User Environment Tracking and Problem Detecton with XALT

Mark R. Fahey

Kapil Agrawal

Robert McLay

First International Workshop on HPC User Support Tools (HUST-14)

Held in conjunction with SC14: The International Conference on High Performance Computing, Networking, Storage and Analysis

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 [26] and the Texas Advanced Computing Center (TACC) at the University of Texas at Austin.

Outline

- Introduction
- Design
- Usage

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 howan executable was built (provenance data)
  - Catch compile time/run time differences
  - Identify applications that are using deprecated libraries or just identify old binaries

XALT

- 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
  - Alpha version for collection
  - Working on subsequent analytics

XALT

- NSF funded project
- Want to balance the need for portability with support for site-specific capabilities
- Process system administrators use to install, configure, and manage
- Trying to build a community around analytics – 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

Portability

- Likely more than one linker and/or job launcher to support at each site
  - 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

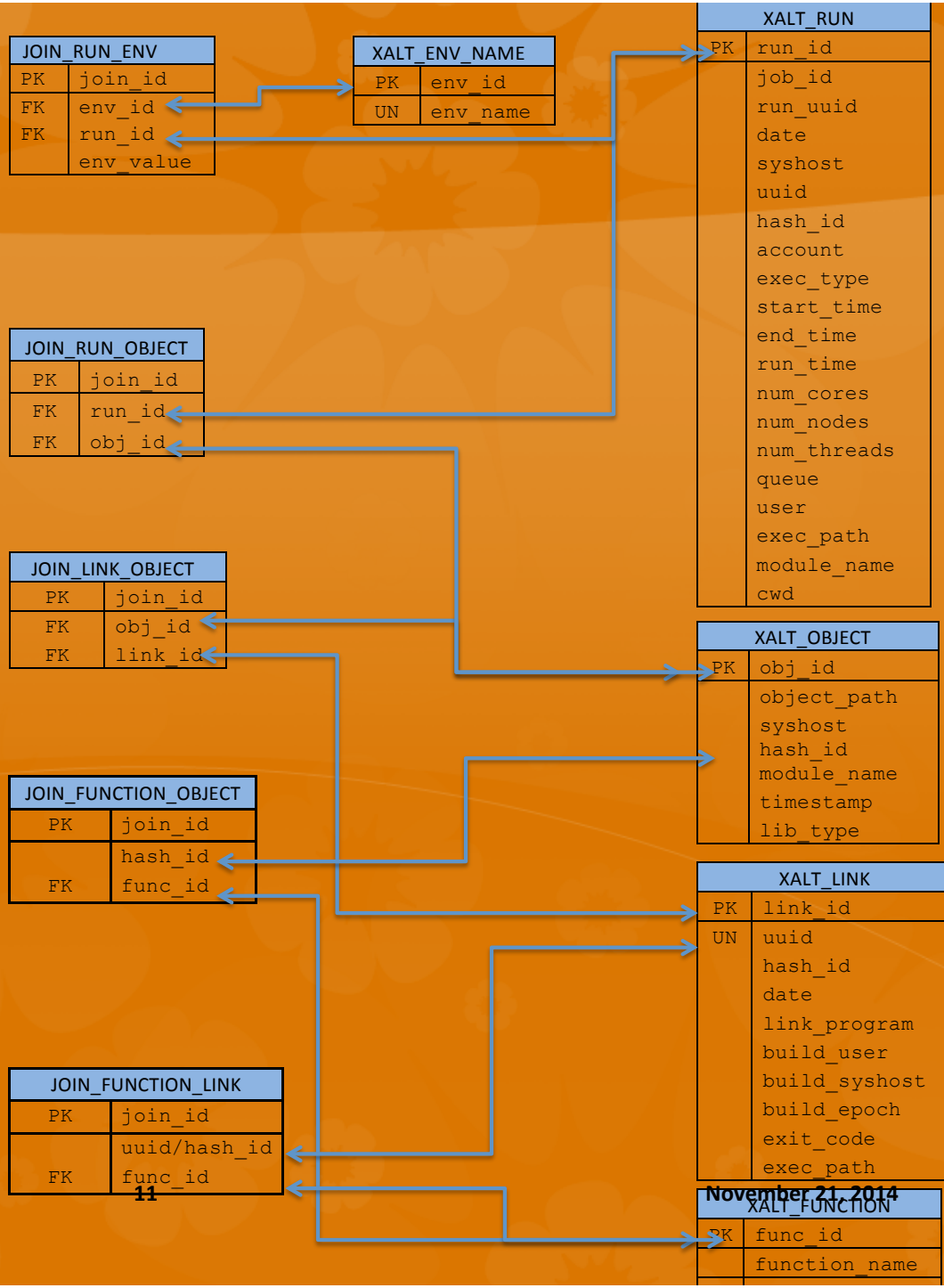
Requirements

- 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

Assumptions

- *More than one linker and/or job launcher to intercept*
- *Not everything on the original link line should be captured*
- *Want to track function calls if possible*
- *Want to track library versions if possible*

Database schema



Usage

- 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
  - Provenance/reproducibility reports

Applications of XALT

- Understanding current library usage and plan for future software need
- Providing usage statistics to developers and vendors
- Restoring the program environment where user applications were built
- Assisting with debugging system issues

Modulefile usage

```
mysql> SELECT _d.syshost, _d.module_name, _d.cnt FROM
(SELECT module_name, syshost, COUNT(*) AS cnt FROM
xalt_object GROUP BY module_name, syshost) _d ORDER BY
_d.syshost, _d.cnt DESC;
```

syshost	module_name	cnt
darter	cray-trilinos/11.6.1.0	128
darter	craype-intel-knc	23
darter	cray-hdf5/1.8.12	16
darter	cray-mpich/6.3.0	14
darter	cray-tpsl/1.4.0	13
darter	fftw/3.3.0.4	9
darter	gcc/4.9.1	8
darter	ncl/6.1.2	8
darter	gcc/4.8.2	8
darter	cray-libsci/12.2.0	8
darter	cray-netcdf/4.3.1	6

Program usage

```
mysql> select link_program, build_syshost,
count(*) from xalt_link group by link_program,
build_syshost;
+--------------+---------------+----------+
| link_program | build_syshost | count(*) |
+--------------+---------------+----------+
| configure    | darter        |        7 |
| driver.CC    | darter        |        8 |
| driver.cc    | darter        |      180 |
| ftn_driver   | darter        |      173 |
| g++          | darter        |     2396 |
| gcc          | darter        |     6190 |
| gfortran     | darter        |      959 |
| icc          | darter        |     1890 |
| icpc         | darter        |      562 |
| ifort        | darter        |      915 |
| make         | darter        |      123 |
+--------------+---------------+----------+
11 rows in set (0.02 sec)
```

Identifying users or codes or libraries

A critical bug was identified in FFTW version 3.3.0.2, affecting code correctness

Find which users have linked this library

```
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?

```
mysql> select distinct xalt_run.run_id, xalt_run.job_id, xalt_run.date, xalt_run.syshost,
xalt_run.user, xalt_run.exec_path from xalt_run, xalt_object, join_run_object where
xalt_object.object_path like '%fftw/3.3.0.2/%' AND xalt_object.obj_id=join_run_object.obj_id
AND join_run_object.run_id=xalt_run.run_id;

+--------+-------------+---------------------+---------+--------+------------------+
| run_id | job_id      | date                | syshost | user   | exec_path        |
+--------+-------------+---------------------+---------+--------+------------------+
|   7273 | 350840.ocoe | 2014-10-23 13:22:29 | darter  | user4  | ~/cp2k/cp2k.psmp |
+--------+-------------+---------------------+---------+--------+------------------+
1 rows in set (0.08 sec)
```

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?

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

+---------+---------------------+-------------+-----------+---------------
+---------------+-----------+-----------------------------------------------+

| link_id | date                | link_program | build_user | build_syshost |
build_epoch   | exit_code | exec_path                                     |

+---------+---------------------+-------------+-----------+---------------
+---------------+-----------+-----------------------------------------------+

|       4 | 2014-09-23 14:17:29 | ftn_driver   | faheymr    | darter        |
1411496249.58 |         0 | /nics/d/home/faheymr/examples/hdf5/hyperslab |

+---------+---------------------+-------------+-----------+---------------
+---------------+-----------+-----------------------------------------------+

3 rows in set (0.01 sec)
```

How did I build my program 2 months ago? (cont)

```
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/librt.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_cray.a                | 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 |
| /opt/gcc/4.4.4/snos/lib/gcc/x86_64-suse-linux/4.4.4/crtbeginT.o | 2014-09-26 15:49:53 |
| /opt/gcc/4.4.4/snos/lib/gcc/x86_64-suse-linux/4.4.4/crtend.o    | 2014-09-26 15:49:53 |
| /opt/gcc/4.4.4/snos/lib/gcc/x86_64-suse-linux/4.4.4/crtfastmath.o | 2014-09-26 15:49:53 |
| /opt/gcc/4.4.4/snos/lib/gcc/x86_64-suse-linux/4.4.4/libgcc.a    | 2014-09-26 15:49:53 |
| /opt/gcc/4.4.4/snos/lib/gcc/x86_64-suse-linux/4.4.4/libgcc_eh.a | 2014-09-26 15:49:53 |
| /opt/gcc/4.4.4/snos/lib64/libstdc++.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)
```

Cray cce/8.2.5
Hdf5/1.8.12

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)

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

Alpha testers - Thanks

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

Thank You

- mfahey@utk.edu
- mclay@tacc.utexas.edu