Computer Physics Communications 185 (2014) 2558–2565

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Performance and precision of histogram calculation on GPUs: Cosmological analysis as a case study

Miguel Cárdenas-Montes[a,*], Juan José Rodríguez-Vázquez[a], Miguel A. Vega-Rodríguez[b], Ignacio Sevilla-Noarbe[a], Eusebio Sánchez Alvaro[a]

[a] CIEMAT, Department of Fundamental Research, Avda. Complutense 40, 28040, Madrid, Spain

[b] University of Extremadura, ARCO Research Group, Department Technologies of Computers and Communications, Escuela Politécnica, Campus Universitario s/n, 10003, Cáceres, Spain

ARTICLE INFO

Article history:
Received 20 January 2014
Received in revised form
30 May 2014
Accepted 4 June 2014
Available online 11 June 2014

Keywords:
GPU computing
Histogram calculation
Cosmology
Correlation function

ABSTRACT

Histogram calculation is an essential part of many scientific analyses. In Cosmology, histograms are employed intensively in the computation of correlation functions of galaxies, as part of Large Scale Structure studies. Among the most commonly used ones are the two-point, three-point and the shear–shear correlation functions. In these computations, the precision of the calculation of the counts in each bin is a key element for achieving the highest accuracy. In order to accelerate the analysis of increasingly larger datasets, GPU computing is becoming widely employed in this field. However, the recommended histogram calculation procedure becomes less precise when bins become highly populated in these sort of algorithms. In this work, an alternative implementation to correct this problem is proposed and tested. This approach is based on distributing the creation of histograms between the CPU and GPU. The implementation is tested using three cosmological analyses with observational data. The results show an increased performance in terms of accuracy while keeping the same execution time.

1. Introduction

The calculation of correlation functions is a computing-intensive activity which uses histograms as an intrinsic part of the algorithm in many implementations. They are frequently employed as part of Large Scale Structure studies in cosmological analyses. Due to the increment in the volume of scientific data available, cosmologists have researched diverse alternatives to accelerate these computations. Currently, GPU computing has the capability to process large datasets within a reasonable execution time and many publications have tackled the analysis of experimental data employing them. However, for large datasets, histogram construction presents some precision weaknesses associated with number representation.

In this work, these weaknesses are shown in relation to the study of correlation functions in Cosmology: the two-point angular correlation function (2PACF), the three-point angular correlation function (3PACF) and the shear–shear correlation function. However, it is expected that these imprecisions are independent of the problem, and only related with factors such as the dataset size, the histogram parameters and the number representation.

In order to overcome them, an alternative implementation for histogram construction is presented and evaluated using observational data. This approach divides the histogram construction task in two phases. Firstly, the counts are accumulated in sub-histograms which reside on the shared memory of GPU and are collated in global memory. Later, these sub-histograms are added up on the host (CPU) memory (hereafter termed host memory), to profit from the double precision representation. The proposed implementation is as fast and is more accurate than with previous schemes. The tests performed and reported in the current work demonstrate that this new approach is able to deal with up to 10^{12} counts per bin. To the authors' knowledge, no modifications in this line have been proposed to overcome the weaknesses of the number representation in histogram construction.

This paper is organized as follows: Section 2 summarizes related work and previous efforts done in this area. In Section 3.1 a brief description of the weaknesses of number-representation is presented. The relevant algorithms for cosmology analysis used in this work are briefly described in Section 3.2. The CFHTLenS dataset, which is used later, is presented in Section 3.3. The comparison of the new implementation presented here versus the

[*] Corresponding author. Tel.: +34 91 346 6281; fax: +34 91 346 6068.
E-mail address: miguel.cardenas@ciemat.es (M. Cárdenas-Montes).

http://dx.doi.org/10.1016/j.cpc.2014.06.002
0010-4655/© 2014 Elsevier B.V. All rights reserved.

classic approach is presented and analysed with both artificial (Section 4) and real data (Section 5). Finally, Section 6 contains the conclusions of this work.

2. Related work

A standard and essential reference for histogram construction on GPUs is the white paper from NVIDIA [1]. Although originally devoted to image processing and data mining, it has served as a guide for many other scientific areas. In this paper, 64-bin and 256-bin histogram implementations are proposed and evaluated. Nowadays, the hardware has evolved enough to allow for larger histogram sizes. The proposed implementation of [1] is the 'classic' one using per-block[1] sub-histograms in shared memory. In this approach, they are later gathered on global memory to form the final histogram. Two more variants are proposed: per-thread and per-warp[2] sub-histograms. Depending on the architecture of the GPU, the histogram size and the data size, these strategies might be a limiting factor in the performance and in the capability to contain the counts in the bins (bin-count accuracy).[3]

In [2], the authors propose two new efficient methods for histogram calculation on GPUs. This article uses the NVIDIA white paper as a starting point to propose their improvements. However, from the two implementations for histogram construction presented in [1], the authors only cite the smallest one in capability. In any case, this paper was written when compute capability was 1.0,[4] and no atomic operations on shared memory were available. Therefore, it can be considered as an obsolete strategy.

In [3], another efficient implementation to compute image histograms on GPUs is proposed. Among other considerations, the authors make considerations on the precision and the bin-count accuracy. In this paper, an implementation which is unable to accumulate more than 256 counts per bin is compared with a new proposal. In order to mitigate the lack of bin-count accuracy, local histograms are created. However, this new implementation introduces errors when accumulating more than 2048 counts per bin. This is clearly insufficient for cosmological analysis such as: 2PACF, 3PACF and shear–shear correlation, as it will be shown later. Neither of these implementations showcase the use of atomic functions or shared memory to enhance the performance.

In the book [4] a very efficient implementation of per-block sub-histograms on shared memory is described. This implementation presents a few advantages: it is easy to implement, it is widely applicable to many different disciplines and it has a great performance and bin-count accuracy. This implementation seems inspired by the per-block implementation of [1], but incorporates atomic functions and uses shared memory for storing the intermediate sub-histograms therefore improving the performance. This implementation has been used by the authors in previous works, such as the analysis of the 2PACF [5,6] and the shear–shear analysis [7]. At the same time, it is employed for comparison purposes in the current work.

In [8] a per-block sub-histogram implementation is proposed. The novelty of this approach relies on the multiple sub-histograms which are embodied in a single thread block. This approach is an extension of the one sub-histogram per thread block approach [9] which includes multiple sub-histograms per thread block. As a final result, the implementation is a hybrid between per-warp and per-block sub-histogram implementations. Given that this implementation gathers the sub-histograms in the final histogram on device memory, it will face difficulties with the largest number attainable for some number-representations. On the other hand, it will show some imprecision when adding small and large numbers in float-representation.

The comparison between the methods proposed by Shams et al. [2] and by Nugteren et al. [9] shows that the different application areas (data mining and image processing) establish different requirements about the histogram size, larger for data mining than for image processing. The input sizes are also different: 8-bits are enough for image processing, whereas, 32-bits are typical in data mining. This restricts the performance study towards different objectives than in Cosmology, and therefore, it forces to implement modifications.

In [9] two histogram implementations are proposed: a per-warp sub-histogram and a per-thread sub-histogram. When processing images, a successful and simple proposition to improve the performance is to shuffle the input data. This is useful for images, however, it has no impact on the analysis of galaxy catalogues used in cosmology. For images, it is probable that close pixels will feed the same bin therefore generating collisions (sequential updates of the number of counts in the same bin) which will degrade the performance. For cosmological inputs, this a priori knowledge of the data structure does not exist, except for the case that the data has been previously ordered.

Neither of the mentioned approaches proposes simultaneously modifying both: the kernel *and* the CPU-part of the code, nor profit from double-representation accessible only on the CPU-part. Furthermore, in all of the previous implementations, after constructing the sub-histograms they are accumulated into the final histogram on the GPU (device memory). In the implementation proposed here, the final gathering is performed on host memory (RAM) which benefits from the double-representation to improve the bin-count accuracy. In addition, no considerations about the input size and how it impacts the precision of the final result are presented in the works mentioned in this section.

In contrast to image processing, cosmological analysis requires trigonometric calculations before feeding the appropriate bin in the histogram. For this reason, performance comparison between the mentioned works and the current work for cosmological problems is not feasible.

Regarding the precision of floating point representation on GPU, a review of this issue is presented at [10]. In this work the inexactnesses associated with the use of float representation (operation accuracy and rounding), and how the programming affects the final result is underlined. Furthermore, in [11] a complete description of the floating point format and its weaknesses are fully described. In [12] the latest version of the floating-point standard can be found.

Concerning other cosmological studies, in [13] the authors present an alternative approach to calculate the 2PACF. In the description of the implementation, the use of integer-representation for the histogram (256 bins and logarithmic binning), as well as the use of *atomicAdd()* function and shared memory for supporting the sub-histograms are highlighted. Although they express that the final aim is to be able to process up to billions of galaxies with this code, the largest dataset processed is composed of one million galaxies. From the description of the implementation, similar difficulties for the bin-count accuracy as the float-based implementation will arise when increasing the size of the dataset. The lack of precision in this implementation is mitigated because the data are processed in bunches, and then accumulated.

Another implementation of the 2PACF is presented in [14]. This approach uses an array in double-representation to hold

[1] A block of threads, thread block or simply block is a logical group of threads which are executed on a streaming multiprocessor.

[2] A warp is a group of 32 threads. The instructions are issued per warp. This is the minimum unit of scheduling in the streaming multiprocessor.

[3] The ability to correctly gather the number of counts assigned to a bin is termed bin-count accuracy.

[4] The compute capability categorizes the features of the compute device. A higher number indicates more advanced features and a newer generation of the device.