Optimal Configuration of Compute Nodes for Synthetic Aperture Radar Processing

Jeffrey T. Muehring and John K. Antonio

Deptartment of Computer Science, P.O. Box 43104, Texas Tech University,
Lubbock, TX 79409-3104
{muehring, antonio}@ttu.edu

Abstract. Embedded systems often must adhere to strict size, weight, and power (SWAP) constraints and yet provide tremendous computational throughput. Increasing the difficulty of this challenge, there is a trend to utilize commercial-off-the-shelf (COTS) components in the design of such systems to reduce both total cost and time to market. Employment of COTS components also promotes standardization and permits a more generalized approach to system evaluation and design than do systems designed at the application-specific-integrated-circuit (ASIC) level. The computationally intensive application of synthetic aperture radar (SAR) is by nature a high-performance embedded application that lends itself to parallelization. A system performance model, in the context of SWAP, is developed based on mathematical programming. This work proposes an optimization technique using a combination of constrained nonlinear and integer programming.

1 Introduction

This work focuses on modeling and optimizing the processor-memory relationships of an embedded system for synthetic aperture radar (SAR) processing. The hardware computing platform of investigation is one constructed with commercial off-the-shelf (COTS) components that are based on daughtercards and the compute-node concept. A daughtercard consists of one or more compute nodes, where a compute node is defined as an entity consisting of one or more processors, a block of shared memory, and the requisite glue logic. Within the framework of the models developed, optimization is performed on parameters such as the convolution section size and the choice and number of daughtercards comprising the system.

Size, weight, and power (SWAP) constraints often motivate the maximization of performance density for a given SAR system, especially in the case of unmanned aerial vehicles (UAVs) or satellites, which often accommodate SAR systems. SAR in itself is an approach to densifying a radar system by substituting a large degree of data postprocessing for radar equipment with prohibitively high size, weight, and power characteristics. Minimizing the power consumption of the compute platform used for

SAR processing is the fundamental objective in this research (although with sufficient parameter guidelines, size and weight could also be minimized using the same approach).

2 Fundamentals of SAR Processing

The specific mode of SAR investigated in this research is known as *stripmapping*. In stripmapping, successive radar pulses are transmitted and returned in the range dimension, which is orthogonal to the line of flight. Each received series of pulses from an individual transmitted pulse is then convolved with a reference kernel to achieve range compression. The entire range dimension is processed at once in this way. Detailed coverage of SAR and SAR processing is available in such works as [1, 2].

To create a two-dimensional SAR image, processing in the azimuth dimension is also necessary. The azimuth dimension is parallel to the line of flight and is conceptually infinite in length. Thus, processing of the entire azimuth vector, created from stacked range-processed vectors, is infeasible. To counter this problem, sectioned convolution is employed.

Sectioned convolution extracts a piece (or section) of the azimuth vector, convolves it with a reference kernel as in the range dimension, and then discards a portion of the result equal to the length of the reference kernel. Successively processed azimuth sections are then overlapped (with overlaps equal to the discarded kernel length) to form continuous vectors in the azimuth dimension. As is intuitive, a large azimuth section length requires more memory than a small section. Correspondingly, small azimuth sections require more total processing than do large sections because the percentage of new data processed, which is not discarded, is low (the size of the reference kernel being fixed).

A key point in this work is the exploitation of the section size and the concomitant processor-memory tradeoff [3]. Different daughtercards are better suited for different scenarios depending on the memory per processor ratio associated with the daughtercard, which is largely dependent on the chosen section size. The combination of the choices for the section size and number and types of daughtercards employed greatly affects the overall performance and associated power consumption of the computational platform.

3 Optimization Models

Two models are presented in this work, which address the problem of determining the optimal parameter values for configuring the system. Both methods are based on mathematical programming, which provides a method of formulating an optimization problem given an objective and set of constraints [4, 5]. This work proposes optimization techniques using a combination of constrained nonlinear and integer programming.

The first model is based on the assumption of an ideal shared-memory system. It treats all the memory contributed by individual daughtercards as a conglomerate block, equally accessible by all processors located on all daughtercards. For a system that is tightly predicated on the compute node with relatively high penalties for inter-compute-node communication, this is an inaccurate oversimplification. However, it is useful to initially investigate the optimization of the SAR system based on such an assumption because it provides clear insight into the interrelationships between variables and the effects of perturbation of other external parameters. In addition, without constraints on the amount of local memory available to a processor, the ideal memory-per-processor ratio can be derived from the optimization solution.

The second model removes the assumption of global shared memory and purposes to address system configuration more realistically. With this goal comes an increase in the complexity of the optimization formulation. The constraint set is modified to ensure only local memory access by processors. To accomplish this optimization, a much higher degree of integer programming is required than in the first model, entailing greater computational intensity to perform the optimization. The benefits of this second model include solutions that consist of a complete specification of how system resources are to be utilized, whereas the first model only specifies which resources are to be employed.

Parallelization of SAR processing involves the allocation of system resources for either range or azimuth processing [6]. In the first model, range and azimuth processors and memory are treated as aggregate requirements that somehow must be met with an appropriate number of daughtercards of each type. The second model, however, specifies how many processors and how much memory on each compute node per daughtercard is allocated for each function to prevent remote memory access during computation. Note that a single compute node can perform both range and azimuth processing, although each processor within a compute node must be dedicated to a single task.

4 Numerical Studies

Test data is based on the availability of two different daughtercards. The first is comprised of two compute nodes. Each compute node on this daughtercard consists of three processors and a shared memory block of 16 MB. The second daughtercard consists of a single compute node with two processors and 64 MB of memory. The first daughtercard consumes 12.2 watts of power and the second 9.6 watts. Throughput data for the significant operations involved in SAR processing is based on SHARC processors [7].

MATLAB’s **constr** function in the Optimization Toolbox was used to solve the nonlinear constrained programming problem presented by both models. The nonlinear nature of the problem results from the equations that express the required system memory and number of processors, which are derived in [8]. The **constr** implements a Sequential Quadratic Programming algorithm [9]. Integer programming, the need for which results from the inherently discrete number of processors per compute node and total compute nodes in a system, is implemented by multiple optimizations over the feasible discrete permutations.

Figs. 1 through 3 illustrate the result of solving the optimization problem of one of the models many times across a range of values for different platform velocities and desired resolutions. In each case, the platform velocity ranges from 50-400 m/s and the resolution from 0.5-2.0 m.

The utility of optimization of the section size is demonstrated by comparison of results produced by a heuristic used to determine section size, which defines the section size to be equal to the kernel size. This section size definition and resultant system configuration is designated as *nominal*. This work finds that the nominal section size, although relatively efficient in processing, is too large for most scenarios because of the excessive memory requirements involved. The optimizations performed show that forcing relatively inefficient processing with an associated reduction in memory requirements is optimal if power is to be minimized. Optimal section sizes thus often are found to be only a fraction of the kernel size, entailing the processing of more old data that is to be discarded than new data.

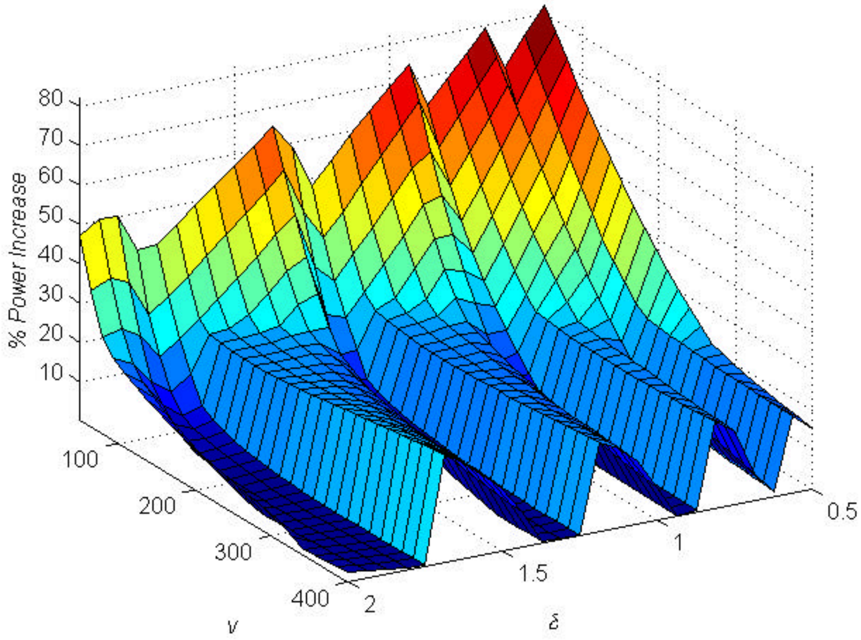

Fig. 1. Ratio of power consumption of the nominal section size to the optimal section size.

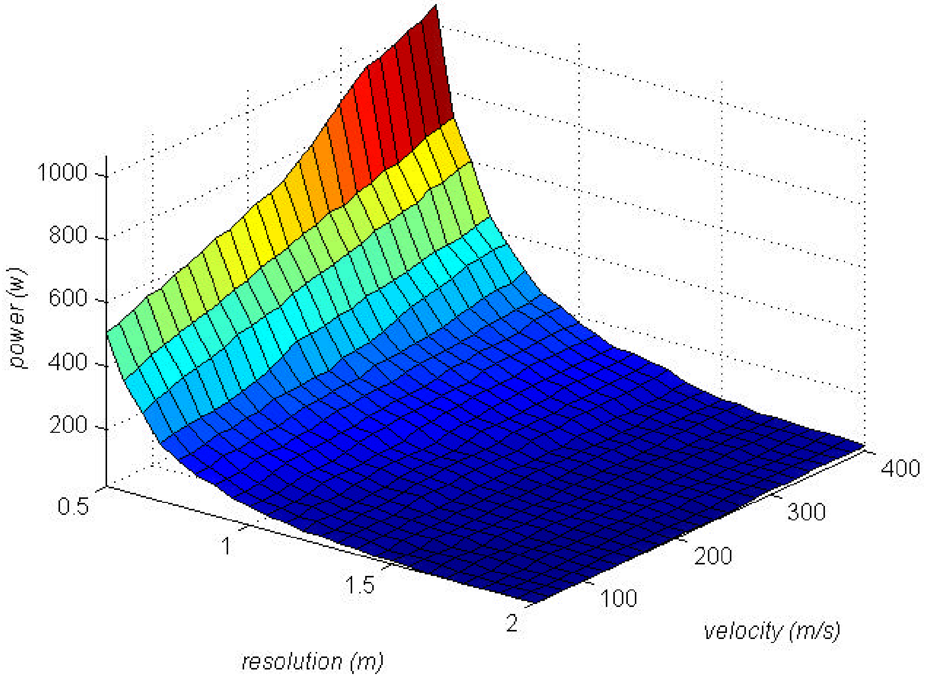

Fig. 2. Power consumption of the CN-constrained model.

Figure 1 shows the surface plot of the ratio of results obtained by employment of the nominal section size to the optimized section size of the first model. As would be expected, the optimized section size always results in equal or lower power consumption than does the nominal section size. The optimized section size adjusts to take advantage of unutilized processor and/or memory resources resulting from changes in system requirements produced by changes in the velocity (axis labeled *v*) and/or resolution (axis labeled *d*).

In both models, higher velocities and/or finer resolutions require more daughtercards and thus more power. All other radar parameters such as wavelength, range, range swath, and pulse width remain fixed at values representative of a real system [6]. These trends are illustrated in Fig. 2, which represents the optimal power consumption associated with the second model.

Fig. 3 displays the daughtercard configurations necessary for the optimal power values represented in Fig. 2. A configuration is defined as the processor and memory allocation (for range or azimuth processing) per compute node for a particular daughtercard type. An optimal system configuration consists of one or two daughtercard configurations. The two configurations are denoted as *X* and *Y*, with the subscipts *T*, *r*, and *a* designating the daughtercard type (*T*), number of range processors per compute node of that type daughtercard (*r*), and the number of azimuth processors (*a*).

5 Summary and Conclusions

Comparison of the two models shows the first model to be a good approximator to the second model. Both the simplicity of formulation and the speed of data collection lend the first model to be a useful method for obtaining a preliminary estimate for total required system power and number of daughtercards. Refer to [8] for a full comparison of the optimal power consumptions produced by the two models.

This work demonstrates the advantage of employing more than one type of daughtercard in a system. Different daughtercards are characterized by different power requirements and the processor-memory ratio of the compute nodes that they house. Optimization exploits these differences and determines the optimal system configurations.

Generalization of the models developed in this work is straightforward. Although data is collected based on sample daughtercards and compute nodes deemed to be representative of actual systems, the values that characterize the daughtercards are expressed as functions that can be immediately adapted to accommodate any number of additional components.

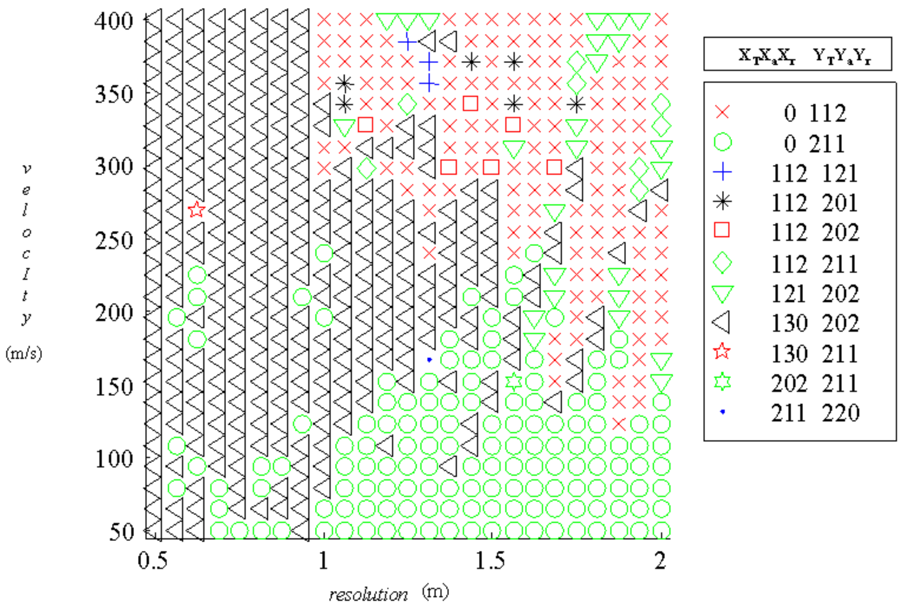

Fig. 3. Configurations of the CN-constrained model.

Acknowledgements

This work was supported by Rome Laboratory under Grant No. F30602-96-1-0098 and Defense Advanced Research Projects Agency (DARPA) under Contract No. F30602-97-2-0297.

References

1. J. C. Curlander and R. N. McDonough, *Synthetic Aperture Radar: Systems and Signal Processing*, John Wiley & Sons, New York, NY, 1991.

2. W. G. Carrara, R. S. Goodman, and R. M. Majewski, *Spotlight Synthetic Aperture Radar: Signal Processing Algorithms*, Artech House, Boston, MA, 1995.

3. J. T. Muehring and J. K. Antonio, "Optimal Configuraion of Parallel Embedded Systems for Synthetic Aperture Radar," *Proceedings of the 7th International Conference on Signal Processing & Applied Technology*, October 1996, pp. 1189-1194.

4. F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, Sixth Edition, McGraw-Hill, New York, NY, 1995.

5. M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Second Edition, John Wiley & Sons, New York, NY, 1993.

6. T. Einstein, "Realtime Synthetic Aperture Radar Processing on the RACE Multicomputer," Application Note 203.0, Mercury Computing Systems, Inc., Chelmsford, MA, 1996.

7. "SHARC DSP Compute Nodes (3.3-Volt)," Mercury Computing Systems, Inc., Chelmsford, MA, Sept. 1995.

8. J. T. Muehring, *Optimal Configuration of a Parallel Embedded System for Synthetic Aperture Radar Processing*, M. S. Thesis, Texas Tech University, 1997 (http://hpcl.cs.ttu.edu/darpa/opt_config/thesis.pdf)

9. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.