Моделирование плазмы методом частиц в ячейках на гетерогенных кластерных системах*

С.И. Бастраков[1], А.А. Гоносков[2], Р.В. Донченко[1],
Е.С. Ефименко[2], А.С. Малышев[1], И.Б. Мееров[1]

Нижегородский государственный университет им. Н.И. Лобачевского[1],
Институт прикладной физики РАН[2]

Ставится задача разработки программного обеспечения для моделирования плазмы на гетерогенных кластерных системах. Для решения задачи используется широко распространенный метод частиц в ячейках (Particle-in-Cell, PIC). Предлагается подход к распараллеливанию вычислительной схемы метода для кластерных систем. Рассматриваются особенности реализации метода для графических процессоров. Приводятся результаты вычислительных экспериментов, характеризующие эффективность и масштабируемость текущей реализации. Формулируются выводы и планы по дальнейшему развитию.

1. Введение

Одной из востребованных в настоящее время областей численного моделирования физических процессов является моделирование динамики плазмы и взаимодействия мощных лазерных импульсов с различными мишенями. В числе важных приложений можно выделить создание компактных источников для адронной терапии при лечении онкологических заболеваний, создание фабрик короткоживущих изотопов для биоимиджинга, разработку приборов для исследования внутримолекулярных и внутриатомных процессов.

Часто, в связи с высокой степенью нелинейности и геометрической сложностью задачи, исследование динамики плазменных структур основывается на моделировании плазмы методом частиц в ячейках (Particle-in-Cell, PIC) [1]. Основная специфика метода заключается в одновременной обработке принципиально разнородных массивов данных, содержащих информацию о считающихся непрерывными координатах и скоростях заряженных частиц плазмы, и об электромагнитном поле, определенном на пространственной сетке. Отсутствие однозначных путей упорядочивания обращений к памяти обуславливает сложность эффективной программной реализации, как для классических кластерных вычислительных систем, так и для гетерогенных систем с использованием графических процессоров. Применение метода для решения прикладных задач часто требует использования суперкомпьютерных технологий – известны задачи, требующие моделирования динамики $\sim 10^9$ и более частиц в пространстве, представленном $\sim 10^8$ ячеек.

Целью настоящей работы является представление текущих результатов, полученных в рамках проекта, направленного на поиск и реализацию наиболее эффективных подходов к параллельному моделированию плазмы методом частиц в ячейках на гетерогенных кластерных системах.

2. Постановка задачи и метод решения

Моделирование плазмы методом частиц в ячейках [1] основано на математической модели, в рамках которой плазма представляется ансамблем отрицательно заряженных электронов и положительно заряженных ионов, создающих и двигающихся под действием электромагнитных полей. Движение заряженных частиц описывается в рамках классических релятивистских

* Работа выполнена в лаборатории «Информационные технологии» ВМК ННГУ при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», госконтракт № 02.740.11.0839.

уравнений движения. Эволюция электромагнитного поля описывается в рамках уравнений Максвелла, в которые также входит векторное поле плотности тока, создаваемое частицами.

Используется классическая схема метода частиц в ячейках, каждая итерация вычислительного цикла состоит из 4 этапов: взвешивание токов, интегрирование уравнений поля, интерполяция полей, интегрирование уравнений движения частиц, и соответствует одному шагу по времени. Взвешивание токов состоит в определении сеточных значений плотности тока по известным положениям и скоростям частиц. Каждая частица вносит вклад в значение плотности тока только в 8-ми ближайших к ней узлах сетки; для получения итоговых значений вклады всех частиц суммируются [1]. Далее выполняется шаг численного интегрирования уравнений Максвелла по времени, определяются новые сеточные значения электрического и магнитного поля; используется метод FDTD [2]. На следующем этапе производится линейная интерполяция сеточных значений электрического и магнитного поля для точек нахождения частиц; для получения каждого значения используются 8 ближайших сеточных значений. Интерполированные значения полей используются при интегрировании уравнений движения частиц методом Boris [1].

3. Реализация для гетерогенных кластерных систем

Параллельная реализация разрабатывается для использования на гетерогенных кластерных системах. Рассматривается случай, когда узлы кластера содержат центральные и графические процессоры. Используется следующая схема работы:

1. Распределение задачи по узлам вычислительного кластера осуществляется при помощи интерфейса MPI.
2. На каждом из узлов кластера запускается несколько MPI-процессов. Каждый процесс использует для вычислений либо одно или несколько ядер центральных процессоров, либо один из графических процессоров (или других устройств).
3. Распараллеливание на ядра центральных процессоров внутри узла производится при помощи технологии OpenMP. Программирование графических процессоров выполнено при помощи технологии OpenCL.

3.1 Декомпозиция задачи

Декомпозиция задачи осуществляется по территориальному принципу: расчетная область разбивается на подобласти (домены), операции над которыми выполняются параллельно разными MPI-процессами. Процесс, на котором производятся операции над определенным доменом, хранит данные об электромагнитных полях и данные о частицах, находящихся в соответствующей части физического пространства. Данные, относящиеся к узлам, попадающим на границы между двумя или более доменами, хранятся во всех вычислительных процессах, обрабатывающих такие домены. Кроме того, процесс хранит данные о магнитных полях в центрах ячеек, граничащих с обрабатываемым им доменом.

Операции параллельной обработки внутренних частей доменов выполняются полностью аналогично последовательной версии. Для поддержания актуальности данных во всех доменах используются обмены данными о токах, полях и частицах. Схема организации обменов устроена следующим образом (рис. 1).

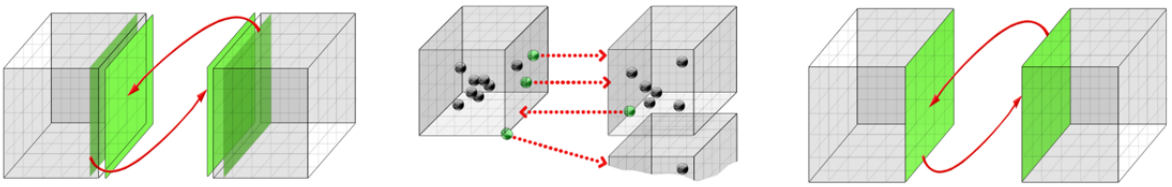

Рис. 1. Схема организации обменов (поля, частицы, токи)

При обменах токами и полями каждый домен взаимодействует с 6 соседями, при обменах частицами – с 26 соседями (в случае, когда частица покидает пределы домена, она попадает в

один из 26 соседних доменов). Таким образом, все обмены данными осуществляются между процессами, обрабатывающими соседние домены.

3.2 Особенности реализации для GPU

В данном разделе будет использоваться терминология OpenCL. Эксперименты проводились с использованием GPU NVIDIA, приведем соответствие терминологии OpenCL и NVIDIA CUDA: локальная память в OpenCL соответствует разделяемой (shared) в CUDA, элемент работы и группа работ – потоку/нити (thread) и блоку потоков/нитей (thread block).

Данные вычислительных экспериментов показывают, что основное время работы распределяется между этапами интегрирования уравнений движения частиц и взвешивания токов. Оба этапа оперируют с частицами, поэтому ключевым является способ представления и обработки данных о частицах на GPU. Важной оптимизацией при работе с глобальной памятью GPU является обеспечение коалесинга (coalescing) [5], в данном случае это обозначает обработку соседних по расположению в памяти частиц соседними (по индексам) элементами работы в одной группе работ.

Традиционным способом упорядочивания обращений к памяти при реализации метода частиц в ячейках (как на CPU, так и на GPU) является группировка частиц по ячейкам. Естественным решением при реализации на GPU является обработка частиц в одной ячейке отдельной группой работ. Возможной проблемой является недостаточная вычислительная нагрузка на группу работ – для ряда задач, в том числе и целевых задач разрабатываемого ПО, среднее количество частиц в ячейке составляет десятки, в то время как для эффективной загрузки современных GPU желательный размер группы работ составляет не менее нескольких сотен [5].

Для решения данной проблемы используется подход, близкий к предложенному в [4]. Соседние ячейки группируются в суперячейки, все суперячейки имеют равный размер, кратный размеру ячейки (например, 2 ячейки по каждой оси). Частицы сгруппированы в памяти по суперячейкам, все частицы суперячейки обрабатываются одной группой работ с интенсивным использованием локальной памяти. При интегрировании уравнений движения частиц локальная память используется как ручной кэш для загрузки из глобальной памяти значений полей, необходимых для интерполяции, при взвешивании токов локальная память используется для накопления промежуточных результатов. Благодаря этому обеспечивается высокая загрузка устройства и эффективный паттерн доступа к памяти GPU.

Отметим, что количество разделяемой памяти в современных GPU NVIDIA (до 48 КБ) накладывает существенные ограничения на размер суперячейки и размер группы работ: например, при использовании группы работы размера 256 в двойной точности, размер суперячейки не может превышать 2 ячейки по каждой оси. Вероятно, использование групп работ и суперячеек большего размера (при наличии большего количества памяти) позволило бы еще более эффективно задействовать GPU.

4. Результаты экспериментов

Для анализа корректности реализации метода частиц в ячейках был использован набор тестовых задач, соответствующие результаты приведены в [3]. В данном разделе приводятся результаты экспериментов, связанные с оценкой эффективности и масштабируемости реализации для традиционных и гетерогенных кластерных систем.

Вычислительные эксперименты проводились на следующих системах:

1. МВС-100К в МСЦ РАН (на каждом узле 2 CPU Intel Xeon E5450, 8 ГБ RAM, Infiniband DDR, CentOS 5.6 x86_64).
2. Akka в High Performance Computing Center North (на каждом узле 2 CPU Intel Xeon L5420, 16 ГБ RAM, Infiniband 4x, CentOS 5.6 x86_64).
3. Кластер ННГУ (на каждом узле 2 CPU Intel Xeon L5630, 2 GPU NVIDIA Tesla X2070, 24 ГБ RAM, Infiniband QDR, Windows Server 2008 HPC Edition SP2 x64).
4. Суперкомпьютер «Ломоносов» в НИВЦ МГУ (на каждом узле 2 CPU Intel Xeon E5630, 24 ГБ RAM, 2x NVIDIA Tesla X2070, Infiniband QDR, Clustrx CNL v1.1 x86_64).

В серии экспериментов по анализу масштабируемости рассматривалась тестовая задача моделирования Ленгмюровских колебаний плазмы с использованием 503 миллионов частиц и 17 миллионов ячеек пространственной сетки, двойная точность. На каждое ядро создавался отдельный MPI-процесс. Результаты для систем МВС-100К и Akka приведены на рис. 2. При использовании *2048 ядер* на рассматриваемых системах достигается *эффективность 71% и 85% относительно 512 ядер*.

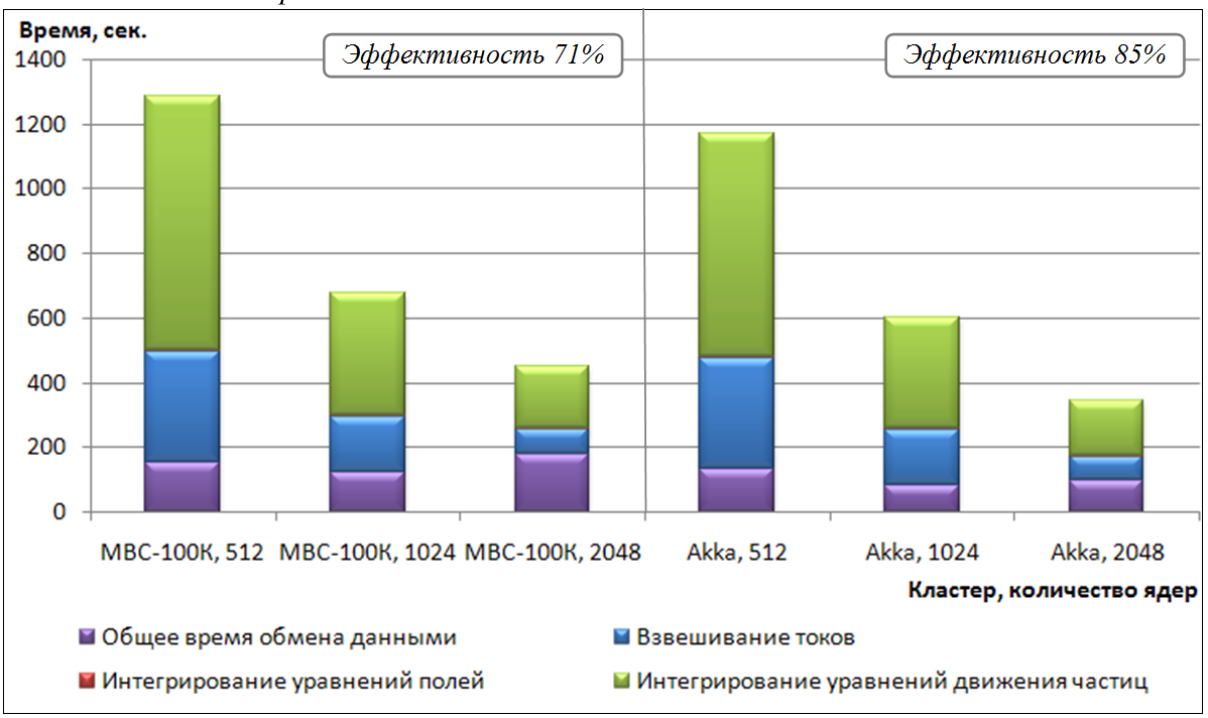

Рис. 2. Масштабируемость на кластерных системах

В серии экспериментов по сравнению производительности CPU и GPU на одном узле решалась задача моделирования Ленгмюровских колебаний плазмы с использованием 983 тысяч частиц, 33 тысяч ячеек пространственной сетки. Результаты для одинарной и двойной точности, полученные на кластере ННГУ, приведены на рис. 3, времена обменов между оперативной памятью и памятью GPU включены во времена соответствующих этапов вычислений. В двойной точности при использовании 1 GPU достигает ускорения около *3,1* раз относительно версии с использованием 8 ядер CPU, при использовании 2 GPU – около *4,7* раз. В одинарной точности ускорения при использовании 1 и 2 GPU относительно 8 ядер CPU составляют *4,3* и *5,8* раз соответственно.

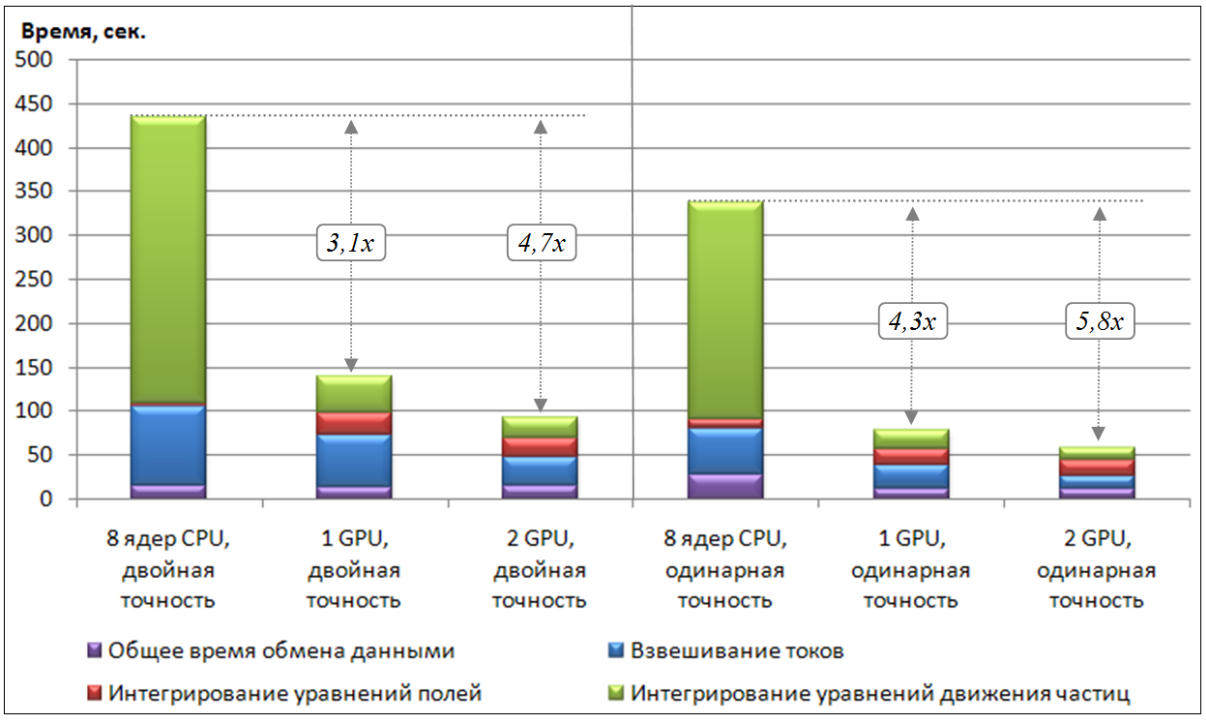

Рис. 3. Сравнение производительности CPU- и GPU-версии на кластере ННГУ

В другой серии экспериментов производилось измерение слабой масштабируемости – вычислительная трудоемкость задачи масштабировалась пропорционально количеству используемых вычислительных устройств (GPU либо ядер CPU). Трудоемкость выполнения одной итерации вычислительного цикла пропорциональна сумме количества частиц и общего количества узлов пространственной сетки. При увеличении количества используемых устройств в N раз, размер расчетной области увеличивался в N раз вдоль одной из осей и оставался неизменным вдоль других, пространственная плотность частиц оставалась неизменной (таким образом, общее количество частиц также увеличивалось в N раз). В качестве базовой выступала задача моделирования Ленгмюровских колебаний плазмы, двойная точность. Результаты с использованием только CPU, полученные на системе Akka, приведены на рис. 4а. Результаты с использованием только GPU, полученные на системе «Ломоносов», приведены на рис. 4б.

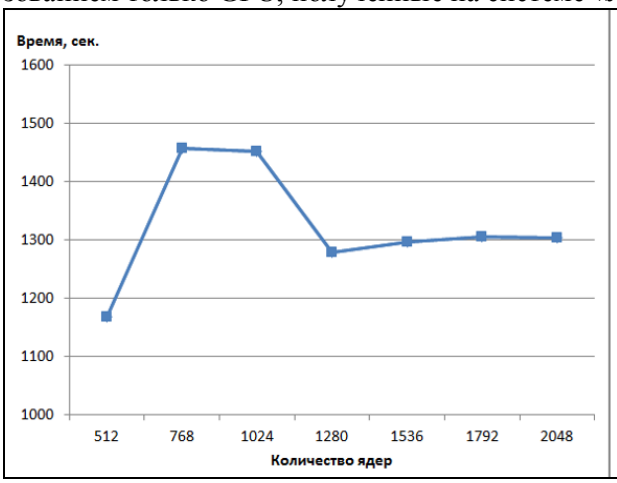

Рис. 4а. Слабая масштабируемость CPU-версии на системе Akka

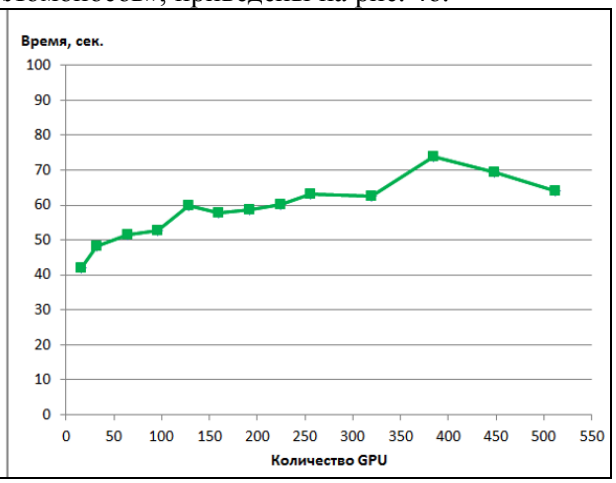

Рис. 4б. Слабая масштабируемость GPU-версии на системе «Ломоносов»

5. Заключение

В работе получены следующие результаты:

1. Разработана параллельная версия программного обеспечения для численного моделирования плазмы методом частиц в ячейках на гетерогенных кластерных системах. При решении модельных задач данная реализация показывает результаты, соответствующие теоретическим предсказаниям.
2. Проведенные вычислительные эксперименты демонстрируют масштабируемость параллельной версии для традиционных кластерных систем по крайней мере до 2048 ядер. При решении одной и той же задачи эффективность относительно запуска для 512 ядер составляет 71% и 85% на системах МВС-100К и Akka соответственно. При увеличении вычислительной трудоемкости задачи пропорционально количеству используемых ядер до 1280 время работы растет на ~11-12% и стабилизируется на этом уровне по крайней мере до 2048 ядер.
3. Разработанная версия для GPU на кластере ННГУ демонстрирует ускорение в 3-4 раза относительно 8 ядер CPU. Параллельная версия для кластера, содержащего графические процессоры, масштабируется следующим образом: при увеличении вычислительной трудоемкости задачи пропорционально количеству используемых GPU до 128 время работы растет на ~50% и стабилизируется на этом уровне по крайней мере до 320 GPU (143360 CUDA-ядер). Далее при увеличении количества используемых GPU наблюдается некоторая деградация производительности, однако при достижении 512 GPU (229376 CUDA-ядер) время работы вновь приближается к достигнутому ранее уровню (320 GPU).

Целью дальнейших исследований является увеличение производительности и масштабируемости на CPU и GPU, одновременное использование всех вычислительных устройств, а также реализация улучшенных численных схем.

Литература

1. Бэдсел Ч., Ленгдон А. Физика плазмы и численное моделирование: Пер. с англ. – М.: Энергоатомиздат, 1989. – 452 с.
2. Taflove A. Computational Electrodynamics: The Finite-Difference Time-Domain Method. – London: Artech House, 1995. – 599 P.
3. Бастраков С.И., Гоносков А.А., Донченко Р.В., Ефименко Е.С., Малышев А.С., Мееров И.Б. Исследование и поиск наиболее эффективных подходов к параллельному моделированию плазмы методом частиц в ячейках на кластерных системах // Параллельные вычислительные технологии (ПаВТ’2011): труды международной научной конференции (Москва, 28 марта – 1 апреля 2011 г.) [Электронный ресурс] – Челябинск: Издательский центр ЮУрГУ, 2011. – С. 411-417. – URL: [http://omega.sp.susu.ac.ru/books/conference/PaVT2011/short/118.pdf].
4. Hoenig W., Schmitt F., Widera R., Burau H,, Juckeland G., Mueller M., Bussmann M. A Generic Approach for Developing Highly Scalable Particle-Mesh Codes for GPUs // SAAHPC, USA, Jul. 2010. – 15pp. – URL: [http://saahpc.ncsa.illinois.edu/10/papers/paper_10.pdf].
5. CUDA C Best Practices Guide v. 4.0. – URL: [http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Best_Practices_Guide.pdf].