Вычисления на GPU средствами Matlab Parallel Computing Toolbox

САМОСТОЯТЕЛЬНАЯ РАБОТА

http://www.kimrt.ru
Онлайн-курс
Суперкомпьютерные технологии в задачах моделирования

Немного о вычислениях на GPU

Многоядерные машины и технология гиперпоточности позволили ученым, инженерам и финансовым аналитикам ускорить выполнение ресурсоемких приложений в различных дисциплинах. Сегодня еще один тип оборудования обещает еще более высокую вычислительную производительность. Это графический процессор (GPU).

Первоначально используемые для ускорения рендеринга графики, графические процессоры все чаще применяются для научных вычислений. В отличие от традиционного центрального процессора (ЦП или CPU), который включает в себя не более нескольких ядер, GPU имеет массивно-параллельный набор целочисленных процессоров и процессоров с плавающей запятой, а также выделенную высокоскоростную память. Типичный графический процессор состоит из сотен таких процессоров меньшего размера (рис.1).

Однако значительно увеличенная пропускная способность, ставшая возможной благодаря графическому процессору, имеет свою цену. Дело в том, что данные должны быть отправлены из CPU в GPU перед вычислением, а затем извлечены из него. Поскольку графический процессор подключен к центральному процессору через шину PCI Express, доступ к памяти происходит медленнее, чем с традиционным процессором. Это означает, что общее ускорение вычислений ограничено объемом передаваемых в алгоритме данных.

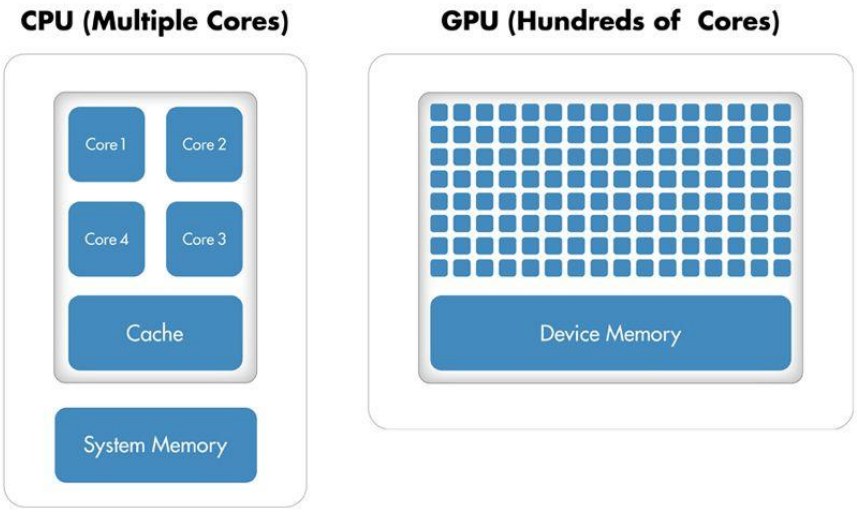

Рис. 1. Сравнение количества ядер у CPU и GPU

Далее обсудим возможности Parallel Computing Toolbox для вычислений на GPU [1].

Отличия синтаксиса вычислений на CPU и GPU в Matlab

Рассмотрим пример. Допустим, мы хотим вычислить БПФ (быстрое преобразование Фурье) какого-либо массива. Для этого в Matlab есть функция *fft*, которая имеет перегрузку на тип входных данных. Это значит, что можно дать ей на вход, например, обычный массив типа *double*, а можно дать массив типа `GPUarray`, специально предназначенного в Parallel Computing Toolbox для вычислений на GPU. Подобных функций в Matlab сотни [2].

Теперь на примере БПФ продемонстрируем отличия работы с GPU. Зададим одномерный массив случайных чисел с помощью функции `rand()` и проведем БПФ с помощью функции `fft()`:

```
A = rand(2^16,1);

B = fft(A);
```

Чтобы выполнить ту же операцию на графическом процессоре, мы сначала используем команду *gpuArray* для передачи данных из

рабочей области Matlab в память устройства. Затем мы можем запустить функцию *fft*.

```
A = gpuArray(rand(2^16,1));

B = fft(A);
```

Можно продолжить работу с массивом *B*, используя функции, поддерживающие работу с графическим процессором. Например, чтобы визуализировать наши результаты, можно использовать команду *plot*, которая поддерживает *gpuArrays*.

```
plot(B);
```

Чтобы вернуть данные обратно в рабочую область Matlab, следует использовать функцию *gather*.

```
C = gather(B);
```

Теперь *C* является массивом типа double в Matlab и может работать с любой из функций Matlab, которые работают с *double*.

В этом простом примере время, сэкономленное при выполнении одной функции *fft*, может быть меньше времени передачи вектора из рабочей области Matlab в память устройства. Задержки при передаче данных могут стать настолько значительными, что это ухудшит общую производительность программы, особенно если нужно постоянно обмениваться данными между CPU и GPU для выполнения небольшого количества операций на GPU. Более эффективно выполнять несколько операций с данными, пока они находятся на GPU, возвращая данные обратно в CPU только при необходимости.

Пример программы на GPU

В этом примере показано, как использовать функции Matlab с поддержкой *gpuArray* [3].

Создадим вектор-строку, который повторяет значения от -15 до 15. Чтобы передать его в GPU и создать объект *gpuArray*, используем функцию *gpuArray*.

```
X = [-15:15 0 -15:15 0 -15:15];
gpuX = gpuArray(X);
```

Используем функции *expm*, *diag*, *mod*, *round*, *fliplr*, которые поддерживают работу с GPU.

```
gpuE = expm(diag(gpuX,-1)) * expm(diag(gpuX,1));
gpuM = mod(round(abs(gpuE)),2);
gpuF = gpuM + fliplr(gpuM);
```

Выводим результаты в виде графиков на экран.

```
imagesc(gpuF);
colormap(flip(gray));
```

И опять же, если нужно вернуть данные обратно в рабочую область Matlab (на CPU), нужно воспользоваться *gather*

```
result = gather(gpuF);
```

Рассмотрим функции, использованные в данном примере. Функция *expm* находит экспоненты от параметров, которыми являются элементы матрицы, заданной на вход; *diag* создаёт диагональную матрицу из вектора; *mod* позволяет найти остаток от деления; *round* округляет до ближайшего целого числа; *fliplr* располагает элементы одномерного массива в обратном порядке; *imagesc* создаёт картинку из матрицы; *colormap* позволяет настроить палитру для графика; *flip* переворачивает матрицы разных типов данных и размерностей; *gray* возвращает массив серой палитры. Все указанные функции поддерживают работу на GPU.

На рис. 2 показаны результаты выполнения данной программы.

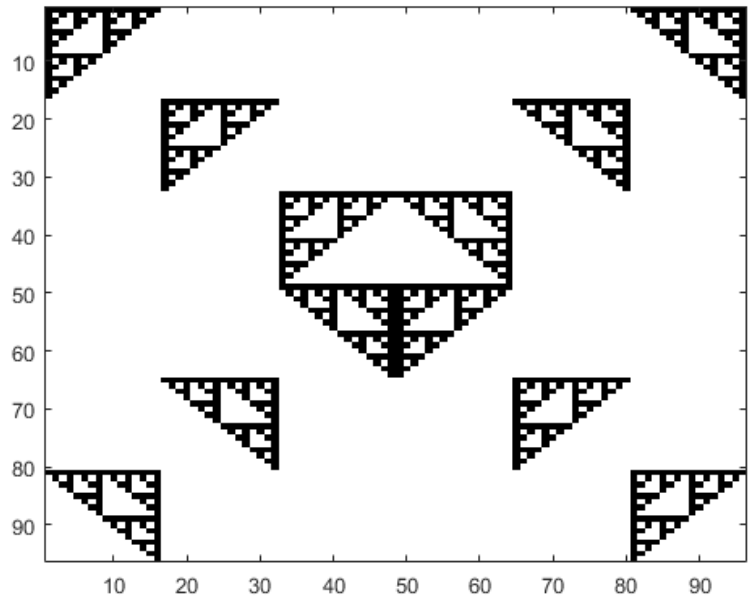

Рис. 2. Результаты выполнения программы на GPU [3]

Критерии, которым должна соответствовать программа, чтобы расчёты в ней можно было ускорить с помощью GPU

Задача, которую можно успешно распараллелить на GPU должна:

1. Предполагать вычислительную интенсивность. Иными словами, время, затрачиваемое на вычисления, должно значительно превышать время, затрачиваемое на передачу данных в память графического процессора и из нее;
2. Иметь возможность массового распараллеливания, то есть вычисления могут быть разбиты на сотни или тысячи независимых исполнителей.

Также необходимо учитывать, что на данный момент Matlab поддерживает расчёты на GPU только от фирмы NVIDIA, базируясь на программно-аппаратной архитектуре параллельных вычислений CUDA.

http://www.kimrt.ru/index/course_stm/0-24

Проект реализуется победителем Конкурса на предоставление грантов преподавателям магистратуры 2020/2021

http://www.kimrt.ru

благотворительной программы «Стипендиальная программа Владимира Потанина» Благотворительного фонда Владимира Потанина.

Источники

1. Parallel Computing Toolbox - MATLAB [Электронный ресурс] // MathWorks - URL: https://www.mathworks.com/products/parallel-computing.html, – (дата обращения: 29.04.2022).
2. GPU Programming in MATLAB [Электронный ресурс] // MathWorks - URL: https://www.mathworks.com/company/newsletters/articles/gpu-programming-in-matlab.html, – (дата обращения: 03.05.2022).
3. Run MATLAB functions on GPU [Электронный ресурс] // MathWorks - URL: https://www.mathworks.com/help/parallel-computing/run-matlab-functions-on-a-gpu.html , – (дата обращения: 13.05.2022).