# GUFI (Grand Unified File Indexer)
# What does it have to offer you?

**03/2025**

**Gary Grider**
**Los Alamos National Laboratory**
LA-UR-25-22418

# Background: POSIX protects Exabytes of data in LANL/everywhere

File permissions (access file contents – read, modify file contents – write, execute a file – execute – that was easy (but there is more…)

- Directory permissions (dirs are different – the interpretation of rwx is different)
  - List file names and inode numbers only in a directory – dir read, file permissions irrelevant
  - Add/del a file into a directory – dir write, file permissions irrelevant
  - CD into the directory – dir execute, file permissions irrelevant
  - Get information from the inode (size,dates,etc.) – dir execute, file permissions irrelevant
  - Well that's not too bad but there is more and an important item
  - To search (traverse) in a tree you need dir execute and dir read (oh my but there is more
  - Almost everything you do traverses, like cat /u/me/files/file – guess what, to access the file contents you need file read permissions – but before you ever get there you need to traverse to that file (even if you know the path the file system has to traverse to it to get the inode and then determine if you have file read permissions. So not only do you need file read on that file you need execute on all the directories above, so you would need dir execute permission in /u, /u/me, /u/me/files.
  - Well, isn't that special!

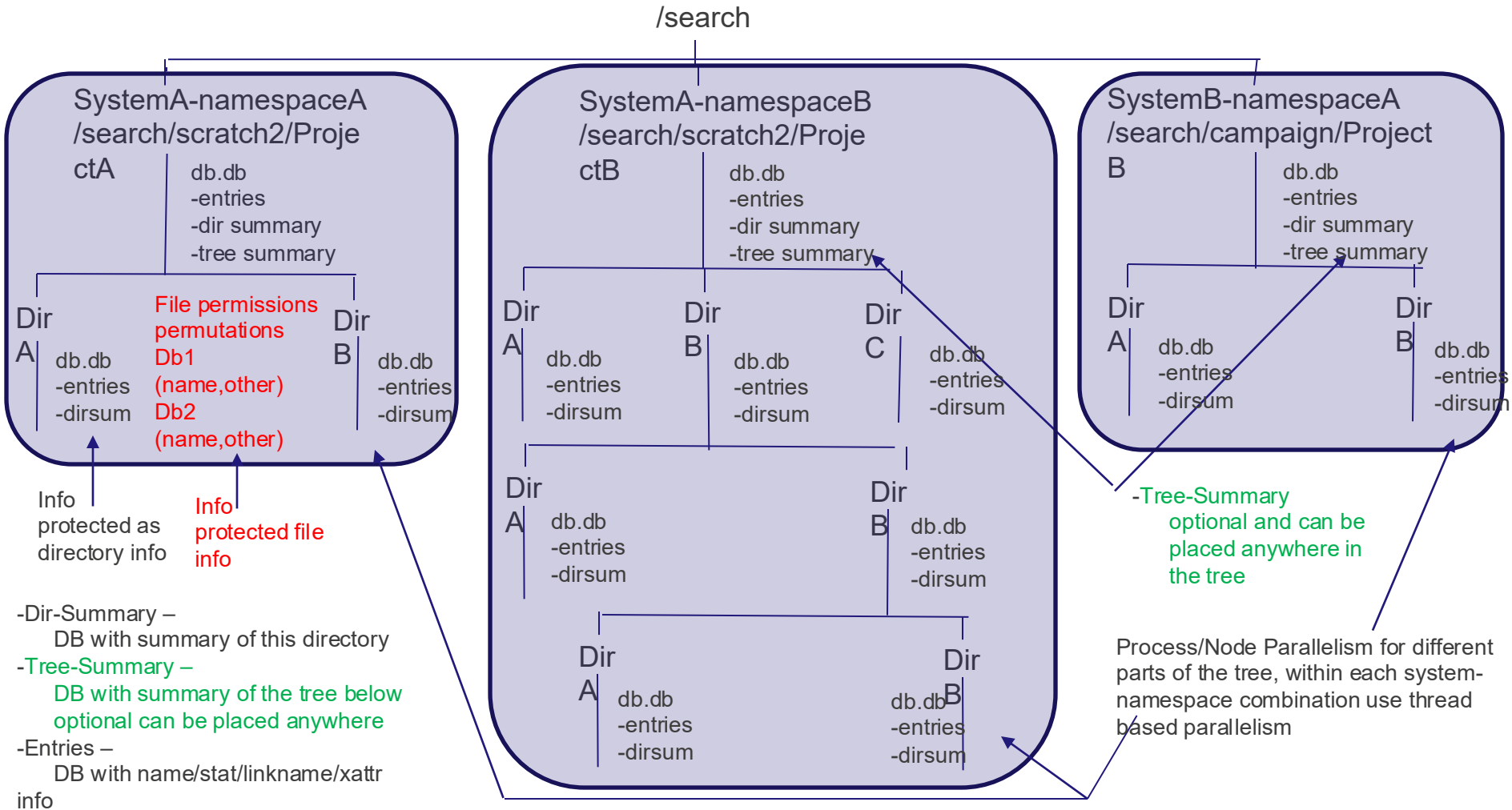
# GUFI Goals

- Unified index over home, project, scratch, campaign, and archive
- Metadata only, metadata protected as the directory and as the files
- Shared index for users and admins
- Parallel search capabilities that are very fast (minutes for billions of files/dirs)
- Can appear as mounted file system where you get a virtual image of your file metadata based on query input
- Full/Incremental update from sources with reasonable update time/annoyance
- So fast people think its broken
- Leverage existing tech as much as possible both hdwr and software:  flash, threads, clusters, sql as part of the interface, commercial db tech, commercial indexing systems, commercial file system tech, threading/parallel process/node run times, src file system full/incremental capture capabilities, posix tree attributes (permissions, hierarchy representation, etc.), open source/agnostic to leveraged parts where possible.
- Simple so that an admin can easily understand/enhance
- It must make massive complex tree walk/joins/concatenations look like a table to the user!

# Initial Design Thoughts

- **Why not a flat namespace?**
  - Performance is great, but…
  - Rename high in the tree is terribly costly
  - Security becomes a nightmare if users/admins can access the namespace
- **Leverage things that already work well, reduce required records to scan:**
  - POSIX permissions / tree walk (readdir+)
  - Breadth first search for parallelization
  - Our trees have inherent namespace divisions for parallelism
  - Embedded DBs are fast if not many joins and individual DB size < TB
  - Flash storage is cheap enough to hold everything with order ~10K IOPs each
  - Entries in file system reduce to essentially <dir count> * 3
  - Dense directories reduce footprint dramatically
  - SQL is easily utilized for general queries of attributes

# GUFI – how does it work?




-Dir-Summary –
DB with summary of this directory
-Tree-Summary –
DB with summary of the tree below
optional can be placed anywhere
-Entries –
DB with name/stat/linkname/xattr info
for each file or link

# Draft Schemas

- Parent-Inode mapping file “directories-parent-inode directories Inode”
  - Parent inode is only kept for directories, not for files as that kills rename/move function
- "CREATE TABLE entries(
  - name TEXT PRIMARY KEY, name of file (Not path due to renames)
  - type TEXT, inode INT, f for file l for link inode
  - mode INT, posix mode bits
  - nlink INT, number of links
  - uid INT, gid INT, uid and gid
  - size INT, blksize INT, size and blocksize
  - blocks INT, blocks
  - atime INT, access time
  - mtime INT, file contents modification time
  - ctime INT, metadata change time
  - linkname TEXT, if link this is path to link
  - xattrs TEXT);"; single text string with key/value pairs with delimitors

# Draft Schemas (continued)

- "CREATE TABLE summary(  summary info for this directory
  - name TEXT PRIMARY KEY,  name not path due to rename
  - type TEXT, inode INT,  d for directory  inode
  - mode INT,  posix mode bits
  - nlink INT,  number of links
  - uid INT, gid INT,  uid gid
  - size INT, blksize INT, blocks INT,  size, blocksize, blocks
  - atime INT, mtime INT, ctime INT,  access time,  dir contents mod time, md chg time
  - linkname TEXT, xattrs TEXT,  if link, path to link, xattrs key/value delimited string
  - totfiles INT, totlinks INT,  tot files in dir, tot links in dir
  - minuid INT, maxuid INT, mingid INT,  maxgid INT,  min and max uid and gid
  - minsize INT, maxsize INT,  minimum file size and max file size
  - totltk INT, totmtk INT, totltm INT,  total number of files lt KB mt KB, lt MB,
  - totmtm INT, totmtg INT, totmtt INT,  total number of files mt MB mt GB, mt TB
  - totsize INT,  total bytes in files in dir
  - minctime INT, maxctime INT,  min max ctime
  - minmtime INT, maxmtime INT,  min max mtime
  - minatime INT, maxatime INT,  min max mtime
  - minblocks INT, maxblocks INT,  min max blocks
  - totxattr INT,  number of files with xattrs
  - depth INT);";  depth this directory is in the tree

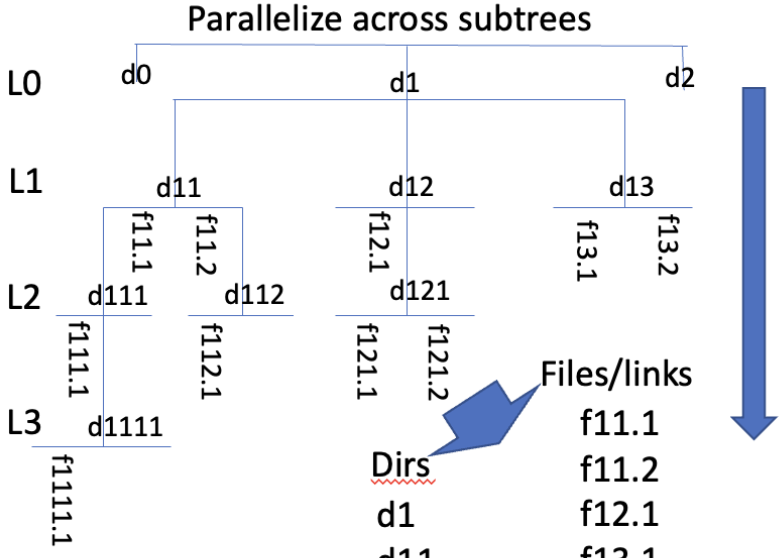
# Draft Schemas (continued)

- "CREATE TABLE treesummary( summary info for this directory
  - totsubdirs INT, tot subdirs in tree
  - maxsubdirfiles INT, maxsubdirlinks INT, maxfiles in a subdir max links in a subdir
  - maxsubdirsize INT, most bytes in any subdir
  - totfiles INT, totlinks INT, tot files in tree, tot links in tree
  - minuid INT, maxuid INT, mingid INT, maxgid INT, min and max uid and gid
  - minsize INT, maxsize INT, minimum file size and max file size
  - totltk INT, totmtk INT, totltm INT, total number of files lt KB mt KB, lt MB,
  - totmtm INT, totmtg INT, totmtt INT, total number of files mt MB mt GB, mt TB
  - totsize INT, total bytes in files in tree
  - minctime INT, maxctime INT, min max ctime
  - minmtime INT, maxmtime INT, min max mtime
  - minatime INT, maxatime INT, min max mtime
  - minblocks INT, maxblocks INT, min max blocks
  - totxattr INT, number of files with xattrs
  - depth INT);"; depth this tree summary is in the tree

# But will it be fast you you have millions of directories which means millions of databases

- Well it was pretty descent performance but not fast enough using lots of threads and breadth first search
- So we need to shard – how do we shard when we need to keep the
- Permission Permutations Sharding – Rollups
- Make an GUFI index, then go to the bottom of the tree and walk up and combine information until you reach a place where rollup rules no longer work, or you have enough info aggregated for efficiency
- Rollups are fast to perform and turn gufi index search into a bandwidth play, so fast its almost unbelievable

# Step one and rules for rollup

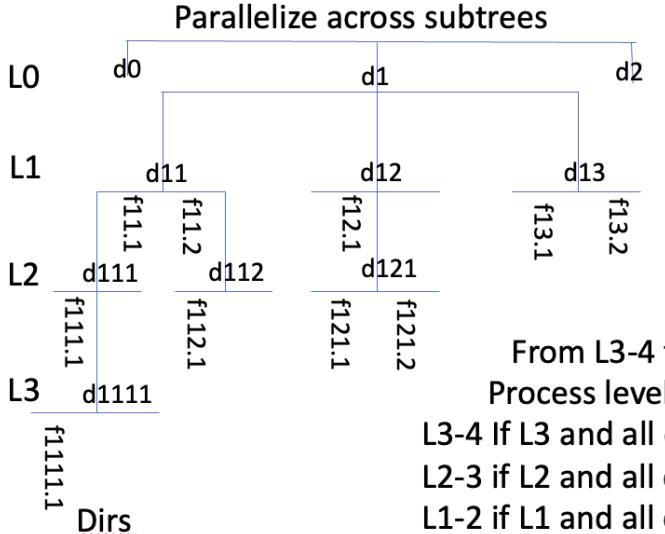


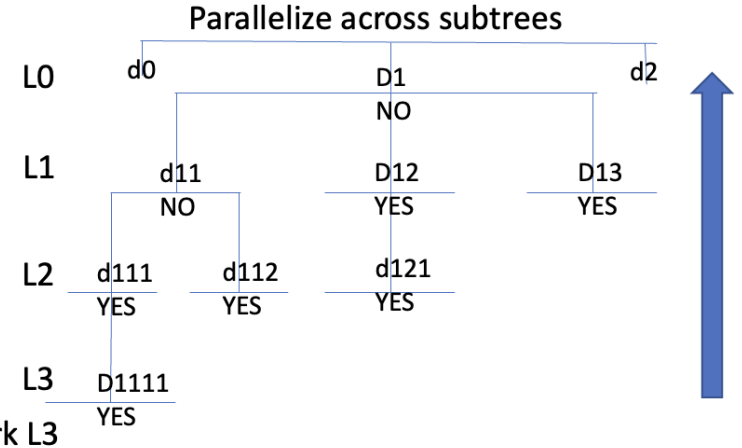

Walk down and record file/link info and directory info (name/path, uid, gid, mode, level, etc.)

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 descendents are o+rx YES
- When me and all my descendents are ugo same and same uid, gid then YES
- When me and all my descendents are ug are same and same uid and gid and top o-rx then YES
- When me and all my descendents u same and top go-rx and same uid then YES
- Else NO

# Rollup process example



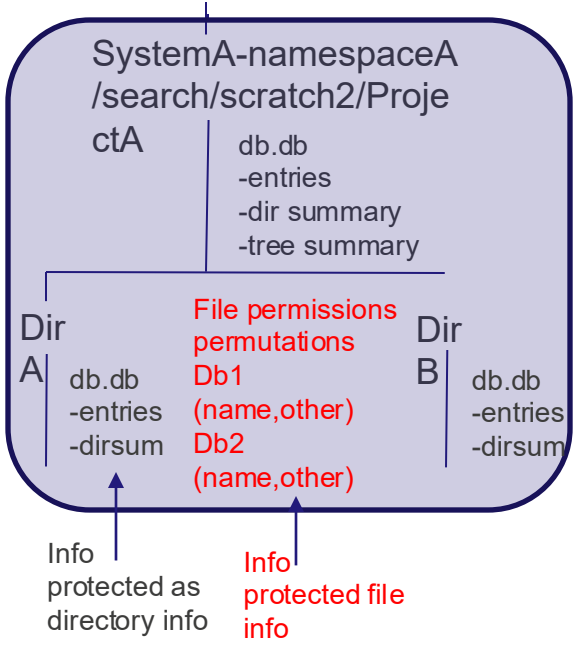

Dirs
D1,uid,gid,mode,level
D11,….
D12,….
D13,…
D111,….
D112,….
D121,….
D1111,….

Process
From L3-4 to L0-1 (decr by 1)
Process level by level up the tree
L3-4 If L3 and all children (none) can roll mark L3
L2-3 if L2 and all children (L3) can roll mark L2
L1-2 if L1 and all children (L2) can roll mark L1
L0-1 if L0 and all children (L1) can roll mark L0
Insert top directory at (L-1) NO roll

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

Dirs - min/max urx,grx,orx, min/max uid,gid
Top, … … NO – this is always NO
D1,…NO
D11,….NO
D12,….YES
D13,…YES
D111,….YES
D112,….YES
D121,…. YES
D1111,…. YES

# Protecting directory level metadata and File level metadata



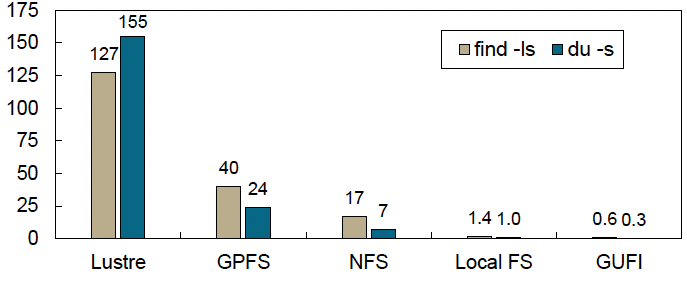

Db.db file is protected the same permissions as the directory, one file per directory (rolled up)

File permissions is how metadata about files (like xattr values, other metadata pulled from files) should be protected

- One DB file per permission permutation
- Roll-in processing concatenates all the db tables that the user can access

# GUFI is State of the Art FAST

- *1) List all file names accessible by the user (top level for root).*
- *2) Print the size and name of every directory accessible by the user (top level for root)..*
- *3) Print the space used by the user's home directory (top level for root).*
- *4) Print the space used by the user's home directory (top level for root).*




List contents/attributes of Linux distro GUFI compared to Linux utilities

| | |
|---|---|
| **Server** | HPE DL380 |
| **Processor** | Dual Intel Xeon 8280 2.7GHz (56 total cores, 112 threads) |
| **Main Memory** | 192GiB |
| **Storage** | 4x Samsung 1725A 1.6TB |
| **Operating System** | CentOS 7.7.1908 |
| **Kernel Version** | Linux 5.7.9 |
| **File System** | XFS |
| **Dataset 1** | 1.6M directories, 13.2M files |
| **Dataset 2** | 2.2M directories, 64.7M files |

Dataset 2 Typical Query Result

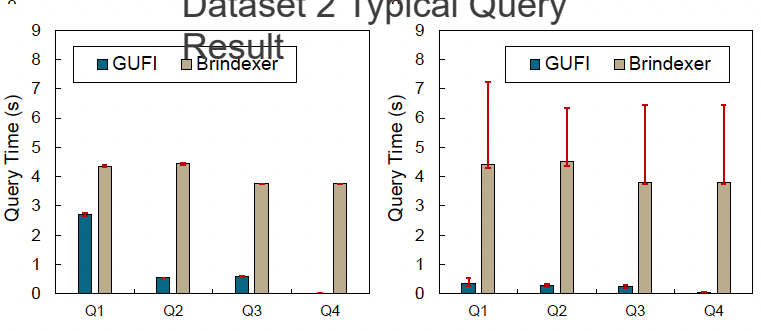



Fig. 10: Comparison of GUFI and Brindexer Query Performance. GUFI provides average speedups of 1.5x - 230x for administrator compared to the Brindexer index using a real file system namespace. User-specific queries result in even greater speedups; however, only GUFI can be accessed safely by users.

GUFI compared to state of the art (which is a variant of GUFI tech which dwarfs the other industry solutions).

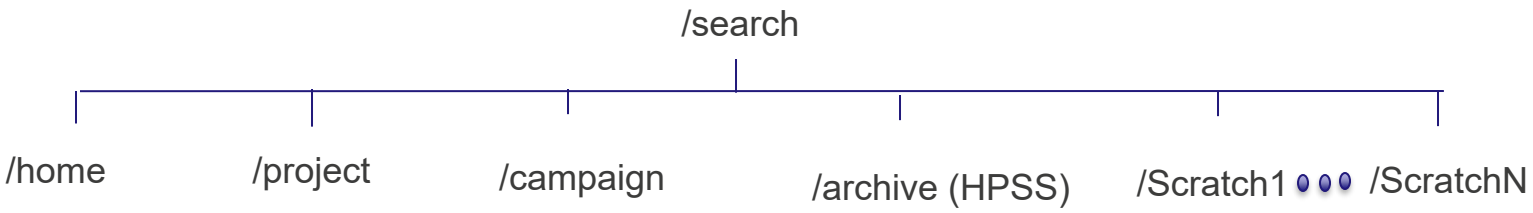
# But how am I going to think about querying my hundreds of thousands of databases

- **Gufi makes all those databases appear as one database – really one table**
- **Gufi can be connected to workflows/data staging/movement tools (conduit/pftool/etc.)**
- **Schema can be changed/added to, just like an sql database**
- **You can use POSIX XATTRS on files and GUFI will index that securely**
- **But there is an extremely powerful feature called external databases**
  - You can have a single external database for information that is for sure common need to know and that can be joined with
  - A more powerful external database feature: databases in the source file system directory that information is about (about the directory or file(s) in that directory)
  - Gufi will index the existence of these external DB's and then you can join user info with GUFI file system info in a completely secure way and if you change permissions in the source file system, GUFI reindexing will enable permissions to change and GUFI will behave like the file system permissions

# How do users, admins, data scientists use it?

- **Simple for users (make these commands available on front ends and fta's) (make the syntax for these commands as close to find/start/ls as possible and the output as well). MAKE IT SO FAST USERS THINK ITS BROKEN ☺**
  - Gufi_find
  - Gufi_stat
  - Gufi_ls
- **Sysadmins and data scientists**
  - Gufi_query (extremely powerful)
  - Storage admins see all, data scientist see what they can see as a user
  - Highly threaded, breadthfirst search, can create output, outputdb's, aggregations of the threads results, connect to external databases of all kinds (user supplied, full text search, ai/ml embedded vector sim search, etc.

# What does the gufi index tree look like?




- **You can search it all by starting your search at /search**
- **If you want to start your search lower you can either by min depth/maxdepth parameters or list subtrees you want to search**

# Vrpentries

## Even supports regex

```
ggrider@pn2201328 src % ./gufi_query -n1 -d
'|' -E "select path(),name,size from vrpent
ries where regexp('f1.big$',name);" testidx
testidx/l1.3/l2.3|f1.big|4
ggrider@pn2201328 src %
```

## simple query

```
ggrider@pn2201328 src % ./gufi_query
-n1 -d'|' -E 'select name,uid,size,mo
de,mtime from vrpentries;' testidx
f2.1.doc|2078|2|33188|1680818365
f3.1.doc|2078|2|33188|1680818372
f1.1.doc|2078|2|33188|1680818343
f2.1.doc|2078|2|33188|1680818356
f1.xls|2078|0|33188|1680817472
f1.xls|2078|0|33188|1680817476
f1.xls|2078|0|33188|1680817469
f1.bigger|2078|7|33188|1680817672
f1.big|2078|4|33188|1680817656
f1.biggest|2078|10|33188|1680817700
f1.exe|2078|0|33188|1680817412
f1.tar|2078|0|33188|1680817430
f1.exe|2078|0|33188|1680817415
f1.tar|2078|0|33188|1680817424
f1.exe|2078|0|33188|1680817390
f1.tar|2078|0|33188|1680817439
ggrider@pn2201328 src %
```

## Powerful functions

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -E "select path()||name,uidtouser(uid),s
ize,modetotxt(mode),strftime('%a %b %e %H:%M:%S %Y', mtime) from vrpentries;" testidx
testidx/l1.2f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:25 2023
testidx/l1.3f3.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:32 2023
testidx/l1.1f1.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:03 2023
testidx/l1.1f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:16 2023
testidx/l1.2/l2.2f1.xls|ggrider|0|-rw-r--r--|Thu Apr  6 15:44:32 2023
testidx/l1.2/l2.3f1.xls|ggrider|0|-rw-r--r--|Thu Apr  6 15:44:36 2023
testidx/l1.2/l2.1f1.xls|ggrider|0|-rw-r--r--|Thu Apr  6 15:44:29 2023
testidx/l1.3/l2.2f1.bigger|ggrider|7|-rw-r--r--|Thu Apr  6 15:47:52 2023
testidx/l1.3/l2.3f1.big|ggrider|4|-rw-r--r--|Thu Apr  6 15:47:36 2023
testidx/l1.3/l2.1f1.biggest|ggrider|10|-rw-r--r--|Thu Apr  6 15:48:20 2023
testidx/l1.1/l2.2f1.exe|ggrider|0|-rw-r--r--|Thu Apr  6 15:43:32 2023
testidx/l1.1/l2.2f1.tar|ggrider|0|-rw-r--r--|Thu Apr  6 15:43:50 2023
testidx/l1.1/l2.3f1.exe|ggrider|0|-rw-r--r--|Thu Apr  6 15:43:35 2023
testidx/l1.1/l2.3f1.tar|ggrider|0|-rw-r--r--|Thu Apr  6 15:43:44 2023
testidx/l1.1/l2.1f1.exe|ggrider|0|-rw-r--r--|Thu Apr  6 15:43:10 2023
testidx/l1.1/l2.1f1.tar|ggrider|0|-rw-r--r--|Thu Apr  6 15:43:59 2023
```

## Powerful functions with a simple where

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -E "select path()||name,uidtouser(uid),s
ize,modetotxt(mode),strftime('%a %b %e %H:%M:%S %Y', mtime) from vrpentries where size
> 0;" testidx
testidx/l1.2f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:25 2023
testidx/l1.3f3.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:32 2023
testidx/l1.1f1.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:03 2023
testidx/l1.1f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:16 2023
testidx/l1.3/l2.2f1.bigger|ggrider|7|-rw-r--r--|Thu Apr  6 15:47:52 2023
testidx/l1.3/l2.3f1.big|ggrider|4|-rw-r--r--|Thu Apr  6 15:47:36 2023
testidx/l1.3/l2.1f1.biggest|ggrider|10|-rw-r--r--|Thu Apr  6 15:48:20 2023
```

# Vrpentries has both file and directory info in the record

## Find the files in directories that have more than 1 file

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -S "select path()||
name, size from vrpentries where dtotfile>1;" testidx
testidx/l1.1f1.1.doc|2
testidx/l1.1f2.1.doc|2
testidx/l1.1/l2.2f1.exe|0
testidx/l1.1/l2.2f1.tar|0
testidx/l1.1/l2.3f1.exe|0
testidx/l1.1/l2.3f1.tar|0
testidx/l1.1/l2.1f1.exe|0
testidx/l1.1/l2.1f1.tar|0
ggrider@pn2201328 src %
```

## Find the files in directories that are in a directory with %1.1 in the directory name

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -S "select path()||
name, size from vrpentries where dname like '%1.1';" testidx
testidx/l1.1f1.1.doc|2
testidx/l1.1f2.1.doc|2
```

## Note the directory summary fields that are in the vrpentries records are prefaced with a d

# Aggregating

order by desc, but wait it didn't work

-E runs that query in every directory, so this order by desc ordered for every directory but not over all the directories/threads

-I makes a out db for each thread
-K makes a single aggregate db
-E inserts each thread output into that threads out db
-J inserts each threads out into global aggregate
-G selects from aggregate

So you can sort/group by over all threads/directories with aggregates

This allows all threads to run as fast as they can until the end and then aggregates so you can do last sort/sum/group/etc. over all the data from all dirs. visited for all threads

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -E "select path()||name,uidtouser(uid),s
ize,modetotxt(mode),strftime('%a %b %e %H:%M:%S %Y', mtime) from vrpentries where size
> 0 order by size desc;" testidx
testidx/l1.2f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:25 2023
testidx/l1.3f3.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:32 2023
testidx/l1.1f1.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:03 2023
testidx/l1.1f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:16 2023
testidx/l1.3/l2.2f1.bigger|ggrider|7|-rw-r--r--|Thu Apr  6 15:47:52 2023
testidx/l1.3/l2.3f1.big|ggrider|4|-rw-r--r--|Thu Apr  6 15:47:36 2023
testidx/l1.3/l2.1f1.biggest|ggrider|10|-rw-r--r--|Thu Apr  6 15:48:20 2023
```

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -I "create table outtable(opath text, on
ame text, ouser text, osize int64, omode text, omtime text);" -K "create temp table agg
table(apath text, aname text, auser text, asize int64, amode text, amtime text);" -E "i
nsert into outtable select path(),name,uidtouser(uid),size,modetotxt(mode),strftime('%a
 %b %e %H:%M:%S %Y', mtime) from vrpentries where size > 0;" -J "insert into aggtable s
elect * from outtable;" -G "select apath||aname,auser,asize,amode,amtime from aggtable
order by asize desc;" testidx
testidx/l1.3/l2.1f1.biggest|ggrider|10|-rw-r--r--|Thu Apr  6 15:48:20 2023
testidx/l1.3/l2.2f1.bigger|ggrider|7|-rw-r--r--|Thu Apr  6 15:47:52 2023
testidx/l1.3/l2.3f1.big|ggrider|4|-rw-r--r--|Thu Apr  6 15:47:36 2023
testidx/l1.2f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:25 2023
testidx/l1.3f3.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:32 2023
testidx/l1.1f1.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:03 2023
testidx/l1.1f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:16 2023
```

# Start/min/max depth

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -E "select path()||name,size from vrpentries;" testidx
testidx/l1.2f2.1.doc|2
testidx/l1.3f3.1.doc|2
testidx/l1.1f1.1.doc|2
testidx/l1.1f2.1.doc|2
testidx/l1.2/l2.2f1.xls|0
testidx/l1.2/l2.3f1.xls|0
testidx/l1.2/l2.1f1.xls|0
testidx/l1.3/l2.2f1.bigger|7
testidx/l1.3/l2.3f1.big|4
testidx/l1.3/l2.1f1.biggest|10
testidx/l1.1/l2.2f1.exe|0
testidx/l1.1/l2.2f1.tar|0
testidx/l1.1/l2.3f1.exe|0
testidx/l1.1/l2.3f1.tar|0
testidx/l1.1/l2.1f1.exe|0
testidx/l1.1/l2.1f1.tar|0
```

## Max depth

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -z1 -E "select path()||name,size from vrpentries;" testidx
testidx/l1.2f2.1.doc|2
testidx/l1.3f3.1.doc|2
testidx/l1.1f1.1.doc|2
testidx/l1.1f2.1.doc|2
```

## Start lower

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -E "select path()||name,size from vrpentries;" testidx/l1.1
testidx/l1.1f1.1.doc|2
testidx/l1.1f2.1.doc|2
testidx/l1.1/l2.2f1.exe|0
testidx/l1.1/l2.2f1.tar|0
testidx/l1.1/l2.3f1.exe|0
testidx/l1.1/l2.3f1.tar|0
testidx/l1.1/l2.1f1.exe|0
testidx/l1.1/l2.1f1.tar|0
```

## Min/Max Depth

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -z2 -y2 -E "select path()||name,size from vrpentries;" testidx
testidx/l1.2/l2.2f1.xls|0
testidx/l1.2/l2.3f1.xls|0
testidx/l1.2/l2.1f1.xls|0
testidx/l1.3/l2.2f1.bigger|7
testidx/l1.3/l2.3f1.big|4
testidx/l1.3/l2.1f1.biggest|10
testidx/l1.1/l2.2f1.exe|0
```

# Limiting

## This orders for each thread and limits to 1 record per directory that thread encounters, the in memory out table will grow during the run

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -I "create table outtable(opath text, oname text, ouser text, osize int64, omode
 text, omtime text);" -K "create temp table aggtable(apath text, aname text, auser text, asize int64, amode text, amtime text);
" -E "insert into outtable select path(),name,uidtouser(uid),size,modetotxt(mode),strftime('%a %b %e %H:%M:%S %Y', mtime) from
vrpentries where size > 0 order by size limit 1" -J "insert into aggtable select * from outtable;" -G "select apath||aname,ause
r,asize,amode,amtime from aggtable order by asize desc;" testidx
testidx/l1.3/l2.1f1.biggest|ggrider|10|-rw-r--r--|Thu Apr  6 15:48:20 2023
testidx/l1.3/l2.2f1.bigger|ggrider|7|-rw-r--r--|Thu Apr  6 15:47:52 2023
testidx/l1.3/l2.3f1.big|ggrider|4|-rw-r--r--|Thu Apr  6 15:47:36 2023
testidx/l1.2f2.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:25 2023
testidx/l1.3f3.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:32 2023
testidx/l1.1f1.1.doc|ggrider|2|-rw-r--r--|Thu Apr  6 15:59:03 2023
```

## This orders for each thread and limits to 1 record per directory that thread encounters, and limits the aggregated total to 3

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -I "create table outtable(opath text, oname text, ouser text, osize int64, omode
 text, omtime text);" -K "create temp table aggtable(apath text, aname text, auser text, asize int64, amode text, amtime text);
" -E "insert into outtable select path(),name,uidtouser(uid),size,modetotxt(mode),strftime('%a %b %e %H:%M:%S %Y', mtime) from
vrpentries where size > 0 order by size limit 1" -J "insert into aggtable select * from outtable;" -G "select apath||aname,ause
r,asize,amode,amtime from aggtable order by asize desc limit 3;" testidx
testidx/l1.3/l2.1f1.biggest|ggrider|10|-rw-r--r--|Thu Apr  6 15:48:20 2023
testidx/l1.3/l2.2f1.bigger|ggrider|7|-rw-r--r--|Thu Apr  6 15:47:52 2023
testidx/l1.3/l2.3f1.big|ggrider|4|-rw-r--r--|Thu Apr  6 15:47:36 2023
```

# Tree summary and combining tree summary with other

## Powerful aggregate information in tree summary, summarizes everything below, at every level, which trees have this in them

## DU command is now free (milliseconds)

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -z1 -T "select path(),totsubdirs,totfiles,totsize,maxsize from treesummary;" testidx
testidx|12|16|29|10
testidx/l1.2|3|4|2|2
testidx/l1.3|3|4|23|10
testidx/l1.1|3|8|4|2
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -z1 -T "select path(),totsubdirs,totfiles,totsize,maxsize from treesummary where totxattr>0;" testidx
testidx|12|16|29|10
testidx/l1.1|3|8|4|2
```

## Very powerful way to limit what all directories in what subtrees you scan based on treesummary info

```
testidx/l1.1f2.1.doc
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -z1 -T "select '.' from treesummary where totxattr>0;" -E "select path()||'/'||name from vrpentries;" testidx
.
.
testidx/l1.1/f1.1.doc
testidx/l1.1/f2.1.doc
```

# Xattrs

## Yes you can list or search on xattrs

## Notice using –x in the qufi_query cmd

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -x -E "select path(),nam
e,xattr_name xattr_value from xpentries;" testidx
testidx/l1.2|f2.1.doc|
testidx/l1.3|f3.1.doc|
testidx/l1.1|f1.1.doc|
testidx/l1.1|f2.1.doc|user.garyxattr
testidx/l1.2/l2.2|f1.xls|
testidx/l1.2/l2.3|f1.xls|
testidx/l1.2/l2.1|f1.xls|
testidx/l1.3/l2.2|f1.bigger|
testidx/l1.3/l2.3|f1.big|
testidx/l1.3/l2.1|f1.biggest|
testidx/l1.1/l2.2|f1.exe|
testidx/l1.1/l2.2|f1.tar|
testidx/l1.1/l2.3|f1.exe|
testidx/l1.1/l2.3|f1.tar|
testidx/l1.1/l2.1|f1.exe|
testidx/l1.1/l2.1|f1.tar|
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -x -E "select path(),nam
e,xattr_name xattr_value from xpentries where xattr_name is not null;"
testidx
testidx/l1.1|f2.1.doc|user.garyxattr
```

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -x -E "select path(),nam
e,xattr_name xattr_value from xpentries where xattr_name like '%gary%';
" testidx
testidx/l1.1|f2.1.doc|user.garyxattr
```

## Why is this special?

## Remember xattr values are protected like the file and not like the directory so they are kept in separate db files (potentially – with different permissions)

# Full text ext db example (history db)

**From hpc history gufi db (has full text search words table joined on inode with entries (wordf MATCH 'grider and cdc'**

**Notice twords is total word count for the document**


**Smart Disks in the Extreme HPC Setting**

**Gary Grider**

Division Leader
High Performance Computing Division
Los Alamos National Laboratory
ggrider@lanl.gov
LA-UR 14-26443
Aug 2014

```
ggrider@pn2201328 ~ % ./sqlite/sqlite-src-3180000/gufi-111618/bfq -n1 -Pp -d'|' -E "select path()||'/'||name from entries
 where path()||name like '%Seagate%';" /Volumes/hpchist/idx
/Volumes/hpchist/idx/Z-GaryGriderHistory/recent-presentations/Seagate-Kinetic-LANL-info-v1.pptx
/Volumes/hpchist/idx/Z-GaryGriderHistory/recent-presentations/Seagate-potential-CRADA-Info-v1.pptx
/Volumes/hpchist/idx/Z-GaryGriderHistory/trinity/Trinity-Seagate-Storge.jpg
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-documents/106273doc/Seagate-Collab-08182014.docx
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-documents/106273doc/HPC-DO/Seagate-PTS4-gg-kl-draft.docx
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/LANL-Seagate-CRADA.pptx
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/Seagate-Xyratex-visit.pptx
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/Seagate-Kinetic-LANL-info-v1.pptx
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/Seagate-potential-CRADA-Info.pptx
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/Seagate-Kinetic-LANL-info.pptx
/Volumes/hpchist/idx/Z-GaryGriderHistory/pictures/morepics/machinepics/Trinity-Seagate-Storge.jpg
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-documents/106273doc/HPC-DO/LANL-Sandia-Capability/LANL--Sandia-Seagate.doc
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/CCN-9/HEC-IWG-FS-IO-Workshop-FY05/Seagate-presenta
tion.ppt
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/asci/Sgpfs-gfs-dfs/sgpfs-workshop/04_10min_Talks/C
hris_Malakapalli_Seagate/Seagatesgpfs.ppt
ggrider@pn2201328 ~ % ./sqlite/sqlite-src-3180000/gufi-111618/bfq -n1 -Pp -d'|' -E "select path()||'/'||name,twords from
entries inner join words on inode=tinode where wordf MATCH 'grider AND cdc' and path()||name like '%Seagate%';" /Volumes
/hpchist/idx
/Volumes/hpchist/idx/Z-GaryGriderHistory/recent-presentations/Seagate-Kinetic-LANL-info-v1.pptx618
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/Seagate-Xyratex-visit.pptx1475
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/Seagate-Kinetic-LANL-info-v1.pptx618
/Volumes/hpchist/idx/Z-GaryGriderHistory/older-presentations/106273ppt/HPC-DO/Seagate-Kinetic-LANL-info.pptx432
```

# The swiss army knives of data curation/science

## Running commands in queries.

- Intop Run a command as a function passing in anything you want and getting an integer back

```
ggrider@pn2201328 src % ./gufi_query -n1 -d"|" -E
 "select path()||'/'||name,intop('cat testsrc'||s
ubstr(path(),instr(path(),'/'))||'/'||name||' | w
c -c') from vrpentries where size>0;" testidx
testidx/l1.2/f2.1.doc|2
testidx/l1.3/f3.1.doc|2
testidx/l1.1/f1.1.doc|2
testidx/l1.1/f2.1.doc|2
testidx/l1.3/l2.2/f1.bigger|7
testidx/l1.3/l2.3/f1.big|4
testidx/l1.3/l2.1/f1.biggest|10
```

- Strop Run a command as a function passing in anything you want and getting a string back

```
ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -E "select
 strop('wc -c testsrc'||substr(path(),instr(path(),'/'))|
|'/'||name) from vrpentries where size>0;" testidx
      2 testsrc/l1.2/f2.1.doc
      2 testsrc/l1.3/f3.1.doc
      2 testsrc/l1.1/f1.1.doc
      2 testsrc/l1.1/f2.1.doc
      7 testsrc/l1.3/l2.2/f1.bigger
      4 testsrc/l1.3/l2.3/f1.big
     10 testsrc/l1.3/l2.1/f1.biggest
```

- Yes run any command on any thing driven by any query you want across your holdings

**A data scientist/curators dream tool for record oriented ops**

# Nearest Neighbor

## Putting it all together

**ggrider@pn2201328 src % ./gufi_query -n1 -d'|' -a -p "testsrc" -I "create table vvecog(vogname text, volen int64,vodist int64);" -K "create temp table vvecag(vagname text, vaglen int64,vagdist int64);"**

**-S "select ' ' from vrsummary where ' '=setstr('myq',(select 'find \"%s\" -maxdepth 1 -name \"vec*.db\" -exec sqlite3 -separator \",\" \"{}\" \"select vname,vlen,abs(4-vlen) as vdist from vec where vlen between '||case when count(vogname)>=3 then min(volen) else 0 end ||' and '||case when count(vogname)>=3 then max(volen) else 1000000 end ||';\" \\;' from vvecog));"** ***** dynamically improve query for vector records (uses setstr()**

**-E "drop table if exists vvec;create virtual table temp.vvec using run_vt(cols='myvname,myvlen,myvdist',d=',',cmd='{myq}');insert into vvecog select '%s'||'/'||myvname,myvlen,myvdist from temp.vvec inner join vrpentries on name=myvname order by myvdist asc limit 3;** ***** run ever improving query and only keep best 3**

**delete from vvecog where vogname not in (select vogname from vvecog order by vodist asc limit 3);"** ***** trim the aggregate per thread per dir to best 3**

**-J "insert into vvecag select * from vvecog;" -G "select vagname,vaglen,vagdist from vvecag order by vagdist asc limit 3;"** ***** limit the final all threads aggregate to best 3**

**testidx**

Putting it all together, aggregate, reduce to best 3 as you run, trim thread aggregate as you run, adjust the query to represent best range of vector lengths per thread per directory as you run

testsrc/l1.2/f2.1.doc|4|0

testsrc/l1.3/f3.1.doc|4|0

testsrc/l1.1/f2.1.doc|2|2

# External DB Vector, gufi can be or gen dynamically a RAG from Exabyte holdings/billions objects / use vector sim to answer a question you have!

```
Query Index with sqlite-lembed
Matching on "icn"
INSERT INTO temp.lembed_models(name, model)
    SELECT 'model', lembed_model_from_file('snowflake-arctic-embed-m-long-q8_0.gguf');

WITH matches AS (
    SELECT
        tinode,
        distance
    FROM embeddings
    WHERE embedding MATCH lembed('model', 'icn')
    ORDER BY distance ASC
    LIMIT 10
)
INSERT INTO intermediate
SELECT
    distance,
    rpath(vrpentries.sname, vrpentries.sroll) || '/' || vrpentries.name
FROM matches
LEFT JOIN vrpentries ON vrpentries.inode == matches.tinode;

0.574531555175781|chistory.snowflake/idx/Bulletins/1991-02-txt-bulletins-included-in-cnews.pdf
0.636207818984985|chistory.snowflake/idx/cnews/1976-12-img-ocr.pdf
0.654783666133881|chistory.snowflake/idx/bits/1994/1994-09.pdf
0.667844891548157|chistory.snowflake/idx/cnews/1977-08-img-ocr.pdf
0.668464779853821|chistory.snowflake/idx/cnews/1977-05-img-ocr.pdf
0.669988930225372|chistory.snowflake/idx/cnews/1986-01-txt.pdf
0.671966791152954|chistory.snowflake/idx/Bulletins/1983-1986-index-txt.pdf
0.672669172286987|chistory.snowflake/idx/cnews/1978-11-img-ocr.pdf
0.673577129840851|chistory.snowflake/idx/cnews/1976-11-img-ocr.pdf
0.676900923252106|chistory.snowflake/idx/cnews/1977-11-img-ocr.pdf
```

AI

The lowest distance document

360000000 24dante 920103162403 0 9201031624dante stext 112/18/85 3 01
Starting in February 1991, the ICN Change Bulletin was incorporated in the Computing and Communications Division News. Issues from February 1991 to the present are available on CFS in /icndoc/news

# Using GUFI as a RAG

- GUFI index plus full text search table with extracted words and vector embedding table with vectors from first part of text all joined on inode
- Provide input phrase for vector nearest neighbor query of GUFI holdings, phrase for full text search, and question for LLM
- Do nearest neighbor + full text search GUFI query and produce top N files and get that text
- Feed that text and the question into LLM
- ./run-run-vecnn-conv
- NO CONTEXT
- what is slingshot?
- WITH GUFI RAG CONTEXT
- ] Slingshot is a high-performance computing (HPC) fabric that provides a scalable, high-bandwidth, and low-latency interconnect for HPC systems. It is based on the Cray Shasta architecture and is optimized for high-performance computing applications. Slingshot provides a high-bandwidth, low-latency interconnect for HPC systems. It is based on the Cray Shasta architecture and is optimized for high-performance computing applications.

# Gufi_vt examples

And if it wasn't powerful enough, making gufi_query look like a table so you can do internal and external db's all appear like a table, like a query that uses gufi native, xattrs, user external db's, full text, and vector similarity lookups all from only the part of the namespace you can see

Above is the fixed field all records table (notice threads 2). Refering to the table forks qufi_query with N threads

Below is non fixed, you provide literally everything gufi_query can do and make it look like a table to consume it with as many hidden threads as you want.

This makes connecting it to other ecosystems easy too, see below

```
ggrider@pn2201328 src % echo -n "SELECT path||'/'
'||name,'|',size  FROM gufi_vt_vrpentries('testi
dx',2) where size>0 order by size desc;" | ./guf
i_sqlite3
testidx/l1.3/l2.1/f1.biggest|10
testidx/l1.3/l2.2/f1.bigger|7
testidx/l1.3/l2.3/f1.big|4
testidx/l1.2/f2.1.doc|2
testidx/l1.1/f1.1.doc|2
testidx/l1.1/f2.1.doc|2
testidx/l1.3/f3.1.doc|2
```

```
ggrider@pn2201328 src % echo -n "CREATE VIRTUAL
 TABLE temp.gufi USING gufi_vt('testidx', E='SE
LECT path() || '/' || name AS fullpath, xattr_n
ame, xattr_value,size FROM vrxpentries where si
ze>0;'); SELECT fullpath,'|', xattr_name,'|' xa
ttr_value,'|',size FROM gufi ORDER BY size;" |
./gufi_sqlite3
testidx/l1.2/f2.1.doc|||2
testidx/l1.3/f3.1.doc|||2
testidx/l1.1/f1.1.doc|||2
testidx/l1.1/f2.1.doc|user.garyxattr||2
testidx/l1.3/l2.3/f1.big|||4
testidx/l1.3/l2.2/f1.bigger|||7
testidx/l1.3/l2.1/f1.biggest|||10
```

# Are there ways to interface to GUFI (Sqlalchemy)

**A few lines of python and you have a gui. Gufi can fit into sqlalchemy which makes it fit into that ecosystem**

**(this is python QTKmpackage)**

```
#!/usr/bin/env python3

import argparse
import sys

import sqlalchemy

from PyQt5.QtWidgets import QApplication, QLineEdit, QTableWidget, QTableWidgetItem,
QVBoxLayout, QWidget

class GUFI_VT_Widget(QWidget):
    def __init__(self, engine):
        super(GUFI_VT_Widget, self).__init__()
        self.conn = engine.connect()

        self.setWindowTitle('GUFI Virtual Table Demo')
        self.setGeometry(0, 0, 600, 400)

        self.layout = QVBoxLayout()

        self.sql_box = QLineEdit()
        self.sql_box.setPlaceholderText('SQL')
        self.sql_box.returnPressed.connect(self.run_query)
        self.layout.addWidget(self.sql_box)

        self.table = QTableWidget()
        self.create_table(None)
        self.layout.addWidget(self.table)

        self.setLayout(self.layout)
```

```
#!/usr/bin/env python3

import argparse
import sys

import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk as gtk

import sqlalchemy

class GUFI_VT_Window(gtk.Window):
    def __init__(self, engine):
        super(GUFI_VT_Window, self).__init__()
        self.conn = engine.connect()

        self.set_title('GUFI Virtual Table Demo')
        self.set_size_request(600, 400)
        self.connect('destroy', self.cleanup)

        # input text box
        self.query_entry = gtk.Entry()

        # button to run SQL
        run_button = gtk.Button.new_with_label('Run')
        run_button.connect('clicked', self.run_query)
```

GUFI Virtual Table Demo@spr01

SELECT name, size, size * 10 FROM gufi_vt_pentries('index')

| | name | size | size * 10 |
|---|---|---|---|
| 1 | CTestTestfile.cmake | 1163 | 11630 |
| 2 | Makefile | 7621 | 76210 |
| 3 | bfwiflat2gufitest | 4408 | 44080 |
| 4 | cmake_install.cmake | 1727 | 17270 |
| 5 | dfw2gufitest | 6501 | 65010 |
| 6 | gitest.py | 20028 | 200280 |
| 7 | gufitest.py | 56232 | 562320 |
| 8 | outq.0 | 719 | 7190 |
| 9 | outq.1 | 1106 | 11060 |
| 10 | robinhoodin | 95 | 950 |
| 11 | runbffuse | 1210 | 12100 |

# Are their ways to interface to GUFI (DuckDB)

```
D select * from read_csv($$ ./gufi_query -n1 -d "," -E "select name,size from vrpentries where name like '%f1%%';" testidx |$$, names=['name','size']) order
by size desc;
```

| name varchar | size int64 |
|---|---|
| f1.biggest | 10 |
| f1.bigger | 7 |
| f1.big | 4 |
| f1.1.doc | 2 |
| f1.xls | 0 |
| f1.xls | 0 |
| f1.xls | 0 |
| f1.exe | 0 |
| f1.tar | 0 |
| f1.exe | 0 |
| f1.tar | 0 |
| f1.exe | 0 |
| f1.tar | 0 |

13 rows 2 columns

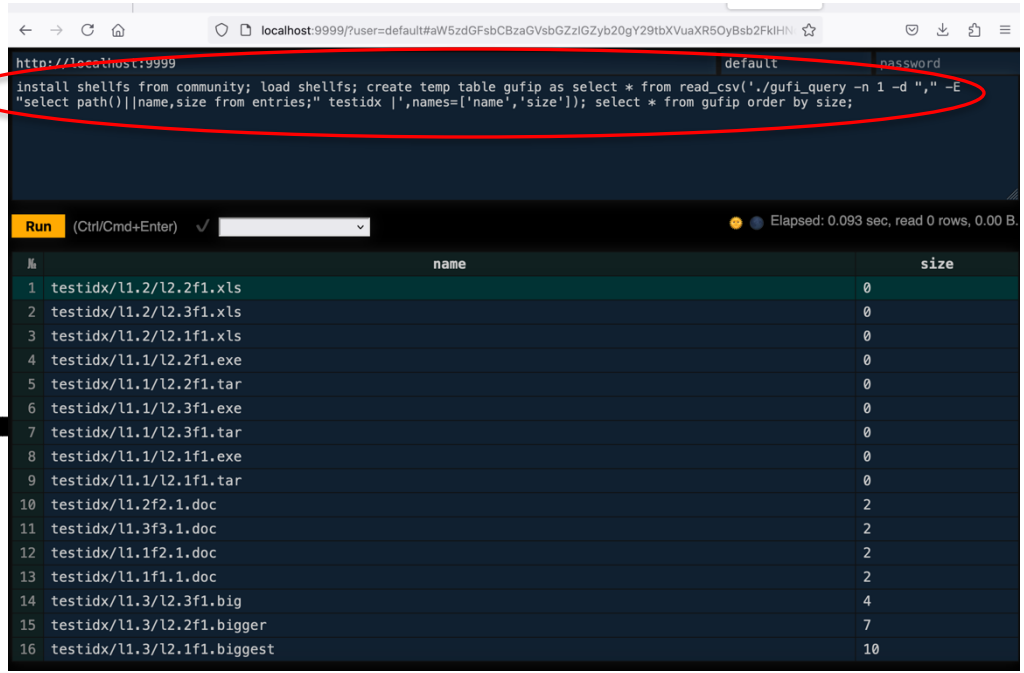
Gufi can appear as a table to DuckDB and fit into that ecosystem
And duckdb can “finish” query (final sort/group)

Here is a quick example, fire up the http duckdb server

```
D install httpserver from community;
100% ▕████████████████████████████████████████▏
D load httpserver;
D select httpserve_start('localhost',9999,'');
```

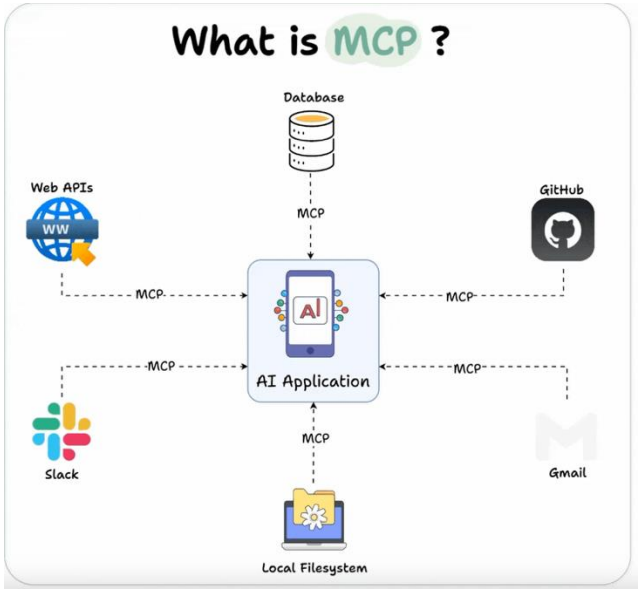
| httpserve_start('localhost', 9999, '') varchar |
|---|
| HTTP server started on localhost:9999 |

And on my browser pointed at localhost:9999




Notice DuckDB can use $$ for a quote

# Are their ways to interface to GUFI (MCP)




**Because GUFI or any GUFI query can look like an Sqlite table. -**

**GUFI is a need-to-know MCP access method for LLM and Agents to all our data holdings**