**КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2010 Т. 2 № 3 С. 319–322**

Ки&М

МАТЕМАТИЧЕСКИЕ ОСНОВЫ И ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

УДК: 004

Реализация клеточных автоматов «игра “Жизнь”» и клеточного автомата Кохомото–Ооно с применением технологии MPI

А. Ю. Субботина[a], Н. И. Хохлов[b]

Московский физико-технический институт,
Россия, 141700, Долгопрудный, пер. Институтский, 9

E-mail: [a] subbotinaanna@gmail.com, [b] k_h@inbox.ru

Получено 17 сентября 2010 г.

Данная работа является анализом результатов, полученных участниками летней школы по высокопроизводительным вычислениям МФТИ-2010 во время практикума по технологии MPI. В качестве проекта была предложена трехмерная версия игры Конвея «Жизнь». Разобраны основные способы решения, используемые участниками при разработке, приведена их теоретическая и практическая оценка по масштабируемости.

Ключевые слова: MPI, игра Конвея «Жизнь», высокопроизводительные вычисления

MPI implementations of Conway’s Game of Life and Kohomoto–Oono cellular automata

A. Yu. Subbotina, N. I. Khokhlov

Moscow Institute of Physics and Technology, 9 Institutskii per, Dolgoprudny, 141700, Russia

Abstract. — Results obtained during practical training session on MPI during high perfomance computing summer school MIPT-2010 are discussed. MPI technology were one of technologies proposed to participants for realization of project. 3D version of Conway’s Game of Life was proposed as a project. Algorithms used in the development, theoretical and practical assessment of their scalability is analyzed.

Keywords: MPI, Conway’s Game of Life, HPC

Citation: *Computer Research and Modeling*, 2010, vol. 2, no. 3, pp. 319–322 (Russian).

В качестве одной из технологий распараллеливания программ в Школе слушателям предлагалась технология MPI. Желающие могли сделать практическое задание — реализовать клеточные автоматы “игра «Жизнь»” и клеточный автомат Кохомото–Ооно для случая трехмерной решетки.

MPI (Message Passing Interface) — программный интерфейс, который позволяет обмениваться информацией в форме передачи сообщений между процессами, решающими общую задачу. MPI является одним из самых распространенных стандартом интерфейса обмена данными в параллельном программировании на системах с распределенной памятью, существуют его реализации для большого числа компьютерных платформ. Часто употребляются два стандарта MPI — 1.1 и 2.0. Слушателям предлагалось пользоваться только средствами стандарта 1.1, но и его возможностей достаточно для решения большинства вычислительных задач.

Платформой для тестирования и разработки программ был учебный кластер МФТИ. Он состоит из 16 узлов, каждый узел имеет два двухъядерных процессора Intel Xeon 1.6 ГГц и 2 Гб оперативной памяти. Это позволяет выполнять расчетные задачи в 64 потока, при условии выполнения одного потока на одном ядре. В качестве управляющей сети используется Gigabit Ethernet, а в качестве быстрой расчетной сети — Myrinet 2000. Это высокоскоростная сеть с латентностью в среде MPI порядка 2.5–3 мкс и пропускной способностью 450 МБ/с. В качестве реализации MPI предоставлялось несколько библиотек, была рекомендована версия MPICH2. Использовались компиляторы gcc и Intel.

На выполнение задания слушателям было отведено два дня. Участникам был предложен один из эффективных и в тоже время довольно простых способов организации обмена данными. Поскольку критерием выполнения задачи было получение любого ускорения при работе в параллельной среде, слушатели могли предложить и другие способы решения задачи.

Рассмотрим наиболее популярные реализации параллельного способа и сравним их производительности.

Первая версия реализации была выбрана многими студентами, поскольку требовала минимального изменения кода последовательной части. При этом каждый процесс хранил данные для всей расчетной сетки, но проводил расчет только на своей части. Зоны ответственности были одинаковы по размеру для всех процессов для синхронизации работы и исключения ожиданий процессами окончания работы других процессов. Обычно сетка делилась по одной оси на число процессов. После расчета процессы пересылали свои данные всем остальным, используя процедуру группового взаимодействия MPI — MPI_Bcast. Характерное время работы этой процедуры $O(\log(n))$, где n — число процессов. В результате выполнения такого алгоритма после каждой серии пересылок каждому процессу доступна вся сетка с актуальными данными. Этот способ прост в реализации, хорошо работает из-за высокоскоростной сети с малым числом потоков за счет высокоскоростной сети, но с большим числом потоков ускорение резко падает за счет множественных коммуникаций. Сложность пересылок оценивается как $O(n \cdot \log(n))$. На рис. 1. приведен типичный график зависимости ускорения от числа потоков такого способа решения.

Во второй версии способа решения выделялся один главный процесс (master), который хранил данные для всей расчетной сетки, а другие процессы (workers) хранили данные только для части, на которой проводился расчет. После перехода на следующий шаг все процессы пересылали свои данные главному процессу, затем он рассылал процессам те части данных, которые были необходимы для перехода на следующий шаг. Этот способ решения использовал функции MPI_Send/MPI_Recv для пересылок, сложность работы пересылок — $O(n)$. Этот способ решения показал лучшие результаты, чем первый, но он и более сложен в реализации. Типичный график ускорения такого способа решения показан на рис. 2.

Третий способ решения, наиболее эффективный по ускорению из реализованных в рамках Школы, — деление сетки между процессами вдоль одной из осей, но обмен данными происходит

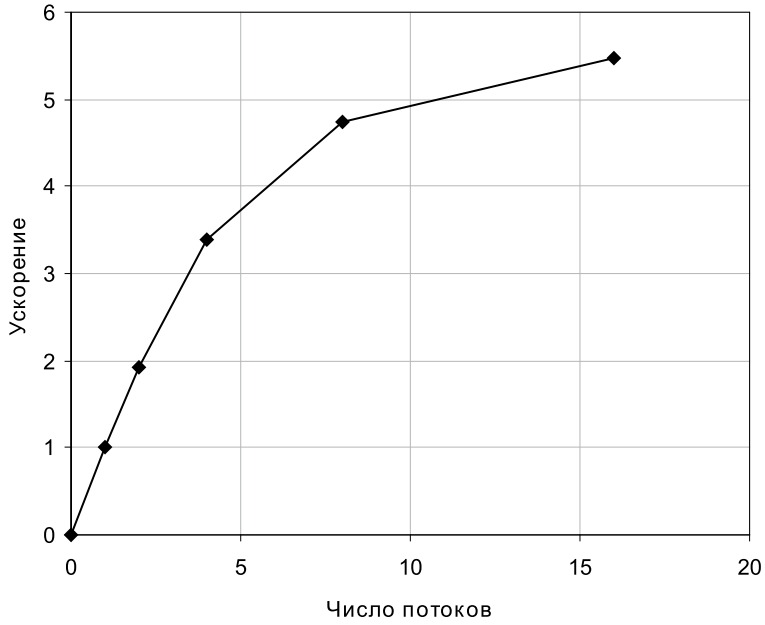

Рис. 1. Ускорение работы первого способа решения

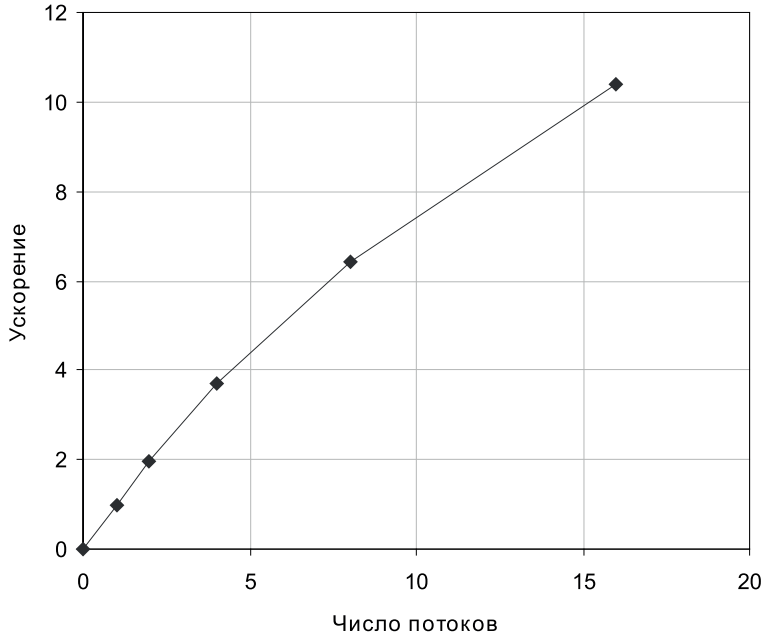

Рис. 2. График ускорения работы второго способа решения

только для граничных ячеек сетки между областями ответственности процессов. Процесс обмена был реализован с использованием процедур MPI_Sendrecv или MPI_Isend/MPI_Irecv. Сложность пересылок в таком способе решения порядка O(1), поэтому такая реализация дала лучшее ускорение. Некоторые реализации такого позволяют всем процессам хранить данные только для части расчетных сеток, что экономит оперативную память. Типичный график ускорения этого способа решения представлен на рис. 3.

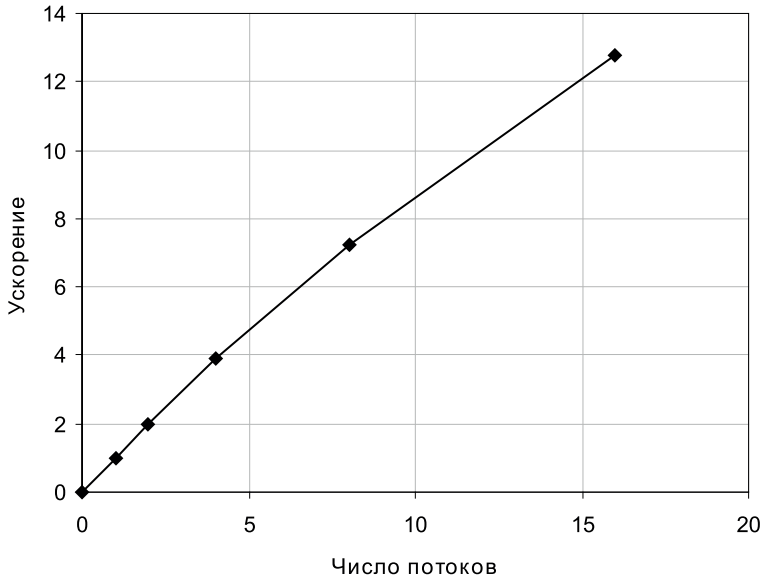

Рис. 3. График ускорения работы третьего способа решения

Наилучший результат имеет третий способ решения — ускорение в 12.8 раза при работе в 16 потоков по сравнению со временем работы одного потока, затем идет способ решения номер два — 10.8 раза, первый способ решения дал ускорение в 5.4 раза.

Несмотря на то, что всем слушателям был предложен один метод распараллеливания, большинство участников пошло своим путем. Многие способы решения показали меньшее ускорение, но они были уникальны и разработаны участниками Школы самостоятельно.

Первый способ решения был полностью реализован 4 слушателями. Второй способ решения удалось довести до работающего состояния 5 слушателям, наиболее эффективный третий способ решения реализовало 3 человека.