DOUDKIN A. A., PERTSEV D.Y. Remote Sensing data compression using context model

In the paper the hyperspectral data compression algorithm is presented. It is accented to a possibility of its application in limited computing resources of the onboard system of the aircraft. The compression ratio was estimated using the AVIRIS Maine dataset.

УДК 004.89

Краснопрошин В. В., Мацкевич В. В.

ЭФФЕКТИВНАЯ ОБРАБОТКА ДАННЫХ НА ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВАХ

Введение. В настоящее время наблюдается быстрый рост вычислительной сложности прикладных задач, которая во многом зависит от объемов обрабатываемых данных, необходимых для их решения [1]. Одновременно с этим растет и вычислительная мощность современных компьютеров. Последнее, как правило, происходит путем увеличения количества вычислительных устройств не только в рамках одного компьютера, но и путем формирования различного рода вычислительных кластеров [2]. На сегодняшний день актуальной стала проблема, связанная с эффективным использованием вычислительных ресурсов [2]. Вычислительные устройства при решении прикладных задач могут оказаться гетерогенными (разными по архитектуре и мощности), в этом случае возникает необходимость эффективной их загрузки [3]. Что является нетривиальной математической задачей [4].

В работе предлагается один из возможных вариантов решения проблемы с использованием технологии распараллеливания данных.

1. Формализация задачи. Любую прикладную задачу (Z) в самом общем виде можно представить двумя составляющими: данными (D) и алгоритмом решения (A) [5], т. е.

$$Z = (D, A) \tag{1}$$

В свою очередь процесс решения можно задать сетевым графиком (G), который состоит из набора вычислительных операций (B) и зависимостей между ними (E) [6], т. е.

$$A = G(B, E). \tag{2}$$

При этом порядок выполнения B может быть не однозначным [6].

В результате прикладную задачу можно записать в виде:

$$Z = (D, G(B, E)). \tag{3}$$

Предположим, что любая операция b_i, ($i = \overline{1,|B|}$) из B является неделимой и требует объема вычислений, равного единице, а используемый для решения задачи компьютер содержит n вычислительных устройств, и каждое i-е устройство ($i = \overline{1,n}$) имеет свою вычислительную мощность U_i.

Предположим также, что время решения T зависит от специфики задачи, мощности вычислительных устройств и нагрузки на устройства (Y) [7], т. е.

$$T = F(Z, U, Y), \tag{4}$$

где U – n мерный вектор, координаты которого задают мощность отдельного вычислительного устройства, а F – функционал, определяющий время решения задачи Z при заданных вычислительных устройствах U с нагрузкой Y.

Далее предположим, что нагрузка Y определяется разбиением множества операций B на n подмножеств B_i, $i = \overline{1,n}$, каждое из которых задает набор операций, выполняемых на i-м вычислительном устройстве.

В результате получаем оптимизационную задачу, связанную с минимизацией функционала F [7].

Для задачи Z с использованием имеющихся вычислительных устройств мощности U_i, $i = \overline{1,n}$ необходимо определить нагрузку на эти устройства Y, при которой время решения задачи будет минимальным, т. е. необходимо определить нагрузку Y^*, удовлетворяющую соотношению:

$$F(Z, U, Y) = \arg\min_{Y} F(Z, U, Y). \tag{5}$$

2. Алгоритм вычислений. В общем случае задача минимизации функционала (5) является достаточно сложной, поэтому рассмотрим частный случай, когда на задачу Z наложен ряд ограничений.

Из соотношения (3) видно, что задача Z зависит от исходных данных D и сетевого графика G, который задает процесс ее решения. Сетевой график, в свою очередь, задается набором вычислительных операций B и связей между ними E.

Предположим, что каждый набор операций B содержит в себе ненулевое количество непересекающихся между собой поднаборов S_q, причем суммарное количество операций, содержащихся в поднаборах S_q соизмеримо с количеством операций в наборе B:

$$\begin{cases} S_q \subseteq B, 1 \le q \le M \\ \sum\limits_{i=1}^{M} |S_q| = O(|B|) \\ S_a \cap S_b = \varnothing, \forall a \ne b \\ M \in \mathrm{N} \end{cases} \tag{6}$$

Предположим также, что каждый поднабор операций S_q, $q = \overline{1,M}$ из B обладает следующими свойствами:

1. S_q, $q = \overline{1,M}$ зависит только от одной вычислительной операции b_q из набора B или является независимым.

2. S_q, $q = \overline{1,M}$ допускает разбиение на m_q непересекающихся поднаборов P_{qi}, $i = \overline{1,m_q}$ одинаковой мощности, где m_q достаточно большое число, т. е. :

$$\begin{cases} \bigcup\limits_{i=1,m} P_{qi} = S_q, \forall q = \overline{1,M} \\ P_{qi} \cap P_{qj} = \varnothing \ , \forall i \ne j, \forall q = \overline{1,M} \\ |P_{qi}| = |P_{qj}| = |P_q|, \forall i, j, \forall q = \overline{1,M} \\ m_q >> 1, \ m_q \in \mathrm{N} \ , \forall q = \overline{1,M} \end{cases} \tag{7}$$

3. Любые пары P_{qi} и P_{qj}, $\forall i \ne j$ из S_q, $q = \overline{1,M}$ независимы друг от друга.

Можно показать, что при сделанных предположениях задача минимизации функционала (5) сводится к решению M оптимизационных задач (для каждого отдельного S_q, $q = \overline{1,M}$):

$$F(D_q, G(S_q, E_q), U, Y_q) \xrightarrow[Y_q]{} \min, \tag{8}$$

где D_q – данные, необходимые для выполнения операций из набора S_q, E_q – связи между операциями, Y_q – разбиение отдельного

Мацкевич Вадим Владимирович, *магистрант кафедры информационных систем управления Белорусского государственного университета.*

220050, г. Минск, пр. Независимости, 4.

набора S_q, $q=\overline{1,M}$ на n подмножеств.

Пусть S_{qi} – набор операций, обрабатываемых на i-м устройстве, и T_{qi} – время его обработки, $i=\overline{1,n}$.

Необходимо построить разбиение набора S_q, которое минимизируем общее время обработки всего набора.

Формально эту задачу можно записать следующим образом:

$$\begin{cases} S_{qi}\cap S_{qj}=\varnothing \\ \bigcup\limits_{i=\overline{1,n}} S_{qi}=S_q \\ \max\limits_{i=\overline{1,n}} T_{qi}\to\min \end{cases} \quad (9)$$

Замечание. В задаче важно нахождение значений S_{qi}, удовлетворяющих (9).

Нетрудно показать, что для получения минимального времени обработки необходимо выполнение двух условий: вычислительные устройства должны (а) работать одновременно и (б) одинаковое количество времени.

С учетом этого задачу можно переписать в виде:

$$\begin{cases} \dfrac{k_{qi}|P_q|}{U_i}=\dfrac{k_{qn}|P_q|}{U_n},\forall i=\overline{1,n-1} \\ |P_q|\sum\limits_{i=1}^{n} k_{qi}=|S_q| \end{cases} \quad (10)$$

Замечание. Ограничение на покрытие набора S_q является несущественным, поэтому оно не отображено в системе (10).

Зафиксируем параметры U_i, $i=\overline{1,n}$ и решим систему (10):

$$\begin{cases} k_{qi}=\dfrac{U_i}{U_n}k_{qn},\forall i=\overline{1,n-1} \\ \left(\sum\limits_{i=1}^{n-1}\dfrac{U_i}{U_n}+1\right)k_{qn}=\dfrac{|S_q|}{|P_q|} \end{cases}$$

$$\begin{cases} k_{qn}=\dfrac{|S_q|}{|P_q|\left(\sum\limits_{i=1}^{n-1}\dfrac{U_i}{U_n}+1\right)} \\ k_{qi}=\dfrac{U_i}{U_n}k_{qn},\forall i=\overline{1,n-1} \end{cases} \quad (11)$$

При вычислении k_{qi} неизвестными в системе (11) остаются соотношения мощностей вычислительных устройств. Возможны два способа их получения:

1) можно использовать табличные значения мощностей вычислительных устройств, указанных производителем. Однако это не совсем подходит в случае видеокарт. В соответствии со своей архитектурой любая видеокарта содержит большое число ядер (порядка тысячи). При решении конкретной задачи далеко не все ее этапы можно распараллелить на такое большое число ядер, более того, в процессе вычислений периодически может возникать проблема синхронизации вычислений. Поэтому одна и та же видеокарта в зависимости от специфики задачи может обладать разной мощностью (иногда десятикратной разницы);

2) можно экспериментально вычислить соотношение мощностей вычислителей.

В этом случае предлагается следующий способ оценки мощности вычислительного устройства.

Пусть x_q - некоторое подмножество операций S_q, необходимое для обработки набора S_q и τ_{qi} – время работы i-го устройства на данном подмножестве. Тогда получим систему уравнений:

$$\tau_{qi}=\frac{|x_q|}{U_i},\forall i=\overline{1,n} \quad (12)$$

Нетрудно заметить, что время работы вычислительного устройства на фиксированном количестве операций обратно пропорционально его мощности.

Для того чтобы получить искомое соотношение мощностей, последнее (n-ое) уравнение разделим последовательно на n-1 предыдущих.

$$\frac{U_i}{U_n}=\frac{\tau_{qn}}{\tau_{qi}},\forall i=\overline{1,n-1} \quad (13)$$

Выражение (13) означает, что соотношение мощностей вычислительных устройств обратно пропорционально соотношению времен, затраченных на обработку фиксированного количества операций.

Полученное выражение достаточно точно отражает соотношение мощностей, т. к. оно учитывает реальные условия, в которых производятся вычисления.

На основе проделанных рассуждений предлагается подход к эффективному использованию вычислительных ресурсов компьютера, основанный на технологии распараллеливания данных.

Формально его можно описать в виде алгоритма, состоящего из последовательности следующих шагов.

Для всех S_q, $q=\overline{1,M}$ выполнить:

Шаг 1. Из общего множества операций выделить (небольшое по мощности) подмножество необходимых для обработки набора S_q.

Шаг 2. Выполнить эти операции (с замером времени на их исполнение) на каждом из устройств.

Шаг 3. С использованием формулы (13) определить соотношение мощностей.

Шаг 4. По формуле (11) вычислить оптимальный объем загрузки для каждого вычислительного устройства. Для этого:

- множество операций S_q разбивается на m_q подмножеств P_{qj}, $j=\overline{1,m_q}$;
- для каждого i-го устройства $i=\overline{1,n}$ объединением P_{qj} формируется подмножество S_q, т. е. имеем:

$$\begin{cases} S_{qi}=\cup_{h=\overline{1,k_{qi}}} P_{qj_{ih}} \\ \bigcup\limits_{i=\overline{1,n}} S_{qi}=S_q \\ S_{qi}\cap S_{qj}=\varnothing,\forall i\neq j \end{cases} \quad (14)$$

Нетрудно показать, что построенное таким образом разбиение Y_q доставляет минимум функционала (8).

Замечания:

1. Среди вычислительных устройств не должно быть сильно различающихся по мощности. В противном случае, из-за накладок по передаче данных и их синхронизации удаление наиболее слабых устройств не увеличит общее время обработки.

Будем считать, что устройства соизмеримы друг с другом по мощности, если выполняется условие:

$$\max_{i=\overline{1,n}} U_i\le c\min_{j=\overline{1,n}} U_j, \quad (15)$$

где c – константа, равная двум.

2. Набор операций S_q, $q=\overline{1,M}$ должен быть достаточно большим, чтобы имело смысл распараллелить их обработку. В противном случае, затраты на передачу данных между устройствами и их синхронизацию могут свести преимущества параллельной обработки к нулю.

3. После вычислений коэффициентов k_{qi} по формуле (11) некоторые из них могут оказаться дробными числами. Необходимо осуществить их округление до целого с учетом ограничения (10). Если m_q существенно больше n, округления несущественно замедлят вычисления. В противном случае необходимо проводить округление до целого по математическим правилам. Затем в случае "недобора", "свободные" наборы операций отдать вычислителям в порядке убывания мощности устройств. В случае "перебора" – убрать один набор операций с устройств в порядке возрастания мощности устройств.

Можно отметить, что предложенный выше подход обладает рядом преимуществ:

1. При решении прикладных задач можно эффективно использовать вычислительные ресурсы стандартного компьютера, содержащего, как правило, многоядерный процессор и видеокарту.
2. Гарантируется высокая точность оценки оптимальных коэффициентов в (11), т. к. при решении задачи замеры осуществляются в "реальных" условиях с учетом всех задержек на синхронизацию потоков, передачу данных и т. п.
3. В процессе решения задачи не требуется никаких дополнительных вычислений, а просто осуществляет решение системы (11).

Кроме того, подход также обеспечивает достижение минимума функционала (8), а с учетом ограничений (6), (7) – значение функционала (5) близкого к оптимальному его значению.

Возможности предложенного подхода рассмотрим на примере решения задачи сжатия цветных изображений с использованием нейронной сети прямого распространения.

3. Пример решения прикладных задач. В качестве исходных использовалась данные «STL-10» из репозитория Стенфордского университета [8]. Выборка содержит сто тысяч немаркированных цветных изображений размером 96x96 пикселей. Каждое изображение описывается 27648 целыми числами (в диапазоне от 0 до 255), задающими содержание красного, зеленого и синего цветов [9]. Исходя из полученных характеристик (выборка задается примерно $2{,}8*10^9$ числами, содержит описания произвольных, немаркированных объектов), можно сделать вывод, что процесс сжатия изображений данной выборки с малыми потерями является достаточно сложной задачей.

Для обработки данных использовался стандартный компьютер с 8-ядерным процессором и видеокартой:
видеокарта nvidia – 1050 2gb; процессор – amd ryzen 7 1700 3.0 GHz; ОЗУ: 2x8 Gb 3200 MHz; жесткий диск – samsung 850 pro 256 Gb; операционная система – Lubuntu 16.04.3.

Видеокарта компьютера мощнее процессора примерно на 60%. Мощности хотя и различны, но соизмеримы друг с другом, следовательно, данная конфигурация вычислительного гетерогенного устройства соответствует отмеченным выше условиям.

В качестве программного обеспечения использовался компилятор nvcc 7.5 (библиотеки CUDA (Версия 9.1 [10]) и OpenMP) с опциями:
nvcc -D_FORCE_INLINES -O2 --machine 64 -lgomp -Xcompiler -fopenmp program_code.cu.

Замер времени операций проводился с помощью функции «gettimeofday».

Замечание. Опция компилятора «--machine 64» показывает, что приложение является 64-битным, следовательно, оно несовместимо с 32 разрядными системами.

При обучении нейронной сети использовался пакетный режим. Размер пакета был фиксированным и равным 6561 изображению. Последнее было обусловлено следующими соображениями:

размер обучающей выборки – сто тысяч изображений. Данное число не делится на три, следовательно, степень тройки может быть выбрана в качестве размера пакета для максимизации НОК и, следовательно, в качестве периода повтора пакетов. Размер пакета – три в восьмой степени, отсюда период повторения пакетов составляет 656100000 изображений.

В рамках вычислительного эксперимента были рассмотрены три различных случая:

- обработка пакета изображений осуществляется только на процессоре,
- затем только на видеокарте
- и, наконец, одновременно на процессоре и видеокарте (с применением разработанного подхода).

Общая схема проведения экспериментов включала следующие основные этапы:

1. Считывание входных данных.
2. Предварительная обработка изображений.
3. Инициализация начальных значений весов нейронной сети.
4. «Бесконечный» цикл обучения (каждый пакет включает 6561 изображение).

4.1. Процесс обучения без учителя.

4.2. Замер времени обучения по 15 пакетам и проверка условия выхода из цикла.

Критерий выхода из цикла: «суммарная среднеквадратическая ошибка по 15 пакетам изображений увеличилась».

В процессе реализации эксперимента (в соответствии с описанным подходом) выполнялись следующие действия:

- на первой итерации «бесконечного» цикла на видеокарте обрабатывается первая половина пакета изображений и замеряется время обработки τ_{gpu};
- после этого вторая половина пакета обрабатывается на процессоре и замеряется время обработки τ_{cpu};
- далее с использованием формул (13) и (11) вычисляется количество изображений, которые подаются соответственно на процессор и видеокарту;
- после этого проводится корректировка весов нейронной сети;
- далее на процессоре и видеокарте одновременно (в цикле) обрабатывается вычисленное на первой итерации количество изображений пакета и снова происходит корректировка весов нейронной сети.

4. Результаты экспериментов. По итогам проведенных экспериментов были получены следующие результаты (см. таблицу 1):

Таблица 1 – Время решения задачи

Используемые вычислительные устройства	Процессор	Видеокарта	Процессор и видеокарта
Время обработки (в сек.)	40	24	15

Сравним результаты, полученные в процессе третьего эксперимента (совместное использование процессора и видеокарты), с оценкой времени работы “тривиального алгоритма”, в котором количество вычислительных операций между гетерогенными устройствами распределялось бы поровну.

Нетрудно подсчитать, что в этом случае время работы “тривиального алгоритма” равно 20 секундам, т. е. выигрыш по времени в итоге составляет около 30 %.

Таким образом, за счет рациональной загрузки гетерогенных вычислительных устройств можно снизить общее время обработки данных, что является важным при решении прикладных задач в условиях реального времени.

5. Заключение. В работе рассмотрена проблема организации эффективной обработки данных на гетерогенных вычислительных устройствах. Предложен один из возможных подходов к решению проблемы с использованием технологии распараллеливания данных. Показано, что в общем случае проблема представляется нетривиальной математической задачей. Для одного из частных случаев предложен алгоритм решения, который можно, в том числе, использовать и для компьютеров стандартной конфигурации (с процессором и видеокартой).

Эффективность предложенного подхода демонстрируется на примере решения задачи сжатия цветных изображений с использо-

ванием нейронной сети прямого распространения. Задача решалась на компьютере с многоядерным процессором и видеокартой.

По результатам экспериментов можно сделать предположение, что выявленное соотношение по снижению вычислительных затрат (на гетерогенных устройствах) сохранится и при пропорциональном увеличении вычислительных мощностей процессора и видеокарты. Поэтому описанные в работе идеи могут оказаться полезными при обработке больших объемов данных на гетерогенных кластерных вычислителях, которые активно развиваются в настоящее время.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Чубукова, И. А. Data mining : учеб. пособие / И. А. Чубукова. – Москва : БИНОМ. Лаб. знаний, 2006. – 382с.
2. Воеводин, В. В. Параллельные вычисления [Текст]: книга / В. В. Воеводин – Питер: БХВ Петербург, 2004. – 608 с.
3. Gregory, R. Andrews Foundations of multithreaded, parallel and distributed programming [Text]: book – 688 p.
4. Ciegis, R. Parallel Scientific Computing and Optimization [Text]: book / R. Ciegis, D. Henti, B. Kagstrom, J. Zilinskas – New York : Springer-Verlag, 2009. – 274 p.
5. Storti, D. CUDA for Engineers: An Introduction to High-Performance Parallel Computing (1st Edition) [Text]: book / D. Storti, M. Yurtoglu – Addison-Wesley Professional, 2015. – 352 p.
6. Hamdy, A. Taha Operations Research: An Introduction (9th Edition) [Text]: book / A. Taha Hamdy – Pearson, 2010. – 832 p.
7. McCool, M. Structured Parallel Programming: Patterns for Efficient Computation (1st Edition) [Text]: book / M. McCool, Arch D. Robison, James Reinders – Morgan Kaufmann, 2012. – 432 p.
8. STL-10 dataset [Электронный ресурс]. – Режим доступа : academictorrents.com/details/a799a2845ac29a66c07cf74e2a2838b6c5698a6a – Дата доступа : 25.02.2018.
9. STL-10 dataset description [Электронный ресурс]. – Режим доступа : stanford.edu/~acoates//stl10/ – Дата доступа : 24.02.2018.
10. CUDA toolkit [Электронный ресурс]: – Режим доступа : developer.nvidia.com/cuda-downloads – Дата доступа : 23.02.2018.

Материал поступил в редакцию 02.02.2019

KRASNOPROSHIN V. V., MATSKEVICH V. V. Algorithm for Fast Image Compression on Heterogeneous Computing Devices

The paper deals with the problem of organizing efficient data processing on heterogeneous computing devices. A possible approach to solving the problem using the data parallelization technology is proposed. It is shown that, in the general case, the problem is a non-trivial mathematical problem. For one of the special cases, a solution algorithm is proposed. The effectiveness of the approach is demonstrated by the example of solving the problem of compressing color images using a direct distribution neural network.

The ideas described in the paper may be useful in processing large amounts of data on heterogeneous clusters.

УДК 519.95

Волчкова Г. П., Котов В. М.

ЗАДАЧА $\mathbf{Qm} \,||\, \mathbf{C}_{\max}$ С ОГРАНИЧЕНИЯМИ НА КОЛИЧЕСТВО РАБОТ, ВЫПОЛНЯЮЩИХСЯ НА КАЖДОМ ПРИБОРЕ

Одной из классических задач теории расписаний является задача $\mathbf{P} \,|\, \mathbf{m} \,|\, \mathbf{C}_{\max}$, которая состоит в распределении n независимых работ на m идентичных параллельных приборов. Каждая работа имеет определенное время выполнения и может быть выполнена на любом из m приборов. При этом требуется так распределить работы на приборы, чтобы время завершения выполнения последней работы было минимальным. Эта проблема *NP*-трудна уже в случае $\mathbf{P} \,|\, \mathbf{2} \,|\, \mathbf{C}_{\max}$ [1].

В [2] рассмотрена эта задача в следующей интерпретации. Предполагается, что на одном приборе может выполняться k работ, так как каждый прибор имеет k единиц специфического ресурса, и каждая работа требует единицу этого ресурса.

Показано, что эта задача может быть сформулирована как задача k-разбиения. В таком случае работы $\{i_1, i_2, \ldots, i_n\}$ идентифицируются с предметами, а процессоры с подмножествами $\{M_1, M_2, \ldots, M_m\}$. Тогда решение задачи теории расписаний соответствует разбиению предметов на подмножества. В [2] показано, что если C^* и $C^*\big\|$ – значения оптимальных решений для задачи k-разбиения и задачи $\mathbf{P} \,|\, \mathbf{m} \,|\, \mathbf{C}_{\max}$ соответственно, то $\dfrac{C^*}{C^*\big\|} < 2 - 1/m$. Предложены приближенные алгоритмы, решающие задачу k-разбиения.

В данной работе рассматривается задача с ограничениями на множестве параллельных приборов различной производительности $\mathbf{Q} \,|\, \mathbf{m} \,|\, \mathbf{C}_{\max}$, которая формулируется следующим образом.

Задано множество $N=\{i_1, i_2, \ldots, i_n\}$ из n-независимых работ, где работа i_j имеет время обработки p_j. Работы выполняются на множестве $M=\{j_1, j_2, \ldots, j_m\}$ из m приборов с различными скоростями s_j, $j = 1,\ldots,m$, при этом время выполнения i-й работы на j-м приборе определим как $t_{ij} = p_i / s_j$.

Дополнительным ограничением является то, что на каждом приборе выполняется не более k ($k \le n$) работ.

Обозначим через Z_j загрузку на j-м приборе:

$$Z_j = \sum_{i=1}^{k_j} p_i \Big/ s_j \,, \quad j = 1,\ldots,m \,.$$

Нижнюю границу значения оптимального решения можно оценить следующим образом:

$$LB = \max\left\{ \sum_{i=1}^{n} p_i \Big/ \sum_{j=1}^{m} s_j \,,\ p_1/s_1 \,,\ \min\{(p_1+p_2)/s_1,\ p_2/s_2\} \right\}. \quad (1)$$

Предлагаемый алгоритм A_1 основывается на идеях редуцирования задачи [3] и динамического пересчета нижних оценок оптимального решения [4, 5].

Волчкова Галина Петровна, *ст. преподаватель кафедры дискретной математики и алгоритмики ФПМИ Белорусского государственного университета, e-mail: Volchkovagp@mail.ru.*

Котов Владимир Михайлович, *заведующий кафедрой дискретной математики и алгоритмики ФПМИ Белорусского государственного университета, e-mail: kotovvm@bsu.by.*

Республика Беларусь, 220050, г. Минск, пр. Независимости, 4.