Физическое моделирование на OpenCL

Боголепов Д. bogdencmc@inbox.ru

Захаров М. maxim.zakharov@inbox.ru

Сопин Д. sopindm@gmail.com

Удалова Т. udalova.t@gmail.com

Блохин О. blohin.o.d@gmail.com

Калишев Г. gleb-kalishev@rambler.ru

# Введение

В конце 2008 года организация Khronos Group утвердила новый стандарт OpenCL. Он предназначен для осуществления вычислений на гетерогенных системах. В апреле 2009 года Nvidia и AMD, выпустили драйвера с поддержкой OpenCL, но до сих пор мало информации о производительности решений на базе этой технологии.

# Постановка задачи

Для тестирования производительности была выбрана задача моделирования из класса N-body problems. Рассматривается эволюция системы N взаимодействующих точечных зарядов в магнитном поле. Действие зарядов друг на друга определяется по закону Кулона[3 p12], а влияние магнитного поля на частицу – силой Лоренца[3 p123].

# Описание решения

Система дифференциальных уравнений, определяющая изменение ускорения частиц интегрируется при помощи методов Эйлера и Рунге-Кутты. Основную долю вычислений при решении задачи составляет расчёт ускорения каждой частицы. Поэтому мы оцениваем количество flops, которое занимает основная доля вычислений, а также количество процессорных циклов, требуемых для них. Код для GPU был написан с использованием OpenCL, а для CPU с применением OpenMP. Для компиляции кода для CPU был переменен Intel C++ Compiler 11.

|  |  |  |  |
| --- | --- | --- | --- |
| code | flops | GPU cycles | CPU cycles |
| for ( int i = 0; i < N; ++i ) |  |  |  |
| { |  |  |  |
| // Calculate acceleration caused by particle j on particle i |  |  |  |
| float3 r = p - pi; | 3 | 3 | 1 |
| float dist = dot ( r, r ); | 5 | 5 | 2 |
| float invDist = inversesqrt ( dist + SOFTENING ); | 2 | 5 | 4 |
| float invDistCube = invDist \* invDist \* invDist; | 2 | 2 | 2 |
| float s = q \* qi \* invDistCube; | 2 | 2 | 2 |
| // accumulate effect of all particles |  |  |  |
| acc += s \* r; | 6 | 3 | 2 |
| } |  |  |  |

Мы не учитываем операции с итеративной переменной, поскольку их вклад ничтожно мал. Более того, для CPU цикл for был полностью развернут компилятором. Различие в количестве циклов, затрачиваемых GPU и CPU на выполнение одинаковых операций обусловлено особенностями их архитектуры. Тестирование производилось на GPU, имеющем скалярную архитектуру и на CPU от фирмы Intel, поддерживающем векторные расширения. Благодаря этому, а также компиляторной векторизации, CPU способен выполнять несколько однотипных операций за один цикл. В связи с этим операция dot выполняется на GPU за 5 циклов (3 умножения и 2 сложения), а на CPU – за 2 (1 векторное умножение и одно векторное сложение). Цифры последней строчки таблицы обусловлены тем, что в системе команд GPU есть инструкция одновременного умножения и сложения (mad – multiple add). На CPU же операция просто векторизуется.

Мы получили, что на одну расчет движения одной частицы приходится 20∙N flops, а на одну итерацию по методу Эйлера – 20∙N2 flops. Число операций растёт пропорционально квадрату количества частиц.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Число частиц** | **2K** | **4K** | **8K** | **16K** |
| Число операций, MFLOPS | 83,89 | 335,54 | 1342,18 | 5368,7 |

Для теоретической оценки производительности мы пользовались формулой:

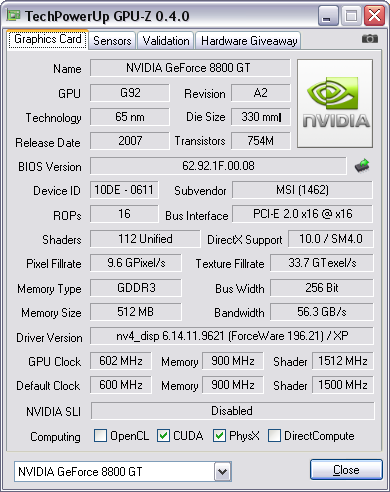
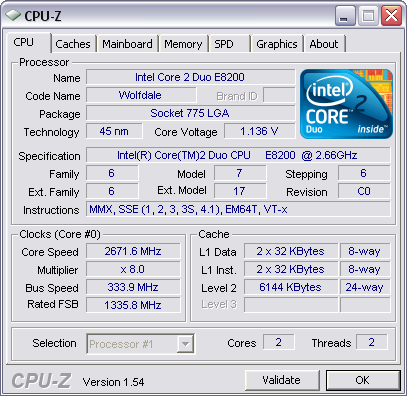
FPS = (*Число ядер\*Частота процессора*) */* (*Число частиц\*Число тактов на частицу*)

Исследование производительности проводилось на двух машинах. Запускалась симуляция системы из 16384 частиц.

Система 1: (тут надо выбрать что именно написать)

????? характеристики компьютера из 110 ауд.

Система 2: (тут надо выбрать, что именно написать)



((((обратил внимание на отсутствие галочки около OpenCL и озадачился))))

Теоретические оценки таковы FPS при использовании метода Эйлера таковы:

При расчете на GPU:

Для системы 1: **21.46** FPS.

Для системы 2: **31.29** FPS (ого!!!) – (((((попробуйте пересчитать. Тут что-то не так. )))))

При расчете на CPU:

Для системы 1: **1.34** FPS.

Для системы 2: **1.53** FPS

Практические результаты приведены в таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| Система | Технология | Численный метод | FPS |
| 1 | OpenCL | Эйлера | **14.9** |
| 1 | OpenCL | Рунге-Кутты | 7.2 |
| 1 | OpenMP | Эйлера | **1.2** |
| 1 | OpenMP | Рунге-Кутты | 0.6 |
| 2 | OpenCL | Эйлера | **34.7(ого!!!)** |
| 2 | OpenCL | Рунге-Кутты | 15.0 |
| 2 | OpenMP | Эйлера | **1.7** |
| 2 | OpenMP | Рунге-Кутты | 0.8 |

(((((нужно выбрать стиль всех таблиц)))))

# Результаты экспериментов

# Заключение

# Литература

1. *The OpenCL Specification*<http://www.khronos.org/registry/cl/specs/opencl-1.0.48.pdf>
2. *Summary of OpenMP 3.0 C/C++ Syntax*  
   <http://www.openmp.org/mp-documents/OpenMP3.0-SummarySpec.pdf>
3. *Учебник по физике*
4. Публикация с задачей N тел