Федеральное государственное бюджетное образовательное учреждение высшего образования

«Уфимский Государственный Авиационный Технический Университет»

Кафедра ВВТиС

Отчет по дисциплине «Суперкомпьютерные технологии» по лабораторным работам №2-3

Выполнила студентка группы ПРО

Проверил старший преподаватель

Юлдашев А.В.

Уфа 2017

Вариант 5

Зависимость ускорения от числа процессоров выражается формулой

,

где – время выполнения параллельной программы на одном процессоре, – на процессорах. Отношение ускорения к количеству процессоров называется эффективностью при использовании p процессоров.

В таблицах представлены результаты ускорения и эффективности для различного количества процессов и N.

При работе с Open MP и MPI программа была протестирована на больших и малых наборах данных, сумма ряда всегда вычислялась одинаково.

**Open MP.**

**Результаты OpenMP:**

|  |  |  |  |
| --- | --- | --- | --- |
| Потоки (p) | Время работы (T) | Ускорение (S) | Эффективность (E) |
| 1 | 4.7520 | 1 | 1 |
| 2 | 4.6583 | 1.02011 | 0.510055 |
| 3 | 4.5862 | 1.03615 | 0.345383 |
| 4 | 4.5021 | 1.05551 | 0.263878 |

График зависимости времени работы (T) программы показывает, что при увеличении количества потоков время выполнения уменьшается.

График зависимости ускорения (S) программы показывает, что при

увеличении количества потоков ускорение выполнения увеличивается.

График зависимости эффективности (E) программы показывает, что при

увеличении количества потоков эффективность выполнения уменьшается.

**MPI.**

**Результаты MPI на локальной машине:**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество процессов(p) | Время работы (T) | Ускорение (S) | Эффективность (E) |
| 1 | 6.3422 | 1 | 1 |
| 2 | 4.0461 | 1.56748 | 0.78374 |
| 3 | 2.9675 | 2.16642 | 0.72214 |
| 4 | 1.9994 | 3.17682 | 0.71301 |

График зависимости времени работы (T) программы показывает, что при увеличении количества процессов время выполнения уменьшается.

График зависимости ускорения (S) программы показывает, что при

увеличении количества процессов ускорение выполнения увеличивается.

График зависимости эффективности (E) программы показывает, что при

увеличении количества процессов эффективность выполнения уменьшается.

**Результаты MPI на суперкомпьютере УГАТУ:**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество процессов | Время работы | Ускорение | Эффективность |
| 1 | 127.51 | 1 | 1 |
| 2 | 67.01 | 1.9028 | 0.9514 |
| 4 | 34.08 | 3.7417 | 0.9353 |
| 16 | 8.744 | 14.5825 | 0.9114 |
| 32 | 4.983 | 25.5890 | 0.7996 |
| 64 | 2.533 | 50.3232 | 0.7863 |
| 128 | 1.334 | 95.6032 | 0.7469 |

График зависимости времени работы программы от количества процессов

График зависимости времени работы программы показывает, что при увеличении процессов время выполнения программы равномерно уменьшается (причем время каждого последующего эксперимента уменьшается примерно в 2 раза).

График зависимости эффективности программы от количества процессов

График зависимости эффективности программы показывает, что при увеличении количества процессов эффективность выполнения уменьшается.

График зависимости ускорения программы от количества процессов

График зависимости ускорения программы показывает, что при увеличении количества процессов ускорение программы равномерно увеличивается (причем ускорение каждого последующего эксперимента увеличивается примерно в 2 раза).

**Заключение**

Были изучены методы распараллеливания программы средствами Open MP и MPI, выявлены основные характеристиками параллельной программы:

- время работы на n потоках/процессах;

- эффективность;

- ускорение.

Практическая часть выявила, что работа с потоками/ процессами может привести к быстродействию выполнения программы в разы. При работе с кластером ускорение не всегда было почти равно количеству процессов, это может объясняться тем, что программа не всегда хорошо масштабируется на используемых наборах данных.

Код программы Open MP:

#include"stdio.h"

#include<iostream>

#include"math.h"

#include<ctime>

#include"omp.h"

using namespace std;

int main()

{

long n = 1400000000;

double sum = 0;

double start, end, time;

start = omp\_get\_wtime();

#pragma omp parallel for reduction (+:sum)

for (int i = 1; i <= n, i++)

sum += pow(i, 1.0 / 3.0) / ((i + 1.0)\*sqrt(i));

end = omp\_get\_wtime();

time = end - start;

cout << "Result" << sum << endl;

cout << "Time" << time << endl;

system("pause");

return 0;

}

Код программы MPI:

#include <mpi.h>

#include <iostream>

#include <math.h>

#include <cstdio>

using namespace std;

const long long n = 1400000000;

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

{

int i, rank, size, namelen;

char name[MPI\_MAX\_PROCESSOR\_NAME];

double sum = 0;

MPI\_Init(&argc, &argv);

MPI\_Status stat;

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

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

double startt = MPI\_Wtime();

int m, start, width;

m = n%size;

width = n / size;

if (rank<m)

{

width++;

start = rank\*(n / size + 1);

}

else

{

start = m\*(n / size + 1) + (rank - m)\*n / size;

}

for (long long i = start + 2; i < start + width + 2; i++)

sum += (i % 2 ? -1 : 1)\*log(i) / double(i);

printf("%lf\n", sum);

if (rank == 0)

{

double s;

for (int i = 1; i < size; i++)

{

MPI\_Recv(&s, 1, MPI\_DOUBLE, i, 1, MPI\_COMM\_WORLD, &stat);

sum += s;

}

}

else

{

MPI\_Send(&sum, 1, MPI\_DOUBLE, 0, 1, MPI\_COMM\_WORLD);

}

double end = MPI\_Wtime();

double search\_time = end - startt;

cout << "sum = " << sum << endl;

cout << "time = " << search\_time << endl;

MPI\_Finalize();

}