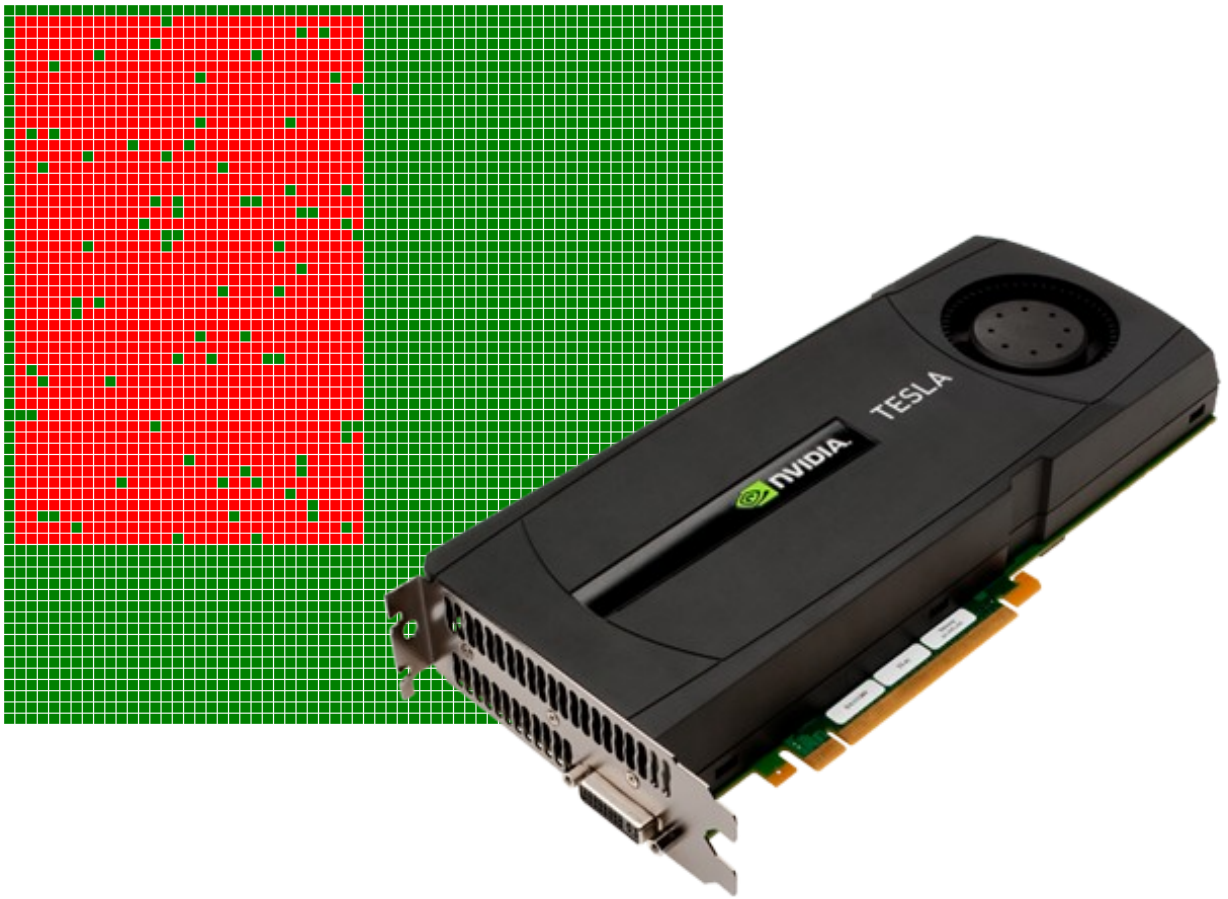
Пример расчёта двумерных данных на CUDA



Элементы программы, использующей GPU

- Подготовка входных данных
- Загрузка входных массивов на GPU
- Запуск CUDA-ядер
- Выгрузка результатов из GPU

Пример pattern2d

- Генерация двумерного поля (nx,ny)
- Манипуляции с загрузкой/выгрузкой на гпу/хост
- Запуск CUDA-ядра, выполняющего локальное преобразование типа разностной схемы
- Расчёт остаточной области на CPU
- Расчёт контрольного результата на CPU
- Сравнение результатов и визуализация отличий

Целевая система

Tesla A/B (тестовые узлы для практикума)

Core 2 Duo / Quad, 4 GB DDR3

Fedora 13 x86_64

cuda toolkit 3.2 (компилятор для gpu, runtime)

gcc 4.4.4 (хост-компилятор)

ImageMagick (визуализация)

Сборка pattern2d

```
[dmikushin@tesla-b pattern2d]$ ls
draw_diffs.c  makefile  pattern2d.cu

[dmikushin@tesla-b pattern2d]$ make
nvcc -g -G0 -arch=sm_13 -c pattern2d.cu
gcc -g -std=c99 -c -I/usr/include/ImageMagick draw_diffs.c
nvcc -g -G0 -arch=sm_13 pattern2d.o draw_diffs.o -o pattern2d -lMagickCore

[dmikushin@tesla-b pattern2d]$ ./pattern2d
Sample 2D field processing
Usage: ./pattern2d <nx> <ny>

[dmikushin@tesla-b pattern2d]$ ./pattern2d 64 64
Done! abs max diff = 0.000000 @ (20,32)

[dmikushin@tesla-b pattern2d]$
```

Как работает сборка

GNU Make - http://www.gnu.org/software/make/

```
# makefile состоит из правил сборки вида
<цель>: <цели, от которых зависит данная цель>
<табуляция><команды shell>
…
<табуляция><команды shell>

# all – это правило, с которого начинается любая сборка
all: pattern2d.o draw_diffs.o
	nvcc -g -G0 -arch=sm_13 pattern2d.o draw_diffs.o -o pattern2d -lMagickCore

# объектный файл программы перестраивается автоматически
# в случае обновления зависимости – исходного файла pattern2d.cu
pattern2d.o: pattern2d.cu
	nvcc -g -G0 -arch=sm_13 -c pattern2d.cu
```

Как работает сборка

nvcc -g -G0 -arch=sm_13 pattern2d.o draw_diffs.o -o pattern2d -lMagickCore

nvcc – обращение к компилятору CUDA

-g – включать отладочную информацию в хост-код

-G0 – включать отладочную информацию в девайс-код и не оптимизировать

-arch=sm_13 – компиляция для архитектуры Tesla 10 (и новее)

***.o** – включаемые объектные файлы

-o <output> – выходной исполняемый файл

-l<name> – линкуемые библиотеки с заменой lib в имени на -l, например для линковки libMagickCore.so нужно указать -lMagickCore или полный путь

Как работает сборка

```
# ещё один объектный файл
draw_diffs.o: draw_diffs.c
	gcc -g -std=c99 -c -I/usr/include/ImageMagick draw_diffs.c

# очищение сборки (обычно - удаление всех файлов, кроме исходников)
clean:
	rm -rf *.o pattern2d
```

Результаты

правильный результат (девайс-версия совпадает с контрольной)

[dmikushin@tesla-b pattern2d]$ ./pattern2d 64 64

Done! abs max diff = 0.000000 @ (20,32)

неправильный результат (строится визуальное представление различий)

[dmikushin@tesla-b pattern2d]$ ./pattern2d 64 64

Wrote diffs to pattern2d.png

Done! abs max diff = 0.100000 @ (17,12)

содержимое pattern2d.png (зелёным и красным помечены узлы, в которых погрешность, соответственно, находится в заданных пределах или превышает допустимый уровень