COUPLED PROBLEMS 2011

Coupled Solution Strategies II

Kos (Greece), June 20th-22nd 2011

CERFACS-ONERA Open-PALM: an open source dynamic parallel coupler

Andrea Piacentini

and the PALM team: Th. Morel, F. Duchaine, A. Thévenin

CERFACS – Toulouse (France)

www.cerfacs.fr/globc/PALM_WEB

Overview

- Rationale and Genesis
- Design
- Implementation
- Technical challenges
- Some applications
- Discussions

Rationale

Dynamic coupling = a coupling where the components execution scheduling and the data exchange patterns cannot be entirely defined before execution

Rationale

Dynamic coupling = a coupling where the component execution scheduling and the data exchange patterns cannot be entirely defined before execution

The PALM rationale:

Historical

- Implementation of data assimilation suites: dynamic aspects of data assimilation algorithms themselves

Current

- Multi-physics coupling in flexible configurations

Future

- MPP, high performances, self-tuning, resilient applications

Genesis

Data assimilation is a technique aiming to improve numerical models skills by the use of observational data.

It is based on computationally expensive algebraic algorithms involving the model, the observation treatments, the statistical characterisation of the errors on both sides.

In 1996, the MERCATOR operational oceanography project faced the problem to set-up a new operational suite with Data Assimilation for Research and Operations in an evolving configuration.

Instead of hard-coding data assimilation routines in the model, or vice-versa, they decided to **couple**
model + observations handling + error statistics + algebra
in a flexible and computationally effective way

Design

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.

$\Rightarrow$ 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

Design

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.

Design

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.

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.

Design

One of the aims of coupling is the reuse of legacy codes.

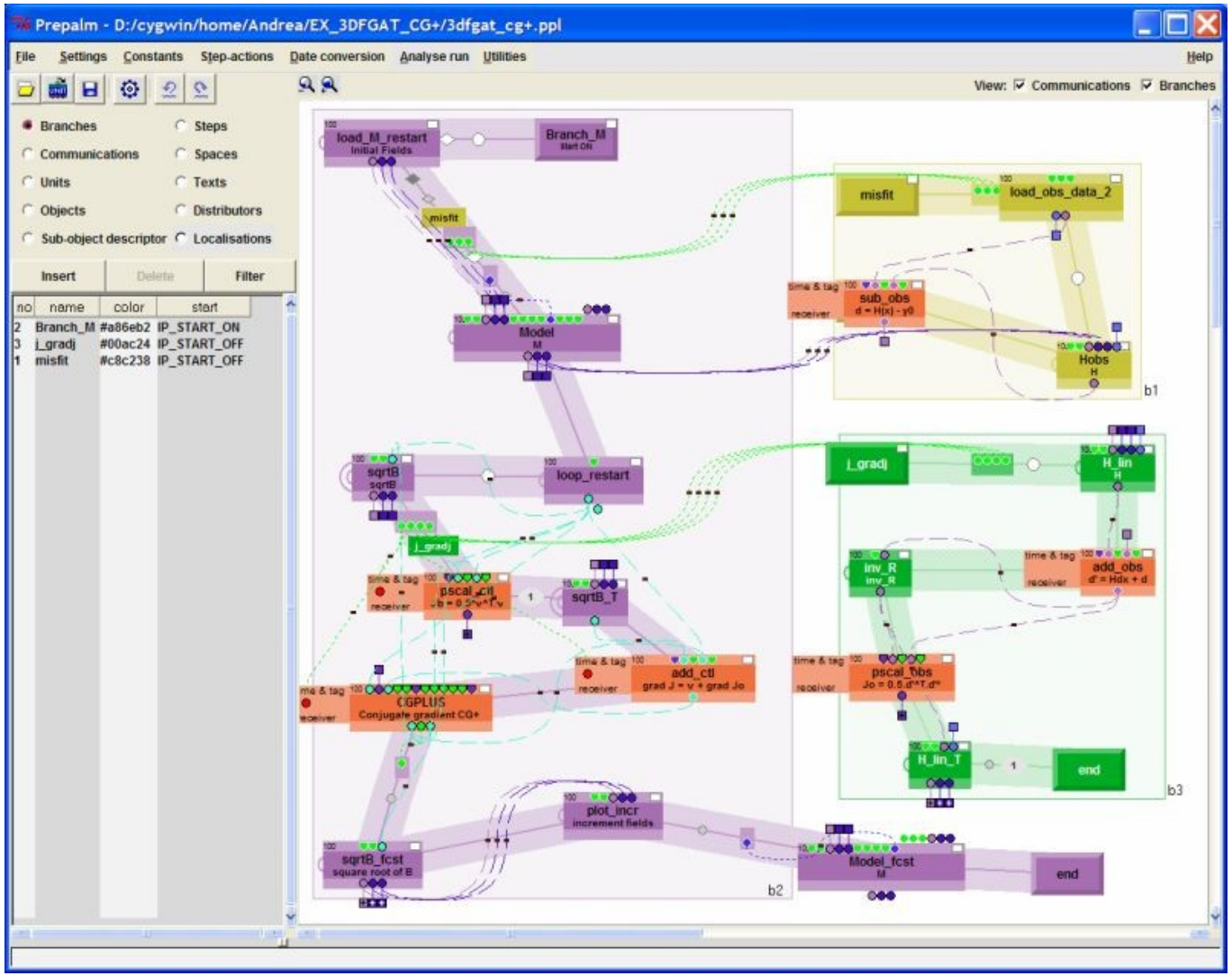
To make it simple we had to grant (reasonably) minimal intrusiveness.

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.

Design


Prepalm - D:/cygwin/home/Andrea/EX_3DFGAT_CG+/3dfgat_cg+.ppl
File
Settings
Constants
Step-actions
Date conversion
Analyse run
Utilities
Help
View: Communications Branches
Branches
Steps
Communications
Spaces
Units
Texts
Objects
Distributors
Sub-object descriptor
Localisations
Insert
Delete
Filter
no name color start
2 Branch_M #a86eb2 IP_START_ON
3 j_gradj #00ac24 IP_START_OFF
1 misfit #c8c238 IP_START_OFF
load_M_restart
Initial Fields
Branch_M
misfit
load_obs_data_2
sub_obs
Model
Hobs
b1
sqrtB
loop_restart
j_gradj
H_lin
inv_R
add_obs
pscal_ctl
sqrtB_T
add_ctl
grad J = v + grad Jo
pscal_obs
CGPLUS
Conjugate gradient CG+
H_lin_T
end
b3
plot_incr
increment fields
sqrtB_fcst
square root of B
Model_fcst
end
b2

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

- F77, F90, C, C++ bindings
- SWIG interfaces for Python, Perl, Java, Tcl/Tk, Octave (Matlab), ...
- API's for precompiled codes via dynamic libraries

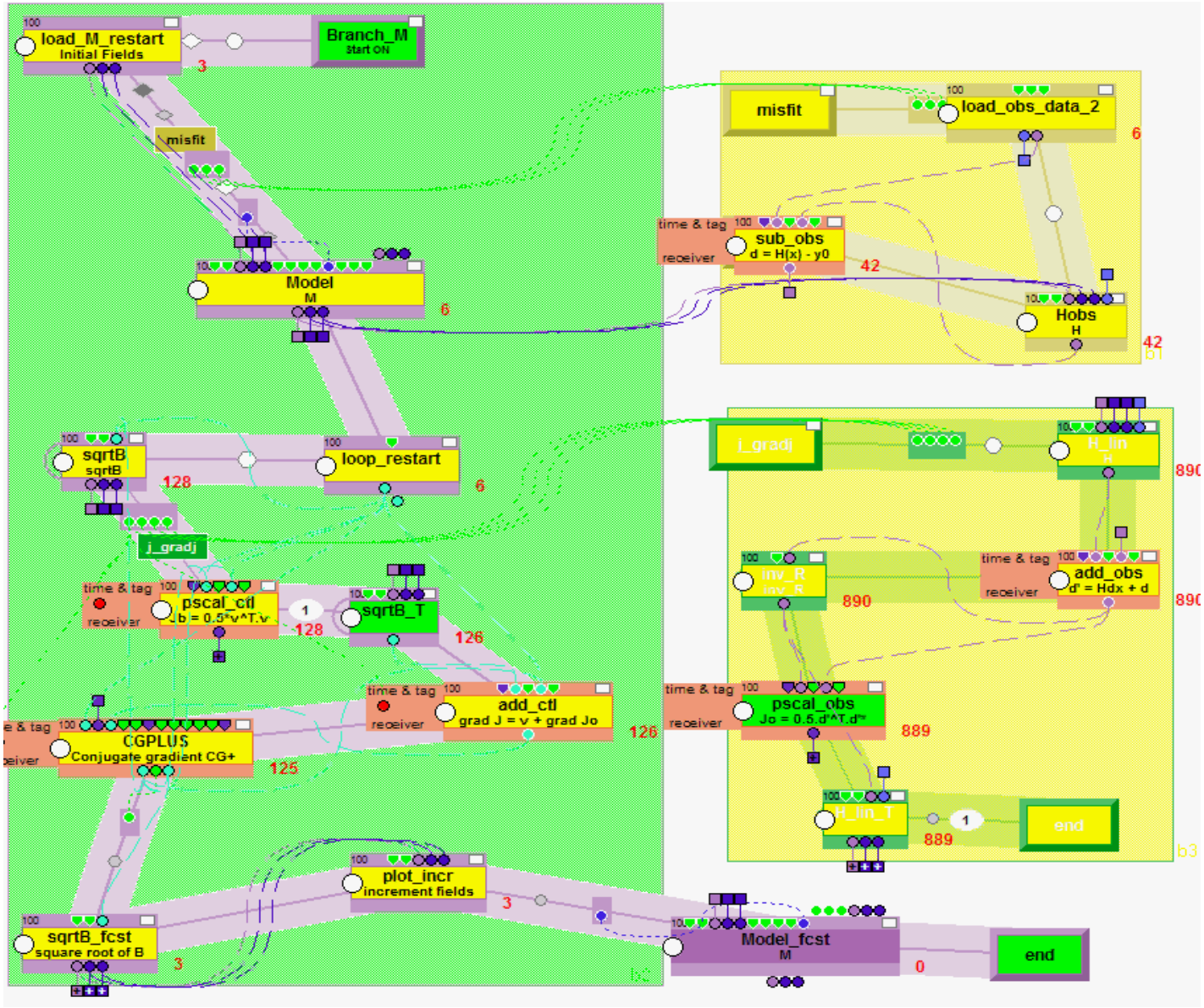
Design

The same tool for

Operational Use

- Performances
- Run-time monitoring

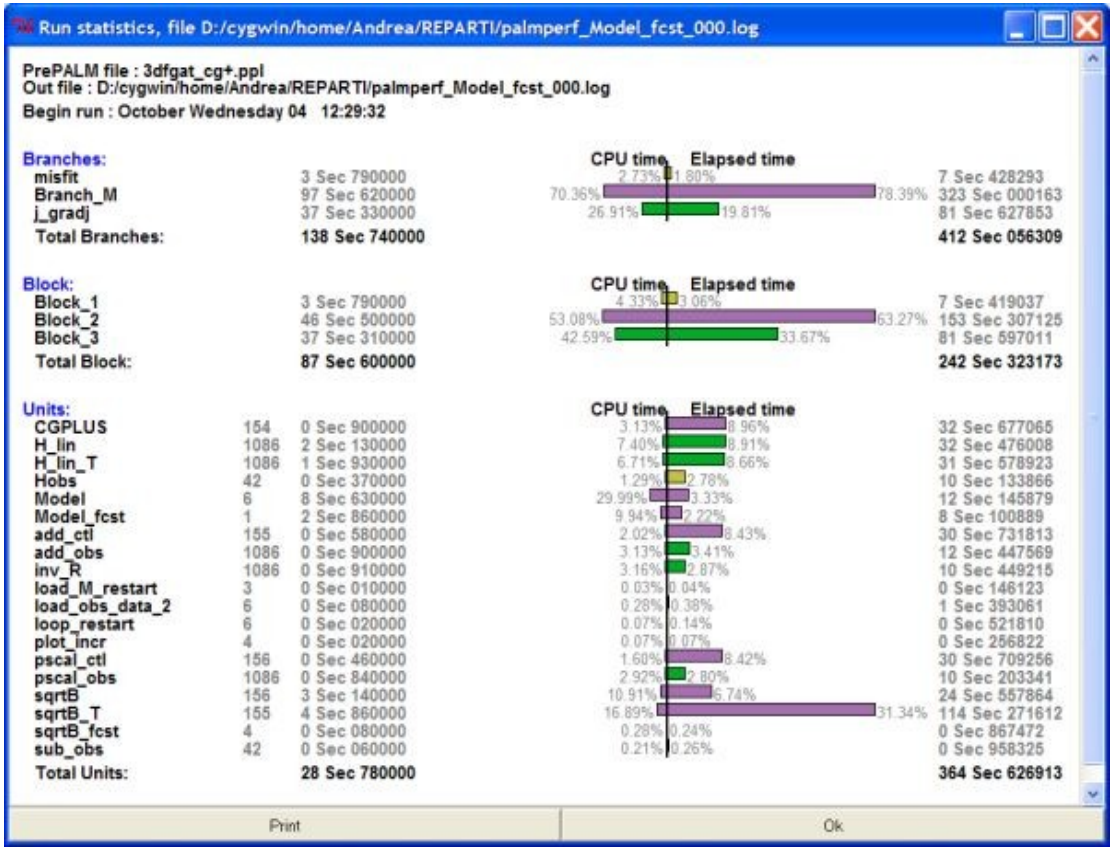
Design


load_M_restart
Initial Fields
Branch_M
start ON
misfit
Model
M
sqrtB
sqrtB
loop_restart
j_gradj
pscal_ctl
b = 0.5*v^T.v
sqrtB_T
add_ctl
grad J = v + grad Jo
CGPLUS
Conjugate gradient CG+
plot_incr
increment fields
sqrtB_fcst
square root of B
Model_fcst
M
end
misfit
load_obs_data_2
sub_obs
d = H(x) - y0
Hobs
H
j_gradj
H_lin
H
inv_R
inv_R
add_obs
d' = Hdx + d
pscal_obs
Jo = 0.5.d'^T.d''
H_lin_T
end

The same tool for

Operational Use

- Performances
- Run-time monitoring
- Post mortem performances analyses


Run statistics, file D:/cygwin/home/Andrea/REPARTI/palmperf_Model_fcst_000.log
PrePALM file : 3dfgat_cg+.ppl
Out file : D:/cygwin/home/Andrea/REPARTI/palmperf_Model_fcst_000.log
Begin run : October Wednesday 04 12:29:32
Branches:
misfit 3 Sec 790000
Branch_M 97 Sec 620000
j_gradj 37 Sec 330000
Total Branches: 138 Sec 740000
CPU time
Elapsed time
2.73% 1.80% 7 Sec 428293
70.36% 78.39% 323 Sec 000163
26.91% 19.81% 81 Sec 627853
412 Sec 056309
Block:
Block_1 3 Sec 790000
Block_2 46 Sec 500000
Block_3 37 Sec 310000
Total Block: 87 Sec 600000
CPU time
Elapsed time
4.33% 3.06% 7 Sec 419037
53.08% 63.27% 153 Sec 307125
42.59% 33.67% 81 Sec 597011
242 Sec 323173
Units:
CPGLUS 154 0 Sec 900000
H_lin 1086 2 Sec 130000
H_lin_T 1086 1 Sec 930000
Hobs 42 0 Sec 370000
Model 6 8 Sec 630000
Model_fcst 1 2 Sec 860000
add_ctl 155 0 Sec 580000
add_obs 1086 0 Sec 900000
inv_R 1086 0 Sec 910000
load_M_restart 3 0 Sec 010000
load_obs_data_2 6 0 Sec 080000
loop_restart 6 0 Sec 020000
plot_incr 4 0 Sec 020000
pscal_ctl 156 0 Sec 460000
pscal_obs 1086 0 Sec 840000
sqrtB 156 3 Sec 140000
sqrtB_T 155 4 Sec 860000
sqrtB_fcst 4 0 Sec 080000
sub_obs 42 0 Sec 060000
Total Units: 28 Sec 780000
CPU time
Elapsed time
3.13% 8.96% 32 Sec 677065
7.40% 8.91% 32 Sec 476008
6.71% 8.66% 31 Sec 578923
1.29% 2.78% 10 Sec 133866
29.99% 3.33% 12 Sec 145879
9.94% 2.22% 8 Sec 100889
2.02% 8.43% 30 Sec 731813
3.13% 3.41% 12 Sec 447569
3.16% 2.87% 10 Sec 449215
0.03% 0.04% 0 Sec 146123
0.28% 0.38% 1 Sec 393061
0.07% 0.14% 0 Sec 521810
0.07% 0.07% 0 Sec 256822
1.60% 8.42% 30 Sec 709256
2.92% 2.80% 10 Sec 203341
10.91% 5.74% 24 Sec 557864
16.89% 31.34% 114 Sec 271612
0.28% 0.24% 0 Sec 867472
0.21% 0.26% 0 Sec 958325
364 Sec 626913
Print
Ok

Design

The same tool for

Operational Use

- Performances
- Run-time monitoring
- Post mortem performances analyses

Research Use

- User friendliness
- Portability ($\Rightarrow$standard solutions)

Design

The same tool for

Operational Use

- Performances **MPI based**
- Run-time monitoring
- Post mortem performances analyses

Research Use

- User friendliness
- Portability ($\Rightarrow$standard solutions) **MPI based**

Design

PALM has been adopted for different applications dealing with flexible coupling configurations.

A typical example are the shape optimization applications in computational fluid dynamics, coupling heavy [proprietor] simulation codes with algebraic iterative optimization algorithms.

For data assimilation there is no need of generic grid-to-grid interpolation tools (at most it is user provided code).

Multi-physics applications (including climate modelling), using independently developed components, have arouse the need of an efficient grid-to-grid parallel interpolation tool in PALM.

Last generation models with automatic load balancing, data redistribution and self-tuning capabilities are defining the next challenges for the use of couplers on MPP machines.

Implementation

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.

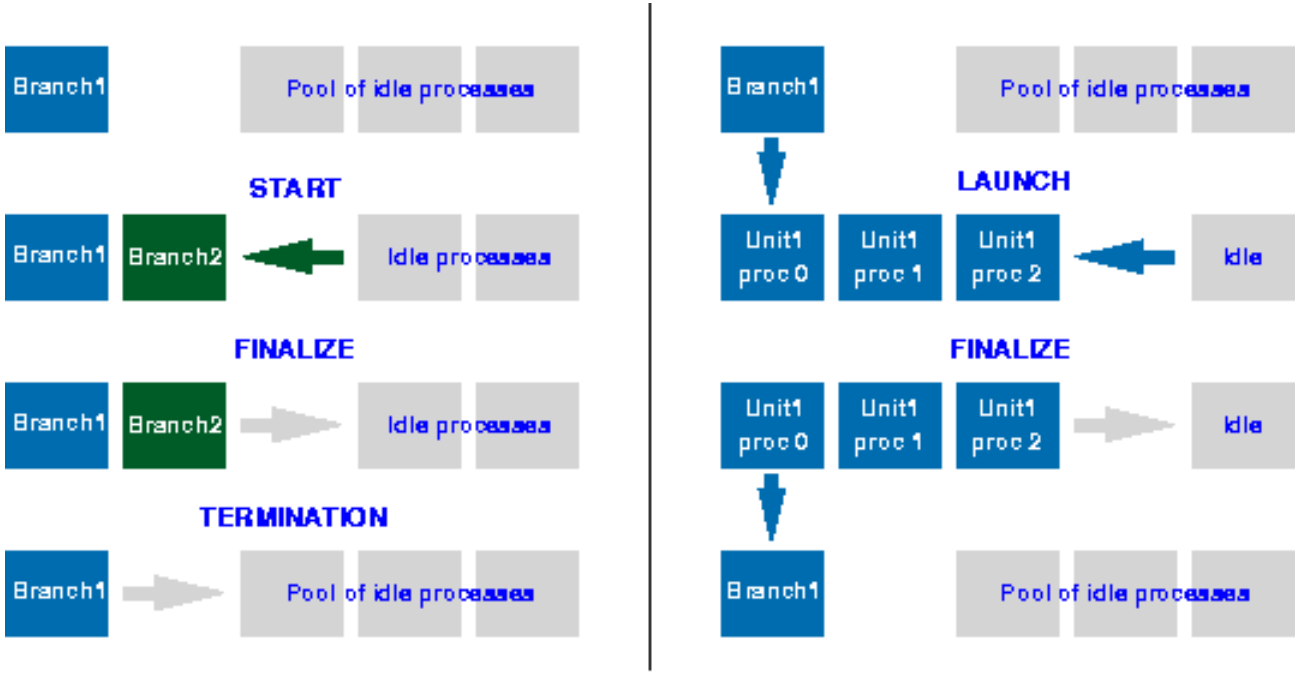
Implementation

In 1997 lack of complete and robust MPI2 implementations:

$\Rightarrow$ MPI1 emulation based on a pool of idle processes.

PALM_SP (a.k.a. PALM_RESEARCH).

Implementation



Implementation

In 1997 lack of complete and robust MPI2 implementations:

$\Rightarrow$ 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 the EU funded MACC project)

Some interesting features that could be recovered in the next Open-PALM versions.

Implementation

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).

Optimisations

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

Optimisations

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)

Optimisations

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

Optimisations

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 now includes the CWIPI interpolation library from ONERA for the grid to grid remapping

Optimisations

Since then some achieved and some under development enhancements:

Starting from January 2011 PALM has become open source with the name Open-PALM.

It is the most suitable environment to accept collaborations and contributions on the coupler development.

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).

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.

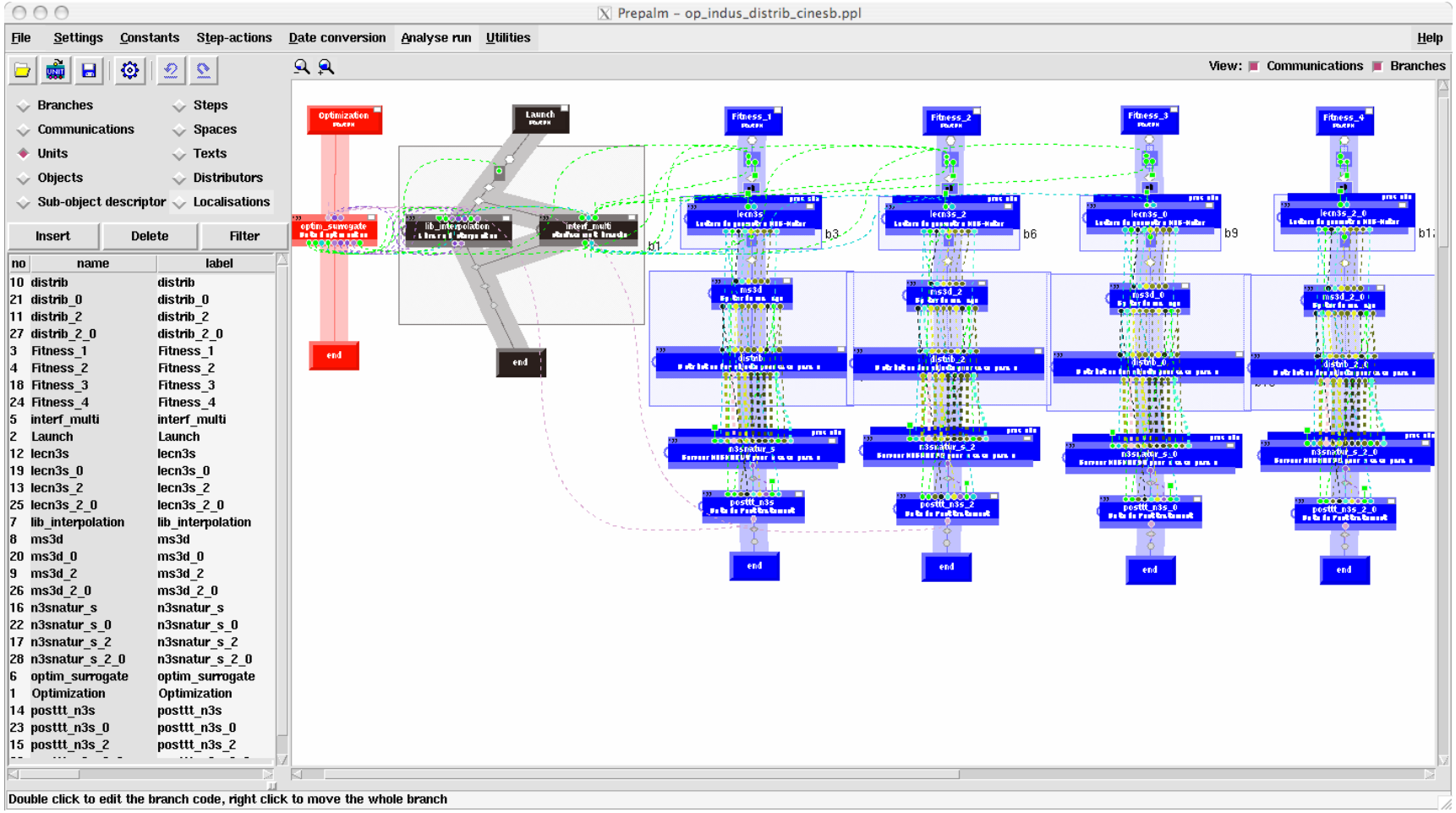
Basic bricks to implement conservative interpolations, but still to do and test.

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

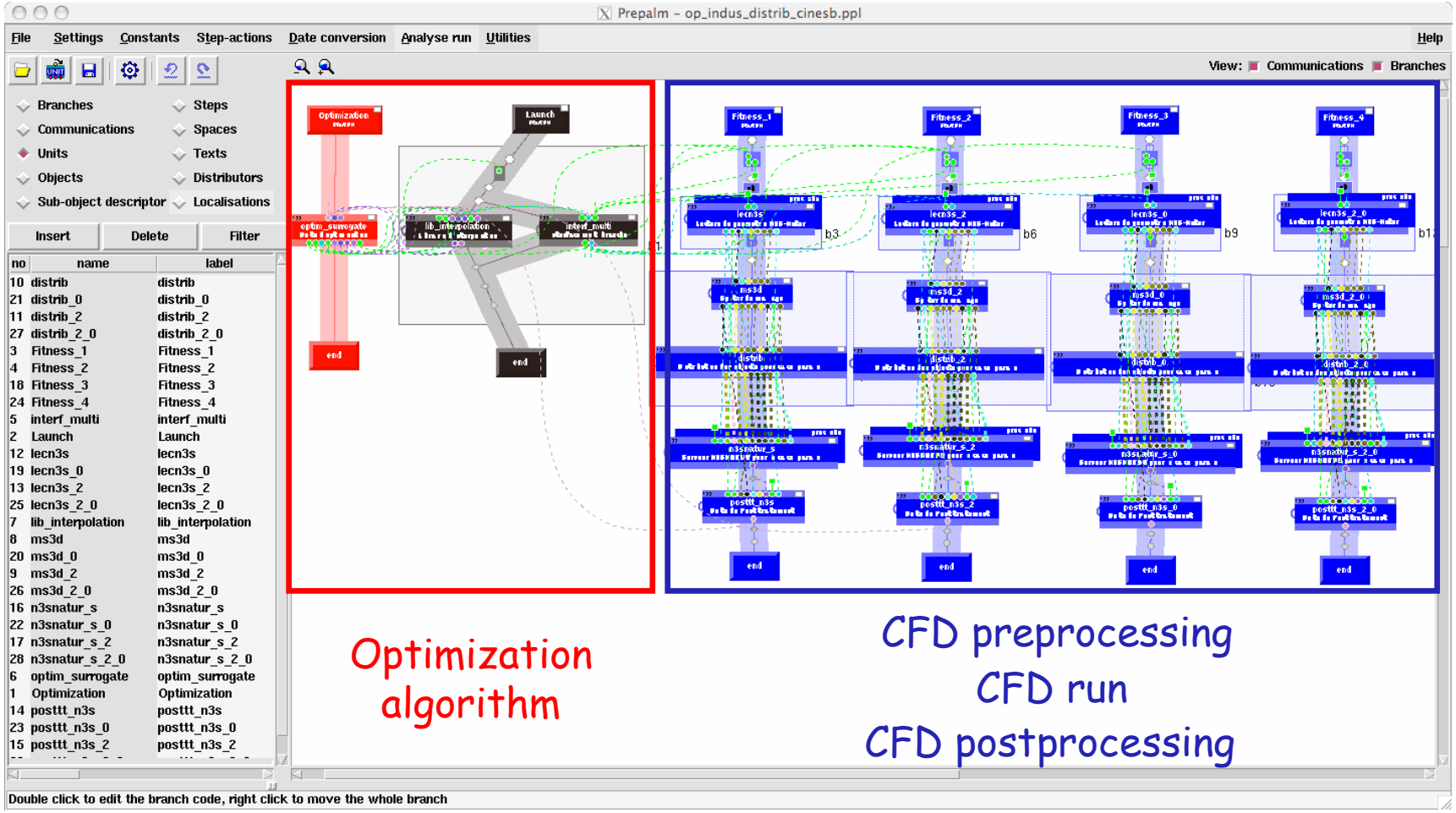
Applications

Some application canvases to stress the importance of the aforementioned features

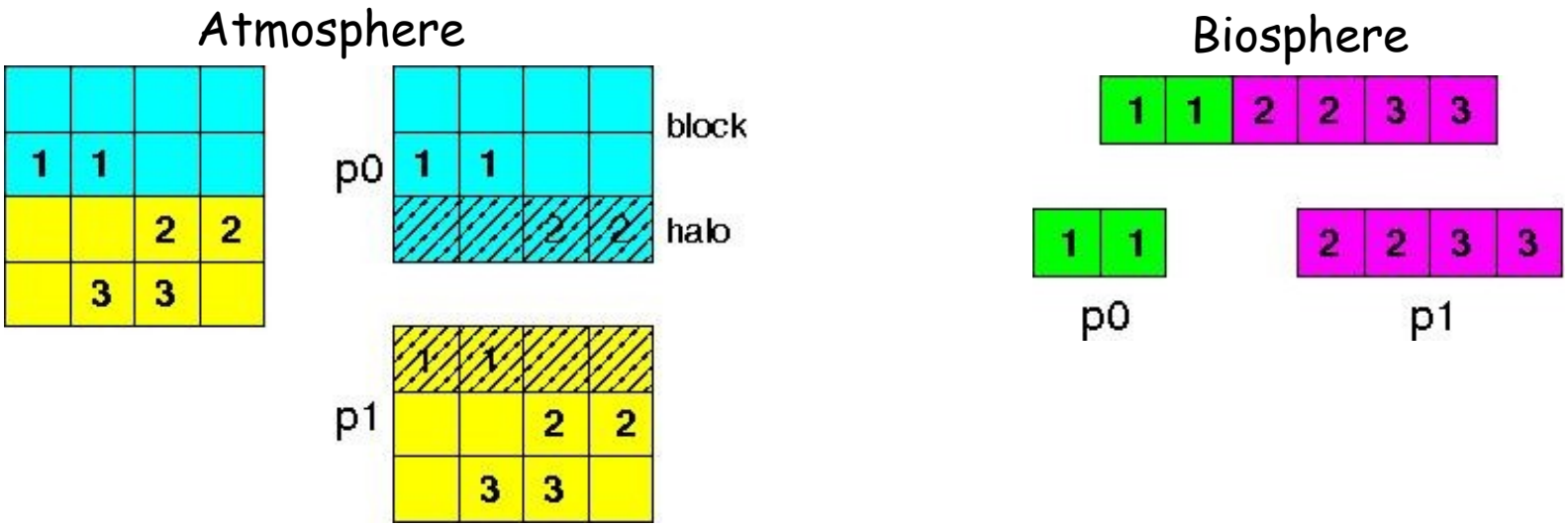
Dynamic coupling for a combustor cooling system optimization



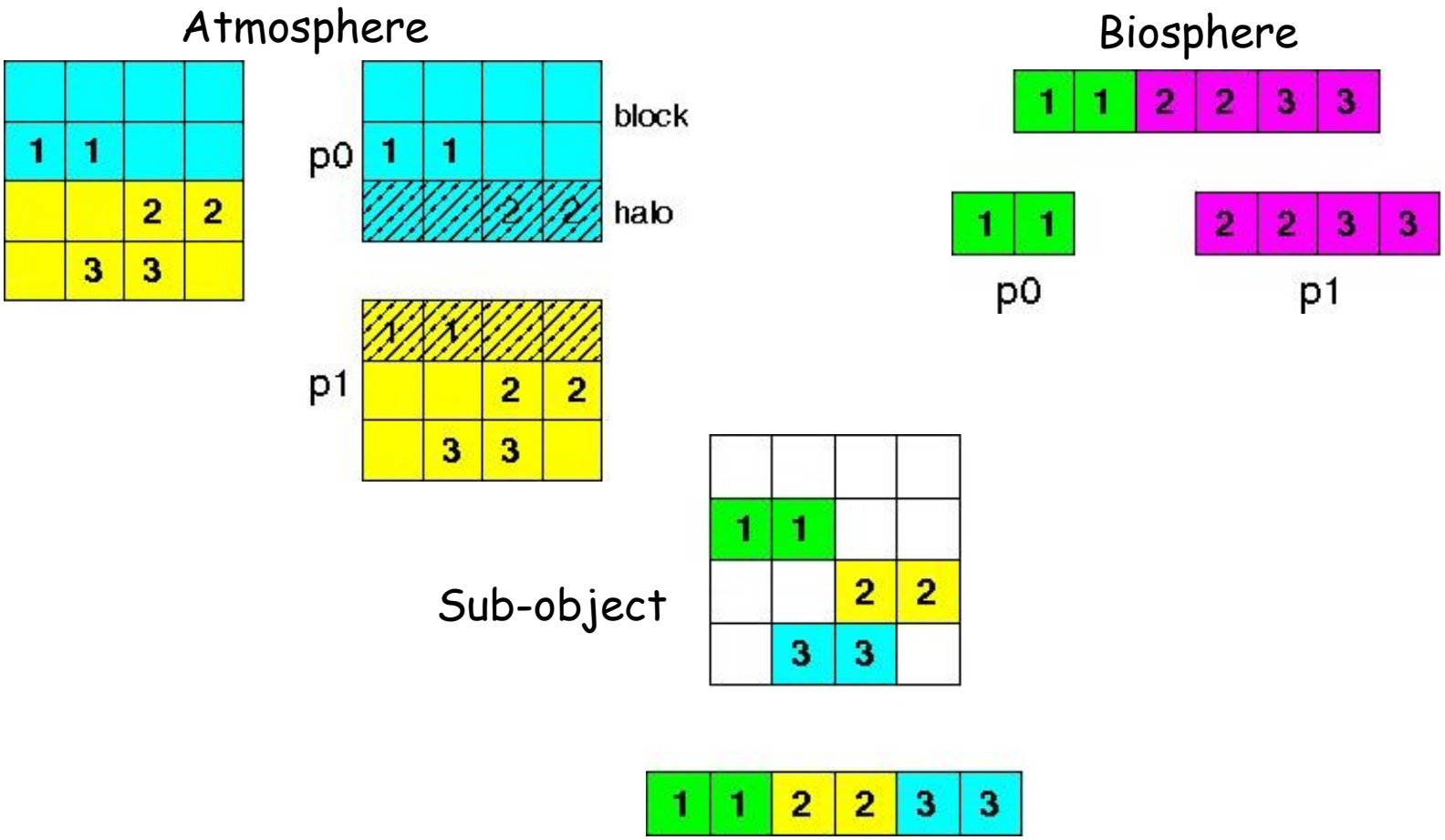
Dynamic coupling for a combustor cooling system optimization



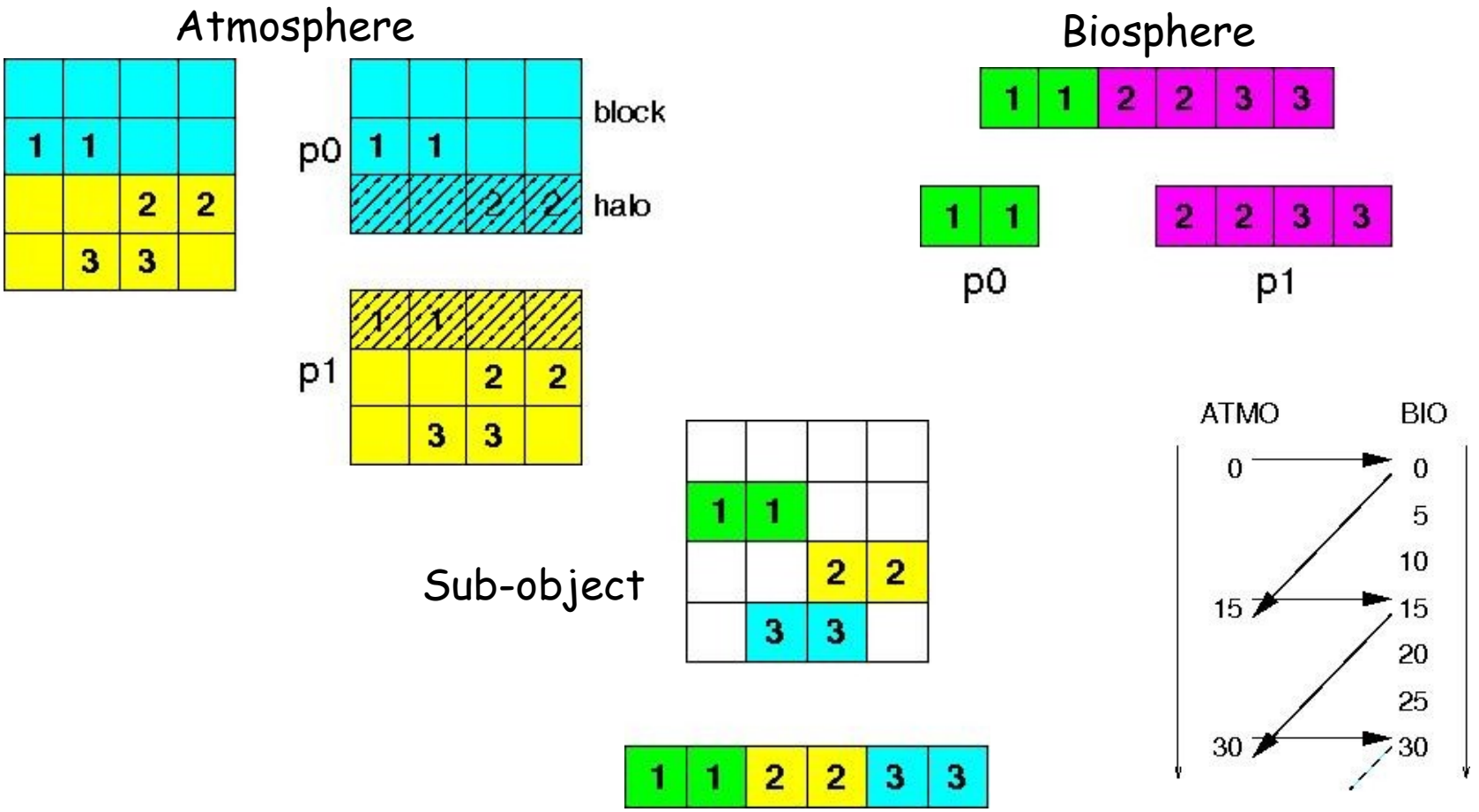
Flexible communication scheme (remapping, sub-objects, time shifts) for atmosphere biosphere coupling



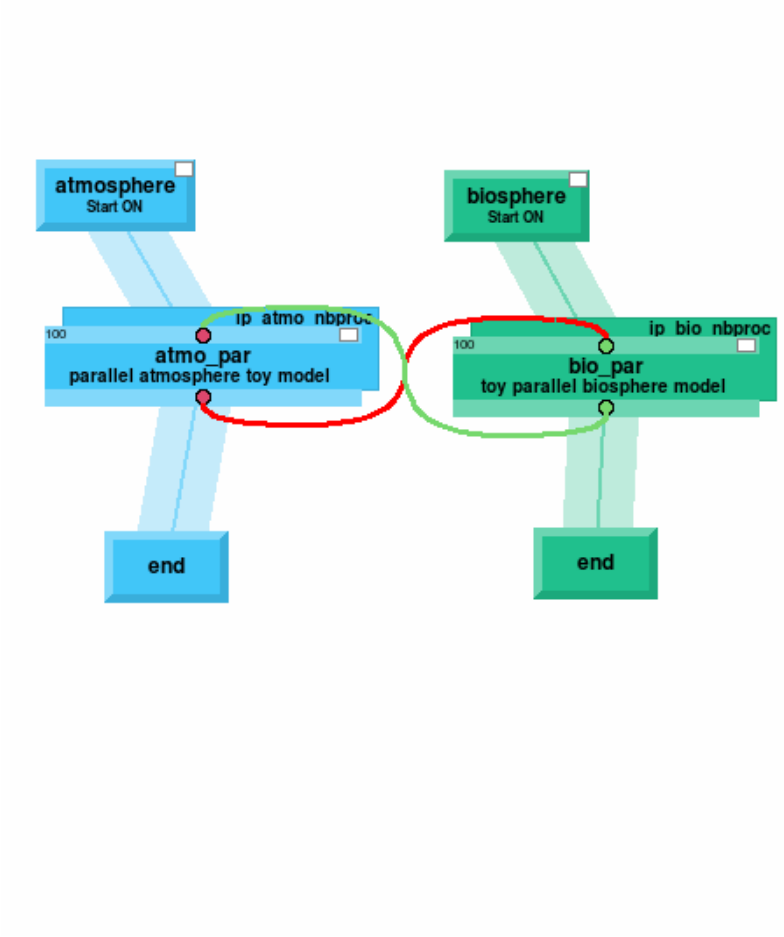
Flexible communication scheme (remapping, sub-objects, time shifts) for atmosphere biosphere coupling

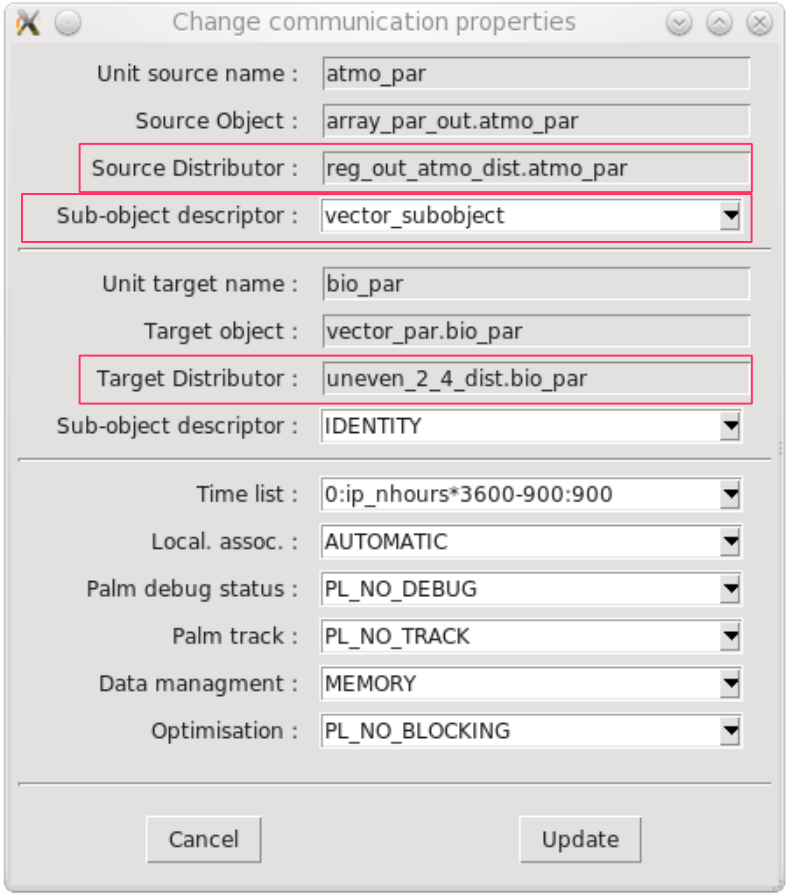


Flexible communication scheme (remapping, sub-objects, time shifts) for atmosphere biosphere coupling

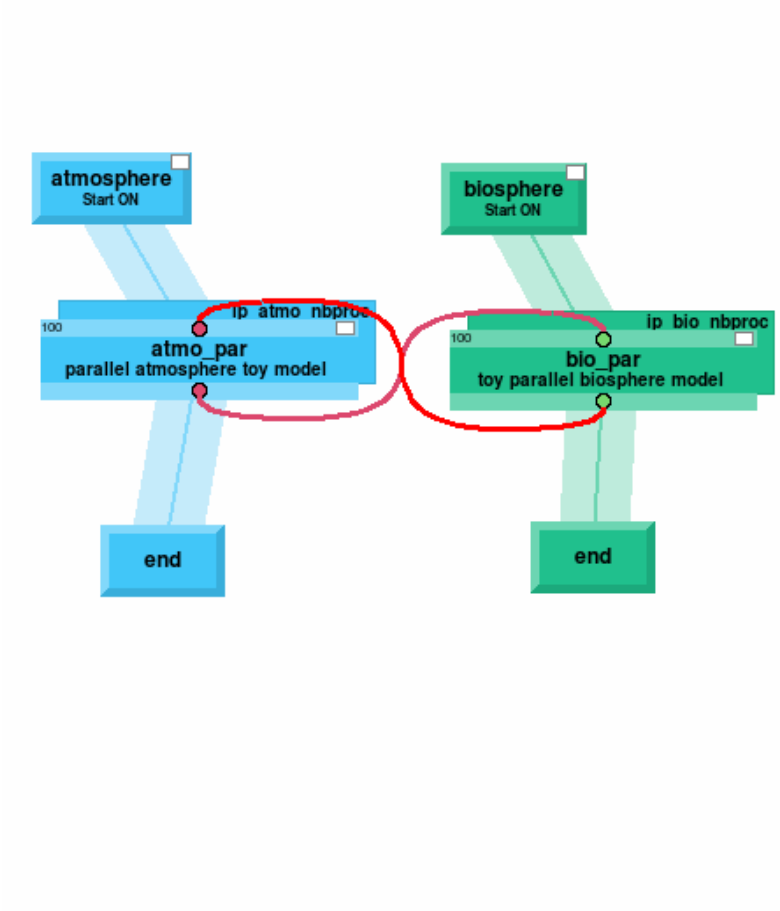


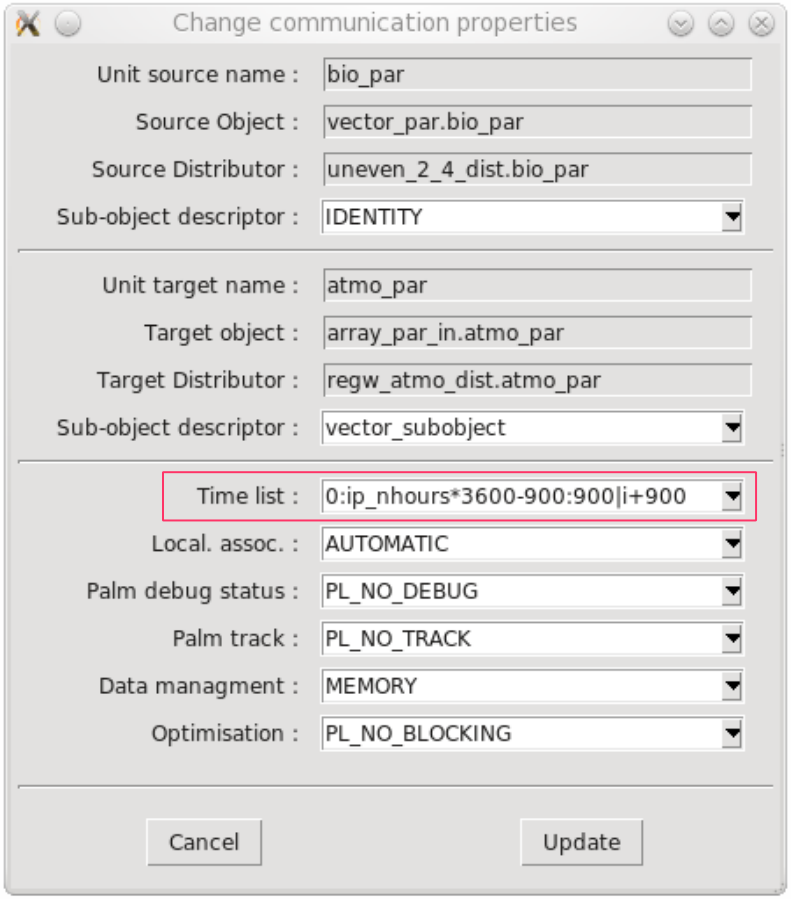
IMK - MOBILE

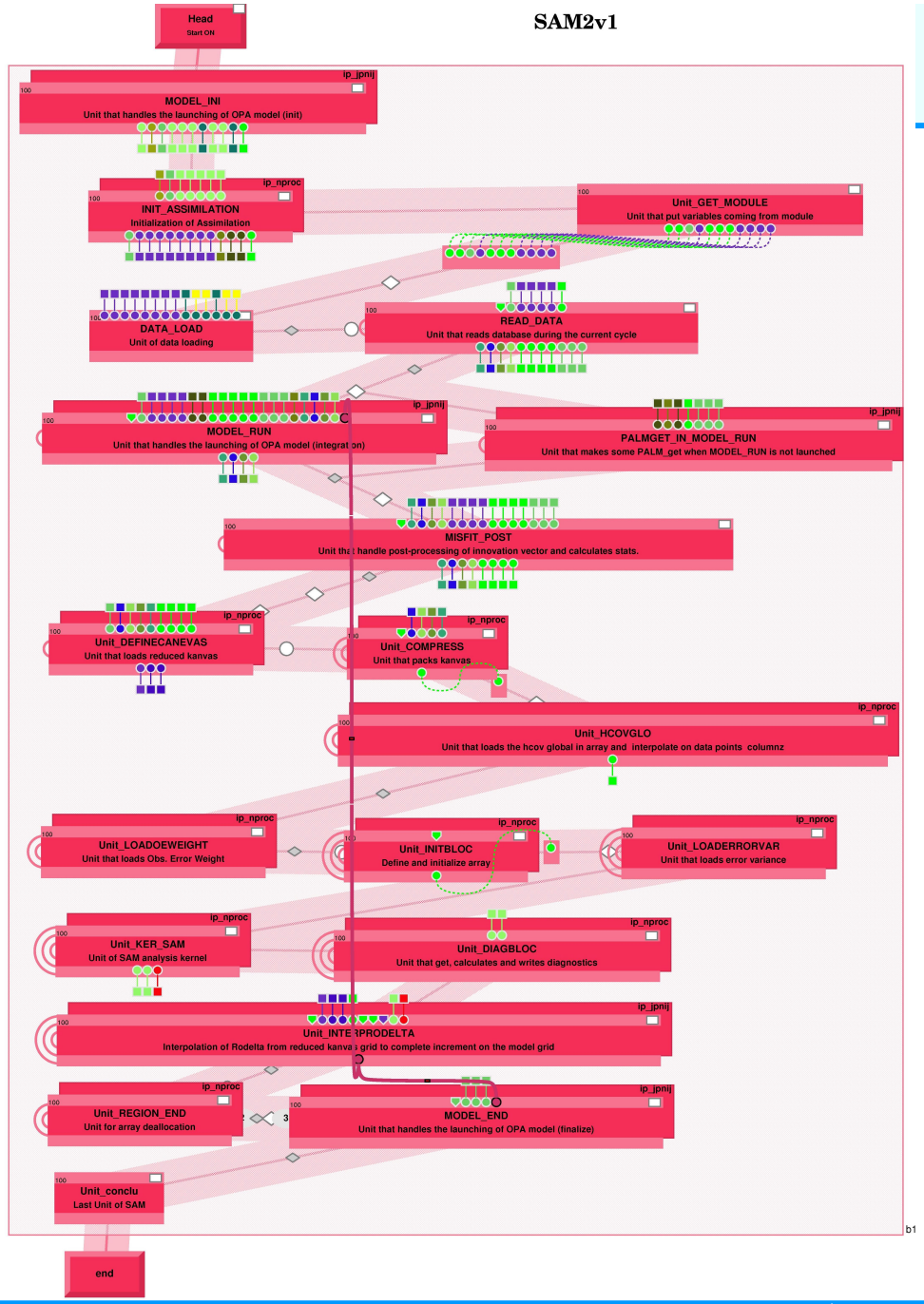

atmosphere
Start ON
biosphere
Start ON
ip_atmo_nbproc
100
atmo_par
parallel atmosphere toy model
ip_bio_nbproc
100
bio_par
toy parallel biosphere model
end
end


Change communication properties
Unit source name : atmo_par
Source Object : array_par_out.atmo_par
Source Distributor : reg_out_atmo_dist.atmo_par
Sub-object descriptor : vector_subobject
Unit target name : bio_par
Target object : vector_par.bio_par
Target Distributor : uneven_2_4_dist.bio_par
Sub-object descriptor : IDENTITY
Time list : 0:ip_nhours*3600-900:900
Local. assoc. : AUTOMATIC
Palm debug status : PL_NO_DEBUG
Palm track : PL_NO_TRACK
Data managment : MEMORY
Optimisation : PL_NO_BLOCKING
Cancel
Update

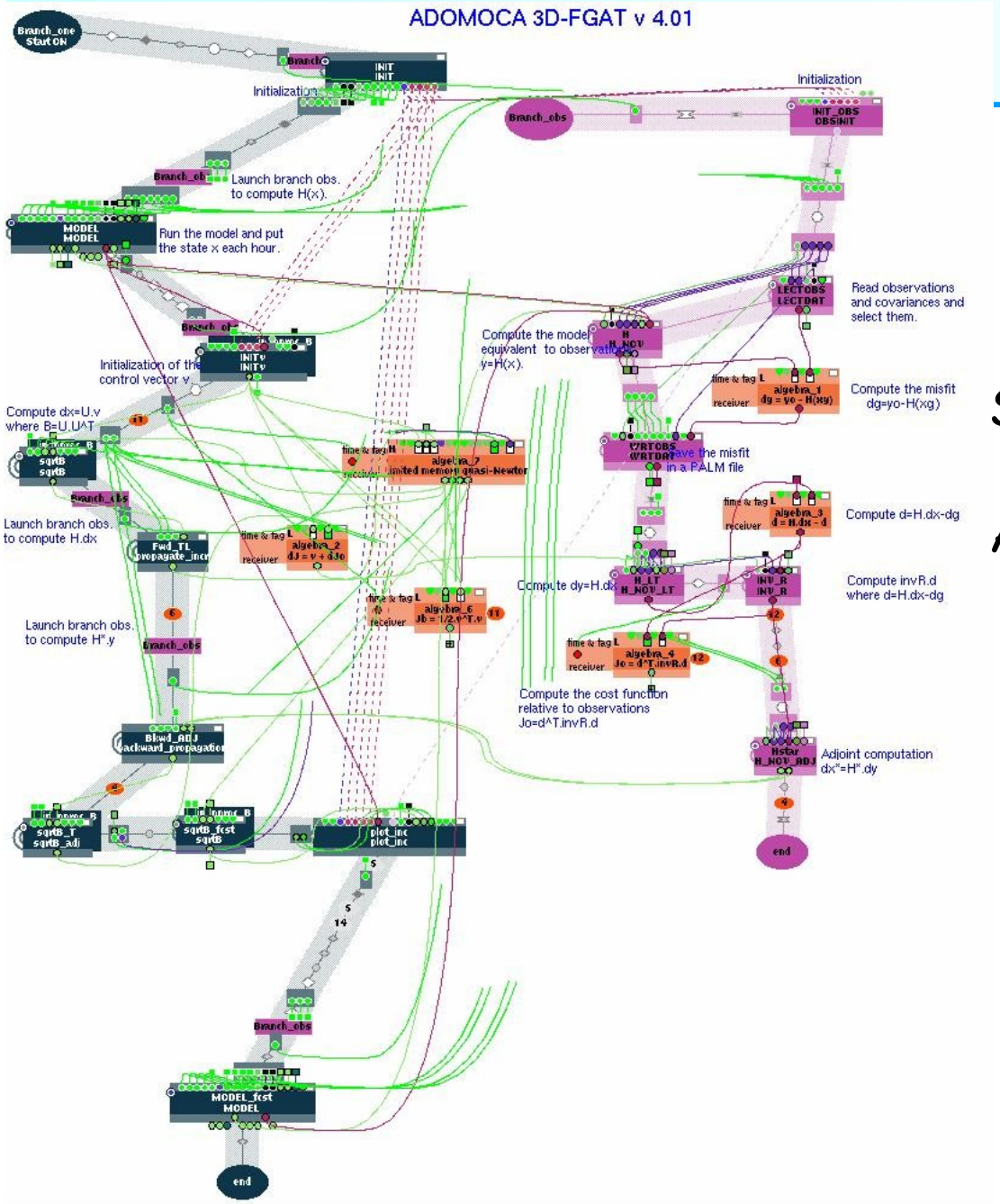
IMK - MOBILE


atmosphere
Start ON
biosphere
Start ON
ip_atmo_nbproc
100
atmo_par
parallel atmosphere toy model
ip_bio_nbproc
100
bio_par
toy parallel biosphere model
end
end


Change communication properties
Unit source name : bio_par
Source Object : vector_par.bio_par
Source Distributor : uneven_2_4_dist.bio_par
Sub-object descriptor : IDENTITY
Unit target name : atmo_par
Target object : array_par_in.atmo_par
Target Distributor : regw_atmo_dist.atmo_par
Sub-object descriptor : vector_subobject
Time list : 0:ip_nhours*3600-900:900|i+900
Local. assoc. : AUTOMATIC
Palm debug status : PL_NO_DEBUG
Palm track : PL_NO_TRACK
Data managment : MEMORY
Optimisation : PL_NO_BLOCKING
Cancel
Update



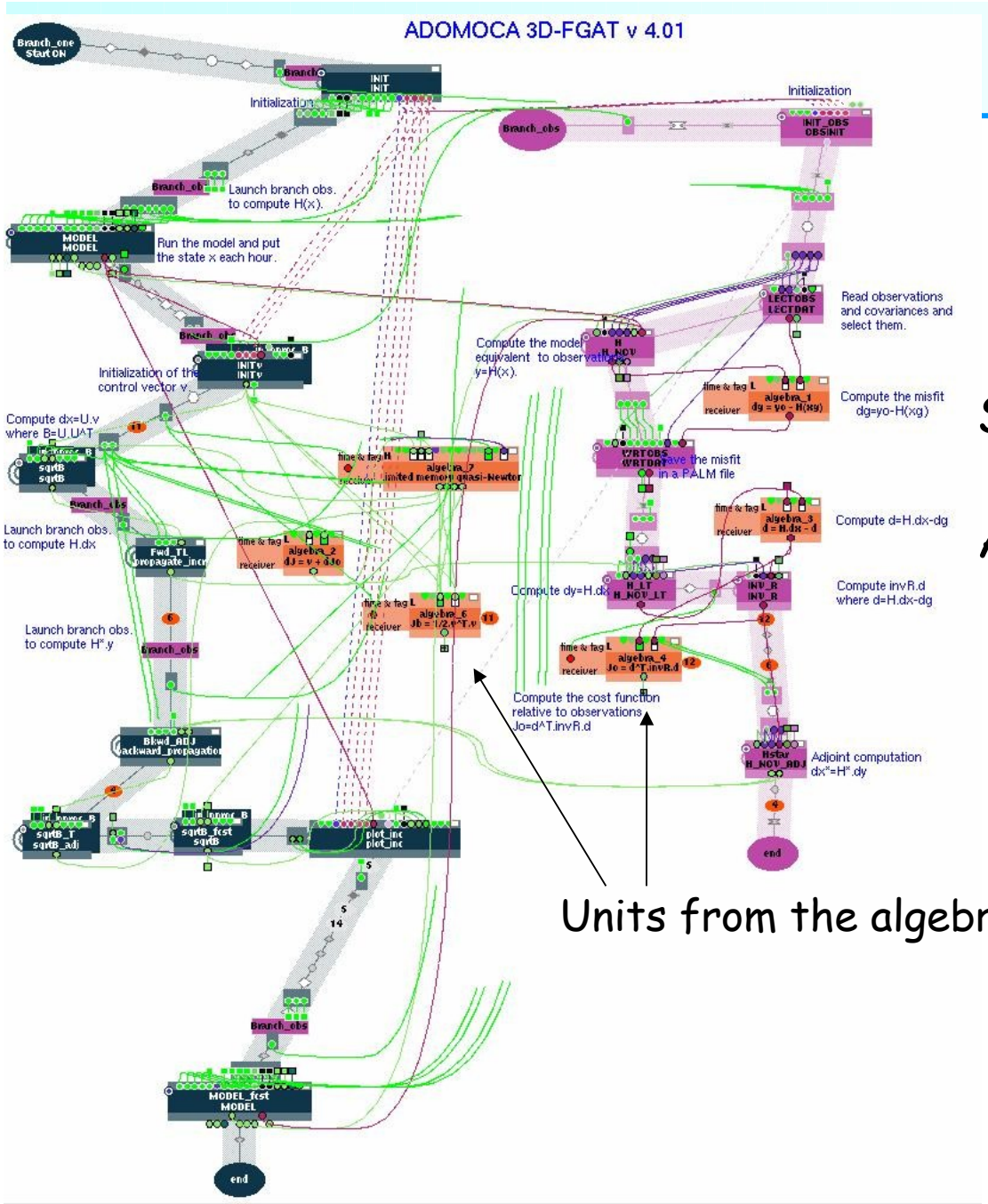
Complex algorithm
High performances
… as required!



Valentina

Still using PALM_SP

And V6 is more complex!

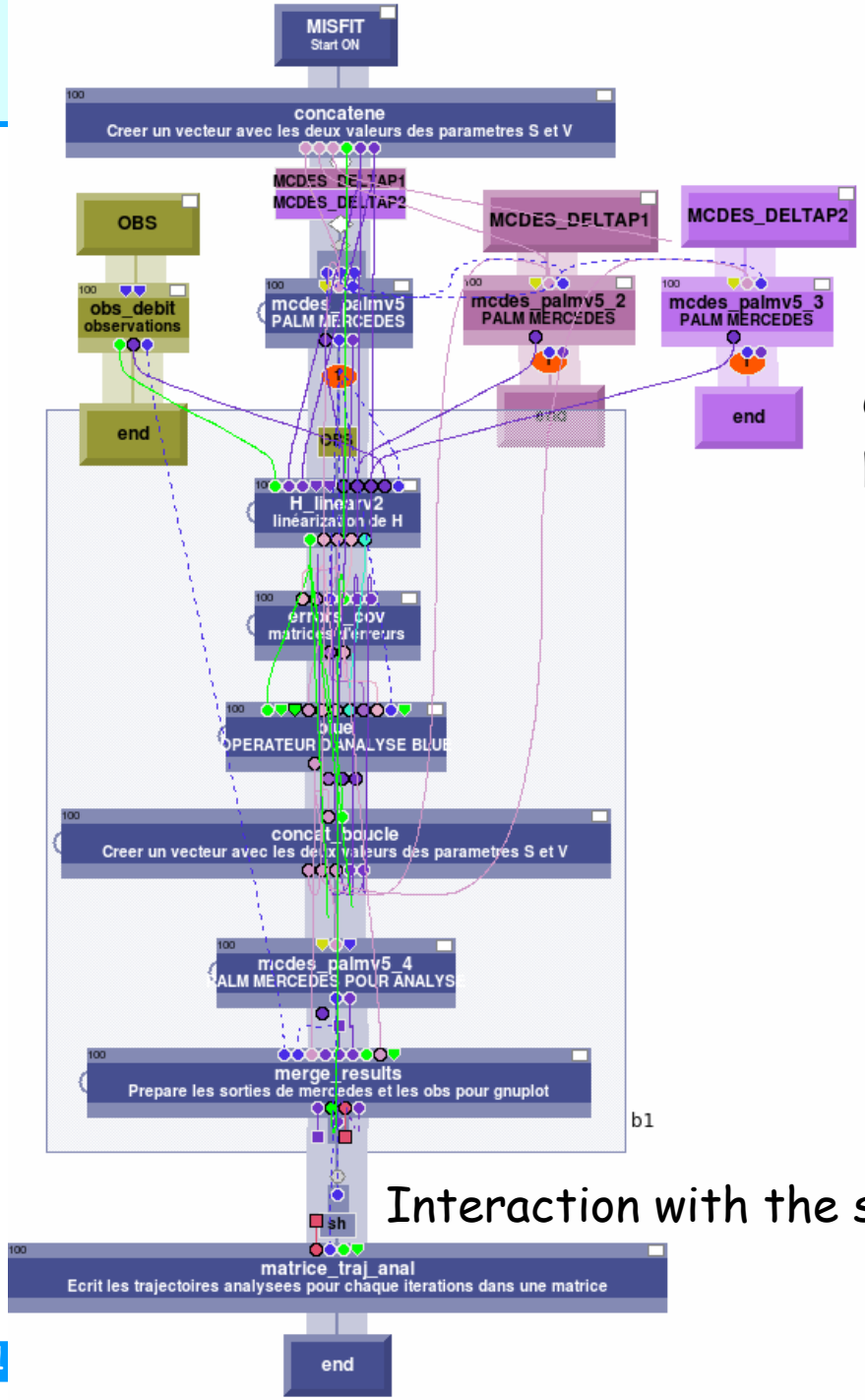


Still using PALM_SP

And V6 is more complex!

Units from the algebra toolbox

Hydrology



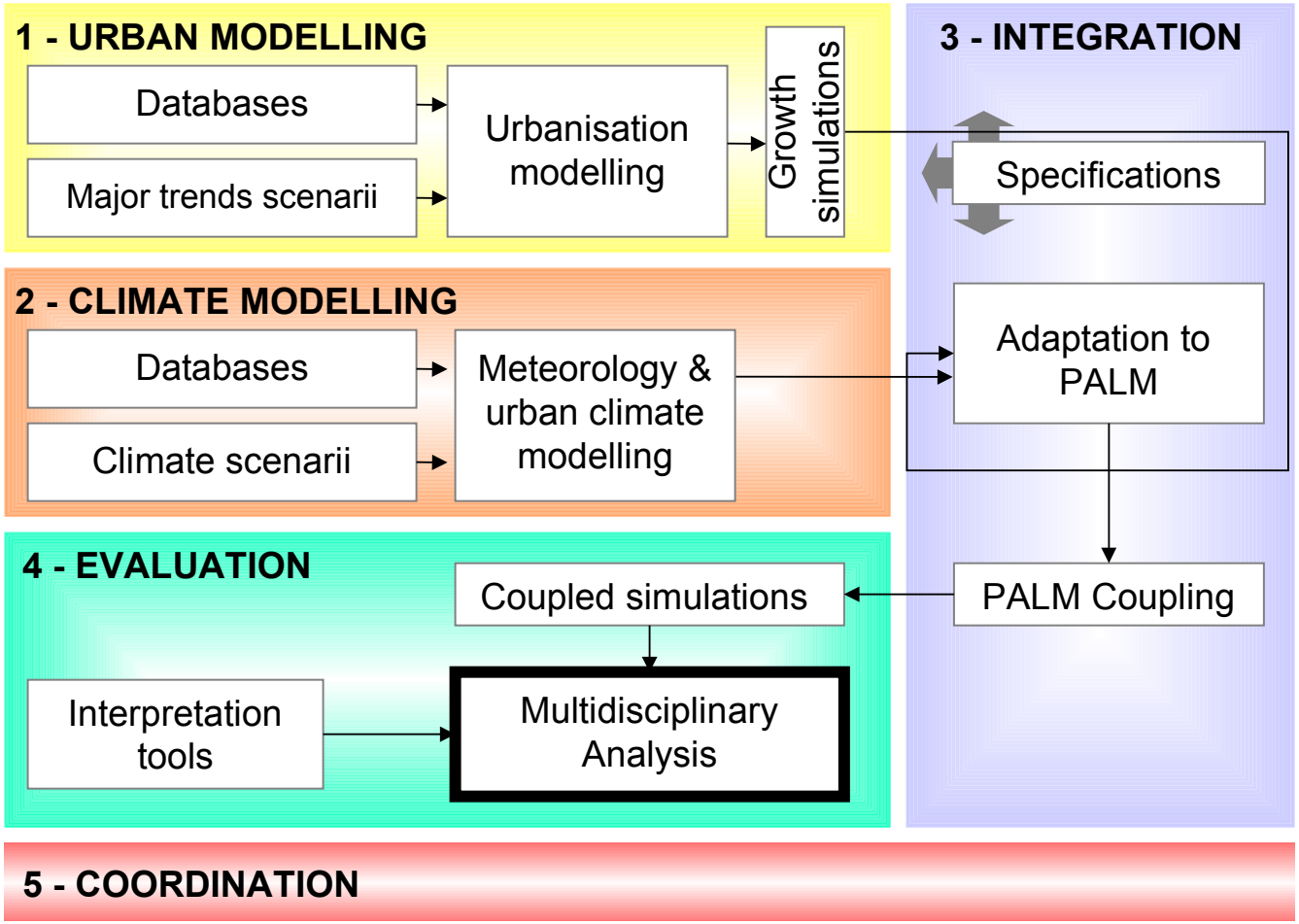
Code reuse for parallel perturbed runs

Interaction with the system for file handling

Urban scale climate evolution

ACCLIMAT project

The ACCLIMAT project has been funded with the support of the STAE-Toulouse Cooperation Foundation



Urban scale climate evolution