ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.11, No.22, 1–8

CPU-GPU hybrid computing for feature extraction from video stream

Sungju Lee[1], **Heegon Kim**[1], **Daihee Park**[1], **Yongwha Chung**[1a)], **and Taikyeong Jeong**[2b)]

[1] *Dept. of Computer and Information Science, Korea University, Sejong, Korea*

[2] *Dept. of Computer Science and Engineering, Seoul Women's University, Korea*

a) *ychungy@korea.ac.kr*

b) *ttjeong@swu.ac.kr*

Abstract: In this paper, we propose a way to distribute the video analytics workload into both the CPU and GPU, with a performance prediction model including characteristics of feature extraction from the video stream data. That is, we estimate the total execution time of a CPU-GPU hybrid computing system with the performance prediction model, and determine the optimal workload ratio and how to use the CPU cores for the given workload. Based on experimental results, we confirm that our proposed method can improve the speedups of three typical workload distributions: CPU-only, GPU-only, or CPU-GPU hybrid computing with a 50:50 workload ratio.

Keywords: CPU, GPU, heterogeneous computing, feature extraction

Classification: Integrated circuits

References

[1] D. Tian: International Journal of Multimedia and Ubiquitous Engineering **8** [4] (2013) 385.

[2] M. Krulis, J. Lokoc and T. Skopal: MMM LNCS **7733** (2013) 446.

[3] S. Ohshima, K. Kise, T. Katagiri and T. Yuba: VECPAR LNCS **4395** (2007) 305.

[4] C. Lee, W. Ro and J. Gaudiot: IEEE INTERACT (2012) 33.

[5] X. Lu, B. Han, M. Hon, C. Xiong and Z. Xu: Adv. Eng. Softw. **70** (2014) 90. DOI:10.1016/j.advengsoft.2014.01.010

[6] W. Sodsong, J. Hong, S. Chung, Y. Lim, S. Kim and B. Burgstaller: ACM PMAM (2014) 80.

[7] L. Wan, K. Li, J. Liu and K. Li: ACM PMAM (2014) 70.

[8] B. Chapman, G. Jost and R. Van: Cambridge MA MIT Press (2008).

[9] B. Gaster, L. Howes, D. Kaeli, P. Mistry and D. Schaa: Revised OpenCL 1.2 Edition Morgan Kaufmann (2012).

1 Introduction

High-performance video analytics is an important challenge for big data processing. Feature extraction in video analytics is an important method to identify and recognize real-world objects. Its purpose is to encode simple geometrical forms

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014

such as straight lines in different directions [1]. Many computer vision applications such as feature extraction, object tracking, and recognition have been parallelized by using multicore platforms such as a CPU or GPU in order to reduce execution time [2].

Typically, when an application is running with a GPU system, only one thread is assigned to a "host CPU core" (*i.e.*, a core of CPU for executing a host program) in order to control a GPU and manage data copy operations; remaining CPU cores are in an idle state while the GPU performs the application-specific tasks. This leads to large amounts of wasted CPU resources. Therefore, it is important to find an effective method to make full use of all the available computational resources of both the CPU and GPU. Recently, some approaches [3, 4, 5, 6, 7] have been developed to perform a specific task using both multi-core CPU and GPU simultaneously, instead of the CPU or GPU alone. In this paper, we present a way to distribute the workload into both the CPU and GPU, with a performance prediction model (*i.e.*, a static strategy) including characteristics of feature extraction from the video stream data.

In order to parallelize feature extraction and determine the optimal workload distribution into both the CPU and GPU, we consider two important characteristics of feature extraction. First, feature extraction is affected by non-negligible parallel overhead such as data copy time, since feature extraction for a single image frame has relatively short execution time. However, feature extraction used for object tracking requires a large number of frames and has data dependency in each frame. Thus, we should parallelize the feature extraction of one image frame for satisfying the real-time requirement of video stream processing. In addition, since the video streaming data are continuous within the static workload, the static strategy can be efficient to distribute the workload into both the CPU and GPU for feature extraction without complicated scheduling overhead.

In this paper, for a given machine (*i.e.*, a multi-core CPU and GPU combination), we derive a performance prediction model of the given machine by using linear interpolation, and determine the optimal CPU:GPU workload ratio by using the performance prediction model. Also, we improve the performance of a CPU-GPU hybrid computing by reducing the idle time of CPU and GPU. Based on our experimental results, we confirm that the proposed method using the optimal workload distribution can perform feature extraction more efficiently than CPU-only, GPU-only, or CPU-GPU hybrid computing with a 50:50 workload ratio.

The rest of the paper is structured as follows: Section 2 describes computational characteristics of feature extraction, and Section 3 describes the CPU-GPU hybrid computing with optimal workload distribution. The experimental results are given in Section 4, and conclusions are provided in Section 5.

2 Computational characteristics of feature extraction

Feature extraction consists of the operations *SmoothedImage*, *Gradients*, and *TrackabilityEachPixel*, and has two important computational characteristics. First, feature extraction is affected by parallel overhead such as data copy time, which is non-negligible. Fig. 1 illustrates parallelized feature extraction using CPU-GPU

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014

hybrid computing. Typically, a single image frame at the host CPU core is divided in two parts and assigned to the "remaining CPU cores" (*i.e.*, the remaining cores of CPU for computing a workload) and GPU, respectively. Although the GPU requires data copy time in order to compute the workload, it should be noted that a GPU can provide better performance than a multicore CPU for many applications having no data dependency. While a GPU requires data copy time from the host CPU core, the data copy time is negligible in the large-scale, compute-bound problems in scientific applications. However, in the feature extraction, the parallel overhead sensitively affects the total execution time of feature extraction, since each operation of feature extraction (*i.e.*, *SmoothedImage*, *Gradients*, and *TrackabilityEachPixel*) has relatively short execution time (*i.e.*, tens of millisecond) for one image frame.

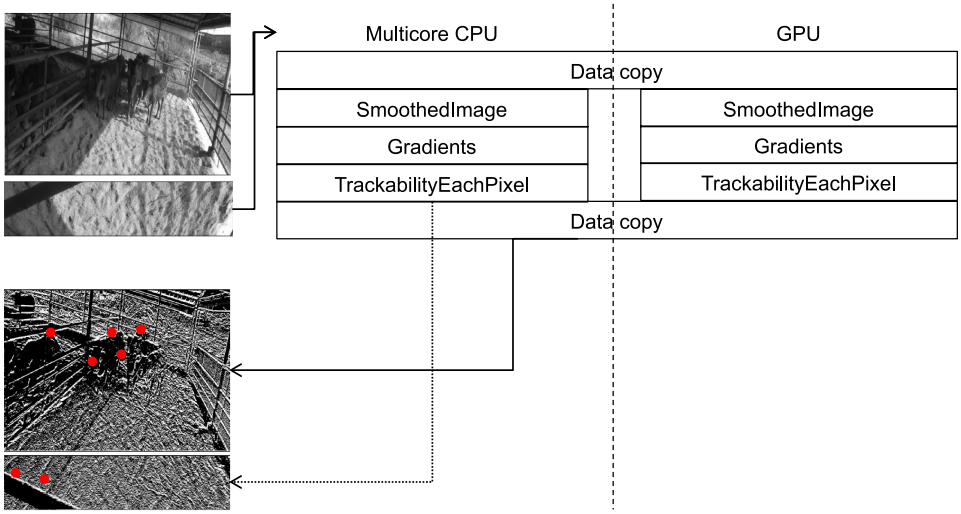

Fig. 1. An example of computational characteristics of feature extraction.

For example, Fig. 2 shows the computational characteristics of CPU (with four cores) and GPU hybrid computing, and the effect of data copy time. In general, the data copy time is negligible in the large-scale, compute-bound, scientific problems as shown in Fig. 2(a). However, if the data copy time is non-negligible and amount of the host CPU core's computation and data copy time is much longer than the computation time of the remaining CPU cores as shown in Fig. 2(b), it significantly affects the total execution time. In this case, since the data copy time is relatively large, the remaining CPU cores have a long idle time. To reduce this idle time, we can distribute the host CPU core's workload into the remaining CPU cores as shown in Fig. 2(c). Since three of the CPU's cores (*i.e.*, remaining CPU cores) are used for additional workload execution, the host CPU core (*i.e.*, core 4) can perform the data copy without any workload execution. Ideally, the workload of the remaining CPU cores can be redistributed into the host CPU core (See Fig. 2(d)). However, the performance improvement from Fig. 2(c) to Fig. 2(d) may be much smaller than the performance improvement from Fig. 2(b) to Fig. 2(c).

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014

Further, since feature extraction is performed with continuous video stream data, a static strategy can efficiently distribute the workload into both the CPU and GPU without complicated scheduling overhead. In this paper, we apply the performance prediction model by using linear interpolation based on a pre-experiment test (*i.e.*, for a given CPU-GPU hybrid computing system, we measure the execution time for feature extraction of an image frame size), and then determine the optimal CPU:GPU workload ratios for other image frame sizes.

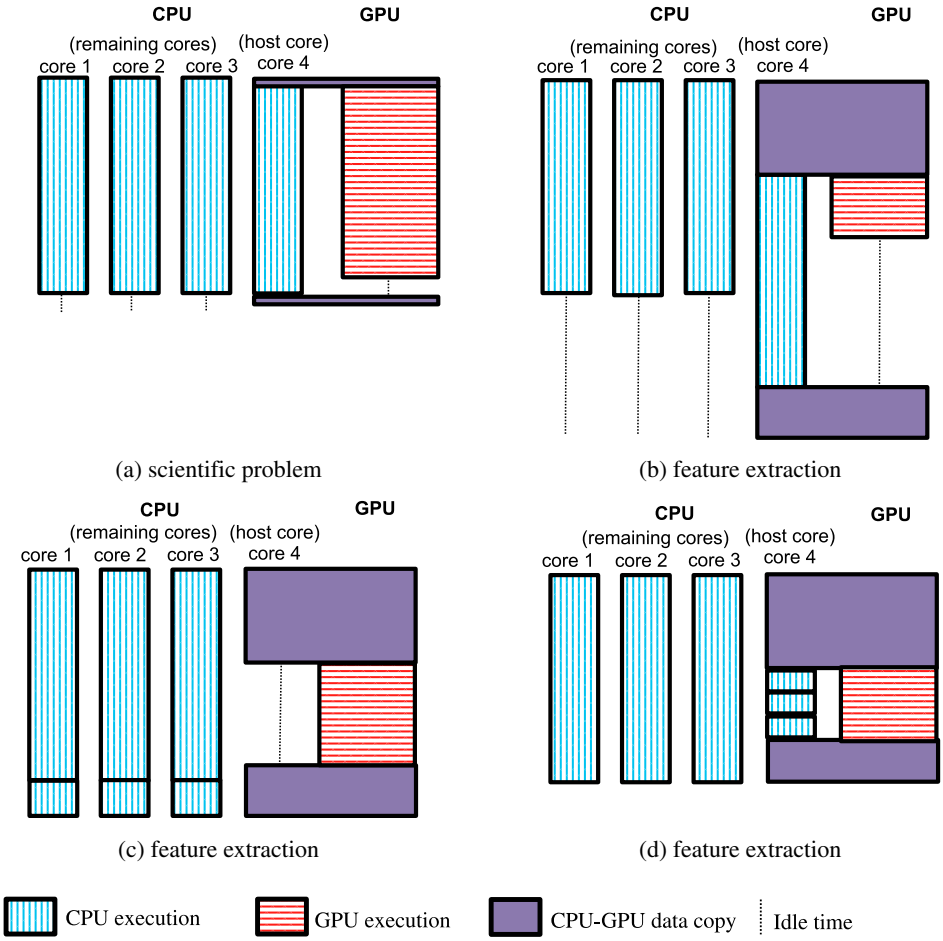

Fig. 2. An example of the computational characteristics of CPU-GPU hybrid computing, showing the effect of data copy time.

3 Proposed method

3.1 Determine the optimal CPU:GPU workload ratio

In this paper, we predict the execution time by using linear interpolation based on a pre-experiment test, and then determine the optimal CPU:GPU workload ratio. To accurately predict the execution time of feature extraction, we analyze the computational characteristics of the CPU and GPU, including data copy time to and from the GPU. First, the execution time is measured with various CPU:GPU workload ratios for a given CPU-GPU hybrid computing system, and then the performance prediction model can be derived by using linear interpolation. For example, we measure the execution time with a 640×480 image, and then the performance

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014

prediction model (*i.e.*, linear equations) is derived by using linear interpolation. These equations can predict the execution time for the feature extraction of various image frame sizes. Also, we can determine the optimal CPU:GPU workload ratio for feature extraction of various image frame sizes by using the performance prediction model. The method to find the optimal CPU:GPU workload ratio is as follows:

The total workload (*i.e.*, the size of a given image frame) W_{TOTAL} is represented with W_{CPU} and W_{GPU}, as shown in equation (1).

$$W_{TOTAL} = W_{CPU} + W_{GPU} \tag{1}$$

The total execution time of hybrid computing $T_{CPU+GPU}(W_{CPU} + W_{GPU})$ is represented by the maximum execution time of $T_{CPU}(W_{CPU})$ or $T_{GPU}(W_{GPU})$ as shown in equation (2).

$$T_{CPU+GPU}(W_{CPU} + W_{GPU}) = \max[T_{CPU}(W_{CPU}),\ T_{GPU}(W_{GPU})] \tag{2}$$

Since the execution time for feature extraction of CPU and GPU increases or decreases linearly according to workload size, we can represent $T_{CPU}(W_{CPU})$ and $T_{GPU}(W_{GPU})$ with equation (3) and equation (4) by using linear interpolation. To obtain the parameters α, β, γ, and δ (*i.e.*, coefficients and constants of linear equations), we should measure the execution time for feature extraction of an image frame size on CPU and GPU at least once, and then the performance prediction model (*i.e.*, linear equations) is derived with the parameters. Note that, to accurately predict the execution time, the GPU's execution time should include the data copy time.

$$T_{CPU}(W_{CPU}) = \alpha W_{CPU} + \beta \tag{3}$$

$$T_{GPU}(W_{GPU}) = \gamma W_{GPU} + \delta \tag{4}$$

The performance of hybrid computing $T_{CPU+GPU}(W_{CPU} + W_{GPU})$ is maximized, when $T_{CPU}(W_{CPU})$ is approximately equal to $T_{GPU}(W_{GPU})$. Therefore, using above equation (3) and equation (4), we can represent the $T_{CPU}(W_{CPU}) = T_{GPU}(W_{GPU})$ as equation (5).

$$\alpha W_{TOTAL} - \alpha W_{GPU} + \beta - (\gamma W_{GPU} + \delta) = 0 \tag{5}$$

Finally, the optimal workload OW_{CPU} and OW_{GPU} are represented with equation (6), and we can determine the optimal CPU:GPU workload ratio.

$$OW_{GPU} = \frac{\alpha W_{TOTAL} + \beta - \delta}{(\alpha + \gamma)}$$

$$OW_{CPU} = W_{TOTAL} - OW_{GPU} \tag{6}$$

3.2 Determine how to use the CPU cores

To reduce the idle time of CPU and GPU, Algorithm 1 shows the workload distribution method. First, the performance prediction model is derived, and the optimal workload ratio of CPU:GPU is determined as shown in step 1 and step 2 (See Fig. 2(a) or (b)). In step 3, we determine how to use the CPU cores. If amount of the host CPU core's computation and data copy time is much longer than the remaining CPU cores' computation time, OW_{CPU} and OW_{GPU} should be updated

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014

by using equation (3), (5), and (6) (See Fig. 2(c)). Finally, some workload of the remaining CPU cores is redistributed into the host CPU core for the ideal workload distribution (See Fig. 2(d)). That is, we treat the host CPU core managing the non-negligible data copy time differently from the remaining CPU cores. On the contrast, if the data copy time is negligible (*i.e.*, step 3-2), CPU-GPU hybrid computing is executed with OW_{CPU} and OW_{GPU} computed by step 2.

Algorithm 1: workload distribution into CPU and GPU
Step 1: Construct the each performance model for CPU and GPU
1-1: Use remaining CPU cores and host CPU core for CPU computation (with n cores)
1-2: Obtain parameters α, β, γ, and δ
Step 2: Determine OW_{CPU} and OW_{GPU}
Step 3: Determine how to use the CPU cores
3-1: IF $\frac{OW_{CPU}}{n}$ + data copy time > $\frac{OW_{CPU}}{n-1}$
Use remaining CPU cores for CPU computation (with $n-1$ cores)
Update performance prediction model for CPU by using equations (3), (5), and (6)
Update OW_{CPU} and OW_{GPU}
Assign $\frac{OW_{CPU}}{n-1} - T_{GPU}(W_{GPU})$ to the host CPU core
Execute CPU-GPU hybrid computation with OW_{CPU} and OW_{GPU}
3-2: ELSE
Use remaining CPU cores and host CPU core for CPU computation (with n cores)
Execute CPU-GPU hybrid computation with OW_{CPU} and OW_{GPU}

4 Experimental results

To evaluate the proposed method, we conducted the experiment with an Intel Core i5-3570 CPU (having 4 cores) and a GeForce GTX 660 GPU (having 960 cores). We used 120 image frames, and the image frame sizes were 1440×1080, 1280×720, 800×600, 720×480 and 640×480. In addition, we used OpenMP [8] and OpenCL [9] in order to parallelize feature extraction on the CPU and GPU. Note that, although OpenMP does not require the data copy time within a CPU, it does not provide the detail operations for asymmetric workload assignment between the host CPU core and the remaining CPU cores. Also, the performance improvement from Fig. 2(c) to Fig. 2(d) may be much smaller than the performance improvement from Fig. 2(b) to Fig. 2(c). Thus, we distributed the workload into both the CPU and GPU such as Fig. 2(c).

To obtain the parameters for the linear equations, we measured the execution time for feature extraction of CPU and GPU with a 640×480 image frame size, and the parameters used for linear equations of CPU (*i.e.*, α and β) and GPU (*i.e.*, γ and δ) are summarized in Table I.

Table I. Parameters for performance prediction model of feature extraction

For CPU		For GPU	
α	β	γ	δ
2.78×10^{-4}	1.61×10^{-4}	7.25×10^{-5}	5.63×10^{-3}

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014

To validate the performance prediction model, we compared the estimated (*i.e.*, using equations (3) and (4)) and the measured execution times as shown in Fig. 3. The results show that the performance prediction model can be accurate enough to estimate the execution time for the optimal CPU:GPU workload ratio. For example, the optimal CPU:GPU workload ratio was 78%:22% for 1440×1080 image frames. For larger image frames, the GPU's optimal workload size should be decreased in order to provide better performance, due to the increased effect of the data copy time. Note that, since the sequential execution time of feature extraction for 1440×1080 image was 760 ms, we should parallelize the feature extraction for satisfying the real-time requirement of video stream processing. In the case of 1440×1080 image frames, the sequential execution provides 1.3 FPS (*i.e.*, frames per second), whereas the parallel execution provides 16.3 FPS with the proposed CPU-GPU hybrid computing method.

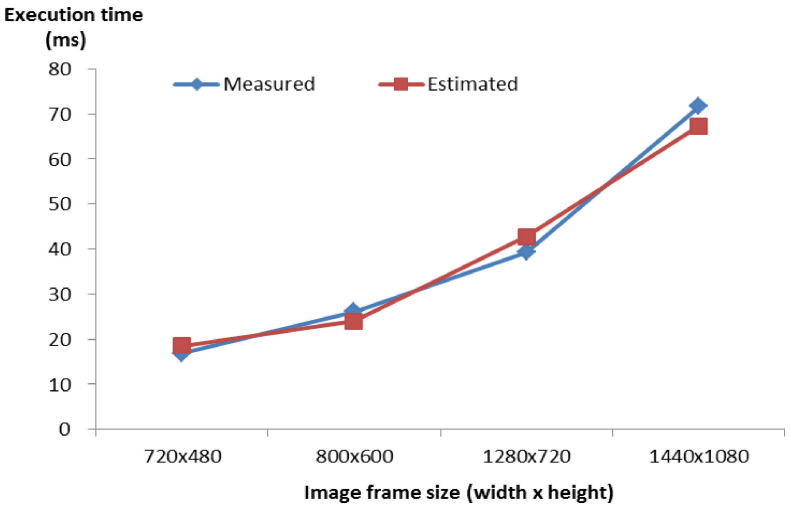

Fig. 3. Comparison of estimated and measured execution times of the proposed CPU-GPU hybrid computing method

To evaluate the proposed method in detail, the performance (*i.e.*, speedup) of the proposed CPU-GPU hybrid computing method is compared with those of three typical workload distributions (*i.e.*, CPU-only, GPU-only, or CPU-GPU hybrid computing with a 50:50 workload ratio). Fig. 4 shows that even if there is data copy time overhead for the GPU, the GPU-only method still provides better performance than the CPU-only method for various image frame sizes. We also considered the performance of the straightforward hybrid method (*i.e.*, using all four of the CPU's cores for workload execution and CPU-GPU computing with a 50:50 workload ratio). In the proposed method, however, we used three of the CPU's cores for workload execution, and the optimal workload ratios were 75:25, 77:23, 78:22, and 78:22 for 720×480, 800×600, 1280×720, and 1440×1080 image frames, respectively. In the results, the straightforward hybrid method provided even worse performance than the GPU-only method. However, we confirmed that the proposed hybrid method can improve the performance of the GPU-only method by up to 23% for 1440×1080 image frames.

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014

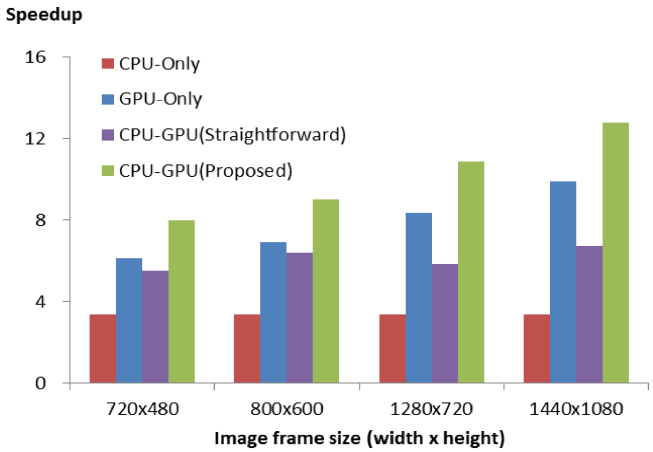

Fig. 4. Performance comparison of workload distributions

5 Conclusions

In this paper, we efficiently parallelized video feature extraction by using the CPU and GPU simultaneously. We estimated the total execution time with a performance prediction model, and determined the optimal CPU:GPU workload ratio between CPU and GPU and how to use the CPU cores for the given workload. To the best of knowledge, this is first report on treating the host CPU core (*i.e.*, managing the non-negligible data copy time) differently from the remaining CPU cores. Based on the experimental results, we confirmed that the proposed method can improve the performance of CPU-only, GPU-only, or CPU-GPU hybrid computing with a 50:50 workload ratio.

Acknowledgments

This research was supported by BK21 Plus Program.
(Corresponding authors: Y. Chung and T. Jeong)

DOI: 10.1587/elex.11.20140932
Received September 29, 2014
Accepted October 17, 2014
Publicized November 4, 2014
Copyedited November 25, 2014