Clone repository and compiling code

```
# Clone repo
git clone https://github.com/mar-file-system/GUFI.git
# make build directory and change into that directory
mkdir build
cd build
# compiling code
cmake .. -DCMAKE_INSTALL_PREFIX=~/.local/gufi
make -j
sudo make install
# Export to PATH (you can also add it to .bashrc)
export PATH="$PATH:~/.local/gufi/bin/"
```

Build GUFI Index

```
gufi_dir2index <input dir> <output dir> -n <thread count>
```

Some Tables in GUFI Index

- `entries`: metadata of the files in the current directory
- `summary`: summary of current directory
- `treesummary`: summary of current and all the subdirectories

How does aggregation work?

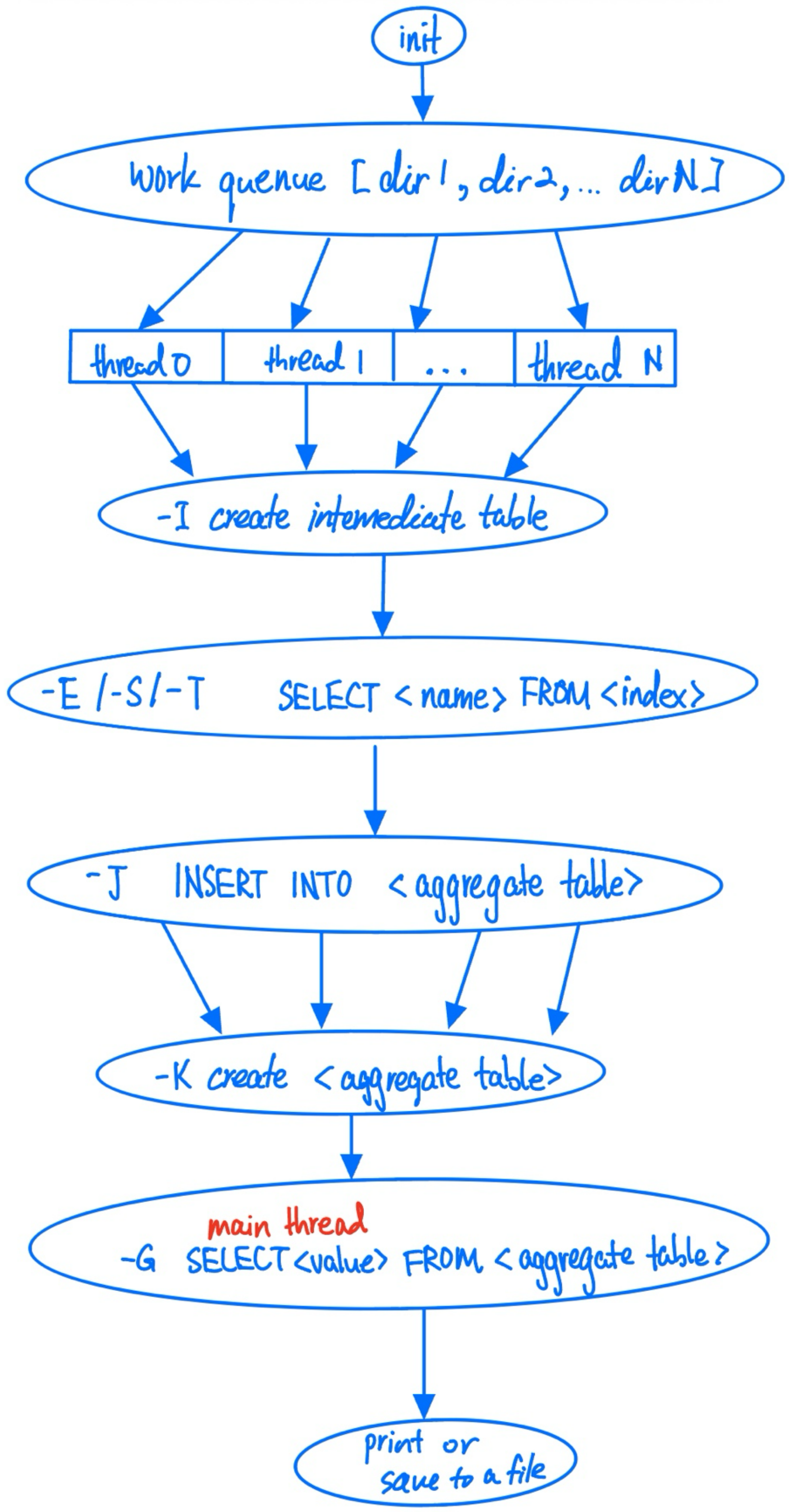
How does work assignment in (generic) thread pools work?

- `-I`: Create the intermediate table
- `-E` / `-S`/`-T`: Each thread executes SQL (e.g., SELECT from entries/summary) and writes to `intermediate`
- `-K`: Create the final aggregate table
- `-J`: The main thread aggregates rows from `intermediate` into `aggregate`
- `-G`: The final SELECT from `aggregate` to produce user output

How does this affect `-S`/`-E` result placement?

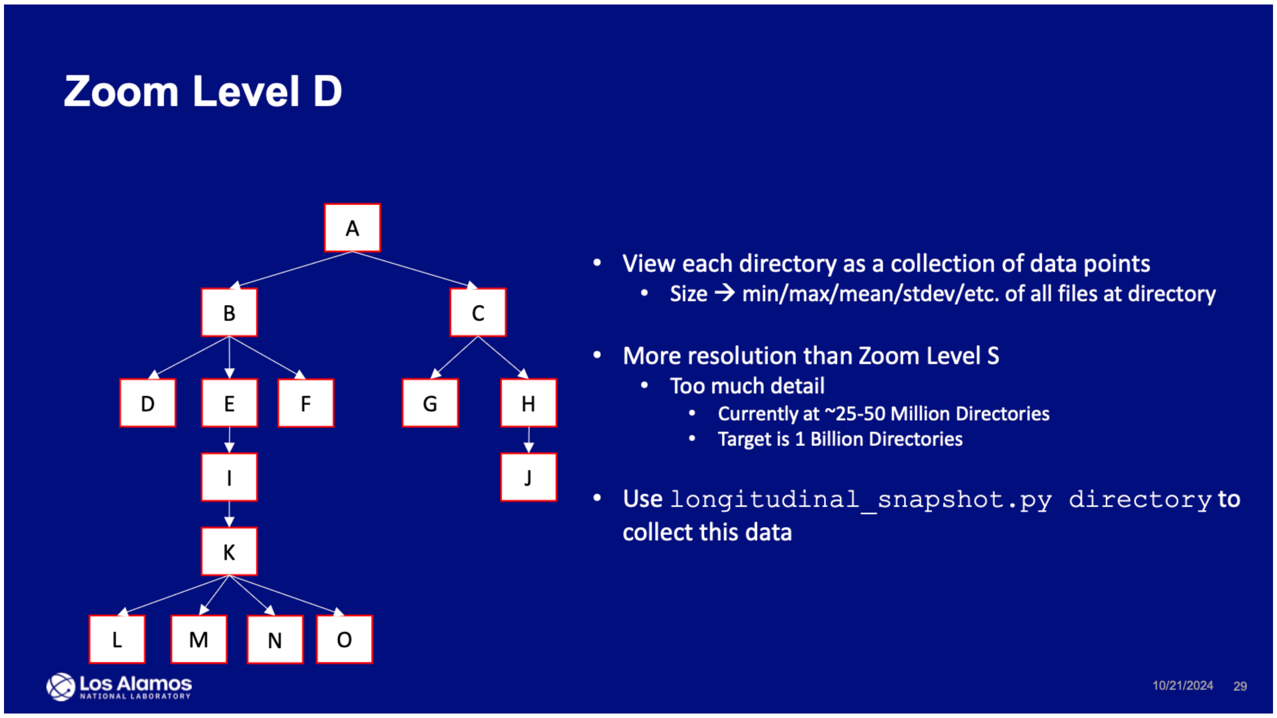
- Each thread runs its own of the `-S` / `-E` query on the directories it processes.
- Results are written into per-thread intermediate table in memory or file, one per thread.
- The data placement depends on which thread processes which directory.
- If using `-o` or `-O`, each thread writes its output to separate files:
    - e.g., `outfile.0`, `outfile.1`, etc.

Workflow


init
work quenue [dir1, dir2, ... dirN]
thread 0
thread 1
...
thread N
-I create intermediate table
-E /-S/-T SELECT <name> FROM <index>
-J INSERT INTO <aggregate table>
-K create <aggregate table>
main thread
-G SELECT <value> FROM <aggregate table>
print or save to a file

`gufi_query` Examples

Running command at Zoom Level D (each directory)



Zoom Level D

- View each directory as a collection of data points
  - Size → min/max/mean/stdev/etc. of all files at directory
- More resolution than Zoom Level S
  - Too much detail
    - Currently at ~25-50 Million Directories
    - Target is 1 Billion Directories
- Use `longitudinal_snapshot.py directory` to collect this data

- `-E`: for `*entries` table, such as `pentries`, `entries`
- `-S`: for `*summary` table, such as `summary`, `vrsummary`
- `-T`: for `treesummary` table

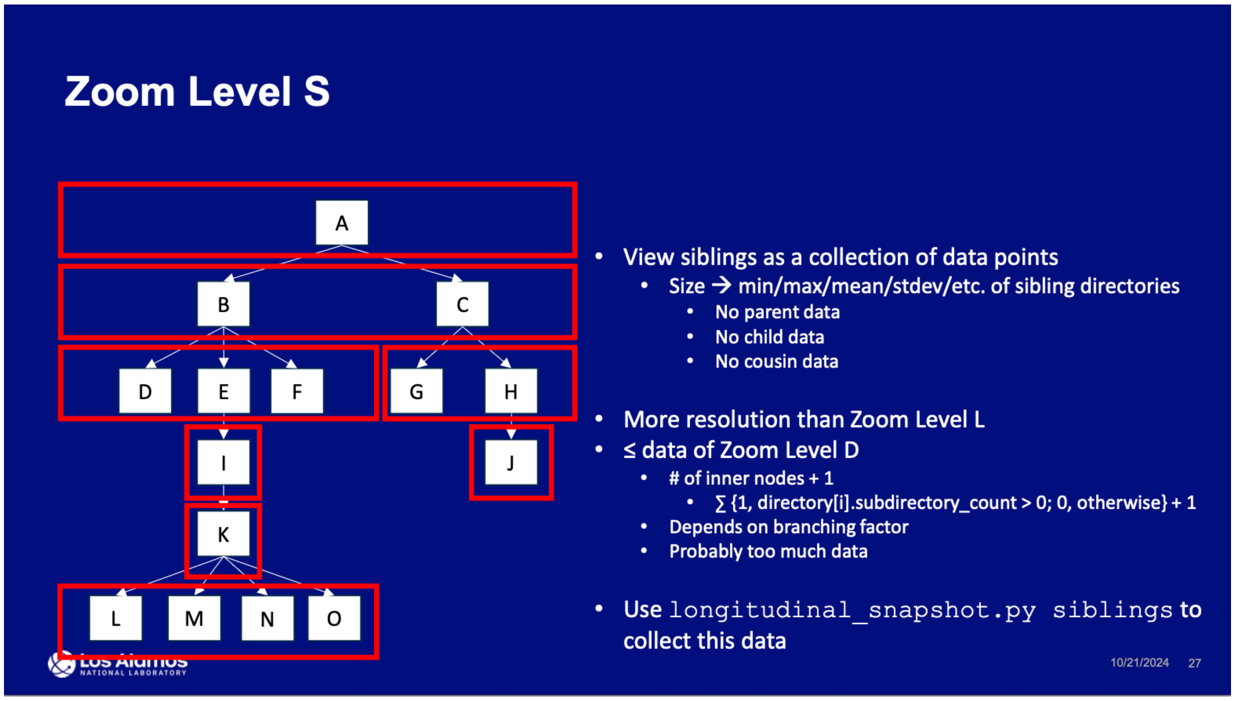
```
# Get size from entries table (32 threads) -- size of each file
in the index
gufi_query <index> -E "SELECT size FROM entries" -n 32

# Get size from summary table (32 threads) -- size of each
directory in the index
gufi_query <index> -S "SELECT size FROM summary" -n 32

# To access treesummary, first needs to create treesummary table
gufi_treesummary_all <index> -n 32

# Get size from treesummary table (32 threads) -- size of each
directory + all subdirectory in the index
# in the example image, if we are getting size at directory B,
it will also includes all the size below (D, E, F, I, K, L, M,
N, O)
gufi_query <index> -S "SELECT size FROM treesummary" -n 32
```

Running command at Zoom Level S (siblings)


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
Los Alamos NATIONAL LABORATORY

Zoom Level S

- View siblings as a collection of data points
  - Size → min/max/mean/stdev/etc. of sibling directories
    - No parent data
    - No child data
    - No cousin data
- More resolution than Zoom Level L
- ≤ data of Zoom Level D
  - # of inner nodes + 1
    - ∑ {1, directory[i].subdirectory_count > 0; 0, otherwise} + 1
  - Depends on branching factor
  - Probably too much data
- Use `longitudinal_snapshot.py siblings` to collect this data

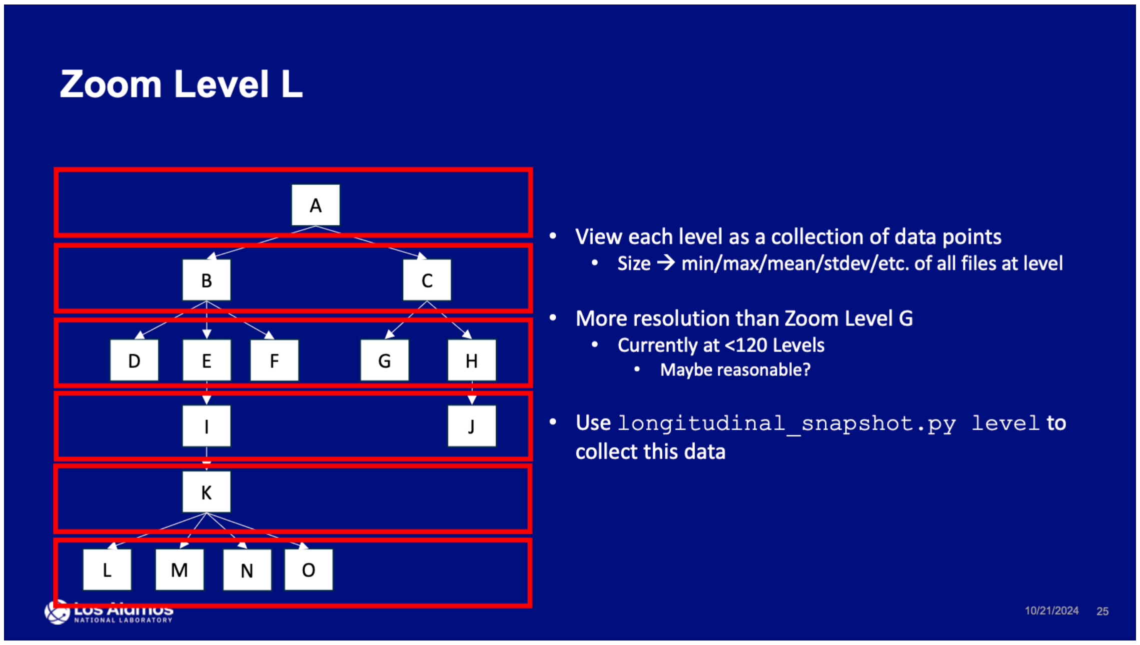
10/21/2024 27

```
# Get size from pentries table (32 threads)

# -I creates an intermediate table to store per thread level,
ppinode (to get the file from same direcory) and sum of size for
-E
# -E gets file's level, ppinode and sum of size from the entries
table insert into intermediate table
# -K creates an aggregate table to combine per thread level,
ppinode and size into one dataset
# -J gets level and sum of size (of matching levels and ppinode)
from intermediate table per thread combine into one data store
into aggregate table
# at this point, there are level, ppinode and sum of size from
each thread potentially with repeated levels and ppinode
originating from different thread
# -G groups them to remove duplicate ppinode and sum to get down
to one size and print
# -d is the 1-character separate for each column
```

```
gufi_query <index> \
-I "CREATE TABLE intermediate(level INT64, ppinode TEXT,
total_size INT64);" \
-E "INSERT INTO intermediate SELECT level() AS level, ppinode,
SUM(size) FROM pentries WHERE type = 'f' GROUP BY level(),
ppinode;" \
-K "CREATE TABLE aggregate(level INT64, ppinode TEXT, total_size
INT64);" \
-J "INSERT INTO aggregate SELECT level, ppinode, SUM(total_size)
FROM intermediate GROUP BY level, ppinode;" \
-G "SELECT level, ppinode, SUM(total_size) FROM aggregate GROUP
BY level, ppinode;" \
-d '|' -n 32
```

Running command at Zoom Level L (level)

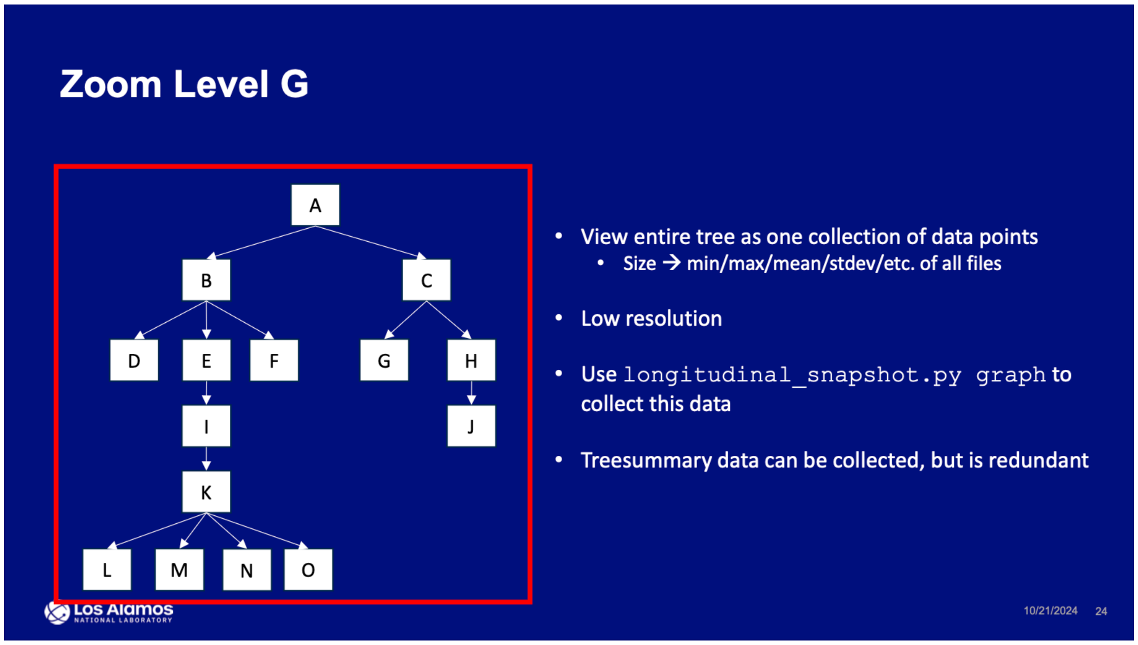


Zoom Level L

- View each level as a collection of data points
  - Size → min/max/mean/stdev/etc. of all files at level
- More resolution than Zoom Level G
  - Currently at <120 Levels
    - Maybe reasonable?
- Use `longitudinal_snapshot.py level` to collect this data

```
# -I creates an intermediate table to store per thread level and
sum of size for -E
# -E gets file's level and sum of size from the entries table
insert into intermediate table
# -K creates an aggregate table to combine per thread level and
size into one dataset
# -J gets level and sum of size (of matching levels) from
intermediate table per thread combine into one data store into
aggregate table
# at this point, there are level and sum of size from each
thread potentially with repeated levels originating from
different thread
# -G groups them to remove duplicate levels and sum to get down
to one size and print
# -d is the 1-character separate for each column
gufi_query \
-I "CREATE TABLE intermediate(level INT64, total_size INT64);" \
-E "INSERT INTO intermediate SELECT level() AS level, SUM(size)
FROM entries WHERE type = 'f';" \
-K "CREATE TABLE aggregate(level INT64, total_size INT64);" \
-J "INSERT INTO aggregate SELECT level, SUM(total_size) FROM
intermediate GROUP BY level;" \
-G "SELECT level, SUM(total_size) FROM aggregate GROUP BY
level;" \
-d ' ' -n 32 <index>
```

Running command at Zoom Level G (global)



Zoom Level G

- View entire tree as one collection of data points
  - Size → min/max/mean/stdev/etc. of all files
- Low resolution
- Use `longitudinal_snapshot.py graph` to collect this data
- Treesummary data can be collected, but is redundant

Los Alamos NATIONAL LABORATORY

10/21/2024 24

```
# -I creates an intermediate table to store per thread data for
-E
# -E gets file's size from the entries table insert into
intermediate table
# -K creates an aggregate table to combine per thread size into
one dataset
# -J gets sum of size from intermediate table per thread combine
into one data store into aggregate table
# at this point, there is one sum of size in the aggregate table
# -G get size and print
# -d is the 1-character separate for each column
gufi_query \
-I "CREATE TABLE intermediate(size INT64);" \
-E "INSERT INTO intermediate SELECT size FROM entries WHERE type
= 'f';" \
-K "CREATE TABLE aggregate(total INT64);" \
-J "INSERT INTO aggregate SELECT SUM(size) FROM intermediate;" \
-G "SELECT total FROM aggregate;" \
-d '|' -n 32 <index>
```