An Adaptive Algorithm of K-means on HSA Platform

Zhenshan Bao
College of Computer Science
Beijing University of Technology
Beijing, China
baozhenshan@bjut.edu.cn

Qi Luo
College of Computer Science
Beijing University of Technology
Beijing, China
luoqi0620@emails.bjut.edu.cn

Wenbo Zhang
College of Computer Science
Beijing University of Technology
Beijing, China
zhangwenbo@bjut.edu.cn

***Abstract*—With the growing demand for high performance computing, heterogeneous system, based on CPU-GPU architecture, were widely applied because of its higher processing capacity. To reduce the communication latency between CPUs and GPUs or other agents, HSA (Heterogeneous System Architecture) Foundation proposed a kind of an open industry standards body. In this paper, we designed the K-means algorithm on Kaveri APU, which complied with HSA standard, and optimized it into an adaptive one to obtain high utilization effect of both CPU and GPU. Experimental results show that our adaptive algorithm gets a 45.2% average decrease in overall execution time.**

Keywords—Heterogeneous computing; HSA; K-means; workload division

I. INTRODUCTION

In recent years, multi-core had becoming the inevitable trend of the processor development. On the other hand, the development of GPU (Graphics Processing Unit) was driven by multi-core era, semiconductor technology greatly help the GPU precision and complexity of growth. Modern GPUs were not only the powerful graphics engines, but also the highly parallel programmable processors, its computing capacity and memory bandwidth far exceeded the CPU (Central Process Unit) [1]. In order to get better adaption in high performance computing, GPU-CPU heterogeneous architectures had been increasingly adopted in scientific research. Because of GPUs' capabilities of providing high computational throughput and the combination of CPU and GPU, GPU-CPU heterogeneous architectures could coordinate and handle the serial working mode and parallel working mode. For example, the recently built supercomputer Tianhe-2 was the world's fastest supercomputer according to the TOP500 lists for June 2013, November 2013, June 2014, November 2014, June 2015, and November 2015.

In addition, we must have a better understanding of hardware during the process of development, because GPU Programming technology was so difficult. With CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language) releasing, developers need not to pay attention to the hardware and were liberated from the heavy development task. There still were some problems in CPU-GPU heterogeneous system, so AMD joint other manufacturers put forward a new industry standards body focused on making it dramatically easier to program heterogeneous computing devices, the industry standards body aimed to solve the communication delay between CPU and GPU. Compared with traditional CPU-GPU heterogeneous systems, HSA had a lot of ascension, the most important one was heterogeneous Uniform Memory Access (hUMA). Based on the feature, GPU and CPU could have the same status in the process of computing tasks. Our paper designed a K-means algorithm on HSA platform which is similar to original CUDA implementation. More importantly, we designed an adaptive algorithm of k-means by using the characteristics of hUMA and making full use of CPU resources.

The rest of this paper is organized as follows. In section II, highlights the differences between this paper and related work. Then an overall about algorithm design and implementation of strategy will be described in Section III. Some preliminary experiment results are presented and discussed in section IV. Section V summarize this work and describes the future works of implementation of strategy in HSA.

II. RELATED WORKS AND MOTIVATION

This section introduces the HSA system, related works and motivation for this study. Section A describes the features of HSA. Section B describes the K-means model, Section C presents some existing research about our works. Section D describes the details of the problem to be solved.

A. HSA

In traditional heterogeneous computing platforms, GPU and CPU are relatively independent of each other. There are no unified software environment and ability to deal with procedures calling CPU and GPU automatically. In order to integrate the discrete CPU and GPU, AMD and some manufacturers set up HSA Foundation. The Heterogeneous System Architecture (HSA) is designed to efficiently support a wide assortment of data-parallel and task-parallel programming models. A single HSA system can support multiple instruction sets based on host CPUs and kernel agents. By reshaping computing systems, processing units that is divided on the same platform closely integrated to a single chip. By using HSA, program can be set up data structure in the unified address space and create a task in the proper processor. The motivations of HSA Foundation is to improve portability, programmability, manageability and performance for the next-generation heterogeneous computing. There are two essential features for users: one is hUMA; the other is heterogeneous Queuing (hQ).

1) hUMA

In the past, even if the GPU and CPU are integrated into the same chip, CPU and GPU data storage areas are independent of each other, chips get memory space information still have to undergo complex processes when performing general purpose computation. When the part of CPU program is operated on the GPU, all the information on the CPU storage area must be copied to the GPU's storage. And when the operation of GPU is completed, the data will be copied to the CPU. In order to solve this problem, HSA Foundation announced hUMA technology, through the hUMA, CPU and GPU can share the same storage space, and CPU can directly access the GPU memory address. In addition, the range of memory that the GPU can access in HSA is now as large as the virtual memory space allows, it can significantly simplify programming on GPU or other accelerators. As shown in Figure 1, GPU and CPU have uniform visibility into entire memory space.

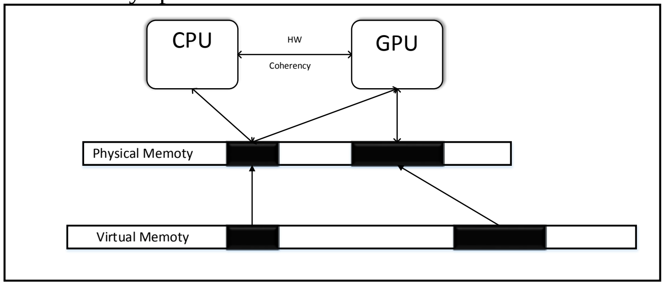

Fig. 1. HUMA technology

2) hQ

HSA devices communicate with one another using queues. Queues are an integral part of the HSA architecture. [2] HSA supports hQ which aims to simplify the distribution of computational jobs among multiple CPUs and GPUs from the programmer's perspective. As shown in Figure 2, GPU and CPU have equal flexibility to be used to create and dispatch work items.

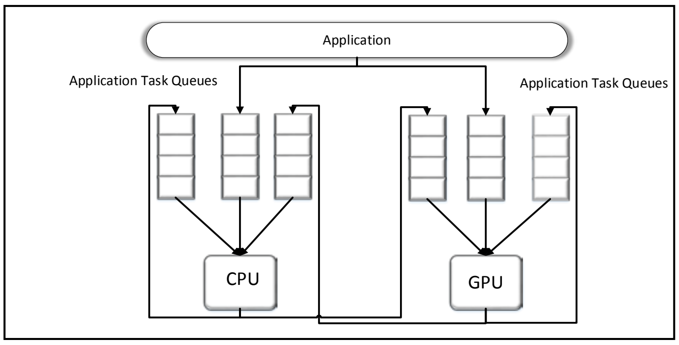

Fig. 2. HQ technology

In the traditional mode, the CPU occupied the dominant position, it can rapidly direct the distribution of task to itself or the GPU. When assigned to the GPU, it must be undergo complex process for copying data. At the same time, it need transcoding through operating system services and kernel mode when the tasks are assigned to the GPU, the scheduling overhead has a great delay. hUMA, famous for the uniform memory access model, is the basis of hQ, and it made the GPU work with the CPU by using shared memory, the GPU can directly access to user tasks in memory, resulting in performance ascension.

B. K-means

K-means algorithm is put forward by j. b. Mac Queen[3] in 1967, it is a classical cluster algorithm based on partition. It had become one of the most widely used clustering algorithm in the field of data mining, machine learning, pattern recognition and quantity statistics, because it had high efficiency and was easy to be combined with other methods. K-means cluster algorithm aims to partition **n** observations into **k** clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. The flow diagram of algorithm is shown in Figure 3.

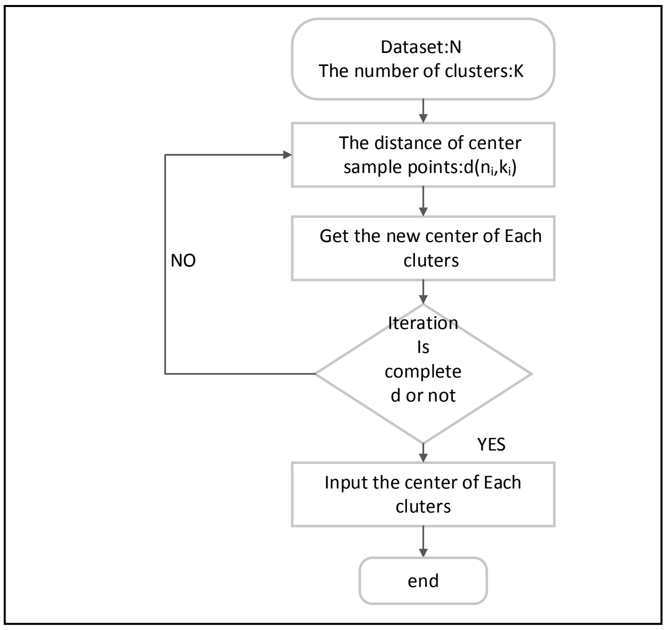
Dataset:N
The number of clusters:K
The distance of center sample points:d(n$_i$,k$_i$)
Get the new center of Each cluters
NO
Iteration Is complete d or not
YES
Input the center of Each cluters
end

Fig. 3. Diagram of algorithm

C. Related works

There are some existing research efforts to use CPU-GPU heterogeneous systems. Chen et al. [4] designed a new parallel algorithm which exploits the parallelism of the column of similarity matrix to parallelize the Smith-Waterman algorithm on a heterogeneous system based on CPU and GPU. Klages et al. [5] proposed optimal task split between CPU and GPU where GPU is only used to compute the calculation of the particle force. In order to improve the performance of the traditional LSM both in terms of efficiency and effectiveness, Balla-Arabé et al. [6] proposed a novel algorithm based on the lattice Boltzmann method (LBM).

Recently, K-means implementation on the GPU drawn the attention of researchers. Zechner et al. [7] proposed an optimized k-means implementation on CUDA. While their target was solely to best exploit the computational capabilities available by using CUDA. Li et al. [8] designed an algorithm that exploits GPU on-chip registers to significantly decrease the data access latency. Compared with those previous studies that only pay close attention to GPU cores, our adaptive algorithm coordinates both GPU cores and CPU cores for maximized efficiency. Recently, Ma et al. [9] used linear regression models to estimate the performance and power of GPU applications in order to decide where (and how) to run GPGPU (General Purpose Graphics Processing Units) programs. However, the coordination between CPU and GPU is not effective in their work.

D. Motivation

CPU-GPU heterogeneous systems transferred data slowly between CPU and GPU before HSA platform appeared. In such an environment, GPUs were assigned to the most of computing workload, and thus they provided up to 10 times the computing capacity of CPUs. However, in practical applications, CPUs are often idle while GPUs take all the workloads. Compared with the use of CPU, it would spend too much time on data transmission before HSA platform appeared. Nowadays, Unified Coherent Memory enables all compute elements to have access to the same data, so it enables the application to run on the best compute element.

If the resources of CPU would be used efficiently in heterogeneous systems, it could reduce the task execution time and further improve the performance of HSA. As mentioned, if the data transmission consumption between CPUs and GPUs could be reduced or removed, the total execution time can be further reduced. Nowadays, the emergence of HSA can realize our ideas. In addition, GPU and CPU are used as a combined computing unit, which makes a good use of all the available hardware resources.

III. Adaptive Algorithm of K-means

Since workload division within an application was a software problem by nature, it did not suit for on-chip hardware implementation. Therefore, we only have to design our algorithm on software level. CPU and GPU were independent of each other in the traditional CPU-GPU heterogeneous system, data was transmitted between CPU and GPU through PCI-E bus and data transfer was slow. As a result, we could import data to the GPU memory for parallel processing at one time, even though large amount of data would be processed. Firstly, we used the original algorithm idea on HSA platform. According to the research on parallelism of K-means algorithm, it need to calculate the distance from each sample points to all central point and get a new clustering which was composed of shortest distance points[10]. The process was conducted on the GPU for parallel processing, then the processing result was passed to the CPU. Our main job was to improve this process.

First of all, it should be pointed out that, tasks on the host will be through the function (hsa_signal_wait_acquire) waiting for the signal that the GPUs completed the tasks, after informing GPU agent of running the program. This function would call “WaitRelaxted” (in HSA runtime library) to loop with reading the value of the semaphore, this operation with the method of continuous polling would cause the task deadlock which was waiting for the change of the semaphore, the result was that the host always occupied a core of CPU on the task, but the other cores of CPU always were idle. We used the synchronous operation to protect the shared data sets in the CPU and GPU tasks. We used the “pthread_mutex_lock” to protect for the description of the data to be processed.

Our algorithm was based on ‘pthread_mutex_lock’. Because there was one core at full capacity, other three core were not used. We created three threads for other three core by pthread and one thread for GPU. Firstly, we have to chop workload in pieces, then we respectively assigned a piece workload for each thread. Once thread have finish working, we assigned next piece workload for it. We used ‘pthread_mutex_lock’ to control coordination mechanism.

IV. Experimental results

In this section, we will introduce our experimental environment and our experimental results.

A. Environment

Our hardware environment was given priority to AMD APU. We use A10-7850K in our test bed. A10-7850K was the first formal comprehensive products supporting hUMA technology. The operating system is Ubuntu 15.04 with a Linux kernel 3.19. We used kfd v1.4 HSA drivers and the HSA Runtime 1.0f. We used AMD CodeXL which was a comprehensive tool to obtain the GPU utilization and usage time. It turned out on a Linux system there was this command called "time" for application execution time. Final result was based on the average of multiple tests.

B. Experimental results

Firstly, in order to verify our algorithm which can improve the execution efficiency, we defined the percentage of work that GPU took in an iteration as r. Then CPU took the rest 1 – r percentage of the work. The value of r was from 0 to 100.The data we used have 1 million data points, each point has 34 properties and the initial number of cluster is 10. We can find that the optimal value of r is between 50 to 70. The experimental results is shown in Figure 4. As a result, the efficiency is improved. On the other hand, we could find that the computing power of GPU is far more than the CPU.

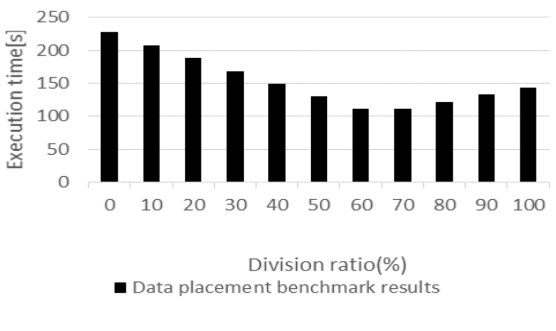

Fig. 4. Fixed workload division

We divide the work into **n** equal parts, CPU and GPU are assigned a part of work for each. Once the CPU or GPU is performed, the next part of work would be assigned to it. As the whole data size is different, the number of iterations will be different. Our test results are the previous 100 iterations time. The data points we used is from 0.1 million to 1.9 million, each point has 34 properties and the initial number of cluster is 10. Figure 5 presents the execution time of our scheme compared with GPU takes all work. The Figure 5 shows that our approach saves 52.64% of execution time on average.

We now give an analysis of the performance of adaptive algorithm of k-means. Firstly, original algorithm only use GPU during the process of classifying clusters, besides, the adaptive algorithm use three thread for parallel processing on CPU. There is no the consumption of data copying and workload adds a work object, so the performance of adaptive algorithm increase significantly.

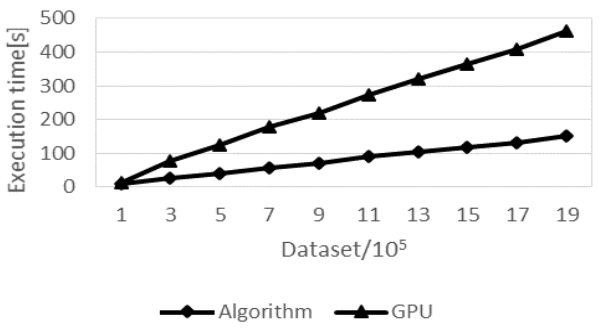

Fig. 5. Adaptive algorithm and original algorithm of k-means under different scale of data

V. CONCLUSION AND FUTURE

Current research on GPU-CPU architectures focuses mainly on the hardware aspects, while the workload division between GPU and CPU of such systems receives much less attention. There are few existing studies that start to use CPU to handle the workload, but it is limited by slow data transfer between CPU and GPU. In this paper, we have presented an adaptive algorithm of k-means on HSA platform. Our solution features an adaptive algorithm design. We implement the adaptive algorithm using the HSA framework on a real physical test bed with AMD A10-7850K. Experiment results show that our algorithm achieves 52.64% average time savings. On the one hand, the algorithms are ported to HSA will be more and more, the adaptive framework would suit other algorithms. On the other hand, the energy efficiency of such systems receives much more attention. In the future, we will study a set of adaptive framework and energy efficiency of HSA.

The growth of heterogeneous systems represents a solid trend in modern systems, and we believe that future work on other machine learning algorithm in this domain can benefit from the promising insights into scalability demonstrated by our experimental study.

ACKNOWLEDGMENT

This work was founded by the significant special project for Core electronic devices, high-end general chips and basic software products (2012ZX01039-004).

References

[1] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). Gpu computing. Proceedings of the IEEE, 96(5), 879-899.

[2] Blinzer, P. (2014). The Heterogeneous System Architecture: It's beyond the GPU. International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (pp.iii-iii). IEEE.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[3] Chen, B., Xu, Y., Yang, J., & Jiang, H. (2010). A New Parallel Method of Smith-Waterman Algorithm on a Heterogeneous Platform.. Algorithms & Architectures for Parallel Processing, International Conference, Ica3pp, Busan, Korea, May (Vol.6081, pp.79-90).

[4] Queen, J. M. (1967). Some methods for classifications and analysis of multivariate observations. Berkeley University of California Press, 281--297.

[5] Klages, P., Bandura, K., Denman, N., Recnik, A., Sievers, J., & Vanderlinde, K. (2012). Astrophysical particle simulations on heterogeneous cpu-gpu systems. Hep Websearch Hep.

[6] Balla-Arabé, S., Gao, X., Ginhac, D., & Yang, F. (2016). Shape-constrained level set segmentation for hybrid cpu–gpu computers. Neurocomputing, 177, 40-48.

[7] Zechner, M., & Granitzer, M. (2009). Accelerating K-Means on the Graphics Processor via CUDA. First International Conference on Intensive Applications and Services (pp.7-15). IEEE Computer Society.

[8] Li, Y., Zhao, K., Chu, X., & Liu, J. (2013). Speeding up k -means algorithm by gpus. Journal of Computer and System Sciences, 79(2), 216-229.

[9] K. Ma, X. Li, W. Chen, C. Zhang, and X. Wang, “GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU Heterogeneous Architectures,” in Proc. of the Int’l Conf. on Parallel Processing (ICPP),2012.

[10] Zailong, W. U., Zhang, Y., Jianliang, X. U., Jia, H., Yan, S., & Xie, Q. (2014). Research on kmeans algorithm optimization based on opencl. Journal of Frontiers of Computer Science & Technology.