Оценка локальности параллельных алгоритмов, реализуемых на графических процессорах*

Н.А. Лиходед, М.А. Полещук

Белорусский государственный университет

Исследуется задача получения блоков операций и потоков операций параллельного алгоритма, приводящих к меньшему числу обращений к глобальной памяти и к эффективному использованию параллельными потоками вычислений кэшей и разделяемой памяти графического процессора. Сформулированы и доказаны утверждения, позволяющие оценить объем коммуникационных операций, порождаемых альтернативными вариантами задания размеров блоков вычислений, а также минимизировать число промахов кэша за счет использования временной и пространственной локальности данных с учетом размера и длины строк кэша. Исследования конструктивны и допускают программную реализацию для практического использования.

Ключевые слова: параллельные вычисления, графический процессор, минимизация объема коммуникационных операций, временная локальность, пространственная локальность.

1. Введение

Время решения задачи на современном компьютере во многом определяется локальностью реализуемого алгоритма – степенью использования памяти с быстрым доступом. В этой работе в качестве компьютера, на котором требуется реализовать параллельную версию алгоритма, будем рассматривать графические процессоры (GPU) – мощные многоядерные вычислительные устройства. При вычислениях на GPU быстрым является процесс обращения к разделяемой памяти мультипроцессора и к кэшам, но не обращение к глобальной памяти GPU.

Графический процессор выполняет множество параллельных потоков вычислений. Потоки объединяются в блоки, а блоки объединяются в сетку. Синхронизация между потоками разных блоков не предусмотрена. Каждый блок потоков выполняется атомарно на одном из мультипроцессоров графического процессора. Должны быть указаны блоки, которые могут выполняться мультипроцессорами одновременно и независимо друг от друга.

Целью этой работы является разработка метода, позволяющего получать блоки вычислений с меньшим числом обращений к глобальной памяти. Используется анализ информационных зависимостей, порождающих коммуникационные операции. Сформулированы и доказаны утверждения, позволяющие ранжировать параметры размера блоков на основе асимптотических оценок объема коммуникационных операций. В работе [1] подобные оценки получены с использованием более сложного (но и более приспособленного для автоматизации) математического аппарата. Известен способ получения более точной оценки количества элементов, к которым осуществляется доступ при выполнении операций блока вычислений, но практическое использование известного результата довольно трудоемкое и требует привлечения специализированных средств автоматизации [2, 3].

В работе также указаны условия наличия временной локальности и условия наличия пространственной локальности данных. Существование локальности позволяет эффективно использовать параллельными потоками вычислений кэши и разделяемую память. Исследование проводится аналитическими методами. Другой способ исследования локальности данных – построение профиля работы приложения с памятью [4].

* Работа выполнена в рамках государственной программы научных исследований Республики Беларусь «Конвергенция-2020», подпрограмма «Методы математического моделирования сложных систем».

2. Предварительные сведения

Будем считать, что алгоритм задан последовательной программой линейного класса[1] [5]. Основную вычислительную часть такой программы составляют циклические конструкции; границы изменения параметров циклов задаются неоднородными формами, линейными по совокупности параметров циклов и внешних переменных. Предполагается, что в гнезде циклов имеется K выполняемых операторов S_β и используется L массивов a_l размерностей ν_l. Область изменения параметров циклов (область итераций) для оператора S_β и размерность этой области обозначим соответственно V_β и n_β.

Вхождением (l,β,q) будем называть q-е вхождение массива a_l в оператор S_β. Индексы элементов l-го массива, связанных с вхождением (l,β,q), выражаются функцией $\overline{F}_{l,\beta,q}$ вида

$$\overline{F}_{l,\beta,q}(J) = F_{l,\beta,q}J + G_{l,\beta,q}N + f^{l,\beta,q},$$

$$J(j_1,...,j_{n_\beta}) \in V_\beta,\ N \in Z^s,\ F_{l,\beta,q} \in Z^{\nu_l \times n_\beta},\ G_{l,\beta,q} \in Z^{\nu_l \times s},\ f^{l,\beta,q} \in Z^{\nu_l},$$

где $N \in Z^s$ – вектор внешних переменных алгоритма, s – число внешних переменных.

Выполнение оператора S_β при конкретных значениях β и вектора параметров цикла J будем называть операцией и обозначать $S_\beta(J)$. Пара вхождений $(l,\alpha,1)$ и (l,β,q) порождает истинную зависимость $S_\alpha(I) \rightarrow S_\beta(J)$, если: $S_\alpha(I)$ выполняется раньше $S_\beta(J)$; $S_\alpha(I)$ переопределяет (изменяет) элемент массива a_l, а $S_\beta(J)$ использует на вхождении (l,β,q) в качестве аргумента тот же элемент массива; между операциями $S_\alpha(I)$ и $S_\beta(J)$ этот элемент не переопределяется.

Зависимости (информационные связи) $S_\alpha(I) \rightarrow S_\beta(J)$ между операциями будем характеризовать векторами $d^{\alpha,\beta}=J-I$. Если размерности векторов J и I не совпадают, то $J-I$ определяется как разность векторов меньшей из размерностей. Векторы $d^{\alpha,\beta}$ часто называют векторами зависимостей: $d^{\alpha,\beta}$ позволяет для операции $S_\beta(J)$ найти операцию $S_\alpha(I)$, от которой $S_\beta(J)$ зависит.

Пусть в гнезде циклов имеется Θ наборов выполняемых операторов. Под набором операторов будем понимать один или несколько операторов, окруженных одним и тем же множеством циклов. Операторы и наборы операторов линейно упорядочены расположением их в записи алгоритма. Обозначим: V^ϑ, $1 \le \vartheta \le \Theta$, – области изменения параметров циклов, окружающих наборы операторов, n^ϑ – размерность области V^ϑ, число циклов, окружающих ϑ-й набор операторов. Будем считать, что если оператор S_β принадлежит набору операторов с номером ϑ^β, то $n^\vartheta = n_\beta$.

Выделим блоки вычислений путем разбиения циклов, окружающих операторы. Пусть $m_\zeta^\vartheta = \min\limits_{J(j_1,j_2,..,j_{n^\vartheta}) \in V^\vartheta} j_\zeta$, $M_\zeta^\vartheta = \max\limits_{J(j_1,j_2,..,j_{n^\vartheta}) \in V^\vartheta} j_\zeta$, $1 \le \zeta \le n^\vartheta$, – предельные значения изменения параметров циклов. Зададим размеры блоков вычислений натуральными числами $r_1^\vartheta,...,r_{n^\vartheta}^\vartheta$. Параметр r_ζ^ϑ обозначает число значений параметра j_ζ, приходящихся на один блок ϑ-го набора операторов. Число частей Q_ζ^ϑ, на которые при формировании блоков разбивается область значений параметра j_ζ цикла, окружающего ϑ-й набор операторов, находится согласно $Q_\zeta^\vartheta = \left\lceil (M_\zeta^\vartheta - m_\zeta^\vartheta + 1)/r_\zeta^\vartheta \right\rceil$ ($\lceil \cdot \rceil$ обозначает ближайшее «сверху» целое число). Нумеровать блоки вычислений будем по каждой координате в пределах от 0 до $Q_\zeta^\vartheta - 1$, $1 \le \zeta \le n^\vartheta$.

Формализованным способом разбить множество операций алгоритма на блоки и потоки вычислений можно путем применения тайлинга – преобразования алгоритма для получения макроопераций [6, 7].

---

[1] В литературе используются также термины «аффинные гнезда циклов», «алгоритмы с аффинными зависимостями».

Пример 1. Рассмотрим алгоритм Флойда-Уоршелла поиска кратчайших путей между всеми парами вершин графа (рис. 1).

```
do k = 1, n
  do i = 1, n
    do j = 1, n
      a(i,j) = min(a(i,j), a(i,k) + a(k,j))
    enddo
  enddo
enddo
```

Рис. 2. Основная часть алгоритма Флойда-Уоршелла

В гнезде циклов имеется один выполняемый оператор S_1 (один набор операторов) и используется один массив a размерности 2. Область изменения параметров циклов (область итераций) $V_1=V^1=\{(k,i,j)\in Z^3 | \ 1\le k\le n,\ 1\le i\le n,\ 1\le j\le n\}$ для оператора S_1 имеет размерность 3. Для матриц $F_{l,\beta,q}$ на вхождениях (l,β,q) имеем:

$$F_{1,1,1}=F_{1,1,2}=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix},\ F_{1,1,3}=\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\end{pmatrix},\ F_{1,1,4}=\begin{pmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}.$$

В рассматриваемом примере векторы зависимостей $d^{\alpha,\beta}$ будем для наглядности помечать не номерами операторов, а элементами массивов, фигурирующими на порождающих зависимости вхождениях. Например, вектор $d^{a(i,j),a(i,k)}$ порождается зависимостью между данными $a(i,j)$ на вхождении $(l,\alpha,1)=(1,1,1)$ в левой части оператора S_1, и данными $a(i,k)$ на вхождении $(l,\beta,q)=(1,1,3)$ в правой части оператора. Укажем итерации, порождающие зависимости, и векторы зависимостей:

- $S_1(k-1,i,j)\to S_1(k,i,j)$: данное $a(i,j)$, вычисленное на итерации $(k-1,i,j)$, является аргументом $a(i,j)$ для вычислений на итерации (k,i,j); $d^{a(i,j),a(i,j)}=(1,0,0)$;
- $S_1(k-1,i,k)\to S_1(k,i,j)$: $a(i,k)$, вычисленное на итерации $(k-1,i,k)$, является аргументом для вычислений на итерации (k,i,j); $d^{a(i,j),a(i,k)}=(1,0,j-k)$;
- $S_1(k-1,k,j)\to S_1(k,i,j)$: $a(k,j)$, вычисленное на итерации $(k-1,k,j)$, является аргументом для вычислений на итерации (k,i,j); $d^{a(i,j),a(k,j)}=(1,i-k,0)$.

Отметим, что для фиксированного k все операции не зависят друг от друга. Кроме того, непосредственно из записи алгоритма следует, что данные $a(i,j)$ не обновляются, если $i=k$ или $j=k$.

Выделим блоки вычислений. Напомним, блоки вычислений должны выполняться атомарно, независимо друг от друга. С учетом зависимостей алгоритма, такие блоки проще всего задать, если положить $r_1=1$, r_2 и r_3 – параметры. Всего будет $Q_1\times Q_2\times Q_3$ двумерных (2D) блоков, где $Q_1=n$, $Q_2=\left\lceil \frac{n}{r_2}\right\rceil$, $Q_3=\left\lceil \frac{n}{r_3}\right\rceil$. Блочный алгоритм имеет следующий вид:

```
do k = 1, n
  do ibl = 0, Q2 - 1
    do jbl = 0, Q3 - 1
      do i = 1 + ibl*r2, min((ibl + 1)*r2, n)
        do j = 1 + jbl*r3, min((jbl + 1)*r3, n)
          a(i,j) = min(a(i,j), a(i,k) + a(k,j))
        enddo
      enddo
    enddo
  enddo
enddo
```

Рис. 2. Основная часть алгоритма Флойда-Уоршелла с выделенными 2D блоками

3. Оценка объема коммуникационных операций

3.1 Выражения для оценки объема операций чтения и записи

Обозначим $M^{\vartheta} = \max_{\zeta}\left(M_{\zeta}^{\vartheta} - m_{\zeta}^{\vartheta}\right)+1$ – наибольшее число итераций циклов, участвующих в получении блоков. Для простоты записи будем использовать обозначение M без индекса ϑ и подразумевать набор операторов ϑ^{β} при упоминании оператора S_{β}. Отметим, что величина $Q_{\zeta}^{\vartheta} r_{\zeta}^{\vartheta}$ имеет порядок M, поэтому r_{ζ}^{ϑ}, Q_{ζ}^{ϑ} могут быть величинами порядка M.

Определим термин «фиксированное данное массива» как конкретное, неизмененное содержимое соответствующей ячейки памяти. Введем в рассмотрение величины $\rho_{l,\beta,q}^{d}$, $\rho_{l,\beta,q}^{d,\zeta}$, которые определим следующим образом: пусть число фиксированных данных, используемых на вхождении (l,β,q), оценивается величиной $O(M^{\rho_{l,\beta,q}^{d}})$, а число фиксированных данных, используемых на вхождении (l,β,q) при фиксированном значении цикла с параметром j_{ζ} оценивается величиной $O(M^{\rho_{l,\beta,q}^{d,\zeta}-1})$. Наличие индекса d в обозначениях подчеркивает, что если вхождение (l,β,q) порождает истинную зависимость $S_{\alpha}(I) \rightarrow S_{\beta}(J)$, то $\rho_{l,\beta,q}^{d}$ и $\rho_{l,\beta,q}^{d,\zeta}$ зависят от вектора $d^{\alpha,\beta}$. Найти $\rho_{l,\beta,q}^{d}$ и $\rho_{l,\beta,q}^{d,\zeta}$ не представляет труда, если детально понимать реализуемый алгоритм. Формализованный способ получения величин $\rho_{l,\beta,q}^{d}$, $\rho_{l,\beta,q}^{d,\zeta}$ изложен в работе [1]. Этот способ использует функции $\overline{F}_{l,\beta,q}$, а также функции, описывающие информационную структуру алгоритма – покрывающие функции графа алгоритма [5] (называемые еще h-преобразованиями [8]).

Отметим, что число фиксированных данных, используемых (в пространстве размерности n_{β}–1) на вхождении при фиксированном значении цикла с параметром j_{ζ}, по порядку либо равно ($\rho_{l,\beta,q}^{d,\zeta}$–1=$\rho_{l,\beta,q}^{d}$), либо на 1 меньше ($\rho_{l,\beta,q}^{d,\zeta}$–1=$\rho_{l,\beta,q}^{d}$–1) числа фиксированных данных, используемых (в пространстве размерности n_{β}) при всех значениях цикла с параметром j_{ζ}.

Обозначим через $d_{\zeta}^{\alpha,\beta,\max}$ максимальные по модулю значения компонент векторов $d^{\alpha,\beta}$. Сделаем естественное для практики предположение: компоненты векторов $d^{\alpha,\beta}$ по модулю или не превосходят нескольких единиц, или неограниченно возрастают с ростом размера задачи. В первом случае $d_{\zeta}^{\alpha,\beta,\max}$ также не превосходят нескольких единиц, во втором случае являются большими, значительно превосходящими r_{ζ}^{ϑ}.

Теорема 1. Пусть вхождение (l,β,q) порождает истинную зависимость: определение некоторого данного происходит на вхождении (l,α,1) в левой части оператора S_{α}, а использование – на вхождении (l,β,q) в правой части оператора S_{β}, причем в окружении операторов S_{α} и S_{β} имеется цикл с параметром j_{ζ}.

Тогда, при получении блоков вычислений для реализации алгоритма на графическом процессоре, разбиение итераций цикла j_{ζ} порождает коммуникационные операции чтения и записи, объем которых имеет следующие оценки:

- $O(Q_{\zeta}^{\vartheta} M^{\rho_{l,\beta,q}^{d}-1})$ операций чтения и операций записи, если $0 < d_{\zeta}^{\alpha,\beta,\max} < r_{\zeta}^{\vartheta}$;
- $O(Q_{\zeta}^{\vartheta} M^{\rho_{l,\beta,q}^{d}})$ операций чтения и $O(M^{\rho_{l,\beta,q}^{d}})$ операций записи, если $d_{\zeta}^{\alpha,\beta,\max} \geq r_{\zeta}^{\vartheta}$, $\rho_{l,\beta,q}^{d,\zeta} \neq \rho_{l,\beta,q}^{d}$;
- $O(M^{\rho_{l,\beta,q}^{d}})$ операций чтения и операций записи, если $d_{\zeta}^{\alpha,\beta,\max} \geq r_{\zeta}^{\vartheta}$, $\rho_{l,\beta,q}^{d,\zeta} = \rho_{l,\beta,q}^{d}$;
- не требуется ни операций чтения, ни операций записи, если $d_{\zeta}^{\alpha,\beta,\max}$=0.

В случае, когда вхождение (l,β,q) не порождает истинной зависимости (происходит обращение к входным данным) или цикл с параметром j_ζ имеется в окружении только оператора S_β, оценки следующие:

- $O(Q_\zeta^\vartheta M^{\rho^d_{l,\beta,q}})$ операций чтения, если $\rho^{d,\zeta}_{l,\beta,q} \neq \rho^d_{l,\beta,q}$;
- $O(M^{\rho^d_{l,\beta,q}})$ операций чтения, если $\rho^{d,\zeta}_{l,\beta,q} = \rho^d_{l,\beta,q}$.

Доказательство. Утверждения теоремы оценивают объем коммуникационных операций, порождаемых разбиением итераций цикла j_ζ, в зависимости от значений ζ-й координаты вектора $d^{\alpha,\beta}$ и величин $\rho^{d,\zeta}_{l,\beta,q}$, $\rho^d_{l,\beta,q}$. Напомним, множество итераций цикла с параметром j_ζ разбивается на Q_ζ^ϑ частей, каждая часть имеет некоторый номер j_ζ^{gl} $(0 \le j_\zeta^{gl} \le Q_\zeta^\vartheta - 1)$ и содержит r_ζ^ϑ итераций (кроме, возможно, части с номером $Q_\zeta^\vartheta - 1$).

Пусть $0 < d_\zeta^{\alpha,\beta,\max} < r_\zeta^\vartheta$, определение данного происходит при выполнении операции $S_\alpha(I(i_1,...,i_{n_\alpha}))$, а использование – при выполнении операции $S_\beta(J(j_1,..., j_{n_\beta}))$.

Число фиксированных данных, используемых на вхождении (l,β,q) при фиксированном значении цикла с параметром j_ζ (т.е. в пространстве размерности n_β–1), оценивается величиной $O(M^{\rho^{d,\zeta}_{l,\beta,q}-1})$. Тогда число фиксированных данных, используемых на вхождении (l,β,q) при всех значениях цикла с параметром j_ζ (в пространстве размерности n_β), оценивается величиной $\frac{M}{d_\zeta^{\alpha,\beta,\max}} O(M^{\rho^{d,\zeta}_{l,\beta,q}-1}) = O(M^{\rho^{d,\zeta}_{l,\beta,q}})$ (напомним, величина $d_\zeta^{\alpha,\beta,\max}$ не превышает нескольких единиц). С другой стороны, оценка числа таких данных есть $O(M^{\rho^d_{l,\beta,q}})$. Таким образом, в рассматриваемом случае верно $\rho^{d,\zeta}_{l,\beta,q} = \rho^d_{l,\beta,q}$.

В каждой из Q_ζ^ϑ частей итераций цикла j_ζ число фиксированных данных, которые определяются при одном значении j_ζ^{gl}, но используются при другом, оцениваются величиной $d_\zeta^{\alpha,\beta,\max} O(M^{\rho^{d,\zeta}_{l,\beta,q}-1})$, причем независимо от значения r_ζ^ϑ величина $d_\zeta^{\alpha,\beta,\max}$, как было условлено, не превышает нескольких единиц. Суммарный объем коммуникационных операций чтения и операций записи по всем Q_ζ^ϑ частям определяется оценкой $Q_\zeta^\vartheta O(M^{\rho^{d,\zeta}_{l,\beta,q}-1}) = O(Q_\zeta^\vartheta M^{\rho^d_{l,\beta,q}-1})$.

Пусть $d_\zeta^{\alpha,\beta,\max} \ge r_\zeta^\vartheta$. Если $\rho^{d,\zeta}_{l,\beta,q} \neq \rho^d_{l,\beta,q}$ (т.е. $\rho^{d,\zeta}_{l,\beta,q} - 1 = \rho^d_{l,\beta,q}$), то число используемых на вхождении (l,β,q) фиксированных данных в каждой части и во всех Q_ζ^ϑ частях оценивается одинаковой величиной $O(M^{\rho^d_{l,\beta,q}})$. Этой величиной следует оценить объем коммуникационных операций записи. Оценка объема коммуникационных операций чтения по всем Q_ζ^ϑ частям есть $Q_\zeta^\vartheta O(M^{\rho^d_{l,\beta,q}}) = O(Q_\zeta^\vartheta M^{\rho^d_{l,\beta,q}})$. Если $\rho^{d,\zeta}_{l,\beta,q} = \rho^d_{l,\beta,q}$, то число используемых фиксированных данных в каждой из Q_ζ^ϑ частей оценивается величиной $O(r_\zeta^\vartheta M^{\rho^{d,\zeta}_{l,\beta,q}-1})$. Суммарный объем коммуникационных операций чтения и операций записи по всем Q_ζ^ϑ частям определяется оценкой $Q_\zeta^\vartheta O(r_\zeta^\vartheta M^{\rho^{d,\zeta}_{l,\beta,q}-1}) = O(Q_\zeta^\vartheta r_\zeta^\vartheta M^{\rho^d_{l,\beta,q}-1}) = O(M^{\rho^d_{l,\beta,q}})$.

Если $d_\zeta^{\alpha,\beta,\max} = 0$, то любое фиксированное данное определяется и используется при одном и том же значении параметра цикла j_ζ^{gl}. Коммуникационных операций не требуется.

Пусть вхождение (l,β,q) не порождает истинной зависимости (происходит обращение к входным данным) или порождает истинную зависимость, но цикл с параметром j_ζ имеется в

окружении только оператора S_β. Если $\rho^{d,\zeta}_{l,\beta,q} \neq \rho^{d}_{l,\beta,q}$ (т.е. $\rho^{d,\zeta}_{l,\beta,q} - 1 = \rho^{d}_{l,\beta,q}$), то объем коммуникационных операций (только чтение) по всем Q^{ϑ}_{ζ} частям оценивается величиной $Q^{\vartheta}_{\zeta} O(M^{\rho^{d}_{l,\beta,q}}) = O(Q^{\vartheta}_{\zeta} M^{\rho^{d}_{l,\beta,q}})$. Если $\rho^{d,\zeta}_{l,\beta,q} = \rho^{d}_{l,\beta,q}$, то объем коммуникационных операций (только чтение) по всем Q^{ϑ}_{ζ} частям определяется оценкой $Q^{\vartheta}_{\zeta} O(r^{\vartheta}_{\zeta} M^{\rho^{d,\zeta}_{l,\beta,q}-1}) = O(M^{\rho^{d}_{l,\beta,q}})$. □

Доказательство теоремы для случая, когда информационная структура алгоритма представлена покрывающими функциями графа алгоритма, приведено в работе [1].

Пример 1 (продолжение). Напомним векторы зависимостей, порождаемые вхождениями массива *a* в правую часть оператора, и укажем оценки числа фиксированных данных, используемых на вхождениях.

Для второго вхождения $a(i,j)$ (использование прежнего значения обновляемого элемента) $d^{a(i,j),a(i,j)}$ =(1,0,0); $\rho^{d}_{1,1,2}$ =3 – число фиксированных данных, используемых на вхождении, оценивается величиной $O(n^3)$; $\rho^{d,1}_{1,1,2} = \rho^{d,2}_{1,1,2} = \rho^{d,3}_{1,1,2}$ =3 – число фиксированных данных, используемых на вхождении при фиксированном значении одного из циклов с параметрами $j_1=i$, $j_2=j$, $j_3=k$, оценивается величинами $O(M^{\rho^{d,\zeta}_{1,1,2}-1}) = O(n^2)$.

Для третьего вхождения $a(i,k)$ (использование при фиксированном k на итерациях (i,j,k) элементов одного столбца) $d^{a(i,j),a(i,k)}$ =(1,0,j–k), $\rho^{d}_{1,1,3}$ =2, $\rho^{d,1}_{1,1,3}$ =2, $\rho^{d,2}_{1,1,3}$ =2, $\rho^{d,3}_{1,1,3}$ =3.

Для четвертого вхождения $a(k,j)$ (использование при фиксированном k на итерациях (i,j,k) элементов одной строки) $d^{a(i,j),a(k,j)}$ =(1,i–k,0), $\rho^{d}_{1,1,4}$ =2, $\rho^{d,1}_{1,1,4}$ =2, $\rho^{d,2}_{1,1,4}$ =3, $\rho^{d,3}_{1,1,4}$ =2.

Оценим, используя теорему 1, объем коммуникационных операций, порождаемых разбиением итераций циклов.

Пусть ζ=1. Тогда для всех вхождений $d^{\alpha,\beta,\max}_{\zeta} = d^{\alpha,\beta}_{\zeta}$ =1. Имеет место случай $0 < d^{\alpha,\beta,\max}_{\zeta} < r^{\vartheta}_{\zeta}$, требуется $O(Q^{\vartheta}_{\zeta} M^{\rho^{d}_{l,\beta,q}-1})$ операций чтения и операций записи: для второго вхождения $O(Q_1 n^2)$ операций, для третьего и четвертого вхождений – $O(Q_1 n)$ операций.

Пусть ζ=2. Для второго и третьего вхождений $d^{\alpha,\beta,\max}_{\zeta} = d^{\alpha,\beta}_{\zeta}$ =0, поэтому не требуется ни операций чтения, ни операций записи. Для четвертого вхождения (случай $d^{\alpha,\beta,\max}_{\zeta} \geq r^{\vartheta}_{\zeta}$, $\rho^{d,\zeta}_{l,\beta,q} \neq \rho^{d}_{l,\beta,q}$, $(l,\beta,q)=(1,1,4)$) требуется $O(Q^{\vartheta}_{\zeta} M^{\rho^{d}_{l,\beta,q}}) = O(Q_2 n^2)$ операций чтения и $O(M^{\rho^{d}_{l,\beta,q}}) = O(n^2)$ операций записи.

Пусть ζ=3. Для второго и четвертого вхождений $d^{\alpha,\beta,\max}_{\zeta} = d^{\alpha,\beta}_{\zeta}$ =0, не требуется ни операций чтения, ни операций записи. Для третьего вхождения (случай $d^{\alpha,\beta,\max}_{\zeta} \geq r^{\vartheta}_{\zeta}$, $\rho^{d,\zeta}_{l,\beta,q} \neq \rho^{d}_{l,\beta,q}$, $(l,\beta,q)=(1,1,3)$) требуется $O(Q_3 n^2)$ операций чтения и $O(n^2)$ операций записи.

3.2 Ранжирование размеров блоков вычислений

Утверждения теоремы 1 позволяют найти асимптотику суммарного объема коммуникационных операций, порождаемых разбиением множества итераций j_ζ на Q^{ϑ}_{ζ} частей, и, следовательно, дают возможность ранжировать (устанавливать соотношение размеров относительно друг друга) параметры размера блоков вычислений для минимизации объема коммуникационных операций. При ранжировании параметров размера блоков вычислений параллельного алгоритма, реализуемого на графическом процессоре, следует в оценках объема коммуникационных операций принимать во внимание только оценки, содержащие множитель Q^{ϑ}_{ζ}; следует также учитывать, что величины r^{ϑ}_{ζ} и Q^{ϑ}_{ζ} связаны обратно пропорциональной зависимостью.

Пример 1 (продолжение). Для ранжирования параметров r_1, r_2 и r_3 размера блоков вычислений, выполняемых на графическом процессоре, укажем, следуя результатам предыдущего раздела, оценки объема коммуникационных операций, содержащие множитель Q_ζ^ϑ – число частей, на которые при формировании блоков разбивается область значений параметра j_ζ.

Если ζ=1, то требуется $O(Q_1 n^2)$ операций чтения и операций записи для второго вхождения и $O(Q_1 n)$ операций чтения и операций записи для третьего и четвертого вхождений. Если ζ=2, то требуется $O(Q_2 n^2)$ операций чтения для четвертого вхождения. Если ζ=3, то требуется $O(Q_3 n^2)$ операций чтения для третьего вхождения.

Таким образом, сравнивая оценки коммуникационных издержек и по чтению и по записи, приходим к выводу, что параметр размера блока вычислений r_1 увеличивать более выгодно, чем параметры r_2 и r_3. В то же время, наиболее простой блочный алгоритм Флойда-Уоршелла (рис. 2) содержит 2D блоки, для которых r_1=1. Блочный алгоритм с 3D блоками (рис. 3), для которых $r_1=r_2=r_3=r>1$ получен в работе [9]. Представляет также интерес разработка блочного алгоритма с 3D блоками, для которых $r_1>r_2$, $r_1>r_3$.

```
do k = 1 + k^bl*r, min((k^bl + 1)*r, n)
  do i = 1 + i^bl*r, min((i^bl + 1)*r, n)
    do j = 1 + j^bl*r, min((j^bl + 1)*r, n)
      a(i,j) = min(a(i,j), a(i,k) + a(k,j))
    enddo
  enddo
enddo
```

Рис. 3. 3D блоки блочного алгоритма Флойда-Уоршелла

4. Условия наличия локальности данных

Потоки одного блока выполняются на мультипроцессоре частями-пулами, называемыми варп. Варп содержит до 32 потоков (два полуварпа по 16 потоков). Если несколько потоков одного полуварпа обращаются к одному и тому же банку разделяемой памяти, константной памяти или кэша текстурной памяти для доступа к различным данным, то происходит конфликт. Но конфликта не происходит, если несколько потоков одного полуварпа обращаются к одному и тому же данному (broadcast). В этом случае запрашиваемое данное передается только один раз, поэтому трафик сокращается в 16 раз (для архитектуры Fermi в 32 раза, так как объединение запросов в память происходит для 32 потоков).

Текстурная память может быть использована эффективно, если каждый поток полуварпа запрашивает близко расположенные в памяти данные, то есть если доступ к памяти характеризуется пространственной локальностью. Наличие пространственной локальности позволяет эффективно использовать кэш текстурной памяти и разделяемую память параллельными потоками.

Зададим в блоках вычислений потоки вычислений посредством выделения блоков второго уровня. Размеры блоков второго уровня зададим натуральными числами $r_{1,2}$, $r_{2,2}$, …, $r_{n_\beta,2}$. Параметр $r_{\xi,2}$ обозначает число значений параметра j_ξ, приходящихся на один поток (на один блок второго уровня). Обозначим через $Q_{\xi,2}$ число частей, на которые при формировании потоков разбивается область значений цикла с параметром j_ξ, приходящихся на один блок; $Q_{\xi,2}=\left\lceil r_\xi / r_{\xi,2} \right\rceil$. Выбор размеров $r_{\xi,2}$, оптимальных по числу кэш-промахов, определяется в значительной степени наличием временной локальности и пространственной локальности данных.

Введем обозначения:

$F_{l,\beta,q,\xi}$ – столбец с номером ξ матрицы $F_{l,\beta,q}$;

$\xi^{l,\beta,q}$ – номер ненулевого столбца матрицы $F_{l,\beta,q}$, правее которого находятся только нулевые столбцы матрицы; если самый правый столбец F_{l,β,q,n_β} не нулевой, то по определению $\xi^{l,\beta,q}=n_\beta$;

$\Xi^{l,\beta,q}$ – множество номеров линейно независимых столбцов матрицы $F_{l,\beta,q}$ (если таких множеств более одного, то зафиксировать любое);

$F^{\Xi}_{l,\beta,q}$ – матрица, составленная из столбцов матрицы $F_{l,\beta,q}$ с номерами из множества $\Xi^{l,\beta,q}$; матрица $F^{\Xi}_{l,\beta,q}$ имеет размеры $\nu_l \times |\Xi^{l,\beta,q}|$, где ν_l – размерность массива a_l, $|\Xi^{l,\beta,q}|$ – число элементов множества $\Xi^{l,\beta,q}$;

L_l – число элементов массива a_l, помещающихся в строку кэша; предполагается, что строка массива помещается в кэш.

Лемма. Пусть для вхождения (l,β,q) массива a_l в правую часть оператора S_β для всех $\xi \notin \Xi^{l,\beta,q}$, $\xi \le \xi^{l,\beta,q}$, выполнены условия $r_{\xi,2}=1$ (условие накладывается при наличии указанных ξ) и $\nu_l \ge |\Xi^{l,\beta,q}|$. Тогда в каждом потоке вычислений на разных итерациях циклов с параметрами j_ξ, $1 \le \xi \le \xi^{l,\beta,q}$, используются разные элементы массива a_l.

Доказательство. Предположим противное утверждению леммы: на двух итерациях потока вычислений с различными параметрами циклов j_ξ, $\xi \le \xi^{l,\beta,q}$, используется один и тот же элемент данных. Это означает $F_{l,\beta,q}J = F_{l,\beta,q}\hat{J}$ на итерациях J и $\hat{J}$ потока вычислений с различными параметрами циклов j_ξ, $\xi \in \Xi^{l,\beta,q}$; учтено, что $r_{\xi,2}=1$ для $\xi \notin \Xi^{l,\beta,q}$, $\xi \le \xi^{l,\beta,q}$.

Так как матрица $F^{\Xi}_{l,\beta,q}$ имеет $|\Xi^{l,\beta,q}|$ линейно независимых столбцов, то можно выделить столько же линейно независимых строк. Матрицу, составленную из выделенных строк, обозначим F_Ξ. Обозначим через J_Ξ и $\hat{J}_\Xi$ векторы размерности $|\Xi^{l,\beta,q}|$, построенные по векторам J и $\hat{J}$ выбором компонент с номерами из множества $\Xi^{l,\beta,q}$. Тогда векторы J_Ξ и $\hat{J}_\Xi$ различны и $F_{l,\beta,q}J - F_{l,\beta,q}\hat{J} = F^{\Xi}_{l,\beta,q} J_\Xi - F^{\Xi}_{l,\beta,q} \hat{J}_\Xi = 0$. Так как строки матрицы F_Ξ составлены из строк матрицы $F^{\Xi}_{l,\beta,q}$, то выполняется равенство $F_\Xi J_\Xi = F_\Xi \hat{J}_\Xi$. Из построения матрицы F_Ξ следует, что она является невырожденной. Умножая последнее равенство слева на $(F_\Xi)^{-1}$, получим $J_\Xi = \hat{J}_\Xi$ (противоречие). □

Отметим, что если $F_{l,\beta,q,\xi}=0$, $\xi < \xi^{l,\beta,q}$, и итерации цикла с параметром j_ξ выполняются параллельными потоками, то имеет место бродкаст.

Теорема 2. Пусть выполняются предположения леммы и не более чем один столбец $F_{l,\beta,q,\zeta}$ имеет вид $(0,\ldots,0,b_{l,\beta,q,\zeta})^T$, $0 < |b_{l,\beta,q,\zeta}| r_{\zeta,2} < L_l$. Если $r_{\zeta,2}=1$, цикл с параметром j_ζ выполняется параллельно, циклы с параметрами j_ξ, $\xi \notin \Xi^{l,\beta,q}$, $F_{l,\beta,q,\xi} \ne 0$, выполняются последовательно (с синхронизацией потоков вычислений между итерациями в случае $r_\xi > 1$), то независимо от выбора $r_{\xi,2}$, $\xi > \xi^{l,\beta,q}$, совокупное число кэш-промахов по всем потокам вычислений является наименьшим. Сформулированные условия накладываются при наличии указанных столбцов.

Доказательство. Сделаем естественное предположение, что строка массива a_l выровнена в памяти по размеру строки кэша; в этом случае обращения к одной строке массива не уменьшат число кэш-промахов при обращении к данным, расположенным в начале следующей строки.

Условие леммы $r_{\xi,2}=1$ для всех $\xi \notin \Xi^{l,\beta,q}$, $\xi \le \xi^{l,\beta,q}$, в случае $r_\xi > 1$, цикл с параметром j_ξ, $\xi \le \xi^{l,\beta,q}$, является последовательным и столбец $F_{l,\beta,q,\xi}$ линейно зависим со столбцами $F_{l,\beta,q,\xi}$, $\xi \in \Xi^{l,\beta,q}$, можно заменить условием синхронизации потоков между итерациями цикла с параметром j_ξ. Так как выполняются предположения леммы, то в каждом потоке вычислений разные элементы данных используются на разных итерациях циклов с параметрами j_ξ, $\xi \le \xi^{l,\beta,q}$. В каждом потоке вычислений на итерациях циклов с параметрами j_ξ, $\xi > \xi^{l,\beta,q}$ (если $\xi^{l,\beta,q} \ne n_\beta$), используется один и тот же элемент данных, так как столбцы $F_{l,\beta,q,\xi}$, $\xi > \xi^{l,\beta,q}$, являются нулевыми; итерации указанных циклов в потоке вычислений выполняются одна за другой. Следовательно, в каждом потоке на всех, кроме первой, итерациях, на которых используется элемент данных, он находится в кэше, то есть кэш-промахи, связанные с его использованием, отсутствуют. Если $F_{l,\beta,q,\xi}=0$, $\xi < \xi^{l,\beta,q}$, и итерации цикла с параметром j_ξ выполняются не одним, а параллельными потоками, то осуществляется бродкаст и, следовательно, совокупное по потокам вычислений число кэш-

промахов остается неизменным в полуварпе. Совокупное по потокам число кэш-промахов также не изменяется при произвольном выборе $r_{\xi,2}$, $\xi \in \Xi^{l,\beta,q}$, $\xi \neq \zeta$, так как разные итерации параллельно выполняемых циклов j_ξ имеют доступ к разным строкам массива. Кроме того, циклы по j_ξ, $\xi \notin \Xi^{l,\beta,q}$, $F_{l,\beta,q,\xi} \neq 0$, выполняются последовательно (по условию теоремы).

Рассмотрим цикл с параметром j_ζ (если имеется столбец указанного в формулировке теоремы вида). Использование элемента данных характеризуется пространственной локальностью относительно цикла с параметром j_ζ, так как используемые данные располагаются в одной строке массива на расстоянии $|b_{l,\beta,q,\zeta}|$ элементов. Число используемых строк кэша потоками равно $\left\lceil \frac{|b_{l,\beta,q,\zeta}| r_{\zeta,2} Q_{\zeta,2}}{L_l} \right\rceil$. Тогда в потоках вычислений имеется $Q_{\zeta,2} - \left\lceil \frac{|b_{l,\beta,q,\zeta}| r_{\zeta,2} Q_{\zeta,2}}{L_l} \right\rceil$ использований строк кэша без кэш-промахов при чтении элементов строки массива (с учетом того, что строка массива a_l помещается в кэш и выполнено условие $|b_{l,\beta,q,\zeta}| r_{\zeta,2} < L_l$). Их число наибольшее при наибольшем $Q_{\zeta,2}$, то есть при $r_{\zeta,2}=1$. Тогда совокупное по всем потокам вычислений число кэш-промахов, связанных с использованием элементов строки массива a_l, наименьшее (потребуется один обязательный кэш-промах для строк кэша). □

Теорема дает условия, при которых в потоках вычислений повторное использование элемента данных возникает только на последовательных итерациях алгоритма (следует из доказательства). Указывается выбор $r_{\xi,2}$, при котором число кэш-промахов наименьшее, а размеры $r_{\xi,2}$, для которых $\xi \in \Xi^{l,\beta,q}$ или $\xi > \xi^{l,\beta,q}$, могут быть заданы произвольно, так как число кэш-промахов на этом вхождении будет одинаково по всем потокам вычислений. Выполнение условий теоремы свидетельствует о наличии пространственной локальности на итерациях цикла с параметром j_ζ (в различных потоках вычислений) и о наличии временной локальности для осуществления бродкаста данного по параллельно выполняемым циклам с параметрами j_ξ, для которых $F_{l,\beta,q,\xi}=0$.

Пример 1 (продолжение). Рассмотрим блоки вычислений (рис. 3). Известно, что самый внешний цикл (цикл с параметром j_1, $j_1=k$) является последовательным, а внутренние циклы по $j_2=i$ и по $j_3=j$ могут выполняться параллельно. Имеем: $F_{1,1,2}=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$, $F_{1,1,3}=\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\end{pmatrix}$, $F_{1,1,4}=\begin{pmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}$, $\xi^{1,1,2}=3$, $\xi^{1,1,3}=2$, $\xi^{1,1,4}=3$. На вхождении (1,1,2) столбец $F_{1,1,2,1}$ является нулевым, поэтому, следуя теореме (предположения леммы справедливы и для теоремы) применительно к вхождению $(l,\beta,q)=(1,1,2)$, положим $r_{1,2}=1$ (или положим $r_{1,2}$ максимально возможным – размеру блока r_1 – и установим синхронизацию потоков между итерациями цикла по k). Далее на вхождении (1,1,2) имеем: $F_{1,1,2,2}=(1\ 0)^{\mathrm{T}}$, $\xi=2$, $\xi \in \Xi^{l,\beta,q}$, поэтому $r_{2,2}$ можно выбрать произвольно (но не больше максимально возможного размера блока); $F_{1,1,2,3}=(0\ 1)^{\mathrm{T}}$, поэтому выберем $r_{3,2}=1$. На вхождении (1,1,3) имеем $F_{1,1,3,1}=(0\ 1)^{\mathrm{T}}$, цикл по j_1 выполняется последовательно, поэтому выбор $r_{1,2}$ не определяется теоремой; $F_{1,1,3,2}=(1\ 0)^{\mathrm{T}}$, поэтому $r_{2,2}$ можно выбрать произвольно; так как $F_{1,1,3,3}=0$, $\xi=3>\xi^{1,1,3}$, то $r_{3,2}$ можно выбрать произвольно. На вхождении (1,1,4) имеем $F_{1,1,4,1}=(1\ 0)^{\mathrm{T}}$, поэтому $r_{1,2}$ можно выбрать произвольно; $F_{1,1,4,2}=0$, $\xi=2\leq\xi^{1,1,4}$, поэтому выберем $r_{2,2}=1$; $F_{1,1,4,3}=(0\ 1)^{\mathrm{T}}$, поэтому выберем $r_{3,2}=1$.

Таким образом, суммируя сведения по каждому вхождению, выберем размеры потоков вычислений следующим образом: $r_{1,2}=1$, $r_{2,2}=1$, $r_{3,2}=1$. Выбор размера потоков указанным образом определяет наименьшее в совокупности по потокам вычислений число кэш-промахов для каждого из вхождений (1,1,2), (1,1,3), (1,1,4) в отдельности. Бродкаст данных осуществляется на вхождении (1,1,3) для параллельного цикла по j_3 (так как $F_{1,1,3,3}=0$) и на вхождении (1,1,4) для параллельного цикла по j_2 (так как $F_{1,1,4,2}=0$). Использование данных на итерациях цикла по j_3

(в различных параллельных потоках вычислений) на вхождениях (1,1,2), (1,1,4) ($F_{1,1,2,3}$=$F_{1,1,4,3}$=$(0\ 1)^T$) характеризуется пространственной локальностью. Как уже отмечалось, можно положить $r_{1,2}$=r_1 и установить синхронизацию потоков после каждой итерации k. Если потоки одного блока не имеют общих данных, то синхронизация не требуется.

5. Вычислительные эксперименты

Пример 1 (продолжение). В работе [10] реализован алгоритм Флойда-Уоршелла с 3D блоками. Как показано в подразделе 3.2, блочный алгоритм с 3D блоками обладает улучшенной, по сравнению с 2D-блочной версией, локальностью; это позволило уменьшить время реализации в 5-6 раз. В работе [11] алгоритм Флойда-Уоршелла с 3D блоками модифицирован, в том числе и с целью использования пространственной локальности данных; время реализации уменьшилось еще в 5 раз.

Пример 2. Рассмотрим алгоритм прямого хода метода Гаусса решения систем линейных алгебраических уравнений с использованием расширенной матрицы (рис. 4).

```
do k = 1, n - 1
  do i = k + 1, n
    do j = k + 1, n + 1
      a(i,j) = a(i,j) - (a(i,k) / a(k,k)) * a(k,j)
    enddo
  enddo
enddo
```

Рис. 4. Основная часть алгоритма Гаусса (без выбора ведущего элемента)

Исследование локальности этого алгоритма практически совпадает с приведенным в разделе 3 исследованием локальности алгоритма Флойда-Уоршелла. Некоторое несущественное дополнение к приведенным выкладкам связано с наличием ведущего элемента $a(k,k)$.

В экспериментах использовалась видеокарта GeForce GT 860M со следующими характеристиками: объем глобальной памяти – 4295 МБ; максимальный размер разделяемой памяти в одном блоке – 49 КБ; количество 32-разрядных регистров в одном блоке – 65 536; количество потоков в варпе – 32; мультипроцессоров – 5; всего ядер – 640.

На рис. 5 продемонстрирована зависимость времени реализации алгоритма от r_1 – параметра размера блока, характеризующего число записей в глобальную память. Число записей результатов вычисления одного блока пропорционально $Q_1 = \left\lceil \frac{n-1}{r_1} \right\rceil$. Использование 3D блоков размером $r_1 \times r_2 \times r_3 = r_1 \times 64 \times 64$ с большим значением r_1 позволяет уменьшить затраты времени на доступ в глобальную память. В разделяемую память заносились только те элементы массива, которые использовались и для чтения, и для записи. Элементы, которые использовались только для чтения, читались из глобальной памяти. Пространственная локальность данных не использовалась.

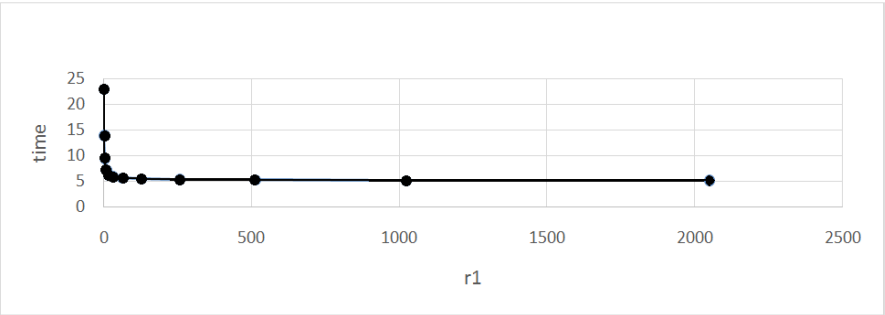

Рис. 5. Зависимость времени реализации блочного алгоритма Гаусса от r_1

Вычислительные эксперименты показали, что время реализации алгоритма уменьшается (до некоторого предела) с ростом r_1, т.е. уменьшается при уменьшении числа операций записи в глобальную память. Стабилизация времени связана с тем, что требуется определенное время на запуски ядра и на планирование потоков.

6. Заключение

Таким образом, в работе сформулированы и доказаны утверждения, позволяющие оценить объем коммуникационных операций, порождаемых разбиением множества итераций. Асимптотические оценки суммарного объема коммуникационных операций дают возможность ранжировать параметры размера блоков для минимизации объема коммуникационных операций. В работе также получены условия, при выполнении которых достигается наименьшее число кэш-промахов в потоках вычислений при наличии локальности данных.

Направления дальнейших исследований: разработка и программная реализация алгоритмов, позволяющих выбирать соотношения размеров блоков вычислений; получение условий и соотношений, точно характеризующих объем коммуникационных операций; обобщения условий наличия локальности данных, пригодные для применения в большем числе случаев; исследования, направленные на минимизацию объема коммуникационных операций между параллельными вычислительными процессами, реализуемыми на суперкомпьютерах с распределенной памятью; разработка и программная реализация параллельных алгоритмов для решения прикладных задач с использованием предлагаемого подхода.

Литература

1. Лиходед Н.А., Полещук М.А. Метод ранжирования параметров размера блоков вычислений параллельного алгоритма // Доклады НАН Беларуси. 2015. Т. 59, № 4. С. 25–33.
2. Kandemir M., Ramanujam J., Irwin M., Narayanan V., Kadayif I., Parikh A. A compiler based approach for dynamically managing scratch-pad memories in embedded systems // IEEE Transactions on Computer-Aided Design. 2004. Vol. 23, No. 2. P. 243–260.
3. Baskaran M., Bondhugula U., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic data movement and computation mapping for multi-level parallel architectures with explicitly managed memories. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming. Salt Lake City, USA, February 20–23, 2008.
4. Воеводин Вл.В., Воеводин Вад.В. Спасительная локальность суперкомпьютеров // Открытые системы. 2013, № 9. С. 12–15.
5. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Санкт-Петербург: БХВ-Петербург, 2002. 608 с.
6. Xue J., Cai W. Time-minimal tiling when rise is larger than zero // Parallel Computing. 2002. Vol. 28, No. 5. P. 915–939.
7. Baskaran M., Ramanujam J., Sadayappan P. Automatic C-to- CUDA code generation for affine programs. Proceedings of the Compiler Construction, 19th International Conference. Part of the Joint European Conferences on Theory and Practice of Software. Paphos, Cyprus, March 20–28, 2010.
8. Bondhugula U., Baskaran M., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model // Lecture notes in computer science. 2008. No. 4959. P. 132–146.
9. Venkataraman G., Sahni S., Mukhopadhyaya S. A blocked all-pairs shortest-paths algorithm // J. Exp. Algorithmics. 2003. Vol. 8. P. 2.2.

10. Katz G.J., Kider J. All-pairs shortest-paths for large graphs on the GPU // Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware. Sarajevo, Bosnia and Herzegovina: Eurographics Association. 2008. P. 47–55.

11. Lund B.D., Smith J.W. A multi-stage cuda kernel for floyd-warshall // CoRR abs/1001.4108. 2010.

Estimate of locality of parallel algorithms implemented on GPUs

N.A. Likhoded, M.A. Paliashchuk

Belarusian State University

The problem of obtaining blocks of operations and threads of parallel algorithm resulting in a smaller number of accesses to global memory and resulting in the efficient use of caches and shared memory graphics processor is investigated. We formulated and proved statements to assess the volume of communication transactions generated by alternative sizing of blocks, as well as to minimize the number of cache misses due to the use of temporal and spatial locality of data. The research is constructive and allows software implementation for practical use.

Keywords: parallel computing, GPU, minimization of communications, temporal locality, spatial locality.

References

1. Likhoded N.A., Poleshchuk M.A. Metod ranzhirovaniya parametrov razmera blokov vychisleniy parallel'nogo algoritma [Method of Ranking Tiles Size Parameters of Parallel Algorithm] // Doklady NAN Belarusi [Proceedings of the National Academy of Sciences of Belarus]. 2015. Vol. 59, No. 4. P. 25–33.
2. Kandemir M., Ramanujam J., Irwin M., Narayanan V., Kadayif I., Parikh A. A compiler based approach for dynamically managing scratch-pad memories in embedded systems // IEEE Transactions on Computer-Aided Design. 2004. Vol. 23, No. 2. P. 243–260.
3. Baskaran M., Bondhugula U., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic data movement and computation mapping for multi-level parallel architectures with explicitly managed memories. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming. Salt Lake Ciy, USA, February 20–23, 2008.
4. Voevodin Vl.V., Voevodin Vad.V. Spasitel'naya lokal'nost' superkomp'yuterov [The Fortunate Locality of Supercomputers] // Otkrytye sistemy [Open Systems]. 2013, No. 9. P. 12–15.
5. Voevodin V.V., Voevodin Vl.V. Parallel'nye vychisleniya [Parallel Computing]. Sankt-Peterburg: BKhV-Peterburg, 2002. 608 p.
6. Xue J., Cai W. Time-minimal tiling when rise is larger than zero // Parallel Computing. 2002. Vol. 28, No. 5. P. 915–939.
7. Baskaran M., Ramanujam J., Sadayappan P. Automatic C-to- CUDA code generation for affine programs. Proceedings of the Compiler Construction, 19th International Conference. Part of the Joint European Conferences on Theory and Practice of Software. Paphos, Cyprus, March 20–28, 2010.
8. Bondhugula U., Baskaran M., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model // Lecture notes in computer science. 2008. No. 4959. P. 132–146.
9. Venkataraman G., Sahni S., Mukhopadhyaya S. A blocked all-pairs shortest-paths algorithm // J. Exp. Algorithmics. 2003. Vol. 8. P. 2.2.
10. Katz G.J., Kider J. All-pairs shortest-paths for large graphs on the GPU // Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware. Sarajevo, Bosnia and Herzegovina: Eurographics Association. 2008. P. 47–55.
11. Lund B.D., Smith J.W. A multi-stage cuda kernel for floyd-warshall // CoRR abs/1001.4108. 2010.