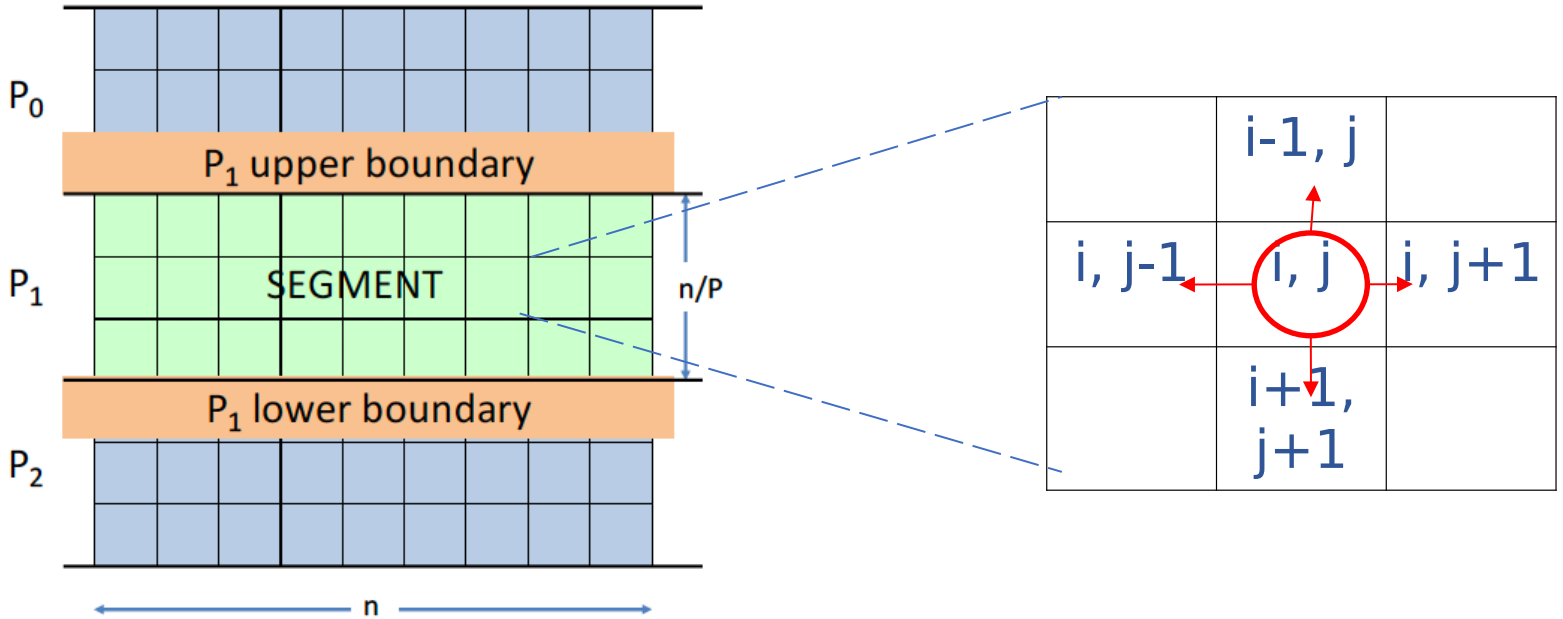
Stencil computation using Jacobi: Row-wise Distribution of the Matrices

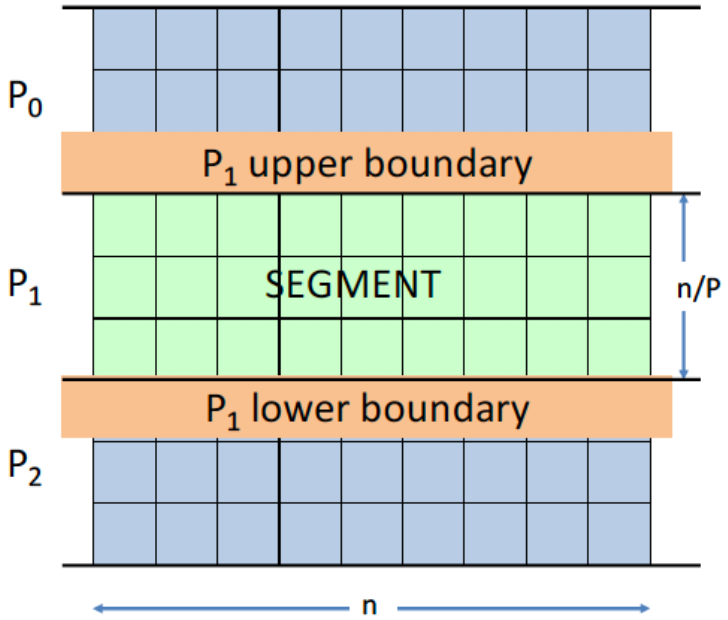
```
void compute(int n, double *u, double *utmp) {
int i, j;
double tmp;

for (i = 1; i < n-1; i++) {
    for (j = 1; j < n-1; j++) {
        tmp = u[n*(i+1) + j] + u[n*(i-1) + j] +  // elements u[i+1][j] and u[i-1][j]
              u[n*i + (j+1)] + u[n*i + (j-1)] -  // elements u[i][j+1] and u[i][j-1]
              4 * u[n*i + j];                    // element u[i][j]
        utmp[n*i + j] = tmp/4;                   // element utmp[i][j]
    }
}}
```

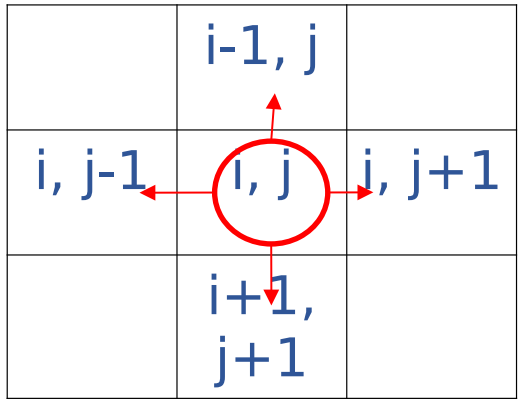


Computation of each element requires access for reading four neighbors (upper, lower, left, right) of matrix u

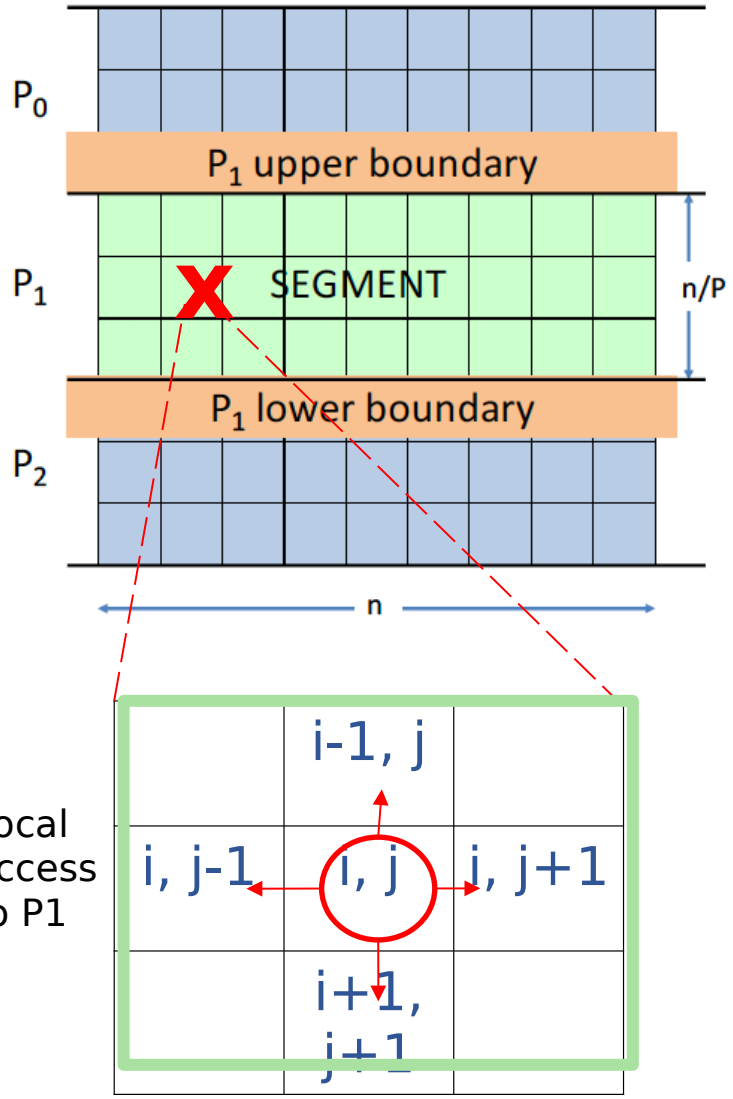
Stencil computation using Jacobi: Row-wise Distribution of the Matrices

Different memory access situations can arise depending on where the element appears:





Stencil computation using Jacobi: Row-wise Distribution of the Matrices

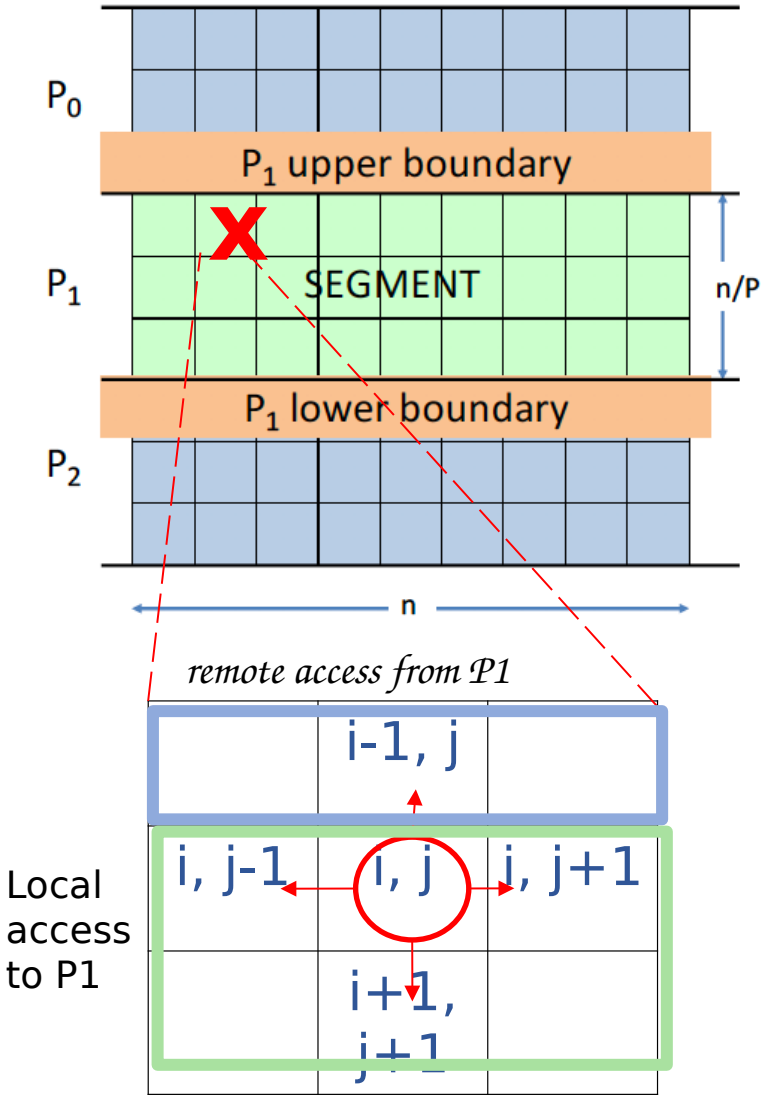


Different memory access situations can arise depending on where the element appears:

1. ***The element is in a row that is not a border:***

   *All 4 neighbors are **local** to the current processor (P1 in the example): local access to data, we assume **zero overhead** in our data sharing model*

Stencil computation using Jacobi: Row-wise Distribution of the Matrices



Different memory access situations can arise depending on where the element appears:

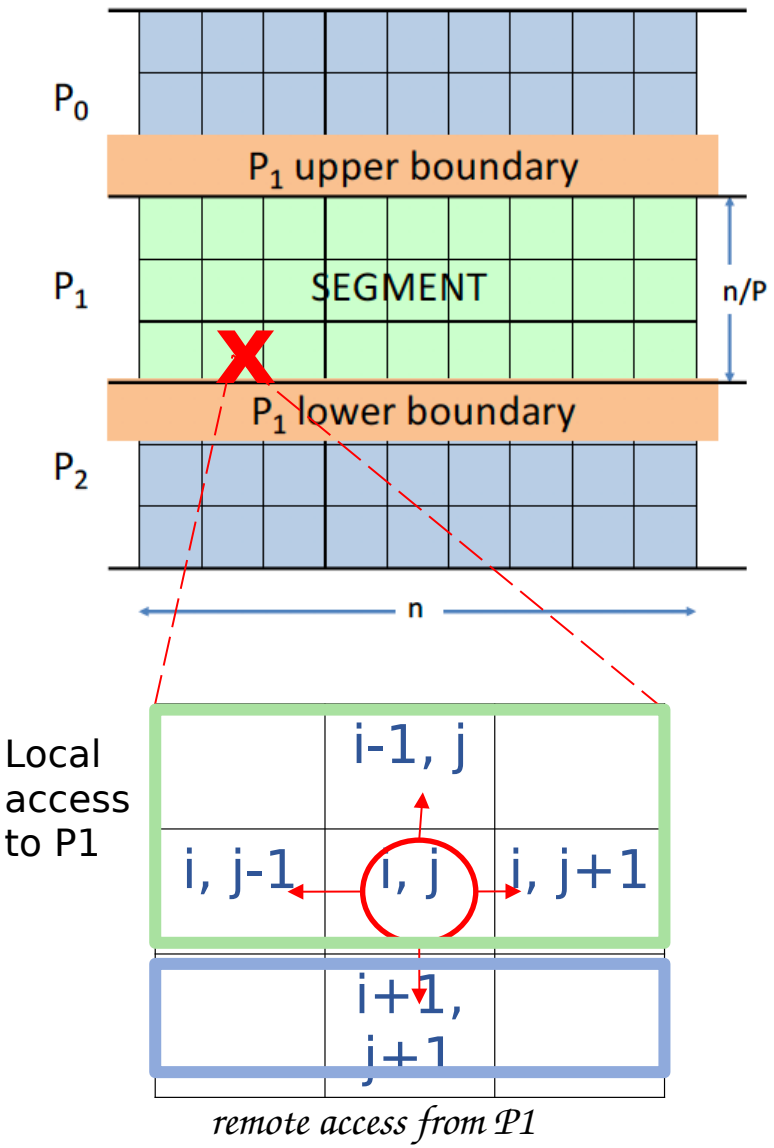
1. ***The element is in a row that is not a border:***

   *All 4 neighbors are **local** to the current processor (P1 in the example): local access to data, we assume **zero overhead** in our data sharing model*

2. ***The element is situated in a row in the upper border:***

   *3 neighbors (left, right, lower) local to the current processor (P1 in the example) and 1 neighbor (**upper**) with **remote access** (to P0 in the example) with data sharing overhead*

Stencil computation using Jacobi: Row-wise Distribution of the Matrices



Different memory access situations can arise depending on where the element appears:

1. ***The element is in a row that is not a border:***

   *All 4 neighbors are **local** to the current processor (P1 in the example): local access to data, we assume **zero overhead** in our data sharing model*

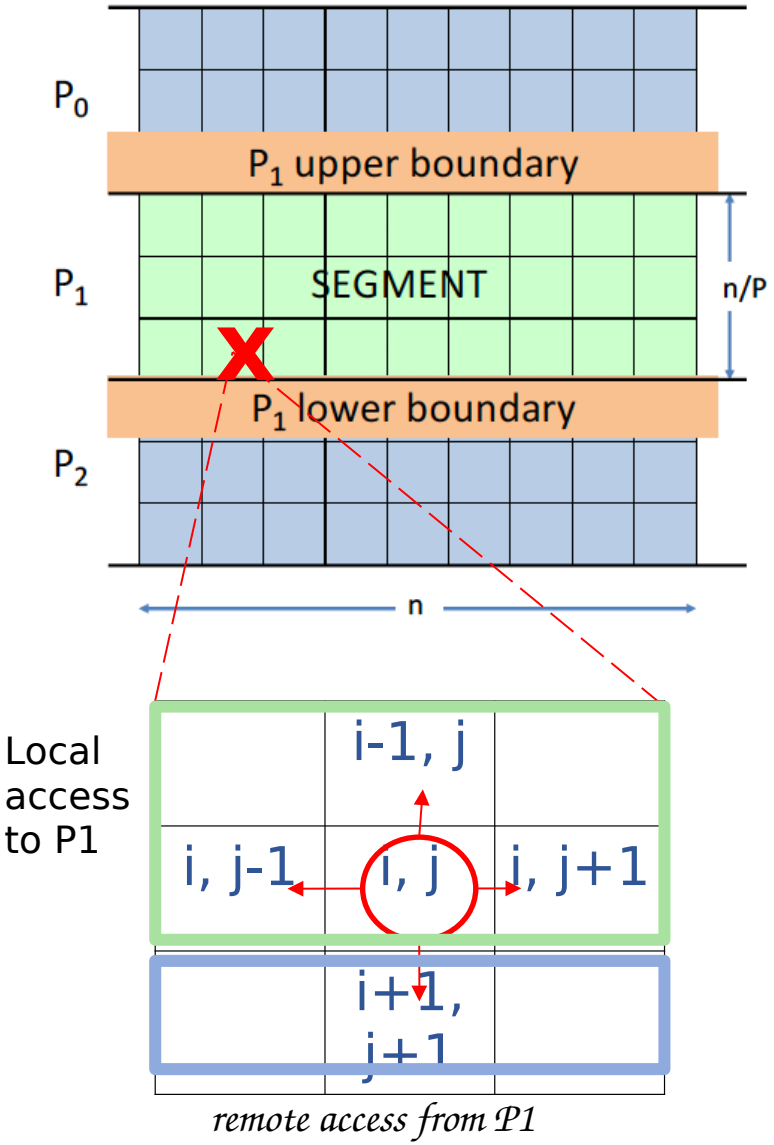
2. ***The element is situated in a row in the upper border:***

   *3 neighbors (left, right, lower) local to the current processor (P1 in the example) and 1 neighbor (**upper**) with **remote access** (to P0 in the example) with **data sharing overhead***

3. ***The element is situated in a row in the lower border:***

   *3 neighbors (left, right, upper) local to the current processor (P1 in the example) and 1 neighbor (**lower**) with **remote access** (to P2 in the example) with **data sharing overhead***

Stencil computation using Jacobi: Row-wise Distribution of the Matrices



Summarizing, there is remote memory access when processing elements in the rows:

- ***in the upper border of a segment***
- ***in the lower border of a segment***