7 Лабораторная работа №4. «Метод доверительных интервалов при измерении времени выполнения параллельной OpenMP-программы»

7.1 Порядок выполнения работы

1. В программе, полученной в результате выполнения ЛР-3, так изменить этап Generate, чтобы генерируемый набор случайных чисел не зависел от количества потоков, выполняющих программу. Например, на каждой итерации i перед вызовом rand_r можно вызывать функцию srand(f(i)), где f – произвольно выбранная функция. Можно придумать и использовать любой другой способ.

2. Заменить вызовы функции gettimeofday на omp_get_wtime.

3. Распараллелить вычисления на этапе Sort, для чего выполнить сортировку в два этапа:

   - Отсортировать первую и вторую половину массива в двух независимых нитях (можно использовать OpenMP-директиву ”parallel sections”);
   - Объединить отсортированные половины в единый массив.

4. Написать функцию, которая один раз в секунду выводит в консоль сообщение о текущем проценте завершения работы программы. Указанную функцию необходимо запустить в отдельном потоке, параллельно работающем с основным вычислительным циклом.

5. Обеспечить прямую совместимость (forward compatibility) написанной параллельной программы. Для этого все вызываемые функции вида «omp_*» можно условно переопределить в препроцессорных директивах, например, так:

```
#ifdef _OPENMP
    #include “omp.h”
#else
    int omp_get_num_procs() { return 1; }
#endif
```

6. Провести эксперименты, варьируя N от $min(\frac{N_x}{2},\ N_1)$ до N_2, где значения N_1 и N_2 взять из ЛР-1, а N_x – это такое значение N,

при котором накладные расходы на распараллеливание превышают выигрыш от распараллеливания. Написать отчёт о проделанной работе. Подготовиться к устным вопросам на защите.

7. **Необязательное задание на «четвёрку».** Уменьшить количество итераций основного цикла с 100 до 10 и провести эксперименты, замеряя время выполнения следующими методами:

    - Использование минимального из десяти полученных замеров;
    - Расчёт по десяти измерениям доверительного интервала с уровнем доверия 95%.

    Привести графики параллельного ускорения для обоих методов в одной системе координат, при этом нижнюю и верхнюю границу доверительного интервала следует привести двумя независимыми графиками.

8. **Необязательное задание на «пятёрку»:** в п.3 задания на этапе Sort выполнить параллельную сортировку не двух частей массива, а k частей в k нитях (тредах), где k – это количество процессоров (ядер) в системе, которое становится известным только на этапе выполнения программы с помощью команды
«k = omp_get_num_procs()».

7.2 Состав отчета

1. Титульный лист с названием вуза, ФИО студентов и названием работы.

2. Содержание отчета (с указанием номера страниц и т.п.).

3. Краткое описание решаемой задачи.

4. Характеристика использованного для проведения экспериментов процессора, операционной системы и компилятора GCC (точное название, номер версии/модели, разрядность, количество ядер и т.п.).

5. Полный текст программы и использованных в процессе работы скриптов и инструментов с указанием параметров запуска.

6. Подробные выводы с анализом каждого из приведённых графиков.

Отчёт предоставляется в бумажном или электронном виде вместе с полным текстом программы. По требованию преподавателя нужно быть готовыми скомпилировать и запустить этот файл на компьютере в учебной аудитории (или своём ноутбуке).

7.3 Подготовка к защите

1. Уметь объяснить каждую строку программы, представленной в отчёте.

2. Уметь объяснить выводы, полученные в результате работы.

3. Знать назначение каждой директивы OpenMP, использованной в программе.

4. Повторить материал лекции №4, прочитать главу про OpenMP в методическом пособии.

5. Знать ответы на вопросы из разделов «Задание» книги Антонова «Параллельное программирование с использованием технологии OpenMP: Учебное пособие» (см. страницы 12, 28, 35, 54).

8 Лабораторная работа №5. «Параллельное программирование с использованием стандарта POSIX Threads»

8.1 Порядок выполнения работы

1. Взять в качестве исходной OpenMP-программу из ЛР-5, в которой распараллелены все этапы вычисления. Убедиться, что в этой программе корректно реализован одновременный доступ к общей переменной, используемой для вывода в консоль процента завершения программы.

2. Изменить исходную программу так, чтобы вместо OpenMP-директив применялся стандарт «POSIX Threads»:

    - для получения оценки **«3»** достаточно изменить только один этап (Generate, Map, Merge, Sort), который является узким местом (bottle neck), а также функцию вывода в консоль процента завершения программы;
    - для получения оценки **«4»** необходимо изменить всю программу, но допускается в качестве расписания циклов использовать «schedule static»;
    - для получения оценки **«5»** необходимо хотя бы один цикл распараллелить, реализовав вручную расписание «schedule dynamic» или «schedule guided».

3. Провести эксперименты и по результатам выполнить сравнение работы двух параллельных программ («OpenMP» и «POSIX Threads»), которое должно описывать следующие аспекты работы обеих программ (для различных N):

    - полное время решения задачи;
    - параллельное ускорение;
    - доля времени, проводимого на каждом этапе вычисления («нормированная диаграмма с областями и накоплением»);
    - количество строк кода, добавленных при распараллеливании, а также грубая оценка времени, потраченного на распараллеливание (накладные расходы программиста);
    - остальные аспекты, которые вы выяснили самостоятельно (**Обязательный пункт**);