International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Survey of Hardware Platforms Available for Big Data Analytics Using K-means Clustering Algorithm

Dr. M. Manimekalai[1], **S. Regha**[2]

[1]Director of MCA, Shrimati Indira Gandhi College, Trichy. India

[2]Research Scholar, Assistant Professor in Computer Science, Shrimati Indira Gandhi College, Trichy, India

Abstract: ***The Purpose of this paper is to provide an in-depth analysis of different platforms available for performing big data analytics. This paper review the different hardware platforms available for big data analytics and assesses the advantages and Negative aspects of each of these platforms based on various metrics such as scalability, data I/O rate, fault tolerance, Concurrent processing, data size supported and iterative task support. In addition to the hardware, a detailed description of the software frameworks used within each of these platforms is also discussed along with their strengths and Weakness. Some of the critical characteristics described here can potentially aid the readers in making an informed decision about the right choice of platforms depending on their computational needs. Using a star ratings table, a rigorous qualitative comparison between different platforms is also discussed for each of the six characteristics that are critical for the algorithms of big data analytics.***

Keywords: Big data, MapReduce, graphics processing units, scalability, big data analytics, big data Platform,k-means clustering.

1. Introduction

Big Data is driving radical changes in traditional data analysis platforms. To perform any kind of analysis on such voluminous and complex data, scaling up the hardware platforms becomes imminent and choosing the right hardware/software platforms becomes a crucial decision if the user's requirements are to be satisfied in a reasonable amount of time. Researchers have been working on building novel data analysis techniques for big data more than ever before which has led to the continuous development of many different algorithms and platforms. There are several big data platforms available with different characteristics and choosing the right platform requires an in-depth knowledge about the capabilities of all these platforms. we will first provide a thorough understanding of all the popular big data platforms that are currently being used in practice and highlight the advantages and drawbacks of each of them. Our work primarily aims at characterizing these concerns and focuses on comparing all the platforms based on these various optimal characteristics, thus providing some guidelines about the suitability of different platforms for various kinds of scenarios that arise while performing big data analytics in practice. we will provide a case study on the implementation of k-means clustering algorithm on various big data platforms. The k-means clustering was chosen here not only because of its popularity, but also due to the various dimensions of complexity involved with the algorithm such as being iterative, compute-intensive, and having the ability to parallelize some of the computations. We will provide a detailed pseudo code of the implementation of the k-means clustering algorithm on different hardware and software platforms and provide an in-depth analysis and insights into the algorithmic details. The remainder of the paper is organized as follows: the fundamental scaling concepts along with the advantages and drawbacks of horizontal and vertical scaling are explained in Section "Scaling". Section "Horizontal scaling platforms" describes various horizontal scaling platforms including peer-to-peer networks, Hadoop and Spark. In section "Vertical scaling platforms", various vertical platforms graphics processing units and high performance clusters are described. Section "Comparison of different platforms" provides thorough comparisons between different platforms based on several characteristics that are important in the context of big data analytics. Section "How to choose a platform for big data analytics?" discusses various details about choosing the right platform for a particular big data application. A case study on k-means clustering algorithm along with its implementation level details on each of the big data platform is described in Section "K-means clustering on different platforms". Finally, the "Conclusion" section concludes our discussion along with future directions.

1.1 Scaling

Scaling is the ability of the system to adapt to increased demands in terms of data processing. To support big data processing, different platforms incorporate scaling in different forms. From a broader perspective, the big data platforms can be categorized into the following two types of scaling:

a) **Horizontal Scaling:** Horizontal scaling involves distributing the workload across many servers which may be even commodity machines. It is also known as "scale out", where multiple independent machines are added together in order to improve the processing capability. Typically, multiple instances of the operating system are running on separate machines.
b) **Vertical Scaling:** Vertical Scaling involves installing more processors, more memory and faster hardware, typically, within a single server. It is also known as "scale up" and it usually involves a single instance of an operating system. Compares the advantages and drawbacks of horizontal and vertical scaling. While

scaling up vertically can make the management and installation straight-forward, it limits the scaling ability of a platform since it will require substantial financial investment. To handle future workloads, one always will have to add hardware which is more powerful than the current requirements due to limited space and the number of expansion slots available in a single machine. This forces the user to invest more than what is required for his current processing needs.

1.2 Horizontal scaling platforms

Some of the prominent horizontal scale out platforms includes peer-to-peer networks and Apache Hadoop. Recently, researchers have also been working on developing the next generation of horizontal scale out tools such as Spark to overcome the limitations of other platforms. We will now discuss each of these platforms in more detail in this section.

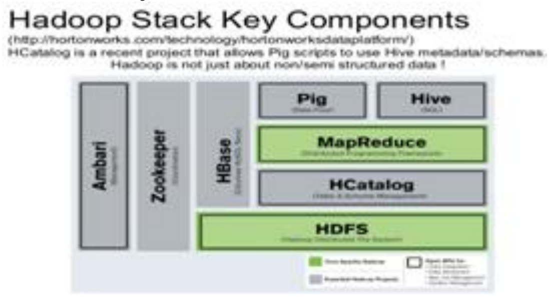
1.3 Peer-to-Peer Networks

Peer-to-Peer networks involve millions of machines connected in a network. It is a decentralized and distributed network architecture where the nodes in the networks (known as peers) serve as well as consume resources. It is one of the oldest distributed computing platforms in existence. Typically, Message Passing Interface (MPI) is the communication scheme used in such a setup to communicate and exchange the data between peers. Each node can store the data instances and the scale out is practically unlimited (can be millions of nodes).MPI, which is the standard software communication paradigm used in this network, has been in use for several years and is well-established and thoroughly debugged. One of the main features of MPI includes the state preserving process i.e., processes can live as long as the system runs and there is no need to read the same data again and again as in the case of other frameworks such as MapReduce (explained in section “Apache hadoop”). Although MPI appears to be perfect for developing algorithms for big data analytics, it has some major drawbacks. One of the primary drawbacks is the fault intolerance since MPI has no mechanism to handle faults. When used on top of peer-to-peer networks, which is a completely unreliable hardware, a single node failure can cause the entire system to shut down. Users have to implement some kind of fault tolerance mechanism within the program to avoid such unfortunate situations. With other frameworks such as Hadoop (that are robust to fault tolerance) becoming widely popular, MPI is not being widely used anymore.

1.4 Apache hadoop

Apache Hadoop is an open source framework for storing and processing large datasets using clusters of commodity hardware. Hadoop is designed to scale up to hundreds and even thousands of nodes and is also highly fault tolerant. The various components of a Hadoop Stack are shown in Figure. The Hadoop platform contains the following two important components:

Distributed File System (HDFS) is a distributed file system that is used to store data across cluster of commodity machines while providing high availability and fault tolerance. Hadoop YARN is a resource management layer and schedules the jobs across the cluster.


Hadoop Stack Key Components
(http://hortonworks.com/technology/hortonworksdataplatform/)
HCatalog is a recent project that allows Pig scripts to use Hive metadata/schemas.
Hadoop is not just about non/semi structured data !
Ambari
Zookeeper
HBase
Pig
Hive
MapReduce
HCatalog
HDFS

Hadoop Stack showing different components

2. MapReduce

The programming model used in Hadoop is MapReduce which was proposed by Dean and Ghemawat at Google. MapReduce is the basic data processing scheme used in Hadoop which includes breaking the entire task into two parts, known as mappers and reducers. At a high-level, mappers read the data from HDFS, process it and generate some intermediate results to the reducers. Reducers are used to aggregate the intermediate results to generate the final output which is again written to HDFS. A typical Hadoop job involves running several mappers and reducers across different nodes in the cluster.

2.1 MapReduce wrappers

A certain set of wrappers are currently being developed for MapReduce. These wrappers can provide a better control over the MapReduce code and aid in the source code development. The following wrappers are being widely used in combination with MapReduce.

Apache Pig is a SQL-like environment developed at Yahoo is being used by many organizations like Yahoo, Twitter, AOL, LinkedIn etc. **Hive** is another MapReduce wrapper developed by Facebook. In addition to these wrappers, some researchers have also developed scalable machine learning libraries such as Mahout using MapReduce paradigm.

2.2 Drawbacks of MapReduce

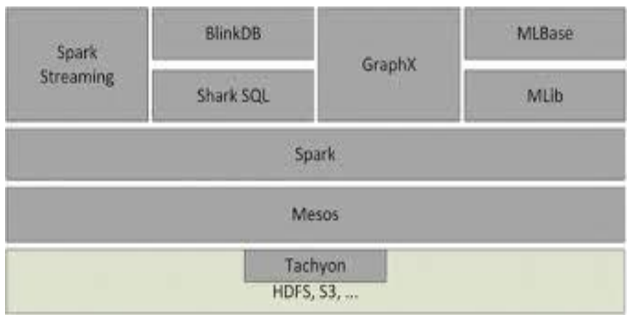
One of the major drawbacks of MapReduce is its inefficiency in running iterative algorithms. MapReduce is not designed for iterative processes. Mappers read the same data again and again from the disk.

Spark: next generation data analysis paradigm
Spark is a next generation paradigm for big data processing developed by researchers at the University of California at Berkeley. It is an alternative to Hadoop which is designed to overcome the disk I/O limitations and improve the performance of earlier systems. The major feature of Spark that makes it unique is its ability to perform in-memory computations. It allows the data to be cached in memory, thus eliminating the Hadoop’s disk overhead limitation for iterative tasks. Spark is a general engine for large-scale data

processing that supports Java, Scala and Python and for certain tasks it is tested to be up to 100× faster than Hadoop MapReduce when the data can fit in the memory, and up to 10× faster when data resides on the disk. It can run on Hadoop Yarn manager and can read data from HDFS. This makes it extremely versatile to run on different systems.

3. Berkeley data analytics stack (BDAS)

The Spark developers have also proposed an entire data processing stack called Berkeley Data Analytics Stack (BDAS). At the lowest level of this stack, there is a component called Tachyon which is based on HDFS. It is a fault tolerant distributed file system which enables file sharing at memory-speed (data I/O speed comparable to system memory) across a cluster. It works with cluster frameworks such as Spark and MapReduce.



An illustration of Berkeley Data Analysis Stack and **its various components**

The major advantage of Tachyon over Hadoop HDFS is its high performance which is achieved by using memory more aggressively. Tachyon can detect the frequently read files and cache them in memory thus minimizing the disk access by different jobs/queries

3.1 Vertical scaling platforms

The most popular vertical scale up paradigms are High Performance Computing Clusters (HPC), Multicore processors, Graphics Processing Unit (GPU) and Field Programmable Gate Arrays (FPGA). We describe each of these platforms and their capabilities in the following sections.

3.2 High performance computing (HPC) clusters

HPC clusters , also called as blades or supercomputers, are machines with thousands of cores. They can have a different variety of disk organization, cache, communication mechanism etc. depending upon the user requirement. These systems use well-built powerful hardware which is optimized for speed and throughput. Because of the top quality high-end hardware, fault tolerance in such systems is not problematic since hardware failures are extremely rare

3.3 Graphics processing unit (GPU)

Graphics Processing Unit (GPUs) is a specialized hardware designed to accelerate the creation of images in a frame buffer intended for display output .Until the past few years, GPUs were primarily used for graphical operations such as video and image editing, accelerating graphics-related processing etc. However, due to their massively parallel architecture, recent developments in GPU hardware and related programming frameworks have given rise to GPGPU (general-purpose computing on graphics processing units) . GPU has large number of processing cores (typically around 2500+ to date) as compared to a multicore CPU.

3.4 Field programmable gate arrays (FPGA)

FPGAs are highly specialized hardware units which are custom-built for specific applications .FPGAs can be highly optimized for speed and can be orders of magnitude faster compared to other platforms for certain applications. They are programmed using Hardware descriptive language (HDL). Due to customized hardware, the development cost is typically much higher compared to other platforms.

4. Comparison of Different Platforms

Comparison of different platforms using the based on the following characteristics: scalability, data I/O performance, fault tolerance, Concurrent processing, data size support and the support for iterative tasks. Clearly, the first three characteristics are system/platform dependent and last three are application/algorithm dependent. **Scalability:** Scalability is defined as the ability of the system to handle growing amount of work load in a capable manner or its ability to be enlarged to accommodate that growth. In our case, scalability is considered to be the ability to add more hardware (scale up or scale out) to improve the capacity and performance of a system.

Data I/O performance: Data I/O performance refers to the rate at which the data is transferred to/from a peripheral device. In the context of big data analytics, this can be viewed as the rate at which the data is read and written to the memory (or disk) or the data transfer rate between the nodes in a cluster. GPU and FPGA receive 5 stars since they have high throughput memory and the data I/O operations are extremely fast. The current generation GPUs are available with DDR5 memory which is many times faster than the DDR3 system memory. HPC clusters and Multicore will fall next in this category with 4 stars. These systems usually make use of system memory which is reasonably faster compared to disk access. Since HPC clusters and Multicore are usually single machines, network access is not a bottleneck.

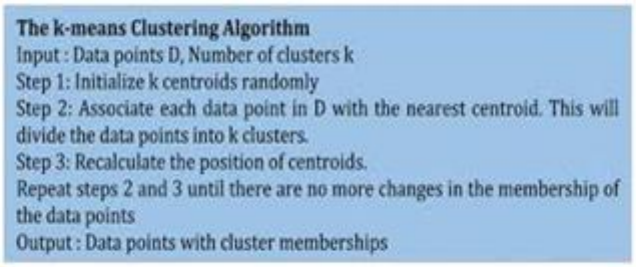
Fault tolerance: Fault tolerance is the characteristic of a system to continue operating properly in the event of a failure of one or more components. Since we created this table with an intent to compare the platforms of similar capacity, we additionally consider the chances of failure in a system and give a high rating if system failures are extremely rare even though it may not have any fault tolerance mechanism. This enables us to make an unbiased comparison between unreliable systems with fault tolerance and reliable hardware with not so good fault tolerance mechanism.

Concurrent processing: Real-time processing of a system is its ability to process the data and produce the results strictly within certain time constrains. Real-time responses are often delivered in the order of milliseconds and sometimes microseconds depending on the application and the user requirements.

Data size supported: Data size support is the size of the dataset that a system can process and handle efficiently. In this category, peer-to-peer networks will receive 5 stars since they can handle even petabytes of data and can theoretically scale out to unlimited number of nodes. Multicore, GPU and FPGA are not well suited for processing large data sets. All these systems get 2 stars for the limited size of the data that they can support. GPUs have a limited on-board memory in the order of several gigabytes. Similarly, Multicore systems rely on system memory which can only be up to hundreds of gigabytes.

Iterative tasks support: This is the ability of a system to efficiently support iterative tasks. Since many of the data analysis tasks and algorithms are iterative in nature, it is an important metric to compare different platforms, especially in the context of big data analytics.

K-means clustering on different platforms: In order to provide more insights into the analytics algorithms on different platforms, we will demonstrate the implementation of the K-Means clustering algorithm on these platforms presented so far. The choice of the K-Means algorithm was made not only because of its popularity and wide usage ,but also due to some of its critical elements that can demonstrate the ability of various platforms in handling other analytics procedures. Some of these characteristics include: Iterative nature of the algorithm wherein the current iteration results are needed before proceeding to the next iteration. Compute-intensive task of calculating the centroids from a set of datapoints.Aggregation of the local results to obtain a global solution when the algorithm is parallelized.It should be noted that many of the analytics algorithms share atleast some of these characteristics. Hence, it is important to understand how these characteristics of K-means clustering algorithm are being handled using different platforms. Figure explains the different steps involved in a basic K-means clustering algorithm. The algorithm starts by initializing the cluster centroids. In the next step, each data point is associated with closest centroid and in the third step, the centroids are recalculated for all the associated data instances for a given cluster. The second and third steps are repeated until the centroids converge (or after a pre-defined number of iterations).



```
The k-means Clustering Algorithm
Input : Data points D, Number of clusters k
Step 1: Initialize k centroids randomly
Step 2: Associate each data point in D with the nearest centroid. This will
divide the data points into k clusters.
Step 3: Recalculate the position of centroids.
Repeat steps 2 and 3 until there are no more changes in the membership of
the data points
Output : Data points with cluster memberships
```

The pseudocode of the K-means clustering algorithm: We will now discuss the implementation details of this algorithm on different platforms to get a deeper understanding of how such iterative algorithms are modified to fit different communication schemes.

K-means on MapReduce: MapReduce is not an ideal choice for iterative algorithms such as K-Means clustering. This will be clearly shown in this section as we explain the K-Means clustering using MapReduce. The pseudo code for mapper and reducer functions for k-means clustering algorithm is given in Figure. Basically, mappers read the data and the centroids from the disk. These mappers then assign data instances to clusters. Once every mapper has completed their operation, reducers compute the new centroids by calculating the average of data points present in each cluster. Now, these new centroids are written to the disk. These centroids are then read by the mappers for the next iteration and the entire process is repeated until the algorithm converges. This shows the disk access bottleneck of MapReduce for iterative tasks as the data has to be written to the disk after every iteration. The first part shows the map function and the second part shows the reduce function.

K-means on MPI:MPI typically have a master–slave setting and the data is usually distributed among the slaves.. In the second step, the master broadcasts the centroids to the slaves. Next, the slaves assign data instances to the clusters and compute new local centroids which are then sent back to the master. Master will then compute new global centroids by aggregating local centroids weighted by local cluster sizes. These new global centroids are then again broadcasted back to the slaves for the next iteration of K-means. In this manner, the process continues until the centroids converge. In this implementation, the data is not written to the disk but the primary bottleneck lies in the communication when MPI is used with peer-to-peer networks since aggregation is costly and the network performance will be low.

K-means on GPU: GPU has a large number of processing cores. Hence, in order to effectively utilize all the cores, the algorithm will need to be modified cautiously. Figure 7 shows the pseudo code of K-means using GPU. In the case of K-means, each processor is given a small task (assigning a data vector to a centroid). Also, a single core in a GPU is not very powerful which is why the centroid recalculation is done on the CPU. The centroids are uploaded to the shared memory of the GPU and the data points are partitioned and uploaded into each multiprocessor. These multiprocessors work on one data vector at a time and associate it with the closest centroid. Once all the points are assigned to the centroids, CPU recalculates the centroids and again will upload the new centroids to the multiprocessors. This process is repeated until the centroids converge or until a pre-defined number of iterations are completed. Another aspect to consider here is the density of the data. If the data is sparse, many multiprocessors will stall due to scarcity of data vectors to compute, which will eventually degrade the performance. In a nutshell, the performance of GPUs will be the best when the data is relatively denser and when the algorithm is carefully modified to take advantage of processing cores.

K-means on other platforms: K-means implementation on Spark is similar to the MapReduce-based implementation described in Section K-means on MapReduce. Instead of writing the global centroids to the disk, they are written into the memory which speeds up the processing and reduces the disk I/O overhead. In addition, the data will be loaded into the system memory in order to provide faster access. The K-means clustering on CPU involves multithreading where each thread associates a data vector to a centroid and finally the centroids are recomputed for the next iteration. On the other hand, K-means implementation on FPGA depends upon the FPGA architecture used and may differ significantly depending on the type of FPGA being used.

5. Conclusion and Future Directions

This paper surveys various data processing platforms that are currently available and discusses the advantages and drawbacks for each of them. Several details on each of these hardware platforms along with some of the popular software frameworks such as Hadoop and Spark are also provided. A thorough comparison between different platforms based on some of the important characteristics (such as scalability and real-time processing) has also been made through star based ratings. The widely used k-means clustering algorithm was chosen as a case study to demonstrate the strengths and weaknesses of different platforms. Some of the important characteristics of k-means algorithm such as its iterative nature, compute-intensive calculations and aggregating local results in a parallel setting makes it an ideal choice to better understand the various big data platforms. It should be noted that many of the analytical algorithms share these characteristics as well. This article provides the readers with a comprehensive review of different platforms which can potentially aid them in making the right decisions in choosing the platforms based on their data/computational requirements.

The future work involves investigating more algorithms such as decision trees, nearest neighbor, page rank etc. over different platforms. For empirical evaluation, different experiments involving varying data size and response times can be performed over various platforms for different algorithms. Through such an analysis we will get valuable insights which can be useful in many practical and research applications. One other important direction of research will be to choose the right platform for a particular application. Based on the specific application needs, one can tailor their platform specific factors such as the amount of hard disk, memory and the speed required for optimally running the application. This study will provide a first step to analyze the effectiveness of each of the platforms and especially the strengths of them for handling real-world applications. Another direction will be to investigate the possibility of combining multiple platforms to solve a particular application problem. For example, attempting to merge the horizontal scaling platforms such as Hadoop with vertical scaling platforms such as GPUs is also gaining some recent attention. This involves several non-trivial tasks not only at the platform level but also becomes challenging to decompose the algorithm into parts and running various algorithmic components in various platforms. A combination of platforms might be more suitable for a particular algorithm and can potentially resolve the issue of making it highly scalable (through horizontal scaling) as well as performing real-time analysis (through vertical scaling).

References

[1] Agneeswaran VS, Tonpay P, Tiwary J: **Paradigms for realizing machine learning algorithms.** *Big Data* 2013, **1**(4)**:**207-214. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Spark SI: **Cluster Computing with Working Sets.** *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing* 2010, 10-10.
[2] Milojicic DS, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, Rollins S, Xu Z: *Peer-to-peer computing. Technical Report HPL-2002-57, HP Labs*. 2002.
[3] Steinmetz R, Wehrle K: *Peer-to-Peer Systems and Applications*. Springer Berlin, Heidelberg; 2005.
[4] Sievert O, Casanova H: **A simple MPI process swapping architecture for iterative applications** *Int J High Perform Comput Appl* 2004, **18**(3):341-352. Borthakur D: *HDFS architecture guide. HADOOP APACHE PROJECT*. 2008.
[5] Vavilapalli VK, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H, Seth S: **Apache hadoop yarn: Yet another resource negotiator.** *Proceedings of the 4th annual Symposium on Cloud Computing* 2013, 5.
[6] Dean J, Ghemawat S: **MapReduce: simplified data processing on large clusters.** *Commun ACM* 2008, **51**(1)**:**107-113. Lee K-H, Lee Y-J, Choi H, Chung YD, Moon B: **Parallel data processing with MapReduce: a survey.** *ACM SIGMOD Record* 2012, **40**(4)**:**11-20. *Proceedings of the VLDB Endowment* 2010, **3**(1–2)**:**285-296. Ekanayake J, Pallickara S, Fox G: **Mapreduce for data intensive scientific analyses.** *Proceesings of IEEE Fourth International Conference on eScience* 2008, 277-284.
[7] Palit I, Reddy CK: **Scalable and parallel boosting with MapReduce.** *IEEE Trans Knowl Data Eng* 2012, **24**(10)**:**1904-1916. Ekanayake J, Li H, Zhang B, Gunarathne T, Bae S-H, Qiu J, Fox G (2010) Twister: a runtime for iterative mapreduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM, pp 810–818
[8] Zhang Y, Gao Q, Gao L, Wang C: **Imapreduce: a distributed computing framework for iterative computation.** *J Grid Comput* 2012, **10**(1)**:**47-68
[9] Agarwal S, Mozafari B, Panda A, Milner H, Madden S, Stoica I: **BlinkDB: Queries with Bounded Errors and Bounded Response times on very Large Data.** *Proceedings of the 8th ACM European Conference on Computer Systems* 2013, 29-42. Xin RS, Gonzalez JE, Franklin MJ, Stoica I: **Graphx: A resilient distributed graph system on spark.** *First International Workshop on Graph Data Management Experiences and Systems* 2013, 2.
[10] Kraska T, Talwalkar A, Duchi JC, Griffith R, Franklin MJ, Jordan MI: **MLbase: A Distributed Machine-learning System.** *Proceedings of Sixth Biennial Conference on Innovative Data Systems Research* 2013.

[11] Buyya R: *High Performance Cluster Computing: Architectures and Systems (Volume 1)*. Prentice Hall, Upper SaddleRiver, NJ, USA; 1999.

[12] Bekkerman R, Bilenko M, Langford J (2012) Scaling up Machine Learning: Parallel and Distributed Approaches. Cambridge University Press

[13] Tullsen DM, Eggers SJ, Levy HM: **Simultaneous Multithreading: Maximizing on-Chip Parallelism.** *ACM SIGARCH Computer Architecture News* 1995, 392-403.

[14] Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC: **GPU computing.** *Proc IEEE* 2008, **96**(5)**:**879-899. Nickolls J, Dally WJ: **The GPU computing era.** *IEEE Micro* 2010, **30**(2)**:**56-69Hong S, Kim H: **An analytical Model for a GPU Architecture with Memory-Level and Thread-Level Parallelism Awareness.** *ACM SIGARCH Computer Architecture News* 2009, 152-163.

[15] Fang W, Lau KK, Lu M, Xiao X, Lam CK, Yang PY, He B, Luo Q, Sander PV, Yang K: *Parallel data mining on graphics processors. Hong Kong University of Science and Technology, Tech Rep HKUST-CS08-07 2*. 2008.

[16] Francis RJ, Rose J, Vranesic ZG (1992) Field Programmable Gate Arrays, vol 180. Springer

[17] Thomas DE, Moorby PR (2002) The Verilog Hardware Description Language, vol 2. Springer

[18] Monmasson E, Idkhajine L, Cirstea MN, Bahri I, Tisan A, Naouar MW: **FPGAs in industrial control applications.** *IEEE Trans Ind Informat* 2011, **7**(2)**:**224-243.

[19] Bouldin D: **Impacting education using FPGAs.** *Proceedings of 18th International conference on Parallel and Distributed Processing Symposium* 2004, 142.

[20] Chen H, Chen Y, Summerville DH: **A survey on the application of FPGAs for network infrastructure security.** *IEEE Commun Surv Tutor* 2011, **13**(4)**:**541-561