ACEAT-0085
FPGA-Based Accelerator Platform for K-Means Clustering Algorithm

Ching-Che Chung, Dai-Hua Lee, and Yu-Hsin Wang
Dept. of Computer Science and Information Engineering,
National Chung Cheng University, Taiwan
E-mail address: wildwolf@cs.ccu.edu.tw

Abstract
Nowadays, the smart phones, web systems, and wireless sensors are enabled to connect the internet and response to the user in real time. Therefore, we are living in the internet of things (IoT) era with big data generation. The "4Vs" characteristics of big data such as variety, volume, velocity, and value make them difficultly to be handled. The personal computer is hard to deal with big data because the capacity of memories and storage devices is not enough and the limited processing rate of the CPU. Therefore, the distributed file system and cloud computing have become popular. Both of them can compute in parallel and share the data of disks. Moreover, the hardware accelerator is suited for data-intensive computation, and the function units of hardware accelerator are processed in parallel. Graphics processing units (GPUs) and field programmable gate arrays (FPGAs) are potential hardware accelerators for these data-intensive computations. K-means clustering algorithm is one of the data mining techniques. In this paper, we implement k-means clustering algorithm to analyze the dataset. The proposed FPGA-based hardware accelerators communicate with the computer through Ethernet switch. The computer wraps data into packets then sends to the FPGAs, and it receives data after the FPGAs finishing computations. Also, the host computer is employed as the master to manage data and dispatch jobs, and the FPGAs are focused on accelerating data computation. Finally, the proposed system performance is compared with the benchmark execution time.
Keywords: Big data, K-means clustering algorithm, hardware accelerator

1. Introduction

Specific technologies such as network, the internet, server, email, and mobile are growing up that generates lots of data from devices [1]. The smart phones have launched a wave of revolution, not only the user can directly access vast amounts of smart phone applications but also enables individuals to use applications to reach mass audiences [2]. The conception of the internet of things (IoT) is connectivity of network entities embedded with devices that can exchange data to physical objects. However, IoT is evolved to the internet of anything (IoA) [3]. IoA can imagine everything as part of the network ecosystem, like the internet operating system [3].

In fact, big data is difficult to be handled, not only the amount of data is large, but also it has special properties. The properties of big data make it not easy to be handled. The "4Vs" is wide big data definition that includes variety, volume, velocity and value [4]. Variety means the structure of data has many flavors. Volume means the data are too large to analyze the data sets. Velocity means the data are generated from everywhere and anytime, the processing rate of devices should match the speed of data production. Finally, by analyzing big data valuable information can be found.

Obviously, processing big data has challenges, both hardware and software are evolved to adapt the characteristics of big data. Therefore, many software frameworks and file systems are developed to against big data issues, such as NoSQL database, Google file system, Facebook's photo storage haystack and Hadoop. NoSQL is complemented replacement relational database management systems (RDBMS). Google file system (GFS) [5] runs on hundred inexpensive machines on Linux operation system, and GFS has fault tolerance and atomic data recovery when inexpensive commodity hardware devices are failed. Facebook developed haystack [6] to store the billions small size of photos. Haystack achieves four goals: high throughput and low latency, fault-tolerant, cost-effective and simple. Hadoop [7] is an open source programming framework that enables distributed processing of data over large clusters of computing servers and numerous data storage.

Besides the development of software frameworks and distributed file systems, the computing servers also require new system capabilities. Computing demand is kept increasing and with only CPUs is dissatisfaction. Therefore, heterogeneous hardware accelerator helps to share computing loading and becomes more attracted in recent years. Graphics processing units (GPUs), and field programmable gate arrays (FPGAs) are potential hardware accelerators. The main advantages of the GPU are high memory bandwidth, many programmable cores with multi-thread execution in parallel, coding with high-level languages like CUDA, and changing functions easier than FPGAs. FPGAs have high-density arrays of logic blocks which can execute computation in parallel. A user can use Verilog or VHDL to implemented circuits in FPGA, and the vendors provide useful IPs to help developer design [8]. Many researchers present the performance for diverse applications. The computation which can process in parallel is a good fit to GPU, and applications which have many memory access times are a bad fit to GPU. FPGA is a good fit at computation low-level bit-wise operations and is a bad fit at the complex algorithm and data flow [8].

After discussion characteristics in both FPGA and GPU, we survey prior embedded hardware accelerators systems architectures. In [9], the IBM Power8 processor contains private data cache per core to improve performance. In addition, the server workloads will continue to evolve, the IBM Power8 embeds the coherent accelerator processor interface (CAPI) to support the general purpose cores for a heterogeneous computing solution with off-chip hardware accelerators. In [10], they integrate ZedBorad platform which combines ARM processor and FPGA, and Hadoop to be a new Zynq-based Hadoop cluster. It can inherit Hadoop frameworks, like the name node, the scheduler, and the HDFS. A CPU system is employed as the name node that maintains file metadata. This architecture can achieve speed up as compared to pure software approaches.

In this paper, we develop FPGA-based hardware accelerator through Ethernet switch to achieve high scalability. We implement K-means algorithm in the VC707 FPGA evaluation boards (EVBs) [11], and cooperate with the computing server and FPGAs. The computing server is employed as the master to manage data and control the status, the FPGAs focus on computation. Besides, the values of data set are single precision floating point numbers. K-means clustering algorithm is one of the clustering algorithms for data analysis, and K-means clustering algorithm has intensive computation and non-sequential working. These properties are suited for hardware accelerators.

The rest of this paper is organized as follows: The hardware accelerator architecture overview is presented in Section 2. Section 3 shows K-means clustering algorithm implemented in the FPGA. Section 4 shows the implementation and measurement results. Finally, the conclusion is given in Section 5.

2. Proposed Hardware Accelerator Architecture

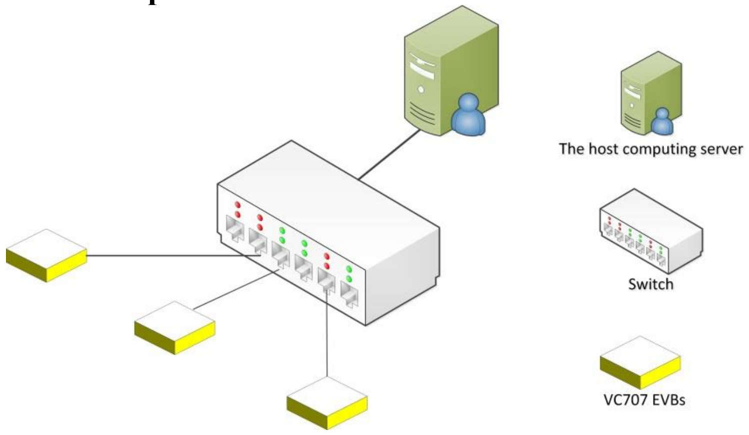

Fig. 1: The proposed FPGA-based hardware accelerator platform

The proposed accelerator platform is composed of three VC707 FPGA EVBs, as shown in Fig. 1. In addition, each VC707 EVB has the same function. The computing server wraps the data into packets and sends them to VC707 EVBs through a Gigabit Ethernet switch. Then, the workloads of the computing server can be shared in FPGA EVBs. The data flow in the VC707 EVB architecture is shown in Fig. 2. We implement three modules in the VC707 EVB including an Ethernet physical layer IP, a DDR3 memory interface controller IP, and a user core. Ethernet physical IP is used to received packets from the computing server, and return the computing results of the VC707 EVB to the computing server. The Ethernet physical layer IP module splits received packets into byte data or combines byte data into packets. The DDR3 memory interface controller IP helps the user core to communicate with the onboard 1GB DDR3 memory. The command (cmd) and address (addr) can be sent to the memory interface controller IP simultaneously when we request to the DDR3 memory. For a write request, the written data (wm_d) need to be prepared before the write request sends to memory interface controller IP. For a read request, the read data (rm_d) are output from memory interface after several cycles.

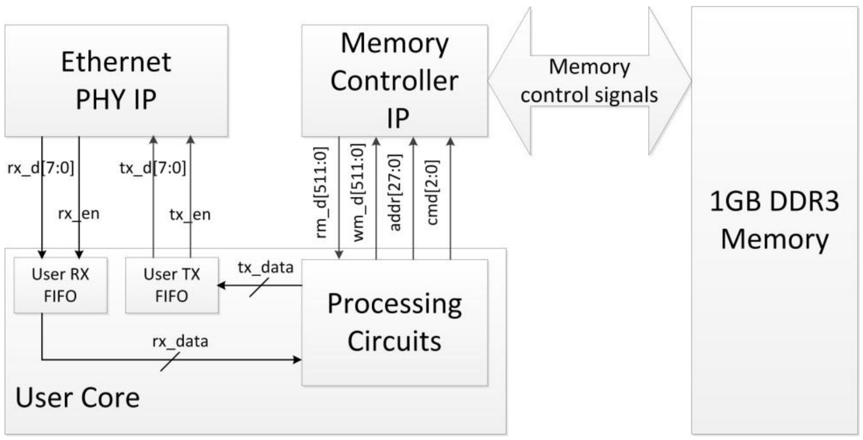

Fig. 2: FPGA data flow in the VC707 EVB

The user core is composed of a TX FIFO, an RX FIFO, and processing circuits. The RX FIFO receives data (rx_d) from the Ethernet physical layer IP and combines them. We use a counter to combine receiving data (rx_d) when the reception signal (rx_en) is enabled. The RX FIFO module outputs the receiving data to the user core module after checking the destination address and the source address is correct. Subsequently, we assert the signal when reception is done. The TX FIFO sends the byte data (tx_d) split from the packet to the Ethernet physical layer IP. The TX FIFO module behavior is similar to the RX FIFO module. We assert the transmission signal (tx_en) after the user core operation is finished. Subsequently, we wrap data into packets when the signal (tx_en) is high. Finally, the packets are sent to the computing server through Ethernet physical IP. The K-means clustering algorithm is implemented in the processing circuit module.

3. K-means Clustering Circuit

K-means clustering algorithm is one of the clustering algorithms for data analysis, and K-means algorithm is used to image processing, cluster analysis, and feature learning. The goal of K-means algorithm is to separate the input data into the number *K* of clusters. The data which are in a cluster have similar properties to each other and be dissimilar in other clusters. Assume that the input data are on a set$X$of D-dimensional real vector, that$X = \{X_n \in R^D,\ \text{with } n = 1, \cdots, N\}$, and partitions Xinto $K (K \leq N)$ sets $S = \{S_1, S_2, \cdots, S_k\}$, and each cluster is associated with a center value. Therefore, the output of the objective function in the Euclidean distance is $\sum_{i=0}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2$, where μ_i is the mean of points (center) in S_i. K-means algorithm is iterative procedures, there are steps as followed.

First, the initialization step selects initial values to be the centers and determines the number of clusters. Basically, the programmer sometimes uses random numbers to be the centers, but initial values affect the result of accuracy or iteration times. Second, the finding clusters step assigns objects to the nearest cluster center. We can get the number of *K* clusters in this step. Third, in finding centers step, calculates each cluster center as the mean of objects in the cluster. If all of the new cluster centers are approximately the same, and then K-means algorithm iteration is finished. Otherwise, the step returns to finding clusters step with new centers to find new cluster sets.

K-means algorithm is applied to many diverse domains, so we focus on hardware implementation. In [12], they implement K-means algorithm for image processing on PCI board with an FPGA and connects external two memories. One memory is storing the pixel data from the host computer and the other memory is saving the cluster results. They use the FPGA to find each pixel cluster numbers, and store sum of pixel values in accumulators. In the host computer side, they send the complete image to the FPGA memories and computes new cluster centers. In [13], they modify architecture of [12]. They add a floating-point division module in the FPGA. This architecture computes new cluster centers on the FPGA side and saves each iteration results in the FPGA before the algorithm is terminated. They also use PCI-e communicating between the host and the FPGA. At the beginning, the host computer sends complete image data to the FPGA memory. Second, initial centers are transmitted to the FPGA from the host. Before K-means algorithm is finished, there is no contact between the host and the FPGA, it saves a lot of transmission time comparing with [12]. The execution time speeds up because of programming in parallelism and reducing transmission time. Nevertheless, the memory capacity is an obstacle, the volume of big data or others data set may be larger than the size of memory.

In [14], they implement K-means cluster for the color image on an FPGA with the Euclidean distance metric. A target pixel is 24-bit full-color RGB images. Each SRAM bank is 32-bit, four pixels are stored in three memory banks and results are stored in rest memory banks. This architecture calculates 96 squared Euclidean distances in parallel and the new center is calculated from four partial sums. Furthermore, they implement filter algorithm that each pixel is less than or equal to 24, and FEKM algorithm [15] can reduce scanning the number of pixels for iteration. They use two techniques for reducing computation time, and the memories can load the next image when others image is processing. However, the number of computing pixels for once operation is four that is not satisfied.

In [16], they implement K-means in hardware for micro array data and compute the minimum distance in parallel. However, all of the input data are stored in the on-chip memory that is not work where the data set is larger than the capacity of the memory. Therefore, we develop FPGA-based hardware accelerator through Ethernet switch to achieve high scalability. In addition, we do not use the onboard DDR3 memory in the design, because the size of data set is larger than the capacity of memories. Furthermore, the VC707 EVB memory access time could cause overload.

The implementation of the K-means clustering algorithm is shown in Fig. 3. First, we create the data set with random three-dimensional nodes and store data set in a text file. Second, the master processor reads the data set into arrays and wraps nodes, cluster centers, control signals into the transmission packets. Subsequently, we send packets to the VC707 EVBs through the Ethernet switch. Third, the VC707 EVBs groups nodes into the clusters after receiving node data and centers. After that, the VC707 EVBs calculate the cluster number of each node. Consequently, the VC707 EVBs return the cluster value of nodes to the host computer. Fourth, the master processor has calculated all cluster centers according to receiving the cluster number of nodes from the VC707 EVBs. Then, the master processor replaces initial centers with the new centers, and next iteration is processed to find the new cluster centers. Finally, we write the results into the text file and display the clusters distribution by MATLAB when the iteration times are terminated.

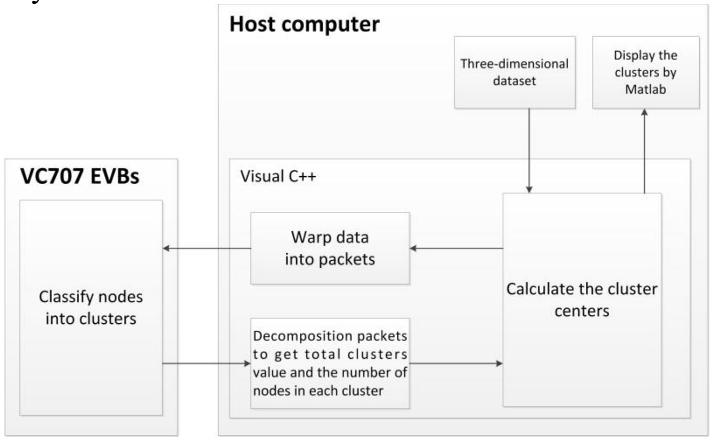
Host computer
Three-dimensional dataset
Display the clusters by Matlab
VC707 EVBs
Visual C++
Warp data into packets
Classify nodes into clusters
Calculate the cluster centers
Decomposition packets to get total clusters value and the number of nodes in each cluster

Fig. 3: Implementation of the K-means clustering algorithm

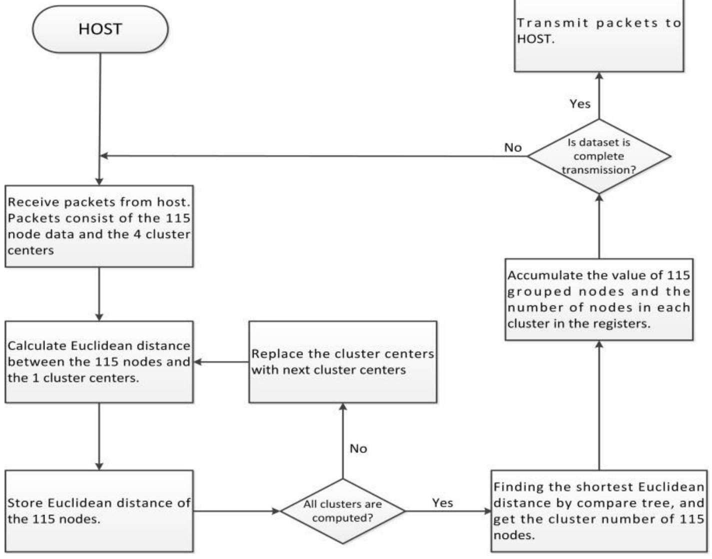
HOST
Transmit packets to HOST.
Yes
No
Is dataset is complete transmission?
Receive packets from host. Packets consist of the 115 node data and the 4 cluster centers
Accumulate the value of 115 grouped nodes and the number of nodes in each cluster in the registers.
Calculate Euclidean distance between the 115 nodes and the 1 cluster centers.
Replace the cluster centers with next cluster centers
No
Store Euclidean distance of the 115 nodes.
All clusters are computed?
Yes
Finding the shortest Euclidean distance by compare tree, and get the cluster number of 115 nodes.

Fig. 4: K-means flow chart in a VC707 EVB

Fig. 4 shows the K-means algorithm flow chart in a VC707 EVB. We share the workloads in three VC707 EVBs, and each VC707 EVB has same components and functions. The single VC707 EVB can handle 115 data nodes in one packet simultaneously. Then, the VC707 EVB returns the sum of the clusters coordinate values with three dimensions and the number of nodes in each cluster to the host computer. Finally, the host computer can calculate new centers and finds the new centers to the next iteration.

In K-means algorithm, the inputs are three-dimensional 115 floating-point nodes and 4 cluster floating-point centers from the host computer. We store the node data and cluster centers when the VC707 EVB has received the packet. Subsequently, we calculate the Euclidean distance to each cluster centers in total 115 nodes. Consequently, we find the shortest distance in 4 Euclidean distance of each cluster. After that, we accumulate the grouped nodes of one cluster coordinate floating-point values that are x-coordinate, y-coordinate and z-coordinate in three 32-bit registers until 115 nodes are computed. Moreover, we store the number of nodes in one cluster into a 96-bit register. Our maximum number of cluster is four, so we have twelve 32-bit registers to store the clusters coordinate values, and four 96-bit registers to store the number of nodes in each cluster. When the transmission of the data set is complete, the VC707 EVB returns the sum of the clusters coordinate values with three dimensions and the number of nodes in each cluster to the host computer. Besides, we do not calculate new centers in the FPGA because we share the workloads to three VC707 EVBs. We need to compute the new centers in the host computer.

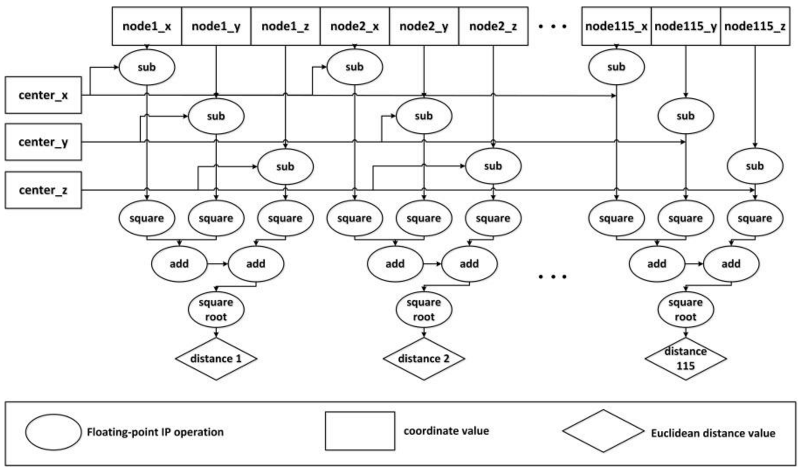

Fig. 5: Euclidean distance calculation circuit

Fig. 5 shows the circuit for calculating the Euclidean distance. The values of nodes and centers are single-precision floating point numbers, and the floating-point IPs help us to process floating-point operations. We calculate 115 Euclidean distances between nodes and centers in parallel. The total of the floating-point IPs has 1035 modules including 345 subtractions, 345 square operations, 230 adders, and 115 square root operations. Both the input and the output of every IP modules are 32 bits. We get 115 distances of each node after six clock cycles. When the calculation is done, we replace center values with next cluster centers.

After four cluster distances between each node and centers are already calculated, we find the shortest distance in four distances by the comparing binary tree circuit, as shown in Fig. 6. Also, we implement 115 comparing tree modules processing in parallel and get the 115 cluster number of nodes after two clock cycles. Finally, we accumulate the coordinate values of 115 grouped nodes and the number of nodes in each cluster into registers. Besides, we do not calculate the 115 grouped nodes in parallel, because the resource of VC707 EVB is not enough to fulfill parallel circuits.

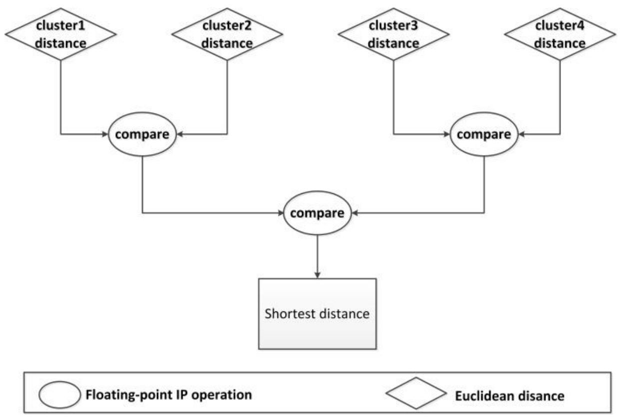

Fig. 6: Comparing binary tree circuit

The master processor is implemented in the host computer by visual studio C++ programming. The master processor is used to composite transmission packets, analyze reception packets, and calculate the new cluster centers. Moreover, the single reception packet from the VC707 EVBs contains the sum of cluster coordinate values and the number of nodes in each cluster. We calculate the new cluster centers after all data set is clustering from three VC707 EVBs. Subsequently, we can do next iteration until the program is finished. Besides, the VC707 EVB is similar as a function that one transmission to the FPGA can process 115 nodes, and one reception from the FPGA can get the sum of cluster values and the number of nodes in each cluster in one iteration. Therefore, we reduce the latency of reception time because the VC707 EVBs return one packet in one iteration.

4. Implementation and Measurement Results

Table 1 shows the hardware resource utilization of the proposed hardware accelerator for K-means clustering circuits. Fig. 7 shows the execution time of K-means clustering algorithm (iteration three times) without the latency of reading and writing data with the 125 million three-dimensional node data set. The execution time is compared to a computer server with Intel I7-4770 CPU, a computer server with Intel I5-3230M CPU, and the proposed hardware accelerator with three VC707 EVBs, and the proposed design has less execution time at 125MHz clock rate. Fig. 8 shows the execution time of K-means clustering algorithm (iteration three times) without the latency of reading and writing data with the different size of three-dimensional nodes data set. The execution time is increased with the growth of the data set size as shown in Fig. 8.

Table 1: FPGA resource utilization for K-means clustering circuit

Slice Logic Utilization	Used	Available	Utilization
K-means clustering			
Number of Slice Registers	125,761	607,200	20%
Number of Slice LUTs	233,865	303,600	77%
Number of Occupied Slices	71,005	75,900	93%
Number of DSP48E1s	2,191	2,800	78%

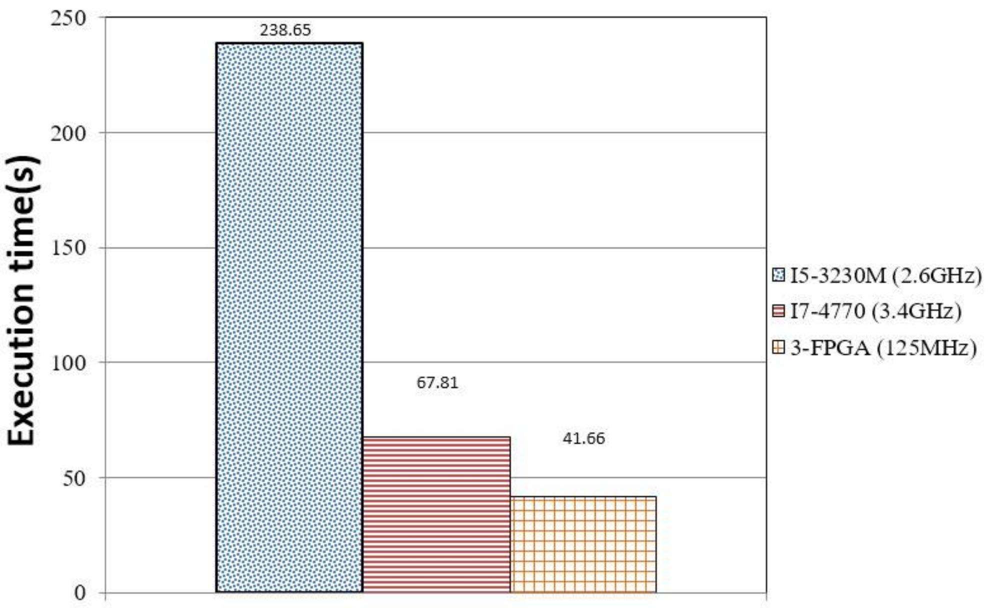

Fig. 7: Execution time of 3-FPGA and the computer server with I5-3230M and I7-4770 CPU (at 125 million three-dimensional nodes data set)

Fig. 9 shows the execution time including the I/O latency of reading and writing data for Hadoop cluster servers and the proposed hardware accelerator with three VC707 EVBs. In the Hadoop cluster, the master server has 8 cores Xeon E5506 CPU(2.13GHz), and all slaves have 8 cores Xeon(R) E5420 CPU(2.5GHz).We use open source Apache Mahout library [17] to implement K-means clustering algorithm in the Hadoop cluster. The execution time of three VC707 EVBs (line 3-FPGA_1) is faster than Hadoop clusters when the size of the data set is small. However, with the large size of data sets, the Hadoop cluster has less execution time than the proposed design with three VC707 EVBs. However, if the data sets are partitioned into three parts and stored on the three host computers, the execution time of three VC707 EVBs with three host computers (line 3-FPGA_2) can be significantly reduced. As a result, the solid-state disks (SSDs) may be required to reduce the I/O latency and the execution time of the proposed FPGA-based accelerator platform can be further reduced.

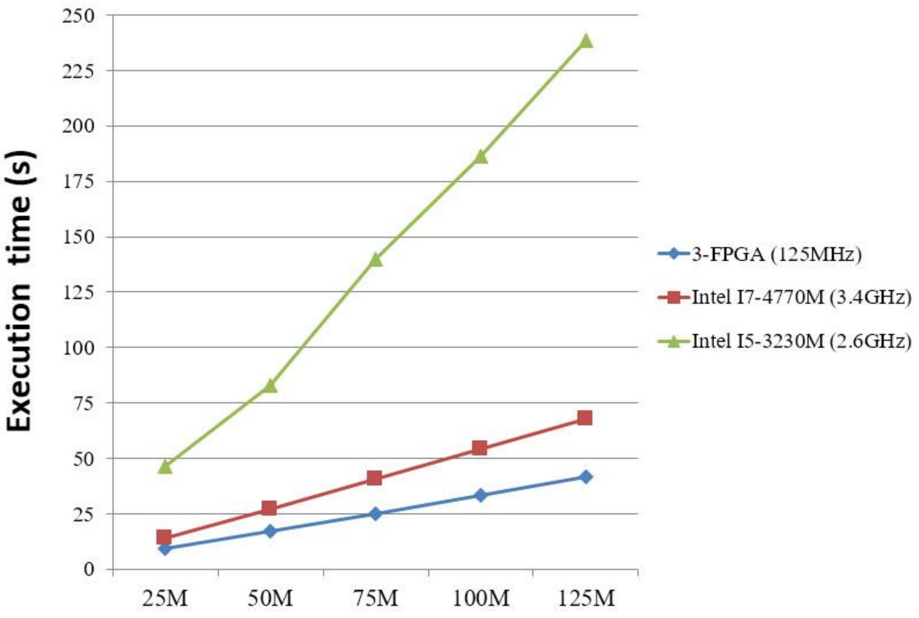

Fig. 8: Execution time at different size of three-dimensional nodes data set

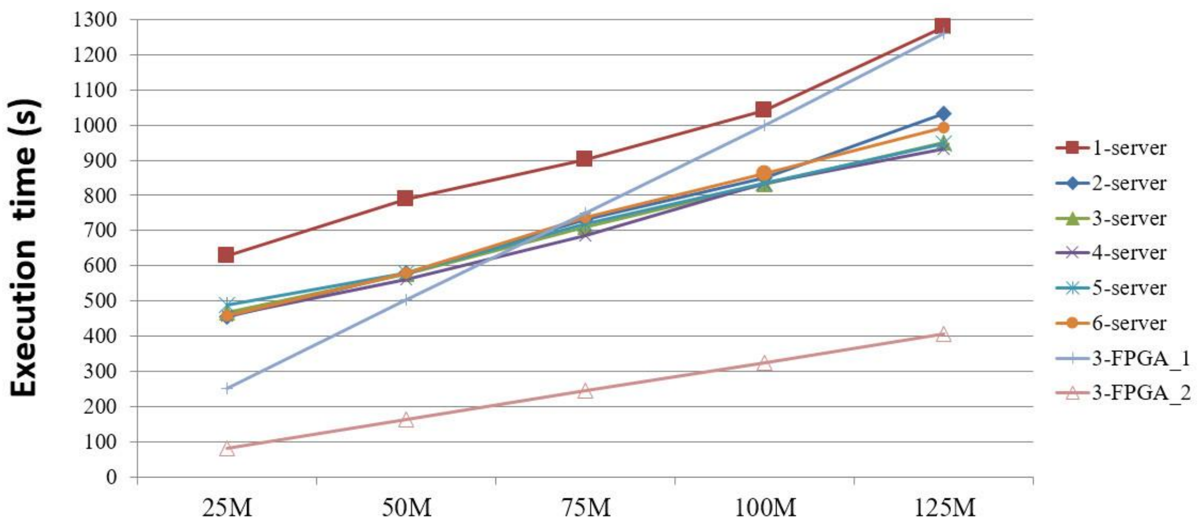

Fig. 9: Execution time of Hadoop clusters with 1/2/3/4/5/6-servers, 3-FPGA, and 3-FPGA with three host computers

5. Conclusion

In this paper, an FPGA-based accelerator platform is proposed to greatly reduce the processing time of the K-mean algorithm. The proposed accelerator platform can use many VC707 FPGA EVBs to speed up the algorithm processing. Since server workloads will continue to evolve, the proposed FPGA-based hardware accelerator platform provides an easy way to support the CPUs for a heterogeneous computing solution with off-chip hardware accelerators.

Acknowledgments

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 106-2221-E-194-059. The EDA tools supported by the National Chip Implementation Center are acknowledged as well.

6. References

[1] Aviv Segev, et al., "Analysis of technology trends based on big data," *in Proc. IEEE International Congress on Big Data (BigData Congress)*, Jun. 2013, pp. 419-420.

[2] Gerd Kortuem, et al., "Market-based user innovation in the internet of things," *in Proc. Internet of Things (IOT)*, Nov. 2010, pp.1-8.

[3] Irena Bojanova, et al., "Imagineering an internet of anything," *Computer*, vol. 47, no. 5, pp. 983-987, Jun. 2014.

[4] Han Hu, et al., "Toward scalable systems for big data analytics: a technology tutorial," *IEEE Access*, vol. 2, pp. 652-687. Jul. 2014.

[5] Sanjay Ghemawat, et al., "The Google file system," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29-34, Dec. 2003.

[6] Doug Beaver, et al., "Finding a needle in haystack: Facebook's photo storage," *in Proc. OSDI*, Oct. 2010, pp.1-8.

[7] Apache Hadoop. Available: https://hadoop.apache.org/

[8] Shuai Che, et al., "Accelerating compute-intensive applications with GPUs and FPGAs," *in Proc. Symposium on Application Specific Processors*, Jun. 2008, pp. 101-104.

[9] Joshua Friedrich, et al., "The POWER8TM processor: designed for big data, analytics, and cloud environments," *in Proc. IEEE International Conference on IC Design and Technology (ICICDT)*, May 2014.

[10] Zhongduo Lin, et al., "ZCluster: a Zynq-based Hadoop cluster," *in Proc. International Conference on Field-Programmable Technology (FPT)*, Dec. 2013, pp. 450-453.

[11] VC707 evaluation board for the Virtex-7 FPGA user guide, Xilinx Inc., Available:http://www.xilinx.com/support/documentation/boards_and_kits/vc707/ug885_VC707

_Eval_Bd.pdf
[12] Mike Estlick, et al., "Algorithmic transformations in the implementation of k-means clustering on reconfigurable hardware," *in Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2001, pp. 103-110.
[13] Xiaojun Wang, et al., "K-means clustering for multispectral images using floating-point divide," *in Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2007, pp. 151-162.
[14] Tsutomu Maruyama, "Real-time k-means clustering for color images on reconfigurable hardware," *in Proc. International Conference on Pattern Recognition*, Aug. 2006, pp. 816-819.
[15] Anjan Goswami, et al., "Fast and exact out-of-core k-means clustering," *in Proc. IEEE International Conference on Data Mining*, Nov. 2004, pp. 83-90.
[16] Hanaa M. Hussain, et al., "FPGA implementation of k-means algorithm for bioinformatics application: an accelerated approach to clustering microarray data," *in Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Jun. 2011, pp. 248-255.
[17] Apache Mahout, Available: http://mahout.apache.org/