Parallel Programming Templates for Massive Remote Sensing Data Processing

Yan Ma¹, Lizhe Wang¹, Dingsheng Liu¹, Peng Liu¹ and Jun Wang²

¹Center for Earth Observation and Digital Earth, Chinese Academy of Science, P. R. China.

²Department of Electrical Engineering & Computer Science, University of Central Florida Orlando, United States

Abstract—Remote Sensing (RS) data processing is characterized by massive remote sensing images and increasing amount of algorithms of higher complexity. Parallel programming for data-intensive applications like massive remote sensing image processing on parallel systems is bound to be especially trivial and challenging. We propose a C++ template mechanism enabled generic parallel programming skeleton for these remote sensing applications in high performance clusters. It provides both programming templates for distributed RS data and generic parallel skeletons for RS algorithms. Through one-side communication primitives provided by MPI, the distributed RS data template could provide a global view of the big RS data whose sliced data blocks are scattered among the distributed memory of cluster nodes. Moreover, by data serialization and RMA (Remote Memory Access), the data templates could also offer a simple and effective way to distribute and communicate massive remote sensing data with complex data structures. Furthermore, the generic parallel skeletons implement the recurring patterns of computation, performance optimization and pass the user-defined sequential functions as parameters of templates for type genericity. With the implemented skeletons, Developers without extensive parallel computing technologies can implement efficient parallel remote sensing programs without concerning for parallel computing details. Through experiments on remote sensing applications, we confirmed that our templates were productive and efficient.

Key word: parallel programming; generic programming; data-intensive computing; remote sensing image processing

I. INTRODUCTION

Recently the amount of remote sensing data continuously acquired by satellite and airborne sensors has been dramatically increased [1][2]: typically several terabytes remote sensing data are acquired per day and the volume of single dataset is several gigabytes. Extremely massive data are required to be processed and analyzed daily. The remote sensing datasets characterized by multi-band image data structures and related geographical information used for image processing are usually organized in various formats, thus making it rather complicated and difficult for algorithms to load and reside remote sensing data. With the proliferation of data, remote sensing data processing becomes extremely challenging because of the massive image data, the pixels of higher dimensionality, and various algorithms with higher complexity, together with complicated data access modes resulted from different correlations between algorithms and remote sensing data. In particular, many time-critical applications like disaster monitoring even require real-time or near real-time processing capabilities.

The enormous computational requirements introduced by the unprecedented massive data and various complicated algorithms especially many time-critical applications have far outpaced the computing capability of single computer. It thus is an effective solution to address these computational challenges to incorporate cluster based high-performance computing (HPC) models in remote sensing image processing[2][3]. However, parallel cluster systems are characterized by increasing scale and multilevel parallel hierarchy. To develop efficient codes for parallel remote sensing algorithms usually involves handling data slicing and distribution, task partition, message passing among distributed memory spaces and shared memory management for multicores, synchronization and communication with low-level APIs. Furthermore, it still remains a big challenge for parallel systems to load massive datasets into memory before processing. It is also quite complicated and inefficient to exchange among cluster nodes the datasets with multi-band images and complex structured geographical information. Therefore, parallel programming on multicore clusters for processing massive remote sensing data is complicated, difficult and error-prone.

Generic parallel algorithms are abstract and recurring patterns lifting from many concrete parallel programs and conceal parallel details as skeletons. The approach of generic parallel computing relies on type genericity to resolve polymorphism at compile time, so as to reduce the runtime overhead while providing readability of high-level. To properly solve the aforementioned problems of processing massive RS data in HPC clusters, we propose the RS-GPPS (Generic Parallel Programming Skeletons) for massive remote sensing data processing applications. The RS-GPPS is enabled by the template class mechanism and work on top of MPI. We focus on so-called programming templates, which are
parameterized computation patterns and used to implement algorithm skeletons. The main contribution of the RS-GPPS is that it provides both template for distributed RS data and skeletal parallel templates for RS algorithms. The massive RS data including multi-dimensional image and complex metadata are required to be divided into blocks and distributed among nodes. The massive RS data are then abstracted and wrapped as generic distributed RSData class template. The skeletal parallel templates are generic parallel algorithms perform computations on distributed RS data. These templates express parameterization of parallelism without concern for implementation details like data distribution and task partition, complicated access modes of RS data, and all low-level architecture dependant parallel behaviors. Developers are only required to program user-defined sequential codes, which is also the type parameter of templates, and instantiate skeletal parallel template. The RS parallel processing programs thus can be generated based on the sequential codes with high parallel efficiency and minimal runtime overhead of polymorphism. In the implementation, the MPI one-sided messaging primitives and serialization of complex data structure are used to offer a simple data accessing and residing of whole massive RS data in distributed memory spaces among nodes.

The rest of this paper is organized as follows. The next section reviews related work, and section III gives the problem definition of massive remote sensing image processing in high performance clusters. Section IV presents the design and implementations of the generic parallel skeletons for massive RS algorithms in multicore clusters. Section V discusses the experimental analysis of program performance, and section VII concludes this paper.

II. RELATED WORK

Clusters of multicore SMP (Symmetric Multi-Processors) nodes have been widely accepted in high performance computing and a number of programming paradigms are developed for this hierarchical architecture. The OpenMP [4] model is used for parallelization of shared-memory and distributed shared-memory clusters [5]. The Message Passing model (MPI) [6] is employed within and across the nodes of clusters. The hybrid programming paradigm MPI+OpenMP [7] exploits multiple levels of parallelism: the OpenMP model is used for parallelization inside the node and MPI is for message passing among nodes. However, to develop parallel programs for hierarchical cluster architectures with low-level programming models and APIs, for example MPI and OpenMP, is still difficult and error-prone.

The skeletal parallel programming with higher-level patterns is adopted by researchers to simplify parallel programming. The eSkel [8] library provides parallel skeletons for C language. It can generate efficient parallel programs but leave users to handling low-level API with many MPI-specific implementation details. Lithium [9] is a platform-independent Java library and provides a set of predefined skeleton classes. Muesli [10] offers polymorphic C++ skeletons with high level of abstraction and simple APIs. The programs can be produced by construction of abstract skeleton classes and deal with data transmission via a distributed container. However, it suffers a large overhead paid for runtime polymorphic virtual function calls. Google’s MapReduce model [11], which supports Map and Reduce operations for distributed data processing in a cluster, is a simple yet successful example of parallel skeletons.

The generic programming approach uses templates to program generically. This concept has been exploited efficiently in the Standard Template Library (STL) [12], which has been extensively applied due to its convenient generic features and efficient implementations. Furthermore, the polymorphism is resolved at compile time because of its usual type generivity. The QUAFF [13] skeleton-based library offers generic algorithms. It relies on C++ templates to resolve polymorphism by means of type definitions processed at the compile time. It reduces the runtime overhead of polymorphism to the strict minimum while keeping a high-level of expressivity and readability.

Low-level parallel programming models like MPI, OpenMP and MPI+OpenMP are extensively employed for remote sensing image processing. Examples include data preprocessing [14], mosaic [15], disaster monitoring [16][17] and global changes. The aforementioned projects provide high-level pattern for parallel programming. However, these implementations did not develop their research for the massive remote sensing datasets with multi-band image data and complex structured metadata. Moreover, they did not handle the different dependencies between computation and data of RS algorithms. In this situation, it remains a big challenge to program effective massive remote sensing data processing algorithms productively.

The RS-GPPS skeletons proposed in this paper aim at addressing the above issues. They rely on the generic approach as QUAFF does to provide generic parallel algorithm skeletons for massive remote sensing processing. The RS-GPPS also develops its implementation for 1) massive RS data with complex data structure and 2) dependences between computation and data.

III. PROBLEM DEFINITION

This section presents the research issues related to the parallel programming for massive remote sensing image processing applications in a multicore cluster. There are three aspects of this problem: massive remote sensing data, difficulties of parallel programming and data processing speed. The remote sensing image processing applications are overwhelmed with tons of remote sensing images. The first issue is related to the large remote sensing image data with multi-dimensionality and complex metadata structures. The question is how to divide and communicate these data among distributed hierarchical cluster architectures (Section III-A). The second issue is related to the programmability of parallel remote sensing algorithms. These algorithms demand a great concern for parallel implementation details and the different dependencies between the computation and its associated remote sensing data sets (Section III-B). The third issue is
about data processing performance, namely how to sufficiently benefit from the multilevel hierarchical parallel architecture of multicore SMP cluster, data I/O and data locality (Section III-C).

A. Massive RS Data

The size of a typical remote sensing image dataset for real-time processing is several gigabytes. Global change and disaster monitoring applications are more likely to process large data sets of hundreds of terabytes, which are globally covered multi-temporal, multi-band remote sensing image data from multi-sensors. It is a research challenge to process such massive RS data to fulfill high performance QoS requirements, such as real-time processing.

With the rapid development of remote sensing technology, the remotely sensed images from multi-spectral even hyper-spectral sensor usually have hundreds of spectral bands, such as HJ-1A hyper-spectral imager produce 128 bands. The multi-band remote sensing images always lead to the pixel’s multi-dimensionality. On the other hand, the RS datasets consist of associated metadata including complex geographical information, which is always involved in the data processing procedure of algorithms. Accordingly, to define and save these RS datasets with multi-dimensional images and complex metadata structure in memory will be rather complicated. The complex data structure of RS data also complicates the RS data slicing and communicating across nodes of the cluster.

In specific, when the remote sensing image datasets are splitted into blocks, the geographical information associated with these blocks is required a recalculation. For example, the latitude and longitude of the four corners of a data block in a geographic coordinate system should be recalculated using projection parameters, geometric position of data block, and the region of the entire image. This process is rather compute-intensive. In most cases, RS data will have to be divided into partially overlapped blocks, as most algorithms require the dependence between computations and data: the computation of each pixel depends on its neighborhood.

The communication of the complex user-defined data types like RS image datasets across nodes are not efficiently supported by current MPI implementations. Programmers have to exchange complex metadata via repeated calling low-level MPI send/receive communication APIs, which significantly degrades the performance.

B. Programmability of Parallel RS data processing

With the process of widely usage of remote sensing images, there emerges a variety of algorithms of high complexity. The dependencies between the computation and associated RS data vary with different algorithms. Examples are data independent computation for pixel-based processing algorithms, region dependent computation for neighborhood-based processing algorithms and global dependent computation for global or irregular processing algorithms [18]. These dependences probably lead to different data parallelism, data computation modes, data/task partition strategies, and even complicated data access modes. Of course, these will make the parallel programming of RS algorithms more difficult. To develop code efficient parallel algorithms, programmers are required to understand detailed hierarchical architecture of multicore clusters. Thereafter they have to deal with both the message passing model for inter-nodes and the shared memory model for intra-node communication.

Fortunately, many algorithms express their parallelism with recurring parallel patterns. If the recurring patterns of parallel computation and communication can be abstracted and pre-implemented, they can be reused by a collection of algorithms. In this situation, easy parallel programming can be offered with a less concern of parallel implementation details.

C. Data Processing Performance

Processing of massive remote sensing data is compute-intensive. Especially time-critical applications like disaster monitoring pose a big challenge on data processing performance. Therefore, the parallelization of remote sensing algorithms should concern for how to take the parallelism of multicore into account and sufficiently benefit from the hierarchical parallel architecture of multicore SMP clusters. Furthermore, there remains a big gap between the performance of processors and I/O systems. Therefore to improve the I/O system performance and overlap the I/O operation with computation would also be critical important. In addition frequent data communication among nodes would also result in poor performance. It is thus critically important to pay special attention to data locality when developing and executing RS data processing programs.

IV. DESIGN AND IMPLEMENTATION

We have designed and implemented generic parallel skeletons for massive remote sensing image data processing problems (RS-GPPS). It can provide a more efficient and easy way to program remote sensing algorithms and deal with massive RS data in a multicore cluster. The RS-GPPS consists of several generic parallel algorithm skeletons for RS image processing (Section IV-A) and the generic RS data type for distributed remote sensing image data objects (Section IV-B).

![Fig. 1. The flow of generating parallel programs with RS-GPPS skeletons](image_url)
RS-GPPS is enabled by template class mechanisms. The data structure of RS image and user-defined data partition strategies are exposed as parameters of the RS data type templates. By specifying these parameters of templates, a distributed RS data object can be defined. Once the distributed RS data are declared, it will be divided into blocks with user-defined data partition strategies. The type parameters and member functions of these generic class templates are exposed to programmers as the interface of skeletons. As showed in Figure 1, the user-defined job class encapsulates the sequential code segments or functions as the parameter of the skeleton templates. These generic parallel algorithm templates then can be instantiated. Thereafter each process locally executes its sequential function respectively by calling the user-defined job class and the communications among nodes will also be implemented. Consequently, the low-level parallel programming details are shield from users. The parallel remote sensing programs can be easily developed based on sequential ones.

The system architecture of the RS-GPPS implementation is illustrated in Figure 2. The parallel architecture-specific details and the data slicing/distribution operations are pre-implemented in generic algorithm skeletons and distributed RS data templates respectively with MPI.

To solve the problem discussed in Section III-A, the remote sensing image dataset with multi-dimensional image data and complex metadata structures are wrapped as distributed remote sensing image data type templates, whose data blocks are distributed across cluster nodes. When the distributed RS data are declared, the entire RS dataset will be logically divided into blocks and then distributed across cluster nodes. The data slicing operation recursively splits the entire RS dataset into partially overlapped blocks with user-defined data partition strategies, and recalculates associated geographical information. Then the sliced data blocks are distributed to nodes according to the mapping relationship of physical data blocks and I/O nodes in the parallel file system. Thereafter the sliced data blocks can be loaded in local memory of each node. These memory spaces are exposed to all nodes for sharing and virtually converged to a global distributed RS data enabled by one-sided messaging primitives provided by MPI. Then programmers can easily access any blocks of the entire distributed RS data objects via one-sided MPI primitives.

To solve the problems discussed in Section III-B and Section III-C, we have designed parallel algorithm skeletons for massive remote sensing data processing algorithms with different computation modes. The recurring patterns of parallel computation and communication in skeletons are pre-implemented in a multicore cluster with MPI API. These pre-implementations include task partition strategies and data accessing modes consistent with different dependences between computation and data, parallel computing architecture dealing with multi-level parallelism of both inter-node and intra-node, and process synchronization. When a generic parallel skeleton is instantiated and declared, the computations on distributed remote sensing data objects are performed as follows:

1) The task will be divided into subtasks by a two-stage task partition strategy, which consistent with the data partition strategy;
2) Load the data blocks owned by each node concurrently through the parallel I/O operations;
3) The user-defined remote sensing sequential codes encapsulated in the job class will be executed in parallel by each process.

In this situation, developers will have a less concern for architecture-specific parallel implementation details.

### A. Generic Algorithm Skeletons for Remote Sensing Applications

The remote sensing image data are featured by the geometric and multi-band structures, which result in the inherent data parallelism of the RS algorithms. In general, the remote sensing data processing algorithms can be classified into four categories according to the different dependence between computation and data: 1) pixel-based processing algorithms with data independent computation, 2) neighborhood-based processing algorithms with region dependent computation, 3) band-based processing algorithms with band dependent computation, and 4) global or irregular processing algorithms with global dependent computation.

#### 1) Categories of Remote Sensing Algorithms

**Pixel-Based Processing** – this category of algorithms refers to those, which perform computation on single pixel and without the request for context. Typically, the computations of these algorithms are data independent and are of excellent data parallelism. This category includes arithmetic or logical operation on pixels of image, radiometric correction and pixel-based classification like maximum likelihood or SVM classifiers. Assume that $S$ for source RS image, $R$ for result RS image, $x$ and $y$ represents the geometric position of pixel $P$ in
image, \( b \) for the band number, \( f() \) for the computation performed on pixel \( P \). Then computation model of this category can be expressed as:

\[
R_{b,x,y} = f(S_{b,x,y})
\]

(1)

**Neighborhood-Based Processing** – this category of algorithms refers to those, which use the corresponding pixel and its close neighbors (regular window) in input image to calculate the value of a pixel in output image. This category includes image filter with convolution, image resampling and geometric correction etc. Here region \((S(b,x,y))\) represents the data region in neighborhood with pixel \( P \) in source image band \( b \). The computation model of this category can be expressed as:

\[
R_{b,x,y} = f\left( \text{region}(S_{b,x,y}) \right)
\]

(2)

**Band-Based Processing** – this category of algorithms refers to those that the computation of single pixel in output image should use the pixels in same position or its close neighbors in several input image bands. This category includes pixel-based fusion, image transformation, DNVI etc. Vector \((S(x,y))\) is for the spectral vector of pixels located in position \((x,y)\) of multiple image bands. Then computation model of this category can be expressed as:

\[
R_{b,x,y} = f\left( \text{vector}(S_{x,y}) \right)
\]

(3)

**Global or Irregular Processing** – The algorithms in this category are those that the irregular pixels or even the global image are required for calculating the value of a single pixel in output image or statistical features. These algorithms are commonly with poor parallelism. Assume that the \( S_B \) represents for image band \( b \), computation model of this category can be expressed as:

\[
R_{b,x,y} = f(S_B)
\]

(4)

2) **Generic RS Farm-pipeline Skeleton**

The Farm-pipeline Skeleton aims at the remote sensing data processing algorithms with excellent data parallelism, such as pixel-based processing algorithms, Neighborhood-Based Processing and band-based algorithms. However, these algorithms should not introduce geometrical warping of image. Thus, this skeleton template is applicable to a plenty of remote sensing applications.

Concerning for the multi-level parallel architecture of multicore clusters, a Farm-Pipeline two level parallel pattern is proposed (Figure 3) in this algorithm skeleton.

1) Inter-node: The master node is responsible for the task partition and assignment, as well as receiving the result data block for output. The slave nodes process the assigned RS data blocks \( A_i \) with user-defined function in parallel. In this pattern, the slave nodes work independently without data communication.

2) Intra-node: The data blocks \( A_i \) assigned to node are further split into sub-blocks \( A_{ij} \). These sub-blocks are then computed by processes of intra-node in parallel. In order to increase performance of I/O operations, the idea of “on-the-flow” processing is adopted: the data loading, multiple data processing, and data sewing processes are organized in a processing pipeline. When the pipeline is running, the sub-block \( A_{ij} \) is computed by several processes in nodes, simultaneously the sub-block \( A_{ij} \) is pre-fetched by data loading process. In this condition, the sub-block \( A_{i,j-2}, A_{i,j-1}, A_{ij} \) are processed in parallel. In this way, the intra-node parallelism can be sufficiently made use of and also the data I/O could be fully overlapped with computation. Figure 3 overviews the parallel pattern of Farm-Pipeline.

![Fig. 3. The Parallel Pattern of Farm-Pipeline](image)

As showed in figure 4, the RS Farm-pipeline Skeleton is wrapped and implemented as a generic algorithm template class RSFarmPipelineSkeleton with template class mechanism. The parallel architecture, common processing flow and parallel implementation details are pre-implemented in the template. The user-defined sequential code or function is encapsulated in the Job class which provide a virtual function interface void *operator () (void*). As the task partition operation in this skeleton is just a data split operation, so it will be done by the constructor of the distributed RS data template.

```c
template <class T, class Job>

class RSFarmPipelineSkeleton {
   protected: //definition of template functions
      void load(&RSBlock<T> block);
      RSBlock<T> = comp(RSBlock<T> block, Job job);
      zip(RSBlock<T> block);
   public: //parallel
      static void doWork(Job job, Dist_RSData<T> dist_A, Dist_RSData<T> &dist_B) {
         //code omitted
         //construct pipeline here with pseudo code
         Stage1: Load (dist_A(i,j));
         Stage2: dist_B(i,j-1)= job (dist_A(i,j-1));
         Stage3: dist_B(i,j)= Zip (dist_B(i,j-2));
         Pipeline(Load (dist_A(i,j), job (dist_A(i,j-1), Zip (dist_B(i,j-2))));
   }
```

Fig. 4. The Generic Definition of RS Farm-Pipeline Skeleton

The API of RS Farm-pipeline Skeleton is provided as static template member functions of RSFarmPipelineSkeleton class (Table 1).
Figure 5 gives an example parallel program for image resampling using RS Farm-pipeline Skeleton.

3) Generic RS Image-Wrapper Skeleton

The RS Image-Wrapper skeleton is designed for the algorithms including pixel-based processing algorithms, neighborhood-Based Processing and band-based algorithms. These algorithms will cause the geometrical warping of image data. Therefore, the irregular processing is supported in the skeleton.

![Image of Computation Mode of Image Partition](image.png)

In these algorithms (Figure 6) the calculation of a single pixel in output image requires the irregular region of input image, which is located in the position determined by a backward geometrical mapping. Typically, the input RS image data will follow a formula partition strategy. In this situation, as the local data block cannot meet the need for calculation, the extra data communications with other nodes are required. Consequently, how to express and get the irregular region of data needed before computation and avoiding extra frequently data communications poses a big challenge for developers.

To address the above issues, a default irregular data partition strategy is embedded in this generic algorithm skeleton. The mapf () function defines the geometrical mapping to calculate the data irregular data region for computing data Bi. The approximate () function uses the rectangular region Ai to approximating the irregular data region.

As showed in Figure 7, the RS Image-Wrapper skeleton is wrapped and implemented as a generic algorithm template class RSImageWrapperSkeleton. The data partition operates on input image A is put off to done by each worker. Each worker node calculates the exact data block Ai needed for computing the assigned data block Bi with mapf and slices Ai from image A. Thus, the frequent data communication incurred by the blind data partition at the beginning of the algorithm could be avoided. The user-defined sequential code or function is encapsulated in the Job class which provide a virtual function interface void *operator () (void*). And the backward geometrical mapping function is wrapped in the MapF class which provide a virtual function interface void *operator () (void*).

```c
template <class T, class Job, class MapF>
class RSImageWrapperSkeleton {
  protected:  //definition of template functions
    void load(&RSBlock<T> & block);
    RSBlock<T> & comp(RSBlock<T> & block, F & f);
    RSBlock<T> & load(block, F & f);
    public:  //parallel
      // Initialize problem B with mapf
      static RSData<T> initTask(MapF mapf, RSData<T> & A);
      RSData<T> & B = mapf(A);
      return B;
  }

  //parallel implemented in worker nodes
  static void doWork(F & f, F & map, RSData<T> & A, 
      RSData<T> & B) {
    //code omitted
    //construct pipeline here with pseudo code
    Stage1: A(i,j-1)=irregularDataSlice(A,mapf,B(i,j-1))
    Stage2: load(A(i,j));
    Stage2: dist_B(i,j)=job(A(i,j-1));
    Stage3: dist_Bi=Zip(dist_B(i,j-2));
    Pipeline(irregularDataSlice(A,mapf),B(i,j-1)),Load(A(i,j),
      job(A(i,j-1)),Zip (dist_B(i,j-2)));
  }
}
```

![Image of Computation Mode of Image Partition](image.png)

**B. Distributed RS Data Templates**

A remote sensing image dataset always consists of complex metadata and multi-band images whose pixels are of multi-dimensionality. The metadata is the data for self-description, such as image information, geographical information and satellite & sensor information. The geographical information is involved in the data processing procedure of algorithms. To facilitate the using of the RS data, we abstract the entire remote sensing image dataset with images and metadata into a remote sensing data object named RSData.

1) RSData Templates

A RSData object consists of T and P. T represents the images with multi-band structure, which stores the spectral information and geometrical location of each pixel. P represents all the metadata. The multi-band images of RSData are expressed via a three-dimensional matrix. For a RSData with k band of images in size of m rows and n samples, T is expressed as Equation 5.
The metadata \( P \) includes image information, geographical information and satellite & sensor parameters. The image information contains the image size, data type, data arranged order and etc. The geometrical information includes the latitude and longitude of the four image corners in a geographic coordinate system, the \( x \) and \( y \) of the four image corners in a projected coordinate system, the projection parameters and the ground resolution of pixel. In addition, the satellite and sensor information includes the satellite orbital parameters and sensor parameters. Regarding that the RS data from different satellites or sensors, the sensor parameters varies and metadata are always organized in different forms. Furthermore, the parameters of different projection methods differ from each other. Thus, we chose the standard projection string of GDAL to express the projection parameters. The similar strings are provided for expressing sensor parameters, in order to adapt to different images from different sensors. The RS data object is wrapped as a generic algorithm template class \( \text{RSData} \) (shown in figure 8).

The distributed \( \text{RS} \) data object \( \text{dist}_A \) is expressed in Equation 6 and 7. \( A \) is for a normal \( \text{RS} \) data, \( \pi (A) \) is a set of sliced blocks \( A_n \), and \( \text{map} (\pi (A)) \) is the set of node mapping of blocks.

\[
\pi (A) = \{ A_1, \cdots, A_n \}, \text{map} (\pi (A)) = \text{map}_f[A_1, \cdots, A_n] \tag{7}
\]

\[
\text{dist}_A = A \cup \text{map} (\pi (A)) \tag{6}
\]

2) \( \text{Dist}_\text{RSData} \) Templates

A remote sensing image dataset whose sliced data blocks are scattered across nodes is abstracted and wrapped by the generic distributed \( \text{RS} \) data class named \( \text{Dist}_\text{RSData} \). Programmers do not need to concern for the details about data partition and node mapping, the distributed \( \text{RS} \) data are operated as a local \( \text{RS} \) data just like \( \text{RSData} \).

The distributed \( \text{RS} \) data object \( \text{dist}_A \) is expressed in Equation 6 and 7. \( A \) is for a normal \( \text{RS} \) data, \( \pi (A) \) is a set of sliced blocks \( A_n \), and \( \text{map} (\pi (A)) \) is the set of node mapping of blocks.

\[
\pi (A) = \{ A_1, \cdots, A_n \}, \text{map} (\pi (A)) = \text{map}_f[A_1, \cdots, A_n] \tag{7}
\]

\[
\text{dist}_A = A \cup \text{map} (\pi (A)) \tag{6}
\]

The distributed \( \text{RS} \) data object \( \text{dist}_A \) is expressed in Equation 6 and 7. \( A \) is for a normal \( \text{RS} \) data, \( \pi (A) \) is a set of sliced blocks \( A_n \), and \( \text{map} (\pi (A)) \) is the set of node mapping of blocks.

\[
\pi (A) = \{ A_1, \cdots, A_n \}, \text{map} (\pi (A)) = \text{map}_f[A_1, \cdots, A_n] \tag{7}
\]

\[
\text{dist}_A = A \cup \text{map} (\pi (A)) \tag{6}
\]
Generic parallel programming skeletons RS-GPPS have successfully employed in the Parallel Image Processing System for remote sensing (PIPS). Dozens of remote sensing data processing algorithms are easily parallelized with these parallel programming skeletons and integrated into system, such as fine correction, NDVI, image registration, fire detection and image classification etc. The fine correction algorithm was implemented on a BJ-1 panchromatic image data with size of 48343 rows and 19058 columns. It uses CC (Cubic Convolution) resample mode, MQ model and cubic polynomial mapping. The implemented result of this algorithm is shown in Figure 12.

![Fig. 12. The Result of Fine Correction algorithm](https://example.com/fig12.png)

The RS-GPPS wraps generic algorithm skeletons for remote sensing data processing and distributed data type templates for RS data. These skeletons offer excellent programmability. We use the SLOC [19] (source lines of code, the comments and empty lines excluded) as a quantitative metrics for the programmability.

Two algorithms of fine correction (RS Image-Wrapper Skeleton) and radiation correction (RS Pipe-line Skeleton) are experimented. The results of code generation are listed in table II. We can see that extra codes of hundreds or thousands lines are required for programming without RS-GPPS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SLOC With Skeletons</th>
<th>SLOC Without Skeletons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine Correction</td>
<td>386</td>
<td>1414</td>
</tr>
<tr>
<td>Radiation Correction</td>
<td>82</td>
<td>554</td>
</tr>
</tbody>
</table>

Communication of remote sensing image datasets with complex metadata structure across nodes is rather inefficient. Traditionally, the plenty of metadata are transferred with MPI send/receive primitives one by one. In this way, the transferring of one RS data will incur repeated MPI communication operations. We have provides a data serialization operation in data type templates of RS data. It serializes complex metadata structures into a string of characters. Accordingly, when the data are serialized, the transferring of RS data would be done by calling the MPI communication operation only once. Assuming that the metadata contains n data items, the overhead for data serialization is $t_s$, and mean overhead for transferring one data item is about $t_c$ seconds. As the total data amount of metadata is small enough (couple KB), the time overhead for transferring serialized data would probably be $k \cdot t_c$ (1 < k < 2). Therefore, the time overhead for transferring in traditional way would be $n \cdot t_s$, but if it is done in our way, the overhead would just be $t_s + k \cdot t_c$. Considering that the data amount is small, the overhead for data serialization $t_s$
would be far less than \((n - 2) \ast tc\). Therefore, data communication performance is much improved.

The performance experiments were conducted on a multicore cluster with 12 nodes connected by a 20 Gigabyte Infiniband using the RDMA protocol. Each node is a blade server with dual Intel(R) Quad core CPU (3.0 GHz) and 8GB memory. The operating system was CentOS 5.0, the C++ compiler was the Intel C/C++ Compiler with optimizing level O3, and the MPI implementation was Intel MPI. The remote sensing data processed by algorithms is a BJ-1 panchromatic image data with size of 48343 lines and 19058 columns.

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>PERFORMANCE OF PARALLEL ALGORITHMS (CHANGE NODE SCALE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes (Cores)</td>
<td>FC (s)</td>
</tr>
<tr>
<td>1 (1)</td>
<td>696.708</td>
</tr>
<tr>
<td>1 (6)</td>
<td>83.374</td>
</tr>
<tr>
<td>2 (16)</td>
<td>37.982</td>
</tr>
<tr>
<td>3 (24)</td>
<td>29.065</td>
</tr>
<tr>
<td>4 (32)</td>
<td>20.775</td>
</tr>
<tr>
<td>5 (40)</td>
<td>18.469</td>
</tr>
<tr>
<td>6 (48)</td>
<td>15.796</td>
</tr>
<tr>
<td>7 (56)</td>
<td>17.903</td>
</tr>
<tr>
<td>8 (64)</td>
<td>13.667</td>
</tr>
<tr>
<td>9 (72)</td>
<td>17.288</td>
</tr>
<tr>
<td>10 (80)</td>
<td>19.4</td>
</tr>
</tbody>
</table>

The performances of Fine Correction (FC) and Radiation Correction (RC) with and without skeletons (RC-n, FC-n) are experimented respectively. The experimental results with increasing number of computing nodes (scale from 1 to 10) are listed in Table III. The runtime and speedup metric with increasing numbers of computing nodes are also illustrated in Figure 13 and 14 respectively. From the experimental results we can see that the time overhead of the algorithms with skeleton are increased by less than 5% when compared to those manual implemented ones. It concluded that the skeleton implementations introduce slight performance degrade.

![Fig. 13. Run time of algorithms](image)

![Fig. 14. Speedup of algorithms](image)

With the amount of RS data increased from 1.3 GB to 507GB, the experimental results of above algorithms FC and RC are listed in Table IV. The runtime performance metrics is illustrated in Figure 15.

![Fig. 15. Performance of Algorithms with Increasing Data Amount](image)

VI. CONCLUSION

Massive remote sensing data processing introduces great challenges for researchers of programmers. As described in this paper, RS_GPPS is proposed to provide 1) program templates for RS data distribution and 2) generic parallel skeletons for RS processing algorithms. The distributed RS data templates offer an easy and efficient way to distribute the massive remote sensing image data across cluster nodes via one-sided memory access. The recurring patterns of computation and communication are pre-implemented in generic parallel skeletons. Only by template instantiation the programs can be developed as sequential ones with a minimum concern for architecture-specific parallel details. The experimental results show that SLOC metrics of skeleton-based programs are greatly reduces. The overhead penalty caused by RS_GPPS is less than 5% compared to manual implantation versions. In addition, these skeleton-based programs also perform excellent scalability with increasing massive amount of data. It is concluded that the generic parallel skeletons provided in this paper is productive and efficient.

VII. ACKNOWLEDGMENT

Dr. Lizhe Wang is supported by "One-hundred talent" program of Chinese Academy of Sciences. Dr. Jun Wang's work in this paper is supported in part by the US National Science Foundation Grant CCF-0811413, CNS-1115665, and National Science Foundation Early Career Award 0953946.

REFERENCES


[17] Lingjun Zhao, Yan Ma, Guoqing Li, Wenyang Yu, Jing Zhang: Rapid Calculation Research on Water Area Extraction from ASAR Image. GCC 2009: 339-343
