Ongoing high performance computing initiatives at ANDRA

… with Porflow™, Tough2-MP and Phreeqc/Phast

April 24th, 2013

© Andra

Porflow

Hydraulic and transport in porous media

Porflow: HYPRE

Main code for hydraulic and transport computation in saturated porous media

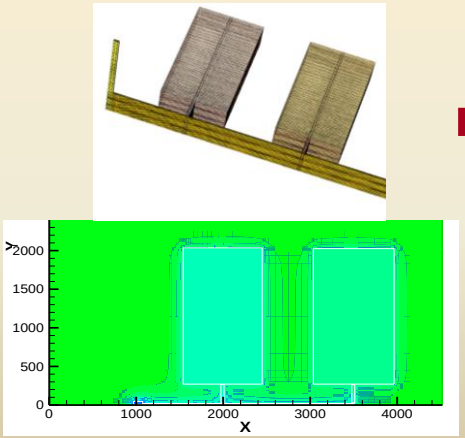
- 2002: first work with HYPRE in sequential mode
  - Plug HYPRE to Porflow™
  - Reduce computation time for hydraulic test cases (with respect to NSPCG or ACRi library)
  - Objective reached
- 2003: first attempt to use HYPRE in parallel mode (MPI)
  - 1 problem on 10 cores and/or processors leads to solve 10 times the initial problem on 1 core/processor.
  - x10 RAM
  - 0 acceleration
  - Failure

Priority on features

2003-2010: material becomes more and more powerful

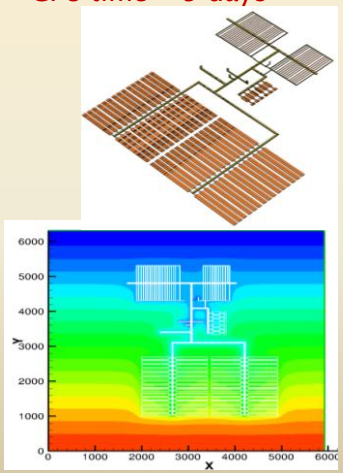
2005
(dossier 2005)

Simplified picture of a **single part** of the repository (HA zone) + COX
600.000 elements
CPU time = 5 days



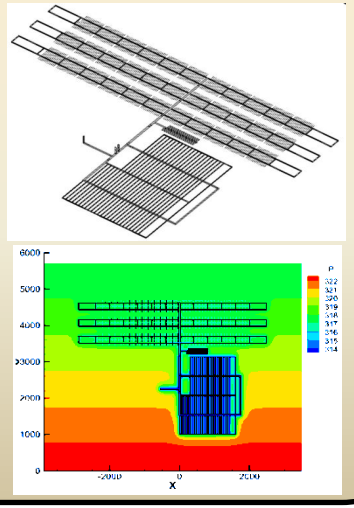
2009
(ZIRA)

Simplified picture of **all** the repository (modules + wells + drifts) + COX
8.000.000 elements
CPU time = 5 days



2012
(esquisses)

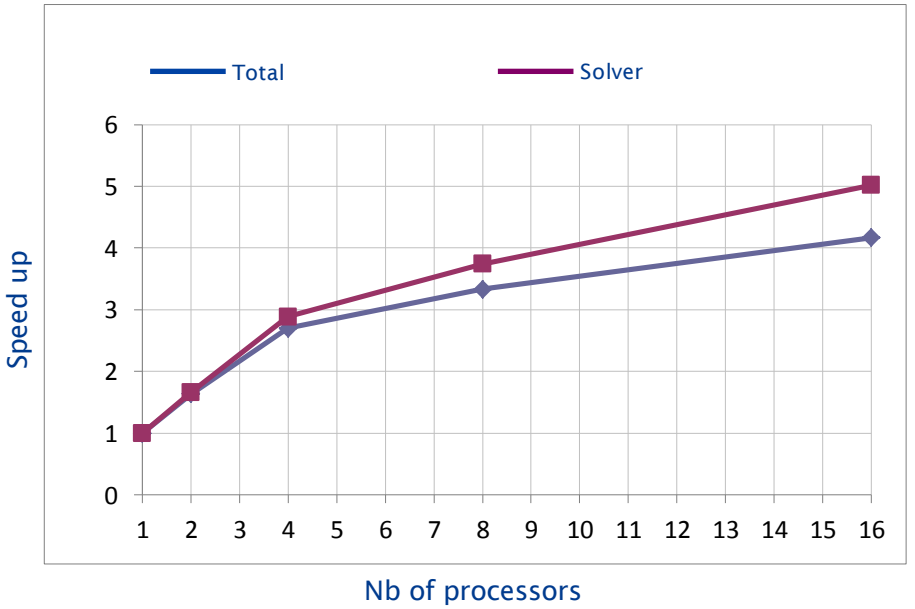
Detailed picture of **all** the repository (**cells** + wells + drifts) + COX
12.000.000 elements
CPU time = 3 days



☐ focus on physical features

Priority on performance (1/2)

- before 2009: add OpenMP support to NSPCG library
  - Max gain around 10% of the computation time with 2 cores
  - Above 2 cores, only gain 1-2%
- 2011: ACRi parallelizes part of the available solvers in the internal library (CONJ, BICO)
  - Test of CONJ solver (hydraulic computation)
  - 1.3M elements
  - 1 node/1 core : 178 min.
  - Limited gain…

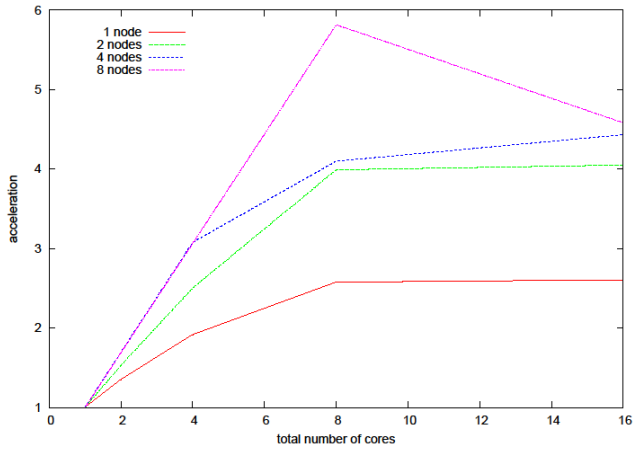


Priority on performance (2/2)

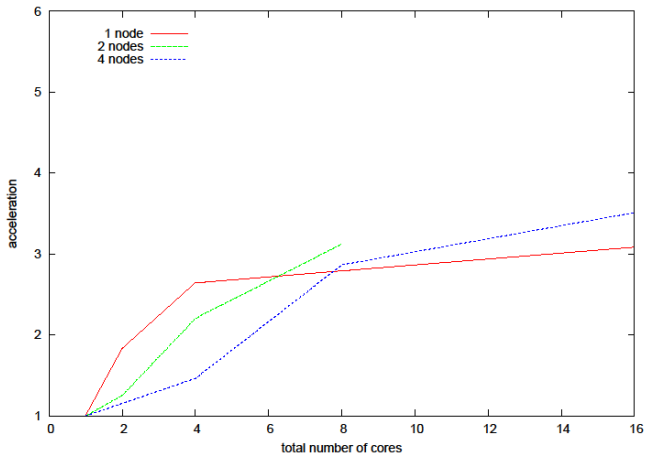
- 2012: compilation optimisation on new cluster (Emeraude)
  - Compilers Intel, PGI, Open64, PathScale
  - Tries with many compilation options
  - Intel delivers the best results
- 2013: new parallel tests on Emeraude (ACRi and HYPRE libraries)
  - ACRi library: hydraulic/transport coupled
    - + *Structured mesh*
    - + *8,45M elements*
    - + *CONJ/NEUM & BICG/NEUM*
  - ACRi library : hydraulic only
    - + *Structured mesh*
    - + *8,05M elements*
    - + *CONJ/NEUM*
  - ACRi library : transport only
    - + *Structured mesh*
    - + *8.05M elements*
    - + *BICG/NEUM*
  - HYPRE library: failure

Results with ACRi library

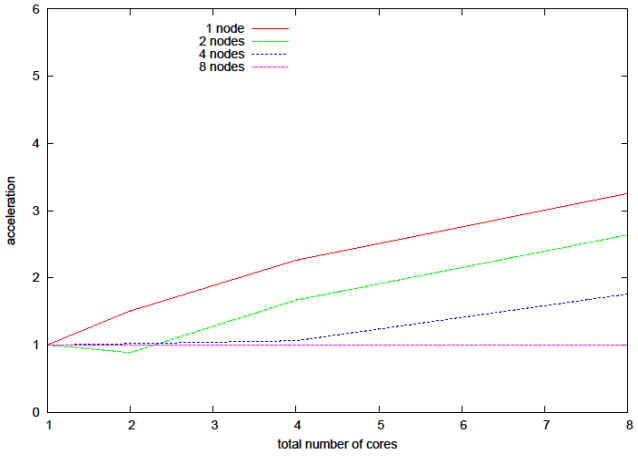
Hydraulic & transport coupled



Hydraulic only



Transport only



Two assumptions to be confirmed (1/2)

ACRi library

- OpenMP parallelisation too « basic »
- example (*fine grain**)

```
do iter=1, nbIter
      !$OMP DO SCHECULE(RUNTIME)
      do i=1,n
                        b(i)=a x b(i) + iter
      enddo
enddo
```

- could be (*coarse grain* with static load balancing*)

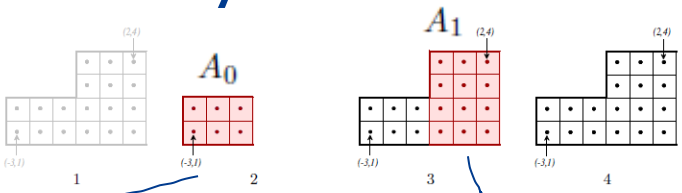
```
!$omprank = OMP_GET_THREAD_NUM()
!$nbthreads = OMP_GET_NUM_THREADS()
beg = 1+(omprank * k)/nbthreads
end = ((omprank+1)*k)/nbthreads
do iter=1, nbIter
      do i=beg,end
                        b(i) = a x b(i) + iter
      enddo
enddo
```

- … but very intrusive approach

* from http://www.idris.fr/data/cours/hybride/form_hybride_proj.pdf

Two assumptions to be confirmed (2/2)

HYPRE library




```
HYPRE_StructGrid grid;
int ndim        = 2;
int ilower[][2] = {{-3,1}, {0,1}};
int iupper[][2] = {{-1,2}, {2,4}};

/* Create the grid object */
1: HYPRE_StructGridCreate(MPI_COMM_WORLD, ndim, &grid);

/* Set grid extents for the first box */
2: HYPRE_StructGridSetExtents(grid, ilower[0], iupper[0]);

/* Set grid extents for the second box */
3: HYPRE_StructGridSetExtents(grid, ilower[1], iupper[1]);

/* Assemble the grid */
4: HYPRE_StructGridAssemble(grid);
```

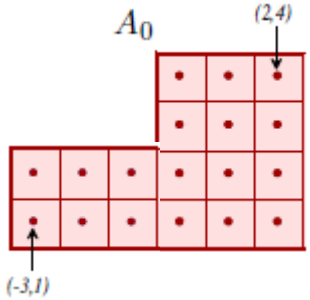
2 small blocks distributed on 2 processes

$$\begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{P-1} \end{bmatrix}$$

p small blocks, p processes

Solve p sub-problem and build the final solution

What Porflow is probably doing



1 big block distributed on 2 processes

$$\begin{bmatrix} A_0 \\ A_0 \\ \vdots \\ A_0 \end{bmatrix}$$

p big blocks, p processes

Solve p times the full problem and build the final solution

Tough2(-MP)

Hydraulic and transport in unsaturated porous media

From Tough2 to Tough2-MP

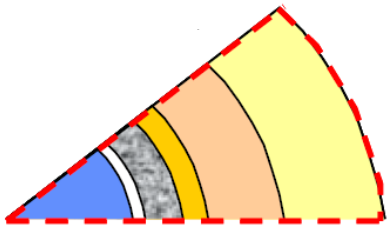
2005

Simplified picture of each component **(cell, drift) +COX**

1D/2D mesh: from **10^2 to 10^3 elements**
CPU time = 5 days

Zone de stockage
Vides résiduels
Béton
Zone fracturée
Zone micro fissurée
Cox sain

MAVL cell 1D/2D



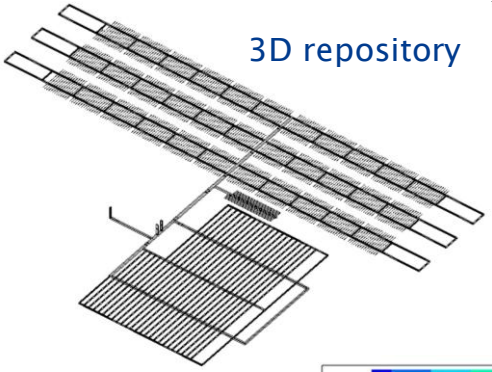
Tough2

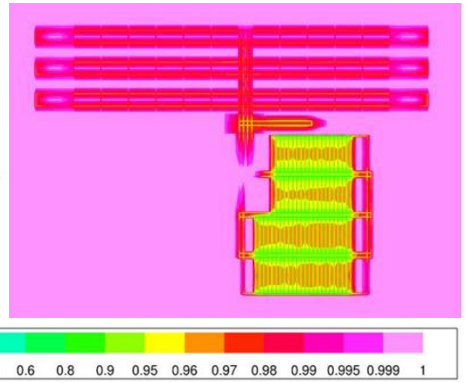
2012

Detailed picture of the repository (**cells** + drifts + wells) + COX

3D mesh: **300.000 elements**
CPU time = 15 days

3D repository





Tough2 //

Water saturation after 1000 years

Description and issues

Distributed memory parallelism

- Solvers from Aztec library
- Partitioning with Metis

Identified issues

- Some MPI processes stop to communicate together
  - Randomly happens
  - Could be correlated with the instantaneous load on the nodes
  - Behavior changes with the number of nodes or cores per node
- Time step collapses
  - Mainly when phase appears/disappears
- Save data on files is time consuming
  - Can not select the data to save

    + *Gb data files*

Solution

Bug with MPI ✓

- Bad label identification
  - ☐ One label could identify two different data block

Main algorithm problem ✓

- Adjust automatic time step algorithm
- Criteria for phase change is chosen by the user

MPI 2 for parallel Input/Output X

Phreeqc/Phast

Chemical transport in porous media

Analysis and optimization

Profiling

- One function* uses 75% of computation time and is called tooooo many times

Analysis

- Probably many under-optimized code due to f2c
- Architecture not suitable for parallelism
  - 1 table per property instead of 1 table per cell (with all cell's properties)

Optimization

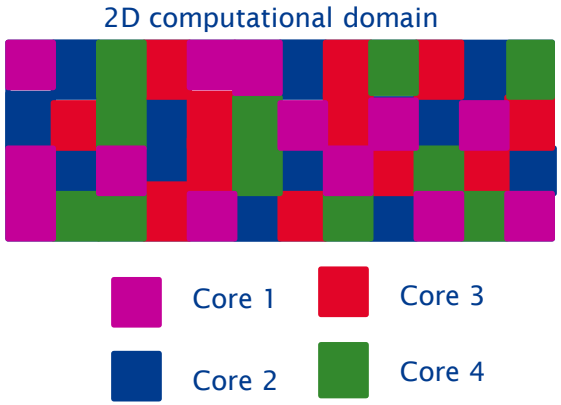
- Reorder Loop
  - Because of Fortran to C automatic conversion (f2c)
- Few rewrite to use OpenMP
- Overall gain around 5-10% only (limited on many cores because of overhead)

* Barrodale-Roberts modified simplex algorithm

Solution: Phast

Description

- PhreeqC (chemistry) + HST3D (transport)
  - Parallelisation through multiple PhreeqC processes
  - Random splitting
  - Each sub-set of cells is solved on one core



Pros

- Open Source and free*
- Straightforward parallelisation
  - One PhreeqC object per cell
- Absolute gain is ok (from 40 days to 10 days with 24 cores)

Cons

- Architecture still not optimized (would require too much work…)
- Issues from f2c probably remain
- Small acceleration (~0.16)

* http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/phast/

Conclusion

Future improvements

2013-…

- Porflow improvements
  - 64 bits integer (mesh > 18M elements)?
  - Optimize MPI and OpenMP parallelism?
- Better solvers for Tough2-MP
- Parallelism with Traces