Институт информатики и
математического моделирования Федерального
исследовательского центра
«Кольский научный центр РАН»

Способы ускорения расчёта электромагнитных процессов в энергетике при применение метода конечных разностей во временной области

Бороздина Евгения Дмитриевна - магистрант 2го курса по направлению подготовки Информационные системы и технологии.
e.borozdina@ksc.ru
Научный руководитель Куклин Д.В., к.т.н., научный сотрудник
Федоров А.М., к.т.н., заместитель директора по научной работе

г. Апатиты, Мурманская область
2022 г.

Актуальность исследования

Для расчетов электромагнитных процессов используются алгоритмы, требующие проведения большого количества однотипных арифметических операций.

Проблема

Установлено, что последовательное выполнение таких алгоритмов времязатратно и неэффективно с точки зрения использования вычислительных ресурсов.

Предполагаемое решение

Применение технологий параллельного программирования и распределенных вычислений к алгоритму расчета электромагнитных процессов

Метод конечных разностей во временной области (finite-difference time-domain, FDTD)

Достоинства

- Прост в понимании и реализации;
- Хорошо распараллеливается;
- Усложнение формы объектов не ведет к увеличению занимаемой памяти и времени расчета.

Недостатки

- Необходимо моделировать среду, которая окружает объекты из-за этого объем памяти и время выполнения программы могут быть слишком большими;
- Повышение точности сетки существенно увеличивает время расчета.

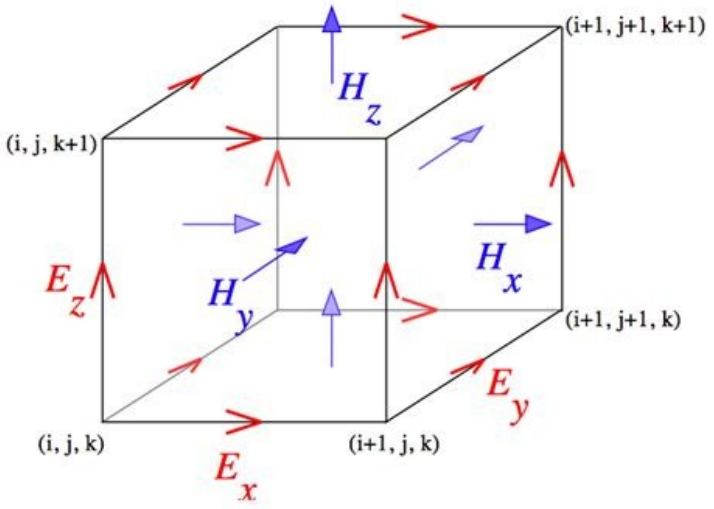


Рисунок 1. Электрическое и магнитное поля в ячейке сетки FDTD.

Реализация базового алгоритма на языке C++. Фрагмент основного цикла.

```
//main loop
for (int cnt = 0; cnt < n; cnt++) {
    for (int i = 0; i < xsize; i++) {
        for (int j = 1; j < ysize; j++) {
            for (int k = 1; k < zsize; k++) {
                ex[i][j][k] = cax[i][j][k] * ex[i][j][k] + cbx[i][j][k] *
                    ( hz[i][j][k] - hz[i][j-1][k] - hy[i][j][k] + hy[i][j][k-1] );
            }
        }
    }

    for (int i = 1; i < xsize; i++) {
        for (int j = 0; j < ysize; j++) {
            for (int k = 1; k < zsize; k++) {
                ey[i][j][k] = cay[i][j][k] * ey[i][j][k] + cby[i][j][k] *
                    ( hx[i][j][k] - hx[i][j][k-1] - hz[i][j][k] + hz[i-1][j][k] );
            }
        }
    }

    for (int i = 1; i < xsize; i++) {
        for (int j = 1; j < ysize; j++) {
            for (int k = 0; k < zsize; k++) {
                ez[i][j][k] = caz[i][j][k] * ez[i][j][k] + cbz[i][j][k] *
                    ( hy[i][j][k] - hy[i-1][j][k] - hx[i][j][k] + hx[i][j-1][k] );
            }
        }
    }
```

Способы оптимизации вычислений

Технологии параллельного программирования

- Параллельные расширения (MMX, SSE и AVX) — векторизация (автовекторизация);
- OpenMP (Open Multi-Processing);
- Многопоточность;
- Технологии GPU (Graphics Processing Unit) программирования (CUDA, OpenCL).

Технология распределенных вычислений

Применение многопоточности (использование класса thread). Фрагмент основного цикла.

```
//main loop
for (int cnt = 0; cnt < n; cnt++) {
    //thread *threads = new thread[threadNum];
    thread *threads = (thread*)malloc(threadNum*sizeof(thread));
    for (int th = 0, bottom = 0, top = slice; th < threadNum; th++) {
        if (th == threadNum - 1) top = obj.xsize;
        threads[th] = thread(threadExyz, obj, bottom, top);
        bottom += slice;
        top += slice;
    }

    for (int th = 0; th < threadNum; th++) threads[th].join();

    //source
    float t0 = 20.0;
    float spread = 6.0;
    obj.ez[point][point][obj.zsize/2] += exp(-0.5*(pow((t0-cnt)/spread,2.0)))/1.0;

    for (int thr = 0, bottom = 0, top = slice; thr < threadNum; thr++) {
        if (thr == threadNum - 1) top = obj.xsize;
        threads[thr] = thread(threadHxyz, obj, bottom, top);
        bottom += slice;
        top += slice;
    }

    for (int thr = 0; thr < threadNum; thr++) threads[thr].join();
    //delete[] threads;
    free(threads); //delete threads;
}
```

Применение технологии OpenMP. Фрагмент основного цикла.

```
//main loop
for (int cnt = 0; cnt < n; cnt++) {
    #pragma omp parallel for num_threads(threads)
    for (int i = 0; i < xsize; i++) {
        for (int j = 1; j < ysize; j++) {
            for (int k = 1; k < zsize; k++) {
                ex[i][j][k] = cax[i][j][k] * ex[i][j][k] + cbx[i][j][k] *
                    ( hz[i][j][k] - hz[i][j-1][k] - hy[i][j][k] + hy[i][j][k-1] );
            }
        }
    }
    #pragma omp parallel for num_threads(threads)
    for (int i = 1; i < xsize; i++) {
        for (int j = 0; j < ysize; j++) {
            for (int k = 1; k < zsize; k++) {
                ey[i][j][k] = cay[i][j][k] * ey[i][j][k] + cby[i][j][k] *
                    ( hx[i][j][k] - hx[i][j][k-1] - hz[i][j][k] + hz[i-1][j][k] );
            }
        }
    }
    #pragma omp parallel for num_threads(threads)
    for (int i = 1; i < xsize; i++) {
        for (int j = 1; j < ysize; j++) {
            for (int k = 0; k < zsize; k++) {
                ez[i][j][k] = caz[i][j][k] * ez[i][j][k] + cbz[i][j][k] *
                    ( hy[i][j][k] - hy[i-1][j][k] - hx[i][j][k] + hx[i][j-1][k] );
            }
        }
    }
```

Сравнение времени выполнения

// Расчетный объем и время расчета одинаково для всех вариантов

Стандартный алгоритм fdtd 379 сек

Многопоточность C++ 271 сек

OpenMP 134 сек

OpenMP позволило добиться почти 3х кратного увеличения скорости расчета.

// Характеристики процессора, но котором производились вычисления

Intel® Core™ i5-6400 CPU @ 2.70GHz × 4

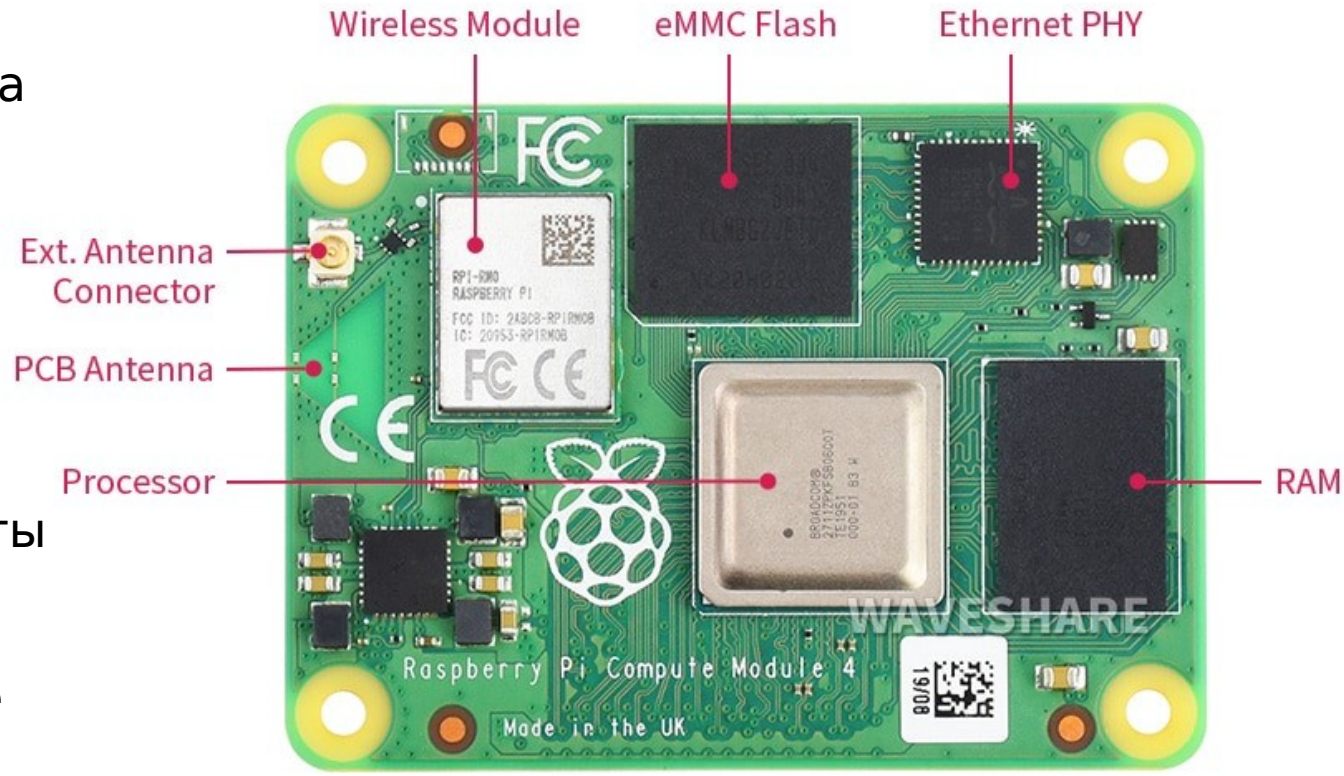
Применение технологии распределенных вычислений

Для расчетов использовался модуль RaspberryPi Compute Module 4.

Расчеты на данном процессоре оказались в 6 раз медленнее по сравнению с расчетами на CPU.

Планировалось использовать несколько таких модулей, подключенных параллельно, но дальнейшие эксперименты не проводились.

Было решено, что данное решение будет экономически невыгодно.



Результаты. Планирование дальнейших экспериментов.

Результаты:

- Реализован базовый алгоритм расчета электромагнитных процессов;
- Применена технология многопоточного программирования;
- Применена технология OpenMP;
- Предпринята попытка рачета на ARM-процессоре.

В ходе экспериментов установлена эффективность применения технологии OpenMP. С помощью ее применения удалось достигнуть ускорения расчета в 3 раза по сравнение с базовым алгоритмом.

Установлена экономическая нецелесообразность использования RaspberryPi CM 4 для расчетов.

Дальнейшие эксперименты:

Планируется применить к алгоритму FDTD технологию GPU-программирования — OpenCl.

Спасибо за внимание

Бороздина Евгения Дмитриевна - магистрант 2го курса по направлению подготовки Информационные системы и технологии.
e.borozdina@ksc.ru