Domain Decomposition

Parallelization of Mesh Based Applications

Panagiotis Adamidis
Thomas Bönisch

University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de

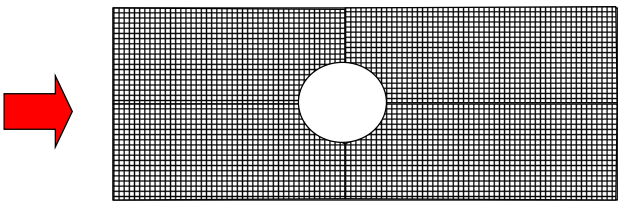
Outline

- Introduction
- Basics
- Boundary Handling
- Example: Finite Volume Flow Simulation on Structured Meshes
- Example: Finite Element Approach on an Unstructured Mesh

Parallelization - Target

- High Application Performance
- Using real big MPP's
  - no loss in efficiency even when using 500 Processors and more
- Using Clusters of SMP's
  - no decrease in Performance due to using external Network connections

A Problem (I)



Flow around a cylinder:
Numerical Simulation using FV, FE or FD

Data Structure: A(1:n,1:m)

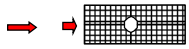
Solve: (A+B+C)x=b

Parallelization strategies

Work decomposition

Scaling ?

do i=1,100
→ i=1,25
i=26,50
i=51,75
i=76,100



Flow around a cylinder:
Numerical Simulation using FV, FE or FD

→ Data Structure: A(1:n,1:m)

→ Solve: (A+B+C)x=b

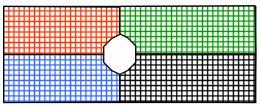
Data decomposition

Scales
too much communication ?

A(1:20,1:50)
A(1:20,51:100)
A(1:20,101:150)
A(1:20,151:200)

Domain decomposition

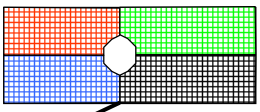
Good Chance

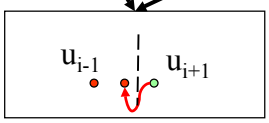


Parallelization Problems

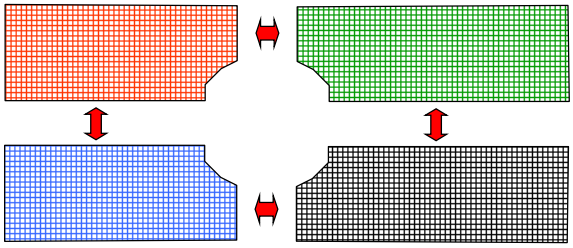
- Decomposition (Domain, Data, Work)
- Communication

$$du / dx = (u_{i+1} - u_{i-1}) / dx$$



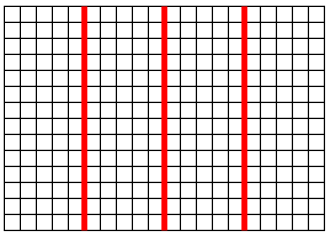


Concepts - Message Passing (II)

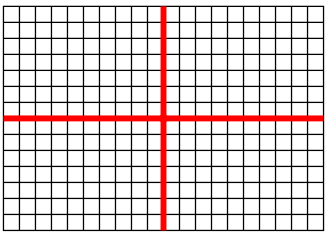


User defined communication

How to split the Domain in the Dimensions (I)



1 - Dimensional



2 (and 3) - Dimensional

How to split the Domain in the Dimensions (II)

- That depends on:
  - computational speed i.e. processor: vectorprocessor or cache
  - communication speed:
    - **latency**
    - **bandwidth**
    - **topology**
  - number of subdomains needed
  - load distribution (is the effort for every mesh cell equal)

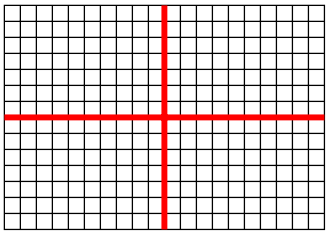
Replication versus Communication (I)

- If we need a value from a neighbour we have basically two opportunities
  - getting the necessary value directly from the neighbour, when needed
    ➡ Communication, Additional Synchronisation
  - calculating the value of the neighbour again locally from values known there
    ➡ Additional Calculation
- Selection depends on the application

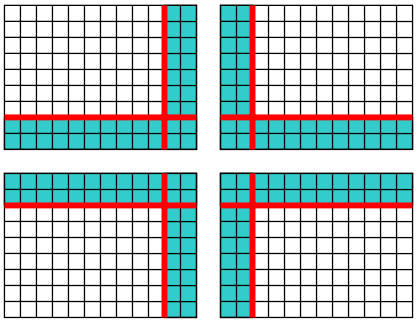
Replication versus Communication (II)

- Normally replicate the values
  - Consider how many calculations you can execute while only sending 1 Bit from one process to another (6 µs, 1.0 Gflop/s ➔ 6000 operations)
  - Sending 16 kByte (20x20x5) doubles (with 300 MB/s bandwidth ➔ 53.3 µs ➔ 53 300 operations)
  - very often blocks have to wait for their neighbours
  - but extra work limits parallel efficiency
- Communication should only be used if one is quite sure that this is the best solution

2- Dimensional DD with two Halo Cells



Mesh Partitioning



Subdomain for each Process

Example:

Parallelization of a 3D Finite Volume Flow Solver

Starting Point: Sequentiell Program

- Written in FORTRAN77
- Using structured meshes
- Parts of the program
  - Preprocessing, reading Data
  - Main Loop
    - **Setup of the equation system**
    - **Preconditioning**
    - **Solving step**
  - Postprocessing, writing Data

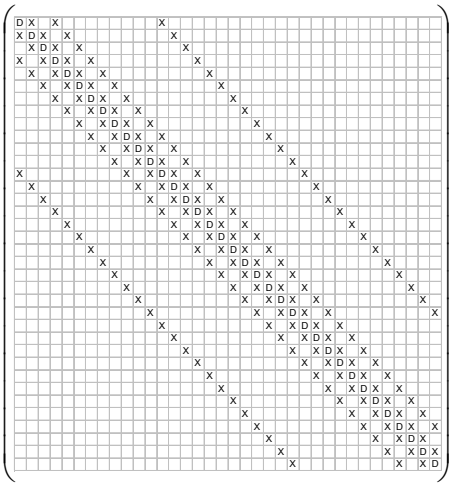
Dynamic Data Structures

- Pure FORTRAN77 is too static
  - number of processors can vary from run to run
    size of arrays even within the same case can vary
  - dynamic data structurs
    - **use Fortran90 dynamic arrays**
    - **use all local memory on a PE for a huge FORTARN77 array and setup your own memory management**
    - **second method has a problem on SMP's and cc-NUMA's we should only use as much memory as necessary**

Main Loop

- Setup of the equation system
  - each cell has 6 neighbours
  - needs data from neighbour cells
  - 2 halo cells at the inner boundaries
- Preconditioning
  - no neighbour infomation needed at all
  - completely done locally
- Solving step
  - Jacobi Line relaxation with subiterations
  - more complicated (next slides)

Heptadiagonalmatrix



Parallelization - Solver (I)

- Sequential:

$$A\vec{u} = \vec{r}$$

$$\vec{u} = A^{-1}\vec{r}$$

- Parallelization problems:
  - Matrix A is distributed
  - Matrix inversion is a priori not parallelizable

Parallelization - Solver (II)

$$A\vec{u} = \vec{r}$$

$$A = L + M$$

$$(L+M)\vec{u} = \vec{r}$$

$$L\vec{u} = \vec{r} - M\vec{u}$$

$$L\vec{u}^{(j)} = \vec{r} - M\vec{u}^{(j-1)}$$

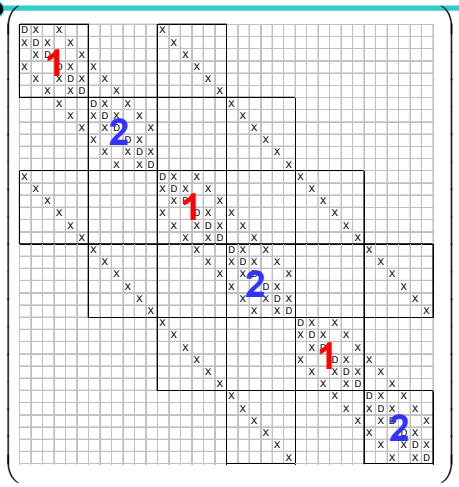
$$\vec{u}^{(j)} = L^{-1}\left(\vec{r} - M\vec{u}^{(j-1)}\right)$$

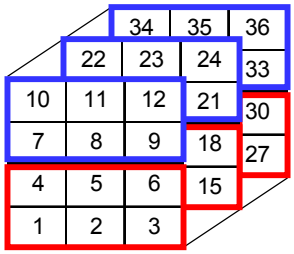
Parallelization - Solver (III)

$$A = \qquad L \qquad + \qquad M$$

$$A = \left(\begin{array}{cccc|cccc|cccc}
m_1 & u_1 & & & & & & & & & & \\
l_2 & m_2 & u_2 & & & & & & & & & \\
 & l_3 & m_3 & u_3 & & & & & & & & \\
 & & l_4 & m_4 & & & & & & & & \\
\hline
 & & & & m_5 & u_5 & & & & & & \\
 & & & & l_6 & m_6 & u_6 & & & & & \\
 & & & & & l_7 & m_7 & u_7 & & & & \\
 & & & & & & l_8 & m_8 & & & & \\
\hline
 & & & & & & & & m_9 & u_9 & & \\
 & & & & & & & & l_{10} & m_{10} & u_{10} & \\
 & & & & & & & & & l_{11} & m_{11} & u_{11} \\
 & & & & & & & & & & l_{12} & m_{12}
\end{array}\right) + \left(\begin{array}{cccc|cccc|cccc}
 & & & & & & & & & & & \\
 & & & & & & & & & & & \\
 & & & & & & & & & & & \\
 & & & & & & & & & & & \\
\hline
 & & & & & & u_4 & & & & & \\
 & & & & l_5 & & & & & & & \\
 & & & & & & & & & & & \\
 & & & & & & & & & & & \\
\hline
 & & & & & & & & & & & u_8 \\
 & & & & & & & & & & l_9 & \\
 & & & & & & & & & & & \\
 & & & & & & & & & & &
\end{array}\right)$$

A Real Heptadiagonalmatrix

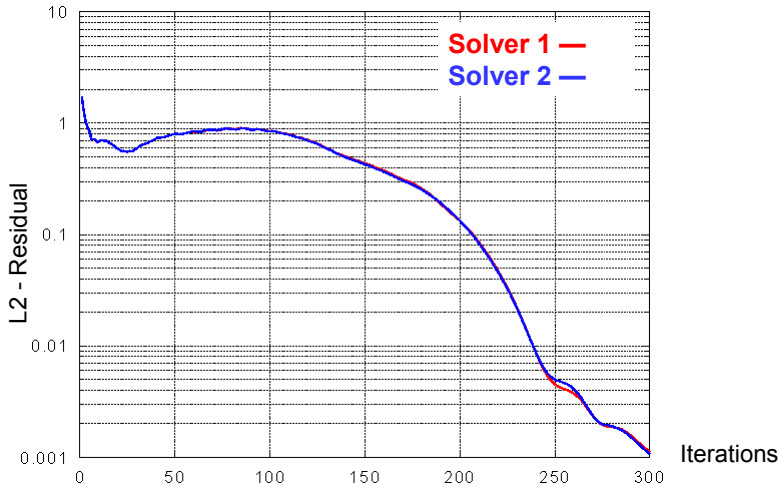




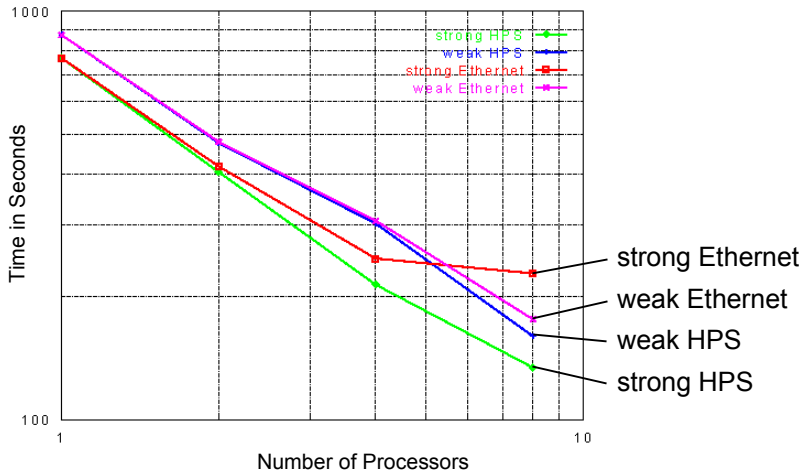
Difference between strong / weak coupling

- Solver with weak coupling
  - Extra computational effort due to additional solving step (but no factorization)
  - Additional update of right hand side
  - Two times communication
    - **one after each solving step**
- Solver with stronger coupling
  - Jacobi line relaxation method with subiterations
  - collapse iterations from line relaxation and parallelzation
  - no additional iterations
  - much more communication

Comparison of Solvers - Convergence



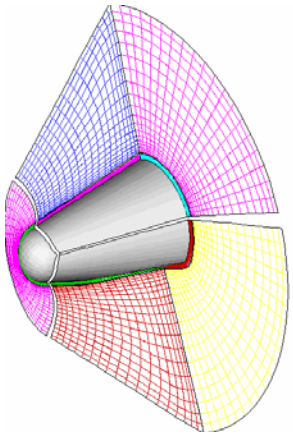
Comparison of Solvers - Performance



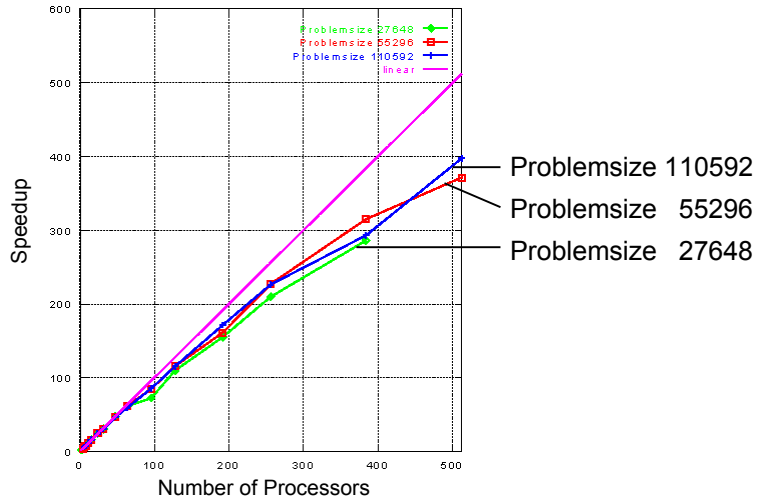
Results - Solving Method

- the presented solver with weak coupling works fine for this CFD problems
- Solutions differ in the scale of one percent
- convergence rate is nearly equal to the sequential program

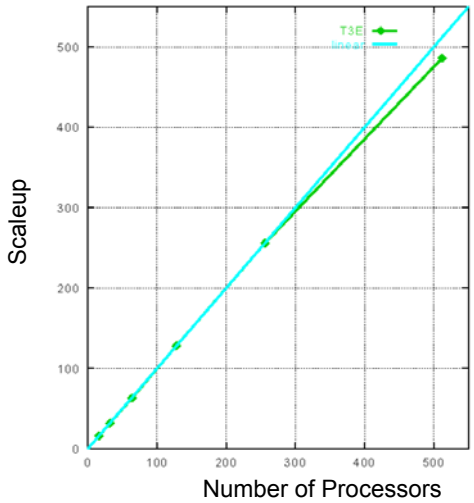
Domain Decomposition



Speedup on Cray T3E

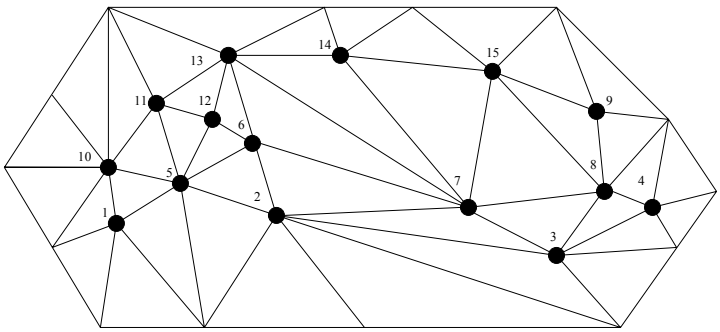


Scaleup on Cray T3E



Domain Decomposition of Unstructured Grids

Unstructured FEM Grid with Global Numbering



Shape of Corresponding System Matrix

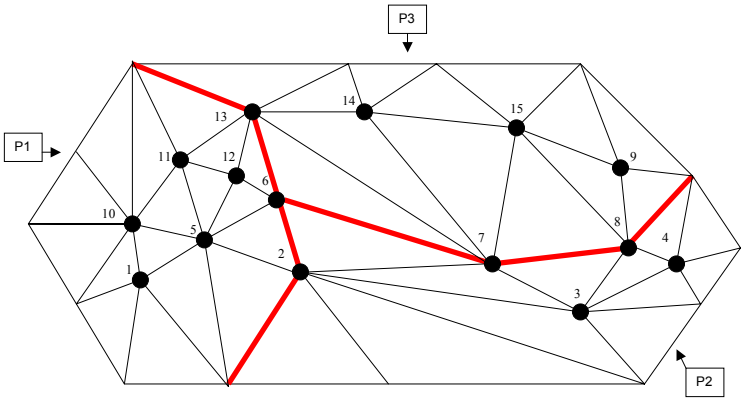
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	x				x					x					
2		x	x		x	x	x								
3		x	x	x			x	x							
4			x	x				x	x						
5	x	x			x	x				x	x	x			
6		x			x	x	x					x	x		
7		x	x			x	x	x					x	x	x
8			x	x			x	x	x						x
9				x				x	x						x
10	x				x					x	x				
11					x					x	x	x	x		
12					x	x					x	x	x		
13						x	x				x	x	x	x	
14							x						x	x	x
15							x	x	x					x	x

x = edge of FEM grid

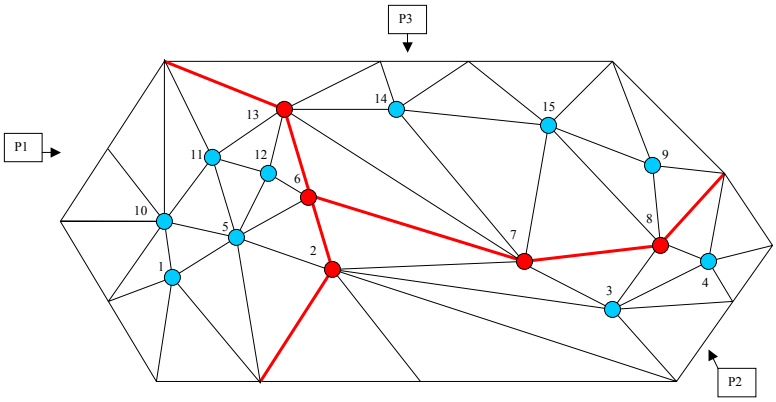
Domain Decomposition

1. Nonoverlapping Domain Decomposition
2. Grid Points Separated into Inner and Boundary Points
3. Renumbering of the Inner and Boundary Points

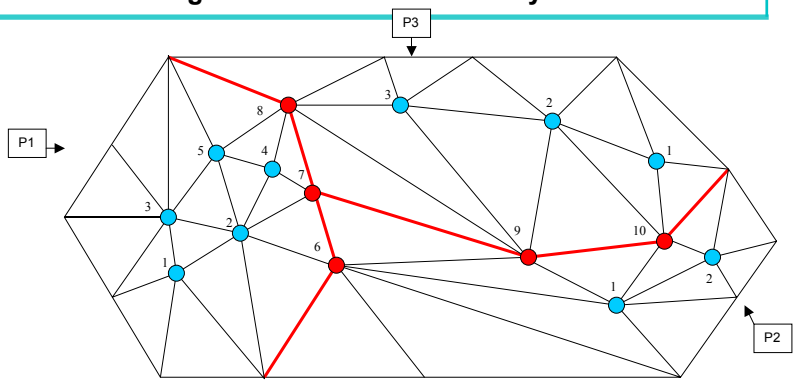
1. Nonoverlapping Domain Decomposition



2. Grid Points Separated into Inner and Boundary Points



3. Renumbering of the Inner and Boundary Points

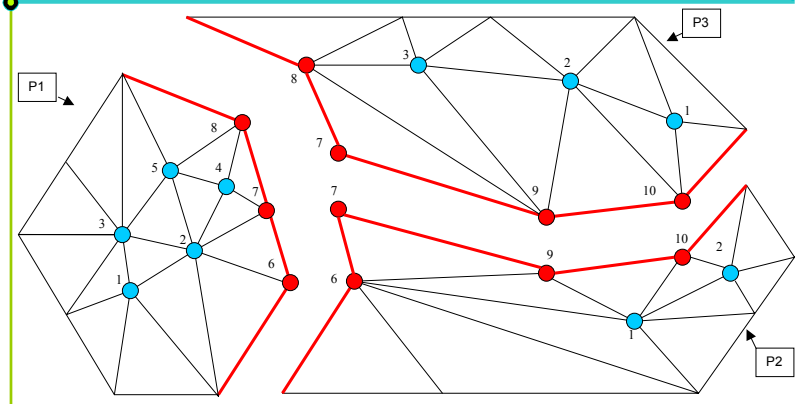


- Inner points are numbered from 1 up to the number of inner points
- Boundary points are numbered globally starting from the maximum of all inner points of all partitions plus 1

Arrow Shaped System Matrix after Renumbering

	P1					P2		P3			logical boundary				
	1	2	3	4	5	1	2	1	2	3	6	7	8	9	10
1	x	x	x												
2	x	x	x	x	x						x	x			
3	x	x	x		x										
4		x		x	x							x	x		
5		x	x	x	x								x		
1						x	x				x			x	x
2						x	x								x
1								x	x						x
2								x	x	x				x	x
3									x	x			x	x	
6		x				x					x	x		x	
7		x		x							x	x	x	x	
8				x	x					x		x	x	x	
9						x			x	x	x	x	x	x	x
10						x	x	x	x					x	x

Data Distribution



Linear System

$$\begin{pmatrix} A_{II}^1 & & & & A_{IB}^1 \\ & A_{II}^2 & & & A_{IB}^2 \\ & & \cdot & & \cdot \\ & & & \cdot & \cdot \\ & & & A_{II}^n & A_{IB}^n \\ A_{BI}^1 & A_{BI}^2 & \cdot & \cdot & A_{BB}^n & A_{BB} \end{pmatrix} \begin{pmatrix} x_I^1 \\ x_I^2 \\ \cdot \\ \cdot \\ x_I^n \\ x_B \end{pmatrix} = \begin{pmatrix} b_I^1 \\ b_I^2 \\ \cdot \\ \cdot \\ b_I^n \\ b_B \end{pmatrix} \quad \text{with} \quad A_{BB} = \sum_{i=1}^{n} A_{BB}^i$$

- Available on local memories

$$\begin{pmatrix} A_{II}^i & A_{IB}^i \\ A_{BI}^i & A_{BB}^i \end{pmatrix} \qquad i = 1, \ldots, n$$

Direct Substructuring

1. Direct factorization
2. Assembling of the Schur complement system
3. Solving of the Schur complement system
4. Solving of the interior unknowns

Transformations of the Original System

$$A=\begin{pmatrix} I & . & . & . & . \\ . & I & . & . & . \\ . & . & . & . & . \\ A_{BI}^{1} A_{II}^{(1)^{-1}} & A_{BI}^{2} A_{II}^{(2)^{-1}} & . & . & I \end{pmatrix} \begin{pmatrix} A_{II}^{1} & . & . & . & . \\ . & A_{II}^{2} & . & . & . \\ . & . & . & . & . \\ . & . & S^{1} + & . & + S^{n} \end{pmatrix} \begin{pmatrix} I & . & . & A_{II}^{(1)^{-1}} & A_{IB}^{1} \\ . & I & . & A_{II}^{(2)^{-1}} & A_{IB}^{2} \\ . & . & . & . & . \\ . & . & . & . & I \end{pmatrix}$$

$$S^{i} = A_{BB}^{i} - A_{BI}^{i} A_{II}^{(i)^{-1}} A_{IB}^{i}$$

$$\begin{pmatrix} A_{II}^{1} & . & . & . \\ . & A_{II}^{2} & . & . \\ . & . & . & . \\ A_{BI}^{1} & A_{BI}^{2} & . & I \end{pmatrix} \begin{pmatrix} I & . & . & A_{II}^{(1)^{-1}} A_{IB}^{1} \\ . & I & . & A_{II}^{(2)^{-1}} A_{IB}^{2} \\ . & . & . & . \\ . & . & . & S^{1} + . + S^{n} \end{pmatrix} \begin{pmatrix} x_{I}^{1} \\ x_{I}^{2} \\ . \\ x_{b} \end{pmatrix} = \begin{pmatrix} b_{I}^{1} \\ b_{I}^{2} \\ . \\ b_{b} \end{pmatrix}$$

Schurcomplement System

$$A_{II}^{i}\, x_{I}^{i} = b_{I}^{i} - A_{IB}^{i}\, x_{B}$$

$$A_{SH}\, x_{B} = b_{SH}$$

with

$$A_{SH} = \sum_{i=1}^{n} \left(A_{BB}^{i} - A_{BI}^{i} \left(A_{II}^{i} \right)^{-1} A_{IB}^{i} \right)$$

$$b_{SH} = b_{B} - \sum_{i=1}^{n} A_{BI}^{i} \left(A_{II}^{i} \right)^{-1} b_{I}^{i}$$

or

$$A_{SH} = \sum_{i=1}^{n} \left(A_{BB}^{i} - A_{BI}^{i} Z_{i} \right)$$

$$b_{SH} = b_{B} - \sum_{i=1}^{n} A_{BI}^{i} z_{i}$$

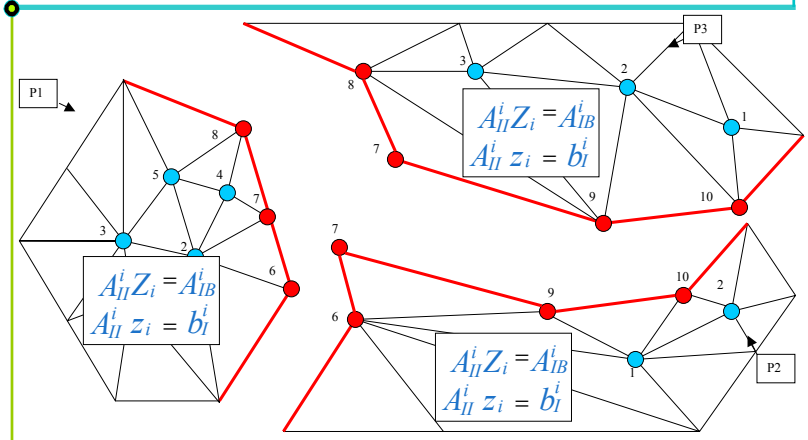
where

$$Z_{i} = \left(A_{II}^{i} \right)^{-1} A_{IB}^{i} \qquad A_{II}^{i} Z_{i} = A_{IB}^{i}$$

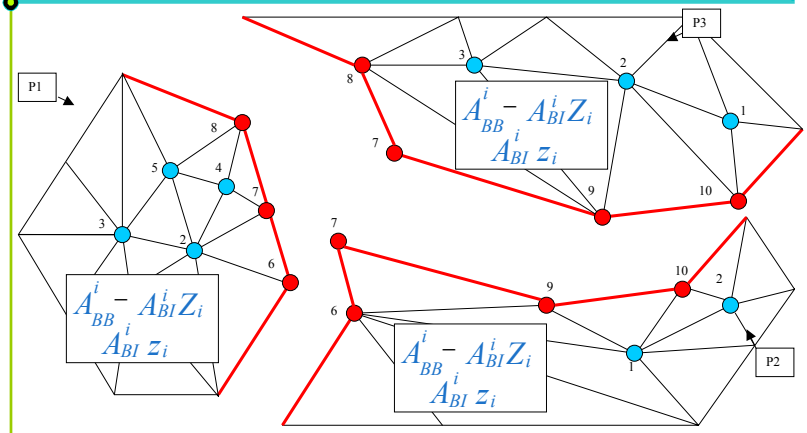
or

$$z_{i} = \left(A_{II}^{i} \right)^{-1} b_{I}^{i} \qquad A_{II}^{i}\, z_{i} = b_{I}^{i}$$

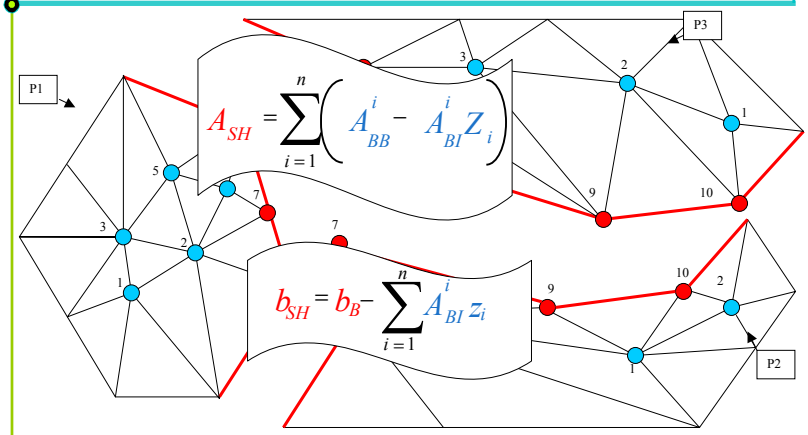
1. Direct factorization



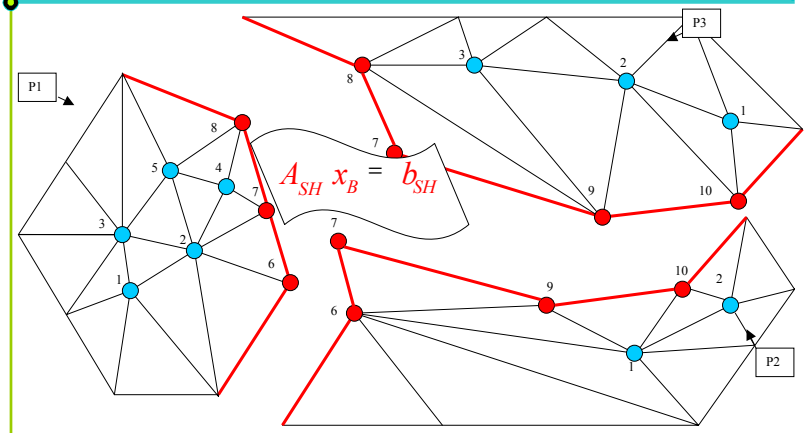
2. Assembling of the Schur complement system (I)



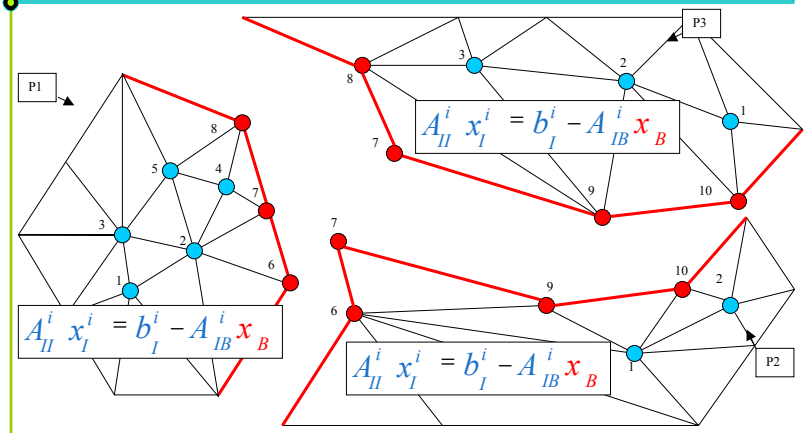
2. Assembling of the Schur complement system (II)



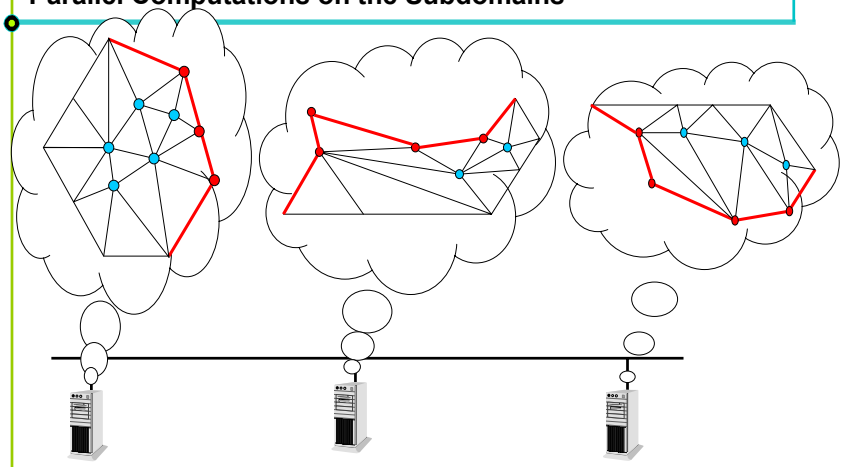
3. Solving of the Schur complement system



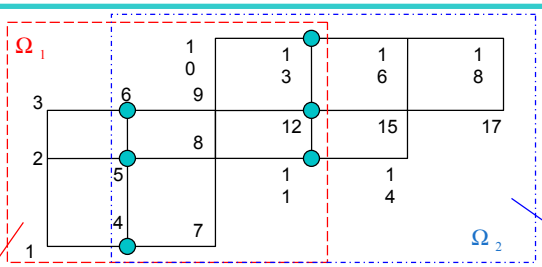
4. Solving of the interior unknowns



Parallel Computations on the Subdomains



Overlapping Domain Decomposition



Literature

- Barry F. Smith, Petter E. Bjoerstad, William D. Gropp:
  Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 1996,
  ISBN 0-521-49589-X

- David E. Keyes, Youcef Saad, Donald G. Truhlar:
  Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering, SIAM, 1995,
  ISBN 0-89871-348-X