GRAPHICS PROCESSOR EFFICIENCY FOR REALIZATION OF RAPID TABULAR COMPUTATIONS

V. A. Dudnik,* V. I. Kudryavtsev, S. A. Us, M. V. Shestakov

National Science Center "Kharkov Institute of Physics and Technology", 61108, Kharkov, Ukraine

(Received February 19, 2016)

Capabilities of graphics processing units (GPU) and central processing units (CPU) have been investigated for realization of fast-calculation algorithms with the use of tabulated functions. The realization of tabulated functions is exemplified by the GPU/CPU architecture-based processors. Comparison is made between the operating efficiencies of GPU and CPU, employed for tabular calculations at different conditions of use. Recommendations are formulated for the use of graphical and central processors to speed up scientific and engineering computations through the use of tabulated functions.

PACS: 89.80.+h, 89.70.+c, 01.10.Hx

1. INTRODUCTION

The tabular calculations imply preliminary computations of the table (array) of single argument values and the respective function values. This provides a way of obtaining further the function values for specific argument values immediately, without additional calculations. This method is used in cases, where the domain of a function is a discrete finite set. In cases when it is necessary to obtain the function values, which correspond to the intermediate values of argument and which are not tabulated, the interpolation procedure is used. The use of tabular procedures provides in many cases a multiple speedup of calculations without making additional special efforts.

The most popular fields of use of tabular procedures are exemplified by audio-visual signal processing applications, programs for numerical filtering of data coming from digitizers, tomography image formation complexes, digital radiographic systems. All these cases call for provision (with no great extra expense) of data array processing speed higher than that provided by conventional computers.

2. PECULIARITIES OF MEDICAL IMAGE BRIGHTNESS EQUALIZATION

One of the necessary operations in digital imaging is the procedure of image brightness equalization that can improve the diagnostic value of the images. The necessity of this procedure stems from the peculiarity of the optical system employed for image acquisition. To enhance the sensitivity of the optical systems (and, as the final result, to reduce the radiation dose for the patient under examination), large-diameter high-speed lenses are used. The brightness transfer coefficient of this lens has a strongly marked shape of a parabola, i.e., the obtained images are substantially less bright on the periphery than at the center (Fig.1). Below we describe the operational experience of the rapid brightness equalization function required for formation of X-ray medical images.

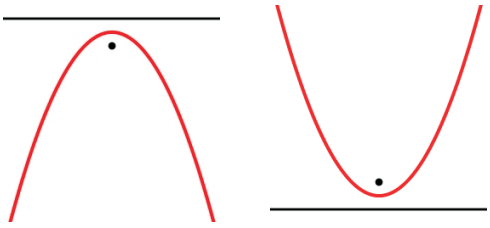

Fig.1. *The brightness transfer coefficient of the lens and the image brightness equalization function*

The general form of the brightness equalization function is:

```
Pixel_New_Light = F(x, y, Current_Light)

where x, y are the coordinates of the

equalized image element (pixel);

Current_Light is the brightness value of the
equalized pixel.
```

In our case, for brightness equalization the following function was used:

```
NewPixelLight[x,y] = A[x,y] * Pixel_Light2[x,y]

+ B[x,y] * Pixel_Light[x,y]+C[x,y];
```

where A, B, C denote the arrays of coefficients (precomputed on the basis of analysis of different-brightness images), which provide compensation of brightness adjusting of the lens.

*Corresponding author. E-mail address: dudnik@mail.ru

At first, the program that fulfilled the brightness equalization by the conventional method was used. In what follows we give a fragment of C-based program, which serves for the brightness equalization:

```
//Brightness correction for all image
elements
for(i=0; i<SizeOfPartLightAlign; i++)//
Image pixel cycle

{

Pixel_Light = A[k] * (double)ImageData[k] *

(double)ImageData[k] + B[k] *

(double)ImageData[k] + C[k];

}

// Image pixel cycle
```

Rather time-consuming operations (i.e., multiplication, conversions from integer-to-floating point format and vice versa) employed in the given program fragment, failed to provide the specified time of image framing. Therefore, an attempt was undertaken to realize this function by means of tabular procedures.

Below, we give the C-based program fragment, which fulfils the brightness equalization through the tabular procedure:

```
// Table function of brightness
correction for all image elements

for(i=0; i<SizeOfPartLightAlign; i++).

// Image pixel cycle

{

AlignImage[i] =[Image[i]];

// Image pixel cycle

}

// Image pixel cycle
```

3. CPU-BASED EQUALIZATION BY THE TABLE METHOD

The use of table methods for conventional CPU has revealed a strong dependence of the execution time on the table size (Figs. 2 and 3). This made the use of table methods for conventional CPU impossible, considering that beginning approximately with the table size of 28 MB, the table methods worked slower than the standard computations procedures. Still, the formation of obtained work images called for the tables of the size of about 600 MB, and with these table sizes the table methods appeared to work slower than the usual computations. So, for image formation we had to resort to using a powerful computer with 8 computation CPU cores) (note that in spite of that, only a 6-fold speedup was attained). Figs. 2 and 3, given below, show the brightness adjusting time versus the size of the table of values of the function.

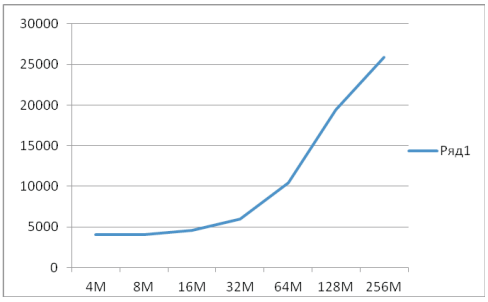

Fig.2. *Image brightness adjusting time versus the size of the table of values of the equalization function for the CPU*

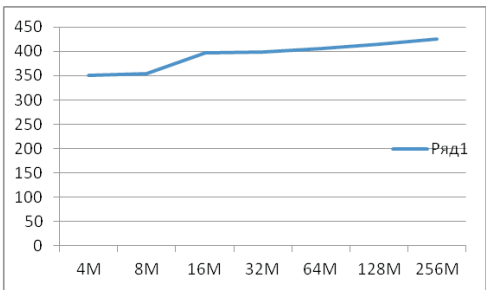

Fig.3. *Image brightness adjusting time versus the size of the table of values of the equalization function for the GPU*

4. EQUALIZATION BY THE CPU-BASED TABLE METHOD

A similar tabulated function of brightness equalization was realized with the use of the GPU. Fig. 4 shows the dependence of the function-execution time on the size of the table of values for the GPU. It is obvious from the figure that the equalization function execution time depends only slightly on the size of the table of function values.

5. ANALYSIS OF POSSIBLE CAUSES OF TABLE METHOD WORK SLOWDOWN

The slowdown is most likely due to some peculiarities in the operation of the page-segment memory addressing, which is used in the CPU architecture. The computation processes in the CPU are performed in the dedicated memory spaces. The virtual space of the process memory area is divided into segments, and in turn, each segment is divided into virtual pages numbered within the segment. The random-access memory is mapped into physical pages. The structure of memory addressing is shown in Figs. 4 and 5. For speed up the addressing process, the CPU

Intel architecture makes use of translation lookaside buffers and the data buffer (cash). As to our case, all the pages with function values resided in the random-access memory. Therefore, as may be supposed, the slowdown of memory access was determined by the peculiarities of Intel processor architecture and was connected with exhaustion of the translation lookaside buffer store capacity of the CPU.

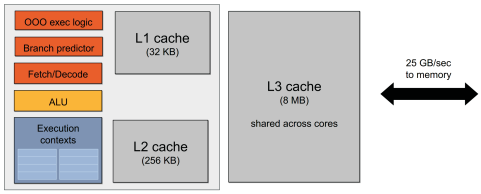

Fig.4. *Structure of memory addressing the CPU*

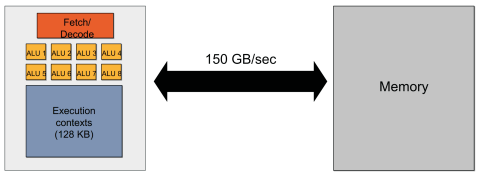

Fig.5. *Structure of direct memory addressing in the GPU architecture*

For the GPU memory access, an easier scheme of direct memory addressing is used. For that reason the table methods for the GPU have appeared more efficient and less dependent on table sizes.

6. CONCLUSIONS

The comparison between the efficiencies of GPU and CPU applications for tabular calculations has shown that unlike the CPU case, with the use of GPU the computation time of tabulated function values is depends only slightly on the data array size. This gives grounds to recommend the use of tabulated functions for performing GPU-based computations of fragments of time-critical scientific and engineering tasks.

References

1. NVIDIA, "Fermi: NVIDIA's Next Generation CUDA Compute Architecture," 2009;

   `http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf.`

2. A. Zubinsky. NVIDIA Cuda: unification of graphics and computations. (In Russian). 8 May, 2007. http://itc.bf/node/27969

3. David Luebke. Graphical processors - not only for graphics. (Russian translation) http://www.osp.ru/os/2007/02/4106864/

4. V.Dudnik, V.Kudryavtsev, T.Sereda, S.Us, M.Shestakov. Using of opportunities of graphic processors for acceleration of scientific and technical calculations //*PAST*. 2009, N3, p.120-123.

5. David Luebke, Greg Humphreys. How GPU Work. IEEE Computer, February, 2007. IEEE Computer Society, 2007

ЭФФЕКТИВНОСТЬ ИСПОЛЬЗОВАНИЯ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ (GPU) ДЛЯ РЕАЛИЗАЦИИ ФУНКЦИЙ БЫСТРЫХ ТАБЛИЧНЫХ ВЫЧИСЛЕНИЙ

В. А. Дудник, В. И. Кудрявцев, С. А. Ус, М. В. Шестаков

Выполнено исследование возможностей использования графических процессоров (GPU) и процессоров обычной архитектуры (CPU) для реализации алгоритмов быстрых вычислений с помощью табличных функций. Приведены примеры реализации табличных функций для процессоров архитектур GPU и CPU. Сделано сравнение эффективности применения GPU и CPU для табличных вычислений в различных условиях использования. Сформулированы рекомендации по использованию графических и обычных процессоров для ускорения научно-технических расчётов за счёт применения табличных функций.

ЕФЕКТИВНІСТЬ ВИКОРИСТАННЯ ГРАФІЧНИХ ПРОЦЕСОРІВ (GPU) ДЛЯ РЕАЛІЗАЦІЇ ФУНКЦІЙ ШВИДКИХ ТАБЛИЧНИХ ОБЧИСЛЕНЬ

В. О. Дуднік, В. І. Кудрявцев, С. О. Ус, М. В. Шестаков

Виконано дослідження можливостей використання графічних процесорів (GPU) і процесорів звичайної архітектури (CPU) для реалізації алгоритмів швидких обчислень за допомогою табличних функцій. Приведені приклади реалізації табличних функцій для процесорів архітектури GPU і CPU. Виконано порівняння ефективності застосування GPU і CPU для табличних обчислень в різних умовах використання. Сформульовані рекомендації по використанню графічних і звичайних процесорів для прискорення науково-технічних розрахунків за рахунок застосування табличних функцій.