Hewlett Packard
Enterprise

AUTO-TIERING IN CLUSTERSTOR

2024-05-02

TOPICS COVERED

- Feature Overview – what and why
- Architectural Framework – components, interactions
- Practical Usage – what users see, how they interact
- Roadmap – where are we going

FEATURE DRIVERS

Managing Lustre filesystems at scale is challenging	• Lustre commands not easily usable by non-expert end users • Must automate repetitive tasks
Customers typically have to supplement Lustre	• Other open-source tools • Lots of scripting
Hybrid storage adds an element of management complexity	• Flash and HDD are separate pools requiring a-priori placement control
What the market wants	• Optimize storage performance vs cost • Automation based upon unique policies and schedule • Require less end-user Lustre expertise

WHAT IS CLUSTERSTOR AUTO-TIERING?

Available starting in ClusterStor Software Release 6.5

- Cost Optimized
- Simple to install and manage

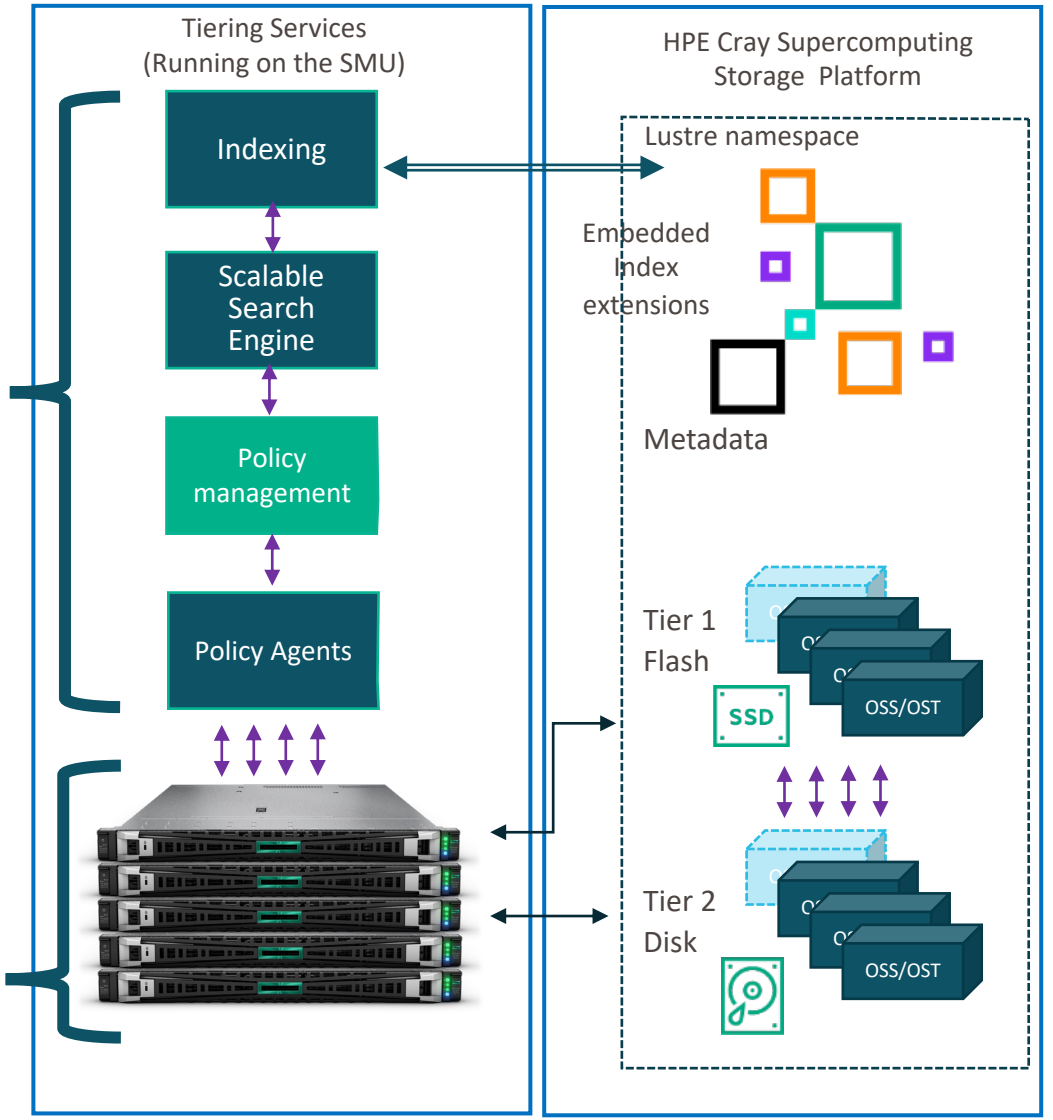
"Within the file system" tiering and file search index for active data

Enables I/O acceleration for HPC jobs

- Purpose-built for Lustre
- Embedded FS Index
- Optimized search
- Customizable policies
- Data movement via policy
- Optional scale-out data movers

- **HPE Cray Supercomputing Storage Auto-Tiering software runs on the SMU**
- Data Mover function is part of the SMU
  - **A base product offering**
  - Up to 20 GB/s performance for data migration
- Up to 5 additional Utility Nodes
  - Scale indexing and policy runs
  - Additional +20 GB/s data movement each

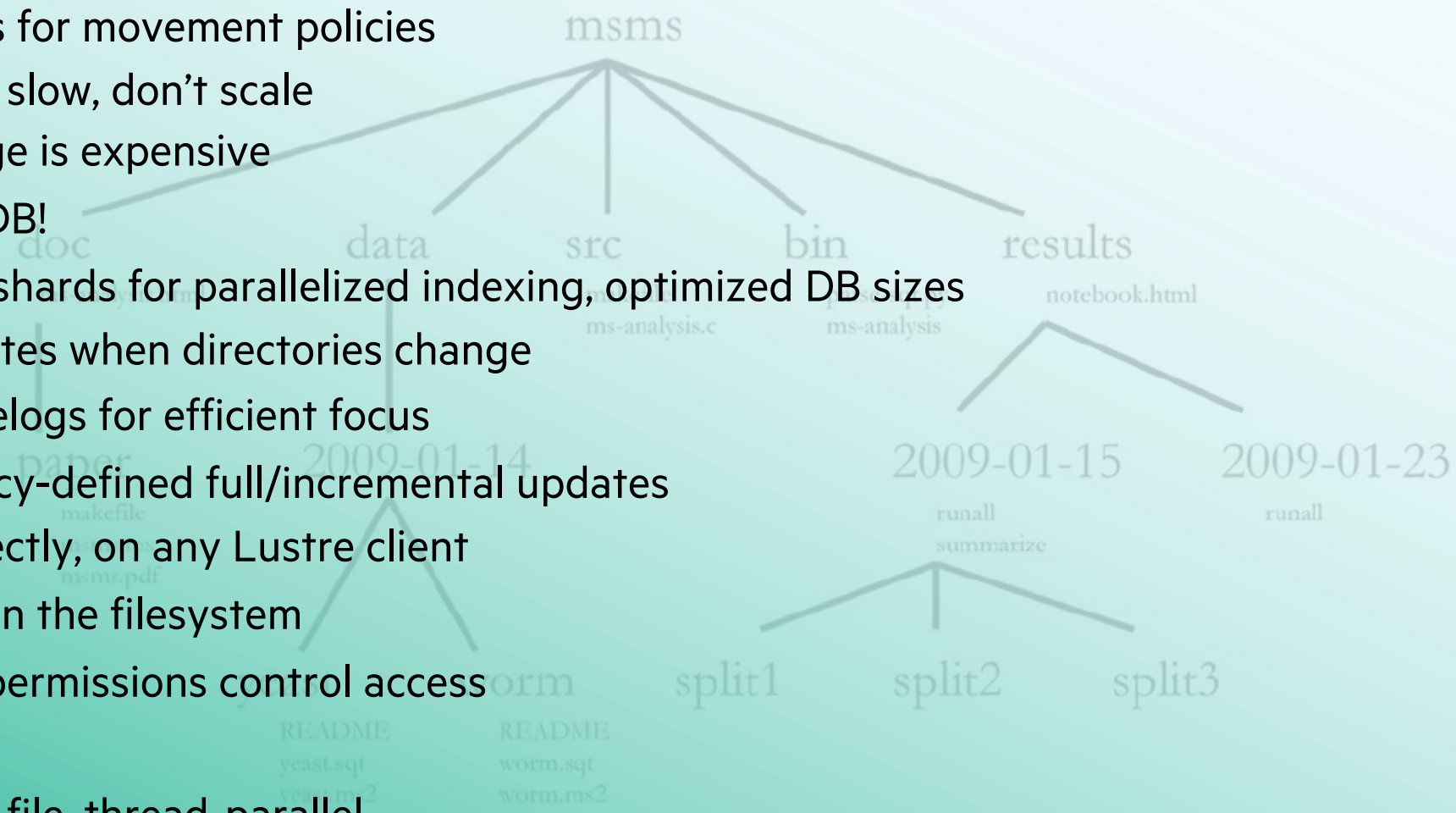
Scaling is Dependent on system configuration and number of flash units



INDEXER AND QUERY

Searchable, scalable metadata database

- Need to efficiently find files for movement policies
- Find / lfs find are much too slow, don't scale
- External scalable DB storage is expensive
  - Use Lustre to store DB!
  - Distribute database shards for parallelized indexing, optimized DB sizes
- Incremental database updates when directories change
  - Utilize Lustre changelogs for efficient focus
  - And/or periodic, policy-defined full/incremental updates
- Query can be executed directly, on any Lustre client
  - We provide an RPM in the filesystem
  - Database shard file permissions control access
  - CSV or JSON output
  - Can operate on each file, thread-parallel



QUERY TOOL

Run Query on any Lustre client to find files matching specific criteria

- All files in Flash pool

```
query --json -C fullpath,uid,gid,sizemib -w "poolname='flash'"  /lus
```

- Files less than 1MB in disk pool

```
query -C relativepath,size,atime,poolname -w "poolname='disk'" -w "size < 1048576" --limit 100 /lus
```

- Files in flash pool in directory subtree ‘moonshot’ that are over 10 days old

```
query --json -C fullpath,sizemb,mtime -w "mtime < $(($(date +%s)-24*60*60*10))000000000"
/lus/projects/moonshot/
```

- Files starting with ‘ftzz’ on OST1

```
query -C relativepath,uid,size,ostindices -w "name LIKE 'ftzz%’” –w "ostindices = ':1:’”   /lus
```

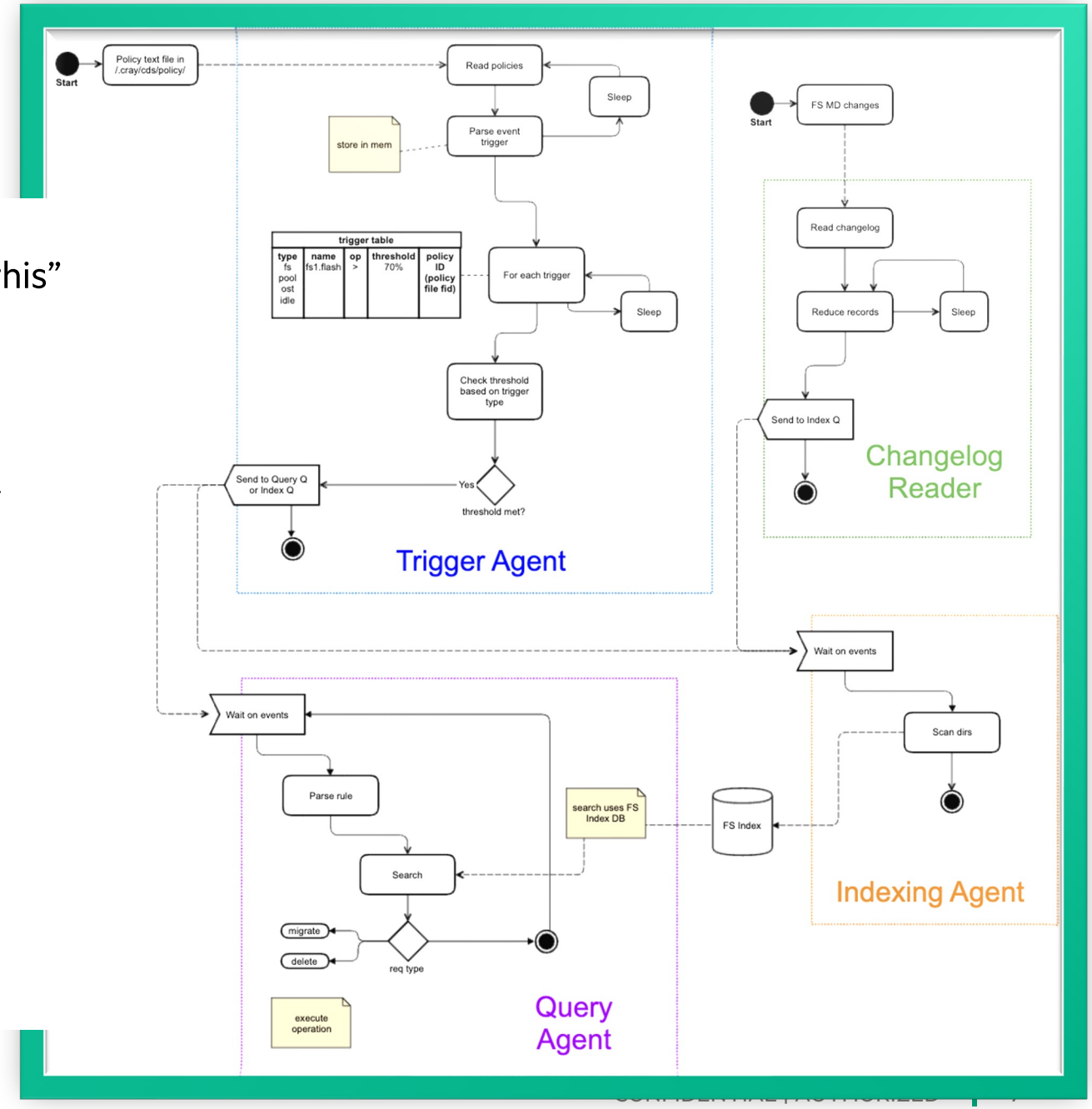
- Files on OST 2 or 3 but not in flash pool

```
query --json -C fullpath,poolname,ostindices -w "poolname NOT LIKE '%flash%'" -w "ostindices LIKE '%:2:%'
OR ostindices LIKE '%:3:%'" --limit 2 /lus/users
{"fullpath":"/lus/users/sean/indexing_test/tesdir17027/tesfile0", "ostindices":":2:", "poolname":"none"}
{"fullpath":"/lus/users/sean/indexing_test/tesdir17027/tesfile1", "ostindices":":3:", "poolname":"none"}
```

POLICY ENGINE

Automation and query scale-out

- Policies
  - Describe *actions to take* ➪ ”When that happens, do this”
  - Can be defined by Admins or Users
  - Rich Robinhood-like definitions
  - Also stored in the filesystem
- Trigger Agent
  - Monitors filesystem for triggering events (eg capacity threshold)
  - Flexible trigger conditions
  - Partitions search space amongst query agents
- Query Agent
  - Finds files that meet policy criteria
  - Operates (action) on each file
- Indexing Agent
  - Updates database indexes
  - Based on changelogs
  - And/or directory mtime changes
  - And Query’s verification failures



POLICY DIRECTIVES

Actions to take upon triggering

- When invoked directly, Query tool lists matching files (but see --exec and --delete)
- Enhanced behavior when invoked via a policy:
  - Parallelized, scale-out operation on fileset partitions
  - Simplified invocations for migrate, report generation
- Indexing directives are sent to parallel Indexing Agents

Directive Name	Description
Migrate	Migrate files from one layout (eg OST pool) to another
Delete	Delete files from namespace based on criteria
Report	Create CSV or JSON reports of filesets
Exec	Perform administrator-defined functions on files
Incremental_index	Update a filesystem subtree based on changes in directory mtime
Full_index	Update all subdirectories in a filesystem subtree

SOME USES

- Flash tier hygiene
- Scrub old files
- Regular usage reports
- Rebalance OSTs
- Drain a failing OST
- Change owner of *.data files in projtree
- Restripe large files into multiple OSTs
- Fun DB things
  - Top 10 directories by file count
  - Top 10 largest files in flash
  - List recently created, single-striped files > 1 TB
  - File size histogram

SETUP

1. Enable on ClusterStor management node
```
> cscli lustre tiering enable
```

2. Configure changelogs if desired

3. Store policies in `<lus>/.cray/cds/policies/*.plc`

4. Profit!

- Policies must be stored in a designated directory
- The files must have a `.plc` extension
- Multiple policies can be created
- Administrators control access
  - Allow or disallow others to create policies
  - Policies run under the UID of the policy file owner

Execute queries on any node – we provide RPM:
```
> rpm -i --nodigest <lus>/.cray/cds/tools/*/cds-brindexer-tools-*.rpm
> /opt/cray/brindexer/bin/query -h
```

USE CASES

- **Tier maintenance**
- Purge old files
- Reports
- Exec

```
# cat /mnt/lustre/.cray/cds/policies/mygrate.plc
fileclass largeflash {
    definition { size > 100MB and pool = flash }
}
flash_migrate_rules {
    rule migrate_large {
        target_fileclass = largeflash;
        action = migrate;
        action_params {
           migrate_pool = disk;
        }
        condition { last_modified > 2d }
    }
}
flash_migrate_trigger {
     check_interval = 600;
     trigger_on = pool_usage(flash);
     high_threshold_pct = 75%;
}
define_policy flash_migrate {}
```

- All files striped in the ‘flash’ pool and larger than 100MB will be migrated to the ‘disk’ pool
- Verify the condition that the last modified time for each file is more than 2 days old (file stat just before migrating)
- Triggered if the space used in the flash pool is > 75%, check every 10 mins

USE CASES

- Tier maintenance
- **Purge old files**
- **Reports**
- Exec

```
# cat /mnt/lustre/.cray/cds/policies/report_purger.plc
fileclass purge_reports {
    definition { tree = admin/reports and name = purge-* }
}
purge_reports_rules {
    rule purge {
        target_fileclass = purge_reports;
        action = delete;
        action_params {
            # New report on purged purge reports
            report_path = admin/reports/purge-reports
        }
        condition { last_modified > 7d }
    }
}
purge_reports_trigger {
     trigger_on = schedule("5 3 * * *");
     partition_count = 1;
}
define_policy purge_reports {}
```

- Delete files under `<mnt>/admin/reports/` tree named `purge-*`
- Report the deleted files as `admin/reports/purge-reports/<policyinfo>-<date>`
- Run daily at 3:05 UTC

USE CASES

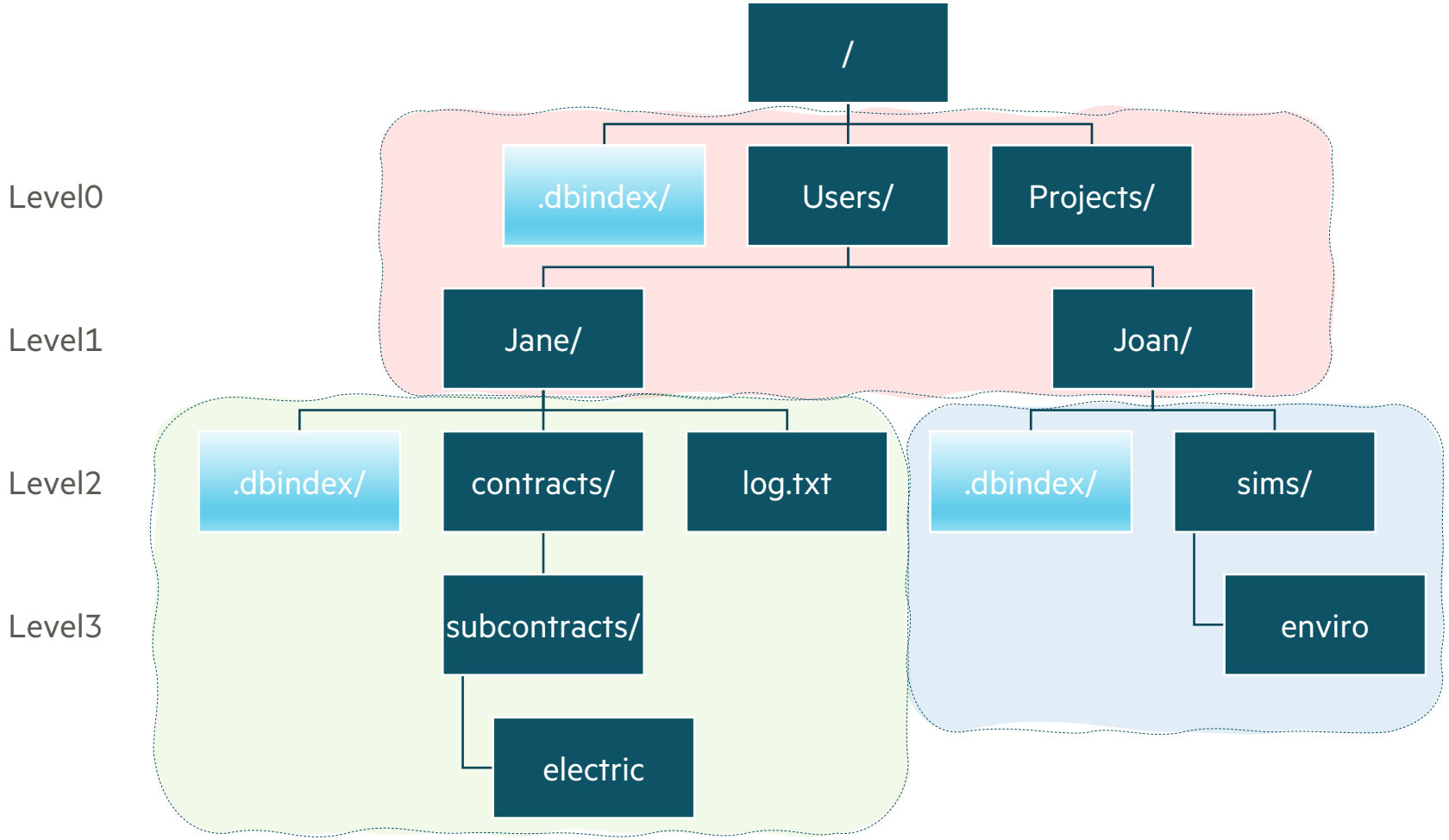
- Tier maintenance
- Purge old files
- Reports
- **Exec**

```
# cat /mnt/lustre/.cray/cds/policies/checksummer.plc
fileclass files {
    definition { tree = datadir and size >= 1kib }
}
shas_rules {
    rule shareport {
        target_fileclass = files;
        action = exec;
        action_params {
            report_path = reports;
            exec_command = /usr/bin/sha256sum;
        }
        condition { last_modified > 1d and last_access > 1d; }
    }
}
shas_trigger {
    partition_count = 1;
    trigger_on = schedule(once);
}
define_policy shas {}
```

- Calculate sha256 on recent files in `datadir`.
- Store output in per-run files under `reports`
- Run when the policy file is touched.

DATABASE PLACEMENT

- Index shards placed at Level 0 & 2 by default
- Auto-split every 2 levels deeper if needed (>5B, configurable)
- Shard placement can be steered by 'level' file in .dbindex dir
- Shard .dbindex dirs owned by parent



SECURITY

POSIX-based trust is simple, flexible, predictable

- Admin-configurable database access params
- Users must have .dbindex *directory access* to see shards at all
  - Must have access to traverse to the parent dir
  - Must have dirmode access (owner or group member or world) to .dbindex
- Users must *also* have shard read perms to see entries in shard (uid/gid/mode)

Database control	Default (open)	Ex. Restrictive
dbindex_uid	0	cstor
dbindex_gid	0	searchers
dbindex_mode	0644	0640
dbindex_dirmode*	0755	0750

- Parent owner decides which others can see their shards
  - No access to dir = no access to shard
  - Access can be changed per-shard at any point
- Admin decides who can read shard *contents* (file info)
  - Admin only, limited group, or anyone with dir access

* .dbindex dir owner is always parent, so they can delete shard - but not necessarily see contents

Policy file security

- Access to the policy directory
  - Admin sets ugw r/w
- Ownership of policy files
  - Policies run under policy file's uid/gid
  - Can't navigate permissionless dirs
  - Can't operate on permissionless files

PERFORMANCE

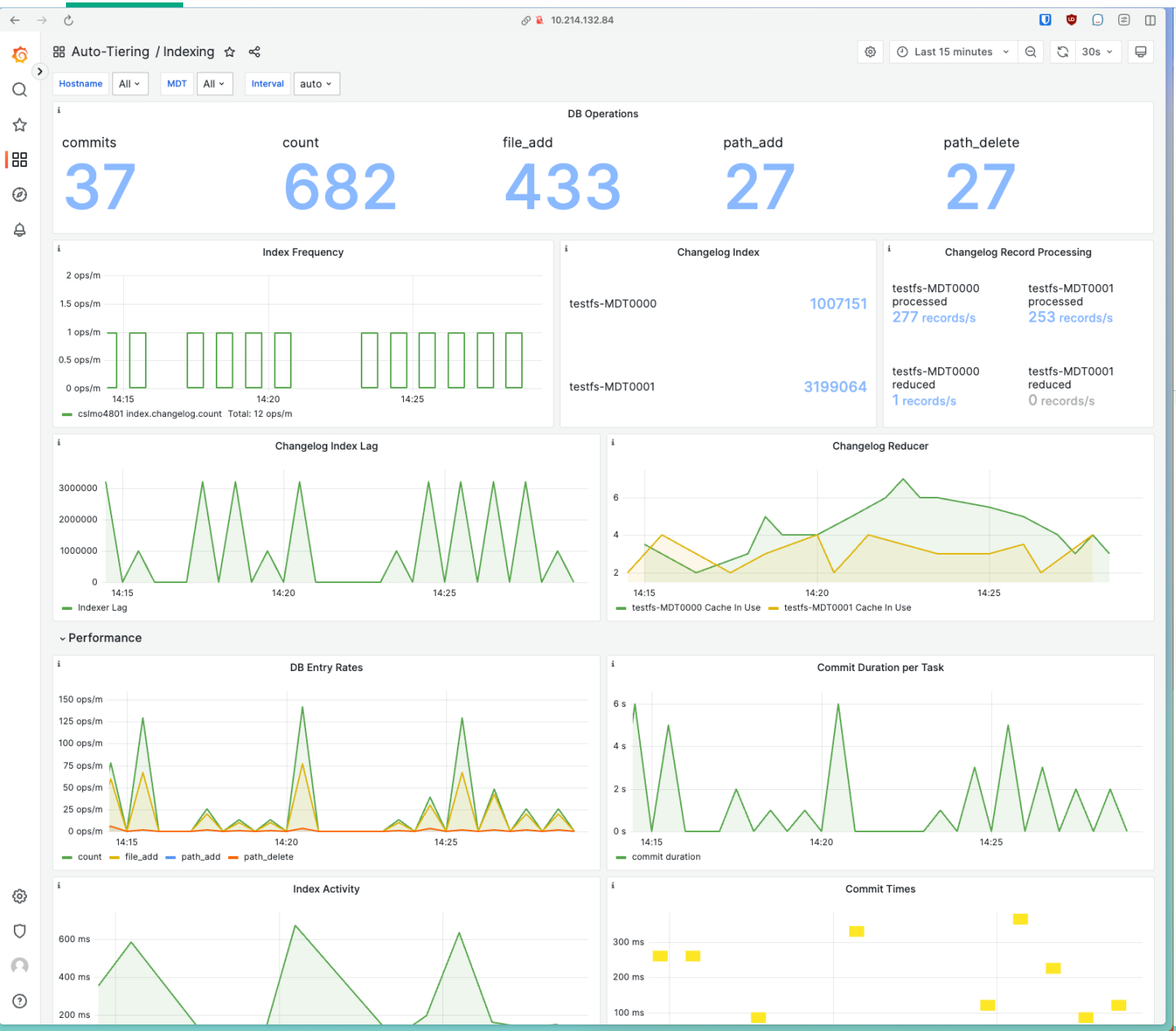
- All operations scale out across SMU + Utility nodes
  - Multiple shards enable parallelized index writers
  - Changelog readers are assigned round-robin to MDTs
  - Filesets are partitioned across Query Agent instances for parallel policy operations
- User queries are confined to deepest shard location containing search tree
  - Eg `query -C fullpath,size -w "size < 1048576" /lus/Users/Jane` will only search shards below `/lus/Users/Jane/`
- User queries run in parallel across shards

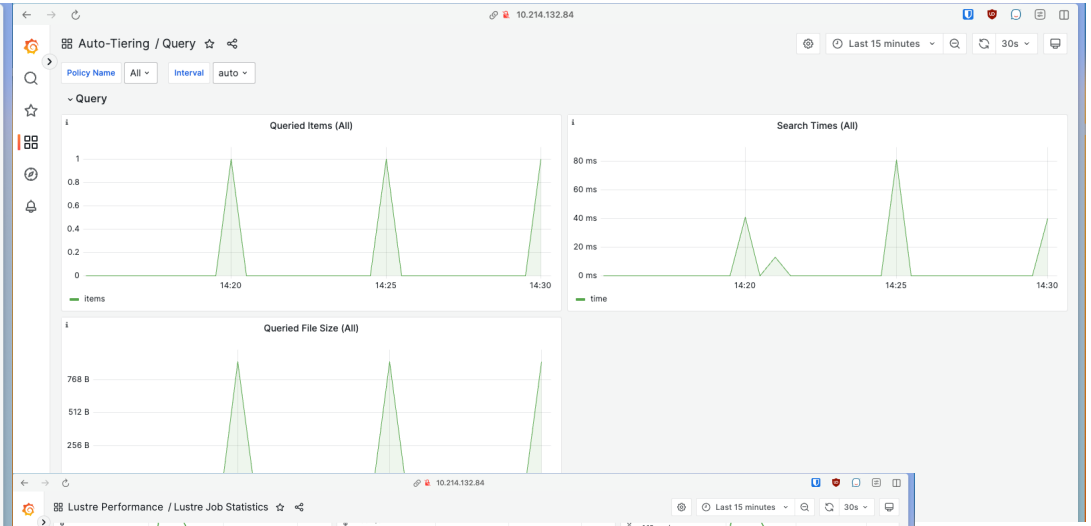
*Performance of course depends on many factors: fabric speed, MDT/OST counts, DB placement, search keys, etc.

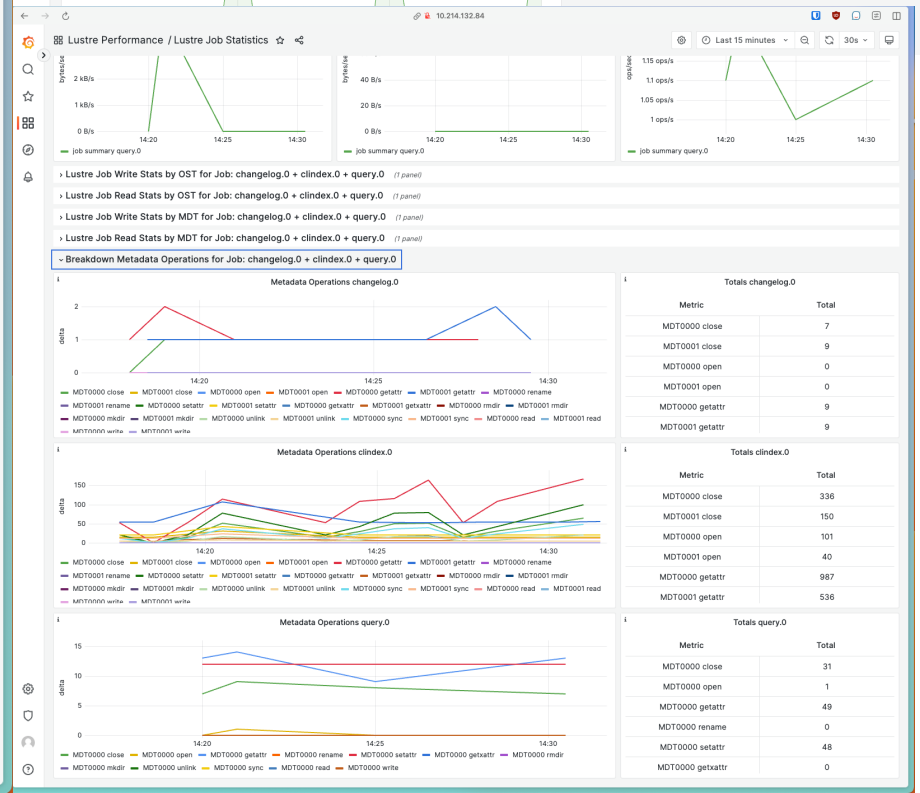
Metric	Per-node value*
Indexing	50,000 files/s
Query	500,000 files/s
Migration	20GB/s (200Gbps HSN)
Migration	3,900 files/s
Delete	44,000 files/s

```
query -v > /dev/null
level=info msg=Records found: 16489473
level=info msg=Result rate (records/sec): 543095
level=info msg=Time elapsed: 30.362s
```

INTEGRATED CLUSTERSTOR MONITORING







ROADMAP

CDS 2.2.2

- Robust fixes for large-scale customers
- No further updates planned

Neo 6.5

- Single-node operations on SMU

Neo 6.6

- Scale-out indexing and query actions, both SMUs plus utility nodes

Future

- Improved indexing performance
- More flexible database placement
- Integrate parallel copy tools

Fin

THANK YOU

Nathan.Rutman@hpe.com

ADDITIONAL RESOURCES

- ClusterStor Admin Guide: https://support.hpe.com/hpesc/public/docDisplay?docId=sd00001354en_us&page=index.html
- Tiering and Scalable Search Use Cases - https://support.hpe.com/hpesc/public/docDisplay?docId=sd00001354en_us&page=GUID-6A832A4B-FD0F-406A-A5D8-71095B317B27.html
- Policy file syntax: https://support.hpe.com/hpesc/public/docDisplay?docId=sd00001354en_us&page=GUID-A9EE2B04-C3DB-4F53-96C2-5CC3940F8CEC.html
- Query examples: https://support.hpe.com/hpesc/public/docDisplay?docId=sd00001354en_us&page=GUID-6969ABB9-BA97-4716-95BC-B0AEFC59A70B.html

SUMMARY TOOL

- Summary tool is run on a client, e.g. secondary management node, and displays bulk statistical information about the entire file system

```
root@kjlmo1301 ~]# summary /run/lustre_tiering/mountpoint/
Total link count:   0
Total dir count:    1191
Total file count:   2258497
Total file size:    143801198627058
Total file objects: 2259688
Maximum file size:  549755813888
Minimum file size:  168
Maximum mtime:      1696013375000000000
Minimum mtime:      1690318619000000000
Maximum ctime:      1696187481000000000
Minimum ctime:      1695064297000000000

Time elapsed:  640ms
```

QUERY POWER USER

Histogram of file sizes

```
> query -v --header -q "select (length(size)-1) AS bin,count(size) as
count,sum(size) as sum from entries_0 group by bin" /lus | awk -F"," 'OFS="," {if
(NR == 1) {print $0} else {groups[$1]; count[$1]+=$2;sum[$1]+=$3}} END {for (grp
in groups) {print "10^"grp, count[grp], sum[grp]}}' | column -s, -t -R 2,3
level=info msg=Records found: 1799
level=info msg=Result rate (records/sec): 26
level=info msg=Time elapsed: 1m7.856s
bin       count           sum
10^0  487622301     467420362
10^1      49435       2689221
10^2     493324     270935066
10^3   68981346  279920926293
10^4      27963     419989129
10^5         49      17777591
10^6         28     117812880
10^7         13     164978712
10^8          1     135266304
10^9        280  300647710720
```

- Roll up file info in each DB shard
- Aggregate shard summaries
- Pretty-print output

➪ **size histogram of half billion files in 68 seconds = 7.4M files/s**

SETTING UP CLUSTERSTOR TIERING AND ADDITIONAL DATA MOVERS

- Tiering is disabled by default. To enable tiering, from the primary management server on ClusterStor System

```
[root@kjlmo1300 ~]# cscli lustre tiering enable
Tiering has been enabled.
```

- We recommend enabling Changelogs for more efficient updates to the index
- Secondary management server runs Auto-Tiering software
- 5 additional utility nodes can be added for additional data migration and purge bandwidth. Managed by ClusterStor, on into the internal management network

```
[root@kjlmo1300 ~]# cscli show_new_nodes
--------------------------------------------------------------------------------
Hostname/MAC                IPMI        Free arrays   Assigned arrays  Pass/Fail  HW profile
--------------------------------------------------------------------------------
* 04:32:01:5A:7A:1C    172.16.0.109        N/A              N/A          Passed  custom node
* 04:32:01:5A:7B:54    172.16.0.110        N/A              N/A          Passed  custom node

[root@kjlmo1300 ~]# cscli configure_hosts -m 04:32:01:5A:7A:1C --hostname kjlmo1308 --location
R1C1/10U --role datamover && cscli configure_hosts -m 04:32:01:5A:7B:54 --hostname kjlmo1309 --
location R1C1/11U --role datamover
```

- Once the above command completes, the two additional data movers are automatically added to the tiering configuration and will be used going forward
- The system now had 3 total data movers: the integrated management node plus 2 external data movers
- Install user tools if desired

```
> rpm -i --nodigest <lustre_mount>/.cray/cds/tools/*/cds-brindexer-tools-*.rpm
```