Computer Science Department

Using Benchmarks to Understand Performance Behavior

Limitless Storage
Limitless Possibilities

https://hps.vi4io.org

Julian M. Kunkel

BoF: Analyzing Parallel I/O; Supercomputing 2018

2018-11-13

LIMITLESS **POTENTIAL** | LIMITLESS **OPPORTUNITIES** | LIMITLESS **IMPACT**

Outline

1 Motivation

2 Benchmarks

3 Regression Testing

4 Lessions Learned

5 Perspective and Needs

Understanding of I/O Behavior and Systems

University of Reading

How can we understand system behavior?

- Observation
  - Measurement of runs on the system
  - Can be many cases to run
  - Slight bias since measurement perturbs behavior
  - Benchmarking: applications geared to exhibit certain system behavior
- Theory: Performance models
  - Used to determine performance for a system/workload
  - Behavioral models

    Build models based on ensemble of observations
- System/application simulation
  - Based on system and workload models

How Can Benchmarks Help to Analyze I/O?

- Benefits of benchmarks
  - Can use simple/understandable sequence of operations
    - Ease comparison with theoretic values (that requires understandable metrics)
  - May use a pattern like a realistic workloads
    - Provides performance estimates or bounds for workloads!
  - Sometimes only possibility to understand hardware capabilities
    - Because the theoretic analysis may be infeasible
- Benefits of benchmarks vs. applications
  - Are easier to code/understand/setup/run than applications
  - Come with less restrictive "license" limitations
- Flexible testing (strategies)
  - Single-shot: e.g., acceptance test
  - Periodically: regression tests

Outline

1 Motivation

2 Benchmarks

3 Regression Testing

4 Lessions Learned

5 Perspective and Needs

Benchmarks

University of Reading

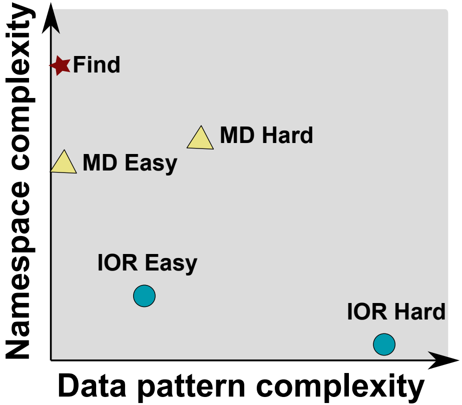
- Many I/O benchmarks exist covering various aspects
  - APIs used
  - Data access pattern
  - Memory access pattern
  - Parallelism and concurrency
- Let’s talk about the IO-500 benchmark suite; it is
  - **Representative**: for optimized and naive workloads
  - **Inclusive**: cover various storage technology and non-POSIX APIs
  - **Trustworthy**: representative results and prevent cheating
  - **Cheap**: easy to run and short benchmarking time (in the order of minutes)
  - Favors a single metric to simplify the comparison across dimensions

Goals of the IO-500 Benchmarking Effort

- Bound performance expectations for realistic workloads
- Track storage system characteristics behavior over the years
  - Foster understanding of storage performance development
  - Support to identify potent architectures for certain workloads
- Document and share best practices
  - Tuning of the system is encouraged
  - Submitters must submit detailed run parameters
- Support procurements, administrators and users

`https://io500.org`

Covered Access Patterns



- IOR-easy: large seq on file(s)
- IOR-hard: small random shared file
- MD-easy: mdtest, per rank dir, empty files
- MD-hard: mdtest, shared dir, 3900 byte
- find: query and filter files based on name and creation time
- Executing concurrent patterns not covered (another dimension)

Comparing Systems

We can use the score as a single value to compare between file systems, or compare for a specific benchmark type

`https://io500.org`

Predictability and Latency Matters

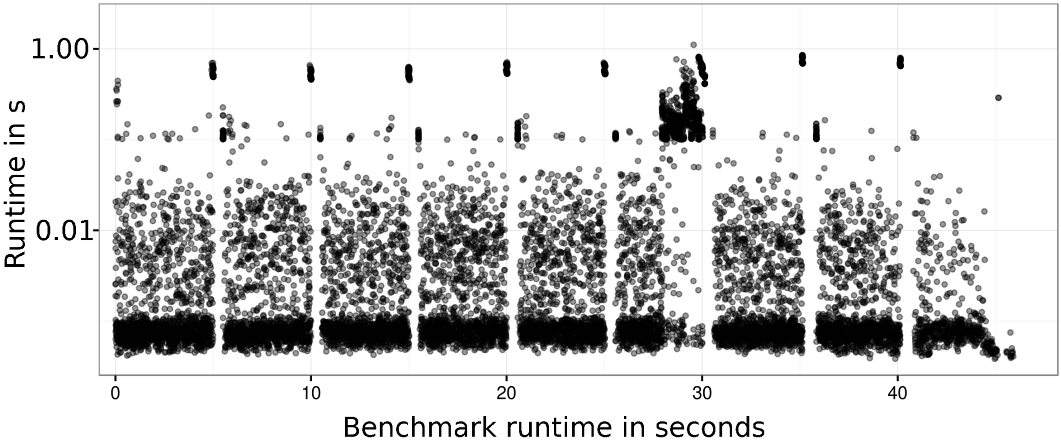
Performance Predictability

- How long does an I/O / metadata operation take?
- Important to predict runtime
- Important for bulk-synchronous parallel applications
  - The slowest straggler defines the performance

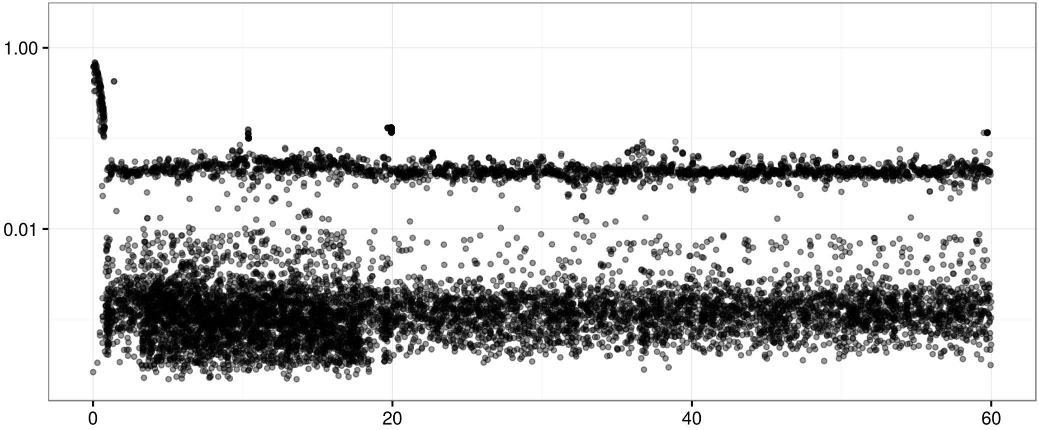
Measurement

- In the following, we plot the timelines of metadata create operations
  - Sparse plot with randomly selected measurements
  - Every point above 0.1s is added
- All results obtained on 10 Nodes using MD-Workbench https://github.com/JulianKunkel/md-workbench
  - Options: 10 PPN, D=1, I=2000, P=10k, precreation phase

Latencies: Lustre / Mistral at DKRZ



Latencies: GPFS / Cooley at ALCF



Probing File System Performance

University of Reading

- Regression: repeated runs of a file system benchmark
  - Compare performance behavior over time
  - Trace impact of upgrades/slowdown of hardware
- Coarse-grained regression done by many sites
  - E.g. DKRZ runs several benchmarks every night
  - Insightful but limited!
- Fine-grained runs allow to understand performance users would see
  - E.g. overloaded servers

Example: Constant probing on the JASMIN cluster

- Using dd and md-workbench constantly

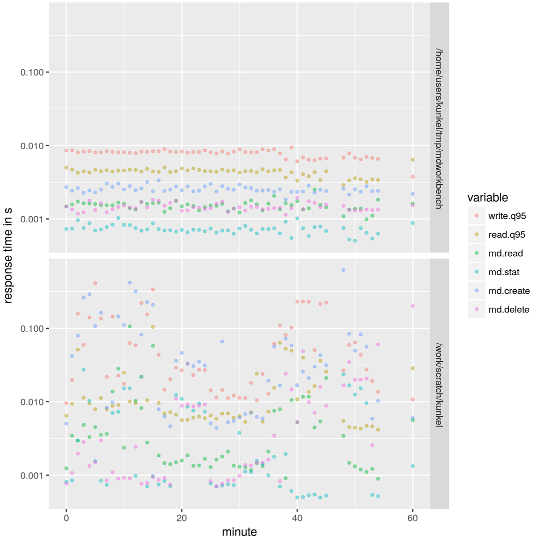
  Run a probe of 1MB data and one file every second!
- Analyse data across time
  - Use 95% quantile to indicate performance behavior for 5% slowest requests

Regression Testing

- The first hour of measurement
- We aggregate measurements per minute together

Observations

- Home file system is stable (good!)
- Work file system differs by 2-3 orders of magnitude
  - For metadata and I/O metrics

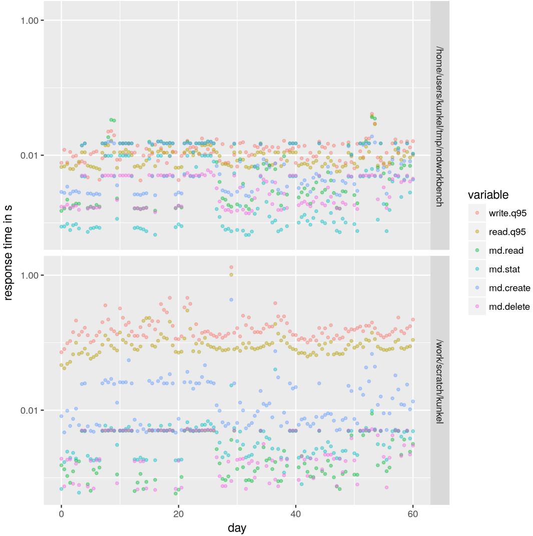


Regression Testing

- 60 days worth of measurement
- We aggregate 12 hours together

Observations

- Phases where > 5% ops are significantly slower
- Metadata varies more
- Data response is stable at 100ms
  - 95% of requests are satisfied



Outline

University of Reading

1 Motivation

2 Benchmarks

3 Regression Testing

4 Lessions Learned

5 Perspective and Needs

Lessions Learned

- Determine system performance with measurements
  - Rely on understandable benchmarks/mini-apps
  - Establish performance expectations
- Latency and performance-predictability important
- Run regression tests
- Have a small test file system (across all servers/disks)
  - Not quite there... Would have been frequently useful

How to Proceed with Benchmarking for Analyzing I/O?

In a perfect world, we have

- Embedded performance models in hardware/software
  - Available to monitoring tools to assess performance
    I/O is good, bad, ...
  - Checkout the VI4IO activity: Next Generation I/O Interfaces
- Embedded benchmarks that verify behavior according to model
  Hardware is too slow, broken, ...

Appendix

Resulting Metrics

How do we weight input from multiple benchmarks?

Tuning for improving the Geom-Mean value

Description	Input (11 values)	**Geom**	Arithmetic	Harmonic
Balanced system	10 . . . 10 10 **10**	10	10	10
One slow bench	10 . . . 10 10 **1**	8.1	9.2	5.5
Tuning worst 2x	10 . . . 10 10 **2**	8.6	9.3	7.3
Tuning good 2x	10 . . . 10 **20 1**	8.6	10.1	5.6
Tuning good 100x	10 . . . 10 **100 1**	10	17.4	5.8

- Geom mean honors tuning equally, insensitive to “outliers”
- Harmonic mean favors balanced systems (complex to scale results)