O-PALM:
an open source dynamic parallel coupler

Andrea Piacentini
and the PALM team
Overview

- Genesis
- From requirements to design
- From design to implementation
- From features to applications
- Technical challenges
- Discussions
Why two couplers at CERFACS?

1996: the MERCATOR operational oceanography project

- New operational suite with Data Assimilation
- Model configuration still uncertain
- Different methods to test
- Different kinds of observations
- Research and Operations
Why two couplers at CERFACS?

1996: the MERCATOR operational oceanography project

   - New operational suite with Data Assimilation
   - Model configuration still uncertain
   - Different methods to test
   - Different kinds of observations

Research and Operations

Instead of hard-coding data assimilation routines in the model, or vice-versa, couple model + observations handling + error statistics + algebra in a flexible and computationally effective way
Some data assimilation algorithms are based on an iterative minimization.

This implies the repeated execution of the tasks. The total number of iterations is not known beforehand.

In some configurations some tasks are activated only if some observations are available at run-time.

This implies the conditional execution of some tasks.

⇒ DYNAMIC COUPLING
   Process management
   Buffered communications
   Object versioning
DYNAMIC COUPLING

Process management
- Starts and synchronizes the tasks
- Handles algorithms (DO and WHILE loops, IF and CASE switches)
DYNAMIC COUPLING

Process management

- Starts and synchronizes the tasks
- Handles algorithms (DO and WHILE loops, IF and CASE switches)

Buffered communications

- Non blocking on the production side
- Order of production and reception is not relevant
- Cumulated objects (linear combinations on the fly)
- Permanently stored objects for repeated receptions
DYNAMIC COUPLING

Process management
- Starts and synchronizes the tasks
- Handles algorithms (DO and WHILE loops, IF and CASE switches)

Buffered communications
- Non blocking on the production side
- Order of production and reception is not relevant
- Cumulated objects (linear combinations on the fly)
- Permanently stored objects for repeated receptions

Object versioning
- Last In Only Out paradigm
- Coherence of the components produced by a parallel task with loose synchronization
Model integration and some data assimilation tasks can run simultaneously.

This requires the handling of concurrent tasks parallelism. Like in other couplers it is just the first level of parallelism.

Models and/or assimilation codes are themselves parallel.

This requires the handling of the execution of parallel codes and the management of their data exchanges, including the remapping between codes with different distributions. It is the second level of inner parallelism.
Data assimilation algorithms require linear algebra (operations on very large vectors and matrices, usually sparse, and effective minimisations).

This suggested to include in PALM an algebra toolbox interfacing the most effective linear algebra libraries in the form of pre-defined generic “entities” (units) that can be coupled with other codes.
One of the aims of coupling is the reuse of legacy codes. To make it simple we had to grant (reasonably) minimal intrusiveness.

Three main assumptions:

1) The “end point” communications paradigm: the producer of an object does not know anything about the recipients (if any) and the other way round. The coupler makes the matching.
One of the aims of coupling is the reuse of legacy codes. To make it simple we had to grant (reasonably) **minimal intrusiveness**.

Three main assumptions:

1) The “end point” communications paradigm: the producer of an object does not know anything about the recipients (if any) and the other way round. The coupler makes the matching.

2) A reduced set of APIs complemented by a very detailed Graphic User Interface.
Design

One of the aims of coupling is the reuse of legacy codes. To make it simple we had to grant (reasonably) **minimal intrusiveness**.

Three main assumptions:

1) The "end point" communications paradigm: the producer of an object does not know anything about the recipients (if any) and the other way round. The coupler makes the matching.

2) A **reduced set of APIs** complemented by a very detailed **Graphic User Interface**.

3) **Multi language APIs** for the most common compiled and interpreted languages used for geophysics modelling.
The same tool for

**Operational Use**

- Performances
- Run-time monitoring
The same tool for

**Operational Use**
- Performances
- Run-time monitoring
- Post mortem performances analyses
The same tool for

**Operational Use**
- Performances
- Run-time monitoring
- Post mortem performances analyses

**Research Use**
- User friendliness
- Portability (⇒ standard solutions)
But, at the time

1) no need of grid to grid interpolation

2) [almost a consequence] focus on the algorithm and not on the treatments (e.g. algebra units on the canvas and not as an attribute of the communication)
But, at the time
1) no need of grid to grid interpolation
2) [almost a consequence] focus on the algorithm and not on the treatments (e.g. algebra units on the canvas and not as an attribute of the communication)

PALM is not OASIS because OASIS did not suit the dynamic requirements.

OASIS4 is not PALM because the process management implementation constraints + the lack of grid to grid interpolations made the PRISM community invest on OASIS.
We focus on the [successive] implementation choices for the PALM driver and libraries.

The GUI is coded in Tcl/Tk and would deserve a separate speech.
In 1997 lack of complete and robust MPI2 implementations:
⇒ MPI1 emulation based on a pool of idle processes.
\texttt{PALM\_SP} (a.k.a. \texttt{PALM\_RESEARCH}).
Implementation

START

FINALIZE

TERMINATION

LAUNCH

FINALIZE

Branch 1

Branch 2

Idle processes

Unit1 proc 0

Unit1 proc 1

Unit1 proc 2

Idle

Branch 1

Pool of idle processes

Branch 1

Pool of idle processes

Branch 1

Pool of idle processes

Branch 1

Pool of idle processes
In 1997 lack of complete and robust MPI2 implementations: 
⇒ MPI1 emulation based on a pool of idle processes. 
**PALM_SP** (a.k.a. PALM_RESEARCH).

Intended for functional tests, it is still used for some applications (e.g. N.R.T. air-quality forecasts in MACC)

Some interesting features that could be recovered now.
In 2003 release of the first version of a full MPMD (Multiple Programs Multiple Data) MPI2 based coupler:

**PALM MP**

Dynamic process management via `MPI_Comm_Spawn` + a scheduler.

Option to merge into a single executable (a block) the coupled components that are started in a sequence.

End-point high bandwidth communication scheme, with the driver acting as a broker (useful for dynamic coupling and for monitoring).
Since then some achieved and some under development enhancements:

The possibility to interface commercial black-box codes (such as Fluent, Abaqus MSC/MARC) by the use of external dynamic libraries and/or a socket based layer
Since then some achieved and some under development enhancements:

The possibility to interface commercial black-box codes (such as Fluent, Abaqus MSC/MARC) by the use of external dynamic libraries and/or a socket based layer

The implementation of a simplified working mode entirely compliant with the MPI-1 library (MPMD mpirun extension)
Since then some achieved and some under development enhancements:

The possibility to interface commercial black-box codes (such as Fluent, Abaqus MSC/MARC) by the use of external dynamic libraries and/or a socket based layer

The implementation of a simplified working mode entirely compliant with the MPI-1 library (MPMD mpirun extension)

The optimisation of repeated well synchronised communications that don't require the intervention of the driver
Since then some achieved and some under development enhancements:

The possibility to interface commercial black-box codes (such as Fluent, Abaqus MSC/MARC) by the use of external dynamic libraries and/or a socket based layer

The implementation of a simplified working mode entirely compliant with the MPI-1 library (MPMD mpirun extension)

The optimisation of repeated well synchronised communications that don't require the intervention of the driver

The enhancement of the parallel algebra toolbox that is soon going to include the CWIPI interpolation library from ONERA for the grid to grid remapping
Since then some achieved and some under development enhancements:

Starting from January 2011 PALM will become open source with the name O-PALM.

It is the most suitable environment to accept collaborations and contributions on the coupler development.
Some application canvases to stress the importance of the aforementioned features
Dynamic coupling for a combustor cooling system optimization
Dynamic coupling for a combustor cooling system optimization
Flexible communication scheme (remapping, sub-objects, time shifts) for atmosphere biosphere coupling
Flexible communication scheme (remapping, sub-objects, time shifts) for atmosphere biosphere coupling
Flexible communication scheme (remapping, sub-objects, time shifts) for atmosphere biosphere coupling

![Diagram showing flexible communication scheme for atmosphere biosphere coupling.](image)
Complex algorithm
High performances
... as required!
Still using PALM_SP
And V6 is much worse!
Valentina

Still using PALM_SP

And V6 is much worse!

Units from the algebra toolbox
Hydrology

Code reuse for parallel perturbed runs

Interaction with the system for file handling
Difficult trade-off between a centralised and a fully distributed approach.

Process management is a key issue for dynamic coupling, but it implies some extra constraints. We've already implemented the MPMD MPI1 extension (mpirun with >1 executables). How to deal with automatic load balancing? Separate phases of reorganization (higher overhead) and simulation (lower overhead) and/or resurrect the link of >1 codes in a single executable (cf. PALM_SP).
Technical challenges

Difficult trade-off between a centralised and a fully distributed approach.

Process management is a key issue for dynamic coupling, but it implies some extra constraints. We've already implemented the MPMD MPI1 extension (mpirun with >1 executables). How to deal with automatic load balancing? Separate phases of reorganization (higher overhead) and simulation (lower overhead) and/or resurrect the link of >1 codes in a single executable (cf. PALM_SP).

Very effective communications on MPP configurations. Trade-off between flexibility and monitoring on one side and performances on the other.
Technical challenges

Difficult trade-off between a centralised and a fully distributed approach.

Process management is a key issue for dynamic coupling, but it implies some extra constraints. We've already implemented the MPMD MPI1 extension (mpirun with >1 executables). How to deal with automatic load balancing? Separate phases of reorganization (higher overhead) and simulation (lower overhead) and/or resurrect the link of >1 codes in a single executable (cf. PALM_SP).

Very effective communications on MPP configurations. Trade-off between flexibility and monitoring on one side and performances on the other.

Role of the coupler code itself: make it parallel or let most of the tasks to the units and/or the system?
Technical challenges

Integration of the communication and of the interpolation layers: CWIPI developed by ONERA. Exchange of fields defined on any kind of non-structured mesh. Surface and volume non conservative interpolations (clouds) + callback of user defined interpolations.

Based on robust industrial MPI wrappers (by EDF).

Very good scaling tested up to 256 processors. Still under investigation for MPP figures.

Basic bricks to implement conservative interpolations, but still to do and test.

Projects of assessment of the CWIPI (and its evolutions) layer for climate couplings.

We'll keep you informed