Title: Matrix-Vector Multiplication with MPI - Performance Analysis

Introduction: This report presents a performance analysis of a C program that performs a matrix-vector multiplication using the Message Passing Interface (MPI) library for parallel processing. The purpose of the analysis is to understand how the program scales with an increasing number of processors.

Method: The matrix-vector multiplication program is run with 1, 2, 3, and 4 processors. The input matrix and vector are generated randomly. The matrix is square with the same number of rows and columns, and the vector has the same number of elements as the matrix has rows. The program's execution time is recorded for each run with different numbers of processors.

Results:

Assuming that the input matrix and vector are generated with the same size and that the program scales linearly, the following execution times are expected for each configuration:

1 Processor: Execution Time: T1

2 Processors: Execution Time: T2 = T1 / 2

3 Processors: Execution Time: T3 = T1 / 3

4 Processors: Execution Time: T4 = T1 / 4

Discussion: The program's execution time is expected to decrease as the number of processors increases. The performance improvement is due to the parallelization of the matrix-vector multiplication. Each processor is responsible for a portion of the matrix rows, reducing the overall computation time.

However, the actual performance improvement might not be linear due to several factors, such as communication overhead between processors and non-uniform distribution of work among the processors. Furthermore, the performance gain can also be affected by the hardware and software environment in which the program is executed.

Conclusion: The matrix-vector multiplication program using MPI demonstrates the potential benefits of parallel processing, as it is expected to show improved performance with an increased number of processors. However, the actual performance improvement might vary depending on factors such as communication overhead, work distribution, and the hardware and software environment. Further testing and analysis are required to obtain a more precise understanding of the program's performance characteristics in different environments.