GWDG
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

FUSE-Based File System Monitoring and I/O Classification with IOFS

Marcus Boden, Lars Quentin, Julian Kunkel

2022-06-02

Outline

Motivation

- Many (new) users have limited knowledge about their I/O patterns
- I/O hard to analyze for users
  - no simple tools for analysis
  - analysis tools may need advanced skills or elevated access
- often treated as a black-box, if at all
- Different file systems have different characteristics
- bad I/O patterns can slow down global file systems

Goals

- give users a tool to monitor their I/O
- easy to use
  - no changes in code necessary
  - no extra libraries/linking
- easy to test and set up
  - dockerfiles for database and Grafana
  - analysis dashboards available as json

FUSE

FUSE (Filesystem in USErspace)

- Userspace interface to create file system
- kernel module and libfuse library
- rewrite file system calls
- examples: sshfs, ntfs-3g

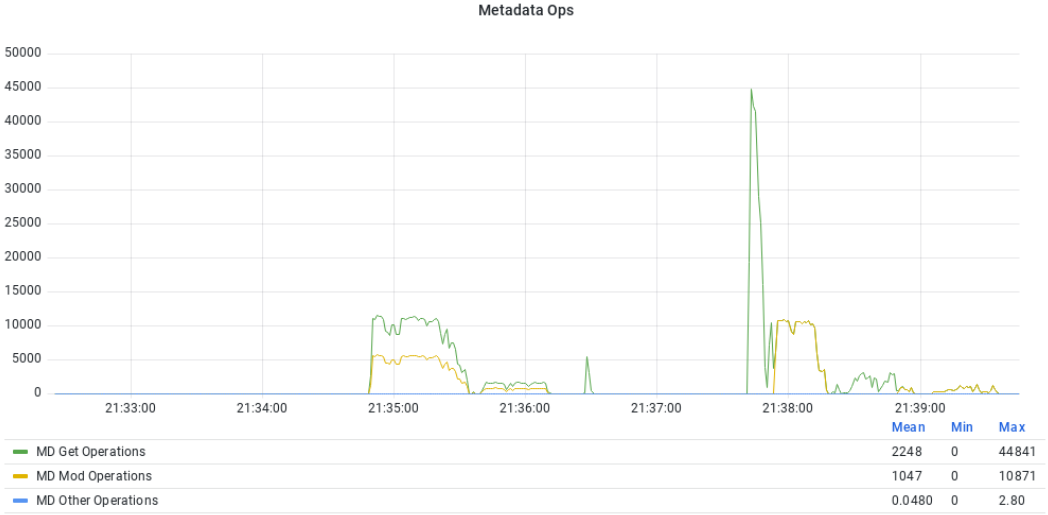
Security considerations:

- setuid binary
- only the user can access the mount
  - can be relaxed with `allow_root` or `allow_other`

Architecture

- FUSE-mount a directory to monitor it
- Intercept all operations on mount
  - start timer
  - change path
  - do operation
  - end timer
  - write timing and size to global struct
- global struct with counter for all operations
- second thread for reporting
  - read counters
  - send to database
  - clear the counters

Results


Metadata Ops
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
21:33:00
21:34:00
21:35:00
21:36:00
21:37:00
21:38:00
21:39:00
Mean Min Max
MD Get Operations 2248 0 44841
MD Mod Operations 1047 0 10871
MD Other Operations 0.0480 0 2.80

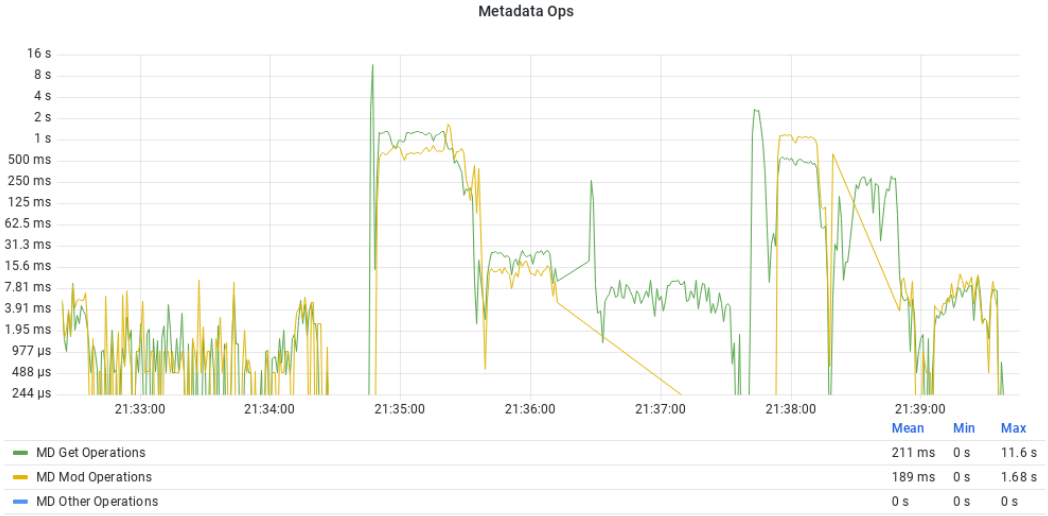
Results


Reads and Writes
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
30000
25000
20000
15000
10000
5000
0
21:33:00
21:34:00
21:35:00
21:36:00
21:37:00
21:38:00
21:39:00
Mean Min Max
reads 804 0 9814
write (right y-axis) 796 0 27180

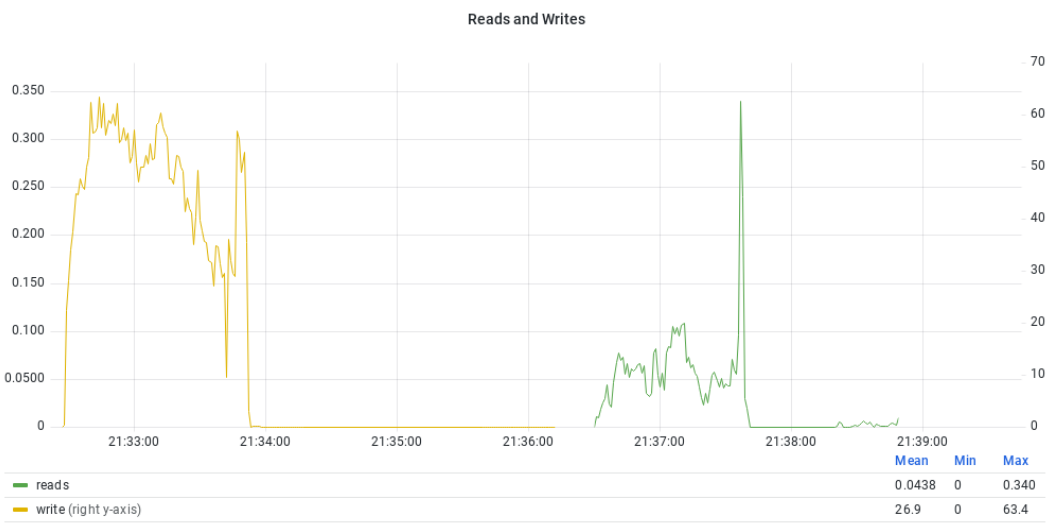
Results

	Mean	Min	Max
Small Reads	16.8	0	516
Large Reads	188	0	2453
Small Writes (right y-axis)	16.8	0	287
Large Writes (right y-axis)	93.3	0	3399

Results


Metadata Ops
16 s
8 s
4 s
2 s
1 s
500 ms
250 ms
125 ms
62.5 ms
31.3 ms
15.6 ms
7.81 ms
3.91 ms
1.95 ms
977 µs
488 µs
244 µs
21:33:00
21:34:00
21:35:00
21:36:00
21:37:00
21:38:00
21:39:00
Mean Min Max
MD Get Operations 211 ms 0 s 11.6 s
MD Mod Operations 189 ms 0 s 1.68 s
MD Other Operations 0 s 0 s 0 s

Results


Reads and Writes
0.350
0.300
0.250
0.200
0.150
0.100
0.0500
0
70
60
50
40
30
20
10
0
21:33:00
21:34:00
21:35:00
21:36:00
21:37:00
21:38:00
21:39:00
Mean Min Max
reads 0.0438 0 0.340
write (right y-axis) 26.9 0 63.4

Performance impact

- IO500 tests with SCC rules
- 10 Nodes, 80 tasks
- Result: 50%-60% of native performance

I/O Classification

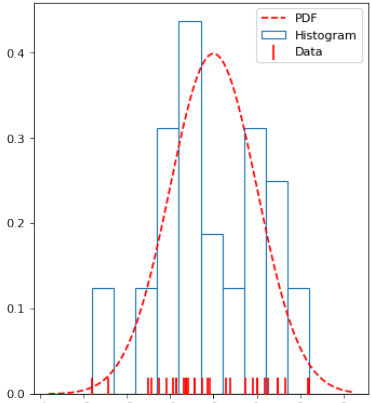
- Idea: Create models for different I/O Operations
- Label accesses by mapping access sizes to characteristics
- Workflow:
  1. Create Benchmarks isolating different characteristics
  2. Analyze each Benchmark; find possible clusters
  3. Create regression models from analyzed benchmarks
  4. Classify new I/O operations via regression models

Creating Benchmarks

- Goal: Find expected durations for different types of I/O requests
- Different benchmark variations to isolate access features
  - File access patterns
  - Access size of I/O operation
  - bypassing cache via `O_DIRECT`
  - Prereading every block
  - Dropping cache after every benchmark
  - Deleting the benchmarked file afterwards

Analyzing a Single Benchmark

- Goal: To find significant clusters
- Unknown underlying distribution
  - Especially no normal distribution
    - Maximum Likelyhood doesn't work!
- Histograms don't work
  - Can hide a lot of information
  - Bucket size selection is non-trivial
  - Long tail is important
- Solution: Kernel density estimations!



Misleading histogram

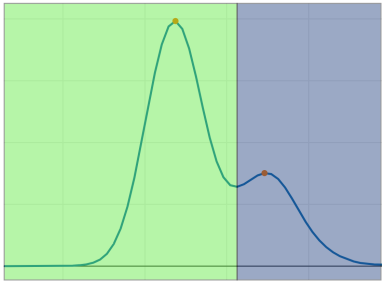
Kernel Density Estimation

We approximate the PDF f with the KDE

$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i)$$

with K being the gaussian normal distribution.

- $x - x_i$ to map x_i onto the kernel K
- $1/N$ normalizes to $\int \hat{f}(x)dx = 1$
- Visually:
  - We put a normal distribution at each observation



Analyzed Benchmark, Clusters Coloured

Creating Models from Benchmarks

- Durations increase linearly in access size $\Rightarrow$ linear interpolation!
- Interpolate over all analyzed benchmarks
  - Points defined as `(access_size, access_time)`
- Compare each new observation with computed models
  - Classify new operations into our benchmark categories

ToDo

- per file accounting
- Slurm integration
- DB-agnostic reporting
- take file system load into account
- integrate modeling and classification to IOFS
- make recommendations based on classifications

Questions

Thanks for your attention

Questions? Feedback? Ideas?
The source code can be obtained at:
https://github.com/gwdg/iofs/
https://github.com/lquenti/blackheap/