XALT: Application Provenance, Tracking, and Problem Detection

Michael Uddstrom (NIWA/NeSI)
Jordi Blasco (NeSI)
08-February 2016

Introduction

XALT[1] is a mechanism for following users' jobs and environments on a cluster. It provides a census of libraries and applications, and automatically filters user issues, e.g.

- How many users and projects use a particular library or executable;
- If a library they maintain is used (and how often)
- If centre provided packages are used more or less often than user-installed packages
- After the fact identify users and code that used a buggy library
- Provide information on how as executable was built (provenance);
- Catch compile time/run time differences
- Identify applications that are using deprecated libraries, or old binaries;

Design objectives include:

- No impact on the user experience;
- Must work seamlessly on any (Linux) cluster;
- Only report on link line libraries that are actually used;
- Must support both static and dynamic libraries;
- Where possible, alert users of software configuration issues.

Datasets and historical reports generated by XALT preserve institutional knowledge and lessons learned, so that users, developers and support staff need not reinvent solutions to problems that have already been encountered. Further while XALT does not collect performance data, it can identify opportunities to improve performance.

XALT is being developed through the collaborative efforts of the National Institute for Computational Sciences (NICS) and the Texas Advanced Computing Center (TACC).

XALT in a Nutshell

1) Wrappers are used to gather information from the GNU linker, and the code launcher (e.g. `mpirun, aprun, srun`) when the code is run.
2) The resulting information is stored in `.json` files (the default), or SYSLOG, or can be written directly (to the XALT database), is accumulated in an XALT database.
3) Reports can be generated from the XALT database[1]. These reports can be configured to provide a census of libraries and applications being used as well as the more detailed provenance information.

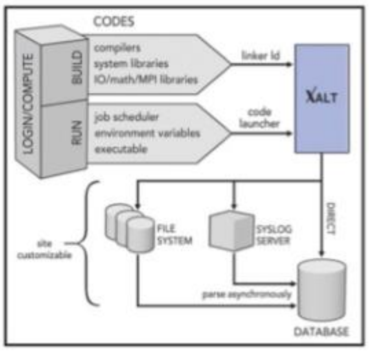

Figure 1 XALT Overview. From [1].

[1] There is no agreed tool yet, since each HPC center could be interested in different metrics. The most common approach is to use a grafana dashboard (http://grafana.org) which relies on Elasticsearch, InfluxDB and other big data technologies

XALT Experience on the NeSI Platform Pan

The New Zealand eScience Infrastructure (NeSI, http://nesi.org.nz) operates three High Performance Computers (HPC), an IBM IdataPlex (Pan), an IBM P575/POWER6 (FitzRoy) and an IBM Blue Gene/P (Foster). XALT has been installed on Pan, is being considered for FitzRoy (AIX) and is being used in trials. Our experience to date suggests:

- Installation is relatively simple once prerequisites are met (`Python 2.7.5` and `Lmod 5.8.rc2` or later versions). The software is available at https://github.com/Fahey-McLay/xalt
- We assume that all the researchers are using `srun` as the code launcher. At this time only three applications are currently outside the `srun` wrapper control (ANSYS, Abaqus and ORCA). The latest XALT pre-release is also able to track those binaries via the new Function Tracking Capability in version 0.7.0 [3].
- Site specific changes need to be made in `$XALT_DIR/site/xalt_syshost_default.py` (e.g. to identify the HPC). On a Cray system this is `$XALT_DIR/site/xalt_find_exec_aprun.py`
- In order to gather the information from each job, the slurm task prolog introduces extensions of the binary path and python path to use the XALT wrappers:

```
echo "export PATH=/share/apps/xalt/bin:$PATH"
echo "export PYTHONPATH=/share/apps/xalt/libexec:$PYTHONPATH"
echo "export PYTHONPATH=/share/apps/xalt/site:$PYTHONPATH"
```

- The data gathered is temporarily stored in `json` files located in the hidden directory `~/.xalt.d` in the home directory of each user. Currently we use the default method to regularly import this data into the XALT database via a Python script, which searches for files in the `~/.xalt.d/` directories. However, as the `~/.xalt.d/` directories are stored in user's directories – they could be accidentally deleted before the data are recovered. A better option would be to load the data directly into the XALT database – we will investigate this "transmission method" in the future.

As noted above – there is no standard method for generating reports from the XALT database, but that is a relatively minor issue.

Within the NeSI context, XALT provides answers to a number of key questions, in particular;

- What applications are being run on the HPCs?
- How much resource is being consumed by each application?
- What libraries are being used by these applications?
- Are the applications as compiled and linked valid (e.g. is it using deprecated libraries, or libraries with known issues)?
- What resources might we need in the future?

In addition, XALT data can be used to provide provenance information for publication purposes.

Community Usage of XALT

Although XALT is at an early stage of development, it has had significant uptake, and has been tested and/or implemented at The National Institute for Computer Sciences, The Oak Ridge Leadership Computing Facility, The National Center for Supercomputing Applications, Baden-Württemberg, The National Energy Research Scientific Computing Center, The Swiss National Supercomputing Centre, The National Oceanic and Atmospheric Administration, Florida State University (FSU) and KAUST Supercomputing Centre, as well as on Stampede at the Texas Advanced Computing Center (TACC). Budiardja et al. (2015) [2] report on the use of XALT at TACC, FSU and KAUST over a one year period. The following plots (Figure 2 and Figure 3) show example analyses from this study.

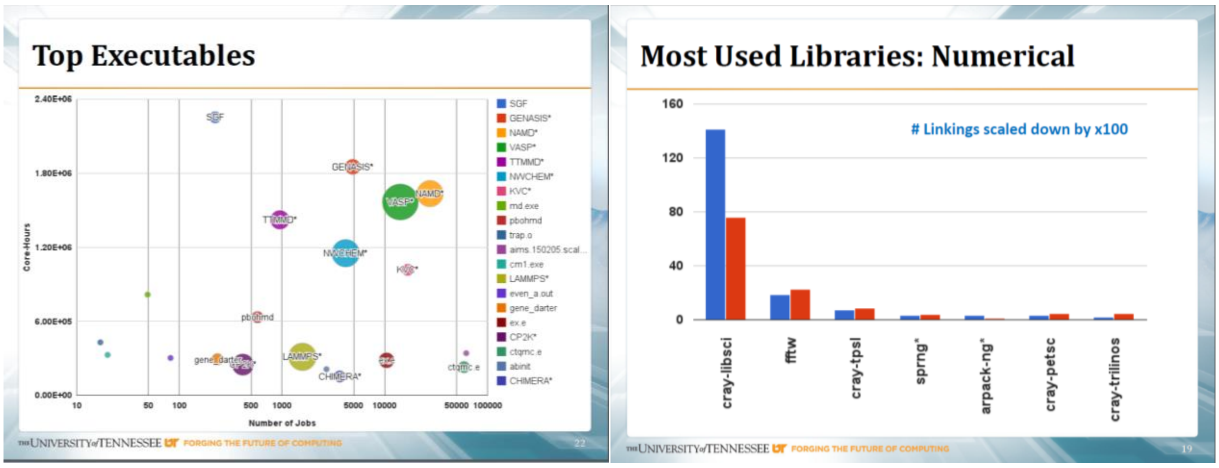

Figure 2 Left hand side: Accumulated Core-h usage by application, Right hand side: Most used numerical libraries. From [2]

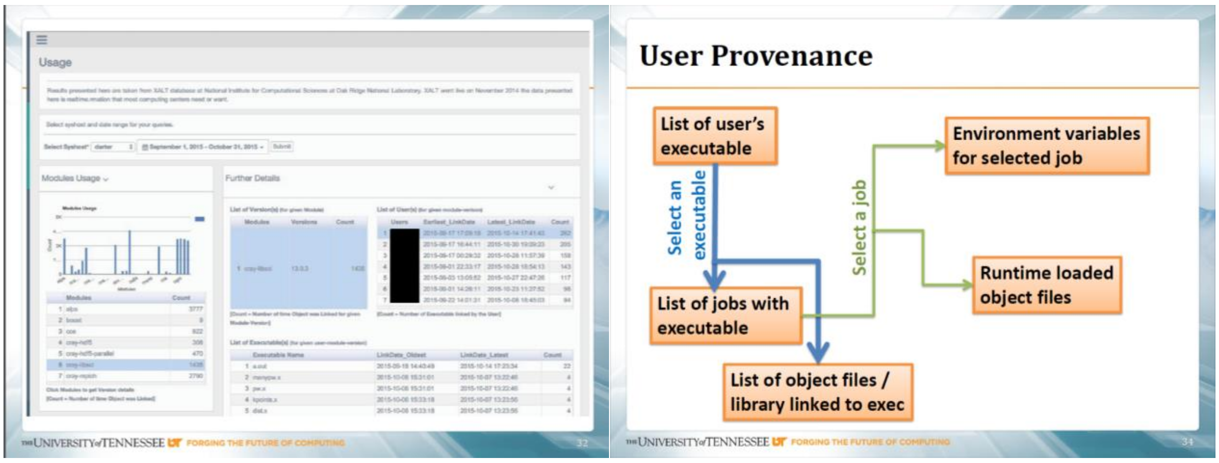
Usage
Modules Usage
Further Details
User Provenance
List of user's executable
Select an executable
List of jobs with executable
Select a job
Environment variables for selected job
Runtime loaded object files
List of object files / library linked to exec
THE UNIVERSITY of TENNESSEE UT FORGING THE FUTURE OF COMPUTING

Figure 3 Left hand side: XALT Portal – a web interface to review XALT data that allows staff to 1) easily get high level library, compiler, and executable usage, 2) from any of those "entry points", drill-down to users associated with library/compiler/executable, and their jobs and job environment, and 3) search who uses a particular library or executable. Right hand side: Examining provenance - analysing the components of an executable – e.g. MPI library, Compiler, Libraries, and versions etc. of a version built n months ago. From [2]

References

[1] Agrawal, K., G. Peterson, R. Budiardja, M. Fahey, R. McLay, 2015 User Environment Tracking and Problem Detection with XALT, Extended Abstract, SC15, Austin, Texas, USA.

http://sc15.supercomputing.org/sites/all/themes/SC15images/tech_poster/poster_files/post281s2-file3.pdf

[2] Budiardja, R.D., M. Fahey, R. McLay, P.D. Don, B. Hadri, D. James, 2015: Community Use of XALT in Its First Year in Production http://hust15.github.io/files/HUST15-Budiardja-XALT-First-Year-Production-Slides.pdf

[3] Function Tracking Capability https://github.com/Fahey-McLay/xalt/releases/tag/0.7.0