Векторизация и распараллеливание метода «частица-частица»

П.О. Медакин, Р.Н. Никулин, О.А. Авдеюк, И.Ю. Королева, Е.С. Павлова, И.Г. Лемешкина

Волгоградский государственный технический университет

Аннотация: В данной работе рассматривается векторизация и распараллеливание метода «частица-частица», применяемого для учета взаимодействий между объектами при математическом моделировании физических процессов, на примере учета пространственного заряда при расчете динамики заряженных частиц. Проведено сравнение и оценка временных затрат (в качестве тестовой задачи рассматривался разлет многокомпонентного ионного пучка в течение одной наносекунды с шагом $\Delta t = 10\text{-}12$ с.) с учетом ускорения за счет векторизации и распараллеливания между ядрами процессора. Сделан вывод, что результаты работы наглядно демонстрируют, что векторизация вычислений позволяет существенно ускорить время расчета, причем явная замена скалярных операций на векторные делает возможным получить дополнительное ускорение по сравнению с использованием автоматической оптимизации кода программы.
Ключевые слова: параллельные вычисления, метод "частица-частица", векторизация вычислений, численное моделирование, кулоновские взаимодействия, динамика заряженных частиц, ионный пучок, код программы, уравнение движения, математическая модель.

1. Введение

Метод «частица-частица» является одним из классических методов учета взаимодействий между объектами. Несмотря на то, что он появился довольно давно, он используется для высокоточных расчетов влияния пространственного заряда в задачах численного моделирования динамики заряженных частиц, а также в астрофизике – в задачах, в которых необходим учет дальнодействующих сил.

Реальные физические модели динамики заряженных частиц включают огромное количество взаимодействующих друг с другом элементов, поэтому актуальной остается проблема повышения производительности вычислений.

В соответствии с названием, в методе «частица-частица» взаимодействующие объекты представляются в виде отдельных частиц, при этом учитываются воздействия со стороны каждой частицы на исследуемую.

Метод обычно включает в себя следующие этапы:

вычисление сил;

интегрирование уравнения движения;

изменение значения счетчика времени.

Основным недостатком метода является большая вычислительная сложность, так как необходимо учесть взаимодействия между каждой парой частиц, сложность алгоритма получается квадратичной. Однако, чтобы снизить сложность расчетов и увеличить производительность, зачастую метод модифицируется для каждой отдельной задачи и/или используются программные методы ускорения вычислений [1,2]. Важной и часто используемой модификацией метода частица-частица, особенно для физической электроники, является введение константы расстояния учета взаимодействия между частицами [3,4]. Воздействие частиц, находящихся на расстоянии большем заданного, считается нулевым и не учитывается в расчетах. В физической электронике, как правило, в качестве константы используется радиус экранирования Дебая.

В силу того, что расчет кулоновских взаимодействий и интегрирование уравнений движения для отдельных частиц проходит независимо друг от друга, их можно провести в различных потоках или процессах, что позволяет не только использовать многоядерность современных процессоров, но и использовать векторные операции. В данной работе рассматривается уменьшение временных затрат на учет кулоновских взаимодействий при векторизации метода «частица-частица».

2. Постановка задачи и математическая модель

Одним из методов построения математической модели системы многих частиц, который позволяет экономить ресурсы ЭВМ, является замена реальной системы, системой с меньшим количеством частиц, но имеющей такие же свойства. Таким образом, математическая модель движения

ионного пучка построена на методе «крупных частиц», заключающемся в последовательном применении Лагранжевого и Эйлерового подходов. Начальный объем ансамбля частиц разбивается на некоторое количество не пересекающихся элементарных объемов, частицы каждого из которых заменяются одной крупной частицей с суммарными зарядом и массой частиц, входящих в начальный элементарный объем.

Для качественного соответствия модели реальности, при расчетах сложных физических систем, необходимо учитывать взаимодействия между объектами. В данной модели для этого используется метод «частица-частица», векторизация которого рассматривается в данной работе.

Для сравнения временных затрат будем рассматривать в качестве тестовой задачи разлет многокомпонентного ионного пучка в течение одной наносекунды с шагом $\Delta t = 10^{-12}$ с.

Уравнение движения имеет следующий вид:

$$\frac{\overrightarrow{dp}}{dt} = \vec{F},$$

где F – силы кулоновского взаимодействия между частицами.

3. Численная реализация и результаты

Для численного решения дифференциального уравнения движения ионного пучка используется классический для данного рода задач метод Рунге-Кутты четвертого порядка [5,6]. Генерация псевдослучайных чисел, использующихся для задания первоначальных параметров пучка, осуществляется с помощью алгоритма *MT19937* [7].

Отметим, что однопоточное исполнение задач данного типа приводит к большим временным затратам, поэтому всегда используются способы распараллеливания вычислений на ядра процессора и графического

ускорителя с помощью технологий: *OpenMP*, *MPI*, *CUDA*, а также векторизации.

Векторизация может быть организована различными способами [8]:

ассемблерные вставки;

векторные операции и типы данных в языке программирования (*intrinsics*);

директивы компилятора;

автовекторизация;

векторизованные библиотеки;

Новейшие *CPU* имеют 256- и 512-битные регистры (набор инструкций *AVX*, *AVX*2 и *AVX*-512), в каждый из которых можно разместить, соответственно, 8 или 16 чисел с плавающей запятой или 4 и 8 чисел с двойной точностью. Однако, отметим, что каким бы способом реализации расчет не производился, ускорение будет, конечно, не кратным из-за того, что в векторные регистры данные нужно сначала загрузить, а после расчета выгрузить [9,10].

В силу того, что целью работы является сравнение уменьшения временных затрат на учет кулоновских взаимодействий именно за счет программных методов ускорения вычислений, то главное требование, предъявляемое к используемому аппаратному обеспечению, заключается лишь в возможности векторизации и распараллеливания задачи на ядра процессора. При проведении расчетов данной работы использовался процессор *Intel® Core™ i*5-8250*U CPU* @ 1.60*GHz* × 8, который позволяет производить векторные операции с использованием технологии *AVX*2, то есть с помощью 256 битных регистров.

Для сравнения и оценки временных затрат решения задачи с помощью различных программных методов ускорения вычислений будем рассматривать три типа реализации с использованием технологий: 1)

OpenMP; 2) OpenMP + автовекторизация; 3) *OpenMP* + автовекторизация + векторные операции (*intrinsics*) для блоков, которые не удалось векторизовать автоматически с помощью компилятора.

Наибольшее время при выполнении программы затрачивается при вычислении кулоновских взаимодействий, а также при интегрировании уравнений движения, поэтому в силу того, что эти процессы проходят для отдельных частиц независимо друг от друга, будем производить распараллеливание именно этого блока программы.

В первом типе реализации перед циклом по частицам установим директиву для распараллеливания:

```
omp_set_num_threads(Threads);
#pragma omp parallel for
for (int i = 0; i <= MAX; i++)
{
double FFx=0,FFy=0,FFz=0,VX,VY,VZ;
for (int j = 0; j <=MAX; j++)
{
double dFFx=0,dFFy=0,dFFz=0;
if (i!=j)
{
Kulon(ion[i].X,ion[j].X,ion[i].Y,ion[j].Y,ion[i].Z,ion[j].Z,ion[
i].qi,ion[j].qi,dFFx,dFFy,dFFz);
}
```

Во втором варианте реализации добавим в начале программы директивы для использования автоматической оптимизации:

```
#pragma GCC target("avx2")
#pragma GCC optimize("O3")
```

В третьем варианте реализации заменим скалярные операции, применяемые при вычислении кулоновских взаимодействий, на векторные:

```
            omp_set_num_threads(Threads);
            #pragma omp parallel for
            for (int i = 0; i < MAX; i++)
            {
                 double
FFx=0,FFy=0,FFz=0,dFFx[MAX]={0},dFFy[MAX]={0},dFFz[MAX]={0},VX,V
Y,VZ;
                  double X,Y,Z;
               __m256d x = _mm256_set1_pd(ion[i].X);
               __m256d y = _mm256_set1_pd(ion[i].Y);
               __m256d z = _mm256_set1_pd(ion[i].Z);
               __m256d q = _mm256_set1_pd(ion[i].qi);
             __m256d KKK = _mm256_set1_pd(K);
               q = _mm256_mul_pd(q,KKK);
               for (int j = 0; j <MAX/4; j++)
                  {
                    __m256d x1 = _mm256_loadu_pd(&ion[j].X);
                    __m256d y1 = _mm256_loadu_pd(&ion[j].Y);
                    __m256d z1 = _mm256_loadu_pd(&ion[j].Z);
                    __m256d qq =_mm256_loadu_pd(&ion[j].qi);
                    x = _mm256_sub_pd(x, x1);
                    y = _mm256_sub_pd(y, y1);
                    z = _mm256_sub_pd(z, z1);
                    __m256d x2 = _mm256_mul_pd(x, x);
                    __m256d y2 = _mm256_mul_pd(y, y);
                    __m256d z2 = _mm256_mul_pd(z, z);
                    __m256d r2 = _mm256_add_pd(x2, y2);
                    r2 = _mm256_add_pd(r2, z2);
                    q = _mm256_mul_pd(q, qq);
                    q = _mm256_div_pd(q, r2);
                    r2 = _mm256_sqrt_pd(r2);
```

```
                x = _mm256_mul_pd(x, q);
                y = _mm256_mul_pd(y, q);
                z = _mm256_mul_pd(z, q);
                x = _mm256_div_pd(x, r2);
                y = _mm256_div_pd(y, r2);
                z = _mm256_div_pd(z, r2);
            _mm256_storeu_pd(&dFFx[j], x);
           _mm256_storeu_pd(&dFFy[j], y);
            _mm256_storeu_pd(&dFFz[j], z);
```

На рис. 1 представлены временные затраты в зависимости от количества обсчитываемых крупных частиц при трех вариантах реализации программы с использованием технологий: 1) *OpenMP*; 2) OpenMP + автовекторизация; 3) *OpenMP* + автовекторизация + векторные операции (*intrinsics*) для блоков, которые не удалось векторизовать автоматически с помощью компилятора.

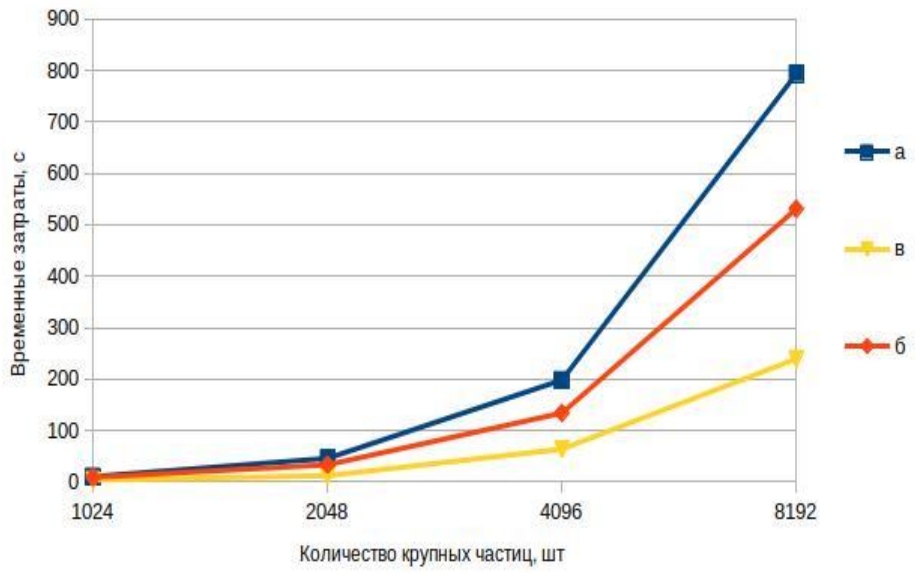

Рис. 1. – Сравнение временных затрат решения задачи о разлете ионного пучка с учетом ускорения за счет использования: а) OpenMP; б) OpenMP + автовекторизация; в) OpenMP + автовекторизация + intrinsics.

Заключение

В ходе данной работы была проведена векторизация и распараллеливание на ядра процессора метода «частица-частица», применяемого для учета взаимодействий между объектами при математическом моделировании физических процессов. Проведено сравнение и оценка временных затрат с учетом ускорения за счет использования программных методов ускорения вычислений. Результаты работы наглядно демонстрируют, что векторизация вычислений позволяет существенно (в данном случае около 3.3 раза) ускорить время расчета, причем явная замена скалярных операций на векторные позволяет получить дополнительное ускорение по сравнению с использованием автоматической оптимизации кода программы.

Литература

1. Рошаль А. С. Моделирование заряженных пучков. Москва : Атомиздат,1979.224с.

2. Белоцерковский О. М. Численное моделирование в механике сплошных сред: 2-е изд., перераб. и доп. Москва: Физматлит, 1994. 448с.

3. Hockney R., Eastwood J. Numerical modeling by the particle method. Computer simulation using particles. McGraw Hill, 1981. 640 p.

4. Григорьев Ю. Н., Вшивков В.А., Федорчук М.П. Численное моделирование методом частицы в ячейках. Новосибирск: Издательство СО РАН, 2004. 360 с.

5. Ковтун Д. Г. Трехмерный релятивистский электронный поток в скрещенных полях // Электромагнитные волны и электронные системы. 2004. Т. 9. № 2. С. 58-65

6. Медакин П.О., Никулин Р.Н., Авдеюк О.А., Поляков И.В., Грецова Н.В. Использование численного моделирования для расчета движения

ионного пучка в лазерном масс - спектрометре ЭМАЛ – 2 // Инженерный вестник Дона, 2020, №7. URL: ivdon.ru/ru/magazine/archive/N7y2020/6552

7. Быков Д.В., Неретин А.Д. Прогнозирование производительности при реализации алгоритмов генерации случайных последовательностей больших размерностей на реконфигурируемых архитектурах с сопроцессорами // Инженерный вестник Дона, 2014, №2. URL: ivdon.ru/ru/magazine/archive/n2y2014/2414.

8. Завьялов Д.В. О векторизации алгоритма Монте-Карло // Математ. физика и компьютер. Моделирование, 2020. Т.23. №1. С. 13-21.

9. Harris D., Harris S. Digital Design and Computer Architecture. N. Y. : Morgan Kaufmann, 2012. 712 p.

10. Intrinsics Guide. URL: software.intel.com

References

1. Roshal’ A. S. Modelirovanie zaryazhennyh puchkov [Charged beam modeling]. Moskva: Atomizdat, 1979. 224 p.

2. Belocerkovskij O. M. CHislennoe modelirovanie v mekhanike sploshnyh sred [Numerical Simulation in Continuum Mechanics]: 2-e izd, pererab. i dop. Moskva: Fizmatlit, 1994. 448 p.

3. Hockney R., Eastwood J. Numerical modeling by the particle method. Computer simulation using particles. McGraw Hill, 1981. 640 p.

4. Grigor'ev YU. N., Vshivkov V.A., Fedorchuk M.P. CHislennoe modelirovanie metodom chasticy v yachejkah [Particle-in-cell numerical simulation]. Novosibirsk: Izdatel'stvo SO RAN, 2004. 360 p.

5. Kovtun D. G. Elektromagnitnye volny i elektronnye sistemy. 2004. T. 9. № 2. pp. 58-65.

6. Medakin P.O., Nikulin R.N., Avdeyuk O.A., Polyakov I.V., Grecova N.V. Inzhenernyj vestnik Dona, 2020, № 7. URL: ivdon.ru/ru/magazine/archive/N7y2020/6552.

7. Bykov D.V., Neretin A.D. Inzhenernyj vestnik Dona, 2014, № 2. URL: ivdon.ru/ru/magazine/archive/n2y2014/2414.

8. Zav'yalov D.V. Matematicheskaya fizika i komp'yuter. Modelirovanie, 2020. T.23. №1. pp. 13-21.

9. Harris D., Harris S. Digital Design and Computer Architecture. N. Y. : Morgan Kaufmann, 2012. 712 p.

10. Intrinsics Guide. URL: software.intel.com