Домашняя работа №5

OpenMP

Цель работы: знакомство со стандартом OpenMP.

Инструментарий и требования к работе: рекомендуется использовать C, C++. Возможно использовать Python и Java. Стандарт OpenMP 2.0.

Порядок выполнения работы:

1. Изложить в письменной форме:
    a. описание принципа работы OpenMP и ключевых элементов (конструкций и переменных окружения), которые понадобились для написания программы;
    b. описание реализуемого алгоритма;
2. Написать программу, решающую задачу.

Содержание отчета

1. Теоретическая часть (пункт 1 из Порядка выполнения);
2. Описание работы написанного кода (пункт 2 из Порядка выполнения, экспериментальная часть);
3. Привести графики времени работы программы (некоторые графики из следующих подпунктов можно объединить в один): [перечислить static, dynamic]
    a. при различных значениях числа потоков при одинаковом параметре schedule*;
    b. при одинаковом значении числа потоков при различных параметрах schedule*;
    c. с выключенным openmp и с включенным с 1 потоком.

    * `schedule(kind[, chunk_size])` – kind in {‘static, ‘dynamic’}, chunk_size – 1 и несколько (2-3) других значений

4. Листинг кода с указанием компилятора/интерпретатора (подробнее Оформление кода в отчёте).

Примечания:

1. Будет оцениваться как правильность результата, так и скорость работы.
2. Без кода теория не оценивается, поэтому пытаться посылать отчет в таком случае бессмысленно.
3. Плагиат карается отрицательными баллами за всю работу.
4. Стандарт OpenMP 2.0: https://www.openmp.org//wp-content/uploads/cspec20.pdf

Описание задания

Аргументы программе передаются через командную строку:

hw5.exe <кол-во_потоков> <имя_входного_файла> <имя_выходного_файла> [<параметры_алгоритма>]

Число потоков может равняться 0 и больше. 0 соответствует значению числа потоков по умолчанию.

Помимо выполнения самого задания (условие ниже), необходимо провести замеры времени работы на вашей рабочей машине – 3 эксперимента, описанных в содержании отчета (с. 1).

В данном случае имеется в виду время работы алгоритма без времени на считывание данных и вывод результат.

На вход вы получаете изображение и результирующее изображение пишете в другой файл, а результаты замеров выводите в консоль. Формат вывода: “Time (%i thread(s)): %g ms\n”

Время работы выводится только в консоль.

Варианты

В этой работе вам предлагается решить одну из задач. Задачи разделены по уровням сложности. Чем выше уровень, тем больше баллов вы получите за задачу.

Easy

Вариант 1. Префиксная сумма

Подсчет инклюзивной префиксной суммы для заданного массива типа float.

<параметры_алгоритма> отсутствуют

Входной файл содержит данные следующим образом:

```
n

/* входной массив данных, элементы разделены пробелом */
```

Вывод:

```
n

/* выходной массив данных, элементы разделены пробелом */
```

Normal

Вариант 2. Gaussian Blur (с использованием Box Blur Approximation)

Применение гауссовского размытия с применением Box Blur Approximation к изображению в оттенках серого.

<параметры_алгоритма> = <numbox> <sigma>

- **<numbox> - number of boxes, натуральное число**
- **<sigma> - параметр алгоритма, положительный float**

Входной файл содержит данные в формате PGM (P5).

В качестве выходного файла будет новое изображение в формате PGM (P5).

Подробнее про формат хранения изображений после условий задач.

Hard

Вариант 3. Автоматическая контрастность изображения

Описание задачи: Изображение может иметь плохую контрастность: используется не весь диапазон значений, а только его часть. Например, если самые тёмные точки изображения имеют значение 20, а не 0.

Задание состоит в том, чтобы изменить значения пикселей таким образом, чтобы получить максимальную контрастность: растянуть диапазон значений до [0; 255], но при этом не изменить оттенки (то есть в цветных изображениях нужно одинаково изменять каналы R, G и B).

<параметры_алгоритма> = <коэффициент>

При вычислении растяжения игнорировать некоторую долю (по количеству) самых тёмных и самых светлых точек (для RGB в каждом канале отдельно): **<коэффициент>** (диапазон значений [0.0, 0.5)). Это позволяет игнорировать шум, который незаметен глазу, но мешает

автоматической настройке контрастности. Растяжение диапазона следует выполнять с насыщением, чтобы проигнорированные пиксели не вышли за границы [0; 255].

Если изображение состоит из одного цвета, то оно не обрабатывается (не меняется).

Формат хранения изображений

Формат входных и выходных изображений: PNM (P5 или P6). Во всех PNM файлах (pgm, ppm) комментарии отсутствуют.

P5 (PGM)	“P5\n<width> <height>\n255\n<Gray_data>”
P6 (PPM)	“P6\n<width> <height>\n255\n<RGB_data>”

Дополнительные сведения (код)

1. Корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок: не удалось открыть файл, формат файла не поддерживается.

   Если программе передано значение, которое не поддерживается – следует сообщить об ошибке;
2. В программе можно вызывать только стандартные библиотеки (например, <bits/stdc++.h> таковой не является и ее использование влечет за собой потерю баллов). То есть сторонние библиотеки использовать нельзя (библиотека <omp.h> и модули для подключения openmp конечно разрешены);
3. Если программа использует библиотеки, которые явно не указаны в файле с исходным кодом (например, <algorithm>), то за это также будут снижаться баллы.

Оформление кода в отчёте

1. Никаких скринов кода – код в отчет добавляется только текстом;
2. Шрифт: `Consolas` (размер 10-14 на ваше усмотрение);
3. Выравнивание по левому краю;
4. Подсветка кода допустима. Текст должен быть читаемым (а не светло-серый текст, который без выделения на белом не разобрать);
5. В раздел Листинг код вставляется полностью в следующем виде:

   <Название файла>

   <Его содержимое>

   Файлы исходных кодов разделяются новой строкой.

   Например,

   main.cpp

   ```
   int main()
   {
     return 0;
   }
   ```

   tmain.cpp

   ```
   int tmain()
   {
     return 666;
   }
   ```

6. Фон белый (актуально для тех, у кого копипаста кода идет вместе с фоном темной темы из IDE).