Faculty of Science, Technology and Medicine

Software-based energy efficiency analysis: benchmarks and assessment methods

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information and Computer Sciences

Author:

Olexandr Stetsenko

Supervisor:

Dr. Alfredo Capozucca

Reviewer:

Prof. Martin Theobald

August 2022

DECLARATION

I hereby declare that the contents of this dissertation are entirely original except where specific references to the work of others are made. They have not been submitted in whole or in part for consideration for any other degree or qualification at this or any other university. Except as specified in the text and Acknowledgements, this dissertation is entirely my work and contains nothing that results from collaborative work with others.

Stetsenko Olexandr

August 2022

ACKNOWLEDGEMENTS

I would first like to thank my supervisor Dr. Alfredo Capozucca for supervising this master's thesis. Under his supervision, I gained many valuable skills which improved my knowledge not only in the thesis subject but as well in general approaches to be used when working on such projects. Whenever I had problems and questions about my thesis subject or writing, his office door was always open for helping me to find solutions and guiding the research in the right direction. His organized way of doing research, helped me achieve huge progress in a short time.

Under his supervision, I learned about the systematic literature review that allowed me to be organized and systematic in my approaches to find relevant benchmarks, tools, hardware, and methodologies that were relevant for conducting the experimental part. During, the experimental part, I got a lot of advice and help which made me understand better the process and the obtained results.

I would also like to thank the reviewing committee: Dr. Alfredo Capozucca and Prof. Martin Theobald for reviewing the thesis, giving me important comments on how to improve further my thesis, and for challenging questions.

It was been a privilege to study Master's in Information and Computer Science at the University of Luxembourg with passionate professors that helped me to learn a lot of valuable skills and improved my knowledge in different areas of Computer Science.

Last but not least, I would like to thank my family for their love and constant support.

ABSTRACT

Huge improvements are done in the last few years in terms of software and hardware performance. Hardware performance increased by creating more powerful components. Particularly, huge improvements are done in processors, RAM, and graphics cards. Software performance also increased because developers are allowed to use more hardware resources, especially in terms of processor performance by using concurrency.

Although continuously increasing the performance of software and hardware is important from a development point of view. But what about energy consumption? Does the energy consumption increase, decrease, or remained stable with the increase in hardware and software performance? Indeed, it is the main topic on which this thesis is focused: understanding what is energy consumption with respect to the increase in performance. This increase in performance can be seen as the improvement in execution time and as the usage of more processor resources by using concurrency.

To achieve this goal, a benchmark for assessing concurrency is used for measuring the energy consumption for answering the questions previously mentioned.

This thesis can be divided into three parts:

1. Firstly, state-of-the-art approaches are observed with the help of a systematic literature review to understand what are existing/alternatives benchmarks which assess concurrency and how energy consumption can be measured.

2. Secondly, based on retrieved data, an experiment is set up for measuring the energy consumption of a predefined benchmark (Vacation2).

3. Lastly, based on the results of the experiment, the behavior of energy consumption is analyzed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem statement

During the past few years, it was possible to observe that hardware and software became more powerful in terms of performance. As consequence, the energy needed for running these hardware and software increased and continue to increase with the new releases.

The big questions are:

1. What in these hardware and software provoke the increase in energy?

2. Does it make sense to continue to increase performance, in terms of speed, at the cost of increasing energy consumption?

The first question is simple to answer. By producing more powerful hardware in terms of processors, memory, graphical cards, and other components, there are more additional resources that software can potentially use during their executions.

As result, by using these additional resources, software performance increased and allowed to run more complex computational tasks. Consequently, energy consumption may also increase.

Indeed, in the majority of cases, the increase in software performance is obtained from the processors because it is the "heart" or "brain" of any device and by using more resources from these processors, computations become faster.

In addition, the programmers will make use of multiple processors by using a technique called concurrency to run their software even faster.

The advantage of concurrency, in today's programming, is that programmers no longer worry about competition over common resources (i.e: shared variables) because concurrency handles these cases while running processes in parallel. So, concurrency manages how processes are running in parallel and how they communicate with each other. (More on what is concurrency in chapter 2)

For the second question, it is not so evident because there is nothing bad in the desire to increase the performance, but is it conformed to the desire of the stakeholders?

The problem is that when performance increases then energy consumption will also increase and stakeholders will have to pay more. Thus, software which may have a good performance may be less valuable because fewer stakeholders will use them.

Thus, programmers have to be careful when designing software and not only focus on performance but also on energy consumption.

In this thesis, energy consumption is observed by looking through the glass of concurrency to answer the questions previously mentioned.

One way how energy consumption can be observed is shown by analyzing the energy performance of an advanced concurrency model, which is evaluated through a state-of-the-art benchmark created for such kinds of models.

1.2 Motivation

Multicore processors are state-of-the-art in today's computers and make concurrency available for software developers to be able to run several tasks simultaneously which allows to improve the execution time.

The main motivation of this thesis is to find out the relationship between performance and energy consumption where performance can be seen as the execution

time of a software and energy consumption is the energy consumed or used while running a software.

The notion of energy is defined as $Energy = Power * Time$. This $Energy$ can vary a lot because it depends on two variables: $Power$ and $Time$. The more the $Power$ is used by the processor(s), the higher will be the $Energy$ when considering $Time$ as constant. When considering, $Power$ as constant, the same holds also for $Time$ which can be seen as the execution time while running a piece of software.

So both, $Power$ and $Time$ have a huge impact on energy consumption ($Energy$). Both variables can be used and manipulated for understanding the variation of energy.

In conclusion, the usage of the hardware resources is what determines the energy consumption.

1.3 Objectives

Let's discuss the objectives of this thesis which will help in the understanding of the relationship between energy consumption and performance(time) with the help of concurrency.

There are in total three objectives:

1. Obj1: Explore how the energy consumption is assessed by benchmarks that also assess concurrency.

2. Obj2: Assess energy consumption of Chocola by using the Vacation2 benchmark.

   By [1], Chocola is an advanced concurrency model implemented in Clojure. It provides futures, transactions, and actors. It is unique in ensuring that these three models work correctly even when they are combined.

   Vacation2 benchmark is an adapted version of the Vacation benchmark of the STAMP benchmark whereas the original Vacation[2] benchmark implements an online transaction processing system but serves the task of emulating a travel reservation system instead of a wholesale company.

In other words, this objective consists to assess the energy consumption of Chocola through the Vacation2 benchmark.

3. Obj3: See how concurrency affects energy consumption and how energy consumption varies with respect to performance.

1.4 Research Questions

Main research questions (RQs)

1. What are the existing benchmarks that have been used to access performance of concurrent models and its associated energy consumption? (related to Obj1)

2. Is there any correlation between time performance (speed or duration) and energy performance (energy consumption)? (related to Obj2 and Obj3)

   (a) What is the impact of concurrency on energy performance?

   (b) How does energy performance increase over time? Is it linear, logarithmic or exponential?

3. Is the industry making efforts in enhancing energy performance? (related to Obj2 and Obj3)

   (a) What is the energy performance of modern computers vs. old computers?

1.5 Research Methodology

1. Systematic Literature Review (SLR)

   (a) Firstly, identify a set of papers that can be potentially relevant for our study by using querying and advanced search tools, available in digital libraries such as ACM and IEEE.

   (b) Secondly, two types of criteria (inclusion and exclusion criteria) are defined. Criteria are used for a more advanced selection of the papers. The goal is to reduce a set of potential papers obtained from the query to a set of relevant papers.

   (c) Third, a more detailed analysis is performed on the papers from the final set. The goal is to retrieve relevant information useful for conducting our experiment and answering some research question(s). Moreover, the obtained information has to be summarized properly for giving a global overview of our SLR results.

2. Experiment

   (a) An experiment is designed based on the information retrieved from the previous step. This information allows us to select the tool(s) and approach(es) needed for conducting the experiment using the Vacation2 benchmark.

   (b) The measurements obtained from the experiment are summarized for obtaining a better overview in the form of graphs and tables.

3. Analysis of the results

   Analyze the results of the Vacation2 benchmark for understanding how concurrency affects energy consumption. This analysis is done using the graphs and tables produced previously.

1.6 Contributions

The contributions made in this thesis can be separated into four major points:

1. First is the Systematic Literature Review which is done by finding the benchmarks which assess concurrency as well as the possible profiling tools needed for measuring the energy consumption. Based on the research results, the state-of-the-art approach is identified for measuring the energy consumption of a Vacation2 benchmark.

2. Secondly, an experiment is done based on a Vacation2 benchmark by using the information and approach(es) observed from the SLR. This experiment shows how concurrency affects energy consumption.

3. Thirdly, a tool support is created to run the experiment and to measure the variables of interest such as energy, power and execution time.

4. From the first and second contributions, a catalog of benchmarks is created. This catalog is useful to access energy consumption. The catalog is composed of benchmarks from SLR and an approach on how the benchmarks can be potentially run is shown with the Vacation2 benchmark. This catalog will show developers where their code is standing in terms of energy consumption by comparing it with benchmarks from the catalog.

Chapter 2

Background

2.1 Benchmark

Benchmarks are one of the most important items in this thesis because the work is closely related to energy consumption and benchmarks, on which this energy consumption is measured.

So, what is a benchmark?

According to a research paper [3], benchmarks are used to compare different platforms, methods, tools, or techniques. They define standardized measurements to provide repeatable, objective, and comparable results. In the computer science domain, benchmarks are used for comparison, for instance, the performance of databases, algorithms, and cloud services.

Based on research done during this thesis, there are two types of what is called a benchmark.

1. A benchmark can be seen as a piece of software that can be run for obtaining a certain result and which results can be used for comparison.

2. A benchmark can also be seen as a collection or a set of many benchmarks. So, a set of benchmarks from point 1. (just above).

2.2 Energy

Energy [4] doesn't only concern applications and software for mobile platforms but as well to other platforms for which these applications/software are developed.

The conclusion reached by the industries is that even small inefficiencies in apps/software over the system, significantly affect battery life, performance responsiveness, and temperature and may give some discomfort to the users not only in terms of performance but in terms of energy costs.

One example of a problem related to costs is the data centers that run the software. The problem is that if software/apps are bad in terms of energy consumption, the cost of running these software becomes higher because energy consumption is huge.

The reason for all energy problems is that unfortunately, during the last few years, little attention has been given to energy consumption. This results that there are not many tools and processes which were designed for improving the understanding of the usage of energy by the software developers.

Just previously, we saw that software has an impact on energy consumption, but the question is how?

Indeed, this relation between software and energy can be expressed with one single formula:

$$E = P \text{ x } t$$

From this formula, E is the energy consumption which is computed by multiplication of P and t, where P is the power and t is the time. Energy is expressed in joules (J), power in watts (W), and time usually in seconds (s). As a small remark, time is the execution time of the software.

When considering software energy consumption, there are 4 aspects to which some attention should be paid:

1. A given software system under execution

2. On a given hardware platform

3. On a given context

4. On time duration

For (1) is the software that must be executed, and the type of software may give some thoughts about potential energy consumption.

For (2) depending on the hardware the consumption of the energy may vary. In other words, by performing a task on one hardware and then performing the same task on another hardware may give different energy consumption.

For (3), context plays a big role, since the way software is built and used, has a critical influence on energy consumption. For instance, the software can stress energy consumption on CPUs or RAM, or any other parts. Thus depending on what software uses, the consummation of energy may be different.

And the time (4), also plays a huge role. There are some discussions by the developers that by reducing the execution time, energy may also decrease. Unfortunately, it may be not always the case because by reducing the execution time, CPUs may work more, and this will lead to an increase in power (P). As consequence, an inverse effect can be obtained.

Energy is an interesting topic and based on problems encountered with energy consumption, the optimization of the software became one of the important aspects in our days.

2.3 Profiling tools

A profiling tool [5] is a tool that allows monitoring and measuring the energy consumption of a component by using the computer's hardware.

Moreover, these profiling tools are software tools that can be installed on a computer for reading the information directly from the computer's hardware. Indeed, it is a big advantage because if developers want to measure energy consumption, they must install only software and run it based on documentation.

As consequence, they don't need to buy any physical device which can be connected from outside to a computer for measuring energy consumption. In addition, these outside devices (also called Power Monitors) are difficult to set up.

Thus, profiling tools have some advantages over physical devices. Profiling tools are simple to use because once installed, they can be easily synchronized with the software execution. The synchronization can be the following: When the software starts his execution, at the same time, a profiling tool starts his measurement as well. When software ends his execution, a profiling tool will also end his measurement.

As a result of the measurement, values will be shown in the terminal or saved in a file for further observations.

Even though profiling tools are easy to use and precise in their results, there are some small disadvantages. Unfortunately, it is not possible to completely isolate the software under test from the entire system, which allows running the operating system, because both the system and the software will be operational at the same time. As consequence, values measured for software will be impacted by the system (operating system) because both are running on the same hardware and profiling tools perform their measurement on the entire hardware.

The only way to reduce this problem is to make sure that no unnecessary applications or processes are running on the device. In addition, if it is possible, the battery-powered device may be removed because it may affect some measurements in some setups.

It is also important to choose the right profiling tool because some tools may be available for some operating systems and hardware, and some not. For instance,

the "perf" profiling tool can be used only on Linux and cannot be used on other platforms.

So, as it is possible to see, even though profiling tools have some disadvantages, they can be considered good options for performing the measurements, if correctly used.

2.4 Concurrency and Parallelism

Multicore processors have become one of the essential parts of running software in today's world. By using multiple processors, the software's performance may improve, resulting in software which are running faster.

This performance improvement is based on concurrency and parallelism. Usually, both terms come together when speaking about multicore processors because they are related but different.

According to [6], the difference between Concurrency and Parallelism is that a system is said to be concurrent if it can support two or more actions in progress at the same time and a system is said to be parallel if it can support two or more actions executing simultaneously. The main difference between these definitions is the phrase "in progress".

For instance, an application that is considered concurrent may have two or more threads in progress, running at a certain time. This would mean that two or more threads are running on a single processor, and they are swapped between each other. Just to be clear, threads on a single processor are not running simultaneously but are switched between each other.

Thus, the role of concurrency is to manage the execution of threads in progress.

For parallel execution, multiple cores are available and for each core, two or more threads can be assigned. Moreover, cores would be running simultaneously. These also result, that some threads would be running also simultaneously.

It is possible to deduce that parallelism is a subset of concurrency because it is possible to have an application that uses multiple threads but if there are no multiple cores then parallel execution cannot be achieved.

The goal of using concurrency and parallelism is to achieve a better performance in terms of execution time. For instance, when running a program by using two processors, the upper bound improvement that can be achieved is half of the time, with respect to a single processor. If using four processors, the program may run at the best 4 times faster.

Unfortunately, these are ideal cases but in reality, the increase in performance is lower. In addition, increasing the number of cores may result in a decrease in relative performance. In other words, the improvement in performance becomes less significant with the increase of the number of cores.

Chapter 3

Systematic Literature Review

In this chapter, a systematic literature review is done in order to collect the needed information from relevant research papers which may be helpful to answer some research questions and to prepare the future experiment.

In order to collect this information, four steps have been performed:

1. Decide about where to look for research papers
2. Definition of a search query
3. Inclusion and Exclusion criteria
4. Data extraction from the relevant papers

By using the SLR, our goal is to obtain papers that are valuable and in line with our research.

Results and all steps of SLR can be found here: https://github.com/olexstet/Master_Thesis_SO/tree/main/SLR

Remark: Some external links contain files that have to be downloaded for seeing the results.

3.1 Decision on where to look for research papers

As the first step of the SLR, a decision has to be done about where to look for research papers. Our goal is to find quality papers that are related to our subject. Our subject is mainly based on energy consumption, benchmarks, profiling tools, hardware, and software.

So, it can be seen clearly from these terms that this subject is in the domain of Computer Science. Thus, the places to look for papers must be related to the Computer Science field.

By considering these factors, two digital libraries are chosen:

- ACM digital library [7]: Is a database composed of journal articles, magazine articles and newsletter articles which focus in the domain of Electrical Engineering, Computer Science and Engineering.
- IEEE Xplore digital library [8]: Is a research database for discovery and access to journal articles, conference proceedings, technical standards, and related materials on computer science, electrical engineering and electronics, and allied fields.

3.2 Search Query Definition

3.2.1 What is a query, why it is needed and how to construct?

Each digital library has a tool to do an advanced search. The goal of this tool is to find more easily the relevant papers with the relevant information in these papers.

This tool allows users to define a search query and then based on the query, a set of papers is found. A query can be seen as (key, value) pairs where the key is the section or property of the paper and value is the word (also called term later)

that has to be found in the key. In addition, these (key, value) pairs are related by logical connectives such as &&, ||, etc...

Unfortunately, each digital library has its syntax and each query has to be adapted for getting equivalent semantically queries. In addition, the size of the query is limited. Thus, the information to include in the query has to be chosen carefully.

3.2.2 Construction of the query

The construction of the query is not an evident task. The reason is that if a query will be too complicated and overloaded then only few papers may be given as the result. If the query doesn't contain sufficiently information, it may give a lot of papers.

Another problem is on which terms to construct the query? This question is simple to answer because the terms must be based on our research questions as well as on our thesis subject.

The research question that can be answered by the SLR is only research question 1: What are the existing benchmarks that have been used to access performance of concurrent models and its associated energy consumption?

Remark: Other research questions are answered later by an experiment!

Based on the research question 1 as well as on our thesis subject, a query can be constructed based on the following requirements:

- R1: The research question is primarily based on energy. So, the papers to search have to be strongly related to energy. So, the term "energy" has to be in the title of each paper.
- R2: The term "energy" may come not alone because a lot of papers will be obtained as output but some other terms must be included in addition, such as performance, consumption, and efficiency. These terms are useful for indicating that energy has to be measured in some ways as well as to know how it is measured.

- R3: The problem right now is that results obtained from R1 and R2 will give papers related to not only software engineering but also to other domains such as physics and engineering.

  To reduce this problem, terms related to software engineering such as "software", "program", "programming", "programming languages", and "framework" should be included in the query and these terms may appear not only in the title but at any part of the paper.

  The term "software engineering" is not included because it is a large field and we want to stay close to the programming level.

- R4: The goal of a systematic literature review is to find benchmarks. The question to ask, is where the term benchmark has to be present in the paper? The problem is, if it will be present only in the title and in the abstract then the number of papers will be reduced a lot. So, the term "benchmark" should be present in more sections and as a solution, this term can be present in any part of the paper. In addition, the term "benchmark" can be also mentioned in the text as "benchmarking". So if any of the two is present in the paper then it can be potentially selected.

  Remark: By including the term "benchmark" it does not guarantee that term "benchmarking" will be found because the query in ACM and IEEE works by exact match (what was been observed by manipulation of the query where 4 papers were lost when not considering the term "benchmarking" and only "benchmark").

3.2.3 Final queries

Based on these four requirements previously mentioned, a query can be constructed for ACM and IEEE.

For ACM:
Title:"Energy" AND
(Title: "consumption" OR Title: "performance" OR Title: "efficiency") AND

(AllField:("software") OR
AllField:("program") OR
AllField:("programming") OR
AllField:("programming languages") OR AllField:("framework")) AND
(AllField:("benchmark") OR AllField:("benchmarking"))

For IEEE:
("Document Title":"Energy" AND
("Document Title": "consumption" OR "Document Title": "performance" OR "Document Title": "efficiency") AND
(("All Metadata":"software") OR
("All Metadata":"programs") OR
("All Metadata":"programming") OR
("All Metadata":"programming languages") OR
("All Metadata":"framework")) AND
("All Metadata":"benchmark" OR "All Metadata":"benchmarking"))

3.2.4 Search Refinements

In addition to the query above, some refinements have to be done manually to reduce further the number of papers received and make them more valuable.

- RF1: The papers selected must be not too old because many of the benchmarks and tools may be not supported. In addition, energy consumption in computer science is relatively a new topic and the tendency shows that a lot of papers have been written in the last 5 and 6 years. So, papers are filtered by dates as 01/01/2016-1/4/2022.
- RF2: Our goal is to look at research articles related to benchmarks and energy. In different library platforms (ACM and IEEE) the results obtained are not only from research articles but also from magazines and journals. Thus, we have only to consider research articles.

- RF3: To ensure the quality of the papers, all papers must be peer-reviewed. A peer-reviewed paper is a paper which was been reviewed and evaluated by a third person and which passes a certain level of quality.

  Fortunately, all papers which are in our two digital libraries (ACM and IEEE) are all reviewed before being published. Thus, all papers have good quality.

3.2.5 Query results

As a final result of the query and the refinements for both digital libraries, a set of papers were obtained. The number of papers obtained for ACM is 132 and for IEEE is 61.

List of papers obtained from the queries can be found here: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/References-Queries%20papers.pdf

As said before, each paper has as a principal topic the energy consumption which may be related on the measurement of the benchmarks. In addition, all obtained papers are in the domain of computer science, more precisely in the domain of programming and software engineering.

Indeed, by looking at the total number of retrieved papers, 193 papers are obtained. This seems to be a lot of papers and a question may arise: Why not introduce more refinements or add some additional terms in the query. The answer is that this final result of papers obtained from the properties above is only the beginning of the SLR and these obtained papers will be analyzed further based on inclusion and exclusion criteria.

Moreover, another reason to keep such big number of papers is to reduce at maximum the loss of potentially relevant information which may be caused by the query and refinements. In other words, a big number of papers is kept initially and then based on our needs, each paper is analyzed to determine whether it has to be accepted or removed.

3.3 Inclusion and Exclusion criteria

Inclusion and exclusion criteria can be seen as further filters which allow us to decide which papers may be relevant or not. The final goal of using the inclusion and exclusion criteria is to identify a set of papers that will be used for detailed analysis.

To obtain the final set of papers, the process is divided into multiple steps where for each step some inclusion and/or exclusion criteria are used. By using this approach the relevant information can be more easily organized and will allow a more fluent understanding of how the final set of papers is obtained.

Inclusion criteria (criteria based on content in the papers):

- I_1: Discusses software-related to energy consumption.
- I_2: Presents a strategy to measure the energy consumption/efficiency of software.
- I_3: The strategy makes use of a benchmark to assess the energy consumption/efficiency.
- I_4: The tools to perform the measurements are software-based.
- I_5: Papers related to concurrency.

Inclusion criteria (criteria based on online sources):

- I_6: Benchmark is open source.
- I_7: Tool is open source.
- I_8: Tool is a profiling tool.

Exclusion criteria:

- E_1: No results present in the paper
- E_2: The results are presented as another included paper.
- E_3: Discusses energy consumption/efficiency of an entire system (e.g. datacenter, cluster, or any other system that is combination of software and hardware, but it is impossible to see the actual impact of software related to energy)

3.3.1 Criteria Explanation

3.3.1.1 Inclusion criteria (criteria based on content in the papers):

- I_1**:** Discusses software-related to energy consumption.

  For this criterion, only the abstract and introduction sections are the source of information. In these sections, there must exist a clear indication that a software (benchmark, program, or application) is used for analysis of energy consumption. In addition, it should be the main focus of the paper.

  If the terms such as "benchmark", "application", "software", and "program" are used in these sections and the research of the paper is based on software then the paper is accepted for this criterion.

- I_2**:** Presents a strategy to measure the energy consumption/efficiency of software.

  For this criterion, in the paper, there must exist a software (usually a benchmark) and a tool (not necessarily a profiling tool) that is used for measuring energy consumption. So, there should exist a section called "Methodology" or a clear indication that a certain tool is used on a benchmark/software for measuring energy consumption.

- I_3**:** The strategy makes use of a benchmark to assess the energy consumption/efficiency.

  In the paper, it should be clearly said which benchmark is used. In case of uncertainties or if the name of the benchmark is not stated in the paper,

then the criterion is not satisfied.

- I_4**:** The tools to perform the measurements are software-based.

  For satisfying this criterion, the name of the tool must be mentioned. In addition, it should be software-based and not a physical device. In case of doubts, if there is no certainty that a tool is software-based, it is kept for the next steps where a more deeply analysis is performed, and in case of a problem, the tool will be later removed.

- I_5**:** Papers related to concurrency.

  The goal of this criterion is to find papers related to the concurrency topic. Finding that a paper is related as well to the "Concurrency" topic would mean that the benchmark(s) mentioned in the paper will be potentially used for assessing concurrency.

3.3.1.1.1 Detailed explanation of approach used for I_3 and I_4:

For the 3rd and 4th criteria, to find the benchmarks and the tools from the papers, research by keywords is done for each paper.

The keywords allow identifying the section in which, with some probability, the relevant information can be found. So, when there is exact match with a word from the paper and the keyword, the text around this keyword is observed for finding the relevant information.

The keywords are the following:

For criterion I_3:

- Benchmark

For criterion I_4:

- RAPL (Remark: as this tool is usually used, with high "luck" it may be present)

- Monitor
- Tool
- Profile/Profiler
- Consumption
- Energy
- Measure

There are some rules on how to use these keywords and when to stop searching for the tools.

First, the keyword "RAPL" is checked. If not found, the next fourth keywords ("Monitor", "Tool", "Profile" and "Profiler") are checked, all of them. But usually only one of them is present.

In case none of the keywords are found, the paper must be analyzed more in detail with more general terms such as "consumption", "energy" and "measure", exactly in that order. The reason is that tools may appear more frequently with the keyword "consumption" than with the keyword "measure".

Additionally, in each paper, the additional information such as operating systems is searched. This additional information is also done based on research by keywords.

For Operating system:

- Linux
- Ubuntu
- Windows
- Android

- Mobile
- Operating system

Here, the first four keywords are checked for exact match. If any of the keywords match exactly, the context still needs to be understood for determining if it is the right operating system used in the paper's experiment.

In case none of the four is match, the keyword "Mobile" is checked. If, the keyword "Mobile" is also not found then the keyword "Operating system" is checked for exact match.

3.3.1.1.2 Detailed explanation of approach used for I_5:

For identifying if a paper considers the concurrency, a search by keywords is done as well, on each paper.

The keywords to look at, are the most useful keywords which are used for indicating that a topic is about "Concurrency".

- Multi
- Processing
- Parallel
- Concurrent
- Concurrency
- Thread

A paper is considered as a "concurrency" paper if at least 5 times of any of these keywords appear in the paper. If any of these keywords appear in the section of references, it doesn't count as an appearance.

In case of doubts, if the paper even contains these keywords but it is still not clear. The content around the keywords is observed. If still doubts exist, further reading is performed and a decision then is taken.

3.3.1.2 Inclusion criteria (criteria based on online sources):

- I_6: Benchmark is open source.

  This criterion consists to look at the benchmark retrieved from the papers if they are open sources or not. The search is done through online sources. If a benchmark can be downloaded then it is considered as open-source, otherwise not.

- I_7: Tool is open source.

  This criterion has as goal to see if the tools retrieved from the papers is open source, by performing a search through online sources. If a tool can be downloaded then it is considered as open-source.

- I_8: Tool is a profiling tool.

  Look if the tool is indeed a profiling tool with the help of online sources. This criterion is needed for removing the doubts. A tool is considered as a profiling tool, if there is a possibility to install it on a PC or if it is mentioned that a tool is not a physical device.

3.3.1.3 Exclusion criteria:

- E_1**:** No results present in the paper.

  There must be a clear indication of results in terms of energy or power in any form such as graphs, tables, and values. In case results are missing or are not relevant, then the paper is discarded.

- E_2**:** The results are presented as another included paper.

  The paper can be considered as a summary of another paper or the results in the paper reference another paper.

- E_3**:** Discusses energy consumption/efficiency of an entire system (e.g. datacenter, cluster, or any other system that is combination of software and hardware, but it is impossible to see the actual impact of software related to energy)

In case of the presence of terms such as cluster, HPC, nodes, and data center, the paper is analyzed for finding if it is possible to obtain results for one single piece of hardware (i.e one node). In case, if the answer is no then the paper is rejected.

3.3.2 Step 1

In this step, the goal is to remove completely irrelevant papers by using some inclusion and exclusion criteria.

The process is the following:

1. In the first part, the four inclusion criteria I_1, I_2, I_3, and I_4 are used where at least one of these criteria must hold. In case if one of these criteria holds, the paper is accepted and kept for analysis based on exclusion criteria. In case if the paper is rejected, the check of exclusion criteria is omitted.

2. As second part, all exclusion criteria E_1, E_2 and E_3 are used. If one of the exclusion criteria holds then the paper is rejected. If there are no criterion which is satisfied then the paper is accepted for step 2.

3.3.3 Results of Step 1

By applying the process previously discussed on each paper, the most irrelevant papers are removed. The results for each paper are summarized in an "inclusion-exclusion" table. In total there are two tables, one for ACM and one for IEEE libraries.

Links to the full tables:

- Table based on ACM papers analysis: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/SLR_Papers_General_Analysis/SLR-ACM_Inclusion_Exclusion.csv

- Table based on IEEE papers analysis: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/SLR_Papers_General_Analysis/SLR-IEEE_Inclusion_Exclusion.csv

As final results for two digital libraries, 193 papers were analyzed where 132 papers for ACM and 61 for IEEE library.

For ACM, 86 papers out of 132 were been rejected and only 46 were been accepted.

For IEEE, 35 out of 61 were been rejected and only 25 were accepted.

One interesting fact can be seen by comparing the results from two digital libraries where for ACM 65% of papers were rejected and for IEEE only 57%. Indeed by looking at these rejections values, they are close to each other.

3.3.3.1 Benchmarks and tools obtained from step 1

Based on the criteria discussed previously, some additional information can be found in the paper. Remember our goal is to find some benchmarks and tools which may indicate how to measure the energy consumption from an experimental point of view.

Thus, as it is possible to see from inclusion criteria I_3 and I_4, these criteria look exactly for the presence of benchmarks and tools. Thus, it is possible to retrieve this information from the papers.

Additionally, this information is needed to be retrieved for applying inclusion criteria I_6, I_7 and I_8 later.

During the checking of inclusion and exclusion criteria, benchmarks and tools were also retrieved for each paper and summarised in the tables. In total there are 4 tables where 2 tables for benchmarks and 2 tables for tools. The reason is that 2 tables (Benchmarks and Tools) are for the ACM library and 2 tables (as well Benchmarks and Tools) are for the IEEE library.

Remark: The benchmarks mentioned in these tables are only from the **accepted** papers from the inclusion and exclusion table. Thus, papers that are rejected are not considered for benchmark tables. For tools tables, all tools are considered. Link to the full benchmark tables (for step1).

Benchmarks tables:

Links to benchmarks tables:

- Benchmarks retrieved from ACM papers: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Step1/Benchmarks_ACM_Step1.csv
- Benchmarks retrieved from IEEE papers: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Step1/Benchmarks_IEEE_Step1.csv

This table is composed of 6 columns where 5 columns are based on SLR and 1 column on Internet source.

- Benchmarks **(SLR)**: This column contains the unique name of the benchmark. Thus the name of each benchmark is retrieved only once from the paper(s). In other words, if the name of the same benchmark is present multiple times in different papers, it is considered only once.
- Number citations **(SLR)**: Contains the number of citations of how much a benchmark is mentioned in different papers.
- Links **(Internet source)**: Is used as a source of information to find some descriptions about the benchmark.
- Comments **(SLR)**: This is a column that contains personal remarks about the benchmark that appeared important from the papers where the benchmark was found.
- Tools **(SLR)**: Clearly indicates which tools are used for measuring energy consumption of the benchmark. It may happen that the same benchmark appears in different papers but different tools are used. Thus, tools and the paper are written i.e: (1) RAPL, jRAPL where (1) is for the id of the paper and RAPL and jRAPL are the tools.

- Papers_id **(SLR)** (Present from step2): Indicates the papers in which the benchmark was been found.

For ACM, the number of unique benchmarks obtained is 46 and for IEEE, this number is equal to 22.

Tools tables:

Links to tools tables:

- Tools from ACM papers: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Tools/Tools-ACM.csv
- Tools from IEEE papers: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Tools/Tools-IEEE.csv

The Tools table is composed of 6 columns such as:

- Tools **(SLR)**: Contains the name of the tool which appears in the papers. If a tool is present in several papers, it is only once mentioned in the column.
- N° **(SLR)**: Contains the number of citations of how much a tool is mentioned in different papers.
- Platform **(SLR)**: Contains the name(s) of the platform(s) on which a tool was been run during the experiment in the papers. Examples of platforms: Ubuntu, Linux, Windows, Mac, Mobile,...
- Open source **(Internet source)**: Basically, a "Yes/No" column where "Yes" is if the tool is open source and "No" if the tool is not open source. The decision is made by checking if the tool can be downloaded on external websites or not.
- Maintainability **(Internet source)**: A column which contains binary answer "Yes/No" where Yes is for maintainable tool and "No" if it is not. A maintainable tool is a tool that was been modified at least in 2018. If the tool was been modified in 2017, then it is considered no more maintainable.

- Links **(Internet source)**: Contain the website where the tool can be potentially downloaded or any information about the tool.

For ACM, the number of unique tools obtained from all papers is equal to 44 and for IEEE, this number is equal to 21.

For accepted papers in step 1, the number of unique tools for ACM is 27 and for IEEE is 18. These results are obtain from Tools column from benchmarks tables.

3.3.4 Step 2

In this step, the criterion I_5 is only evaluated on each paper. This means that the papers are analyzed for this criterion and a decision is taken but papers are not rejected (this is done in further steps).

By considering again the "inclusion-exclusion" table, the new I_5 criterion is added.

The goal is to find papers which are relating to "Concurrency" which would mean that benchmarks mentioned in these "Concurrency" papers are potentially used for assessing concurrency.

3.3.5 Results of step 2

As a final result, some 15 papers are considered as "Concurrency" papers out of 25 accepted papers for IEEE and some 18 "Concurrency" papers for ACM out of 46 accepted papers from step 1.

Remark: Results for concurrency analysis for papers can be seen in the same tables where inclusion and exclusion criteria are checked:

- Table based on ACM papers analysis: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/SLR_Papers_General_Analysis/SLR-ACM_Inclusion_Exclusion.csv

- Table based on IEEE papers analysis: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/SLR_Papers_General_Analysis/SLR-IEEE_Inclusion_Exclusion.csv

By considering the new information obtained in this step about "concurrency" papers, the benchmarks tables have to be also modified.

A new column "Concurrency?" was been added to the benchmark table where a "Yes" is written followed by id(s) of papers if a benchmark appears in "Concurrency" papers. Otherwise, a "No" is written for a benchmark, if the benchmark doesn't appear in any of the "Concurrency" papers.

Results of concurrency analysis for benchmarks:

- Based on benchmarks from ACM: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Step2%20and%203/Benchmarks_ACM_Step2%20and%203.csv
- Based on benchmarks from IEEE: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Step2%20and%203/Benchmarks_IEEE_Step2%20and%203.csv

3.3.6 Step 3

In this step, the benchmarks and tools are evaluated based on online sources criteria. For benchmarks, the goal is to identify which benchmarks are open sources.

For tools, there are two goals. The first is is to identify which are open source. The second goal is to clarify which tools are profiling tools.

This step allows us to identify tools and benchmarks which satisfy these criteria: I_6, I_7, and I_8.

3.3.7 Results of step 3

Remember from step 1 that number of unique benchmarks for ACM was 46 and for IEEE was 22. By looking now at the open-source criterion I_6, there are 36

open-source benchmarks for ACM and 14 open-source benchmarks from IEEE.

As it is possible to see, almost 2/3 of benchmarks are open sources.

As a final result for criterion I_7 based on tools which have to be open-source, there are in total 11 tools for ACM which are open-source, and 8 profiling tools for IEEE. As a small remark, the tools considered are only from accepted papers from step 1.

For criterion I_8, the number of tools that are considered as profiling tools for ACM is also 11 and for IEEE is 8. Indeed the tools which are open-source are as well the tools that are profiling tools.

Results of open source analysis for benchmarks:

- Based on benchmarks from ACM: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Step2%20and%203/Benchmarks_ACM_Step2%20and%203.csv
- Based on benchmarks from IEEE: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Step2%20and%203/Benchmarks_IEEE_Step2%20and%203.csv

3.3.8 Final step

This is the final step where the final set of papers is obtained. In this step, the results from steps 2 and 3 are used together on the papers from step 1. It can be seen as an intersection of step 1, step 2, and step 3.

What was been done in this final step, is the following:

1. Select papers where paper is a "concurrency" paper. (Results of step 2)
2. Select papers where benchmarks are open source. (Results of step 3)
3. Select papers where tools are open source. (Results of step 3)
4. Select papers where there is at least one tool for a benchmark. (I_3 and I_4)

Remark: Results of interactions for step2 and step3 can be found here: https://github.com/olexstet/Master_Thesis_SO/tree/main/SLR/Papers_Criteria_Satisfaction/Intermidiate%20step%20(Intersection%20results%20step%202%20and%203)

3.3.9 Results of final step

As a final result, new benchmark tables are created where all the steps discussed previously are considered.

Remark: Benchmarks tables are more representative and give a better overview of results.

As the final results of these benchmarks tables:

Benchmarks	Links	Comments	Tools	Papers
NAS Parallel Benchmark	https://github.com/cwang9208/NPB		(1) RAPL (54) Application Power Management library	1,54
PARSEC	Can be found: https://github.com/bamos/parsec-benchmark Also here: https://github.com/cirosantilli/parsec-benchmark		(7) RAPL	7
Linpack and HPL 2.1 benchmarks	http://www.netlib.org/benchmark/hpl/		(10) RAPL	10
Java's Thread-Safe Collections	No link (simply java collection)	The benchmarks used in this study consist of 16 commonly used collections (13 thread-safe, 3 non-thread-safe) available in the Java programming language such as List, Maps and Sets operations	(19) jRAPL	19
OpenMP	https://www.openmp.org/resources/openmp-benchmarks/ https://github.com/LangdalP/EPCC-OpenMP-micro-benchmarks		(54) Application Power Management library	54
Rodinia	http://www.cs.virginia.edu/rodinia/doku.php		(54) Application Power Management library (55) PAPI	54,55

Table 3.1: Final set of papers and related benchmarks for IEEE

Benchmarks	Links	Comments	Tools	Papers_id
Rodinia	http://www.cs.virginia.edu/rodinia/doku.php		(3) MeterPU, x-MeterPU (18) lm_sensors	3,18
Java Collections	https://github.com/greensoftwarelab/Collections-Energy-Benchmark		(6) RAPL, jRAPL	6
DACAPO	One Open source on github: https://github.com/dacapobench/dacapobench Same open source on gitlab: https://gitlab.anu.edu.au/dacapo/anu-dev/dacapobench/-/tree/ac7322d853b85a1555bf8d8677bb560d508d3eac	Taken from paper: OpenBenchmarking.org	(19) RAPL, jRAPL (42) Intel's Performance Counter Monitor	19,42
Renaissance benchmark sui	https://github.com/renaissance-benchmarks/renaissance	Taken from paper: OpenBenchmarking.org	(19) RAPL, jRAPL	19
MemBench	https://github.com/nicktehrany/membench		(42) Intel's Performance Counter Monitor	42
SciMark	https://openbenchmarking.org/test/pts/scimark2		(42) Intel's Performance Counter Monitor	42
COSMIC	https://eexu.home.ece.ust.hk/COSMIC.html		(87) DSENT	87

Table 3.2: Final set of papers and related benchmarks for ACM

Both tables can be found here:

- Benchmarks table for IEEE: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Final_step/Benchmarks_IEEE_Final_Step.csv
- Benchmarks table for ACM: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Papers_Criteria_Satisfaction/Final_step/Benchmarks_ACM_Final_Step.csv

For IEEE:

- Number of unique benchmarks: 6
- Number of final papers: 6
- Number of unique tools: 4

For ACM:

- Number of unique benchmarks: 7
- Number of final papers: 6

- Number of unique tools: 5

As the conclusion to the final set of papers selection, from 193 papers that were initially, only 12 papers are considered highly relevant to our research. These papers will be used for more detailed analysis (described in the next section of data extraction) for understanding experimental details.

Final set of papers

Paper_id	Reference	Title
IEEE: 1	[9]	How Programming Languages and Paradigms Affect Performance and Energy in Multithreaded Applications
IEEE: 7	[10]	Towards New Metrics for Appraising Performance and Energy Efficiency of Parallel Scientific Programs
IEEE: 10	[11]	Performance and Energy Consumption Analysis of Coprocessors using Different Programming Models
IEEE: 19	[12]	A Comprehensive Study on the Energy Efficiency of Java's Thread-Safe Collections
IEEE: 54	[13]	The Impact of Turbo Frequency on the Energy, Performance, and Aging of Parallel Applications
IEEE: 55	[14]	Automatic Energy-Efficiency Monitoring of OpenMP Workloads

ACM: 3	[15]	Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption
ACM: 6	[16]	The Influence of the Java Collection Framework on Overall Energy Consumption
ACM: 18	[17]	PETRAS: Performance, Energy and Thermal Aware Resource Allocation and Scheduling for Heterogeneous Systems
ACM: 19	[18]	Evaluating the Impact of Java Virtual Machines on Energy Consumption
ACM: 42	[19]	Cross-Layer Memory Management to Improve DRAM Energy Efficiency
ACM: 87	[20]	Approximate Communication Strategies for Energy-Efficient and High Performance NoC: Opportunities and Challenges

3.3.10 General Overview of steps

Steps	Criteria & Results
Step1	$(I_1 \vee \mathrm{I}_2 \vee \mathrm{I}_3 \vee \mathrm{I}_4) \wedge (\mathrm{E}_1 \wedge \mathrm{E}_2 \wedge \mathrm{E}_3) = Res_1$
Step 2	$Res_1 \wedge \mathrm{I}_5 = Res_2$
Step 3	$Res_1 \wedge \mathrm{I}_6 \wedge \mathrm{I}_7 \wedge \mathrm{I}_8 = Res_3$
Final step	$Res_2 \wedge \mathrm{Res}_3 \wedge \mathrm{I}_3 \wedge \mathrm{I}_4 = Res_{final}$

3.4 Data Extraction

Now the data extraction has to be done on the final set of primary studies. The data extraction consists to look into details in each paper to retrieve the information that is relevant for understanding the experimental setups.

Before diving into the detailing analysis, some questions have to be defined to understand the information to retrieve.

The questions are the following:

- Q1: What are the alternatives to vacation2: i.e. what is/are the case studies used when running the reported experiments?
- Q2: What are the settings of the experiment (e.g. hardware, software, free variables and constant variables)?
- Q3: What is/are the variable(s) authors are observing apart of energy consumption, if any?
- Q4: What are the characteristics of the experiments: number of times the same experiment is repeated? Does it rely on the mean or the median?
- Q5: Does it exists a replication package for the reported experiments?

Each paper has to be read in details to find answers to the questions above. The answers to all these questions are written as the first step to a text file. Next, the retrieved data has to be generalized in the tables to get a summary and a global view.

3.4.1 Data Extraction Results

As final results of data extraction, some summary tables are created.

Tables can be found here: https://github.com/olexstet/Master_Thesis_SO/blob/main/SLR/Data%20extraction%20final%20papers/Summary_Tables.csv

3.4.1.1 Data extraction questions satisfaction

IEEE

Paper_id	Q1	Q2	Q3	Q4	Q5
1	++	++	++	++	--
7	++	++	++	--	--
10	+-	++	++	--	--
19	+-	++	++	++	--
54	++	++	++	++	--
55	++	+-	++	--	--

Table 3.3: Questions satisfaction for IEEE

ACM

Paper_id	Q1	Q2	Q3	Q4	Q5
3	++	+-	++	--	--
6	+-	++	++	++	++
18	++	+-	++	++	--
19	++	+-	++	++	++
42	+-	+-	+-	++	--
87	++	--	+-	++	--

Table 3.4: Questions satisfaction for ACM

Both tables (Table 3.3 and 3.4) give a global overview of how previously cited questions are satisfied. A fully satisfied question is marked by a ++ which means that the data extracted is what is needed, +- means that the obtained data answers partially the question and not completely, - - means that nothing is found.

Satisfaction of questions for IEEE:

- Q1: (4,++), (2,+-)
- Q2: (5,++), (1,+-)

- Q3: (6,++)
- Q4: (3,++), (3,- -)
- Q5: (6,- -)

From the results of the IEEE table, questions 1,2, and 3 are more less complete and allow us to answer our questions. As a small remark, questions 1,2, and 3 are important questions because they give an overview of the experimental parts of the paper.

For question 4, half of the papers answer this question which relies on experiment.

For question 5 as it can be seen, there are no papers that answer this question. This means that there are no papers from IEEE that contain a replication package.

Satisfaction of questions for ACM:

- Q1: (4,++), (2,+-)
- Q2: (1,++), (4,+-), (1, - -)
- Q3: (4,++), (2,+-)
- Q4: (5,++), (1,- -)
- Q5: (2,++), (4,- -)

From the results of the ACM table, it is possible to see that all papers plus-minus answer question 1. However for question 2, only 1 paper completely satisfies this question, 4 papers partially answer the question and 1 paper has nothing.

For questions 3 and 4, almost all papers answer the questions.

For question 5, only 2 papers contain the replication package for their experiments which is still a good result because in the majority of cases they are not available.

3.4.1.2 Benchmarks

Benchmarks	Usage & Refrences
NAS Parallel Benchmark	- Programming languages (IEEE: 1) - Assess configurations of parallel applications on a turbo-compliant processor (IEEE: 54)
PARSEC	- Metrics evaluation (IEEE: 7)
Linpack	- Evaluating and comparing Intel programming models based on scenarios (IEEE: 10)
Java's Thread-Safe Collections	- Java collections evaluation (IEEE: 19) - Assessing Java Collection Framework (JFC) implementations (ACM: 6)
Rodinia	- Assess configurations of parallel applications on a turbo-compliant processor (IEEE: 54) - OpenMP workloads on multicore systems (IEEE: 55) - Programming frameworks (OpenMP, openACC, OpenCL and CUDA) (ACM: 3) - Resource Allocator and Scheduler mechanism (ACM: 18)
Dacapo	- Impact of Java Virtual Machines (ACM: 19) - Framework evaluation (ACM: 42)
Renaissance benchmark suite	- Impact of Java Virtual Machines (ACM: 19)
MemBench	- Framework evaluation (ACM: 42)
SciMark	- Framework evaluation (ACM: 42)
COSMIC	- Network-on-Chips (NoCs), chips evaluation (ACM: 87)

Table 3.5: Benchmarks and their usage

Based on the results from the table above (Table 3.5), there are in total 10 benchmarks that can be considered as an alternative to Vacation2 (related to Q1). All these benchmarks are obtained from the final set of papers. Potentially all these benchmarks should be used for assessing concurrency because papers are based on concurrency subtopic. In addition, it is possible to see the usage of these benchmarks.

3.4.1.3 Tools and Benchmarks

Tools	Reference	Benchmarks
RAPL	IEEE: 1	NAS Parallel Benchmarks
RAPL	IEEE: 7	PARSEC
RAPL	IEEE: 10	Linpack and HPL 2.1 benchmarks
jRAPL	IEEE: 19	Java's Thread-Safe Collections
Application Power Management library	IEEE: 54	NAS Parallel Benchmarks
Application Power Management library	IEEE: 54	Rodinia
Intel Performance Counter Monitor (PCM) API	IEEE: 55	Rodinia
x-MeterPU	ACM: 3	Rodinia
RAPL, jRAPL	ACM: 6	Java Collections
lm_sensors application	ACM: 18	Rodinia
RAPL	ACM: 19	Dacapo
RAPL	ACM: 19	Renaissance
Intel's Performance Counter Monitor (PCM) tool	ACM: 42	Dacapo
Intel's Performance Counter Monitor (PCM) tool	ACM: 42	SciMark
Intel's Performance Counter Monitor (PCM) tool	ACM: 42	MemBench
DSENT	ACM: 87	COSMIC

Table 3.6: Usage of profiling tools on benchmarks

This table (Table 3.6) contains all profiling tools used for measuring each benchmark from all selected papers (table related to Q2).

As an observation, the most used profiling tool for measuring the benchmarks is "RAPL" and it is present in 6 papers (7 if considering jRAPL as RAPL. Remark: jRAPL is also RAPL because it is only a variation).

3.4.1.4 Hardware and software usage

Benchmarks	Paper_id	OS	CPU	GPU	RAM Memory	Threads
NAS Parallel Benchmark	IEEE: 1	Ubuntu 16.04 kernel 4.4.0-21	Intel Core i7 (4cores)	No mention	32 GB	1,2,3,4,8
NAS Parallel Benchmark	IEEE: 54	Ubuntu kernel v4.15	AMD Ryzen 7 1700 (Zen)	No mention	No mention	1,2,4,6,8,10,12, 14,16
PARSEC	IEEE: 7	Linux	Intel Core i7-4770, Intel Core i7-6700	No mention	16 GB	1,2,4,8
Linpack	IEEE: 10	Linux kernel 3.10	2 x E5-2699v3 2x18 cores	No mention	128 GB	8,12,16,24,32,6 4,72
Java's Thread-Safe Collections	IEEE: 19	Linux Debian	System1: AMD Opteron 6378 processor, 2.4 GHz System 2: Intel Xeon E5-2670 processor 2.60 GHz	No mention	Both System 64 GB	1,2,4,8,16,32,6 4,128, 256
Java's Thread-Safe Collections	ACM: 6	Linux 3.13.0-74-generic OS	Intel Core i3-3240, 4.40 GHz	No mention	8 GB	No mention
Rodinia	IEEE: 54	Ubuntu kernel v4.15	AMD Ryzen 7 1700 (Zen)	No mention	No mention	1,2,4,6,8,10,12, 14,16
Rodinia	IEEE: 55	Linux	Intel Xeon server with two Xeon E5-2630 v4 microprocessors	No mention	No mention	Fixed to 10 per chunk
Rodinia	ACM: 3	Linux	Intel Xeon E5-2695 v2, Intel Xeon E5-2650 v4, Intel Xeon Phi, 24 cores in total	GTX Titan X	No mention	No mention
Rodinia	ACM: 18	Ubuntu 10.04 (X64)	Intel Core i7-920, 3.06 GHz, 4 cores	NVIDIA Tesla C2070	No mention	2,4,8,16,32,64, 128
Dacapo	ACM: 19	Linux Debian 9 (4.9.0 kernel version), Cluster	Intel Xeon Gold 6130	No mention	192 GB	No mention
Dacapo	ACM: 42	Linux	Intel E5-2520v3	No mention	4x16 GB	No mention
Renaissance benchmark suite	ACM: 19	Linux Debian 9 (4.9.0 kernel version), Cluster	Intel Xeon Gold 6130	No mention	192 GB	No mention
MemBench	ACM: 42	Linux	Intel E5-2520v3	No mention	4x16 GB	No mention
SciMark	ACM: 42	Linux	Intel E5-2520v3	No mention	4x16 GB	No mention
COSMIC	ACM: 87	Gem5 platform (simulation platform)	No mention	No mention	No mention	No mention

Table 3.7: Experimental hardware and software used in papers

Table 3.7 gives an overview of the hardware and software used for running the case studies in papers (table related to Q2). The properties observed are OS (Operating System), CPU, GPU, RAM Memory, and Threads.

As a final result, it is possible to see for the operating system that Linux and Ubuntu are used the most often.

For CPU, Intel Xeon and Intel i7 are the most used which seems to be predictable because a lot of PCs are based on Intel CPUs.

In terms of GPU, there is not much to say because only two GPUs was been used and both are from Invidia.

The most appeared RAM memory for a single pc is 16 RAM.

For threads, it is difficult to see the commune parts because the number of threads may vary for each case study because each case study has different hardware and different objectives.

3.4.1.5 Benchmarks and observed result properties

Benchmarks	Energy	Time	#Threads	EDP	Programming languages / Versions	Power	Frequency	Speedup	Increase	Performace (GFlops)
NAS Parallel Benchmark	X	X	X	X	X					
PARSEC	X		X			X	X	X	X	
Linpack	X	X	X							X
Java's Thread-Safe Collections	X	X	X			X				
Rodinia	X	X	X	X		X				
Dacapo	X	X	X		X	X				
Renaissance benchmark suite	X	X	X		X	X				
MemBench	X	X								
SciMark	X	X								
COSMIC	X									

Table 3.8: Results properties observed for benchmarks (part1)

Benchmarks	Workload size	Performance per Watt (GFlops/W)	Config	Temperature	Aging	#Cores	Libraries (i.e Cuda)	Perf, GB/s, DRAM W, OA (GB), #Sites, S/M, #Hot, #Types, Samps/s	Latency
NAS Parallel Benchmark			X	X	X				
PARSEC									
Linpack	X	X							
Java's Thread-Safe Collections									
Rodinia			X	X	X	X	X		
Dacapo								X	
Renaissance benchmark suite									
MemBench								X	
SciMark								X	
COSMIC									X

Table 3.9: Results properties observed for benchmarks (part2)

This table (Table 3.8 and 3.9 is a single table) gives a general view of all results properties observed in the papers for each benchmark (table related to Q3). So, if a property is observed for a benchmark, then this property is added to the table, if not yet present, and marked with a cross (X).

For our case, the most important properties are "Energy", "Time" and "#Threads" because these properties will allow us to answer questions about energy by using concurrency, and where concurrency is based in terms of time and threads performance.

By looking on these three properties:

- Energy: All benchmarks contain the energy evaluation in at least one paper.
- Time: Only 8 over 10 benchmarks observed this property where time can be seen as execution time.

- #Threads: 7 over 10 benchmarks satisfy this property where the number of threads are observed in the results.

3.4.1.6 Benchmark and the results properties by papers

Benchmarks	Properties
NAS Parallel Benchmark	(1-IEEE) Time - #Threads (over PL) Energy - #Threads EDP - #Threads (54-IEEE) Rel. Perf (Time) - #Threads - Config Rel. Energy - #Threads - Config Rel. EDP - #Threads - Config Rel. Temperature - #Threads - Config Rel. Aging - #Threads - Config
PARSEC	(7-IEEE) Energy - #Threads - frequency Power- frequency Frequency - Power consumption threads Speedup - #Threads Power speedup - Benchmarks App - p(#cores) Increase - frequency (over benchmarks)
Linpack	(10-IEEE) Performance (GFlops) - Workload size Energy (joules/s) - Workload size Energy (joules/s) - Threads (included CPU) Execution Time - Workload size Perfromance per Watt - #Threads
Java's Thread-Safe Collections	(19-IEEE) Energy - Benchmarks - Power Energy - #Threads Time - Benchmarks (6-ACM) Energy(J) - Time (ms)

Table 3.10: Results properties observed for each paper and for each benchmark (part1)

Benchmarks	Properties
Rodinia	(54-IEEE) Rel. Perf (Time) - #Threads - Config Rel. Energy - #Threads - Config Rel. EDP - #Threads - Config Rel. Temperature - #Threads - Config Rel. Aging - #Threads - Config (55-IEEE) CpS - time CpJ - time Watt - time CpJ - #Cores CpJ - Configuration (3-ACM) Time - Benchmarks - Libraries (i.e Cuda) Energy - Benchmarks - Libraries (i.e Cuda) (18-ACM) Time(ms) - Benchmarks - #Cores Energy(J) - Bencharks - #Cores Peak Power (W) - Benchmarks - #Cores CPU Peak Temperature - Benchmarks - #Cores Time (ms) - Benchmarks - #Threads Energy (J) - Benchmarks - #Threads Peak Power (W) - Benchmarks - #Threads CPU Peak Temparature - Benchmarks - #Threads
Dacapo	(19-ACM) Energy Consumption ratio (%) - Java Version Energy Consumption ratio (%) - Benchmarks Average Power (W) - Time (s) #Threads - Time (s) (42-ACM) Perf, GB/s, DRAM W, OA (GB), #Sites, S/M, #Hot, #Types, Samps/s Execution Time - Benchmarks Energy - Benchmarks

Table 3.11: Results properties observed for benchmarks (part2)

Benchmarks	Properties
Renaissance benchmark suite	(19-ACM) Energy Consumption ratio (%) - Java Version Energy Consumption ratio (%) - Benchmarks Average Power (W) - Time (s) #Threads - Time (s)
MemBench	(42-ACM) Perf, GB/s, DRAM W, OA (GB), #Sites, S/M, #Hot, #Types, Samps/s Execution Time - Benchmarks Energy - Benchmarks
SciMark	(42-ACM) Perf, GB/s, DRAM W, OA (GB), #Sites, S/M, #Hot, #Types, Samps/s Execution Time - Benchmarks Energy - Benchmarks
COSMIC	(87-ACM) Latency (ns) - Benchmarks Energy (J) - Benchmarks

Table 3.12: Results properties observed for benchmarks (part3)

This table (Table 3.10-12 is a single table) gives a more detailed overview than the previous table and allows us to see in which paper each property is evaluated

for a benchmark.

As a small observation, the three properties (”Energy”, ”Time” and ”#Threads” are the most appeared ones. Thus, it is possible to see here that our SLR and selection of a final set of papers was been done correctly.

3.4.1.7 Computational properties of experiments

Benchmarks	Paper_id	Experiment repetition	Average/Median
NAS Parallel Benchmark	IEEE: 1	10	Average
NAS Parallel Benchmark	IEEE: 54	10	Average
PARSEC	IEEE: 7	Not mentioned	Not mentioned
Linpack	IEEE: 10	Not mentioned	Not mentioned
Java's Thread-Safe Collections (Java Collections)	IEEE: 19	10	Average(last 3 runs)
Java Collections	ACM: 6	10	Average
Rodinia	IEEE: 54	10	Average
Rodinia	IEEE: 55	Not mentioned	Not mentioned
Rodinia	ACM: 3	Not mentioned	Not mentioned
Rodinia	ACM: 18	1000	Average
Dacapo	ACM: 19	20	Not mentioned
Dacapo	ACM: 42	5	Average
Renaissance benchmark suite	ACM: 19	20	Not mentioned
MemBench	ACM: 42	5	Average
SciMark	ACM: 42	5	Average
COSMIC	ACM: 87	Not mentioned	Not mentioned

Table 3.13: Experiments properties for benchmarks execution

Table 3.13 shows what are the properties for running the experiments in each paper for each benchmark (table related to Q4). As observation a lot of case studies compute the results obtained from experiments repetition, based on the average and never on the medium. But by thinking about this observation, it seems to be more a design choice due to the fact that medium and average may give potentially close results.

What concerns the number of repetitions, the value of 10 is the most appeared one in different papers. Moreover, usually, this number is a small number as 5, 10, or 20 which means that there is no need to do a lot of repetition.

3.4.2 Complementary analysis

In this section, some additional information is observed for complementing the analysis for data extraction. Some of this information is based on the papers

and some are based on online sources. The goal is to obtain a better overview of benchmarks and results from papers.

3.4.2.1 Benchmarks and their properties

Benchmarks	Citations	Primary Languages (Java, Python,C)	Maintenability (min: 2017)	Documentation found	Single PC run?	Concurrency?
NAS Parallel Benchmark	2	Java, C++, Fortran I Yes	Yes	Yes	Yes	Yes
PARSEC	1	C I Yes	No (2009)	Yes	Yes	Yes
Linpack	1	Fortran, C, Java I Yes	Yes (2018)	Yes	Yes	Yes
Java's Thread-Safe Collections	2	Java | Yes	Yes	No	Yes	Yes
Rodinia	4	C I Yes	Yes (2018)	Yes	Yes	Yes
Dacapo	2	Java I Yes	Yes (2018)	Yes	Yes	Yes
Renaissance benchmark suite	1	Scala I No	Yes	Yes	Yes	Yes
MemBench	1	C I Yes	Yes	No	Yes	No
SciMark	1	Java I Yes	Yes	Yes	Yes	No
COSMIC	1	C, C++, Fortran I Yes	Yes	Yes	Yes	Yes

Table 3.14: Properties of benchmarks

Table 3.14 represents the benchmarks and the properties of benchmarks. In all papers, from IEEE and ACM, in total 10 benchmarks were retrieved where the "Rodinia" benchmark is the most cited one because it appears 4 times. There are also some other benchmarks such as "NAS Parallel benchmark", "Dacapo" and "Java's Thread-Safe Collections" which also appeared 2 times in papers.

This table allows to see some general information retrieved from online sources about the benchmarks such as primary languages, maintainability, documentation, and single PC runs and Concurrency. All these properties show how well is a benchmark.

From the column of primary languages, our goal is to see if the benchmarks retrieved use the popular programming languages such as Java, Python, or/and C. As it is possible to see, a lot of benchmarks are written in one or multiple popular programming languages which means that some benchmarks are rewritten to another programming languages from the original one.

For the column of maintainability, it is important to check if benchmarks are still or were recently updated because as everything advances really fast in our days in the domain of computer science, it is important that everything is updated and

still supported.

A lower bound was set for maintainability as the separation between old benchmarks and still maintainable benchmarks. The lower bound is of the year 2017 (gap of 5 years).

From the results of this column, 9 out of 10 benchmarks are still maintainable and only 1 (PARSEC) benchmark is considered as an old benchmark.

The documentation is also an important part, if someone would like to run a benchmark.

By looking at the column of documentation, 2 out of 10 benchmarks don't have any documentation which can be considered as a big minus because they can not be run by third parties.

Another important column is the "Single PC run". Remember that our study is based on personal devices, more precisely on laptops and desktop computers. Thus this property has to be satisfied to be considered as a relevant benchmark for us. From the results, all 10 benchmarks satisfy this property.

The last property column is "Concurrency". The goal is to see if the benchmarks are used for assessing concurrency. The reason to perform this check it is because that previously if the paper had a subtopic concurrency then all benchmarks in this paper are considered as benchmarks which assess concurrency. But in reality, it may not be the case because, for instance, only one benchmark out of 3 benchmarks is used for assessing concurrency in a paper and others are not. So, it is important to perform this check through online sources.

As a result of this check, 2 out of 10 benchmarks (MemBench and SciMark) are present in concurrency papers but are not used for assessing concurrency.

By looking on all properties, some potential "good" benchmarks can be identified. A "good" benchmark is a benchmark where all properties are satisfied.

- NAS Parallel Benchmark

- Linpack
- Rodinia
- Dacapo
- COSMIC

These 5 benchmarks listed above can be considered as potential "good" benchmarks. Only potential because other additional properties have to be seen.

3.4.2.2 Identification of good benchmarks

By considering tables 3.8, 3.9 and 3.14. Some "good" benchmarks can be identified. Again a "good" benchmark is a benchmark where all properties (from tables 3.8, 3.9 and 3.14) are satisfied.

The "good" benchmarks are:

- NAS Parallel Benchmark
- Linpack
- Rodinia
- Dacapo

Thus this 4 benchmarks can be considered as "good" benchmarks.

3.4.2.3 Concurrency in papers

Paper_id	Benchmarks	How much concurrency?
IEEE: 1	NAS Parallel Benchmarks	+++
IEEE: 7	PARSEC	+++
IEEE: 10	Linpack and HPL 2.1 benchmarks	+++
IEEE: 19	Java's Thread-Safe Collections	+++
IEEE: 54	NAS Parallel Benchmarks	+++
IEEE: 54	Rodinia	+++
IEEE: 55	Rodinia	++
ACM: 3	Rodinia	+
ACM: 6	Java Collections	+
ACM: 18	Rodinia	+++
ACM: 19	Dacapo	++
ACM: 19	Renaissance	++
ACM: 42	Dacapo	+
ACM: 42	SciMark	+
ACM: 42	MemBench	+
ACM: 87	COSMIC	+

Table 3.15: Level of concurrency in papers

Table 3.15 is composed of three columns such as Paper_id, Benchmarks, and How much concurrency? The goal of this summary table is to check how much concurrency is present in each selected paper. The column of benchmarks allows us to see which benchmarks are present in each paper.

For evaluating the concurrency in each paper, three levels are considered. These levels are defined with the help of following properties:

Properties:

- P1: Benchmarks used for assessing concurrency
- P2: Results related to concurrency (i.e: graphs as threads over energy consumption or threads over time. It can be also represent as tables)

- P3: Experimental settings consider concurrency (i.e: #threads,...)

Levels:

1. +++: $P1 \wedge P2 \wedge P3$
2. ++: $(P1 \wedge P2) \vee (P1 \wedge P3) \vee (P2 \wedge P3)$
3. +: $P1 \vee P2 \vee P3$

As final result, 6 papers out of 12 have a good level of concurrency where the level 1 (+++) is satisfied. Only 2 papers out of 13 where level 2 (++) is satisfied and only 4 papers where level 3 (+) is satisfied.

3.4.2.4 Closeness of papers and benchmarks to Vacation2

Experiment close to Vacation2?

Benchmarks	Paper_id	Closeness
NAS Parallel Benchmark	IEEE: 1	++
NAS Parallel Benchmark	IEEE: 54	++
PARSEC	IEEE: 7	++
Linpack	IEEE: 10	++
Java's Thread-Safe Collections	IEEE: 19	++
Java Collections	ACM: 6	--
Rodinia	IEEE: 54	++
Rodinia	IEEE: 55	--
Rodinia	ACM: 3	--
Rodinia	ACM: 18	++
Dacapo	ACM: 19	+-
Dacapo	ACM: 42	--
Renaissance benchmark suite	ACM: 19	+-
MemBench	ACM: 42	--
SciMark	ACM: 42	--
COSMIC	ACM: 87	--

Paper close to Vacation2: IEEE: 1,54,7,10,19, ACM: 18

Table 3.16: Closeness to Vacation2

Table 3.16 shows, how close each benchmark is to Vacation2 based on results from the papers. For understanding, how close a benchmark is to Vacation2, some criteria have to be introduced based on knowledge of Vacation2.

Vacation2 is a benchmark that assesses concurrency aimed at evaluating transaction mechanisms. What has been already observed is the variation of time over threads as well as speedup over threads. Our future goal will be to see this benchmark regarding energy consumption.

Based on this knowledge, some criteria can be introduced:

1. ++: Considering multi-threading in experimental setup, Graph/results based Energy, Execution time and Threads.
2. +-: One of two conditions above is satisfied
3. - -: Static Multi-threading or no mention

Remark: If at least one time, the highest criteria is satisfied, then only the highest is considered. Example: For Rodinia benchmark, there is ++ and also some - - but as there exists at least one ++ then this benchmark will be considered close to Vacation2.

Based on results from the papers, 5 benchmarks (NAS Parallel Benchmark, PARSEC, Linpack, Java's Thread-Safe Collections (Java Collections), and Rodinia) out of 10 can be considered close to Vacation2, based on the papers description and the results in the papers where these benchmarks are present.

Two benchmarks (Dacapo, Renaissance benchmark suite) have some similarities with Vacation2 but are still not so close. For Dacapo and Renaissance benchmarks some graphs related to energy, execution time and a number of threads are present. Thus one of two conditions is satisfied.

Other benchmarks (MemBench, SciMark, and COSMIC) are not close to Vacation2.

By considering two previous tables (Table 3.15 and 3.16) about concurrency and closeness to Vacation2. It is possible to see which benchmarks are similar to Vacation2.

By considering the "good" benchmarks and now the results from two tables (3.15 and 3.16), some benchmarks can be identified as relevant for comparison

with Vacation2.

The useful benchmarks are the following:

- NAS Parallel Benchmark
- Linpack
- Rodinia

3.5 Answering Research Questions 1

Remember that at the beginning of this thesis, some research questions have been mentioned and which we would like to respond.

Indeed, by looking at our SLR, the first research question (what are the existing benchmarks that have been used to access performance of concurrent models and its associated energy consumption?) can be answered.

To answer this research question, almost all benchmarks can be considered from the final set of papers, except 2 (MemBench and SciMark) because by looking at the benchmarks properties table (Table 3.14), it is possible to see that they are not benchmarks that allow assessing concurrency. Thus in total, there are 8 benchmarks.

List of benchmarks:

- NAS Parallel Benchmark
- PARSEC
- Linpack
- Java's Thread-Safe Collections (Java Collections)
- Rodinia
- Dacapo
- Renaissance benchmark suite

- COSMIC

In addition, some further researches were done for identifying the "good" benchmarks which are NAS Parallel Benchmark, Linpack, Rodinia and Dacapo benchmarks in 3.4.2.2.

3.6 Catalog of benchmarks

Based on results obtained during the SLR, a catalog of benchmarks can be created.
A catalog of benchmarks will allow developers to compare their software/benchmarks/models with the benchmarks present in the catalog regarding the energy consumption.

The catalog is composed of four attributes:

1. Benchmarks
2. Profiling tools
3. Benchmarks usage
4. Paper references

Benchmarks	Profiling tools	Usage & References
NAS Parallel Benchmark	- RAPL (IEEE: 1) - Application Power Management library (IEEE: 54)	- Programming languages (IEEE: 1) - Assess configurations of parallel applications on a turbo-compliant processor (IEEE: 54)
PARSEC	- RAPL (IEEE: 7)	- Metrics evaluation (IEEE: 7)
Linpack	- RAPL (IEEE: 10)	- Evaluating and comparing Intel programming models based on scenarios (IEEE: 10)
Java's Thread-Safe Collections (Java Collections)	- RAPL (ACM: 6) - jRAPL (IEEE: 19) (ACM: 6)	- Java collections evaluation (IEEE: 19) - Assessing Java Collection Framework (JFC) implementations (ACM: 6)
Rodinia	- Application Power Management library (IEEE: 54) - Intel Performance Counter Monitor (PCM) API (IEEE: 55) - x-MeterPU (ACM: 3) - lm_sensors application (ACM: 18)	- Assess configurations of parallel applications on a turbo-compliant processor (IEEE: 54) - OpenMP workloads on multicore systems (IEEE: 55) - Programming frameworks (OpenMP, openACC, OpenCL and CUDA) (ACM: 3) - Resource Allocator and Scheduler mechanism (ACM: 18)
Dacapo	- RAPL (ACM: 19) - Intel's Performance Counter Monitor (PCM) tool (ACM: 42)	- Impact of Java Virtual Machines (ACM: 19) - Framework evaluation (ACM: 42)
Renaissance benchmark suite	- RAPL (ACM: 19)	- Impact of Java Virtual Machines (ACM: 19)
COSMIC	- DSENT (ACM: 87)	- Network-on-Chips (NoCs), chips evaluation (ACM: 87)

Table 3.17: Catalog of benchmarks

With the help of this catalog, developers will be able to compare their code or models with the benchmarks with respect to energy consumption.

For instance, when a developer would like to do a comparison based on languages, he can look at the usage of the benchmark where benchmarks were used for languages and then look in the referenced papers to understand how much effort/modifications, he has to do to the benchmark for being able to do a comparison for his case. A catalog gives references and clues about how to perform a comparison based on the benchmarks and their usages for measuring energy consumption.

At the end of this thesis, a new benchmark can be included in this catalog which is the Vacation2 benchmark.

Chapter 4

Experiment: Energy performance analysis of concurrency models

In this chapter, an experiment is done to perform an analysis of concurrency models for assessing their energy consumption. In order to achieve this goal, the Vacation2 benchmark is used, based on Chocola (a Clojure library for concurrent and parallel programming).

Replication package for this experiment can be found here: https://github.com/olexstet/Master_Thesis_SO/tree/main/Experiments

4.1 Vacation2 benchmark

In the 1st chapter (section 1.3), we had a small overview on what is the Vacation2 benchmark and Chocola. Let's now see how the Vacation2 benchmark works based on Chocola [1][2].

The Vacation2 benchmark is used for emulating a travel reservation system. During the execution of the benchmark, there are several actors (primary actors) that keep the requests coming from customers in order to perform transactions. These actors can be seen as threads. In addition, only one transaction at time can be performed by an actor.

A transaction is a reservation from a customer. A transaction is composed of three items: flights, room and car where flights are book in two directions for a more realistic situation. Moreover, the cheapest flights, room, and car are booked which give even more realism.

In Vacation2 benchmark, there are two modes Original and Txact. For original mode, each transaction can be seen as an atomic action for booking flights, room and car. However for Txact, it is not atomic but a more advanced concurrency technique is used for booking. More precisely, a transaction is split in three parts (3 items) and messages are sent to secondary actors where messages can be seen as a request for booking one of three items.

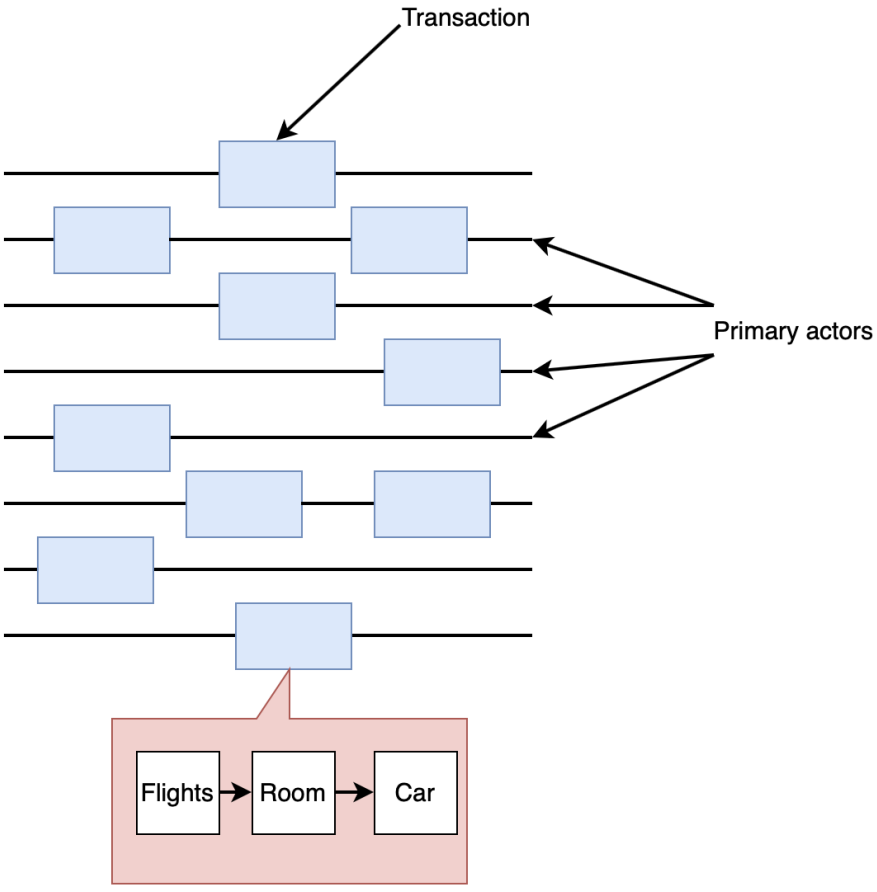

Figure 4.1: Vacation2: Example of Original mode

The figure 4.1 represents an instance of how Vacation2 benchmark is used for Original mode. It is possible to see that in this particular instance, there are 8 primary actors which accept requests for performing transactions and each transaction can be seen as one block where three items are booked sequentially at once.

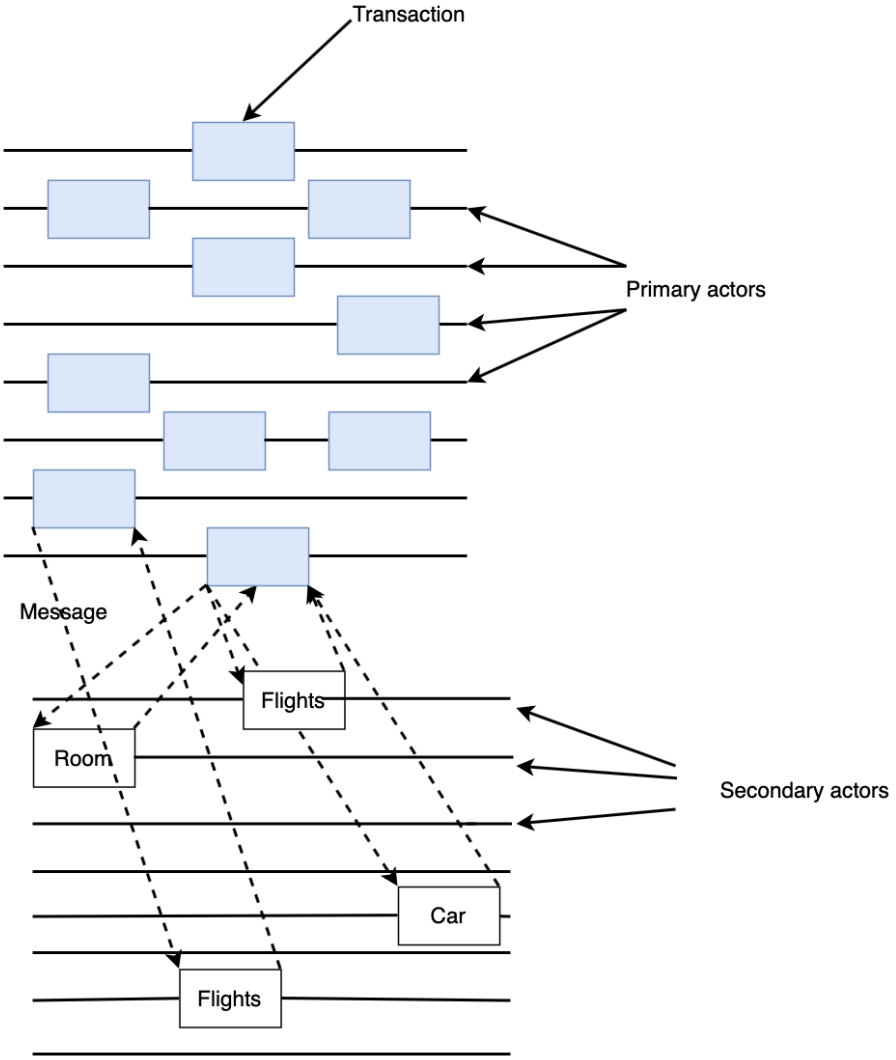

Figure 4.2: Vacation2: Example of Txact with 8 secondary threads

The figure 4.2 represents an instance of how Vacation2 benchmark would work for Txact mode. From this figure, it is possible to see that again 8 primary actors accept requests for transactions. But each transaction is processed differently than previously because now a transaction is separated and messages are sent to

secondary actors where a message can be seen as a booking for one of three items. In secondary actors, the message is received and is processed by booking the item.

The main difference between Original and Txact modes, is that in Original, the three items are booked sequentially, one after another, and in Txact they are booked in parallel but still into the content of a common transaction.

4.2 Experiment description

This experiment can be separated into three phases. The first phase is experiment setup. The second phase is execution of the experiment and the third phase is the post-processing.

4.2.1 Experiment setup

Before the experiment, there are two things to be clarified. The first is the profiling tool to use for measuring the energy consumption and second is which hardware to use for the experiment.

4.2.1.1 Profiling tool

For selecting the profiling tool, let's again look on SLR where tools are observed for different papers (Table 3.6). As conclusion, the RAPL tool is the most used. Thus, RAPL is a good candidate. A small problem with RAPL is that it is a bit difficult to use. Thus, another tool was been used which is based on RAPL. This tool is perf [21].

Perf is a profiling tool which allows us to measure energy consumption. It can be only used on Linux operating systems such as Ubuntu. Moreover, it is really simple in use.

4.2.1.2 Hardware

The experiment is done on two different machines. The reason for using two machines and not only one is related to our research question (3.a in 1.4) which

consists to see what is the energy performance of modern computers vs old computers.

The characteristics of two machines are the following:

Components	PC1	PC2
Operating system	Ubuntu 20.04.3 LTS	Ubuntu 20.04.3 LTS
Processor	i5-8500 CPU @ 3.00GHz	i7-4790 CPU @ 3.60GHz
Core Count	6	4
Thread Count	6	8
RAM	8 GB	8 GB
Purchase year	2016	2020

Table 4.1: Hardware characteristics

From hardware point of view, the most important is the processors that have to be different in comparison because our experiment is based on concurrency and parallelism. What can be seen is that PC1 has a processor of Core i5 with 6 threads at max and PC2 has Core i7 with 8 threads at max. The reason to use Intel cores can be seen in the SLR (Hardware and software Table 3.7) where the processors from Intel were the most used, usually the Core i7.

Another choice can be seen for the operating system. Based on the SLR (Table 3.7), Ubuntu is the most used. This fact also has some importance when choosing the profiling tool because "perf", as indicated previously, can be used only on Linux operating systems.

4.3 Experiment execution

The execution of the experiment is done through a bash script. This script allows setting the variables for the Vacation2, executing the command for the Vacation2, using the profiling tool (perf) for measuring the Vacation2 energy consumption,

and defining the number of times the experiment is repeated.

Let's take a better look at how the experiment is running.

4.3.1 Vacation2 variables

Based on [22], there are in total 8 variables to be set when running the Vacation2 benchmark. However, only 7 of them are relevant as the 8th variable is used for logging.

The relevant variables are the following:

- v: type of transactional model to be used when running Chocola. Possible values are: Orginal or Txact
- w: number of primary actors (this actor is implemented as a thread). The parameter ranges from 1,2,4 to 64, with steps of 2.
- s: number of secondary actors. The parameter ranges on 0,1,2,8,64. If the variable "v" is "Original", then the value of secondary actors must be 0. In case, its value is "Txact", then the value is 1,2,8 or 64.
- t: #reservations

  This number indicates the number of requests to book a reservation. We assume that a request is made by a customer. In the implementation of vacation2, a customer (c) books a holiday (i.e. booking flights, car and room) for people between 1 and 5.
- n: the number of queries for each requested reservation.
- r: #flights/rooms/cars. These are also referred to as items.
- p: password generation. This is a password which is generated after the reservation for three items (flights, car and room).

Based on the above variables 4 configurations for experiment are defined:

Variables	Default configuration	Configuration 1	Configuration 2	Configuration3
v	Original & Txact	Original & Txact	Original & Txact	Original & Txact
w	1,2,8,..., 64 (step 2)	1,2,8,..., 64 (step 2)	1,2,8,..., 64 (step 2)	1,2,8,..., 64 (step 2)
s	0,1,2,8,64	0,1,2,8,64	0,1,2,8,64	0,1,2,8,64
t	1000	10	50	100
n	10	10	10	10
r	50	50	50	50
p	5	5	5	5

Table 4.2: Experiment configurations

From the table above, it is possible to see that the 4 configurations are almost the same. The only variable which is modified is t.

Before, the experiment some pre-tests are perform for identifying the variables which have the highest effect on performance (execution time and load). As result, the variable t was selected.

All 4 configurations are repeated 30 times in order to increase the accuracy of the measurements. Then, median is computed over these 30 executions for obtaining a more global values and a better overview.

Why to use "median"? Despite the fact that in SLR, a lot of studies used "mean" for representing their results, in this experiment median is preferred. The reason is that "median" can not be easily influenced by outliers, which is not the case when using "mean" where one single outlier can have a big influence on a final value.

4.3.2 Execution of Vacation2 and usage of Perf

The execution of the Vacation2 benchmark and the perf profiling tool is related because perf measures the energy consumption based on the command which allows running the Vacation2.

The command with the perf tool is the following:

$output = \$(sudo\ perf\ stat\ -o\ perf-\${j}-\${i}-\$1.txt\ -r\ 1\ -e\ power/energy-cores/,power/energy-ram/,power/energy-gpu/,power/energy-pkg/$

$lein\ run\ --\ -v\ \$1\ -w\ \${i}\ -s\ \$2\ -t\ \${num\_reservations}\ -r\ \${num\_flights\_cars}\ -n\ \${num\_queries}\ -p\ \${p}\})$

The first line (in orange) of this command is how the perf tool is used for measuring energy consumption. From the command, it is possible to see that the perf tool is called with $perf$. For indicating to perf that something has to be measured, $stat$ is used. To write the result $-o$ is used followed by the name of the perf file where the data measured is written. Then, the components on which the measurements have to be done such as cores, ram, gpu and pkg are declared as $power/energy-cores$, $power/energy-ram$, $power/energy-gpu$ and $power/energy-pkg$.

The second line (in cyan) is the execution of the Vacation2 where $lein\ run$ is used for running the Vacation2 benchmark following by the variables.

An instance of execution of Vacation2 with "perf" can be found here: https://github.com/olexstet/Master_Thesis_SO/blob/main/Experiments/Linux-PF0222/Experiment1/energy-efficiency/src/Vacation2-energy-consumption-RAPL.bash

4.3.3 Collection of results

During the execution of the Vacation2 benchmark, the perf profiling tool writes the result for a particular setting of the experiment, based on the command above, into a txt file.

```
# started on Mon Jun 13 18:07:14 2022

 Performance counter stats for 'system wide':

            100,91 Joules power/energy-cores/
             11,56 Joules power/energy-ram/
              2,92 Joules power/energy-gpu/
            158,90 Joules power/energy-pkg/

          5,603450285 seconds time elapsed
```

Figure 4.3: Example of perf output

In Figure 4.3, 5 values are written as the output of the perf profiling tool. Four values are the values measured for energy consumption of cores, ram, gpu, and pkg. The last value is the time taken by the perf for measuring these values (execution time of Vacation2 is included in this time).

The Figure 4.3 allows to see only the result for one combination of variables, but our goal is to have all of them in one single file.

In order to achieve this, for each setting of the experiment, there are 5 folders such as Original, Txact-01, Txact-02, Txact-08 and Txact-64 where the results are stored for each run as a text file. Thus, in total in each folder, there are 30 txt files where each file is the concatenation of perf results for one single run.

Remark: The name of the folders is based on the mode and secondary actors.

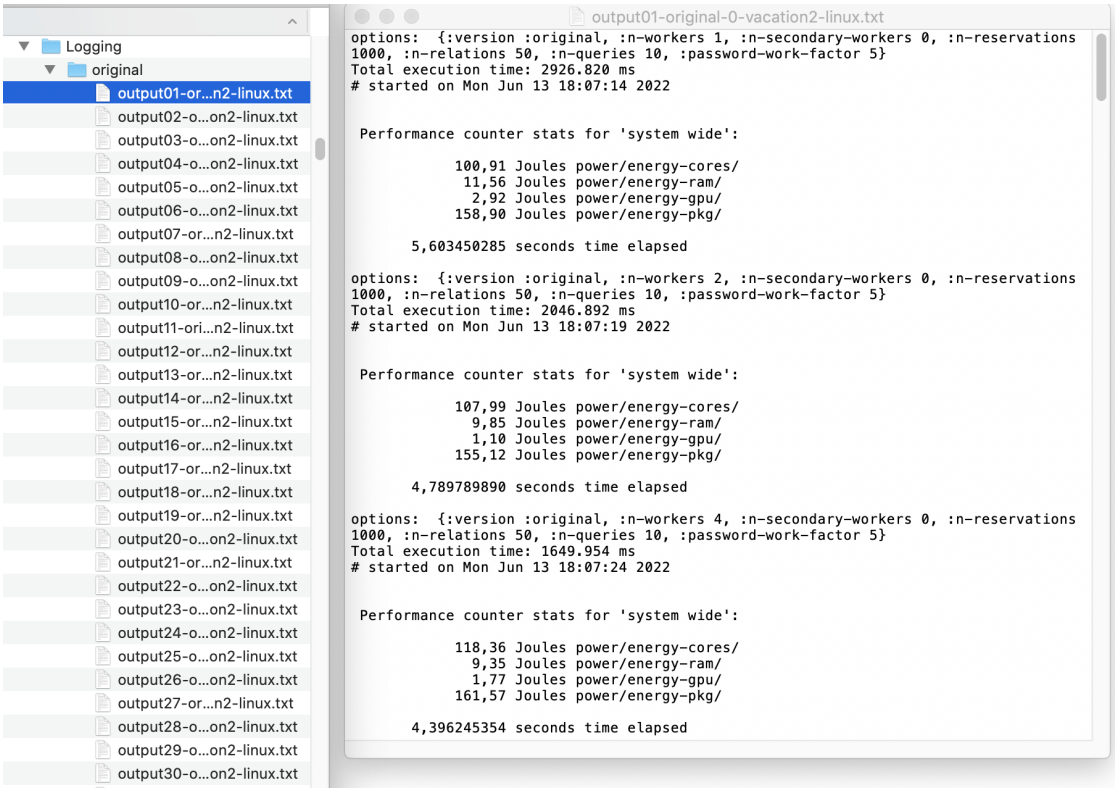

Figure 4.4: Example of output of combined perf files

Figure 4.4 shows the 30 files (https://github.com/olexstet/Master_Thesis_SO/tree/main/Experiments/Linux-PF0222/Experiment1/energy-efficiency/output/Logging/original) for the original mode where the secondary actor is 0. On the right side, there is a text file that contains all the results, for one single run over 30, for the original mode with secondary actor 0.

This file is a concatenation of all perf files (https://github.com/olexstet/Master_Thesis_SO/blob/main/Experiments/Linux-PF0222/Experiment1/energy-efficiency output/Logging/original/output01-original-0-vacation2-linux.txt) where in addition variables and execution time (Vacation2 time) is mentioned for each record.

4.4 Post-processing

Based on the data collected from the experiment, a processing of this data has to be done for better organizing and creating some visual representations to ease the

analysis.

To achieve this goal, a python script is written where the data from output files (Figure 4.4) is processed line by line.

Python script: https://github.com/olexstet/Master_Thesis_SO/blob/main/Experiments/processing.py

4.4.1 Data processing

What is done is the following:

1. Retrieve the variables of the Vacation2 execution
2. Retrieve data about execution time for a given experimental setting
3. Retrieve data about energy consumption (Joules) for a given experimental setting
4. Retrieve execution time of perf (seconds) for a given experimental setting

The process described above is done sequentially until the end of a text file when there are no more lines to be analysed.

All the data for all output files for each folder were processed and represented in a table to obtain a more clear view of the results. Moreover, some additional properties are computed based on the collected data such as:

- Power: for each measured component (Processor, RAM, GPU and PKG), power is computed based on P = E/t
- Ratio time: represent the ratio of execution time of the command for running the Vacation2 and the perf execution time. Our goal is to see whether this ratio is constant.

  If a ratio is constant, this means that execution time (perf) can be used in our analysis because the execution time (perf) can be divided into two parts: running a particular setting of experiment and measuring the values.

By saying that the ratio is constant would mean that measurement of values is constant and doesn't affect the execution time when running a setting. In other words, the execution time of perf will be the execution time (when running a setting) plus constant time.

version	n-workers	n-secondary-workers	n-reservations	n-relations	n-queries	password-work-factor	Energy Cores	Energy Ram	Energy Gpu	Energy Pkg	Time (perf) (s)	Time (exec) (s)	Ratio time (%)	Power_Cores	Power_Ram	Power_Gpu	Power_Pkg
original	1	0	1000	50	10	5	100.91	11.56	2.92	158.9	5.6034503	2.92682	0.5223246	18.01	2.06	0.52	28.36
original	2	0	1000	50	10	5	107.99	9.85	1.1	155.12	4.7897899	2.046892	0.4273448	22.55	2.06	0.23	32.39
original	4	0	1000	50	10	5	118.36	9.35	1.77	161.57	4.3962454	1.649954	0.3753098	26.92	2.13	0.4	36.75
original	6	0	1000	50	10	5	123.23	9.13	0.79	164.69	4.3281993	1.593345	0.3681312	28.47	2.11	0.18	38.05
original	8	0	1000	50	10	5	131.51	9.27	1.59	173.11	4.3074861	1.560492	0.3622744	30.53	2.15	0.37	40.19
original	10	0	1000	50	10	5	136.04	9.44	1.45	178.35	4.4242575	1.687023	0.3813121	30.75	2.13	0.33	40.31
original	12	0	1000	50	10	5	133.82	9.06	0.1	173.99	4.3422551	1.672436	0.3851538	30.82	2.09	0.02	40.07
original	14	0	1000	50	10	5	140.42	9.33	0.07	181.39	4.4667197	1.779165	0.3983158	31.44	2.09	0.02	40.61
original	16	0	1000	50	10	5	139.1	9.19	0.07	179.65	4.4195705	1.757289	0.3976153	31.47	2.08	0.02	40.65
original	18	0	1000	50	10	5	142.17	9.38	0.08	183.52	4.4958559	1.803911	0.4012386	31.62	2.09	0.02	40.82
original	20	0	1000	50	10	5	148.38	9.62	0.08	190.98	4.5929418	1.894279	0.4124326	32.31	2.09	0.02	41.58

Figure 4.5: Experiment results (some results)

One instance of full table for one experiment configuration can be found here: https://github.com/olexstet/Master_Thesis_SO/blob/main/Experiments/Linux-PF0222/Experiment1/Visualization/Table_full.csv

4.4.2 Results visualization

Based on the table (Figure 4.5) obtained previously, some results can be represented visually as graphs. As the experiment was repeated 30 times for each experimental setting, the values used in graphs are the medium over the 30 runs.

In total there are three types of graphs:

- Energy - #Threads (primary actors)
- Power - #Threads (primary actors)
- Execution time (perf time) - #Threads (primary actors)

There are in total 9 graphs: 4 graphs are for energy, 4 for power and one for execution time. The reason to have 4 graphs for energy and power is that there are 4 components on which the energy measurements was done: cores, ram, gpu and pkg.

Some examples of graphs:

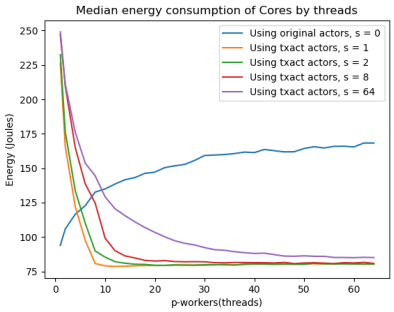

Figure 4.6: Instance of Energy - #Threads for cores

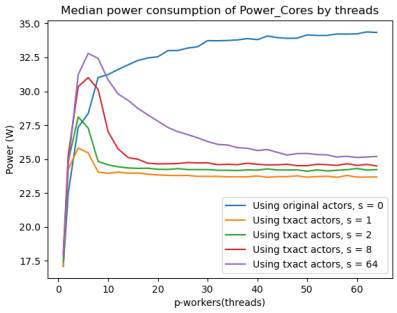

Figure 4.7: Instance of Power - #Threads for cores

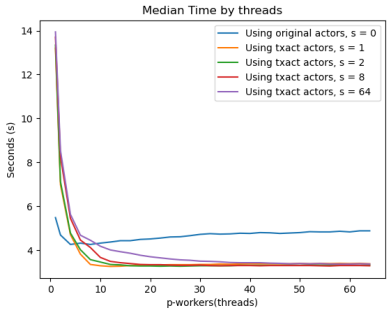

Figure 4.8: Instance of Execution time (perf) - #Threads

This is the final result for one configuration (Default configuration) of the experiment. The same is done for all other configurations. The analysis of graphs is done in the following sections in order to answer some research questions.

4.5 Results analysis

Based on the graphs created previously, an analysis of results can be done based on these graphs. Our goal is to observe results related to concurrency and energy consumption. Thus, not all created graphs have to be considered but only graphs related to cores (processors) components.

Moreover, with the help of 4 different configurations for 2 different types of hardware, some conclusions about the energy and Chocola transactional system can be done.

Results for PC1

Default configuration:

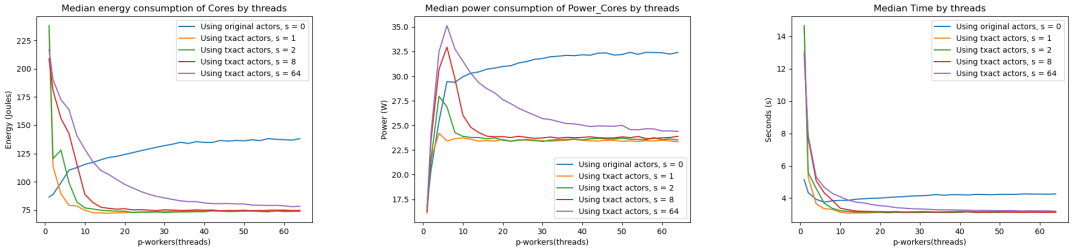

Figure 4.9: Results default configuration

Configuration 1:

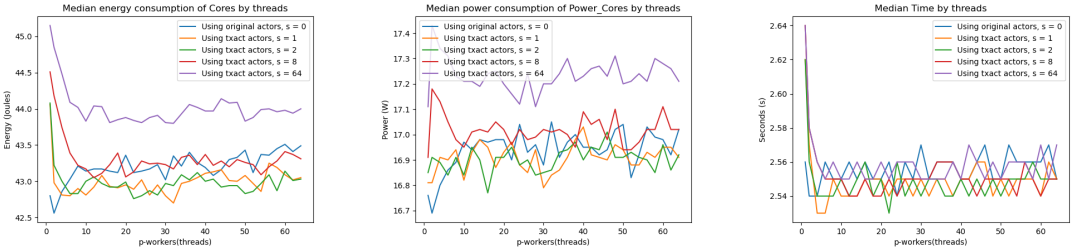

Figure 4.10: Results configuration 1

Configuration 2:

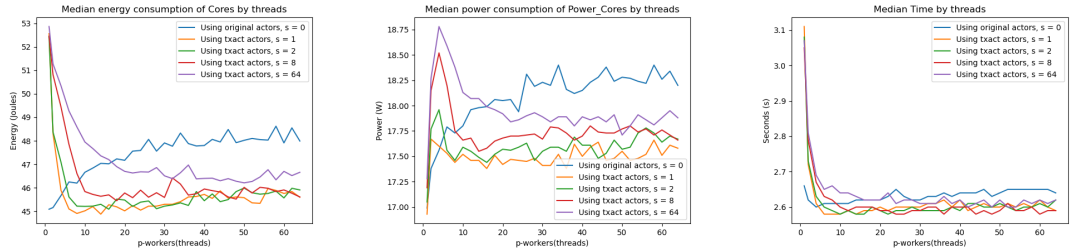

Figure 4.11: Results configuration 2

Configuration 3:

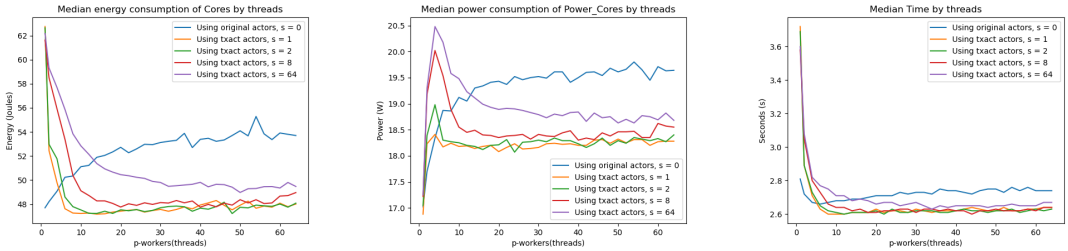

Figure 4.12: Results configuration 3

Results for PC2

Default configuration:

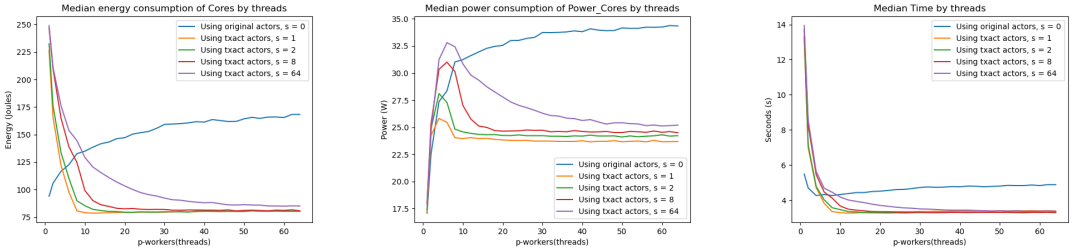

Figure 4.13: Results default configuration

Configuration 1:

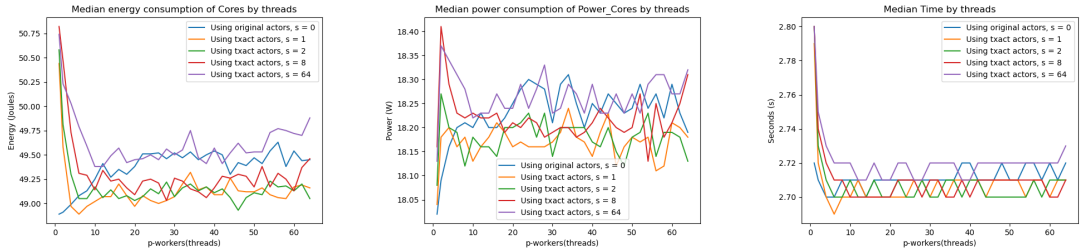

Figure 4.14: Results configuration 1

Configuration 2:

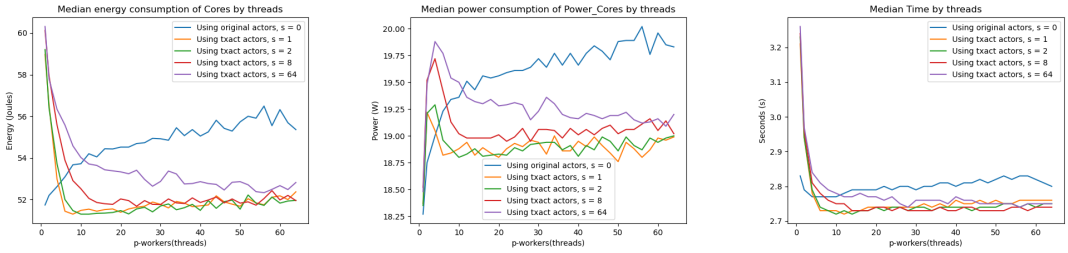

Figure 4.15: Results configuration 2

Configuration 3:

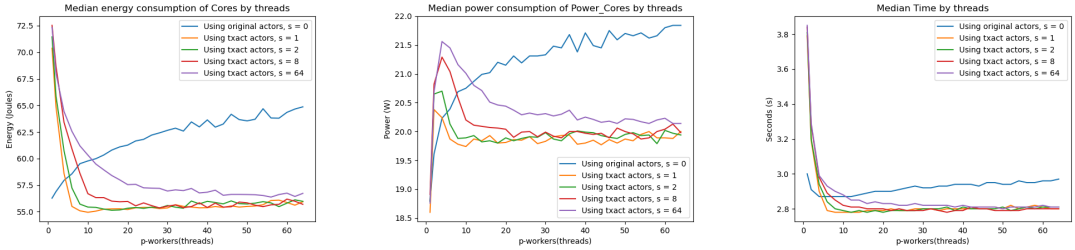

Figure 4.16: Results configuration 3

4.5.1 Results description

By observing the graphs for two types of hardware, some similarities can be identified. To be more precise both hardware have the same graphs representation in terms of behavior for two modes (Original and Txact) (There are no much difference for curves representation in the graphs). The only difference is when the peaks happen for energy consumption, power consumption, and execution time.

4.5.1.1 Original vs Txact

By using the Vacation2 with Chocola is possible to see the difference between the Original and Txact modes where the Orginal mode is part of the original vacation benchmark and Txact is the mode that is based on Chocola and which goal is to speed up the execution by using a more advance concurrency technique.

By looking at all configurations results, in the majority of cases, Txact performs better than Original in general. By looking at values of threads (p-workers) from 0 to 10, the energy consumption and the execution time are better for Original mode until the number of threads is approximately equal to 5 and then Txact mode starts to perform better.

This means that the Original mode works better for smaller number of threads but then starts to have a disadvantage over the Txact mode.

However, there are some exceptions, for instance for configuration 1 where the Original mode is always better than Txact with 64 secondary actors. This case happens because the number of reservations is much smaller than the number of available items. One reason for this problem is based on deadlocks because too many secondary actors are waiting for available resources.

Another interesting difference between Original and Txact is that in the beginning, the Original mode increases quickly and Txact decreases quickly but then both start to converge. By looking at graphs, energy consumption for Original will continue to increase and for Txact, it will stay stable and even increase also a bit.

For the Original, this continuation of energy increase can be explained that more threads being used. Thus more power is required, it is what can be seen on power graphs.

For Txact, it is different because energy consumption doesn't continue to increase much as for Original but still increases. This is due to the power decrease after the peak.

One possible reason why the power decreases is due to the concurrency model used for Txact because even if all processors and threads are used, less power is required.

4.5.1.2 Peaks

In the previous section, we saw that Txact performs better than the Original after a certain number of threads. In addition, at a certain point, both start to converge and stabilize in terms of energy consumption, power, and execution time.

Now, let's take a look at why these peaks happen and what happens before and after the peaks more in detail?

Before the peaks, the Original mode performs as expected because with fewer threads, lower is the energy and power consumption but higher is the execution time.

However, for Txact (1,2,8, and 64 secondary actors) it is not the same behavior because energy consumption and execution time are initially too high and power increase also quickly. The key for understanding this phenomenon is that, energy consumption and execution time decrease with a higher number of threads. This means that Vacation2 with Chocola has a problem with a smaller number of threads.

A reason why this happens is due to deadlocks because by looking at graphs for execution time , it is possible to see that execution time initially is too big which means that threads are in waiting mode.

By increasing the number of threads (until a peak happen), execution time become smaller. This means that the problem with deadlocks is reduced. Moreover, by looking on power graphs, it can be seen that power consumption increases, it is normal because they are more threads that are running.

At a certain moment for energy consumption, peaks happen which can be seen with Txact mode but less with Original mode because for Original, it continues to increase further. After a peak, convergence and stabilization can be observed. There is a simple explanation for this phenomenon. The reason is that the maximum capacity of hardware in terms of processors and available threads is reached.

In addition, when a peak happens for Txact, a peak also happens for power because processors are at their maximum potential. The same behavior (peaks)

can be observed also for execution time.

For Original it seems that even reaches the maximum capacity of processors, power is not at its maximum (if compared with Txact). One possible reason is the way how concurrency is implemented for these two modes which is different.

By looking more precisely at Txact and Original modes, both start to behave in the same way after the peaks because both increase in energy consumption (for Original it is more evident to see). These increases happen because for both modes, actors (threads) are waiting for resources because as maximum processor capacity is reached, some threads are in "waiting mode" and waiting for resources or for something to be executed.

4.5.1.3 Performances of secondary actors

Now, let's focus only on Txact mode. Remember that there are four different types of Txact modes where the number of secondary actors varies such as 1,2,8 and 64.

By looking at all configurations, the best out of four can be seen, where the best is the Txact with 1 secondary actor (orange line in Figures 4.9 - 4.16). Why it is the best? It is because it has the lowest value (peak) for energy consumption. It has the lowest value for energy consumption because it uses less power and execution time is also the lowest. In addition, by comparing the values in general for different threads (primary actors), it is possible to see that Txact with 1 secondary actors has lower values than other ones.

Then, there is also Txact with 2 secondary actors and 8 secondary actors. Both are close to each other but 2 secondary actors perform in general better than 8 secondary actors because by looking on energy consumption for both, it is possible to see that in general Txact for 2 secondary actors has lower energy values than Txact with 8 secondary actors.

The last is the Txact with 64 secondary actors which are relatively far in terms of energy consumption with respect to other secondary actors and can be consid-

ered as the worst out of four.

As conclusion, more there are secondary actors, worst is the performance for Txact mode. But this is only the case by considering our hardware and our settings because with different hardware and settings, it may be also different.

4.5.1.4 Hardware differences

For performing a comparison in terms of performance of two hardware, let's see what are the highest and lowest values of energy consumption, power, and execution time for each configuration.

PCs	Configurations	Energy consumption (Joules) (measured)	Power consumption (W) (computed)	Execution time (s) (measured)
PC1	Default configuration	[75-235]	[17-35]	[3-15]
PC2	Default configuration	[75-250]	[17-35]	[3-14]
PC1	Configuration 1	[42.5-45.5]	[16.7-17.5]	[2.53-2.64]
PC2	Configuration 1	[49.0-50.8]	[18.0-18.4]	[2.68-2.80]
PC1	Configuration 2	[45-53]	[17.0-18.8]	[2.6-3.1]
PC2	Configuration 2	[51-61]	[18.3-20.0]	[2.7-3.3]
PC1	Configuration 3	[47-63]	[16.9-20.5]	[2.6-3.7]
PC2	Configuration 3	[55-72.5]	[18.5-22.0]	[2.8-3.9]

Table 4.3: Hardware comparison results

Based on the table 4.3, both hardware have difference in values. For energy consumption, power consumption, and execution time, PC2 has values which are in general higher than for PC1. This means that PC1 is a better hardware for this experiment. It is better because the goal is to reduce at maximum these 3 values.

4.6 Answering research question 2

RQ2: Is there any correlation between time performance (speed or duration) and energy performance (energy consumption)?

For answering this research question, both Original and Txact modes are considered .

From the results for Original, when execution time decreased, energy consumption increased. The reason is based on this formula $E = P * t$ because when execution time decreased, power increased a lot (by using more processors resources) and as consequence energy consumption increased. This means that power has a bigger influence than execution time.

From the results for Txact, it is possible to see that whenever execution time decreased, energy consumption also decreased. Based on the formula $E = P * t$, it is possible to conclude that increase in power has no big influence because even though power increased and execution time also decreased, energy consumption also decreased.

In conclusion, the correlation between time performance and energy performance depends a lot on the influence of the power.

4.6.1 Research question 2.a

RQ2.a: What is the impact of concurrency on energy performance?

For answering this research question, only the Original mode (blue line in graphs) can be considered. What can be seen is that by using more processors resources (threads), energy consumption become higher. Thus more there are threads, higher is the energy consumption.

Indeed, in our experiment, a maximum capacity of threads has been reached and there was a stabilization for energy consumption, execution time and power. If there is a hardware with 64 available threads then the energy consumption may be much higher.

4.6.2 Research question 2.b

RQ2.b: How does the energy performance increase over time? Is it linear, logarithmic or exponential?

Based on the curves for Original mode, the energy performance increases logarithmically over time based on our experiment. But during the experiment, the maximum capacity of processors resources is reached. Before reaching the maximum capacity, the curve is close to linear. It can happen that by using a hardware with more processors resources, energy consumption can be linear.

In reality, the energy performance over time is between linear and logarithmic.

4.7 Answering research question 3

Is the industry making efforts in enhancing energy performance?

For answering this research question, research question 3.a: **What is the energy performance of modern computers vs. old computers?** has to be also considered.

By looking at our results for Hardware differences (section 4.5.1.4), it can be concluded that the industry doesn't make a lot of efforts for reducing the energy consumption or even to keep it stable because knowing that PC2 has better hardware than PC1 in terms of processors (i7 vs i5), PC1 performs everywhere better. Better means lower values.

This is for hardware but what about software because in this experiment two modes for concurrency were tested. Indeed, in terms of software, the energy consumption improved because by looking on Txact mode, energy consumption is decreased compared to Original (if not considering a small number of threads).

In conclusion, even if in terms of hardware, the energy consumption didn't improve much, this fact is less visible because, in terms of software, energy consumption improved.

Chapter 5

Threats to Validity

In this chapter, let's look at the threats to validity. Threats to validity can be separated into two major parts: internal and external validity. For internal validity, the confidence in our SLR and experiment is discussed to show that the used approaches and obtained results are the right ones and there are no external factors that can affect the results.

For external validity, the goal is to show that everything done by SLR and experiment can be generalized to other cases and studies.

5.1 Internal validity

For internal validity, let's look at the objectives defined at the beginning of the thesis. The first objective consists to explore how the energy consumption is measured for benchmarks that assess concurrency.

For achieving this objective, a systematic literature review is done where firstly the query is defined. Then, some inclusion and exclusion criteria are applied and lastly, a detailed analysis is done.

In the end, the results obtained show how energy is measured from an experimental point of view and which benchmarks are used for assessing energy consumption.

The results for SLR gave us, the results needed based on our first objective. In terms of the correctness of SLR, it is always debatable because it is always possible

to include some additional keywords or change slightly the query as well as include some additional criteria.

For the experimental part, the objective was to assess the energy consumption of Chocola by using the Vacation2 benchmark. This objective was also achieved by using the Vacation2 benchmark and a profiling tool (Perf).

During the SLR, the results for the profiling tool showed that RAPL is the most used one. But as RAPL is a bit difficult to use, it is why perf is preferred, which is based on RAPL. Thus the link between observation in SLR and our experiment can be rightly seen.

For the experiment, 4 different configurations are used where only one variable was changed every time which gives a better overview of the results (changing multiple variables at once may cause misunderstanding). By using this approach, the correct observation of how the results vary can be guaranteed.

The last objective was to see how concurrency affects energy consumption and how energy consumption varies with respect to performance. This objective is also achieved by looking at our graphs. At the beginning of the thesis, a small hypothesis was defined and which says that by increasing the performance (processors resources), energy consumption will also increase. By observing the results from the experiment (blue line in graphs), this hypothesis is satisfied because by augmenting the number of threads, energy consumption increased.

In addition by using the Vacation2 benchmark with Chocola, how concurrency affects the energy consumption is observed not only for normal concurrency model but even by using more advanced concurrency model.

In conclusion to internal validity, all goals are achieved. For achieving the results, step by steps approaches are used for SLR and reasonable approaches are used for the experimental part where each approach has a reason to be used.

External factors that could influence the results of our experiment are more linked to the hardware because depending on the hardware used, the results can

be different.

In our experiment, the results are obtained from desktop computers, but performing the same experiment on battery-embedded device such as a laptop can influence the results because the power used can be different which will also influence the execution time.

5.2 External validity

For external validity, the goal is to see how our study can be generalized to other studies.

What consists the SLR, the approach used for SLR can be generalized to other studies by using the systematic approach which consists of three main parts: query, inclusion and exclusion criteria, and data extraction. By slightly modifying each part based on research needs, some results for SLR can be obtained.

For the experiment, the final results showed that by increasing the performance, energy consumption will also increase. These results show us that by using concurrency, in general, energy consumption will increase. The same should hold for other case studies where other benchmarks/software are used.

In the results, we saw that energy consumption has an exponential curve with initially some linearity. This means that energy consumption will always at the beginning increase quickly and then at a certain moment, this increase will become lower.

The approaches used for obtaining the results for energy consumption can be also generalized. What was been done, can be separated into three steps: benchmark execution, applying a profiling tool during the execution, and obtention of results from the profiling tool. These can be considered as three main steps for obtaining the energy consumption (in the section of related work, it is possible to see that all experiments work in the same way in different researches).

Chapter 6

Related works

In this chapter, an overview of research works related to our thesis is discussed.

Based on SLR previously done, some related research papers can be identified which are strongly related to the thesis. Remember, in SLR, 12 papers are chosen for the final set which was been then analyzed in detail. During the detailing analysis, the closeness of papers (Table 3.16) to our study was checked based on the experimental approach.

Based on the result of this table, 6 papers are considered as related (close) to our experiment and thus to our thesis. The papers related are the following:

- IEEE: 1 [9]

  In this paper, the NAS benchmark is used for observing the energy consumption of programming languages such as C and Java. As for our thesis, the Ubuntu operating system and an Intel Core i7 are used. For measuring energy consumption, the RAPL profiling tool was used. As final results, three properties are observed such as energy consumption, number of threads, and time for each type of programming language. The final results are represented as graphs such as Energy-#Threads and Time (Seconds)-#Threads.

  In conclusion, a lot of relations to our thesis with this paper can be observed. Moreover, the only big difference is in the benchmark because, in our thesis, the Vacation2 benchmark is used for evaluating Chocola which is a concur-

rent programming language.

- IEEE: 7 [10]

  In this paper, a Parsec benchmark is used to perform metrics evaluation. For the experiment, two different hardware are used, both equipped with a processor of Intel core i7. RAPL profiling tool is used for measuring energy consumption. The properties observed in this paper are the following: energy, power, threads, frequency, and speedup. All these properties are represented in the form of graphs as well as in a table.

  Based on the description of this paper, it can be also seen that this paper is close to our thesis. Moreover, in this paper, the authors did observation also on two different hardware.

- IEEE: 10 [11]

  For this paper, a Linpack and HPL 2.1 benchmark is used for evaluating and comparing Intel programming models based on scenarios where scenarios consider various environment parameters and execution on the Intel host, offload, and native programming models. As hardware, there are two processors of Intel Xeon Phi E5. The operating system on which the experiment is done is a Linux kernel 3.10. The energy consumption is measured with the help of the RAPL profiling tool.

  The final results are represented in the forms of graphs such as Performance (GFfops) - Workload size, Threads - Energy, Execution time - workload size, and Performance per watt - threads.

  Based on this paper, the same approach is used as in our thesis. The only difference is in the benchmark and his usage. In addition, in the paper, some additional properties such as performance (GFfops) and workload size are observed.

- IEEE: 19 [12]

  For this paper, a benchmark of Java's Thread-Safe Collections is used for evaluating the java collections based on the traversal, insertion, and removal methods. The goal is to see what is the energy consumption when performing each method on a different type of collection.

  For measuring this energy consumption, the jRAPL profiling tool is used which is an extension of RAPL for Java programming language. For processors, two different types are used, the first is a 16-core AMD Opteron 6378 processor and the second is an 8-core Intel(R) Xeon(R) E5-2670 processor. The observed threads are: 1, 2, 4,8, 16, 32, 64, 128, and 256. As final results, two graphs are created: Energy - Type of method (Traversal, Insertion, and Removal) and Energy - Number threads.

  By comparing our thesis with this paper, the authors used the same approach by using a profiling tool called jRAPL for measuring energy consumption. They also observed energy over the number of threads as a final result. As well they used two different hardware.

- IEEE: 54 [13]

  In this paper, NAS Parallel and Rodinia benchmarks are used. Both have a goal to assess configurations of parallel applications on a turbo-compliant processor. As hardware, a processor of AMD Ryzen 7 is used for running the experiment. Moreover, the experiment is done using Ubuntu operating system. Application Power Management library is used for measuring energy consumption. As final results, a lot of properties are observed:

  - Relative performance - number of threads
  - Relative Energy - number of threads
  - Relative EDP - number of threads
  - Relative temp - number of threads
  - Relative Aging - number of threads

There are some differences with our thesis. They used a different type of processor, a different profiling tool, and the usage of benchmarks is much different. But the approach used for measuring energy consumption remains the same.

- ACM: 18 [17]

  For this paper, a Rodinia benchmark is used and which has the goal to test a resource allocator and scheduler mechanism. For measuring the energy consumption lm_sensors application is used which is based on embedded sensors in the Linux system. The observation of energy is done on a processor of Intel core i7 and based on 2,4,8,16,32,64,128 threads. The final results are also represented as graphs such as:

  - Execution time - #threads
  - Energy (kWh) - #threads
  - Peak Power (Watt) - #threads
  - CPU Peak Temperature - #threads

  For their experiment, the operating system and hardware used are the same as for our thesis. In addition, the authors also observed the results of energy over the number of threads, execution time over the number of threads, and power over the number of threads. Thus, in terms of approach and representation of the results, it is the same approach as for our thesis.

For observation of energy consumption over all these papers, it is difficult to see the general behavior because it is true that the approaches and representation of results are a bit the same but in terms of energy results, the observation may be different because the benchmarks and their usage are different. Thus results are also different.

Chapter 7

Conclusion

During this thesis, three main parts are performed.

Firstly, a systematic literature review is done for understanding how to measure the energy consumption for benchmarks that assess concurrency.

During the SLR, a set of relevant papers is obtained based on queries, refinements, and criteria. Based on data extraction, the benchmarks which assess concurrency are retrieved as well as additional information such as the profiling tools, operating systems and hardware the most used. All this information help us to set up the experiment. In addition, based on the extracted data, a catalog of benchmarks is created which will help developers to assess the energy consumption.

Secondly, an experiment is done based on the Vacation2 benchmark by using a "perf" profiling tool for collecting the data of energy consumption and performing some post-processing for better visualization of results.

Lastly, an analysis of the results is performed based on obtained data from the experiment. As result, the energy consumption increased while using more processor performance for the Original mode. However, by looking at a more advanced concurrency model Txact, the energy consumption was much lower after a certain processor's performance than for Original. This means that Txact is less performant than Original at the beginning.

Based on these three parts, some research questions could be answered.

From the SLR, how energy consumption can be measured and what are benchmarks for assessing concurrency are understood. (RQ1)

By performing the experiment and observing the results, we saw that energy consumption increased exponentially while increasing the processor's performance (threads). (RQ2)

Moreover, by observing the results of the Vacation2 benchmark with two different modes (Txact and Original) and for different hardware, the industries didn't improve much in energy consumption from the hardware point of view, but they improved from the software point of view. (RQ3)

As a final conclusion, all the objectives set at the beginning of this thesis are achieved and all defined research questions are answered.

For the future work, it may be a good idea to measure the energy consumption on another concurrency benchmark and then to do a comparison with the our experimental results for understanding the energy consumption behavior. In addition, an extension of our catalog of benchmarks can be done by finding and adding some additional benchmarks.

References

1. Janwillem Swalens, Joeri De Koster, and Wolfgang De Meuter. 2021. Chocola: Composable Concurrency Language. ACM Trans. Program. Lang. Syst. 42, 4, Article 17 (January 2021), 56 pages. https://doi.org/10.1145/3427201

2. Chi Cao Minh, JaeWoong Chung, C. Kozyrakis and K. Olukotun, "STAMP: Stanford Transactional Applications for Multi-Processing," 2008 IEEE International Symposium on Workload Characterization, 2008, pp. 35-46, doi: 10.1109/IISWC.2008.4636089. https://ieeexplore.ieee.org/document/4636089

3. Wilhelm Hasselbring. . Benchmarking as Empirical Standard in Software Engineering Research. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3463274.3463361

4. Gustavo Pinto and Fernando Castor. 2017. Energy efficiency: a new concern for application software developers. Commun. ACM 60, 12 (December 2017), 68–75. https://doi.org/10.1145/3154384

5. Luís Cruz. (2021, July 20). Tools to measure software energy consumption from your computer. Luís Cruz. Retrieved June 18, 2022, from https://luiscruz.github.io/2021/07/20/measuring-energy.html

6. Breshears, C. (2009). The art of concurrency. O'Reilly.

7. ACM Digital Library. UCSB Library. (2017, September 13). Retrieved May 23, 2022, from https://www.library.ucsb.edu/research/db/acm

8. Wikimedia Foundation. (2022, February 8). IEEE Xplore. Wikipedia. Retrieved May 23, 2022, from https://en.wikipedia.org/wiki/IEEE_Xplore

9. G. G. Magalhães, A. L. Sartor, A. F. Lorenzon, P. O. A. Navaux and A. C. Schneider Beck, "How Programming Languages and Paradigms Affect Performance and Energy in Multithreaded Applications," 2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC), 2016, pp. 71-78, doi: 10.1109/SBESC.2016.019. https://ieeexplore.ieee.org/document/7828287

10. T. Rauber, G. Rünger and M. Stachowski, "Towards New Metrics for Appraising Performance and Energy Efficiency of Parallel Scientific Programs," 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017, pp. 466-474, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.75. https://ieeexplore.ieee.org/document/8276794

11. R. Gonçalves, A. Girardi and C. Schepke, "Performance and Energy Consumption Analysis of Coprocessors Using Different Programming Models," 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2018, pp. 508-512, doi: 10.1109/PDP2018.2018.00086. https://ieeexplore.ieee.org/document/8374509

12. G. Pinto, K. Liu, F. Castor and Y. D. Liu, "A Comprehensive Study on the Energy Efficiency of Java's Thread-Safe Collections," 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2016, pp. 20-31, doi: 10.1109/ICSME.2016.34. https://ieeexplore.ieee.org/document/7816451

13. S. M. V. N. Marques et al., "The Impact of Turbo Frequency on the Energy, Performance, and Aging of Parallel Applications," 2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC), 2019, pp. 149-154, doi: 10.1109/VLSI-SoC.2019.8920389. https://ieeexplore.ieee.org/document/8920389

14. M. Mirka, G. Devic, F. Bruguier, G. Sassatelli and A. Gamatié, "Automatic Energy-Efficiency Monitoring of OpenMP Workloads," 2019 14th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2019, pp. 43-50, doi: 10.1109/ReCoSoC48741.2019.9034988. https://ieeexplore.ieee.org/document/9034988

15. Suejb Memeti, Lu Li, Sabri Pllana, Joanna Kołodziej, and Christoph Kessler. 2017. Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption. In Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing (ARMS-CC '17). Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3110355.3110356

16. Rui Pereira, Marco Couto, João Saraiva, Jácome Cunha, and João Paulo Fernandes. 2016. The influence of the Java collection framework on overall energy consumption. In Proceedings of the 5th International Workshop on Green and Sustainable Software (GREENS '16). Association for Computing Machinery, New York, NY, USA, 15–21. https://doi.org/10.1145/2896967.2896968

17. Shouq Alsubaihi and Jean-Luc Gaudiot. 2017. PETRAS: Performance, Energy and Thermal Aware Resource Allocation and Scheduling for Heterogeneous Systems. In Proceedings of the 8th International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM'17). Association for Computing Machinery, New York, NY, USA, 29–38. https://doi.org/10.1145/3026937.3026944

18. Zakaria Ournani, Mohammed Chakib Belgaid, Romain Rouvoy, Pierre Rust, and Joël Penhoat. 2021. Evaluating the Impact of Java Virtual Machines on Energy Consumption. In

Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '21). Association for Computing Machinery, New York, NY, USA, Article 15, 1–11. https://doi.org/10.1145/3475716.3475774

19. Matthew Benjamin Olson, Joseph T. Teague, Divyani Rao, Michael R. JANTZ, Kshitij A. Doshi, and Prasad A. Kulkarni. 2018. Cross-Layer Memory Management to Improve DRAM Energy Efficiency. ACM Trans. Archit. Code Optim. 15, 2, Article 20 (June 2018), 27 pages. https://doi.org/10.1145/3196886

20. Md Farhadur Reza and Paul Ampadu. 2019. Approximate Communication Strategies for Energy-Efficient and High Performance NoC: Opportunities and Challenges. In Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI '19). Association for Computing Machinery, New York, NY, USA, 399–404. https://doi.org/10.1145/3299874.3319455

21. Linux support for power measurement interfaces. (n.d.). Retrieved July 14, 2022, from https://web.eece.maine.edu/~vweaver/projects/rapl/rapl_support.html

22. Jswalens. (n.d.). Jswalens/vacation2: "vacation" benchmark from stamp adapted to use transactional actors. GitHub. Retrieved June 25, 2022, from https://github.com/jswalens/vacation2

23. Full SLR: https://github.com/olexstet/Master_Thesis_SO/tree/main/SLR

24. Experiments replication package: https://github.com/olexstet/Master_Thesis_SO/tree/main/Experiments

25. Master thesis (Experiments, SLR and manuscript): https://github.com/olexstet/Master_Thesis_SO