International Journal of Hybrid Information Technology
Vol.8, No.6 (2015), pp.375-380
http://dx.doi.org/10.14257/ijhit.2015.8.6.36

GPU-accelerated Large Scale Analytics using MapReduce Model

RadhaKishan Yadav[1], Robin Singh Bhadoria[2] and Amit Suri[3]

[1]*Research Assistant*
[2]*Research Scholar*
[1, 2]*Indian Institute of Technology, Indore*
[3]*Senior Business Intelligence Consultant*
[3]*Microsoft Inc., Redmond, Washington, USA*
[1]*cse1200125@iiti.ac.in, robin19@ieee.org,* [3]*amitsuri21@gmail.com*

Abstract

Analysis and clustering of very large scale data set has been a complex problem. It becomes increasingly difficult to compute the results in a reasonable amount of time as data amount increases and with its feature dimensions. The GPU (graphics processing unit) has been a point of attraction in a last few years for its ability to compute highly-parallel and semi-parallel problems way faster than any traditional sequential processor. This paper explores the capability of GPU with MapReduce Model. This highly scalable model for distributed programming can be scaled upto thousands of machines. This was developed by Google's developers Jeffrey Dean and Sanjay Ghemawat and has been implemented in many programming languages and frameworks like Apache Hadoop, Hive, and Pig etc. For this paper we'll mainly focus on Hadoop framework. First two sections present the introduction and background. The working mechanism of this combination has been shown in section 3. Then further we explore frameworks present to implement MapReduce on GPU. In section 5, a comparative experiment was performed on GPU and CPU, both implementing MapReduce Model. The paper ends conclusion.

Keyword: *Graphical Processing Unit (GPU), Hadoop, large Scale Analytics, Map Reduce Model, Java Compute Unified Device Architecture (JCUDA)*

1. Introduction

"Hadoop framework "is an open source kind of system software. When this framework used with parallel processing, it can improve performance by many folds. The very next thought strikes our mind is "can this big data processing be made even faster? What if the processes are moved from CPU processing to GPU processing? The latter is designed to perform complex graphical and mathematical tasks in 2D and 3D. Theoretically proven that a GPU can perform calculations 50-100 times faster than CPU on a process which optimized for parallel computing. This paper presents the feasibility of Hadoop over GPU and further the results of performance gain are shown in the results section.

2. Background

Since years researchers have tried to induce the capacity of GPU into Hadoop or MapReduce framework. Mars framework by Bingsheng et al was the first successful result of such efforts. Mars is a MapReduce framework for graphics processor units. Mars achieves 1.5x-16x performance jump when processing Web documents and performing big calculations and analyzing large scale web data such as searches or logs.

ISSN: 1738-9968 IJHIT

After Mars came into picture, other frameworks were also developed by the scientists to accelerate data intensive calculations using molecular dynamics, image processing, mathematical modeling such as the Monte Carlo method, financial analysis, block-based matrix multiplication etc.

With all these frameworks, BOINC came in limelight which was a fast evolving and volunteer driven middleware system for grid computing. BOINC does not use the Hadoop. It has been adopted widely and made the foundation for accelerating many scientific projects. GPUGRID is one of such projects that relies on BOINC's GPU and distributed computing for performing molecular simulations. These simulations further helps to analyze the function of proteins in health and disease. However, many projects like BOINC related to medicine, mathematics, physics and biology could be implemented using the combination of Hadoop and GPU, too.

So, there is a huge demand for accelerating parallel computing systems in combination of GPUs. Institutions which requires intensive computations either invest in supercomputers with GPUs or develop their own private solutions. Hardware vendors like Cray have launched machines which are equipped with GPUs and are pre-configured with Hadoop.

But does one size fit all? Supercomputers cost millions of dollars to provide the highest possible performance. Projects which last for a few months can only use Amazon EMR. For larger projects with duration of two to three years, investment in your own hardware can be more cost-effective. Even if the speed of calculations is increased using GPU within a Hadoop cluster, what about performance bottlenecks in transfer of data? This requires exploration in detail.

3. Working Mechanism

Data processing implies exchange of data among CPU, GPU, HDD and DRAM. Figure 1 shows the transfer of data in a commodity machine which performs computations with a CPU and a GPU.

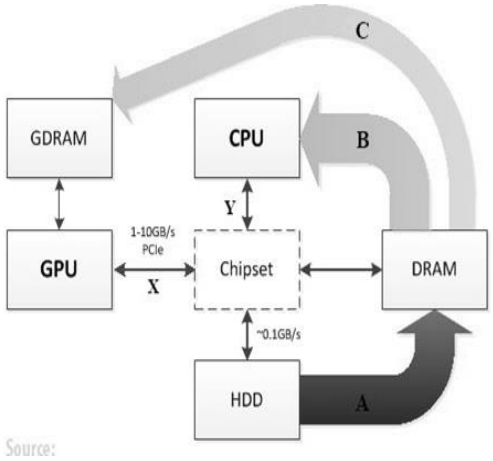

Figure 1. Data exchange among components of a commodity machine while executing a task

A: Data transfer from HDD to DRAM which is an initial step common for both CPU and GPU computing.

B: Transferring data from DRAM to chipset to CPU and processing data with a CPU.

C: Transferring data from DRAM to chipset to CPU to chipset to GPU to GDRAM to GPU and processing data with a GPU.

As a result, total time to complete any task includes:

- The amount of time required by a CPU or GPU to perform computations.
- The amount of time required for the transfer of data among all of the components.

The performance of an average CPU lies in the range of 15 to 130 GFLOPS. At the same time, Nvidia GPUs performance varies within a range of 100 to 3,000+ GFLOPS (Source: Tom's Hardware, CPU Charts 2012). However, all of these measurements are approximate and depends on the type of task and algorithm implemented to a great extent. Anyway, GPUs can increase up the computation speed by nearly 5 to 25 times per node in some scenarios. However, Claims by some developers states performance speed up of 50x-200x if the cluster consists of several nodes. For instance, MITHRA project developers achieved a 254x increase.

However, different types of hardware have different data transfer speeds. Although the supercomputers are mostly optimized to work with GPUs, a regular computer or a server can be much slower when exchanging the data.

The Data rate transfer between CPU and chipset is around 10 to 20GB per second (as depicted in Figure 1, point Y). The data transfer speed of a GPU to Dynamic RAM is around 1 to 10GB per second (as depicted in Figure 1, point X). Although some systems such as PCIe v3 may reach up to around 10 GB per second, in most of the standard configurations data flows between a GDRAM (GPU's DRAM) and the DRAM of the system at the rate of around 1GB per second.

On a concluding note, although GPUs provide faster computing, the main bottleneck is the slow transfer of data between CPU memory and GPU memory (Point X). Therefore every time, the time spent on data transfer from/to a GPU (refer Arrow C) will be measured against the time saved due to GPU acceleration. So, the best thing will be first evaluate the actual performance on the small cluster and then perform an estimation of how the system will perform on a larger scale.

A case study in 2010 by Intel provides performance results for 14 different types of ideal use cases. “Performance boost up of 10x to 1000x can be achieved per node, instead 2.5x-3x or so will be more realistic. The overall optimization for a cluster can be even smaller”, said the Intel's figures.

So, since the data transfer speed may be slower, the ideal use case is when the input and output data is relatively small for each GPU, compared to the number of computations to be performed. It must be noted that type of the task to be performed should match the GPU's capabilities. This task should be divided into independent sub-processes running with Hadoop framework in parallel.

Some examples of such tasks can be calculating a complicated mathematical formula like matrix multiplication or generating very large sets of random values or a similar scientific modeling tasks or any other general-purpose GPU applications.

Some libraries or bindings have to be used for creating a prototype and accelerate the big data system using Hadoop in combination with GPUs, which allow for accessing a GPU. Today, the main tools which can be used to extract GPU’s capabilities are as follows:

1. **JCUDA:** This project provides Java bindings for the Nvidia CUDA and many related libraries like JCusparse (this library is for working with the matrix), JCublas, JCurand (this library is for generating random numbers in the GPU), JCufft (Java bindings for general signal processing), etc. Point to be noted is that, these libraries will only work for Nvidia GPUs.

2. **Java Aparapi:** Aparapi performs conversion of Java bytecode to OpenCL at runtime and execution of it on a GPU. Of all of the systems using GPUs with Hadoop for computations, Aparapi and OpenCL method have got the best long term perspectives. AMD JavaLabs, laboratory of AMD developed the Aparapi. It was released in 2011 as an open-source product and since then it is growing rapidly. Some real-life use cases are worth looking for this technology at AMD Fusion Developer Summit conference's official website.

   Being an open and cross-platform standard, OpenCL is supported by a large number of hardware vendors which facilitate writing the same code base for both the GPU and the CPU. In case when there is no GPU installed on a machine, OpenCL uses its CPU.

Creating native code for accessing GPU: Consequently, the performance will be way faster than in the solutions which are using connectors and bindings. However, the solution is to be delivered in the shortest possible time, the frameworks like Aparapi can be opted. But its performance is not satisfactory, the native code can partially or completely replace the original Aparapi code. The resulting product will be considerably faster, at the same time, much less flexible.

C language API can be used with OpenCL or Nvidia CUDA for creating the native code that empowers Hadoop to use the GPU via JNA (in case the application is coded in Java) or Hadoop Streaming (in case the application is coded in C).

4. GPU-Hadoop Frameworks

There were a lot of custom GPU-Hadoop frameworks after the launch of Mars project. These include C-MR, SteamMR, GPMR, Grex, Panda, Shredder and many others. However, most of these frameworks are no longer supported and were developed for some particular scientific projects. Therefore, a Monte Carlo simulation framework can be hardly applied for a bioinformatics project (say) based on the other algorithms.

Moreover, processor technologies are evolving very rapidly. Many new revolutionary architectures have been developed for Sony PlayStation 4, Mali GPU by ARM, Adapteva Multicore Microprocessor etc. Both Mali GPU and Adapteva will be compatible with OpenCL. Xeon Phi co-processor works with OpenCL too, which was launched by INTEL. It has an x86-like architecture and is a 60 core co-processor that supports the PCI Express standard. It consumes just 300watts of energy and provides by giving a performance of 1 TFLOPS in double precision. Tianhe-2 which is the most powerful supercomputer till date implements this co-processor. Though, it is very difficult determine which Processor Computing framework architecture will more robust and high performance.

5. Results & Simulation Analysis

A sample job was creating which implemented MapReduce model in both of its version. Each version was written in JAVA. One was optimized to run on CPU and other on GPU. Hardware used for running CPU job was intel (R) core 2 duo 1.7 GHz and for GPU job was Asus NVIDIA GeForce EN210 Silent 589 MHz. The figure below shows the outcomes of the experiment. A huge difference in runtime was observed. GPU and MapReduce combination outperformed CPU + MapReduce. This analysis is presented into figure2.

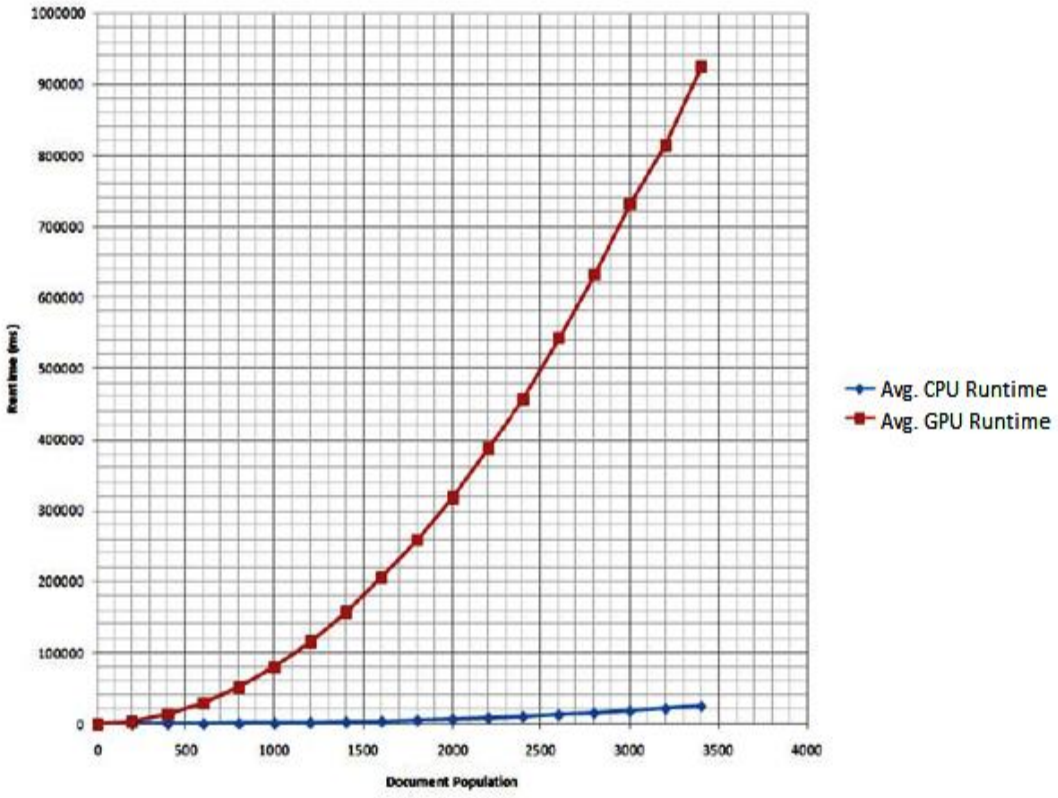

Figure 2. Shows COMPARATIVE PERFORMANCE for both CPU and GPU Runtime

6. Conclusion

This paper presents the implementation of MapReduce Model over GPU. The results from section 5 shows how much better is GPU over CPU for parallel processing when both run a MapRed job. The challenges presented in the very next section requires attention and are needed to be addressed. But difference in performance of GPU and CPU predicts that the MapReduce and GPU combination is the future of High Processing Computation.

Reference

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In Proceedings of 6th Symposium on Operating System Design & Implementation (OSDI 2004), San Francisco, USA, 2004, pp. 137-150.
[2] J. Stuart and J. Owens. Multi-GPU MapReduce on GPU Clusters. In Proceedings of IEEE International Parallel & Distributed Symposium (IPDPS 2011), Anchorage, USA, 2011.
[3] R. Favriar, A. Verma, E.Chan, R. Campbell. MITHRA: Multiple data independent Tasks on a Heterogeneous Resource Architecture. In Proceedings of IEEE International Conference on Cluster Computing and Workshops (CLUSTER 2009), New Orleans, USA, Sept 2009, pp.1-10.
[4] K. Shirahata, H. Sato, S. Matsuoka. Hybrid Map Task Scheduling for GPU-Based Heterogeneous Clusters. In Proceedings of IEEE 2nd International Conference on Cloud Computing Technology and Science (CloudCom 2010), Indianapolis, USA, Dec 2010, pp.733-740.
[5] D. B. Kirk and W. W. Hwu, Programming Massively Parallel Processors. Morgan Kaufmann, 2010.
[6] V. Starostenkov. Hadoop + GPU: Boost performance of your big data project by 50x 200x?, in CIO Asia Magazine, June 2013.
[7] J. Ekanayake and G. Fox. High performance parallel computing with clouds and cloud technologies, in Proceedings of the First International Conference on Cloud Computing, Oct. 2009.

[8] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis. Evaluating MapReduce for Multi-core and Multiprocessor Systems. Symposium on High Performance Computer Architecture (HPCA), 2007.
[9] http://bookmarks.hadoopsphere.com/
[10] P. ZHOU, J. LEI and W. YE, "Large-Scale Data Sets Clustering Based on MapReduce and Hadoop," Journal of Computational Information Systems, vol. 7, no. 16, pp. 5956- 5963, 2011.
[11] Miao Xin; Hao Li, "An Implementation of GPU Accelerated MapReduce: Using Hadoop with OpenCL for Data and Compute-Intensive Jobs," In Proceedings of International Joint Conference on Service Sciences (IJCSS 2012) , pp.6-11.

