TACLeBench – Title is TBD*

Heiko Falk[1], Martin Schoeberl[n], and Add yourself in alphabetical order, but keep Heiko first.[2]

1 **Dummy University Computing Laboratory**
Address, Country
open@dummyuni.org
2 **Department of Informatics, Dummy College**
Address, Country
access@dummycollege.org

Abstract

We need benchmarks that fit the WCET and embedded community needs,

1998 ACM Subject Classification Dummy classification – please refer to `http://www.acm.org/about/class/ccs98-html`

Keywords and phrases Dummy keywords – please provide 1–5 keywords

Digital Object Identifier 10.4230/OASIcs.xxx.yyy.p

1 Notes

TODO: Just internal notes to collect bullet points on ideas.

- On issue with original benchmarks optimized away - we can now check with Patmos easily
- Working on getting the license correct - issue with code from unknown source
- Useful in general for embedded systems/barebone systems where less libraries are available
- Usage of the benchmarks (version, no subsetting, no source change)
- How to contribute

2 Introduction

TODO: A brief introduction what the paper is about. It shall include briefly the main contributions and findings. The contributions can be bullet listed.

This paper... *TODO: purpose statement, latest in 4th paragraph*

TODO: We are aiming on self-contained benchmarks, that do not need any operating system services, including file IO. Therefore,...

TODO: Talk about the TACLe COST action and the subgroup around the benchmark collection. And plans how to further develop and maintain the collection.

TODO: Talk about usage of the benchmarks: WCET tools, compiler, hardware architecture,...

The first version of TACLeBench (version 1.0, available from[1]), which was produced by Heiko Falk, was a collection of 102 benchmark programs from several different research

* This work was partially supported TACLe. *add the right phrase*

[1] `http://www.tacle.eu/index.php/activities/taclebench`

Conference/workshop/symposium title on which this volume is based on.
Editors: Martin Schoeberl; pp. 1–4

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

groups. We keep this first version tagged with "V1.0" in the public GitHub repository.[2] The version described in this paper is version 2.0 and tagged as such in the repository.

The contributions of this paper are: (1) ... (2) ...

This paper is organized in N sections: The following section presents related work. Section 3 presents related work. Section 4 presents the benchmark collection, its classification, and the updates to make them useful. Section 5 evaluates the benchmark collection on... Section 6 concludes.

3 Related Work

TODO: Show that you know the field. All related work shall be put into context or contrast to our current work.

The Mälardalen WCET benchmarks is the first collection of programs especially intended for benchmarking WCET analysis tools [**?**]. This collection of C programs was collected from several sources in 2005 and was used in many WCET research projects and for the WCET Tool Challenge 2006. A subset of the Mälardalen benchmarks has even been translated to Java [**?**]. Most benchmarks are relative small, except two C programs that have been generated from tools. The benchmarks also contain all input data. As this input data is fixed and some programs do not provide any return value, good compilers with optimization turned on can optimize most of the code away. We include most of the benchmarks from the Mälardalen WCET benchmark suite in TACLeBench. However, we changed the way input data is represented in variables (make them `volatile`) and make the return of `main` dependent on the benchmark calculation. Furthermore, we dropped benchmarks where this licensing terms are unknown or even disallow distributing the source.

EEMBC

MiBench [**?**] We include some of the MiBench benchmarks, especially those where it was possible to include the input data with the C source.

Debie [**?**]

PapaBench [**?**]

General purpose benchmarking, with a focus on open source and embedded systems.

JemBench.

4 The Benchmark Collection

4.1 Benchmark Sources

TODO: what did we collect. How many indirections in collections have there be? E.g., adpcm.c: (1) SOURCE : C Algorithms for Real-Time DSP by P. M. Embree, (2) SNU-RT Benchmark Suite, (3) MDH, and (4) TACLeBench. How have benchmarks changed? Legal changes?

4.2 Classification

TODO: Talk about kernel, medium size, and application benchmarks and their usage.

- Static analysis
- Measurement based analysis
- Schedulability analysis using taskset generator

[2] `https://github.com/tacle/tacle-bench`

4.3 Issues with the Original Sources

TODO: Show the issues with code snippets

- Compiler optimization
- byte order
- low-level access to device registers
- Some benchmarks, which are available for download in the Internet have a license that does not allow open-source distribution. We removed those benchmarks from the original source.

4.4 Benchmark Changes

- unique names for functions and global variables
- Sometimes movements of stack allocated variables into global
- splitting of initialization and computation code into two functions
- Loop bounds added

4.5 Usage Recommandations

TODO: Mmh shall we have this?

TODO: can we include a paragraph about the taskset generator for schedulability analysis

- Use a stable version. At the time of this writing we should have V 2.0
- Do not edit the benchmarks
- Use all benchmarks in your evaluation. Don't introduce a bias in your evaluation by *selecting the most representative benchmarks* for your improvement

4.6 Licenses

TODO: What we have. The issue with getting the right ones.

5 Evaluation

TODO: Computer engineering is a constructive science. We build stuff and we measure. Therefore, there shall always be an evaluation section.

Possible Options for the Evaluation:

- Compile for an embedded processor and measure execution time
- Compile for Patmos and use pasim for execution cycles
- Show how compiler optimizations optimized away the original code and how we fixed this
- Complexity numbers with bar charts on execution time and maybe tools
- Use some WCET tools

6 Conclusion

TODO: Rephrase what this paper is about and list the main contributions and results.

Acknowledgements *TODO: TACLe COST action phrase*

We want to thank . . . for helping in restructuring the benchmarks. We want to thank Bendikt Huber for porting the Lift benchmark from Java to C.

TODO: Ack Niclas for contributing DEBIE in open-source.

A Source Access and Compiling the Benchmarks