



On the use of a Computational Framework for Data Assimilation Applications

Samuel Buis, Damien Déclat, Etienne Gondet, Sébastien Massart, Thierry Morel
Global Change and Climate modelling team, CERFACS, Toulouse, France
E-mail: palm@cerfacs.fr Web site: www.cerfacs.fr/~palm

Genesis

The PALM software was originally designed to handle the MERCATOR project's operational data assimilation system in a flexible and performant way.

It came out very quickly that the PALM concepts can also be applied to more general applications.

Today, the PALM software is freely available for research purposes and is used in several projects, dealing with data assimilation in oceanography, atmospheric chemistry, but also CFD coupled simulations, etc...

Concept

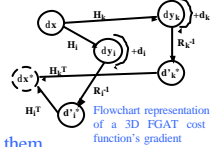
The aim of PALM is to handle complex applications in a modular and parallel way to ensure their evolutivity and guarantee their performance.

Applications are created in PrePALM, the PALM GUI, by assembling independent components (PALM Unit) in sequences (PALM Branch).

PALM is in charge of executing the sequences and making the components exchange some data.

Data Assimilation

Data Assimilation methods can be described in terms of sequences of operators and data flows. (1)



Most of the methods uses the same operators but in a different sequence or with different data flows between them.

Thus, given a set of operators, PALM allows to implement easily several methods and facilitates their evolution and their integration in existing systems.

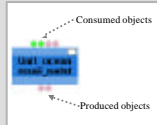
(1) Th. Lagarde, A. Piacentini and O. Thuau: "A new representation of data assimilation methods: the PALM flow charting approach." Q.J.R.M.S. (2001), Vol. 127, pp. 189-207

Building an application with PALM

Development Phase

A PALM Unit is composed of :

- A piece of code (fortran subroutines, C or C++ functions) that can consume and produce data PALM Object) by calling PALM primitives.
- An Identity Card (text file containing information about the unit: name and type of the produced and consumed objects, ...).



The PALM primitives:

```
CALL PALM_Put(space_name, obj_name, time, tag, local_var, err_code)
CALL PALM_Get(space_name, obj_name, time, tag, local_var, err_code)

space_name: string identifying the object space.
A space is a class of objects that have the same machine representation (element size and shape).
obj_name: string identifying the object name.
time: integer indicating the time associated to this object instance.
tag: integer indicating the tag associated to this object instance.
local_var: local variable of type XXXX which will contains the field associated to the object.
err_code: output integer error code.
```

Identity Card:

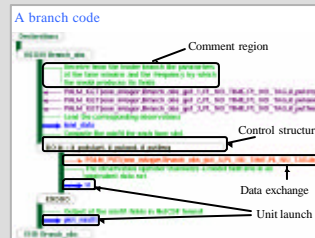
```
PALM_UNIT
+
+ <sub>subroutine_name</sub>
+ <comment (optional comment)>
+
+ PALM_SPACE
+
+ <name>space_name</name>
+ <shape (---)>
+ <element_size space_type (integer, real, ...)>
+ <comment (optional comment)>
+
+ PALM_OBJECT
+
+ <name>object_name</name>
+ <source_object_location (IN, OUT, INOUT)>
+ <space space_name</space space_name>
+ <comment (optional comment)>
```

Integration Phase in the PALM GUI

1. Definition of the algorithm

The application is created by assembling units in one or several sequences (PALM Branch).

Branch codes are graphically defined by the user in PrePALM the PALM GUI.



2 levels of parallelism in an application:

- Several branches can run in parallel.
- Within a branch, Units can be parallel.

2. Definition of the communication scheme



The user must define the data exchanges between units by plugging output objects with input ones.

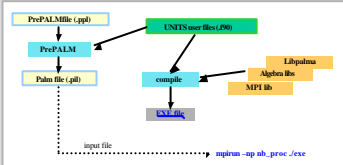
PALM ensures the redistribution of objects differently distributed on the two sides of the communication.

PALM provides predefined units to compute algebraic operations between objects.

Running an application with PALM

PALM_Research is the SPMD (Simple Program Multiple Data) version of the PALM software.

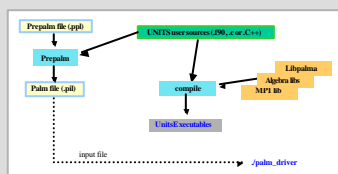
An executable is created by compiling and linking the units file with the PALM_Research library.



PALM_MP is the MPMD (Multiple Programs Multiple Data) version of the PALM software.

The PALM driver's executable is launched on one process, and is able to spawn the units or block of units' executables.

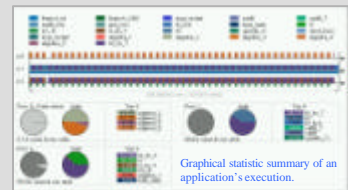
PALM_MP is language interoperable (units can be developed in fortran, C or C++) and contains more functionalities than PALM_Research (sub-object, dynamic change of the space and distribution definition).



Supervision tools

PALM provides some supervision tools to debug control and optimise applications:

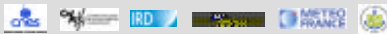
- > Debug functions in PALM_PUT and PALM_GET to verify objects values.
- > Output files containing information about the run with several levels of verbosity per message category.
- > Performance analyser with graphical statistic summaries.
- > Runtime monitoring of the application.



Some Data Assimilation Applications using PALM

MERCATOR

MERCATOR is an operational oceanography project supported by:



Aim: To implement an operational oceanic forecast system able to simulate the global ocean with a high resolution model by assimilating in-situ and satellite data.

Strategy: Evolutive implementation of the system based on the PALM software, from regional to global models, from simple (OI) to complex (SEEK, 4DVAR) assimilation methods.



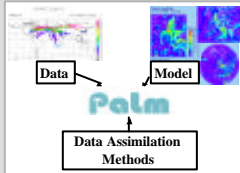
<http://www.mercator.com.fr>

ASSET

A European project to exploit and develop earth observation data from Envisat satellite using data assimilation (01 / 2003 -> 12 / 2005).

Partners: METEO FRANCE, CNRS, etc.

PALM will be used to develop a data assimilation system on the Meteo-France atmospheric chemistry model MOCAGE.



<http://darc.nerc.ac.uk/asset>

A Variational Suite

A complete suite of data assimilation methods (3DFGAT, 3DPSAS, 4DINC, 4DVAR, 4DPSAS) implemented with PALM on a Shallow Water model.

Described in a tutorial presented during the PALM training courses regularly organized at CERFACS.

This modular and easy evolutive suite can be a model for developing your own data assimilation application with PALM

