Code Parallelization

m.cestari@cineca.it

Casalecchio di Reno, 17/06/2015

24th Summer School on PARALLEL COMPUTING

Code Parallelization

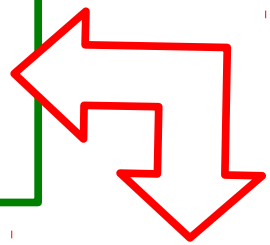
two **stages** to write a parallel code

- **problem domain**
  - algorithm
- **program domain**
  - implementation

Problem domain

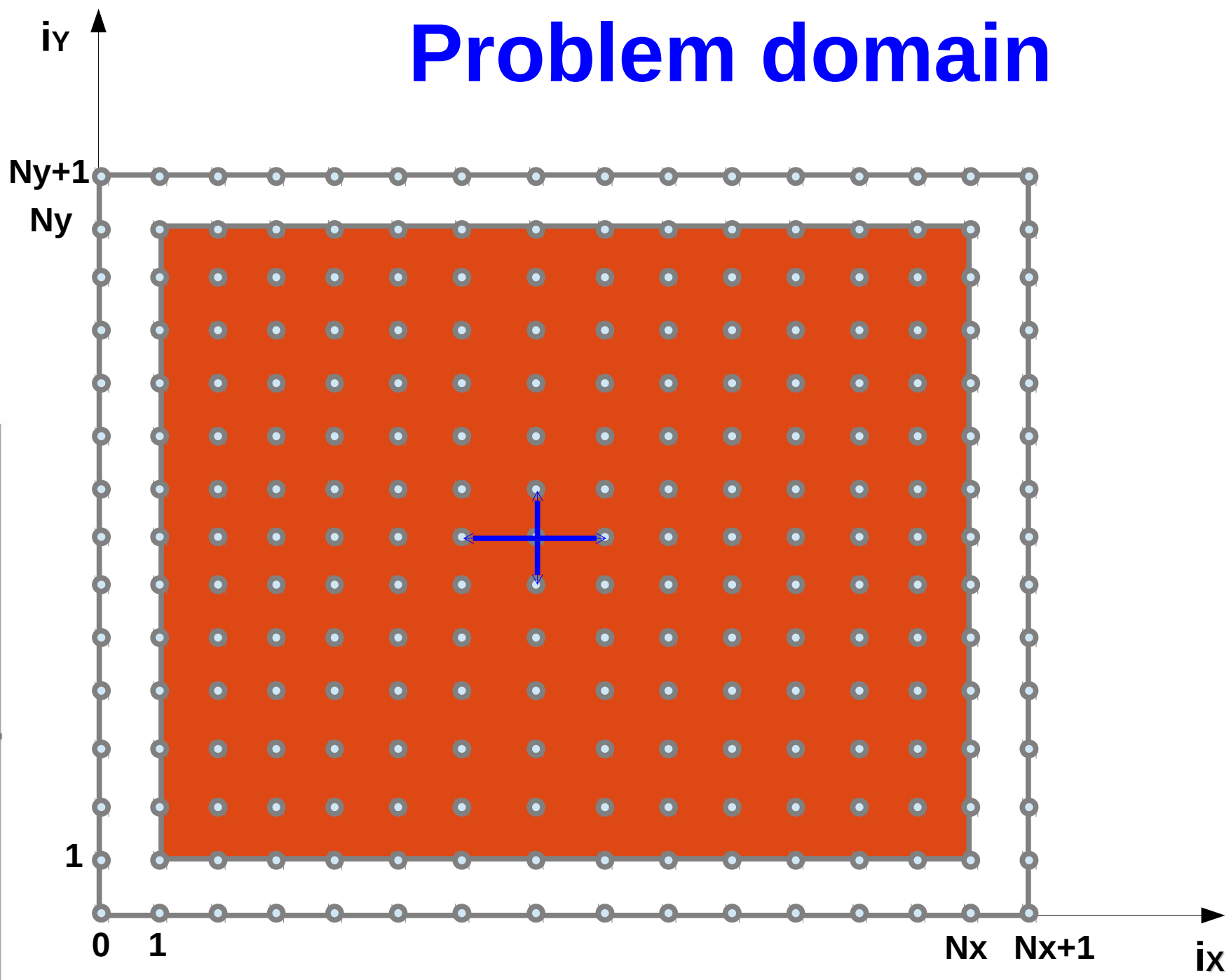
- Understanding the problem
- Identifying the most **computationally demanding** parts of the problem

```
for(i=1;i<=500;++i) {
        evolve(0.1, temp, temp_new);
        update_boundaries_FLAT(temp);
 }
```



```
do iy=1,NY
  do ix=1,NX
    temp0 = temp(ix,iy)
    temp_new(ix,iy) = (temp(ix+1,iy)-temp(ix-1,iy)+temp(ix,iy+1)-temp(ix,iy-1)) / dx
  enddo
enddo
```

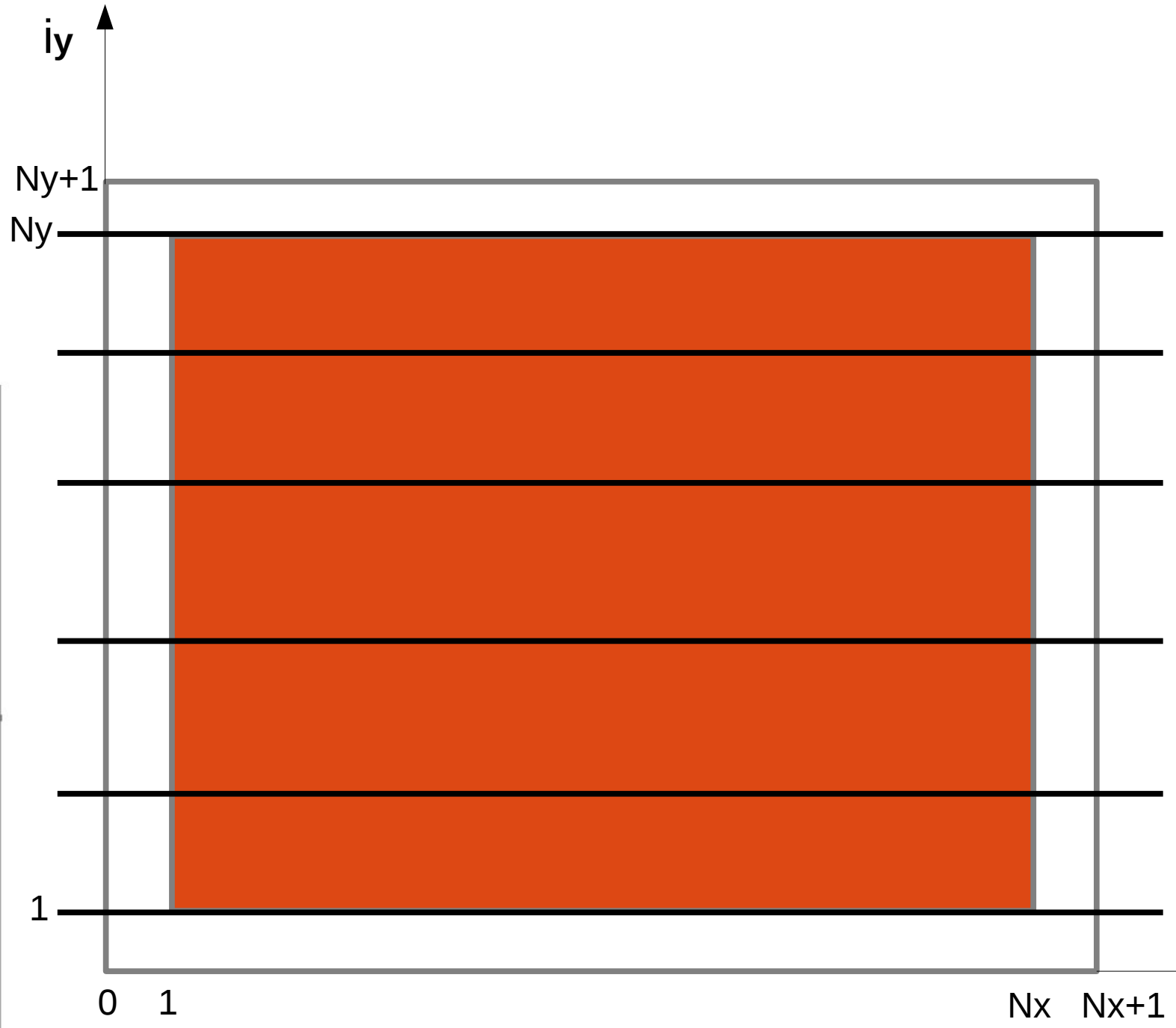
Problem domain



Concurrency

Find concurrency:

- **similar** operations that can be applied to **different parts** of the data structure
- domain **decomposition**: divide data into chunks that can be operated concurrently
  - a task works only **its chunk** of data
  - map **local** to **global** variables



Let's assume the following domain decomposition

Fortran

mat(?,?)

C

Mat(?)

Dependencies

Handle dependencies among tasks:

- Tasks needs access to some portion of another task local data (**data sharing**)

Code Parallelization

2 different **stages** to parallelize a serial code

- **problem domain**
  - algorithm
- **program domain**
  - Implementation **(the fun part)**