Hessisches Kompetenzzentrum
für Hochleistungsrechnen

Automatic Code Parallelization

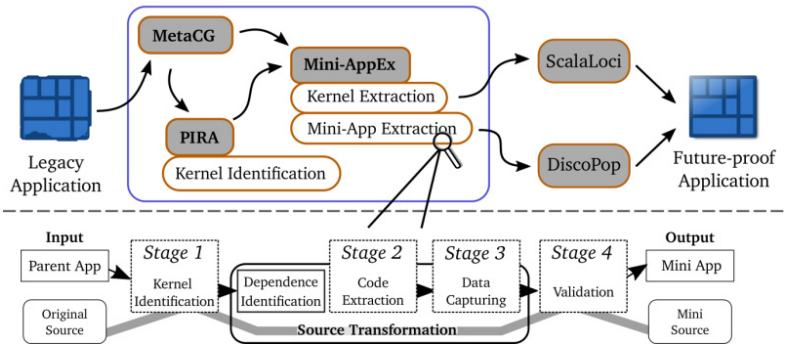

Figure 1: The SF4.0 workflow with a focus on the mini-app extraction approach.

Introduction

The Software-Factory 4.0 (SF4.0) is a 4-year LOEWE1 project funded by the German State of Hesse and involves several research groups at Technical University of Darmstadt. SF4.0 is concerned with the adaptation of legacy software due to changed requirements and technical advances. The main focus is on three topics: more flexible software systems in the context of the application area ”Industrie 4.0”, the parallelization of existing software in the context of ”High Performance Computing” (HPC), and simplification of the re-engineering in both areas. In this Lichtenberg project, the automatic identification of application kernels and their extraction into a mini-app. Furthermore, a subsequent automatic parallelization of the mini-app is performed.

Methods

We develop and apply the workflow that is envisioned by the Software-Factory 4.0 project, see www.sf40.de, and depicted in Figure 2: (1) a mini-app is created from an existing, sequential target application and evaluated for its representativeness of the parent application. (2) The mini-app is used to apply the tool DiscoPoP [2] to discover potential parallelism, and generate OpenMP suggestions how to parallelize the target. (3) The suggestions are implemented and evaluated for their respective correctness and speed up. As another approach for the parallelization in step (2), the identified kernels are re-composed using the reactive multitier language ScalaLoci [3] to construct a scalable and fault tolerant distributed application.
For the initial Kernel Identification, PIRA [1] is developed and used, which combines a static and dynamic program analysis. The list of its automatically identified kernels is passed to a source-to-source translator, which performs the source extraction. As part of this source translation, the application data is captured using a newly developed type-safe checkpoint-restart library. Finally, the resulting mini-app can be used to perform experiments with the tool DiscoPoP to discover so-far

Project Manager
Jan-Patrick Lehr

Researchers
Daniel Sokolowski, Florian Dewald, Mohammad Norouzi, Usman Ahmad, Janis Mittelstädt, Peter Arzt, Jonas Rickert and Moritz Fischer

Principal Investigator
Prof. Dr. Christian Bischof

Project Term
2020 - 2021

Clusters
Lichtenberg Cluster Darmstadt

Additional Software
Clang, LLVM, TypeART, Score-P, Caliper, PAPI, openBLAS, GSL v2.5

Institute
Institute of Scientific Computing

University
Technische Universität Darmstadt

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

unexploited parallelism.

Results

As part of the kernel identification step, reducing the influence of the measurement system is crucial to obtaining a valid picture of the target application's execution. Hence, we develop the tool PIRA as open-source software and available at https://github.com/tudasc/pira to automatically reduce this influence by limiting the measurements to only relevant regions. Moreover, we have investigated the influence of the measurement system and PIRA's capability to reduce this influence on hardware performance counters.
For example, PIRA allows to automatically detect load imbalances in parallel applications – a source for significant efficiency loss for largely-parallel applications. The approach if able to correctly identify the load imbalance present in the target applications for both applications we investigated. The approach is also used to identify kernel regions that are suitable for subsequent extraction as kernel. Such kernels can then be tied together using high-level languages such as ScalaLoci to enable hybrid on-premise and in-the-cloud execution.
Finally, we developed a tool-supported approach to automatically extract these kernels together with all their dependencies to result in an executable mini-app of the original application.

Discussion

Our methods show good results in terms of their ability to reduce measurement overhead, and to identify load imbalances. The kernel identification worked well in the test cases evaluated. However, some target applications still exceed an acceptable level of overhead, due to their code structure and the challenges that instrumentation-based measurements face in such situations. The extraction approach is promising and works on a significantly large code base of ≈ 8.5 million lines of code. However, the source-to-source translator is still a research prototype.

Publications

Lehr JP., Jammer T., Bischof C.: "MPI-CorrBench: Towards an MPI Correctness Benchmark Suite", In: HPDC '21: Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing, ACM, 2021 https://doi.org/10. 1145/3431379.3460652

Lehr JP., Bischof C., Dewald F., Mantel H., Norouzi M., and Wolf F.: "Tool-Supported Mini-App Extraction to Facilitate Program Analysis and Parallelization", In: 50th International Conference on Parallel Processing, ACM, 2021 https://doi.org/10.1145/3472456.3472521

Arzt P., Fischler Y., Lehr JP., Bischof C.: "Automatic Low-Overhead Load-Imbalance Detection in MPI Applications", In: Sousa L., Roma N., Tomás P. (eds) Euro-Par 2021: Parallel Processing. Euro-Par 2021. Lecture Notes in Computer Science, vol 12820. Springer, Cham, 2021 https://doi.org/10.1007/978-3-030-85665-6_2

Sokolowski D., Lehr JP., Bischof C., Salvaneschi G.: "Leveraging Hybrid Cloud HPC with Multitier Reactive Programming", In: SuperCompCloud 2020: 3rd Workshop on Interoperability of Supercomputing and Cloud Technologies, IEEE, 2020 https://doi.org/10.1109/SuperCompCloud51944.2020.00010

Lehr JP., Hück A., Fischler Y., Bischof C.: "MetaCG : Annotated call-graphs to facilitate whole-program analysis", In: 11th ACM SIGPLAN International Workshop on Tools for Automatic Program Analysis, ACM, 2020 https://doi.org/10.1145/3427764.3428320

Lehr JP., Hück A., Fischer M., Bischof C.: "Compiler-Assisted Type-Safe Checkpointing", In: Jagode H., Anzt H., Juckeland G., Ltaief H. (eds) High Performance Computing. ISC High Performance 2020. Lecture Notes in Computer Science, vol 12321. Springer, Cham. https://doi.org/10.1007/978-3-030-59851-8_1

Reference

[1] Lehr JP., Hück A. and Bischof C.: "PIRA: performance instrumentation refinement automation", In: Proceedings of the 5th ACM SIGPLAN International Workshop on Artificial Intelligence and Empirical Methods for Software Engineering and Parallel Computing Systems (AI-SEPS 2018). Association for Computing Machinery, New York, NY, USA, 1–10, 2018 https://doi.org/10.1145/3281070.3281071

[2] Li Z., Atre R., Huda Z., Jannesari A. and Wolf F.: "Unveiling parallelization opportunities in sequential programs", In: Journal of Systems and Software, Volume 117, 2016, Pages 282-295, ISSN 0164-1212, 2016 https://doi.org/10.1016/j.jss.2016.03.045

[3] Weisenburger P., Köhler M. and Salvaneschi G.: "Distributed system development with ScalaLoci. Proc. ACM Program. Lang. 2, OOPSLA, Article 129 (November 2018), 30 pages, 2018 https://doi.org/10.1145/3276499

Last Update: 2023-03-16 01:27