ComPASS: a tool for parallel construction

Computing Parallel Architecture to Speed up Simulations

Cindy GUICHARD

Marseille - MoMaS 2012
Context and Applications

- **same pictures as Pascal and Roland ...**

- **.. same applications**: Reservoir simulations, CO₂ geological storages, ...

**Wish**: develop a code prototype with a parallel architecture to test promising formulation and schemes on realistic cases
Centre d’Eté Mathématique de Recherche Avancée en Calcul Scientifique

annual topic : Numerical Methods and Algorithms for High Performance Computing

included a 5 weeks research session, organised by project / team

team members of ComPASS :

Roland Masson, CG - Université de Nice and team COFFEE of INRIA
Pascal Havé - IFP Energies nouvelles
Eric Dalissier - Université de Grenoble
Chang Yang - Université de Lyon
Wei Zhang - Université d’Orsay

CEMRACS : starting point (from scratch) of the project

Objectives :

– Development of basic tools for parallel computing
– Working in a way of learning and formation
ComPASS: specifications and strategy during the CEMRACS

Specifications

- Parallel programming with MPI using Fortran 90
- General meshes (polyhedral cells, possibly non planar faces)
- Test of a large class of compact Finite Volume schemes (MPFA, VFH, ...)
  ⇒ one layer of ghosts cells
  ⇒ upon request connectivity (cell, face, node)
- Connected the code with scientific computing libraries (Metis, PETSc)
- Up to $10^7$ cells and 1000 CPU

Strategy

- Implementation of the heat equation for testing purposes
  \[ \partial_t u - \triangle u = 0 \]
  with an implicit first order time integration and VAG scheme for the diffusion term
Discrete Model using the VAG scheme \cite{Eymard,Cances,Herbin,RolandMaire}

- **DOF**: cell unknowns \( (u_K)_{K \in M} \) and vertices unknowns \( (u_s)_{s \in V} \)

- **Equation on cell** \( K \)

\[
|K| \left[ \frac{u_K^{n+1} - u_K^n}{\delta t} + \sum_{s \in V_K} \sum_{s' \in V_K} T_{K,s,s'}^s (u_K^{n+1} - u_{s'}^{n+1}) \right] = 0
\]

and **equation on vertex** \( s \)

\[
- \sum_{K \in M_s} \sum_{s' \in V_K} T_{K,s,s'}^s (u_K^{n+1} - u_{s'}^{n+1}) = 0
\]

where \( (T_{K,s,s'}^s) \) are coefficients of the VAG scheme

- **Cell-unknowns** are eliminated in the linear system

  \( \Rightarrow \) **compact vertex-centered scheme** (using a Schur complement)

- **Thanks to the barycentric cutting of the cell**

  \( \Rightarrow \) **can be applied** on **general meshes**
**Global to Local Mesh**

- Read the mesh file
- Global connectivity
- Partition using Metis:
  \[
  \text{Partition}(K) = \text{num proc}, \forall K \text{ in the global mesh}
  \]
- Distribution (one layer of ghosts): Cell, Face, Node following a given rule of belonging

**Synchronization**

- Each process
  - Reception of a local mesh
  - Switch form global id to local id
  - Local connectivity
Synchronization

How manage communications to retrieve ghost data?

Each proc. has a list of messages
Synchronization

How manage communications to retrieve ghost data?

Each proc. has a list of messages

```
<table>
<thead>
<tr>
<th>P0</th>
<th>send</th>
<th>recv</th>
<th>recv</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>send</td>
<td>P1</td>
<td>P2</td>
</tr>
<tr>
<td>deadlock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>send</td>
<td>send</td>
<td>recv</td>
</tr>
<tr>
<td></td>
<td>P0</td>
<td>P2</td>
<td>P0</td>
</tr>
<tr>
<td>P2</td>
<td>send</td>
<td>recv</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P0</td>
<td>P1</td>
<td></td>
</tr>
</tbody>
</table>
```

without rules for ordering ⇒ possible deadlock !!!

Rules

- each proc does its send/recv by increasing order of proc
- if 2 proc communicate, the one with the lowest rank send
How manage communications to retrieve ghost data?

list of messages by ordering
**Time loop preparations**

- Storage of VAG scheme coefficients $T_{K}^{s,s'}$
- Allocate the memory for the parallel matrix $A$ and vector $RHS$ with vertices as dof
- Initialization of $(u_s)$ and $(u_K)$

**Assembly**

- Construct local matrices $A_K$ and $RHS_K$
- Schur reduction and storage of the complement
- Assembly of $RHS$ and $A$
- Implicit communication by PETSc

**Resolution**

- Solve the linear system $(u_s)$ by Krylov method
- Reconstruct cell values $(u_K)$ by Schur complement
First result - Disturbed grid with $128^3 = 2,097,152$ cells

\[
\text{Speed Up (n proc.)} = \frac{\text{Time with 1 proc.}}{\text{Time with n proc.}}
\]
Short time perspectives

- Optimisation of the code
- Improved the post-processing
- Update the documentation
- Switch to 2-phase flow problems

Conclusion

- Thanks to CEMRACS: implementation of classical tools for parallel simulations of time-evolution non-linear problems
- Written by beginners ⇒ may be used by beginners: ready to share!