PARALLELIZATION OF RANGE DOPPLER ALGORITHM FOR STRIPMAP IMAGE FORMATION IN AIR BORNE PLATFORM

MAMIDI VIDHYA SAGAR, KADALI LOKESH KIRAN, AM PRASAD

Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India -533003
E-mail: vidhyasagar.1246@gmail.com

Abstract - RDA (Range Doppler algorithm) SAR image formation algorithm is implemented in parallel processing approach for real time implementation in airborne platform in this paper. The operation of range compression and azimuth compression is reduced by using this parallel approach. The algorithm will decrease the computation load and increase the performance. The parallelization of RDA algorithm is implemented & tested on the Intel Xeon-E5 processor and verify that the algorithm has high real time property and parallel efficiency by using MPI dataflow.

Keywords - SAR, Image, Parallel Algorithm, RDA, MPI

I. INTRODUCTION

Synthetic-aperture radar (SAR) is a form of radar that is used to create 2-D images or 3-D reconstructions of objects, such as landscapes. The SAR image processing therefore can be summarized as three major tasks: Range compression, range cell migration correction, and azimuth compression. The raw data is after range compression, provide valuable information on the targets locations in the range direction but do present two problems: the range compressed signal is spread in the azimuth direction, and the signal migrates to other range cells. The purpose of SAR processing, ideally, is to convert the raw data into a single pixel in the final processed image. Several techniques are available to deal with the three tasks. Each technique has its advantages in either computation efficiency or high-quality imaging, and has its own beam width limitation and/or bandwidth restriction.

The data received from the Radar system are referred to as "signal data" or "raw data". The data are first demodulated to baseband, so that the nominal center range frequency is zero. The demodulated radar signal, \( S_0(\tau,\eta) \) received from a point target as

\[
S_0(\tau,\eta) = A_0 b_0(\tau-2R(\eta)/c) W_0(\eta-\eta_c) \exp \left\{ -j 4\pi f_o \tau R(\eta)/c - j 4\pi f_c \eta \right\} \exp \left\{ j K_0(\tau - 2R(\eta)/c) \right\} \rightarrow (1)
\]

where:

- \( A_0 \) is an arbitrary complex constant
- \( \tau \) is Range time
- \( \eta \) is Azimuth time referenced to closest approach
- \( \eta_c \) is beam centre offset time
- \( b_0(\tau) \) is Range envelope (a rectangular function)
- \( W_0(\eta) \) is Azimuth envelope (a sinc-square function)
- \( f_o \) is Radar centre frequency
- \( K_0 \) is Range chirp FM rate
- \( R(\eta) \) is instantaneous slant range
- \( f_c \) is Doppler centroid frequency

Let \( S_0(\tau,\eta) \) be the Range Fourier transform of \( S_0(\tau,\eta) \) of raw data , & \( G(f_0) \) be the output of the range matched filter can expressed as

\[
S_r(\tau,\eta) = \text{IFFT}_r \{ S_0(\tau,\eta) G(f_0) \} = A_0 b_0(\tau - 2R(\eta)/c) W_0(\eta - \eta_c) \exp \left\{ -j 4\pi f_o \tau R(\eta)/c \right\} \rightarrow (2)
\]

The data after the Azimuth FFT can be expressed as

\[
S_1(\tau, f_\eta) = \text{FFT}_\eta \{ S_0(\tau,\eta) \} = A_0 b_0(\tau - 2R(\eta)/c) W_0(\eta - \eta_c) \exp \left\{ -j 4\pi f_\eta R_\eta /c \right\} \exp \left\{ j K_0(\tau - 2R(\eta)/c) \right\} \rightarrow (3)
\]

In this paper Parallelization of RDA SAR image formation techniques will be presented: The range–Doppler algorithm, developed in the early 1980s, has been the most commonly used algorithm for processing SAR data images. It is computationally efficient and an accurate approximation for processing radar images. Fast Fourier Transform (FFT) is a key part in signal processing and forms an important part of building Synthetic Aperture Radar (SAR) processor [1-3]. Range Doppler algorithm is the most common method used in implementing SAR image processor, which accepts raw data and produces SAR images as output. FFT implementation affects the performance of SAR image formation process. Although many implementations of FFT on GPU (general processing unit) are available today, most of them are vendor specific, more generic in nature thereby require additional time overheads in terms of data accesses from memory. These overheads result in time delays of SAR image generation. In this, efficient parallelization approach of implementing FFT on GPU specific to SAR processor. This approach reduces the computation load and increase the performance of intermediate results, This implementation improves performance of SAR processor by decreasing overall time taken, and makes it as an ideal implementation for processing of SAR data in real time[1-2].
Parallelization of Range Doppler Algorithm for Strip Map Image Formation in Air Borne Platform

II. RDA SAR IMAGE FORMATION ALGORITHM

The 2D raw data in Fig.1 refer to the received baseband signal, which is a complex number after in-phase–quadrature-phase (I–Q) demodulation. The range compression is performed by the blocks of range FFT (on 2D raw data), FFT (on range reference function), range window function, multiplier, and inverse range FFT[4].

Range compression is a one dimensional convolution between range data and the range reference function, in order to decrease computation load which can be realized in frequency domain. The inner product of vectors can be obtained by the reformed algorithm with high degree of parallelism as well as FFT and IFFT operation. Azimuth compression is performed by the blocks of FFT on azimuth reference function, azimuth window function, multiplier, and inverse azimuth FFT. The output of azimuth compression is the reconstructed image of focused targets[4]. Similar to range compression, the procedure of azimuth compression is also one dimensional convolution, however, which is more complex. Strictly speaking, the reference function for the data of each range cell is different from each other. Furthermore, range cell migration should be corrected before azimuth compression, secondary range compression can be necessary if the migration is large enough. All the operations above can easily be realized in parallel as shown in the Fig. 2.

In order to decrease the computation load and increase the speed, it need to parallelize the FFT function in the range Doppler algorithm by master and slave process method

Master Process
The master process creates a table of real and imaginary numbers. The master process distributes a whole portion of the table to each individual slave process as a sub tables. So that every individual slave process can perform their respected task. finally the master process gathers all the calculated information from the slave processes.

Slave Process
The slaves process receives a portion of the table that hold imaginary and real values. It perform the FFT algorithm on their respected portion. This portion is later gathered by the master process.

There are many forms of parallel systems available viz., shared memory multiprocessors and message based multi-processors. With the increasing popularity of clusters, MPI (message passing interface) has become a popular form of writing parallel programs for massively parallel multi-processors.

Microsoft MPI (MS-MPI) is a Microsoft implementation of the Message Passing Interface standard for developing and running parallel applications on the Windows platform.

MPI function will divide the FFT data into number of cores present in the processor by two equal half parts
• Even part = \( \text{bigN} / \text{Comm.sz} / 2 \) → array to save the data for even half
• Odd part = \( \text{bigN} / \text{Comm.sz} / 2 \) → array to save the data for odd half
• bigN = problem size
• Comm.sz = how many cores used in the processor.

MPI_Scatter will scatter the data table into sub tables to different cores and then at the ending MPI_Gather will collect the all data from even part and odd part array.

III. APPROACHES TO PARALLELIZATION

Scope for parallelism increases with stages

The definition of the FFT is represented by

\[
X(k) = \sum_{n} x(n) e^{-2\pi jkn/N} \rightarrow (4)
\]

where \( x(n) \) is the input sequence, \( X(k) \) is the output sequence, \( N \) is the transform length, Both \( x(n) \) and \( X(k) \) are complex valued sequences of length \( N = r^2 \), where \( r \) is the radix.

IFFT:

\[
x(n) = \frac{1}{N} \sum_{k} X(k) e^{2\pi jkn/N} \rightarrow (5)
\]

\( N = 0 \)

Let us consider the Decimation-in-frequency (DIF) form of the length 8 Radix-2 algorithm. In this first stage all the data is mingled and it performs a single 8-point DFT in the starting stage [5-6].

In the second stage, it visibly breaks into two separate 4-point DFTs and in the third stage, it has eight separate 2-point DFTs. Fig 3 below shows the parallel composition of 8-point Radix-2 FFT algorithm. The data in the oval is the number of elements on which computation is performed and the number below the oval is the processor number that would perform the computation. Solid lines indicate distribution of data to another core[6].

In the Decimation-in-time form (DIT), the arrows in the a Fig 3 is to be reversed and the stages should be numbered from bottom-up as shown in the Fig 4.

In Decimation-in-frequency it have more parallelism in the first stage and the number of divisions clearly decreases by a factor of two. The advantage of this form is that the result is already in the master node by the end of the last stage where as in the case of decimation-in-frequency form. The result gathering phase after the last FFT stage[6]. The disadvantage of this technique is that while only one processor will be active in the first stage of the decimation-in-frequency form, all the other processors will be idle in the first stage. But the number of processors used will be increased in the subsequent stages.

IV Implementation

The algorithm have been implemented in C by using Message Passing Interface (MPI) in windows operating system and tested on Intel Xeon -E5 processor which has 24 cores as shown in the fig 5. Internally they communicate using the shared L3 cache for Non-uniform memory access systems (NUMA), as the core count goes up, it gets increasingly complex to keep cache coherency, Intel uses the MESIF (Modified, Exclusive, shared, Invalid and Forward) protocol for cache coherency. At the hardware level all cores, the integrated memory, PCIe (peripheral component interconnect express) and other controllers are connected to the shared L3 using a common data fabric, Intel uses the Ring Interconnect systems.
For 16384 samples:

<table>
<thead>
<tr>
<th>Sequential (no. of cores)</th>
<th>1</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallelization (no. of cores)</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>25.6</td>
<td>14.0</td>
<td>7.62</td>
<td>4.41</td>
<td>2.66</td>
</tr>
<tr>
<td>Speed</td>
<td>0.03</td>
<td>0.07</td>
<td>0.13</td>
<td>0.22</td>
<td>0.37</td>
</tr>
<tr>
<td>efficiency</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
<td>0.02</td>
<td>0.02</td>
</tr>
</tbody>
</table>

### IV. RESULTS & ANALYSIS

Execution time, speed, efficiency are considered as performance measure for evaluating the proposed parallelization technique.

Execution time will be decreased with the more processors that are being utilized. The exception is when number of processors is greater than 16. This is due to each node in the cluster has 16 possible cores. When there is more than 16 processors the overhead of communication between nodes slows then execution will slow down as shown in the Fig 6.

**Fig 6. Execution time**

Speed is greatest when using the max amount of cores in the processor as shown in the Fig 7.

**Fig 7. Speed**

Efficiency is decreases by increasing the number of cores. The efficiency will be depends on the ratio of speed by the number of cores using for that sample as shown in the Fig 8.

\[ \eta = \frac{\text{speed}}{\text{node}} \rightarrow (6) \]

**Fig 8. Efficiency**

### V. CONCLUSION

The tested results show that the parallel RDA SAR image formation algorithm processing of high degree parallelism. The parallelization method requires less imaging time and has higher parallel efficiency than the Sequential method when the quantity of cores is large. MPI dataflow plays a key role in the parallelization of range Doppler algorithm in strip map image formation. By using this parallelization method in the RDA algorithm, the computational load is decreased and it increases the performance, which facilitates application of RDA parallelization in finer resolution strip map SAR with high velocity platform.

### REFERENCES


[6] Michael Balducci, Ajitha Choudary and Jonathan Hamaker, Comparative Analysis of FFT algorithms in sequential and parallel form, Mississippi State University