Mathematical Approach to the Performance Evaluation of Matrix-matrix Multiply Algorithm on a Two Level Parallel Architecture

Luisa D'Amore,
Valeria Mele,
Giuliano Laccetti
Almerico Murli

Krakow, 7th September 2015
PPAM 2015 - Workshop on Models, Algorithms and Methodologies for Hybrid Parallelism in New HPC Systems

Summary

1. Goal of the talk
   - A structured designing approach to predict performances
2. The choosen problem: matrix-matrix multiply
   - Decomposition in square blocks
3. Performance prediction of a sequential software
4. Performance prediction of a parallel software (BroadcastMultiplyRolling)
5. The subproblem: blocks multiply
   - Submatrices: matrix-matrix multiply of a different dimension
   - Repeat the steps 1-4
6. Merging the approaches
   - Performance prediction of a two level parallel software
7. Conclusions about the method

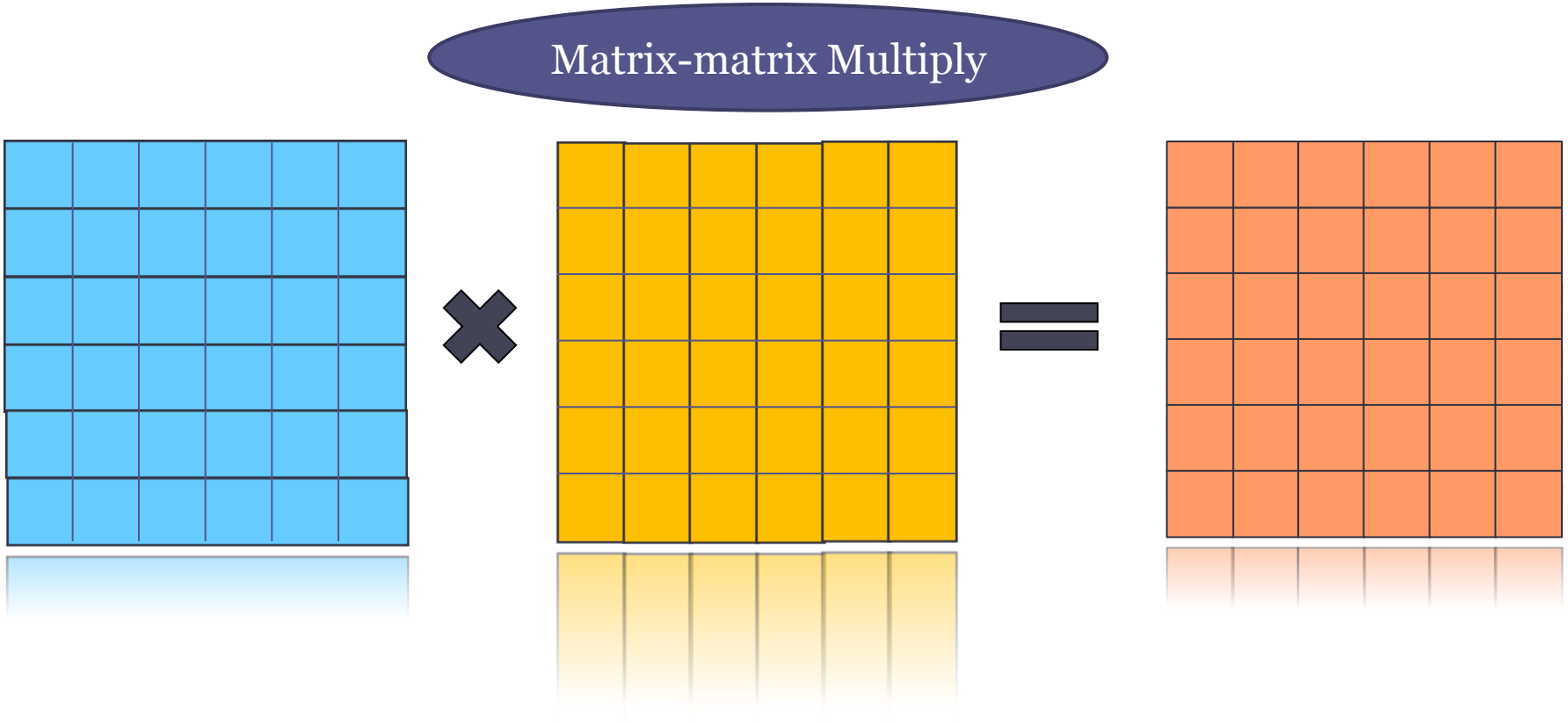
Goal

Show how to
structurate the software designing approach
to address
the best (requested) performance

Using a mathematical approach as described in

L. D'Amore, G. Laccetti, V. Mele, D. Romano, A. Murli , *On a Mathematical Approach for Analyzing Parallel Algorithms*, submitted to CALCOLO - A Quarterly on Numerical Analysis and Theory of Computation

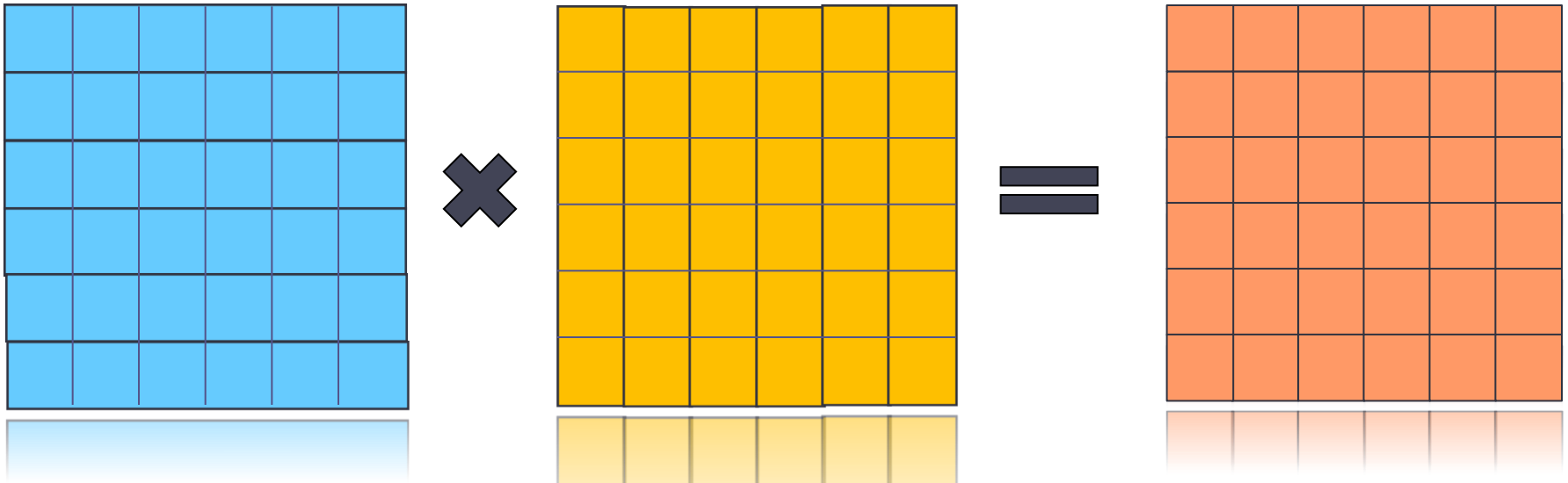
The Problem



The Problem

- A, B, C are nxn matrices

Matrix-matrix Multiply

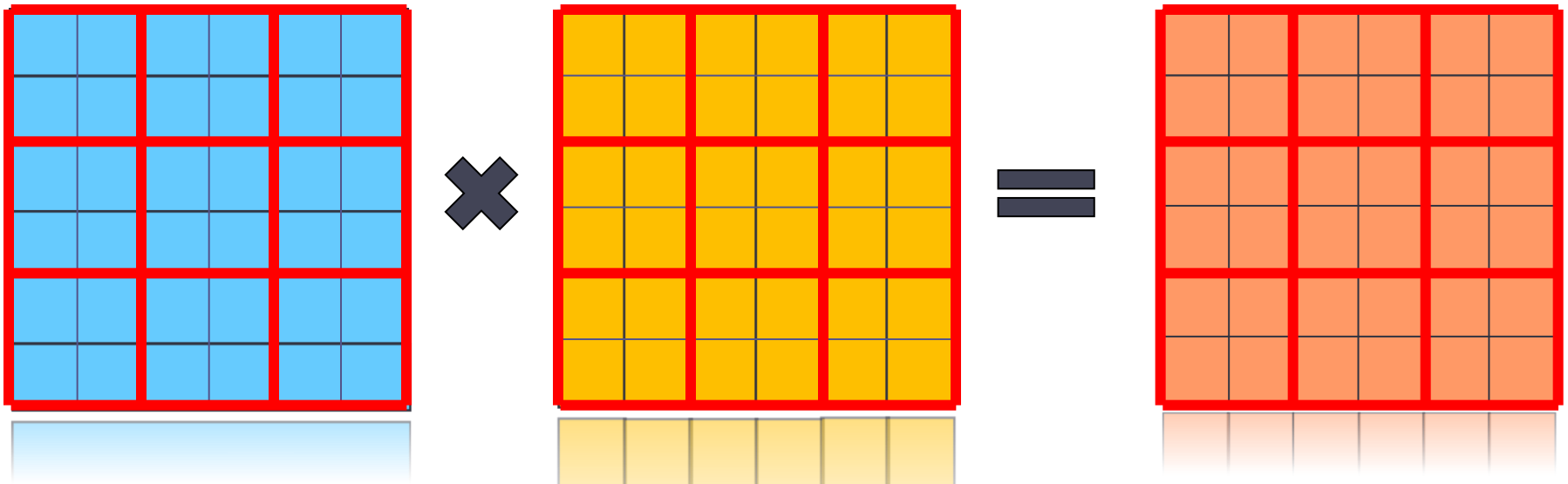


We can decompose the problem…

The Problem

- A, B, C are nxn matrices
- 9 sub-blocks of n/3 x n/3 dimension

Matrix-matrix Multiply

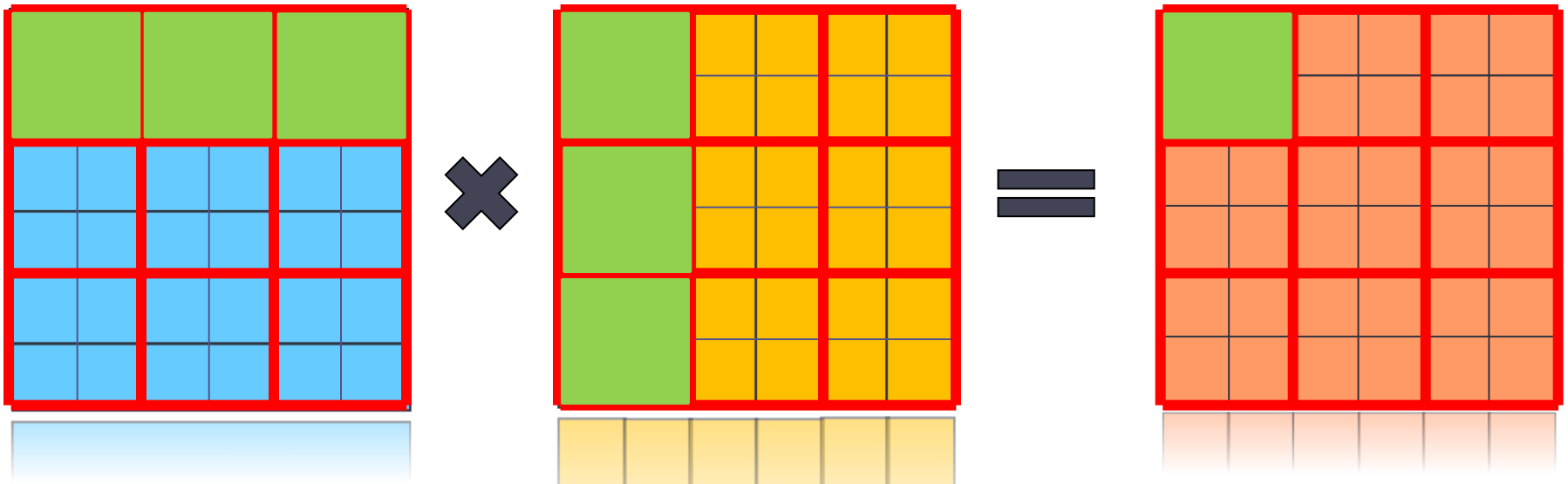


...partitioning the matrices in submatrices (blocks)

The Problem

- A, B, C are nxn matrices
- 9 sub-blocks of n/3 x n/3 dimension

Matrix-matrix Multiply



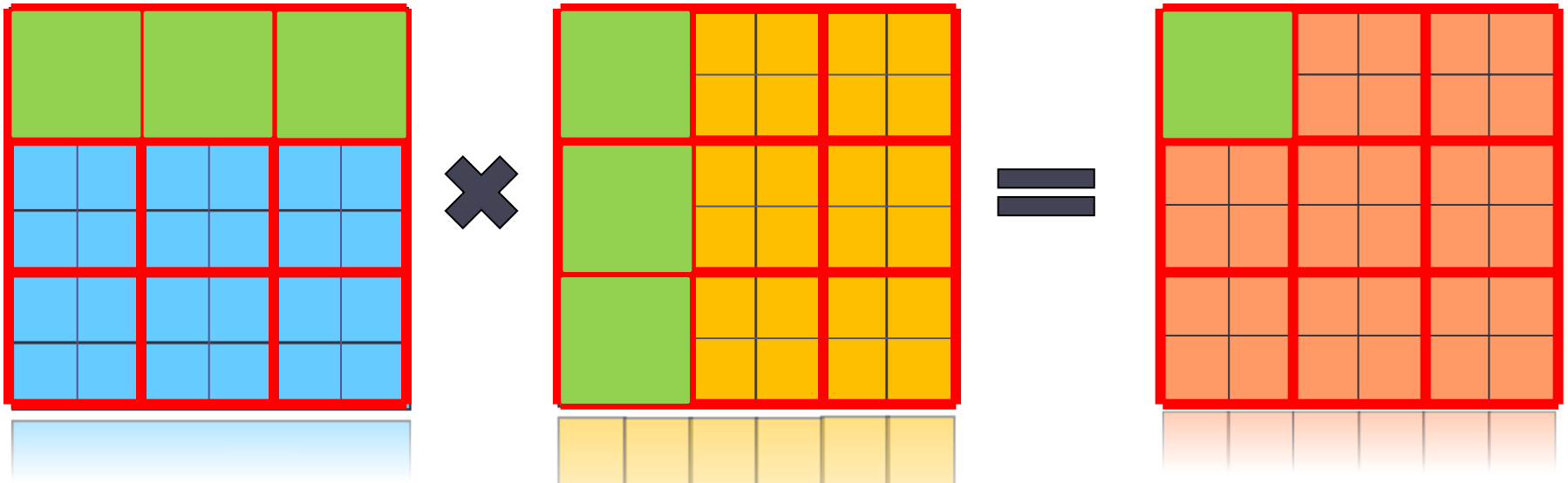
$$Cblock_{i,j} = Ablock_{i,0} \times Bblock_{0,j} + Ablock_{i,1} \times Bblock_{1,j} + Ablock_{i,2} \times Bblock_{2,j}$$

3 matrix-matrix multiply for each Cblock

The Problem

- A, B, C are nxn matrices
- 9 sub-blocks of n/3 x n/3 dimension

Matrix-matrix Multiply



$$Cblock_{i,j} = Ablock_{i,0} \times Bblock_{0,j} + Ablock_{i,1} \times Bblock_{1,j} + Ablock_{i,2} \times Bblock_{2,j}$$

27 sub-problems

The Problem

So we have the decomposition

$$D = \{matmul^{i}_{\frac{n}{3}\times\frac{n}{3}}\}_{0\leq i<(27)}$$

Where the subproblems are independent from each other 9 a time:
-A product for each Cblock can be treated at a time
-For each Cblock,
-the second product need the first solution to add
-The third product need the first and the second solution to add

And we have the **dependence matrix**

$$M_D = \begin{bmatrix} matmul^{0}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{1}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{2}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{8}_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^{9}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{10}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{11}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{17}_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^{18}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{19}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{20}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{26}_{\frac{n}{3}\times\frac{n}{3}} \end{bmatrix}$$

The Problem

So we have the decomposition

$$D = \{matmul^{i}_{\frac{n}{3}\times\frac{n}{3}}\}_{0\leq i<(27)}$$

Where the subproblems are independent from each other 9 a time:
-A product for each Cblock can be treated at a time
-For each Cblock,
-the second product need the first solution to add
-The third product need the first and the second solution to add

9 columns
3 rows

And we have the **dependence matrix**

$$M_D = \begin{bmatrix} matmul^{0}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{1}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{2}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{8}_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^{9}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{10}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{11}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{17}_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^{18}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{19}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{20}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{26}_{\frac{n}{3}\times\frac{n}{3}} \end{bmatrix}$$

The Problem

The problem AxB=C has

- Concurrency Degree: 9
- Dependency Degree: 3

So we have the decomposition

$$D = \{matmul^{i}_{\frac{n}{3}\times\frac{n}{3}}\}_{0\leq i<(27)}$$

Where the subproblems are independent from each other 9 a time:
-A product for each Cblock can be treated at a time
-For each Cblock,
-the second product need the first solution to add
-The third product need the first and the second solution to add

And we have the **dependence matrix**

$$M_D = \begin{bmatrix} matmul^{0}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{1}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{2}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{8}_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^{9}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{10}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{11}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{17}_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^{18}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{19}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{20}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{26}_{\frac{n}{3}\times\frac{n}{3}} \end{bmatrix}$$

The Algorithm (sequential)

MACHINE $M_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer a n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,
- for each execution we need a reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D,M1,1}$ containing 27 of this operators to execute one after another, and we have the **execution matrix** $E_{D,M1,1}$

$$E_{D,M1,1} = \begin{bmatrix} \otimes_0 \\ \otimes_1 \\ \vdots \\ \otimes_{26} \end{bmatrix}$$

r_{E1}=27 rows

The Algorithm (sequential)

MACHINE $M_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer a n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,
- for each execution we need a reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D,M1,1}$ containing 27 of this operators to execute one after another, and we have the **execution matrix** $E_{D,M1,1}$ and the **memory matrix** $MEM_{D,M1,1}$

$$E_{D,M1,1}=\begin{bmatrix}\otimes_0\\ \otimes_1\\ \vdots\\ \otimes_{26}\end{bmatrix}$$

r_{E1}=27 rows

$$MEM_{D,M1,1}=\begin{bmatrix}rmem_0(\cdot)\\ wmem_0(\cdot)\\ rmem_1(\cdot)\\ wmem_1(\cdot)\\ \vdots\\ rmem_{26}\\ wmem_{26}\end{bmatrix}$$

r_{MEM1}=52 rows

The Algorithm (sequential)

MACHINE $M_{1,1}$
•One processing unit
•the processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer a n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,
•for each execution we need a reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D,M1,1}$ containing 27 of this operators to execute one after another, and we have the **execution matrix** $E_{D,M1,1}$ and the **memory matrix** $MEM_{D,M1,1}$

$$\begin{bmatrix} rmem_0(\cdot) \\ \\ \\ \\ \\ \\ \\ \\ wmem_{26} \end{bmatrix}$$

$E_{D,}$ ows

The algorithm becomes software

The Algorithm (sequential)

MACHINE $M_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer a n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,
- for each execution we need a reading (before the execution) and a writing (after the execution)

The execution time of $ALG_{D,M1,1}$ will be

$$T(ALG_{D,\mathcal{M}_{1,1}}) = r_{E1} \cdot T_r = 26 \cdot C(\otimes) \cdot tflops$$

Complexity of the operator (number of operations)

$$E_{D,M1,1} = \begin{bmatrix} \otimes_0 \\ \otimes_1 \\ \vdots \\ \otimes_{26} \end{bmatrix}$$

r_{E1}=27 rows

$$MEM_{D,M1,1} = \begin{bmatrix} rmem_0(\cdot) \\ wmem_0(\cdot) \\ rmem_1(\cdot) \\ wmem_1(\cdot) \\ \vdots \\ rmem_{26} \\ wmem_{26} \end{bmatrix}$$

r_{MEM1}=52 rows

The Algorithm (sequential)

MACHINE $M_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer a n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,
- for each execution we need a reading (before the execution) and a writing (after the execution)

The memory access time of $SW_{D,M1,1}$ will be

$$T_M(Sw_{D,\mathcal{M}_{1,1}}) = r_{MEM1} \cdot tblock_{mem} = 52 \cdot tblock_{mem}$$

$$E_{D,M1,1} = \begin{bmatrix} \otimes_0 \\ \otimes_1 \\ \vdots \\ \otimes_{26} \end{bmatrix}$$

r_{E1}=27 rows

$$MEM_{D,M1,1} = \begin{bmatrix} rmem_0(\cdot) \\ wmem_0(\cdot) \\ rmem_1(\cdot) \\ wmem_1(\cdot) \\ \vdots \\ rmem_{26} \\ wmem_{26} \end{bmatrix}$$

r_{MEM1}=52 rows

The Algorithm (sequential)

MACHINE $M_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer a n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,
- for each execution we need a reading (before the execution) and a writing (after the execution)

The execution time of $SW_{D,M1,1}$ will be

$$\begin{aligned} T(Sw_{D,\mathcal{M}_{1,1}}) &= T(ALG_{D,\mathcal{M}_1}) + T_M(Sw_{D,\mathcal{M}_{1,1}}) \\ &= 26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem} \end{aligned}$$

The Algorithm (parallel)

MACHINE $M_{9,9}$
•9 processing unit
•each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,

The Algorithm (parallel)

We cannot use more than 9 units without changing decomposition of the problem, becouse of the concurrency degree!

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,

The Algorithm (parallel)

We can imagine a cluster with (at least) 9 nodes

MACHINE $M_{9,9}$
•9 processing unit
•each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,

The Algorithm (parallel)

MACHINE $M_{9,9}$
•9 processing unit
•each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{mem}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,

This is, for example, the case of communications among nodes in a cluster, so now we call it $tblock_{com}$ for semplicity

The Algorithm (parallel)

MACHINE $M_{9,9}$
•9 processing unit
•each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer 9 n/3 x n/3 block in time $\mathbf{tblock}_{\mathbf{com}}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,

The Algorithm (parallel)

MACHINE $M_{9,9}$
•9 processing unit
•each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer 9 n/3 x n/3 block in time $\mathbf{tblock}_{com}$, with the $rmem_i$ (reading) or $wmem_j$ (writing) operators,

If transfer is a communication between nodes, “reading” means “receiving” and “writing” means “sending”, so we call all the operator “transfers”, that is $trans_i$

The Algorithm (parallel)

MACHINE $M_{9,9}$
•9 processing unit
•each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer 9 n/3 x n/3 block in time $\mathbf{tblock}_{\mathbf{com}}$, with the $\mathbf{trans}_{\mathbf{i}}$ operators

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

We build the algorithm $ALG_{D,M9,9}$, containing 27 of this operators to execute 9 a time, and we have the **execution matrix** $E_{D,M9,9}$

$$E_{D,M9,9}=\begin{bmatrix} \otimes_0 & \otimes_1 & \otimes_2 & \otimes_3 & \otimes_4 & \otimes_5 & \otimes_6 & \otimes_7 & \otimes_8 \\ \otimes_9 & \otimes_{10} & \otimes_{11} & \otimes_{12} & \otimes_{13} & \otimes_{14} & \otimes_{15} & \otimes_{16} & \otimes_{17} \\ \otimes_{18} & \otimes_{19} & \otimes_{20} & \otimes_{21} & \otimes_{22} & \otimes_{23} & \otimes_{24} & \otimes_{25} & \otimes_{26} \end{bmatrix}$$

r_{E9}=3 rows

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

We build the algorithm $ALG_{D,M9,9}$, containing 27 of this operators to execute 9 a time, and we have the **execution matrix** $E_{D,M9,9}$

Perfectly parallel: no empty elements

$$E_{D,M9,9}=\begin{bmatrix} \otimes_0 & \otimes_1 & \otimes_2 & \otimes_3 & \otimes_4 & \otimes_5 & \otimes_6 & \otimes_7 & \otimes_8 \\ \otimes_9 & \otimes_{10} & \otimes_{11} & \otimes_{12} & \otimes_{13} & \otimes_{14} & \otimes_{15} & \otimes_{16} & \otimes_{17} \\ \otimes_{18} & \otimes_{19} & \otimes_{20} & \otimes_{21} & \otimes_{22} & \otimes_{23} & \otimes_{24} & \otimes_{25} & \otimes_{26} \end{bmatrix}$$

r_{E9}=3 rows

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

We build the algorithm $ALG_{D,M9,9}$, containing 27 of this operators to execute 9 a time, and we have the **execution matrix** $E_{D,M9,9}$

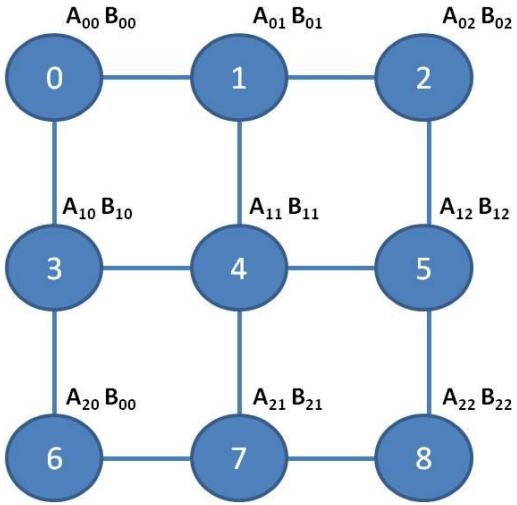
How does the algorithm work with datas and memory?

$$E_{D,M9,9}=\begin{bmatrix} \otimes_0 & \otimes & & & & & & & \\ \otimes_9 & \otimes_{10} & \otimes_{11} & \otimes_{12} & \otimes_{13} & \otimes_{14} & \otimes_{15} & \otimes_{16} & \otimes_{17} \\ \otimes_{18} & \otimes_{19} & \otimes_{20} & \otimes_{21} & \otimes_{22} & \otimes_{23} & \otimes_{24} & \otimes_{25} & \otimes_{26} \end{bmatrix}$$

E_9 – 3 rows

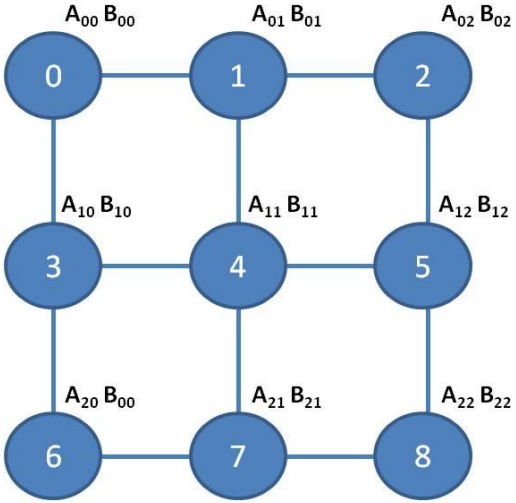
The Algorithm (parallel)

If we suppose that the execution units are logically distribuited in a 3x3 grid, at the beginning each unit has a n/3 x n/3 block of each matrix.



The Algorithm (parallel)

If we suppose that the execution units are logically distribuited in a 3x3 grid, at the beginning each unit has a n/3 x n/3 block of each matrix.
A possible $ALG_{D,M9,9}$, with **execution matrix** $E_{D,M9,9}$ is the well known BMR (Broadcast Multiply Rolling)



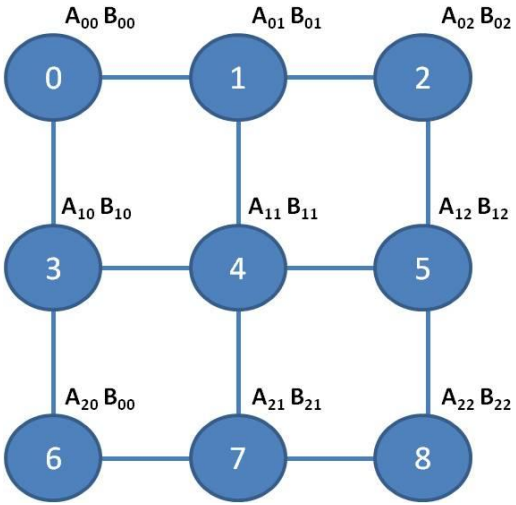
BMR Algorithm $ALG_{D,\mathcal{M}_{9,9}}$

```
for p:=1 to 3
  (1) nodes on the ith grid diagonal broadcast their A block
      to all its grid row
  (2) the node i solves  matmul(i*p)
  (3) if p<3, each node sends its B block to the upper one
      on the same column of the grid (rolling)
endfor
```

(matmul(p*i) is the subproblem $matmul^{p \cdot i}_{\frac{n}{3} \times \frac{n}{3}} \in D$)

The Algorithm (parallel)

If we suppose that the execution units are logically distribuited in a 3x3 grid, at the beginning each unit has a n/3 x n/3 block of each matrix.
A possible $ALG_{D,M9,9}$, with **execution matrix** $E_{D,M9,9}$ is the well known BMR (Broadcast Multiply Rolling)



BMR Algorithm $ALG_{D,\mathcal{M}_{9,9}}$

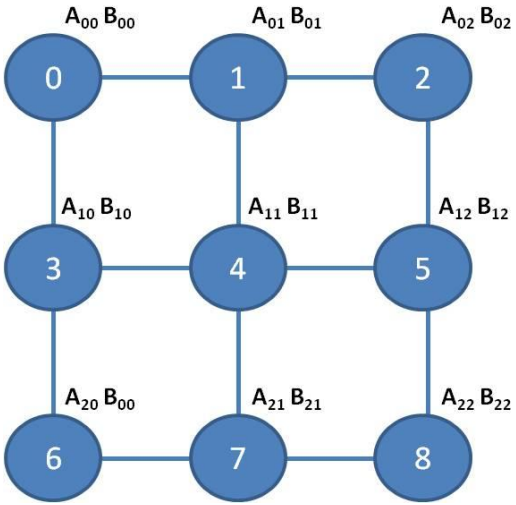
```
for p:=1 to 3
   (1) nodes on the ith grid diagonal broadcast their A block
       to all its grid row
   (2) the node i solves  matmul(i*p)
   (3) if p<3, each node sends its B block to the upper one
       on the same column of the grid (rolling)
endfor
```

(matmul(p*i) is the subproblem $matmul^{p \cdot i}_{\frac{n}{3} \times \frac{n}{3}} \in D$)

Broadcast is a sending and 8 receiving simultaneously, that is 9 transfers

The Algorithm (parallel)

If we suppose that the execution units are logically distribuited in a 33 gridS, at the beginning each unit has a n/3 x n/3 block of each matrix.
A possible $ALG_{D,M9,9}$, with **execution matrix** $E_{D,M9,9}$ is the well known BMR (Broadcast Multiply Rolling)



BMR Algorithm $ALG_{D,\mathcal{M}_{9,9}}$

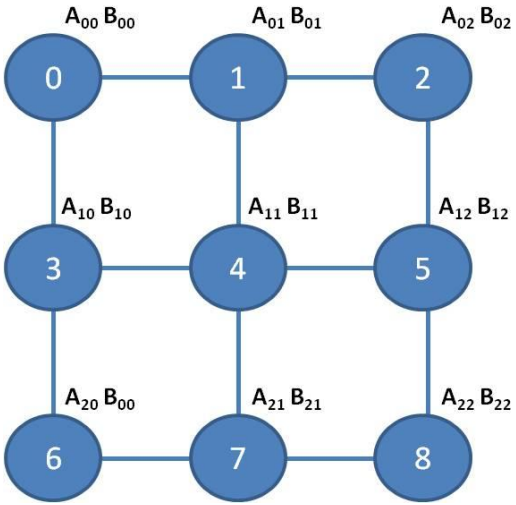
```
for p:=1 to 3
  (1) nodes on the ith grid diagonal broadcast their A block
      to all its grid row
  (2) the node i solves  matmul(i*p)
  (3) if p<3, each node sends its B block to the upper one
      on the same column of the grid (rolling)
endfor
```

(matmul(p*i) is the subproblem $matmul^{p \cdot i}_{\frac{n}{3} \times \frac{n}{3}} \in D$)

Rolling is 9 simultaneous sending followed by 9 simultaneous receiving that is 9 transfers depending on other 9 transfers

The Algorithm (parallel)

If we suppose that the execution units are logically distribuited in a 33 gridS, at the beginning each unit has a n/3 x n/3 block of each matrix.
A possible $ALG_{D,M9,9}$, with **execution matrix** $E_{D,M9,9}$ is the well known BMR (Broadcast Multiply Rolling)



BMR Algorithm $ALG_{D,\mathcal{M}_{9,9}}$

```
for p:=1 to 3
   (1) nodes on the ith grid diagonal broadcast their A block
       to all its grid row
   (2) the node i solves  matmul(i*p)
   (3) if p<3, each node sends its B block to the upper one
       on the same column of the grid (rolling)
endfor
```

(matmul(p*i) is the subproblem $matmul^{p \cdot i}_{\frac{n}{3} \times \frac{n}{3}} \in D$)

So, with 9 units so logically distribuited, for each p in [1,2] we will have 27 block transfers, and other 9 for p=3, that is 27*2+9= 63 total transfer operators

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

We implement the BMR in the software $SW_{D,M9,9}$, that will have the **memory matrix** $MEM_{D,M9,9}$

$$MEM_{D,M9,9} = \begin{bmatrix} trans_0(\cdot) & trans_1(\cdot) & trans_2(\cdot) & \dots & trans_8(\cdot) \\ trans_9(\cdot) & trans_{10}(\cdot) & trans_{11}(\cdot) & \dots & trans_{17}(\cdot) \\ trans_{18}(\cdot) & trans_{19}(\cdot) & trans_{20}(\cdot) & \dots & trans_{26}(\cdot) \\ trans_{27}(\cdot) & trans_{28}(\cdot) & trans_{29}(\cdot) & \dots & trans_{35}(\cdot) \\ trans_{36}(\cdot) & trans_{37}(\cdot) & trans_{38}(\cdot) & \dots & trans_{44}(\cdot) \\ trans_{45}(\cdot) & trans_{46}(\cdot) & trans_{47}(\cdot) & \dots & trans_{53}(\cdot) \\ trans_{54}(\cdot) & trans_{55}(\cdot) & trans_{56}(\cdot) & \dots & trans_{62}(\cdot) \end{bmatrix}$$

broadcast
rolling send
rolling receive
broadcast
rolling send
rolling receive
broadcast

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

We implement the BMR in the software $SW_{D,M9,9}$, that will have the **memory matrix** $MEM_{D,M9,9}$

$$MEM_{D,M9,9}= \begin{bmatrix} trans_0(\cdot) & trans_1(\cdot) & trans_2(\cdot) & \dots & trans_8(\cdot) \\ trans_9(\cdot) & trans_{10}(\cdot) & trans_{11}(\cdot) & \dots & trans_{17}(\cdot) \\ trans_{18}(\cdot) & trans_{19}(\cdot) & trans_{20}(\cdot) & \dots & trans_{26}(\cdot) \\ trans_{27}(\cdot) & trans_{28}(\cdot) & trans_{29}(\cdot) & \dots & trans_{35}(\cdot) \\ trans_{36}(\cdot) & trans_{37}(\cdot) & trans_{38}(\cdot) & \dots & trans_{44}(\cdot) \\ trans_{45}(\cdot) & trans_{46}(\cdot) & trans_{47}(\cdot) & \dots & trans_{53}(\cdot) \\ trans_{54}(\cdot) & trans_{55}(\cdot) & trans_{56}(\cdot) & \dots & trans_{62}(\cdot) \end{bmatrix}$$

r_{MEM9}=7 rows

The Algorithm (parallel)

MACHINE $M_{9,9}$
•9 processing unit
•each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
•2 memory levels,
•between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

The execution time of $ALG_{D,M9,9}$ will be

$$T(ALG_{D,\mathcal{M}_{9,9}}) = r_{E9} \cdot T_r = 3 \cdot C(\otimes) \cdot tflops$$

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

The execution time of $ALG_{D,M9,9}$ will be

$$T(ALG_{D,\mathcal{M}_{9,9}}) = r_{E9} \cdot T_r = 3 \cdot C(\otimes) \cdot tflops$$

Complexity of the operator (number of operations)

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

The execution time of $ALG_{D,M9,9}$ will be

$$T(ALG_{D,\mathcal{M}_{9,9}}) = r_{E9} \cdot T_r = 3 \cdot C(\otimes) \cdot tflops$$

Complexity of the operator (number of operations)

The memory access time of $SW_{D,M9,9}$ will be

$$T_M(Sw_{D,\mathcal{M}_{9,9}}) = r_{MEM9} \cdot tblock_{com} = 7 \cdot tblock_{com}$$

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

The execution time of $SW_{D,M9,9}$ will be

$$T(Sw_{D,\mathcal{M}_{9,9}}) = T(ALG_{D,\mathcal{M}_{9,9}}) + T_M(Sw_{D,\mathcal{M}_{9,9}})$$
$$= 3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}$$

The Algorithm (parallel)

MACHINE $M_{9,9}$
- 9 processing unit
- each processing unit is able to perform the (sequential) matrix-matrix multiply operator $\otimes$
- 2 memory levels,
- between the levels we can transfer 9 n/3 x n/3 block in time $tblock_{com}$, with the $trans_i$ operators

The execution time of $SW_{D,M9,9}$ will be

$$\begin{aligned} T(Sw_{D,\mathcal{M}_{9,9}}) &= T(ALG_{D,\mathcal{M}_{9,9}}) + T_M(Sw_{D,\mathcal{M}_{9,9}}) \\ &= 3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com} \end{aligned}$$

The speed up respect to $SW_{D,M1,1}$ (sequential one) will be

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

The Algorithm (parallel)

MACHIN
•9 proces
•each pro
•2 memo
•between

The exe

Could we improve this speed up without re-design all the software?

The speed up respect to $SW_{D,M1,1}$ (sequential one) will be

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

The Algorithm (parallel)

MACHIN
•9 proces
•each pro
•2 memo
•betweer

We can act on the basic operators on the parallel machine

The ex

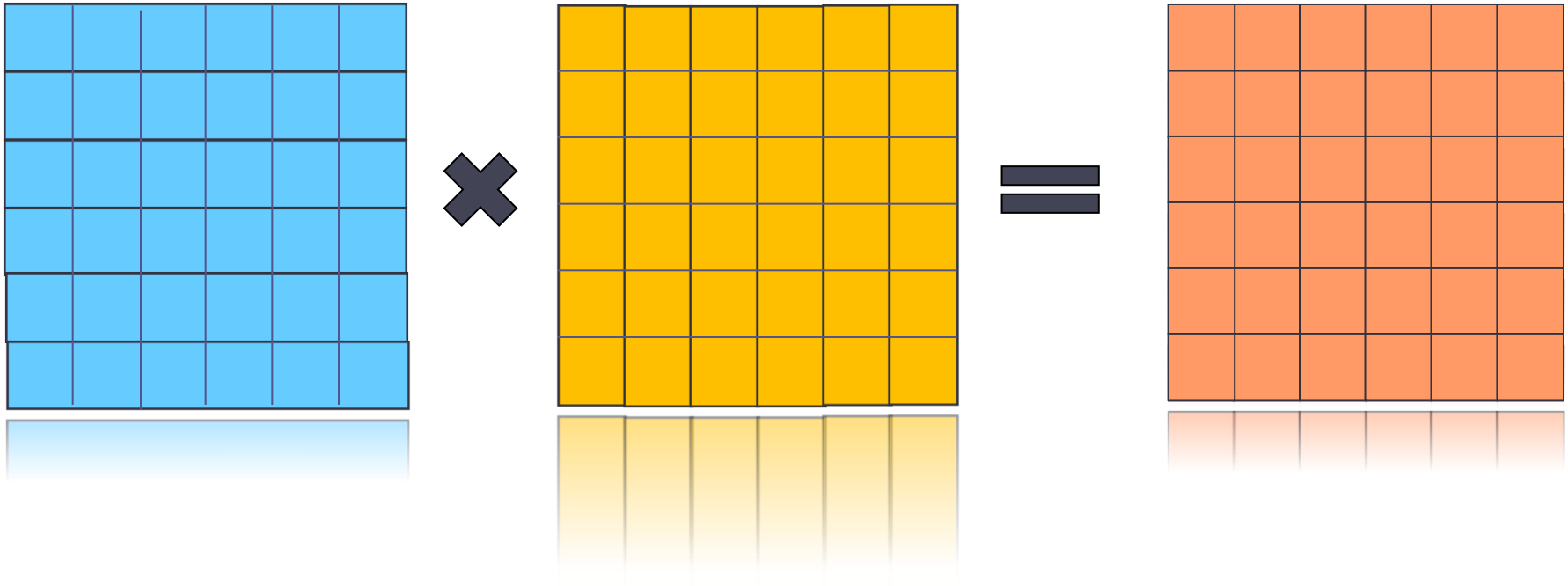
The speed up respect to $SW_{D,M1,1}$ (sequential one) will be

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

The subproblem

- A, B, C are n/3 x n/3 matrices

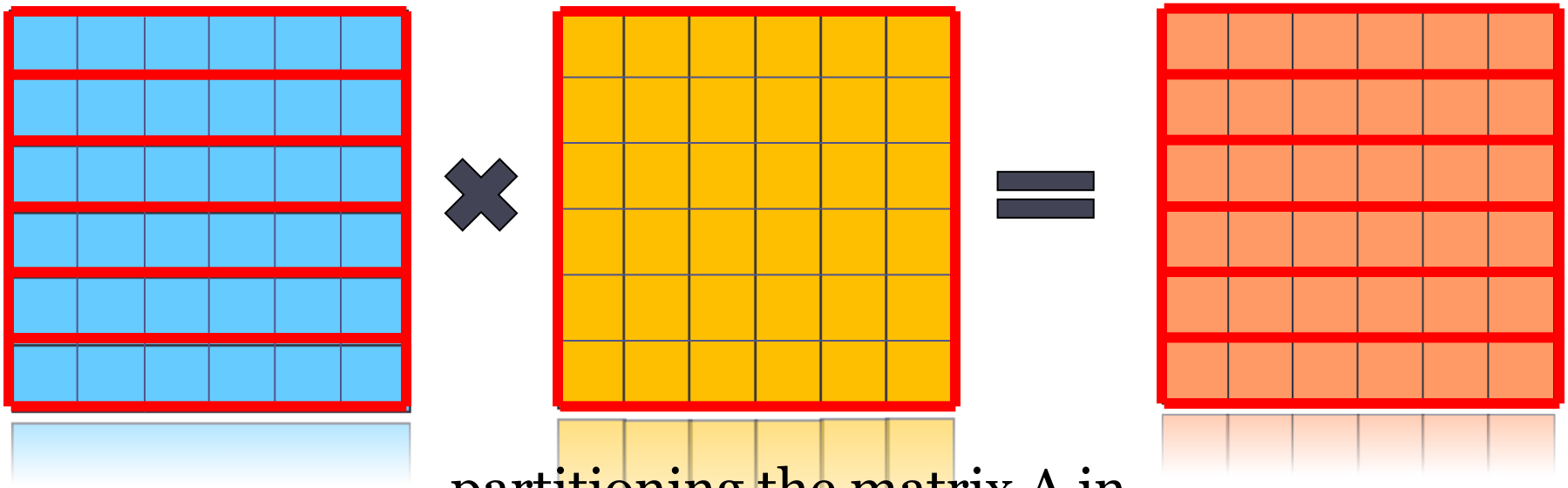
Operator $\otimes$ solves itself a matrix-matrix multiply problem, that is we can decompose it again.



The subproblem

- A, B, C are n/3 x n/3 matrices
- A and C are decomposed in n/3 vectors

Operator $\otimes$ solves itself a matrix-matrix multiply problem, that is we can decompose it again.

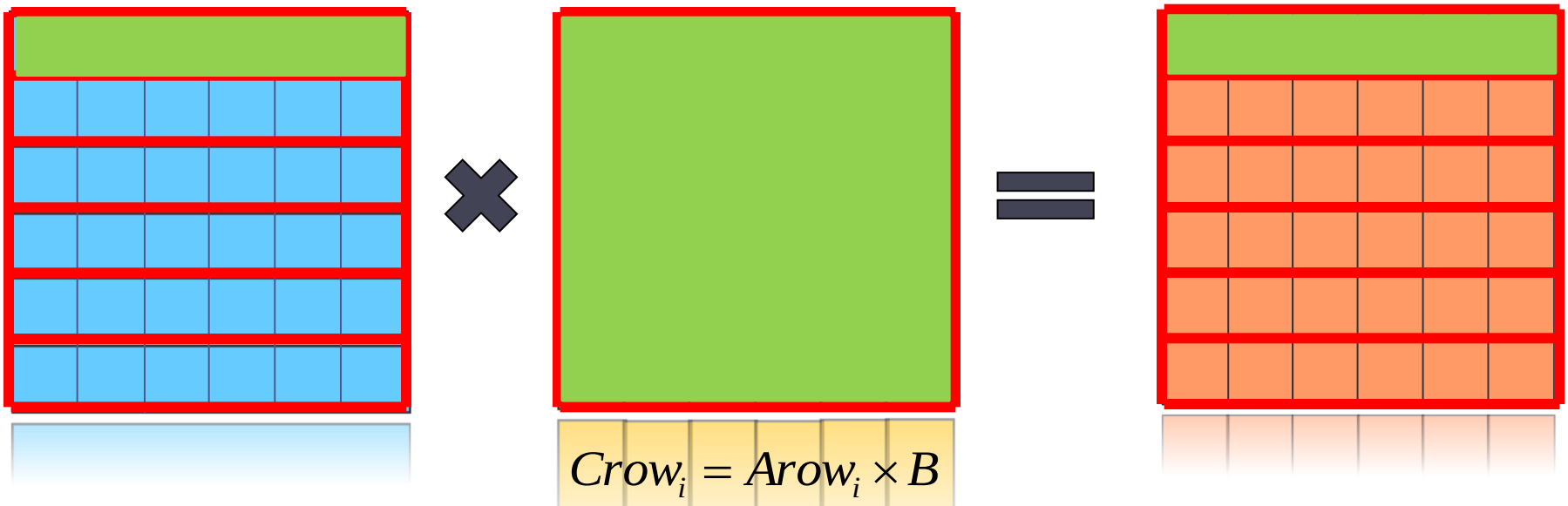


...partitioning the matrix A in subvectors

The subproblem

- A, B, C are n/3 x n/3 matrices
- A and C are decomposed in n/3 vectors

Operator $\otimes$ solves itself a matrix-matrix multiply problem, that is we can decompose it again.



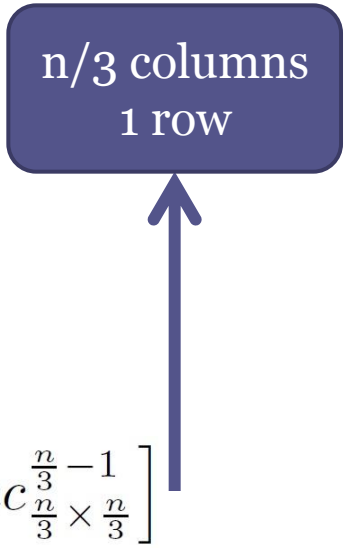
The subproblem

So we have the decomposition

$$D' = \{matvec^{i}_{\frac{n}{3}\times\frac{n}{3}}\}_{0\leq i<(\frac{n}{3}-1)}$$

where the subproblems are all independent from each other
And we have the **dependence matrix**

$$M_{D'} = \left[matvec^{0}_{\frac{n}{3}\times\frac{n}{3}}\ matvec^{1}_{\frac{n}{3}\times\frac{n}{3}}\ \ldots\ matvec^{\frac{n}{3}-1}_{\frac{n}{3}\times\frac{n}{3}}\right]$$



The subproblem

The problem AxB=C has

- Concurrency Degree: n/3
- Dependency Degree: 1

So we have the decomposition

$$D' = \{matvec^{i}_{\frac{n}{3}\times\frac{n}{3}}\}_{0\leq i<(\frac{n}{3}-1)}$$

where the subproblems are all independent from each other
And we have the **dependence matrix**

$$M_{D'} = \left[matvec^{0}_{\frac{n}{3}\times\frac{n}{3}}\ matvec^{1}_{\frac{n}{3}\times\frac{n}{3}}\ \ldots\ matvec^{\frac{n}{3}-1}_{\frac{n}{3}\times\frac{n}{3}}\right]$$

The subAlgorithm (sequential)

MACHINE $M'_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer a n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The subAlgorithm (sequential)

MACHINE $M'_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-vector multiply operator ⊠
- 2 memory levels,
- between the levels we can transfer a n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D',M'1,1}$ containing n/3 of this operators to execute one after another, and we have the **execution matrix** $E_{D',M'1,1}$ with

$r_{E1D'}$=n/3 rows

The subAlgorithm (sequential)

MACHINE $M'_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-vector multiply operator ⊠
- 2 memory levels,
- between the levels we can transfer a n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D',M'1,1}$ containing n/3 of this operators to execute one after another, and we have the **execution matrix** $E_{D',M'1,1}$ with

$r_{E1D'}$=n/3 rows

The subAlgorithm (sequential)

MACHINE $M'_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer a n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D',M'1,1}$ containing n/3 of this operators to execute one after another, and we have the **execution matrix** $E_{D',M'1,1}$ with

$r_{E1D'}$=n/3 rows

The related software $SW_{D',M'1,1}$ (that we can call $\otimes$) will have also a **memory matrix** $MEM_{D',M'1,1}$ with

$$r_{MEM1D'}=\left(\frac{n}{3}+2\right)\cdot\frac{n}{3} \text{ rows}$$

The subAlgorithm (sequential)

MACHINE $M'_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer a n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The execution time of $\otimes$ will be

$$T(\otimes) = T(ALG_{D',\mathcal{M}'_{1,1}}) + T_M(\otimes)$$
$$= \frac{n}{3} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{3} \cdot tvec_{mem}$$

The subAlgorithm (sequential)

MACHINE $M'_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer a n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The execution time of $\otimes$ will be

$$T(\otimes) = T(ALG_{D',\mathcal{M}'_{1,1}}) + T_M(\otimes)$$
$$= \frac{n}{3} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{3} \cdot tvec_{mem}$$

Complexity of the operator (number of operations)

The subAlgorithm (sequential)

MACHINE $M'_{1,1}$
- One processing unit
- the processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer a n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The execution time of $\otimes$ will be

$$T(\otimes) = \frac{n}{3} \cdot 2 \cdot \left(\frac{n}{3}\right)^2 \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{3} \cdot tvec_{mem}$$
$$= 2 \cdot \left(\frac{n}{3}\right)^3 \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{3} \cdot tvec_{mem}$$

The subAlgorithm (parallel)

This choise is to simulate a cluster node with 8 cores

MACHINE $M'_{8,8}$
- 8 processing unit
- each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The subAlgorithm (parallel)

MACHINE $M'_{8,8}$
- 8 processing unit
- each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D',M'8,8}$ containing n/3 of this operators to execute 8 a time, and we have the **execution matrix** $E_{D',M'1,1}$ with

$$r_{E8D'} = \frac{n}{3 \cdot 8} = \frac{n}{24} \text{ rows}$$

The subAlgorithm (parallel)

MACHINE $M'_{8,8}$
- 8 processing unit
- each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D',M'8,8}$ containing n/3 of this operators to execute 8 a time, and we have the **execution matrix** $E_{D',M'1,1}$ with

$$r_{E8D'} = \frac{n}{3 \cdot 8} = \frac{n}{24} \text{ rows}$$

If n/3 is multiple of 8, the matrix won't have empty elements, and the algorithm will be perfectly parallel

The subAlgorithm (parallel)

MACHINE $M'_{8,8}$
- 8 processing unit
- each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

We build the algorithm $ALG_{D',M'8,8}$ containing n/3 of this operators to execute 8 a time, and we have the **execution matrix** $E_{D',M'1,1}$ with

$$r_{E8D'} = \frac{n}{3 \cdot 8} = \frac{n}{24} \text{ rows}$$

If n/3 is multiple of 8, the matrix won't have empty elements, and the algorithm will be perfectly parallel

The related software $SW_{D',M'1,1}$ (that we can call $\otimes_{Par}$) will have also a **memory matrix** $MEM_{D',M'8,8}$ with

$$r_{MEM8D'} = \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \text{ rows}$$

The subAlgorithm (parallel)

MACHINE $M'_{8,8}$
- 8 processing unit
- each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The execution time of $\otimes_{Par}$ will be

$$T(\otimes_{Par}) = T(ALG_{D',\mathcal{M}'_{8,8}}) + T_M(\otimes_{Par})$$
$$= \frac{n}{24} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \cdot tvec_{mem}$$

The subAlgorithm (parallel)

MACHINE $M'_{8,8}$
- 8 processing unit
- each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The execution time of $\otimes_{Par}$ will be

$$T(\otimes_{Par}) = T(ALG_{D',\mathcal{M}'_{8,8}}) + T_M(\otimes_{Par})$$
$$= \frac{n}{24} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \cdot tvec_{mem}$$

Complexity of the operator (number of operations)

The subAlgorithm (parallel)

MACHINE $M'_{8,8}$
- 8 processing unit
- each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
- 2 memory levels,
- between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
- for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The execution time of $\otimes_{Par}$ will be

$$T(\otimes_{Par}) = \frac{n}{24} \cdot 2 \cdot \left(\frac{n}{3}\right)^2 \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \cdot tvec_{mem}$$

The subAlgorithm (parallel)

MACHINE $M'_{8,8}$
•8 processing unit
•each processing unit is able to perform the (sequential) matrix-vector multiply operator $\boxtimes$
•2 memory levels,
•between the levels we can transfer 8 n/3 vector in time $tvec_{mem}$, with reading and writing operators,
•for each execution step we need n/3+1 reading (before the execution) and a writing (after the execution)

The speed up respect to $\otimes$ (that is the sequential one) will be

$$Sp_{Sw}(\otimes_{Par}) = \frac{T(\otimes)}{T(\otimes_{Par})} = \frac{2 \cdot \left(\frac{n}{3}\right)^3 \cdot tflop + \left(\frac{n}{3}+2\right) \cdot \frac{n}{3} \cdot tvec_{mem}}{\frac{n}{24} \cdot 2 \cdot \left(\frac{n}{3}\right)^2 \cdot tflop + \left(\frac{n}{3}+2\right) \cdot \frac{n}{24} \cdot tvec_{mem}} > 1$$

Two levels of parallelism

What if we implement both the levels of parallelism to solve the first nxn problem

AxB=C

?

Two levels of parallelism

The speed up of $SW_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

Two levels of parallelism

The speed up of $SW_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

Sequential mat-mat operators

Two levels of parallelism

Sequential mat-mat operators

The speed up of $SW_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

Calling $SW'_{D,M9,9}$ the software where the operator $\otimes$ is substituted by $\otimes_{Par}$ the speed up of $SW'_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$\begin{aligned} Sp_{Sw}(Sw'_{D,\mathcal{M}_{9,9}}) &= \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw'_{D,\mathcal{M}_{9,9}})} \\ &= \frac{26 \cdot \left(\frac{n}{3} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{3} \cdot tvec_{mem}\right) + 52 \cdot tblock_{mem}}{3 \cdot \left(\frac{n}{24} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \cdot tvec_{mem}\right) + 7 \cdot tblock_{com}} \\ &> Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) \end{aligned}$$

Two levels of parallelism

Sequential mat-mat operators

The speed up of $SW_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

Calling $SW'_{D,M9,9}$ the software where the operator $\otimes$ is substituted by $\otimes_{Par}$ the speed up of $SW'_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$\begin{aligned} Sp_{Sw}(Sw'_{D,\mathcal{M}_{9,9}}) &= \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw'_{D,\mathcal{M}_{9,9}})} \\ &= \frac{26 \cdot \left(\frac{n}{3} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3}+2\right) \cdot \frac{n}{3} \cdot tvec_{mem}\right) + 52 \cdot tblock_{mem}}{3 \cdot \left(\frac{n}{24} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3}+2\right) \cdot \frac{n}{24} \cdot tvec_{mem}\right) + 7 \cdot tblock_{com}} \\ &> Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) \end{aligned}$$

Two levels of parallelism

The speed up of $SW_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw_{D,\mathcal{M}_{9,9}})} = \frac{26 \cdot C(\otimes) \cdot tflops + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tflops + 7 \cdot tblock_{com}}$$

Sequential mat-mat operators

Calling $SW'_{D,M9,9}$ the software where the operator $\otimes$ is substituted by $\otimes_{Par}$ the speed up of $SW'_{D,M9,9}$ respect to $SW_{D,M1,1}$ is

$$Sp_{Sw}(Sw'_{D,\mathcal{M}_{9,9}}) = \frac{T(Sw_{D,\mathcal{M}_{1,1}})}{T(Sw'_{D,\mathcal{M}_{9,9}})}$$

$$= \frac{26 \cdot \left(\frac{n}{3} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{3} \cdot tvec_{mem}\right) + 52 \cdot tblock_{mem}}{3 \left(\frac{n}{24} \cdot C(\boxtimes) \cdot tflop + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \cdot tvec_{mem}\right) + 7 \cdot tblock_{com}}$$

$$> Sp_{Sw}(Sw_{D,\mathcal{M}_{9,9}})$$

Parallel mat-mat operators

Conclusions

Summarize: What did we do?

1. We started supposing a problem decomposed for a machine like a cluster, and we choosen a particular decomposition, that is in 9 submatrices.

Conclusions

Summarize: What did we do?

2. Using the model and the procedure that the model lead to,
   - we built a sequential algorithm for a machine with 1 processing unit and then a parallel one for a cluster of 9 nodes, that need to communicate
     - We observed easily a speed up that considers both execution and memory access (communication) time and shows very precisely the gain (when machine parameters are known)

Conclusions

Summarize: What did we do?

3. We supposed that the nodes of the cluster have their own parallel architecture, that is they are multicore, and focused on the work of a single node: the subproblem it solves
   - we built a sequential algorithm for a machine with 1 processing unit and then a parallel one for a multicore with 8 cores, that access the memory concurrently
     - We observed easily another speed up at this second level

Conclusions

Summarize: What did we do?

4. We mixed the two design
   - we built an algorithm that meet all the <<cluster of multicore>> possibilities
     - We observed easily the speed up respect to the completely sequential one
     - If you know the parameters that characterize the machine, the resut is a prediction very closed to reality, and all the bonds and limits are highlight.

Conclusions

What can we do?

1. We can substitute in the ratio many combinations of the modules we explosed, to analyze different gains with the same machine

2. We can change the dimension of the problem
   - we should just change n in the procedure

Conclusions

What can we do?

3. We can change the starting decomposition
   - It will lead just different dependency, execution and memory matrices with different numbers of rows
4. We can change the machine characteristics
   - It will lead just different execution and memory matrices with different numbers of rows
5. We can analyze other levels of parallelism
   - The procedure is recursive: we can build matrices for each operator involved, to explore the parallelism possibilities and benefits

At the end

We are still working on the model to extend it, and explore other details.

We are also developing new applications using this formal point of view that leads a structured increment of levels of parallelism, with good control of the performance change.

For now…

THANK YOU FOR YOUR ATTENTION