Improving Current Hadoop MapReduce Workflow and Performance

Hamoud Alshammari
Department of Computer Science
221 University Ave, University of Bridgeport
Bridgeport, CT, USA

Jeongkyu Lee
Department of Computer Science
221 University Ave, University of Bridgeport
Bridgeport, CT, USA

Hassan Bajwa
Department of Electrical Engineering
221 University Ave, University of Bridgeport
Bridgeport, CT, USA

ABSTRACT
This study proposes an improvement andimplementation of enhanced Hadoop MapReduce workflow that develop the performance of the current Hadoop MapReduce. This architecture speeds up the process of manipulating BigData by enhancing different parameters in the processing jobs. BigData needs to be divided into many datasets or blocks and distributed to many nodes within the cluster. Thus, tasks can access these blocks in parallel mode and be processed easily. However, accessing the same datasets each time the job is executed causes data overloading problem, so we developed the current MapReduce workflow to improve the performance in terms of data size that is read in the relative jobs.This work uses a bioinformatics DNA datasets to implement the solution.

Index Terms
Cloud Computing, Hadoop, bioinformatics, BigData.

1. INTRODUCTION
BigData analysis in cloud computing is considered as one of the very important topics because of the continuous increasing in data size. BigData is defined as huge datasets that cannot be processed using any traditional applications, plays a significant role in many aspects of society such as communications, retails, finance and science[1].

Bioinformatics datasets areconsidered as a BigData not only because of its size but also due to its complexity and dimensions[2]. The features that BigData has are volume, velocity and variety of the data, each of one of these features can be considered as a significant challenge and contribute of the bioinformatics data[3].

MapReduce is a popular algorithm that manipulates BigData with high efficiency, fault-tolerance, good scalability, and simple programming [4].MapReduce algorithm has two main functions map and reduce functions. Map functions work in parallel mode to process specific tasks, produce initial results, and then pass the initial results to the next step which

is the reduce functions. Reduce functions also work in parallel mode to get the map initial results and collect the results into one final result to be the only result of the whole job [5].

Different tools have been used to process BigData; one of the most popular tools is Hadoop. It has been developed using MapReduce algorithm, which divides the data to many blocks in the same size [5]. The process of representing data in Hadoop goesthroughtwo main steps.First, divides the data into many blocks,seconddistributes these blocks between nodes in the cluster Hadoop Distributed File System (HDFS)format [6, 7].Some processing limitations have been addressed in cloud computing such as network bandwidth, data security and processing costs [8, 9].

In this study, we present anenhanced Hadoop MapReduce algorithm workflow and use it for fining sequence in bioinformatics data. We built a lookup table that stores ametadatafor relative jobs in order to reduce the number of read operations during fining sequence. Compared to native Hadoop, the proposed framework reduces the number of blocks that is read during the computation. In section II we cover the importance of Hadoop to the bioinformatics data. In section III overviews the Hadoop infrastructure in cloud computing. Section IV discusses the proposed enhanced Hadoop MapReduce architecture and Workflow.We discuss the implementation and results in section V. We have the conclusion in section VI.

2. HADOOP AND BIOINFORMATICS APPLICATIONS
The limitation in the memory can cause some limitations in the data processing jobs. One of the common jobs in bioinformatics is finding sequence in the chromosomes, however, this process remains a challenge due to the increasing in compute and memory limitations [10].

Human DNA genome datasets are presented in 24 chromosomes each in a single text file. DNA chromosomes datasets are considered as a big data even it is not very huge data but because it is an unstructured and unrelated data. So, with this kind of data any process could have some complexities to be achieved with a high degree of reliability and efficiency[11].

Figure 1 shows the process of finding sequence in DNA flowchart using <key, value> format. One of the most popular sequences is a Zinc Finger, which is represented by the sequence GGGGCGGGG.

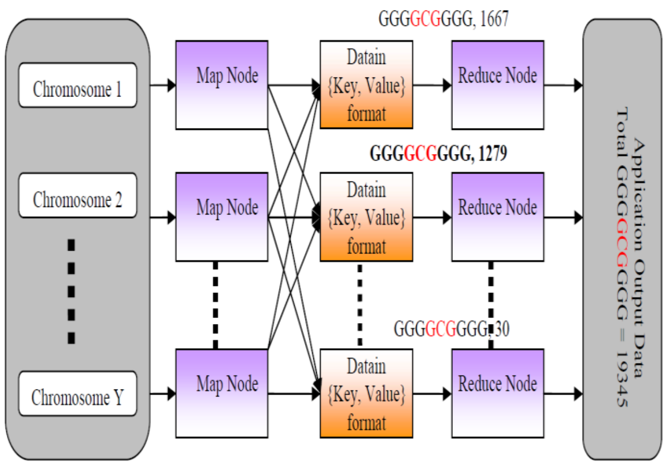

Figure 1: Sequence Detection using MapReduce

Many variables in DNA sequences can change the degree of performance of Hadoop MapReduce algorithm, such as the number of that the sequence is located in the chromosomes and the length of the sequence itself. However, there are different formats of DNA that can be used in couple applications based on the user needs. In addition, users can reformat the data of DNA in format that they need [12].

3. HADOOP CLOUD COMPUTING INFRASTRUCTURE

Hadoop infrastructure contains a support system that works aside with Hadoop, which is named by Hadoop Ecosystem, this ecosystem contains different components such as Zookeeper, HBase, Hive [13]. In addition, hadoop is divided into two main components, which are Hadoop Distributed File System and MapReduce. Figure 2, shows the component of the current Hadoop MapReduce architecture.

3.1 Hadoop Distributed File System (HDFS)

In the current Hadoop and MapReduce framework, the client sends MapReduce job to the Hadoop cluster administrator (NameNode), which is also the master of the cluster. Before sending a job to the NameNode, the data source files should be divided into 64 or 128 MB of blocks, and then uploaded to the Hadoop Distributed File System (HDFS). Data blocks are distributed among different Data Nodes within the cluster. Any new job must have the name of the data file in HDFS, the source file of MapReduce code (e.g. Java file), and the name of the file where the result will be stored in the HDFS.

The current Hadoop MapReduce architecture follows the concept of “write-onceread-many”. No changes can be made in a source file in HDFS. Multiple jobs with the same data set work independent of each other. Each job has the ability to access the data from all blocks. Iterative computations, computations that need to pass over the same data many times, are unable to utilize the native cloud computing architecture very efficiently.

Several research groups have presented locality aware solutions to address the issue of latency while reading data from DataNodes [19]. Hadoop falls short of query optimization and reliability of conventional database systems. In our research, we discovered that each time we executesame MapReduce job, it requires sameamount of time to find same results. Also, when we search for supper-sequence that contains the original sequence, which means there is no relationship between jobs.

3.2 HadoopMapReduce Algorithm

Many Hadoop MapReducejobs, especially tasks associated with the science data such as the genomic data,focusonthe similarities between the sequences, superstring and sub-sequence searches[14].These operations may apply different MapReduce jobs on the same data frequently. Hadoop workflow is shown in Figure2.

In current Hadoop architecture, client A sends a request to NameNode. In case the data is in local file system, the request includes the need to copy the data files to the Hadoop Distributed File System in DataNodes. NameNode replays the IP addressesofthe DataNodes to the client A. Then, client Aformats the data files,and sendsmultiple copies of each data block to different DataNode. These steps are from step 1 to step 5 in figure 2.

Following these steps, client A sends a MapReduce job (job1) to the JobTracker daemon with the name of the data source file that is stored in HDFS. Then, the JobTracker divides the job into many tasks and sends these tasks to all TaskTrackers who hold the blocks of the source data.Each TaskTracker executes a specific task on each block then sends the results back to the JobTracker. JobTracker collect the results and create a one final result then sends it to Client A. These steps are from step 6 to step 9 in figure 2.

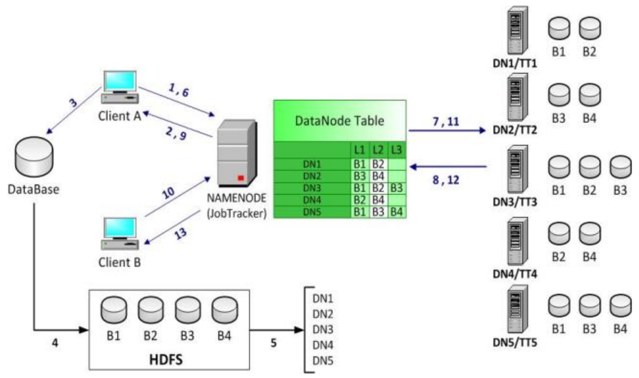

Figure 2: Current Hadoop MapReduce Architecture

If client A launch the same job again on the same datasets, then all steps above will be repeated.

As shown in Figure 2,assume that ClientA and Client B are searching for the same sequence inBigData source files. Once Client A finds the sequence, Client B willfollow the samestepsin order to find the sameresults (steps 10 to 13). Since each job is independent, clients do not share results.Repeated results may occur, and therefore, process redundancy remains a major, unsolved problem in native Hadoop infrastructure.

Figure 3 shows the flowchart workflow of the current Hadoop MapReduce algorithm, which is a simple algorithm that has no conditions unless writing the data to HDFS:

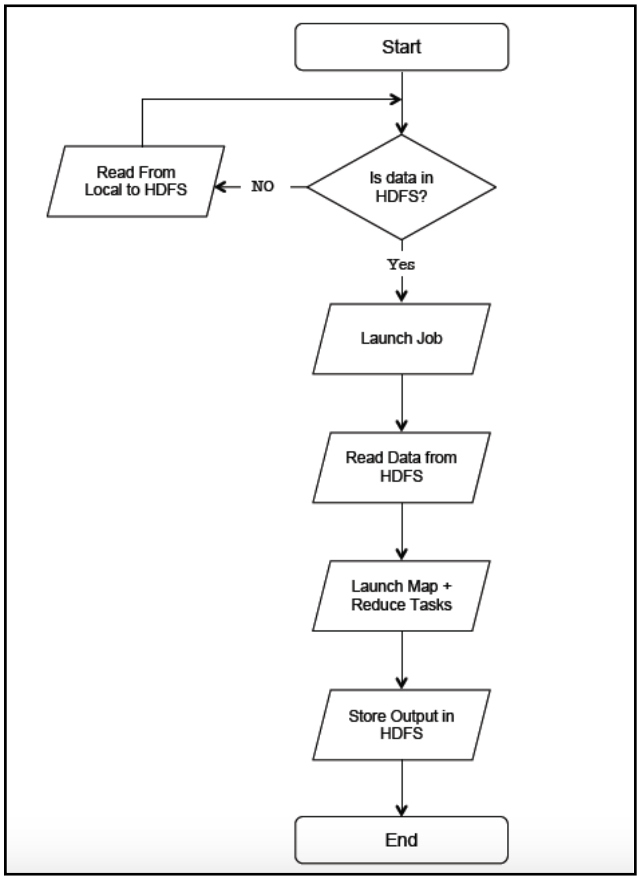

Figure 3: Current Hadoop MapReduce Algorithm Workflow

In a DNA sequence-matching task,whenthe sequence exists in a specific Block in a DataNode, a superstring sequencecontaining that sequence can only be found in the same block. Current Hadoop frameworkdoes not support caching of data. It ignores the location ofDataNodes that contain asequence, and reads data from all DataNodes for every new job.

4. ENHANCED HADOOP MAPREDUCE

In current Hadoop architecture, NameNode can recognize the location of the blocks in HDFS by DataNode IP address. NameNode is a responsible for receiving jobs from the clients,and divides the jobs into tasks and assigns these tasks to the DataNodes (TaskTrakers). The process ofrecognizing which DataNode contains the blocks with the required data gives the NameNode a capability to direct the tasks to specific DataNodes without going through the whole cluster. Proposed architecture leverages on a metadata table to identify the blocks with common features. Any job with the same common featuresshould only read the data from the specific blocks that contains the sequence.This solution eliminates the process ofreading the entire data blocks again.

In terms of hardware, network, and nodes, the enhanced Hadoop MapReduce architecture is identical to the original Hadoop. However, the enhancing is processed on the software level. We added features in the NameNode that allows it to save specific data (metadata of blocks) in a lookup table named the Common Job Blocks Table (CJBT). CJBT stores some information about the jobs and the location of the blocks that store the results of these jobs. It allows the related jobs to get some information about the location of the required sequence in the HDFS without checking the entire cluster.

Typically, a sequence is aligned through the use of dynamic programming and conventional alignment algorithms. Every CJBT contains common features. A common feature may consist of a sequence or a subsequence that is identified and updated in CJBT. In addition, the Common Feature in CJBT can be compared and updated each time a client submits a new job to Hadoop. A typical CJBT consists of three main components or columns, which are explained in Table 1:

Table1: Common Job Blocks Table components

Common Job Name	Common Feature	Block Name		
Sequence_Alignment	GGGC	B1	B2	B3
	GGGG	B5	B4	
Fining_Sequence	TTTAGCC	B3	B6	
	GCCATTAA	B1	B3	B4
	AATCCAGG	B3	B5	

Common Features store data that is shared between jobs like DNA sequence. Enhanced Hadoop architecture supports caching, which enables output (or part of output) to be written in CJBTafter the reduce step. JobTracker directs any new job with the shared common features to blocks listed in CJBT. However, feature selection should be done carefully since the response time for the jobs can increase if the common features exist in every DataNode. For example, in genomic data, regulatory sequences and protein binding sites are highly recurring sequences. Using such sequences as common features can degrade the performance of the proposed architecture.

Feature format, also, depends upon the job and the datasets. For example, in sequence alignment jobs, a nucleotide in a sequence is represented by capital letters, and a sequence is composed of several letters as shown in Table1. The way that we build the CJBT is by having a training phase that comes before launching jobs. Also, by having new jobs, we can create a new record in the CJBT.

Figure 4 shows the flowchart workflow of the enhanced Hadoop MapReduce algorithm, which contains couple testing steps searching for common features on the CJBT.

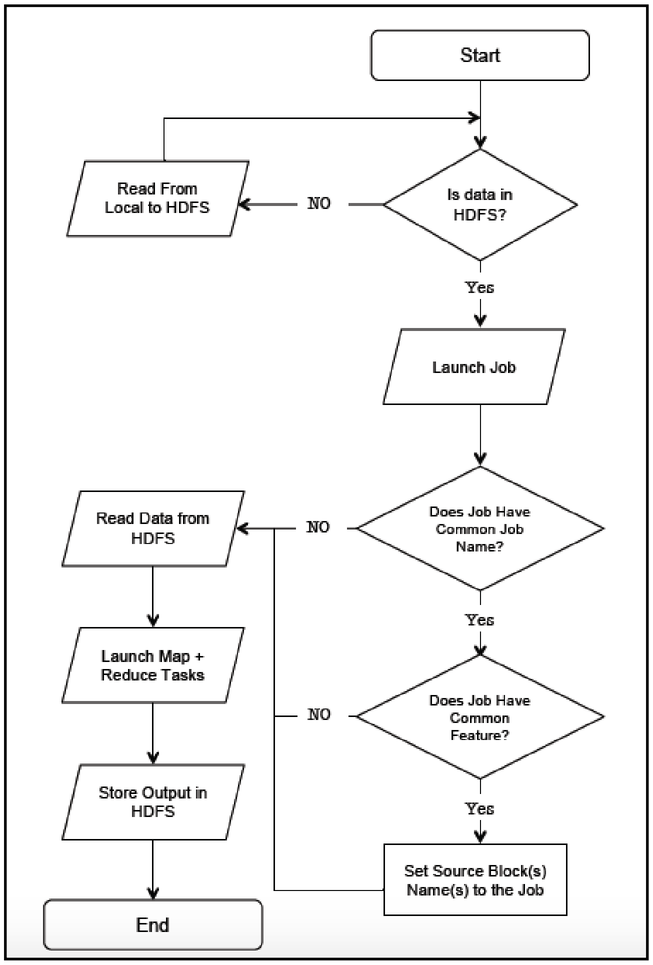
Start
Is data in HDFS?
NO
Read From Local to HDFS
Yes
Launch Job
Does Job Have Common Job Name?
NO
Read Data from HDFS
Yes
Does Job Have Common Feature?
NO
Launch Map + Reduce Tasks
Yes
Store Output in HDFS
Set Source Block(s) Name(s) to the Job
End

Figure 4: Enhanced Hadoop MapReduce Algorithm Workflow

As the same as the current Hadoop, first, we test the data is it in HDFS format or no. After launching a job, we tested either this job uses or new architecture by having a common job name from a list that we have in the user API or not. If not, it goes to be processed via a traditional way. Otherwise, we tested either it has a common feature or not. If it doesn't have a common feature, which means it has new features that are not related to any previous job, it goes to be processed via a traditional way. Otherwise it means it has a common job name and a common feature, NameNode checks the CJBT and retrieves block that contain the common feature and assigns the job to read data from these blocks.

5. IMPLEMENTATION AND RESULTS

We simulated the enhanced Hadoop MapReduce architecture and we built the CJBT using Apache HBase NoSQL database. The reason of using HBase is because its compatibility with Hadoop and its, also, works on top of Hadoop and using shell user interface. We apply that using Linux OpenSUSE as an operating system.

Table 2 contains some sequences that we use for training purposes to establish CJBT. Sq1 is a common sequence in human DNA, which is the Zinc Finger protein, and it is located on every single chromosome. One of the observations is the longest sequence requires more CPU processing time, and that is clear in Sq5.

Table2: Common features for sequence alignment

No	Symbol	Common Feature
1	Sq1	GGGGCGGGG
2	Sq2	AAGACGGTGGTAAGG
3	Sq3	CATTTCTGCTAAGA
4	Sq4	GAATGTCCTTTCTCT
5	Sq5	GATCTCAGCCAGTGTGAAA

Figures 5 and 6 present the numbers of read operations and the CPU time for the sequences presented in Table 4. The histograms in figures below demonstrate the CPU time and the number of read operations for selected queries in native and enhanced Hadoop architecture. It is observed that the proposed architecture outperforms native Hadoop in almost all cases.

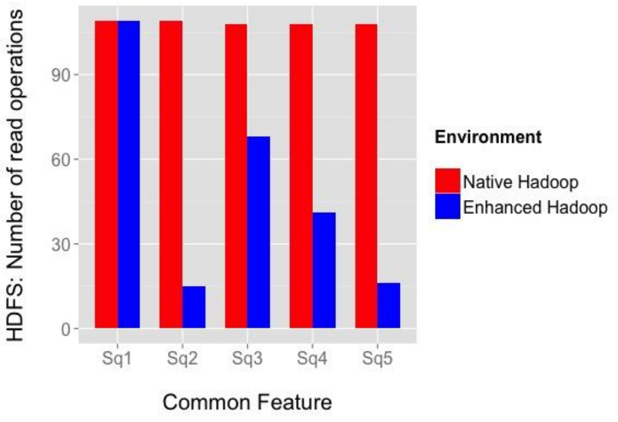

Figure5: Number of read operations of selected queries in Native Hadoop and Enhanced Hadoop.

As we see in Figure 5 one of the results that we got by applying the Enhanced Hadoop workflow. This experiment shows the improving in the performance by reducing the data size that is read to execute a job. We developed that by using the metadata of the related jobs and specify the related sections or blocks on the dataset, and select these data blocks to be the whole source of the data for that job.

There are 5 DNA sequences that we use for searching purposes on the DNA data as shown in Figure 5. The number of read operations clearly is less in the Enhanced Hadoop than that in the Native Hadoop in the first sequence, which is “AAGACGGTGGTAAGG”. In Native Hadoop it is 119 operations and in Enhanced Hadoop it is only 15 operations, and the percent of the improving is about 86% less in the number of read operations. Also, the rest of the results show the huge developing and best results comparing with native Hadoop.

The improving of the framework is related to the frequently of finding the sequence in the data. So, if the sequence is exists on all data blocks, the MapReduce job will read the all blocks again, and that is clear in the result from (Sq1) which is “GGGGCGGGG”. So, the number of read operations stays the same in Native Hadoop and Enhanced Hadoop.

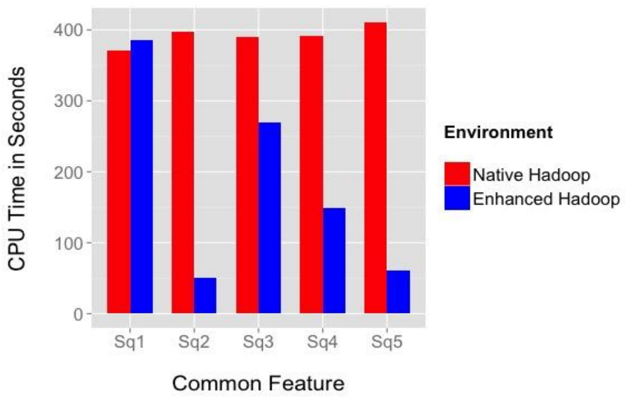

Figure6: Overall CPU performance of selected queries in native Hadoop and enhanced Hadoop.

Figure 6 shows that CPU execution time in Native Hadoop and Enhanced Hadoop. It shows excellent results in CPU execution time for the sequences that are exist in some location in the data. For example, it gives about 86% of reducing in the CPU time execution in (Sq2).

However, one of the observations is, when searching for highly conserved sequencese.g. (Sq1) in Table4, we found poor performances in the enhanced Hadoop compared to the native Hadoop. That comes from the required time for NameNode to access the CJBT and search for the required common name and common features.

6. CONCLUSION

This paper discusses the limitations of the current Hadoop MapReduce architecture and workflow. We proposed an Enhanced Hadoop MapReduce algorithm to speed up and improve the performance of Hadoop.We focus on building a lookup table to store a metadata of the jobs to use it when we run the same job again or run different jobs that are relative to the original job. This way improved the performance of current Hadoop MapReduce in different levels such as CPU processing time and number of read data operations.

Although proposed architecture improves the Hadoop MapReduce job performance, there are couple points that we need to discuss and develop as future work. For example, the size of the lookup table, analyzing the Enhanced Hadoop workflow mathematically and determine the optimum size of the common feature.

7. ACKNOWLEDGMENTS

We would like to mention about the environment that we use to develop the work, which is the Cloud Computing Cluster CCC at the University of Bridgeport. We would thank all CCC members either people or organizations.

8. REFERENCES

[1] S. Lohr, "The age of big data," *New York Times*, vol. 11, 2012.

[2] V. Marx, "Biology: The big challenges of big data," *Nature*, vol. 498, pp. 255-260, 06/13/print 2013.

[3] T. White, *Hadoop: The definitive guide*: " O'Reilly Media, Inc.", 2012.

[4] J. B. Buck, N. Watkins, J. LeFevre, K. Ioannidou, C. Maltzahn, N. Polyzotis, *et al.*, "SciHadoop: Array-based query processing in Hadoop," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, 2011, pp. 1-11.

[5] A. B. Patel, M. Birla, and U. Nair, "Addressing big data problem using Hadoop and Map Reduce," in *Engineering (NUiCONE), 2012 Nirma University International Conference on*, 2012, pp. 1-5.

[6] W. Xu, W. Luo, and N. Woodward, "Analysis and optimization of data import with hadoop," pp. 1058-1066.

[7] S. Wu, F. Li, S. Mehrotra, and B. C. Ooi, "Query optimization for massively parallel data processing," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, p. 12.

[8] L. D. Stein, "The case for cloud computing in genome informatics," *Genome Biol*, vol. 11, p. 207, 2010.

[9] M. C. Schatz, B. Langmead, and S. L. Salzberg, "Cloud computing and the DNA data race," *Nature biotechnology*, vol. 28, p. 691, 2010.

[10] P. C. Church, A. Goscinski, K. Holt, M. Inouye, A. Ghoting, K. Makarychev, *et al.*, "Design of multiple sequence alignment algorithms on parallel, distributed memory supercomputers," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, 2011, pp. 924-927.

[11] H. Alshammari, H. Bajwa, and J. Lee, "Hadoop Based Enhanced Cloud Architecture," presented at the ASEE, USA, 2014.

[12] S. Leo, F. Santoni, and G. Zanetti, "Biodoop: Bioinformatics on Hadoop, Parallel Processing Workshops, International Conference on, pp. 415-422, 2009 International Conference on Parallel Processing Workshops, 2009," 2009.

[13] A. H. Zookeeper, "http://hadoop.apache.org/zookeeper/," accessed Feb 2015.

[14] A. Matsunaga, M. Tsugawa, and J. Fortes, "CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, 2008, pp. 222-229.

9. AUTHOR'S PROFILE

Hamoud Alshammari (First Author) received a BS in Computer Information Systems from King Saud University, Saudi Arabia in 2002. He received his first MS degree in Business MBA from Yarmok University, Jordan. Then, he received the second MS degree in Computer Science from University of Bridgeport, CT-USA. He is doing his Ph.D. in Computer Science and Engineering at University of Bridgeport, CT-USA. Alshammari is doing his research in BigData and Hadoop MapReduce performance. He is also has interesting in data analysis. Mr. Alshammari is a member in Upsilon Pi Epsilon Honor Society.

Dr. Jeongkyu Lee received a B.S. from Sungkyunkwan University in Mathematic Education and an M.S. from Sogang University in Computer Science, both of Seoul, Korea in 1996 and 2001, respectively. He worked as a database administrator for seven years with companies including IBM. In fall 2002, he entered the Doctoral program in Computer Science and Engineering at the University of Texas at Arlington. Currently he is an Associate Professor of Computer Science at the University of Bridgeport. His research interest is in the multimedia database management and big data analytics. His work also includes techniques for multimedia data mining, video processing, multimedia ontology, and medical imaging.

Dr. Hassan Bajwa received his BSc degree in Electrical Engineering from NYU Polytechnic University of New York in 1998. From 1998 to 2001 he worked for Software Spectrum. He received his MS from the City College of New York in 2003, and his Doctorate in Electrical Engineering from City University of New York in 2007.Currently he is an Associate Professor of Electrical Engineering at the University of Bridgeport. His research interests include modeling and simulation of Nano-electronic architectures, low power sensor networks, flexible electronics, bioelectronics, and Bioinformatics