Отчёт

по

Самостоятельной Работе

Преподаватель: Кулябов Дмитрий Сергеевич

Выполнил: Кремер Илья

Группа: НК-401

Москва, РУДН – 2011

Оглавление

Введение

Этот отчёт содержит только самые необходимые комментарии и соответствует последовательности выполнения работы.

Постановка задачи

Разработать алгоритм передачи данных между процессами по кругу с помощью MPI. Пусть работает M+1 процесс:

1. 0-ой процесс отправляет сообщение 1-ому, а сам начинает ждать сообщение от M-ого;
2. 1-ый процесс получает сообщение от 0-ого и отправляет его 2-ому;
3. 2-ой процесс получает сообщение от первого и отправляет его 3-ему;
4. и т.д. до M-ого процесса.

Протестировать программу на скорость и построить графики.

Решение

Разберём основной код программы:

```
if (np == 1) then
    print *, "BAD NP"
    call mpi_finalize(ierr)
    stop
endif

if (pid == 0) then
    start = mpi_wtime()
    call mpi_send(a, N, MPI_INTEGER, 1,      5, MPI_COMM_WORLD, ierr)
    call mpi_recv(a, N, MPI_INTEGER, np - 1, 5, MPI_COMM_WORLD, status, ierr)
    ! вывод сообщения
    print '(a, f12.8, a)', "затрачено   ", mpi_wtime() - start, "   сек"
else if (pid == np - 1) then
    next = 0
    prev = pid - 1
else
    next = pid + 1
    prev = pid - 1
endif

if (pid /= 0) then
    call mpi_recv(a, N, MPI_INTEGER, prev, 5, MPI_COMM_WORLD, status, ierr)
    ! тут можно вставить какое-либо изменение сообщения
    call mpi_send(a, N, MPI_INTEGER, next, 5, MPI_COMM_WORLD, ierr)
endif
```

Вначале мы избегаем проблем, связанных с неверным запуском – количество процессов не должно быть равно единице (отправка сообщения самому себе не имеет особого смысла в соответствии с условием задачи).

Затем, в нулевом процессе начинаем цепь передачи. Там же замеряем скорость. Отправляем сообщение первому процессу и ждём его с последнего. Как только мы получим сообщение – выведем сообщение о затраченном времени и само сообщение, если это необходимо.

Дальше мы определяем значения переменных `next` и `prev`. Поскольку для нулевого процесса их вычислять уже не нужно, то у нас будет два случая: последний процесс или любой другой.

Ставим блок `if`, чтобы больше не затрагивать работу нулевого процесса. Каждый процесс делает одно и то же: принимает сообщение с предыдущего и передаём следующему.

Тесты

В виртуальной машине, установленной на ОС Windows 7 были произведены запуски программы для 100 и 10 процессов. Длины сообщения – 1, 15.000 и 1.500.000. Всего на каждую длину сообщения с выбранным количеством процессов было проведено 9 испытаний. То есть всего 9 * 3 * 2 = 54 запуска программы.

Для длины сообщения в одну переменную (100 процессов) среднее время прохода цикла передач было 1,19 сек. При этом длительность самого первого запуска была самой большой и выделялась среду других: 2,08 секунды.

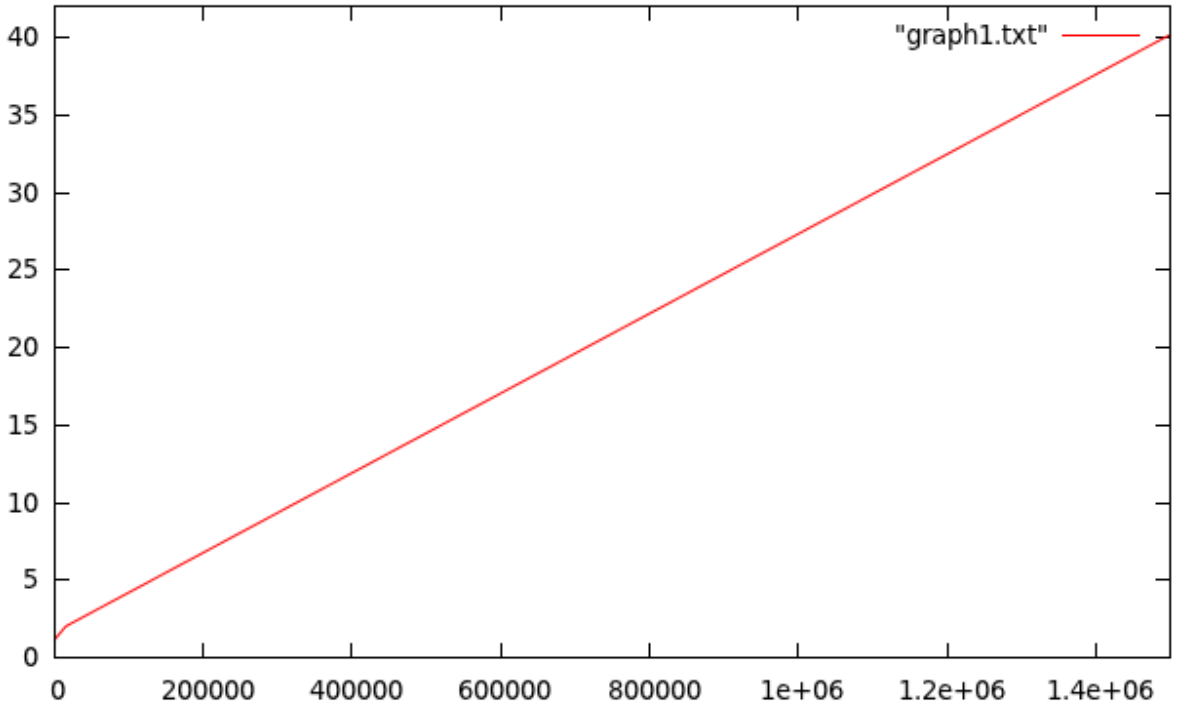
Для 15-ти тысяч переменных (100 процессов) в массиве среднее время составило 2,44 секунды, при этом опять первый запуск стремился завысить это показание: 5,73 секунды.

Для сообщения в полтора миллиона переменных (100 процессов) среднее время составило 40,16 секунд и разница между каждым запуском не была так принципиально заметна, хотя разность между самым медленным и быстрым запусками составила 28% от среднего значения.

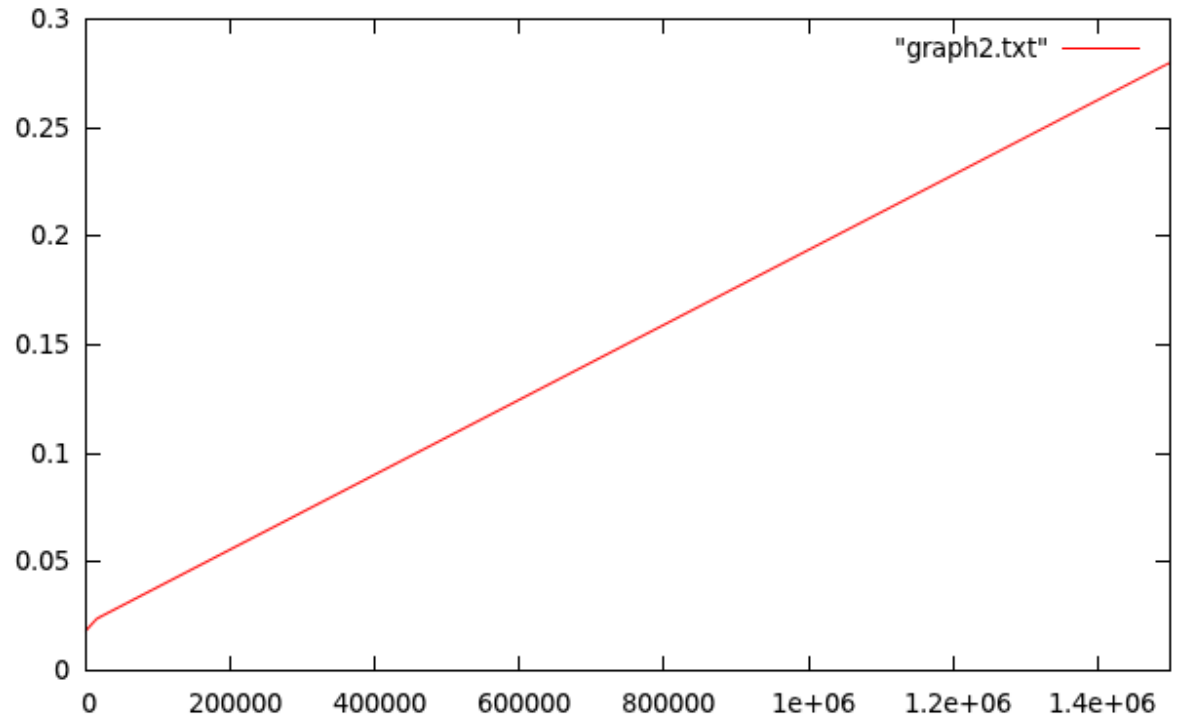
Для 10 процессов имеем следующие данные (в том же порядке): 0,0185 сек; 0,024 сек; 0,28 сек.

Графики

Ниже представлены два графика. По осям абсцисс – количество переменных в сообщении, по ординатам – время выполнения цикла передач:



100 процессов



10 процессов

Хорошо видно, что пропорциональность почти сохраняется.

Литература и др. источники

1. А.С. Антонов, «Параллельное программирование с использованием технологии MPI», – издательство Московского Университета, 2004.
2. http://parallel.ru – Информационно-аналитический центр.