8th International Workshop on Worst-Case Execution Time Analysis

WCET 2008, July 1, 2008, Prague, Czech Republic

Edited by

Raimund Kirner

OASIcs – Vol. 8 – WCET 2008 www.dagstuhl.de/oasics

Editor

Raimund Kirner
Real Time Systems Group
Department of Computer Engineering
Vienna University of Technology
Treitlstraße 1–3/182/1
1040 Wien, Austria
raimund@vmars.tuwien.ac.at

ACM Classification 1998
C.4 Performance of Systems, D.2.4 Software/Program Verification

ISBN 978-3-939897-10-1

Published online and open access by
Schloss Dagstuhl – Leibniz-Center for Informatics GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany.

Publication date
November, 2008.

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/OASIcs.WCET.2008.i

ISBN 978-3-939897-10-1 **ISSN 2190-6807** **http://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 2190-6807

www.dagstuhl.de/oasics

2008 WCET Abstracts Collection
8th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis

Raimund Kirner

Universität Wien, AT
raimund@vmars.tuwien.ac.at

Abstract. The workshop on Worst-Case Execution Time Analysis is a satellite event to the annual Euromicro Conference on Real-Time Systems. It brings together people that are interested in all aspects of timing analysis for real-time systems. In the 2008 edition, 13 papers were presented, organized into four sessions: methods for WCET computation, low-level analysis, system-level analysis and flow-analysis. The workshop was also the opportunity to report from the 2007 WCET tool challenge.

Keywords. Worst-case execution time, real-time systems, timing analysis

2008 WCET Report – Proceedings of the 8th Intl. Workshop on Worst-Case Execution Time Analysis (WCET'08)

Following the successful WCET Tool Challenge in 2006, the second event in this series was organized in 2008, again with support from the ARTIST2 Network of Excellence. The WCET Tool Challenge 2008 (WCC'08) provides benchmark programs and poses a number of "analysis problems" about the dynamic, run-time properties of these programs. The participants are challenged to solve these problems with their program analysis tools. Two kinds of problems are defined: WCET problems, which ask for bounds on the execution time of chosen parts (subprograms) of the benchmarks, under given constraints on input data; and flow-analysis problems, which ask for bounds on the number of times certain parts of the benchmark can be executed, again under some constraints. We describe the organization of WCC'08, the benchmark programs, the participating tools, and the general results, successes, and failures. Most participants found WCC'08 to be a useful test of their tools. Unlike the 2006 Challenge, the WCC'08 participants include several tools for the same target (ARM7, LPC2138), and tools that combine measurements and static analysis, as well as pure static-analysis tools.

ECRTS 2008
8th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis
http://drops.dagstuhl.de/opus/volltexte/2008/1671

Joint work of: Holsti, Niklas; Gustafsson, Jan; Bernat, Guillem; Ballabriga, Clément; Bonenfant, Armelle; Bourgade, Roman; Cassé, Hugues; Cordes, Daniel; Kadlec, Albrecht; Kirner, Raimund; Knoop, Jens; Lokuciejewski, Paul; Merriam, Nicholas; de Michiel, Marianne; Prantl, Adrian; Rieder, Bernhard; Rochange, Christine; Sainrat, Pascal; Schordan, Markus

Keywords: WCET analysis, benchmark

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1663

Towards an Automatic Parametric WCET Analysis

Bygde, Stefan; Lisper, Björn

Static WCET analysis obtains a safe estimation of the WCET of a program. The timing behaviour of a program depends in many cases on input, and an analysis could take advantage of this information to produce a formula in input variables as estimation of the WCET, rather than a constant. A method to do this was suggested in [12]. We have implemented a working prototype of the method to evaluate its feasibility in practice. We show how to reduce complexity of the method and how to simplify parts of it to make it practical for implementation. The prototype implementation indicates that the method presented in [12] successfully can be implemented for a simple imperative language, mostly by using existing libraries.

Keywords: WCET, Flow Analysis, Parametric, Symbolic

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1659

Improving the WCET computation time by IPET using control flow graph partitioning

Ballabriga, Clément; Cassé, Hugues

Implicit Path Enumeration Technique (IPET) is currently largely used to compute Worst Case Execution Time (WCET) by modeling control flow and architecture using integer linear programming (ILP). As precise architecture effects requires a lot of constraints, the super-linear complexity of the ILP solver makes computation times bigger and bigger. In this paper, we propose to split the control flow of the program into smaller parts where a local WCET can be computed faster - as the resulting ILP system is smaller - and to combine these local results to get the overall WCET without loss of precision. The experimentation in our tool OTAWA with lp_solve solver has shown an average computation improvement of 6.5 times.

Keywords: Static analysis, SESE regions, ILP

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1670

Towards Predicated WCET Analysis

Marref, Amine; Bernat, Guillem

In this paper, we propose the use of constraint logic programming as a way of modeling context-sensitive execution-times of program segments. The context-sensitive constraints are collected automatically through static analysis or measurements. We achieve considerable tightness in comparison to traditional calculation methods that exceeded 20% in some cases during evaluation. The use of constraint-logic programming in our calculations proves to be the right choice when compared to the exponential behaviour recorded by the use of integer linear-programming.

Keywords: WCET Analysis, Implicit-Path Enumeration-Technique, Constraint-Logic Programming, Static Analysis

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1667

INFER: Interactive Timing Profiles based on Bayesian Networks

Zolda, Michael

We propose an approach for timing analysis of software-based embedded computer systems that builds on the established probabilistic framework of Bayesian networks. We envision an approach where we take (1) an abstract description of the control flow within a piece of software, and (2) a set of run-time traces, which are combined into a Bayesian network that can be seen as an interactive timing profile. The obtained profile can be used by the embedded systems engineer not only to obtain a probabilistic estimate of the WCET, but also to run interactive timing simulations, or to automatically identify software configurations that are likely to evoke noteworthy timing behavior, like, e.g., high variances of execution times, and which are therefore candidates for further inspection.

Keywords: Bayesian networks, embedded systems, hardware modeling, measurement-based execution time analysis, software modeling, probabilistic modeling, profilin

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1669

Towards a Common WCET Annotation Language: Essential Ingredients

Kirner, Raimund; Kadlec, Albrecht; Prantl, Adrian; Schordan, Markus; Knoop, Jens

Within the last years, ambitions towards the definition of common interfaces and the development of open frameworks have increased the efficiency of research on WCET analysis.

The Annotation Language Challenge for WCET analysis has been proposed in line with these ambitions in order to push the development of common interfaces also to the level of annotation languages, which are crucial for the power of WCET analysis tools. In this paper we present a list of essential ingredients for a common WCET annotation language. The selected ingredients comprise a number of features available in different WCET analysis tools and add several new concepts we consider important. The annotation concepts are described in an abstract format that can be instantiated at different representation levels.

Keywords: WCET, worst-case execution time, hard real-time, embedded systems, abstract interpretation, pipeline analysis, cache analysis, symbolic state traversal BDD

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1657

Computing time as a program variable: a way around infeasible paths

Holsti, Niklas

Conditional branches connect the values of program variables with the execution paths and thus with the execution times, including the worst-case execution time (WCET). Flow analysis aims to discover this connection and represent it as loop bounds and other path constraints. Usually, a specific analysis of the dependencies between branch conditions and assignments to variables creates some representation of the feasible paths, for example as IPET execution-count constraints, from which a WCET bound is calculated. This paper explores another approach that uses a more direct connection between variable values and execution time. The execution time is modeled as a program variable. An analysis of the dependencies between variables, including the execution-time variable, gives a WCET bound that excludes many infeasible paths. Examples show that the approach often works, in principle. It remains to be seen if it is scalable to real programs.

Keywords: WCET, flow analysis, infeasible paths, dependency analysis

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1660

Merging Techniques for Faster Derivation of WCET Flow Information using Abstract Execution

Gustafsson, Jan; Ermedahl, Andreas

Static Worst-Case Execution Time (WCET) analysis derives upper bounds for the execution times of programs. Such bounds are crucial when designing and verifying real-time systems.

A key component in static WCET analysis is to derive flow information, such as loop bounds and infeasible paths. We have previously introduced abstract execution (AE), a method capable of deriving very precise flow information. This paper present different merging techniques that can be used by AE for trading analysis time for flow information precision. It also presents a new technique, ordered merging, which may radically shorten AE analysis times, especially when analyzing large programs with many possible input variable values.

Keywords: Worst-Case Execution Time (WCET) analysis, flow analysis

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1658

On Composable System Timing, Task Timing, and WCET Analysis

Puschner, Peter; Schoeberl, Martin

The complexity of hardware and software architectures used in today's embedded systems make a hierarchical, composable timing analysis impossible. This paper describes the source of this complexity in terms of mechanisms and side effects that determine variations in the timing of single tasks and entire applications. Based on these observations, the paper proposes strategies to reduce the complexity. It shows the positive effects of these strategies on the timing of tasks and on WCET analysis.

Keywords: Real-time systems, timing analysis, WCET analysis, predictable timing, composability

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1662

WCET Analysis for Preemptive Scheduling

Altmeyer, Sebastian; Gebhard, Gernot

Hard real-time systems induce strict constraints on the timing of the task set. Validation of these timing constraints is thus a major challenge during the design of such a system. Whereas the derivation of timing guarantees must already be considered complex if tasks are running to completion, it gets even more complex if tasks are scheduled preemptively – especially due to caches, deployed to improve the average performance. In this paper we propose a new method to compute valid upper bounds on a task's worst case execution time (WCET). Our method approximates an optimal memory layout such that the set of possibly evicted cache-entries during preemption is minimized. This set then delivers information to bound the execution time of tasks under preemption in an adopted WCET analysis.

Keywords: WCET, Preemption

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1664

Applying WCET Analysis at Architectural Level

Gilles, Olivier; Hugues, Jérôme

Real-Time embedded systems must enforce strict timing constraints. In this context, achieving precise Worst Case Execution Time is a prerequisite to apply scheduling analysis and verify system viability. WCET analysis is usually a complex and time-consuming activity. It becomes increasingly complex when one also considers code generation strategies from high-level models. In this paper, we present an experiment made on the coupling of the WCET analysis tool Bound-T and our AADL to code generator OCARINA. We list the different steps to successfully apply WCET analysis directly from model, to limit user intervention.

Keywords: WCET, AADL, Bound-T, Ocarina

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1665

Traces as a Solution to Pessimism and Modeling Costs in WCET Analysis

Whitham, Jack; Audsley, Neil

WCET analysis models for superscalar out-of-order CPUs generally need to be pessimistic in order to account for a wide range of possible dynamic behavior. CPU hardware modifications could be used to constrain operations to known execution paths called traces, permitting exploitation of instruction level parallelism with guaranteed timing. Previous implementations of traces have used microcode to constrain operations, but other possibilities exist. A new implementation strategy (virtual traces) is introduced here. In this paper the benefits and costs of traces are discussed. Advantages of traces include a reduction in pessimism in WCET analysis, with the need to accurately model CPU internals removed. Disadvantages of traces include a reduction of peak throughput of the CPU, a need for deterministic memory and a potential increase in the complexity of WCET models.

Keywords: WCET superscalar cpu virtual traces

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1666

A tool for average and worst-case execution time analysis

Hickey, David Early, Diarmuid; Schellekens, Michel

We have developed a new programming paradigmwhich, for conforming programs, allows the average-case execution time (ACET) to be obtained automatically by a static analysis.

This is achieved by tracking the data structures and their distributions that will exist during all possible executions of a program. This new programming paradigm is called MOQA and the tool which performs the static analysis is called Distritrack. In this paper we give an overview of both MOQA and Distritrack. We then discuss the possibility of extending Distritrack for static worst-case execution time (WCET) analysis ofMOQA programs using the tight tracking of data structures already being performed.

Keywords: Tool, static timing, worst-case, average-case, execution time

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1668

TuBound - A Conceptually New Tool for Worst-Case Execution Time Analysis

Prantl, Adrian; Schordan, Markus; Knoop, Jens

TuBound is a conceptually new tool for the worst-case execution time (WCET) analysis of programs. A distinctive feature of TuBound is the seamless integration of a WCET analysis component and of a compiler in a uniform tool. TuBound enables the programmer to provide hints improving the precision of the WCET computation on the high-level program source code, while preserving the advantages of using an optimizing compiler and the accuracy of a WCET analysis performed on the low-level machine code. This way, TuBound ideally serves the needs of both the programmer and the WCET analysis by providing them the interface on the very abstraction level that is most appropriate and convenient to them. In this paper we present the system architecture of TuBound, discuss the internal work-flow of the tool, and report on first measurements using benchmarks from Mälardalen University. TuBound took also part in the WCET Tool Challenge 2008.

Keywords: Worst-case execution time (WCET) analysis, Tool Chain, Flow Constraints, Source-To-Source

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1661