

Experimental Analysis of Parallel Beamforming Algorithms on a Cluster of Personal Computers

Keonwook Kim, Alan D. George and Priyabrata Sinha

HCS Research Lab, Electrical and Computer Engineering Department, University of Florida
216 Larsen Hall, Gainesville, FL 32611-6200, USA

Abstract

As innovations in acoustics and signal processing, such as adaptive processing, continue to result in beamforming algorithms better able to cope with quiet sources and cluttered environments, the computational requirements of the algorithms also rise, often at a pace exceeding that of conventional processor performance. Parallel processing algorithms coupled with advanced networking and distributed computing architectures can be used to turn the telemetry nodes of sonar arrays into processing nodes and thereby perform as a distributed processing system for autonomous, in-situ sonar beamforming. In this paper, we compare the performance of two methods of parallelization, applied to two different beamforming algorithms, on a scalable cluster of personal computers. Various performance characteristics are measured and analyzed.

1. Introduction

Beamforming algorithms are particularly vital in sonar, radar and wireless communication systems. The sequential implementation of beamforming algorithms for many sensors and over a wide band of frequencies presents a significant computational challenge. In order to implement beamforming algorithms for sonar array signal processing in real-time, considerable processing power is necessary to cope with these demands. A distributed processing approach, in which each sensor node is replaced by a ‘smart’

node comprised of a processor and sensor, and the nodes are connected by a network, holds the potential to eliminate the need for a centralized data collector and processor, and increase overall computational performance, dependability, and versatility. The parallel algorithms considered here are designed for such a distributed system.

2. Overview of Beamforming Algorithms

Beamforming is a class of array processing algorithms that optimizes an array gain in a direction of interest and/or locates directions of arrival (DOA) for sources. The two parallel beamforming algorithms discussed in this paper are based on conventional beamforming (CBF) and subspace projection beamforming (SPB), respectively.

2.1 Conventional Beamforming (CBF)

In CBF, signals sampled across an array are linearly phased (i.e. delayed) assuming a configuration with uniform distance between elements in the array. Incoming signals are steered by complex-number vectors called steering vectors. If the beamformer is properly steered to an incoming signal, the multi-channel input signals will be amplified coherently, maximizing the beamformer output power. Otherwise, the output of the beamformer is attenuated to some degree. Thus, peaks in the beamforming output indicate DOA for sources. The output power of CBF for each steering angle θ is defined as

$$P_{CBF}(\theta) = s(\theta)^* \cdot R \cdot s(\theta) \quad (1)$$

where $s(\theta)$ is the steering vector, R is the Cross-Spectral Matrix (CSM), and operator $*$ indicates complex-conjugate transposition.

2.2 Subspace Projection Beamforming (SPB)

The SPB algorithm used in this study is a subspace projection beamformer based on QR decomposition [1]. SPB is categorized as a subspace adaptive beamforming (ABF) algorithm. This technique makes use of the property that columns of the Q matrix associated with noise are orthogonal to the subspace spanned by the incident signal mode vectors. The reciprocal of steered noise subspace indicates peak points at signal locations. The Q matrix is obtained from the QR decomposition of the CSM using elementary reflectors in this study. The output of the subspace beamformer is defined as

$$P_{ABF}(\theta) = \frac{1}{s^*(\theta)E_N E_N^* s(\theta)} \quad (2)$$

where E_N is a matrix whose columns are composed of the columns of Q matrix corresponding to the noise space.

3. Parallel Beamforming Algorithms

In a distributed sonar array for in-situ parallel processing, the degree of parallelism is linked to the number of physical nodes in the system. However, an increase in the number of nodes increases the problem size. Two methods of parallelization have been employed in this paper for each of the two beamforming algorithms.

3.1 Iteration Decomposition

The first parallelization technique, known as *iteration decomposition* [2,3], focuses on the partitioning of beamforming jobs across iterations, with each iteration processing a different set of array input samples. Successive iterations are assigned to successive processors in the array and are overlapped in execution with one another by

pipelining. A single node performs the beamforming task for a given sample set while the other nodes simultaneously work on their respective beamforming iterations. At the beginning of every iteration, each node executes an FFT on data that has been newly collected by its sensor, and the results are communicated to other processors before the beamforming for that iteration commences. The block diagram in Figure 1 illustrates the manner in which beamforming iterations are distributed across the nodes in the distributed array, in this case using three nodes.

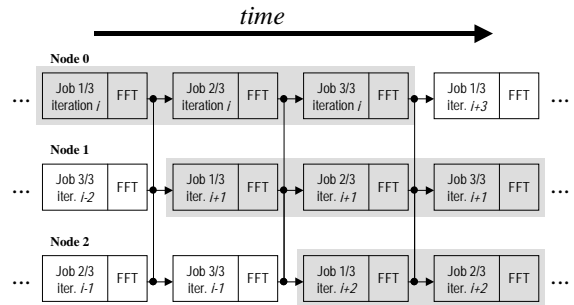


Figure 1: Iteration decomposition

Each processor calculates an index based on its node number, the current job number and the number of nodes. This index tells the node from which point in its iteration it must continue after executing the FFT and communication stages and when it must pause to begin another iteration.

The communication pattern that is required in *iteration decomposition* for both CBF and SPB algorithms is an ‘all-to-one’ pattern, as only one of the nodes needs to receive the data sampled each cycle to perform a given beamforming iteration on data collected throughout the array.

3.1 Frequency Decomposition

The second parallelization technique is termed *frequency decomposition*. This method relies on the fact that almost all the computational tasks in either beamforming algorithm can be performed for each fre-

quency bin independent of other frequency bins. Hence, each computing node is assigned the responsibility for processing a different set of frequency bins (i.e. a different frequency band). After each node has executed an FFT on the data that has been obtained by its sensor, it distributes the data to the other nodes on the basis of frequency. Thus, the first node gets the data corresponding to the first set of frequencies, the second node gets the data corresponding to the second set of frequencies, and so on. At the end of each iteration, the outputs computed by each node for its assigned frequency bins are sent to the other nodes. Every node averages the individual outputs to obtain the final power output.

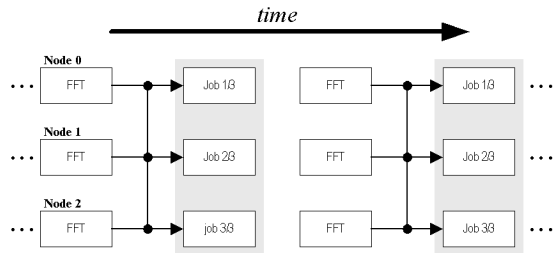


Figure 2: Frequency decomposition

It is apparent from the preceding discussion that this method, unlike *iteration decomposition*, requires an ‘all-to-all’ communication pattern for the initial distribution of data and final accumulation of individual outputs. This communication pattern causes the overhead of initiating communication to be incurred twice, resulting in a significant increase in communication time. This overhead is reduced by a ‘data-packing’ technique, wherein the two instances of communication performed by each node are combined into one packet, thereby incurring the communication overhead only once rather than twice.

Thus, it can be seen that the two parallelization techniques involve different communication patterns, which play a vital role in determining parallel performance.

4. Parallel Performance Analysis

The work presented in this paper focuses on parallel performance analysis of these algorithms using C-MPI and executed on a cluster of PCs known as CARRIER, the Cluster Array for Interconnect Evaluation and Research. The subset of this cluster used for this work is comprised of 32 nodes, each with a 400MHz Intel Celeron processor, 128KB of integrated L2 cache, and 64MB of RAM. The processing nodes are interconnected by a Fast Ethernet network and use the Linux operating system. In the experiments described in this section, a sampling frequency of 1500Hz is assumed and the beamforming is performed for 192 frequency bins. The FFT length of the processed data is 2048 samples. A total of 181 steering angles are resolved. Figure 3 shows the sequential execution times for the two beamforming algorithms using a single node in the cluster.

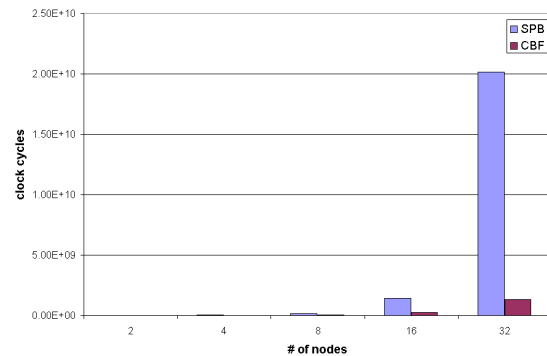


Figure 3: Sequential execution times

The execution time for sequential CBF is seen to be lower than for sequential SPB. This phenomenon is due to the computationally intensive QR decomposition stage of SPB, which is not present in CBF. Overall, the sequential complexities of the CBF and SPB algorithms are $O(N^2)$ and $O(N^3)$, respectively. Thus, due to the difference in these computational complexities, the sequential execution times for SPB grow far more rapidly with problem size as compared to CBF.

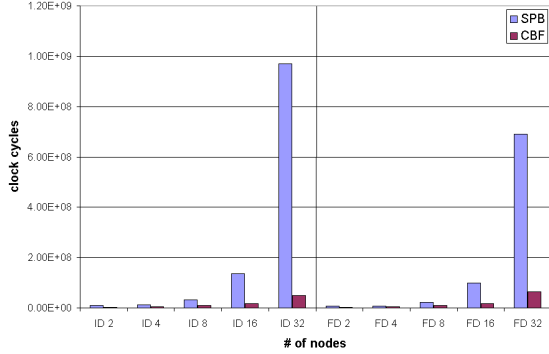


Figure 4: Parallel execution times

Figure 4 shows the parallel execution times for the two different parallelization schemes, *iteration decomposition* (ID) and *frequency decomposition* (FD). In the case of CBF, the computational complexities of both the parallel algorithms are reduced to $O(N)$. By contrast, the computational complexity of the two parallel algorithms for SPB are reduced to $O(N^2)$. These reductions in complexity are confirmed by the parallel execution times, in which the parallel CBF execution times increase linearly as the number of nodes is increased, while the parallel SPB execution times increase in a quadratic manner.

A comparison between the parallel execution times for the two parallelization techniques indicates that ID has higher execution times than FD. This difference is due to the pipeline management overhead inherent in the *iteration decomposition* method.

It is apparent from the results in Figures 3 and 4 that the effective execution times of both the parallel algorithms are much lower than their sequential counterparts. Moreover, the increase in parallel execution time as the problem size increases is less pronounced than in the sequential case. Thus, the parallel algorithms are seen to provide higher throughputs of execution compared to the sequential CBF and SPB algorithms.

Speedup is defined as the ratio of the sequential execution time versus parallel execution time, where ideal speedup is equal to the number of processors employed. Scaled

speedup recognizes that, as in this case, an increase in the number of processors brings with it an increase in the problem size, since each node possesses both a processor and a sensor. As seen in Figure 5, the scaled speedups achieved in all four cases are observed to be nearly linear. However, in the case of CBF, the speedups provided by FD are less than those provided by ID. This lower speedup occurs because the all-to-all communication in FD requires much larger communication time than the all-to-one communication in ID, resulting in a reduction in the overall speedup. In the case of SPB, higher speedups are obtained in FD since the computational component that can be parallelized in SPB is greater than in CBF with an increase in number of nodes. This reduces the relative effect of communication times in the case of SPB, while the higher pipeline overhead in ID causes it to have a lower speedup.

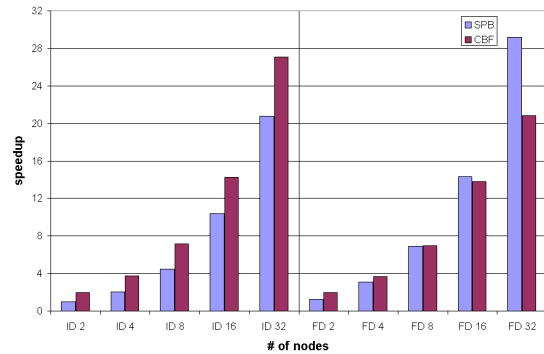


Figure 5: Beamformer Scaled Speedup

Parallel efficiency is defined as the ratio of speedup versus the number of processing nodes (i.e. the ideal speedup). As illustrated in Figure 6, the algorithms achieve parallel efficiencies ranging from 48% to almost 100%. Interestingly, for CBF the parallel efficiencies decrease with increasing number of nodes, while for SPB the efficiencies increase. In SPB, as mentioned earlier, the communication has a relatively lower effect. As the amount of computation increases, the amount of parallelism that can be exploited also increases, leading to higher efficiencies

for larger problem sizes. By contrast, CBF has smaller computational requirements than SPB. Thus, the increasing communication time plays a more significant role, reducing the efficiencies as the number of nodes is increased. This decrease in parallel efficiency with increasing problem size is relatively less in the case of ID, since the all-to-one communication times increase less rapidly as compared to the all-to-all communication of FD.

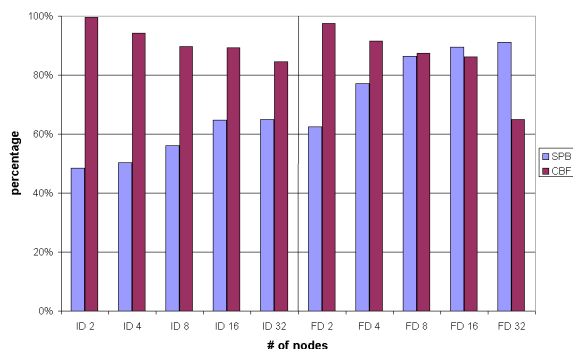


Figure 6: Beamformer Parallel Efficiency

For faster execution, most of the invariant parameters are stored in memory space. In this configuration, the majority of the memory requirement arises from the steering procedure. Iteration decomposition requires storage of the entire steering vector for all frequencies, because each processor performs a whole beamforming task for an incoming data set. By contrast, frequency decomposition needs only part of the memory space for steering, since individual processors generate only part of the beamforming result for a given data set. For both the sequential algorithm and iteration decomposition, the demands for memory space for the steering procedure grow linearly when the number of nodes is increased, as shown in Figure 7. However, little change is observed for frequency decomposition.

5. Conclusions

In this paper, two different decomposition techniques are applied to two beamforming algorithms, CBF and SPB. It is ob-

served that communication has a significantly detrimental effect on the performance of parallel programs for less complex algorithms such as CBF. However, communication time plays a smaller role in determining the performance of more complex algorithms such as SPB. Parallel efficiency increases with SPB and decreases with CBF as the problem and system sizes increase. Due to its lower cost in communication, ID-based parallel beamformers are better suited for systems with a slower network. By contrast, FD-based beamformers are better suited for systems with limited memory capacity. Future directions for this work include the application and analysis of parallelization techniques for other beamforming algorithms from the classes of adaptive and matched-field processing.

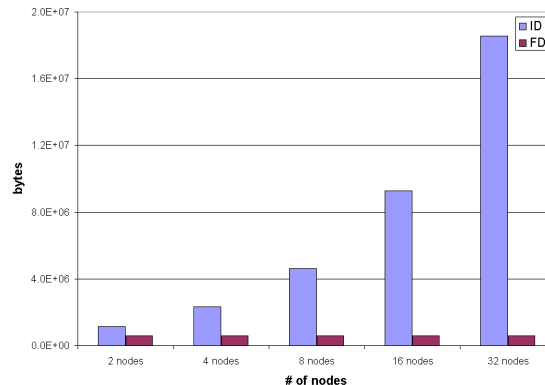


Figure 7: Memory usage in steering stage

Acknowledgements

This work was sponsored in part by the Office of Naval Research on grant N00014-99-1-0278.

References

- [1] M.J. Smith and I.K. Proudler, "A one-sided algorithm for subspace projection beamforming," *SPIE Vol. 2846*, 100-111, 1996.
- [2] A. George, J. Markwell, and R. Fogarty, "Real-Time Sonar Beamforming on High-Performance Distributed Computers," *Parallel Computing*, Vol. 26, No. 10, Aug. 2000, pp. 1231-1252.
- [3] A. George and K. Kim, "Parallel Algorithms for Split-Aperture Conventional Beamforming," *Journal of Computational Acoustics*, Vol. 7, No. 4, 225-244, 1999.