Non-Intrusive Monitoring and I/O Classification with IOFS

Christian Boehme, Lars Quentin, Julian Kunkel

2022-15-11

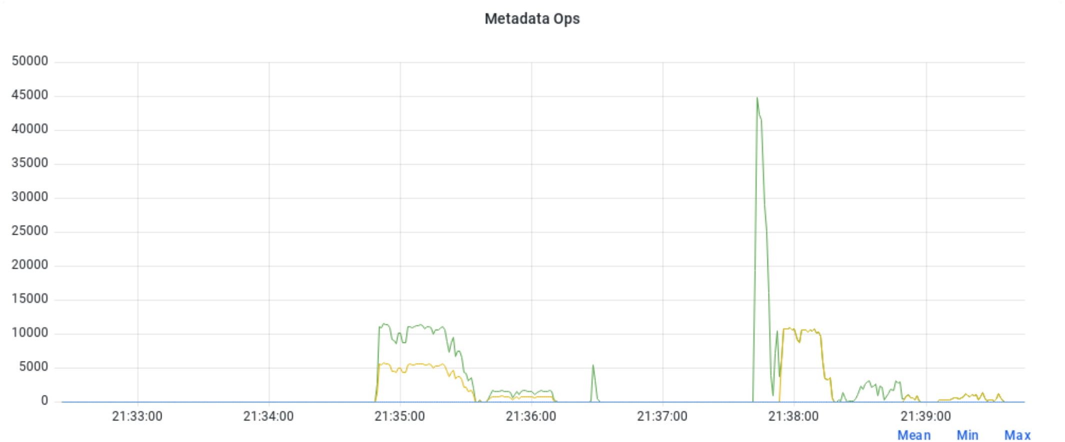
Goals

1. Give users a tool to monitor their I/O (IOFS)
   - Easy to use
     - no changes in code necessary
     - no extra libraries/linking
   - Easy to test and set up
     - just re-route mountpoint
       e.g. /monitor/work instead of /work
2. Support assessment of performance (Blackheap prototype)
3. Integrate assessment into monitoring (work in progress!)

Architecture

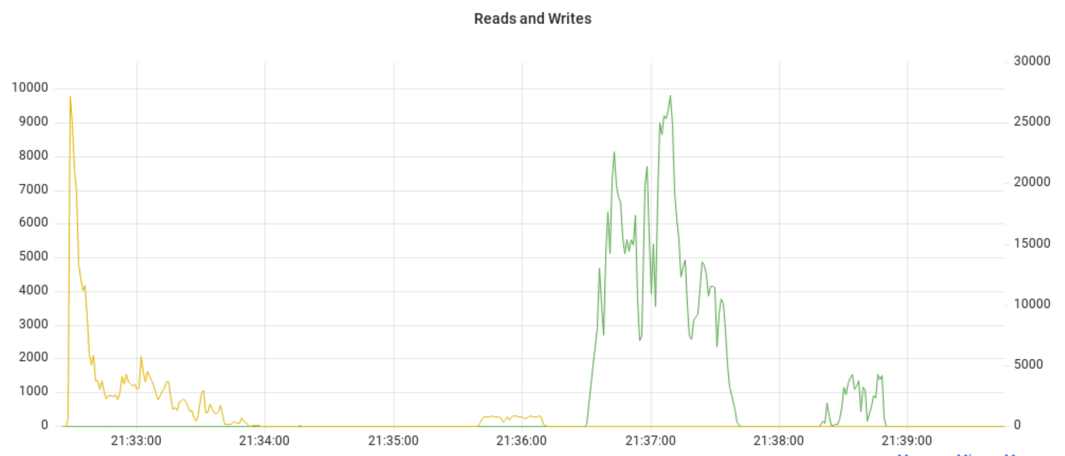
- FUSE-mount a directory to monitor it (can be done globally)
- Intercept all operations on mount
  - start timer
  - change path
  - do operation
  - end timer
  - write timing and size to global struct
- Global struct with counter for all operations
- Second thread for reporting
  - read counters
  - send to database
  - clear the counters
- Basically support for reporting per "file"

Results



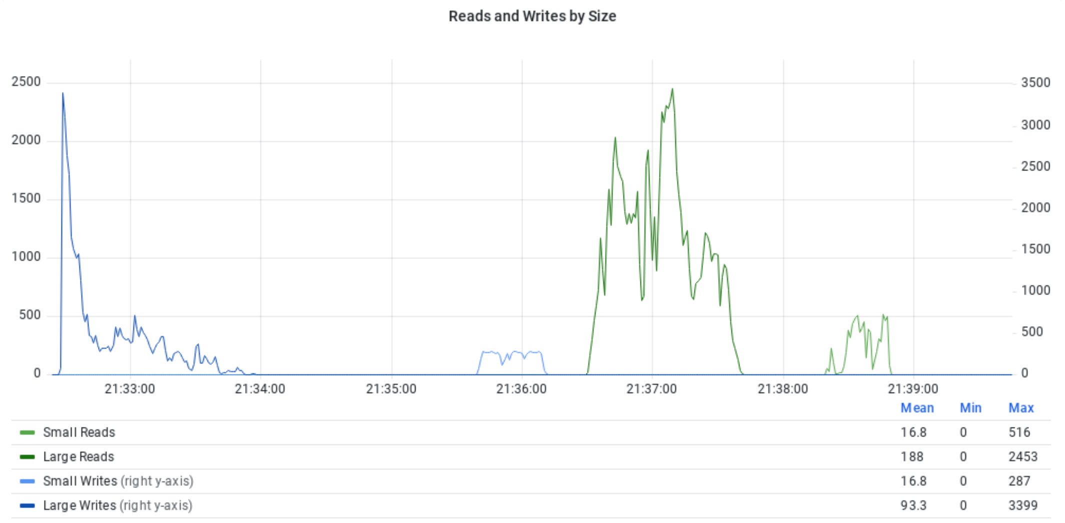
	Mean	Min	Max
🟩 MD Get Operations	2248	0	44841
🟨 MD Mod Operations	1047	0	10871
🟦 MD Other Operations	0.0480	0	2.80

Results



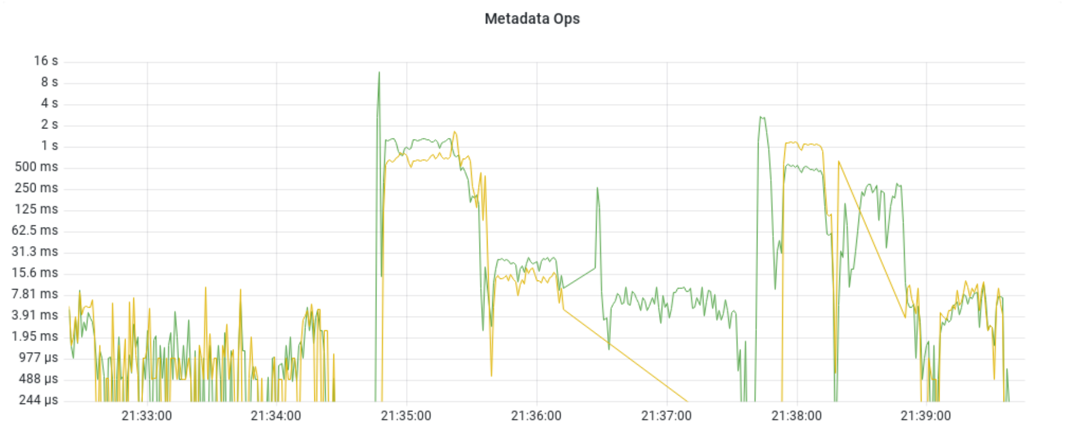
	Mean	Min	Max
🟩 reads	804	0	9814
🟨 write (right y-axis)	796	0	27180

Results



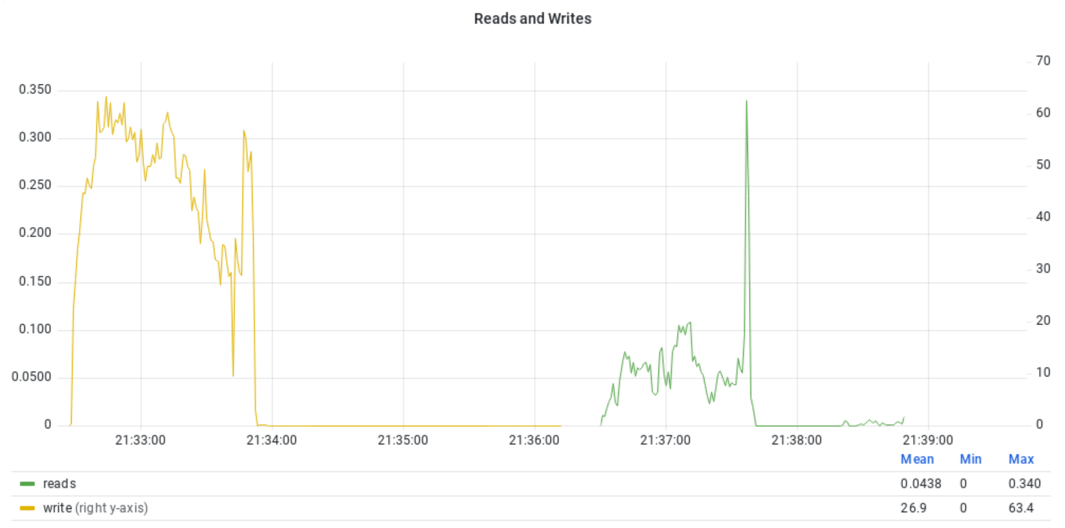
	Mean	Min	Max
🟩 Small Reads	16.8	0	516
🟩 Large Reads	188	0	2453
🟦 Small Writes (right y-axis)	16.8	0	287
🟦 Large Writes (right y-axis)	93.3	0	3399

Results



	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



	Mean	Min	Max
🟩 reads	0.0438	0	0.340
🟨 write (right y-axis)	26.9	0	63.4

Performance impact

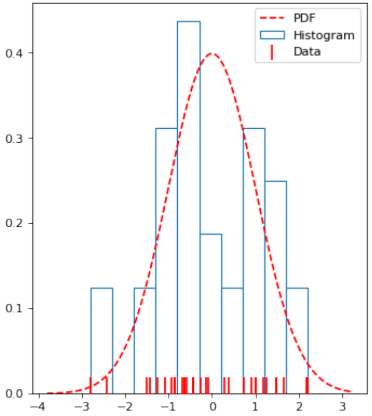
- IO500 tests with SCC rules
- 10 Nodes, 80 tasks
- Result: Obtained 50%-60% of native performance
  - We believe this is alright for non-invasive performance assessment
  - Remember: you can choose which files to access via /monitor!

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
- We have a prototype (blackheap) that automatizes the assessment!

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

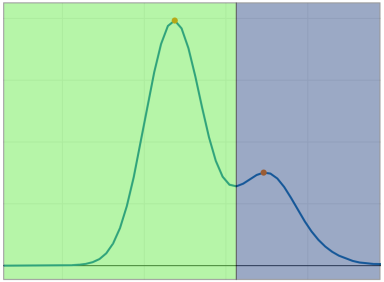
Kernel Density Estimation

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

Outlook

- Support per file accounting (again)
- DB-agnostic reporting
- Integrate modeling and classification to IOFS
- Make recommendations based on classifications

Useful Links

The source code can be obtained at:
`https://github.com/gwdg/iofs/`
`https://github.com/lquenti/blackheap/`

Documentation can be found at:
`https://gwdg.github.io/iofs/book/`