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

31 октября
Москва, 2024

Содержание

1 Описание задания и программной реализации

1.1 Краткое описание задания

Необходимо реализовать многопоточную программу для решения систем линейных алгебраически 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

...

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>
make -j 4
cd ../bin
```

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

```
OMP_NUM_THREADS=k ./a.out Nx Ny K1 K2
```

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

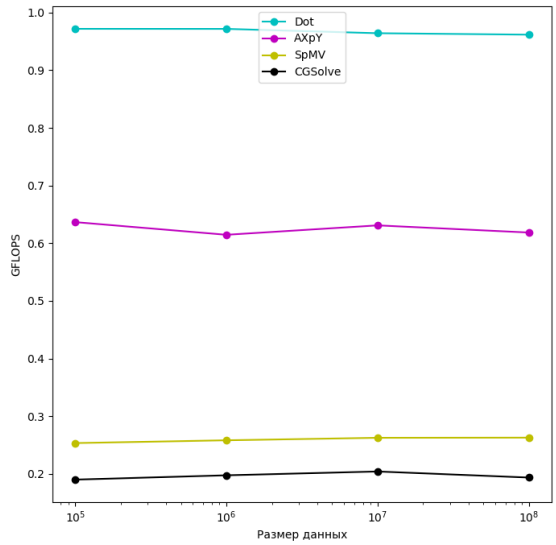
```
mpisubmit.pl params a.out -- a.out params
```

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

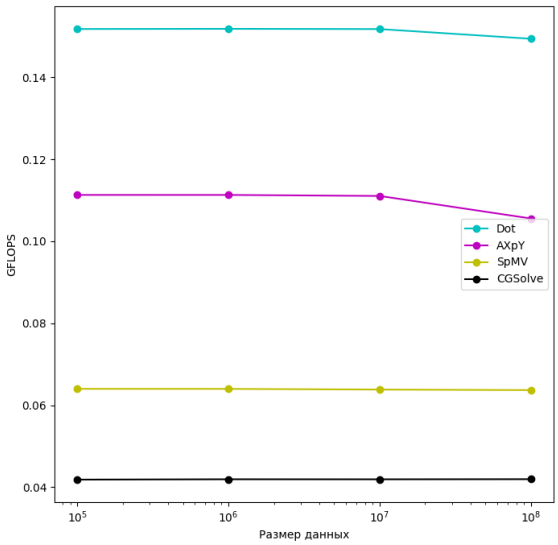
```
source /polusfs/setenv/setup.SMPI
#BSUB -W 00:15
#BSUB -o ../out/DotMeasure.%J.out
#BSUB -e ../out/DotMeasure.%J.err
#BSUB -m polus-c4-ib
#BSUB -R affinity[core(10):distribute=pack(socket=1)]
OMP_NUM_THREADS=1
/polusfs/lsf/openmp/launchOpenMP.py ../DotMeasure
```

2.2 Результаты измерений производительности

2.2.1 Последовательная производительность

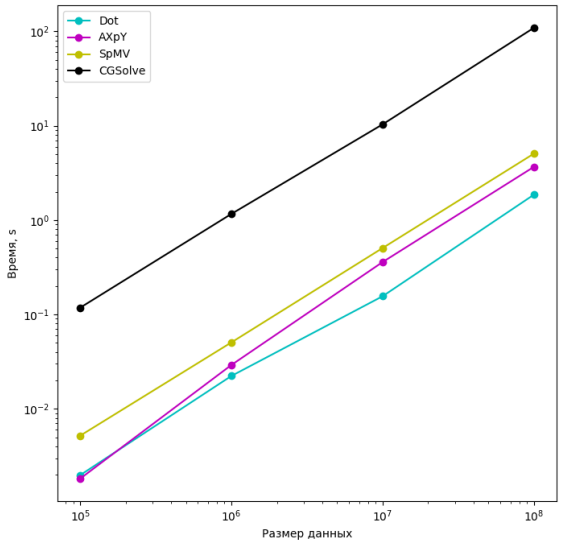


(а) PC

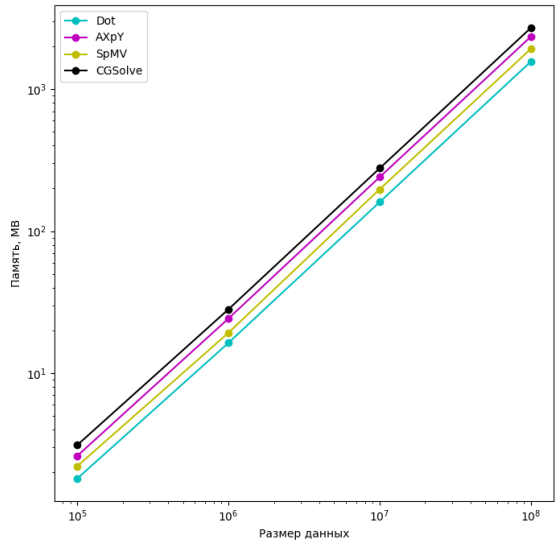


(б) Polus

Рис. 1: Зависимость производительности от размера входных данных



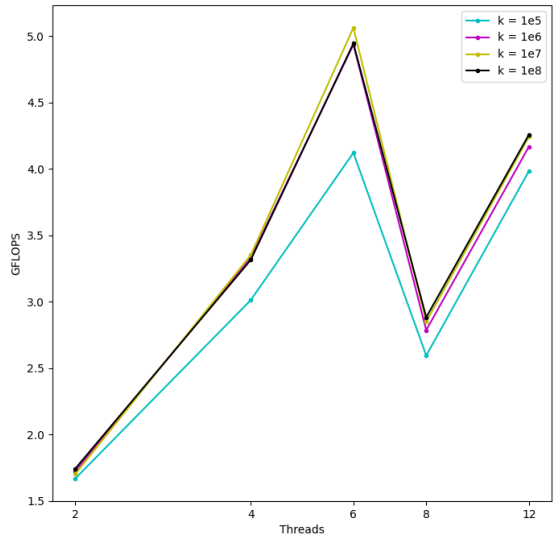
(а)



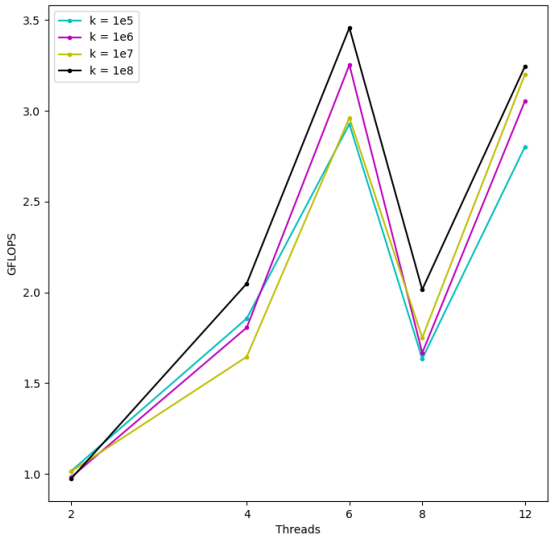
(б)

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

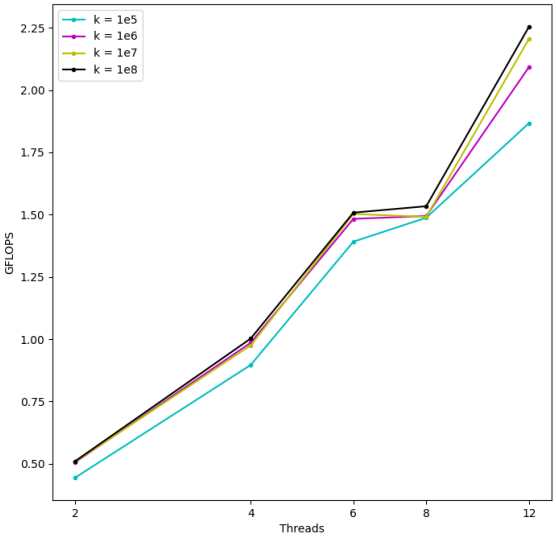
2.2.2 Параллельное ускорение



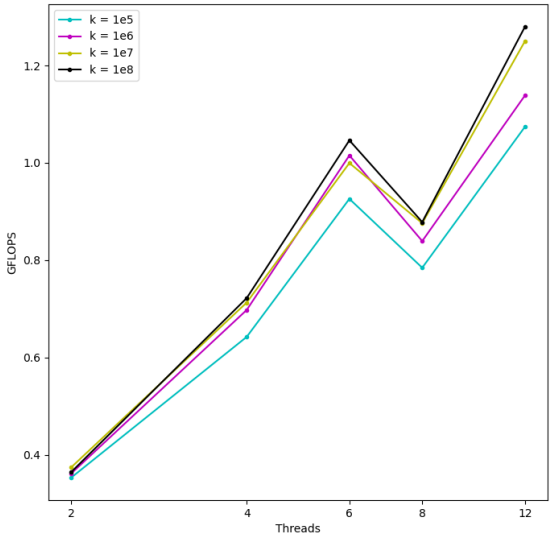
(а) Dot



(б) AXpY

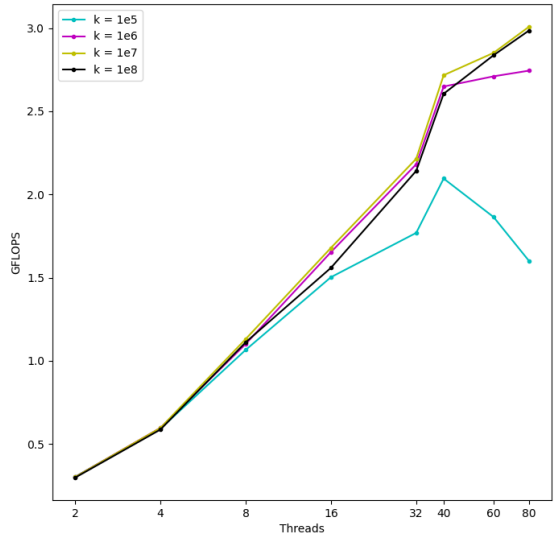


(в) SpMV

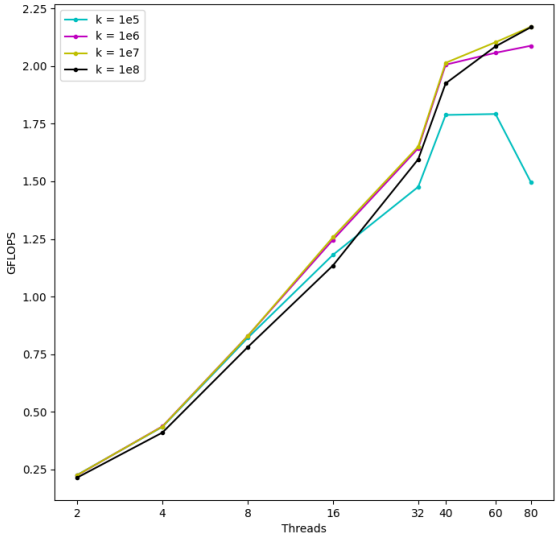


(г) CGSolver

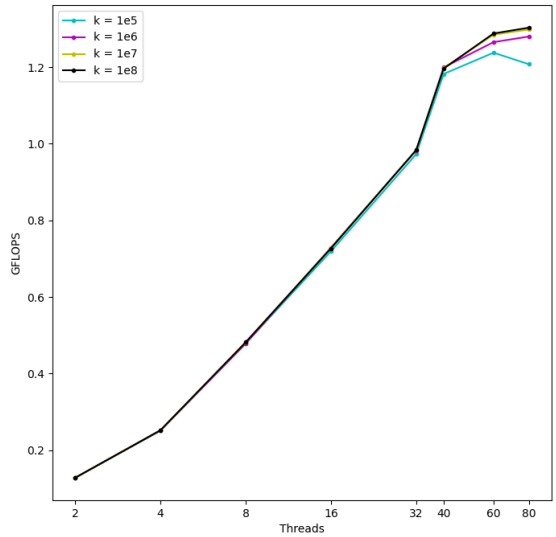
Рис. 3: Зависимость производительности от числа нитей (PC)



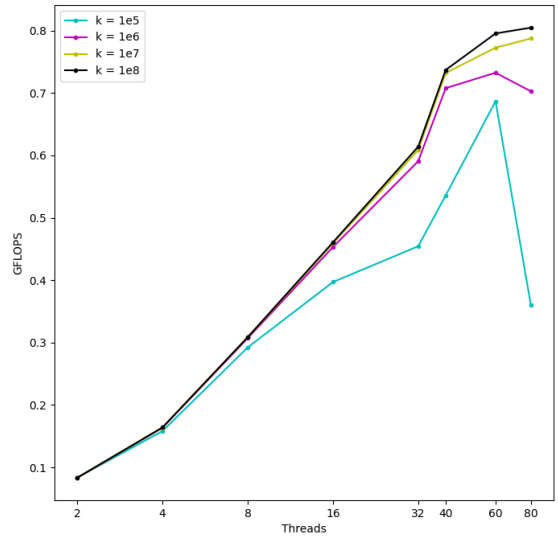
(а) Dot



(б) AXpY



(в) SpMV



(г) CGSolver

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

T	2	4	8	16	32	40	64	80
Dot								
AXpY								
SpMV								
CGSolve								

Рис. 5: Расчеты ускорения для каждой из операций при $N = 10^8$

sdfs

3 Анализ полученных результатов

$TPP_{PC} = 4\ GHz \cdot 6\ Cores \cdot 2\ Threads/Core \cdot 512/64 = 384\ GFLOPS$
$TPP_{Polus} = 290 GFLOPS$
$BW_{PC} = 2\ Channels \cdot 5600\ MT/s \cdot 8\ bytes = 89,6\ GB/s$
$BW_{Polus} = 8\ Channels \cdot 2400\ MT/s \cdot 8\ bytes = 153,6\ GB/s$

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

$TBP = \min(TPP, BW \cdot AI)$
$TBP_{PC,dot} = \min(384\ GFLOPS, 89,6\ GB/s \cdot 1/8) = 5.6 GFLOPS = 0.8\%\ TPP_{PC}$
$TBP_{PC,AXpY} = \min(384\ GFLOPS, 1.28e11/24) = 5.3 GFLOPS = 0.8\%\ TPP_{PC}$
$TBP_{PC,SpMV} = \min(6.4e11, 1.28e11/24) = 5.3 GFLOPS = 0.8\%\ TPP_{PC}$
$TBP_{Polus,dot} = \min(6.4e11, 1.28e11/24) = 5.3 GFLOPS = 0.8\%\ TPP_{Polus}$
$TBP_{Polus,AXpY} = \min(6.4e11, 1.28e11/24) = 5.3 GFLOPS = 0.8\%\ TPP_{Polus}$
$TBP_{Polus,SpMV} = \min(6.4e11, 1.28e11/24) = 5.3 GFLOPS = 0.8\%\ TPP_{Polus}$

...	Dot	AXpY	SpMV
PC			
polus			

Рис. 6: