EVALUATING MITIGATIONS FOR MELTDOWN AND SPECTRE
Benchmarking performance of mitigations against database management systems with OLTP workload

BEDÖMNING AV MITIGERING MOT SPECTRE OCH MELTDOWN
Prestandamätningar av databashanteringssystem efter mitigering mot Spectre och Meltdown med OLTP arbetsbelastning

Bachelor Degree Project in Information Technology
G2E 22.5 hp
Spring term 2018

Victor Nilsson, a15vicni@student.his.se

Supervisor: Dennis Modig
Examiner: Jianguo Ding
Abstract

With Spectre and Meltdown out in the public, a rushed effort was made to patch these vulnerabilities by operating system vendors. However, with the mitigations against said vulnerabilities there will be some form of performance impact. This study aims to find out how much of an impact the software mitigations against Spectre and Meltdown have on database management systems during an online transaction processing workload.

An experiment was carried out to evaluate two popular open-source database management systems and see how they were affected before and after the software mitigations against Spectre and Meltdown was applied.

The study found that there is an average of 4–5% impact on the performance when the software mitigations is applied. The study also compared the two database management systems with each other and found that PostgreSQL can have a reduced performance of about 27% when both a hypervisor and the operating system is patched against Spectre and Meltdown.

Keywords: Spectre, Meltdown, Database, Database Management Systems, MariaDB, PostgreSQL, OLTP, Transactions Per Minute, Performance

Sammanfattning

När Spectre och Meltdown tillkännagavs gjordes en snabb insats för att korrigera dessa sårbarheter av operativsystemleverantörer. Men med mildringarna mot dessa sårbarheter kommer det att finnas någon form av prestationspåverkan. Denna studie syftar till att ta reda på hur mycket av en påverkan uppdateringarna mot Spectre och Meltdown har på databashanteringssystem under en online-transaktionsbehandlings arbetsbelastning.

Ett experiment gjordes för att utvärdera två populära databashanteringssystem baserat på fri mjukvara och se hur de påverkades före och efter att uppdateringarna mot Spectre och Meltdown applicerats i en Linux maskin.

Studien fann att det i genomsnitt är 4–5% påverkan på prestandan när uppdateringarna tillämpas. Studien jämförde också de två databashanteringssystemen med varandra och fann att PostgreSQL kan ha en reducerad prestanda på cirka 27% när både det virtuella maskinhanteringssystemet och operativsystemet är uppdaterad mot Spectre och Meltdown.

Nyckelord: Spectre, Meltdown, Databas, Databashanteringssystem, MariaDB, PostgreSQL, OLTP, Transaktioner per minut, Prestanda
# Table of Contents

1. Introduction .................................................................................................................. 1
2. Background .................................................................................................................... 2
   2.1. Spectre and Meltdown ............................................................................................ 2
   2.2. Databases and database management systems ....................................................... 3
   2.3. MariaDB .................................................................................................................. 3
   2.4. PostgreSQL ............................................................................................................ 4
   2.5. Transactional Workload .......................................................................................... 4
   2.6. Previous Work ....................................................................................................... 4
3. Problem Definition ........................................................................................................... 4
   3.1. Purpose .................................................................................................................. 5
   3.2. Aim of the study .................................................................................................... 5
   3.3. Research question ................................................................................................. 5
   3.4. Motivation ............................................................................................................. 5
   3.5. Hypothesis ............................................................................................................ 5
   3.6. Objectives ............................................................................................................. 6
4. Methodology .................................................................................................................... 7
   4.1. Experimentation Process ....................................................................................... 7
   4.2. Independent variables ......................................................................................... 7
   4.3. Dependent variables ............................................................................................ 7
   4.4. Scope .................................................................................................................... 7
   4.5. HammerDB ........................................................................................................... 7
   4.6. Experiment ............................................................................................................ 8
   4.7. HammerDB Configuration .................................................................................... 9
   4.8. Pilot test ............................................................................................................... 9
   4.9. Ethics ................................................................................................................... 10
   4.10. Validity threats ................................................................................................... 10
5. Implementation ............................................................................................................... 11
   5.1. Debian .................................................................................................................. 11
   5.2. MariaDB .............................................................................................................. 11
   5.3. PostgreSQL ........................................................................................................ 11
   5.4. HammerDB ......................................................................................................... 11
6. Results ........................................................................................................................... 13
   6.1. Transactions Per Minute - MariaDB ...................................................................... 13
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2. Transactions Per Minute – MariaDB with updated hypervisor</td>
<td>14</td>
</tr>
<tr>
<td>6.3. Transactions Per Minute – MariaDB summary</td>
<td>15</td>
</tr>
<tr>
<td>6.4. Transactions Per Minute - PostgreSQL</td>
<td>16</td>
</tr>
<tr>
<td>6.5. Transactions Per Minute – PostgreSQL with updated hypervisor</td>
<td>17</td>
</tr>
<tr>
<td>6.6. Transactions Per Minute – PostgreSQL summary</td>
<td>18</td>
</tr>
<tr>
<td>6.7. New Orders Per Minute</td>
<td>19</td>
</tr>
<tr>
<td>6.8. New Orders Per Minute – Patched hypervisor</td>
<td>20</td>
</tr>
<tr>
<td>6.9. New Orders Per Minute – Comparing results</td>
<td>21</td>
</tr>
<tr>
<td>7. Analysis</td>
<td>22</td>
</tr>
<tr>
<td>8. Conclusion</td>
<td>23</td>
</tr>
<tr>
<td>9. Discussion</td>
<td>24</td>
</tr>
<tr>
<td>10. Future Work</td>
<td>25</td>
</tr>
<tr>
<td>References</td>
<td>26</td>
</tr>
</tbody>
</table>
1. Introduction

With an ever-increasing need of creating and accessing data, databases and Database Management Systems (DBMS) is in ever-expanding need. Databases and DBMSs power websites around the world by providing them with an easy, powerful and scalable backend which keeps data structured and organized, (Bartholomew, Getting Started with MariaDB, 2013).

In January 2018 the two hardware vulnerabilities Spectre and Meltdown was revealed to the public with patches being made by all operating system (OS) vendors to mitigate these vulnerabilities. However, with these mitigations comes a performance impact which varies depending on the workload. This study aims to find out how much of an impact the software mitigations against Spectre and Meltdown have on DBMSs during an OLTP workload.
2. Background

With the recent hardware vulnerabilities disclosed as Spectre (CVE-2017-5753 and CVE-2017-5715) and Meltdown (CVE-2017-5754), operating system vendors rushed to patch the disclosed vulnerabilities. However, due to the nature of Meltdown and Spectre the mitigation against them will include some degree of decreased performance in certain type of workloads. Red Hat has shared their results of the impact on the performance in different workloads after the system has been patched against Spectre and Meltdown. According to Red Hat, they found a decreased performance of 8-19% with high kernel-to-user space transitions and online transaction processing (OLTP) database workloads, (Red Hat, 2018).

2.1. Spectre and Meltdown

Spectre and Meltdown is the names given to the vulnerabilities disclosed in January 2018 which utilize speculative execution to its advantage. Speculative execution is an optimization technique where the central processing unit (CPU) will try and predict the next set of instructions to be as efficient as possible. It does this by executing code branches the program have not yet arrived at, to be ahead of the program and minimize CPU idle time, (Intel, 2018).

However, if the CPU predicts wrong it must undo the speculative code and roll back to the start of the branch. The issue is that it does not roll back every step that the CPU took before it realized it was going down the wrong branch and therefore can leave remnants of data that can be fetched in caches later by the attacker, (Intel, 2018).

Today’s operating systems separate user memory and kernel memory to keep user applications from reading each other’s memory and to restrict access to kernel memory. This is called memory isolation, (Jomaa, Nowak, Grimaud, & Hym, 2016). Meltdown effectively breaks down that barrier and allows for an attacker to read kernel memory from an unprivileged user process, (Lipp, et al., 2018). Spectre can be exploited to extract information from other running processes such as the web browser. It does this by tricking the processor into speculatively executing instructions that would not have been executed under normal circumstances, (Kocher, et al., 2018).

Modern operating systems runs code at different security levels, the kernel runs code at the most privileged level (kernel space) and almost all other applications run at a lower privilege (user space). The access between the kernel space and user space is handled with page tables. It prevents access to kernel space from applications that should not be allowed to access it. However, sometimes the application might need access to certain privileged actions that is handled by the kernel. So, to be as efficient as possible, the kernel space has traditionally always been mapped in the page table. The mitigations developed against Spectre and Meltdown improve upon the security implemented by the page table.

The mitigations against these attacks are different to each attack. The first one is called Kernel page-table isolation (KPTI) and is used to mitigate against Meltdown. KPTI works by separating the user-space and kernel-space page tables. Spectre is mitigated through CPU-microcode updates which includes the Indirect Branch Restricted Speculation (IBRS), Indirect Branch Predictor Barrier (IBPB), Single Thread Indirect Branch Predictor (STIBP) and retpoline. In a virtualized environment, the patches must be applied to both the hypervisor and to the guest VM, (Ubuntu, 2018).
Retpoline, IBRS, IBPB and STIBP implements further mitigations against Spectre. These mitigations restrict the way Spectre can take advantage of speculative execution and is the only way to be fully protected against Spectre, (Google, n.d.).

With these patches applied, Red Hat published its findings regarding the performance impact of these patches on various systems. According to Red Hat, they saw a decrease of 8-19% in performance in OLTP database workloads and with the recent retpoline patches applied, the expected performance impact improved to 4-8%, (Red Hat, 2018).

2.2. Databases and database management systems

A Database management systems (DBMS) is a software for creating, manipulating, processing and retrieving information from a database, (Poljak, Poščić, & Jakšić, 2017). A database is a data structure that stores information that is organized. Most databases are often structured in the form of tables, these tables would then be used to store information that has a need to be accessed. This makes databases and DBMS useful for e-commerce websites which need to store their products in an organized structure and for easy access and manipulation. Most databases today are relational databases, which uses tables of data that are related by common fields, (MariaDB, 2015). Databases uses the language Structured Query Language (SQL) to make changes or to retrieve data, (MariaDB, 2017). All SQL statements that requests data is called an SQL query.

2.3. MariaDB

MariaDB is an open-source fork of the original MySQL. Since its initial release in January 2009, MariaDB has become one of the most popular relational database management systems (DBMS) today. MariaDB came to life when Sun Microsystems bought MySQL for $1 billion dollars, which then got acquired by Oracle. Unhappy with the acquisition, some developers forked MySQL and named it MariaDB. Names including Google, (Clark, 2013), Wordpress, (MariaDB, n.d.) and the Wikimedia foundation, (Feldman, 2013) all use MariaDB. MariaDB has the goal to act as a drop-in replacement for MySQL. MariaDB is nowadays most likely default database of many Linux distributions, (Bartholomew, MariaDB Cookbook, 2014). Both Red Hat Enterprise Linux (RHEL), (Red Hat, n.d.) and SUSE Linux Enterprise Server (SLES), (SUSE, 2016), have replaced MySQL with MariaDB.

The development of MariaDB is fully open and acceptable of contributions, it has both a public mailing list and a public bug tracker. Contribution is easy and as of April 2018, a total of 151 contributors have helped in the development of MariaDB, (MariaDB, 2018). Compared to MySQL which has 58 contributors, MariaBD has become more popular than its parent, (MySQL, 2018). Fedora, which is a Linux distribution developed by Red Hat, has also changed the default database installation from MySQL to MariaDB. Fedora (2013) writes:

Fedora will have a truly open-source MySQL implementation and won't depend on what Oracle decides to do with MySQL in the future. Compared to MySQL, MariaDB is faster in some cases, it includes some new engines and features, while the existing engines and features are enhanced and still compatible.
2.4. PostgreSQL

PostgreSQL, or Postgres, is another popular DBMS which is also completely open-source and free. It is not owned by any company or organization and is a product of community developers. Postgres is known for its reliability, data integrity and data correctness, (PostgreSQL, n.d.). Postgres has been in development since 1986 as part of the POSTGRES project at the university of California and has since then become the chosen DBMS of several well-known companies. Some of these are the U.S State Department, Debian, Apple, Red Hat and Cisco, (PostgreSQL, n.d.).

Postgres claims that PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions, (PostgreSQL, n.d.)

PostgreSQL works on several platforms which includes but are not limited to Linux, Unix and Windows.

2.5. Transactional Workload

A transactional workload regarding databases and DBMS is a workload that is identified by the database receiving and requests to access data and at the same time request to modify this data from several users over a period, this operation is called a transaction. A transaction can modify and manipulate the data that is stored within the database, but it can also roll back changes and commits to its starting point. Databases remain consistent by adhering to ACID (Atomicity, Consistency, Isolation, Durability), this is especially important when working with transactions since data is read and modified at the same time.

2.6. Previous Work

One previous study by Sasalak Tongkaw and Aumnat Tongkaw compared the database performance of MariaDB and MySQL with OLTP workload and found that MySQL vastly outperforms MariaDB in OLTP workloads, (Tongkaw & Tongkaw, 2016). The study used the benchmarking suite Sysbench with a modified workload suitable to OLTP workloads.

Another study by Mohiuddin Ahmed, Mohammad Moshee Uddin, Md. Saiful Azad and Shariq Haseeb which analyzed the general performance of MySQL running on a limited resource system found that given the same hardware, MySQL performed better on an Ubuntu server rather than a server running Fedora, (Ahmed, Uddin, Azad, & Haseeb, 2010).

However, none of the studies take in to account the hardware vulnerabilities Spectre and Meltdown which as explained above, will have some performance impact because of the mitigations against said vulnerabilities. Also, the previous work does not compare the performance between different DBMSs that does not originate from one and other. Since MariaDB is based on MySQL, Sysbench works on both DBMSs out of the box. When benchmarking two different DBMSs however, there is a need for a unified benchmarking tool that works on multiple platforms.

3. Problem Definition

This part describes the purpose of the study and the motivation of why it’s important.
3.1. **Purpose**

The purpose of this study is to evaluate the impact that the software mitigations against the two recently disclosed hardware vulnerabilities Spectre and Meltdown have on the performance in database management systems. As previously mentioned, these mitigations will have some reduced performance on these systems because of the frequent user-to-kernel switching. Depending on how big of an impact these mitigations have, companies that operate on a large scale will notice these degradations and potentially lose profit.

3.2. **Aim of the study**

The aim of the study is to find out how much impact the software mitigations against spectre and meltdown have on the performance of database management systems.

3.3. **Research question**

The research question that this study aims to answer is the following:

*How much of an impact does the software mitigations against Spectre and Meltdown have on performance in OLTP workloads?*

3.4. **Motivation**

With the ever-increasing mass of storage being produced and the need to access this vast amount of data, the need for quality-controlled DBMS arises. IBM claims that 90% of the world’s data have been created in the last two years, (IBM, n.d.). Access to stored data whether it is from an application to a user browsing an e-commerce site is crucial.

Earlier studies can be viewed as outdated since the meltdown and spectre vulnerabilities have an impact on all earlier studies which include any type of benchmarking. Because of the nature of these software mitigations there will be some impact in terms of decreased performance in certain type of applications, database management systems are one of them.

3.5. **Hypothesis**

The hypothesis is that by applying the software mitigations against Spectre and Meltdown, there will be an impact on the overall performance of the DBMS when applied with an OLTP workload.
3.6. Objectives

The study involves several objectives which are done in order to try and achieve the aim of the study and to answer the research questions.

1. Conduct a literature study to gain knowledge of the subject in order to make informed decisions that helps in the study.
2. Review previous work in the field.
3. Research validity threats in order to find both valid and non-valid validity threats.
4. Perform a pilot study in order to gain knowledge of various tools and to identify potential issues before conducting the quantitative study.
5. Develop a sound method to achieve the objectives of the study
6. Conduct the experiments and compare the results
7. Conduct an analysis of the results
4. Methodology
This chapter will explain the different strategies and methods required to complete the objectives of the study. Relevant validity threats will also be covered in this section.

4.1. Experimentation Process
According to (Wohlin, et al., 2012) experiments are used when there is a need to control subjects, objects and instrumentation. This ensures the ability to draw general conclusions and to perform statistical analysis. This makes it the perfect method for a benchmark where there is a need to both statistically analyse the results and to visualize the results.

4.2. Independent variables
According to (Wohlin, et al., 2012) an independent variable is a variable that can be manipulated and controlled. This study incorporates two independent variables, the Linux kernel and the hypervisor build.

4.3. Dependent variables
According to (Wohlin, et al., 2012) a dependent variable is the effect of altering the independent variables. In this study there are two different dependent variables, TPM and NOPM. TPM will be used to compare the results from the same DBMSs and NOPM will be used to compare different DBMSs with each other.

4.4. Scope
This study chose to use the following DBMSs to achieve the objectives of the study:
- MariaDB
- PostgreSQL

While there were a plethora of open source DBMSs, the two DBMSs was chosen because of the familiarity and each DBMSs history. MariaDB which is a derivate of the all known MySQL and PostgreSQL which is known in the database community as equally powerful and reliable.

Debian was chosen as the OS of choice because of familiarity and the fact that it is considered to be a stable production environment, (Sadowski, Sadowski-Rasters, & Duysters, 2008). Other popular Linux server distributions that were considered was Ubuntu and CentOS. However, Ubuntu is derived from Debian and is not as minimal as the default installation of Debian. CentOS was not chosen because of the familiarity with Debian.

4.5. HammerDB
The tool HammerDB will be used to benchmark the two DBMS. HammerDB supports both MariaDB and Postgres which makes it the ideal tool for benchmarking the performance of each DBMS before and after the system have been patched against spectre and meltdown, (HammerDB, 2017).

While traditional database benchmarking tools uses Transactions Per Minute (TPM), HammerDB uses New Orders Per Minute (NOPM). This is to correctly evaluate the performance across different
DBMS. Since each DBMS reports TPM differently there is no consistency when benchmarking different DBMS, (HammerDB, 2017). However, TPM can be used to compare the results from the same DBMS.

When benchmarking OLTP workloads with HammerDB there is the option to choose between two separate tests, TPC-C and TPC-E. TPC-C and TPC-E are both benchmarks created by the Storage Performance Council (SPC). The TPC-C benchmark is centred around an order-entry environment which includes entering and delivering orders, recording payments, checking the status of orders and monitoring the level of stock at the warehouse, (TPC, n.d.). The TPC-E benchmark is a new variant of the original TPC-C, but which have been adapted to simulate a brokerage firm with customers who generate transactions related to trades, account inquiries and market research, (TPC, n.d.). The TPC-C benchmark essentially simulates a company taking orders and handling transactions at multiple warehouses with multiple concurrent users.

4.6. **Experiment**

An experiment was developed and carried out in a lab using two computers, one computer acting as a client for configuring the server environment and accessing the hypervisors web interface, and another computer which is running the hypervisor. The hypervisor is using version 6.5 of ESXi and is hosting the two virtual machines necessary for this experiment. The first virtual machine is the Debian-MariaDB combo, and the second virtual machine is the Debian-PostgreSQL combo.

A pilot test was performed to figure out the correct parameters required for this study and the hardware resources available, explained in section 4.8.

The client will be using the benchmarking software HammerDB with the setting of 1 virtual user and 1 million transactions for each test where each test lasts 5 minutes and will be executed 10 times. HammerDB was also configured to use 10 warehouses to distribute the load against these warehouses.

Both the machines were configured with the same hardware specifications:

- Intel Xeon W3550 @ 3.07GHz CPU 8 cores and 8 threads
- 24 GB DDR3 Random Access Memory (RAM)
- Western Digital 500GB 5400 rpm

The experiment will be using two different Linux kernels and two different ESXi build numbers. Since the mitigations against Spectre and Meltdown is distributed via the Linux kernel and the ESXi builds, altering these two parameters made it easy to switch between a protected and a non-protected state against Spectre and Meltdown.

In total eight experiments was conducted:

1. MariaDB – Hypervisor unpatched
   1.1. MariaDB using kernel 4.9.0.3 which lacks any mitigation against Spectre and Meltdown
   1.2. MariaDB using kernel 4.9.0.6 which uses KPTI
2. MariaDB – Hypervisor patched
   2.1. MariaDB using kernel 4.9.0.3 which lacks any mitigation against Spectre and Meltdown
2.2. MariaDB using kernel 4.9.0.6 which uses KPTI

3. PostgreSQL – Hypervisor unpatched
   3.1. PostgreSQL using kernel 4.9.0.3 which lacks any mitigation against Spectre and Meltdown
   3.2. PostgreSQL using kernel 4.9.0.6 which uses KPTI

4. PostgreSQL – Hypervisor patched
   4.1. PostgreSQL using kernel 4.9.0.3 which lacks any mitigation against Spectre and Meltdown
   4.2. PostgreSQL using kernel 4.9.0.6 which uses KPTI

4.7. HammerDB Configuration
HammerDB will be configured with the TPC-C setting for each DBMS. HammerDB will also be configured to use 10 warehouses. The number 10 were chosen via testing and consulting with the lead developer of HammerDB. Each test will be executing a total of 1 million transactions per virtual user for five minutes. Each test will produce a TPM value and a NOPM value. Each test will run 10 times in order to ensure that the average TPM and NOPM is accurately portrayed.

4.8. Pilot test
Initial testing was made to ensure the validity of the experiments and the correctness of the idea itself. In order to conduct the pilot study, all the software needed for the study was downloaded and installed on a test system. A Debian machine was virtualized using the hypervisor VMware ESXi and the kernel was downgraded to 4.9.0.3 in order to use a kernel without KPTI implemented. The system was assigned 4 cores with hyperthreading activated and 12 Gigabytes of RAM. On the client machine, the benchmarking tool HammerDB was installed which is going to be used to stress the DBMS.

HammerDB was set up to stress MariaDB and PostgreSQL with a fixed number of transactions, namely 1 million transactions. HammerDB was configured to use a total of 10 warehouses. During the pilot study the tests identified an issue with the default configuration of MariaDB. MariaDB in default settings are set only listen on the local interface or 127.0.0.1 which would have to be changed in order to make remote connections possible. There was also the issue with the max connections settings with MariaDB. MariaDB uses the default setting of 151 which is too low for the number of virtual users that are going to be used, (MariaDB, 2018). This setting was bumped up to 700.

The pilot study showed that the CPU IOwait metric spiked whenever a test was run on the system. An effort was made to reduce this metric by using a RAM-disk. This was configured to store the temporary files of each DBMS in order to reduce writing to disk. However, this proved to have no effect on the results of the tests. After contacting the developer of HammerDB about this issue it was made clear that the culprit of the high IOwait metric was the result of a too slow hard drive for the number of virtual users and warehouses configured. After testing it was found that the optimal settings for the test system was to use 10 warehouses, 1 virtual user and 1 million transactions for each test. The tests will be performed 10 times lasting 5 minutes each. These parameters were chosen because there was no effect on the performance of either DBMS with longer test times or more tests.
4.9. Ethics

All the tests conducted by this study has no intention to harm or affect the copyright holder of each software in question. The software chosen for this study were all free, open-source and uncommercial. This study does not experiment using people, instead it only uses software.

4.10. Validity threats

The study does not consider the overhead produced by patching the CPU via microcode since that is out of the scope of the aim of the study. The study also acknowledges the network topology as a validity threat since the two systems involved are both connected to the same switch.

The study also does not try to optimize any of the two DBMSs, each DBMSs will be tested with the default configuration that they have when installed via the repository. If, however there is a need to change any configuration options of either DBMS the study will point it out in the relevant chapter.

Other validity threats include, (Wohlin, et al., 2012):

Reliability of measures – Whenever an experiment is conducted there is a chance that the results are not trustworthy which was handled by conducting many tests and by automating the testing itself.

Random irrelevancies in experimental setting – There is a chance that elements outside the experiment may disturb the results such as the network being used by other machines. This was handled by separating the two machines in the experiment and having them use the same switch and only them.

Reliability of treatment implementation – There is a chance that the implementation is not similar between different tests which was handled by automating the testing and using the same variables for each subject.
5. Implementation

This chapter will explain each configuration made to implement the experiment developed in order to achieve the objectives of the study.

5.1. Debian

Debian was installed with the minimal installation, acting solely as a server. The only other configuration made was to install an older kernel than the default one. When installed, Debian comes with the latest kernel as of this writing, 4.9.0.6. This kernel has implemented the mitigations against Spectre and Meltdown and was verified using a shell script developed by Stéphane Lesimple, (Lesimple, 2018).

The kernel was downgraded to kernel version 4.9.0.3 which lacks any mitigation against Spectre and Meltdown, also verified by the shell script developed by Stéphane Lesimple, (Lesimple, 2018).

5.2. MariaDB

MariaDB was installed on the server via the built-in tool apt which is the standard package manager for Debian distributions, (Debian, 2016). The version available in the repository was 10.1.26 and was installed via the package: mariadb-server.

The only configuration made to default implementation of MariaDB was to change the default password of the root user, increment the total number of allowed connections to MariaDB and allowing remote connections to MariaDB.

5.3. PostgreSQL

PostgreSQL was installed on the server using the same built-in tool apt which is the standard package manager for Debian distributions, (Debian, 2016). The current version available in the repository and used in this experiment was 9.6 and was installed via the package: postgresql-9.6.

In contrast to the configuration of MariaDB, a few more changes were needed for PostgreSQL. PostgreSQL. The default listen-address was changed from localhost to its private IP address to allow remote connections. Along with that, PostgreSQL uses a file called “pg_hba.conf” as a method for client authentication, (Postgres, n.d.). The file was altered to allow remote connections from the client machine.

5.4. HammerDB

HammerDB was installed and configured on the client machine which was running Ubuntu 16.04. The current version of HammerDB and the one used in this study are HammerDB 3.0. HammerDB uses the same configuration for each DBMS which is the following:

- 10 warehouses
- 1 virtual user
- 1 million transactions
- 5-minute duration per test
- 10 tests in total
The results of each test were output to a file called “hammerdb.log” located in “/tmp” of the client machine. This file contains the TPM and NOPM value which will be used to compare and analyse the results.

Before HammerDB can conduct any testing at all, it must create the schema used in the benchmark. The schema is comprised of nine types of tables with a wide range of population and records that involves a mix of five different concurrent transactions with varying complexity. HammerDB will distribute the load against the available 10 warehouses to simulate distributed workloads. The schema is similar to the official TPC-C benchmark from the Storage performance council, (Google, 2018).
6. Results

In this section the result of the experiment will be presented. A system that is not in any way patched against Spectre and Meltdown is using the alias “pre” whereas the system that have been patched and have the mitigations against Spectre and Meltdown goes by the alias “post”.

6.1. Transactions Per Minute - MariaDB

This section will be used to compare MariaDB with itself before and after any mitigation against Spectre and Meltdown is applied.

Figure 1 represents the result from the total of 10 tests when the operating system does not incorporate any of the mitigations against Spectre and Meltdown, pre, and when the operating system has incorporated the mitigations against Spectre and Meltdown, post.

![Figure 1 MariaDB TPM pre- and post-mitigation](chart)

Figure 1 shows a substantial decrease in terms of TPM when the OS have been patched against Spectre and Meltdown compared to when the OS does not have any mitigation against Spectre and Meltdown.

Figure 2 shows the average TPM pre- and post-mitigation against Spectre and Meltdown when the hypervisor is unpatched against the vulnerabilities. It also shows the standard deviation which are represented as error bars on each column. The chart shows that there is a decreased value of about 5.4 % when the OS has been patched against Spectre and Meltdown. The chart also shows that the standard deviation is higher after the OS have been patched against Spectre and Meltdown. The standard deviation pre-mitigation is around 0.84% where as post-mitigation the standard deviation is around 1.76%.
6.2. Transactions Per Minute – MariaDB with updated hypervisor

Figure 3 represents the result from the total of 10 tests when the operating system does not incorporate any of the mitigations against Spectre and Meltdown, pre, and when the operating system has incorporated the mitigations against Spectre and Meltdown, post.

Figure 3 shows a decrease in terms of TPM when the OS have been patched against Spectre and Meltdown compared to when the OS does not have any mitigation against Spectre and Meltdown. The values have decreased further when compared to figure 1.

Figure 4 shows the average TPM pre- and post-mitigation against Spectre and Meltdown when the hypervisor is patched against the vulnerabilities. The standard deviation is represented as error bars on each column. The chart shows the there is a decreased value of about 5.6% after the OS have
been patched against Spectre and Meltdown. The standard deviation pre-mitigation is around 0.85% whereas post-mitigation the standard deviation is around 1.84%.

**Figure 4 MariaDB average TPM pre- and post-mitigation with updated hypervisor**

### 6.3 Transactions Per Minute – MariaDB summary

This section will compare the results above with each other. Figure 5 shows the average TPM of pre- and post-mitigation both before and after the hypervisor has been patched.

**Figure 5 MariaDB summarization of TPM both before and after both OS and hypervisor is patched**

The chart shows that the average TPM have been steadily declining for each test where the first test which does not incorporate any mitigation against Spectre and Meltdown on any level performs the best. The last test which incorporates full mitigation against Spectre and Meltdown via both OS patches and an updated hypervisor performs the worst. The standard deviation is also increasing for each test.
6.4. Transactions Per Minute - PostgreSQL

This section will be used to compare PostgreSQL with itself before and after any mitigation against Spectre and Meltdown is applied.

Figure 6 represents the result from the total of 10 tests when the operating system does not incorporate any of the mitigations against Spectre and Meltdown, pre, and when the operating system has incorporated the mitigations against Spectre and Meltdown, post.

![Figure 6 PostgreSQL TPM pre- and post-mitigation](image)

Figure 6 does not show a significant change pre- and post-mitigation against Spectre and Meltdown. The results are also more stable when compared to MariaDB.

Figure 7 shows the average TPM pre- and post-mitigation against Spectre and Meltdown when the hypervisor is unpatched against the vulnerabilities. It also shows the standard deviation which are represented as error bars on each column. The chart shows that there is a decreased value of about 5.4% when the OS has been patched against Spectre and Meltdown. The chart also shows that the standard deviation is higher after the OS have been patched against Spectre and Meltdown. The standard deviation pre-mitigation is around 0.84% whereas post-mitigation the standard deviation is around 1.76%.
6.5. Transactions Per Minute – PostgreSQL with updated hypervisor

Figure 8 represents the result from the total of 10 tests when the operating system does not incorporate any of the mitigations against Spectre and Meltdown, pre, and when the operating system has incorporated the mitigations against Spectre and Meltdown, post.

Figure 8 shows no significant change pre- and post-mitigation against Spectre and Meltdown.

Figure 9 shows the average TPM pre- and post-mitigation against Spectre and Meltdown when the hypervisor is patched against the vulnerabilities. It also shows the standard deviation which are represented as error bars on each column. The chart shows that there is a decreased value of about 1.48 % when the OS has been patched against Spectre and Meltdown. The chart also shows that the standard deviation is higher after the OS have been patched against Spectre and Meltdown. The
standard deviation pre-mitigation is around 1.19% whereas post-mitigation the standard deviation is around 3.8%.

**Figure 9** PostgreSQL average TPM pre- and post-mitigation with updated hypervisor

### 6.6. Transactions Per Minute – PostgreSQL summary

This section will compare the results above with each other. Figure 10 shows the average TPM of pre- and post-mitigation both before and after the hypervisor has been patched.

**Figure 10** PostgreSQL summarization of TPM both before and after both OS and hypervisor is patched

The chart shows that the average TPM have been steadily declining for each test where the first test which does not incorporate any mitigation against Spectre and Meltdown on any level performs the best. The last test which incorporates full mitigation against Spectre and Meltdown via both OS patches and an updated hypervisor performs the worst.
6.7. New Orders Per Minute

This section will be used to compare DBMSs with each other with an unpatched hypervisor.

Figure 11 and 12 represents the result from the total of 10 tests when the operating system does not incorporate any of the mitigations against Spectre and Meltdown, pre, and when the operating system has incorporated the mitigations against Spectre and Meltdown, post.

The charts show that MariaDB has more fluctuation compared to PostgreSQL, which also has higher values than MariaDB.
6.8. New Orders Per Minute – Patched hypervisor

This section will be used to compare DBMSs with each other when the hypervisor has been patched against Spectre and Meltdown.

Figure 13 and 14 represents the result from the total of 10 tests when the operating system does not incorporate any of the mitigations against Spectre and Meltdown, pre, and when the operating system has incorporated the mitigations against Spectre and Meltdown, post.

Figure 13 MariaDB NOPM pre- and post-mitigation with updated hypervisor

Figure 14 PostgreSQL NOPM pre- and post-mitigation with updated hypervisor

Figure 13 and 14 show the NOPM metric compared between MariaDB and PostgreSQL after the hypervisor has been patched against Spectre and Meltdown. The fourth test in figure 13 shows an anomaly where the post-NOPM metric is far higher than the pre-NOPM metric.
6.9. *New Orders Per Minute – Comparing results*

This section will be used to compare the DBMSs with each other.

Figure 15 shows the average NOPM value for each test. PostgreSQL with no mitigation at all is performing better than all other results at around 7708 NOPM while MariaDB with no mitigation is slightly under at around 7562 NOPM. As soon as a mitigation is applied to either the OS, the hypervisor and/or both, the NOPM metric drops.

*Figure 15 Comparing MariaDB and PostgreSQL*
7. Analysis

Looking at the charts there is a clear pattern that whenever an element of mitigation against Spectre and Meltdown is added, the performance of the DBMSs drops. However, with an average of 4-5% drop in performance, the performance degradation does not seem substantial. MariaDB seem to have more of a performance impact after the mitigation has been applied than PostgreSQL.

The performance impact is incremental with each layer of protection against said vulnerabilities. The biggest performance impact is when the OS is using kernel 4.0.9.6 which incorporates KPTI. The average performance impact by just applying the OS mitigation against Spectre and Meltdown is around 4-5%. This percentage correlates with the results published by Red Hat which found that the average performance impact of applying the mitigations against Spectre and Meltdown was around 4-8%, (Red Hat, 2018).

PostgreSQL performed better with an OLTP workload compared to MariaDB, which is evident when looking at the NOPM values produced by both DBMSs. Figure 15 breaks down each test and how they affect the NOPM metric and PostgreSQL performs better in every test compared to MariaDB.

While PostgreSQL is performing better than MariaDB it also has a higher standard deviance which is unknown as to why this is. PostgreSQL also have a very high-performance drop when the system is completely patched against the vulnerabilities via both OS and hypervisor updates. Figure 10 shows a drop of 27% when comparing the first and fourth column.
8. Conclusion

The aim of this study was to find out how much of an impact the software mitigations against the vulnerabilities Spectre and Meltdown had on DBMSs. The study conducted eight experiments to figure out just how much of an impact each layer of protection added impacted the performance of said DBMSs.

The results of the experiments show that by applying the software mitigations against Spectre and Meltdown there will be some performance impact. The results showed that when both the hypervisor and the OS is completely unprotected against Spectre and Meltdown it also performs the best. When the OS is patched against said vulnerabilities, the performance drops by 4-5%. The performance of these DBMSs drops even further when the hypervisor has been updated with protection against the vulnerabilities.

PostgreSQL seem to be especially affected by the mitigations as shown by figure 10. Figure 10 shows that there is a 27% drop of TPM when comparing a completely unpatched PostgreSQL instance to a completely patched PostgreSQL instance. While PostgreSQL outperformed MariaDB when compared in figure 15, PostgreSQL also seem to be affected the most.

The hypothesis for this study was that by applying the software mitigations against Spectre and Meltdown, the overall performance of a DBMS under OLTP workload will decline. The results of this study validate the hypothesis.
9. Discussion

The study shows that by applying the software mitigations against Spectre and Meltdown on both the OS and/or the hypervisor, there will be a performance impact. The experiment would have benefitted of better hardware resources, especially in the I/O which would have allowed to increase the number of virtual users during each test. The study did not take Windows or Unix into account which makes this relevant for Linux systems only.

As both Meltdown and Spectre is relatively new, there has been some issues to find information and previous studies. The vulnerabilities themselves are very technical and therefore have been proven very hard to fully grasp their impact on modern systems and its routine.

There is also the possibility that other changes in the kernel when using version 4.9.0.3 and 4.9.0.6 could impact the results. But seeing as 4.9.0.6 is the version available that has full mitigations against Spectre and Meltdown, it was the only choice, thus skipping kernel 4.9.0.4-4.9.0.5.

With an impact of 4-5% decreased performance when the mitigations are applied to the system, the average user will probably not notice this degradation at all. However, companies hosting cloud services where there are hundreds and thousands of users using their hardware at the same time, 4-5% will probably have a noticeable impact and can even impact their profit.
10. Future Work

In the future the experiment would benefit by having better I/O performance to increase the number of users to test against each DBMS. The experiment could benefit by adding more operating systems to compare cross-platform.

There is also a plethora of other DBMSs available for both Linux, Unix and Windows which might also be affected to varying degrees. It would be interesting to benchmark other DBMS before and after the mitigations against Spectre and Meltdown is applied to a system and compare the results to the results of this study.

There is also the possibility to benchmark each individual mitigation against a system to see which affects the system the most and compare the performance impact to the security advancements.
References


MariaDB. (n.d.). *About MariaDB*. Retrieved April 17, 2018, from mariadb: https://mariadb.org/about/


