tetracom

Technology Transfer in Computing Systems

D3.13: Individual TTP13 abstract

Project no.:	609491
Funding scheme:	Collaborative project
Start date of the project:	1st September 2013
Duration:	36 months
Work programme topic:	FP7-ICT-2013-10
Deliverable type:	Report
Deliverable reference number:	ICT-609491 / D3.13
WP and tasks contributing:	WP 3 / all
Due date:	30/04/2015
Actual submission date:	30/06/2015
Responsible Organization:	CTUNING
Dissemination Level:	Public
Revision:	1.0

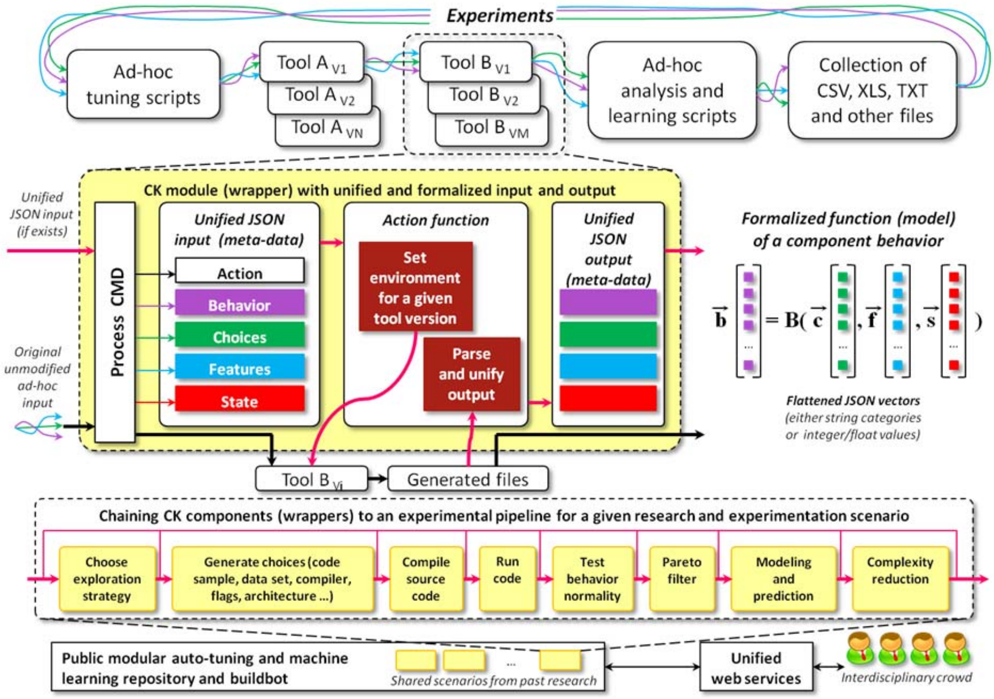
TETRACOM D3.13: Collective Mind for ARM (collaborative, systematic and reproducible benchmarking and optimization of computer systems)

Grigori Fursin (cTuning foundation), Anton Lokhmotov and Ed Plowman (ARM)

Designing, modeling and benchmarking of computer systems in terms of performance, power consumption, size, reliability and other characteristics is becoming extraordinary complex and costly. This is due to a large and continuously growing number of available design and optimization choices, lack of common performance analysis and optimization methodology, and lack of common ways to create, preserve and reuse vast design and optimization knowledge. As a result, optimal characteristics are achieved only for a few ad-hoc benchmarks while often leaving real-world applications underperforming. Eventually, these problems lead to a dramatic increase in the development, optimization and maintenance costs, increasing time to market for new products, eroding return on investment (ROI), and slowing down innovation in computer engineering.

ARM is the UK based and world's leading semiconductor intellectual property (IP) supplier and as such is at the heart of the development of digital electronic products. Over 60 billion ARM based chips have been shipped to date in the world. Some of ARM's most tedious, time consuming and ad-hoc tasks include benchmarking and optimization of the new processor designs. The purpose of this TTP is to investigate how cTuning foundation's open-source technology including the latest BSD-licensed Collective Knowledge Framework (2nd version of Collective Mind framework) can solve some of the above problems via

- unification, automation and crowdsourcing of performance analysis, optimization and run-time adaptation of continuously shared and realistic programs and data sets from the community across new ARM heterogeneous architectures using LLVM compiler in terms of performance, power consumption, code size and any other exposed characteristic;
- automatic, machine-learning based, multi-objective optimization (performance, energy, accuracy, size, faults) of various realistic user applications instead of outdated benchmarks;
- improvement of the efficiency of the internal R&D process by decreasing fragmentation of development, benchmarking and optimization efforts;
- reduction of the time to market for the new processors and increase return on investment.


Experiments
Ad-hoc tuning scripts
Tool A V1
Tool A V2
Tool A VN
Tool B V1
Tool B V2
Tool B VM
Ad-hoc analysis and learning scripts
Collection of CSV, XLS, TXT and other files
CK module (wrapper) with unified and formalized input and output
Unified JSON input (if exists)
Original unmodified ad-hoc input
Process CMD
Unified JSON input (meta-data)
Action
Behavior
Choices
Features
State
Action function
Set environment for a given tool version
Parse and unify output
Unified JSON output (meta-data)
Tool B Vi
Generated files
Formalized function (model) of a component behavior
b = B(c , f , s)
Flattened JSON vectors (either string categories or integer/float values)
Chaining CK components (wrappers) to an experimental pipeline for a given research and experimentation scenario
Choose exploration strategy
Generate choices (code sample, data set, compiler, flags, architecture ...)
Compile source code
Run code
Test behavior normality
Pareto filter
Modeling and prediction
Complexity reduction
Public modular auto-tuning and machine learning repository and buildbot
Shared scenarios from past research
Unified web services
Interdisciplinary crowd

Collective Knowledge Framework (*http://github.com/ctuning/ck*) allows engineers gradually implement light-weight wrappers around any software piece (benchmarks or realistic application) with more than one implementation or optimization choice available. These wrappers are connected with Collective Knowledge repository (JSON-based web service with Hadoop-based database) to continuously monitor all important characteristics of these pieces (treated as computational species) across numerous hardware configurations (mobile devices, architecture simulators in a cloud, etc) together with randomly selected optimizations.

Similar to natural sciences, we can now continuously track all winning solutions (optimizations for a given hardware such as compiler flags, OpenCL/CUDA/OpenMP/MPI/skeleton parameters, number of threads and any other exposed by users) that minimize all costs of a computation (execution time, energy spent, inaccuracy, code size, failures, memory and storage footprint, optimization time, contentions, and so on) of a given species on a Pareto frontier along with any unexpected behavior. Furthermore, engineers can work with data scientists to continuously classify solutions, prune redundant ones, and correlate them with various features of software, its inputs (data sets) and used hardware either manually (similar to Wikipedia) or using available "big data" predictive analytics and machine learning techniques.

Finally, CK also help create a realistic, large, diverse, distributed, representative, and continuously evolving benchmark with related optimization knowledge while gradually covering all possible software and hardware to be able to predict best optimizations and improve compilers depending on usage scenarios and requirements. Such continuously growing collective knowledge accessible via simple web service then becomes an integral part of the practical software and hardware co-design of self-tuning computer systems!

Further information is available in the following publications (related to this TTP):

- *http://arxiv.org/pdf/1506.06256v1.pdf*
- *http://cknowledge.org/repo/web.php?wcid=29db2248aba45e59:cd11e3a188574d80*
- *http://cknowledge.org/repo/web.php?wcid=report:b0779e2a64c22907*

In this TTP, we have successfully applied our Collective Knowledge framework to perform systematic analysis, data mining and online/offline learning on vast amounts of benchmarking data available at ARM.

Our technology showed good potential to automatically find various important correlations between numerous in-house benchmarks, data sets, hardware, performance, energy and run-time state. Such correlations can, in turn, help derive representative benchmarks and data sets, quickly detect unexpected behavior, suggest how to improve architectures and compilers, and speed up machine-learning based multi-objective autotuning.

For example, on a realistic OpenCL application (KFusion from SLAMBench) we could increase the performance by tenfold (10x) at the same tracking accuracy. This performance increase results in better than real-time performance on several mobile platforms; by reducing the GPU frequency by 30%, we can reduce energy consumption further by a similar amount, while still meeting real-time constraints.

Furthermore, our technology has also showed potential to enable collaborative and reproducible experimentation within and across workgroups.

Finally, our positive results have motivated us to establish a UK-based startup called dividiti (*http://dividiti.com*) to accelerate computer engineering and research by further developing our technology and applying it to real-world problems.