Paper : **Iterative Compilation in a Non-Linear Optimisation Space**

Objective : reduce the search space using profile feedback

Compiler : compiler framework developed to optimise multimedia codes for embedded systems

CPU : UltraSparc, R10000,Pentium Pro, and TriMedia -1000

Method/algorithm : search algorithm that tries to find the best performance within the fewest number of evaluations

Benchmarks/programs : Matrix multiplication

Optimizations : Loop Unrolling 1-20, Tiling 1 -100, Padding 1-10

Comparison : compare to non-optimized version

Metrics: compilation time/ execution time 🡪 simulator.

**Paper :** Evaluating Iterative Compilation

a platform independent optimisation approach based on feedback-directed program restructuring.

Objective : examines a feedback assisted approach based on traversing an optimisation space.

Compiler : native compiler

CPU : six different platforms : Alpha 21164, Alpha 21264, Pentium II, Pentium III, HP-PA 9000/712, Ultrasparc

Method/algorithm : two strategies that search the optimisation space by means of profiling to find the best possible program variant. These strategies have no a priori knowledge of the target machine and can be run on any platform

Benchmarks/programs : 3 full SPEC benchmarks (Tomcatv, Swim, Mgrid)

Optimizations : Padding, Loop Unrolling, Loop Tiling

Comparison : to native compiler with full optimisation/ Compaq compiler with the optimisation level set to -O5

Metrics: execution time

**Paper : Black box approach for selecting optimization options using budget-limited genetic algorithms**

Objective : The ESTO framework described here searches the option set space using various types of genetic algorithms, ultimately determining the option set that maximizes the performance of the given application and workload. ESTO regards the compiler as a black box, specified by its external-visible optimization options

Compiler : GCC, XLC and FDPR-Pro

CPU : Linux machines with Power architecture/ 4-processor 1.5GHz Power5 running SUSE Linux Enterprise Server 9 operating system

Method/algorithm : the application of GA for the problem of selecting an optimal option set for a specific application and workload. **using budget-limited genetic algorithms** that reduce the population size exponentially

Benchmarks/programs : SPEC2000 benchmark suite

Optimizations : 60 optimization options (binary and range (parameterized).)

Comparison : over GCC -O1 and -O3 and XLC -O3 and FDPR-Pro -O3

Metrics: execution time

**Paper :** Efficient Program Compilation through Machine Learning Techniques

Objective : reduce the compile time while presearving the code quality

Compiler : IBM’s TPO (Toronto Portable Optimizer).

CPU : IBM server with 4 x IBM Power5TM 1900 MHz processors and 32GB of memory, running the IBM AIXR 5.3 operating system

Method/algorithm : ML (logistic regression)

Benchmarks/programs : 21 benchmarks from SPEC2000 and benchmarks from IBM customers

Optimizations : 24 optimizations included at the -qhot -O3 level

Comparison : -qhot -O3

Metrics: decrease compile time while preserving the execution time.

**Paper : Milepost gcc: Machine learning enabled self-tuning compiler**

Objective : modular, extensible, self-tuning optimization infrastructure to automatically learn the best optimizations across multiple programs and architectures based on the correlation between program features, run-time behavior and optimizations

Compiler : GCC

CPU : – a cluster of 16 AMD Opteron 2218, 2.6 GHz, 4GB main memory, 2MB L2 cache, running Debian Linux Sid x64 with kernel 2.6.28.1

Intel – a cluster of 16 Intel Xeon EM64T, 3 GHz, 2GB main memory, 1MB

L2 cache, running Debian Linux Sid x64 with kernel 2.6.28.1

ARC – FPGA implementation of the ARC 725D reconfigurable processor, 200MHz, 32 KB L1 cache, running Linux ARC with kernel 2.4.29

Method/algorithm : Our machine learning models predict the best GCC optimization to apply to an input program based on its program structure or program features. two machine learning techniques to select combinations of optimization passes based on construction of a probabilistic model or a transductive model on a set of M training programs, and then use these models to predict “good” combinations of optimization passes for unseen programs based on their features.

Benchmarks/programs : MiBench/cBench benchmark

Optimizations : 100 flags in O1 –O3

Comparison : over O3 and random search

Metrics: compilation time/execution time/code size with **multi objective**

**Paper :** Multi-Objective Exploration of Compiler Optimizations for Real-Time Systems

Objective : WCET-aware compiler framework for an automatic search of compiler optimization sequences which yield highly optimized code.

Compiler : compiler WCC

CPU : Intel Quad-Core Xeon 2.4 GHz machine

Method/algorithm : Indicator Based Evolutionary Algorithm (IBEA)/ Nondominated Sorting Genetic Algorithm 2 (NSGA-II) [16], and Strength Pareto Evolutionary Algorithm 2 (SPEA-2)

Benchmarks/programs : 35 DSPstone, MediaBench, MiBench, UTDSP, NetBench, and MRTC WCET Benchmarks

Optimizations : 30 source code optimizations

Comparison : to 3 multi objective algo and o3 O2 O1

Metrics: WCET↔ACET and WCET↔Code Size

**Paper : Automatic Tuning of Compiler Optimizations and Analysis of their Impact**

Objective : it has the tools for analysis of tuning results that allow to identify compiler optimizations that contribute the most to the improvement, supports parallel compilation and execution on several devices to speedup the tuning, and is capable of performing multi-objective optimization to evolve Pareto-front of optimal configurations.

Compiler : GCC 4.8

CPU : ARM architecture (ARM Cortex-A9)

Method/algorithm : multiple optimization objectives SPEA2 and genetic algorithm

Benchmarks/programs : w popular open-source applications C-Ray, Crafty Chess, libevas (part of Enlightenment Foundation Libraries), SciMark, x264 and zlib

Optimizations : 200 options and parameters

Comparison : to O2 O3 and Os

Metrics: multi objective performance and code size (or compile time). And also mono – objective

**Paper :** Performance Potential of Optimization Phase Selection During Dynamic JIT Compilation

Objective : (a) analyze and understand the phase selection related behavior of optimization phases in a production-quality JVM, (b) determine the steady-state performance potential of optimization selection, and (c) evaluate the effectiveness of existing feature-vector based heuristic techniques in achieving this performance potential and suggest improvements

Compiler : Sun/Oracle’sHotSpot JIT compiler and HotSpot JVM

CPU : n a cluster of Dell PowerEdge 1850 server machines running Red Hat Enterprise Linux 5.1 as the operating system. Each machine has four 64-bit 2.8GHz Intel Xeon processors, 6GB of DDR2 SDRAM, and a 4MB L2 cache

Method/algorithm : t long-running (genetic algorithm based) iterative searches to determine the ideal benefits of phase selection in online JIT environments

Benchmarks/programs : SPECjvm98 benchmarks and 12 (of 14) applications from the DaCapo benchmark suite

Optimizations : 28 optimizations

Comparison : baseline the default HotSpot server compilation sequence

Metrics: ideal steady-state performance gains

**Paper : Finding effective compilation sequences**

**Problem they adress : local optimal, exploring the search space**

Objective :

Compiler :

CPU : SPARC architecture and petium and mac(intel)

Method/algorithm : Greedy Constructive Algorithms and GA and Hill Climbing

Benchmarks/programs :

Optimizations :

Comparison :

Metrics:

**Paper :** COLE: Compiler Optimization Level Exploration

Objective : COLE , a framework for automatically finding Pareto optimal optimization levels through multi-objective evolutionary searching.

Compiler : GCC 4.1.2

CPU : Intel Pentium 4 Prescott 3.0 GHz processor based machine running Fedora Linux (Core 4).

Method/algorithm :

Benchmarks/programs : SPEC CPU benchmarks

Optimizations : 60 flags involved in Os O1 O2 O3

Comparison : to (-Os, -O1, -O2 and -O3 and SPEA2 and random search

Metrics: Multi objective compilation and execution time

**Paper :**

**Problem :** exercise iterative optimization on a few tens of data sets.

Objective : find best optimization combination accross all data sets

Compiler : e Intel ICC compiler v11.1 and GCC 4.4.1

CPU : 3GHz Intel Xeon dualcore processors (E3110 family), 2GB RAM, 2×3MB L2 cache. We use the CentOS 5.3 Linux distribution based on Red Hat with kernel 2.6.18

Method/algorithm : Random search

Benchmarks/programs : KDataSets versus MiDataSets with MiBench benchmarks

Optimizations : 53 optimization flags for ICC and 132 flags for GCC

Comparison : Ofast – O3

Metrics: performace

**Paper :**

Objective :

Compiler :

CPU :

Method/algorithm :

Benchmarks/programs :

Optimizations :

Comparison :

Metrics: