DATABASE ENGINES: EVOLUTION OF GREENNESS

by

Zainab Al-Zanbouri

MSc Computer Science, Liverpool John Moores University, United Kingdom, 2012

BSc Computer Science, University of Technology, Iraq, 1991

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Database Engines: Evolution of Greenness

Master of Science 2015

Zainab Al-Zanbouri

Computer Science

Ryerson University

Abstract

Information Technology uses up to 10% of the world's electricity generation, contributing to CO_2 emissions and high energy costs. Data centers consume up to 23% of this energy, and a large fraction of this energy is consumed by databases. Therefore, building an energy efficient (green) database engine will reduce associated energy consumption and CO_2 emissions.

To understand the factors driving database energy consumption and execution time over the course of their evolution, we conducted an empirical case study of energy consumption of two MySQL database engines, InnoDB and MyISAM, across 12 releases. Moreover, we examined the relation between four software metrics and energy consumption & execution time, to determine the software metrics affecting the greenness and performance of a database.

Our analysis shows that database engines energy consumption and execution time increase as databases evolve. Moreover, the Lines of Code metric is strongly correlated with energy consumption and execution time.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Andriy Miranskyy, and Dr. Ayse Bener, for their continuous support of my MSc study and related research, for their understanding, patience, immense knowledge and their professional way of solving problems.

I gained from Dr. Miranskyy a significant amount of knowledge, soft skills and an amazing amount of technical information. I also learned the importance of understanding the root of a problem before starting to solve it. All these factors helped to make my period of study a very rich, enjoyable and memorable experience. I do not have enough words to express how grateful that I had the chance to work with Dr. Miranskyy.

Also, I would like to thank Dr. Bener for her guidance and valuable recommendations.

I would like to thank my two inspiring daughters, Yusur and Jana. Their patience and their wonderful company always push me to keep moving forward; because of them and for them, I am doing my best.

Finally, I would like to thank my family, especially my two amazing brothers, and friends; because of their love, support and prayers, I am here.

Table of Contents

List of Appendices

List of Tables

List of Figures

List of Abbreviations

ACID	Atomic, Consistent, Isolated, Durable
ACPI	Advanced Configuration and Power Interface
API	Application Programming Interface
b	Linear regression equation's slope value
CC	Cyclomatic Complexity
Ci	Confidence interval
CO_2	Carbon Dioxide
CPU	Central Processing Unit
CSV	Comma-Separated Values
DBGEN	Database generator utility
DBMS	Database Management System
DRAM	Dynamic random-access memory
DVFS	Dynamic Voltage Frequency Scaling
FP	Function Point
FPA	Function Point Analysis
FSM	Functional Size Measurement
GB	Gigabyte
GUI	Graphical User Interface
I/O	Input /Output
IDE	Integrated Development Environment
IT	Information Technology
KSLOC	Kilo Source Line Of Code
LCT	Life Cycle Thinking
LOC	Lines Of Code
LOCC	Lines Of Code Change
MB	Megabyte
MCC	Modified Cyclomatic Complexity
MSR	Mining Software Repository
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
ρ	Pearson correlation coefficient
QEP	Query Execution Plan
QGEN	Query Generator Utility
R2	Coefficient of determination
RAM	Random-access Memory
RDBMS	Relational Database Management System

ReinDB	Renewable Energy Integrated Database
SEI	Software Engineering Institute
SLOC	Source Line Of Code
SQL	Structured Query Language
TAR	Tape Archive
TCC	Traditional Cyclomatic Complexity
TPC	Transaction Processing Performance Council
TPC-H	Ad-Hoc, Decision Support Benchmark
TWh	Terawatt Hours
USB	Universal Serial Bus
Wh	Watt-hour

Chapter 1

1 Introduction

1.1 Motivation

Information Technology (IT) requires a huge amount of energy. In fact, 1500 Terawatt hours[1] (TWh) per year (or 10% of the worldwide energy generation) are consumed by IT [1]. Consequently, this leads to high carbon dioxide (CO_2) emissions. The current global CO_2 emission of 9.1 billion tons per year is at the highest level in human history (49% higher than in 1990) [1].

As a result of the high demand on IT services, data centers consume a massive amount of energy and are responsible for using up to 350 TWh per year [1]. Generally, the energy is consumed by hardware; but since hardware is driven by software, software is responsible for consuming energy as well. Databases are considered to be the backbone in the software world; hence, they are responsible for a significant proportion of the overall software energy consumption. Therefore, we focus on understanding how to make databases more efficient (greener) by trying to identify the main factors that affect database energy consumption and execution time.

IT, in general, and software, in particular, can participate in the development of sustainability[2] in different ways, e.g., using energy efficiently via decreasing the used resources and as a result of that the CO_2 emissions will be reduced as well. Moreover, IT processes can be made more sustainable by decreasing the energy consumption and the negative emissions of companies and individuals.

There is no doubt that reducing the energy consumption and the related carbon emissions of IT systems contributes to and pushes the development of environmental sustainability forward. In addition, the reduction of energy consumption leads to a reduction of maintenance

[1] Terawatt hour = 10^{12} watt hours.

[2] Sustainability research focuses on areas such as the environment, economy, community and the impact on the human beings.

expenses and the total costs of ownership, giving a competitive advantage to manufacturers of such products; this important goal forces the companies to implement energy efficient products and energy-aware technology services. As a result, green[3] hardware designs and products, as well as green service operations, have gained much importance in attaining environmental sustainability. In terms of sustainability requirements, both energy and time need to be considered, and this is the main reason we focus on these two factors in this dissertation.

The software part has received less attention than the hardware part and few solutions (such as energy test suites) have been put forth to comprehend energy efficiencies. Green IT is the study and practice of efficient use of computing resources to decrease the negative impact on the environment; it is applicable to various high-tech domains, such as data centers, mobile computing and embedded systems [2][3].

Software systems usually evolve over time for different reasons such as, requirements changes, code optimization, and defect patches. As a result, conducting research for specific software will be more valuable if this research takes into consideration these evolutions and covers as many software releases as possible; therefore, that is precisely what we have done in this dissertation.

1.2 Research Statement

We aim to identify how the energy consumption and execution time of a database engine can change when the database evolves from one release to another, in addition to understanding how these changes are related to some database-associated properties such as raw data size, engine storage type (MyISAM/InnoDB), and database memory size.

Furthermore, this research searches for the software metrics that can have direct effects on the sustainability development in a database represented by its effect on energy consumption and execution time.

Defining these subjects can help to move the industry forward toward the green database goal particularly and green software generally.

[3] Green refers to the consumption of less energy.

1.3 Objective

In order to tackle the research statement, we focus on answering two research questions:

RQ1: How does the energy consumption and execution time of a database engine change as the product matures (from release to release)?

RQ2: Which software metrics affect energy consumption and execution time?

Answering RQ1 will help us identify the factors leading to green and fast database s from the software evolution perspective. Answering RQ2 will help us build models that can predict software greenness and performance based on software metrics that can be easily extracted from source code (such as code size or code churn metrics). This information should be of interest to practitioners, since software vendors such as Apple, IBM and Microsoft are seeking more sustainable products with lower levels of energy consumption and lower levels of execution time as well [4]. It should also be of interest to theoreticians, since the information can be used to build universal models of software's energy consumption and performance.

1.4 Proposed Solution

In order to answer these research questions, in this dissertation, we study energy consumption and execution time across 12 different releases of two database engines (MyISAM and InnoDB) of the MySQL database. MySQL is the most commonly used and most popular open source database in the world [5]. We chose MySQL as our software under study, because the MySQL database is a mature product (having been developed since 1995) with a large (approximately 1 million lines of code) codebase being actively developed. This gives us enough data to study the product's evolution.

There are different types of MySQL database engines. We are focusing on MyISAM and InnoDB, because these are the default engines for the various MySQL versions under study. We are investigating the effects of software changes on energy use and execution time.

To answer RQ1, we study the relation between energy consumption (or execution time) for all the MySQL versions under study. To answer RQ2, we examine the relation between software metrics from one side and energy consumption (or execution time) from the other side.

This type of work required building a framework to automate all the necessary processes such as database installation, upgrade from version to version, executing the workload, reading and collecting the measurements from the special measurement meter and recording the measurements for all the MySQL releases used in addition to creating a database for all the experimental data results. It also required building a system to extract software metrics[4] from the code base of MySQL so that the relation between the metrics on one side and energy consumption (or execution time) on the other side could be established. All these requirements are addressed in this dissertation.

1.5 Novelty and Significance

This is the first study examining the relation between different MySQL database releases and their energy consumption as well as execution times. This research differs from the previous research that has examined the link between different application versions (not a database's software) and their performance. In addition, to the best of our knowledge, this work is the first study to establish a link between MySQL databases' energy consumption and their execution time from one side and the software metrics LOC, LOCC and TCC/MCC from another side.

Our research focused on the database engine that is considered to be the mainstay in the software world, and the data we used to populate our databases (1GB and 3GB data) are considered to be Big Data compared to the size and properties of the machine used; therefore, this research is considered highly valuable to many other research areas such as Big Data and cloud computing studies as well.

Moreover, this work has a significant environmental impact and may be considered very useful in environmental research because it shows how to identify the better performance applications that use less energy and have lower execution time. Consequently, this will help in choosing software that consumes fewer resources and accordingly has a direct reduction on system CO_2 emissions.

[4] Such as LOC (Line of Code), LOCC (Line of Code Change), and TCC/MCC (Traditional/Modified Cyclomatic Complexity).

From another point of view; this work can be considered an extension to the "Green mining research" [4] in the research on power consumption within software versions; the previous research used a different application (Firefox web browser), while we are focusing on the MySQL database.

Furthermore, this work has industrial effects because vendors are interested in Green IT—for example Microsoft, Apple, Intel and IBM—and they have already started their investigations and research in power utilization documentation and tools [4].

1.6 Contributions

The contributions of our research are as follows:

1. We introduced a methodology to examine the energy consumption and execution time in two database engine types (MyISAM and InnoDB). We showed how the energy consumption and execution time vary between different database versions, among different raw data sizes and different memory sizes.
2. We introduced a case study of energy consumption and execution time within MySQL's different versions. Our results confirm the following:
    a. Energy consumption and execution time does change over the lifetime of a product.
    b. We found out that in one MySQL database engine (MyISAM), both energy consumption and execution time increase when the MySQL versions age. In essence, newer releases are slower and less green than the older ones. However, in the other case (InnoDB), the relation between the database aging and its greenness is not pronounced.
    c. Not all energy consumption is related to the hardware; there are many software issues that can affect the energy consumption and execution time and consequently the software's performance.
3. We presented a new procedure that was used to build a framework for automating all the related processes. The designed framework has the following functions:
    a. Decompress the downloaded MySQL TAR files.

b. Install a specific MySQL version.
c. Upgrade from previous version to current one.
d. Execute the 22 TPC-H queries one by one.
e. Calculate the individual query execution time.
f. Measure the energy consumption and system statuses per each query.
g. Record all the measurements to build a centralized database that represents the datasets we used in the data analysis subsequent sections.

4. We wrote scripts to find the metrics' measurements for each database version under study by using different tools, such as the CLOC tool [6]. Then we observed the results with our previous workload's energy (time) results. We found that some software metrics have a strong correlation with energy (time) consumption and some not. Our findings, at the workload level, are the following:
    a. LOC is strongly correlated with energy consumption and execution time, which means when the release has more code, it will consume more energy and hence be less green.
    b. LOCC is weakly correlated with energy consumption and execution time.
    c. TCC/MCC has a moderate to strong correlation with energy and time consumption.

Moreover, we explained the correlation between the software metrics as follows:

d. LOC is strongly correlated with MCC and TCC.
e. MCC and TCC are perfectly correlated.
f. LOCC is weakly correlated with the other metrics under study.

In addition, we calculated two new variables — namely, change in energy and change in time — and examined the correlation with LOCC as follows:

g. LOCC has a moderate to strong correlation with time and energy consumption changes in the case of MyISAM, while this correlation is weaker in the case of InnoDB.

1.7 Organization of Chapters

The rest of the dissertation is organized as follows. Chapter 2 introduces the background knowledge necessary for this research along with a general description for the ideas behind software metrics and their importance. Chapter 3 explains the methodology and experimental setup. Chapter 4 discusses the results of our experiments and threats to the validity of our study. Finally, Chapter 5 concludes the dissertation.

Chapter 2

2 Background

This chapter includes information about the background and work related to our research. In Section 2.1, we described green and sustainable computing and related information such as green software and green databases. In Section 2.2 and 2.3, we described the dissertation-related research and the latest studies concerning our work. Section 2.4 describes software metrics.

2.1 Green and Sustainable Computing

Green and sustainable computing deals with the research and practices that have a significant impact on the environment [7]. There are many concerns related to this field, such as increasing energy costs and the reduction of natural resources. Although the huge amounts of computer production and usage have a direct negative impact on the environment, there are many good steps toward green computing goals, such as lower energy consumption and new recycling programs in addition to the application of new alternatives to the use of hazardous materials in the manufacturing process [7][8].

Many researchers have been attracted to this domain because of its significance and direct impact on the economy, natural resources, energy consumption and, moreover, its effects on human lives [7]. The importance of this field began with the significant expansion in the field of computing in 1990. The U.S. Environmental Protection Agency launched Energy Star, which is a labeling program intended to support and identify energy-efficiency in, for example, monitors and climate control equipment [7]. Energy Star was applied to many technological devices such as computer monitors and heat control system devices like air conditioners. Consequently, this led to the spread of the main result of green computing, which is the sleep mode function in many of these electronic devices [8]. The sleep mode idea is to put electronic devices into standby mode after a certain period of time has passed with no user activity detected. The next step in green computing improvement began to cover many other aspects, such as client solutions, energy cost measurements, virtualization practices, e-Waste and many other solutions [8].

2.1.1 Green and Sustainable Software

There are many negative impacts on the environment as a result of using IT hardware-software systems due to increasing energy consumption and, consequently, rising carbon dioxide emissions [8]. Therefore, software is considered green and sustainable when its direct and indirect negative impacts on the sustainability process are at a minimal level or could have a positive impact [9]. This definition needs to be considered at all stages of the software lifecycle: from development, design, deployment, usages and ending with disposal. Moreover, software plays a significant role in the life cycles of many other products or services, because it can be used in product design and production processes or even in the utilization of other products or services [9]. All these factors have made the study of the relationship between software and sustainability very significant.

2.1.2 Green and Sustainable Database

Understanding the idea of the green database starts with a consideration of the basic components of the database and then examining the effects of each of these parts on the progress of the sustainability within the database. The database system is a general software system for manipulating databases. The main functionality for this system is to store information and allow the users to create, delete, update and read that information. It allows the concurrent use of a database by multiple users and provides a tool for accessing and manipulating the data in the database. A Database Management System (DBMS) is a group of programs that allows the users to create and maintain a database. It is a general-purpose software system that services the process of defining, creating, and manipulating the data [10]. The database system environment includes many interrelated components, such as

- Software,
- Hardware,
- People,
- Procedure, and
- Data.

All of these combined elements contribute to the effects on the performance and efficiency of the database, and consequently they affect the sustainability development within the database. Figure 2.1 shows these significant components that make up the database structure.

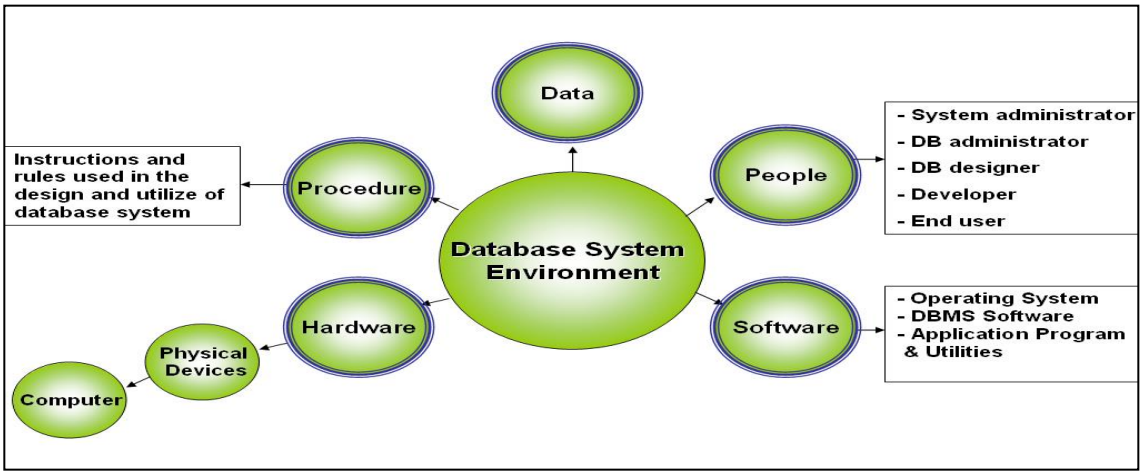


Figure 2.1: Database system environment.

2.1.2.1 Sustainability in a Database's Hardware

The most significant part in the hardware system of a database is the database server. Therefore, understanding the power consumption in the server will help to give a clearer idea about how this important part affects the sustainability development in the whole database.

In any hardware-software system and, in particular, in a database system, the total power consumption of a computer is the sum of power consumption of hardware components, such as CPU, memory, hard drives, etc. [11]. The power consumption of any hardware element depends on the settings of that hardware, such as the maximum frequency for CPU and the runtime utilization [12], and there are different methods for changing the hardware settings:

- DVFS, which refers to Dynamic voltage/frequency scaling on both CPU and DRAM.
- Changing the hard disk settings to a multi-speed option.

Several settings of the hardware cannot be changed at runtime. Most of the modern servers are not energy efficient [13], because the power consumed while the server is idling may represent the biggest portion of the server's total power consumption (see Figure 2.2). For instance, it can be more than 50% of the energy consumption of a database server [11], [14].

ACPI S3 (Advanced Configuration and Power Interface) is commonly used to decrease the idle consumption; however, using S3 has the drawback that it could be the reason for the latency of the server to restart and respond [12].

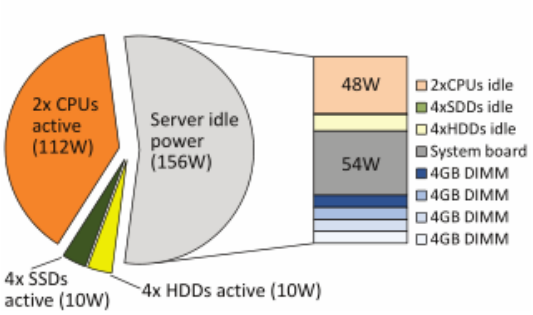
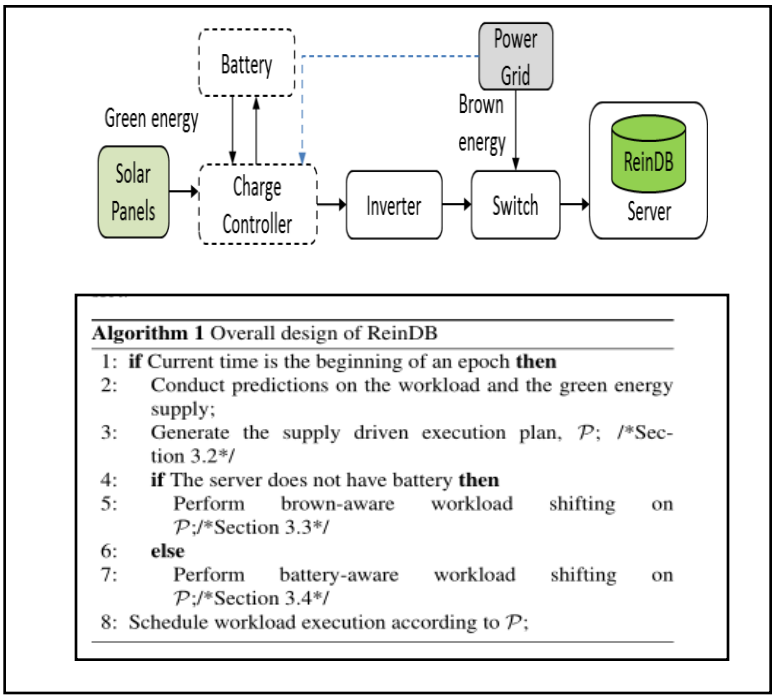


Figure 2.2: Power breakdown of test machine (reproduced from [11]).

The other significant hardware element that affects the power consumption is the battery. The processes of charging and discharging a battery are not simple. Therefore, this issue attracts the attention of many researchers [15]. The charge process leads to a loss of energy implicitly, and this loss mostly increases in many batteries over the charging cycles. In the discharge process, the battery cannot discharge completely, because of the inverter, which disconnects the battery supply when voltage is lower than a certain threshold [12].

2.1.2.1.1 Renewable Energy Integrated Database (ReinDB)

The information in this section was adopted from the original source paper of the ReinDB design [12]. Carbon-intensive energy is referred to as “brown energy,” in contrast with “green” or renewable energy [16]. The main goal of ReinDB is to reduce the brown energy consumption on a database server using both green and brown energy supplies. In this design, a developed green supply driven execution paradigm and adaptive power management techniques to adapt to the green energy supply are used for this implementation. The writers consider this work to be the first work of attempting to integrate green energy into a database server. Figure 2.3 represents the ReinDB design and the algorithm used in the design of this energy-aware database.



Algorithm 1 Overall design of ReinDB

```
1: if Current time is the beginning of an epoch then
2:     Conduct predictions on the workload and the green energy supply;
3:     Generate the supply driven execution plan, P; /*Section 3.2*/
4:     if The server does not have battery then
5:         Perform brown-aware workload shifting on P;/*Section 3.3*/
6:     else
7:         Perform battery-aware workload shifting on P;/*Section 3.4*/
8: Schedule workload execution according to P;
```

Figure 2.3: ReinDB runs on a solar-powered database server (reproduced from [12]).

2.1.2.2 Sustainability in a Database's Software

As discussed in the previous sections, software is considered to be one of the most important components in the database system environment. Therefore, in order to identify the features that lead to a green database, we first need to know the features that lead to green software. In the following section, we introduce a model (GREENSOFT model) [9] that is considered to be a pattern that can be used to apply and measure the sustainability requirements in any software system. All the related information about this model is adapted from the model's main source paper [9].

The GREENSOFT model: This model is a theoretical reference model that consists of all the necessary stages in the product life cycle model for software products, sustainability metrics and the criteria for software and software engineering extensions to sustainably simulate software design and development [9].

Moreover, this model has procedure examples for different beneficiaries and recommendations for action in addition to tools that support beneficiaries in developing, purchasing, supplying and using a specific software in a green and sustainable manner.The main

purpose of this model is to help different kinds of users, such as software developers, administrators and software end users, in creating, maintaining and using software efficiently. Figure 2.4 shows this model with its main components [9].

The main parts in this model are as follows:

- A Life Cycle of Software Products

This stage is in contrast to the traditional life cycles of software provided to Life Cycle Thinking (LCT), which has the aim of assessing the environmental, community, human, and economic consistency of the product during its complete life cycle. It starts with the early steps of product development and ends with the product's disposal and recycling. The results obtained from these assessments can then be used for a balanced improvement of the product or for comparing a product with its rivals.

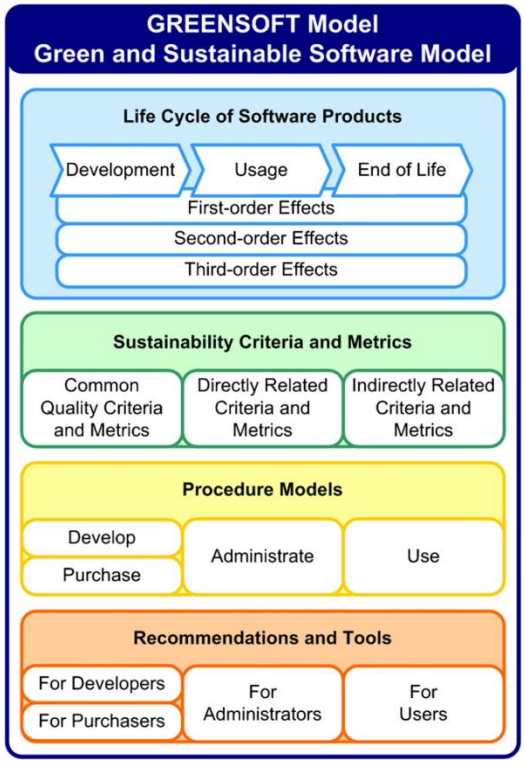


Figure 2.4: The GREENSOFT Model, a reference model for "Green and Sustainable Software" (reproduced from [9]).

- Sustainability Criteria and Metrics

This is the second part of this model; it covers the general metrics and criteria for measuring the software quality, and it allocates a classification of criteria and metrics for evaluating a software product's sustainability. Suitable standards and metrics may cover models for measuring the software's fineness and procedure models for developing the software in addition to methods borrowed from Life Cycle Assessment. In this part, they distinguish between direct criteria and those that are indirect, concerning sustainability.

- Procedure Models

This component focuses on the procedure models that wrap the achievement and development of software, the maintenance of IT systems and user guidelines. For example, they provide a generic extension for the ambiguous software development processes that allow the systematic consideration of sustainability characteristics during software development.

- Recommendations and Tools

This is the last part of this model, and it represents the support provided to interested parties with different professional levels to apply green or sustainable methods, in general, in the development, purchase, administration and use of the software products. Moreover, there are different roles covered in this part such as software administrators and developers in addition to regular and specialized users.

2.1.2.3 Sustainability in a Database's Data

The database system environment diagram (Figure 2.1) illustrates the existence of the data as one of the most important components in the database system. Additionally, we can consider the data to be the backbone in the database system; as a result, there is no doubt that these data have a significant impact on green computing and sustainability development. Data have become the raw material of production, a new source of immense economic and social value.

- Big Data and Green Database

These days, the amount of data that is generated every two days is assessed to be 5 exabytes[5] [17]. This amount of data is equivalent to the amount of data been created from the beginning of time up until 2003 [17]. In addition, it was assessed that 2007 was the first year in which it was not possible to store all the data that was produced during that year [17]. As a result, Big Data need an appropriate database for storage and manipulation.

The main concerns about Big Data from the viewpoint of green computing are as follows [7] [12]:

- Data security and data protection.
- Data storage and saving.
- Data manipulating, searching and analyzing.
- Filtering the unused and unwanted data.
- Data backup and recovery.

A Big Data database needs to have the following characteristics [18] [10]:

- A scalable database engine, in contrast with the classic DBMS (Database Management System),
- SQL (Structured Query Language) for data description and manipulation,
- ACID (Atomic, Consistent, Isolated, Durable) transactions to confirm data consistency and integrity and
- A highly available and fault-tolerant system.

The Greenpeace report "How dirty is your data?" [19] Reported that 1.2 zettabytes[6] of digital information have been generated by Twitter, Facebook—where over 30 billion pieces of content are shared each month, emails, YouTube and other social media.

Cloud computing provides solutions for many issues related to Big Data and sustainability. Cloud computing describes data processing operations that are stored to server

[5] 1 Exabyte = 1 billion gigabytes.

[6] 1 zettabyte = 10^{21} bytes.

farms rather than being powered onsite. This method makes the software that is carried over the Internet to appear like the software operating on a PC and is reachable from any computer in the world. Accordingly, various IT analysts are expecting the demise of the PC, while others think it will just become another device for contacting the online world. Progressively, the engine of the IT sector is comprised of large-scale data servers that are pushing the cloud computing revolution forward. Regarding energy, cloud computing decreases the cost for users, in addition to reducing greenhouse gas emissions, by aggregating data and workloads on single, fast, multi-tenant machines [20].

In order to store and process these zettabytes of data, the Cloud solutions rely on database engines for persistent storage. Therefore, it is important to understand what factors affect database greenness, so that the solutions using database engines become greener as well.

2.1.2.4 Sustainability in Database Utilization

2.1.2.4.1 Efficient Query Execution

The traditional query optimizer of the DBMS focuses only on the time cost of a unit of work in a database with no concern for the power consumption. The time cost of a query plan is the total sum of the holding times of all system resources such as CPU, disks and communication channels [21]. This matter gave researchers the opportunity to redesign the query optimizer in a way to enable it to produce the power efficient query execution plan (QEP). In order to create a power-efficient QEP, the query optimizer should be able to estimate the power consumption in that plan. The suggested steps for this useful plan are as follows:

- Design and construct a power model to anticipate the power consumption of QEP.
- Start the query plan set[7] that contains the power optimal query plan and the algorithm to produce the possible query plan set.
- Seek through the possible query plan set to find the power-efficient query plan.

[7] The possible query plan set is all the possible query plans for a query task [21].

Since the time cost is the main concern in traditional performance driven query optimizers, the best QEP is a query plan that uses the least amount of time. In order to calculate the time cost of a query, the total times of all system resources, including CPU, disks and communication channels, needs to be found [21] .

Relating to sustainability requirements, we have to consider both power and time, since the time optimal query plan may not be the most energy efficient one. When the query optimizer computes the query plan time cost, it takes into account the I/O time. The optimal query execution plan is a query plan with the lowest I/O operations and the power consumption of CPU is greater than that of the storage systems [22], therefore relating to power-efficiency, the traditional optimal query execution is not be considered the best query plan. The authors in [21] first expected that their invented query optimizer could produce a power optimal query plan, and later they proved the accuracy of their power model and the effectiveness of their algorithm. Figure 2.5 summarizes the process relating to the work of [21].

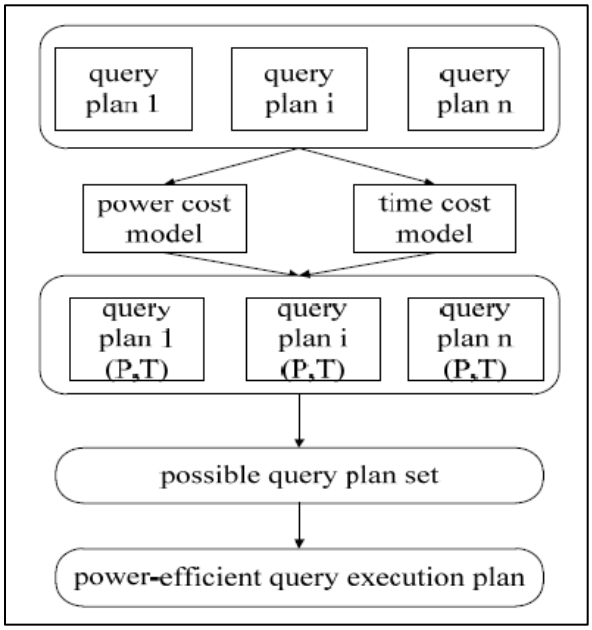


Figure 2.5: Process to obtain power-efficient query execution (reproduced from [21]).

2.2 Related Work

The energy consumption and, consequently, the energy measurement, is considered a significant concern for many vendors. In addition, the research in this area has a direct relation with the industry field. Moreover, the customers and users are interested in managing the power in their devices. Green computing represents a step towards power-aware computing and, consequently, the efficient management of power consumption.

A number of researchers have focused on energy consumption in IT. Delaluz et al. [23] conducted a comprehensive study of software and hardware systems to determine the benefit of the DRAM mode control abilities for energy savings. They addressed an essential issue in energy saving for mobile and computing environments by specifically concentrating on the memory system, which consumes around 90% of the complete energy consumed by the system when ignoring input/output processes [23].

Some researchers have concentrated on the idea of benchmarking and examining power measurement. Asmel et al. [24] described a tool that approximates the energy consumption of software in order to help concerned consumers make knowledgeable decisions about the software they use. Gurumurthi et al. [25] introduced a complete system power simulator that represents the CPU, the hierarchy of memory and a low-power disk subsystem and calculates the power performance of both side applications and the operating system.

Tiwari et al. [26] presented the power usage of a single CPU. They defined an assessment-based instruction-level power analysis method, which provides an accurate and practical way of measuring the power cost of software and describes an assessment-based instruction-level power analysis method that makes it possible to effectively analyze software power consumption. Mittal et al. [27] presented an energy simulation tool that allows developers to estimate the energy use for their mobile apps on their development workstation itself.

There are several studies about the power consumption of devices. Bircher et al. [28] produced power models for the complete system depending on processor performance events. Greenwalt et al. [29] measured and modeled the power consumption of hard drives. The hard disk state model provides both the quantitative data and insight necessary to design an efficient power management system. Stemm et al. [30] studied two types of optimization (namely,

transport-level and application-level) of network interfaces to decrease their energy consumption.

Li et al. [31] performed a quantitative analysis of the costs and benefits of spinning down a disk drive as a power management technique. The main idea behind the power consumption measurement movement is to be followed by suggestions or actions taken in order to find solutions to any undesirable outcomes. Selby et al. [32] applied methods to analyze the relationship between global variable usage and the efforts required by software maintenance and examined the effects of optimizations upon power usage. Fei et al. [33] employed source code change techniques to decrease the energy overheads accompanying application/OS connections and modified the source code changes and compiler optimizations in order to reduce power usage. Feng et al. [34] introduced a framework for studying the power-performance efficiency of the NAS parallel benchmarks on a 32-node Beowulf cluster.

Researchers have studied the performance of database engines, but not their energy consumption. For example, Shange et al. [36] investigated change in performance over multiple versions of a database engine (PostgreSQL) and a data processing framework (Hadoop). The closest to our work are [4, 37]. Gupta et al.'s [37] study focused on combining Mining Software Repository (MSR) [35] techniques with power performance and presented the first study from a software engineering perspective on energy awareness problems. The authors of [37] introduced a method for gathering and analyzing power data on mobile devices running Windows Phone 7. Their methodology describes and quantifies power consumption, detects differences in power consumption and predicts power consumption. The work by [37] is complementary to ours, because it focuses on examining the power consumption in different modules (a module is a part of a program) within the same software (Windows Phone 7) and finding which module consumes the most power. Moreover, it focused on finding the typical energy shape patterns of certain modules. We, on the other hand, are focusing on multiple versions of the same product (MySQL). Additionally, we concentrate on understanding the relation between energy consumption (or execution time) and the product development of MySQL.

Hindle [4] demonstrated combining the MSR research and energy consumption by studying multiple versions of the Firefox web browser regarding characteristic energy consumption patterns of multiple modules of the web browser. He also examined the relation

between the LOC and LOCC software metrics and energy consumption. This work is complementary to ours, because we focus on a different product (database instead of web browser) and study the effect of multiple software metrics (LOCC, MCC and TCC in addition to LOC) on energy consumption and execution time.

The works cited above demonstrate the significance and importance of the study of power consumed by software in various areas of IT.

2.3 Examining Power Consumption across Versions

The effect of software on power consumption has been overlooked [24, 26] as a result of the assumption that in software engineering we can use all the existing resources. This assumption changed [4] after the development of new technologies such as mobile and cloud computing, and it has become obvious that the design and implementation of software have a vital impact on power consumption; therefore, it is highly important that the people who work in this area be aware of this issue.

Mobile computing platforms have many limitations such as the memory size, the speed of CPU, the battery life, disk I/O, heat, and network bandwidth. These parts consume power in addition to the power consumed by software services, just for their availability [4]. Cloud computing data and processing centers used an estimated 1.5% of the total US electricity consumed in 2006, for a total cost of about $4.5 billion [38]. In addition to this, the PCs in the US consume 2% of all electricity [38]. Suggesting solution lies in resource management, which leads to more resource availability by increasing battery life and decreasing power bills.

Most of the power consumption researchers focus on CPU usage and overlook the power consumption caused by software evolution and change. In [4], the author introduced green mining, which is a research area that can be used to help the developers predict power consumption of software product based on the amount of software changes. Green mining has two paths. The first is mining software repositories in order to examine software changes. This path focuses on the analysis of objects found within software repositories. The second path combines repository analysis with dynamic analysis in order to measure and correlate power usage and code changes. This path is used to examine the factors and resource utilization of

changing software by looking at each change in a version control system and measuring its effect on power consumption [4].

"Green mining influences historical information extracted from the corpus of publicly available software to help provide software power consumption advice" [4]. Green mining demonstrates how the maintenance of a software affects a system's power usage. The main goal of green mining is to help software developers decrease software's power consumption by approximating the influence of software changes on power use.

2.4 Software Metrics

This section discusses the general idea of software metrics and introduce their importance, types, and ways to select a specific metric. In the following sub-sections, we introduce software measurement by using different metrics such as lines of code, function point, code churn metrics, and structural complexity metrics.

2.4.1 Importance of Software Measurement

One of the main objectives of software engineering is to control the software development process, which, in turn, leads to control over the expenses, schedules, and product quality. Software measurement is an important element of good software engineering. Software metrics can help achieve this objective, by evaluating (quantitatively) the quality of software development processes as well as the quality of software products [39].

There are two groups of software metrics: metrics that are related to the software product and metrics that are related to the software process. The former quantify software product—such as product's source code size—while the latter quantify the process of software development—such as the amount of human resources (measured in man-hours), spent in the design and coding phase [39].

2.4.2 Why Software Size Matters

There are two methods for measuring source code size: physical lines of source code and logical source statements. The former measures the size by counting the number of new line delimiters in the source code [40]. The latter tries to identify the boundaries of software instructions (independent of the source code formatting) and count the number of these instructions [40].

Common abbreviations for physical lines of source code are as follows: LOC, KLOC, SLOC, and KSLOC, where LOC stands for lines of code, K (kilo) specifies that the scale is in thousands, and S stands for source [40].

2.4.3 How to Measure Software Size

Software size can be measured in LOC, since it is a cost driver and a normalization feature. This metric can be used to standardize other metrics and compare similar projects. The only important associated issue is how we use it.

LOC is used to measure the physical length of the software. However, with nonprocedural and visual languages (for example, Visual Basic) LOC cannot be used as a reliable measure of software size, because code that is automatically generated by a GUI (graphical user interface) IDE (integrated development environment) is not considered when using LOC methods of measurement. Moreover, LOC fails to measure the amount of functionality (implemented in a given system), as well as the complexity and the technology involved. In addition, LOC cannot be used to compare two different projects, especially those written in different languages. Therefore, we may need to consider other size measurements in addition to LOC, as discussed in Section 2.4.5 [41].

There are various types of software metrics, and each one has a special convention and purpose. For example:

- Function points can be used to calculate lines of code (LOC) (depending on language).
- Lines of code can be used to calculate effort, where the associated mathematical model is

$$\text{Effort} = A * LOC^{B} + C \quad \textbf{(1)}$$

Procedures exist for finding A, B and C [41].

It is important to be aware of how to measure the size of the software in order to know how this size can affect the software's performance and consequently the energy consumption by the software. The traditional way to measure software size is through the number of lines of code (LOC). This is considered a simple measurement method, but, on the other hand, the fact that it is very easy can be considered undesirable when used improperly. For example, if productivity is measured in LOC per day, the more lines are written, whether they are useful or not, the more productive a programmer will appear.

2.4.4 Measuring Lines of Code

Counting source lines of code is an established process. There are many tools that can be used to count lines of code. For example, in our work we used an open source tool called "CLOC" [6]. The main concern with counting code is defining which rules to use so that assessments are valid—for example, whether to include/exclude empty lines or comments.

The Software Engineering Institute (SEI) of Carnegie-Mellon University has distributed a framework to count the source code [40], which includes a code counting checklist, allowing the user to determine how to count the code, leading to standardized benchmarking of code counting.

2.4.5 Function Point

As mentioned in Section 2.4.2, there is one important issue with using LOC, or any other physical size measurement for that matter. This issue relates to efficiency measurement; measuring the efficiency has little to do with the problem being solved and more to do with the expansion of software technology itself. Clients usually look for solutions, and they do not care about the language or technology used to create them [41].

Considering which strategy might be used to size a system, based on what it needs to do—that is, its functionality—rather than how it does it internally, is difficult. The number of features is independent of the language or coding style used by a given project [41].

One of the ways to measure functionality is to count the inputs, outputs, interfaces and databases in a system. This approach is called the function point (FP) method. Function Point Analysis (FPA) was designed as a way to measure the amount of business functionality of a system. Although many other measures have developed from function points, it is possibly still the most commonly used Functional Size Measurement (FSM) [41]. However, it is still hard to compute FP since it is not dependent on the physical line of code, but rather on project functionality.

2.4.6 LOCC: Code Churn Metrics

Code churn software metrics are used to measure the modifications made to a software product in a given time period, quantifying the amount of these modifications. It is defined as a weighted sum of lines added, modified or deleted to/from a file from one version of the file to another [42].

It can be calculated using data extracted from the source code management systems that track the changes automatically. Then a diff tool can be used to compute number of lines added, deleted and changed by a programmer in a given code commit [42]. The churn measure is used in our work to compute the overall change in terms of the lines of code (added, deleted and modified) and is defined as

Sum of (2 * lines_modified_code + lines_added_code + lines_removed_code). **(2)**

In Eq. 2, we give twice the weight to the modified lines, in comparison with added or removed lines. We do it because line modification can be represented as two operations: the removal of an old version of the line and the addition of a new version of the line.

Additional software evolution measures include the following [43]:

- Churn Files: The number of files modified within a given commit.
- Churn count: The number of modifications made to the set of files of interest.

2.4.7 Structural Complexity

The structural complexity metrics focus on the design and structure of the software. Design simplicity will lead to the idea of modularity and weak coupling. Structure simplicity will lead to the idea of simple control flows. The structural complexity metrics quantify these ideas [41].

Various structural complexity metrics are available that have been defined, used and developed. In our work, we focus on cyclomatic complexity (CC), which measures the control stream within a software item. It is one of the oldest and most popular complexity metrics (proposed by McCabe in 1976) [44]. CC is used to measure the number of control flows (paths) within a software product. The greater the number of paths – the higher the complexity. McCabe's metric was created to measure the ability to test and understand a software product. Essentially, the metric shows the smallest number of test cases needed to "cover" every executable statement. There are two ways to compute the cyclomatic complexity (leading to the same answer) [41]:

$$V(g) = e - n + 2, \quad \textbf{(3)}$$

where *g* represents the control graph of the module, *e* represents the edges number and *n* the nodes number; or

$$V(g) = bd + 1, \quad \textbf{(4)}$$

where *bd* is the count of binary decisions in the control graph [41].

The higher the CC of a given module, the more difficult it is to test and maintain (as a result of the higher complexity). Conversely, it is easier to test and maintain a module with low CC. Therefore, CC can be used to do the following:

- Recognize complex parts of code that need full design assessments.
- Detect simple parts of code that do not need an assessment.
- Estimate testing effort and maintenance effort (proportional to CC).

Chapter 3

3 Methodology and Implementation of Experiments

This chapter describes the research methodology and the implementation of the experiments, which aim to show the relation between the different MySQL database versions and their performance (and greenness) by executing the same standard workload (TPC-H) and using different setups such as varying database engines type (MyISAM and InnoDB), different amount of raw data loaded into the database (1GB or 3 GB) and the amount of memory allocated to the engine's buffer (256MB or 1024MB). In the following sections, we describe all related elements that are required for experiment implementation, such as experiment setup and experiment classification, and demonstrate the experiment framework implementation and the gathering of the required software metrics.

3.1 Experimental Setup

This section aims to provide information about all the experiments' setup requirements in addition to giving details about the devices and software used.

3.1.1 Hardware

The machine used in our experiments has the following characteristics:

- CPU model name: Intel(R), Pentium(R) 4 CPU 3.00GHz
- CPU Cache size: 2048 KB
- RAM: 3 GB

3.1.2 Operating System

The computer uses the Ubuntu S v.14.04 operating system with Linux kernel v.3.13.0-32-generic x86_64. We chose the Linux platform because it is a de facto standard server platform, and it is better designed for capturing computer-related statistics.

3.1.3 The Meter (Watts up? PRO)

The device used to measure the energy consumption, called "Watts up? PRO," has the following characteristics: 120 v, 60 Hz and 15 amps, and its memory capacity is related to the number of parameters that is stored and the memory mode. The device can store 32,000 records in stop/overwrite mode and only when logging watts; and approximately 1000 records [45] when the device is in automatic mode with all parameters recorded.

This model has an accuracy of ±1.5% and a resolution of 0.1 Wh [45]. All these factors led to the choice of this device among all the other available devices. The device allows the direct reading of the measurements to a computer via USB (Universal serial Bus) port. The device's high accuracy and rich API (application programming interface) make it a good choice to address our needs (to build an automatic framework for measuring energy consumption for prolonged periods of time).

3.1.4 Software under Study: MySQL Database

3.1.4.1 MySQL Versions and the Reason for Choosing Them

We chose the MySQL database for use in our experiments because it is the most widely used open-source RDBMS [5] and "it is the world's second most widely used Relational Database Management System (RDBMS)" [46] after Oracle. Since we used Linux as our operating system, we selected precompiled Linux binaries (for x86 64-bit platform) of MySQL, taken from the manufacturer's website [47]. There are 135 versions of Linux binaries versions available on the website [47]. These versions are available to download and install.

3.1.4.2 Major and Minor Database Releases

In our final experiments, we focused on four major releases with three minor releases for each major one. The three minor releases per major release are selected as follows: an early release (when the code is still somewhat unstable), an intermediate release (when the code is beginning to stabilize) and the latest release (when the code is stable). Such selection of releases allows us to obtain a representative subset of the release and track the development of the release as it matures. In total, we have 12 MySQL versions under study. A list of these releases is shown in Appendix A.1.

3.1.4.3 Different MySQL Engines Storage Types

We used two main types of MySQL storage engines in our experiments—MyISAM and InnoDB. MySQL v.5.5 and later versions use InnoDB by default. InnoDB offers ACID-compliant transaction features (atomicity, consistency, isolation and durability), which is important to confirm data consistency and integrity in the database, in addition to the foreign key support [47]. On the other hand, prior to v.5.5 MyISAM was the default storage engine for the MySQL [48]; it is better suited for analytic workloads and has many useful extensions (see Appendix A.5 for comparison of MyISAM and InnoDB).

3.1.4.4 MySQL Configuration File

Since we used two different types of MySQL engines, MyISAM and InnoDB, we had to use different commands in the MySQL configuration file that match the specific engine requirements, such as increasing the size of the memory buffer (using innodb_buffer_pool_size variable) in InnoDB, in order to be able to execute queries that deal with a data size bigger than the available RAM (in our case of 3 GB); otherwise, the execution would be very slow, which we noticed in our preliminary experiments.

When we ran the experiments for the MyISAM engine, we had to disable InnoDB's functionality using the configuration file to be sure that MyISAM would be the active engine (since MyISAM and InnoDB are bundled in the same binaries). A copy of the MySQL configuration file can be found in Appendix A.2.

3.1.4.5 Memory Setup

Tuning the MySQL default installation is very important to improve its performance, and the key buffer cache is an essential element to be changed in this tuning process; this subject is illustrated in Appendix A.4. A key buffer is used to cache the data from the hard drive into the memory. Theoretically, the more memory that is allocated to the cache, the faster the data processing. We set the buffer cache value to either 256 MB or 1024 MB.

Consequently, each database version is installed in the two memory configurations as result of the fact that each experiment is run two times (using 256MB and 1024MB); the reason for this is to get a clear idea of how the experiments' results change according to the changes in memory configuration.

3.1.5 Workload / TPC-H

"The TPC Benchmark™ H (TPC-H) is a decision support benchmark"; it is considered the standard benchmark in the database community [49]. This benchmark has a set of business-oriented ad-hoc queries. The data and queries simulate business practices and requirements. This benchmark mimics the systems for decision support that use large volume of data, execute very complicated queries and answer many industry-related questions. We used TPC-H version 2.17.0.

The TPC-H package has two utilities, namely, DBGEN and QGEN, which generate the test data and queries of the TPC-H workload, respectively. We used DBGEN to generate data of various sizes to be loaded into the database, and we used QGEN to generate the 22 queries to manipulate the data. In our experiments, we used all available TPC-H queries (22 queries); these queries are compatible with the MySQL database engine. A listing of these queries is available at the TPC-H website [49]. For populating the database, we used 1 GB from the available TPC-H dataset in some of our experiments and 3 GB in the rest of the experiments. We used the term "workload" to refer to the set of all the queries used and the term "statement" to refer to a specific query.

The TPC-H is an OLAP (on-line analytical processing) workload, typically having a relatively low volume of transactions. The workload queries are usually very complex and include aggregations. OLAP workloads are often used for data mining [50].The other type of workload is the OLTP (on-line transaction processing) workload, typically having a large number of short transactions (INSERT, UPDATE and DELETE) that have to be completed in near real time. For example, workload related to banking transactions would fit into OLTP category [50].

3.1.6 Different Sizes of Raw Data

We created raw datasets of reference data using the DBGEN tool (provided in the TPC-H package), which is described in Section 3.1.5 in this dissertation. The dataset sizes are 1, 2, 3 and 4 GB (DBGEN generated workloads in 1GB increments). Our goal was to identify the amount of data that would 1) fit into the memory and 2) force reads from the hard drive for each query. The

former would emulate an in-memory database (tailored for analytic workload), while the latter a large database not fitting into the memory. Empirically,[8] we found that a 1GB dataset is small enough to fit into the memory, while a 3GB dataset is large enough to trigger reads from the hard drive for each query.

3.2 Classification of Experiments

We implemented two experiments types: introductory and final experiments. Since we are using different MySQL engines and did not have a clear idea at the beginning of our work regarding the execution time or the energy required to execute any single query, we implemented many introductory experiments that helped us to make the decision about the final experiments that we used in our final analysis.

3.3 Framework of Experiments

At the beginning, we downloaded all the required compressed TAR archive files of the binary MySQL versions under study [47]. We used Python programming language to design a framework that automates the processes required to implement the experiments. The reason for this automation is to reduce the threat to validity related to human errors.

The framework automates the installation and measurement process: it decompresses the downloaded TAR files, installs the specific MySQL version, upgrades the database, executes the 22 TPC-H queries one by one and, finally, measures the execution time, system state statistics (such as CPU and IO load) and energy consumption per each statement.

3.3.1 Data Loading

In our eight experiments, we loaded 1 GB and 3 GB of raw data into the database. The DBGEN tool generates eight files in TBL format (plain text CSV-like format with | character as a field separator), one file per database table [49]. We then load the data from the TBL files into the corresponding tables using MySQL's LOAD utility. The relationships between these tables are illustrated in Figure A.1 in Appendix A.3.

[8] We used the SYSTAT tool [51] to capture IO statistics.

3.3.2 Limitations of the Measurement Process

We gather the data per statement. However, due to the limitations of the SYSTAT tool, we cannot capture computer state statistics if a query is executed in less than one second.

We also cannot measure the energy consumption of a query that is executed in less than 10 seconds. The resolution of our energy meter is 0.1Wh; the approximate amount of time in which our computer consumes 0.1Wh of energy is 10 seconds.

3.3.3 Results Files

In our results files, we collected and saved the following data: DB engine, Version, Key buffer size, Workload, Run, Energy consumption, Time spent, and Hardware-related statistics (such as load on CPU and IO systems).

3.4 Gathering Software Metrics

We gathered the source code for each of the MySQL versions under study by downloading them from the original MySQL website [47]. Then we created scripts to calculate the software code metrics, namely size, churn and complexity metrics, for each MySQL version. We called CLOC tool [6] from our scripts to extract size metrics, lines of code (LOC), churn metric and lines of code changed (LOCC) and the PMCCABE tool [52] to compute modified (MCC) and traditional cyclomatic complexity (TCC). The details of the metrics are given in Chapter 3 in this dissertation. Before computing the source code metrics, we eliminated a number of source code files. First, we eliminated source code files of test cases, since the code from them is not included in the production binaries of the database engine. Second, we removed source code not written in C and C++, as they are excluded from the production binaries as well.

Chapter 4

4 Results of Experiments

This chapter introduces and describes the experiments' results along with their data analysis, aiming to answer the underlying research questions. In Section 4.1, we provide a bird's-eye overview of the experiments. We then explore the relation between the energy and time spent during workload execution for all experiments in Section 4.2. We discuss the results of the experiments, in relation to RQ1 and RQ2, in Sections 4.3 and 4.4. Finally, in Section 4.5 we provide answers to the research questions.

4.1 Overview of Experiments

We grouped the resulting data using two approaches, as follows. First, we analyzed the results for the conglomerate of all 22 SQL statements (queries) as one unit; we denoted this approach as "Per-run analysis." Second, we examined the data from each of the 22 SQL statements (queries) independently, denoting this approach as "Per-statement analysis"; in the following sections, we explain these two approaches. Table 4.1 shows the eight different experimental setups with the corresponding memory buffer size; numbers in brackets represent the raw data size used in each experiment. Table 4.2 shows the list of the MySQL major releases used in our experiments along with a list of the minor MySQL releases under study.

Let us examine the results of each experiment in detail.

Table 4.1: A list of the experiments with the corresponding memory buffer size; numbers in brackets represent the amount of raw data used in each experiment.

Memory buffer size	MyISAM	InnoDB
256MB	Experiment 1 (1GB)	Experiment 5 (1GB)
	Experiment 2 (3GB)	Experiment 6 (3GB)
1024MB (1 GB)	Experiment 3 (1GB)	Experiment 7 (1GB)
	Experiment 4 (3GB)	Experiment 8 (3GB)

Table 4.2: A list of the four major MySQL releases under study with their corresponding three minor versions used in the experiments.

Major release	Minor release		Major release	Minor release
v.5.0	v.5.0.15		v.5.5	v.5.5.10
	v.5.0.67			v.5.5.20
	v.5.0.96			v.5.5.39
v.5.1	v.5.1.30		v.5.6	v.5.6.10
	v.5.1.50			v.5.6.15
	v.5.1.72			v.5.6.21

We calculated the Pearson correlation coefficient[9] between all the variables used in our experiments (such as energy consumed and time spent, energy consumed and lines of code changed, and time spent and modified code complexity). The Pearson correlation coefficient is computed as:

$$\rho = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \quad \textbf{(5)}$$

where $\{x_1,...,x_n\}$ is a dataset containing n values and $\{y_1,...,y_n\}$ another dataset containing n values. We used the correlation data to answer the research questions, as explained in subsequent sections. The mapping between a value of correlation coefficient and the strength of correlation is given in Table 4.3.

[9] The Pearson correlation coefficient is a method to measure the linear relation (dependence) between any two variables *X* and *Y*. We chose linear regression to analyze our data; therefore, we chose the Pearson correlation over the Spearman correlation, because Pearson is designed to match the sign and magnitude of a linear regression slope [53].

Table 4.3: Correlation coefficients and their strength as per [53].

Absolute Value of Coefficient of Correlation	Strength of Correlation
1	Perfect
0.7 - 0.9	Strong
0.4 - 0.6	Moderate
0.1 – 0.3	Weak
0	Zero

4.2 Execution Time vs. Energy Consumption

Figure 4.1 shows the relation between energy consumption and execution time for all eight experiments. We can see that the strong positive linear relation between these two variables: coefficient of determination, R2 = 0.9992 and the slope value, represented by *b* in Figure 4.1, is positive. The Pearson correlation coefficient between these two variables is equal to 0.9996, which is an almost perfect correlation (see Table 4.3). We need to keep this fact in mind, because in the next sections we will discuss the influence of various factors on energy consumed and time spent. Not surprisingly, the effect will be nearly identical.

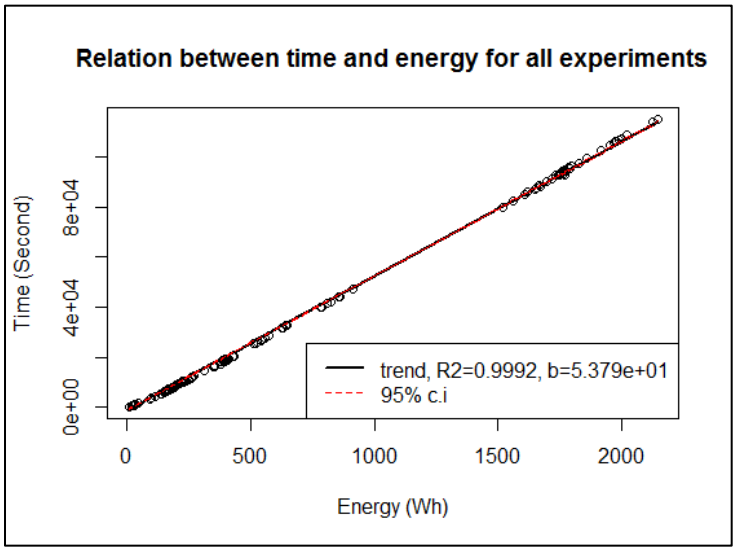


Figure 4.1: Relation between time spent and energy consumed for all experiments. A data point represents the time and energy data gathered for a given run of an experiment. The black line depicts the trend line obtained using linear regression, and the dotted red lines show the 95% confidence interval of the trend line.

4.3 RQ1: Experiments

In this section, we will analyze the results of our experiments to answer RQ1:

> How does the energy consumption and execution time of a database engine change as the product matures (from release to release)?

For the sake of brevity, we provide examples of detailed analysis of the data from Experiment 1 (for the MyISAM engine) and Experiment 5 (for the InnoDB engine) in Sections 4.3.1 and 4.3.2, respectively. The details of the remaining six experiments are given in Appendix B.1 and Appendix B.2. We then analyze all the experiments in Section 4.3.3.

4.3.1 Experiment 1: Engine=MyISAM, Key Buffer Size=256M, Database Size=1GB

In this experiment, we loaded 1 GB of raw data into the database and allowed the MyISAM database engine to use 256 MB of memory for its buffer.

Figure 4.2 shows that, overall, the energy consumption and execution time increased as MySQL matured. By eyeballing the plots, we can see that this trend is clearly pronounced as we move from one previous major release to the next one. To formally confirm this observation, we plot a linear trend line (computed using linear regression). We map release names to real numbers in order to compute linear regression. The first minor release v.5.0.15 is mapped to 1, the second minor release v.5.0.67 is mapped to 2, etc. The linear model explains most of the data variability: R2 for both energy and execution time are approximately equal to 0.83. The slope of the linear function (denoted by *b* in Figure 4.2) is positive in both cases.

However, when we examine the data per each single release in Figure 4.2, the results are not following one stable trend. In the case of major release v.5.0, energy consumption and execution time increase with MySQL development. The slope of linear functions, fitting both datasets, is positive; the function explains most of the variability: R2 equals 0.74 and 0.72, respectively, which reflects a strong trend.

In the next two major releases, v.5.1 and v.5.5, the energy consumption and execution time follow a Λ pattern: for instance, in v.5.1, the energy consumption and execution time of an intermediate release (v.5.1.50) is higher than that of an earlier release (v.5.1.30); energy

consumption and execution time of the mature release, v.5.1.72, is lower than that for the intermediate release v.5.1.50. However, the pattern is not very strong if we consider the variability of the measurements of each run. This is further confirmed by the high spread of the confidence interval lines (for this release) in Figure 4.2. The same explanation applied to release v.5.5, as shown in Figure 4.2.

In release v.5.6, energy consumption and execution time decreased with MySQL development. The slope of the linear functions fitting both datasets are negative R2=0.68 and 0.61 in the two datasets, respectively, which illustrates that strong trend line and the slope of linear functions, fitting both datasets, is negative, which proves the negative relation.

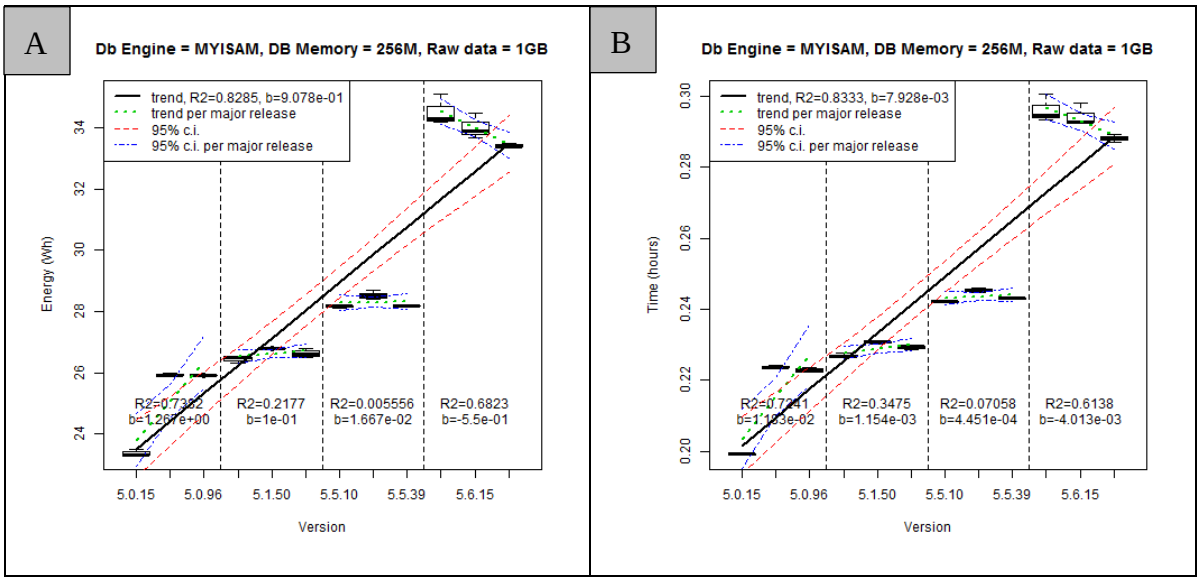


Figure 4.2: Experiment 1: subplot A is a box plot of the energy consumed by MySQL's versions; subplot B is a box plot of the execution time in MySQL's versions. In both subplots, the black line depicts a trend line obtained using linear regression; the dotted red lines show the 95% confidence interval of the trend line; the dotted green line represents the trend per major release and the dotted blue line shows the 95% confidence interval of the trend line per major release.

4.3.2 Experiment 5: Engine=InnoDB, Key Buffer Size=256M, Database Size=1GB

In this experiment, we loaded 1 GB of raw data into the database and allowed the InnoDB database engine to use 256 MB of memory for its buffer. Figure 4.3 shows that, overall, the energy consumption and execution time do not follow one fixed pattern as MySQL matures. Fitting the data with a linear function (using linear regression) does not explain the data

variability: R2 for energy and execution time are equal to 0.14 in both datasets, accordingly. Figure 4.3-A suggests that the v.5.5 minor releases are the greenest and the v.5.6 minor releases are the least green. The data for release v.5.0 are volatile (the R2 values for energy consumption and time are 0.32 and 0.26, respectively): v.5.0.15 shows results on par with the v.5.5 releases, but the remaining two minor releases, v.5.0.67 and v.5.0.96, are not as energy efficient. All minor releases of v.5.1 yield consistent (the R2 values for energy consumption and time are 0.68 and 0.89, respectively).

However, when we examine the data per major release in Figure 4.3, the results follow different patterns. Energy consumption and execution time for major release v.5.0 follow a Λ pattern: the energy consumption and execution time of intermediate release v.5.0.67 is higher than that of the early release v.5.0.15; the energy consumption and execution time of the mature release, v.5.0.96, is lower than that for intermediate release v.5.0.67. However, the pattern is not very strong as the variability of the data is high (R2 is equal to 0.32 and 0.26 for the energy and time datasets, accordingly). The slopes of linear functions fitting both datasets are positive.

Figure 4.3 suggests that for major release v.5.1, energy consumption and execution time may follow a positive pattern, and the pattern is considered strong if we take into account the variability of the measurements for each run: R2 equals 0.68 and 0.89 in the two datasets, accordingly. The slopes of linear functions fitting both datasets are positive, which represents this positive trend.

Energy consumption and execution time for major release v.5.5 follow a V pattern: the energy consumption and execution time of intermediate release v.5.20 is lower than that of the earlier release v.5.10; the energy consumption and execution time of the mature release, v.5.5.39, is higher than that of intermediate release v.5.5.20. However, positive of major release v.5.5, the pattern is not very strong, since the variability of the measurements per each run is high; R2 equals 0.23 and 0.17 in both datasets, accordingly, as shown in Figure 4.3. On the other hand, the slopes of linear functions fitting both datasets are positive in both datasets, which reflects a positive trend.

In the case of major release v.5.6, there is a clear negative trend based on the slope value *b*, which is negative in both datasets. Moreover, the variability of the data is not fixed, as the value of R2 is equal to 0.24 and 0.73 in the two datasets, respectively.

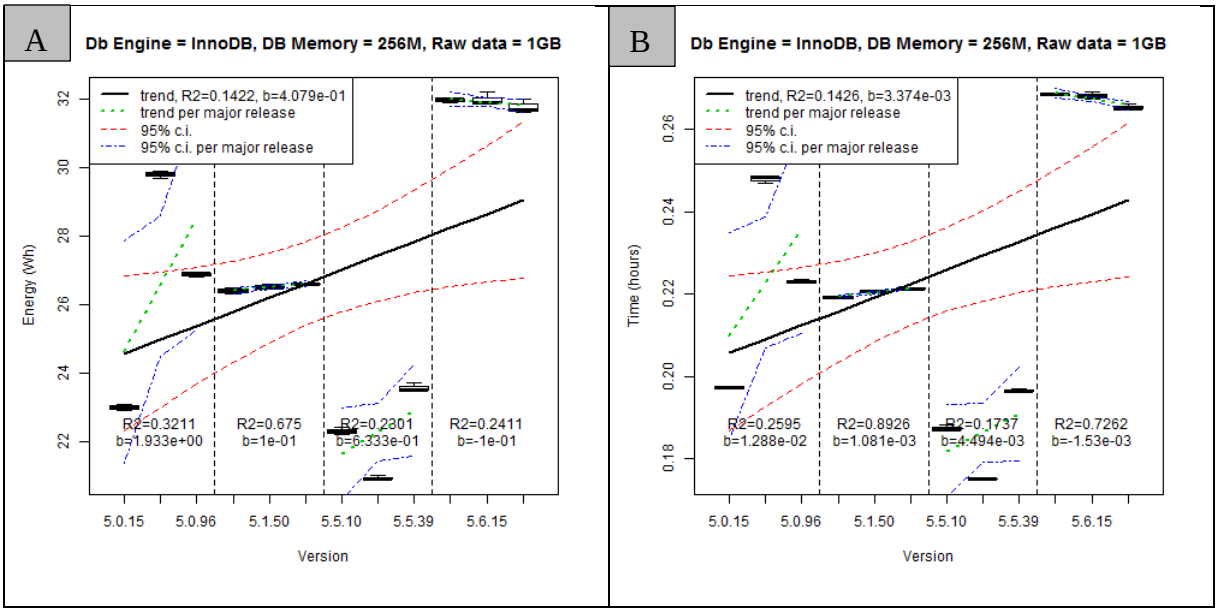


Figure 4.3: Experiment 5: subplot A is a box plot of the energy consumed in MySQL's versions; subplot B is a box plot of the execution time in MySQL's versions. Both subplots' black lines depicts a trend line obtained using linear regression; the dotted red lines show the 95% confidence interval of the trend line; the dotted green line represents the trend per major release and the dotted blue line shows the 95% confidence interval of the trend line per major release.

4.3.3 RQ1: Per-Run Analysis

4.3.3.1 MyISAM

As mentioned in Section 4.1, we performed four Experiments (#1–4) on the MyISAM engine. The details of the analysis for Experiment 1 are given in Section 4.3.1, and those for experiments 2–4 are provided in Appendix B.1. The analysis of the per-run data, collected during these four experiments, suggests that both the energy consumption and execution time of the MyISAM engine increased as the engine matured. In essence, on average, the newer major releases are slower and less green than the older ones.

Table 4.4: MyISAM minimum energy consumption data per major release with relative percentages of difference between each pair of adjacent releases. Green color represents minimum energy consumed by a specific release in each experiment; Red color represents the maximum energy consumed by a specific release in each experiment.

MyISAM							
	Minimum energy (Eq. 6) consumed by a minor release for a given major release (Wh)				Energy consumption: relative difference (Eq. 7) between previous and current release (%)		
Major release	**v.5.0**	**v.5.1**	**v.5.5**	**v.5.6**	**v.5.1 vs. v.5.0**	**v.5.5 vs. v.5.1**	**v.5.6 vs. v.5.5**
Experiment1	23.37	26.43	28.17	33.43	13.12	6.56	18.70
Experiment2	115.43	155.53	166.53	187.27	34.74	7.07	12.45
Experiment3	23.23	26.40	28.07	33.17	13.63	6.31	18.17
Experiment4	156.17	208.83	214.63	253.67	33.72	2.78	18.19

Table 4.4 summarizes the energy consumption for the MyISAM engine. In this table, we show the average energy consumption of a minor release (average of three runs) that consumes the least amount of energy within a set of minor releases belonging to a given major release. Formally, for a given major release, the energy reading displayed in the table is computed as follows:

$$min\left(\frac{r_{1,1}+r_{1,2}+r_{1,3}}{3}, \frac{r_{2,1}+r_{2,2}+r_{2,3}}{3}, \frac{r_{3,1}+r_{3,2}+r_{3,3}}{3}\right), \quad \textbf{(6)}$$

where $r_{i,j}$ represents energy consumption for the *j*-th run of the *i*-th minor release belonging to the major release of interest. The average of three runs per minor release in Eq. 6 is taken to minimize the measurement error; then the minimum amount of energy from the three average readings is chosen. Furthermore, the relative percentage of difference between each two adjacent releases is calculated using the following formula:

$$((\text{new\_release\_value} - \text{old\_release\_value}) / \text{old\_release\_value}) \times 100. \quad \textbf{(7)}$$

The oldest release, v.5.0, is the greenest (of all major releases), because it consumed the least amount of energy to execute the same reference workload. On the other hand, the newest release, 5.6, is the least green, because it consumed the largest amount of energy, as shown in Table 4.4.

All four experiments showed a consistent increase in energy consumption and time spent between any two major consecutive releases (all the difference values in Tables 4.4 and 4.5 are positive, representing this increase). For example, Experiment 1 shows that release v.5.1 is 13.12% less efficient than v.5.0, release v.5.5 is 6.56% less efficient than release v.5.1 and release v.5.6 is 18.70% less efficient than release v.5.5—see Table 4.4 for details. The timing results per major release are computed using a formula that is structurally similar to that of Eq. 6, as shown in Eq.8:

$$min\left(\frac{t_{1,1}+t_{1,2}+t_{1,3}}{3}, \frac{t_{2,1}+t_{2,2}+t_{2,3}}{3}, \frac{t_{3,1}+t_{3,2}+t_{3,3}}{3}\right), \quad \textbf{(8)}$$

where $t_{i,j}$ represents workload execution time for the j-th run of the i-th minor release belonging to the major release of interest.

As discussed in Section 4.2, energy consumption and execution time are correlated almost perfectly. Therefore, the timing results (summarized in Table 4.5) are very close to the energy consumption results. In other words, execution time increases as the engine matures, with the v.5.0 release being the fastest and v.5.6 being the slowest. We conjecture that this behavior of the MyISAM engine can be explained by the fact that the inclusion of additional functionality to the MyISAM engine in each subsequent release requires additional computational resources, leading to an increase of time and energy consumption.

Table 4.5: MyISAM minimum execution time data per major release with the relative percentage of difference between each pair of adjacent releases. Green color represents minimum time spent by a specific release in each experiment; Red color represents the maximum time spent by a specific release in each experiment.

MYISAM							
	Minimum execution time (Eq. 8) taken by a minor release for a given major release (Hr)				Execution time: relative difference (Eq. 7) between previous and current release (%)		
Major release	**v.5.0**	**v.5.1**	**v.5.5**	**v.5.6**	**v.5.1 vs. v.5.0**	**v.5.5 vs. v.5.1**	**v.5.6 vs. v.5.5**
Experiment1	0.20	0.23	0.24	0.29	13.92	6.74	18.97
Experiment2	1.22	1.75	1.87	2.16	43.57	6.97	15.33
Experiment3	0.20	0.23	0.24	0.29	11.13	6.87	18.14
Experiment4	1.83	2.56	2.61	3.17	39.53	1.98	21.26

4.3.3.2 InnoDB

We conducted four Experiments (#5–8) on the InnoDB engine. The analysis of Experiment 5 is given in Section 4.3.2; the analysis of the remaining three experiments, for the sake of brevity, is given in Appendix B.2.

As discussed in Appendix B.2.2, in the cases of Experiments 7 and 8, the values of energy consumption and time spent for v.5.0.15 are dramatically different from the values of the other two minor releases of v.5.0 (namely, v.5.0.67 and v.5.0.96). We conjecture that the functionality of the InnoDB engine in this release behaves abnormally, leading to anomalous data. In Tables 4.6 and 4.7, we provide the data with and without anomalous readings. However, in our analysis we chose to discard the anomalous observations.

The summary of the energy consumption (computed using Eq.6) for Experiments 5–8 is given in Table 4.6 along with the relative percentage of difference between each pair of adjacent releases, which is calculated using Eq. 7. We can see that the results for the InnoDB engine are less uniform (in comparison with the MyISAM engine). As in the case of MyISAM, the least green major release is the most recent one (namely, v.5.6), except in Experiment 8, where v.5.5 appear to be the least green.

By examining Tables 4.6 and 4.7, we can see some negative difference values, which represent a decrease in energy or time spent between any two adjacent releases. We obtained different results for the greenest engine: In the case of Experiments 6 and 7, the greenest version is the oldest one[10] (v.5.0), while, in the case of the remaining Experiments, 5 and 8, the greenest engine is not the oldest one; in fact, the greenest engines are v.5.5 and 5.1, respectively. Experiments 5 and 7 cache all the data in the memory (using 1 GB raw data size); Experiments 6 and 8 have to read the data from the hard drive (using 3 GB raw data size). This implies that the functionality of v.5.5 may be better suited for handling an IO-intensive workload.

We conjecture that this difference in behavior between the MyISAM and InnoDB engines can be partially explained by the fact that InnoDB was designed to handle OLTP workloads. Our workload, as discussed in Section 3.1.5, is the OLAP workload. Since the developers have not

[10] In Experiment 7, v.5.0 and v.5.5 yield identical results, but the timing results for v.5.0 are slightly better (as shown in Table 4.7).

focused on satisfying OLAP requirements, the performance and energy consumption results for the current OLAP workload are volatile. Peculiarly, before major release v.5.5, MYSAM was a default database engine; starting from v.5.5, InnoDB became the default one [48]. Database administrators interested in executing an OLAP workload should take into account this difference while setting up their databases.

Table 4.6: InnoDB minimum energy consumption data per major release with the relative percentage of difference between each pair of adjacent releases; numbers in brackets represent the values with outlier release v.5.0.15 included. Green color represents minimum energy consumed by a specific release in each experiment; Red color represents the maximum energy consumed by a specific release in each experiment.

InnoDB							
	Minimum energy (Eq. 6) consumed by a minor release for a given major release (Wh)				Energy consumption: relative difference (Eq. 7) between previous and current release (%)		
Major release	**v.5.0**	**v.5.1**	**v.5.5**	**v.5.6**	**v.5.1 vs. v.5.0**	**v.5.5 vs. v.5.1**	**v.5.6 vs. v.5.5**
Experiment5	23.00	26.40	20.93	31.77	14.78	-20.71	51.75
Experiment6	92.30	404.50	391.77	636.20	338.24	-3.15	62.39
Experiment7	8.73 (8.73)	9.03	8.73	11.30	3.44 (3.44)	-3.32	29.39
Experiment8	1731.13 (101.00)	1646.27	1783.73	1748.93	-4.90 (1529.97)	8.35	-1.95

Table 4.7 summarizes the execution time for Experiments 5–8. The timing data are computed using Eq.8. The relative percentage of difference between each two adjacent releases is calculated using Eq. 7. Due to correlation of time and energy consumption (discussed in Section 4.2), the timing data mirrors the energy data: v.5.6 is the slowest in three Experiments (5, 6 and 7), and v.5.5 is the slowest in Experiment 8. The fastest release in Experiments 6 and 7 is v.5.0, while v.5.5 is the fastest for Experiments 5, and v.5.1 is the fastest for Experiment 8.

Table 4.7: InnoDB minimum execution time data per major release with the relative percentage of difference between each pair of adjacent releases; numbers in brackets represent the values with the outlier release v.5.0.15 included. Green color represents minimum time spent by a specific release in each experiment; Red color represents the maximum time consumed by a specific release in each experiment.

InnoDB							
	Minimum execution time (Eq. 8) taken by a minor release for a given major release (Hr)				Execution time: relative difference (Eq. 7) between previous and current release (%)		
Major release	**v.5.0**	**v.5.1**	**v.5.5**	**v.5.6**	**v.5.1 vs. v.5.0**	**v.5.5 vs. v.5.1**	**v.5.6 vs. v.5.5**
Experiment5	0.20	0.22	0.18	0.27	11.12	-20.12	51.53
Experiment6	0.87	5.27	5.22	8.89	509.00	-1.01	70.21
Experiment7	0.075 (0.075)	0.079	0.076	0.097	4.8 (4.86)	-4.14	28.71
Experiment8	25.45 (1.00)	24.31	26.64	26.02	-4.48 (2340.22)	9.57	-2.33

4.3.3.3 Examining the effects of the different experiments setups on the final outcomes:

By comparing Tables 4.4 and 4.6 (energy consumption) and Tables 4.5 and 4.7 (execution time) for MyISAM's and InnoDB's outcomes respectively, we found that MyISAM's results are less volatile than InnoDB ones (if we compare release to release or experiment to experiment within the same engine). This can be explained by the fact that, as discussed in Section 4.3.3.2 MyISAM was designed to handle OLAP workloads, while InnoDB was designed for OLTP ones.

Effect of Raw Data Size: From the Tables 4.4-4.7, we can see that in Experiments 1 vs. 5 and 3 vs. 7, dealing with 1GB of raw data that can fit into computer's memory, InnoDB engine is greener and faster than MyISAM engine. In the case of Experiments 2 vs. 6 and 4 vs. 8, dealing with 3GB of raw data that cannot fit into memory, MyISAM engine is greener and faster than InnoDB engine with one exception: InnoDB's early release v.5.0 (in Experiment 6) is greener and faster than MyISAM's one (in Experiment 2).

This suggests that InnoDB may be better suited for in-memory processing than MyISAM. However, for larger datasets that cannot fit into memory (which is more common for analytic workloads), MyISAM may be a better choice

Effect of Memory Buffer: Tables 4.4-4.7 show that in the case of in-memory processing (for Experiments 1 vs. 3 and 5 vs. 7, dealing with 1GB raw data) increase of the memory buffer from

256MB to 1024MB leads to marginal (less than 1%) improvement in greenness and performance for MyISAM engine. In the case of InnoDB the improvement is significant (approximately 60%). This can be explained by the fact that MyISAM has a rudimentary memory manager, relying mainly on OS mechanism for file caching, while InnoDB has a more sophisticated memory manager (see Section 3.1.4.3 and Appendix A.4 for details).

In the case of processing data that cannot fit into memory (experiments 2 vs. 4 and 6 vs. 8, dealing with 3GB of raw data) increase of the memory buffer leads to significant (> 35%) degradation of greenness and performance for both MyISAM and InnoDB engines. This can be explained by the fact that OS mechanism for file caching outperforms database engines memory manager.

4.4 RQ2: Experiments

In this section, we will focus on answering RQ2:

Which software metrics affect energy consumption and execution time?

In order to answer RQ2, we analyzed the relations between energy consumption and software metrics—namely, LOC, LOCC, TCC and MCC in all eight experiments. Details of the metrics are given in Chapter 3.

First, we examined the correlation among the software metrics (see Table 4.8). We found that LOC is strongly correlated with MCC and TCC. This behavior has been observed in other software products in the past [4] [54]. MCC and TCC are perfectly correlated, by construction, since their formulas are similar (see Section 2.4.7 for details). LOCC is weakly correlated with the other variables.

Table 4.8: Pearson correlation coefficient values between the software metrics used by using a per-run-analysis.

MyISAM and InnoDB				
	LOC	LOCC	MCC	TCC
LOC	**1.00**	-0.24	**0.76**	**0.75**
LOCC	-0.24	**1.00**	-0.23	-0.23
MCC	**0.76**	-0.23	**1.00**	**1.00**
TCC	**0.75**	-0.23	**1.00**	**1.00**

We investigated the relation between energy consumption (time spent) and each software metric graphically, using corresponding scatter plots, and numerically, by analyzing the trend lines obtained using linear regression and correlation between variables.

In this section, we introduce two additional response variables: change in energy consumption and change in time spent. They are defined, respectively, as:

(average_energy_conusmed_by_minor_release_N) – (average_energy_conusmed_by_minor_release_N-1) **(8)**

and

(average_time_spent_by_minor_release_N) – (average_time_spent _by_minor_release_N-1). **(9)**

The variables represent the difference between energy consumed (or time spent) by a given release and previous release.

Tables 4.9 and 4.11 show the Pearson correlation coefficient values between energy consumption (or time spent) and various software metrics in each experiment. By examining these tables, we see that the results are identical due to an almost perfect correlation between energy and time, as disused in Section 4.2.

Table 4.9: Pearson correlation coefficient values between energy consumption and the software metrics used using a per-run analysis. The values in brackets for Experiments 7 and 8 show correlation values with outlier data points (see Section 4.3.3.2 for a discussion on outliers).

	MyISAM				InnoDB			
Metric	Exper.1	Exper.2	Exper.3	Exper.4	Exper.5	Exper.6	Exper.7	Exper.8
LOC	**0.92**	**0.72**	**0.89**	0.66	0.65	**0.78**	**0.88** (0.02)	0.48 (0.34)
LOCC	-0.01	0.01	0.02	0.02	0.09	-0.21	0.01 (0.01)	0.16 (0.16)
TCC	0.53	0.19	0.46	0.28	**0.80**	0.50	**0.76** (0.26)	0.11 (-0.03)
MCC	0.55	0.20	0.48	0.30	**0.81**	0.52	**0.75** (0.249)	0.11 (-0.01)

Tables 4.10 and 4.12 show the Pearson correlation values between changes in energy consumption and changes in time spent, respectively, along with the LOCC metric. These two tables are identical (similar to Tables 4.9 and 4.11) as a result of the almost perfect correlation between energy consumed and time spent.

Table 4.10: Pearson correlation coefficient values between changes in energy consumption and LOCC. The values in brackets for Experiments 7 and 8 show correlation values with outlier data points (see Section 4.3.3.2 for a discussion on outliers).

	MyISAM				InnoDB			
Metric	Exper.1	Exper.2	Exper.3	Exper.4	Exper.5	Exper.6	Exper.7	Exper.8
LOCC	0.55	**0.77**	**0.73**	0.64	0.25	0.16	0.49 (0.11)	0.18 (0.04)

Table 4.11: Pearson correlation coefficient values between time spent and the software metrics used using a per-run analysis. The values in brackets for Experiments 7 and 8 show correlation values with outlier data points (see Section 4.3.3.2 for a discussion on outliers).

	MyISAM				InnoDB			
Metric	Exper.1	Exper.2	Exper.3	Exper.4	Exper.5	Exper.6	Exper.7	Exper.8
LOC	**0.92**	**0.70**	**0.89**	0.64	0.67	**0.80**	**0.86** (0.01)	0.49 (0.34)
LOCC	-0.03	0.03	0.02	0.03	0.09	-0.21	0.02 (0.02)	0.14 (0.14)
TCC	0.53	0.17	0.47	0.28	**0.82**	0.50	**0.75** (0.25)	0.09 (-0.03)
MCC	0.55	0.19	0.48	0.29	**0.83**	0.52	**0.75** (0.24)	0.09 (-0.01)

Table 4.12: Pearson correlation values between changes in time spent and LOCC. The values in brackets for Experiments 7 and 8 show correlation values with outlier data points (see Section 4.3.3.2 for a discussion on outliers).

	MyISAM				InnoDB			
Metric	Exper.1	Exper.2	Exper.3	Exper.4	Exper.5	Exper.6	Exper.7	Exper.8
LOCC	0.53	**0.82**	**0.75**	0.66	0.28	0.16	0.52 (0.11)	0.17 (0.04)

The rest of the section is structured as follows. We give examples of the detailed analysis of the data from Experiment 1 (for the MyISAM engine) and Experiment 5 (for the InnoDB engine) in Sections 4.4.1 and 4.4.2, respectively.[11] We then analyze the results of the experiments in Section 4.4.3.

[11] The details of the remaining six experiments are given in Appendix B.3.

4.4.1 Experiment 1: Engine=MyISAM, Key Buffer Size=256M, Database Size=1GB

The following scatter graphs (Figures 4.4 and 4.5) represent the relation between the energy consumption (time spent) and the different software metrics we used. By examining the relation between the size of code (represented by the LOC metric for all the versions under study) and the energy consumed, we found a strong positive linear relation between the variables: as can be seen from Figure 4.4-A, R2=0.85, and slope coefficient *b* is positive. This is further confirmed by a strong (ρ=0.92) correlation between the variables, as shown in Table 4.9.

As can be seen from Figures 4.4-C and 4.4-D and Table 4.9, there exists a moderate positive linear relation between TCC (MCC) and consumed energy: R2=0.28 (R2=0.30) with a positive slope, and the correlation coefficient is equal to ρ=0.53 (ρ=0.55). As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results. We found no correlation between LOCC and consumed energy. As can be seen from Figure 4.4-B and Table 4.9, R2=0.0002 and ρ= - 0.01.

Even though the relation between LOCC and consumed energy is low, there exists a moderate positive linear relation between LOCC and the change in energy consumption. As can be seen from Figure 4.4-E and Table 4.10, the variability of the data is high (R2=0.3), but the correlation is moderate (ρ=0.55).

Experiment 1's data, related to time spent (shown in Table 4.11), are identical to the energy consumption data in Table 4.9 as a result of the almost perfect correlation between energy and time (discussed in Section 4.2). Therefore, the LOC metric has a strong correlation with time spent, MCC and TCC have moderate correlation with time spent and no correlation is found between the LOCC metric and time spent. Table 4.12 shows a moderate correlation (ρ=0.53) between changes in time with the LOCC metric.

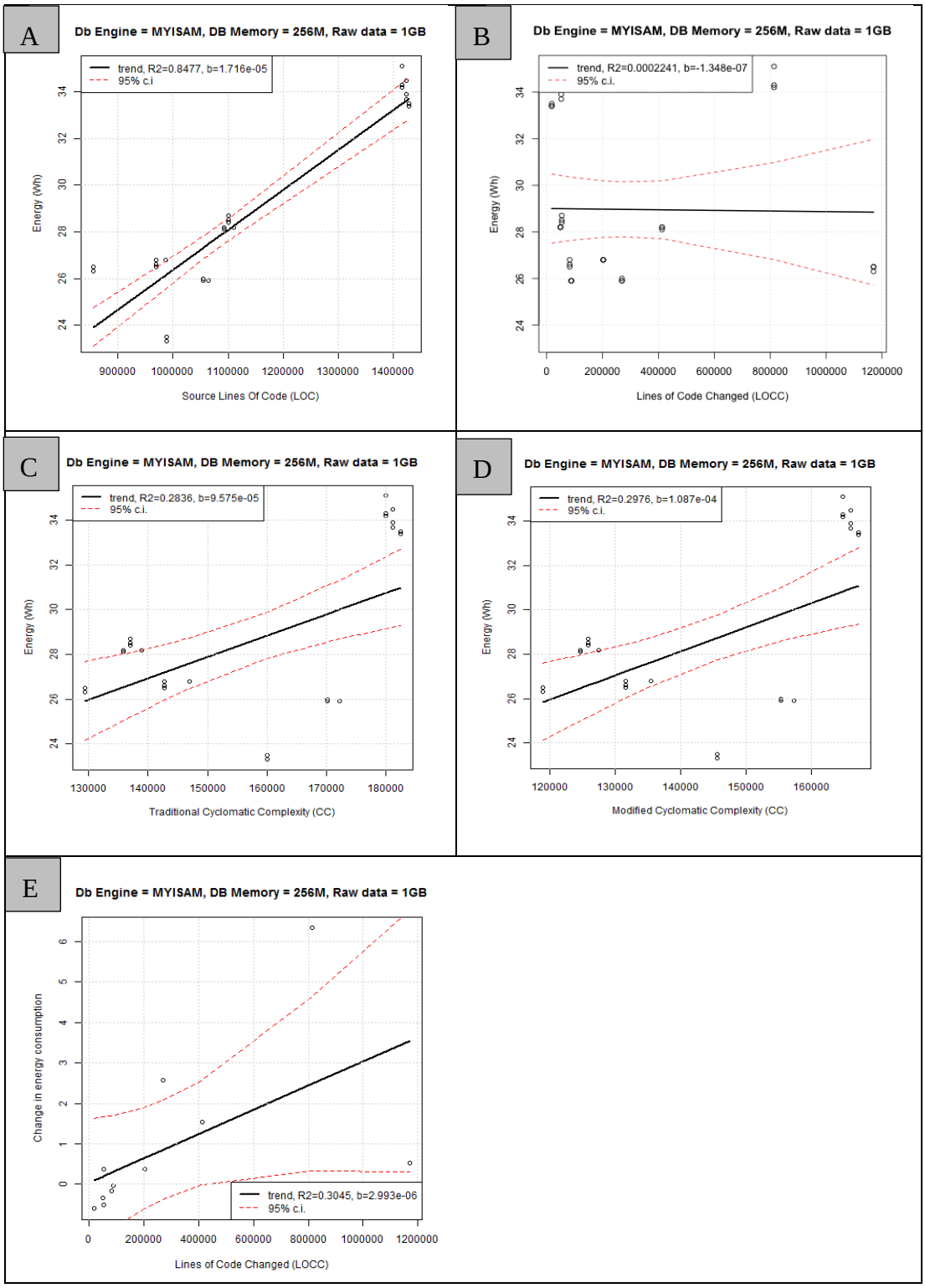


Figure 4.4: Relation between energy consumed and various software metrics for Experiment 1. Subplot A is a scatterplot of LOC against energy; subplot B is a scatterplot of LOCC against energy; subplot C is a scatterplot of TCC against energy; subplot D is a scatterplot of MCC against energy and subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots, the black line depicts a trend line obtained using linear regression, and the dotted red lines show the 95% confidence interval of the trend line.

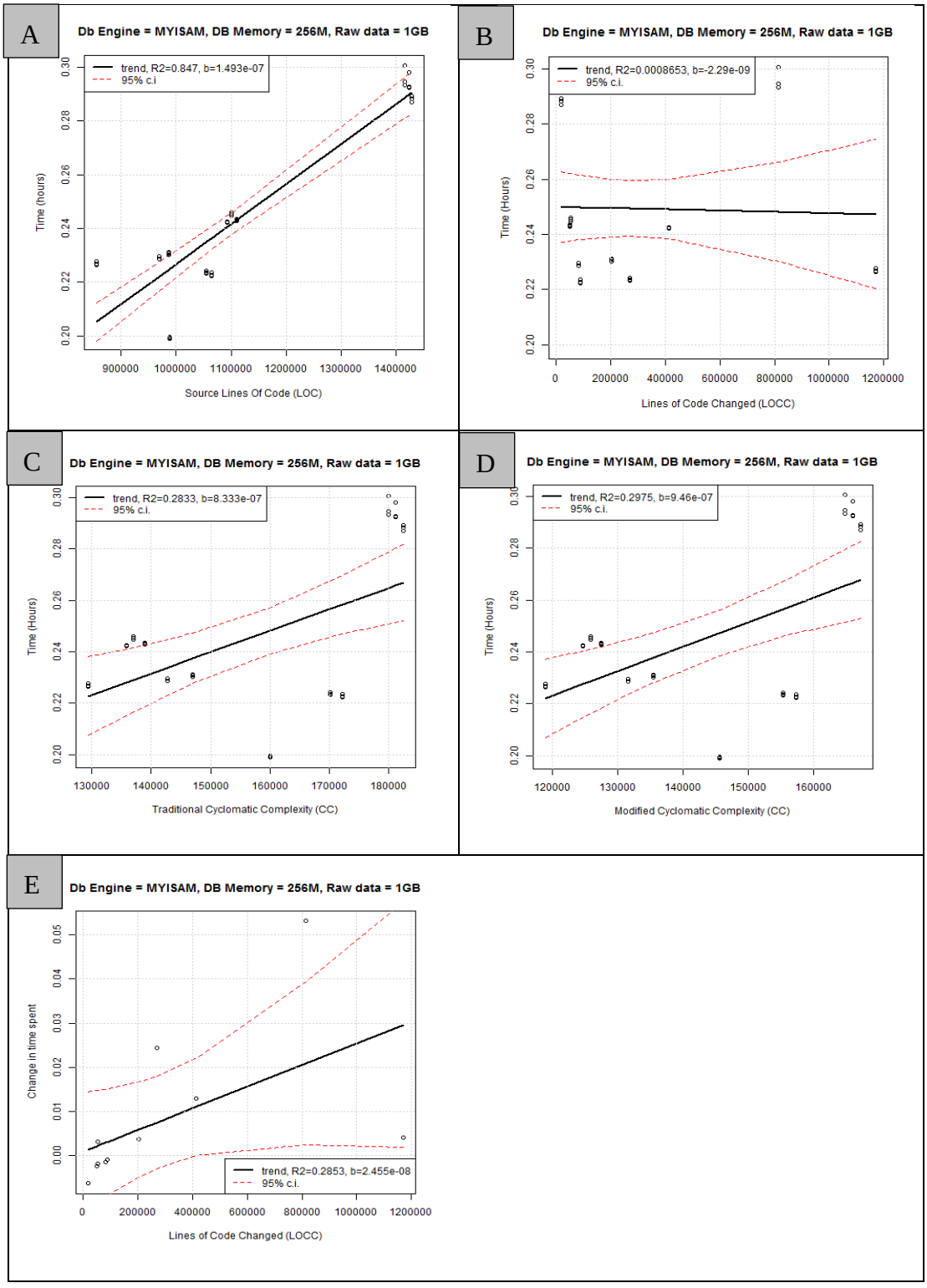


Figure 4.5: Relation between time spent and various software metrics for Experiment 1. Subplot A is a scatterplot of LOC against time; subplot B is a scatterplot of LOCC against time; subplot C is a scatterplot of TCC against time; subplot D is a scatterplot of MCC against time and subplot E is a scatterplot of LOCC against change in time spent. On all subplots, the black line depicts a trend line obtained using linear regression, and the dotted red lines show the 95% confidence interval of the trend line.

4.4.2 Experiment 5: Engine=InnoDB, Key Buffer Size=256M, Database Size=1GB

By investigating the scatter graphs in Figures 4.6 and 4.7 and examining the relation between the LOC metric and the energy consumption (time spent), we found a moderate linear relation between the variables: as can be seen from Figure 4.6-A, R2=0.42, and slope coefficient *b* is positive. This is further confirmed by the moderate (ρ=0.65) correlation between the variables, as shown in Table 4.9.

As shown in Figures 4.6-C and 4.6-D, and Table 4.9, there exists a moderate to strong positive linear relation between TCC (MCC) and consumed energy: R2= 0.64 (R2=0.66) with a positive slope, and the correlation coefficient is equal to 0.80 (ρ=0.81). As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results.

We found no correlation between LOCC and consumed energy. As can be seen from Figure 4.6-B and Table 4.9, R2=0.0087, and the correlation coefficient is equal to 0.09. The low relation between LOCC and consumed energy is identical to the relation between LOCC and the change in energy consumption. Figure 4.6-E and Table 4.10 show that the variability of the data is high (R2=0.06), and the correlation is weak (ρ=0.25).

The time-spent-related data of Experiment 5 (shown in Table 4.11) are identical to Table 4.9's energy consumption data due to the almost perfect correlation between energy and time (discussed in Section 4.2). Thus, the LOC metric has a moderate correlation with time spent (ρ=0.67), MCC and TCC have strong correlation with time spent (ρ > 0.82) and no correlation is found between the LOCC metric and time spent (ρ=0.09). Table 4.12 shows a weak correlation (ρ=0.28) between changes in time with the LOCC metric.

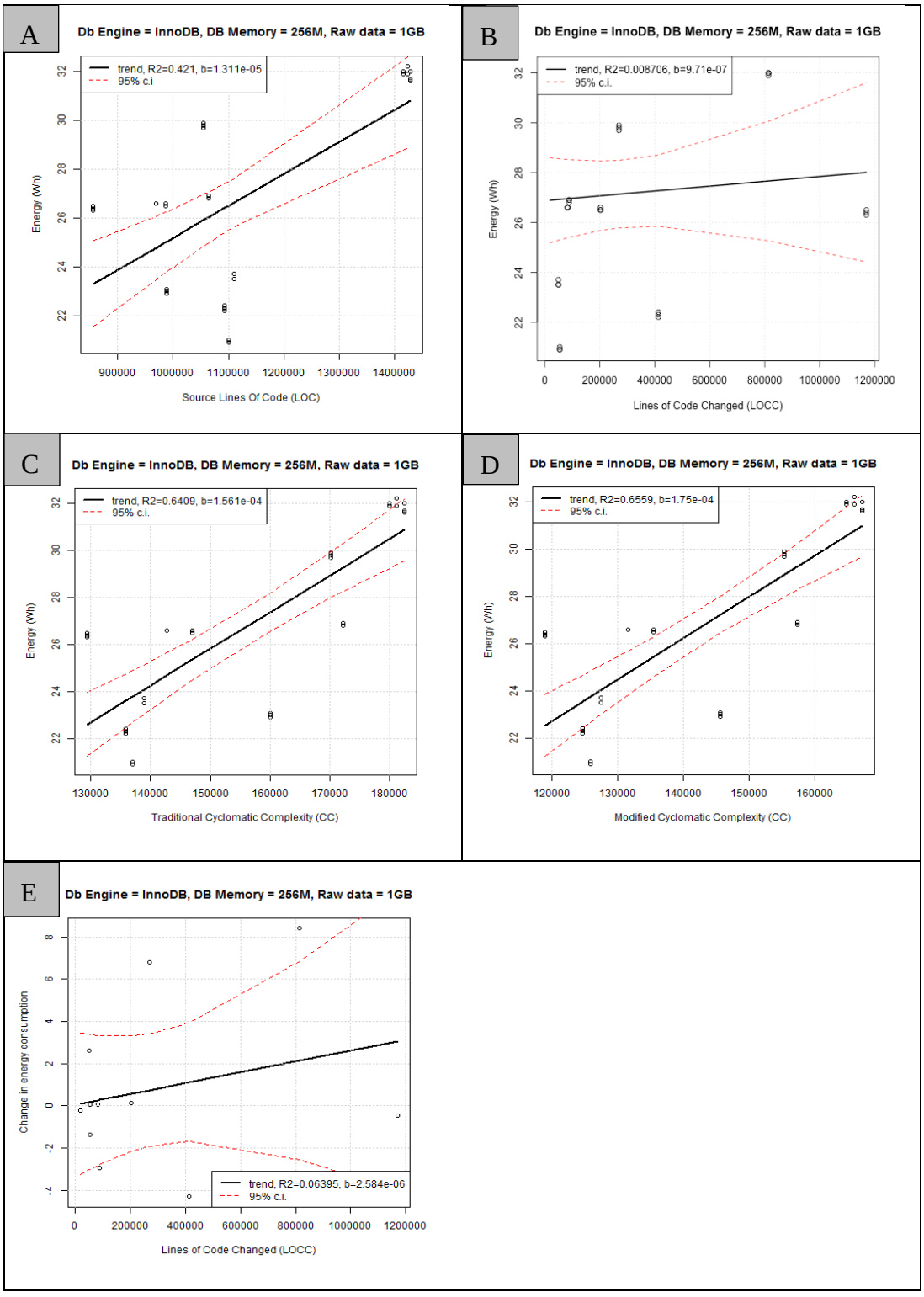


Figure 4.6: Relation between energy consumed and various software metrics for Experiment 5. Subplot A is a scatterplot of LOC against energy; subplot B is a scatterplot of LOCC against energy; subplot C is a scatterplot of TCC against energy; subplot D is a scatterplot of MCC against energy and subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots, the black line depicts a trend line obtained using linear regression, and the dotted red lines show the 95% confidence interval of the trend line.

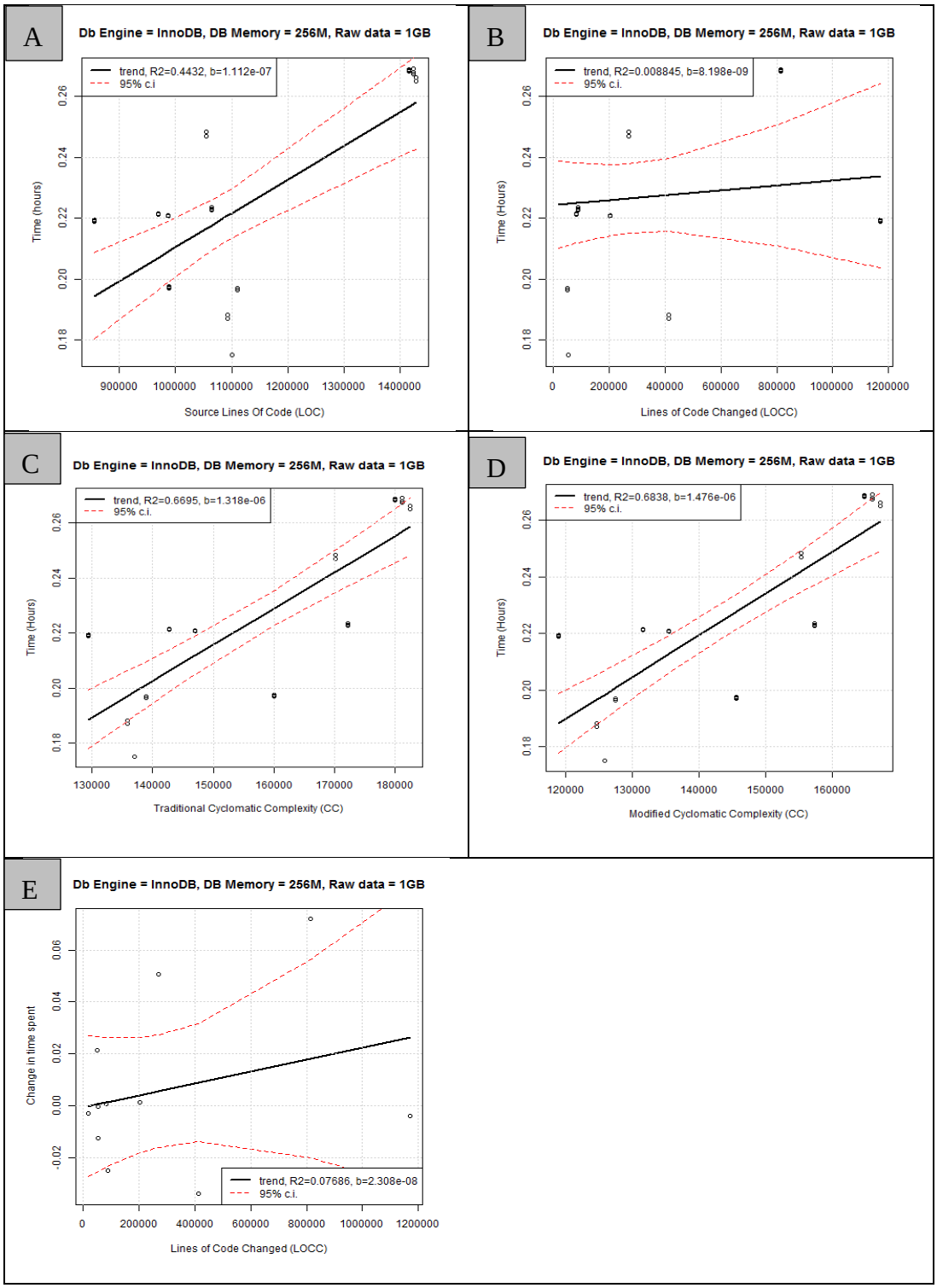


Figure 4.7: Relation between time spent and various software metrics for Experiment 5. Subplot A is a scatterplot of LOC against time; subplot B is a scatterplot of LOCC against time; subplot C is a scatterplot of TCC against time; subplot D is a scatterplot of MCC against time and subplot E is a scatterplot of LOCC against change in time spent. On all subplots, the black line depicts a trend line obtained using linear regression, and the dotted red lines show the 95% confidence interval of the trend line.

4.4.3 RQ2: Per-Run Analysis

As shown in Tables 4.9 and 4.11, there exists a moderate to strong positive correlation between consumed energy (or time spent) and codebase size, measured in LOC, for both database engines: MyISAM and InnoDB. In the case of MyISAM, there is no correlation between LOCC and consumed energy (or time spent); while in case of InnoDB there is none to weak correlation.

Tables 4.10 and 4.12 show that the change in energy consumption (or change in time spent) between releases is correlated with the code churn, measured by LOCC; the correlation for the MyISAM engine is moderate to strong, while the correlation for InnoDB engine is weak.

The correlation between energy consumed (or time spent) and code complexity metrics (MCC and TCC) is weak to strong, suggesting that it cannot be used as a consistent predictor of consumed energy. However, an examination of Table 4.8 suggests that the relation between LOC and complexity metrics (MCC and TCC) is strong, suggesting that complexity grows proportionally to codebase size (as has been shown in the past [54]).

4.5 Discussion

4.5.1 RQ1

The answer to RQ1: "How does the energy consumption and execution time of a database engine change as the product matures (from release to release)?" is as follows.

In the case of MyISAM, the overall energy consumption increases as the product matures, suggesting that the additional functionality added with every new release consumes additional resources.

In the case of InnoDB, the earliest major release is the greenest in 50% (2 out of 4) of the experiments, while the latest major release is the brownest in 75% (3 out of 4) of the experiments. In the case of the remaining experiments, the greenest and the brownest titles are claimed by intermediate releases.

As discussed in Section 4.3.2, we conjecture that the difference between the engines can be explained by the fact that MyISAM was designed to handle mainly OLAP workloads, while InnoDB is OLTP oriented, hence the variability.

The results of execution time (performance) are similar to energy consumption results (due to the almost perfect correlation between energy consumed and time spent, see Section 4.2):

in MyISAM case, the overall execution time increases as the product matures, i.e., newer releases are slower than the older ones. InnoDB execution time findings are identical to energy consumptions findings as well.

This is different from the results of the experiments on the Firefox web-browser [4], where energy consumption decreased as the product matured. This suggests that, depending on the product and its domain, the results may vary.

This difference can potentially be explained by the fact that the two products' (Firefox and MySQL) application domain, construction methods, and coding styles are different.

4.5.2 RQ2

The answer to RQ2: "Which software metrics affect energy consumption and execution time?" is as follows. Consumed energy is governed mainly by the size of the code base. The code size LOC metric serves as a strong predictor of energy consumption for both database engines. The code churn (LOCC) and complexity (MCC and TCC) metrics results are weaker.

The results for execution time (performance) are similar (due to the almost perfect correlation between energy consumed and time spent, see Section 4.2): The time is governed mainly by LOC; LOCC, MCC and TCC have a lesser effect on performance. These results suggest that the amount of consumed energy and time spent are governed by the sheer volume of code to execute rather than the amount of changes introduced or code complexity.[12] If we treat high energy consumption as a defect [55], our results differ from the results seen for functional defects, where LOCC acts as a better predictor of defects than does LOC [56, 57]. This result is also different from the findings of Hindle [4], who found that, in the case of the Firefox web browser, LOC is not correlated with power consumption.

4.5.3 Per-statement Analysis

We provide details of the per-statement analysis in Appendix B.5. In essence, as the level of granularity becomes finer (when we move from workload level to statement level), the amount

[12] Peculiarly, even though the complexity metrics MCC and TCC are strongly correlated with the size metric LOC, the LOC acts as a better predictor of time spent and energy consumed, suggesting that code volume is more important than its complexity.

of variability increases. Our analysis shows that some statements become greener as releases mature—and some not. However, from a practical perspective, one needs to look at the workload holistically, because a database engine user would execute workload as a whole, as recommended by TPC, rather than a particular statement.

4.6 Threats to Validity

In this section, we discuss three groups of threats to the validity of our study: construct, internal and external validity.

4.6.1 Construct Validity

Construct validity is a sort of statistical validity that guarantees that the genuine experimentation and data gathering follow the hypothesis of interest. It is related to whether we measure what we intend to measure.

The construct threats in our study are represented by the following:

- Threat #1: The use of different releases—since MySQL is a mature product, there are hundreds of available versions classified as major and minor releases. The threat here is related to how to select the correct set of versions from all those available.
- Threat #2: The limitation in MySQL behavior (response) if using a single configuration, one engine type, single dataset size and single key buffer size.
- Threat #3: The consistency in the workload used, which affects the results accuracy.
- Threat #4: The experiment results accuracy and availability.
- Threat #5: The choice and calculation of the most appropriate software metrics that may have a potential effect on the database greenness/execution time.

Starting from our main two research questions and in order to eliminate the threats associated with this validity we did the following.

In order to address Threat #1, we chose a subset of three minor releases from all the available major MySQL releases (v.5.0, v.5.1, v.5.5 and v.5.6). These minor releases represent the oldest, intermediate and newest versions within each major release. Choosing these versions helped to gain a broad and clear idea about MySQL's conduct.

In order to address Threat #2 we did the following:

1- We chose two different database storage engines types (MyISAM and InnoDB) in order to get a more general idea about the database engine's behavior.

2- We chose different key buffer sizes (256MB and 1024MB), which helped in examining the various situations for each database engine.

3- We chose different raw data sizes (1 GB and 3 GB), which helped to gain a clear idea about the different responses of the database engine when the raw data was less than the available memory (all data can be cashed in the case of 1 GB) or exceeded the available memory (in the case of 3 GB).

To address Threat #3, we ran the same standard TPC-H workload to be sure that we had a consistent execution in all experiments and accurate results.

To address Threat #4, we did the following:

1- We ensured that all the required measurements were calculated, such as energy consumption and execution time, in addition to all essential system statuses, which were derived by the implementation of the eight experiments.

2- We ran each experiment three times in order to have various readings for the measurements and avoid measurements errors, which helped to ensure the correctness of the data results.

In order to address Threat #5, we examined popular software metrics for all the releases under study; the metrics were calculated using standard tools widely used by researchers and practitioners.

4.6.2 Internal Validity

Internal validity refers to how well our experiments are conducted, since incorrect implementation may affect the output. In order to mitigate this threat, we automated the process of data gathering and analysis, reducing the risk of human error. We created Python scripts for

gathering code metrics from the source code of the database engine. We also created a Python script that profiled workload execution: the script automatically downloads database engines, installs them, creates database objects, generates raw data and loads it into the database, executes workload queries and measures execution time and energy consumption. The results of the experiments as well as the source code metrics were stored in a SQLite[13] database. We then automatically generated the figures and correlation tables using R scripts (which accessed data from the SQLite database). An external reviewer reviewed all the scripts, ensuring their correctness.

4.6.3 External Validity

This type of validity refers to the extent to which the results of our study can be disseminated to the outside world and be used more generally, rather than being limited only to the scope of our study.

We studied two database engines. The generalization to other database engines or other software products is, obviously, not possible. However, the software under study represents a critical case [58] of a relational database management system. Our experimental framework can be applied to other projects with well-designed and controlled experiments.

[13] "SQLite is an in-process library that implements a self-contained, server less, zero-configuration, transactional SQL database engine" [59].

Chapter 5

5 Conclusion and Future Work

The increasing usage of IT systems leads to an increase in energy consumption and, consequently, an increase of CO_2 emissions. Therefore, the creation of green IT systems will lead to a reduction in energy consumption and CO_2 emissions. Most of the existing research focuses on building energy-efficient hardware parts, while software has received little attention in this matter. Software controls hardware and, if not properly designed, can lead to significant power consumption.

Database systems are used widely in the IT field, contributing significantly to energy consumption. Therefore, it is important to understand how to build a green database. Software systems, including databases, evolve over time. Therefore, understanding how the energy consumption changes as database engine evolves is also important.

To reach these objectives, we answered two research questions:

RQ1: How does the energy consumption and execution time of a database engine change as the product matures (from release to release)?

RQ2: Which software metrics affect energy consumption and execution time?

To answer these questions, we performed a case study, measuring the energy consumption and execution time of two database engines—namely, MySQL MyISAM and MySQL InnoDB—across 12 releases on a reference analytic workload, called TPC-H, developed by the Transaction Processing Performance Council.

To automate the measurement process, we developed a framework for 1) measuring the energy consumption and execution time of a database workload and 2) extracting the software metrics under study for each specific MySQL release. Our answers to the research questions are as follows.

RQ1. Our study shows that the MySQL MyISAM engine becomes less green and less efficient as the product matures, leading to increased energy consumption and higher CO_2 emissions. In the case of MySQL InnoDB, the earliest major release is the greenest in 50% (2 out of 4) of our experiments, while the latest major release is the brownest in 75% (3 out of 4) of our experiments. The greenest and the brownest titles are claimed by intermediate releases in the remaining experiments. Our findings differ from the previous work [4] studying a web browser, where greenness increases as the product matured. This suggests that the evolution of greenness is product-specific.

RQ2. Consumed energy and performance are governed mainly by the size of the code base. The code size LOC metric serves as a strong predictor of energy consumption and performance for both database engines. The smaller the code base, the greener and more efficient the database engine is. The code churn (LOCC) and complexity (MCC and TCC) metrics results have a lesser effect on energy consumption and performance. These results differ from previous work [4], where LOC was not a strong predictor of power consumption.

Our findings are of value to both practitioners and theoreticians. Database administrators can use our findings to select a green and fast release of the MySQL database engine. Developers of MySQL database engines can assess the greenness and performance of their product with the help of software metrics. The findings are also of interest to researchers, as they lay a foundation for models predicting the greenness and performance of databases, which, in turn, will aid in developing green software.

In the future, we plan to expand our work by studying other relational, NoSQL, and NewSQL database engines as well as study other enterprise products, such as middleware servers.

Appendices

Appendix A

A.1 List of MySQL Versions under Study

1	v.5.0.15
2	v.5.0.67
3	v.5.0.96
4	v.5.1.30
5	v.5.1.50
6	v.5.1.72
7	v.5.5.10
8	v.5.5.20
9	v.5.5.39
10	v.5.6.10
11	v.5.6.15
12	v.5.6.21

A.2 MySQL Configuration File

The listing below represents configuration file for MySQL engine [47].

```
#
# The MySQL database server configuration file.
#
# You can copy this to one of:
# - "/etc/mysql/my.cnf" to set global options,
# - "~/.my.cnf" to set user-specific options.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html

# This will be passed to all mysql clients
# It has been reported that passwords should be enclosed with ticks/quotes
```

```
# escpecially if they contain "#" chars...
# Remember to edit /etc/mysql/debian.cnf when changing the socket location.
[client]
port            = 3306
#socket         = /var/run/mysqld/mysqld.sock
socket          = /tmp/mysql.thesock
# Here is entries for some specific programs
# The following values assume you have at least 32M ram

# This was formally known as [safe_mysqld]. Both versions are currently parsed.
[mysqld_safe]
#socket         = /var/run/mysqld/mysqld.sock
socket          = /tmp/mysql.thesock
nice            = 0

[mysqld]
#
# * Basic Settings
#
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
#socket         = /var/run/mysqld/mysqld.sock
socket          =/tmp/mysql.thesock
port            = 3306
basedir         = /home/avm/mysql
datadir         = /home/avm/data
tmpdir          = /tmp
#lc-messages-dir        = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 127.0.0.1
#
# * Fine Tuning
#
max_allowed_packet      = 16M
thread_stack            = 192K
thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover          = BACKUP
#
```

```
# * Query Cache Configuration
#
query_cache_limit       = 0  #disable query_cashe; original value = 1M
query_cache_size        = 0  #disable query_cashe; original value = 16M


# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
expire_logs_days        = 10
max_binlog_size         = 100M

[mysqldump]
quick
quote-names
max_allowed_packet      = 16M
[mysql]
#no-auto-rehash # faster start of mysql but no tab completition
# [isamchk]
# key_buffer              = 16M

#
# The values below are uncommented only if running experiments for InnoDB
#
# InnoDB is enabled by default with a 10MB datafile in /var/lib/mysql/.
# Read the manual for more InnoDB related options. There are many!
#
# * Security Features
#
# Read the manual, too, if you want chroot!
# chroot = /var/lib/mysql/
#
# For generating SSL certificates I recommend the OpenSSL GUI "tinyca".
#
# ssl-ca=/etc/mysql/cacert.pem
# ssl-cert=/etc/mysql/server-cert.pem
# ssl-key=/etc/mysql/server-key.pem
#loose-innodb-trx=0
#loose-innodb-locks=0
#loose-innodb-lock-waits=0
#loose-innodb-cmp=0
#loose-innodb-cmp-per-index=0
#loose-innodb-cmp-per-index-reset=0
```

```
#loose-innodb-cmp-reset=0
#loose-innodb-cmpmem=0
#loose-innodb-cmpmem-reset=0
#loose-innodb-buffer-page=0
#loose-innodb-buffer-page-lru=0
#loose-innodb-buffer-pool-stats=0
#loose-innodb-metrics=0
#loose-innodb-ft-default-stopword=0
#loose-innodb-ft-inserted=0
#loose-innodb-ft-deleted=0
#loose-innodb-ft-being-deleted=0
#loose-innodb-ft-config=0
#loose-innodb-ft-index-cache=0
#loose-innodb-ft-index-table=0
#loose-innodb-sys-tables=0
#loose-innodb-sys-tablestats=0
#loose-innodb-sys-indexes=0
#loose-innodb-sys-columns=0
#loose-innodb-sys-fields=0
#loose-innodb-sys-foreign=0
#loose-innodb-sys-foreign-cols=0
#innodb_buffer_pool_size=50MB
```

A.3 Schema of TPC-H Tables

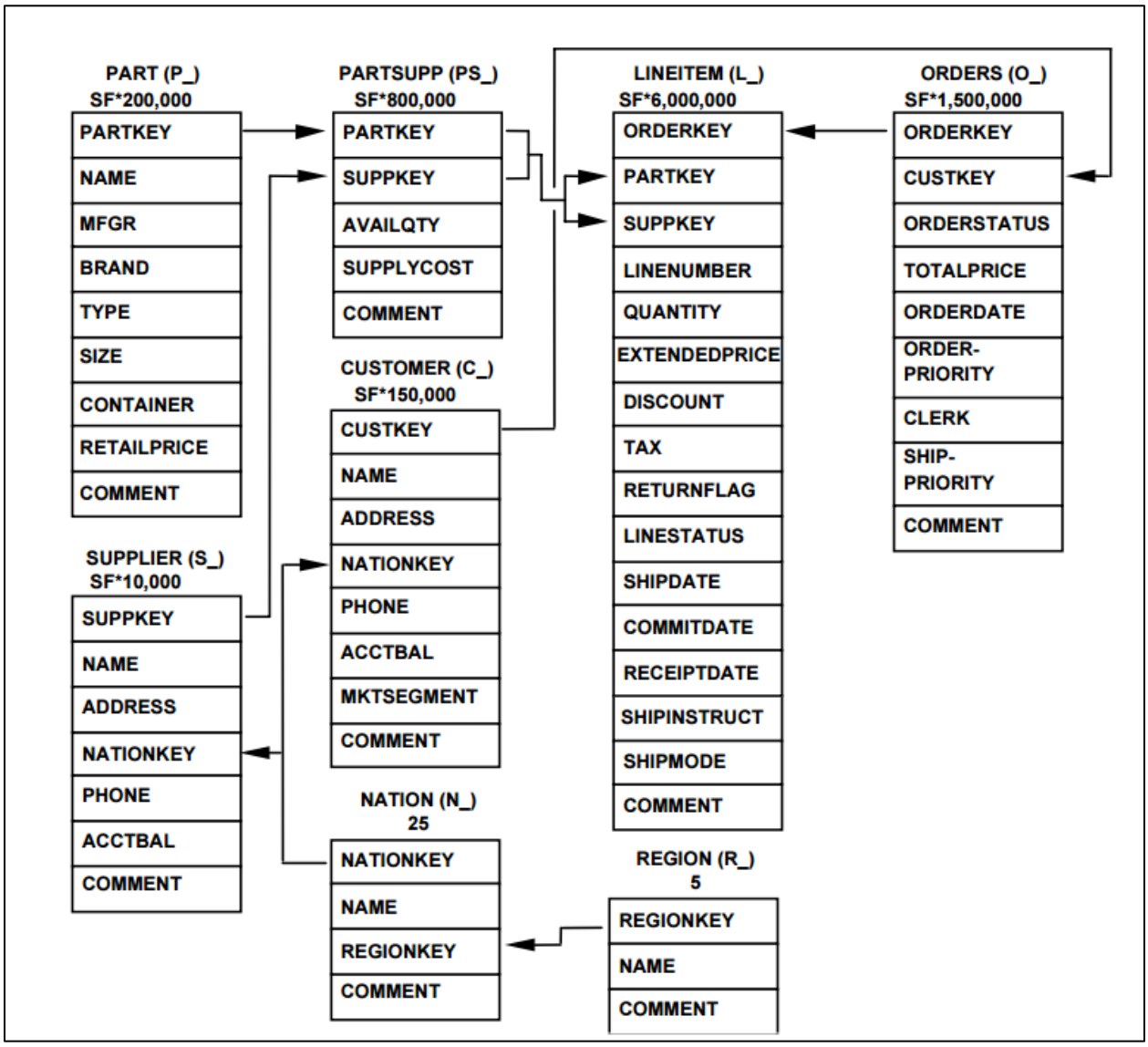


Figure A.1: The TPC-H Schema; SF refers to the scale factor of the generated benchmark data (reproduced from [49]).

A.4 MySQL key Buffer and The Database Performance

The key buffer or the key cache is representing the requested memory amount; the value for this key can be set by using the key_buffer_size variable. The engine assigns as much memory as needed, up to the value of key_buffer_size. Index block is defined as a connecting unit of access to the MyISAM index files for MyISAM tables. These blocks are buffered (stored in the key cache) and are used usually by all related and became available to all processes that use MySQL [60].

The key_buffer_size variable represents the buffer size that MySQL MyISAM[14] uses to keep indexes in memory. The cache keeps index blocks in memory in order to avoid reading from the hard disk repeatedly. The key_buffer_size is considered as one of the most significant variables to tune MySQL database, in order to increase database performance [61] [60]. The value of this variable can be increased to get better index management for concurrent reads and writes. If the main purpose of a system is to run MySQL with MyISAM storage engine, in this case, the recommend value for this variable is 25% of the total memory available on the machine. When this value is excessively large, there is possibility that the system will become slow. This happens because MySQL depends on the operating system to carry out file system caching for data reads, therefore some space needs to be left for the file system caching.

It is very important to tune the MySQL default installation to improve its performance and the key buffer cache is an essential variable in this tuning process. Increasing the key_buffer_size can also result in a highly reduced number of major page faults [61]. When data from table index block needs to be accessed, a check is implemented by the server to see whether it exists in the key cache. If it is available, the server accesses data in the key cache instead of the disk. Therefore, the server reads and writes from or into the cache instead of reading from or writing to disk. If it is not available, the server reads the data from disk and caches it as well for future use. If the cache is full then the newly read data replaces some old data in the cache [61].

[14] InnoDB engine, uses innodb_buffer_pool_size variable instead of key_buffer_size, to control the size of its cache.

A.5: MyISAM and InnoDB Comparison

MySQL database has two major types of table storage engines, InnoDB and MyISAM. The following list summarizes the differences of features and performance between these two engines (reproduced from [47] and [48]):

1. MyISAM is simple while InnoDB is more complex.
2. MyISAM is the old engine while InnoDB is newer.
3. InnoDB is an ACID compliant and therefore fully transactional with ROLLBACK and COMMIT while MyISAM is not.
4. MyISAM implements table-level lock while InnoDB implements row-level lock for insert and update processes.
5. InnoDB has transactions while MyISAM does not.
6. InnoDB has foreign keys and relationship constraints while MyISAM does not.
7. MyISAM is considered loose in data integrity while InnoDB is stricter.
8. MyISAM is poor at recovering data integrity at system crashes while InnoDB has better crash recovery.
9. MyISAM has no ordering in storage of data while InnoDB stored Row data in pages in primary key order.
10. MyISAM has full-text search index while InnoDB has not.

Appendix B

B.1 MyISAM Experiments 2-4 for Per-run Analysis (RQ1)

B.1.1 Experiment 2: Engine = MyISAM, Key buffer size = 256M, Database Size = 3GB

In this experiment, we loaded 3GB of raw data into the database and allowed the database engine to use 256MB of memory for its buffer. Figure B.1 shows that, overall, energy consumption and execution time increased as MySQL matured. This is especially pronounced as we move from one major release to the next one (e.g., from v.5.0 to v.5.1 or from v.5.1 to v.5.5). Fitting the data with a linear function (using linear regression) explains most of the data variability: R2 for energy and execution time are equal to 0.87 and 0.84, accordingly. The slopes of the linear functions (denoted by *b* in Figure B.1) are positive.

However, when we examined the data per major release on Figure B.1, the results are less univocal. In the case of major release v.5.0, energy consumption and execution time increased with "aging" of MySQL. The slopes of linear functions fitting both datasets are positive; the functions explain most of the variability: R2 = 0.82 and 0.84, accordingly. The remaining three major releases v.5.1, v.5.5, and v.5.6 do not exhibit a pronounced trend. Linear fit cannot adequately describe the trends (R2 in all cases is less than 0.1, i.e., very low).

Eyeballing of the Figure B.1 suggests that for major release v.5.1 energy consumption and execution time may follow V pattern: energy consumption and execution time of an intermediate release v.5.1.50 is lower than of an early release v.5.1.30; energy consumption and execution time of the mature release v.5.1.72 is higher than for intermediate release v.5.1.50. However, the pattern is not very strong if we take into account the variability of the measurements per each run. This is further confirmed by the high spread of the confidence interval lines on Figure B.1.

Energy consumption and execution time for major release v.5.5 may follow Λ pattern (an inverse of the V pattern): energy consumption and execution time of an intermediate release v.5.5.20 is higher than of an early release v.5.5.10; energy consumption and execution time of the mature release v.5.5.39 is lower than for intermediate release v.5.5.20 However, as in the case of the major release v.5.1, the pattern is not very strong, since the variability of the measurements per each run is high. In case of major release v.5.6, there may exist a slight

positive trend (based on the slope value). Moreover, the variability of the data is high (R2 equal 0.05 and 0.04) in both cases accordingly, suggesting that the trend is stable rather than positive.

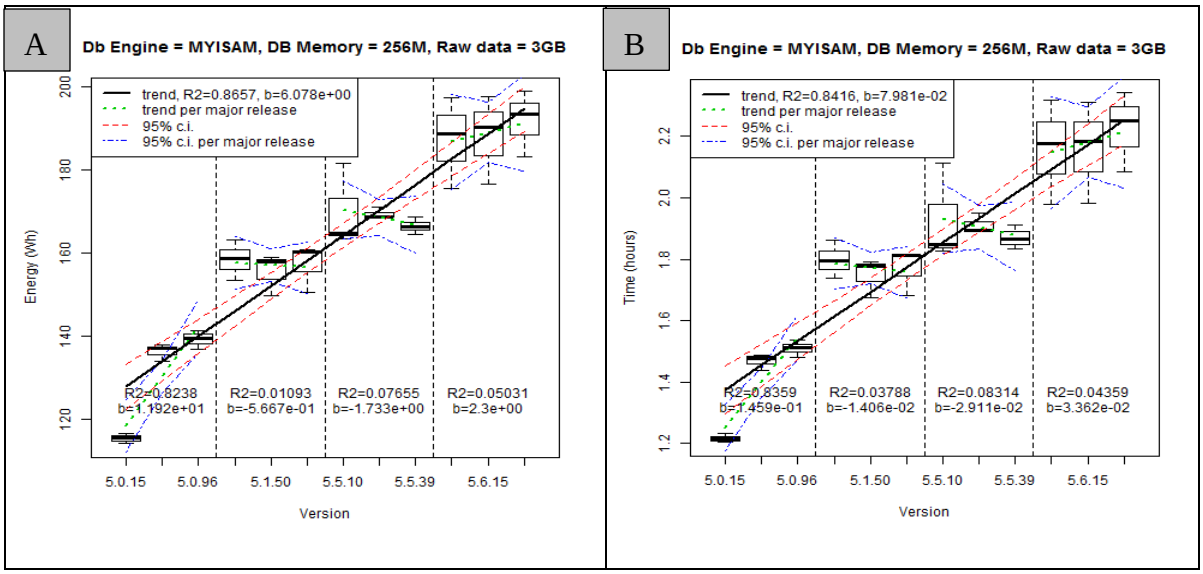


Figure B.1: Experiment 2; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions. On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

B.1.2 Experiment 3: Engine = MyISAM, Key buffer size =1024M, Database Size= 1GB

In this experiment, we loaded 1GB of raw data into the database and allowed the database engine to use 1024MB of memory for its buffer. Figure B.2 shows that, overall, energy consumption and execution time grow as MySQL matured. This is especially pronounced as we move from one major release to the next one (e.g., from v.5.0 to v.5.1 or from v.5.1 to v.5.5). Fitting the data with a linear function (using linear regression) explains most of the data variability: R2 for energy and execution time are equal to 0.86 and 0.85, accordingly. The slopes of the linear functions (denoted by *b* in Figure B.2) are positive in the two cases.

However, when we examine the data per major release on Figure B.2, the results are non-stable and not following the same pattern. Energy consumption and execution time for major release v.5.0 may follow Λ pattern: energy consumption and execution time of an intermediate release v.5.0.67 is higher than of an early release v.5.0.15; energy consumption and execution time of the mature release v.5.0.96 is lower than for intermediate release v.5.0.67. However, the pattern is not very strong as the variability of the data is high; R2 = 0.45 and 0.18 in the two datasets accordingly.

The slopes of linear functions fitting both datasets are positive. The data for the two major releases v.5.1 and v.5.5 exhibit a stable trend (the trend line is almost horizontal[15]). Note that data variability makes it difficult to deduce the trend direction (all $R2 \leq 0.14$ for both energy consumption and execution time).

On the other hand, the slopes of linear functions fitting both datasets are negative in first dataset (energy), and negative in v.5.1 while it's positive in v.5.5 in time dataset which represent non-stable slope. In case of major release v.5.6, there is a clear negative trend, based on the slope value which is equal -3.33 and -3.05 in both datasets, accordingly. Additionally, by looking to this major release versions in Figure B.2 we can easily recognize this negative trend. Moreover, the variability of the data is low as the value of R2 equals 0.77 and 0.83 in both dataset accordingly, suggesting that the trend is stable within this release.

[15] The slope coefficient *b* values for v.5.1 and v.5.5 are approximately 10 times smaller than for v.5.0.

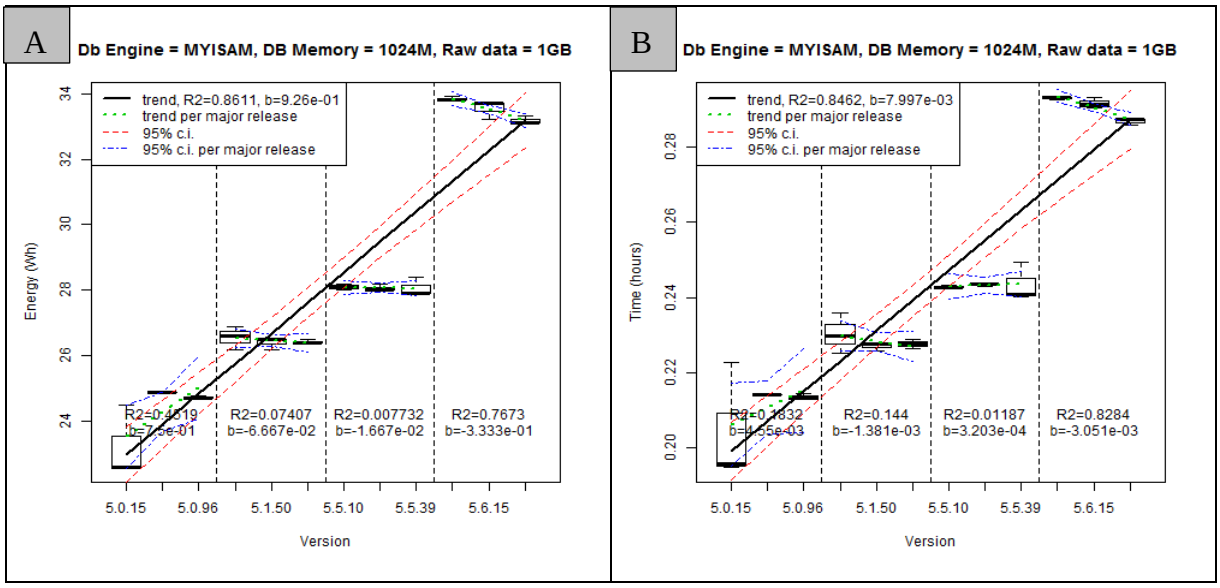


Figure B.2: Experiment 3; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions .On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

B.1.3 Experiment 4: Engine = MyISAM, Key buffer size =1024M, Database Size= 3GB

In this experiment, we loaded 3GB of raw data into the database and allowed the database engine to use 1024MB of memory for its buffer. Figure B.3 shows that, overall, energy consumption and execution time grow as MySQL matured (as confirmed by the slopes). This is especially pronounced as we move from one major release to the next one. Fitting the data with a linear function (using linear regression) explains some of the data variability: R2 for energy and execution time = 0.60 and 0.56, accordingly. However, the R2 values are lower than in the case of Experiments 2 and 3. This can be explained by the fact that the variability in measurements for runs a given release is higher for Experiment 4, in comparison with Experiments 2 and 3 (reflected by the "size" of each single box in Figure B.3). We conjecture that this can be explained as follows. Unlike 1GB dataset, 3GB of data cannot fit into memory, leading to non-deterministic reads from the hard drive, hence the increase of variability.

However, when we examine the data per major release on Figure B.3, the results are following different patterns. Energy consumption and execution time for major release v.5.0 follow a clear positive pattern. However, the pattern is not very strong as the variability of the data is high, R2 equals 0.33 and 0.30 in the two datasets accordingly. The slopes of linear

functions fitting both datasets are positive. The remaining three major releases v.5.1, v.5.5, and v.5.6 do not exhibit a pronounced trend. Linear functions show slight negative or positive trends. However, due to variability in the data, the R2 values are extremely low ($R2 \leq 0.004$) making it impossible to draw a conclusion on the trend direction (as further confirmed by the wide confidence intervals of the linear models shown in Figure B.3).

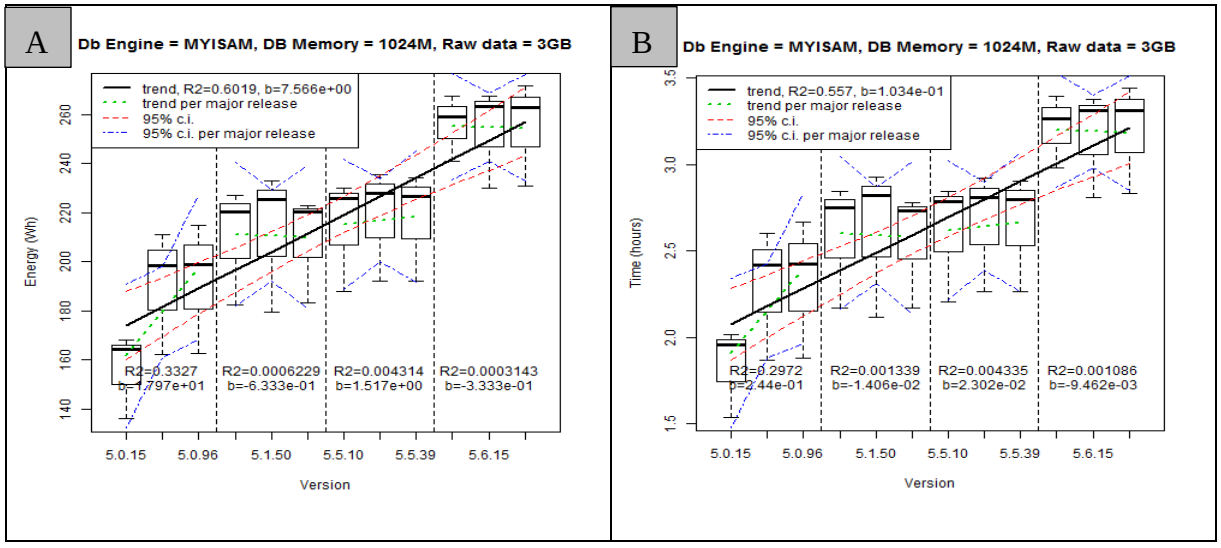


Figure B.3: Experiment 4; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions .On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

B.2 InnoDB Experiments 6-8 for Per-run Analysis (RQ1)

B.2.1 Experiment 6: Engine = InnoDB, Key buffer size = 256M, Database size = 3GB

In this experiment, we loaded 3 GB of raw data into the database and allowed the database engine to use 256MB of memory for its buffer. Figure B.4 shows that, as with previous experiments, energy consumption increased and performance decreased, as the engine matured. However, the variability is higher and the trend is less pronounced (R2=0.68 for energy consumption and R2=0.70 for execution time).

The energy consumption and execution time are not following the same pattern in all releases. In release v.5.0 the trend shows increase in both energy and time spent when versions improved, if we take in consideration the R2 values which are more than 0.75 in the two datasets in addition to the positive slope values which confirm this positive relation. Releases v.5.1 and

v.5.5 follow Λ pattern: energy consumption and execution time of an intermediate release is higher than of an early release; energy consumption and execution time of the mature release is lower than for intermediate release in the two cases. However, the pattern is not very strong as the variability of the data is very high, R2 is very small value in the two datasets for the two releases v.5.1 and v.5.5. In release v.5.6 the trend follows obvious positive relation and the R2 is more than 0.94 in both datasets with positive slope values that confirm energy consumption and time spent are increasing when the versions developed.

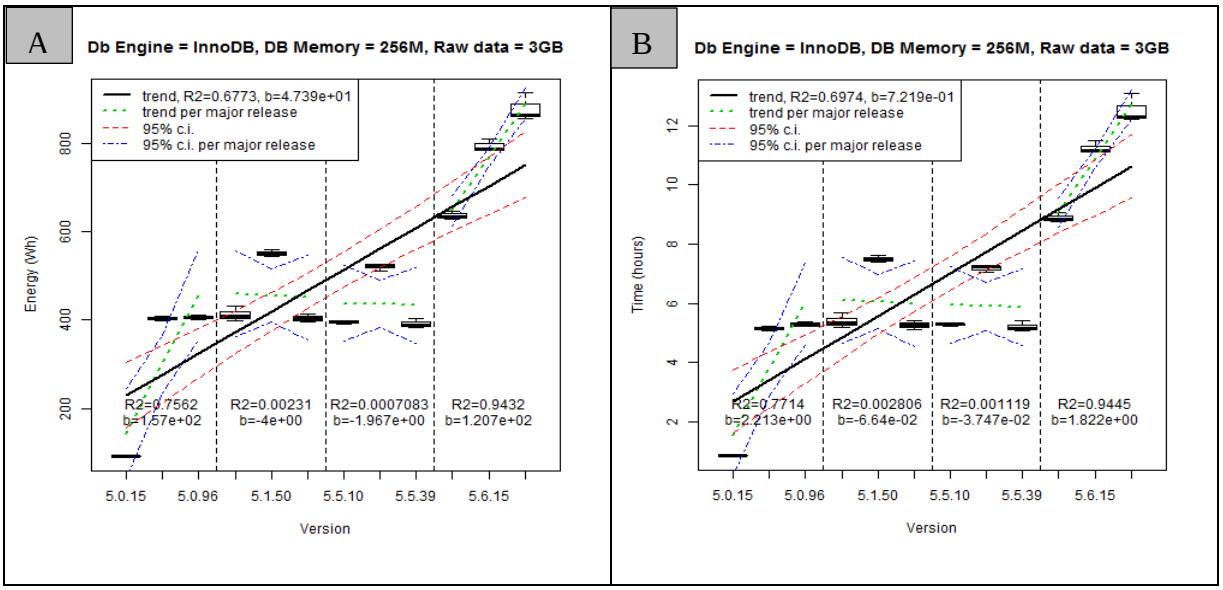


Figure B.4: Experiment 6; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions .On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

B.2.2 Experiment 7: Engine = InnoDB, Key buffer size = 1024M, Database size= 1GB

In this experiment, we loaded 1GB of raw data into the database and allowed the database engine to use 1024MB of memory for its buffer. Figure B.5 shows that, overall, energy consumption and execution time following a clear negative pattern as MySQL mature. Fitting the data with a linear function (using linear regression) does not explain any of the data variability: R2 for energy and execution time are more than to 0.08 in both datasets. The slopes of the linear functions (denoted by *b* in Figure B.5) are negative in the two cases; which represents a negative trend in general. However, when we examine the data per major release on Figure B.5, the results are following different patterns.

Energy consumption and execution time for major release v.5.0 follow negative pattern: energy consumption and execution time are decrease when version mature. Nevertheless, the pattern is considered strong as the R2 values are 0.77 and 0.78 in both datasets accordingly. The slopes of linear functions fitting both datasets are negative which reflects this negative trend.

The remaining three major releases v.5.1, v.5.5, and v.5.6 do not exhibit a fixed trend. Linear function gives various description for the trends; R2 has variable values in both datasets which represents not fixed pattern.

Examining Figure B.5 suggests that for major release v.5.1 energy consumption and execution time may follow positive pattern and the pattern is considered strong if we take into account the variability of the measurements per each run; R2 equals 0.88 and 0.91 in the two datasets accordingly. The slopes of linear functions fitting both datasets are positive which reflects this positive trend.

Energy consumption and execution time for major release v.5.5 Follow V pattern: energy consumption and execution time of an intermediate release v.5.5.20 is lower than of an early release v.5.5.10; energy consumption and execution time of the mature release v.5.5.39 is higher than for intermediate release v.5.5.20. Since variability of the measurements per each run is high; R2 equals 0.38 and 0.62 in two datasets accordingly. On the other hand, the slopes of linear functions fitting both datasets are positive in both cases which reflects a positive trend.

In case of major release v.5.6, there is a clear negative trend based on the slope value which is negative in both datasets. Additionally, by looking to this major release's versions in

Figure B.5 we can easily recognize this negative trend. Moreover, the variability of the data is not fixed as the values of R2 are equal 0.21 and 0.09 in both datasets accordingly.

In the case of Experiments 7 and 8, as can be seen from Figures B.5 and B.7, the values of energy consumption and time spent in the case of v.5.0.15 are dramatically different from the values of other versions in a given release. We conjecture that functionality of InnoDB engine in this release behaves abnormally, leading to anomalous data. If we remove these anomalous readings, as can be seen in Figures B.6 and B.8, the results become consistent with the results of previous experiments.

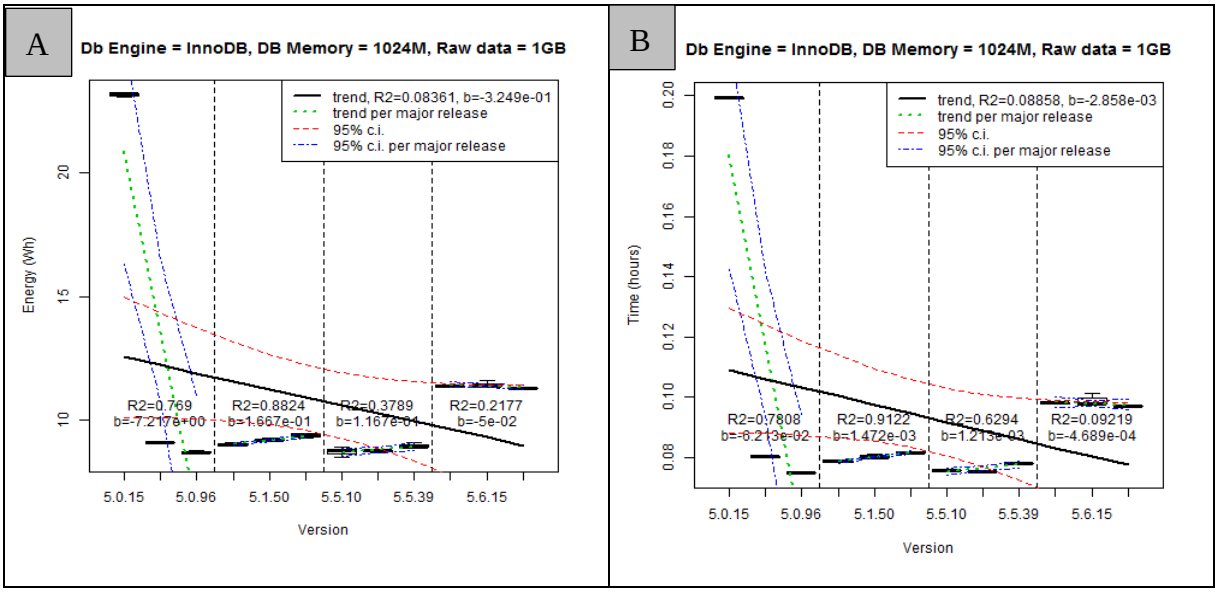


Figure B.5: Experiment 7; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions .On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

Figure B.6 shows the energy consumption and time spent in release v.5.0 (Experiment 7) after removing the outlier release (v.5.0.15). The trend in both cases is negative, as confirmed by slope values in Figure B.6 (A and B). The trend is considered strong, as R2 > 0.97 for both datasets.

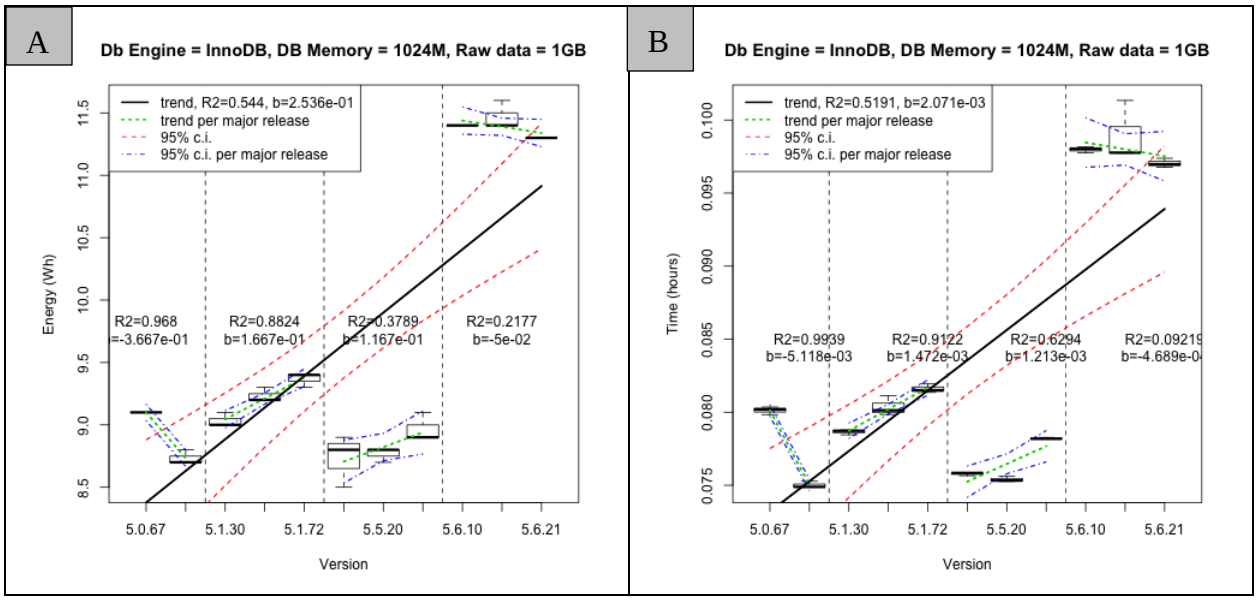


Figure B.6: Experiment 7 without outlier version; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions .On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

B.2.3 Experiment 8: Engine = InnoDB, Key buffer size =1024M, Database size= 3GB

In this experiment, we loaded 3 GB of raw data into the database and allowed the database engine to use 1024MB of memory for its buffer. Figure B.7 shows that generally the energy consumption and execution time are not following the same pattern in all release, the R2 values are more than 0.35 in the two datasets and the slope is positive in both cases.

In release v.5.0 the trend shows increase in both energy consumption and execution time when versions improved, if we take in consideration the R2 values which are more than 0.75 in the two datasets in addition to the positive slope values as well, confirming this positive relation.

In releases v.5.1, v.5.5, and v.5.6, the pattern may follow ∨ shape: energy consumption and execution time of an intermediate release is lower than of an early release; energy consumption and execution time of the mature release is higher than for intermediate release in all the three releases. However, the pattern is not very strong as the variability of the data is very high, R2 has very small values in the two datasets for the three releases. The slope values for v.5.5 and v.5.6 are negative which confirm a negative relation while in version v.5.1 it's positive, showing that there may be a positive relation.

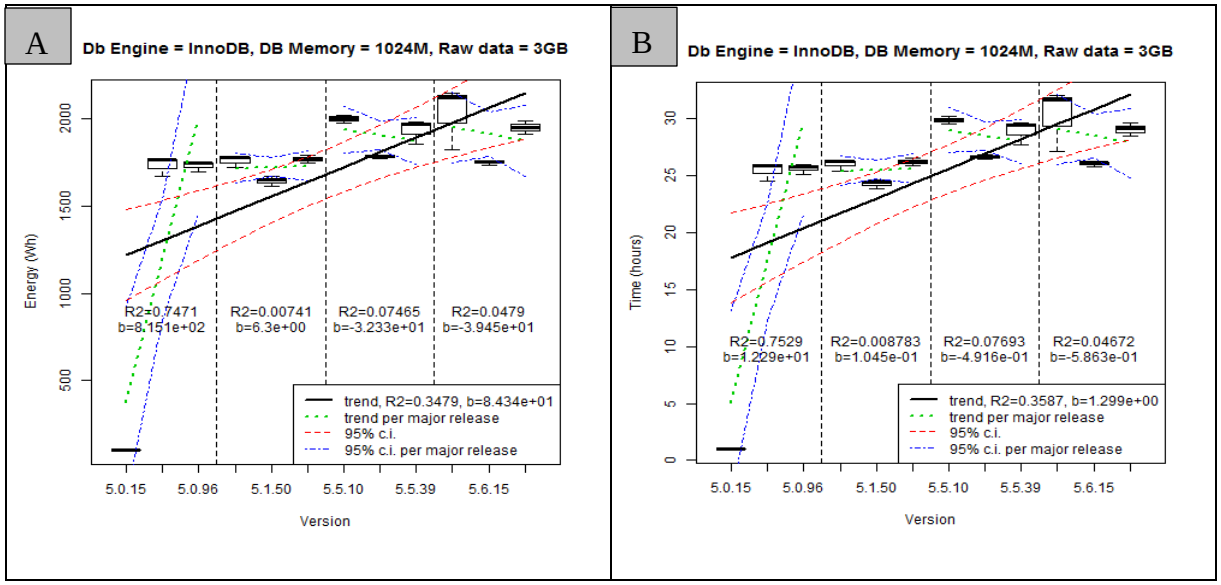


Figure B.7: Experiment 8; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions .On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

As we discussed in Experiment 7, regarding the outlier release (v.5.0.15), Figure B.8 shows the energy consumption and time spent in release v.5.0 after removing the outlier release .The trend in the two datasets is negative by eyeballing Figure B.8 (A and B), in addition to the slope value is negative in energy dataset while it's positive in time dataset, showing not stable trend. The trend is considered weak as R2 value is less than or equals to 0.02 in the two datasets.

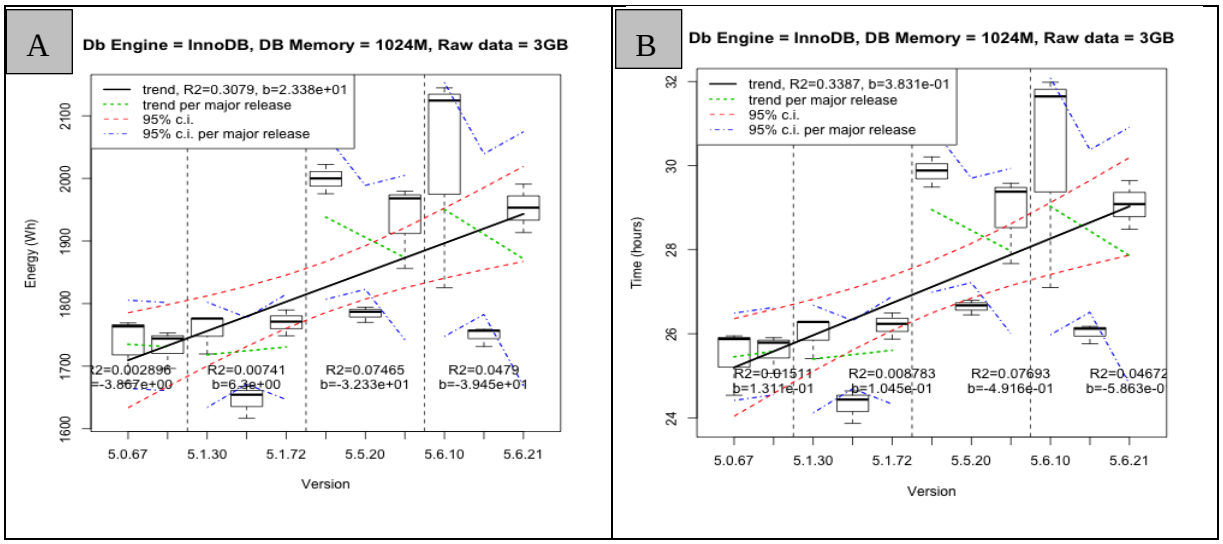


Figure B.8: Experiment 8 without the outlier version; subplot A is box plot of energy consumed against MySQL's versions; subplot B is box plot of execution time against MySQL's versions .On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

B.3 MyISAM Experiments 2-4 Per-run Analysis (RQ2)

B.3.1 Experiment 2: Engine = MyISAM, Key buffer size = 256M, Database size = 3GB

The following scatter graphs (Figures B.9 and B.10) represent the relation between the energy consumption (or time spent) and the different software metrics we used. By examining the relation between the size of code (represented by LOC metric for all the versions under study) and the energy consumed, we found a positive linear relation between the variables: as can be seen from Figure B.9-A, R2=0.51, and slope coefficient *b* is positive. This is further confirmed by a strong (ρ=0.72) correlation between the variables, as shown in Table 4.9.

As can be seen from Figures B.9-C and B.9-D, and Table 4.9, there exists a weak linear relation between TCC (MCC) and consumed energy: R2=0.04 in both cases with a positive slope, and the correlation coefficient is equal to ρ=0.19 (ρ=0.20). As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results.

We found no correlation between LOCC and consumed energy. As can be seen from Figure B.9-B and Table 4.9, R2=4.30 e-05 and ρ=0.01.

Even though the relation between LOCC and consumed energy is low, there exists a strong positive linear relation between LOCC and the change in energy consumption. As can be seen from Figure B.9-E and Table 4.10, the variability of the data is high (R2=0.60), but the correlation is moderate (ρ=0.77).

Experiment 2's data, related to time spent (shown in Table 4.11), are identical to the energy consumption data in Table 4.9 as a result of the almost perfect correlation between energy and time (discussed in Section 4.2). Therefore, the LOC metric has a strong correlation with time spent, MCC and TCC have weak correlation with time spent and no correlation is found between the LOCC metric and time spent. Table 4.12 shows a strong correlation (ρ=0.82) between changes in time with the LOCC metric.

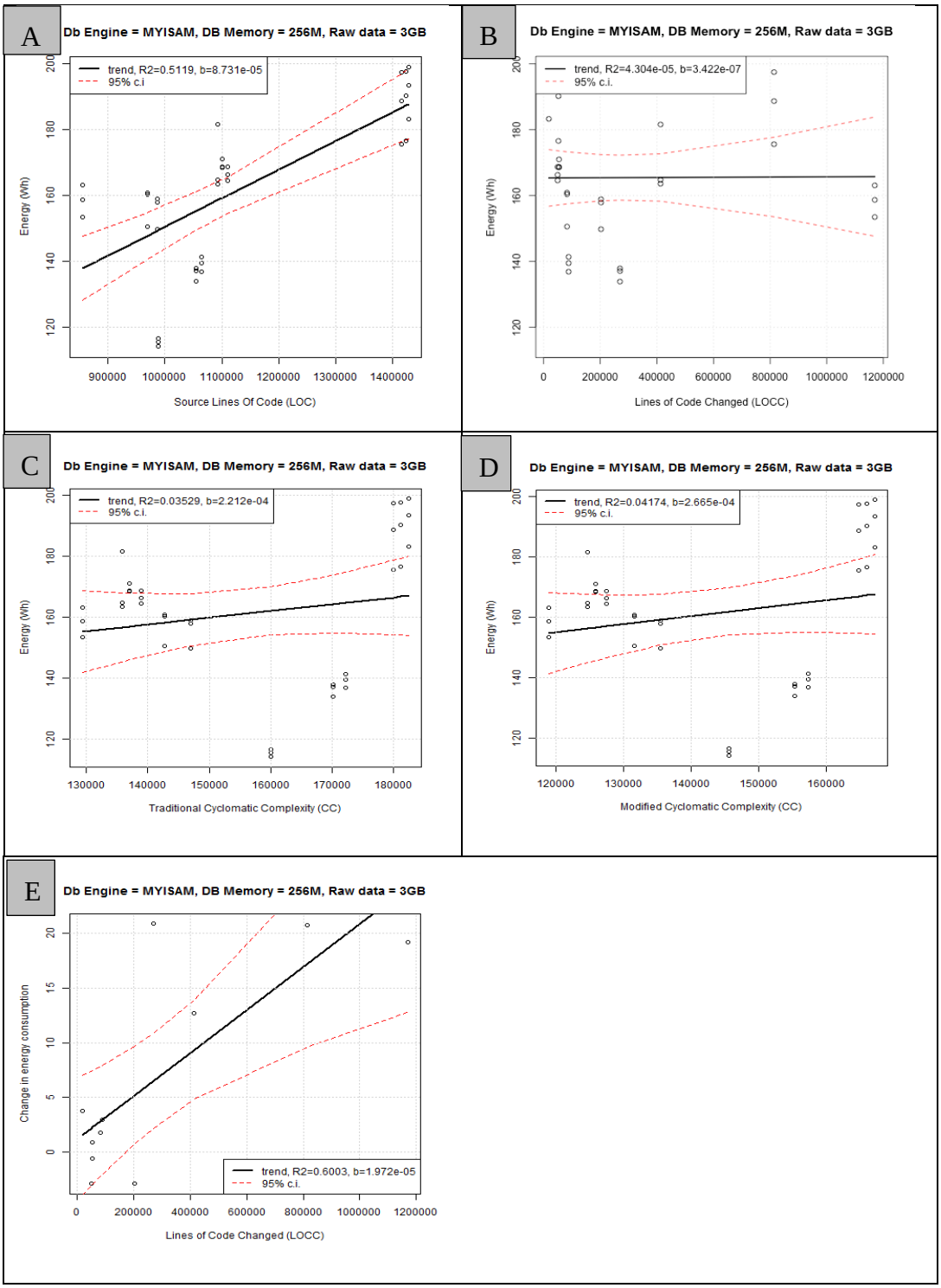


Figure B.9: Relation between energy consumed and various software metrics for Experiment 2. Subplot A is a scatterplot of LOC against energy; subplot B is a scatterplot of LOCC against energy; subplot C is a scatterplot of TCC against energy; Subplot D is a scatterplot of MCC against energy; Subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

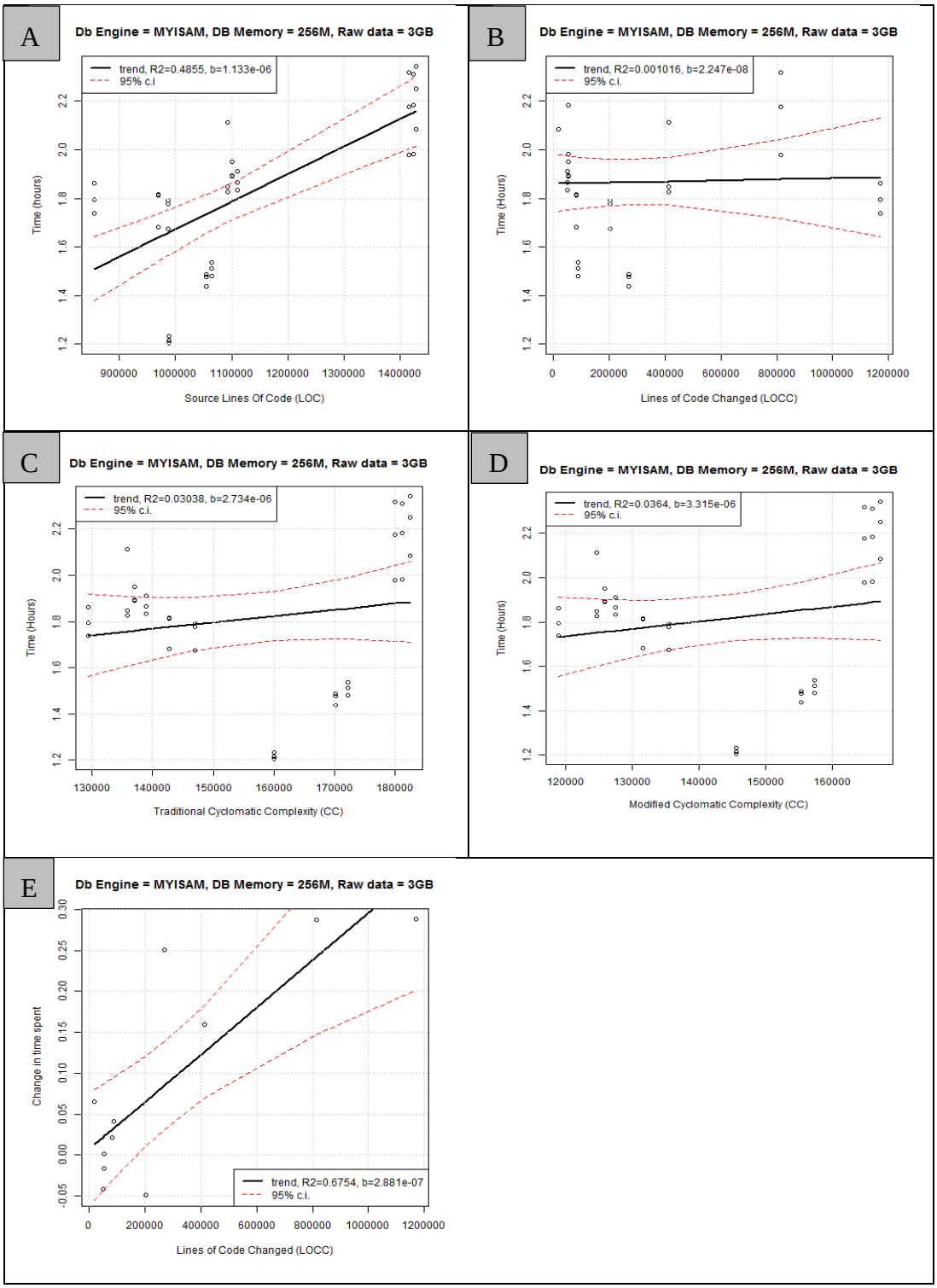


Figure B.10: Relation between time spent and various software metrics for Experiment 2. Subplot A is a scatterplot of LOC against time; subplot B is a scatterplot of LOCC against time; subplot C is a scatterplot of TCC against time; Subplot D is a scatterplot of MCC against time; Subplot E is a scatterplot of LOCC against change in time consumption. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.3.2 Experiment 3: Engine = MyISAM, Key buffer size =1024M, Database size= 1GB

The following scatter plots (Figures B.11 and B.12) represent the relation between the energy consumption (time spent) and the different software metrics we used. By examining the relation between the size of code (represented by the LOC metric for all the versions under study) and the energy consumed, we found a strong positive linear relation between the variables: as can be seen from Figure B.11-A, R2=0.80, and slope coefficient *b* is positive. This is further confirmed by a strong (ρ=0.89) correlation between the variables, as shown in Table 4.9.

As can be seen from Figures B.11-C and B.11-D and Table 4.9, there exists a moderate positive linear relation between TCC (MCC) and consumed energy: ρ=0.46 (ρ=0.48); even though linear model cannot properly explain data variability, since R2=0.22 (R2=0.23). As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results. We found no correlation between LOCC and consumed energy. As can be seen from Figure B.11-B and Table 4.9, R2=0.0003 and ρ=0.02.

Even though the relation between LOCC and consumed energy is low, there exists a moderate positive linear relation between LOCC and the change in energy consumption. As can be seen from Figure B.11-E and Table 4.10, the variability of the data is high (R2=0.53), but the correlation is strong (ρ=0.73).

Experiment 3's data, related to time spent (shown in Table 4.11), are identical to the energy consumption data in Table 4.9 as a result of the almost perfect correlation between energy and time (discussed in Section 4.2). Therefore, the LOC metric has a strong correlation with time spent, MCC and TCC have moderate correlation with time spent and no correlation is found between the LOCC metric and time spent. Table 4.12 shows a strong correlation (ρ=0.75) between changes in time with the LOCC metric.

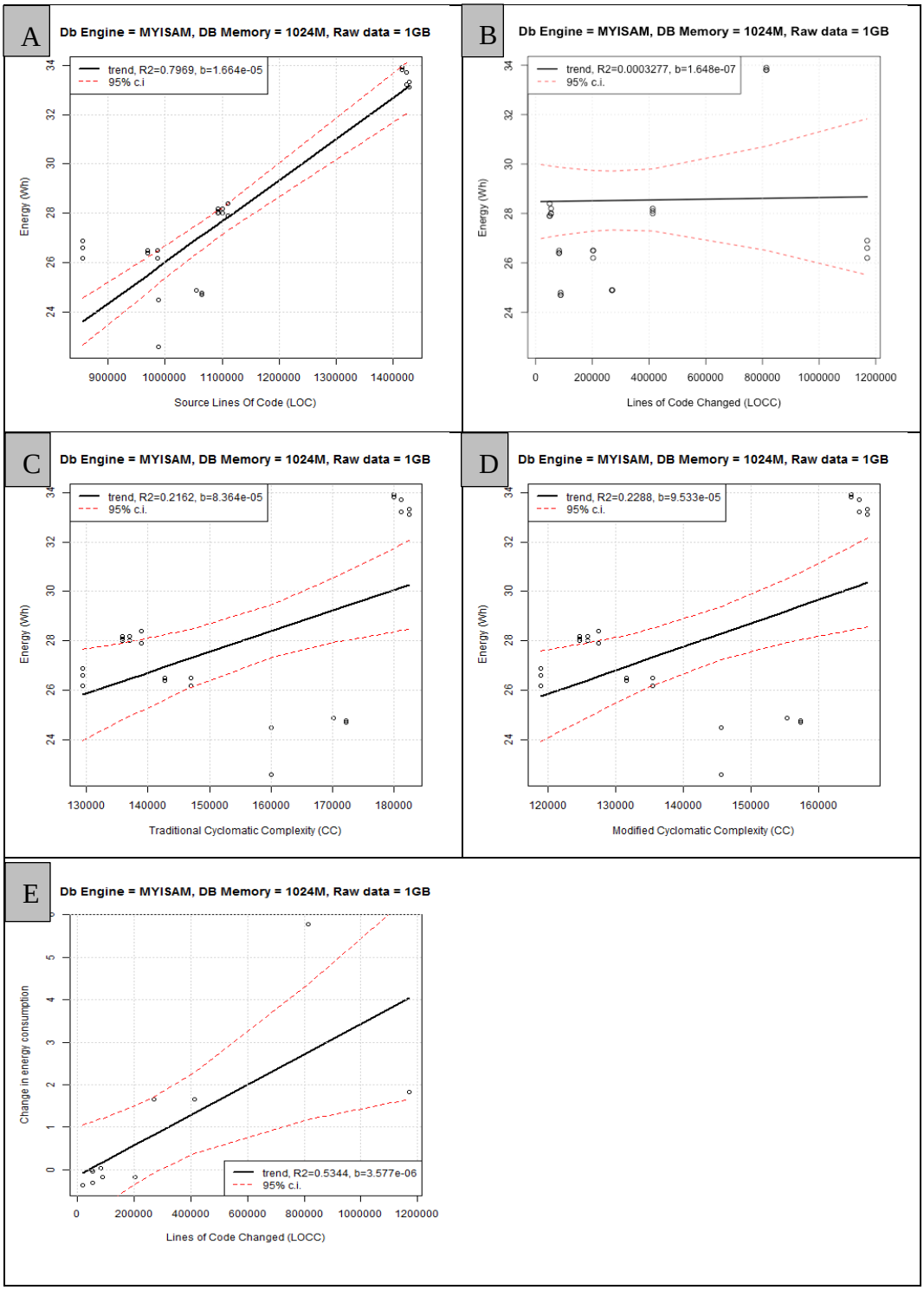


Figure B.11: Relation between energy consumed and various software metrics for Experiment 3. Subplot A is a scatterplot of LOC against energy; subplot B is a scatterplot of LOCC against energy; subplot C is a scatterplot of TCC against energy; Subplot D is a scatterplot of MCC against energy; Subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

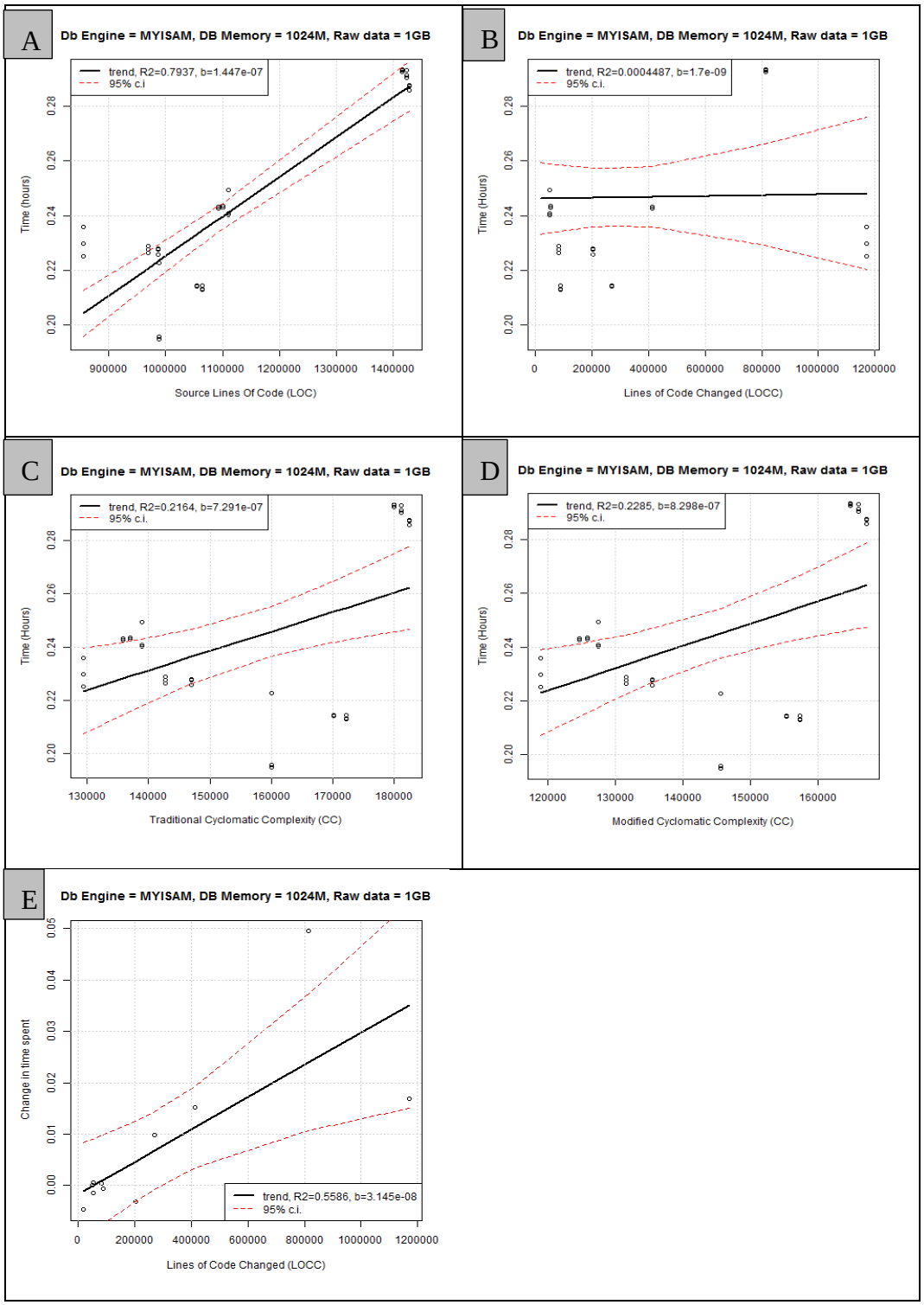


Figure B.12: Relation between time spent and various software metrics for Experiment 3. Subplot A is a scatterplot of LOC against time; subplot B is a scatterplot of LOCC against time; subplot C is a scatterplot of TCC against time; Subplot D is a scatterplot of MCC against time; Subplot E is a scatterplot of LOCC against change in time spent. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.3.3 Experiment 4: Engine = MyISAM, Key buffer size =1024M, Database size= 3GB

The following scatter graphs (Figures B.13 and B.14) represent the relation between the energy consumption (time spent) and the different software metrics we used. By examining the relation between the size of code (represented by the LOC metric for all the versions under study) and the energy consumed, we found a moderate positive linear relation between the variables: as can be seen from Figure B.13-A, R2= 0.43, and slope coefficient *b* is positive. This is further confirmed by a moderate (ρ=0.66) correlation between the variables, as shown in Table 4.9.

As can be seen from Figures B.13-C and B.13-D and Table 4.9, there exists a weak positive linear relation between TCC (MCC) and consumed energy: R2 is less than 0.09 in both cases with a positive slope, and the correlation coefficient is equal to ρ=0.28 (ρ=0.30). As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results.

We found no correlation between LOCC and consumed energy. As can be seen from Figure B.13-B and Table 4.9, R2 has a very small positive value and ρ=0.02.

Even though the relation between LOCC and consumed energy is low, there exists a positive linear relation between LOCC and the change in energy consumption. As can be seen from Figure B.13-E and Table 4.10, the variability of the data is high (R2=0.41), but the correlation is moderate (ρ=0.64).

Experiment 4's data, related to time spent (shown in Table 4.11), are identical to the energy consumption data in Table 4.9 as a result of the almost perfect correlation between energy and time (discussed in Section 4.2). Therefore, the LOC metric has a moderate correlation with time spent, MCC and TCC have weak correlation with time spent and no correlation is found between the LOCC metric and time spent. Table 4.12 shows a moderate correlation (ρ=0.66) between changes in time with the LOCC metric.

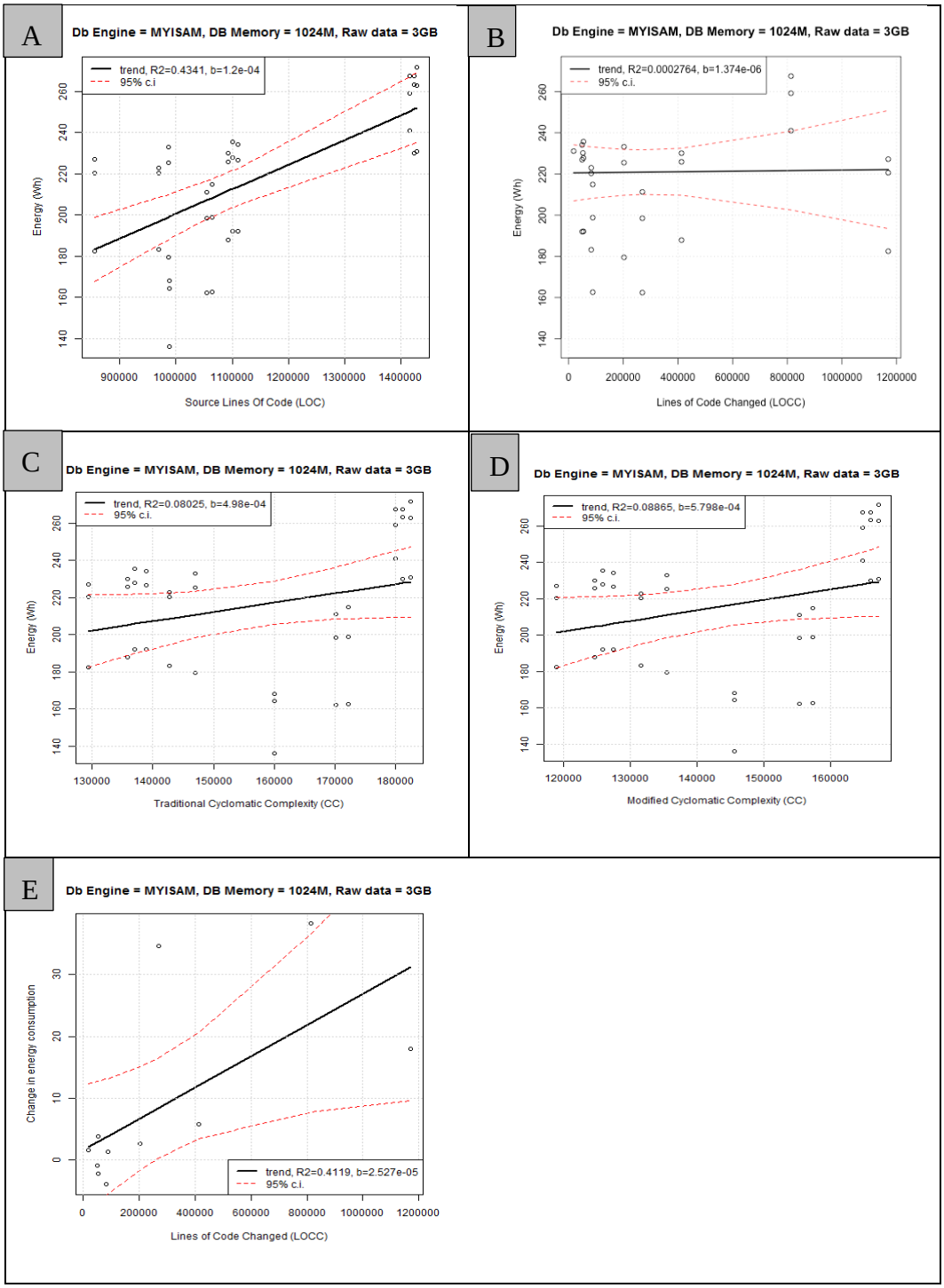


Figure B.13: Relation between energy consumed and various software metrics for Experiment 4. Subplot A is a scatterplot of LOC against energy; subplot B is a scatterplot of LOCC against energy; subplot C is a scatterplot of TCC against energy; Subplot D is a scatterplot of MCC against energy; Subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

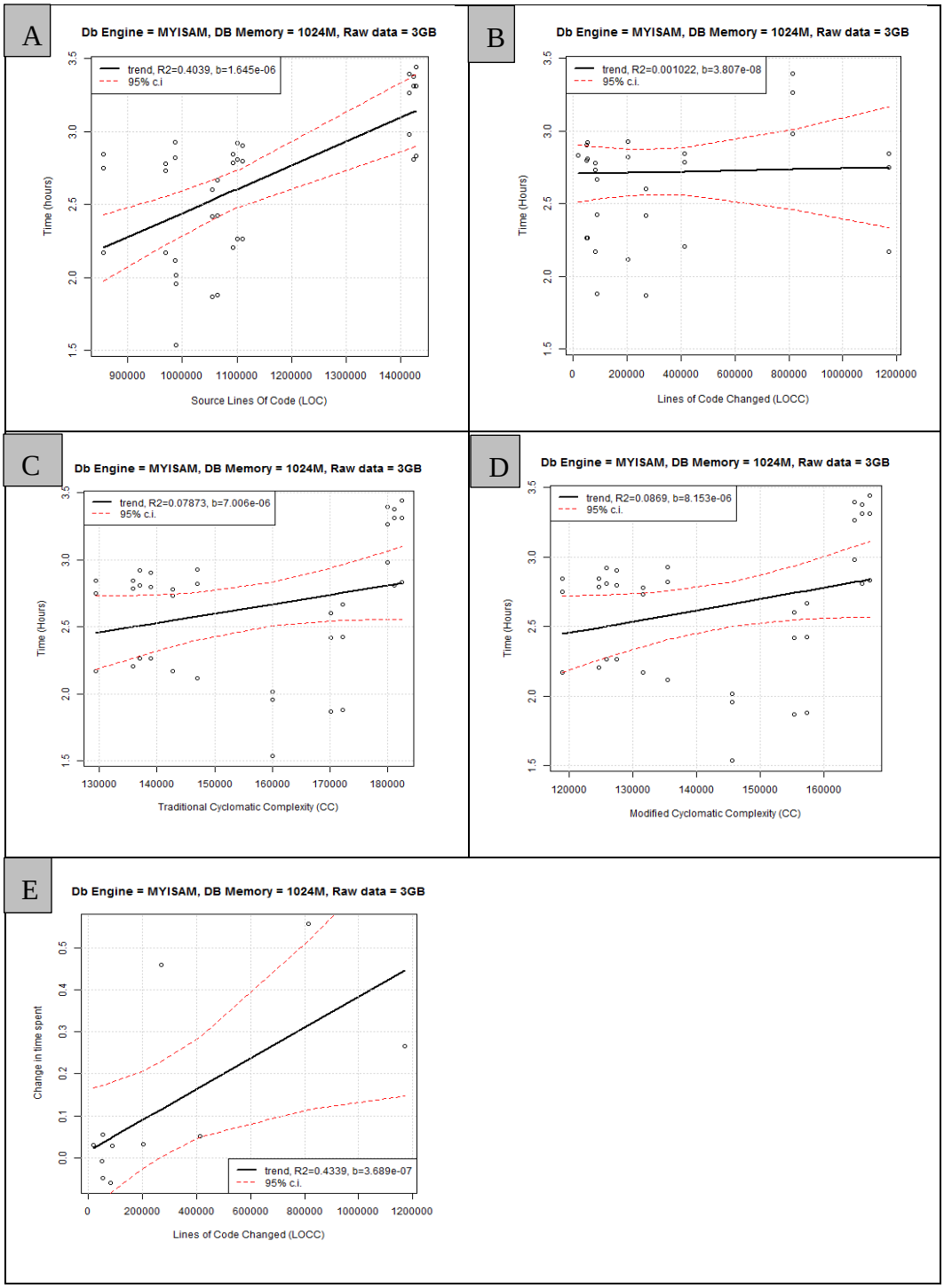


Figure B.14: Relation between time spent and various software metrics for Experiment 4. Subplot A is a scatterplot of LOC against time; subplot B is a scatterplot of LOCC against time; subplot C is a scatterplot of TCC against time; Subplot D is a scatterplot of MCC against time; Subplot E is a scatterplot of LOCC against change in time spent. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.4 InnoDB Experiments 6-8 Per-run Analysis (RQ2)

B.4.1 Experiment 6: Engine = InnoDB, Key buffer size =256M, Database size= 3GB

The following scatter graphs (Figures B.15 and B.16) represent the relation between the energy consumption (time spent) and the different software metrics we used. By examining the relation between the size of code (represented by the LOC metric for all the versions under study) and the energy consumed, we found a positive linear relation between the variables: as can be seen from Figure B.15-A, R2=0.61, and slope coefficient *b* is positive. This is further confirmed by a strong (ρ=0.78) correlation between the variables, as shown in Table 4.9.

As can be seen from Figures B.15-C and B.15-D, and Table 4.9, there exists a moderate positive linear relation between TCC (MCC) and consumed energy: R2=0.25 (R2=0.27) with a positive slope, and the correlation coefficient is equal to ρ= 0.50 (ρ=0.52). As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results.

We found no correlation between LOCC and consumed energy. As can be seen from Figure B.15-B and Table 4.9, R2 has extremely small value with negative slop and ρ=-0.21.

In addition to the low relation between LOCC and consumed energy, there exists a weak positive linear relation between LOCC and the change in energy consumption. As can be seen from Figure B.15-E and Table 4.10, the variability of the data is high (R2=0.02), and the correlation is weak (ρ=0.16).

Experiment 6's data, related to time spent (shown in Table 4.11), are identical to the energy consumption data in Table 4.9 as a result of the almost perfect correlation between energy and time (discussed in Section 4.2). Therefore, the LOC metric has a strong correlation with time spent, MCC and TCC have moderate correlation with time spent and no correlation is found between the LOCC metric and time spent. Table 4.12 shows a weak correlation (ρ=0.16) between changes in time with the LOCC metric.

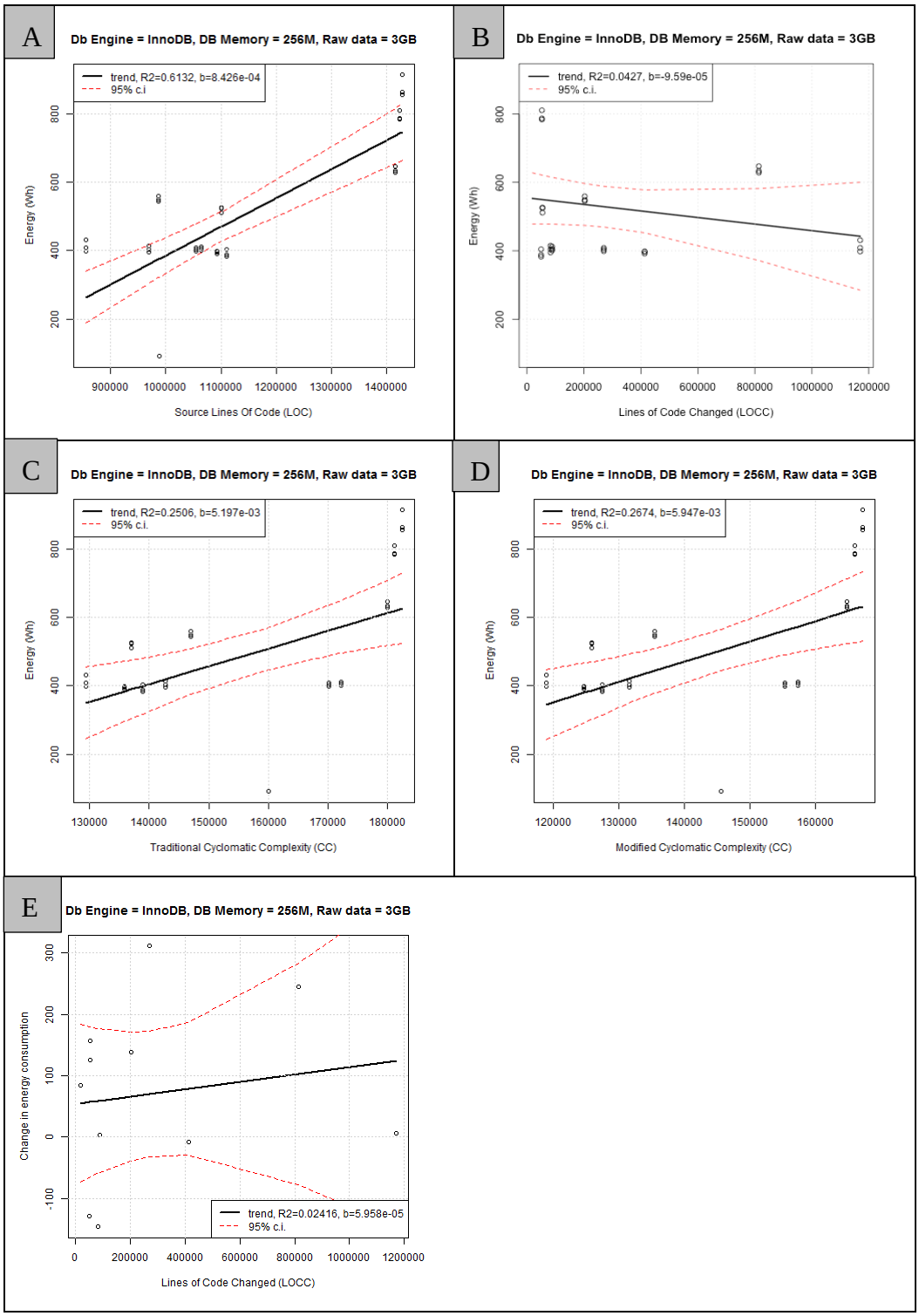


Figure B.15: Relation between energy consumed and various software metrics for Experiment 6. Subplot A is a scatterplot of LOC against energy; subplot B is a scatterplot of LOCC against energy; subplot C is a scatterplot of TCC against energy; Subplot D is a scatterplot of MCC against energy; Subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

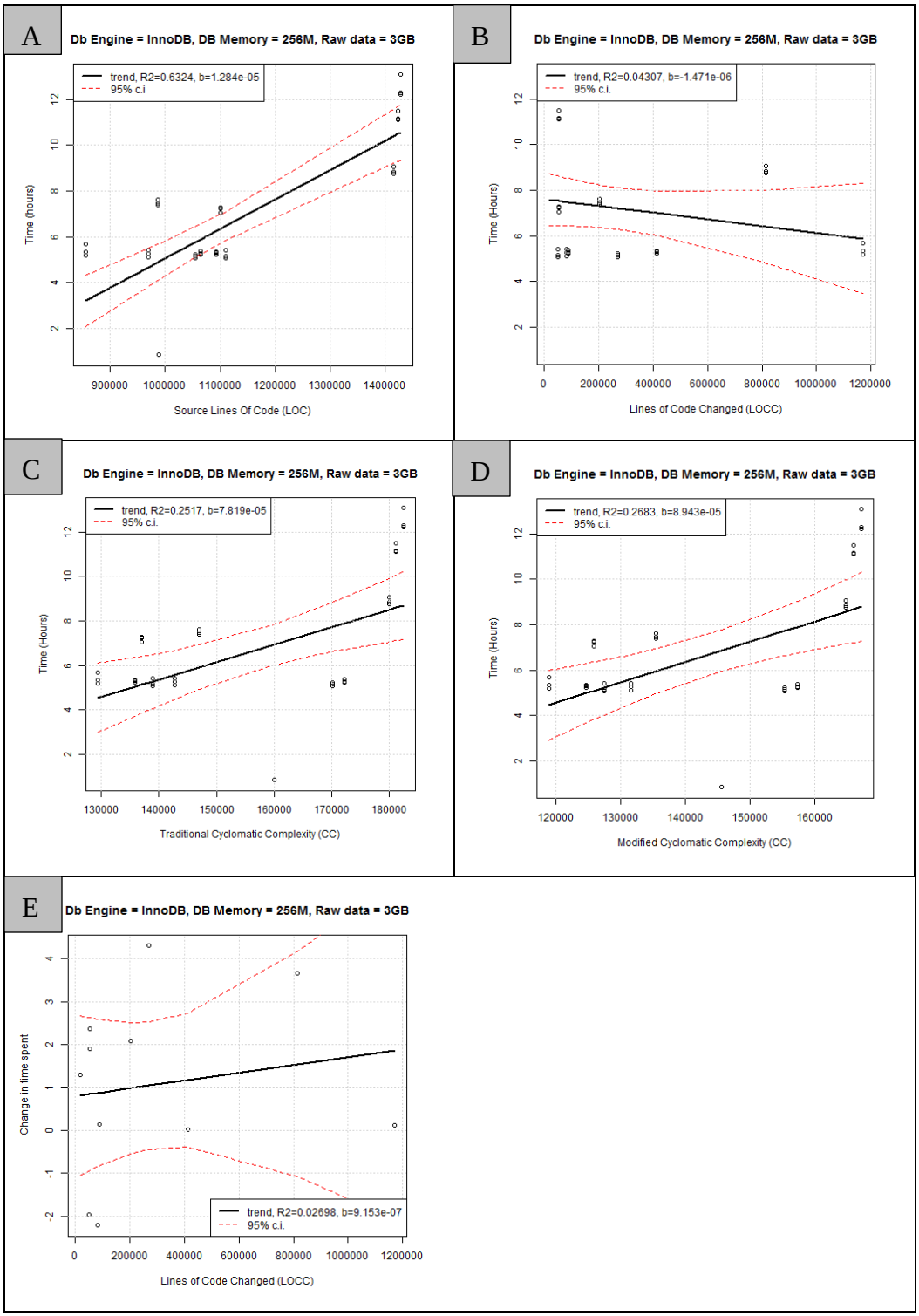


Figure B.16: Relation between time spent and various software metrics for Experiment 6. Subplot A is a scatterplot of LOC against time; subplot B is a scatterplot of LOCC against time; subplot C is a scatterplot of TCC against time; Subplot D is a scatterplot of MCC against time; Subplot E is a scatterplot of LOCC against change in time spent. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.4.2 Experiment 7: Engine = InnoDB, Key buffer size =1024M, Database size= 1GB

The following scatter graphs (Figures B.17 and B.18) represent the relation between the energy consumption (time spent) and the different software metrics we used. As discussed in Section B.2.2, we remove data for the outlier release (v.5.0.15) from our analysis. By examining the relation between the size of code (represented by the LOC metric for all the versions under study) and the energy consumed, we found a strong positive linear relation between the variables: as can be seen from Figure B.17-A, R2 has a very small value, and slope coefficient *b* is positive. This is further confirmed by a strong (ρ=0.88) correlation between the variables, as shown in Table 4.9.

As can be seen from Figures B.17-C and B.17-D, and Table 4.9, there exists a strong positive linear relation between TCC (MCC) and consumed energy: ρ=0.76 (ρ=0.75), even though variability of the data is high, R2=0.07 (R2= 0.06). As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results.

We found no correlation between LOCC and consumed energy. As can be seen from Figure B.17-B and Table 4.9, R2 has a very small value and ρ= 0.01.

Even though the relation between LOCC and consumed energy is low, there exists a moderate positive linear relation between LOCC and the change in energy consumption. As can be seen from Figure B.17-E and Table 4.10, the variability of the data is high (R2=0.01), but the correlation is moderate (ρ=0.49).

Experiment 7's data, related to time spent (shown in Table 4.11), are identical to the energy consumption data in Table 4.9 as a result of the almost perfect correlation between energy and time (discussed in Section 4.2). Therefore, the LOC metric has a strong correlation with time spent, MCC and TCC have strong correlation with time spent and no correlation is found between the LOCC metric and time spent. Table 4.12 shows a moderate correlation (ρ=0.52) between changes in time with the LOCC metric.

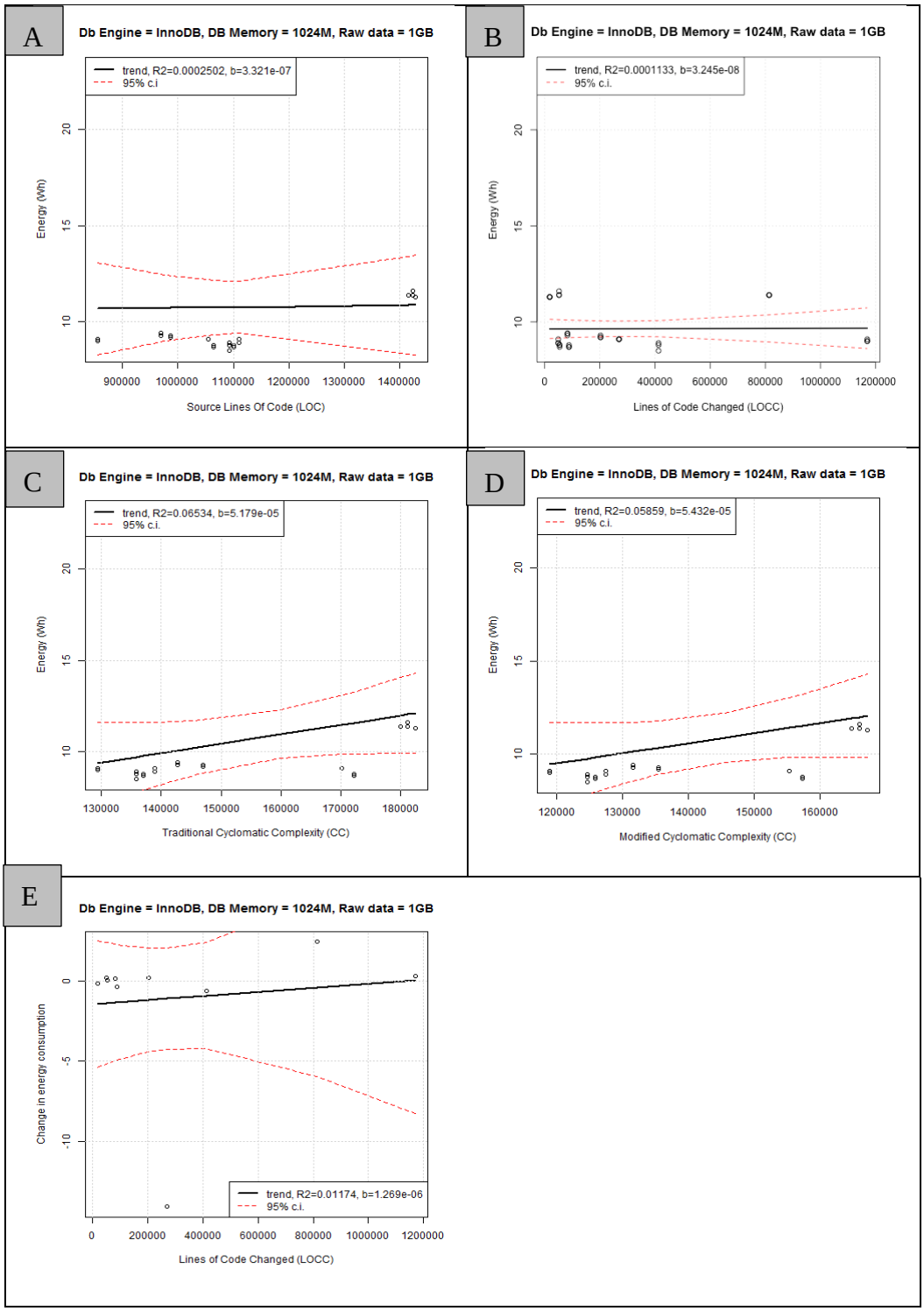


Figure B.17: Relation between Energy consumed and various software metrics for Experiment 7. Subplot A is a plot of LOC against energy; subplot B is a plot of LOCC against energy; subplot C is a plot of TCC against energy; Subplot D is a plot of MCC against energy; Subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

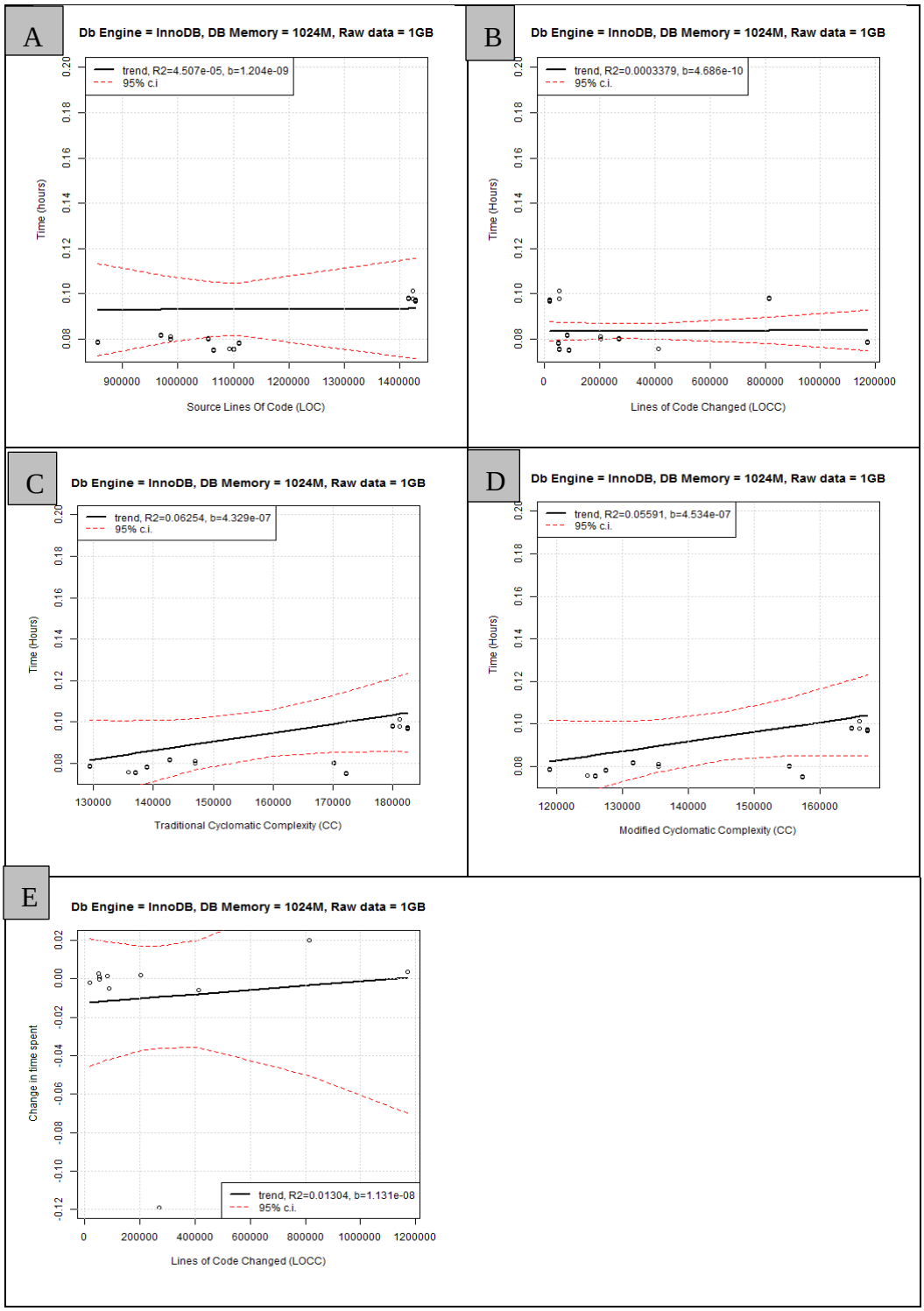


Figure B.18: Relation between time spent and various software metrics for Experiment 7. Subplot A is a plot of LOC against time; subplot B is a plot of LOCC against time; subplot C is a plot of TCC against time; Subplot D is a plot of MCC against time; Subplot E is a scatterplot of LOCC against change in time spent. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.4.3 Experiment 8: Engine = InnoDB, Key buffer size =1024M, Database size= 3GB

The following scatter graphs (Figures B.19 and B.20) represent the relation between the energy consumption (time spent) and the different software metrics we used. By examining the relation between the size of code (represented by the LOC metric for all the versions under study) and the energy consumed, based on Table 4.9, we found a moderate positive linear relation between the variables: ρ=0.48, even though the model variability is high (as can be seen from Figure B.19-A, R2=0.01).

As can be seen from Figures B.19-C and B.19-D, and Table 4.9, there exists a weak (borderline to none) positive linear relation between TCC (MCC) and consumed energy: R2=0.03 in both cases with a positive slope, and the correlation coefficient is equal to ρ= 0.11 in both cases. As mentioned in the previous section, MCC and TCC are almost perfectly correlated, hence the similarity in results.

We found weak correlation between LOCC and consumed energy. As can be seen from Figure B.19-B and Table 4.9, R2=0.03 and ρ= 0.16. In addition to the low relation between LOCC and consumed energy, there exists a weak positive linear relation between LOCC and the change in energy consumption. As can be seen from Figure B.19-E and Table 4.10, the variability of the data is high (R2= 0.03), and the correlation is weak (ρ=0.18).

Experiment 8's data, related to time spent (shown in Table 4.11), are identical to the energy consumption data in Table 4.9 as a result of almost perfect correlation between energy and time (discussed in Section 4.2). Therefore, the LOC metric has a moderate correlation with time spent, MCC and TCC have weak correlation with time spent, and weak correlation is found between the LOCC metric and time spent. Table 4.12 shows a weak correlation (ρ=0.17) between changes in time with the LOCC metric.

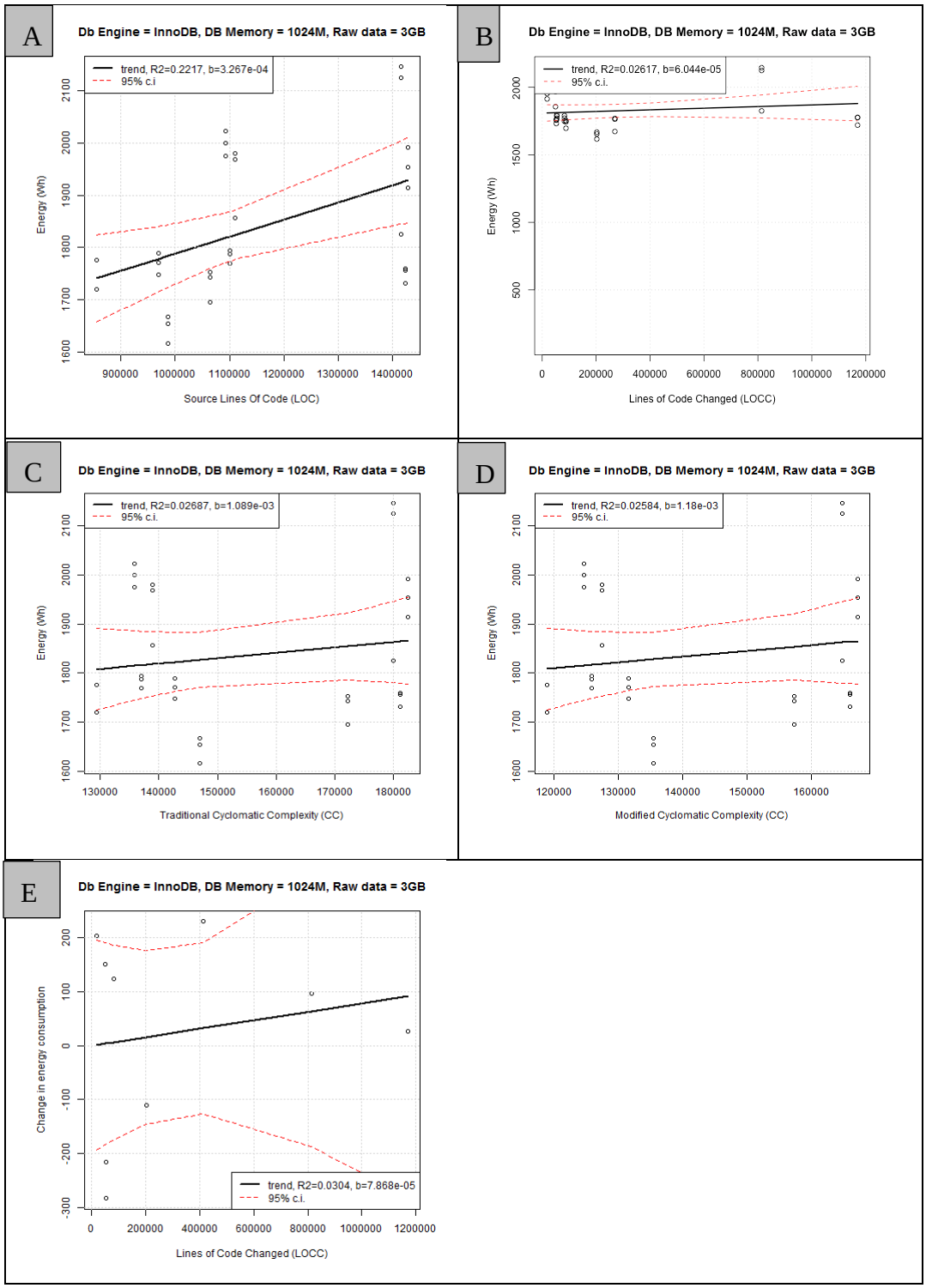


Figure B.19: Relation between energy consumed and various software metrics for Experiment 8. Subplot A is a scatterplot of LOC against energy; subplot B is a scatterplot of LOCC against energy; subplot C is a scatterplot of TCC against energy; Subplot D is a scatterplot of MCC against energy; Subplot E is a scatterplot of LOCC against change in energy consumption. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

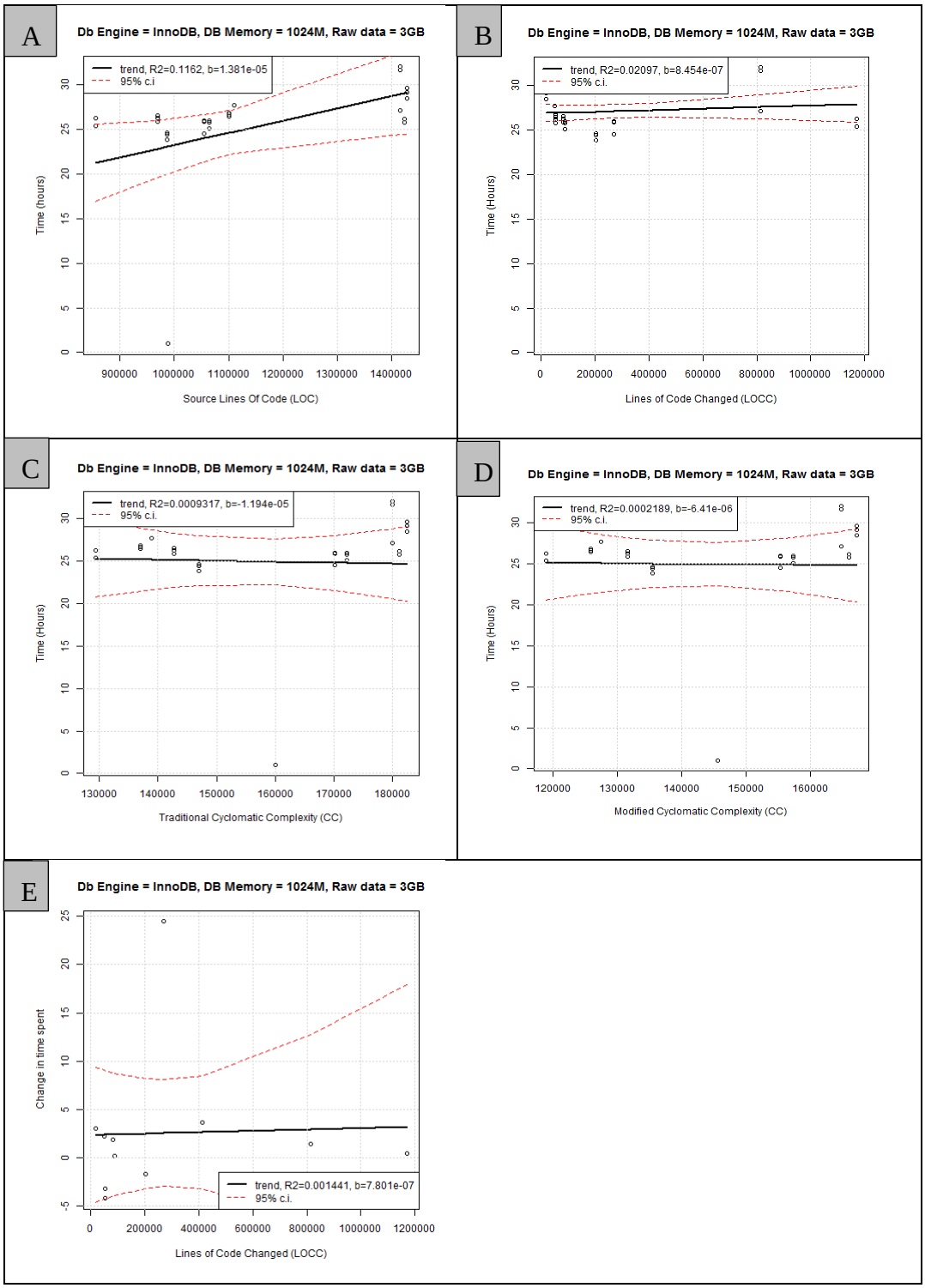


Figure B.20: Relation between time spent and various software metrics for Experiment 8. Subplot A is a scatterplot of LOC against time; subplot B is a scatterplot of LOCC against time; subplot C is a scatterplot of TCC against time; Subplot D is a scatterplot of MCC against time; Subplot E is a scatterplot of LOCC against change in time spent. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.5 Per-statement Analysis (RQ1)

In addition to the overall analysis of the workload (discussed in Chapter 5), we investigated behavior of each of the 22 queries in the workload individually. For the sake of brevity, we discuss here only energy consumption. Given that the time spent and energy consumed are correlated, as per Section 4.2, our energy-related findings will apply to the execution time as well.

We also know that MCC and TCC are perfectly correlated (see Table 4.8); therefore, we used MCC in this section as an example of the complexity metric, and our finding for MCC will apply to TCC as well. The following sections show the relation between the energy consumption and the improvement in the MySQL versions from release to release, for each single query per each experiment. We focused on the analysis of three queries consuming the least amount of energy, denoted by Bottom-3, and three queries consuming the largest amount of energy, denoted by Top-3.

The Bottom-3 and Top-3 queries for both database engines under study (MyISAM and InnoDB) are shown in Tables B.1, B.3, B.5, and B.7. As we can see in Table B.1, MyISAM Bottom-3 set for all experiments and releases contains six distinct queries (Queries # 11, 16, 17, 19, 20, and 22) and MyISAM Top-3 set – five distinct queries (Queries # 5, 9, 14, 18, and 21) as shown in Table B.3; InnoDB Bottom-3 set contains ten distinct queries (Queries # 2, 4, 6, 11, 16, 7, 19, 18, 20, 22,) as shown in Table B.5; and InnoDB Top-3 set – eight distinct queries (Queries # 1, 3, 5, 9, 10, 14, 18, 21) as shown in Table B.7. These findings suggest that the sets of most problematic and least problematic queries (from the energy consumption perspective) are fairly robust and are independent of the database engine's type and setup.

Energy consumption of queries, frequently appearing in Bottom-3 and Top-3 sets, are given in Tables B.2, B.4, B.6, and B.8; and Figures B.21 and B.22. Some statements become greener as releases mature, and some not. However, greener values are observed more frequently on older releases of the engines.

Table B.1: MyISAM; Bottom-3 energy consuming queries per each release and experiment.

MyISAM: Bottom-3 energy Consumed queries per Release													
Experiment		**5.0.15**	**5.0.67**	**5.0.96**	**5.1.30**	**5.1.50**	**5.1.72**	**5.5.10**	**5.5.20**	**5.5.39**	**5.6.10**	**5.6.15**	**5.6.21**
Experiment 1	Bottom-1	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11
	Bottom-2	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 19	Query 19	Query 19
	Bottom-3	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 22	Query 22	Query 22
Experiment 2	Bottom-1	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 22
	Bottom-2	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 19
	Bottom-3	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17
Experiment 3	Bottom-1	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11
	Bottom-2	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19
	Bottom-3	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 20	Query 20	Query 20	Query 22	Query 22	Query 22
Experiment 4	Bottom-1	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22
	Bottom-2	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17
	Bottom-3	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 16	Query 16	Query 16

Table B.2: MyISAM; Energy consumption of frequently occurring Bottom-3 queries (in Wh) for each experiment and release; Green color represents minimum value; Red color represents maximum value.

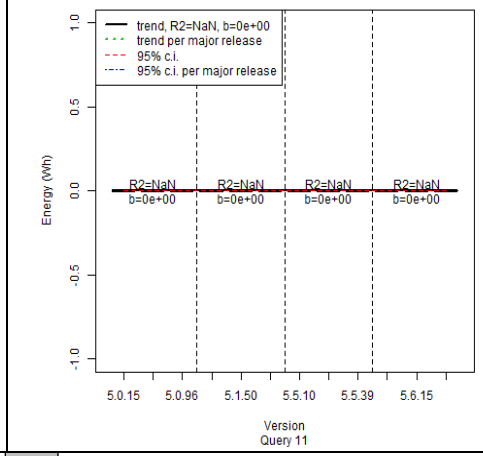
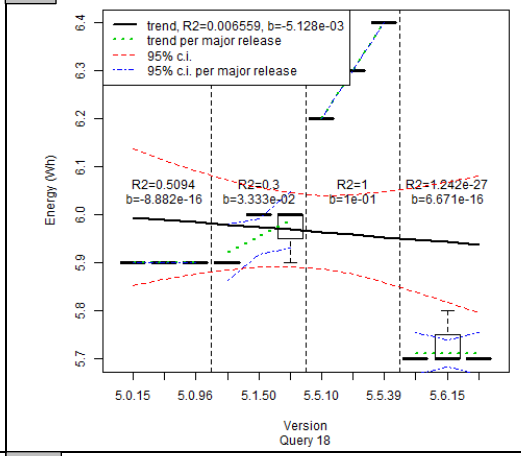
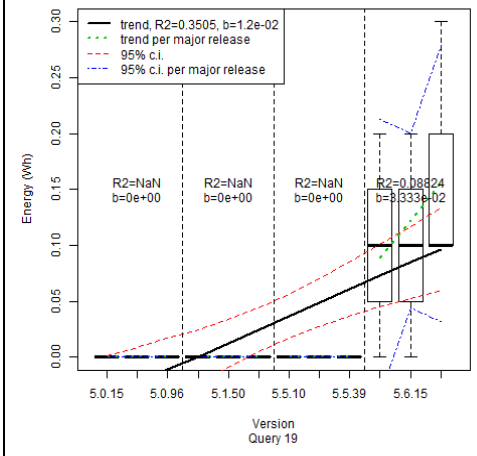
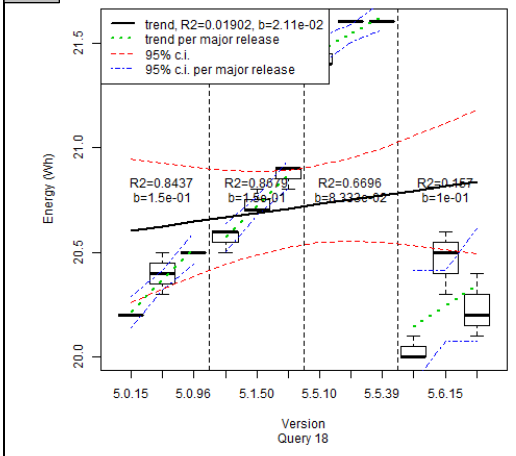
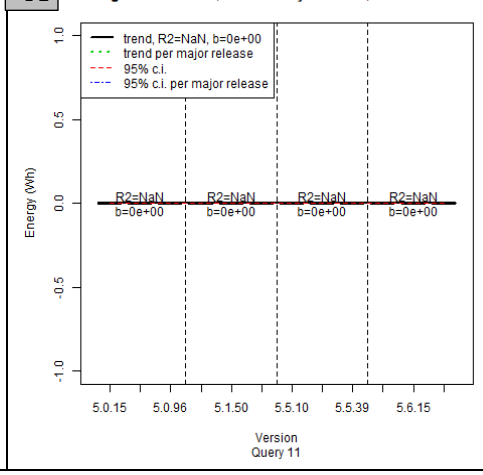
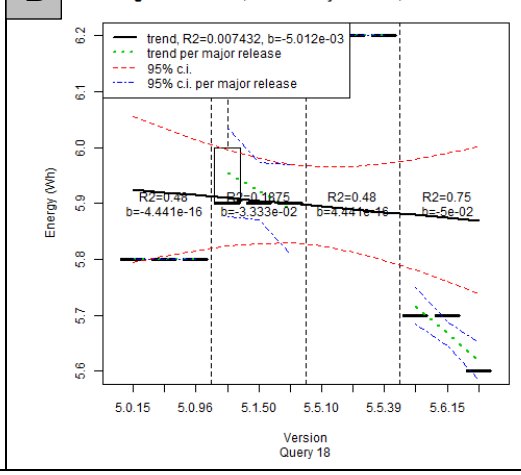
MyISAM: Bottom-3 energy consuming queries														
Experiment	**Query**	**MySQL releases**											**Min**	**Max**
		5.0.15	**5.0.67**	**5.0.96**	**5.1.30**	**5.1.50**	**5.1.72**	**5.5.10**	**5.5.20**	**5.5.39**	**5.6.10**	**5.6.15**	**5.6.21**	
Experiment 1	Query 11	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
	Query 19	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
	Query 22	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
Experiment 2	Query 19	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.10	0.10	0.17	< 0.1	0.17
	Query 22	0.03	0.07	0.03	0.03	0.10	0.03	0.10	0.10	0.07	0.10	0.10	0.10	0.03	0.10
	Query 17	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.20	0.17	0.20	0.20	0.20	0.10	0.20
Experiment 3	Query 11	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
	Query 17	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.03	0.03	0.00	0.07	0.03	0.10	< 0.1	0.10
	Query 19	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
Experiment 4	Query 22	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.10	< 0.1	0.03	0.07	0.03	0.03	< 0.1	0.10
	Query 17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.27	0.20	0.30	0.27	0.30	0.17	0.30
	Query 19	0.40	0.40	0.30	0.50	0.37	0.43	0.50	0.50	0.50	1.50	1.47	1.50	0.30	1.50

Table B.3: MyISAM; Top-3 energy consuming queries per each release and experiment.

MyISAM: Top-3 energy Consumed queries per Release													
Experiment		**5.0.15**	**5.0.67**	**5.0.96**	**5.1.30**	**5.1.50**	**5.1.72**	**5.5.10**	**5.5.20**	**5.5.39**	**5.6.10**	**5.6.15**	**5.6.21**
Experiment 1	Top-1	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 21	Query 5	Query 5
	Top-2	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 5	Query 21	Query 21
	Top-3	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 18	Query 18	Query 18
Experiment 2	Top-1	Query 18	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9
	Top-2	Query 21	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 21	Query 21	Query 14
	Top-3	Query 14	Query 21	Query 21	Query 14	Query 21	Query 21	Query 21	Query 14	Query 14	Query 5	Query 5	Query 21
Experiment 3	Top-1	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 21	Query 5	Query 21
	Top-2	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 21	Query 5	Query 21	Query 5
	Top-3	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 5	Query 18	Query 18	Query 18
Experiment 4	Top-1	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9
	Top-2	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14
	Top-3	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 18	Query 20	Query 20	Query 20

Table B.4: MyISAM; Energy consumption of frequently occurring Top-3 queries (in Wh) for each experiment and release; Green color represents minimum value; Red color represents maximum value.

MyISAM: Top-3 energy consuming queries														
Experiment	**Query**	**MySQL releases**											**Min**	**Max**
		5.0.15	**5.0.67**	**5.0.96**	**5.1.30**	**5.1.50**	**5.1.72**	**5.5.10**	**5.5.20**	**5.5.39**	**5.6.10**	**5.6.15**	**5.6.21**	
Experiment 1	Query 18	5.90	5.90	5.90	5.90	6.00	5.97	6.20	6.30	6.40	5.70	5.73	5.70	5.70	6.40
	Query 21	5.30	5.50	5.53	5.57	5.63	5.60	6.10	6.10	6.00	7.13	6.80	6.60	5.30	7.13
	Query 5	3.63	5.33	5.33	5.27	5.30	5.27	5.43	5.70	5.50	6.83	6.87	6.73	3.63	6.87
Experiment 2	Query 18	20.20	20.40	20.50	20.57	20.73	20.87	21.43	21.60	21.60	20.03	20.47	20.23	20.03	21.60
	Query 21	16.63	17.43	17.50	17.57	17.53	17.73	19.07	19.00	18.83	21.23	21.43	21.33	16.63	21.43
	Query 9	13.63	24.97	26.17	43.10	41.67	41.70	48.93	44.87	43.63	46.07	44.63	46.53	13.63	48.93
Experiment 3	Query 18	5.80	5.80	5.80	5.97	5.90	5.90	6.20	6.20	6.20	5.70	5.70	5.60	5.60	6.20
	Query 21	5.20	5.50	5.40	5.60	5.60	5.60	5.90	6.00	6.00	6.77	6.77	6.60	5.20	6.77
	Query 5	3.50	5.20	5.13	5.13	5.23	5.20	5.53	5.50	5.40	6.63	6.80	6.50	3.50	6.80
Experiment 4	Query 9	35.73	46.47	47.07	65.63	68.57	64.50	65.30	66.20	66.37	69.13	66.20	66.30	35.73	69.13
	Query 14	27.53	41.03	42.70	44.53	42.50	42.23	43.10	44.77	44.53	46.67	46.70	47.90	27.53	47.90
	Query 18	20.27	20.50	20.60	20.47	20.67	20.70	21.50	21.57	21.53	20.13	20.73	20.63	20.13	21.57

Experiment	Greener/Faster Query	Least green /Slower Query
Experiment 1	A 	B 
Experiment 2	A 	B 
Experiment 3	A 	B 

Continued

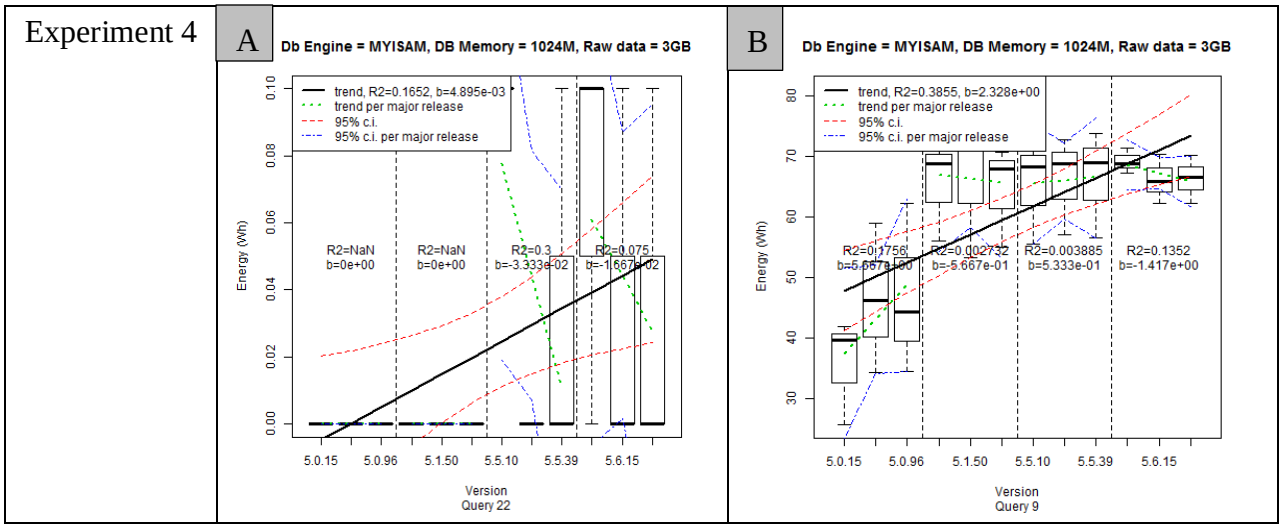


Figure B.21: MyISAM experiments; subplot A is box plot of energy consumed against MySQL's versions for greener query; subplot B is box plot of energy consumed against MySQL's versions for least green query per each experiment. On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

Table B.5: InnoDB: Bottom-3 energy consuming queries per each release and experiment.

InnoDB: Bottom-3 energy consumed queries per release													
Experiment		**5.0.15**	**5.0.67**	**5.0.96**	**5.1.30**	**5.1.50**	**5.1.72**	**5.5.10**	**5.5.20**	**5.5.39**	**5.6.10**	**5.6.15**	**5.6.21**
Experiment 5	Bottom-1	Query 11	Query 11	Query 11	Query 11	Query 16	Query 11	Query 11	Query 11	Query 11	Query 11	Query 19	Query 19
	Bottom-2	Query 17	Query 17	Query 17	Query 17	Query 11	Query 17	Query 17	Query 17	Query 17	Query 17	Query 22	Query 22
	Bottom-3	Query 19	Query 19	Query 19	Query 19	Query 17	Query 19	Query 19	Query 19	Query 19	Query 19	Query 17	Query 17
Experiment 6	Bottom-1	Query 19	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22
	Bottom-2	Query 22	Query 20	Query 11	Query 20	Query 20	Query 20	Query 4	Query 17	Query 4	Query 17	Query 17	Query 17
	Bottom-3	Query 17	Query 17	Query 20	Query 11	Query 11	Query 11	Query 17	Query 4	Query 17	Query 4	Query 4	Query 4
Experiment 7	Bottom-1	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11
	Bottom-2	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17	Query 17
	Bottom-3	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19	Query 19
Experiment 8	Bottom-1	Query 19	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22	Query 22
	Bottom-2	Query 22	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11	Query 11
	Bottom-3	Query 17	Query 2	Query 2	Query 2	Query 2	Query 2	Query 6	Query 4	Query 18	Query 6	Query 6	Query 6

Table B.6: InnoDB; Energy consumption of frequently occurring Bottom-3 queries (in Wh) for each experiment and release; Green color represents minimum value; Red color represents maximum value.

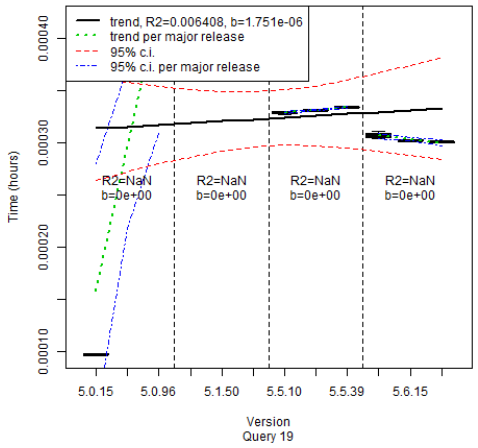
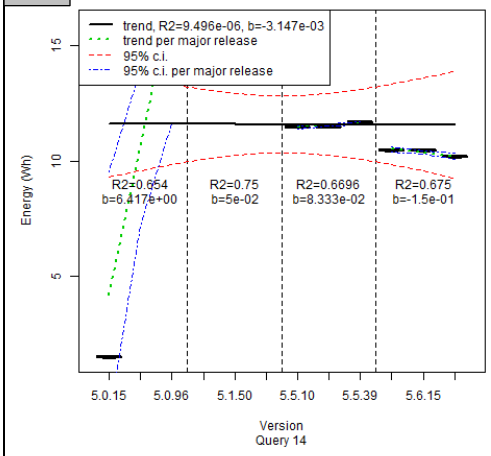
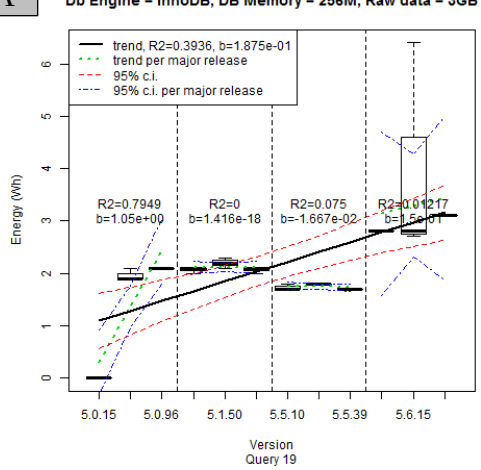
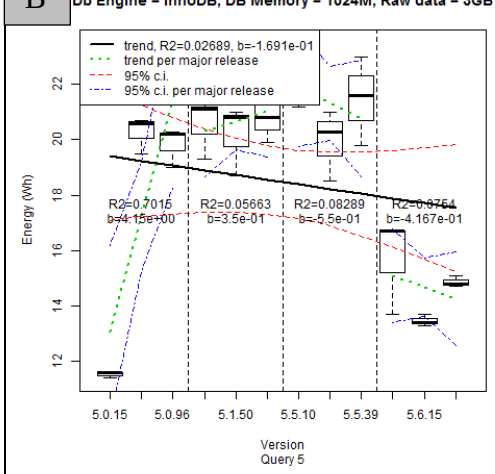
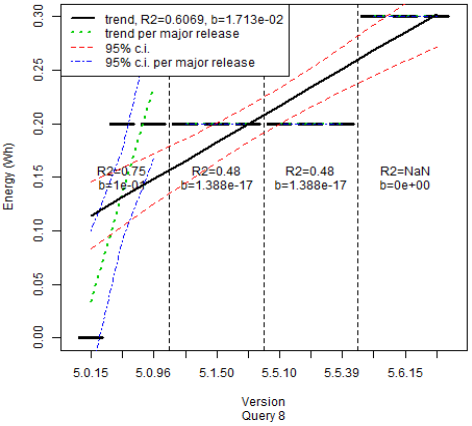
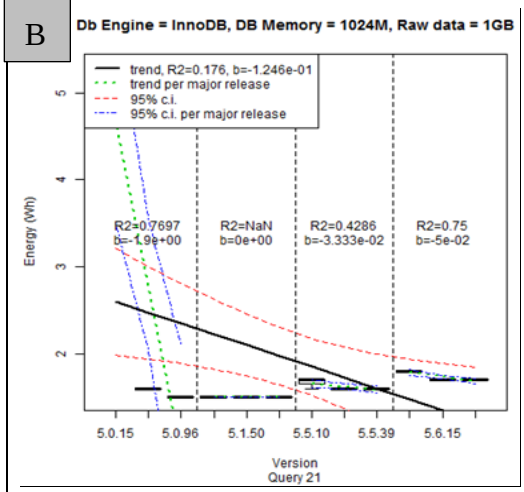
InnoDB: Bottom-3 energy consuming queries															
Experiment	Query	MySQL releases												Min	Max
		5.0.15	5.0.67	5.0.96	5.1.30	5.1.50	5.1.72	5.5.10	5.5.20	5.5.39	5.6.10	5.6.15	5.6.21		
Experiment 5	Query 19	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
	Query 22	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
	Query 17	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.03	< 0.1	< 0.1	0.03
Experiment 6	Query 19	< 0.1	1.97	2.10	2.07	2.20	2.07	1.73	1.80	1.70	2.80	3.97	3.10	< 0.1	3.97
	Query 22	0.03	0.40	0.40	0.40	0.40	0.40	0.53	0.50	0.53	0.50	0.50	0.60	0.03	0.60
	Query 17	0.13	1.00	1.00	1.03	1.10	1.10	1.03	1.10	1.07	1.10	1.10	1.20	0.13	1.20
Experiment 7	Query 8	< 0.1	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.30	0.30	0.30	< 0.1	0.30
	Query 20	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.20	0.20	0.20	< 0.1	0.20
	Query 11	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	<0.1
Experiment 8	Query 19	< 0.1	3.63	3.60	3.63	3.40	3.63	4.00	3.63	3.90	4.87	4.17	4.53	< 0.1	4.87
	Query 22	0.07	0.20	0.20	0.20	0.20	0.20	0.60	0.43	0.57	0.60	0.50	0.60	0.07	0.60
	Query 17	0.10	1.93	1.90	1.97	1.80	1.97	2.10	1.90	2.10	2.13	1.83	1.97	0.10	2.13

Table B.7: InnoDB; Top-3 energy consuming queries per each release and experiment.

InnoDB: Top-3 energy consumed queries per release													
Experiment		5.0.15	5.0.67	5.0.96	5.1.30	5.1.50	5.1.72	5.5.10	5.5.20	5.5.39	5.6.10	5.6.15	5.6.21
Experiment 5	Top-1	Query 18	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14
	Top-2	Query 21	Query 9	Query9	Query 9	Query 9	Query 9	Query 9	Query 1	Query 9	Query 18	Query 18	Query 18
	Top-3	Query 5	Query 10	Query 10	Query 10	Query 1	Query 1	Query 1	Query 21	Query 1	Query 9	Query 9	Query 9
Experiment 6	Top-1	Query 18	Query 9	Query 9	Query 9	Query 14	Query 9	Query 9	Query 14	Query 9	Query 18	Query 18	Query 18
	Top-2	Query 21	Query 14	Query 14	Query 14	Query 9	Query 14	Query 14	Query 9	Query 14	Query 9	Query 14	Query 14
	Top-3	Query 5	Query 3	Query 3	Query 3	Query 3	Query 3	Query 3	Query 3	Query 3	Query 14	Query 9	Query 9
Experiment 7	Top-1	Query 18	Query 21	Query 21	Query 14	Query 14	Query 14	Query 1	Query 1	Query 14	Query 18	Query 18	Query 18
	Top-2	Query 21	Query 1	Query 1	Query 21	Query 1	Query 1	Query 21	Query 21	Query 1	Query 21	Query 1	Query 1
	Top-3	Query 5	Query 14	Query 14	Query 1	Query 21	Query 21	Query 14	Query 14	Query 21	Query 1	Query 21	Query 21
Experiment 8	Top-1	Query 18	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14	Query 14
	Top-2	Query 21	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 9	Query 18	Query 18	Query 18
	Top-3	Query 5	Query 3	Query 3	Query 3	Query 3	Query 3	Query 3	Query 3	Query 3	Query 9	Query 9	Query 9

Table B.8: InnoDB; Energy consumption of frequently occurring Top-3 queries (in Wh) for each experiment and release; Green color represents minimum value; Red color represents maximum value.

InnoDB: Top-3 energy consuming queries															
Experiment	Query	MySQL releases												Min	max
		5.0.15	5.0.67	5.0.96	5.1.30	5.1.50	5.1.72	5.5.10	5.5.20	5.5.39	5.6.10	5.6.15	5.6.21		
Experiment 5	Query 14	1.47	15.97	14.30	13.80	13.90	13.90	11.50	11.50	11.67	10.47	10.47	10.17	1.47	15.97
	Query 21	5.30	1.57	1.43	1.50	1.50	1.50	1.50	1.50	1.50	1.87	1.80	1.80	1.43	5.30
	Query 1	1.30	1.50	1.50	1.50	1.57	1.60	1.70	1.60	1.60	1.70	1.77	1.70	1.30	1.77
Experiment 6	Query 5	11.53	10.80	11.13	10.73	10.83	10.83	10.57	10.57	10.63	10.57	10.70	12.33	10.57	12.33
	Query 9	6.63	125.33	133.10	129.27	124.63	131.33	130.90	121.37	130.97	128.87	119.97	137.80	6.63	137.80
	Query 7	5.10	12.77	13.70	13.17	15.53	13.73	14.00	15.60	13.87	13.30	15.00	17.87	5.10	17.87
Experiment 7	Query 21	5.30	1.60	1.50	1.50	1.50	1.50	1.67	1.60	1.60	1.80	1.70	1.70	1.50	5.30
	Query 14	1.47	1.40	1.40	1.80	1.90	1.87	1.50	1.50	1.70	1.60	1.63	1.60	1.40	1.90
	Query 1	1.30	1.53	1.43	1.43	1.53	1.63	1.67	1.63	1.60	1.70	1.80	1.70	1.30	1.80
Experiment 8	Query 5	11.53	20.27	19.83	20.57	20.17	21.27	22.57	19.93	21.47	15.70	13.47	14.87	11.53	22.57
	Query 9	8.90	204.87	204.90	206.77	200.27	210.73	223.07	212.73	223.63	178.80	148.77	173.67	8.90	223.63
	Query 14	6.33	1303.33	1303.33	1322.90	1230.63	1325.93	1541.43	1352.53	1476.93	1127.20	961.43	1081.50	6.33	1541.43

Experiment	Greener/Faster Query	Least green /Slower Query
Experiment 5	A 	B 
Experiment 6	A 	B 
Experiment 7	A 	B 

Continued

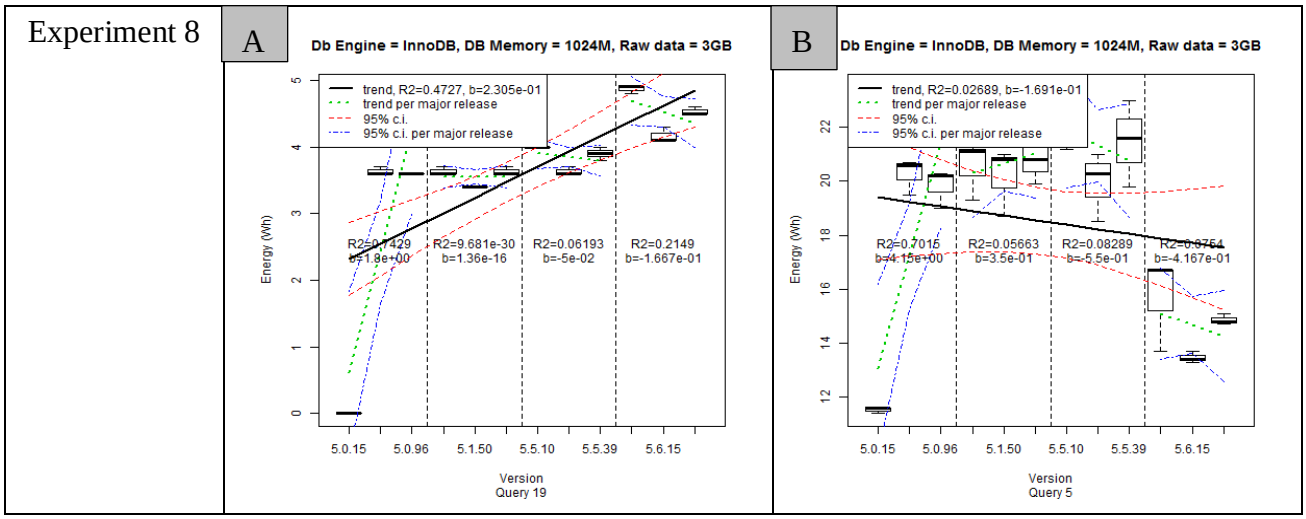


Figure B.22: InnoDB experiments; subplot A is box plot of energy consumed against MySQL's versions for greener query; subplot B is box plot of energy consumed against MySQL's versions for least green query per each experiment. On both subplots black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line; dotted green line represents trend per major release and dotted blue line show 95% confidence interval of trend line per major release.

B.5.1 Per statement analysis – observations

We provide additional observations related to RQ1 below.

- MyISAM experiments showed that the older releases are greener than the new one in most of the cases as we can see in Table B.2 and B.4. The minimum values for average energy consumption are associated with release v.5.0 (except Query 18 in Table B.4), while the maximum values of average energy consumption are associated with the mature releases v.5.5 and v.5.6 in all cases. These findings are similar to our "per run" results, given in Chapter 5.
- All InnoDB experiments (except Query 5 in Table B.8) showed that the greenest release is the oldest one (v.5.0); while the least green ones are, in most cases, the mature releases v.5.5 and v.5.6 (only queries 14 and 21 gave different results in Experiment 5 and 7, Table B.8) as we can see in Tables B.6 and B.8.
- As discussed in Section 4.2, the energy consumption and execution time are perfectly correlated. Therefore, for per statement analysis, we can say that the older releases are using less energy and time, hence they are greener and faster than the mature releases.

B.6 Per-statement Analysis (RQ2)

B.6.1 MyISAM Experiments

Tables B.9 to B.12, show the correlation between the energy consumption and software metrics for each individual query in all the four MyISAM experiments. The NA values in these tables appear when the correlation coefficient values cannot be computed. This happens due to limitation of the Pearson correlation formula: when all the energy consumption values are identical, the standard deviation of the energy consumption (used in the denominator Pearson correlation formula) is 0, leading to division by 0. This happened for fast queries with energy readings of less than 0.1 Wh, which our measuring device cannot capture; we exclude these queries from our analysis.

In the following sections we analyze these results per each experiment.

B.6.1.1 Experiment 1: Engine = MyISAM, Key Buffer Size = 256M, Database Size = 1GB

As we can see in Table B.9, energy consumption of all the queries, except queries #16 and 18, exhibit moderate to strong positive correlation with LOC (Pearson correlation coefficient, ρ, ranges between 0.54 and 0.93). Queries #16 and 18, on the other hand, exhibit moderate to strong negative correlation (ρ ranges between -0.93 and -0.45). These findings are confirmed by linear regression analysis (samples of the analysis are given in Figure B.23). Energy consumption is weakly correlated (or has no correlation) with LOCC (as per Table B.9); ρ ranges between -0.24 and 0.21.

Correlation between energy consumption and MCC/TCC ranges between weak and strong. Energy consumption has strong to moderated positive correlation for 14 queries out of 22 (Queries# 1, 3, 5-8, 10, 12 - 15, 17, 20, and 21) and ρ ranging between 0.42 and 0.76; strong negative correlation for Queries #16 and 18 with ρ ranging between -0.77 and -0.75 (similar to the LOC case); and none to weak correlation for Queries # 2 and 9 with ρ ranging between -0.03 and 0.20.

Table B.9: Shows the correlation between the energy consumption and software metrics per each query for MyISAM Experiment 1.

		MYISAM																						
Exp.	Metrics	TPC-H 22 queries																					max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
Exp.1	LOC	0.86	0.68	0.87	NA	0.83	0.79	0.87	0.57	0.54	0.86	NA	0.86	0.93	0.85	0.84	-0.93	0.72	-0.45	NA	0.93	0.89	NA	0.93	-0.93
	LOCC	-0.17	0.05	-0.18	NA	-0.05	0.10	0.01	-0.07	0.12	0.21	NA	0.21	0.00	-0.08	-0.05	0.00	0.18	-0.24	NA	0.00	0.03	NA	0.21	-0.24
	MCC	0.43	0.20	0.53	NA	0.50	0.64	0.44	0.47	-0.01	0.69	NA	0.69	0.76	0.47	0.49	-0.76	0.58	-0.77	NA	0.76	0.48	NA	0.76	-0.77
	TCC	0.42	0.18	0.52	NA	0.48	0.64	0.43	0.46	-0.03	0.68	NA	0.68	0.75	0.45	0.48	-0.75	0.57	-0.77	NA	0.75	0.47	NA	0.75	-0.77

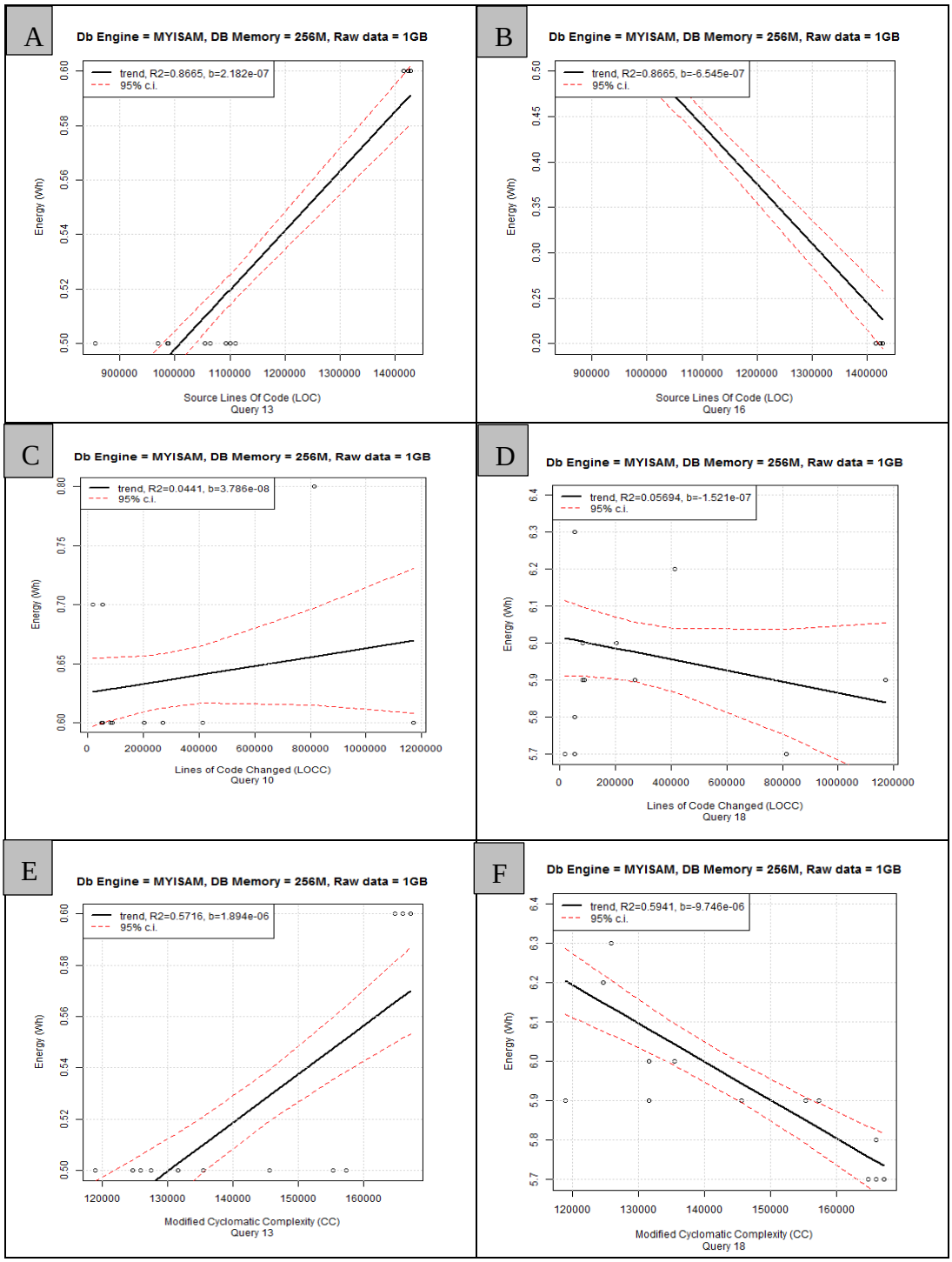


Figure B.23: MyISAM Experiment1; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.6.1.2 Experiment 2: Engine = MyISAM, Key buffer size = 256M, Database size = 3GB

As shown in Table B.10, energy consumption for 16 queries (Queries #1-7, 10, 12, 14, 15, 17, and 19-22), exhibit moderate to strong positive correlation with LOC: ρ ranges between 0.44 and 0.93. The rest of the queries (Queries # 8, 9, 11, 13, and 18), on the other hand, exhibit weak correlation with ρ ranges between -0.30 and 0.36. Query #16 showed strong negative correlation with ρ = - 0.88 .This findings are confirmed by linear regression analysis (samples of the analysis are given in Figure B.24). Energy consumption is weakly correlated (or has no correlation) with LOCC (as per Table B.10); ρ ranges between and -0.30 and 0.14.

Correlation between energy consumption and MCC/TCC ranges between weak to strong. Energy consumption has strong to moderate positive correlation for 7 queries (Queries # 4, 5, 12, 15, 19, 20 and 21) with ρ ranging between 0.41 and 0.74; strong negative correlation for Queries # 16 and 18 with ρ ranging between -0.78 and -0.74 (similar to Experiment 1 findings), and none to weak correlation for queries (Queries # 1, 2, 3, 6 - 11, 13, 14, 17, and 22) with ρ ranging between -0.29 and 0.39.

Table B.10: Shows the correlation between the energy consumption and software metrics per each query for MyISAM Experiments 2.

		MYISAM																						
Exp.	Metrics	TPC-H 22 queries																					max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
Exp.2	LOC	0.69	0.66	0.72	0.60	0.78	0.50	0.75	-0.21	0.36	0.52	-0.20	0.81	0.31	0.52	0.83	-0.88	0.80	-0.30	0.71	0.93	0.93	0.44	0.93	-0.88
	LOCC	-0.29	-0.20	-0.19	-0.11	-0.08	-0.11	-0.12	-0.03	0.14	-0.02	-0.21	-0.23	-0.07	0.06	-0.25	-0.07	-0.20	-0.30	-0.06	-0.02	-0.09	-0.13	0.14	-0.30
	MCC	0.19	0.11	0.38	0.41	0.47	0.35	0.34	-0.29	-0.20	0.14	0.29	0.70	-0.06	0.05	0.46	-0.78	0.39	-0.74	0.58	0.74	0.52	0.16	0.74	-0.78
	TCC	0.18	0.09	0.37	0.41	0.45	0.34	0.33	-0.30	-0.22	0.12	0.31	0.70	-0.07	0.04	0.45	-0.78	0.39	-0.74	0.57	0.73	0.51	0.15	0.73	-0.78

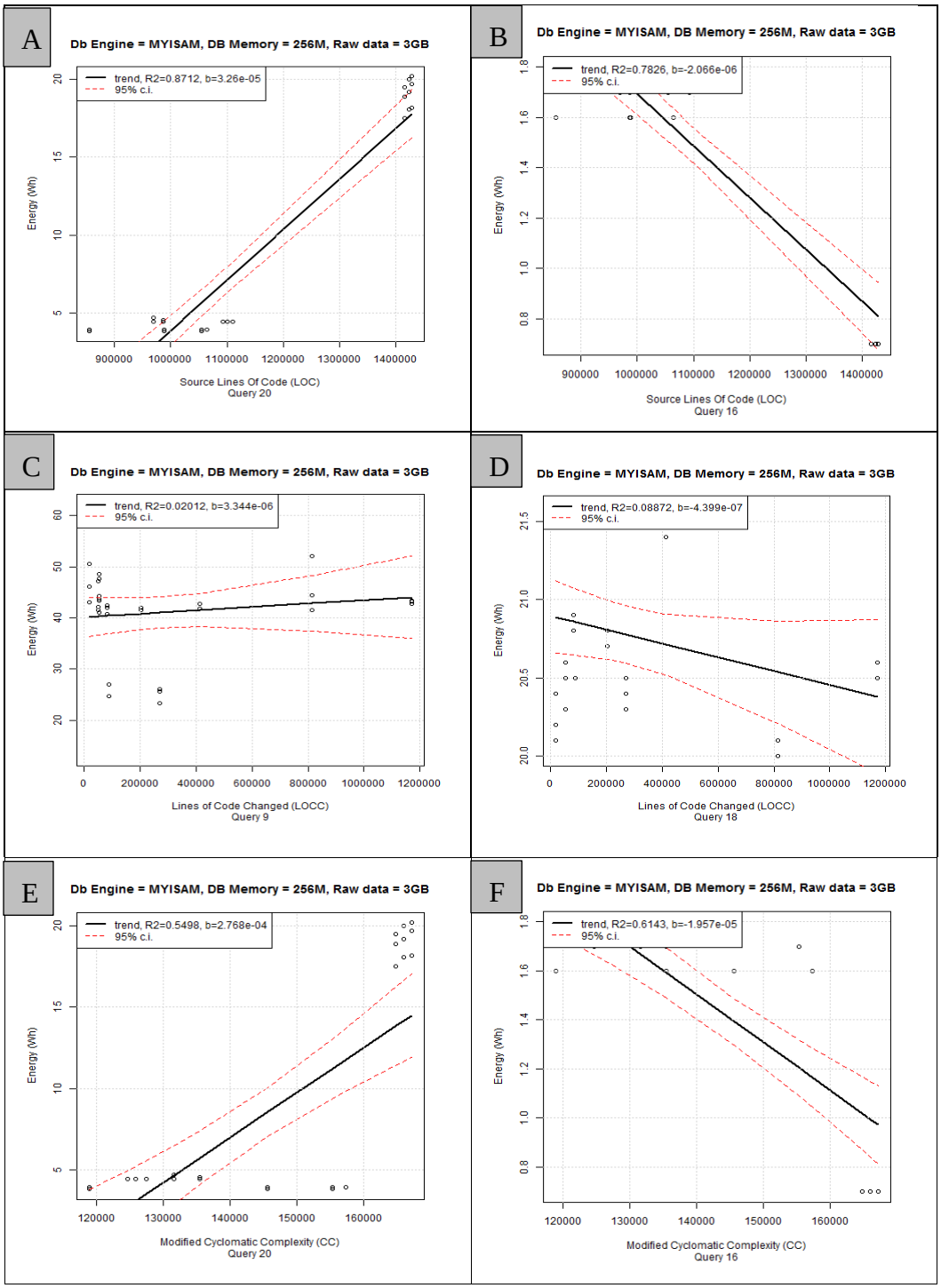


Figure B.24: MyISAM Experiment2; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.6.1.3 Experiment 3: Engine = MyISAM, Key buffer size =1024M, Database size= 1GB

As we can see in Table B.11 , energy consumption of all the queries, except Queries # 2, 4, 8, 16 and 18, exhibit moderate to strong positive correlation with LOC (ρ ranges between and 0.43 and 0.93). Queries #16 and 18, on the other hand, exhibit moderate to strong negative correlation with ρ ranging between -0.93 and -0.50. Queries # 2, 4, and 8, exhibit weak correlation with ranges between 0.12 and 0.18 (samples of the analysis are given in Figure B.25). Energy consumption is weakly correlated (or has no correlation) with LOCC; ρ ranges between - 0.20 and 0.34.

Correlation between energy consumption and MCC/TCC ranges between weak and strong. Energy consumption has strong to moderate positive correlation for (Queries # 1, 5, 10, 12, 13, 20, and 21) and ρ ranges between 0.44 and 0.76; strong negative correlation for Queries #16 and 18 with ρ ranging between -0.87 and -0.75 (similar to the LOC case and Experiments 1 and 2 findings). Queries 2, 3, 4, 6, 7, 8, 9, 14, 15, and 17 exhibit non to weak correlation with ρ ranging between 0.01 and 0.39.

Table B.11: Shows the correlation between the energy consumption and software metrics per each query for MyISAM Experiment 3.

		MYISAM																						
Exp.	Metrics	TPC-H 22 queries																					max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
Exp.3	LOC	0.92	0.12	0.63	0.18	0.81	0.70	0.79	0.18	0.43	0.81	NA	0.79	0.93	0.69	0.79	-0.93	0.63	-0.50	NA	0.93	0.91	NA	0.93	-0.93
	LOCC	0.07	0.05	-0.20	NA	-0.05	0.03	-0.01	0.34	0.25	0.03	NA	-0.05	0.00	0.04	0.04	0.00	-0.03	0.00	NA	0.01	-0.03	NA	0.34	-0.20
	MCC	0.64	0.01	0.20	-0.03	0.46	0.30	0.26	0.11	-0.10	0.50	NA	0.65	0.76	0.14	0.32	-0.76	0.39	-0.87	NA	0.76	0.48	NA	0.76	-0.87
	TCC	0.64	0.02	0.20	-0.05	0.44	0.30	0.24	0.11	-0.11	0.49	NA	0.64	0.75	0.13	0.31	-0.75	0.38	-0.86	NA	0.75	0.47	NA	0.75	-0.86

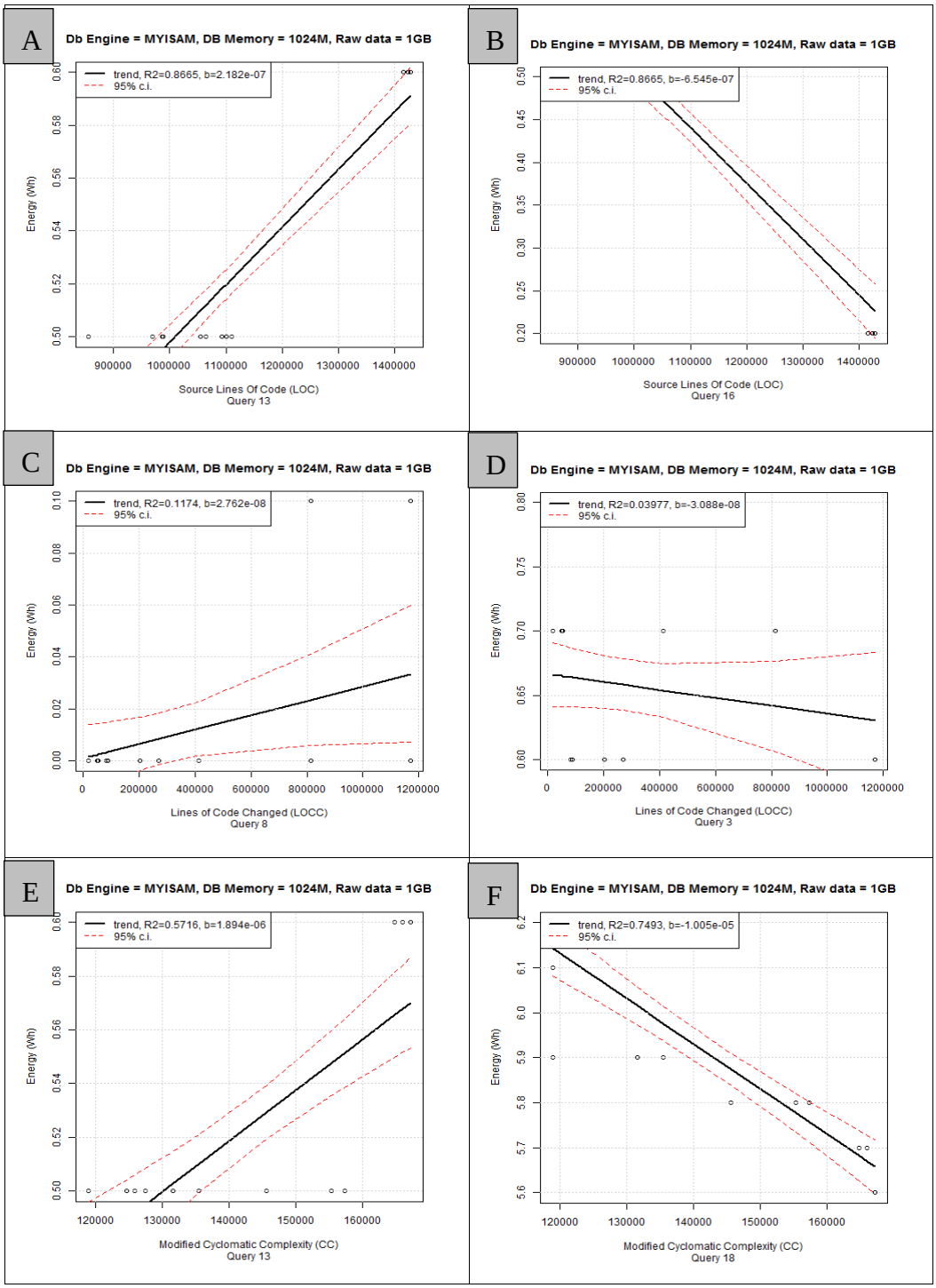


Figure B.25: MyISAM Experiment3; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.6.1.4 Experiment 4: Engine = MyISAM, Key buffer size =1024M, Database size= 3GB

As we can see in Table B.12, energy consumption of all the queries, except queries # 9, 11, 13, 14, 16, 18 and 22, exhibit moderate to strong positive correlation with LOC, ρ ranges between 0.43 and 0.93. Queries # 9, 11, 13, 14, 18 and 22 exhibit none to weak correlation with ρ ranges between -0.14 and 0.39. Query #16 exhibit strong negative correlation with ρ = - 0.89 (samples of the analysis are given in Figure B.26).

Energy consumption exhibits none or weak correlation with LOCC (ρ ranges between - 0.31 and 0.16). The only exception is Query #18, exhibiting moderate negative correlation with ρ = - 0.42.

Correlation between energy consumption and MCC/TCC ranges between weak and strong. Energy consumption has strong to moderate positive correlation for queries # 3, 5, 6, 7, 10, 12, 13, 15, 17, 19, 20, and 21, with ρ ranging between 0.40 and 0.74. Queries # 1, 2, 4, 8, 9, 11 and 22 exhibit none to weak correlation; ρ ranging between -0.19 and 0.39. Queries # 16 and 18 exhibit moderate to strong negative correlation; ρ ranging between - 0.79 and - 0.56.

Table B.12: Shows the correlation between the energy consumption and software metrics per each query for MyISAM Experiment 4.

		MYISAM																						
Exp.	Metrics	TPC-H 22 queries																					max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
Exp.4	LOC	0.83	0.70	0.88	0.43	0.75	0.63	0.85	0.61	0.27	0.81	-0.14	0.50	0.15	0.23	0.81	-0.89	0.66	-0.08	0.89	0.93	0.93	0.39	0.93	-0.89
	LOCC	-0.31	-0.19	-0.21	-0.02	-0.11	-0.15	-0.06	0.16	0.13	-0.07	-0.17	-0.13	-0.09	0.01	-0.24	-0.05	-0.08	-0.42	0.04	-0.01	-0.11	0.13	0.16	-0.42
	MCC	0.37	0.13	0.55	0.30	0.43	0.61	0.44	0.39	-0.18	0.47	0.30	0.62	0.58	0.06	0.43	-0.79	0.40	-0.56	0.65	0.74	0.51	0.07	0.74	-0.79
	TCC	0.36	0.12	0.54	0.29	0.42	0.61	0.43	0.38	-0.19	0.46	0.32	0.62	0.59	0.05	0.42	-0.78	0.40	-0.57	0.64	0.73	0.50	0.07	0.73	-0.78

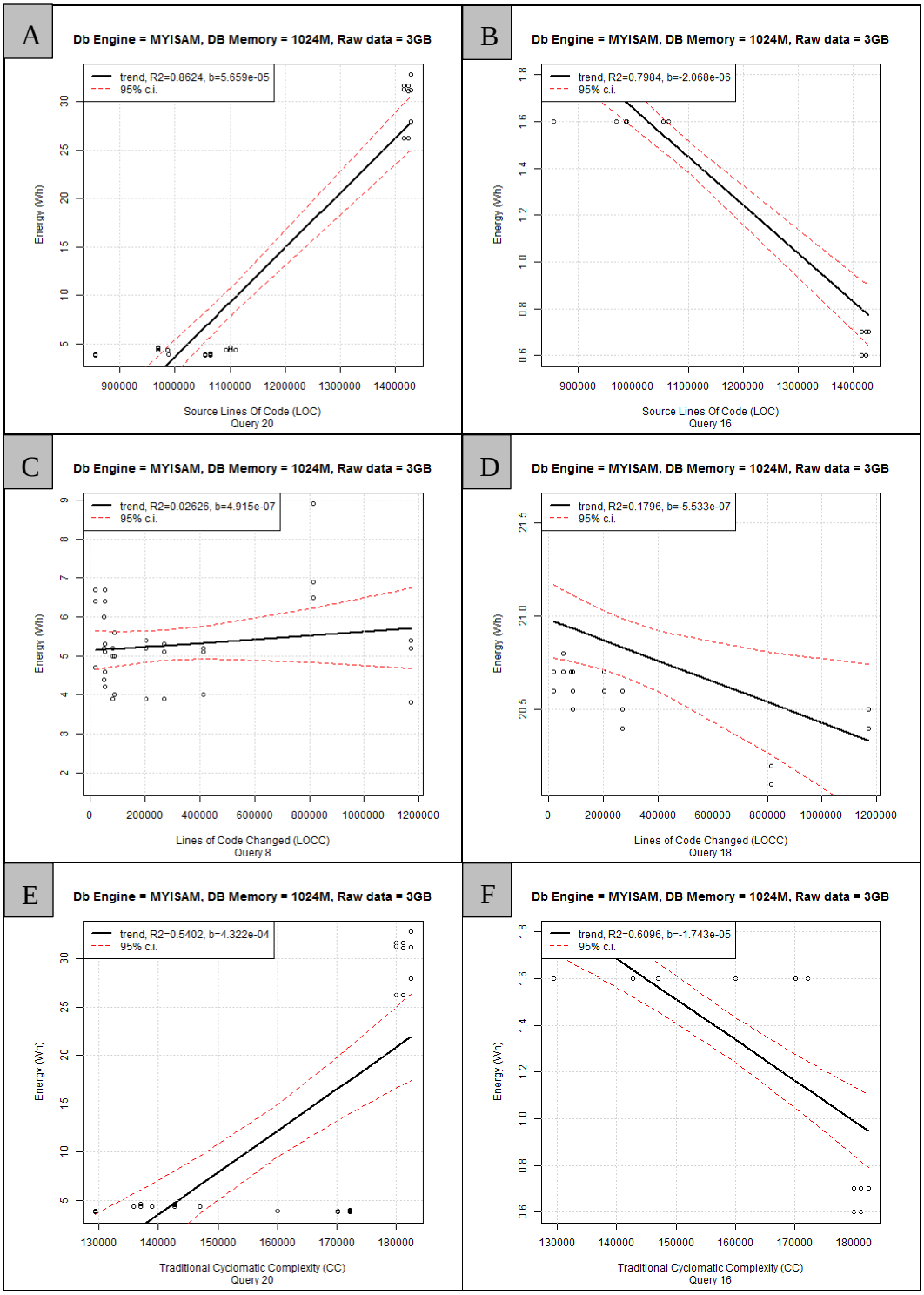


Figure B.26: MyISAM Experiment3; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line.

B.6.2 RQ2: Per-statement Analysis for MyISAM

At the per-statement level, for MyISAM engine, energy consumption of most queries increases with the growth of the code base (measured using LOC); the correlation between these two variables is moderate to strong. However, 1 to 2 queries in each experiment exhibit strong negative correlation (Table B.13).

LOCC has weak to no correlation in most of the cases; 1 query showed moderate negative correlation in Experiment 4 only. The MCC and TCC both showed weak to strong positive correlation in most cases; 2 queries showed moderate to strong negative correlation in the four experiments (Table B.13).

Our discovery of strong negative correlation between software metrics and energy consumption of a small number of queries differs from of our per-run findings (discussed in Chapter 5). We conjecture that this variability can be explained by the fact that the engine executes a small fraction of it's codebase for each particular query; our metrics, on the other hand, were collected for all of the database engine's code. Therefore, by construction, we cannot draw strong conclusions from this analysis.

Since energy consumption and execution time are perfectly related, these findings apply to the correlation between software metrics and execution time as well.

Table B.13: MyISAM; Statistics on negative correlation of software metrics and energy consumption.

MyISAM: number of queries having strong to moderate negative correlation per each experiment				
	LOC	LOCC	MCC	TCC
Experiment 1	2	0	2	2
Experiment 2	1	0	2	2
Experiment 3	2	0	2	2
Experiment 4	1	1	2	2

B.6.3 InnoDB Experiments

Tables B.14 to B.17 show the correlation between the energy consumption and software metrics under study for each individual query for all the four InnoDB experiments. In the following sections, we analyze the results of each experiment.

B.6.3.1 Experiment 5: Engine= InnoDB, Key buffer size =256M, Database size= 1GB

As we can see in Table B.14, energy consumption has weak to strong positive correlation with LOC, ρ ranges between -0.63 and 0.93. Queries # 1, 2, 8, 10, 12, 13, 18, and 20 exhibit moderate to strong positive correlation (ρ ranges between 0.42 and 0.93). Queries # 3, 5, 6, 7, 9, 11, 14, 15, 16, 17, and 21 exhibit weak to no correlation with ρ ranges between - 0.25 and 0.39. Query #4 exhibit moderate negative correlation with ρ = - 0.63. These findings are confirmed by linear regression analysis (samples of the analysis are given in Figure B.27).

Energy consumption is weakly correlated (or has no correlation) with LOCC (as per Table B.14); ρ ranges between - 0.18 and 0.22.

Correlation between energy consumption and MCC/TCC ranges between weak and strong. Energy consumption has strong to moderate positive correlation for queries # 2, 3, 8, 10, 13, 18, and 20, ρ ranging between 0.41 and 0.78. Queries # 1, 4, 5, 6, 7, 9, 11, 12, 14, 15, 16, 17, and 21 exhibit weak to no correlation, ρ ranging between -0.29 and 0.32.

Table B.14: Shows the correlation between the energy consumption and software metrics per each query for InnoDB Experiment 5.

		InnoDB																							
Exp.	Metrics	TPC-H 22 queries																						max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
Exp.5	LOC	0.70	0.42	0.10	-0.63	-0.18	0.21	-0.25	0.52	0.16	0.68	0.39	0.46	0.91	-0.19	0.17	-0.13	0.27	0.73	NA	0.93	-0.10	NA	0.93	-0.63
	LOCC	-0.18	-0.02	0.04	0.10	0.04	-0.06	0.14	0.03	0.22	0.04	-0.17	0.10	-0.02	0.09	-0.02	-0.14	-0.12	0.01	NA	0.00	0.08	NA	0.22	-0.18
	MCC	0.23	0.53	0.42	-0.28	0.10	0.06	0.05	0.44	0.22	0.67	0.32	0.27	0.78	-0.16	0.03	0.14	0.22	0.72	NA	0.76	0.14	NA	0.78	-0.28
	TCC	0.21	0.54	0.41	-0.29	0.12	0.05	0.07	0.42	0.21	0.65	0.31	0.26	0.77	-0.17	0.01	0.16	0.22	0.72	NA	0.75	0.16	NA	0.77	-0.29

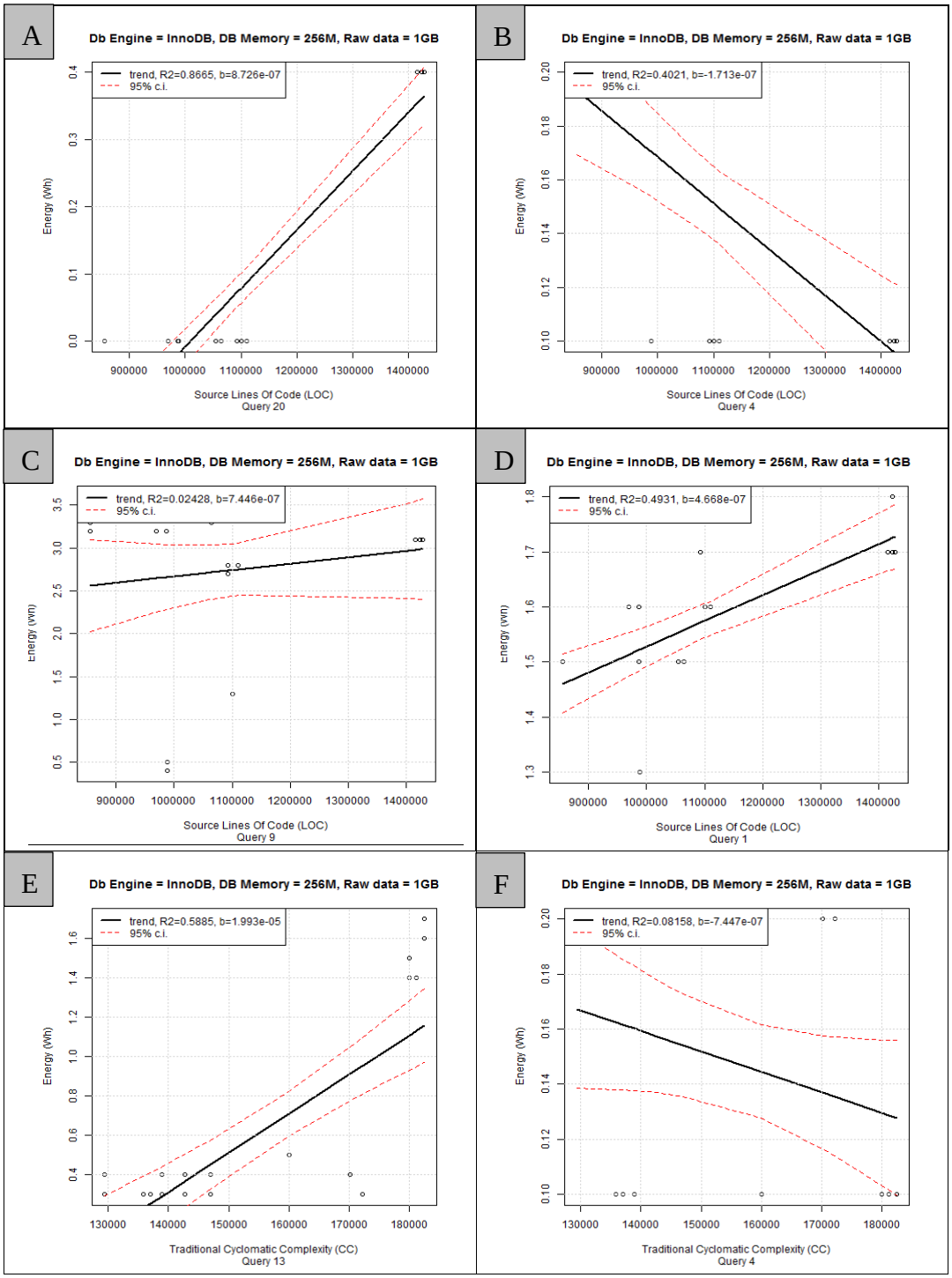


Figure B.27: InnoDB Experiment5; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line

B.6.3.2 Experiment 6: Engine= InnoDB, Key buffer size =256M, Database size= 3GB

As we can see in Table B.15, energy consumption exhibit no to strong correlation with LOC. Queries # 1, 4, 8, 10, 11, 13, and 15, has moderate to strong positive correlation with ρ ranges between 0.49 and 0.93. Queries # 2, 3, 6, 7, 9, 12, 14, and 16-22, have weak to no correlation: ρ ranges between -0.21 and 0.36; Query #5 exhibit moderate negative correlation with = - 0.40. (Samples of the analysis are given in Figure B.28).

Energy consumption exhibit weak correlation with LOCC (except queries # 6 and 7) as shown in Table B.15; ρ ranges between - 0.35 and 0.27. Queries #6 and 7 exhibit moderate negative correlation with ρ ranges between - 0.43 and - 0.42.

Correlation between energy consumption and MCC/TCC ranges between weak and strong. Energy consumption has strong to moderated positive correlation for queries # 10, 11, and 13, ρ ranging between 0.42 and 0.76. Queries #1 - 9, 12, and 14 - 22 exhibit weak to no correlation with ρ ranging between -0.19 and 0.31.

Table B.15: Shows the correlation between the energy consumption and software metrics per each query for InnoDB Experiment 6.

		InnoDB																						
Exp.	Metrics	TPC-H 22 queries																					max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
Exp.6	LOC	0.51	0.00	0.04	0.49	-0.40	0.35	-0.02	0.56	0.33	0.93	0.61	-0.21	0.93	-0.12	0.52	0.31	0.08	0.16	0.15	0.36	0.26	0.22	0.93	-0.40
	LOCC	-0.20	0.16	-0.30	0.08	0.12	-0.43	-0.42	-0.17	-0.27	-0.03	-0.08	0.00	-0.02	0.04	-0.24	-0.10	0.27	-0.19	0.03	-0.35	-0.25	0.03	0.27	-0.43
	MCC	0.16	-0.08	0.18	0.31	-0.18	0.11	-0.11	0.03	0.02	0.76	0.44	0.08	0.75	0.12	0.07	0.04	0.08	0.30	0.10	0.07	-0.01	-0.03	0.76	-0.18
	TCC	0.14	-0.09	0.20	0.29	-0.19	0.10	-0.13	0.02	0.00	0.75	0.42	0.10	0.75	0.13	0.05	0.02	0.07	0.30	0.08	0.05	-0.03	-0.04	0.75	-0.19

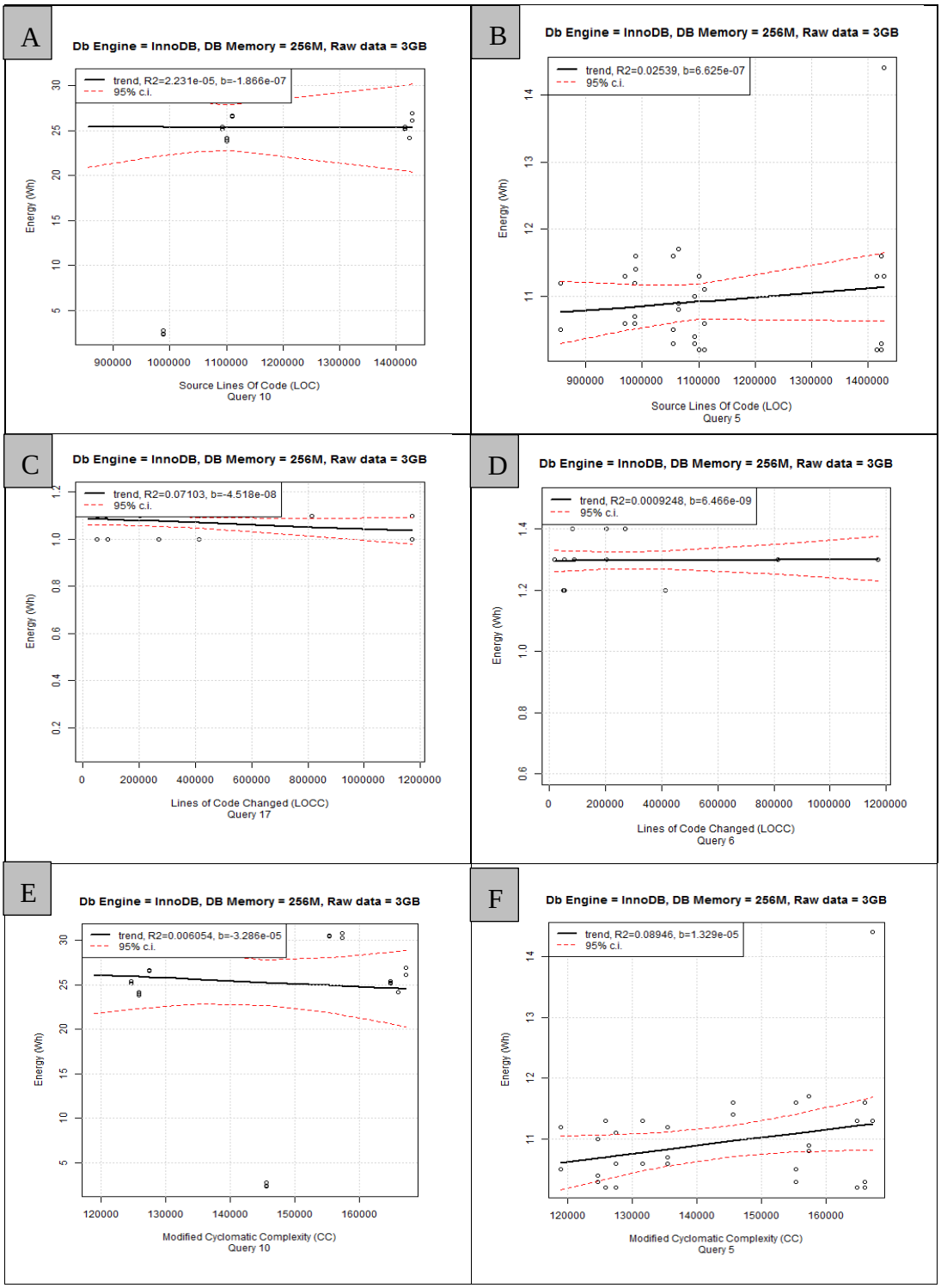


Figure B.28: InnoDB Experiment 6; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line

B.6.3.3 Experiment 7: Engine=InnoDB, Key buffer size=1024M, Database size= 1GB

As we can see in Table B.16, energy consumption exhibit strong to moderate correlation with LOC. Queries #1, 6, 8, 12, 15, and 20 exhibit strong to moderate positive correlation, (ρ ranges between 0.64 and 0.93. Queries #2, 3, 5, 7, 9, 10, 14, 16, 18, and 21 exhibit weak to no correlation, ρ ranges between and - 0.29 and 0.21. Query #13 exhibit moderate negative correlation with ρ= - 0.63. (Samples of the analysis are given in Figure B.29).

Energy consumption is weakly correlated (or has no correlation) with LOCC (as per Table B.16); ρ ranges between - 0.30 and 0.18. Correlation between energy consumption and MCC/TCC ranges between weak and strong. Energy consumption has strong positive correlation for queries #6, 15, and 20, with ρ ranging between 0.75 and 0.76. Queries #1-3, 5, 7-10, 12-14, 16, 18, and 21, exhibit moderate to no correlation with ρ ranging between - 0.38 and 0.40. Query #14 exhibit moderate negative correlation with ρ = - 0.40 in TCC correlation.

Table B.16: Shows the correlation between the energy consumption and software metrics per each query for InnoDB Experiment 7.

		InnoDB																						
Exp.	Metrics	TPC-H 22 queries																					max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
Exp.7	LOC	0.71	-0.21	-0.16	NA	-0.22	0.93	-0.22	0.69	-0.28	-0.29	NA	0.64	-0.63	-0.21	0.93	-0.22	NA	0.21	NA	0.93	-0.14	NA	0.93	-0.63
	LOCC	-0.30	-0.06	0.15	NA	NA	0.00	NA	0.00	0.14	-0.15	NA	-0.06	-0.03	0.18	0.00	NA	NA	-0.01	NA	0.00	0.06	NA	0.18	-0.30
	MCC	0.26	0.06	0.15	NA	0.04	0.76	0.04	0.40	-0.11	0.24	NA	0.38	-0.23	-0.38	0.76	0.04	NA	0.38	NA	0.76	0.09	NA	0.76	-0.38
	TCC	0.25	0.08	0.16	NA	0.06	0.75	0.06	0.38	-0.12	0.26	NA	0.36	-0.22	-0.40	0.75	0.06	NA	0.39	NA	0.75	0.11	NA	0.75	-0.40

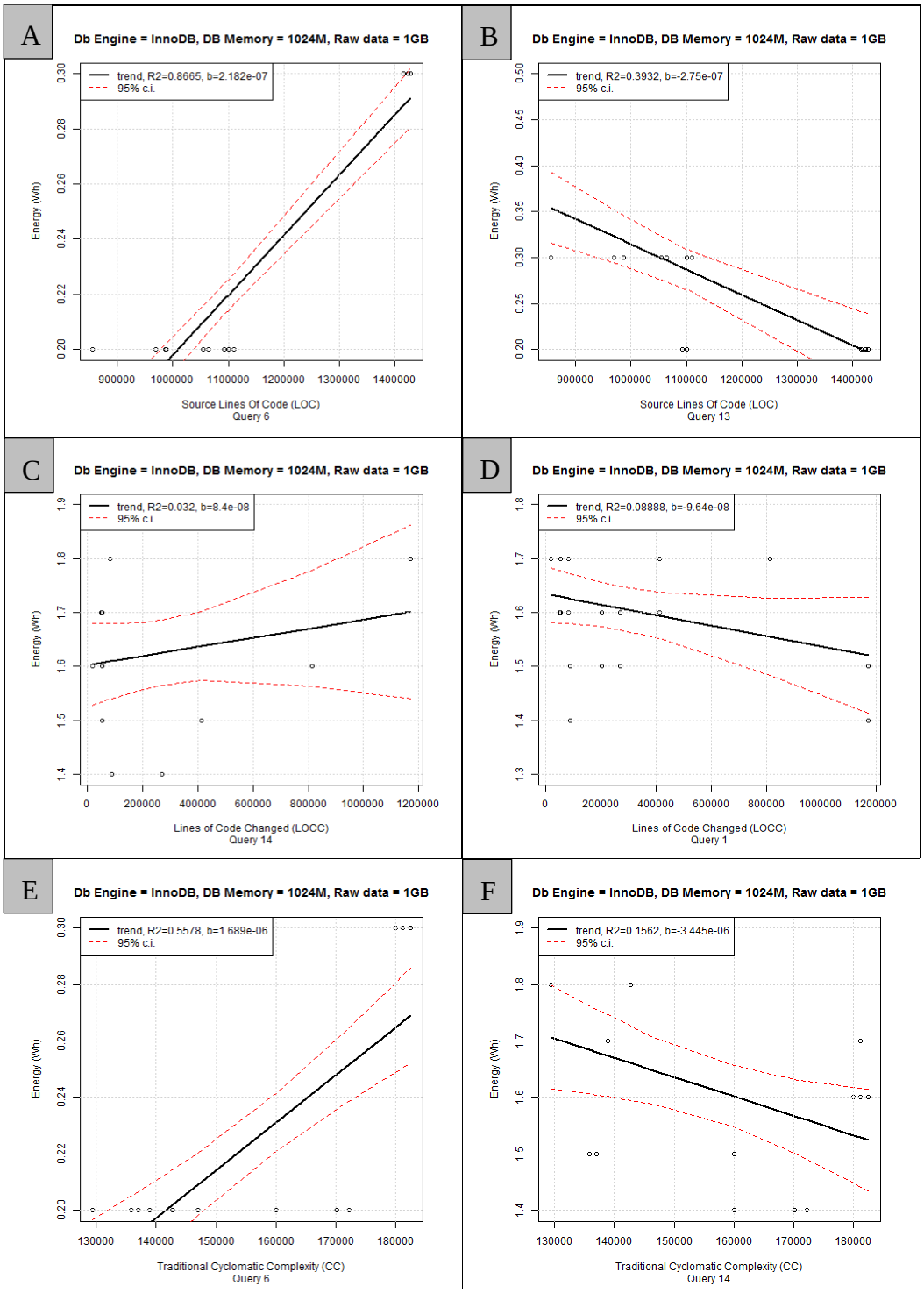


Figure B.29: InnoDB Experiment 7; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line

B.6.3.4 Experiment 8: Engine=InnoDB, Key buffer size=1024M, Database size= 3GB

As we can see in Table B.17, Queries # 1, 4, 5, 8, 10, 11, 13, 15, and 17, exhibit moderate to strong correlation between LOC and energy consumed with ρ ranges between 0.42 and 0.93. Queries # 2, 3, 6, 7, 9, 12, 14, 16, and 19 – 22, exhibit weak to no correlation, ρ ranges between -0.09 and 0.25; Query #18 exhibit moderate negative correlation with ρ = - 0.52. Samples of the analysis are given in Figure B.30. Energy consumption exhibits none or weak correlation with LOCC (as per Table B.17);ρ ranges between -0.30 and 0.33.

Correlation between energy consumption and MCC/TCC ranges between weak and strong. Energy consumption has strong to moderate positive correlation for Queries #10 and 13 with ρ ranging between 0.74 and 0.76. The remaining queries, except Query #18, have none or weak correlation, with ρ ranging between -0.36 and 0.34. Query #18 exhibit moderate negative correlation with ρ between -0.66 and -0.65.

Table B.17: Shows the correlation between the energy consumption and software metrics per each query for InnoDB Experiment 8.

		InnoDB																							
Exp.	Metrics	TPC-H 22 queries																						max	min
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
Exp.8	LOC	0.61	0.01	0.06	0.55	0.76	-0.06	0.25	0.81	0.25	0.93	0.51	-0.09	0.92	-0.09	0.74	-0.06	0.42	-0.52	0.22	0.00	0.10	-0.07	0.93	-0.52
	LOCC	-0.30	0.20	-0.18	-0.02	-0.22	0.07	-0.26	-0.15	0.33	0.03	0.16	-0.18	0.04	0.02	-0.10	0.13	0.19	0.08	-0.14	0.25	0.16	0.06	0.33	-0.30
	MCC	0.13	-0.20	0.11	0.34	0.34	-0.35	0.05	0.27	-0.07	0.76	0.17	0.09	0.75	0.14	0.18	-0.26	0.13	-0.65	0.08	-0.17	-0.18	-0.33	0.76	-0.65
	TCC	0.12	-0.21	0.13	0.33	0.33	-0.36	0.04	0.26	-0.08	0.75	0.16	0.11	0.74	0.16	0.17	-0.28	0.12	-0.66	0.07	-0.18	-0.19	-0.34	0.75	-0.66

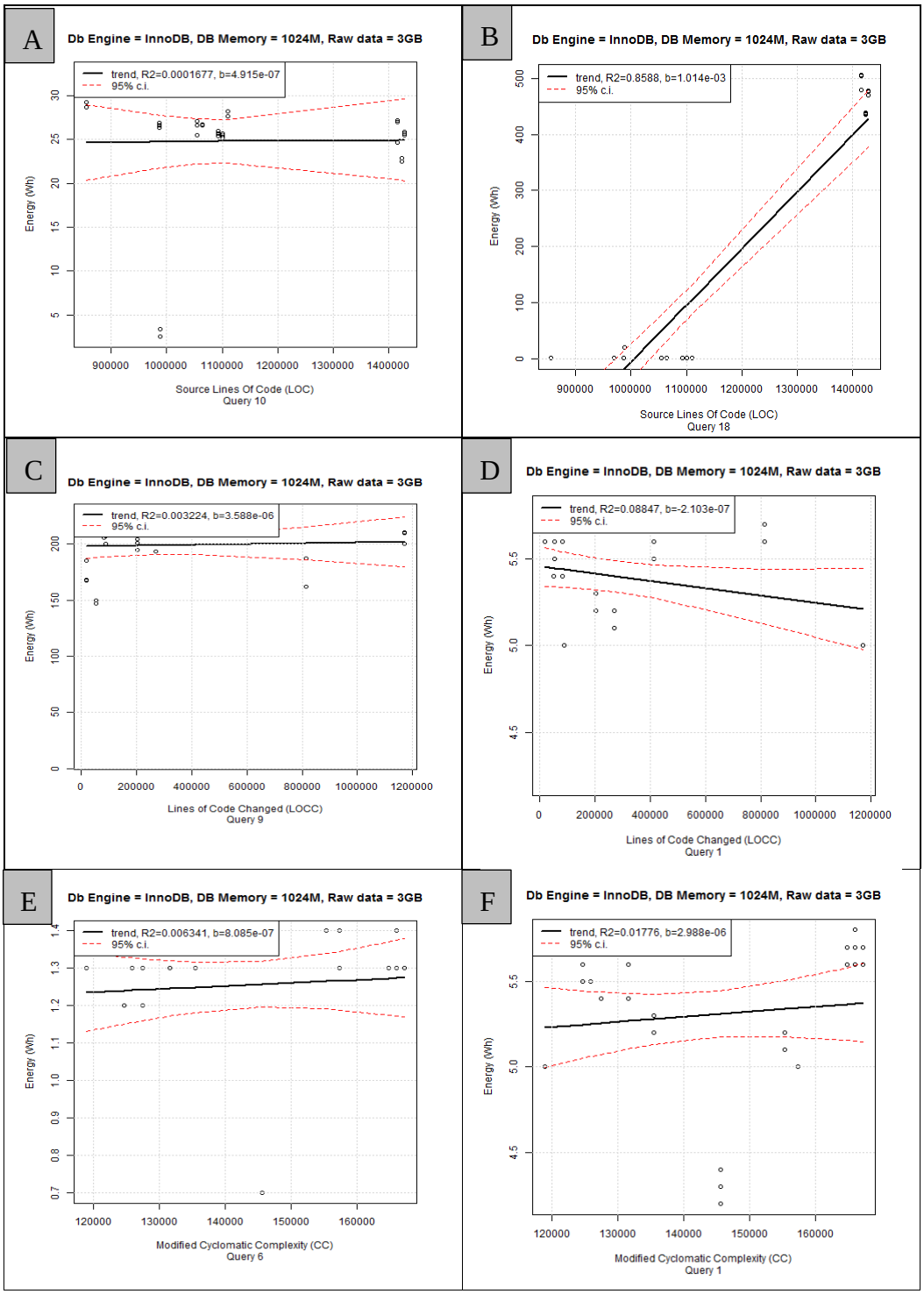


Figure B.30: InnoDB Experiment 7; Subplot A is a scatterplot of query with max LOC against energy; subplot B is a scatterplot of query with min LOC against energy; Subplot C a scatterplot of max LOCC against energy; Subplot D is a scatterplot of min LOCC against energy; Subplot E is a scatterplot of query with max MCC against energy; subplot F is a scatterplot of query with min MCC against energy. On all subplots: black line depicts a trend line obtained using linear regression; dotted red lines show 95% confidence interval of the trend line

B.6.4 RQ2: Per-statement Analysis for InnoDB

We showed that energy consumption of the majority of queries in InnoDB experiments has a moderate to strong correlation with LOC. However, 1 query showed moderate negative correlation in all four experiments (Table B.18).

LOCC has weak to no correlation with energy consumption; only Experiment 6 showed that 2 queries exhibit moderate negative correlation. MCC and TCC both showed moderate to strong correlation with energy consumption; only Experiment 7 and 8 showed that 1 to 2 queries exhibit moderate negative correlation (Table B.18).

Since energy consumption and execution time are perfectly related, the findings also apply to the correlation between software metrics and execution time.

Table B.18: InnoDB; Statistics on negative correlation of software metrics and energy consumption.

InnoDB: number of queries having moderate negative correlation per each experiment				
	LOC	LOCC	MCC	TCC
Experiment 5	1	0	0	0
Experiment 6	1	2	0	0
Experiment 7	1	0	0	1
Experiment 8	1	0	1	1

6 References

[1] Mills, M. P. (2013). The cloud begins with coal. *Digital Power Group.* Retrieved 25 July 2015, *Online at: http://www. tech-pundit. com/wpcontent/uploads/2013/07/ Cloud_Begins_With_Coal. pdf.*

[2] Green IT and Green Software | October 2014 Theme - IEEECS. Retrieved 25 July 2015, from http://www.computer.org/web/computingnow/archive/october2014

[3] Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO_2 Emissions. Retrieved 25 July 2015, from http://www.gartner.com/newsroom/id/503867

[4] Hindle, A. (2012, June). Green mining: investigating power consumption across versions. In *Software Engineering (ICSE), 2012 34th International Conference on* (pp. 1301-1304). IEEE.

[5] MySQL | The Most Popular Open-Source Database | Oracle. Retrieved 25 July 2015, from http://www.oracle.com/us/products/mysql/overview/index.html

[6] CLOC -- Count Lines of Code.Retrieved 25 July 2015, from http://cloc.sourceforge.net/

[7] Patil, A., & Goti, A. (2013). Green computing in communication networks. *Proc Int J Comput Appl*, *68*(23), 1-5.

[8] Kochhar, E. N., & Garg, E. A. (2011). Eco-friendly computing: green computing. *International Journal of Computing and Business Research, ISSN (Online)*, 2229-6166.

[9] Naumann, S., Dick, M., Kern, E., & Johann, T. (2011). The greensoft model: A reference model for green and sustainable software and its engineering.*Sustainable Computing: Informatics and Systems*, *1*(4), 294-304.

[10] Singh, S. K. (2011). *Database Systems: Concepts, Design and Applications*. Pearson Education India.

[11] Tsirogiannis, D., Harizopoulos, S., & Shah, M. A. (2010, June). Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (pp. 231-242). ACM.

[12] Chen, C., He, B., Tang, X., Chen, C., & Liu, Y. (2013). Green Databases Through Integration of Renewable Energy. In *CIDR*.

[13] Kunjir, M., Birwa, P. K., & Haritsa, J. R. (2012, March). Peak power plays in database engines. In *Proceedings of the 15th International Conference on Extending Database Technology* (pp. 444-455). ACM.

[14] Lang, W., Kandhan, R., & Patel, J. M. (2011). Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race. *IEEE Data Eng. Bull.*, *34*(1), 12-23.

[15] Xu, Z., Tu, Y. C., & Wang, X. (2010, March). Exploring power-performance tradeoffs in database systems. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on* (pp. 485-496). IEEE.

[16] Le, K., Bianchini, R., Nguyen, T. D., Bilgir, O., & Martonosi, M. (2010, August). Capping the brown energy consumption of internet services at low cost. In*Green Computing Conference, 2010 International* (pp. 3-14). IEEE.

[17] Bifet, A. (2013). Mining big data in real time. *Informatica*, *37*(1).

[18] Madden, S. (2012). From databases to big data. *IEEE Internet Computing*, (3), (pp.4-6).

[19] Cook, G., & Van Horn, J. (2011). How dirty is your data? A look at the energy choices that power cloud computing. *Greenpeace (April 2011)*.

[20] Big Data Goes Green - Renewable Energy World. Retrieved 25 July 2015, from http://www.renewableenergyworld.com/articles/print/volume-15/issue-3/solar-energy/big-data-goes-green.html

[21] Liu, X., Wang, J., Wang, H., & Gao, H. (2013, July). Generating Power-Efficient Query Execution Plan. In *2nd International Conference on Advances in Computer Science and Engineering (CSE 2013)*. Atlantis Press.

[22] Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., & Price, T. G. (1979, May). Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data* (pp. 23-34). ACM.

[23] Delaluz, V., Kandemir, M., Vijaykrishnan, N., Sivasubramaniam, A., & Irwin, M. J. (2001). DRAM energy management using software and hardware directed power mode control. In *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on* (pp. 159-169). IEEE.

[24] Amsel, N., & Tomlinson, B. (2010, April). Green tracker: a tool for estimating the energy consumption of software. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems* (pp. 3337-3342). ACM.

[25] Gurumurthi, S., Sivasubramaniam, A., Irwin, M. J., Vijaykrishnan, N., & Kandemir, M. (2002, February). Using complete machine simulation for software power estimation: The softwatt approach. In *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*(pp. 141-150). IEEE.

[26] Tiwari, V., Malik, S., Wolfe, A., & Lee, M. T. C. (1996). Instruction level power analysis and optimization of software. In *Technologies for wireless computing*(pp. 139-154). Springer US.

[27] Mittal, R., Kansal, A., & Chandra, R. (2012, August). Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual international conference on Mobile computing and networking* (pp. 317-328). ACM.

[28] Bircher, W. L., & John, L. K. (2007, April). Complete system power estimation: A trickle-down approach based on performance events. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on* (pp. 158-168). IEEE.

[29] Greenawalt, P. M. (1994, January). Modeling power management for hard disks. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1994., MASCOTS'94., Proceedings of the Second International Workshop on* (pp. 62-66). IEEE.

[30] Stemm, M. (1997). Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE transactions on Communications*, *80*(8), 1125-1131.

[31] Li, K., Kumpf, R., Horton, P., & Anderson, T. E. (1994, January). A quantitative analysis of disk drive power management in portable computers. In *USENIX winter* (pp. 279-291).

[32] Selby, J. (2011). *Unconventional applications of compiler analysis* (Doctoral dissertation, University of Waterloo).

[33] Fei, Y., Ravi, S., Raghunathan, A., & Jha, N. K. (2007). Energy-optimizing source code transformations for operating system-driven embedded software.*ACM Transactions on Embedded Computing Systems (TECS)*, *7*(1), 2.

[34] Feng, X., Ge, R., & Cameron, K. W. (2005, April). Power and energy profiling of scientific applications on distributed systems. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International* (pp. 34-34). IEEE.

[35] Kagdi, H., Collard, M. L., & Maletic, J. I. (2007). A survey and taxonomy of approaches for mining software repositories in the context of software evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, *19*(2), 77-131.

[36] Shang, W., Jiang, Z. M., Adams, B., Hassan, A. E., Godfrey, M. W., Nasser, M., & Flora, P. (2014). An exploratory study of the evolution of communicated information about the execution of large software systems. *Journal of Software: Evolution and Process*, *26*(1), 3-26.

[37] Gupta, A., Zimmermann, T., Bird, C., Naggapan, N., Bhat, T., & Emran, S. (2011). *Energy Consumption in Windows Phone*. Microsoft Research, Tech. Rep. MSR-TR-2011-106.

[38] Jimeno, M., Christensen, K., & Nordman, B. (2008, December). A network connection proxy to enable hosts to sleep and save energy. In *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International* (pp. 101-110). IEEE.

[39] Li, W., & Henry, S. (1993). Object-oriented metrics that predict maintainability.*Journal of systems and software*, *23*(2), 111-122.

[40] Park, R. E. (1992). *Software size measurement: A framework for counting source statements* (No. CMU/SEI/92-TR-20). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[41] Laird, L. M., & Brennan, M. C. (2006). *Software measurement and estimation: a practical approach* (Vol. 2). John Wiley & Sons.

[42] Nagappan, N., & Ball, T. (2005, May). Use of relative code churn measures to predict system defect density. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on* (pp. 284-292). IEEE.

[43] Nagappan, N., & Ball, T. (2007, September). Using software dependencies and churn metrics to predict field failures: An empirical case study. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on* (pp. 364-373). IEEE.

[44] McCabe, T. J. (1976). A complexity measure. *Software Engineering, IEEE Transactions on*, (4), 308-320.

[45] Watts up? Products: Meters. Retrieved 25 July 2015, from https://www.wattsupmeters.com/secure/products.php?pn=0&wai=23&spec=4

[46] DB-Engines Ranking - popularity ranking of relational DBMS. Retrieved 25 July 2015, from http://db-engines.com/en/ranking/relational+dbms

[47] MySQL :: The world's most popular open source database. Retrieved 25 July 2015, from http://www.mysql.com/

[48] MySQL :: MySQL 5.5 Reference Manual :: 14 The InnoDB Storage Engine. Retrieved 25 July 2015, from https://dev.mysql.com/doc/refman/5.5/en/innodb-storage-engine.html

[49] TPC-H - Homepage. Retrieved 25 July 2015, from http://www.tpc.org/tpch/

[50] OLTP vs. OLAP. Retrieved 25 July 2015, from http://datawarehouse4u.info/OLTP-vs-OLAP.html

[51] SYSSTAT. Retrieved 25 July 2015, from http://sebastien.godard.pagesperso-orange.fr/

[52] PMCCABE pmccabe.1 man page. Retrieved 25 July 2015, from https://people.debian.org/~bame/pmccabe/pmccabe.1

[53] Christine, D. P., & John, R. (2002). Statistics Without Maths for Psychology Using SPSS for Windows. Prentice Hall.

[54] Lind, R. K., & Vairavan, K. (1989). An experimental investigation of software metrics and their relationship to software development effort. *IEEE Transactions on Software Engineering*, (5), 649-653.

[55] Penzenstadler, B., Raturi, A., Richardson, D., & Tomlinson, B. (2014). Safety, security, now sustainability: The nonfunctional requirement for the 21st century.*Software, IEEE*, *31*(3), 40-47.

[56] Mısırlı, A. T., Çağlayan, B., Miranskyy, A. V., Bener, A., & Ruffolo, N. (2011, May). Different strokes for different folks: a case study on software metrics for different defect categories. In *Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics* (pp. 45-51). ACM.

[57] Miranskyy, A., Caglayan, B., Bener, A., & Cialini, E. (2014, September). Effect of temporal collaboration network, maintenance activity, and experience on defect exposure. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (p. 27). ACM.

[58] Yin, R. K. (2013). *Case study research: Design and methods*. Sage publications.

[59] SQLite Home Page. Retrieved 25 July 2015, from https://www.sqlite.org/

[60] MySQL :: MySQL 5.0 Reference Manual :: 5.1.4 Server System Variables. Retrieved 25 July 2015, from https://dev.mysql.com/doc/refman/5.0/en/server-system-variables.html

[61] What is the recommended value of key_buffer_size in MySQL? Retrieved 25 July 2015, from http://www.ewhathow.com/2013/09/what-is-the-recommended-value-of-key_buffer_size-in-mysql/