Библиотека динамической адаптации программ к гетерогенным архитектурам вида CPU + GPU

Кривов М.А., m_krivov@ttgLabs.com
Гризан С.А. , s_grizan@ttgLib.org

Проблемы

- Как обеспечить загрузку мультипроцессоров?
  Необходимо учесть особенности GPU, количество регистров и объём разделяемой памяти, требуемой одному блоку
- Как правильно задать grid?
  При смене размера блока производительность может измениться более чем в 2 раза. Пример из жизни: 16x16 «медленнее» 64x4 в 1.69 раз.
- Как совместно использовать CPU и GPU?
  На ряде задач Intel Xeon соответствует NVidia Tesla, а обычные процессоры Intel Core i7 обходят некоторые модели NVidia GeForce

Решение

- Авто-подбор «магических» констант
  Если значение параметра не очевидно, то достаточно заменить его тип на Parameter<>. Оптимальное значение будет подобрано во время работы программы автоматически.
- Балансировщик вычислений между CPU и GPU
  Если есть потребность и в CPU, и в GPU, то достаточно просто передать две функции (CUDA и C++).
- Сохранение оптимизационных сессий
  Если удалось подобрать оптимальные параметры, то они будут использованы при последующем запуске как начальное приближение для дальнейшей оптимизации

Схема работы

Шаг 1. Замена «магических» констант параметрами

```
//...

dim3 block(256);
myKernel<<<grid, block>>>(...)
//...
```

⇒

```
//...
RangedParameter<int> p(32, 1024);
dim3 block(p);
myKernel<<<grid, block>>>(...)
//...
```

Шаг 2. Включение оптимизации

```
//...
while (delta > 1e-3) {
    Simulate();
}
//...
```

⇒

```
//...
while (delta > 1e-3) {
    Optimizer->StartIteration();
    Simulate();
    Optimizer->FinishIteration();
}
//...
```

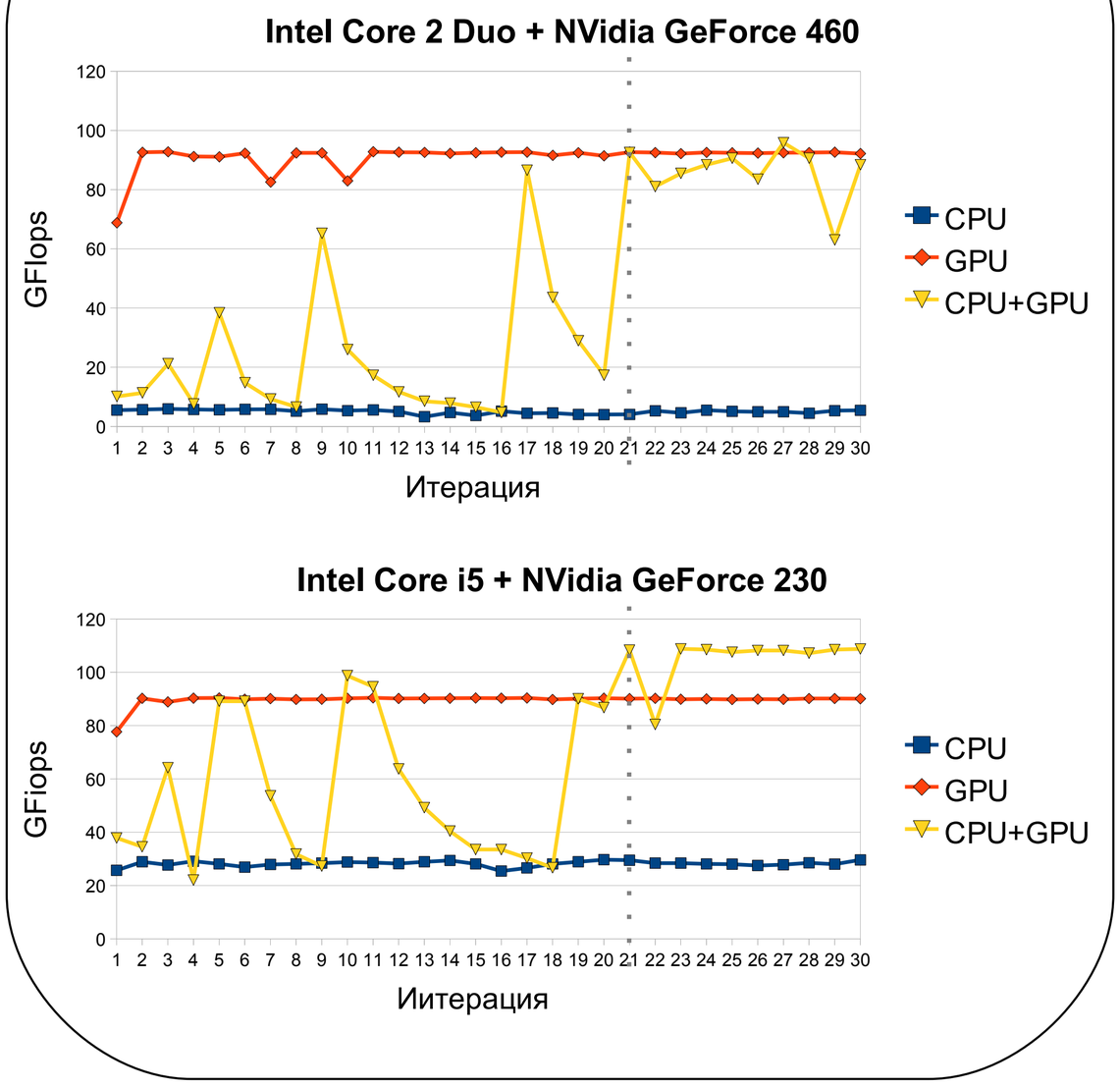
Шаг 3. Совместное использование CPU+GPU

```
//...
ProcessCPU(data, 0, 128);
ProcessGPU(data, 128, 1024);
//...
```

⇒

```
//...
hybrid_for f;
f.CUDA() += ProcessGPU;
f.CPU()  += ProcessCPU;
f.Process(data, 0, 1024);
//...
```

Результаты тестирования (задача N тел)



Типы параметров

Типы параметров

Parameter<T> - аналог обычных типов данных

Примеры: `Parameter<float> p1; Parameter<bool> p2;`

RangedParameter<T> - ограничение обычного типа данных

Примеры: `RangedParameter<float> p(0.1f, 1.0f);`

ParameterEnum — набор элементов

Примеры: `ParameterEnum p(«a, b, c, d»);`

ParameterGrid — сетка для 1/2/3-мерной области.

Примеры: `ParameterGrid p(0.0, 128.0, 4);`

ParameterAction — сигнализирует о действии пользователя

Примеры: `ParameterAction p; p.OnChange() += Stop;`

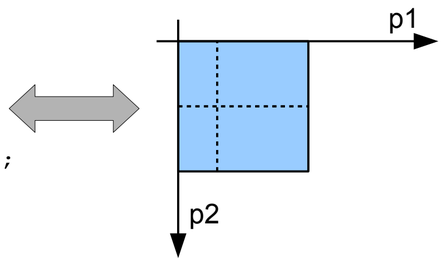
Внутреннее представление

Все параметры отображаются в N-мерный гиперкуб со сторонам, равными единице. Для каждого измерения используется информация о типе, которая может учитываться при оптимизации (например — дискретный или непрерывный параметр, сильно ли он влияет на скорость работы, где находится предполагаемый экстремум).

```
RangedParameter<int> p1(0, 128);
P1 = 42;

RangedParameter<float> p2(-1.0f, 1.0f);
p2 = 0.0f;
```

⇔



Модель оптимизации

Задача сводится к поиску экстремума функции времени в заданном гиперкубе. Один запуск программы или одна итерация алгоритма — одно измерение функции времени.

Режимы работы:

Агрессивный — пытается найти новые локальные экстремумы. Используется первые N итераций.

Основной — пытается остаться в выбранном локальном экстремуме (который может «бегать»).

Используемые алгоритмы:

Градиентный (основной)

Покоординатного спуска (для небольшого количества параметров)

Генетический (в разработке)