Hessisches Kompetenzzentrum
für Hochleistungsrechnen

Performance Monitoring for Future Exascale Systems

Project Manager
Dr. Ahmad Tarraf

Principal Investigator
Dr. Ahmad Tarraf

Project Term
2021 - 2022

Clusters
Lichtenberg Cluster Darmstadt

Institute
Parallel Programming

University
Technische Universität Darmstadt

Introduction

Typical HPC applications represent simulations of large scientific problems. Usually, these applications alternate between computational and input/output (I/O) phases. For applications using the Message parsing interface (MPI), these I/O operations are usually done synchronously. That is, the application is blocked until the I/O operation completes. An alternative to synchronous I/O is asynchronous I/O. Here the application performs the I/O in the background of the computational phases. While asynchronous I/O outperforms synchronous I/O in many aspects, there has been little shift toward using it. This can be partially traced back to the fact that it is difficult to state if the shift is worth it, as the application can as well be blocked by asynchronous I/O, e.g., when the computational phases are too short. In this work, we analyse the I/O requirements an application must fulfil in order to utilize asynchronous I/O effectively.

Methods

In order to analyse the I/O requirements of an HPC application that uses asynchronous MPI-IO, several metrics must first be collected. This includes the number of bytes transferred as well as the start and end times of I/O operations. By dividing these values over each other the throughput of the application is found. In the next step, the available time window for the I/O operation is determined, such that the I/O operation happens completely in the background of the compute phase. By dividing the same number of bytes transferred over this time window, the required bandwidth is found. Hence, whenever the

throughput is larger than the required bandwidth, the MPI rank is not blocked by the I/O operation. Finally, the collected metrics are generalized at the application level to yield a metric for a single job configuration (i.e., number of ranks). Our methodology has been realized as a C++ library. Using the LD_PRELOAD mechanism, our library can be attached to any HPC application (C++). After the execution completes, the traced data is saved to a *.json file. By repeating this step with different configurations (number or ranks), the scaling behaviour of the application can be analysed.

Results

Using our methodology, the I/O requirements of HPC applications utilizing asynchronous I/O can be determined. This allows for effectively utilizing the system by boosting the performance of the application. The most suitable configuration can be found, such that the application spends as less as possible time waiting I/O operations to finish. Moreover, our approach is not limited to selecting the configuration, but can also be used to determine the best application settings (MPI-IO writing method, problem size, etc.) such that the I/O does not block the application.

Discussion

Till now, we only identified suitable metrics and analysed their scaling behaviour in regard to the number of ranks and problem size. In the next steps, we intend to generate performance models using the tool Extra-P to model the I/O requirements scaling behaviour. Moreover, we intend to investigate more on identifying I/O phases. Our approach was also just applied to applications utilizing asynchronous MPI-IO. We intend to extend our approach for application that use synchronous MPI-IO as well as propose a similar methodology for systems equipped with burst buffers.

Last Update: 2023-03-08 17:21