Общая информация

По всем вопросам писать в группу в телеграмме или на почту: morozov@infway.ru.

Требования к лабораторным работам

Все работы должны удовлетворять следующим условиям:

1. Программа должна использовать технологию CUDA (MPI, OpenMP).
2. Языки программирования: C или C++. При проверке используется компилятор nvcc версии 12.0 (g++ версии 13.3.0) на 64-х битной Ubuntu 24.04.1 LTS. Параметры графического процессора:
   Compute capability : 7.5
   Name : NVIDIA GeForce GTX 1650
   Total Global Memory : 4086169600
   Shared memory per block : 49152
   Registers per block : 65536
   Max threads per block : (1024, 1024, 64)
   Max block : (2147483647, 65535, 65535)
   Total constant memory : 65536
   Multiprocessors count : 14
3. Использовать в программах готовые библиотеки алгоритмов (thrust, cublas, cufft и др.), если не указано иного, запрещается.
4. **Формат ввода и вывода строго регламентирован** в каждой лабораторной работе.
5. Программа не должна создавать лишних (временных) файлов во время работы.
6. Везде, если не указано иного, подразумевается, что данные вводятся из **файла стандартного ввода (stdin)**, а выводятся в **файл стандартного вывода (stdout)**.
7. Программа не должна ни при каких условиях завершится аварийно — если в какой-то момент времени программа не может быть выполнена по каким-либо причинам (отсутствие оперативной/графической памяти, проблемы с разбором входных данных и т.п.) она должна напечатать в **файл стандартного вывода ошибок (stderr)** строчку в формате ERROR: и дальше описание возникшей проблемы в свободной форме, после чего завершиться с кодом возврата 0.
8. Программа не должна пытаться навредить проверяющей системе (копировать / пересылать / удалять файлы, вешать систему и тп). Преднамеренные и непреднамеренные попытки будут строго караться.
9. **Количество используемых вычислительных потоков в программе должно быть фиксированным и не зависеть от размера входных данных.**

Задания проверяются автоматически. Компиляция производится командой (если не указано иного):

```
nvcc -Werror cross-execution-space-call -lm
```

Сроки сдачи лабораторных работ

На оценку 5: необходимо получить “ОК” в течении **двух недель** после даты выдачи ЛР.
На оценку 4: необходимо получить “ОК” в течении **четырех недель** после даты выдачи ЛР.
На оценку 3: необходимо получить “ОК” до конца семестра.
На оформление отчета и устную защиту работы дополнительно дается **одна неделя**.

Требования к отчету

Каждый отчет должен быть оформлен согласно выданному шаблону и прислан в формате **pdf** на checkpgp@gmail.com. В конце семестра все отчеты необходимо распечатать и сшить в одну папку.

Проверка лабораторных работ

В первую очередь, чтобы можно было отправлять на проверку программы, нужно создать gpg-ключи, и через старост предоставить преподавателям открытый ключ.

Для проверки лабораторной работы, необходимо отправить файл с исходным кодом программы и его gpg-подпись (файл с расширением .asc) на checkpgp@gmail.com. В теме письма указать “pgp:n”, где n - номер ЛР. **Аналогично для отчетов.** Если программа состоит из нескольких файлов то необходимо написать Makefile и упаковать все в архив (пример далее). Сдавать написанные не вами лабораторные и отчеты **категорически запрещено**. Все присланные работы проходят проверку на антиплагиат.

Сгенерировать ключи:

```
gpg --full-generate-key
```

При создании ключа необходимо указать **фамилию и имя в транслитерации в нижнем регистре,** в виде **surname-name** (пример: ivanov-ivan), а также, ту **почту**, с которой будет выполняться отправка на проверку работ и **которую указывали ранее (!)**. Основные команды:

Посмотреть список своих ключей:	gpg --list-secret-keys --keyid-format=short
Экспортировать открытый ключ:	gpg -a --export "XXXXXXXX" > **surname-name**.asc
Экспортировать закрытый ключ:	gpg -a --export-secret-key "XXXXXXXX" > private-key.asc
Создать подпись файла lab.cu:	gpg -u "XXXXXXXX" -ab lab.cu

где XXXXXXXX - key id созданного ключа.

Перед отправкой файла с открытым ключом **обязательно проверить**, что в нем содержится только **ОДИН** ключ (gpg --import-options show-only --import surname-name.asc) и что кодировка файла **UTF-8** или **ASCII** (uchardet surname-name.asc или file -i surname-name.asc)**.**

Настоятельно рекомендуется дополнительно сохранить где-то свой закрытый ключ на случай поломки компьютера или переустановки операционной системы.

Ответы проверяющей системы

- “Wrong answer at test {n}.t.” — Неправильный ответ на тесте номер n.
- “Runtime error at test {n}.t. Program exited with code {code}.” — Ошибка во время исполнения программы на тесте номер n. Программа завершилась с кодом возврата code.
- “Runtime error at test {n}.t., got signal {signal}.” — Ошибка во время исполнения программы на тесте номер n. Программа получила сигнал signal.
- “Time limit exceeded at test {n}.t.” — Превышено максимальное время работы программы на тесте с номером n (в ЛР по MPI — данная ошибка проявляется как runtime error с killed 9).

- “OK” — Все тесты пройдены успешно.
- “Compilation error.” — Ошибка при компиляции программы.
- “Error: wrong subject.” — Неправильная тема письма.
- “Bad signature.” — Плохая подпись файла.
- “Bad attachments.” — В письме нету содержательных файлов (ни программы, ни отчета).
- “No signature.” — Нет подписи к файлу.
- “Thanks, your code is saved.” — Для работ, тесты для которых не предусмотрены в системе (в частности для 6-ой ЛР).
- “Thanks, your report is saved.” — Отчет сохранен.
- “Error: unregistered user.” — Неизвестный студент.

При перечисленных выше первых четырех ответах, в письме также будут приходить первые и последние 10 строк вывода в stderr (но не более 2000 символов). Студент по своему усмотрению может выводить в stderr любую отладочную информацию.

Доступ к удаленному серверу с CUDA

Если отсутствует видеокарта Nvidia поддерживающая технологию CUDA, то можно использовать Google Colab (см. пример). Или можно получить ssh доступ к серверу с такой видеокартой. Для этого необходимо написать запрос на почту преподавателю и приложить к письму свой открытый ssh-ключ. Пример ключа:
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHJLEGGAVLINre3efx8MsROfc7pUNny/k+KWaCFrbTcu user@laptop

Дополнительные возможности за рамками курса

Для студентов, которые знают sycl и хотят делать работы на нем, на проверяющем сервере дополнительно установлен компилятор icpx (Intel(R) oneAPI DPC++/C++ Compiler 2024.2.0 (2024.2.0.20240602)). По умолчанию у него прописаны флаги -fsycl -fsycl-targets=nvptx64-nvidia-cuda. Для того, чтобы код компилировался с помощью icpx нужно написать соответствующий Makefile. **Важно**, что что при использовании sycl в любом случае необходимо строго соблюдать все требования ЛР.

Оценка автоматом за экзамен

Оценка автоматом за экзамен - это средняя оценка за ЛР округленная к ближайшему целому. В случае наличия попыток сдачи плагиата - максимальная оценка три балла.

Пример упаковки программы в архив и создания её электронной подписи

```
sasha@work:~/mpi/7$ cd lab7/
sasha@work:~/mpi/7/lab7$ ls
makefile  mpi7.cpp
sasha@work:~/mpi/7/lab7$ cat makefile
CC = /usr/local/bin/mpic++
CFLAGS = --std=c++11 -fopenmp -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long -lm
SOURSES = mpi7.cpp
BIN = lab7
all:
    $(CC) $(CFLAGS) -o $(BIN) $(SOURSES)
sasha@work:~/mpi/7/lab7$ make
/usr/local/bin/mpic++ --std=c++11 -fopenmp -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long -lm -o lab7 mpi7.cpp
sasha@work:~/mpi/7/lab7$ ls
lab7  makefile  mpi7.cpp
sasha@work:~/mpi/7/lab7$ rm lab7
sasha@work:~/mpi/7/lab7$ cd ..
sasha@work:~/mpi/7$ tar -cvf lab7.tar lab7
lab7/
lab7/mpi7.cpp
lab7/makefile
sasha@work:~/mpi/7$ ls
lab7  lab7.tar
sasha@work:~/mpi/7$ gpg -ab lab7.tar

Необходима фраза-пароль для доступа к закрытому ключу пользователя: "alexander-morozov <alex-icex@mail.ru>"
2048-битный ключ RSA, ID 7E3A1A1E, создан 2016-06-08
sasha@work:~/mpi/7$ ls
lab7  lab7.tar  lab7.tar.asc
sasha@work:~/mpi/7$
```

Важно чтобы имя исполняемого файла совпадало с именем каталога и названием архива.

Вместо утилиты tar можно использовать zip: sasha@work:~/mpi/7$ zip **lab7**.zip **lab7**/*
или утилиту rar: sasha@work:~/mpi/7$ rar a **lab7**.rar **lab7**/*

При написании makefile необходимо указывать следующие пути к компиляторам:
/usr/bin/nvcc
/usr/bin/mpic++