TECHNISCHE UNIVERSITÄT DRESDEN

Centre for Information Services and High Performance Computing (ZIH)

Analyzing Multi-layer I/O Behavior of HPC Applications

Motivation

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.

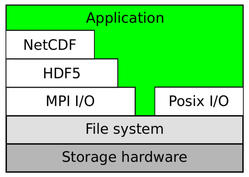

Figure 1: Example of multiple I/O paradigms used by an application. On the one hand, the application can use multiple I/O libraries independently, e.g. calls to NetCDF and Posix I/O functions. On the other hand, I/O libraries can interact transparently. For example, the application calls only NetCDF functions but the NetCDF library uses HDF5 internally.

Approach

In our work we enhance the Score-P measurement infrastructure. 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.

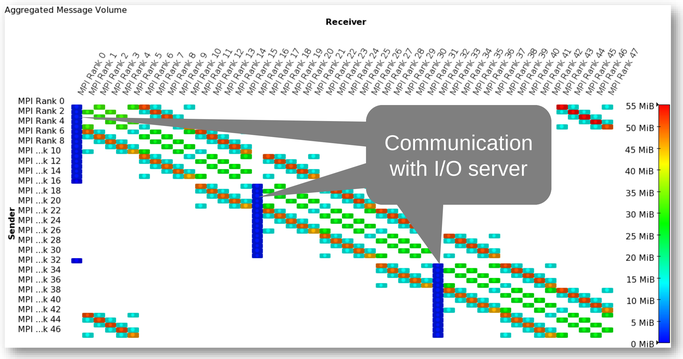

Figure 5: MONC uses separate I/O server processes to decouple simulation progress and I/O activities. As shown in the MPI communication matrix 15 simulation processes are connected to one I/O server process.

Recent Work

In our current work we intercept calls to:

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

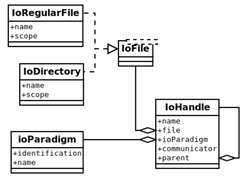

Figure 2: Structure of I/O entity definitions.

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.

Events represent I/O activities during application 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, see Figure 3. Figure 4 illustrates the event sequence in case of a non-blocking I/O operation.

Figure 3: Generated event sequence in case of a blocking I/O operation.

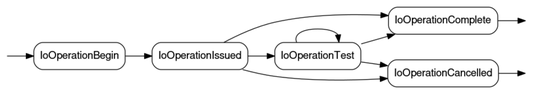

Figure 4: Generated event sequence in case of a non-blocking I/O operation.

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. Our work provides the fundament for application tuning by recording inefficiency patterns in the usage of I/O routines. Thereby, allowing users to identify performance bottlenecks. The analysis of the Met Office/NERC Cloud Model (MONC) code in Vampir (Figure 5 to 8) showcases the value of this work.

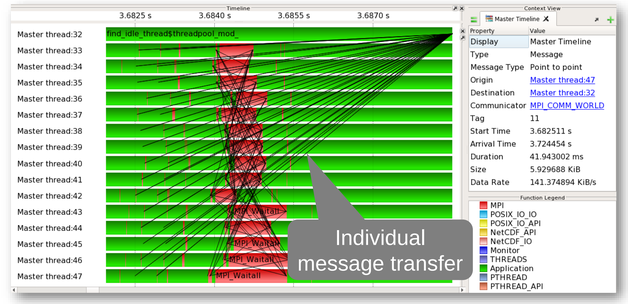

Figure 6: Simulation processes forward their data to the I/O server process. The communication is handled via MPI messages shown as bold lines.

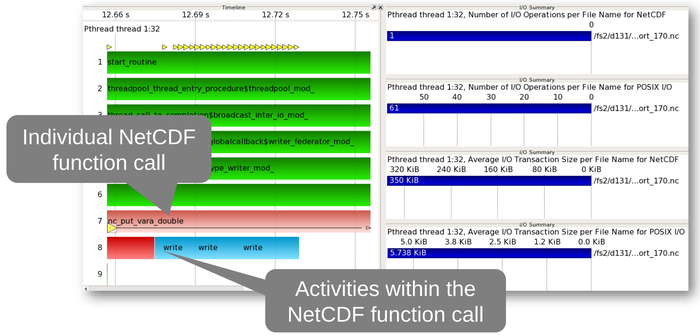

Figure 7: The I/O server process writes data to disk using NetCDF routines. Multi-layer I/O analysis reveals that some NetCDF routines transparently invoke MPI communication and split the data transfer into multiple Posix I/O function calls.

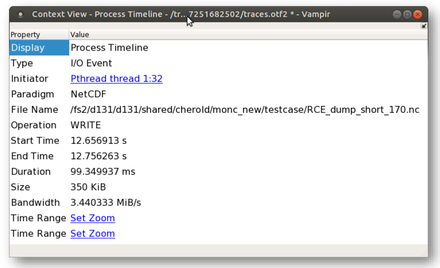

Figure 8: The recorded trace data contains fine-granular information for each event enabling in-depth application analysis.

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.

Ronny Tschüter (ronny.tschueter@tu-dresden.de), Christian Herold (christian.herold@tu-dresden.de)
Bert Wesarg (bert.wesarg@tu-dresden.de) , Matthias Weber (matthias.weber@tu-dresden.de)
Technische Universität Dresden, ZIH
Germany

nextgenio

ZIH Center for Information Services & High Performance Computing

DRESDEN concept Exzellenz aus Wissenschaft und Kultur