Multiple regression analysis considering multicollinearity for estimating CPU cycles using performance counters

Ryota Hattori

Graduate School of Science and Engineering
Kindai University
Higasi-Osaka, Osaka 577-8502
2233340407d@kindai.ac.jp

Yoshinori Takeuchi

Faculty of Science and Engineering
Kindai University
Higasi-Osaka, Osaka 577-8502
takeuchi@ele.kindai.ac.jp

Abstract— Currently, the development of industrial controller devices requires the estimation of the execution time of control software running on embedded processors. However, embedded processors have the complex functions and functional specifications are black box. Thus, estimating the execution time is difficult. Recently, many studies offer estimating methods of CPU cycles using performance counters as methods for estimating execution time. Since only a limited number of performance counters can be measured at one time, repeated measurements are required in order to get many performance counter values, which take a lot of time. This study proposes multiple regression analysis considering multicollinearity (MRACM) to reduce the number of measurements for estimating CPU cycles. This study compares estimation accuracy of CPU cycles by linear programming (LP), multiple regression analysis (MRA), and multiple regression analysis considering multicollinearity (MRACM). This study discusses the best analytical approach for each program. Experimental results show that MRACM can reduce the number of required performance counters to 2 and estimate CPU cycles within the almost same estimation errors as conventional methods when multicollinearity occurs and counters with high and low correlation coefficients exist.

I. Introduction

Today, control software on embedded processors is used for developing industrial controllers. Since the industrial controllers need to guarantee real time performance, it is important to estimate the execution time of the control software. However, estimating the execution time is limited by the complexity of recent embedded processors.

As a means for observing various characteristics of the CPU architecture related to software execution, recent CPUs have a function called performance counter[1], which measures occurrence events when executing the software. Performance counters can confirm the details regarding the performance when executing the software. Thus, various studies proposed methods for estimating the CPU cycles using performance counters[2][3].

Benchmark software are evaluation programs that assume control software executing on embedded processors. Benchmarks are used for assessing the relative performance among various software and hardware platforms. A number of benchmarks have been proposed including SPEC[4], MediaBench[5], Mibench[6], and WCET (Worst-Case Execution Time) benchmark code[7], which focus on specific areas of computation.

This paper compares estimation of accuracy of CPU cycles by linear programming, multiple regression analysis, and multiple regression analysis considering multicollinearity. In addition, this study discusses the best analytical approach for each program.

The organization of this paper is as follows. Section II explains related work and introduces their limitations. Then, section III explains the performance counter and estimation methods. Section IV explains the evaluation board and benchmarks. Section V conducts some experiments and shows some considerations to estimation methods. Finally, section VI concludes this paper and shows some future work.

II. Related Studies

Various methods for modeling execution performance have been proposed. Tanaka et al. have proposed an execution performance estimation method which use linear programming (LP) for a linear regression model of various performance counter value[2]. This method enables to eliminate inappropriate results such as sign inversion that occurred in the least squares estimation, and to obtain a more accurate coefficient under constraints on the sign of the coefficient. Reference[3] proposes multiple regression analysis considering multicollinearity (MRACM) and multiple regression analysis (MRA) as an execution performance estimation method. These methods can select performance counters that are strongly related to estimating execution performance.

References[2][3] propose to adopt the outputs of the CPU performance counter as the factors in the estimation model. These studies measure performance counter values and use performance analysis methods to create estimation models. Performance counter values can fully reflect both CPU architecture and target software, which is supposed to facilitate the performance estimation and improve the estimation accuracy.

This study compares estimation accuracy of CPU cycles by LP, MRA, and MRACM, and discusses the best analytical approach for each program.

III. Performance counter and Estimation methods

This study uses execution performance estimation methods with various performance counter value. This section explains the performance counter on the CPU and estimation methods.

A. Performance counter

The performance counter is an embedded circuit for measuring internal event counts during software execution in a general commercial CPU. It is widely implemented from high-end CPUs like Intel processors, PowerPCs, to embedded CPUs like ARM Cortex processor. It is used to collect data on processor operation, estimate power consumption, and detect malware[8][9][10].

In this paper, NXP LA1021A processor is used for the performance monitoring. NXP LS1021A includes an ARM CortexA7 Multicore IP. TABLE I shows typical events that can be measured by the performance monitoring unit (PMU). Event name indicates the name of each performance counter. Description indicates the feature of each performance counter. ARM Cortex-A7 core can measure a total of 44 events. However, this processor can measure only 4 events at once. Therefore, in order to get all performance counter values, 11 measurements are required for each benchmark program.

TABLE I
Cortex-A7 PMU Events (example)[1]

Event Name	Description
INST_RETIRED (IR)	Number of issued instructions
PC_WRITE_RETIRED (PWR)	Number of writes to program counter
BR_IMMED_RETIRED (BIR)	Number of immediate instruction executions
BR_RETURN_RETIRED (BRR)	Number of return instruction executions
BR_MIS_PRED (BMR)	Number of branch prediction errors
BR_PRED (BP)	Number of branch prediction errors
CPU_CYCLES	Cycle
L1I_CACHE (L1IC)	Level 1 instruction cache access
L1D_CACHE (L1DC)	Level 1 data cache access
MEM_ACCESS (MA)	Data memory access
L1D_CACHE_REFILL (L1DCR)	Level 1 data cache refill
L2D_CACHE_WB(L2DCW)	Level 2 data cache clear
L2D_CACHE_REFILL(L2DCR)	Level 2 data cache refill

B. Estimation methods

This section explains estimation methods used in this study.

B.1. Linear programming (LP)

This estimation method is a software performance model that linearly combines multiple performance counter values and introduces sign constraints on model parameter values. Execution performance may calculate negative coefficients due to the increase in performance impact. Thus, introducing sign constraints can set the coefficients to positive values. This method can eliminate the improper result of sign reversal of parameter values that occurs in the general least squares method and estimate the performance model with high validity.

A model formula for performance estimation is as follows. Let the estimated CPU cycles be $\hat{c_i}$. $\hat{c_i}$ is modeled using eq (1).

$$\hat{c_i} = x_I + \sum_{i=1}^{N} x_i \cdot E_i \tag{1}$$

where E_i indicates the values of various events measured by PMU, x_i(i = 1, . . . , N) indicates the coefficient parameters for each event, and x_I indicates the intercept (offset).

The estimation method by the linear programming method is defined as follows:

$$\begin{aligned}
&\textit{minimize :} && \textstyle\sum_{i=1}^{M} \varepsilon_i \\
&\textit{subject to :} && \varepsilon_1 = && c_1 - (x_I + \sum_{i=1}^{N} x_i \cdot E_i^1) \geq 0, \\
& && \varepsilon_2 = && c_2 - (x_I + \sum_{i=1}^{N} x_i \cdot E_i^2) \geq 0, \\
& && \cdots \\
& && \varepsilon_M = && c_M - (x_I + \sum_{i=1}^{N} x_i \cdot E_i^M) \geq 0, \\
& && x_I \geq 0, && x_i \geq 0 \, (i = 1, \cdots, N)
\end{aligned}$$

where c_j indicates the j-th measured CPU_CYCLE event value, and E_i^j indicates the j-th measured E_i event value which is indexed by i. Each ε_j is a residual error for estimated CPU cycles and x_i (i = 1, . . . , N) indicates the coefficient parameters for each event. The intercept x_I , which is the offset value, is introduced by assuming the pipeline operation of the CPU.

B.2. Multiple regression analysis (MRA)

This estimation method is a statistical data analysis. This method indicates the influence of several explanatory variables related to the objective variable and quantifies in the form of a function. In this study, CPU cycles are used as the objective variable and performance counter values are used as explanatory variables to estimate CPU cycles.

A model formula for performance estimation is eq (1). The estimation method by the multiple regression analysis is defined as follows:

$$minimize: \quad \sum_{i=1}^{N}(c_i - \hat{c_i})^2$$

where c_i is the i-th measured CPU cycles and $\hat{c_i}$ is the i-th estimated CPU cycles.

B.3. Multiple regression analysis considering multicollinearity (MRACM)

Multicollinearity indicates the presence of a combination of explanatory variables with high correlation coefficients. Although multicollinearity does not directly affect estimation accuracy, it can cause errors in the model coefficients, making the model unstable and potentially generating inaccurate forecasts. Therefore, when selecting explanatory variables, it is necessary to select only the necessary explanatory variables without using redundant information that could lead to multicollinearity. This is important for accurate and stable models. Variance inflation factor (VIF) is used to detect multicollinearity among variables.

This method finds the correlation coefficient of the selected counters of the selected counters by multiple regression analysis. When counters with high and low correlation coefficients occur, each counter is combined to estimate CPU cycles. The counter combination with the highest estimation accuracy is selected.

IV. Evaluation environment

This section explains the evaluation board and benchmarks used in this study.

A. Evaluation board

This study uses NXP QorlQ LS1021A Tower System Module. This evaluation board is one of the typical embedded CPU designed for a wide range of industrial applications. The evaluation board is equipped with two ARM Cortex-A7 cores and a branch predictor. As for the cache size, the instruction and data L1 cache is 32KB and the L2 cache is 512KB. In addition, the evaluation software was compiled as bare metal for this CPU, and the generated program was loaded and executed directly from the boot loader. Using bare metal can eliminate the impact of timer interrupt process.

TABLE II
MiBench Benchmarks[6]

Category	Program name	Description
Telecomm	FFT	Fast Fourier Transform and its inverse transform on an array of data
Auto/Industrial	basicmath_rad2deg	Angle conversions
Network	dijkstra	The shortest path problem

TABLE III
WCET benchmark[7]

Program name	Description
fibcall	Iterative fibonacci, used to calculate fib
cnt	Counting non-negative numbers in a matrix
duff	Unstructured loop

B. Benchmarks

This study uses Mibench and WCET benchmark as evaluation software. Mibench is a free benchmark for embedded systems. It ties to capture the application diversity of the embedded system area and consists of 35 embedded applications from six categories. The six categories are Automotive and Industrial Control, Consumer Devices, Office Automation, Networking, Security, and Telecommunications. TABLE II shows programs used in this study.

The WCET benchmark software are provided by the Mälardalen WCET benchmarks[7]. The WCET benchmark code is that it is useful for performing the worst case execution time (WCET) evaluation required for the controller, and software having various execution time characteristics are provided. TABLE III shows programs used in this study. This study selects six different programs with complex processing.

V. Experimental Results

This section confirms the availiability of multiple regression analysis considering multicollinearity for each program and compares the estimation accuracy results of CPU cycles by linear programming (LP), multiple regression analysis (MRA), and multiple regression analysis considering multicollinearity (MRACM).

A. Benchmark Results

This study defines the TEST# for each benchmark software. TEST# is a program in which the matrix size and the number of iterations are changed in the same program code. The names of TEST#, like TEST1, TEST2, are determined according to the order of the number of cycles of programs.

This study uses data from TEST1 to TEST10 to estimate the coefficients of the parameters of the performance model.

A.1. FFT Benchmark Result

TABLE IV shows the performance counters selected for MRA and the correlation coefficient between each performance counter and CPU cycles. TABLE IV indicates that the selected performance counters have some counters with high correlation coefficients and others with low correlation coefficients. Selected event indicates the selected performance counters and correlation coefficient indicates result of correlation coefficient between each performance counter and CPU cycles. TABLE IV indicates that the counter with the lowest correlation coefficient is L1D_CACHE_REFILE. Fig.1 shows the estimation accuracy of CPU cycles when L1D_CACHE_REFILE is combined with counters with a high correlation coefficient. In Fig.1, horizontal axis indicates counter combination and vertical axis indicates the error rate of CPU cycles for each TEST. From Fig1, MRACM selects a combination of INST_RETIRED and L1D_CACHE_REFILE. The value of VIF is 1.08 and this combination has no multicollinearity effect. Thus, this benchmark uses LP, MRA, and MRACM to estimate CPU cycles.

TABLE IV
SELECTED PERFORMANCE COUNTERS AND CORRELATION COEFFICIENTS (FFT)

Selected event	Correlation coefficients
INST_RETIRED	0.99
PC_WRITE_RETIRED	0.99
BR_IMMED_RETIRED	0.99
BR_RETURN_RETIRED	0.99
BR_MIS_PRED	0.99
BR_PRED	0.99
L1I_CACHE	0.99
L1D_CACHE	0.99
L1D_CACHE_REFILL	0.11

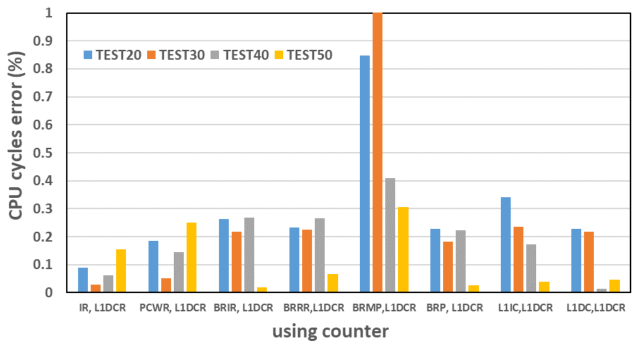

Fig. 1. Combined results of performance counters (FFT)

A.2. basicmath_rad2deg Benchmark Result

TABLE V shows the performance counters selected for MRA and the correlation coefficient between each performance counter and CPU cycles. Due to only counters available with high correlation coefficients generated, MRACM cannot be used in this benchmark. Thus, this benchmark uses LP and MRA to estimate CPU cycles.

TABLE V
SELECTED PERFORMANCE COUNTERS AND CORRELATION COEFFICIENTS (BASICMATH_RAD2DEG)

Selected event	Correlation coefficients
INST_RETIRED	0.99
PC_WRITE_RETIRED	0.99
BR_IMMED_RETIRED	0.99
BR_RETURN_RETIRED	0.99
BR_MIS_PRED	0.99

A.3. dijkstra Benchmark Result

TABLE VI shows the performance counters selected for MRA and the correlation coefficient between each performance counter and CPU cycles. Due to only counters available with high correlation coefficients generated, MRACM cannot be used in this benchmark. Thus, this benchmark uses LP and MRA to estimate CPU cycles.

A.4. fibcall Benchmark Result

TABLE VII shows the performance counters selected by MRA and the correlation coefficient between each performance counter and CPU cycles. TABLE VII indicates that the counter with the lowest correlation coefficient is BR_MIS_PRED. Fig.2 shows the estimation accuracy of CPU cycles when BR_MIS_PRED is combined with counters with a high correlation coefficient. In Fig.2, horizontal axis indicates counter combination and vertical axis indicates the error rate of CPU cycles for each TEST. From

TABLE VI
NUMBER OF SELECTION COUNTERS AND ESTIMATION RESULTS OF CPU CYCLES(DIJKSTRA)

Selected event	Correlation coefficients
INST_RETIRED	0.99
PC_WRITE_RETIRED	0.99
BR_IMMED_RETIRED	0.99
BR_RETURN_RETIRED	0.99
BR_MIS_PRED	0.99
L1I_CACHE	0.99
L1D_CACHE	0.99
MEM_ACCESS	0.99
L1D_CACHE_REFILL	0.99

Fig.2, MRACM selects a combination of INST_RETIRED and BR_MIS_PRED. The value of VIF is 1.15 and this combination has no multicollinearity effect. Thus, this benchmark uses LP, MRA, and MRACM to estimate CPU cycles.

TABLE VII
SELECTED PERFORMANCE COUNTERS AND CORRELATION COEFFICIENTS (FIBCALL)

Selected event	Correlation coefficients
INST_RETIRED	0.99
PC_WRITE_RETIRED	0.99
BR_MIS_PRED	0.37

A.5. cnt Benchmark Result

TABLE VIII shows the performance counters selected for MRA and the correlation coefficient between each performance counter and CPU cycles. TABLE VIII indicates that the counter with the lowest correlation coefficient is L2D_CACHE_WB. This study estimates accuracy of CPU cycles when L2D_CACHE_WB is combined with counters with a high correlation coefficient. From the same method used in FFT and fibcall, MRACM selects a combination of L2D_CACHE_REFILE and L2D_CACHE_WB. The value of VIF is 1.14 and this combination has no multicollinearity effect. Thus, this benchmark uses LP, MRA, and MRACM to estimate CPU cycles.

A.6. duff Benchmark Result

TABLE IX shows the performance counters selected for MRA and the correlation coefficient between each performance counter and CPU cycles. TABLE IX indicates that the counter with the lowest correlation coefficient is L2D_CACHE_WB. This study estimates accuracy of CPU cycles when L2D_CACHE_WB is combined with counters with a high correlation coefficient. From the same method used in FFT and fibcall, MRACM selects a combination of and INST_RETIRED and L2D_CACHE_WB. The value of VIF is 1.03 and this combination has no multicollinearity effect. Thus, this benchmark uses LP, MRA, and MRACM to estimate CPU cycles.

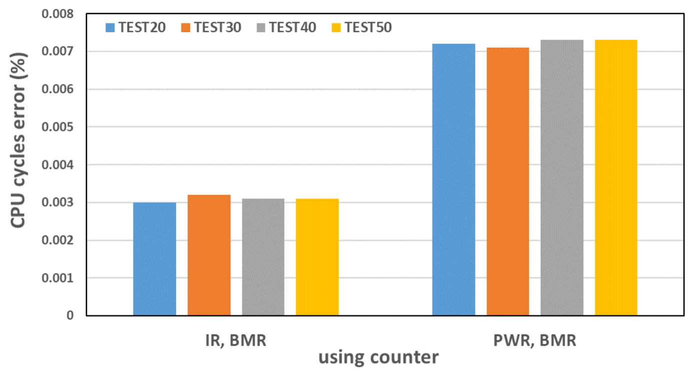

Fig. 2. Combined results of performance counters (fibcall)

TABLE VIII
SELECTED PERFORMANCE COUNTERS AND CORRELATION COEFFICIENTS (CNT)

Selected event	Correlation coefficients
INST_RETIRED	0.99
PC_WRITE_RETIRED	0.99
BR_IMMED_RETIRED	0.99
L1I_CACHE	0.99
L1D_CACHE_REFILL	0.99
L2D_CACHE_WB	0.35
BUS_ACCESS	0.99
L2D_CACHE_REFILL	0.99
LINEFILL_PREFETCH	0.99

B. *Results of CPU cycle estimation accuracy*

This section presents results of CPU cycle estimation accuracy by LP, MRA, and MRACM in FFT, basicmath_rad2deg, dijkstra, fibcall, cnt and duff.

B.1. Mibench and WCET Comparison Results

TABLE X shows the number of selected counters and the error rate of estimated CPU cycles at TEST50 in FFT, basicmath_rad2deg, dijkstra, fibcall, cnt, and duff. No. of counters indicates number of selected performance counters. Error indicates the error rate at TEST50 when using each analysis method. NA indicates that MRACM is not

TABLE IX
SELECTED PERFORMANCE COUNTERS AND CORRELATION COEFFICIENTS (DUFF)

Selected event	Correlation coefficients
INST_RETIRED	0.99
L1I_CACHE	0.99
L1D_CACHE_REFILL	-0.31
L2D_ CACHE	-0.40
L2D_CACHE_WB	0.16
BUS ACCESS	0.96
BUS_ACCESS RD	-0.084
ENT_RD_ALLOC_MODE	0.27

applicable. Error is defined as follows:

$$Error = (\frac{\hat{c} - c}{c}) \times 100$$

where c is the measured CPU cycles and ĉ is the estimated CPU cycles.

TABLE X
NUMBER OF SELECTED COUNTERS AND ESTIMATION ERROR OF CPU CYCLES AT TEST50 (FFT, BASICMATH_RAD2DEG, DIJKSTRA, FIBCALL, CNT, DUFF)

benchmark		LP	MRA	MRACM
FFT	No. of counters	6	9	2
	Error [%]	0.011	0.12	0.15
basicmath_rad2deg	No. of counters	2	5	NA
	Error [%]	0.0051	0.0024	-
dijkstra	No. of counters	6	9	NA
	Error [%]	0.012	2.02	-
fibcall	No. of counters	4	3	2
	Error [%]	0.00000029	0.00000012	0.0031
cnt	No. of counters	4	9	2
	Error [%]	0.080	0.25	0.11
duff	No. of counters	7	8	2
	Error [%]	0.012	0.053	0.021

From TABLE X, MRACM provides highly accurate estimation with a small number of counters in FFT, fibcall, cnt, and duff. MRACM can estimate with high accuracy in benchmarks with various characteristics.

LP provides highly accurate estimation in dijkstra. MRA provides highly accurate estimation in basicmath_rad2deg.

VI. CONCLUSIONS

When multicollinearity occurs and counters with high and low correlation coefficients exist, MRACM provides highly accurate estimation with a small number of counters and estimate with high accuracy in benchmarks with various characteristics. Although the above conditions do not meet, traditional LP or MRA method can be applied.

The future work will study a performance analysis method of a small number of performance counters with higher estimation accuracy for various benchmarks, inplement more complex programs and estimate CPU cycles.

REFERENCES

[1] " Performance Monitoring Unit, " Cortex-A7 MPCore Technical Reference Manual (Revision F), Chapter 11, ARM, 2013. https://developer.arm.com/documentation/ddi0464/f/performance-monitoring-unit (Last access: Oct. 12, 2023)

[2] Teruaki Tanaka, Masanori Hashimoto and Yoshinori Takeuchi, " Linear Programming Based Reliable Software Performance Model Construction with Noisy CPU Performance Counter Values, " SASIMI 2021, pp. 45–50, 2021.

[3] Ryota Hattori, Kosuke Kohara and Yoshinori Takeuchi, "Evaluation and estimation of CPU cycles by multiple regression analysis considering multicolinearity, " (in Japanese), Technical report of IPSJ, vol. 2022-EMB-61, no. 6, pp. 1-7, 2022.

[4] B. Case, " SPEC2000 Retires SPEC92", The Microprocessor Report, vol. 9, 1995.

[5] C. Lee, M. Potkonjak and H. Mangione-Smith, " MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems," Micro-30, November 1997.

[6] M.R. Guthaus, J.S. Ringenberg, D. Ernest, T.M. Austin, T. Mudge and R.B. Brown, " MiBench: A free, commercially representative embedded benchmark suite, " Proc. of the IEEE Int'l Workshop on Workload Charac-terization (WWC-4), Dec. 2001.

[7] 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), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

[8] Rance Rodrigues, Arunachalam Annamalai, Israel Koren and Sandip Kundu, " A Study on the Use of Performance Counters to Estimate Power in Microprocessors, " IEEE Transactions on Circuits and Systems II: Express Briefs, pp. 882–886, 2013.

[9] Jordan Pattee and Byeong Kil Lee,"Design Alternatives for Performance Monitoring Counter based Malware Detection, " 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), pp. 1–2, 2020.

[10] Reza Azimi, David K.Tam, LivioSoares and Michael Stumm, " Enhancing Operating System Support for Multicore Processors by Using Hardware Performance Monitoring, " ACM SIGOPS Operating Systems Review, Volume 43, Issue 2, pp. 56–65, 2009.