

# COL874

# Advanced Compiler Techniques

Modules 141-145

Presented by: Jai Arora

# Lec-141: Affine Space Partitions

```
for(i = 0; i < 1000; i++)  
  for(j = 0; j < 1000; j++)  
    A[i, j] = A[i+1, j-1]
```

```
par for(k = 0; k <= 1998; k++)  
  for(l = 0; l <= min(k, 1999 - k); l++)  
    A[k - l, l] = A[k - l + 1, l - 1]
```

- Data dependencies exist across different iterations of the first loop
- Possible to transform the axes of the program to exploit **Synchronization Free Parallelization**
- Iterations of the outer loop in the transformed program can be done in parallel
- **Why?** Because there are no data dependencies across different values of  $k$
- Increased locality due to the transformation as we decrease the reuse distance

# Data Dependence Constraints

$$\begin{aligned} B\underline{i} + \underline{b} &\geq 0 \\ B'\underline{i}' + \underline{b}' &\geq 0 \\ \underline{i} &= \begin{pmatrix} k \\ \ell \end{pmatrix} & k \neq k' \\ \underline{i}' &= \begin{pmatrix} k' \\ \ell' \end{pmatrix} & F\underline{i} + f \\ & & = F'\underline{i}' + f' \end{aligned}$$

- If these constraints yield No Solution, then there is no data dependence across different values of  $k$  for the given pair of static accesses
- Check this for all possible pairs of static accesses (including self pairings)
- If the conditions are satisfied: the loop has 1 degree of parallelism (1 level of loop nest that can be parallelized)

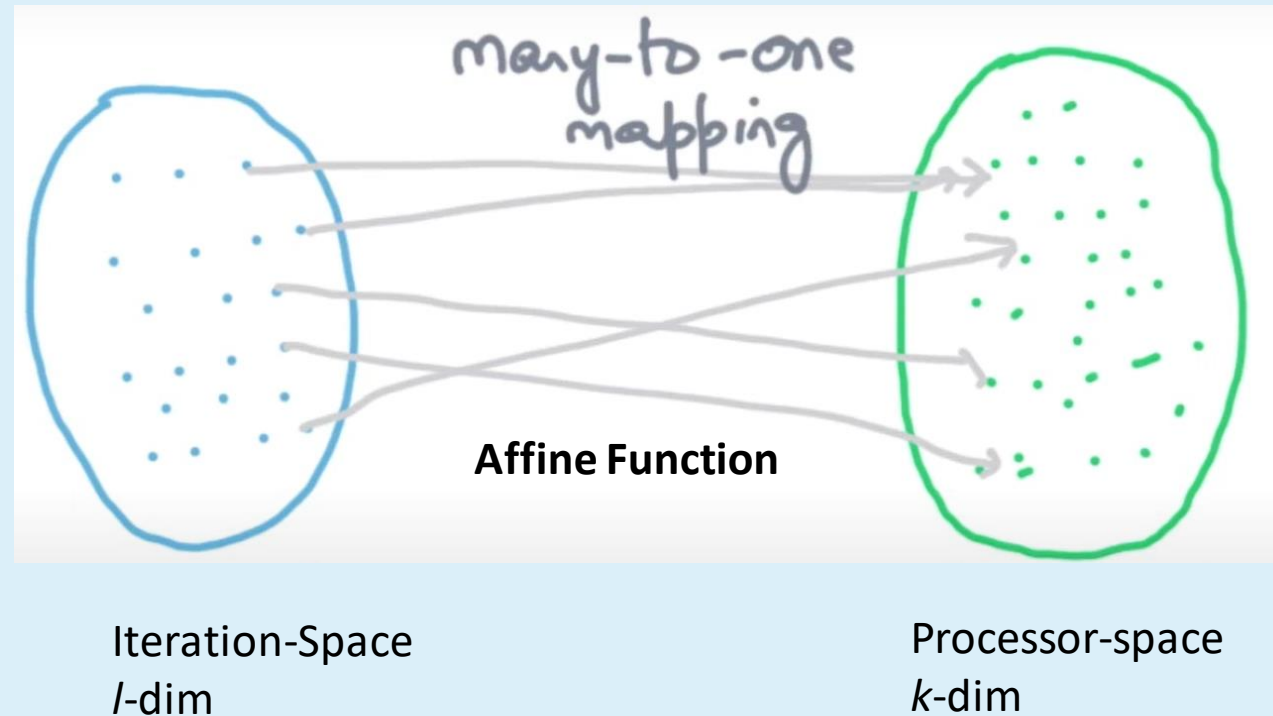
# Degrees of parallelism

- A loop nest has  $k$  degrees of parallelism if it has, within the nest,  $k$  parallelizable *for* loops
- Can create  $O(n^k)$  parallel virtual processors

```
par for(i = 0; k < n; i++)
  par for(j = 0; l < n; j++)
    for(k = 0; k < n; k++)
      A[i, j, k] = A[i, j, k - 1]
```

**2 degrees of parallelism**

# Affine Space Partitions



- $l \geq k$ : it is a many-to-one map, and a function (maps all the iterations)
- Need to use as many as (virtual) processors
- Partition: All the iterations in the partition are mapped to the same processor
- Constraint: This map needs to be an affine function

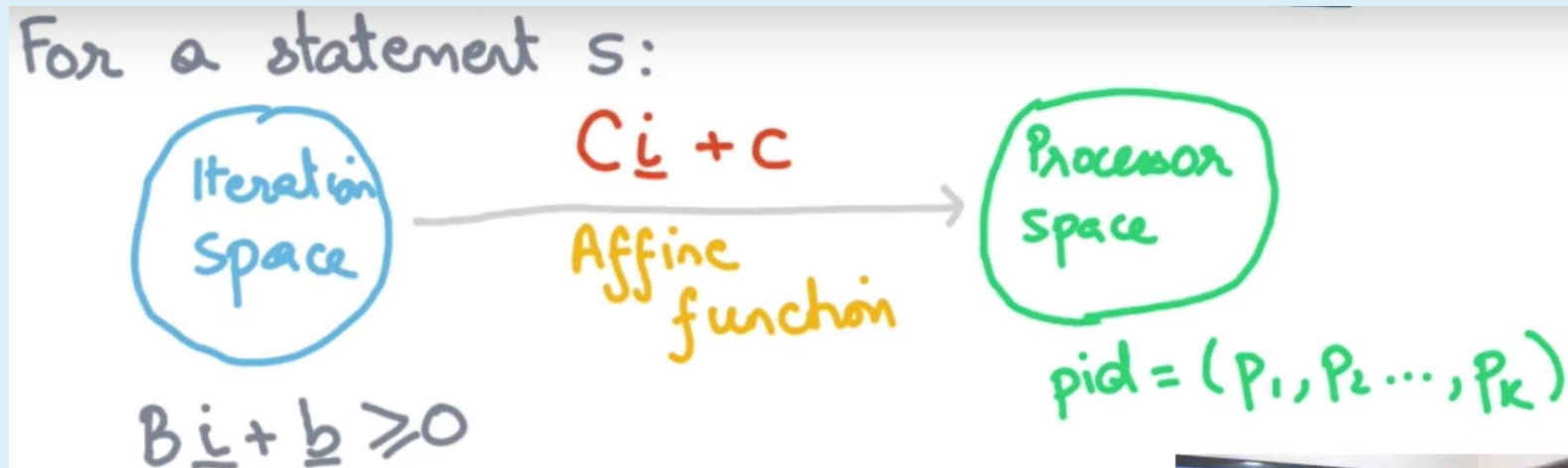
# Affine Space Partitions

```
for (i=0; i < n; i++)  
  A[i]=0;  
  B[i]=1;
```

→ No data dependency

- Can have  $O(2n)$  virtual processors
- Our analysis should be able to exploit this
- Each 3AC statement (static access) is analyzed separately for maximum parallelism

# Lec-142: Space Partition Constraints



$$pid = Ci + c$$

- $(C, c)$  are different for each statement  $s$
- Variation: Could have a piecewise affine function instead of an affine function
- Piecewise affine functions can be solved using polyhedral analysis (potentially giving better results)
- Tradeoff: Performance vs Cost
- Restrict ourselves to Affine functions

# Space Partition Constraints

- Need to find a solution to  $(C, c)$  satisfying data dependency constraints
- Trivial Solution:  $C = (0 \ 0 \ \dots \ 0)$  and  $c = (0)$
- Represents a zero-dimensional processor space
- Everything mapped to the same processor
- Valid solution but not very useful due to no parallelism
- Also need to maximize the processor space dimension / rank of  $C$
- Affine Partition: a  $(C, c)$  solution for each statement  $s$  represents an **Affine partition**



# Space Partition Constraints

For  $s_1 = (F_1, f_1, B_1, b_1)$  and  $s_2 = (F_2, f_2, B_2, b_2)$ ,  
the partitions  $C_1, c_1$  and  $C_2, c_2$  must be  
such that:

For all  $\underline{i}_1 \in \mathbb{Z}^{d_1}$  and  $\underline{i}_2 \in \mathbb{Z}^{d_2}$   
if  $B_1 \underline{i}_1 + \underline{b}_1 \geq 0$        $B_2 \underline{i}_2 + \underline{b}_2 \geq 0$

$$F_1 \underline{i}_1 + f_1 = F_2 \underline{i}_2 + f_2$$

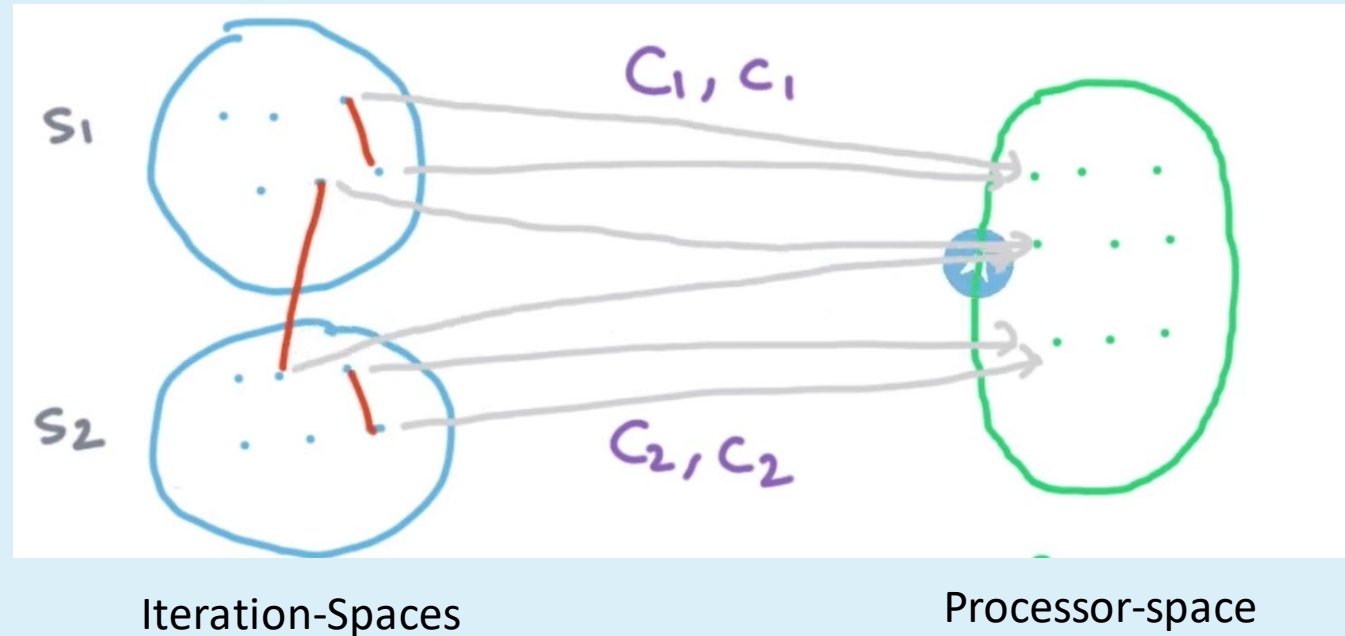
(whenever there is a data dependence)

then  $C_1 \underline{i}_1 + \underline{c}_1 = C_2 \underline{i}_2 + \underline{c}_2$



- Data dependent iterations are mapped to the same processor
- Unknowns:  $C_1, c_1, C_2, c_2$

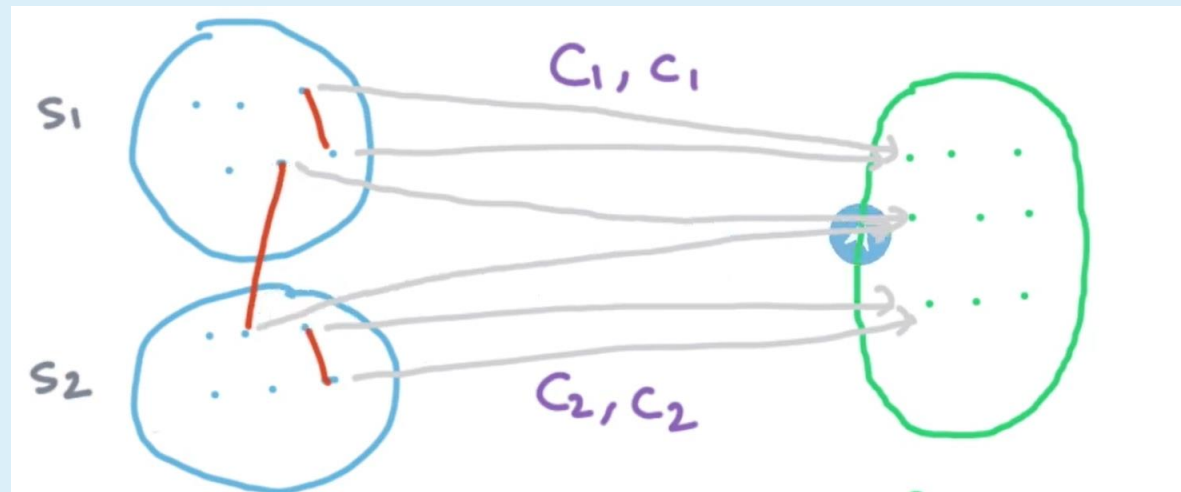
# Space Partition Constraints



- Data dependence could be across the same statement or different statements
- Chose these unknowns such that the constraints are satisfied, and the ranks are maximized

# Lec-143: Maximum Rank Affine Partition

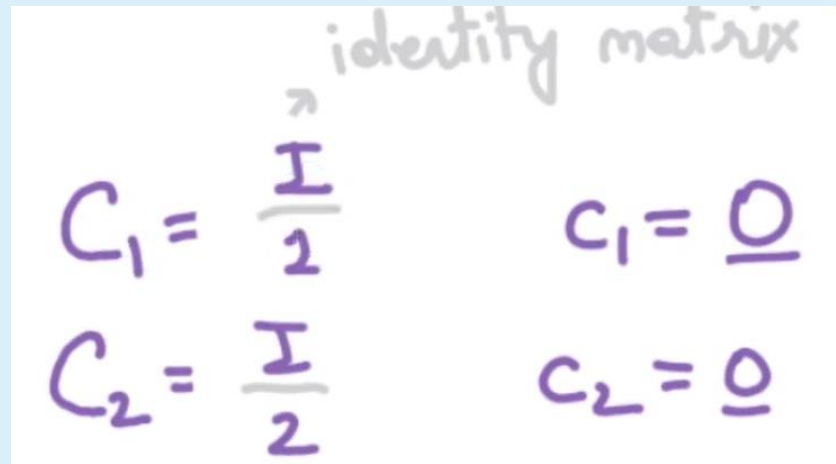
- Affine Partitions help us to argue about the processors, iterations and the data in a homogeneous way



- Trivial Solution:  $C_1 = \underline{0}$ ,  $c_1 = \underline{0}$ ,  $C_2 = \underline{0}$ ,  $c_2 = \underline{0}$
- But no parallelism

# Maximum Rank Affine Partition

- Max Rank Solution:



A handwritten note on a white background. At the top, it says "identity matrix" with an arrow pointing to the 'I' in the first equation. Below that, there are two rows of equations. The first row shows  $C_1 = \frac{I}{2}$  and  $C_1 = \underline{0}$ . The second row shows  $C_2 = \frac{I}{2}$  and  $C_2 = \underline{0}$ . The 'I' and '0' in the second equations are underlined.

$$C_1 = \frac{I}{2} \quad C_1 = \underline{0}$$
$$C_2 = \frac{I}{2} \quad C_2 = \underline{0}$$

- Desirable, but may not satisfy Data dependency constraints
- Interested in the Max Rank solution satisfying the data dependence constraints

# Max Rank constraints

For  $k$  statements  $s_1, s_2, \dots, s_k$ , choose  $(c_1, c_1)$   
 $(c_2, c_2), \dots, (c_k, c_k)$  of maximum rank  
such that for every pair of statements  $s_a, s_b$   
 $\forall \underline{i}_a \in \mathbb{Z}^{d_a}, \underline{i}_b \in \mathbb{Z}^{d_b}$   
if  $B_a \underline{i}_a + \underline{b}_a \geq 0$      $B_b \underline{i}_b + \underline{b}_b \geq 0$   
 $F_a \underline{i}_a + \underline{f}_a = F_b \underline{i}_b + \underline{f}_b$   
then  $C_a \underline{i}_a + \underline{c}_a = C_b \underline{i}_b + \underline{c}_b$



# Space Partitioning Example

- Running Example:

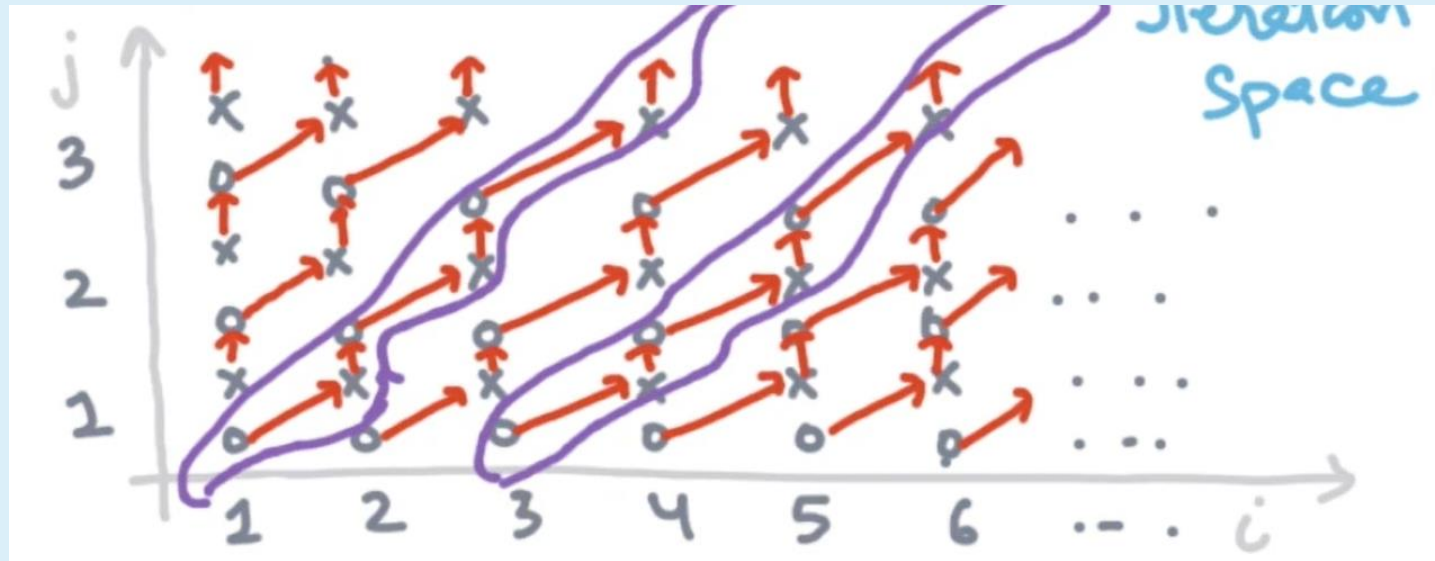
```
for (i=1; i<=100; i++)  
  for (j=1; j<=100; j++)  
    X[i,j] = X[i,j] + Y[i-1,j];  
    Y[i,j] = Y[i,j] + X[i,j-1];
```

- Six Statements (one for each access); two writes
- Data dependences:
  - $X[i, j] \leftrightarrow X[i, j]$  (both R/W)
  - $X[i, j] \leftrightarrow X[i, j-1]$
  - $Y[i, j] \leftrightarrow Y[i, j]$  (both R/W)
  - $Y[i, j] \leftrightarrow Y[i-1, j]$

# Space Partitioning Example

- $X[i, j]$  with itself won't have any space partitioning because  $[i, j]$  is a full rank access
- $X[i, j] = X[i, j] + Y[i-1, j]$ : we do have a data dependency, but in the same iteration
- No meaningful constraints
- We need only 2 affine functions, one for each C-statement (based on scalar dependencies)

# Space Partitioning Example



- All the dependent iterations should be mapped to the same processor
- Disjoint chains are formed
- Could map each of the chains to a separate processor
- Need to find  $C_1, c_1, C_2, c_2$  such that each chain is mapped to the same processor
- Answer: 1-D mapping for each statement
  - $P_1 = i - j - 1$  for  $s_1$
  - $P_2 = i - j$  for  $s_2$



# Lec-144: Space Partition Constraints Example

- Only data dependencies:
  - $X[i, j] \leftrightarrow X[i, j-1]$  (I)
  - $Y[i-1, j] \leftrightarrow Y[i, j]$  (II)
- 12 unknowns

Handwritten matrices for  $C_1$  and  $C_2$  with some elements crossed out:

$$C_1 = \begin{pmatrix} C_{111} & C_{112} \\ \underline{C_{121}} & \underline{C_{122}} \end{pmatrix} \quad C_1 = \begin{pmatrix} C_{11} \\ \underline{C_{12}} \end{pmatrix}$$
$$C_2 = \begin{pmatrix} C_{211} & C_{212} \\ \underline{C_{221}} & \underline{C_{222}} \end{pmatrix} \quad C_2 = \begin{pmatrix} C_{21} \\ \underline{C_{22}} \end{pmatrix}$$

- Could use previous knowledge that dimensionality of the processor space = 1 ( $C_1, C_2$  have dependent rows)
- For now, assume that  $C_{121} = C_{122} = 0 = C_{221} = C_{222}$

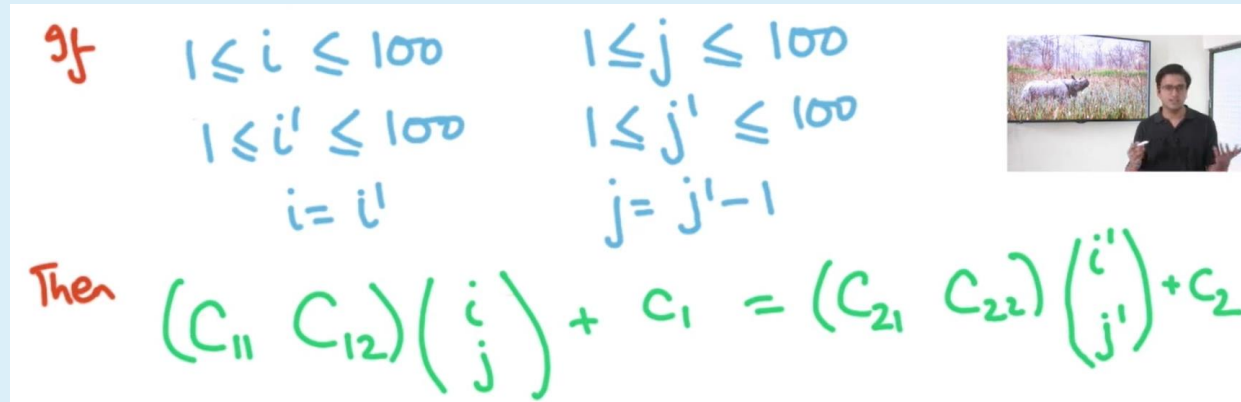
# Space Partition Constraints Example

- New Problem:

$$C_1 = (\underline{C_{11}} \quad C_{12}) \qquad C_1 = (c_1)$$
$$C_2 = (C_{21} \quad \underline{C_{22}}) \qquad C_2 = (c_2)$$


# Space Partitioning Constraints

- For dependency (1):



of  $1 \leq i \leq 100$        $1 \leq j \leq 100$   
 $1 \leq i' \leq 100$        $1 \leq j' \leq 100$   
 $i = i'$        $j = j' - 1$

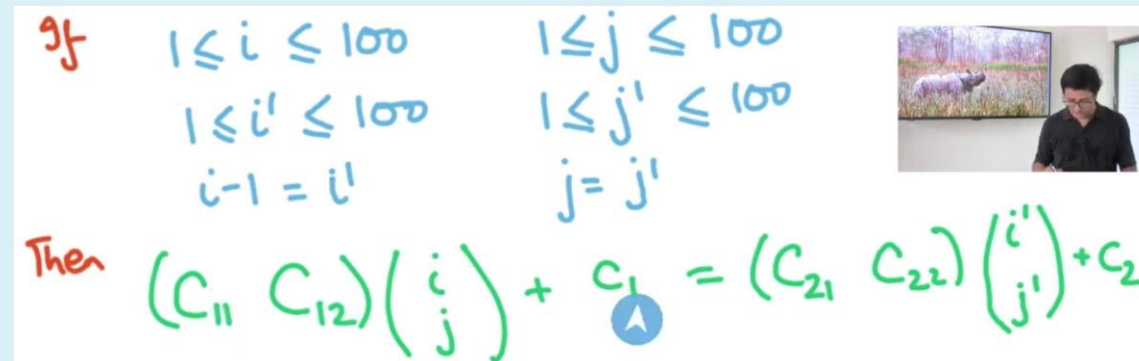
Then  $(C_{11} \ C_{12}) \begin{pmatrix} i \\ j \end{pmatrix} + c_1 = (C_{21} \ C_{22}) \begin{pmatrix} i' \\ j' \end{pmatrix} + c_2$



- Possible Solution:  $C_{11} = C_{21} = 1$ ,  $C_{12} = C_{22} = 0$ ,  $c_1 = c_2 = 0$
- Iteration  $(i, j)$  is mapped to processor  $i$
- All conditions are satisfied (one of the possible solutions)
- But this is not the only constraint

# Space Partitioning Constraints

- For dependency (II):



if  $1 \leq i \leq 100$        $1 \leq j \leq 100$   
 $1 \leq i' \leq 100$        $1 \leq j' \leq 100$   
 $i-1 = i'$        $j = j'$

Then  $(C_{11} \ C_{12}) \begin{pmatrix} i \\ j \end{pmatrix} + c_1 = (C_{21} \ C_{22}) \begin{pmatrix} i' \\ j' \end{pmatrix} + c_2$

- Possible Solution:  $C_{11} = C_{21} = 0$ ,  $C_{12} = C_{22} = 1$ ,  $c_1 = c_2 = 0$
- This solution satisfies the second dependency but not the first
- Previous solution does not satisfy this dependency
- Need a solution that satisfies both the constraints

# Space Partitioning Example Solution

- Solution:  $C_{11} = C_{21} = 1, C_{12} = C_{22} = -1, c_1 = -1, c_2 = 0$

$$i - j - 1 = i' - j'$$

- This holds for both the constraints

# Lec-145: Solving Space Partition Constraints

Handwritten notes showing two constraints for space partitioning. The left side shows a constraint where  $i = i'$  and  $j = j' - 1$ . The right side shows a constraint where  $i - 1 = i'$  and  $j = j'$ . Both constraints involve a system of linear equations with binomial coefficients.

Left side:

$$\forall i, j, i', j'$$

if  $1 \leq i, i' \leq 100$   
 $1 \leq j, j' \leq 100$   
 $i = i' \quad j = j' - 1$

then  $(C_{11} \ C_{12}) \binom{i}{j} + c_1 =$   
 $(C_{21} \ C_{22}) \binom{i'}{j'} + c_2$

Right side:

$$\forall i, j, i', j'$$

if  $1 \leq i, i' \leq 100$   
 $1 \leq j, j' \leq 100$   
 $i - 1 = i' \quad j = j'$

then  $(C_{11} \ C_{12}) \binom{i}{j} + c_1 =$   
 $(C_{21} \ C_{22}) \binom{i'}{j'} + c_2$

- Both these constraints must be satisfied
- $i, j, i', j'$  are not unknowns

# Solving Space Partition Constraints

- Use gaussian elimination to get rid of some variables
- Use the constrains to eliminate  $i', j'$

The image shows two columns of handwritten mathematical work. The left column starts with the universal quantifier  $\forall i, j, i', j'$  where  $i', j'$  are crossed out in green. Below this, a red 'if' is followed by the constraints  $1 \leq i, i' \leq 100$  and  $1 \leq j, j' \leq 100$ . A green line is drawn through the equations  $i = i'$  and  $j = j'$ . The word 'then' is written in red, followed by the matrix equation  $(C_{11} \ C_{12}) \begin{pmatrix} i \\ j \end{pmatrix} + c_1 = (C_{21} \ C_{22}) \begin{pmatrix} i \\ j+1 \end{pmatrix} + c_2$ . The right column follows the same structure but with the green line through  $i = i'$  and  $j = j'$  crossed out, and the final matrix equation uses  $\begin{pmatrix} i-1 \\ j \end{pmatrix}$  instead of  $\begin{pmatrix} i \\ j+1 \end{pmatrix}$ .

$$\forall i, j, i', j'$$

if  $1 \leq i, i' \leq 100$   
 $1 \leq j, j' \leq 100$

~~$i = i'$     $j = j'$~~

then  $(C_{11} \ C_{12}) \begin{pmatrix} i \\ j \end{pmatrix} + c_1 = (C_{21} \ C_{22}) \begin{pmatrix} i \\ j+1 \end{pmatrix} + c_2$

$$\forall i, j, i', j'$$

if  $1 \leq i, i' \leq 100$   
 $1 \leq j, j' \leq 100$

~~$i = i'$     $j = j'$~~

then  $(C_{11} \ C_{12}) \begin{pmatrix} i \\ j \end{pmatrix} + c_1 = (C_{21} \ C_{22}) \begin{pmatrix} i-1 \\ j \end{pmatrix} + c_2$

# Solving Space Partition Constraints

- Rewrite the equations:

$$\begin{pmatrix} (C_{11} - C_{21}) & (C_{12} - C_{22}) \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + C_1 - C_{22} - C_2 = 0$$

$$\begin{pmatrix} (C_{11} - C_{21}) & (C_{12} - C_{22}) \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + C_1 + C_{21} - C_2 = 0$$



# Solving Space Partition Constraints

- Overapproximate the behavior of iteration variables for these constraints
- Assume that the equations hold for all real values of  $i, j$
- Which means that the coefficient of  $i, j$  and the constant term are zero

$$\begin{aligned}C_{11} &= C_{21} \\C_{12} &= C_{22} \\c_1 &= c_2 + C_{22} \\c_1 &= c_2 - C_{21}\end{aligned}$$

# Solving Space Partition Constraints

- On solving, we get:

$$\begin{aligned} -C_{12} &= -C_{22} = C_{21} = C_{11} = C \\ c_1 &= c_2 - C \end{aligned}$$

- Actual constant values don't matter because they only shift the space of processor IDs
- WLOG, pick  $C = 1$ ,  $c_2 = 0$
- So, we get the same solution as before
  - $P_1 = i - j - 1$
  - $P_2 = i - j$
- Remaining: we need to make sure that all the iterations mapped to the same processor preserve the relative order of execution of these iterations

# Thank You!

- Jai Arora