Design-Time Analysis to improve Energy-efficiency of HPC Applications

Extended Abstract

Anamika Chowdhury, Madhura Kumaraswamy, Michael Gerndt∗
Technical University of Munich
Boltzmannstr. 3
Garching 85748, Germany
chowdhua@in.tum.de

ABSTRACT

The European Union Horizon 2020 READEX project is developing a tool suite for dynamic energy tuning of HPC applications. The tool suite performs an analysis during design-time before production run to construct a tuning model encapsulated with the best-found configurations that is then fed to the runtime tuning library. The library switches the configurations at runtime to adapt the application for energy-efficiency.

CCS CONCEPTS

→Computer systems organization → Parallel architectures;
→Hardware → Power and energy; 
→Software and its engineering → Development frameworks and environments;

KEYWORDS

Automatic tuning, High Performance Computing, tools for parallel computing, energy efficiency

DOI: XX.XXX/XXX

1 INTRODUCTION

Energy management has become one of the dominant factors nowadays in High Performance Computing (HPC) systems. Large scale installation of powerful and heterogenous data centers and complex systems have led to enormous electrical power consumption. As a result, from energy-efficient hardware technology, extensive cooling solutions, improvement in applications’ algorithm uptil tuning applications are required to improve the overall energy-efficiency.

The European Union READEX† project develops a tool suite to improve energy efficiency by influencing tuning parameters during application execution at runtime [5]. The READEX methodology is a two-stage approach and consists of Design Time Analysis (DTA) and Runtime Application Tuning (RAT). Both uses the same monitoring system, Score-P [3] as the common instrumentation and measurement infrastructure. During design-time, a tuning model is precomputed by using the Periscope Tuning Framework (PTF) [1, 4]. A novel tuning plugin of PTF, the READEX Tuning Plugin was developed to determine the best configurations or settings for the runtime situations (rts’s), i.e., dynamic instances of significant regions [6]. PTF evaluates configurations during experiments with an instrumented version of the application. This tuning model is forwarded to the READEX Runtime Library (RRL) to tune the application by dynamically switching to the best configurations for upcoming rts’s at runtime. It is also possible to specify domain knowledge to distinguish more application characteristics in order to detect more accurate configuration.

This paper briefly presents DTA as the first step of the READEX methodology. This outlines the precomputation of the tuning model by PTF and the READEX tuning plugin in Section 2. Section 3 briefly presents results obtained for the BT benchmark of the NAS Parallel Benchmarks [2].

2 DESIGN-TIME ANALYSIS

Pre-analysis steps are performed prior to DTA in which the application is instrumented with Score-P and analyzed from performance measurement for tuning potential. There is significant instrumentation overhead caused by fine granular regions. This is reduced by creating a filter file consisting of these regions. Score-P reads this filter file and suppresses all measurements for those fine granular regions. Coarse granular program regions that cover most of the execution time and possess tuning potential are selected for dynamic tuning. For this purpose, we developed another Score-P profile based tool, readex-dyn-detect. The tool detects coarse granular regions and computes tuning potential for those regions by investigating dynamically changing main characteristics.

DTA is performed by PTF, a distributed framework consisting of a frontend and a hierarchy of analysis agents. The agents connect to the application through the Online-Access Interface of Score-P to send/receive measurement request/results and tuning requests from Score-P. The PTF frontend executes the READEX tuning plugin to perform DTA for certain aspects (such as tuning parameter, search strategy and significant regions) and given tuning objectives (Energy, CPU Energy, Execution Time, Energy Delay Product or Energy Delay Product Squared). It applies an online search, executing experiments with different configurations of the tuning parameters for a single program run.

It evaluates tuning objectives of the rts’s of the significant regions giving the best configuration of the tuning parameters and there by generating tuning model used at runtime.
At first, the READEX plugin reads a READEX configuration file provided by the user and initializes the ranges of the tuning parameters. The READEX tuning plugin supports currently three tuning parameters: core frequency, uncore frequency and the number of OpenMP threads. Next, it defines a search space with a search algorithm (exhaustive, individual, random and genetic search) given by the user or loads default search algorithm (exhaustive search). Next, the plugin sets the objective to application tuning. It then reads the significant regions from the configuration file and creates experiments consisting of the values of the tuning parameters, the objective, and the rtss’s for which the objective will be evaluated. The experiments measure the objective value for all the valid rtss’s of the significant regions.

After executing all the experiments, the plugin returns the best configuration for the phase and individual best for the rtss’s. In the end, the plugin creates dynamically three additional experiments by setting the tuning parameters for the phase and the individual rtss’s to their best configuration. The first experiment determines the dynamic energy saving that is achieved as compared to the static configuration setting. The other two experiments verify the reproducibility of the measured objective values.

### 3 EVALUATION

This section presents the results obtained from readex-dyn-detect and DTA. The results are reported for hybrid version of the NAS Parallel Benchmark (NPB) suite.

First, readex-dyn-detect is applied to determine the significant regions. For BT-MZ, these regions are: exch_qbc, x_solve, y_solve and z_solve.

These regions are exported into the READEX configuration file so that the READEX tuning plugin can read it. The plugin performs 16 different experiments using the exhaustive search strategy and returns the best configuration (setting) for the OpenMP threads and the CPU frequency tuning parameters.

Columns two and three in Table 1 present the total energy consumed by the significant regions and the entire phase for the worst and best configurations. The static configuration is applied for the entire phase and the worst configuration gives the worst energy consumption for the phase while the best configuration gives the best node energy. As we can see, the best setting is not necessarily the best setting for all the significant regions. This is where the READEX dynamic tuning can be applied. Columns 4 presents the best result for each significant region obtained with a region specific configuration (column 5 and 6).

The static energy saving amounts to 43.60% for all the significant regions and 24.60% for the phase region as compared to the worst static energy values in column 2. The energy savings obtained using the READEX tuning for the significant regions is 4.40% when compared to the best static energy for these regions in column 3.

The detailed steps of the design-time analysis will be presented as live demo with the poster.

### 4 CONCLUSION

This paper presented the Design-Time Analysis step of the READEX methodology for tuning the energy efficiency of HPC applications. In contrast to previous approaches [5], READEX focuses on runtime tuning guided by a tuning model that is pre-computed during application design-time.

### REFERENCES

**Design-Time Analysis to improve Energy-efficiency of HPC Applications**

Anamika Chowdhury; Madhura Kumaraswamy; Michael Gerndt
Technical University of Munich (TUM)

---

### Abstract
- Tune HPC applications dynamically for improved energy-efficiency and performance.
- Switching between configurations by exploiting dynamic characteristics of HPC applications.
- Develop tool aided auto-tuning methodology.
  - Design-time analysis
  - Runtime Application Tuning
- Detect during design-time, exploit during runtime.

### Design-Time Analysis Workflow

- **Instrumentation**: Uses Periscope Tuning Framework (PTF) to gather data.
- **Dynamism Detection**: Identifies regions with significant variations.
  - **readex_dyn_detect**: Used to detect dynamism.
- **READEX Configuration file**: Stores configurations used during tuning.
- **Design-time Analysis**: Performs analysis on detected regions.
- **TPF & RRL**: Tools used for runtime and post-runtime analysis.
- **Tuning Model**: Generates configurations for best performance.

### Design-Time Analysis (DTA)
- Performed by the Periscope Tuning Framework (PTF)
  - Tunes performance and energy.
  - Evaluates alternatives online.
  - Supports different tuning strategies.
- **The READEX Tuning Plugin**
  - Multiple objectives.
  - Configurable search space via READEX configuration file.
  - Multiple search strategies for searching.
  - Tuning Parameters:
    - core frequency, uncore frequency, # of threads
    - Experiments for selected configurations
    - Energy and time measured for all RTS’s.
    - Identification of static best for phase and specific best configurations for RTS’s.

### Initial DTA Results

**Table 1. Results of DTA for the BT-M2 benchmark using the READEX Tuning Plugin**

<table>
<thead>
<tr>
<th>Significant regions</th>
<th>Energy for worst configuration (1, 1.6)</th>
<th>Energy for best configuration (4, 1.6)</th>
<th>The READEX Energy</th>
<th># of thread</th>
<th>Core Frequency (GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>exch_qbc</td>
<td>3245</td>
<td>6649</td>
<td>2760</td>
<td>1</td>
<td>2.4</td>
</tr>
<tr>
<td>x_solve</td>
<td>74219</td>
<td>41341</td>
<td>39962</td>
<td>4</td>
<td>2.0</td>
</tr>
<tr>
<td>y_solve</td>
<td>73536</td>
<td>39497</td>
<td>39497</td>
<td>4</td>
<td>1.6</td>
</tr>
<tr>
<td>z_solve</td>
<td>76336</td>
<td>40699</td>
<td>40386</td>
<td>4</td>
<td>2.0</td>
</tr>
<tr>
<td>SUM</td>
<td>227393</td>
<td>128186</td>
<td>122605</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Energy for phase</td>
<td>376722</td>
<td>284223</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Discussion
- The static configuration is applied for the entire phase.
- The worst configuration
  - worst energy consumption for the entire phase.
- The best configuration
  - best energy consumption for the entire phase.
- The best setting for the phase
  - not necessarily for all the significant regions.
- Static energy savings w. r. t. worst energy consumption
  - 43.60% for all the significant regions.
- Dynamic energy saving due to switching configuration dynamically w. r. t. best energy consumption.
  - 2.40% for all the significant regions.

### Conclusions
- Presents the Design-Time Analysis step of the READEX methodology for tuning to improve energy efficiency.
- READEX focuses on runtime tuning guided by a tuning model pre-computed during design-time.

### Future Goals
- Inter-phase dynamism.
- Handling multiple input files.
- Domain knowledge specification.
  - Allows the user to provide domain knowledge as identifiers.
  - Application Tuning Parameters.
  - Input identifiers.

---

### Acknowledgements
- Funded by the European Union’s Horizon 2020
- Grant agreement no. 671657

### More Info
- [www.readex.eu](http://www.readex.eu)
- [www.researchgate.net/project/READEX](http://www.researchgate.net/project/READEX)

---

**THE READEX Tool-Suite**

**Design-Time Analysis**
- Detect program regions having variations in characteristics.
- Determine different Runtime Situations (RTS) of the detected regions.
- Determine best configurations for RTSs.
- Classify RTS’s based on similar configurations into scenarios.
- Encapsulate the scenario information into a tuning model.

**Runtime Application Tuning**
- Propagate the generated tuning model for the production run.
- Performed by the READEX Runtime Library
  - Lightweight.
  - Switch to the best configuration for a detected RTS retrieved from the tuning model.
- Calibration mechanism
  - Calibrates regions which were not seen during design-time.

---

**Contact Information**

Robert Schoene  
Technische Universitaet Dresden (TUD)  
Email: Robert.schoene@tu-dresden.de  
Phone: +49 (351) 463-42483

---

**Collaborations**

---

**Acknowledgements**
- Funded by the European Union’s Horizon 2020
- Grant agreement no. 671657

---

**More Info**
- [www.readex.eu](http://www.readex.eu)
- [www.researchgate.net/project/READEX](http://www.researchgate.net/project/READEX)