Задание самостоятельной работы студента

В рамках выполнения данной самостоятельной работы Вам будет необходимо написать и отладить программное обеспечение выполняемое на ПК с МП Intel, оценить точность решения задачи, проанализировать ее производительность, модернизировать программное обеспечение по рекомендациям от инструментария oneAPI, спрогнозировать увеличение производительности при использовании гетерогенной системы, определить узкие места реализации ПО, дать оценку составляющих производительности приложения.

Постановка задачи

Необходимо написать программу, выполняющую симуляцию гравитационного взаимодействия n-тел для трехмерного пространства (n-body).

Задана система из n тел, каждое из которых обладает массой m_i, и эти тела, пребывая в вакууме, попарно взаимодействуют друг с другом согласно закону тяготения Ньютона в течение времени t:

$$\overrightarrow{F_i} = G \times m_i \times \sum_{j \neq i} m_j \times \frac{\vec{r}_j - \vec{r}_i}{\left|\vec{r}_j - \vec{r}_i\right|^3}$$

- сила гравитационного взаимодействия, оказываемая на i-е тело со стороны системы. Где $G -$ гравитационная постоянная (6,67430(15)·10^{-11} Н·м²·кг$^{-2}$), m_i, m_j, r_i, v_i — масса тела, применительно к которому действует рассматриваемая сила, массы остальных тел системы, радиус-вектор, описывающий положение тела в пространстве и скорость i-го тела соответственно.

Необходимо написать программу, моделирующую эволюцию динамической системы (изменение положения тел в пространстве), которую можно описать с помощью системы дифференциальных уравнений:

$$\frac{\partial \vec{r}_i}{\partial t} = \vec{v}_i$$

$$\frac{\partial \vec{v}_i}{\partial t} = \vec{a}_i$$

Поскольку нам нужно определить, как изменится положение тел в пространстве с течением времени, и согласно второму закону Ньютона $\vec{a}_i = \vec{F}/m_i$, то получаем систему следующего вида:

$$\frac{\partial \vec{r}_i}{\partial t} = \vec{v}_i$$

$$\frac{\partial \vec{v}_i}{\partial t} = G \times \sum_{j \neq i} m_j \times \frac{\vec{r}_j - \vec{r}_i}{\left|\vec{r}_j - \vec{r}_i\right|^3}$$

В начальный момент времени тела системы должны быть равномерно распределены в пределах единичного куба. Начальные скорости так же для простоты должны быть распределены равномерно в пределах единичного куба, центр которого совпадает с центром системы координат (по каждой из осей Евклидовой системы координат получается в диапазоне [-0.5;0.5]). Начальные ускорения тел равны 0.

Метод решения

Далее необходимо согласно варианту решить поставленную задачу с применением одного из следующих численных методов:

Вариант 1. Метод средней точки или модифицированный метод Эйлера;

Вариант 2. Метод трапеций, также называют методом Хойна 2 порядка (Х2);

Вариант 3. Метод Хойна 3 порядка (Х3);

Вариант 4. Метод Рунге-Кутты 4 порядка (РК4).

К примеру, рассмотрим метод Рунге-Кутты 4 порядка.

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(x_n, y_n)$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(x_n + h, y_n + hk_3),$$

где h – dвеличина шага по x, что в контексте данной задачи соответствует шагу во времени.

В работе можно обойтись использованием четырех базовых функций:

- ***compute_func(…)***:

$$pos_{new} = pos_{curr} + time_{step} \times v_{curr}$$

$$a_{curr} = G \times \sum_{j \neq i} m_j \times \frac{\vec{r}_j - \vec{r}_i}{\left|\vec{r}_j - \vec{r}_i\right|^3}$$

$$v_{new} = v_{curr} + time_{step} \times a_{curr},$$

  в данной функции вычисляются значения системы для одного шага метода (например, для метода Эйлера, эта функция вычисляется один раз за шаг, для РК4 – 4, для каждого k).

- ***update_data(…)***:

$$pos_{curr} = pos_{new}$$

$$v_{curr} = v_{new},$$

  в данной функции обновляются значения переменных нашей системы для всех тел.

- ***advise_time(…)***:

$$time_{curr} += time_{step},$$

в данной функции обновляется системное время (время симуляции), при переходе на следующий шаг. Условием завершения симуляции можно считать момент, когда $time_{curr} = 1$ сек.

– ***get_energy(…)***:

$E_{\text{сумм}} = E_{\text{потенциальная}} + E_{\text{кинетическая}} = F_{\text{гравитационного взаимодействия}} \times h + m \times v^2$,

функция для вычисления полной энергии системы тел для проверки точности симуляции на каждом шаге. Исходя из закона сохранения энергии на каждом шаге нужно смотреть на изменения относительной погрешности при вычислении величины полной энергии системы. Потенциальная энергия рассчитывается как сумма потенциальных энергий каждой частицы в гравитационном поле других частиц

Моделирование взаимодействия тел выполнять в течение 1 секунды. Установление величины шага – на усмотрение студента. Естественно, с большим шагом мы получим большую точность, однако это приведет к увеличению времени выполнения алгоритма. В качестве примера рассмотрим разделение временного интервала на 10 частей, тогда каждый раз время моделируемой системы будет инкрементироваться на 1/10 секунды. H = 0,1 секунды. Кол-во тел, для которых производится симуляция гравитационного взаимодействия ≥ 10000.

Для решения задачи необходимо реализовать следующую систему классов и структур:

1. ***Body –*** структура, описывающая состояние одного тела, содержит поля:

– float pos_x, pos_y, pos_z;

– float vel_x, vel_y, vel_z;

– float acc_x, acc_y, acc_z;

– float mass;

2. ***Simulation*** – класс, описывающий состояние симулируемой системы, содержит поля: particles – массив тел. Методы класса: start – начать симуляцию, init_system – проинициализировать систему начальными значениями. Соответственно, в методе start выполняются описанные выше базовые функции на протяжении n шагов, итераций численного метода.

Используемые константы:

– const double G = 6.67259e-11; // Гравитационная постоянная

– const double softeningSquared = 1e-6; // Используется для случая, когда расстояния между телами слишком малы, чтобы не получить 0 в знаменателе.

На каждом шаге для отладки необходимо выводить следующую информацию:

– номер шага;

– текущее моделируемое время (= номер шага * размер шага);

– суммарная энергия системы тел, в качестве альтернативного варианта, можно рассмотреть закон сохранения импульса системы тел;

– время, затраченное на итерацию.

Выполнение по этапам

Работа выполняется и сдается в 4 этапа, каждый из которых оценивается отдельно:

1. Написать и отладить ПО, реализующее один из методов решения задачи, измерить эффективность его работы с помощью Intel Advisor, изменить точность данного решения.
2. Проанализировать код с помощью Intel Advisor, применить предлагаемые шаги по оптимизации, оценить возможности по распараллеливанию (выделить параллельные секции).
3. Реализовать параллельную версию метода с использованием средств OpenMP.
4. Предоставить отчет об эффективности выполнения параллельной версии алгоритма, проанализировать на наличие проблем в параллельной версии (Intel VTune + Intel Inspector)

Вспомогательные материалы

1. https://en.wikipedia.org/wiki/N-body_problem
2. https://github.com/SandalovKY/n-body-example
3. https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods
4. https://en.wikipedia.org/wiki/Heun%27s_method
5. https://www.cfm.brown.edu/people/dobrush/am33/Mathematica/ch3/modify.html
6. https://en.wikipedia.org/wiki/List_of_Runge%E2%80%93Kutta_methods
7. https://mipt.ru/upload/medialibrary/87d/rk.pdf