Towards ML-based Performance Estimation of Embedded Software: A RISC-V Case Study *

Weiyan Zhang1, Muhammad Hassan1,2, Rolf Drechsler1,2
1Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
2Institute of Computer Science, University of Bremen, Bremen, Germany
{weiyan.zhang, muhammad.hassan}@dfki.de, drechsler@uni-bremen.de

Abstract

Performance estimation of embedded software techniques mimic the behavior of real hardware, consistently navigating a balance between simulation accuracy and speed. Designers usually use real hardware, simulators, or static analyzers to obtain the performance. However, these methods suffer from serious drawbacks as real hardware is not available in the early stage of the design process, simulators either do not support any timing accuracy or require large execution time, and static analyzers need details of the hardware microarchitecture. Recently, Machine Learning (ML) has been successfully applied to estimate performance, in particular the clock cycles. It can significantly facilitate the exploration of a wide range of microarchitecture solutions at the early stage, applicable across various architectures. In this paper, we delve into the advancements of ML-based performance estimation on RISC-V processors, leveraging performance statistics obtained directly from a fast functional simulator using built-in counters. The proposed approach uses dynamic analysis for feature extraction and different ML algorithms including regression algorithms and Neural Network (NN) for the generation of Predictive Models (PMs). We present a comprehensive analysis of their effectiveness across diverse RISC-V implementations.

1 Introduction

Performance, measured in clock cycles, is a crucial design factor in system development. Predicting executed cycles aids in profiling embedded software, helping developers identify and address performance bottlenecks, such as optimizing resource utilization and enhancing overall system efficiency. Techniques for estimating performance in embedded software involve a trade-off between simulation accuracy and speed, categorized as simulation and analytic-based models.

Simulation based approaches model system architectures at different levels of abstraction. Functional simulator allows fast prototyping but lack precision. At the Electronic System Level (ESL) [1], SystemC-based Virtual Prototype (VP) is often used before the detailed hardware implementation is finalized. This abstraction gives some speedup over cycle-accurate modelling at a low abstraction level. Register Transfer Level (RTL) simulation offers a high degree of accuracy in verifying the functionality of digital circuit designs. It can detect errors at an early stage, leading to reduced cost in the design process. However, its simulation speed is comparatively slow. To effectively explore potential options for different choices of processor, performance estimation of embedded software at a higher level of abstraction is necessary.

Analytic-based models are another option that can be linear or nonlinear. The basic idea is to collect the reference executed cycles and performance-related parameters, such as dynamic instruction counts, during hardware execution, and apply Machine Learning (ML) algorithms to train the models. The performance-related parameters are selected in such a way that the approach has the lowest dependency on microarchitectural and software details. To predict the performance of new software, a fast functional simulator is used to quickly obtain their performance-related parameters, which are then applied to the trained PMs.

As a free and open-source Instruction Set Architecture (ISA), RISC-V demonstrates considerable potential, particularly for embedded systems in various utilization domains. The RISC-V ecosystem provides a variety of implementations at different abstraction levels, establishing it as a versatile choice for a wide range of applications. In this paper, we propose an ML-based methodology to estimate the performance of embedded software across various microarchitectures. Considering that the real physical hardware may not be available, we employ a combination of fast functional simulator, cycle-accurate RISC-V implementations, and ML techniques to generate PMs. Subsequently, the PMs are combined with a fast functional simulator to enable fast and accurate prediction of the performance of new embedded software. We demonstrate the generalizability of our approach across multiple microarchitectures for the RISC-V ISA in [2][3][4], applying it to a cycle-accurate simulator and four real-world, cycle-accurate RTL cores of the RISC-V ISA. Our evaluation focuses on the performance of our approach using standard benchmarks from TACLeBench [5].

2 Methodology

In this section, we present our proposed approach for performance estimation, which involves coupling a fast functional simulator with ML-based models. An overview of our approach is depicted in Fig. 1. It encompasses two crucial phases: the training phase and the prediction phase. In the subsequent subsections, we will delve into a detailed

---

*This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within projects Scale4Edge under grant no. 16MEO127, ECXL under grant no. 01IW22000 and VE-HEP under grant no. 16KIS1342.
The foundation of our performance analysis is the extraction of dynamic instruction counts and reference executed cycles from runtime information. Dynamic instruction counts efficiently capture software execution dynamics and serve as crucial performance features. These features are then converted into vectors, forming the training dataset for model training. We employed supervised ML algorithms to predict the performance of new software. For RISC-V VP, we generated three models—PM1, PM2, and PM3—based on the different feature vectors. PM1 and PM2 [2] were trained using Linear Regression (LR) algorithm, incorporating the total instruction count and six RISC-V formats counts, respectively. In contrast, PM3 [3] employed the Artificial Neural Network (ANN) for training, utilizing individual instruction counts. For each RTL core, we implemented four classic ML algorithms: Ordinary Least Squares (OLS) regression, LR with Mini Batch Gradient Descent (MGD), ridge regression and ANN. OLS regression and LR with MGD are LR techniques that optimize the model’s parameters differently. Ridge regression is a regularized form of LR that prevents overfitting. ANN is a more complex model capable of capturing nonlinear relationships among variables.

\[ APE = \frac{y - \hat{y}}{y} \times 100\% \]  

where \(y\) and \(\hat{y}\) represent the actual and estimated clock cycle for each benchmark. The APE quantifies the extent of deviation between the real and estimated clock cycle.

To provide a more comprehensive evaluation, the Mean Absolute Percentage Error (MAPE) is computed. This involves summing up the APE values and dividing the sum by the total number of benchmarks.

\[ \text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} |\frac{y_i - \hat{y}_i}{y_i}| \times 100\% \]
Table 1: Experimental Results of all Benchmarks used for Validation of PM1, PM2, and PM3.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#inst-exec.</th>
<th>VP</th>
<th>Ours</th>
<th>PM1</th>
<th>PM2</th>
<th>PM3</th>
</tr>
</thead>
<tbody>
<tr>
<td>aladdin_dec</td>
<td>2830 767</td>
<td>1506 785</td>
<td>1761</td>
<td>241</td>
<td>1332</td>
<td>3927 879</td>
</tr>
<tr>
<td>adpcm_enc</td>
<td>2898 910</td>
<td>3636 185</td>
<td>1751</td>
<td>244</td>
<td>7176</td>
<td>1396 579</td>
</tr>
<tr>
<td>cubic</td>
<td>28 338 774</td>
<td>37 255 979</td>
<td>17 129</td>
<td>2343</td>
<td>7637</td>
<td>38 580 028</td>
</tr>
<tr>
<td>deg2rad</td>
<td>510 732</td>
<td>659 172</td>
<td>332</td>
<td>56</td>
<td>5929</td>
<td>695 312</td>
</tr>
<tr>
<td>fft</td>
<td>3678 523</td>
<td>5020 639</td>
<td>2573</td>
<td>319</td>
<td>8066</td>
<td>5007 945</td>
</tr>
<tr>
<td>gsm_dec</td>
<td>9168 157</td>
<td>12 320 312</td>
<td>5942</td>
<td>779</td>
<td>7628</td>
<td>12 481 536</td>
</tr>
<tr>
<td>isqrt</td>
<td>1002 079</td>
<td>1 838 966</td>
<td>844</td>
<td>123</td>
<td>6862</td>
<td>1364 232</td>
</tr>
<tr>
<td>lstm</td>
<td>5814 944</td>
<td>7430 919</td>
<td>3472</td>
<td>487</td>
<td>7129</td>
<td>7916 470</td>
</tr>
<tr>
<td>rad2deg</td>
<td>420 104</td>
<td>571 911</td>
<td>312</td>
<td>56</td>
<td>3501</td>
<td>571 931</td>
</tr>
<tr>
<td>st</td>
<td>3684 067</td>
<td>4842 675</td>
<td>2319</td>
<td>313</td>
<td>4083</td>
<td>5015 492</td>
</tr>
</tbody>
</table>

APE: Mean absolute percentage error

The application of the ML algorithms and PMs was programmed using Python 3.8. The algorithms were implemented with publicly available libraries, where the OLS regression and ridge regression were implemented using Scikit-learn 1.0 [14] and TensorFlow 2.6.0 [15] was used to implement LR, LR with MGD and ANN.

3.2 Performance and Simulation Analysis

Table 1 presents the experimental results of applying various standard benchmarks to our proposed performance estimation approach for cycle-accurate VP. The first column enumerates the names of the benchmarks, while the subsequent column displays the total instruction counts for each benchmark. Column VP includes the cycle count (#Cycle) and simulation time (Time) reported from VP. Column Ours displays the execution time of our proposed approach and the achieved speedup relative to VP. Considering that each PM requires approximately 0.06 ms to estimate all 10 benchmarks, which is negligible in comparison to the Whisper simulation time, the overall simulation time on Whisper can be interpreted as the time dedicated to estimating the number of cycles for the new software. Columns PM1, PM2, and PM3 represent PMs based on the total instruction count, instruction format counts, and individual instruction counts, respectively. The subcolumns #Cycle and APE denote the number of cycles estimated by PM and the APE of PM compared to VP for each benchmark, respectively. Experimentally, the results show that our approach achieves a speedup up to more than 8× in comparison to the cycle-accurate VP. Additionally, we assessed the quality of each generated PM using the MAPE metric. For the overall benchmarks, PM3 exhibits the best accuracy in performance estimation, with a MAPE of 1.900%. This superior accuracy can be attributed to leveraging more detailed runtime information (i.e., individual instruction counts) and utilizing ANN, which provide a more effective estimation model for capturing non-linear behavior.

In this paper, we present a novel ML-based approach for performance estimation of embedded software across various processors, using RISC-V processors as a case study. Our method utilizes the runtime trace of the software via a fast functional simulator to achieve accurate
performance estimation. We demonstrate its applicability through validation across a cycle-accurate simulator and four real-world, cycle-accurate RTL cores of the RISC-V ISA. The quality of our proposed approach was validated in terms of simulation speed and the accuracy of cycle count predictions, using a set of standard benchmarks.

In the future, we plan to extend this approach to other computer architectures. Furthermore, we intend to explore the use of Large Language Models (LLM) and fine-tune them to adapt our approach for estimating the performance of embedded software, exploring the potential efficiency gains through transfer learning.

5 Literature


