Optimizing Memory Usage and Accesses on CUDA-Based Recurrent Pattern Matching Image Compression

Patricio Domingues[1], João Silva[1], Tiago Ribeiro[1], Nuno M.M. Rodrigues[1,2], Murilo B. De Carvalho[3], and Sérgio M.M. De Faria[1,2]

[1] School of Management and Technology,
Polytechnic Institute of Leiria,
Leiria, Portugal
[2] Instituto de Telecomunicações, Portugal
[3] TET/CTC, Universidade Federal Fluminense,
Niterói - RJ, Brazil

Abstract. This paper reports the adaptation of the Multidimensional Multiscale Parser (MMP) algorithm to CUDA. Specifically, we focus on memory optimization issues, such as the layout of data structures in memory, the type of GPU memory – shared, constant and global – and on achieving coalesced accesses. MMP is a demanding lossy compression algorithm for images. For example, MMP requires nearly 9000 seconds to encode the 512×512 Lenna image on a 2013's Intel Xeon. One of the main challenges to adapt MMP to manycore is related to the dependency over a pattern codebook which is built during the execution. This forces the input image to be processed sequentially. Nonetheless, CUDA-MMP achieves a $12\times$ speedup over the sequential version when ran on an NVIDIA GTX 680. By further optimizing memory operations, the speedup is pushed to $17.1\times$.

Keywords: CUDA, memory optimization, manycore computing, image compression.

1 Introduction

Creation and usage of multimedia content is growing exponentially [1]. On one hand, the exploding growth of cheap commodity digital capture devices such as recorders, cameras, and more recently smartphones, tablets and other mobile gadgets enable not only professionals but also individuals to record and produce massive amounts of multimedia material. On the other hand, multimedia devices and standards are evolving toward delivering higher quality, namely higher definition. Indeed, some of the newest smartphones are able to capture video in so-called Full HD (1080p), with some high end models even delivering so-called 4K video [2]. These applications impose high demands on storage and bandwidth. For instance, streaming bandwidth for video content is 8 Mb/s for the standard television format (SD), 19.3 Mb/s for HD and 195 Mb/s for Ultra-HDTV (4K),

B. Murgante et al. (Eds.): ICCSA 2014, Part IV, LNCS 8582, pp. 560–575, 2014.

this last one considering the H.265 format [3]. All this contributes to a massive increase in the size of multimedia content, stressing not only storage but also networks, used to transfer content. Compression mechanisms targeting multimedia content are thus essential to reduce storage and bandwidth demands to a manageable level. Examples include lossless mechanisms such as the Portable Network Graphic (PNG) and JPEG XR formats for images, Windows Media Lossless (WMA Lossless) and Free Lossless Audio Codec (FLAC) for sounds, just to name a few. Lossy methodologies, which sacrifice part of the digital data to achieve higher compression ratios, have an important contribute to reduce the demands of multimedia content. Examples includes MP3 and Ogg Vorbis for sound, JPEG for images and MPEG-4 for video. Even so, current networks struggle to deliver HD content, let alone 4K content [3].

The Multidimensional Multiscale Parser (MMP) is a lossy pattern-matching-based algorithm for the compression of multimedia content, namely images [4,5]. It relies on a dynamically generated set of codebook patterns, which are used to approximate each block of the input image. It uses a multiscale approach, which enables the approximation of image blocks with different sizes.

MMP achieves high compression/quality ratio as measured through peak signal to noise ratio (PSNR), surpassing standards such as H.264 and JPEG2000 [6]. However, the compression/quality ratio achieved by MMP requires massive single precision floating point computation to deal with the pattern matching process. Indeed, MMP is computationally very complex, with the sequential CPU version requiring lengthy execution times when compressing images. For instance, the 512×512 Lenna 8-bit gray image requires more than two hours of wall clock time on a state of the art 2013's Intel Xeon machine.

MMP resorts to a dynamic codebook of blocks that are used to replace the original blocks of the image to compress. In fact, for each original block of the input image, many operations are performed in order to identify and if needed, to create, the replacement block(s) that yields the lowest distortion. Moreover, the best fitted blocks are added to the codebook after the encoding operation of each input block. The use of an adaptive codebook constitutes an important challenge to the parallelization of MMP and henceforth to successfully port the algorithm to parallel accelerators such as Graphical Processing Units (GPUs). In this work, we analyze the techniques used to achieve meaningful speedups over commodity GPUs resorting to the CUDA framework. We essentially focus on the memory optimization issues.

GPUs have revolutionized high performance computing, with affordable commodity hardware hosting thousands of execution cores, yielding massive speedups for certain types of applications. For instance, Lee et. al [7] report that a commodity GPU is, on average, 14 times faster than a state of the art 6-core CPU over a set of CPU- and GPU-optimized kernels. However, important adaptations need to be made to the CPU-based applications to achieve proper speedup on GPUs. The popularity of GPUs within the gamer community brings large scale volumes that not only help to reduce prices but also push performance forward by way of intense competition among manufacturers of GPUs, namely AMD and NVIDIA [8].