OASIS3-MCT & Open-PALM: 2 open source codes couplers

https://verc.enes.org/oasis/
http://www.cerfacs.fr/globc/PALM_WEB/

Anthony Thevenin (thevenin@cerfacs.fr)
Why coupling scientific computing codes?

- To treat a global system
  e.g. ocean - atmosphere coupling for climate modeling

- To set up new applications from existing codes
  e.g. fluid - structure coupling for the research of the optimal position of the intakes in a combustion chamber

- To build modular applications by splitting complex problems and assembling their elementary components
  e.g. data assimilation
**Code coupling issues**

- **What does it imply?**
  - Drive the information exchanges between codes
  - Manage codes execution (serial or parallel)

**DATA exchanges**

- **Commitments:**
  - Easy to implement: *non intrusive interfaces*
  - Efficiency: *no loss in performances*
  - Portability: *standard technical solutions*
  - Flexibility: *test different configurations with a few changes*
  - Genericity: *reuse components in other couplings*
Technical coupling solutions

- First solution: code merging

Code 1: ocean model

Program prog1
...
Call sub_prog2(in, out)
...
end

Code 2: atmosphere model

Program prog2
Subroutine sub_prog2(in, out)
...
end

- Very efficient (memory exchange) and portable
- BUT:
  - integration problems (no independency between codes)
  - No flexibility (problems in maintenance, evolution, …)
  - Potential memory wastes
  - parallelism of the master code imposed to the application

- This is merging and not coupling
Technical coupling solutions

- Second solution: use a communication protocol (MPI, CORBA, files, …)

<table>
<thead>
<tr>
<th>Code 1: ocean model</th>
<th>Code 2: atmosphere model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program prog1</td>
<td>Program prog2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Call XXX_send(prog2, data)</td>
<td>Call XXX_recv(prog1, data)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>end</td>
<td>end</td>
</tr>
</tbody>
</table>

- Can be very efficient and preserve existing codes (parallelism)
- But:
  - Coupling is not generic
  - Good experience in parallel computing required
  - Lack of flexibility (interpolation, transformation, …)
  - Not always portable depending on the chosen mechanism
  - Too complex with more than two codes and many exchanges
Third solution: choose a coupler (or a coupling library)

- A coupler is a software allowing data exchanges between several codes and managing their executions.
- It starts the executables in a sequence or in parallel.
- Data exchanges are invoked by general PUT/GET primitives.
- It uses a portable and efficient communication protocol (MPI).
- It provides tools:
  - time and spatial interpolations,
  - parallel data redistribution,
  - monitoring, performance analysis.
Technical coupling solutions

 Static coupling (OASIS)

 Codes are started at the beginning of the simulation and exit together at the end of the application.

 Dynamic coupling (Open-PALM)

 A component can be launched and can release resources upon termination at any moment during the simulation. Computing resources (required memory and the number of concurrent processors) are handled by the coupler.

 Programs can be executed in loops or under logical conditions (faculty of describing complex coupling algorithms).
Managing data exchanges between components: end-point communications

When a code produces a data potentially interesting for other components, it informs the coupler:
- appeal to a generic Put primitive (non-blocking)

When a calculation code needs data, it tells the coupler that it is waiting for some information:
- appeal to a generic Get primitive (blocking)

Total independence between the codes:
- NO explicit indication of the receiving units on the producer side, NOR of the producer on the receivers side
- replacement of a code by another one without interfering with the rest the application.
Technical coupling solutions

➡ The coupler: probably the best solution to couple independently developed code

➡ At run time, the component models remain separate executables and the coupler acts as a communication library linked to the models

➡ Two levels of parallelism:
   ➡ distributed components (codes can be parallel programs)
   ➡ task parallelism (codes can be executed in parallel)

➡ BUT:
   ➡ multi-executable: possible waste of resources if sequential execution of the components is enforced
   ➡ multi-executable: more difficult to debug; harder to manage for the OS
   ➡ more or less efficient depending on the coupling algorithm
OASIS3 – MCT, an open source codes coupler for massively parallel climate modeling

https://verc.enes.org/oasis/
PRISM: Partnership for Research Infrastructure in Earth System Modeling

IS-ENES: InfraStructure for ENES: European project

Current developers: 2 permanents (CERFACS, CNRS) & 1 consultant

All sources are written in F90 and C

Open source product, distributed under a LGPL license

All external libraries used are public domain: MPI, NetCDF; or open source:
SCRIP (Los Alamos National Laboratory), MCT (Argonne National Laboratory)
About 35 groups world-wide (climate modelling or operational monthly/seasonal forecasting):

- France: CERFACS, METEO-FRANCE, IPSL (LOCEAN, LMD, LSCE), OMP, LGGE, IFREMER
- Europe: ECMWF + Ec-Earth community
- Germany: MPI-M, IFM-GEOMAR, HZG, U. Frankfurt
- UK: MetOffice, NCAS/U. Reading, ICL
- Denmark: DMI
- Norway: U. Bergen
- Sweden: SMHI, U. Lund
- Ireland: ICHEC, NUI Galway
- The Netherland: KNMI
- Switzerland: ETH Zurich
- Italy: INGV, ENEA, CASPUR
- Czech_Republic :CHMI
- Spain: U. Castilla
- Tunisia: Inst. Nat. Met
- Japan: JMA, JAMSTEC
- China: IAP-CAS, Met. Nat. Centre, SCSIO
- Korea: KMA
- Australia: CSIRO
- New Zealand: NIWA
- Canada: RPN-Environment Canada, UQAM
- USA: Oregon State U., Hawaii U., JPL, MIT
- Peru: IGP

OASIS3 is used in 5 European ESMs that participate in IPCC AR5
To use OASIS3-MCT:
- Register on the website and download the sources
- Compile and run the tutorial on the target platform

To configure OASIS3-MCT:
- Identify the component models and their grids
- Identify the coupling fields to be exchanged between those models
- Identify the coupling parameters:
  - total duration
  - sources and targets symbolic names
  - exchange period
  - field transformations and interpolations
- Write a full namcouple configuration file
- Insert calls of the OASIS3-MCT communication library

And then:
- Compile OASIS3-MCT and link the component models with it
- Start the models and let OASIS3-MCT manage the coupling exchanges
Component model instrumentation:

- Initialization: call oasis_init_comp(...)
- Grid definition: call oasis_write_grid(...)
- Local partition definition: call oasis_def_partition(...) 
- Coupling field declaration: call oasis_def_var(...) 
- End of definition phase: call oasis_enddef(...) 

Coupling field exchange: in model time stepping loop:

- call oasis_put(..., date, field, ...) 
- call oasis_get(..., date, field, ...) 

  - user’s defined source or target 
  - sending or receiving at appropriate time only 
  - automatic averaging / accumulation if requested 
  - automatic writing of coupling restart file at the end of run 

Termination: call oasis_terminate(...)
**Communication:**

- Fully parallel communications between parallel models, based on MPI.

- Interpolation of the coupling fields performed directly on the source or target component processes.

- Parallel redistribution of the coupling fields done directly from the source to the target component processes. *without gathering the whole coupling field on an additional process as in the previous OASIS3 versions (no bottleneck)*

**I/O functionality:**

- Switch between coupled and forced mode.
OASIS3 - MCT

- Interpolations & transformations:
  - On 2D vector fields
  - On different types of grids:
    - lat-lon, rotated, gaussian reduced, unstructured

- Transformations:
  - Statistics, addition / multiplication by a scalar, global conservation

- Interpolations:
  - Nearest-neighbor
  - Bilinear
  - Conservative

\[
\begin{align*}
\text{x source grid point} & \quad & \text{ϑ target grid point}
\end{align*}
\]
Performance:

Time for one ping-pong exchange on Curie (toy models) for OASIS-MCT and OASIS3.3.
Conclusions:

- First version of OASIS3-MCT released in August 2012
- Very simple to use for traditional OASIS3 users
- OASIS3-MCT solves the problem of the coupling bottleneck observed at high number of cores for previous OASIS versions, with which the coupling fields had to be reassembled onto one (or more) coupler process to perform a mono-process interpolation
- Excellent example of the benefit of shared software

Perspectives:

- Bi-cubic interpolation
- Work on scalability at >1000 cores

Regular training sessions at CERFACS
Open-PALM : an open source codes coupler for massively parallel multi-physics/multi-components applications and dynamic algorithms

http://www.cerfacs.fr/globc/PALM_WEB/
Open-PALM

History:

- 1996: Mercator project: operational oceanography
  - data assimilation with a high resolution model
  - need for a modular and efficient tool to compose assimilation algorithm (component coupling approach)

- Expertise at CERFACS:
  - OASIS coupler for climate modeling
  - parallel computations, data assimilation

- Human power for the project:
  - 2 Mercator engineers
  - 1 CNRS engineer
  - 1 to 2 CERFACS engineers

- 2000: first proto PALM_PROTO for Mercator
- 2002: PALM_Research, SPMD version that simulates MPMD
- 2003: PALM_MP, true MPMD
- 2011: Open-PALM, open-source (LGPL) and collaboration with ONERA
60 groups use OpenPALM, half for data assimilation, half for multi physic and multi component simulations

> 300 people trained to OpenPALM

Various domains of use:
- D.A.: Oceanography, hydrology, atmospheric chemistry, ...
- Multi physic: hydrology, CFD, agriculture...

At CERFACS :
3 teams regularly use OpenPALM: GLOBC, CFD, AE,
More than 15 regular users among the about 120 people,
Most of CERFACS partners use the code or plan to use it: Météo-France, EDF, CNES, SAFRAN, ONERA

Current developers :
CERFACS : 3 permanents & 1 post-doc
ONERA : 1 permanent
Open-PALM

The graphical user interface PrePALM (CERFACS):

- Definition of the coupling scheme
- Monitoring and post-processing
- Runs on standard PC or on clusters, allows to define coupling algorithms and prepare compilations process

PALM library (CERFACS):

- Dynamic launching by the PALM driver which drives the parallel/sequential execution of the components
- Management of data exchanges
- The PALM library is compiled on the different machines where the applications are executed

CWIPPI library (ONERA):

- Manages fully parallel data exchanged based on distributed mesh definition
- Interpolation between non coincident meshes

Open-PALM = PALM (GUI and library) + CWIPPI
Open-PALM

PrePALM: a friendly and simple GUI for:

- Representing the components
- Handling the parallelism (the two levels)
- Managing the resources
- Describing loops and conditions
- Describing the communications (exchanges)

implement and supervise coupled applications

With all the benefits of a GUI as, for instance:

- Coherency checks on the exchanged data
- Pictorial flow chart representation of the algorithm
- Performance analysis, debugging and run-time monitoring
Open-PALM
Open-PALM
Open-PALM

Branch code:
Control structures
Fortran regions

Branch: sequence of units and instructions
⇒ algorithm

Unit: computational component
- Code
- Module of a code
- Data loaders (NetCDF)
- Pre-defined unit

Communications: described by connecting the plugs
Two levels of parallelism to take the best advantage from the intrinsic parallelism of an application (*MPI*+*OpenMP possible*)

1. Task parallelism: by drawing branches we program task parallelism

2. Distributed components (each component is a MPI executable): parallel units
Open-PALM

→ Process management:

- The PrePALM scheme is converted into a graph for the PALM driver
- This driver schedules this graph in order to launch the components depending on the available resources
- MPI-2 spawn as well as MPI-2 client/server are used for the dynamic launching of the units. MPI-1 launching available (choice during installation)

→ Target machines:

- HPC or clusters
- Designed for homogeneous computing but functionalities exist for heterogeneous computing (TCP/IP)
- Open-PALM is ported on many architectures (BULL, NEC, CRAY, IBM, SGI, Linux, Mac OS X, Windows via Cygwin)
Open-PALM

Transformations of a code into a PALM unit:

- Creation of an ID card (Read by PrePALM)
- Replace PROGRAM (FORTRAN) by subroutine, or MAIN (C) by a function name
- For a parallel unit:
  - skip the calls to MPI_INIT and MPI_FINALIZE
  - replace MPI_COMM_WORLD by PL_COMM_EXEC
  - replace the calls to STOP, EXIT by PALM_Abort
- Add the PALM Primitives into the codes:
  - PALM_Put(*args): to send data
  - PALM_Get(*args): to receive data

Pre-requisites:

- Codes must run under Linux/Unix
- Code sources should be available AND have to be written in a compiled language (C, C++, FORTRAN 77 ou 90)
  - OR
- User functions (C or FORTRAN) with pointers on data provided by the code
Algebra toolbox:

Pre-defined units interfacing common mathematical libraries (BLAS, LAPACK, ...) applied on the fly to objects exchanged between units (Interesting for units conversion, ...).

Load the unit from the Algebra Toolbox

Execution and communications are managed as if it were a user unit
Monitoring Tools:

- **Run time monitoring**: PrePALM offers the opportunity of monitoring the computation evolution and the resources usage during the execution of an application.

- **Performances analysis**:
  - CPU and elapsed time statistics for the user defined units
  - Post-mortem replay of the algorithm progress in the graphical interface

- **Debugging tools**:
  - A user defined debugging function, called on the PALM_Put and PALM_Get sides of an active communication to check the coherency of the objects, to track the execution and compute the global performances of the parallel execution
  - A PALM output file per branch with different verbosity levels set by the user for different classes of messages
Open-PALM

Run statistics, file /home/globc/morel/iplib_test/mode_mpi1/palmperr_Block_2_000.log

PrePALM file: /home/globc/morel/iplib_test/mode_mpi1/test.ppl
Out file: /home/globc/morel/iplib_test/mode_mpi1/palmperr_Block_2_000.log

Branches:
B1
187 Sec 480000 49.96%
B2
187 Sec 800000 50.04%
Total Branches:
375 Sec 280000

Block:
Block_1
93 Sec 560000 49.91%
Block_2
93 Sec 900000 50.09%
Total Block:
187 Sec 460000

Units:
code
4 374 Sec 180000 97.74%
mirror_code_onip
1 8 Sec 630000
recep
1 0 Sec 000000
unitC
4 0 Sec 010000
Total Units:
382 Sec 820000

PALM_Put:
dynobj.mirror_code_onip
1 0 Sec 000000
objtest.unitC
1 0 Sec 000000
vector.code
8000 22 Sec 920000 90.02%
vector.mirror_code_onip
2000 2 Sec 540000
Total PALM_Put:
25 Sec 460000
The CWIPI: library for data exchange with interpolation in parallel

- Open source (LGPL), developed at ONERA since 2009
- Management of coupling between $n$ parallel codes
- Direct data exchange of interpolated fields between a source and a target meshes (non coincident and differently partitioned on several processes)
- CWIPI primitives Interfaced for C/C++, Fortran and Python codes

Type of meshes:

- 1D: edges
- 2D: triangles, quadrangles, polygons
- 3D: tetrahedrons, pyramids, prisms, hexahedra, polyhedra

Source and target mesh type can be the same or not

Primitives are called in the codes
Palm is a software dealing with couplings or with complex applications built upon sub-modules while respecting the performances of the codes.

Friendly and simple G.U.I. to describe, implement and supervise applications.

Many interests:

- Facilitate evolution and maintenance of complex applications and of its components,
- Easy integration of new codes replacement, in existing applications (multi-physic coupling, collaborative development …)
- Maximize the use of intrinsic parallelism of applications and algorithms.

Regular training sessions at CERFACS (or in situ)
OASIS3-MCT & Open-PALM: 2 open source code couplers

https://verc.enes.org/oasis/
http://www.cerfacs.fr/globc/PALM_WEB/

Anthony Thevenin (thevenin@cerfacs.fr)