User Environment Tracking and Problem Detection with XALT

Kapil Agrawal[1], Gregory Peterson[1] (Advisor), Mark Fahey[2], Robert McLay[3] , Reuben Budiardja[4]

[1]Department of Electrical Engineering, & Computer Science, University of Tennessee, Knoxville, TN 37996

[2]Argonne National Laboratory, Argonne, IL 60439

[3]Texas Advanced Computing Center (TACC), University of Texas at Austin, Austin, TX 78712

[4]National Institute for Computational Sciences (NICS) at ORNL, Oak Ridge, TN 37831

❖ Abstract

This work enhances our understanding of individual users' software needs, then leverages that understanding to help stakeholders conduct business in a more efficient, effective and systematic manner. XALT is designed to track linkage and execution information for applications that are compiled and executed on any Linux cluster, workstation, or high-end supercomputers. XALT allows administrators and other support staff to consider demand when prioritizing what to install, support and maintain. Datasets, dashboards, and historical reports generated by XALT and the systems with which it interoperates will preserve institutional knowledge and lessons learned so that users, developers, and support staff need not reinvent the wheel when issues arise that have already been encountered.

❖ Objective

- Must work seamlessly on any cluster, workstation or high-end computer.
- Avoid impacting user experience.
- Must support both dynamic and static libraries.
- Alert users and support staff of the software configuration issues.
- Lightweight solution.

❖ XALT Overview

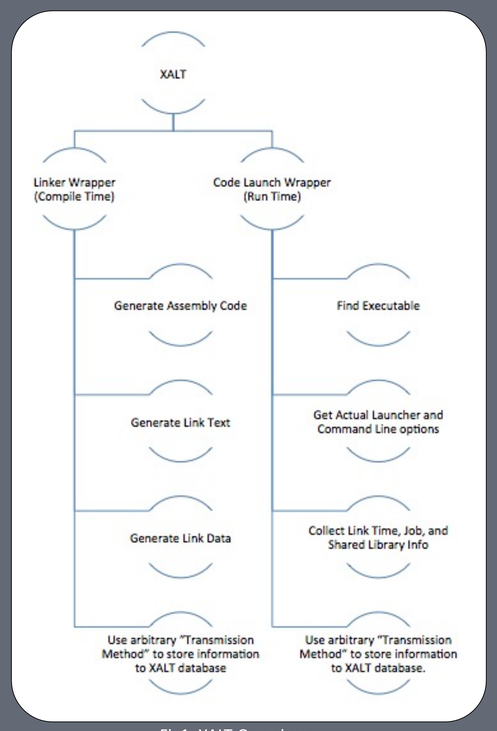


Fig1: XALT Overview

```
.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>%%kagrawal%%"
.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"
```

Fig2: XALT Assembly Code

❖ Transmission Methods

- Files: All information is dumped into ".json" files, which then is parsed through a python script and uploaded to a database.
- Syslog: An substitute to Files (above point) can be used for sites which may see storing these files in user's home directories as a security and/or performance issue.
- Direct to Database (D2D): This approach is straightforward, information can be inserted directly to XALT database.

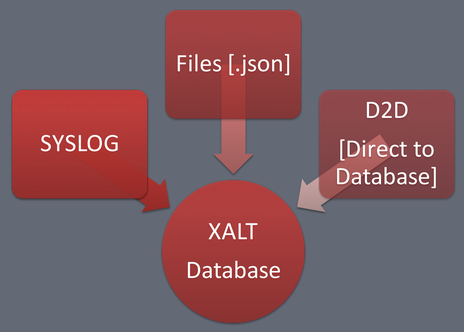
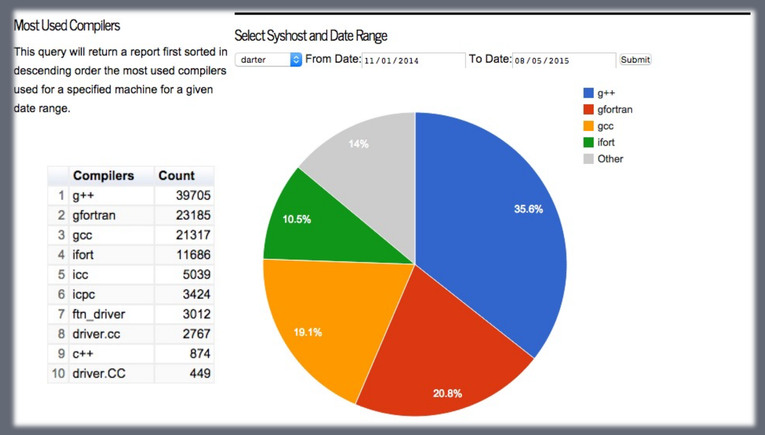


Fig3: XALT Transmission Methods

❖ Preliminary Reports

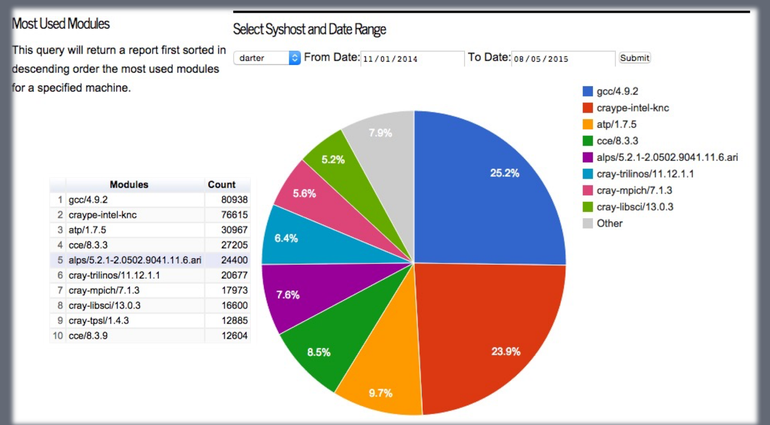
- Most Used Compilers:



	Compilers	Count
1	g++	39705
2	gfortran	23185
3	gcc	21317
4	ifort	11686
5	icc	5039
6	icpc	3424
7	ftn_driver	3012
8	driver.cc	2767
9	c++	874
10	driver.CC	449

Fig4: Most Used Compiler on Darter

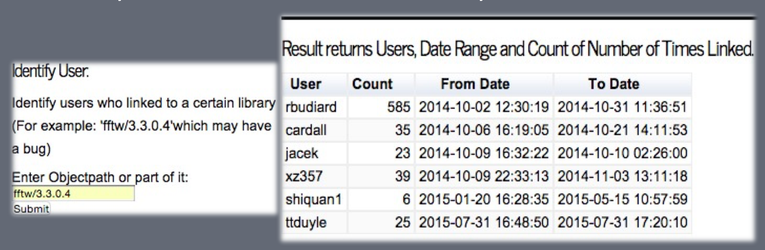
- Most Used Modules:



	Modules	Count
1	gcc/4.9.2	80938
2	craype-intel-knc	76615
3	atp/1.7.5	30967
4	cce/8.3.3	27205
5	alps/5.2.1-2.0502.9041.11.6.ari	24400
6	cray-trilinos/11.12.1.1	20677
7	cray-mpich/7.1.3	17973
8	cray-libsci/13.0.3	16600
9	cray-tpsl/1.4.3	12885
10	cce/8.3.9	12604

Fig5: Most Used Modules on Darter

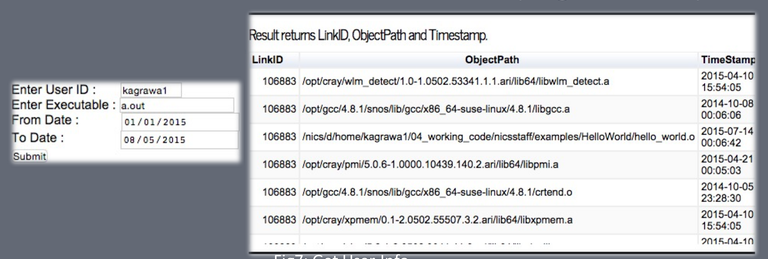
- Identify Users linked to a certain library:



User	Count	From Date	To Date
rbudiard	585	2014-10-02 12:30:19	2014-10-31 11:36:51
cardall	35	2014-10-06 16:19:05	2014-10-21 14:11:53
jacek	23	2014-10-09 16:32:22	2014-10-10 02:26:00
xz357	39	2014-10-09 22:33:13	2014-11-03 13:11:18
shiquan1	6	2015-01-20 16:28:35	2015-05-15 10:57:59
ttduyle	25	2015-07-31 16:48:50	2015-07-31 17:20:10

Fig6: Identify User linked to a certain library

- Get User Info: How did a user build their program in the past.



LinkID	ObjectPath	TimeStamp
106883	/opt/cray/wlm_detect/1.0-1.0502.53341.1.1.ari/lib64/libwlm_detect.a	2015-04-10 15:54:05
106883	/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/libgcc.a	2014-10-08 00:06:06
106883	/nics/d/home/kagrawa1/04_working_code/nicsstaff/examples/HelloWorld/hello_world.c	2015-07-14 00:06:42
106883	/opt/cray/pmi/5.0.6-1.0000.10439.140.2.ari/lib64/libpmi.a	2015-04-21 00:05:03
106883	/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/crtend.o	2014-10-05 23:28:30
106883	/opt/cray/xpmem/0.1-2.0502.55507.3.2.ari/lib64/libxpmem.a	2015-04-10 15:54:05

Fig7: Get User Info

❖ Current Capabilities

- Track if maintained library is used and how often.
- Identify users and code that used a buggy library.
- Support tracking of both static and dynamic libraries.
- Track if center provided packages are used more or less than user-installed packages.
- Track how many users and projects use a library or executable.
- Identify applications that are using deprecated libraries or just identify old binaries.
- Provide information on how an executable was built (provenance data).

❖ Future Work

- Function tracking at link time – tracking of function call resolved by libraries external to user code.
- Runtime environment check against compile-time environment – XALT will detect runtime differences with compile time environment and warn users.
- Real Time Portal to access above listed reports.

❖ Acknowledgement

- Work was supported by the NSF award 1339690 entitled "Collaborative Research: SI2-SSE: XALT: Understanding the Software Needs of High End Computer Users".
- Resources supported by University of Tennessee's Joint Institute of Computational Sciences and Texas Advanced Computing Center (TACC) at University of Texas at Austin.

❖ Download XALT on below links OR scan this >>

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

>> Watch this video

❖ Learn more about XALT from Dr. Mark Fahey/ Dr. Robert McLay.