Лабораторное задание №1

Описание задачи:

Необходимо рассчитать определенный интеграл $\int_0^1 \frac{4}{\sqrt{4-x^2}} dx$

Аналитическое решение:

$$\int_0^1 \frac{4}{\sqrt{4-x^2}} dx = 4\int_0^1 \frac{1}{\sqrt{4-x^2}} dx = 4\int_0^1 \frac{1}{2\sqrt{1-\frac{x^2}{4}}} dx = 2\int_0^1 \frac{1}{\sqrt{1-\frac{x^2}{4}}} dx =$$

$$\left(u = \frac{x}{2}, du = \frac{1}{2}dx\right)$$

$$= 4\int_0^{1/2} \frac{1}{\sqrt{1-u^2}} du =$$

Первообразная от $\frac{1}{\sqrt{1-u^2}}: sin^{-1}(u)$

$$= 4sin^{-1}(u)\Big|_0^{1/2} = \frac{2\pi}{3} \approx 2.0944$$

Программа ***integral.cpp*** последовательно решает данную задачу методом центральных прямоугольников при разбитии на *N*. Все анализы рекомендуется запускать на ~ 100M и большем размере задачи.

Задание:

1. Необходимо собрать проект с исходным файлом ***integral.cpp*** и запустить собранный исполняемый файл. Оценить время работы программы и корректность ее работы. Зафиксируйте это в отчете. Оцените вычислительную сложность задачи.
2. Добавить вычисления методом (в зависимости от варианта):
    I. Левых прямоугольников
    II. Правых прямоугольников
    III. Трапеций
    IV. Парабол (метод Симпсона)

    Добавление метода выполняется путем подмены исходной функции **mid_rectangles()** – метод средних прямоугольников на новую функцию, реализующую добавляемый метод. Сравнить точность и время работы с исходным вариантом.

3. С помощью инструментария Advisor необходимо получить метрики исполнения программы в последовательном режиме: Program Elapsed Time, GFLOPs, доля векторных/скалярных вычислений. Получить список хотспотов и roofline график.

Если горячие циклы/графики преимущественно ограничены memory крышами, обратить внимание на эффективность использования подсистемы памяти. **Изучить список рекомендаций для хотспотов, если таковые имеются и по возможности применить**, предварительно сохранив snapshot (для оценки эффективности примененных оптимизаций).

4. Оцените потенциальный прирост производительности при введении параллелизма через **Suitability** анализ.
5. Добавить параллелизм, там, где это возможно (например, в горячий цикл, итерации которого не зависимы между собой, либо зависимости разрешимы (использование private переменных, редукция)). Проверить эффективность работы параллельной версии приложения через Intel VTune. Проверить наличие ошибок в параллельной программе через Intel Inspector.
6. Подготовьте отчет (разделы ниже) и продемонстрируйте работу приложений и их характеристики преподавателю.

Подготовьте отчет со следующими разделами:

1. Опишите каким образом Вы проводили оптимизацию последовательно исполняемого приложения (смена последовательности выполнения циклов, разбивка циклов на подциклы, использование SoA вместо AoS и наоборот, раскручивание циклов и пр.). Оцените влияние последовательности доступа к данным и их выравнивания в памяти на векторизацию, если идет работа с данными из памяти.
2. Какие значения основных метрик производительности последовательно исполняемого приложения в Intel Advisor Вы получили (производительность в GFLOPs, пропускная способность памяти, использование векторных инструкций, roofline модель). Как изменились метрики при введении/оптимизации векторизации? Представьте сравнение метрик производительности последовательной работы с приложением, в котором используется «ручная» оптимизация.
3. Проведите прогнозирование ускорения выполнения программы за счет введения annotations в Intel Advisor. Осуществите введение параллелизма в последовательно исполняемую программу с использованием программной модели OpenMP, проведите анализ утилизации ресурсов процессора в Intel VTune, сравните подтверждение выполнения прогноза ускорения в Intel Advisor. Проверьте корректность выполнения параллельной программы, воспользовавшись Intel Inspector.