

# Comparative Performance Analysis of Parallel Beamformers

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

Advancements in beamforming algorithms are exceeding the computation and communication capabilities of traditional sonar array systems. Implementing parallel beamforming algorithms in situ on distributed array systems holds the potential to provide increased performance and fault tolerance at a lower cost. This paper compares three parallel algorithms for distributed arrays in terms of execution throughput, result latency, scaled speedup, and parallel efficiency.

## 1. Introduction

Passive sonar beamforming is a class of array processing that optimizes an array gain in a direction of interest to detect and locate objects in an undersea environment. Beamforming algorithms are particularly vital in radar and sonar applications. The parallel algorithms considered here are designed for a distributed array of sonar transducer nodes, each with its own processing element and interconnected by a network. The necessity for parallel beamforming algorithms is a direct result of the development of advanced beamforming algorithms that are better able to cope with quiet sources and cluttered environments. These developments have resulted in increased demands for real-time computation. Moreover, the use of larger sonar arrays has in turn led to larger problem sizes. Thus, a beamformer based on a centralized processing system may prove insufficient to meet these demands, and parallel

processing in-situ on a distributed array is a promising alternative.

## 2. Overview of Beamforming Algorithms

The basic operation in most beamforming algorithms is to sum the manipulated outputs from many spatially separated sensors. The three parallel beamforming algorithms discussed in this paper are based on Conventional Beamforming (CBF), Split-Aperture Conventional Beamforming (SA-CBF) [1] and an Adaptive Beamforming (ABF) algorithm for subspace projection [2], respectively.

### 2.1 Conventional Beamforming (CBF)

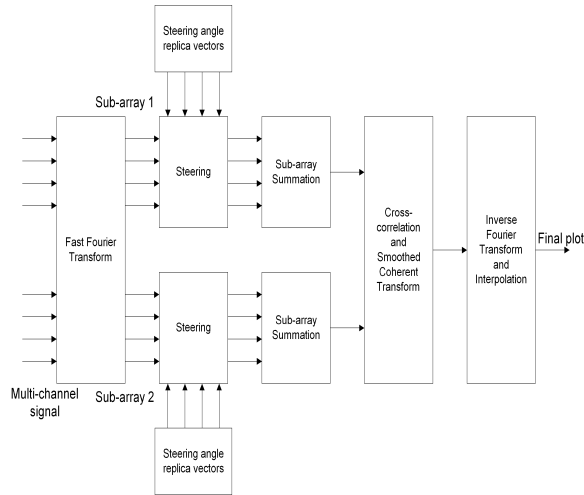
In a sonar array, the determination of the direction of arrival relies on the detection of the time delay of the signal between sensors. In CBF, signals sampled across an array are linearly phased (i.e. delayed) assuming a configuration with uniform distance between elements in the array. Incoming signals 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 directions of arrival for sources. The output power of CBF for each steering angle  $\theta$  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 Split-Aperture CBF (SA-CBF)

SA-CBF is based on single-aperture conventional beamforming in the frequency domain. The beamforming array is logically divided into two sub-arrays. Each sub-array independently performs CBF using steering vectors on its own data. The two sub-array beamforming outputs are cross-correlated to detect the time delay of the signal for each steering angle. Interpolation is used to generate outputs for angles other than the steering angles (e.g. a ratio of 4-to-1 is used in this study). The cross-correlated data, with knowledge of the steering angles and several other parameters, will map the final beamforming output. Figure 1 shows the block diagram of the SA-CBF algorithm.



**Figure 1: Block diagram of SA-CBF**

## 2.3 Adaptive Beamforming (ABF)

The ABF algorithm used in this study is a subspace-projection beamformer based on QR decomposition [2]. Subspace beamforming algorithms for ABF such as MUSIC make use of the property that eigenvectors associated with noise are orthogonal to the space spanned by the incident signal mode

vectors. The reciprocal of steered noise subspace indicates peak points at signal locations. However, subspace identification to separate noise and signal using the Singular Value Decomposition (SVD) is computationally expensive to perform and difficult to implement in a parallel algorithm due to the many dependencies between the computational tasks. Instead of using the eigenvectors of CSM matrix, the columns of the  $Q$  matrix are used, which correspond to the noise subspace. The  $Q$  matrix is from the QR decomposition of the CSM matrix using elementary reflectors in this study. The output of the subspace beamformer is defined as

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

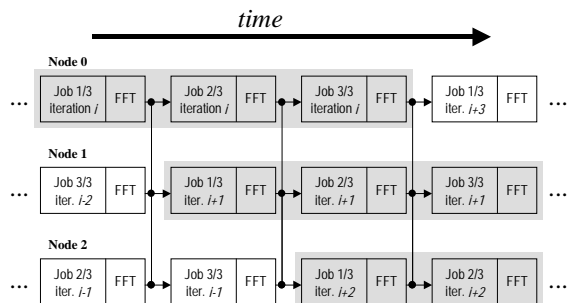
where  $E_N$  is the columns of  $Q$  matrix corresponding to the noise space.

## 3. Parallel Beamforming Algorithms

In a distributed sonar array for parallel processing in situ, 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. The goal is to obtain minimum processor stalling through equal distribution of work and minimum communication overhead.

The method of parallelization employed by the parallel algorithms in this paper, known as *iteration decomposition* [3,4], 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 2 illustrates the manner in which beamforming iterations are distributed across the nodes in the distributed array, in this case using 3 nodes.



**Figure 2: 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.

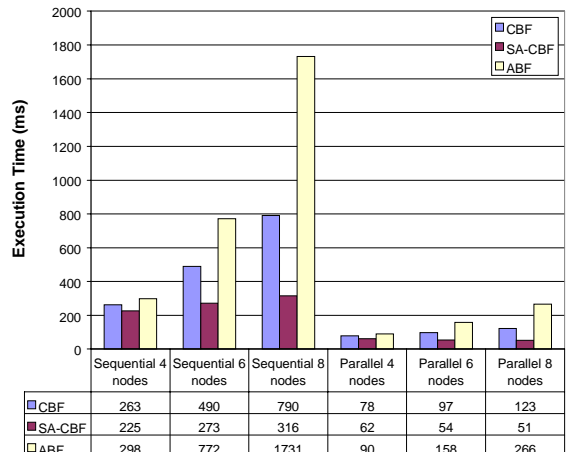
The communication pattern that can be expected in the CBF and SA-CBF algorithms is an ‘all-to-one’ pattern, as only one of the nodes needs to receive the data sampled each cycle to perform a given beamforming iteration on data collected throughout the array. However, ABF algorithms require ‘all-to-all’ communication so that the cross-spectral matrix on each of the nodes is updated with each cycle of sampling. Thus, to provide a common framework for comparisons, an ‘all-to-all’ type of communication is used with all the parallel algorithms, where each node sends its Fourier transformed data to all other nodes.

#### 4. Parallel Performance Analysis

The parallel algorithms in this paper were implemented in MPI-C and executed on a cluster of SPARCstation-20 workstations connected by a 155 Mb/s ATM net-

work. In the experiments described in this section, a sampling frequency of 1500Hz is assumed and the beamforming is performed for 200 frequency bins. The FFT length of the processed data is 2048 samples, and no frequency-bin averaging is performed. Approximately 180 steering angles are resolved (i.e. 181 for CBF and ABF, and 177 for SA-CBF), with a 4-to-1 ratio of interpolation used with SA-CBF.

It is apparent from the results in Figure 3 that the effective execution times of 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 a higher throughput of execution.

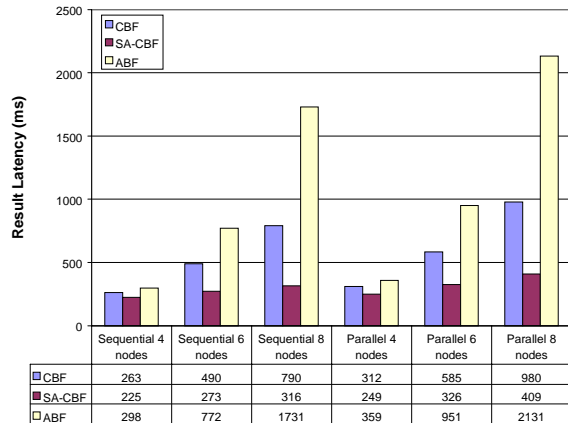


**Figure 3: Beamformer execution times**

The execution time for parallel SA-CBF is always less than that for both parallel CBF and parallel ABF. Unlike SA-CBF, the CBF and ABF algorithms must directly process for all steering angles, no interpolation is performed, and hence they involve more computation. This characteristic is an inherent strength of the SA-CBF algorithm. ABF requires a higher execution time than both CBF and SA-CBF because of the complexity of the QR decomposition stage.

Figure 4 shows the result latencies for the different algorithms. Result latency re-

fers to the time required for the final output to be available after the data has been read by the sensors. In the case of sequential algorithms, result latency is the same as the execution time, as there is no pipelining involved.

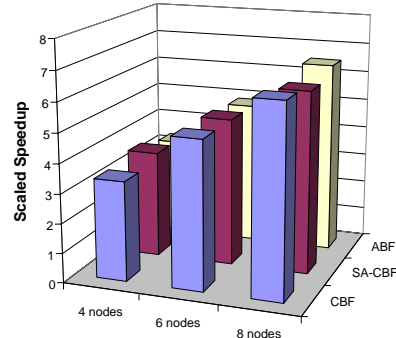


**Figure 4: Beamformer Result Latency**

Result latencies with the parallel algorithms are slightly higher than those of their sequential counterparts. This difference can be attributed to the fact that each beamforming job has been divided into pipeline stages and hence involves pipeline management overhead and communication time between successive stages. Thus, there is an obvious tradeoff between execution throughput and result latency when using parallel algorithms based on the technique of iteration decomposition.

Speedup is defined as the ratio of the sequential execution time versus the parallel execution time, where ideal speedup is equal to the number of processors employed. Scaled speedup recognizes that, 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 for the three parallel algorithms are observed to be near linear. However, for a higher number of nodes, parallel SA-CBF appears to provide a lower speedup compared to parallel CBF and parallel ABF.

This outcome is a result of the presence of loops of different sizes, different number of steering angles and different number of output angles, leading to a slight imbalance.



|        | 4 nodes | 6 nodes | 8 nodes |
|--------|---------|---------|---------|
| CBF    | 3.38    | 5.02    | 6.45    |
| SA-CBF | 3.62    | 5.01    | 6.17    |
| ABF    | 3.32    | 4.87    | 6.5     |

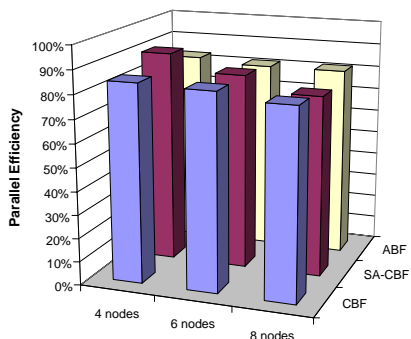
**Figure 5: Beamformer Scaled Speedup**

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 parallel algorithms achieve levels of parallel efficiency in the range of 77-91%, with an average of approximately 80% for the largest cases. Since communication overhead plays a more significant role as the size of the system increases, the efficiencies decrease slightly with increase in the number of nodes. However, the relatively flat nature of these results demonstrates that scalability is achieved at least for arrays of moderate size and complexity.

## 5. Conclusions

This paper has presented a comparative analysis of the performance of several parallel algorithms for in-situ beamforming on a distributed system. Each of the algorithms is based on the same technique of pipelined decomposition, where consecutive iterations of the beamforming process are scheduled in a round-robin fashion to execute on consecutive processing nodes in the array. These algorithms were implemented as message-passing parallel programs and executed

on a cluster of workstations connected by ATM and their performance measured.



|        | 4 nodes | 6 nodes | 8 nodes |
|--------|---------|---------|---------|
| CBF    | 85%     | 84%     | 81%     |
| SA-CBF | 91%     | 84%     | 77%     |
| ABF    | 83%     | 81%     | 81%     |

**Figure 6: Beamformer Parallel Efficiency**

With respect to execution time, the parallel algorithms demonstrate a consistent relationship regardless of the system size, where split-aperture CBF performs the fastest, followed by the single-aperture CBF and lastly the ABF. The sequential algorithms demonstrate this same trend. However, despite the increased complexity associated with ABF, the results in these experiments indicate that their execution throughput surpasses the simple CBF by at most only a factor of 2.

One of the disadvantages of using a pipelined approach to parallel processing is an increase in result latency. However, measurements indicate that the pipelining overhead that increases the latency in producing results is marginal.

Finally, the scaled speedup and parallel efficiency achieved with each of the parallel beamformers was found to be within approximately 80% of the ideal for systems of four, six, and eight nodes. The general trends indicate that comparable performance can be expected for larger arrays, since the decrease in efficiency as system size increases is relatively slow.

The parallel beamforming algorithms compared in this paper present many opportunities for increased performance, reliabil-

ity, and flexibility in a distributed system for sonar signal processing. Undertaking and coordinating the computations and communications to perform beamforming in situ is a challenging task, and is becoming more so as the beamformers themselves continue to become more sophisticated. Some beamformers, such as Minimum Variance Distortionless Response (MVDR), exhibit an even larger degree of communication overhead and thus require a more elaborate scheme to achieve reduction and hiding of communication latency [5].

Future research activities on the subject of parallel algorithms for in-situ processing on distributed arrays will continue to focus on adaptive techniques in the near term. However, new studies and developments are underway to help address the tremendous challenges in computation and communication associated with advanced beamforming in the littoral environment using matched-field processing.

### Acknowledgements

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

### References

- [1] F. Machell, "Algorithms for broad-band processing and display," ARL Technical Letter No. 90-8 (ARL-TL-EV-90-8), Applied Research Laboratories, Univ. of Texas at Austin, 1990.
- [2] M.J. Smith and I.K. Proudler, "A one sided algorithm for subspace projection beamforming," SPIE Vol. 2846, 100-111, 1996.
- [3] A. George, J. Markwell, and R. Fogarty, "Real-time sonar beamforming on high-performance distributed computers," Parallel Computing, submitted Aug. 1998.
- [4] A. George and K. Kim, "Parallel Algorithms for Split-Aperture Conventional Beamforming," Journal of Computational Acoustics, to appear.
- [5] A. George and J. Garcia, "A Parallel Algorithm for Distributed MVDR Beamforming," Journal of Computational Acoustics, submitted July 1999.