ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Technische Informatik
und Kommunikationsnetze
Computer Engineering and
Networks Laboratory

Prof. L. Thiele

Semester / Master Thesis:

Automated program analysis for predicting memory access collisions

What is the memory access behaviour of a thread? Use LLVM, an open source compiler toolchain, to analyse that! Then predict how it will conflict with other threads.

Modern multi-core platforms often have shared memory, in many cases even organised in hierarchies of different memory levels. The cores use this memory for storing their private data, but also for communicating with each other. If multiple cores try to access the same memory bank, a so-called *access conflict* occurs. Only one core is served at a time, the others have to wait. Clearly, this can have a significant impact on the execution time of the tasks running on the different cores.

If it was possible to predict how frequently each task accesses each memory bank, that would allow a better prediction of the task execution times. Also, one could use this information to organise the task management: Perhaps, multiple tasks with heavy memory access do not need to run at the same time, or they could use different memory banks.

On the other hand, even tasks with frequent memory accesses do not necessarily cause a high number of conflicts, e.g. if they have regular access patterns that do not interfere. Information on these memory access patterns might give further insights on this.

There exists an open-source compiler toolchain called LLVM, which implements many state-of-the-art code analysis and optimisation techniques. It works on a layer between the programming language and the target hardware assembly code, the so-called *intermediate representation*. It can easily be interfaced and has been used for many different program analysis techniques.

Task: The task in this work will be to extend LLVM such that it can analyse functions on how often they access certain variables, and maybe also to extract certain access patterns. With this information, a probabilistic prediction method for task execution times shall be devised. The following steps will be part of this process:

1. Understand the concepts of LLVM.
2. Extend LLVM (write a *pass*) to quantify accesses of given functions to given variables.
3. Devise (with the help of the advisors) probabilistic models and prediction mechanisms on how concurrent tasks may influence the execution time of each other.
4. Compare the different prediction mechanisms.

Requirements: You should feel comfortable with programming C++ and with basic probability theory. Knowledge about compiler optimisation techniques would be an asset, but is not required.

Interested? Please have a look at http://www.tec.ethz.ch/research.html **and contact us for more details!**

Contacts

- Andreas Tretter: andreas.tretter@tik.ee.ethz.ch, ETZ G81
- Stefan Draskovic: stefan.draskovic@tik.ee.ethz.ch, ETZ G81
- Lothar Thiele: thiele@ethz.ch, ETZ G87

Further Reading

LLVM: http://www.llvm.org