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 30, 2020

XALT: Outline

- What is XALT and what it is not?
- How it works
- What is new with XALT?
- What can you do with it?
- How can I help you?

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
- Function Tracking

Design Goals

- Be extremely light-weight
- How many use a library or application?
- What functions are users calling in system modules
- 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
- Optionally add codes that executes before main() and after main() completes

Design: Transmission to DB

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

Lmod to XALT connection

- Optional support to connect paths to modules
- 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.

Using XALT Data

- Targetted Outreach: Who will be affected
- Largemem Queue Overuse
- XALT and TACC-Stats
- Who is running NWChem or ...?
- Function Tracking: Who or What is using MPI-3?

Who is using MPI-3: MPI_I*

Function Name	N Users	N Progs
MPI_Ibarrier	8	4
MPI_Ialltoall	24	4
MPI_Ineighbor_alltoall	4	3

What is using MPI-3: MPI_Ibarrier()

Libraries using MPI_Ibarrier()

Library
libhmlp.so
libparmetis.so
libfmpich.so.12.0.0
libfmpich.so.10.0.0

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;
```

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=.*'],
   ]
```

Sampling Non-MPI programs

- XALT has sampling rules (site configurable!)
- TACC rules are:
- 0 mins $<$ 30 mins $\Rightarrow$ 0.01% recorded
- 30 mins $<$ 120 mins $\Rightarrow$ 1% recorded
- 120 mins $<$ ∞ $\Rightarrow$ 100% recorded
- Can now track/sample perl, awk, sed, gzip etc

Sampling MPI programs

- Some user are using many short MPI programs to train Deep Learning engine
- TACC rules are:
- Task counts $<$ 256 tasks are sampled with the same rules as for scalar programs.
- Task counts $>=$ 256 task are always recorded.
- Need to Capture long running MPI prog that never end.

What is new with XALT?

- Tracking R, Python, MATLAB
- Signal handler
- Optionally Track GPU Usage
- Track Singularity Container Usage
- Removed two system calls for improved speed

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.
- Where was it build.

Example of xalt_extract_record output

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

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

Build_CWD                    /home/user/t/hello
Build_Epoch                  1510257139.4624
Build_LMFILES               /opt/apps/modulefiles/in-
tel/17.0.4.lua:...
Build_LOADEDMODULES    intel/18.0.4:impi/18.0.3: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                  Fri Jun 09 13:52:19 2019
Build_host             c455-011.stampede2.tacc.utexas.ed
XALT_Version                 2.7
```

XALT Signal Handler

- Program that fail with SEGV, ILL, FPE ... produce an XALT record.
- But not SIGINT.
- The signal handlers are assigned before main() $\Rightarrow$ Doesn't interfere with user handlers

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.

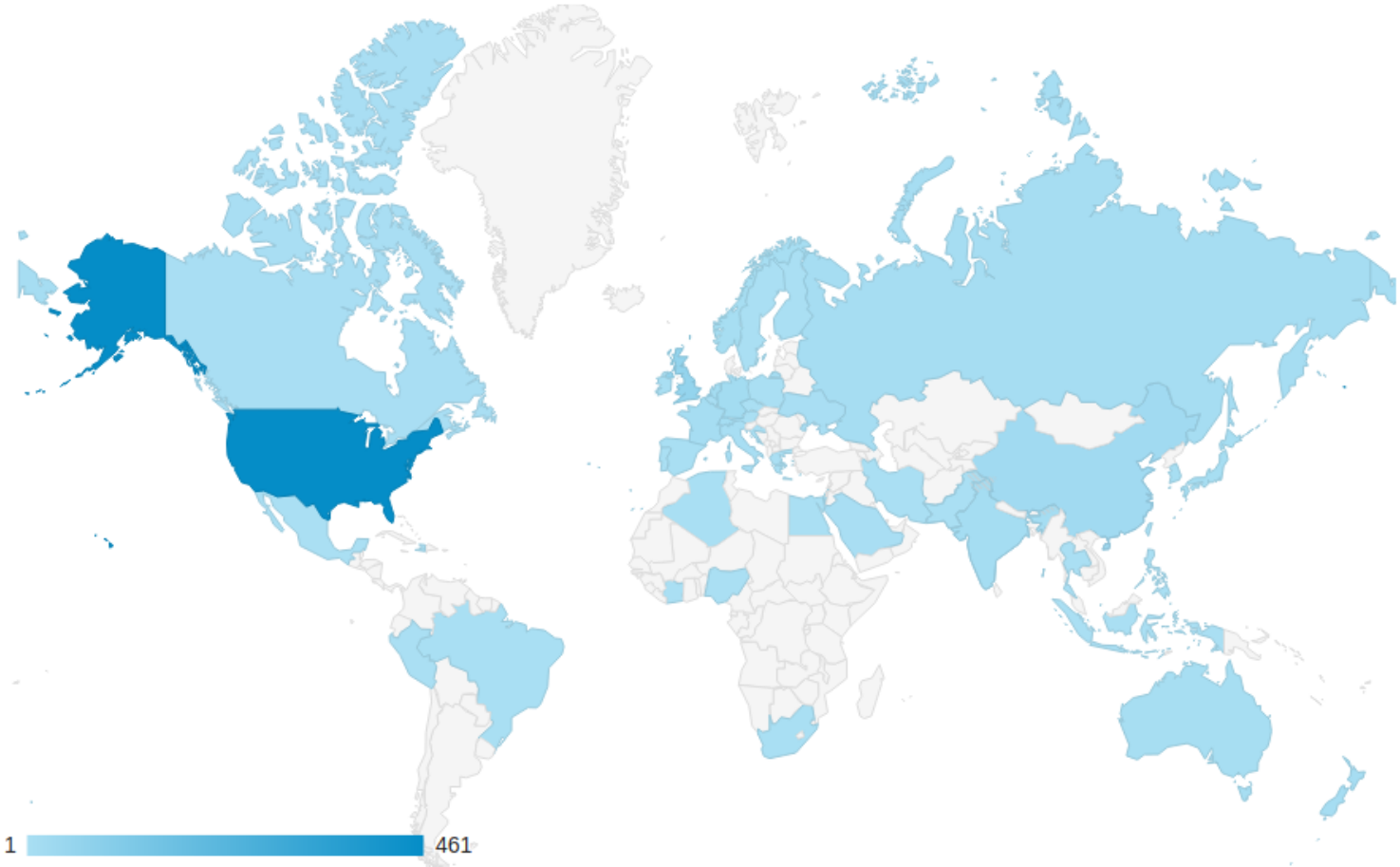
Removed two system calls during Tracking

- Reads /proc/$PID/maps instead of running ldd.
- Uses the vendor note to hold the XALT watermark.
- Improves XALT penalty from 0.01 to 0.001 seconds.

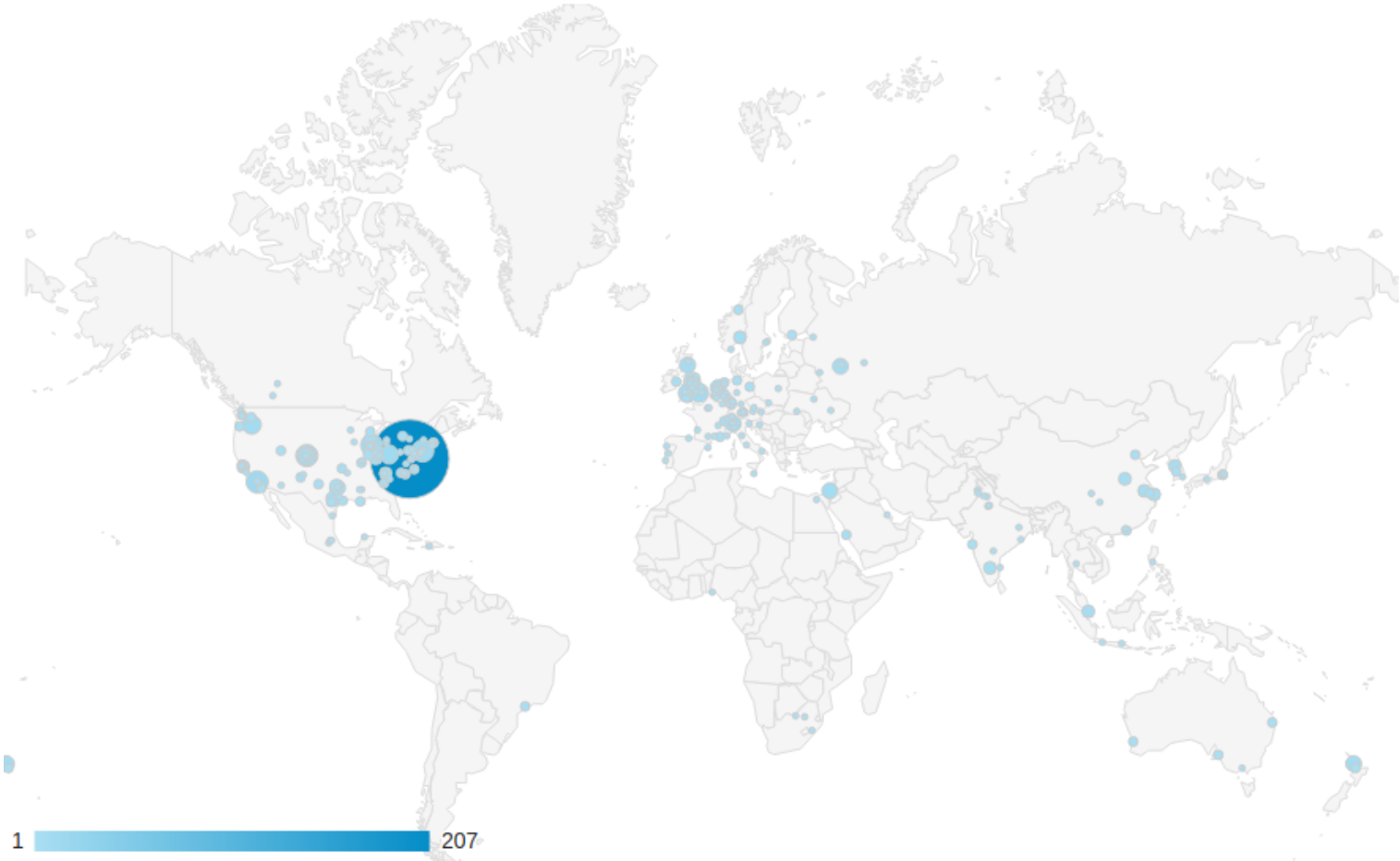
libuuid and libcrypto

- A user's code segfaulted with XALT
- Their code had a routine named *random*.
- libuuid has routine named *random*!
- XALT no longer links in libuuid with user code.

XALT Doc usage by Country



XALT Doc usage by City



Conclusion

XALT

- 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