УДК 004.942

МОДЕЛИРОВАНИЕ ПЛАЗМЫ МЕТОДОМ ЧАСТИЦ В ЯЧЕЙКАХ С ИСПОЛЬЗОВАНИЕМ СОПРОЦЕССОРОВ INTEL XEON PHI

И. А. Сурмин[1], С. И. Бастраков[2], А. А. Гоносков[3], Е. С. Ефименко[4], И. Б. Мееров[5]

Рассматриваются вопросы высокопроизводительной реализации метода частиц в ячейках для моделирования лазерной плазмы. Используется программный комплекс PICADOR. Обсуждается возможность эффективного применения новых сопроцессоров Intel Xeon Phi. Показывается, что хорошо оптимизированный для обычных процессоров код не всегда работает эффективно на сопроцессоре без дополнительных улучшений. Демонстрируются пути достижения приемлемой производительности при численном моделировании плазмы на сопроцессоре. Приводятся результаты вычислительных экспериментов, показывающие ускорение в 1.8 раза на Xeon Phi по сравнению с кодом, оптимизированным для многоядерного процессора.

Ключевые слова: физика плазмы, метод частиц в ячейках, высокопроизводительные вычисления, Xeon Phi, оптимизация производительности.

1. Введение. Актуальные задачи современной физики все чаще требуют проведения ресурсоемкого численного моделирования в качестве единственно возможного средства изучения динамики сложных систем. Для этого необходимо создание специализированного программного обеспечения, достигающего высокой производительности и масштабируемости на современных суперкомпьютерах. В физике плазмы общепризнанным методом исследования является моделирование методом частиц в ячейках [1]. Существующие и перспективные задачи требуют моделирования систем, включающих в себя более 10^9 частиц и состоящих из более 10^8 узлов пространственной сетки. К настоящему моменту существуют несколько эффективных реализаций метода частиц в ячейках, ориентированных на традиционные и гетерогенные кластерные системы: OSIRIS [2], VLPL [3], VPIC [4], PIConGPU [5] и др.

Программный комплекс PICADOR [6, 7] разрабатывается коллективом сотрудников ННГУ и ИПФ РАН с 2010 г. Этот комплекс ориентирован на решение больших задач в области моделирования лазерной плазмы на гетерогенных кластерных системах и демонстрирует сравнимую с аналогами производительность и масштабируемость [8]. Отличительной особенностью комплекса PICADOR является ориентация на широкий класс вычислительных устройств: кластерные системы с многоядерными CPU и GPU на узле. В настоящей статье рассматривается использование сопроцессоров Intel Xeon Phi.

Сопроцессоры Xeon Phi, появившиеся в 2012 г., ориентированы на решение научных и инженерных задач [9]. Особенностью такого сопроцессора является использование большого количества “легковесных” ядер x86-совместимой архитектуры. В отличие от GPU, поддерживаются традиционные языки и технологии параллельного программирования, включая MPI и OpenMP, что существенно упрощает портирование существующих приложений. Вместе с тем, в ряде задач эффективность такого портирования невысока. В связи с этим большой практический интерес представляет выработка подходов к оптимизации кода для Xeon Phi. Основная сложность в достижении высокой производительности на Xeon Phi состоит в обеспечении хорошей масштабируемости на 60 ядер на общей памяти (до 240 потоков) и эффективном исполь-

[1] Нижегородский государственный университет им. Н. И. Лобачевского (ННГУ), факультет вычислительной математики и кибернетики, просп. Гагарина, 23, 603950, г. Нижний Новгород; мл. науч. сотр., e-mail: i.surmin@gmail.com

[2] Нижегородский государственный университет им. Н. И. Лобачевского (ННГУ), факультет вычислительной математики и кибернетики, просп. Гагарина, 23, 603950, г. Нижний Новгород; ассистент, e-mail: sergey.bastrakov@gmail.com

[3] Институт прикладной физики РАН (ИПФ РАН), ул. Ульянова, 46, 603950, г. Нижний Новгород; науч. сотр., e-mail: arkady.gonoskov@gmail.com

[4] Институт прикладной физики РАН (ИПФ РАН), ул. Ульянова, 46, 603950, г. Нижний Новгород; мл. науч. сотр., e-mail: nnreene@mail.ru

[5] Нижегородский государственный университет им. Н. И. Лобачевского (ННГУ), факультет вычислительной математики и кибернетики, просп. Гагарина, 23, 603950, г. Нижний Новгород; зам. зав. каф., e-mail: meerov@vmk.unn.ru

зовании векторных операций при увеличенной вдвое по сравнению с AVX (Advanced Vector Extensions) длиной векторного регистра.

В настоящей работе предлагается способ оптимизации метода частиц в ячейках для сопроцессоров Xeon Phi. Основное внимание уделяется вопросам улучшения масштабируемости при большом количестве ядер, обеспечения локальности доступа к памяти и эффективного использования векторных операций. Применение рассматриваемых оптимизаций также приводит к повышению производительности и на CPU, но в меньшей степени по сравнению с Xeon Phi. Предлагаемая реализация для Xeon Phi является одной из первых в мире.

Статья имеет следующую структуру: вычислительная схема метода частиц в ячейках приводится в разделе 2, раздел 3 содержит краткое описание программного комплекса PICADOR, раздел 4 посвящен описанию архитектуры и средств программирования сопроцессоров Xeon Phi; портирование и оптимизация кода для Xeon Phi описываются в разделе 5.

2. Постановка задачи и вычислительная схема. В данном разделе приводится краткое описание вычислительной схемы метода частиц в ячейках, подробное описание метода содержится в [1]. Область моделирования имеет форму прямоугольного параллелепипеда со сторонами, параллельными осям координат. В расчетной области заданы электрическое и магнитное поля $\boldsymbol{E}$ и $\boldsymbol{B}$, динамика электромагнитного поля описывается системой уравнений Максвелла. В методе частиц в ячейках плазма моделируется набором из N заряженных квазичастиц, каждая из которых характеризуется переменными импульсом $\boldsymbol{p}$ и координатами $\boldsymbol{r}$, а также постоянными массой m и зарядом q. Координаты и скорость частиц $\boldsymbol{v}$ меняются согласно второму закону Ньютона в релятивистской формулировке. Движение заряженных частиц создает плазменные токи $\boldsymbol{j}$, которые входят в качестве источников в уравнения Максвелла, замыкая таким образом самосогласованную систему уравнений.

Вычислительная схема метода с основными уравнениями и схемой зависимости по данным приведена на рис. 1. Начальные условия состоят из значений электрического и магнитного полей, а также координат и скоростей частиц в начальный момент времени. На каждой итерации по времени по текущим данным частиц и полей вычисляется состояние системы в следующий момент времени. Каждая итерация состоит из 4 основных этапов: интегрирование уравнений Максвелла; интерполяция полей в точке нахождения каждой частицы и вычисление силы Лоренца, действующей на частицу; интегрирование уравнений движения частиц; вычисление (взвешивание) токов. При программной реализации вычисление силы Лоренца и интегрирование уравнений движения обычно объединяются в целях повышения производительности, в дальнейшем изложении эти этапы иногда объединяются под названием “движение частицы”.

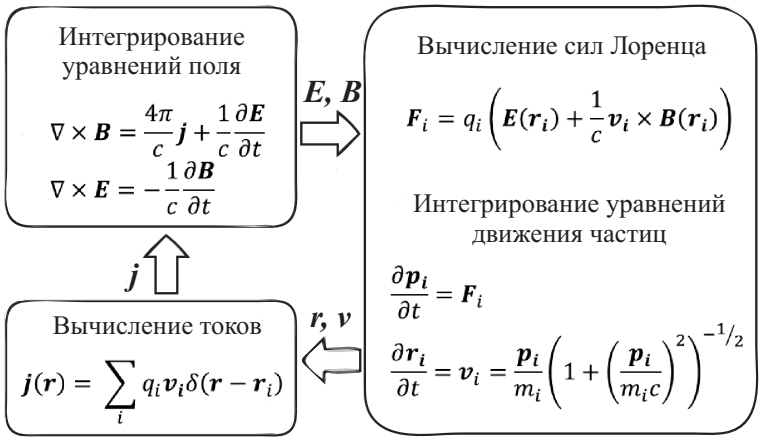

Рис. 1. Вычислительная схема метода частиц в ячейках. Итерация цикла соответствует моделированию одного шага по времени. Уравнения приведены в системе СГСЭ. Надписи над стрелками показывают зависимости по данным между этапами метода

Метод частиц в ячейках оперирует двумя принципиально разнородными наборами данных: ансамбль заряженных частиц с непрерывно изменяющимися координатами и сеточные значения электромагнитного поля и плотности тока. Этапы интерполяции полей и взвешивания токов являются связывающими, а их реализация имеет определяющее значение как с точки зрения корректности и точности схемы, так и с точки зрения производительности и масштабируемости реализации.

3. Программный комплекс PICADOR. Программный комплекс PICADOR предназначен для трехмерного численного моделирования плазмы на гетерогенных кластерных системах. PICADOR основан на методе частиц в ячейках с многочисленными расширениями для повышения точности и учета дополнительных физических эффектов. В настоящее время поддерживаются конечно-разностные схемы FDTD (Finite Difference Time Domain) [10] и NDF (Numerical Differentiation Formulas) [3] для численного интегрирования уравнений поля, генерация лазерного импульса на границе, периодические и поглощающие граничные условия [11], метод Бориса для интегрирования уравнений движения [1], формфакторы

частиц первого и второго порядка [1], схема взвешивания токов Езиркепова [12], бегущее окно, а также динамическая балансировка нагрузки [7].

Программный код PICADOR написан на языке программирования C++, поддерживается сборка под Windows и POSIX-совместимые системы (POSIX: англ. Portable Operating System Interface for Unix). Работа на кластерных системах осуществляется через интерфейс передачи сообщений MPI. На уровне одного узла реализован параллелизм на общей памяти с помощью технологии OpenMP и использование GPU с помощью CUDA. Общий объем кода составляет около 20 000 строк.

4. Сопроцессор Xeon Phi. Сопроцессор Intel Xeon Phi реализован в виде отдельной платы, подключаемой через слот PCI Express (Peripheral Component Interconnect Express). Сопроцессор содержит 60 ядер (в некоторых модификациях 61 ядро), соединенных высокоскоростной кольцевой шиной. Каждое ядро является полнофункциональным, но облегченным по сравнению с обычными CPU, в частности не поддерживаются предсказание ветвлений и внеочередное выполнение. Xeon Phi поддерживает выполнение векторных инструкций с 512-битными векторами по сравнению со 128- или 256-битными на CPU. Производительность одного ядра Xeon Phi меньше, чем одного ядра современных CPU, но сочетание множества таких ядер с высокопроизводительной шиной данных предоставляет бóльшую, по сравнению с традиционными процессорами, пиковую производительность и дает широкие возможности для использования параллелизма в научных и инженерных приложениях.

Xeon Phi поддерживает популярные языки и средства параллельного программирования: языки программирования C, C++, Fortran, технологии MPI, OpenMP, Cilk Plus, OpenCL, библиотеки TBB (Threading Building Blocks) и MKL (Intel Math Kernel Library), что упрощает портирование существующих кодов и разработку новых. Разработка приложений для Xeon Phi предполагает выбор одного из трех режимов: исполнения только на Xeon Phi (native), исполнения головной программы на CPU с вызовом вычислительных ядер на сопроцессоре (offload) или одновременное выполнение MPI-процессов на процессоре и сопроцессоре (symmetric) [9].

Анализ архитектуры Xeon Phi позволяет сделать выводы, приведенные ниже.

1. Xeon Phi сочетает значительное число ядер с большими векторными регистрами и высокопроизводительную шину данных, что дает широкие возможности для использования параллелизма. Для эффективного решения многих задач потребуется разработка новых вычислительных схем, ориентированных на существенное число параллельно работающих потоков и векторную обработку данных. Игнорирование этих аспектов может привести к катастрофически низкой производительности.

2. Ядра Xeon Phi имеют архитектуру, совместимую с x86-архитектурой, что позволяет запускать на сопроцессоре уже существующий код без его переписывания с использованием новых технологий и средств программирования. Вместе с тем, эффективная работа не гарантирована, и для многих приложений может понадобиться оптимизация кода.

3. Подходы к оптимизации кода для CPU и Xeon Phi в целом похожи; следовательно, в большинстве случаев оптимизация производительности для Xeon Phi будет приводить к выигрышу и на CPU, и наоборот. Это позволяет разработчикам использовать весь имеющийся опыт распараллеливания и оптимизации.

5. Использование сопроцессоров Xeon Phi в коде PICADOR.

5.1. Бенчмарк. Для анализа производительности кода PICADOR использовалась тестовая задача моделирования холодной плазмы с сеткой $40 \times 40 \times 40$ ячеек и 50 частицами на ячейку, 1000 итераций по времени. Использовались схемы интерполяции и взвешивания первого порядка, арифметика с плавающей запятой двойной точности. Время обновления полей составляло менее 1% от времени работы вычислительной части; в связи с этим приводится только время движения частиц и взвешивания токов. В качестве метрики измерения производительности использовалось количество наносекунд на обработку частицы на одной итерации. Используемый бенчмарк и способ измерения производительности являются широко применяемыми для кодов моделирования плазмы, например [2, 13]. Лучший известный авторам результат на аналогичном бенчмарке достигается кодом OSIRIS и составляет 7.4 нс/частицу на 8-ядерном Intel Xeon E5-2680 [2]. В работе [13] приводится результат 105.5 нс/частицу на одном ядре Intel Xeon X5650 для двумерного кода.

Эксперименты проводились на узле кластера ННГУ “Лобачевский” с процессором Intel Xeon E5-2660 с частотой 2.2 ГГц (Sandy Bridge, 8 ядер, 20 МБ кэш-памяти, поддержка AVX) и сопроцессором Intel Xeon Phi 5110P (60 ядер, 240 потоков, 8 ГБ ОЗУ). Пиковая производительность используемого CPU составляет 140 GFlops, пиковая производительность Xeon Phi — 1 TFlops. Использовался компилятор Intel.

5.2. Базовая версия. Операции типа частица–сетка, выполняемые при интерполяции поля и взвешивании токов, являются пространственно локальными. Хорошо известно [2–5, 13], что для эффективной

реализации метода частиц в ячейках необходимо использовать эту физическую локальность для достижения локальности шаблона доступа к памяти. В коде PICADOR для каждой ячейки хранится отдельный массив частиц, содержащихся в данной ячейке. Обход частиц происходит по ячейкам, при этом достигается локальность доступа к памяти как по частицам, так и по сеточным значениям.

Так как Xeon Phi является многоядерным устройством, для приемлемой производительности необходима масштабируемость кода на большое число потоков. Поэтому перед портированием кода на Xeon Phi были предприняты меры по улучшению масштабируемости на CPU. При взвешивании токов первого порядка каждая частица вносит вклад в 8 ближайших сеточных значений тока, вклады всех частиц суммируются. Из-за необходимости суммирования данная операция не является независимой по данным. В предыдущей версии PICADOR использовалась схема, при которой каждый поток хранил собственный набор сеточных значений тока и в процессе параллельного обхода частиц записывал в него частичные результаты, по окончании взвешивания вклады всех потоков суммировались. Такая схема плохо применима для Xeon Phi из-за гораздо большего по сравнению с CPU количества потоков. Была реализована новая параллельная схема взвешивания: параллельно обрабатываются ячейки, расположенные в шахматном порядке с шагом в две ячейки по каждой из трех осей. Таким образом, обход всех ячеек разбивается на 27 обходов в шахматном порядке. Распараллеливание производится на уровне внутреннего обхода, так как на этом уровне частицы разных ячеек не могут вносить вклад в ток одного и того же узла, что избавляет от необходимости синхронизации.

Аналогичный подход применяется при распараллеливании миграции частиц между ячейками. Для каждой ячейки вводится дополнительный буфер для вновь прибывающих частиц. После вычисления новых координат частицы производится проверка, покидает ли она текущую ячейку; в этом случае она записывается в буфер для новой ячейки. Благодаря тому, что за один шаг частица не может пролететь расстояние, большее размера ячейки, гонки данных невозможны и единственная синхронизация производится по окончании итерации внешнего обхода.

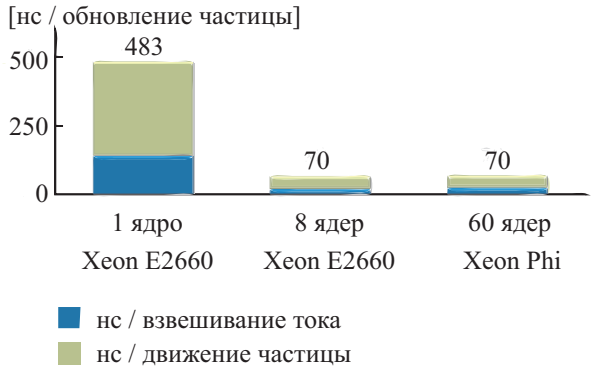

Рис. 2. Производительность базовой версии

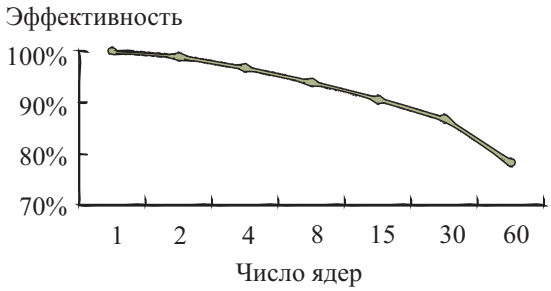

Рис. 3. Эффективность масштабируемости базовой версии на Xeon Phi

Для использования Xeon Phi был выбран режим исполнения только на сопроцессоре, так как доля последовательного кода в расчетной части невелика. Портирование базовой версии заключалось в перекомпиляции кода и используемых сторонних библиотек с опцией компилятора для использования Xeon Phi (–mmic). На одном Xeon Phi время работы составляет 70 нс/частицу, на одном 8-ядерном CPU достигается аналогичное время (рис. 2). За счет использования эффективных схем распараллеливания код имеет хорошую масштабируемость на Xeon Phi с эффективностью 78% на 60 ядрах, используется 4 потока на ядро (рис. 3). Эффективность масштабируемости с 1 до 8 ядер на CPU составляет 99%.

Основными причинами плохой производительности на CPU и Xeon Phi являются неэффективное использование памяти и большая доля скалярного кода. Тем не менее, портирование без дополнительных усилий и адаптации кода позволило получить производительность на Xeon Phi, близкую к CPU.

5.3. Использование локальности данных. Способ хранения частиц оказывает существенное влияние на производительность и специфику производимых оптимизаций. Для достижения хорошей производительности как на CPU, так и на Xeon Phi, код должен эффективно использовать кэш-память. С этой целью в PICADOR частицы хранятся по ячейкам. При интерполяции полей и взвешивании токов для каждой частицы в ячейке необходимо обращаться к одним и тем же данным, которые хранятся в глобальном массиве сеточных значений. Для интерполяции полей используются соседние сеточные значения полей по каждому измерению, но в памяти эти значения хранятся со значительным смещением. Поэтому для

улучшения локальности имеет смысл сгруппировать используемые данные. Для реализации этой идеи на этапах интерполяции и взвешивания были введены локальные массивы. Перед началом обработки частиц в ячейке необходимые для интерполяции значения поля предзагружаются в локальные массивы $3 \times 3 \times 3$. Далее интерполяция производится только с использованием локальных массивов.

Аналогичный подход применяется для взвешивания: вклады частиц в токи узлов ячейки предварительно накапливаются в локальном массиве. По окончании обхода частиц в ячейке элементы локального массива добавляются в глобальный массив токов. Данная оптимизация позволила улучшить время работы на CPU в 3.7 раза, до 18.8 нс/частицу, и на Xeon Phi в 4.5 раза, до 15.7 нс/частицу (рис. 4).

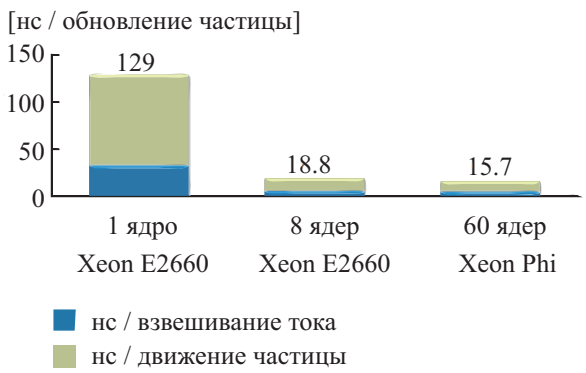

Рис. 4. Производительность оптимизированной версии с использованием локальности данных

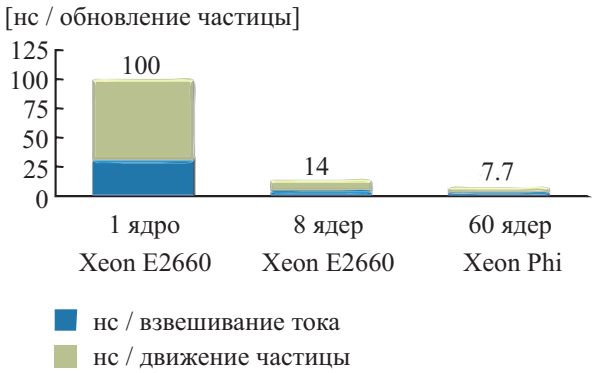

Рис. 5. Производительность оптимизированной версии с использованием локальности данных и с векторизацией кода

5.4. Векторизация кода. Векторизация заключается в одновременном выполнении одной и той же инструкции над несколькими операндами. Частным случаем векторизации является векторизация циклов, когда одновременно выполняются операции на соседних итерациях цикла. Некоторые компиляторы способны автоматически векторизовывать циклы простой структуры. Для этого цикл должен содержать известное количество итераций, не содержать условий и зависимостей по данным между итерациями, все вызовы функций должны быть встроены. На процесс векторизации можно воздействовать с помощью директив компилятора (например, #pragma vector always и #pragma ivdep для компилятора Intel).

Первой попыткой векторизации было использование директив компилятора для векторизации циклов обновления полей и интегрирования уравнений движения частиц. Однако выполненная таким образом векторизация не привела к повышению производительности на Xeon Phi. Это связано с тем, что частицы хранились в виде массива структур и накладные расходы на загрузку данных, лежащих в памяти непоследовательно, нивелировали выигрыш от векторизации. Более предпочтительным является хранение частиц в виде структуры массивов, которое позволяет загружать последовательный участок памяти в векторный регистр одной инструкцией. Замена способа хранения на структуру массивов позволила эффективно векторизовать интегрирование уравнений движения частиц на Xeon Phi.

Однако векторизация циклов интерполяции поля и взвешивания токов все равно приводила к замедлению этих этапов в несколько раз. Из-за особенностей сетки для поля и токов используемые для интерполяции и взвешивания сеточные значения зависят от положения частицы внутри ячейки. Косвенная адресация по трем измерениям привела к генерированию компилятором большого количества инструкций scatter/gather и существенным накладным расходам. Поэтому предлагается векторизовать интерполяцию не по итерациям цикла обработки частиц, а по компонентам поля. Для этого компоненты помещаются в векторный регистр и в дальнейшем обрабатываются векторными инструкциями. Недостаток такого подхода состоит в том, что компилятор плохо векторизует блоки кода, не являющиеся циклом. В связи с этим была разработана реализация интерполяции поля и взвешивания токов с использованием интринсиков.

Интринсики являются специальными функциями в языке C, инкапсулирующими ассемблерные инструкции из векторных наборов команд MMX (MultiMedia Extensions), SSE (Streaming SIMD Extensions), AVX и др. В сочетании с дополнительно определенными типами данных они ориентированы на использование некоторых возможностей процессора без явного программирования на ассемблере. Оптимизирующие компиляторы способны векторизовывать вычисления в циклах при соблюдении ряда условий, но не всегда оптимальным образом. Напротив, разработка кода с использованием ассемблерных вставок существенно понижает его читаемость. Применение интринсиков является некоторым промежуточным решением: с одной стороны, сохраняет за программистом полный контроль над кодом, а с другой — поз-

воляет по-прежнему программировать на языке C. Разумеется, это решение, как и в случае ассемблерных вставок, плохо влияет на переносимость кода, но для достижения требуемой производительности иногда приходится идти на подобные жертвы.

Так как наборы векторных инструкций для Xeon Phi отличаются от используемых в CPU, версия с интринсиками была разработана только для Xeon Phi. За счет векторизации удалось улучшить время работы на CPU в 1.3 раза, до 14 нс/частицу, и на Xeon Phi — в 2 раза, до 7.7 нс/частицу (рис. 5). Версия кода для Xeon Phi демонстрирует ускорение в 1.8 раз по сравнению с CPU.

6. Заключение. В настоящей статье предложена оптимизация программной реализации метода частиц в ячейках для архитектуры Intel Xeon Phi. Разработка выполнена в контексте программного комплекса PICADOR и находится в стадии опытной эксплуатации. Выполненные оптимизации в основном затронули версию кода для систем с общей памятью.

Проведенная работа показала, что простая пересборка кода для Xeon Phi не всегда приводит к хорошим результатам, так как реализации, изначально ориентированные на 8–16 потоков и длину векторного регистра 128–256 бит, могут плохо масштабироваться на 60–240 потоков и векторные регистры размером 512 бит. Применяемые стандартные способы оптимизации — улучшение локальности работы с памятью и векторизация — обычно приводят к выигрышу как на CPU, так и на Xeon Phi, но степень влияния может существенно отличаться. В результате проделанных оптимизаций ускорение относительно базовой версии составляет 5 раз на CPU и 9 раз на Xeon Phi. Достигнута эффективность 99% на 8 ядрах CPU и 78% на 60 ядрах Xeon Phi. Ускорение на Xeon Phi относительно CPU составляет 1.8 раза. На CPU достигается показатель 14 нс/частицу, что сравнимо с ведущими реализациями метода частиц в ячейках. Реализация для Xeon Phi является одной из первых в мире.

Работа выполнена в лаборатории ННГУ-Intel “Информационные технологии” при поддержке гранта РФФИ № 14–07–31211. Статья рекомендована к публикации Программным комитетом Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: многообразие суперкомпьютерных миров” (http://agora.guru.ru/abrau2014).

СПИСОК ЛИТЕРАТУРЫ

1. *Бэдсел Ч., Ленгдон А.* Физика плазмы и численное моделирование. М.: Энергоатомиздат, 1989.
2. *Fonseca R.A., Vieira J., Fiuza F., Davidson A., Tsung F.S., Mori W.B., Silva L.O.* Exploiting multi-scale parallelism for large scale numerical modelling of laser wakefield accelerators // Plasma Physics and Controlled Fusion. 2013. **55**, N 12. doi:10.1088/0741-3335/55/12/124011.
3. *Pukhov A.* Three-dimensional electromagnetic relativistic particle-in-cell code VLPL (Virtual Laser Plasma Lab) // Journal of Plasma Physics. 1999. **61**, N 3. 425–433.
4. *Bowers K.J., Albright B.J., Yin L., Daughton W., Roytershteyn V., Bergen B., Kwan T.J.T.* Advances in petascale kinetic plasma simulation with VPIC and Roadrunner // J. Phys.: Conf. Ser. 2009. **180**, N 1. doi:10.1088/1742-6596/180/1/012055.
5. *Burau H., Widera R., Hönig W., et al.* PIConGPU: a fully relativistic particle-in-cell code for a GPU cluster // IEEE Transactions on Plasma Science. 2010. **33**, N 10. 2831–2839.
6. *Bastrakov S., Donchenko R., Gonoskov A., Efimenko E., Malyshev A., Meyerov I., Surmin I.* Particle-in-cell plasma simulation on heterogeneous cluster systems // Journal of Computational Science. 2012. **3**, N 6. 474–479.
7. *Бастраков С.И., Мееров И.Б., Сурмин И.А., Гоносков А.А., Ефименко Е.С., Малышев А.С., Ширяев М.А.* Динамическая балансировка в коде PICADOR для моделирования плазмы // Вычислительные методы и программирование. 2013. **14**. 67–74.
8. *Bastrakov S., Meyerov I., Surmin I., Efimenko E., Gonoskov A., Malyshev A., Shiryaev M.* Particle-in-cell plasma simulation on CPUs, GPUs and Xeon Phi coprocessors // Lecture Notes in Computer Science. Vol. 8488. Heidelberg: Springer, 2014. 513–514.
9. *Jeffers J., Reinders J.* Intel Xeon Phi coprocessor high performance programming. Waltham: Morgan Kaufmann, 2013.
10. *Taflove A.* Computational electrodynamics: the finite-difference time-domain method. London: Artech House, 1995.
11. *Berenger J.-P.* A perfectly matched layer for the absorption of electromagnetic waves // Journal of Computational Physics. 1994. **114**, N 2. 185–200.
12. *Esirkepov T.Zh.* Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor // Computer Physics Communications. 2001. **135**, N 2. 144–153.
13. *Decyk V.K., Singh T.V.* Particle-in-cell algorithms for emerging computer architectures // Computer Physics Communications. 2014. **185**, N 3. 708–719.

Поступила в редакцию
7.08.2014

Particle-in-Cell Plasma Simulation Using Intel Xeon Phi Coprocessors

I. A. Surmin [1], S. I. Bastrakov [2], A. A. Gonoskov [3], E. S. Efimenko [4], and I. B. Meyerov [5]

[1] *Lobachevsky State University of Nizhni Novgorod, Faculty of Computational Mathematics and Cybernetics; prospekt Gagarina 23, Nizhni Novgorod, 603950, Russia; Junior Scientist, e-mail: i.surmin@gmail.com*

[2] *Lobachevsky State University of Nizhni Novgorod, Faculty of Computational Mathematics and Cybernetics; prospekt Gagarina 23, Nizhni Novgorod, 603950, Russia; Assistant, e-mail: sergey.bastrakov@gmail.com*

[3] *Institute of Applied Physics, Russian Academy of Sciences; ulitsa Ul'yanova 46, Nizhni Novgorod, 603950, Russia; Ph.D., Scientist, e-mail: arkady.gonoskov@gmail.com*

[4] *Institute of Applied Physics, Russian Academy of Sciences; ulitsa Ul'yanova 46, Nizhni Novgorod, 603950, Russia; Junior Scientist, e-mail: nnreene@mail.ru*

[5] *Lobachevsky State University of Nizhni Novgorod, Faculty of Computational Mathematics and Cybernetics; prospekt Gagarina 23, Nizhni Novgorod, 603950, Russia; Ph.D., Associate Professor, e-mail: meerov@vmk.unn.ru*

Received August 7, 2014

Abstract: A high performance implementation of particle-in-cell methods for laser plasma simulation is considered. The PICADOR code is used. An efficient utilization of the new Intel Xeon Phi coprocessors is discussed. It is shown that a code optimized well for traditional CPUs is not always efficient on coprocessors without additional optimization. A number of ways for the performance optimization of numerical plasma simulation are analyzed. The results of computing experiments show 1.8 times speed up on Xeon Phi compared to an optimized code on CPU.

Keywords: plasma physics, particle-in-cell method, high-performance computing, Xeon Phi, performance optimization.

References

1. C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation* (McGraw-Hill, New York, 1985; Energoatomizdat, Moscow, 1989).

2. R. A. Fonseca, J. Vieira, F. Fiuza, et al., "Exploiting Multi-Scale Parallelism for Large Scale Numerical Modelling of Laser Wakefield Accelerators," Plasma Phys. Contr. Fusion **55** (12) (2013). doi: 10.1088/0741-3335/55/12/124011

3. A. Pukhov, "Three-Dimensional Electromagnetic Relativistic Particle-in-Cell Code VLPL (Virtual Laser Plasma Lab)," J. Plasma Phys. **61** (3), 425–433 (1999).

4. K. J. Bowers, B. J. Albright, L. Yin, et al., "Advances in Petascale Kinetic Plasma Simulation with VPIC and Roadrunner," J. Phys.: Conf. Ser. **180** (1) (2009). doi: 10.1088/1742-6596/180/1/012055

5. H. Burau, R. Widera, W. Hönig, et al., "PIConGPU: A Fully Relativistic Particle-in-Cell Code for a GPU Cluster," IEEE Trans. Plasma Sci. **33** (10), 2831–2839 (2010).

6. S. Bastrakov, R. Donchenko, A. Gonoskov, et al., "Particle-in-Cell Plasma Simulation on Heterogeneous Cluster Systems," J. Comput. Sci. **3** (6), 474–479 (2012).

7. S. I. Bastrakov, I. B. Meyerov, I. A. Surmin, et al., "Dynamic Load Balancing in the PICADOR Plasma Simulation Code," Vychisl. Metody Programm. **14**, 67–74 (2013).

8. S. Bastrakov, I. Meyerov, I. Surmin, et al., "Particle-in-Cell Plasma Simulation on CPUs, GPUs and Xeon Phi Coprocessors," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2014), Vol. 8488, pp. 513–514.

9. J. Jeffers and J. Reinders, *Intel Xeon Phi Coprocessor High Performance Programming* (Morgan Kaufmann, Waltham, 2013).

10. A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method* (Artech House, London, 1995).

11. J.-P. Berenger, "A Perfectly Matched Layer for the Absorption of Electromagnetic Waves," J. Comput. Phys. **114** (2), 185–200 (1994).

12. T. Zh. Esirkepov, "Exact Charge Conservation Scheme for Particle-in-Cell Simulation with an Arbitrary Form-Factor," Comput. Phys. Commun. **135** (2), 144–153 (2001).

13. V. K. Decyk and T. V. Singh, "Particle-in-Cell Algorithms for Emerging Computer Architectures," Comput. Phys. Commun. **185** (3), 708–719 (2014).