Matrix Multiplication of Big Data Using MapReduce: A Review

Mais Haj Qasem, Alaa Abu Sarhan, Raneem Qaddoura, and Basel A. Mahafzah
Computer Science Department
The University of Jordan
Amman, Jordan

***Abstract*— One popular application for big data is matrix multiplication, which has been solved using many approaches. Recently, researchers have applied MapReduce as a new approach to solve this problem. In this paper, we provide a review for matrix multiplication of big data using MapReduce. This review includes the techniques for solving matrix multiplication using MapReduce, the time-complexity and the number of mappers needed for each technique. Moreover, this review provides the number of articles published between the year 2010 and 2016 for each of the following three topics: Big Data, Matrix Multiplication, and MapReduce.**

Keywords — Big Data; MapReduce; Matrix Multiplication

I. INTRODUCTION

Big data analysis is the process of examining and evaluating large and varied data sets. Large size of data is continually generated every day and is useful for many applications including social networks, news feeds, business and marketing, and cloud services. In order to extract useful information from these large size of data, machine learning algorithms or algebraic analysis are performed.

Matrix multiplication has many related real-life applications, which is a fundamental operation in linear algebra. It became a problem when matrices are huge and considered as "big data". Recently, researchers have found many applications for matrices due to the extensive use of personal computers, which increased the use of matrices in a wide variety of applications, such as economics, engineering, statistics, and other sciences [1].

A massive amount of computation is being involved in matrix multiplication especially when it is considered as big data, this encouraged researchers to investigate computational problems thoroughly to enhance the efficiency of the implemented algorithms for matrix multiplication. Hence, over the years, several parallel and distributed systems for matrix multiplication problem have been proposed to reduce the communication and computation time [2-4].

Parallel and distributed algorithms over various interconnection networks; such as Chained-Cubic Tree (CCT), Hyper Hexa-Cell (HHC), Optical CCT, and OTIS HHC [5-8], divide large problems into smaller sub-problems and assign each of them to different processors, then all processors run in parallel. This makes the problem more time feasible. MapReduce is among these computing systems [9].

MapReduce is a parallel approach that consist of two sequential tasks which are Map and Reduce tasks. These tasks are implemented with several subtasks. There are many applications using MapReduce; such as, MapReduce with K-means for remote-sensing image clustering [10], MapReduce with decision tree for classification [10], and MapReduce with expectation maximization for text filtering [11]. MapReduce has also been used in real-time systems [12] and for job scheduling [13].

The rest of the paper is organized as follows. Section 2 presents big data analysis and research trends and a brief summary about Matrix Multiplication and MapReduce. Section 3 reviews work related to using the MapReduce in solving the matrix multiplication problem and compares between the related works. Section 4 summarizes and concludes the paper.

II. BIG DATA ANALYSIS AND RESEARCH TRENDS

Big data is a hot research topic which has many applications where complex and huge data should be analyzed. Number of published articles in this topic is shown in Table I and illustrated in Fig. 1. The research in this topic is increasing as it is used almost anywhere these days in news articles, professional magazines, and social networks like tweets, YouTube videos, and blog discussions. Google scholar is used to extract number of articles published for each year using the query string "Big Data" as exact search term. As shown from the figure, number of published articles is significantly increasing from 2010 to 2015 and an insignificant decrease is recognized in 2016.

TABLE I. BIG DATA PUBLISHED ARTICLES

Year	No. of Articles
2010	2,520
2011	4,000
2012	11,200
2013	27,900
2014	47,500
2015	67,300
2016	56,800

978-1-5386-1986-5/17/$31.00 ©2017 IEEE

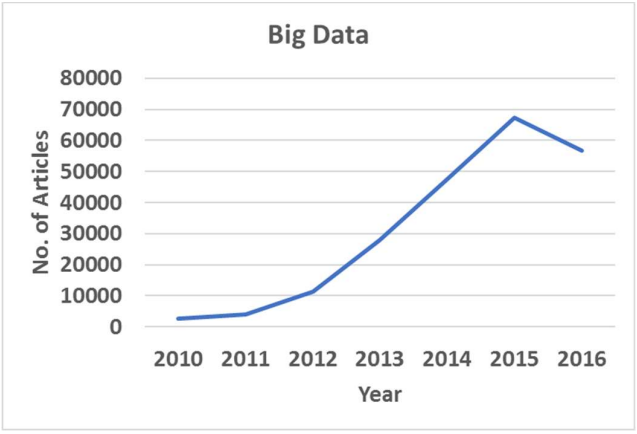

Fig. 1. Big data published articles.

One of the most important factors discussed in big data is the speed in which it is analyzed which is used for measuring the efficiency of the algorithms used, especially when big data are stored in cloud platforms. One of the big challenges facing big data analysis is multiplying matrices which is implemented using many approaches and frameworks like MapReduce. The following sub sections discuss Matrix Multiplication and MapReduce.

A. Matrix Multiplication in Big Data

Many problems are solved using matrix multiplication as it is an essential operation in linear algebra. Thus, the efficiency of the matrix multiplication algorithms can be enhanced by investigating these problems. Fig. 2 shows how matrices are multiplied to form the resulting matrix. In matrix multiplication, number of columns of the first matrix is the same as number of rows of the second matrix, where the sizes of the first matrix and the second matrix are n × m and m × q, respectively.

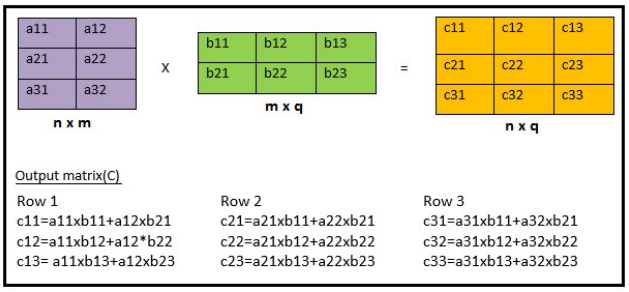

Fig. 2. Matrix multiplication operations.

The time complexity of the algorithm is $O(n^3)$, which requires to locate every element of the arrays that are multiplied. Better approaches have been proposed over the years with less time complexity than the brute-force algorithm; such as, MapReduce [9].

The research in this topic is also increasing as shown in Table II and illustrated in Fig. 3. Google scholar is used to extract number of articles published for each year using the query string "Matrix Multiplication" as exact search term. As shown from the figure, number of published articles is increasing from 2010 to 2016.

TABLE II. MATRIX MULTIPLICATION PUBLISHED ARTICLES

Year	No. of Articles
2010	7,340
2011	7,720
2012	9,250
2013	9,460
2014	9,670
2015	9,850
2016	10,600

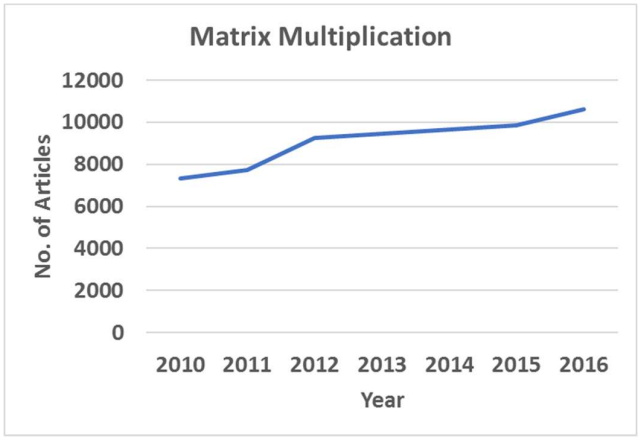

Fig. 3. Matrix multiplication published articles.

B. Role of MapReduce in Big Data

Parallel computation has been largely used for matrix multiplication which is replaced recently with MapReduce [14]. MapReduce is a framework for big data in parallel distributed environments. It consists of two sequential tasks; Map and Reduce:

- Map: takes a set of data and change it to another set of data so that each processor work on different set of data which is formed using a key-value pairs.
- Reduce: combine identical key-value pairs to form the intended output. This should always start after the map task.

The advantage of the map and reduce tasks that a program can execute on multiple processors in parallel [15]. Hadoop is an implementation of MapReduce and is an open source java-based platform developed by Google [16], but includes an additional task called "shuffle" which sorts the data so that data with identical key is located to the same processor. Hadoop architecture is illustrated in Fig. 4.

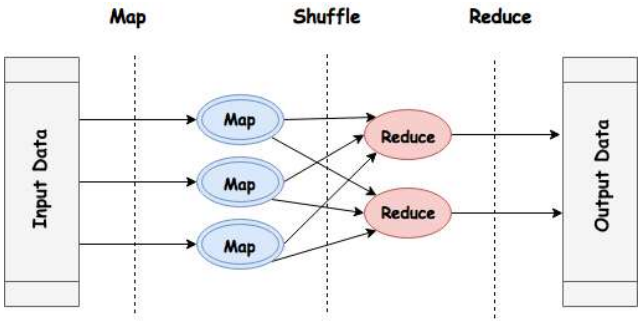

Fig. 4. Hadoop architecture.

Hadoop distributes the input datasets to the available processors which are called mappers. Each mapper implements the map task and sends its output to be shuffled and then sent to the reducer which combines all the mappers' output to form the final output.

TABLE III. MAPREDUCE PUBLISHED ARTICLES

Year	No. of Articles
2010	3380
2011	4990
2012	8040
2013	10700
2014	13300
2015	15300
2016	15100

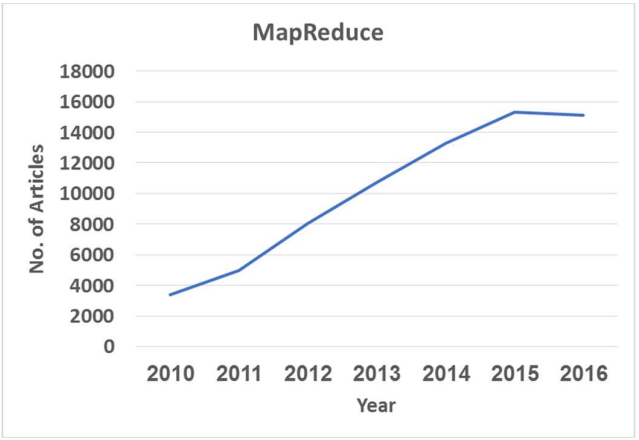

Fig. 5. MapReduce published articles.

The research in this topic is also increasing as shown in Table III and illustrated in Fig. 5. Google scholar is used to extract number of articles published for each year using the query string "MapReduce" as exact search term. As shown from the figure, number of published articles is increasing from 2010 to 2016. Qaddoura

III. MATRIX MULTIPLICATION USING MAPREDUCE

There are a lot of applications that use matrix multiplication in which the matrices are considered big. Thus, finding efficient matrix multiplication algorithm is a popular research topic. Time and cost are the main challenges facing this problem in which several algorithms were proposed in the literature in order to solve this problem [2, 17].

Using the sequential algorithm as shown previously in Fig. 2 takes too much space and time. That is why parallel algorithms were introduced for this type of problems. Ballard [18] proposed a new parallel algorithm that is based on Strassen's fast matrix multiplication and minimizes communication. Choi [19] presented Parallel Universal Matrix Multiplication Algorithm on distributed memory concurrent computers. Agarwal [20] proposed a scheme for matrix-matrix multiplication on a distributed-memory parallel computer.

Recently MapReduce has been used instead of traditional parallel-based algorithms, where the number of research articles combining the matrix multiplication with the MapReduce is only 4 articles that we will discuss and compare through this paper.

MapReduce is a parallel framework for big data, which contains two jobs when is applied on matrix multiplication:

- First job: the reduce task is inactive, while the map task is simply used to read the input file and create a pair of elements for multiplication.
- Second job: the map task implements the multiplication independently for each pair of elements, while reduce job combines the results for each output element.

Operations in each job of MapReduce are presented in Table IV, the first job responsible for reading the input elements from the input file and the other job does the multiplication and combination. This schema is called element-to-element technique since each mapper implements element by element multiplication, this technique is shown in Fig. 6.

TABLE IV. ELEMENT BY ELEMENT OPERATION

	Job 1: Map	Job2: Map	Job 2: Reduce
Input	Files	$\langle a_{ij}, b_{jk} \rangle$	< key, [a_{ij}* b_{jk}]… [a_{im}* b_{mk}] >
Output	$\langle a_{ij}, b_{jk} \rangle$	‹key, a_{ij}* b_{jk}›	< key, a_{ij}* b_{jk} +… + a_{im}* b_{mk}>

n: number of rows for the first matrix, m: number of columns for the first matrix or number of rows for the second matrix, q: number of columns for the second matrix

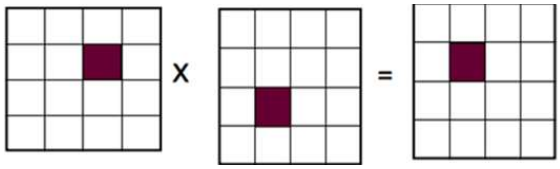

Fig. 6. Element by element schema matrix multiplication.

In order to reduce the overall computational and overcome the disadvantage of the element-by-element technique, a blocking technique has been used. Sun and Rishe [21] proposed blocking technique which is MapReduce matrix factorization

technique, in order to enhance the efficiency of matrix multiplication.

In this technique, also two jobs were used to complete the multiplication process. This technique decomposes first matrix into row vectors while it composes the second matrix into column vectors, as shown in Fig. 7. Using this technique, the communication overhead and memory utilization is decreased, and the computation process for each map is increased. Operations in each job of MapReduce are presented in Table V.

TABLE V. ROW BY COLUMN OPERATION

	Job 1: Map	**Job2: Map**	**Job 2: Reduce**
Input	Files	‹a_ij…a_im, b_(jk)…b_(mk)›	‹key, [a_ij*b_(jk)]… [a_im*b_(mk)] ›
Output	‹$\mathbf{a_{ij}}$ ... $\mathbf{a_{im}}$, $\mathbf{b_{jk}}$ ... $\mathbf{b_{mk}}$ ›	‹key, [$\mathbf{a_{ij}}$* $\mathbf{b_{jk}}$] ... ... [$\mathbf{a_{im}}$ * $\mathbf{b_{mk}}$] ›	‹key, [$\mathbf{a_{ij}}$*$\mathbf{b_{jk}}$] + ⋯ + [$\mathbf{a_{im}}$ * $\mathbf{b_{mk}}$] ›

n: number of rows for the first matrix, m: number of columns for the first matrix or number of rows for the second matrix, q: number of columns for the second matrix

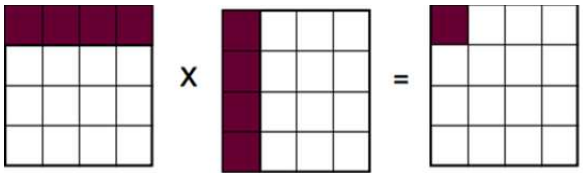

Fig. 7. Row by column schema matrix multiplication.

Zheng et al. [22] presented new two techniques for matrix multiplication which are element-by-row, and column-by-row, this is done by decomposing the first matrix into element or columns instead of rows and the second matrix also into rows instead of columns, as shown in Figs. 8 and 9. Tables VI and VII show the map and reduce processes for these two techniques.

TABLE VI. ELEMENT BY ROW OPERATION

	Job 1: Map	**Job2: Map**	**Job 2: Reduce**
Input	Files	‹$\boldsymbol{a_{ij}}$,$\boldsymbol{b_{jk}}$ ... $\boldsymbol{b_{jq}}$›	‹key, [$\boldsymbol{a_{ij}}$*$\boldsymbol{b_{jk}}$] ... [$\boldsymbol{a_{ij}}$ * $\boldsymbol{b_{jq}}$] ›
Output	‹$\boldsymbol{a_{ij}}$,$\boldsymbol{b_{jk}}$ ... $\boldsymbol{b_{jq}}$ ›	‹key, [$\boldsymbol{a_{ij}}$*$\boldsymbol{b_{jk}}$] ... [$\boldsymbol{a_{ij}}$ * $\boldsymbol{b_{jq}}$] ›	‹key,$\boldsymbol{a_{ij}}$*$\boldsymbol{b_{jk}}$ + ⋯ + $\boldsymbol{a_{im}}$ * $\boldsymbol{b_{mk}}$ ›

n: number of rows for the first matrix, m: number of columns for the first matrix or number of rows for the second matrix, q: number of columns for the second matrix

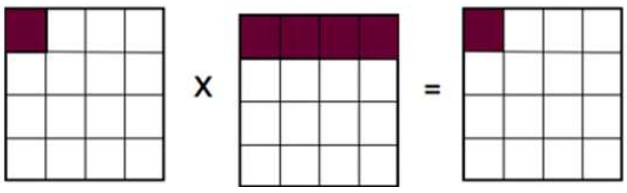

Fig. 8. Element by row schema matrix multiplication.

TABLE VII. COLUMN BY ROW OPERATION

	Job 1: Map	**Job2: Map**	**Job 2: Reduce**
Input	Files	‹$\boldsymbol{a_{ij}}$ ... $\boldsymbol{a_{nj}}$, $\boldsymbol{b_{jk}}$ ... $\boldsymbol{b_{jq}}$ ›	‹key, [$\boldsymbol{a_{ij}}$*$\boldsymbol{b_{jk}}$] ... [$\boldsymbol{a_{nj}}$ * $\boldsymbol{b_{jq}}$] ›
Output	‹$\boldsymbol{a_{ij}}$ ... $\boldsymbol{a_{nj}}$, $\boldsymbol{b_{jk}}$ ... $\boldsymbol{b_{jq}}$ ›	‹key, [$\boldsymbol{a_{ij}}$*$\boldsymbol{b_{jk}}$] ... [$\boldsymbol{a_{nj}}$ * $\boldsymbol{b_{jq}}$] ›	‹key, [$\boldsymbol{a_{ij}}$*$\boldsymbol{b_{jk}}$] + ⋯ + [$\boldsymbol{a_{im}}$ * $\boldsymbol{b_{mk}}$] ›

n: number of rows for the first matrix, m: number of columns for the first matrix or number of rows for the second matrix, q: number of columns for the second matrix

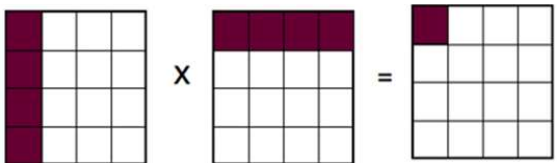

Fig. 9. Column by row schema matrix multiplication.

Deng and We [23] compared between the element-to-row and row-to-column matrix multiplication which are considered as block-based technique. The experiments performed show that the element-to-row technique runs faster than row-to-column technique, and these techniques are faster than element-to-element technique. Deng and Wu [23] modified the reading process in Hadoop by reading the input file in preprocessing rather than reading it in map, this will lead to minimizing the number of MapReduce jobs to one, it also reduces the overall computation and reduce the memory consumption. This preprocessing stage has been implemented in the HAMA project [24] for the same purpose.

In order to enhance the efficiency of matrix multiplication in MapReduce, Kadhum et al. [25] proposed new technique that balances between the processing overhead and the I/O overhead by using a balanced number of mappers so that they are not too many which reduce the I/O overhead, and not too few which reduce the processing overhead. Their technique implements matrix multiplication as element-to-block technique, as shown in Figs. 10 and 11. In the element-by-row-block scheme, the second matrix is divided into row-based blocks. In the row-block by column-block scheme the first matrix is divided into row-based blocks and the second matrix is divided into column-based blocks.

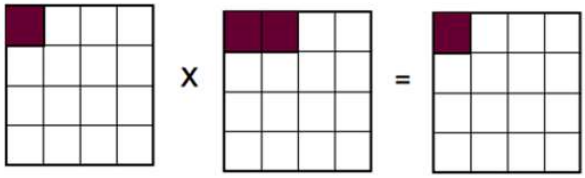

Fig. 10. Element by row-block schema matrix multiplication.

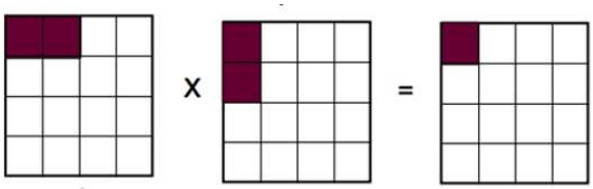

Fig. 11. Row-block by column-block schema matrix multiplication.

We compare all the pervious augments strategies for matrix multiplication in terms of time complexity and number of mappers, as listed in Table VIII. We assume that the matrices are square (i.e., $n \times n$ matrices), and L block size for block partition.

TABLE VIII. COMPARISON BETWEEN STRATEGIES OF MATRIX MULTIPLICATION

Matrix Multiplication Strategies	Time Complexity	Number of Mappers
Element-by-Element	O(log$n$)	n^3
Column-by-Row	O(n^2)	n
Element-by-Row	O(n)	n^2
Row-by-Column	O(n)	n^2
Element-by-Column-Block	O(n/L)	$n^2 \times L$
Column-Block-by-Row-Block	O(n/L)	$n^2 \times L$

For example, in element-by-element matrix multiplication, n mappers collaborate in computing each element of output matrix. Thus, n^3 mappers are needed, as shown in Table VIII. Regarding time complexity, Fig. 12 shows task-interaction graph for element-by-element matrix multiplication for matrices of size 4×4. Number of mappers for each level of the graph equals the number of mappers in the previous level divided by 2. Thus, the time complexity for element-by-element matrix multiplication is O(log n), which is the lower bound for parallel matrix multiplication; this means that this technique is consider as the most efficient technique in terms of time complexity, as shown in Table VIII.

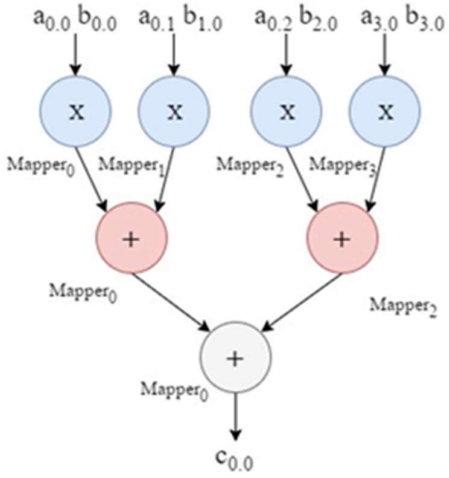

Fig. 12. Task-interaction graph for element-by-element matrix multiplication.

It is also observed from Table VIII that the time complexity of column-by-row technique is O(n^2); that is n columns multiplied by n rows which makes it the least feasible one in terms of time complexity, but not in terms of the number of mappers, where each mapper holds one column of elements for the first matrix and one row of elements for the second matrix. Thus, n mappers would be needed.

The element-by-column-block and column-block-by-row-block strategies compromise between the time complexity of the algorithm, which is O(n/L), and the number of mappers, which is $n^2 \times L$, which might be the most feasible technique for some applications.

IV. CONCLUSION

In this paper, we provide statistics of the number of articles published between the year 2010 and 2016 for Big Data, Matrix Multiplication, and MapReduce using the google scholar search engine. We observed that articles are increasingly published for these three areas of research, while combining them together as one research area is still a recent research area, where only four papers are presented; which have been discussed through this paper.

We reviewed all the techniques used by MapReduce to solve the problem of multiplying huge matrices and we compared between the presented techniques in terms of time complexity and number of needed mappers. We concluded that column-by-row technique having a time complexity of O(n) and n number of mappers is probably the best technique, while element-by-column-block and column-block-by-row-block are moderately acceptable ones, which compromises between the time complexity of the algorithm and the number of mappers. The element-by-element technique is the worst one in terms of number of mappers.

REFERENCES

[1] L. Xiufeng, N. Iftikhar, and X. Xie, "Survey of real-time processing systems for big data," In: Proceedings of the 18th International Database Engineering & Applications Symposium, ACM, 2014.

[2] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," Journal of Symbolic Computation, vol. 9, pp. 251-280, 1990.

[3] A. Grama, A. Gupta, G. Karyp, and G. Kumar, Introduction to Parallel Computing, Addison Wesley, USA, 2003.

[4] W. Cohen and B. Mahafzah, "Statistical analysis of message passing programs to guide computer design," In: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences (HICSS 1998), IEEE Computer Society, vol. 7, pp. 544-553, Kohala Coast, Hawaii, USA, January 1998.

[5] M. Abdullah, E. Abuelrub, and B. Mahafzah, "The chained-cubic tree interconnection network," International Arab Journal of Information Technology, vol. 8, pp. 334-343, 2011.

[6] B. Mahafzah and I. Al-Zoubi "Broadcast communication cperations for hyper hexa-cell interconnection network" Telecommunication Systems, May 2017. DOI: 10.1007/s11235-017-0322-3.

[7] B. Mahafzah, M. Alshraideh, T. Abu-Kabeer, E. Ahmad, and N. Hamad, "The optical chained-cubic tree interconnection network: Topological structure and properties," Computers & Electrical Engineering, vol. 38, pp. 330-345, March 2012.

[8] B. Mahafzah, A. Sleit, N. Hamad, E. Ahmad, T. Abu-Kabeer "The OTIS hyper hexa-cell optoelectronic architecture," Computing, vol. 94, pp. 411-432, 2012.

[9] J. Norstad, "A mapreduce algorithm for matrix multiplication," 2009. http://www.norstad.org/matrix-multiply/index.html [Accessed on May 2017].

[10] G-Q. Wu, et al. "MReC4. 5: C4. 5 ensemble classification with MapReduce," 2009 Fourth ChinaGrid Annual Conference, IEEE, 2009.

[11] J. Lin and C. Dyer, "Data-intensive text processing with MapReduce," Synthesis Lectures on Human Language Technologies, vol. 3, pp. 1-177, 2010.

[12] X. Liu, N. Iftikhar, and X. Xie, "Survey of real-time processing systems for big data," In: Proceedings of the 18th International Database Engineering & Applications Symposium, ACM, 2014.

[13] Z. Matei, et al. "Job scheduling for multi-user mapreduce clusters," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-55, 2009.

[14] U. Catalyurek and C. Aykanat, "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication," IEEE Trans. Parallel Distrib. Syst. vol. 10, pp. 673-693, 1999.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, pp. 107-113, 2008.

[16] J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," Commun. ACM, vol. 53, pp. 72-77, 2010.

[17] K. Thabet and S. Al-Ghuribi, "Matrix multiplication algorithms," Int. J. Comput. Sci. Netw. Secur. (IJCSNS), vol. 12, pp. 74-79, 2012.

[18] G. Ballard, et al. "Communication-optimal parallel algorithm for strassen's matrix multiplication," In: Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures, 2012.

[19] J. Choi, D. Walker, and J. Dongarra, "PUMMA: Parallel universal matrix multiplication algorithms on distributed memory concurrent computers," Concurrency: Practice and Experience, vol. 6, pp. 543-570, 1994.

[20] R. Agarwal, F. Gustavson, and M. Zubair, "A high-performance matrix-multiplication algorithm on a distributed-memory parallel computer, using overlapped communication," IBM Journal of Research and Development, vol. 38, pp. 673-681, 1994.

[21] Z. Sun, T. Li, and N. Rishe, "Large-scale matrix factorization using mapreduce," In: 2010 IEEE International Conference on Data Mining Workshops, 2010.

[22] J. Zheng, R. Zhu, and Y. Shen, "Sparse matrix multiplication algorithm based on MapReduce," J. Zhongkai Univ. Agric. Eng. vol. 26, pp. 1-6, 2013.

[23] S. Deng and W. Wenhua, "Efficient matrix multiplication in Hadoop," Int. J. Comput. Sci. Appl. vol. 13, pp. 93-104, 2016.

[24] S. Seo, et al. "An efficient matrix computation with the mapreduce framework," In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 721-726, November 2010.

[25] M. Kadhum, M. Qasem, A. Sleit, and A. Sharieh, "Efficient MapReduce matrix multiplication with optimized mapper set," In: Computer Science On-line Conference, pp. 186-196, 2017.