TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

XALT: Understanding HPC Usage via Job Level Collection

Robert McLay

January 31, 2019

XALT: What runs on the system

- Was a NSF Funded project.
- A Census of what programs and libraries are run
- Running at TACC, NICS, U. Florida, KAUST, LLNL, ...
- Integrates with TACC-Stats.
- Has commercial support from Ellexus

Understanding what your users are doing

- What programs, libraries are your users using?
- What are the top programs by core-hours? by counts ? by users?
- Are they building their own programs or using someone elses?
- Are Executables implemented in C/C++/Fortran?
- Track MPI: tasks? nodes?
- Track Threading via $OMP_NUMTHREADS

History of XALT

- Mark Fahey (was NICS, now ANL): ALT-D (MPI only)
- Robert McLay (TACC) Lariat (MPI only)
- Reuben Budiardja (was NICS now ORL)
- ALT-D + Lariat $\Rightarrow$ XALT 1: (MPI only)
- XALT 2: All programs

Design Goals

- Be extremely light-weight
- Provide provenance data: How?
- How many use a library or application?
- Collect Data into a Database for analysis.

Design: Linker

- XALT wraps the linker to enable tracking of exec's
- The linker (ld) wrapper intercepts the user link line.
- Generate assembly code: key-value pairs
- Capture tracemap output from ld
- Transmit collected data in *.json format
- Adds codes that executes before main() and after main() completes

Design: Transmission to DB

- File: collect nightly
- Syslog: Use Syslog filtering (or ELK)

Lmod to XALT connection

- Lmod spider walks entire module tree.
- Can build a reverse map from paths to modules
- Can map program & libraries to modules.
- /opt/apps/i15/mv2_2_1/phdf5/1.8.14/lib/libhdf5.so.9 $\Rightarrow$ phdf5/1.8.14(intel/15.02:mvapich2/2.1)
- Also helps with function tracking.
- Tmod Sites can still use Lmod to build the reverse map.

Protecting XALT from users

- Must protect XALT from unexpected changes in the environment.
- Solution: LD_LIBRARY_PATH="@ld_lib_path@" PATH=/usr/bin:/bin C++-exec ...
- set PATH at configure time.

Using XALT Data

- Targetted Outreach: Who will be affected
- Largemem Queue Overuse
- XALT and TACC-Stats

Tracking Non-mpi jobs (I)

- Originally we tracked only MPI Jobs
- By hijacking mpirun etc.
- Now we can use ELF binary format to track jobs

ELF Binary Format Trick

```
void myinit(int argc, char **argv)
{
  /* ... */
}
void myfini()
{
  /* ... */
}
  static __attribute__((section(".init_array")))
       typeof(myinit) *__init = myinit;
  static __attribute__((section(".fini_array")))
       typeof(myfini) *__fini = myfini;
```

Using the ELF Binary Format Trick

- This C code is compiled and linked in through the hijacked linker
- It can also be used with LD_PRELOAD
- By default we only use LD_PRELOAD but can do both.

Challenges (I)

- Do not want to track every mv, cp, etc
- Only want to track some executables on compute nodes
- Do not want to get overwhelmed by the data.

Answers

- XALT Tracking only when told to
- Compute node only by host name filtering
- Executable Filter based on Path
- Protection against closing stderr before fini.
- Sampling of serial programs.
- Site configurable!

Path Filtering

- Uses FLEX to compile in patterns
- Use regex expression to control what to keep and ignore.
- Three files containing regex patterns, converted to code.
- Accept List Tests: Track /usr/bin/ddt, /bin/tar, /usr/bin/perl
- Ignore List Tests: /usr/bin, /bin, /sbin, ...

TACC_config.py

```
hostname_patterns = [
  ['KEEP', '^c[0-9][0-9][0-9]-[0-9][0-9][0-9]:*']
   ]
path_patterns = [
     ['PKGS',  r'.*/python[0-9][^/][^/]*'],
     ['PKGS',  r'.*/R'],
     ['KEEP',  r'^/usr/bin/ddt'],
     ['SKIP',  r'^/usr/.*'],
     ['SKIP',  r'^/bin/.*'],
   ]
env_patterns = [
     [ 'SKIP', r'^MKLROOT=.*' ],
     [ 'SKIP', r'^MKL_DIR=.*' ],
     [ 'KEEP', r'^I_MPI_INFO_NUMA_NODE_NUM=.*'],
   ]
```

Speeding up XALT 2

- Minimal impact on jobs ($>$ 0.09 secs)
- All Non-MPI programs only produce end record
- This supports sampling!

Sampling Non-MPI programs

- XALT has sampling rules (site configurable!)
- 0 mins $<$ 5 mins $\Rightarrow$ 0.01% recorded
- 5 mins $<$ 10 mins $\Rightarrow$ 1% recorded
- 10 mins $<$ ∞ $\Rightarrow$ 100% recorded
- Can now track perl, awk, sed, gzip etc

Tracking R packages

- XALT 2 can now track R package usage
- James McComb & Michael Scott from IU developed the R part
- They do this by intercepting the “imports”
- Plan to support Python and MATLAB later.

New program: xalt_extract_record

- This program reads the watermark.
- Find out who built this program on what machine
- Find out what modules where used.

Example of xalt_extract_record output

```
*************************************************

XALT Watermark: hello
*************************************************

Build_Epoch                  1510257139.4624
Build_LMFILES               /opt/apps/modulefiles/in-
tel/17.0.4.lua:...
Build_LOADEDMODULES    intel/17.0.4:impi/17.0.3:pytho
Build_OS                              Linux  3.10.0-
514.26.2.el7.x86_64
Build_Syshost                 stampede2
Build_UUID                   586d5943-67eb-480b-a2fe-
35e87a1f22c7
Build_User                    mclay
Build_compiler                icc
Build_date                   Thu Nov 09 13:52:19 2017
Build_host             c455-011.stampede2.tacc.utexas.ed
XALT_Version                  1.7.7-devel
```

New Feature: Track GPU usage

- Optionally, XALT can know if a GPU was used.
- XALT will only know if one or more GPU's were accessed
- No performance data
- Thanks to Scott McMillan from NVIDIA for the contribution.

New Feature: Track Singularity Container Usage

- Sites can configure their Singularity script to include XALT
- It works well with syslog or file transfer of data
- Thanks to Scott McMillan from NVIDIA for the contribution.

Conclusion

- Lmod:
  - Source: github.com/TACC/lmod.git, lmod.sf.net
  - Documentation: lmod.readthedocs.org
- XALT:
  - Source: github.com/xalt/xalt.git, xalt.sf.net
  - Documentation: XALT 2 $\Rightarrow$ xalt.readthedocs.org