Performance evaluation of parallel stripmap CS-SAR imaging on NVLink-connected GPUs

Poster extended abstract

Masato Gocho, and Takehiro Hoshino
Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Japan
Email: { Gocho.Masato@ds, Hoshino.Takehiro@ap }.MitsubishiElectric.co.jp

***Abstract*—To develop a real-time CS-SAR (compressive sensing synthetic aperture radar) system, in which signals are randomly truncated and recovered using iterative forward and inverse imagings, we studied its parallel implementation that performs before and after fourier-transforms and is accelerated by the callback-customized cuFFT library on the NVLink-connected GPUs. When observing 8192×32768-signals every 6.0 s, we found that our CS-SAR reconstructions with four GPUs takes 2.1 s, which is 29.5 times faster than 96-threaded CPUs.**

I. Introduction

CS (compressive sensing) imaging, in which collected signals are randomly truncated and recovered using iterative forward and inverse imagings, is an effective technique to shrink the observation time, data-record size, and data-transfer size/time between sensors and computation devices. Thus, CS-applied systems for remote sensing have been well studied in recent years, such as satellite/airplane-embedded stripmap CS-SAR (synthetic aperture radar) imaging [1], medical CS-MRI (magnetic resonance imaging) [2] and CS-based 3D reconstruction from optical images [3]. However, the CS imaging time is at least 80 times the traditional imaging time if 40 iterations are required to reconstruct the truncated signals, despite the fact that the computation device continually receives observation signals every observation time. To reduce the CS imaging time, GPU-based CS imaging for MRI is well-studied [2], but there are no reports on the implementations and performances of stripmap SAR.

To develop a real-time CS imaging system within the observation time, like a traditional real-time system for stripmap SAR imaging [4], in this paper, we present the implementation and perforamce of parallelized stripmap CS-SAR reconstructions that perform on multiple GPUs, which communicate with one another through 100 GB/s NVLink.

II. Parallelized Stripmap CS-SAR Imaging

The computational flow of CS-SAR imaging is shown as Algorithm 1, where the input $\mathbf{S} \in \mathbb{C}^{M\times N}$ and output $\mathbf{V}$ are, respectively, the CS-observed sparse signal and SAR imagery, and N is the number of reflected pulses, each of which has M samples. The operators $\Psi^\dagger$ and Ψ represent, respectively, forward and inverse imagings, and operator $\mathcal{H}_\varepsilon$ represents the IHT (iterative hard thresholding) algorithm that replaces elements less than ε with zero. The input $\mathbf{S}$ is copied to the residual and reconstructed signals $\mathbf{u}_r$, $\mathbf{u}_s$ before K iterations. At each iteration, a max-element $v_{\max}$ is found from the residual imagery $\tilde{\mathbf{v}}$, which is converted from the residual signal $\mathbf{u}_r$ using operator $\Psi^\dagger$, to determine the threshold $\varepsilon = \alpha v_{\max}$, where α is assigned as 0.7 in this paper. A sparse imagery $\mathbf{d}$, which is thresholded $\tilde{\mathbf{v}}$, is reverted to signals $\tilde{\mathbf{u}}$ using the operator Ψ. The residual and reconstructed signals $\mathbf{u}_r$ and $\mathbf{u}_s$ are updated using the reverted signal $\tilde{\mathbf{u}}$ with β, which is calculated from $\tilde{\mathbf{u}}$ and $\mathbf{u}_r$. After K iterations, the operator $\Psi_*^\dagger$, which is assigned as $\Psi^\dagger$ or an accurate imaging [4], generates the SAR imagery $\mathbf{V}$ from the reconstructed signal $\mathbf{u}_s$.

In this paper, $\Psi^\dagger$ and Ψ are assigned as forward and inverse Omega-K algorithms [5], which comprise of the 2D-FFT (fast fourier transform), the Hadamard product of the phase compensation function $\theta(f_\tau, f_\eta)$, the resampling using 1D-LERP (linear interpolation), and the 2D-iFFT (inverse FFT). The resampling formula is given by

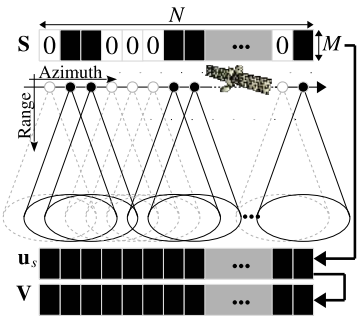
$$f'_\tau = \sqrt{(f_c + f_\tau)^2 - f_\eta^2} - f_c, \tag{1}$$

where f_τ and f_η are, respectively, the range and azimuth frequencies with uniform M and N samplings, f'_τ is the resampled frequency with nonuniform M samplings, and f_c is the center frequency. As with the forward Omega-K algorithm, the inverse alrotihm comprises the 2D-FFT, $\frac{1}{M}$scaling, the nonuniform iLERP (inverse LERP), the Hadamard product of the conjugate $\theta(f_\tau, f_\eta)$, and the 2D-iFFT.

Figure 1 shows the computational flow of the parallelized CS-SAR imaging. In our implementation using the cuFFT library, let P be the number of GPUs, where each GPU has column-major $\frac{M}{P}\times N$ and $M\times\frac{N}{P}$ matrices that fit the *inplace-shuffled* and *inplace* layouts for cuFFT APIs on multiple GPUs, and receives input signals $\mathbf{S} \in \mathbb{C}^{M/P\times N}$ behind the pre-computation of matrices that are the $\theta(f_\tau, f_\eta)$, indices

Algorithm 1 $\mathbf{V}$ = ***CS-Imaging***($\mathbf{S}$)

1: $\mathbf{u}_s^{(0)} = \mathbf{u}_r^{(0)} = \mathbf{S}$
2: **loop** $k = 1, 2, \ldots, K$
3: $\quad\tilde{\mathbf{v}} = \Psi^\dagger(\mathbf{u}_r^{(k-1)})$
4: $\quad\mathbf{d} = \mathcal{H}_\varepsilon(\tilde{\mathbf{v}})$
5: $\quad\tilde{\mathbf{u}} = \Psi(\mathbf{d})$
6: $\quad\beta = \dfrac{\langle\tilde{\mathbf{u}}, \mathbf{u}^{(k-1)}\rangle}{\langle\tilde{\mathbf{u}}, \tilde{\mathbf{u}}\rangle}$
7: $\quad\mathbf{u}_r^{(k)} = \mathbf{u}_r^{(k-1)} - \beta\tilde{\mathbf{u}}$
8: $\quad\mathbf{u}_s^{(k)} = \mathbf{u}_s^{(k-1)} + \beta\tilde{\mathbf{u}}$
9: **end loop**
10: $\mathbf{V} = \Psi_*^\dagger(\mathbf{u}_s^{(K)})$



2168-9253/18/$31.00 ©2018 IEEE
DOI 10.1109/CLUSTER.2018.00027

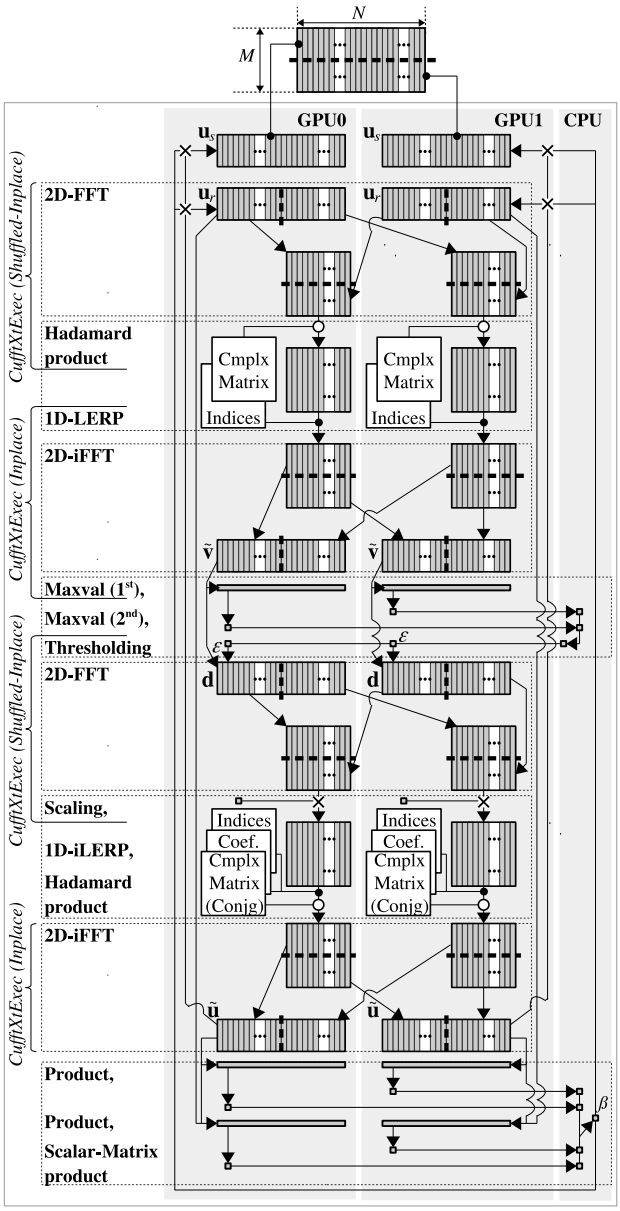


Fig. 1. Proposed parallel implementation for the CS-SAR reconstruction.

for LERP/iLERP, and distance coefficients for nonuniform grids at iLERP. The 2D-FFTs and 2D-iFFTs perform using *cufftXtExec* on the multiple GPUs, and each transform and other computations are fused at the load/store step using cuFFT-callback interfaces to reduce the number of memory accesses. At the $\Psi^\dagger$, each element of the 2D-FFT is stored after the Hadamard product, that of the 2D-iFFT is loaded with the 1D-LERP, and that of the 2D-iFFT is stored after warp-size reduction for the max-element search (1st Maxval). At the Ψ, each element of the 2D-FFT is loaded with the thresholding, and stored after the scaling. For other computations, $\frac{M \times N}{P}$-threads, whose block size is 32×16, perform on each GPU independently. The results of the "Product" and "Maxval" per GPU are reduced on the CPU, and the CPU-reduced/modified

TABLE I
EXPERIMENTAL PARAMETERS.

Center freq. (f_c) [GHz]		16.45
Resolution [cm]		10×10
Observation area [km]		0.9×0.7
# of samplings (M)		8,192
# of pulse hits (N)		32,768
CS	# of truncated hits	16,384
	Average PRF [Hz]	1,350
	Observation time[s]	6.0
	# of iterations (K)	40

PRF: pulse repetition frequency

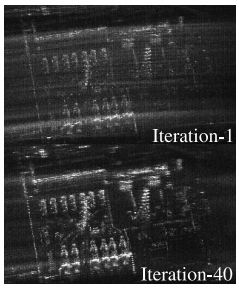


TABLE II
PROCESSOR CONFIGURATIONS AND FP32 PERFORMANCES.

Configuration		[GFLOP/s]	[GB/s]
CPU	4-way Xeon E7-8890v4, 96 cores	10,445	340
GPU	NVLink-connected Tesla V100×4	59,597	3,600

TABLE III
EXPERIMENTAL RESULTS (40 ITERATIONS).

	Total	Average iteration time [ms]					Avg.
	[s]	$\Psi^\dagger$	IHT	Ψ	update	Total	[Gp/s]
CPU	62.2	700.1	66.0	707.7	80.1	1554	0.2
GPU×1	5.5	56.9	7.8	59.4	12.6	137	2.0
GPU×1 (CB)	5.2	60.4	0.4	55.4	12.6	130	2.1
GPU×2 (CB)	3.7	37.9	0.4	40.0	8.3	87	3.1
GPU×4 (CB)	2.1	21.5	0.3	23.7	4.4	50	5.4

CB: Callback, Gp/s: Giga pixel operations per second.

value ε is located on the constant memory to reduce the number of memory accesses.

III. PERFORMANCE EVALUATION

We prepared the SAR signals, which were observed by a MELCO (Mitsubishi Electric Co.)-manufactured Ku-band airplane/traditional SAR [1], and truncated them 50% to simulate the CS-SAR system, as shown in Table I. Table III shows the measured results on CPU- and GPU-specialized single nodes, each of which is specified in Table II. When observing 50%-truncated 8192×32768-signals every 6.0 s, we found that our implementation with four GPUs takes 2.1 s, which is 30 times faster than the 96-threaded CPUs, within the observation time.

IV. CONCLUSION

In this paper, we presented a real-time CS-SAR imaging and evaluated its performance using airplane SAR imagery.

REFERENCES

[1] T. Hoshino, T. Hara, Y. Yokota, H. Hasegawa, and Y. Okada: Experimental Studies of Compressive Sensing for SAR with Ka-band Chamber Room and Ku-band Airplane SAR Data, in *Proc. 2017 IEEE Int'l Geoscience and Remote Sensing Symposium*, pp. 5366–5369, 2017.

[2] T. M. Quan, S. Han, H. Cho, and W. K. Jeong: Multi-GPU Reconstruction of Dynamic Compressed Sensing MRI, *Medical Image Computing and Computer-Assisted Intervention*, Springer, pp. 484–492, 2015.

[3] Y. Shen, J. Li, Y. Zhang, and Z. Zhu: Image compressed sensing reconstruction with 3D transform domain collaborative filtering, in *Proc. 2014 IEEE Int'l Conference on Multimedia and Expo*, pp. 1–6, 2014.

[4] M. Gocho, N. Oishi, and A. Ozaki: Distributed Parallel Backprojection for Real-Time Stripmap SAR Imaging on GPU Clusters, in *Proc. 2017 IEEE Int'l Conference on Cluster Computing*, pp. 619–620, 2017.

[5] T. Hoshino, T. Hara: A synthetic aperture processing for detecting underground cavities, in *Proc. 2017 Int'l Conference on Electromagnetics in Advanced Applications*, pp. 1112–1115, 2017.