Parallel Synthetic Aperture Radar Processing on Workstation Networks

Peter G. Meisl  Mabo R. Ito  Ian G. Cumming

Dept. of Electrical Engineering
University of British Columbia
Vancouver, B.C. Canada
peterm@ee.ubc.ca, mito@ee.ubc.ca, ianc@ee.ubc.ca

Abstract

Synthetic Aperture Radar (SAR) signal processing poses a significant challenge due to its very large computation and data storage requirements. This paper presents the computational requirements of a typical high resolution satellite SAR data processing scenario. A classification of approaches to partitioning the SAR problem for parallel processing is given. The suitability of Networks of Workstations (NOW) for SAR processing is analyzed for a number of partitioning approaches. The network throughputs required to support SAR processing on NOW are derived. SAR processing is found to demand extremely high network throughput that is difficult to achieve with today's technology.

1 Introduction

Synthetic Aperture Radar (SAR) is a remote sensing technique for obtaining high resolution images of the earth’s surface [5]. One of the traditional problems with SAR as a remote sensing tool is the huge amount of signal processing that is required to form an image from the raw data. High speed digital SAR processing systems have usually been very complex and expensive and have often relied on custom hardware designs. However, the power of general purpose computer systems has been catching up with the requirements posed by SAR processing, opening up new possibilities for SAR processor architectures.

This paper considers the accurate processing of satellite SAR data to form high resolution images. This is the most demanding type of SAR processing, with large computation and memory requirements. The context for satellite SAR processing is established in the two sections that follow this one. Section 2 gives an overview of the SAR processing problem and defines some of the necessary terminology. The SAR processing algorithm used in the analysis in this paper is the range-Doppler algorithm. The algorithm is presented in Section 3. Although this paper focusses on range-Doppler, the work on finding efficient parallel implementations is also applicable to most other SAR algorithms.

Due to the large computational requirements, SAR processors usually make extensive use of parallel processing. The partitioning of SAR processing is discussed in Section 4. A classification of approaches for coarse-grained partitioning is given.

A very promising new option for computationally intensive applications is the use of networked groups of many powerful and inexpensive workstations [1]. The key to successful algorithm implementation on networks of workstations (NOW) is to come up with a partitioning of the problem that has a computation to communication mix that matches that of the workstations and their interconnection network. Since SAR processing can be split into fairly large chunks of computation it appears to be a promising application for workstation clusters. Little experience exists in predicting the usefulness of NOW for high performance SAR applications. (A PVM-based system using older technology is described in [3].)

Section 5 analyzes the use of NOW for parallel SAR processing. The most appropriate partitioning approach from Section 4 is used as a basis for a simple model of a NOW-based SAR processor. The model is used to derive some network bandwidth requirements.

2 SAR Processing Context

The basic geometry of the SAR processing application is shown in Figure 1. The sensor transmits pulses and receives the echoes in a direction approximately perpendicular to the direction of travel. The received signal is digitized and stored in the memory of the signal processor as a two dimensional array of samples. One dimension of the array represents distance in the slant range direction between the sensor and the target and is commonly referred to as the range direction. The other dimension represents the along-track or azimuth direction. Each line of data with a common azimuth index represents data collected during one pulse repetition interval.

In essence, the processing of SAR data is a two-dimensional space-variant convolution. When the received data is viewed as a superposition of returns from point scatterers, the processing problem consists of compressing the target response that is spread out in both range and azimuth directions.
to a single point. As the range and azimuth time scales are so different, the range and azimuth directions can be treated separately during processing. In each dimension the data is convolved with a matched filter. In azimuth, the situation is complicated by the fact that the returned signal from a single point target appears in a number of range cells during the time that it is illuminated by the radar. The energy corresponding to a target needs to be collected into one range cell before it can be compressed with the matched filter.

The key parameters that determine the sizing of a SAR processing system are the sizes of the data array in range, $N_r$, and azimuth, $N_{az}$, the matched filter lengths in range, $L_r$, and azimuth, $L_{az}$, and the data collection rate. All of the signal and coefficient data are complex numbers with a precision of usually at least 16 bits.

In order to process SAR data to its full inherent resolution with reasonable computing resources, an algorithm is required that is both accurate and efficient. An algorithm that satisfies these requirements, the range-Doppler algorithm, is discussed in the next section.

### 3 Range-Doppler Algorithm

The range-Doppler algorithm and its variants is probably the most widely used SAR algorithm. It was first developed by MacDonald Dettwiler and Associates (MDA) and the Jet Propulsion Lab (JPL) in 1979 for the processing of SEASAT data [4] [6]. The core steps in the algorithm are shown in Figure 2 and are described below.

1. **Range FFT**
   
   An FFT is performed on the data in the range direction.

2. **Range Compression (RC) with Secondary Range Compression (SRC)**
   
   Range compression and SRC are performed by means of an array multiply in the range direction.

3. **Range IFFT**
   
   An IFFT is performed in the range direction.

4. **Azimuth FFT**
   
   An FFT is performed in the azimuth direction.

5. **Range Cell Migration Correction (RCMC)**
   
   Range cell migration correction is performed by a shift and interpolation operation that lines up the target trajectories in memory.

6. **Azimuth Compression (AC)**
   
   Azimuth compression is accomplished with an array multiply in the azimuth direction.

7. **Azimuth IFFT**
   
   An IFFT in the azimuth direction completes the formation of the image.

![Figure 2: Range-Doppler SAR processing algorithm](image-url)
of several basic operation types. The operation types considered are FFTs, complex vector multiplications (CVM), filtering or interpolation operations, and scalar operations. When calculating computation amounts, a single operation (OP) (add, multiply, etc.) on real operands is used as the base unit. It is seen from Figure 3 that the majority of the computation in the range-Doppler algorithm consists of FFTs.

Numerous operations that are not part of the core SAR processing algorithm or that have much smaller computation requirements than the major steps were not included in the above analysis. A production SAR processor would have a significantly more complicated sequence of operations than the one shown in Figure 2. However the basic computational breakdown of the compute-intensive tasks remains similar.

Some typical processing parameter values for a Radarsat SAR processor are shown in Table 1. This scenario requires a computation rate of 4.5 GOP/s for real-time operation. The memory requirements for the algorithm vary during the course of execution between 140 MB and 280 MB. These values assume the basic range-Doppler algorithm described above with the addition of some typical post-processing operations, namely azimuth and range resampling with over-sampling ratios of 1.4 in each direction.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse repetition frequency, PRF</td>
<td>1345 Hz</td>
</tr>
<tr>
<td>Range swath length, N_r</td>
<td>6000</td>
</tr>
<tr>
<td>Range compression FFT length, N_{fft}</td>
<td>8192</td>
</tr>
<tr>
<td>Range matched filter length, L_r</td>
<td>600</td>
</tr>
<tr>
<td>Azimuth compression FFT length, N_{az}</td>
<td>4096</td>
</tr>
<tr>
<td>Azimuth matched filter length, L_{az}</td>
<td>600</td>
</tr>
<tr>
<td>Computation word length, N_{bit}</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

Table 1: SAR processing scenario parameters

4 SAR Processor Partitioning

The high computational requirements for SAR processors result in a need to partition the SAR algorithm to run on multiple processors. This section describes some partitioning approaches. Section 4.1 discusses some characteristics of SAR processors that affect partitioning. Section 4.2 looks at the issue of granularity. Section 4.3 presents a classification of coarse-grain partitioning approaches.

4.1 Considerations for Parallelism

Some of the characteristics of the range-Doppler SAR algorithm that affect the partitioning in a parallel SAR processor are:

- **structure of the processing flow diagram:** Range-Doppler decomposes naturally into large sequential steps with fairly simple data dependencies and synchronization requirements.
- **data array sizes in range and azimuth directions:** The large data arrays used in SAR processing imply higher computational requirements but make partitioning easier and more efficient.
- **matched filter lengths:** The long matched filter lengths required for high accuracy processing and the resulting overlaps required between data partitions have an influence on efficiency.
- **matched filter update intervals:** The filter update intervals can place upper bounds on data partition sizes.

4.2 Granularity of Parallelism

The granularity of a parallel system is the size of the units by which work is allocated to processors. For SAR processing, two general levels of granularity can be established. The bulk of SAR processing is made up of digital signal processing (DSP) operations like FFTs and vector multiplies. Fine-grain parallelism can be considered to be parallelism...
that subdivides these fundamental operations. Coarse grain parallelism allocates each fundamental DSP operation to a single processor. Coarse grain parallelism in SAR processing offers the advantages of simpler scheduling, fewer synchronization problems, and the ability to use commercial system components. In contrast, fine grain parallelism offers the potential of more parallelism and therefore higher speed-ups but requires requires low communication times between processing elements (PEs), and the lack of suitable commercial components usually implies custom VLSI designs.

Both approaches have potential application in SAR processing. Since the intent of this work is to examine the suitability of the coarse grain architecture provided by workstation networks, only coarse grain partitioning will be examined.

4.3 Classification of Partitioning Approaches

This section gives a classification of common coarse grain partitioning approaches for SAR. The classification is based on the scheme given in Section 5.2 of [2]. There are many characteristics that need to be considered when classifying parallelizations of algorithms, and no one set covers all situations. The partitioning approaches in this section should be taken as starting points for describing a specific design.

The partitioning approaches are illustrated in Figures 5 through 7. The meanings of the symbols that are used in these figures are shown in Figure 4.

![Figure 4: Symbols used to depict approaches to parallelism](image)

4.3.1 Vertical Partitioning

The first approach to partitioning SAR processing is vertical partitioning, which is also known as pipelining or temporal parallelism (see Figure 5). The granularity of the partitioning can be varied by changing the number of pipeline stages. This form of partitioning increases the throughput of the processor but increases its latency. Pipelining is well suited to SAR processing since most SAR algorithms consist of a number of sequential steps.

![Figure 5: Vertical Partitioning](image)

4.3.2 Horizontal Partitioning

The next approach is data parallelism or horizontal partitioning as shown in Figure 6. Horizontal partitioning divides the data set among the processors. Each processor performs essentially identical operations on subsets of the data. There are a number of options that can be applied to horizontal partitioning:

- the use and granularity of pipelining in each of the parallel processors:
  - none (strictly horizontal partitioning)
  - pipelining with some amount of granularity (horizontal-vertical partitioning)
- the orientation of the data partitioning:
  - azimuth subswaths or strips
  - range subswaths or strips
  - submatrices
- the size of the data partitions: Smaller partitions will allow for more parallelism but will result in decreased efficiency.

![Figure 6: Horizontal partitioning](image)
of distributed corner turns. Each processor has enough data to perform processing in both range and azimuth directions. This places a lower limit on the size of the data partitions since they must be at least as long as the matched filters in each direction. In a dynamic partitioning, the option exists to perform corner turn operations in which PEs exchange data until each has a data set suitable for processing in the orthogonal direction.

4.3.3 Vertical-Horizontal Partitioning

Vertical-horizontal partitioning is a combination of the previous two partitioning methods. The processing is partitioned vertically through the use of pipelining and horizontally by dividing the data among the processors. This approach can be used to take advantage of the fact that in many SAR algorithms range processing is performed first, then the data is corner turned, and then azimuth processing is performed. The corner turn buffer is a natural synchronization point for the parallel processors. Vertical-horizontal partitioning and some of its data partitioning options are shown in Figure 7. Some of the options that can be chosen are as follows:

- range
  - range-azimuth
  - range-range
- azimuth
  - azimuth-azimuth
  - azimuth-azimuth
- submatrices or other options
- the size of the data partitions

5 Networks of Workstations

This section examines the suitability of NOW in SAR processing. Section 5.1 takes a general look at the advantages and disadvantages of using general purpose workstations for DSP applications like SAR. Section 5.2 presents a model and performance analysis of a NOW-based SAR processor.

5.1 General Purpose Workstations in SAR Processing

The rapidly increasing performance of RISC-based workstations has made them suitable for a wide variety of tasks that previously required large computers with special accelerators. The large market for workstations gives them an advantage over other high performance systems in that they are usually quick to incorporate new technologies and software advances.

Although workstations can be used for DSP purposes, they are really optimized for running applications under general purpose operating systems like UNIX. This leads to a number of inefficiencies when they are applied to large DSP applications. Some potential problems are listed below.

- Workstations make extensive use of caching. Depending on how the caching is handled, it is not necessarily useful and may even be detrimental for SAR processing.
- The shared memory and bus can be a major bottleneck in multiprocessor systems, especially for corner turn operations.
- Performance is difficult to predict in comparison with more specialized architectures.
- Workstations are slower at FFTs by a factor of ten in comparison with the fastest FFT processor chips.

Unfortunately most SAR processing situations are still too demanding for cost effective implementation on individual single processor or multiprocessor workstations. This leads us to look for a parallel processing solution that incorporates the advantages of workstations: NOW.

5.2 NOW Performance Analysis

This section presents a simple model that will be used to test the potential usefulness of NOW for SAR processing. Static horizontal partitioning is used as the partitioning approach since it involves the least inter-processor communication of
all the partitioning strategies. The details of the model are given below.

- A network of ten workstations is assumed, where each workstation has either 25, 50, 75, or 100 MOP/s of sustained processing speed for SAR operations. These are reasonable values for current and near-future workstations. A 1 ms 1k complex FFT time is typical for many modern workstations, which translates into 50 MOP/s. Ten 50 MOP/s workstations would be capable of approximately 1/10 real-time processing. As an example, an Alpha workstation has a peak performance of 200 MOP/s and achieves about 100 MOP/s on a 1k complex FFT.

- The processing scenario is that of the range-Doppler algorithm characterized in Table 1. The real-time computation rate is about 4.5 GOP/s. The image size is 160 MB at beginning of execution and 140 MB at the end.

- All data is assumed to originate from a single node in the network and the output data is collected at the same node (see Figure 8). This is a realistic situation since a single frame synchronizer or tape drive is usually the source of the data. The network bandwidth figures used in this section refer to the bandwidth required at the connection(s) of the I/O node to the network. No other assumptions about the network topology are made.

![Interconnection Network](image)

**Figure 8: NOW SAR processor**

- The data is partitioned between the processors at beginning of computation and is collected at the end. Thus the only inter-processor communication is the transferring of the complete image data array twice: once at the beginning and once at the end.

- It is assumed that communication can be completely overlapped with computation. This implies that the execution time will be determined by the longer of the computation time and the communication time.

- Algorithm overheads due to things like data overlaps are ignored. This allows the effect of network communications to be studied in isolation. It should be remembered that the actual speed-up may be considerably less depending on the algorithm parameters.

The plot of speed-up versus network throughput in Figure 9 shows that the network bandwidth must be very large before the full computational power of the workstations can be realized. The speed-up is defined as the execution time on one workstation divided by the execution time on ten workstations of the same type. For example, an attempt to process a full size SAR image on a network of workstations connected by a conventional 10 Mbps Ethernet network would be a disaster: the speed-up would be less than one.

![Speed-up for ten workstation network vs. network throughput](image)

**Figure 9: Speed-up for ten workstation network vs. network throughput (numbers on graph indicate speed of workstations in MOP/s)**

The relationship between computational performance and communications bandwidth can be described by the computation to communication ratio (R/C ratio). For example, a workstation capable of 100 MOP/s and a network capable of 10 MB/s have an R/C ratio of 10 OPs of computation per byte of communication. The SAR processing scenario used in this paper has an R/C ratio of about 38 OPs/byte. If the R/C ratio of the architecture is lower than that of the algorithm then the processors will be the bottleneck, if it is higher, then communication will be the bottleneck. R/C ratios are plotted as a function of network bandwidth in Figure 10. A horizontal line indicates the ratio that characterizes the hypothetical SAR problem. The intersections between the curves and the line correspond to the communications throughputs at which the ten workstation network achieves linear speed-up.

Since the R/C ratio of the SAR problem is known, the network bandwidth required for a given amount of processing power can be predicted. This is shown in Figure 11, which plots the required communications bandwidth against the number of workstations. It can be seen that extremely high network bandwidths are required to achieve linear speed-up.

If the more realistic assumption is made that only 50% of the computation and communication can be overlapped
in time, then it becomes much more difficult to achieve linear speed-up. This can be seen from Figure 12 which is a plot of speed-up versus network bandwidth for this new assumption.

The effects of the data overlaps required in the horizontal partitioning approach must also be considered. The most common data partitioning method involves splitting the data into range subswaths, where each range line is split into sublines with enough overlap to account for the filter lengths. The range FFT size is usually chosen to be the next power of two larger than the length of the range sublines. Figure 13 shows the operation rates required by this method for up to 16 processors. The operation rate required of each subswath processor and the total operation rate, which is the sum of the individual computation rates, are shown in the graph. The discontinuities in the graphs occur at points where the range FFT length changes. The graph shows that the data overlaps can lead to large inefficiencies. This is especially apparent in portions of the graph where the number of processors is increased, but the workload per processor does not decrease very much since the range FFT length has not changed.

A global corner turn operation can be used to overcome the inefficiencies caused by filter overlaps. The addition of a global corner turn step allows the required data overlap regions to be much smaller, since the direction of the data partitioning can be made the same as the direction of the matched filtering. A corner turn could be accommodated by using dynamic horizontal or vertical-horizontal partitioning. While this helps solve the efficiency problem due to data overlaps, it worsens the bandwidth-imposed efficiency problem. The data array now needs to be transferred over the network four times instead of twice. In this case, the R/C ratio for the two stages of the algorithm falls to about 10 OPs/byte. This exasperates the network bandwidth problem as can be seen in Figure 10.
6 Conclusions

A performance analysis of the range-Doppler SAR processing algorithm on a NOW architecture has shown that very large network bandwidths are necessary for practical implementation of SAR processing on NOW. For example, in a ten workstation network where each workstation has a performance of 100 MOP/s, an effective network bandwidth of over 200 Mbps is required. This is beyond most commonly available networking technologies. Extremely low overhead communications software is also a necessity. It should be stressed that the bandwidth numbers shown in this paper are effective numbers after all overheads. Protocol and operating system overheads can easily add 100% to the bandwidth required.

In the analysis of the NOW architecture, the horizontal partitioning approach was used since it requires the minimum amount of inter-processor communication. However it was seen that this approach also suffers serious efficiency problems due to data overlap requirements. An approach that includes a global corner turn could potentially solve this problem but it would increase interprocessor communication significantly.

The main disadvantages of existing NOW systems are the severe performance limitations due to the high latency and low bandwidth of their communication networks. In order for NOW implementations to be successful, both a high bandwidth network and low overhead communications are necessary. NOW-like supercomputers like the IBM SP-2 and the Cray T3D overcome some of these limitations, but at increased cost.

NOW become more appropriate for SAR processing if the algorithm requires a significantly higher amount of computation per image pixel than the standard image formation algorithms like range-Doppler. A possible candidate application is a SAR processor that incorporates large amounts of computation in post processing functions such as those required for image analysis and understanding.

References