Rethinking Benchmarks for Non-Volatile Memory Storage Systems

Ethan L. Miller
Symantec Presidential Chair in Storage & Security
University of California, Santa Cruz

SSRC Storage Systems Research Center

Center for Research in Storage Systems

Baskin Engineering UC Santa Cruz

Why benchmark storage?

CENTER FOR RESEARCH IN STORAGE SYSTEMS

- Understand storage system limitations
  - Feed into storage system development
- Compare storage systems to one another
- Predict performance in *your* environment

But we have lots of benchmarks!

CENTER FOR RESEARCH IN STORAGE SYSTEMS

fio

SPC-1

PostMark

TPC

iozone

TPC-x

bonnie

vdbench

YCSB

iometer

…and many more!

What makes a benchmark?

CENTER FOR RESEARCH IN STORAGE SYSTEMS

- Location: where are the requests?
  - And where are they relative to other requests?
- Size: how big are the requests?
  - Constant / variable?
- Access type: what kind of requests (read/write/other)?
  - Block vs. file
- Content: what data is in the requests (for writes)?
  - This can be very tricky!
- Timing: when are the requests?
  - Start time vs. interval
- *All* of these can be different for NVM!

Location, location, location

- Disk-based systems: location really matters!
  - Seek times are location-dependent
  - Virtualized storage can muddle this

- NVM: location shouldn't matter, right?
  - Location in NVM doesn't determine latency
  - NVM storage is even more highly virtualized anyway!

Access location & NVM

- Access order impacts NVM-based data structures
  - Number of copies / invalidations for copy-on-write and write-out-of-place structures can differ
  - More overwrites ➡ more garbage collection
- More invalid copies ➡ more accesses on reads

- Example: preloading storage system with data
  - Standard approach: write data sequentially
    - May be “best case” for copy-on-write systems
    - This isn’t realistic!
  - Better (realistic) approach: write and overwrite data in “patches”
    - Better reflection of reality
    - More fragmentation

Size matters

- Disk: sequential I/O is significantly faster than random

- NVM: I/O size is less important
  - Each I/O pays one overhead
  - Larger I/Os might pay additional cost due to *internal* structures

- Example: single 128KB I/O vs. 16 8KB I/Os
  - Log-structured system: might be able to read entire 128KB sequentially
    - But not if there was significant overwrite, scattering data
  - Content-addressable system: relatively faster for many smaller I/Os
    - Forces log-structured system to pay the same overheads it already does
  - Compression: alters boundaries

Access type: blocks vs. objects

- Disk-based systems are either block or file-based
  - Blocks: little or no lookup time
  - Files: lots of extra overhead on operations

- NVM-based systems are likely to be byte or object-based
  - Byte-level operations: very small, *very* fast: expect millions to billions per second
  - Object-based operations: less overhead than file-based: still may have millions per second

Content is king

- Disk: content doesn't matter that much
  - Disk systems can use compression and deduplication, but most don't
  - Why bother "increasing" capacity when IOPS matter more?
  - Even worse, compression & deduplication decrease performance
  - Result: content isn't as important for disk systems

- NVM is different!
  - Expensive: data reduction reduces cost
  - Very fast: performance implications of data reduction are minimal
  - Data reduction is very content-dependent

Compressing content

- Data must be compressible, but not too much
  - Typical data compresses at 2:1–4:1
  - Different algorithms compress in different ways
  - Benchmark must generate data that compresses in the same way user data does

- Potential benchmark approaches
  - Reuse sampled data from deployed systems?
  - Generate data that matches that from deployed systems?

Deduplicating content

- NVM-based systems reduplicate heavily
- Benchmarks need to mirror real-world systems' duplicate data patterns
  - Fraction of deduplicated data
  - Layout of deduplicated data
  - Access pattern (write order) needs to match as well
    - Detection of duplicates can be order-dependent
- Example: deduplication & VMs
  - Writing VMs sequentially can result in *less* deduplication than writing VMs in parallel
  - Using XCOPY (as real-world systems likely would) can be more efficient
  - Minor data rearrangement matters: on what boundary is deduplication detected?

Time is of the essence

- Timing at which I/Os are issued determine performance
  - Typical approach: multiple threads issue as quickly as possible
    - Performance determined by the number of threads
  - Problem: faster systems issue I/Os faster
    - This isn't always realistic

- Better approach: determine timing from start-to-start
  - I/Os issued at a fixed rate
    - Or, follow a given inter-arrival distribution
    - Rate itself may vary over time!
  - Faster systems aren't penalized
  - Important for NVM systems that can run at 1,000,000+ IOPS!

Speed kills

- Disk-based systems are speed limited
  - 1000 disks only runs at 200,000 IOPS
- NVM-based systems are *much* faster
  - Millions of IOPS!
  - Benchmarks need to keep up!

- Issuing requests at over 1M/second is challenging for a single system
  - Using multiple cores and threads helps, but…
  - Coordinating multiple threads is difficult, especially if the benchmark needs to be repeatable

Repeatability

- Often useful to be able to re-run the benchmark
  - Repeatable results
  - Regenerate the same data later
- Requires deterministic random numbers
  - Straightforward: any DRNG will work *if* all threads cooperate properly
  - Problem: coordinating threads is difficult
- Threads need to use random numbers in a fixed order
  - Centralized RNG may work
  - Need to ensure that threads use the numbers in the same order each time
  - Better approach: one thread generates workload while worker threads consume it

The elephant in the room: overall workload

CENTER FOR RESEARCH IN STORAGE SYSTEMS

- Benchmarks are supposed to mirror real workloads
- We don't yet know how applications will use NVM-based storage with microsecond latency!
  - Memory-like access?
  - Object / variable-sized chunk access?
- Big question: what will the access *pattern* look like?
  - Different characteristics enable different application access patterns
  - What will applications do with 1M+ IOPS?
- This will have one of the largest impacts on benchmarks

Implications for benchmarks

- Benchmarks need to be more accurate
  - NVM-based system performance will be more dependent on the software that manages them
  - GIGO: if the benchmark doesn't mirror real-world conditions, system designers may be optimizing for the wrong thing
- Benchmarks need to be faster
  - Need to support 2M+ IOPS for a single benchmark
- Benchmarks need to be deterministic
  - "Random" is helpful, but we need repeatability
- Achieving all of these goals will be difficult

CENTER FOR RESEARCH IN STORAGE SYSTEMS

Questions?

elm@cs.ucsc.edu