УДК [004.421.2+004.9](045)

***ВОРОБЬЁВ Владимир Анатольевич**, доктор технических наук, профессор кафедры программирования и высокопроизводительных вычислений института математики, информационных и космических технологий Северного (Арктического) федерального университета имени М.В. Ломоносова.*

Автор 97 научных публикаций, в т. ч. одной монографии и 5 учебных пособий

***ЮФРЯКОВА Ольга Алексеевна**, старший преподаватель кафедры программирования и высокопроизводительных вычислений института математики, информационных и космических технологий Северного (Арктического) федерального университета имени М.В. Ломоносова.Автор 13 научных публикаций, в т. ч. одного учебного пособия*

МЕЛКОЗЕРНИСТЫЕ ЛОКАЛЬНО-ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

В работе рассмотрено понятие мелкозернистого локально-параллельного алгоритма, сформулированы его свойства. Описаны два примера оригинальных МЛП-алгоритма на кольцевой и тороидальной вычислительных структурах. Приведены оценки сложности и эффективности распараллеливания.

***Ключевые слова:** вычислительная сложность, структура алгоритмов, параллельные вычисления, массовое распараллеливание, мелкозернистые локально-параллельные алгоритмы, локальность, архитектура параллельной вычислительной системы, ускорение вычислений.*

Заставить параллельную вычислительную систему или суперкомпьютер работать с максимальной эффективностью на конкретной программе – это сложная задача. Поэтому необходимо тщательное согласование структур данных и алгоритмов с особенностями архитектуры параллельных вычислительных систем. Для этого нужны соответствующие инструментальные средства – операционные системы, поддерживающие многопроцессорную работу и языки программирования, способные конструировать виртуальные вычислительные структуры специального вида и описывать выполнение массово-параллельных, локальных алгоритмов решения трудоемких задач.

Нас интересуют массово-параллельные системы с КАИС-структурами, содержащие до 10^6 и более узлов (ядер, элементарных машин, процессорных элементов и т. п.) на пределах технологических возможностей ближайшего будущего. Проблемы здесь следующие:

1. Достижение фундаментальных технологических пределов.
2. Отказоустойчивая неразрезная технология СБИС.
3. Синтез КАИС-структур на НПМ СБИС.
4. Мелкозернистые локально-параллельные алгоритмы (МЛП-алгоритмы) для прикладных задач [2].

На данный момент преимущественным способом распараллеливания задач является крупноблочное распараллеливание, когда задача разбивается на крупные подзадачи и каждый процессор в составе суперкомпьютера решает выделенную ему часть.

Подобный подход имеет ряд недостатков. Во-первых, процессоры общего назначения менее эффективны, чем специализированные устройства. Во-вторых, для решения определенной задачи большая (оставшаяся) часть процессорной логики является избыточной, таким образом, с одной стороны мы имеем незначительное использование ресурсов, а с другой стороны стоимость подобной архитектуры оказывается значительно выше. Кроме того, в вычислительную сложность параллельных алгоритмов входят не только вычислительные, но и коммуникационные аспекты, которые зачастую являются «узким местом» как в производительности, так и в масштабируемости. Дело в том, что с увеличением числа процессоров блоки, на которые разбивается задача, измельчаются, увеличиваются коммуникационные затраты, решение задачи замедляется: происходит вырождение параллелизма [2, 4, 5].

Создание специализированного компьютера, решающего конкретную задачу требует больших финансовых и временных затрат. Другой путь – использование реконфигурируемых логических устройств, когда есть возможность изменять «внутреннюю логику» процессоров, позволяет обойти вышеперечисленные проблемы. Одним из устройств этого класса является программируемая логическая интегральная схема (ПЛИС) FPGA. FPGA является матрицей элементарных логических модулей, которые содержат блоки умножения, суммирования, а также логические вентили и их блоки коммутации, что позволяет объединять эти модули необходимым образом, формируя произвольную логическую схему.

Еще одной возможностью реализации мелкозернистой локально-параллельной архитектуры матриц вычислителей является использование современных графических процессоров (GPU) таких как, например, графических процессоров фирмы NVIDIA. Это массово параллельные вычислительные устройства, высокая производительность которых достигается за счет использования мультипроцессорного массива, состоящего из SIMD наборов процессоров, так называемая SIMD-архитектура (одна инструкция – много потоков) [3].

Мелкозернистый параллелизм предполагает возможность разбить задачу на множество небольших однотипных подзадач, которые будут исполняться параллельно на отдельных простых процессорах. Данные максимально распределены по системе, а каждый процессор использует минимально возможный набор данных. Таким образом, мелкозернистость, или массовое распараллеливание, означает, что в каждом вычислительном процессе в каждый момент времени содержится минимальное число команд (тело внутреннего цикла) и данных (элементы массивов, необходимые для вычисления одного витка цикла). Обязательным условием эффективного программирования при массовом распараллеливании является локальность взаимодействий, когда обмен данными происходит только в пределах ограниченного физического и структурного радиуса, независимо от размеров задачи и системы.

Рассмотрим подробнее идею мелкозернистого локально-параллельного программирования. Она сводится к следующим требованиям.

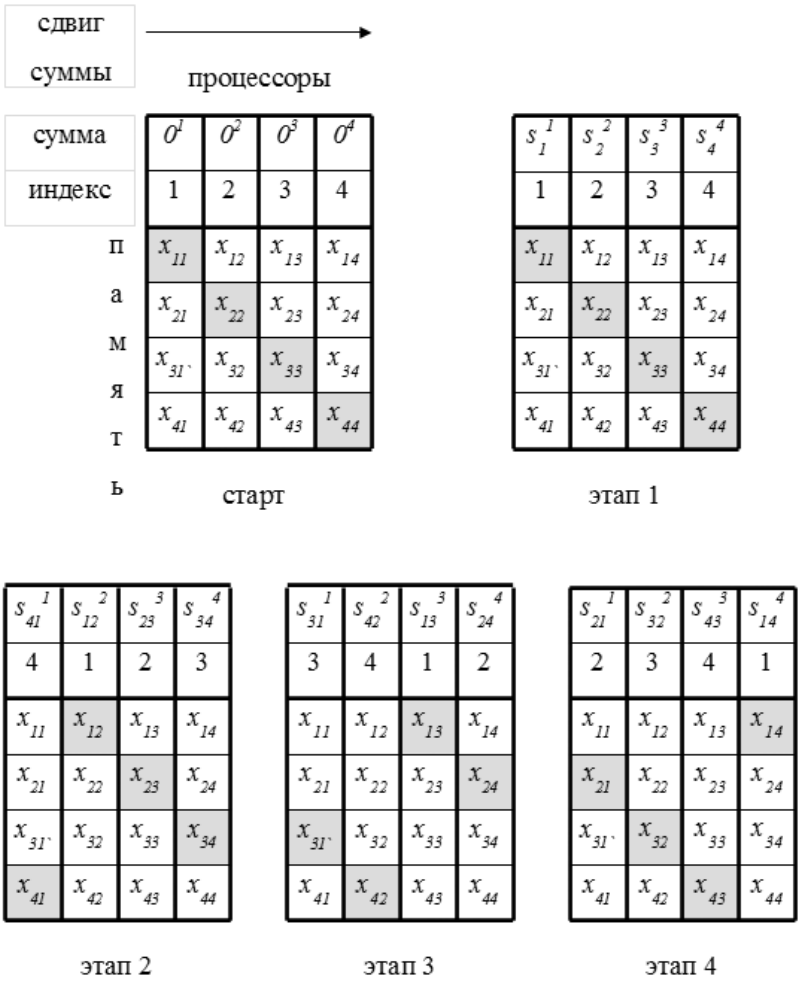
1. Необходимо, чтобы структура вычислительной системы отвечала требованию локальности и масштабируемости. Локальность означает, что логически соседние процессоры должны быть соседями и в физическом пространстве и связи между соседями должны быть ограниченной длины. Масштабируемость означает, что система допускает неограниченное наращивание числа процессоров без нарушения локальности. Таковы линейные, матричные, кольцевые и тороидальные структуры. Гиперкуб такими свойствами не обладает.

2. В памяти каждого процессора хранится минимально возможная часть данных, необходимая только для вычисления одного вит-

ка цикла. Каждый новый вычисляемый виток цикла требует подкачки памяти процессора новой порцией данных от соседей (и только от соседей!).

3. Глобальные взаимодействия процессоров с внешней средой допускаются только изредка. Таковы загрузка данных и программы, выдача результата, пуск и останов программы, некоторые условные переходы по состоянию всей системы.

В качестве примера мелкозернистого локального распараллеливания рассмотрим задачу суммирования элементов матрицы размера $N \times M$ по строкам на кольце из M процессоров. Предполагается, что длина строк значительно превосходит длину столбцов, и что строка целиком не помещается в локальную память одного процессора (в отличие от столбца). Если в кольце столбцы матрицы расположены в памяти отдельных процессоров, то суммирование элементов всех N строк происходит параллельно по наилучшему последовательному алгоритму за время $t(M-1)$, с ускорением M и эффективностью ≈ 1. Идея этого алгоритма представлена на *рисунке*. Он суммирует параллельно элементы N строк за время Mt с ускоре-



Локально-параллельное суммирование строк матрицы на кольце

нием *(M – 1)* и эффективностью $(M-1)/M \to 1$ при $M \to \infty$. Каждый процессор содержит в локальной памяти кроме столбца матрицы значение текущей суммы s_{ij}^{k} и текущий индекс i – номер строки, элемент которой в данный момент прибавляется к текущей сумме. Символ s_{ij}^{k} обозначает сумму от i-го до j-го элементов k-той строки с учетом циклического сдвига индекса по кольцу слева направо.

Например, $s_{42}^{3} = x_{34} + x_{31} + x_{32}$. Вектор-сумма s_{ij} на каждом этапе вычислений циклически сдвигается на один элемент вправо и компонента s_{ij}^{k} суммируется с очередным элементом k-той строки (выделен штриховкой). Очередные значения индекса строки получаются тем же циклическим сдвигом вправо или вычисляются как обычно, если это быстрее.

В качестве другого примера мелкозернистого локального распараллеливания рассмотрим умножение двух матриц размера n^2 при использовании n^2 процессоров, образующих двумерный $n \times n$ тор. Все три матрицы можно поэлементно распределить по процессорам, так что в каждом K_{ij} в начальный момент располагаются три элемента a_{ik} , b_{kj} и $c_{ij} = 0$ (см. *таблицу*).

Тогда, при соответствующих сдвигах строк и столбцов, в каждом процессоре можно выполнять операцию накопления $c_{ij} = c_{ij} + a_{ik} \times b_{kj}$. На *рисунке* представлен вариант реализации такого процесса при $n = 3$. Серым цветом выделены те процессоры, в которых на данном этапе выполняются операции накопления. Те строки и столбцы матрицы, в которых выполнялись вычисления, принимают участие в

МАСШТАБИРУЕМОЕ ЛОКАЛЬНО-ПАРАЛЛЕЛЬНОЕ УМНОЖЕНИЕ МАТРИЦ

a_{11}	a_{12}	a_{13}	a_{13}	a_{11}	a_{12}	a_{12}	a_{13}	a_{11}
b_{11}	b_{12}	b_{13}	b_{31}	b_{12}	b_{13}	b_{21}	b_{32}	b_{13}
c_{11}	c_{12}	c_{13}	c_{11}	c_{12}	c_{13}	c_{11}	c_{12}	c_{13}
a_{21}	a_{22}	a_{23}	a_{21}	a_{22}	a_{23}	a_{23}	a_{21}	a_{22}
b_{21}	b_{22}	b_{23}	b_{11}	b_{22}	b_{23}	b_{31}	b_{12}	b_{23}
c_{21}	c_{22}	c_{23}	c_{21}	c_{22}	c_{23}	c_{21}	c_{22}	c_{23}
a_{31}	a_{32}	a_{33}	a_{31}	a_{32}	a_{33}	a_{31}	a_{32}	a_{33}
b_{31}	b_{32}	b_{33}	b_{21}	b_{32}	b_{33}	b_{11}	b_{22}	b_{33}
c_{31}	c_{32}	c_{33}	c_{31}	c_{32}	c_{33}	c_{31}	c_{32}	c_{33}
I			II			III		
a_{11}	a_{12}	a_{13}	a_{13}	a_{11}	a_{12}			
b_{11}	b_{22}	b_{33}	b_{31}	b_{12}	b_{23}	c_{11}	c_{12}	c_{13}
c_{11}	c_{12}	c_{13}	c_{11}	c_{12}	c_{13}			
a_{22}	a_{23}	a_{21}	a_{21}	a_{22}	a_{23}			
b_{21}	b_{32}	b_{13}	b_{11}	b_{22}	b_{33}	c_{21}	c_{22}	c_{23}
c_{21}	c_{22}	c_{23}	c_{21}	c_{22}	c_{23}			
a_{33}	a_{31}	a_{32}	a_{32}	a_{33}	a_{31}			
b_{31}	b_{12}	b_{23}	b_{21}	b_{32}	b_{13}	c_{31}	c_{32}	c_{33}
c_{31}	c_{32}	c_{33}	c_{31}	c_{32}	c_{33}			
IY			Y			YI		

циклических сдвигах исходных матриц *A* и *B*. Элементы матрицы *A* циклически сдвигаются вправо, элементы *B* – вниз. Всего 5 этапов параллельного накопления потребовалось, чтобы выполнить все необходимые 27 таких операций. Вообще, если умножение матриц требует n^3 операций накопления, то рассмотренный масштабируемый алгоритм на n^2 процессорах укладывается в $(2n - 1)$ этапов параллельного накопления.

Можно заметить, что для выполнения такого параллельного алгоритма требуется всего $O(n)$ операций сдвига, умножения и сложения на каждом процессоре, когда при последовательном алгоритме потребовалось бы выполнить все необходимые $O(n^3)$ операций умножения и сложения.

Оценим производительность этого МЛП-алгоритма. Его сложность $O(n)$. Процесс вычислений состоит из 3-х этапов: пересылка исходных данных на каждый процессор; вычисление произведения блоков; пересылка результата. Пересылка данных занимает n тактов.

Пусть $t_1(n)$ – время умножения матриц размерности $n \times n$ при использовании одного процессора, а $t_p(n)$ – время умножения матриц на $p = n^2$ процессорах. При последовательном алгоритме потребуется n^3 операций. В случае параллельного алгоритма умножение блоков выполняется за время nT_{mult}. Тогда $t_1(n) = n^3 \, T_{mult}$, $t_p(n) = n \, T_{mult}$, и ускорение $S_p(n) = t_1(n)/ t_p(n) = n^3 \, T_{mult}/(n \, T_{mult}) = n^2$.

Итак, ускорение МЛП-алгоритма по сравнению с наилучшим последовательным алгоритмом равно n^2.

На этих примерах отчетливо видны характерные особенности МЛП-алгоритмов, их жесткая зависимость от архитектуры вычислительной системы. Перефразируя известный афоризм Э. Дейкстры: «Программа = алгоритм + структура данных», можно высказать следующую формулу.

Параллельная программа = архитектура параллельной вычислительной системы + распределенная структура данных + параллельный алгоритм.

В настоящее время построено и исследовано мало МЛП-алгоритмов. Это обстоятельство можно объяснить недостаточной, до последнего времени, распространенностью доступных аппаратных платформ, удовлетворяющих принципу локальности и содержащих большое количество простых вычислительных ядер с возможностью быстрых обменов между ядрами. Но ситуация быстро меняется в связи с быстрым развитием GPU, инструментов для их программирования в качестве параллельных вычислителей и гибридных суперкомпьютерных архитектур с их использованием. К сожалению, SIMD-архитектура GPU сама по себе не удовлетворяют требованиям локальности и не может использоваться для МЛП-вычислений. Систематическое построение и исследование МЛП-алгоритмов, как существенных компонентов больших вычислительных задач, становится все более актуальным [1–5], поскольку может стимулировать разработку неразрезных процессорных матриц (НПМ) с локальной MIMD-архитектурой.

Список литературы

1. *Воеводин В.В.* Вычислительная математика и структура алгоритмов. М., 2010.

2. *Воробьёв В.А*. Об эффективности параллельных вычислений // Автометрия.2000. № 1.С. 50–58.

3. *Дербина Ю.В., Березовский В.В.* Подходы к реализации мелкозернистой локально-параллельной архитектуры для моделирования экологических рисков // материалы IX междунар. конф.-семинара «Высокопроизводительные параллельные вычисления на кластерных системах». Владимир, 2009. С. 140–144.

4. *Юфрякова О.А.* Оптимизация количества процессоров для эффективного исполнения параллельных алгоритмов // Научный сервис в сети Интернет: поиск новых решений: тр. междунар. суперкомпьютерной конфер. М., 2012.

5. *Юфрякова О.А.* Препятствия к неограниченному масштабированию параллельных систем // Научный сервис в сети Интернет: экзафлопсное будущее: тр. междунар. суперкомпьютерной конфер. М., 2011.

References

1. Voevodin V.V. *Vychislitel'naya matematika i struktura algoritmov* [Computational Mathematics and Algorithm Structure]. Moscow, 2010. 168 p.

2. Vorob'ev V.A. Ob effektivnosti parallel'nykh vychisleniy [On the Efficiency of Parallel Computing]. *Avtometriya,* 2000, no 1, pp. 50–58.

3. Derbina Yu.V., Berezovskiy V.V. Podkhody k realizatsii melkozernistoy lokal'no-parallel'noy arkhitektury dlya modelirovaniya ekologicheskikh riskov [Approaches to Implementation of Fine-grained Local-parallel Architecture for Environmental Risks Modeling]. *Materialy IX mezhdunarodnoy konferentsii-seminara «Vysokoproizvoditel'nye parallel'nye vychisleniya na klasternykh sistemakh»* [Proc. 9th int. conf.-seminar "High-performance parallel computing on cluster systems"]. Vladimir, 2009, pp. 140–144.

4. Yufryakova O.A. Optimizatsiya kolichestva protsessorov dlya effektivnogo ispolneniya parallel'nykh algoritmov [Optimization of the Number of Processors for Efficient Execution of Parallel Algorithms]. *Nauchnyy servis v seti Internet: poisk novykh resheniy: Trudy Mezhdunarodnoy superkomp'yuternoy konferentsii.* [Scientific service in the Internet: exaflop future: Proc. int. supercomputing conf.]. Moscow, 2012.

5. Yufryakova O.A. Prepyatstviya k neogranichennomu masshtabirovaniyu parallel'nykh system [Obstacles to Unlimited Scalability of Parallel Systems]. *Nauchnyy servis v seti Internet: ekzaflopsnoe budushchee: Trudy Mezhdunarodnoy superkomp'yuternoy konferentsii* [Scientific service in the Internet: exaflop future: Proc. int. supercomputing conf.]. Moscow, 2011.

Vorobyev Vladimir Anatolyevich
Institute of Mathematics, Information and Space Technologies,
Northern (Arctic) Federal University named after M.V. Lomonosov (Arkhangelsk, Russia)

Yufryakova Olga Alekseevna
Institute of Mathematics, Information and Space Technologies,
Northern (Arctic) Federal University named after M.V. Lomonosov (Arkhangelsk, Russia)

FINE-GRAINED LOCAL PARALLEL ALGORITHMS

The paper examines the concept of fine-grained local parallel algorithm and states its properties. Two examples of original MLP-algorithms on the ring and toroidal computing structures are given. Complexity and efficiency of parallelization were estimated.

Keywords: *computational complexity, algorithm structure, parallel computing, massive parallelism, fine-grained local parallel algorithms, locality, architecture of a parallel computing system, computational speed-up.*

Контактная информация:
Воробьёв Владимир Анатольевич
e-mail: vva100@atnet.ru
Юфрякова Ольга Алексеевна
e-mail: yufryakova@mail.ru

Рецензент – *Андреев П.Д.*, кандидат физико-математических наук, доцент кафедры информационной безопасности института математики, информационных и космических технологий Северного (Арктического) федерального университета имени М.В. Ломоносова