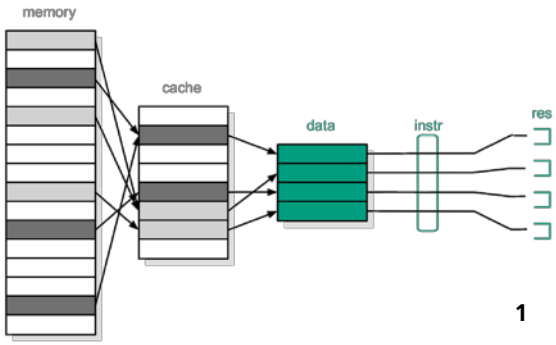
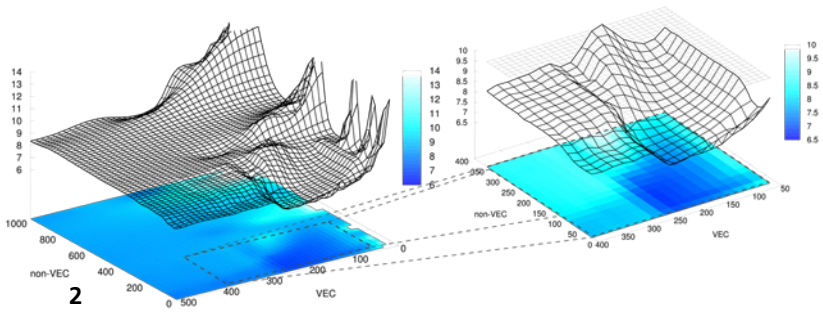
Fraunhofer
SCAI

FRAUNHOFER INSTITUTE FOR ALGORITHMS AND SCIENTIFIC COMPUTING SCAI



1



2

1 *Data-access hierarchy*
2 *Optimal parameter region for cache-aware code transformation*

(SEMI-)AUTOMATIC CODE OPTIMIZATION AND PARALLELIZATION

Fraunhofer SCAI has a longterm working experience in massively parallel computing and the corresponding optimization. The HPC group at Fraunhofer SCAI is developing mechanisms to extract the maximal potential of a given hardware through (semi-)automatic code transformation and parallelization techniques. We are working on tools to optimize a user-given source code for specific hardware architectures, either on a source-2-source base or within an entire compiler toolchain. The objects of interest for such transformations predominantly are compute intensive nested loops. Our tools parallelize the given code (e.g. by suitable transforming it) and setting the appropriate pragmas for the compiler or the runtime library. Our implementations are partially based on prestigious code transformators to exploit their full potential. To accomplish optimized performance for x86 CPUs through a (semi-) automatic process, we developed an adaptive, code- and hardware-related transformation process based on cache and SIMD-unit aware tile-size-selection strategies. With our SICA extension within PluTo (PluTo-SICA) it is possible to generate an optimized code for concurrent execution on SIMD units (e.g. SSE or AVX) and multiple cores (through OpenMP). Therefore, the framework automatically inserts corresponding pragmas for both levels of parallelism into the generated code. The optimizations implemented in PLUTO-SICA rely on the polyhedral model and force an optimized load of data through the cache hierarchy to provide the vector units with fast accessible data. Our approach can thus exploit the full potential of modern CPUs regarding memory access behavior and multi-level parallelism. PLUTO-SICA is the parallelization base of the ENHANCE project (www.enhance-project.de) and available OpenSource under LGPL2 license. We are working on further automatic optimization strategies for several parallel architectures like GPU or IntelMIC as well as for various correspond-

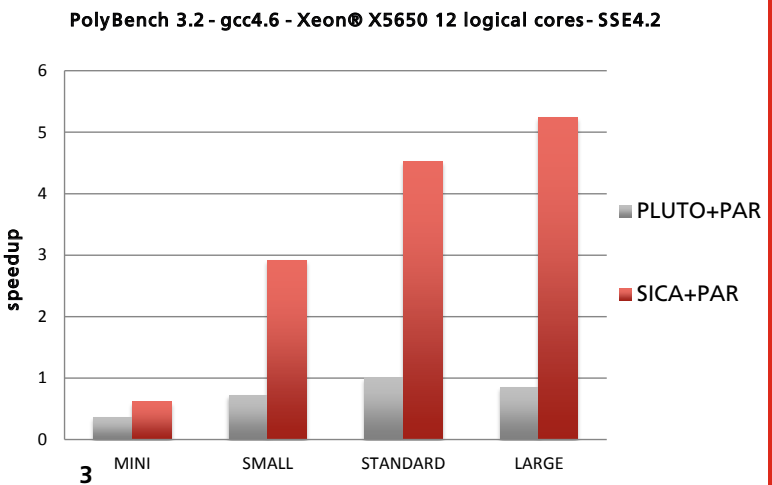
Fraunhofer Institute for Algorithms and Scientific Computing SCAI

Schloss Birlinghoven 1
53757 Sankt Augustin

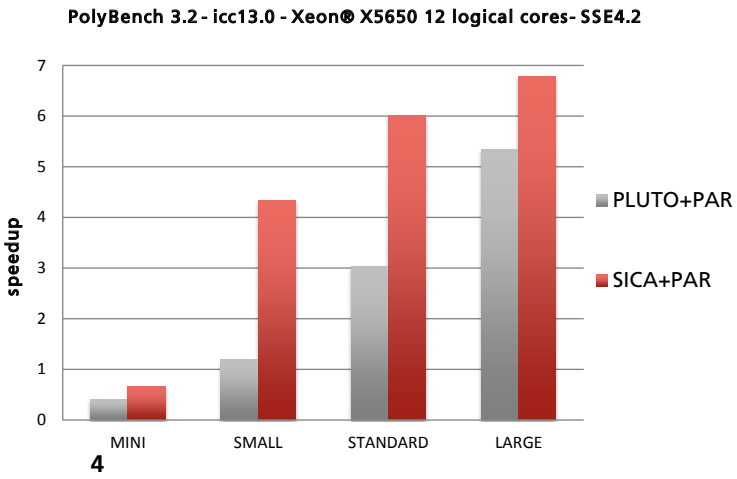
Contact:

Dr. Thomas Soddemann
Phone +49 2241 14-4076
thomas.soddemann@
scai.fraunhofer.de

www.scai.fraunhofer.de



3



4

ing memory layouts. Furthermore, we are working preliminary on optimizations for FPGA based architecture layouts.
Due to our experiences in the development of optimizations as well as other projects (e.g. The Fraunhofer SCAI Library for Accelerated Math Applications – LAMA) and industrial assignments, we provide expertise in the field of parallelization and optimization.

Performance

Our transformation frameworks rely on the polyhedral model and therefore require to set up a valid polyhedral representation of the codes' hot-spots. This mainly restricts the prospected code regions to nested loops with affine array accesses.
To benchmark the performance, there is a unified benchmark-suite for polyhedral optimizers, the PolyBench (http://www.cse.ohio-state.edu/~pouchet/software/polybench/). Our optimization approach leads to speedups throughout the different fields of applications and leverages great performance by code- and hardware-specific automatic tiling strategies that support the cache use and the concurrent execution on parallel operating calculation units. Our approach leads to an optimal and comprehensive vectorization for many codes. It reduces cache-misses and increases the vectorization rate.
We achieve notable performance gains by our transformations for a variety of codes and inputs and for several compilers (like the GNU compiler gcc or the Intel® compiler icc).

Where To Get It

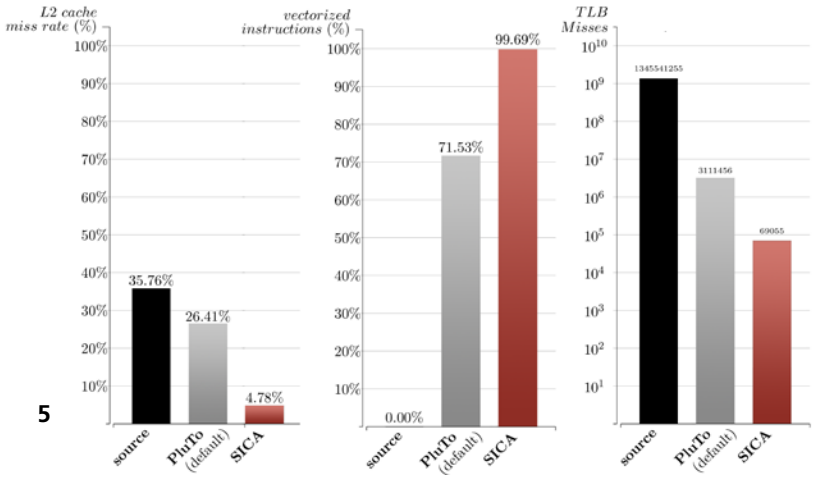
Our SIMD- and cache optimization approach PLUTO-SICA is included in the polyhedral optimizer PLUTO and therefore can be downloaded from the official PLUTO git repository (http://repo.or.cz/w/pluto.git).

How To Use It

PLUTO-SICA can be used for source-2-source optimizations as well as in a library based version natively within the LLVM compiler toolchain. You are able (but not obliged) to specifically mark the region to be optimized by two simple pragmas within the code or to use an automatic polyhedral-scope detector from Polly in LLVM. Feel free to contact us for more details!

Questions and Suggestions?

Besides our x86 optimizations, please ask us for specific purpose solutions. If your architecture captures unused parallel potential, we can look for solutions to use them in a user-friendly way.
Please contact us for any feedback!



5

3+4 *Average speedups from PolyBench for the different benchmark input size constellations with the GNU compiler gcc (3) and the Intel® compiler icc (4). The bars show the performance benefit of a static tiling technique (like the default PLUTO implementation) as well as the speedup from our dynamic and adaptive PLUTO-SICA approach. The codes were automatically tiled and parallelized.*

5 *Performance Counter measurements for PLUTO and PLUTO-SICA and (for comparison) the original code.*