High Performance Numerical Linear Algebra

Victor Eijkhout

- Dense matrix-vector product

- Sparse matrix-vector product

- Nested dissection

- Parallel preconditioners

Dense matrix-vector product

Parallel matrix-vector product; dense

- Assume a division by block rows
- Every processor p has a set of row indices I_p

Mvp on processor p:

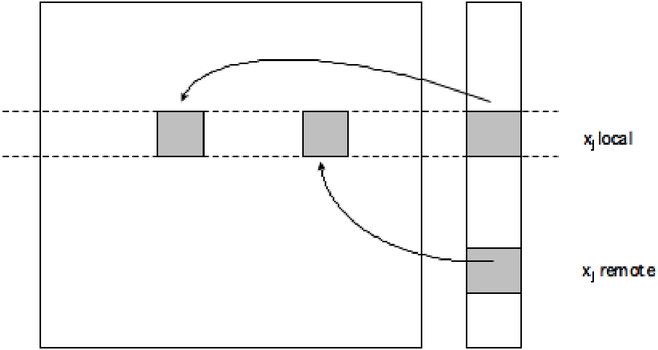
$$\forall_i \colon y_i = \sum_j a_{ij} x_j$$

$$\forall_i \colon y_i = \sum_q \sum_{j \in I_q} a_{ij} x_j$$

Local and remote parts:

$$\forall_i \colon y_i = \sum_{j \in I_p} a_{ij} x_j + \sum_{q \neq p} \sum_{j \in I_q} a_{ij} x_j$$

Local part I_p can be executed right away, I_q requires communication.



Combine:

Input: Processor number p; the elements x_i with $i \in I_p$; matrix elements A_{ij} with $i \in I_p$.
Output: The elements y_i with $i \in I_p$

for $q \neq p$ **do**
  Send elements of x from processor q to p, receive in buffer B_{pq}.
end
$y_{local} \leftarrow Ax_{local}$
for $q \neq p$ **do**
  $y_{local} \leftarrow y_{local} + A_{pq}B_q$
end

Procedure Parallel MVP$(A, x_{local}, y_{local}, p)$

Note possible overlap communication and computation

Cost computation 1.

Algorithm:

Step	Cost (lower bound)
Allgather x_i so that x is available on all nodes	
Locally compute $y_i = A_i x$	$\approx 2\frac{n^2}{P}\gamma$

Allgather

Assume that data arrives over a binary tree:

- latency $\alpha \log_2 P$
- transmission time, receiving n/P elements from $P-1$ processors

Algorithm with cost:

Step	Cost (lower bound)
Allgather x_i so that x is available on all nodes	$\lceil\log_2(P)\rceil\alpha + \frac{P-1}{P}n\beta \approx \log_2(P)\alpha + n\beta$
Locally compute $y_i = A_i x$	$\approx 2\frac{n^2}{P}\gamma$

Parallel efficiency

$$E_p^{\text{1D-row}}(n) = \frac{S_p^{\text{1D-row}}(n)}{p} = \frac{1}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{p}{2n}\frac{\beta}{\gamma}}.$$

Strong scaling, weak scaling?

Two-dimensional partitioning

x_0				x_3				x_6				x_9	
a_{00}	a_{01}	a_{02}	y_0	a_{03}	a_{04}	a_{05}		a_{06}	a_{07}	a_{08}		a_{09}	a_0
a_{10}	a_{11}	a_{12}		a_{13}	a_{14}	a_{15}	y_1	a_{16}	a_{17}	a_{18}		a_{19}	a_1
a_{20}	a_{21}	a_{22}		a_{23}	a_{24}	a_{25}		a_{26}	a_{27}	a_{28}	y_2	a_{29}	a_2
a_{30}	a_{31}	a_{32}		a_{33}	a_{34}	a_{35}		a_{37}	a_{37}	a_{38}		a_{39}	a_3
	x_1				x_4				x_7				x
a_{40}	a_{41}	a_{42}	y_4	a_{43}	a_{44}	a_{45}		a_{46}	a_{47}	a_{48}		a_{49}	a_4
a_{50}	a_{51}	a_{52}		a_{53}	a_{54}	a_{55}	y_5	a_{56}	a_{57}	a_{58}		a_{59}	a_5
a_{60}	a_{61}	a_{62}		a_{63}	a_{64}	a_{65}		a_{66}	a_{67}	a_{68}	y_6	a_{69}	a_6
a_{70}	a_{71}	a_{72}		a_{73}	a_{74}	a_{75}		a_{77}	a_{77}	a_{78}		a_{79}	a_7
		x_2				x_5				x_8			
a_{80}	a_{81}	a_{82}	y_8	a_{83}	a_{84}	a_{85}		a_{86}	a_{87}	a_{88}		a_{89}	a_8
a_{90}	a_{91}	a_{92}		a_{93}	a_{94}	a_{95}	y_9	a_{96}	a_{97}	a_{98}		a_{99}	a_9
$a_{10,0}$	$a_{10,1}$	$a_{10,2}$		$a_{10,3}$	$a_{10,4}$	$a_{10,5}$		$a_{10,6}$	$a_{10,7}$	$a_{10,8}$	y_{10}	$a_{10,9}$	a_{10}
$a_{11,0}$	$a_{11,1}$	$a_{11,2}$		$a_{11,3}$	$a_{11,4}$	$a_{11,5}$		$a_{11,7}$	$a_{11,7}$	$a_{11,8}$		$a_{11,9}$	a_{11}

Algorithm

- Collecting x_j on each processor p_{ij} by an *allgather* inside the processor columns.
- Each processor p_{ij} then computes $y_{ij} = A_{ij}x_j$.
- Gathering together the pieces y_{ij} in each processor row to form y_i, distribute this over the processor row: combine to form a *reduce-scatter*.
- Setup for the next A or A^t product

Analysis 1.

Step	Cost (lower bound)
Allgather x_i's within columns	$\lceil \log_2(r) \rceil \alpha + \frac{r-1}{p} n\beta \approx \log_2(r)\alpha + \frac{n}{c}\beta$
Perform local matrix-vector multiply	$\approx 2\frac{n^2}{p}\gamma$
Reduce-scatter y_i's within rows	

Reduce-scatter

	$t=1$	$t=2$	$t=3$
p_0	$x_0^{(0)}, x_1^{(0)}, x_2^{(0)}\downarrow, x_3^{(0)}\downarrow$	$x_0^{(0:2:2)}, x_1^{(0:2:2)}\downarrow$	$x_0^{(0:3)}$
p_1	$x_0^{(1)}, x_1^{(1)}, x_2^{(1)}\downarrow, x_3^{(1)}\downarrow$	$x_0^{(1:3:2)}\uparrow, x_1^{(1:3:2)}$	$x_1^{(0:3)}$
p_2	$x_0^{(2)}\uparrow, x_1^{(2)}\uparrow, x_2^{(2)}, x_3^{(2)}$	$x_2^{(0:2:2)}, x_3^{(0:2:2)}\downarrow$	$x_2^{(0:3)}$
p_3	$x_0^{(3)}\uparrow, x_1^{(3)}\uparrow, x_2^{(3)}, x_3^{(3)}$	$x_0^{(1:3:2)}\uparrow, x_1^{(1:3:2)}$	$x_3^{(0:3)}$

Time:

$$\lceil \log_2 p \rceil \alpha + \frac{p-1}{p} n(\beta+\gamma).$$

Step	Cost (lower bound)
Allgather x_i's within columns	$\lceil \log_2(r) \rceil \alpha + \frac{r-1}{p} n\beta \approx \log_2(r)\alpha + \frac{n}{c}\beta$
Perform local matrix-vector multiply	$\approx 2\frac{n^2}{p}\gamma$
Reduce-scatter y_i's within rows	$\lceil \log_2(c) \rceil \alpha + \frac{c-1}{p} n\beta + \frac{c-1}{p} n\gamma \approx \log_2(r)\alpha + \frac{n}{c}\beta + \frac{n}{c}\gamma$

Efficiency

$$E_p^{\sqrt{p}\times\sqrt{p}}(n) = \frac{1}{1 + \frac{p\log_2(p)}{2n^2}\frac{\alpha}{\gamma} + \frac{\sqrt{p}}{2n}\frac{(2\beta+\gamma)}{\gamma}}$$

Weak scaling:
for $p \to \infty$ this is $\approx 1/\log_2 P$:
only slowly decreasing.

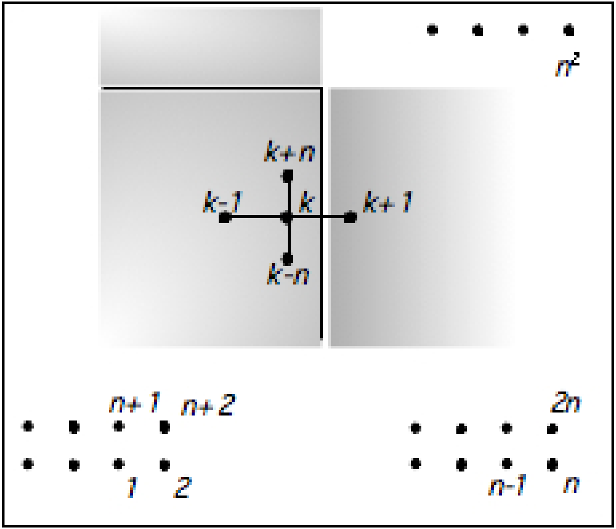
LU factorization scaling

Needs a cyclic distribution

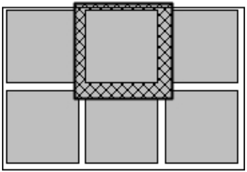
Sparse matrix-vector product

Sparse matrix-vector product

Difference stencil



induces ghost region:



Limited number of neighbours, limited buffer space

Scaling

Separately 1D and 2D partitioning of the domain.

Nested dissection

Fill-in during LU

Fill-in: index (i,j) where $a_{ij} = 0$ but $\ell_{ij} \neq 0$ or $u_{ij} \neq 0$.

2D BVP: Ω is $n \times n$, gives matrix of size $N = n^2$, with bandwidth n.

Matrix storage $O(N)$

LU storage $O(N^{3/2})$

LU factorization work $O(N^2)$

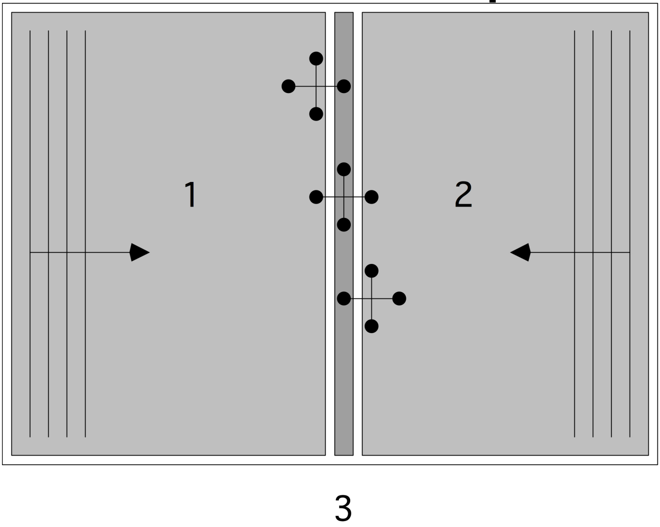
Cute fact: storage can be computed linear in #nonzeros

Fill-in is a function of ordering

$$\begin{pmatrix} * & * & \cdots & * \\ * & * & & \emptyset \\ \vdots & & \ddots & \\ * & \emptyset & & * \end{pmatrix}$$

After factorization the matrix is dense.
Can this be permuted?

Domain decomposition



$$
\left(\begin{array}{ccccc|ccccc|c}
\star & \star & & & & & & & & & 0 \\
\star & \star & \star & & & & & & & & \vdots \\
 & \ddots & \ddots & \ddots & & & & \emptyset & & & \vdots \\
 & & \star & \star & \star & & & & & & 0 \\
 & & & \star & \star & & & & & & \star \\
\hline
 & & & & & \star & \star & & & & 0 \\
 & & & & & \star & \star & \star & & & \vdots \\
 & & \emptyset & & & & \ddots & \ddots & \ddots & & \vdots \\
 & & & & & & & \star & \star & \star & 0 \\
 & & & & & & & & \star & \star & \star \\
\hline
0 & \cdots & \cdots & 0 & \star & 0 & \cdots & \cdots & 0 & \star & \star
\end{array}\right)
\qquad
\begin{array}{l}
\left.\vphantom{\begin{array}{c}a\\a\\a\\a\\a\end{array}}\right\}\ (n^2-n)/2 \\
\left.\vphantom{\begin{array}{c}a\\a\\a\\a\end{array}}\right\}\ (n^2-n)/2 \\
\left.\vphantom{a}\right\}\ n
\end{array}
$$

DD factorization

$$A^{\mathrm{DD}} = \begin{pmatrix} A_{11} & \emptyset & A_{13} \\ \emptyset & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} =$$

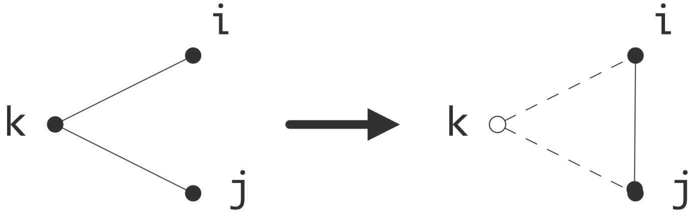
$$\begin{pmatrix} I & & \\ \emptyset & I & \\ A_{31}A_{11}^{-1} & A_{32}A_{22}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & \emptyset & A_{13} \\ & A_{22} & A_{23} \\ & & S \end{pmatrix}$$

$$S = A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23}$$

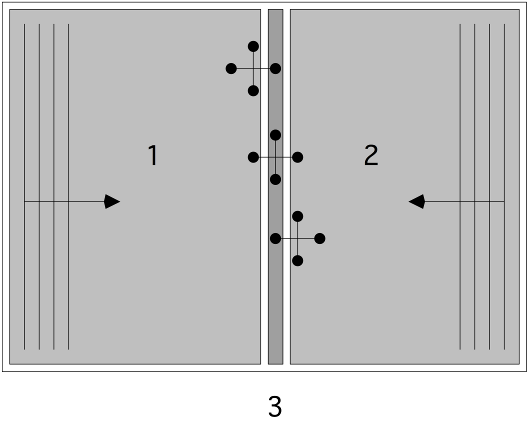
Parallelism...

Graph theory of sparse elimination

$$a_{ij} \leftarrow a_{ij} - a_{ik} a_{kk}^{-1} a_{kj}$$

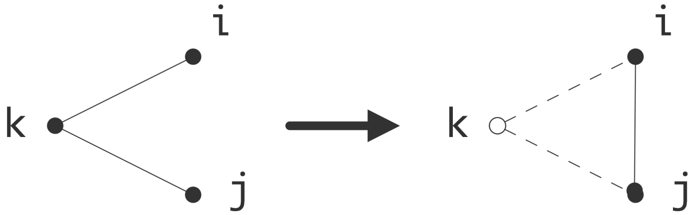


Graph theory of sparse elimination



Graph theory of sparse elimination

$$a_{ij} \leftarrow a_{ij} - a_{ik} a_{kk}^{-1} a_{kj}$$



So inductively S is dense

Recursive bisection

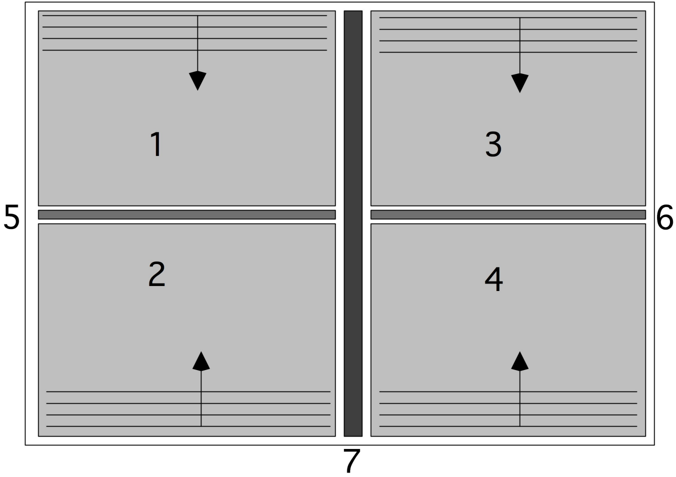
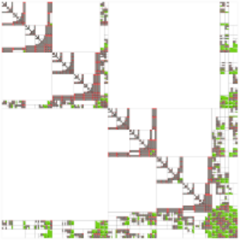


Figure : A four-way domain decomposition

$$A^{\mathrm{DD}} = \begin{pmatrix} A_{11} & & & & A_{15} & & A_{17} \\ & A_{22} & & & A_{25} & & A_{27} \\ & & A_{33} & & & A_{36} & A_{37} \\ & & & A_{44} & & A_{46} & A_{47} \\ A_{51} & A_{52} & & & A_{55} & & A_{57} \\ & & A_{63} & A_{64} & & A_{66} & A_{67} \\ A_{71} & A_{72} & A_{73} & A_{74} & A_{75} & A_{76} & A_{77} \end{pmatrix}$$

The domain/operator/graph view is more insightful, don't you think?

How does this look in reality?



Complexity

With $n = \sqrt{N}$:

- one dense matrix on a separator of size n, plus
- two dense matrices on separators of size $n/2$
- $\rightarrow 3/2\, n^2$ space and $5/12\, n^3$ time
- and then four times the above with $n \rightarrow n/2$

$$\begin{aligned}\text{space} &= 3/2n^2 + 4 \cdot 3/2(n/2)^2 + \cdots \\ &= N(3/2 + 3/2 + \cdots) \quad \text{log } n \text{ terms} \\ &= O(N \log N)\end{aligned}$$

$$\begin{aligned}\text{time} &= 5/12n^3/3 + 4 \cdot 5/12(n/2)^3/3 + \cdots \\ &= 5/12N^{3/2}(1 + 1/4 + 1/16 + \cdots) \\ &= O(N^{3/2})\end{aligned}$$

More direct factorizations

Minimum degree, multifrontal,. . .

Finding good separators and domain decompositions is tough in general.

Parallel preconditioners

Sparse operations in parallel: mvp

Mvp $y = Ax$

```
for i=1..n
  y[i] = sum over j=1..n a[i,j]*x[j]
```

In parallel:

```
for i=myfirstrow..mylastrow
  y[i] = sum over j=1..n a[i,j]*x[j]
```

How about ILU solve?

Consider $Lx = y$

```
for i=1..n
  x[i] = (y[i] - sum over j=1..i-1 ell[i,j]*x[j])
         / a[i,i]
```

Parallel code:

```
for i=myfirstrow..mylastrow
  x[i] = (y[i] - sum over j=1..i-1 ell[i,j]*x[j])
         / a[i,i]
```

Problems?

Block method

```
for i=myfirstrow..mylastrow
  x[i] = (y[i] - sum over j=myfirstrow..i-1 ell[i,j]*x[j]
         / a[i,i]
```

Block Jacobi with local GS solve

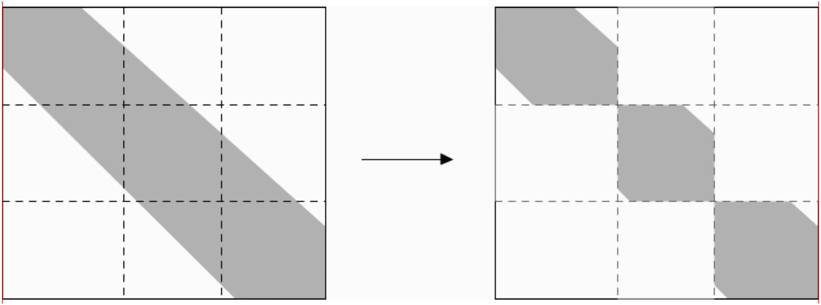


Figure : Sparsity pattern corresponding to a block Jacobi preconditioner

Multicolour ILU