

Практикум по параллельному программированию

Задание [MPI-4]

Гаврилов Олег Алексеевич
МГУ-Саров

Описание проблемы. Рассматривается организация пересылок данных между двумя процессами MPI. Проблема поэлементной пересылки заключается в латентности коммуникационной сети. Известно, что время на подготовку к передаче данных по каналу связи не зависит от их объёма, поэтому при большом числе пересылок (то есть при малом объёме данных в одной пересылке) увеличивается задержка, связанная с латентностью. Решением такой проблемы масштабируемости является удлинение пересылок до некоторого объёма, близкого к оптимальному.

Проблемный код. Рассматривается пример с двумя процессами, один из которых пересылает другому массив длины N по блокам, общее число которых равно n . Например, если $n = N$, то массив пересылается поэлементно; если $n = 1$, то за один раз передаются все элементы массива. Ниже представлен фрагмент программы на C++.

```
double start = MPI_Wtime();

if (rank == 0) {
    int* a = new int[N];

    //n - number of blocks of data (we suppose that N is divisible by n)
    for (int i = 0; i < n; ++i)
        MPI_Recv(a+i*N/n, N/n, MPI_INT, 1, 111, MPI_COMM_WORLD, &status);

    delete[] a;
}

if (rank == 1) {
    //initialization
    int* a = new int[N];
    for (int i = 0; i < N; ++i)
        a[i] = i;

    //sending the array by blocks of data
    for (int i = 0; i < n; ++i)
        MPI_Send(a + i*N/n, N/n, MPI_INT, 0, 111, MPI_COMM_WORLD);

    delete[] a;
}

double end = MPI_Wtime();

if (rank == 0)
    printf("N = %u. Number of blocks: %u. Time: %f s\n", N, n, end-start);
```

Результаты серии испытаний. Пусть $N = 10.000.000$. Запуск проводился на двух процессах на полигоне ЦХАБТ. Конфигурация: четыре 36-ядерных узла на базе Intel Xeon Gold 6140 @ 2.30 GHz, суммарно 144 физических ядра.

n (число блоков)	N/n (элементов в блоке)	Время, сек
1	10.000.000	0.086079
10	1.000.000	0.089011
100	100.000	0.116041
1000	10.000	0.109842
10.000	1.000	0.111924
100.000	100	0.377222
1000.000	10	3.603154
10.000.000	1	36.827669

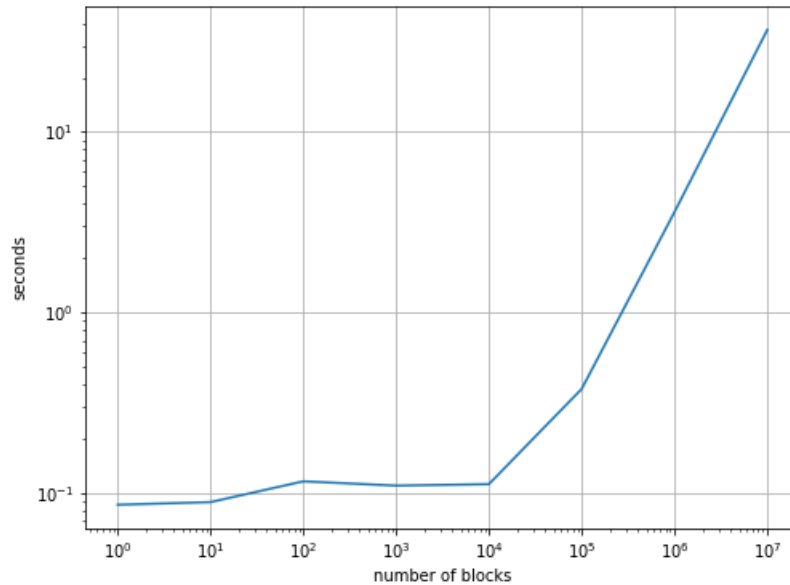
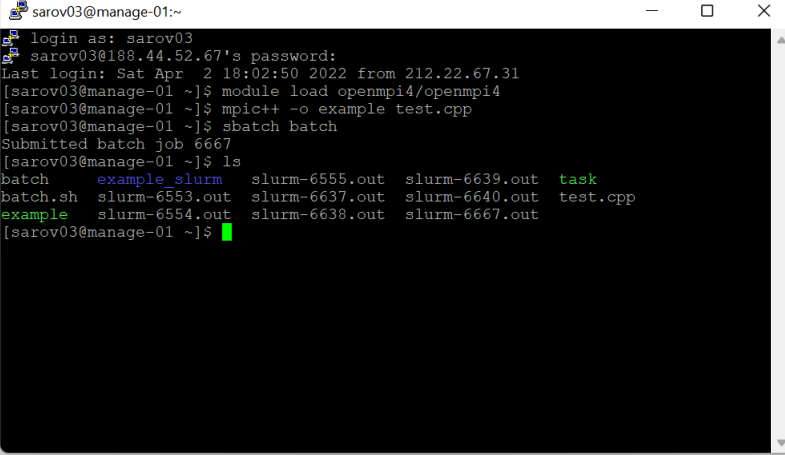


Рис. 1: Зависимость времени от числа блоков данных (с логарифмическими осями).

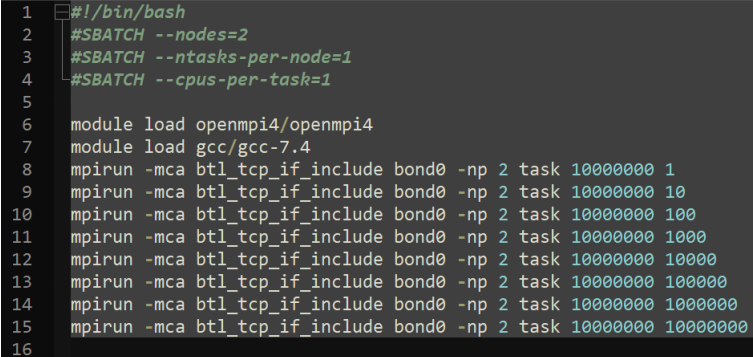
Исходная ("плохая") версия программы – случай $n = N$, то есть пересылка массива происходит поэлементно. В этом случае затрачивается много времени из-за латентности коммуникационной сети, поскольку при каждом вызове `MPI_Send` или `MPI_Recv` имеют место задержки, не зависящие от объёма передаваемых данных и которые нельзя уменьшить. Оптимизированный случай – когда n существенно меньше N . Из таблицы и графика видно, что для нашего примера хороший результат дают значения $n = 1$ или $n = 10$, когда за один раз передаётся весь массив либо 1/10 часть.

Компиляция проводилась при помощи `mpicc++`, который используется для MPI-программ. Перед этим необходимо загрузить модуль `openmpi4`. После компиляции запускается `batch`-файл с заранее написанными командами. Ниже приведён порядок компиляции и запуска программы.



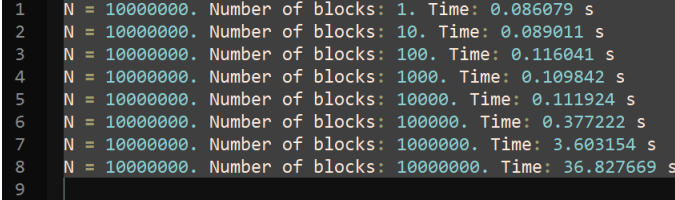
```
sarov03@manage-01:~  
login as: sarov03  
sarov03@188.44.52.67's password:  
Last login: Sat Apr 2 18:02:50 2022 from 212.22.67.31  
[sarov03@manage-01 ~]$ module load openmpi4/openmpi4  
[sarov03@manage-01 ~]$ mpicc++ -o example test.cpp  
[sarov03@manage-01 ~]$ sbatch batch  
Submitted batch job 6667  
[sarov03@manage-01 ~]$ ls  
batch      example slurm      slurm-6555.out  slurm-6639.out  task  
batch.sh    slurm-6553.out  slurm-6637.out  slurm-6640.out  test.cpp  
example     slurm-6554.out  slurm-6638.out  slurm-6667.out  
[sarov03@manage-01 ~]$
```

Рис. 2: Порядок компиляции и запуска на сервере.



```
1 #!/bin/bash  
2 #SBATCH --nodes=2  
3 #SBATCH --ntasks-per-node=1  
4 #SBATCH --cpus-per-task=1  
5  
6 module load openmpi4/openmpi4  
7 module load gcc/gcc-7.4  
8 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 1  
9 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 10  
10 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 100  
11 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 1000  
12 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 10000  
13 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 100000  
14 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 1000000  
15 mpirun -mca btl_tcp_if_include bond0 -np 2 task 10000000 10000000  
16
```

Рис. 3: Содержимое `batch`-файла.



```
1 N = 10000000. Number of blocks: 1. Time: 0.086079 s  
2 N = 10000000. Number of blocks: 10. Time: 0.089011 s  
3 N = 10000000. Number of blocks: 100. Time: 0.116041 s  
4 N = 10000000. Number of blocks: 1000. Time: 0.109842 s  
5 N = 10000000. Number of blocks: 10000. Time: 0.111924 s  
6 N = 10000000. Number of blocks: 100000. Time: 0.377222 s  
7 N = 10000000. Number of blocks: 1000000. Time: 3.603154 s  
8 N = 10000000. Number of blocks: 10000000. Time: 36.827669 s  
9
```

Рис. 4: Вывод программы.