The Recent Work in Earth System Modeling

Xiaoge Wang
Department of Computer Science and Technology
Tsinghua University
Beijing, China
October 22, 2010
Outline

- Background
- Some Results
Background

Earth science application (funded by 863 plan, 2010-2012)

- Infrastructure for Earth science research
  - Parallel Coupler
  - Parallel I/O and Data Management
  - Parallel Visualization
  - Resource Management

- Performance acceleration
  - Performance Tuning for existing programs
  - Development of new algorithms
Some Results

- **Parallel Coupler Design**
  - C-Coupler

- **Performance Tuning**
  - Gamil 1x1

- **New Parallel Algorithms**
  - Interpolation
  - Data Fitting
Some Results

- **Parallel Coupler Design**
  - C-Coupler

- **Performance Tuning**
  - For Gamil 1x1

- **New Parallel Algorithms**
  - Interpolation
  - Data Fitting
Objectives of the Coupling Tools

- Efficient data movement
- Fast transformation (include remapping, merging, etc.)
- Coordination between coupled model component
- Assist coupled system design and implementation
CCSM coupler

![Diagram of CCSM coupler](image)

- Ice Model
- Atmosphere Model
- Coupler
- Ocean Model
- Land Model
Recent Work

- The coupling framework for earth system modeling
- Repository of coupling functions
  - An interpolation algorithm
  - A non-negative linear least squares solver
Philosophy of the Design

- Allow independent development of different sub-models
- Interactions of sub-models are not pre-determined and can be defined/modified by the multi-model system developers with the coupling tools
- Encourage resource sharing and collaborations across research institutes
- Coupling overhead should be limited and capable of system level performance management
C-coupler: A Framework

Coupling Enabled Model components

Repository of Coupling functions and tools
Define the couplings:
(1) source and/target 
(2) Coupling fields 
(3) Type of coupling functions 
(4) Coupling frequency 

Select coupling functions, algorithms and parameters 

Select diagnosis level and actions 

Case setup 

Front end

Select component models 
Select coupler grid 

Validate model interfaces and grid selection and parameters 

Validate the selection, collect the necessary information of coupling functions and coupling frequency 

Validate the selection and collect information 

Validate and complete information, start generate the driver and interface codes 

Back end

Code generation
Some Results

- **Parallel Coupler Design**
  - C-Coupler

- **Performance Tuning**
  - Gamil 1x1

- **New Parallel Algorithms**
  - Interpolation
  - Data Fitting
Performance Tuning

- **Gamil 1x1**
  Atmosphere model, resolution 1°, used in ICPP experiment.

- **Performance improvement**
  - Some preliminary test on DeepComp7000;

<table>
<thead>
<tr>
<th></th>
<th>before</th>
<th>after</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallelism</td>
<td>MPI only</td>
<td>MPI + OpenMP</td>
</tr>
<tr>
<td>scalability</td>
<td>Use up to 60 cores</td>
<td>Use up to 480 cores</td>
</tr>
<tr>
<td>speedup</td>
<td>———</td>
<td>5-6</td>
</tr>
</tbody>
</table>
Some Results

- **Parallel Coupler Design**
  - C-Coupler

- **Performance Tuning**
  - Gamil 1x1

- **New Parallel Algorithms**
  - Interpolation
  - Data Fitting
Objective:

- Develop additional and more stable method
- 2D interpolation: Apply cubic spline on each direction (horizontal and vertical)
- For remapping between two Atmosphere and Ocean models in the earth system.
- It is an addition to SCRIP
Experiment

- Machine: Intel(R) Xeon(R) E5520 @ 2.27GHz
- Compiler: icpc version 11.1 with –O3 –openmp option
- Test data:
  - Grids from Atmosphere and Ocean model (next page)
  - Functions:
    1. $2 + \cos(5 \arccos(-\cos x \cos y))$
    2. $2 + \cos^2 y \sin 2x$
    3. $2 + \sin^{16} 2y \cos 16x$
## Grid of Test Case

<table>
<thead>
<tr>
<th>case</th>
<th>Source Grid</th>
<th>Target grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gamil_360x180_Grid</td>
<td>R42_Gaussian_Grid</td>
</tr>
<tr>
<td>2</td>
<td>R42_Gaussian_Grid</td>
<td>Gamil_360x180_Grid</td>
</tr>
<tr>
<td>3</td>
<td>Gamil_360x180_Grid</td>
<td>T85_Gaussian_Grid</td>
</tr>
<tr>
<td>4</td>
<td>T85_Gaussian_Grid</td>
<td>Gamil_360x180_Grid</td>
</tr>
<tr>
<td>5</td>
<td>Gamil_360x180_Grid</td>
<td>T62_Gaussian_Grid</td>
</tr>
<tr>
<td>6</td>
<td>T62_Gaussian_Grid</td>
<td>Gamil_360x180_Grid</td>
</tr>
<tr>
<td>7</td>
<td>Gamil_360x180_Grid</td>
<td>T42_Gaussian_Grid</td>
</tr>
<tr>
<td>8</td>
<td>T42_Gaussian_Grid</td>
<td>Gamil_360x180_Grid</td>
</tr>
<tr>
<td>9</td>
<td>Gamil_360x180_Grid</td>
<td>2x25d_CAM2_finite_volume_grid</td>
</tr>
<tr>
<td>10</td>
<td>2x25d_CAM2_finite_volume_grid</td>
<td>Gamil_360x180_Grid</td>
</tr>
</tbody>
</table>
Preliminary Results

Comparison of time of weight matrix generation

- Bilinear
- Distwgt
- Spline
Comparison of matrix-vector multiplication step

- bilinear
- distwgt
- spline
Some Results

- **Parallel Coupler Design**
  - C-Coupler

- **Performance Tuning**
  - Gamil 1x1

- **New Parallel Algorithms**
  - Interpolation
  - Data Fitting
Non-negative LLS

- Used in coupling simulations of molecular dynamics and the first principle simulations.
- LLS with the constrain on the solution $\mathbf{x} \geq 0$.
- Matrix $A$ is large and sparse. Sparsity is destroyed after a few steps. So the data is stored as full matrix.
- Only consider the solver on shared memory multi-core processor.
Algorithm Design

- Start from classical method in the book by Lawson and Hanson.
- Explore the possibility to maximize the computation of level 3 BLAS.
Classical Algorithm

- **Good references and easy to understand**
  - Book
  - *matlab* code

- **Structure of the algorithm**
  - Iteration until solution is found
    - Select a column into active set of columns
    - Remove column correspondent to the negative element of solution from active set
    - Apply LLS on active set of columns
  - Level 2 BLAS is dominant
\[ \text{activeset} \leftarrow \emptyset \]
\[ A' \leftarrow [A \ y] \]
\[ x \leftarrow 0 \]
\[ \text{while } (\exists i \notin \text{activeset}, \nabla f(x_i) < 0) \]
\[ \text{find } t \notin \text{activeset} \text{ that minimize } \nabla f(x_i) \]
\[ \text{activeset} \leftarrow \text{activeset} + \{t\} \]
\[ v \leftarrow \text{Householder vector for column } t \text{ in } A' \]
\[ A' \leftarrow (I - 2vv^T)A' \]
\[ x' \leftarrow \text{Solve LLS based on } A' \]
\[ \text{while } (\exists i \in \text{activeset}, x'_i \leq 0) \]
\[ \text{find maximum } \alpha \text{ that makes } x + \alpha(x' - x) \geq 0 \]
\[ x \leftarrow x + \alpha(x' - x) \]
\[ \text{zero} = \{i \mid i \in \text{activeset}, x_i \leq 0\} \]
\[ \text{activeset} \leftarrow \text{activeset} - \text{zero} \]
\[ \text{update } A' \text{ by erasing columns in } \text{zero} \text{ with Givens rotations} \]
\[ x' \leftarrow \text{Solve LLS based on } A' \]
\[ \text{endwhile} \]
\[ x \leftarrow x' \]
\[ \text{endwhile} \]
Main idea:
- Single column QR update is not efficient for multi-core
- Try to update QR in block fashion.

Aggregated updating:
\begin{verbatim}
activeset ← ∅ 
A' ← [A  y] 
x ← 0,  V ← 0,  T ← 0,  now ← 0 
while (∃i ∉ activeset, ∇f(x_i) < 0) 
   find t ∈ activeset that minimize ∇f(x_t) 
   activeset ← activeset + {t} 
   tmp ← (I - VTV^T)^T a_t 
   v ← Householder vector for tmp 
   T ← \begin{bmatrix} T & -TV^T v 
                    0 & 2 \end{bmatrix},  
   V ← [V  v] 
   if (now ≥ b) 
      A' ← (I - VTV^T)^T A' 
      V ← 0,  T ← 0,  now ← 0 
   endif 
   x' ← Solve LLS based on A'
\end{verbatim}
while (∃i ∈ activeset, x_i' ≤ 0)
    find maximum α that makes \( x + \alpha(x' - x) \geq 0 \)
    \( x \leftarrow x + \alpha(x' - x) \)
    zero = \{i | i \in activeset, x_i \leq 0\}
    activeset \leftarrow activeset - zero
    A' \leftarrow (I - VTV^T)^T A'
    V \leftarrow 0, \quad T \leftarrow 0, \quad now \leftarrow 0
    update A' by erasing columns in zero with Givens rotations
    x' \leftarrow \text{Solve LLS based on } A'
endwhile
    \( x \leftarrow x' \)
endwhile
More Refinements

- Select multiple columns each iteration
- Blocking in updating intermediate data V and T
Preliminary Experiment

Machine:
1. Intel core2 2.94Ghz, 4 cores
2. SGI machine with 64 cores

Compiler: GCC with –O3 and –OpenMP options

Test matrix: random matrix and matrices from U of Florida,

LAPACK function dgeqrf is used for scalability comparison
Effectiveness of Blocking:

- Compare two blocking algorithm with straightforward implementation of classical algorithm on two test matrices on 4-core machine

<table>
<thead>
<tr>
<th># cores</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000 × 3600 random matrix</td>
<td>straight</td>
<td>3m6.5s</td>
<td>2m45.7s</td>
<td>2m49.8s</td>
<td>2m42.7s</td>
</tr>
<tr>
<td></td>
<td>Blocking</td>
<td>19.1s</td>
<td>12.1s</td>
<td>9.6s</td>
<td>7.9s</td>
</tr>
<tr>
<td>5224 × 3005 from U of F</td>
<td>straight</td>
<td>2m22.7s</td>
<td>2m6.8s</td>
<td>2m9.9s</td>
<td>2m5.2s</td>
</tr>
<tr>
<td></td>
<td>Blocking</td>
<td>26.5s</td>
<td>20.0s</td>
<td>18.7s</td>
<td>17.8s</td>
</tr>
</tbody>
</table>
Scalability Study

Compare with LAPACK on the same test matrix
QUESTION?
THANKS!