МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Юго-Западный государственный университет

(ЮЗГУ)

Кафедра вычислительной техники

ИЗМЕРЕНИЕ ПРОПУСКНОЙ СПОСОБНОСТИ ПАМЯТИ ВИДЕОКАРТ С ПОДДЕРЖКОЙ ТЕХНОЛОГИИ CUDA

Методические указания по выполнению лабораторных и практических работ для студентов направлений подготовки 09.03.01 и 09.04.01

Курск 2016

УДК 621.3
Составитель: Э.И. Ватутин

Рецензент
Кандидат технических наук, доцент *А.Г. Спеваков*

Измерение пропускной способности памяти видеокарт с поддержкой технологии CUDA: методические указания по выполнению лабораторных и практических работ по дисциплине «Параллельное программирование» / Юго-Зап. гос. ун-т; сост.: Э.И. Ватутин; Курск, 2016. 8 с.: ил. 1.

Методические рекомендации содержат сведения по разработке программных средств для обмена данными между процессором, оперативной памятью и памятью видеокарт с использованием инструментария NVidia CUDA на современных языках программирования высокого уровня.

Предназначены для студентов направлений подготовки 09.03.01 и 09.04.01 «Информатика и вычислительная техника».

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60x84 1/16.
Усл. печ. л. Уч. – изд.л. Тираж 30 экз. Заказ . Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул. 50 лет Октября, 94.

Содержание

Основные теоретические положения

Цель работы: научиться измерять пропускную способность памяти GPU при различных типах и направлениях передачи данных.

Работа с памятью является одним из краеугольных камней при разработке эффективных программ с использованием технологии CUDA. Ее правильное использование зачастую позволяет увеличить скорость обработки данных до 10 и более раз.

Видеокарта не имеет доступа к оперативной памяти компьютера (host), поэтому все обрабатываемые данные должны быть загружены в динамическую память GPU (device). В данной работе используется глобальная память, для обмена данными с ней используется функция

```
cudaError_t cudaMemcpy(void *dst, void *src, size_t count,
    cudaMemcpyKind kind);
```

которая осуществляет копирование `count` байт из источника `src` в приемник `dst`. Направление копирования задается параметром `kind` и может принимать следующие значения:

```
cudaMemcpyHostToHost     /* Host   -> Host */
cudaMemcpyHostToDevice   /* Host   -> Device */
cudaMemcpyDeviceToHost   /* Device -> Host */
cudaMemcpyDeviceToDevice /* Device -> Device */
```

В первом случае копирование производится в пределах оперативной памяти (функция работает как `memcpy()`), во втором – из оперативной памяти в глобальную память GPU, в третьем – из глобальной памяти GPU в оперативную память, и в четвертом – в пределах глобальной памяти GPU.

Для выделения области памяти в глобальной памяти GPU используется функция

```
cudaError_t cudaMalloc(void **devPtr, size_t size);
```

возвращающая в первом параметре указатель на выделенную область памяти размером `size` байт. Для выделения области динамической памяти в оперативной памяти можно использовать функции

```
void * malloc(size_t size);
```

и

```
cudaError_t cudaMallocHost(void **ptr, size_t size);
```

Функция `malloc()` входит в стандартную библиотеку, она выделяет блок динамической памяти размера `size` и возвращает указатель на его начало. Функция `cudaMallocHost()` отличается от нее тем, что помечает выделяемые страницы виртуальной памяти так, что они гарантированно находятся в оперативной памяти и не могут быть вытеснены из нее в файл подкачки (т.н. page locking). Это упрощает процедуру наблюдения за данной памятью со стороны драйвера видеокарты и увеличивает скорость обмена. Однако пользоваться данной функцией необходимо с осторожностью, т.к. при выделении большого объема данных производительность системы в целом может существенно ухудшиться, могут появиться системные ошибки и т.п.

Копирование в рамках глобальной памяти GPU осуществляется асинхронно: функция `cudaMemcpy()` возвращает управление раньше, чем завершается копирование. Чтобы учесть это, можно добавить в код вызов функции

```
cudaError_t cudaThreadSynchronize();
```

которая ожидает завершения текущей асинхронной операции и лишь затем возвращает управление. Фактически таким образом происходит синхронизация потока CPU с GPU.

Задание

1. Выделить блоки памяти одинакового размера в оперативной памяти и глобальной памяти видеокарты. Скопировать содержимое блоков между:
    - двумя буферами в оперативной памяти;
    - между оперативной памятью и глобальной памятью видеокарты в направлении CPU→GPU и GPU→CPU с использованием обычной и page-locked памяти (должно быть 4 различных варианта копирования);
    - между двумя буферами в глобальной памяти видеокарты.
2. Убедиться в том, что копирование происходит корректно.
3. Измерить время копирования и, зная размер копируемого блока, определить пропускную способность (в ГБ/с) для каждого типа копирования. Сделать выводы о скорости копирования в различных режимах.
4. В отчет включить краткое описание типа и параметров видеокарты и процессора.

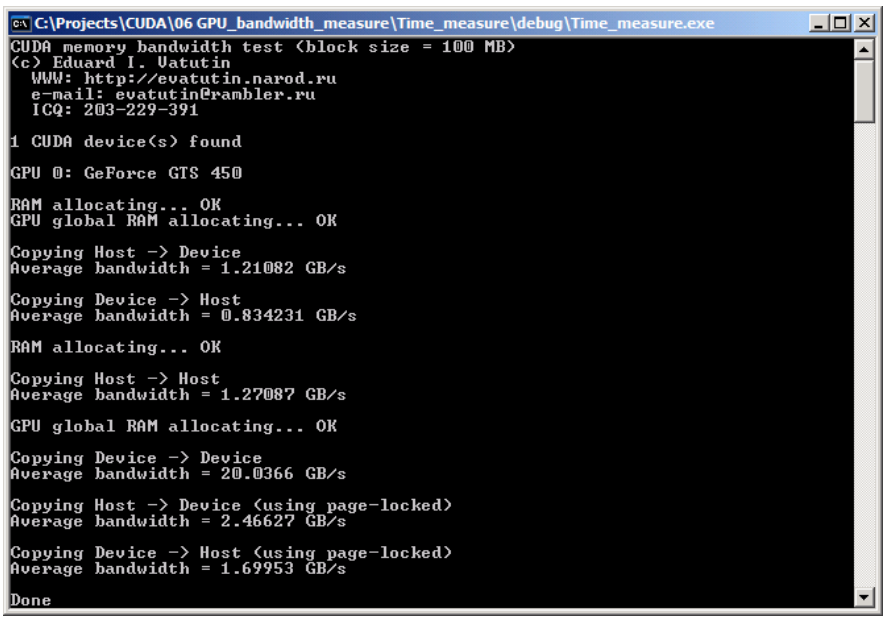

Рис. Результаты измерения пропускной способности

Содержание отчета

1. Титульный лист
2. Цель работы
3. Задание
4. Листинг программы
5. Результаты измерения пропускной способности
6. Выводы

Контрольные вопросы

1. Для чего предназначен инструментарий CUDA?
2. Как производится программирование NVidia GPU с использованием CUDA?
3. Как производится обмен данными между процессором, оперативной памятью и видеокартой?

4. Какая подсистема в составе компьютера лимитирует скорость обмена данными между оперативной памятью и глобальной памятью видеокарты?

Библиографический список

1. Емельянов С.Г., Ватутин Э.И., Панищев В.С., Титов В.С. Процедурно-модульное программирование на Delphi: учебное пособие. М.: Аргамак-Медиа, 2014. 352 с.
2. Зотов И.В., Ватутин Э.И., Борзов Д.Б. Процедурно-ориентированное программирование на C++: учебное пособие. Курск: КурскГТУ, 2008. 211 с.