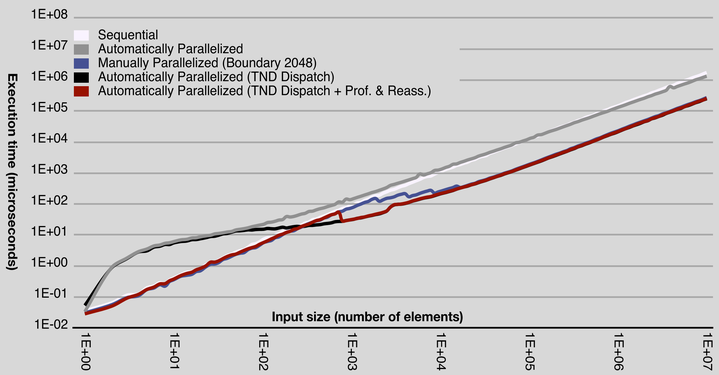
Sambamba

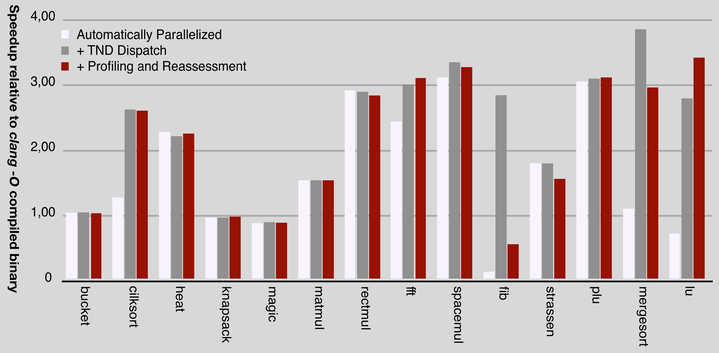
http://www.sambamba.org

Runtime Adaptive Parallelization

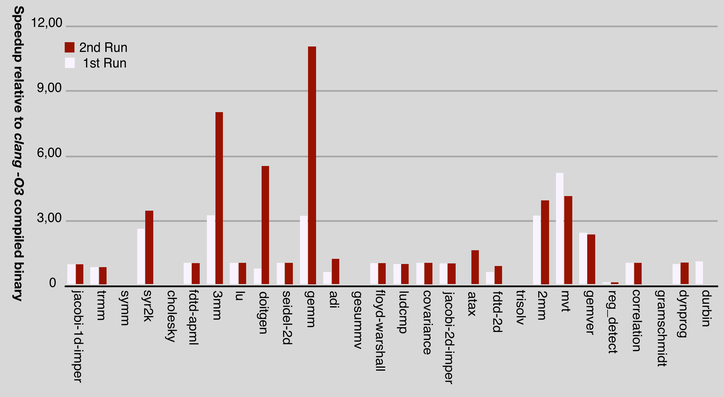
Parallelized Mergesort: Comparison of parallelization schemes on different input sizes.



Cilk Example Programs: Comparison of parallelization schemes on different Cilk example programs. Experiments conducted on a quad-core with Hyperthreading.



PolyBench: Effect of dynamic polyhedral optimization with SPolly on programs from PolyBench. Experiments have been conducted on an octa-core with Hyperthreading.



Input

Sambamba takes as input a program in LLVM IR.
It is developed and tested for C and C++ applications.

1

Program Analysis

One of the most important analyses used in Sambamba so far is a modified version of the DataStructureAnalysis (DSA). DSA is used to determine data dependences for the construction of a block-level program dependence graph (PDG) used for parallelization.

2

Candidate Selection

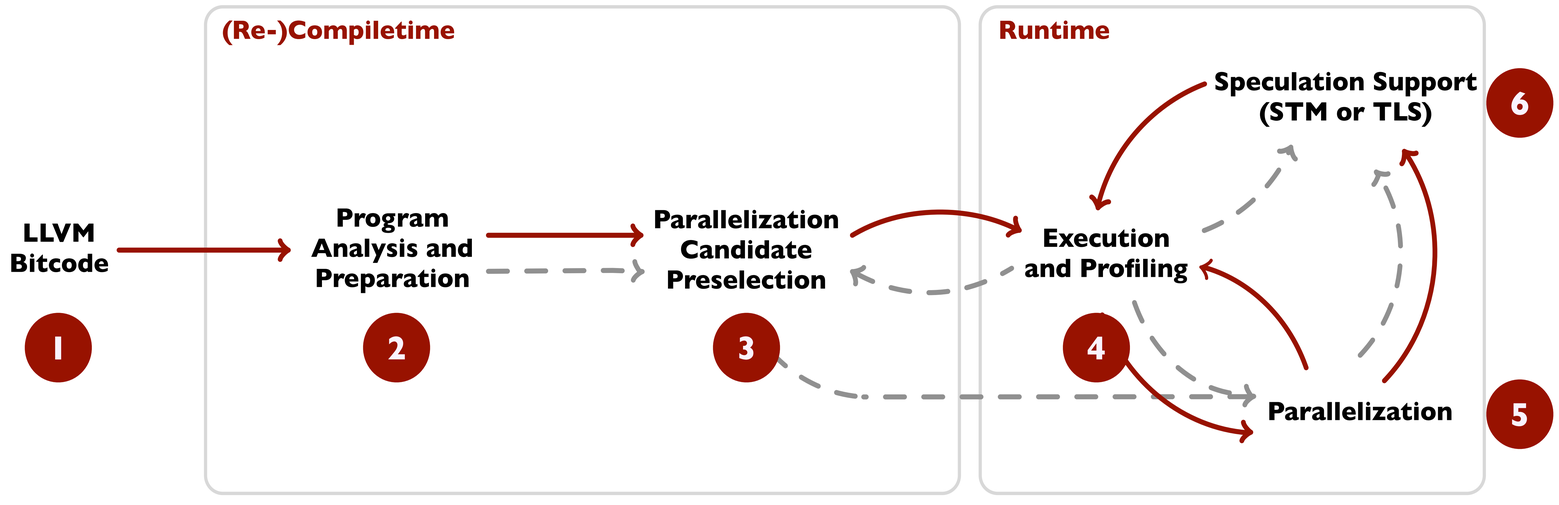
Currently, two parallelization methods are implemented:

3

One method to select and schedule parallelization candidates employs an ILP solver to find optimal parallel schedules with respect to the cost-model. The resulting task-parallelism is not tied to loops, or even special forms thereof.

Another approach tries to widen the applicability of program optimization (and parallelization) in the polyhedral model. Its candidates are code regions (sSCoPs) which are rejected by Polly due to the limitations of the polyhedral representation.

(Re-)Compiletime



Execution and Profiling

All statically gathered information is packed, together with the Sambamba runtime system, into a fat binary, ready for standalone execution.
The runtime system allows to selectively enable branch- and executiontime profiling for individual functions and even callsites.
Not only the original, but also newly installed versions are tracked.

4

Parallelization

Statically found parallelization candidates are combined to find the best combination of parallelization decisions for the situation at hand. The executing hardware as well as runtime profiles are taken into account.
Depending on the parallelization decisions, Intel TBB (for task parallelism) or OpenMP (e.g. used by Polly) are used for parallel execution.

5

Speculation

Sambamba currently implements protection of speculatively parallelized code by a state-of-the-art software transactional memory (STM) with commit orders. An implementation of thread-level-speculation (TLS) based on process forking is currently being implemented.

Based on relevant profiling information like execution time, data size and misspeculation rate of previous versions, the speculation mechanism of Sambamba is able to automatically select the most efficient speculation method (STM or TLS) and its corresponding parameters (size of transaction log, method of conflict detection, ...).
These can be adjusted individually for each parallelized function.

6

Runtime

UNIVERSITÄT DES SAARLANDES

Compiler Design Lab

Software Engineering Chair

Sambamba: A Runtime System for Online Adaptive Parallelization K. Streit, C. Hammacher, A. Zeller, S. Hack. Conf. on Compiler Construction (CC), 2012
Sambamba: Runtime Adaptive Parallel Execution K. Streit, C. Hammacher, A. Zeller, S. Hack. WS on Adaptive Self-tuning Computing Systems (ADAPT), 2013.
SPolly: Speculative Optimizations in the Polyhedral Model J. Doerfert, C. Hammacher, K. Streit, S. Hack. WS. on Polyhedral Compilation Techniques (IMPACT), 2013.

Clemens Hammacher <hammacher@cs.uni-saarland.de>
Kevin Streit <streit@cs.uni-saarland.de>
Sebastian Hack <hack@cs.uni-saarland.de>
Andreas Zeller <zeller@cs.uni-saarland.de>