

## Παρουσίαση 4ης Άσκησης

*Παραλληλοποίηση και βελτιστοποίηση αλγορίθμων  
σε αρχιτεκτονικές κατανεμημένης μνήμης*

Συστήματα Παράλληλης Επεξεργασίας  
9ο Εξάμηνο, ΣΗΜΜΥ



*Εργ. Υπολογιστικών Συστημάτων  
Σχολή ΗΜΜΥ, Ε.Μ.Π.*



# Αναδρομή στις Μερικές Διαφορικές Εξισώσεις

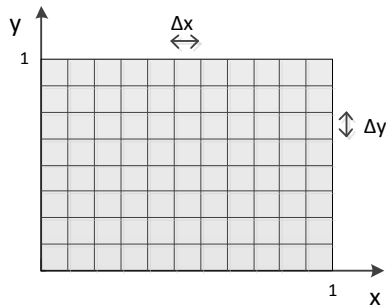
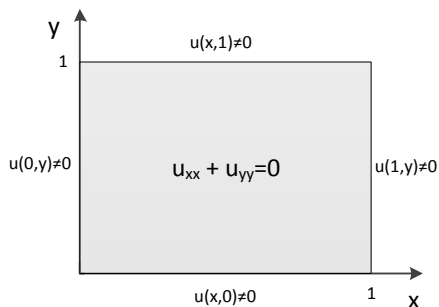
- **Εξίσωση Poisson:**  $\nabla^2 u = f$  σε χωρίο  $\Omega$ 
  - ▶ Γνωστή και ως *κυματική εξίσωση*
- **Σε καρτεσιανές συντεταγμένες (2Δ):**  $(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})u(x, y) = f(x, y)$
- **Εξίσωση Laplace:**  $(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})u(x, y) = 0$ 
  - ▶ Γνωστή και ως *εξίσωση θερμότητας*
  - ▶ Ειδική περίπτωση της Poisson:  $f=0$

## Διάδοση θερμότητας σε δύο διαστάσεις

- Απλή εφαρμογή - Διάδοση θερμότητας σε επιφάνεια από το σύνορο στο εσωτερικό



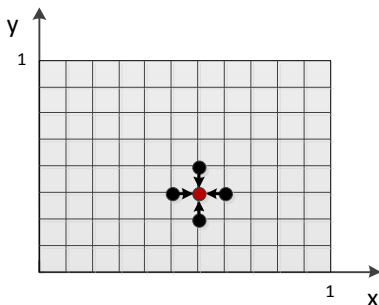
- Επίλυση εξίσωσης θερμότητας με συνοριακές συνθήκες



# Υπολογιστικές μέθοδοι (I) - Μέθοδος Jacobi

## Επαναληπτική μέθοδος Jacobi

$$u_{x,y}^{t+1} = \frac{u_{x-1,y}^t + u_{x,y-1}^t + u_{x+1,y}^t + u_{x,y+1}^t}{4}$$

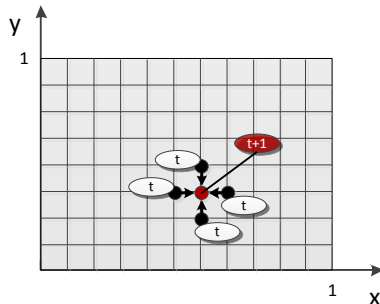
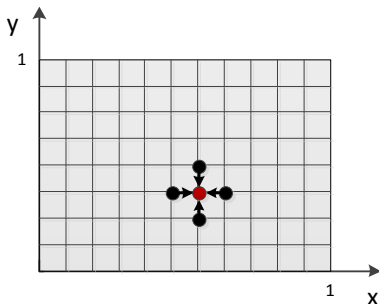


- 5-point stencil
- Κάθε στοιχείο υπολογίζεται από τα γειτονικά του

# Υπολογιστικές μέθοδοι (I) - Μέθοδος Jacobi

## Επαναληπτική μέθοδος Jacobi

$$u_{x,y}^{t+1} = \frac{u_{x-1,y}^t + u_{x,y-1}^t + u_{x+1,y}^t + u_{x,y+1}^t}{4}$$



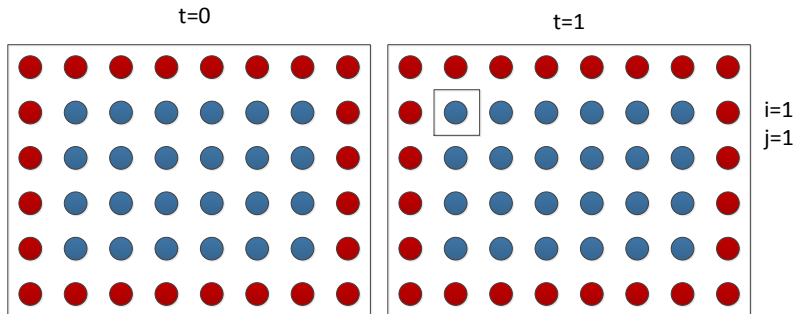
# Υπολογιστικές μέθοδοι (I) - Μέθοδος Jacobi

## Αλγόριθμος Jacobi

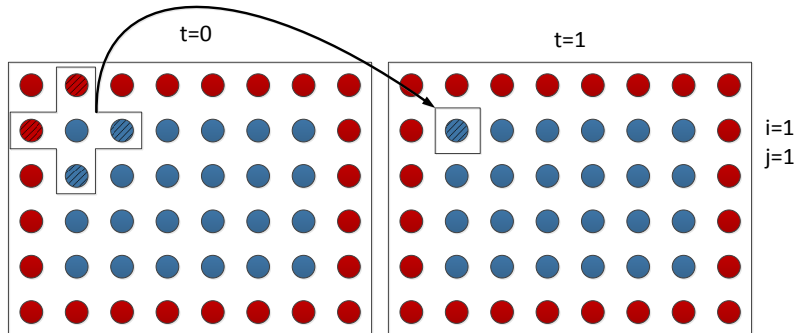
```
for  $t = 0; t < T \&\& !converged; t++$  do
  for  $i = 1; i < X - 1; i++$  do
    for  $j = 1; j < Y - 1; j++$  do
       $u[t + 1][i][j] =$ 
         $(u[t][i - 1][j] + u[t][i][j - 1] + u[t][i + 1][j] + u[t][i][j + 1])/4$ 
    end
  end
   $converged = checkConvergence(u[t], u[t + 1])$ 
end
```

- Για τον υπολογισμό των τιμών της χρονικής στιγμής  $t+1$ , χρησιμοποιούνται οι τιμές της χρονικής στιγμής  $t$ 
  - ▶ Απαιτούνται τουλάχιστον δύο πίνακες, ένας για την τρέχουσα ( $t+1$ ) κι ένας για την προηγούμενη ( $t$ ) χρονική στιγμή
- Στο τέλος κάθε βήματος, πραγματοποιείται έλεγχος σύγκλισης

# Εκτέλεση Jacobi (I)

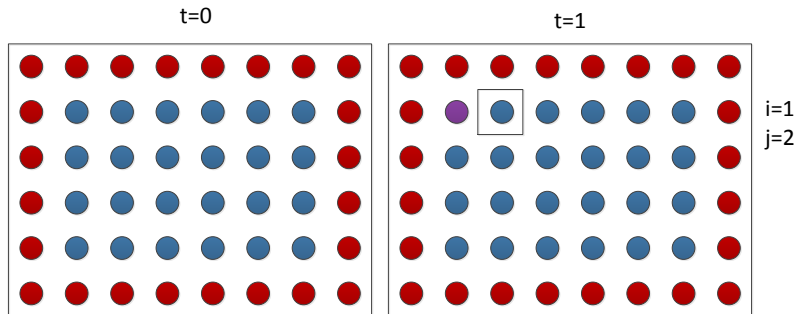


## Εκτέλεση Jacobi (II)

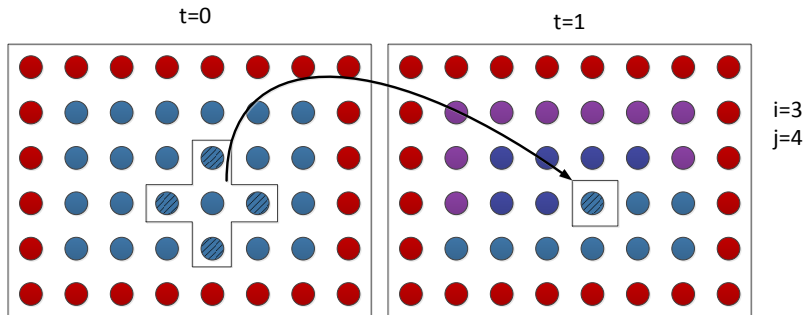




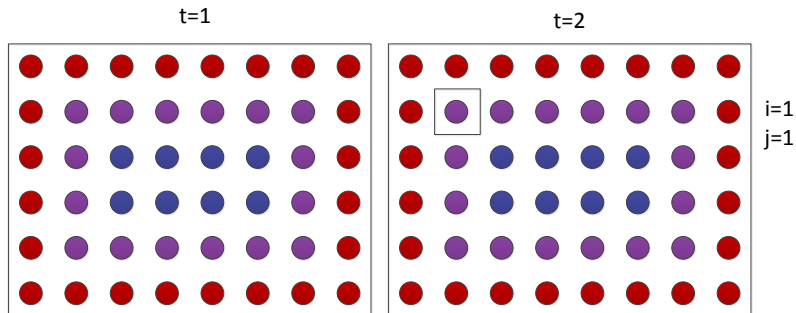
## Εκτέλεση Jacobi (III)



## Εκτέλεση Jacobi (IV)

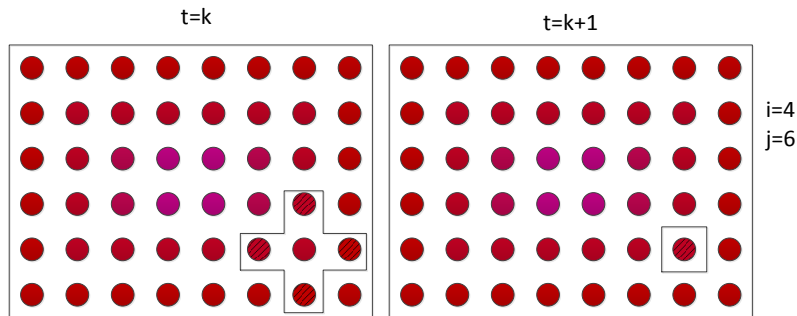


## Εκτέλεση Jacobi (V)



# Εκτέλεση Jacobi (VI)

Σύγκλιση

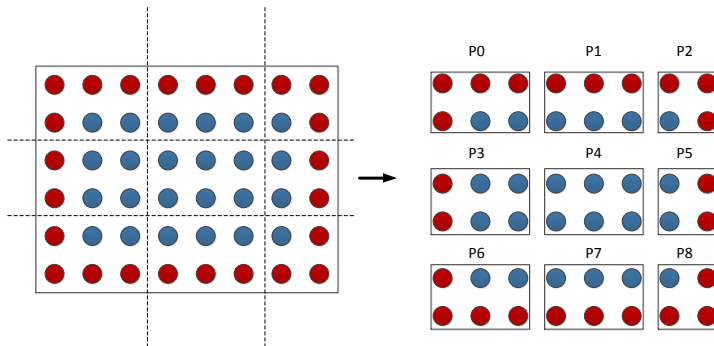


- Τα στοιχεία του πίνακα δε διαφέρουν (ή διαφέρουν ελάχιστα) στις χρονικές στιγμές  $t$  και  $t+1$
- Ο αλγόριθμος έχει συγκλίνει

# Ζητήματα Υλοποίησης - MPI

## Διαμοιρασμός πίνακα

- Θεωρούμε δισδιάστατο πλέγμα επεξεργαστών

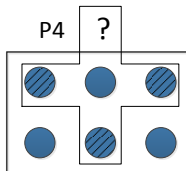


- Πώς διαμοιράζουμε τον πίνακα στο MPI; Ποιες συναρτήσεις χρησιμοποιούμε;

# Ζητήματα Υλοποίησης - MPI

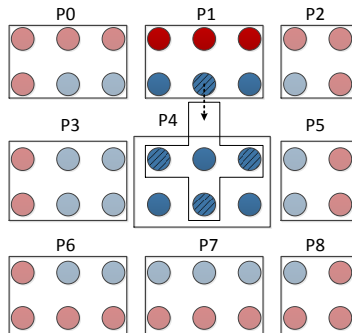
Υπολογιστικός πυρήνας

- Έχει κάθε διεργασία τα στοιχεία που χρειάζεται;



# Ζητήματα Υλοποίησης - MPI

Υπολογιστικός πυρήνας



- Ποια στοιχεία πρέπει να ανταλλάσσουν οι επεξεργαστές;
- Πότε γίνεται η ανταλλαγή δεδομένων;
- Ποιες συναρτήσεις MPI χρησιμοποιούμε για την επικοινωνία;
- Πού αποθηκεύονται τα δεδομένα αυτά;

# Ζητήματα Υλοποίησης - MPI

## Έλεγχος σύγκλισης

### Έλεγχος σύγκλισης

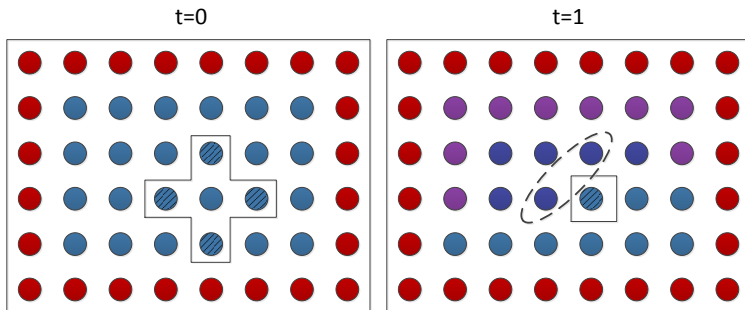
```
for  $i = 1; i < X - 1; i++$  do  
  for  $j = 1; j < Y - 1; j++$  do  
    if  $|u[t + 1][i][j] - u[t][i][j]| > \varepsilon$  then  
      return 0  
    end  
  end  
end  
return 1;
```

- Ο έλεγχος σύγκλισης γίνεται τοπικά σε κάθε διεργασία.
- Πώς θα ελέγξουμε τη σύγκλιση σε όλο το χωρίο;



## Υπολογιστικές μέθοδοι (II) - Μέθοδος Gauss-Seidel

- Η μέθοδος Jacobi λύνει το πρόβλημα, αλλά έχει πολύ αργό ρυθμό σύγκλισης
- **Παρατήρηση 1:** Όταν υπολογίζω το σημείο  $u[t+1][i][j]$ , έχω ήδη υπολογίσει τα σημεία  $u[t+1][i-1][j]$ ,  $u[t+1][i][j-1]$

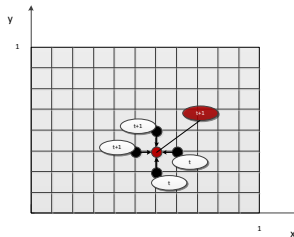
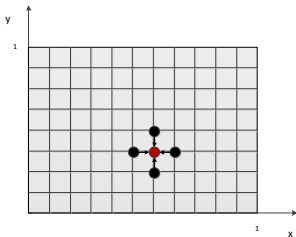


## Υπολογιστικές μέθοδοι (II) - Μέθοδος Gauss-Seidel

- Η μέθοδος Jacobi λύνει το πρόβλημα, αλλά έχει πολύ αργό ρυθμό σύγκλισης
- **Παρατήρηση 2:** Τα σημεία  $u[t+1][i-1][j]$ ,  $u[t+1][i][j-1]$  έχουν updated τιμές σε σχέση με τα  $u[t][i-1][j]$ ,  $u[t][i][j-1]$

### Επαναληπτική μέθοδος Gauss-Seidel

$$u_{x,y}^{t+1} = \frac{u_{x-1,y}^{t+1} + u_{x,y-1}^{t+1} + u_{x+1,y}^t + u_{x,y+1}^t}{4}$$



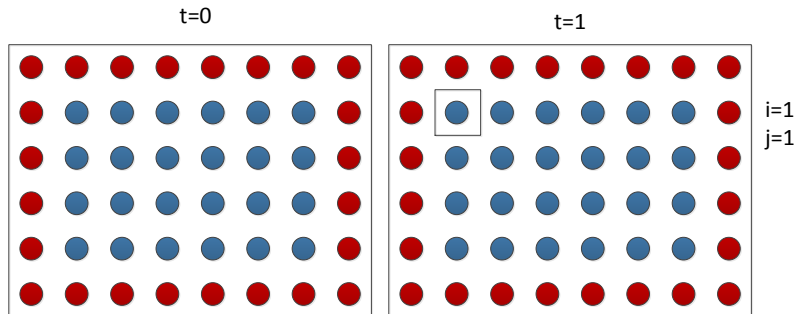
## Υπολογιστικές μέθοδοι (II) - Μέθοδος Gauss-Seidel με SOR

- Successive Over-Relaxation (SOR): Ένας τρόπος για ταχύτερη σύγκλιση
- Αρχική επαναληπτική μέθοδος:  $u^{t+1} = f(u^t)$
- Επαναληπτική μέθοδος με SOR:  $u^{t+1} = (1 - \omega)u^t + \omega f(u^t)$

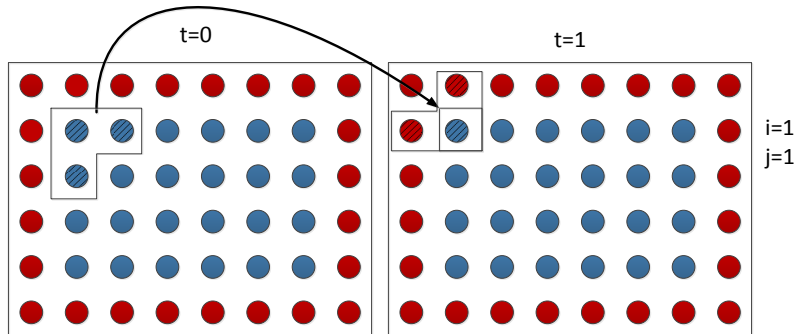
### Επαναληπτική μέθοδος Gauss-Seidel-SOR

$$u_{x,y}^{t+1} = u_{x,y}^t + \omega \frac{u_{x-1,y}^{t+1} + u_{x,y-1}^{t+1} + u_{x+1,y}^t + u_{x,y+1}^t - 4u_{x,y}^t}{4}, \omega \in (0, 2)$$

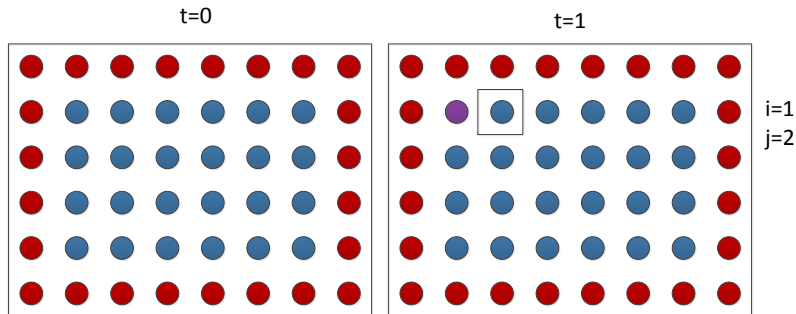
# Εκτέλεση Gauss-Seidel-SOR (I)



## Εκτέλεση Gauss-Seidel-SOR (II)



## Εκτέλεση Gauss-Seidel-SOR (III)



# Ζητήματα Υλοποίησης - MPI

## Υπολογιστικός πυρήνας

- Τι αλλάζει στην επικοινωνία της Gauss-Seidel σε σχέση με τη Jacobi;
- Πότε γίνεται η ανταλλαγή των δεδομένων μεταξύ των διεργασιών;
  - ▶ Για τον υπολογισμό κάθε στοιχείου, απαιτούνται στοιχεία από την προηγούμενη και την τρέχουσα χρονική στιγμή
- Ποιος υπολογιστικός πυρήνας θα είναι πιο γρήγορος, αυτός της Jacobi ή αυτός της Gauss-Seidel; Γιατί;

## Υπολογιστικές μέθοδοι (III) - Μέθοδος Red-Black SOR

- Το Red-Black ordering επιτυγχάνει καλύτερο ρυθμό σύγκλισης
- Red-Black ordering:
  1. Στοιχεία σε άρτιες θέσεις  $(i + j) \% 2 = 0 \rightarrow \text{Red}$
  2. Στοιχεία σε περιττές θέσεις  $(i + j) \% 2 = 1 \rightarrow \text{Black}$
- Υπολογισμός σε δύο φάσεις



# Υπολογιστικές μέθοδοι (III) - Μέθοδος Red-Black SOR

## Red-Black SOR - Φάση 1η - Update Red

$$u_{x,y}^{t+1} = u_{x,y}^t + \omega \frac{u_{x-1,y}^t + u_{x,y-1}^t + u_{x+1,y}^t + u_{x,y+1}^t - 4u_{x,y}^t}{4}, \text{ when } (x+y)\%2 == 0$$

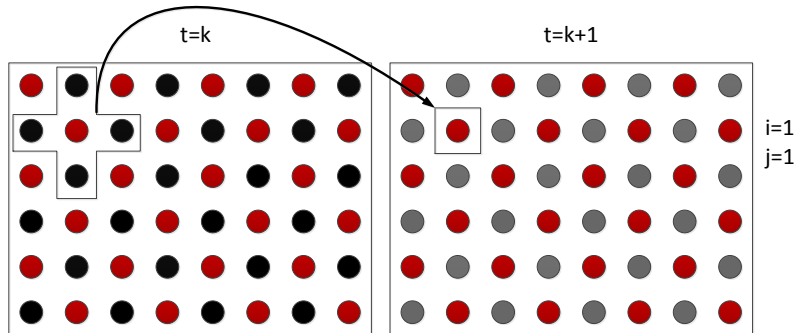
- Στη red φάση υπολογίζονται τα *red* στοιχεία από τα *black*

## Red-Black SOR - Φάση 2η - Update Black

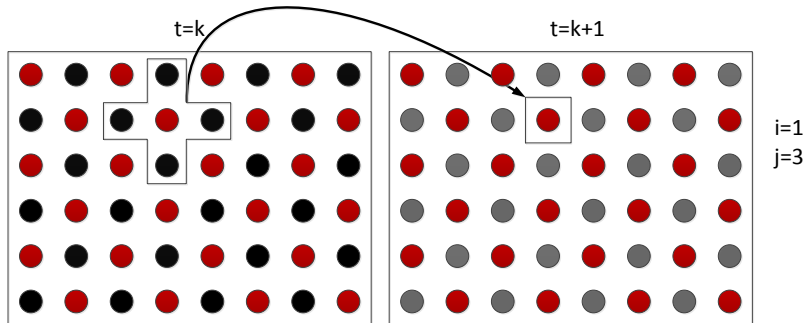
$$u_{x,y}^{t+1} = u_{x,y}^t + \omega \frac{u_{x-1,y}^{t+1} + u_{x,y-1}^{t+1} + u_{x+1,y}^{t+1} + u_{x,y+1}^{t+1} - 4u_{x,y}^t}{4}, \text{ when } (x+y)\%2 == 1$$

- Στη black φάση υπολογίζονται τα *black* στοιχεία από τα *red*

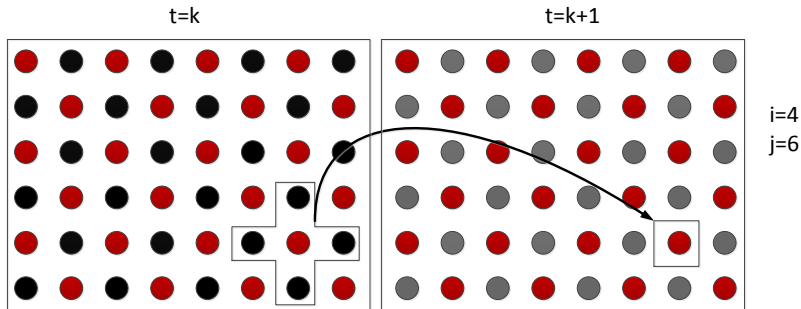
## Εκτέλεση Red-Black-SOR - 1η φάση (I)



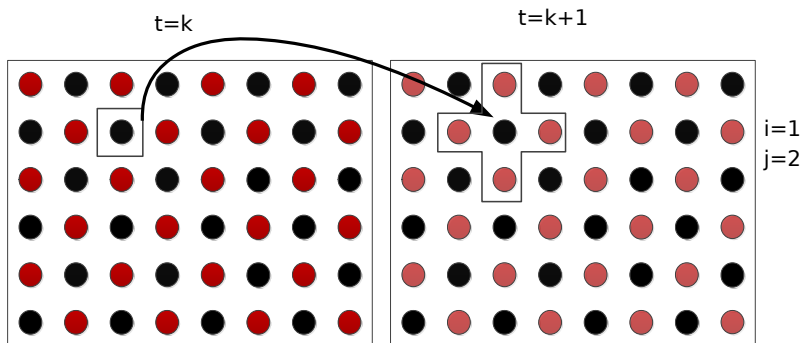
## Εκτέλεση Red-Black-SOR - 1η φάση (II)



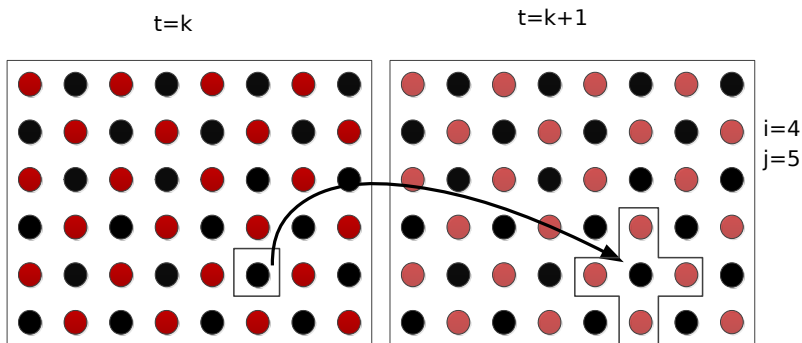
## Εκτέλεση Red-Black-SOR - 1η φάση (III)



## Εκτέλεση Red-Black-SOR - 2η φάση (I)



## Εκτέλεση Red-Black-SOR - 2η φάση (II)



# Ζητήματα Υλοποίησης - MPI

## Υπολογιστικός πυρήνας

- Ο υπολογιστικός πυρήνας της Red-Black SOR περιλαμβάνει δύο φάσεις. Πώς αλλάζει αυτό την επικοινωνία; Τι όφελος/κόστος μπορεί να έχει;

# Ζητούμενα

**Υποχρεωτικά** για τη μέθοδο *Jacobi*

**Προαιρετικά** για τις μεθόδους *Gauss-Seidel SOR* και *Red-Black SOR*

- Αναπτύξτε παράλληλα προγράμματα στο μοντέλο ανταλλαγής μηνυμάτων με τη βοήθεια της βιβλιοθήκης MPI
  - ▶ Σειριακές υλοποιήσεις στα αρχεία *Jacobi\_serial.c*, *GaussSeidelSOR\_serial.c* και *RedBlackSOR\_serial.c*
  - ▶ Σκελετός υλοποίησης σε MPI στο αρχείο *mpi\_skeleton.c*
- Πραγματοποιήστε μετρήσεις επίδοσης με βάση το σενάριο μετρήσεων
- Συγκεντρώστε τα αποτελέσματα, τις συγκρίσεις και τα σχόλιά σας στην τελική αναφορά