Universidade do Minho
Escola de Engenharia

Tiago André Araújo Monteiro

Energy Consumption on Database Management Systems

July, 2021

Universidade do Minho
Escola de Engenharia

Tiago André Araújo Monteiro

Energy Consumption on Database Management Systems

Master's Dissertation
Master's in Informatics Engineering

Work supervised by
João Alexandre Baptista Vieira Saraiva
Rui Alexandre Afonso Pereira

July, 2021

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Acknowledgements

After this crucial step in my academic career, I would like to thank the following people for their help and support throughout this dissertation:

First of all, I would like to express my sincere thanks to my supervisor and professor, João Saraiva, for the time, patience, knowledge, dedication, and motivation throughout this dissertation and for proposing the subject of this master's thesis. Without his guidance, I would not have solved the problems and challenges I have encountered along this journey.

I would like to extend my sincere thanks to my co-supervisor, Rui Pereira, for his assistance at every stage of this thesis. Whose knowledge in Green Computing was essential to developing the research questions and methodology of this dissertation. His feedback leads me to develop a better and more rigorous work, bringing my work to a higher level.

I would like to offer my special thanks to all the members of the Green Software Lab for the willingness to help with any issues I had during my master's thesis, especially Rui Calheno for the tremendous help provided.

Also, I am deeply grateful to my friends. Together, we made countless hours of stories, adventures, and memories. I want sincerely praise all the support and friendship during these years.

Last but not least, my appreciation also goes out to my family for their encouragement and support all through my studies. Especially to my mom, I'm grateful for her support and belief.

Abstract

In recent years, with the growth of energy consumption by computing devices, energy efficiency is a crucial concern in the IT area due to its economics and environmental impact. The recent but widespread use of powerful computing devices, namely smartphones, which rely on "the cloud" to store large amounts of information (like, for example, photos and videos), is demanding the construction and maintenance of large data centers. Such data centers run large-scale internet-based systems like cloud services. As a consequence, the energy consumed by data centers is growing fast, which is a crucial concern in the IT area due to its economics and environmental impact.

The growing reliance on cloud construction services is one of the main reasons for the rapid rise in research and development of energy efficient software and hardware for data centers. Nowadays, the most popular usage of data centers is the Database Management Systems (DBMS) that, normally, are responsible for the access, management, manipulation, and organization of data. While there have been advances and studies in energy-awareness in this area, there isn't enough knowledge on the energy efficiency provided by different database systems.

This master thesis intends to tackle this lack of knowledge by analyzing the energy consumption of DBMS software. Through benchmarks that simulate real usage environments, this research plays a key role in improving the knowledge on the energy efficiency of DBMS. We analyze four systems, namely MySQL, Postgres, MariaDB, and Redis. Moreover, we use the HammerDB benchmark framework for the simulation of DBMS in a real environment. Thus, to have a precise knowledge of the energy consumption of DBMS, we analyze the energy consumption in various subsystems of the computer, namely like CPU, DRAM, GPU, and Disk. Moreover, we present further analysis of the energy consumption per performance ratio in all subsystems levels.

Our results show that, indeed, there are significant differences in the energy consumption of which DBMS and that in some scenarios, the one with better run time performance is not what consumes more energy.

Keywords: Energy Efficiency, DBMS, Green Software, Green Computing, Program Analysis.

Resumo

Nos últimos anos, com o crescimento do consumo de energia pelos dispositivos computacionais, a eficiência energética é uma preocupação crucial na área de TI devido ao seu impacto económico e ambiental. Com a recente generalizada utilização de potentes dispositivos informáticos, nomeadamente smartphones, que dependem da "Cloud" para armazenar grandes quantidades de informação (como por exemplo, fotos e vídeos), está a exigir a construção e manutenção de grandes centros de dados. Esses centros de dados executam aplicações baseadas na Internet em grande escala, como serviços em nuvem. Como consequência, a energia consumida pela data centre está aumentar rapidamente, o que é uma preocupação crucial devido ao impacto económico e ambiental que estes trazem.

O aumento da dependência destes serviços em nuvem é uma das principais razões para o interesse em estudos e desenvolvimento de software e hardware com baixo consumo de energia. Hoje em dia, o uso mais popular dos data centre são os Sistemas de Gestão de Base de Dados (SGBD) que, normalmente, são responsáveis pelo acesso, gestão, manipulação e organização dos dados. Embora tenha havido alguns avanços e estudos em eficiência energética nesta área, ainda existe falta de conhecimento nesta área.

Esta dissertação pretende reduzir a falta de conhecimento do consumo de energia do software DBMS. Ao usar ferramentas de benchmarks que simulam ambientes reais, este estudo desempenha um papel fundamental no aprimoramento do conhecimento sobre a eficiência energética de diferentes tipos SGBD. Analisamos quatro sistemas, nomeadamente MySQL, Postgres, MariaDB e Redis. Além disso, usamos o framework de benchmark HammerDB para a simulação de SGBD em um ambiente real. Para ter um conhecimento aprefundado sobre o consumo de energia do SGBD, analisamos o consumo de energia em vários subsistemas do computador, nomeadamente como CPU, DRAM, GPU e Disco. Além disso, apresentamos uma análise mais aprofundada do consumo de energia relacionada com o desempenho em todos os níveis dos subsistemas. Esta tese apresenta resultados aonde pode ser verificado que existem diferenças significativas no consumo de energia das diferentes SGBD e em alguns cenarios, a Base de dados com melhor desempenho de performance de execução não é o que consome mais energia.

Palavras-chave: Green Software, Green Computing, SGDBS, Eficiência Energética.

Contents

List of Figures viii

List of Tables x

Listings xi

Acronyms xii

1 Introduction 1

1.1 Context and Motivation . . . 1
1.2 Research Questions . . . 3
1.3 Document Structure . . . 3

2 Literature Review 5

2.1 Database Management Systems . . . 5
2.1.1 Advantages of DBMS . . . 6
2.1.2 Relational Models . . . 7
2.1.3 Non Relational Data Models . . . 9
2.1.4 Benchmarking DBMS . . . 13
2.2 Green Software . . . 16
2.2.1 Related Work . . . 16
2.3 Energy Consumption . . . 17
2.3.1 Types of Energy Consumption . . . 18
2.3.2 Relevant Hardware Components . . . 19
2.3.3 Energy Measurement Frameworks . . . 20

3 Benchmark Architecture and Design 26

3.1 Energy Measure Method . . . 26
3.2 Databases Under Test . . . 28
3.2.1 MySQL . . . 28
3.2.2 MariaDB . . . 29
3.2.3 Postgres . . . 29

3.2.4 Redis 29
3.3 Design 30
3.4 Execution 31

4 Energy Efficiency of DBMS **34**
4.1 Benchmark Results 34
4.1.1 DBMS in Single User: Energy Consumption 34
4.1.2 DBMS in Multi User: Energy Consumption 39

5 Threats to Validity **43**

6 Conclusion and Future Work **46**
6.1 Final Considerations 46
6.2 Future Work 48

Bibliography **49**

Appendices **59**

A 10 minutes Benchmark Results **59**

B Other Relevant Graphs **64**

List of Figures

2.1 Visualization of Consistency, Availability and Partition Tolerance (CAP) theorem. 10
2.2 Example of a data model of a key-value store 11
2.3 Example of a JSON use in a data model of document store 12
2.4 Example of a simple social network in a graph data model. 13
2.5 HammerDB GUI. 15
2.6 Intel's Running Average Power Limit (RAPL) Power Domain. 20
2.7 Scheme of connections between Serial Advanced Technology Attachment (SATA) cable, current sensor and Arduino used to measurement secondary storage. 24

3.1 Benchmark Architecture and flow of events 31

4.1 Median of energy consumption on Package, Disk and Total. 35
4.2 Distribution of energy consumed 36
4.3 Median of HammerDB results. 37
4.4 Distribution of performance on HammerDB results 37
4.5 Median of energy consumption per TPM 38
4.6 Distribution of Energy consumption per TPM 39
4.7 Median of energy consumption per NOPM 39
4.8 Distribution of energy consumption per NOPM 40
4.9 Energy consumption with different number of users 41
4.10 HammerDB results with different number of users 41
4.11 Energy consumption per TPM with different number users 42
4.12 Energy consumption per NOPM with different number of users 42

A.1 Median of energy consumption on Package, Disk and Total. 59
A.2 Distribution of energy consumed 60
A.3 Median of HammerDB metrics. 61
A.4 Distribution of performance on HammerDB metrics 61
A.5 Median of energy consumption per TPM 62
A.6 Distribution of Energy consumption per TPM 62
A.7 Median of energy consumption per NOPM 63

A.8 Distribution of energy consumption per NOPM 63

B.1 MySQL energy behavior during a 5 minutes benchmark 64
B.2 MariaDB energy behavior during a 5 minutes benchmark 65
B.3 Redis energy behavior during a 5 minutes benchmark 65
B.4 Postgres energy behavior during a 5 minutes benchmark 66
B.5 MySQL energy behavior during a 30 minutes benchmark 66
B.6 Redis energy behavior during a 30 minutes benchmark 67
B.7 Postgres energy behavior during a 30 minutes benchmark 67

List of Tables

2.1 List of electrical properties, units, and symbols 18
2.2 RAPL power domains supported by different models 21

3.1 Physical server specifications 32
3.2 Specifications of CPU used 32
3.3 Specifications of Disk used 32
3.4 Software Configuration on physical server 33

6.1 Classification of each DBMS in each Scenario 47

Listings

2.1 Exemple of reading RAPL energy in C 22
2.2 Arduino source code for reading the analog signal from the current sensor 24

Acronyms

ACID	Atomicity, Consistency, Isolation, and Durability
BASE	Basically Available, Soft-state, eventually consistent
CAP	Consistency, Availability and Partition Tolerance
CODASYL	Conference on Data Systems Languages
CPU	Central Processing Unit
DBMS	Database Management System
DDL	Data Definition Language
DML	Data Manipulation Language
DRAM	Dynamic Random Acess Memory
EDT	Extending Database Technology
GPU	Graphics Processing Unit
HDD	Hard Drive Disk
IMS	Information Management System
IT	Information Technology
JCF	Java Collection Framework
MSR	Model Specific Registers
NOPM	New Orders Per Minute
NOSQL	Not Only SQL

OLTP	Online transactional processing
OS	Operating system
PB	Petabyte
RAM	Random Acess Memory
RAPL	Running Average Power Limit
SATA	Serial Advanced Technology Attachment
SIGMOD	Special Interest Group on Management of Data
SQL	Structured Query Language
SSD	Solid State Disk
SSDBM	Statistical and Scientific Database Management
TB	Terabyte
TPC-C	TPC Benchmark C
TPM	Transactions per Minute
TWH	Terawatt-hour
VLDB	Very Large Data Base Endowment Inc
VU	Virtual users

Chapter 1

Introduction

This chapter introduces the theme and objectives of this Master's thesis. First, Section 1.1 provides background on energy consumption, including why it is an issue in the Information Technology (IT) area and how it is a stimulus for this study. Afterwards in Section 1.2, the research questions of this thesis are presented along with an explanation of their reasoning. Finally, in Section 1.3, the remainder document structure is presented.

1.1 Context and Motivation

There has been an increase in the number of users with internet access over the past decades. In 2018, more than half of the world's population is already using the internet daily, and almost 60 percent of the world's homes have access to the internet [115]. With the unprecedented increase in users, the IT sector is eagerly more concerned with energy management in hardware and software development. These environmental issues caused by energy consumption are already evident. Whatever it urges the governments to track the IT capacity of companies around the world [11, 37, 48, 57, 124].

According to Mills, the energy consumption of the worldwide IT sector is approximately 1,500 TWH, corresponding to roughly 10 percent of the worldwide energy produced. The indicated values are equivalent to the global energy consumed by Japan and Germany together. Although there are policies to reduce data center energy consumption, the proportion of electricity cost still increases year by year in today's large-scale data center [8].

Due to this increase, data centers are becoming a vital part of IT operations that offer computing facilities to large, medium, and small organizations, such as online social networks, cloud computing providers, online companies, banks, hospitals, and universities. With the rise of cloud computing, hosting services in data centers has become a multi-billion-dollar sector that plays a pivotal role in the IT industry [37, 99]. Because of the environmental and economic implications of cloud services, new software and hardware

energy efficiency challenges for data centers have emerged [37].

Moreover, the energy costs of running a data center are exceeding the price of its hardware, which is prejudicial to its density, scalability, and associated environmental design [57, 66, 96]. It is essential to know that the energy consumed by a data center can be of two types: energy use by IT equipment (e.g., servers, networks, storage.) and usage by infrastructure facilities (e.g., cooling and power conditioning systems) [37, 93].

The amount of energy used by these two components changes with the established architecture. Some articles that analyze the energy consumption of the data centers have found that between 40% and 50% of the energy used in the data centers comes from cooling systems. While servers and storage devices consume about 26%, being the second most consumed in a data center [37, 55, 93, 117]. Additionally, the energy efficiency increase of servers is far below estimates [52, 66].

The energy efficiency of servers appears to have untapped potential. In general, database servers are the largest consumers of computing resources in data centers, making Database Management System (DBMS) one of the largest energy consumers. A particular usage of these systems is data systems warehousing. Data systems warehousing seeks to store the information of an organization, to facilitate the decision recovery processes that involve the decision-makers. These systems can integrate information from different sources, store historical and current data that can be a source of information that, when properly exploited, can guarantee relevant advantages in the market segments in the market segments where the companies fit. The accumulation of this historical information makes these systems elements with a high growth rate [50, 63, 64].

Since there are so many distributed database management systems available, choosing one can be difficult. While DBMS performance benchmarking is a supportive approach to deciding between different DBMS, due to the need to reduce the power consumption of database servers, that isn't the only factor nowadays [57, 106]. This urge has drawn attention from some well-known journals and conferences in the database field [57]. Some examples are the Journal of Network and Computer Applications [57], EDT [48], SIGMOD [128] , IEEE Data Engineering Bulletin [74], VLDB [85],and SSDBM [114].

With this surge, we face the challenge of not choosing only an energy-efficiency DBMS or performance DBMS but a performance-energy efficient one.

Choosing an energy-efficient DBMS comes with two main problems common to energy-efficient software development: the lack of knowledge on green software of software developers and the lack of tools to reason about software energy consumption [79, 91].

Consequently, software engineers tend to use existing software benchmark tools and (runtime) profilers to reason about the energy consumption of the software. The usual intuition is that faster software is also greener software. However, as several studies have shown, [32], time is not the only factor in the software's energy consumption, and slower software may be more energy-efficient than faster software. In the context of DBMS, software developers face another challenge: the lack of energy consumption knowledge in DBMS.

Thus, this thesis aims at reducing the lack of knowledge on the energy efficiency of the most popular DBMS, making it easier for developers and enterprises to choose the DBMS when they are concerned

about energy efficiency and energy proportionality. A significant aspect of our work that distinguishes it from previous works in this field is that our motivation is to explore the real impact of these systems in the most realistic environment.

1.2 Research Questions

As previously mentioned, this research aims at understanding: *Which DBMS software is the greenest running in a real-world environment, and what is the difference between them in terms of energy consumption?* Moreover, as DBMS relies on intensive disk operations, we also want to reason about disk energy consumption. Finally, we wish to study the energy impact of having multiple users performing real-world DBMS actions.

Thus, this thesis wishes to answer the following three research questions:

- **RQ1**:*Which Database Management System is the most energy-efficient?* In this research question, we want to understand which DBMS is the greenest in both Central Processing Unit (CPU) and disk energy consumption. Moreover, we would like to understand the DBMS energy efficiency at the CPU level and the disk level. This research question is vital because DBMS heavily relies on accessing external memory and CPU operations. Understanding this can help choose the greenest DBMS in different contests: For example, when we need to perform intensive and complex queries, which demand heavy CPU computations, which DBMS should we choose?

- **RQ2**:*Which Database Management System has the best energy versus runtime tradeoff?* With this research question, we desire to understand which DBMS has the better energy consumption per performance. For this, we want to know which one spent less energy per performance metric. This research question is necessary because choosing energy-friendly DBMS does not imply a DBMS with the worst performance, so by doing this research question. We want to understand the DBMS most suited for performance and low energy consumption.

- **RQ3**:*How does the increasing number of users of the Database Management System impact their energy consumption?* While some DBMS may be energy-efficient at the CPU level and others at the disk storage level, we want to understand the overall impact the user's traffic has on the different DBMS on the overall energy consumption. Here means that we want to comprehend how the scalability of DBMS affects energy consumption.

1.3 Document Structure

The remaining chapters of this thesis are into five parts. The following is a list of the chapters:

- **Chapter 2 - Literature Review:** Here, we present the literature review needed for this project. This chapter includes a description of the brief history of DBMS, its advantages, and different DBMS

models. Then an introduction to Green Software and related work in the area. This chapter ends with an explanation of energy consumption and its monitoring tools.

- **Chapter 3 - System Prototype:** This chapter details our study on a design and methodology level. Additionally, it explains the energy model used and DBMS studied.
- **Chapter 4 - Results:** Here is the chapter that shows and analyses the results obtained in this study. First, start by showing and explaining the graph visualization made and then analyzing these graphs.
- **Chapter 5 - Threats to Validity:** This chapter provides validation of this study. Here shows the degree to which evidence and theory support the interpretations of results.
- **Chapter 6 - Conclusion and Future Work:** The last chapter of this dissertation includes a conclusion to the research questions, final considerations, and future work proposals to give continuity and improvement to this study.

Chapter 2

Literature Review

This chapter presents the literature review and the state of the art of technologies involved in carrying out this project. First, Section 2.1 shows in detail DBMS: it starts by describing a brief history of DBMS, the advantages of using them, and the different DBMS models. Then how to benchmark DBMS and their benchmark systems. Next, in Section 2.2, we present green software focusing on the work related to this study. Finally, Section 2.3 explains the concept of energy, how energy can be measured, which are the most relevant subsystems in energy consumption, and the various solutions for monitoring the energy consumption of a system.

2.1 Database Management Systems

The Database Management System (DBMS) is the software designed to assist, maintain, and use a large set of data. It facilitates the definition, construction, manipulation, restriction, and sharing of data [47].

The first general-purpose DBMS was designed and developed by Charles Bachman in the early 1960s [59] and it was called The Integrated Data Store. It formed the basis for the data model of the network, which through the 1960s was standardized by the Conference on Conference on Data Systems Languages (CODASYL) and strongly influenced database systems [47, 59]. In the late 1960s, IBM developed the Information Management System (IMS) DBMS, used even today in many major installations. IMS formed the basis for an alternative data representation framework called the hierarchical data model [47].

DBMS soon entered all sectors of organizations allowing the implementation of comprehensive and indispensable information systems. Historically they have evolved from the hierarchical model, which had the problem of the relationship between the entities represented, for the Relational Model that solved the relationship problem between the entities represented, allowing great flexibility navigation and search of stored data [47].

Database management continues to gain importance as more and more data is brought online and

made ever more accessible through computer networking. The global DBMS market reached an estimated value of almost 58.4 billion American dollars in 2019. The global DBMS industry is further expected to grow at a CAGR of 13.81% between 2020 and 2025 to reach a value of almost 126.9 billion American dollars by 2025. We can't deny the importance of the DBMS on the world. mention the advantages of why we should be using an instead of other options.

The DBMS is the software that interacts with the user's application programs and the database. Typically, a DBMS provides the following facilities [27]:

- It allows users to define the database, through a Data Definition Language (DDL). The DDL allows users to specify the data types and structures and the constraints on the data to be stored in the database.
- It allows users to insert, update, delete, and retrieve data from the database, usually through a Data Manipulation Language (DML). Having a central repository for all data and data descriptions allows the DML to provide a general inquiry facility to this data, called a query language.
- It provides controlled access to the database like a security system, which prevents unauthorized users, an integrity system, which maintains the consistency of stored data, etc.

2.1.1 Advantages of DBMS

In Gehrke book [47], he presents the main advantages of DBMS:

- **Data Independence:** Ideally, application programs should not be exposed to data representation and storage information, and the DBMS offers an abstract view of the data that hides those details.
- **Efficient Data Access:** A DBMS utilizes a range of advanced techniques to efficiently store and retrieve information.this function is particularly important, If the data is stored on external storage devices.
- **Data Integrity and Security:** A DBMS to maintain data integrity implement restrictions on data access. Also, it can enforce access control that govern what data is visible to different classes of users.
- **Data Administration:** If many users share the data, major changes can be made by centralizing data administration. Experienced professionals who understand the complexities of the data being handled, and how it is used by various groups of users, may be responsible for arranging data representation to reduce duplication and fine-tuning data storage to make recovery efficient.
- **Concurrent Access and Crash Recovery:** A DBMS schedules simultaneous data access in such a way that users can think of the data as being accessed by just one user at a time. In addition, the DBMS protects users against the impact of device failures.

- **Reduced Application Development Time:** DBMS already support several common functions between different applications and also with its high level interface. This facilitates rapid application creation in accordance with the high-level interface to the data. DBMS applications are therefore likely to be more stable than equivalent stand-alone applications since the DBMS performs several significant tasks.

Even with these advantages, it can bring some disadvantages:

- **Danger of overkill:** For small and simple applications the use of database system is often not advisable and can have a negative impact on the overall performance.
- **Complexity:** Further complexity and requirements are created by a database system. This can be quite costly and demanding to supply and operate a database management system with multiple users and databases.
- **Costs:** With a use of database system it will create a new costs for the system itself, but also for additional hardware and the more complex handling of the system.
- **Higher impact of a failure:** The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the availability of the DBMS, the failure of certain components can bring operations to a halt [27].
- **Cost of conversion:** In some situations, the cost of the DBMS and extra hardware may be insignificant compared with the cost of converting existing applications to run on the new DBMS and hardware [27].

Every DBMS as is own way to store data but every one is define by a data model. A data model is a collection of high-level data description constructs that hide many low-level storage details. A DBMS allows a user to define the data to be stored in terms of a data model.

2.1.2 Relational Models

The relational data model is based on relational algebra and was proposed by Codd [24]. The Relational Model made a revolution in the way that users used it as databases, which was just the manipulation of physical structures. Codd believe that adopting his vision would allow users to use a higher-level language and abstraction, not depending to specify the physical representation of data and that improved the productivity for database users [26].

Relational Databases have specific characteristics like the structural aspect, Support for a language at least as powerful as relational algebra, and rules to manipulate data [25].

The relational models are more know as Structured Query Language (SQL) databases, which Structured Query Language (SQL) is the query and maintenance language used in these applications.

A relational database is in simple terms, a set of tables. Each table contains data on aspects of one subject, such as a client, an order, a product, a team, or a city. Restrictions on the data in individual tables and even between tables may be set. It is possible to usefully represent how the tables are interrelated with a data model. Different attributes of the same table may or may not have the same domain [23].

This model has a lot of cautions and one of these cautions is that we have to normalize our data. Normalization is a technique for producing a set of relations with desirable properties, given the data requirements of an enterprise. Normalization is a formal method that can be used to identify relations based on their keys and the functional dependencies among their attributes [27].

Other rules this model have is in their transaction, In this context, a transaction is an operation, or a chain of operations, carried out by a single user or application program that accesses or modifies the database content. A transaction is a logical work unit that brings the database from one state of consistency to another. Transactions can be successfully terminated [27]. A transaction should follow the four basic properties Atomicity, Consistency, Isolation, and Durability (ACID) as they know which stand for the first letter of the proprieties [58]:

- **Atomicity**: The transaction operations are all done or none are. It is an all or none rule.
- **Consistency**: A transaction must transform the database from one consistent state to another consistent state and any of the states the database can be not consistent.
- **Isolation**: Different Transaction are independent one of other.
- **Durability**: After a transaction, the effect of a transaction have permanently recorded on the database and must survive a system failure.

Nowadays, relational DBMS are the most popular in the DBMS market following the ranking made and study by DB-Ranking [38] the five market leaders in Relational DBMS are:

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL
- IBM Db2

Even though that IBM developed one of the first DBMS, IBM struggle to be the most popular DBMS because of the competitiveness of this market. However, with the rise in internet traffic in recent years, the volume of accumulated data is increasingly increasing to a scale of 1,000 Terabyte (TB) and even 100 Petabyte (PB). The scalability of SQL storage is thus being challenged [126].

To scale the relational DBMS it is required to replace existing hardware with more efficient ones, which come at a high cost. Besides, if this improvement is not enough to accommodate all data volumes, the alternative is to spread the device through several servers. Relational databases have a hard time adapting and scale to distributed environments because it is not easy to put different tables across different servers since relational databases are built to manage data on the same server rather than partitioning data. Additionally, it may bring additional complexity to the data model when dealing with a large quantity of data that does not easily fit into a table, which leads to a decline in the efficiency of reads and writes. Also, the query language is a problem, as it can only deal with standardized information. Moreover, conventional SQL solutions don't work well with agile development, which requires large quantities of complicated code [60, 126].

NOSQL solutions have become a hot subject and alternative to SQL databases to solve these scalability and performance problems [60, 126].

2.1.3 Non Relational Data Models

The early concept of Not Only SQL (NOSQL) was first used in 1998 by Carlo Strozz [110] that did not expose the SQL interface, though it was based on a lightweight relational mode [70, 120, 126]. However, Strozzi used the term simply to distinguish his solution from other relational DBMS [70].

Nowadays, the term NOSQL start to regain popularity with the rise of the era of Big Data [70]. Big data was defined by Apache Hadoop in 2010 as "datasets which could not be captured, managed, and processed by general computers within an acceptable scope." Big data is linked to Internet companies' services and that is rising rapidly nowadays. For example, Google processes data from hundreds of PB, Facebook produces log data of more than 10 PB per month, or Baidu, a Chinese business, processes data from tens of PBs [20]. Even though traditional SQL databases have proven to be highly effective, secure, and consistent in terms of structured (or relational) data storage and processing, they fall short of Big Data processing, which is characterized, among other things, by the large volume, variety, velocity, openness, lack of structure and high visualization demands [17, 70].

With this information, the new definition of NOSQL merges as Not Only SQL. This term means that any DBMS that doesn't follow all proprieties and principles of the relational models [70, 120, 126].

NOSQL systems generally have six key features [18]:

- the ability to horizontally scale "simple operation" throughput over many servers.
- the ability to partition and replicate data over many servers.
- a simple call level interface or protocol.
- a weaker concurrency model than the ACID transactions of most relational SQL database system.
- efficient use of distributed indexes and Random Acess Memory (RAM) for data storage.
- the ability to dynamically add new attributes to data records.

A key feature of NOSQL systems is replicating and partitioning data over many servers. This allows them to support a large number of simple read/write operations per second. This simple operation load is traditionally called Online transactional processing (OLTP), but it is also common in modern web applications [18].

To guarantee the integrity data in all levels of data management, SQL implement ACID proprieties on them transitions. However, these proprieties have poor scaling. With the face of these problems, the CAP theorem arises on NOSQL databases. CAP was designed by Brewer[14] stands for Consistency, Availability and Partition Tolerance, and this theorem consist in 3 different aspects [49, 83]:

- **Strong Consistency:** All clients see the same version of the data, even on updates to the dataset.
- **Availability:** All clients can always find at least one copy of the requested data, even if some of the machines in a cluster are down.
- **Partition-tolerance:** The total system keeps its characteristic even when being deployed on different servers, transparent to the client.

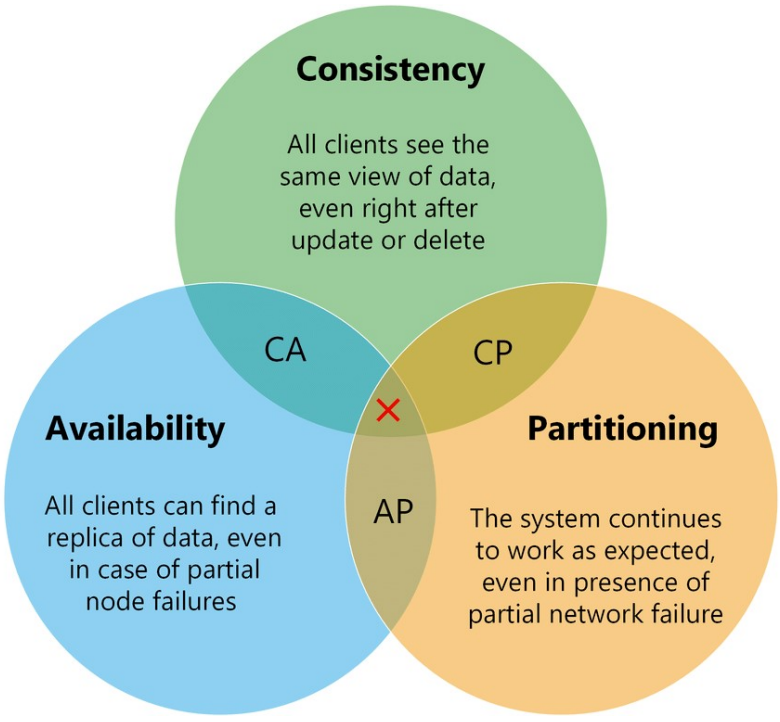


Figure 2.1: Visualization of CAP theorem.

The CAP Theorem proposes that only two of the three different aspects of scaling out can be achieved fully at the same time, in Figure 2.1 we can send the 3 states that can be archive. Still systems can be improved towards the maximum proximity of the three characteristics, this state are [13, 14, 18, 58]:

- **CA:** databases can only be consistency and availability at the same time if they don't have data into multiple server peers.

- **CP:** when working with a partitioned database, it is only possible to achieve consistency by giving up of temporary availability to have time for the partitions recovered to avoid inconsistency.
- **AP:** when working with partitioned data across multiple servers it is only possible to achieve total availability when giving up consistency at all times.

Many of the NOSQL databases above all have loosened up the requirements on Consistency to achieve better Availability and Partitioning. This resulted in systems know as Basically Available, Soft-state, eventually consistent (BASE) [18, 94, 116]:

- **Basic availability:** Each request is guaranteed a response successful or failed execution
- **Soft state:** The state of the system may change over time, at times without any input.
- **Eventual consistency:** The database may be momentarily inconsistent but will be consistent eventually.

There is been a lot of approaches and models on NOSQL. This led to creating a bunch of new categories for the NOSQL data models. The most popular are Key-value stores, Document stores, Graph databases, and Wide Column Stores [70, 75, 76].

With more detail, it will explain the above-mentioned NOSQL models:

Key-value stores The Key-value stores is the most simple model as the name suggest, this database only stores data as pair key and values, where stored values can be retrieved with the respective key. With this simplicity means that does any complex structured because data is organized as an array of entries but these simple systems are normally not adequate for complex applications. These models are popular due to their simplicity, stability and efficiency, as they have, in general, linear access to the database data and usually, these models are used when you want a good cache management [70, 82, 113, 116, 126].

In Figure 2.2 shows an example of this model. If for the sets a hash map or an associative array is used as a data structure, information can be retrieved in constant time.

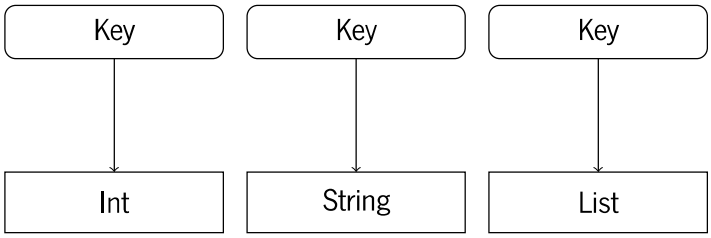


Figure 2.2: Example of a data model of a key-value store

It is possible to classify key-value stores into various groups. If they store data in the cache or on disk, store the keys sorted or are eventually consistent, they can be separate [82].

The drawbacks are that all key-value stores share, they only support basic key/value data structures, joins are not supported and among the various implementations there is no specific query language [82].

Document stores Data stores, also known as document-oriented DBMS. This model is a type of database that stores uniform fields with a non-standard amount of information for each record and is distinguished by its schema-free data organization. There is no need for records or documents to have a standardized layout, and different records which have different columns. For each record, the types of individual column values can be different, and columns can have more than one value. This is possible by encapsulating documents such as JSON, XML, BSON or similar metadata technologies in a particular document type in order to compose nested semi-structured content [70, 82, 113, 119, 126].

These stores organize the records into collections as a way of understanding what each document relates to. In terms of their own schema, any documents associated may be included in these collections. When dealing with data requests from a particular array, this allows to retrieve processes. A few different open-source document databases are available today but the most prominent among the available options are MongoDB and CouchDB [70, 113].

In the Figure 2.3 of a structure of a documents on a document store database.


```
{
"EmployeeID": "SM1",
"FirstName": "Anuj",
"LastName": "Sharma",
"Age": 45,
"Salary": 10000000
}
```


(a) Document of employee a


```
{
"EmployeeID": "MM2",
"FirstName": "Anand",
"Age": 34,
"Salary": 5000000,
"Address":  "Line1": "123, 4th Street", "City":
"Bangalore", "State": "Karnataka",} "Projects": [
"nosql-migration", "top-secret-007"]
}
```


(b) Document of employee b

Figure 2.3: Example of a JSON use in a data model of document store

Graph stores Graph stores are also known as graph-oriented or graph-oriented DBMS. Databases are therefore distinct from specialized data management tools that in their implementation, use graph notions to represent data as nodes and edges in graph structures that represent relationships between nodes. They make it easy to process the data in that form and easy to calculate specific graph properties, such as the number of steps required to get from one node to another node. Graph databases allow us to enforce specifications for graph processing at the same level of query language as we use for graph data fetching without the additional abstraction layer for graph nodes and edges. This implies less overhead and more versatility and performance for graph processing [5, 6, 70].

Figure 2.4 depicts an example of a simple social network in a graph database.

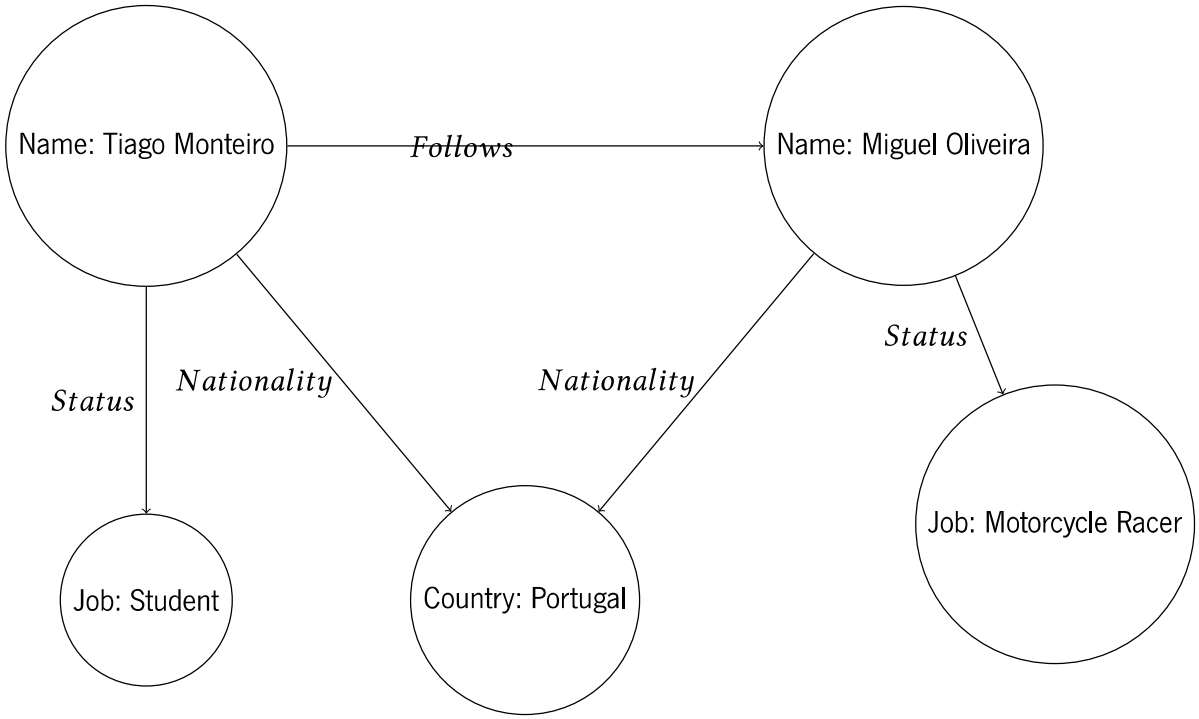


Figure 2.4: Example of a simple social network in a graph data model.

Wide Column stores The first column stores appeared in 1969 [2]. In 1975 was developed the second column store and the first one with application in healthcare, for store clinical records [123].

Wide column stores store data in records with the capacity to carry very large numbers of complex columns, also called extensible record stores. A column-based record solution works well to act as write-optimized operations for these types of semi-structured data and it is not the case in the conventional relational DBMS that it is not optimized to write data to a smaller subset of records in order to update the records, it must read the whole set of tables. [70, 126].

2.1.4 Benchmarking DBMS

One of the harder aspects of data management is the performance analysis and tuning. For complex systems that perform high workloads on large datasets, it is particularly challenging because many factors can affect the performance. In terms of DBMS, this is done by the usage of a benchmark that allows one to evaluate the system's main performance metrics under stressful conditions [41].

Benchmarks are techniques that seek to collect and compare a wide variety of activities, to achieve the best result, an objective criterion for determining which practice or software is superior in certain scenarios that the user who is doing the simulation built. An example of popular questions is "Which domain is the best system?". The SPECCpu benchmark [62], for instance, addresses the question, "What is the best CPU?""and the TPC Benchmark C (TPC-C) [29] responds to the question, "What is the best OLTP database system?" [21, 46].

For Gray, a domain-specific benchmark must meet four criteria to be an effective benchmark. It must be:

- **Relevant:** In performing typical operations within that problem domain, it must measure the peak performance and price/performance of systems.
- **Portable:** It should be easy to implement the benchmark on many different systems and architectures.
- **Scaleable:** The benchmark should apply to computer systems small and large. As computer performance and architecture evolve, it should be possible to scale the benchmark up to bigger systems and to parallel computer systems.
- **Simple:** The benchmark must be understandable, otherwise credibility will be lacking.

2.1.4.1 TPC Benchmark C

TPC Benchmark C (TPC-C) is an OLTP workload [29]. It is a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:

- The simultaneous execution of multiple transaction types that span a breadth of complexity
- On-line and deferred transaction execution modes
- Multiple on-line terminal sessions
- Moderate system and application execution time
- Significant disk input/output
- Transaction integrity (ACID properties)
- Non-uniform distribution of data access through primary and secondary keys
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Contention on data access and update

While these specifications express implementation in terms of the relational data model with a traditional locking framework, any commercially available DBMS, database servers, file systems, or other data repositories offering a functionally equivalent implementation can be used to implement the database.

TPC-C uses metrics and terminology that are similar to other benchmarks originating from the TPC or others. In no way does this similarity in terminology imply that the results of TPC-C are comparable to other benchmarks. Other TPC-C results compliant with the same revision are the only benchmark results comparable to TPC-C.

The performance metric reported by TPC-C is a "business throughput"measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions per minute.

TPC-C is accepted in the industry as the most credible transaction processing benchmark with a large body of results across all major hardware and database platforms. The highly tuned and optimized nature of the TPC-C configurations makes it the best candidate for study DBMS power consumption [92].

2.1.4.2 HammerDB

HammerDB is an open source and widely used benchmark framework for the worlds most popular DBMS [4, 21, 44, 71, 72, 104, 130]. HammerDB emulates a TPC-C scenario and through OLTP workloads it sets up a company's sales processing environment. It reduces the testing costs by simplifying the TPC-C rules, which can be modified and run on a custom environment. The above factors result in a low-cost solution, rapid deployment, and customized DBMS benchmark system [21, 44, 107]. HammerDB currently supports Oracle, SQL Server, Db2, TimesTen, MySQL, MariaDB, PostgreSQL, Greenplum, Postgres Plus Advanced Server, and Redis. Moreover, it can run on a variety of Operating system (OS), making it a flexible and heterogeneous benchmarking framework [21].

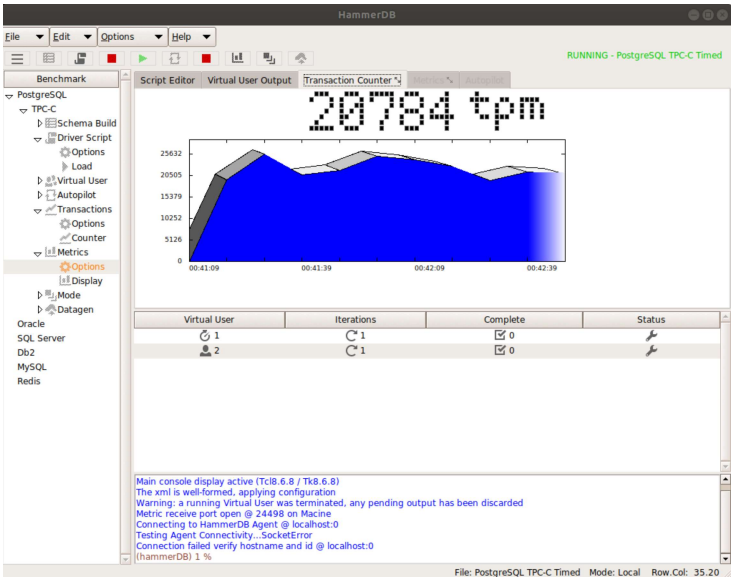


Figure 2.5: HammerDB GUI.

Although HammerDB implements a workload based on the TPC-C specification, it does not implement a complete TPC-C benchmark specification. As a consequence, the transaction results from HammerDB can not be compared to the official TPC-C benchmarks. HammerDB workloads generate 2 statistics. Transactions per Minute (TPM) is the transactional measurement of the specific database typically defined as the number of user commits plus the number of user rollbacks. TPM values are database-specific and, thus, they cannot be compared among different DBMS. The New Orders Per Minute (NOPM) value, on the

other hand, is a performance metric independent of any particular database implementation and it is the recommended primary metric to use [107].

HammerDB is being actively used by researchers to study the performance of DBMS: Elgrably use HammerDB to compare open-source DBMS performance like Mysql, MariaDB and Postgres [44]. Another study that uses Hammerdb is Knoche (2016) work that uses HammerDB to test the impact of database lock contention in the TPC-C benchmark scenario [71]. In the context of green computing, Koçak (2018) uses HammerDB with MySQL to define a dataset to use on his software energy consumption prediction [72].

HammerDB is also being widely used by most leading database and technology companies, such as Oracle, IBM, Intel, Dell/EMC HPE, Huawei, Lenovo, and hundreds more [107]. It has been downloaded hundreds of thousands of times in more than 180 countries.

2.2 Green Software

Since the exploding of the IT in all areas of our activity, offering great benefits, convenience, opportunities and irreversibly transforming businesses and society, it has also been contributing to environmental problems [84].

With an increase of concerns with energy consumption in all areas, it grows the use of Green computing, also known as Green IT in the computer science area. Green computing is the area that study and practice environmentally friendly and sustainable computing. The goals of this area are to reduce and understand the energy consumption of different technologies, Software, or hardware and which choice can we make to reduce energy consumption. Green Software is a sub-area of green computing that mains goals are to reduce energy consumption through software analysis and optimization. Therefore, developing green software can contribute significantly to preserve the environment and reduce energy consumption globally.

2.2.1 Related Work

Even though green software isn't as popular as it should be a wide range of studies have already been made towards introducing and creating more energy-friendly and energy-aware approaches. At the moment, the work done on energy consumption is clear proof of the paradigm shift in the creation of software. There is a lot of examples of this shift on energy awareness on software development, examples of this is the work made by Pereira et al. (2016), that analyze the energy consumption of the different Java Collection Framework (JCF), present an energy optimization approach for Java programs: based on calls to JCF methods in the source code of a program and define a green ranking for Java Collections. In Couto et al. (2017), authors define a ranking of energy efficiency in the programming language in ten well-known programming languages by running a set of computing problems in each language and monetize the energy consumption.In the area of programming languages, there are other examples of studies [15, 77, 86–89]. Other studies can be found in different areas like the mobile area, a study developed by Couto et al. (2014) aimed at detecting Anomalous Energy Consumption in Android Applications. Rua et al. (2019)

developed the GreenSource infrastructure: a large body of open-source Android applications tailored for energy analysis and optimization. There are a lot more examples in the mobile area [30, 33–35, 80, 100, 102].

DBMS doesn't escape the concerns and care of the green movement, Agrawal et al. (2008) made an early approach concerning energy consumption on database systems and his report, he said people should take into consideration designing power-aware DBMS that limit energy costs without sacrificing scalability. Harizopoulos et al. (2009) focuses on finding software-level optimization characteristics that might improve the energy efficiency of Data Management Systems. In work provide by Wang et al. (2011), he presents a survey about energy efficiency in data management operations.

Although most of the study's show previously focus more on the hardware base premises. There is other more focus on software like, Xu et al. (2010) presented a solution of power-performance tradeoffs on DBMS where are results show that exists attractive tradeoffs between average-power and time-efficiency. Xu et al. (2012) also proposes query optimization intending to reduce energy consumption. Additionally, To reduce the peak of energy usage in database management systems Kunjir et al. (2012) proposed several alternatives. In Rodriguez-Martinez et al. (2011)'s paper, he presents an empirical methodology to estimate the power and energy cost of database operations, on a similar context Rodríguez et al. (2013) developed a work related to the prediction of the energy consumption of join queries. Later, Gonçalves et al. (2014) redesigned the DBMS execution plan to include both the average energy consumption value for the most common database operators and the total query energy estimation. Afterward, these authors made similar work on a different domain about measure energy consumption for green star-queries in data warehousing systems [10]. After this, as a simple guideline for reducing the energy consumption of a given query within a relational DBMS, [56] devised a collection of heuristics. There are studies for NOSQL, one example of that are Duarte and Belo (2017) work that focuses on energy consumption on document stores based systems. Saraiva et al. (2017) developed a work about query energy consumption comparing the energy efficiency between a relational and a non-relational DBMS. Another study related to the previous one is Mahajan (2016) work that compares energy consumption between different DBMS like MySQL, MongoDB, and Cassandra on a query level.

In this work, we extend our experience in energy consumption evaluation on different databases. Where, unlike the other studies show we want to evaluate and compare the energy efficiency between different DBMS in a realistic environment.

2.3 Energy Consumption

This Section presents the definition of energy, what are the relevant hardware components on energy consumption, which methods exist to measure energy consumption, and more detailed insight into the tools used in this work.

2.3.1 Types of Energy Consumption

In order to understand how energy consumption can be measured, we must first understand what power is, what energy is and how different it is.

The scientific definition of Power is the total workload or energy transferred divided by the time interval. In this context, power is the amount of energy consumed per unit of time. According to The international system of units [111], the universal unit used for power is Watts or Joules/s. Energy is the total workload of a system during a period of time and is the unit used for Energy is Joules.

The Power can be calculated in two ways, the first is when we know the total energy and the time interval,

$$P = \frac{E}{\Delta t} \tag{2.1}$$

The other possible way of calculating is through Voltage and current intensity. Voltage is defined to be the potential energy of two different points and the Unit is Volt and Current intensity is the magnitude of an electric current as measured by the quantity of electricity crossing a specified area of equipotential surface per unit time and unit of measurement is Amperes.

With this we can calculate the power by this formula:

$$P = VI \tag{2.2}$$

With this two formulas we can deduce this formula to calculate the energy,

$$E = VI\Delta t \tag{2.3}$$

	Unit	Symbol
Voltage	Volt (V)	V
Current	Amperes (A)	I
Energy	Joules (J)	E
Power	Watts (W)	P

Table 2.1: List of electrical properties, units, and symbols

To measure the energy consumption of total and individual components, we have to distinguish different types of consumption. These types of energy consumption are:

- **Idle Consumption:** is the base consumption needed to ensure that the system is ready, and able to respond to any user need quickly and effectively. This energy is consumed by the system regardless of the state of operation. This includes energy consumed inactive by the disks, network interfaces, CPU, caches, memory, motherboard, etc. This category also includes the energy needed to maintain the basic requirements of the OS and other tasks in progress [93].

- **Dynamic Consumption:** This refers to the basic consumption necessary for an operation requested by the user of the system. Dynamic Energy includes the energy consumed by the CPU, the extraordinary operations on the Dynamic Random Acess Memory (DRAM), the disk when searching, reading or/and writing data, the network interface when receiving and transmitting data packets, and other needs to respond to user requests when executing instructions. Dynamic power consumption values mainly depend on the type of workload that runs on the machine. [93].

The methods available to measures these types of consumptions are [7]:

- **Instant Power Measurement:** The instantaneous current consumed by the unit is measured and then multiplied by the voltage, and the result is the power done. With power and time, it is possible to get energy consumption over that period. If the sampling frequency is high, instant power measurements are reliable, but his drawback is that physical instrumentation is needed. This method generally works at the level of the system, although measuring the component-level of hardware is possible.

- **Time Measurement:** Another way to collect the energy consumption of a device is through measurement of time. Assuming a constant consumption over time, the speed at which energy is depleted depends on the power consumption of the device.

- **Model Estimation:** Energy consumption calculations in this method are measured in a way where a power consumption of a specific computer is connected to internal resource use indicators, such as CPU states, commands, memory or disk access, and network adapters. This method utilizes machine calls to estimate the usage of resources.

2.3.2 Relevant Hardware Components

It is important to distinguish which hardware components are relevant to global consumption before deciding on a method for measuring energy consumption because it is easy to understand and compare the behavior of the general system with the energy consumption of the subsystem. While global consumption can also be constant and unchanging, that doesn't mean that there is a steady consumption of energy in all subsystems.

According to several studies carried out during the last 15 years, it shows that the main subsystems that generate dynamic energy consumption are the CPU, the Memory, and Disk [93]. According to google research [9] provided some insight into how energy is used in modern IT equipment at the time and got by breaking down the peak power usage of one generation of WSCs deployed at Google in 2007 categorized by main component group, they got that using modern data center using late 2012 generation servers the 3 main hardware components besides cooling components on energy total consumption was CPU with 42%, Disks with 14,3% and DRAM with 12,3%. Also, Kansal et al. mentions in an article that the subsystems

with more impact on the dynamic consumption are CPU, memory, and disk, represent, respectively, 58%, 28%, and 14% .

With this information, we can conclude that our energy measurements must be around this 3 components and it also facilitates the study of the causes of energy variations in the DBMS.

2.3.3 Energy Measurement Frameworks

2.3.3.1 RAPL

Intel's RAPL is a tool to measure the energy on CPU and primary storage (DRAM). Here we will explain how it works, and the limitations it has.

RAPL is a well-known and accepted interface for measuring the power consumption of a computer system. Various studies used this interface as a measuring tool, and others review Intel RAPL measurements in terms of accuracy, performance, granularity, usability [39, 69].

Intel introduced RAPL on their compilers in the architecture of the Intel Sandy Bridge, and since the first appearance, it has been evolving in the newer architectures. RAPL has Model Specific Registers (MSR) that are not part of the architecture of the processor but address power values required for energy consumption management [65, 69, 93].

The main functionalities of RAPL are to measuring the energy consumption on the CPU and primary storage and also to limit the energy consumption on the components mentions. In the context of this research, we won't use the last functionality, as it does not fit in the context of this study [39, 65, 69].

The RAPL does not measure energy based on an analog energy meter because energy consumption is estimated through the analysis of various hardware performance counters, temperature sensors, leakage energy, and IO models. Registers reserved for energy readings are updated approximately every millisecond (1kHz) [93, 122].

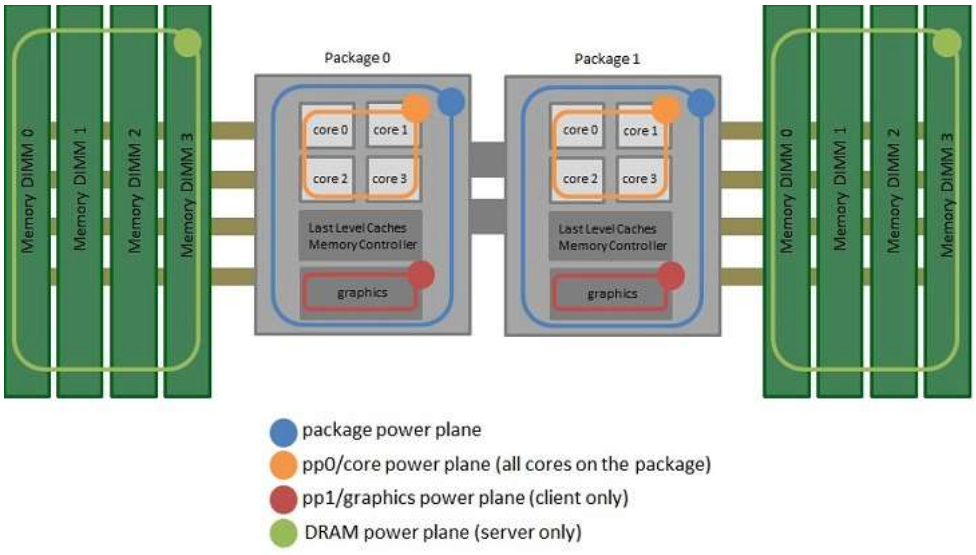


Figure 2.6: Intel's RAPL Power Domain.

We can check the different power domains supported by RAPL in Figure 2.6. Each power domain has a different MSR and reports the domain's energy consumption, allowing us to limit that domain's power usage over a specified time. Each domain represents distinct physical component sets, currently, these are:

- **Package**: the package domain refers to the package's energy consumption of the entire socket, including the core and uncore components energy [65].
- **PP0/Core Power Plane 0**: This domain measurement the energy consumption of all processors socket [65, 93].
- **PP1/Graphics Power Plane 1**: This domain measures the energy consumed by all processors in a Graphics Processing Unit (GPU).
- **DRAM**: This Domain refers to the energy consumption of RAM [65, 93].
- **Psys**: Intel *Skylake* has introduced a new RAPL Domain named PSys. It monitors and controls the thermal and power specifications of the entire SoC and it is particularly useful when the source of power consumption is neither the CPU nor the GPU [69].

For multi-socket server systems, each socket reports its own RAPL values [69].

There are some distinctions between the list of available domains, depending on the type of platform. The available domains on platforms intended for the client are Package, Power Plane (PP) 0, and PP1. On the other hand, the Package, PP0, and DRAM domains are available on the Platform intended for Servers [39]. In the Table 2.2 presents an overview of RAPL domains supported by different processor model.

Model	Power domain supported				
	PKG	PPO	PP1	DRAM	PSYs
Sandy Bridge	YES	YES	YES	NO	NO
Sandy Bridge-EP	YES	YES	NO	YES	NO
Haswell	YES	YES	YES	YES	NO
Haswell-EP	YES	NO	NO	YES	NO
Skylake	YES	YES	YES	YES	YES*

*Not All Skylake versions support PSys

Table 2.2: RAPL power domains supported by different models

The MSR interfaces available in each of the domains mentioned are the following:

- **Power Limit**: interface serves to specify the time interval and the limit of energy to be consumed [39, 93].
- **Energy Status**: This interface provides the energy consumed. The register reports the actual power used by the domain. This interface is read-only.

- **Perf Status**: This interface is optional and provides the effects of restrictions used [65, 93].
- **Power Info**: This interface is optional and presents the information for a given domain (Power, Energy, etc.) [93].
- **Policy**: This Interface is optional and it allows defining control policies for energy to distribute costs, that is, to balance the power consumed between domains subdomains [39].

With all this information we can conclude that the largest domain we can get on RAPL is with this equation:

$$E_{PPO} + E_{PP1} <= E_{Packgage} \tag{2.4}$$

To gather results, we develop software in C, that is running in parallel with the task we want to measure. On the listing 2.1 , is a example of the code we made.

Listing 2.1: Exemple of reading RAPL energy in C

```
void rapl_after(FILE * fp , int core){
  int fd;
  long long result;
  fd=open_msr(core);

  result=read_msr(fd,MSR_PKG_ENERGY_STATUS);
  package_after=(double)result*energy_units;
  fprintf(fp,"%.6f , ",package_after-package_before); // PACKAGE

  result=read_msr(fd,MSR_PP0_ENERGY_STATUS);
  pp0_after=(double)result*energy_units;
  fprintf(fp,"%.6f , ",pp0_after-pp0_before); //CORE

  if ((cpu_model==CPU_SANDYBRIDGE) || (cpu_model==CPU_IVYBRIDGE) ||
     (cpu_model==CPU_HASWELL)) {
      result=read_msr(fd,MSR_PP1_ENERGY_STATUS);
      pp1_after=(double)result*energy_units;
      fprintf(fp,"%.6f , ",pp1_after-pp1_before); // GPU
  }
  else fprintf(fp," , ");

  if ((cpu_model==CPU_SANDYBRIDGE_EP) || (cpu_model==CPU_IVYBRIDGE_EP) ||
     (cpu_model==CPU_HASWELL)) {
```

```
        result=read_msr(fd,MSR_DRAM_ENERGY_STATUS);                                    24
        dram_after=(double)result*energy_units;                                        25
        fprintf(fp,"%.6f , ",dram_after-dram_before); // DRAM                          26
}                                                                                      27
else fprintf(fp," , "); }                                                              28
```

The MSR driver must be enabled for direct MSR access with */dev/cpu/*/msr* command, and read access permission must be set for the driver. Reading RAPL domain values directly from MSR requires the CPU model to be detected and the RAPL energy units read before reading the consumption values of the RAPL domains [69, 122]. Once the CPU model is detected, the RAPL domains can be read per package of the CPU by reading the corresponding "MSR status" register [65, 69].

2.3.3.2 Arduino

Here it will show why Arduino was chosen to measure energy consumption on disk storage, which method he uses, and how he works.

In this case, for measured disk consumption we opt for an instant power measurement approach made by Portela instead of a Model Estimation. An Instant power measurement approach is a reliable choice because it is one of the simple solutions to achieve the objectives intended, and it is low-cost. Here we want to ensure that our results are precise and reliable, and since our measurements are on a small scale, it doesn't have an impact on the overall system. This solution provides what is needed without significant drawbacks [93].

Since it is necessary for data storage and handling of energy consumption, a mere ammeter won't do the work here. Thus to gather the measurements it was chosen an Arduino Uno with a current sensory. Figure 2.7 presents the scheme of an Arduino UNO with a current sensory connected to a SATA cable. The SATA cable is responsible for distributing energy to the disk and data exchange between the secondary storage and the computer. So to get the energy consumption on the disk, the current sensor must be connected to the SATA cable that is responsible for distributing energy [93].

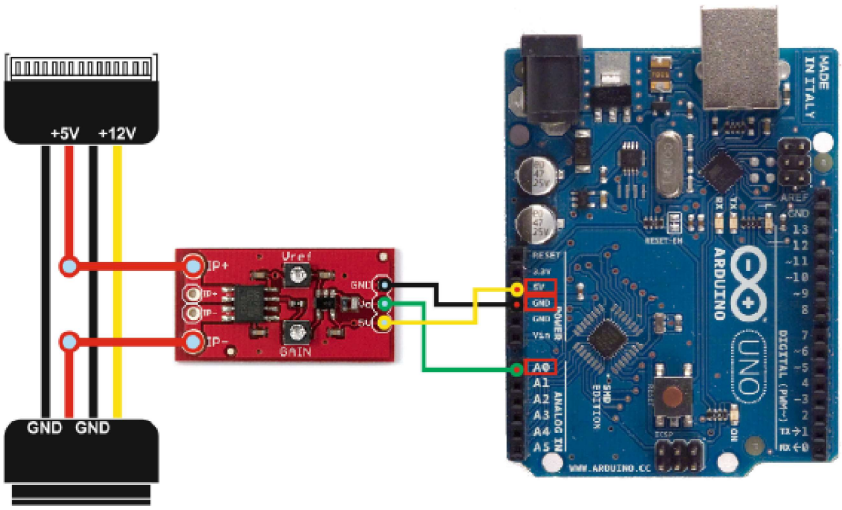


Figure 2.7: Scheme of connections between SATA cable, current sensor and Arduino used to measurement secondary storage.

The current sensor used was the Low Current Sensor Breakout. This current sensor is made by SparkFun and can measure the current regardless of whether the signal is continuous or alternating. This sensor has an operational amplifier phase to control the gain, which can measure lower currents more precisely [93].

Furthermore, Arduino UNO must be able to read the analog voltage presented by the sensor and communicate these values to the computer connected to it through the serial port. For this, a program was made that would get take constant readings at the analog voltage and send them every 0.1 seconds. An example of this code is on the Listing 2.2 where the reading is made every millisecond and then is made the average of the last one hundred milliseconds, which send this information to the computer system connected to the Arduino.

Listing 2.2: Arduino source code for reading the analog signal from the current sensor

```
void loop() {
  /* Initialization */
  float average_a0 = 0; // Raw reading from pin
  float voltage = 0; // Voltage in V
  float current = 0; // Current in A
  float wattage = 0; // Wattage in W
  float power = 0; // Power in J
  /* Average loop */
  for(int i = 0; i < n_reads ; i++) {
    average_a0 += analogRead(sensorPin_0);
    delay(loop_delay); }
  /* Formula based computations */
```

```
average_a0 /= n_reads;                                    13
voltage = (average_a0 / 1024.0) * 5;                      14
current = current_eq(voltage);                            15
wattage = voltage * current;                              16
power = wattage * interval;                               17
Serial.println(power, 3);                                 18
Serial.flush();}                                          19
```

Chapter 3

Benchmark Architecture and Design

This Chapter presents the necessary steps of our methodology to measure and compare different DBMS. Section 3.1 defines the energy model that supports our methods. Afterward, in Section 3.2, we present the DBMS we consider in our study. Then, Section 3.3 describes the design of the benchmarks we defined. Finally, in Section 3.4, we show their execution process.

3.1 Energy Measure Method

Before deciding the energy model we consider in our benchmark of the DBMS, it is essential to understand that the overall energy consumption of software systems divides into two groups: consumption when idle and dynamic consumption (that is, consumption when running).

As mention in Section 2.3.1, , idle consumption represents the energy needed for the system to run on minimal usage without any interference of user activity. Dynamic consumption is the energy consumed by a task or activity triggered by the user, where this energy is the difference between total consumption and idle consumption. Thus, the total energy consumed by the following equation:

$$E_{Total} = E_{Idle} + E_{Dynamic} \tag{3.1}$$

Where E_{Total} represents the total energy consumed by the software system consumption, E_{Idle} the idle consumption, and $E_{Dynamic}$ the dynamic consumption.

Idle Consumption: Idle consumption is the base consumption needed to ensure that the system is ready. It is obtained by measuring the energy consumption of the system during an interval of time. During which the system remains running with the lowest possible activity and without any user interference.

Using the frameworks mentioned in Section 2.3.3. It is possible to divide the E_{Idle} subgroups that we define as relevant in Section 2.3.2. The following equation presents these subgroups:

$$E_{Idle} = E_{Idle\_Others} + (E_{Idle\_CPU} + E_{Idle\_DRAM} + E_{Idle\_DISK}) \tag{3.2}$$

Where E_{Idle_CPU} represents the energy consumed by the CPU when the software is in an idle model, E_{Idle_DRAM} the idle consumption of the main memory, E_{Idle_DISK} the idle consumption of the disk, and E_{Idle_Others} represent the remaining energy consumed.

As mention in Section 2.3.3.1 , the E_{Idle_CPU} and E_{Idle_DRAM} can be estimated by RAPL. For measuring the E_{Idle_DISK}, we will use the Arduino method mentioned in Section 2.3.3.2.

By using the RAPL Package estimation metric, which includes CPU, and GPU, the following equation can be an alternative definition for the total consumption in idle mode:

$$E_{Idle} = E_{Idle\_Others} + (E_{Idle\_Package} + E_{Idle\_DRAM} + E_{Idle\_DISK}) \tag{3.3}$$

Dynamic Consumption: The dynamic energy consumption can only be determined after we have measured the idle consumption. Only then can we distinguish between idle and the energy impacts caused by the user. In this phase, any slight increase in the overall energy counts towards the dynamic consumption. Very much like in idle mode, this consumption must divide into the same groups. The following equation represents this separation :

$$E_{Dynamic} = E_{Dynamic\_CPU} + E_{Dynamic\_DRAM} + E_{Dynamic\_DISK} \tag{3.4}$$

As for the idle consumption, the $E_{Dynamic\_CPU}$ and $E_{Dynamic\_DRAM}$ can be measure with RAPL, and the $E_{Dynamic\_DISK}$ by the Arduino. To obtain these values, we need to remove the idle consumption of the total consumption in those components. Thus, for each subsystem, the equations are as follows:

$$E_{Dynamic\_CPU} = E_{Total\_CPU} - E_{idle\_CPU} \tag{3.5}$$

$$E_{Dynamic\_DRAM} = E_{Total\_DRAM} - E_{idle\_DRAM} \tag{3.6}$$

$$E_{Dynamic\_DISK} = E_{Total\_DISK} - E_{idle\_DISK} \tag{3.7}$$

As in Equation 3.3, CPU and GPU can be estimated together by the Package metric. The dynamic consumption also translates into the following equation:

$$E_{Dynamic\_Package} = E_{Total\_Package} - E_{idle\_Package} \tag{3.8}$$

$$E_{Dynamic} = E_{Dynamic\_Others} + (E_{Dynamic\_Package} + E_{Dynamic\_DISK} + E_{Dynamic\_DRAM}) \tag{3.9}$$

Through these analyzes, it is to conclude that idle consumption is not relevant to compare different DBMS on the same system since it is a static value. Therefore, we decided only to show and compare the dynamic energy consumption of all measurements in Chapter 4 for a better and easier understanding of the results obtained.

3.2 Databases Under Test

In our digital information age, where more and more information (only) exists in digital form (see, for example: how photography evolved in the last two decades), databases are a vital part of all types and organization sizes (from small to large) [41]. Moreover, data centers are also becoming an increasingly critical part of the infrastructure in our digitalized society. As a consequence of the concern with energy expenditure, a company/software engineer needs to understand their scenario and select the fitting DBMS that combines runtime performance with energy efficiency.

Although there is extensive (research) work on analyzing and benchmarking the runtime performance of DBMS [19, 105, 106, 109], there is still a lack of knowledge on the energy efficiency of the different DBMS that supports the data centers. In this thesis, we decide to compare four well-known and widely used DBMS: *MySQL*, *MariaDB*, *Postgres*, and *Redis*. Our decision on the DBMS was also based on whether the HammerDB supported it.

3.2.1 MySQL

The first DBMS chosen was MySQL. MySQL is the most popular open-source relational DBMS in the market and is known for providing high-performance, robust SQL, multi-threaded, multi-user access to several databases. Allan Larsson and Michael Widenius created MySQL in 1995, and now it is owned by Oracle Corporation. Some of its customers include GitHub, Uber, NASA, Tesla, Netflix .

MySQL's other features are: high compatibility, high portability, usage of fast B-tree disk tables with index compression, provides transactional and nontransactional storage engines, thread-based memory allocation system, optimized nested-loop join, in-memory hash tables used for temporary tables, and other things [43].

3.2.2 MariaDB

Another DBMS chosen was MariaDB. MariaDB is a popular open-source relational DBMS. The original developers of MySQL in 2009 made MariaDB ensure a free and open-source DBMS. Originally designed as an enhanced, drop-in replacement for MySQL, MariaDB is a fast, scalable, and many other tools that make it very versatile for a wide variety of use cases [68].

Since MariaDB is built on top of the latest version of MySQL, it has most of the features of MySQL and high compatibility between them. MariaDB provides some improvements from MySQL like more Storage Engines, some Speed Improvements like parallel replication, better testing, and other things [68].

3.2.3 Postgres

The last relational DBMS chosen was Postgres. Postgres is an open-source object-relational DBMS that uses and extends the SQL combined with many features that safely store and scale the most complicated data workload [108]. PostgreSQL started its development in 1986 at the University of California and has more than 30 years of active development on the core platform.

PostgreSQL's main attraction is its architecture, consistency, data integrity, robust feature set, extensibility, and the open-source community's commitment to delivering efficiency and creative solutions consistently. PostgreSQL is currently used in several research applications and comes with several add-ons, such as the popular PostGIS geospatial database extender. PostGIS is widely used for geographic data, and in many universities, they use as an educational tool due to its open-source code. It has some object-oriented features, such as inheritance and custom types, in addition to the characteristics of a relational DBMS.

3.2.4 Redis

The last DBMS chosen was Redis. Redis is an open-source non-relational DBMS of the type key-value that supports in-memory data structure store, used as a database, cache, and message broker [36]. Redis is a well-established open-source project, and many companies use Redis like Twitter, Tumblr, Instagram, Flick, and The New York Times. Redis is one of the most popular non-relational DBMS in the market[36]. Redis is known for its fast key-value database that stores a mapping of keys to five types of values: strings, lists, sets, hashes, sorted sets [1, 16]. It supports in-memory persistent storage at the disk, replication to scale read performance, and client-side sharding to scale write performance. Depending on the use case, the persisted data are either periodically dumped to disk or appending each command to a disk-based log. Redis also provides asynchronous replication[1, 16].

Additionally, Redis has configurable key expiration, transaction, and publish/subscribe features. It also provides Lua scripting to create new commands. With these tools, it makes a very versatile database[1, 36].

3.3 Design

We decided to use HammerDB as a benchmark tool that can replicate the behavior of the most cloud-based applications as a service, verify comprehensive performance and multiple metrics in a simple real-world environment on a virtual environment with Virtual users (VU) [107]. Another reason was the automated software testing [107] so we can configure how many times the HammerDB will run and how many VU it will use, and for how long it will run, providing at the end of every execution the TPM and the NOPM. With these numbers and the energy spent, we can have ample information to answer the **RQ2**, and with an adjustable number of users, it can assist in the **RQ3**.

The initial step to obtaining comparable results about the energy consumed in each DBMS is to define a configuration for HammerDB for each scenario.

With the Research Question mentioned in Section 1.2, we can deduce various scenarios. The number of users we run in the system differentiates these scenarios. So, we decide to go with four different scenarios of server participation. The first case is a server running with the lowest users possible, so we execute the benchmark with only 1 VU. A second scenario is a small group of users practicing on the server, so we run with 8 VU. The third scenario is a simulation of the server doing intensive work with a big group of users, and for that, we use 64 VU. The final one is the server saturated with users, and for that, we decide with 128 VU. While the first scenario is enough to answer **RQ1** and **RQ2**, the **RQ3** needs the others.

For each of them, we created a script for each DBMS. These scripts must simulate the behavior of daily usage of a DBMS. So using a custom script inside the HammerDB was out of the question, and we decided to use the TPC-C benchmark. We decide to do one warehouse in every scenario to simplify this study, with the intention in future work to expand to more warehouses.

To simplify this study, we decide to do one warehouse in every scenario with the intention of future work to expand this study to more warehouses.

Then we must decide on the time that the benchmark builds up the transaction rate by caching data in the buffer cache database before the benchmark is executed. This is known as the rampup time. In our study, we decide to set the ramup to 1/5 of the execution time [107].

For greater precision and adequate replication of various usage scenarios, each case was performed with 5, 10, and 30 minutes.

Finally, in addition to the previous scripts, we also created a script that would remove idle consumption from the measured energy consumption using a previous measured idle consumption.

It is presented and discussed in Chapter 4 only the 5 minutes tests. Even with most of the 10 and 30 minutes results measured, they weren't discussed here due to their conclusions being very similar to the conclusions of 5 minutes. These results are available in the annex of this document.

3.4 Execution

Figure 3.1 shows the execution of the benchmark and measurements of energy consumption of DBMS, where it displays the architecture of our energy benchmarking system and the flow of actions.

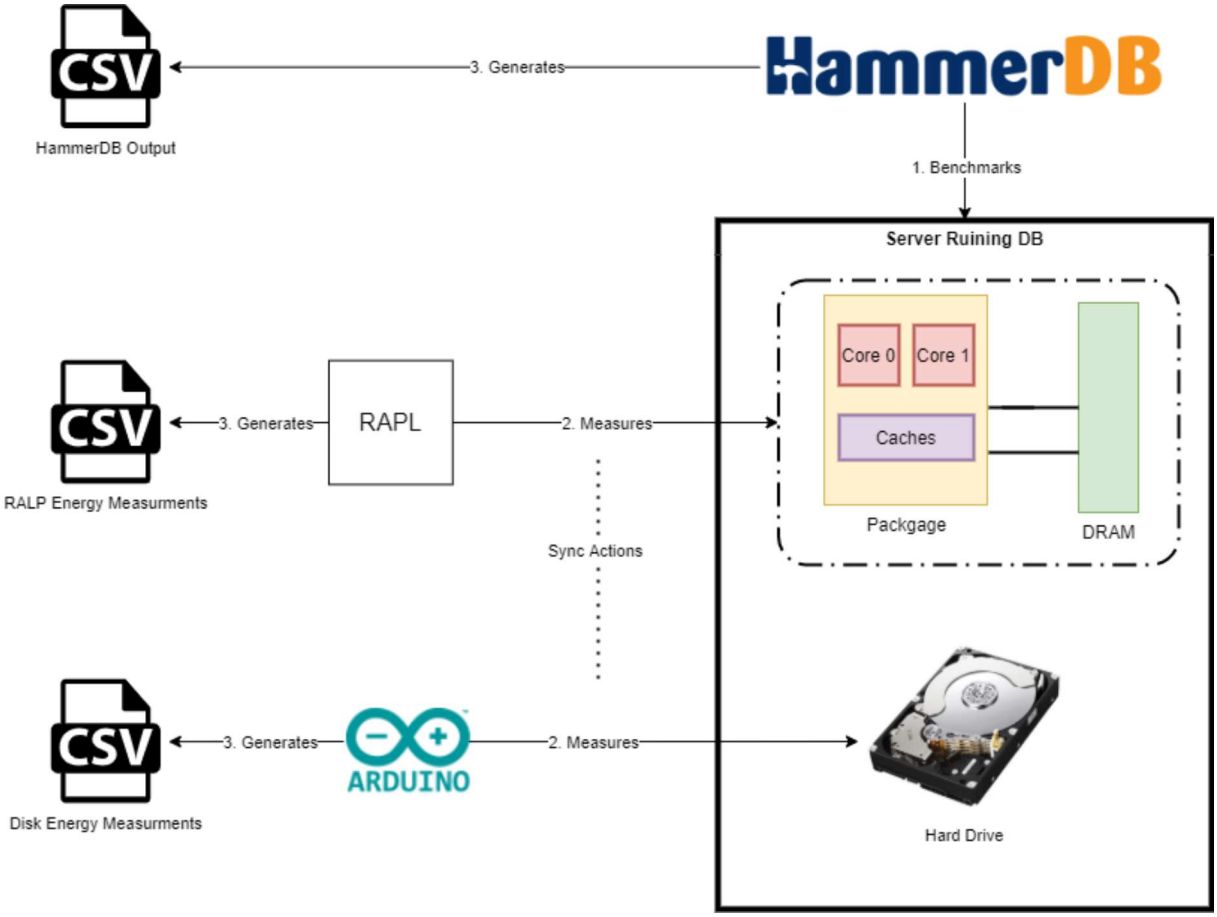


Figure 3.1: Benchmark Architecture and flow of events

As seen in Figure 3.1, the first action of our execution is the Benchmark initialization and execution of HammerDB. To obtain proper and reliable energy readings, we needed to diminish the effects of cold starts, warm-ups, cache effects, and other effects that may influence energy consumption. So, each scenario is executed ten times with a sleep time of 2 minutes between each execution.

Immediately after the start of the Benchmark execution, The measurements of the energy consumption start. The framework tools used were the ones mentioned in Section 2.3.3. In the case of the RAPL, the sample used was 100 measurements per second, and for the Arduino was ten measures per second.

These two measurement tools are in sync throughout the entire execution. We accomplished this by using a C program that starts two independent sync threads: one that accesses the RAPL to measure the CPU is power consumption, and the other accesses the Arduino to measure the disk's power consumption. They are active during the execution of HammerDB and accumulate energy readings in static arrays to limit their computing overhead.

The third and final phase of our execution begins after the HammerDB execution is completed. This step saves each measurement mad in its respective CSV file for further analysis.

Also, for each database, we decide to use dockers containers because it is a solution that simplifies the setup by providing a pre-built image that is portable, simple to maintain, and providing a facility to automate the applications when they are deployed into containers [12, 95]. Although several studies recognize that dockers provide an overhead in running these containers [22, 40, 45, 118], dockerized benchmarks can be acceptable when comparing different database systems if the idle consumption values are disregarded [53].

System Configuration We ran this study on a server with the following specifications presented in Table 3.1, this consists of an Intel(R) Core i5-4460 3.28 GHz CPU, 16 GB of RAM, 250 GB Hard Drive Disk (HDD), and operating system Ubuntu 16.10 . A detailed overview of the CPU is in Table 3.2 and for Second Storage is in Table 3.3.

Hardware	Model
CPU	Intel(R) Core i5-4460 3.28 GHz
Operation System	Ubuntu 16.10
Ram Size	16G
Secondary Storage	Hitachi Travelstar 250 GB

Table 3.1: Physical server specifications

CPU	
Brand	Intel
Model	i5-4460
Microarchitecture	Haswell
Number of cores	4
Clock Speed	3.20 GHz
Max Turbo Frequency	3.40 GHz
Cache	6 MB
Interface	Sata 3
Buffer Size	8MB

Table 3.2: Specifications of CPU used

Second Storage	
Brand	Hitachi GST
Model	HTE543225A7A384
Series	Travelstar Z5K320
Type	HDD
Capacity	250 GB
RPM	5400
Interface	Sata 3
Buffer Size	8MB

Table 3.3: Specifications of Disk used

This system has no additional software installed or running other than required to run this research. Additionally, we had the caution to use the most recent and compatible version of all external software used here at the time of the measurements. So listed in Table 3.4 are all the versions used.

Software	Version
Redis	5.0.5
Postgres	11.5
MySQL	8.0
MariaDB	10.0
HammerDB	3.2
Docker	19.03.11

Table 3.4: Software Configuration on physical server

All software artifacts shown in Figure 3.1 and mention in this chapter, are available as a public repository at `https://github.com/greensoftwarelab/GreenSGDBS`.

Chapter 4

Energy Efficiency of DBMS

This chapter presents the results obtained when running the benchmark on the four DBMS considered in this master thesis. Here, we show and analyze the results obtained by the HammerDB and the energy consumption obtained by the RAPL and Arduino.

4.1 Benchmark Results

In this Section, we will present and analyze in great detail the performance of MySQL, MariaDB, Postgres, and Redis. We consider the energy consumed for each DBMS when executed by HammerDB both in a single user mode and when multiple VU are performing DBMS actions. Moreover, we distinguish the energy consumed by the Package, estimated by RAPL, and the energy consumed by the hard disk, as measure by the Arduino system. Concerning the RAPL estimations, we will consider the estimation provided by the RAPL package metric: it includes the consumption of the Package cores, DRAM, and other components of the chip.

As described in Chapter 2.1.4.2, HammerDB computes two DBMS-specific performance metrics: the Transactions per Minute (TPM) and the New Orders Per Minute (NOPM). Different DBMS may show different per minute performances, which can have an impact on energy consumption. Thus, we will analyze the impact of such HammerDB metrics on both package and disk energy consumption.

4.1.1 DBMS in Single User: Energy Consumption

First will discuss energy consumption on every level for every DBMS with only one VU . Figure 4.1 shows the median values, and Figure 4.2 shows the distribution of these values in a box plot. Here we can see the median, the approximate quartiles, the lowest and highest data points to convey the level, spread, and symmetry of a distribution of data values [112].

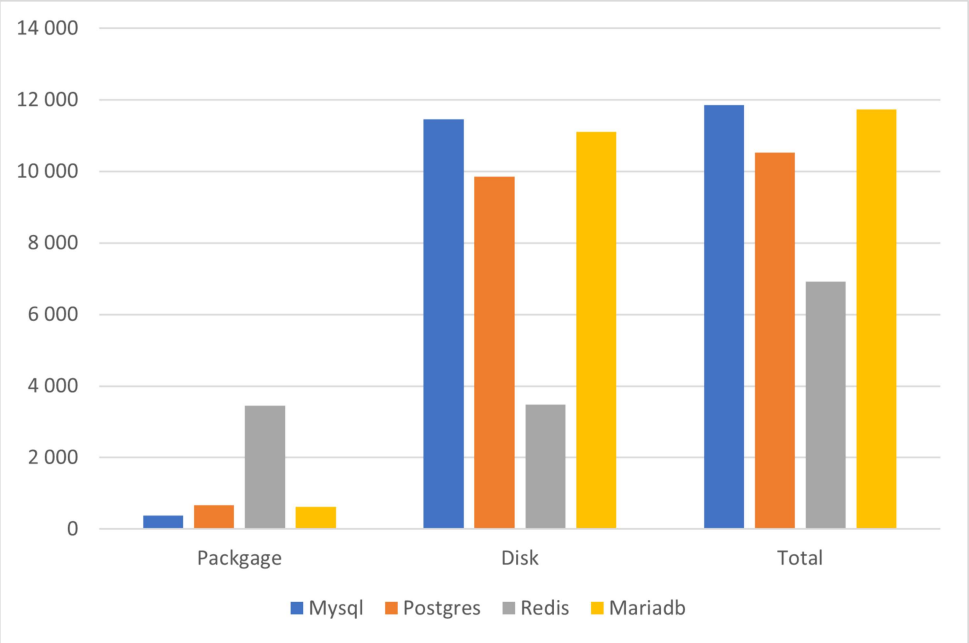


Figure 4.1: Median of energy consumption on Package, Disk and Total.

Looking at the package measurements, Redis shows the highest consumption energy median value, followed by Postgres, MariaDB, and MySQL. We can see that all relational DBMS have a lower energy consumption compared to Redis.

The inverse of the Package level happens at the disk level, where the relational DBMS spends in median the most energy and the non-relational spends the least. In this situation, MySQL was the one that spent the most afterward was MariaDB and then Postgres and lastly Redis.

It is important to notice even though Redis spent in median more than the others on the Package level, the energy he saves on the disk pays off in terms of total energy consumption, making it the least expensive one, followed by MariaDB, Postgres, and MySQL. The reason for this is that secondary storage has a higher effect than the Package on the overall energy consumed.

The measured values can be explained by Redis being a non-relational DBMS of type Key-Value where the data is stored in-memory explains why the consumption on disk is low comparing with the Relational DBMS. This naturally implies a higher consumption on the Package level since use of cache affect energy consumption on the Package [16].

(a) Distribution of energy consumed on Package

(b) Distribution of energy consumed on Disk

(c) Distribution of energy consumed on Overall System

Figure 4.2: Distribution of energy consumed

When talking about the distribution of the values between the different executions, we can see in Figure 4.2a that the distribution on the Package level is smaller on the relational databases, and Redis has a large magnitude distribution on values of energy consumption. Also, it is worth mentioning a specific case in relational DBMS. Although, on average, Postgres is the most expensive of relational DBMS, the maximum energy consumption made by MySQL and MariaDB surpass the minimum energy consumption of Postgres.

In secondary storage, seen in Figure 4.2b, there is almost no variation of Redis results, and the same case occurs in Postgres, but the distribution in MySQL and MariaDB is wide. The median between MySQL and MariaDB was very close. But when observing both distributions on this level, we can see the magnitude of the values of MySQL is a lot bigger comparing with MariaDB. Here also worth mention that the maximum of Postgres surpasses the minimum of both MySQL and MariaDB.

As it happens with the median of energy consumption, the distribution of total energy consumed follows almost the same pattern of distribution of energy consumed of the disk because of the impact it has on total energy consumed where the distribution of MySQL remains the most significant. These values are presented in Figure 4.2c.

Runtime Performance Now discussing the HammerDB metrics, similar graphs are presented in Figures 4.3 for the median of these metrics and Figure 4.4 for the distribution.

As seen in Figure 4.3, Redis has a much better performance, in terms of TPM, than any relational DBMS. We can also see a relationship between TPM with less consumption in the disk. When talking about NOPM, Postgres is the one with better performance. MariaDB and Redis have the same number of NOPM, and the lowest is MySQL. Here the results are very close, and there's not much margin between

them, and here can't be draw any conclusion between energy consumption and performance.

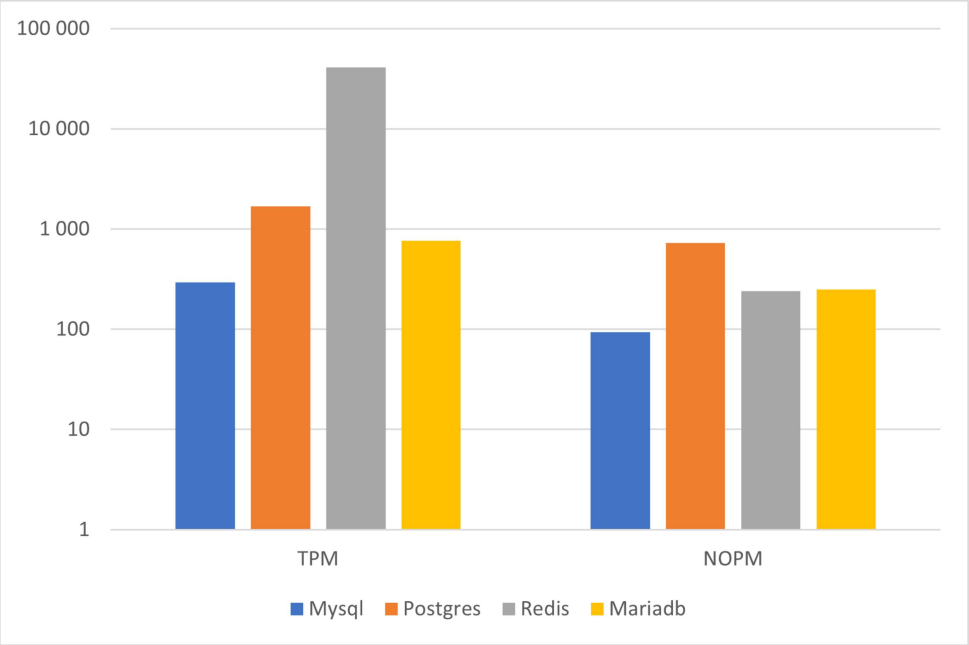
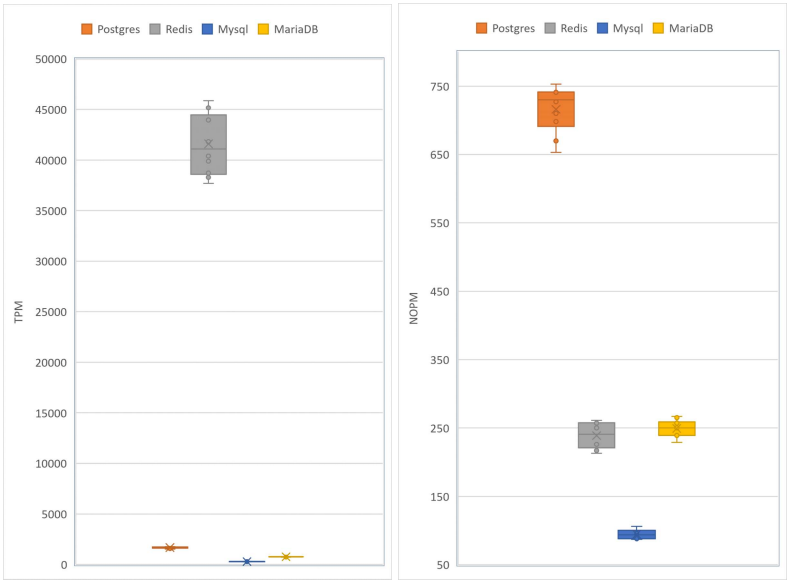


Figure 4.3: Median of HammerDB results.

When look at the HammerDB results, TPM distribution between different tests doesn't have much variation except on Redis where the variation is noted in Figure 4.4a. When looking at the distribution of NOPM in Figure 4.4b, we can see that it doesn't have much variation between different DBMS the only thing worth mention is that even knowing that MariaDB and Redis have the same median, Redis has a larger distribution meaning that in some executions Redis can have worst performance than MariaDB.



(a) Distribution of TPM (b) Distribution of NOPM

Figure 4.4: Distribution of HammerDB results

Energy Consumption per Runtime Performance The last discussion of single user is the Energy Consumption per HammerDB metrics. For Energy Consumption per TPM, Figure 4.5 for the median of Energy Consumption Per TPM and Figure 4.6 for the distribution of Energy Consumption Per TPM. For the Energy Consumption per NOPM, Figures 4.7 for the median of Energy Consumption Per NOPM and Figure 4.8 for the distribution of Energy Consumption Per NOPM.

In terms of Joules per TPM on the package, disk, and total, as seen in Figure 4.5, it follows the same trend of the most expensive in terms of energy consumption where the MySQL is the most expensive followed by MariaDB, Postgres, and Redis. The distribution of these values in Figure 4.6 show that MySQL is always the most expensive followed by MariaDB, Postgres, and Redis.

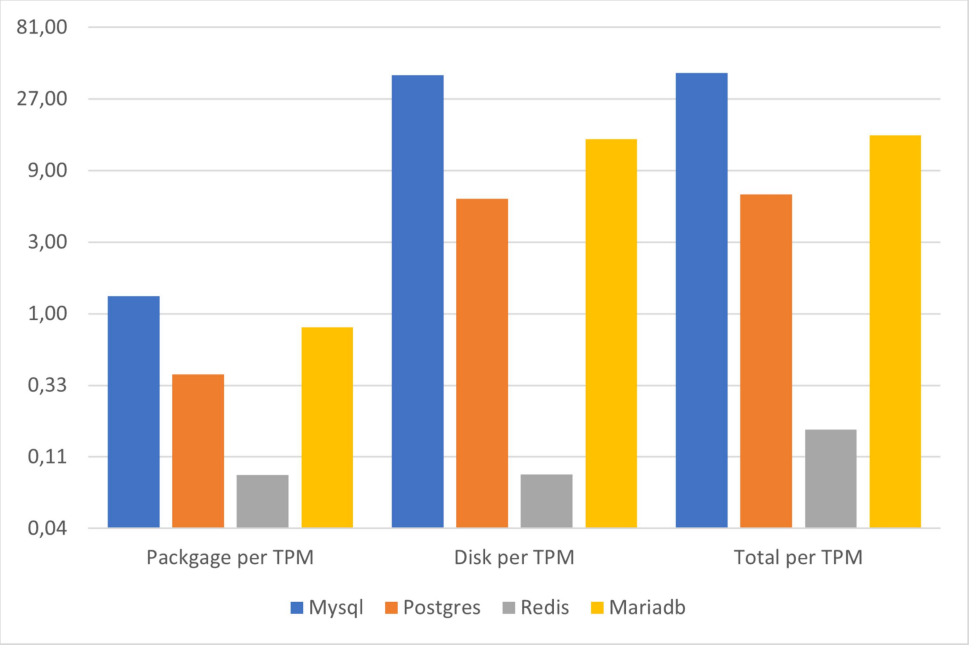
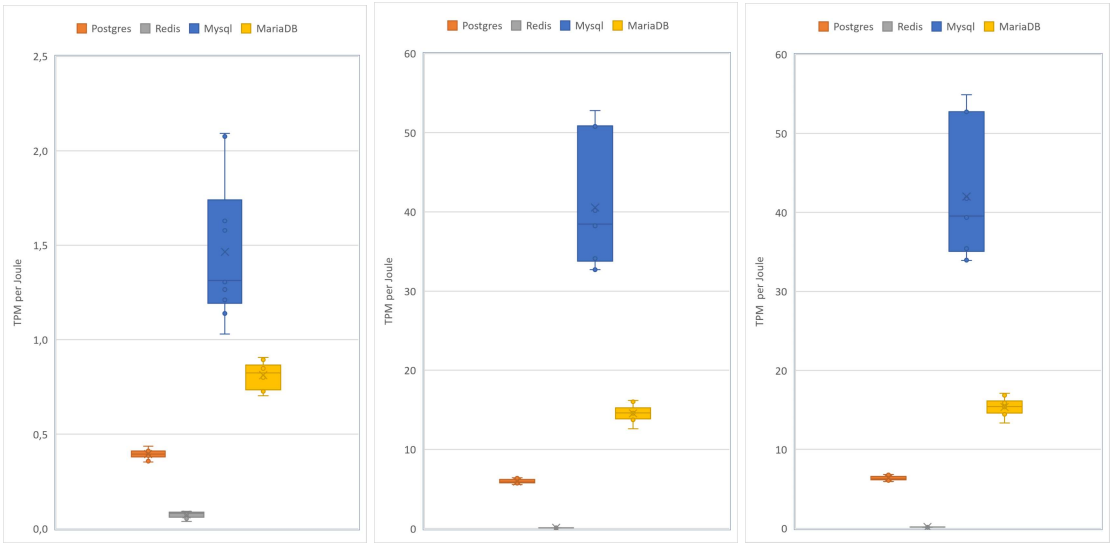


Figure 4.5: Median of energy consumption per TPM

In terms of Joules per NOPM, we can see the median in Figure 4.7 that in the package the one with the most NOPM per Joules is Redis then MySQL, MariaDB, and Postgres. On the disk the one with the most Joules per NOPM is MySQL then MariaDB, Redis, and finally Postgres. When talking of the system as a whole the MySQL is the one with the Joules per NOPM, follow by MariaDB, Redis, and Postgres. This result doesn't follow any trend of the rational database being the most expensive or non relational database being the less expensive in any of the cases.



(a) Energy consumption on Package per TPM (b) Energy consumption on Disk per TPM (c) Energy consumption on Overall System per TPM

Figure 4.6: Distribution Energy consumption per TPM

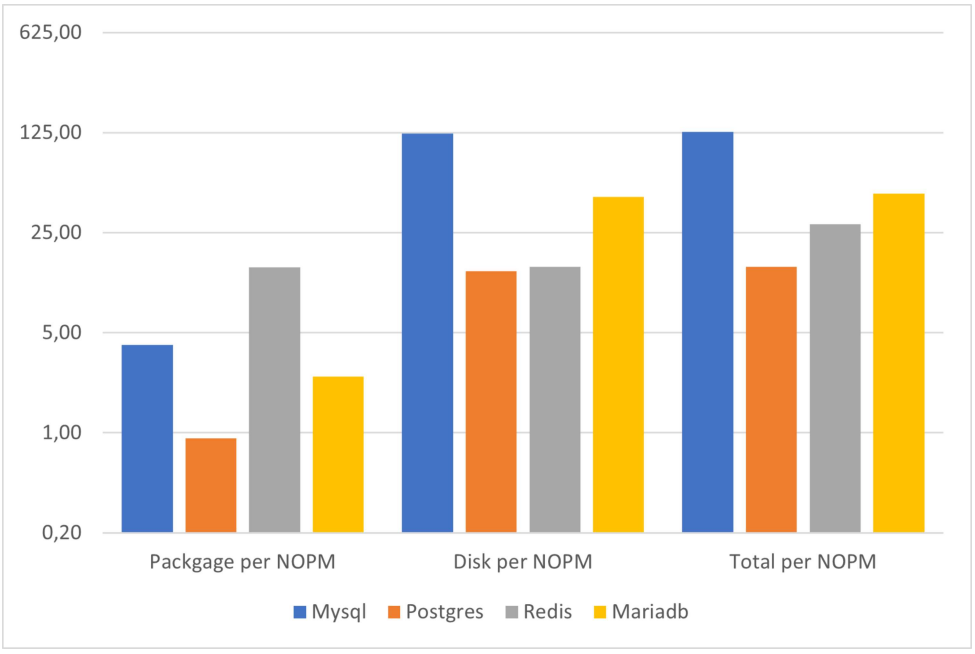
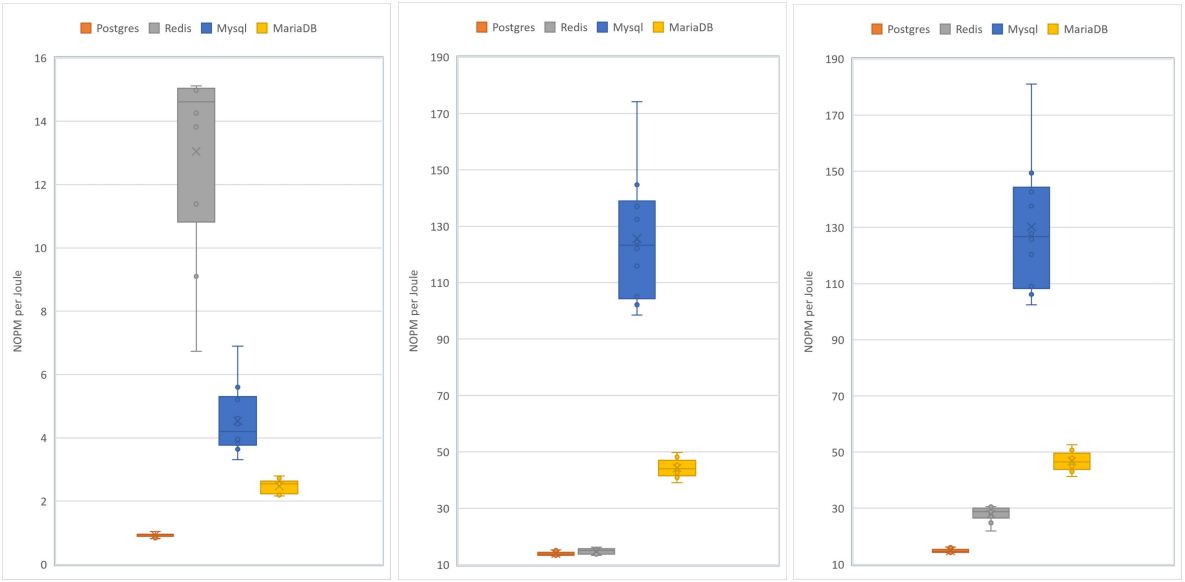


Figure 4.7: Median of energy consumption per NOPM

4.1.2 DBMS in Multi User: Energy Consumption

Now for the Multi users, we will first be discussing the energy consumption on Package as shown in Figure 4.9a. Redis is the only one with slightly different behavior with an increase of VU, where the other three DBMS had a rise of energy consumption, also Redis has decreased with 8 VU followed by a rise with 32 VU follow by another drastic decrease. Also noted that of the relational DBMS, MariaDB is the one with more increase, and MySQL is the one with the lowest increase.



(a) Energy consumption on Package per NOPM

(b) Energy consumption on Disk per NOPM

(c) Energy consumption on Overall System per NOPM

Figure 4.8: Distribution of energy consumption per NOPM

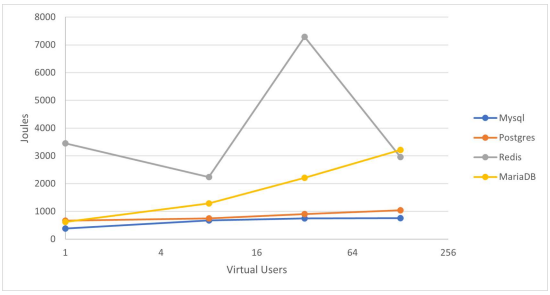
In Figure 4.9c can be seen the energy consumption on disk with an increase of VU. MariaDB has an noticeable increase, Postgres also has an increase, MySQL and Redis has a improvement in energy consumption. And with these variations of VU, the MariaDB become the most expensive followed by Postgres, MySQL, and Redis.

On a general view, Redis has the same pattern as in the disk being the lowest in terms of energy consumption except on 32 VU, where the increase in Package has a large impact on the overall energy and putting him in the second-lowest behind MySQL. MySQL starts as the highest and with the increase becomes the second-lowest energy consumption DBMS expect on 32 VU where he is the lowest. Postgres is second highest. Finally, MariaDB with the increase of VU become the most expensive DBMS.

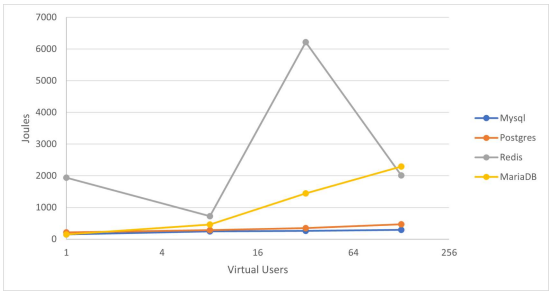
The only conclusion that can maybe be drawn here is that Redis has an increase in energy efficiency that can be possible due to its nature as a non-relational database with data in memory.

Runtime Performance with Multi Users When observing the performance of HammerDB benchmark with an increase of VU, first can be seen that all DBMS had an increase of TPM and NOPM with the variation of VU. The only instance that had a decrease was from 32 to 128 VU on Redis. With the increase of VU, Redis still maintain the DBMS with better TPM and NOPM and MySQL stays the worse, while Postgres with an increase of VU start to get worse results comparing with MariaDB. This results are in Figure 4.10a and Figure 4.10b.

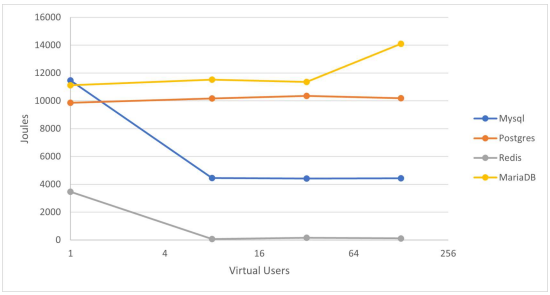
Energy Consumption per Runtime Performance with Different Users When discussing energy consumption per TPM as seen in Figure 4.11, we can see with the increase of VU that Redis have a better ratio in terms of energy per TPM, and in general, all the databases except Postgres improved very well even



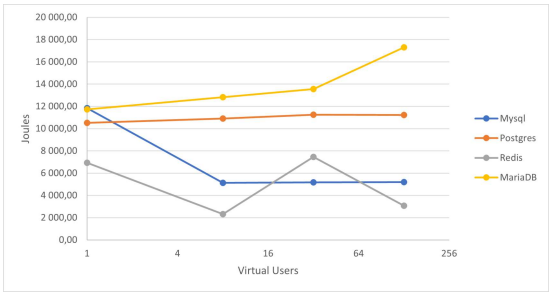
(a) Energy consumption on Package with different number of users



(b) Energy consumption on CPU with different number of users

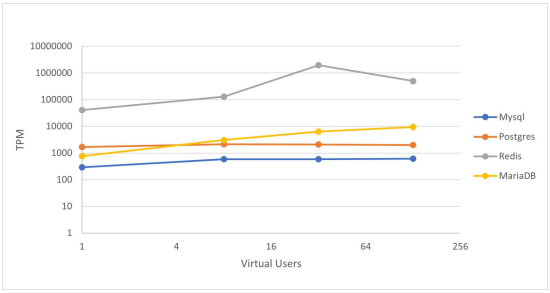


(c) Energy consumption on Disk with different number of users

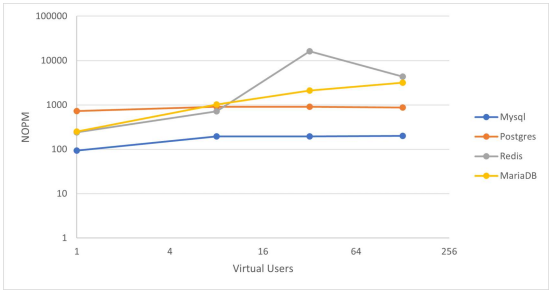


(d) Energy consumption on Overall System with different number of users

Figure 4.9: Energy consumption with different number of users



(a) TPM with different number of users



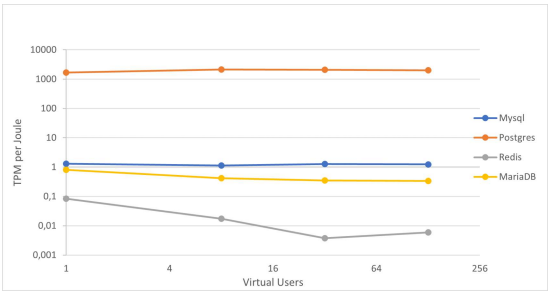
(b) NOPM with different number of users

Figure 4.10: HammerDB results with different number of users

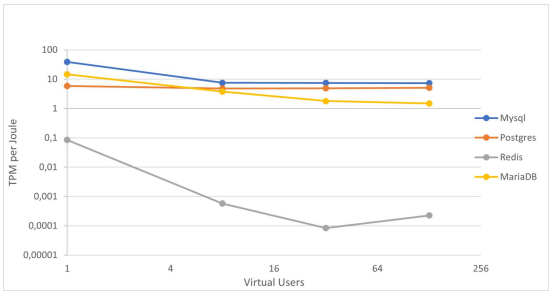
though that MariaDB is the relational database with better results. These improvements happen because all DBMS except Postgres had a reduction of energy consumption on the disk per TPM that provides a decrease in total energy consumption per TPM.

Finally, in Figure 4.12 we can check the energy consumption per NOPM, we can see that all DBMS improve very well and we can see that Redis became the most efficient on energy consumption per NOPM with the increase of VU, follow by MariaDB, Postgres, and MySQL. These improvements are very similar to the improvement in energy consumption per TPM where the decrease of energy consumption on disk per NOPM has a great impact on total energy consumption per NOPM.

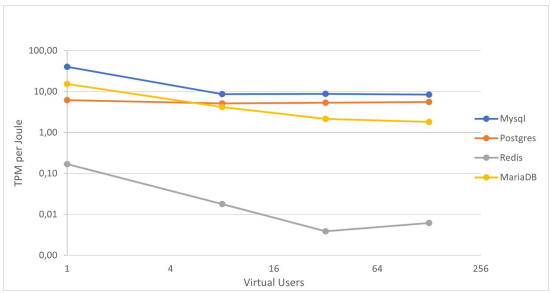
On an overall note, it can be concluded that that the non relational DBMS Redis start to get better results on NOPM and TPM without a large increase in energy consumption making Redis the most scalable one.



(a) Energy consumption on Package per TPM with different number users

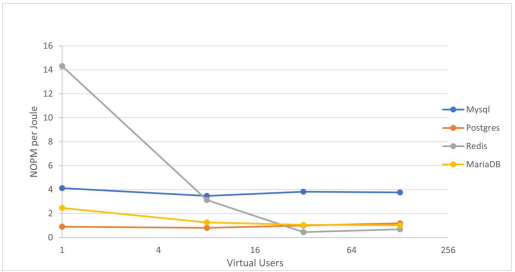


(b) Energy consumption on Disk per TPM with different number users

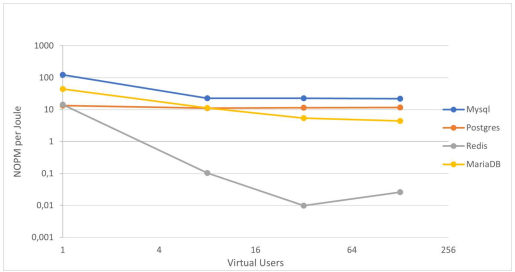


(c) Energy consumption on System per TPM with different number users

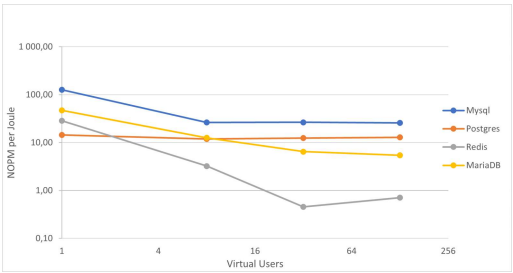
Figure 4.11: Energy consumption per TPM with different number users



(a) Energy consumption on Package per NOPM with different number of users



(b) Energy consumption on Disk per NOPM with different number of users



(c) Energy consumption on System per NOPM with different number users

Figure 4.12: Energy consumption per NOPM with different number of users

When talking about relational DBMS, MariaDB had the most increase in HammerDB performance and energy consumption per TPM and NOPM with the increase of VU and Postgres maintain or get worse results comparing with only one VU.

Chapter 5

Threats to Validity

Here we speak about risks that could threaten the accuracy of the presented results and conclusions. Also, we divide every threat to validity found into four categories as defined in Cook et al. (1979):

Conclusion Validity In this category are the threats which may influence the capacity to draw correct conclusions [125].

Another threat of this category is the reliability of measures. In this thesis, we made various comparisons between the different consumptions and performance in every DBMS. In addition, comparisons between subsystems, such as Package, DRAM, or secondary storage. Also, all DBMS execution and benchmark were in an equal manner, and every test was executed ten times, and calculated the median, mean, standard deviation, min, and max values of these tests. So our measures are reliable.

Fishing for particular results can be an issue since the analyses are no longer independent [125]. Here, it doesn't apply because this study isn't trying to promote any DBMS, and it isn't looking for any specific outcom

Reliability of treatment implementation is another issue on conclusions validity. The software used was made by external developers. So, all the software made is independent of this study, and we reused it here to satisfy our needs.

Random heterogeneity of subjects is the last threat concerning our study. While we only compare four DBMS, this set includes some of the most popular ones, and even if we wanted more variety, we have the restriction of the HammerDB compatibility.

Regarding the Hammerdb results conclusions, we only compared TPM and NOPM between each DBMS. Comparing these two metrics may not be the most appropriate thing to do. While NOPM is used to compare different DBMS and is the optimal way to analyze DBMS, the same can't be said of TPM since it is not a recommended metric to compare DBMS [107]. The unreliability of these metrics happens because every database reports different transaction rate metrics in their online tools

Construct Validity Here are threats that involve the generalization of the results to the concept or theory behind the experiment [125].

The first threat here is the inadequate pre-operational explication of constructs. This threat means that the constructs are not sufficiently defined before they are measured [125]. Here, we measure energy and HammerDB. So the measurements were clearly defined, making this issue minor or nonexistent here

Another danger is the interaction of different treatments, which could be a problem if the subject enrolls in several studies that could overlap [125]. That does not happen here because we guarantee that only the minimum processes required to run each test are running. Additionally, a two-minute idle time rest was essential to the system cool down to decrease overheating between each execution, allowing the system to treat garbage collecting and not affect the results with these actions. In addition, when analyzing every database, they are in the same environment where we execute each script, which was similar in terms of workload configuration on HammerDB making all DBMS tested by the same rules

Mono-method bias is another issue, and it is about the usage of a single type of measure that involves a risk that the experiment could be misleading [125]. Even though here we only use the same methods to measure energy. However, both are known to be very precise and reliable for measuring energy consumption. So this doesn't affect the quality of the results.

The last threat in this category is the mono-operation bios, which is the experiment that includes only a single case, and it may under-represent the construct and thus does not give the whole picture of the theory [125]. Therefore, it does not apply here because even though we only use one context, the context is very reliable. After all, it represents an overall usage of a DBMS, making these results credible

Internal Validity Threats to internal validity are influences that can affect the results of the study. Therefore, they endanger the conclusion of a possible association between treatment and outcome [125].

One of these threats is instrumentation. This threat is the effect of the artifacts used for the execution of the experiment [125]. In this case, it was used scripts to collect the energy and execute HammerDB. Despite this, the scripts used on HammerDB are simple scripts to orchestrate the flow of SQL statements to the database to generate the required load. The other script used is to call software that executes energy measurement frameworks, where this software ensures that RAPL and Arduino are in sync, and saves in a struck in C and write the energy measurements inside it. This can be seen as an overhead during measurements. But because this happens in every execution, this impact can be considered non-important.

External Validity This category is concerned with the generalization of the results to industrial practice [125].

Interaction of selection and treatment is the selection made not representative of the population wanted to generalize [125]. It doesn't apply here because the databases selected are among the most popular on the market. As a result, this only applies to an industrial setting where such databases are used or may be used.

Another threat in this category is the interaction of setting and treatment. It is the effect of not having the experimental settings or materials representative of industrial practice [125]. In this study case, the benchmark tool HammerDB reproduces the industrial market because it is a well-known open-source tool used by different companies like Oracle, IBM, Intel, Dell/EMC HPE, Huawei, Lenovo, and hundreds more. Besides, the compilers, software versions, and computers used are recent and in line with industry standards. It is possible to adapt the Arduino to any industrial system, but it needs calibration. In the Portela study, there is a calibration for Arduino used in this work.

Chapter 6

Conclusion and Future Work

This thesis fits in the green software area, which is a crucial topic in today's world which over the years, energy-efficient of large-scale computing has become more relevant with the high growth in the volume of data processed through cloud services. Since DBMS can work as a cloud service, it is essential to understand and study their consumption.

This dissertation ends with this chapter, which focuses on the main conclusions taken from the results benchmark of DBMS. All things said we divided this chapter into two sections. First, Section 1 starts with a summary of the study carried out during this thesis and a brief answer to the research questions formulated in Chapter 3. Finally, Section 2 makes some ideas about research for the future on this topic and some limitations in this work.

6.1 Final Considerations

In summary, the main aim of this master's thesis was to compare various DBMS applications in the most practical environment scenario possible.

The impulse that drove this project was the concern about energy awareness in developing software and the lack of knowledge in DBMS energy consumption. One of the motives to choosing DBMS and not other software was the current studies and practices on databases emphasizing performance more than energy efficiency. Also, this area presents solutions for improving energy consumption in a data center when dealing with DBMS.

Table 6.1 is a summary view of the results obtained in this thesis. There we present the results into a classification table of each SDGB in each scenario.

Scenarios	MySQL	Postgres	MariaDB	Redis
Static Virtual User (1 VU)				
Energy Consumption	4	2	3	1
Performance TPM	4	2	3	1
Performance NOPM	4	1	3	2
Energy per TPM	4	2	3	1
Energy per NOPM	4	1	3	2
Multi Virtual Users				
Energy Consumption	2	3	4	1
Performance TPM	4	3	2	1
Performance NOPM	4	3	2	1
Energy per TPM	4	3	2	1
Energy per NOPM	4	3	2	1

Table 6.1: Classification of each DBMS in each Scenario

After analyzing these results, it is possible to answer the three research questions presented in Section 1.2:

- **RQ1**:*Which Database Management System is the most energy-efficient?* Here the conclusion is that with only one virtual user, Redis is the most energy-efficient on a general note by far. If we only compare Relational DBMS, the best is Postgres, follow by MariaDB then MySQL. On a more specific level like Package/CPU, due to its non-relational nature, Redis is less energy-efficient, while it is, due to the same reason, the most efficient on the Disk level.

- **RQ2**:*Which Database Management System has the best energy versus runtime tradeoff?* Here is a bit different since this question is about which database has the best performance per energy consumption. Here it has two observations, the first one of TPM and the more reliable NOPM. On energy consumption per TPM, Redis is the most efficient in all subcomponents followed by Postgres, MariaDB, and the worst MySQL. Here at all levels, Postgres is the one with the lowest energy consumption, followed by Redis, MariaDB, and MySQL.

- **RQ3**:*How does the increasing number of users of the Database Management System impact their energy consumption?* In this final Research Question, there are two topics to be covered: The first is what DBMS has better scalability in terms of energy consumption, where MariaDB is by far the worst energy-efficient DBMS with an increase of users, Postgres maintains similar energy consumption, MySQL and Redis improve their energy efficiency. The other topic is energy consumption per performance scalability in both energy consumption per TPM and energy consumption per NOPM, which has similar results, which Redis is who improves most, followed by MariaDB, Postgres, and MySQL.

6.2 Future Work

The presented study in this master thesis, as initially proposed, reduces some of the lack of knowledge about energy efficiency in DBMS. Even though this study furthered the advancement of Energyware Engineering in DBMS, there is still work left to do.

In terms of work that could boost this study would be doing the same benchmark with longer times. Even though we did tests for 5, 10, and 30 minutes, the objective would be doing it for an even longer time, for example, 24 hours. Additionally, extend this study to all DBMS available on HammerDB but different types of DBMS not available in HammerDB.

Another path to this study would be to do this study with different hardware and compare the results. Examples of this would be to use a different size of RAM, SSD, or a CPU.

Since the benchmark applied here is a standard benchmark, it would be interesting to explore the DBMS energy consumption administered to different benchmark software and workloads. With this, it would bring a vision of DBMS energy consumption applied to distinct environments. The last suggestion would be doing a benchmark but applying energy-aware query optimization in this benchmark as a custom script and with this understand the effect of this optimization on an energy-efficiency and performance.

Bibliography

[1] url: https://redis.io/.

[2] D. J. Abadi, P. A. Boncz, and S. Harizopoulos. "Column-oriented database systems." In: *Proceedings of the VLDB Endowment* 2.2 (2009), pp. 1664–1665.

[3] R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, et al. "The Claremont report on database research." In: *ACM Sigmod Record* 37.3 (2008), pp. 9–19.

[4] Q. Ali and P. Yedlapalli. "Persistent Memory Performance in vSphere 6.7." In: (2019).

[5] R. Angles. "A Comparison of Current Graph Database Models." In: *2012 IEEE 28th International Conference on Data Engineering Workshops*. 2012, pp. 171–177. doi: 10.1109/ICDEW.2012.31.

[6] R. Angles and C. Gutierrez. "Survey of graph database models." In: *ACM Computing Surveys (CSUR)* 40.1 (2008), pp. 1–39.

[7] L. Ardito, R. Coppola, M. Morisio, and M. Torchiano. "Methodological Guidelines for Measuring Energy Consumption of Software Applications." In: *Scientific Programming* 2019 (2019).

[8] M. Avgerinou, P. Bertoldi, and L. Castellazzi. "Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency." In: *Energies* 10 (2017), p. 1470. doi: 10.3390/en10101470.

[9] L. A. Barroso and U. Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines." In: *Synthesis lectures on computer architecture* 4.1 (2009), pp. 1–108.

[10] O. Belo, R. Gonçalves, and J. Saraiva. "Establishing Energy Consumption Plans for Green Star-Queries in Data Warehousing Systems." In: *2015 IEEE International Conference on Data Science and Data Intensive Systems*. 2015, pp. 226–231. doi: 10.1109/DSDIS.2015.108.

[11] K. Bilal, S. U. R. Malik, S. U. Khan, and A. Y. Zomaya. "Trends and challenges in cloud datacenters." In: *IEEE cloud computing* 1.1 (2014), pp. 10–20.

[12] C. Boettiger. "An Introduction to Docker for Reproducible Research." In: *SIGOPS Oper. Syst. Rev.* 49.1 (Jan. 2015), pp. 71–79. issn: 0163-5980. doi: 10.1145/2723872.2723882. url: https://doi.org/10.1145/2723872.2723882.

[13] E. Brewer. “CAP Twelve years later: How the ”Rules”have Changed.” In: *Computer* 45 (Feb. 2012), pp. 23–29. doi: 10.1109/MC.2012.37.

[14] E. A. Brewer. “Towards robust distributed systems.” In: *PODC*. Vol. 7. 10.1145. Portland, OR. 2000, pp. 343477–343502.

[15] T. Carçao. “Measuring and visualizing energy consumption within software code.” In: *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2014, pp. 181–182.

[16] J. L. Carlson. *Redis in Action*. USA: Manning Publications Co., 2013. isbn: 1617290858.

[17] R. Casado and M. Younas. “Emerging trends and technologies in big data processing.” In: *Concurrency and Computation: Practice and Experience* 27.8 (2015), pp. 2078–2091. doi: https://doi.org/10.1002/cpe.3398. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3398. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3398.

[18] R. Cattell. “Scalable SQL and NoSQL data stores.” In: *Acm Sigmod Record* 39.4 (2011), pp. 12–27.

[19] F. Chen and M. Hsu. “A Performance Comparison of Parallel DBMSs and MapReduce on Large-Scale Text Analytics.” In: *Proceedings of the 16th International Conference on Extending Database Technology*. EDBT ’13. Genoa, Italy: Association for Computing Machinery, 2013, pp. 613–624. isbn: 9781450315975. doi: 10.1145/2452376.2452448. url: https://doi.org/10.1145/2452376.2452448.

[20] M. Chen, S. Mao, and Y. Liu. “Big data: A survey.” In: *Mobile networks and applications* 19.2 (2014), pp. 171–209.

[21] Y. Chen, X. Xie, and J. Wu. “A Benchmark Evaluation of Enterprise Cloud Infrastructure.” In: *Web Technologies and Applications*. Ed. by R. Cheng, B. Cui, Z. Zhang, R. Cai, and J. Xu. Cham: Springer International Publishing, 2015, pp. 832–840. isbn: 978-3-319-25255-1.

[22] M. T. Chung, N. Quang-Hung, M.-T. Nguyen, and N. Thoai. “Using docker in high performance computing applications.” In: *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*. IEEE. 2016, pp. 52–57.

[23] C. Churcher. *Beginning SQL Queries: From Novice to Professional (Beginning from Novice to Professional)*. USA: Apress, 2008. isbn: 1590599438.

[24] E. F. Codd. “A relational model of data for large shared data banks.” In: *Communications of the ACM* 13.6 (Jan. 1970). doi: 10.1145/362384.362685.

[25] E. F. Codd. “Extending the Database Relational Model to Capture More Meaning.” In: *ACM Trans. Database Syst.* 4.4 (Dec. 1979), pp. 397–434. issn: 0362-5915. doi: 10.1145/320107.320109. url: https://doi.org/10.1145/320107.320109.

[26] E. F. Codd. “Relational Database: A Practical Foundation for Productivity.” In: *Commun. ACM* 25.2 (Feb. 1982), pp. 109–117. issn: 0001-0782. doi: 10.1145/358396.358400. url: https://doi.org/10.1145/358396.358400.

[27] T. M. Connolly, C. E. Begg, and A. D. Strachan. *Database Systems: A Practical Approach to Design, Implementation and Management*. USA: Addison Wesley Longman Publishing Co., Inc., 1996. isbn: 0201422778.

[28] T. D. Cook, D. T. Campbell, and A. Day. *Quasi-experimentation: Design & analysis issues for field settings*. Vol. 351. Houghton Mifflin Boston, 1979.

[29] T. P. P. Council. “TPC BENCHMARK C Standard Specification Revision 5.11.0.” In: (2010).

[30] M. Couto, J. Saraiva, and J. P. Fernandes. “Energy Refactorings for Android in the Large and in the Wild.” In: *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2020, pp. 217–228. doi: 10.1109/SANER48275.2020.9054858.

[31] M. Couto, T. Carção, J. Cunha, J. P. Fernandes, and J. Saraiva. “Detecting Anomalous Energy Consumption in Android Applications.” In: *Programming Languages*. Ed. by F. M. Quintão Pereira. Cham: Springer International Publishing, 2014, pp. 77–91. isbn: 978-3-319-11863-5.

[32] M. Couto, R. Pereira, F. Ribeiro, R. Rua, and J. a. Saraiva. “Towards a Green Ranking for Programming Languages.” In: *Proceedings of the 21st Brazilian Symposium on Programming Languages*. SBLP 2017. Association for Computing Machinery, 2017. isbn: 9781450353892. doi: 10.1145/3125374.3125382. url: https://doi.org/10.1145/3125374.3125382.

[33] L. Cruz, R. Abreu, and J. Rouvignac. “Leafactor: Improving Energy Efficiency of Android Apps via Automatic Refactoring.” In: *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. 2017, pp. 205–206. doi: 10.1109/MOBILESoft.2017.21.

[34] L. Cruz and R. Abreu. “Catalog of energy patterns for mobile applications.” In: *Empirical Software Engineering* 24.4 (2019), pp. 2209–2235.

[35] L. Cruz and R. Abreu. *EMaaS: Energy Measurements as a Service for Mobile Applications*. Feb. 2019.

[36] M. D. Da Silva and H. L. Tavares. *Redis Essentials*. Packt Publishing Ltd, 2015.

[37] M. Dayarathna, Y. Wen, and R. Fan. “Data center energy consumption modeling: A survey.” In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 732–794.

[38] *DB-Engines Ranking*. url: https://db-engines.com/en/ranking.

[39] S. Desrochers, C. Paradis, and V. Weaver. “A Validation of DRAM RAPL Power Measurements.” In: Oct. 2016, pp. 455–470. doi: 10.1145/2989081.2989088.

[40] P. Di Tommaso, E. Palumbo, M. Chatzou, P. Barja, M. Heuer, and C. Notredame. "The impact of Docker containers on the performance of genomic pipelines." In: *PeerJ* 3 (Sept. 2015). doi: 10.7717/peerj.1273.

[41] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux. "OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases." In: 7.4 (2013). issn: 2150-8097. doi: 10.14778/2732240.2732246. url: https://doi.org/10.14778/2732240.2732246.

[42] D. Duarte and O. Belo. "Evaluating Query Energy Consumption in Document Stores." In: *International Conference on Emerging Technologies for Developing Countries*. Springer. 2017, pp. 79–88.

[43] P. DuBois. *MySQL*. Pearson Education, 2008.

[44] I. S. Elgrably. "UMA ANÁLISE EXPERIMENTAL ENTRE SISTEMAS GERENCIADORES DE BANCO DE DADOS OPEN SOURCE." In: ().

[45] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. "An updated performance comparison of virtual machines and linux containers." In: *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE. 2015, pp. 171–172.

[46] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun. "Benchmarking in the Cloud: What It Should, Can, and Cannot Be." In: Jan. 2013, pp. 173–188. isbn: 978-3-642-36726-7. doi: 10.1007/978-3-642-36727-4_12.

[47] J. Gehrke. *Database management systems*. Mcgraw-hill Education - Europe, 2002.

[48] S. GeSI. "Enabling the low carbon economy in the information age." In: *A Report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI)* (2008).

[49] S. Gilbert and N. Lynch. "Perspectives on the CAP Theorem." In: *Computer* 45.2 (2012), pp. 30–36.

[50] R. J. F. Gonçalves. "Estabelecimento de planos de consumo energético para queries sobre data warehouses." Doctoral dissertation. 2014.

[51] R. Gonçalves, J. Saraiva, and O. Belo. "Defining Energy Consumption Plans for Data Querying Processes." In: Dec. 2014. doi: 10.1109/BDCloud.2014.109.

[52] G. Graefe. "Database servers tailored to improve energy efficiency." In: *Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data management*. 2008, pp. 24–28.

[53] M. Grambow, J. Hasenburg, T. Pfandzelter, and D. Bermbach. "Is it safe to dockerize my database benchmark?" In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pp. 341–344.

[54] J. Gray. *Benchmark handbook: for database and transaction processing systems*. Morgan Kaufmann Publishers Inc., 1992.

[55] I.-T. R. Group et al. “Top 10 Energy-Saving Tips for a Greener Data Center.” In: *Info-Tech Research Group.]* (2007).

[56] M. Guimarães, J. Saraiva, and O. Belo. “Some Heuristic Approaches for Reducing Energy Consumption on Database Systems.” In: *DBKDA 2016* (2016), p. 59.

[57] B. Guo, J. Yu, B. Liao, D. Yang, and L. Lu. “A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing.” In: *Journal of Network and Computer Applications* 84 (Feb. 2017). doi: 10.1016/j.jnca.2017.02.015.

[58] T. Haerder and A. Reuter. “Principles of Transaction-Oriented Database Recovery.” In: *ACM Comput. Surv.* 15.4 (Dec. 1983), pp. 287–317. issn: 0360-0300. doi: 10.1145/289.291. url: https://doi.org/10.1145/289.291.

[59] T. Haigh. “How Charles Bachman invented the DBMS, a foundation of our digital world.” In: *Communications of the ACM* 59.7 (2016), pp. 25–30.

[60] J. Han, E. Haihong, G. Le, and J. Du. “Survey on NoSQL database.” In: *2011 6th international conference on pervasive computing and applications*. IEEE. 2011, pp. 363–366.

[61] S. Harizopoulos, M. Shah, J. Meza, and P. Ranganathan. “Energy Efficiency: The New Holy Grail of Data Management Systems Research.” In: (Sept. 2009).

[62] J. L. Henning. “SPEC CPU2006 benchmark descriptions.” In: *ACM SIGARCH Computer Architecture News* 34.4 (2006), pp. 1–17.

[63] W. H. Inmon. “What is a data warehouse?” In: *Prism Tech Topic* 1.1 (1995), pp. 1–5.

[64] W. H. Inmon. “The data warehouse and data mining.” In: *Communications of the ACM* 39.11 (1996), pp. 49–51.

[65] I. Intel. “and IA-32 architectures software developer’s manual.” In: *Volume 3A: System Programming Guide, Part* 1.64 (64), p. 64.

[66] P.-q. JIN, B.-p. XING, Y. Jin, and L. YUE. “Survey on energy-aware green databases.” In: *Journal of Computer Application* 34.1 (2014), pp. 46–53.

[67] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. “Virtual machine power metering and provisioning.” In: *Proceedings of the 1st ACM symposium on Cloud computing*. 2010, pp. 39–50.

[68] E. Kenler and F. Razzoli. *MariaDB Essentials*. Packt Publishing Ltd, 2015.

[69] K. Khan, M. Hirki, T. Niemi, J. Nurminen, and Z. Ou. “RAPL in Action: Experiences in Using RAPL for Power Measurements.” In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 3 (Jan. 2018). doi: 10.1145/3177754.

[70] H. Khazaei, M. Fokaefs, S. Zareian, N. Beigi, B. Ramprasad, M. Shtern, P. Gaikwad, and M. Litoiu. “How do I choose the right NoSQL solution? A comprehensive theoretical and experimental survey.” In: *Journal of Big Data and Information Analytics (BDIA)* 2 (Oct. 2015). doi: 10.3934/bdia.2016004.

[71] H. Knoche. “Combining Application-Level and Database-Level Monitoring to Analyze the Performance Impact of Database Lock Contention.” In: (2016).

[72] S. A. Koçak. “SOFTWARE ENERGY CONSUMPTION PREDICTION USING SOFTWARE CODE METRICS.” Doctoral dissertation. Ryerson University, 2018.

[73] M. Kunjir, P. Birwa, and J. Haritsa. “Peak power plays in database engines.” In: (Mar. 2012). doi: 10.1145/2247596.2247648.

[74] W. Lang, R. Kandhan, and J. M. Patel. “Rethinking query processing for energy efficiency: Slowing down to win the race.” In: *IEEE Data Eng. Bull.* 34.1 (2011), pp. 12–23.

[75] N. Leavitt. “Will NoSQL Databases Live Up to Their Promise?” In: *Computer* 43.2 (2010), pp. 12–14. doi: 10.1109/MC.2010.58.

[76] Y. Li and S. Manoharan. “A performance comparison of SQL and NoSQL databases.” In: *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. IEEE. 2013, pp. 15–19.

[77] L. G. Lima, F. Soares-Neto, P. Lieuthier, F. Castor, G. Melfe, and J. P. Fernandes. “On Haskell and energy efficiency.” In: *Journal of Systems and Software* 149 (2019), pp. 554 –580. issn: 0164-1212. doi: https://doi.org/10.1016/j.jss.2018.12.014. url: http://www.sciencedirect.com/science/article/pii/S0164121218302747.

[78] D. Mahajan. “Energy efficiency analysis and optimization of relational and NoSQL databases.” In: (2016).

[79] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, and J. Clause. “An Empirical Study of Practitioners’ Perspectives on Green Software Engineering.” In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE ’16. Austin, Texas: Association for Computing Machinery, 2016, pp. 237–248. isbn: 9781450339001. doi: 10.1145/2884781.2884810. url: https://doi.org/10.1145/2884781.2884810.

[80] H. Matalonga, B. Cabral, F. Castor, M. Couto, R. Pereira, S. a. M. de Sousa, and J. a. P. Fernandes. “GreenHub Farmer: Real-World Data for Android Energy Mining.” In: MSR ’19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 171–175. doi: 10.1109/MSR.2019.00034. url: https://doi.org/10.1109/MSR.2019.00034.

[81] M. P. Mills. *The Cloud begins with coal*. Digital Power Group, 2013.

[82] T. Mohring. “Design and Implementation of a NoSQL-concept for an international and multicentral clinical database.” Dec. 2016. url: http://dbis.eprints.uni-ulm.de/1448/.

[83] A. B. M. Moniruzzaman and S. Hossain. “NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison.” In: *Int J Database Theor Appl* 6 (June 2013).

[84] S. Murugesan. “Making IT green.” In: *IT Professional* 12 (May 2010), pp. 4 –5. doi: 10.1109/MITP.2010.60.

[85] S. Pelley, K. LeFevre, and T. F. Wenisch. “Do query optimizers need to be SSD-aware?” In: *ADMS@ VLDB*. 2011, pp. 44–51.

[86] R. Pereira. “Locating Energy Hotspots in Source Code.” In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017, pp. 88–90. doi: 10.1109/ICSE-C.2017.151.

[87] R. Pereira, T. Carção, M. Couto, J. Cunha, J. P. Fernandes, and J. Saraiva. “Helping Programmers Improve the Energy Efficiency of Source Code.” In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017, pp. 238–240. doi: 10.1109/ICSE-C.2017.80.

[88] R. Pereira, T. Carção, M. Couto, J. Cunha, J. P. Fernandes, and J. Saraiva. “SPELLing out energy leaks: Aiding developers locate energy inefficient code.” In: *Journal of Systems and Software* 161 (2020), p. 110463. issn: 0164-1212. doi: https://doi.org/10.1016/j.jss.2019.110463. url: http://www.sciencedirect.com/science/article/pii/S0164121219302377.

[89] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva. “Energy efficiency across programming languages: how do energy, time, and memory relate?” In: *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*. 2017, pp. 256–267.

[90] R. Pereira, M. Couto, J. Saraiva, J. Cunha, and J. Fernandes. “The Influence of the Java Collection Framework on Overall Energy Consumption.” In: Feb. 2016, pp. 15–21. doi: 10.1145/2896967.2896968.

[91] G. Pinto and F. Castor. “Energy Efficiency: A New Concern for Application Software Developers.” In: *Commun. ACM* 60.12 (Nov. 2017), pp. 68–75. issn: 0001-0782. doi: 10.1145/3154384. url: https://doi.org/10.1145/3154384.

[92] M. Poess and R. Nambiar. “Energy cost, the key challenge of today’s data centers: A power consumption analysis of TPC-C results.” In: *PVLDB* 1 (Aug. 2008), pp. 1229–1240.

[93] C. G. F. A. Portela. “Modelação de consumo de energia em Linux.” 2016.

[94] D. Pritchett. “BASE: An Acid Alternative: In Partitioned Databases, Trading Some Consistency for Availability Can Lead to Dramatic Improvements in Scalability.” In: *Queue* 6.3 (May 2008), pp. 48–55. issn: 1542-7730. doi: 10.1145/1394127.1394128. url: https://doi.org/10.1145/1394127.1394128.

[95] B. B. Rad, H. J. Bhatti, and M. Ahmadi. “An introduction to docker and analysis of its performance.” In: *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017), p. 228.

[96] N. Rasmussen. "Determining total cost of ownership for data center and network room infrastructure." In: *Relatório técnico, Schneider Electric, Paris* 8 (2011).

[97] M. Rodriguez-Martinez, H. Valdivia, J. Seguel, and M. Greer. "Estimating Power/Energy Consumption in Database Servers." In: *Procedia Computer Science* 6 (2011). Complex adaptive sysytems, pp. 112 –117. issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2011.08.022. url: http://www.sciencedirect.com/science/article/pii/S187705091100487X.

[98] M. Rodríguez, D. Jabba, E. E. Zurek, A. Salazar, P. Wightmam, A. Barros, and W. Nieto. "Analyzing power and energy consumption of large join queries in database systems." In: *2013 IEEE Symposium on Industrial Electronics Applications*. 2013, pp. 148–153. doi: 10.1109/ISIEA.2013.6738985.

[99] H. Rong, H. Zhang, S. Xiao, C. Li, and C. Hu. "Optimizing energy consumption for data centers." In: *Renewable and Sustainable Energy Reviews* 58 (2016), pp. 674 –691. issn: 1364-0321. doi: https://doi.org/10.1016/j.rser.2015.12.283. url: http://www.sciencedirect.com/science/article/pii/S1364032115016664.

[100] R. Rua, M. Couto, A. Pinto, J. Cunha, and J. Saraiva. "Towards using Memoization for Saving Energy in Android." In: *CIbSE*. 2019.

[101] R. Rua, M. Couto, and J. Saraiva. "GreenSource: A Large-Scale Collection of Android Code, Tests and Energy Metrics." In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 176–180. doi: 10.1109/MSR.2019.00035.

[102] R. Rua, T. Fraga, M. Couto, and J. a. Saraiva. "Greenspecting Android Virtual Keyboards." In: *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*. MOBILESoft '20. Seoul, Republic of Korea: Association for Computing Machinery, 2020, pp. 98–108. isbn: 9781450379595. doi: 10.1145/3387905.3388600. url: https://doi.org/10.1145/3387905.3388600.

[103] J Saraiva, M Guimarales, and O Belot. "AN ECONOMIC ENERGY APPROACH FOR QUERIES ON DATA CENTERS." In: *PROCEEDINGS OF THE 3RD INTERNATIONAL CONFERENCE ON ENERGY AND ENVIRONMENT (ICEE 2017)*. Ed. by I Soares and J Resende. ICEE International Conference on Energy % Environment. Citations: wos. 2017, 679–685.

[104] B. Scalzo. *Database Benchmarking and Stress Testing: An Evidence-Based Approach to Decisions on Architecture and Technology*. Apress, 2018.

[105] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz. "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance." In: *Proc. VLDB Endow.* 3.1–2 (Sept. 2010), pp. 460–471. issn: 2150-8097. doi: 10.14778/1920841.1920902. url: https://doi.org/10.14778/1920841.1920902.

[106] D. Seybold, M. Keppler, D. Gründler, and J. Domaschka. "Mowgli: Finding your way in the DBMS jungle." In: *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*. 2019, pp. 321–332.

[107] S Shaw. *HammerDB: the open source oracle load test tool, 2012*.

[108] M. Stonebraker. "Object management in POSTGRES using procedures." In: *On Object-Oriented Database Systems*. Springer, 1991, pp. 53–64.

[109] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin. "MapReduce and parallel DBMSs: friends or foes?" In: *Communications of the ACM* 53.1 (2010), pp. 64–71.

[110] C. Strozzi. *NoSQL Relational Database Management System: HomePage*. http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/HomePage.

[111] B. N. Taylor. *The international system of units*. US G.P.O., 1991.

[112] "The Box Plot: A Simple Visual Method to Interpret Data." In: *Annals of Internal Medicine* 110.11 (1989). PMID: 2719423, pp. 916–921. doi: 10.7326/0003-4819-110-11-916. eprint: https://www.acpjournals.org/doi/pdf/10.7326/0003-4819-110-11-916. url: https://www.acpjournals.org/doi/abs/10.7326/0003-4819-110-11-916.

[113] S. Tiwari. *Professional NoSQL*. Wiley, 2011.

[114] Y.-c. Tu, X. Wang, and Z. Xu. "Power-aware DBMS: potential and challenges." In: *International Conference on Scientific and Statistical Database Management*. Springer. 2011, pp. 598–599.

[115] I. T. Union. *Measuring the Information Society Report 2018*. 2018, p. 201. doi: https://doi.org/http://handle.itu.int/11.1002/pub/8114a552-en. url: https://www.itu-ilibrary.org/content/publication/pub-8114a552-en.

[116] G. Vaish. *Getting Started with NoSQL*. Packt Publishing, 2013. isbn: 1849694982.

[117] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester. "Trends in worldwide ICT electricity consumption from 2007 to 2012." In: *Computer Communications* 50 (2014). Green Networking, pp. 64 –76. issn: 0140-3664. doi: https://doi.org/10.1016/j.comcom.2014.02.008. url: http://www.sciencedirect.com/science/article/pii/S0140366414000619.

[118] B. Varghese, L. T. Subba, L. Thai, and A. Barker. "Container-based cloud virtual machine benchmarking." In: *2016 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE. 2016, pp. 192–201.

[119] H. Vera, W Boaventura, M Holanda, V Guimaraes, and F Hondo. "Data modeling for NoSQL document-oriented databases." In: *CEUR Workshop Proceedings*. Vol. 1478. 2015, pp. 129–135.

[120] H. Vyawahare. "Brief Review on SQL and NoSQL." In: (July 2018).

[121] J. Wang, L. Feng, W. Xue, and Z. Song. "A survey on energy-efficient data management." In: *ACM SIGMOD Record* 40.2 (2011), pp. 17–23.

[122] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. "Measuring Energy and Power with PAPI." In: *2012 41st International Conference on Parallel Processing Workshops*. 2012, pp. 262–268.

[123] S. Weyl, J. Fries, G. Wiederhold, and F. Germano. "A modular self-describing clinical databank system." In: *Computers and Biomedical Research* 8.3 (1975), pp. 279 –293. issn: 0010-4809. doi: https://doi.org/10.1016/0010-4809(75)90045-2. url: http://www.sciencedirect.com/science/article/pii/0010480975900452.

[124] B. Whitehead, D. Andrews, A. Shah, and G. Maidment. "Assessing the environmental impact of data centres part 1: Background, energy use and metrics." In: *Building and Environment* 82 (2014), pp. 151–159.

[125] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012. isbn: 3642290434.

[126] T.-L. S. Wu. *An Overview of Present NoSQL Solutions and Features*.

[127] Z. Xu, Y. Tu, and X. Wang. "Exploring power-performance tradeoffs in database systems." In: *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 2010, pp. 485–496. doi: 10.1109/ICDE.2010.5447840.

[128] Z. Xu. "Building a power-aware database management system." In: *Proceedings of the Fourth SIGMOD PhD Workshop on Innovative Database Research*. 2010, pp. 1–6.

[129] Z. Xu, Y.-C. Tu, and X. Wang. "PET: Reducing database energy cost via query optimization." In: *Proceedings of the VLDB Endowment* 5 (Aug. 2012), pp. 1954–1957. doi: 10.14778/2367502.2367546.

[130] K. Yu, Y. Gao, P. Zhang, and M. Qiu. "Design and architecture of dell acceleration appliances for database (DAAD): A practical approach with high availability guaranteed." In: *2015 IEEE 17th international conference on high performance computing and communications, 2015 IEEE 7th international symposium on cyberspace safety and security, and 2015 IEEE 12th international conference on embedded software and systems*. IEEE. 2015, pp. 430–435.

Appendix

10 minutes Benchmark Results

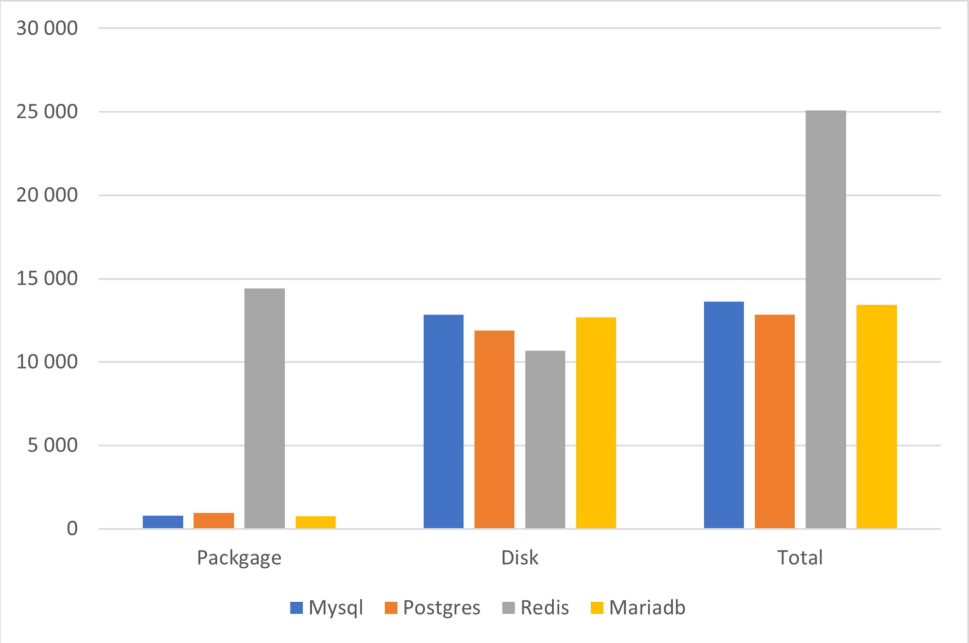
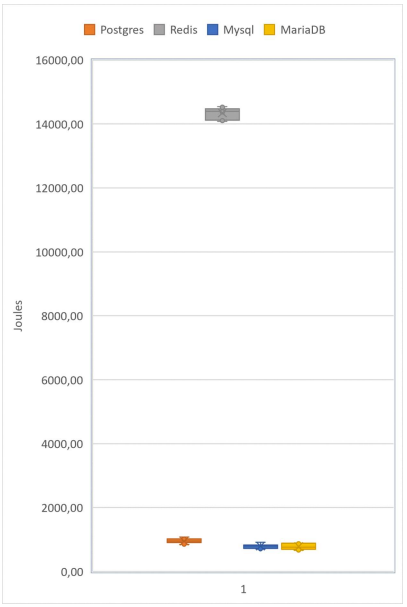
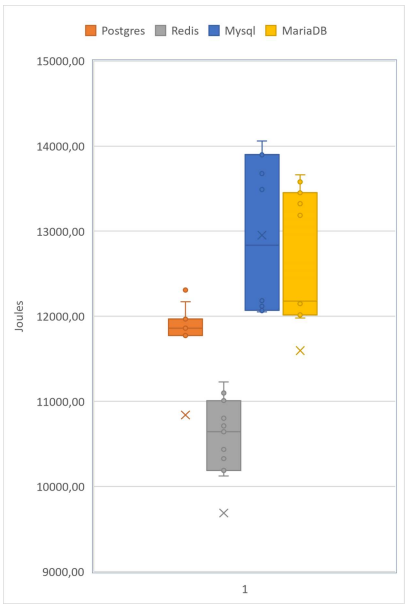


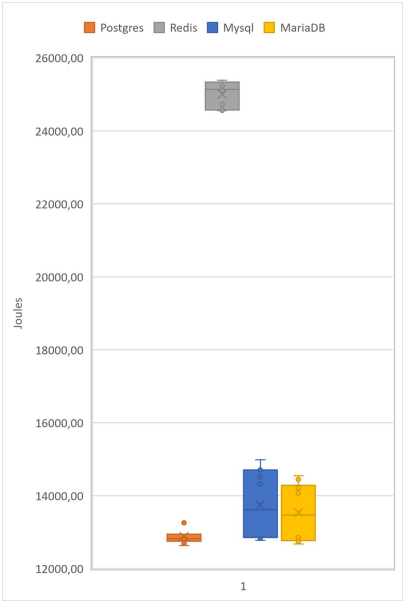
Figure A.1: Median of energy consumption on Package, Disk and Total.



(a) Distribution of energy consumed on Package



(b) Distribution of energy consumed on Disk



(c) Distribution of energy consumed on Overall System

Figure A.2: Distribution of energy consumed

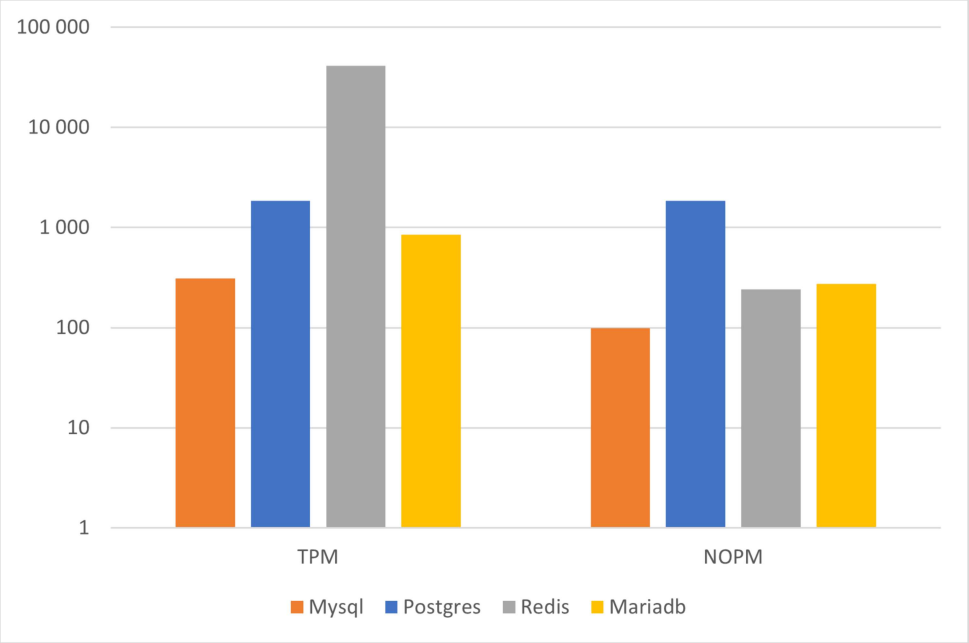
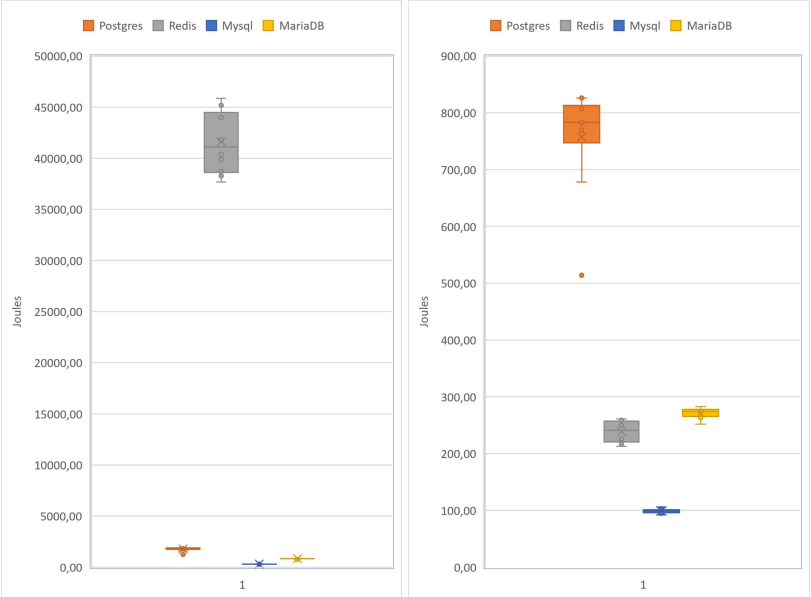


Figure A.3: Median of HammerDB metrics.



(a) Distribution of TPM (b) Distribution of NOPM

Figure A.4: Distribution of HammerDB metrics

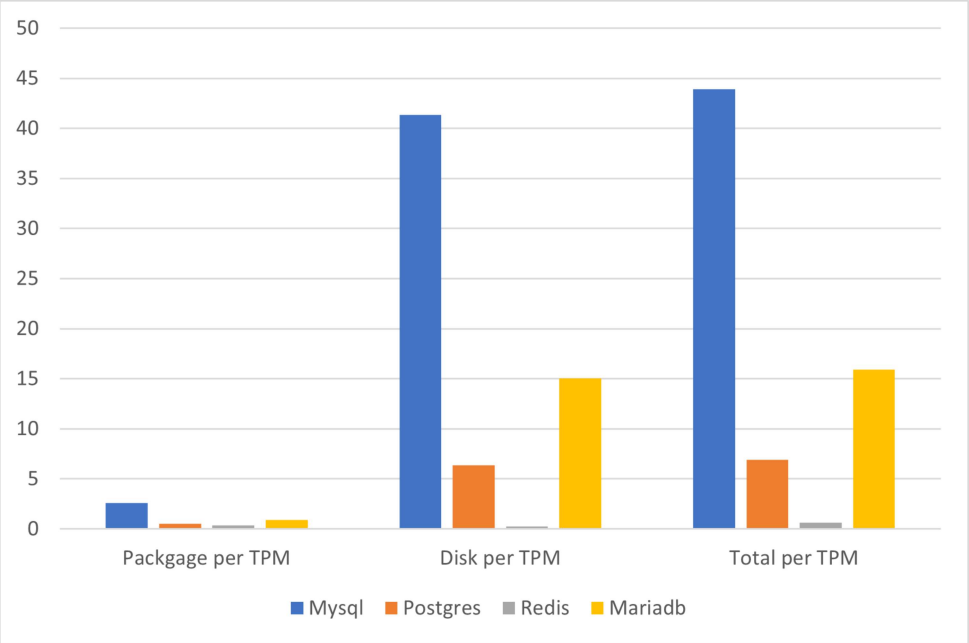
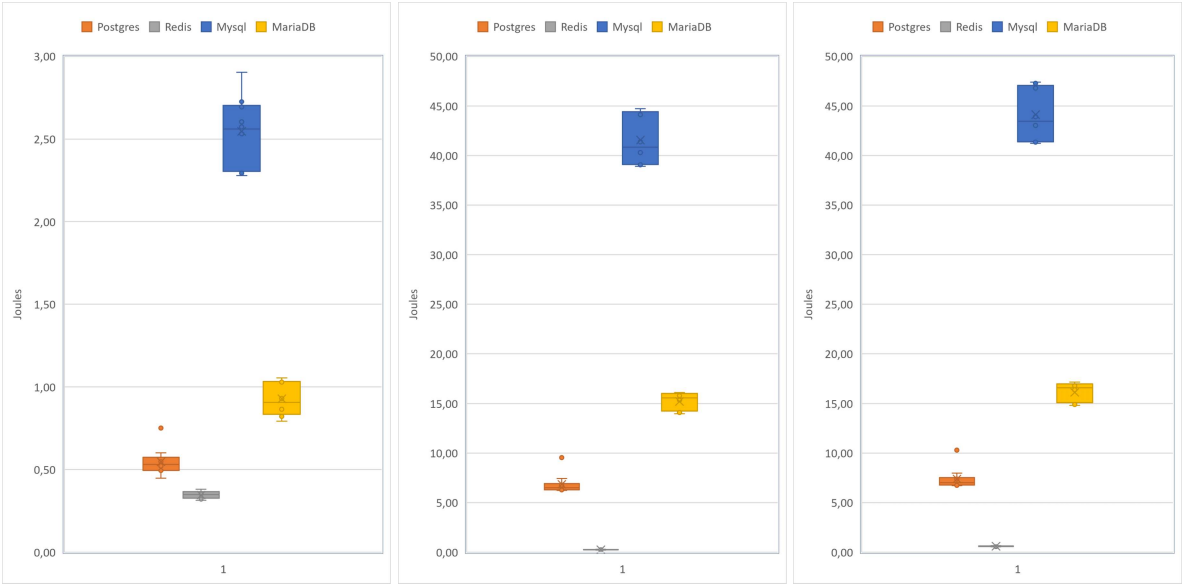


Figure A.5: Median of energy consumption per TPM



(a) Energy consumption on Package per TPM (b) Energy consumption on Disk per TPM (c) Energy consumption on Overall System per TPM

Figure A.6: Distribution Energy consumption per TPM

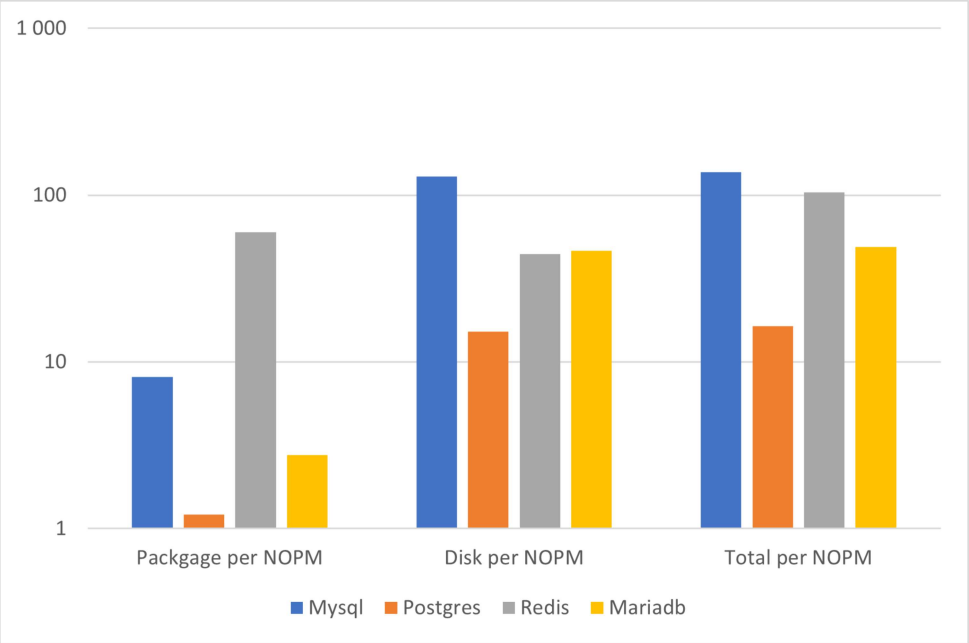


Figure A.7: Median of energy consumption per NOPM

(a) Energy consumption on Package per NOPM (b) Energy consumption on Disk per NOPM (c) Energy consumption on Overall System per NOPM

Figure A.8: Distribution of energy consumption per NOPM

Appendix B

Other Relevant Graphs

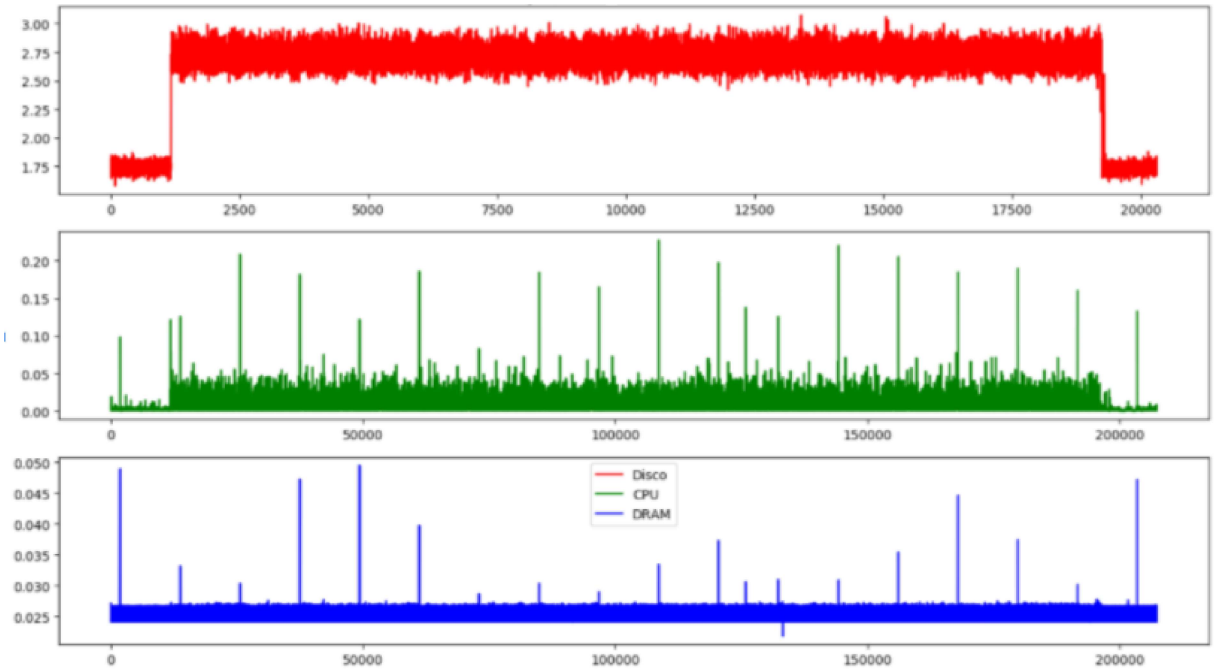


Figure B.1: MySQL energy behavior during a 5 minutes benchmark

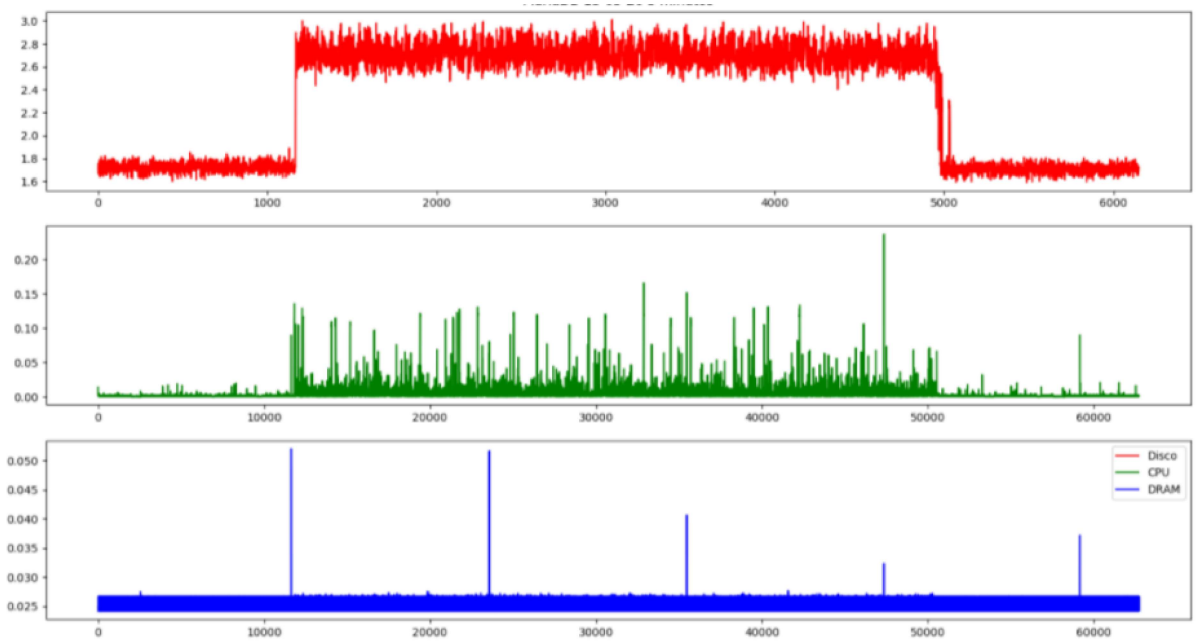


Figure B.2: MariaDB energy behavior during a 5 minutes benchmark

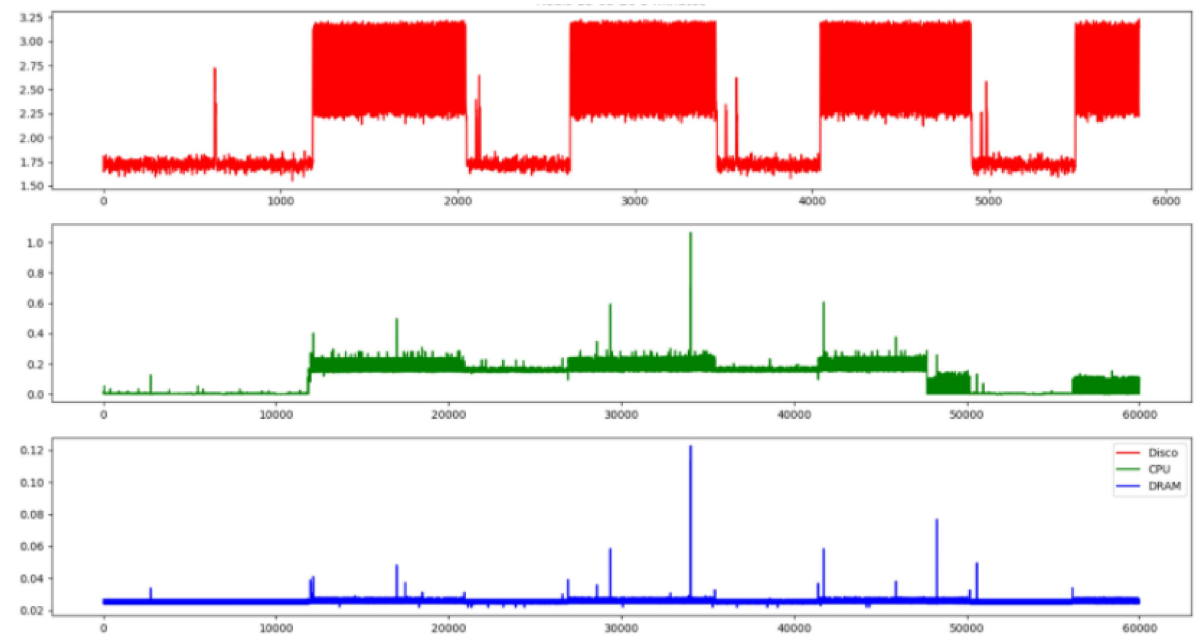


Figure B.3: Redis energy behavior during a 5 minutes benchmark

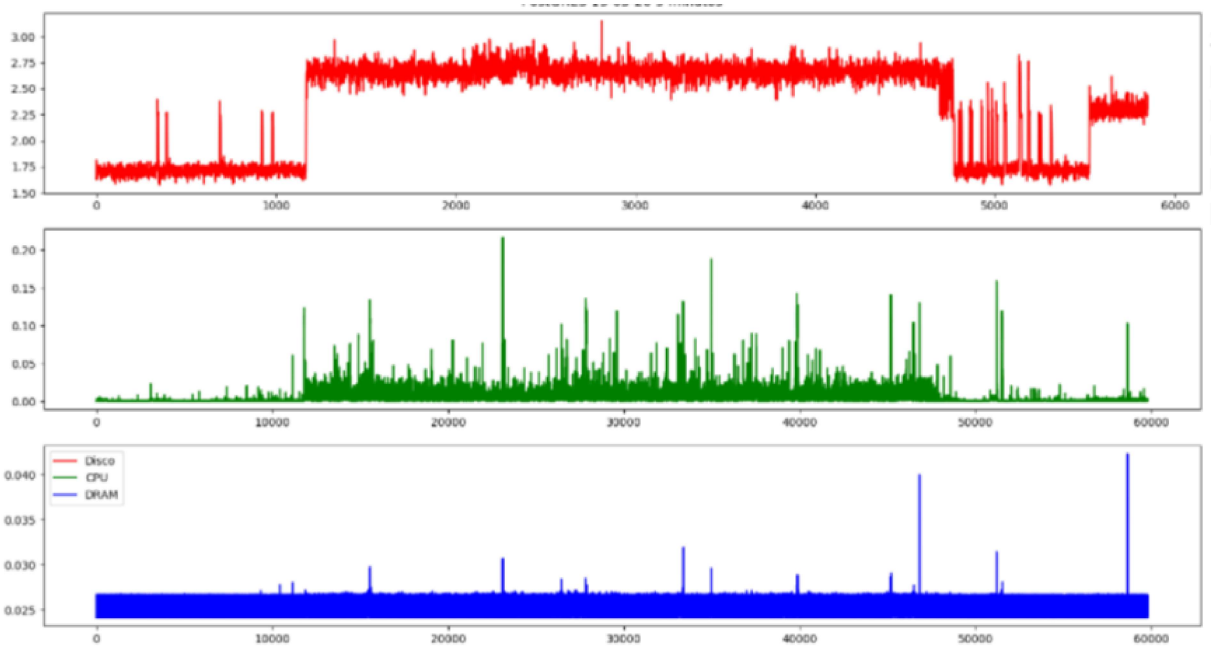


Figure B.4: Postgres energy behavior during a 5 minutes benchmark

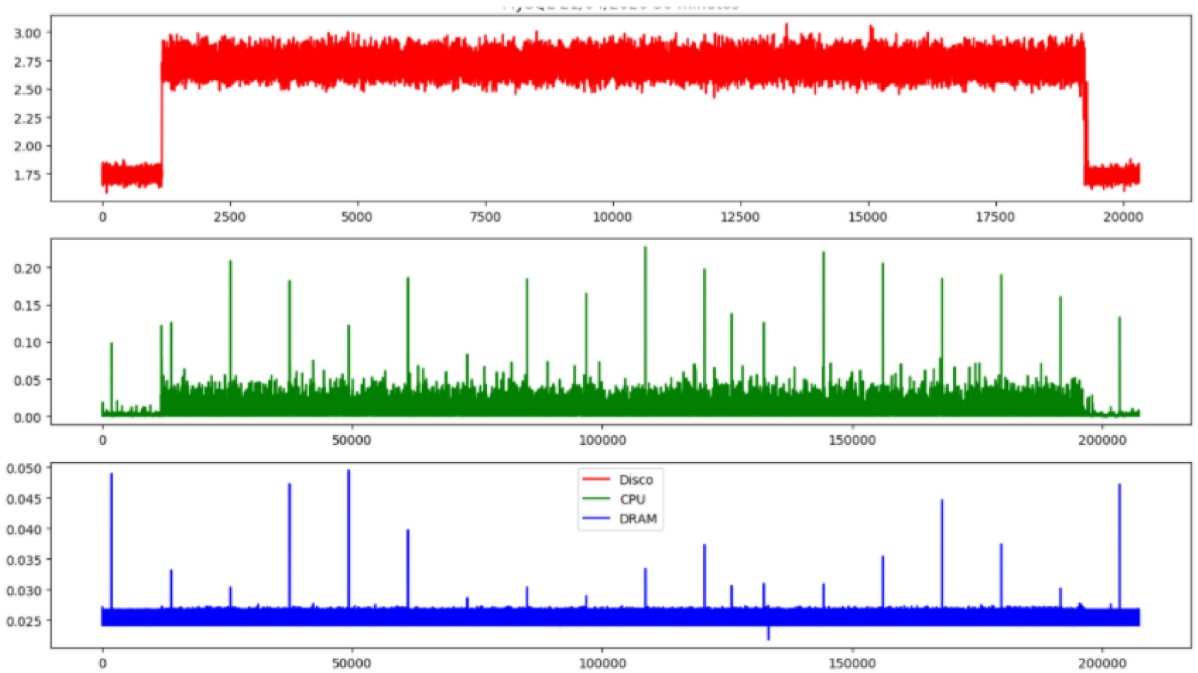


Figure B.5: MySQL energy behavior during a 30 minutes benchmark

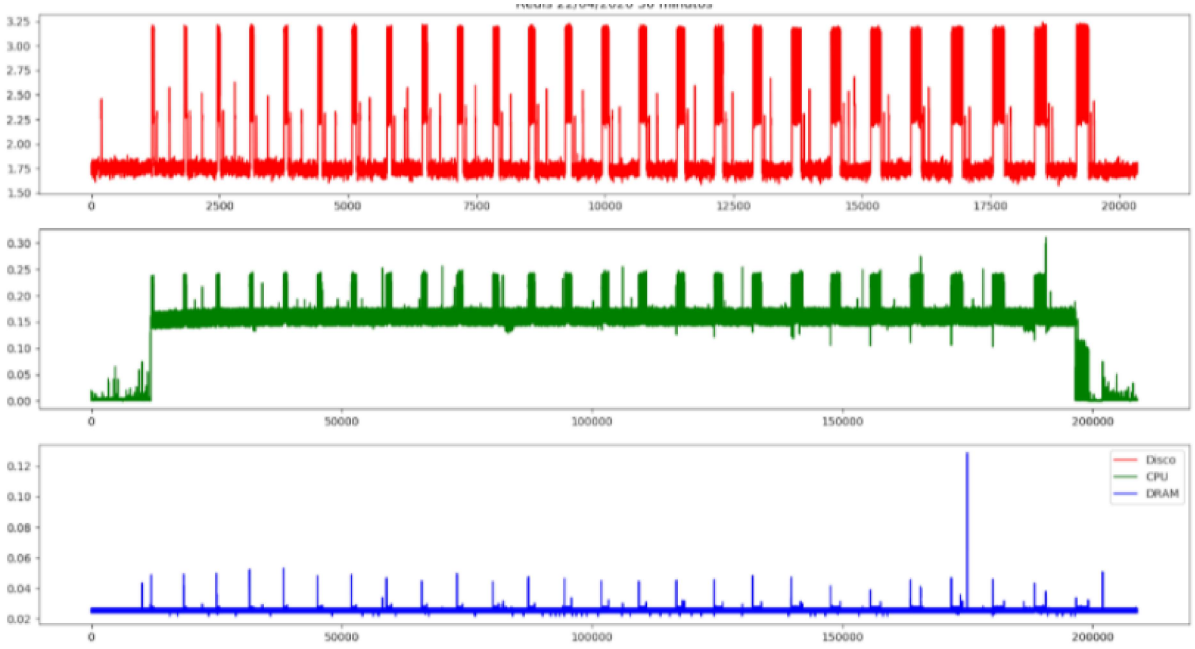


Figure B.6: Redis energy behavior during a 30 minutes benchmark

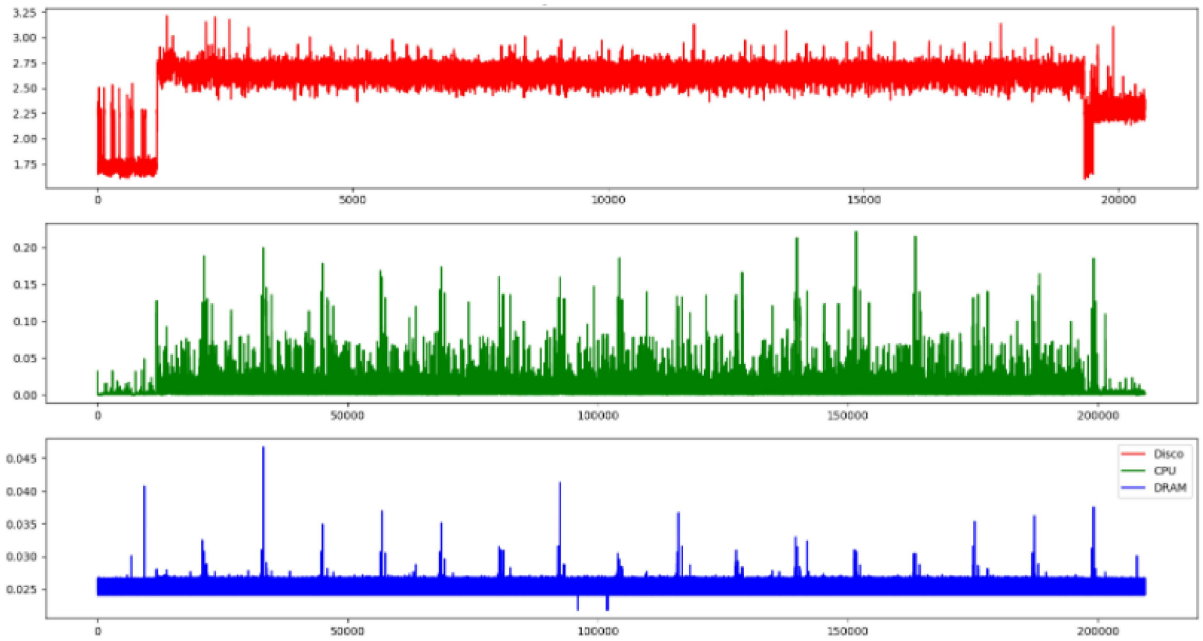


Figure B.7: Postgres energy behavior during a 30 minutes benchmark