國 立 中 正 大 學

資訊工程研究所碩士論文

建構於可程式化邏輯板實現硬體加速之 Hadoop 叢集用於資料探勘演算法

Hadoop Cluster with FPGA-based Hardware Accelerators for Data Mining Algorithms

研究生： 王宥芯

指導教授： 鍾菁哲 博士

中華民國 一零五 年 七 月

國立中正大學碩士班研究生

學位考試同意書

本人所指導
資訊工程學系

研究生 王宥芯 所提之論文

Hadoop Cluster with FPGA-based Hardware Accelerators for Data Mining Algorithms 建構於可程式化邏輯板實現硬體加速之Hadoop叢集用於資料探勘演算法

同意其提付 碩 士學位論文考試

指導教授 鍾葉哲 簽章

105 年 6 月 6 日

國立中正大學碩士學位論文考試審定書

資訊工程學系

研究生 王宥芯 所提之論文

Hadoop Cluster with FPGA-based Hardware Accelerators for Data Mining Algorithms 建構於可程式化邏輯板實現硬體加速之Hadoop叢集用於資料探勘演算法

經本委員會審查，符合 碩 士學位論文標準。

學位考試委員會
召集人 李順裕 簽章

委員

李順裕	鍾菁哲
唐 鐸	

指導教授 鍾菁哲 簽章

中華民國 105 年 7 月 5 日

博碩士論文授權書

（本聯請裝訂於論文紙本書名頁前空白處，供學校圖書館做為授權管理用）

ID:104CCU00392071

本授權書所授權之論文為授權人在 國立中正 大學(學院) 資訊工程研究所 系所 _____ 組 104 學年度第 二 學期取得 碩 士學位之論文。

論文題目： 建構於可程式化邏輯板實現硬體加速之Hadoop 叢集用於資料探勘演算法

指導教授： 鍾菁哲.Ching-Che Chung

茲同意將授權人擁有著作權之上列論文全文(含摘要)，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印，此項授權係非專屬、無償授權國家圖書館及本人畢業學校之圖書館，不限地域、時間與次數，以微縮、光碟或數位化方式將上列論文進行重製，並同意公開傳輸數位檔案。

紙本論文：茲同意將授權人擁有著作權之上列論文全文(含摘要)，提供讀者基於個人非營利性質之閱覽或列印，此項授權系非專屬、無償授權國立中正大學圖書館做為編目上架及公開陳列閱覽使用。

☐ 校內外立即開放
☐ 校內立即開放，校外於 年 月 日後開放
☑ 校內於 2021 年 12 月 31 日；校外於 2021 年 12 月 31 日後開放
☐ 其他

授權人：王宥芯

簽　名：王宥芯　　日期：2016 年 08 月 12 日

摘要

由於物聯網的興起，人們經由網路與伺服器交換的資料量越來越龐大，隨著大數據的演進，如何從巨量資料中挖掘出有價值的資訊，是現今的重要課題。因此資料探勘演算法被廣泛使用在各個領域上。而如何處理這些海量的資料以及分析不同的資料型態成為大數據會面臨到的問題。

為了解決儲存設備與運算能力的限制，分散式系統和雲端運算近年也越來越普及，透過多個伺服器叢集執行平行化的運算，克服 CPU 運算速度的瓶頸；此外，藉由多個伺服器的串聯來增加儲存的容量，彌補單一設備空間不足的問題。為了提升運算的效能，在處理任務的時候，可以藉由硬體加速平台來分擔運算的負載。硬體加速平台最常見的有圖形處理加速器(GPU)和可程式邏輯陣列(FPGA)，通常擁有數量眾多的運算單元，用來執行高密度且獨立的運算並達到運算平行化。

本論文針對巨量資料的儲存平台與運算能力的增進，提出一個軟硬體整合的方案，在 Hadoop 系統串聯以 FPGA 為基礎的硬體加速平台，利用 Hadoop 叢集的分散式檔案系統(HDFS)以及 MapRdeuce 的平行運算優勢，再藉由網路分享器提升擴充性，建構一個用於資料探勘演算法的 Hadoop 與 FPGA 整合的加速平台。我們使用在資料探勘中最常見的 K-means 分群演算法以及 KNN 最近鄰居分類演算法來呈現此整合加速平台的優勢。

關鍵字: 大數據，Hadoop 叢集，現場可程式邏輯門陣列，K-means 分群演算法，KNN 最近鄰居分類演算法，硬體加速平台

Abstract

Since the growing popularity of the internet of things (IoT), the amount of data people exchange via web servers are increasing huge. With the evolution of big data, it is important to extract the valuable information from the massive data. Therefore, data mining algorithms are widely used in various fields. The “5Vs” including volume, velocity, variety, veracity and, value are the challenges of big data processing and analyzing.

In order to overcome the limitations of storage devices and computing capability, the distributed systems and cloud computing are becoming popular in recent years. The parallel computing cluster by multiple servers can conquer the bottleneck of CPU computing capability. In addition, the distributed systems can provide the advantage of storage capacity to make up for the lack of the disk space issue. Graphic processing units (GPUs) and field programmable gate arrays (FPGAs) are potential hardware accelerators and usually have a large number of arithmetic units for performing high density and independent operations in parallel to enhance the effectiveness.

In this thesis, the implementation of the K-means clustering algorithm and K-nearest neighbor algorithm on a Hadoop cluster with FPGA-based hardware accelerators is presented. The proposed design follows MapReduce programming model and uses Hadoop distribution file system (HDFS) for storing large dataset. The proposed FPGA-based hardware accelerator for speed up the proposed algorithms is implemented on Xilinx VC707 evaluation boards (EVBs).

Keywords: big data, Hadoop, field programmable gate, K-means clustering algorithm, K-nearest neighbor algorithm, FPGA-based hardware accelerators

Content

List of Figure

List of Table

Chapter 1

Introduction

1.1 Introduction to Big Data

In the past few years, information technology continues to penetrate and innovate in all areas of social, economic and life. In fact, all of the industries have to confront the issues of big data analytics. Under the support of mobile computing, internet of things (IoT), cloud computing and a series of emerging technologies, social media, collaborative creation, virtual services and other new application models continue to expand the scope and form of human creativity and information exploitation. The global amount of data explosively grow up day by day.

According to the IDC report [1], the digital universe will grow by a factor of 300, from 130 Exabytes to 40,000 Exabytes, or 40 trillion gigabytes from 2005 to 2020, and it will about double every two years. Although the portion of the digital universe holding potential analytic value is growing, only a tiny fraction of territory has been explored. IDC estimates that by 2020, as much as 33% of the digital universe will contain information that might be valuable if analyzed. The data are exponential growth as shown in Fig. 1.1. It is obviously the era of big data is coming.

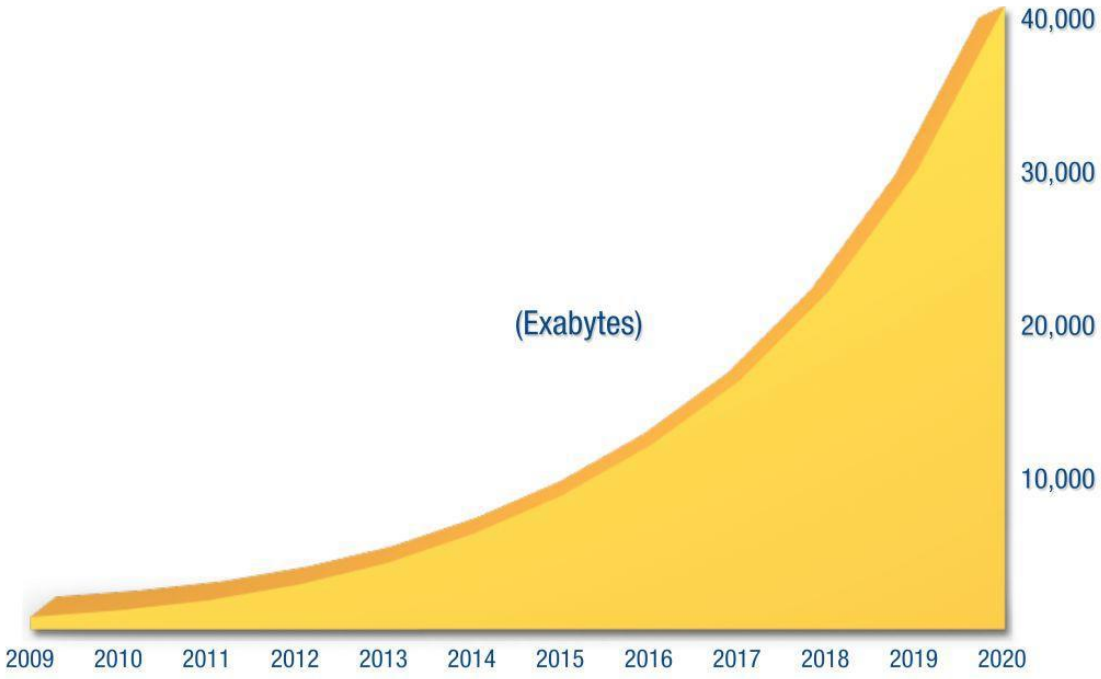


Figure 1.1: 50-fold Growth form beginning of 2010 to the end of 2020 [1]

Generally, the range of big data covers a wide discrepancy between the various definitions. Big data is a collection of data set that too complex and large to be managed, analyzed and processed by using the traditional database system [5][6]. It includes activity logs, business transaction, images, and surveillance videos that can reach massive proportions over time [5][6]. First "3Vs" model was proposed by Gartner Inc. [2], and they pointed to three key challenges of data processing, volume, velocity, and variety [4]. Besides the explanations of big data based on [3][7][8], "5Vs" is widely applied to the definition of big data [9][10], including volume, velocity, variety, veracity and, value. Fig. 1.2 below illustrates the feature related to "5Vs."

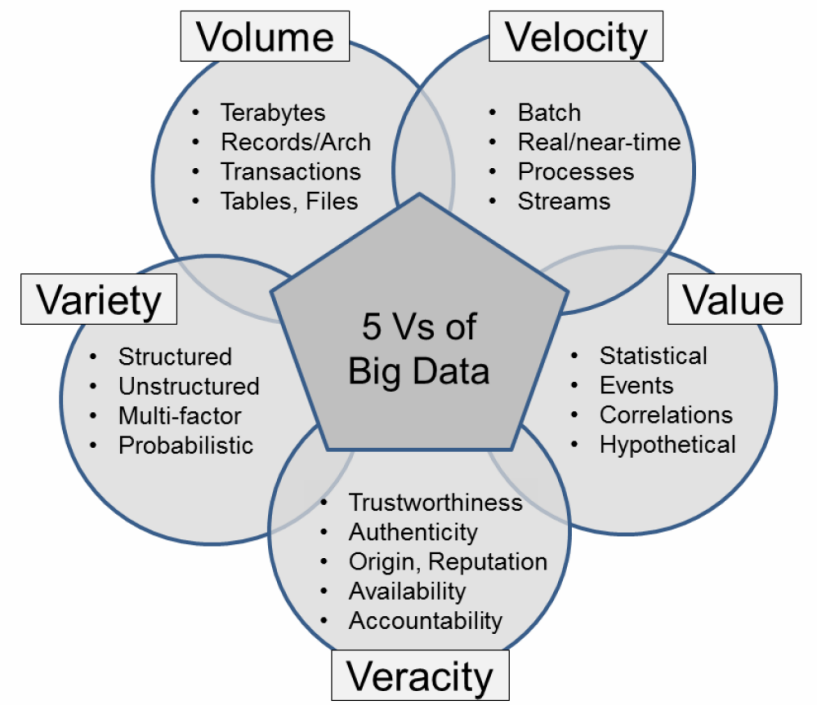


Figure 1.2: The 5Vs of Big Data [10]

Volume is the amount of the data to produce, process, and preservation. These data generated exceed 2.5 quintillion bytes everyday [5], and with 90 percent of the world's data created in the last 2 years [1]. The traditional database and hardware don't have the capability to store big data.

Velocity represents that the data streaming into the servers in real time is continuous and fast. As more and more machines, internet users, social networks, and results of searching are growing every second, the processing time is very important for marketing prediction. The important information should be obtained immediately to maximize the value, and therefore, big data must be analyzed at a rate that matches the speed of data production.

Variety means big data includes the structured data of pure text, audio, video, web, and streaming, semi-structured, and unstructured data. These data are not in the same format, and most of them are unstructured so they are not easy to be handled. Hence, dealing with many different formats of data is one of the challenges of big data.

Veracity of big data should be considered during processing and analyzing. Because the resource of data become diverse, the reliability and quality of the information are not stable. We must be sure about the information which is correct in order to prevent dirty data damage the system completeness and correctness.

Value is an important feature of the data. Data value depends on the events or processes they represent such as stochastic, probabilistic, regular or random. In some respect, data value is closely related to the data volume and data variety.

Due to the certain essential characteristics of big data, big data is not easy to be analyzed with a single computer, and have many challenges and issues which need to be solved [15-18]. We summarize the discussion of above references as follows:

Data Privacy and Security: People share their own personal information every day in social media network such as Facebook. These personal information are collected and used in order to add the value of the business. In certain domains, like financial data, medical information, as well as government intelligence, those data have standards for data security and confidentiality requirements. These private data must be encrypted in case hackers hijack these valuable intelligence.

Data Storage and Management: Big data is excessive large and has a rapid growth rate. Therefore the available storage are not enough for storing the large amount of data. In addition, data management addresses massive amounts of heterogeneous and complex data, such as semantics, structure, video and text. Under the trend that big data is growing at an alarming rate and data explosion causes the consumption of system resource seriously. The static storage solution can't satisfy the challenge of data dynamic evolution. To conquer the issue, Han Hu [15] classifies the data management framework into three layer which consist of file systems, database technology, and programming models as shown in Fig. 1.3.

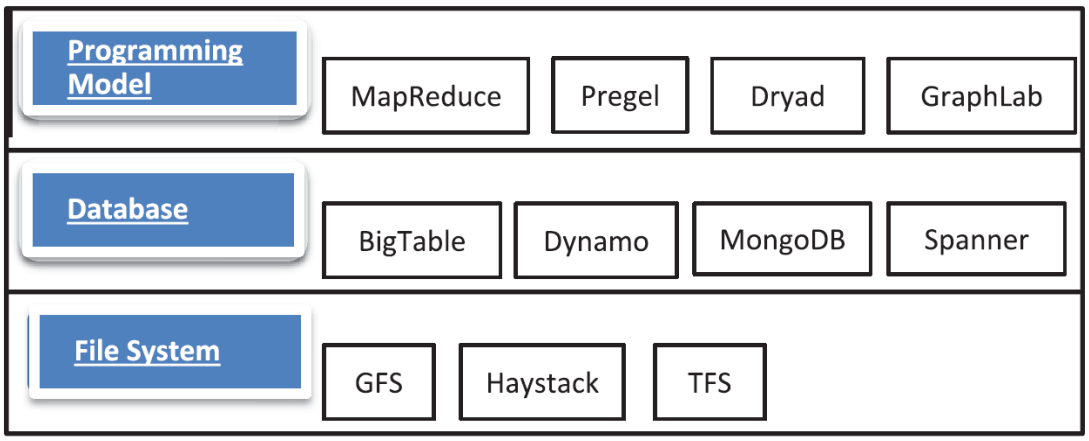


Figure 1.3: Data Management Framework [15]

The brief of this framework is as following:

✓ **File System**: The file system is the basis of big data storage. Google designed and implemented the Google File System (GFS) [14][19] as a scalable distributed file system for data intensive application. GFS runs on hundred inexpensive servers to provide fault tolerance and high performance to a large number of clients. Additionally, Facebook implemented Haystack [22] to store large amount of small size of photos to serve the long tail of request seen by sharing photos in a large social network. Haystack achieved four main goals, including high throughput and low latency, fault tolerant, cost effective and simple. Furthermore, Taobao File System (TFS) [23] is also a large-scale and high performance similar distributed file system for a massive amount of small files.

✓ **Database**: NoSQL [24] database is becoming the standard to deal with the big data problems and the systems are high scalability, reliability, and availability that are suitable for unstructured data and management of datasets. Bigtable [25] is a type of column-oriented of NoSQL databases and is a compressed, high performance, and proprietary data storage system built on GFS to store large-scale structured data for cloud computing [20]. In addition, Amazon Dynamo [26] is a type of key-value stores

of NoSQL database service which can form a highly available key-value structured storage system and a distributed data store. Moreover, one of NoSQL databases, document stores databases, support more complex data than key-value stores. MongoDB [27] is a type of cross-platform document-oriented of NoSQL database written by C++ in order to solve a lot of problems in application development community. Because the relational databases and NoSQL databases have their own advantages and disadvantages, there are hybrid databases to combine relational and NoSQL databases to gain advanced performance. Spanner [28] is the first system to globally distributed data and support externally consistent distributed transactions. Unlike the key-value store model in Bigtable, each table of Spanner must have a primary key column. Moreover, Spanner has evolved into a temporal multi-version database and the special features of Spanner are externally consistent reads and writes and the globally consistent reads across the database at a timestamp.

- ✓ **Programming Models**: Although NoSQL databases are attractive for many reasons, unlike relational database systems, they don't support declarative expression and offer limited support of querying and analysis operations. The programming models is the key of implementing the application logics and facilitating the data analysis applications. One of the processing models is the generic processing model which addresses general application problems and is used in MapReduce [29] and Dryad [30]. Both of them are the distributed system for parallel application for large-scale computations. Besides, graph processing model is a type of programming model and can express a growing class of applications and capture the related entities. Pregel [32] by Google is one of the graph processing model to specialize in large-scale computing. GraphLab [31] is another graph processing model to target parallel machine learning algorithms and data mining tasks.

Data Analysis and Application: The analysis on the huge data which can be structured, semi-structured, and unstructured requires advance skills. It is difficult to analyze entire dataset to extract the valuable information from the large amount of data. The emerging analytics has six critical technical areas, structured data analytics, text analytics, multimedia analytics, web analytics, network analytics, and mobile analytics. We expect machine learning and data mining can be more helpful in big data analysis to explore the valuable benefits. Moreover, processing the large amount of data also takes large amount of time. Thus, we can use large-scale parallel system to dramatically reduce the response time for data-intensive operations on large databases.

With big data growing up every second, the traditional database systems obviously can't address the variety and scale challenges and both hardware and software are evolved to adapt the characteristics of big data. Thus, many software frameworks are developed to against big data issue to extend the high scalability and fault tolerance, for example, Hadoop [11] and Spark [13], are provided to handle massive data.

1.2 Software Framework

Hadoop [11] is a cluster system that enables massive data storage and distributed processing over large number of computing servers. It is designed to scale up from a single server to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, Hadoop itself is designed to detect and handle failures at the application layer, and so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. The Hadoop cluster server division of roles is shown in Fig. 1.4. Hadoop has Master Node and Slave Node which are cluster computing servers. There are one Name Node and one JobTracker in the Master Node, and a Data Node and a Task Tracker in each Slave Node. There are two main modules, Hadoop distributed file system (HDFS) and MapReduce, we will use these two modules in our propsed system.

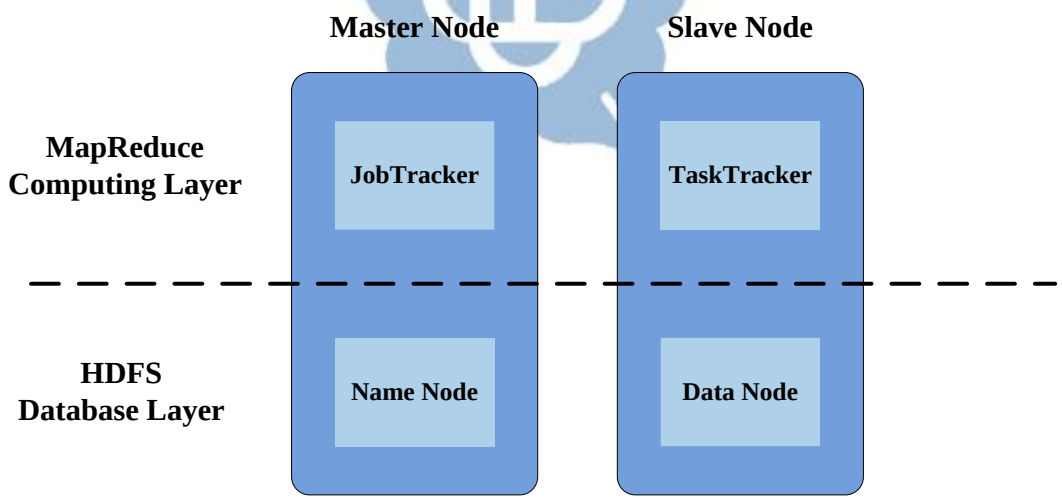


Figure 1.4: Hadoop cluster server division of roles

The HDFS architecture is shown in Fig. 1.5. HDFS is a distributed file system which provides high throughput access to the application data and has a master/slave architecture. An HDFS cluster consists of a single NameNode, and a master server that manages the file system namespace and regulates access to files by clients. In addition,

there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. In HDFS, a file will be split into one or more blocks, and each block will have several replications on different DataNode to prevent missing data.

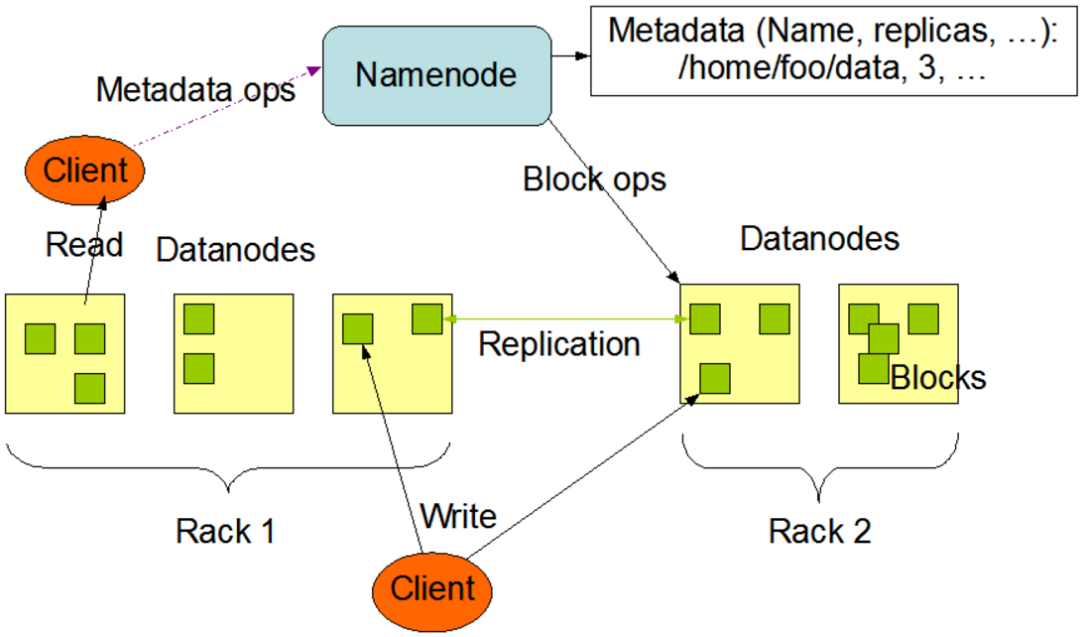


Figure 1.5: Hadoop Distributed File System Architecture

MapReduce [29] is a software framework for easily writing applications which process vast amounts of data in parallel on large computing clusters in a reliable and fault-tolerant manner. It is a concept of divide and conquer and a MapReduce program is composed of a Map() procedure that performs filtering and sorting and a Reduce() method that performs a summanry results. The MapReduce operation is shown in Fig. 1.6. The MapReduce framework operates exclusively on <key, value> pairs. The framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job. The Mapper maps input <key, value> pairs to a set of intermediate <key, value> pairs and the Reducer reduces a set of intermediate values which share a key to a smaller set of values.

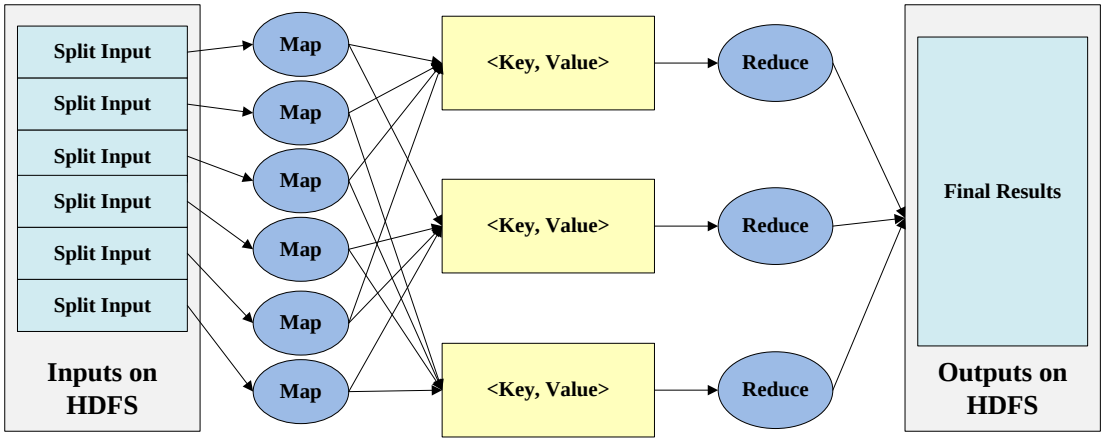


Figure 1.6: MapReduce Framework Operation

Spark [13] is a fast and general engine for large-scale data processing that supports cyclic data flow and in-memory computing and provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance. Spark is also compatible with Hadoop. However, the difference with Hadoop MapReduce, Spark works in-memory speeding up orcessing time by offering over 80 high-level operators that make it easy to build parallel apps and you can use it interactively from the Scala, Python and R shells. In the structure, each Spark application is composed by a driver program which runs the user's main() fuction to execute variety parallel operations on clusters. The main abstract of Spark is to provide a Resilient Distributed Dataset (RDD) which represents an immutable, partitioned collection of elements that can be operated on in parallel. Another abstract of Spark is shared variables which are run in parallel computations. Spark powers a stack of libraries including Spark SQL which is the module for working with structured data for fast engine interactive queries, Spark Streaming which builds scalable fault-tolerant streaming application easily for real time streaming data analysis, GraphX which is an API for graphs and graph-parallel computation, and MLlib which is a scalable machine laerning library for machine learning as shown in Fig. 1.7.

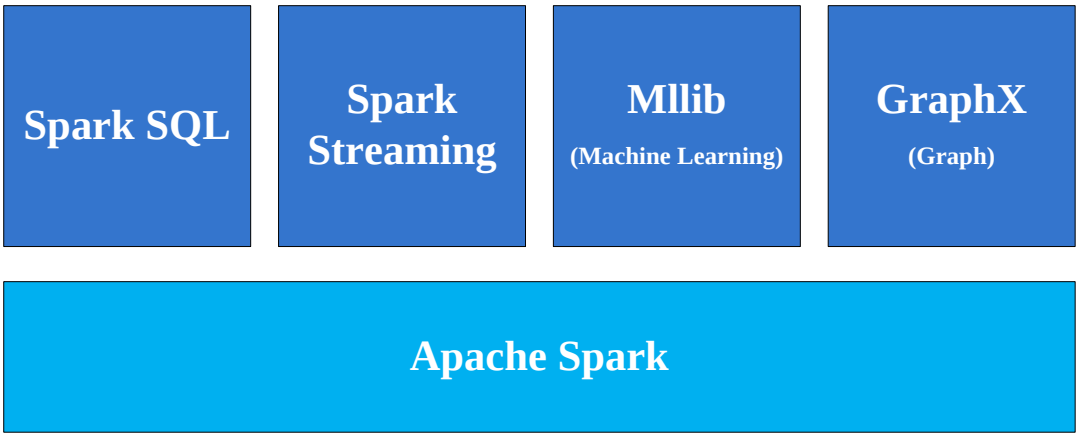


Figure 1.7: Spark Stack of libraries

Besides the development of software frameworks, the computing servers are also required new system capabilities to adapt to the data explosion. A single specification of CPUs is dissatisfied with the increasing computing demand. Therefore, we must rely on high-efficiency computing capability to deal with big data and enhance the performance per watt ratio. In this situation, the heterogeneous computing is not only a secret weapon for supercomputers but also a powerful tool which can share the computing loading to improve the performance of data mining for developers.

1.3 Hardware Accelerator

Heterogeneous computing [40] is getting more attention in recent years, because the issues of traditional ways by raising the CPU clock frequency and adding the number of cores inevitably lead to greater complexity and heat. In addition, the scalar processors have a fundamental limitation of difficultly extracting more instruction-level parallelisms (ILPs). Meanwhile, heterogeneous computing systems can gain the performance and energy efficiency by employing the specialized processors like graphics processing units (GPUs) and field programmable gate arrays (FPGAs) to accelerate some operations. Moreover, heterogeneous architecture property can separate jobs and allow the appropriate hardware co-processors for each operation within given applications, so hardware accelerators can excellently work on computational intensive tasks, and software programs can process complex functions and manage data.

The general purpose on graphics processing units (GPGPU) is a powerful engine for computationally demanding applications [41]. Both AMD's and NVIDIA's flagship GPUs feature unified as shown in Fig. 1.8. The advantages of the GPU are high memory bandwidth, many programmable cores with multi-thread execution in parallel, coding with high level languages like Nvidia's compute unified device architecture (CUDA) which provides a highly parallel architecture with hundreds of cores and very high memory bandwidth, and easy to reprogram functions. For efficiency, the GPU processes many elements in parallel using the same program. Each element is independent from the other elements, and in the base programming model, elements cannot communicate with each other.

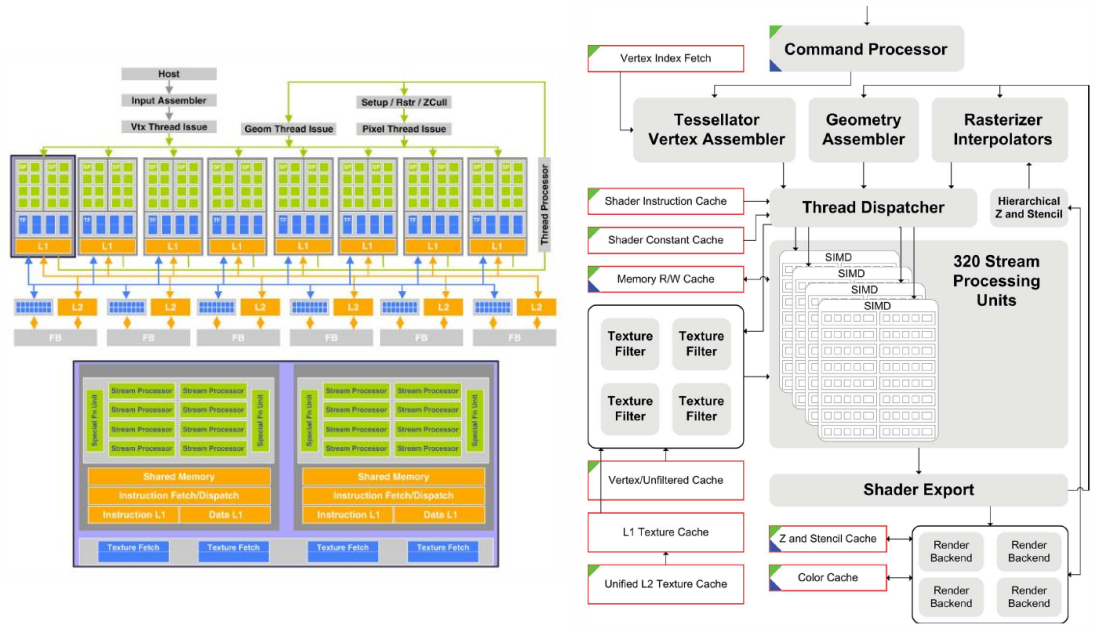


Figure 1.8: Both AMD and NVIDIA build architectures with unified, massively parallel programmable units at their cores. [41]

A GPU implementation speedup of the K-means algorithm [42] presented two different strategies for low-dimensional data sets and high-dimensional data sets respectively, in order to make the best use of the power of GPUs. They used NVIDIA GTX280 GPUs as shown in Fig. 1.9 to develop their platform. For low dimensional data sets, they exploited GPU on-chip registers to significantly decrease data access latency. For high dimensional data sets, they design parallel programming pattern used in matrix multiplication to accelerate the K-means algorithm. Overall, their GPU-based K-means algorithm was three to eight times faster than the best reported GPU-based algorithm. However, the implementation could deal with a finite scale of data set and the size is limited by the global memory size of the GPU, but the strategy couldn't handle the larger size of data set which is out of the limitation.

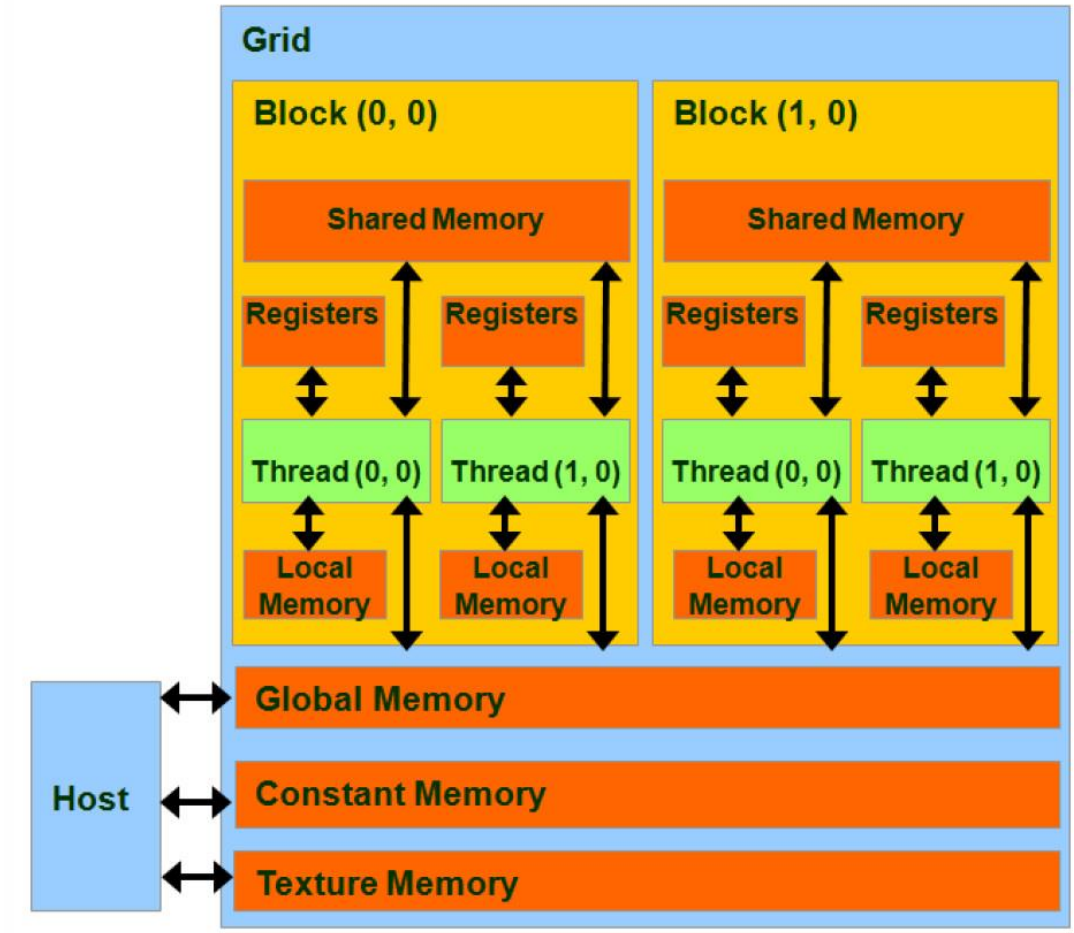


Figure 1.9: NVIDIA GTX280 hardware architecture of the GPU [42]

GPU in Hadoop framework [37] presented the advantages of GPU massively parallel processors and Hadoop large-scale MapReduce clusters. According to Fig. 1.10, GPU accelerators are added in each slave node, and the native program can control both the CPU and GPU resource. The map and reduce task are the processing units in the Hadoop framework which can be accelerated by the GPU. They used GPU to accelerate the big map-task which can be about 100 times faster than the same computation assign to the CPU. However, the better performance of GPU depends on higher IO speed and running application characteristics. In order to address this problem, a hybrid system for GPU-based homogeneous clusters was presented [38].

The hybrid scheduling technique for GPU-based computer clusters [38] is shown

in Fig. 1.11 which minimizes the execution time of a submitted job using dynamic profiles of map tasks running on CPU cores and GPU devices when a client submits a MapReduce job and the tasks can run on both CPU cores and GPU devices. They demonstrates the results and the speedup can be achieved 1.93 times faster than the Hadoop original scheduling algorithm at 64 nodes (1024 CPU cores and 128 GPU devices).

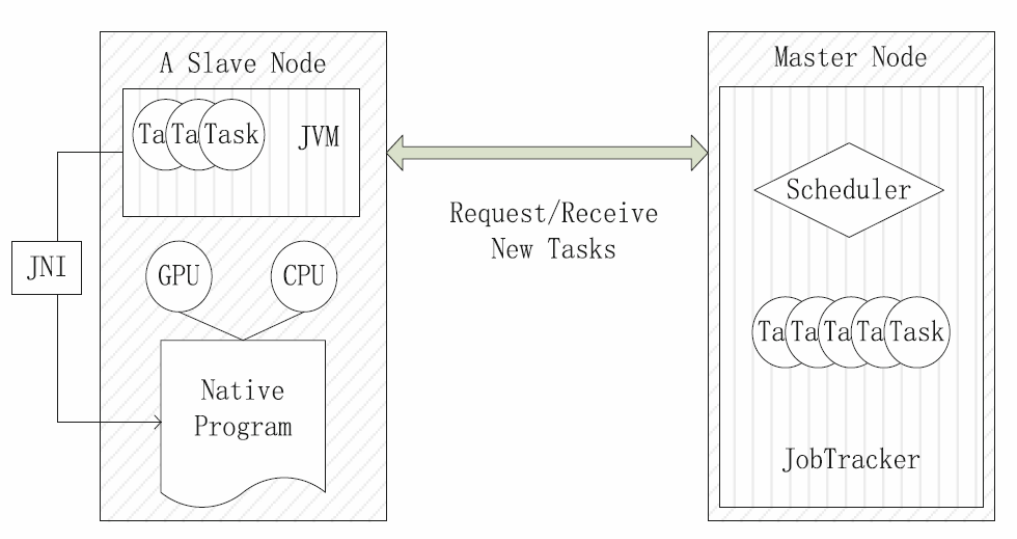


Figure 1.10: Hadoop-GPU Framework [37]

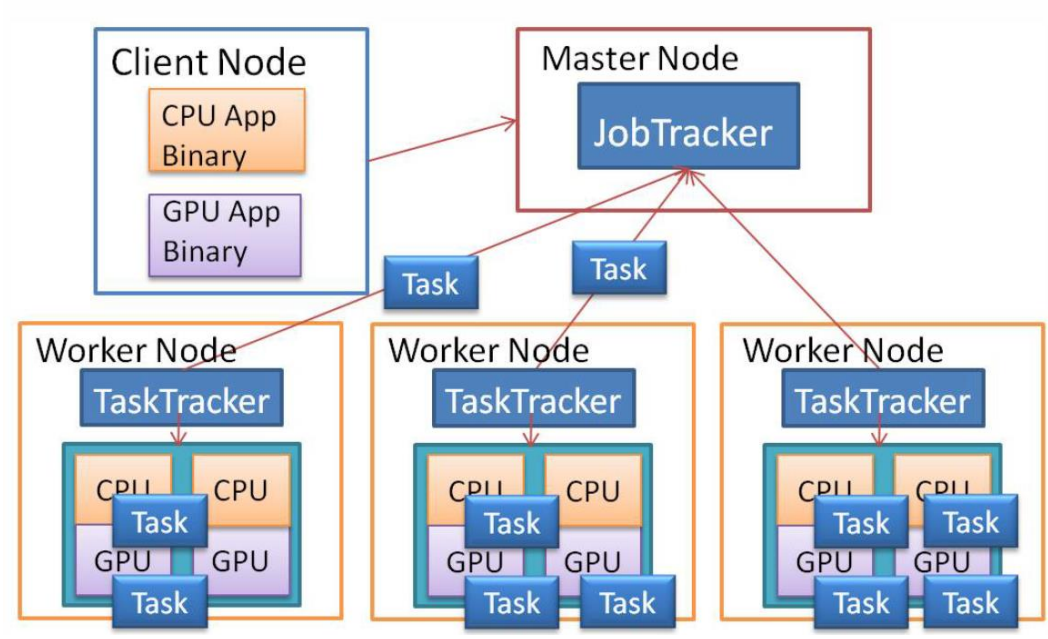


Figure 1.11: Overview of the hybrid scheduling technique [38]

Unlike CPU and GPU, FPGAs don't have fixed pipeline or instruction set, instead, FPGAs have high density arrays of logic blocks which can execute computation in parallel. Users can program by using hardware description languages (HDL) and vendors can provide intellectual property cores (IP cores) to speed up the design of the projects. Besides, the hardware accelerator offers huge performance and energy improvements over general purpose processors are more popular for use in implementing application specific accelerators [35]. Efficient computing of machine learning and data mining has gained much more attention in recent years, while becoming more and more challenges with growing data size and higher performance requirements [36]. Recently, the physical constraints of CPUs and the power consumption are becoming a critical problem so FPGA has been widely explored in various high performance computing applications because of its low power, low cost, and reconfigurable [34].

A CPU-FPGA heterogeneous architecture was proposed for accelerating a short reads mapping algorithm [43], and it was built on the concept of hash-index. The proposed architecture is implemented and evaluated on a customized FPGA accelerator card with a Xilinx Virtex5 LX330 FPGA which is mainly composed of a processing element (PE) controller, an array of PEs and data buses, as shown in Fig. 1.12. The accelerator is as a co-processor connected to the host CPU through the PCIe interface and consists a processing array which is composed of 100 four stage pipelined processing units, a central controller and three separated data buses. The implementation which is designed to take the advantage of the spatial parallelism on FPGA can achieve the speedup of 22.2 to 42.9 times than an Intel six-cores CPU. However, the architecture can be improved to upgrade the PEs to align long reads with more tolerance due to the increased sequencing error rate.

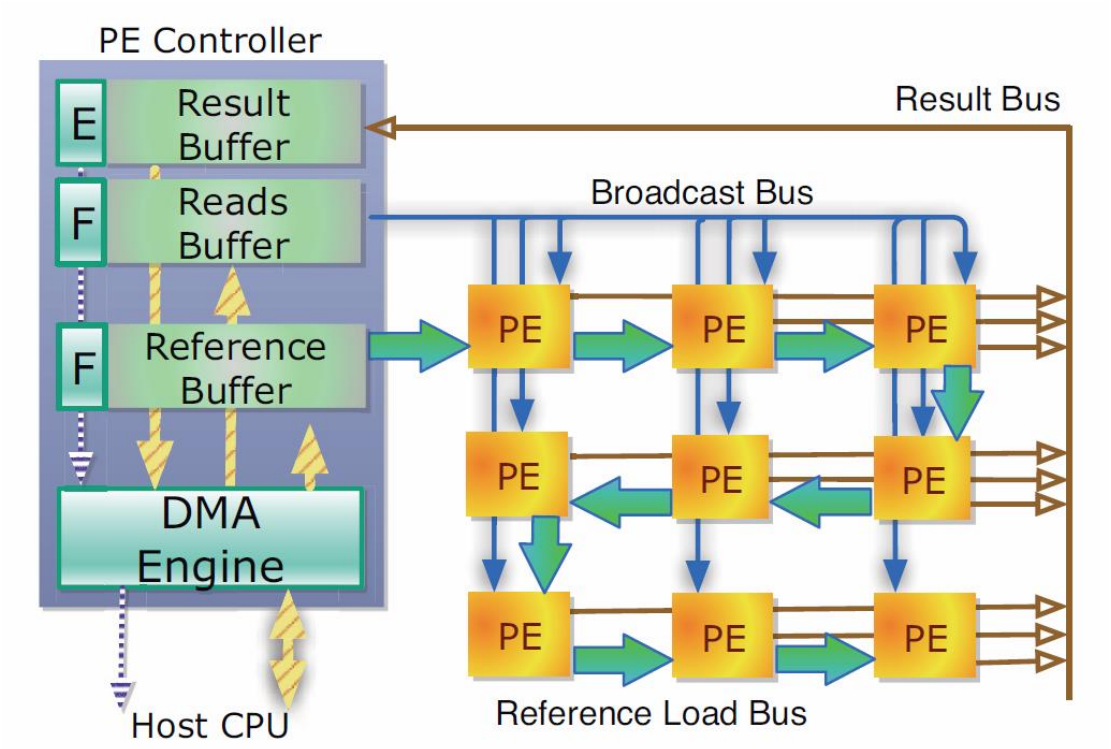


Figure 1.12: Proposed FPGA accelerator architecture overview [43]

A novel framework was presented for implementing portable and scalable data-intensive applications on reconfigurable hardware [44]. They focus on standard PCs and PCI-Express extension cards featuring FPGAs and memory as shown in Fig. 1.13. Both components of a PC and an FPGA communicate through a PCIe bus and the PCIe provides fast and scalable communication. The maximum bandwidth is determined by the capabilities of both the host PC and the FPGA board. The proposed FPGA components is shown in Fig. 1.14. This frame work consists of hardware implementation rules, a communication API along with corresponding hardware wrappers, and a method to automatically select an application's optimal replication factor R for a given hardware platform. The parallel kernels achieve the speedup 9.2 times over the single-PE versions, and the area of the circuits is well within the available FPGA area. However, the automatic optimization method can be extended of considering memory bandwidth and allowing more hardware parameters.

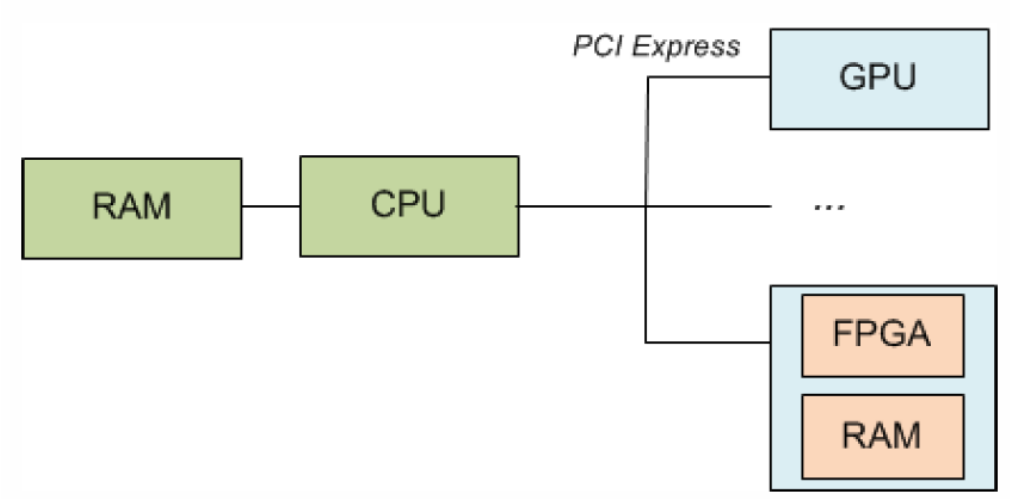


Figure 1.13: Accelerator Architecture: Standard PC with PCIe cards [44]

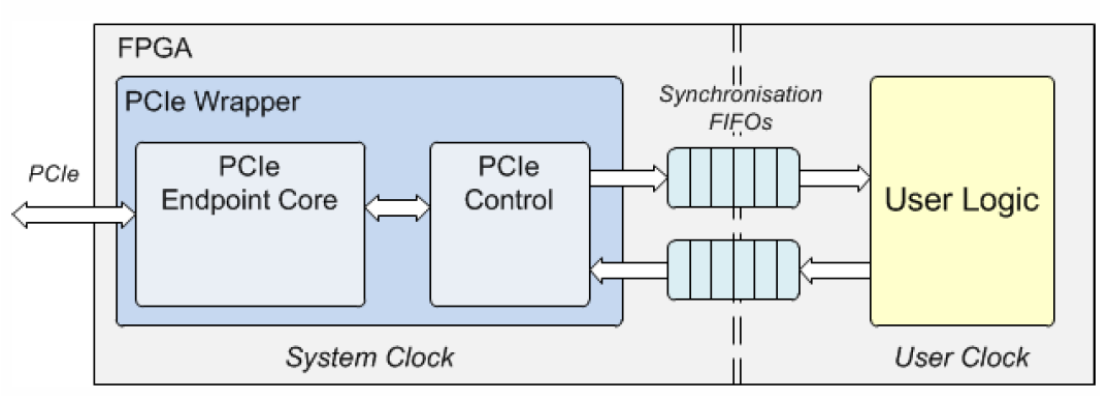


Figure 1.14: Proposed FPGA components [44]

An on-chip processor scheduler is implemented to maximize the utilization of computation resources and achieve better load balancing. An efficient data access scheme is carefully designed to maximize data reuse and throughput. Meanwhile, the FPMR [36], a MapReduce framework on FPGA, as shown in Fig. 1.15, hides the task control, synchronization, and communication away from designers so that more attention can be paid to the application itself. The initial <*key, value*> pairs are prepared by CPU and then transferred to the FPGA through PCI-E bus or CPU bus. They demonstrates 31.8x speedup of RankBoost than the CPU-based implementation and the performance is comparable to a fully manually designed version, which achieves 33.5x speedup.

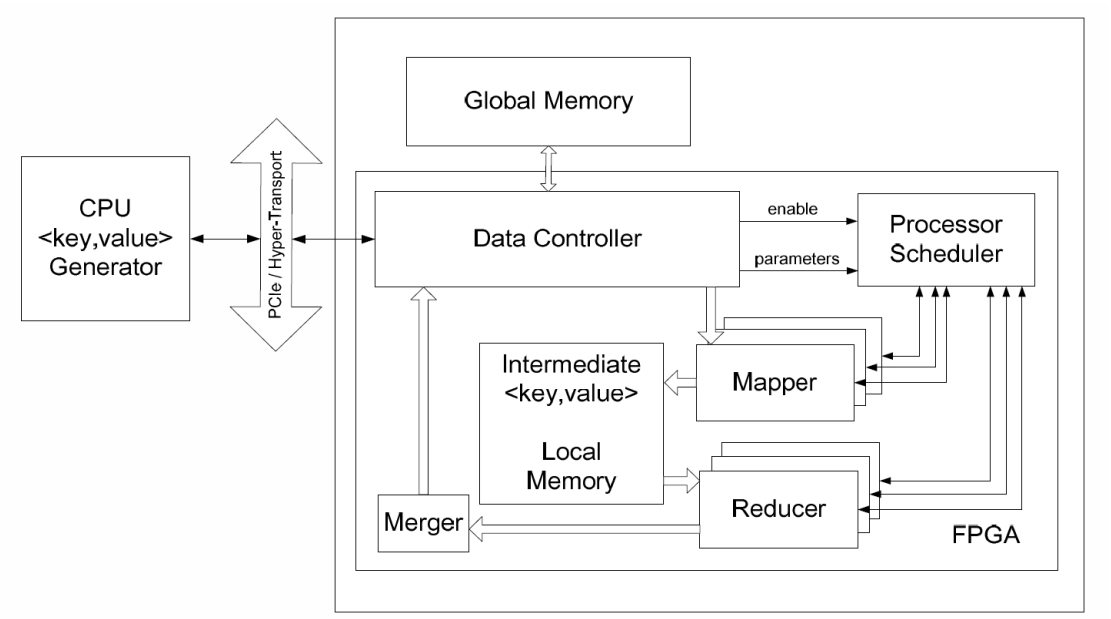


Figure 1.15: FPMR Framework [36]

The ARM-based Hadoop clusters with FPGA-based hardware accelerators to improve the performance in big data analytics are proposed. In ZCluster [21], Zynq platform which integrates the ARM processor and the FPGA in the single chip is used as the Slave Node as shown in Fig. 1.16. Then, the FPGA can act as a co-processor to share the loading of the ARM processor. Thus, the execution time of applications can be reduced. However, in addition to the jobs executed in the FPGA, some irregular jobs of the applications still need to run with ARM processor. Thus, if only a small portion of jobs can be executed by the FPGA, the relatively slow performance of the ARM processor as compared to the Intel processors will be the problem.

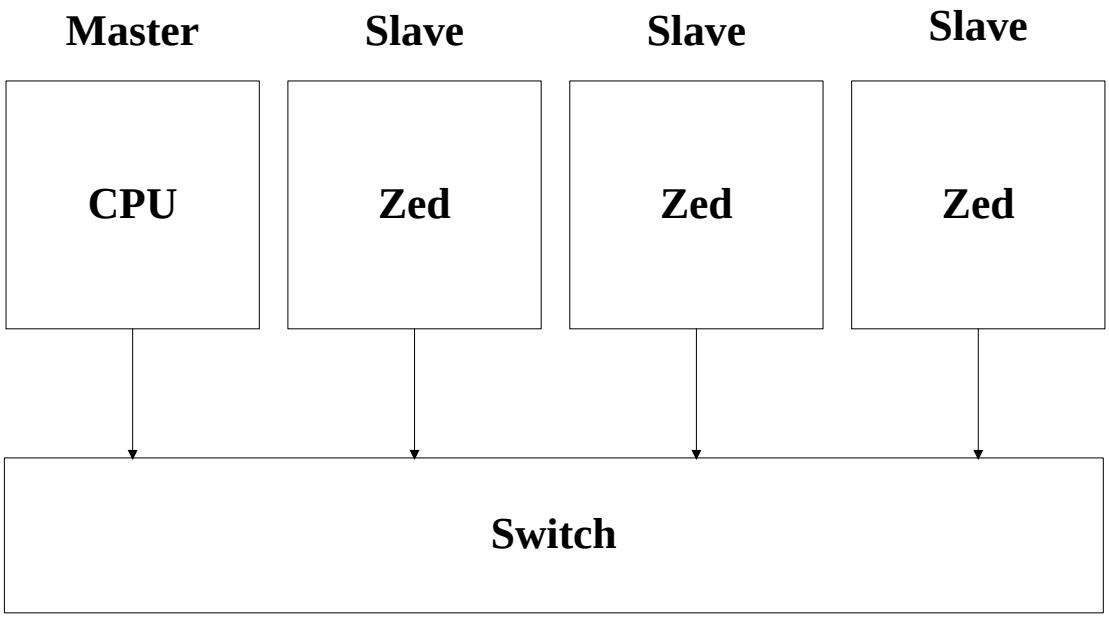


Figure 1.16: ZCluster System Overview [21]

In this paper, we use Xilinx VC707 evaluation boards (EVBs) [33] to communicate with the host computers through Gigabit Ethernet. The host computer with high performance Intel I7 processors can share the loading of applications by sending the jobs to the EVBs. We use K-means clustering algorithm and K-Nearest Neighbor algorithm implementation as examples to demonstrate the speedups of the Hadoop cluster with FPGA-based hardware accelerators.

1.4 Motivation

Nowadays, big data analytics has become the focus of research in the various fields of government, manufacturing, healthcare, media, and science because of the growing popularity of internet of things (IoT). Big data analysis has become an important marketing trend involved in a number of areas. It is important to filter out valuable information effectively and respond to the needs quickly. Hence, many software frameworks provide high scalability and fault tolerance system to enable massive data storage and distributed processing over large number of computing servers. However, the physical constraints of CPUs limit the performance in computing so we can use the hardware accelerator which is low cost, low power and high scalability to improve the performance of computation.

According to our previous study [39] as shown in Fig. 1.17, we developed the high scalability FPGA-based hardware accelerator for data-intensive computation. There were three FPGA evaluation boards (EVBs) [33] and one host computer that communicated with FPGA EVBs with Gigabit Ethernet switch. We shared the workload to three VC707 EVBs and the usage of Ethernet rate was approximate 99 percent. The Ethernet rate was reached limitation in our previous system and the bottleneck of Ethernet rate will lower the performance. Therefore, we cooperate with Hadoop system and FPGA-based hardware accelerators to achieve high scalability and high performance of computation in this thesis.

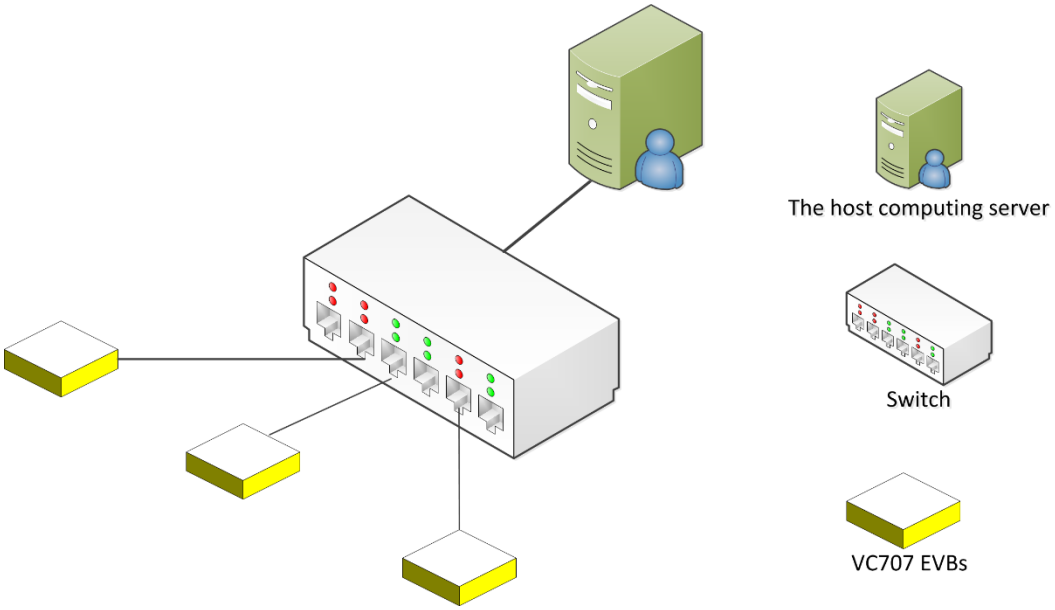


Figure 1.17: The FPGA-based hardware accelerator with VC707 EVBs [39]

In our work, we use Xilinx VC707 EVBs to communicate with the host computers through Gigabit Ethernet. The host computer with high performance processors can share the loading of applications by sending the jobs to the EVBs. The Hadoop cluster consists of one Master Node and three Slave Nodes, and in each Slave Node, it consists of one host computer and one VC707 EVB. In the Slave Node, the host computer communicates with VC707 EVB through Gigabit Ethernet. The proposed design follows MapReduce programming model to process jobs in parallel and uses Hadoop distribution file system (HDFS) for storing large dataset to manage and split files into several blocks. The proposed high-scalable heterogeneous computing solution is suitable to be applied to different machine learning algorithms for big data analytics.

Furthermore, we will introduce Hadoop cluster in our platform and FPGA architecture in Chapter 2. Then, we will depict K-means clustering algorithm and K-Nearest Neighbors algorithm in the FPGA in Chapter 3. Moreover, we will demonstrate our experiment environment and the results of speedup in Chapter 4. Finally, in Chapter 5, we will make a conclusion and discuss about the future works.

Chapter 2

Proposed Hadoop Cluster with FPGA-based Hardware Accelerator Architecture

2.1 System Overview

The proposed Hadoop cluster with FPGA-based hardware accelerators is shown in Fig. 2.1. We write the input dataset to HDFS and execute the MapReduce program in the Master Node. Then, in map stage, Hadoop sends map tasks to the appropriate servers in the Hadoop cluster. In each map task, the host computer reads the dataset from HDFS, and it wraps data into transmission packets. Subsequently, these packets are sent to the VC707 EVBs. The VC707 EVB receives packets and the hardware accelerator will perform the proposed algorithms with parallel hardware circuits. After that, the hardware accelerator implemented in the FPGA calculates and returns the partial results to the host computer. Finally, in the reduce stage, Hadoop sends reduce tasks to the appropriate servers in the Hadoop cluster. The specifications of the host computers are shown in Table 2.1. In the host computers, Hadoop with version: 2.7.1 was installed. The Hadoop cluster consists of one Master Node and four Slave Nodes, and in each Slave Node, it consists of one host computer and one VC707 EVB. The host computer communicates with VC707 EVB through Gigabit Ethernet.

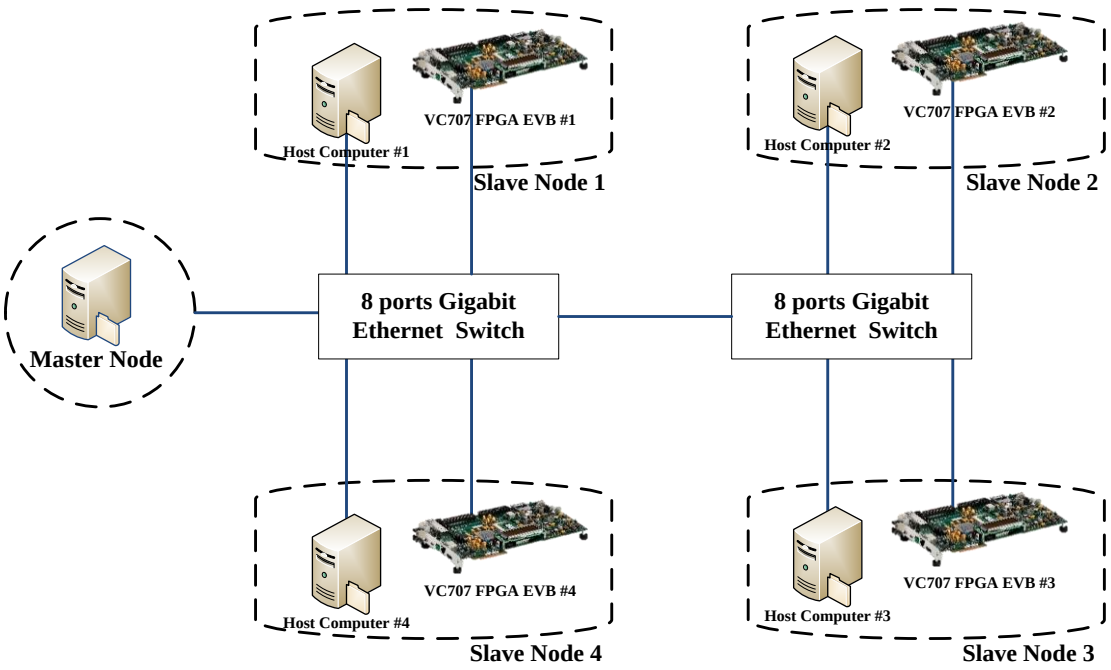


Figure 2.1: The proposed system overview

Table 2.1: The specifications of proposed system

Host Computer	CPU	Disk	OS	Hadoop Version
Master Node	Intel(R) Core(TM) i7-6700 @ 3.4GHz	SATAIII SSD 240GB	64-bit CentOS 6.7	2.7.1
Slave Node 1	Intel(R) Core(TM) i7-4790 @ 3.6GHz			
Slave Node 2	Intel(R) Core(TM) i7-4790 @ 3.6GHz			
Slave Node 3	Intel(R) Core(TM) i7-4770 @ 3.4GHz			
Slave Node 4	Intel(R) Core(TM) i7-4770 @ 3.4GHz			

The components of the VC707 EVB are shown in Fig. 2.2 and the descriptions of components we use in our design are shown in Table. 2.2.

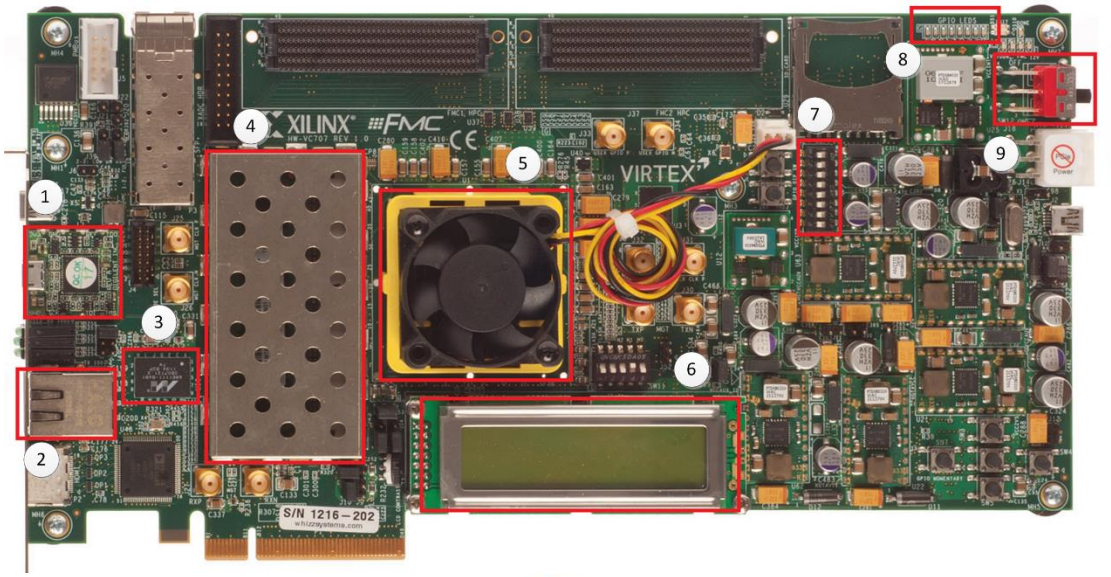


Figure 2.2: VC707 EVB [33]

Table 2.2: VC707 EVB descriptions of components

Locations	Component Description
1	USB JTAG interface
2	Network cable port
3	10/100/1000 Mb/s Ethernet PHY
4	DDR3 SODIMM memory (1 GB)
5	Virtex-7 FPGA with cooling fan
6	LCD character display
7	User DIP Switch
8	User LEDs
9	Power on/off switch

In the proposed Hadoop cluster with FPGA-based hardware accelerators, the disk I/O with reading and writing data rates of SSD are 488.57 Mbps and 448.48 Mbps, respectively. In addition, the data transmission time between the host computer and the hardware accelerator platform depends on the I/O speed limitations of the Ethernet PHY IP. The transmission data rate from the host computer to the VC707 EVB and from the VC707 EVB to the host computer are approximate 436 Mbps and 125 Mbps, respectively as shown in Table 2.3.

Table 2.3: Transmission data rate between the host computer and VC707 EVB

Direction	Packet Length (bytes)	Number of Packet	Time (s)	Transmission Data Rate
Host to VC707	1500	100,000	2.752	436.047 Mbps
VC707 to Host	1500	100,000	9.593	125.091 Mbps

2.2 Hadoop Cluster Setting

2.2.1 Configuration of HDFS

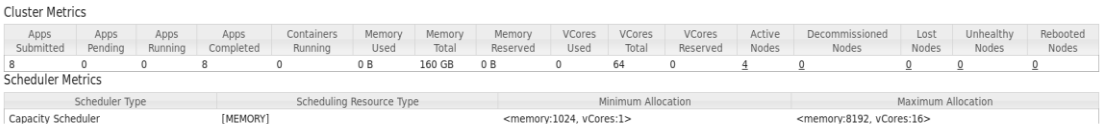
HDFS in Hadoop 2.7.1 can support for the file truncate, the quotas per storage type, and the files with variable-length blocks. The block size in Hadoop 2.7.1 is 128MB and the replication number of files is 3, as shown in Table 2.4. We allocate 40GB resource memory and provide 16 CPU virtual cores (vcores) for each node manager as shown in Table 2.5. Therefore, the total resources of the HDFS built by four Data Nodes are 160GB memory and 64 vcores as shown in Fig. 2.3.

Table 2.4: Configuration in hdfs-site.xml

Parameter Name	Value	Description
dfs.blocksize	134217728	The block size for new files, in bytes.
dfs.replication	3	The actual number of replications can be specified when the file is created.

Table 2.5: Configuration in yarn-site.xml

Parameter Name	Value	Description
yarn.nodemanager.resource.memory-mb	40960	Amount of physical memory, in MB.
yarn.nodemanager.resource.cpu-vcores	16	Number of vcores that can be allocated for containers.


Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
8	0	0	8	0	0 B	160 GB	0 B	0	64	0	4	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:16>

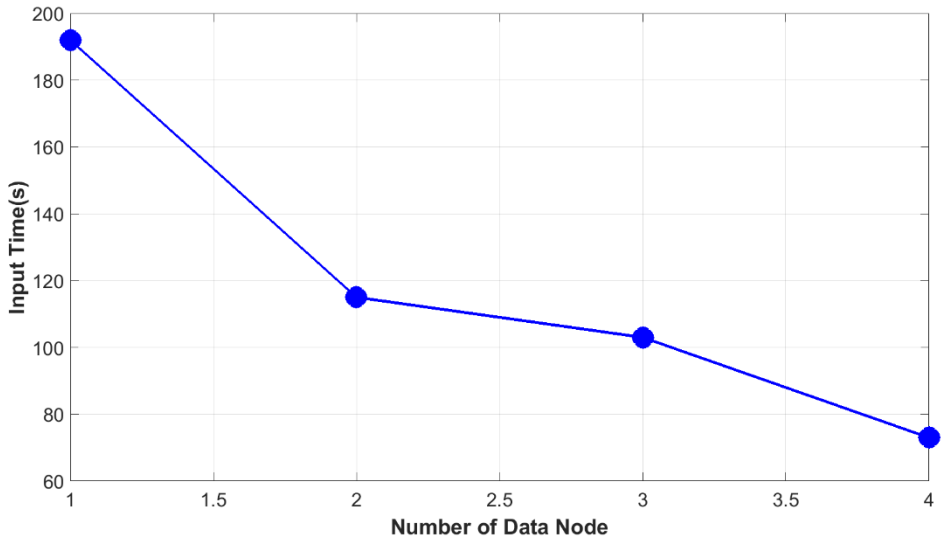
Figure 2.3: HDFS total resource overview

In the proposed Hadoop cluster, we build the HDFS for the implementation by one Name Node and four Data Nodes. We monitor the transmission data rates of uploading files from local drives to HDFS and downloading files from HDFS to local drives as shown in Table 2.6. Moreover, we compare the input time on HDFS during Hadoop Mahout processing with one to four Data Nodes. Fig. 2.4 shows the input time will decrease with increasing the Data Nodes to demonstrate the HDFS parallel progressing.

Table 2.6: Transmission data rate on HDFS with 4 Data Nodes

Direction	File Size	Time	Transmission Data Rate
Upload File to HDFS	3.18 GB (3424997624 bytes)	34.041 s	95.659 MB/s
Download File from HDFS	3.18 GB (3424997624 bytes)	19.854 s	164.013 MB/s

Figure 2.4: Input time of Mahout processing with Data Nodes



2.2.2 MapReduce of Proposed Architecture

The MapReduce of our proposed architecture is as shown in Fig. 2.5. We connect the Data Node to the VC707 EVB through Gigabit Ethernet to accelerator implementations during each mapping. We can share the workload of computation to each node and achieve the speedup of the execution. When the Name Node receives the commands from users, the Name Node will start to map the jobs to each Data Node by the configuration of HDFS. In the map stage, we will separate the programs to software and hardware by the complex of computations which will be discussed in Chapter 3 for each of the proposed algorithms. The outputs of map stage are <Key, Value> pairs to be sent to the reduce stage. In the reduce stage, after all jobs are done successfully, the final results will be summarized as the outputs on HDFS by the reducers.

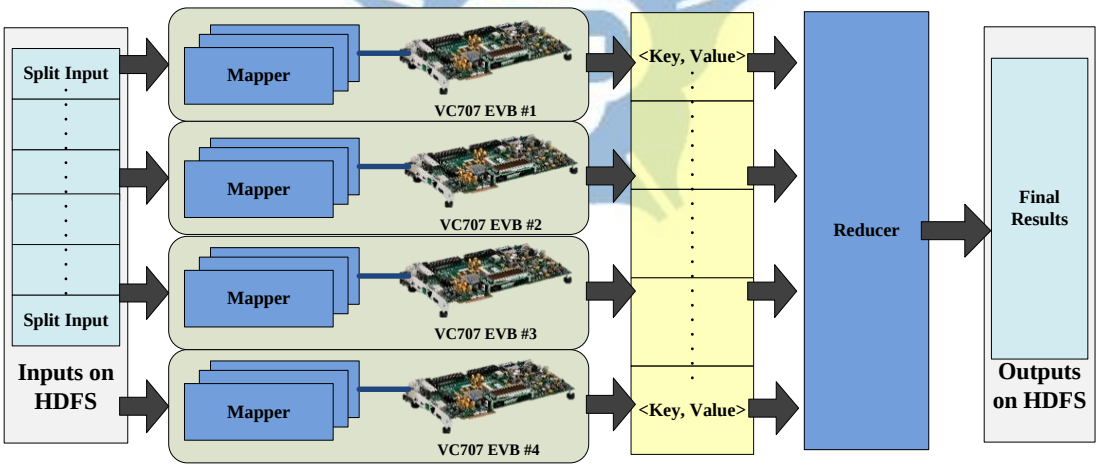


Figure 2.5: The MapReduce of proposed architecture

2.2.3 Hadoop Streaming

Although the Hadoop framework is implemented in Java, MapReduce applications need not be written in Java. Hadoop Streaming is a utility which allows users to create and run jobs with any executable as the mapper and the reducer. The executable reads the standard inputs from STDIN, writes outputs to STDOUT, and produces results to standard inputs as shown in Fig. 2.6. The utility will create a MapReduce job, submit the job to a cluster, and monitor the progress of the job until it completes. We can use the Hadoop tool, Hadoop Streaming [45], to help us execute the C/C++ programs which we implement on Hadoop system.

As the mapper task, the mapper converts the inputs into lines and feeds the lines to the STDIN of the process. In the meantime, the mapper collects the line oriented outputs from the STDOUT of the process and converts each line into a <key, value> pair as the output of the mapper. On the other hand, as the reducer task, the reducer converts the input <key, value> pairs into lines and feeds the lines to the STDIN of the process. Meanwhile, the reducer collects the line oriented outputs from the STDOUT of the process, converts each line into <key, value> pairs as the output of the reducer.

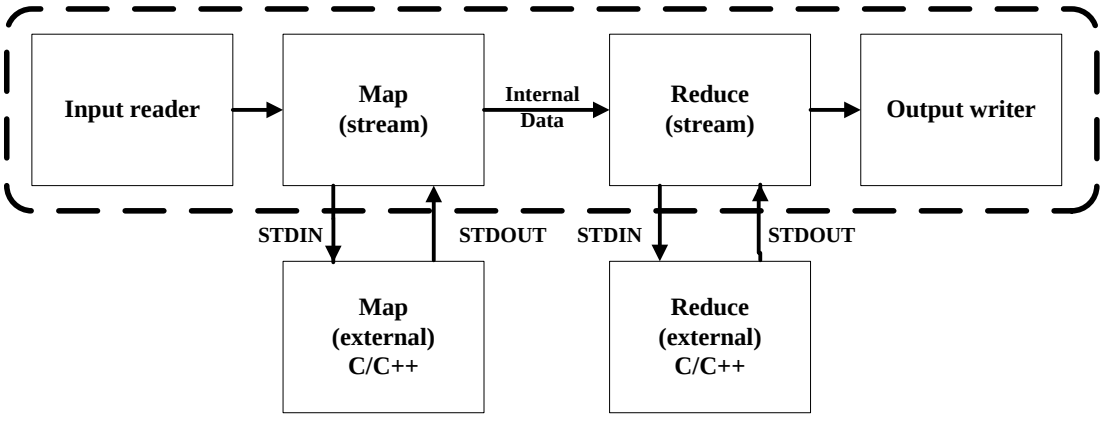


Figure 2.6: Hadoop Streaming behavior

2.3 Hardware Accelerator development

2.3.1 Architecture of VC707 EVB

Fig. 2.7 shows a Slave Node is composed of one host computer and one VC707 EVB connected by Ethernet. We implement three modules in FPGA including Ethernet Physical IP, Mixed-Mode Clock Manager (MMCM), and user code. The host computer wraps data into packets and sends them to VC707 EVB through a Gigabit Ethernet switch and the VC707 EVB sends back the computation results as packets to host computer through Ethernet Physical IP which connect to the Gigabit Ethernet switch. The proposed computation data mining algorithms are implemented in the user code. The MMCM creates dynamic reconfiguration of the phase, duty cycle, and clock output frequency.

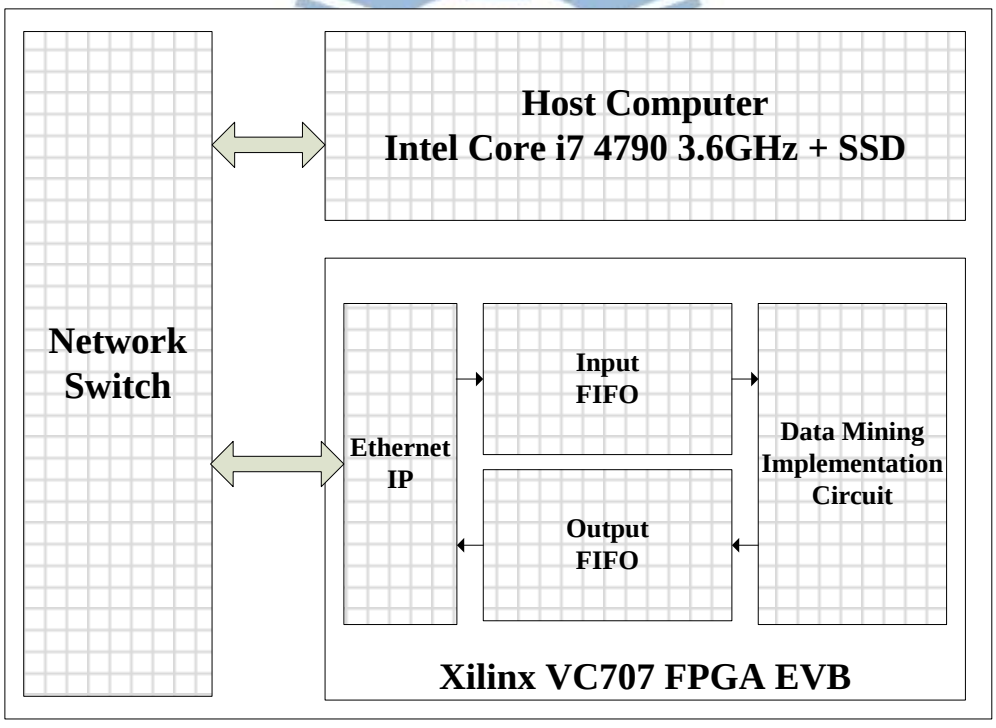


Figure 2.7: One Slave Node Implemented Module

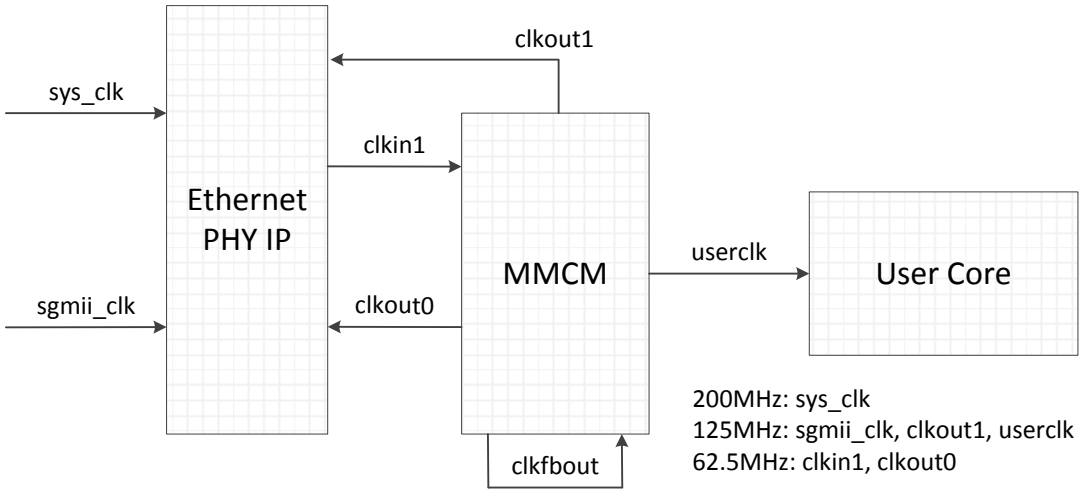


Figure 2.8:VC707 EVB clock distribution

The VC707 EVB clock distribution is shown in Fig. 2.8. The system clock named sys_clk is a 200MHz differential signal pair including SYSCLK_P and SYSCLK_N to provide a high-speed clock for the top module. The SGMII clock named sgmii_clk is a 125MHz differential signal pair including SGMIICLK_Q0_P and SGMIICLK_Q0_N to provide a reference clock of the high-quality and low-jitter for the Ethernet PHY IP. The block of Ethernet PHY IP generates a clock of 62.5 MHz as the reference clock of the MMCM named clkin1. The MMCM is a phase-locked loop (PLL) IP. After the MMCM aligns phase of clkin1, it multiplied the clkfbout by 10 times and the MMCM divided clkfbout by 10 times and 5 times. Then, the MMCM generates two different frequency clocks of 62.5MHz and 125MHz, respectively. The clock of clkout0 is 62.5 MHz to be the transmission clock of the Ethernet PHY IP. The clocks of userclk and clkout1 are 125MHz. The userclk is used as an operation clock for the user core and the clkout1 is used for physical medium attachment (PMA) IP module and serial-gigabit media independent interface (SGMII) module inside the Ethernet physical IP.

2.3.2 Ethernet PHY Controller

Xilinx provides the Ethernet PHY IP to make it easy to use the Ethernet protocol for the transmission. The format of the transmission packet is shown in Fig. 2.9.

	Preamble	SFD	DA	SA	length	data	FCS
length (bytes)	7	1	6	6	2	1500	4

Figure 2.9: Transmission packet format

- Preamble: The preamble consists of 7-byte allowing devices on the network to easily synchronize their receiver clocks.
- Start frame delimiter (SFD): The SFD is one-byte which is the first field of an Ethernet packet.
- Destination address (DA): The DA contains target MAC address of packet.
- Source address (SA): The SA contains source MAC address of packet.
- Length: Length is data length which contains 2 bytes.
- DATA: The data contain 1500 bytes for data transmission.
- Frame check sequence (FCS): The FCS contains 4 bytes cyclic redundancy check code to verify that whether the data frame is damaged or not.

The RX FIFO module operation is shown in Fig. 2.10. The counter is aligned the receiving data (rx_d) when the signal of the receiver (rx_en) is enabled (rx_en). The RX FIFO module outputs the receiving data to the user core module after checking the destination address (DA) and the source address (SA) are correct. Finally, we assert the signal (rx_finish) when the reception has be done.

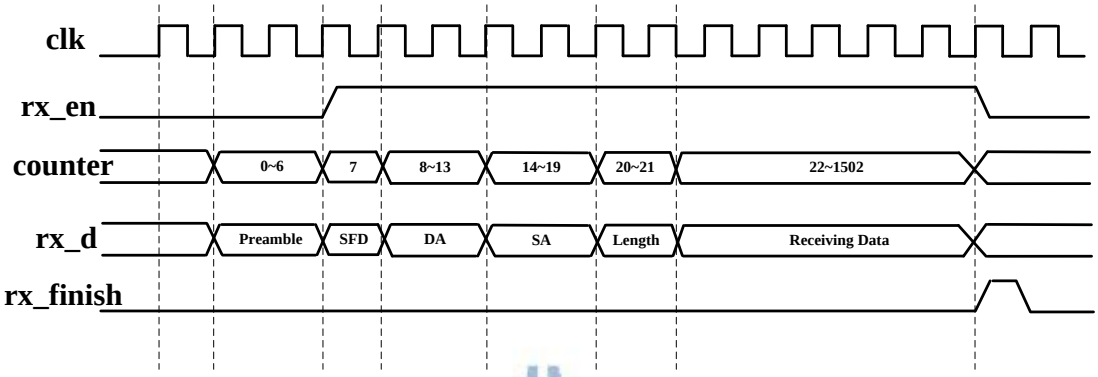


Figure 2.10: Behavior in RX FIFO module

The TX FIFO module behavior is similar to the RX FIFO module as shown in Fig. 2.11. We assert the transmission signal (tx_en) after the user core operation is finished. Subsequently, we wrap the data into packets when the signal (tx_en) is high. Finally, the packets are sent to the host computers through Ethernet PHY IP.

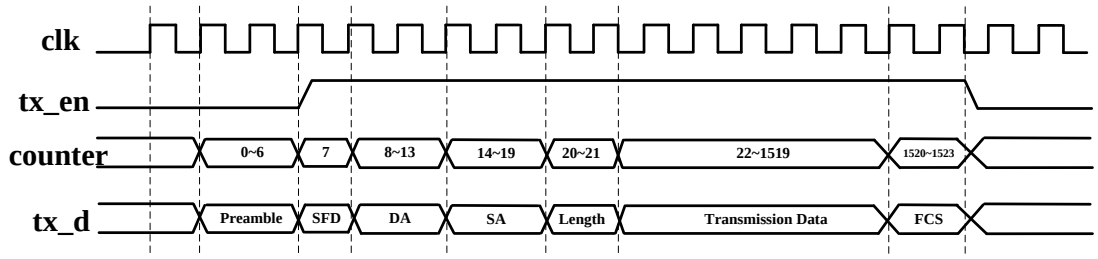


Figure 2.11: Behavior in TX FIFO module

2.4 Ethernet Connection Library in Linux

For the Ethernet connection between Hadoop cluster and VC707 EVBs in our implementation, we use a portable C/C++ library which is the packet capture library (libpcap) for network traffic capture. This library provides a consistent C functions programming interface for different platforms and users can freely call these functions to program applications. The structure of libpcap is simple and easy to use. We can use these API functions to complete the connection between Hadoop nodes and VC707 EVBs. The three main libpcap function we use in our implementation are *pcap_open_live(), pcap_next_ex(), pcap_sendpacket ().*

- pcap_t *pcap_open_live (const char *device, int snaplen, int promisc, int to_ms, char *errbuf);
- int pcap_next_ex(pcap_t *p, struct pcap_pkthdr **pkt_header, const u_char **pkt_data);
- int pcap_sendpacket(pcap_t *p, const u_char *buf, int size);

We use *pcap_open_live()* to open a device for capturing our transmission packets on the network. When *pcap_open_live()* returns a packet capture handle on success, we can connect to the VC707 EVBs successfully. In addition, we use *pcap_next_ex()* for receiving data and *pcap_sendpacket ()* for transmitting data through the network interface.

2.5 Brief Summary

We will integrate the Hadoop system and the FPGAs as above to achieve the high scalability and high performance of computation system. The HDFS provides high throughput to read and write data, MapReduce supports the applications execute in parallel, Hadoop Streaming assists us to program in C/C++, and the libpcap library serves the Ethernet to connect with the Ethernet PHY IP in VC707 EVBs. In addition, the MMCM in VC707 EVBs helps the system to easily generate the clock signals the architecture needs.

In order to take the advantages of software and hardware, we separate the programs of the proposed algorithms as the mappers and the reducers based on the characteristic of MapReduce, and implement the repetition of the iterations in FPGAs to accelerate the operations.

The purpose of the proposed architecture is to speed up the operations to demonstrate the contribution of the hardware accelerator. The analyses to separate the proposed algorithms to software and hardware are important tasks. Therefore, in next chapter, we will focus on the implementation of the designs of K-mean clustering algorithm and K-nearest neighbor algorithm in VC707 EVBs.

Chapter 3

Acceleration of Data Mining Algorithms

3.1 Introduction to Data Mining

Data mining is the computational process of analyzing, discovering patterns in big data from different perspectives and summarizing them into useful information. There are many ways to dig out the value of big data. In an effort to identify some of the most influential algorithms that have been widely used in the data mining community, the ICDM [46] identified the top 10 algorithms in data mining. The top 10 algorithms cover classification, clustering, statistical learning, association analysis, and link mining, which are all among the most important topics in data mining research and development. K-means clustering algorithm and K-nearest neighbor classification algorithm are two of the top 10 algorithms.

The K-means algorithm is a simple iterative method for data clustering in a large number of high-dimensional data sets. The K-nearest neighbor algorithm is a non-parametric method of the basic classifiers for pattern recognition or data classification. The properties of two algorithms are intensive computation and non-sequential jobs which are suited for hardware accelerations.

3.2 Acceleration of K-means Clustering Algorithm

3.2.1 Introduction to K-means Clustering Algorithm

K-means clustering is the most popular data mining algorithm and is used in image processing and machine learning. The goal of K-means algorithm is to partition the input data into the number *k* of clusters. The K-means clustering algorithm operates on a set of *D*-dimensional set $X = \{x_n \epsilon R^D, with\ n = 1, \dots, N\}$, and partitions *X* into *k* ($k \leq N$) clusters, where N is the total number of the input data points. The end result is a set of *D*-dimensional centroids for the clusters $C = \{C_1, C_2, \dots, C_k\}$, each cluster is associated with the center value. Therefore, the output of objective function in Euclidean distance is $\sum_{i=0}^{k} \sum_{x \in C_i} ||x - u_i||^2$, where u_i is the mean of points (center) in C_i.

The K-means clustering flow is described as following. Firstly, the number *k* of clusters and the initial cluster centers are determined. Then, the objects are partitioned to the nearest cluster by calculating the Euclidean distance between the centers and each D-dimensional point. Subsequently, the new center values are computed by calculating the mean of the objects in the clusters to replace the previous centers. After that, the iteration of partitioning object to the nearest cluster and calculating the new centers has been repeated until the new cluster centers are approximated. Finally, when the new cluster sets are all assigned to the nearest cluster, the K-means clustering is finished.

3.2.2 Software Implementation

We prepare the three-dimensional input data by MATLAB and upload the files to HDFS. In addition, we implement the initialization and calculate the new cluster centers in MapReduce program.

Algorithm 1 shows the pseudo code of K-means mapper. In each map task, the mapper reads the partial datasets which are three-dimensional floating-point data nodes and four cluster three-dimensional floating-point centers from HDFS. When the inputs are ready, the mapper connects to the VC707 and wraps the MAC addresses of the destination and the source into the transmission packet. Moreover, the mapper wraps the 4 cluster centers and 115 nodes into transmission packet. Then, the host computer sends the whole transmission packet to the VC707 EVB, and then the host computer receives the packet of the partial results from VC707 EVB. Finally, the mapper constructs the outputs as <key, value> pairs to the reduce stage.

Algorithm 2 presents the pseudo code of K-means reducer. In the reduce stage, the reducer adds the partial sum of each dimension and the partial amount of nodes in the clusters from the mapper. Then, the reducer calculates the new cluster centers for the next iteration and update the cluster center file to HDFS.

- -

Algorithm 1 : K-means Mapper

- -

Input: *partial 3-d floating-point nodes (32bits* N *3) and 4 cluster 3-d floating-point centers (32bits*4*3=384bits)*

Output: *partial <key, value> pairs, where the key is the index of cluster and the value is the partial sum of 3-d floating-point (32bits*4*3=384bits) with the amount of nodes in the clusters (32bits*4*3=384bits)*

max_number_of_cluster is 4

max_node_count is 115

number of input nodes is N

```
receive input data from HDFS and calculate the number N of input nodes;
connect to the VC707 EVB;
wraps MAC address into the transmission packet;
for row = 0 to N do
        wrap the 4 cluster centers into transmission packet;
        do {
                wrap the data nodes into transmission packet;
                row = row + 1;
        } while (row % 115 != 0);
        send the packet to VC707 EVB;
        receive the packet of the partial results from VC707 EVB;
end for
// construct the outputs as <key, value> to reducer
for index = 0 to max_number_of_cluster, index++ do
        output <index, partial results and the number of nodes in the clusters> as <key, value> pair
end for
```

- -

Algorithm 2 : K-means Reducer

- -

Input: *<key, value> pairs of all mappers, where the key is the index of cluster and the value is the partial sum of 3-d floating-point (32bits*4*3=384bits) with the number of amount in the clusters (32bits*4*3=384bits)*

Output: *4 new cluster 3-d floating-point centers (32bits*4*3=384bits) to HDFS*

max_number_of_cluster is 4

```
receive input data <key, value> from all of mapper;
do {
        calculate x coordinate total sum of each cluster
        calculate y coordinate total sum of each cluster
        calculate z coordinate total sum of each cluster
        calculate total amout of each cluster
} while (inputs != NULL)
// Update new 4 cluster centers
for index = 0 to max_number_of_cluster do
        x of new cluster center = the sum of x coordinate / the amount of x coordinate;
        y of new cluster center = the sum of y coordinate / the amount of y coordinate;
        z of new cluster center = the sum of z coordinate / the amount of z coordinate;
end for
write the 4 new cluster centers into HDFS;
```

3.2.3 Circuit Design in FPGA

We separate the computations of calculating the Euclidean distance and clustering to the VC707 EVB. **Algorithm 3** [39] shows the pseudo code of the K-means clustering algorithm in the VC707 EVB.

In K-means clustering circuit, the inputs are 115 nodes three-dimensional single precision floating point coordinate values and the initial center coordinate values for four clusters from the host computer. When the K-means clustering circuit receives the coordinate values of 115 nodes, it calculates the Euclidean distances of each node to the cluster centers with parallel hardware circuits and finds the shortest distances to the cluster centers. After that, the nodes can be grouped into four clusters. When the input nodes are all grouped into clusters, the number of nodes in each cluster and the partial sum of the coordinate values of each dimension in each cluster are sent back to the host computer. Then, the host computer sends another 115 nodes to the VC707 EVB until all input dataset are grouped by the proposed FPGA-based hardware accelerator. In addition, the host computer calculates new centers for the next iteration by accumulating the partial sum and the number of nodes in each cluster which are sent by the VC707 EVB.

- -

Algorithm 3 : K-means clustering algorithm in VC707 EVB

- -

Input: *3-d 115 floating-point nodes (32 bits*3*115=11040 bits) and 4 cluster floating-point centers (32 bits*4*3=384bits)*

Output: *the registers cluster_value[3][2] store clusters coordinate value of 3-d floating-point (32 bits*4*3=384 bits) and the registers cluster_amount[3] store number of nodes in each cluster (96 bits*4=384 bits)*

max_number_of_cluster is 4

max_node_count is 115

1: *receive packet input from host computer;*

2: *store input in the node data registers and center data registers in sequential;*

3: **for***(cluster_count=1; cluster_count <= max_number_of_cluster; cluster_count++)* **begin**

4: *set_cluster_center(cluster_count);*

5: *calculate Euclidean distance to cluster centers in total 115 nodes;*

6: *save 115 Euclidean distances from 115 nodes;*

7: **end**

8: *In 115 nodes with 4 Euclidean distance, find the shortest distance from 115 nodes;*

9: **for***(node_count=1; node_count <= max_node_count; node_count++)* **begin**

10: **if***(cluster number of node == cluster number "n" of node with shortest distance)* **then**

11: *cluster_value[n][0] = cluster_value[n][0] + node_x_coordinate;*

12: *cluster_value[n][1] = cluster_value[n][1] + node_y_coordinate;*

13: *cluster_value[n][2] = cluster_value[n][2] + node_z_coordinate;*

14: *cluster_amount[n] = cluster_amount[n] + 1;*

16: **end**

17: end

18: if *dataset is complete transmission* **then**

19: *wrap the registers cluster_value[3][2] and the registers cluster_amount[3] into packet and send to host computer;*

20: *reset the registers cluster_value[3][2] and the registers cluster_amount[3] value for next iteration;*

21: end

The Euclidean distance equation is shown in Eq. 3.1.

$$\text{distance} = \sqrt[2]{(node_x - center_x)^2 + \left(node_y - center_y\right)^2 + (node_z - center_z)^2}$$

Eq. 3.1

We implement the Euclidean distance with 115 group parallel hardware circuits as shown in Fig. 3.1 by Xilinx IP core generators including the IP cores of floating-point subtractors, floating-point adders, floating-point square circuits, and floating-point square root circuits.

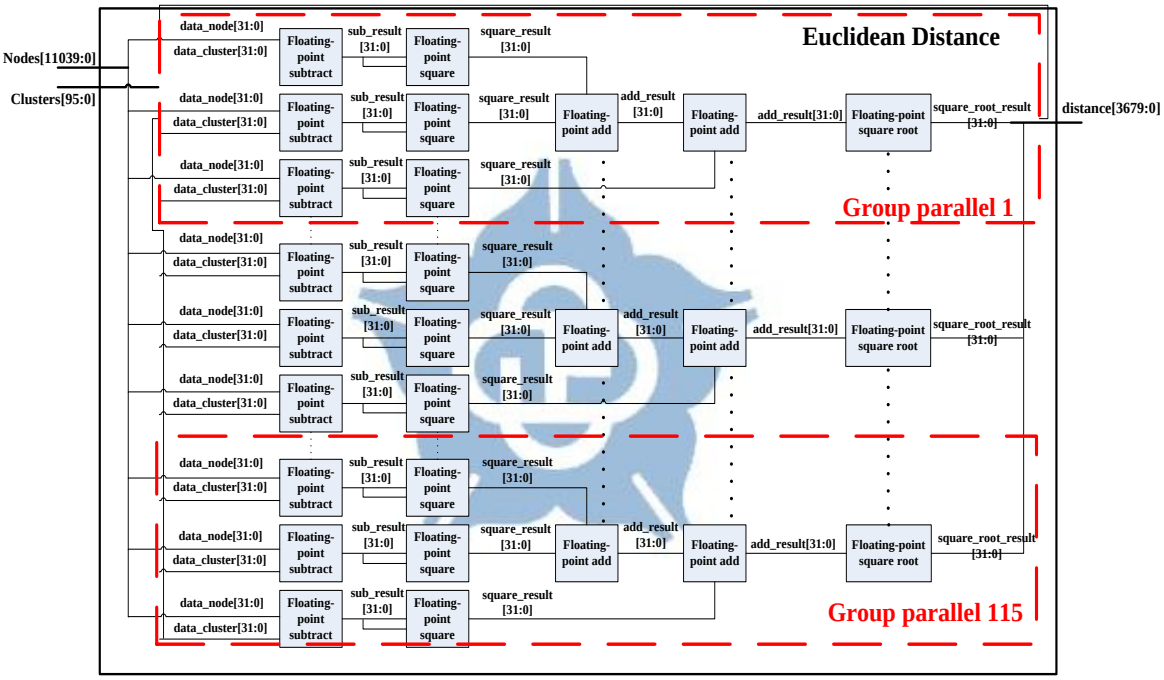


Figure 3.1: Euclidean distance parallel hardware circuits between one three-dimensional cluster center node and 115 three-dimensional nodes. (Node: 115 nodes x 32bits = 11040 bits; Cluster: 1 cluster center x three-dimensional x 32bits = 96 bits)

3.3 Acceleration of K-Nearest Neighbor Algorithm

3.3.1 Introduction to K-Nearest Neighbor Algorithm

The K-nearest neighbor (KNN) algorithm is also one of the most popular data mining algorithms. The KNN algorithm is mostly used in pattern recognition to solve regression or classification problem. The goal of KNN algorithm is to find the K-closest neighboring training samples to the object and classify the object by a majority vote of its K nearest neighbors to assign the object to the class, K is a user-defined constant. The training sample data are given the set $S = \{(x_1, y_1), \ldots (x_i, y_i), \ldots, (x_n, y_n)\}$, where x is the feature vector and y is the corresponding target class. In order to find the nearest neighbors, a commonly used distance is the Euclidean distance. An unlabeled vector is classified by assigning the label y_i which is the most frequent among the K training samples *S* nearest to the query object.

The KNN algorithm flow is described as following. Firstly, the number of K and training samples are assigned, and the target test data are defined. Then, the distances between the training samples and the target test data are calculated. Finally, when the target test data are classified by finding the K shortest distances of the training samples and voted by the weight of the samples, the KNN algorithm is finished.

3.3.2 Software Implementation

The KNN algorithm software implementation is also divided into the mapper stage and the reducer stage. **Algorithm 4** shows the pseudo code of the KNN algorithm mapper. We use the same of K-means dataset by MATLAB and also send the data into HDFS. When the inputs are ready, the mapper connects to the VC707 and wraps the MAC addresses of the destination and the source into the transmission packet. Moreover, the mapper wraps the test data and 115 sample nodes into transmission packet. Then, the host computer sends the whole transmission packet to the VC707 EVB, and receives the packet of the partial results from VC707 EVB. Finally, the mapper constructs the outputs as <key, value> pairs to the reduce stage.

Algorithm 4 : KNN algorithm Mapper

Input: *partial 3-d floating-point sample nodes (32bits*N*3) and one 3-d floating-point test data (32bits*1*3=96bits)*

Output: *partial <key, value> pairs, where the key is the Euclidean distance (32bits*K), and the value is the label of the sample data (32bits*K)*

```
1:  receive input data from HDFS, calculate the number N of input nodes and give labels to the input
        data;
2:  connect to the VC707 EVB;
3:  wraps MAC address into the transmission packet;
4:  for row = 0 to N do
5:      wrap the test data into transmission packet;
6:      do {
7:          wrap the sample nodes into transmission packet;
8:          row = row + 1;
9:       } while (row % 115 != 0);
10:      send the packet into VC707 EVB;
11:      receive the packet of the K shortest Euclidean distances from VC707 EVB;
12: end for
13: // construct the outputs as <key, value> to reducer
14: for index = 0 to K do
15:      output <the Euclidean distance, label>;
16: end for
```

Algorithm 5 presents the pseudo code of the KNN algorithm. In the reduce stage, the reducer will get the sorted Euclidean distance values and the label of the sample dataset from the mapper. Then, the reducer votes by the weight of the samples to determine where the test data belongs to.

Algorithm 5: KNN algorithm Reducer

Input: *partial <key, value> pairs, where the key is the Euclidean distance (32bits*K), and the value is the label of the sample data (32bits*K)*

Output: *Show the test data which class it belongs to.*

K is a user-defined constant;

1: *receive input data <key, value> from all of mapper;*

2: for *i = 0 to K* do

3: *find the max frequency of the labels;*

4: end for

5: *output the label which the test data belongs to;*

3.3.3 Circuit Design in FPGA

We implement the Euclidean distance calculation circuits in the VC707 EVB. **Algorithm 6** shows the pseudo code of the KNN algorithm in the VC707 EVB. In KNN circuit, the inputs are 115 nodes three-dimensional single precision floating point coordinate values and the test data coordinate values from the host computer. When the KNN circuit receives the coordinate values of 115 nodes, it calculates the Euclidean distances of each sample data to the test data and sorts with parallel hardware circuits and the 115 Euclidean distance are sorted by floating-point compare IP cores with parallel hardware circuits. After that, the K shortest distances will be sent back to the host computer. Then, the host computer sends another 115 nodes to the VC707 EVB until all input dataset are grouped by the proposed FPGA-based hardware accelerator.

Algorithm 6 : KNN algorithm in VC707 EVB

Input: *3-d 115 floating-point sample nodes (32 bits*3*115=11040 bits) and 1 floating-point test data (32 bits*1*3=96bits)*

Output: *the register of K shortest Euclidean distances (32bits*K)*

max_node_count is 115

1: *receive packet input from host computer;*

2: *store input in the node data registers and center data registers in sequential;*

3: *set_cluster_center(cluster_count);*

4: *calculate Euclidean distance to the test data in total 115 nodes;*

5: *save 115 Euclidean distances from 115 nodes;*

6: *In 115 node Euclidean distances, find the K shortest distance from 115 node Euclidean distances;*

7:: *sorting and keep the K shortest Euclidean distances;*

8: *wrap the registers K shortest Euclidean distances into packet and send to host computer;*

9: *reset the registers of Euclidean distances value for next iteration;*

Chapter 4

Experimental Results

4.1 K-means Clustering Experimental Results

Table 4.1 shows the hardware resource utilization of the proposed hardware accelerator for K-means clustering algorithm.

Table 4.1: FPGA resource utilization

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	125,761	607,200	20%
Number of Slice LUTs	234,713	303,600	77%
Number of Occupied Slices	70,535	75,900	92%
Number of DSP48E1s	2,185	2,800	78%

Fig. 4.1 shows the total time required for K-means clustering algorithm with different size of input dataset. The proposed Hadoop streaming with FPGA-based hardware accelerators can achieve 3x speedup than the Hadoop cluster using Mahout machine learning libraries [19] with one Master Node and four Slave Nodes configuration.

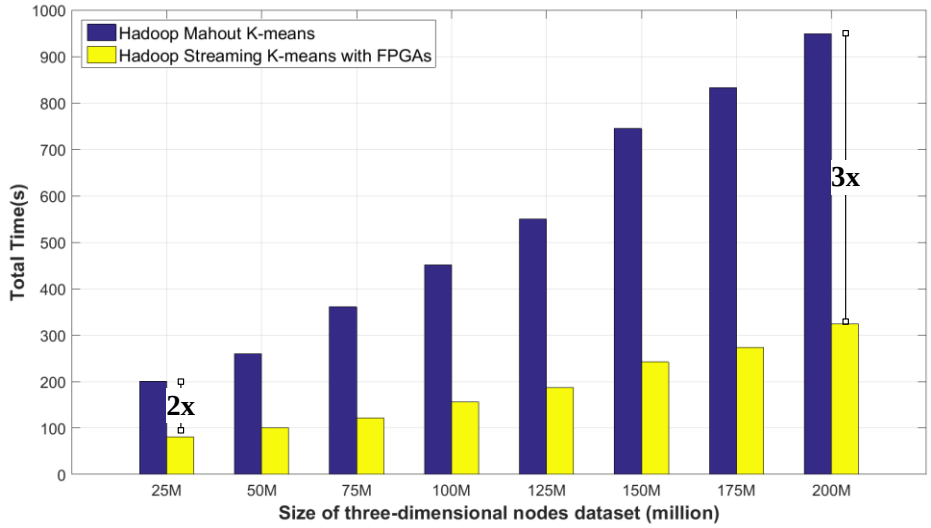


Figure 4.1: Compare Hadoop cluster with FPGA-based accelerators with Mahout libraries on 1 Master Node + 4 Slave Nodes.

Fig. 4.2 shows the total time required for K-means clustering algorithm with different number of Slave Nodes. When the number of Slave Nodes is increased, the total time required for K-means clustering algorithm can be reduced accordingly. Moreover, it also shows the performance improvement saturation when the number of Slave Nodes is increased.

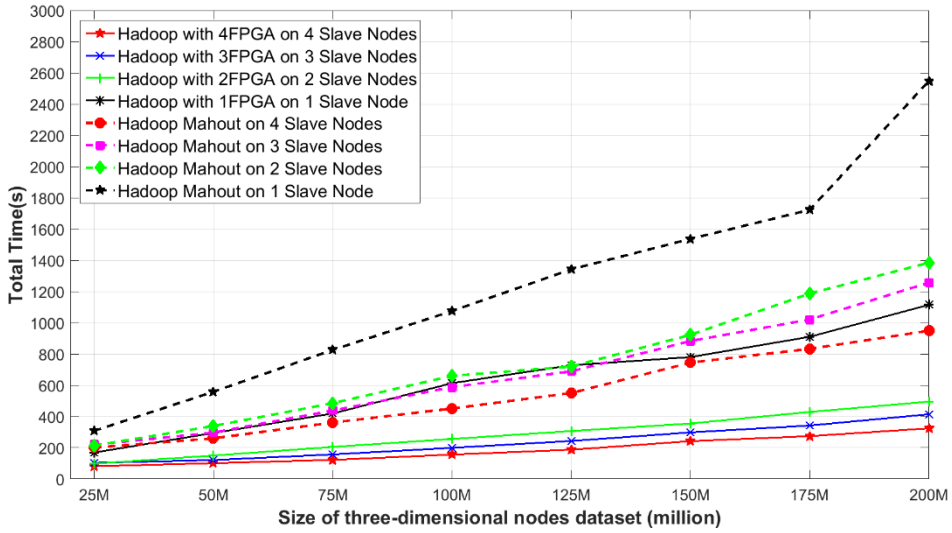


Figure 4.2: Hadoop cluster with 1/2/3/4 Slave Nodes

Fig. 4.3 presents that the speedup of our proposed Hadoop cluster with FPGA-based hardware accelerators can be achieved the speedup of 21x than Intel i5-3230M at 2.6GHz [39]. In our proposed architecture, the I/O latency has been reduced because we use SDD in each host computer and store dataset on HDFS to raise the disk I/O rate. In addition, Hadoop MapReduce supports the parallel computation and HDFS provides the high throughput to contribute the speedup.

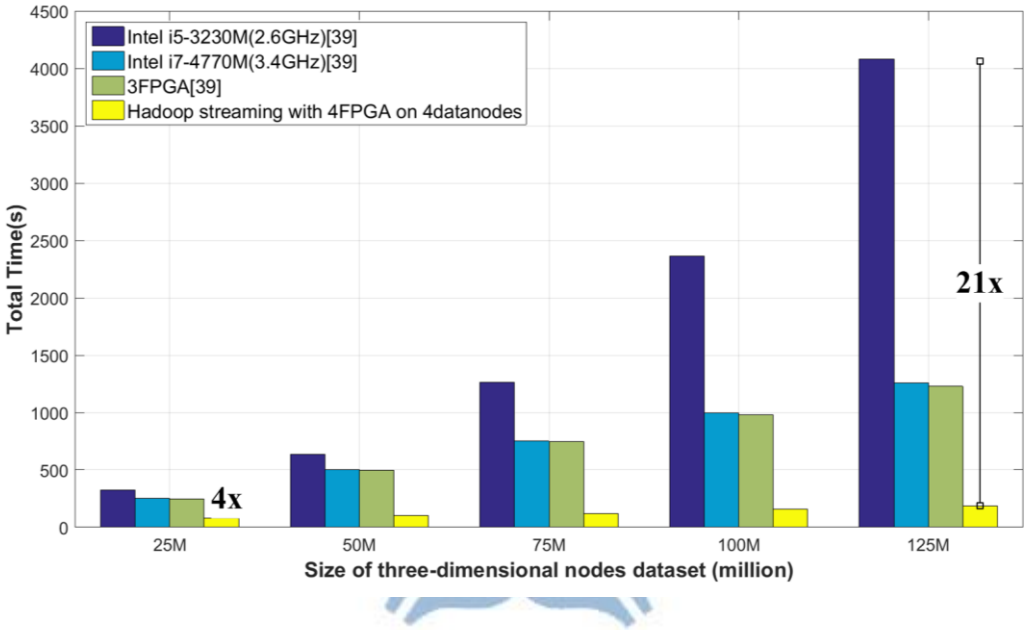


Figure 4.3: Compare proposed results with [39]

4.2 K-Nearest Neighbor Experimental Results

Table 4.2 shows the hardware resource utilization of the proposed hardware accelerator for KNN algorithm.

Table 4.2: FPGA resource utilization

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	101,312	607,200	16%
Number of Slice LUTs	216,511	303,600	86%
Number of Occupied Slices	72,965	75,900	96%
Number of DSP48E1s	2,185	2,800	78%

Fig. 4.4 shows the compared results of the KNN algorithm between Hadoop cluster without FPGA-based hardware accelerators and Hadoop cluster with FPGA-based hardware accelerators. The Hadoop environment described in Table 2.1 is composed of the Intel I7 CPU and the SSD. In the design of KNN algorithm, we implement the calculation of Euclidean distance and matrix comparisons sorting in the VC707. We accelerate the calculation of Euclidean distance and sorting in the VC707 EVB by parallel hardware circuits. The speedup of the hardware accelerator can achieve 3.5x than Hadoop cluster without FPGAs.

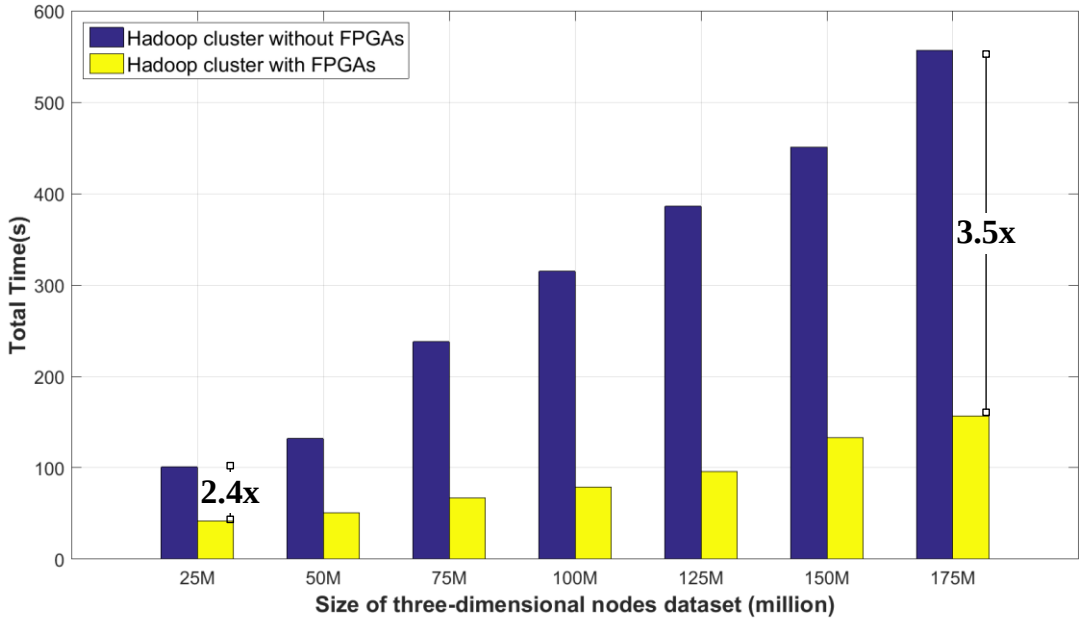


Figure 4.4: Compare Hadoop cluster without FPGAs with Hadoop cluster with FPGAs on 1 Master Node + 4 Slave Nodes

Fig. 4.5 shows the total time required for KNN algorithm with different number of Slave Nodes. When the number of Slave Nodes is increased, the total time required for KNN clustering algorithm can be reduced accordingly. Moreover, it also shows the performance saturation when the number of Slave Nodes is increased.

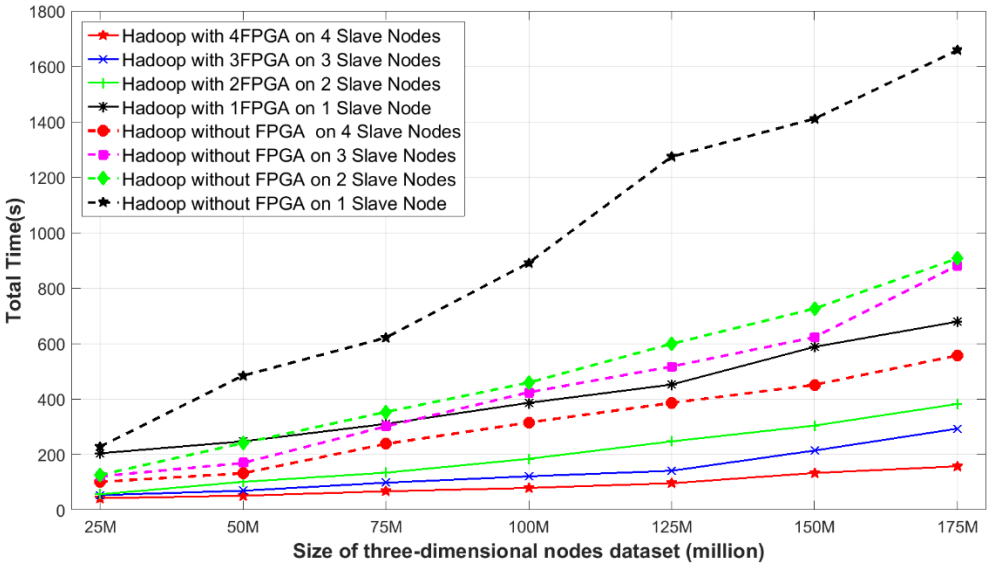


Figure 4.5: Hadoop cluster with 1/2/3/4 Slave Nodes

Chapter 5

Conclusion and Future Works

5.1 Conclusion

In this thesis, the proposed Hadoop cluster with FPGA-based hardware accelerators for K-means clustering algorithm and K-nearest neighbor algorithm can share the loading in the Slave Nodes to the VC707 EVBs. The Hadoop cluster provides the parallel processing to speed up the computations of the massive datasets.

The experimental results show that for clustering three-dimensional input dataset, the proposed design in K-means clustering algorithm can achieve 3x speedup than the Hadoop cluster without FPGA-based hardware accelerators and the implementation in K-nearest neighbor algorithm can achieve 3.5x speedup than Hadoop cluster without FPGA-based hardware accelerators. As a result, the proposed high-scalable heterogeneous computing solution is suitable to be applied to different machine learning algorithms for big data analytics.

5.2 Future Works

In this thesis, we communicate with the Hadoop cluster and VC707 EVBs through Ethernet. The Ethernet PHY clock is subject to the specification of VC707 EVBs that is the obstruction to speed up while transmitting data packets. Therefore, we can consider the other way to communicate with the host computer and VC707 EVBs like PCI Express (PCI-E). The PCI-E is a high-speed serial computer expansion bus standard and has high bandwidth to provide higher system bus throughput. We can take the advantage of the PCI-E to speed up the data transmission rate. Besides, we can increase the system clock rate for whole hardware accelerator to speed up the operations. Thus, we should rebuild and redesign the architecture of the VC707 EVBs to achieve the speedup.

Moreover, we can find the new software platform like Spark that the performance of iterations is higher than Hadoop. Spark can achieve the speedup faster than Hadoop because Spark perform many operations in memory. The Spark RDD technique effectively solves the I/O latency during processing. In addition, the open-source hardware is the future trend in recently years, we expect our hardware implementations can be more reconfigurable that users can easily modify some parameters to execute the programs instead rewriting the source code for their needs.

However, the integration of software and hardware is the current trend in these years, we committee to develop a platform which is more flexible to achieve high scalability and high performance for different requirements.

Reference

[1] John Gantz and David Reinsel, “The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the Far East,” *in Proceedings of IDC iView, IDC Analyze the Future,* Dec. 2012, pp. 1-16.

[2] Doug Laney, “3D data management: controlling data volume, and variety,” Appl. Delivery Strategies Meta Group (949), Feb. 2001.

[3] Diya Soubra, “The 3Vs that define big data,” posted on Jul. 5, 2012, http://www.datasciencecentral.com/forum/topics/the-3vs-that-define-big-data

[4] Gartner, “Big Data”, available: http://www.gartner.com/it-glossary/big-data/

[5] Udaigiri Chandrasekhar, Amareswar Reddy and Rohan Rath, "A comparative study of enterprise and open source big data analytical," *in Proceedings of IEEE Conference on information & Communication Technologies (ICT),* Apr. 2013, pp. 372-377.

[6] Jinson Zhang and Mao Lin Huang, "5Ws model for big data analysis and visualization," *in Proceedings of IEEE Conference on Computational Science and Engineering (CSE),* Dec. 2013, pp. 1021-1028.

[7] IBM, “What is big data?” available: http://www-01.ibm.com/software/data/bigdata/

[8] E.Dumbill, “What is big data? An introduction to the big data landscape,” available: http://strata.oreilly.com/2012/01/what-is-big-data.html

[9] Yuri Demchenko, Cees de Laat, and Peter Membrey, “Defining architecture components of the big data ecosystem,” *in Proceedings of IEEE Conference on Collaboration Technologies and Systems (CTS),* May 2014, pp.104-122.

[10] Yuri Demchenko, Paola Grosso, Cees de Laat, and Peter Membrey, “Addressing big data issues in scientific data infrastructure,” *in Proceedings of IEEE Conference on Collaboration Technologies and Systems (CTS)*, May 2013, pp. 48-55.

[11] Apache Hadoop, available: https://hadoop.apache.org/

[12] Apache Mahout, available: https://mahout.apache.org/

[13] Apache Spark, available: http://spark.apache.org/

[14] Mengdi Wang, Bo Li, Yongxin Zhao, and Geguang Pu, “Formalizing Google file system,” *in Proceedings of 20th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Nov. 2014, pp. 18-21.

[15] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li, “Toward scalable systems for big data analytics: a technology tutorial,” *IEEE Access*, vol. 2, pp. 652-687. Jul. 2014.

[16] Avita Katal, Mohammad Wazid, and R. H. Goudar, ”Big data: issues, challenges, tools and good practices,” *in Proceedings of International Conference on Contemporary Computing (IC3)*, Aug. 2013, pp. 404-409.

[17] Stephen Kaisler, Frank Armour, J. Alberto Espinosa, and William Money, “Big data: issues and challenges moving forward,” *in Proceedings of Hawaii International Conference on System Sciences (HICSS)*, Jan. 2013, pp. 995-1004.

[18] Jinsong Zhang, Yan Chen, and Taoying Li, “Opportunities of innovation under challenges of big data,” *in Proceedings of IEEE International Conference on Computational Science and Engineering (CSE)*, Dec. 2013, pp. 1021-1028.

[19] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google file system,” *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29-34, Dec. 2003.

[20] Peter Mell and Timothy Grance, “A NIST definition of cloud computing,” available: http://csr c.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[21] Zhongduo Lin and Paul Chow, "ZCluster: a Zynq-based Hadoop cluster," *in Proceedings of International Conference on Field-Programmable Technology (FPT)*, Dec. 2013, pp. 450-453.

[22] Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, and Peter Vajgel, “Finding a needle in haystack: Facebook’s photo storage,” *in Proceedings of 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Oct. 2010, pp.1-8.

[23] Taobao File System, available: http://code.taobao.org/p/tfs/src/

[24] Rick Cattell, "Scalable SQL and NoSQL data stores," *in Proceedings of ACM SIGMOD Record*, vol. 39, no. 4, pp. 12-27. Dec. 2010.

[25] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber, “Bigtable: A Distributed storage system for structured data,” *ACM Transactions on Computer Systems (TOCS)*, vol. 26 no. 2, pp. 1-26, Jun. 2008.

[26] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, “Dynamo: Amazon’s highly available key-value store,” *in Proceedings of ACM SIGOPS Operating Systems Review (SOSP)*, vol. 41, no. 6, pp. 205-220, Dec. 2007.

[27] MongoDB, available: http://www.mongodb.org/

[28] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, JJ Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li,

Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford, "Spanner: Google's globally-distributed database," *in Proceedings of 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI),* Oct. 2012, pp. 251-264.

[29] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM,* vol. 51, no. 1, pp. 107-113, Jan. 2008.

[30] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," *in Proceedings of 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, Jun. 2007, pp. 59-72.

[31] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein, "Distributed GraphLab: a framework for machine learning and data mining in the cloud," *in Proceedings of VLDB Endowment,* vol. 5, no. 8, pp. 716-727, Apr. 2012.

[32] Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski, "Pregel: A system for large-scale graph processing," *in Proceedings of ACM SIGMOD International Conference on Management of Data* , Jun. 2010, pp. 135-146.

[33] 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

[34] Jian Ouyang, "SDA: software-defined accelerator for large-scale deep learning system," *in Proceedings of International Symposium on VLSI Design, Automation and Test (VLSI-DAT),* Apr. 2016.

[35] David Koeplinger, Raghu Prabhakar, Yaqi Zhang, Christina Delimitrou, Christos Kozyrakis, and Kunle Olukotun, “Automatic generation of efficient accelerators for reconfigurable hardware,” *in Proceedings of 43rd International Symposium on Computer Architecture (ISCA),* Jun. 2016.

[36] Yi Shan, Bo Wang, Jing Yan, Yu Wang, Ningyi Xu, and Huazhong Yang, “FPMR: MapReduce framework on FPGA: A case study of RankBoost acceleration,” *in Proceedings of 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays,* Feb. 2010, pp. 93-102.

[37] HuanXin Zheng and JunMin Wu, “Accelerate K-means algorithm by using GPU in the Hadoop framework,” *Web-Age Information Management (WAIM) 2014 International Workshops: BigEM, HardBD, DaNoS, HRSUNE, BIDASYS*, vol. 8597, pp. 177-186, Oct. 2014.

[38] Koichi Shirahata, Hitoshi Sato, and Satoshi Matsuoka, “Hybrid map task scheduling for GPU-based heterogeneous clusters,” *in Proceedings of 2nd International Conference on Cloud Computing Technology and Science (CloudCom),* Nov. 2010, pp. 733-740.

[39] Dai-Hua Lee, “High Scalability FPGA-based hardware accelerator for data-intensive computation,” *Master’s thesis, National Chung Cheng University, 2015*.

[40] Amar Shan, “Heterogeneous processing: a strategy for augmenting Moore's law,” *Linux Journal*, vol. 2006, no. 142, pp.7, Feb. 2006.

[41] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips, “GPU computing,” *in Proceedings of the IEEE, v*ol. 96, no. 5, pp. 879-899, May 2008.

[42] You Li, Kaiyong Zhao, Xiaowen Chu, and Jiming Liu, “Speeding up K-means algorithm by GPUs,” *in Proceedings of 10th International Conference on*

Computer and Information Technology (CIT), Jun. 2010, pp. 115-122.

[43] Wen Tang, Wendi Wang, Bo Duan, Chunming Zhang, Guangming Tan, Peiheng Zhang, and Ninghui Sun, "Accelerating millions of short reads mapping on a heterogeneous architecture with FPGA accelerator," *in Proceedings of 20th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Apr. 2012, pp. 184-187.

[44] Markus Weinhardt, Alexander Krieger, and Thomas Kinder, "A framework for PC applications with portable and scalable FPGA accelerators," *in Proceedings of International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Dec. 2013, pp. 1-6.

[45] Hadoop Streaming , available: http://hadoop.apache.org/docs/r2.7.1/hadoop-streaming/HadoopStreaming.html

[46] XindongWu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1-37, Dec. 2007.