Hessisches Kompetenzzentrum
für Hochleistungsrechnen

Compiler-Assisted Optimization and Modernization of MPI

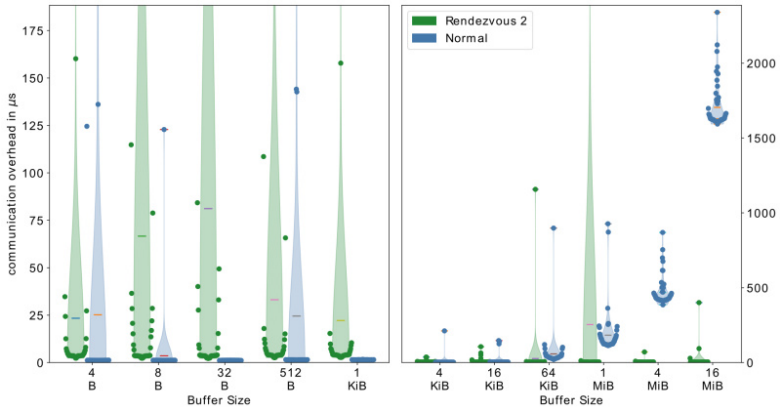

Figure 1: Communication overhead for varying message buffer sizes for two processes. The Y axis depicts the communication overhead that can not be overlapped by computation, while the X axis shows different message buffer sizes. Normal means the existing implementation. Rendezvous 2 depicts our improved protocol without tag matching.

Project Manager
Dr. Tim Jammer

Researchers
Tobias Dollenbacher

Principal Investigator
Prof. Dr. Christian Bischof

Project Term
2022 - 2023

Clusters
Lichtenberg II Cluster Darmstadt

Additional Software
Self-deveolped software: https://github.com/tudasc, LLVM, OpenMPI

Institute
Department of Computer Science

University
Technische Universität Darmstadt

JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN
TECHNISCHE UNIVERSITÄT DARMSTADT
UNIKASSEL VERSITÄT
Philipps Universität Marburg

Introduction

In the realm of High-Performance Computing (HPC), the Message Passing Interface (MPI) has established itself as a cornerstone for developing applications that harness the immense power of large computing clusters. MPI's ubiquity in this field is a testament to its capability to enable parallel programming, thereby facilitating the execution of complex computations across distributed systems. Over the years, MPI has undergone a series of advancements as its standard evolves, introducing novel features that hold the promise of unlocking new dimensions of parallelism and performance.
While the evolution of the MPI standard brings exciting opportunities, there exists a prevailing challenge within the HPC community. Despite the emergence of innovative features, their adoption remains limited, and their potential benefits often go untapped. A significant factor contributing to this phenomenon is the inherent complexity associated with incorporating these novel features into programming paradigms. As a result, many developers shy away from utilizing these advanced attributes, thereby missing out on potential performance gains.
The core predicament stemming from underutilization of these novel features is a two-fold issue. On one hand, if these features are left untapped by a majority of the HPC community, the incentive for efficient implementation diminishes. Developers and researchers might be less inclined to invest effort in optimizing these features if their adoption is not widespread. On the other hand, without well-optimized implementations, the potential benefits of these novel MPI features cannot be fully realized, perpetuating a cycle of underutilization.
This project embarks on a mission to address this challenge by investigating strategies to facilitate the effective utilization of new MPI features in programming large HPC clusters. Our primary goal is to bridge the gap between the capabilities of the evolving MPI standard and their practical incorporation into applications.

Methods

In pursuit of our overarching goal to enhance the utilization of novel MPI features for efficient programming on High-Performance Computing (HPC) clusters, we have devised a comprehensive approach that encompasses three primary areas of focus:

1. **Improving Correctness checking for Usability.** A key hurdle in the adoption of new MPI features is the complexity of ensuring their correct usage within parallel applications. To tackle this challenge, we have concentrated our efforts on enhancing the correctness checking mechanisms. By refining and expanding the capabilities of existing MPI correctness checking tools, we aim to provide developers with reliable means to validate the proper integration of novel features. This will not only bolster the confidence of developers in utilizing these features but also reduce the potential for subtle and hard-to-diagnose errors that might arise from their misuse. However, a critical observation is the lack of support for these newer features within existing MPI correctness checking tools. To address this limitation, we have

undertaken the task of evolving the MPI-Corrbench correctness benchmark suite (https://github.com/tudasc/MPI-Corrbench). This will facilitate that tools correctness checking tools capabilities can be expanded to a more complete feature set of MPI.

2. **Compiler-Aided Automation of Feature Incorporation**. The process of incorporating new MPI features into existing codebases can be intricate and time-consuming. To streamline this process, we are exploring the integration of compiler assistance. By leveraging compiler capabilities, we aim to automate and simplify the integration of novel features into parallel applications. This approach not only reduces the burden on developers but also promotes consistent and accurate utilization of advanced MPI features, further encouraging their adoption.
3. **Demonstrating Performance Gains through Efficient Implementation.** A vital incentive for the adoption of novel MPI features is the potential for improved performance. To showcase this potential, we are dedicated to developing more efficient implementations of these features. By optimizing their utilization and execution, we aim to highlight the tangible benefits that developers can attain by embracing advanced MPI attributes. This demonstration of enhanced performance serves as a catalyst, motivating developers to explore and integrate these features more extensively.

Results

As one key result to showcase, we find that the compiler can do some required matching work for persistent MPI operations. As persistent MPI requests can be used multiple times, the compiler can, in some cases, prove that message matching is only needed for the first occurrence and can be entirely skipped for subsequent instances. In our project, we developed the required compiler analysis, as well as an implementation of a communication scheme that skips the message envelope matching and directly transfers the data via RDMA instead. This allows us to substantially reduce the communication overhead that cannot be overlapped with computation. Using the Intel IMB-ASYNC Benchmark, we can observe a communication overhead reduction of up to 95 percent for larger message sizes.

Discussion

As part of our efforts to automate feature incorporation, we successfully developed a Proof of Concept implementation of a compiler analysis. This analysis transforms standard MPI operations into more efficient Partitioned ones.
By leveraging compiler knowledge and optimizations, we demonstrated the potential to significantly reduce communication overhead. In some cases, the use of persistent MPI operations led to an impressive overhead reduction of up to 95%.
We incorporated more complex test cases into the MPI-Corrbench correctness benchmark suite that target specific

JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN
TECHNISCHE UNIVERSITÄT DARMSTADT
UNIKASSEL VERSITÄT
Philipps Universität Marburg

aspects of novel MPI features. By doing so, we have facilitated the development and refinement of more capable correctness tools.

Publications

Jammer, T; Bischof, C. Compiler-enabled optimization of persistent MPI Operations. In: 2022 IEEE/ACM International Workshop on Exascale MPI (ExaMPI), Dallas, TX, USA, pp. 1-10, (2022) https://doi.org/10.1109/ExaMPI56604.2022.00006

Jammer, T; Huck, A; Lehr, J-P; Protze, J; Schwitanski, S; Bischof, C. Towards a Hybrid MPI Correctness Benchmark Suite. In: EuroMPI/USA'22: Proceedings of the 29th European MPI Users' Group Meeting, pp. 46-56, ACM, 29th European MPI Users' Group Meeting, hattanooga, USA (2022) http://dx.doi.org/10.1145/3555819.3555853

(to be published soon) Hück, A; Jammer, T; Protze, J; Bischof, C. Investigating the Usage of MPI at Argument-Granularity in HPC Codes. In: ACM, Proceedings of EuroMPI2023: the 30th European MPI Users' Group Meeting (EUROMPI '23), September 11--13, Bristol, United Kingdom (2023)

Reference

Jammer, T; Bischof, C. Automatic Partitioning of MPI Operations in MPI+OpenMP Applications. In: Jagode H., Anzt H., Ltaief H., Luszczek P. (eds) High Performance Computing. ISC High Performance 2021. Lecture Notes in Computer Science, vol 12761. Springer, Cham. (2021) https://doi.org/10.1007/978-3-030-90539-2_12

Last Update: 2023-08-10 19:25