Inria Project Lab C2S@Exa

Parallelizing the Traces software

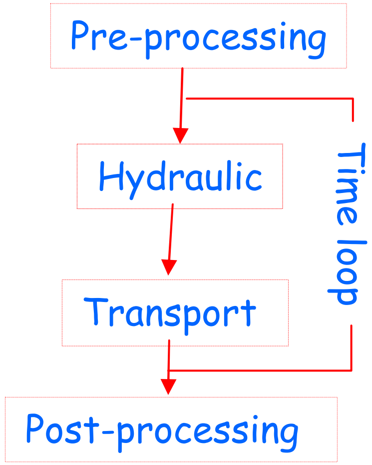
Rachid El khaoulani El idrissi

December 2013

Context & Motivation

Traces: numerical simulation of radio-active waste storage in profound geological layers

Two sorts of problem can be treated:



- Hydraulic: single-phase flow in porous media
- Transport: migration of radioactive waste in porous media

Large-scale problems in both points of view: spatial and temporal

- Long-term performance and safety assessment
- Large-size domains have to be dealt with

Parallelizing the Traces software

- To make more realistic and reliable studies
- To take advantage from computing capabilities

Outline

1. Hydraulic problem
2. Distributed-memory parallelism
   A. MPI-based model
   B. Mesh partitioning
   C. Parallel assembling and resolving
   D. Performance evaluation
3. Shared-memory and combined parallelisms
   A. OpenMP-based model
   B. Numerical results
4. Conclusion

Outline

1. Hydraulic problem
2. Distributed-memory parallelism
    A. MPI-based model
    B. Mesh partitioning
    C. Parallel assembling and resolving
    D. Performance evaluation
3. Shared-memory and combined parallelisms
    A. OpenMP-based model
    B. Numerical results
4. Conclusion

Hydraulic: Mathematical Model

$$\begin{cases} s\dfrac{\partial u}{\partial t} = -div(q) - s\lambda u + f & \text{Mass balance equation} \\ q = D.\nabla u & \text{Darcy's law} \end{cases}$$

Unknowns

u hydraulic charge

q filtration velocity

Parameters

D hydraulic conductivity

s storage coefficient

f source/sink term

λ a kinetic term

- Temporal discretization: implicit
- Spatial discretization: Mixed Hybrid Finite Element Method

→ Algebraic linear system whose unknowns are associated to the mesh faces

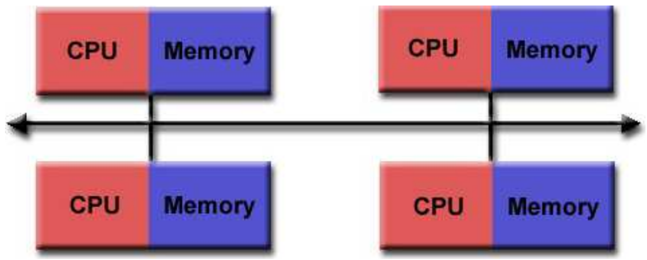
→ Parallel assembling and resolving of the resulting linear algebraic system is the most challenging part of the hydraulic problem

Outline

1. Hydraulic problem
2. Distributed-memory parallelism
    A. MPI-based model
    B. Mesh partitioning
    C. Parallel assembling and resolving
    D. Performance evaluation
3. Shared-memory and combined parallelisms
    A. OpenMP-based model
    B. Numerical results
4. Conclusion

Distributed-memory parallelism

Distributed-memory architecture



- Private memory
- Data transfer should be programmed explicitly

Parallelization method

- *Mesh partitioning*
- *Message passing programming through MPI standard*

Outline

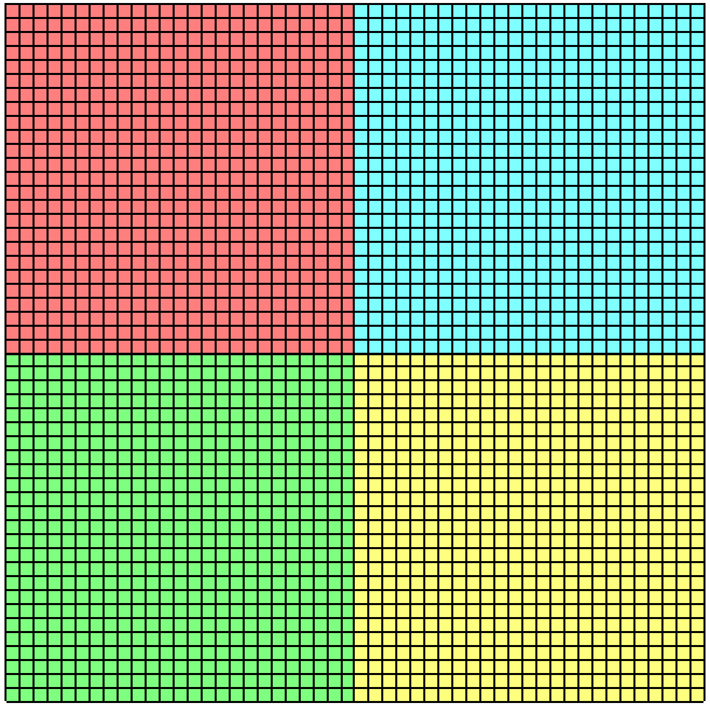
1. Hydraulic problem
2. Distributed-memory parallelism
    A. MPI-based model
    B. Mesh partitioning
    C. Parallel assembling and resolving
    D. Performance evaluation
3. Shared-memory and combined parallelisms
    A. OpenMP-based model
    B. Numerical results
4. Conclusion

Unstructured mesh partitioning

Static, non-overlapping and homogeneous partitioning

Partitioning software: Metis, Scotch

Mapping of the mesh elements: divide mesh elements into groups of elements



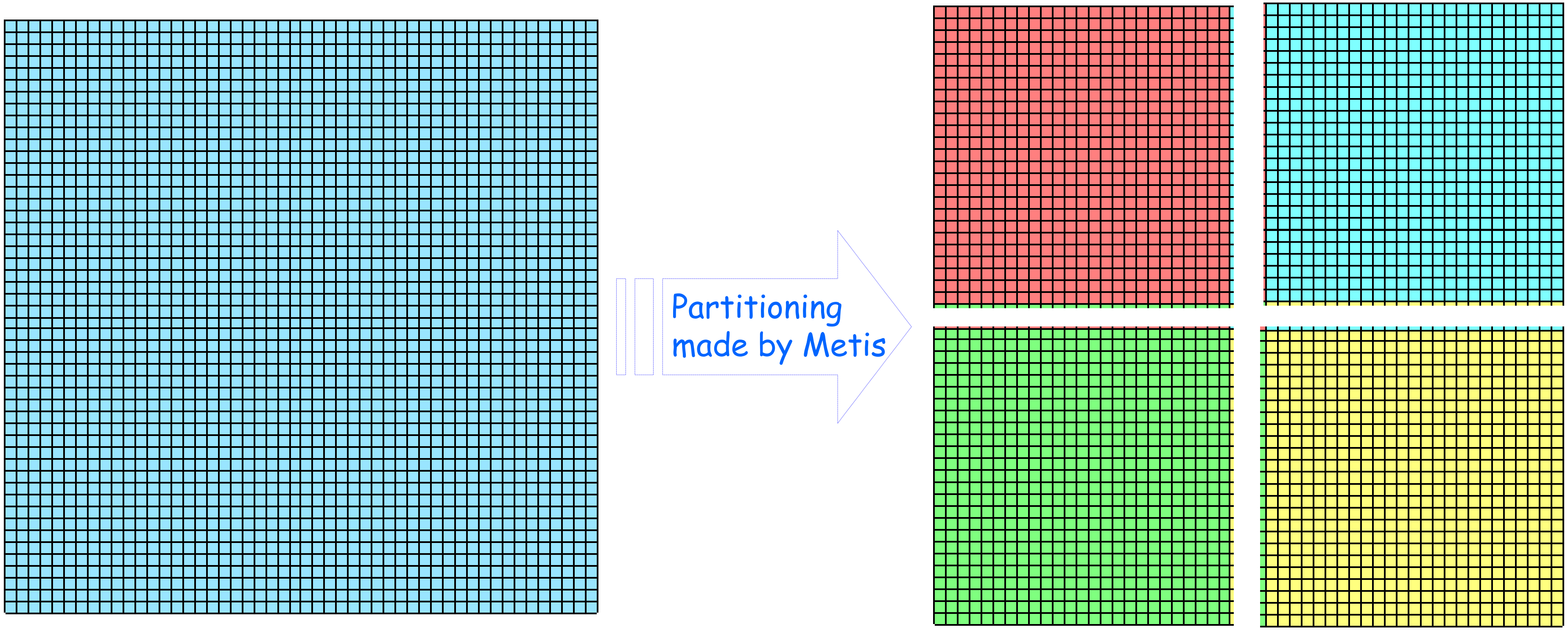
- Partitioning of mesh nodes
- Partitioning of mesh faces (edges in 2D)
- Neighboring relations between MPI-processes

→ New input file

→ Distributed data

Mesh partitioning

Non-overlapping homogeneous mesh partitioning

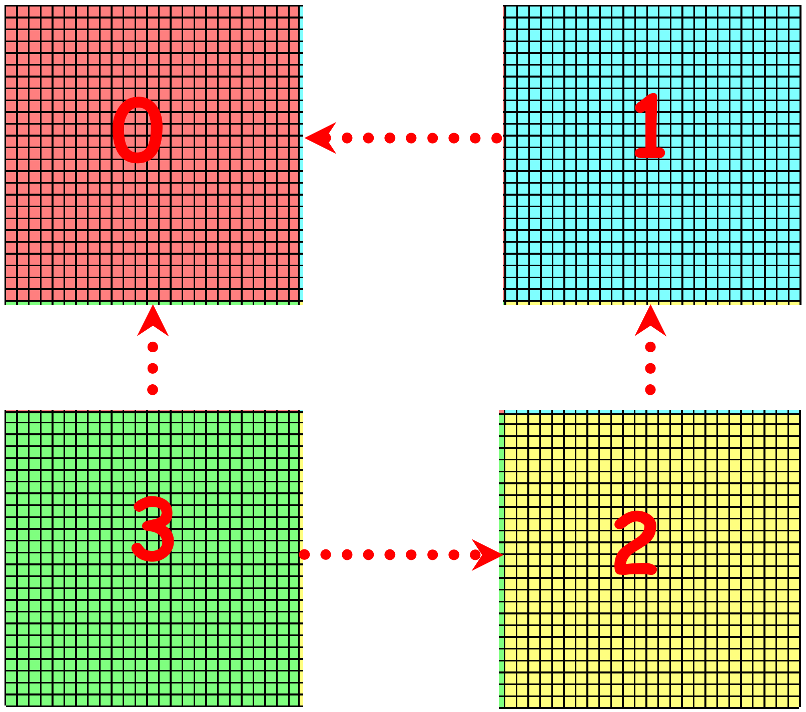


Communication lists

Communication list: list of the local numbers of the common faces in each couple of neighbors

[neighbor1, neighbor2] → [(neighbor1,list1) , (neighbor2,list2)]

Matched lists



4 couples of neighbors

4 messages transferred

Transferring only 4 messages of integers to build matched communication lists

There are as many messages as the number of couples of neighbors

Outline

1. Hydraulic problem
2. Distributed-memory parallelism
   - A. MPI-based model
   - B. Mesh partitioning
   - C. Parallel assembling and resolving
   - D. Performance evaluation
3. Shared-memory and combined parallelisms
   - A. OpenMP-based model
   - B. Numerical results
4. Conclusion

The Hypre librairy

What is Hypre?

Software library of high performance preconditioners and solvers for the solution of **large, sparse linear systems** on massively parallel computers

Krylov space solvers

- Symmetric system: Conjugate Gradient
- Asymmetric system: GMRES, Bi-Conjugate Gradient stabilized …

Preconditioners

Algebraic Multigrid, ILU(k), Block Jacobi ILU(k), Diagonal …

How to use Hypre

Linear-Algebraic System interface (IJ)

Hypre-Traces interfacing

Distributed data form

Matrices are assumed to be distributed across the MPI-Processes by contiguous blocks of rows

$$A = \begin{pmatrix} A_0 \\ A_1 \\ . \\ . \\ A_{p-1} \end{pmatrix}$$

Hypre defines a <u>new numbering</u> of the DOF

The DOF 1 to n_0 reside in MPI-Process 0

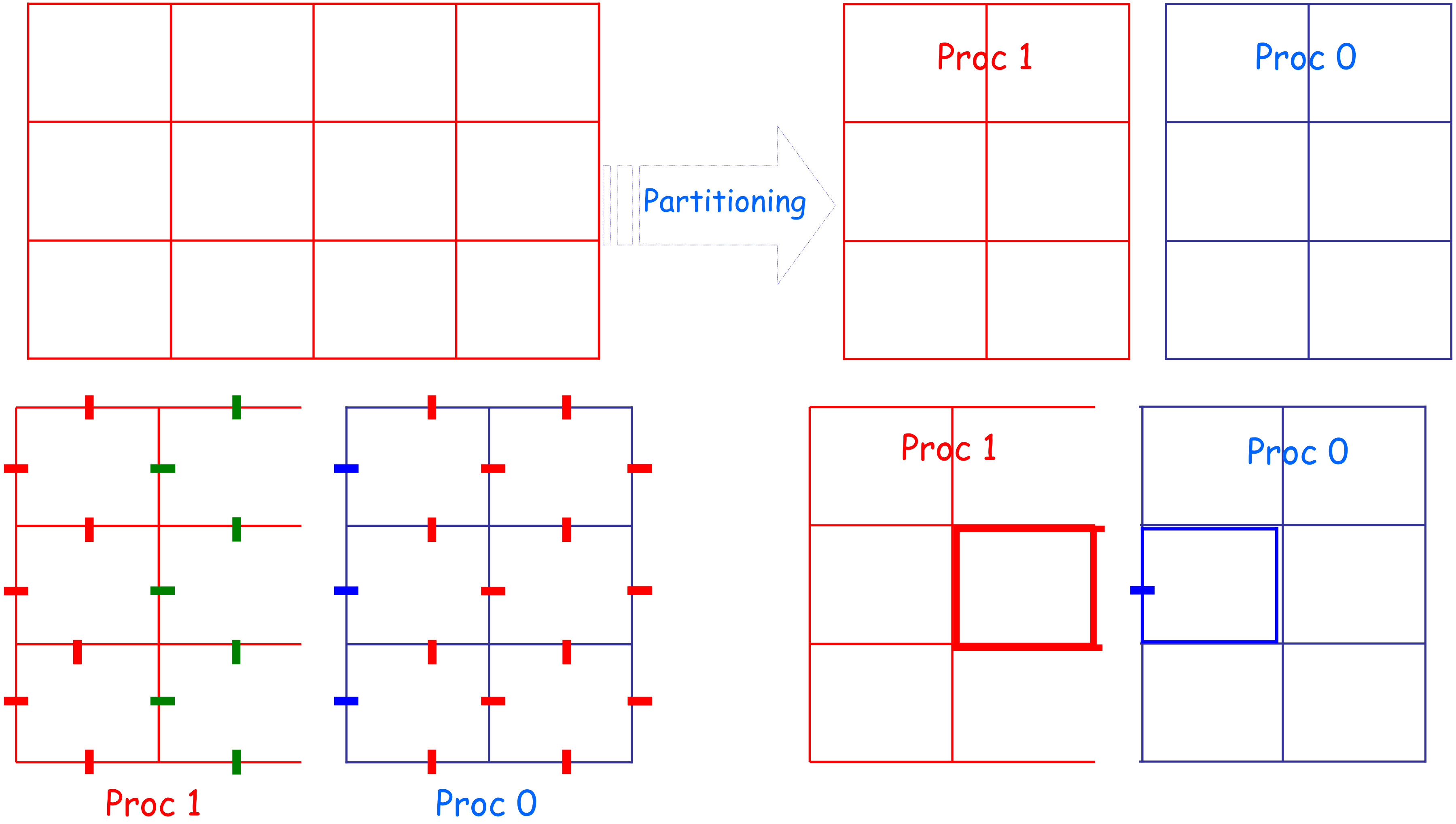
The DOF n_0+1 to n_1 reside in MPI-Process 1

The DOF $n_{k-1}+1$ to n_k reside in MPI-Process k

Main points

- Hypre defines its own numbering of the DOF
- Hypre requires a mapping of the DOF on the MPI-Processes
- MPI-Processes define actual blocks of the system for Hypre independently

Matrix parallel assembling



For each DOF in common

- ❖ From process 1 to process 0: 4 Coefficients + 4 indices

→ Two messages are transferred

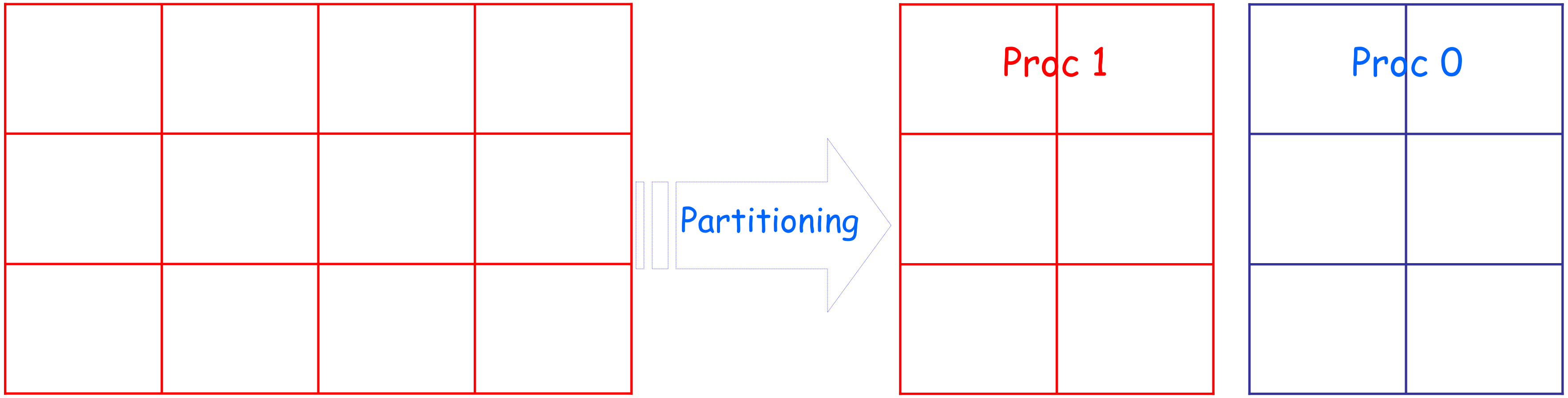
Matrix parallel assembling

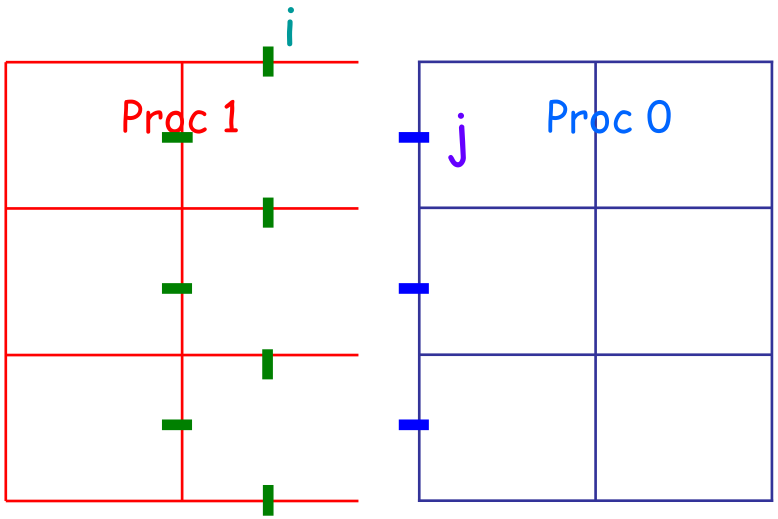
Each MPI-Process computes its own FE matrix then transmits it to Hypre

$$\begin{pmatrix} \\ \beta_1 \;.\; \beta_2 \;.\; \beta_3 \;.\;.\; \beta \\ \\ \end{pmatrix} \xleftrightarrow{\text{DOF in common}} \begin{pmatrix} \\ \alpha_1 \;.\; \alpha_2 \;.\; \alpha_3 \;.\;.\; \alpha \\ \\ \end{pmatrix}$$

$$\begin{pmatrix} \\ \beta_1 \;.\; \beta_2 \;.\; \beta_3 \;.\;.\; \beta \\ \\ \end{pmatrix} \qquad \begin{pmatrix} \\ \alpha_1 \;.\; \alpha_2 \;.\; \alpha_3 \;.\;.\; \alpha+\beta \;.\;.\; \beta_1 \;.\; \beta_2 \;.\; \beta_3 \\ \\ \end{pmatrix}$$

Matrix parallel assembling





Process 1 needs the column number j from the process 0 to put correctly the coefficient a_{ij} in the Hypre matrix

Process 0 sends the Hypre numbering of the DOF in common to process 1

→ One message of integers from process 0 to process 1

Hypre: solving

Parallel assembling and transmitting the RHS to Hypre

$$RHS = \begin{pmatrix} RHS_0 \\ RHS_1 \\ . \\ . \\ RHS_{p-1} \end{pmatrix}$$

Define an initial guess of the solution …

Choice a solver and its parameters, preconditioner…

Get the solution from Hypre and adapt it to the Traces numbering

Outline

1. Hydraulic problem
2. Distributed-memory parallelism
    A. MPI-based model
    B. Mesh partitioning
    C. Parallel assembling and resolving
    D. Performance evaluation
3. Shared-memory and combined parallelisms
    A. OpenMP-based model
    B. Numerical results
4. Conclusion

Performance: PCG/DS

Mesh: hexahedral, 2 506 140 elements, 7 591 723 faces

Solver: Preconditioned Conjugate Gradient PCG, Convergence tolerance 1e-10

PCG / DS			
Number of MPI-Processes	CPU Time(s) Solve+Setup	Speed-up	Number of iterations
1	8272.1		10480
2	4124	2	10480
4	2062.6	4.01	10480
8	1045.57	7,91	10480
16	525	15,76	10480
20	420	19.7	10480

Validation of parallel interfacing of Traces and Hypre software

Setup

- Parallel passing of the linear system to Hypre
- Hypre's setup

Performance: PCG/ Block Jacobi - ILU(1)

Mesh: hexahedral, 2 506 140 elements, 7 591 723 faces

Solver: Preconditioned Conjugate Gradient PCG, Convergence tolerance 1e-10

PCG/Block Jacobi-ILU(1)			
Number of MPI-Processes	CPU Time(s) Solve+Setup	Speed-up	Number of iterations
1	4524		2083
2	2222	2.03	2096
4	1111	4.07	2101
8	162.5	8.04	2118
16	282	16.04	2151
20	227.87	19.85	2184

Performance: PCG/AMG

Mesh: hexahedral, 2 506 140 elements, 7 591 723 faces

Solver: Preconditioned Conjugate Gradient PCG, Convergence tolerance 1e-10

PCG / AMG			
Number of MPI-Processes	CPU Time(s) Solve+Setup	Speed-up	Number of iterations
1	321.52		2
2	175.56	1.83	2
4	90.05	3.57	2
8	49.01	6.56	2
16	26.88	12	2
20	22.9	14.04	2

Performance: comparison

Mesh: hexahedral, 2 506 140 elements, 7 591 723 faces

Solver: Preconditioned Conjugate Gradient PCG, Convergence tolerance 1e-10

	PCG / DS		PCG / Block Jacobi - ILU(1)		PCG / AMG	
Number of MPI-Processes	CPU Time(s) Solve+Setup	Speed-up	CPU Time(s) Solve+Setup	Speed-up	CPU Time(s) Solve+Setup	Speed-up
1						
2		2		2.03		1,83
4		4.01		4.07		3,57
8		7,91		8.04		6.56
16		15,76		16.04		12
20		19.7		19.85		14.04

PCG/AMG is less scalable than PCG/DS and PCG/Block Jacobi - ILU(1)

Performance: comparison

Mesh: hexahedral, 2 506 140 elements, 7 591 723 faces

Solver: Preconditioned Conjugate Gradient PCG, Convergence tolerance 1e-10

	PCG / DS		PCG / Block Jacobi-ILU(1)		PCG / AMG	
Number of MPI-Processes	CPU Time(s) Solve+Setup	Speed-up	CPU Time(s) Solve+Setup	Speed-up	CPU Time(s) Solve+Setup	Speed-up
1	8272.1		4524		321.52	
2	4121.6		2222		175.56	
4	2062.6		1111		90.05	
8	1045.57		162.5		49.01	
16	525		282		26.88	
20	420		227.87		22.9	

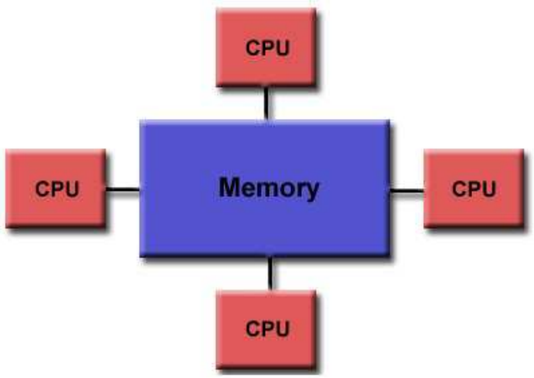
PCG/AMG is less scalable than PCG/DS and PCG/Block Jacobi-ILU(1)
However
It is more efficient in bringing down the CPU Time Than the others

Outline

1. Hydraulic problem
2. Distributed-memory parallelism
    A. MPI-based model
    B. Mesh partitioning
    C. Parallel assembling and resolving
    D. Performance evaluation
3. Shared-memory and combined parallelisms
    A. OpenMP-based model
    B. Numerical results
4. Conclusion

Multi-threading parallelism

Shared-memory architecture



- Memory may be accessed concurrently
- Threads communicate with each other by reading and writing in the shared memory

OpenMP: Open Multi-Processing

- Industry standard for shared-memory programming
- It's an API to realize multi-treaded parallelism

OpenMP performance

Mesh: hexahedral, 2 506 140 elements, 7 591 723 faces

Solver: Preconditioned Conjugate Gradient PCG, Convergence tolerance 1e-10

Intel Xeon , 1 NUMA, 8 cores, 2666 Mhz, 16GB of RAM

PCG / DS		
Number of threads	CPU Time(s) Solve	Speed-up
1	8263.1	
2	6003.69	1.38
4	5865.81	1.41
8	5752.19	1.44

PCG / AMG		
Number of threads	CPU Time(s) Solve	Speed-up
1	312.49	
2	213.78	1.46
4	191.58	1.63
8	186.72	1.67

Multi-threading performances aren't satisfying in the current version

Significant reduction of the CPU time is obtained

Combined OpenMP-MPI parallelism

Mesh: hexahedral, 2 506 140 elements, 7 591 723 faces

Solver: Preconditioned Conjugate Gradient PCG/DS, Convergence tolerance 1e-10

Combined OpenMP-MPI model			
Number of MPI-Processes	Number of threads	CPU Time(s) Solve	Speed-up
1	1	8263.1	
2	2	3005.69	2.75
4	2	1516.84	5.45
8	2	760.97	10.86
16	2	374.17	22.08
20	2	301.73	27.39

The speed-up obtained by OpenMP-model using 2 threads

Speed-up MPI-model
2
4.01
7.91
15.76
19.7

X 1.38 =

2.76
5.53
10.91
21.75
27.27

Performance of the combined model unites MPI-based and OpenMP-based models performances

Outline

1. Hydraulic problem
2. Distributed-memory parallelism
    A. MPI-based model
    B. Mesh partitioning
    C. Parallel assembling and resolving
    D. Performance evaluation
3. Shared-memory and combined parallelisms
    A. OpenMP-based model
    B. Numerical results
4. Conclusion

Conclusion

- Hydraulic problem has been parallelized using MPI and Hypre librairies
- Scotch and Metis were used to perform mesh partitioning
- Distributed data form
- Shared-memory and combined parallelims are well in progress
- Other Parallel solvers of linear systems can be easily interfaced with TRACES software