

THE COMPARISON OF MPI AND LUNA CAPABILITIES USING THE IMPLEMENTATION OF CELLULAR AUTOMATA WAVE INTERFERENCE

V. P. Markova*, M. B. Ostapkevich**

The Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russian Federation

*The Novosibirsk National Research State University,
630090, Novosibirsk, Russian Federation

**The Novosibirsk State Technical University,
630073, Novosibirsk, Russian Federation

In this paper, a parallel implementation of the cellular-automata interference algorithm for two waves using the fragmented programming technology and Luna system based on it is proposed. The technology is based on a strategy of data flow control. Unlike existing systems and technologies, LuNA provides a unified technology for implementing parallel programs on a heterogeneous multicomputer. The LuNA program contains a description of data fragments, computational fragments, and information dependencies between them. In the work, the LuNA program was executed on a computational cluster with homogeneous nodes. The results of comparison of the LuNA and MPI implementations showed that the execution time of the LuNA program exceeded that of the MPI program. This is due to the peculiarities of algorithms used for the distribution, search and transfer of data and computation fragments between the nodes of a cluster. The complexity of writing the LuNA program is much lower than for the MPI program.

Key words: parallel programming, fragmented programming, graph of information dependencies, LuNA system, cellular automata, lattice gas, interference simulation.

References

1. StreamIt Project Homepage. [Electron. resource]. <http://groups.csail.mit.edu/cag/streamit/>, last accessed 2017/03/01.
2. Language for Streaming Applications // Proceeding CC '02. Proceedings of the 11th International Conference on Compiler Construction. 2002. P. 179–196.
3. Michel Steuwer, Toomas Rimmelg, and Christophe Dubach. Lift: a functional data-parallel IR for high-performance GPU code generation // CGO'17 Proceedings of the 2017 International Symposium on Code Generation and Optimization. 2017. P. 74–85.
4. V. Malyshkin. Active Knowledge, LuNA and Literacy for Oncoming Centuries // LNCS. Springer, Heidelberg. 2015. V. 9465. P. 292–303.
5. M. Zhang, D. Cule, L. Shafai, G. Bridges and N. Simons. Computing Electromagnetic Fields in Inhomogeneous Media Using Lattice Gas Automata // Proceedings of 1998 Symposium on Antenna Technology and Applied Electromagnetic. 1998. Aug. 14–16, Ottawa, Canada.

6. V. Markova. Designing a collision matrix for a cellular automaton with rest particles for simulation of wave processes // Bull. Nov. Comp. Center, Comp. Science. NCC Publisher, Novosibirsk. 36. 2014. P. 47–56.

7. Conditions for interference. [Electron. resource]. http://physics.bu.edu/~duffy/sc545_notes09/interference_conditions.html

8. V. Malyshkin, V. Perepelkin The PIC implementation in LuNA system of fragmented programming // Journal of Supercomputing. 2014. V. 69, I. 1. P. 89–97.

СРАВНЕНИЕ ВОЗМОЖНОСТЕЙ MPI И LuNA НА ПРИМЕРЕ РЕАЛИЗАЦИИ МОДЕЛИ КЛЕТОЧНО-АВТОМАТНОЙ ИНТЕРФЕРЕНЦИИ ВОЛН

В. П. Маркова*, М. Б. Остапкевич**

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

*Новосибирский национальный исследовательский государственный университет,
630090, Новосибирск, Россия

**Новосибирский государственный технический университет,
630073, Новосибирск, Россия

УДК 519.688

В статье рассматривается параллельная реализация алгоритма клеточно-автоматной интерференции двух волн с использованием технологии фрагментированного программирования и основанной на ней системы LuNA. Технология основана на стратегии управления потоками данных. В отличие от других известных технологий и систем, LuNA предоставляет унифицированный способ написания параллельных программ для мультимикомпьютеров с неоднородными узлами. Программа на LuNA содержит описание фрагментов данных, вычислительных фрагментов и информационных зависимостей между ними. В данной работе LuNA программа выполняется на мультимикомпьютере с однородными узлами. Результаты сравнения LuNA и MPI показали, что время выполнения LuNA программы больше, чем время выполнения MPI программы. Это обусловлено особенностями алгоритмов распределения, поиска и передачи данных и вычислительных фрагментов между узлами мультимикомпьютера. Написание программы для LuNA существенно проще, чем написание MPI программы.

Ключевые слова: параллельное программирование, фрагментированное программирование, система LuNA, клеточный автомат, решеточный газ, моделирование интерференции.

Введение. В настоящее время для написания параллельных программ на мультимикомпьютерах (компьютерах с распределенной памятью) используется стандарт MPI. Он получил самое широкое распространение на большинстве мультимикомпьютеров с однородными узлами благодаря тому, что он позволяет создавать переносимые программы. Используя MPI, программист при написании программ, помимо описания собственно вычислений, должен программировать выделение ресурсов, межузловые коммуникации, балансировку нагрузки между узлами и определять порядок вычислений.

С появлением компьютеров с гетерогенными узлами задача распараллеливания усложнилась, так как разные вычислители в составе узла имеют разную архитектуру и каждый из них программируется с использованием отдельного интерфейса (технологии). Построение эффективной программы для таких компьютеров выполняется двумя способами.

Первый заключается в использовании MPI для организации межузлового параллелизма, а OpenMP, OpenCL, CUDA, HLS — для организации внутриузлового параллелизма.

Второй способ предполагает построение единой технологии реализации параллельных программ на гетерогенном мультикомпьютере. Примерами таких технологий являются StreamIt [1, 2] и Lift [3]. В рамках второго способа в ИВМ и МГ СО РАН была реализована технология фрагментированного программирования, основанная на стратегии управления потоками данных. На ее основе построена система программирования LuNA [4].

В статье исследованы реализации алгоритма клеточно-автоматной интерференции двух волн. Одна реализация построена с использованием MPI, а другая — на базе системы LuNA. Перечислены характеристики системы LuNA. Проведено их сравнение, и сформулированы преимущества использования системы LuNA.

1. Основные определения и характеристики системы LuNA. Система LuNA — это инструмент для построения параллельных программ на базе технологии фрагментированного программирования. LuNA система состоит из транслятора языка сборки и подсистемы исполнения фрагментированных программ.

Базовыми понятиями в системе являются фрагменты данных, фрагменты кода и фрагменты вычислений.

Фрагмент данных — это множество соседних ячеек массива заданного размера.

Фрагмент кода — это функция, которая получает значения некоторых входных фрагментов данных и вычисляет по ним значения выходных фрагментов данных. При построении фрагментированных программ (ФП) используются два типа фрагментов кода: атомарные и структурированные. *Атомарные фрагменты кода* в системе представляются как функции Си программ. *Структурированные* фрагменты кода содержат сборку из фрагментов вычислений.

Фрагмент вычислений — это обращение (вызов на исполнение) к фрагменту кода с указанием имен всех входных и выходных фрагментов данных.

Написание LuNA-программы состоит из следующих шагов.

1) В исходной последовательной программе на языках Си/Си++ выделяются участки кода, которые отвечают за вычисления. В этих участках кода обработка произвольных объектов данных заменяется обработкой фрагментов данных, а сами участки оформляются как атомарные фрагменты кода.

2) На языке сборки LuNA описываются структурированные фрагменты кода и информационные зависимости между фрагментами вычислений и фрагментами данных. В отличие от MPI программы, в сборочной программе (это синоним LuNA-программы) нет необходимости жестко задавать порядок вычислений, управлять выделением ресурсов и программировать межузловые коммуникации (см. таблицу).

3) Вставляются инструкции, которые обеспечивают освобождение памяти, занимаемой фрагментами данных, которые больше не используются.

4) Определяются размеры фрагментов данных, при которых время выполнения минимально.

На вход системы LuNA поступают фрагментированная программа на языке сборки и модуль на Си/Си++, реализующий атомарные фрагменты кода. Эта программа содержит все фрагменты кода, фрагменты данных и информационные зависимости между ними. Система LuNA выполняет следующие шаги.

1) Транслятор системы преобразует исходную LuNA-программу во внутреннее представление, с которым может работать исполнительная подсистема.

2) Исполнительная подсистема LuNA автоматически распределяет фрагменты кода и данных по узлам в соответствии с имеющимися ресурсами мультимониторного компьютера. При программировании с использованием MPI программист должен сам проводить такое распределение (см. таблицу).

3) При выполнении фрагментированной программы исполнительная подсистема определяет окончательный порядок выполнения фрагментов вычислений. Для этого она ведет учет фрагментов кода, готовых к исполнению, и распределяет их на исполнение по потокам. При выполнении программы на нескольких узлах система автоматически обеспечивает необходимую пересылку фрагментов данных между узлами. При написании же MPI программы разработчик в явном виде определяет порядок выполнения и описывает все межузловые пересылки данных (см. таблицу). При несбалансированности вычислительной нагрузки на узлы исполнительная подсистема перераспределяет нагрузку между узлами. В MPI такую балансировку нагрузки реализует сам программист в своей программе.

4) Исполнение фрагментированной программы продолжается, пока есть исполняющиеся или готовые к исполнению фрагменты кода.

Опыт программирования в LuNA-системе позволяет сформулировать ее преимущества перед исполнением MPI-программой.

1) Отсутствие жесткой привязки фрагментов данных в LuNA к MPI процессам позволяет LuNA-программе адаптироваться к имеющимся доступным ресурсам вычислительной системы. Это возможно как при запуске программы, так и при ее исполнении, когда множество доступных ресурсов изменяется.

2) Малый размер фрагментов кода и фрагментов данных обеспечивает сбалансированность вычислительной нагрузки как на процессах в разных узлах вычислительной системы, так и на отдельных потоках внутри узлов в автоматическом режиме. В MPI программе задача по балансировке или не решается вообще, или решается программистом.

3) Описание информационных зависимостей между фрагментами в LuNA — более простая задача для программиста, чем описание жесткой последовательности действий в MPI программе, особенно при большом их числе. Следствием этого упрощения является уменьшение количества ошибок в программе, связанных с неверно заданной последовательностью действий. Такие ошибки в параллельных программах трудно выявлять и исправлять.

2. Клеточно-автоматное моделирование волновых процессов. Для моделирования интерференции двух волн используется простейший класс одночастичных недетерминированных клеточных автоматов с одной частицей покоя массой 2, известный в литературе как HPP1gr [5]. Клеточные автоматы (КА) описывают природные явления множеством гипотетических частиц, которые движутся в решеточном пространстве по некоторым правилам. Эти правила представляют моделируемое явление на микроуровне, исходя из общих законов физики.

Интерес к клеточно-автоматному (КА) моделированию объясняется рядом его достоинств. Во-первых, отсутствие ошибок округления и способность моделировать нелинейные и разрывные процессы. Во-вторых, простота задания граничных условий. И наконец, неограниченные возможности параллельной реализации задач на современных суперкомпьютерах.

Клеточный автомат HPP1gr определен на 2D решетке. Каждый узел решетки соединен с соседями единичными решеточными векторами e_i , $i = 1, 2, 3, 4$, и содержит 4 движущиеся частицы и частицу покоя. Движущиеся частицы имеют единичную массу и единичную скорость c_i , направленную вдоль одного из четырех векторов решетки. В каждом направ-

Таблица

Реализация системных функций в MPI и LuNA

Функция	MPI	LuNA
Порядок вычислений	Жестко задается программистом, неизменен при выполнении	Окончательный порядок определяется уже при исполнении, когда есть информация о фрагментах кода, для которых готовы и собраны в одном узле все исходные данные и которые могут запускаться
Управление выделением ресурсов в узле	Реализуется программистом	Реализуется системой LuNA
Распределение данных по узлам	Реализуется программистом	Выполняется системой на основании информации о доступных ресурсах вычислительной системы
Межузловые коммуникации	Явно кодируются программистом	Активизируются подсистемой исполнения на основе информации о местонахождении запрашиваемых данных
Балансировка нагрузки	Не реализуется или реализуется программистом	Реализуется системой LuNA

лении может двигаться только одна из частиц, находящихся в узле решетки. Частица *покоя* имеет массу 2 и нулевую скорость.

Каждому узлу решетки с именем r поставлена в соответствие клетка с тем же именем. Множество частиц в клетке определяет ее состояние $\mathbf{s}(r)$. Вектор $\mathbf{s}(r)$ состоит из 5 элементов. Значение первых четырех элементов вектора \mathbf{s} показывает наличие ($s_i(r) = 1$) или отсутствие ($s_i(r) = 0$) движущейся частицы со скоростью \mathbf{c}_i в клетке с именем r , последний элемент вектора $\mathbf{s}(r)$ показывает наличие $s_5(r) = 1$ или отсутствие $s_5(r) = 0$, частицы покоя массой 2 в клетке с именем r . Пара (\mathbf{s}, r) называется *клеткой*. Общая сумма масс частиц в клетке с именем r называется *модельной плотностью* $\rho(r)$. Множество клеток, в котором все клетки имеют уникальные имена, образует *клеточный массив*. Множество состояний всех клеток массива $\Omega(t)$ в момент времени t называется *глобальным состоянием автомата*. Смена глобальных состояний автомата описывает *эволюцию* КА.

Клеточный автомат работает синхронно: все клетки автомата меняют свои состояния одновременно на каждом итерационном шаге. Шаг состоит из 2-х фаз: столкновение и сдвиг. На фазе *столкновения* частицы в каждой клетке соударяются друг с другом таким образом, что суммарные масса и импульс частиц сохраняются. Функция столкновения создает или разрушает движущуюся частицу со скоростью в клетке с именем r в момент времени t и зависит только от ее исходного состояния в данный момент времени. Например, клетка в состоянии $(00101)(\mathbf{5})$ на фазе столкновения меняет его на 3 состояния: $(00101)(\mathbf{10})$ с вероятностью $p_{5 \rightarrow 10}$ (рис. 1, а), $(10000)(\mathbf{16})$ с вероятностью $p_{5 \rightarrow 16}$ (рис. 1, б) и остается в своем состоянии с вероятностью $p_{5 \rightarrow 5}$ (рис. 1, в). На фазе *сдвига* движущаяся

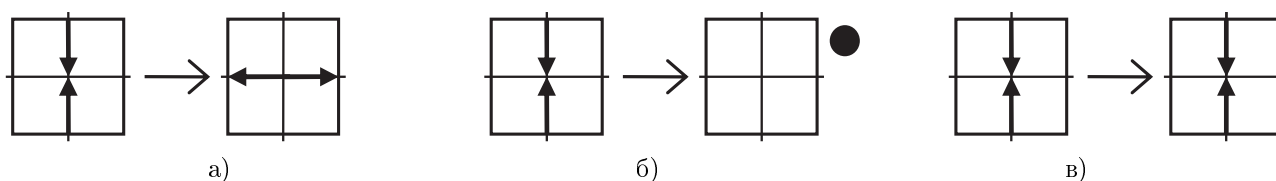
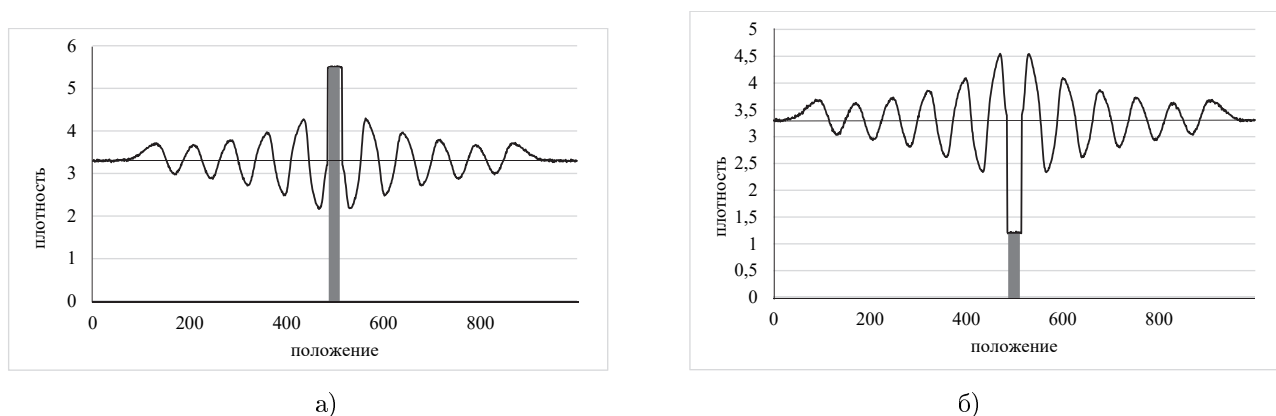
Рис. 1. Пример правила столкновения: а) переход $5 \rightarrow 10$, б) переход $5 \rightarrow 16$, в) переход $5 \rightarrow 5$.

Рис. 2. Функционирование источника периодических волн

щиеся частицы в каждой клетке сдвигаются в сторону ближайшего соседа с единичной скоростью.

Периодическая волна моделируется эволюцией КА, определенного на массиве 1400×1200 . Источник внешнего возмущения расположен в центре и представляет собой окружность радиусом 30 клеток. Исходное состояние клеток массива (модельная плотность) задается генератором случайных чисел в течение одного такта. Модельная плотность клеток источника возмущения составляет $5,5$: генератор с вероятностью 1 генерирует движущиеся частицы и с вероятностью $0,7$ генерирует частицы покоя с массой 2. Модельная плотность клеток остальной части массива равна $3,3$ (движущиеся частицы генерируются с вероятностью $0,7$, частицы покоя $0,5$ генерирует с вероятностью $0,25$). Граничные условия — периодические. Ниже клеточный массив с исходной (невозмущенной) плотностью, граничными условиями, источником внешнего возмущения и матрицей столкновения частиц будем называть *средой*.

Моделирование периодической волны основано на периодическом изменении плотности клеток источника. В начале каждого периода колебаний в источнике формируется исходная плотность частиц $5,5$ (см. рис. 2, а). Через полпериода колебания плотность клеток источника изменяется таким образом, чтобы амплитуда волны над уровнем невозмущенной плотности $3,3$ и под ним была одинакова (см. рис. 2, б). Для данного примера значение плотности частиц составляет $1,1$. Моделирование показало, что заданной среды период волны равен 150 шагам, а ее длина составляет 75 клеток.

3. Моделирование интерференции двух волн. Интерференция двух волн моделируется эволюцией клеточного автомата размера 1400×1200 и плотности клеток среды $3,3$. Два круговых источника периодических волн находятся на расстоянии четырех длин волны (целое число длин волн — условие существования конструктивной интерференции

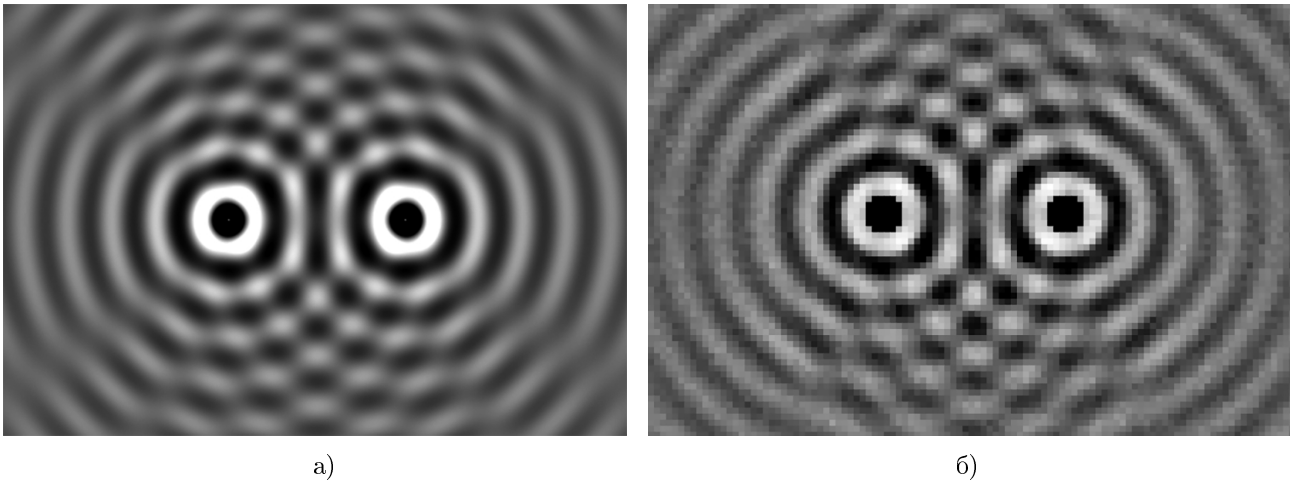


Рис. 3. Интерференция по формуле сферической волны (а) и в модели НРPr (б)

[7]). На рис. 3 показаны две похожие интерференционные картины двух волн, которые получены разными способами и на разных шагах моделирования. Первая промоделирована традиционным способом на шаге времени 100 (рис. 3, а), а вторая — клеточно-автоматным способом на 975-м шаге эволюции клеточного автомата (рис. 3, б).

4. Описание фрагментированного алгоритма КА-интерференции. В реализации исходный клеточный массив разбивается на фрагменты данных с топологией „линейка“.

Главными фрагментами данных в реализации являются:

- 1) $a[t][x]$ — фрагменты, хранящие начальное состояние клеточного массива на шаге моделирования t ,
 - 2) $b[t][x]$ — фрагменты, хранящие состояние клеточного массива после вычисления столкновения частиц на шаге моделирования t .
 - 3) $bu[t][x]$, $bd[t][x]$ — фрагменты для хранения теневых граней.
- Во фрагменте $bu[t][x]$ дублируется верхняя строка фрагмента $b[t][x]$. Во фрагменте $bd[t][x]$ дублируется нижняя строка фрагмента $b[t][x]$. Введение фрагментов данных $bu[t][x]$, $bd[t][x]$ позволяет уменьшить объем пересылок между узлами, так как их размер существенно меньше, чем у фрагмента $b[t][x]$.

Индекс x определяет положение фрагмента в пространстве. Для фрагмента, содержащего самые верхние строки клеточного массива, значение x равно 0. x для фрагмента, содержащего самые нижние строки массива, равен числу фрагментов, на которое разбит массив, минус единица.

Фрагментированная реализация алгоритма КА-интерференции представлена фрагментами кода:

- 1) *init* — инициализация начального состояния клеточного массива,
- 2) *collision* — вычисление операции столкновения частиц в клетках,
- 3) *propagation* — вычисление перелета частиц между клетками,
- 4) *print* — печать результата моделирования в файл.

На рис. 4 показан фрагмент графа информационных зависимостей для двух исходных фрагментов $a[t][x]$ и $a[t][x + 1]$ на шаге времени t . Они являются исходными фрагментами для фрагментов вычислений $collision[t][x]$ и $collision[t][x + 1]$. Как только эти фрагмен-

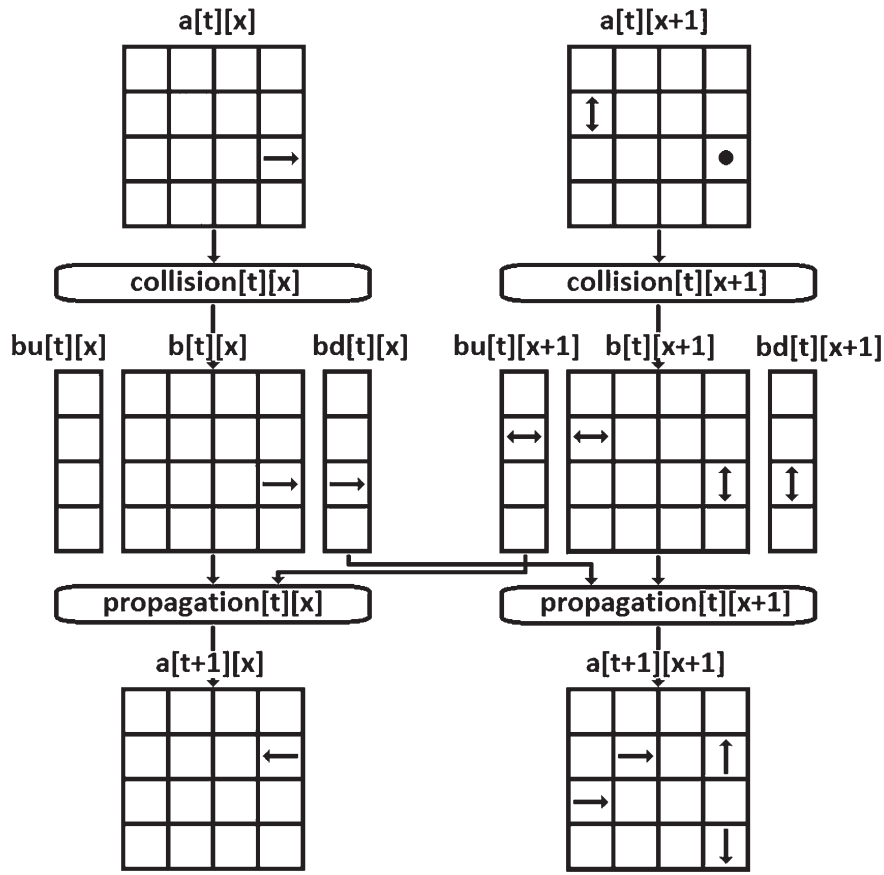


Рис. 4. Часть графа информационных зависимостей реализации HPPgr в LuNA

ты данных вычислены, фрагменты вычислений $collision[t][x]$ и $collision[t][x + 1]$ готовы к выполнению. В какой-то момент времени они запускаются на исполнение. После завершения выполнения $collision[t][x]$ создаются фрагменты данных $b[t][x]$, $bu[t][x]$, $bd[t][x]$. Эти фрагменты хранят состояние частей клеточного массива на шаге t после применения правила коллизии. Аналогично, после завершения $collision[t][x + 1]$ создаются фрагменты данных $b[t][x]$, $bu[t][x]$, $bd[t][x]$. В этот момент фрагменты данных $a[t][x]$ и $a[t][x + 1]$ больше не будут использоваться никакими фрагментами вычислений и могут быть удалены из системы, а освободившаяся память использоваться для хранения новых фрагментов данных. Когда вычислены фрагменты данных $bd[t][x - 1]$, $b[t][x]$, $bu[t][x + 1]$, фрагмент вычислений $propagation[t][x]$ готов к исполнению. Аналогично, фрагмент $propagation[t][x + 1]$ может запускаться, когда посчитаны $bd[t][x]$, $b[t][x + 1]$, $bu[t][x + 2]$. Результатом работы фрагментов вычислений $propagation[t][x]$, $propagation[t][x + 1]$ являются фрагменты данных $a[t + 1][x]$ и $a[t + 1][x + 1]$, хранящие состояние клеточного массива на шаге времени $t + 1$.

LuNA-программа реализует КА-интерференцию в клеточном массиве размером 4096×4096 . На рис. 5 показана зависимость времени выполнения программы от размера фрагментов. Все замеры проводились при числе узлов 1, потоков 8. Минимальное время наблюдается при размере фрагментов 256. При уменьшении размера фрагментов увеличивается их число и растут накладные расходы системы на их обработку. При увеличении размера фрагментов количество обрабатываемых потоком фрагментов становится слишком

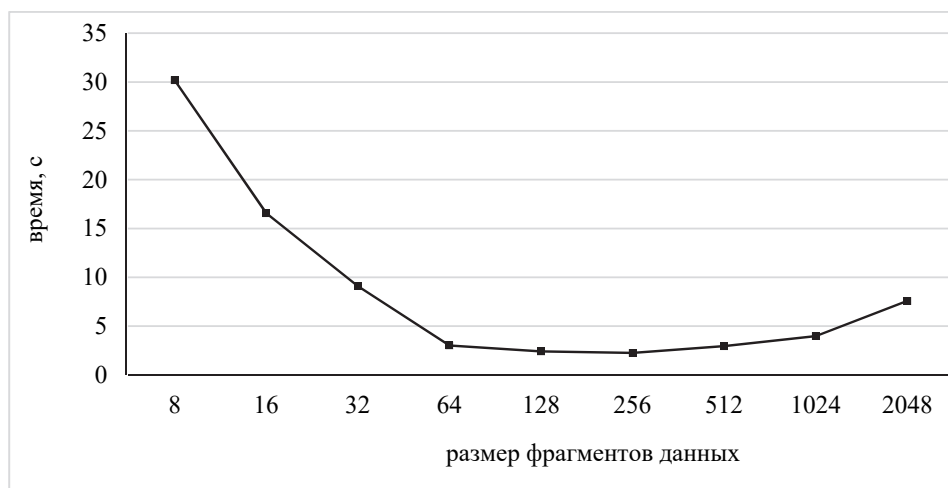


Рис. 5. Зависимость времени выполнения программы от размера фрагмента

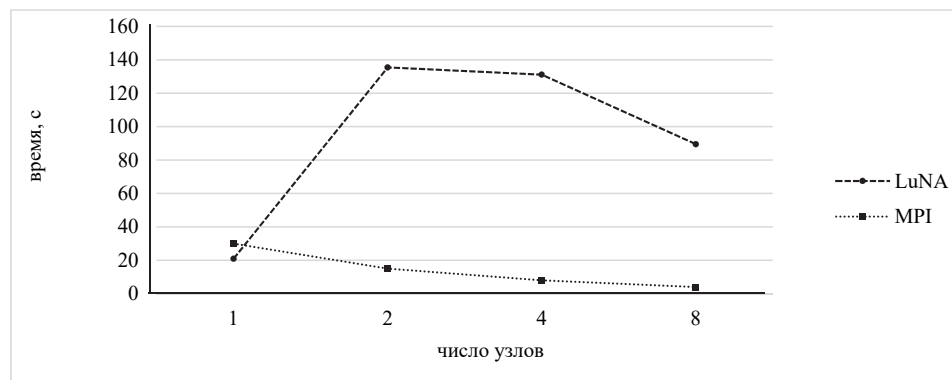


Рис. 6. Зависимость времени выполнения программы от числа используемых узлов

маленьким, и система менее качественно балансирует вычислительную нагрузку между потоками.

Результаты сравнения времени выполнения реализаций на MPI и LuNA представлены на рис. 6. Время выполнения LuNA реализации при использовании нескольких узлов существенно превышает время выполнения MPI реализации. Первые эксперименты и анализ временной сложности LuNA-программы показали: улучшение временной сложности может быть достигнуто за счет оптимизации алгоритмов распределения, поиска и передачи фрагментов данных между узлами мультимасштабных компьютеров и алгоритмов управления выделением памяти для фрагментов данных внутри узлов.

Заключение. Опыт реализации программ показал, что, хотя текущая реализация LuNA проигрывает MPI в плане времени выполнения, у технологии LuNA есть ряд существенных преимуществ перед MPI.

Для задач с неравномерной вычислительной нагрузкой на узлы использование встроенной в LuNA динамической балансировки дает сопоставимый по времени выполнения результат по сравнению с ручной реализацией балансировки в MPI программе [8].

LuNA программы не имеют такой привязки к архитектуре и доступным ресурсам компьютера, как MPI программы. Поэтому LuNA программы обладают существенно более высокой переносимостью.

Отсутствие необходимости задавать порядок вычислений и программировать межузловые коммуникации существенно упрощает программирование в LuNA по сравнению с MPI. Эти же особенности LuNA устраняют появление некоторых видов ошибок, свойственных при программировании в MPI. Описание информационных зависимостей между фрагментами имеет локальный характер, в отличие от задания порядка вычислений в MPI. Это упрощает отладку LuNA программ. Все это вместе уменьшает время разработки параллельных реализаций вычислительных алгоритмов в системе LuNA по сравнению с MPI.

Список литературы

1. StreamIt Project Homepage. [Electron. resource]. <http://groups.csail.mit.edu/cag/streamit/>
2. William Thies, Michal Karczmarek, Saman P. Amarasinghe. StreamIt: A Language for Streaming Applications // Proceeding CC '02. Proceedings of the 11th International Conference on Compiler Construction. 2002. P. 179–196.
3. Michel Steuwer, Toomas Rimmelg, and Christophe Dubach. Lift: a functional data-parallel IR for high-performance GPU code generation // CGO'17 Proceedings of the 2017 International Symposium on Code Generation and Optimization. 2017. P. 74–85.
4. V. Malyshkin. Active Knowledge, LuNA and Literacy for Oncoming Centuries // LNCS. Springer, Heidelberg. 2015. V. 9465. P. 292–303.
5. M. Zhang, D. Cule, L. Shafai, G. Bridges and N. Simons. Computing Electromagnetic Fields in Inhomogeneous Media Using Lattice Gas Automata // Proceedings of 1998 Symposium on Antenna Technology and Applied Electromagnetic. 1998. Aug. 14–16, Ottawa, Canada.
6. V. Markova. Designing a collision matrix for a cellular automaton with rest particles for simulation of wave processes // Bull. Nov. Comp. Center, Comp. Science. NCC Publisher, Novosibirsk. 36. 2014. P. 47–56.
7. Conditions for interference. [Electron. resource]. http://physics.bu.edu/~duffy/sc545_notes09/interference_conditions.html
8. V. Malyshkin, V. Perepelkin The PIC implementation in LuNA system of fragmented programming // Journal of Supercomputing. 2014. V. 69. I. 1. P. 89–97.



Маркова Валентина Петровна — канд. техн. наук, доцент, старш. науч. сотр. Института вычислительной математики и математической геофизики (ИВМ и МГ СО РАН, г. Новосибирск); e-mail: markova@ssd.sccc.ru.

Маркова В. П. окончила факультет автоматики и вычислительной техники Новосибирского государственного технического университета в 1970 г. С 1970 по 1986 гг. работала в институте математики СО АН СССР. В 1980

г. защитила кандидатскую диссертацию „Спектральные методы синтеза функций k -значной логики“. С 1986 года и по настоящее время работает в ИВМ и МГ СО РАН. Научные исследования связаны с клеточно-автоматным моделированием пространственной динамики. Маркова В. П. является доцентом кафедры Параллельных вычислений в Новосибирском национальном исследовательском государственном университете.

Markova V. graduated from the Novosibirsk State Technical University in 1970. From 1970 to 1986 she worked at the Institute of Mathematics

SB AS. She received Ph.D. in 1980. Since 1987 she works at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS. Her research interests include cellular automata simulation of spacial dynamics. She is an assistant professor of the department of Parallel computations at the Novosibirsk National Research State University.



Остапкевич Михаил Борисович — младш. науч. сотр. Института вычислительной математики и математической геофизики; e-mail: ostap@ssd.sccc.ru.

Окончил факультет АВТ Новосибирского государственного технического университета в 1997 г. С 1999

года и по настоящее время работает в ИВМ и МГ СО РАН. Научные интересы связаны с клеточно-автоматным моделированием и разработкой программного обеспечения для задач имитационного моделирования. Является ассистентом кафедры Параллельных вычислительных технологий в Новосибирском государственном техническом университете.

Ostapkevich M. graduated from the Novosibirsk State Technical University in 1997. Since 1999 he works at the ICM&MG SB RAS. His research interests include cellular automata simulation and the software development for simulating systems. He is an assistant of the department of Parallel computing technologies at the Novosibirsk State Technical University.

Дата поступления — 10.04.2017