Metadata/Small File Benchmarking
Status of the IO-500

Julian M. Kunkel

Deutsches Klimarechenzentrum GmbH (DKRZ)

2017-03-22

Outline

Metadata Benchmarks

Classes of (metadata) benchmarks

- Applications would be best to estimate performance but have issues
  - Difficult to setup, confidential code / data, time consuming to run
  - Difficult to understand the access pattern and observed performance
  - How many applications are needed to represent the I/O workload?
- Mini-apps
  - Much simplified versions of the applications
- IO-kernels
  - Only the I/O (evtl. communication) of mini-apps
- Trace/Replay
  - Record application (IO) behavior and replay
- Synthetic
  - Typically designed to stress well-known / relevant access patterns
  - Workload generators allow to program various patterns

Existing Metadata Benchmarks

The following list of benchmarks focuses on small files / object accesses:

Sequential benchmarks

- Postmark: randomly chooses list of operations on a working set
- fdtree: bash based benchmark
- mds-survey (Lustre specific)

MPI parallel benchmarks

- MDTest: operates in phases: create, read, delete
- metarates: an MDTest with more POSIX calls
- MD-REAL-IO: (I will talk about this later)
- Parabench: programmable workload generator
- FEIGN/SIOX: Trace/Replay

Existing Metadata Benchmarks

Cloud storage benchmarks

- Mimesis: Trace/Replay: statistical workload generator
- 1 billion document benchmark
- COSBench: workload generator; supports adapters/plugins
- The Virtual Instruments Object Storage performance validation

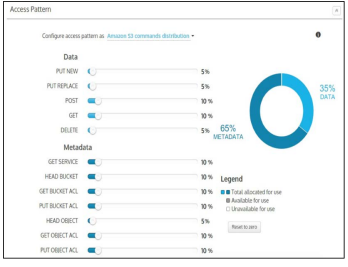


Figure1: Sample screen capture, Source: Virtual Instruments Info Sheet, 2016

Challenges Designing (Metadata) Benchmarks

- Definition of an appropriate workload according to the benchmark's goals
  - Trend: cloud benchmarks move towards workload generators
- Supporting various (future) storage architectures; choice of interfaces
  - POSIX: very flexible making it difficult to define the workload
    - Is the workload actually useful or is it abuse of semantics (to, e.g., control job flow)?
  - MPI-IO is too restricted
  - Object storage and cloud: SWIFT, S3, MongoDB, ...
  - Data bases?
  - Trend from cloud computing: plugins/drivers to support various backends
- Reproducible results (on the same system)
  - Re-running the benchmark should retain similar performance results
  - Degradation of hardware due to production usage?

Challenges for Running (Metadata) Benchmarks

- Interference with optimizations
  - Storage systems and on-disk formats are optimized for certain workloads
  - Caching/pre-fetching can increase performance significantly
  - Is caching what we want to measure?
- Interpretation of the benchmarking results
  - What can we learn about the system from the results?
  - Can we transfer results to predict production performance?
- Comparability of results between systems
  - Apple-to-apple comparison?
  - Workloads have a huge impact of performance
  - Difficult when using workload generation or trace-replay
- Preventing (self) cheating
  - Mitigation: relating results to hardware capabilities should be easy
  - We should also prevent vendors to cheat by deploying a “new” optimization

Pretending Good Performance by (Self) Cheating

Experiences talking to vendors talking years ago

- SSDs are unnecessary/do not improve file system performance for metadata

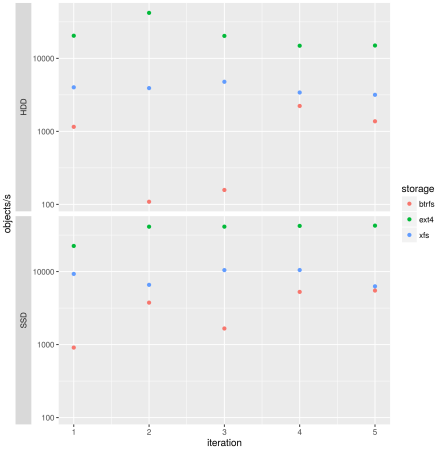
Experiences during our last procurement

- When asking for metadata performance
  Vendor A: we can easily do millions of IOOPs
  We: When using 200 hard disk drives for the metadata server?
  We: Could you use our benchmark?
- After checking our benchmark, in the offer a realistic value was listed

The cause is simple: using a benchmark for which systems are optimized / cache

Measurements for a Mixed Read/Write/Stat Workload

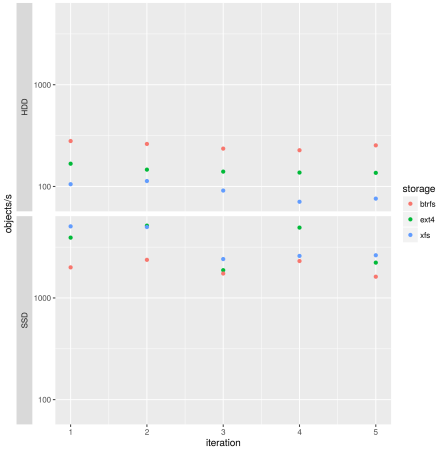
- Node local storage, 1000 MByte free memory, 2 processes, 5 repeats



- HDD performance is good
- With EXT4, SSD and HDD similar
- For btrfs, varies between repeats
  - An indicator for the issue...
  - Cause: flush timer inside OS is sometimes triggered
- This workload is cached...

Measurements for a Mixed Read/Write/Stat Workload

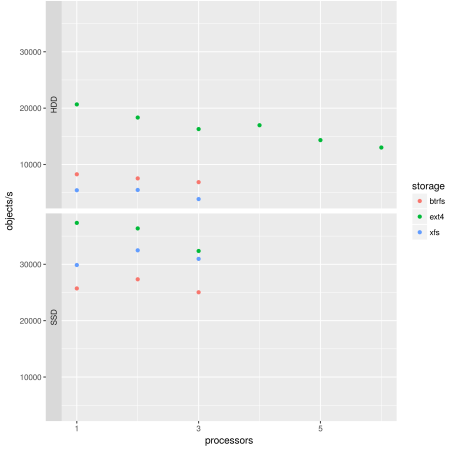
- Same setup, larger working set



- More realistic?

Optimized Bulk-Creates aka md-test

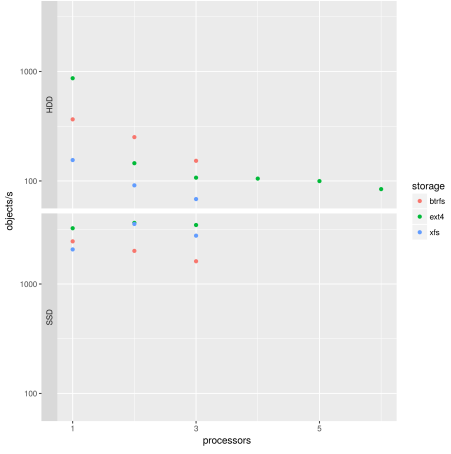
- Only creating files; 600,000 objects a 3900 Byte; 1 GByte memory
- Fixed working set size 2 GByte partitioned across processors



- The working set should be enough to show true HDD performance
- EXT4 still optimizes bulk imports well
- Vendor statements: SSD may improve performance by 2x ...
- Which file system would a vendor use?
- Which storage media would he sell?
- Actually performance reduces only slightly increasing the working set

Adjusting the Workload to Prevent Caching

Same experiment/working set, adjusted workload/op order to prevent caching



- Some aggregation can still be done
  - Write behind / HDD cache improves performance for 1 proc
- SSD and HDD behave realistic

Towards Realistic Values

- Know the hardware characteristics
  - With 7ms avg. latency, 150 IOOPs/s per (metadata) HDD
  - Coordination mechanism of the file system will reduce this value
  - How much cache is available?
- Know the semantics
  - When are modifications visible to other clients or become durable?
  - What information could be potentially be cached somewhere?
- Define a workload that can be scaled to exceed any cache
  - Scale the workload down to see how well the system optimizes / caches
    $\Rightarrow$ best-case
  - Scale the workload up to exhibit hardware behavior
  - Make sure in the tender that you can adjust the workload for acceptance

MD-REAL-IO

- An open source benchmark
  `https://github.com/JulianKunkel/md-real-io`
- Plugins for POSIX, MPI-IO, Postgres, MongoDB, S3
- Operates on shared “directories” / objects
- Phases:
  - Precreate a working set (optional)
  - Benchmark
    - stat, open, read, close, unlink a single object from the working set
    - open, write, close a new object ⇒ the working set stays the same throughout the test
  - Cleanup (optional, one can run the test repeatedly over the working set)
- Interpretation:
  - Multiple FIFO producer/consumer systems processing small data
  - Interactive usage from many users on a HPC system

Results

- The mixed workload shown before uses MD-REAL-IO
- Realistic working set: runtime on Mistral 12 minutes
- Creating a working set can take more time but a small set yields nearly same performance results
- The working set is 3,000,000 objects, 11 GiB
- Performance on our last-generation Blizzard supercomputer: 250 objects/s (x 8 ops/iteration)
- Mistral using a single metadata server (we have 5+7 servers)
  - Phase 1 (in production): 1200 iter/s, 9 MiB/s
  - Phase 2 (nearly empty): 7000 iter/s, 53 MiB/s
- Earth-Simulator: 1880 iter/s, 14 MiB/s

Status of the IO-500

Goal: the development of I/O benchmarks for tracking I/O performance

History

- Dec. 2015: The High-Performance Storage list has been created
  - Contains a trivial approach to overcome obstacles
- June 2016: Talks from Lofstead, Kunkel about benchmarking during ISC BoF
- Nov. 2016: joint BoF from Kunkel, Lofstead and Bent during SC
- Nov. 2016: Creation of a mailing list for subsequent discussion
  - There have been discussions about the approach, benchmarks

---

- April 2017: Voting about (initial) benchmarks
- Mid 2017: Identify benchmarking rules, ask Top500 sites to run benchmarks
- Nov. 2017: Show results during SC in a BoF

Challenges When Creating a Benchmark

- Storage systems are heterogeneous
  - Storage hardware: SSDs, HDDs, NVRAM
  - Availability of optimizations for random and sequential workloads
  - High-level concepts, e.g., staging (K Computer), burst buffers
- Representativeness of a single metric / benchmark
  - Workloads are very diverse, what do we want to measure?
  - With a fitting benchmark systems extract close to peak performance
  - With another benchmark only 1/100 th of performance
- Runtime for executing a benchmark
  - Executing a specific I/O benchmarks may take quite some time
  - Are you willing to pay for it just to be included on a storage list?

Compatible Approach of the High-Performance Storage List

Strategy

- Community-managed list tracking many (theoretic) characteristics

Mitigating the obstacles

- *Storage systems are heterogeneous*
  - Communicate a system model that fits most use cases
- *Representativeness of a single metric / benchmark*
  - Rely mostly on theoretic values
  - Allow users to utilize any benchmark/app to determine sustained performance
- *Runtime for executing a benchmark*
  - Optional values: a site can publish computers with a subset of values
  - No overhead, since users can use their own benchmark

Collected Information

Peak Performance

- Theoretical value based on hardware limits
  - e.g. network (server) throughput, SATA limits
- Best performance of one server x number of servers.
- Describe in the text how the peak is computed

Sustained Performance

- Actually observed performance with an application or benchmark
- You can use any benchmark and measurement protocol
- Just make sure you are not measuring cache effects
- Describe in the text how the value has been measured

Back to the IO-500

Requirements of the benchmarking

- Representative: for optimized or naive workloads
  - Describe the natural requirements for users
  - IO-easy: upper bound for optimized IO-heavy workloads
  - IO-hard: expected performance for non-optimized applications
  - MD-easy, MD-hard: likewise but cover small-objects/metadata
- Inclusive: cover various storage technology and non-POSIX APIs
  - At best: useful for HPC and Big Data workloads
- Trustworthy: representative results and prevent cheating
- Cheap: easy to run and short benchmarking time (in the order of minutes)

IO-500

Strategy

- Build on (knowledge of) existing benchmarks
  - IOR
  - MD*
- Plugin systems should allow for alternative storage technology
- Initially report one (two) metrics per benchmark
  - Decide later about one representative number
  - A single number should favor balanced (useful) systems
  - Potentially storage capacity should also be part of this metrics

Summary

- Existing benchmarks: mini-apps/kernels, trace-replay, synthetic
  - Results from application alike benchmarks are difficult to assess
- Many existing tools cannot solve MD benchmarking challenges
  - Bulk operations on independent data sets can be well optimized
- A benchmark should
  - Reveal optimized / caching and true hardware performance
  - Be representative, inclusive, trustworthy and cheap to run

`https://www.vi4io.org`

See `https://www.vi4io.org/listinfo` for the IO-500

You are welcome to participate in the VI4IO and the IO-500 efforts