Advances in Intelligent Systems Research (AISR), volume 155
2018 International Conference on Advanced Control, Automation and Artificial Intelligence (ACAAI 2018)

Workload Partitioning Algorithm Based on Performance Curve of GPU in Heterogeneous Platforms

Hongyu Yang[1], Hui Chen[1], Chengming Li[1], Qingshan Jiang[1,*], Xueyuan Cai[2]
[1]Shenzhen Institutes of Advanced Technology, Shenzhen 518055, China
[2]Shenzhen Vispractice Technology Corporation, Shenzhen 518055, China
*Corresponding author

Abstract—**With the development of GPU's general computing power, hybrid systems composed of multi-core CPU and GPU are becoming more and more popular in data parallel applications. Because the performance of GPU is related to the magnitude of the load received, effective load allocation methods are very important for improving the performance of data parallel applications. The existing static load distribution methods fail to use the characteristics effectively - GPU performance changed with the load, causing the load unbalanced. Dynamic load distribution methods easily reduce the performance of the system due to the excessive synchronization and data transmission operation. In this paper, we propose a new workload partitioning algorithm, which takes advantage of the characteristics of GPU performance varying with the workload in off-line analysis stage, and uses the successive decreasing method to determine the optimal load allocation ratio between multi-core CPU and GPU. The effectiveness of the load allocation algorithm is verified on the remote sensing data set based on the median filtering algorithm.**

Keywords—GPU; hybrid system; data parallel applications; workload partitioning

I. Introduction

With the rapid development of the semiconductor technologies, a hybrid system which has a multi-core processor and GPU is widely used in a modern computer system, which system has the potential to improve special application performance by using the GPU distinct hardware architecture [1], [2].

For ease use of GPU unique computing power, some low-level programming languages such as OpenCL[3], CUDA[4],OpenMP4.5[5] have been proposed.

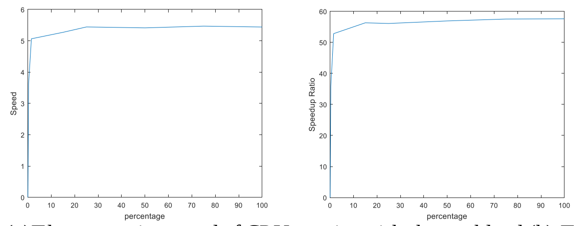
Data parallel application assumes that CPU and GPU are processing the same task whose data can be processed in parallel simultaneously. For a given data parallel application, the best performance of the application depends not only on the single processing unit in the system, but also on the cooperation among the processing units. So it is important to study the proper workload distribution between CPU and GPU to achieve load balance [6]. The performance of GPU varies with the workload, which makes a challenge to workload partitioning between CPU and GPU. We propose a method of workload partitioning based on the performance curve of GPU, which can effectively use the characteristics of GPU performance varying with the workload in off-line analysis stage, so as to determine the optimal partition proportion between CPU and GPU. We choose the image median filtering algorithm [7] as benchmark algorithm, and the result shows better effectiveness of the algorithm in remote sensing data sets.

The rest of the paper is organized as follows: Section 2 introduces the background and related works. Section 3 introduces the details of the proposed workload partitioning algorithm. Section 4 carries out the experiment and analyses of the experimental results. Section 5 draws the conclusion.

II. Background and Related Work

In recent years, there are a lot of works on how to achieve load balance in CPU-GPU hybrid system. Qilin [8] use the pre-trained linear model to represent the performance of GPU for increasing workload. [9] uses functional performance model to solve the optimal distribution of the workload. Huang [10] uses the computation and communication overlapped through multi-stream concurrent technology to reduce the data transmission bottleneck between CPU and GPU. [11] adopt machine learning to determine the optimal partitioning. [12] proposed a systematic approach by using modeling, profiling and prediction technique to solve workload partitioning. Tse [13] uses two basic scheduling policies, exponential incremental and linear incremental. For each time a processor requests a block of data, the schedulers increase the data blocks in an exponential or linear way. HDSS [14] hand out the data blocks in an exponential way in the adaptive phase, until the processing speed of each processing unit is relatively stable. In the execution stage, the remaining data blocks are allocated directly according to the relative execution speed of the adaptive stage. In [15], a similar way was proposed, but it dynamically adjusts the proportion of the workload partition until proportion is similar at near two times. However the above work fails to utilities the characteristics of GPU performance varying with the load, may cause uneven load. We study the performance of GPU varies with the different load that it received as shown in Figure 1.

The experimental environment is shown in Table 1, and the experiment uses the first picture of the first data set. The X axis represents the percentage of the data allocated to GPU processing. In Figure 1(a), the Y axis represents the row number



(a)The processing speed of GPU varying with the workload (b) The speedup of GPU varying with the workload

FIGURE I THE PERFORMANCE OF GPU VARYING WITH THE WORKLOAD

of image data processed in milliseconds, and the Y axis in Figure 1(b) represents the acceleration ratio when dealing with the same size data with single core of CPU.

As we can see, the performance of GPU increased fast at the beginning of the curve, and then become almost invariable. Our new workload partitioning algorithm which takes the characteristics of GPU performance varying with the workload into account show better performance.

III. WORKLOAD PARTITIONING ALGORITHM BASED ON PERFORMANCE CURVE OF GPU

The workload partitioning algorithm based on the performance curve of GPU determines the workload of each processor, which has two stages – the offline analysis and the execution.

For better description of the proposed workload partitioning algorithm, let α denotes the speedup for only GPU—the ratio between the processing time of CPU and the processing time of GPU when process the same data volume, αcg denotes the speedup for CPU+GPU—the ratio between the processing time of CPU and the processing time of CPU + GPU.

The algorithm of the offline analysis stage is as follows: First, introduce a formula for data partition assuming that the processing speed of CPU and GPU is a constant in the different workload. The total data volume is S, the data allocated to CPU and GPU are S_{cpu} and S_{gpu}. The optimal distribution is achieved when CPU and GPU work at the same time, then we can get S_{cpu} as follows:

$$S_{cpu} = \frac{S}{1+\alpha} \tag{1}$$

Equation (1) determines the amount of data that should be allocated to the CPU when the CPU and GPU processing speed are constant. Second, form figure 1 we know that the processing speed of GPU is basically stable only when a large amount of data is assigned to the GPU, so we assume that when the optimal distribution is achieved the data assigned to CPU and GPU respectively are S_{Rcpu} and S_{Rgpu}, and the processing speed of CPU and GPU respectively are V_{Rcpu} and V_{Rgpu}, we have:

$$S_{Rcpu} = S_{cpu} \times \beta \tag{2}$$

$$S_{Rcpu} + S_{Rgpu} = S \tag{3}$$

$$\frac{S_{Rcpu}}{V_{Rcpu}} ; \frac{S_{Rgpu}}{V_{Rgpu}} \tag{4}$$

Equation (2), β is an adjustment factor, used to adjust the size of the data assigned to the CPU with α. When the right and left of (4) is equal, the optimal distribution is achieved, thus the run time is the least and the acceleration ratio of the system is the largest, therefore, we need to determine the optimal value β in the off-line analysis stage.

We use $\alpha cg - \alpha$ to determine whether β reaches the optimal value, this's because when the heterogeneous system composed of GPU and CPU achieve load balance the run time is the least, then $\alpha cg - \alpha$ has the maximum value, and α remains the same. So as $\alpha cg - \alpha$ takes the maximum value, we get the optimal value of β.

Third, how to get the optimal value of β at the same time reducing the time of the offline analysis? Based on the characteristics of the GPU performance varying with the load, we use successive decline method to solve β. According to the results of the offline analysis, we find that when the initial value β is 0.9, and each iteration is $\beta = \beta$ -0.1, the best value β can be obtained after very few iterations. The algorithm flowchart is shown in Figure 2. In this flowchart, α and β are as global variables, and α is a constant—the ratio of CPU processing time to GPU processing time when processing the same data volume S.

Figure 2 shows a complete process of solving the optimal value β. First, let β is 0.9, then run CPU+GPU mixture program (process the workload by CPU and GPU in parallel in Figure 2). On account of (2), the data is allocated to CPU, the rest to GPU, then make CPU and GPU run at the same time,

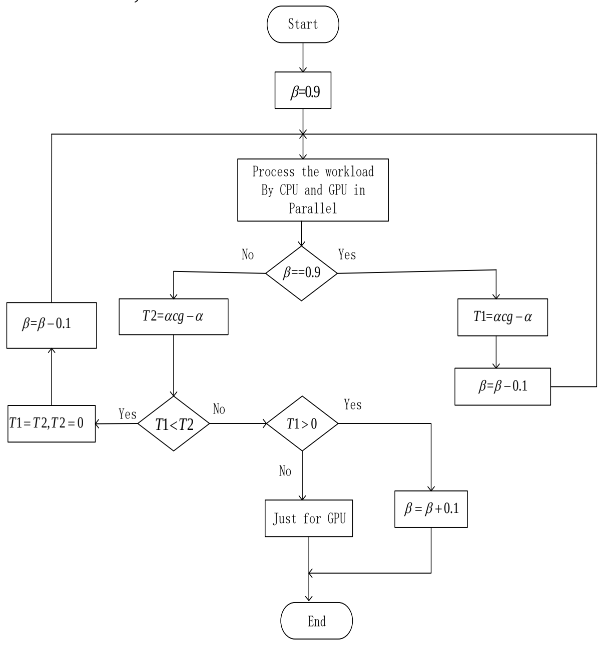


FIGURE II THE PROCESS TO GET THE OPTIMAL VALUE β

and measure the operation time, calculate αcg. If the value of β is 0.9, then T1=αcg-α, and β is reduced by 0.1. If the value of β is not equal to 0.9, then T2=αcg-α, then T1 and T2 will be compared, if T1<T2, β is reduced by 0.1, then CPU+GPU mixture program is run until T1>T2. Thus, it's considered we get the best proportion of data distribution, and the off-line analysis stage is over.

Once the off-line analysis stage is over, we allocate the data to CPU according to (2) or only uses GPU at the execution stage.

IV. Experiment and Analysis

A. Experimental Environment and Implementation

The evaluation environment is listed in Table 1. We chose OpenCV [16] to read color satellite remote sensing pictures in TIFF format. For each picture, we adopt three versions: CPU only (single thread), GPU only (using CUDA), and a hybrid version. The hybrid version has two scheduling strategies which are Qilin algorithm and workload partitioning algorithm based on performance curve of GPU.

Qilin algorithm requires a pre-training period to develop a linear model of performance for increasing data. Just as the technique suggested, we have performed two runs, the initial being 5% of the total data, followed by 5% every time, then we get a specific linear model. Later, the workload is partitioned of each processing units bases on their linear model. We use Qilin1, Qilin2, Qilin3 and Qilin4 to express that the number of the physical cores of CPU working with GPU are separately one core, two cores, three cores, and four cores. For workload partitioning algorithm based on performance curve of GPU, we consider the different number of physical cores of CPU as a whole and represent their results as 1*CPU+GPU, 2*CPU+GPU, 3*CPU+GPU and 4*CPU+GPU. E.g. in the two CPU physical cores participating the operation, we put the two physical cores as a whole, and each physical core processes the same amount of data in the offline, also when the data are distributed, and their results are expressed as 2*CPU+GPU.

B. Applications and Data

We use median filtering algorithm as a benchmark to evaluate workload partitioning algorithm based on performance curve of GPU. The median filtering algorithm is non-linear smoothing technique, which sets the gray value of each pixel in the image to the median of the gray values of all the pixels in a certain neighborhood of the point.

The data used in the experimental evaluation are derived from the 16bit color remote sensing images which was provided by Geospatial Data Cloud site, Computer Network Information Center, Chinese Academy of Sciences [17]. The specific remote sensing data come from the satellite of Landsat8 OLI-TIRS, and the administrative scope is Nanshan District, Shenzhen city, Guangdong Province, and the remote sensing image storage format is TIFF. According to the selected time range, our experimental data sets are divided into four groups, first sets of data are form January 1, 2015 to December 31, 2015, which label as LC81220442015291LGN00; second sets of data are from January 1, 2016 to October 31, 2016, which label as LC81220442016054LGN00; third sets of data are from November 1, 2016 to December 25, 2016, which label as LC81220442016326LGN00; fourth sets of data are from January 1, 2017 to September 19, 2017, which label as LC81220442017120LGN00. The details of picture using are shown in Figure 3.

C. Results and Analysis

For the effectiveness of the new workload partitioning algorithm, we use the four groups of data to test the new workload partitioning algorithm proposed in this paper. The experimental results are shown in Figure 4, Figure 5, Figure 6, and Figure 7. Speedups are obtained by comparing with single-thread CPU execution.

TABLE I. EVALUATION ENVIRONMENT

Name	Description	Name	Description
CPU	Intel Xeon E5-2620	CPU code compiler	GCC/G++ 4.8.4
GPU	NVIDIA Tesla K20C	GPU code compiler	NVCC 7.0

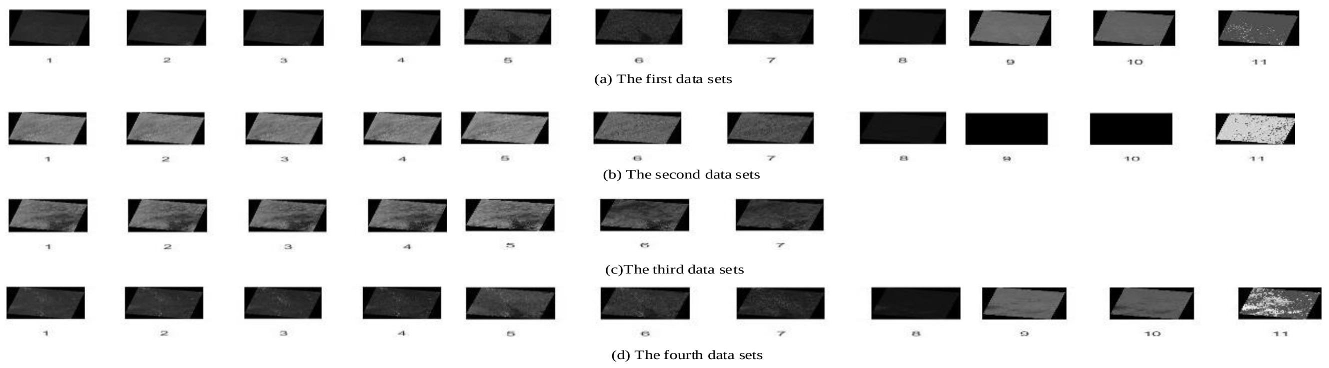
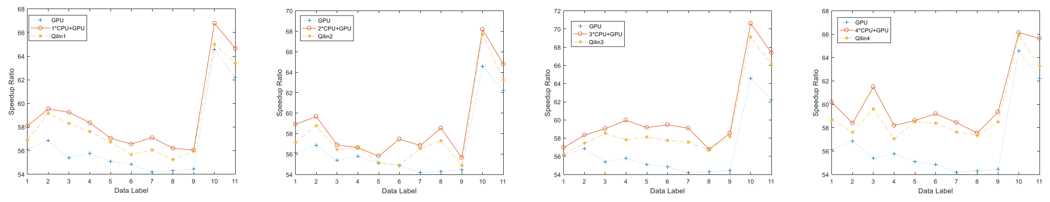
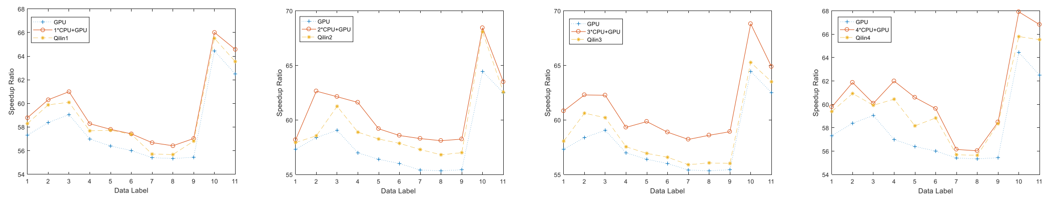


FIGURE III. DATA SETS FOR EXPERIMENTS



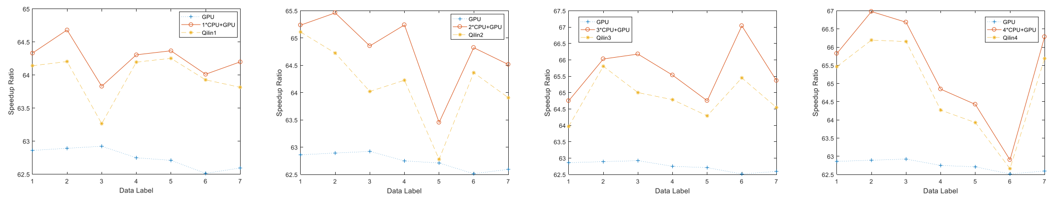
(a) 1*CPU+GPU (b) 2*CPU+GPU (c) 3*CPU+GPU (d) 4*CPU+GPU

FIGURE IV. THE CONTRAST DIAGRAM OF SPEEDUP RATIO CURVE FOR FIRST DATA SETS



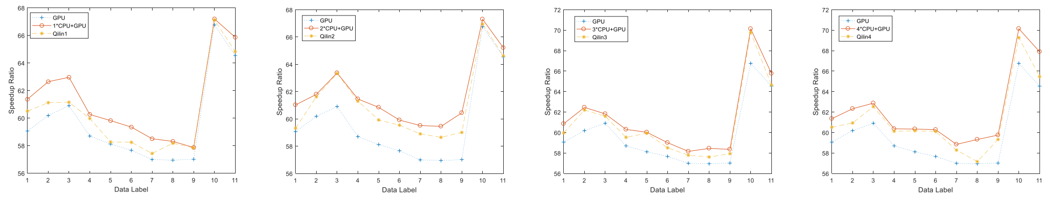
(a) 1*CPU+GPU (b) 2*CPU+GPU (c) 3*CPU+GPU (d) 4*CPU+GPU

FIGURE V. THE CONTRAST DIAGRAM OF SPEEDUP RATIO CURVE FOR SECOND DATA SETS



(a) 1*CPU+GPU (b) 2*CPU+GPU (c) 3*CPU+GPU (d) 4*CPU+GPU

FIGURE VI. THE CONTRAST DIAGRAM OF SPEEDUP RATIO CURVE FOR THIRD DATA SETS



(a) 1*CPU+GPU (b) 2*CPU+GPU (c) 3*CPU+GPU (d) 4*CPU+GPU

FIGURE VII. THE CONTRAST DIAGRAM OF SPEEDUP RATIO CURVE FOR THIRD DATA SETS

From Figure 4, Figure 5, Figure 6, and Figure 7, it can be seen that in comparison with the GPU version, the speedup ratio of hybrid version is larger, which meets the basic goal of load balance in the heterogeneous system. Under the same hardware configuration, the new workload partitioning algorithm can provide greater speedup ratio than Qilin algorithm. This is because the new load balance algorithm gradually reduces the data allocated to CPU in the off-line analysis stage until the difference between the speedup reaches the max, so as to get the best data allocation ratio. Then the CPU and GPU execution time are approximately equal, which reduce the overall execution time and make the acceleration ratio reach the maximum.

V. CONCLUSIONS

Hybrid systems composed of CPU and GPU are becoming more and more popular in high performance computing. Workload can be split and distributed to CPU and GPU to utilize them for data parallel computing, so it can improve the overall performance of the hybrid system. This paper introduced the workload partitioning algorithm based on performance curve of GPU which takes advantage of the characteristics of GPU performance varying with the load in off-line analysis stage. In addition, it uses the gradual decreasing method to determine the optimal data allocation proportion between processing units. Moreover, it verifies the effectiveness of the algorithm based on the image median filtering algorithm. The results show that the proposed workload partitioning algorithm can effectively improve the processing efficiency of the data parallel applications.

ACKNOWLEDGMENTS

This work is supported by Shenzhen Science and Technology Foundation Grant No. JSGG20160229123657040 and No.JCYJ20150630114942277.

REFERENCES

[1] Yang. Wangdong, Li. Kenli, and Li. Keqin, "A parallel computing method using blocked format with optimal partitioning for SpMV on GPU," Journal of Computer and System Sciences, vol. 92, pp. 152-170, March 2018.

[2] Howison. Mark, and E. W. Bethel, "GPU-accelerated denoising of 3D magnetic resonance images," Journal of Real-Time Image Processing ,vol. 13, pp. 713–724, december 2017.

[3] The OpenCL Specification V2.0. [Online]. Available: http://www.khronos.org.

[4] CUDA C Bes t Practices Guide V7.0. [Online]. Available: http:// docs.nvidia.com, NVIDIA.

[5] OpenMP Application Program Interface V4.5. [Online]. Available: http://www.openmp.org,OpenMP ARB.

[6] Shen. Jie, A. L. Varbanescu, and H. Sips, "Look before You Leap: Using the Right Hardware Reso-urces to Accelerate Applications," in High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS). 2014 IEEE Intl Conf on. IEEE, 2015, pp. 383-391.

[7] Wikipedia, “median filtering,”. [Online]. Available: https://en.wikipedia.org/wiki/Median_flter.

[8] Luk. Chi Keung, S. Hong, and H. Kim, "Qilin: Exploiting parallelism on heterogeneous multiprocessors with adaptive mapping," in Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2009, pp. 45-55.

[9] Zhong. Ziming, V. Rychkov, and A. Lastovetsky, "Data Partitioning on Multicore and Multi-GPU Platforms Using Functional Performance Models," IEEE Transactions on Computers, vol. 64, pp. 2506-2518, September 2015.

[10] Huang. Wen, Yu. Licheng, Ye. Mingjiao, Chen. Tianzhou, and Hu. Tongsen, "A CPU-GPGPU Scheduler Based on Data Transmission Bandwidth of Workload," in International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). 2012 13th International Conference on. IEEE, 2012, pp. 610-613.

[11] S. Daniel Nemirovsky, Tugberk. Arkose, Nikola. Markovic, Mario Nemirovsky, Osman. Unsal, and Adrian. Cristal, "A Machine Learning Approach for Performance Prediction and Scheduling on Heterogeneous CPUs," in Computer Architecture and High Performance Computing (SBAC-PAD). 2017 29th International Symposium on. IEEE, 2017, pp. 121-128.

[12] Shen. Jie, Varbanescu. Ana luck, Lu. YuTong, Zou. Peng, and Sips. Henk, "Workload Partitioning for Accelerating Applications on Heterogeneous Platforms," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 2766 - 2780, September 2016.

[13] Tse. Anson. H. T, Thmoas. David B, and Tsoi. K. H, Luk. Wayne,"Dynamic scheduling Monte-Carlo framework for multi-accelerator heterogeneous clusters," in International Conference on Field-Programmable Technology (FPT). 2010 International Conference on. IEEE, 2010, pp. 233-240.

[14] Belviranli. Mehmet E, L. N. Bhuyan, and R. Gupta, "A dynamic self-scheduling scheme for heterogeneous multiprocessor archi-tectures," Acm Transactions on Architecture and Code Optimiz-ation(TACO), vol. 9, pp. 1-20, January 2013.

[15] Wang. Zhenning, Zheng. Long, Chen. Quan, and Guo. Miniyi, "CPU + GPU scheduling with asymptotic profiling," Parallel Computing, vol. 40, pp. 107-115, February 2014.

[16] The OpenCV Specification V2.4.8. [Online]. Available: https://opencv.org/.

[17] Geospatial Data Cloud site, Computer Network Information Center, Chinese Academy of Sciences. [Online]. Available: http://www.gscloud.cn.