



Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра вычислительных методов

Построение разреженной матрицы и решение СЛАУ

Параллельные высокопроизводительные вычисления

ВЫПОЛНИЛ:
Петров Т. П.
группа 504

5 ноября
Москва, 2024

Содержание

1	Описание задания и программной реализации	3
1.1	Краткое описание задания	3
1.2	Краткое описание программной реализации	3
1.2.1	Запуск программы	3
2	Исследование производительности	4
2.1	Характеристики вычислительной системы	4
2.2	Результаты измерений производительности	4
2.2.1	Последовательная производительность	4
2.2.2	Параллельное ускорение	6
3	Анализ полученных результатов	8

1 Описание задания и программной реализации

1.1 Краткое описание задания

Необходимо реализовать многопоточную программу для решения систем линейных алгебраических уравнений (СЛАУ) на неструктурированной сетке с использованием OpenMP. Алгоритм должен состоять из нескольких этапов:

1. Генерация графа сетки и его матричного представления – создание графа, связей элементов и его представление в разреженном формате CSR
2. Заполнение матрицы СЛАУ – построение матрицы коэффициентов и вектора правой части с использованием тестовых формул
3. Решение СЛАУ – реализация итерационного метода сопряженных градиентов для решения уравнения с поддержкой параллелизма
4. Проверка производительности – измерение времени выполнения каждого этапа и анализ многопоточного ускорения и эффективности алгоритма

1.2 Краткое описание программной реализации

...

1.2.1 Запуск программы

Для запуска на локальных системах достаточно указать количество нитей, а также все необходимые входные данные:

```
OMP_NUM_THREADS=k ./CGSolver Nx Ny K1 K2
```

Для запуска на кластере, использующем систему очередей, запускается скрипт со следующими параметрами:

```
mpisubmit.pl -t k CGSolver -- Nx Ny K1 K2
```

Однако желательно редактирование командного файла для запуска на заданных узлах и привязки к определенному сокету узла.

```
source /polusfs/setenv/setup.SMPI
#BSUB -W 00:15
#BSUB -o CGSolver.%J.out
#BSUB -e CGSolver.%J.err
#BSUB -m polus-c4-ib
#BSUB -R affinity[core(10):distribute=pack(socket=1)]
OMP_NUM_THREADS=k
/polusfs/lsf/openmp/launchOpenMP.py CGSolver Nx Ny K1 K2
```

2 Исследование производительности

2.1 Характеристики вычислительной системы

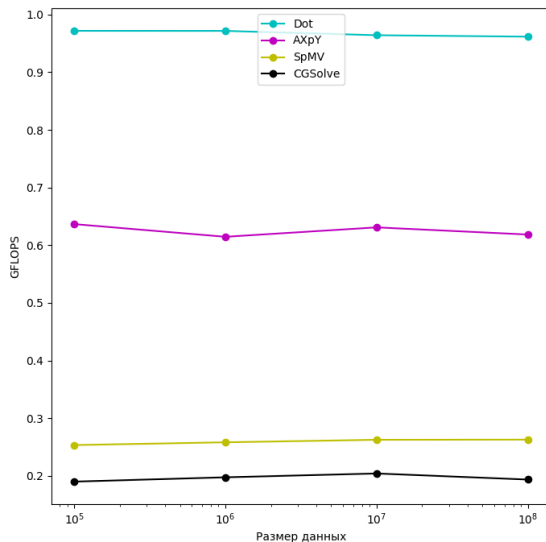
	PC	Polus
CPU	i5-12400F	IBM POWER 8
Cores	6	20
Threads	2	8
TPP	384 GFLOPS	290 GFLOPS
RAM	2xDDR5-5600	4xDDR4-2400
BW	89.6 GB/s	307.2 GB/s

Для того, чтобы собрать программу и скомпилировать все файлы, необходимо выполнить ряд следующих действий:

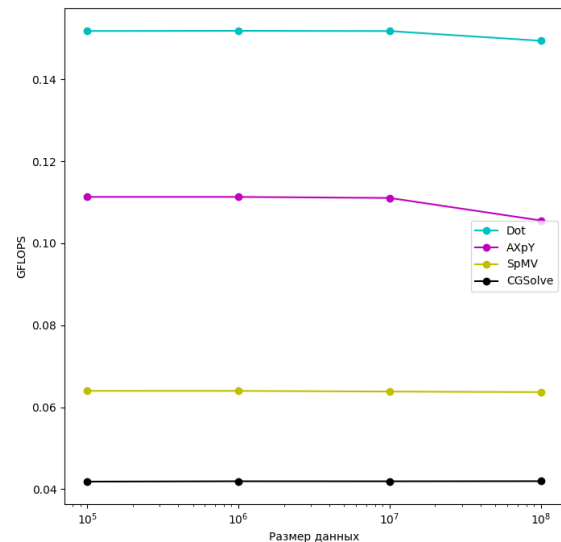
```
mkdir build && cd build
cmake -DENABLE_TESTS=<On|Off> -DUSE_DEBUG_MODE=<On|Off> ..
make -j 4
cd ../bin
```

2.2 Результаты измерений производительности

2.2.1 Последовательная производительность

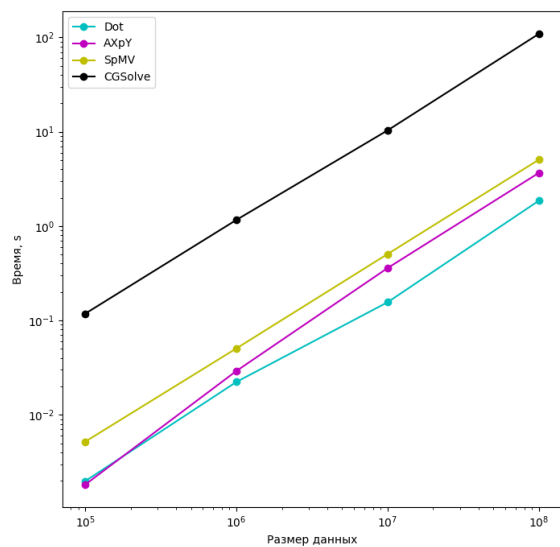


(a) PC

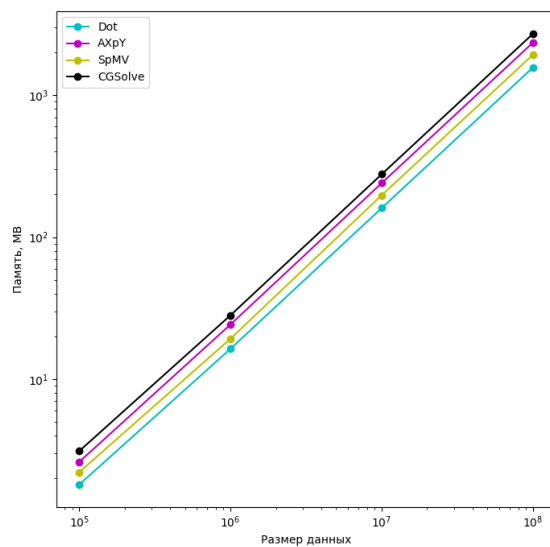


(б) Polus

Рис. 1: Зависимость производительности от размера входных данных



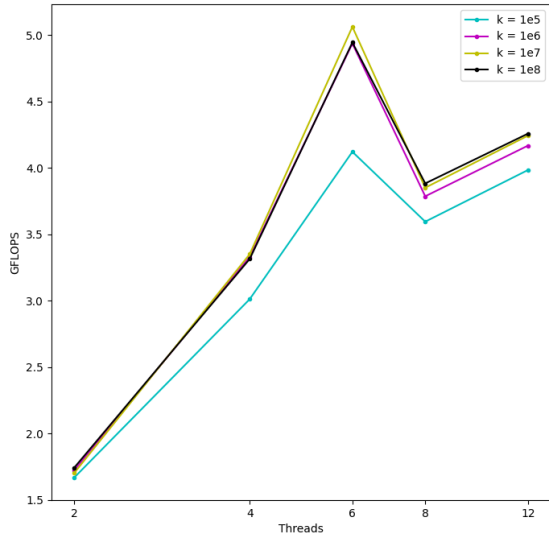
(а)



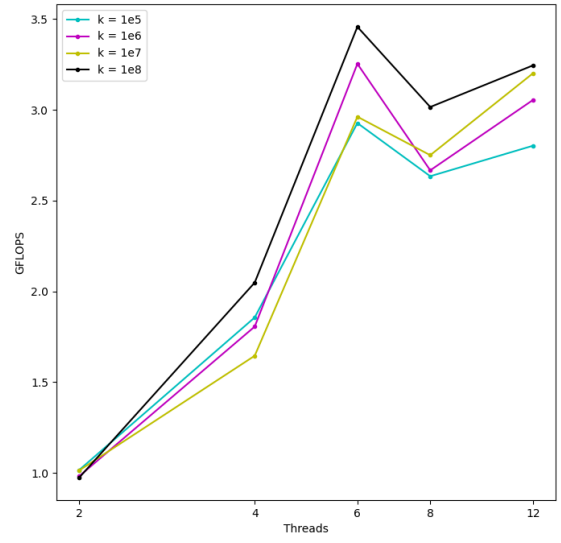
(б)

Рис. 2: Зависимость времени выполнения (а) и выделяемой памяти (б) в зависимости от размера входных данных

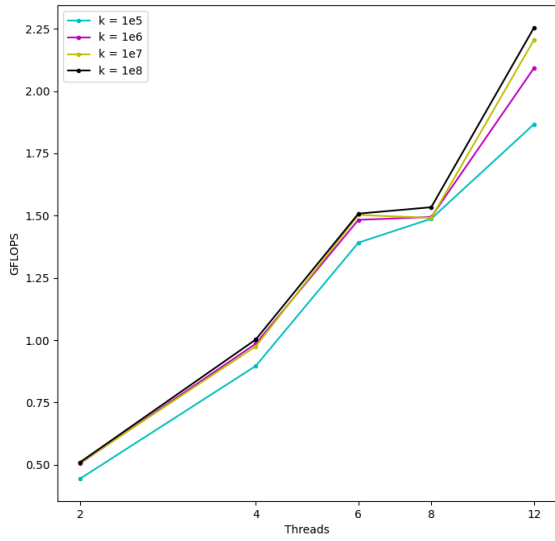
2.2.2 Параллельное ускорение



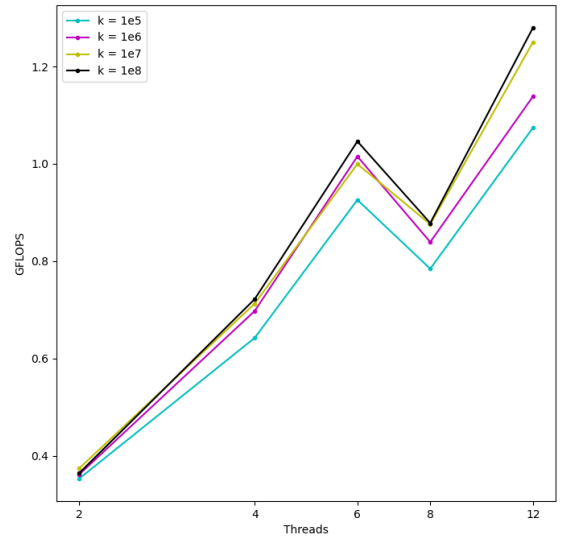
(a) Dot



(б) AXpY

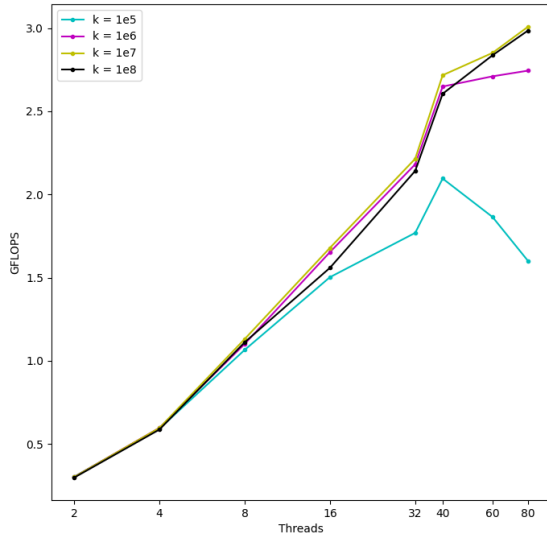


(в) SpMV

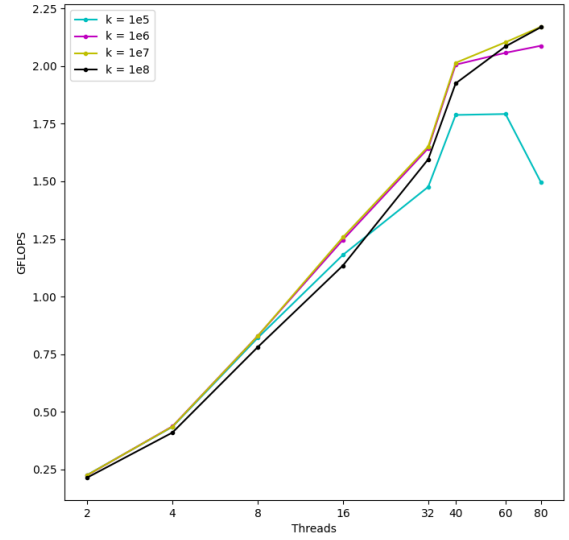


(г) CGSolver

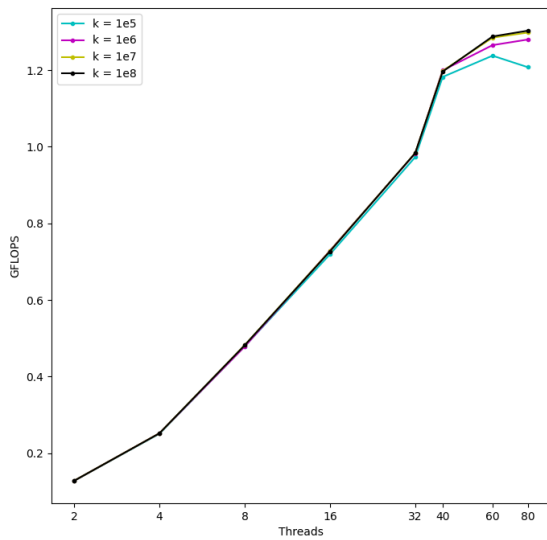
Рис. 3: Зависимость производительности от числа нитей (PC)



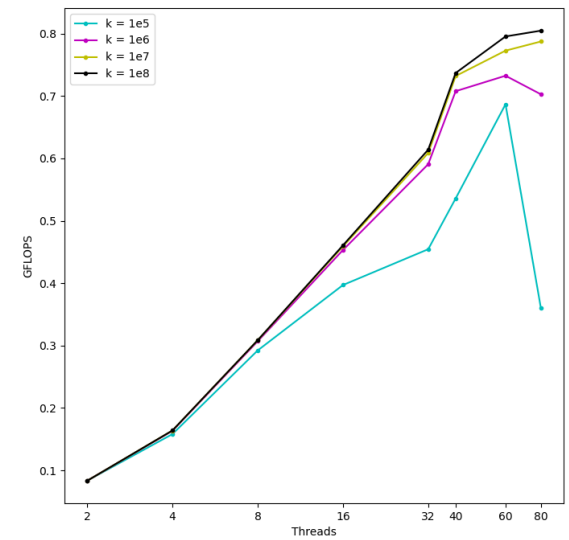
(a) Dot



(б) AXpY



(в) SpMV



(г) CGSolver

Рис. 4: Зависимость производительности от числа нитей (Polus)

T	2	4	8	16	32	40	64	80
Dot	1.99	3.92	7.44	10.43	14.34	17.44	18.99	19.98
AXpY	2.04	3.89	7.40	10.75	15.12	18.24	19.76	20.55
SpMV	2.01	3.96	7.57	11.42	15.45	18.79	20.23	20.46
CGSolve	1.99	3.91	7.36	10.98	14.64	17.56	18.96	19.18

Рис. 5: Расчеты ускорения для каждой из операций при $N = 1e8$ (Polus)

3 Анализ полученных результатов

$$TPP_{PC} = 4 \text{ GHz} \cdot 6 \text{ Cores} \cdot 2 \text{ Threads/Core} \cdot 512/64 = 384 \text{ GFLOPS}$$

$$TPP_{Polus} = 290 \text{ GFLOPS}$$

$$BW_{PC} = 2 \text{ Channels} \cdot 5600 \text{ MT/s} \cdot 8 \text{ bytes} = 89,6 \text{ GB/s}$$

$$BW_{Polus} = 8 \text{ Channels} \cdot 2400 \text{ MT/s} \cdot 8 \text{ bytes} = 153,6 \text{ GB/s}$$

$$AI_{dot} = 2 \text{ FLOP}/(3 \cdot 8 \text{ bytes}) = 1/12$$

$$AI_{AXpY} = 2 \text{ FLOP}/(3 \cdot 8 \text{ bytes}) = 1/12$$

$$AI_{SpMV} = 12 \text{ FLOP}/(13 \cdot 8 \text{ bytes}) = 3/26$$

$$TBP = \min(TPP, BW \cdot AI)$$

$$TBP_{PC,dot} = \min(384 \text{ GFLOPS}, 89,6 \text{ GB/s} \cdot 1/12) = 7.5 \text{ GFLOPS}$$

$$TBP_{PC,AXpY} = \min(384 \text{ GFLOPS}, 89,6 \text{ GB/s} \cdot 1/12) = 7.5 \text{ GFLOPS}$$

$$TBP_{PC,SpMV} = \min(384 \text{ GFLOPS}, 89,6 \text{ GB/s} \cdot 3/26) = 10.3 \text{ GFLOPS}$$

$$TBP_{Polus,dot} = \min(290 \text{ GFLOPS}, 153,6 \text{ GB/s} \cdot 1/12) = 12.8 \text{ GFLOPS}$$

$$TBP_{Polus,AXpY} = \min(290 \text{ GFLOPS}, 153,6 \text{ GB/s} \cdot 1/12) = 12.8 \text{ GFLOPS}$$

$$TBP_{Polus,SpMV} = \min(290 \text{ GFLOPS}, 153,6 \text{ GB/s} \cdot 3/26) = 17.7 \text{ GFLOPS}$$

...	Dot	AXpY	SpMV
<i>PC TBP_{Analytical}</i>	<i>7.5 GFLOPS</i>	<i>7.5 GFLOPS</i>	<i>10.3 GFLOPS</i>
<i>PC RealP</i>	<i>5.06 GFLOPS</i>	<i>3.45 GFLOPS</i>	<i>2.25 GFLOPS</i>
<i>Polus TBP_{Analytical}</i>	<i>12.8 GFLOPS</i>	<i>12.8 GFLOPS</i>	<i>17.7 GFLOPS</i>
<i>Polus RealP</i>	<i>3.01 GFLOPS</i>	<i>2.16 GFLOPS</i>	<i>1.31 GFLOPS</i>

Рис. 6: Аналитические и реальные значение производительности для каждой из операций