Community Use of XALT in Its First Year in Production

HUST 2015
Austin, TX

Reuben D. Budiardja
National Institute for Computational Sciences
The University of Tennessee

with Mark Fahey (ANL), Robert McLay (TACC),
Prasad Maddumage Don (FSU),
Bilel Hadri (KAUST), Doug James (TACC)

https://github.com/Fahey-McLay/xalt

Talk Outline

- Introduction to XALT
  - Motivation
  - How It Works

Getting Data Out of XALT

  - Compilers, Libraries, Executables Usage Reports
  - Other Use Cases

- New Functionality
  - Function Tracking
  - GUI (Web)-Based Reports

  User Software Provenance

Introduction to XALT

Motivation

Most computing center needs to answer the questions:

- How many users and projects use a particular library or executable ?

  How many users use which compilers ?
- Which center provided packages are used often ? and which one are never used ?
- Which users or applications still use old version of certain library, compiler, or executable ?

  Are there any widely used user-installed package that a center should provide instead ?

XALT is a tool to collect accurate, detailed, and continuous job-level and link-time data, and store them in a database.

XALT is a tool to collect accurate, detailed, and continuous job-level and link-time data, and store them in a database.

XALT collects information to answer questions on software usage

Goals

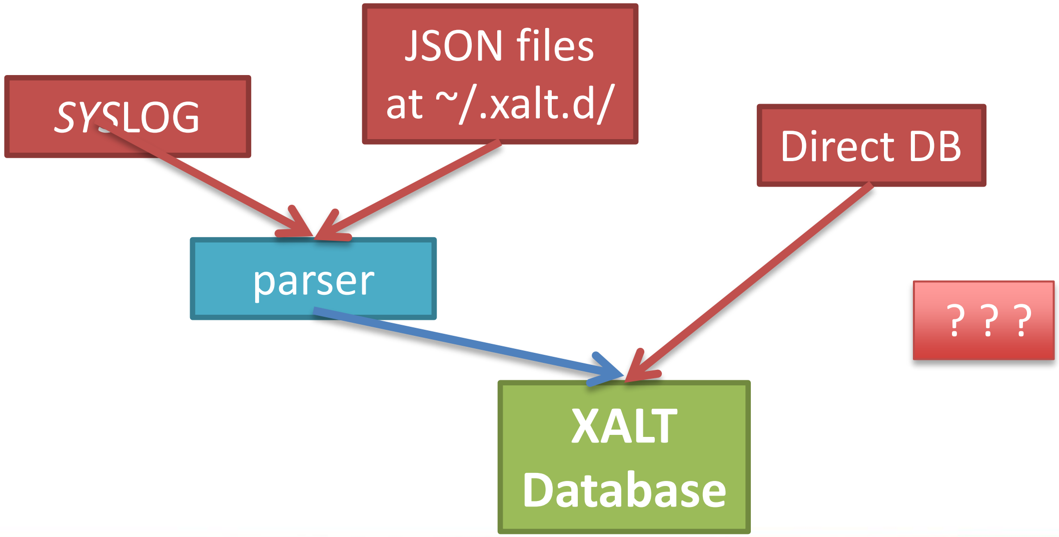
- ***Automatic, continuous*** census of libraries and applications
- Collect job-level and link-time level data for subsequent analytics

  Must be transparent to user, avoid impacting the user experience
- Must work seamlessly on any system: workstation, cluster, high-end supercomputer
- Must be a lightweight solution

Approach: Link-time Level

Intercept linker at link-time:

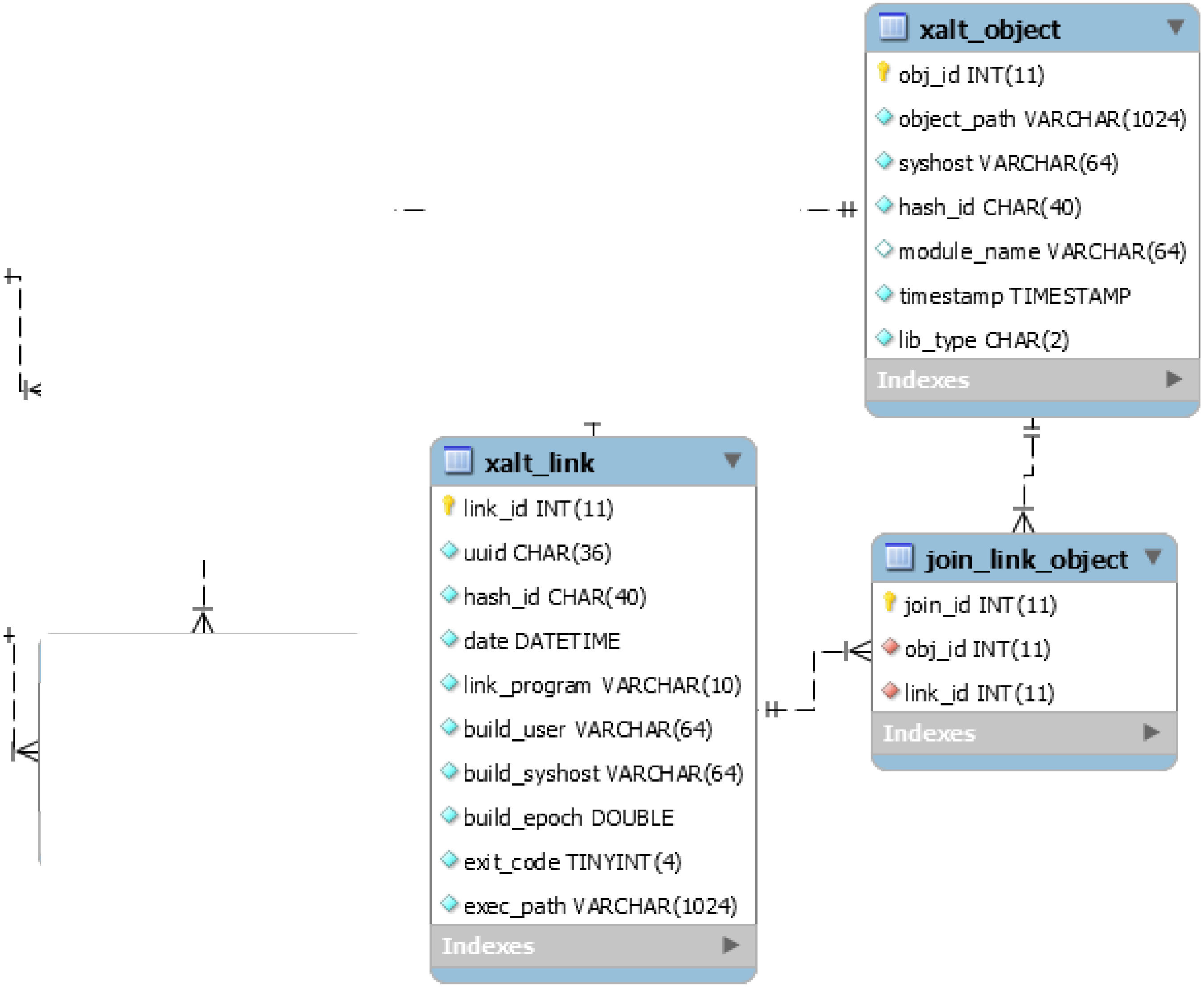
- Wrap the (GNU) linker (ld) and parse the command line
    - Capture only the object files actually linked with the executable
- Stores the results using a chosen *transmission style*
- Insert an XALT's ELF section header to the executable



Approach: Execution-time Level

Intercept job launcher to get execution environment:

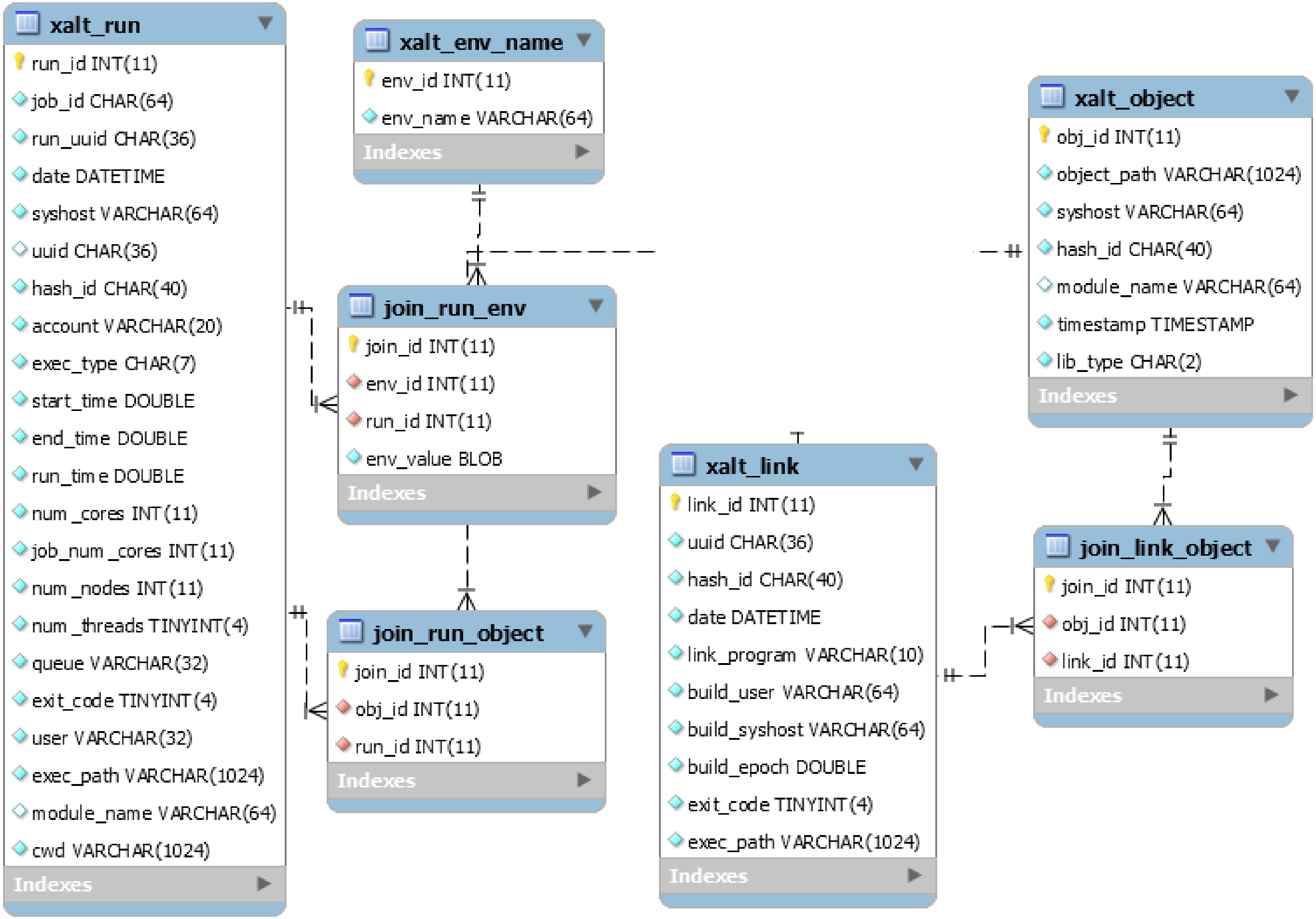
- Wrap job-launcher (aprun, ibrun, mpirun, …) with a corresponding script
- Extract previously inserted XALT's ELF header (if any)
- Extract environment variables
  - Job-specific environment (e.g. PBS_JOBID, etc)
  - Dynamics libraries loaded at runtime
  - Record job start and end time



xalt_object
obj_id INT(11)
object_path VARCHAR(1024)
syshost VARCHAR(64)
hash_id CHAR(40)
module_name VARCHAR(64)
timestamp TIMESTAMP
lib_type CHAR(2)
Indexes

xalt_link
link_id INT(11)
uuid CHAR(36)
hash_id CHAR(40)
date DATETIME
link_program VARCHAR(10)
build_user VARCHAR(64)
build_syshost VARCHAR(64)
build_epoch DOUBLE
exit_code TINYINT(4)
exec_path VARCHAR(1024)
Indexes

join_link_object
join_id INT(11)
obj_id INT(11)
link_id INT(11)
Indexes



xalt_run
- run_id INT(11)
- job_id CHAR(64)
- run_uuid CHAR(36)
- date DATETIME
- syshost VARCHAR(64)
- uuid CHAR(36)
- hash_id CHAR(40)
- account VARCHAR(20)
- exec_type CHAR(7)
- start_time DOUBLE
- end_time DOUBLE
- run_time DOUBLE
- num_cores INT(11)
- job_num_cores INT(11)
- num_nodes INT(11)
- num_threads TINYINT(4)
- queue VARCHAR(32)
- exit_code TINYINT(4)
- user VARCHAR(32)
- exec_path VARCHAR(1024)
- module_name VARCHAR(64)
- cwd VARCHAR(1024)

xalt_env_name
- env_id INT(11)
- env_name VARCHAR(64)

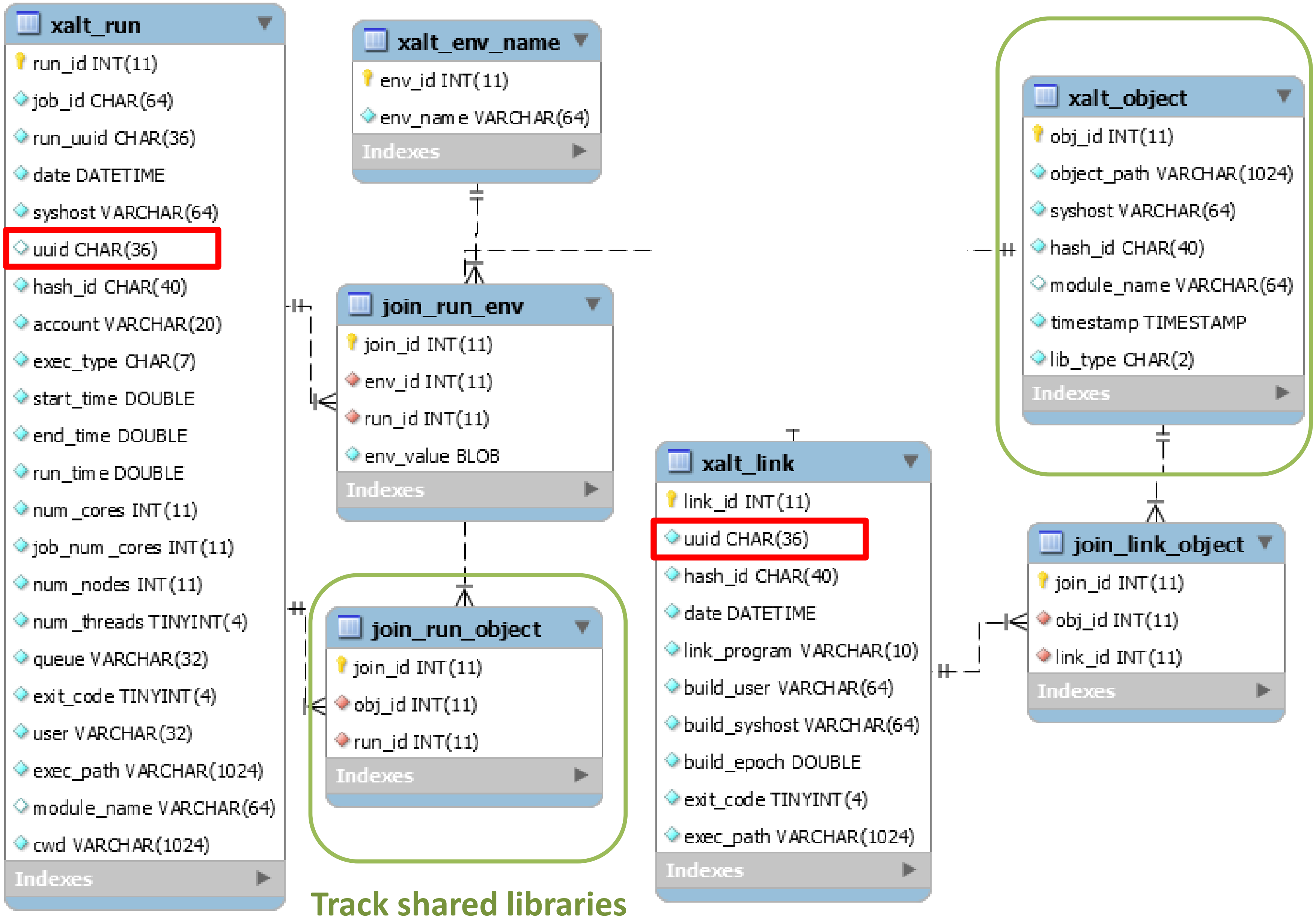
join_run_env
- join_id INT(11)
- env_id INT(11)
- run_id INT(11)
- env_value BLOB

join_run_object
- join_id INT(11)
- obj_id INT(11)
- run_id INT(11)

xalt_object
- obj_id INT(11)
- object_path VARCHAR(1024)
- syshost VARCHAR(64)
- hash_id CHAR(40)
- module_name VARCHAR(64)
- timestamp TIMESTAMP
- lib_type CHAR(2)

xalt_link
- link_id INT(11)
- uuid CHAR(36)
- hash_id CHAR(40)
- date DATETIME
- link_program VARCHAR(10)
- build_user VARCHAR(64)
- build_syshost VARCHAR(64)
- build_epoch DOUBLE
- exit_code TINYINT(4)
- exec_path VARCHAR(1024)

join_link_object
- join_id INT(11)
- obj_id INT(11)
- link_id INT(11)



xalt_run
- run_id INT(11)
- job_id CHAR(64)
- run_uuid CHAR(36)
- date DATETIME
- syshost VARCHAR(64)
- uuid CHAR(36)
- hash_id CHAR(40)
- account VARCHAR(20)
- exec_type CHAR(7)
- start_time DOUBLE
- end_time DOUBLE
- run_time DOUBLE
- num_cores INT(11)
- job_num_cores INT(11)
- num_nodes INT(11)
- num_threads TINYINT(4)
- queue VARCHAR(32)
- exit_code TINYINT(4)
- user VARCHAR(32)
- exec_path VARCHAR(1024)
- module_name VARCHAR(64)
- cwd VARCHAR(1024)

Indexes

xalt_env_name
- env_id INT(11)
- env_name VARCHAR(64)

Indexes

join_run_env
- join_id INT(11)
- env_id INT(11)
- run_id INT(11)
- env_value BLOB

Indexes

join_run_object
- join_id INT(11)
- obj_id INT(11)
- run_id INT(11)

Indexes

xalt_link
- link_id INT(11)
- uuid CHAR(36)
- hash_id CHAR(40)
- date DATETIME
- link_program VARCHAR(10)
- build_user VARCHAR(64)
- build_syshost VARCHAR(64)
- build_epoch DOUBLE
- exit_code TINYINT(4)
- exec_path VARCHAR(1024)

Indexes

xalt_object
- obj_id INT(11)
- object_path VARCHAR(1024)
- syshost VARCHAR(64)
- hash_id CHAR(40)
- module_name VARCHAR(64)
- timestamp TIMESTAMP
- lib_type CHAR(2)

Indexes

join_link_object
- join_id INT(11)
- obj_id INT(11)
- link_id INT(11)

Indexes

Track shared libraries

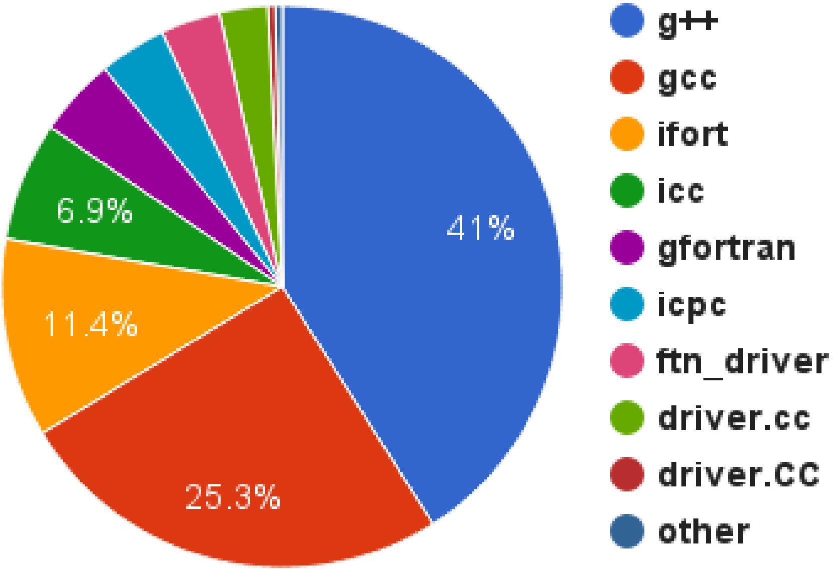
Getting Data Out of XALT

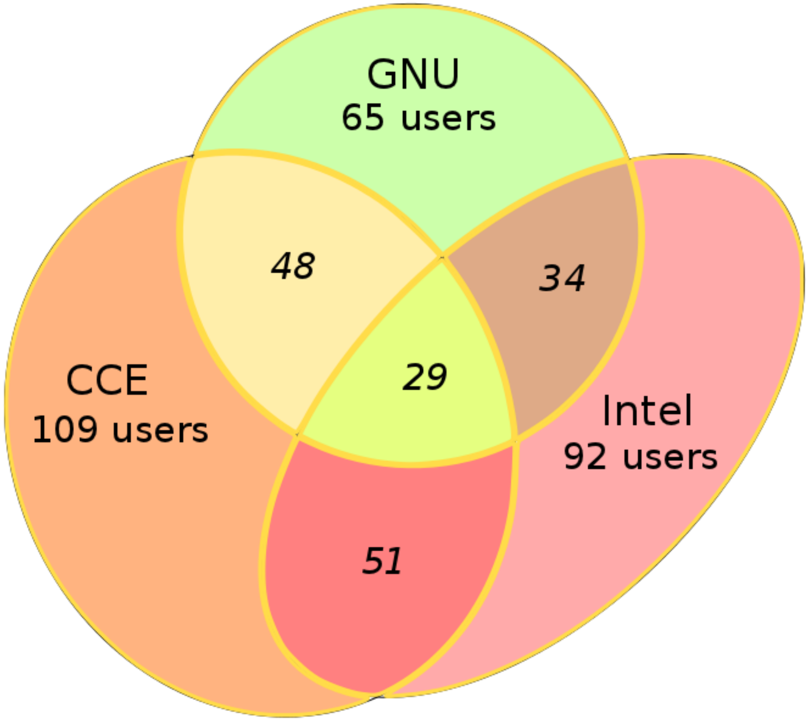
Community Usage Reports

Compiler Usage

- XALT stores “link program”: the program that calls the linker
  - A proxy for the compiler → main() compiler
  - Will miss mixed language compilation
- Can associate “compiler” with every linking event

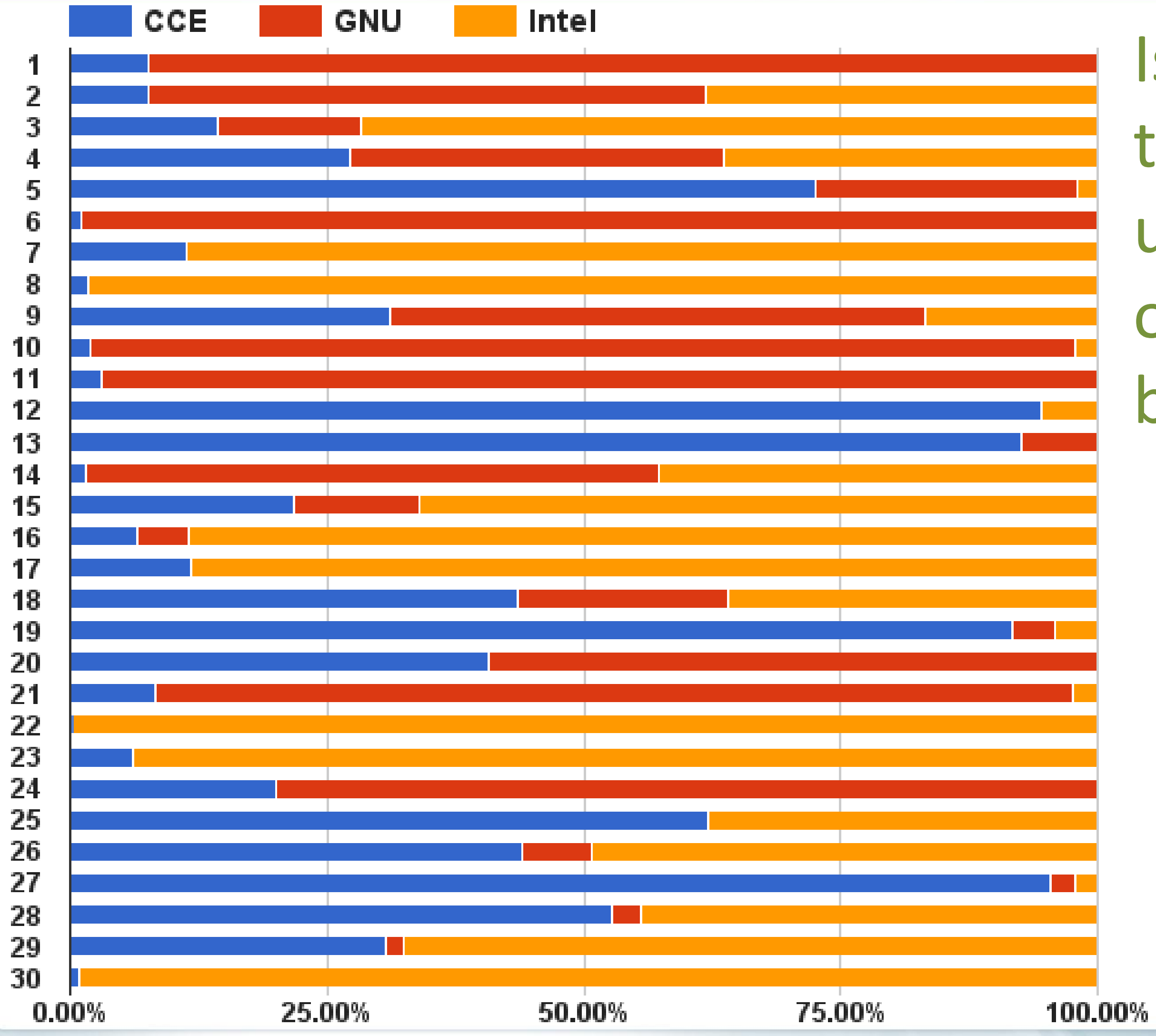
Compiler Usage on Darter





```
SELECT link_program, count(*) as count
  FROM xalt_link
 WHERE build_syshost = [syshost]
 GROUP BY link_program ORDER BY count desc
```

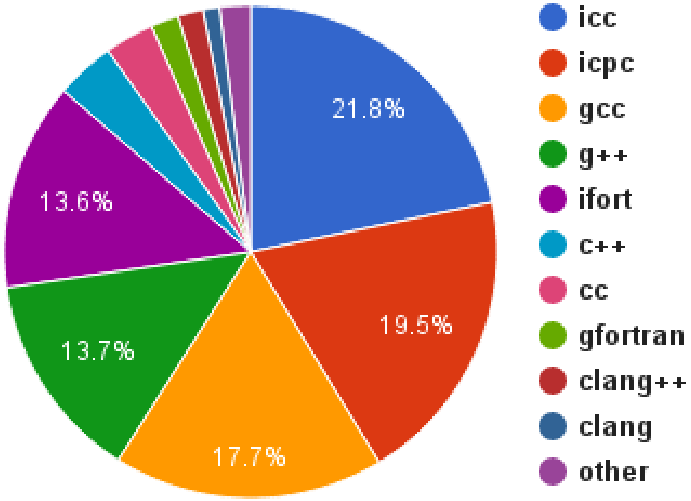
Compiler Usage Ratio per User



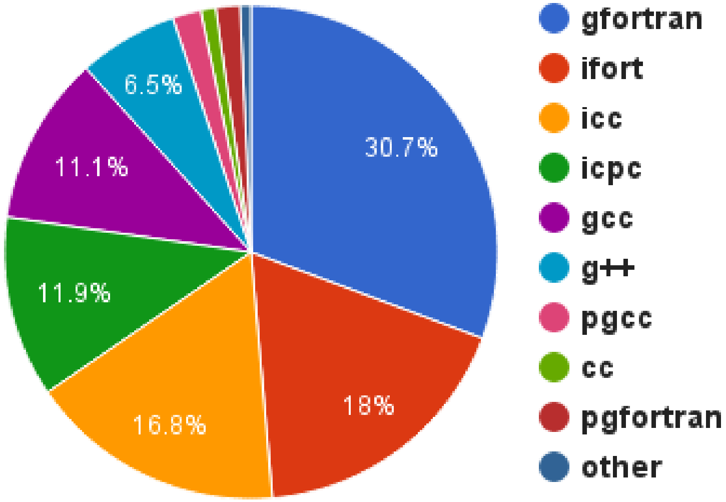
Is there a way to tell if someone used a compiler once (or a little), before giving up ?

Compiler Usage: TACC, FSU, KAUST

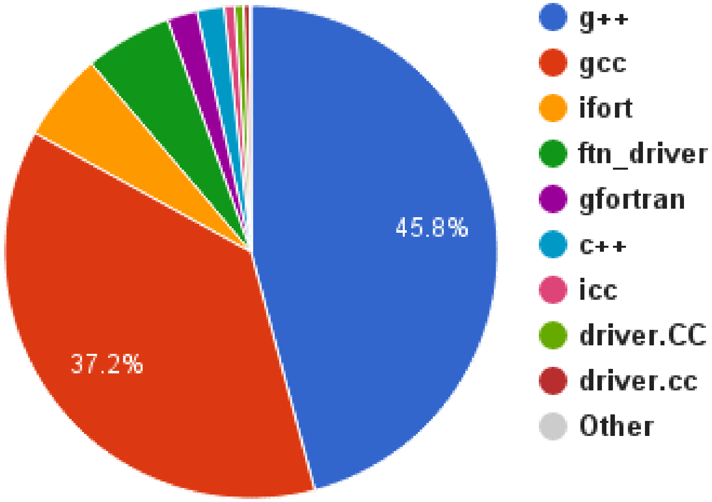
TACC



FSU



KAUST

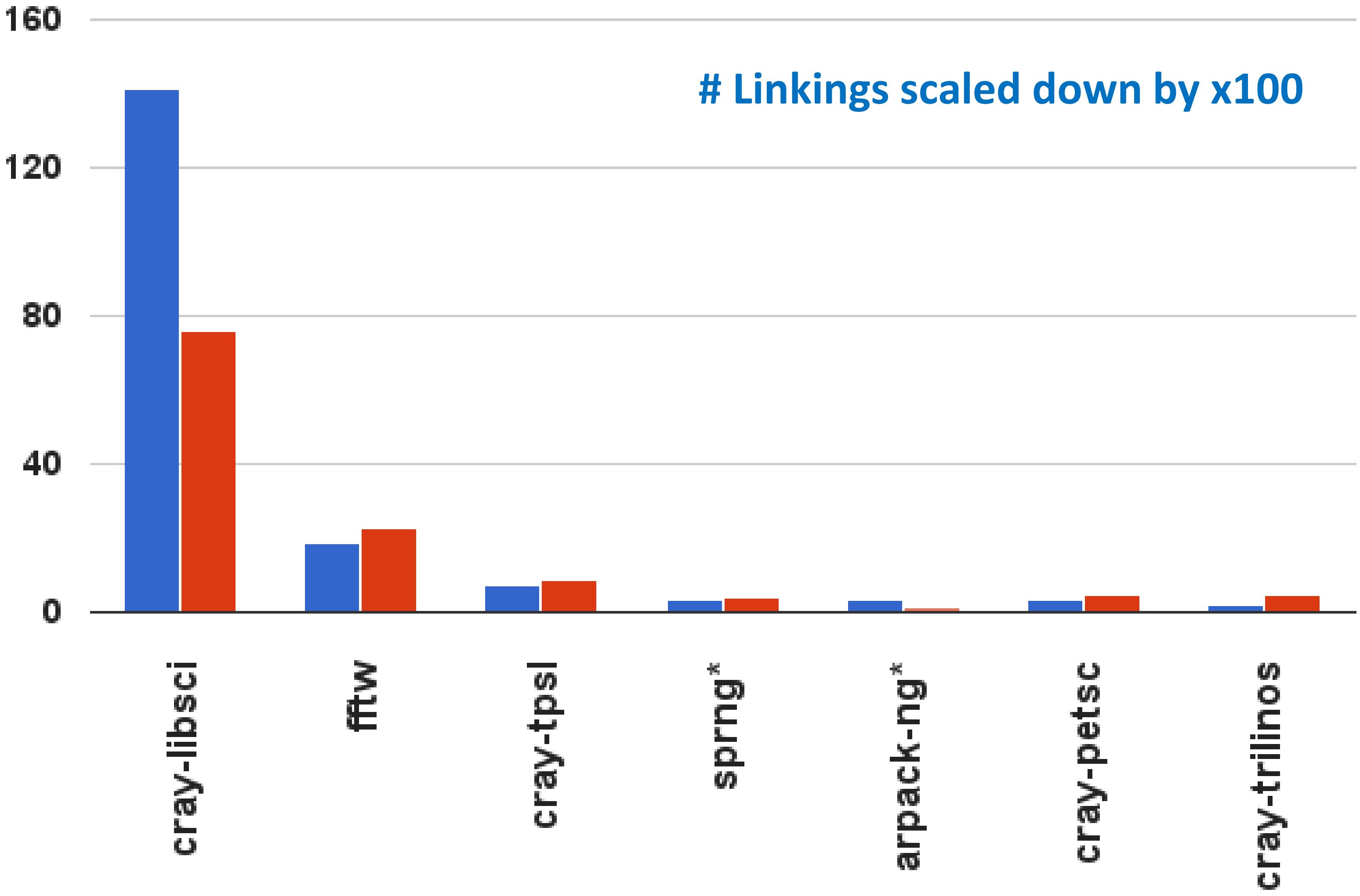


Most Used Libraries

- What is “the most used” ?
  - By the number of linkings
  - By the number of unique users
- Use “module name” to identify library
  - Multiple object files may be associated with a module
  - Likely these libraries are provided via modulefile by vendor or center’s staff

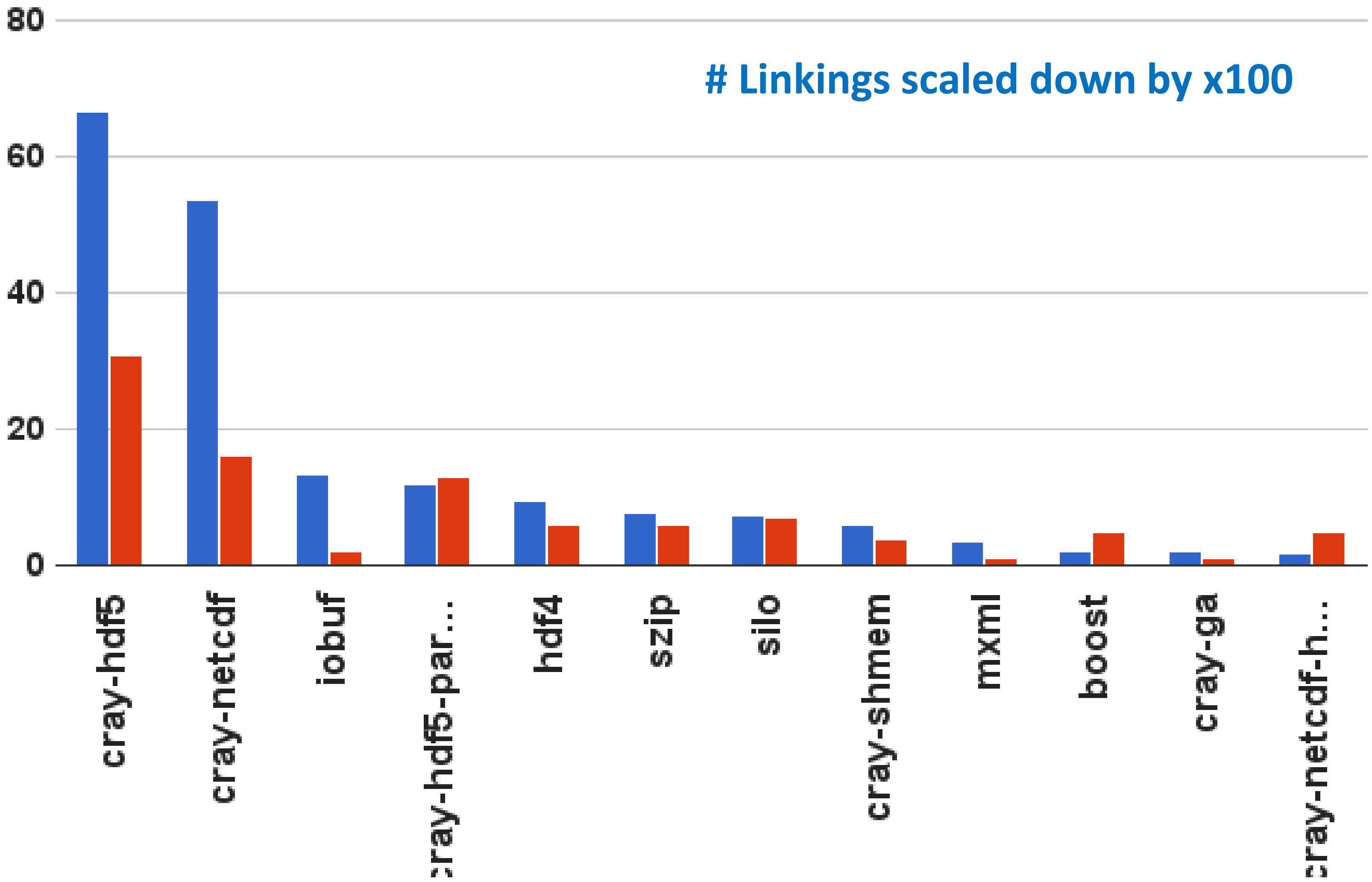
    Resistance to path changes as long as ReverseMap is maintained
- Script: contrib/library_usage.py

Most Used Libraries: Numerical



# Linkings scaled down by x100

Most Used Libraries: Prog. & I/O

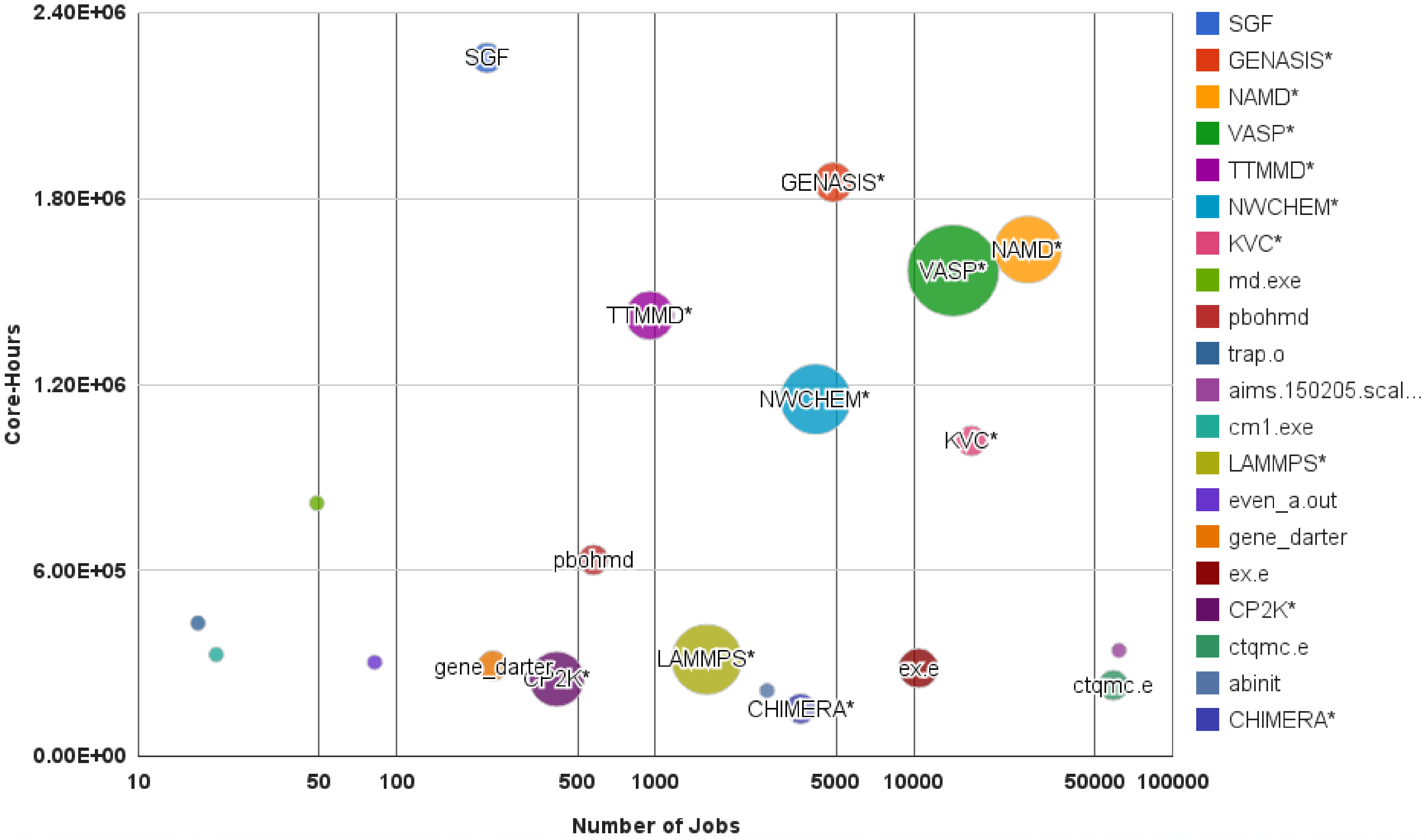


# Linkings scaled down by x100

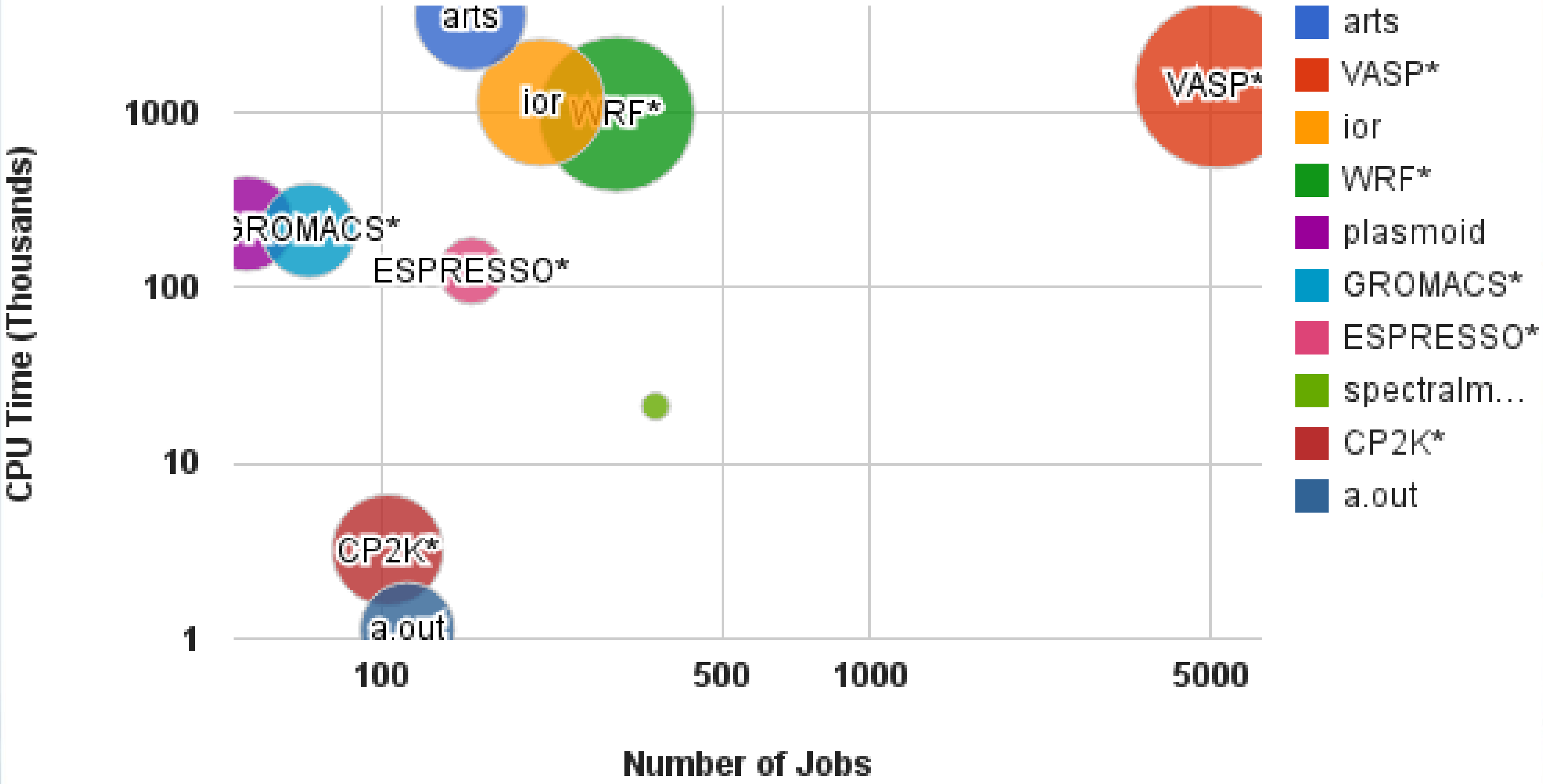
Top Executables

- Track only how much time spent by the parallel job
  - Not the entire job script
  - Can be correlated with other accounting to get the ratio of the parallel job over the entire job script
- Track the actual number of compute cores used in the parallel job
  - Done by parsing the argument given to parallel launcher
- Can show how the launched executable was built → provenance data

Top Executables



Top Executables: KAUST



Software Pruning

- How or when to remove software (version) on the system ?
  - Because newer versions are available
  - Because of lack of use
  - To free up disk space and/or support time
- XALT can provide data-driven decision

  Show when the last time each library was used (linked against), and by whom (user)

  - Allow for targeted notification to users (to upgrade version, migrate to different library, etc)

New Functionality

Function Tracking

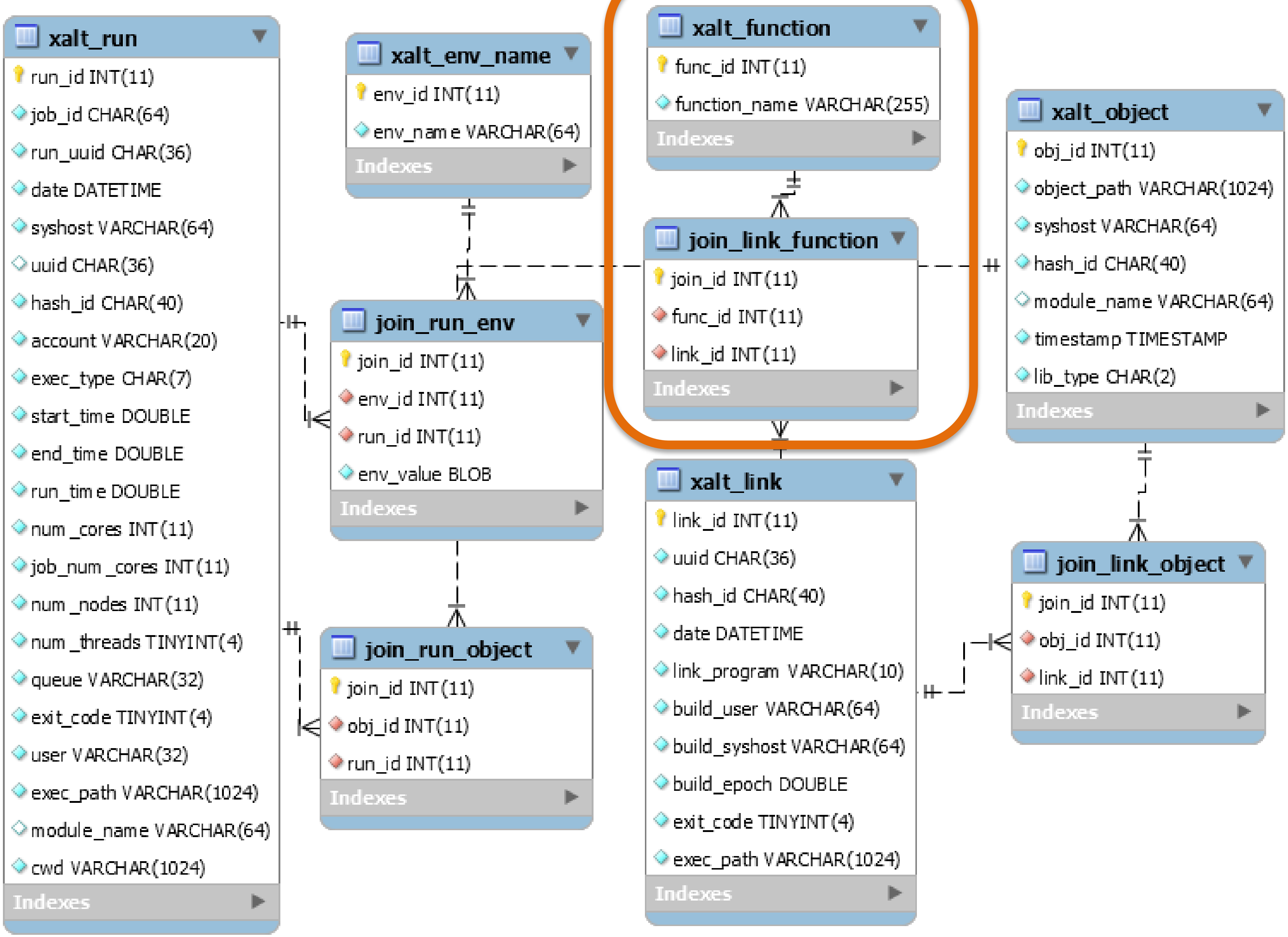
- Recently added functionality (version >= 0.7.0)
- Only track functions (a.k.a. subroutines / symbol names) that are resolved by external libraries
  - Does not track user defined functions

    Does not track auxiliary functions in libraries
- Currently does not track which library resolves the functions

    Although this can be done heuristically after the fact

Function Tracking (2)

- Collect the list of library / object files whose functions we are interested in tracking
  - Generated by traversing the directories of library files in modulefiles (typically used as argument to “-L” linker flag)← already in ReverseMap file



xalt_run
- run_id INT(11)
- job_id CHAR(64)
- run_uuid CHAR(36)
- date DATETIME
- syshost VARCHAR(64)
- uuid CHAR(36)
- hash_id CHAR(40)
- account VARCHAR(20)
- exec_type CHAR(7)
- start_time DOUBLE
- end_time DOUBLE
- run_time DOUBLE
- num_cores INT(11)
- job_num_cores INT(11)
- num_nodes INT(11)
- num_threads TINYINT(4)
- queue VARCHAR(32)
- exit_code TINYINT(4)
- user VARCHAR(32)
- exec_path VARCHAR(1024)
- module_name VARCHAR(64)
- cwd VARCHAR(1024)

Indexes

xalt_env_name
- env_id INT(11)
- env_name VARCHAR(64)

Indexes

join_run_env
- join_id INT(11)
- env_id INT(11)
- run_id INT(11)
- env_value BLOB

Indexes

join_run_object
- join_id INT(11)
- obj_id INT(11)
- run_id INT(11)

Indexes

xalt_function
- func_id INT(11)
- function_name VARCHAR(255)

Indexes

join_link_function
- join_id INT(11)
- func_id INT(11)
- link_id INT(11)

Indexes

xalt_link
- link_id INT(11)
- uuid CHAR(36)
- hash_id CHAR(40)
- date DATETIME
- link_program VARCHAR(10)
- build_user VARCHAR(64)
- build_syshost VARCHAR(64)
- build_epoch DOUBLE
- exit_code TINYINT(4)
- exec_path VARCHAR(1024)

Indexes

xalt_object
- obj_id INT(11)
- object_path VARCHAR(1024)
- syshost VARCHAR(64)
- hash_id CHAR(40)
- module_name VARCHAR(64)
- timestamp TIMESTAMP
- lib_type CHAR(2)

Indexes

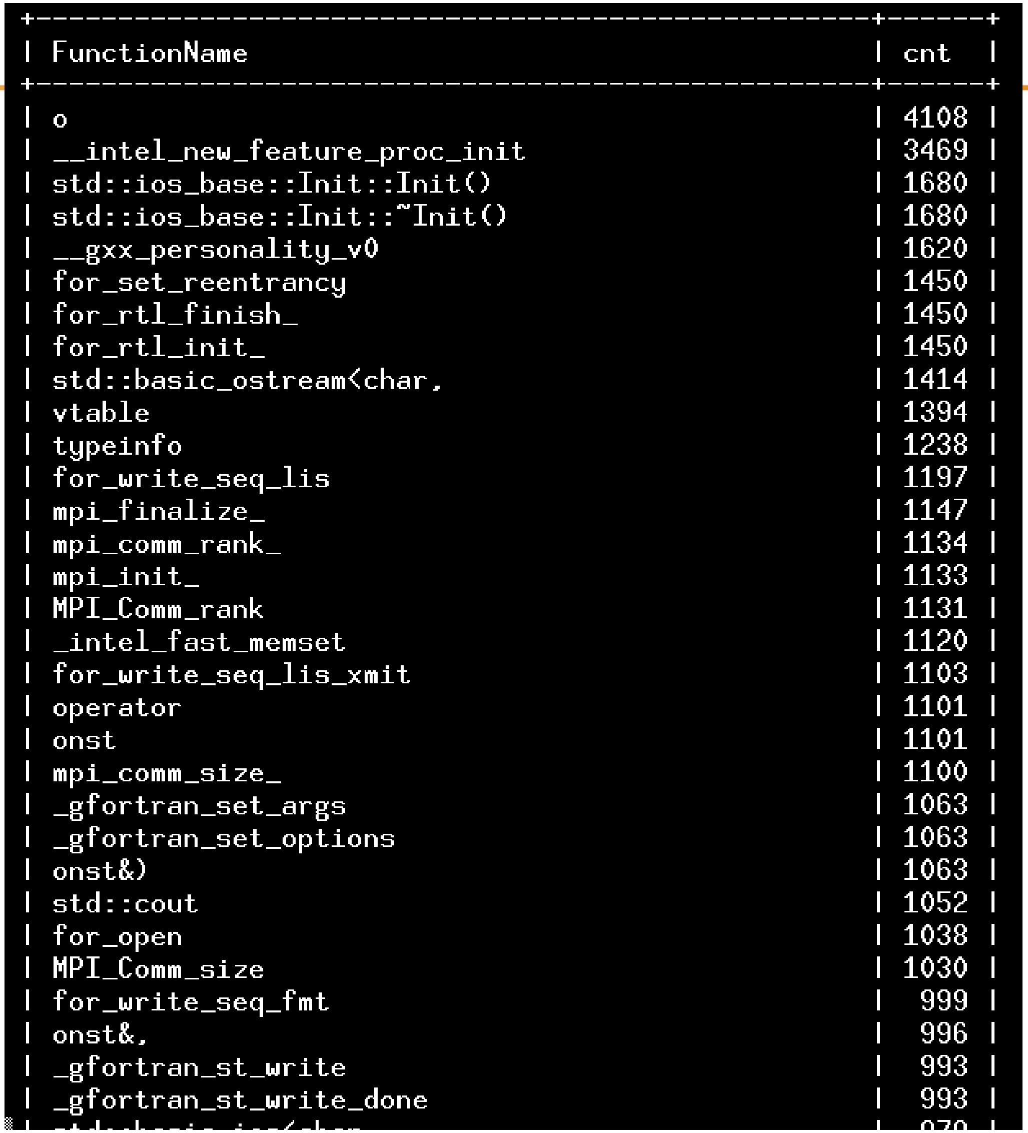
join_link_object
- join_id INT(11)
- obj_id INT(11)
- link_id INT(11)

Indexes

Example Query

- Most called functions

```
SELECT trim(function_name),
          count(*)
  FROM xalt_link xl,
          join_link_function lf,
          xalt_function xf
WHERE build_syshost = 'darter'
    AND xl.link_id = lf.link_id
    AND lf.func_id = xf.func_id
GROUP BY function_name
ORDER BY cnt DESC
LIMIT 100
```

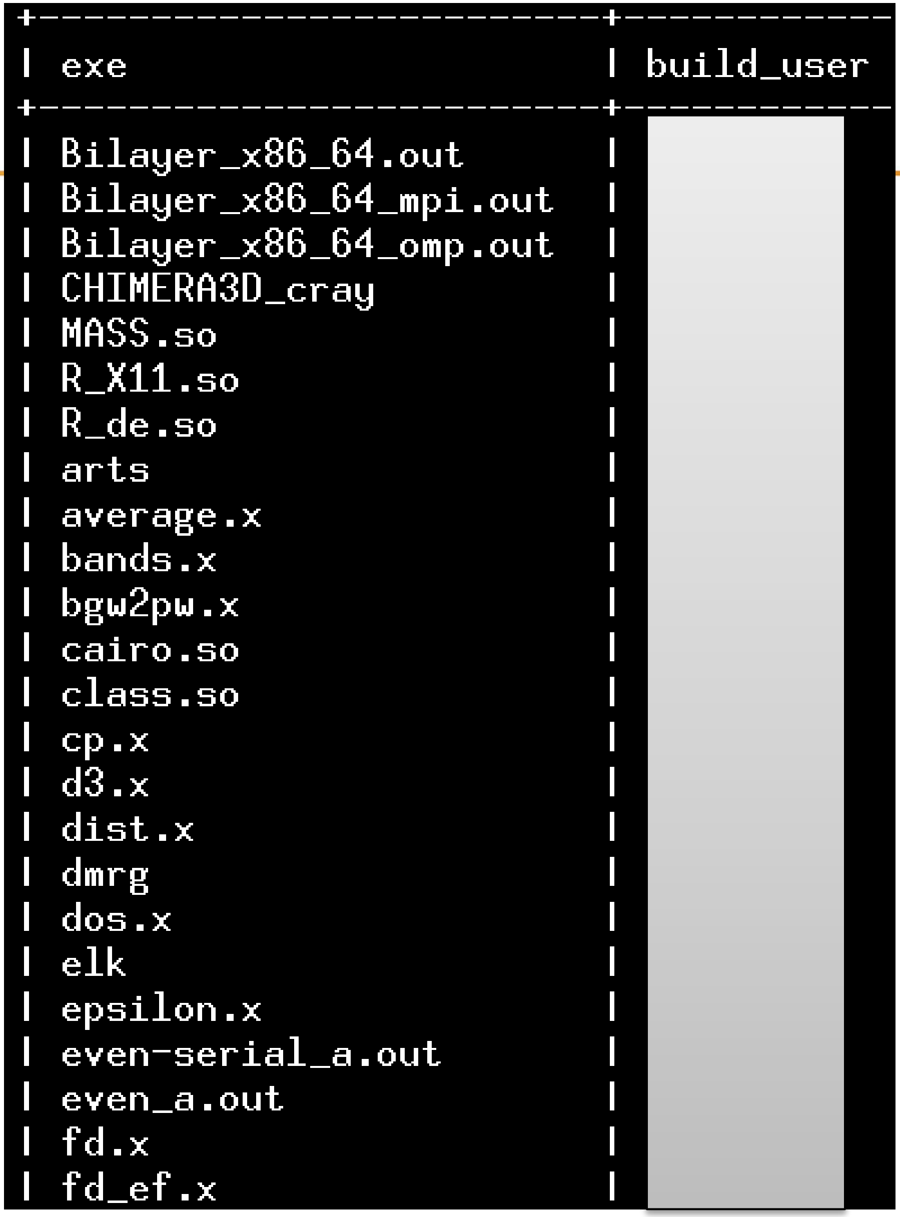


```
+----------------------------------------------------+------+
| FunctionName                                       | cnt  |
+----------------------------------------------------+------+
| o                                                  | 4108 |
| __intel_new_feature_proc_init                      | 3469 |
| std::ios_base::Init::Init()                        | 1680 |
| std::ios_base::Init::~Init()                       | 1680 |
| __gxx_personality_v0                               | 1620 |
| for_set_reentrancy                                 | 1450 |
| for_rtl_finish_                                    | 1450 |
| for_rtl_init_                                      | 1450 |
| std::basic_ostream<char,                           | 1414 |
| vtable                                             | 1394 |
| typeinfo                                           | 1238 |
| for_write_seq_lis                                  | 1197 |
| mpi_finalize_                                      | 1147 |
| mpi_comm_rank_                                     | 1134 |
| mpi_init_                                          | 1133 |
| MPI_Comm_rank                                      | 1131 |
| _intel_fast_memset                                 | 1120 |
| for_write_seq_lis_xmit                             | 1103 |
| operator                                           | 1101 |
| onst                                               | 1101 |
| mpi_comm_size_                                     | 1100 |
| _gfortran_set_args                                 | 1063 |
| _gfortran_set_options                              | 1063 |
| onst&)                                             | 1063 |
| std::cout                                          | 1052 |
| for_open                                           | 1038 |
| MPI_Comm_size                                      | 1030 |
| for_write_seq_fmt                                  |  999 |
| onst&,                                             |  996 |
| _gfortran_st_write                                 |  993 |
| _gfortran_st_write_done                            |  993 |
| std::basic_ios<char                                |  979 |
```

Example Query

- BLAS’ mat-mul use

SELECT distinct(SUBSTRING_INDEX(
exec_path,'/',-1)) as exe,
build_user
FROM xalt_link xl,
join_link_function lf,
xalt_function xf
WHERE build_syshost = 'darter'
AND xl.link_id = lf.link_id
AND lf.func_id = xf.func_id
AND xf.function_name
LIKE '%gemm%'
GROUP BY exe



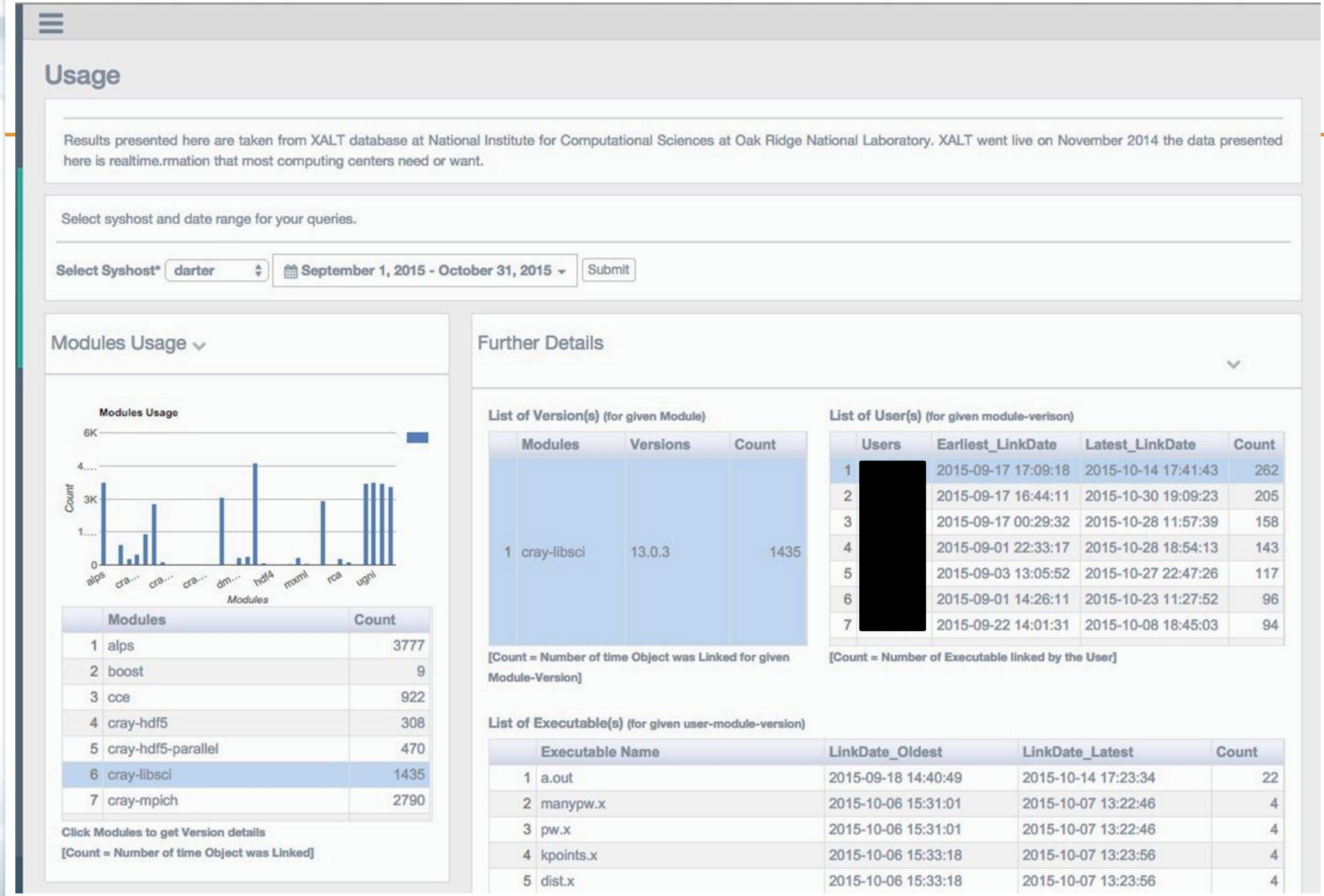
```
+--------------------------+------------
| exe                      | build_user
+--------------------------+------------
| Bilayer_x86_64.out       | [redacted]
| Bilayer_x86_64_mpi.out   |
| Bilayer_x86_64_omp.out   |
| CHIMERA3D_cray           |
| MASS.so                  |
| R_X11.so                 |
| R_de.so                  |
| arts                     |
| average.x                |
| bands.x                  |
| bgw2pw.x                 |
| cairo.so                 |
| class.so                 |
| cp.x                     |
| d3.x                     |
| dist.x                   |
| dmrg                     |
| dos.x                    |
| elk                      |
| epsilon.x                |
| even-serial_a.out        |
| even_a.out               |
| fd.x                     |
| fd_ef.x                  |
```

XALT Portal

A web interface to more easily get XALT data:

- Used by center's staff to easily get high level library, compiler, and executable usage
- From any of those "entry points", can drill-down to users associated with library/compiler/executable, and their jobs and job environment
- Can search who uses a particular library or executable

  Allow targeted notification in case of buggy library, retired versions, etc



Usage

Results presented here are taken from XALT database at National Institute for Computational Sciences at Oak Ridge National Laboratory. XALT went live on November 2014 the data presented here is realtime.rmation that most computing centers need or want.

Select syshost and date range for your queries.

Select Syshost* darter
September 1, 2015 - October 31, 2015

Modules Usage

	Modules	Count
1	alps	3777
2	boost	9
3	cce	922
4	cray-hdf5	308
5	cray-hdf5-parallel	470
6	cray-libsci	1435
7	cray-mpich	2790

Click Modules to get Version details
[Count = Number of time Object was Linked]

Further Details

List of Version(s) (for given Module)

	Modules	Versions	Count
1	cray-libsci	13.0.3	1435

[Count = Number of time Object was Linked for given Module-Version]

List of User(s) (for given module-verison)

	Users	Earliest_LinkDate	Latest_LinkDate	Count
1	[redacted]	2015-09-17 17:09:18	2015-10-14 17:41:43	262
2	[redacted]	2015-09-17 16:44:11	2015-10-30 19:09:23	205
3	[redacted]	2015-09-17 00:29:32	2015-10-28 11:57:39	158
4	[redacted]	2015-09-01 22:33:17	2015-10-28 18:54:13	143
5	[redacted]	2015-09-03 13:05:52	2015-10-27 22:47:26	117
6	[redacted]	2015-09-01 14:26:11	2015-10-23 11:27:52	96
7	[redacted]	2015-09-22 14:01:31	2015-10-08 18:45:03	94

[Count = Number of Executable linked by the User]

List of Executable(s) (for given user-module-version)

	Executable Name	LinkDate_Oldest	LinkDate_Latest	Count
1	a.out	2015-09-18 14:40:49	2015-10-14 17:23:34	22
2	manypw.x	2015-10-06 15:31:01	2015-10-07 13:22:46	4
3	pw.x	2015-10-06 15:31:01	2015-10-07 13:22:46	4
4	kpoints.x	2015-10-06 15:33:18	2015-10-07 13:23:56	4
5	dist.x	2015-10-06 15:33:18	2015-10-07 13:23:56	4

XALT Portal for User Provenance

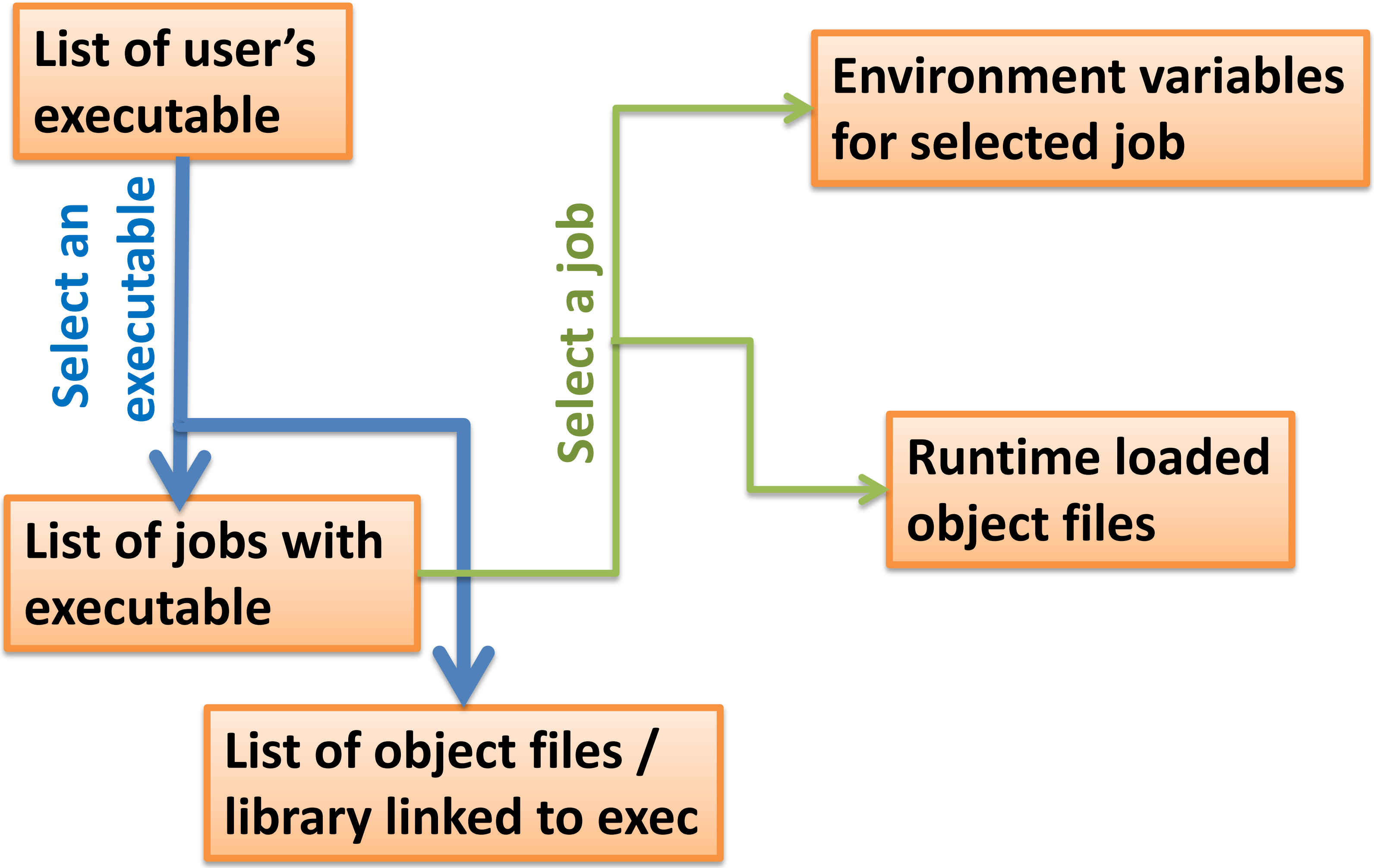
- “How did I build my exec x months ago ?”
  “What was the default MPI / compiler / libraryX at the time ?”
- Allow user to know the history and origin, i.e. “provenance”, of the software they run

  Different type of users:
  - Run their own executable

    Run executable provided by the Center

    Run executable built by another user
- Helps with reproducibility of research conducted with such software

User Provenance



List of user's executable

Select an executable

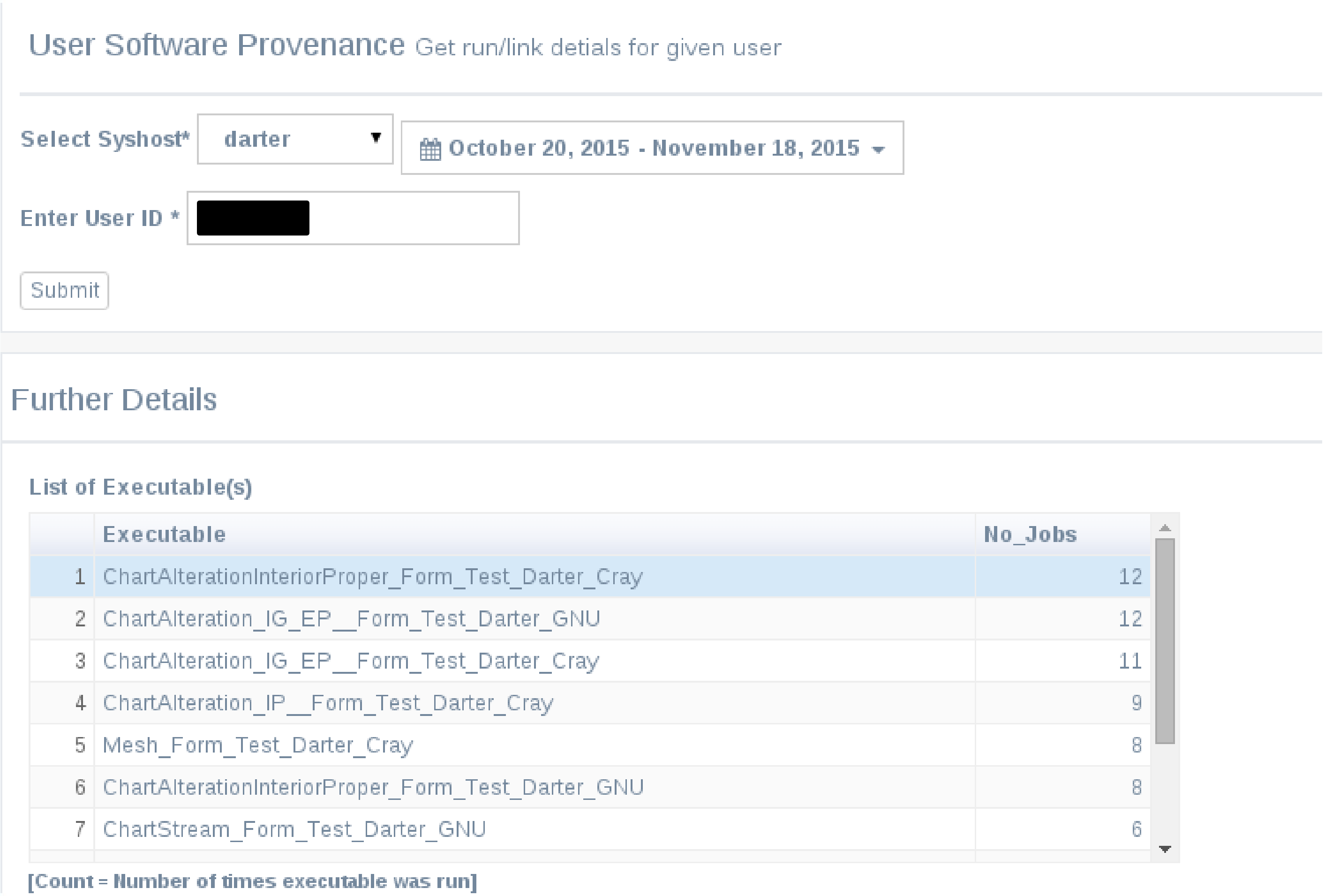
List of jobs with executable

List of object files / library linked to exec

Select a job

Environment variables for selected job

Runtime loaded object files



User Software Provenance Get run/link detials for given user

Select Syshost* darter ▾ 📅 October 20, 2015 - November 18, 2015 ▾

Enter User ID * [redacted]

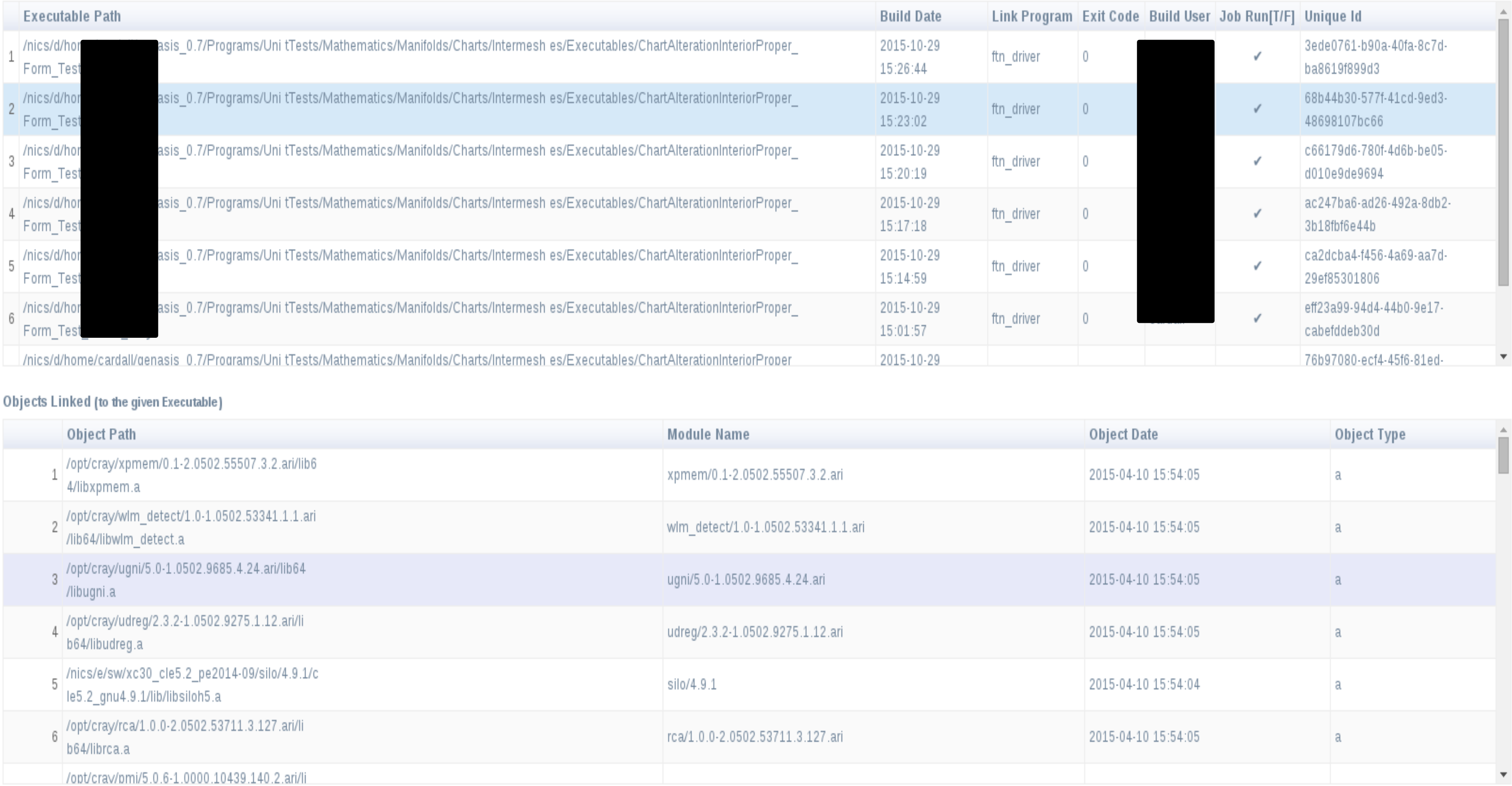
Submit

Further Details

List of Executable(s)

	Executable	No_Jobs
1	ChartAlterationInteriorProper_Form_Test_Darter_Cray	12
2	ChartAlteration_IG_EP__Form_Test_Darter_GNU	12
3	ChartAlteration_IG_EP__Form_Test_Darter_Cray	11
4	ChartAlteration_IP__Form_Test_Darter_Cray	9
5	Mesh_Form_Test_Darter_Cray	8
6	ChartAlterationInteriorProper_Form_Test_Darter_GNU	8
7	ChartStream_Form_Test_Darter_GNU	6

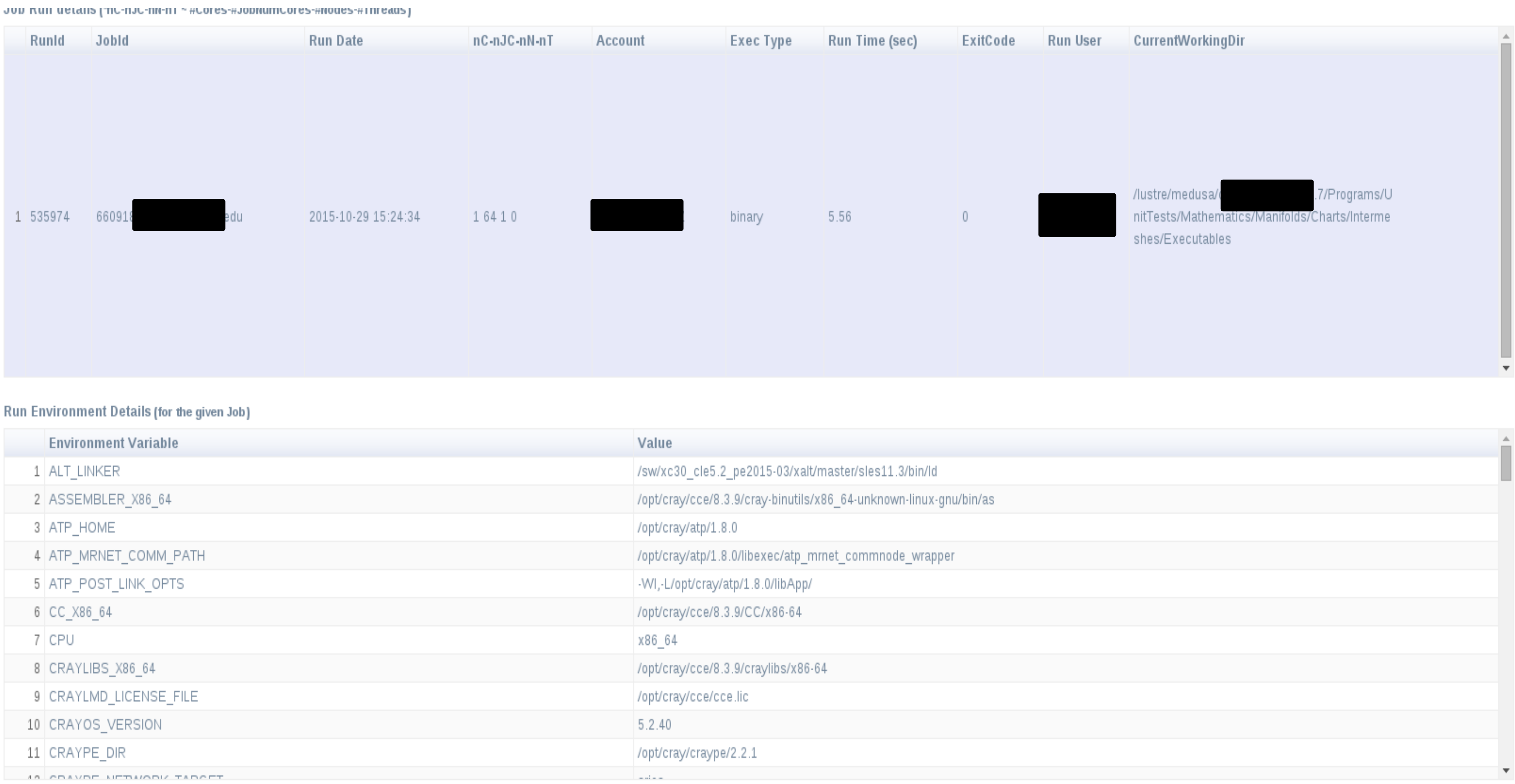
[Count = Number of times executable was run]



	Executable Path	Build Date	Link Program	Exit Code	Build User	Job Run[T/F]	Unique Id
1	/nics/d/hor[redacted]asis_0.7/Programs/Uni tTests/Mathematics/Manifolds/Charts/Intermesh es/Executables/ChartAlterationInteriorProper_ Form_Test	2015-10-29 15:26:44	ftn_driver	0	[redacted]	✓	3ede0761-b90a-40fa-8c7d-ba8619f899d3
2	/nics/d/hor[redacted]asis_0.7/Programs/Uni tTests/Mathematics/Manifolds/Charts/Intermesh es/Executables/ChartAlterationInteriorProper_ Form_Test	2015-10-29 15:23:02	ftn_driver	0	[redacted]	✓	68b44b30-577f-41cd-9ed3-48698107bc66
3	/nics/d/hor[redacted]asis_0.7/Programs/Uni tTests/Mathematics/Manifolds/Charts/Intermesh es/Executables/ChartAlterationInteriorProper_ Form_Test	2015-10-29 15:20:19	ftn_driver	0	[redacted]	✓	c66179d6-780f-4d6b-be05-d010e9de9694
4	/nics/d/hor[redacted]asis_0.7/Programs/Uni tTests/Mathematics/Manifolds/Charts/Intermesh es/Executables/ChartAlterationInteriorProper_ Form_Test	2015-10-29 15:17:18	ftn_driver	0	[redacted]	✓	ac247ba6-ad26-492a-8db2-3b18fbf6e44b
5	/nics/d/hor[redacted]asis_0.7/Programs/Uni tTests/Mathematics/Manifolds/Charts/Intermesh es/Executables/ChartAlterationInteriorProper_ Form_Test	2015-10-29 15:14:59	ftn_driver	0	[redacted]	✓	ca2dcba4-f456-4a69-aa7d-29ef85301806
6	/nics/d/hor[redacted]asis_0.7/Programs/Uni tTests/Mathematics/Manifolds/Charts/Intermesh es/Executables/ChartAlterationInteriorProper_ Form_Test	2015-10-29 15:01:57	ftn_driver	0	[redacted]	✓	eff23a99-94d4-44b0-9e17-cabefddeb30d
	/nics/d/home/cardall/genasis_0.7/Programs/Uni tTests/Mathematics/Manifolds/Charts/Intermesh es/Executables/ChartAlterationInteriorProper	2015-10-29					76b97080-ecf4-45f6-81ed-

Objects Linked (to the given Executable)

	Object Path	Module Name	Object Date	Object Type
1	/opt/cray/xpmem/0.1-2.0502.55507.3.2.ari/lib64/libxpmem.a	xpmem/0.1-2.0502.55507.3.2.ari	2015-04-10 15:54:05	a
2	/opt/cray/wlm_detect/1.0-1.0502.53341.1.1.ari /lib64/libwlm_detect.a	wlm_detect/1.0-1.0502.53341.1.1.ari	2015-04-10 15:54:05	a
3	/opt/cray/ugni/5.0-1.0502.9685.4.24.ari/lib64 /libugni.a	ugni/5.0-1.0502.9685.4.24.ari	2015-04-10 15:54:05	a
4	/opt/cray/udreg/2.3.2-1.0502.9275.1.12.ari/lib64/libudreg.a	udreg/2.3.2-1.0502.9275.1.12.ari	2015-04-10 15:54:05	a
5	/nics/e/sw/xc30_cle5.2_pe2014-09/silo/4.9.1/cle5.2_gnu4.9.1/lib/libsiloh5.a	silo/4.9.1	2015-04-10 15:54:04	a
6	/opt/cray/rca/1.0.0-2.0502.53711.3.127.ari/lib64/librca.a	rca/1.0.0-2.0502.53711.3.127.ari	2015-04-10 15:54:05	a
	/opt/cray/pmi/5.0.6-1.0000.10439.140.2.ari/li			



Job Run details (nC-nJC-nN-nT ~ #Cores-#JobNumCores-#Nodes-#Threads)

	RunId	JobId	Run Date	nC-nJC-nN-nT	Account	Exec Type	Run Time (sec)	ExitCode	Run User	CurrentWorkingDir
1	535974	66091[redacted]edu	2015-10-29 15:24:34	1 64 1 0	[redacted]	binary	5.56	0	[redacted]	/lustre/medusa/[redacted].7/Programs/UnitTests/Mathematics/Manifolds/Charts/Intermeshes/Executables

Run Environment Details (for the given Job)

	Environment Variable	Value
1	ALT_LINKER	/sw/xc30_cle5.2_pe2015-03/xalt/master/sles11.3/bin/ld
2	ASSEMBLER_X86_64	/opt/cray/cce/8.3.9/cray-binutils/x86_64-unknown-linux-gnu/bin/as
3	ATP_HOME	/opt/cray/atp/1.8.0
4	ATP_MRNET_COMM_PATH	/opt/cray/atp/1.8.0/libexec/atp_mrnet_commnode_wrapper
5	ATP_POST_LINK_OPTS	-Wl,-L/opt/cray/atp/1.8.0/libApp/
6	CC_X86_64	/opt/cray/cce/8.3.9/CC/x86-64
7	CPU	x86_64
8	CRAYLIBS_X86_64	/opt/cray/cce/8.3.9/craylibs/x86-64
9	CRAYLMD_LICENSE_FILE	/opt/cray/cce/cce.lic
10	CRAYOS_VERSION	5.2.40
11	CRAYPE_DIR	/opt/cray/craype/2.2.1
12	CRAYPE_NETWORK_TARGET	aries

Conclusions

- XALT has been in production for over a year
- XALT has been successfully deployed on multiple HPC centers to support their operations
- XALT helps stakeholders make data-driven decision on software support
- Further analysis on XALT data may yield more understanding of interesting users' behavior
- Source: **https://github.com/Fahey-McLay/xalt**

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.”
- Thanks to the XALT community for feedback and bug reports