# Introduction to HPC
## Part 1: OpenMP

lsxm74*

## I. Introduction

This paper is described a design and implementation of an OpenMP parallelisation which helps the small Gauss-Seidel iteration to solve the image reconstruction problem.

## II. Design parallelisation

The specific part we need paralleled is the Gauss-Seidel iteration contains nested for loop, the main problem is to avoid data races by different threads. To do that we need to split the loops to different part and then parallel them. For this question it used Red-Black ordering as following Fig 1: which because

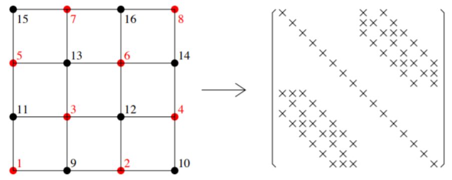

Fig. 1: Red-Black Ordering

red point is only depending on surrounding black point and black point is only depending on surrounding red point, so the two part can be run parallel without data races.

There is many way of doing that, the way I chose is by looking at parity of point, as computer counting from 0, sum of the row and column of red point will all be even number, such as in Fig 1, sum will be $0$ for point $1$ $(0,0)$ and $2$ for point $2$ $(0,2)$, and sum will be odd for all black point with same reason.

## III. Coding

The way of implementing Red-black ordering as follow:

```
1      #pragma omp parallel
2       {
3          #pragma omp for
4          for (int i = 0; i < NY; i++) { /* rows */
5              for (int j = 0; j < NX; j++) { /* columns */
6                  if ((i+j)%2 == 0){
7                      do something
8                      ...
9          #pragma omp for
10         for (int i = 0; i < NY; i++) { /* rows */
11             for (int j = 0; j < NX; j++) { /* columns */
12                 if ((i+j)%2 != 0){
13                     do something
14                     ...
```

the main part is line 6 and 12, which using sum of row and columns divide by 2 if remainder is $0$, it is even number belongs to red group and if not $0$ it is odd number belongs black group, and each group is under different parallel region so data races will not happen.

## IV. Thinking and improvement

As it was nested for loops, I try to find nested way to parallel both $i$ and $j$ loop but failed when I used $\#pragma\ omp\ for$ inside $\#pragma\ omp\ sections$. I believe it can be done some how, and It need more work on it.