

Санкт-Петербургский политехнический университет Петра Великого
Высшая Школа Прикладной Математики и Вычислительной Физики
Секция «Телематика»

Отчет
по дисциплине «Параллельное программирование на суперкомпьютерных системах»

Выполнил: _____ Студент Жэнь Мэнгуан, гр. 3540201/20301
Проверил: _____ Руководитель Лукашин А.А.

Санкт-Петербург
2022 год

Содержание

Введение	3
1 Математическое описание метода сопряженных градиентов	4
2 Суперкомпьютерный центр «Политехнический»	5
2.1 Состав и характеристики	5
2.2 Технология подключения	5
2.3 Настройка окружения	6
2.4 Запуск задач	7
3 Распараллеливание алгоритма метода сопряженных градиентов	8
4 Исследование результата	9
4.1 Проверки корректности алгоритма	9
4.2 Результат работы	10
5 Заключение	16

Введение

В этом задании реализовал алгоритм метода сопряженных градиентов на суперкомпьютере. А алгоритм метода сопряженных градиентов распараллеливается следующим методом.

- C & Linux pthreads
- C & MPI
- Python & MPI
- C & OpenMP

Наконец, сравниваются и анализируются результаты реализации метода сопряженных градиентов различными методами.

1 Математическое описание метода сопряженных градиентов

Метод сопряженных градиентов — численный метод решения систем линейных алгебраических уравнений, является итерационным методом Крыловского типа.

Пусть дана система линейных уравнений: $Ax = b$. Причём матрица системы — это действительная матрица, обладающая следующими свойствами: $A = A^T > 0$, то есть это симметричная положительно определённая матрица. Тогда процесс решения СЛАУ можно представить как минимизацию следующего функционала:

$$(Ax, x) - 2(b, x) \rightarrow \min$$

Шаги алгоритма метода сопряженных градиентов показаны вниз на Рис 1.1:

```
r0 := b − Ax0
p0 := r0
k := 0
repeat
    
$$\alpha_k := \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A} \mathbf{p}_k}$$

    xk+1 := xk + αkpk
    rk+1 := rk − αkApk
    if rk+1 is sufficiently small, then exit loop
    
$$\beta_k := \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$$

    pk+1 := rk+1 + βkpk
    k := k + 1
end repeat
```

Рис. 1.1: Алгоритм метода сопряженных градиентов

Поскольку минимизируемый функционал квадратичный, то процесс должен дать ответ на *n*-й итерации, однако при реализации метода на компьютере существует погрешность представления вещественных чисел, в результате чего может потребоваться и больше итераций. Так же оценивать точность можно по относительной невязке; $\frac{\|r^k\|}{\|b\|}$, и прекращать итерационный процесс, когда невязка станет меньше заданного числа.

2 Суперкомпьютерный центр «Политехнический»

2.1 Состав и характеристики

В качестве вычислительного ресурса для решения поставленной задачи используется суперкомпьютерный комплекс Санкт-Петербургского Политехнического Университета.

В вычислительном центре представлены три суперкомпьютера:

- «Политехник - РСК Торнадо» - кластер с пиковой производительностью 943 Тфлопс; содержит 612 узлов с прямым жидкостным охлаждением серии "Торнадо" (производитель "РСК Технологии РФ), имеющие каждый два CPU Intel Xeon E5-2697 v3 (14 ядер, 2.6 ГГц) и 64 ГБ оперативной памяти DDR4 и 56 узлов с прямым жидкостным охлаждением серии "Tornado" содержащие каждый два CPU Intel Xeon E5-2697 v3 и два ускорителя вычислений NVIDIA Tesla K40X, 64 ГБ оперативной памяти DDR4.
- «Политехник - РСК ПетаСтрим» - массивно-параллельный компьютер с ультравысокой многопоточностью; единственная в России система, способная поддерживать более 70 тыс. потоков; пиковая производительность 291 Тфлопс; содержит 288 узлов на сопроцессорах Intel Xeon Phi;
- «Политехник - NUMA» - массивно-параллельная система с кеш-когерентной глобально адресуемой памятью объемом более 12 ТБ; содержит 64 узла, 192 процессора; пиковая производительность 30 ТФлопс.

Все вычислительные системы работают с общей системой хранения данных (Seagate ClusterStore, 1.1ПБ), имеют единую систему управления и мониторинга.

2.2 Технология подключения

Одключение к суперкомпьютеру возможно при наличии логина и ключа доступа, который выдаётся сотрудниками суперкомпьютерного центра. С помощью этого ключа и производится доступ к суперкомпьютеру вместо использования пароля.

В данном задании использовать VScode и его соответствующие расширения для подключения к суперкомпьютеру.

Как показано на рисунке b, отредактировать файл конфигурации config ssh, выбирать указанный закрытый ключ ssh, а также пользователя и его адрес для подключения.

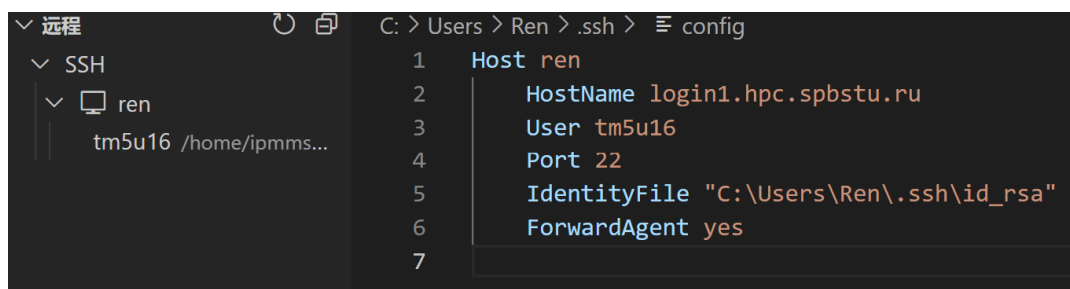


Рис. 2.1: Одключение к суперкомпьютеру 1

Результат подключения показан ниже. Причина выбора VScode заключается в том, что VScode предоставляет хороший рабочий интерфейс, который удобен для передачи файлов между локальным компьютером и суперкомпьютером, а VScode предоставляет хорошую функцию редактирования документов.

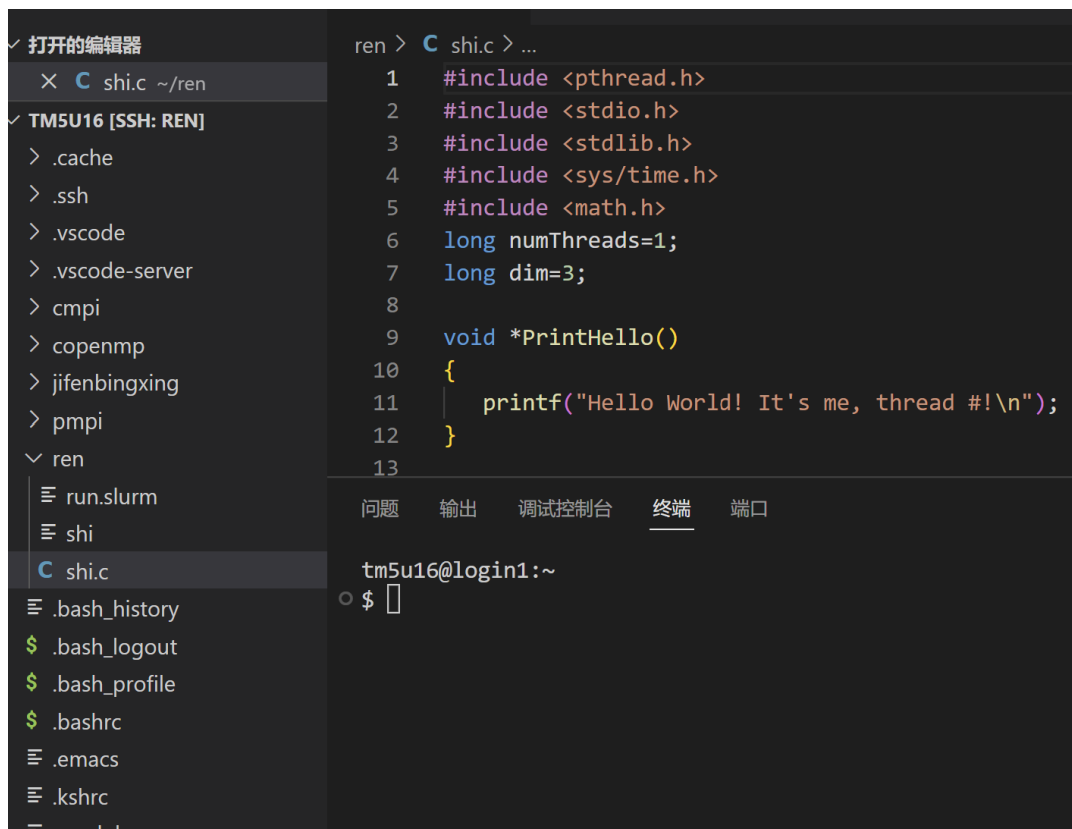


Рис. 2.2: Одключение к суперкомпьютеру 2

2.3 Настройка окружения

Для управления окружением на ресурсах СКЦ «Политехнический» используется система так называемых Environment Modules. С помощью них модифицируется окружение пользователя. Список доступных модулей можно посмотреть так:

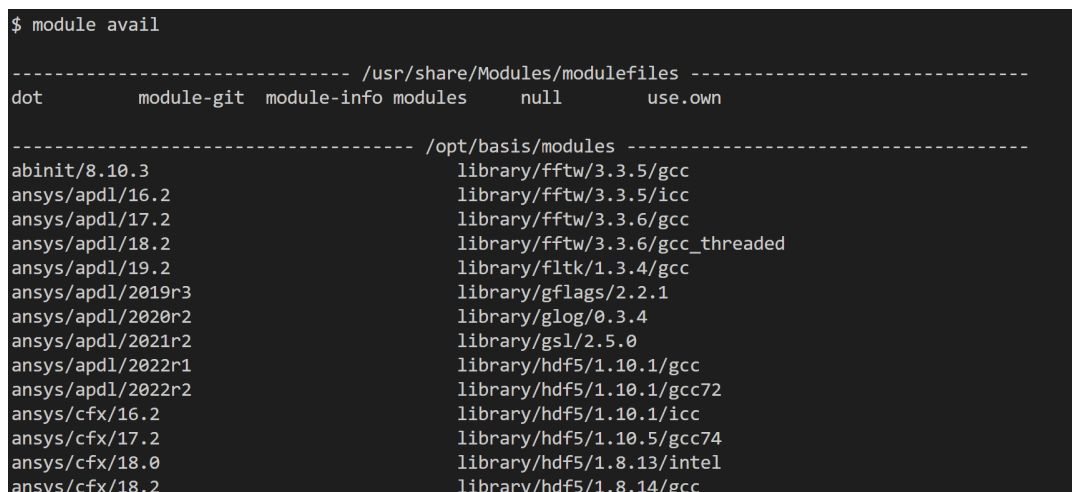


Рис. 2.3: Настройка окружения

Таким образом можно выбрать наборы компиляторов, библиотек а также предустановленного программного обеспечения. Загрузка требуемых модулей осуществляется с помощью команды `module load`

```
$ module load compiler/gcc/6.2.0
```

2.4 Запуск задач

Управление ресурсами кластера осуществляется с помощью программного пакета SLURM. Принцип его работы можно описать следующим образом: пользователь запрашивает некоторый ресурс (процессорные ядра, память и т.п.), размещая свою задачу в очереди; система, основываясь на приоритетах пользователя и текущем заполнении очереди, выбирает момент запуска задачи. Под очередью понимается последовательность задач, которая должны решаться на определенном вычислительном ресурсе (группе узлов).

При этом, каждый узел в текущий момент времени может быть занят только одной задачей, одного пользователя; таким образом узел отводится в монопольное использование размещенной на нем задачи, т.е. другие задачи на занятом узле выполняться не будут.

Пример скрипта `sbatch` для компиляции и выполнения файла:

```
cmpi > $ run.sl
1  #!/bin/bash
2  #SBATCH --nodes=1
3  #SBATCH -p tornado-k40
4  #SBATCH -t 00-02:00:00
5  #SBATCH -J Ren_cgm
6  #SBATCH -o ./result.txt
7  #SBATCH -e ./error.txt
8  if [ -f /etc/profile.d/modules-basis.sh ];
9  then
10 source /etc/profile.d/modules-basis.sh
11 fi
12 module purge
13 module load compiler/gcc/9
14 module add mpi/openmpi/4.0.1/gcc/9
15
16 mpicc cmpi.c -o cmpi
17 mpiexec -n 20 ./cmpi
```

Рис. 2.4: Запуск задач

командный файл можно поставить в очередь с помощью команды `sbatch`:

```
$ sbatch run.sl
```

3 Распараллеливание алгоритма метода сопряженных градиентов

Видно из рисунка 1.1. Алгоритм метода сопряженного градиента содержит большое количество повторяющихся операций между векторами и матрицами. Следовательно, общий алгоритм метода градиента можно оптимизировать для ускорения за счет распараллеливания этих матричных и векторных операций.

Среди этих операций между векторами и матрицами умножение векторов и матриц является наиболее сложным (алгоритм не включает умножение между матрицами) Поэтому в этом задании мы в основном изучаем влияние различных методов на результаты распараллеливания путем распараллеливания умножения между матрицами и векторами.

Когда вектор умножается на матрицу, результатом является вектор. Чтобы распараллелить эту операцию, можно заставить каждый поток вычислять часть элементов результирующего вектора. Вычисления с несколькими потоками могут ускорить вычисления.

Код выглядит следующим образом:

```
void MultMatrix(int id,int numprocs,struct Matrix mul1,struct Matrix mul2,struct Matrix prod){
    resetMatrix(prod);
    int i,j,k,X,Y;
    long num=dim/numprocs+1;
    long indexProd=id*num;
    for(i=0;i<num;i++){
        if(indexProd+i==prod.x*prod.y)
            break;
        X=(indexProd+i)/prod.y;
        Y=(indexProd+i)%prod.y;
        double sum=0;
        for(j=0;j<mul1.y;j++){
            sum+=mul1.m[X*mul1.y+j]*mul2.m[j*mul2.y+Y];
        }
        prod.m[indexProd+i]=sum;
    }
}
```

Рис. 3.1: Умножение матрицы на вектор

4 Исследование результата

4.1 Проверки корректности алгоритма

Поскольку алгоритм метода сопряженного градиента требует, чтобы матрица коэффициентов A была положительно определенной матрицей. Поэтому в данной работе реализован алгоритм построения положительно определенных матриц.

```
void*positive_definite_matrix(double*m){
    int i, j;
    double sum = 1;
    for (i = 0;i < dim;i++) {
        for (j = 0;j < dim;j++) {
            if (j < i)
                m[i * dim + j] = 0;
            else
                m[i * dim + j] = sum++;
        }
    }
}
```

Рис. 4.1: алгоритм построения положительно определенных матриц

Положительно определенные матрицы, сгенерированные этим алгоритмом, имеют следующий вид.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

Первый элемент матрицы равен единице, а затем постепенно увеличивается на единицу. Элементы ниже диагонали матрицы равны 0. Из определения положительно определенной матрицы мы знаем, что этот тип матрицы является положительно определенной.

В этом задании предполагается, что все элементы вектора B равны 10, а элементы исходного вектора решения X равны 0. В силу особенностей матрицы коэффициентов A . Последние несколько элементов вектора решения X могут быть легко получены по теореме Гаусса. Поскольку размер матрицы X слишком велик, в этой работе последние две цифры вектора X находятся с помощью теоремы Гаусса. Затем сравнить его с двумя последними элементами вектора X , рассчитанными алгоритмом сопряженного градиента, чтобы проверить точность алгоритма.

При размерах вектора 10, 50 и 100 максимальное количество итераций не превышает 20, и результаты показаны ниже.

```
Количество итерации:20
Метод Гаусса:х(n):0.181818,х(n-1)=0.003431
Метод сопряженных градиентов:х(n)=0.181858,х(n-1)=0.002904
Размерность матрицы:10
Количество потоков:1
Время:0.001469s
```

Рис. 4.2: Размер 10

```
Количество итерации:20
Метод Гаусса:х(n):0.007843,х(n-1)=0.000006
Метод сопряженных градиентов:х(n)=0.007843,х(n-1)=0.000007
Размерность матрицы:50
Количество потоков:1
Время:0.003175s
```

Рис. 4.3: Размер 50

```
Количество итерации:20
Метод Гаусса:х(n):0.001980,х(n-1)=0.000000
Метод сопряженных градиентов:х(n)=0.001980,х(n-1)=0.000001
Размерность матрицы:100
Количество потоков:1
Время:0.004853s
```

Рис. 4.4: Размер 100

Результаты испытаний показывают, что результаты, полученные двумя методами, практически совпадают. Разница может быть вызвана недостаточным количеством итераций алгоритма метода сопряженного градиента. Таким образом, можно сделать вывод, что реализованный алгоритм метода сопряженных градиентов эффективен.

4.2 Результат работы

Установить размерность матрицы коэффициентов A на 1000, Значение элемента вектора B равно 100, максимальное количество итераций 20, использовать 1, 2, 4 и 8 потоков в разных методах для сравнения времени алгоритма.

Результаты при использовании технологии `c++thread` показаны вниз:

```
1  Количество итерации:20
2  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
3  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
4  Размерность матрицы:1000
5  Количество потоков:1
6  Время:0.228925s
```

Рис. 4.5: Количество потоков 1 при c+pthread

```
1  Количество итерации:20
2  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
3  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
4  Размерность матрицы:1000
5  Количество потоков:2
6  Время:0.149630s
```

Рис. 4.6: Количество потоков 2 при c+pthread

```
1  Количество итерации:20
2  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
3  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
4  Размерность матрицы:1000
5  Количество потоков:4
6  Время:0.121615s
```

Рис. 4.7: Количество потоков 4 при c+pthread

```
1  Количество итерации:20
2  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
3  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
4  Размерность матрицы:1000
5  Количество потоков:8
6  Время:0.108339s
```

Рис. 4.8: Количество потоков 8 при c+pthread

Результаты при использовании технологии c+mpi показаны вниз:

```

18  Количество итерации:20
19  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
20  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
21  Размерность матрицы:1000
22  Количество потоков:1
23  Время:0.180364s

```

Рис. 4.9: Количество потоков 1 при c+mpi

```

18  Количество итерации:20
19  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
20  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
21  Размерность матрицы:1000
22  Количество потоков:2
23  Время:0.142223s

```

Рис. 4.10: Количество потоков 2 при c+mpi

```

18  Количество итерации:20
19  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
20  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
21  Размерность матрицы:1000
22  Количество потоков:4
23  Время:0.093607s

```

Рис. 4.11: Количество потоков 4 при c+mpi

```

18  Количество итерации:20
19  Метод Гаусса: $x(n):0.000200, x(n-1)=0.000000$ 
20  Метод сопряженных градиентов: $x(n)=0.000109, x(n-1)=0.000074$ 
21  Размерность матрицы:1000
22  Количество потоков:8
23  Время:0.098391s

```

Рис. 4.12: Количество потоков 8 при c+mpi

Результаты при использовании технологии c+OpenMP показаны вниз:

```
1  Количество итерации:20
2  Размерность матрицы:1000
3  Количество потоков:1
4  Время:0.424195s
```

Рис. 4.13: Количество потоков 1 при c+OpenMP

```
1  Количество итерации:20
2  Размерность матрицы:1000
3  Количество потоков:2
4  Время:0.719849s
```

Рис. 4.14: Количество потоков 2 при c+OpenMP

```
1  Количество итерации:20
2  Размерность матрицы:1000
3  Количество потоков:4
4  Время:0.846315s
```

Рис. 4.15: Количество потоков 4 при c+OpenMP

```
1  Количество итерации:20
2  Размерность матрицы:1000
3  Количество потоков:8
4  Время:0.991053s
```

Рис. 4.16: Количество потоков 8 при c+OpenMP

Результаты при использовании технологии Python+mpi показаны вниз:

```
18    Количество итерации20
19    Размерность матрицы:1000
20
21    Количество потоков:1
22    Время:0.987254s
```

Рис. 4.17: Количество потоков 1 при Python+mpi

```
21    Количество итерации20
22    Размерность матрицы:1000
23
24    Количество потоков:2
25    Время:0.251595s
```

Рис. 4.18: Количество потоков 2 при Python+mpi

```
21    Количество итерации20
22    Размерность матрицы:1000
23
24    Количество потоков:4
25    Время:0.077636s
```

Рис. 4.19: Количество потоков 4 при Python+mpi

```
21     Количество итерации:20
22     Размерность матрицы:1000
23
24     Количество потоков:8
25     Время:0.243190s
```

Рис. 4.20: Количество потоков 8 при Python+mpi

5 Заключение

Реализовал алгоритм сопряженного градиента. При размерности исходной матрицы коэффициентов A 1000 и значении постоянного вектора 100. Последние два элемента полученного вектора решения X равны соответственно 0.000109 и 0.000074. Сравнение времени работы алгоритма при использовании разных методик показано ниже.

Количество потоков / Метод	c+pThread	c+mpi	c+openmp	Python+mpi
1	0.23s	0.18s	0.42s	0.99s
2	0.15s	0.14s	0.72s	0.25s
4	0.12s	0.09s	0.85s	0.07s
8	0.10s	0.10s	1.00s	0.24s

Помимо использования технологии openmp, чем больше потоков, тем короче время. Среди них эффект c+mpi лучший.

Приложения

https://github.com/ren-m-g/parallel_program