PARALLEL AND DISTRIBUTED PROGRAMMING

ASSIGNMENT 3

Group 50

Jiahao LU, You WU, Zonghao LU

11 June 2019

## Problem Formulation

Given two dense matrices **A**, **B** in **R**n×n, the task is to implement a parallel matrix multiplication algorithm using C and MPI and evaluate the performance.

The input file contains 2n2 + 1 numbers. The first number is n (an integer indicating the matrix size). The subsequent n2 numbers are the elements of **A**, stored in row-major order. The last n2 elements are the elements of **B**, also stored in row-major order.

## Solution Method

The matrix A is read from file row-wised while matrix B column-wise. Before multiplication, A and B are partitioned row-wise and scattered to different processes. In this case, n is supposed to be divisible by the number of processes p. The results of partitioned chunks multiplication are then gathered into a new matrix C which is the result of the multiplication of A and B.

The row-wised partition strategy is chosen because it is easy to implement while the workload on each process is almost balanced thus it is good for parallelisation. Each process contains scattered part of matrices A and B, then send the local part of B to other processes and receive other parts of B from other processes. In each process, compute a local result of the matrix multiplication. After all local results are computed, gather them into the final results, which is the multiplication of matrices A and B.

## Experiments

1. **Testing environment**

Host: rackham.uppmax.uu.se

1. **Testing method**

The output time is measured in seconds by MPI\_Wtime() function on the master process. Only the parallelised computation time is measured.

The strong scalability is tested for input file input3600.txt. To satisfy the requirement that the matrix size n is divisible by the number of processes p, this input file is tested on process number 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16.

The weak scalability is tested when the number of processes is increased, while the number of intervals is increased simultaneously in a way that the workload per processor is kept to an approximate constant. With the given matrix size n, the workload is . Meanwhile, the requirement that the matrix size n is divisible by the number of processes p must be satisfied. The analysis of the given input files is shown below in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matrix size n | Workload | Number of processes | Number of processes (appr) | Actual workload |
| 3600 | 46656000000 | 1.00041731 | 1 | 46656000000 |
| 5716 | 1.86757E+11 | 4.004518972 | 4 | 46689225424 |
| 7488 | 4.19853E+11 | 9.002667332 | 9 | 46650359808 |
| 9072 | 7.46636E+11 | 16.00968621 | 16 | 46664771328 |
| 10525 | 1.16591E+12 | 25 | 25 | 46636538125 |

The number of processes (appr) is taken by the nearest factor of the matrix size n to the calculated number of processes.

## Results and Discussion

The tested strong stability is shown as follow.

Since only the parallelised computation time is measured, the theoretical speedup for fixed problem size should equal to the number of processes. As the figure shows above, the measured speedup for fixed problem size curve is approximately linear. Therefore, it is reasonable to assess that the strong scalability is well achieved.

The results of weak scalability evaluation are presented below. The ideal speedup for scaled problem size is calculated by the actual workload per process.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix size n | Workload | Number of processes | Number of processes (appr) | Max memory usage/GB | Time/s | Experimental speedup | Ideal speedup |
| 3600 | 4.666E+10 | 1.0004 | 1 | 0.4 | 63.62 | 1.00 | 1.00 |
| 5716 | 1.868E+11 | 4.0045 | 4 | 1.1 | 65.36 | 0.97 | 1.00 |
| 7488 | 4.199E+11 | 9.0027 | 9 | 1.9 | 78.00 | 0.82 | 1.00 |
| 9072 | 7.466E+11 | 16.0097 | 16 | 3.0 | 70.75 | 0.90 | 1.00 |
| 10525 | 1.166E+12 | 25.0000 | 25 | 2.1 | 96.72 | 0.66 | 1.00 |

That the curve moves further from the ideal line as the number of cores increases, which means the weak scalability is not that satisfying.

In conclusion, the strong scalability is very close to the ideal speedup because the matrices are divided into more pieces when the number of processes increases. Each process has its independent cache, so the cache utilization efficiency increases as the number of processes grows. However, the weak scalability is far away from the ideal curve. We think it is because each process needs to communicate with all other processes circularly. When the number of processes increases and the workload in each process keeps the same, the overhead of communication limits the weak scalability.