ИПМ им.М.В.Келдыша РАН

Абрау-2016 • Труды конференции

А.Н. Андрианов, Е.С. Анискова,
К.Н. Ефимкин

О параллельной реализации метода частиц в ячейках

Рекомендуемая форма библиографической ссылки

Андрианов А.Н., Анискова Е.С., Ефимкин К.Н. О параллельной реализации метода частиц в ячейках // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции (19-24 сентября 2016 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2016. — С. 14-22. — doi:10.20948/abrau-2016-36

Размещена также презентация к докладу

О параллельной реализации метода частиц в ячейках

А.Н. Андрианов, Е.С. Анискова, К.Н. Ефимкин

ФИЦ ИПМ им.М.В. Келдыша РАН

Аннотация. 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. Приводятся результаты расчета модельной задачи на параллельных компьютерах.

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

1. 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 этот метод 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 [3, 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, возникать и уничтожаться, а также взаимодействовать между собой. 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, при расчете частиц используются значения сеточных переменных. Такое взаимодействие осложняет распараллеливание, так как необходимо согласовывать способы распараллеливания этих вычислений.

Предлагаемые методы распараллеливания ориентированы на расчет с таким числом частиц, когда ресурс оперативной памяти становится критичным – увеличение числа частиц (или числа взаимодействий частиц) может привести к необходимости использования внешней памяти и свопингу, что резко увеличивает время счета. Одна из основных целей проведенного исследования – определение структур данных и методов параллельных вычислений на этих

данных, которые: 1) позволяют проводить вычисления в оперативной памяти и избежать свопинга и 2) эффективно использовать возможности параллельной вычислительной техники. Конечно, архитектуры параллельных вычислителей весьма разнообразны, и мы не претендуем на всеобъемлющее исследование, а рассматриваем параллельные архитектуры с общей памятью, с распределенной памятью, и их комбинации.

Рассмотренные в данной работе способы создания параллельных программ опробованы на прикладных и модельных задачах с числом ячеек сетки порядка $10^8 - 10^9$, с числом частиц порядка $10^8 - 10^{11}$ (без взаимодействия частиц) и с числом взаимодействующих частиц порядка $10^6 - 10^7$ (число взаимодействий частиц порядка 10^9). Например, предлагаемый метод был реализован авторами в ИПМ им.М.В. Келдыша РАН, в параллельном расчетном комплексе для моделирования электромагнитных полей радиационного происхождения [3]. Некоторая информация о результатах решения предлагаемым метод расчетных задач приведена также ниже, в п.2 и п.3.

Для разработки параллельных программ были использованы интерфейс передачи сообщений MPI [8] и стандарт OpenMP [9], расчеты в основном проводились на вычислительном кластере К100 (ИПМ им.М.В. Келдыша РАН, [10]) и суперкомпьютере "Чебышев" (НИВЦ МГУ) .

2. Метод параллельного расчета частиц без взаимодействия

При расчете частиц без взаимодействия, при распараллеливании статической прямоугольной расчетной сетки можно применить известный метод распараллеливания по данным - проблема распараллеливания расчетов на регулярных статических сетках достаточно хорошо изучена, и полученные результаты широко применяются.

Однако при расчетах методом частиц возникает ряд новых проблем. Во-первых, число частиц в процессе решения динамически изменяется. Частицы порождаются и уничтожаются неравномерно в различных ячейках сетки. Во-вторых, в ходе вычислений частицы перемещаются между ячейками в соответствии с уравнениями движения, распределяясь по ним также неравномерно.

Расчетом частицы будем называть вычисление расчетных переменных, характеризующих частицу, для фиксированной частицы. Все частицы связаны с ячейками исходной сетки задачи. По координатам частицы однозначно можно определить номер ячейки сетки, в которой эта частица находится в текущий момент моделирования.

Для проведения расчета частицы необходимы значения расчетных переменных частицы с предыдущего временного шага моделирования, а также значения из некоторого числа ячеек исходной сетки задачи. Номера таких ячеек обычно задаются шаблоном, применяемым при решении задачи. В рассматриваемом классе задач для частицы, находящейся в ячейке с

координатами (i, j, k), требуются значения сеточных функций из ячеек с координатами $(i \pm 1, j \pm 1, k \pm 1)$.

Для хранения расчетных переменных частицы используются одномерные массивы (вектора), длина которых должна быть равна числу частиц, и, следовательно, должен быть механизм динамического управления памятью для хранения этих переменных, так как число частиц меняется в процессе счета.

Замечание. При решении задач, в которых число частиц является достаточно большим ($10^8 - 10^{11}$), способ выделения памяти для вектора, предназначенного для хранения переменных частиц, становится важным, и, как правило, требуется применение динамического управления памятью.

После завершения расчета частицы обновляются значения расчетных переменных частицы на текущем временном шаге моделирования. Кроме того, вычисляются необходимые сеточные значения в ячейках, на которые оказывает влияние данная частица.

Таким образом, для расчета частицы требуются сеточные значения, а сеточные значения используют результаты вычисления частиц.

Поскольку расчет каждой частицы не зависит от результатов расчета других частиц, то расчет всех частиц можно проводить независимо (параллельно) друг от друга. При этом, необходимо учитывать тот факт, что может сложиться такая ситуация, когда разные частицы вычисляют сеточные значения для одной и той же ячейки сетки.

Для параллельного выполнение расчета частиц необходимо решить следующие задачи:

1) ***распределение частиц*** - распределить частицы по процессорам вычислительной системы, на которой решается задача;

2) ***обеспечение требуемых данных*** - обеспечить расчет частиц всеми необходимыми исходными данными, которые используются при расчете частиц;

3) ***расчет значений частиц*** – проведение собственно расчета;

4) ***возврат результатов расчета частиц*** - размещение вычисленных при расчете частиц значений в соответствующей памяти (которые часто находятся в другом процессоре);

5) ***расчет сеточных значений***, которые зависят от значений частиц.

Расчет частицы можно проводить на том же процессоре, в котором находится ячейка сетки, содержащая данную частицу. Далее такое распределение будем называть **распределением "по месту"**. При этом может, конечно, возникнуть ситуация, когда вычисления будут сильно разбалансированы по времени, так как в процессе моделирования число частиц меняется, и распределение их по ячейкам сетки может быть произвольным.

Второй подход к распределению частиц учитывает время счета задачи на предыдущем шаге решения и общее число частиц. Далее такое распределение будем называть **распределением "по времени"**. Частный случай такого подхода состоит в том, что, зная общее количество частиц, равномерно

распределить их по всем процессорам, на которых решается задача. Такое распределение частиц в дальнейшем будем называть **равномерным распределением**.

Распределение частиц основано на распределении расчетной сетки, в которой частицы находятся (при этом распределение между процессорами расчетной сетки и распределение частиц, вообще говоря, не совпадают). В предлагаемом алгоритме распределение частиц проводится на линейку процессоров по третьему пространственному измерению **z**, и основано на определении (для каждого процессора с номером ***r***, ***r = 1..P***, ***P*** - число процессоров) диапазона $\left[z_n^r, z_k^r\right]$, который содержит одинаковое, в определенном смысле, число частиц.

Способы определения диапазонов $\left[z_n^r, z_k^r\right]$ для ***равномерного*** распределения, распределения ***по времени***, распределения ***по месту*** различные**.**

При распределении ***по времени***, с каждым процессором связывается дополнительно весовой коэффициент, зависящий от времени расчета частиц на этом процессоре на предыдущем итерационном шаге.

Распределения частиц ***по месту*** самое “простое” - искомый диапазон для каждого процессора совпадает с диапазоном, задающим распределение сетки задачи по процессорам.

На основе полученных диапазонов проводится построение карт отправки и приема частиц. Карта отправки состоит из записей (N_r , P_r), где N_r – номер процессора, в который отправляются частицы из текущего процессора, P_r – количество частиц, которые надо отправить в этот процессор. Аналогичный вид имеет карта приема.

Перемещение частицы из одного процессора в другой состоит в "переносе" всех значений расчетных переменных, требуемых для расчета частицы, в тот процессор, в который направляется эта частица.

Технические подробности описываемых методов приведены в [7].

Для проведения расчета каждой частицы обычно **требуются** следующие исходные значения:

1) значения расчетных переменных частицы с предыдущего временного шага моделирования (*предыдущие* ***значения частиц***). С каждой частицей обычно связано порядка 10 значений переменных (это зависит от задачи);

2) значения из некоторого числа соседних ячеек сетки (***сеточные значения***); соседние ячейки обычно задаются шаблоном, применяемым при решении задачи. Для определенности будем считать, что для частицы, находящейся в ячейке с координатами (i, j, k), требуются сеточные значения из ячеек с координатами $(i \pm 1, j \pm 1, k \pm 1)$.

Таким образом, при перераспределении по процессорам частиц для расчета, необходимо обеспечить подкачку в процессор соответствующих *значений частиц* и *сеточных значений*.

Определение *сеточных значений*, которые необходимо подкачать, основывается на информации о распределении частиц, обсуждавшемся выше. Заметим, что даже при распределении частиц ***по месту*** необходимо проводить подкачку требуемых *сеточных значений*.

Результатом расчета каждой частицы являются следующие значения:

1) значения расчетных переменных частицы на текущем временном шаге моделирования (*текущие* ***значения частицы***);

2) значения, которые являются частью интегральных функций в соседних ячейках, на которые оказывает влияние данная частица (***сеточные значения***).

Для хранения *текущих значений частицы* используются динамические вектора, а для хранения вычисленных *сеточных значений* возможен следующий способ. Для вычисляемых сеточных значений отводятся вектора, и определяются специальные правила их “плотного” заполнении. Правила представления информации в этих векторах (в том числе вспомогательная функция специальной нумерации) обеспечивают хранение только необходимых вычисленных значений и позволяют экономить память за счет несущественного, с практической точки зрения, увеличения времени обращения к векторам. Наши практические эксперименты показали, что этот способ, в условиях дефицита памяти, оказывается полезен.

После завершения расчета частиц на текущем шаге моделирования, каждая частица (точнее, *текущие значения частицы*) остается в том же процессоре, в котором проводился ее расчет. Вычисленные при расчете частиц *сеточные значения* связаны с ячейками сетки задачи, поэтому эти значения необходимо вернуть в те процессоры, в которые эти ячейки сетки распределены, так как распределение ячеек сетки, в общем случае, не совпадает с распределением частиц. Как и в случае с перемещением частиц между процессорами, строятся карты отправки и приема сеточных значений.

Рассматриваемые методы распараллеливания опробованы практически, в частности, на суперкомпьютере “Чебышев” (НИВЦ МГУ) [11].

Параметры одной из решаемых задач [3] следующие. Расчетная сетка задачи имела размерность 273×273×655. Проводился расчет 6000 итерационных шагов. К концу расчета общее число частиц составляло 1 391 253 130. Общий объем последней контрольной точки 85.32 Gb, время записи этой контрольной точки составило 59 сек. Вычисления проводились на 328 процессорах.

Исследования свойств рассмотренных методов проводились для 3-х вариантов использования процессоров - 82, 164 и 328 процессоров.

Наилучшая общая эффективность достигается при использовании метода ***распределения по времени***. Это не является неожиданным, так как метод ***распределения по времени*** является наиболее “тонким”, учитывающим затраты времени наиболее аккуратно.

В докладе более подробно рассматриваются полученные результаты, характеристики эффективности предложенных методов распараллеливания.

3. Метод параллельного расчета частиц с взаимодействия частиц

Достаточно часто при решении задач математической физики метод частиц используется 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]).

Предлагаемый метод основан на следующих предположениях.

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 необязательно, что в какой-то ячейке частицы присутствуют вообще. Необходимо решить две задачи:

1) ***установить связи*** между частицами,

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

При решении задачи ***установления связей*** вводится трехмерная сетка по пространству. Размер ячейки не превосходит заданной величины $\boldsymbol{r}$. Считается, что две частицы взаимодействуют друг с другом, если расстояние между ними не превосходит заданной величины $\boldsymbol{r}$. Из условия построения сетки следует, что для определения взаимодействия частиц из фиксированной ячейки необходимо рассматривать частицы, размещенные только в соседних ячейках.

Для простоты изложения будем рассматривать двухмерный случай пространственной сетки. Пусть частица находится в ячейке с индексными значениями $\boldsymbol{(i,j)}$, тогда возможность установления связи рассматривается только для частиц, находящихся в ячейках с индексными значениями $\boldsymbol{(i+c_1, j+c_2)}$, где $\boldsymbol{c_1, c_2}$ целые константы из множества {-1,0,+1}. Установленная связь характеризуется парой $\boldsymbol{(n_1, n_2)}$, где $\boldsymbol{n_1, n_2}$ – номера частиц, между которыми установлена связь. Следует отметить, что если установлена связь ($\boldsymbol{n_1, n_2}$), то связь ($\boldsymbol{n_2, n_1}$) устанавливать не надо.

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.*

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

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. Вообще говоря, задача установления связей является затратной по времени (даже в используемых нами сильно упрощающих предположениях), поэтому существенными становятся способы представления данных и обращения к ним. Вопросы описания в программе и использование структур данных подробнее рассматривается в докладе.

В данной работе мы не останавливаемся на точной фиксации используемых в конкретном расчете математических расчетных функций. Учитываются лишь зависимости по данным, существующие в функциях, и решается задача 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:

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

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 приведенного выше представления 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 основан на следующем соображении. 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.

Рассматриваемый выше метод разработки 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 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).

Число процессоров	**10**	**20**	**30**	**40**	**50**
Объем ОП (Mb)	12975	10007	8821	8821	8227
Общее время	678.22	357.12	244.19	224.58	161.84
Время FIND_NEIGHBOROUGH	176.65	96.20	79.39	64.14	47.58
Время CALC_RADIUS	475.56	241.23	147.12	146.77	99.60
Время REDEFINE	0.01	0.01	0.03	0.03	0.04
Время COLLECT	6.53	6.97	6.63	6.00	6.42
Время BUILD_MATR_CELL	16.88	8.73	6.10	6.47	3.78

Приведенные результаты показывают, что предлагаемый метод параллельного расчета методом частиц со взаимодействием дает неплохие (по времени выполнения программы) результаты, а также позволяет проводить расчеты с достаточно большим числом взаимодействующих частиц.

Работа выполнена при поддержке Программы N 3 Отделения математических наук РАН.

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. — 27 с.
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. — 25 с.
7. Андрианов А.Н., Ефимкин К.Н. Подход к параллельной реализации метода частиц в ячейках // Препринты ИПМ им. М.В.Келдыша. — 2009. — № 9. — 21 с.
8. MPI. Message Passing Interface Forum. http://www.mpi-forum.org/
9. OpenMP Specifications for parallel programming. http://openmp.org/wp/
10. Вычислительный кластер К100. http://www.kiam.ru/
11. СКИФ МГУ “Чебышев”. https://parallel.ru/cluster/actual.html