Отчёт

На начальном этапе имеется возможность выбрать несколько типов и размеров поля, которые будут собраны в compile time для быстрого их использования в run time при необходимости. Параметры поля, ускорение свободного падения и значения rho можно задавать в файле. Все изменения существующего Simulator.hpp будут внесены в файл BetterSimulator.hpp

Пройдёмся по пунктам из системы оценки и разберём способы их решения .

* Возможность параллельного запуска и выбора количества потоков.  
  Для реализации этого пункта была использована библиотека <omp.h> и функция omp\_set\_num\_threads. Для её подключения необходимо доработать CMakeLists.txt, добавив две строчки:  
  find\_package(OpenMP REQUIRED) – чтобы CMake нашёл библиотеку  
  target\_link\_libraries(HW3 PRIVATE OpenMP::OpenMP\_CXX) – для привязки к моему исполняемому файлу.  
  В нужных местах будут вставлены строчки вида  
  #pragma omp parallel for  
  Для параллельного исполнения (с использованием количества потоков не превосходящего числа, указанного в omp\_set\_num\_threads() ) следующего цикла for.  
  #pragma omp atomic  
  Для выполнения следующей операции атомарно.  
  Для выбора количества потоков была добавлена проверка на наличие в argv строки --threads-cnt=, в которой можно указать желаемое количество потоков.
* Программа корректно работает в несколько потоков.  
  Пункт выполняется, если библиотека <omp.h> работает корректно, и если я корректно использую
* Существенное ускорение без использования потоков.  
  Для достижения этой цели была переработана функция VectorField.get(). Поиск индекса через std::ranges::find() на массиве из четырёх элементов работает значительно медленнее, чем switch-case на возможных значениях элементов этого массива. Данное изменение значительно уменьшило время исполнения программы в 2 раза. Добавление inline перед некоторыми функциями позволяет экономить на вызовах, немного уменьшает время исполнения.
* Наличие большого ускорения за счёт параллелизма на достаточно большом тесте.  
  Многие циклы, используемые в вычислениях симулятора могут выполняться параллельно, так как затрагивают только свою часть массива данных. При использовании 8 потоков такая оптимизация ускоряет ранее упомянутые циклы на 3 или 4 порядка.
* Запуск в потоков на достаточно большом тесте даёт ускорение хотя бы в раз (на какой-то реальной машине).   
  Настолько сильного ускорения мне добиться не получилось, но описанное ранее распараллеливание циклов позволило добиться ускорения на 12%.
* Оформление отчёта.  
  Стараюсь.

Замечание

Использование статических массивов вызывает замедление работы программы (ориентировочно на 10%, причину я так и не нашёл).

Итог

Исходная программа (файл main.cpp с Simulator.hpp, цель сборки HW2) была улучшена с использованием многопоточности и нескольких других оптимизаций (файл second\_main.cpp с BetterSimulator.hpp, цель сборки HW3). Далее предоставлена таблица времени исполнения DEBUG версии всех реализаций на одинаковом тесте (500 тиков симуляции на поле 100×83)

|  |  |  |  |
| --- | --- | --- | --- |
| Реализация | Без оптимизаций | Оптимизированная | 8-поточная оптимизированная |
| Время исполнения | 9.67486 секунд | 3.88982 секунд | 3.3896 секунд |