http://dx.doi.org/10.21172/ijiet.121.01

Profiling The Database Queries Executed On The Hdd And Ssd Disks Using Performance Schema

Nikola Davidovic[1],Borivoje Milosevic[2], Slobodan Obradovic[3]
[1,3]*University of East Sarajevo,Faculty of Electrical Engineering,Republic of Srpska, Bosnia and Hercegovina*
[2]*University UNION - Nikola Tesla,Faculty of Business and Law, Belgrade, Serbia*

Abstract- This paper deals with analysis and testing of the characteristics of HDD and SSD disks when the same database is organized on them. The database response analysis will be performed via the WAMP server on which the MySql server is mounted. All tests will be performed at WWW localhostenvironment, which implies performing a series of transactions and measuring base response located at different disk locations. By comparing these times, it is possible to determine which of these two disks has better characteristics and which of them should be used to gain better performance when executing database transactions.Flash Solid State Disks induce a drastic changein storage technology that impacts database systems. Flashmemories exhibit low latency (especially for small block sizes),very high random read and low random write throughput, andsignificant asymmetry between the read and write performance.These properties influence the performance of database joinalgorithms and ultimately the cost assumptions in the queryoptimizer.Although it is still commonly assumedthat HDDs have higher sequential throughput, the sequential readsare fast on an HDD as well. Also, sort-merge join has better relative performance on HDDs. In this paper we will try to prove these claims.
Key words: Database, Query, Execution time, WAMP, PhpMyAdmin, MySQL server, Profiling solutions.

I. INTRODUCTION

Testing the database characteristics in a distributed environment cannot provide valid information about how the database behaves and responds to all types of transactions. Since this article is about comparing the database characteristics when the same database islocated on the HDD and the SSD disk, the effect of a distributed environment reflected in the data flow rate between the master and slave nodes, the database partitioning model, and replication must be avoided. That is why the databases are executed on a server organized at the local host and phpMyAdmin.

The database is imported through the WAMP server and PhpMyAdminenvironment and placed on the MySQL server. The design of the database, its tables, indexes, views, procedures and functions will be discussedin the paper, because we think that this is something which needs to be known in advancein this area of research.Importing thesame database to a MySQL server in a WAMP environment via PhpMyAdmin is done by defining the database path in the my.ini file script that represents the configuration file of the MySQL server. We will choose:

a) When the base is on the HDD discdatadir = "c: /wamp/bin/mysql/mysql5.7.21/data"

b) When the base is on the SSD diskdatadir = "e: / DATAssd"

In this case, we will test the same twodatabases, one installed on the HDD and the other on the SSD disk. We will measure the characteristics of the first and the second base, executing a whole series of queries and transactions, and therefore determine their response and the speed of execution of each process.The base has 16 tables, 3 functions, 3 procedures, and 7 views. The base is 29176 tupleswith all the tables, and each table has its own number of columns. Size is 3.1MB.

II. BASIC SETTINGS

The database is organized within the WAMP server and phpMyAdmin environment and is mounted on the HDD disk and SSD disk. The MySql and Apache serversare configured [1,2,3].

The characteristics of the performance scheme are obtained by query:

SHOWVARIABLESLIKE'performance_schema'

+ Options

Variable_name	Value
performance_schema	ON

SELECT * FROM INFORMATION_SCHEMA.ENGINESWHEREENGINE='PERFORMANCE_SCHEMA'

+ Options

ENGINE	SUPPORT	COMMENT	TRANSACTIONS	XA	SAVEPOINTS
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

```
USEperformance_schema
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = 'performance_schema';
```

To start running profile events on PhpMyAdminexecuted over the database, in first case the database on the HDD disk, and then the same one on the SSD disk, we must do the following:

1. Limit the collection of historical events to a user who initiates a query or any transaction. By default, the setup_actors table is configured to track the entire history collection that takes place in the foreground. We will confirm that the performance_schema option is configured by the setup_actors table:

```
mysql> SELECT * FROM performance_schema.setup_actors;
```

+ Options

HOST	USER	ROLE	ENABLED	HISTORY
%	%	%	YES	YES

2. Update the default order in the setup_actors table to reset collecting and tracking past history events for all of the top issues and inserting a new line that now enables new tracking and collection of historical events for a user to initiate a query or make a transaction over a database:

```
mysql> UPDATE performance_schema.setup_actors
SET ENABLED = 'NO', HISTORY = 'NO'
WHERE HOST = '%' AND USER = '%';
mysql> INSERT INTO performance_schema.setup_actors
(HOST,USER,ROLE,ENABLED,HISTORY)
VALUES('localhost','root','%','YES','YES');
```

The data in the setup_actors tables will now look like this:

```
mysql> SELECT * FROM performance_schema.setup_actors;
```

+ Options

HOST	USER	ROLE	ENABLED	HISTORY
%	%	%	NO	NO
localhost	root	%	YES	YES

We will check the sql script commands and update the setup_instruments table if the stage of instrumentalization of our requests is enabled. Some instruments may already have been set by default, so we can test them:

```
mysql> UPDATE performance_schema.setup_instruments
SET ENABLED = 'YES', TIMED = 'YES'
WHERE NAME LIKE '%statement/%';
mysql> UPDATE performance_schema.setup_instruments
SET ENABLED = 'YES', TIMED = 'YES'
WHERE NAME LIKE '%stage/%';
```

We will check with sql script commands whether events_statements_ * and events_stages_ * are enabled:

```
mysql> UPDATE performance_schema.setup_consumers
SET ENABLED = 'YES'
WHERE NAME LIKE '%events_statements_%';
mysql> UPDATE performance_schema.setup_consumers
SET ENABLED = 'YES'
WHERE NAME LIKE '%events_stages_%';
```

To keep track of the required process within our user access, execute the sql command and impersonate the timing events. For example, when the databaseon the HDD,prompted for a query –show, and all the tags from the actor table are foundwhen actor_id = 1, we get:
mysql> SELECT * FROM table1 WHERE table_name = No;
We will identify EVENT_ID event by query in the events_statements_history_long table. This step is similar to launching the SHOW PROFILES option to identify Query_ID. The following query output is similar to SHOW PROFILES:
mysql>SELECT EVENT_ID, TRUNCATE(TIMER_WAIT/1000000000000,6) as Duration, SQL_TEXT FROM performance_schema.events_statements_history_long WHERE SQL_TEXT like '% table_name %'
Inquiry into the events_stages_history_long table, we take over all the defined event elements. The phases are related to statements by means of latent events. Each stage event has a NESTING_EVENT_ID row that contains EVENT_ID superordinate parent statements.
mysql> SELECT event_name AS Stage, TRUNCATE(TIMER_WAIT/1000000000000,6) AS Duration FROM performance_schema.events_stages_history_long WHERE NESTING_EVENT_ID=179;

Finally, we get a description of the time line and operation performed:

```
+--------------------------------+----------+
| Stage                          | Duration |
+--------------------------------+----------+
| stage/sql/starting             | 0.000080 |
| stage/sql/checking permissions | 0.000005 |
| stage/sql/Opening tables       | 0.027759 |
| stage/sql/init                 | 0.000052 |
| stage/sql/System lock          | 0.000009 |
| stage/sql/optimizing           | 0.000006 |
| stage/sql/statistics           | 0.000082 |
| stage/sql/preparing            | 0.000008 |
| stage/sql/executing            | 0.000000 |
| stage/sql/Sending data         | 0.000017 |
| stage/sql/end                  | 0.000001 |
| stage/sql/query end            | 0.000004 |
| stage/sql/closing tables       | 0.000006 |
| stage/sql/freeing items        | 0.000272 |
| stage/sql/cleaning up          | 0.000001 |
+--------------------------------+----------+
```

III. ANALYSIS

The database is on a single server, the WAMP server represents the MySqlWeb serveron which the complete databasesare located.
What we are testing and what we are interested in is the time of execution of the query, i.e. database response. Hypothetically, the database response for a different input of the code must be different because the number of alphanumeric data for each predefined table is not the same, so the response time of the database needs to be somewhat different [4,5,6,7]. The experimental method confirms this.
Reading complete tables of the databases located on HDD and SSD disks – SELECT statement
The following example shows how to run events and their query phases and database transactions using the Performance Schema to get information about the process that is executed using SHOW PROFILES options. The setup_actors table can be used for limitting historical collections by host, user, or account.The schema displayed peak event data (trillion seconds) to normalize data time to the standard unit. In the following example, TIMER_WAIT is divided by 1 trillion to display data in units of seconds. The values are also shortened to 6 decimal places to display data in the same format as SHOW PROFILES and SHOW PROFILE statements.

SELECT * FROM performance_schema.performance_timers;

+ Options

TIMER_NAME	TIMER_FREQUENCY	TIMER_RESOLUTION	TIMER_OVERHEAD
CYCLE	2706904370	1	69
NANOSECOND	*NULL*	*NULL*	*NULL*
MICROSECOND	2648931	1	94
MILLISECOND	988	1	96
TICK	1667	1	17

SELECT * FROM `table1…or…tableN`;
SELECT EVENT_ID, TRUNCATE(TIMER_WAIT/1000000000000,6) as Duration, SQL_TEXT
FROM performance_schema.events_statements_history_long WHERE SQL_TEXT like '%table 1%';

Database response times after executing this series of SELECT query are given in the Figure1:

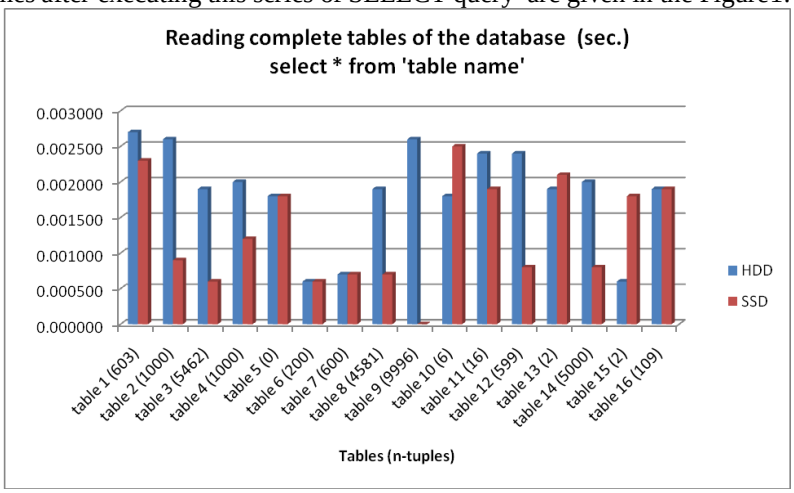

Fig. 1 SELECT table

It is easy to conclude that reading the complete database tables of the SSD disk showed better characteristics, even in cases where database tables are very small.
Executing VIEWS of the databases located on HDD and SSD disks
Profiling the views and responses of the database is accomplished by realizing the following script. In doing so, the views of the database include a different number of tables, Figure 2.

SELECT * FROM view 1...view 7 LIMIT10
SELECT EVENT_ID, TRUNCATE(TIMER_WAIT/1000000000000,6) as Duration, SQL_TEXT FROMperformance_schema.events_statements_history_longWHERE SQL_TEXT like'% view 1...view 7%'

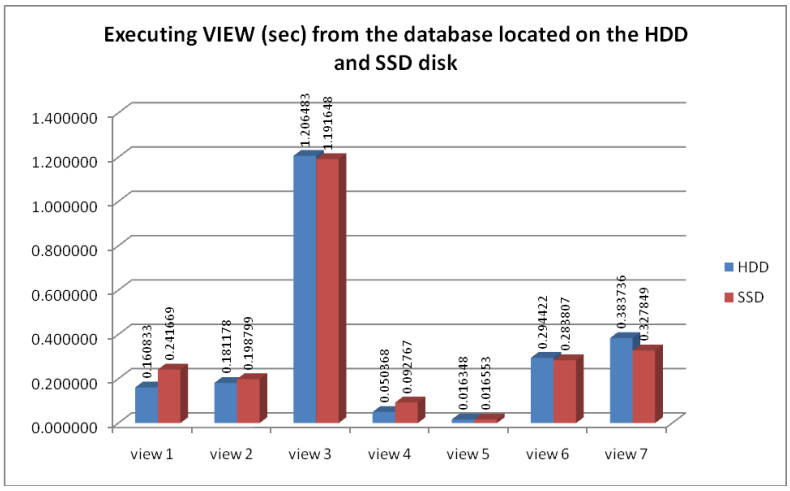

Fig. 2Executing VIEW

As can be seen from the figure, executing a different view of the databases and response time does not differ much on HDD and SSD disks.This is because the database view includes several tables of different size, so their view execution time is very comparative.

Update records of the databases located on HDD and SSD disks
Profiling the UPDATE statementand responses of the database is accomplished by realizing the following script. In doing so, the views of the database include a different number of tables tuples, Figure 3.

```
UPDATE table 1…table N
SET personal_id = number1
WHERE    iznajmljeni_id = number2;
```

SELECT EVENT_ID, TRUNCATE(TIMER_WAIT/1000000000000,6) as Duration, SQL_TEXT FROM performance_schema.events_statements_history_long WHERE SQL_TEXT like '%number2%'

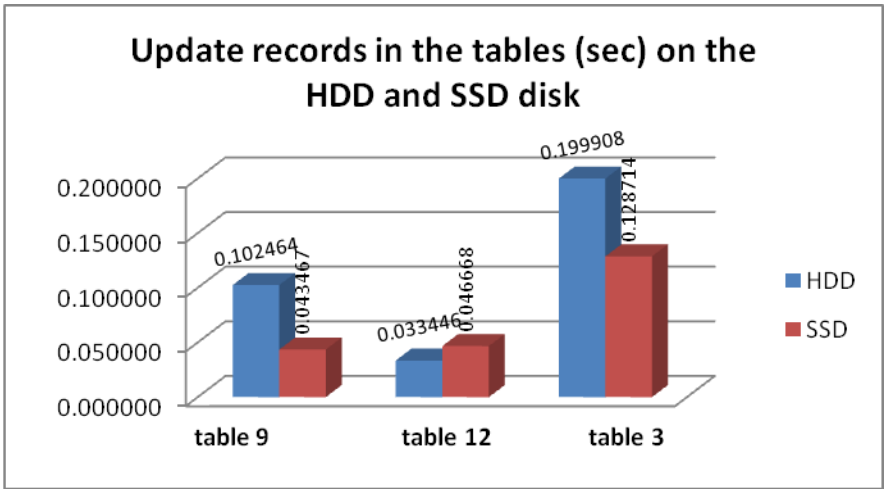

Fig. 3Executing UPDATE

As it can be seen from the figure, executing a different UPDATE statement on the database tables and response time is much betteron SSD disk except forvery small tables.

Write in the database tables located on HDD and SSD disks
Profiling the INSERT statementand responses of the database is accomplished by realizing the following script. In doing so, the views of the database include a different number of tables tuples, Figure 4.

```
INSERTINTOtable N(attribute1…….attributeN) VALUES('values1….valuesN')
```

SELECT EVENT_ID, TRUNCATE(TIMER_WAIT/1000000000000,6) as Duration, SQL_TEXT FROM performance_schema.events_statements_history_long WHERE SQL_TEXT like '%table N%'

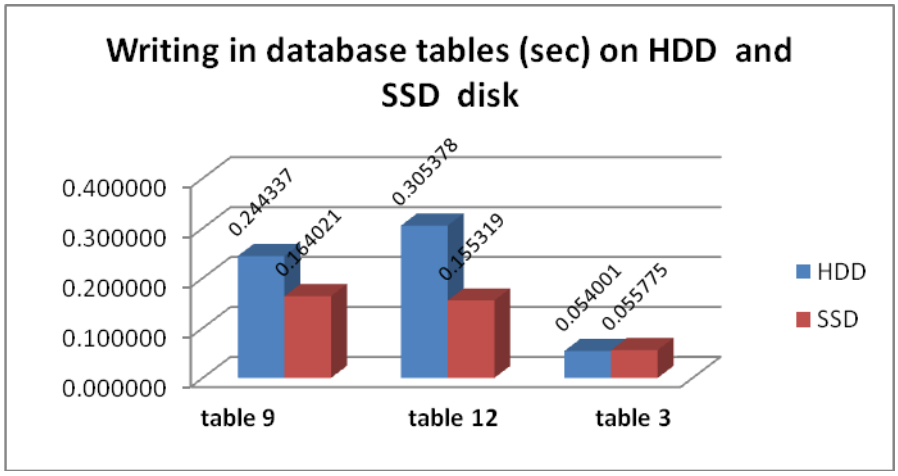

Fig. 4Executing WRITING statement

As it can be seen from the figure, executing a different INSERT statement on the database tables and response time is much betteron a SSDdisk.

Delete the database rows on the tables located on HDD and SSD disks
Profiling the DELETE statementand response of the database is accomplished by realizing the following script. In doing so, the views of the database include a different number of tables tuples, Figure 5.

```
DELETE FROM table N
WHERE  tableN_id = idnumber;
```

```
SELECT EVENT_ID, TRUNCATE(TIMER_WAIT/1000000000000,6) as Duration, SQL_TEXT FROM performance_schema.events_statements_history_long WHERE SQL_TEXT like '%idnumber%'
```

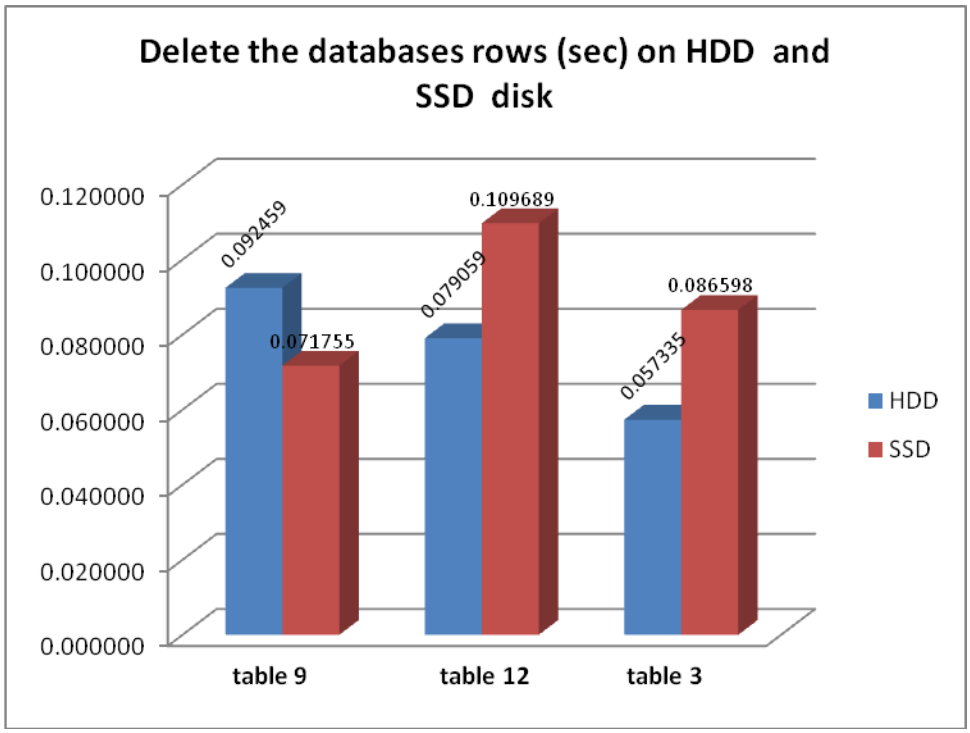

Fig. 5Executing DELETE statement

As it can be seen from the figure, executing a different DELETE statement of the database tables and response time is much betteron a SSD disk when it works with large tables.

IV. CONCLUSIONS

Based on the previous results we can conclude that:
Reading the complete database tables of the SSD disk showed better characteristics, even in cases where database tables are very small.Executing a different UPDATE statement on the database tables and response time is much better on SSD disk except forvery small tables.Executing a different INSERT statement on the database tables and response time is much better on a SSD disk.Executing a different DELETE statement on the database tables and response time is much better on a SSD disk when it works with large tables.
There is a widely spread myth that fragmentation does not matter in the case of SSDs because the access to all cells is equally fast. Even though this statement is partially true, there are more factors to account for, aside from the read latency. In short, the fragmentation of table indexes causes more reads than for indexes which are not fragmented, and also the fragmentation diminishes the available disk capacity.Of course, even for SSDs there are differences between sequential / random reads and writes (the sector and block sizes matter as well as the Queue Depth settings). The SSDs reads outperform the HDDs, but the performance of the SSDs significantly diminishes in the case of writes and, specifically, updates.

Fragmentation and wear cause lower capacity, but we have faster speed than spindle disk. As we mentioned above, fragmentation means more IOPS to read the same amount of data and also means that less cells are available for use, i.e. larger fragmentation, less capacityof the drive that can be used. Also, frequent updates increase the 'wear-and-tear' of the drive, which implies that as cells get exhausted and become unusable, the drive loses capacity.

It is true that SSDs bring great benefit to our database systems by providing faster data access and by diminishing the latencies to minimum. But the question still remains, how much data can the other components handle? If we have unlimited access to data, then we actually need to have unlimited access to RAM and to CPU to process the data further. But, random writes exhibit low performance, which also degrades over time.Therefore, to achieve balanced performance, random writes should be avoided atthe cost of random reads.

We have mentioned the allocation units for SSDs and for HDDs so it is clear that the more data updates we have, the greater performance problem SSDs face. This is because when data is updated on an SSD, the entire block has to be moved to a new block together with the update and then the old block has to be erased. This wears the drive significantly, and slows down performance.SSDs are great for static data, however. If the data is mostly read and not changed so often, then the SSDs are the best way to go.

A significant difference between SSDs and HDDs is that, when HDD fails, there are many ways to extract most of the data. Depending on what is damaged, a replacement of the head may help, or maybe just some of the platters can be recovered, etc. In the case of SSDs, it is almost impossible to extract any data if the drive stops working. Because of the complex algorithms used for wear leveling and because of the specifics of the NAND technology, it is almost impossible to recover data once the SSD fails in service.The best state for the SSD is to be in use. When the SSD is powered up, it has the best chance of keeping its data intact – it maintains the data in a consistent state. If an SSD drive is not powered up for a significant period of time (the shelf life of data on SSD is about 7 years) then some data loss is to be expected.

V. REFERENCES

[1] Vladimir Saso, SrdjanJovkovic, Borivoje Milosevic, Nikola Davidovic, Analysis Of The Disrtributed Databases In Different Environment, International JOURNAL of Innovations in Engineering and Technology (IJIET) http://dx.doi.org/10.21172/ijiet.102.04, Volume 10 Issue 2 May 2018 , ISSN: 2319-1058, Thomson Reuters Reseacher ID=D-6658-2018, pages 022-028, INDIAJeffrey,

[2] Kifer,M, Bernstein, A, Lewis, P.M. (2004): Database, Systems, Pearson, Addison Wesley

[3] Weikum,G, Vossen, G. (2001): Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery, Morgan Kaufmann, ISBN 1558605088

[4] Daniel Bausch, Ilia Petrov, Alejandro Buchmann, On The Performance Of Database Query Processing AlgorithmsOn Flash Solid State Disks, Conference Paper · August 2011, DOI: 10.1109/DEXA.2011.60 · Source: DBLP, publication at: https://www.researchgate.net/publication/221646194

[5] MySQL Workbench, https://downloads.mysql.com /docs/workbench-en.a4.pdf

[6] "About the Apache HTTP Server Project". Apache Software Foundation. Archived from the original on 7 June 2008. Retrieved 2008-06-25.

[7] Dave Stokes, MySQL Workbench, http://www.oracle. com/technetwork/mysql-hands-on-lab-403032.pdf

[8] SQL Management Studio for SQL Server, https://www.sqlmana ger.net/download/msstud

[9] Welling, L, Thomson, L. (2009): PHP and MySQL: Web Applications, Micro book, Belgrade, 2010.

[10] Michael Stonebraker, Paul M. Aoki, Robert Devine, WitoldLitwin and Michael Olson, Mariposa: A New Architecture for Distributed Data, Computer Science Div., Dept. of EECS, University of California, Berkeley, California 94720

[11] Justin DeBrabant Andrew Pavlo Stephen Tu, Anti-Caching: A New Approach to Database Management System Architecture, The 39th International Conference on Very Large Data Bases, August 26th - 31st 2013, Riva del Garda, Trento, Italy. Proceedings of the VLDB Endowment, Vol. 6, No. 14

[12] Umesh Dubey, How to install Apache, PHP and MYSQL on Windows 10 Machine, https://www.znetlive.com/blog/how-to-install-apache-php-and-mysql-on-windows-10-machine/