SDC 19

September 23-26, 2019
Santa Clara, CA

Enabling Rapid and Secure Metadata Search Across Storage Tiers with GUFI

Dominic Manno

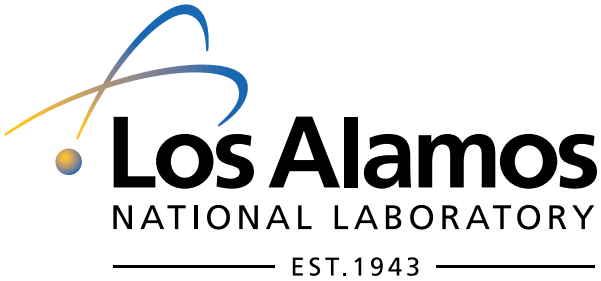
Los Alamos National Laboratory

LA-UR-19-29542

Acknowledgments

- Some slides and content/diagrams provided by LANL colleagues:
  - David Bonnie, Gary Grider, Jason Lee, Brad Settlemyer

Overview



- HPC at LANL
- GUFI Description
- Deployment Details
- Road to Performance
- Next Steps

LANL HPC Environment

HPC at LANL

- Eight decades of weapons computing support to keep the nation safe
  - Simulation to determine stability, defects, etc.
- Cutting edge technology enables large, long-running, multi physics 3D simulations
  - Jobs can last months running on 80% of the machine

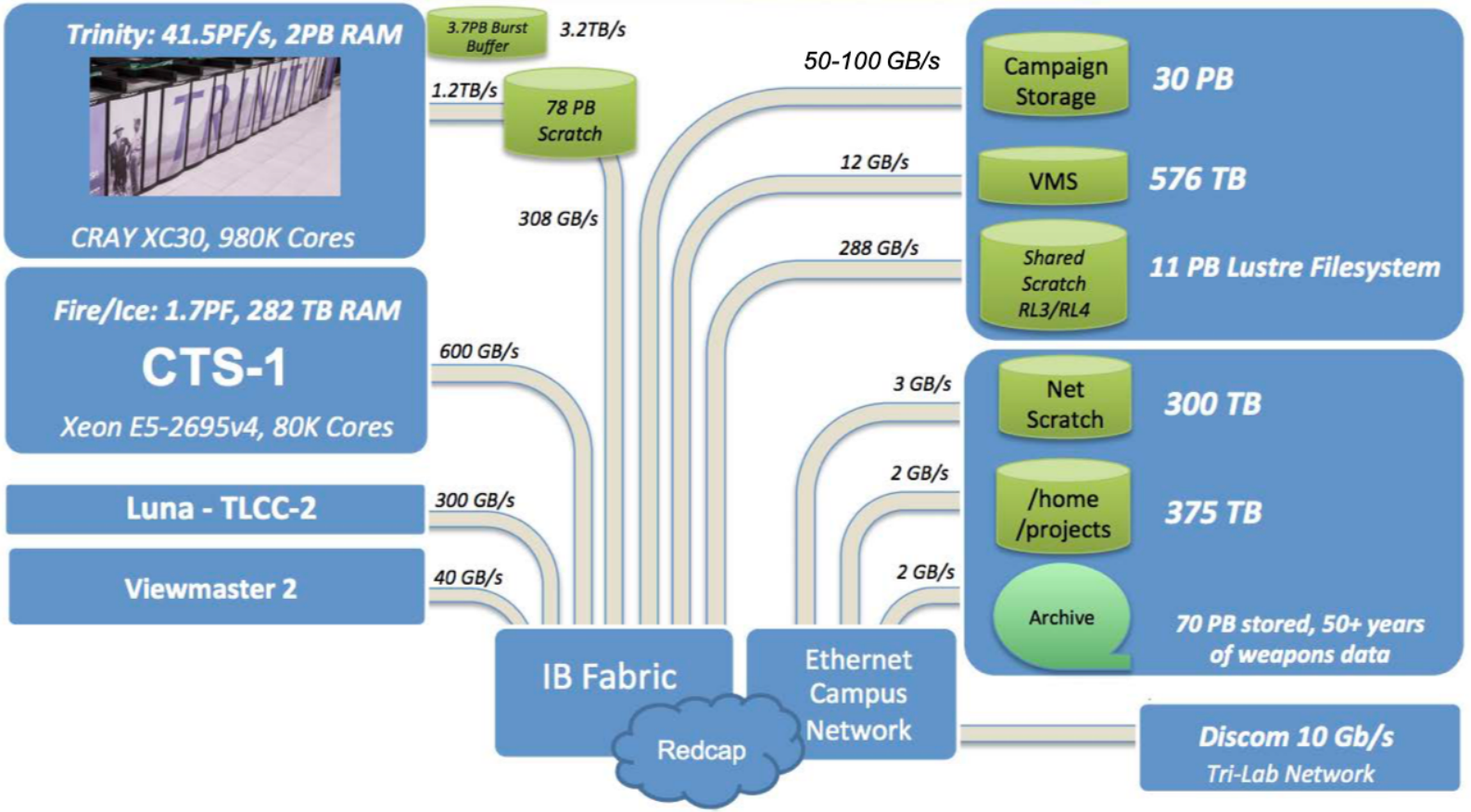
“Better” Science calls for Better Computers

Roadrunner (2007)
1st Petaflop/Accelerator Platform

Cielo (2011)
1.7 Petaflop Platform

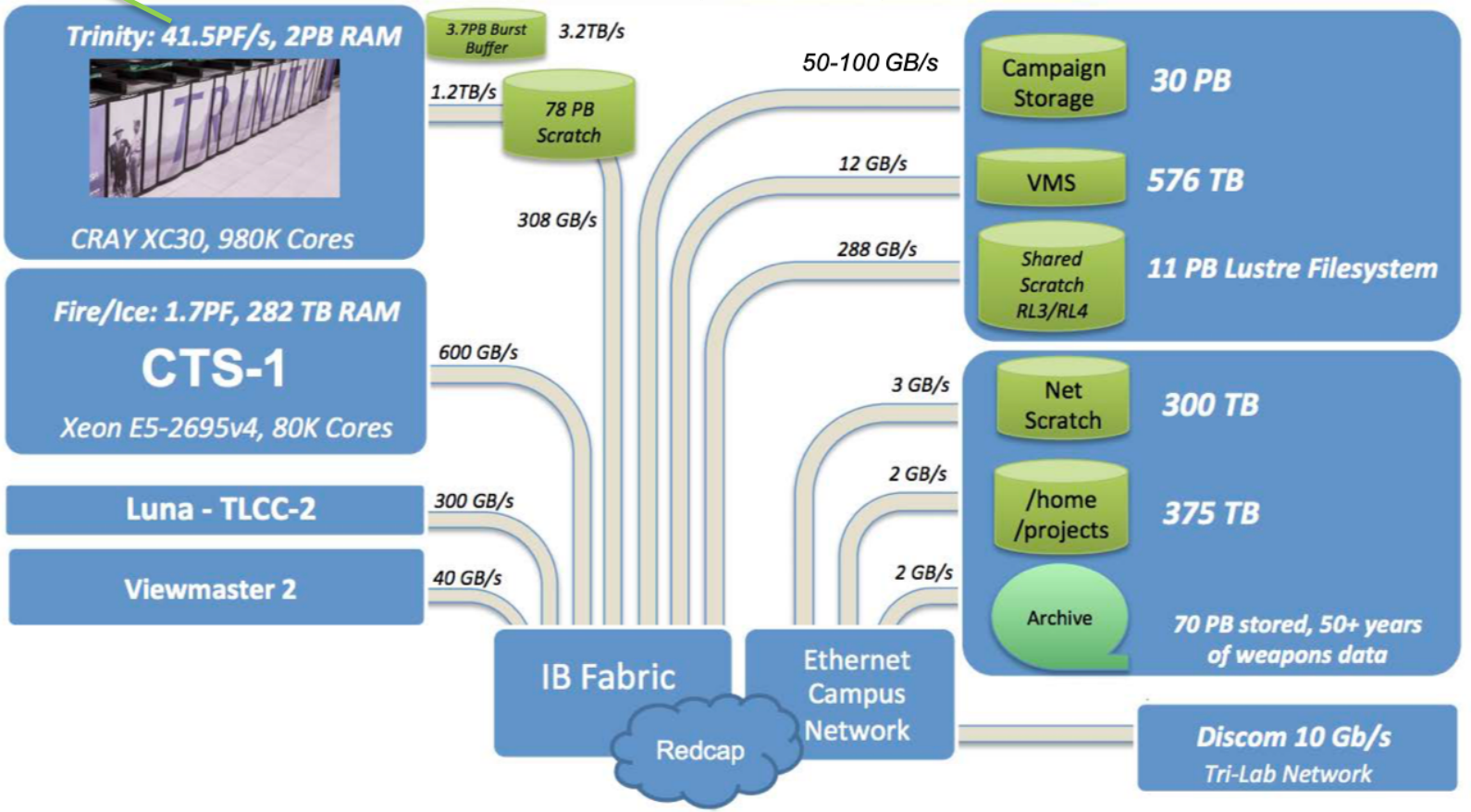
Trinity (2015)
~20 Petaflops, 4 PB Burst Buffer

Storage Tiers

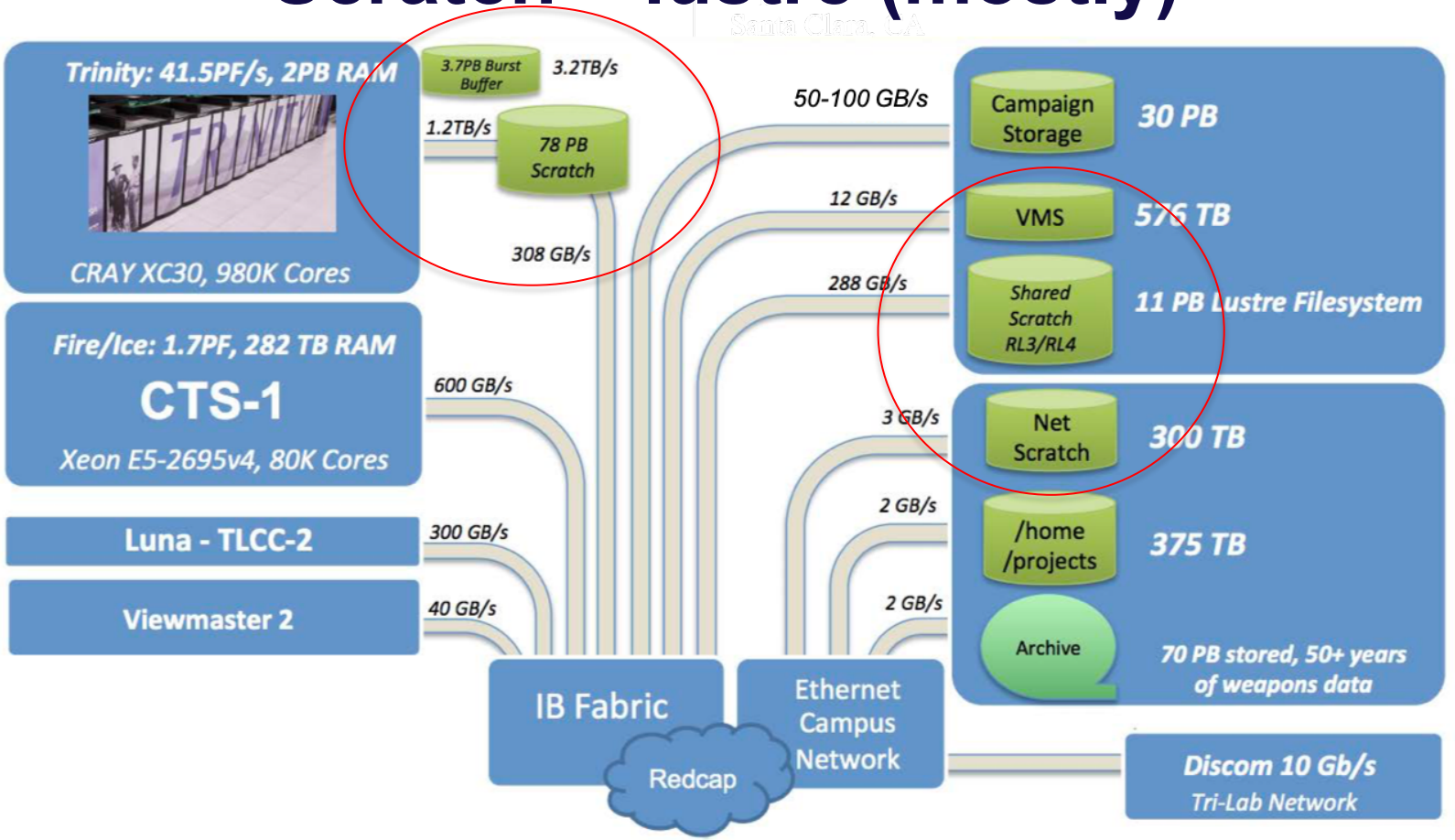


Storage Tiers

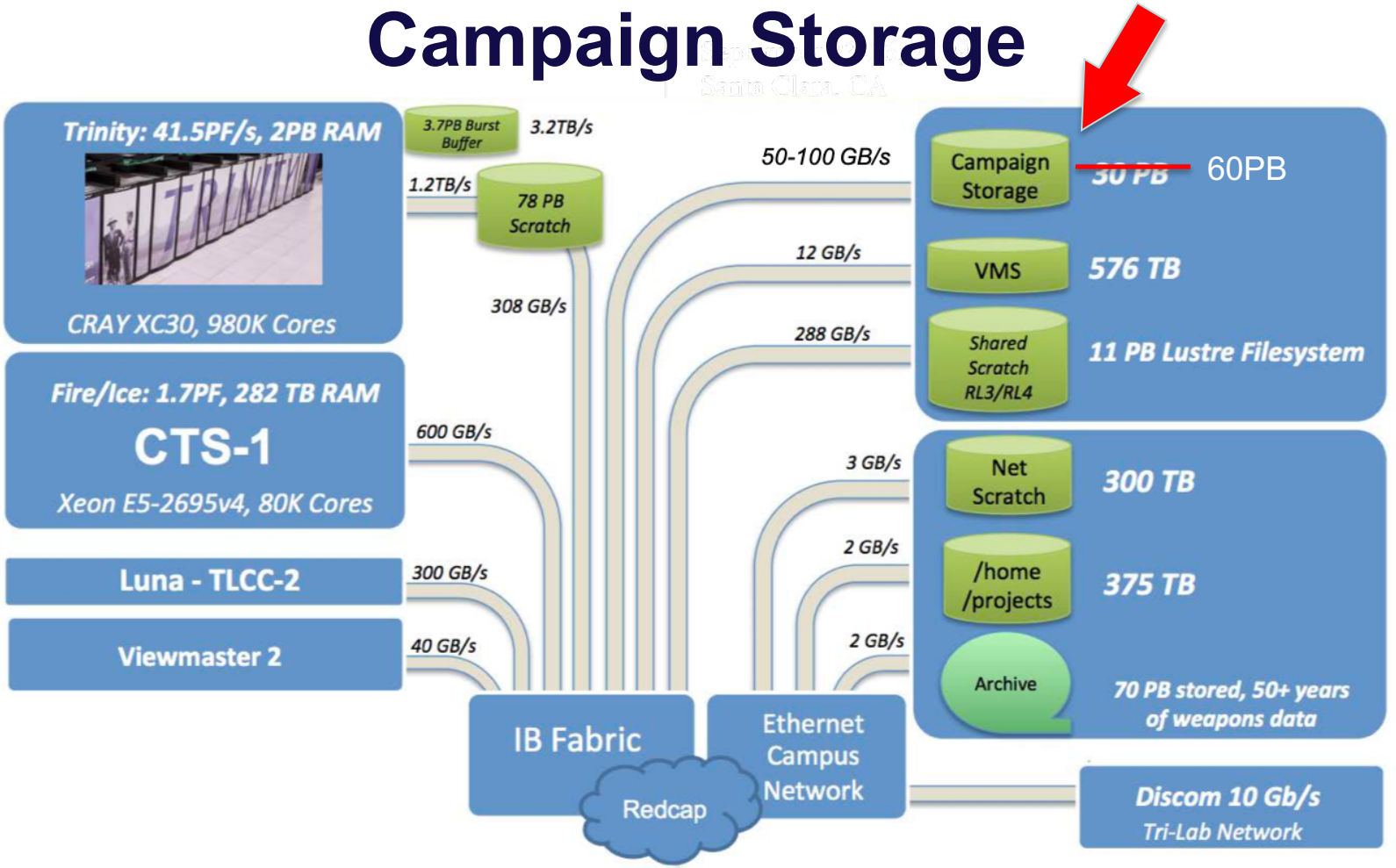
20.158 PF/s measured



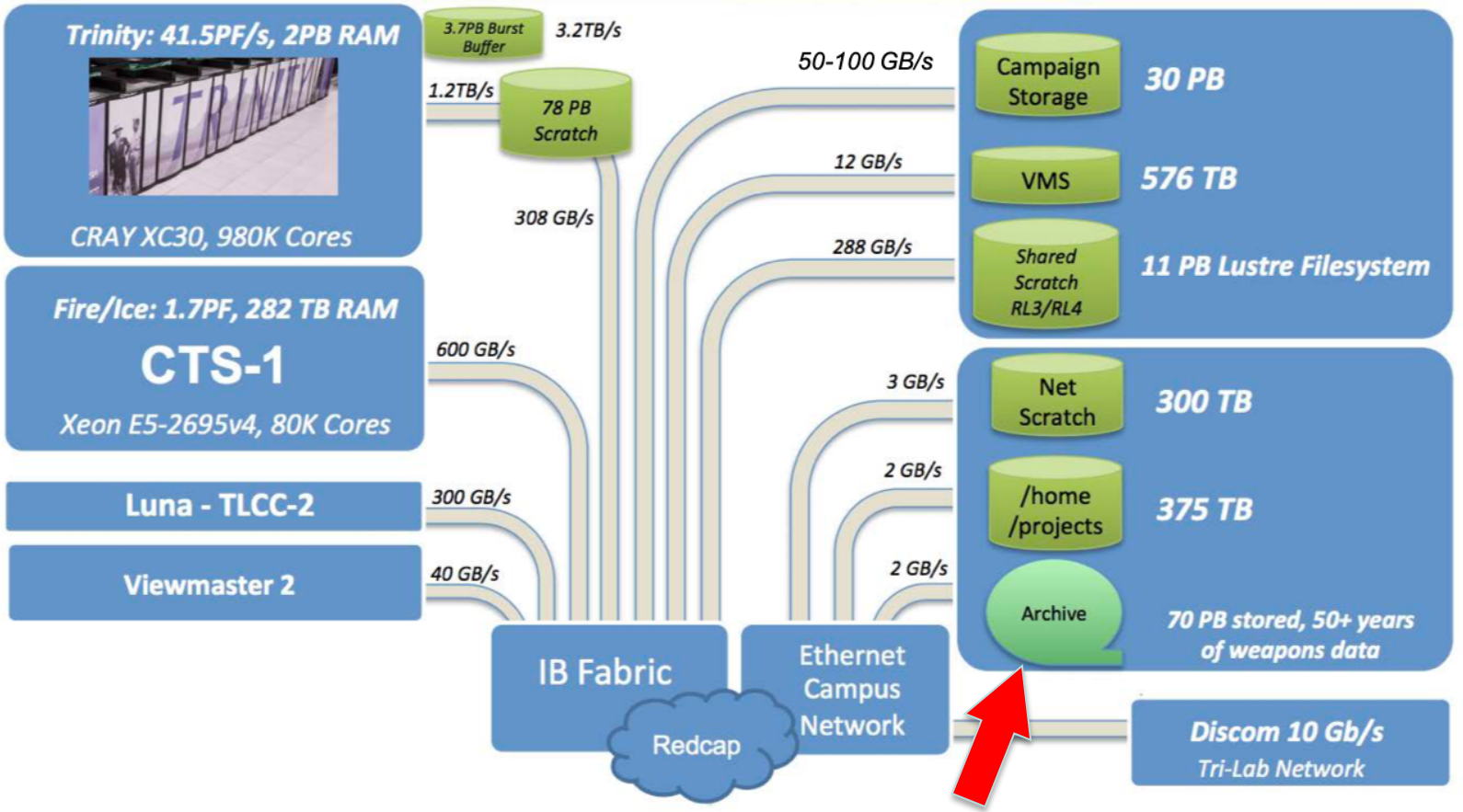
Scratch – lustre (mostly)



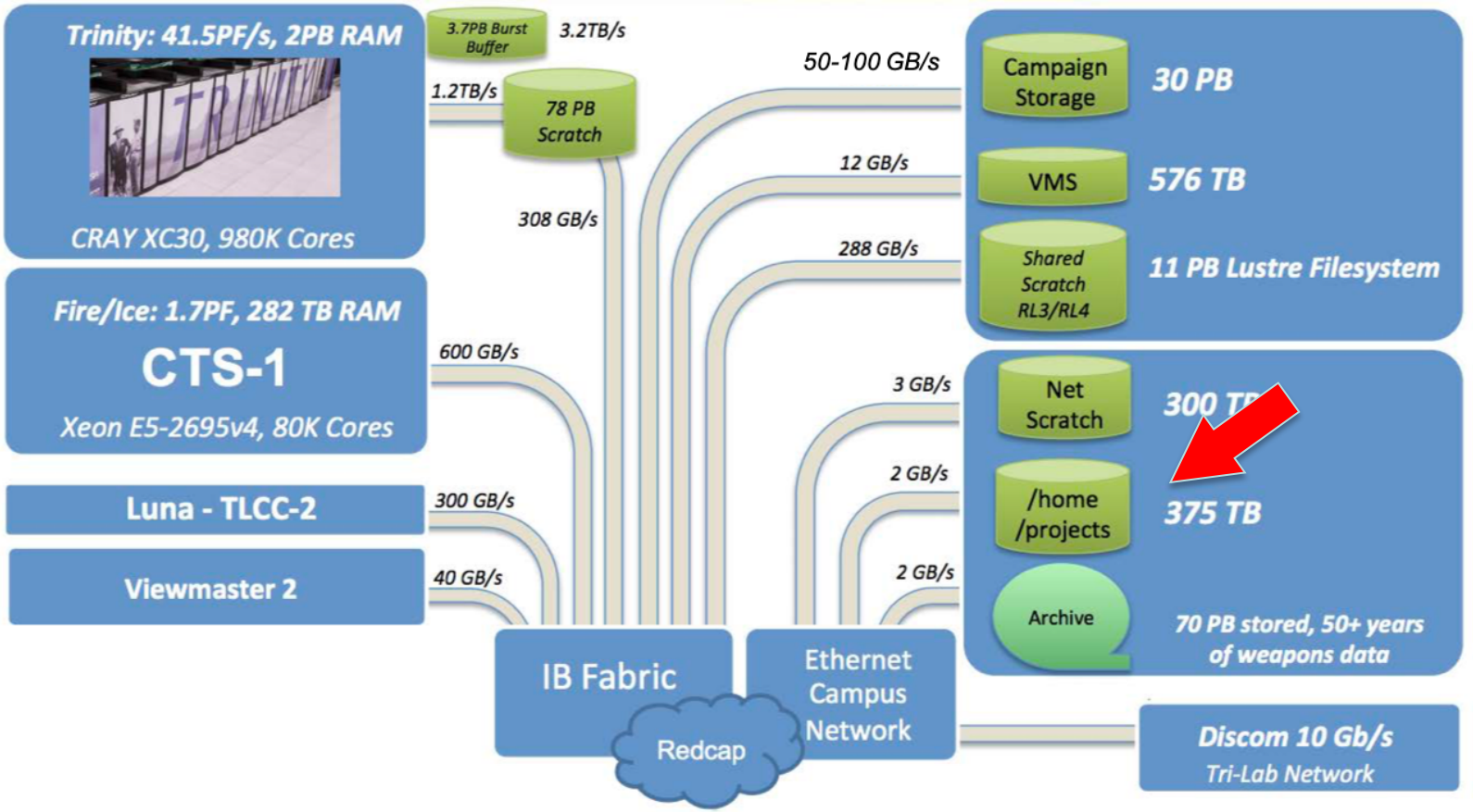
Campaign Storage



Archive



Home and Project Space



Metadata Problem

- This model depends on users knowing about their data
  - Where did it get written?
  - Does it need to be backed up? If so, did I already save a copy?
  - Good naming and hierarchy
- Without explicit management the archive would collect far too much data
- Need to provide better tools

GUFI Overview

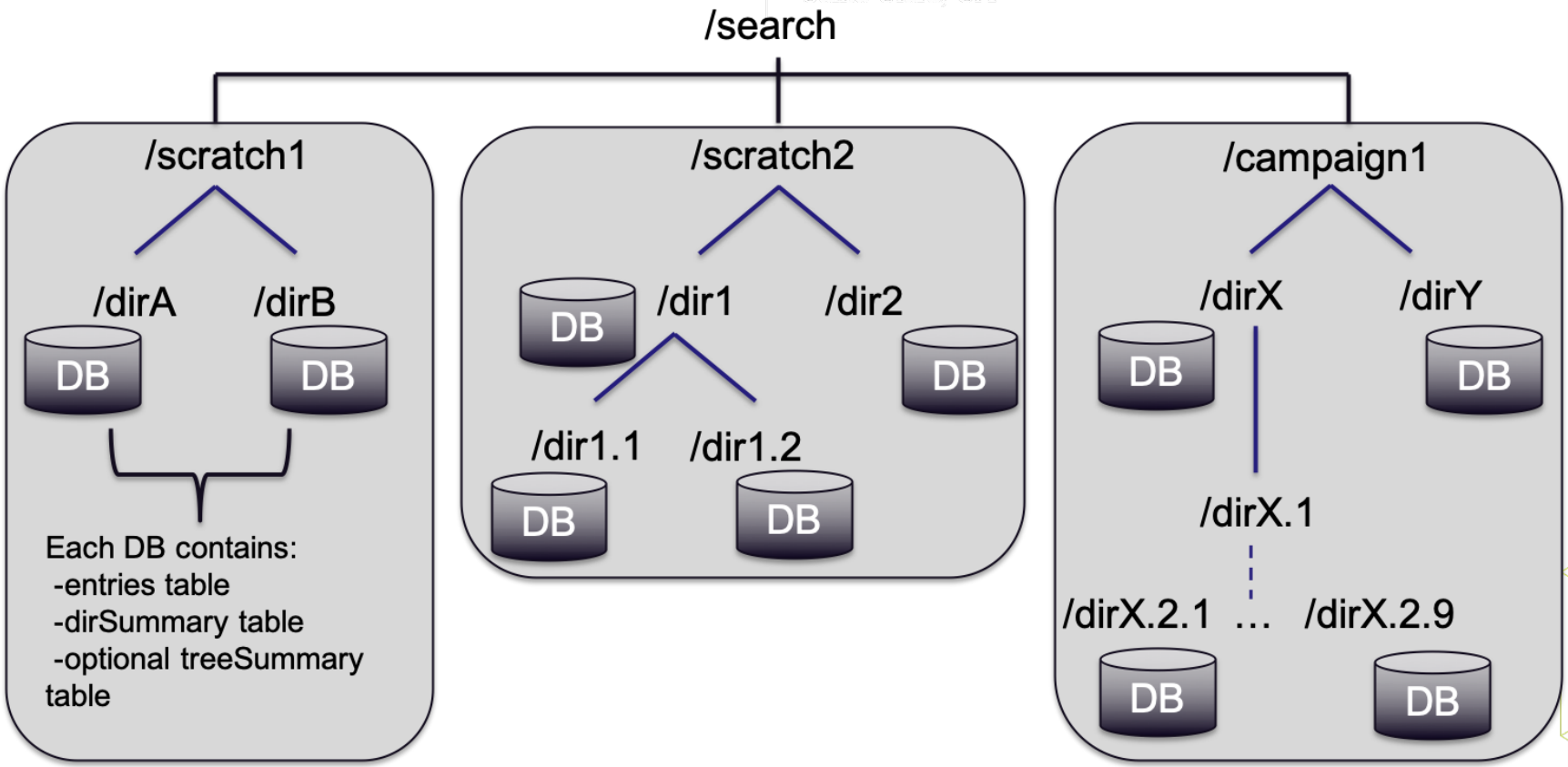
Early Design Discussions

- Index that could cover all storage systems
- Share index – admins and users, securely!
- Reasonable update times, minimizing impact to source file systems
- Parallel
- Xattrs
- Leverage existing tech

GUFI Design

- Re-create source FS tree
  - Maintain ownership and permissions on the newly created tree
  - Secure – we already depend on these permissions on the source
- Use embedded DB in every dir
  - sqlite
  - This is where all file information goes
- Threads!

GUFI Design



GUFI Design

/search

gufi — -bash — 123×35

```
[bws@pn1809254:gufi:549 [master]]$ sloccount src/
Creating filelist for src
Categorizing files.
Finding a working MD5 command....
Can't exec "md5sum": No such file or directory at /opt/local/bin/break_filelist line 688, <CODE_FILE> line 15.
Found a working MD5 command.
Computing results.


SLOC    Directory       SLOC-by-Language (Sorted)
6541    src             ansic=5739,cpp=802


Totals grouped by language (dominant language first):
ansic:         5739 (87.74%)
cpp:            802 (12.26%)
```

-entries table
-dirSummary table
-optional treeSummary table

/dirX.2.1 … /dirX.2.9

DB DB

Alternative Approaches

- Flatten the namespace
  - Rename on high in the tree is costly
  - Implementing security for users and admins to share is hard and likely a performance hit
- Why not just write MPI or MPI libcircle jobs to do this?
  - Resources
  - Users like find | grep and ls --with-color
- Storing index in source trees

Development

GUFI_* tools

- Very powerful queries enabled with base gufi code
- Simple user interface required for those “easier” more frequent queries
- Try to emulate popular MD tools: ls, find, stat, du, etc
  - Don’t require ton of new code, just basic wrappers

Ingest Tools

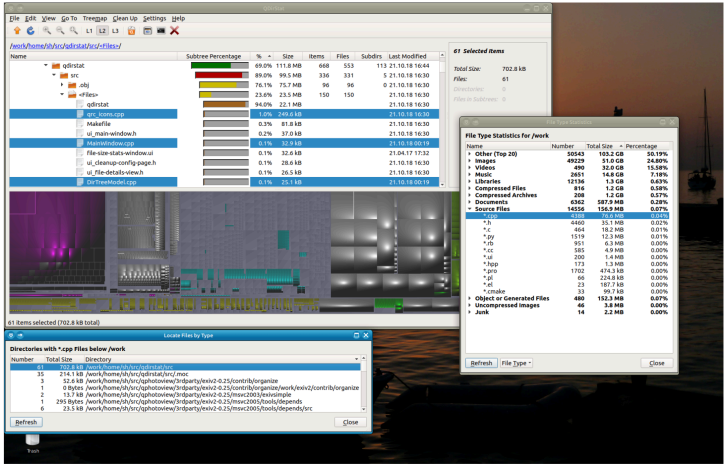
- POSIX tree-walk ingest done
- FS specific optimizations exist – use when applicable
- Incremental update capability could provide speedup

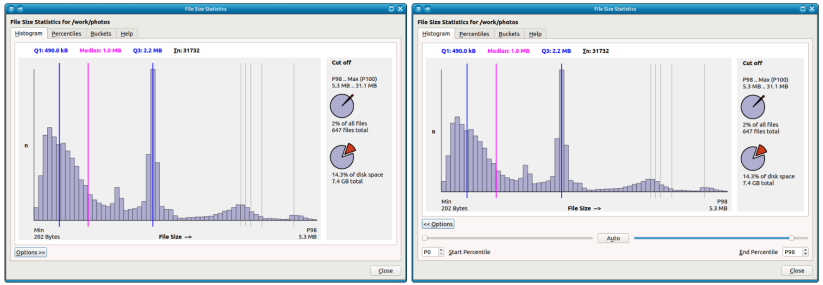
Hardening

- Bug fixes
- Stable releases and build env
- Travis – auto builds for RedHat, SUSE, macOS

Reports and Web Interface

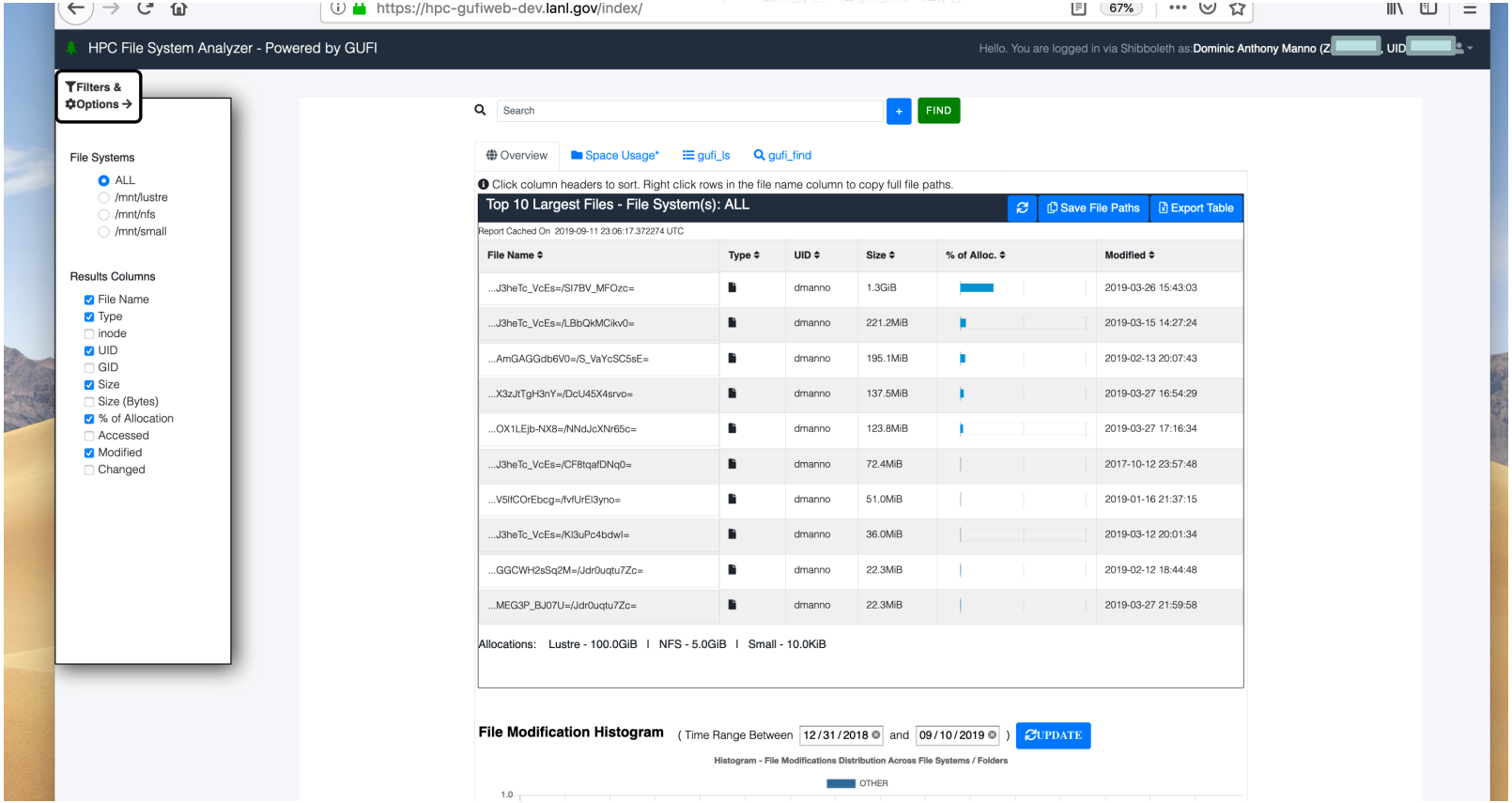
- Represent commonly used queries as reports
- Provide ability to find “hot-spots” in tree
- Enable intuitive visuals of user data



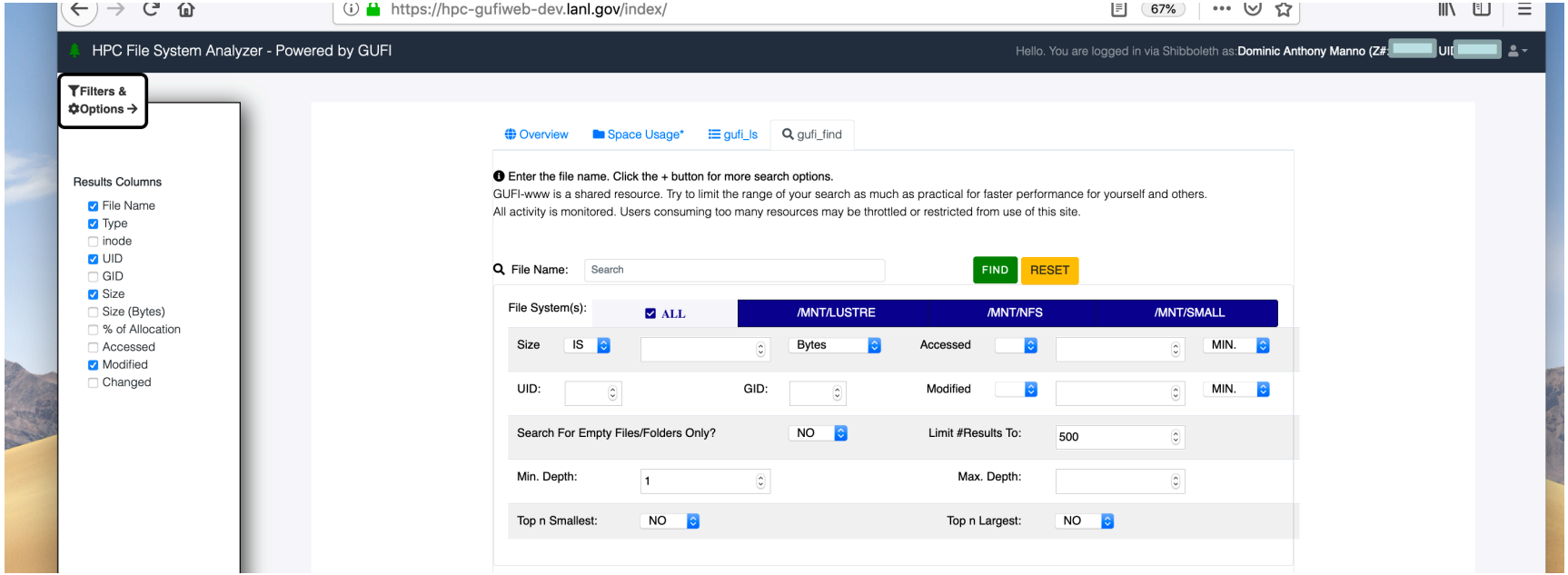


*images from qdirstat – windirstat linux variant

Reports and Web Interface



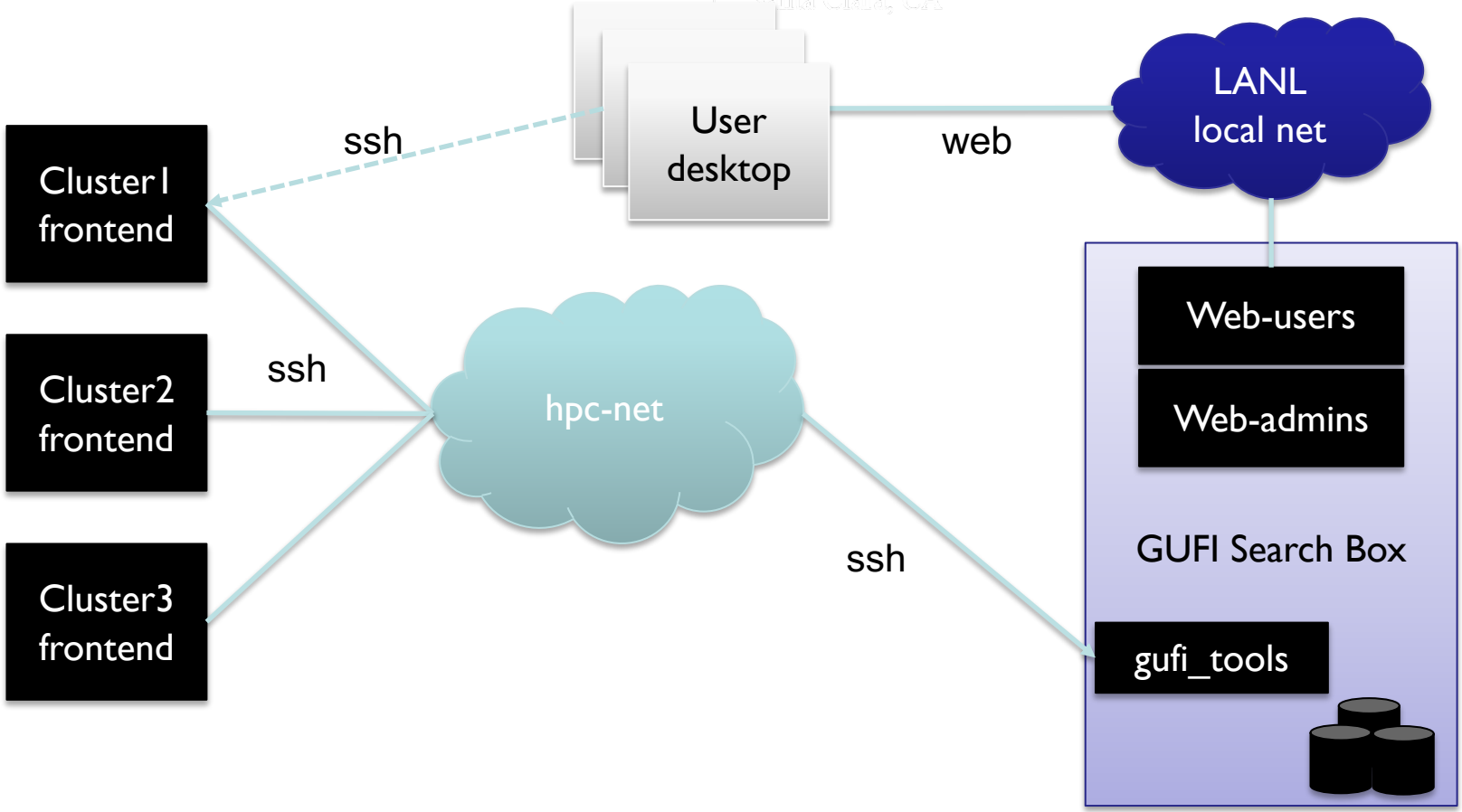
Reports and Web Interface



User Interaction Phase I

- Web – user built and canned reports
- Deploy to allow ease of use for users
- Encourages more use and demonstrates power of index
- Allow remote queries

Phase I Model



Performance Path

Test Setup

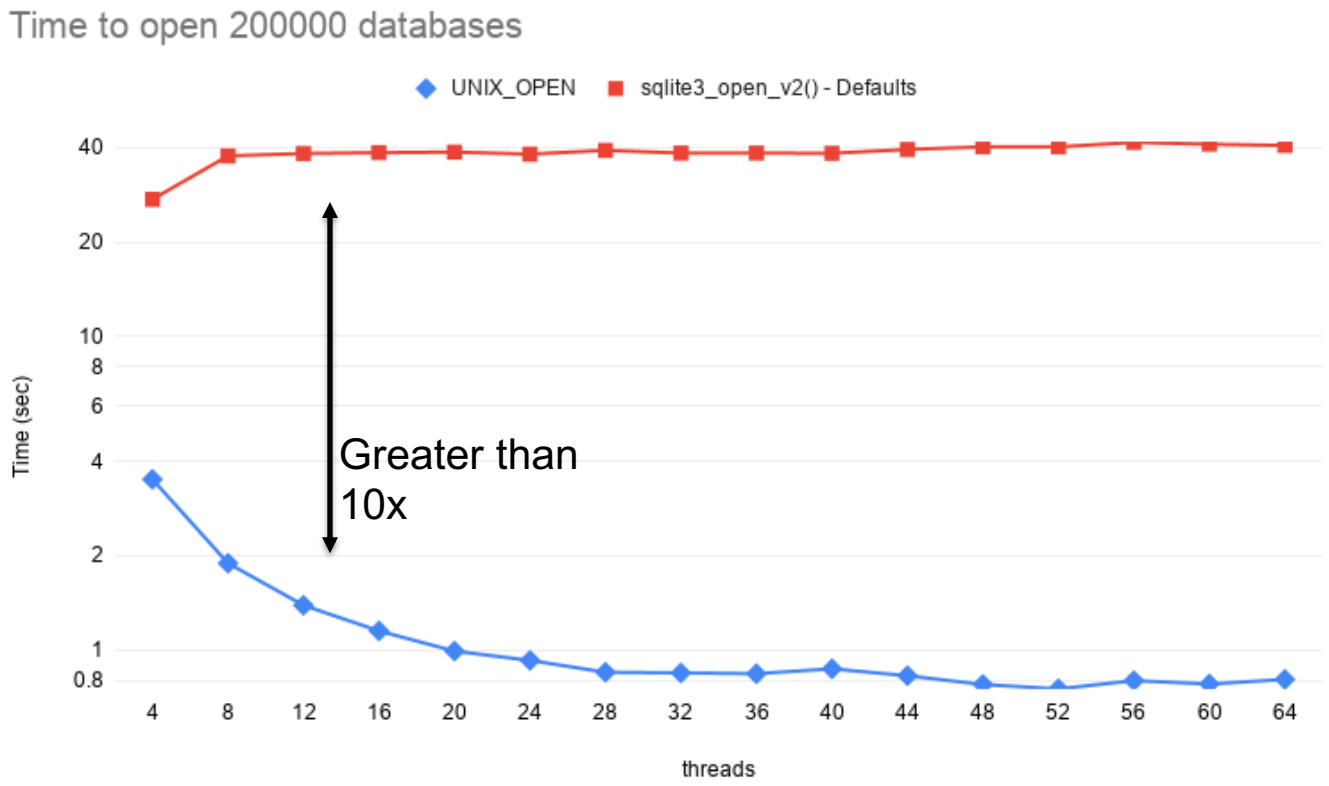
- Single server, Dell R7425
- CPU: AMD Epyc 7401
- Memory: 512 GB
- Kernel 3.10
- Using NVMe SSDs – reported results are only using 1 SSD (unless otherwise specified)
- XFS filesystem

Earliest Performance from Real Trees

- OK – not what we expected
- Best case ~25x over POSIX tools
- Worst case only ~4x

Opening DBs

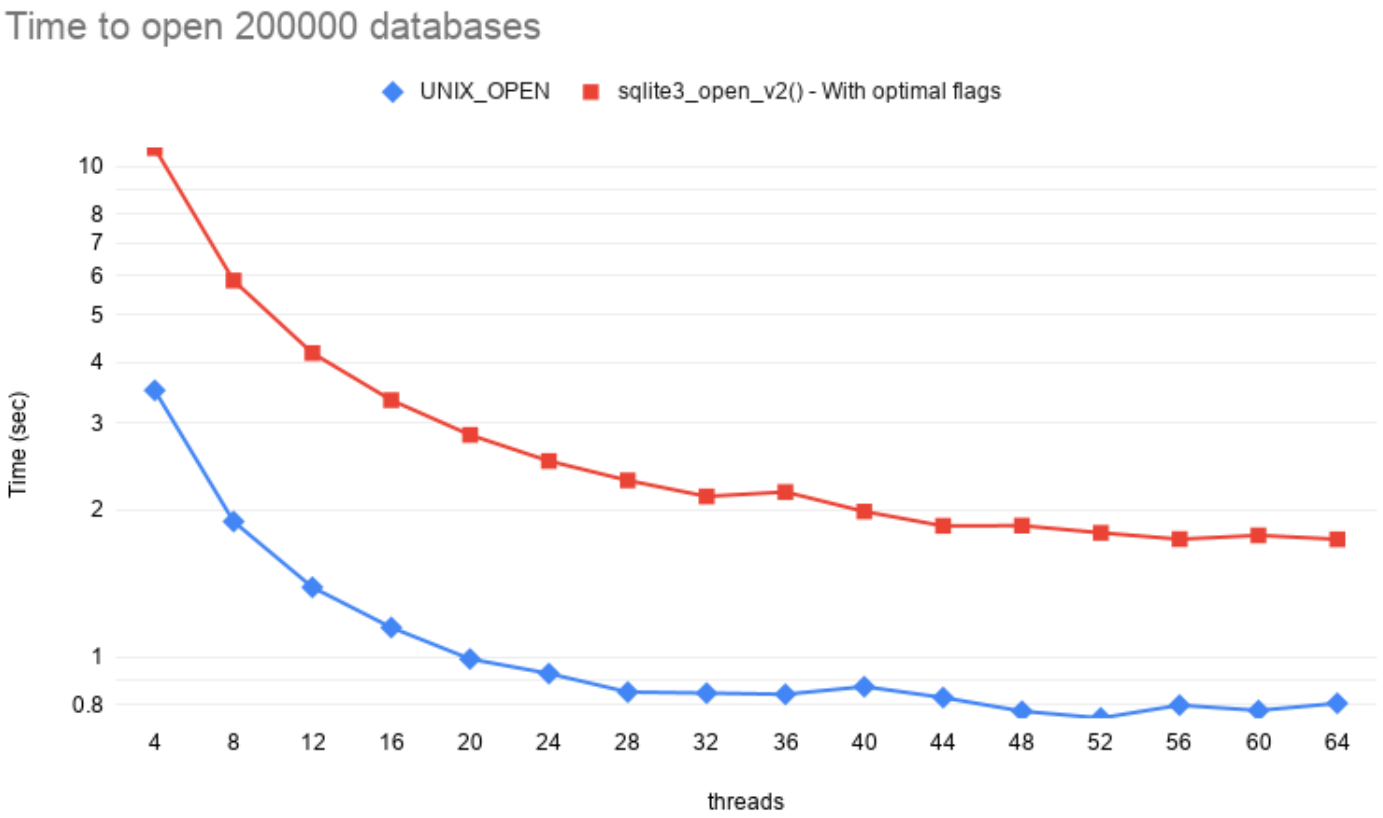
Opening DBs



Tuning SQLite

- Sqlite3 has protections
- No need for multiple threads to ever access the same DB at the same time
- VFS: unix-none
- Thread-safe = 0

Opening DBs -- Improved



Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	294,188	294,188	13,360,753	13,229,405
Dirs	13,012	13,012	1,633,564	1,622,424
Time	32.1	0.47	2,040	39.2
Files/sec	9,164	625,931	6,549	337,484

Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	**294,188**	**294,188**	**13,360,753**	**13,229,405**
Dirs	13,012	13,012	1,633,564	1,622,424
Time	32.1	0.47	2,040	39.2
Files/sec	9,164	625,931	6,549	337,484

Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	294,188	294,188	13,360,753	13,229,405
Dirs	13,012	13,012	1,633,564	1,622,424
Time	**32.1**	**0.47**	2,040	39.2
Files/sec	**9,164**	**625,931**	6,549	337,484

68x

Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	294,188	294,188	13,360,753	13,229,405
Dirs	13,012	13,012	1,633,564	1,622,424
Time	32.1	0.47	**2,040**	**39.2**
Files/sec	9,164	625,931	**6,549**	**337,484**
		68x		**51x**

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	**22,771,329**	**22,509,652**	**119,296,067**	**118,509,899**	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	531.6	14.5	11,309	134.2	-	14.9
Files/s	42,835	1,553,956	10,548	883,413	-	1,511,553

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	22,771,329	22,509,652	119,296,067	118,509,899	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	**531.6**	**14.5**	11,309	134.2	-	14.9
Files/s	**42,835**	**1,553,956**	10,548	883,413	-	1,511,553

36x

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	22,771,329	22,509,652	119,296,067	118,509,899	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	531.6	14.5	**11,309**	**134.2**	-	14.9
Files/s	42,835	1,553,956	**10,548**	**883,413**	-	1,511,553
		36x		**84x**		

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	22,771,329	22,509,652	119,296,067	118,509,899	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	531.6	14.5	11,309	134.2	-	**14.9**
Files/s	42,835	1,553,956	10,548	883,413	-	**1,511,553**
		36x		**84x**		

Index Creation Results

- 13,229,405 files from NFS home filesystem into GUFI tree: 38.4s -- 344,515 files/s
- 118,509,899 files from lustre filesystem into GUFI tree: 148.9s -- 795,902 files/s
- 154,045,072 files from HPSS archive into GUFI tree: ~175 sec -- 880,257 files/s

jemalloc Improvements

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	294,188	294,188	13,360,753	13,229,405
Dirs	13,012	13,012	1,633,564	1,622,424
Time	32.1	~~0.47~~ 0.48	2,040	~~39.2~~ 22.9
Files/sec	9,164	~~625,931~~ 612,891	6,549	~~337,484~~ 576,947
		~~68x~~ ~68x		~~51x~~ ~89x

jemalloc improvements

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	22,771,329	22,509,652	119,296,067	118,509,899	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	531.6	~~14.5~~ 11.5	11,309	~~134.2~~ 110.7	-	~~14.9~~ 11.5
Files/s	42,835	~~1,553,956~~ 1,957,361	10,548	~~883,413~~ 1,070,550	-	~~1,511,553~~ 1,958,446
		~~36x~~ 46x		~~84x~~ 101x		

Rollup Optimization

- Find “like” permissions in sub-tree, if allowed, place all information into single DB higher up in tree. (post-process of created gufi tree)
- Rollup generalization on NFS home tree:
  - Scripts for analysis available.
  - At Level 1: 1197 rollups possible (2004 dirs, ~60%)
- Expected speedup of 5-10x
- More complex? updates

Rollup Optimization

- NFS was 68x over normal tools, so expect over 100x, likely much more
- Lustre was 46x over normal tools, so expect over 80x, likely much more
- Those are for users, admins queries can be flattened further

(extrapolations from early test runs on dev box)

Next Steps in Performance

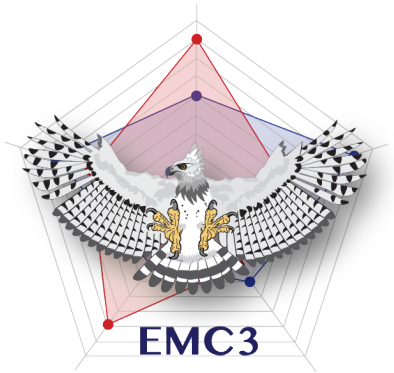
- Sharding
  - Likely to see speedup, but need to incorporate into code
- FS comparison
  - User-space?
- Roll-up integration – how far?
- More sqlite_open optimizations

Next Steps in Performance

- Characterization of cost related to different categories of operations
  - Equations to analyze impact of things like rollup level choice and other optimizations

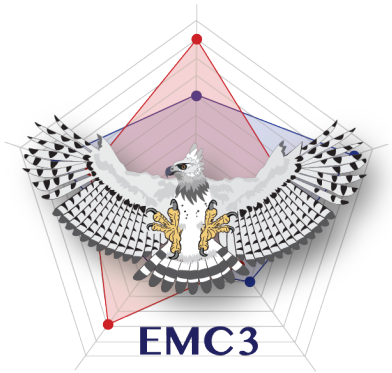
Questions?

- Thank you!
- Contribute, test, file bugs
- Collaborate!



Links

- https://github.com/mar-file-system/GUFI
- Other on-going work and research can be found via Ultrascale Systems Research Center webpage: https://usrc.lanl.gov/
- Join us in our effort to obtain higher efficiency with the Efficient Mission Centric Computing Consortium: https://usrc.lanl.gov/emc3.php

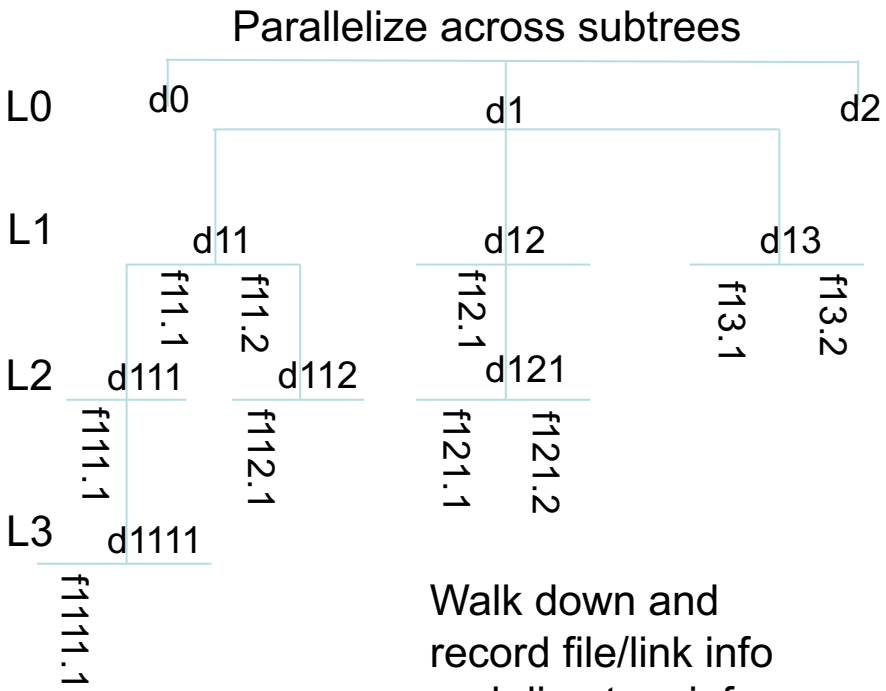


Rollup Rules

Non exhaustive relatively simple rules to see if rollup is legal:

- If any of my children are not rolled up then NO
- When me and all my descendants are o+rx YES
- When me and all my descendants are ugo same
  and same uid, gid then YES
- When me and all my descendants are ug are same
  and same uid and gid and top o-rx then YES
- When me and all my descendants u same and top go-rx
  and same uid then YES
- Else NO

Rollup Step 1

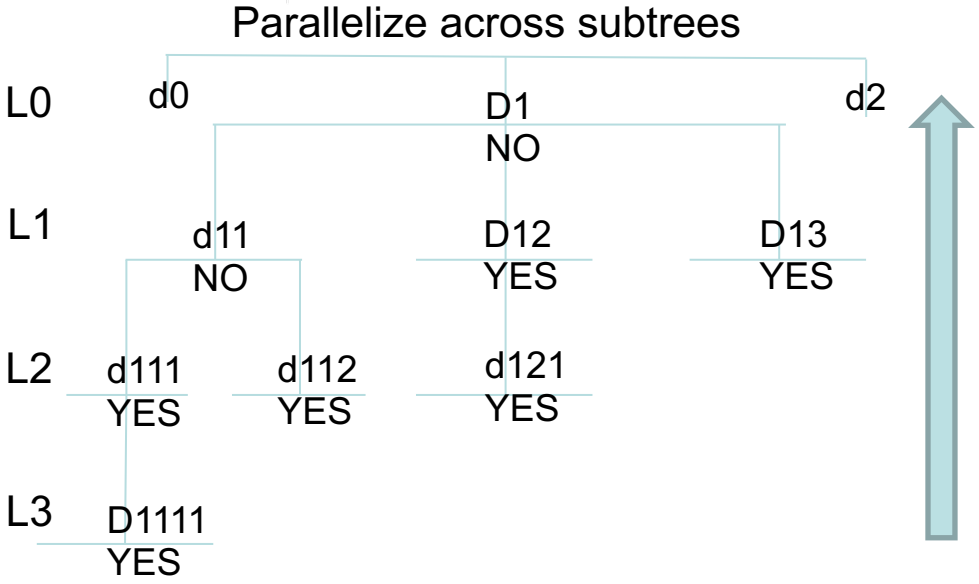


Walk down and record file/link info and directory info (name/path, uid, gid, mode, level, etc.)

Dirs	Files
d1	f11.1
d11	f11.2
d12	f12.1
d13	f13.1
d111	f13.2
d112	f111.1
d121	f112.1
d1111	f121.1
	f121.2
	f1111.1

Rollup Step 2

Dirs
Top,uid,gid,mode,level
D1,uid,gid,mode,level
D11,….
D12,….
D13,…
D111,….
D112,….
D121,….
D1111,….



Process
From L3-4 to L0-1 (decr by 1)
Process level by level up the tree
Determine if each dir at the level can be marked as ”rollable”

Rollup where child is YES and parent is NO
D111, D112, D12, D13
NOT D1111, D121 (as parent is YES)
Not D1, D11 (as child is NO)
Totals: total of 8 but 6 rolled into 4 so Dirs to visit went from 8 to 6