Establishing the IO-500 Benchmark

Julian M. Kunkel*, John Bent†, Jay Lofstead‡, George S. Markomanolis§,
* DKRZ, † Sandia National Laboratories, ‡ Cray, § KAUST Supercomputing Laboratory

***Abstract*—Benchmarking of HPC storage systems is a complex task. Parallel I/O is not only influenced by CPU performance for latency and the networking stack but also on the underlying storage technology and software stack. With the IO-500, we have defined a comprehensive benchmark suite that enables comparison of high-performance storage systems. Similar to the TOP500 list for compute architectures, IO500 will allow tracking performance growth over the years and analyze changes in the storage landscape. The IO-500 will not only provide key metrics of expected performance, but serve as a repository for fostering sharing best practices within the community. Unlike other benchmarking efforts, by encouraging "gaming" the benchmark for the "easy" tests—and requiring disclosure of how these tests were performed—we can document how to achieve best practices while avoiding dishonest benchmark results.**

I. Challenges and the IO-500 Approach

The following list stems from the discussion of the benchmark suite in smaller groups and meetings around birds-of-a-feather sessions during the ISC High Performance and the Supercomputing conference as well as a Dagstuhl workshop. Additionally, the community ran the suite on several sites, and we incorporated feedback into the benchmarks to make them easier to run and more clear. We settled on a collection of "hard" tests with preconfigured parameters designed to show worst-case scenarios. A second set of "easy" tests are completely up to the user and should show the potential of the storage system tested. The product of the harmonic mean of the bandwidth and IOPS measures generate a single, overall number to represent the ranked balance.

Representative: a suite of benchmarks should represent typical workloads observed on real systems. This includes well tuned and optimized but also interactive, naive and unoptimizable workloads. We harness IOR, mdtest, and standard POSIX find to assess system performance representing optimized sequential and random I/O and metadata workloads. While none of these are perfect, they can be made to perform well for both "easy" and "hard" configurations.

Understandable: meaningful metrics should be generated for systems. At best, metrics are meaningful for data center staff and users. Variability should be low, thus, repeated measurements should achieve similar results. We use I/O bandwidth in GiB/s or thousands of I/O operations per second (k-IOPS) for each individual benchmark run. In addition to the composite score, the individual values are retained and can be examined for a more nuanced view.

Scalable: it shall run at any scale on a large-scale computer and some storage systems. The resulting metrics should be assessed according to the number of client/server nodes used and processes per node.

Portable: the benchmark should cover various storage technology and non-POSIX APIs. Our core benchmarks use the AIOR interface from IOR that supports several backends. We will continue to extend and port other POSIX code from the benchmark suite to it. During testing, we faced incompatible options in standard command line tools and within Python wasting user's time. Shell scripting to sequence the benchmarks turned out to be error-prone prompting a more portable, easily configurable approach. We now are working on a C-version of the benchmark code that does not rely on external tools but still links library versions of mdtest and IOR. The C-API across systems is not without challenges. For example, Cray's DataWarp does not support the d_type flag within `readdir()`, the `stat()` call on the Earth Simulator needs two more arguments; it also provides another low-level timer function.

Inclusive: optimizations available on individual storage technology, like the GPFS `find` that searches the database, are allowable. Vendors are keen to use these features or they will be reluctant to support submission of benchmark results. The suite honors that by enabling extensions to IOR's AIORI and integration of alternative find implementations. From the high-level, the overall access patterns cannot be changed but we will work on extensions to the covered metrics that complement the current space. However, all changes must be submitted to the public benchmarks and cited in the results submitted to ensure propagation of best practices.

Lightweight: runtime of the benchmark should be in the order of minutes with easy setup and tuning possible. This should reduce the costs of running the benchmark to allow a comprehensive list. For a valid run, the creation/write phases of individual benchmark must be configured to run for at least 5 minutes; read only operations can go faster. To simplify meeting this requirement, we improved the existing stonewalling option in IOR to ensures that all process write the same amount of data (preserving stragglers as a bulk synchronous application would see). A stonewall option for mdtest is in the incubator. Since 5 minutes per hour is a typical platform acquistion performance requirement for checkpoint/restart write operations, the community agreed it is the best option to address cache flushing and keep the tests realistic.

Trustworthy: Experts must trust the benchmark results and it should prevent (unintended) cheating. A submission to the IO-500 requires revealing all tunings made. These will be shared as part of the results giving insights for others to understand and potentially benefit from the experiments by adapting useful options.