ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 71 за 2016 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Андрианов А. Н., Ефимкин К. Н.

Метод частиц в ячейках: учет в параллельной реализации взаимодействия частиц

Рекомендуемая форма библиографической ссылки: Андрианов А. Н., Ефимкин К. Н. Метод частиц в ячейках: учет в параллельной реализации взаимодействия частиц // Препринты ИПМ им. М.В.Келдыша. 2016. № 71. 16 с. doi:10.20948/prepr-2016-71
URL: http://library.keldysh.ru/preprint.asp?id=2016-71

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

Андрианов А.Н., Ефимкин К.Н.

Метод частиц в ячейках: учет в параллельной реализации взаимодействия частиц

Москва — 2016

Андрианов А.Н., Ефимкин К.Н.

Метод частиц в ячейках: учет в параллельной реализации взаимодействия частиц

Рассматриваются вопросы построения параллельных программ для решения вычислительных задач с использованием метода частиц (Particles-In-Cell, PIC). Обсуждаются вопросы учета взаимодействия частиц в процессе расчета. Рассматриваются организация структур данных в параллельной программе. Приводятся результаты расчета модельной задачи на параллельном компьютере.

Ключевые слова: метод частиц в ячейках, суперкомпьютер, параллельное программирование

Alexander Nikolaevich Andrianov, Kirill Nikolaevich Efimkin.

Particles-in-cell method: accounting of particle interaction in parallel implementation

Development of parallel programs for computational tasks solution using the Particles-In-Cell (PIC) method is considered. Accounting of particle interaction in parallel implementation is discussed. Some problems of data structure organization are analyzed. Results of parallel execution of a model program are presented.

Key words: Particles-In-Cell, PIC, supercomputer, parallel programming.

Работа выполнена при поддержке Программы N3 Отделения математических наук РАН.

Оглавление

Введение 3
Модельная задача взаимодействия частиц 4
Задача установления связей между частицами 4
Структуры данных, используемые при установлении связей 5
Параллелизм расчет функций частиц 7
Распределение частиц между процессорами 8
Нумерации линий связей и параллелизм вычислений 11
Структура модельной программы 12
Временные результаты 14
Литература 16

Введение

Метод частиц (Particles In Cell, PIC) [1] получил в настоящее время широкое распространение при решении сложных задач математической физики [2-4]. В данной работе мы не исследуем собственно численный метод и особенности задач, которые решаются с его использованием. Проблема рассматривается с точки зрения программиста, перед которым поставлена задача построения параллельной программы для метода частиц. Математическая постановка прикладной задачи, послужившей исходной постановкой для данной работы, приведена в [5-6].

С точки зрения построения параллельных программ метод частиц не является простым. Использование данного метода предполагает, наряду со статическими объектами (регулярная статическая сетка), использование динамических объектов – частиц, которые могут двигаться, а также возникать и уничтожаться. Кроме того, частицы могут взаимодействовать. На каждом итерационном шаге решения задачи по времени, вычисления состоят из расчета на сетке и расчета частиц. Эти расчеты взаимосвязаны – при расчете на сетке используются результаты, полученные при расчете частиц и наоборот, что осложняет распараллеливание, так как необходимо согласовывать способы распараллеливания этих вычислений.

В [7] мы изложили ряд методов, которые можно применять при разработке параллельных программ для расчета методом частиц на многопроцессорных компьютерах. В частности, распределение по процессорам многопроцессорного компьютера расчетной сетки, распределение частиц, обеспечение наличия в процессоре требуемых для расчета частицы данных, вопросы хранения вычисленных результатов и т.п.

В данной работе основное внимание уделяется вопросам параллельной реализации метода частиц в условиях, когда частицы взаимодействуют. Неформально, взаимодействие частиц означает, что выполнено некоторое условие взаимодействия (не все частицы взаимодействуют друг с другом), и для частиц, удовлетворяющих условию взаимодействия, необходимо вычислить некоторую функцию, зависящую от атрибутов “участников” взаимодействия. Предложенные в данной работе способы создания параллельных расчетных программ опробованы на модельных задачах с числом взаимодействующих частиц порядка $10^6 - 10^7$ (число взаимодействий частиц порядка 10^9). Для разработки параллельных программ были использованы интерфейс передачи сообщений MPI [8], и стандарт OpenMP [9], а расчеты, в основном, проводились на вычислительном кластере К100 (ИПМ им.М.В. Келдыша РАН, [10]).

Модельная задача взаимодействия частиц

В дальнейшем используются следующие предположения.

Задана прямоугольная статическая трехмерная сетка по пространству. В ячейках этой сетки размещаются частицы. В общем случае, в ячейках сетки может располагаться различное число частиц, а также не обязательно, что в какой-то ячейке частицы присутствуют вообще. Необходимо решить две задачи.

1. ***Установить связи*** между частицами. В модельной задаче, связь между частицами считается установленной, если эти частицы находятся в соседних ячейках пространственной сетки.

2. Для каждой связи необходимо провести ***расчет функций частиц***, то есть некоторых функций для частиц, между которыми эта связь установлена.

В данной работе рассматриваются вопросы, возникающие при распараллеливании указанных выше задач, и предлагается подход к решению этих проблем.

Задача установления связей между частицами

Каждая частица связана с ячейкой пространственной сетки, в которой она находится. При установлении связей между частицами мы опираемся на тот факт, что для каждой частицы надо рассматривать возможность установления связи только с частицами, находящимися в соседних ячейках сетки.

Пусть частица находится в ячейке с индексными значениями (i, j)[1], тогда возможность установления связи рассматривается только для частиц, находящихся в ячейках с индексными значениями $(i+c_1, j+c_2)$, где c_1, c_2 целые константы из множества {-1,0,+1}. Установленная связь характеризуется парой (n_1, n_2), где n_1, n_2 – номера частиц, между которыми установлена связь. Следует отметить, что если установлена связь (n_1 ,n_2), то связь (n_2 ,n_1) устанавливать не надо, что важно для сокращения перебора при установлении связей.

Установление связей между частицами проводится следующим образом. Фиксируем ячейку сетки с координатами ***(i, j)***. Назовем ее *исходной ячейкой*. Далее, для каждой частицы из этой ячейки:

1) проводится проверка на возможность установления связи с другими частицами из исходной ячейки;

2) проводится проверка – возможно ли установление связей частиц из исходной ячейки с частицами из ячеек с координатами ***(i-1, j+1), (i, j+1), (i+1, j), (i+1, j+1)*** - *контрольными ячейками.*

Ниже, на рис. 1 схематично изображен результат установления связей для частицы из исходной ячейки с координатами ***(i, j)***. Сплошная линия соответствует установлению связей частицы из исходной ячейки ***(i, j)*** с частицами из контрольных ячеек. Пунктирные линии соответствуют

[1] *Для простоты изложения приводится двухмерный случай пространственной сетки*

установлению связей, когда ячейка с координатами ***(i, j)*** является контрольной ячейкой, а ячейки с координатами ***(i-1, j-1), (i-1, j), (i, j-1), (i+1, j-1)*** являются исходными.

Таким образом, видно, что проводится процесс установление связей для всех частиц из ячейки с исходными координатами.

Нетрудно заметить, что установление связи для каждой частицы из одной ячейки не зависит от процесса установления связи для частиц из других ячеек, то есть установление связей имеет высокую степень параллельности.

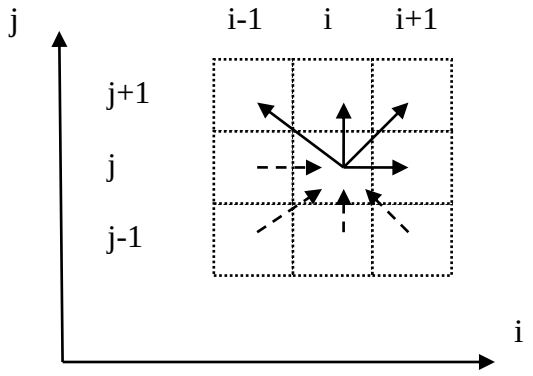

Рис. 1. Схема установления связей между частицами

Структуры данных, используемые при установлении связей

Решение задачи установления связей между частицами существенно опирается на знания о том, в какой ячейке сетки находится частица, и какие частицы рассматриваются в качестве кандидатов на установление связей. Поэтому, необходимо знать номера частиц, которые входят в ячейку сетки. Для представления этой информации введем вектор VECT_ATOM и матрицу MATR_CELL. Номера частиц, принадлежащих ячейке с индексами ***(i, j)***, последовательно записываются в вектор VECT_ATOM. При этом первый компонент, с которого записываются номера частиц, принадлежащих ячейке с индексами ***(i, j)***, указан в элементе матрицы MATR_CELL(1,i,j), а число частиц, принадлежащих этой ячейке, указывается в элементе матрицы MATR_CELL(2,i,j).

Ниже представлены описание и инициализация предлагаемых структур данных.

```
Integer pointer, :: VECT_ATOM(:)
Integer pointer, :: MATR_CELL (:,:,:,)
ALLOCATE (MATR_CELL (1:2,1:NX,1:NY))
ALLOCATE (VECT_ATOM (numatom[2]))
```

[2] *Значение* numatom *является некоторым положительным числом. В общем случае, следует определить некоторый параметр, и предусмотреть его динамическое увеличение*

На рис. 2 иллюстрируется связь вектора VECT_ATOM и матрицы MATR_CELL. Начиная с компонента N1 в векторе VECT_ATOM, расположены номера частиц, размещенных в ячейке с координатами ***(i, j)***. Начиная с компонента N2 в векторе VECT_ATOM, расположены номера частиц, размещенных в ячейке с координатами ***(i+1, j)***.

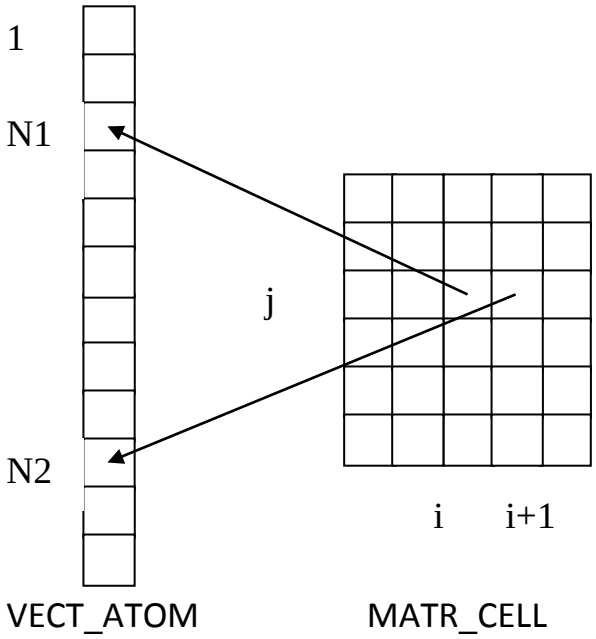

Рис. 2. Структуры данных, поддерживающие расположение частиц в ячейках

При этом

MATR_CELL (1,i,j) = N1
MATR_CELL (1,i,j+1) = N2
MATR_CELL (2,i,j)[3] = N2 – N1 +1

В результате выполнения действий установление связей между частицами, строится матрица CONNECTION_LINE. Описание и инициализация этого матрицы имеет следующий вид:

integer, pointer :: CONNECTION_LINE(:,:) (1)
allocate (CONNECTION_LINE_R(1:2, 1:num_line[4])

Здесь каждая строка соответствует одной установленной связи. В строке указываются номера двух частиц $\boldsymbol{n_1}$ и $\boldsymbol{n_2}$, между которыми установлена связь.

[3] *Конечно, компонент матрицы MATR_CELL(2,i,j) - количество частиц в ячейке – избыточен, и его можно определять на основе значения в ячейке MATR_CELL(1,i,j)*

[4] *Эта величина первоначально имеет некоторое положительное значение*

Размерность (по второму направлению) матрицы CONNECTION_LINE заранее, конечно, не известна – это количество связей, которые будут установлены в решаемой задаче. Поэтому, необходимо предусмотреть возможность динамического увеличения этой размерности.

Параллелизм расчета функций частиц

В данной работе мы не останавливаемся на точной фиксации используемых в конкретном расчете функций. В основном, мы обращаем внимание на общие зависимости по данным, существующие в функциях, и задаче параллельного расчета функций по всем выявленным (установленным) связям.

Обозначим множество линий связи, в которые входит частица с номером n_1, через $line(n_1)$. И определим расчет некоторой переменной U для частицы с номером n_1 следующим образом:

$$U(n_1) = \sum_{k \in line(n_1)} F(k), \qquad (2)$$

где k – номер линии связи, в которой задействована частица с номером n_1, $F(k)$- некоторая функция, вид которой определятся конкретной задачей.

Заметим, что рассматривая фиксированную линию связи (n_1, n_2), можно одновременно проводить вычисления соответствующей переменной U как для частицы с номером n_1, так и для частицы с номером n_2.

Из представления (2) следует, что нельзя одновременно (параллельно) вычислять переменную U для линий связи $line$, в которых задействована одна и та же частица. С другой стороны, если все множество линий связей разбить на непересекающиеся (по использованию частиц) подмножества, то тогда в каждом из этих подмножеств можно проводить вычисления на линиях связей независимо от вычислений в других подмножествах, что и определяет возможный параллелизм вычислений.

Один из способов построения таких независимых (по вычислениям) подмножеств опирается на следующее соображение. Когда рассматривается ячейка с координатами (i, j), то в построенных линиях связей для частиц из этой ячейки не будет, например, ни одной частицы из ячейки с координатами $(i+2, j)$. Следовательно, линии связей, построенные из ячейки с координатами (i, j), можно отнести к одному подмножеству, а линии связей, построенные для частиц из ячейки с координатами $(i+2, j)$, можно отнести к другому (независимому) подмножеству. Вычисления на линиях связей для полученных независимых подмножеств можно проводить параллельно и независимо.

Параллелизм вычислений на линиях связей обсуждается также ниже, в разделе о нумерации линий связей.

Распределение частиц между процессорами

Ниже приведена схема одного из возможных способов распараллеливания при решении задачи распределения частиц с учетом их связей.

Этот способ основан на распределении частиц по процессорам многопроцессорной системы. Поскольку частицы, расположенные в некоторой ячейке, связаны с частицами из соседних ячеек, задача распределения частиц можно свести к задаче распределения узлов исходной пространственной сетки по процессорам.

Замечание. При этом распределение сетки по процессорам (для расчета сеточных значений) может проводиться по другим правилам, и распределение сетки в этом случае может иметь другой вид.

Общая схема распределения расчетной сетки по процессорам приведена на рис. 3.

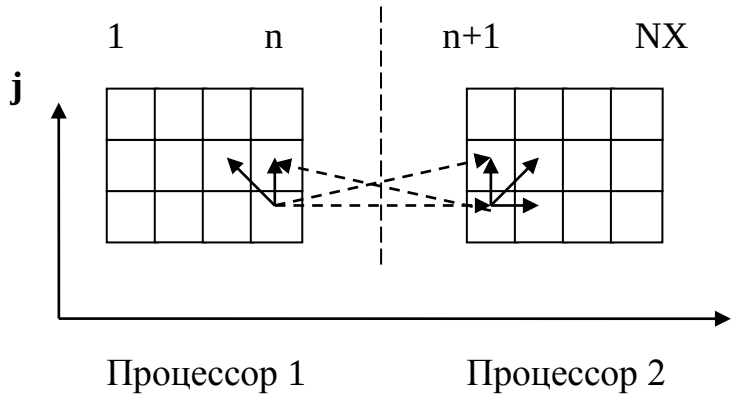

Рис. 3. Схема распределения расчетной сетки по процессорам.

Пусть мы хотим решать задачу на *iproc* процессорах. Разобьем исходную сетку по одному[5] направлению на *iproc* частей, и получим таким образом *iproc* “полос” на плоскости счета. Размеры (“ширина”) этих “полос” не обязательно одинаковые. Это обусловлено тем, что, распределяя сетку между процессорами, мы стремимся к тому, чтобы в каждой “полосе” было примерно равное число частиц. На рис. 3. видно, что часть требуемых для установления связей данных могут находиться в разных процессорах. Эти связи обозначены пунктирными линиями.

Конечно, распределяя сетку по процессорам, необходимо обеспечивать в каждом процессоре необходимые для расчета *в этом процессоре* данные,

[5] *При разбиении по одному направлению проще организовывать обмен необходимыми данными между процессорами - меньше количество операторов обмена. Возможно разбиение и по большему числу направлений, что иногда может дать определенный выигрыш в эффективности по времени счета*

которые были распределены и вычисляются *в других процессорах*. На рис. 4 представлена схема разбиения сетки на два процессора, с “теневыми гранями”, которые используются для хранения *требуемых в данном процессоре данных*, которые *вычисляются в других процессорах*. Разбиение проводится по оси ***i***. В первом процессоре размещаются точки исходной расчетной сетки с координатами по оси ***i*** от **1** до **n**. Во втором процессоре размещаются точки исходной сетки с координатами по оси ***i*** от **n+1** до **NX**. Поскольку в задаче установления связей для ячейки сетки с координатой **n** (по оси ***i***), необходимы значения из ячейки с координатой **n+1** (частицы и их атрибуты), то процессора с номером 1 необходимо:

1) предусмотреть в *текущем процессоре* место для размещения необходимых *удаленных* данных (так называемая *теневая грань*); в отличие от традиционного сеточного разбиения, в рассматриваемой задаче теневая грань должна содержать необходимую информацию *обо всех* требуемых частицах;

2) провести пересылку из второго процессора в первый этих удаленных данных.

Аналогичные действия необходимы и для второго процессора.

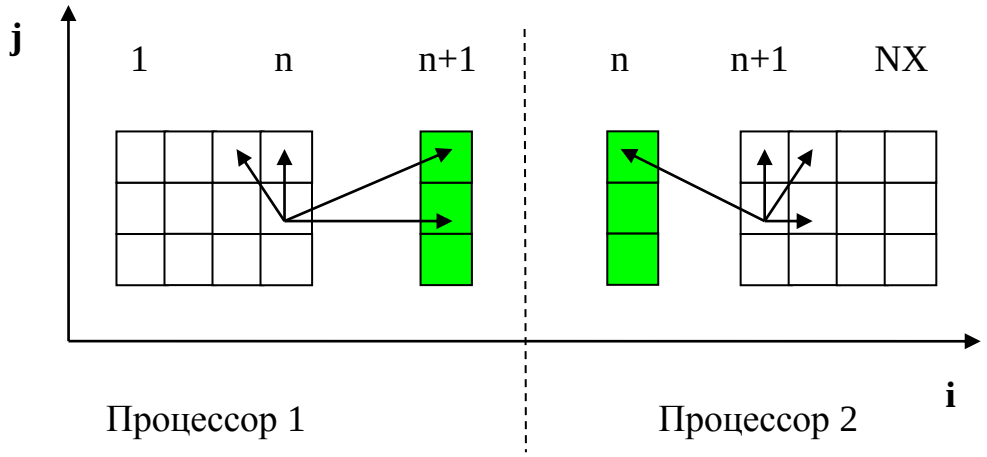

Рис. 4. Схема распределения расчетной сетки с теневыми гранями

Распределяя частицы между процессорами, необходимо обеспечить корректировку всех используемых структур данных, поддерживающих решение рассматриваемой задачи.

Предложенный способ распараллеливания опирается на модель и средства распараллеливания интерфейса передачи сообщений MPI [8], и ориентирован на модель параллельности с распределенной памятью.

Другой возможный способ распараллеливания опирается на стандарт распараллеливания программ OpenMP [9], и ориентирован на модель параллельности с общей памятью.

Разобьем исходную расчетную сетку по направлению **j** на **k** непересекающихся “полос” (см. рис. 5). Как и при первом способе, в каждой

“полосе” (подмножестве сетки) задача определения связей решается независимо от других подмножеств. При этом, для каждого подмножества необходимо ввести свою структуру данных, для хранения установленных связей (например, массив CONNECTION_LINE, см. выше (1)). Количество подмножеств может соответствовать числу параллельных нитей OpenMP, физически поддерживаемых на используемом компьютере[6]. В данном случае, так же, как и в первом способе, при разбиении исходной расчетной сетки можно основываться на количестве частиц в построенных подмножествах. Требуемые для расчета данные в этом случае подкачивать не надо (в отличие от распределения для распределенной памяти).

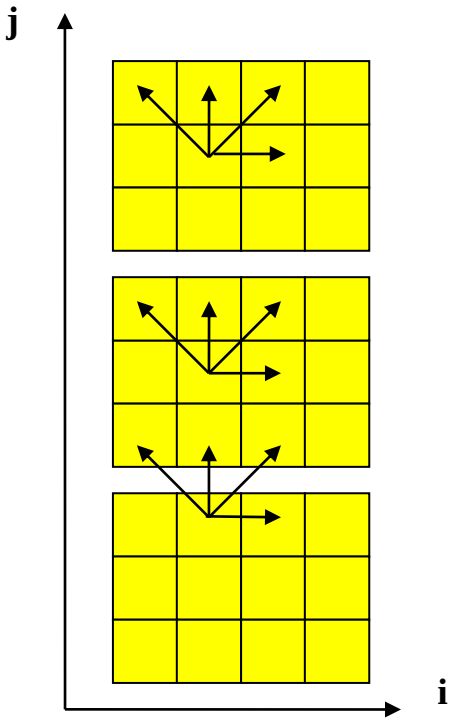

Рис. 5. Схема распределения расчетной сетки в модели OpenMP

Использование второго способа может быть полезно в случае большого объема обрабатываемых данных (количество частиц и построенных линий связи), например, в сочетании с первым способом. В этом случае первоначально проводится разбиение исходных данных *по узлам* многопроцессорной системы, с использованием механизма передачи сообщений MPI. Далее, в рамках одного узла, для эффективного использования ядер одного процессора, можно использовать второй способ распараллеливания, средствами стандарта OpenMP.

---

[6] *Это число может задаваться и как параметр задачи при счете на конкретной вычислительной системе*

Возможно также использование графических ускорителей (GPU) для ускорения вычислений при решения задачи расчета функций.

Следует отметить, что при использовании метода частиц, в процессе расчета *число частиц может меняться* – могут возникать новые частицы, а также, существующие частицы, при определенных условиях, могут удаляться. Кроме этого, *частицы меняют свои координаты* и, как следствие, перемещаются в другие ячейки сетки, что может приводить к дисбалансу статического распределения частиц. Решение этих проблем требует обеспечить в параллельной программе возможность динамической балансировки распределения (см. например [7]) и, как следствие, корректировку используемых структур данных.

Нумерации линий связей и параллелизм вычислений

В параллельной реализации способ нумерация линий связей является существенным фактором, влияющим на эффективность распараллеливания.

Определим возможности параллельной обработки при нумерации линий связей предложенным способом. Рассмотрим линии связи, построенные при фиксированном значении координаты $j = j_0$. Пусть выявлена линия связи (n_1, n_2). Координаты частиц n_1 и n_2 (по оси **j**) по построению могут иметь одно из двух значений: $j = j_0$ или $j = j_0 + 1$. Возьмем точку с координатой $j = j_0 + 2$. Тогда координаты (по оси **j**) частиц, при установленной связи, могут иметь одно из значений - $j = j_0 + 2$ или $j = j_0 + 3$. В результате, частицы, составляющие линии связи для первого случая (при $j = j_0$), не используются в линии связей во втором случае ($j = j_0 + 2$). Таким образом, линии связи, построенные при $j = j_0$ могут обрабатываться параллельно и независимо от линий связи, построенных при $j = j_0 + 2$.

Этот факт делает возможной следующую схему параллельных вычислений функций на линиях связи.

Разобьем исходную область (по направлению **j**) на два подмножества. Первое будет состоять из линий связи, образованных в ячейках сетки с нечетными значениями индекса **j**, а второе - из линий связи, образованных в ячейках сетки с четными значениями индекса **j**. Для элементов этих подмножеств справедливо утверждение - линии связи первого подмножества не имеют общих номеров частиц с элементами второго подмножества. Таким образом, возможны 2 последовательных вычисления (по четным индексам и по нечетным индексам), в каждом из которых вычисления по индексам параллельны.

В программной реализации обход ячеек сетки и установление связей между частицами можно выполнять по схеме, задаваемой следующими операторами.

```
ipos= 0
do  j = 1,NY
    do i = 1,NX
          call HYPERPLANE(i,j,ipos)
    enddo                                          !  i = 1,NX
    FIXED_LINE(j) = ipos
enddo                                                  !  j = 1,NY
```

Здесь переменная ipos используется для подсчета общего числа установленных связей, подпрограмма HYPERPLANE(i,j,ipos) определяет наличие связи, и управляет счетчиком ipos нумерации установленных связей.

Компонент вектора FIXED_LINE(j) используется для фиксации последнего номера связи, установленной при фиксированном значении индекса j (FIXED_LINE(0) = 1), и служит основой для разбиения всего множества полученных линий связей на непересекающиеся подмножества. Линии связи, построенные при фиксированном значении $\boldsymbol{j = j_0}$ располагаются в массиве CONNECTION_LINE, см. выше (1), начиная со строки с номером

$$\mathbf{j_n} = \text{FIXED\_LINE}(\mathbf{j_0 - 1}) \mathbf{+1} ,$$

и заканчивается строкой с номером

$$\mathbf{j_k} = \text{FIXED\_LINE}(\mathbf{j_0}) .$$

Структура модельной программы

Для оценки предлагаемого подхода была разработана параллельная программа решения модельной задачи расчета методом частиц со взаимодействием, общая структура которой приведена ниже. Программная реализация выполнена с использованием механизма передачи сообщений MPI [8] и стандарта OpenMP [9].

Программа состоит из итеративного цикла istep по времени счета Nstep, в котором последовательно проводится вызов ряда подпрограмм.

```
do istep=1, Nstep
        if (mod(istep,10).eq.0)then
              call REDEFINE_ATOM
              call BUILD_MATR_CELL
              call COLLECT_DATA_BUFFER
              call FIND_NEIGHBOUR
        endif
              call CALC_RADIUS
enddo
```

Вычисление функций частиц (подпрограмма CALC_RADIUS) проводится на каждом итерационном шаге. Все остальные подпрограммы выполняются один раз через каждые 10 итерационных шагов. В этих подпрограммах проводится перераспределение частиц между процессорами (REDEFINE_ATOM), построение структур данных (BUILD_MATR_CELL). заполнение теневых граней (COLLECT_DATA_BUFFER), и установление связей между частицами (FIND_NEIGHBOUR).

***Подпрограмма* REDEFINE_ATOM**

Каждый процесс проводит обработку атомов, размещенных в ячейках, координаты которых лежат в фиксированном диапазоне (по оси Z).

В подпрограмме проводятся действия, обеспечивающие равномерное распределение частиц по параллельным процессам. Сначала считается число частиц в каждом процессе, и все процессы обмениваются этими значениями. На основании этой информации строится новое распределение частиц по процессам. Далее, каждый процесс строит локальные карту приема и отправки атрибутов частиц. на основе этих карт определяются размерности буфером приема и отправки значений, и проводятся операции обмена данными. После проведения операций обмена данными проводится корректировка структур данных – векторов VECT-ATOM и матрицы MATR_CELL.

***Подпрограмма* COLLECT_DATA_BUFFER**

В подпрограмме проводится подкачка значений теневых граней. Так как для расчета линий связи необходимы только значения координат частиц, остальные атрибуты частиц не подкачиваются.

***Подпрограмма* BUILD_MATR_CELL**

В подпрограмме проводится построение структур данных – матрицы MATR_CELL и вектора VECT_ATOM.

***Подпрограмма* FIND_NEIGHBOUR**

В подпрограмме проводится построение линий связей между частицами: заполнение структуры CONNECTION_LINE, см. выше (1). В этой подпрограмме, при построение линий связей, реализуются параллельные вычисления средствами стандарта OpenMP. Каждая параллельная нить строит свой локальный вектор результатов, и по завершении этого параллельного вычисления все полученные локальные результаты помещаются в единый вектор CONNECTION_LINE. Кроме этого, строится вектор FIXED_LINE используемый для разбиения всего множества полученных линий связей на непересекающиеся подмножества.

***Подпрограмма* CALC_RADIUS**

В подпрограмме проводится вычисление функций частиц для всех построенных линий связей в текущем процессе. Как и в случае построения линий связи, обработка линий связи распараллеливается средствами стандарта OpenMP.

Результаты применения метода

Приведенная выше схема создания параллельных программ для решения вычислительных задач с использованием метода частиц со взаимодействием была опробована на модельной задаче, на вычислительном кластере К-100, установленном в ИПМ им. М.В,Келдыша РАН [10].

Ниже приведены некоторые результаты времени выполнения расчетов, полученных для следующих параметров:

- размеры трехмерной расчетной сетки – 8 млн. точек (200x200x400);
- в начальный момент времени частицы расположены в подсетке исходной сетки с координатами (50..150)x(50..150)x(200..300);
- в каждой ячейке этой подсетки размещается 64 частицы;
- общее число частиц - 64 млн.

Расчет состоял из 200 итеративных шагов по времени. Каждые 10 шагов значение координаты каждой частицы (по оси Z) увеличивалось на 1.0. Далее проводилось установление связей между частицами. Расчет функций частиц проводился на каждом шаге по времени. Общее число установленных связей равно 7586611200.

Результаты замеров времени вычислений приводятся в Таблице 1 ниже.

В первой строке указано число процессоров, на которых проводились вычисления.

Во второй строке (Объем ОП) указан общий объем в мегабайтах (Mb) оперативной памяти, используемой одни процессом.

В третьей строке (Общее время) указано общее время, затраченное на решение задачи.

В четвертой строке (Время FIND_NEIGHBOROUGH) указано время, затраченное на установление связей между частицами.

Время, затраченное на расчет функций частиц, приводится в пятой строке (Время CALC_RADIUS).

В шестой строке (Время REDEFINE) указано время, использованное для построения карт отправки и приема частиц между процессорами, а также время, затраченное на перемещение частиц между процессорами.

В седьмой строке (Время COLLECT) приводится время, затраченное на построение теневых граней.

Время построения новых структур (MATR_CELL и VECT_ATOM) представлено в последней, восьмой, строке Таблицы (Время BUILD_MATR_CELL).

Таблица 1

Результаты расчета методом частиц со взаимодействием

Число процессоров	**10**	**20**	**30**	**40**	**50**
Объем ОП (Mb)	12975	10007	8821	8821	8227
Общее время	1441.32	846.11	633.21	508.48	416.30
Время FIND_NEIGHBOROUGH	97.33	63.42	49.26	40.72	35.30
(Время CALC_RADIUS	1173.75	657.47	406.27	319.77	263.09
Время REDEFINE	8.95	7.99	6.50	5.92	6.29
Время COLLECT	5.28	5.52	5.56	5.52	6.76
Время BUILD_MATR_CELL	8.96	4.24	2.73	2.14	1.72

Приведенные результаты показывают, что предлагаемые метод параллельного расчета методом частиц дают достаточно хорошие по времени выполнения программы, а также позволяет проводить расчеты с достаточно большим числом взаимодействующих частиц.

Литература

1. Ф.Х. Харлоу. Численный метод частиц в ячейках для задач гидродинамики // Вычислительные методы в гидродинамике. — М.: Мир, 1967, 460 с.
2. Вшивков В.А., Краева М.А., Малышкин В.Э. Параллельная реализация метода частиц. // Программирование, 1997, № 2, с. 39-51.
3. Андрианов А.Н., Березин А.В., Воронцов А.С., Ефимкин К.Н., Марков М.Б. Моделирование электромагнитных полей радиационного происхождения на многопроцессорных вычислительных системах. // Математическое моделирование, 2008, т. 20, № 3, с. 98-114.
4. Киреев С.Е. Параллельная реализация метода частиц в ячейках для моделирования задач гравитационной космодинамики. // Автометрия, 2006, т. 42, № 3, с. 32-39.
5. Мажукин В.И., Шапранов А.В. Математическое моделирование процессов нагрева и плавления металлов. Часть I. Модель и вычислительный алгоритм // Препринты ИПМ им.М.В.Келдыша. — 2012. — № 31. — 27с.
6. Мажукин В.И., Шапранов А.В. Молекулярно-динамическое моделирование процессов нагрева и плавления металлов. Часть II. Вычислительный эксперимент // Препринты ИПМ им.М.В.Келдыша. — 2012. — № 32. — 25с.
7. Андрианов А.Н., Ефимкин К.Н. Подход к параллельной реализации метода частиц в ячейках // Препринты ИПМ им.М.В.Келдыша. — 2009. — № 9. — 21с.
8. MPI. Message Passing Interface Forum. http://www.mpi-forum.org/
9. OpenMP Specifications for parallel programming. http://openmp.org/wp/
10. Вычислительный кластер К-100. http://www.kiam.ru/MVS/resourses/