Los Alamos
NATIONAL LABORATORY

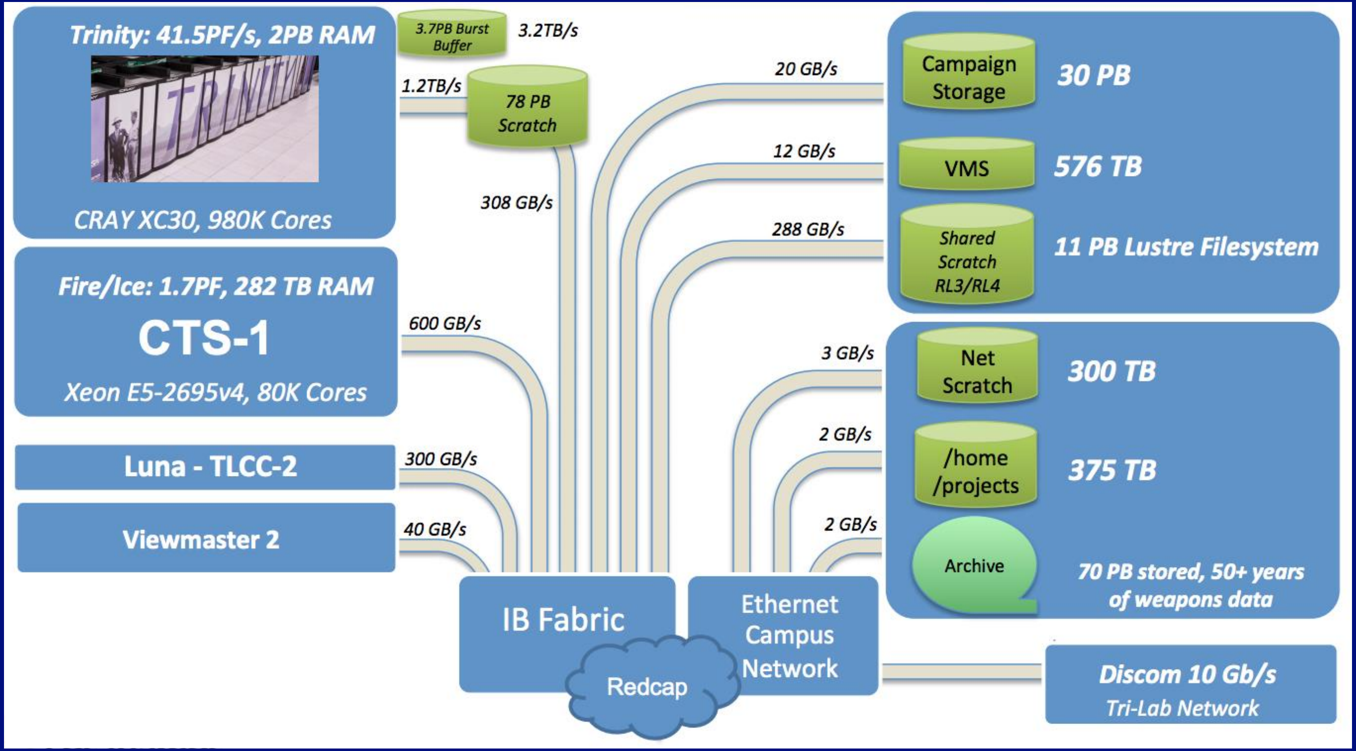
Grand Unified File Index (GUFI)

Jason Lee
HPC-DES Storage Team

July 21, 2022

LA-UR-22-26715

LANL Compute/Storage Environment (Secure) Circa early 2017



Sampling of LANL Filesystems Circa 2020/2021 (Turquoise)

Filesystem	Directory Count (Millions)	File Count (Millions)
Home	3.8	36.6
Projects	10.7	114.2
Scratch 1	1.1	237.2
Scratch 2	5.1	857.7
Campaign	0.4	13.5
Archive 1	1.1	49.9

Filesystem Usage

- Users searching for data in files
  - Do not always know where files are
  - Lots of filesystems with lots of files
  - Files within a directory might be organized poorly
  - Want fast results (or will terminate search)

- Admins need to manage filesystem
  - Find users taking up the most space
  - Find stale files that can be deleted
  - Want reasonably fast results

No Unified Set of Performant Tools

- Different admin tools for different filesystems
  - Admins only

- Standard command line tools
  - Slow
    - Single threaded
  - Unwieldy
    - Must chain multiple commands together to get results
  - Uses resources of mission critical jobs

Grand Unified File Index

- Highly parallel for fast index traversal
- Stores metadata
    - Complex queries with SQL
    - Support for extended attributes
- Enforces permissions so users and admins can use the same index
- Single index for all filesystems
- Leverages well developed technologies
    - POSIX filesystem hierarchy, permissions, attributes
    - SQLite 3, PCRE, jemalloc, CMake 3
    - Flash Storage

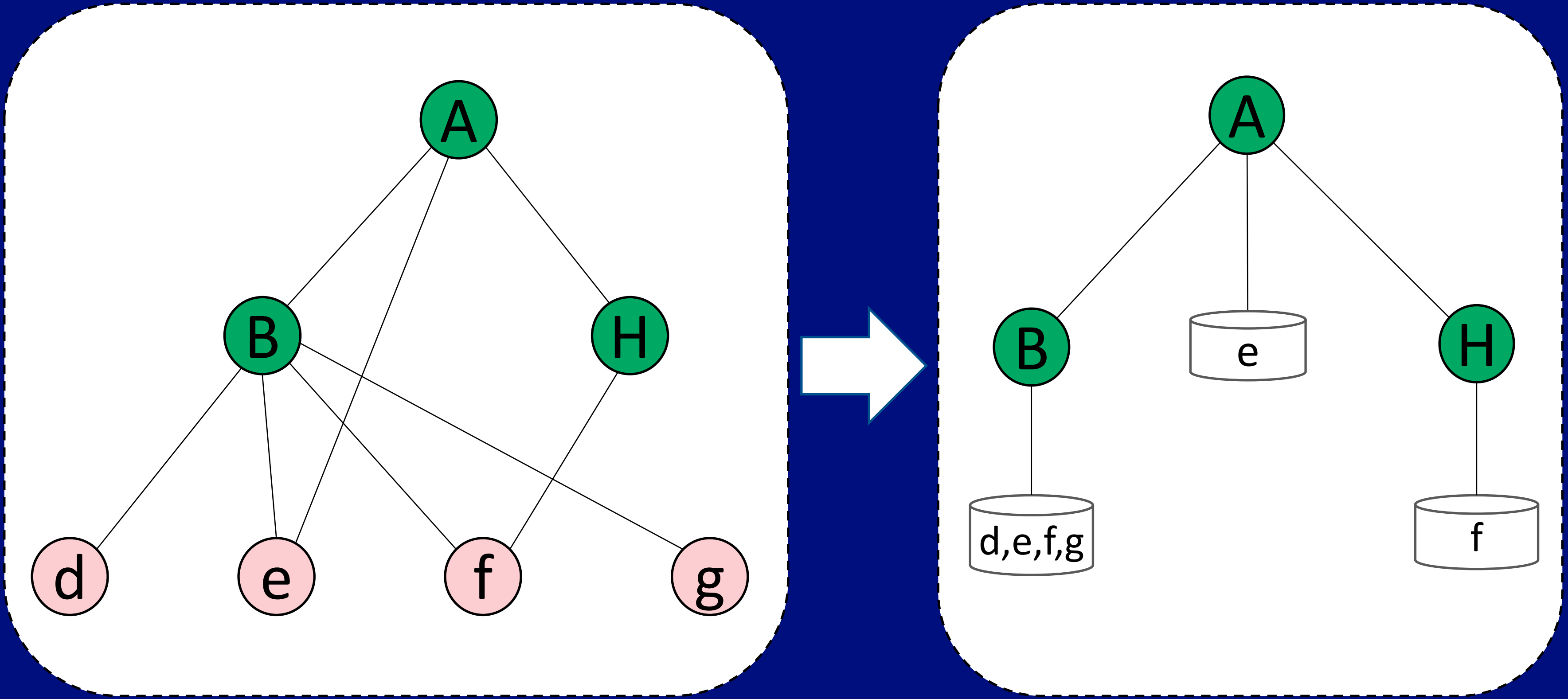
2018 R&D 100 WINNER

GUFI Grand Unified File Index

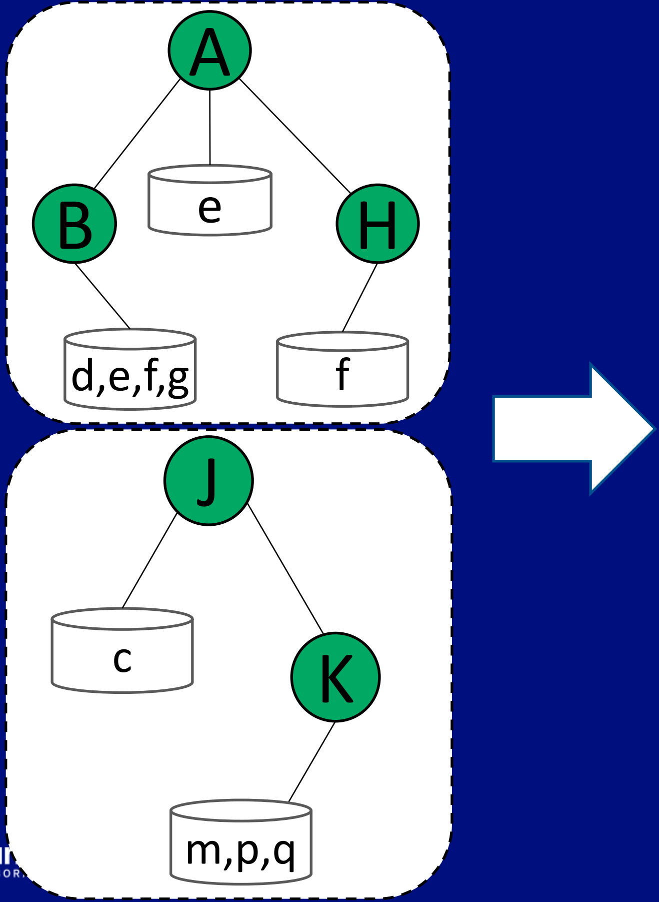
Fastest open-source software for supercomputer user-queried metadata

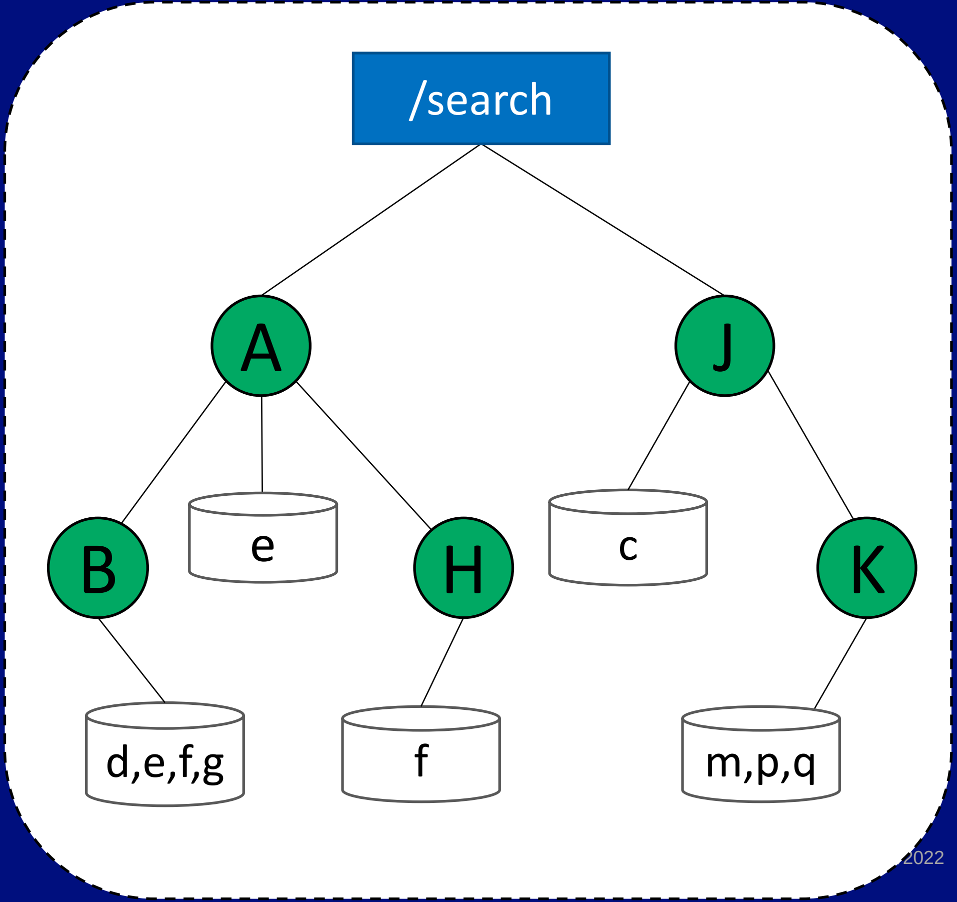
- Provides metadata queries of ultrascale file-system trees in seconds
- Maintains security structure while facilitating custom user metadata queries
- Economizes supercomputing resources — enhancing the role of the user
- Offers open-source software of a mere few thousand lines that is concise and extensible

Source Filesystem to Index



Combining Indices





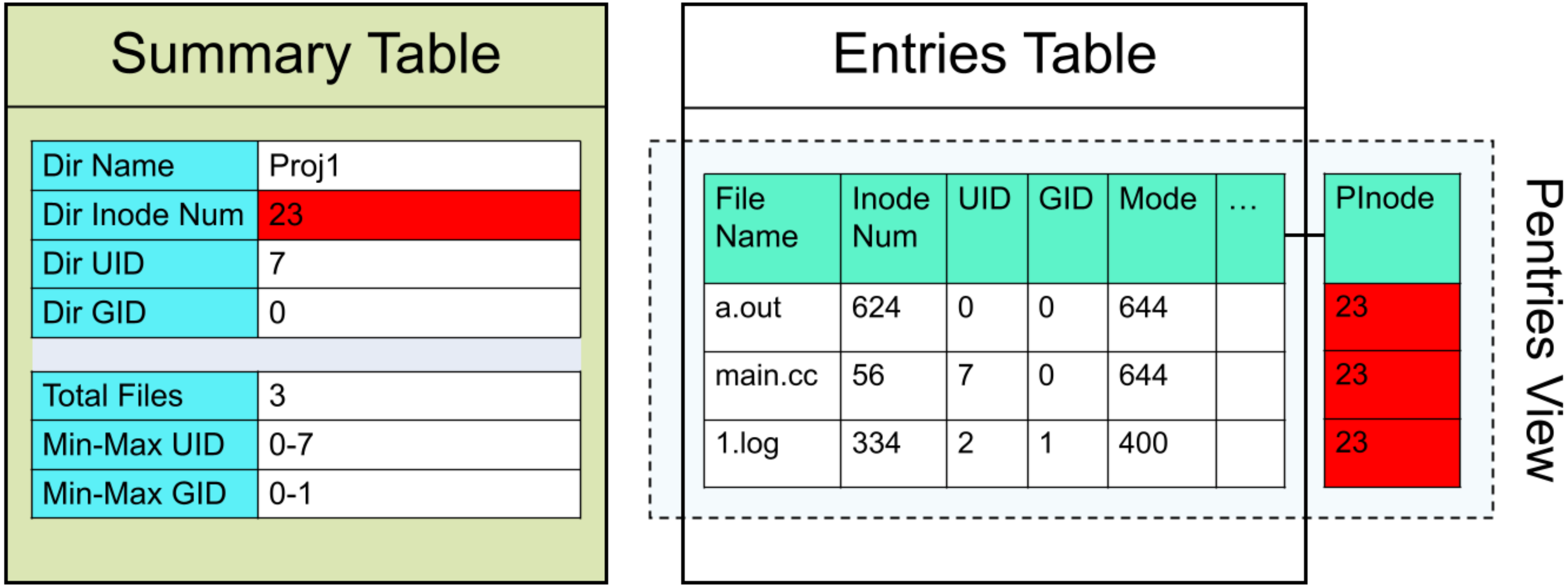
Why not a flat index?

- Very performant for simple queries
  - No tree traversal
  - One/few database(s) to open

- Multiple uids/gids in one database
  - Custom per row permission checking or admin only

- Must scan all entries when querying
  - Constant time queries
    - Queries do not scale based on caller
  - Scan multiple times when joining
  - Lots of I/O

Database Table Schema



Summary Table

Dir Name	Proj1
Dir Inode Num	23
Dir UID	7
Dir GID	0
Total Files	3
Min-Max UID	0-7
Min-Max GID	0-1

Entries Table

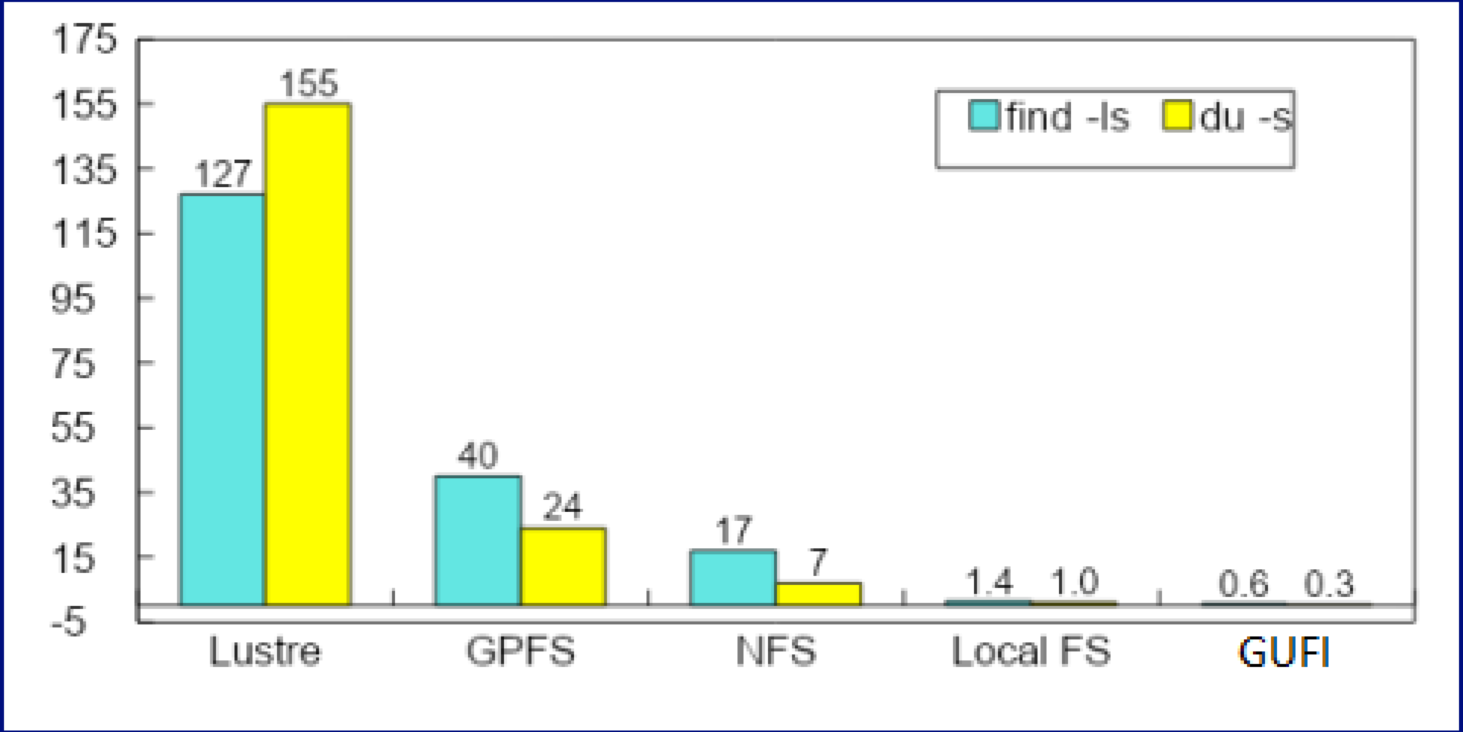
File Name	Inode Num	UID	GID	Mode	…	PInode
a.out	624	0	0	644		23
main.cc	56	7	0	644		23
1.log	334	2	1	400		23

Basic Querying

- gufi_query
  - Runs SQL statements
    - Need to know database and table schemas
    - Meant for advanced users/admin
      - User facing tools wrapping gufi_query
  - Highly parallel
    - Each directory is processed by a thread

- Get results from directories in parallel
  - `SELECT name, size FROM entries;`

Querying Linux Kernel 5.8.9 Source (74K dirs + files)



Advanced Querying

- Use the summary table to determine whether to run query on entries table
  - Quickly find out if the current directory contains an entry with value X
- Use the tree summary table
  - Summary of entire subtree starting at current directory
  - Determine whether a subtree should be traversed
    - Quickly find out if a subtree contains an entry with value X
    - Quickly get a value without walking the subtree
  - Not generated by default

Where X can be

- Subdir count
- File count
- Link count
- Min/max
  - size
  - uid
  - gid
  - …
- User defined values
  - Minor schema/code changes

Aggregate Results

Goal	Shell	GUFI

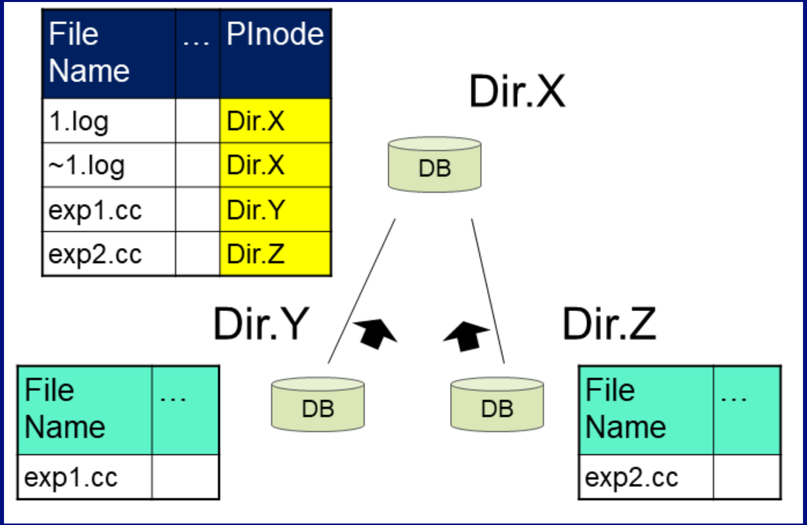
Top 10 Largest Files	`find -printf "%P %s\n" | sort -n -r -k 2 | head -n 10`	`INSERT INTO aggregate SELECT name, size FROM entries;`
		
		`SELECT name, size FROM aggregate ORDER BY size DESC LIMIT 10;`

Top 10 Largest Files by UID	`find -printf "%P %s %U\n" |` **`???`**
	
	Associative arrays?
	awk/perl?
	uniq + grep + sort?
		`INSERT INTO aggregate SELECT name, size, uid FROM entries;`
		
		`SELECT name, size FROM aggregate GROUP BY uid ORDER BY size DESC LIMIT 10;`

Los Alamos NATIONAL LABORATORY

Rollup

- Most subdirectories under a directory have compatible uid, gid, and read permissions
  - Why traverse the subtree and open multiple directories/databases if one will suffice?
- Copy data from subdirectories upwards if uid, gid, and permissions allow for it
  - Skip traversing entire subtrees and still get the same data
- Copy data up one level at a time
  - Lots of duplicate data and used space
  - Allows for querying to start at any level and still take advantage of rollup
- Don't always roll up fully
  - Large directories can cause large tail latency

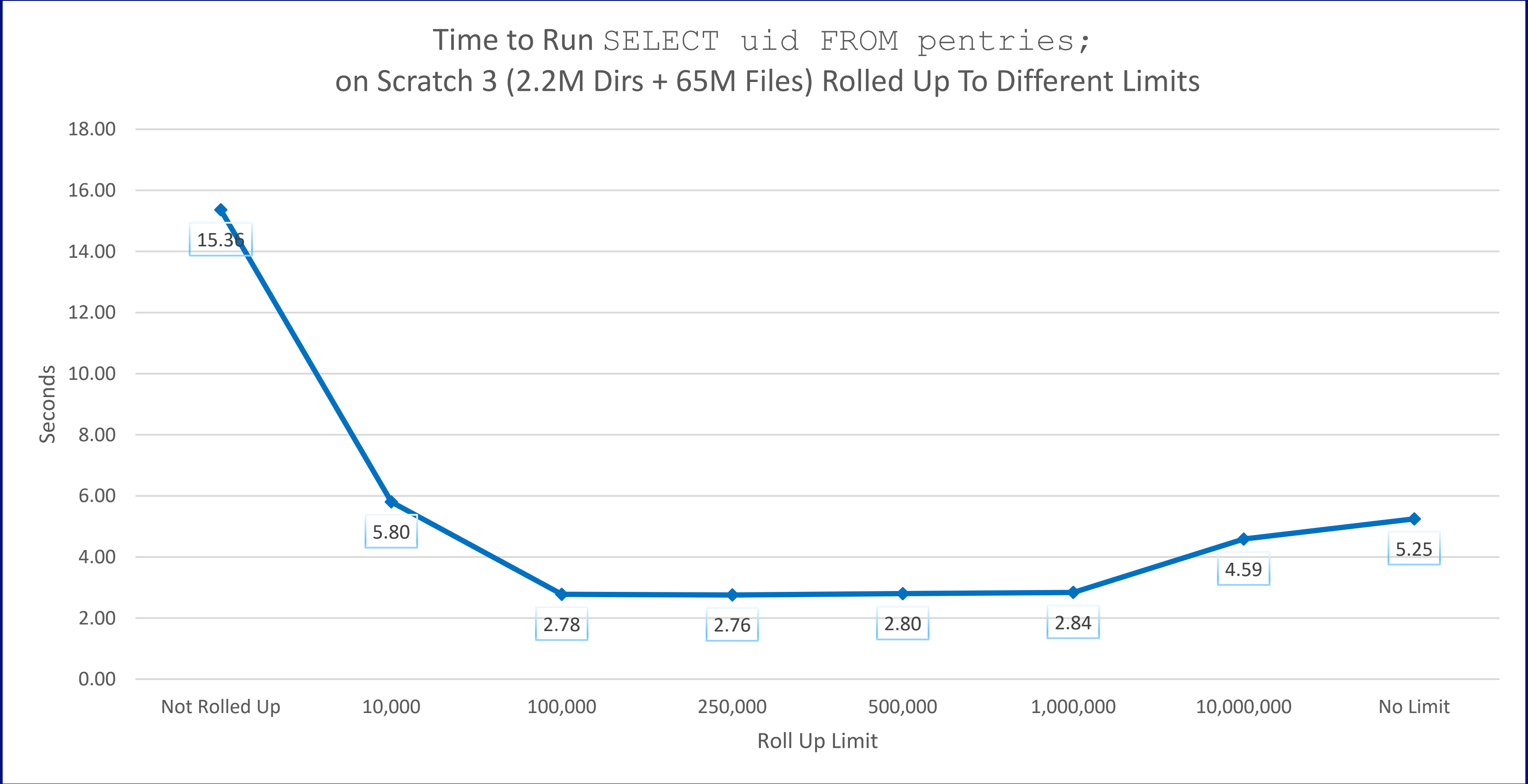


File Name	…	PInode
1.log		Dir.X
~1.log		Dir.X
exp1.cc		Dir.Y
exp2.cc		Dir.Z

File Name	…
exp1.cc	

File Name	…
exp2.cc	

Index	Original Directory Count	# of Directories to Traverse	% of Directories to Traverse
anony	7.35M	2873	0.04%
yelluser	1.62M	6406	0.39%
scratch 3	2.20M	5049	0.23%


Time to Run `SELECT uid FROM pentries;`
on Scratch 3 (2.2M Dirs + 65M Files) Rolled Up To Different Limits
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Seconds
15.36
5.80
2.78
2.76
2.80
2.84
4.59
5.25
Not Rolled Up
10,000
100,000
250,000
500,000
1,000,000
10,000,000
No Limit
Roll Up Limit

Extended Attributes (xattrs)

- Small user data stored with metadata
  - Tag files

- Different permission handling than stat(2) data
  - Need read permission of files instead of the directory they are in
  - Compatible xattrs are stored in the main database
  - Incompatible xattrs are stored in per-uid and per-gid databases are attached during querying
    - Successful attach indicates that the user can read the xattr values

- Includes rolling up xattrs

More Information

- Source Code
  - https://github.com/mar-file-system/GUFI

- Anonymized Traces From LANL Systems
  - https://github.com/mar-file-system/GUFI-Filesystem-Traces

- Supercomputing 2022 Paper

Thank you!

Sampling of LANL Filesystems Circa 2020/2021 (Yellow)

Filesystem	Directory Count (Millions)	File Count (Millions)
Home	3.3	23.4
Projects	18.5	178.9
Scratch 3	5.9	165.1
Scratch 4	16.5	225.0
Scratch 5	7.4	159.3
Archive 2	5.6	161.3

Indexing a filesystem

- Directly
  - gufi_dir2index

- With traces
  - gufi_dir2trace
  - gufi_trace2index

- Filesystem specific tools
  - Lustre
  - GPFS
  - HPSS
  - NFS

Indices do not have to reside near the source filesystem

Rollup Rules

1. World read and exec (i.e. o+rx)

2. Matching perms (usr, grp, and other), with same usr and grp

3. Matching usr and grp perms, read and exec (ug+rx) with same usr and grp, and not world read and exec (i.e. o-rx)

4. Matching usr perms, read and exec(u+rx) with the same usr, and not grp or world read and exec (go-rx)

xattr Rules

1. File is 0+R (doesn’t matter what the parent dir perms or ownership is)

2. File is UG+R doesn’t matter on other, with file and parent same usr and grp and parent has only UG+R with no other read

3. File is U+R doesn’t matter on grp and other, with file and parent same usr and parent dir has only U+R, no grp and other read

4. Directory has write for every read: drw*rw_*rw* or drw*rw*___ or drw*______
   - if you can write the dir you can   chmod the files to see the xattrs

Deployment

- Indices are not up to date
  - Scans take time to complete
  - Live filesystems are always churning (unless indexing snapshots)
  - Scan filesystems every so often
  - LANL runs every 4 hours

- Symlink to index root
  - During update, change the symlink to point to the latest index
    - Active queries will still complete
    - New queries will use new index

User Facing Tools

- gufi_find
  - find(1)

- gufi_ls
  - ls(1)

- gufi_stat
  - stat(1)

- gufi_stats
  - Common queries that are probably useful