GPU based Suffix Array Pattern Matching Approach for Big Data

Vinay Katoch
M.Tech,
Dept. of CSE
UIT, RGPV
Bhopal, India

Sanjay Silakari, PhD
Professor
Dept. of CSE
UIT RGPV
Bhopal, India

Uday Chourasia
Assistant Professor
Dept. of CSE
UIT, RGPV
Bhopal, India

ABSTRACT
Big data has been an emerging problem these days. To solve this problem Hadoop has evolved as a most widely used tool and adopted by various popular MNCs like Facebook and Yahoo. To search large number of pattern in big data is a challenging task. Map/Reduce is used to write codes to perform pattern matching on big data. In this work OpenCL is combined with Apache Hadoop to write fast Map/Reduce for pattern matching in data using suffix arrays.

Keywords
OpenCL, GPU, Hadoop, Map/Reduce.

1. INTRODUCTION
Big data is defined as large quantity of data which have need of new technologies and architecture to make possible to extort value from it by capturing and analysis process. New sources of big data include location specific data which has arrived from traffic management and from the tracking of personal devices such as Smartphone's. Big data has come into view because we are living in the world which makes mounting use of data intensive technologies. Due to such large size of data it becomes very difficult to achieve effective analysis using existing traditional techniques.

Since Big data is new upcoming technology in the market which can bring the huge benefits to the business organizations, it becomes necessary various challenges and issues associated in bringing and adopting to this technology are need to be understand. Big data concept means a dataset which continues grew so much that it becomes difficult to manage it using existing database models and tools. So at last

Big data is data that exceeds the processing capacity of conventional database systems. The data is huge sized, moves too fast, or doesn't fit the structures of our database architectures. To gain value from this data, you must choose a substitute way to process it.

What are the problems?
There are many problems to handle big data like storage, processing etc.

Data integration – The structure of merging data is not so easy task with a reasonable cost.

Data volume – The ability to process the volume at a suitable rate so that the information is available to result analyzers when they need it.

Skills availability –There are shortage of people. Who have the proficiency to bring all data mutually, analyze it and publish the results.

Solution cost –To ensure a positive ROI on a Big Data project; it is crucial to reduce the cost of the solutions.

What are the solutions?
Big data is very difficult to process and store. Mainly Hadoop is used to process the big data. Hadoop used HDFS to store the data efficiently and Map/Reduce framework for processing the data. MPI is also used to process the big data.

2. TYPES OF BIG DATA
Mainly Big data is divided in 3 types.

Structured data: It concerns all data which stored in the database in tabular form. Structured data represent only 5 to 10% of all informatics data.

Ex. Relational data.

Semi Structured data: Semi-structured data is information that does not inhabit in a relational database but that does have some organizational properties that make it easier to analyse.

Ex. CSV but XML and JSON documents are semi structured documents, NoSQL databases are considered as semi structured.

Unstructured data: Unstructured data is everywhere. In fact, most individuals and organizations achieve their lives around free data. Unstructured data represent around 80% of data.

Ex: videos, word processing documents, photos, presentations, webpages and many other kinds of business documents, audio files, E-mail messages, Word, PDF, Text, Media Logs.

3. CHARACTERISTICS
Big data is characterized in 3 terms; figure 2 show the characteristics of the big data.

Volume: Our personal system might have 500 GB of storage. But every day Facebook consume 500TB of new data. Extreme use of smartphone with new tools like sensor which create additional data like position and former information as well as videos.

Velocity: The data is created vastly rapid. Like on-line game is played by millions of users simultaneously, stock trading algorithm create huge amount of data every second, sensors are producing the data in real time, ad impression detain user actions at millions of actions per seconds. So the data are created at a swift pace and we need efficient tools in order to deal.

Variety: All the data are of distinct type. Some may be video, audio, text which may be distinct. It may not be only numbers, dates and strings.

4. HADOOP

Hadoop is an Apache open source framework which supports java and also java code for implementation that allows distributed processing of large records across clusters of computers using easy programming models. The Hadoop framework application moving in an atmosphere that offer distributed storage and estimation across clusters of computers. Hadoop is made for level up from single server to thousands of equipment, each offering local calculation and storage.

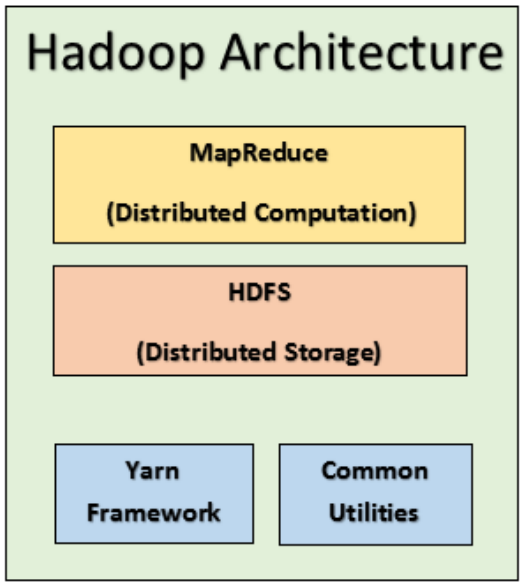


Figure 1: Architecture of Hadoop [17]

Hadoop mainly consist of 2 components.

4.1 Map-Reduce

Map-Reduce applications can precede multiple terabytes of data in parallel on large clusters in a fault-tolerant manner and reliable. Map-Reduce is an estimation paradigm in which a function is divided into self-reliant units of work. Each of these units of process can be issued on any node in the cluster.

A Map-Reduce job divides the input data into several chunks that are processed by map jobs in similar way. The framework arranges the map outputs, which are then input to reduce tasks. Task's inputs and outputs are lain up in the file system. The Map-Reduce framework and the HDFS (Hadoop Distributed File System) are classically on the same set of nodes, which allow the structure to schedule tasks on nodes that hold data.

The Map-Reduce framework having a single key JobTracker and one secondary Task Tracker per node. The main node plan for job section tasks, re-executes, and monitors tasks abortive tasks. The secondary node processes as directed by the primary node.

Map-Reduce have two phases
i)Map
ii)Reduce

i. The map phase

The map phase is the first part of the data processing sequence within the Map-Reduce framework. Map functions serve as worker nodes that can process several smaller snippets of the entire data set. The Map-Reduce framework is responsible for dividing the data set input into smaller chunks, and feeding them to a corresponding map function. When you write a map function, there is no need to incorporate logic to enable the function to create multiple maps that can use the distributed computing architecture of Hadoop. These functions are oblivious to both data volume and the cluster in which they are operating. As such, they can be used unchanged for both small and large data sets (which is most common for those who are using Hadoop).

ii. The reduce phase

As with the map function, developers also must create a reduce function. The key/value pairs from map outputs must correspond to the appropriate reducer partition such that the final results are aggregates of the appropriately corresponding data. When the shuffle process is completed and the reducer copies all of the map task outputs, the reducers can go into what is known as a merge process. During this part of the reduce phase, all map outputs can be merged together to maintain their sort ordering that is established during the map phase. When the final merge is complete then this reduce task of consolidating results for every key within the merged output (and the final result set), is written to the disk on the HDFS.

4.2 Hadoop Distributed File System (HDFS)

The Hadoop is specialized design file system for storing huge data set with cluster of commodity hardware with streaming access pattern. It is highly fault tolerant and designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for application having large datasets.

5. OPEN COMPUTING LANGUAGE (OPENCL)

OpenCL [7] is a low-level programming model which works on heterogeneous environment. It is supported by Khronos Compute Working Group to use it for the implementation for the GPUs from the various companies like Intel, AMD, NVIDIA etc. It is built to use all computational resources in a system like; CPUs, GPUs, APUs, FPGAs etc. OpenCL code programs also called kernels which run on GPUs and cores of CPU.

Fig. 2 explains single Compute device consists of multiple compute units which are consists of multiple processing elements. A host consists of multiple compute devices. An OpenCL program runs all the processing elements in parallel using a single kernel code. So, it provides Inter-Node level parallelism on processing elements in heterogeneous environment.

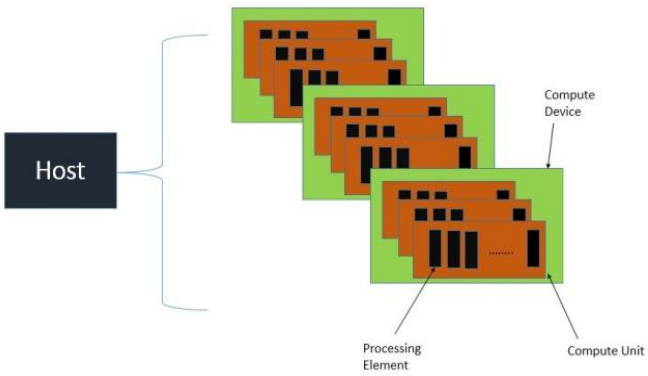


Figure 2: OpenCL Architecture Platform Model

5.1 APARAPI

Aparapi [8], a parallel API, an open source tool, developed at AMD, is used to implement Hadoop with OpenCL. This tool is used to help in running the Java code on OpenCL devices. It converts java byte code to OpenCL kernels at run time. It has a "Kernel" class which has a run() method which runs parallel on OpenCL device. Since it is built on the top of OpenCL, it is used to run code on GPUs and cores of CPUs. It is used to generate OpenCL kernels from java byte code automatically. It handles Kernel Translation, OpenCL Memory Allocation, Data Transfer etc.

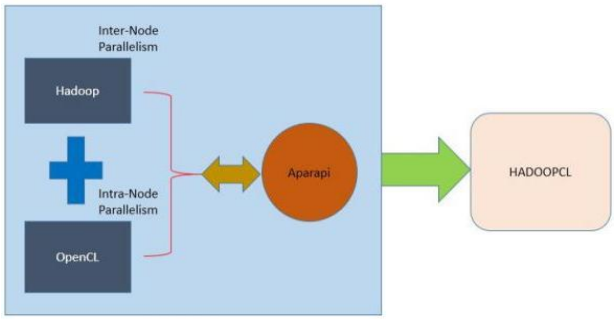


Figure 3: Integration of Hadoop and OpenCL using Aparapi

Fig. 3 explains that HADOOPCL is an integration of Hadoop with OpenCL using Aparapi tool. Hadoop provides Inter-Node parallelism and OpenCL provides Intra-Node parallelism.

5.2 HADOOPCL

HadoopCL [5], an integration of Hadoop and OpenCL. The purpose of HadoopCL is to enable the use of heterogeneous process in distributed system. It is used to execute user-written java kernels on heterogeneous devices. It executes user-written map and reduce computation on all available devices in a platform. The processing of data is done at thread-level (Intra-node) at each cluster (Inter- node) of HadoopCL using OpenCL kernels code. So, these two level parallelisms produce the result much faster and the recommendations provided to the users are much faster, efficiently and more accurately. The utilization of resources is also improved which hides the hardware complexities from the users and allows tuning experts to manipulate platform configuration in order to optimize performance, energy efficiency and reliability.

6. PREVIOUS WORK

Advances in next generation sequencing technologies in addition to decreasing wet science lab prices has resulted in unexampled acquisition of immense genomic knowledge sets. To translate the promise of that knowledge into new biological discoveries, innovative machine approaches area unit needed for timely and economical process and analysis. Orientating sequences to see similarity is a necessary and wide used machine procedure for biological sequence analysis in machine biology and bioinformatics (Quail MA n.d.)A good vary of machine algorithms are applied to the sequence alignment challenge, as well as slow, nevertheless correct, strategies like dynamic programming and quicker however less correct heuristic or probabilistic strategies. the fundamental native Alignment Search Tool (BLAST) ,a heuristic version of the pairwise native alignment Smith boatman rule, remains the foremost wide used machine procedure for alignment interrogating biological databases supported a heuristic version of the pairwise native alignment Smith boatman rule. It compares the similarity of a reference super molecule or deoxyribonucleic acid sequence against data of sequences, higher than a nominative threshold, and returns similar, statistically important, matches. In spite of its heuristic approach, it still faces important measurability challenges associated primarily with the need to go looking new and ever increasing knowledge base; like UniMES for met genomic data sets that still expand exponentially as Next Generation Sequencing (NGS) prices still decline. BLAST, together with most different bioinformatics algorithms, is meant to execute domestically i.e. consecutive. However, the augmented turnout of ordering sequencing has light-emitting diode to large knowledge generation requiring a big increase within the speed of execution of those algorithms. the appearance of cloud computing and massive knowledge ''scale out'' technologies like Hadoop give value effective process of T sized knowledge sets therefore it's currently potential to analyse these immense datasets apace; a very important demand within the rapidly increasing field of molecular medicine. Thus, because the size of genomic knowledge sets increase earlier than native process power and disk scan speed, it's intuitive to port these naturally parallel bioinformatics tasks to use the Hadoop Map Reduce framework. Standard approaches to parallelizing BLAST mistreatment Hadoop area unit 3 fold: the primary and commonest approach distributes the input question sequences amongst a cluster of nodes, the second approach partitions the data amongst nodes and at last a hybrid approach partitions each the input sequences and therefore the data. The downside of the primary approach is that it exhibits restricted measurability and cargo equalisation doesn't occur with a little range of input sequences. The second approach needs a complicated rule to partition the data so as to make sure measurability and optimum performance. moreover, it ends up in high disk I/O. the ultimate hybrid approach is desirable because it handles giant databases yet as an oversize range of input question sequences, but it's the foremost difficult to implement and deploy whereas minimizing inter-node communications and optimizing the partitioning strategies.

DumitrelLoghinet. al. 2015 presents a time–energy performance analysis of Map-Reduce on heterogeneous systems with GPUs. To execute Map-Reduce on heterogeneous systems with GPUs, we introduce a novel lazy processing technique which simplifies application development and requires no modifications to the underlying Hadoop framework. Based on this experiment, the wimpy (performance improvements in low-power) nodes achieve similar execution times compared to a single brawny node and also exhibit energy savings of up to two-thirds.

RazvanNituet. al. 2014 proposes An Improved GPU Map-Reduce Framework for Data Intensive Applications. This framework improves the Map-Reduce performance by adding GPU capabilities by implementing a hybrid CPU-GPU framework for heterogeneous environments. All the functionalities regarding GPU programming are already implemented. The users just have to define the functions specific to the Map-Reduce paradigm, without having advanced knowledge about GPU programming. The GPU tasks are implemented using the OpenCL library. Since Hadoop is written in Java, we used the JOCL (OpenCL Java language binding) solution to integrate these two languages.

Can Basaran et.al 2013 present a new Map-Reduce framework, called Grex (a new GPU-based Map-Reduce framework), designed to leverage general purpose graphics

processing units (GPUs) for parallel data processing. The experimental results show that our system is up to 12.4× and 4.1× faster than two state-of-the-art GPU-based Map-Reduce frameworks for the tested applications.

Miao Xinet. al. 2012 presents an approach of Map-Reduce improvement with GPU acceleration, which is implemented by Hadoop and OpenCL. As a heterogeneous multi-machine and multicore architecture, it aims at both data- and compute-intensive applications. Java language is the best practice in Hadoop programing, for achieving a better seamless-integration; we select an OpenCL Java language binding (JOCL) to integrate these two frameworks together. JOCL use Java Native Interface (JNI) to call the kernel program that drives the GPUs. An almost 2 times performance improvement has been validated, without any farther optimization.

Wenbin Fang et. al. 2011 proposes Accelerating Map-Reduce with Graphics Processors: MARS. Mars is a Map-Reduce runtime system accelerated with graphics processing units (GPUs).It runs on NVIDIA GPUs, AMD GPUs as well as multicore CPUs.It is implemented Mars CUDA using NVIDIA CUDA. The experimental results show that, the GPU-CPU co-processing of Mars on an NVIDIA GTX280 GPU and an Intel quad-core CPU outperformed Phoenix, the state-of-the-art Map-Reduce on the multicore CPU with a speedup of up to 72 times and 24 times on average, depending on the applications. Additionally, integrating Mars into Hadoop enabled GPU acceleration for a network of PCs.

7. PROPOSED APPROACH

In suffix array and suffix tree based approach, there given a pattern and text in which pattern is to be searched.

Sequential_Suffix_tree
{
1. Fetch the pattern entered by the user.
2. Form all the suffixes of the pattern entered by the user.
3. Arrange all the suffixes formed in alphabetical order.
4. Form the tree with root element be the smallest in alphabetical ordered list.
}

Searching will be done by parsing each word of the text in the tree and if leaf node is reached pattern will be matched.

Parallel_Suffix_tree
{
1. Fetch all the patterns entered by the user.
2. Form all the suffixes of the patterns entered by the user.
3. Arrange all the suffixes formed in alphabetical order.
4. Form the tree with root element be the smallest in alphabetical ordered list.
5. Split the text into words and initialize equal number of threads using OpenCL.
6. Search for all the words by corresponding threads in the tree formed.
}

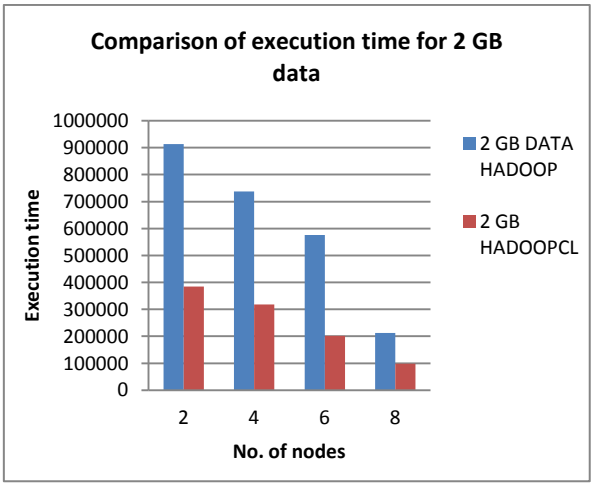
8. RESULTS

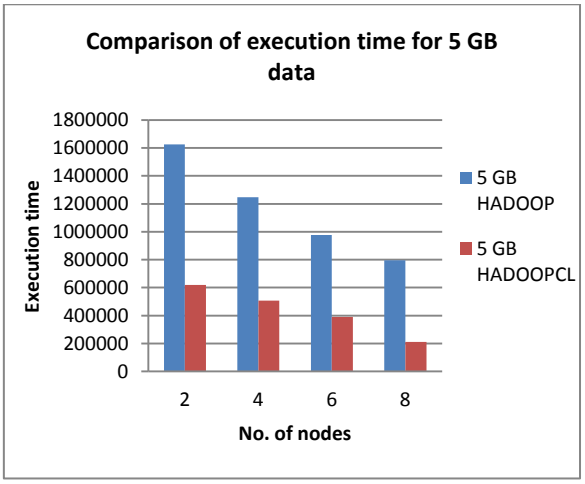
Table I. Execution Time for Hadoop

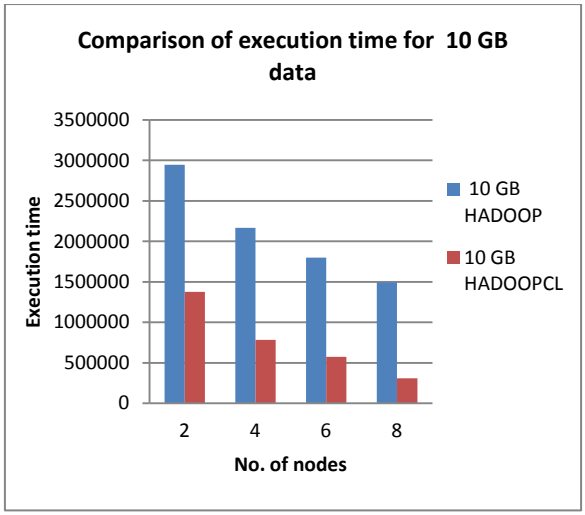
EXECUTION TIME ON HADOOP IN MILLISECONDS				
Data (In GB)	**Time on 2 nodes**	**Time on 4 nodes**	**Time on 6 nodes**	**Time on 8 nodes**
2	912751	736946	576148	212751
5	1625487	1247952	976425	794562
10	2947541	2167190	1801307	1497038

Table II Execution Time for HADOOPCL

EXECUTION TIME ON HADOOPCL IN MILLISECONDS				
Data (In GB)	**Time on 2 nodes**	**Time on 4 nodes**	**Time on 6 nodes**	**Time on 8 nodes**
2	384657	317945	201907	98706
5	619037	507640	390450	210721
10	1376420	783170	576103	310640







9. CONCLUSION

In this paper it is found as a conclusion of survey that map/reduce can be accelerated by using the concept of GPUs. Map-Reduce offers inter node parallelism and when integrated with GPUs using OpenCL it can offer intra node parallelism also. In this paper a suffix array algorithm is used to search pattern in text. This algorithm is analyzed to find parallelism and implemented using combination of OpenCL and Apache Hadoop for fast pattern matching.

10. REFERENCES

[1] Cheikh Kacfah Emani, Nadine Cullot and Christophe Nicolle "Understandable Big Data: A survey" in Computer Science Review Volume 17, August 2015, Pages 70–81.

[2] H. Hu, Y. Wen, T.-S. Chua, and X. Li, ''Towards scalable systems for big data analytics: A technology tutorial,'' IEEE Access, vol. 2, pp. 652–687, 2014.

[3] Felfernig, A., Jeran, M., Ninaus, G., Reinfrank, F., Reiterer and, S., Stettinger, M.: Basic approaches in recommendation systems. In: Robillard, M., Maalej, W., Walker, R.J., Zimmermann, T. (eds.) Recommendation Systems in Software Engineering, Chap. 2. Springer, Heidelberg (2014).

[4] S. Meng, W. Dou, X. Zhang and J. Chen, "KASR: A keyword-aware service recommendation method on Map-Reduce for big data application", IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 12, pp. 3221-3231, 2014.

[5] M. Grossman, M. Breternitz and V. Sarkar, "HadoopCL: Map-Reduce on Distributed Heterogeneous Platforms Through Seamless Integration of Hadoop and OpenCL", Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, pp. 1918-1927.

[6] A. Rabkin and R. H. Katz, "How Hadoop Clusters Break," IEEE Software, vol. 30, pp. 88-94, 2013.

[7] P. Jaaskelainen, C. Lama, P. Huerta, and J. Takala, "OpenCL-based design methodology for application-specific processors," Embedded Computer Systems (SAMOS), 2010 International Conference, pp. 223- 230, 2010.

[8] Gupta, K.G., Agrawal, N. and Maity, S.K., "Performance analysis between aparapi (a parallel API) and JAVA by implementing sobel edge detection Algorithm," in PARCOMPTECH, Bangalore, Feb. 2013, pp. 1-5.

[9] Niwattanakul S, Singthongchai J, Naenudorn E, Wanapu S. Using of Jaccard coefficient for keywords similarity. In: Proc. of the international multi conference of engineers and computer scientists, vol I; 2013. p. 380–4.

[10] A. Huang, Similarity measures for text document clustering, in: Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, 2008, pp. 49–56.

[11] P. WillettThe Porter stemming algorithm: then and now Program: Electr Libr Inform Syst, 40 (3) (2006), pp. 219–223.

[12] Wang Jun, Li Lei and Ren Fuji, "An Improved method of Keywords Extraction Based on Short Technology Text", International Conference on Natural Language processing and Knoledge Engineering (NLP-KE), pp. 1-6.

[13] Adomavicius G., Kwon Y.: New recommendation techniques for multicriteria rating systems. IEEE Intel. Syst. 22(3), 48–55 (2007).

[14] X. Zhang, J.-J. Lu, X. Qin and X.-N. Zhao, "A high-level energy consumption model for heterogeneous data centers", Simul. Model. Pract. Theory, vol. 39, pp. 41-55, 2013 .

[15] X. Peng and Z. Sai, ``A low-cost power measuring technique for virtual machine in cloud environments," Int. J. Grid Distrib. Comput., vol. 6, no. 3, p. 69, 2013.

[16] Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recommender Systems Handbook, pp. 217–253 (2011).

[17] Hadoop Architecture. Image Online: http://ercoppa.github.io/HadoopInternals/HadoopArchitectureOverview.html