Performance Comparison of Bitcoin Prediction in Big Data Environment

(Spark Architecture and GPU Environment))

Sumarsih C. Purbarani
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia

Wisnu Jatmiko
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia
wisnuj@cs.ui.ac.id

***Abstract*—In today's world, everything is transforming to digital forms. These yield large amount of data. A good analysis of these data can lead to new knowledge about the present situation as well as the future insight. While many advantages could be obtained from these large data, the issue on how to run the Machine Learning on a large dataset as effective and efficient as possible remains an open problem. In this paper, data processing simulation using machine learning algorithm of Linear Regression is conducted to learn from Bitcoin trading dataset. The simulation is carried out in Apache Spark cluster architecture and GPU. The running time and error of the algorithm implementation in both architectures are compared with each other. The simulation results show similar error performance between Apache Spark cluster and GPU. Yet, Apache Spark can run the simulation 2x faster than GPU.**

Keywords—Spark; GPU; parallelization; bitcoin

I. Introduction

Machine Learning learns from data. Machine Learning algorithms often work in an iterative way, namely Linear Regression, Gradient Descent, and so on. Machine Learning has many useful applications, such as pattern recognition, text mining, predictions, recommender systems, etc. These applications undoubtedly require many data to perform the best. Moreover, everything is transforming into digital forms: business activities, government programs, education, health services, etc. These all yield large amount of data. A good analysis of these data can lead to new knowledge about the present situation as well as the future insight. While many advantages are obtained from these large data, how to run the Machine Learning on a large dataset as effective and efficient as possible remains an open problem [1]. Some researchers utilize data stream algorithm to solve big data problems [2][3].

Apache Spark, an in-memory data processing engine, is known as a fast and scalable system to handle big data [4][5]. It works by distributing the workload parallelly instead of executing serially. Apache Spark make use of memory in Random Access Memory (RAM) instead of accessing I/O disk, which is slow. Thus, it is advantageous, especially when an algorithm run on it needs an iterative work [6]. It is suitable for running Machine Learning algorithms [6][7]. Apache Spark consists of cores and libraries to execute distributed computation. The algorithm which does this job is MapReduce. Once a problem encounters an Apache Spark cluster, it is divided into smaller computational tasks. Each of them is carried out by an executor. Apache Spark works parallelly in a cluster of distributed nodes consists of master and worker node. Master node manages and distributes tasks to worker nodes. Worker nodes execute each task parallelly, then aggregate those computations into single result and send it back to the master node. Apache Spark may optimally make use of all memory available in node, but the processing done on a Central Processing Unit's (CPU's) limited parallelism [8]. Adding memories (RAMs) since they are getting cheaper now can be a solution, but how effective it is remains debatable[1].

A Graphics Processing Unit (GPU) offers an alternative in parallel computing. GPU computes data to be displayed on the monitor. It has powerful parallelization scheme so that it can send display continuously in an almost no time lag. Even nowadays, as more sophisticated GPU technology is developed that makes offer higher resolution, hence clearer, faster, and

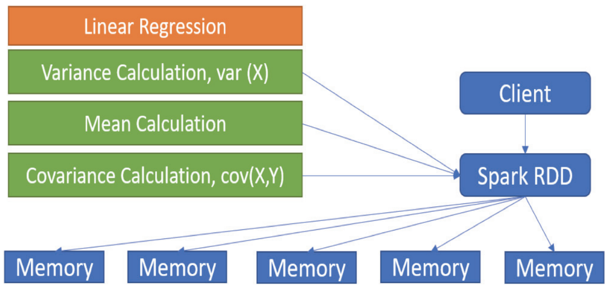

Fig. 1. Apache Spark's work mechanism – illustration of Linear Regression implementation

[1] https://www.nextplatform.com/2016/02/19/the-myth-of-in-memory-computing/

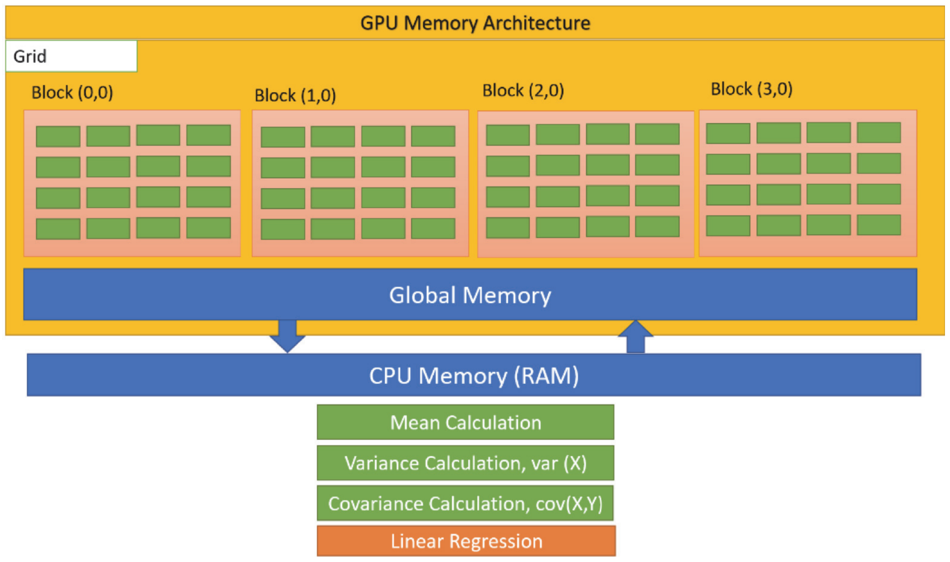

Fig. 2. GPU's work mechanism – illustration of Linear Regression implementation

smoother display[2]. Not only to process graphic data, GPU's parallelization is utilized to process non-graphic applications.

An idea to marry Apache Spark with GPU is also considered in several researches. Ohno, et al. combine the CPU limitation of Spark and the parallelization power of GPU [9]. The experiment shows that Spark with GPU can accelerate the standard Spark's performance by 21.4x.

A cloud computing platform to process big data application has been made [11]. In addition, some techniques and algorithm to process big data has been evaluated to have better accuracy and performance [12][13].

II. Methods

Apache Spark is a big data framework that emerges by providing libraries for scalable machine learning, structured data, streaming data processing, and graph analysis [4]. In this experiment, a scalable Linear Regression –a machine learning algorithm for regression problem– is conducted.

A. Linear Regression

Linear Regression examines the interdependencies between targeted or predicted variable and explanatory variables [10]. Given the following equation(1), Linear Regression is simply a linear algebraic operation.

$$y = a + b\boldsymbol{X} \tag{1}$$

where y represents targeted variable, $\boldsymbol{X}$ represents explanatory variable vector, a and b represent bias and weight respectively. When $\boldsymbol{X}$ and $\boldsymbol{Y}$ plotted in an cartesian coordinate, a is the intercept of the graph, while b is the slope of the line.

B. Data Partitioning

In Spark, parallel computing is handled by data partitioning scheme. Spark partitions data in form of Resilient Distributed Datasets (RDDs).

In Fig. 1, the work mechanism of Apache Spark's is explained. Some calculation of how Linear Regression works is also briefly illustrated. Those calculation includes mean, variance and covariance as can be formulated by the following equation (2), (3) and (4).

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n} \tag{2}$$

$$cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - E(X)\right)\left(y_i - E(Y)\right) \tag{3}$$

$$Variance\ (X) = \sum_{i=0}^{n} p_i\,(x_i - \mu)^2 \tag{4}$$

These calculations can be executed using Apache Spark RDD, which is an abstraction of data structure in Apache Spark. Apache Spark RDD is distributed in memory of each Apache Spark node's, hence works parallelly. Parallelization in GPU

In Fig. 2, the work mechanism of GPU is explained. Compared to CPU, GPU has more limited memory capacity. In Linear Regression, where only linear arithmetic calculation is considered, GPU is still able to execute the arithmetic calculation but not efficient enough in utilizing cores. Whereas, GPU has more cores than that of CPU has. Each arithmetic calculation is processed several times through a data passing mechanism from CPU memory to the global memory of GPU.

[2] https://www.pcmag.com/encyclopedia/term/43886/gpu

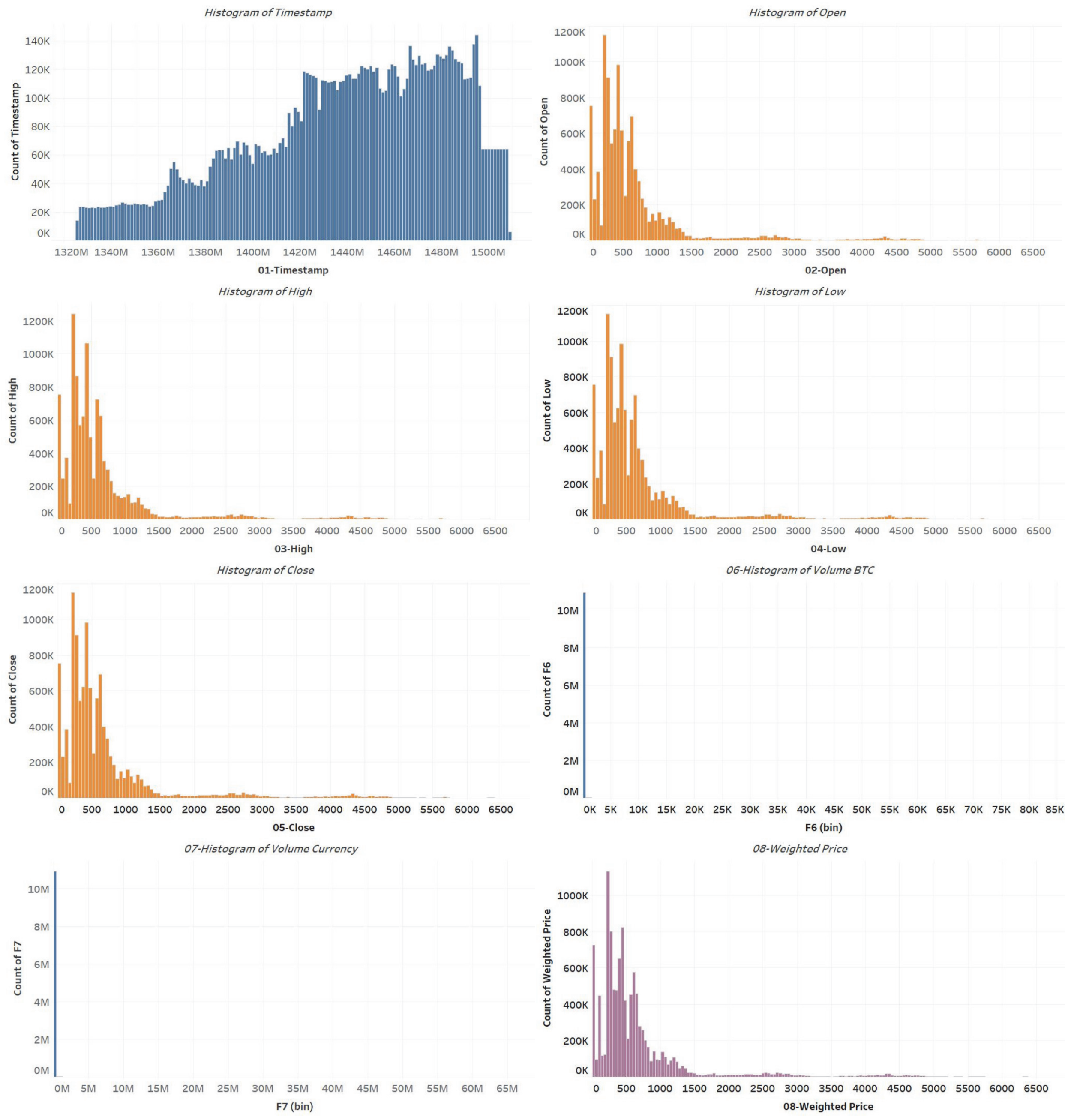

Fig. 3. Histogram of dataset

The process is then continued to be operated in blocks inside the GPU grids. Mean, variance and covariance calculation of dataset is not feasible to be done in GPU since the process cannot be parallelized. In other words, GPU operates like CPU but using more limited memory.

C. Data Collection and Preprocessing

Data used in this research is a bitcoin price dataset from January 2012 until October 2017. The dataset is available for public download in Kaggle. It is made up of seven .csv files each of which has size of 877 MB in total. Each file in the dataset is bitcoin trading dataset in several bitcoin exchange platforms in various currencies, i.e. US Dollar (Bitstamp (https://www.bitstamp.net/), BTCE (https://btc-e.com/), Coinbase (https://www.coinbase.com/) and Kraken (https://www.kraken.com/)); China Yuan (BTC China (https://www.btcchina.com/)); Japan Yen (Coincheck (https://coincheck.com/)); and Euro (Kraken).

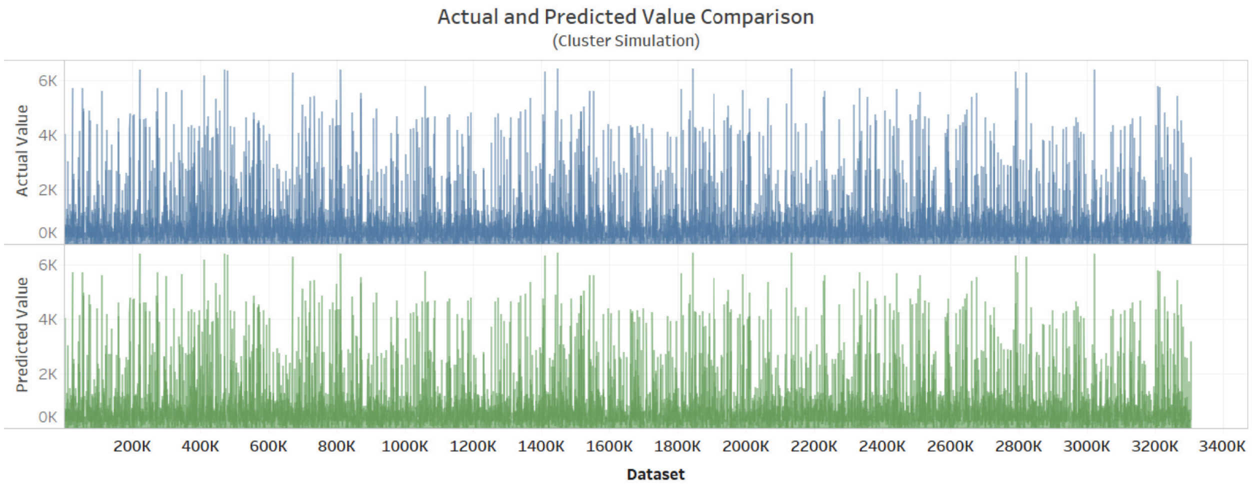

Fig. 4. Predictio n result of Bitcoin compared to its actual value

Each dataset consists of eight features, i.e. 'timestamp' in Unix format, 'open' that represents Bitcoin price at the opening of that period of time, 'high' that represents the Bitcoin highest price at that period of time, 'low' that represents the Bitcoin lowest price at that period of time, 'close' that represents the Bitcoin price at the closing of that period of time, 'volume-btc' that shows the volume of bitcoin, 'volume-currency' shows the volume of bitcoin currency's.

These files are concatenated into a single dataset hence creating a total record of 11,021,404 and making the total size of 1.1 GB. Since the range between feature value in this dataset is wide enough, a normalization into z-score is conducted. The normalized form of all features can be seen in Fig. 3.

Fig. 3 depicts the histogram of each features in bitcoin dataset. 01-timestamp histogram colored in blue shows the timestamp feature. 02-open, 03-high, 04-low, 05-close histogram colored in orange represent bitcoin opening price, highest, lowest and closing price respectively. The remaining two histograms, i.e. 06-volume-btc and 07-volume-currency shows volume-btc and volume-currency respectively. According to the histogram, these features have major distribution of 0 value. Thus, the left part of the histograms' is very high. Lastly, 08-weighted-price plotted in purple colored histogram shows the targeted or predicted value.

This raw dataset still consists of invalid data such as null values. A preprocessing is conducted to cleanse up the dataset by removing null value and selecting only noteworthy features. The feature selection process is done based on the correlation of each features toward the target or labeled feature, i.e. 'bitcoin price'.

TABLE I. CORRELATION RESULT OF ALL FEATURES

Feature	Correlation
timestamp	0.588317
open	0.999997
high	0.999999
low	0.999991
close	999999
volume-btc	0.00875
volume-currency	0.061673

The Pearson Correlation is conducted to measure it. Table I shows the correlation result of all features. The correlation is ranged from -1 to +1. Negative sign shows that the feature has an opposite direction towards the predicted value, i.e. as the value of the respected feature is getting lower, the value of targeted feature is higher. On the other hand, positive shows that it has a positive or same direction towards the predicted value. Values between -1 and +1 indicate how strong the feature is correlate with the target.

According to the correlation test result, four most significant features are selected, i.e. 'timestamp', 'open', 'high', 'low' and 'close'. While 'volume-btc' and 'volume-currency' remain unused due to the sparsity level in the features. It complies with the histogram in Fig. 3 that normally distributed histograms have a strong correlation or that slight changes in the features' value have significant impact to the targeted value. Finally, these five features are trained in a regression algorithm to predict the targeted feature, i.e. the weighted price of bitcoin. The performance of the regression model is evaluated in terms of Mean Squared Error (MSE). The definition is as formulated by equation(5).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \qquad (5)$$

III. EXPERIMENT AND RESULT

The experiment is conducted using two architectures, namely GPU and Apache Spark architecture. Linear Regression is implemented to the same dataset running in both architectures. The dataset is split into 70% for training and 30% for testing. The same treatment is given to both architecture with each of which has the following specifications as seen in Table II.

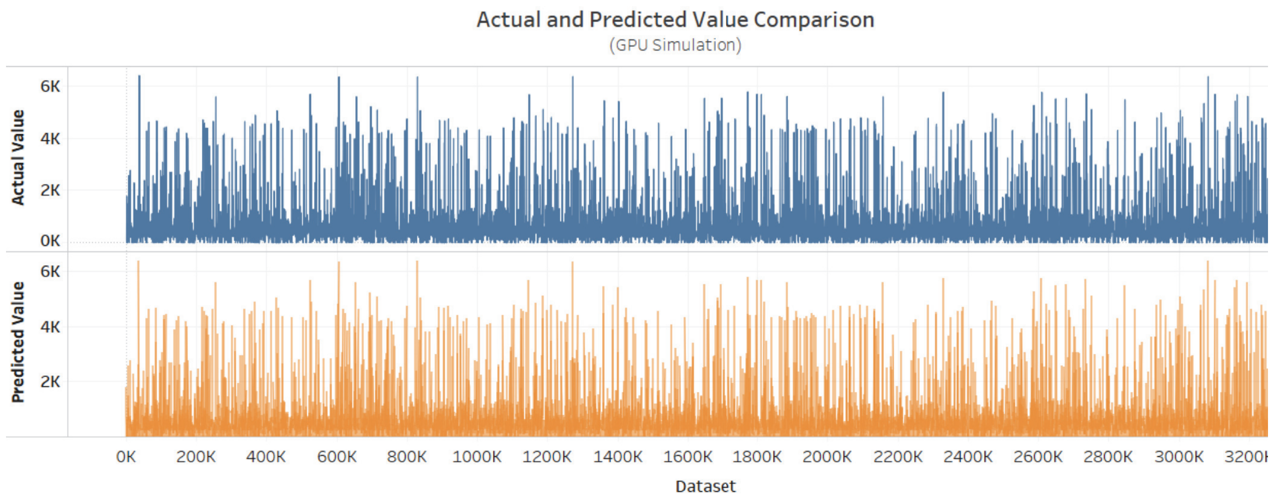

Fig. 5. Prediction result of Bitcoin compared to its actual value

According to the table, GPU Nodes with 2x NVIDIA provides the latest specification and has the most powerful GPU compared to the others. Meanwhile, Apache Spark cluster nodes uses 2600 series of Intel i7, a quite outdated version of processor.

TABLE II. SPECIFICATION OF APACHE SPARK AND GPU ARCHITECTURE

	Apache Spark Cluster Nodes		**GPU Nodes**	
Processor	Intel I7-2600 3.4GHz (8 Cores)	Intel I5-760 2.8 GHz (4 Cores)	Intel I7-4790 4.0 GHz (8 Cores)	Intel I5-7400 3.0 GHz (4 Cores)
RAM	32 GB	16 GB	32 GB	24 GB
GPU	GTX 560	GTS 250	2x NVIDIA GTX 780, 3GB Memory	2x NVIDIA GTX 1070, 8GB Memory
Role	Head node	Slave node	Single node	Single node
Number of Nodes	1 node	3 nodes	-	-

A. Cluster Simulation

The characteristic prediction of bitcoin price being compared to its actual price is plotted in Fig. 4. This graph is the result of a simulation done in Apache Spark cluster architecture. The blue-colored-graph illustrates the actual Bitcoin value, while the green-colored one illustrates the predicted value of each dataset. According to the measurement, the maximum prediction value is 6.417 and the maximum actual value is also 6.417. This happens in the 1,448,496th record. On the other hand, the biggest error value happens in the 375,551st record with the error of 16.18. Overall, the MSE value for Linear Regression performance running in Apache Spark cluster environment is 0.0004.

B. Cluster Simulation

The characteristic prediction of bitcoin price being compared to its actual price is plotted in Fig. 4. This graph is the result of a simulation done in Apache Spark cluster architecture. The blue-colored-graph illustrates the actual Bitcoin value, while the green-colored one illustrates the predicted value of each dataset. According to the measurement, the maximum prediction value is 6.417 and the maximum actual value is also 6.417. This happens in the 1,448,496th record. On the other hand, the biggest error value happens in the 375,551st record with the error of 16.18. Overall, the MSE value for Linear Regression performance running in Apache Spark cluster environment is 0.0004.

C. GPU Simulation

Fig. 5 shows the actual price characteristic compared to Bitcoin predicted price. This simulation is run using 2x GPU GTX 780 and 2x GPU GTX 1070 architecture. The blue-colored-graph illustrates the actual Bitcoin value, while the green-colored one illustrates the predicted value of each dataset. According to the measurement, the maximum prediction value is 6.417 and the maximum actual value is also 6.417. On the other hand, the biggest error value happens in the 1,819,867th record with the error of 123.7. Overall, the MSE value for Linear Regression performance running in Apache Spark cluster environment is 0.0017.

D. Running Time and Error Performance

Fig. 6 shows the error of Apache Spark cluster simulation and GPU simulation separately. It is ranged between 0-16.18 in the former simulation and 0-123.7 in the latter. In general, both simulations show similar error values, i.e. ranged between 0-20. Yet, while run in GPU architecture, the 1,819,867th record of the dataset gains a significant error, i.e. 123.7.

Fig. 7 depicts the time performance of each architecture in processing the dataset. According to the figure, GPU NVIDIA 2x GTX 780 performs the simulation in 6 minutes 32 seconds

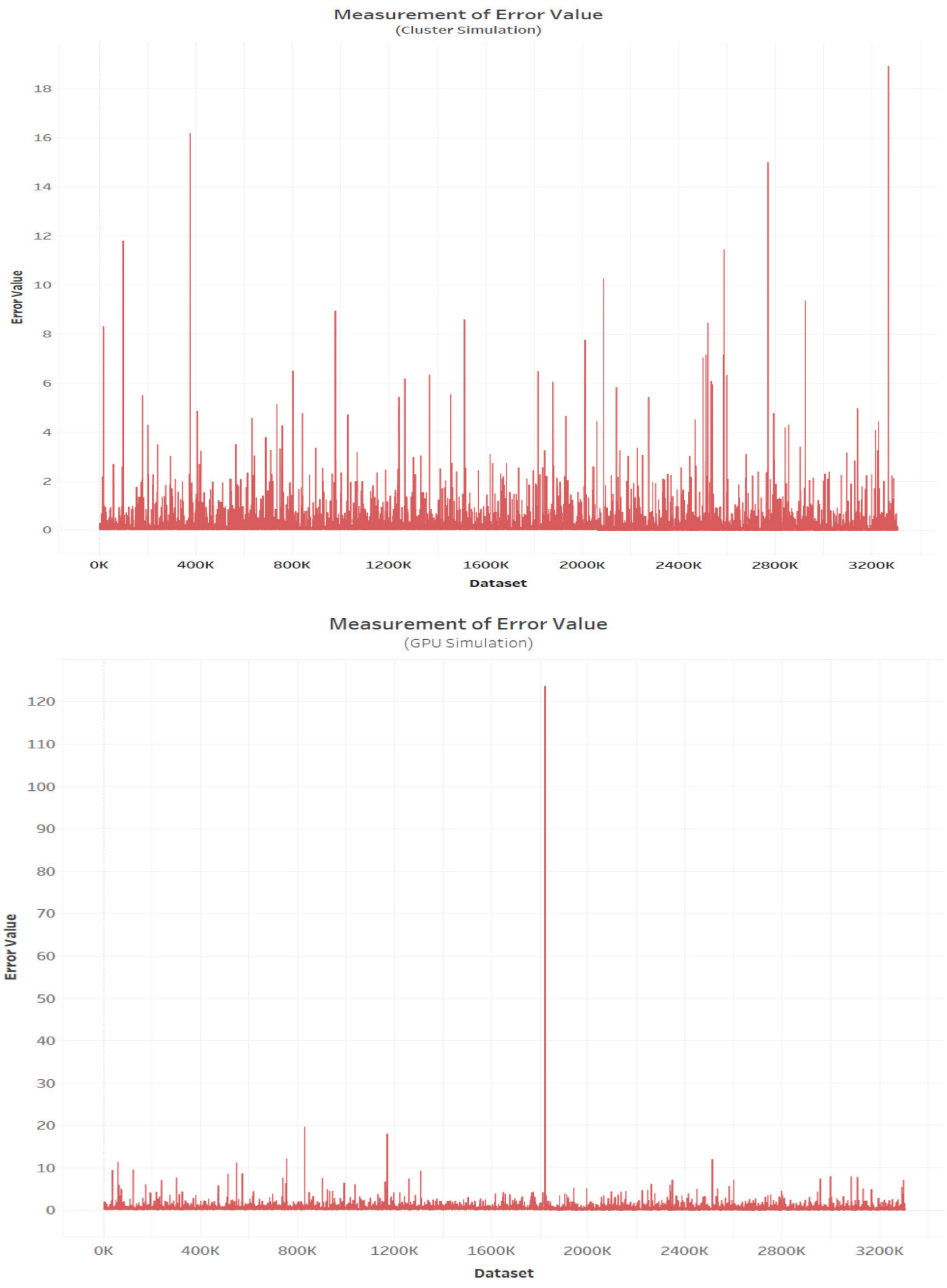

Fig. 6. Measurement error value in cluster and GPU simulation

which is the slowest among others. GPU NVIDIA 2x GTX 1070 shows a little bit faster performance by accomplishing the task in 5 minutes 6 seconds. The fastest simulation executed in Apache Spark cluster by 3 minutes 51 seconds.

IV. Analysis and Discussion

As a measurement result of error, it can be stated that practically, errors in both architecture are quite similar. It is shown by the MSE in Apache Spark cluster is 0.0013 and in GPU nodes is 0.0014. According to running time performance, time for processing data in Apache Spark cluster is faster compared to that in 2xGPU NVIDIA GTX 780 and 2x GPU

NVIDIA GTX 1070. The Apache Spark cluster used in this research is not built in the latest hardware technology, instead, it is built in a standard desktop computer. The adequate performance of Apache Spark cluster in this experiment is due to Apache Spark's work principle that is to distribute jobs to other nodes. Thus, the modelling algorithm works based on the submitted notion of job.

The most reasonable justification why GPU processes the data in a slower time compared to Apache Spark is that Linear Regression is not able to effectively utilize its parallelization ability when run in GPU. In Linear Regression algorithm, the regression function is determined by measuring the distance between any point to the perfect line (actual value) in each feature. This process is quite different from matrix multiplication problem, in which each addition of row*column multiplication can be parallelized. Each parallelization can then be aggregated to get the result of matrix multiplication. One way to obtain a better performance of Linear Regression is by escalating the hardware's serial processing ability.

Further development of this paper is more simulation analysis of various algorithms will be conducted in Apache Spark cluster architecture as well as in GPU. The experiment and analysis will be implemented in different dataset, hence different problem. For instance, clustering web news feeds [14], improving classification performance on health data [15][16], or even high-dimensional data [17], etc.

V. Conclusion

In this paper, data processing simulation using machine learning method of Linear Regression is conducted. The simulation is carried out in Apache Spark cluster architecture and GPU. The testing results shows similar error performance between Apache Spark cluster and GPU. However, the running time performance of Apache Spark cluster is faster than that of GPU, both using GPU 2xGTX780 and GPU 2xGTX1070 hardware. Based on the experiments, it can be concluded that Apache Spark architecture outperforms GPU in terms of running time. The main rationale that causes hundreds of cores in GPU cannot optimize the simulation's running time is that the regression function in Linear Regression algorithm cannot be maximized using GPU parallelization. The utilization of GPU parallelization can be optimized if each core in GPU is given a balance load of tasks for then all results are aggregated, hence improving its running time performance. The performance of Linear Regression algorithm can also be improved using the hardware's processing ability. According to this experiment, GPU utilization can be adapted with the algorithm it is going to run. If the algorithm can maximize the parallel processing ability in GPU, then the simulation can run more efficiently with a shorter running time. Nevertheless, if the algorithm cannot make use of the parallel processing ability in GPU maximally, then using Apache Spark cluster can be an alternative since it provides a better running time compared to GPU.

Acknowledgment

This article's publication is partially supported by the United States Agency for International Development (USAID) through the Sustainable Higher Education Research Alliance (SHERA) Program for Universitas Indonesia's Scientific Modeling, Application, Research and Training for City-centered Innovation and Technology

(SMART CITY) Project, Grant #AID-497-A-1600004, Sub Grant #IIE-00000078-UI-1.]. No:0141/UN2.R3.SC/HKP.05. 01/2018. We would thank to Ari Wibisono for dataset supply, infrastructure support both GPU and Spark environment, source codes and tutors.

References

[1] M. Grossman and V. Sarkar, "Swat," in Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing - HPDC '16, 2016, pp. 81–92.

[2] Jatmiko, W., Purnomo, D.M.J., Alhamidi, M., Wibisono, A., Wisesa, H., Mursanto, P., Bowolaksono, A., Hendrayanti, D., Addana, F. Algal growth rate modeling and prediction optimization using incorporation of MLP and CPSO algorithm (2016) 2015 International Symposium on Micro-NanoMechatronics and Human Science, MHS 2015, art. no. 7438293.

[3] W. Jatmiko, A. Wibisono, H. A. Wisesa, P. Mursanto and D. Sarwinda, "Perceptron rule improvement on FIMT-DD for large traffic data stream," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, 2016, pp. 5161-5167.

[4] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," Int. J. Data Sci. Anal., vol. 1, no. 3–4, pp. 145–164, 2016.

[5] Y. Yuan, M. F. Salmi, Y. Huai, K. Wang, R. Lee, and X. Zhang, "Spark-GPU: An accelerated in-memory data processing engine on clusters," Proc. - 2016 IEEE Int. Conf. Big Data, Big Data 2016, pp. 273–283, 2016.

[6] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "MLlib: Machine Learning in Apache Spark," J. Mach. Learn. Res., vol. 17, pp. 1–7, 2016.

[7] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin, "Large-scale Logistic Regression and Linear Support Vector Machines using Spark," 2014 IEEE Int. Conf. Big Data (Big Data), pp. 519–528, 2014.

[8] P. Li, Y. Luo, N. Zhang, and Y. Cao, "HeteroSpark: A heterogeneous CPU/GPU Spark platform for machine learning algorithms," Proc. 2015 IEEE Int. Conf. Networking, Archit. Storage, NAS 2015, pp. 347–348, 2015.

[9] Y. Ohno, S. Morishima, and H. Matsutani, "Accelerating spark RDD operations with local and remote GPU devices," Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS, pp. 791–799, 2017.

[10] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," Int. J. Inf. Manage., vol. 35, no. 2, pp. 137–144, 2015.

[11] Wibisono, A., Suhartanto, H. Cloud computing model and implementation of molecular dynamics simulation using Amber and Gromacs (2012), 2012 International Conference on Advanced Computer Science and Information Systems, ICACSIS 2012 - Proceedings, art. no. 6468763, pp. 31-36. -4

[12] Wiska, R., Habibie, N., Wibisono, A., Nugroho, W.S., Mursanto, P. Big sensor-generated data streaming using Kafka and Impala for data storage in Wireless Sensor Network for CO2 monitoring (2017) 2016 International Workshop on Big Data and Information Security, IWBIS 2016, art. no. 7872896, pp. 97-101. -4

[13] Wibisono, A., Wisesa, H.A., Jatmiko, W., Mursanto, P., Sarwinda, D. Perceptron rule improvement on FIMT-DD for large traffic data stream (2016) Proceedings of the International Joint Conference on Neural Networks, 2016-October, art. no. 7727881, pp. 5161-5167. -4

[14] Fadllullah, A., Kamudi, D.D., Nasir, M., Arifin, A.Z. and Purwitasari, D., 2016. Web News Documents Clustering in Indonesian Language Using Singular Value Decomposition-principal Component Analysis (Svdpca) and Ant Algorithms. Jurnal Ilmu Komputer dan Informasi, 9(1), pp.17-25.

[15] Jatmiko, W., Nulad, W.P., Elly, M.I., Setiawan, I.M.A. and Mursanto, P., 2011. Heart beat classification using wavelet feature based on neural network. vol, 10, pp.17-26.

[16] Setiawan, I.M.A., Imah, E.M. and Jatmiko, W., 2011, December. Arrhytmia classification using fuzzy-neuro generalized learning vector quantization. In Advanced Computer Science and Information System (ICACSIS), 2011 International Conference on (pp. 385-390). IEEE.

[17] Purbarani, S.C., Sanabila, H.R., Wibisono, A. and Jatmiko, W., Preliminary Research on Continuous Conditional Random Fields in Predicting High-Dimensional Data.