Application of Parallel Arrays for Parallelisation of Data Parallel Algorithms

A. Jakušev, V. Starikovičius
Contents

- Introduction;
- Tools;
- Parallel arrays – ParSol:
  - Features;
  - Structure and usage;
- Par-Sol applications:
  - Porous media;
  - Image smoothing;
- Conclusions.
Parallelization Benefits

- Bigger problems;
- Faster solution time;
- Economically effective;
- Better resource utilization;
Parallelization Difficulties

- Sequential code can’t be parallelized right away (generally):
  - Special method and algorithms required;
  - Knowledge of specific tools and technologies required;
- It takes (long) time to become good at parallelization;
- Not enough simple parallelization tools and technologies.
Parallel Programming Tools and Standards

- MPI (Message Passing Interface):
  - + For both C and FORTRAN;
  - + Portable and Widespread;
  - - Complicated;

- HPF (High Performance FORTRAN):
  - + Simple to parallelize;
  - - Fortran is less popular now;
  - - Separate FORTRAN compiler has to be developed.
Parallel Programming Tools and Standards - II

OpenMP:
- For both C/C++ and FORTRAN;
- Simple to parallelize;
- Gaining popularity;
- Well suited for shared memory systems;
- Not designed for distributed memory systems (ex: PC clusters);
- Special compiler required.
ParSol Features

- ParSol – parallel array library:
  - Written in C++;
  - Operates similar to HPF – simplicity;
  - Uses MPI – portability; No special compiler required;
  - Open-source (now).
ParSol Is Like HPF

- FORTRAN and HPF
  1. Write and debug serial version of program using certain restrictions
  2. Add processor topology comments
  3. Recompile code with HPF

- C++ and ParSol
  1. Write and debug serial version of program using certain restrictions
  2. Add/change some instructions:
     1. Create topology object with certain properties
     2. Change some class and header names
     3. Add global initialization and finalization code
     4. Specify when to exchange data among neighbor processes
  3. Recompile program (same compiler)
ParSol restrictions

- Use ParSol array classes instead of C++ arrays
  - very convenient restriction
  - natural in FORTRAN
- For every array, a stencil used to process array elements must be specified
- Array manipulations **must not depend** on order in which all array elements are processed
Main ParSol Elements

- **Sequential array classes** – these are the classes to be used instead of native C/C++ arrays even in sequential programmes;

- **Parallelisation and parallel array classes** – used instead of sequential classes in parallel versions of the program.
Main ParSol Elements – II

- **Topology classes** – the purpose of these classes is to ensure that all processes are in proper order for parallel array functionality;

- **Stencil classes** – stencil is determined depending on what computational scheme is used. Based on stencil, different amount of information needs to be exchanged among neighbours.
ParSol operation scheme

sequential

parallel
ParSol class diagram
ParSol vs. OpenMP

- Use whatever you are comfortable with 😊
- When ParSol is for you:
  - Have strong background in HPF;
  - Code portability is an issue;
  - Computations on distributed memory systems;
- ParSol effectiveness – depends on underlying MPI implementation.
Porous Media Mathematical Model

- **Mass conservation**
  \[
  \frac{\partial}{\partial t} \left( \Phi \rho_\alpha S_\alpha \right) + \nabla \cdot \left\{ \rho_\alpha \mathbf{u}_\alpha \right\} = \rho_\alpha q_\alpha
  \]

- **Darcy law**
  \[
  \mathbf{u}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K \left( \nabla p_\alpha - \rho_\alpha \mathbf{g} \right)
  \]

- **Capillary pressure**
  \[
  p_{c\beta\alpha}(\mathbf{x}, t) = p_\beta(\mathbf{x}, t) - p_\alpha(\mathbf{x}, t), \quad \beta \neq \alpha
  \]

- **Saturation**
  \[
  \sum_\alpha S_\alpha = 1
  \]
ParSol With Implicit Diffusion – SP4

\[ T_1(160) = 64.97, \quad T_1(320) = 241.4, \quad T_1(480) = 281.9 \]

Table 1: Implicit nonlinear diffusion algorithm on SP4

<table>
<thead>
<tr>
<th>( p )</th>
<th>( S_p(160) )</th>
<th>( E_p(160) )</th>
<th>( S_p(320) )</th>
<th>( E_p(320) )</th>
<th>( S_p(480) )</th>
<th>( E_p(480) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.42</td>
<td>1.211</td>
<td>2.78</td>
<td>1.392</td>
<td>2.38</td>
<td>1.189</td>
</tr>
<tr>
<td>4</td>
<td>5.04</td>
<td>1.260</td>
<td>5.98</td>
<td>1.495</td>
<td>4.41</td>
<td>1.102</td>
</tr>
<tr>
<td>6</td>
<td>7.03</td>
<td>1.172</td>
<td>8.97</td>
<td>1.495</td>
<td>6.58</td>
<td>1.097</td>
</tr>
<tr>
<td>8</td>
<td>8.56</td>
<td>1.070</td>
<td>11.30</td>
<td>1.412</td>
<td>8.69</td>
<td>1.086</td>
</tr>
<tr>
<td>16</td>
<td>13.45</td>
<td>0.841</td>
<td>23.44</td>
<td>1.465</td>
<td>17.15</td>
<td>1.072</td>
</tr>
</tbody>
</table>
ParSol With Implicit Diffusion – PC cluster

\[ T_1(188 \times 100) = 24.10, \quad T_1(350 \times 200) = 366.54. \]

Table 2: 3D Poisson equation, using CG method and 7-point stencil on PC cluster

<table>
<thead>
<tr>
<th></th>
<th>( S_p(188 \times 100) )</th>
<th>( E_p(188 \times 100) )</th>
<th>( S_p(350 \times 200) )</th>
<th>( E_p(350 \times 200) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.82</td>
<td>0.911</td>
<td>1.98</td>
<td>0.988</td>
</tr>
<tr>
<td>4</td>
<td>3.63</td>
<td>0.906</td>
<td>3.86</td>
<td>0.965</td>
</tr>
<tr>
<td>8</td>
<td>5.97</td>
<td>0.747</td>
<td>7.11</td>
<td>0.888</td>
</tr>
</tbody>
</table>
Image Smoothing Using Diffusion Filters

- **Space** – finite volume method;
- **Time** – explicit Euler method:

\[
\partial_t U_{ij}^{n+1} = \sum_{\alpha=1}^{2} \partial_{x_\alpha}^+ \left( a_\alpha \left( U_{ij}^n \right) \partial_{x_\alpha}^- U_{ij}^n \right) + f\left( u_{0,ij} - U_{ij}^n \right)
\]

\[
\partial_t U_{ij}^n = \frac{U_{ij}^n - U_{ij}^{n-1}}{\tau}, \quad \partial_{x_1}^+ U_{ij}^n = \frac{U_{i+1,j}^n - U_{ij}^n}{h}, \quad \partial_{x_1}^- U_{ij}^n = \frac{U_{i,j}^n - U_{i-1,j}^n}{h}
\]
ParSol With Explicit Diffusion – SP4

\[ T_1(80) = 57.24, \quad T_1(160) = 471.2, \quad T_1(320) = 770.4. \]

Table 4: The speedup and efficiency for explicit diffusion algorithm on SP4

<table>
<thead>
<tr>
<th>( p )</th>
<th>( S_p(80) )</th>
<th>( E_p(80) )</th>
<th>( S_p(160) )</th>
<th>( E_p(160) )</th>
<th>( S_p(320) )</th>
<th>( E_p(320) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.975</td>
<td>0.988</td>
<td>1.984</td>
<td>0.992</td>
<td>2.004</td>
<td>1.002</td>
</tr>
<tr>
<td>3</td>
<td>2.794</td>
<td>0.931</td>
<td>2.950</td>
<td>0.985</td>
<td>2.970</td>
<td>0.990</td>
</tr>
<tr>
<td>4</td>
<td>3.741</td>
<td>0.935</td>
<td>3.928</td>
<td>0.982</td>
<td>3.986</td>
<td>0.996</td>
</tr>
<tr>
<td>6</td>
<td>5.168</td>
<td>0.861</td>
<td>5.463</td>
<td>0.910</td>
<td>5.916</td>
<td>0.986</td>
</tr>
<tr>
<td>8</td>
<td>6.766</td>
<td>0.846</td>
<td>7.293</td>
<td>0.911</td>
<td>7.831</td>
<td>0.979</td>
</tr>
<tr>
<td>9</td>
<td>6.784</td>
<td>0.754</td>
<td>7.604</td>
<td>0.845</td>
<td>8.467</td>
<td>0.941</td>
</tr>
<tr>
<td>12</td>
<td>8.701</td>
<td>0.725</td>
<td>10.19</td>
<td>0.849</td>
<td>11.216</td>
<td>0.934</td>
</tr>
<tr>
<td>16</td>
<td>10.84</td>
<td>0.677</td>
<td>12.75</td>
<td>0.797</td>
<td>15.041</td>
<td>0.940</td>
</tr>
<tr>
<td>24</td>
<td>14.18</td>
<td>0.591</td>
<td>18.24</td>
<td>0.760</td>
<td>21.961</td>
<td>0.915</td>
</tr>
</tbody>
</table>
ParSol With Explicit Diffusion – PC cluster

\[ T(160) = 0.1, \quad T_1(160) = 213.3, \quad T(240) = 0.03, \quad T_1(240) = 332.8, \]
\[ T(320) = 0.01, \quad T_1(320) = 361.6. \]

Table 3: The speedup and efficiency for explicit diffusion algorithm on PC cluster

<table>
<thead>
<tr>
<th>( p )</th>
<th>( S_p(160) )</th>
<th>( E_p(160) )</th>
<th>( S_p(240) )</th>
<th>( E_p(240) )</th>
<th>( S_p(320) )</th>
<th>( E_p(320) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.56</td>
<td>0.780</td>
<td>1.76</td>
<td>0.880</td>
<td>1.87</td>
<td>0.934</td>
</tr>
<tr>
<td>4</td>
<td>2.36</td>
<td>0.590</td>
<td>3.00</td>
<td>0.750</td>
<td>3.45</td>
<td>0.862</td>
</tr>
<tr>
<td>6</td>
<td>2.78</td>
<td>0.463</td>
<td>3.93</td>
<td>0.655</td>
<td>4.77</td>
<td>0.795</td>
</tr>
<tr>
<td>8</td>
<td>2.95</td>
<td>0.369</td>
<td>4.69</td>
<td>0.585</td>
<td>5.88</td>
<td>0.735</td>
</tr>
<tr>
<td>9</td>
<td>3.16</td>
<td>0.351</td>
<td>5.04</td>
<td>0.560</td>
<td>6.28</td>
<td>0.698</td>
</tr>
<tr>
<td>11</td>
<td>3.33</td>
<td>0.303</td>
<td>5.50</td>
<td>0.500</td>
<td>7.09</td>
<td>0.644</td>
</tr>
<tr>
<td>12</td>
<td>3.35</td>
<td>0.279</td>
<td>5.64</td>
<td>0.470</td>
<td>7.47</td>
<td>0.623</td>
</tr>
<tr>
<td>15</td>
<td>3.39</td>
<td>0.226</td>
<td>6.38</td>
<td>0.425</td>
<td>8.56</td>
<td>0.571</td>
</tr>
</tbody>
</table>
Conclusions

- ParSol – a way to quickly parallelize data parallel algorithms;
- ParSol is well suited for distributed memory systems;
- ParSol gives good parallelization efficiency for diffusion algorithms.
THANK YOU

Any questions?