Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра цифрових технологій в енергетиці

ЗВІТ   
з виконання лабораторної роботи №4  
з дисципліни «Технології паралельних обчислень в енергетичних комплексах»

«Розв’язання систем лінійних алгебричних рівнянь»

Варіант 21

Виконав: студент групи ТР-12 Руденко В.І

Завдання:

Розробити паралельну реалізацію алгоритму Гауса в середовищі MPI.

Хід Роботи:

*#include <math.h>*

*#include <mpi.h>*

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <string.h>*

*#include <sys/types.h>*

*#include <sys/times.h>*

*#include <sys/time.h>*

*#include <time.h>*

*#include <unistd.h>*

*// MPI Variables*

*#define MASTER 0*

*int total\_processes;*

*int rank\_process;*

*// File Variables*

*char \*output\_filename;*

*// Matrix Variables*

*#define MAXN 5000*

*int N;*

*double A[MAXN][MAXN], B[MAXN], X[MAXN];*

*// Prototypes*

*void gauss();*

*void back\_substitution();*

*void finalize();*

*// Function for random based on time*

*unsigned int time\_seed()*

*{*

*struct timeval t;*

*struct timezone tzdummy;*

*gettimeofday(&t, &tzdummy);*

*return (unsigned int)(t.tv\_usec);*

*}*

*// Command line program parameters*

*void parameters(int argc, char \*\*argv)*

*{*

*// There must be atleast one argument -- matrix dimension*

*if (argc < 2) {*

*if (rank\_process == MASTER)*

*{*

*printf("Use: %s <matrix\_dimension> [output\_filename] [seed]\n", argv[0]);*

*}*

*finalize();*

*}*

*// First argument - matrix dimension*

*N = atoi(argv[1]);*

*if (N < 1 || N > MAXN)*

*{*

*if (rank\_process == MASTER)*

*{*

*printf("N = %i is out of range.\n", N);*

*}*

*finalize();*

*}*

*// Second argument - output filename*

*if (argc == 3) {*

*if (rank\_process == MASTER) {*

*int length = strlen(argv[2]);*

*output\_filename = (char \*) malloc(length + 1);*

*output\_filename = argv[2];*

*}*

*}*

*else*

*{*

*output\_filename = "output.txt";*

*}*

*}*

*void initializeInput()*

*{*

*int row, col;*

*srand(time\_seed());*

*for (col = 0; col < N; col++)*

*{*

*for (row = 0; row < N; row++)*

*{*

*A[row][col] = (double)rand() / 32768.0;*

*}*

*B[col] = (double)rand() / 32768.0;*

*X[col] = 0.0;*

*}*

*}*

*void write\_entries(FILE \*filePtr)*

*{*

*int row, col;*

*fprintf(filePtr, "A = [\n\t");*

*for (row = 0; row < N; row++)*

*{*

*for (col = 0; col < N; col++)*

*{*

*fprintf(filePtr, "%9.1lf%s", A[row][col], (col < N - 1) ? ", " : ";\n\t");*

*}*

*}*

*fprintf(filePtr, "]");*

*fprintf(filePtr, "\nB = [");*

*for (col = 0; col < N; col++)*

*{*

*fprintf(filePtr, "%9.1lf%s", B[col], (col < N - 1) ? "; " : "]\n");*

*}*

*fprintf(filePtr, "]");*

*fprintf(filePtr, "\nX = [");*

*for (row = 0; row < N; row++)*

*{*

*fprintf(filePtr, "%9.1lf%s", X[row], (row < N - 1) ? "; " : "]\n");*

*}*

*fprintf(filePtr, "]");*

*fprintf(filePtr, "\n");*

*}*

*int main(int argc, char \*\*argv)*

*{*

*struct timeval etstart, etstop;*

*struct timezone tzdummy;*

*clock\_t etstart2, etstop2;*

*unsigned long long usecstart, usecstop;*

*struct tms cputstart, cputstop;*

*FILE \*filePtr;*

*MPI\_Init(&argc, &argv);*

*MPI\_Comm\_size(MPI\_COMM\_WORLD, &total\_processes);*

*MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank\_process);*

*parameters(argc, argv);*

*if (rank\_process == MASTER)*

*{*

*filePtr = fopen(output\_filename, "w+");*

*initializeInput();*

*gettimeofday(&etstart, &tzdummy);*

*etstart2 = times(&cputstart);*

*}*

*gauss();*

*if (rank\_process == MASTER)*

*{*

*write\_entries(filePtr);*

*gettimeofday(&etstop, &tzdummy);*

*etstop2 = times(&cputstop);*

*usecstart = (unsigned long long)etstart.tv\_sec \* 1000000 + etstart.tv\_usec;*

*usecstop = (unsigned long long)etstop.tv\_sec \* 1000000 + etstop.tv\_usec;*

*printf("Time spent: %g ms.\n", (float)(usecstop - usecstart) / (float)1000);*

*}*

*finalize();*

*}*

*void gauss()*

*{*

*int norm, row, col, multiplier[N], rownum[N];*

*MPI\_Bcast(&A, MAXN \* MAXN, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);*

*MPI\_Bcast(&B, N, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);*

*for (row = 0; row < N; row++)*

*{*

*rownum[row] = row % total\_processes;*

*}*

*for (norm = 0; norm < N; norm++)*

*{*

*MPI\_Bcast(&A[norm][norm], N - norm, MPI\_DOUBLE, rownum[norm], MPI\_COMM\_WORLD);*

*MPI\_Bcast(&B[norm], 1, MPI\_DOUBLE, rownum[norm], MPI\_COMM\_WORLD);*

*for (row = norm + 1; row < N; row++)*

*{*

*if (rownum[row] == rank\_process)*

*{*

*multiplier[row] = A[row][norm] / A[norm][norm];*

*}*

*}*

*for (row = norm + 1; row < N; row++)*

*{*

*if (rownum[row] == rank\_process)*

*{*

*for (col = 0; col < N; col++)*

*{*

*A[row][col] = A[row][col] - (multiplier[row] \* A[norm][col]);*

*}*

*B[row] = B[row] - (multiplier[row] \* B[norm]);*

*}*

*}*

*}*

*for (norm = N-1; norm >= 0; norm--)*

*{*

*MPI\_Bcast(&X[norm], 1, MPI\_DOUBLE, rownum[norm], MPI\_COMM\_WORLD);*

*for (row = norm + 1; row >= 0; row--)*

*{*

*if (rownum[row] == rank\_process)*

*{*

*X[row] = B[row];*

*for (col = N - 1; col > row; col--)*

*{*

*X[row] -= A[row][col] \* X[col];*

*}*

*X[row] /= A[row][row];*

*}*

*}*

*}*

*}*

*void finalize() {*

*MPI\_Finalize();*

*exit(0);*

*}*

**Код програми Завдання**

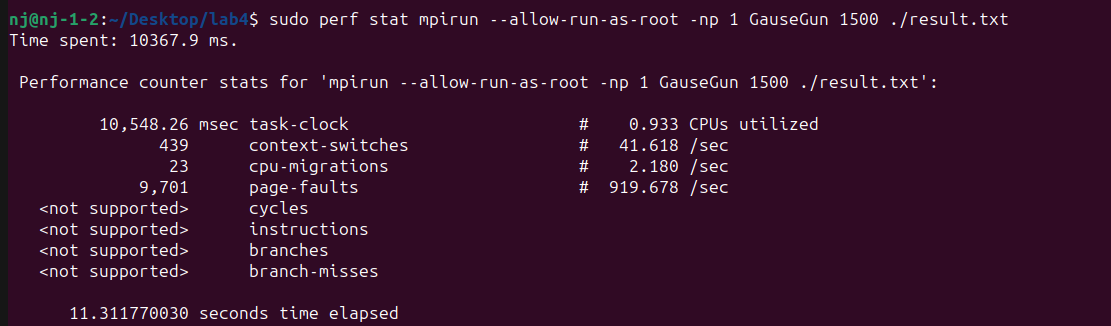


Table 2 Виконання завдання в однопоточному режимі

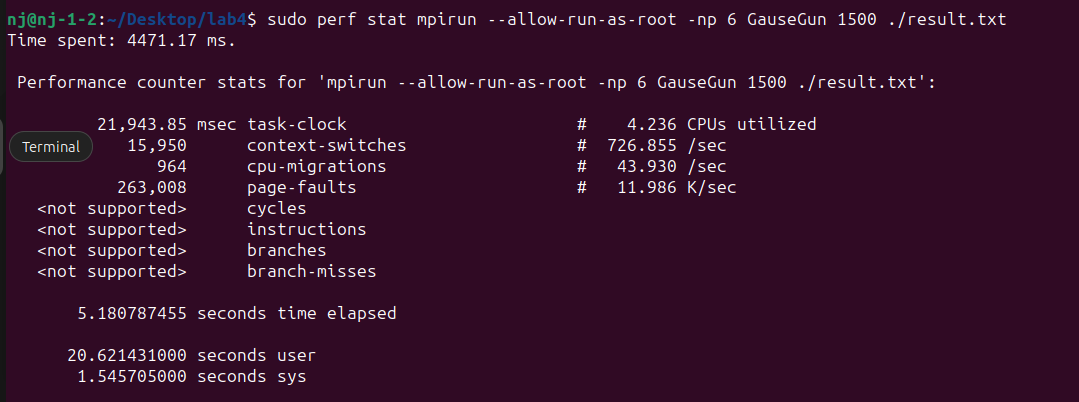


Table 4 Виконання Завдання а багатопоточному режимі

**Контрольні запитання:**

У чому полягає постановка задачі розв'язання системи лінійних рівнянь?

Постановка задачі розв'язання системи лінійних рівнянь полягає в формалізації процесу знаходження значень невідомих змінних, які задовольняють усі рівняння, що складають дану систему. У системі лінійних рівнянь кожне рівняння є лінійною функцією змінних, тобто кожне рівняння містить коефіцієнти, які множаться на змінні, а потім додаються або віднімаються. Постановка задачі включає в себе визначення кількості рівнянь та невідомих змінних у системі, а також представлення рівнянь у матричній або векторній формі для подальшого застосування алгоритмів розв'язання, таких як метод Гаусса, метод оберненої матриці тощо. Ця постановка дозволяє визначити, чи існують розв'язки системи, і як їх знайти.

Опишіть схему програмної реалізації паралельного варіанта методу Гауса.

У паралельній програмній реалізації методу Гауса використовується розподіл роботи між різними обчислювальними вузлами. Кожен вузол отримує підматрицю або частину даних, над якими він виконує операції методу Гауса локально. Після цього вузли можуть обмінюватись даними для вирішення залежностей між різними частинами матриці. Такий підхід дозволяє розпаралелити обчислення та зменшити час виконання алгоритму, особливо при великих розмірах матриць.