Remote Sensing of Environment xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Remote Sensing of Environment

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

Near-real-time focusing of ENVISAT ASAR Stripmap and Sentinel-1 TOPS imagery exploiting OpenCL GPGPU technology

Achille Peternier, John Peter Merryman Boncori *, Paolo Pasquali

Sarmap SA, Purasca, Switzerland

ARTICLE INFO

Article history:
Received 1 July 2016
Received in revised form 13 February 2017
Accepted 11 April 2017
Available online xxxx

Keywords:
Synthetic Aperture Radar
Image focusing
GPGPU
OpenCL

ABSTRACT

This paper describes a SAR image focuser application exploiting General-purpose Computing On Graphics Processing Units (GPGPU), developed within the European Space Agency (ESA) funded SARIPA project. Instead of relying on distributed technologies, such as clustering or High-performance Computing (HPC), the SARIPA processor is designed to run on a single computer equipped with multiple GPUs. To exploit the computational power of the latter, while retaining a high level of hardware portability, SARIPA is written using the Open Computing Language (OpenCL) framework rather than the more widespread Compute Unified Device Architecture (CUDA). This allows the application to exploit both GPUs and CPUs without requiring any code modification or duplication. A further level of optimization is achieved thanks to a software architecture, which mimics a distributed computing environment, although implemented on a single machine. SARIPA's performance is demonstrated on ENVISAT ASAR Stripmap imagery, for which a real-time performance of 8.5 s is achieved, and on Sentinel-1 Interferometric Wideswath (IW) raw data products, for which a near-real time processing time of about 1 min is required. Such a performance has the potential of significantly reducing the storage requirements for wide-area monitoring applications, by avoiding the need of maintaining large permanent archives of Level 1 (focused) imagery, in favor of lighter Level 0 (raw) products, which can be focused on-the-fly within the user's application processing pipelines at almost no overhead.

1. Introduction

With the successful launch in the last two years of the first four Sentinel satellites, two of which (Sentinel-1A and 1B) carrying a Synthetic Aperture Radar (SAR) payload, users and service providers in the remote sensing field are more and more faced with Big Data handling problems. The Sentinel routine acquisition plans, their free and open data policy, and the commitment of the European Copernicus Programme to ensure mission continuity up to 2030 is unprecedented and has already significantly increased the number of Earth Observation (EO) data users. This increased data availability, combined with the maturity level of several data processing techniques, which have been developed since the early '90s, has the potential of boosting research activities and commercial services based on satellite data, and also represents a prerequisite for the development of cost-effective, and potentially also near-real-time, monitoring services.

On the other hand, it is well established that new technological solutions are required to handle Big Data. An effective approach for some users and applications is to run computationally-intensive algorithms on supercomputers or distributed computing systems, which consist of a large number of physical machines (worker nodes), located in central processing facilities, such as the ESA G-POD environment (e.g., De Luca et al., 2015), or virtualized through cloud computing (e.g., Zinno et al., 2015). At the same time, several research institutions and SAR-data service providers currently rely on small to medium size in-house processing facilities (e.g., local clusters or just powerful workstations). Both scenarios benefit from solutions which improve the degree to which the computational resources of single machines are exploited. In particular General-purpose computing on Graphics Processing Units (GPGPU) has received lots of attention in recent years, since potentially it can provide access to massively-parallel processing capabilities (up to several thousands of cores) on single workstations, enabling these to perform as "personal supercomputers". Furthermore, the availability of frameworks including programming APIs for standard languages (like C/C++) allows applications with a high level of portability across different hardware platforms (ranging from laptops to multiple-GPU servers) to be built, although an effort is required in terms of software architecture design and code refactoring.

This paper describes a performance-optimized SAR image focusing software, developed from scratch within the ESA SARscape Image Processor Accelerator project (SARIPA), and which is not related to the operational processor used by ESA for the generation of Level 1 SAR

* Corresponding author at: Sarmap SA, Via Cascine di Barico 10, 6989 Purasca, Switzerland.
E-mail address: jmerryman@sarmap.ch (J.P. Merryman Boncori).

http://dx.doi.org/10.1016/j.rse.2017.04.006
0034-4257/

products. The main goal of SARIPA is to explore the potential of GPGPU technology to reduce processing time on a single machine, thus tackling the data deluge problem by providing a faster and portable local processing, requiring less computational resources compared to High-performance Computing (HPC) and cloud computing. ENVISAT ASAR Image Mode (IM) and Sentinel-1 Interferometric Wideswath (IW) raw images are used as a test-case, with the goal of achieving near-real-time focusing performance on a single computer with multiple GPUs.

Near-real-time SAR image focusing has been the topic of several recent papers, which have addressed the exploitation of multi-core CPUs (Imperatore et al., 2016), or GPUs, using the Compute Unified Device Architecture technology (NVIDIA, 2016a) proprietary to NVIDIA (Zhang et al., 2016; Trittico et al., 2014; Di Bisceglie et al., 2010). SARIPA is based on the more portable Open Computing Language technology (OpenCL, 2016), which unlike CUDA allows the generated software to be executed on a wider range of devices, including computers with non-NVIDIA GPUs or without any GPU at all.

This paper shares the experience gathered through the design and development of SARIPA, whose outcome is not limited to SAR focusing, but could be reused for several other computationally intensive SAR processing algorithms (e.g., interferometry, measurement of deformation time series). The strengths and weaknesses of basing SARIPA on the more liberal OpenCL are discussed in Section 2, whereas the software architecture is described in Section 3. Section 4 details the processor's performance and its potential impact on the Big Data problem in a concrete application scenario. Discussions and conclusions are provided in Sections 5 and 6 respectively.

2. Frameworks for multi-core CPU and GPU exploitation

Several frameworks have emerged in recent years to leverage the computational power provided by modern GPUs. These range from dedicated libraries and Software Development Kits (SDKs) to extensions directly embedded into compilers such as Microsoft's C++AMP and OpenACC. Concerning GPU computing, the two most widespread frameworks are the Compute Unified Device Architecture (NVIDIA, 2016a), and the Open Computing Language (OpenCL, 2016). CUDA and OpenCL are also frameworks that provide developers with the finest control over code implementation and performance, unlike C++AMP and OpenACC, which focus on making the GPU-side aspects as transparent and automatic as possible through high-level abstractions (Hoshino et al., 2013).

CUDA is an NVIDIA proprietary parallel computing platform and programming model, supporting various languages (e.g., C, C++, FORTRAN), and providing optimized libraries for standard mathematical algorithms, like Basic Linear Algebra Subprograms (BLAS) and the Fast Fourier Transform (FFT). CUDA only works on GPUs that are produced by NVIDIA, which on one hand limits code portability to other hardware platforms (i.e., GPUs produced by AMD or Intel and GPU-less computers), on the other it provides a simpler and high efficiency framework, since NVIDIA-specific optimizations can be automatically performed and new features can be added without requiring the consensus of other hardware manufacturers. Furthermore, NVIDIA is also the producer of the most widespread cards for GPU computing, namely the TESLA series (NVIDIA, 2016b), which are often used in HPC.

OpenCL is a framework for writing applications that can be executed across a series of heterogeneous computational devices that include not only GPUs, but also CPUs, Digital Signal Processors (DSPs), Field-Programmable Gate Arrays (FPGAs) and ARM processors. OpenCL is an open standard maintained and supported by the nonprofit Khronos Group consortium (Khronos, 2016). Compared to CUDA, OpenCL provides a more abstract framework, allowing direct portability of the code between hardware solutions of different vendors (NVIDIA, AMD and multi-core CPUs), at the expense of a slightly steeper learning curve and less versatile implementation of hardware-specific optimizations, e.g., concerning on-board memory and data communication with the CPU and with other GPU cards.

From the programming point of view, the CUDA and OpenCL frameworks show many similarities. In both a distinction is made between two logical parts of the code, namely a host part, to be executed on the CPU of the host machine, and a so-called device part, to be executed by many parallel threads (kernels) on the selected GPGPU device(s). Similarly, both frameworks distinguish between host and device memory, and provide functions to handle allocation and data-transfer between these. Concerning the API, this is unique in OpenCL, whereas for CUDA two APIs providing the same performance are available: the CUDA Driver API and the CUDA Runtime API. The former is also very similar to OpenCL, with a high correspondence between functions of the two frameworks.

Concerning performance, it is expected for GPU-based implementations to outperform CPU-based ones, as the workload, i.e., number of floating-point operations per second (FLOPS), increases (Lee et al., 2010). Due to its higher abstraction level and portability, OpenCL implementations have been found in the past to be slower than CUDA ones (Fang et al., 2010), although recently this gap has been reduced significantly (Kim et al., 2015) and mainly depends on the quality of the runtime implementation.

In addition, a common limitation for a generalized GPGPU-based approach is related to the size of the workload. The GPU is a kind of secondary computer within the main computer, featuring its own processor (the GPU), its own memory (the video RAM, or simply VRAM), its own conventions (instruction set, caching, data alignment, timing, etc.) and its own connectivity (usually a fast PCI-Express bus connecting the graphics card with the rest of the machine). This means that for a given task, the GPU-side execution performs a series of additional steps that are not required by its conventional CPU counterpart. A typical GPU-side computation consists in: 1) copying relevant information from system to device memory; 2) the compilation, parameterization and execution of a specific series of instructions; 3) waiting for the asynchronous execution to terminate; 4) copying the results back from device to system memory. Depending on the size and complexity of the problem, the overhead incurred by these four operations can be higher than the computational speedup provided through GPGPU.

These considerations make the choice of when and how to use GPGPU in SAR processing a more delicate matter, since typically image-wide operations (in principle a perfect match for GPU-side execution) are interleaved by smaller operations (e.g., setting up filtering parameters, sampling sub-image portions, etc.) which may introduce a severe overhead.

To gauge the impact of the aforementioned aspects and to guide the design and optimization of SARIPA, several tests, detailed in Peternier et al. (2013), were performed to analyze the behavior of common SAR-related algorithms carried out via GPGPU. In this section we discuss a performance test concerning 1D FFT calculation, since this is a core algorithm, which is heavily used within many SAR processing applications, including image focusing. The tests were carried out on Machine-A (Table 1).

We compare the highly-optimized FFT provided in the Intel's Math Kernel Library (MKL), taken as a reference FFT implementation for CPU-only performance, with GPU processing using CUDA (using

Table 1
Platform used for the FFT performance tests.

	Machine-A (FFT performance test)
OS	Windows 7
CPUs	1× Intel Core i7-930 @2.8GHz (4 cores)
RAM	12 GB
GPU	1× NVIDIA TESLA M2050 Fermi (3 GB of VRAM)
Storage	Not relevant for the test

Please cite this article as: Peternier, A., et al., Near-real-time focusing of ENVISAT ASAR Stripmap and Sentinel-1 TOPS imagery exploiting OpenCL GPGPU technology, Remote Sensing of Environment (2017), http://dx.doi.org/10.1016/j.rse.2017.04.006