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

Препринты ИПМ • Препринт № 9 за 2009 г.

Андрианов А.Н., Ефимкин К.Н.

Подход к параллельной реализации метода частиц в ячейках

Рекомендуемая форма библиографической ссылки: Андрианов А.Н., Ефимкин К.Н. Подход к параллельной реализации метода частиц в ячейках // Препринты ИПМ им. М.В.Келдыша. 2009. № 9. 20 с. URL: http://library.keldysh.ru/preprint.asp?id=2009-9

А. Н. Андрианов, К. Н. Ефимкин

Подход к параллельной реализации метода частиц в ячейках[♦]

Аннотация

В данной работе рассматриваются некоторые проблемы построения параллельных программ для решения вычислительных задач с использованием метода частиц (Particles In Cell, PIC). Обсуждаются вопросы распределения данных, балансировки вычислений. Приведенные результаты получены в ИПМ им. М.В.Келдыша РАН при создании и использовании комплекса параллельных программ для решения задач электродинамики.

A.N. Andrianov, K.N. Efimkin

Approach to parallel implementation of the particles in cell method

Abstract

Some problems of parallel program creation for numerical tasks solution using the Particles In Cell (PIC) method are analyzed. The problems of data distribution and load balancing are discussed. The results of this work were obtained in Keldysh Institute of Applied Mathematics in a process of creation and usage of the parallel solver for electrodynamics tasks.

[♦] Работа выполнена при частичной финансовой поддержке гранта президента для ведущих научных школ НШ-2139.2008.9 и гранта РФФИ N08-01-00024

1. Введение

В данной работе рассматриваются некоторые проблемы построения параллельных программ для решения вычислительных задач с использованием метода частиц (Particles In Cell, PIC). Применение метода PIC получает в настоящее время широкое распространение при решении сложных задач математической физики [1-4]. В 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 PIC, основное внимание уделено способам создания такой программы.

С точки зрения построения параллельных программ метод частиц представляет несомненный интерес, поскольку 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. 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.

При расчете на сетке общая схема распараллеливания определяется основными используемыми структурами данных – статической прямоугольной расчетной сеткой. Распараллеливание выполняется в модели распараллеливания по данным. При использовании этого известного подхода массивы значений расчетных переменных распределяются равномерно между процессорами. Решение проблемы распараллеливания расчетов на регулярных статических сетках достаточно хорошо изучены. В разделе 2 кратко описывается схема распределения статической сетки на линейку процессоров.

При расчете 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. В разделе 3 рассматриваются некоторые проблемы, возникающие при построении параллельной программы расчета частиц, в том числе вопросы планирования и балансировки вычислений. Различные варианты рассмотренных решений были опробованы на многопроцессорных системах, приводятся полученные характеристики времени выполнения программ.

Один из предложенных подходов к планированию расчета частиц опирается на учет при планировании времени, затрачиваемого на расчет. При проведении вычислительных экспериментов обнаружены несколько

неожиданные результаты, которые следует учитывать при составлении плана распределения частиц. Эти результаты рассматриваются в разделе 4.

Применение созданной параллельной программы на многопроцессорных системах с большим числом процессоров позволило решать задачи на большой сетке, число ячеек порядка $10^9 - 10^{10}$, и с числом частиц порядка $10^8 - 10^{11}$. При решении задач с такими объемами данных и вычислений актуальным становится вопрос о создании механизма контрольных точек, обеспечивающих проведение счета в несколько этапов, с возможностью продолжения вычислений. Кратко эти вопросы рассматриваются в разделе 5.

Приведенные результаты получены в ИПМ им. М.В.Келдыша РАН при создании и использовании комплекса параллельных программ для решения задач электродинамики [1]. Комплекс поддерживает масштабируемость по числу процессоров, используемых при счете, а также масштабируемость по параметрам задачи – числу ячеек сетки по каждой из размерностей и числу частиц.

2. Распределение сетки

Кратко остановимся на вопросе распределения расчетной прямоугольной сетки между процессорами. Для хранения сеточных значений функций используются трехмерные массивы.

В модели распараллеливания по данным обычно выбирается некоторое измерение исходной матрицы, и проводится разрезание его на заданное число процессоров. Пусть N_Z – число элементов по выбранному измерению и $\boldsymbol{P}$ – число процессоров, на которое проводится разрезание. Тогда в каждом из $\boldsymbol{P}$ процессоров после разрезания будет расположено $ipn = \frac{N_z - 1}{P} + 1$ элементов исходной матрицы (массива) по выбранному направлению. Распределение можно проводить одновременно по нескольким измерениям.

При распределении на матрицу процессоров объем передаваемой информации меньше, чем при распределении на линейку процессоров. При этом конечно, количество операторов обмена информацией увеличивается. С точки зрения объема передаваемой информацией между процессорами, для трехмерной задачи лучшим является распределение одновременно по трем направлениям. Однако это существенно усложняет процесс написания параллельной программы.

В разработанной параллельной программе мы использовали распределение сетки только по одному направлению (по третьему измерению, ось z) на линейку процессоров. Такое распределение обусловлено схемой, применяемой при распределении частиц между процессорами[1] и необходимостью согласовать распределение сетки и

[1] Подробно эта схема рассматривается в разделе 3.1 “Распределение частиц”.

распределение частиц.

Таким образом, трехмерные массивы для хранения расчетных переменных, определенных в ячейках сетки, в программе распределяются по третьему измерению. Выбор именно третьего измерения связан с размещением в памяти многомерных массивов, принятым в языке Фортран - “по столбцам”. В ряде случаев такое разбиение сетки позволяет избегать переписывания данных в буфер при проведении операций обмена между процессорами, а также при переписывании значений из буфера приема.

Замечание. Распределение на линейку процессоров, вообще говоря, вводит ограничение на число процессоров, которое можно использовать для решения задачи. В этом случае число процессоров ограничено величиной N_Z – числом элементов по выбранному для разрезания измерению сетки.

Таким образом, если ячейка сетки имеет исходные координаты (i, j, k), то при распределении на P процессоров (по направлению k) данная ячейка попадает в процессор с номером ipr, если

$$(ipr - 1) * ipn + 1 \le k \le ipr * ipn ,$$

где ipn – число ячеек, размещаемых в одном процессоре. Фактически, исходный диапазон $[1, N_Z]$ разбивается на P непересекающихся диапазонов

$$\left[s_n^r, s_k^r\right], r = \overline{1, P}, \quad \text{где } s_n^r = (r-1) * ipn + 1, \quad s_k^r = r * ipn \qquad (1)$$

3. Расчет частиц

Второй важной особенностью рассматриваемых задач является расчет методом частиц. Обычно расчет методом частиц занимает гораздо большее время, чем расчет на сетке. В ряде задач расчет методом частиц может занимать до 90% общего времени решения задачи.

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 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. В рассматриваемой реализации выбрана следующая схема. При первоначальном размещении длина вектора задается равной 1. Изменение числа частиц в процессе расчета может произойти в 2-х случаях. Во-первых, возможно порождение новых частиц или уничтожение, а, во-вторых, при планировании перераспределения частиц число частиц в каждом процессоре может измениться. В этих случаях используется простейшая процедура переопределения длины вектора и перераспределения памяти. Следует отметить, что время, затрачиваемое на выполнение такой процедуры, является незначительным. Некоторые характеристики времени выполнения такой процедуры приведены в п.7.

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) распределить частицы по процессорам вычислительной системы, на которой решается задача;

2) обеспечить расчет частиц всеми необходимыми исходными данными, которые используются при расчете частиц;

3) разместить вычисленные при расчете частиц значения в требуемых ячейках.

3.1. Распределение частиц

С точки зрения распараллеливания, задача расчета частиц является очень привлекательной. Каждая частица размещена в ячейке сетки задачи. Расчет каждой частицы проводится по значениям переменных в ячейках сетки (которые задаются выбранным шаблоном) и не зависит (по входным данным) от расчета других частиц.

Однако есть три существенных момента, которые необходимо учитывать:

1) число частиц меняется в процессе решения задачи;

2) расчет каждой частицы незначительно, но отличается (по времени) от расчета других частиц;

3) частицы в процессе счета могут перемещаться из ячейки в ячейку.

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**.

Кратко опишем принципы распределения частиц. Как отмечалось выше, распределение исходной сетки проводится на линейку процессоров по третьему измерению. Алгоритм распределения частиц также заключается в разрезании исходной области по третьему измерению. Пусть N_Z – протяженность по третьему измерению сетки, $\boldsymbol{P}$ – число процессоров, на которое проводится разрезание. С каждым процессором с номером $\boldsymbol{r}$ связывается диапазон $\left[z_n^r, z_k^r\right], r = \overline{1,P}$. При этом допускается $\boldsymbol{z_k^{r-1} = z_n^r, r = \overline{2,P}}$.

Сначала рассмотрим вопрос определения данных интервалов на примере равномерного распределения частиц по процессорам. Зная общее число частиц, участвующих в расчете и число процессоров, на которых проводится расчет, определим среднее число частиц на процессор. Обозначим его $\boldsymbol{Npart}$. Далее введем вектор, $\boldsymbol{Vpart(1:Nz).}$ Рассмотрим все ячейки сетки, у которых координата по третьему измерению равна $\boldsymbol{r}$. Просуммируем количество частиц, размещенных в этих ячейках, и сумму поместим в $\boldsymbol{r}$-ый компонент введенного вектора. Далее переходим к определению границ диапазонов $\left[z_n^r, z_k^r\right], r = \overline{1,P}$. В первый процессор направляются частицы из ячеек, для которых справедливо равенство $\sum_{j=1}^{q} \boldsymbol{Vpart(j) = Npart}$. Таким образом, из этого соотношения определяется верхняя граница $\boldsymbol{q}$ диапазона для первого процессора ($\boldsymbol{q}$ – верхний предел суммирования в приведенном равенстве).

Замечание. В ячейке сетки с номером $\boldsymbol{q}$ может оказаться частиц больше, чем надо, чтобы получить $\boldsymbol{Npart}$. В этом случае часть частиц будет

направлено в первый процессор, а оставшиеся будут распределяться в другие процессоры.

Далее переходим к определению границ диапазона для второго процессора. В качестве нижней границы берется значение q (если там остались еще частицы) или $q+1$, если из ячейки q выбраны все частицы. Просматриваем аналогично последовательно ячейки до тех пор, пока опять не наберем $Npart$ частиц. Данная процедура продолжается, пока не завершится распределение всех частиц по P процессорам.

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

Замечание. В случае распределения частиц “по месту” искомый диапазон для каждого процессора совпадает с диапазоном (1), задающим распределение сетки задачи по процессорам.

На основе полученных диапазонов проводится построение карт отправки и приема частиц. Карта отправки состоит из записей $\langle N_r, P_r \rangle$, 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.

Проиллюстрируем затраты времени, необходимые для обеспечения введенных распределений частиц, на примере расчета модельной задачи.

Обозначим время[2] решения задачи расчета частиц на процессоре с номером i через t_i. Среднее время решения на P процессорах равно $T_{av} = \left(\sum_{i=1}^{p} t_i\right) / P$. Величина T_{av} иллюстрирует "идеальное время"[3] решения задачи. Введем величину $T_{max} = max_{i=1,p}\, t_i$, показывающую реальное время, затрачиваемое на решение задачи расчета частиц и $T_{min} = min_{i=1,p}\, t_i$. Поскольку все вычисления в задаче взаимосвязаны, очевидно, что на общее время счета задачи влияет только величина T_{max}.

На приведенном ниже рисунке 1 представлены три графика, соответствующие распределению частиц ”по времени”. Каждый график показывает изменение времени расчета частиц на 20 итерационных шагах. Первый график иллюстрирует изменение минимального времени расчета

---

[2] Измеряется только время расчета – никакие обмены информацией в рассмотрение не берутся.

[3] Конечно, предполагается, что производительность всех процессоров одинакова.

T_{min}. На втором графике показано “реальное”, затраченное на расчет частиц, соответствующее величине T_{max}. Третий график соответствует “идеальному времени” расчета – T_{av}.

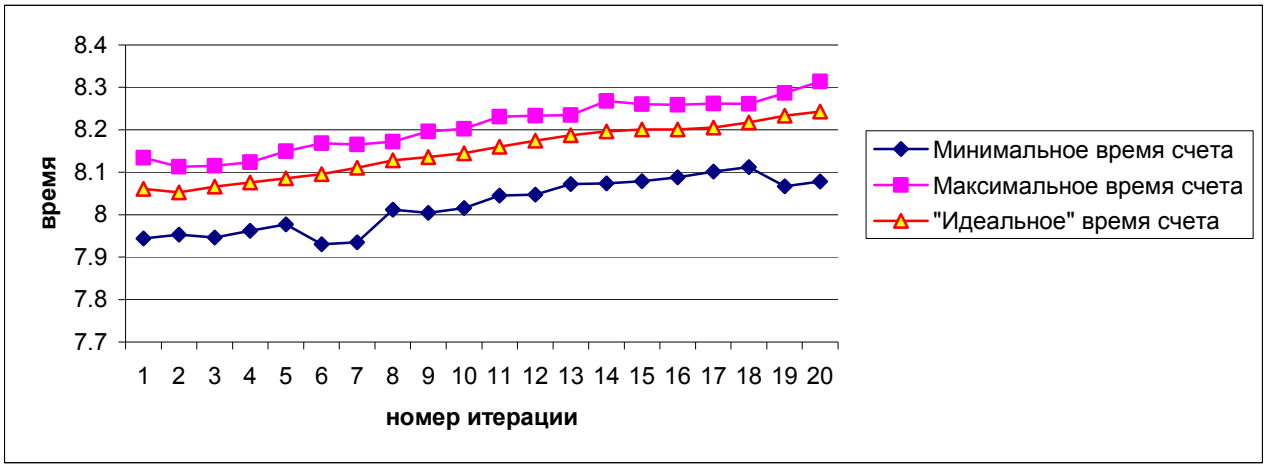

Рисунок 1. Распределение частиц по времени

Определим эффективность планирования расчета частиц $E_{plan} = \frac{T_{av}}{T_{max}} \times 100\%$. При счете 700 итерационных шагов (распределение по времени) были получены следующие результаты T_{av} = 3471.44, и T_{max} = 3503.79. В результате получаем E_{plan} = 99.1%. Конечно, представляет интерес не только как спланирован расчет частиц, но и как влияют накладные расходы на общее время решения. К накладным расходам отнесем планирование распределения частиц, само распределение частиц, а также перемещение требуемых для расчета частиц сеточных значений. Обозначим полное время расчета частиц с учетом накладных расходов – T_{full}. Определим суммарную эффективность $E_{sum} = \frac{T_{av}}{T_{full}} \times 100\%$. В результате суммарная эффективность в данном случае равна E_{sum} = 84.1%.

На рисунке 2 представлены три графика, соответствующие случаю, когда в каждый процессор помещается одинаковое число частиц. Каждый график показывает изменение времени расчета частиц на 20 итерационных шагах. Первый график иллюстрирует изменение минимального времени расчета T_{min}. На втором графике показано “реальное”, затраченное на расчет частиц, соответствующее величине T_{max}. Третий график соответствует “идеальному времени” расчета – T_{av}.

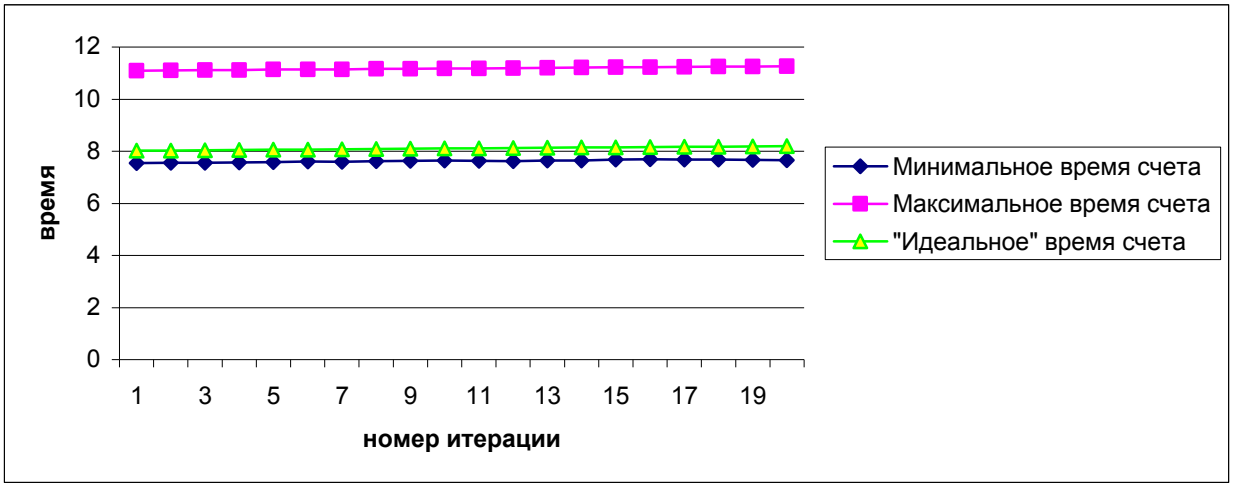

Рисунок 2. Равномерное распределение частиц

При равномерном распределении частиц по процессорам ожидается и равномерное время, затрачиваемое на расчет частиц. Однако приведенные на рисунке 2 графики показывают, что это не так. На рисунке 3 представлена функция, показывающая время расчета частиц на фиксированном шаге на каждом из процессоров. Несмотря на то, что каждый процессор обрабатывает одинаковое число частиц, времена расчета различны. Минимальное значение – 7.21 сек. достигается на процессоре с номером 8, а максимальное – 9.98 сек. на процессоре с номером 82. Эта разница определяется наличием условных вычислений при расчете. Условия зависят от исходных данных задачи (например, от геометрии расчетных тел) и различны в разных процессорах. Фактически оказывается, что эффективность равномерного подхода к распределению частиц зависит от особенностей конкретно решаемой задачи.

Рисунок 3. Время расчета одного шага при равномерном распределении.

На рисунке 4 представлены два графика для распределения частиц по “месту”. На первом графике показано “реальное”, затраченное на расчет частиц, соответствующее величине T_{max}. Второй график соответствует “идеальному времени” расчета – T_{av}.

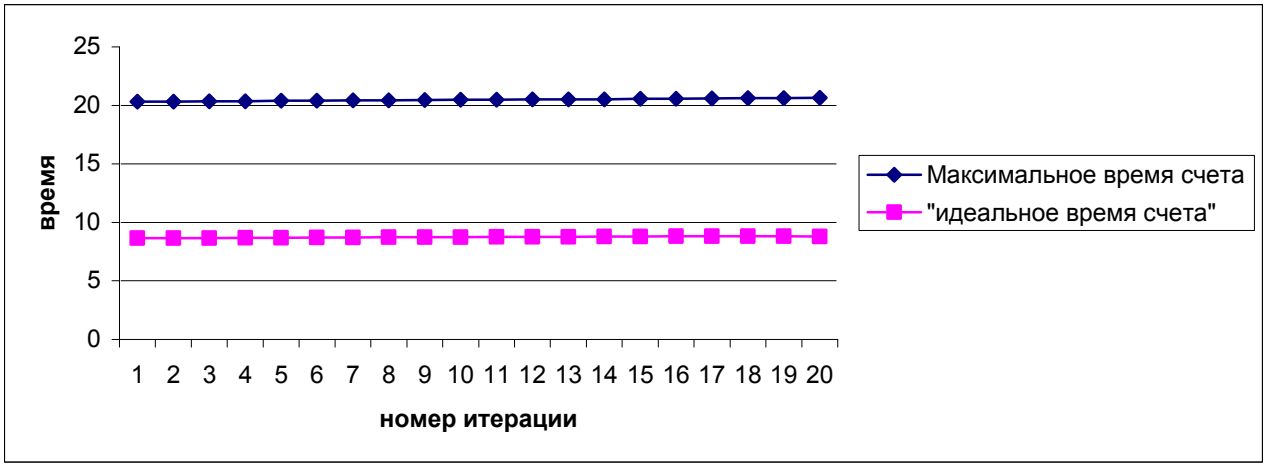

Рисунок 4. Распределение частиц по “месту”

При распределении частиц по месту, проблемы, связанные с перемещением частиц между процессорами, существенно упрощаются. Необходимо только перемещать между процессорами частицы, которые в процессе расчета переместились в другие ячейки и эти ячейки распределены в другие процессоры. Однако величина эффективности низкая, $E_{plan} = 48.2\%$, и это обусловлено тем, что часть процессоров вообще не принимает участие в расчете частиц. С другой стороны, расходы, связанные с подготовкой требуемых сеточных значений, также минимальны, что и определяет незначительное снижение суммарной эффективности расчета, $E_{sum} = 44.7\%$.

В таблице 1 приведены эффективность планирования частиц и суммарная эффективность расчета. Результаты получены на суперкомпьютере СКИФ МГУ ”Чебышев”, установленном в НИВЦ МГУ[4]. Проводился расчет 700 шагов модельной задачи на 82 процессорах.

способ планирования	E_{plan}	E_{sum}
по месту	48.2%	44.7%
равномерное	75.0%	64.7%
по времени	99.1%	84.1%

Таблица 1. Эффективность планирования и расчета частиц

3.2. Требуемые данные для расчета частиц

Для проведения расчета каждой частицы требуются следующие исходные значения:

1) значения расчетных переменных частицы с предыдущего временного шага моделирования (*предыдущие* ***значения частиц***). С каждой частицей в рассматриваемом случае связано 8 значений переменных;

[4]http://parallel.ru/

2) значения из некоторого числа соседних ячеек сетки (***сеточные значения***); соседние ячейки обычно задаются шаблоном, применяемым при решении задачи. В рассматриваемом случае для частицы, находящейся в ячейке с координатами (i, j, k), требуются сеточные значения из ячеек с координатами $(i \pm 1, j \pm 1, k \pm 1)$, из каждой ячейки требуется 7 значений.

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

Поскольку частицы имеют собственное распределение по процессорам, отличающееся, в общем случае, от распределения ячеек сетки, то для определения *сеточных значений*, которые необходимо подкачать, используются следующие правила. Пусть для процессора с номером $r, r = \overline{1, P}$ диапазон расчета частиц имеет вид $\left[z_n^r, z_k^r\right]$, см. п.3.1. Тогда, в рассматриваемом случае, надо провести подкачку требуемых *сеточных значений* из диапазона $\left[z_n^r - 2, z_k^r + 2\right]$. Смещение 2 определяется, во-первых, применяемым при решении задачи шаблоном (для частицы, находящейся в ячейке с координатами (i, j, k) **требуются** сеточные значения из ячеек с координатами $(i \pm 1, j \pm 1, k \pm 1)$), а, во-вторых, тем, что частицы в процессе расчета могут перемещаться в соседние ячейки. Заметим, что даже при распределении частиц “по месту” необходимо проводить подкачку требуемых *сеточных значений*.

3.3. Расчет значений частицы

Результатом расчета каждой частицы являются следующие значения:

1) значения расчетных переменных частицы на текущем временном шага моделирования (*текущие* ***значения частицы***). С каждой частицей в рассматриваемом случае связано 8 значений переменных;

2) значения, которые являются частью интегральных функций в соседних ячейках, на которые оказывает влияние данная частица (***сеточные значения***, 4 значения переменных для каждой соседней ячейки, на которую оказывает влияние частица).

Для хранения *текущих значений частицы* используются соответствующие вектора, обсуждавшиеся в начале п. 3.

Для хранения вычисленных *сеточных значений* возможны два способа. Первый способ состоит в том, что для трехмерных *сеточных значений* определяются трехмерные массивы, при этом протяженность по третьему измерению для процессора с номером r, $r = \overline{1, P}$, имеет следующий вид - $\left[z_n^r - 2, z_k^r + 2\right]$. Результаты расчета сеточных значений будут временно размещаться в этих массивах. После завершения расчета частиц необходимо организовать пересылку этих значений в те процессоры, в которых размещены соответствующие ячейки сетки, так как в общем случае распределение ячеек сетки не совпадает с распределением частиц. Кроме

того, необходимо учитывать тот факт, что в процессе расчета не во все ячейки могут попадать вычисленные результаты. Поэтому, отправляя в другие процессоры все значения из введенных массивов, мы можем отправлять избыточную информацию.

При втором способе 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. В разработанной программе реализован второй способ.

3.4. Возврат результатов расчета частиц

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.

4. Экстремальная ситуация

При проведении расчетов мы столкнулись с ситуацией, которая демонстрирует необходимость достаточно аккуратного и тщательного учета различных факторов, влияющих на эффективность и “жизнеспособность” параллельной программы, однако не всегда учитываемых (в случае, если это возможно) в момент разработки программы. Одним из наиболее тонких и сложных таких факторов является, конечно, время (вычислений, обменов, записи данных на диск, расчетов на различных параллельных системах и т.п.).

Реализованное в разработанной нами параллельной программе распределение частиц “по времени” существенно зависит от времени, которое затрачено на предыдущем шаге моделирования; оно основано на двух эвристиках:

- время вычислений на двух соседних итерационных шагах меняется плавно;
- характеристики производительности аппаратуры стабильны.

Оказалось, что нарушение этих эвристик приводит к неприятным последствиям.

Наблюдалась следующая картина расчета. Время расчета до некоторого момента плавно увеличивается по времени. Увеличение времени

объясняется плавным увеличением числа частиц в моделируемых физических процессах. Однако в некоторый момент максимальное время расчета частиц резко увеличивается, то есть нарушаются эвристики распределения частиц по процессорам (какая именно и по какой причине – моделируется сложный физический процесс, или нестабильно работает аппаратура вычислительной системы, сейчас не важно).

Иллюстрацией данной ситуации служит приведенный ниже рисунок 5. На рисунке представлены три графика. Первый график показывает минимальное время расчета на каждом шаге моделирования. На втором графике показано максимальное время расчета на каждом шаге. Третий график представляет ”идеальное” время расчета на шаге. Как видно из второго графика, на 5-м шаге происходит резкое увеличение времени счета.

В результате возникает следующая проблема. Резкое увеличение в каком-то процессоре времени счета ведет к тому, что при планировании распределения частиц на следующий шаг расчета “по времени” большинство частиц из данного процессора надо перемещать в другие процессоры. Для перемещения частиц между процессорами используется динамический буфер. Так как число частиц в процессоре может быть достаточно большим, порядка $10^6 - 10^7$, и, учитывая, что перемещение одной частицы требует перемещения 8 значений с двойной точностью (*текущие значения частицы*), размеры динамического буфера становятся большими. Поскольку оперативная память является критическим ресурсом в расчетах данного типа, то выделение памяти под большой буфер может приводить к ошибочной ситуации – нехватке памяти.

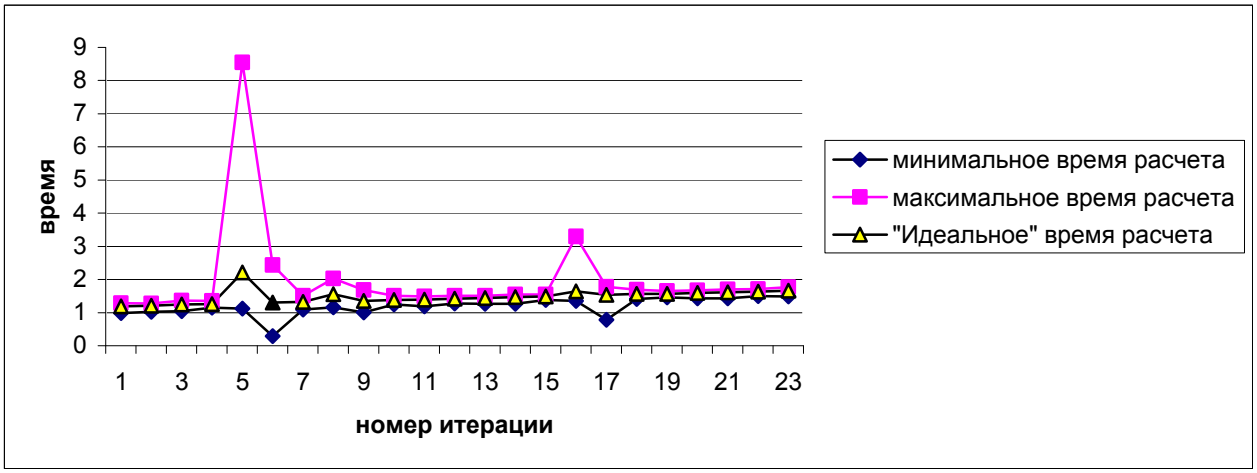

Рисунок 5. “Скачок” времени при расчете частиц

Решить эту непредусмотренную проблему можно например, сглаживая “пики” по времени, то есть, фактически за счет уточнения эвристик.

Заметим, что если памяти достаточно для динамического буфера, то алгоритм позволяет “сгладить” появившейся скачок за 5 шагов. Второй скачок, произошедший на 16 шаге, менее сильный, чем скачок на 5 шаге, и удаление его последствий происходит еще быстрее – за 2 шага.

5. Контрольные точки

При проведении расчета в разработанной параллельной программе предусмотрена возможность установки контрольных точек, с которых возможно продолжение расчета, например, в случае сбоев аппаратуры, или завершения лимита времени счета.

Контрольные точки – это моменты времени, в которые во внешние файлы записываются значения всех расчетных переменных, необходимых для продолжения расчета с этого момента времени. Запись значений для контрольной точки осуществляется в бесформатном представлении для экономии памяти. Контрольная точка определяется набором внешних ***файлов контрольных точек*** (каждый процессор создает свой файл), в которых запоминаются значения всех переменных, необходимых для продолжения расчета.

При решении больших задач, контрольные точки, конечно, жизненно необходимы. При этом, объем информации, который необходимо сохранять, весьма значителен. Более того, для повышения надежности счета следует предусмотреть возможность хранения двух контрольных точек (текущей и предыдущей). Это объясняется, например, возможностью потери контрольной в случае, когда программа завершается из-за истечения заказанного времени, а в это время проводится запись контрольной точки, либо если произошел аппаратный сбой во время записи контрольной точки.

В том случае, когда расчет продолжается с контрольной точки, каждый процессор начинает свои действия со считывания файла контрольной точки. Созданная программа поддерживает продолжение расчета на числе процессоров, отличном от числа процессоров, на котором проводилась предыдущая часть расчета.

6. Метрики эффективности счета

В данном разделе рассмотрим вопрос об эффективности параллельной программы.

Традиционно для оценки эффективности распараллеливания используется следующая величина $E(p) = \frac{T_1}{T_p * p} * 100\%.$ Здесь T_1 – время затраченное для решения исходной задачи на одном процессоре, а T_p – время затраченное для решения исходной задачи на P процессорах. Оценка эффективности вводится с целью определить, как влияют затраты, связанные с поддержкой распараллеливания, на общее время исполнения параллельной программы. При использовании большого числа процессоров время, затраченное только на счет задачи, начинает сокращаться. Это обусловлено двумя причинами. Во-первых, каждый процессор (при увеличении общего числа процессоров) проводит обычно меньше вычислений, так как обычно сокращается объем рассчитываемых данных. Во-вторых, сокращение

объемов данных позволяет использовать возможности кэш-памяти процессорных элементов. На последнее никак не влияет качество построенной параллельной программы.

Более того, использование такой метрики часто бывает невозможно при решении больших (по объему данных) задач, так как невозможно провести расчет таких задач на одном процессоре.

Для того, чтобы избежать отмеченные проблемы, можно использовать другую метрику для оценки эффективности. Пусть время счета задачи на каждом ***i***-ом процессоре состоит из времени, затраченного на передачу данных между процессорами T_{com}^{i} и времени, затрачиваемого на вычисление T_{run}^{i}.

Замечание. Величина T_{com}^{i} состоит не только из времени, затраченного непосредственно на передачу сообщений. Эта величина также содержит время, которое затрачивается на подготовку данных. Например, формирование буфера отправки и разбор буфера приема. Кроме того, в эту величину входит время построения карт отправки и приема и.т.п.

Для конфигурации из p процессоров определим следующие величины:

$T_{com}(p) = \sum_{i=1}^{p} T_{com}^{i}$ (суммарное время, по всем p процессорам, затраченное только на передачу информации);

$T_{run}(p) = \sum_{i=1}^{p} T_{run}^{i}$ (суммарное время, по всем p процессорам, затраченное только на вычисления).

Эффективность рассчитывается по следующей формуле

$$E(p) = \frac{T_{run}(p)}{(T_{run}(p) + T_{com}(p))} \times 100\% = \left(1 - \frac{T_{com}(p)}{(T_{run}(p) + T_{com}(p))}\right) \times 100\% \qquad (2)$$

Заметим, что величина $E(p)$ всегда меньше 100%; она показывает, как сильно влияют на общее время исполнения программы накладные расходы, связанные с передачей данных между процессорами. Эту метрику будем использовать ниже.

7. Характеристики времени счета задач

Ниже приводятся временные характеристики фрагмента расчета, который проводился на суперкомпьютере СКИФ МГУ ”Чебышев”, установленном в НИВЦ МГУ. Некоторые данные получены на ЭВМ МВС 100K установленном в Межведомственном суперкомпьютерном центре[5].

Исходная сетка решаемой задачи имела размерность 273×273×655. Проводился расчет 700 итерационных шагов. Контрольная точка содержит 42.33Gb. Запись такой контрольной точки на ЭВМ “Скиф”заняла 32 сек, а на

[5] http://www.jscc.ru/scomputers.shtml

ЭВМ МВС 100К – 2397сек. Число частиц, участвующих в моделировании к 700-му шагу равно 657 647 724.

Расчет выполнялся для 3-х вариантов использования процессоров - 82, 164 и 328 (число процессоров указано в первой строке приводимых ниже таблиц). Расчет состоит из двух частей – расчет полей на сетке (GRID) и расчет методом частиц (PARTICLES), соответствующие столбцы в таблицах ниже – GRID и PARTICLES. В заключительном столбце SUMMARY представлены суммарные значения.

Во второй строке каждой из таблиц, приведенных ниже, указано осредненное время, затраченное как на вычисления, так и на передачу информации, которое вычисляется по формуле $\frac{T_{run}(p)+T_{com}(p)}{p}$. В третьей строке каждой из таблиц указывается осредненное время, затраченное только на передачу информации. Эта величина вычисляется по формуле $\frac{T_{com}(p)}{p}$. В четвертой строке таблиц представлены эффективности соответствующих частей расчета, рассчитанные по формуле (2).

В таблицах 2-5 представлены результаты, полученные на ЭВМ СКИФ МГУ ”Чебышев” и на ЭВМ МВС 100К для различных способов распределения частиц, описанных выше (“по месту”, равномерное, “по времени”).

	GRID			PARTICLES			SUMMARY		
Проц-ры	**82**	**164**	**328**	**82**	**164**	**328**	**82**	**164**	**328**
$T^{run}+T^{com}$	348	177	98	9070	4222	2088	9418	4399	2186
T^{com}	25	16	17	5014	2380	1211	5040	2397	1229
E	92.7%	90.7%	81.9%	44.7%	43.6%	42.0%	46.5%	45.5%	43.8%

Таблица 2. Результаты расчета на ЭВМ СКИФ МГУ ”Чебышев” (распределение ”по месту”)

	GRID			PARTICLES			SUMMARY		
Проц-ры	**82**	**164**	**328**	**82**	**164**	**328**	**82**	**164**	**328**
$T^{run}+T^{com}$	363	182	130	6009	2598	1634	6372	2780	1764
T^{com}	45	22	42	2074	806	640	2110	828	682
E	90.2%	87.9%	67.8%	65.5%	69.0%	60.8%	66.9%	70.2%	61.3%

Таблица 3. Результаты расчета на ЭВМ СКИФ МГУ ”Чебышев” (распределение равномерное, число частиц в каждом процессоре одинаково)

	GRID			PARTICLES			SUMMARY		
Проц-ры	**82**	**164**	**328**	**82**	**164**	**328**	**82**	**164**	**328**
$T^{run} + T^{com}$	350	184	107	4208	2131	1346	4558	2316	1453
T^{com}	27	24	25	668	437	479	696	461	504
E	92.1%	87.1%	74.3%	84.1%	79.5%	66.5%	84.7%	80.1%	65.3%

Таблица 4. Результаты расчета на ЭВМ СКИФ МГУ ”Чебышев” (распределение ”по времени”)

	GRID			PARTICLES			SUMMARY		
Проц-ры	**82**	**164**	**328**	**82**	**164**	**328**	**82**	**164**	**328**
$T^{run} + T^{com}$	1348	441	213	10309	4117	2967	11658	4558	3180
T^{com}	913	225	76	6230	2151	1237	7144	2377	1314
E	32.3%	48.8%	64%	39.6%	47.7%	58.3%	38.7%	47.8%	58.7%

Таблица 5. Результаты расчета на ЭВМ МВС 100K (распределение ”по времени”)

Наилучшая общая эффективность достигается при использовании метода распределения ”по времени” (Таблица 4, расчет на ЭВМ СКИФ МГУ ”Чебышев”). Это не является неожиданным, так как метод распределения ”по времени” представляется наиболее “тонким”, учитывающим затраты времени наиболее аккуратно. Несколько неожиданными выглядят результаты, приведенные в Таблице 5, когда при расчете на другой вычислительной системе эффективность метода распределения ”по времени” проигрывает другим методам планирования, оставаясь всего лишь чуть лучше метода распределения “по месту”.

На наш взгляд, это может определяться “скачками” времени при расчете частиц (данная ситуация обсуждалась выше в п. 4). На рисунке 6 приведен график максимального времени расчета частиц на различных итерационных шагах.

Рисунок 6. “Скачки” времени при расчете частиц

Отметим, что такие скачки происходят в различных процессорах. По-видимому, замедление вычислений связано с поведением аппаратуры или системного программного окружения, его трудно учесть в алгоритмах балансировки, но это явление сильно влияет на эффективность работы программы. К сожалению, традиционно применяемые метрики оценки эффективности параллельных программ не учитывают возможность различного поведения параллельной программы на разных вычислительных системах.

В процессе расчета программа проводит переопределение длины векторов, используемых для хранения значений частиц (см. п.3). При расчете на 82 процессорах такая процедура выполнялась 10 раз. Итоговая длина вектора составила 10^7 (длина должна быть достаточной для хранения обрабатываемых частиц). Переопределение длины вектора состоит из двух шагов. В начале значения из исходного вектора переписываются в рабочий вектор. Далее оператором ALLOCATE создается вектор с новой длиной и в этот вектор переписываются из рабочего массива необходимые значения. Конечно, время работы такой процедуры определяется количеством переписываемых значений. Среднее время работы процедуры составляет 0.1 сек. на каждые 10^6 значений.

Полный расчет был проведен на ЭВМ СКИФ МГУ ”Чебышев” и состоял из 6000 итерационных шагов. Вычисления проводились на 328 процессорах. 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.

На последнем шаге расчета минимальное (по процессорам) время расчета частиц - 4.04 сек., максимальное (по процессорам) время – 4.30 сек. Общее “идеальное” время расчета частиц составило 20068 сек., а реальное – 20945. Эффективность планирования вычисления частиц – 95.8%. Полное время решения задачи – 30333 сек.

	GRID	PARTICLES	SUMMARY
$T^{run} + T^{com}$	957	27776	28733
T^{com}	264	7497	7762
E	72.4%	73.0%	73.0%

Таблица 6. Результаты полного расчета на 328 процессорах ЭВМ СКИФ МГУ ”Чебышев”

Литература

1. А.Н. Андрианов, А.В. Березин, А.С. Воронцов, К.Н. Ефимкин, М.Б. Марков. Моделирование электромагнитных полей радиационного происхождения на многопроцессорных вычислительных системах. “Математическое моделирование”, 2008, т. 20, № 3, с. 98-114.

2. В.А. Вшивков, М.А. Краева, В.Э. Малышкин. Параллельная реализация метода частиц. “Программирование”, 1997, № 2, с. 39-51.

3. С.Е. Киреев. Параллельная реализация метода частиц в ячейках для моделирования задач гравитационной космодинамики. “Автометрия”, СО РАН, 2006, т. 42, № 3, с. 32-39.

4. Э.А. Кукшева, В.Н. Снытников. Решение нестационарных задач гравитационной физики на суперЭВМ. В сб. Труды конференции ”Научный сервис в сети Интернет: решение больших задач”, 2008, с.52-54, (http://agora.guru.ru/abrau2008/pdf/033.pdf).