Задание 1
Оптимизированное распараллеливание программы, реализующей метод Якоби в 3D пространстве
Отчёт

Фролова О.В

1 Постановка задачи

Требуется реализовать максимально оптимизированное распараллеливание программы, которая реализует 3D метод Якоби.

- Менять программу можно (значения выходных массивов должны совпадать)
- Рекомендуется параметризовать тип данных с помощью typedef, чтобы можно было запустить программу как с использованием double, так и с использованием float типов для сравнения производительности.
- В выводе программы также должна сдержаться информация о модели ГПУ и о количестве памяти, которое на нем доступно.

2 Makefile

```
NVCC       := nvcc

NVCCFLAGS  := -O3 -arch=sm_60 -std=c++11 -Xcompiler -fopenmp

TARGET     := jacob
SRC        := jacob.cu

.PHONY: all float double clean

all: $(TARGET)

$(TARGET): $(SRC)
 $(NVCC) $(NVCCFLAGS) $< -o $@

float: NVCCFLAGS += -DUSE_FLOAT
float: clean all
double: clean all

clean:
 rm -f $(TARGET)
```

3 Результаты выполнения

3.1 Тип double

Были выбраны значения размеры сетки L = 900, количества итераций i = 20
Запуск производился на Polus
Результат выполнения команды

```
make
./jacob -L 900 -i 20 --compare
```

```
[edu-cmc-sqi24-14@polus-ib ~]$ ./jacob -L 900 -i 20 --compare
Jacobi3D: 900^3, 20 iterations
Data type: double
GPU: Tesla P100-SXM2-16GB, memory 16280 MB
 IT =    1   EPS = 2.6980000E+03
 IT =    2   EPS = 1.3495000E+03
 IT =    3   EPS = 5.2458333E+02
 IT =    4   EPS = 3.3714352E+02
 IT =    5   EPS = 2.7466667E+02
 IT =    6   EPS = 2.3302469E+02
 IT =    7   EPS = 1.9677308E+02
 IT =    8   EPS = 1.6295002E+02
 IT =    9   EPS = 1.3928905E+02
 IT =   10   EPS = 1.2331360E+02
 IT =   11   EPS = 1.1221165E+02
 IT =   12   EPS = 1.0546532E+02
 IT =   13   EPS = 9.8133253E+01
 IT =   14   EPS = 9.1255019E+01
 IT =   15   EPS = 8.4403267E+01
 IT =   16   EPS = 7.8237112E+01
 IT =   17   EPS = 7.2296906E+01
 IT =   18   EPS = 6.7634362E+01
 IT =   19   EPS = 6.3714148E+01
 IT =   20   EPS = 6.0632024E+01
 CPU time = 9.694s
 GPU IT =    1   EPS = 2.6980000E+03
 GPU IT =    2   EPS = 1.3495000E+03
 GPU IT =    3   EPS = 5.2458333E+02
 GPU IT =    4   EPS = 3.3714352E+02
 GPU IT =    5   EPS = 2.7466667E+02
 GPU IT =    6   EPS = 2.3302469E+02
 GPU IT =    7   EPS = 1.9677308E+02
 GPU IT =    8   EPS = 1.6295002E+02
 GPU IT =    9   EPS = 1.3928905E+02
 GPU IT =   10   EPS = 1.2331360E+02
 GPU IT =   11   EPS = 1.1221165E+02
 GPU IT =   12   EPS = 1.0546532E+02
 GPU IT =   13   EPS = 9.8133253E+01
 GPU IT =   14   EPS = 9.1255019E+01
 GPU IT =   15   EPS = 8.4403267E+01
 GPU IT =   16   EPS = 7.8237112E+01
 GPU IT =   17   EPS = 7.2296906E+01
 GPU IT =   18   EPS = 6.7634362E+01
 GPU IT =   19   EPS = 6.3714148E+01
 GPU IT =   20   EPS = 6.0632024E+01
 GPU time = 0.914s
 Max diff = 0.000000e+00
 Verification: SUCCESSFUL
 Speedup: 10.61x
```

Результаты показывают значительное ускорение работы на GPU по сравнению с CPU. Максимальная разница (Max diff) между результатами CPU и GPU составила 0.000000e+00, что означает полное совпадение данных. Это подтверждает корректность работы программы (Verification: SUCCESSFUL)

3.2 Тип float

Были выбраны значения размеры сетки L = 900, количества итераций i = 20
Запуск производился на Polus

Результат выполнения команды

```
make float
./jacob -L 900 -i 20 --compare
```

```
[edu-cmc-sqi24-14@polus-ib ~]$ ./jacob -L 900 -i 20 --compare
Jacobi3D: 900^3, 20 iterations
Data type: float
GPU: Tesla P100-SXM2-16GB, memory 16280 MB
 IT =    1   EPS = 2.6980000E+03
 IT =    2   EPS = 1.3495000E+03
 IT =    3   EPS = 5.2458325E+02
 IT =    4   EPS = 3.3714380E+02
 IT =    5   EPS = 2.7466663E+02
 IT =    6   EPS = 2.3302454E+02
 IT =    7   EPS = 1.9677307E+02
 IT =    8   EPS = 1.6294995E+02
 IT =    9   EPS = 1.3928918E+02
 IT =   10   EPS = 1.2331372E+02
 IT =   11   EPS = 1.1221167E+02
 IT =   12   EPS = 1.0546558E+02
 IT =   13   EPS = 9.8132935E+01
 IT =   14   EPS = 9.1255371E+01
 IT =   15   EPS = 8.4402954E+01
 IT =   16   EPS = 7.8237305E+01
 IT =   17   EPS = 7.2296875E+01
 IT =   18   EPS = 6.7634399E+01
 IT =   19   EPS = 6.3714233E+01
 IT =   20   EPS = 6.0632202E+01
 CPU time = 9.037s
 GPU IT =    1   EPS = 2.6980000E+03
 GPU IT =    2   EPS = 1.3495000E+03
 GPU IT =    3   EPS = 5.2458325E+02
 GPU IT =    4   EPS = 3.3714380E+02
 GPU IT =    5   EPS = 2.7466663E+02
 GPU IT =    6   EPS = 2.3302454E+02
 GPU IT =    7   EPS = 1.9677307E+02
 GPU IT =    8   EPS = 1.6294995E+02
 GPU IT =    9   EPS = 1.3928918E+02
 GPU IT =   10   EPS = 1.2331372E+02
 GPU IT =   11   EPS = 1.1221167E+02
 GPU IT =   12   EPS = 1.0546558E+02
 GPU IT =   13   EPS = 9.8132935E+01
 GPU IT =   14   EPS = 9.1255371E+01
 GPU IT =   15   EPS = 8.4402954E+01
 GPU IT =   16   EPS = 7.8237305E+01
 GPU IT =   17   EPS = 7.2296875E+01
 GPU IT =   18   EPS = 6.7634399E+01
 GPU IT =   19   EPS = 6.3714233E+01
 GPU IT =   20   EPS = 6.0632202E+01
 GPU time = 0.764s
 Max diff = 0.000000e+00
 Verification: SUCCESSFUL
 Speedup: 11.82x
```

GPU демонстрирует более значительный прирост скорости, что подтверждает его эффективность для задач с пониженной точностью. Результаты остались корректными

3.3 Вывод

Использование GPU позволило добиться почти 11-кратного ускорения вычислений для типа double и почти 12-кратного ускорения для типа float без потери точности, что демонстрирует эффективность применения графических процессоров для задач подобного типа (например, решения уравнений методом Якоби).