УДК 004.942

# ДИНАМИЧЕСКАЯ БАЛАНСИРОВКА В КОДЕ PICADOR ДЛЯ МОДЕЛИРОВАНИЯ ПЛАЗМЫ

**С. И. Бастраков[1], И. Б. Мееров[1], И. А. Сурмин[1], А. А. Гоносков[2], Е. С. Ефименко[2], А. С. Малышев[1], М. А. Ширяев[1]**

Рассматривается задача балансировки нагрузки при моделировании плазмы методом частиц в ячейках на кластерных системах. Предлагается динамическая схема балансировки нагрузки, основанная на методе прямолинейного разбиения. Обсуждаются вопросы эффективной реализации оценки дисбаланса нагрузки и выполнения переразбиения. Эксперименты показывают, что на существенно несбалансированных задачах реализация демонстрирует не менее чем двукратное превосходство по сравнению с равномерным разбиением. Накладные расходы на поддержку балансировки составляют менее 1% от общего времени счета. Работа выполнена в лаборатории ННГУ-Intel "Информационные технологии" при поддержке ФЦП "Научные и научно-педагогические кадры инновационной России" (соглашение № 14.B37.21.0393), а также при поддержке Совета по грантам Президента Российской Федерации (код проекта НШ–1960.2012.9). Статья рекомендована к публикации Программным комитетом Международной научной конференции "Научный сервис в сети Интернет: все грани параллелизма" (http://agora.guru.ru/abrau2013).

**Ключевые слова:** балансировка нагрузки, физика плазмы, метод частиц в ячейках, высокопроизводительные вычисления.

**1. Введение.** Решение многих современных фундаментальных и прикладных задач требует численного моделирования процессов лазерного ускорения частиц, в частности, задачи взаимодействия сверхмощных лазерных импульсов с различными мишенями. Одним из наиболее разработанных и актуальных методов вычислительной физики является моделирование плазмы методом частиц в ячейках (Particle-in-Cell, PIC) [1]. Часто для адекватного численного анализа и достижения приемлемой точности требуется примерно $10^9$ частиц и порядка $10^8$ узлов пространственной сетки, что делает задачу вычислительно трудоемкой и приводит к необходимости создания специализированного программного обеспечения, ориентированного на использование суперкомпьютеров.

В настоящий момент созданы и продолжают развиваться программные комплексы для моделирования плазмы (PIC-коды) VLPL [2], OSIRIS [3], PIConGPU [4] и др. Как правило, подобные пакеты не распространяются открыто, поэтому для проведения исследований требуется разработка собственных средств численного моделирования. С 2010 г. коллективом сотрудников ННГУ и ИПФ РАН на основе опыта, полученного при создании PIC-кода ELMIS [5], разрабатывается программный комплекс PICADOR [6, 7], ориентированный на использование гетерогенных кластерных систем.

Одной из ключевых проблем при создании PIC-кода является эффективная организация распределения данных и вычислительной нагрузки в случае использования сотен и тысяч узлов кластерной системы. Метод частиц в ячейках оперирует двумя основными наборами данных: ансамбль заряженных частиц и сеточные значения электромагнитного поля. Независимое распределение частиц и сеточных значений по вычислительным узлам невозможно, так как взаимодействия "частица–сетка" являются пространственно локальными. В связи с этим вместе с подмножеством ансамбля частиц каждый из вычислительных узлов должен хранить и все пространственно близкие к ним сеточные значения поля. Возникает задача балансировки, состоящая в достижении по возможности равномерного распределения вычислительной нагрузки и используемой памяти по узлам кластера, а также минимизации затрат на обмен данными.

[1] Нижегородский государственный университет им. Н.И. Лобачевского (ННГУ), пр. Гагарина, 23, 603950, г. Нижний Новгород; С. И. Бастраков, ассистент, e-mail: sergey.bastrakov@gmail.com; И. Б. Мееров, зам. зав. каф., e-mail: meerov@vmk.unn.ru; И. А. Сурмин, программист, e-mail: i.surmin@gmail.com; А. С. Малышев, студент, e-mail: al.s.malyshev@rambler.ru;. М. А. Ширяев, студент, e-mail: migelurban@mail.ru

[2] Институт прикладной физики РАН (ИПФ РАН), ул. Ульянова, 46, 603950, г. Нижний Новгород; А. А. Гоносков, мл. науч. сотр., e-mail: arkady.gonoskov@gmail.com; Е. С. Ефименко, мл. науч. сотр., e-mail: nnreene@mail.ru

Применительно к моделированию плазмы данная задача рассматривается достаточно давно [8]. Исторически первым методом балансировки является распределение частиц по вычислительным узлам и хранение полной сетки на каждом из узлов. Данный метод хорошо подходит для задач с небольшой сеткой, однако плохо масштабируется на большое число узлов и накладывает существенные ограничения на максимальный размер сетки. В настоящее время применяется другая стратегия — декомпозиция расчетной области по территориальному принципу на домены, каждый из которых обрабатывается одним вычислительным узлом, хранящим все необходимые данные и обменивающимся данными с соседними доменами.

Методы разбиения на домены различаются, главным образом, формой доменов и контролем за количеством соседних доменов. Так, методы ортогональной рекурсивной бисекции (ORB) [9] и октодерева [10] осуществляют рекурсивное подразбиение расчетной области и обеспечивают хороший баланс по частицам, но приводят к увеличению количества соседних доменов и, следовательно, к усложнению обменов данными. Метод, предложенный разработчиками кода Quicksilver [11], основан на динамически “плавающих” границах доменов и введении промежуточной области — “окна”. Его модификация One-handed help [12] обеспечивает идеальный баланс по частицам и близкий к идеальному баланс по узлам сетки, но вносит накладные расходы на обмен значениями поля.

В настоящей статье предлагается динамическая схема балансировки нагрузки, основанная на идеологии метода прямолинейного разбиения [13]. Построенная таким образом декомпозиция требует обменов лишь с 26 соседними доменами и, поэтому, не вносит значительных накладных расходов. Рассматриваются вопросы эффективной реализации оценки дисбаланса нагрузки и выполнения переразбиения. Эффективность реализации демонстрируется на модельной задаче. Статья имеет следующую структуру: описание метода частиц в ячейках содержится в разделе 2, постановка задачи балансировки нагрузки и предлагаемая схема описываются в разделе 3, результаты вычислительных экспериментов приводятся в разделе 4.

**2. Постановка задачи и метод решения.** Расчетная область имеет форму прямоугольного параллелепипеда со сторонами, параллельными осям координат; в этой области заданы электрическое поле $\boldsymbol{E}$ и магнитное поле $\boldsymbol{B}$, которые представляются сеточными значениями. Плазма в расчетной области моделируется набором $N$ квазичастиц, характеризующихся постоянными массой $m$ и зарядом $q$, а также переменными импульсом $\boldsymbol{p}$ и положением $\boldsymbol{r}$. Каждая квазичастица представляет собой некоторое число физических частиц, электронов или ионов, что делается для уменьшения необходимых для моделирования ресурсов.

Динамика электромагнитного поля описывается уравнениями Максвелла, импульс и скорость частиц определяется из второго закона Ньютона в релятивистской формулировке, после чего положение частицы определяется из уравнений кинематики. Движение частиц создает плазменные токи $\boldsymbol{j}$, которые входят в виде источников в уравнения Максвелла, чем обеспечивается самосогласованность системы уравнений. Начальные условия состоят из значений электрического и магнитного полей, а также положения и скорости частиц в начальный момент времени. Используются периодические граничные условия, а также поглощающие граничные условия PML (Perfectly Matched Layer) [14].

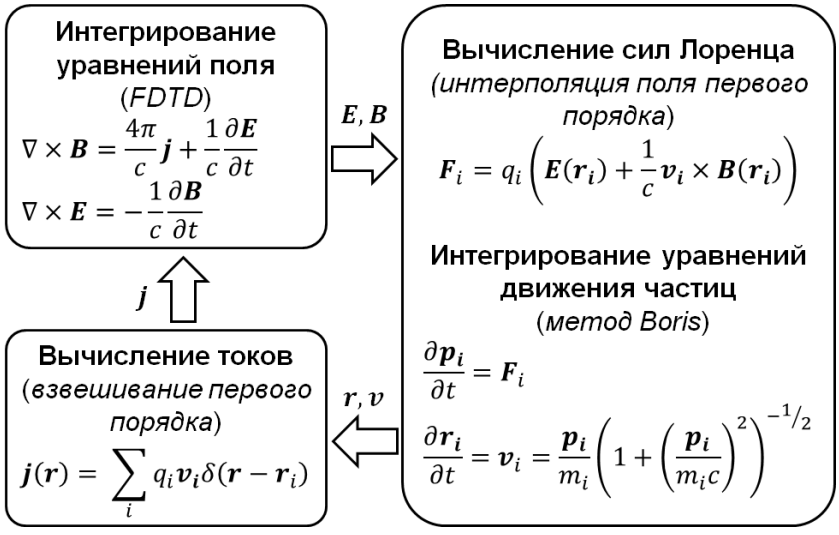


Рис. 1. Вычислительный цикл метода частиц в ячейках. Итерация соответствует моделированию одного шага по времени. Уравнения приведены в системе СГС (сантиметр–грамм–секунда). В скобках указаны используемые численные методы. Надписи над стрелками показывают зависимости по данным между этапами

На каждой итерации метода частиц в ячейках рассчитывается состояние системы в следующий момент времени. На $i$-й итерации хранятся значения электрического поля, токов и положения частиц в момент времени $i\Delta t$, магнитного поля в момент времени $i\Delta t + \Delta t/2$, импульсы частиц в момент времени $i\Delta t - \Delta t/2$, где $\Delta t$ — шаг по времени. Поля и характеристики частиц до первой итерации определяются из начальных условий. Итерация

метода включает в себя взвешивание токов, интегрирование уравнений Максвелла и уравнений движения частиц. Вычислительный цикл метода показан на рис. 1. Подробное описание метода частиц в ячейках приведено в [1].

Схема вычислений для кластерных систем устроена следующим образом. Декомпозиция задачи осуществляется по территориальному принципу: расчетная область разбивается на подобласти (домены), операции над которыми выполняются параллельно. Вычислительный узел, на котором производятся операции над определенным доменом, хранит данные об электромагнитном поле и частицах, находящихся в соответствующей части физического пространства. Данные, относящиеся к узлам, попадающим на границы между двумя или более доменами, хранятся на всех вычислительных узлах, обрабатывающих такие домены. Для поддержания актуальности данных во всех доменах используются обмены данными о токах, полях и частицах. При обменах токами и полями каждый домен взаимодействует с 6 соседями, при обменах частицами — с 26 соседями, так как в силу ограничений на соотношение шагов пространственной сетки и шага по времени покидающая домен частица обязательно попадает в один из соседних доменов. Таким образом, все обмены данными осуществляются между вычислительными узлами, обрабатывающими соседние домены [15].

**3. Балансировка нагрузки.** Реализация метода частиц в ячейках для кластерных систем использует технологию MPI, каждый домен обрабатывается одним MPI-процессом [15]. Реализация должна быть масштабируемой как по количеству частиц, так и по размеру сетки. Для этого необходимо по возможности равномерно распределять между процессами частицы и узлы сетки. При этом, если разделить поровну расчетную область, то из-за неравномерного распределения частиц вычислительная нагрузка для разных процессов может оказаться существенно различной. С другой стороны, если разделить частицы поровну между процессами без учета их расположения, то необходимо будет хранить значения полей на всей расчетной области и обмениваться ими на каждой итерации, что сильно увеличит накладные расходы. Поэтому для эффективной реализации необходимо найти компромисс между вычислительной несбалансированностью и коммуникационной сложностью.

Поставим задачу балансировки нагрузки следующим образом. Будем нумеровать ячейки пространственной сетки трехмерными индексами $(i, j, k)$, $0 \leqslant i < N_x$, $0 \leqslant j < N_y$, $0 \leqslant k < N_z$, где $N_x$, $N_y$, $N_z$ — количество ячеек расчетной области по соответствующим осям. При выполнении декомпозиции число доменов вдоль осей $x$, $y$ и $z$ считается фиксированным и равным $m$, $n$ и $l$ соответственно. Прямолинейное разбиение определяется тройкой целочисленных векторов, множество всех прямолинейных разбиений имеет следующий вид:

$$\mathcal{S} = \big\{(P, Q, R) : P = (0, p_1, p_2, \ldots, p_{m-1}, N_x),\ Q = (0, q_1, q_2, \ldots, q_{n-1}, N_y),\ R = (0, r_1, r_2, \ldots, r_{l-1}, N_z)\big\}.$$

При заданном прямолинейном разбиении $(P, Q, R) \in \mathcal{S}$ домен $(I, J, K)$ содержит ячейки с индексами $\big\{(i, j, k) :\ p_I \leqslant i < p_{I+1},\ q_J \leqslant j < q_{J+1}, r_K \leqslant k < r_{K+1}\big\}$. Каждой из ячеек сетки предлагается сопоставить значение вычислительной нагрузки $L_{ijk} = n\,\mathrm{Particles}_{ijk} + 1$, где $n\,\mathrm{Particles}_{ijk}$ — количество частиц в ячейке $(i, j, k)$. Вычислительная нагрузка домена складывается из вычислительных нагрузок всех его ячеек и, таким образом, равна суммарному количеству частиц и ячеек в подобласти. Введем функцию нагрузки домена $(I, J, K)$ при прямолинейном разбиении $(P, Q, R)$:

$$\mathrm{Domain\,Work}\,(P, Q, R, I, J, K) = \sum_{i=p_I}^{p_{I+1}-1} \sum_{j=q_J}^{q_{J+1}-1} \sum_{k=r_K}^{r_{K+1}-1} L_{ijk}\,.$$

Выбор этой функции нагрузки обусловлен следующими факторами: трудоемкость операций метода частиц в ячейках линейно зависит от количества частиц и ячеек, сумма количества частиц и ячеек (почти точно) пропорциональна количеству используемой памяти. Определим дисбаланс разбиения как отношение максимальной вычислительной нагрузки к средней:

$$\mathrm{Im\,balance}\,(P, Q, R) = \frac{\max\limits_{I=\overline{0,m-1},\ J=\overline{0,n-1},\ K=\overline{0,l-1}} \mathrm{Domain\,Work}\,(P, Q, R, I, J, K)}{\dfrac{1}{mnl} \displaystyle\sum_{I=0}^{m-1} \sum_{J=0}^{n-1} \sum_{K=0}^{l-1} \mathrm{Domain\,Work}\,(P, Q, R, I, J, K)}\,.$$

Стоящая в знаменателе средняя нагрузка не зависит от разбиения, но является удобной нормировкой.

Задача нахождения оптимального прямолинейного разбиения имеет следующий вид:

$$(P^{\mathrm{opt}}, Q^{\mathrm{opt}}, R^{\mathrm{opt}}) = \arg \min_{(P,Q,R)\in\mathcal{S}} \mathrm{Im\,balance}\,(P, Q, R).$$

Нахождение глобального минимума является NP-полной задачей [13]. Для выполнения балансировки задача решается эвристическим алгоритмом, позволяющим быстро найти достаточно хорошее решение. Алгоритм состоит в последовательном нахождении оптимального одномерного разбиения вдоль заданной оси при фиксированном разбиении вдоль двух других осей. Итерационный процесс продолжается до тех пор, пока следующее разбиение дает меньший дисбаланс, чем текущее.

Практическое применение рассмотренной схемы требует эффективной реализации вычисления дисбаланса на системах с распределенной памятью. Для быстрого вычисления суммарной нагрузки в домене будем использовать предвычисление префиксных сумм.

Пусть распределение нагрузки по ячейкам хранится в массиве $L_{ijk}$. Префиксные суммы образуют массив $W_{i,j,k} = \sum\limits_{ii=0}^{i-1} \sum\limits_{jj=0}^{j-1} \sum\limits_{kk=0}^{k-1} L_{ii,jj,kk}$. Для ячеек, два или один индекс которых равен нулю (лежащих на ребрах или гранях куба), значение вычисляется аналогично одно- или двумерному случаю, например $W_{i,0,0} = L_{i,0,0} + W_{i-1,0,0}$, $W_{i,j,0} = L_{i,j,0} + W_{i-1,j,0} + W_{i,j-1,0} - W_{i-1,j-1,0}$.

Пусть префиксные суммы вычислены в соседних ячейках с меньшими индексами. Тогда

$$W_{ijk} = L_{ijk} + W_{i-1,j,k} + W_{i,j-1,k} + W_{i,j,k-1} - W_{i-1,j-1,k} - W_{i-1,j,k-1} - W_{i,j-1,k-1} + W_{i-1,j-1,k-1}.$$

Таким образом, массив префиксных сумм вычисляется за один проход по исходному массиву. Далее суммарная нагрузка в любом домене вычисляется за константное время.

Так как каждый процесс обрабатывает свою область пространства, в нем хранятся сведения о распределении частиц лишь в одном домене. Простейшая реализация нахождения прямолинейного разбиения заключается в том, чтобы собирать данные о количестве частиц в ячейках во всей расчетной области на один процесс. Однако такая реализация является неэффективной, так как на больших сетках передачи приведут к значительным накладным расходам. Кроме того, для больших задач количество памяти на узле может быть недостаточным. Используется распределенная схема вычислений префиксных сумм без сбора всех данных на один процесс. Для этого в каждом процессе вычисляется префиксная сумма для соответствующей ему подобласти и производится обмен ее значениями со всеми процессами. На основе полученных данных каждый процесс вычисляет соответствующие значения $W_{ijk}$.

Метод балансировки реализован в виде двух схем: статической, выполняющейся однократно при запуске вычислений, и динамической, периодически производящей оценку дисбаланса в текущей декомпозиции и, при достижении определенного порога дисбаланса, выполняющей перебалансировку. Динамическая схема имеет два настраиваемых параметра: частоту проверки и пороговое значение дисбаланса. В настоящее время их значения подбираются вручную, в дальнейшем возможно введение эвристики для автоматической подстройки параметров. Для переразбиения реализована распределенная схема передач, согласно которой данные передаются только в тех областях пространства, которые переходят из одного домена в другой. Расчетная область делится на блоки, образованные объединением границ старого и нового разбиений. Передача блоков происходит в два этапа: на первом этапе процессы с меньшим рангом передают блоки процессам с бóльшим рангом, а процессы с бóльшим рангом принимают блоки; затем отправители и получатели меняются местами. Такая схема предотвращает блокировки процессов.

**4. Результаты экспериментов.** Для расчетов использовался кластер Межведомственного суперкомпьютерного центра РАН МВС-100К (на каждом узле 2 CPU Intel Xeon E5450, 8 GB RAM, Infiniband DDR). В качестве задачи, в которой необходима балансировка нагрузки, был выбран следующий тест: частицы распределены равномерно в шаре небольшого радиуса, расположенного в центре расчетной области. Число частиц — 42 млн, размеры сетки — $128 \times 128 \times 128$. В начальный момент времени частицы имеют высокую температуру и в процессе расчета разлетаются от центра к границам расчетной области. Рассмотрим расчет в течение 100 и 1000 итераций по времени.

Распределение частиц по доменам показано на рис. 2. Слева и в центре отображены не изменяющиеся от итерации к итерации границы доменов при равномерном и статическом разбиении соответственно, а также распределение частиц в начальный момент времени. Динамическая схема в начальный момент имеет такое же разбиение, как и статическая схема, однако в дальнейшем положение границ доменов определяется динамикой частиц и дисбалансом согласно вышеописанной схеме.

Финальное распределение границ доменов и частиц показано на рис. 2 справа; в данном случае разбиение близко к равномерному, что объясняется близким к равномерному распределением частиц по доменам. При расчете в течение 100 итераций статическая схема обгоняет равномерную в 2.5–5 раз в зависимости от количества процессов. Применение динамической схемы балансировки в данном случае не сильно сказывается на производительности, так как распределение частиц в течение расчета не успевает сильно измениться (рис. 3).

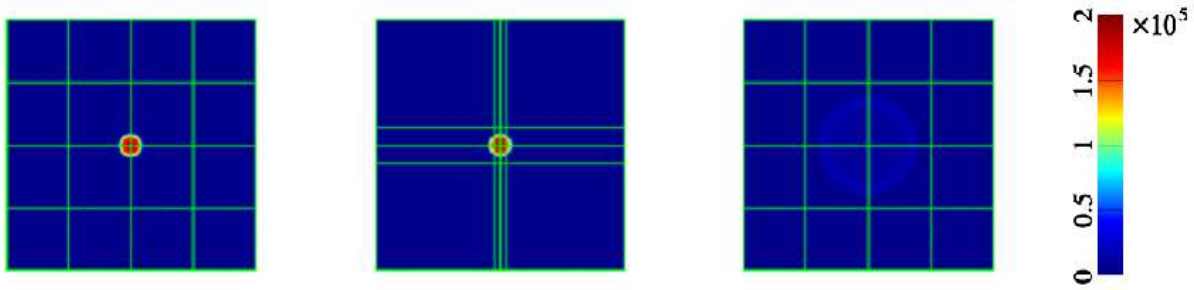


Рис. 2. Декомпозиция в модельной задаче, двумерный срез. Границы доменов обозначены линиями зеленого цвета, плотность частиц показана цветом. Слева: начальный момент времени, равномерное разбиение. В центре: статическое разбиение (совпадает с динамическим в начальный момент времени). Справа: динамическое разбиение на итерации 100

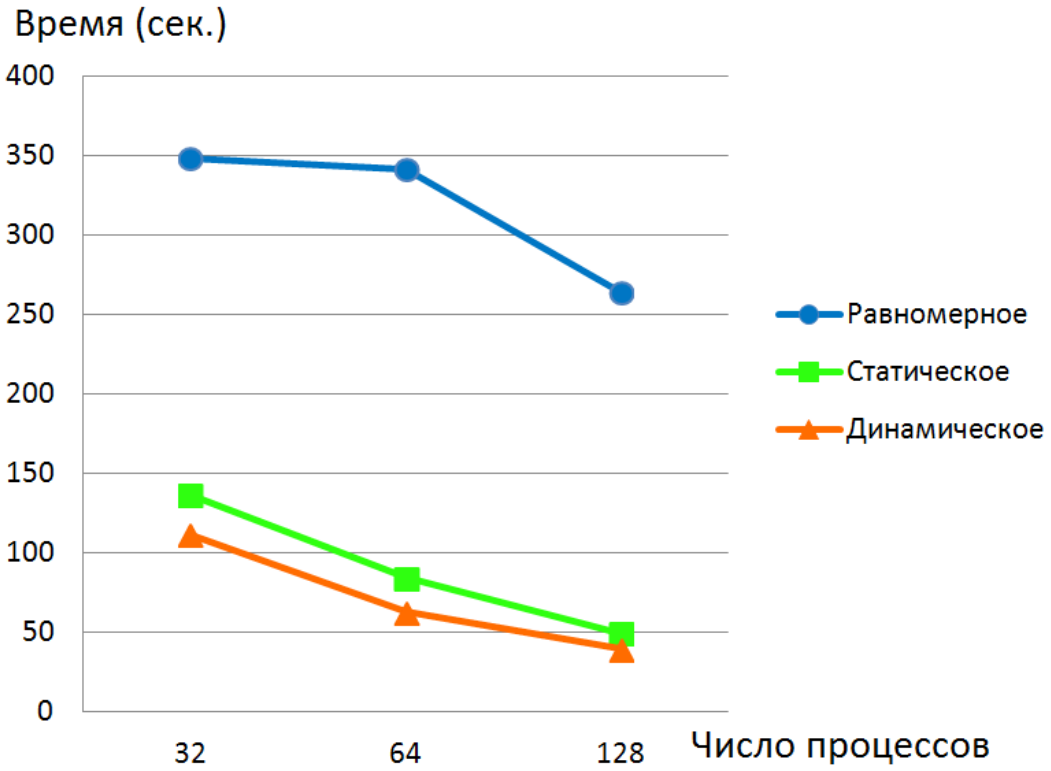


Рис. 3. Сравнение равномерного, статического и динамического разбиения расчетной области при моделировании 100 шагов по времени на сетке $128 \times 128 \times 128$ с 42 млн частиц

При увеличении времени расчета до 1000 итераций частицы в процессе разлета постепенно заполняют всю расчетную область. Так как распределение частиц значительно изменяется, статическая декомпозиция по начальному распределению только ухудшает время работы по сравнению с равномерной. Применение динамической схемы балансировки позволяет добиться двукратного увеличения производительности по сравнению с равномерным разбиением (рис. 4). Для того чтобы оценить, насколько результат близок к оптимуму, рассмотрим идеально сбалансированную задачу с тем же количеством частиц и ячеек сетки. Пусть в начальный момент частицы распределены равномерно на всей расчетной области, тогда с течением времени распределение будет также оставаться равномерным, что соответствует невозмущенной плазме. Время решения несбалансированной задачи превышает время решения идеально сбалансированной в 1.5 раза, что является достаточно хорошим результатом (рис. 4).

На рис. 5 приведено распределение времени работы между процессами, цветами показаны различные этапы вычислений. При равномерном разбиении частицы сосредоточены в центральных доменах, на графике этим значениям соответствуют пики в центре. При статической декомпозиции наибольшее время тратится на обработку угловых доменов, при использовании динамической балансировки времена работы

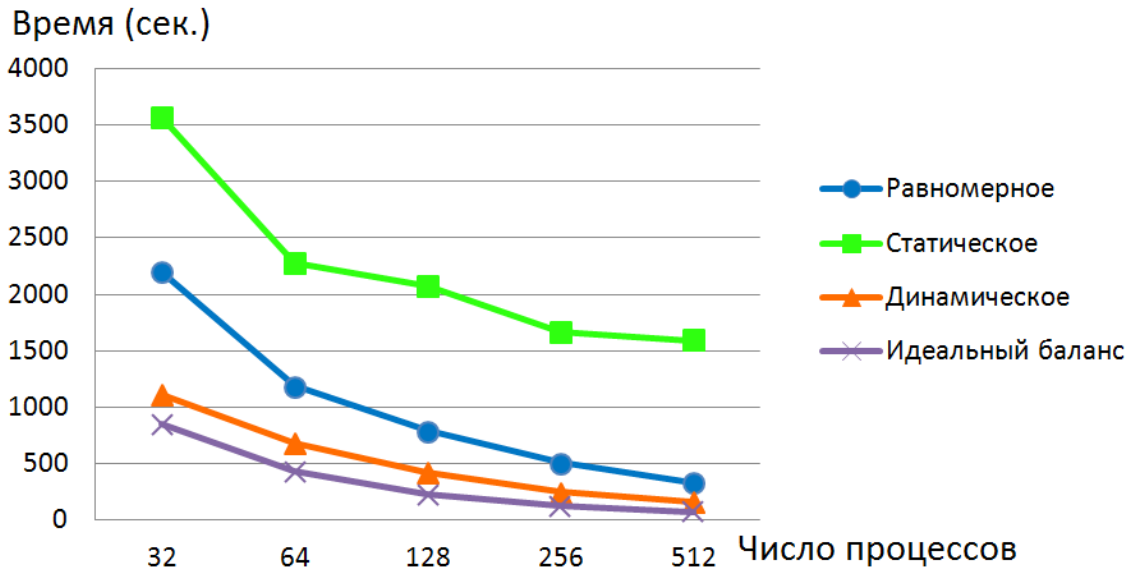


Рис. 4. Сравнение равномерного, статического и динамического разбиения при моделировании 1000 шагов по времени на сетке $128 \times 128 \times 128$ с 42 млн частиц. Под идеальным балансом понимается задача с тем же количеством ячеек сетки и частиц, равномерно распределенных по расчетной области

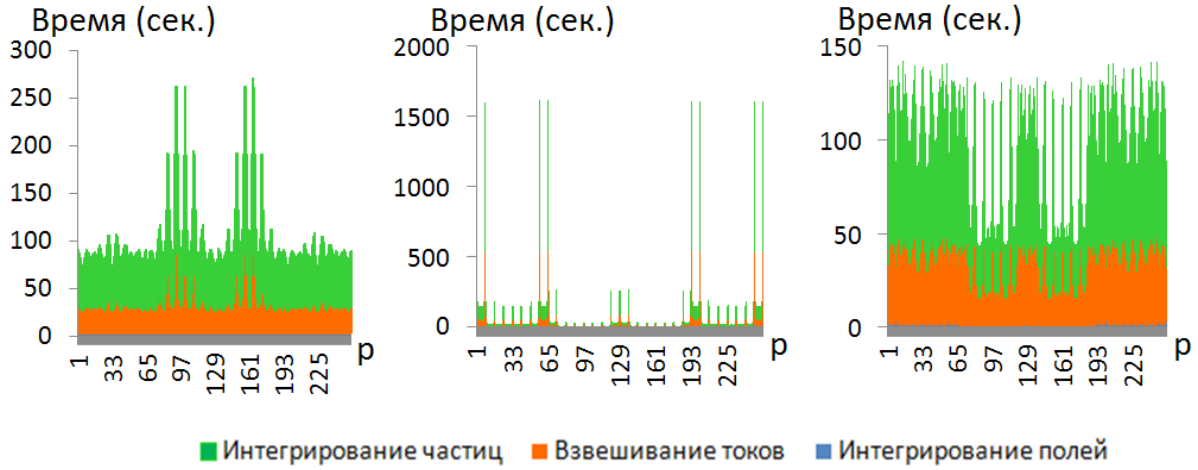


Рис. 5. Распределение времени работы по процессам при моделировании 1000 шагов по времени на сетке $128 \times 128 \times 128$ с 42 млн частиц с использованием 256 процессов. Слева: равномерное разбиение. В центре: статическое разбиение. Справа: динамическое разбиение

различных процессов в значительной степени выравниваются.

Зависимость дисбаланса от времени показана на рис. 6. При равномерном разбиении в начальный момент времени все частицы сосредоточены в центральных доменах, что соответствует большому дисбалансу, однако со временем частицы разлетаются и дисбаланс уменьшается. При статическом разбиении, напротив, большинство частиц попадает в угловые домены, из-за этого изначально низкий дисбаланс сильно возрастает. При динамической балансировке дисбаланс скачкообразно уменьшается после каждого разбиения.

Во всех расчетах были выбраны следующие параметры алгоритма балансировки нагрузки: проверка необходимости переразбиения производилась 1 раз в 50 итераций, порог дисбаланса, при превышении которого выполнялось переразбиение, был установлен равным 1.2 . Заметим, что время на проверку и переразбиение не превышает 1% времени счета, поэтому алгоритм может быть использован и в относительно хорошо сбалансированных задачах.

**5. Заключение.** В настоящей работе получены следующие результаты.

1. Поставлена задача балансировки нагрузки в многопроцессорной реализации численного моделиро-

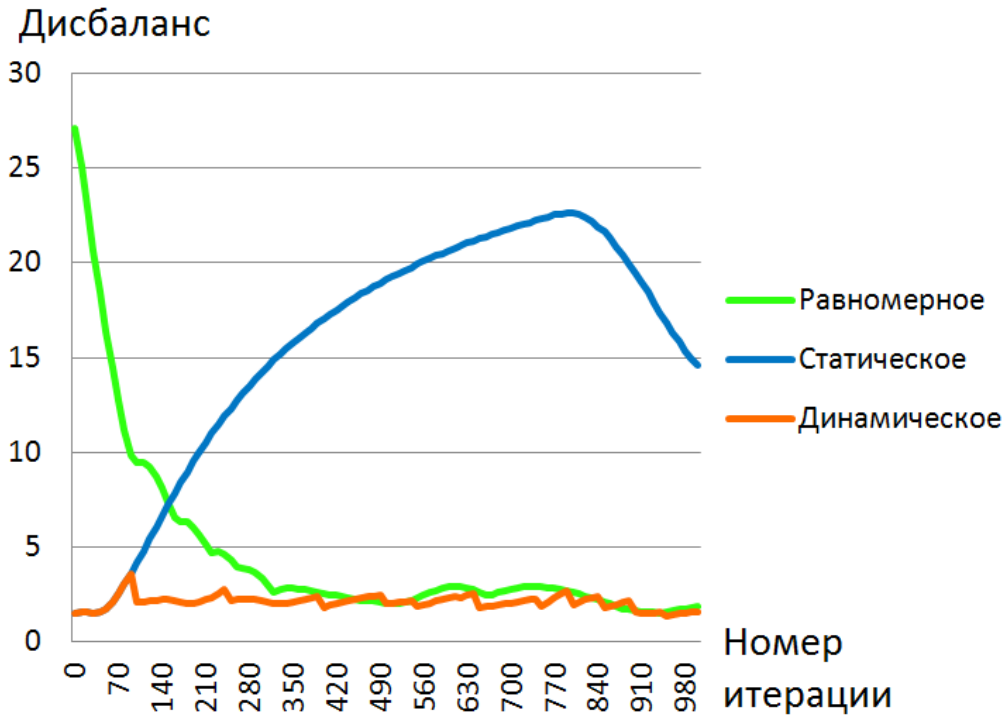


Рис. 6. Зависимость дисбаланса равномерного, статического и динамического разбиения от времени при моделировании 1000 шагов по времени на сетке $128 \times 128 \times 128$ с 42 млн частиц с использованием 256 процессов

вания плазмы.

2. Для решения указанной NP-полной задачи реализован алгоритм, основанный на схеме прямолинейной декомпозиции [13]. Предложена эффективная реализация, минимизирующая накладные расходы, связанные с передачами и затратами времени на оценку текущего дисбаланса.

3. Алгоритм реализован в рамках программного комплекса PICADOR. Эксперименты показали, что на существенно несбалансированных задачах реализация демонстрирует не менее чем двукратное превосходство по сравнению с равномерным разбиением. Оценка отставания от теоретического оптимума составляет 1.5 раза. Незначительные накладные расходы позволяют применять алгоритм и для решения хорошо сбалансированных задач.

Далее планируется автоматизировать настройку параметров алгоритма: частоту проверки дисбаланса и пороговое значение дисбаланса для переразбиения.

## Список литературы

1. *Бэдсел Ч., Ленгдон А.* Физика плазмы и численное моделирование. М.: Энергоатомиздат, 1989.
2. *Pukhov A.* Three-dimensional electromagnetic relativistic Particle-In-Cell code VLPL // J. of Plasma Physics. 1999. **61**, N 3. 425–433.
3. *Fonseca R.A., et al.* OSIRIS: a three-dimensional, fully relativistic Particle-In-Cell code for modeling plasma based accelerators // Lecture Notes in Computational Science. Vol. 2331. Berlin: Springer, 2002. 342–351.
4. *Burau H., Widera R., Honig W., et al.* PIConGPU: a fully relativistic Particle-In-Cell code for a GPU cluster // IEEE Trans. on Plasma Science. 2010. **33**, N 10. 2831–2839.
5. *Korzhimanov A., Gonoskov A.* ELMIS — a fully parallel Fourier-based multidimensional PIC code for laser-plasma interaction simulations // Poc. of the 22-nd Intl. Conf. on Numerical Simulation of Plasmas (ICNSP 2011). September 7–9, 2011. Long Branch, New Jersey, USA (http://icnsp2011/pppl.gov/).
6. *Bastrakov S., Donchenko R., Gonoskov A., Efimenko E., Malyshev A., Meyerov I., Surmin I.* Particle-In-Cell plasma simulation on heterogeneous cluster systems // J. of Computational Science. 2012. **3**, N 6. 474–479.
7. *Bastrakov S., Meyerov I., Gergel V., et al.* High performance computing in biomedical applications // Procedia Computer Science. 2013. **18**. 10–19.
8. *Liewer P.C., Decyk V.K.* A general concurrent algorithm for plasma particle-in-cell codes // J. Computational Physics. 1989. **85**, N 2. 302–322.

9. *Fox G.C.* A review of automatic load balancing and decomposition methods for the Hypercube // Numerical Algorithms for Modern Parallel Computer Architectures. The IMA Volumes in Mathematics and Its Applications / Ed. by M. Schultz. New York: Springer, 1988. Vol. 1. 63–76.
10. *Barnes J., Hutt P.* A hierarchical O(N logN) force calculation algorithm // Nature. 1986. **324**. 446–449.
11. *Seidel D.B., Plimpton S.J., Pasik M.F., et al.* Dynamic load balancing for a parallel electromagnetic particle-in-cell code // Proc. IEEE Intl. Pulsed Power Conf. Las Vegas: IEEE Press, 2001. 1000–1003.
12. *Nakashima H., Miyake Y., Usui H., Omura Y.* OhHelp: a scalable domain-decomposing dynamic load balancing for particle-in-cell simulations // Proc. of the 23rd Intl. Conf. on Supercomputing (ICS-2009). New York: ACM Press, 2009. 90–99.
13. *Nicol D.N.* Rectilinear partitioning of irregular data parallel computations // J. of Parallel and Distributed Computing. 1994. **23**, N 2. 119–134.
14. *Taflove A.* Computational electrodynamics: the finite-difference time-domain method. London: Artech House, 1995.
15. *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 // Тр. Междунар. конф. “Параллельные вычислительные технологии, 2011” (ПаВТ-2011). Челябинск: Издательский центр ЮУрГУ, 2011. 411–417.

Поступила в редакцию
08.09.2013