МГУ им. М. В. Ломоносова, факультет ВМК

Задание 3 Многопоточная реализация солвера CG для СЛАУ с разреженной матрицей, заданной в формате ELLPACK.

Арбузов Николай Романович
группа 323

Постановка задачи

Требуется написать параллельную программу с использованием технологии OpenMP для решения системы линейных уравнений $\boldsymbol{Ax} = \boldsymbol{b}$ методом сопряженных градиентов.

Алгоритм

Алгоритм является итерационным и выполняется до тех пор, пока не будет достигнута необходимая точность или не будет превышено максимально допустимое число итераций.

Алгоритм предобусловленного метода CG имеет следующий вид:

Choose an initial guess x_0;
$r_0 = b - Ax_0$;
convergence = false;
k = 1;
repeat
  $z_k = M^{-1} r_{k-1}$;
  $\rho_k = (r_{k-1}, z_k)$;
  if $k = 1$ **then**
    $p_k = z_k$;
  else
    $\beta_k = \rho_k / \rho_{k-1}$;
    $p_k = z_k + \beta_k p_{k-1}$;
  end if
  $q_k = Ap_k$;
  $\alpha_k = \rho_k / (p_k, q_k)$;
  $x_k = x_{k-1} + \alpha_k p_k$;
  $r_k = r_{k-1} - \alpha_k q_k$;
  if $(\rho_k < \varepsilon)$ **or** $(k \geq maxiter)$ **then**
    convergence = true;
  else
    k = k + 1;
  end if
until convergence

В случае предобуславливателя Якоби матрица M – диагональная матрица,

с диагональю из матрицы A. Начальное приближение – нулевое.

Так как наша матрица по алгоритму построения является матрицей с диагональным преобладанием, было принято решение использовать алгоритм без предобуславливателя.

Компиляция и запуск

Все вычисления производились на машине Polus.

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

- main.cpp
- CG.cpp
- CG.h
- matrix.cpp
- matrix.h

Компилировалась с использованием Makefile:

```
all: main

main: *.cpp *.h
    g++ *.cpp -o prog -std=c++17 -fopenmp

omp_polus: *.cpp *.h
    xlc++ *.cpp -o prog -Wall -std=c++11 -qsmp=omp -fopenmp

clean:
    rm -rf ./prog
```

Запуск производился постановкой в очередь с помощью lsf-файлов вида:

Для 1 и 2 потоков:

```
#BSUB -n 1
#BSUB -W 00:30
#BSUB -o \"./out_files/j/i.out\"
#BSUB -e \"./err_files/j/i.err\"
#BSUB -R \"span[hosts=1]\"
OMP_NUM_THREADS=i ./prog j
```

Для 4, 8, 16 и 32 потоков:

```
#BSUB -W 00:15
#BSUB -o "./out_files/j/i.out"
#BSUB -e "./err_files/j/i.err"
#BSUB -R "affinity[core(M)]"
OMP_NUM_THREADS=i
/polusfs/lsf/openmp/launchOpenMP.py ./prog j
```

Где **i** – количество потоков, на которых будет запускаться программа, **M** – количество ядер (**M = i / 2**), **j** – размер матрицы, на которой будут производиться вычисления.

Результаты

Тесты проводились для кубических матриц с размерами 100x100x100, 200x200x200 и 250x250x250.

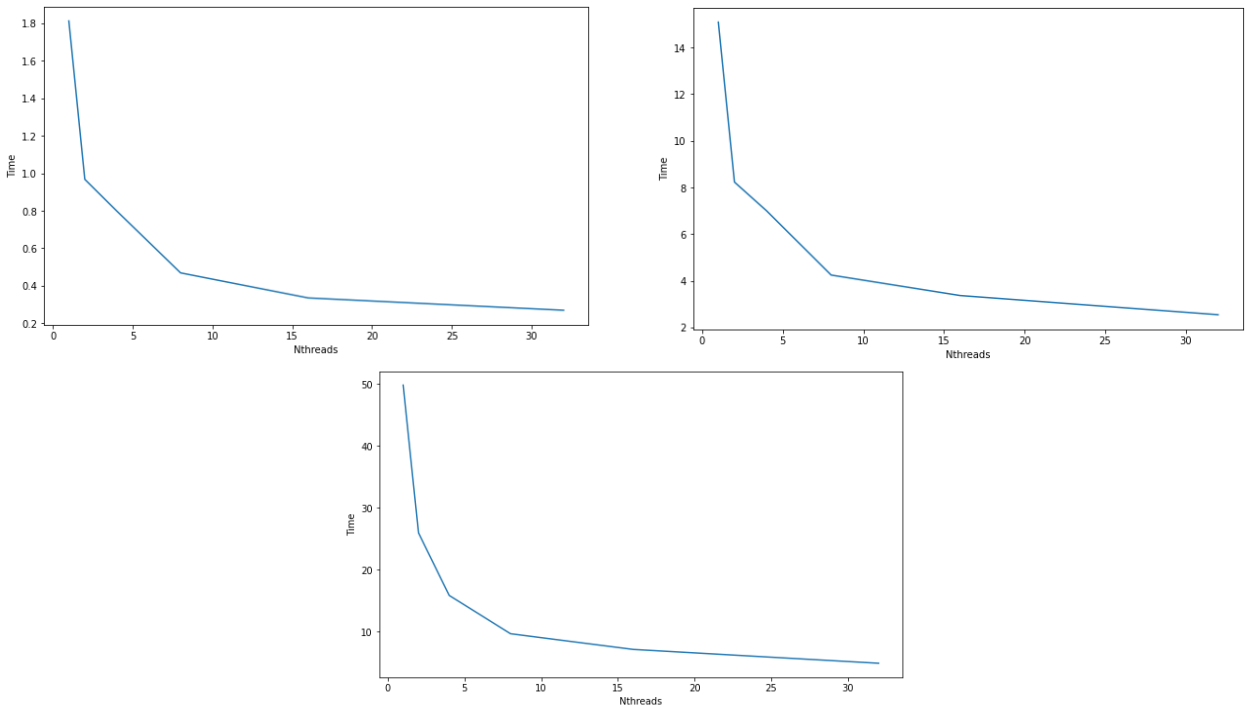

Рисунок 1. На первом графике время для матрицы размерности 100x100x100, на втором – 200x200x200, на третьем – 250x250x250

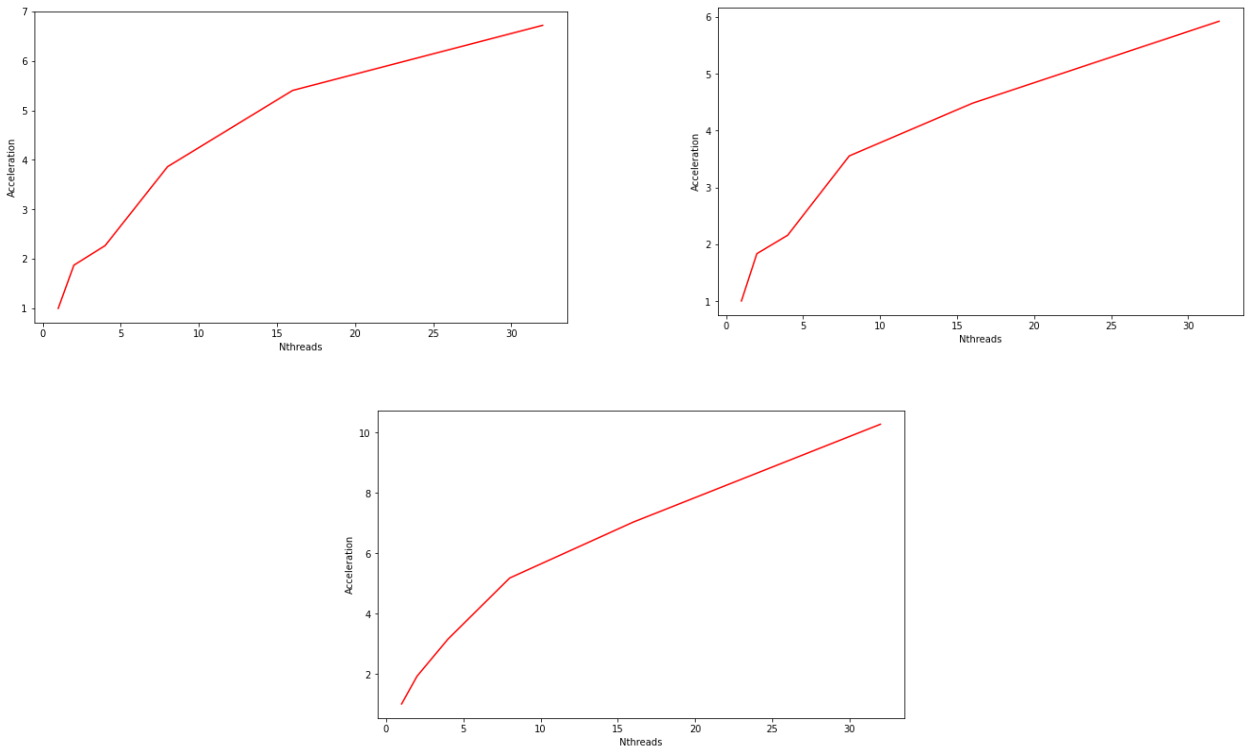

Рисунок 2. Зависимость ускорения от количества потоков. На первом графике для матрицы размерности 100x100x100, на втором – 200x200x200, на третьем – 250x250x250

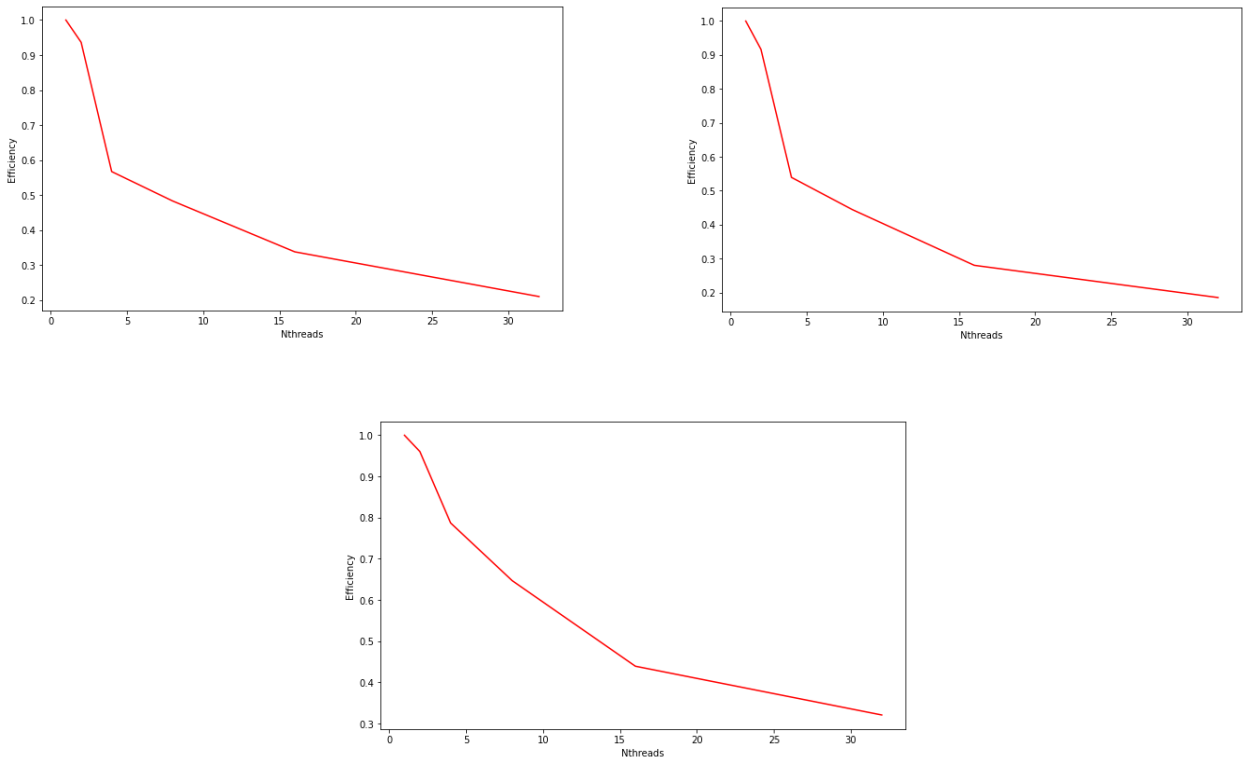

Рисунок 3. Зависимость эффективности от количества потоков. На первом графике для матрицы размерности 100x100x100, на втором – 200x200x200, на третьем – 250x250x2503

Полная таблица результатов:

MatrixSize	*Nthreads*	*Time*	*Residual*	*Error*	*Acceleration*	*Efficiency*
1000000	1	1.811702	3.162160e-12	8.182540e-13	1.000000	1.000000
1000000	2	0.967487	3.160670e-12	8.182860e-13	1.872586	0.936293
1000000	4	0.798827	3.160730e-12	8.182650e-13	2.267953	0.566988
1000000	8	0.468965	3.162710e-12	8.185870e-13	3.863198	0.482900
1000000	16	0.335326	3.161430e-12	8.182680e-13	5.402810	0.337676
1000000	32	0.269666	3.161440e-12	8.185060e-13	6.718308	0.209947
8000000	1	15.084775	3.583650e-09	5.125880e-09	1.000000	1.000000
8000000	2	8.231357	3.583650e-09	5.125880e-09	1.832599	0.916299
8000000	4	6.994245	3.583650e-09	5.125880e-09	2.156741	0.539185
8000000	8	4.245385	3.583650e-09	5.125880e-09	3.553217	0.444152
8000000	16	3.366417	3.583650e-09	5.125880e-09	4.480958	0.280060
8000000	32	2.546768	3.583650e-09	5.125880e-09	5.923106	0.185097
15625000	1	49.844700	2.133200e-09	3.037050e-09	1.000000	1.000000
15625000	2	25.946700	2.133200e-09	3.037050e-09	1.921042	0.960521
15625000	4	15.833167	2.133200e-09	3.037050e-09	3.148119	0.787030
15625000	8	9.629632	2.133200e-09	3.037050e-09	5.176179	0.647022
15625000	16	7.090103	2.133200e-09	3.037050e-09	7.030180	0.439386
15625000	32	4.851773	2.133200e-09	3.037050e-09	10.273503	0.321047