МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский ядерный университет «МИФИ»»

ЛАБОРАТОРНАЯ РАБОТА №4:

«Технология OpenMP. Особенности настройки»

Аверин Владислав

Группа Б19-505

Октябрь, 2021

Содержание

Характеристики лабораторного оборудования

Процессор: 11th Gen Intel Core i7-1185G7 3.00Ghz (8 CPUs)

RAM: 16Гб DDR4 3200МГц

Используемая версия _OpenMP: 201511 (December, 2015)

Операционная система: OS Linux Manjaro KDE Plasma 5.22.5; версия ядра: 5.10.68-1-MANJARO (64-бита), работа от сети

Редактор кода: Visual Studio Code 1.60.1

Компилятор: GCC 11.1.0

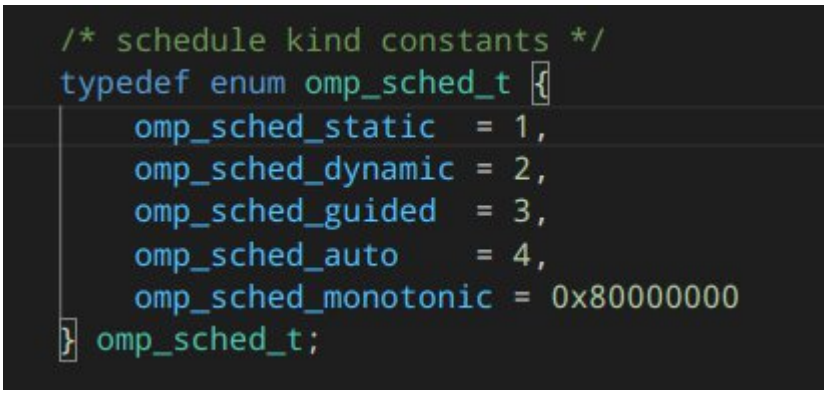
Параметры командной строки: -O0 -fopenmp (без оптимизации и отладочной информации для чистоты эксперимента)

Описание директив, опций и функций

- _***OPENMP*** - переменная препроцессора, определяющая дату принятия используемого стандарта OpenMP; возвращает int переменную в виде yyyy:mm

- Опция ***schedule(type, chunk_size)*** определяет способ разделения итераций цикла for по нитям. Параметр type может принимать следующие значения (типы распределения):
    - ***static*** - блочно-циклическое распределение итераций цикла; размер чанка - chunk_size (default : кол-во итераций / кол-во нитей; большие по размеру чанки отдаются наипервейшим нитям).
    - ***dynamic*** - динамическое распределение итераций с фиксированным размером блока: сначала каждая из нитей получает chunk_size итераций (default : 1), далее та нить, которая закончила выполнение своей порции, получает следующую свободную порцию из chunk_size итераций. Так будет происходить до тех пор, пока все итерации не будут обработаны.
    - ***guided*** - так же динамическое распределение итераций, но с уменьшающимся размером чанка пропорционально оставшейся части цикла: начальное значение - зависит от реализации; конечное - chunk_size (default: 1);
    - ***auto*** - способ распределения выбирается компилятором и/или исполняющей системой;
    - ***runtime*** - способ распределения выбирается во время работы программы по значению директивы OMP_SCHEDULE. Параметр chunk_size при этом не задается (определяется дефолтным для данного способа);
- ***int omp_get_dynamic()*** - возвращает значение переменной *OMP_SCHEDULE*, которая показывает, разрешено ли системе динамически изменять кол-во нитей, используемых для выполнения параллельной области (например, для оптимизации использования ресурсов системы);
- ***double omp_get_wtick()*** - возвращает в вызывавшей нити разрешение таймера в секундах, что можно рассматривать как предельную точность таймера;
- ***int omp_get_nested()*** - возвращает значение переменной *OMP_NESTED*, которая показывает, разрешен ли вложенный параллелизм. По умолчанию (*OMP_NESTED* = 0) любой вложенный цикл выполняется последовательно соответствующей зашедшей в него нитью. Если же *OMP_NESTED* = 1, то эта нить породит свою группу нитей и станет в ней нитью-мастером.
- ***int omp get max active levels()*** - возвращает значение переменной *OMP_MAX_ACTIVE_LEVELS*, которая обозначает максимальную глубину возможного распаралелленногго цикла (default : 1)

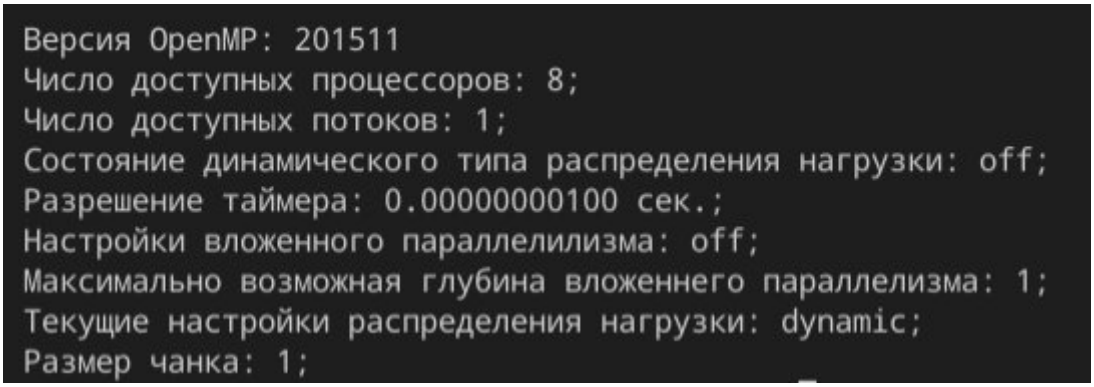
- ***void omp get schedule(omp_sched_t* type, int* chunk_size)*** - возвращает обновленные значения chunk_size и type, соответствующие текущим дефолтным настройкам системы. Значения type определены в енаме библиотеки OpenMP:



```
/* schedule kind constants */
typedef enum omp_sched_t {
    omp_sched_static  = 1,
    omp_sched_dynamic = 2,
    omp_sched_guided  = 3,
    omp_sched_auto    = 4,
    omp_sched_monotonic = 0x80000000
} omp_sched_t;
```

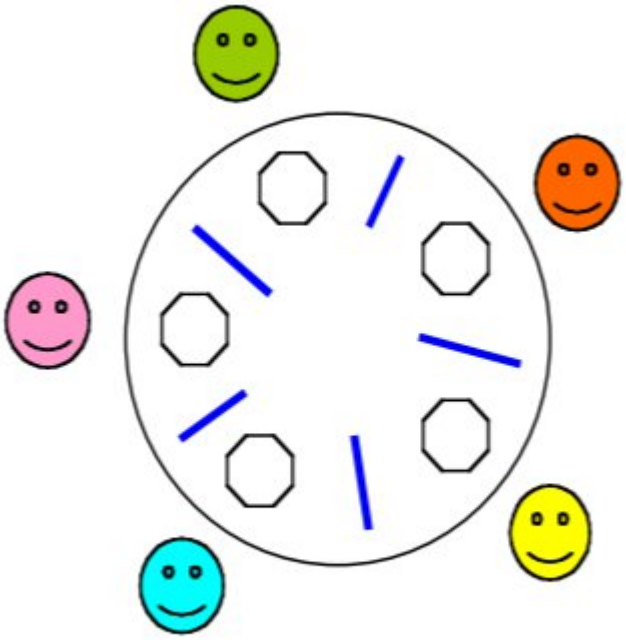
Полученные настройки рабочей среды

Программный вывод:



Пример вычислительного алгоритма на основе механизма явных блокировок

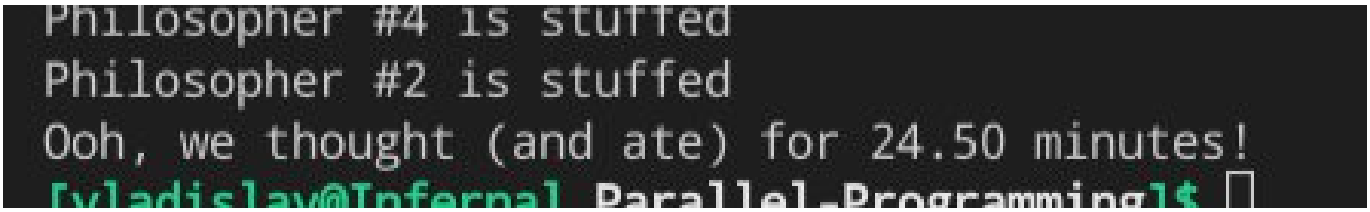
На самом деле, алгоритм не сколько вычислительный, сколько интерактивный. Он решает одну конкретную задачу, но наглядно показывает возможности явных блокировок нитей. Задачу назовем ***«Филосовский обед».***



За круглым обеденным столом сидят N китайских философов.Перед каждым философом стоит тарелка с рисом - его обед. Чтобы начать трапезу, философу необходимы свободные левая и правая палочки, причем каждая палочка разделена между двумя соседними философами. Но как известно, китайским философам дай только повод пофилософсвовать. Таким образом, пока оба соседа получают вселенскую энергию от первозданности бытия и пребывания на триллионах таких же планет, как эта, философ, находящийся между ними, может придаться тихому и планоразмерному чревоугодию. После того, как он выполнил свою миссию по истреблению съестных запасов, перед ним возникает как на Скатерти-Самобранке следующая порция (Ма-а-agic), дабы не отвлекать его от гармонии со слиянием вечного с бесконечным и созерцания этого фрактального подобия. Продолжительность процесса философствования каждого из участников «голодных» игр определяется случайным образом, так же, как и время приема одной порции (но в два раза больше:)). Если при окончании периода раздумий перед философом не наблюдается полного набора палочек, он начинает снова думать (видимо, пытаясь материализовать новые палочки).

Необходимо посчитать время, за которое все философы за столом примут _MEALS порций.

Пример работы программы для значений N = 5 и _MEALS = 3:



Время, полученное таким образом, всегда будет разным (в каком-то интервале для конкретных значений), т.к. все процессы длятся разное время, и неизвестно, какой из элементов массива в данный момент залочет / разлочет «палочки»