Sustainable Computing: Informatics and Systems 19 (2018) 43–51

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

# Prediction assisted runtime based energy tuning mechanism for HPC applications[☆]

Shajulin Benedict

*Indian Institute of Information Technology Kottayam, Kerala, India*

ARTICLE INFO

*Keywords:*
Compiler switches
DiscretePSO
Energy tuning
HPC applications
Tools

ABSTRACT

Performance tuning has become a crucial step for large-scale HPC applications, including HPC based Cloud applications. A need for an energy-aware autotuning solution has recently widened research thrusts among energy conscious scalable application developers. There exist a few standalone energy reduction approaches such as reducing MPI wait times, diligently selecting CPU frequencies, efficiently mapping workloads to CPUs, and so forth for HPC applications. Implementing energy-aware autotuning mechanisms for HPC applications, however, might require multiple executions if exhaustively tested. This paper proposes a prediction assisted energy tuning mechanism named Random Forest Modeling based Compiler Optimization Switch Selection mechanism (RFM-COSS) for HPC applications. RFM-COSS was implemented using RFM algorithm and its variants, namely RFM-SRC and RFM-Ranger. The training datasets of RFM-COSS were created using DiscretePSO algorithm for a few candidate benchmarks such as hpcc, MPI-Matrix, and Jacobi. The experimental results of the proposed RFM-COSS prediction mechanism achieved 17.7 to 88.39 percentage points of energy efficiencies for HPC applications.

## 1. Introduction

HPC application developers, in general, are interested in enhancing the performance of applications at various levels of runs. Given the limited electrical budgeting and increased electrical billing of HPC architectures, in recent years, the decision to design energy-aware autotuning frameworks for HPC applications has occupied the minds of HPC researchers [28], including the supercomputing/HPC community.

Energy tuning of HPC/exascale applications, in general, is a scientific keyword teaming with so many tuning options and objectives that there will never be enough time for users to evaluate and understand the tuning options (even the satisfactory fraction of them) if manually tested.

There exist a few energy tuning mechanisms for HPC applications such as MPI parameter tuning, problem size tuning, CPU frequency tuning, and so forth [7,29,8], including the selection of compiler optimization switches. However, selecting the most energy efficient combination of compiler optimization switches for HPC applications is a challenging task due to the following reasons:

1. the most commonly used compilers such as icc, gcc, opencc, or pgcc, including mpi wrappers (bullxmpi), have numerous compiler optimization switches. This increases the time required for selecting the energy-aware set of compiler optimization switches for applications.
2. utilizing vendor specific compiler switch optimization options namely, -Ox, for applications, is not always a performance-efficient solution. This is due to the fact that the -Ox compiler optimization switches have a fixed set of compiler optimization switches irrespective of applications or architectures.

Precisely, users can hardly find time to manually probe further into the underlying effects of adding/removing each compiler optimization switch for their applications on HPC machines. This arduousness further worsens when a specific code region of an application is concerned. In addition, energy-aware productive compilers are very rare in the HPC market that focuses on the code regions of HPC applications. These reasons lead HPC application developers or autotuning designers to opt for a tool or a method which automatically finds the energy-aware combinations of compiler optimization switches for the code regions of applications.

This paper proposes an RFM-based Compiler Optimization Switch Selection (RFM-COSS) mechanism which automatically tunes the compiler optimization switches for HPC applications. Executing all

☆ This work is partially supported by the Indo-Austrian project – DST No: INT/AUA/FWF/P-02/2013.

*E-mail address:* shajulin@iiitkottayam.ac.in.
*URL:* http://www.iiitkottayam.ac.in/shajulin.php.

https://doi.org/10.1016/j.suscom.2018.06.004
Received 13 December 2016; Received in revised form 27 April 2018; Accepted 20 June 2018
Available online 02 July 2018

combinations of compiler switches can lead to several executions of applications. In order to elegantly handle the tuning process, RFM-COSS works in two phases: Phase I prepares an *EAPerfDBRepo* repository, a mongodb based repository of EnergyAnalyzer tool (EA), which consists of energy-efficient compiler optimization switches. These compiler optimization switches are identified from a few candidate benchmarks/applications (known applications) using a newly developed DiscretePSO energy tuning option of EA (see Section 3). DiscretePSO energy tuning option of RFM-COSS not only collects the energy efficient compiler optimization switches for the candidate benchmarks to *EAPerfDBRepo* repository, but also the corresponding performance and energy consumption values of these benchmarks; Phase II of RFM-COSS utilizes the *EAPerfDBRepo* repository data as training datasets and attempts to predict the energy-aware compiler optimization switches for the code regions of applications. The predictions are carried out using Random Forest Modeling (RFM) and its variants such as Random Forest-Survival, Regression, Classification (RFM-SRC) and RFM-Ranger. The predicted compiler optimization switches from RFM and its variants are evaluated on HPC machines and the best energy-aware combination of compiler switches is selected. In addition, the proposed RFM-COSS approach was tested with a battery of applications.

In succinct, the contributions of this paper are listed as follows:

1. an energy prediction assisted RFM-COSS mechanism, a combined effort of predictions made by RFM and its variants, was proposed for automatically tuning the compiler optimization switches for the code regions of HPC applications.
2. the potential energy-aware optimization switches were collected during the training phase of RFM-COSS using the DiscretePSO tuning process of EnergyAnalyzer.
3. Random Forest Modeling prediction mechanism and its variants such as RFM-SRC and RFM-Ranger were investigated for HPC applications/benchmarks; and, the experimental results of the RFM-COSS approach were compared with the linear regression based prediction approach.

The rest of the paper is organized as follows. The existing autotuning and energy tuning approaches are presented in Section 2. Section 3 explains the proposed RFM based Compiler Optimization Switch Selection (RFM-COSS) mechanism for applications. Section 4 presents the experimentation carried out for hpc applications. Finally, Section 5 presents a few conclusions of the work.

## 2. Related work

Performance engineering research is obviously progressing in the context of autotuning designs for HPC applications. Several researchers are vigorously working toward framing autotuning architectures [29,10,2,15,23,12,24], including cloud environments [19,20].

In addition, HPC researchers, for the past few years, have gotten interested in designing energy efficient software and energy efficient HPC systems. Most of them are more focused on designing energy reduction mechanisms for HPC applications [1,7,25]. For instances, [5] had proposed a notion which endeavored to efficiently utilize the parallel systems. In fact, a parallel application might not require all allocated cores or threads throughout its execution. The authors in [5] have avoided the idleness of cores/threads by switching on/off the resources and clocking the CPU frequencies to low/high power states. Knobloch et al. [17] and Flin et al. [9] have studied the effects of the wait times of real-world MPI applications and their respective energy consumption details.

Subsequently, the performance analysis tool developers and application developers [3,31,18] have oriented their investigations for designing energy-aware autotuning frameworks. Designing autotuning frameworks, in general, is a challenge. This is due to the fact that a sophisticated autotuning framework should scale very well, should have high portability, should be insightful, should automatically decide on the executions (as an astronomer), and should have the capacity to handle the heterogeneous nature of complex architectures.

Selecting the right optimization switches of compilers in order to improve the performance of applications has been a research interest among a wide sector of HPC researchers. With the rising complexity in compilers and architectures, finding the best set of compiler switches in an automatic fashion, however, has become a serious challenge to HPC researchers. Accordingly, a few researchers have studied the selection of compiler optimizations using performance counters [4], statistical approaches [22], reduced exhaustive search mechanisms, iterative elimination approaches [21], and heuristic approaches, including Genetic Algorithms [11] in the recent past years. In addition, a few other researchers have identified the compiler bugs and the faults of threaded applications using compiler optimization options.

For instances, in [4], authors have utilized the performance results of a few benchmarks and then applied the logistic regression modeling approach for predicting the compiler optimization switches of applications. This approach is similar in terms of using hardware counters for constructing the prediction model. However, the authors of [4] had to run benchmarks each and every time for predicting the combinations which could lead to overheads. In [22], authors have statically selected the compiler optimization switches by iteratively executing a few selective applications from the multimedia and cryptography domains on selective machines such as ARM MPCore processor machines. Hence, this work has restricted their research to a few selective application domains. In [11], authors have applied a black box technique of identifying a few compiler optimization options for applications. However, this approach did not either construct models or predict the compiler optimization options of applications.

In this paper, DiscretePSO based energy tuning mechanism is applied for collecting the best possible compiler optimization switches in a repository with respect to a few candidate benchmarks in Phase I of RFM-COSS. In fact, this phase is required only once. The performance inference availed during the tuning process of benchmarks/applications is utilized to predict the compiler optimization switches for the code regions of HPC applications using RFM and its variants in the second phase of the RFM-COSS mechanism.

## 3. RFM-COSS energy tuning approach

There are more than 10s of 100s of compiler optimization switches for the most commonly available compilers such as icc, gcc, or pgcc. For instance, bullxmpi-1.2.8.4, a gcc-5.4.0 compiler wrapper of mpi, has around 198 compiler optimization switches. Predicting an energy-aware combination of compiler optimization switches is considered to be an ideal primordial task for HPC applications. This is due to the fact that the compile time of real applications may be extremely high.

This paper proposes a Random Forest Modeling based Compiler Optimization Switch Selection (RFM-COSS) mechanism which operates in two phases – (i) DiscretePSO based training dataset preparation phase for a few candidate benchmarks and (ii) RFM prediction phase (see Fig. 1). The RFM prediction phase is a combined effort of RFM and its variants namely RFM-SRC and RFM-Ranger. A detailed information of these phases is elaborated in Sections 3.1 and 3.2.

### 3.1. Phase I – DiscretePSO based energy tuning of benchmarks

The notion of Phase I of RFM-COSS is to create a repository of energy-aware compiler optimization switches, the training dataset of RFM-COSS, using the DiscretePSO algorithm [16] based energy tuning process of EnergyAnalyzer tool (EA) (see Fig. 1). RFM-COSS collects the hardware performance counter values of the code regions of benchmarks such as total number of executions, total cache misses, and so forth (including energy consumption values) when benchmarks were