Computer Science Department

HPS

University of Reading

Understanding Metadata Latency with MDWorkbench

Limitless Storage
Limitless Possibilities

https://hps.vi4io.org

Julian M. Kunkel, George Markomanolis

WOPSSS 2018

2018-06-28

Outline

University of Reading

Motivation

University of Reading

The current situation

- Sometimes one file access takes 10s of seconds
  - Access of small files barely possible on DKRZ Lustre system
  - Annoying workarounds need to be deployed
- Performance looks OK with existing benchmarks; motivation for vendors?
- Burst buffers don't help much for metadata (look at IO-500)

Existing MD benchmarks

- Report throughput but don't care about latency
- Focus on bulk-synchronous metadata access
  - Loading one metadata operation at a time is different (locks, idle systems?)
- Take a long time to run, difficult for regression testing

Outline

University of Reading

Features

University of Reading

- Deterministic access pattern that is random-alike
  - Pattern for interactive users and producer/consumer queues
- Fixed working set size during the benchmark run
  - Select to fit or exceed arbitrary cache sizes
- Keep directory tree for performance regression testing!
  - Avoid long create/deletion phases
- Various storage backends
  - POSIX, MPI, S3, MongoDB, PSQL
- Timing of individual operations
  - Investigate distribution of latencies
- Adaptive mode for assessing bottleneck and identifying configurations
  - Are the server overloaded or are the CPUs insufficient to utilize servers?

Benchmark Phases

University of Reading

- Precreation
  - Create objects fast but in the order they will be used
  - Each process operates on its own datasets
- Benchmark
  - A process accesses datasets from D neighbors
  - This phase can be repeated multiple times
    - Regression testing: information about iteration stored in a file
- Cleanup
  - Remove all objects
  - Again each process operates on individual datasets

Parameters for the Access Pattern

University of Reading

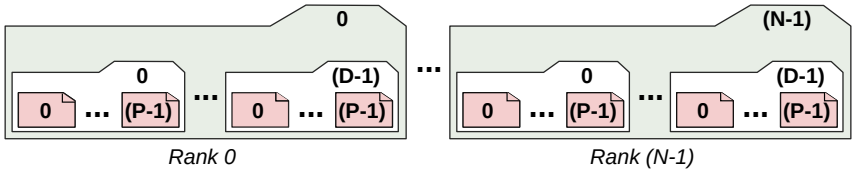
- N: The number of MPI processes
- D: Working set size: number of datasets to create per process
- P: Working set size: number of objects to create per dataset
- I: Benchmarking phase – iterations to perform per dataset
- S: Size per object
- O: Offset in ranks between writer and reader
- T: Waiting factor (discussed later)

Creation Phase

Figure: Pseudocode: Creation phase

```
1  for(p=0; p < P; p++){
2    for(d=0; d < D; d++){
3      dataset = (rank, d) // POSIX namespace is: rank/d/p
4      write(dataset, p)
5    }
6  }
```

Figure: The content of the working set (directory tree) after pre-creation phase



Benchmarking Phase

University of Reading

```
1   for(i=0; i < I; i++){
2     for(d=0; d < D; d++){
3         // The "rank" directories owning the files
4       read_rank = (rank - O * (d+1)) % N
5       write_rank = (rank + O * (d+1)) % N
6         // Access previously created data in FIFO order
7       dataset = (read_rank, d)
8       stat(dataset, i)
9       read(dataset, i)
10      delete(dataset, i)
11        // Append new data to increase the working set size
12      dataset = (write_rank, d)
13      write(dataset, P + i)
14    }
15  }
```

Figure: Pseudocode: Benchmarking phase

File Tree Changes During Benchmark Phase

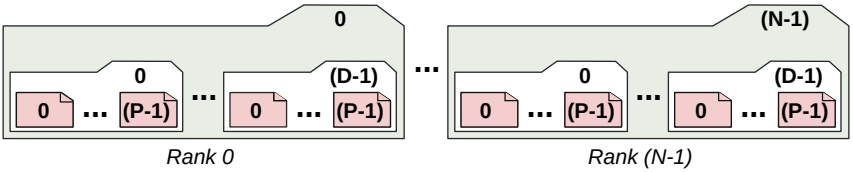


Figure: Working set (directory tree) after pre-creation phase

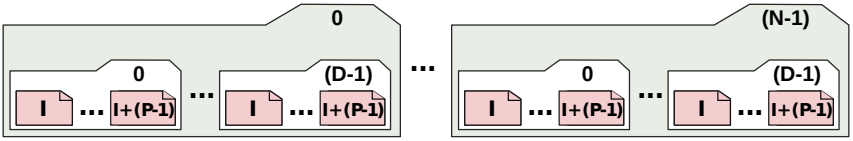


Figure: Working set (directory tree) after one iteration of the benchmark phase

Waiting Time and Excursion: Adaptive Mode

Strategy

- Measure performance for multiple settings
- Adjusts waiting factor to throttle operations
  - Waiting factor $T > 0$, wait T times the time the last I/O took
  - Simulates processing time, e.g., T = 1, wait half the time
- Compute corrected throughput, i.e., expected performance without waiting

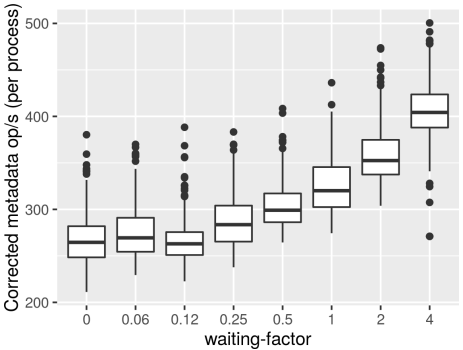
Adaptive Mode: Assessing corrected performance

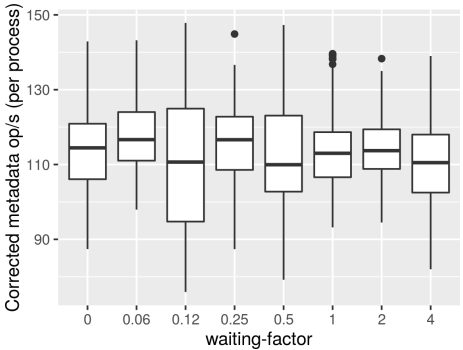
- Performance is **similar** regardless of the waiting time
  - $\Rightarrow$ Servers can handle more clients; clients slow
- Performance **increases** with increasing waiting time
  - $\Rightarrow$ Servers are overloaded, hence queuing of requests
- Performance **decreases** with increasing waiting time
  - $\Rightarrow$ Background operations to the storage dominate

Example for Oberved Performance with Adaptive Mode

University of Reading

- Results for two systems using the same parameters (Cooley, DKRZ)
- 3x repeated measurement
- Each point is the performance reported by one process





Outline

University of Reading

Experiments

Systems

- Cooley at ACLF with GPFS
- Mistral at DKRZ with ClusterStor L300
- IME at DDN test facility in Düsseldorf
- Kaust / NERSC with Lustre and CrayDataWarp

Configuration parameters

- Nodes: 10 (8), 100
- PPN: 1, 10
- D (datasets): 1, 10
- Working sets, either:
  - 10k precreated and 2k obj accessed during benchmark
  - 1k precreated and 200 obj accessed

Comparing Performance Across Phases

University of Reading

System	Nodes	PPN	D	Creation rate (creates/s) Precreate	Bench T=0	Bench T=4
ALCF Cooley (GPFS)	10	10	1	6,500	5,640	8,300
Düsseldorf (Lustre)	8	10	1	47,600	12,600	30,700
Düsseldorf (IME+Lustre)	8	10	1	4,500	1,550	4,460
DKRZ Mistral (Lustre)	10	10	1	21,800	2,380	2,220
KAUST (1 DataWarp BB)	10	10	1	3,800	3,390	14,600
KAUST (8 DataWarp BB)	10	10	1	25,600	8,190	32,000
NERSC (8 DataWarp BB)	10	10	1	19,000	8,560	35,100

- Performance with bulk-synchronous operation differs significantly
- Sometimes it is the same though we do 4x operations in benchmark phase

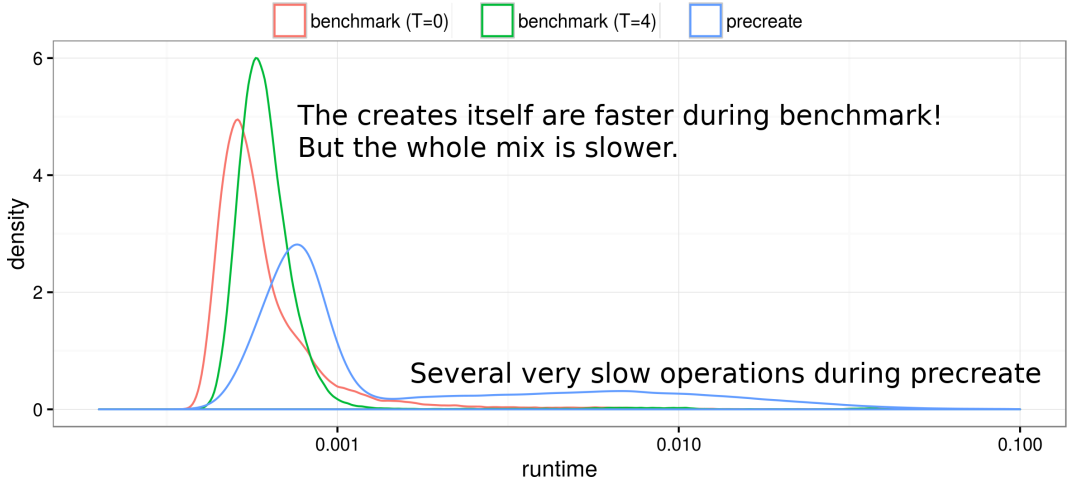
Assessing Density of Create Operations

University of Reading

- Let's investigate the latencies of only the create operations
- Timing of each individual operation is measured and investigated
- Density diagram is roughly a smoothened histogram
- Parameters for the run: D=1, P=10000, I=2000

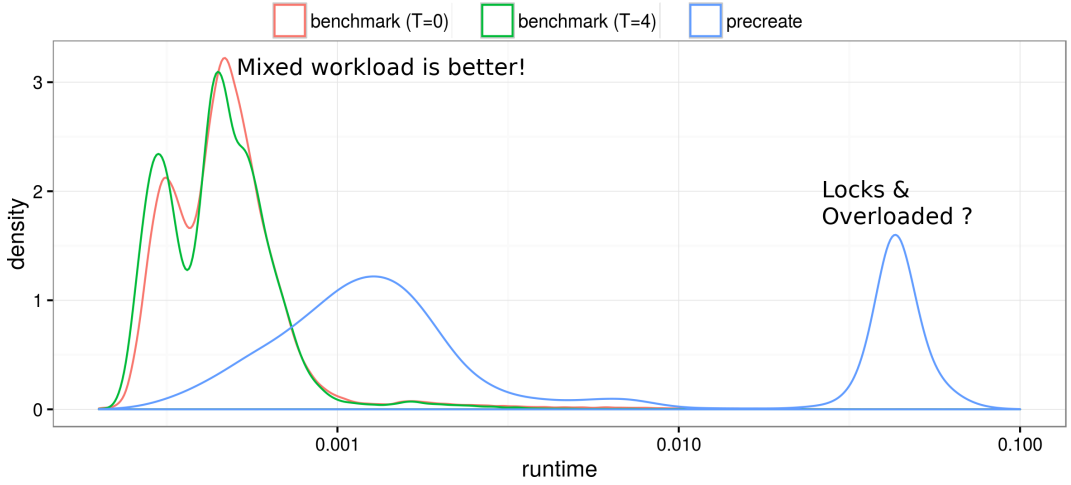
Create Operations: Mistral (Lustre) 10 Nodes, 10 PPN

University of Reading



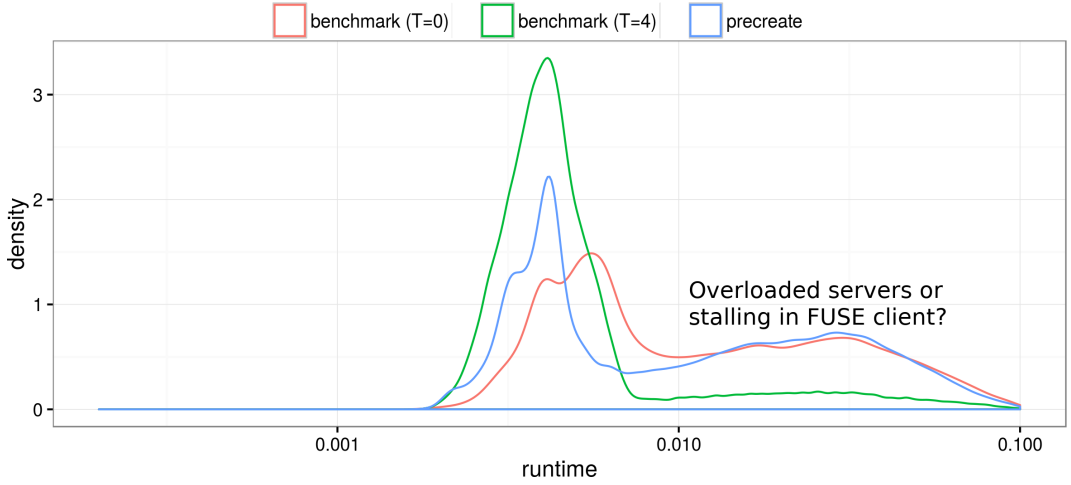
Create Operations: Cooley (GPFS) 10 Nodes, 10 PPN

University of Reading



Create Operations: Düsseldorf (IME) 8 Nodes, 10 PPN

University of Reading



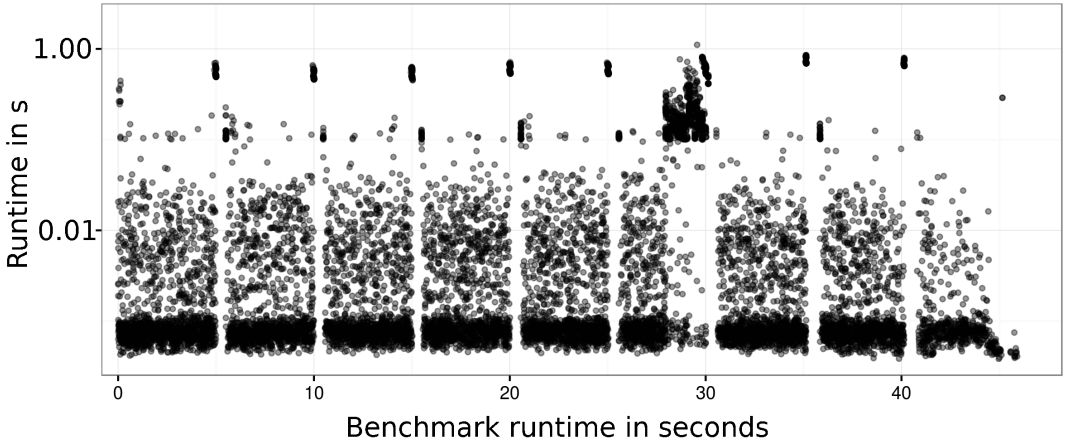
Understanding Latencies

University of Reading

- In the following, we plot the timelines of create operations
  - Sparse plot with randomly selected measurements
  - Every point above 0.1s is added
- All results obtained on 10 Nodes

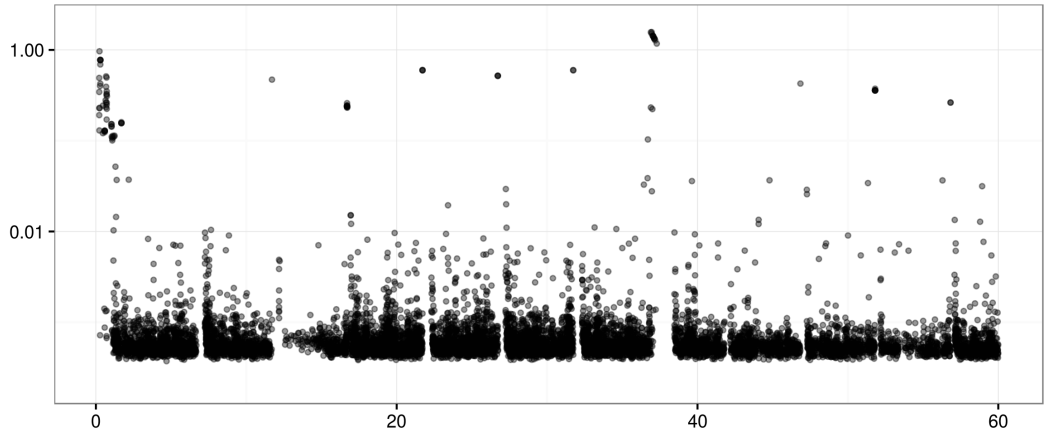
Latencies: Mistral 10 PPN, D=1, I=2000, P=10k, precr.

University of Reading



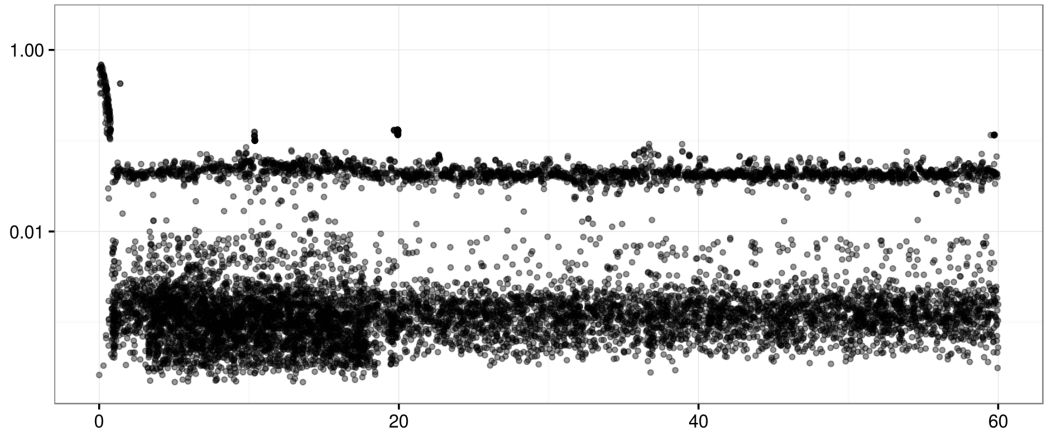
Latencies: Mistral 10 PPN, D=1, I=2000, P=10k, bench.

University of Reading



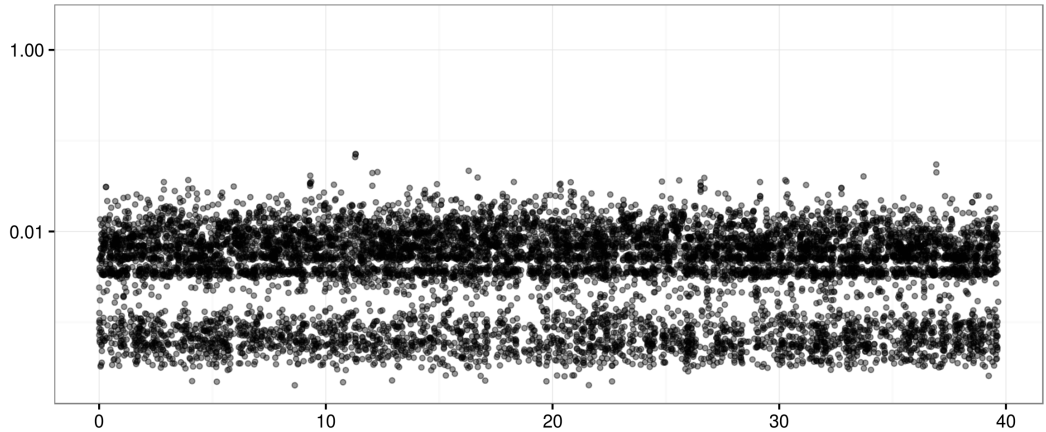
Latencies: GPFS 10 PPN, D=1, I=2000, P=10k, precr.

University of Reading



Latencies: GPFS 1 PPN, D=1, I=200, P=1000, bench.

University of Reading



Comparing File Systems with the Benchmark Phase

University of Reading

- Comparing of key metrics for different file systems
  - Varying configurations
- Max time: for any operation in seconds (interactivity)
  - Personal opinion: should be below 1s, better 0.1s
- Balance: percentage of slowest/fastest process
  - Should be close to 100
- Quartile 3 (time below this 75% of operations complete)
  - Obtained by collecting results from all nodes and computing Q3

Comparing File Systems

University of Reading

Nodes	PPN	D	O	P	Rate creat/s	balance %	max in s	Latency of quartile s read	stat	create	delete
ACLF Cooley GPFS											
10	1	1	2000	10k	1530	95.5	0.13	9.2E-3	8.4E-4	3.9E-4	3.0E-4
10	1	10	200	1k	500	99.8	0.17	1.4E-2	8.5E-4	8.1E-3	4.5E-3
10	1	10	2000	10k	540	100	0.17	1.4E-2	9.8E-4	7.6E-3	4.1E-3
10	10	1	2000	10k	5280	60.7	0.20	1.8E-2	8.3E-4	5.6E-4	2.8E-4
DKRZ Mistral Lustre											
10	1	1	2000	10k	290	86.3	3.60	4.8E-3	6.4E-4	6.4E-4	5.8E-4
10	10	1	2000	10k	2180	68.2	3.50	4.6E-3	4.0E-4	6.7E-4	4.9E-4
10	10	10	2000	10k	2140	78.4	7.90	5.6E-3	4.2E-4	6.3E-4	3.4E-4
100	1	1	2000	10k	1610	72.2	4.60	4.8E-3	6.4E-4	6.5E-4	6.2E-4
100	10	1	2000	10k	4890	77.3	16.00	3.3E-2	6.9E-4	1.1E-2	9.9E-3
DKRZ Mistral Lustre (Procurement phase 1 file system)											
10	1	1	2000	10k	1640	100	0.54	1.0E-3	5.9E-4	7.3E-4	5.3E-4
10	1	10	2000	10k	980	100	3.90	3.8E-3	4.4E-4	7.9E-4	2.9E-4
10	10	1	2000	10k	2660	100	7.40	1.2E-2	8.7E-4	5.4E-3	5.7E-3

Comparing File Systems

University of Reading

Nodes	PPN	D	O	P	Rate creat/s	balance %	max in s	Latency of quartile 3 read	stat	create	delete
Düsseldorf DDN Lustre											
8	1	1	2000	10k	4750	92.4	0.00	5.0E-4	3.1E-4	4.5E-4	3.3E-4
8	1	10	200	1k	4980	95	0.01	5.6E-4	3.2E-4	4.5E-4	3.4E-4
8	10	1	2000	10k	11850	49.5	1.00	1.5E-3	8.1E-4	1.7E-3	1.7E-3
8	10	10	200	1k	10390	40.1	0.10	1.8E-3	9.7E-4	2.0E-3	2.0E-3
Düsseldorf DDN IME											
8	1	1	2000	10k	820	94.9	0.05	7.4E-4	5.5E-4	4.2E-3	4.4E-3
8	1	10	200	1k	820	96.1	0.06	7.3E-4	5.5E-4	4.1E-3	4.4E-3
8	10	1	2000	10k	1540	89.7	0.86	5.4E-3	2.0E-2	2.6E-2	1.2E-2
8	10	10	200	1k	1460	93.4	0.20	8.8E-3	2.3E-2	2.8E-2	1.4E-2

Comparing File Systems

University of Reading

Nodes	PPN	D	O	P	Rate creat/s	balance %	max in s	Latency of quartile s read	stat	create	delete
Kaust DataWarp 1 burst buffer node											
10	1	1	2000	10k	3170	99.2	0.03	6.7E-4	3.3E-4	2.2E-3	3.7E-4
10	1	10	200	1k	3130	98.8	0.06	7.5E-4	3.5E-4	2.1E-3	3.8E-4
10	10	1	2000	10k	3340	94.4	0.18	4.7E-3	7.6E-3	1.6E-2	7.9E-3
Kaust DataWarp 8 burst buffer nodes											
10	1	1	2000	10k	4650	97.6	0.01	5.3E-4	3.3E-4	1.2E-3	3.0E-4
10	1	10	200	1k	5000	96.9	0.01	4.7E-4	2.9E-4	1.1E-3	2.9E-4
10	10	1	2000	10k	7250	82.1	0.16	1.2E-3	4.8E-4	4.1E-3	1.3E-3
10	10	10	200	1k	6510	91.3	0.16	1.2E-3	3.9E-4	3.9E-3	1.2E-3
10	10	10	2000	10k	1860	91.9	0.43	1.2E-2	1.1E-3	1.8E-2	4.9E-3
NERSC DataWarp 8 burst buffer nodes											
10	1	1	2000	10k	4000	95.3	0.03	5.2E-4	1.4E-4	8.8E-4	9.7E-5
10	1	10	200	1k	6670	93.4	0.02	4.2E-4	1.8E-4	8.2E-4	1.2E-4
10	1	10	2000	10k	5540	94.9	0.05	4.8E-4	2.1E-4	1.0E-3	1.4E-4
10	10	1	2000	10k	8770	84.9	0.15	2.4E-3	2.0E-3	5.3E-3	1.2E-3
10	10	10	200	1k	8730	96.8	0.08	2.7E-3	1.9E-3	6.4E-3	1.3E-3

Summary

University of Reading

- MDWorkbench is a tool for understanding performance better
  - Interactivity and latency
- Unique key features
  - Rapid regression testing
  - Identification of server/client load status with adaptive mode
- File system behavior is highly complex
  - Investigation of density diagrams for latency is useful
  - Beware missleading conclusions from investigating mixed workloads
- Source code: https://github.com/JulianKunkel/md-workbench