РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Отчет по лабораторной работе № 3

Дисциплина: Параллельное программирование

Студент: Логинов Сергей Андреевич

Группа: НФИбд-01-18

МОСКВА 2021г

Задание

Задание №1

- Напишите простейшую программу, которая ждет некоторое время, например 1 секунду. Для ожидания используйте встроенную процедуру ***sleep*** с аргументом 1.
- Замерьте время ее работы N раз с помощью ***omp_get_wtime( )***. Распечатайте результаты замеров времени, например для N = 10. Убедитесь, что почти всегда полученное время отличается от 1.
- Проведем N = 10^6 замеров времени. Вместо процедуры sleep необходимо использовать произвольные вычисления в цикле.
- Замерьте время работы новой подпрограммы для N = 10^6.
- Вычислить среднее значение и выборочную дисперсию от полученных замеров.

Сперва напишем подпрограмму wait, которая реализует функцию sleep с переданным в подпрограмму аргументом:

```
subroutine wait(i) ! подпрограмма, реализующая "сон" на 1 секунду

    implicit none

    integer, intent(in) :: i

    call sleep(i)

end subroutine wait
```

Далее в теле основной программы main проведем замеры времени работы подпрограммы в цикле

```
! замеряем время работы подпрограммы wait
    do i=1,10,1
        t1 = omp_get_wtime()
        call wait(1)
        t2 = omp_get_wtime()
        print *,i, "замер : ", t2-t1
    end do
```

Результат выполнения:

```
(base) sergejloginov@MacBook-Air-Sergej 2_openmp % gfortran -fopenmp lab
01_1.f90
(base) sergejloginov@MacBook-Air-Sergej 2_openmp % ./a.out
           1 замер :    1.0025540003553033
           2 замер :    1.0048519996926188
           3 замер :    1.0004489999264479
           4 замер :    1.0040889997035265
           5 замер :    1.0019970005378127
           6 замер :    1.0050290003418922
           7 замер :    1.0050550000742078
           8 замер :    1.0007750000804663
           9 замер :    1.0050249993801117
          10 замер :    1.0010899994522333
```

Действительно, результаты во всех десяти замерах отличаются от единицы.

Далее проводим N = 10^6 замеров времени, функцию sleep заменим на вычисление экспоненты и натурального логарифма от переменной, в которую записывается системное время. Все значения наших измерений запишем в массив arr (понадобится для дальнейшего задания). Не забудем в самом начале выделить память под массив.

```
! проведем большее число замеров (10^6), результаты будем записывать
! в массив для дальнейшей работы с его элементами
    N = 10**6
    if( .not. allocated(arr) ) allocate(arr(1:N)) ! выделяем память(с
проверкой выделения) под массив
    do i=1, N
        t1 = omp_get_wtime()
        summ = exp(t1) + log(t1) ! произвольные вычисления
        t2 = omp_get_wtime()
        arr(i) = t2-t1  ! запись в массив
    end do
```

От полученных замеров требуется найти среднее значение и выборочную дисперсию.

Найдем среднее значение в массиве замеров:

```
med = sum(arr) / size(arr) ! находим среднее значение массива
```

Формула выборочной дисперсии:

$$S_n^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \dot{x})^2, \text{ где } \dot{x} - \text{выборочное среднее}$$

Разобьем формулу на части. Для начала заполним массив значениями под знаком суммы. Далее проведем суммирование и поделим на N.

Заполняем массив, не забудем выдлить под него память:

```
    if( .not. allocated(arr_disp) ) allocate(arr_disp(1:N)) ! второй массив для поиска дисперсии, в нем будут элементы вида
        !(Xi - X`)^2(часть формулы дисперсии), где Xi - замер времени выполнения вычислений из прошлого цикла а X` - найденное среднее значение
        do i=1, N
            arr_disp(i) = (arr(i) - med)**2
        end do
```

Далее вычисляем дисперсию. Просуммируем элементы прошлого массива и поделим эту сумму на N, после чего вычислим корень от полученного значения:

```
    disp = sqrt(sum(arr_disp)/N) ! дополним формулу и найдем дисперсию
```

Осталось распечатать полученные значения. Результат:

```
Median =   4.7870090231299399E-008
Disp =   2.1879373013270343E-007
```

Задание №2

- Создайте программу с параллельной областью. Распечатайте количество созданных потоков. С помощью какой функции это делается?
- Реализовать динамическое изменение потоков в двух вариантах: переменная окружения и функция omp_set_num_threads
- Узнать и создать оптимальное количество потоков для своего процессора
- Создать многопоточную программу с четырьмя потоками, которая будет суммировать входящий массив.
- Протестировать программу из пункта выше.