NEAR-REAL-TIME MULTI-GPU ωK ALGORITHM FOR SAR PROCESSING

A. Davide Tiriticco, B. Marco Fratarcangeli

Sapienza University of Rome A,
Chalmers University of Technology B

C. Riccardo Ferrara, D. Stefano Marra

Advanced Computer Systems Spa C-D

ABSTRACT

This paper presents a Near-Real-Time multi-GPU accelerated solution of the ωk Algorithm for Synthetic Aperture Radar (SAR) data focusing, obtained in Stripmap SAR mode. Starting from an input raw data, the algorithm subdivides it in a grid of a configurable number of bursts along track. A multithreading CPU-side support is made available in order to handle each graphic device in parallel. Then each burst is assigned to a separate GPU and processed including Range Compression, Stolt Mapping via $Chirp_Z$ and Azimuth Compression steps. We prove the efficiency of our algorithm by using Sentinel-1 raw data (approx. 3.3 GB) on a commodity graphics card; the single-GPU solution is approximately 4x faster than the industrial multi-core CPU implementation (*General ACS SAR Processor*, GASP), without significant loss of quality. Using a multi-GPU system, the algorithm is approximately 6x faster with respect to the CPU processor.

***Index Terms*—** Remote Sensing, Image Processing, GPGPU, SAR, Real-Time Processing

1. INTRODUCTION

Synthetic Aperture Radar (SAR) is a remote sensing system able to provide wide area imaging by emitting an electromagnetic impulse and by receiving sequentially the echoes generated by the signal reflection with the Earth surface.

The obtained data is affected by deterioration caused by the satellite movements along and across track. A computational expensive processing, known as SAR focusing, is required in order to generate the final image and deliver it to the EO users.

More and more surveillance and operational services based on satellite data dictate stringent requirements to EO Service Providers about timeliness of the image production. For example, in the case of the EMSA Maritime Services for Oil Spill monitoring, the SAR-Native1 product should be delivered to the CleanSeaNet Data Centre as soon as possible. The maximum delivery time of Sentinel 1 (S1) products shall be between 20 and 35 min depending on the resolution class. S1 Satellite image licences will be procured through an EMSA-ESA agreement, however, image prices accounted by EO Service Providers to EMSA will depend on a coefficient for delivery time delay. This is to say that NRT production performance has economic consequences and this will be more and more the case as Copernicus data and services will proliferate.

The exponential improvements of computing power of the latest GPUs, allowed developer to efficiently implement low-cost Near-Real-Time (NRT) versions of the most important SAR processing algorithms such as Chirp Scaling Algorithm (CSA) and Range Doppler Algorithm (RDA), but there isn't yet a NRT implementation of the ωk Algorithm. In referenced works [1, 2, 3], an efficient parallel design of the RDA is presented, which achieved remarkable speed-ups without loss of accuracy. Moreover, in [4], an accelerated GPU implementation of the CSA has been proposed, able to process a queue of raw data acquired using a gigabit LAN.

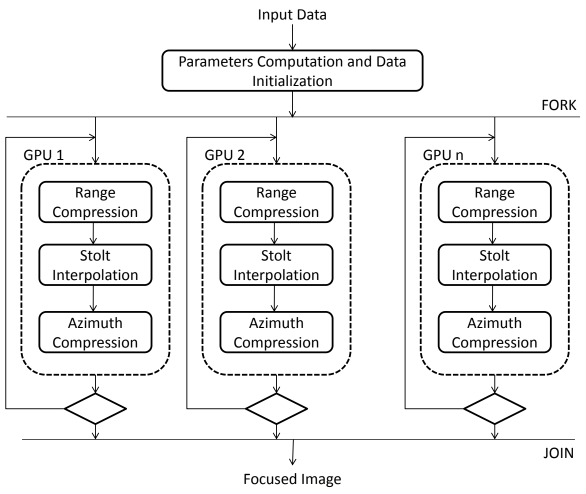

Fig. 1. General structure of the multi-GPU implementation of the ωk algorithm.

In this paper, continuing the work presented in [5], we prove how the ωk algorithm for Stripmap SAR data can be accelerated using CUDA on a multi-GPU architecture supplied directly by Nvidia Corporation. All the results presented in section 3, refers to a Sentinel-1 input raw data of size 22018 x 18903 pixels in float complex values, for a total occupation of almost 3.3 GB.

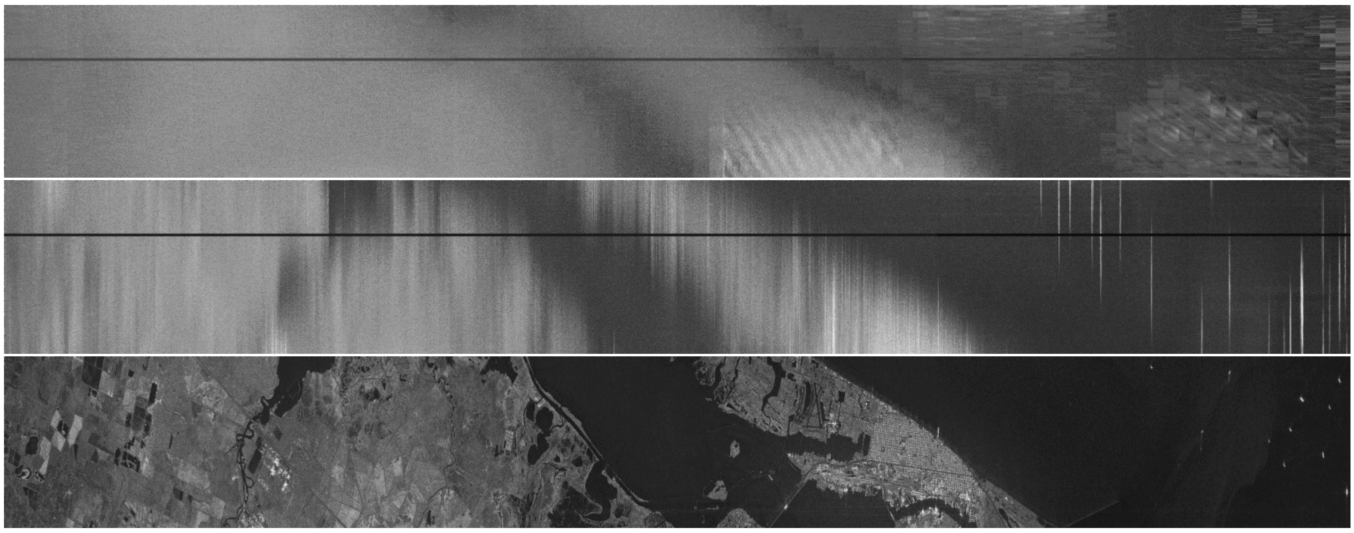

Fig. 2. *Top:* raw burst of Sentinel-1 in input. *Middle:* the Range compressed burst obtained after the application of the Range Compression step to the raw burst in input. *Bottom:* the focused burst after the application of the Azimuth Compression step.

2. METHODS

In figure 1, the general structure of the algorithm is presented. It operates by subdividing the input raw data in a configurable number of azimuth bursts. Each burst is processed entirely on GPU and independently from the others: this solution allows to easily scale with respect to the number of available GPUs. The implementation is based on CUDA and relies on cuFFT and cuBLAS libraries, respectively for Discrete Fast Fourier Transform and matrix operations.

The ωk Algorithm, executed on each block, is structured in three main steps: the algorithm works on Range Compressed data, so in the *Range Compression* step the input burst is first transformed in frequency domain in the Range direction. Then the block is multiplied by a reference function, obtained from the product metadata in form of a matched filter used to build the reconstructed nominal replica [6], which provides to correctly focus targets at the referenced range, R_{ref}, and to partially focus those far from R_{ref}. Before proceeding to the next step the input burst is transformed back to time domain; the *Stolt Mapping via* $Chirp_Z$ is used for the correction of the Range Cell Migration Effect (RCME)[7]. In this stage the input block is subdivided in a configurable number of range bursts. Each burst is transformed in frequency domain both in the Azimuth and Range directions, and then multiplied by a set of filter, computed from the doppler centroids and doppler rates, to compensate for the phase shift and for the echoes delay in the Range and Azimuth directions; the *Azimuth Compression* step provides to focus the input data in the Azimuth direction, by performing a correlation between the input burst and a frequency modulated filter, that depends on the satellite velocity and on the pulse repetition frequency. Finally the block is transformed back to time domain both along and across track. In figure 2, the output of each stage is shown.

As highlighted in this brief description, the algorithm heavily relies on filters application in the frequency domain, thus being well suited for the Single Instruction Multiple Thread (SIMT) architecture of the GPUs. Particular care has been taken, in choosing block size optimized for FFT operations [8], and avoiding memory transfers between host/device memory, by computing filters directly on GPU.

In the next section a comparison among CPU, single GPU and double GPU runs of the algorithm on the same input data is presented.

3. RESULTS

3.1. Hardware Specifications

- *CPU:* 2 x Intel(R) Xeon(R) CPU E5-2620 (6 cores at 2.10GHz).
- *RAM:* 8 x DIMM DDR3L 1600MHz 4GB.
- *SSD:* Intel SSD SATA 6Gb/s, 2.5” 240 Gb
- *GPUs:* 2 x Nvidia Tesla K40 (12GB VRAM).
- *OS:* CentOS 6.5 64-bit.

3.2. Performance Analysis

In this section we illustrate the performance of the GPU solution, comparing them with an industrial multi-core CPU implementation developed by Advanced Computer Systems ACS Spa (*General ACS SAR Processor*, GASP). The algorithm has been tested on a Sentinel-1 raw data as input, of size 22018 x 18903 pixels, in float complex values, for a total occupation of almost 3.3 GB. NRT processing is achieved when the execution time of the ωk Algorithm is lower with respect to the satellite acquisition time, that, for the proposed input, is approximately 10 seconds.

In table 1 the time performance comparison between the Single-GPU implementation and the CPU implementation (GASP) is shown.

Table 1. Performance comparison between the Single-GPU and the CPU implementations.

Steps	GASP(s)	Single-GPU(s)	Speedup
RGC	4.6	4.42	1.1x
Focusing	28.86	6.29	4.58x
Total	38.6	10.87	3.55x

The single-GPU implementation achieves a total throughput of 303.5 MB/s against the 85 MB/s shown by the CPU one.

Table 2 presents the results when the algorithm runs in a double GPU configuration. The total throughput reaches 509.2 MB/s and the algorithm performance scale almost linearly with the number of GPUs. An overhead is introduced by the imprecise subdivision of the number of Azimuth bursts among the available GPUs, and by the memory bandwidth sharing while copying data from/to host memory.

Table 2. The performance comparison between the Double-GPU and the CPU implementations.

Steps	GASP(s)	Double-GPU(s)	Speedup
RGC	4.6	2.41	1.9x
Focusing	28.86	2.83	10.19x
Total	38.6	6.48	5.95x

Figure 3 shows a comparison of the algorithm execution time as a function of the input size .

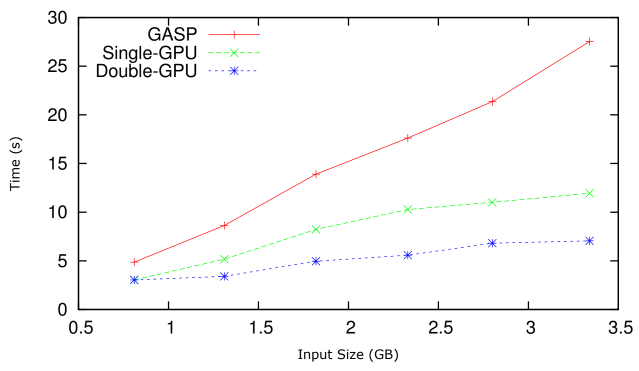

Fig. 3. ωk Algorithm performance comparison on input size variation.

It must be noted that the timing results described so far, do not include I/O operations that, for the proposed input, take a total time of 20.7 seconds. This work did not focus on this aspect that will be object of future investigations. Currently, the input data, is read and stored in host memory before the focusing algorithm starts and is written back on disk when all the Azimuth blocks have been processed. Alternative storage solutions, such as SSD units connected through the PCIe bus and advanced RAID configuration, will be considered for future enhancements, along with a re-implementation of the data block I/O pipeline to allow concurrency between disk operations and blocks processing.

4. CONCLUSIONS

In this paper we have presented a multi-GPU implementation of the ωk algorithm for StripmapSar data. We shown how it was possible to reach Near-Real-Time processing using low-cost and power efficient architectures such as GPUs. The research gives the opportunity to reach a turning point for remote sensing, where SAR processors could be directly mounted on ground stations, and the products made available almost in Real-Time to the EO community.

Future works will focus on the investigation of more efficient I/O storage solutions, and on the extension of the processor to support other SAR modes, such as Scan-SAR and Spotlight.

5. ACKNOWLEDGMENTS

Thanks to Nvidia for the support and for the test hardware provisioning.
Sentinel-1 test product courtesy of ESA.

6. REFERENCES

[1] C. Clemente, M. Di Bisceglie, M. Di Santo, N. Ranaldo, and M. Spinelli, "Processing of synthetic aperture radar data with gpgpu," in *Signal Processing Systems, 2009. SiPS 2009. IEEE Workshop on*, Oct 2009, pp. 309–314.

[2] Xingxing Jin and Seok-Bum Ko, "Gpu-based parallel implementation of sar imaging," in *Electronic System Design (ISED), 2012 International Symposium on*, Dec 2012, pp. 125–129.

[3] Bin Liu, Kaizhi Wang, Xingzhao Liu, and Wenxian Yu, "An efficient sar processor based on gpu via cuda," in *Image and Signal Processing, 2009. CISP '09. 2nd International Congress on*, Oct 2009, pp. 1–5.

[4] Yewei Wu, Jun Chen, and Hongqun Zhang, "A real-time sar imaging system based on cpugpu heterogeneous platform," in *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, Oct 2012, vol. 1, pp. 461–464.

[5] D. Tiriticco, R. Ferrara, S. Marra, and M. Fratarcangeli, “Gpu accelerated sar omega-k focusing,” Presented at Nvidia GPU Technology Conference, March 2014.

[6] Ian G. Cumming and Frank H. Wong, *Digital Processing of Synthetic Aperture Radar*, Arthec House, 2005.

[7] R. Lanari, “A new method for the compensation of the sar range cell migration based on the chirp z-transform,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 33, no. 5, pp. 1296–1299, Sep 1995.

[8] Nvidia, *CUFFT Library Users’s Guide*, 2014.