HBM513E Parallel and Distributed Computing

Fall 2019

Term Project

Parallel Sparse Matrix-Vector (SpMV) Multiplication

Consider a sparse matrix-vector multiplication (**B**asic **L**inear **A**lgebra **S**ubroutines: BLAS2 operation) for various sparse matrix sizes with one of the following patterns. Select one matrix type and then register your name to the project list.

Write a parallel sparse matrix-vector multiplication program with MPI using one of the following sparse matrices and compression algorithms. Be aware that your matrix-vector multiplication algorithm must be based on your selected compression algorithm (for further details on sparse matrix vector multiplication look at **the supplementary notes**).

Sparse Matrix Types and Their Compression Algorithms

1) Lower triangular matrix filled with zero in even numbered rows using Block Compress Sparse Row Format (BCSR) storage scheme.
2) Upper triangular matrix filled with zero in even numbered rows using Compress Sparse Row Format (CSR) storage scheme.
3) Hermitian banded matrix using Compress Sparse Column Format (CSC)
4) Tri-diagonal matrix using Coordinate Format (ij-value) storage scheme.
5) Symmetric seven-banded matrix using Compress Sparse Row Format (CSR) storage scheme.
6) Non-symmetric seven-banded matrix using Block Compress Sparse Row Format (BCSR) storage scheme.
7) Symmetric nine-banded matrix using Block Compress Sparse Row Format (BCSR) storage scheme
8) Non-symmetric nine-banded matrix using Compress Sparse Row Format (CSR) storage scheme.
9) Non-symmetric tri-diagonal matrix using Block Compress Sparse Row Format (BCSR) storage scheme.
10) Tri-diagonal matrix using Compress Sparse Row Format (CSR) storage scheme.
11) Symmetric nine-banded matrix using Compress Sparse Row Format (CSR) storage scheme.
12) Symmetric eleven-banded matrix using Compress Sparse Row Format (CSR) storage scheme.
13) Hermitian banded matrix using Compress Sparse Row Format (CSR) storage scheme.
14) Lower triangular matrix filled with zero in even numbered rows using Compress Sparse Row Format (CSR) storage scheme.
15) Non-symmetric Tri-diagonal matrix using Compress Sparse Row Format (CSR) storage scheme.
16) Non-symmetric eleven-banded matrix using Compress Sparse Row Format (CSR) storage scheme.
17) Non-symmetric thirteen-banded matrix using Compress Sparse Row Format (CSR) storage scheme.

18) Symmetric five-banded matrix using Compress Sparse Row Format (CSR) storage scheme.
19) Non-symmetric five-banded matrix using Compress Sparse Row Format (CSR) storage scheme.

After selecting one of the sparse matrices and compression algorithm above, follow the instructions below:

Present the results of questions (1-6) in your project report and presentation.

1. Implement parallel code optimization and tuning approaches to display the potential improvements (scalability) in your algorithm.

2. Test matrix sizes ranging from 200,000 to 1,000,000 with the increment 200,000.

3. Plot a graph showing potential improvements in your wall clock time vs. speedup and efficiency using optimization and tuning methods for the fixed problem size (Amdahl's Law).

4. Plot a graph showing potential improvements in your wall clock time vs. effective speedup and effective efficiency for the varied problem sizes (Gustafson's Law).

5. Plot a graph showing the performance (GFlop/sec.) vs matrix sizes, and compare with the peak performance.

6. Specify and consider the hardware architecture metrics where application running on. And explain why your scalability and efficiency is limited? Discuss the results obtained.

7. Prepare a ***term project report*** and 10 minutes ***oral presentation*** discussing your parallel algorithm and results.

8. **Interval Check 1**: Submit your preparation work on 10th week.

9. **Interval Check 2**: Submit your work without compression algorithm on 13th week.

10. **Final Submission**: Submit your term project report to Dr. ÇELEBİ, and software copy to teaching assistants on the following due date.

DUE DATE: December 16, 2019

Note:
To create the matrices above, it is advised to generate matrices synthetically in a loop or use any matrix library such as University of Florida Matrix Library (UFL).