Практикум по курсу
"Суперкомпьютеры и параллельная обработка данных"
Разработка параллельной версии программы для задачи Red-Black2D

ОТЧЕТ
о выполненном задании

студента 327 учебной группы факультета ВМК МГУ
Черепниной Марии Сергеевны

Москва, 2020 г.

1. Постановка задачи

Требуется реализовать параллельную версию программы для задачи Red-Black2D с помощью технологии OpenMP и MPI.

2. Реализация с помощью OpenMP

Для параллельной работы программы требовались следующие модификации исходного кода:

а. Изменить порядок работы с элементами матриц с «внешний цикл итерируется по столбцам, внутренний - по строкам» на обратный: «внешний цикл итерируется по строкам, внутренний - по столбцам»

б. Над первым циклом в функции `relax()` добавить клаузу

```
omp parallel for schedule(static) collapse(2) reduction(max:eps)
```

в. Над циклом в функции `verify()` добавить клаузу

```
omp parallel for schedule(static) collapse(2) reduction (+:s)
```

г. Над оставшимися двумя циклами добавит клаузу

```
omp parallel for collapse(2) schedule(static)
```

Компиляция программы производилась командой

```
$ gcc -std=c99 -Wall -fopenmp -o run redb_2d.c
```

Пример запуска на 8-ми потоках:

```
$ OMP_NUM_THREADS=8 ./run
```

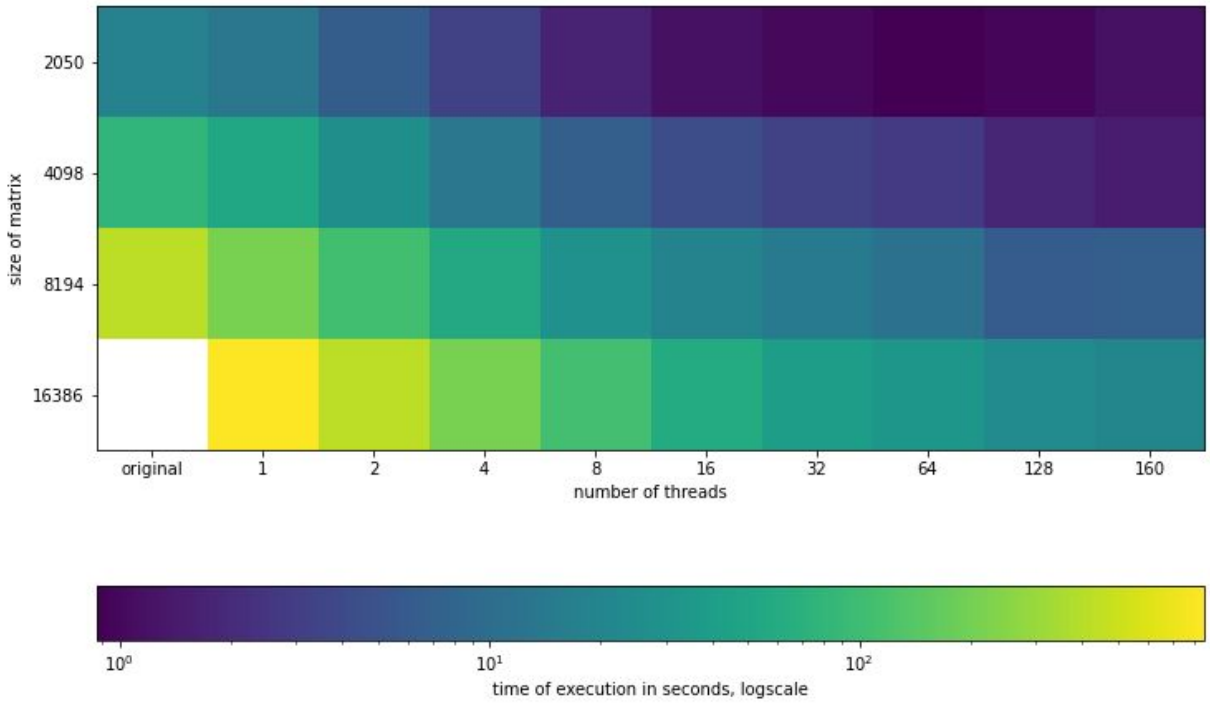
3. Результаты версии с OpenMP

Время выполнения программы в секундах

		Размер одной стороны квадратной матрицы			
		2050	4098	8194	16386
Количество потоков	Последовательное исполнение	18.30678	80.18694	429.29351	-
	1	13.41572	53.64603	215.37072	858.83583
	2	6.70976	26.83452	107.37579	429.65468

	4	3.38475	13.50519	55.20752	215.70689
	8	1.73417	6.94515	28.46837	109.35427
	16	1.18653	4.42065	18.91277	59.71843
	32	1.01328	3.39907	14.72773	40.90655
	64	0.86701	2.80041	11.31484	32.96731
	128	0.98630	1.82509	6.52785	23.87496
	160	1.17019	1.50613	7.03084	20.70814

Для наглядности построим по табличным данным тепловую карту:



4. Выводы по OpenMP

С увеличением числа потоков производительность обычно растёт. Для матриц с размером от 4096 до 16386 наименьшее время работы достигается на количестве потоков 128 - 160. Для матрицы размера 2050 лучшее время достигнуто при использовании 64 потоков.

5. Реализация с помощью MPI

Вычисления делятся на K процессов: каждый процесс обрабатывает N/K строк, где N - размерность входной матрицы. Поскольку значения i-ой строки зависят от значений (i-1)-ой и (i+1)-ой, после каждого обновления (как черных, так и красных клеток) необходимо обмениваться значениями между процессами, работающими в соседних областях матрицы:

```
if (rank!=0) {MPI_Send(A[first_row], N, MPI_FLOAT, rank-1, up_send_tag,
MPI_COMM_WORLD);}
    if (rank!=num_workers-1){
        MPI_Recv(tmp_A_row, N, MPI_FLOAT, rank+1, up_send_tag, MPI_COMM_WORLD,
&status);
        // (j + last_row) % 2 == 1   =>   j = 1+(last_row % 2)
        for (int j=1+(last_row % 2); j<=N-2; j+=2) {A[last_row][j] =
tmp_A_row[j];}
    }
    if (rank!=num_workers-1) {MPI_Send(A[last_row-1], N, MPI_FLOAT, rank+1,
down_send_tag, MPI_COMM_WORLD);}
    if (rank!=0){
        MPI_Recv(tmp_A_row, N, MPI_FLOAT, rank-1, down_send_tag,
MPI_COMM_WORLD, &status);
        for (int j=1+((first_row-1) % 2); j<=N-2; j+=2) {A[first_row-1][j] =
tmp_A_row[j];}
    }
```

Когда процессы посылают соседям некоторую строку, всегда остается процесс, не вызывающий `MPI_Send` (например, первый процесс никому не посылает первую обрабатываемую им строку), следственно взаимная блокировка невозможна.

Компиляция программы производилась командой

```
$ mpicc -std=c99 -o run_1 redb_2d.c -lm
```

Пример запуска на 8-ми процессах:

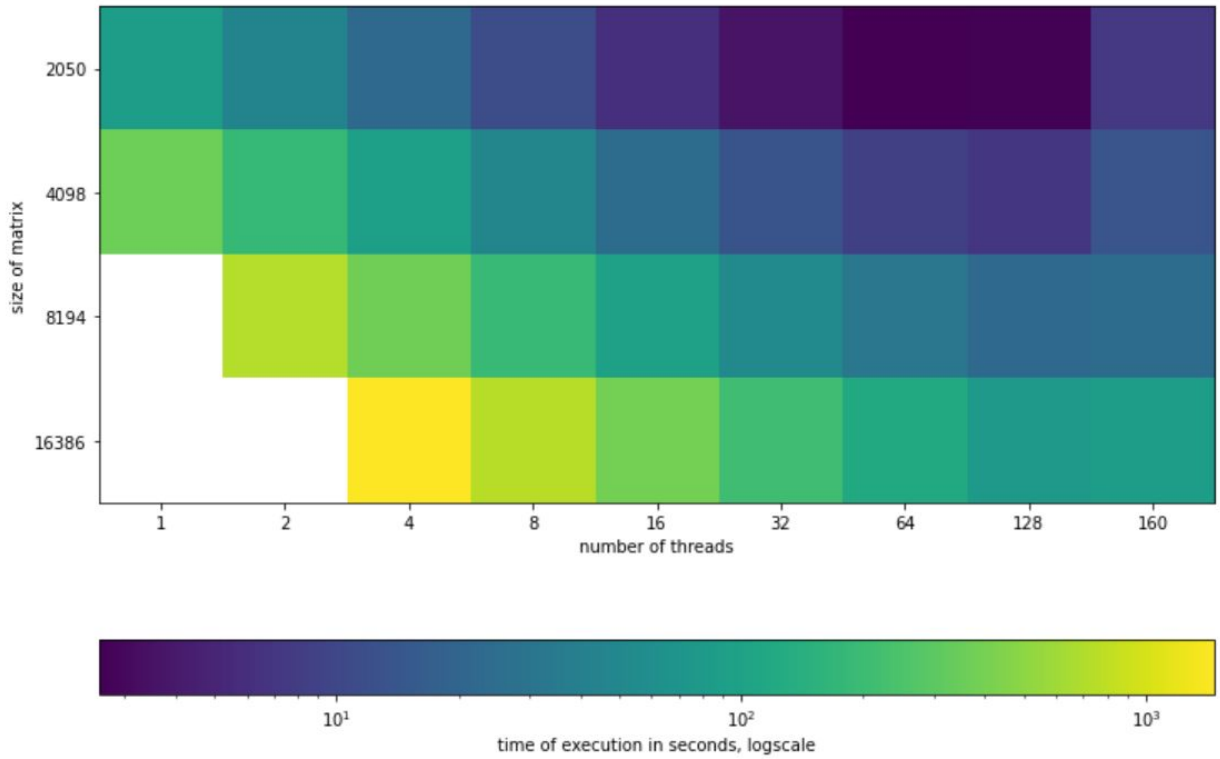
```
$ mpisubmit.bg -n 8 -w 00:30:00 run
```

6. Результаты версии с MPI

Время выполнения программы в секундах

		Размер одной стороны квадратной матрицы			
		2050	4098	8194	16386
Количество потоков	1	89.28956	365.03554	-	-
	2	44.88615	183.39658	733.35296	-
	4	22.66178	92.47809	370.09185	1479.85751
	8	11.59989	47.13755	188.57530	753.51259
	16	6.14681	24.58855	98.05730	390.79387
	32	3.57050	13.57350	53.27449	210.32155
	64	2.58568	8.57687	31.78598	121.87612
	128	2.70318	7.10625	22.92413	81.23123
	160	7.37793	13.89423	24.39141	91.08153

Для наглядности построим по табличным данным тепловую карту:



7. Выводы по MPI

С увеличением числа процессов производительность обычно растёт. В большинстве случаев наименьшее время работы достигается при исполнении на 128 процессах.