

# Parallel SpMV and Cartesian Distribution

## (PSC §4.4)

original slides by Rob Bisseling, Universiteit Utrecht,  
accompanying the textbook *Parallel Scientific Computation*

adapted for the lecture *HPC – Algorithms and Applications*,  
Michael Bader, TUM, WS 2016/17



# 1D and 2D processor numbering – Cartesian distribution

- ▶ Use 2D processor numbering, e.g.,  $P(0, 0)$ ,  $P(1, 0)$ , etc.
- ▶ Natural **column-wise identification** for  $p = MN$  processors:

$$P(s, t) \equiv P(s + tM), \text{ for } 0 \leq s < M \text{ and } 0 \leq t < N.$$

- ▶ This can also be written as

$$P(s) \equiv P(s \bmod M, s \operatorname{div} M), \text{ for } 0 \leq s < p.$$

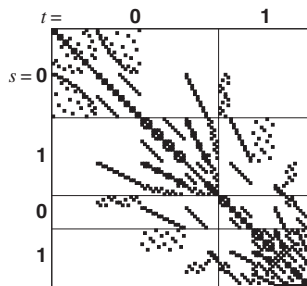
- ▶ For a **Cartesian distribution**  $(\phi_0, \phi_1)$ , we map nonzeros  $a_{ij}$  to processors  $P(\phi(i, j))$  by

$$\phi(i, j) = \phi_0(i) + \phi_1(j)M, \text{ for } 0 \leq i, j < n \text{ and } a_{ij} \neq 0.$$

- ▶ We use 1D or 2D numbering, whichever is more convenient.



## Example: Cartesian distribution of cage6



$n = 93$ ,  $nz = 785$ ,  $p = 4$ ,  $M = N = 2$ .

- ▶ The processor row of a matrix element  $a_{ij}$  is  $s = \phi_0(i)$ ; the processor column is  $t = \phi_1(j)$ .
- ▶ Matrix diagonal assigned in blocks to processors  
 $P(0) \equiv P(0, 0)$ ,  $P(1) \equiv P(1, 0)$ ,  $P(2) \equiv P(0, 1)$ ,  
 $P(3) \equiv P(1, 1)$ .



# Advantages of a Cartesian distribution

## Advantages:

- ▶ Main advantage for sparse matrices (same for dense matrices): row-wise operations require communication only within processor rows. (Similar for columns.)
- ▶ Vector component  $v_j$  has to be sent to **at most  $M$**  processors, and vector component  $u_i$  is computed using contributions received from **at most  $N$**  processors.
- ▶ Simplicity: Cartesian distributions partition the matrix orthogonally into **rectangular submatrices**. Non-Cartesian distributions create **arbitrarily-shaped matrix parts**.

## Disadvantage:

- ▶ Less general, so may not offer the optimal solution.



Cartesian distribution

## Matching matrix and vector distribution

- ▶ Vector component  $v_j$  is needed only by processors that possess an  $a_{ij} \neq 0$ , and these processors are contained in **processor column**  $P(*, \phi_1(j))$ .
- ▶ Assigning vector component  $v_j$  to one of the processors in  $P(*, \phi_1(j))$  implies that  $v_j$  has to be sent to **at most**  $M - 1$  processors, instead of  $M$ .
- ▶ If we are lucky (or clever), we may even avoid communication of  $v_j$  altogether.
- ▶ If  $v_j$  were assigned to a different processor column, it would always have to be communicated.
- ▶ Assigning  $u_i$  to a processor in processor row  $P(\phi_0(i), *)$  reduces the number of contributions sent for  $u_i$  to **at most**  $N - 1$ .



# A trivial but powerful theorem

**Theorem 4.4** Let  $A$  be a sparse  $n \times n$  matrix and  $\mathbf{u}, \mathbf{v}$  vectors of length  $n$ . Assume that:

1. distribution of  $A$  is Cartesian,  $\text{distr}(A) = (\phi_0, \phi_1)$ ;
2. distribution of  $\mathbf{u}$  is such that  $u_i$  resides in  $P(\phi_0(i), *)$ ;
3. distribution of  $\mathbf{v}$  is such that  $v_j$  resides in  $P(*, \phi_1(j))$ .

Then: if  $u_i$  and  $v_j$  are assigned to the same processor,  $a_{ij}$  is also assigned to that processor and does not cause communication.

**Proof** Component  $u_i$  is assigned to  $P(\phi_0(i), t)$ . Component  $v_j$  to  $P(s, \phi_1(j))$ . Since this is the same processor, we have  $(s, t) = (\phi_0(i), \phi_1(j))$ , so that this processor also owns  $a_{ij}$ .  $\square$



## Special case $\text{distr}(\mathbf{u}) = \text{distr}(\mathbf{v})$

The conditions

1. distribution of  $A$  is Cartesian,  $\text{distr}(A) = (\phi_0, \phi_1)$ ;
2. distribution of  $\mathbf{u}$  is such that  $u_i$  resides in  $P(\phi_0(i), *)$ ;
3. distribution of  $\mathbf{v}$  is such that  $v_j$  resides in  $P(*, \phi_1(j))$ ;
4.  $\text{distr}(\mathbf{u}) = \text{distr}(\mathbf{v})$ ;

imply that  $u_i$  and  $v_i$  are assigned to  $P(\phi_0(i), \phi_1(i))$ , which is the owner of the **diagonal element**  $a_{ii}$ .

- ▶ For a fixed  $M$  and  $N$ , the choice of a Cartesian matrix distribution determines the vector distribution.
- ▶ The reverse is also true.







# Vector distribution for tridiagonal matrix

- ▶  $a_{ij} \neq 0$  if and only if  $i - j = 0, \pm 1$ .
- ▶ Assume we require  $\text{distr}(\mathbf{u}) = \text{distr}(\mathbf{v})$ . Theorem 4.4 says that it is best to assign  $u_i$  and  $v_j$  (and hence  $u_j$ ) to the same processor if  $i = j \pm 1$ .
- ▶ Therefore, a suitable vector distribution over  $p$  processors is the **block distribution**,

$$u_i \mapsto P\left(i \operatorname{div} \left\lceil \frac{n}{p} \right\rceil\right), \text{ for } 0 \leq i < n.$$



## Example: $12 \times 12$ 1D Laplacian matrix

Distribution matrix for  $n = 12$  and  $M = N = 2$ :

$$\text{distr}(A) = \begin{bmatrix} 0 & 0 & & & & & & & & & & \\ 0 & 0 & 0 & & & & & & & & & \\ & 0 & 0 & 0 & & & & & & & & \\ & & 1 & 1 & 1 & & & & & & & \\ & & & 1 & 1 & 1 & & & & & & \\ & & & & 1 & 1 & 3 & & & & & \\ & & & & & 0 & 2 & 2 & & & & \\ & & & & & & 2 & 2 & 2 & & & \\ & & & & & & & 2 & 2 & 2 & & \\ & & & & & & & & 3 & 3 & 3 & \\ & & & & & & & & & 3 & 3 & 3 \\ & & & & & & & & & & 3 & 3 \end{bmatrix}.$$



## Example: $12 \times 12$ 1D Laplacian matrix (cont'd)

Position  $(i, j)$  of  $\text{distr}(A)$  gives 1D identity of the processor that owns matrix element  $a_{ij}$ ;  $\text{distr}(A)$  is obtained by:

- ▶ distributing the vectors by the 1D block distribution
- ▶ distributing the matrix diagonal in the same way as the vectors
- ▶ translating the 1D processor numbers into 2D numbers by  $P(0) \equiv P(0, 0)$ ,  $P(1) \equiv P(1, 0)$ ,  $P(2) \equiv P(0, 1)$ ,  $P(3) \equiv P(1, 1)$ .
- ▶ determining the owners of the off-diagonal nonzeros:  $a_{56}$  is in the **same processor row** as  $a_{55}$ , owned by  $P(1) = P(1, 0)$ ; it is in the **same processor column** as  $a_{66}$ , owned by  $P(2) = P(0, 1)$ . Thus,  $a_{56}$  is owned by  $P(1, 1) = P(3)$ .



Cartesian distribution

## Cost analysis – Cartesian distribution

Assuming a good spread of **nonzeros** and **vector components** over processors, **matrix rows** over processor rows, **matrix columns** over processor columns:

$$T_{(0)} = (M - 1) \frac{ng}{p} + l,$$

$$T_{(1)} = \frac{2cn}{p} + l,$$

$$T_{(2)} = (N - 1) \frac{ng}{p} + l,$$

$$T_{(3)} = \frac{Nn}{p} + l.$$

$$T_{MV, M \times N} \leq \frac{2cn}{p} + \frac{n}{M} + \frac{M + N - 2}{p} ng + 4l.$$



## Efficient computation for $M = N = \sqrt{p}$

$$T_{MV, \sqrt{p} \times \sqrt{p}} \leq \frac{2cn}{p} + \frac{n}{\sqrt{p}} + 2 \left( \frac{1}{\sqrt{p}} - \frac{1}{p} \right) ng + 4l.$$

- ▶ Computation is **efficient** if  $\frac{2cn}{p} > \frac{2ng}{\sqrt{p}}$ , i.e.,  $c > \sqrt{p}g$ .
- ▶ Improvement of factor  $\sqrt{p}$  compared to previous general efficiency criterion.



# Dense matrices

- ▶ Dense matrices are the **limit** of sparse matrices for  $c \rightarrow n$ .
- ▶ Analysing the dense case is easier and it can give us insight into the sparse case as well.
- ▶ Substituting  $c = n$  in previous cost formula gives

$$T_{\text{MV, dense}} \leq \frac{2n^2}{p} + \frac{n}{\sqrt{p}} + 2 \left( \frac{1}{\sqrt{p}} - \frac{1}{p} \right) ng + 4l.$$

- ▶ All spreading assumptions must hold.
- ▶ **Which distribution** will yield this cost?



Cartesian distribution

## Block/Square cyclic distribution? No!

- ▶ Block cyclic (here in particular square blocks  $\rightarrow$  square cyclic) distribution useful for LU decomposition and all parallel linear algebra.
- ▶ Diagonal element  $a_{ij}$  is assigned to  $P(i \bmod \sqrt{p}, i \bmod \sqrt{p})$ , so that the matrix diagonal is assigned to the **diagonal processors**  $P(s, s)$ ,  $0 \leq s < \sqrt{p}$ .
- ▶ Only  $\sqrt{p}$  processors have part of the matrix diagonal and the vectors.
- ▶ Recall: if  $\text{distr}(\mathbf{u}) = \text{distr}(\mathbf{v})$ , then  $u_i$  and  $v_i$  are assigned to  $P(\phi_0(i), \phi_1(i))$ , which is the owner of the **diagonal element**  $a_{ij}$ .
- ▶ Thus: vectors only distributed to  $\sqrt{p}$  processors
- ▶ “Diagonal” processors therefore must send  $\sqrt{p} - 1$  copies of  $\frac{n}{\sqrt{p}}$  vector components:  $h_s = n - \frac{n}{\sqrt{p}}$  in (0).
- ▶ The total cost for the square cyclic distribution is

$$T_{\text{MV, dense, } \sqrt{p} \times \sqrt{p} \text{ cyclic}} = \frac{2n^2}{p} + n + 2 \left( 1 - \frac{1}{\sqrt{p}} \right) ng + 4l.$$

Cartesian distribution



## Cyclic row distribution? No!

- ▶ Communication balance can be improved by choosing a distribution that spreads the matrix diagonal evenly,  $\phi_{\mathbf{u}}(i) = \phi_{\mathbf{v}}(i) = i \bmod p$ , and translating from 1D to 2D.
- ▶ We still have the freedom to choose  $M$  and  $N$ , where  $MN = p$ . For the choice  $M = p$  and  $N = 1$ , this gives the **cyclic row distribution**  $\phi_0(i) = i \bmod p$  and  $\phi_1(j) = 0$ .
- ▶ The total cost for the cyclic row distribution is

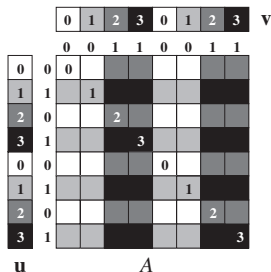
$$T_{\text{MV, dense, } p \times 1 \text{ cyclic}} = \frac{2n^2}{p} + \left(1 - \frac{1}{p}\right) ng + 2l.$$

- ▶ This distribution skips supersteps (2) and (3), since each matrix row is completely contained in one processor.
- ▶ The trouble is that the fanout is very expensive: each processor has to send  $\frac{n}{p}$  vector components to all others.





# Square Cartesian distribution? Yes!



$n = 8$ ,  $p = 4$ ,  $M = N = 2$ . Square Cartesian distribution based on a cyclic distribution of the matrix diagonal.

- ▶ We take the same distribution method,  $\phi_{\mathbf{u}}(i) = \phi_{\mathbf{v}}(i) = i \bmod p$ , but now we choose  $M = N = \sqrt{p}$  when translating from 1D to 2D.
- ▶ *Et voilà!* We achieve the optimal BSP cost.



# Summary

- ▶ For Cartesian distributions, we use both 1D and 2D processor numberings to our advantage, with the identification  $P(s, t) \equiv P(s + tM)$ .
- ▶ We have seen the example of a **tridiagonal matrix**, where we obtained a 2D matrix distribution, slightly different from a 1D block row distribution. For **band matrices** with a wider band, this may be advantageous.
- ▶ A square Cartesian matrix distribution based on a **cyclic distribution of the matrix diagonal and the input and output vectors** is an optimal data distribution for dense matrices and for sparse matrices that are relatively dense.
- ▶ There exist other optimal data distributions, e.g. based on a block distribution of the matrix diagonal.

