Parallel Computation of Synthetic SAR Raw Data

M. Kalkuhl, P. Droste, W. Wiechert
Department of Simulation
University of Siegen
Paul-Bonatz-Strasse 9-11
D-57068 Siegen, Germany
marc.kalkuhl@uni-siegen.de

H. Nies, O. Loffeld
Center for Sensorsystems (ZESS)
University of Siegen
Paul-Bonatz-Strasse 9-11
D-57068 Siegen, Germany
nies@zess.uni-siegen.de

M. Lambers
Institute for Vision and Graphics
University of Siegen
Hölderlinstraße 3
D-57068 Siegen, Germany
martin.lambers@uni-siegen.de

***Abstract*—For modern SAR data acquisition, bi- and multistatic SAR missions become increasingly important. Established methods for processing monostatic SAR signals need to be adapted to new algorithms for signal processing. In order to support the evolution and development of these new algorithms simulated SAR raw data of arbitrary bi- and multistatic SAR scenarios are essential. This paper refers to a modular SAR simulator, which is able to simulate complex bi- and multistatic SAR scenarios. It focuses on the geometrical simulation approach of the simulator and the computationally intensive synthesis of SAR raw data. The main part descripts the parallel implementation of the radar lobe footprint scan and of the succeeding SAR raw data generation executed on a compute cluster. An example will be shown, which compares the simulation of the same SAR scenario on three different compute systems and their runtimes.**

I. INTRODUCTION

New complex bi- and multistatic SAR methods are currently being developed because they promise improved precision and more facilities in contrast to established monostatic ones. During the evolution of new signal processing algorithms for these constellations test data is often not available, since corresponding real SAR experiments [1] are still not accessible or too expensive. In this case a simulation tool can provide test data like in [2]. Such a SAR simulator for arbitrary mono-, bi- and multistatic SAR constellations was presented in [3]. The basic philosophy of this SAR simulator is to supply a freely configurable framework with different components that can be used for planning real SAR missions and for providing SAR raw data to test newly developed signal processing algorithms. While established SAR simulators are often developed for one special flight constellation, the modular simulator is able to deal with scenarios of arbitrary complexity (see Figure 1). To satisfy the requirements of these SAR scenarios in a flexible way, the simulator was realized in a modular way. By assembling different modules from a set of building blocks (e.g. satellite, plane, SAR sensors, GPS, earth surface) a huge variety of scenarios can be configured. Additionally, to improve the benefit of the simulator, different surface models are eligible in order to account for backscattering behaviour and other diverse effects like *shadowing*, *layover* and *foreshortening*. Besides a synthetic curved surface (e.g. a single point target or a defined array of point targets) a realistic earth surface model, based on height data of the SRTM mission in the year 2000 [4], can be selected in different resolutions. Due to the fact that the simulation run time drastically increases corresponding to the resolution of the data, it is necessary to find approaches to reduce the simulation time. Distributed computation is very promising and is the topic of this paper.

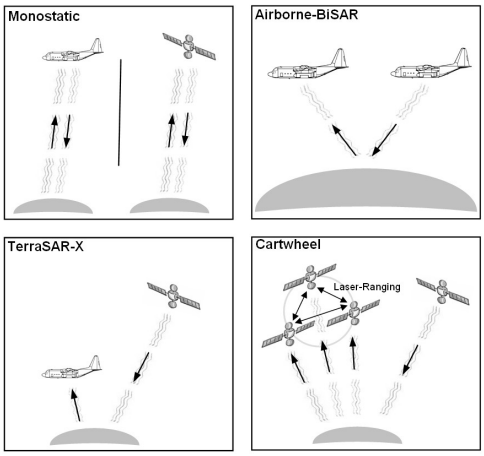

Figure 1. Selection of possible mono-, bi- and multistatic SAR scenarios

II. GEOMETRICAL SIMULATION APPROACH

In order to simulate a complex scenario and to generate the corresponding SAR raw data the specification of the signal runtime and the signal path length between transmitter and receiver is crucial. This suggests a geometrical approach (See Figure 2). Starting from an inertial coordinate system (IS), the whole scenario is set up with all its modules as described before. In addition to the IS, several other coordinate systems are used to define the positions of the different modules or to define other important dependencies. (E.g. Earth Coordinate System ECS, Trajectory System TS, Satellite System SATS and some others) Each coordinate system can be converted into another one. Due to this concept, it is possible to calculate all geometric quantities (e.g. distances between two modules) and positions required in the scenario in one reference coordinate system (mostly IS).

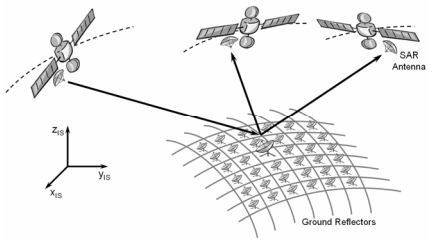

Figure 2. Geometrical scenario setup

The computational primitive to calculate the SAR signal response is the evaluation of the distance or runtime, respectively, between a SAR sensor and a point target. All further computations rely on this basic operation. For this reason the backscattering earth‘s surface is discretized into a grid, where each ground cell represents a reflector or a point target, respectively, which reflects the SAR signals (See Figure 2). Each ground cell has a fixed defined coordinate to reference its position to enable the calculation of the mentioned distances. Furthermore additional information from each cell is required to perform the simulation:

- Is the reflector in the footprint area?
- Is the reflector farther?
- Is the reflector in a radar shadow?

Each of these three questions has to be answered for each reflector, for each transmitter/receiver footprint and for each time step of the simulation. With every refinement of the surface grid, the number of cells/reflectors to be checked increases drastically. For this reason effective scan methods are required.

The overall scan of the complete surface delivers all reflectors or point targets respectively, which were illuminated by one or more radar lobes. The intersection between transmitter and receiver cells enables the determination of these ground cells, which are able to reflect the incoming signal from the transmitter to the receiver. With the geometrical distance between each SAR sensor and the illuminated point target the complex radar signal can be calculated.

III. COMPUTATIONAL COMPLEXITY OF THE SCAN

As seen in the preceding paragraph the scan of the earth’s surface is one of the most time consuming parts of the simulator. To get an impression of the number of surface cells two examples are chosen:

A “coarse” surface with a resolution of approximate 1100 m by 1100 m consists of approximately 270 cells using a footprint dimension of 3 km by 30 km of a satellite (typical TerraSAR-X footprint size). The scan duration of such an area lies at 1-3 sec with an AMD Opteron 2 GHz. For the same size of footprint a “fine” surface resolution of 10 m by 10 m ends with about 1.6 million cells and needs 80-200 sec with the same processor. Down to a resolution of 1 m by 1 m there are approximately 160 million cells to scan. At the same time the memory requirements also increases drastically, because all cells are represented by objects and have to be managed in the memory. Not only the number of scanned cells is one of the crucial things but also the number of used SAR transmitters and receivers in the scenario, because they increase the number of overall scans. For example a simple scenario (e.g.: one point target, a single SAR transmitter and receiver, short flight path of five seconds and with a pulse repetition frequency of 3 kHz) with a resolution of 11 km by 11 km can still be calculated on a single computer in approximately 5 minutes. In contrast, a “fine” resolution (real surface grid with only a resolution in the order of 10 m by 10 m) would take about 30 days. Reasons are the footprint scan and the calculation of the SAR raw data depending on the surface resolution.

The goal is to scan the surface as effective as possible. As seen before three main criteria arise for the determination of the illuminated cells, which have to be satisfied for each of them. In case of fulfilment these cells are accepted for the further calculation.

A. *Footprint check*

The first criterion mentioned above checks if a ground cell lies in the footprint area. As seen before a rectangular grid with fixed resolution is used to discretize the earth‘s surface to position the point targets. In contrast another possible approach to discretize the radar lobe itself like a cone of light consisting of individual rays would yield an irregular resolution of the footprint area. To check if a ground cell is illuminated a new coordinate system (SAR-LobeSystem - SLS) for each radar antenna is introduced (cf. Figure 3). This coordinate system allows defining the radar lobe with only two angles. The position of the checked ground sensor can be transformed with the coordinate systems mentioned before to the related SLS. Subsequently the determined angle of the ground cell position is compared with the angle of beam spread. If the calculated angles fit to the radar beam angles, the cell lies inside the illuminated footprint area.

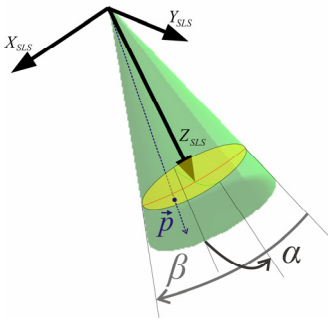

Figure 3. A new coordinate system: SAR LobeSystem (SLS)

B. *Reflection*

The second criterion for an illuminated target selects if a ground cell is faced to the antenna so that it reflects the incoming signal. The normal vector of the ground cell and the direction vector of its base have to satisfy the following geometrical condition (c.f. Figure 4):

$$90° < \alpha < 180°.$$

Each rectangular ground cell and its orientation (normal vector) are defined by the height of the four vertices of the related ground cell.

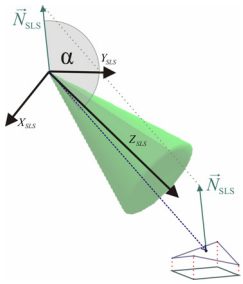

Figure 4. Detection of the facing grid cells

C. *Radar shadow*

The last criterion gives information about radar shadows. To detect them an assumption is made: The target in the perpendicular base or Nadir, respectively, is never in a shadow. Starting from this target the algorithm scans the whole footprint in a spiral way and determines the angle to the perpendicular. If the angle increases in

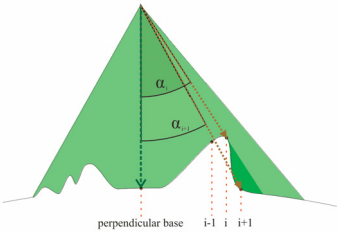

Figure 5. Scan mechanism to detect radar shadows

reference to the previous target the cell lies not in a shadow and vice versa. (See Figure 5) Here it is very important to obey the right scan sequence. (cf. Figure 6) This consideration is a special case of the geometry. For further details of the general case and the right scan sequence see [5]

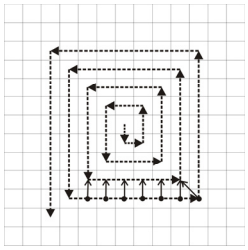

Figure 6. Shadow detection: spiral scan sequence (dashed line), angle comparison with previous related cell (solid line)

IV. Parallelized Synthesis Of SAR Raw Data

To perform a raw data computation in acceptable time a parallel approach on a compute cluster is advantageous and has been implemented in this simulator. Hence, it is an essential fact to find an approach, which allows to calculate single parts of the SAR raw data matrix on a single node of a compute cluster, with the assumption that the raw data matrix can be reassembled without shifts or time delays within the matrix for the whole flight path. A further requirement occurring from this geometrical approach is an effective data management, to avoid huge amount of geometrical data.

The basic idea for the parallelisation is to divide the azimuth time axis or the flight path, respectively, into different single parts and distribute them to the different computation nodes (See Figure 7). To implement this strategy some different approaches are possible, which will be discussed in the following paragraphs.

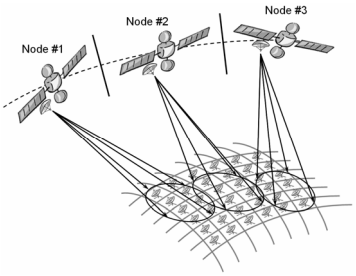

Figure 7. Parallelization approach: Trajectory is divided into three parts and distributed to the different computation nodes.

To understand the background of the different parallelization approaches it is important to understand the architecture of the used High-Performance Cluster (HPC) *Rubens*. The whole management is done by a job queuing and management system on a special management node. Only this node can communicate to all computation nodes and so it is required to include this node in the overall parallel architecture.

A. Brute force parallelization

The following approach represents the standard parallel procedure. Starting from a master process on the manage node of the cluster a controller process and respective numbers of workers are initialized. The master handles the network communication between controller and worker. (See Figure 8) The controller loads the simulation file, which contains all states of the simulation, and corresponding parameter files. It divides the simulation file into the given number of trajectory parts, distributes them to the workers and receives all calculated signal runtimes. The controller collects all these data and stores it on the hard disk. From all signal runtime data the minimum runtime and the size of the overall SAR raw data matrix respectively is determined by the controller. With this information each signal runtime package is send again to the workers on the same way as the simulation files are done to calculate the part of the SAR raw data matrix. Each finished raw data package is sent back to the controller, who composes the whole SAR raw data matrix from the parts. Additionally all requested parameter and data files (e.g.: like positions and velocities of the platforms and antennas) are arranged from the simulation file and stored on the hard disk.

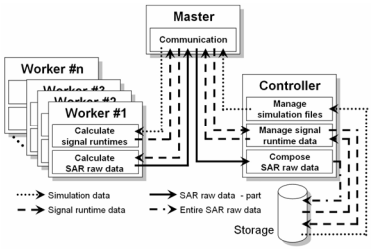

Figure 8. Schematic functionallity of the brute force parallelization

Clearly, this approach is easy to realize and serves as the reference implementation for the others. The major disadvantage lies in the high network load: a lot of high volume data packages have to be exchanged between the controller and the computation nodes. The restricting factor of this implementation is the raw data calculation algorithm. In this version it is not able to calculate the matrix dimension before the simulation. So the raw data matrix can only be calculated if the minimum runtime of all signals is known.

B. Controlled parallelization

The previous approach can be improved by an enhancement of the overall raw data generation algorithm, in terms of calculating the size and the start index of the raw data matrix at the beginning of each simulation from each single worker by its own. The general procedure of the simulation is the same described in section "A" except for the transfer and storage of the signal runtime files (cf. Figure 9). The SAR raw data can be calculated directly from the signal runtimes by the worker and the calculated data is sorted into the SAR raw data matrix in the correct way. Only the SAR raw data matrix is sent back to the controller to arrange all parts of the matrix to the entire SAR raw data.

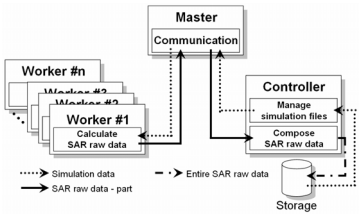

Figure 9. Schematic functionallity of the controlled parallelization

In contrast to the first approach from section "A" this approach is an enhancement. The network load and the data volume are reduced drastically, because only the simulation files and the results are exchanged between controller and worker. Therefore long read/write operations of temporary files are avoided.

C. Autonomous parallelization

The last parallelization approach is useful if the load of the cluster is very high and the job management system is not able

to reserve the requested number of workers. It is a simplification of the one presented before (See Figure 10). First of all, on the management node of the cluster a shell script is started manually, which produces two scripts: one for the simulation itself and one for the subsequent post process. The simulation script contains all job calls of all needed calculations. Each calculation is autonomous and can be handled by one single computation node. By calling this script, all jobs are transferred to the job management system of the cluster, which distributes all jobs to the single computation nodes depending on the load of the cluster and nodes. Each job stores its result in a specified directory. After all started jobs are finished and all data files are stored in the directory, the post process script can be started. This script starts a job to arrange all result files to the desired SAR raw data matrix and all parameter and data files.

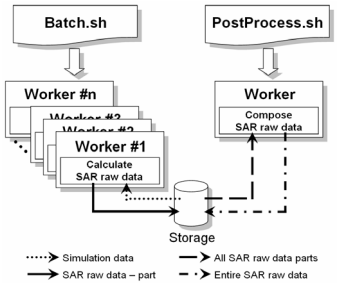

Figure 10. Schematic functionallity of the automomous parallelization

A disadvantage of this approach can be found in the not fully automated simulation procedure. Some manual steps are required to perform the full simulation as seen above. Additionally a higher data volume has to be accepted, because all nodes have to save all these data, which is needed to setup the entire SAR raw image and all required parameters or data files for the SAR processing in the end of the simulation. Compared to the disadvantage the benefit of this approach lies in the minimal network communication. The whole data needed for the simulation is taken from the hard disk. Also the results are written to the disk. Additionally, the use of this approach ensures a simulation when the cluster is under high load. Single jobs can be performed whenever a node is not used.

V. EXAMPLE

To get an impression of the speedup by using the parallel distribution a test scenario is chosen. The scenario is based on the TerraPAMIR mission [1]. The scenario consists of a satellite with a SAR transmitter/receiver and an airplane with a receiver. The resolution lies at 550 by 550 meters for a flight path of 3 sec. Three different platforms were used to simulate this scenario. First of all a simple workstation (32-bit system) with an AMD Athlon XP 2500+ (1.8 GHz) and 3 GB memory is used. The second test platform is a high performance workstation (64-bit system) equipped with an Intel Core2 Extreme QX6700 Quadcore (each core with 2.666 GHz) and 8 GB memory. Finally the simulation is performed on the "Rubens Cluster" (64-bit system) of the University of Siegen. The compute cluster consists of 148 nodes each equipped with two AMD Opteron-64bit-processors (2-2.8 GHz) and 2-4 GB memory. For the simulation on the cluster altogether 100 processors were used and the parallelization approach "B" is chosen. Each simulation on the three different platforms is performed with the same setting to compare the runtime results.

The simple workstation serves as a reference and needs for the chosen scenario 18 hours and 42 minutes. In contrast the high performance workstation needs 47 min to perform the simulation and is 23 times faster than the reference. The high gain between the high performance workstation and the simple one is remarkable and can be explained by different clock frequencies, system architectures (32 and 64-bit system) and Java versions. The cluster completes the simulation in only 7 min and executes the simulation 160 times faster than the simple workstation. As seen above the same reasons for the performance enhancement can be mentioned. In comparison to the high performance workstation the improvement is not proportional to the number of used nodes because of the required network communication and administration of the compute cluster.

VI. SUMMARY AND OUTLOOK

The design of a new modular bi- and multistatic SAR simulator is based on a geometrical simulation approach. Associated with this approach a problem occurs that a complex simulation can take a long, non acceptable runtime effected by the surface scan to determine all illuminated ground cells of the reflecting surface. In order to afford such simulations a parallelized implementation of the simulator for a compute cluster of the radar lobe footprint scan and of the succeeding SAR raw data generation was realized. Three different approaches were shown, where approach "B" and "C" are the best choice depending on the load of the used compute cluster. At the end the analysis of runtimes on different systems illustrates the great potential of reducing the overall simulation time by the parallel computation of the SAR raw data.

To improve the parallelization and the whole simulation some approaches are conceivable. As seen before a huge data volume is handled. To avoid conflicts during the network communication or data storage process the development of a new data management or new data structure would be desirable. Also an enhancement of the parallelization approach is applicable: the spiral scan algorithm could be segmented into 4 parts and treated as autonomous threads. Probably this would work well for large footprint areas.

VII. ACKNOWLEDGMENTS

This project is supported by a grant (WI 1705/9-1) from the German Research Foundation (DFG).

REFERENCES

[1] U. Gebhardt, O. Loffeld, H. Nies, K. Natroshvili, and S. Knedlik, "Bistatic Space Borne / Airborne Experiment: Geometrical Modeling and Simulation" in IGARSS 06, Denver, Colorado, USA, Aug. 2006

[2] H. Nies, O. Loffeld, M. Kalkuhl, and K. Natroshvili: "The bistatic aspect of the TanDEM-X mission" in IGARSS 2007, Barcelona, Spain, Jul. 2007

[3] M. Kalkuhl, P. Droste, W. Wiechert, H. Nies, and O. Loffeld "Modular SAR simulator for bi- and multistatic constellations" in IGARSS 2006, Denver, Colorado, USA, Aug. 2006

[4] Void-filled seamless SRTM data V1, 2004, International Centre for Tropical Agriculture (CIAT), available from the CGIAR-CSI SRTM 90m Database: http://srtm.csi.cgiar.org.

[5] P. Droste: "Development of an Object-Oriented Framework for Simualtion and Visualization of SAR-scenarios", Diploma Thesis, University of Siegen, March 2006, in German.