SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 3, NO. 1, APRIL 2020
DOI: 10.35377/saucis.03.01.705777

MBBench: A WCET Benchmark Suite

Metin KUZHAN[1], Veysel Harun ŞAHİN[2]
[1]Corresponding Author; Sakarya University, Institute of Natural Sciences, Department of Computer and Information Sciences; metin.kuzhan@ogr.sakarya.edu.tr; ORCID: 0000-0001-6420-8327
[2] Sakarya University, Faculty of Computer and Information Sciences, Department of Software Engineering; vsahin@sakarya.edu.tr; ORCID: 0000-0002-3381-1702

Received 18 March 2020; Accepted 22 April 2020; Published online 30 April 2020

Abstract

One of the important features of any real-time software is the worst-case execution time (WCET). To get an understanding of the timing behavior of real-time systems and to prove that the real-time software meets its deadlines, WCET analysis is performed. Today, researchers actively develop new WCET analysis methods and tools. Therefore, they need benchmark programs to evaluate and compare their work. To meet this need, in this study we present a new benchmark suite, called MBBench. MBBench includes a collection of C programs for Linux operating system and RTEMS real-time operating system. Its main aim is to help the evaluation and comparison of measurement-based WCET analysis methods/tools. MBBench has been published as open source. It can be obtained freely over the Internet.

Keywords: Real-Time Systems, Benchmark, WCET analysis, Software Engineering

MBBench: WCET Kıyaslama Kümesi

Öz

Herhangi bir gerçek zamanlı yazılımın en önemli özelliklerinden birisi, en kötü durum yürütme süresidir (WCET). Gerçek zamanlı sistemlerin zamanlama davranışını anlamak ve gerçek zamanlı yazılımın son teslim tarihlerini karşıladığını kanıtlamak için WCET analizi yapılır. Günümüzde araştırmacılar aktif olarak yeni WCET analiz yöntemleri ve araçları geliştirmektedir. Dolayısıyla, çalışmalarını değerlendirmek ve karşılaştırmak için kıyaslama programlarına ihtiyaç duymaktadırlar. Bu çalışmada, bu ihtiyacı karşılamaya yardımcı olmak amacıyla MBBench isminde yeni bir kıyaslama kümesi sunuyoruz. MBBench, Linux işletim sistemi ve RTEMS gerçek zamanlı işletim sistemi için C programları koleksiyonu içermektedir. Kıyaslama kümesinin temel amacı, ölçüm tabanlı WCET analizi yöntemlerinin/araçlarının değerlendirilmesine ve karşılaştırılmasına yardımcı olmaktır. MBBench, açık kaynak kodlu olarak yayınlanmıştır ve İnternet üzerinden ücretsiz olarak edinilebilir.

Anahtar Kelimeler: Gerçek Zamanlı Sistemler, Kıyaslama, WCET analizi, Yazılım Mühendisliği

1. Introduction

Today, real-time systems [1], [2] are widely used in many different areas from aviation to automotive industry, from home appliances to health equipment. Large amount engineering and scientific study are being held on real-time systems. One of the essential parts of any real-time system is real-time software. Real-time software's correctness depends both on producing the correct output(s) and meeting its deadlines. Because of this, a crucial property of any real-time software is the execution time. Before deploying any real-time system, the developers should prove that the real-time software meets its deadlines. For this purpose, the worst-case execution time (WCET) of the real-time software is calculated. The process of this calculation is called WCET analysis or timing analysis.

WCET analysis of real-time software is a hot topic in real-time research. Many approaches are proposed, and new tools are developed actively. Detailed information and comparison of different timing analysis approaches can be found in [3]–[7].

One type of WCET analysis approaches is measurement-based approach. In this approach, firstly the real-time software is run several times with different inputs on actual hardware or on architecture simulator. During this phase, the execution time of each run is collected. Then this information is processed by using various methods to inference about the WCET of the software. Readers can get more information about measurement-based approach from [8], [9]. Researchers who develop these approaches, need to evaluate their methods. They also need to compare the performance of their methods against others'. During this evaluation and comparison, they need common programs. In other words, they need benchmark programs.

In addition to WCET analysis methods, there are also WCET analysis tools. Main objective of these tools is to ease WCET analysis procedure. There are currently several tools available, both commercial and open source. For example Chronos [10], Otawa [11] and Heptane [12], [13], are open source WCET tools. RapiTime [14] developed by Rapita Systems and aiT [15] developed by AbsInt are two commercial tools. WCET tool developers also need benchmark programs to evaluate and compare their software with others'.

The main motivation of this study is the need for benchmark programs for the evaluation of WCET analysis methods and tools. For this purpose, in this paper we introduce the first version of a new benchmark suite, named MBBench (MBBench 1.0). There are also different benchmark suites available today for WCET community. We give detailed information about benchmarks in the next section. Each benchmark addresses different aspects of WCET analysis. The principle aim of MBBench is to cover the need for benchmark programs to evaluate measurement-based WCET analysis methods/tools. The contributions of this study to the WCET community are listed below:

- This paper introduces MBBench benchmark suite which aims to help the evaluation and comparison of measurement-based WCET analysis methods/tools.
- MBBench supports multiple platforms. It includes programs for Linux operating system [16] and RTEMS real-time operating system [17] operating systems.
- MBBench includes multi-path programs.
- MBBench is published as open source. It is freely available over the Internet. Readers can access the benchmark suite through the Sakarya University, Faculty of Computer and Information Sciences, Department of Software Engineering, Real-Time Systems Research Laboratory homepage [18].

Although MBBench specifically targets measurement-based WCET methods/tools, it can also be used for the evaluation and comparison of other types of WCET methods/tools, real-time platforms and computer systems.

The rest of the paper is organized as follows. Second section describes benchmarking and several benchmark suites. Third section gives a detailed explanation of MBBench. In the fourth section, we tell about our experiences and discuss the MBBench. The sixth section concludes the paper with the information about future work.

2. Benchmark

Currently benchmark programs are used widely in computer science and engineering for several different purposes like evaluating central processing unit (CPU) performance, measuring power characteristics, timing analysis, testing network resources. In addition, they help the evaluation of various computer systems, e.g. personal computers (PC), servers, virtualization platforms, real-time and embedded systems, internet of things (IoT), mobile systems. Besides these they are also used to validate and compare newly created methods/tools by researchers. Benchmarks are of vital importance for both commercial organizations and research community. Therefore, many benchmark suites for several different purposes are present both commercial and open source. In this section we explain some of the benchmark suites.

One of the well-known benchmark suites are developed by Standard Performance Evaluation Corporation (SPEC) [19]. SPEC develops many kinds of benchmarks for different computing environments and for evaluating different characteristics. For example, while SPEC CPU 2017 is used to evaluate CPU performance of systems, SPEC SFS 2014 is used to evaluate the performance of file servers.

Embedded Microprocessor Benchmark Consortium (EEMBC) [20] develop different types of benchmarks specially for mobile and embedded systems. Some of them are IoTMark for IoT devices, ADASMark for advanced driver-assistance systems (ADAS), and DENBench for digital entertainment products.

Apart from these, there are individual benchmark suites developed by different scientific communities for specific needs. For example, PARSEC benchmark suite [21] includes several applications to help the study of chip-multiprocessors (CMPs). BigDataBench-S [22] is an open source benchmark suite that focuses the evaluation of big data systems. The DaCapo benchmark suite [23] has been developed to evaluate the Java platforms. It is open source and includes client-side Java programs. CDx [24] is another benchmark suite specifically targets real-time specification for Java (RTSJ) [25].

As for the WCET analysis there are also several benchmark suites and applications are available. One of the well-known WCET benchmark suite is the Mälardalen benchmark suite [26], [27]. It includes several programs which have various programming constructs and properties. The programs in Mälardalen benchmark suite are mostly single path and small programs mainly aimed at flow analysis. They were written in C programming language.

PapaBench [28] is an open source benchmark application based on Paparazzi. Paparazzi is an unmanned aerial vehicle (UAV) control application. PapaBench is also developed for the comparison of WCET analysis methods/tools.

TACLeBench [29] is a benchmark collection which can be used in WCET research. As its name implies it is a collection of several open source benchmark applications and suites like sequential benchmarks, parallel benchmarks, real-life applications. The programs in TACLeBench are self-contained. The inputs of the programs are embedded in the source code.

PBench [18], [30] is another open source benchmark suite. The objective of PBench is to provide parallel benchmark programs for WCET analysis in the context of multi-core platforms. It includes both sequential and parallel versions of each program to better help the comparison. It is developed in C programming language.

3. MBBench

In this section we give a detailed explanation of the first version of MBBench benchmark suite (MBBench 1.0). MBBench is an open source benchmark suite, designed and developed to help measurement-based WCET analysis research. However, it can also be used for the comparison of the other type WCET analysis approaches, real-time platforms and computer systems.

3.1 Method

In this section we explain the method which we followed during the MBBench study. Because it is a software development project, we have four main phases: design, development, test, and publish.

In the design phase of the MBBench, we considered three design criteria which are listed below.

- programming language
- operating system
- algorithms

For the first criterion we decided to use C programming language, as it is widely preferred in the development of real-time systems. All of the programs in MBBench is written in this language.

For the second criterion we decided to support two different kinds of operating systems: Linux operating system and RTEMS real-time operating system. This choice has two motives. The first one is the wide usage of these two operating systems in both industry and academia. The second one is about RTEMS. To the best of authors' knowledge there are not many benchmarks available that target RTEMS. To support both operating systems, we developed two versions of each program. We tried to write the different versions of the same program as similar as possible for better comparison of different platforms.

For the third criterion (algorithms), we first identified the requirements for the benchmark. The requirements are listed below.

- Our main focus is measurement-based WCET analysis. Therefore, we need programs that take inputs and operate on those inputs.
- Each program should include various program constructs (loops, decisions etc.).
- Each program should include several program properties (single-threaded, external routine usage etc.).
- Programs should carry out operations on different data types (floating point, integer etc.) and data structures (array).
- At least one program should include bit level operations.

Then depending on the requirements, we determined the features which we want to support. Afterward, we gave an acronym to each feature for simplicity. The features are shown in Table 1.

Table 1 Features

Name	Acronym	Description
Single-threaded	ST	The program is single-threaded.
Multi-threaded	MT	The program is multi-threaded.
External routine	ER	The program uses external routine.
Single path	SP	The program always runs the same execution path in each run.
Multi path	MP	The program may follow different execution paths in each run.
Dynamic Memory	DM	The program makes dynamic memory allocation.
Loop	L	The program includes loops.
Nested loop	NL	The program includes nested loops.
Recursion	R	Recursive function calls are present.
Decision	D	Decision structures (if…else etc.) are used.
Array	A	The program operates on arrays.
Bit Level Operation	BLO	Bit level operations are present.
Floating Point Operation	FPO	Floating point operations are present.
Integer Operation	IO	Integer operations are present.
Input Vector	IVEC	The program takes a vector as input.
Input Value	IVAL	The program takes a single value as input.
Input File	IF	The program takes a file as input.

Later, taking into account those features we selected algorithms that will be implemented. The algorithms were selected from well-known computer science problems. We selected algorithms in a way that the implementation of each of them will require to use different sets of features. By doing this, we aimed to create a benchmark suite that successfully represents all types of program structures and properties.

After the planning phase we wrote programs in C programming language. Each program implements one algorithm and thus solves one kind of problem. Also, each program has two versions for each operating system.

The Linux versions of the programs in MBBench and the supported features are shown in Table 2. In the table, rows indicate features, and columns indicate programs. A plus sign (+) in a cell means that the program in that column provides the corresponding feature. A minus sign (-) means the feature is not

supported. As seen in Table 2, we currently don't support all the features we determined. In future versions of the MBBench, we plan to cover more features by adding new programs

Table 2 MBBench 1.0 Programs (Linux version) feature matrix

Programs / Features	**booth**	**bucket_sort**	**cesar**	**counting_sort**	**gcd**	**huffman**	**knapsack**	**merge_sort**	**miller_rabin**	**pollard_rho**	**quick_sort**	**rabin_karp**	**radix_sort**	**rsa**	**standard_deviation**
ST	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+
MT	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-
ER	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
SP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MP	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DM	-	-	-	-	+	-	-	-	-	-	+	-	+	-	-
L	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+
NL	-	+	-	+	+	+	+	-	-	-	-	+	+	-	+
R	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+
D	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
A	+	+	+	+	-	+	+	+	-	-	+	+	+	-	+
BLO	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FPO	-	-	-	-	-	+	-	-	-	-	-	-	-	+	-
IO	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IVEC	+	+	+	+	+	+	+	+	-	-	+	+	+	+	-
IVAL	-	-	-	-	-	-	-	-	+	+	-	-	-	-	-
IF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+

The RTEMS versions of the programs in MBBench, and the supported features are shown in Table 3. In the table, rows indicate features, and columns indicate programs. A plus sign (+) in a cell means that the program in that column provides the corresponding feature. A minus sign (-) means the feature is not supported. As seen in Table 3, we currently don't support all the features we determined. In future versions of the MBBench, we plan to cover more features by adding new programs

Table 3 MBBench 1.0 Programs (RTEMS version) feature matrix

Programs / Features	**booth**	**bucket_sort**	**cesar**	**counting_sort**	**gcd**	**huffman**	**knapsack**	**merge_sort**	**miller_rabin**	**pollard_rho**	**quick_sort**	**rabin_karp**	**radix_sort**	**rsa**	**standard_deviation**
ST	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
MT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ER	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
SP	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
MP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DM	-	-	-	-	+	-	-	-	-	-	+	-	+	-	-
L	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+
NL	-	+	-	+	+	+	+	-	-	-	-	+	+	-	+
R	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+
D	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
A	+	+	+	+	-	+	+	+	-	-	+	+	+	-	+
BLO	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FPO	-	-	-	-	-	+	-	-	-	-	-	-	-	+	-

Table 3 MBBench 1.0 Programs (RTEMS version) feature matrix (cont.)

IO	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IVEC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IVAL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

The compilation of Linux versions of the programs was performed by using GNU Compiler Collection (GCC) 7.2.0. Both the compilation and test runs were executed on XUbuntu 17.10 linux distribution.

RTEMS versions of the programs were compiled with the cross-compilation tools bundled with RTEMS 5.0.0. Test runs were executed on SPARC emulator.

3.2 Details of Programs in MBBench 1.0

The programs in MBBench 1.0 is listed in Table 4. In the second column of the table we give the name of the algorithm which the corresponding program implements. Each program implements one algorithm that is well known in computer science. Interested readers can get more information about algorithms from [31]. Booth's algorithm can be found in [32]. In the third column of the table brief descriptions of the algorithms and programs can be found.

Table 4 MBBench 1.0 Programs

Program Name	Algorithm	Description
booth	Booth's algorithm	Multiplies two single-digit positive numbers.
bucket_sort	Bucket sort algorithm	Sorts integers.
cesar	Cesar algorithm	Encrypts a string.
counting_sort	Counting sort algorithm	Sorts integers.
gcd	Greatest common divisor	Finds the greatest common divisor of two positive numbers.
huffman	Huffman code algorithm	Compresses a string.
knapsack	0-1 knapsack problem	Finds the maximum value that can be fitted in a backpack.
merge_sort	Merge sort algorithm	Sorts integers.
miller_rabin	Miller-Rabin primality test	Finds whether a number is prime or not.
pollard_rho	Pollard rho algorithm	Finds the multipliers of a number.
quick_sort	Quick sort algorithm	Sorts integers.
rabin_karp	Rabin-Karp algorithm	String matching algorithm.
radix_sort	Radix sort algorithm	Sorts integers.
rsa	RSA Cryptosystem	Encryption and decryption of a string.
standard_deviation	Standard deviation	Calculates the standard deviation.

3.3 Benchmark Directory Organization

In the repository, each benchmark program is stored in its own directory. The directory name is the name of the program. Inside each program's directory there two separate directories: linux and rtems. As their name implies, linux directory includes the Linux version, and rtems directory includes the RTEMS version.

In each version's directory we provide the source code of the program with a "c" extension; a README file which gives information about the program; a Makefile which helps the compilation of the program; call graph and scope hierarchy graph of the program. A sample call graph of the Linux version of the huffman program is shown in Figrue 1. In Figure 2, a sample scope hierarchy graph of the same program is shown.

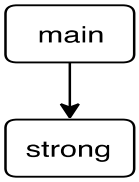

Figure 1 Call graph of huffman program (Linux version)

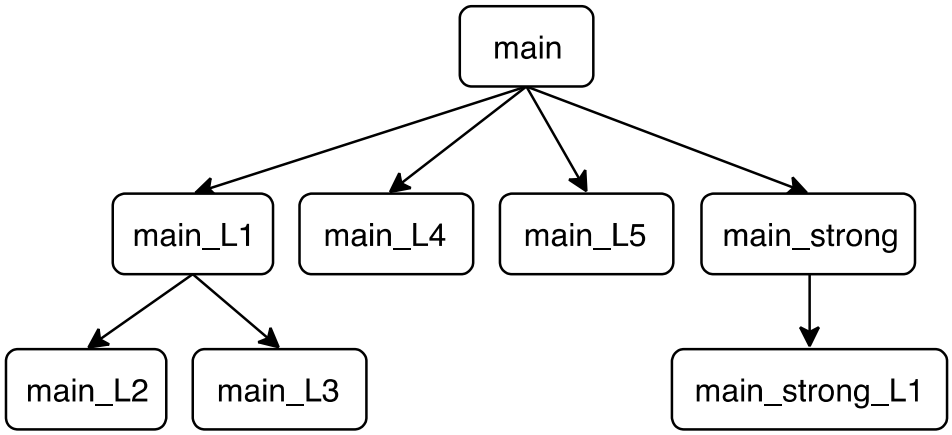

Figure 2 Scope hierarchy graph of huffman program (Linux version)

4. Results and Discussion

In this work we have developed a benchmark suite and released its first version MBBench 1.0. It has been published as open source. It is freely available over the Internet [18]. Research community can freely obtain the benchmark and use it to their needs freely.

In this section firstly, we discuss the MBBench and other benchmarks. Then we share our experiences and explain the problems we faced during the development.

4.1 Discussion of Benchmarks

The main motivation of this study is to help the evaluation and comparison of WCET analysis methods/tools. There are several benchmark suites available to help WCET community. Each suite covers different aspects of WCET analysis.

Mälardalen benchmarks are mainly focused on flow analysis and they have weak support for measurement-based WCET analysis [27]. Main aim of MBBench is to help the evaluation and comparison of measurement-based WCET analysis methods/tools. Especially the Linux version of MBBench has strong support for measurement-based WCET analysis.

TACLeBench benchmarks include self-contained programs [29]. The inputs are embedded into the source code of programs. On the other hand, Linux version of MBBench supports inputs from the environment. 12 programs take input vector from command line, 2 programs take single input value from command line and 1 program uses input file. RTEMS version MBBench does not support to take input from command line and files. The inputs are hardcoded in these programs. The researcher needs to change hardcoded inputs of RTEMS versions, for strong support of measurement-based WCET analysis.

As the programs of MBBench Linux version get input from the environment, they can take different paths during different runs. In other words, these programs are multipath programs.

MBBench targets different operating systems: Linux and RTEMS. For this support, we developed two versions of each program; one for Linux and one for RTEMS. We also tried to make different versions as similar as possible.

PBench focuses on parallel benchmarks for WCET analysis [30]. It helps the evaluation of WCET analysis methods/tools from multithreaded viewpoint. MBBench currently does not include parallel programs. Both benchmarks support RTEMS operating system.

PapaBench [28] is based on an unmanned aerial vehicle (UAV) control application. From this viewpoint it resembles an industrial real-time application. MBBench does not include real-time industrial applications.

4.2 Experinces

4.2.1 Input Problem

Because of our main focus is measurement-based WCET analysis methods/tools, we concentrated on the problems which need inputs during algorithm selection. We made those selections based on three kinds of inputs: single input value (IVAL), input vector (IVEC), and input file (IF).

We have successfully implemented all of the algorithms in Linux versions of programs. 11 programs use vectors, 2 programs use single values, and 1 program uses file as input.

However as stated above, RTEMS is a real-time embedded operating system. It is not designed to run programs from command line, and to get inputs from command line environment. Therefore, we did not use external command line inputs in the RTEMS versions of the programs. Instead, we hard coded the needed inputs inside the program by hand. If researchers plan to use different inputs for RTEMS programs they need to change inputs by hand and compile the programs before every run.

Also, as a result of this, Linux versions of the programs are multipath programs, and RTEMS versions of the programs are single path programs. Linux versions of the programs can follow different paths depending on the inputs. On the other hand, RTEMS versions of the programs always follow the same path because inputs are hard coded. To change this behavior, researchers may follow the abovementioned procedures.

4.2.2 Random Number Generation Problem

In booth, bucket_sort, counting_sort, gcd, merge_sort, radix_sort, and rsa programs we generate random values in Linux versions successfully. But during our tests, we could not generate random values in RTEMS versions of the programs. In each run of the programs the generated random values were the same.

This is because of the emulator usage during the tests of RTEMS programs. Random value generation is based on time information. In each run of the emulator, time information starts from a fixed value. Also, the program code does not change between runs, and hence the random value generating instructions persist at the same location of the program. Therefore, whenever a random value is generated it is based on the exact same time information. As a result of this, the same value is generated by the program in each run instead of different values.

Because of this, we decided to remove random number generation property in RTEMS versions of the programs. We hard coded the values by hand. Because of this, if researchers plan to use different values for RTEMS programs they need to change these hard-coded values by hand and compile the programs before every run. Alternatively, if they work on a system which can get current time information or some other kind of randomization source, they may consider adding randomize function in RTEMS versions as well.

5. Conclusions and Future Work

In this study we developed a benchmark suite to help researchers who work on measurement-based WCET analysis. Although we focused on measurement-based approach, we believe that our benchmark suite can also be used for other WCET analysis methods/tools. They also can be used for general computer system benchmarking.

Because real-time and embedded systems become prevalent, we believe that scientific and engineering studies will increase in this field. As a result of this, more and different kind of benchmarks will be needed. Benchmark development will continue to be an important field in computer science.

Currently MBBench does not cover all the features we determined. In the future, we plan to add more benchmark programs to better represent our feature matrix.

From operating system perspective, supporting more real-time operating systems can be a good addition to MBBench. There are several open source real-time operating systems. For example, Zephyr [33] from the Linux Foundation can be a good candidate.

From a programming perspective, supporting different programming languages like Ada, Java and some functional programming languages may be very valuable.

Acknowledgments

The authors would like to acknowledge that this work is supported by the Real-Time Systems Research Laboratory at Sakarya University, Faculty of Computer and Information Sciences, Department of Software Engineering. The MBBench benchmark suite can be obtained from the Real-Time Systems Research Laboratory homepage freely [18].

References

[1] G. C. Buttazzo, *Hard Real-Time Computing Systems*, 3rd ed., vol. 24. Boston, MA: Springer US, 2011.

[2] H. Kopetz, *Real-Time Systems*, 2nd ed. Boston, MA: Springer US, 2011.

[3] F. J. Cazorla, L. Kosmidis, E. Mezzetti, C. Hernandez, J. Abella, and T. Vardanega, "Probabilistic Worst-Case Timing Analysis: Taxonomy and Comprehensive Survey," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–35, 2019, doi: 10.1145/3301283.

[4] R. I. Davis and L. Cucu-Grosjean, "A Survey of Probabilistic Timing Analysis Techniques for Real-Time Systems," *Leibniz Transactions on Embedded Systems (LITES)*, vol. 6, no. 1, pp. 3:1–3:60, 2019, doi: 10.4230/LITES-v006-i001-a003.

[5] J. Abella, D. Hardy, I. Puaut, E. Quinones, and F. J. Cazorla, "On the comparison of deterministic and probabilistic WCET estimation techniques," *Proceedings - Euromicro Conference on Real-Time Systems*, pp. 266–275, 2014, doi: 10.1109/ECRTS.2014.16.

[6] J. Abella *et al.*, "WCET analysis methods: Pitfalls and challenges on their trustworthiness," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, Siegen,Germany, pp. 1–10, 2015, doi: 10.1109/SIES.2015.7185039.

[7] R. Wilhelm *et al.*, "The worst-case execution-time problem—overview of methods and survey of tools," *Transactions on Embedded Computing Systems*, vol. 7, no. 3, pp. 1–45, Apr. 2008, doi: 10.1145/1347375.1347389.

[8] L. Cucu-Grosjean *et al.*, "Measurement-based probabilistic timing analysis for multi-path programs," *Proceedings - Euromicro Conference on Real-Time Systems*, pp. 91–101, 2012, doi: 10.1109/ECRTS.2012.31.

[9] F. J. Cazorla *et al.*, "PROXIMA: Improving Measurement-Based Timing Analysis through Randomisation and Probabilistic Analysis," *Proceedings - 19th Euromicro Conference on Digital System Design, DSD 2016*, pp. 276–285, 2016, doi: 10.1109/DSD.2016.22.

[10] X. Li, Y. Liang, T. Mitra, and A. Roychoudhury, "Chronos: A timing analyzer for embedded software," *Science of Computer Programming*, vol. 69, no. 1–3, pp. 56–67, 2007, doi: 10.1016/j.scico.2007.01.014.

[11] Université of Toulouse, "OTAWA - WCET is coming...," 2020. [Online]. Available: http://otawa.fr. [Accessed: 17-Mar-2019].

[12] D. Hardy, B. Rouxel, and I. Puaut, "The Heptane Static Worst-Case Execution Time Estimation Tool," in *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*, Dagstuhl, Germany, vol. 57, pp. 8:1—-8:12, 2017, doi: 10.4230/OASIcs.WCET.2017.8.

[13] Inria, "Heptane static WCET estimation tool - PACAP," 2020. [Online]. Available: https://team.inria.fr/pacap/software/heptane. [Accessed: 17-Mar-2019].

[14] Rapita Systems, "Rapita Systems | On-target software verification solutions," 2020. [Online]. Available: https://www.rapitasystems.com. [Accessed: 17-Mar-2020].

[15] AbsInt, "AbsInt: Cutting-Edge Tools for Static Analysis of Safety-Critical Software," 2020. [Online]. Available: https://www.absint.com. [Accessed: 17-Mar-2020].

[16] Linux Kernel Organization, "The Linux Kernel Archives," 2020. [Online]. Available: https://www.kernel.org. [Accessed: 17-Mar-2020].

[17] The RTEMS Project, "RTEMS Real Time Operating System (RTOS)," 2020. [Online]. Available: https://www.rtems.org. [Accessed: 17-Mar-2020].

[18] Sakarya University, Faculty of Computer and Information Sciences, Department of Software Engineering, "Real-Time Systems Research Laboratory homepage," 2020. [Online]. Available: http://rtsrlab.sakarya.edu.tr. [Accessed: 17-Mar-2020].

[19] Standard Performance Evaluation Corporation, "Spec - Standard Performance Evaluation Corporation," 2020. [Online]. Available: https://www.spec.org. [Accessed: 17-Mar-2020].

[20] EEMBC, "Embedded Microprocessor Benchmark Consortium," 2020. [Online]. Available: https://www.eembc.org. [Accessed: 17-Mar-2020].

[21] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques - PACT '08*, New York, New York, USA, 2008, p. 72, doi: 10.1145/1454115.1454128.

[22] X. Tian *et al.*, "BigDataBench-S: An Open-Source Scientific Big Data Benchmark Suite," presented at the 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Lake Buena Vista, FL, USA, 2017, pp. 1068–1077, doi: 10.1109/IPDPSW.2017.111.

[23] S. M. Blackburn *et al.*, "The DaCapo benchmarks," in *Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications - OOPSLA '06*, New York, New York, USA, 2006, p. 169, doi: 10.1145/1167473.1167488.

[24] T. Kalibera, P. Parizek, G. Haddad, G. T. Leavens, and J. Vitek, "Challenge benchmarks for verification of real-time programs," in *Proceedings of the 4th ACM SIGPLAN workshop on Programming languages meets program verification - PLPV '10*, New York, New York, USA, 2010, p. 57, doi: 10.1145/1707790.1707800.

[25] G. Bollella *et al.*, *The Real-Time Specification for Java*. Addison-Wesley, 2000.

[26] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The Mälardalen WCET Benchmarks: Past, Present and Future," in *10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010)*, Dagstuhl, Germany, 2010, vol. 15, pp. 136–146, doi: 10.4230/OASIcs.WCET.2010.136.

[27] Mälardalen Real-Time Research Center, "The Mälardalen WCET Benchmarks," 2013. [Online]. Available: http://www.mrtc.mdh.se/projects/wcet/benchmarks.html. [Accessed: 17-Mar-2020].

[28] F. Nemer, H. Cassé, P. Sainrat, J.-P. Bahsoun, and M. D. Michiel, "PapaBench: a Free Real-Time Benchmark," in *6th International Workshop on Worst-Case Execution Time Analysis (WCET'06)*, Dagstuhl, Germany, 2006, vol. 4, doi: 10.4230/OASIcs.WCET.2006.678.

[29] H. Falk *et al.*, "TACLeBench: A Benchmark Collection to Support Worst-Case Execution Time Research," in *16th International Workshop on Worst-Case Execution Time Analysis (WCET 2016)*, Dagstuhl, Germany, 2016, vol. 55, pp. 2:1—-2:10, doi: 10.4230/OASIcs.WCET.2016.2.

[30] S. Serttaş and V. H. Şahin, "PBench: A Parallel, Real-Time Benchmark Suite," in *Academic Perspective Procedia*, Alanya, Antalya, Turkey, 2018, vol. 1, pp. 178–186, doi: 10.33793/acperpro.01.01.37.

[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Algoritmalara Giriş (Üçüncü Baskıdan Çeviri)*. Palme Yayıncılık, 2017.

[32] M. M. Mano, *Bilgisayar Sistemleri Mimarisi (3. Basımdan Çeviri)*. Literatür Yayınları, 2015.

[33] Linux Foundation, "Zephyr Project homepage," 2020. [Online]. Available: https://www.zephyrproject.org. [Accessed: 17-Mar-2020].