International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Java Script Data Transformation Library using Fork Join Pool and Web Workers Technology

Drevendy Harianto, Seng Hansun, Andre Rusli

Abstract: *Transforming large amounts of data takes a lot of processing time so that the optimization technique is required. One way that can be used to perform optimization is multithreading. Nowadays, processor is proliferating. The average processor in community is multi-core processor that can do parallel processing. Prior to the emergence of Web Workers, JavaScript is a poor programming language for parallel programming. The emergence of Web Workers allows JavaScript to do a better job in parallel programming. Fork Join Pool is a method that implements the Divide and Conquers algorithm, so it is suitable for the use in multithreading. This data transformation library was created by implementing the ForkJoinPool method using Web Workers technology in JavaScript. This program is written in JavaScript and HTML language. Based on results of testing phase that has been done, it is proven that ForkJoinPool method can be implemented using Web Workers technology in JavaScript as a data transformation library. In addition, it can be concluded that the data transformation library usage affects the speed of data transformation which depends on the data transformation complexity. The higher the complexity of data transformation performed, the effectiveness in the use of data transformation libraries will increase.*

Index Terms: *Fork Join Pool, JavaScript, Data Transformation Library, Multithreading, Web Workers.*

I. INTRODUCTION

In the past two decades, multithreading on multi-core processors has been used to improve software performance [1]. Currently, multi-core processors are hardware, which is growing very rapidly, so it is a must to develop multi-thread software [2]. When compared to other programming languages, JavaScript is a poor programming language in parallel programming [3]. Today, on modern computer systems, on average they have used multi-core processors, even on mobile devices. However, this is very contrary to traditional JavaScript that still uses a single-threaded execution model. Therefore, HTML5 provides multithreading capabilities through Web Workers that can be implemented in JavaScript [4].

Research conducted by Okamoto and Kohana has successfully implemented Web Workers in a web-based multiplayer game. Web Workers are used to calculating the position of the avatar then update the position to the server. At first, the entire load calculation of the avatar position is done by the server. However, as players number increase, server load also increases. Therefore, Web Workers are used to sharing the server load to the client. If you use Web Workers, the position update is done every half a second, while without using Web Workers the updates are done every five seconds [5]. To transform massive amounts of data still takes much time. Research conducted by Aminudin and Alwi has succeeded in utilizing multithreading programming to speed up data transformation time on Apache Mahout. In their study, it is known that the existence of multithreading programming can improve performance in transforming data on Apache Mahout. From the test that had been carried out using 20 million data, it was obtained the results of testing on a single-thread with transformation data time of 2,218 seconds. Whereas in testing using four threads, it was obtained 764 seconds [6]. Another research on the usage of multithreading to optimize the Apple Lossless Audio Codec Algorithm using NVIDIA CUDA architecture also had been done by Ahmed et al. [7]. ForkJoinPool is a method from Java that allows parallel programming and is responsible for managing all fork-join tasks in all programs [8]. ForkJoinPool implements the Divide and Conquer algorithm and the Work Stealing algorithm. Divide and Conquer is a paradigm in computing. By using this paradigm, the solution of a problem is obtained by solving the subproblems recursively [9]. Meanwhile, Work Stealing is an algorithm that allows an idle processor to help other processor jobs [10]. Therefore, ForkJoinPool is suitable for transforming large amounts of data by utilizing multi-core processors to run the multithreading process [8]. Research conducted by Ponge has successfully implemented ForkJoinPool in Java to calculate the number of words in a collection of documents. The calculation of the words number by using two cores takes 11.026 seconds. The calculation of the words number by using four cores takes 8.329 seconds. Calculation of the words number by using eight cores requires 4.208 seconds, while the calculation of the words number by using 12 cores takes 2.876 seconds. If the calculation of the words number is done by using single-thread, it takes approximately 18 seconds [11]. Based on the series of problems described above, this study implements the ForkJoinPool method using Web Workers technology in JavaScript as a data transformation library and to know the effect of this implementation on data transformation speed in JavaScript. Library in the context of programming is a set of codes or functions that have been compiled so that they can be used by programmers by calling defined functions [12]. By using the library, the programmer does not need to write the same code repeatedly because the programmer can call the function that has been defined in a library [13].

Revised Manuscript Received on December 30, 2019.
* Correspondence Author

Drevendy Harianto, Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia
Seng Hansun, Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia
Andre Rusli, Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia

II. RESEARCH METHOD

In this section, first, we will briefly explain the multithreading and web workers technology concept in HTML5. Then, we continue with the explanation of data transformation and ForkJoinPool method being used in this study.

A. Multithreading

Multithreading is a type of process execution model that allows multiple threads to run several processes separately [14]. Multithreading can only be implemented on processors that have more than one core.

Based on the size of each task that must be completed by one thread, the level of parallelism is divided into two, namely Fine-grain parallelism and Coarse-grain parallelism. In Fine-grain parallelism, an enormous task will be broken down into several small-sized tasks. Whereas in Coarse-grain parallelism, an enormous task will be broken down into several tasks whose size is still relatively large. The advantage of Fine-grain parallelism over Coarse-grain parallelism no other than it is facilitating the load balancing, where each thread will get a balanced portion of the task. In Fine-grain parallelism, to do 100 small-sized tasks requires 100 threads, whereas in Coarse-grain parallelism, to do 100 tasks requires less than 100 threads. Based on the explanation above, the creation of a library will use the Coarse-grain parallelism level [15].

B. Web Workers Technology

Web Workers is a new feature introduced in HTML5, which allows JavaScript to create a separate thread from the main thread and allows JavaScript to be executed in the background. The advantages of Web Workers are that foreground tasks and user experience are not affected by the script execution in the background [16].

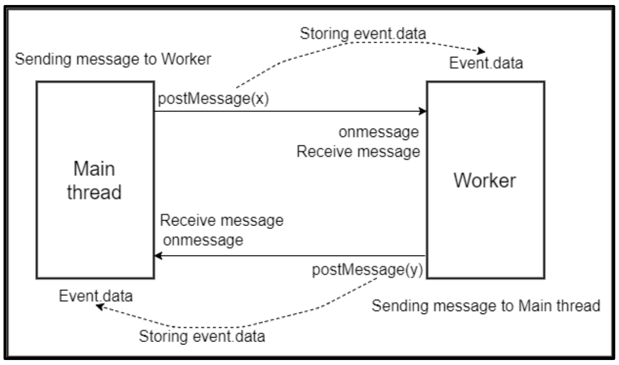


Figure1. How the Web Workers Works [17]

Figure 1 describes how Web Workers works; meanwhile Table 1 informs the limitation of Web Workers.

Table1. Limitation of Web Workers [17]

Function name	Enabled or Disabled
Navigator object	Enabled
Location object	Enabled (read-only)
XMLHttpRequest	Enabled
setTimeout()	Enabled
setInterval()	Enabled
Window object	Disabled
Document object	Disabled
Parent object	Disabled
DOM	Disabled

C. Data Transformation

Data transformation is the process of changing each element of the data, whether changes in the value or the structure of the element. There are several types of data transformation referred to in this study, such as basic Mathematical operations [18] and data processing with string data types [19].

There are four kinds of basic Mathematical operations, namely addition, subtraction, multiplication, and division. The addition is a Mathematical operation that sums up a number with other numbers. The reduction is a Mathematical operation that subtracts a number from other numbers. Multiplication is a Mathematical operation that multiplies a number by another number, while division is a Mathematical operation that divides a number from other numbers [20]. Whereas there are several operations that can be performed on data with string data types, such as concat and replace [21]. Equation 1 is a type of data transformation that will be used in the trial process. Equation 1 is the formula used to calculate a student grade at Universitas Multimedia Nusantara.

$$Grade = (0.3 * Assignment) + (0.3 * MID) + (0.4 * FINAL) \quad (1)$$

D. Fork Join Pool Method

ForkJoinPool is a method that allows parallel processing of data, where the processing work is divided into k tasks where each task will be processed in parallel by the processor cores (Fork). After all tasks have been processed, the results of each task will be combined (Join) into one [22]. The maximum number of task branching is as many as the number of cores on the processor [23].

One of the advantages of ForkJoinPool is implementing the Work Stealing algorithm [24]. When a worker thread has finished executing a task and no work can be done by the worker thread, the worker thread can help to complete the task on another thread.

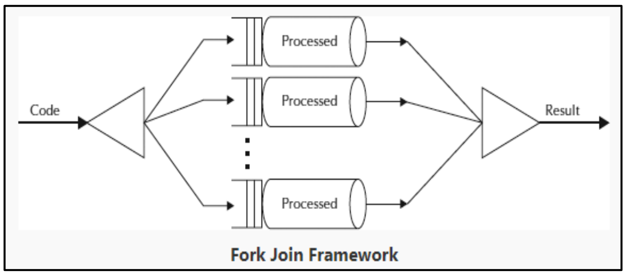


Figure2. How the ForkJoinPool Works [25]

Figure 2 explains how the ForkJoinPool method works.

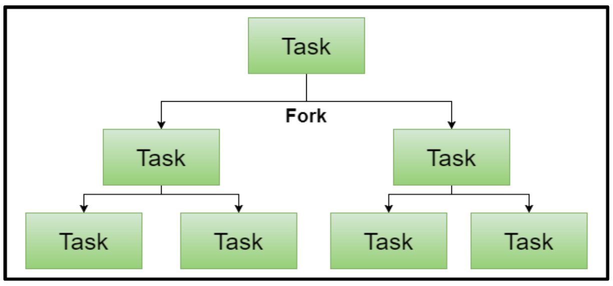


Figure3. How the Fork Works [26]

Figure 3 explains how the Fork works. By dividing each task into smaller tasks, each of these smaller tasks can be executed in parallel using a different processor or even a different thread on the processor. A task will only divide itself into a smaller task if the task is sizeable. Because if the task to be broken down is a small task, there is a possibility that the time needed for the execution process in parallel will be higher than sequential execution [26].

Figure 4 explains how the Join works. When a task splits itself into several small tasks, the tasks will wait for each other until all tasks are executed. Then the main task will combine all the results into one [26].

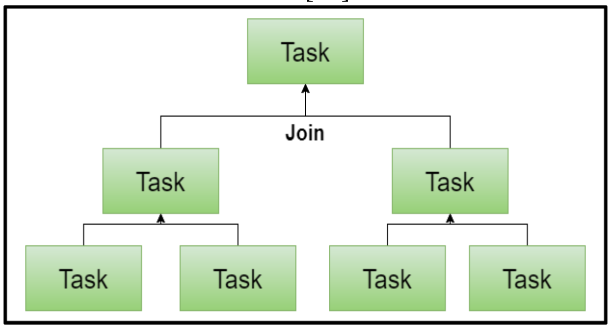


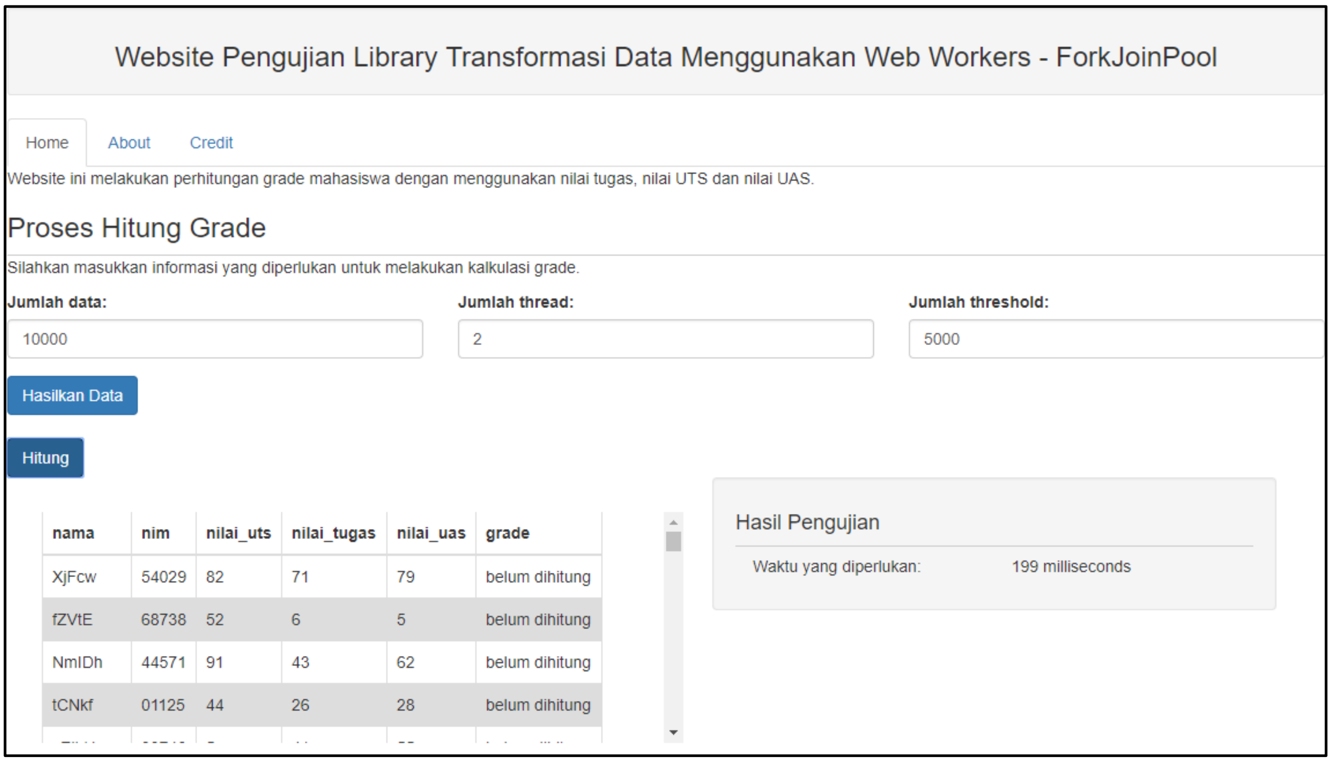
Figure4. How the Join Works [26]

III. RESULTS AND DISCUSSION

In this section, we explain the research results which had been implemented into two different web-based systems. The first one is to calculate simple grading results using a vast number of randomly generated data, while the second one is the cryptographic data website to encrypt and decrypt the vast number of data automatically. Later, the testing results will be discussed in the last subchapter.

A. Implementation Results

The appearance of the grade calculation website can be seen in Figure 5.



Website Pengujian Library Transformasi Data Menggunakan Web Workers - ForkJoinPool

Website ini melakukan perhitungan grade mahasiswa dengan menggunakan nilai tugas, nilai UTS dan nilai UAS.

Proses Hitung Grade

Silahkan masukkan informasi yang diperlukan untuk melakukan kalkulasi grade.

Jumlah data:	Jumlah thread:	Jumlah threshold:
10000	2	5000

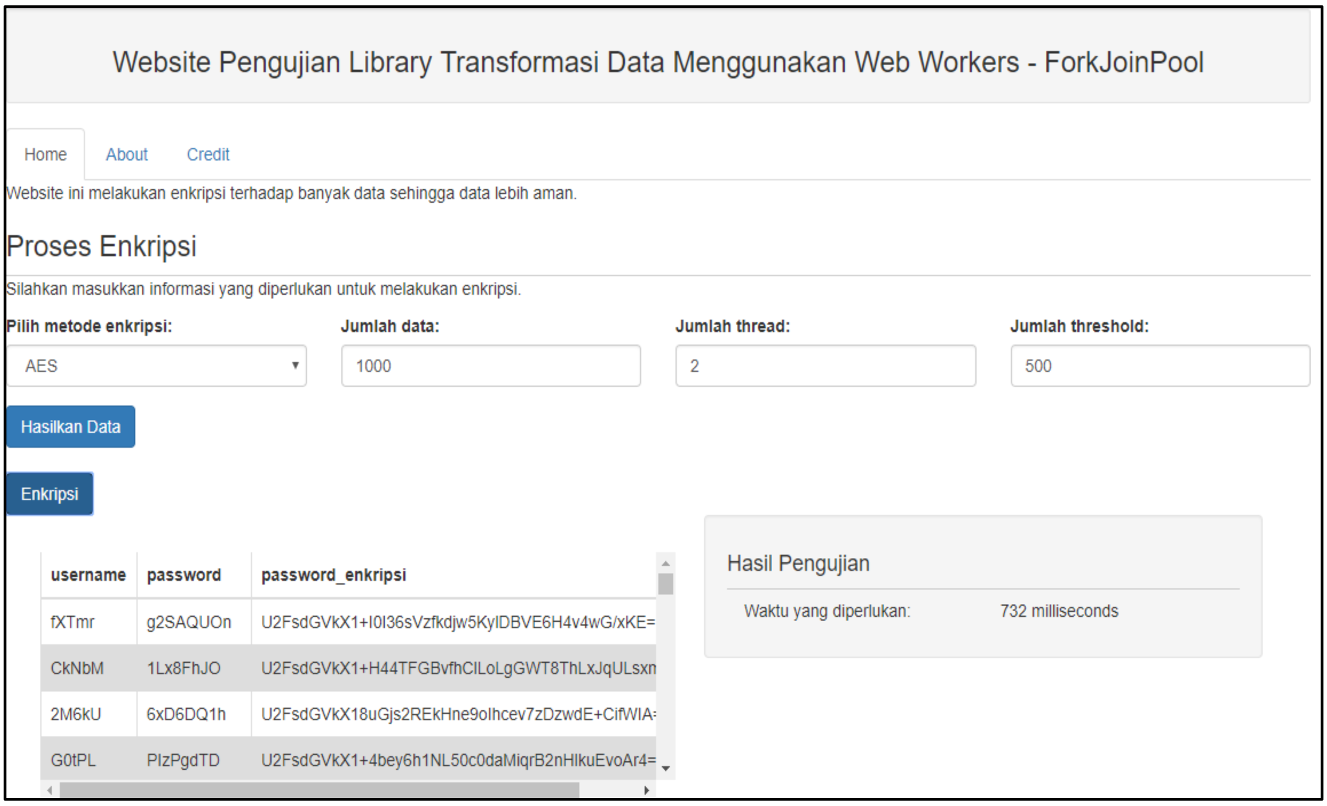
nama	nim	nilai_uts	nilai_tugas	nilai_uas	grade
XjFcw	54029	82	71	79	belum dihitung
fZVtE	68738	52	6	5	belum dihitung
NmlDh	44571	91	43	62	belum dihitung
tCNkf	01125	44	26	28	belum dihitung

Hasil Pengujian

Waktu yang diperlukan: 199 milliseconds

Figure 5. The Appearance of the Grade Calculation Website

In Figure 5, there is information about the data format used in the testing, the amount of data used, the number of threads used, and the total time needed to calculate the grade for all data. The cryptographic website display can be seen in Figure 6.


Website Pengujian Library Transformasi Data Menggunakan Web Workers - ForkJoinPool

Website ini melakukan enkripsi terhadap banyak data sehingga data lebih aman.

Proses Enkripsi

Silahkan masukkan informasi yang diperlukan untuk melakukan enkripsi.

Pilih metode enkripsi: AES
Jumlah data: 1000
Jumlah thread: 2
Jumlah threshold: 500

username	password	password_enkripsi
fXTmr	g2SAQUOn	U2FsdGVkX1+I0I36sVzfkdjw5KyIDBVE6H4v4wG/xKE=
CkNbM	1Lx8FhJO	U2FsdGVkX1+H44TFGBvfhCILoLgGWT8ThLxJqULsxn
2M6kU	6xD6DQ1h	U2FsdGVkX18uGjs2REkHne9oIhcev7zDzwdE+CifWIA
G0tPL	PIzPgdTD	U2FsdGVkX1+4bey6h1NL50c0daMiqrB2nHIkuEvoAr4=

Hasil Pengujian

Waktu yang diperlukan: 732 milliseconds

Figure 6. Cryptographic Website Display

In Figure 6, there is information about the data format used in the testing, the amount of data used, the number of threads used, the algorithm used, and the total time needed to encrypt or hash the data.

B. Testing Results

There are two websites that are used to test the library, namely grade calculating and cryptographic websites. Both websites are tested using the same scenario.

The library testing scenario is to transform data using the library that has been created where the library is implemented on both websites that have been made using one, two, four, and eight threads. The testing of the grade calculation website was carried out using Equation 1. The testing of cryptographic websites was carried out using CryptoJs library with three popular encryption methods, namely Advanced Encryption Standard (AES), Secure Hash Algorithms 384 (SHA384), and SHA512 [27, 28]. The trial was conducted to determine whether the ForkJoinPool method implementation uses Web Workers technology in JavaScript as a data transformation library affecting the speed of data transformation. As for some limitations that need to be known, namely the number of threshold variables always initialized by the amount of data divided by the number of threads and the variable tested in this study is only the speed of data transformation.

The data transformation process is performed on N data and Y thread. The data used in the trial is random data. The threshold amount is determined based on the results of N divided by Y. Trials are carried out three times for each N data group and Y thread. As an illustration, the password encryption test was carried out three times for 1,000 data and one thread. Then the results obtained from the trial were averaged three times. After that, the trial was conducted for 1,000 data and two threads three times. The trial results between using one thread and two threads are compared. If the results obtained show a non-significant difference, then the amount of data is changed to 10,000 data. Therefore, the amount of data used is adaptive, depending on the results of previous tests. The parallelism that is done is Coarse-grain parallelism to the amount of data that is processed. After all trials have been carried out, the results of the trial recapitulation are carried out.

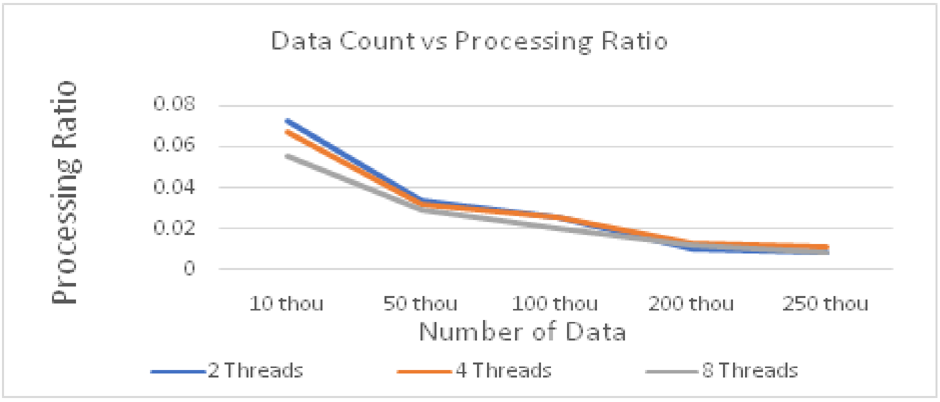


Figure7. Relation of Data Amount with Speed Ratio on Grade Calculation Website

Figure 7 is a line graph that shows the relationship between the amount of data and speed on the grade calculation website. The speed ratio in question is the speed ratio between N threads to 1 thread where the amount of data used is appropriate on the graph. From Figure 7, it can be seen that the more data used during the trial, the lower the speed ratio between N threads to 1 thread.

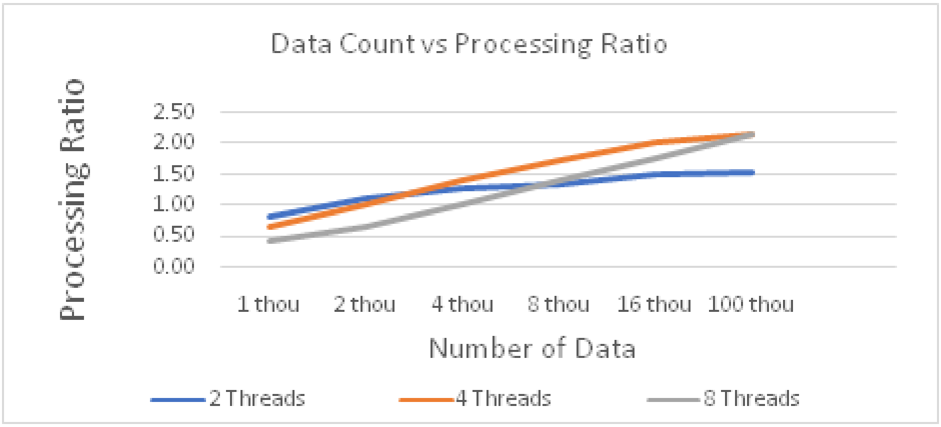


Figure 8. Relation of Data Amount with Speed Ratio on Cryptographic Website using AES Algorithm

Figure 8 is a line graph showing the relationship between the amount of data and speed on a cryptographic website with the AES algorithm. The speed ratio in question is the speed ratio between N threads to 1 thread where the amount of data used is appropriate on the graph. From Figure 8, it can be seen that the more data used during the trial, the higher the speed ratio between N threads to 1 thread.

IV. CONCLUSION

Based on the research conducted, it can be concluded that the ForkJoinPool method has been successfully implemented using Web Workers technology in JavaScript as a data transformation library. This system is created using HTML, CSS, and JavaScript languages. There are two websites built and used for testing, namely grade calculation website and cryptographic website. The parallelism that is done is Coarse-grain parallelism to the amount of data that is processed.

Based on the results of testing the data transformation library on two testing websites using randomly generated data, it can be concluded that the implementation of ForkJoinPool method using Web Workers technology in JavaScript as a library of data transformation affecting the speed of data transformation. If the processing time of the transformation is too low, then the use of the data transformation library created could reduce the speed of data transformation. If the processing time of a transformation is large enough, the use of a data transformation library will increase the speed of data transformation. In addition, the data transformation function run by the library is self-defined by the library users.

REFERENCES

1. S. Kvatinsky, Y. H. Nacson, Y. Etsion, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-Based Multithreading," *IEEE Computer Architecture Letters*, vol. 13, no. 1, pp. 41-44, 2013.
2. M. Endres and W. Kiebling, "Parallel Skyline Computation Exploiting the Lattice Structure," *Journal of Database Management (JDM)*, vol. 26, no. 4, pp. 18-43, 2015.
3. M. Wenzel and C. Meinel, "Parallel Network Data Processing in Client Side JavaScript Applications," in *Proceedings of 2015 International Conference on Collaboration Technologies and Systems (CTS)*, Atlanta, GA, USA, 2015.
4. R. Gravelle. *Introducing HTML 5 Web Workers: Bringin Multi-threading to JavaScript*. [Online]. Available: https://www.htmlgoodies.com/html5/tutorials/introducing-html-5-web-workers-bringing-multi-threading-to-javascript.html.
5. S. Okamoto and M. Kohana, "Load distribution by using Web Workers for a real-time web application," *International Journal of Web Information Systems*, vol. 7, no. 4, pp. 381-395, 2011.

6. Aminudin and M. Alwi, “Analisa Multithreading pada Sistem Rekomendasi Menggunakan Metode Collaborative Filtering dengan Apache Mahout,” *Techno.com: Jurnal Teknologi Informasi*, vol. 17, no. 1, pp. 1-11, 2018.
7. R. Ahmed, Md S. Islam, J. Uddin, “Optimizing Apple Lossless Audio Codec Algorithm using NVIDIA CUDA Architecture,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 1, pp. 70-75, 2018.
8. D. Grossman. *Beginner's Introduction to Java's ForkJoin Framework.* [Online]. Available: http://homes.cs.washington.edu/~djg/teachingMaterials/spac/grossmanSPAC_forkJoinFramework.html.
9. I-C. Wu and H. T. Kung, “Communication complexity for parallel divide-and-conquer,” in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS)*, San Juan, Puerto Rico, USA, 1991, pp. 151-162.
10. U. A. Acar, G. E. Blelloch, and R. D. Blumofe, “The Data Locality of Work Stealing,” in *Proceedings of the twelfth annual ACM symposium on Parallel algorithms and Architectures*, Bar Harbor, Maine, USA. 2000, pp. 1-12.
11. J. Ponge. *Fork and Join: Java Can Excel at Painless Parallel Programming Too.* [Online]. Available: http://www.oracle.com/technetwork/articles/java/fork-join-422606.html.
12. V. Beal. *Library*. [Online]. Available: http://www.webopedia.com/TERM/L/library.html.
13. M. Rouse. *Library*. [Online]. Available: http://searchsqlserver.techtarget.com/definition/library.
14. D. Janssen and C. Janssen. *Multithreading*. [Online]. Available: https://www.techopedia.com/definition/24297/multithreading-computer-architecture.
15. B. Barney. *Introduction to Parallel Computing*. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/#DesignGranularity.
16. Y. Pan, J. White, and Y. Sun, “Assessing the Threat of Web Worker Distributed Attacks,” in *Proceedings of IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, PA, USA. 2016.
17. Y. Watanabe, S. Okamoto, M. Kohana, M. Kamada, and T. Yonekura, “A Parallelization of Interactive Animation Software with Web Workers,” in *Proceedings of 16th International Conference on Network-Based Information Systems (NBiS)*, Gwangju, South Korea. 2013.
18. O. Ndemo and Z. Ndemo, “Secondary School Students' Errors and Misconceptions in Learning Algebra,” *Journal of Education and Learning (EduLearn)*, vol. 12, no. 4, pp. 690-701, 2018.
19. J. Long. *Transducers.js: A JavaScript Library for Transformation of Data*. [Online]. Available: http://jlongster.com/Transducers.js--A-JavaScript-Library-for-Transformation-of-Data.
20. Septeriyan. *Operasi Matematika Dasar*. [Online]. Available: https://independent.academia.edu/SepteriyanFs.
21. H. Refsnes, S. Refsnes, and J. E. Refsnes. *HTML5 Web Workers.* [Online]. Available: https://www.w3schools.com/html/html5_webworkers.asp.
22. M. Fidler and Y. Jiang, “Non-Asymptotic Delay Bounds for (k,l) Fork-Join Systems and Multi-Stage Fork-Join Networks,” in *Proceedings of IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications,* San Francisco, CA, USA. 2016.
23. C. Maia, P. M. Yomsi, L. Nogueira, and L. M. Pinho, “Semi-Partitioned Scheduling of Fork-Join Tasks using Work-Stealing,” *in Proceedings of IEEE 13th International Conference on Embedded and Ubiquitous Computing (EUC)*, Porto, Portugal. 2015.
24. R. Stewart and J. Singer. *Comparing Fork/Join and MapReduce*. [Online]. Available: https://researchget.net/publication/267958333_Comparing_ForkJoin_and_MapReduce.
25. L. Gupta. *Fork/Join Framework Tutorial: ForkJoinPool Example*. [Online]. Available: https://howtodoinjava.com/java7/forkjoin-framework-tutorial-forkjoinpool-example/.
26. J. Jenkov. *Java Fork and Join using ForkJoinPool*. [Online]. Available: http://tutorials.jenkov.com/java-util-concurrent/java-fork-and-join-forkjoinpool.html.
27. A. R. Hadi and M. I. Prasetiyowati, “Rancang Bangun Aplikasi Transport Booking Berbasis Android dengan Teknik Enkripsi Advanced Encryption Standard (Studi Kasus: PT Indodev Niaga Internet),” *ULTIMATICS*, vol. 4, no. 2, pp. 16-21, 2012.
28. K. Kusnardi and D. Gunawan, “Guillou-Quisquater Protocol for User Authentication based on Zero Knowledge Proof,” *TELKOMNIKA*, vol. 17, no. 2, pp. 826-834, 2019.

AUTHORS PROFILE

Drevendy Harianto had graduated from Universitas Multimedia Nusantara in 2018 and received his Bachelor's degree in Computer Science. He is one of the best graduates from IF Department and has actively participated in many events during his study at UMN.

Seng Hansun had finished his Bachelor and Master's degree from Universitas Gadjah Mada, majoring Mathematics and Computer Science program. Since 2011, he has been a lecturer and researcher at Universitas Multimedia Nusantara and published more than 90 papers both nationally and internationally. His research interests mainly in time series analysis and machine learning domain where he has successfully granted some research grants from the government and UMN institution.

Andre Rusli lives in Tangerang, Indonesia. He received his Bachelor Degree in Computer Science (S.Kom) from Universitas Multimedia Nusantara and Master of Science from Tokyo Denki University, Japan. Then, he began his career in university as Assistant Lecturer, until he came back from his master study and became Lecturer in the Informatics Department, Universitas Multimedia Nusantara. His research interests are software engineering, mobile technology and application development, and web development.