Towards extended validation of new branches and main cycles.

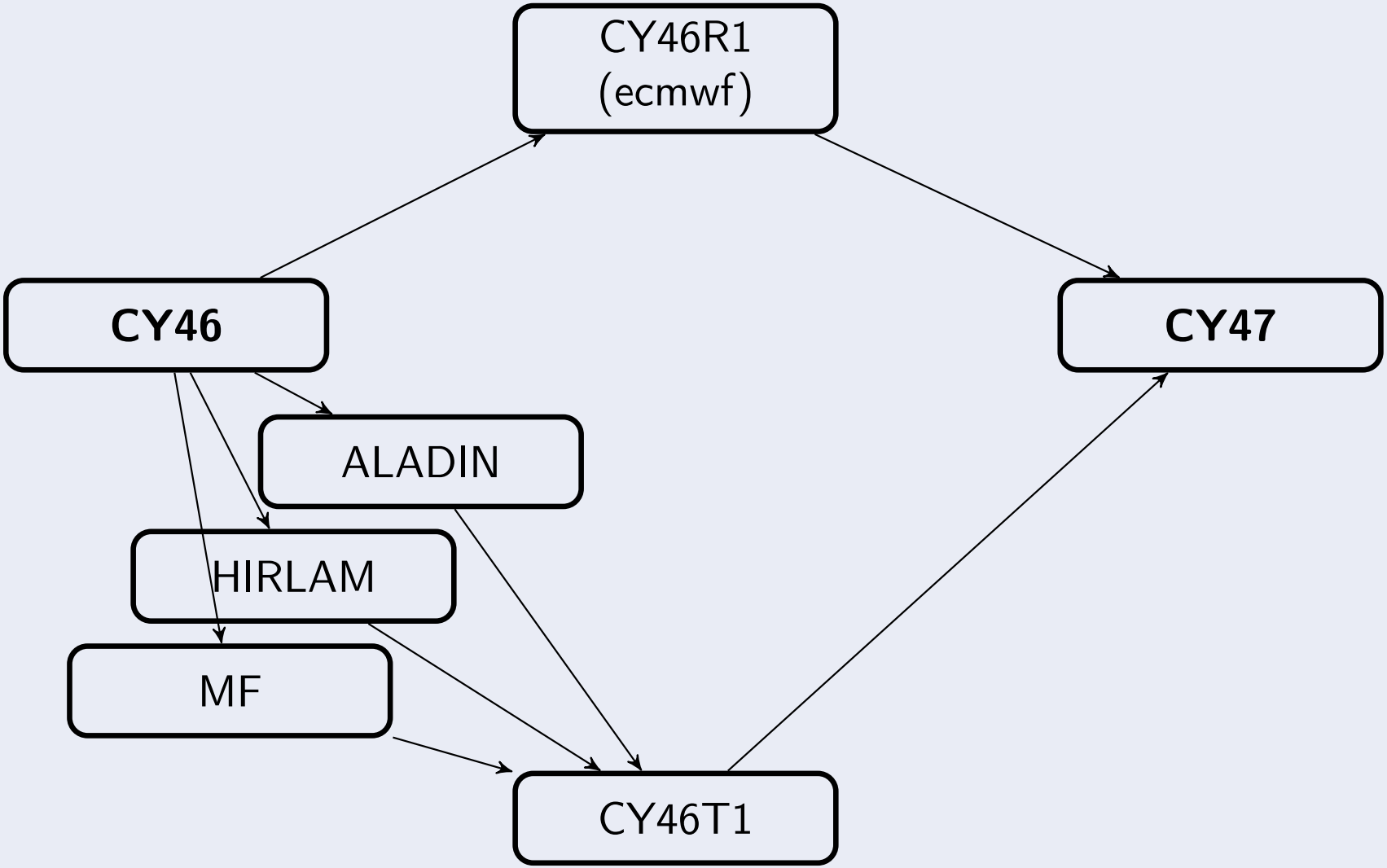
Alexandre Mary, Météo-France

ALADIN-HIRLAM Wk/ASM 2019 - Madrid

Outline

Outline

A collaborative code



Merge

- **T cycle** : merge of individual or grouped pre-merged branches is done at MF, eventually in several passes.
  $\hookrightarrow$ ***pre-cycle***, e.g. `CY46_t1.nn` | then phasing/corrections $\rightarrow$ `nn += 1`, till declaration of `CY46T1`
- **Common cycle** : merge of R + T cycles is done at MF.
  $\hookrightarrow$ ***pre-cycle***, e.g. `CY46T1_r1.nn` | then phasing/corrections $\rightarrow$ `nn += 1`, till declaration of `CY47`

Merge

- **T cycle** : merge of individual or grouped pre-merged branches is done at MF, eventually in several passes.
  ↪ ***pre-cycle***, e.g. `CY46_t1.nn` | then phasing/corrections → `nn += 1`, till declaration of `CY46T1`
- **Common cycle** : merge of R + T cycles is done at MF.
  ↪ ***pre-cycle***, e.g. `CY46T1_r1.nn` | then phasing/corrections → `nn += 1`, till declaration of `CY47`

Phasing/corrections :

- adaptations : e.g. IFS modifications to be reported to LAM code
- validation : ***Mitraillette*** tool : a collection of jobs, `NCONF` $\in \{1, 401, 501, 601, 901, 923, 927\}$ with various geometries & model options.

  All jobs must reproduce the ***reference*** (usually, the previous main cycle), or differences must be understood (hardcoded changes).

Difficulties from uphill

- Build of a T-cycle : ***insufficiently validated branches***.
- Build of a common cycle : R-cycle not tested against our configurations

↪ many ***broken confs*** & hard to find back the problems

Difficulties from uphill

- Build of a T-cycle : ***insufficiently validated branches***.
- Build of a common cycle : R-cycle not tested against our configurations

↪ many ***broken confs*** & hard to find back the problems

Difficulties in the process

Phasing/validation of a new cycle is difficult :

- ***Mitraillette*** is essential, but...
- some weaknesses : lack of ***global view***, chaining, duplication of code, maintenance, user-friendliness...
- scope of jobs to be re-defined ?
  👍 wide range of options
  👎 configs are too integrated vs. elementary tests (e.g. DFI w/o fcst)
- incomplete (DA postponed by *months*)

$\Rightarrow$ Need for more *systematic* and *elementary* testing.

$\Rightarrow$ Need for more *systematic* and *elementary* testing.

What to test ?

Testing in 2 directions :

- **Non Regression Validation** (NRV) : check that ***everything that enters*** is non regressive on all aspect of our NWP systems.
  $\hookrightarrow$ Test all branches on a small set of integrated ***core tests***

$\Rightarrow$ Need for more *systematic* and *elementary* testing.

What to test ?

Testing in 2 directions :

- **Non Regression Validation** (NRV) : check that ***everything that enters*** is non regressive on all aspect of our NWP systems.
  $\hookrightarrow$ Test all branches on a small set of integrated ***core tests***
- **Exploration and Localization of Problems** (ELP) : where does a problem come from ? (which part of the code and from which contribution)
  $\hookrightarrow$ Split tests in ***small*** bits, as small as possible, to guide identification of crashes or numerical changes in results.

All of this covering the ***widest possible scope of options in the code***.

What is a "test" ?

Tentative definition :

> ***Execution of a +/- integrated component of the code, with an expected result to reach (possibly with a tolerance)***

Different kinds of tests :

- ***continuity*** : target result comes from a reference execution of that test (previous version) — e.g. as in usual use with *Mitraillette*
- ***consistency*** : target result comes from another job from the same set of tests — e.g. test *simple precision* forecast vs. *double precision*
- ***auto-test***, or ***implicit target test*** : target result is implicit to the test itself — e.g. adjoint-test (of obs-op within OOPS, or conf 401)

How to define one test ?

A test is composed of :

- ***input resources*** (initial state, obs, static parameters files, namelist, ...)
- ***environment*** and ***execution*** modalities (MPI, env variables, ...)
- ***output*** and expected result (norms, Jo, RMSE...)

How to define one test ?

A test is composed of :

- ***input resources*** (initial state, obs, static parameters files, namelist, ...)
- ***environment*** and ***execution*** modalities (MPI, env variables, ...)
- ***output*** and expected result (norms, Jo, RMSE...)

What's the "status" of a test ?

Status $\in$	OK	success : expected result is reached
	KO	fail
	! ?	suspicious or unknown result (e.g. for technical reasons)
	X	test crashed

How to test DA

- higher complexity (ODB, environment, input files) + tasks dependency (bator > screening > minim).
  ↪ *Mitraillette* probably not suitable
- ODB usually not compatible from cycle to cycle
- **OOPS** enables simpler tests of restrained parts of the code.
  ↪ observation operators "*out of the box*"
- test ***obstypes*** separately
- separate tests of ***obs operators*** from tests of ***DA algorithms***
- get ***model*** out of the tests of ***DA*** (to be able to start both in //)

Outline

A/ checkpack

Development of a ***user-friendly*** wrapper of *Mitraillette* for contributors to test their code on a subset of jobs :

- applies to a single binary or a *pack* (gmkpack)
- includes a simple jobs monitor to avoid *Mitraillette* chaining and run jobs *partly in parallel*.

A/ checkpack

Development of a ***user-friendly*** wrapper of *Mitraillette* for contributors to test their code on a subset of jobs :

- applies to a single binary or a *pack* (gmkpack)
- includes a simple jobs monitor to avoid *Mitraillette* chaining and run jobs *partly in parallel*.

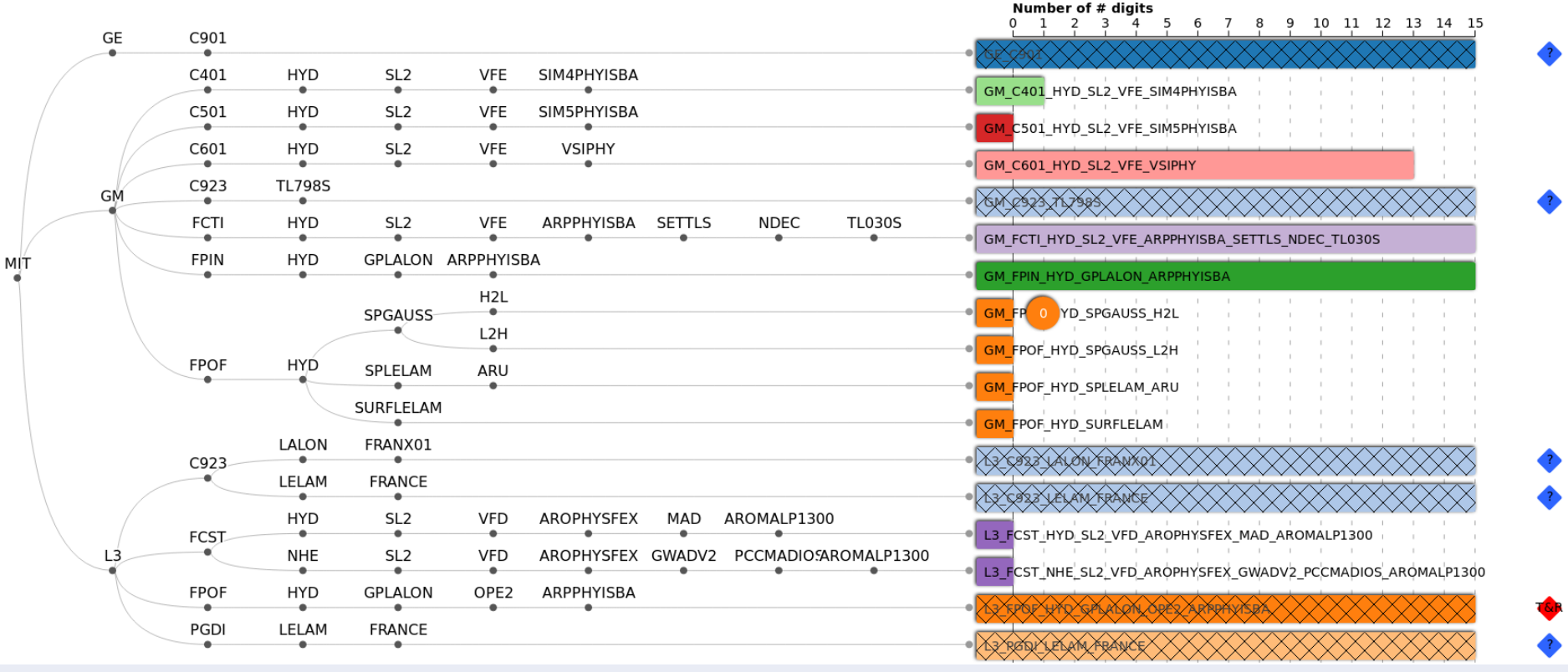
B/ ciboulette

Development of

- a tool to parse `arpifs` ***output listings*** through a *Mitraillette* tree, look for ***norms*** and ***compare*** to a reference *Mitraillette* execution.
- a graphical interface to have a quick ***global view*** of a whole bunch of *Mitraillette* tests. Goal being to ***raise focus*** on crashed tests or tests diverging from reference.
- plugged on **checkpack**

B/ ciboulette

Number of jobs: 17
Compared: 11 || Num differences: 4 | Bit-repro: 7
Unknown status: 5
Crashed: 1 || Test: 1 | Ref: 1 | (Both: 1)
Missing outputs: 0 || Test: 0 | Ref: 0 | (Both: 0)



C/ systematical branch testing

For CY46T1, an experimentation of systematical testing of all GIT branches integrated in the process of building CY46T1 was led *a posteriori* (after merge), almost automatised.

$\Rightarrow$ proved to be **very appropriate** :

- raised compilation/execution ***problems in individual branches***, that were actually met during merge, or corrected afterwards
- ***guided the research*** of numerical changes to the actual guilty contributions, hence to the appropriate corrections
  (e.g. changes in ALARO surface forecasts, caused by modifications in ACHMT in an Arpege data assimilation branch).

All of this

$\Downarrow$

Time to build a new validation system ?

Outline

We want

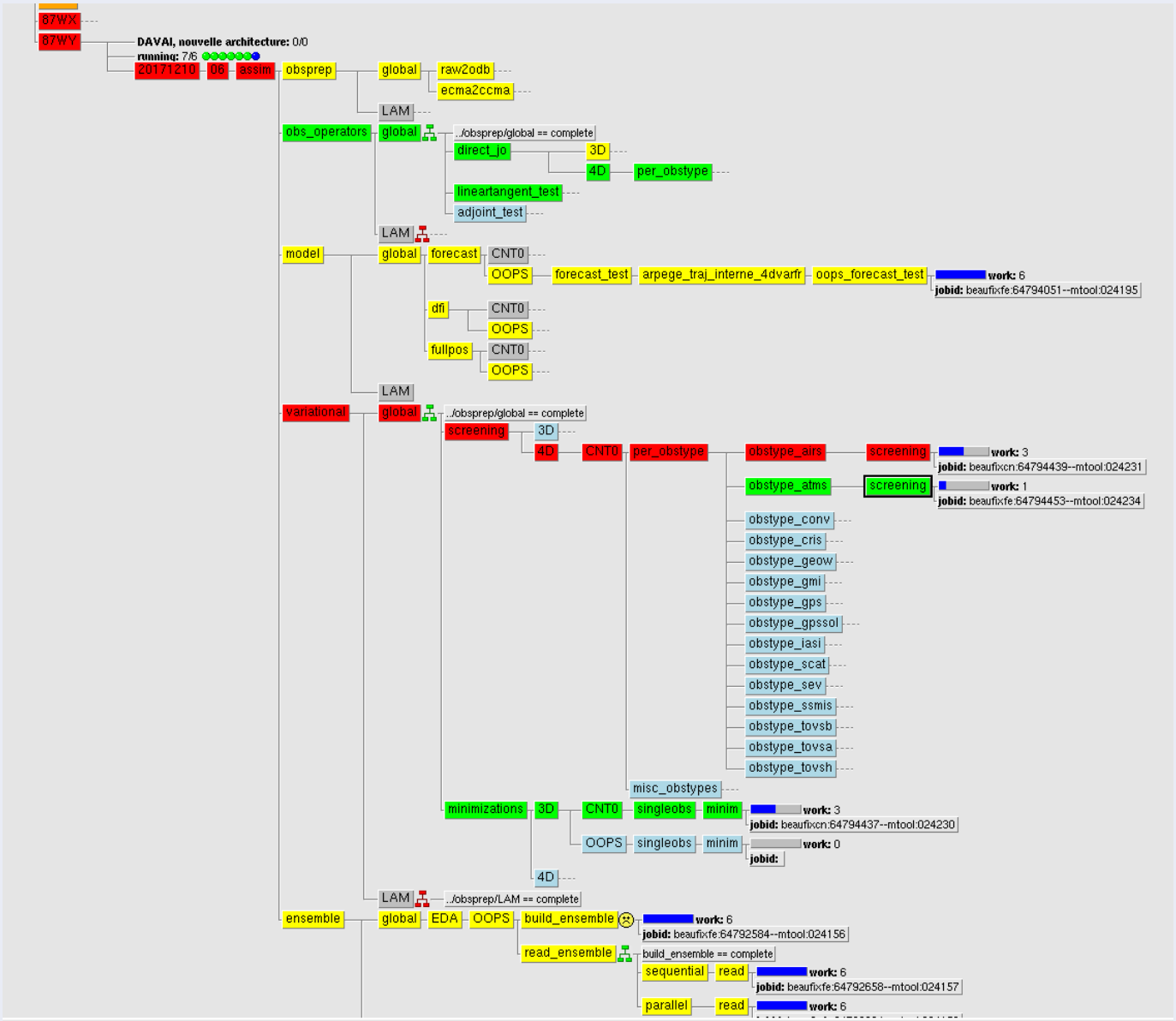
- ***graphical interface*** (global status view) + summary of each test status (incl. performance)
- wide range (*Mitraillette* + **DA** + ...) of ***small***[a] tests
- portability : probably not as such (jobs scripts), but as precise tests definition ?
- ***systematisation*** : a branch must pass the tests to enter a new cycle
- developers/scientists invited to propose their test configuration : ***opportunity*** for early validation rather than a constraint
- technical implementation : python, JSON, html5...

a. in terms of *code components* as well as *computing resources*

Prototype

- name proposal : ***DAVAI***
  (**D**ispositif d'**A**ide à la **V**alidation d'**A**rpege-**A**rome-**I**FS)
- **CY46T1**, toy T149 Arpege only for now :
  - BATOR
  - OOPS observation operators (D/TL/AD) *per obstype* – 3D/4D
  - screenings *per obstype* – 3D/4D
  - OOPS vs. CNT0 "*single obs*" 3D/4D-Var minimization
  - a few OOPS tests of : forecast, DFI, FullPos
  - OOPS tests for handling of ensembles (towards EnVar)
- *soon* : LAM versions, IFS forecast, *Mitraillette* jobs incl. ***revision of their scope***, tools for parsing and analysing results automatically

DAVAI


87WX
87WY
DAVAI, nouvelle architecture: 0/0
running: 7/6
20171210
06
assim
obsprep
global
raw2odb
ecma2ccma
LAM
obs_operators
global
../obsprep/global == complete
direct_jo
3D
4D
per_obstype
lineartangent_test
adjoint_test
LAM
model
global
forecast
CNT0
OOPS
forecast_test
arpege_traj_interne_4dvarfr
oops_forecast_test
work: 6
jobid: beaufixfe:64794051--mtool:024195
dfi
CNT0
OOPS
fullpos
CNT0
OOPS
LAM
variational
global
../obsprep/global == complete
screening
3D
4D
CNT0
per_obstype
obstype_airs
screening
work: 3
jobid: beaufixcn:64794439--mtool:024231
obstype_atms
screening
work: 1
jobid: beaufixfe:64794453--mtool:024234
obstype_conv
obstype_cris
obstype_geow
obstype_gmi
obstype_gps
obstype_gpssol
obstype_iasi
obstype_scat
obstype_sev
obstype_ssmis
obstype_tovsb
obstype_tovsa
obstype_tovsh
misc_obstypes
minimizations
3D
CNT0
singleobs
minim
work: 3
jobid: beaufixcn:64794437--mtool:024230
OOPS
singleobs
minim
work: 0
jobid:
4D
LAM
../obsprep/LAM == complete
ensemble
global
EDA
OOPS
build_ensemble
work: 6
jobid: beaufixfe:64792584--mtool:024156
read_ensemble
build_ensemble == complete
sequential
read
work: 6
jobid: beaufixfe:64792658--mtool:024157
parallel
read
work: 6

Merci de votre attention !