Практическая работа №10.

Задание 1. Анализ производительности CPU-параллельной программы (OpenMP)

Разработайте параллельную программу на C++ с использованием OpenMP для обработки большого массива данных (например, вычисление суммы, среднего значения и дисперсии).

Требуется:

- реализовать **базовую параллельную версию**;
- выполнить профилирование программы с использованием omp_get_wtime() и/или профилировщика (Intel VTune, gprof);
- определить:
    - долю параллельной и последовательной части программы;
    - влияние числа потоков на ускорение;
- проанализировать результаты в контексте **закона Амдала**.

Задание 2. Оптимизация доступа к памяти на GPU (CUDA)

Реализуйте ядро CUDA для обработки массива данных, демонстрирующее **разные паттерны доступа к памяти**.

Требуется:

1. реализовать две версии ядра:
    a. с эффективным (коалесцированным) доступом к глобальной памяти;
    b. с неэффективным доступом к памяти;
2. измерить время выполнения с использованием cudaEvent;
3. провести оптимизацию за счёт:
    a. использования разделяемой памяти;
    b. изменения организации потоков;
4. сравнить результаты и сделать выводы о влиянии доступа к памяти на производительность GPU.

Задание 3. Профилирование гибридного приложения CPU + GPU

Разработайте гибридную программу, в которой часть вычислений выполняется на CPU, а часть — на GPU.

Требуется:

1. реализовать гибридный алгоритм обработки массива данных;
2. использовать асинхронную передачу данных (cudaMemcpyAsync) и CUDA streams;

3. выполнить профилирование приложения:
    a. определить накладные расходы передачи данных;
    b. выявить узкие места при взаимодействии CPU и GPU;
4. предложить и реализовать одну оптимизацию, уменьшающую накладные расходы.

Задание 4. Анализ масштабируемости распределённой программы (MPI)

Реализуйте распределённую программу на MPI для вычисления агрегатной функции над большим массивом (например, сумма, минимум, максимум).

Требуется:

- измерить время выполнения при различном числе процессов;
- оценить **strong scaling** и **weak scaling**;
- проанализировать влияние коммуникационных операций (MPI_Reduce, MPI_Allreduce);
- сделать вывод о масштабируемости алгоритма и его практических ограничениях.

Контрольные вопросы

1. В чём отличие измерения времени выполнения от профилирования?
2. Какие виды узких мест характерны для CPU, GPU и распределённых программ?
3. Почему увеличение числа потоков или процессов не всегда приводит к ускорению?
4. Как законы Амдала и Густафсона применяются при анализе масштабируемости?
5. Какие факторы наиболее критичны для производительности гибридных приложений?