©2018 Published in 6th International Symposium on Innovative Technologies in Engineering and Science 09-11 November 2018 (ISITES2018 Alanya – Antalya - Turkey)

PBench: A Parallel, Real-Time Benchmark Suite

[1,2,*]Sevil Serttaş and [3]Veysel Harun Şahin
[1]Department of Computer and Information Engineering, Institute of Natural Sciences, Sakarya University, Sakarya, Turkey
[2]Department of Computer Engineering, Faculty of Engineering, Dumlupınar University, Kütahya, Turkey
[3]Department of Software Engineering, Faculty of Computer and Information Sciences, Sakarya University, Sakarya, Turkey

Abstract

Real-time systems are widely used from the automotive industry to the aerospace industry. The scientists, researchers, and engineers who develop real-time platforms, worst-case execution time analysis methods and tools need to compare their solutions to alternatives. For this purpose, they use benchmark applications. Today many of our computing systems are multiprocessor/multicore systems. Therefore, to be able to compare the effectiveness of real-time platforms, worst-case execution time analysis methods and tools, the research community need multi-threaded benchmark applications which scale on multicore systems. In this paper, we present the first version of PBench, a parallel, real-time benchmark suite. PBench includes different types of multi-threaded applications which implement various algorithms from searching to sorting, matrix multiplication to probability distribution calculation. In addition, PBench provides single-threaded versions of all programs to allow side by side comparisons.

Key words: Real-Time Systems, Benchmark Suite, Parallel Programming, Software Engineering

1. Introduction

Benchmark programs are used to evaluate and compare computer systems, real-time systems, algorithms, databases and many other technologies. Benchmark programs are used both by manufacturers and researchers. By using benchmark programs, researchers can evaluate their developments and compare them against alternatives. Today, benchmark programs are widely used in different computer science disciplines. Benchmark studies have also a strong positive effect on software engineering research. Benchmark programs help to make software engineering researches more consistent. Readers can get more information about benchmarking and research from [1].

With the rapid progress of computer hardware technology, multiprocessor/multicore systems became widespread. The number of programs which take advantage of these architectures increase every day. These programs include several threads which run simultaneously on different cores of the system. These kinds of programs are called parallel programs and this kind of programming is called parallel programming. There are different threading libraries which help developers create and manage threads and thus are used for parallel programming. One of the most well-known libraries is POSIX threads (Pthreads). For parallel program development there are also implicit models in which the compilers take the burden of thread management from developers. One of the well-known models is OpenMP. Interested readers can get more information about parallel programming from [2]–[4].

*Corresponding author: Address: Department of Computer and Information Engineering, Institute of Natural Sciences, Sakarya University, Sakarya, Turkey. E-mail address: sevil.serttas@ogr.sakarya.edu.tr, Phone: +902742654459

Today, scientists and engineers also try to make real-time programs run on multicore systems. For this purpose, a lot of research are carried out on real-time systems. Interested readers can get more information about real-time systems can look at [5]. It is clear that people who make research on parallel aspects of real-time systems need to compare their developments. Therefore, they need parallel benchmark programs which can scale on multicore systems. This is one of the motivations of our study.

The most important criterion in real-time systems is time concept. Before deploying real-time programs, developers need to make sure that the program does not exceed the timing limits. To be able to get an information about the execution time of the whole or the parts of the program, worst-case execution time (WCET) analysis is performed. Readers can get more information about WCET analysis from [6], [7].

WCET analysis is an open question in industry and academia. Several researchers are developing new WCET analysis methods and tools to improve the reasoning about the timing behavior of real-time programs. Readers can get more information about the current research on WCET analysis from [8], [9], [10], [11], [12]. Two of the WCET analysis tools are RapiTime [13] and aiT [14].

Like in other areas, researchers who work on new WCET analysis methods and tools need to evaluate and compare their work. For this purpose, they use benchmark programs preferably developed for WCET analysis. There are several benchmark programs and benchmark suites -a collection of benchmark programs- specific to WCET analysis [15]–[18]. In the second section of this paper we explain these benchmark programs and suites in addition to other well-known benchmarks. In this study we focused on WCET analysis and developed benchmark programs to help comparison of WCET analysis tools and methods.

As multicore systems become widespread, WCET community also make research on these systems. They need multi-threaded benchmark programs especially, to test the parallel aspects of their tools and methods. Therefore, parallel benchmark programs which scale on multicore systems are becoming necessity also for WCET community which is another motivation of our study.

In this paper we present the first version of PBench (PBench 1.0), a parallel benchmark suite which can be used for the comparison of real-time platforms, WCET analysis methods and tools from a multi-threaded perspective. PBench 1.0 can be accessed through our web site [19].

In second section we give information about several benchmark suites. Third section describes PBench and fourth section tells about conclusions and future work.

2. Benchmarking

Benchmarking is an important procedure in all areas of computer science either real-time or not. All kinds of things from newly developed tools, software, and hardware to newly proposed methods and algorithms need to be evaluated and compared. For this purpose, benchmark programs are used.

There are a lot of benchmark programs and suites available for different purposes. For example, Standard Performance Evaluation Corporation (SPEC) [20] develops several different benchmark suites -generally named as SPEC Benchmarks- to evaluate different areas of computer systems like power characteristics (e.g., SPECpower_ssj2008) and CPU performance (e.g., SPEC CPU2017). As another example, BigDataBench-S [21] is a big data benchmark suite. It is open source and focuses on scientific scenarios. Because our benchmarks are for real-time systems, the sequel of this section focuses on benchmark suites for embedded and real-time systems.

One of the most well-known benchmark suites for real-time systems is the Mälardalen WCET benchmarks [15], [16]. This benchmark suite is especially aimed for WCET analysis. It includes many programs written in C programming language. The programs use different kinds features like array operations, floating point calculations etc. to enable scientists and developers to test different aspects of their work. The Mälardalen WCET benchmarks consist of mostly small programs which are focused on flow analysis.

PapaBench [18] is a parallel real-time benchmark application focused on WCET and scheduling analysis. It is based on a real-life UAV control software and with this feature it represents real-time embedded industrial application.

TACLeBench [17] is a collection of benchmarks programs which aims to help worst case execution time research. It contains 53 benchmark programs in different types including parallel applications.

Embedded Microprocessor Benchmark Consortium (EEMBC) [22] develops benchmarks especially targeted for embedded systems. EEMBC provides many benchmarks in several categories. Although we will not go into detail of these benchmarks in this paper we think it is useful to give the main categories. EEMBC benchmarks are classified in five groups: Ultra-Low Power (ULP) and Internet of Things, Heterogeneous Compute, Single-core Processor Performance, Symmetric Multi-core Processor Performance, Phone and Tablet.

The MiBench benchmark suite includes a variety of programs in different embedded system areas such as automotive and industrial control, consumer devices, office automation, networking, security, and telecommunications [23], [24]. The programs are platform independent and were written C programming language.

Table 1. Feature Matrix of PBench 1.0 Programs

Programs / Features	p_matrix_multiplication	matrix_multiplication	p_selection_sort	selection_sort	p_array_search	array_search	p_factorial	factorial	p_binomial_distribution	binomial_distribution
ST	-	+	-	+	-	+	-	+	-	+
MT	+	-	+	-	+	-	+	-	+	-
ER	+	+	+	+	+	+	+	+	+	+
SP	-	-	-	-	+	+	-	-	+	+
MP	+	+	+	+	-	-	+	+	-	-
DM	-	-	+	-	-	-	-	-	-	-
L	+	+	+	+	+	+	+	-	+	+
NL	+	+	-	-	-	-	-	-	-	-
D	-	-	+	+	+	+	+	+	-	-
A	+	+	+	+	+	+	+	-	+	-
BLO	-	-	-	-	-	-	-	-	-	-
R	-	-	-	+	-	-	-	+	-	-
FPO	-	-	-	-	-	-	-	-	-	-
IO	+	+	+	+	+	+	+	+	+	+
IVEC	-	-	-	-	-	-	-	-	-	-
IVAL	-	-	-	-	-	-	+	+	-	-
IF	-	-	-	-	-	-	-	-	-	-

3. PBench

In this study we developed parallel benchmark suite. The name of the benchmark suite is PBench which is an acronym of “Parallel Benchmark”. It is open source and freely available. The readers can access the our benchmark suite through our web page [19]. In this section we explain the first version of PBench benchmark suite.

Although PBench is a parallel benchmark suite, we also provide single-threaded versions of all programs. By doing this we aimed to give the interested researchers the opportunity to make side by side comparisons of their work both in parallel and sequential fashion.

PBench includes five different applications. Each application has both single-threaded and multi-threaded versions. Thereby there are ten applications in PBench in total. The programs are written in C programming language. It should be noted that we continue our work on PBench, and program features, and numbers can change in the future.

During the design of PBench we first determined the features that will be needed to test during comparisons. Then we selected sample problems from different study areas. During this problem selection phase, we considered the features that we determined earlier. After that we coded the programs that solve the selected problems. We used a number of features in each of our programs in a manner that in the end PBench benchmark suite cover most of the features. The list of programs and corresponding features is given as a feature matrix in Table 1. In the table, supported features of programs are shown with plus (+) sign and not supported features of programs are shown with minus (-) sign.

Below we show the determined features. For simplicity and compact usage, we gave abbreviations to each feature.

- Single-threaded (ST): The program is sequential.
- Multi-threaded (MT): The program is parallel.
- External Routine (ER): The program uses external library.
- Single path (SP): In each run of the program, it always follows the same execution path. We provided this feature by hard coding the inputs.
- Multi path (MP): In each run of the program, it may follow a different execution path. We provided this feature by giving the inputs to the program externally.
- Dynamic Memory (DM): The program dynamically allocates and releases memory for program data.
- Loop (L): The program includes loop.
- Nested Loop (NL): The program includes nested loop.
- Recursion (R): If a function calls itself in the program, then recursion occurs.
- Decision (D): The program includes decision construct(s) (if…then etc.).
- Array (A): The program performs array operations.
- Bit Level Operation (BLO): The program performs bit level operation(s).
- Float Point Operation (FPO): The program performs floating point operation(s).
- Integer Operation (IO): The program performs integer operation(s).
- Input Vector (IVEC): The program takes an external input vector data as parameter.
- Input Value (IVAL): The program takes a single external input value as parameter.
- Input File (IF): The program takes external input file as parameter. In this case the file is opened and the data in the file is used in the program.

To cover the features listed above we have selected common well-known problems from computer science, mathematics and probability theory. The selected problems are sorting, searching, matrix multiplication, factorial calculation, and binomial distribution calculation.

For our programs we determined a pretty straightforward naming scheme. We named our programs from the names of the related problems. For example, the name of program which makes matrix multiplication is matrix_multiplication. Also, as we developed both single-threaded and multi-threaded versions programs for each problem we distinguished the parallel and sequential versions of programs with the letter "p". For example, the name matrix_multiplication represents the sequential version and the name p_matrix_multiplication represents the parallel version.

3.1 Benchmark Details

In this section we give detailed information about the benchmarks. As the problems are common and well-known the working principles of sequential versions of our programs are clear. Therefore, in this section, we will explain the details of the parallel programs.

p_selection_sort, as its name implies performs a parallel selection sort operation. This program sorts the elements of an array in ascending order. The values of the elements are assigned randomly.

p_array_search performs search operation with the help of 4 threads. Array is divided to 4 parts and each thread operates on a part of the array. In this program we hard coded the values of the elements of the array to be searched. Therefore, this program is a single path program.

p_matrix_multiplication is the parallel version of matrix multiplication program. The program performs multiplication of two 4x4 matrix using 4 threads. The values of the elements of the matrix is generated randomly.

p_factorial calculates the factorial of a number by using 2 threads. It takes the number as an external input from command line.

p_binomial_distribution calculates the binomial distribution. It uses 3 threads for the factorial calculations.

For the development platform we used Pardus 17.3 [25] Linux distribution. We have developed our programs in C programming language and we successfully compiled all of our applications by using the GNU Compiler Collection (GCC) 6.3.0. We used Pthreads API for multi-threading.

3.2 Directory Structure

Each program in the benchmark suite has its own directory. There are several files inside of the each of the programs' directory. Below we give the list and the short descriptions of these files.

- C source code of the program.
- Makefile, to help compilation process of the program.
- Call graph of the program as a PDF file to help understand the flow of program easily. A sample call graph of selection_sort application is shown in Figure 1.
- Scope hierarchy graph of the program as a PDF file to help understand function calls and loop entries easily. A sample scope hierarchy graph of selection_sort application is shown

in Figure 2.

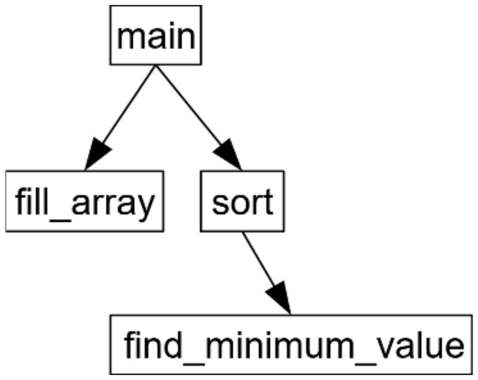

Figure 1. Call graph for selection_sort program.

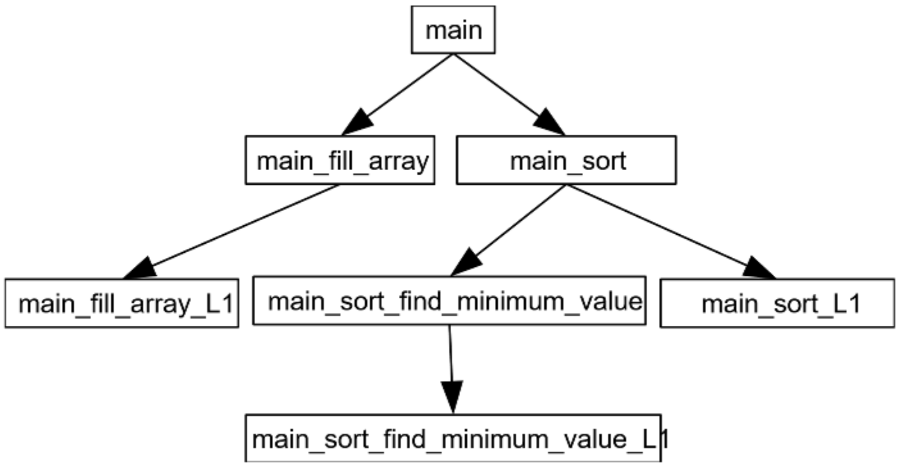

Figure 2. Scope graph hierarchy for selection_sort program.

4. Conclusion and Future Work

In this study we presented free and open source parallel benchmark suite targeted for comparison of real-time platforms, WCET analysis methods and tools. Although the programs are mainly for real-time systems they can also be used for other areas as well. We believe that as the range of benchmark suites increase, the success rate of test measurements will also increase. Therefore, more work and study on benchmarks are significant.

As the reader may see in Table 1, the current benchmark programs do not cover all the features. In the future we plan to add more programs to our benchmark suite to cover more features. We also plan to support different real-time operating systems.

Also, our study does not include statistics about the benchmark suite. In the future we plan to add statistics to help community making their evaluations and comparisons. The statistics can include, the results gathered from running the benchmarks on different timing analysis tools, worst case execution times on different architectures and operating systems, etc.

Acknowledgements

The authors would like to acknowledge that this work is supported by the Real-Time Systems Research Laboratory [4] at Sakarya University, Faculty of Computer and Information Sciences, Department of Software Engineering.

References

[1] S. E. Sim, S. Easterbrook, and R. C. Holt, "Using benchmarking to advance research: a challenge to software engineering," *25th Int. Conf. Softw. Eng. 2003. Proceedings.*, pp. 74–83, 2003.

[2] A. Grama, A. Gupta, G. Karypis, and V. Kumar, "Introduction to Parallel Computing, Second Edition," Addison-Wesley, 2003, p. 656.

[3] B. K. David and H. Wen-mei W., *Programming Massively Parallel Processors, 2nd Edition*. Morgan Kaufmann, 2012.

[4] J. Diaz, C. Muñoz-Caro, and A. Niño, "A survey of parallel programming models and tools in the multi and many-core era," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1369–1386, 2012.

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

[6] R. Wilhelm *et al.*, "The worst-case execution-time problem—overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 1–53, Apr. 2008.

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

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

[9] S. Altmeyer, L. Cucu-Grosjean, and R. I. Davis, "Static probabilistic timing analysis for real-time systems using random replacement caches," *Real-Time Syst.*, vol. 51, no. 1, pp. 77–123, 2015.

[10] L. Kosmidis *et al.*, "Fitting processor architectures for measurement-based probabilistic timing analysis," *Microprocess. Microsyst.*, vol. 47, pp. 287–302, 2016.

[11] F. J. Cazorla *et al.*, "PROXIMA: Improving Measurement-Based Timing Analysis through Randomisation and Probabilistic Analysis," in *2016 Euromicro Conference on Digital System Design (DSD)*, 2016, pp. 276–285.

[12] S. Altmeyer and R. I. Davis, "On the correctness, optimality and precision of Static Probabilistic Timing Analysis," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 2014, pp. 1–6.

[13] "Rapita Systems homepage." [Online]. Available: https://www.rapitasystems.com. [Accessed: 13-Sep-2018].

[14] "AbsInt homepage." [Online]. Available: https://www.absint.com. [Accessed: 13-Sep-

2018].

[15] 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*, 2010, vol. 15, pp. 136–146.

[16] "The Mälardalen WCET Benchmarks homepage." [Online]. Available: http://www.mrtc.mdh.se/projects/wcet/benchmarks.html. [Accessed: 13-Aug-2018].

[17] H. Falk *et al.*, "TACLeBench: A Benchmark Collection to Support Worst-Case Execution Time Research," *16th Int. Work. Worst-Case Exec. Time Anal. (WCET 2016)*, vol. i, no. 2, p. 10, 2016.

[18] F. Nemer, H. Cassé, P. Sainrat, J.-P. Bahsoun, and M. De Michiel, "PapaBench: a Free Real-Time Benchmark," in *6th International Workshop on Worst-Case Execution Time Analysis (WCET'06)*, 2006, vol. 4.

[19] "Sakarya University, Faculty of Computer and Information Sciences, Department of Software Engineering, Real-Time Systems Research Laboratory homepage." [Online]. Available: http://www.rtsrlab.sakarya.edu.tr. [Accessed: 13-Aug-2018].

[20] "Spec - Standard Performance Evaluation Corporation homepage." [Online]. Available: https://www.spec.org. [Accessed: 13-Aug-2018].

[21] X. Tian *et al.*, "BigDataBench-S: An open-source scientific big data benchmark suite," in *Proceedings - 2017 IEEE 31st International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2017*, 2017, pp. 1068–1077.

[22] "Embedded Microprocessor Benchmark Consortium homepage." [Online]. Available: https://www.eembc.org. [Accessed: 16-Aug-2018].

[23] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," *2001 IEEE Int. Work. Workload Charact. WWC 2001*, pp. 3–14, 2001.

[24] "MiBench homepage." [Online]. Available: http://vhosts.eecs.umich.edu/mibench/. [Accessed: 16-Aug-2018].

[25] "Pardus homepage." [Online]. Available: https://www.pardus.org.tr. [Accessed: 25-Aug-2018].