A Software Developing Environment for Earth System Modeling

Depei Qian
Beihang University
CScADS Workshop, Snowbird, Utah
June 27, 2012
Outline

- Motivation
- Purpose and Significance
- Research Contents
- Technology Roadmap
- IDE design & development functions
- ESM Module Encapsulation
- Runtime Support
- Deployment
Motivation

CCSM/CESM

- Coupled climate models used to simulate the Earth's climate system
- Includes four separate models: the atmosphere, oceans, land surface and sea ice
- Support the research of global climate change in the past, present and future
Motivation (cont‘d)

Lack of integrated HPC software development environment for earth system modeling

- Primitive program development approaches
- Various model development tools
- Program development coupled with machine hardware / software details

Graphical IDE is in urgent need
1. All-in-one platform, comprehensive tool library and the module / template library, to improve development efficiency
2. Decoupled with software/hardware details to allow researchers to focus on model development
3. Customized development environment to provide lifecycle support in developing the earth system models

Lack of integrated HPC software development environment for earth system modeling

Motivation (cont’d)
Outline

- Motivation
- Purpose and Significance
- Research Contents
- Technology Roadmap
- IDE design & development functions
- ESM Module Encapsulation
- Runtime Support
- Deployment
Purpose and significance

- Focus on support to the development of earth system models
- Build an integrated development platform for research on global climate change

- Computing Resources Abstraction
  - Abstraction of SW/HW details
  - Satisfy model developers' specific needs

- Scalability
  - Reduce the cost of switching between tools and improve efficiency

- All-In-One Platform
  - Lifecycle Management
    - Support the complete lifecycle of earth system modeling
Outline

- Motivation
- Purpose and Significance
- Research Contents
- Technology Roadmap
- IDE design & development functions
- ESM Module Encapsulation
- Runtime Support
- Deployment
Research content

Architecture of the all-in-one IDE

- Integrating development tools by plug-ins
- Template-based graphical development library
- Tool library for developing earth system models
- Demo Application developed by using IDE
Outline

- Motivation
- Purpose and Significance
- Research Contents
- Technology Roadmap
- IDE design & development functions
- ESM Module Encapsulation
- Runtime Support
- Deployment
Technology roadmap

Key Technologies

- Remote code organization and management
- Reusable code blocks for model development
- Description of heterogeneous HPC resources
- Parallel support, e.g., MPI & JASMIN
- Portability, scalability and feasibility of platform
- Remote compilation, debugging and execution
- Graphical drag&drop-based program coding
- Integrating existing tools

Research outcomes

- Module / Template Library
- Tool Library
- IDE interface

Plug-in Integration & Graphical interface support

Integrated Development Platform
Based on the open source Eclipse: mature community, abundant references, plenty of free plug-ins, wide range of languages

Overall strategy

- **Photran Code assist**
- **PTP Parallel support**
- **RDT Remote Support**
- **RSE Remote client**
- **EMF Graphic modeling code generation**
IDE design & development functions

- Works have been done:
  - IDE overall design
  - Fortran code assistant
  - Source code version control
  - Remote online multi-collaboration
  - Integration of other useful tools
IDE overall design

IDE Platform
- Workbench
- Editor
- Version Control
- Perspective
- Debug

Platform Runtime

ESM IDE

Java Development Tools (JDT)

Plug-in Development Environment (PDE)

MPMD Program Diagnosis

Visualization

Model Data Extraction
Fortran code assistant

ESM Perspective

VI Edit
Highlighted & File Layout

Fortran Subroutine Outline
Source code version control
Remote online multi-collaboration
Integrated tools
Remote function invocation

Remote editing, compilation, debugging and execution

File Services
Indexing Service
Compiling Service
Launch Service
Debugging Service
Remote Develop

Source Code

edit
search/index/navigator
build
executable
run
debug

Computing

Local Machine

Remote Machine
Outline

- Motivation
- Purpose and Significance
- Research Contents
- Technology Roadmap
- IDE design & development functions
- ESM Module Encapsulation
- Runtime Support
- Deployment
ESM module encapsulation

- Encapsulation of reusable code blocks for model development

- Radiation Module
- Time integration module
- Boundary layer module
- Solver module
- Coupler module

Modular Encapsulation Specification

- module unit
  - High-performance
  - Reusable

- model module library
  - Enrichment
  - Optimization
Program decomposition

- Decomposition of POP (Parallel Ocean Program) into modules
Resource allocation

- Parallel computing resources allocation for POP components
Dynamic process organization

- Dynamic process organization based on grid partition
Validation and performance evaluation

Functional validation and performance evaluation

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>POP (s)</th>
<th>CPOP (s)</th>
<th>K.E Diagnosis</th>
<th>Tracer Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7482</td>
<td>7856</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>2</td>
<td>4042</td>
<td>4243</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>4</td>
<td>1981</td>
<td>2083</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>8</td>
<td>1191</td>
<td>1250</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>16</td>
<td>1082</td>
<td>1130</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>32</td>
<td>1238</td>
<td>1242</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>64</td>
<td>1257</td>
<td>1251</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>128</td>
<td>1534</td>
<td>1506</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>256</td>
<td>2582</td>
<td>2453</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
Code generation

- Graphic Drag&Drop Editor
- Code Auto-Generation
- Module Repository
- Module Property
Component specification

- Specification for component encapsulation
  - Templates to guide component development
  - Each component is associated with a xml-based description file before importing to the component library
Component specification (cont’d)
Code generation (cont’d)

- Drag-Drop based component usage
  - Automatically locate and intercept the information about the component based on CDL, and then fill in the panel of attributes about this component
  - Component substitution can be done easily through selection
Code generation (cont’d)

[Image of code generation interface with code snippets and comments]
Module auto-verification

- Primary auto-verification for module validity
Model components substitution

- Model component encapsulation allows the components developed by different groups or communities to be inter-changeable, reducing duplicated programming efforts
  - Component substitution is demonstrated between POP and MOM
  - MOM
    - a three-dimensional ocean circulation model designed primarily by the Geophysical Fluid Dynamics Laboratory (GFDL) of the National Oceanic and Atmospheric Administration
  - POP
    - developed and supported by Los Alamos National Laboratory (LANL)
POP vs MOM

Framework Comparison

- POP is based on ESMF, while MOM is based on FMS

**ESMF**

- Components are executed sequentially
- All components reside in flat
- Single coupler
- Data is transferred through shared public variables

**MOM**

- Child components are executed simultaneously
- Invocation among components is nested
- Multiple couplers
- Data is transferred through data structure called state
MOM code structure analysis
MOM code structure analysis

- Analyzing the code structure of MOM
  - MOM component interface
    - `init()` #initialize and configure
    - `run()`
      - `update()` #update data, time step, status and so on
      - `restart()` #reload configuration and restart
    - `final()`: #release the resources

The above interface would make sure that the components derived from MOM, which is based on FMS specification, are compatible with components derived from POP which is based on ESMF, so that the components are substitutable.
Outline

- Motivation
- Purpose and Significance
- Research Contents
- Technology Roadmap
- IDE design & development functions
- ESM Module Encapsulation
- Runtime Support
- Deployment
Runtime support

- Work have been done:
  - Seamless integration with Slurm batch system with an easy-to-use management portal
  - Support ESM program execution monitoring and trace analysis
Integration with the batch system

- Integration with Condor to provide **persistent execution environment**
- The jobs submitted to the IDE are delegated to underlying slurm batch system
  - Extend condor to talk to slurm
  - Submit, query and management
Program execution monitoring

Resource Monitor

Job Submit

Job Query
Trace analysis based on STAT
Trace analysis based on STAT (cont’d)
Outline

Motivation

Purpose and Significance

Research Contents

Technology Roadmap

IDE design & development functions

ESM Module Encapsulation

Runtime Support

Deployment
Deployment

- Alpha version released (March 2011)
- Five versions (v0.2, v0.4, v0.6, v1.0, v2.0) released so far
- Deployed in Beijing Normal University (since March 2011)
  - Pilot usage of four versions (v0.4, v0.6, v1.0, v2.0)
  - Used in BNU_ESM development
  - Latest version is v2.0
- Deployed in Institute of Atmospheric Physics, Chinese Academy of Sciences (since March 2011)
  - Latest version is v2.0
- Deployed in National Climate Center (since June 2011)
  - Latest version is v2.0
- IDE and model encapsulation approach adopted by the coupler development project (since July 2011)
Thank You!