OMUSE: Oceanographic Multipurpose Software Environment

Henk Dijkstra (IMAU)
Simon Portegies Zwart (Sterrewacht Leiden)
Adam Candy (TUDelft)
Jan Viebahn (IMAU)
Arjen van Elteren (Sterrewacht Leiden)
Ben van Werkhoven (Netherlands eScience Center)
Inti Pelupessy (IMAU and Sterrewacht Leiden)
AMUSE -> OMUSE

Transfer of model coupling technology from Computational Astrophysics to Environmental Computing

- Radiative transfer
- Gravitational dynamics
- AMUSE
- Stellar evolution
- Hydrodynamics
- Spectral wave models
- OMUSE
- General circulation models
- Regional shallow water models
- Atmospheric models
OMUSE: Philosophy

- Build on existing codes, and use them ‘as is’
- Couple using physical interfaces
- Hide implementation and algorithmic details
- Standardize interfaces
- Automate as much as possible
- Test everything
OMUSE: Overview

- Python
- High-level object-oriented interfaces
- Enforces the use of units
- Integrated support for grids and remapping
- State model manages simulations
Example of using OMUSE

A simple Python script to start a run of the Parallel Ocean Program on a compute cluster:

```python
from omuse.community.pop.interface import POP
from amuse.units import units
from distributed_amuse import init_das5_only
from matplotlib import pyplot
init_das5_only("username", num_nodes=56, num_cores=16)

p = POP(channel_type="distributed", mode="3600x2400x42", number_of_workers=num_nodes*num_cores)
... #set the input files for POP

p.evolve_model(p.get_model_time() + (1.0 | units.day))

sst = p.elements.temperature.value_in(units.C).T
pyplot.imshow(sst, origin='lower')
pyplot.show()
```
OMUSE interfaces to models

High-level Python Interface
- Object-Oriented Interface
- State machine
- High-level data types
- Model data with units

Low-level Python Interface
- Interface to remote functions
- Units of returned data

Channel
- MPI-based
- Socket-based
- Distributed (Xenon-based)

Code Interface
- Implements interface to code
- State transitions
- Getters Setters
- Running the model
- Written in code’s language
- Compiled together with the code

Community Code
- All of original except “main()”
- Patched only if necessary
State model

The high-level interface specifies:

- The model states
- State transition functions
- Method-state mappings
Integrated support for units

- When interacting with a code, all data has a unit
- A value with a unit is a quantity
- Example:
  ```
  >>> distance = 1.0 | units.km
  >>> distance
  quantity<1.0 km>
  >>> speed = 5 | units.m / units.s
  >>> speed
  quantity<5 m / s>
  >>> time = distance / speed
  >>> time.value_in(units.s)
  200.0
  >>> time.as_quantity_in(units.hour)
  quantity<0.0555555555556 hr>
  ```
Grid support

Grid datatype hierarchy

- Grid
  - StructuredGrid
  - RectilinearGrid
  - RegularGrid
  - CartesianGrid
  - Unstructured Grid

Staggered grids
Grid Remapping

Conservative Spherical Remapper

Climate Data Operators (CDO)
Example of grid remapping

remapper = conservative_spherical_remapper(pop.elements, adcirc.elements)
remapper.forward_mapping(["temperature"])
Online Data Analysis

- With OMUSE it is very easy to compose applications out of multiple simulations and/or analysis programs:

```python
p = POP(... )  # start POP as you would do normally
tracker = EddyTracker(grid=p.nodes, domain='Regional',
                       lonmin=0., lonmax=50., latmin=-45., latmax=-20., days_between=7)

tend = p.get_model_time() + (7.0 | units.day)
stop_time = p.get_model_time() + (1 | units.yr)
while (tend < stop_time):
    p.evolve_model(tend)
    tracker.find_eddies(ssh=p.nodes.ssh, rtime=p.get_model_time())
    tend = p.get_model_time() + (days_between | units.day)

tracker.stop(tend)
p.stop()
```
Online Eddy Tracking
Future work

Research projects
• Modeling future coastal erosion in the Caribbean
• Modeling current and wave fields of Kraken Mare on Titan
• Cloud-resolving
• Southern ocean mode

Technical development
• Distributed grids
• Create interfaces for new codes
Conclusions

Flexible model coupling framework:
• High-level object-oriented interfaces
• High-level data structures and state models
• Integrated support for units and grid remapping

OMUSE is freely available at: https://bitbucket.org/omuse/omuse
• Tests and examples included
• Open-source license (GPL3)

Questions?

Offline questions: b.vanwerkhoven@esciencecenter.nl