ПрИб-181	Лабораторный практикум СРС	Зачёт
Кащенко В. А.	СРС 01.11.2021	

Цель работы: Выполнить задание m.volsu

Задание:
Напишите параллельную программу с использованием OpenMP, в которой N нитей параллельно вычисляют сумму чисел от 1 до N. Для сложения результатов вычисления нитей воспользуйтесь OpenMP-параметром reduction, директивой atomic, директивой critical, и замками (lock). Проведите серию экспериментов на персональном компьютере по исследованию влияния различных инструментов синхронизации на время работы программы. Оформить подробный отчет в формате pdf.

Код программы:

```
#include <stdio.h>
#include <omp.h>

int main()
{
	int k, N;
	int sum = 0;
	double start_time, end_time;

	scanf("%d%d", &k, &N);

	// reduction
	start_time = omp_get_wtime();
	#pragma omp parallel num_threads(k) reduction(+: sum)
	{
		#pragma omp for
		for (int i = 1; i <= N; ++i)
			sum += i;
	}
	end_time = omp_get_wtime();
	printf("Sum = %d\n", sum);
	printf("Время работы reduction = %f\n\n", end_time - start_time);

	sum = 0;

	// atomic
	start_time = omp_get_wtime();
```

```
	#pragma omp parallel num_threads(k)
	{
		#pragma omp for
		for (int i = 1; i <= N; ++i)
		{
			#pragma omp atomic
			sum+=i;
		}
	}
	end_time = omp_get_wtime();
	printf("Sum = %d\n", sum);
	printf("Время работы atomic = %f\n\n", end_time - start_time);

	sum = 0;

	// critical
	start_time = omp_get_wtime();
	#pragma omp parallel num_threads(k)
	{
		#pragma omp for
		for (int i = 1; i <= N; ++i)
		{
			#pragma omp critical
			sum+=i;
		}
	}
	end_time = omp_get_wtime();
	printf("Sum = %d\n", sum);
	printf("Время работы critical = %f\n\n", end_time - start_time);
	sum = 0;

	// lock
	start_time = omp_get_wtime();
	omp_lock_t lock;
	omp_init_lock(&lock);
	#pragma omp parallel num_threads(k)
	{
		omp_set_lock(&lock);
		sum += omp_get_thread_num() + 1;
		omp_unset_lock(&lock);
	}
	end_time = omp_get_wtime();
	omp_destroy_lock(&lock);
	printf("Sum = %d\n", sum);
	printf("Время работы lock = %f\n\n", end_time - start_time);
}
```

Результат замеров:

12 потоков, N = 100

```
(base) uke_zebrano@pop-os:~/Рабочий стол/labs_parallel/special_lab$ ./a.out
12
100
Sum = 5050
Время работы reduction = 0.000921

Sum = 5050
Время работы atomic = 0.000022

Sum = 5050
Время работы critical = 0.000087

Sum = 5050
Время работы lock = 0.005619
```

12 потоков, N = 1000

```
(base) uke_zebrano@pop-os:~/Рабочий стол/labs_parallel/special_lab$ ./a.out
12
1000
Sum = 500500
Время работы reduction = 0.010015

Sum = 500500
Время работы atomic = 0.000031

Sum = 500500
Время работы critical = 0.000149

Sum = 500500
Время работы lock = 0.017146
```

12 потоков, N = 10000

```
(base) uke_zebrano@pop-os:~/Рабочий стол/labs_parallel/special_lab$ ./a.out
12
10000
Sum = 50005000
Время работы reduction = 0.000829

Sum = 50005000
Время работы atomic = 0.001146

Sum = 50005000
Время работы critical = 0.003914

Sum = 50005000
Время работы lock = 0.148497
```

6 потоков, N = 100

```
(base) uke_zebrano@pop-os:~/Рабочий стол/labs_parallel/special_lab$ ./a.out
6
100
Sum = 5050
Время работы reduction = 0.000470

Sum = 5050
Время работы atomic = 0.000022

Sum = 5050
Время работы critical = 0.000056

Sum = 5050
Время работы lock = 0.005713
```

6 потоков, N = 1000

```
(base) uke_zebrano@pop-os:~/Рабочий стол/labs_parallel/special_lab$ ./a.out
6
1000
Sum = 500500
Время работы reduction = 0.000489

Sum = 500500
Время работы atomic = 0.000126

Sum = 500500
Время работы critical = 0.000407

Sum = 500500
Время работы lock = 0.020008
```

6 потоков, N = 10000

```
(base) uke_zebrano@pop-os:~/Рабочий стол/labs_parallel/special_lab$ ./a.out
6
10000
Sum = 50005000
Время работы reduction = 0.000444

Sum = 50005000
Время работы atomic = 0.001240

Sum = 50005000
Время работы critical = 0.003386

Sum = 50005000
Время работы lock = 0.146696
```

По результатам замеров можно сказать, самым лучшим в плане скорости инструментом синхронизации является директива atomic и reduction.
Поэтому её использование можно ставить в приоритет.
(atomic блокирует лишь одну переменную, а не все возможные как critical или lock)
При маленьких значениях лучше использовать atomic, а при больших reduction.

Вывод: Выполнено задание m.volsu.