International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.03, April 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Performance Prediction of embedded system at Source Level

Asst. Prof. Pooja Thakre[1], Gaurav Deshmukh[2], Akshay Wakle[3], Mahesh Bhakare[4], Mayur Ghare[5]

[1,2,3,4,5]*Department of Computer Engineering,*

Savitribai Phule Pune university, G. S. Moze College of Engineering, Pune

ABSTRACT

Developers need performance prediction tools that are capable of providing information on the future performance of the embedded system. This paper describes a performance analyser tool developed to predict the performance. we have implement to simpler, realistic and implementable analytical models based on the sound principles of Performance Engineering and Regression Techniques. System design is for reducing the turnaround time of software development. Also reducing the turnaround time after the modification of the source code due to changes in problem specification. predicting the performance of application software at source code level using comprehensive method that combines analytical modeling and statistical approach. We take samples from EEMBC and SMV benchmarks and gather the static attributes from the source code of those samples as our learning set. We then apply multiple linear regression technique enhanced with statistical tool SPSS23 to predict the performance of these functions.

Keywords **– Performance prediction; multiple linear regression; analytic model; source code level**

I.INTRODUCTION

In this work, we use predictive model to predict the performance. which uses a comprehensive method that combines analytical modeling and statistical approach. Performance prediction is critical in embedded system design for reducing the turnaround time of software. Using simulation to measure the performance of the whole source code is often too slow, particularly after the modification of the source code due to changes in problem specification. In this paper we present a comprehensive method that combines analytical modeling and statistical approach to predicting the performance of application software at source code level.

The analytical performance model represent light-weight mathematical models which are easy to implement. These models predict performance metrics of enterprise software systems, namely throughput, maximum concurrent user load, response time, and CPU utilization for anticipated loads in future with acceptable deviation and minimal data collected from a scaled-down environment. These models can reduce the effort required to run several incremental load tests and also help save on licensing costs of load simulation tools. The validation of these models using data from real-world enterprise applications is currently in-progress. The analytic model on which this paper is partially based is a modification of our earlier work on DSP processors

[2]. The model is modified by introducing a new parameter, a product term of *TNinner * Linner*, where TNinner is the number of statements and Linner represnets the number of total iterations of the innermost loop of source code. Over the past decade, several investigators used statistical approach to predict software performance with certain success [3, 4, and 5]. They used the hardware attributes of different processors to predict the performance of software that ran on them. To enhance our original analytic model, we take into consideration of the static attributes of the source code and make use of statistical techniques, and some new approaches to predict the performance.

Group of EEMBC and all eight functions of SMV benchmarks as the samples of learning set. EEMBC benchmark is an industry standard for embedded processors and software developed by the Embedded Microprocessor Benchmark Consortium, a non-profit organization [6]. SMV benchmark is developed by us in 2006 [7] which consists of eight kernels chosen from the Selectable Mode Vocoder (SMV) application program for 3G wireless communications. We then gather the static attributes from the source code of those samples, conduct multiple linear regressions analysis by using a popular statistical tool SPSS23.

II.OVERVIEW OF PERDECTIVE MODELING

In this work, we use predictive models that reducing the turnaround time of software development by new execution time prediction methodology based on a statistic modeling method, called *Multiple Linear Regression*. The performance model presented below can be used to predict throughput of a given software system. The execution time of a program is mainly determined by its attributes such as the number of total iterations of innermost loop, the total number of statements of innermost loop, the total number of function calls, and the total number of branches. Multiple regression analysis is a statistical technique for investigating and modeling the relationship between two or more independent variables and a dependent variable by a linear equation with a set of observations. In this model, we have n observations y=y1,…,yn called the response variables and xi=xi,1,…,xi,p for i=1..n that are predictor or regressor variables. very value of the independent variable *x* is associated with a value of the dependent variable *y*. Formally, given *n* bservations, the model for *multiple linear regression* of *p* independent variables is

$$yi = a0 + \sum_{j=1}^{p} ajxji + ei \qquad for\ i = 1,2,\ldots n$$

and

$uy = a0 + \sum_{j=1}^{p} a_j xj$ is the so called *population regression line.* There have been different approaches to estimate regression model (to estimate *b0, b1... bp*). The most popular one is leastsquares modeling, in which the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical variations from each data point to the line. Because the variations are first squared, then summed, there are no cancellations between positive and negative values. The least squares estimating *b0, b1 ... bp* are usually computed by statistical software, such as IBM SPSS.

The estimated regression model is also referred to as the *fitted model*. The observations, *yi*, may be different from the fitted values $zi = b0 + \sum_{j=1}^{p} b_j x_{ij}$ obtained from this model. The difference between these two values is the variation, *vi = yi - zi*.

The simple way to predict the execution time is using TNinner * Linner, where TNinner is the number of total iterations of innermost loop and Linner is the total number of statements of the source code of the innermost loop. However, we observed that the errors of the predicted results are quite large. Thus, we propose a comprehensive approach to predicting the execution time based on multiple linear regression method.

To construct the multiple line ar regression model, we use statistical tool and 16 attributes. Statistial tool like IBM SPSS23(Statistical Package for the Social Sciences) is use to conduct multipal linear regression. The current versions (2015) are officially named IBM SPSS Statistics. SPSS generates a multiple variable linear equation as shown below:

$$Ypredicted = b0 + \sum_{j=1}^{p} bjXj$$

Where,

Ypredicted =Dependent variable

X1, X2, ..Xp=Independent variable

b0, b1, b2,... bp =Coefficients generated by SPSS.

If the range of attributes are large,then prediction error occur. To overcome this problem we use RT(Repeating time) variable. It shrink the range of execution time and obtain better prediction accuracy. we run each of them RT time to obtain adjusted execution time. SPSS is used to conduct multiple linear regression, which gives best RT each sample in learning set.As a result, average relative error reduced to minimum.

APETk is the Adjusted Predicted Execution Time for the *k*-th sample in the testing set. Following equation can be use to predict *APETi* the Adjusted Predicted Execution Time for the *i*-th sample in the learning set.

$$APETi = (b0 + \sum_{j=1}^{p} bjXji) + bRTRTi$$

The Predicted Execution Time can be calculated by following equation.

$$PETk = APETk / RTk$$

APETi, PETk and *RTk* is an important factor that can be determined by using the product of TNinner * Linner and some heuristics. RTtesting is the set of RTk of all samples in the testing set, which comes from the analytical model. Some samples' RTk will be heuristically adjusted later.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.03, April 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

III.LEARNING SET AND TESTING SETS FOR MULTIPLE LINEAR REGRESSION

We used *gprof*, a popular profiling tool in UNIX to select eight most frequent executed kernel samples in PHY benchmark to form the testingset for performance prediction. We gather 16 static attributes from the source code of the learning and testing sets as shown in Table 2. Below is the list

Table 1 Attributes of Learning set and Testing set

Benchmark	Function name	Attributes																Execution Time
		Number of	No. of iterations in outer	No. of iterations in inner	Number of statements in outer	Number of statements in inner	Number of loops	levels of nested	total iterations	CC*	function calls	branches	Number of inputs	Number of outputs	Sizes of input data (byte)	Sizes of output data (byte)	$TN_{inner} * L_{inner}$	
EEMBC	Autocorrelation - Data1	5	8	12	3	2	1	2	100	3	0	0	2	1	16	8	200	509
	Autocorrelation - Data2	5	16	1017	3	2	1	2	16272	3	0	0	2	1	1024	16	32544	49193
	Autocorrelation - Data3	5	32	485	3	2	1	2	15520	3	0	0	2	1	500	32	31040	47297
	Convolutional Encoder - Data1	13	1024	5	5	3	2	3	5120	5	0	1	5	1	512	512	15360	42014
	Convolutional Encoder - Data2	13	1024	4	5	3	2	3	4096	5	0	1	5	1	512	512	12288	41501
	Convolutional Encoder - Data3	13	1024	3	5	3	2	3	3072	5	0	1	5	1	512	512	9216	40297
	Fixed-point Bit Allocation - Da	17	80	256	4	11	1	2	20480	7	0	3	6	1	256	256	225280	159448
	Fixed-point Bit Allocation - Da	17	70	20	4	11	1	2	1400	7	0	3	6	1	20	20	15400	20968
	Fixed Point Bit Allocation - Da	17	130	100	4	11	1	2	13000	7	0	3	6	1	100	100	143000	114856
	FFT/IFFT - Data1	22	8	128	5	14	3	3	1024	5	0	0	2	2	2048	5500	14336	45218
	FFT/IFFT - Data2	22	8	128	5	14	3	3	1024	5	0	0	2	2	2525	5635	14336	45218
	FFT/IFFT - Data3	22	8	128	5	14	3	3	1024	5	0	0	2	2	2717	3549	14336	45218
	Viterbi Decoder - Data1	##	42	128	9	48	1	1	5376	17	8	3	1	1	1196	1196	258048	233467
	Viterbi Decoder - Data2	##	42	128	9	48	1	1	5376	17	8	3	1	1	2109	2109	258048	233478
	Viterbi Decoder - Data3	##	42	128	9	48	1	1	5376	17	8	3	1	1	2034	2034	258048	233483
	Viterbi Decoder - Data4	##	42	128	9	48	1	1	5376	17	8	3	1	1	2395	2395	258048	233483
SMV	FLT_filterAP_fx	11	170	9	8	3	1	2	1530	3	0	0	4	2	724	4	4590	3197
	LPC_Chebps_fx	16	1	3	8	7	1	1	3	1	0	0	3	1	16	2	21	56
	LPC_autocorrelation_fx	9	17	240	3	7	1	2	4080	4	2	0	5	1	622	34	28560	7965
	FCS_Excit_Enhance_fx	31	80	20	1	6	3	2	1600	12	1	0	8	1	398	340	9600	12857
	LSF_Q_New_ML_search_fx	30	1152	10	14	4	5	3	11520	32	0	3	11	3	2900	380	46080	45729
	c_fft_fx	34	6	128	4	16	3	3	768	15	0	1	3	1	512	256	12288	10565
	FCB_add_sub_contrib_phi	19	1	8	9	10	1	1	8	6	0	2	9	1	3458	4	80	243
	PIT_LT_Corr_Rmax_fx	45	68	80	31	2	3	3	5440	10	0	0	7	3	848	8	10880	21079
PHY	cholsolve_4xX_complex	34	2	3	5	23	2	3	6	11	8	0	3	1	128	64	138	702
	ant_comb	29	1	1200	0	29	1	1	1200	8	7	0	4	1	4800	4800	34800	479246
	mf	10	1	1200	0	9	2	1	1200	3	1	0	5	1	9600	4800	10800	70431
	soft_demap	18	43200	6	1	14	1	2	3E+05	8	2	1	5	1	4800	4800	4E+06	10374917
	ifft	40	10	700	17	17	1	3	7000	5	0	0	3	1	2400	2400	119000	110922
	chest	10	3	300	0	6	2	1	900	4	0	0	5	1	4800	4800	5400	3423
	fft	38	10	700	17	17	1	3	7000	5	0	0	3	1	2400	2400	119000	109975
	matrix_a_a_herminte_plus_b	60	3	3	1	50	3	1	9	21	11	0	5	1	128	64	450	3143
	matrix_mult_4xX_complex	27	3	3	1	26	1	2	9	8	7	0	4	1	128	64	234	1569

IV.RELATED WORK

There have been numerous works done in the area of performance modelling. [1] use regression models to predict performance and power usage of the applications found in the SPECjbb and SPEC2000 benchmarks. As in the previous reference, the data points are created using simulations. Kahn et al. [10]this paper described the use of model trees for performance analysis. [3,4, 5] gives idea about predict the performance of source code running on different hardware. [8] proposes cross-architecture performance prediction. It is a machine learning based technique using both static and dynamic attributes from many programs from some benchmarks and learning set of different input data from telecommunication group of EEMBC benchmark [6] and eight samples from SMV benchmark [7]. [11] have presented an analytical approach to the design space exploration of caches that avoids exhaustive simulation.

V.CONCLUSION

In this work, we use predictive model to predict the performance. which uses a comprehensive method that combines analytical modeling and statistical approach. We use the multiple linear regression method with the static attributes at source code level with statistical tool SPSS23 to predict the execution time of some typical DSP functions. Another technique that can be used for Performance Modeling will be using Software Simulation. Some simulation software products are available as COTS (commercial-off-the-shelf) products. These are licensed, so prediction accuracy comes at a cost and hence the adoption is slower by the organizations. As a result, these models will reduce turnaround time and development cost. Also provide reducing the turnaround time after the modification of the source code due to changes in problem specification. Predictions are more accurate for systems that are tuned for performance and scalability.

REFERENCES

[1] Lee B. C. and Brooks D. M. Accurate and Efficient Regression Modeling form Microarchitectural Performance and Power Prediction. In Proc. of the ASPLOS, Oct 2006, San Jose, CA.

[2] B. Su et al., A New Source-Level Benchmarking for DSP Processors, Proc. of ISPC2003, 2003

[3] E. Ould-Ahmed-Vall, J. Woodlee, C. Yount, and K. Doshi, On the Comparison of Regression Algorithms for Computer Architecture Performance Analysis of Software Applications, SMART'07, 2007

[4] B. Oziskikyilmaz, G. Memik, and A. Choudhary, Machine Learning Models to Predict Performance of Computer System Design, Proc. of 37th International Conference on Parallel Processing

[5] F. Eichinger, D. Kramer, K. Bohm, and W. Karl, From Source Code to Runtime Behaviour: Software Metrics Help to Select the Computer
Architecture, Proc. of 29th SGAI Int. Conf. on Artificial Intelligence, 2009

[6] TELE BENCH, an EEMBC Bench, http://www.eembc.org/benchmark/telecom_sl.php

[7] E. Hu C. Ku, A. Russo, B. Su, and J. Wang, "New DSP Benchmark based on Selectable Mode Vocoder (SMV)," Proc. of the 2006 International

Conference on Computer Design, pp.175-181, June 2006

[8] N. Ardalani, C. Lestourgeon, K. Sankaralingam, and X. Zhu, Cross- Architecture Performance Prediction (XAPP) Using CPU Code to Predict GPU Perfromance, Proc. of MICRO-48, Dec. 2015

[9] Madhu Tanikella (Infosys) "Future-ready IT Systems with Performance Prediction using Analytical Models"

[10] ElMoustapha Ould-Ahmed-Vall and James Woodlee and Charles Yount and Kshitij A. Doshi and Seth Abraham Using Model Trees for Computer Architecture Performance Analysis of Software Applications

[11] Ghosh A. and Givargis T. Analytical Design Space Exploration of Caches for Embedded Systems. In the Proc. of the DATE Conference. Mar 2003, Munich Germany.