An Efficient Method for Speeding up Large-Scale Data Transfer Process to Database: A Case Study

Ginanjar Wiro Sasmito[1], M. Nishom[2]
Informatics Engineering
Polytechnic of Harapan Bersama
Tegal, Indonesia

***Abstract*—Among the of characteristics of Large Data complexity comprising of volume, velocity, variety, and veracity (4Vs), this paper focuses on the volume to ensure a better performance of data extract, transform, and load processes in the context of data migration from one server to the other due to the necessity of update to the population data of Tegal City. An approach often used by most programmers in the Department of Population and Civil Registration of Tegal City is conducting the transfer process by transferring all available data (in specific file format) to the database server regardless of the file size. It is prone to errors that may disrupt the data transfer process like timeout, oversized data package, or even lengthy execution time due to large data size. The research compares several approaches to extract, transform, and load/transfer large data to a new server database using a command line and native-PHP programming language (object-oriented and procedural style) with different file format targets, namely SQL, XML, and CSV. The performance analysis that we conducted showed that the big scale data transfer method using LOAD DATA INFILE statement with comma-separated value (CSV) data source extension is the fastest and effective, therefore recommendable.**

***Keywords*—Big data; speeds up; data processing; data transfer**

I. INTRODUCTION

The existence of an information system in an organization can help improve different aspects, namely improving the organization's efficiency and effectiveness of the business process, decision making, productivity, and competitive advantages[1]. In an organization, data are processed on a daily basis and stored in the server, therefore the volume is always increasing every year [2]. Indonesia, as the fourth biggest country in terms of population [3], has utilized information system to manage its population data. The volume size of population data is increasing every year, and it requires new server upgrade as well as database migration to server. In practice, there are several methods that can be used in the database migration, namely data import to database server using default (built-in) import feature, third party application suite like phpmyadmin [4] and navicat [5], and using standalone application developed by the programmer itself using Extract, Transform, Load (ETL) Procedures. The Procedures are performed by collecting the data from different sources as needed, modifying it according to the needs, and uploading it to specific database server to be processed or displayed as needed [6]. The most frequently used tools for ETL process are spreadsheet, relational database, non-SQL database, and many more [7].

During the ETL process, selecting correct file format to import or transfer large data is a challenge for the programmer and database administrator, because as the size getting bigger, it affects the execution or transfer time. In the MySQL database server, the data loading process can be performed using different file formats, namely SQL, XML, CSV format, or Excel spreadsheet [5]. Some authors chose different file format to import the data, like [8] chose CSV file format because it is accessible through excel spreadsheet application and data is presented in tabular form, therefore it is easier to access and modify according to the specific needs, and chose SQL format or text file by using mysqldump and mysql to export and import the data [9][10][11], while delisle (in his book) used phpmyadmin to import the data [12]. Other than file format, selecting the appropriate method in the transfer process also need to be considered, such as whether the data is transferred using single-row INSERT statement method, or using INSERT statement with multiple VALUES lists (or generally referred to as extended INSERT) to include all data simultaneously, or using LOAD DATA statement. LOAD DATA Statement is the most appropriate method to transfer large data [13].

One of the issues frequently faced by a database administrator even the programmer is the allowed limit of package size per connection (packet too large). In server MySQL terminology, a package is a request sent to the server and processed by the server in a chunk or batch, simultaneously the server allocates the temporary memory to sore each package and ensures that the memory of servers is still sufficient to prevent a server from running out the memory [14]. The default value for a package allowed in MySQL 8.0 server is 64 MB, but we can set a bigger value for this system variable as the size of package will depend on the availability of memory in the server [15]. The rule of thumb for this is that buffer_pool must be set to 75% or 80% of the server's memory. However, the ratio does not have sufficient basis, therefore only works intermittently, but no guarantee whether it will be working better in a specific case [16]. Hence, in a big size data transfer, an appropriate approach or selection of statement used and file format are required to ensure smooth data transfer.

For that reason, this research aims to achieve the above objectives by comparing the ETL process using different file formats with different statements to find the most appropriate and efficient method to transfer large data into the database server.

II. METHODS

In general, the data transfer process into the server is performed regardless of the file size or amount of data to be transferred, while the database server usually has certain limitations to process the data transfer, such as the maximum file size for transfer, the limit of allowed package size on each statement and transaction, and the net buffer size limit. Despite in practice it is possible to change the value of these limits; such changes may affect the execution time, which may relatively need a longer time and the possibility of transfer failure due to timeout as the result of processing significant amount of data. The approach proposed in the research is Extract-Transform-Load (ETL) approach using appropriate file format and statement. The concept of this approach is performing data source extraction process, transforming data to SQL, XML, and CSV file format, increase buffering and decrease durability in the server configuration, and transferring the data and test the speed of its execution time.

III. RESULTS AND DISCUSSION

A. Extraction and Transformation of Data Source to the Expected Format File

The file or data source used in the research is the population data of Tegal City obtained from the Depatment of Population and Civil Registration of Tegal City. The file is an Excel spreadsheet sized 106MB containing 284917 data rows consisting of 55 fields/columns. The Excel spreadsheet was chosen because data source can be presented in tabular format and enables the stakeholders to customize the data according to their needs. First, data source is extracted using PHP language program to determine the value of data to be transferred to the new database server. Second, the result of the extraction then converted into different file formats, namely SQL, XML, and CSV. Table I shows the conversion of data source to the file format to be used in the data transfer tests, and each file has a different size.

Table I explains the conversion scenario to obtain SQL format file using two approaches, i.e. using single-insert statement (one-row insert per statement) and extended-insert (multiple-row insert per statement) in a transaction.

TABLE. I. RESULT OF DATA SOURCE CONVERSION TO FILE FORMATS TO BE TRANSFERRED

Data Source	Output Format	Statement	Transaction	Output File Size
Excel Spreadsheet (106MB)	SQL	Single-insert (one-row per statement)	1	136 MB
	SQL	Extended-insert (multiple-row per statement)	1	126 MB
	XML	LOAD XML INFILE	1	704.2 MB
	CSV	LOAD DATA INFILE	1	120.2 MB

B. Methods used for Data Transfer

This research used MySQL as the database. The configuration is applied to speed up processing, shown in Table III. There are two ways to transfer the data into the database server online. First, data sources (*.sql, *.csv, and *.xml extension files) are uploaded to the directory of the root server using FTP or cPanel. Second, data is imported or transferred using specified approaches. The import/transfer of data to database server can be performed using several methods. First, database is imported using mysql command using command line (Windows, Linux, or Unix); the file format used was SQL (single-row insert and multiple-row insert per statement). However, this method of data import sometimes takes time especially if the size of the file being imported is large or extensive. Secondly, using LOAD DATA INFILE statement, the file format used is CSV. The third, using LOAD XML INFILE statement to transfer the data using XML file format. Since the server used is an online server, the command line is executed using SSH network protocol using the third-party application. From all three methods, the result of the test showed that the data transfer process using LOAD DATA INFILE statement is the fastest compared to other statements, as shown in Table II. The average of total data transferred per second is shown in Fig. 1.

TABLE. II. COMPARISON OF DATA TRANSFER TIME USING SEVERAL APPROACHES

File Format	Statement	Transaction	Rows	Transfer Time
Command Line Way				
SQL	Single-insert	1	284917	21.6 sec
	Extended insert	1	284917	18.6 sec
XML	LOAD XML INFILE	1	30000	7 min 34 sec
CSV	LOAD DATA INFILE	1	284917	6.05 sec
Native-PHP: Prosedural Way				
SQL	Single-insert	1	284917	18.73 sec
	Extended insert	1	284917	13.74 sec
XML	LOAD XML INFILE	1	30000	8 min 2.3 sec
CSV	LOAD DATA INFILE	1	284917	6.56 sec
Native-PHP: Object Oriented Way (MySQLi)				
SQL	Single-insert	1	284917	18.65 sec
	Extended insert	1	284917	13.35 sec
XML	LOAD XML INFILE	1	30000	8 min 2.2 sec
CSV	LOAD DATA INFILE	1	284917	6.68 sec
Native-PHP: Object Oriented Way (PDO)				
SQL	Single-insert	1	284917	18.55 sec
	Extended insert	1	284917	13.7 sec
XML	LOAD XML INFILE	1	30000	8 min 2.6 sec
CSV	LOAD DATA INFILE	1	284917	6.7 sec

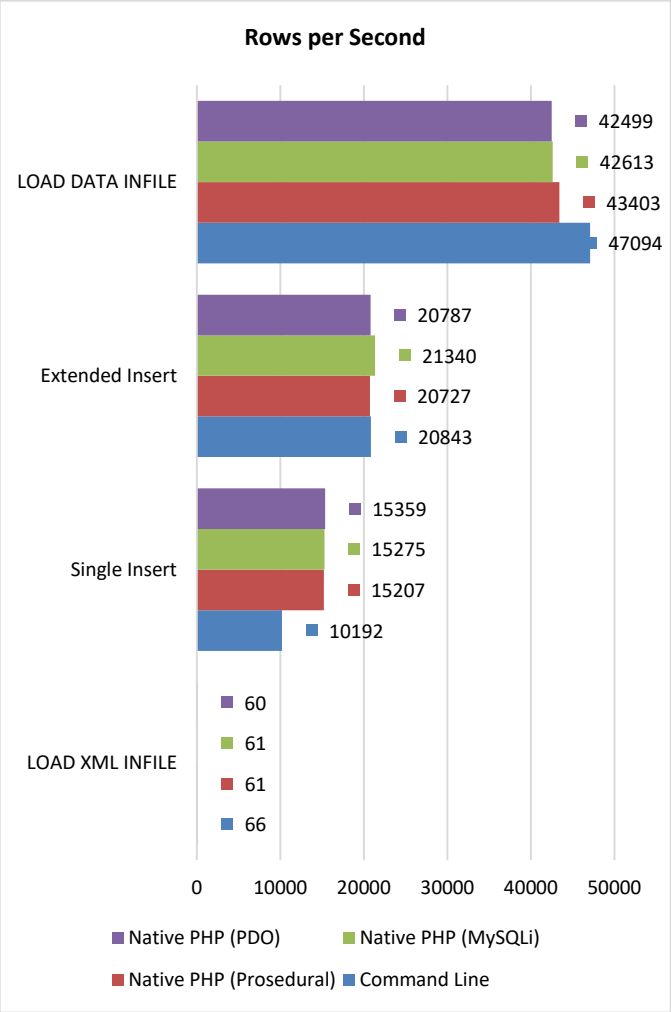

Fig. 1. The Average of Total Data Transferred Per Second using Several Approaches.

TABLE. III. INCREASE PACKET AND BUFFERING, AND DECREASE DURABILITY FOR SPEEDS UP PROCESSING

Name	Value	Description
max_allowed_packet	2048M	Adjust size for handling big queries
key_buffer	64M	Speeds up of index reads
innodb_buffer_pool_size	4096M	25% of physical memory (tested using 16G)
innodb_log_file_size	1024M	25% of buffer pool size
innodb_log_buffer_size	256M	25% of log file size

C. Recommendation and Challenge of Data Transfer Methods

Fig. 1 present the result of the tests using several methods related to the number of transferable data rows to server per second. This transfer method using single-row insert statement is deemed to be less effective in data transfer process (of big size), as it takes a longer time, therefore not recommended. Different from single-row insert statement, the extended-insert statement has slightly better performance. Unfortunately, there is no comprehensive approach to determine the number of ideal data rows in each per package data transfer process. Since the data transfer using extended-insert statements relies heavily on the maximum configuration of maximum allowed packet), therefore the probability of timeout even error is high. For that reason, a database administrator or programmer must be able to determine the number of maximum packets allowed according to the specifications of the server's hardware products. The specification of hardware used (ideally must be more significant than the size of the data source to be transferred). The better the server specification to be used and the bigger the configuration size or value will prevent the issues from occurring during the transfer data process using this statement. Nevertheless, using this statement is recommended as it is faster than the single-row insert statement. To speed up the processing, it is better to increase the Buffer Pool size by 20-50% from the physical memory size, and increase the Log File size to 25% from the Buffer Pool size.

Different from the previous two statements, the data transfer method using LOAD XML statement has poor time performance. The bigger the file size, the longer the execution process or data transfer. The LOAD DATA statement is inversely proportional to extended-insert statement and does not rely on the server configuration (the maximum size of allowed packet size per statement); therefore, this method is the most suitable for large data transfer process. Moreover, the statement has excellent data transfer speed as described in Fig. 1, that data transfer using LOAD DATA statement is the best way as it is the fastest in processing data transfer.

IV. CONCLUSION

The objective of the research is to determine the most efficient method to transfer the data in large size to database server. The testing of the methods was performed by using extract-transform-load approach using single-insert statement, multiple-insert (extended-insert) statement, LOAD XML statement, and LOAD DATA statement. The result of the tests showed that the data transfer data method with single-insert statement is not recommended due to low-speed transfer. Multiple-insert statement to transfer large data is also not recommended because the possibility of timeout or errors if the packet delivered per statement exceeds the allowed packet value in the server configuration. The data transfer method using LOAD XML statement is the worst choice and suggested not to be used for large scale data transfer. Instead, data transfer method using LOAD DATA is the most recommended method to transfer large data. However, a database administrator or programmer must consider the appropriate server configuration to avoid problems during large scale data process.

ACKNOWLEDGEMENT

The author would like to thank the Depatment of Population and Civil Registration and Statistics Agency of Tegal City for sharing the data source and Ministry of Technology Research and Higher Education of the Republic of Indonesia for funding this research.

REFERENCES

[1] R. M. Stair and G. W. Reynolds, Fundamentals of Information Systems, 8 ed., Boston: Cengage Learning, 2014.

[2] K. A. I. Hammad, M. A. I. Fakharaldien, J. M. Zain and M. A. Majid, "Big Data Analysis and Storage," in International Conference on Operations Excellence and Service Engineering, Florida, 2015.

[3] CIA, The World Factbook, Langley: Central Intelligence Agency, 2019.

[4] P. McFedries, Web Coding & Development All-in-One For Dummies, Hoboken: Wiley, 2018.

[5] G. Ozar, "MySQL Management and Administration with Navicat," Birmingham, 2012.

[6] L. Baldacci, M. Golfarelli, S. Graziani and S. Rizzi, "QETL: An approach to on-demand ETL from non-owned data sources," Data & Knowledge Engineering, vol. 112, pp. 17-37, 2017.

[7] S. Challawala, J. Lakhatariya, C. Mehta and K. Patel, MySQL 8 for Big Data: Effective data processing with MySQL 8, Hadoop, NoSQL APIs, and other Big Data tools, Birmingham: Packt publishing, 2017.

[8] R. J. Dyer, Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB, Sebastopol: O'Reilly Media, 2015.

[9] S. K. Cabral and K. Murphy, MySQL Administrator's Bible, Hoboken: Wiley, 2009.

[10] P. DuBois, MySQL Cookbook, 3 ed., Sebastopol: O'Reilly Media, 2014.

[11] P. Zhang, Practical Guide to Large Database Migration, Boca Raton: CRC Press, 2019.

[12] M. Delisle, Mastering Phpmyadmin 3.4 for Effective MySQL Management, Birmingham: Packt Publishing, 2012.

[13] P. Scobey and P. Lingras, Web Programming and Internet Technologies: An E-commerce Approach, Burlington: Jones & Bartlett, 2012.

[14] S. Pachev, Understanding MySQL Internals: Discovering and Improving a Great Database, Sebastopol: O'Reilly Media, 2007.

[15] C. Mehta, A. K. Bhavsar, H. Oza and S. Shah, MySQL 8 Administrator's Guide: Effective guide to administering high-performance MySQL 8 solutions, Birmingham: Packt Publishing, 2018.

[16] B. Schwartz, P. Zaitsev and V. Tkachenko, High Performance MySQL: Optimization, Backups, and Replication, Sebastopol: O'Reilly Media, 2012.