Energy Prediction of OpenMP Applications using Random Forest Modeling Approach

Shajulin Benedict, Rejitha R.S.
HPCCLoud Research Laboratory,
SXCCE, Anna University,
Nagercoil, India, Email: shajulin,rejitha@sxcce.edu.in

Philipp Gschwandtner, Radu Prodan, and Thomas Fahringer
Institute of Computer Science,
University of Innsbruck, Austria,
Email: philipp,radu,tf@dps.uibk.ac.at

***Abstract*—OpenMP, with its extended parallelism features and support for radically changing HPC architectures, spurred to a surge in developing parallel applications among the HPC application developers community, leading to severe energy consumption issues. Consequently, a notion of addressing the energy consumption issue of HPC applications in an automated fashion increased among compiler developers although the underlying optimization search space could increase tremendously. This paper proposes a Random Forest Modeling (RFM) approach for predicting the energy consumption of OpenMP applications in compilers. The approach was tested using OpenMP applications, such as, NAS benchmarks, matrix multiplication, n-body simulations, and stencil applications while tuning the applications based on energy, problem size, and other performance concerns. The proposed RFM approach predicted the energy consumption of code variants with less than 0.699 Mean Square Error (MSE) and 0.998 R^2 value when the testing dataset had energy variations between 0.024 joules and 150.23 joules. In addition, the influences of energy variations, number of independent variables used, and the proportion of testing dataset used during the RFM modeling process are discussed.**

***Keywords*-Energy Prediction; HPC; Modeling; OpenMP; Scientific Applications**

I. Introduction

Developing HPC applications uprooted heavily among the minds of application developers due to the paradigm shift in recent architectures - most of the CPU vendors moved away from improving CPU clock speed to adding multi-cores on chips. Hence, application developers are urged to write efficient parallel algorithms considering the hardware parallelism of machines and various other performance concerns, including energy.

OpenMP, with its simplest programming approach (using OpenMP constructs) and its support to the varying HPC architectures, has widely attracted application developers from diverse fields, such as, High Energy Physics, cloud manufacturing sector, and so forth. In succinct, the HPC application developers community preferably opts for writing their parallel applications using OpenMP constructs [3].

However, due to the emerging unskilled HPC application developers and architecture-application mismatches, applications could easily lead to performance concerns, including the energy consumption issue. For instance, an application developer could develop OpenMP applications with too much sequential code or too many fine granular parallel codes. In fact, tradeoffs exist in terms of optimization solutions when a specific performance problem would be addressed. Interestingly, these optimization possibilities are comparatively increasing from time to time due to the muddled up complicated architectures and varying application requirements. Thus, a need for an auto-tuning mechanism for compilers is emerging as a mandatory solution for HPC application developers.

In general, an auto-tuning mechanism of compilers could be assisted using predictive modeling solutions so that a wide coverage of solutions in an optimization search space is possible with limited executions.

This paper proposes an energy prediction mechanism of OpenMP applications using a Random Forest Modeling (RFM) approach for compilers. RFM is a bagging tree based modeling mechanism. The proposed energy prediction mechanism was developed at the HPCCLoud Research Laboratory, India. To validate the proposed mechanism, the training and testing data are gotten after experiments were remotely conducted on the Thomson machine of the University of Innsbruck, Austria. Several HPC applications, such as, NAS benchmarks, n-body simulations, stencil computations, and a matrix multiplication application, were predicted for various problem sizes. In succinct, the RFM mechanism was used to predict the energy consumption of various problem sizes of applications.

The rest of the paper is organized as follows. Section II presents the existing energy prediction mechanisms. Section III explains the need for an energy prediction mechanism and a proposed energy prediction architecture for compilers and Section IV explains the proposed RFM mechanism. Section V validates the proposed approach using OpenMP-based applications. And, finally, Section VI presents a few conclusions.

II. Related Work

There have been several attempts to predict the execution time of sequential applications employing a variety of

methods (e.g. combining application models with machine profiles [7], [22], [25], analytical models [2], [6], [29], statistical models [4], hybrid analytical and statistical methods [8], historical data [16], data mining methods [16], [21], queuing theory [27], partial program executions [28], simulation [24], and skeleton [23]). Similarly, the performance of full applications has been analytically modelled in [14], [17]. Numerous efforts use [16] machine learning to predict task execution times, the effects of compiler transformations [10], schedulingand the performance of networks.

In terms of energy modeling, technologies such as Intel Speedstep and AMD CoolnQuiet / PowerNow involve scaling the operational frequency and voltage of processors. The ACPI standard (http://www.acpi.info) defines a set of power states for subsystems to facilitate this. [1], [9] explored power saving strategies on specific scientific workloads like matrix multiplication and LU factorization.

Although many works focus on estimating energy consumption of systems [5], [11], [15], [26], there are a very few works that focus exclusively on estimating the energy consumption of OpenMP applications in compilers.

III. Energy Prediction Mechanism

This Section explains the need for an energy prediction mechanism for recent architectures and it explains the proposed energy prediction architecture for compilers.

A. Need for Energy Prediction Mechanism

On the path to exa-scale computing, a need for developing energy tuning mechanisms has emerged recently. To illustrate, the Department of Energy, US, has planned to reduce the energy consumption of exa-scale systems to 20 MW by 2022 [18]. In order to achieve this goal, several energy reduction mechanisms were proposed at various levels of computing systems by various researchers. For instance, energy reduction techniques that are practiced at the application level are discussed in [20].

Applying energy tuning mechanisms over a preferably large optimization search space, which emerges due to various factors, such as, number of cores used, behavior of applications, performance concern of applications, and so forth, could lead to a hectic overhead and thereby to an unprecedented consumption of energy. Thus, the energy tuning mechanism could effectively be eased if a suitable energy prediction mechanism was accomplished so that executing every possible solution of the optimization search space could be avoided.

A few approaches how models could remarkably assist auto-tuning tools is described below:

1) Finding Optimal Problem Size: Solving an application using an optimal problem size could improve the performance of an application. However, finding an optimal problem size for OpenMP applications might lead to various execution requirements if an exhaustive search mechanism is applied to the process. In most cases, the problem size of an application is reflected in the iterative parts of its code in terms of application parameters. Modeling based solutions could assist user in finding an optimal problem size for an application.

2) Finding Optimal Hardware Resource: An application scales only if the algorithm is capable of scaling well. In fact, most HPC applications are restricted to a limited number of processors or threads. Finding an optimal combination of number of threads or cores for OpenMP applications can become a challenging task when exa-scale applications are considered - ie., exa-scale applications would require thousands to tens of thousands of cores. Training a few sets of combinations of processes and threads for an application could be utilized to predict the other few combinations of processes and threads while finding an optimal hardware resource setup.

3) Finding Optimal CPU Frequency: CPU frequencies are controlled in modern architectures. Each core of a machine could work in different possible CPU frequencies as specified by the user or OS. The code regions of OpenMP applications are either parallel or sequential. If the code regions of OpenMP applications are coarsely granular, CPU frequencies of cores could be elegantly altered so that the energy consumption of the application becomes minimal. Finding an optimal CPU frequency of cores while running applications could be effected using energy models.

4) Finding Optimal Code Variants: In addition to several energy optimal solutions, an HPC application could be tuned by applying various traditional code optimization techniques, such as, loop fusion, loop unrolling, scalar replacement, tail call reduction, and so forth in compilers. As code optimization techniques are massive solutions when applied to the application-as-a-whole, it is advisable to prefer modeling approaches while finding an optimal code variant option in an application.

B. Energy Prediction Architecture for Compilers

The proposed architecture has the capability of predicting the energy consumption of OpenMP code variants using RFM modeling approach in compilers. The processes involved while predicting the energy consumption of OpenMP applications for compilers are expressed in five steps as follows:

1) OpenMP-based parallel applications are given as input to the analyzer entity of the Insieme compiler [19] which does the initial analysis of applications in terms of parallelism and code regions and various other factors.
2) The identified code regions are handed to the optimizer entity of compiler in order to find the energy optimal solution considering performance concerns of the code and the underlying machine characteristics.

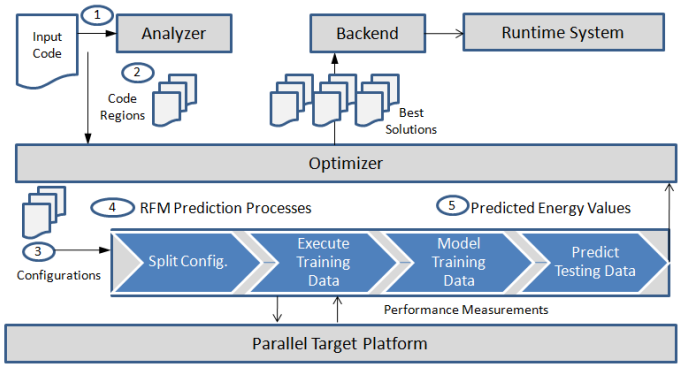

Figure 1. Energy Prediction Architecture for Compilers

3) In order to find the energy optimal solution, the optimizer prepares a list of configurations and submits them to the proposed RFM-based energy prediction mechanism.
4) The role of the RFM prediction processes is to split the list of available configuration settings into the training and testing dataset. The list of configurations in the training dataset, the list of the various problem sizes of OpenMP applications, is executed on the underlying parallel machines. Subsequently, the performance values of code regions due to the corresponding configuration setting are entered as a training dataset. This includes energy consumption values in joules, memory hierarchy issues, execution time, total number of instructions, and so forth. Having the performance values in the training dataset, RFM is applied for constructing RFTrees. Later, the energy consumption values of testing dataset are predicted using RFM.
5) The predicted energy consumption values gotten from RFM, in addition to the energy consumption values of the training dataset, are fed to the optimizer of compilers. Later on, the optimizer would provide the best solutions to the backend and thereby to the runtime system.

IV. RFM Mechanism

As discussed in Section III, prediction based on modeling assists autotuning tools or application developers while predicting an energy optimal solution amongst a larger optimization search space. The prediction could be done either as a regression approach or a classification approach.

This paper discusses the application of RFM based energy prediction mechanism using regression approach for predicting the energy consumption of OpenMP applications during the process of selecting an optimal problem size. This section, hence, discusses the basics of modeling, the RFM algorithm, and the regression-based RFM prediction approach.

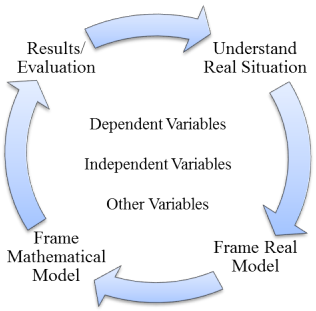

Figure 2. Model Development Processes

A. Modeling Basics

In general, a model is a quantifying system that mimics a real-life situation in terms of mathematical expressions; it could be used for predicting future events or for understanding the current situations.

The three most important variables used in a modeling process are i) dependent variable or response variable or outcome variable, ii) independent variable, and iii) the other variable. A dependent variable could be realized as a variable which is required to get an answer after prediction; an independent variable is a variable that directly influences the dependent variable during the process of prediction; the other variables, e.g., coefficient variables or weights, are the variables which do nothing with real experiments. However, these variables could heavily influence the modeling outcome in terms of prediction results or goodness of fit.

A model development process has four step-wise activities as shown in Figure 2:

1) *Understand Real Situation:* Understanding a contextual situation where modeling should be applied leads to questioning the real working scenario. For instance, what would be an optimal problem size if a matrix multiplication application should be energy efficient is an initial step for understanding the real working scenario.

2) *Framing Real Model:* In this step, a few statements are framed and a set of variables that probably drive the scenario are identified based on the real world contextual situation. This step would remain as a preliminary stage for developing a mathematical model.
3) *Framing Mathematical Model:* Considering the statements that are framed in the previous step of the model development, pseudo-mathematical expressions are designed by quantifying the statements with the identified variables. Suitable techniques, such as, probability, statistics, heuristics, or so forth, could be adopted in this step. Thus, this step, finally, provides a set of mathematical expressions which relate the identified variables and which become solvable.
4) *Results and Evaluation:* In the final step of the model development process, results that are emerging out of the modeling techniques are evaluated with the original question that were a driving force for understanding the real situation in the first step of the model development process. In fact, the results could be used to evaluate the modeling techniques or to evaluate the range of values applied to each variables of the mathematical expressions. In addition, the assumptions, if any, levied while framing mathematical expressions may also be analyzed at this step.

B. RFM Prediction Approach

RFM is a tree based modeling technique that reduces the variance of an estimated prediction function; it embodies an ensemble learning method; it is a modification of a bagging technique; and, it builds a large collection of decorrelated trees during the process of prediction by reducing the correlation between the trees. The success of RFM relies much on its averaging capability.

1) RFM Algorithm: RFM algorithm, in terms of regression analysis, is illustrated in Algorithm 1. As shown, the experimental data having variables and values, the training dataset, are given as an input dataset to the model. Then, the model constructs Random Forest trees (RFTrees).

The processes involved in modeling the training data set could be represented as follows:

1) *Bagging Process:* During this process, bootstrap samples of size N are drawn from the available training set. Later, RFTrees are grown by recursively following the steps, such as, i) selecting m variables from p available variables in the training dataset, ii) electing the best variable which would be used as a split determination point while constructing the trees, iii) splitting the nodes into two sibling nodes, and iv) ensuring the creation of RFTrees.
2) *Ensembling process:* Ensembling process in RFM is a sort of creating sense out of the created bags of RFTrees during the bagging process of RFM.

Algorithm 1 RFM Algorithm

Require: $Training\ Data \leftarrow on\ experiments$
Require: $Bagging\ Process$
$For\ B\ Bags$
1. $Frame\ a\ bootstrap\ sample\ of\ size\ N\ using\ training\ data$
2. $Grow\ RFTrees$
2.1 $Select\ m\ variables\ out\ of\ p\ variables\ for\ each\ data\ set$
2.2 $Elect\ the\ best\ variable\ ie.\ a\ split\ point\ in\ the\ node$
2.3 $Split\ the\ node\ to\ two\ sibbling\ nodes$
Ensure: 2.4 $Ensure\ RFTrees\ are\ created$
Require: $Ensembling\ Process$
$Output\ ensemble\ of\ trees\ RFTrees_1^B$
print $Modeling\ Result$

2) RFM Regression-based Prediction Approach: Once when the model is created based on the training dataset, prediction of a dependent variable, mostly the expected question to be solved for a problem $f(x)$, could be achieved based on a RFM regression mechanism (see Algorithm 2). The prediction mechanism is applied to the testing dataset, which can be either a single testing data or a set of testing data.

The idea behind the RFM prediction mechanism is to average the noisy RFTrees which consequently could reduce the variances of bagging.

Algorithm 2 RFM Regression-based Prediction Mechanism

Require: $Testing\ Data$
Require: $Regression-based\ Prediction\ Process$
$Calculate f(x) = \frac{1}{B}\sum_{bags=1toB} RFTrees_{bags}(x)$
print $Predicted\ Values$

3) Model Validation Parameters: RFM is validated in terms of R^2 and Mean Square Error (MSE). R^2 is a measure of goodness of fit of RFM regression. Theoretically, the R^2 value lies between 0 and 1; it has no units; 1 means the predicted values are perfectly related to independent variables - resulting in good prediction; 0 means the predicted values are not related to independent variables - resulting in poor prediction. MSE is used to measure the average of the squares of errors which incorporates the variance of RFM predictor and its bias.

V. Experimental Results

Experiments were conducted to study the application of RFM mechanism for predicting the energy consumption of the variants of OpenMP applications.

A. Experimental Setup

In this study, predicting the energy consumption of applications, such as, NAS benchmarks, n-body simulations, stencil computations, and a matrix-multiplication application, was revealed when different problem sizes S were considered. It should be noticed that a similar procedure could be applied for predicting the other energy optimal solutions which were discussed in Section III. The performance data, including energy measurements, were remotely measured on the Thomson machine of the University of Innsbruck, Austria, and the RFM-based energy prediction mechanism was remotely tested from the HPCCLoud Research Laboratory [13], India. The target hardware, Thomson, is a shared memory node equipped with four Intel Xeon E5-4650 Sandy Bridge EP processors, each offering 8 cores clocked at 2.7 GHz and featuring core-private L1 and L2 caches of 64 KB and 256 KB, in addition to a CPU-wide shared cache of 20 MB. The system provides 128 GB of main memory and runs a Linux-based operating system with kernel version 3.5.0.

We rely on Intel RAPL to obtain energy consumption data since it offers an internal sampling rate of approximately 1 KHz and a value resolution of 15.3 microjoules. Recent related work has shown RAPL to be accurate enough for our use case [12], and samples are read at least every 30 seconds to capture any register overflows. In addition, we measure time via the `rdtsc` x86 instruction, and employ PAPI to count hardware events that are to be used for modeling.

B. Performance Data and Modeling Dataset

The training and testing dataset, which were required for undergoing RFM modeling and verification process, were obtained by executing the applications on the Thomson machine.

The experimentation was carried out for the different problem sizes S of applications and the performance data were observed for the corresponding experiments. The problem sizes (see Table I) of applications were chosen based on the nature of the applications as given below:

1) In the matrix multiplication application, the problem size, *S=1098*, represents 1098 x 1098 matrix multiplications. In n_body simulations, the number of particles 'Np' parameter was considered for representing the problem size S.
2) In the stencil application, *S=1401* represents unique problem sizes considering two varying parameters of the application, namely, size of matrices and size of stencils (SIZE_N vs. STENCIL_SIZE). The parameters were selected with some specific conditions: i) both the parameters should have odd numbers and ii) the STENCIL_SIZE parameter should have values that are less than half the value of SIZE_N parameter. Thus, experiments were conducted accordingly. For example, we experimented the application with parameter values (7,3), (9,3), (11,3), and so forth while uniquely naming the problem size as S=1, 2, 3, and so forth.
3) In the NAS-BT benchmark, *S=1728* represents problem sizes which are the combinations of L_PROBLEMSIZE x M_PROBLEMSIZE x N_PROBLEMSIZE parameters. These parameter values are fed to the grid points of the benchmark via. the npbparam.h file of BT. We modified the input.bt file of BT to ensure that the problem size varied from 1x1x1 (S=1) to 12x12x12 (S=1728) - each combination having one unique problem size.
4) In the NAS-CG benchmark, *S=6017* was chosen based on the parameter 'NA' of the benchmark. It was noticed that the 'NA' parameter was responsible for varying the problem sizes of the application. However, the benchmark could not be experimented with a lower NA values. Thus, the experiments were conducted with NA=100 to NA=6116. This means that S=1 was assigned for NA=100 and S=6016 for NA=6116.
5) Similarly, each OpenMP application had various influencing parameters which clouted the problem size S - NAS-EP had an M variable; NAS-FT had NX, NY, and NZ variables; NAS-LU had ISIZ1, ISIZ2, and ISIZ3; NAS-MG had NX_DEFAULT, NY_DEFAULT, and NZ_DEFAULT; and NAS-SP had a PROBLEM_SIZE parameter. Thus, experiments were conducted for different problem sizes of applications.

At the end of the experiments, performance values of eight variables, namely, cpu_energy, problem_size, PAPI_TOT_INS, wall_time, cpu_time, PAPI_L1_DCM, PAPI_L3_TCM, and PAPI_FP_OPS were observed for the experiments. Throughout the experiments, cpu_energy was kept as dependent variable and the others were considered as independent variables during the RFM modeling process. The observed experimental results were fragmented into two parts, namely, training dataset and testing dataset.

C. RFM Modeling Results

RFM modeling was applied to the training dataset of observed performance data in order to predict the energy consumption of the testing dataset of OpenMP applications under consideration.

RFM modeling was studied in various directions as follows:

1) Energy prediction of OpenMP Applications' Testing Dataset
2) Influence of Number of RFTrees in RFM
3) Influence of Independent variables in RFM
4) Influence of the proportion of training datasets in RFM

1) Energy Prediction and Results: With 50 percent performance data kept as training dataset and the other kept as testing dataset, RFM was applied to model the energy consumption of the different problem sizes of OpenMP

Table I
CHOSEN PROBLEM SIZES, TRAINING DATASET, ENERGY VARIATIONS, AND THE OBTAINED R^2 VALUES FOR OPENMP APPLICATIONS

OpenMP Applications	Energy Variation (joules)	Problem Size	Training vs. Testing	R^2 in %
Matrix	0 - 111.397	1000	500 vs. 500	99.92
n-Body	0.4517 - 95.925	1000	500 vs. 500	99.99
stencil	0.39 - 47467	1401	700 vs. 701	99.86
NAS-BT	0.0660 - 3.28	1728	864 vs. 864	98.6
NAS-CG	122.4 - 200.5	6017	3008 vs. 3008	95.62
NAS-FT	1.678 - 2.669	1331	665 vs. 666	81.41
NAS-LU	41.71 - 38098	101	50 vs. 51	98.72
NAS-MG	11.18 - 9767.48	893	446 vs. 447	99.94
NAS-SP	573.4 - 137510	70	35 vs. 35	98.93

applications based on the other seven performance values. Later, the testing data were used to predict the energy consumption of the remaining code variants of applications.

RFM modeling for the OpenMP applications were considered using RFTrees=100 and $m = 5$. The chosen problem sizes, numbers of data used in the training and testing datasets, energy variations of the training dataset which should be fed to the RFM modeling, and the obtained R^2 values are shown in Table I. Similarly, Figure 3 and 4 show the observed (represented as dots) and predicted (represented as lines) energy consumption values of the training and testing datasets. From the figures, it could be observed that the energy consumption values of matrix, n_body, NAS-LU, NAS-MG, and NAS-SP applications gradually increased with the increasing problem sizes. In stencil application, the STENCIL_SIZE parameter of the application varied from minimal to SIZE_N/2 values for each SIZE_N parameter value. Thus, the energy consumption value of this application jumped from low to high for each problem size 'S'. In NAS-BT, the energy consumption values increased from 1x1x1 (S=1) combination to 12x12x12 (S=1728) combination of parameters (L_PROBLEMSIZE x M_PROBLEMSIZE x N_PROBLEMSIZE). This showed a stepwise increase in the energy consumption value for each increase in the L_PROBLEMSIZE parameter value.

It could be noticed from the figures (3 and 4) and Table I that the energy consumptions of the testing data were predicted almost accurately with R^2 as 0.998 for most of the applications. However, a few applications, such as, NAS-FT and NAS-SP had lesser R^2 value as the training data had diverse energy consumption values.

2) Influence of Number of RFTrees in RFM: Having the number of RFTrees in the range of 50 to 100, RFM showed an abundant convergence of energy modeling results for applications. However, it was observed that RFM showed no greater improvements when the number of RFtrees were increased. In addition, while selecting a larger number of RFTrees, the computation time increased tremendously (see Table II for Matrix Multiplication application). Thus, the number of RFTrees used in RFM modeling heavily influenced the energy modeling results.

Table II
INFLUENCE OF THE NUMBER OF RFTREES IN MODELING RESULTS

RFTrees (m=2)	MSE	Computation Time
1	7.800	0.3869
5	3.5	0.3992
10	2.47	0.407
15	2.38	0.428
20	1.85	0.4316
25	1.90	0.435
35	1.88	0.456
45	1.69	0.413
55	1.538	0.502
65	1.66	0.52
90	1.715	0.59
100	1.89	0.6
1000	1.705	2.496
10000	1.65	20.13

3) Influence of Independent Variables: Independent variables play a major role during the bagging process of RFM modeling. As discussed in subsection V-C2, the energy consumption value, the value of the dependent variable, was modeled when all other variables were used as independent variables. The details about the dependent and independent variables are discussed in section V-B.

It was interesting enough to understand the influence of independent variables. To do so, the following experiments were conducted:

1) Varying the number of independent variables that was used while splitting the nodes in RFM during the bagging process of RFM, and
2) Removing some specific independent variables

Varying Independent Variables: RFM uses a set of independent variables, m variables out of p variables, while splitting the RFTrees (see section IV-B1). Based on the number of independent variables used while splitting the RFTrees, the RFM had differences in modeling results. As shown in Table III, it could be observed that RFM showed poor results if either a minimal set of independent variables were chosen or a maximal set of independent variables were chosen while splitting RFTrees for a matrix multiplication application. In the experiments, it was noticed that RFM had revealed better results when $m = 5$ and $p = 7$ for the applications except for NAS-Stencil application, where the better results were obtained when $m = 2$ and $p = 7$ (MSE was 12.81 for $m = 2$ and 16.74 for $m = 5$).

Fixing $m = 5$ during the RFTree creation process, the number of trees were increased from 55 to 100, 1000, and 10000. It was observed that MSE of RFM was gradually improving while challenging the computation time. For instances, NAS-CG application showed an additional 142.53 seconds of computation time when moved from

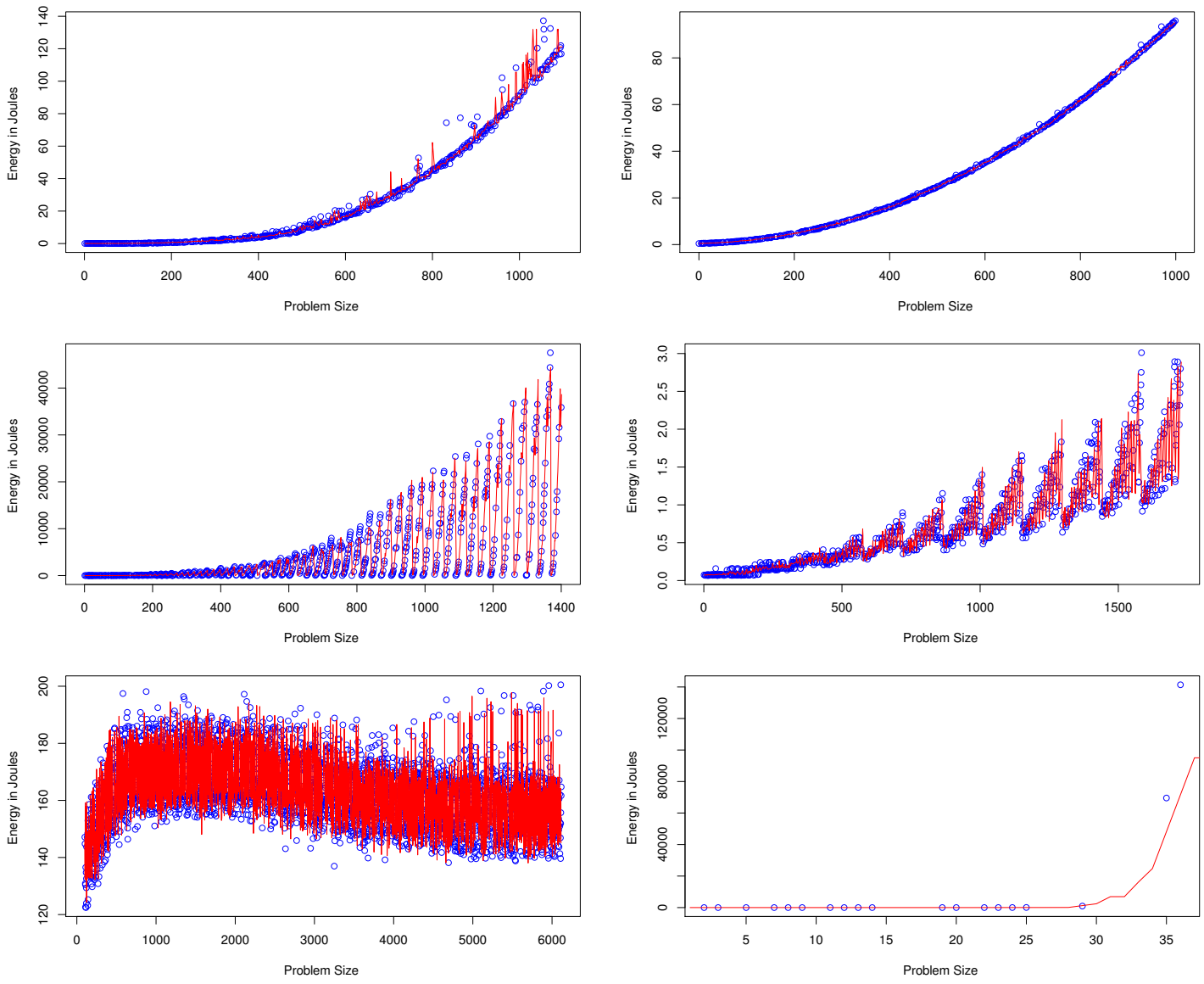

Figure 3. Energy Prediction of OpenMP Applications - Matrix (top left), n_body (top right), stencil, NAS_BT, NAS_CG (bottom left) and NAS_EP (bottom right)

Table III
RFM RESULTS DUE TO NUMBER OF INDEPENDENT VARIABLES TRIED AT EACH SPLIT OF RFTREES FOR MATRIX MULTIPLICATION APPLICATION

Number of m variable (RFTrees=55)	MSE
1	4.21
2	0.98
3	0.77
4	0.789
5	0.663
6	0.79
7	0.78

RFTrees=1000 to RFTrees=10000; NAS-FT had an additional 25.3 seconds of computation time; and the matrix multiplication application had 0.6096, 0.65, and 0.6739 MSE values with the corresponding computation times as 0.718, 3.64, and 31.56 seconds when the RFTrees were 100, 1000, and 10000. Thus, it could be concluded that RFM provided better results when RFTrees=100 and m=5 while considering the computation time and MSE values of RFM.

Removing Independent Variables: Diving further down to understand if any improvements were happened in RFM results while removing the independent variables and its corresponding values, it was noticed that removing certain

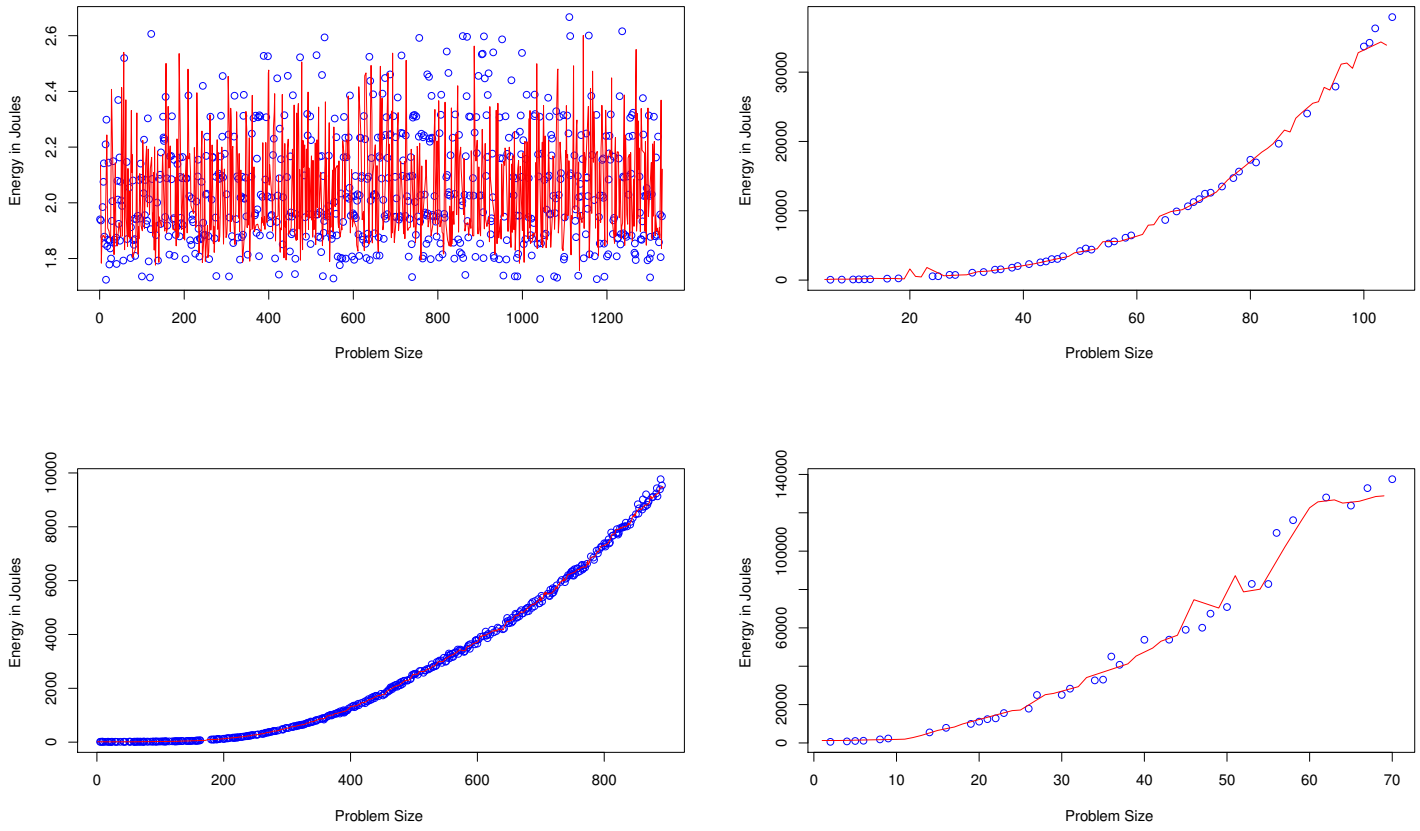

Figure 4. Energy Prediction of OpenMP Applications (contd.)- NAS_FT (top left), NAS_LU (top right), NAS_MG (bottom left), and NAS_SP (bottom right)

independent variables showed reasonably better results - for instance, removing PAPI_TOT_INS and wall_time when $m = 5$ and RFTrees=100 were used in RFM modeling resulted in MSE=0.607 for the Matrix Multiplication application. Table IV shows the most influenced independent variable for applications while pursuing with RFM modeling.

In Figure 5, MSE of RFM and the computation time are represented in the matrix form - the upper diagonal represents MSE and the lower diagonal represents the computation time. The first row of the matrix shows the resultant value when independent variables IV1-IV2, IV1-IV3, and so forth, were removed: IV1 represents problem size, IV2 represents PAPI_TOT_INS, IV3 represents wall_time, IV4 represents cpu_time, IV5 represents PAPI_L1_DCM, IV6 represents PAPI_L3_TCM, and IV7 represents PAPI_FP_OPS.

As m=5 and RFTrees=100 showed better RFM results in our experiments, we fixed the number of independent variables as 5. Thus, the other two variables were removed and the RFM modeling tests were conducted.

MSE / Comp	IV1 - IV2	IV1 - IV3	IV1 - IV4	IV1 - IV5	IV1 - IV6	IV1 - IV7
	0.6766	0.85	0.887	0.772	0.867	0.800
IV2 - IV1	IV2 - IV3	IV2 - IV4	IV2 - IV5	IV2 - IV6	IV2 - IV7	
0.705	0.607	0.7231	0.648	0.58	0.606	
IV3 - IV1	IV3 - IV2	IV3 - IV4	IV3 - IV5	IV3 - IV6	IV3 - IV7	
0.706	0.703	3.315	0.83	0.805	0.842	
IV4 - IV1	IV4 - IV2	IV4 - IV3	IV4 - IV5	IV4 - IV6	IV4 - IV7	
0.704	0.708	0.713	0.839	0.846	0.858	
IV5 - IV1	IV5 - IV2	IV5 - IV3	IV5 - IV4	IV5 - IV6	IV5 - IV7	
0.703	0.703	0.714	0.709	0.698	0.843	
IV6 - IV1	IV6 - IV2	IV6 - IV3	IV6 - IV4	IV6 - IV5	IV6 - IV7	
0.713	0.714	0.708	0.705	0.713	0.706	
IV7 - IV1	IV7 - IV2	IV7 - IV3	IV7 -IV4	IV7 - IV5	IV7 - IV6	
0.707	0.705	0.704	0.703	0.706	0.708	

Time (in sec)

Figure 5. Influence of Removing Independent Variables

4) Influence of the Proportion of Training Dataset: Changing the proportion of training dataset while performing RFM modeling of OpenMP applications varied RFM

Table IV
THE MOST INFLUENCED INDEPENDENT VARIABLE ON APPLICATIONS

OpenMP Applications	wall_time	cpu_time	PAPI_TOT_INS	PAPI_FP_OPS	PAPI_L1_DCM	problem size	PAPI_L3_TCM
Matrix							**
n-Body					**		
stencil					**		
NAS-BT					**		
NAS-CG			**				
NAS-FT			**				
NAS-LU			**				
NAS-MG				**			
NAS-SP	**						

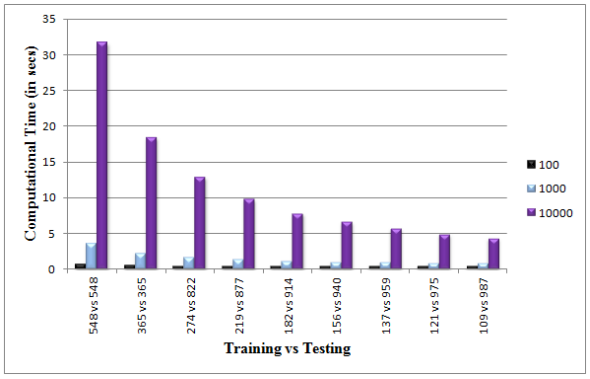

Figure 6. Computation Time for Different Proportion of Training vs. Testing DataSet

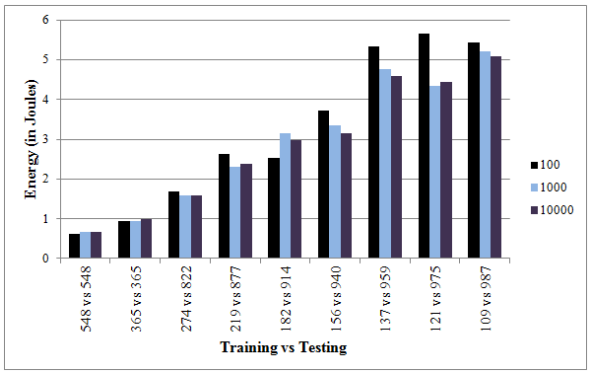

Figure 7. MSE for Different Proportion of Training vs. Testing DataSet

modeling results for OpenMP applications. To illustrate the case, the training and testing datasets of Matrix Multiplication application were partitioned from the total number of performance dataset (1096) into 10 different cases - ie., 548 vs 548, 365 vs 731, and so forth (see Figure 7).

It was observed on experiments that the computation time of RFM modeling was minimal when the number of observations in the training dataset was less. However, it increased vice-versa in addition to having contradictory results in MSE of RFM. For instance, see Figure 7 - the value of MSE increased abruptly in latter cases. In the experiments, it was identified that the MSE value was minimal when 50 percentage of training and 50 percentage of testing data are used (ie., the first case of our experiments produced better results) from the available performance dataset.

In addition, it was observed that increasing the number of RFTrees would become an advantage to MSE calculations, especially when the number of observations of the training dataset is minimal. This could be noticed in the latter test cases (see Figure 7) although there were no remarkably higher reduction in the MSE value.

VI. CONCLUSION

This paper described the energy prediction mechanism of OpenMP applications using Random Forest Modeling approach for compilers. The proposed approach was validated using applications, such as, NAS benchmarks, matrix multiplication, stencil codes, and n_body simulations. In addition, the proposed RFM approach was studied by varying RFTrees, independent variables, and varying proportion of training datasets. From the experiments, it was observed that RFM predicted the applications almost accurately with R^2 as 0.998 for most of the applications.

ACKNOWLEDGMENT

This project was funded by the Indo-Austrian project - DST No:INT/AUA/FWF/P-02/2013 and FWF Austrian Science Fund No:I1523.

REFERENCES

[1] P. Alonso, M. F. Dolz, F. D. Igual, R. Mayo and E. S. Quintana-Orti, "Saving Energy in the LU Factorization with Partial Pivoting on Multi-core Processors," in Int. Conf. on Parallel, Distributed and Network-based Processing, Garching, 2012.

[2] D. A. Bacigalupo, S. A. Jarvis, L. He, D. P. Spooner, D. N. Dillenberger and G. R. Nudd, "An Investigation into the Application of Different Performance Prediction Methods to Distributed Enterprise Applications," JoS, vol. 34, no. 2, 2005.

[3] Barbara Chapman Gabriele Jost, Ruud Van De Pas, Using OpenMP - Portable Shared Memory Parallel Programming (Scientific and Engineering Computation), MIT Press, 2007.

[4] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. de Supinski and M. Schulz, "A regression- based approach to scalability prediction," in 22nd ICS08, 2008.

[5] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer and G. Giuliani, "A Methodology to Predict the Power Consumption of Servers in Data Centres," 2nd International Conference on Energy-Efficient Computing and Networking, pp. 1-10, 2011.

[6] J. Brehm and P. Worley, "Performance Prediction for Complex Parallel Applications," in 11th International Symposium on Parallel Processing, 1997.

[7] J. Cao, S. A. Jarvis, D. P. Spoon, J. D. Turner, D. J. Kerbyson and G. R. Nudd, "Performance Prediction Technology for Agent-Based Resource Management in Grid Environments," in 16th International Parallel and Distributed Processing Symposium, Washington, DC, USA, 2002.

[8] L. Carrington, A. Snavely and N. Wolter, "A performance prediction framework for scientific applications," Future Generation Computer Systems, vol. 22, no. 3, February 2006.

[9] M. Castillo, J. C. Fernandes, R. Mayo, E. S. Quintana-Orti and V. Roca, "Analysis of strategies to save energy for message-passing linear algebra kernels," in 20th International Euromicro Conference on Parallel, Distributed and Network-based Processing, Garching, 2012.

[10] C. Dubach, J. Cavazos, B. Franke, G. Fursin and M. F. P. O'Boyle, "Fast compiler optimisation evaluation using code-feature based performance prediction," in 4th International Conference on Computing Frontiers, 2007.

[11] A. Gandhi, M. Harchol-Balter, R. Das and C. Lefurgy, "Optimal power allocation in server farms," 11th International Joint Conference on Measurement and Modeling of Computer Systems, pp. 157- 168, 2009.

[12] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using RAPL," ACM SIGMETRICS Performance Evaluation Review, vol. 40, no. 3, pp. 13–17, 2012.

[13] HPCCLoud Research Laboratory, http://www.sxcce.edu.in/hpccloud/, accessed in 2014.

[14] K.-H. Kim and C. A. Ellis, "Performance Analytic Models and Analyses for Workflow Architectures," Information Systems Fronteers, vol. 3, no. 3, pp. 339-355, 2001.

[15] J. Kim, M. Ruggiero and D. Atienza, "Free Cooling-aware Dynamic Power Management for Green Datacenters," 2012 International Conference on High Performance Computing and Simulation, pp. 140-146, 2012.

[16] H. Li, D. Groep, J. Templon and L. Wolters, "Predicting Job Start Times on Clusters," in International Symposium on Cluster Comp. and the Grid, 2004.

[17] J. Li, Y. Fan and M. Zhou, "Performance modeling and analysis of workflow," IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 34, no. 2, pp. 229-242, March 2004.

[18] Patrick Thibodeau, Exascale Computing Seen in this Decade, http://www.itworldcanada.com/article/exascale-computing-seen-in-this-decade/45058, accessed in Dec 2014.

[19] Philipp Gschwandtner, Juan J. Durillo, Thomas Fahringer, Multi-Objective Auto-Tuning with Insieme: Optimization and Trade-Off Analysis for Time, Energy and Resource Usage, EuroPar 2014, pp. 87-98, 2014.

[20] Shajulin Benedict, Application of Energy Reduction Techniques using Niched Pareto GA of EnergyAnalzyer for HPC Applications', in 7th IEEE IC3 2014, http://dx.doi.org/10.1109/IC3.2014.6897234, 2014.

[21] W. Smith, I. Foster and V. Taylor, "Predicting application run times with historical information," JPDC, vol. 64, no. 9, pp. 1007-1016, September 2004.

[22] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia and A. Purkayastha, "A framework for performance modeling and prediction," in Supercomputing Conference, 2002.

[23] S. Sodhi, J. Subhlok and Q. Xu, "Performance prediction with skeletons," Cluster Comp., vol. 11, no. 2, pp. 151-165, 2008.

[24] R. Susukita, H. Ando, M. Aoyagi, H. Honda, Y. Inadomi, K. Inoue, S. Ishizuki, Y. Kimura, H. Komatsu, M. Kurokawa, K. J. Murakami, H. Shibamura, S. Yamamura and Y. Yu, "Performance prediction of large-scale parallell system and application using macro-level simulation," in Supercomputing Conference, 2008.

[25] V. Taylor, X. Wu, J. Geisler and R. Stevens, "Using Kernel Couplings to Predict Parallel Application Performance," in 11th IEEE International Symposium on High Performance Distributed Computing, Washington, DC, USA, 2002.

[26] G. Wen, J. Hong, C. Z. Xu, P. Balaji, S. Feng and P. Jiang, "Energy-aware Hierarchical Scheduling of applications in Large Scale Data Centers," International Conference on Cloud and Service Computing, pp. 158-165, 2011.

[27] Y. Wu, L. Liu, J. Mao, G. Yang and W. Zheng, "An analytical model for performance evaluation in a computational grid," in 3rd workshop on High performance computing in China, 2007.

[28] L. T. Yang, X. Ma and F. Mueller, "Cross-Platform Performance Prediction of Parallel Applications Using Partial Execution," in Supercomputing, 2005.

[29] E. J. H. Yero and M. A. A. Henriques, "Contention-sensitive static performance prediction for parallel distributed applications," Performance Evaluation, vol. 63, no. 4, pp. 265-277, May 2006.