Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. και Μηχανικών Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Παρουσίαση 2ης Άσκησης:

Ανάπτυξη παράλληλου κώδικα και μελέτη της επίδοσης του αλγορίθμου LU Decomposition σε μοντέρνες πολυπύρηνες αρχιτεκτονικές

Συστήματα Παράλληλης Επεξεργασίας

9ο Εξάμηνο

National Technical University of Athens
CSLab

Σκοπός άσκησης

- Εξοικείωση με παράλληλα προγραμματιστικά μοντέλα / εργαλεία για πολυπύρηνες αρχιτεκτονικές
- Υλοποίηση / βελτιστοποίηση παράλληλων προγραμμάτων
- Μελέτη επίδοσης
- Αξιολόγηση προγραμματιστικής ευκολίας (productivity /programmability)
- Όπως στην Άσκηση 1, δουλεύουμε με τον αλγόριθμο LU Decomposition

Αλγόριθμος LU Decomposition

- Εύρεση ενός κάτω τριγωνικού πίνακα L και ενός άνω τριγωνικού πίνακα U ώστε A=LU

```
for (k=0;k<N-1;k++) {
        for (i=k+1;i<N;i++) {
                l=A[i][k]=A[i][k]/A[k][k];
                for (j=k+1;j<N;j++)
                        A[i][j]=A[i][j]-l*A[k][j];
        }
}
```

- A: NxN πίνακας προς παραγοντοποίηση
- Οι πίνακες L και U αποθηκεύονται in place στον A
- Πολυπλοκότητα: $O(n^3)$

LU Decomposition

A00	A01	A02
A10	A11	A12
A20	A21	A22

=

1	0	0
L10	1	0
L20	L21	1

*

U00	U01	U02
0	U11	U12
0	0	U22

- Τελικά ο Α γίνεται….

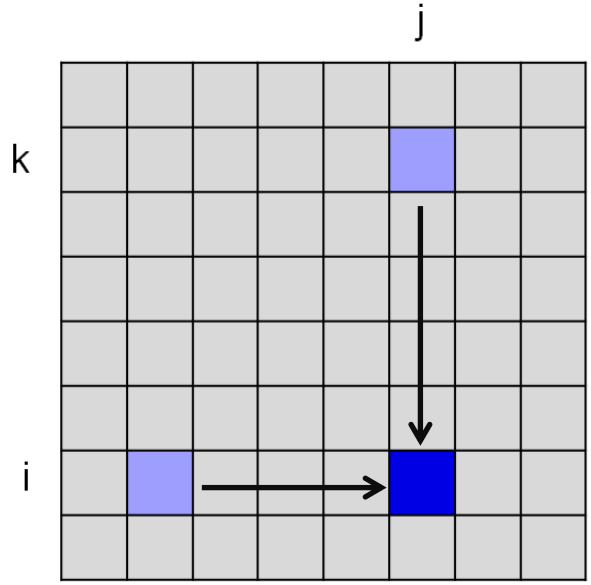
A00	A01	A02
A10	A11	A12
A20	A21	A22

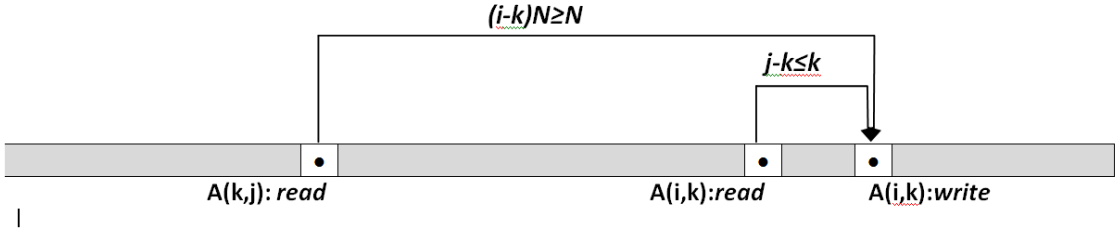
→

U00	U01	U02
L10	U11	U12
L20	U21	U22

Πρόσβαση στη μνήμη…

- Για μεγάλα N, ο πίνακας δε χωράει στην cache
  - **Memory-bound** αλγόριθμος
  - Μη ευνοϊκό access pattern για data reuse
- Ο πίνακας πρέπει να μεταφέρεται από την κύρια μνήμη σε κάθε επανάληψη
  - Ο αλγόριθμος δεν κλιμακώνει σε αρχιτεκτονικές κοινής μνήμης





Εναλλακτικές Υλοποιήσεις: Recursive LU Decomposition

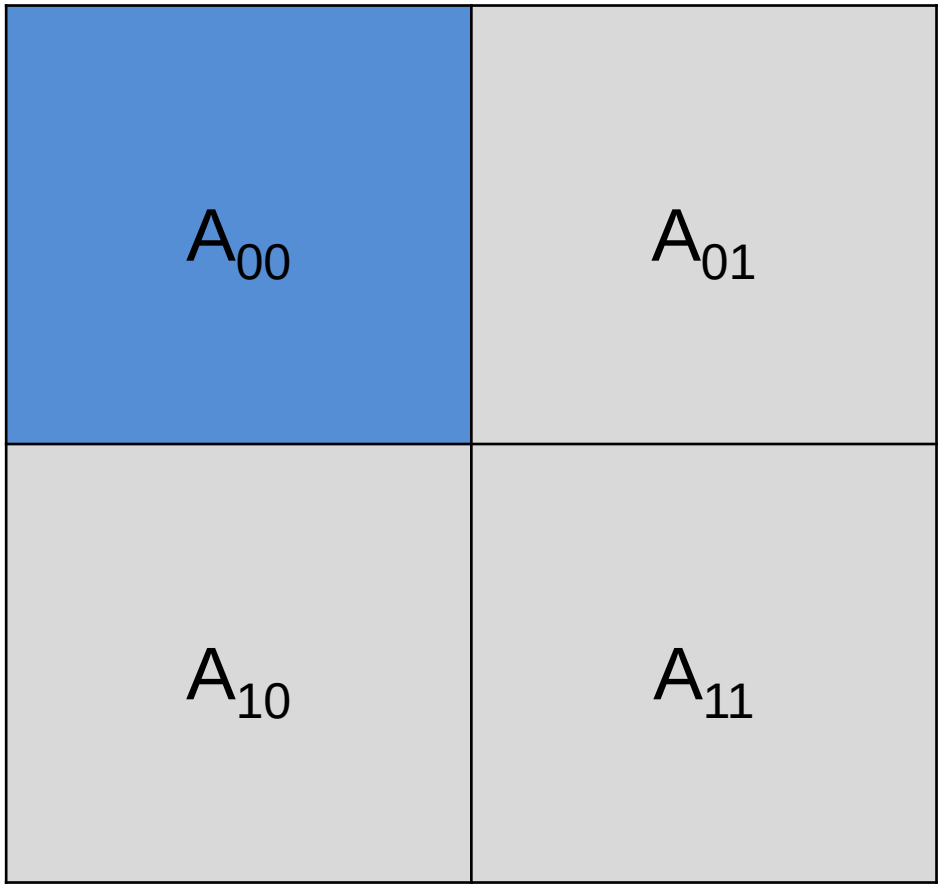
lu_recursive(A)
- if (base case)
  - lu_kernel(A)
- else
  - lu_recursive(A_{00})
  - lower_solve(A_{01},A_{00})
  - upper_solve(A_{10},A_{00})
  - schur(A_{11},A_{10},A_{01})
  - lu_recursive(A_{11})

Forward Substitution: $L_{00}\, A_{01}' = A_{01} \rightarrow A_{01}'$

Forward Substitution: $A_{10}' U_{00} = A_{10} \rightarrow A_{10}'$

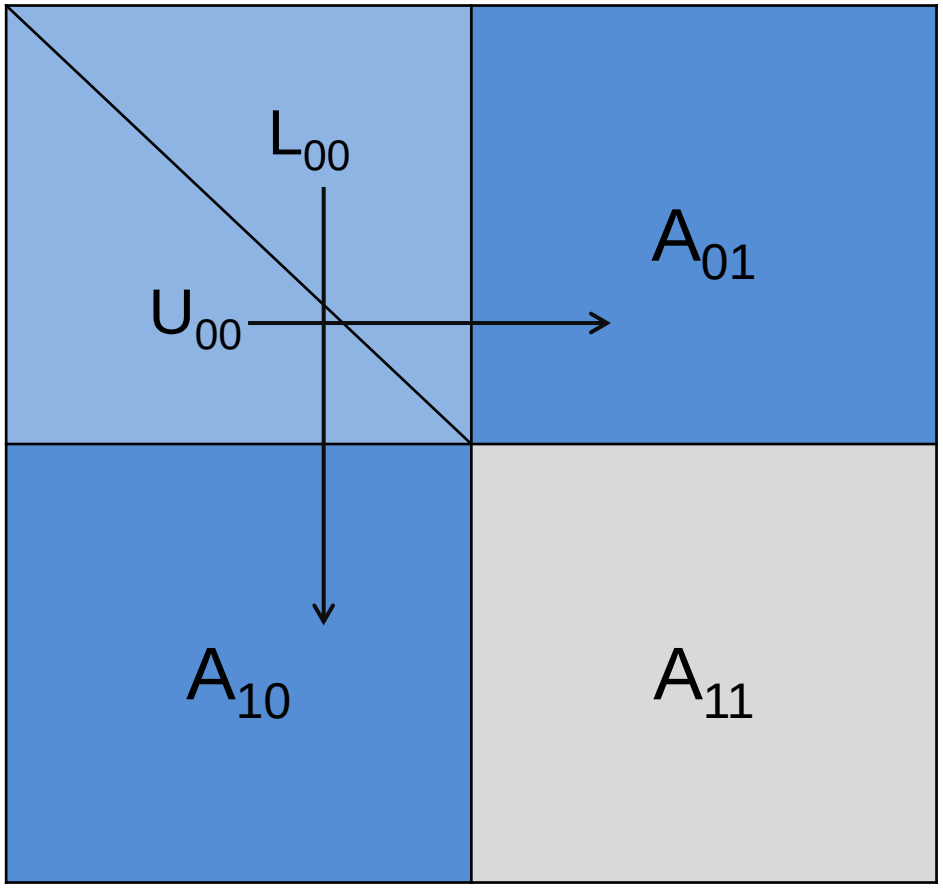
Matrix Multiplication: $A_{11}' = A_{11} - A_{10}'\, A_{01}'$

Εναλλακτικές Υλοποιήσεις: Recursive LU Decomposition



lu_recursive(A_{00})

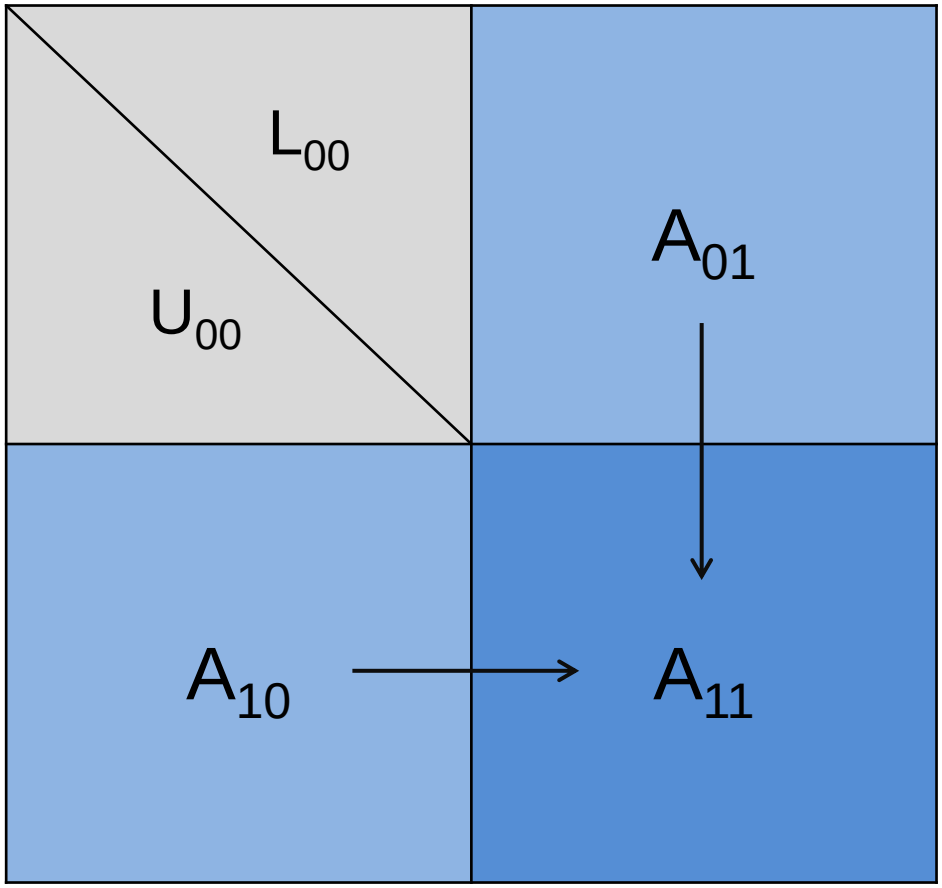
Εναλλακτικές Υλοποιήσεις: Recursive LU Decomposition



lu_recursive(A_{00})

lower_solve(A_{01},A_{00})
upper_solve(A_{10},A_{00})

Εναλλακτικές Υλοποιήσεις: Recursive LU Decomposition

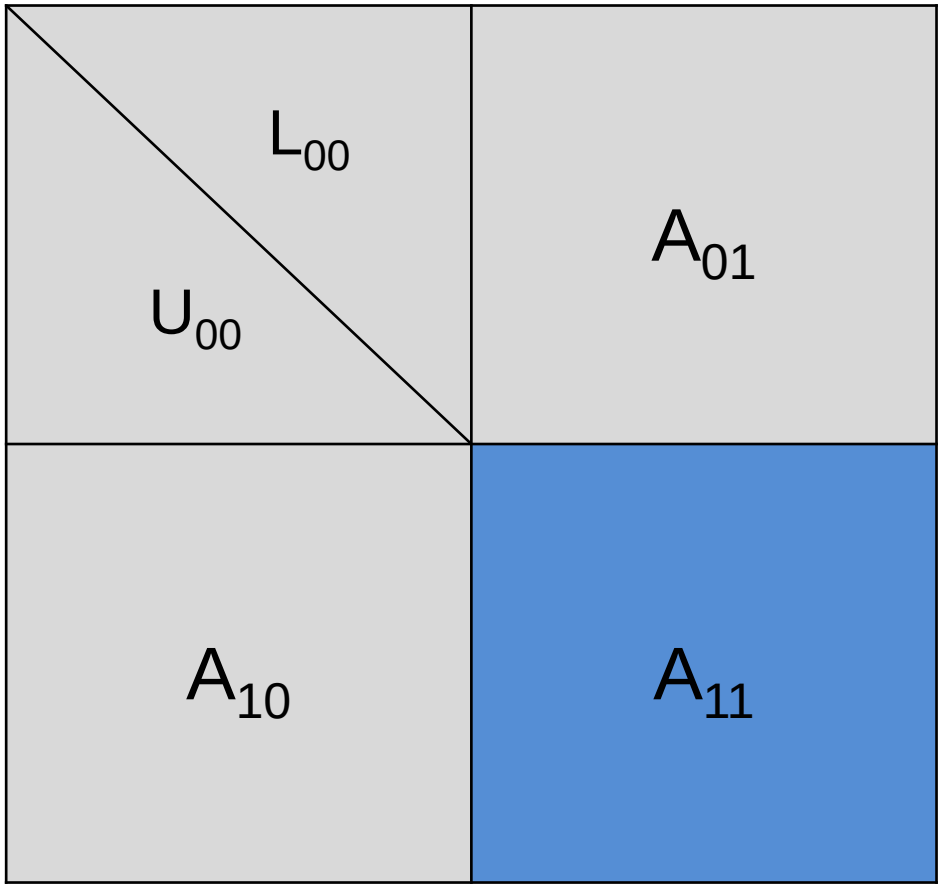


lu_recursive(A_{00})

lower_solve(A_{01},A_{00})
upper_solve(A_{10},A_{00})

schur(A_{11},A_{10},A_{01})

Εναλλακτικές Υλοποιήσεις: Recursive LU Decomposition



lu_recursive(A_{00})

lower_solve(A_{01},A_{00})
upper_solve(A_{10},A_{00})

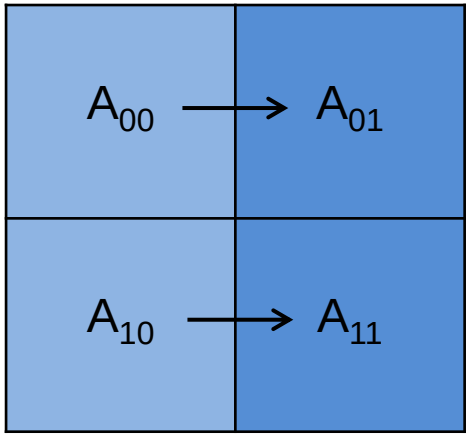
schur(A_{11},A_{10},A_{01})

lu_recursive(A_{11})

Εναλλακτικές Υλοποιήσεις: Recursive LU Decomposition

- Περισσότερη αναδρομή…
- lower_solve(A,L) : αναδρομική επίλυση του συστήματος LA'=A, για την εύρεση του A'

```
lower_solve(A,L)
    if (base case)
        block_lower_solve(A,L)
    else
        aux_lower_solve(A00,A10,L)
        aux_lower_solve(A01,A11,L)
```



- Δεν υπάρχει εξάρτηση στα δεδομένα των δύο κλήσεων της aux_lower_solve – παράλληλη εκτέλεση
- Tasks και για την aux_lower_solve!
- Ομοίως για την upper_solve

Εναλλακτικές Υλοποιήσεις: Recursive LU Decomposition

- Κι άλλη αναδρομή…
- schur(A,V,W) : A'=A-V*W

```
schur(A,V,W)
    if (base case)
        block_schur(A,V,W)
    else
        schur(A00,V00,W00)
        schur(A01,V00,W01)
        schur(A10,V10,W00)
        schur(A11,V10,W01)

        schur(A00,V01,W10)
        schur(A01,V01,W11)
        schur(A10,V11,W10)
        schur(A11,V11,W11)
```

Πίνακας εγγραφής : A

schur(A_{00},V_{00},W_{00})
schur(A_{01},V_{00},W_{01})
schur(A_{10},V_{10},W_{00})
schur(A_{11},V_{10},W_{01})

schur(A_{00},V_{01},W_{10})
schur(A_{01},V_{01},W_{11})
schur(A_{10},V_{11},W_{10})
schur(A_{11},V_{11},W_{11})

Εναλλακτικές Υλοποιήσεις: Tiled LU decomposition

lu_tiled(A)	
range=N/B	
for (k=0;k<range-1;k++) {	
lu_kernel(A_{kk})	Upper diagonal tile - LU
l_inv=get_inv_l(A_{kk})	Υπολογισμός L_{kk}^{-1} , U_{kk}^{-1}
u_inv=get_inv_u(A_{kk})	
for (i=k+1;i<range;i++) {	
mm_lower(l_inv, A_{ki}, A_{ki})	$A_{ki}=L_{kk}^{-1}A_{ki}$ - upper horizontal frame
mm_upper(A_{ik}, u_inv, A_{ik})	$A_{ik}=A_{ik}\ U_{kk}^{-1}$ - left vertical frame
}	
for (i=k+1;i<range;i++)	
for (j=k+1;j<range;j++)	
mm(A_{ik},A_{ki},A_{ij})	Update trailing tiles – $A_{ij}=A_{ij}-A_{ik}A_{kj}$
}	
lu_kernel($A_{range-1,range-1}$)	LU decomposition-final diagonal tile

Εναλλακτικές Υλοποιήσεις: Tiled LU Decomposition

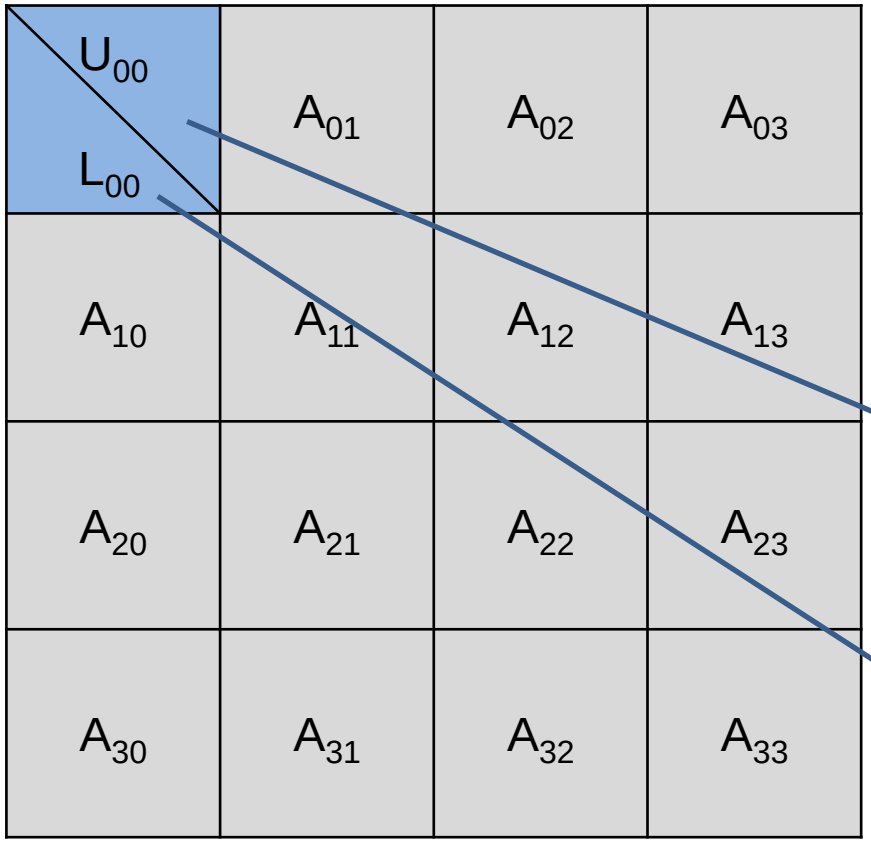
k=0, N, B, N/B=4

A_{00}	A_{01}	A_{02}	A_{03}
A_{10}	A_{11}	A_{12}	A_{13}
A_{20}	A_{21}	A_{22}	A_{23}
A_{30}	A_{31}	A_{32}	A_{33}

lu_kernel(A_{00})

Εναλλακτικές Υλοποιήσεις: Tiled LU Decomposition

k=0, N, B, N/B=4



lu_kernel(A_{00})

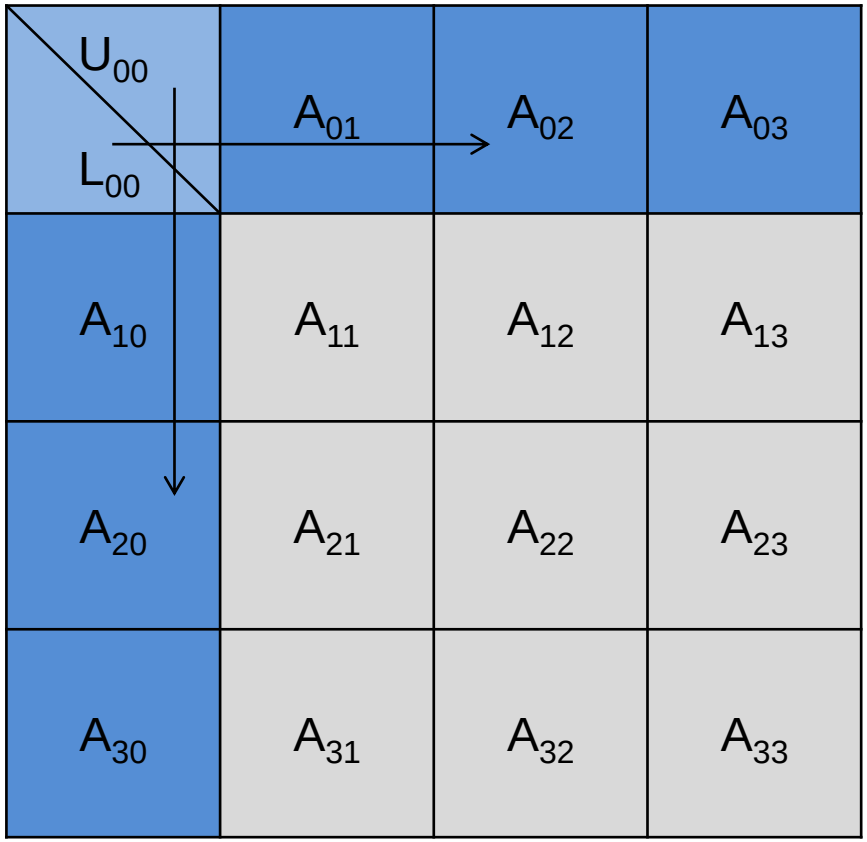
l_inv=get_inv_l(A_{00})
u_inv=get_inv_u(A_{00})

U_{00}^{-1}

L_{00}^{-1}

Εναλλακτικές Υλοποιήσεις: Tiled LU Decomposition

k=0, N, B, N/B=4



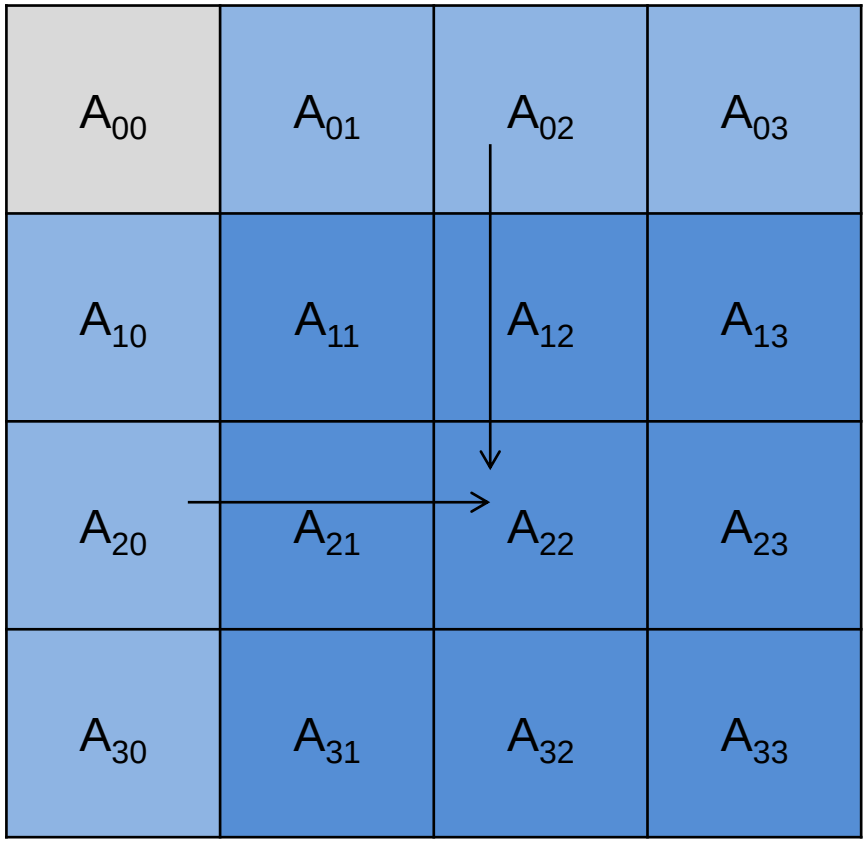
lu_kernel(A_{00})

l_inv=get_inv_l(A_{00})
u_inv=get_inv_u(A_{00})

```
for (i=1;i<range;i++) {
        mm_lower(l_inv, A0i, A0i)
        mm_upper(Ai0, u_inv, Ai0)
    }
```

Εναλλακτικές Υλοποιήσεις: Tiled LU Decomposition

k=0, N, B, N/B=4



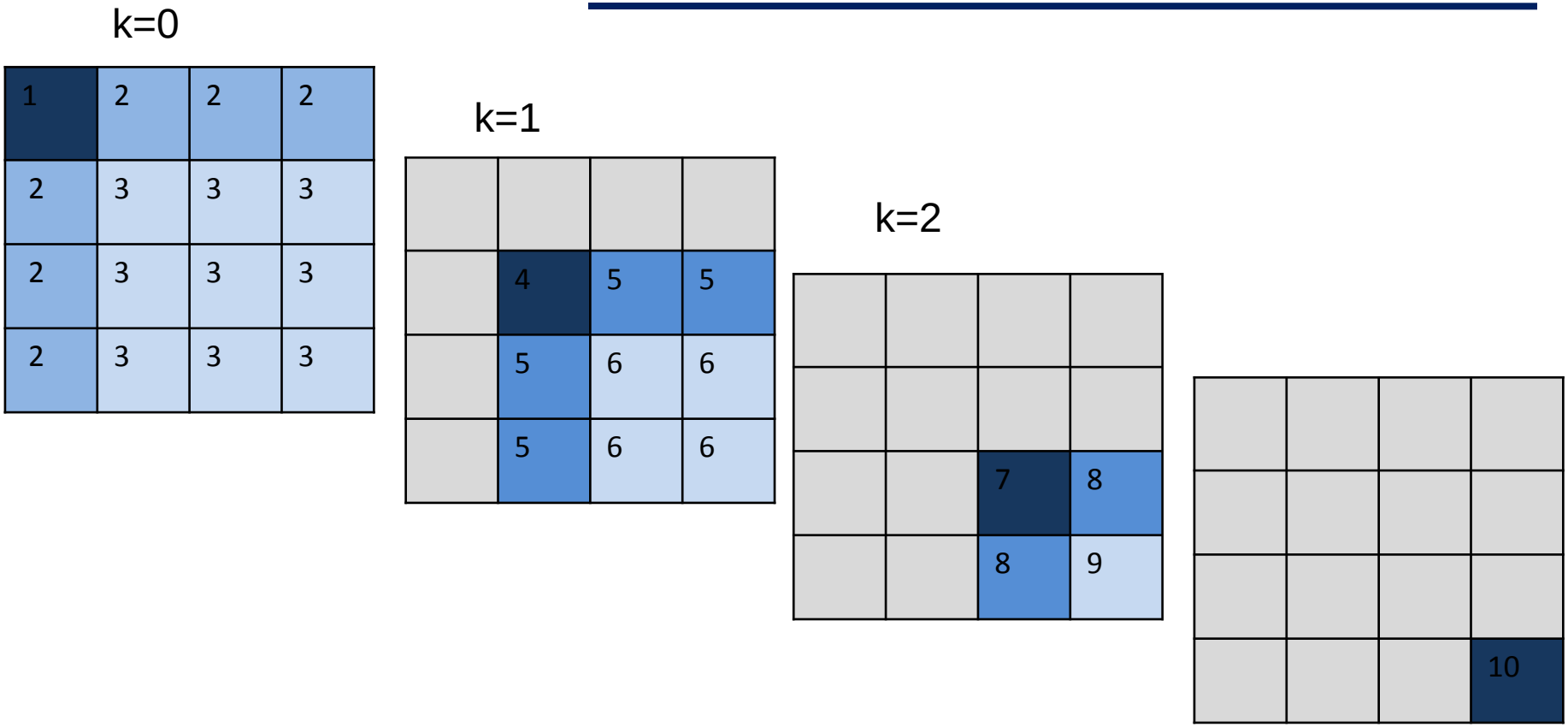
lu_kernel(A_{00})

l_inv=get_inv_l(A_{00})
u_inv=get_inv_u(A_{00})

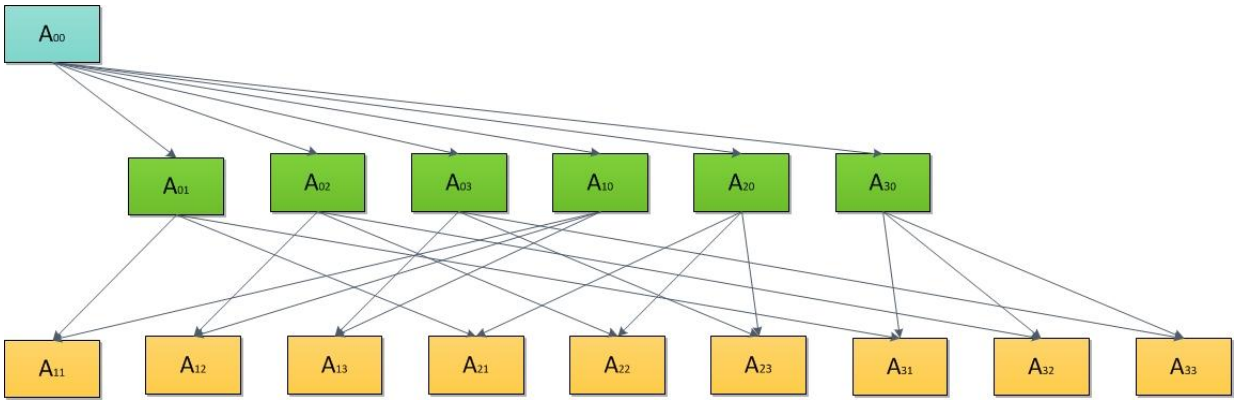
```
for (i=1;i<4;i++) {
        mm_lower(l_inv, A0i, A0i)
        mm_upper(Ai0, u_inv, Ai0)
    }
```

```
for (i=1;i<4;i++)
        for (j=1;j<4;j++)
            mm(Ai0,A0i,Aij)
```

Εναλλακτικές Υλοποιήσεις: Tiled LU decomposition-Scheduling 1



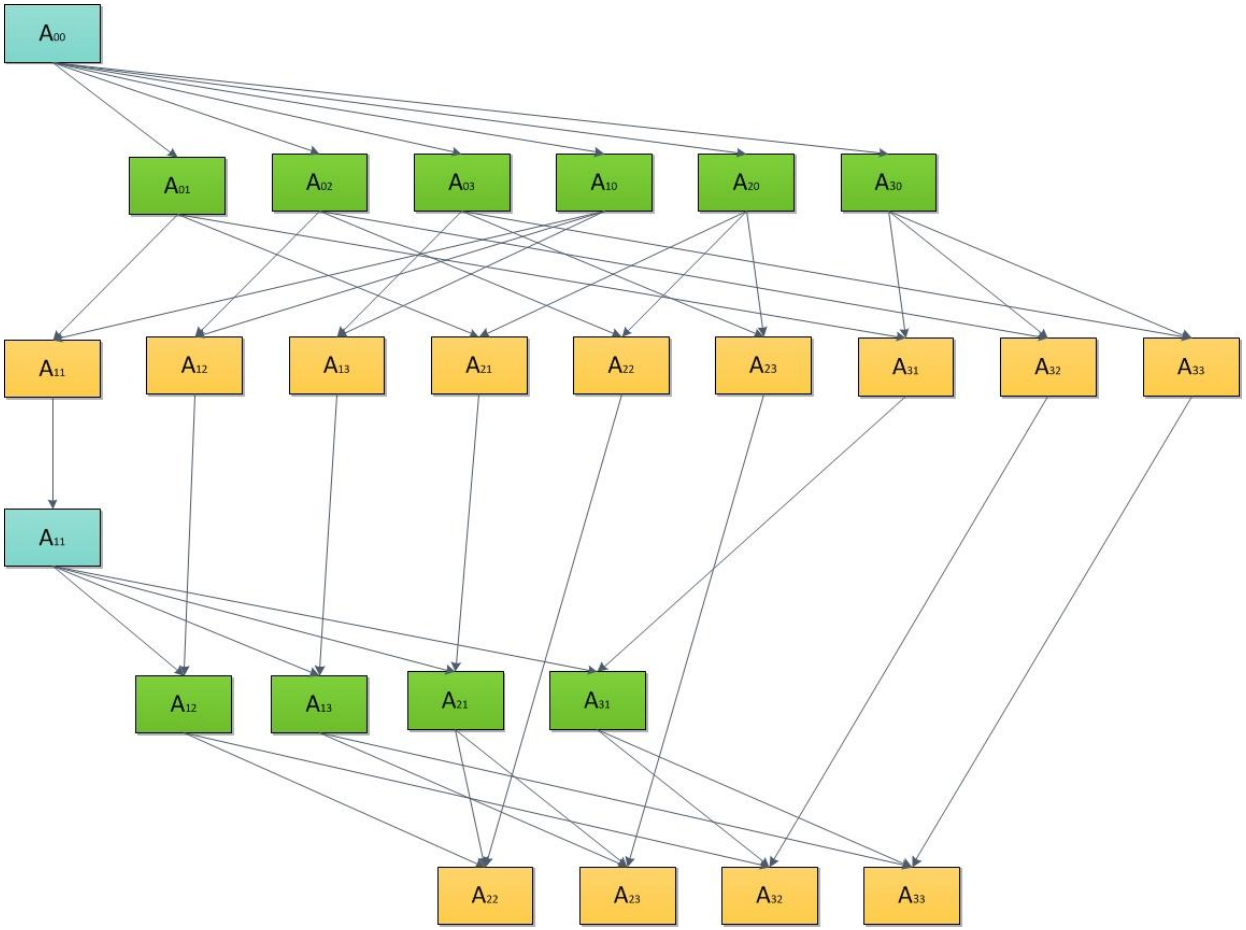
Εναλλακτικές Υλοποιήσεις: Tiled LU decomposition-Task Graph



k=0

A_{00}	A_{01}	A_{02}	A_{03}
A_{10}	A_{11}	A_{12}	A_{13}
A_{20}	A_{21}	A_{22}	A_{23}
A_{30}	A_{31}	A_{32}	A_{33}

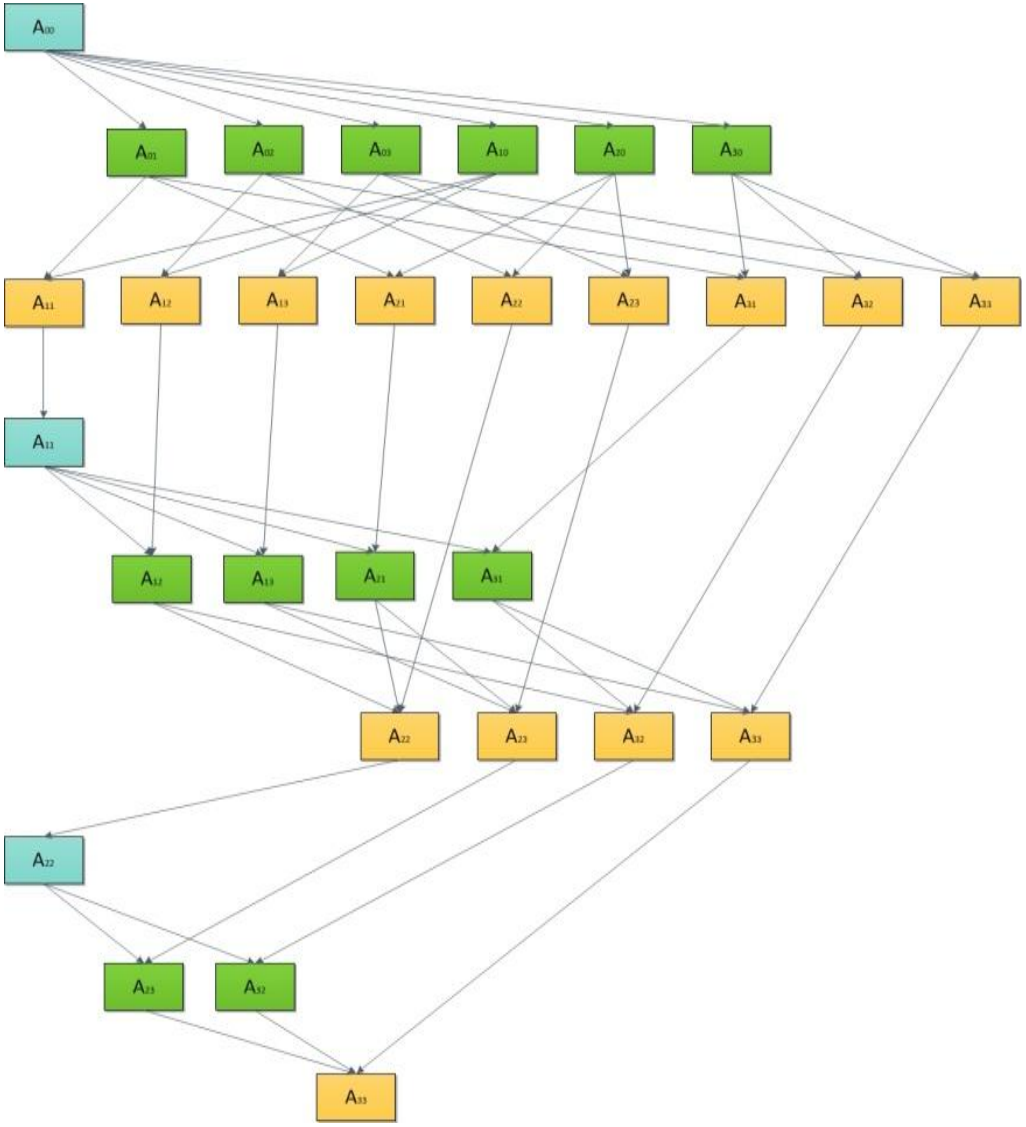
Εναλλακτικές Υλοποιήσεις: Tiled LU decomposition-Task Graph



k=1

A_{00}	A_{01}	A_{02}	A_{03}
A_{10}	A_{11}	A_{12}	A_{13}
A_{20}	A_{21}	A_{22}	A_{23}
A_{30}	A_{31}	A_{32}	A_{33}

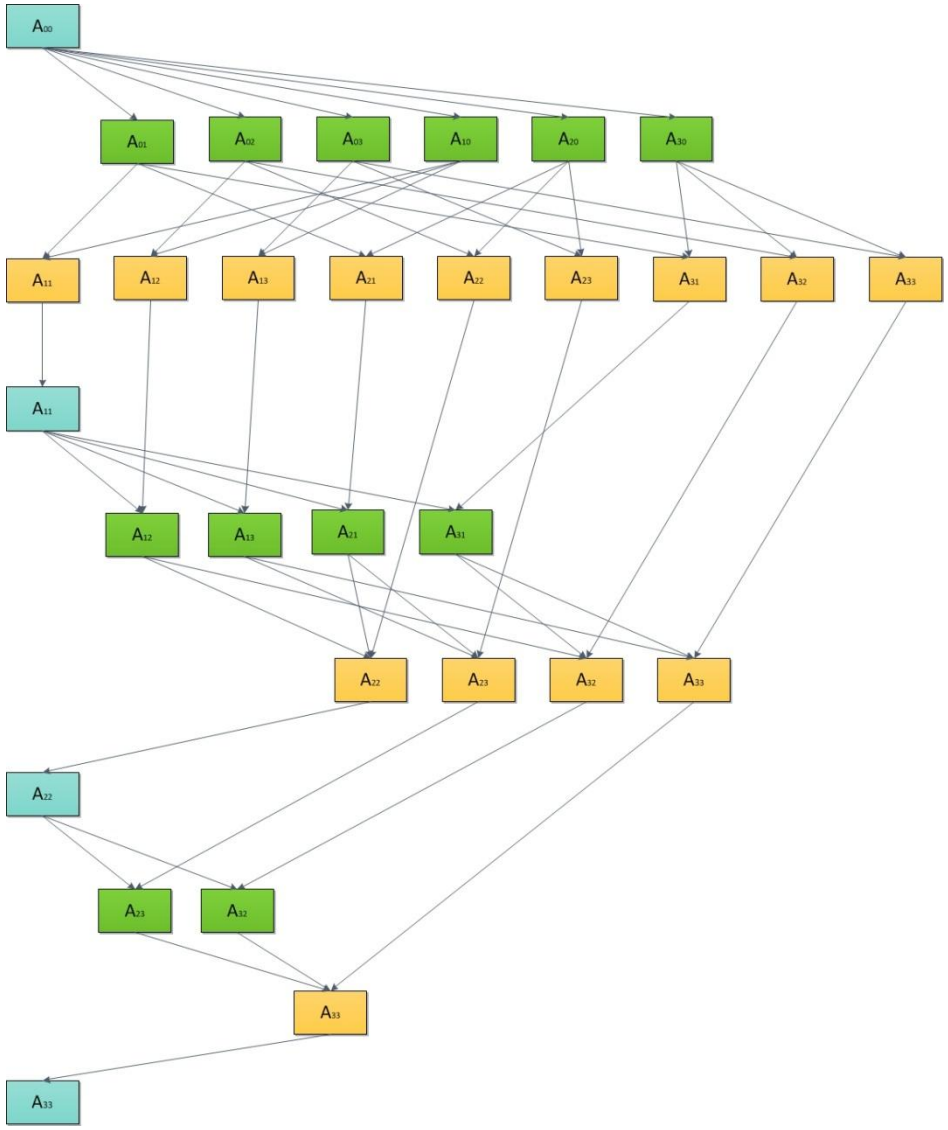
Εναλλακτικές Υλοποιήσεις: Tiled LU decomposition-Task Graph



k=2

A_{00}	A_{01}	A_{02}	A_{03}
A_{10}	A_{11}	A_{12}	A_{13}
A_{20}	A_{21}	A_{22}	A_{23}
A_{30}	A_{31}	A_{32}	A_{33}

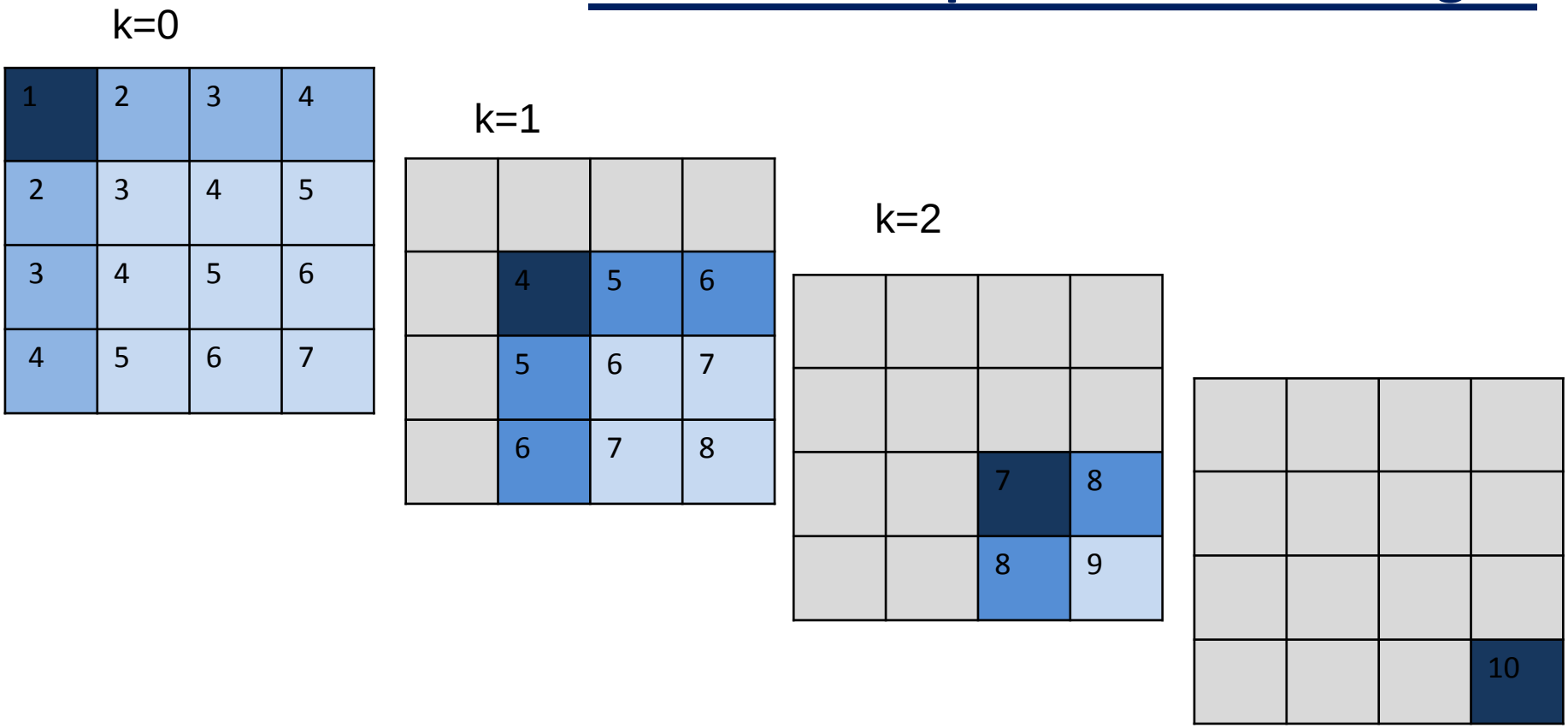
Εναλλακτικές Υλοποιήσεις: Tiled LU decomposition-Task Graph



k=2

A_{00}	A_{01}	A_{02}	A_{03}
A_{10}	A_{11}	A_{12}	A_{13}
A_{20}	A_{21}	A_{22}	A_{23}
A_{30}	A_{31}	A_{32}	A_{33}

Εναλλακτικές Υλοποιήσεις: Tiled LU decomposition-Scheduling 2



Βήματα υλοποίησης

- Ομάδες των 5 ή 6 ατόμων
- Κάθε ομάδα δουλεύει σε ένα εργαλείο
- Επιλέξτε εργαλείο με το οποίο θα δουλέψετε έως 18 Δεκ
  - Οι ομάδες θα καταρτιστούν επιτόπου στο μάθημα 18 Δεκ.
- **Φάση 1:** Υλοποιήσεις / βελτιστοποιήσεις παράλληλων εκδόσεων
  - Standard / recursive / tiled
  - Parallel for / tasks
  - Υβριδικές προσεγγίσεις
  - Scheduling 1 / Scheduling 2 (βλ. διαφάνειες 18, 23)
- **Φάση 2:** Επιλογή των πιο ενδεικτικών υλοποιήσεων και λήψη μετρήσεων
- **Φάση 3:** Παρουσίαση (μεθοδολογία, αποτελέσματα, συμπεράσματα) από κάθε ομάδα
- **Φάση 4:** Συγγραφή αναφοράς (ανά ομάδα)
- **Φάση 5:** Συγγραφή research paper (από όλους)