Benchmarking Application I/O in the Community

Julian M. Kunkel
Deutsches Klimarechenzentrum GmbH
20146 Hamburg
kunkel@dkrz.de

Olga Mordvinova
Ruprecht-Karls-Universität
69120 Heidelberg, Germany
mordvinova@informatik.uni-heidelberg.de

Dennis Runz
Ruprecht-Karls-Universität
69120 Heidelberg, Germany
dennis.runz@gmx.de

Michael Kuhn
Department of Informatics,
University of Hamburg
22527 Hamburg, Germany

Thomas Ludwig
Department of Informatics,
University of Hamburg
22527 Hamburg, Germany

Keywords
Benchmarking, I/O, Application Benchmark, Tracing

ABSTRACT
Benchmarking I/O performance of a high performance computer is a tough task. Many sequential and parallel benchmarks exist, however they differ in access patterns, tested interfaces and internal behavior. Interpretation of results on one hand depends on the benchmark, on the other, projection of obtained results to particular applications is not easy or even impossible. Porting scientific applications to an architecture is complicated, library and architecture dependencies are a burden to developers. Therefore, a few application benchmarks exist, which try to mimic application behavior in a small core. However, writing an individual application kernel as a new benchmark for each application leads to a wide diversity in benchmarks. We propose the benchmarking tool *Parabench*, which allows to mimic a rich variety of application programs. *Parabench* interprets an easy programming language during runtime to avoid portability or licensing problems. In addition, a future *Trace-Replay* mechanism will allow to replay application traces directly on other systems by interpretation of the traces. Standardized tests of the results ease evaluation and futher result comparison. We will start an open community and collaborations to exchange patterns and measured results on the website. Vendors can take the access patterns and provide results, obtained on new architectures and systems.

1. BENCHMARKING APPLICATIONS
In this poster we present the programmable I/O benchmark *Parabench* [9]. *Parabench* allows to mimic I/O behavior of a wide range of applications. Additionally, the concept of *Trace-Replay* is introduced. *Trace-Replay* allows to execute traces of applications on different systems. Typically, application traces are used for performance optimizations, however an interpreter could reproduce the traced behavior. *Trace-Replay* will allow to record communication, I/O and quantitative CPU activity i.e. CPU counters and replay the behavior. While there is no implementation so far, we will prepare an implementation. Compared to run an application directly on a system this technique allows to record runs of real applications. These traces can be shared and run without dependency on particular libraries or systems, or without dealing with licensing issues. Without the need to port the scientific application to the new system, this technique offers to record inefficient application runs or store runs as a gold standard. Developers of middleware can use these traces to optimize their software-stack. Depending on the implemented tracing, calls to third party libraries such as HDF5 could be traced and optimized to the same extent. Both *Parabench* and *Trace-Replay* check return values of calls to ensure proper benchmarking – by providing a core of functions with correct error handling derived benchmarks will avoid common mistakes in benchmarking. Finally, the exchange of patterns and traces in an open community will allow third parties to check the performance of their systems under realistic conditions. With such an open platform we hope the community will share patterns of common applications to provide a rich and diverse repository for evaluating current and future systems. Versioning of trace files will allow to compare old program version with new version without providing the outdated and even incompatible environment. We are looking for further partners to drive the ideas forward for the sake of the community.

2. RELATED WORK
Benchmarking systems provides metrics to estimate system usability regarding the stressed characteristic of the system. By using benchmarks multiple systems can be compared and rated.

Typically, applications are complex and assessing obtained performance is no easy task. New architectures and environments force application users to port their application. This requires to deal with library dependencies on the new system and to optimize the application towards the new system. Applications might reveal inefficiencies in the middleware. Developers can be supported by providing code triggering

the suboptimal behavior. Therefore, benchmarks with reduced complexity help to reveal the bottlenecks.

Microbenchmarks try to reveal performance of the smallest possible component like floating point capability of the system. Measured values might be projected to application performance, however, applications have complex usage patterns of CPU, network and I/O subsystem. Therefore, projecting microbenchmark results to application behavior is non-trivial and often inaccurate.

In contrast, derived from an application or a group of applications, application driven I/O benchmarks allow studies of the architectural system performance under realistic usage patterns i.e. I/O and communication requirements. An example for such a tool is MADbench [4], which emulates a Cosmic Microwave Background data analysis. Emulating the particular I/O behavior, application driven tools can only be used for testing applications with similar I/O requirements.

Synthetic I/O benchmarks measure I/O using system performance standard or customized I/O access patterns. IOZone [7] is a popular I/O benchmark for local and network file systems. It is not used for evaluating HPC applications, since it does not provide a parallel implementation and patterns common in this area. The synthetic tool LLNL IOR [1] supports not only POSIX but also MPI-IO. IOR exercises concurrent read/write on one file or on separate files (one-file-per-processor). The benchmark is highly parameterized and offers a wide variety of access patterns. However, it is difficult to relate the data collected from it back to the original application requirements [4]. Similar to IOR, b_eff_io [2] allows a very precise test configuration by using different parameter groups [10]. Its main purpose is to give a limited statement about I/O performance after a defined time period in which production system is used for testing. Even if b_eff_io is a powerful benchmark, it is challenging to add new access patterns to its framework. Synthetic benchmarks can be used more generally than application driven ones. But there are some disadvantages in this approach. First, it is not easy to relate measured performance back to the real applications. Second, every benchmark provides its own pattern set, such that comparison of obtained results between them is not easy.

No existing benchmark allows to define complex workloads for a specific parallel application. For this reasons there are efforts to develop portable benchmarking tools with adjustable input. The BWT [6] and FileBench [5] are benchmarks where test cases can be modified to resemble workloads. While both programs support threads, FileBench is designed to stress only local file systems. BWT supports parallel I/O commands to a limited extent (process coordination via barrier, implemented using IP multicast), its parameters for executing tests have to be specified in input files and cover the majority of file system access patterns. Even in an early stage of implementation, it provides an approach close to the one we implemented with *Parabench*. As far as we know there is no implementation which allows to trace MPI applications and replay the operation according to the traces. To a limited extent a few researchers implemented workload replays privatly according to their demands, but nobody implemented a full featured MPI trace/replay environment.

Efforts have been made to provide standardized benchmarks, the Standard Performance Evaluation Group (SPEC) provides benchmarks for a wide range of systems. However, mostly the benchmarks assess performance of a single non-parallel system. BenchIT [3, 8] is a project which aims to provide microbenchmarks and kernels from HPC applications, and it allows to compare performance of different systems with the help of a GUI. In contrast to existing solutions we aim for an open community which allows to exchange application patterns. Therefore, we will provide tools to either program, or to record and replay application traces. An easy generation of benchmarks, web-supported analysis and comparison of results will enable the community to evaluate new systems quickly. Furthermore, patterns can be shared with developers of middleware like MPI in order to optimize system and middleware.

3. REFERENCES

[1] ASCI I/O Stress. `http://www.llnl.gov/asci/purple/benchmarks/limited/ior/`.

[2] b_eff_io Benchmark. `https://fs.hlrs.de/projects/par/mpi//b_eff_io/`.

[3] BenchIT. `http://www.benchit.org`.

[4] J. Borrill, L. Oliker, J. Shalf, and H. Shan. Investigation of leading HPC I/O performance using a scientific-application derived benchmark. In *Proc. of SC '07*, pages 1–12. ACM, 2007.

[5] FileBench. `http://www.solarisinternals.com/wiki/index.php/FileBench`.

[6] Filesystem IO Test Program BWT. `http://people.web.psi.ch/stadler_h/`.

[7] IOzone Filesystem Benchmark. `http://www.iozone.org/`.

[8] G. Juckeland, M. Kluge, W. E. Nagel, and S. Pfluger. Performance Analysis with BenchIT: Portable, Flexible, Easy to Use. In *Proc. of the 1th International Conference of Quantitative Evaluation of Systems (QEST '04)*, pages 320–321, Washington, DC, USA, 2004. IEEE Computer Society.

[9] O. Mordvinova, D. Runz, J. M. Kunkel, and T. Ludwig. I/O Performance Evaluation with Parabench – Programmable I/O Benchmark. In *Proc. of the 10th International Conference on Computational Science 2010 (ICCS '10)*, Amsterdam, NL, 2010. to appear.

[10] R. Rabenseifner and A. E. Koniges. Effective file-I/O bandwidth benchmark. In *Proc. of Euro-Par '00*, pages 1273–1283. Springer-Verlag, 2000.