

МГУ им. М. В. Ломоносова, факультет ВМК

Задание 2

Проблемы масштабируемости в MPI. Пренебрежение отложенными запросами на взаимодействие .

Арбузов Николай Романович
группа 423

Проблема

При реализации параллелизма в решении задач с помощью MPI пренебрежение отложенными запросами на взаимодействие повышает время выполнения задачи без влияния на качество выполнения программы.

Алгоритм

В качестве показательного алгоритма для этой проблемы был выбран алгоритм обмена массивами между процессами:

Проблемный код:

```
for (int i = 0; i < msg_size; i++)
{
    MPI_Isend(send_buf, msg_size, MPI_INT, (rank + size - 1 - (i % (size - 1))) % size, MSG_TAG, MPI_COMM_WORLD, &req_send);
    MPI_Irecv(&recv_buf[msg_size * i], msg_size, MPI_INT, (rank + 1 + (i % (size - 1))) % size, MSG_TAG, MPI_COMM_WORLD, &req_recv);

    MPI_Wait(&req_send, MPI_STATUS_IGNORE);
    MPI_Wait(&req_recv, MPI_STATUS_IGNORE);
    for (int j = 0; j < msg_size; j++)
    {
        if (recv_buf[msg_size * i + j] != (rank + 1 + (i % (size - 1))) % size)
        {
            std::cerr << "Error: received value " << recv_buf[msg_size * (i < rank
? i : i - 1) + j] << " from process " << i << std::endl;
            break;
        }
    }
}
```

Код без проблемы:

```
for (int i = 0; i < msg_size; i++)
{
    MPI_Send_init(send_buf, msg_size, MPI_INT, (rank + size - 1 - (i % (size - 1))) % size, MSG_TAG, MPI_COMM_WORLD, &send_reqs[i]);
    MPI_Recv_init(&recv_buf[msg_size * i], msg_size, MPI_INT, (rank + 1 + (i % (size - 1))) % size, MSG_TAG, MPI_COMM_WORLD, &recv_reqs[i]);
}

MPI_Startall(msg_size, send_reqs);
MPI_Startall(msg_size, recv_reqs);

MPI_Waitall(msg_size, send_reqs, statuses);
MPI_Waitall(msg_size, recv_reqs, statuses);

for (int i = 0; i < msg_size; i++)
{
    for (int j = 0; j < msg_size; j++)
    {
```

```

        if (recv_buf[msg_size * i + j] != (rank + 1 + (i % (size - 1))) % size)
        {
            std::cerr << "Error: received value " << recv_buf[msg_size * i + j] <<
" from process " << (rank + i + 1 + (i + 1) / size) % size << std::endl;
            break;
        }
    }
}

```

Компиляция и запуск

Все вычисления производились на машине Polus.

Сама программа написана на языке C++ и состоит из файлов:

- main.cpp / main_problem.cpp (в зависимости от того, как мы хотим запустить с без отложенных запросов на взаимодействие или с ними)

Компилировалась с использованием Makefile:

```

all: main

problem: *.cpp *.h
    mpicxx problem_main.cpp -o prog -std=c++11

main: *.cpp *.h
    mpicxx main.cpp -o prog -std=c++11

clean:
    rm -rf ./prog

```

Запуск производился постановкой в очередь с помощью планировщика mpisubmit.pl:

```

mpisubmit.pl -p $i -w 00:30 --stdout ./out_files/$j/$i.out --stderr
./err_files/$j/$i.err ./prog -- polus $j

```

Где *i* – количество процессов, на которых будет запускаться программа, а *j* – размер матрицы, на которой будут производиться вычисления.

Оценка результативности изменений проводилась по средством измерения времени с помощью функции gettimeofday(&Tp, NULL).

Результаты

| | Size | Nthreads | Time_Normal | Time_Problem | diff |
|---|-------|----------|--------------------|--------------------|----------------------|
| 0 | 10000 | 1 | 1.7664849999999999 | 1.058165 | -0.7083199999999998 |
| 1 | 10000 | 2 | 1.444425 | 1.2770325 | -0.16739250000000006 |
| 2 | 10000 | 4 | 2.5417525 | 1.7839775 | -0.7577749999999999 |
| 3 | 10000 | 8 | 1.57318 | 2.4924850000000003 | 0.9193050000000003 |
| 4 | 10000 | 16 | 2.3740075000000003 | 3.1709050000000003 | 0.7968975 |

| | | | | | |
|---|-------|----|--------------------|------------|----------------------|
| 5 | 25000 | 1 | 7.1447775 | 6.6493825 | -0.49539500000000025 |
| 6 | 25000 | 2 | 17.8770275 | 7.8975275 | -9.979500000000002 |
| 7 | 25000 | 4 | 23.048907500000002 | 10.4532325 | -12.595675000000002 |
| 8 | 25000 | 8 | 19.1776575 | 13.2277 | -5.949957499999998 |
| 9 | 25000 | 16 | 14.98165 | 18.709 | 3.7273499999999995 |

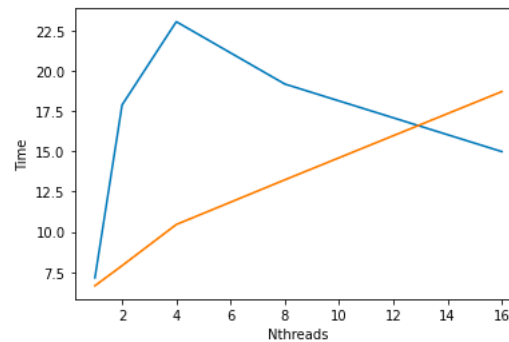
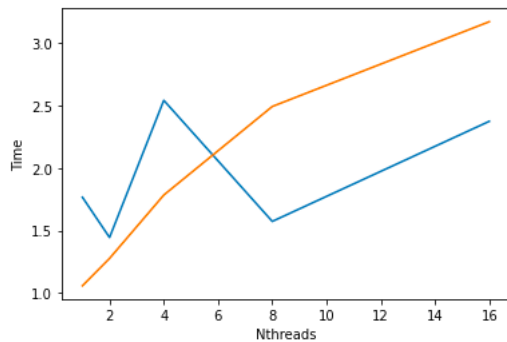


Рисунок 1. Графики времени выполнения программы: Синим цветом без использования операторов инициации пересылки, оранжевым цветом – с ними. Графика по порядку для матриц размером 10000 и 25000.

По графикам видно, что с увеличением числа процессов программа без использования отложенных запросов работает менее эффективно, это связано с тем, что при большем числе процессов иницируется больше запросов на взаимодействие