Parallel Computing for Array Processing
Chin-Hwa Lee, code EC/Le
Electrical Computer Engineering
Naval Postgraduate School
Monterey, Ca. 93943
email: chlee@ece.nps.navy.mil

Abstract:

A typical array processing task is used here as an example to demonstrate a conceptual decomposition procedure. This procedure partitions the total task into a large number of Multiple Interconnected Modules (MIM). These modules will have adequate annotations to estimate the capacity requirements of a specific partition of the task. It is useful to estimate the computation or communication requirements in a parallel architecture program mapping process. Details of the requirements are developed in this paper with experimental results for the array processing tasks.

Introduction:

Significant progress was made in past few years in parallel computation. Now, there is more incentive to do parallel processing on off the shelf parallel systems. Two main problems persists[5]. Real time requirement of signal processing is difficult to achieve in a general purpose system. There is no standard parallel language or OS for general purpose parallel systems. This is a study of acoustic array processing on a parallel system. The goal is to represent the beam forming algorithm at a generic level so that it can be mapped to commercial available parallel systems[6]. The decomposition process into and mapping process out of the massively interconnected module (MIM) allows us to explore inherent algorithm parallelism on one hand and target architecture optimization on the other hand. The algorithm decomposition to modules allow us to exam the grain size of the inherent parallelism so that proper parallel system can be employed. Initially, the beam forming problem is introduced. A set of *metrics* are developed to characterize the

MIM decomposition. The computation requirements for acoustic beam forming in terms of these *metrics* are described. These *metrics* are used to guide the mapping process so that proper parallel system can be employed. A narrow band beam forming problem is decomposed and executed on a network of Sparcs workstations. Preliminary results demonstrate the potential benefit of beam forming on parallel systems.

MIM: Conceptual Decomposition

When a mathematical model is translated to a detailed structural model for implementation, often the communication consideration outweighs the computational considerations. Such a structural model is referred to as the *Massively Interconnected Models* (MIM). Numerous examples exist in the areas of parallel computation, distributed computer communication networks, and artificial neural networks. The beam forming problem in acoustic array processing can be described into *Massively Interconnected Model*, which challenges the designers hopping to exploit parallel processing technology of the future.

Beam Forming is a spatial filtering problem. Signals are coming in from multiple acoustic sensors numbered in thousands and placed over a spatially limited area (2D). These signals are processed and enhanced coherently so that directivity of the arrival beam can be discriminated in both the azimuth and elevation directions. These signals will be transformed and manipulated according to a mathematical model that requires MIM for implementation. The beam forming problem is studied here to demonstrate a procedure necessary to describe typical MIM problems. It is used also to

characterize available architectures that can fit the MIM implementation.

In mathematical model abstract formula and notations are used to describe the algorithm. Later, to implement this algorithm a practical architecture was selected to accommodate the processing. The architectures may be VLSI chips, multiple processor systems using DSP chips, SIMD machines such as CM-2, or MIMD machine such as Hypercube or iWarp. Modeling in MIM is an intermediate step in which alternative partition of the algorithm is studied. Vital issues are the resource allocation and scheduling when MIM modules are fitted to the architectures. Therefore, MIM modules are not considered as direct hardware architectural components but as conceptual components of the algorithm.

Since the objective is to implement a signal processing algorithm (Beam forming) in a system at a minimal cost that includes the whole life cycle cost (hardware/software design cost and maintainability costs), it is necessary to develop *metrics* that can help to answer the following questions:

1. What are the capacity requirements in the algorithm?
2. Among all the architectures, such as VLSI, DSP chips, SIMD machines, and MIMD machines, which architecture can fit the beam former best? This is a complicated issue. The MIM module decomposition is used to resolve this question.
3. How to decompose the algorithm into MIM modules? The mapping of MIM to architecture depends very much on the architecture under consideration.

Bandwidth Considerations:

The beam forming is a computation intensive operation. Here, a set of *metrics* are developed to characterize this computation. The requirements of a beam former is determined from the mathematical model developed in the previous section. In order to show it as a computation intensive operation the following metrics are used.

M1: Computation Bandwidth (BW) requirement: A real-time beam former receives acoustic signals, processes the data, and provides beam information $y(t)$ for a specific frequency to subsequent systems. Due to hard real-time delays required for processing it is necessary to consider processing capacity in unit time. Frequency of operations per unit time is characterized in terms of the computational BW requirement (Megaflops/sec).

M2: Communication Bandwidth (BW) requirement: The beam former typically processes data from sensors and steering vectors from disk. The hard real-time delay has to be satisfied. These are specified as I/O bandwidth (BW) requirements (bytes/sec). The beam former algorithm requires certain capabilities, and whether an architecture can fulfill that requirement is an important question.

M3: Memory Bandwidth
Requirement: Very often in order to speed up operation some of the coefficients can be precalculated and stored into memory. Sometimes, the memory reads and writes are so frequent and intensive that the total run time slows down. In real-time applications it is also imperative to characterize this impact in terms of memory bandwidth requirement.

Beam Former? Be Specific!

The requirement of a beam former depends very much on the size of the array and the number of beams involved. A typical Passive Sonar Example is used here as a guideline. This example has been shown as follows.

Very Simplified Beam Former:

1. Only fixed beam forming is included.
2. A 1-D linear uniformly spaced array is considered.
3. Total sensors is 100.
4. Frequency domain approach is adopted.
5. Azimuthal coverage of 360° with 3° resolution.
6. Frequency coverage is from 0-256 Hz.

Peak Rate versus Sustainable Rate

At a first look of the capacity requirement discussed before, it does not seem to be stringent. Keep in mind that the very simplified beam former is a trivial example. Realistic beam former has much larger capacity requirements than these. Even for such a small task the bandwidth described so far is the *sustainable* bandwidth not the *peak* bandwidth. Many commercially available processors or DSP chips claim a capacity in this order of magnitude. But, their claims are universally *peak* bandwidth. It means under the most ideal situations without delay of data and delay of instructions the processor can achieve, for example, 10 MFLOPS capacity. The overall sustainable processing rate very much depends on the communication of data or instructions in the system, which is much smaller than the peak rate. Exactly how communication affects the computation depends on the type of algorithms involved. Some commercial systems can achieve a very high CPU processing rate (FLOPS). However, sustainable rate for other types of jobs is very poor. As far as beam forming is concerned sustainable rate of a potential candidate architecture is of primary interest.

Communication Impact *metrics*

Before we attempt to address the question of how to partition algorithm into MIM modules, it is necessary to characterize the MIM modules to see whether it can be accommodated in a particular architecture. In addition to the bandwidth requirement developed previously, it is also essential to consider the following metric.

M4: FLOPS-I/O ratio (α): This is a metric to characterize the proportion of computation done versus communication (I/O) required in the partition. It can be used to describe the MIM module requirement. It can also be used to characterize the architecture element. If the peak FLOPS-I/O ratio (α) of an architecture element is *less* than the MIM

module requirement, it is possible to fit the MIM module into the element. If the peak FLOPS-I/O ratio (α) of an element is greater than the MIM requirement, problem exists to use this architecture element.

M5: Latency -FLOPS product (β): This is a metric to compare different decompositions of MIM modules for architectural elements. Generally, an element can be tuned to achieve a high FLOPS rate, but the latency associated with the data stream is also increased. Consequently, it is necessary to compare the latency-FLOPS product among different architecture elements. For a particular MIM module partition it is also necessary to calculate the latency-FLOPS product. Only elements with less latency-FLOPS product can accommodate the MIM module of larger latency-FLOPS product.

MIM modules with small FLOPS-I/O ratio and latency-FLOPS product are generally referred to as *fine grain* tasks. On the other hand, architecture elements usually have limited achievable FLOPS-I/O ratio and latency-FLOPS product. Fine grain MIM modules can only be accommodated in fine grain architecture elements. Essentially, the FLOPS-I/O ratio and the latency-FLOPS product are metrics to characterize computational activities relative to communication activities. With these metrics it will be easier to analyze the results of different mapping approaches.

Narrow band beam former on parallel system

For experimentation on parallel system a narrow band beam former was used. It satisfies the requirements of the very simplified beam former discussed previously. To increase the computation load the azimuthal resolution was increased from 3° to 1° . This is a correlation based narrow band beam former whose tasks are shown in Figure 1. The program consists of five modules:

bf3h: The top module that does the beam forming and produce the output spectrum plot in Figure 2.

cor3h: The sensor response module that simulate the noise contaminated

signal. There is provision for uncorrelate, coherent, or correlated noise among multiple acoustic sources.

strxxs: The spectrum response module that accepts the correlation matrix and produces directional response. This task is repeated 180 times for each degree of the azimuthal resolution. This task is decomposed into four groups and accommodated in four Sparcs workstations.

Figure 1 Narrow band beam former tasks

beamform: The control and plotting module that accepts output of the spectrum response and plots the final results. A SIMD computation model is followed in a network of 4 Sparc workstations. Programs residing on each node are identical. Each workstation produces a group of spectrum response, and the final results are sent back to node 0 for plotting. This narrow band beam former is run on several different computers including the 4-node network, and its results are shown in Figure 3. As shown a speedup of 3 can be achieved in this parallel system. The cost of this benefit is really the storage required to the system. 573.4 Kbyte are required to accommodate the program and Express runtime routines in the network.

The parallel system is a collection of workstations based on ethernet. The time spent falls in three categories. The first is the calculation time. The second is the node communication time. The third is the network I/O time. Figure 4 shows the percentage bar chart of the node

activities in the system. It is clear that for most of the node large percentage of time is spent in network I/O. Since this is a ethernet based system, message contention during result transfer is significant. If more workstations are incorporated in this parallel system the node calculation time can be reduced, but the network I/O time will be increased. For a large acoustic array

Real time (sec.) Program size (Kbytes)

Apollo 4000	500.6	6.3
hp/Apollo 400	184.9	16.3
Sparc 1+	96.6	204.8
Parallel 4 Sparcs	32.0	573.4

Figure 3 Execution time of narrow band beam former on various machines.

problem the achievable speedup is still valuable even though the network contention is serious on the ethernet.

Conclusions

The algorithm and characteristics of a beam former was introduced and described. Due to the nature of a beam former, communication considerations outweigh the computational considerations in this problem. That is a typical Massively Interconnected Modeling (MIM) type of problem. A set of capacity metrics in terms of bandwidth and communication impact are developed here. These metrics are used in the mapping process from mathematical algorithms to MIM modules and from MIM modules to architectural elements. Often the communication considerations are the main factors in all fitting/mapping problems for a beam former. A narrow beam former is implemented on a network of 4 Sparcs workstations. Experimental results shows the benefit of parallel processing. Network I/O for transferring data constitutes a high percentage of execution time. This situation can be relieved in other kind of parallel systems.

References

- [1]. Signal Processing Handbook, edited by Chen, "Digital Time Delay Beamforming," Roger Pridham, Dekker, 1988.
- [2]. B.D. Van Veen and K.M. Buckley, "Beamforming: A Versatile Approach to Signal Filtering," IEEE ASSP Magazine, April 1988.
- [3]. W. Anderson, G. Lamb, W. Smith, "Acoustic Signal Processing on a Connection Machine," NRL Report 9285, Nov. 28, 1990.
- [4]. S. Borkar, et al., "Supporting Systolic and Memory Communication in iWarp," proceedings of the 17th Annual International Symposium on Computer Architecture, Seattle, WA, May 28, 1990.
- [5]. A. S. Tanenbaum, M. K. Kaashoek, M. E. Bal, "Parallel Programming Using Shared Objects and Broadcasting", IEEE computer, August 1992.

Figure 2. Directional response of the narrow band beam former.

Figure 4. Percentage activities among nodes.