EUROPEAN COMMISSION

Community research

Climate Information Portal

CLIPC DELIVERABLE (D -N°: 4.4)
Integrated toolbox report

File name: CLIPC_Deliverable-4.4-Integrated-toolbox-report.pdf

Dissemination level: PU (public)

Author(s): *Maarten Plieger, Ernst de Vreede, Andrej Mihajlovski, Alessandro Spinuso, Alessandro Benincasa, Sandro Fiore, Wim Som de Cerff*

Reviewer(s): *Peter Thijsse, MARIS Sandro Fiore, CMCC*

Reporting period: *01/12/2013 – 31/11/2017*

Release date for review: 28-10-2016

Final date of issue: 24-11-2016

Revision table			
Version	**Date**	**Name**	**Comments**
1	October	Draft	Circulated for discussion
2	October	Review	For internal review
3	November	Final copy	Final version

Abstract

The integrated toolbox deliverable (D4.4) is a technical document, describing WPS services, provenance functions, connection of CLIPC and Climate4Impact, the use of Access tokens, the storage API, combine functions and the Caching system. All these services provide the backend for the toolkit functionality as provided to users via the CLIPC portal user interfaces: compare, combine, MyCLIPC and indicator processing.

Project co-funded by the European Commission's Seventh Framework Programme (FP7; 2007-2013) under the grant agreement n°607418

Table of contents

Executive Summary

The CLIPC architecture has been build using Open Standards to provide well defined interfaces between data, service back-end and the end user portal. Data is provided in standardized NetCDF-CF format. This enables easy integration into the back-end services. The back-end services are provided to a front-end portal, enabling the CLIPC toolkit functions: Compare, Combine, Calculate indices and MyCLIPC.

The Compare functionality is provided by the ADAGUC WMS service, which is capable to automatically generate a WMS service on NetCDF-CF data.

The Combine functionality is a co-development with WP8. In the combine function two maps can be normalized and combined, using a combine function. At this moment 3 different normalization functions (none, min/max, Z-score) and 8 combine functions (Add, Subtract, Multiply, Divide, Multiply, Less, Equal, Greater) are provided. Besides the result, also the provenance metadata is stored in the resulting NetCDF-CF file. The combine function uses the WCS and WPS standards for implementation.

Calculate indices is provided by using indices calculations functions from ICCLIM, using pyWPS as implementation for the WPS standard. For processing services a caching system was developed and tested.

MyCLIPC enables users to store, combine and calculate results and to check on longer running processes. The user simply logs in using a Google account. This functionality is provided using the StorageAPI of Climate4Impact.

All is secured using Public Key Infrastructure (PKI), OpenID and OAUTH2.

The data and backend services are hosted at KNMI in the Climate4Impact portal and in the ESGF network. The CLIPC front-end portal is hosted and operated by MARIS, at a different location.

The use of Climate4Impact services and expanding on this framework enabled CLIPC to have a head start and a proven platform for development. The services can easily be expanded to provide new and improved services. Next steps include expanding the Combine services with compatibility guidance for the creation of sensible combinations (or issue a warning). A skeleton WPS has already been made available for this purpose.

1. Introduction

The CLIPC architecture is relying on Open Standards to provide well defined interfaces between data, service back-end and the end user portal. The architecture is depicted in Figure 1. On the left side all data providers are mentioned. They deliver the data in NetCDF-CF format. This enables easy integration in the backend, which provides services to the front-end portal. The data and backend services are hosted at KNMI and in the ESGF network; the frontend portal is run and hosted by MARIS, at a different location.

The WPS services run at the Climate4Impact portal, close to the indicator data. They are invoked through the graphical user interface in the CLIPC portal (on the right side).

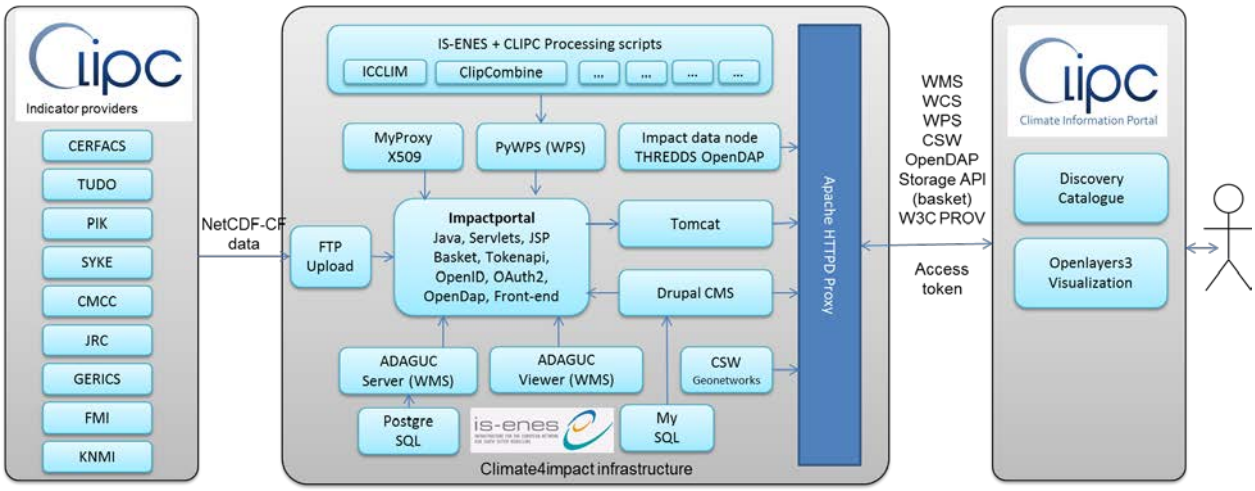

Figure 1: Overview of CLIPC services, standards and data

Web services in general

Figure 2 describes several open standards and the type of work they can do for a client.

- HTTP/FTP download: A client/researcher does all the work by himself, downloads a huge amounts of data, does data format interpretation, file convention interpretation and needs to understand geographical projections.
- OPeNDAP [14]: A client/researcher only requests the piece of data he wishes, he gets the data in a harmonized way: a user gets data, not a file in a format. He still needs to understand the conventions and geographical projections.
- WCS: A client researcher can subset from a large dataset the data he really wants to use, he gets georeferenced data in the data format and geographical projection he wants.
- WMS: Similar to WCS, but instead of data, a graphical representation is given.
- WPS: A client/researcher tells the server to do a specific analysis task on selected datasets according to protocol from the getCapabilities of the WPS. He does not have

to download the data, when the analysis task is completed he just needs to work on the result. E.g. a daily to yearly averaging task could be such a job.

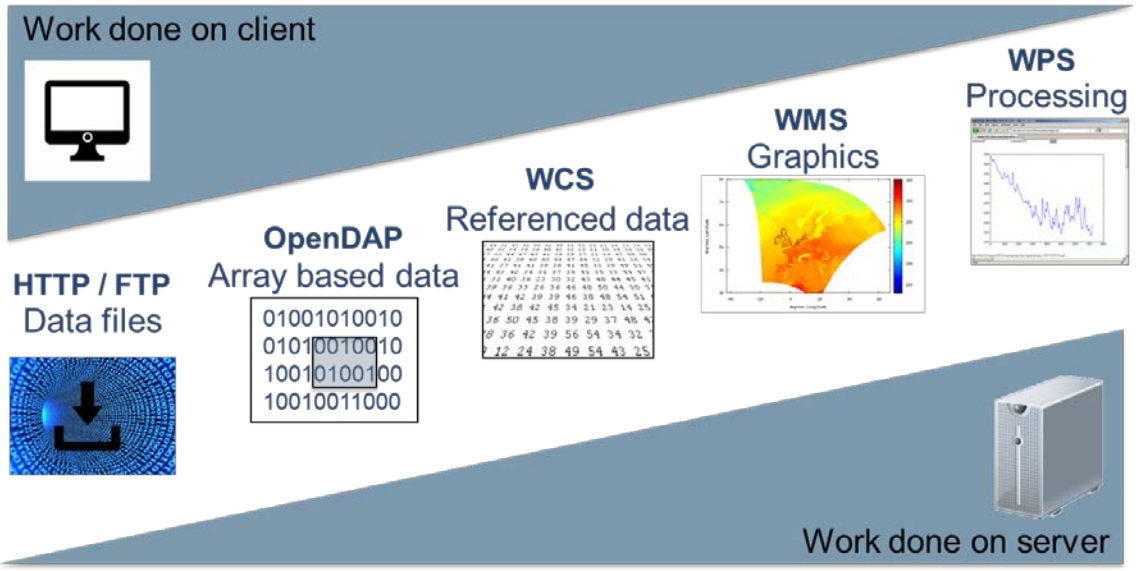

Figure 2 Work done on client and work done on server for several open standards.

2. Access restrictions and security

Using access tokens for connecting CLIPC portal and climate4impact WPS services
The CLIPC portal needs to access Climate4Impact's webservices in a secure way. This is arranged via Public Key Infrastructure (PKI). The following explains how MARIS (and other portals) can access webservices at KNMI/CLIPC using PKI x509 client authentication.

These actors come into action:

- C4I: Climate4impact portal running at KNMI, exposes webservices for other portals. Currently only the accesstoken webservice can be used over PKI, see API.
- CA: KNMI-CLIPC Certificate Authority installed at C4I for the CLIPC project
- MARIS: The CLIPC portal running at MARIS, wants to access webservices at climate4impact using PKI

A certificate authority is created at KNMI/C4I, and is registered in an Apache HTTP server at climate4impact (C4I). MARIS registers users at C4I and is able to access C4I webservices with a private key and a certificate in a secure way.

The access token API at C4I
The webservice that can be accessed over PKI is the access token API at Climate4Impact. The access token API can register new users and generates tokens (uuid v4) for designated users. These tokens can be used temporarily in URL's to gain access to most webservices at C4I,

like WMS, WCS, WPS and the user basket with its - per user secured - OPeNDAP server. An important aspect here is that this is done over HTTPS, as the accesstoken is part of the URL.

Access to ESGF resources
Within CLIPC Google OAuth2 is used to obtain user ID's. As Google ID's have no meaning for ESGF, it is currently not possible to access restricted ESGF resources in the CLIPC portal. When CEDA OAuth2 is used, and the user is registered at ESGF for the specific role, it will be possible to access ESGF resources with webservices at C4I. Keep in mind that it is always possible to access unrestricted ESGF resources, like the data from the CLIPC project that stored at ESGF nodes.

Detailed instructions are described here:
https://dev.knmi.nl/projects/impactportal/wiki/CLIPC_%3C--%3E_KNMI_Server_connection

3. WPS services in CLIPC

3.1 PyWPS implementation of the OGC Web Processing Service standard

PyWPS is a server side implementation of the OGC Web Processing Service (OGC WPS) standard, using the Python programming language. PyWPS is currently supporting WPS 1.0.0. Support for the version 2.0.0. of OGC WPS standard is presently being planned.

The OpenGIS® Web Processing Service (WPS) Interface Standard provides rules for standardizing how inputs and outputs (requests and responses) for geospatial processing services, such as polygon overlay. The standard also defines how a client can request the execution of a process, and how the output from the process is handled. It defines an interface that facilitates the publishing of geospatial processes and clients' discovery of and binding to those processes. The data required by the WPS can be delivered across a network or they can be available at the server.

3.2 CLIPC processing services

In CLIPC PyWPS is used for processing climate data. PyWPS is a flexible framework where tools from different environments can be integrated and harmonized into one infrastructure. For CLIPC the following processes have been developed:

- Combine: Combine two files accessible over OpenDAP
- Convert, regrid and subset: Extract a region and a set of dates from a dataset, regrid it and convert it to other formats suitable for scripting or GIS systems
- Climate indicator calculation: Calculates climate indicators based on the package ICCLIM

- Averager: Average a dataset to e.g. another time frequency can be used to convert daily data to a 30 year average.
- Nuts region extraction: Calculates statistics for a grid field by NUTS region (European wide definition of administrative areas), is used to calculate all mean, min or max values for a country or province.

3.3 Storage API

Climate4Impact offers a personal basket where users can upload their own data and do research with the provided tools. The basket supports formats like NetCDF and GeoJSON. The basket has an access token mechanism to make data sharing and command line access to webservices easier. This enables client side scripting of the Climate4Impact portal possible and makes it possible to connect third party portals, like the EU FP7 CLIPC portal.
Every user in the CLIPC/Climate4impact ecosystem has a personal basket where he can store and upload his own data. For each user data is secured by X509/OpenID and access tokens. The data stored in the basket becomes accessible via OpenDAP, enabling other webservices like WMS and WCS to visualize and access the data in an easy way. Results from processing services is stored in the same basket in a scratch folder and can be reused for further processing.

Figure 3 CLIPC data basket and Climate4impact basket, showing the same content.

For details see:
https://dev.knmi.nl/projects/impactportal/wiki/API#Basket-requests

3.4 Combine function

The combine function is a processing web service for combining indicators offered and produced by CLIPC. A tool is delivered for the array arithmetic combination of 2D temporal slices. OGC standards are used as interface for the CLIPC combination tool. The data is stored in NetCDF format using Climate Forecast conventions. Access to the data sets by the combination tool is facilitated using OpenDAP services. Downscaling to temporal slices is executed using WCS via services provided by the ADAGUC platform. The web processing service (WPS) is implemented as open source Python software; based on the pywps 3.2.5

libraries. Array arithmetic using numpy allows multiple methods of array arithmetic, piped normalization functions, provenance, and general data transformation tools via other WPS services. All outputs are cached locally for users to download in NetCDF format or to visualize in the indicator toolkit using WMS.

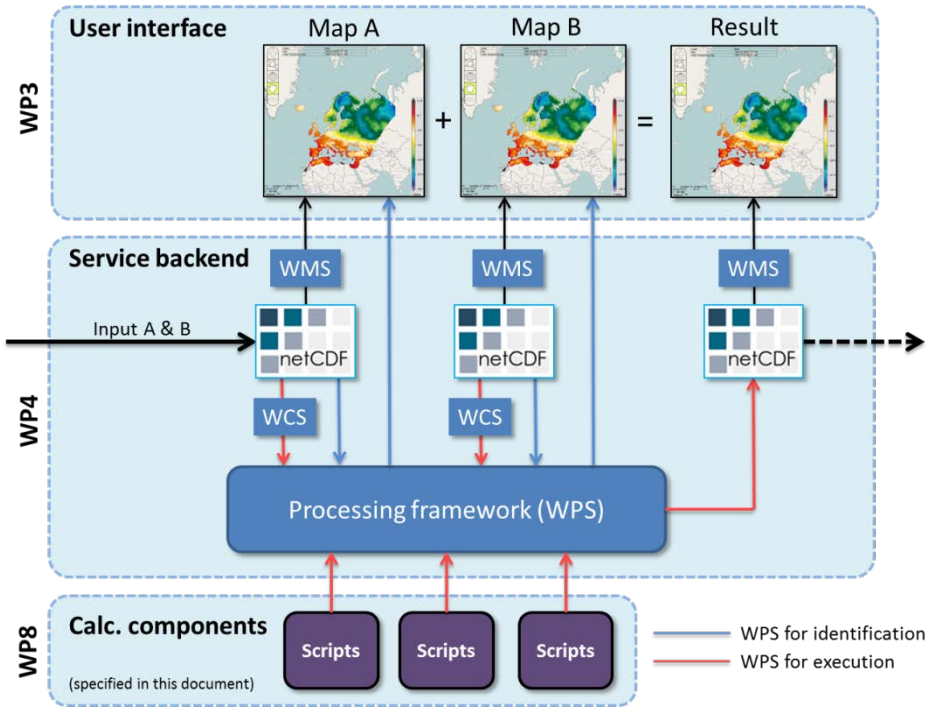

Figure 4 : CLIPC combine function architecture

The CLIPC portal, uses the front end interfaces shown in figure 4 below to visualise data processed via the Combine Function. The interface uses WMS services to visualise the two data sets and the output – see figure 3. The original data used here is accessed via an OpenDAP request to remote Thredds servers. All transformations of the output are executed on the Climate4Impact.eu (C4I) server using numpy. The WCS service which provides a single 2D temporal slice of each of the two inputs is accessed from KNMIs Adaguc web service. The output is stored on a local scratch drive. The contents become available for the same user on both the Climate4Impact.eu and clipc.eu portals. In this way it is demonstrating a distributed web based infrastructure for processing and visualisation.

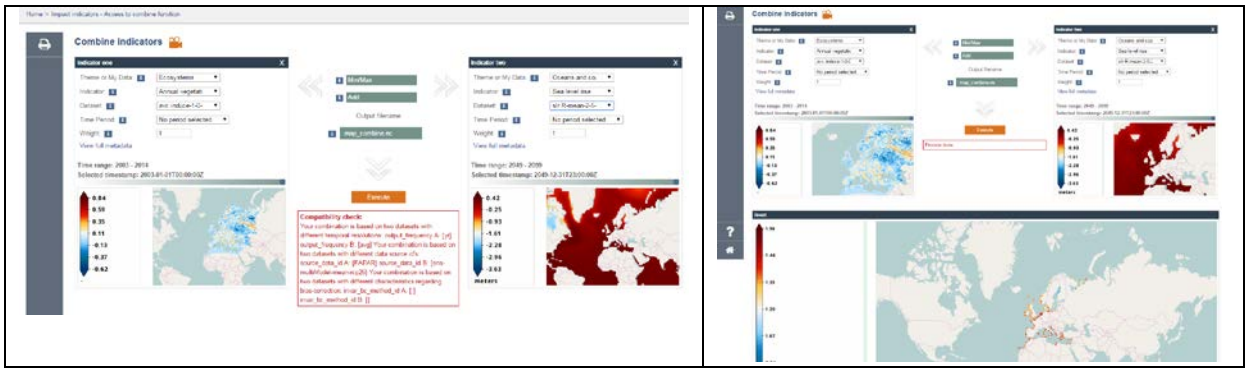

Figure 5 Combine function in CLIPC: compatibility check and advice (left image) and process executed anyway (image right)

The combine process web service backend, identified as "knmi_advanced_combine" is available as a WPS via the following service point:

https://climate4impact.eu/impactportal/WPS?

The rest API requires the following variables to describe the process:

service=WPS&version=1.0.0&request=describeprocess&identifier=knmi_advanced_combine

'title': 'CLIPC Advanced Combine',
'identifier': 'knmi_advanced_combine',
'abstract': 'KNMI WPS Process: CLIPC Advanced combine two inputs into a single netCDF. The combine function provides a visual exploration tool for dataset pairs. Any two datasets can be resized via wcs to a single time instance and compared using numpy arithmetic and normalisation tools. The two datasets being compared are normalised and can be weighted to provide improved visual comparison. The combine function is primarily an exploration tool, with a high level of uncertainty.',
'grassLocation': False,
'metadata': 'METADATA D4P',
'statusSupported': True,
'storeSupported': True,
'version': '1.0.0'

The inputs of the combine function are outlined in the following table.This input is reduced when using the combined view in the CLIPC portal. Access to this web service is also made available through other portals such as C4I.

Table 1 : Combine process input fields

input	*type*	*description*
netcdf_source1	Opendap link	Combine input: Source 1 netCDF opendap.
variable1	variable name	Combine input: VariableName 1.
weight1	float	Combine input: Weight 1 of netCDF input.
norm1	string ["normnone" , "normminmax", "normstndrd"]	Combine input: Norm 1 of netCDF input.

netcdf_source2	Opendap link	Combine input: Source 2 netCDF opendap.
variable2	Variable name	Combine input: VariableName 2.
weight2	float	Combine input: Weight 2 of netCDF input.
norm2	String ["normnone" , "normminmax", "normstndrd"]	Combine input: Normalisation 2 of netCDF input. Normalisation is implemented using
netcdf_target	file name	Combine input: Output netCDF.
operation	string ["add", "subtract" , "multiply" , "divide" , "equal" , "less" , "greater"]	Combine input: Operation. String representing the numpy equations equivalent.
bbox	Bounding box	Copy input: BBOX of WCS slice.
width	pixels	Copy input: width of WCS slice.
height	pixels	Copy input: height of WCS slice.
time1 Copy input: Time of WCS slice 1.	iso time	Time contained in the time dimension of the dataset
time2 Copy input: Time of WCS slice 2.	iso time	Time contained in the time dimension of the dataset
tags	String	Combine input: User Defined Tags CLIPC user tags. These are provenance related, allowing for interaction with the PROV-DM infrastructure described in the following chapter.

The normalisation functions in the CLIPC portal application focus on min/max and z-score implementations. However users of the web service have further access to linear normalisation and bounded normalisation. The breakdown of the various normalisation services available to users of the basket include: “knmi_norm_adv”, “knmi_norm_linear” and “knmi_weight”. The two NetCDF inputs for the combine function are open to all NetCDF OpenDAP links, including datasets available in the user’s MyCLIPC basket. However the dataset must contain a variable name and time component for selection. The tags input is used by the provenance architecture to provide added filtering for users.

3.5 Provenance in combine function

CLIPC provides access to a large range of datasets for climate modeling and impact. The datasets are complex in nature and are stored in self-described data frames using the NetCDF-CF format. The data is therefore easily accessible through web services by frameworks using R, Python, and Matlab. The uniqueness of the data is described in the attributes contained in each datasets NetCDF and are validated in detail by domain scientist focused on modeling

and impact studies. The combine functions allow users to bridge the climate modeling and impact domains, by allowing the outputs of different datasets to be compared and combined. The outputs carry an added complexity and high uncertainty, multiple uses of the web services create a need to monitor, visualise and understand the workflows which impact the changes to the datasets.

The standard method of tracking provenance within NetCDF datasets is the use of the history attribute. This is a free text field which is not used in a standard way by data users. Issues arise in interpreting changes to the data, and machine oriented provenance tracing is not possible. Thus a contemporary approach to provenance is taken and the result is stored as a W3C PROV standard XML output within a provenance variable of the NetCDF file. The provenance variables are read and updated each time the datasets are processed within the CLIPC portal. The use of the attributes from multiple inputs and piping all the changes occurring to a given dataset allows us to monitor and visualise provenance with standard tools.

The diagram below shows a visualisation of PROV-DM.

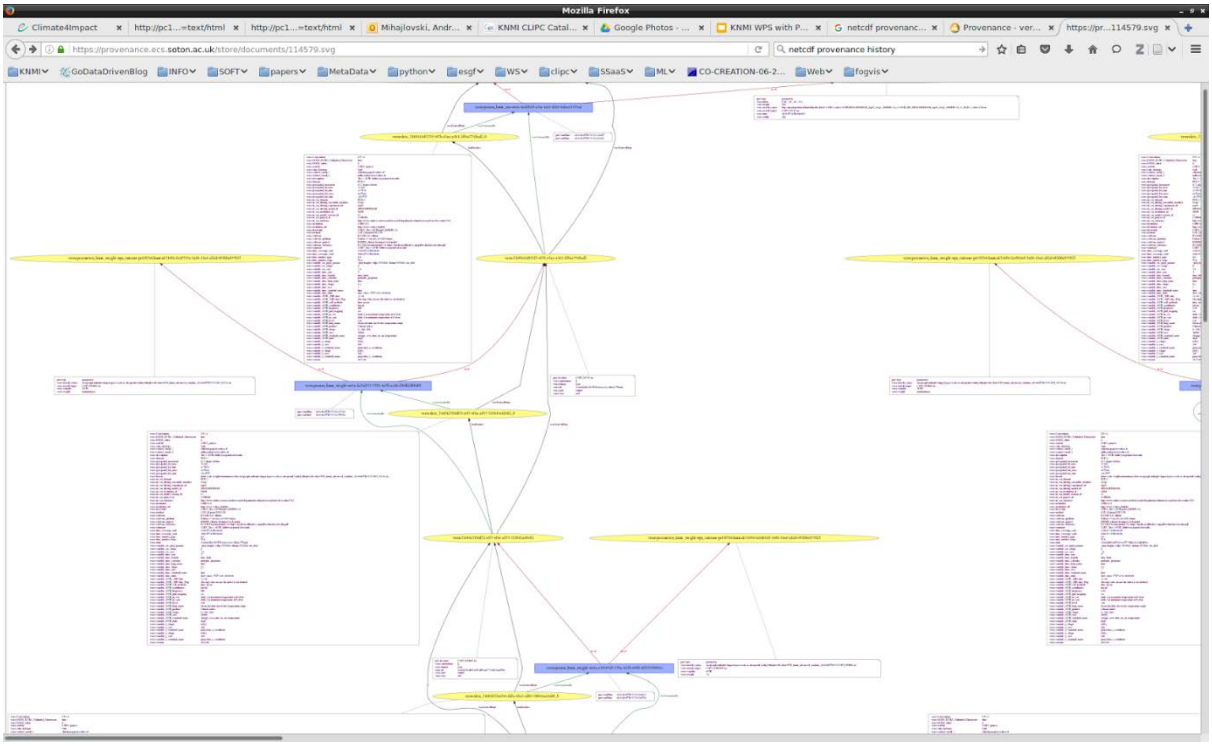

Figure 6 Visualization of PROV-DM

Provenance method and software tools developed for provenance tracking chains of data stored and processed via web services in a NetCDF CF format. All web processing services including the combine tool are integrated with provenance hooks and visualised using semantic web provenance standard (W3C PROV-DM). The provenance information is encoded within the NetCDF files as both json and XML. Visualization for users is made

available under the C4I portal through the catalog/basket web interfaces and is locally available through public online W3C PROV tools.

The diagrams below show an example of a provenance trace of a combine web processing service. The trace demonstrates an output chain including a WCS services, Normalisation services, and a combine function. The blue bubbles represent the chain of workflows within the web process combined to process the output. The final provenance traces are effectively visualised using boundary diagrams, to show workflows between, users, processing elements and services.

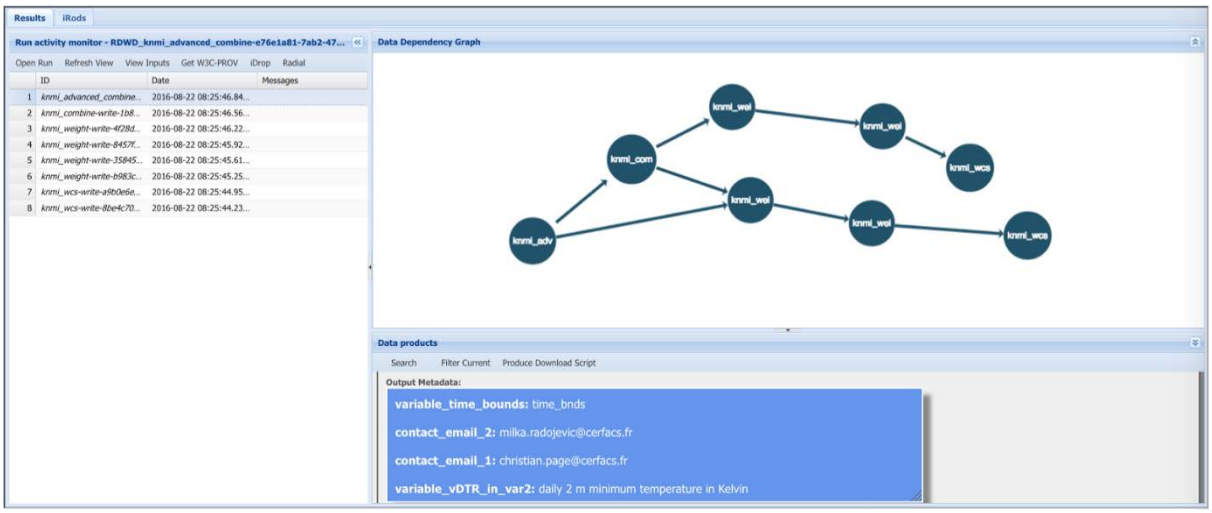

Figure 7 Provenance trace of Combine Web processing

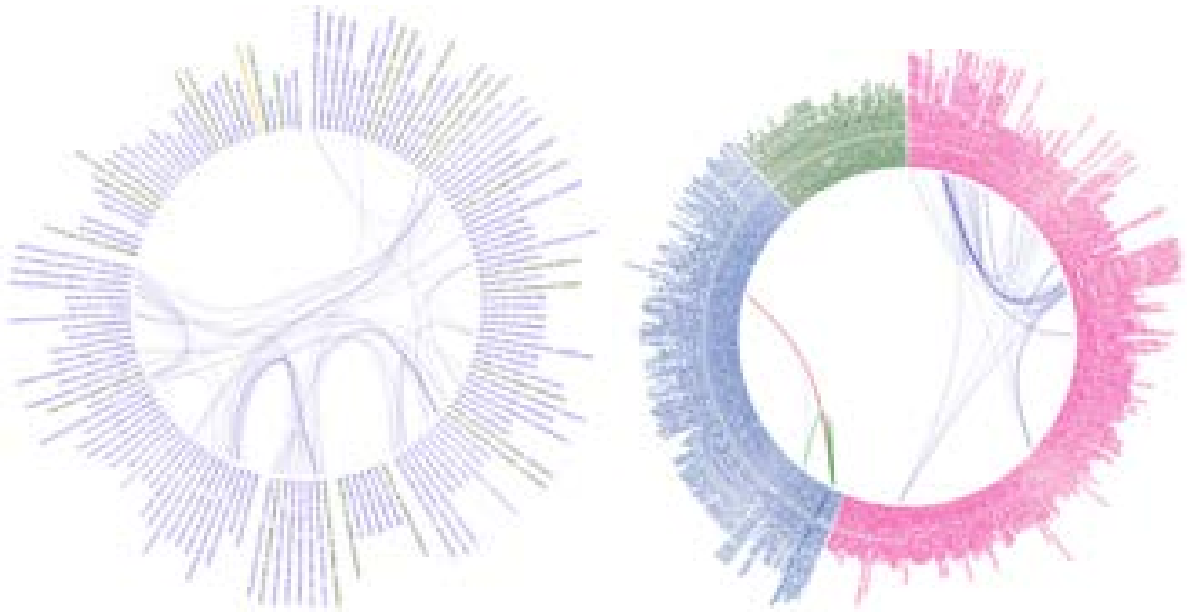

Figure 8 Boundary diagrams showing workflow between user, processing elements and services

Figure 8 Edges Bundles Visualisation showing the collaborative interactions between users, data and the execution of workflows or WPSs. The diagrams are obtained by searching for those executions that have produced data characterized by having a set of metadata with values within a certain range. The vertexes represent workflows or WPS executions and the

edges indicate the exchange and re-use of data. The images show two different vertex grouping, (a) by workflow or WPS name and (b) by user-name . Different colours indicate different users. The image (a) contains less edges, as result of requesting the logical conjunction of values-range for the same metadata set.

4. The CLIPC Indicator toolbox Caching system

4.1 Introduction

The objective of the CLIPC Indicator Toolbox Caching System (prototype), developed in the context of the CLIPC project, is to allow the CLIPC users to speedup the processing of highly time consuming tasks and the consequent retrieval and visualisation of the produced output with the aim to integrate a not-invasive component in the pre-existing WPS portal services.
In this perspective, the developed caching system acts as an intermediate layer between the user client, devoted to start the execution, and the WPS backend process able to produce the requested output (in the form of climate indicator). With the aim of avoiding the re-execution of the tasks already performed (on a per-user basis), the caching system represents a request interceptor able to redirect the user to the already produced output or, alternatively, to dispatch the incoming request to the underlying processing system.
The architecture modeled to support its functionalities, led to a general purpose module able to support different kind of back-end climate tools or framework; icclim[1] for impact indicators production, cdo[2] or nco[3] for climate data analysis and transformation or Ophidia[4][5] for data analytics on large volumes of climate data.

In the next chapters the caching system will be fully described starting from the requirements which drove the development. Then the architecture will be presented and the different software modules will be analyzed. Finally the integration steps needed to make the caching mechanism available in the CLIPC back-end processing system will be described.

4.2 Requirements

The Caching System design phase was driven by the preliminary definition of the requirements it had to satisfy: those requirements could be subdivided in functional and non-functional. Functional requirements refer to the behaviours (functions or services) of the system that support user goals, tasks or activities. Non-functional requirements include constraints, qualities, and properties or characteristics of the system that its stakeholders care about.

FR01: The caching system must act as a middle layer between the upper layer (as users' web interface) and the underlying layer (as the processing/analysis back-end components);
FR02: The caching system must manage two kind of scenarios: a cache hit and a cache miss;
FR03: The caching system must implement a data cache replacement policy.

NF01: Robustness: the caching system must be robust in terms of errors management;
NF02: Extensibility: the caching system must have the ability to easily incorporate new functionalities;
NF03: Performance: the caching system must serve the incoming request in near real-time;
NF04: Integration: the caching system must be easily integrated in the pre-existing back-end with no or minor modifications on the environment;
NF05: Resilience: the caching system must implement a fault tolerance mechanism; in case of errors the underlying process execution must proceed in a "no cache" modality.

4.3 Design and Modelling

As stated in the introduction, the objective of the developed software modules involved in the cache management represent a middle-layer between the client side and the back-end analysis system which has the task to produce the required output. It is based on the concepts of "cache hit" and "cache miss" as two distinct modalities to manage the incoming requests: in the former the caching system simulates to the client the execution of a task (without involving the computational layers) by replying with the requested outputs.

The following sequence diagram illustrates the caching system behaviour in a cache hit modality.

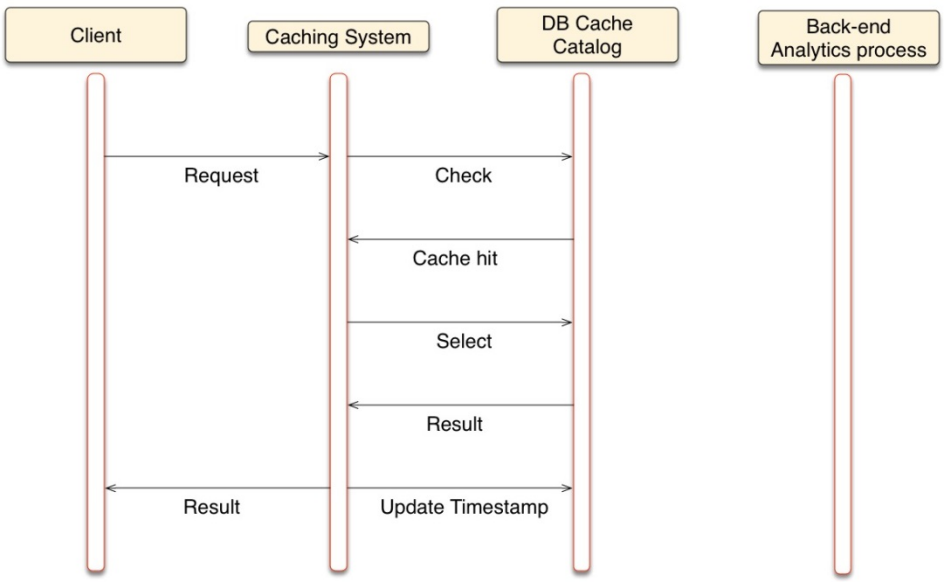

Figure 9 The caching system during a "cache hit"

It is worth noting that if a "cache hit" occurs no interactions with the analytics layer will take place as the needed outputs are already available in the user environment; the correspondent timestamp is also update in order to foster a replacement policy able to preserve storage space.

The “cache miss” modality, on the other hand, concerns the real execution of a new task: in this case the caching module contacts in the proper way the computational layers storing the produced outputs in order to serve in a “cache hit” way the next similar requests.

The following sequence diagram illustrates the caching system behaviour in a cache miss modality.

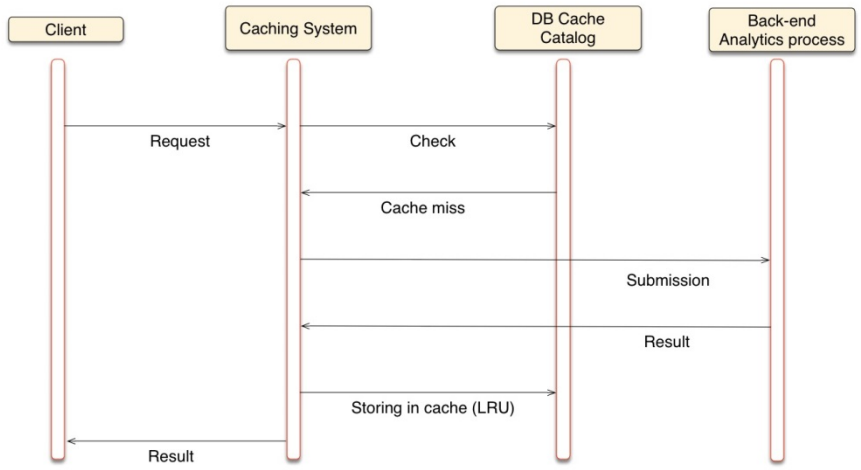

Figure 10 The caching system during a “cache miss”

As the previous figure shows, a “cache miss” involves the back-end computational layer submitting the user request and waiting for the results. A new entry is added in the cache catalog and a replacement policy is applied.

4.4 Architecture

From an architectural point of view, the caching system relies on two separate components:

- a caching system module able to store and retrieve data;
- a caching catalog to store all the needed information (resources path, metadata, timestamp, etc.) related to the users requests.

Moreover, the designed architecture has taken into account the pre-existing environment of the CLIP-C processing environment and toolbox. Incoming processing requests exploit a WPS[6] interface while the impact indicator analysis are coded in Python language. In order to minimize the effort needed to integrate the caching service in the back-end system, it has been developed using the Python language and the PyWPS[7] module has been used as Python implementation of the WPS specifications.

Concerning the DBMS acting as a caching catalog, able to collect all the information related to the processed requests, MySQL[8] has been used. Since the current processing environment relies on a PostgreSQL[9] Database, the caching catalog has been later moved to this DBMS. Few modifications have been required in order to carried out this movement since

the use of standard SQL speeded up this process and no query needed to be modified: the PostgreSQL Python drivers have been included and a PostgreSQL database table has been created to replace the MySQL one.

In the next figure is shown a client interaction with the caching system and the other back-end components.

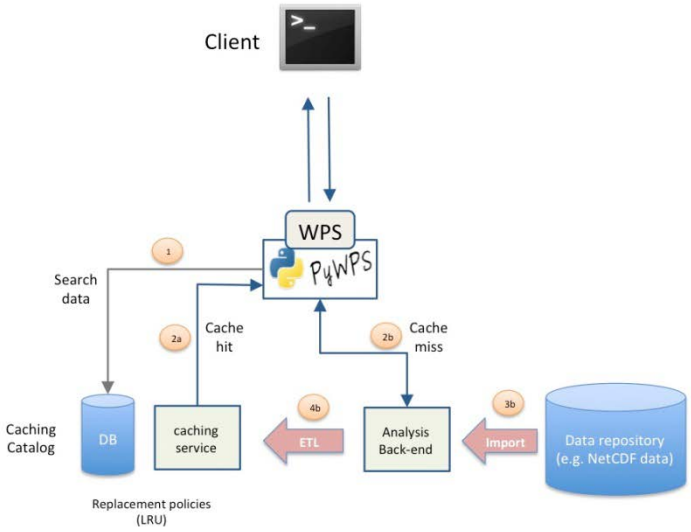

Figure 11 Interactions between a client request, the caching system and the back-end analysis sy

Specifically, the steps involved in this process are:

1. the caching system intercepts the WPS user request exploiting the PyWPS environment. It uses the input parameters as a searching key for querying the caching catalog;

2a. in a “cache hit” state the system are able to send back to the user the already available results;

2b. in a “cache miss” state the caching system triggers the back-end analysis processing in order to perform the needed computation: the request parameters are forwarded to the back-end system;

3b. if needed, input data are taken from the data repository;

4b. at the job completion, the produced results are sent back to the user. In addition the caching system uses the request parameters to create a new entry in the caching catalog enabling a “cache hit” for similar next incoming requests.

4.5 Cache Replacement Policies

Cache replacement policies represent algorithms or instructions used in software program able to preserve storage space (RAM or Hard disk space) in order to better manage the information stored improving the retrieval performance.

Many algorithms are available to implement the cache replacement policy[10][11][12]; the most efficient algorithm should be able to discard from the cache all the information which

will not be accessed for the longest time in the future. This algorithm, known as Belady algorithm[13], represent the optimal replacement policy but it cannot be implemented since it is not possible to predict when in the future a certain section of the cache will be accessed.

Other available solutions are:

- **LRU** - Least Recently Used: discards the least recent accessed information;
- **FIFO** - First In First Out: discards the oldest information stored regardless of how recently it was accessed;
- **LFU** - Least Frequently Used: discards the entries that are used least;
- **RR** - Random Replacement: randomly selects a candidate item to discard;

Among them, the LRU replacement policy has been chosen as the most appropriate and effective one; in this perspective a LRU policy has been implemented with the aim of discard the least used entries in the caching catalog. In order to extend the flexibility of the system, a configurable threshold has been used so only a limited set of entries will be stored in the caching catalog.

4.6 Development of the Caching System

As stated in the previous sections, the development phase of the caching system has been driven of easing the integration in the pre-existing back-end analysis engine of the CLIPC portal.

The development of the system concerned two different stages: the first was the definition and implementation of the system catalog able to store all the information related to the users requests for enabling a “cache hit” state, the second was the implementation of the Python software modules able to intercept the user requests, interact with the catalog and implement a cache hit or cache miss policy.
Regarding the database used as caching catalog, a single table database instance has been used to support the caching system need. Specifically it is formed by the following columns:

- *id*: unique identifier (primary key) of the related entry;
- *query*: used to store the input parameters which uniquely identify the input request;
- *result*: contains the output associated to the stored request;
- *last_request*: timestamp related to the last accessed date;
- *hit_count*: a counter representing how many times the associated result has been accessed. It could be useful for statistical purposes or to implement other kind of replacement policies;
- *status*: could be used for distinguishing between requests currently under processing and ready.

Concerning the Python module acting as caching service (specifically named *clipccache.py*), it contains the class and the functions able to manage the caching system and the interaction

with the other components (WPS environment, caching catalog and back-end analysis engine).
The following structures and functions have been developed:

- a class for storing the information related to the user request and for managing the connection with the caching catalog database. It contains also the value related to the LRU threshold.
- a class method which implements the searching phase. It queries the caching catalog searching for the entry correspondent to the user request parameters; in case of a cache hit, it updates the *last_request* field with the current timestamp;
- a class method which implements the ingestion of a new entry in the caching system catalog in case of a cache miss.

More in detail it is worth noting that:

- the parameters used to form the field which identifies the entry (the *query* field in the database table) are extracted from the user request;
- the order of the parameters in the users request does not affect the construction of the *query*;
- to simplify the storing of the *query* in the database and to reduce the occupied space, an hash of the input parameters set has been used;
- the *result* field is designed for accepting any kind of textual output; in order to make it compliant with the CLIPC processing environment, an OPeNDAP[14] url is stored as result of the performed computation.

4.7 Integration in the CLIPC back-end

As stated in a previous section, the CLIPC back-end processing system relies on a PostgreSQL DBMS while the first implementation of the caching system exploit a MySQL DBMS for its catalog.
Thanks to the standard SQL used for the implementation, few modifications have been performed in order to move the database catalog from a MySQL DBMS to PotgreSQL: the MySQL python drivers have been replaced by the PostgreSQL ones and a new database table has been created in the PostgreSQL DBMS.
In addition, in order to separate the users working space, the system has been modified for storing the caching information on a per-user basis. This prevent a user to access data outside from his own basket. Finally, a new routine has been implemented able to detect if an output file has been deleted. This new feature has been included in the system in order to properly manage any potential file deletion made in the user's output basket.

5. Conclusion and next steps

The use of Climate4Impact services and expanding on this framework enabled CLIPC to have a head start and a proven platform for development The current developed services are available on CLIPC and useable for end-users. Testing the caching system proved it can be a valuable addition to the CLIPC system for performance improvement.

The services can be further improved. At user consultations in June 2016 and at the final user consultation in Brussels (October 2016) lots of great suggestions for improvement were made. Below we suggest directions for further improvements as possible next steps:

Compare:

1) Improve styling capabilities
2) Improve automated legends by providing same legends at maps which are compared (if possible).

Combine:

1) Expand the Combine services with guiding creating sensible combinations (or issue a warning). A skeleton WPS (and online demonstrator with first check) is already made available for this purpose.
2) Expand on the number of combine and normalization functions
3) Expand to 3D combining – so with one combine action over the whole timerange?
4) Expand on combining more than two maps
5) Improve provenance metadata storage (in Mongo database)

Calculate indices:

1) Include the caching system in the operational environment
2) Implement caching security

MyCLIPC

1) Enable users to upload their own data
2) Enable automated cleanup of user space
3) Enable users to share data from MyCLIPC
4) Improved integration of data use from ESGF

The services can easily be expanded to provide new and improved services.

References

[1] icclim - http://icclim.readthedocs.io/en/latest/
[2] cdo - Climate Data Operators - https://code.zmaw.de/projects/cdo
[3] nco - NetCDF Operators - http://nco.sourceforge.net/
[4] Ophidia Framework - http://ophidia.cmcc.it/
[5] Ophidia Framework - S. Fiore, A. D'Anca, D. Elia, C. Palazzo, I. Foster, D. Williams, G. Aloisio, "Ophidia: A Full Software Stack for Scientific Data Analytics", proc. of the 2014 International Conference on High Performance Computing & Simulation (HPCS 2014), July 21 – 25, 2014, Bologna, Italy, pp. 343-350, ISBN: 978-1-4799-5311-0
[6] WPS - OGC 05-007r7, OpenGIS® Web Processing Service, Version: 1.0.0 at http://portal.opengeospatial.org/files/?artifact_id=24151
[7] PyWPS - http://pywps.org/
[8] MySQL - https://www.mysql.com/
[9] PostgreSQL - https://www.postgresql.org/
[10] Cache algorithms - https://en.wikipedia.org/wiki/Cache_algorithms
[11] Cache Replacement Policies - Prof. Mikko H. Lipasti, University of Wisconsin - Madison ECE/CS 752 Spring 2016 - https://ece752.ece.wisc.edu/lect11-cache-replacement.pdf
[12] Page Replacement Policies - http://web.cs.wpi.edu/~cs3013/a06/week5-pagereplace.pdf
[13] Belady, L. 1966. A study of replacement algorithms for virtual-storage computer. IBM Systems Journal 5, 2, 78–101.
[14] OPeNDAP - Open-source Project for a Network Data Access Protocol - https://www.opendap.org/
[15] D4.1 Toolbox Interface Specification - http://www.clipc.eu/content/content.php?htm=42
[16] ADAGUC website http://adaguc.knmi.nl