)
Alpha testers - Thanks

- Tim Robinson, CSCS
- Bilel Hadri, KAUST
- Julius Westerman, LANL
- Davide Del Vento, NCAR
- Andrew Elwell, iVEC
Future

- Will add a feature to track function calls
  - Only those function calls resolved by external libraries
  - Will not track
    - User defined functions
    - Auxiliary functions in a library

- Ability to compare run time environment against compile time environment
  - Provide warning messages to users
  - May help users self-diagnose run time problems

- Much, much more data analysis
Conclusion

- Developed XALT
  - Released beta Nov 10, 2014

- Tracks user codes and environments
  - Libraries, modulefiles, executables, env vars

- Data can be used to generate
  - most/used and trend reports on libraries, executables, etc
  - Last time something was used
  - Restoring the program environment where user applications were built

- More new features coming

- Product is in production at TACC and NICS and looking for wider adoption
Thank You

- mfahey@utk.edu
- mclay@tacc.utexas.edu
- xalt-users@lists.sourceforge.net

Mailing list

- http://sourceforge.net/projects/xalt
- http://github.com/Fahey-McLay/xalt
Assumptions

- More than one linker and/or job launcher to intercept
- Want to track library versions if possible
- Want to only track the function calls from the user code that need to be resolved by external libraries