Analyzing Multi-layer I/O Behavior of HPC Applications

Ronny Tschüter
Technische Universität Dresden
ronny.tschueter@tu-dresden.de

Christian Herold
Technische Universität Dresden
christian.herold@tu-dresden.de

Bert Wesarg
Technische Universität Dresden
bert.wesarg@tu-dresden.de

Matthias Weber
Technische Universität Dresden
matthias.weber@tu-dresden.de

ABSTRACT

In this work we present an approach to analyze multi-layer I/O behavior of HPC applications. For a comprehensive analysis it is not sufficient to track I/O operations on a single layer because applications may combine multiple I/O paradigms (e.g., MPI I/O, NetCDF, HDF5). Furthermore, I/O libraries may use another I/O paradigm internally. With our approach I/O activities can be captured at any layer. This work introduces methodologies to intercept calls to I/O libraries and presents an event design to record applications' I/O activities. We implement our approach in the Score-P measurement system and prove its usability with a study of the Met Office/NERC Cloud Model (MONC) code.

KEYWORDS

I/O, performance, tracing, analysis

ACM Reference format:
Ronny Tschüter, Christian Herold, Bert Wesarg, and Matthias Weber. 2017. Analyzing Multi-layer I/O Behavior of HPC Applications. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis - Regular Poster, Denver, Colorado USA, November, 2017 (SC2017),* 2 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Current HPC systems provide powerful storage systems equipped with high bandwidth interconnects and parallel file systems. Highly-scalable applications have to run I/O operations in parallel to leverage available resources. Typical HPC applications do not directly interact with the file system but make use of high-level I/O libraries for reading and writing data. However, internally the operations are mapped to common parallel I/O paradigms such as MPI I/O. In addition, applications might use multiple I/O paradigms in combination. For performance analysis and optimization it is essential to have information from all I/O layers involved to investigate their interaction.

On the one hand, the application can use multiple I/O libraries independently, e.g. calls to NetCDF and Posix I/O functions. On the other hand, I/O libraries can interact transparently. For example, the application calls only NetCDF functions but the NetCDF library uses HDF5 internally.

SC2017, November, 2017, Denver, Colorado USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Darshan [3] is an application-level I/O characterization tool. The relevant I/O behavior of the applications can be captured at production scale with low overhead. However, Darshan focuses on the analysis of MPI applications. Our approach is more generic supporting also sequential, OpenMP, Pthread, and SHMEM codes.

TAU [7] also provides several methods to instrument I/O activities and allows monitoring of multiple layers in the I/O software stack. However, the recorded performance data is stored in an internal data format. Our implementation uses standardized open data formats, e.g. trace data is stored in the Open Trace Format 2 (OTF2) [4]. This enables interoperability with a wide range of analysis tools such as Vampir [6], Scalasca [5], and TAU [7].

3 METHODOLOGY

In our work we enhance the Score-P measurement infrastructure [1]. We implement software components to intercept calls to specific I/O libraries. This wrapping of function calls into a library can be achieved at link-time via the `--wrap` feature of the GNU linker or at run-time using `LD_PRELOAD`. If the application calls one of these wrapped functions, the call is intercepted and the control flow is passed to the Score-P measurement system. The measurement system records performance relevant data and calls the original function. Afterwards the control flow returns to the application and program execution continues.

4 IMPLEMENTATION

In our current work we intercept calls to:

- MPI I/O
- NetCDF
- Parallel NetCDF
- ADIOS
- HDF5
- Posix I/O

Definitions represent the entities used in subsequent I/O operations. At the moment there are definitions for files, directories, and handles. Each time one of these entities is used for the first time a corresponding definition is recorded. Definition attributes provide additional information. For example, the scope attribute of a file records its accessibility (global vs. node-local). Definitions are referenced by events.

Figure 1: Generated event sequence in case of a blocking I/O operation.

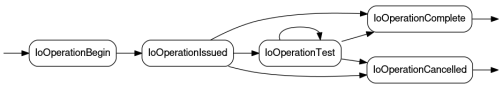

Figure 2: Generated event sequence in case of a non-blocking I/O operation.

Events represent I/O activities during applications runtime. The events can be distinguished into two basic event categories: meta data (e.g., open/create, close) and data transfer operations (e.g., read/write). Depending on the event type additional information is recorded. As an example for data transfer operations, such as read/write file access, the following data is recorded:

- Time when the event happened
- Affected I/O file reference
- Mode of operation
- Special semantic of this operation
- Bytes transferred

An individual I/O operation might be split into basic operations while recording. For example, a Posix `fread` operation is recorded as two events—one for the begin of the operation and one for the completion (Figure 1). Figure 2 illustrates the event sequence in case of a non-blocking I/O operation.

5 CASE STUDY

In order to evaluate the usability of our approach we present results from the analysis of the Met Office/NERC Cloud Model (MONC) code [2] with Vampir. MONC uses separate I/O server processes to decouple simulation progress and I/O activities. In our experiments 15 simulation processes were connected to one I/O server process. Simulation processes forward their data to the I/O server process. The communication is handled via MPI messages. Figure 3 illustrates the callstack of an I/O server process. The I/O server process writes data to disk using NetCDF routines. Multi-layer I/O analysis reveals that some NetCDF routines transparently invoke MPI communication and split the data transfer into multiple Posix I/O function calls (Figure 4). The recorded trace data contains fine-granular information for each event enabling in-depth application analysis and generation of event statistics.

6 CONCLUSION

Our work enables user not only to investigate the I/O behavior of their applications but also to analyze the interaction of multiple I/O libraries in use. The Score-P measurement system is capable of both profiling and tracing. The profiling mode records aggregated information providing hints to hot spots of the application. In contrast, the tracing mode records all individual events allowing in-depth analysis of the application's dynamic runtime behavior. Inefficiency patterns in the usage of I/O routines can be identified and are the fundament for application tuning.

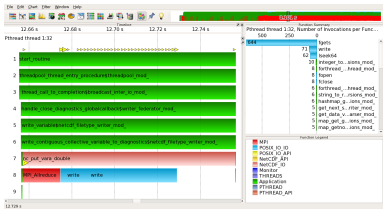

Figure 3: The I/O server process writes data to disk using NetCDF routines. The chart on the left side shows the call-stack of an I/O server process. The top left chart presents statistics about the function calls issued by the I/O server process within the selected time interval.

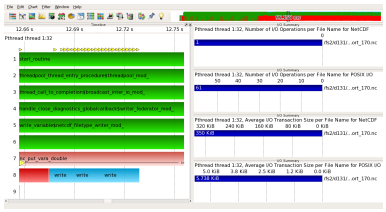

Figure 4: Some NetCDF routines transparently invoke MPI communication and split the data transfer into multiple Posix I/O function calls.

7 FUTURE WORK

The next Score-P release will contain the I/O recording functionality presented in this work.

8 ACKNOWLEDGEMENT

This research was undertaken as part of the NEXTGenIO project, which is funded through the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 671951.

REFERENCES

[1] An Mey, D. et al. 2012. Score-P: A Unified Performance Measurement System for Petascale Applications. In *Competence in High Performance Computing 2010.*

[2] Nick Brown, Michele Weiland, Adrian Hill, Ben Shipway, Chris Maynard, Thomas Allen, and Mike Rezny. 2015. A Highly Scalable Met Office NERC Cloud Model. In *Proceedings of the 3rd International Conference on Exascale Applications and Software (EASC '15)*. University of Edinburgh, Edinburgh, Scotland, UK, 132–137. http://dl.acm.org/citation.cfm?id=2820083.2820108

[3] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley. 2009. 24/7 Characterization of petascale I/O workloads. In *2009 IEEE International Conference on Cluster Computing and Workshops*. 1–10. https://doi.org/10.1109/CLUSTR.2009.5289150

[4] Dominic Eschweiler, Michael Wagner, Markus Geimer, Andreas Knüpfer, Wolfgang E. Nagel, and Felix Gerd Eugen Wolf. 2012. Open Trace Format 2 - The Next Generation of Scalable Trace Formats and Support Libraries. In *Applications, Tools and Techniques on the Road to Exascale Computing: proceedings of the 14th biennial ParCo conference ; ParCo2011 (Advances in Parallel Computing)*, Vol. 22. IOS Press, Amsterdam [u.a.], 481–490.

[5] Markus Geimer, Felix Wolf, Brian J. N. Wylie, Erika Ábrahám, Daniel Becker, and Bernd Mohr. 2010. The Scalasca Performance Toolset Architecture. *Concurr. Comp. : Pract. Exper.* 22, 6 (April 2010), 702–719. https://doi.org/10.1002/cpe.v22:6

[6] Knüpfer, A. et al. 2008. The Vampir Performance Analysis Tool-Set. In *"Tools for High Performance Computing", Proceedings of the 2nd International Workshop on Parallel Tools for High Performance Computing*. Springer-Verlag.

[7] Sameer Shende, Allen D Malony, Wyatt Spear, and Karen Schuchardt. 2011. Characterizing I/O Performance Using the TAU Performance System. In *PARCO*. 647–655.