ДИНАМИЧЕСКАЯ БАЛАНСИРОВКА В КОДЕ PICADOR ДЛЯ МОДЕЛИРОВАНИЯ ПЛАЗМЫ

С.И. Бастраков, И.Б. Мееров, И.А. Сурмин, А.А. Гоносков, Е.С. Ефименко, А. С. Малышев, М.А. Ширяев

Введение

Одной из актуальных задач вычислительной физики является *моделирование плазмы методом частиц в ячейках* (Particle-in-cell, PIC) [1]. Решение прикладных задач требует моделирования процессов взаимодействия сверхмощных лазерных импульсов с различными мишенями, в частности, лазерного ускорения частиц. Для достижения приемлемой точности требуется ~10^9 частиц и ~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 в мире созданы и продолжают развиваться программные комплексы для моделирования плазмы (PIC-коды) VLPL [2], OSIRIS [3], PIConGPU [4] и другие. Как правило, подобные пакеты не распространяются открыто, поэтому для проведения исследований требуется разработка собственных средств численного моделирования. С 2010 года 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.

При создании PIC-кода встает проблема организации схемы распараллеливания, эффективной в случае использования тысяч и десятков тысяч процессоров. Так, 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 типа 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 нагрузки*, состоящая в достижении по возможности равномерного распределения вычислительной нагрузки и используемой памяти по узлам, а также минимизации затрат на обмены данными. Применительно к моделированию плазмы данная задача рассматривается достаточно давно. 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) [8] и октодерева [9] осуществляют рекурсивное подразбиение расчетной области и обеспечивают хороший баланс по частицам, но приводят к увеличению количества соседних доменов, и, следовательно, сложности обменов данными. Метод, предложенный разработчиками кода Quickvilver [10], основан на динамически «плавающих» границах доменов и введении промежуточной области – «окна». Его модификация One-handed help [11] обеспечивает идеальный баланс по частицам и близкий к идеальному баланс по узлам сетки, но вносит накладные расходы на обмен полями.

В работе предлагается эффективная реализация *алгоритма динамической балансировки нагрузки*, основанного на идеологии метода прямолинейного разбиения [12]. Построенная методом декомпозиция требует обменов лишь с 26 соседними доменами и, тем самым, вносит минимум накладных расходов. Эффективность реализации демонстрируется на модельной задаче.

Постановка задачи и метод решения

Под расчетной областью понимается область пространства, в которой происходит моделирование плазмы. Расчетная область имеет форму прямоугольного параллелепипеда со сторонами, параллельными осям координат. В каждой точке области заданы электрическое поле E и вектор магнитной индукции B. Область заполнена вакуумом, в котором находится N частиц, в совокупности называемых плазмой. Каждая частица задается постоянными массой m и зарядом q, а также переменными импульсом p и положением r.

Электромагнитное поле изменяется с течением времени согласно уравнениям Максвелла (рисунок 1), где c – скорость света, J – вектор плотности тока, индуцированного движением частиц. Координаты и скорость частиц меняются согласно второму закону Ньютона в релятивистской формулировке. Начальные условия состоят из значений электрического и магнитного полей во всех точках расчетной области, а также положений и скоростей частиц в начальный момент времени. Используются периодические граничные условия.

Для численного решения уравнений Максвелла вводятся основная и вспомогательная пространственные сетки, узлы которых сдвинуты относительно друг друга на половину шага (по каждой

координате). В узлах основной сетки заданы значения электрического поля и токов, в узлах вспомогательной – значения магнитного поля. На каждой итерации метода частиц в ячейках рассчитывается состояние системы в следующий момент времени. На i-ой итерации хранятся значения электрического поля, токов и положения частиц в момент времени $i\Delta t$, магнитного поля в в момент времени $i\Delta t + \Delta t/2$, импульсы частиц в момент времени $i\Delta t - \Delta t/2$, где Δt – шаг по времени. Поля и характеристики частиц до первой итерации определяются из начальных условий. Итерация метода включает взвешивание токов, интегрирование уравнений Максвелла и уравнений движения частиц. Вычислительный цикл метода показан на рисунке 1. Подробное описание модели и метода решения приведено в [1].

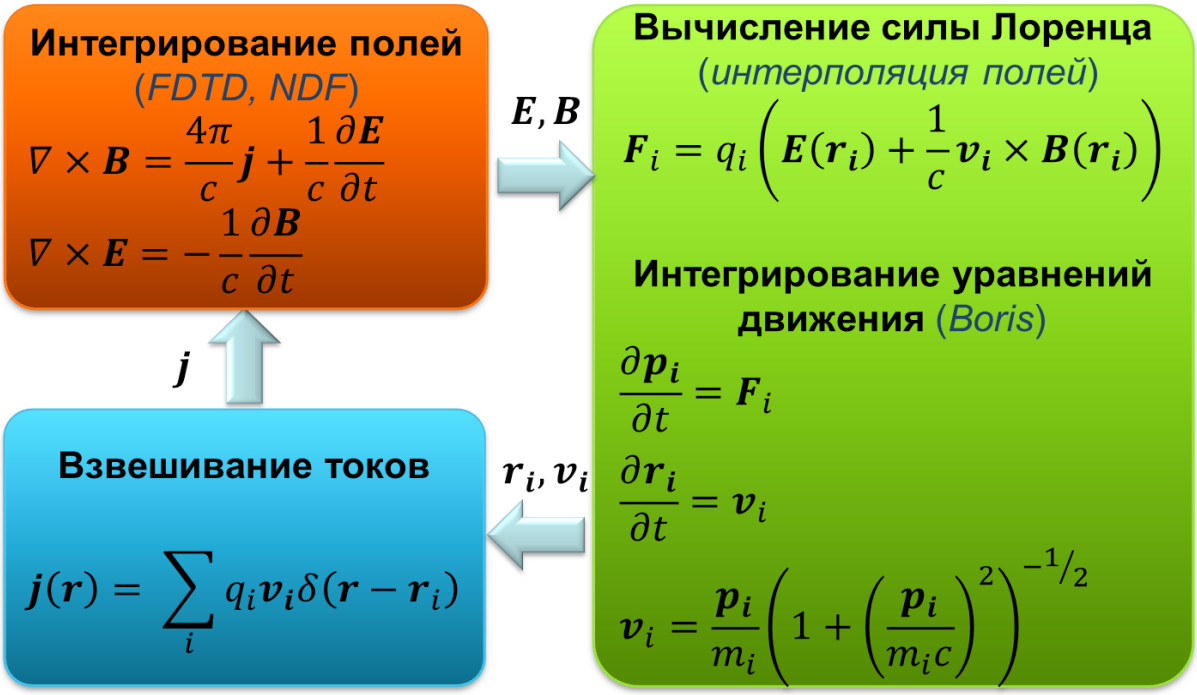

Рис. 1. Вычислительная схема метода частиц в ячейках

Базовая схема вычислений для систем устроена следующим образом. Декомпозиция задачи осуществляется по территориальному принципу: расчетная область разбивается на подобласти (домены) равного размера, операции над которыми выполняются параллельно. Вычислительный узел, на котором производятся операции над определенным доменом, хранит данные об электромагнитных полях 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. Вычислительный узел хранит данные о магнитных полях в центрах ячеек, граничащих с обрабатываемым им доменом. 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 (в случае, когда частица покидает пределы домена, она попадает в один из соседних доменов). Таким образом, все обмены данными осуществляются между вычислительными узлами, обрабатывающими соседние домены [13].

Балансировка нагрузки

Реализация метода частиц в ячейках должна быть масштабируемой как по числу частиц, так и по размеру сетки. Для этого необходимо по возможности разделять частицы и узлы сетки поровну между процессами. При этом, если разделить поровну расчетную область, то из-за неравномерного распределения частиц вычислительная нагрузка для разных процессов будет различной. С другой стороны, если разделить частицы поровну между процессами без учета их расположения, необходимо будет хранить значения полей на всей расчетной области и обмениваться ими на каждой итерации, что сильно увеличит накладные расходы. Поэтому для эффективной реализации нужно найти компромисс между вычислительной несбалансированностью и коммуникационной сложностью.

Поставим задачу балансировки нагрузки следующим образом. Будем нумеровать ячейки трехмерными индексами (i, j, k), $0 \le i < N_x$, $0 \le j < N_y$, $0 \le k < N_z$, где N_x, N_y, N_z – количество ячеек расчетной области по соответствующей оси. При выполнении декомпозиции число доменов вдоль каждой из осей считается фиксированным и равным m, n и l, соответственно. Прямолинейное разбиение определяется тройкой целочисленных векторов, множество всех прямолинейных разбиений имеет вид:

$S = \{(P, Q, R): P = (p_0 = 0, p_1, p_2, \ldots, p_m = N_x), Q = (q_0 = 0, q_1, q_2, \ldots, q_n = N_y), R = (r_0 = 0, r_1, r_2, \ldots, r_l = N_z)\}$.

При заданном прямолинейном разбиении *(P, Q, R)* домен (*I*, *J*, *K*) содержит ячейки *{(i, j, k): $p_I \le i < p_{I+1}$, $q_J \le j < q_{J+1}$, $r_K \le k < r_{K+1}$}*. Каждой из ячеек сетки предлагается сопоставить значение вычислительной нагрузки $L_{i,j,k} = nParticles_{i,j,k} + 1$, где $nParticles_{i,j,k}$ – количество частиц в ячейке (*i, j, k*). Вычислительная нагрузка домена равна сумме вычислительных нагрузок всех его ячеек, которая равна суммарному количеству частиц и ячеек в подобласти. Введем функцию нагрузки домена (*I*, *J*, *K*) при прямолинейном разбиении *(P, Q, R)*:

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

Выбор данной функции 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

$$Imbalance(P, Q, R) = \frac{max_{I=0,m-1,J=0,n-1,K=0,l-1} DomainWorkload(P, Q, R, I, J, K)}{\frac{1}{mnl} \sum_{I=0}^{m-1} \sum_{J=0}^{n-1} \sum_{K=0}^{l-1} DomainWorkload(P, Q, R, I, J, K)}$$

Стоящая в знаменателе средняя нагрузка не меняется при переразбиении, но является удобной нормировкой. Задача нахождения оптимального прямолинейного разбиения имеет следующий вид:

$$(P^{opt}, Q^{opt}, R^{opt}) = arg\ min_{(P,Q,R) \in S} Imbalance(P, Q, R)$$

Нахождение глобального минимума является NP-полной задачей. 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, 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 $L_{i,j,k}$. Префиксные суммы образуют массив $W_{i,j,k} = \sum_{i'=0}^{i-1} \sum_{j'=0}^{j-1} \sum_{k'=0}^{K-1} L_{i',j',k'}$. Для ячеек, два индекса которых равны нулю (лежащих на ребрах куба), значение вычисляется аналогично одномерному случаю: $W_{i,0,0} = L_{i,0,0} + W_{i-1,0,0}$. Для ячеек, один из индексов которых равен нулю (лежащих на гранях куба): $W_{i,j,0} = L_{i,j,0} + W_{i-1,j,0} + W_{i,j-1,0} - W_{i-1,j-1,0}$. Аналогично вычисляются значения, лежащие на других гранях.

Пусть префиксные суммы вычислены в соседних ячейках с меньшими индексами. Тогда $W_{i,j,k} = L_{i,j,k} + W_{i-1,j,k} + W_{i,j-1,k} + W_{i,j,k-1} - W_{i-1,j-1,k} - W_{i-1,j,k-1} - W_{i,j-1,k-1} + W_{i-1,j-1,k-1}$. Таким образом, массив префиксных сумм вычисляется за один проход по исходному массиву. Далее за константное время вычисляется суммарная нагрузка в любом домене.

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 достаточно 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 каждой процесс вычисляет соответствующие значения $W_{i,j,k}$.

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 большим рангом, а процессы с большим рангом принимают блоки. Затем отправители и получатели меняются местами. Такая схема предотвращает блокировки процессов.

Результаты экспериментов

Для расчетов используется кластер Межведомственного Суперкомпьютерного Центра РАН МВС-100К (на каждом узле 2 CPU Intel Xeon E5450, 8 ГБ 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 – 128x128x128. В начальный момент времени частицы имеют высокую температуру, поэтому разлетаются от центра к границам расчетной области. Рассмотрим расчет в течение 100 и 1000 итераций по времени.

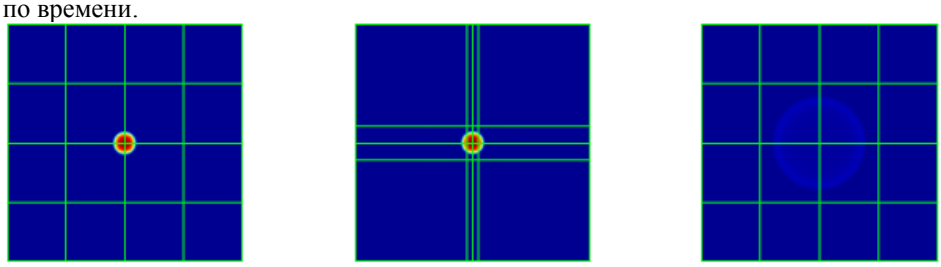

Рис. 2. Распределение частиц по доменам. Слева: итерация 0, равномерное разбиение. В центре: статическое разбиение; итерация 0, динамическое разбиение. Справа: итерация 100, динамическое разбиение.

Распределение частиц по доменам показано на рисунке 2. Слева и в центре отображены не изменяющиеся от итерации к итерации границы доменов при равномерном и статическом разбиении (соответственно). Распределение частиц соответствует начальному моменту времени. Изображение в центре также соответствует начальной итерации динамической схемы разбиения. Финальное распределение границ доменов и частиц показано на рисунке 1 справа.

Оценим производительность и накладные расходы обсуждаемых схем. При расчете в течение 100 итераций статическая схема обгоняет равномерную в 2.5-5 раз на различных топологиях процессов. Балансировка нагрузки не сильно сказывается на производительности, так как распределение частиц в течение расчета сильно не изменяется (рисунок 3).

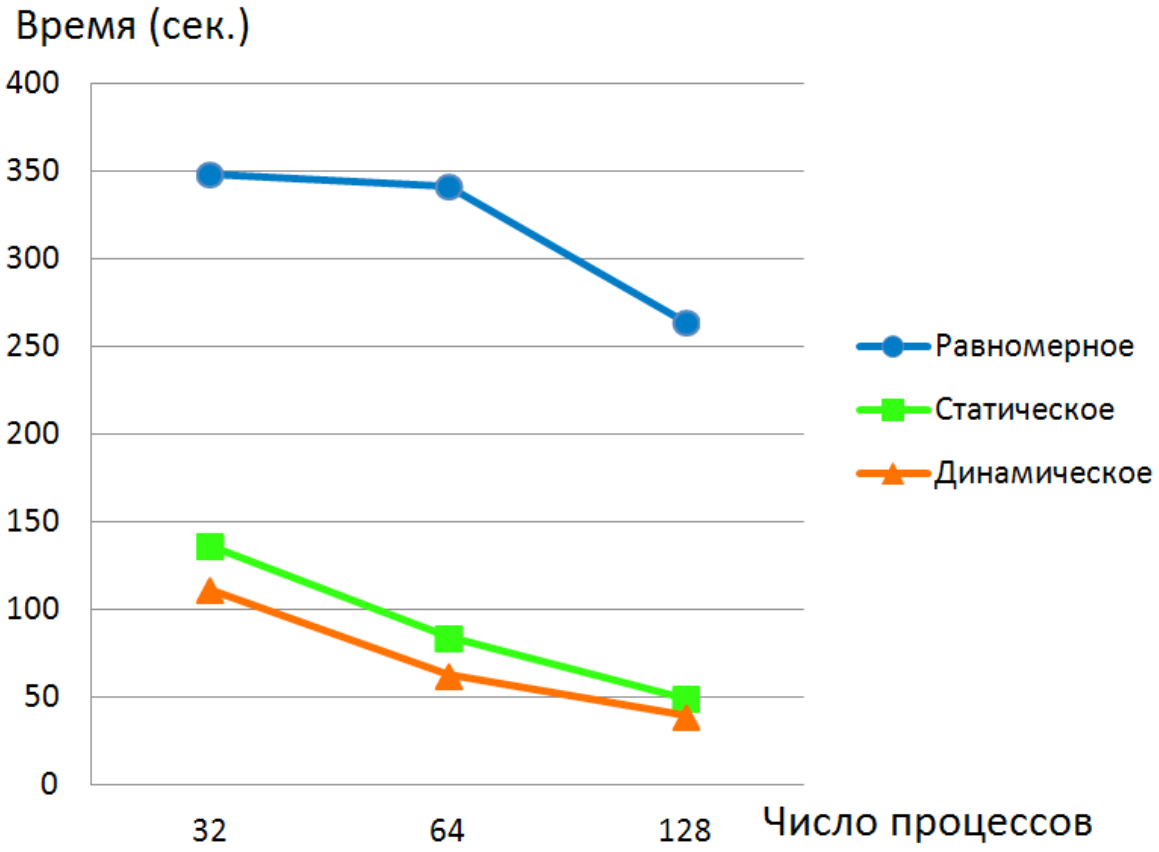

Рис.3. Сравнение схем разбиения при выполнении 100 итераций

Проведем эксперимент для 1000 итераций. За данное время частицы заполнят всю расчетную область. Так как распределение частиц значительно изменяется, статическая декомпозиция только ухудшает время работы по сравнению с равномерной. Динамическая балансировка позволяет добиться двукратного увеличения производительности по сравнению с равномерным разбиением (рисунок 4).

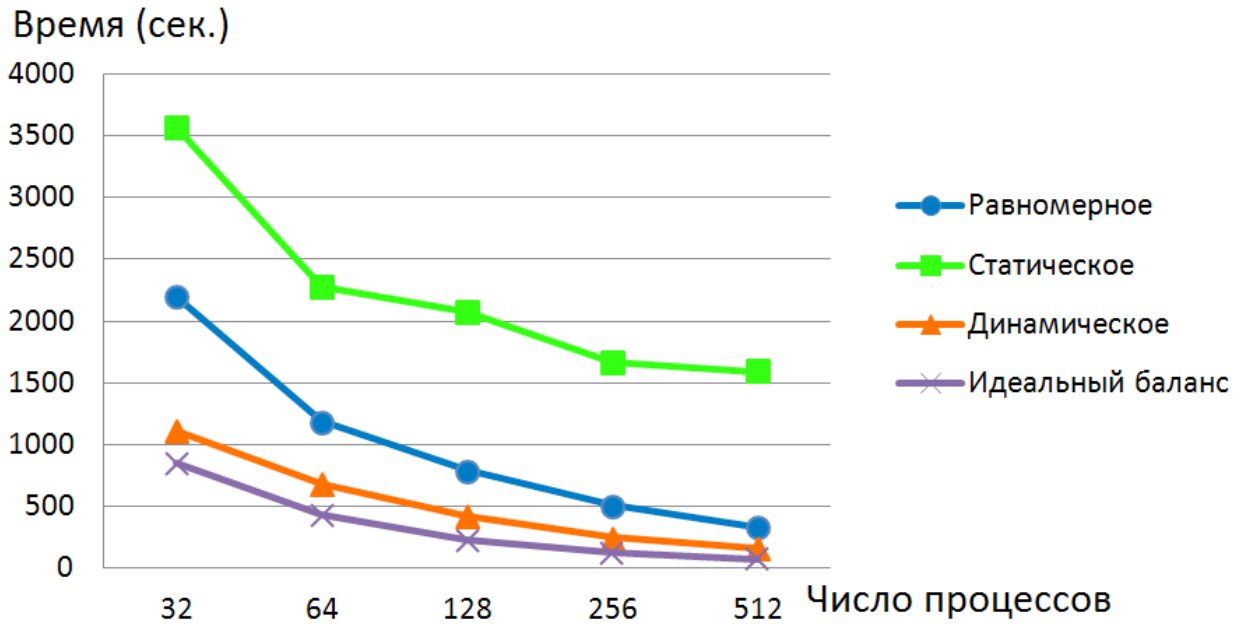

Рис. 4. Сравнение схем разбиения при выполнении 1000 итераций

Чтобы оценить, насколько результат близок к оптимуму, рассмотрим идеально сбалансированную задачу. Пусть в начальный момент частицы распределены равномерно на всей расчетной области. С течением времени распределение будет оставаться равномерным. Время решения несбалансированной задачи превышает время решения идеально сбалансированной в 1.5 раза, что является достаточно хорошим результатом (рисунок 4).

На рисунке 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 различных процессов различаются существенно меньше.

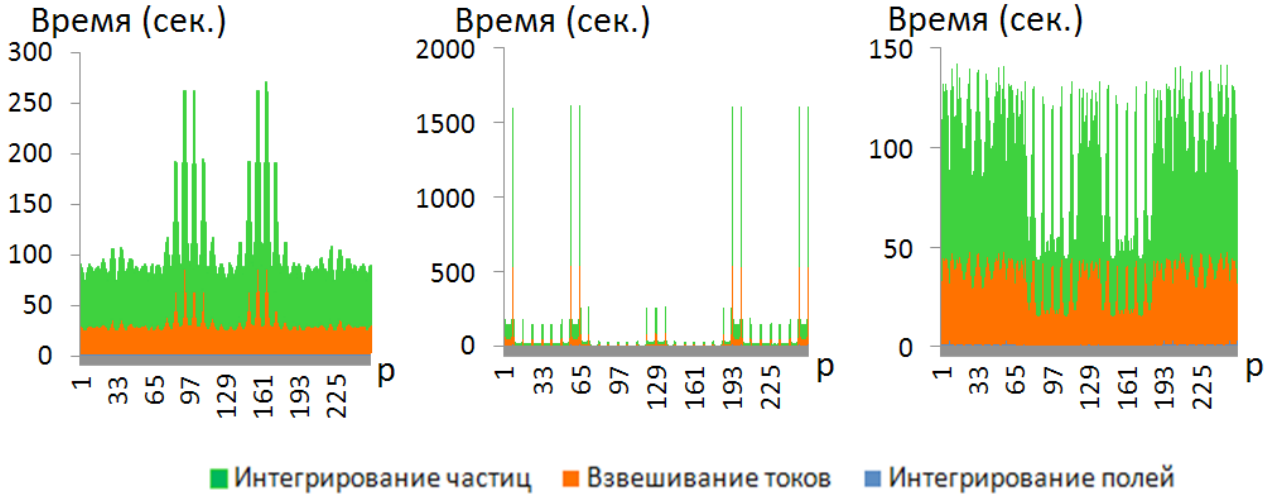

Рис. 5. Распределение времени работы по стадиям

Зависимость дисбаланса от времени показана на рисунке 6**.** При равномерном разбиении в начальный момент времени все частицы сосредоточены в центральных доменах. Со временем частицы разлетаются, из-за этого 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.

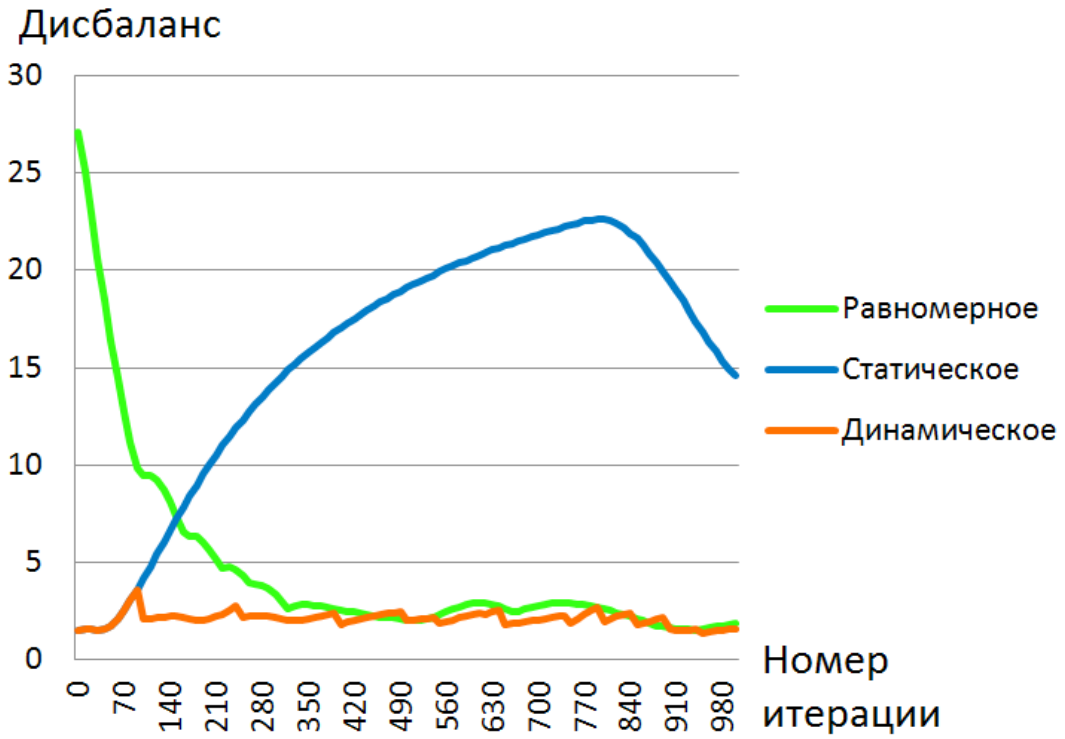

Рис. 6. Зависимость дисбаланса от времени

Во всех расчетах были выбраны следующие параметры алгоритма балансировки нагрузки: проверка необходимости переразбиения производилась1 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 хорошо сбалансированных задачах.

Заключение

В работе получены следующие результаты:

1. Поставлена задача балансировки нагрузки в многопроцессорной реализации численного моделирования плазмы.
2. Для решения указанной NP-полной задачи реализован алгоритм, основанный на схеме прямолинейной декомпозиции [12]. 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 алгоритм демонстрирует не менее чем двукратное превосходство по сравнению с равномерным разбиением. Оценка отставания от теоретического оптимума – 1.5 раза. Незначительные накладные расходы на использование алгоритма позволяют использовать его и в хорошо сбалансированных задачах.

Далее планируется реализовать автоматическую настройку параметров алгоритма.

Работа выполнена в лаборатории ННГУ-Intel «Информационные технологии».

ЛИТЕРАТУРА:

1. Ч. Бэдсел, А. Ленгдон. Физика плазмы и численное моделирование: Пер. с англ. – М.: Энергоатомиздат, 1989. – 452 С.
2. A. Pukhov. Three-Dimensional Electromagnetic Relativistic Particle-in-Cell code VLPL // Journal of Plasma Physics, vol. 61, 1999. – p. 425–433.
3. R.A. Fonseca, et al. OSIRIS: A Three-Dimensional, Fully Relativistic Particle in Cell Code for Modeling Plasma Based Accelerators // Lecture Notes in Computational Science 2331, 2002. – p. 342-351.
4. H. Burau, R. Widera, W. Honig et al. PIConGPU: A Fully Relativistic Particle-in-Cell Code for a GPU Cluster // IEEE Transactions on Plasma Science, vol. 33, 2010. – p. 2831–2839.
5. A. Korzhimanov, A. Gonoskov. ELMIS – a fully parallel Fourier-based multidimensional PIC code for laser-plasma interaction simulations // 22-nd Int. Conf. on Num. Sim. of Plasmas (ICNSP 2011), Long Branch, NJ, USA, September 7-9, 2011.

6. S. Bastrakov, R. Donchenko, A. Gonoskov, E. Efimenko, A. Malyshev, I. Meyerov, I. Surmin. Particle-in-cell plasma simulation on heterogeneous cluster systems // Journal of Computational Science. – 2012. – N3. – P. 474-479.
7. S. Bastrakov, I. Meyerov, V. Gergel et al. High performance computing in biomedical applications // Procedia Computer Science. – Elsevier, 2013. (В печати)
8. G. C. Fox. A review of automatic load balancing and decomposition methods for the Hypercube. InNumerical Algorithms for Modern Parallel Computer Architectures, pages 63–76. Springer-Verlag, 1988
9. J. Barnes, P. Hutt. A hierarchical O(N logN) force calculation algorithm // Nature, vol. 324, 1986. – p. 446–449.
10. D. B. Seidel, S. J. Plimpton, M. F. Pasik et al. Dynamic load balancing for a parallel electromagnetic particle-in-cell code // Proc. IEEE Intl. Pulsed Power Conf., 2001. – p. 1000-1003.
11. H. Nakashima, Y. Miyake, H. Usui, Y. Omura. OhHelp: a scalable domain-decomposing dynamic load balancing for particle-in-cell simulations // ICS-2009 Proceedings, 2009. – p. 90-99.
12. D. N. Nicol. Rectilinear partitioning of irregular data parallel computations // Journal of Parallel and Distributed Computing, vol. 23, 1994. – p. 119-134.
13. С. И. Бастраков, А. А. Гоносков, Р. В. Донченко, Е. С. Ефименко, А. С. Малышев, И. Б. Мееров. Исследование и поиск наиболее эффективных подходов к параллельному моделированию плазмы методом частиц в ячейках на кластерных системах // труды ПаВТ’2011 – Челябинск: Изд. центр ЮУрГУ, 2011. – С. 411-417.