Caching ESMvalTool results

S.Sénési – jan 2022

CliMAF implements a results cache which proved to be useful and handy

Similar principles could be applied to ESMValTool

CliMAF results handling

- CliMAF objects are either graphic or numeric
- CliMAF handles a symbolic expression for each object (CRS – CliMAF Reference Syntax expression)
  - The expression fully defines the object. It is built step-wise during object construction (this actually implements provenance handling)
  - An expression interpreted by python (with CliMAF extension) returns the actual CliMAF object, which can be evaluated to a file result.
  - The syntax allows user scripts to have multiple outputs and to give them distinct labels (or symbolic names)
  - Example :

    principal_components(ds(project=CMIP6, experiment=historical, variable='tas', period='2012-2013', ...)).first_eigen
- The expression translates to a filename in CliMAF cache
  - This, using a secure hash digest of the CRS expression (hashlib.sha224)
  - User is free to use this filename rather as a (hard or soft) link to its choice of filename
- CliMAF systematically stores results in cache and checks cache content for avoiding to re-compute an object
  - Escape mechanics do exist

ESMValTool results handling

- ESMValTool results are defined by a cascade :
  - *recipes* include one or more *diagnostics* and define *datasets*
  - *variables* of *datasets* underwent some *preprocessing(s)*
  - *preprocessing* results feed *scripts*
  - more complex cases are allowed *(ancestor tasks)*

- Scripts may create multiple outputs, which are described **after run,** through a provenance file, by a caption and ancestors.
- ESMValTool has no way to **anticipate** which (or even how many) results will be created ; it gets informed after run by the provenance file.
- When faced with multiple outputs, it cannot ‘interpret’ which is what (captions have no convention for that)
- However combining recipe parameters (for pre-processings and scripts) and recipe structure, plus *ancestors*, basically allows to fully define each **script** run, before run

Caching ESMValTool results

- **Basic need** is that scripts generating multiple outputs provide **a distinct label for each output**

  Through some kind of declaration, that link such labels to output filenames
- And that scripts declare when they use only a part of ancestor’s tasks output (using those labels)
- **A syntax** for expressions fully defining scripts results would have to be designed (and coded)
  - Either **use recipe structure plus output labels**

    Syntax must not distinguish results which are identical :
    - define a canonical form (e.g. for parameters order),
    - superfluous recipe parameters should not be part of the definition
    - forbid default parameter values (or declare them) …
  - Or **serialize ordered provenance trees** provided they can also be computed before run
- A quasi-unique identifier for each result could then be derived
  - Which is the basis for implementing a cache and avoiding to re-compute results
- Driver added duties :
  - Store outputs (when user activates caching)
  - Before launching a script, compute the expressions for all its outputs, and test if they are in cache

Referencing ESMValTool results