Введение в параллельные вычисления

Лекция 6. Some further tricks on optimization Of OpenMP programs (еще несколько трюков по оптимизации OpenMP-программ)

КС-40, КС-44
РХТУ

Преподаватель
Митричев Иван Игоревич, к.т.н.,
ассистент кафедры ИКТ

2017

Оптимизация (избавление от присвоений и раскрутка цикла)

```
for (i=1;i<N;i++)
        for (j=1;j<N;j++)
        newgrid[i][j]= 0.25 * (grid[i1][j]+grid[i+1][j]+grid[i][j1]+grid[i][j+1]);
for (i=1;i<N;i++)
        for (j=1;j<N;j++)
        grid[i][j]= 0.25 * (newgrid[i-1][j]+newgrid[i+1][j]+newgrid[i][j-1]+newgrid[i][j+1]);
maxdiff=0;
for (i=1;i<N;i++)
        for (j=1;j<N;j++)
        maxdiff = max(maxdiff, fabs(newgrid[i][j]grid[i][j]));
```

l05_04.cpp

```
for (i=1;i<N;i++)
        for (j=1;j<N;j++)
        grid[i][j]=newgrid[i][j];
```

l05_05.cpp

Размер области	200	1000
До оптимизации (OpenMP + collapse)	0,208	6,741
После оптимизации (OpenMP + collapse)	0,123	**3,656**
До оптимизации кода, но с оптимизацией компилятора gcc -O2	0,077	2,099
После оптимизации кода, с оптимизацией компилятора gcc –O2	0,043	**0,931**
После оптимизации кода, с оптимизацией компилятора, использование **parallel for**	0,047	0,976

Вложенное распараллеливание (сортировка слиянием, sections)

```
 #pragma omp parallel num_threads(2)
shared(array) if (R-L>1000000)
{
#pragma omp sections nowait
{
        #pragma omp section
        {
        merge_sort(array, L,(L+R)/2);
        }
        #pragma omp section
        {
        merge_sort(array, (L+R)/2+1,R);
        }
}
}
```

l06_01.cpp

Вложенное распараллеливание (tasks)

```
#pragma omp task shared(array,L,R)
{
merge_sort(array, temp, L,(L+R)/2);
}

#pragma omp task shared(array,L,R)
{
merge_sort(array, temp, (L+R)/2+1,R);
}

#pragma omp taskwait // barrier!
```

```
// in main():
 clock_t begin = clock();

#pragma omp parallel num_threads(4)
shared(array,temp) if (N>=100)
{
        #pragma omp single
        merge_sort(array,temp,0,N-1);
}
clock_t end = clock();
```

l06_02.cpp

Тестирование (10 000 000 элементов)

1 поток (num_threads=1)	5,2
OpenMP – 2 потока, sections	10,8
OpenMP – 2 потока, tasks	6,5
1 поток (num_threads=1), без использования выделения памяти в рекурсивной функции	**1,22**
OpenMP – 2 потока, tasks, без использования выделения памяти в параллельном коде l06_03.cpp	1,40