ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

СИМУЛЯЦИЯ НЕСЖИМАЕМОЙ ЖИДКОСТИ

Отчет по проделанной работе

Исполнитель:

студент группы БПИ 231

В. Попов

«18» декабря 2024 год

АННОТАЦИЯ

Данный отчет является документом, отображающим результаты проделанной работы по оптимизации алгоритма.

В отчете рассматриваются все внесенные изменения в первоначальный код. Также представлены результаты измерения времени до и после внесенных изменений.

СОДЕРЖАНИЕ

1. **Внесенные изменения**

1.1 История изменений

Номер коммита	**Название коммита**	**Описание внесенных изменеий**
b3f4f09	Source	Добавление в проект исходной кодовой базы
f5d88e2	First commit	Создание функции void initialize_field() для построения поля произвольного размера N x M. Перенос функционала «Recalculate p with kinetic energy» из main в функцию void recalculate_kinetic_energy(size_t start_x, size_t end_x) для распараллеливания. Также были удалены неиспользуемые переменные eps, inf и total_delta. Последняя же вычислялась, но нигде не использовалась.
69def50	Optimization: Apply forces from p	Распараллеливание секции «Apply forces from p» в main при помощи OpenMP.
eabc498	Added reading the number of threads from the keyboard	Добавление возможности задавать количество потоков с клавиатуры
d313bc3	Optimization: Apply external forces	Распараллеливание секции «Apply external forces» в main при помощи OpenMP.

1.2 Подробное описание работы с потоками:

В **f5d88e2** используется работа с нитями (thread). Каждый поток обрабатывает свой диапазон строк матрицы. Каждый поток работает в

диапазоне строк [start_x ; end_x). Массивы p, velocity и velocity_flow будут обновляться разными потоками, но без пересечения обрабатываемых областей, т.к. каждое обновления локально для определенных ячеек.

В **69def50, d313bc3** директива #pragma omp parallel for автоматически запускает требуемое количество потоков, оптимально распределяя нагрузку. #pragma omp atomic гарантирует, что уменьшение значения p[x][y] выполняется безопасно, даже если несколько потоков одновременно обращаются к одной ячейке, т.е. предотвращает эффект гонки.

2. Временные замеры

Все замеры проводятся с помощью директивы <ctime> и функции clock(). Замер проводится путем вычисления разности значений функции в начале и в конце алгоритма. Результат измеряется в количестве тактов процессора (ms). Тесты проводятся на системе с процессором 12th Gen Intel(R) Core(TM) i7-12700H. 14 ядер.

1.1 Маленькие тесты

Проведем замеры на исходном тесте для исходного кода, где N = 36, M = 84. Зададим T = 50. Количество потоков = 4. Результаты:

source.cpp

Номер теста	**Результат**
№1	**5114551ms**
№2	**5121067ms**
№3	**5140673ms**
№4	**5123992ms**

Среднее значение: 5125070,75ms

Проведем замеры на коде с распараллеливанием и теми же параметрами:

fluid.cpp

Номер теста	**Результат**
№1	**4298055ms**
№2	**4310777ms**
№3	**4294289ms**
№4	**4314763ms**

Среднее значение: 4304471ms

Таким образом, прирост производительности составляет: **19%**

Проведем тест с теми же параметрами, но с большим количеством тиков (T = 10000) :

1. Для **source.cpp** время выполнения составляет **1701649701ms**
2. Для **field.cpp** время выполнения составляет **989344014ms**

В данном случае, прирост производительности составляет: **72 %**

1.2 Большой тест

Проведем тест при N=200, M=200, T=100, Количество потоков = 8. Результат выполнения:

1. Для **source.cpp** время выполнения составляет **156698150ms**
2. Для **field.cpp** время выполнения составляет **118936925ms**

Тогда прирост в производительности составляет: **31,7%**

3. ИТОГ:

Как можно видеть, оптимизация при помощи распараллеливания дала значительный результат. Можно заметить, что с увеличением тиков программа работает быстрей по сравнению с исходной. Также с полем большего размера программа работает быстрей, чем с полем меньшего размера. Так как с увеличением количества тиков и размера поля увеличивается производительность, можно констатировать, что при больших параметрах, можно увидеть многократное преимущество в производительности.