ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ОПЕРАЦИИ ПЕРЕСЕЧЕНИЯ МНОЖЕСТВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ АНАЛИЗА ФОРМАЛЬНЫХ ПОНЯТИЙ

С. В. Синцов
Кафедра интеллектуальных информационных технологий, Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: ssivikt@gmail.com

В данном докладе рассматривается алгоритм операции пересечения мультимножеств, выполненный как часть решения задачи анализа формальных понятий и допускающий реализацию на параллельной вычислительной архитектуре за время $O(n/p * log(n))$ от суммарой мощности множеств n и количества процессоров p. Приводятся результаты тестирования реализации алгоритма операции пересечения, выполненной средствами платформы OpenCL.

Введение

Наличие данных большого объёма в информационных системах требует эффективных механизмов их обработки, нередко основывающихся на использовании моделей и методов искусственного интеллекта. В связи с этим в работе [3] были рассмотрены подход и алгоритмы, способные обеспечить автоматизацию процесса построения онтологических структур и поддержку создания компонентов [2] при проектировании баз знаний, использующих для представления информации унифицированные семантические сети с теоретико-множественной интерпретацией [1]. В основе рассмотренного подхода лежит один из методов анализа данных – анализ формальных понятий (АФП).

Данная работа продолжает [3], описывая результаты и особенности параллельной реализации алгоритма операции пересечения мультимножеств с использованием средств платформы OpenCL. Актуальность работы вызвана как минимум двумя причинами: 1) ключевой операцией для большинства алгоритмов АФП является операция пересечения множеств [5]; 2) возможностью уменьшения временных затрат на построение концептуальных решёток при использовании параллельного исполнения некоторых частей алгоритма АФП на отдельных устройствах гетерогенной вычислительной архитектуры.

Известны следующие работы, рассматривающие алгоритмы базовых операций над множествами и, в частности, операцию пересечения: [6], [7], [8] и др. Работа [6] предлагает набор базовых алгоритмов операций над мультимножествами, однако их реализация выполнена с использованием средств платформы CUDA. В работе [7] рассматривается операция пересечения, применимая лишь ко множествам без кратных элементов. Кроме того, реализация описанного алгоритма так же осуществлена с использованием средст платформы CUDA. В работе [8] описывается математическая модель параллельных операций над множествами без приведения результатов их практической реализации. Обширный список литературы приведён в работе [7].

Особенность предлагаемого в данной работе алгоритма заключается в том, что он оперирует мультимножествами, представленными в виде отсортированных массивов целых беззнаковых 32х битных, в общем случае повторяющихся, чисел и, в отличие от приведённых выше работ, реализован средствами платформы OpenCL.

Алгоритм и результаты тестирования реализации операции пересечения

Входными данными алгоритма параллельной операции пересечения $IntersectParallel(\langle A, B\rangle)$ являются упорядоченные массивы A и B такие, что их длины связаны отношением $Length(A) \leq Length(B)$. Приведём пошаговое описание этого алгоритма:

1. Выполнить поиск минимального и максимального индексов вхождения каждого элемента массива A в массив B: $\langle L, U\rangle \leftarrow RangeBinarySearch(\langle L, U\rangle)$.
2. Отметить в массиве C минимальные индексы равных элементов массива A:
$$C[i] = \begin{cases} i, A[max(\{0\} \cup \{i-1\})] < A[i] \\ 0, A[max(\{0\} \cup \{i-1\})] = A[i] \end{cases}.$$
3. Вычислить максимумы $D \leftarrow MaxScatter(C)$:
$k \leftarrow ((\sim 0) >> (clz(Length(A)) + 1)) + 1$
пока $(k > 0)$:
если $(i \geq k), mo$
$D[i] = max(\{D[i]\} \cup \{D[i-k]\})$
$k >>= 1$
4. Вычислить:
$$E[i] = \begin{cases} 1, i - D[i] < U[i] - L[i] \\ 0, i - D[i] \geq U[i] - L[i] \end{cases}.$$
5. Вычислить массив F префиксных сумм массива E: $F[i] \leftarrow E[i] + PrefixSum(E)[i]$.
6. Вычислить:
если $(E[i] = 1), mo$ $R[F[i]-1] \leftarrow A[i]$.
7. Возвратить: $\langle R, F[Length(A)-1]\rangle$.

Библиотека БГУИР

Алгоритм $IntersectParallel$ удалось реализовать так, что его пространственная сложность равна $\Theta(Length(A))$. Теоретическая же оценка временной сложности алгоритма $IntersectParallel$ без учёта временных затрат на управление памятью не превосходит $O(n/p * log(n))$, где $n = Length(A)+Length(B)$ – суммарная длина массивов, p – количество процессоров.

Для оценки выигрыша (проигрыша) в производительности при использовании параллельной реализации алгоритма $IntersectParallel$ был реализован последовательный алгоритм $IntersectSerially$, в основе которого лежит слияние упорядоченных списков [4], имеющий пространственную сложность $O(Length(A))$ и выполненный с определёнными оптимизациями так, что его временная сложность оценивается как $O(n)$ в худшем случае и $O(1)$ – в лучшем.

Для тестирования использовались вычислительные архитектуры: CPU (Intel Core i7 3520M CPU) и GPU (Intel HD Graphics 4000). OpenCL гранулы, соответствующие реализации $IntersectParallel$, исполнялись только устройством GPU, одной рабочей группой и максимально возможным количеством рабочих элементов. Реализация алгоритма $IntersectSerially$ исполнялась 1) в виде OpenCL гранулы устройством GPU, одной рабочей группой и одним рабочим элементом, 2) устройством CPU в виде программы, написанной на языке C++. Сравнение реализаций $IntersectParallel$ и $IntersectSerially$ проводилось для входных массивов A и B, состоящих из следующих элементов: 1) каждый $A[i]$ и $B[j]$ – есть случайная величина, равномерно распределённая на $[0, max(Length(A), Length(B))]$; 2) $A = \{1, 3, 5, ...\}, B = \{0, 2, 4, ...\}$; 3) A и B не содержат кратных элементов и любой $A[i]$ больше любого $B[j]$; 4) $A = \{0, 1, 4, 5, 8, 9, ...\}, B = \{2, 3, 6, 7, 10, 11, ...\}$; 5) $A = \{1, 2, 3, ...\}, B = \{1, 2, 3, ...\}$; 6) $A = \{0, 0, 0, ...\}, B = \{0, 0, 0, ...\}$ (см. таблицу 1). При этом $Length(A) = 1..2^{23}, Length(B) = 2^{23}$.

Заключение

Таким образом, был реализован алгоритм $IntersectParallel$ операции пересечения мультимножеств, код программы которого допускает параллельное исполнение на различных устройствах гетерогенной вычислительной архитектуры. При этом пропускная способность соответствующей операции пересечения оказывается гораздо ниже пропускной способности операции пересечения, выполненной последовательным алгоритмом на устройсте CPU. Возможные причины этого: различие в архитектуре устройств CPU и GPU; большее число операций записи(чтения) в(из) память(и), совершаемое алгоритмом $IntersectParallel$ и требующее перекрёстных обращений к памяти, что в свою очередь делает алгоритм $IntersectParallel$ неэффективным для некоторых типов входных множеств (см. таблицу 1): например, для типа 3 операция пересечения не совершает полезной работы, однако расходует много времени. Однако в лучшем случае он опережает алгоритм $IntersectSerially$ (GPU) в десятки раз, хотя и не достигает значений идеальной теоретической временной оценки. Алгоритм $IntersectParallel$, а также его реализация, требуют доработки. Так, например, возможен запуск OpenCL гранул большим числом рабочих групп.

1. Голенков, В. В. Графодинамические модели параллельной обработки знаний: принципы построения, реализации и проектирования / В. В. Голенков, Н. А. Гулякина // OSTIS-2012. – Минск: БГУИР, 2012. С. 23–52.
2. Голенков, В. В. Семантическая технология компонентного проектирования систем, управляемых знаниями / В. В. Голенков, Н. А. Гулякина // OSTIS-2015. – Минск: БГУИР, 2015. С. 57–78.
3. Ивашенко, В. П. Алгоритмы параллельной реализации анализа формальных понятий для приближённых множеств в однородных семантических сетях / В. П. Ивашенко, С. В. Синцов // конф. «BIG DATA and Advanced Analytics. BIG DATA и анализ высокого уровня». – Минск: БГУИР, 2016. С. 110–118.
4. Новиков Ф. Дискретная математика для программистов / Ф. Новиков. – СПб.: Питер, 2009. – 384 с.
5. Kuznetsov S. Comparing Performance of Algorithms for Generating Concept Lattices / S. Kuznetsov, S. Obiedkov // 14, J. Exp. Theor. Artif. Intell. – 2002. – P. 189–216.
6. Multisets [Электронный ресурс] / Baxter S. NVIDIA Research. – 2013. – Режим доступа: https://nvlabs.github.io/moderngpu/sets.html. Дата доступа: 11.09.2016.
7. Di Wu. Efficient Lists Intersection by CPU-GPU Cooperative Computing / Di Wu, F. Zhang, Naiyong Ao, et al. // IPDPS. – 2010. – P. 1–8.
8. Guan X. Time-Space Optimal Parallel Set Operations / X.Guan, M.A. Langston // PARBASE-90. – 1990. – P. 155–157.

Таблица 1 – Результаты тестирования реализации алгоритмов $IntersectParallel$ и $IntersectSerially$

Тип входных A и B	Диапазон средних значений ускорения, при использовании реализации $IntersectParallel$	
	$IntersectSerially$(CPU)/$IntersectParallel$	$IntersectSerially$(GPU)/$IntersectParallel$
1	(0.00041, 1.04197)	(0.97823, 43.6212)
2	(0.00014, 0.05322)	(0.76352, 17.5089)
3	(0.00014, 0.00026)	(0.75270, 0.86065)
4	(0.00017, 0.12327)	(0.75642, 25.2153)
5	(0.00016, 0.03309)	(0.85701, 6.32933)
6	(0.00014, 0.03228)	(0.75261, 5.78407)

Библиотека БГУИР