А. Н. Полуянов

Расчет диагностической шкалы на графических процессорах

Омский филиал Федерального государственного бюджетного учреждения науки Института математики им. С.Л. Соболева Сибирского отделения Российской академии наук.

В работе рассматривается технология расчета медицинских диагностических шкал. Описана реализация параллельного алгоритма расчета шкалы с использованием графических процессоров (технология CUDA).

Введение

Анализ накопленной информации является актуальной проблемой для многих исследовательских и прикладных задач. Традиционным способом ее решения в настоящее время является технология оперативной аналитической обработки данных OLAP (online analytical processing). Основой OLAP-технологии является построение многомерных представления данных.

Можно выделить следующие шаги при работе с данными:

1. Исходные данные представлены в реляционном нормализованном виде, и к ним обеспечивается доступ по технологии OLTP (online transaction processing);
2. Пользовательское многомерное представление данных, реализующее технологию OLAP, обеспечивается инструментарием, преобразующим исходные данные в гиперкуб.
3. Пользовательское представление далее используется для анализа данных.

Реализация данной технологии с использованием последовательных алгоритмов подробно представлена в работах [1,2].

Время работы алгоритмов формирования пользовательского представления данных и анализа данных можно значительно сократить, используя технологию CUDA [3,4], предназначенную для разработки приложений исполняемых на графических процессорах.

Графический процессор представляет собой вычислительное устройство, которое:

- является сопроцессором к центральному процессору (CPU);
- имеет собственную память;
- выполняет одновременно большое количество нитей (аналог потоков CPU);

Код приложения, разработанного по технологии CUDA, состоит как из последовательных, так и из параллельных частей. Последовательная часть выполняется на CPU, а параллельная часть оформляется в виде функции ядра (kernel function) и выполняется на графическом процессоре.

В данной работе рассмотрен этап анализа данных, а именно реализация параллельного алгоритма расчета диагностической шкалы с использование технологии CUDA.

Описание задачи

Традиционно для расчета диагностической шкалы [5] используется линейная комбинация N значимых параметров, называемая в литературе решающей функцией [6,7]:

$$F(x)=a_1x_1+a_2x_2+\ldots+a_Nx_N,$$

где $x=(x_1,x_2,\ldots,x_N)$ – вектор значений выделенных параметров (координат в пространстве параметров), $a=(a_1,a_2,\ldots,a_N)$ – веса выделенных параметров (коэффициенты).

Для значений функции $F(x)$ определяются границы (оценочная шкала):

$$g_0, g_1, \ldots, g_K,$$

где K – количество групп объектов O_1, O_2, …, O_K. При условии, что $g_0<g_1<\ldots<g_K$, определение принадлежности произвольного объекта o с вектором значений параметров x' к группе O_j сводится к проверке выполнения неравенства:

$$g_{j-1}<F(x')<g_j.$$

При выполнении равенства значения функции F какой-либо границе $F(x')=g_j$ возникает ситуация неопределенности.

Для определения значений коэффициентов $(a_1,a_2,\ldots,a_N)$ и значений границ g_0, g_1, …, g_K, в распознавании образов традиционно используются обучающие выборки, заданные множеством групп объектов O_1, O_2, …, O_K. Пусть объект $o_{ij}\in O_i$ характеризуется вектором значений параметров: $x_{ij}=(x_{ij1},x_{ij2},\ldots,x_{ijN})$. Функционалом риска выберем суммарное количество ошибок E при отнесении объекта к группе. Для текущих значений вектора $(a_1,a_2,\ldots,a_N)$ и границ g_0, g_1, …, g_K значение функционала риска вычисляется в следующем алгоритме:

```
E=0;
for i=1 to K;
  for each o_ij in O_i;
    if F(x_ij)<g_{i-1} or F(x_ij)>g_i then E=E+1;
  endfor;
endfor;
```

В данном алгоритме ситуация неопределенности не считается ошибкой.

Таким образом, задача построения оценочной шкалы может быть записана в следующем виде:

$$E \rightarrow min,\ g_0<g_1<\ldots<g_K,\ -1 \le a_i \le 1,\ i=1,2,\ldots,N.$$

Ограничения на коэффициенты a_i реализуются за счет масштабирования. Общая схема решения задачи следующая:

1) выбирается начальное приближение для вектора весов $a^o=(a^0_1,a^0_2,\ldots,a^0_N)$;

2) циклически для $i=1,2,\ldots,N$ фиксируются все веса, кроме a_i, осуществляется спуск по координате a_i с определенным шагом, на каждом шаге выполняются пункты 3-6;

3) для текущего набора весов $a=(a_1,a_2,\ldots,a_N)$ вычисляются значения $F_{ij}=F(x_{ij})$ для всех групп и всех объектов в группах с запоминанием номера группы для каждого значения;

4) значения F_{ij} сортируются по возрастанию;

5) $g_0=min(F_{ij})-\varepsilon$, $g_K=max(F_{ij})+\varepsilon$, где ε – малая величина;

6) остальные значения границ $g_1, \ldots, g_{K-1}$ определяются перебором возможных вариантов при условии минимизации функционала риска и выполнения неравенств: $g_0<g_1<\ldots<g_K$, причем, значения выбираются посредине между соседними F_{ij}, чтобы не было ситуации неопределенности на обучающей выборке.

Последний шаг является наиболее трудоемким с вычислительной точки зрения.

Последовательная реализация представленных алгоритмов описана в работах [8,9].

Параллельный алгоритм расчета шкалы

Специфика вычислительных устройств, построенных на базе графических процессоров, состоит в том, что программа для своего выполнения задействует как центральный процессор (CPU), так и графические процессоры (GPU). Поскольку на CPU выполняется последовательный код, то для ускорения работы программы основная часть вычислений должна выполняться на графических процессорах. Так же для ускорения работы программы необходимо оптимизировать работу с памятью.

При анализе последовательного алгоритма построения диагностической шкалы выявлено, что наибольшее время затрачивается на выполнение шага 6 - 96% от общего времени работы программы, таким образом, по закону Амдаля теоретически можно получить 25-кратный прирост производительности параллельной программы.

В рассматриваемом алгоритме на GPU параллельно выполняются наиболее трудоемкий шаг 6. Функция ядра (выполняемая на графическом процессоре) на входе получает массив с рассчитанными значениями решающей функции F_{ij} для каждого объекта выборки при текущем наборе весов. Каждая нить, в зависимости от своего номера, рассчитывает значение функционала риска E для определенного набора границ $g_0, \ldots, g_K$. Полученные результаты записываются в массив, в котором в качестве индекса выступает номер нити, а в качестве элемента – значения функционала риска, рассчитанного нитью для своего обрабатываемого

набора границ. После завершения вычислений на GPU, CPU определяет минимальное значение функционала риска для текущего набора весов и выполняется следующий шаг алгоритма.

Поскольку при работе функции ядра происходят частые обращения к массиву значений решающей функции $\boldsymbol{F_{ij}}$, данный массив целесообразно поместить в константную память графического процессора, обладающую меньшей латентностью, по сравнению с глобальной памятью.

Параллельный алгоритм позволил значительно сократить временные затраты по обработке данных при построении диагностических шкал. Результаты расчетов показали в среднем десятикратное ускорение вычислений по сравнению с последовательным алгоритмом.

Отдельно хотелось бы остановиться на моменте оптимизации программного кода. Конечное время выполнения программы сильно зависит от архитектуры используемого графического процессора. Так, например, использование константной памяти на графических процессорах с архитектурой Tesla для хранения массива значений решающей функции (версия вычислителя до 1.3) позволило получить двукратное ускорение работы алгоритма. Анализ алгоритма в NVIDIA Visual Profiler показал, что при использовании константной памяти основное время работы графического процессора использовалось для выполнения инструкций функции ядра, в то время как без использования константной памяти основное время работы графического процессора уходило на запросы к глобальной памяти вычислителя. При запуске алгоритма на графических процессорах с архитектурой Fermi (версия вычислителя 2.0 и выше) значительных ускорений работы алгоритма при использовании константной памяти не наблюдалось, т.к. в архитектуре Fermi улучшена работа глобальной памяти GPU, добавлены дополнительные кэши.

При использовании в функции ядра операций с плавающей точкой скорость вычислений также определяется архитектурой GPU. В архитектуре Fermi, в отличие от Tesla, каждое ядро может производить вычисления с плавающей точкой, что существенно повышает скорость работы алгоритмов.

Также возможны некоторые сложности при распараллеливании алгоритма на несколько графических процессоров. Разработанный алгоритм использовался для вычислений на суперкомпьютере Tesla ОФ ИМ СО РАН, представляющим из себя гибридный кластер из 3-х узлов (2 узла из трех графических процессоров с архитектурой Tesla, 1 узел из двух графических процессоров с архитектурой Fermi). Для распараллеливания алгоритма на несколько графических процессоров с архитектурой Tesla, необходимо для каждого

графического процессора создавать отдельный поток на CPU. На графических процессорах с архитектурой Fermi один поток CPU может использовать сразу несколько графических процессоров (функции ядра могут вызываться в цикле по доступным GPU, смена текущего GPU осуществляется путем вызова функции CudaSetDevice). Также на вычислителях с архитектурой Fermi все графические процессоры одного узла имеют единое виртуальное адресное пространство, что позволяет осуществлять копирование данных глобальной памяти между различными GPU напрямую без использования оперативной памяти узла.

Представленные особенности технологии CUDA необходимо учитывать при разработке, отладке и оптимизации программного кода.

Заключение

Использование графических процессоров позволяет значительно повысить эффективность расчета диагностических шкал. Несмотря на некоторые сложности оптимизации параллельного алгоритма, связанные с активным развитием архитектуры графических процессоров и технологии их программировании, применение графических процессоров оправдало себя, позволив уменьшить на порядок временные затраты по обработке исходных данных.

В настоящее время продолжается работа по данной тематике. В перспективе планируется создание программного комплекса формирования и анализа многомерных данных с использованием графических процессоров.

Работа выполнена по проекту РФФИ № 12-07-00066-а.

Список литературы

1. Зыкин, С. В. Формирование гиперкубического представления реляционной базы данных / С. В. Зыкин // Программирование. – 2006. – № 6. – С. 71–80.
2. Полуянов, А. Н. Автоматизация формирования гиперкубического представления данных / А. Н. Полуянов // Системы управления и информационные технологии. – 2008. – № 2.2(32). – С. 289–293.
3. Боресков, А. В. Основы работы с технологией CUDA / А. В. Боресков, А. А. Харламов. – М. : ДМК Пресс, 2010. – 232 с.
4. Сандерс, Дж. Технология CUDA в примерах: введение в программирование графических процессоров / Дж. Сандерс, Э. Кэндрот. – М. : ДМК Пресс, 2011. – 232 с.

5. Александрович, Ю. С. Оценочные и прогностические шкалы в медицине критических состояний. Справочник / Ю. С. Александрович, В. И. Гордеев. – СПб. : Сотис, 2007. – 137 с.
6. Журавлев, Ю. И. Об алгебраическом подходе к решению задач распознавания или классификации / Ю. И. Журавлев // Проблемы кибернетики. – 1978. – Т. 33. – С. 5–68.
7. Лбов, Г. С. Метод адаптивного поиска логической решающей функции / Г. С. Лбов, В. М. Неделько, С. В. Неделько // Сибирский журнал индустриальной математики. – 2009. – Т. XII, № 3(39). – С. 66–74.
8. Зыкин, С. В. Технология подготовки и анализа данных для построения медицинских оценочных шкал / С. В. Зыкин, А. Н. Полуянов, А. К. Чернышев, А. И. Ревзин // Информационные технологии. – 2010. – № 12. – С. 57–62.
9. Зыкин, С. В. Формирование представлений данных для построения медицинских диагностических шкал / С. В. Зыкин, П. Г. Редреев, А. К. Чернышев // Омский научный вестник. Серия "Приборы, машины и технологии". – 2011. – № 2(100). – С. 190–193.