ESGF data usage metrics through the esgf-dashboard

Documentation

The esgf-dashboard component provides a distributed and scalable monitoring framework responsible for capturing usage metrics at the single site level and at the global ESGF level.

To achieve this purpose, a specific architecture has been defined to collect new kinds of data usage statistics about models, variables, datasets etc., besides the metrics contained in the logging information. For this reason, alongside the dashboard and the *access_logging* table of the Node Manager component, a new component of the ESGF distributed architecture was involved in the process: the SOLR search engine.

Figure 1 represents the current configuration and workflow of the monitoring framework.

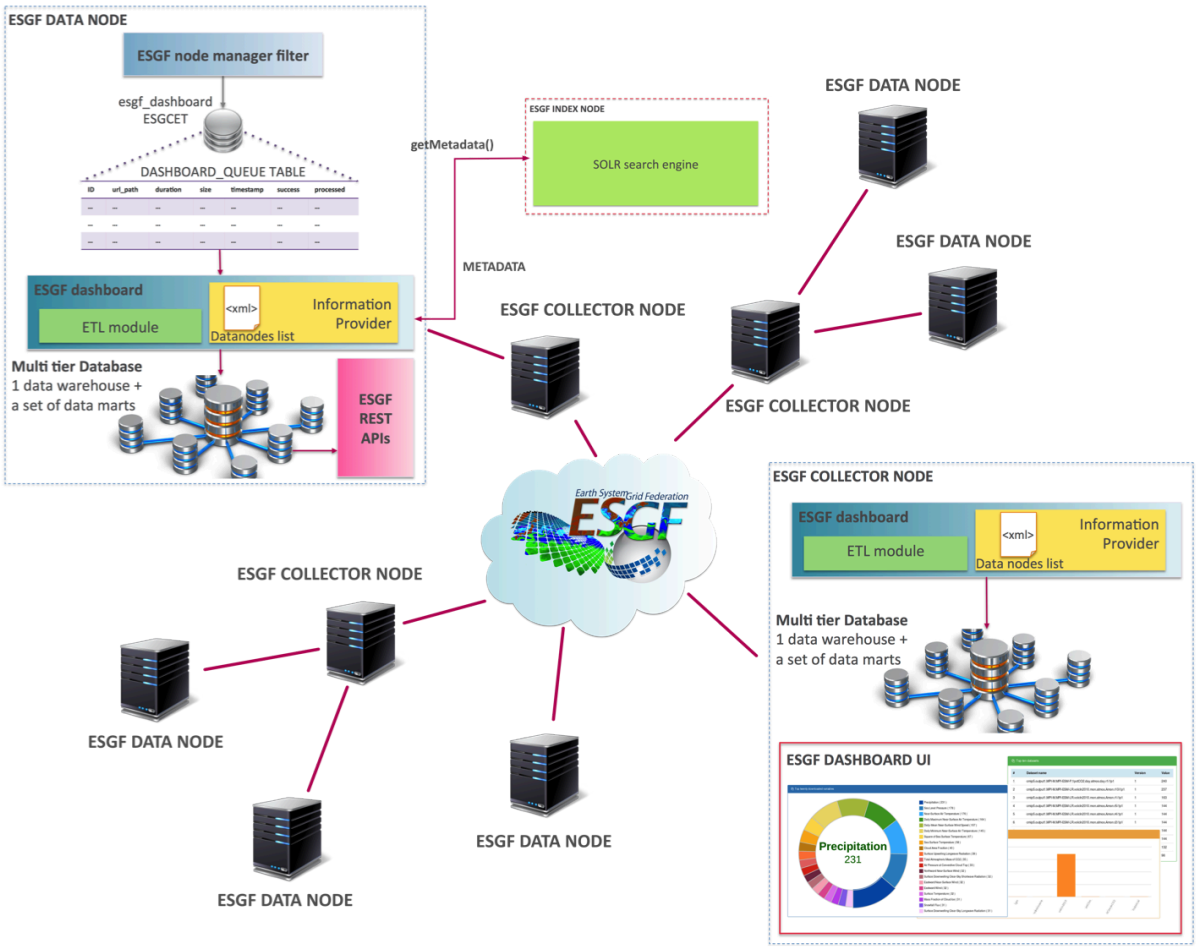

Fig. 1: The esgf-dashboard configuration and workflow

The esgf-dashboard relies on the Node Manager filter to gather the basic logging information from the *access_logging* table. Such data are collected by an ETL subroutine, normalized and stored into the *esgf_dashboard* namespace of the *esgcet* database.
Other information is collected by querying the SOLR search engine on the related Index Node, which is able to provide a wider set of metadata coupled with the downloaded datasets.

A logging table and the data warehouse system

An extension of the *esgcet* database was required to enable the collection of an extended set of statistics not only about logging information, but also project-specific metadata, clients geolocation , status of the published datasets.
To this purpose, the following groups of tables have been added to the original *esgf_dashboard* namespace:

1. a table called *dashboard_queue,* where the logging information is queued, waiting to be processed by the dashboard back-end (Figure 2);
2. a data warehouse system containing general cross-project information;
3. a data warehouse system for each project in ESGF (e.g. CMIP5, CMIP6, Obs4MIPs, CORDEX etc.);
4. a set of data marts as specific aggregated views on the data to support the ESGF dashboard user interface.

Each ESGF data node will expose the same database schema and the dashboard back-end will take care of properly feeding the system with the data coming from local sources as well as additional metadata retrieved from the Index Node.

The dashboard_queue table

The *dashboard_queue* table contains information about the downloads carried out by the users. The figure below shows a complete description of the table.

```
CREATE TABLE dashboard_queue (
    id integer NOT NULL,                          -- unique id
    url_path character varying NOT NULL,          -- path of the downloaded file
    remote_addr character varying NOT NULL,       -- user ip address
    user_id_hash character varying,               -- hash code of the user id
    user_idp character varying,                   -- user identity provider
    service_type character varying,               -- download service type
    success boolean,                              -- outcome of the download operation
    duration double precision,                    -- duration of the download operation
    size bigint DEFAULT (-1),                     -- file dimensions
    "timestamp" double precision NOT NULL,        -- download time instant
    processed smallint DEFAULT 0 NOT NULL         -- dashboard flag
);
```

Fig. 2: dashboard_queue schema

The fields can be grouped in three main categories:

- user information (remote address, user id hash and IdP node);
- file information (path to the file and its size);
- download information (outcome, duration, time instant, kind of download service).

The last 'processed' field is used by the back-end to mark the entries that have been analysed.
The table is the starting point for the ETL (Extraction – Transformation – Loading) process undertaken by the back-end: some of the information will be normalized; some other will be used to contact the Index Node and retrieve the related metadata. At the end, all these data will be ingested into the data warehouse system.

Cross-project and project-specific data warehouse

The cross-project data warehouse contains general information about the data usage statistics without taking into account any project-specific details.
The different colors on the diagram in Figure 3 refer to the data sources from which information is retrieved.

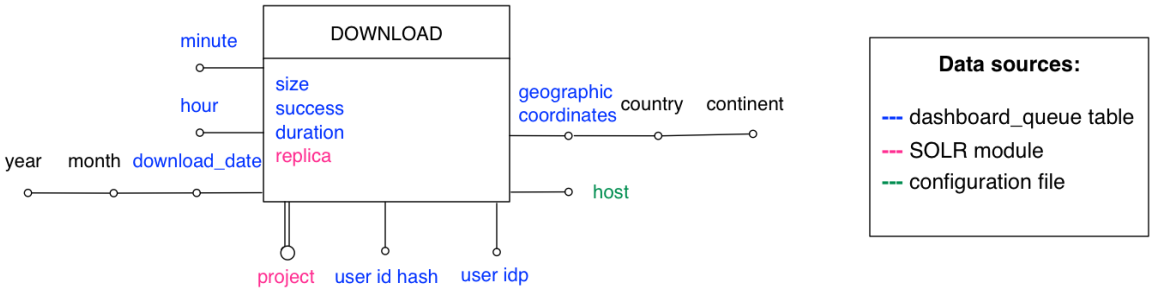

Fig. 3: Cross-project Dimensional Fact Model (DFM)

The system will keep track of the downloads through a set of relevant measures and dimensions of analysis.
More specifically, the measured quantities are:

- the download size;
- the operation outcome;
- the duration;
- if the downloaded file is a replica of the original one or not.

These quantities can be measured from different perspectives, over time, by project, by user, by identity provider, by host or spread out on a global map, in order to have a more accurate overview of what this information means.
Figure 4 is related to the snowflake schema of the previous DFM, where facts, measures and dimensions are translated into tables connected by foreign keys.

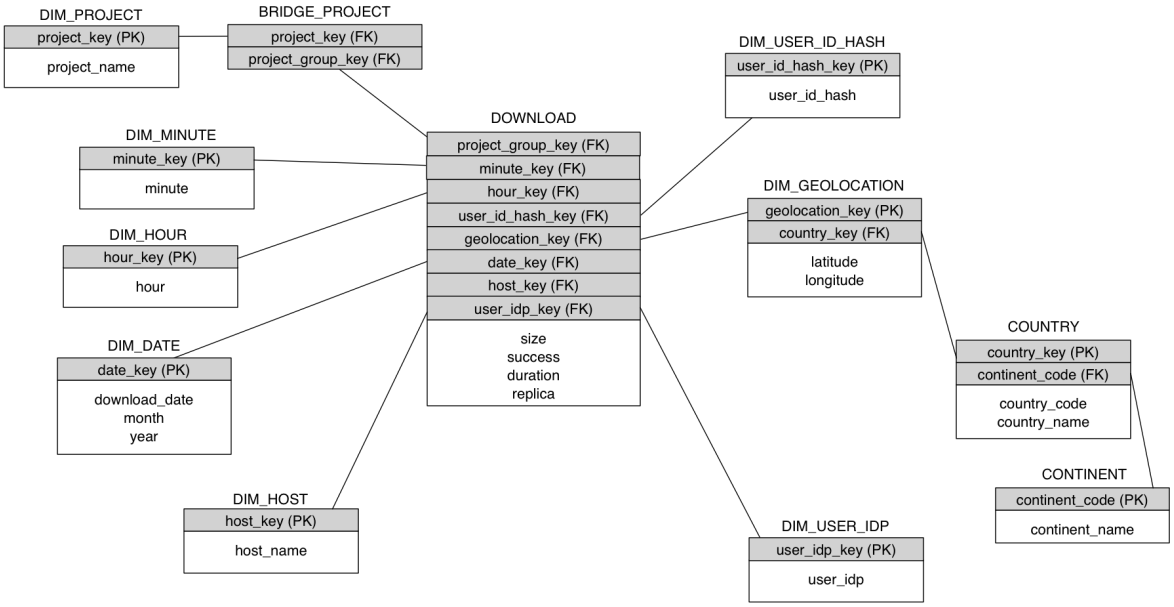

Fig. 4: Cross-project snowflake schema

Data marts are aggregated views of the entire system focusing on some of the dimensions available and providing an easy access to frequently needed data in order to improve the response time of the web front-end.
Figures 5 and 6 present two data marts of the cross-project data warehouse.

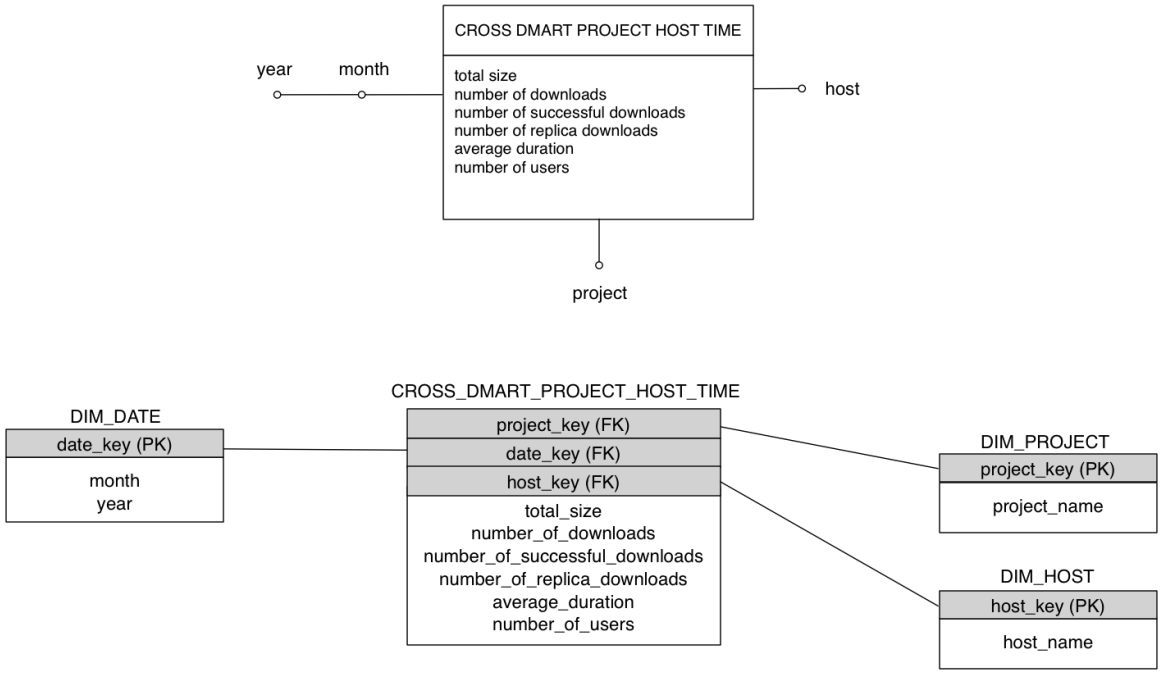

Fig. 5: Cross-project data mart (correlation between time, host and project dimensions)

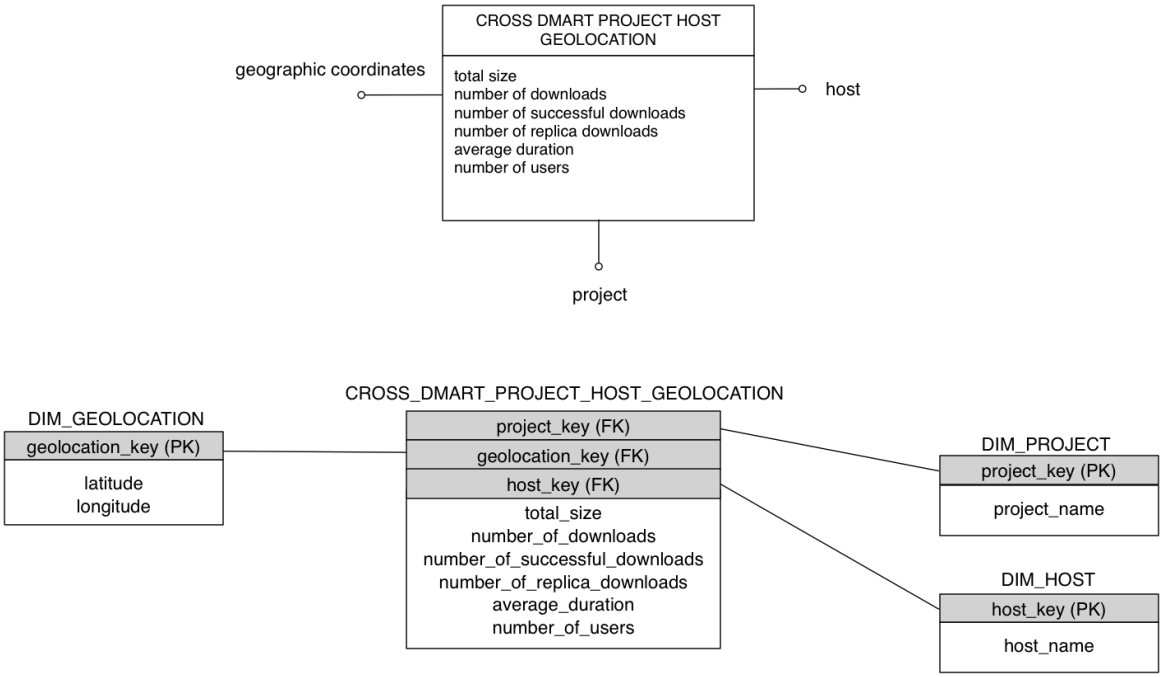

Fig. 6: Cross-project data mart (correlation between geolocation, host and project dimensions)

In both cases, the metrics resulting from the aggregation process are:

- total volume of downloads;
- total number of downloads;
- total number of successful downloads;
- total number of downloads related to a replicated file;
- average duration of the download time;
- total number of users that performed a download.

The two data marts represent a geographical and a temporal perspective on the metrics analysed by project and host. In practical terms, they support for instance the number of users distributed on a map through proportional markers or a multiple time series about the number of downloads per project.

A similar approach has been adopted to generate the data marts related to the project-specific section.

The esgf-dashboard architecture in the small

The architecture of the *esgf-dashboard* module is based on a main program and two sub-routines that are responsible for retrieving metadata from SOLR and aggregate the data into the storage database. The first sub-routine queries the *dashboard_queue* table, retrieves the URLs, then queries the SOLR and stores the data into specific tables of the database. The second subroutine aggregates the data and stores them into specific tables (data mart tables). The aggregation is made by following an upgrade at each given period. A

registry table is used to store the last timestamp in which an update is made by the dashboard.
The collector node is equipped with a third sub-routine responsible for gathering the data from the different data nodes. A configuration file contains the information about the data nodes to be queried in order to collect the federated statistics.

In terms of the dashboard itself, the ESGF federation is based on a hierarchical protocol, with leaf and collector nodes (Figure 7). The former are the lower-level nodes, the latter are involved to gather all the information about the related leaf nodes. The leaf nodes represent the data nodes where the NetCDF files are downloaded by users from the scientific community. All statistical information on the resources provided to users is contained in a data warehouse. Then, from this amount of data, extractions and combinations of information are performed based on some criteria, and the results are stored in the data marts.

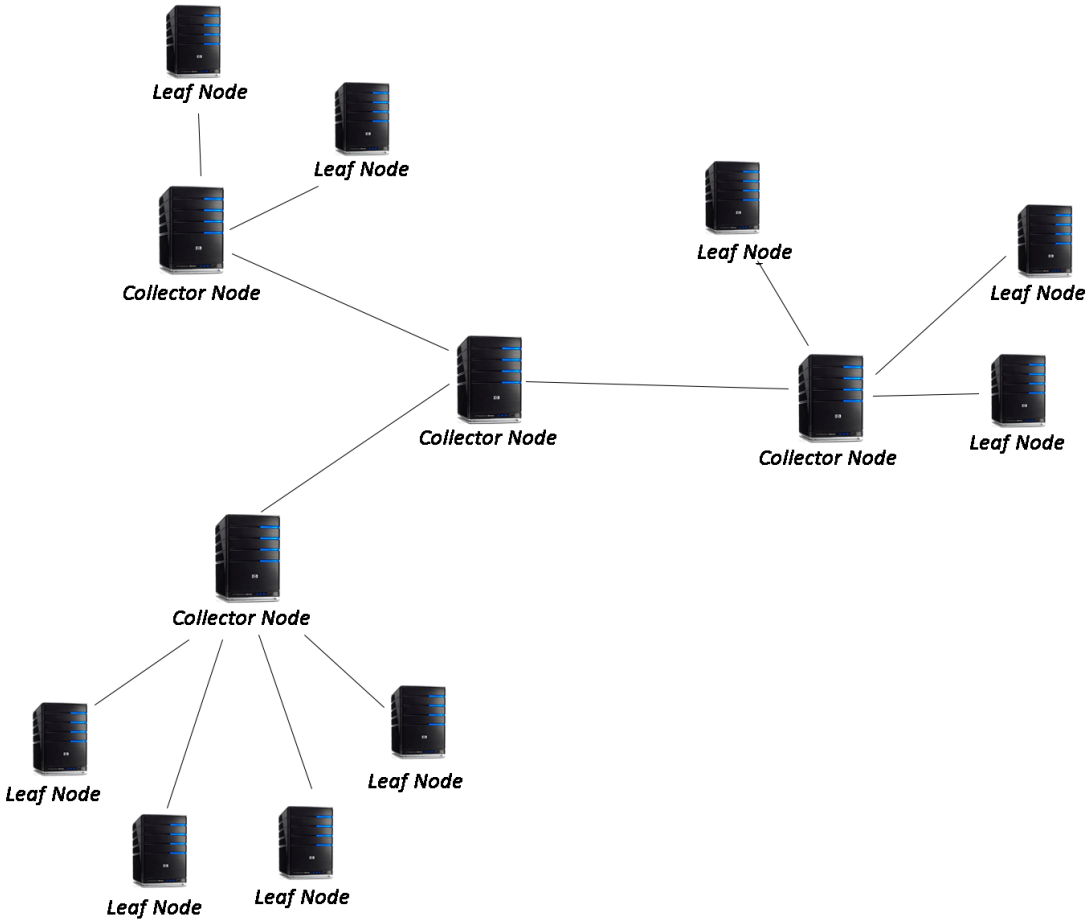

Fig. 7: Hierarchical protocol.

The leaf nodes are able to share their information with other nodes, through the implementation of the REST APIs service (the *esgf-stats-api* module). When a node receives a request via its REST interface, it provides an answer through the following simple steps:

- analysis of the request (checking what data to collect and retrieving them from the proper data mart);
- recovery of data;
- serialization of data in an XML file and transfer to the node that made the request.

The collector node architecture

Collector nodes have a more complex structure because, in addition to making their information available to the collector through the REST APIs, they are in charge of querying the leaf nodes associated with them.
The collector node is composed of:

- esgf-stats-api service;
- data warehouses and data marts;
- XML configuration file;
- federation component.

At the moment, there is only one instance of collector node, located at the CMCC Supercomputing Center, while all the remaining nodes act as leaf nodes and provide their statistics to the CMCC collector node.

Considering the architecture of the collector module (Figure 8), the daemon is a process that, once in execution, remains in background and, at regular intervals, launches the federation process. Other components represent the heart of the architecture and contain instructions for the management of the entire federation process. The first operation is the parsing of the configuration file in order to know all the leaf nodes. Then, the query generator prepares a set of queries for each leaf node. The controller runs the queries on each leaf node through REST APIs, and the receiver receives the query results from the leaf nodes in XML format, parses them and performs data ingestion in the data marts of the collector node. Finally, it updates a timestamp in the configuration file.

Following, a list of the esgf-stats-api urls available on the data nodes:

Cross-project

```
http(s)://<host>:<port>/esgf-stats-api/cross-project/stats-by-time/xml
http(s)://<host>:<port>/esgf-stats-api/cross-project/stats-by-space/xml
```

Project-specific

```
http(s)://<host>:<port>/esgf-stats-api/obs4mips/stats-by-space/xml
http(s)://<host>:<port>/esgf-stats-api/obs4mips/stats-by-dataset/xml
http(s)://<host>:<port>/esgf-stats-api/obs4mips/stats-by-realm/xml
http(s)://<host>:<port>/esgf-stats-api/obs4mips/stats-by-source/xml
http(s)://<host>:<port>/esgf-stats-api/obs4mips/stats-by-variable/xml
```

```
http(s)://<host>:<port>/esgf-stats-api/cmip5/stats-by-space/xml
http(s)://<host>:<port>/esgf-stats-api/cmip5/stats-by-dataset/xml
http(s)://<host>:<port>/esgf-stats-api/cmip5/stats-by-experiment/xml
http(s)://<host>:<port>/esgf-stats-api/cmip5/stats-by-model/xml
http(s)://<host>:<port>/esgf-stats-api/cmip5/stats-by-variable/xml

http(s)://<host>:<port>/esgf-stats-api/cmip6/stats-by-space/xml
http(s)://<host>:<port>/esgf-stats-api/cmip6/stats-by-dataset/xml
http(s)://<host>:<port>/esgf-stats-api/cmip6/stats-by-experiment-id/xml
http(s)://<host>:<port>/esgf-stats-api/cmip6/stats-by-source-id/xml
http(s)://<host>:<port>/esgf-stats-api/cmip6/stats-by-variable/xml
```

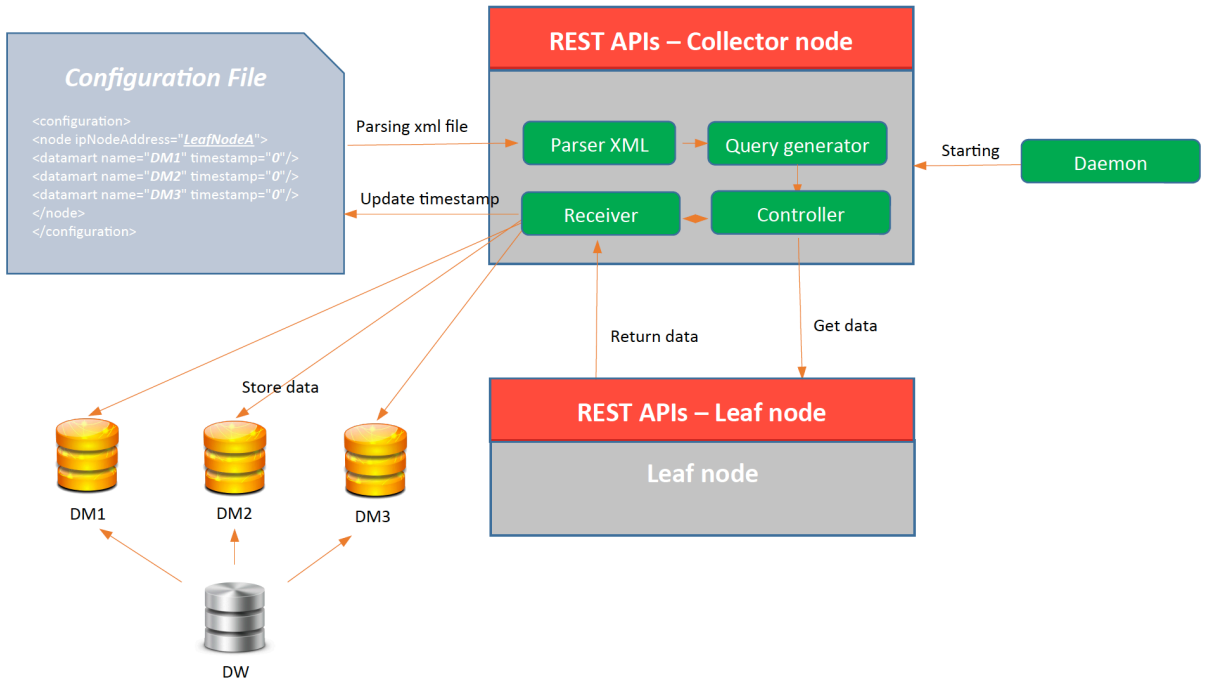

Fig. 8: Architecture of the collector node.

ESGF dashboard-ui

The main objective of the dashboard-ui (dashboard user interface) is to provide an integrated and usable way to visualize statistics at single site and ESGF federation level. Different categories of statistics, concerning the data usage and the clients activities and distribution, are provided and shown through a set of simple and attractive graphical widgets (like charts, maps, tables and so on), giving the user a comprehensive view to access the data aggregated by the dashboard back-end.

Data visualization is often a difficult task for administrators and developers, as users need quick and simple access to lots of data and information.

A dashboard itself is a set of graphical components such as menu, charts, tables etc., showing information like statistics, analytics, schedules and much more. The key challenge of such a concept is to communicate the most important information in a straightforward way and allow users to go through specific details at the same time.
The dashboard-ui can be thought of as a control panel through which the user can access the different metrics/statistics at node and ESGF federation level.
All the perspectives through which the information can be analysed are grouped into several menus and submenus, easily accessible from the left side of the GUI.
Being a centralized web-application, deployed on the collector node at CMCC, further requirements or customization requests should be previously discussed with the community partners and the GUI administrators.
Here is a complete description of the different sections provided by the web application:

- Published data;
- Cross-project statistics;
- Project-specific statistics;
- Federated data archive;
- IS-ENES2 KPIs.

The first section, in Figure 9, provides a summary view of the total amount of data published on the ESGF federated archive in terms of total number of datasets as well as distinct and replica datasets with the related total data volume (in Terabytes); moreover, it also displays the number of datasets, along with distinct and replica datasets and their related data volume for CMIP5, CMIP6, INPUT4MIPs, Obs4MIPs and CORDEX projects (other projects can be added). Also, it is possible to select a specific data node and obtain information on the total number of published dataset and the total amount of data (in Terabytes) for that node.

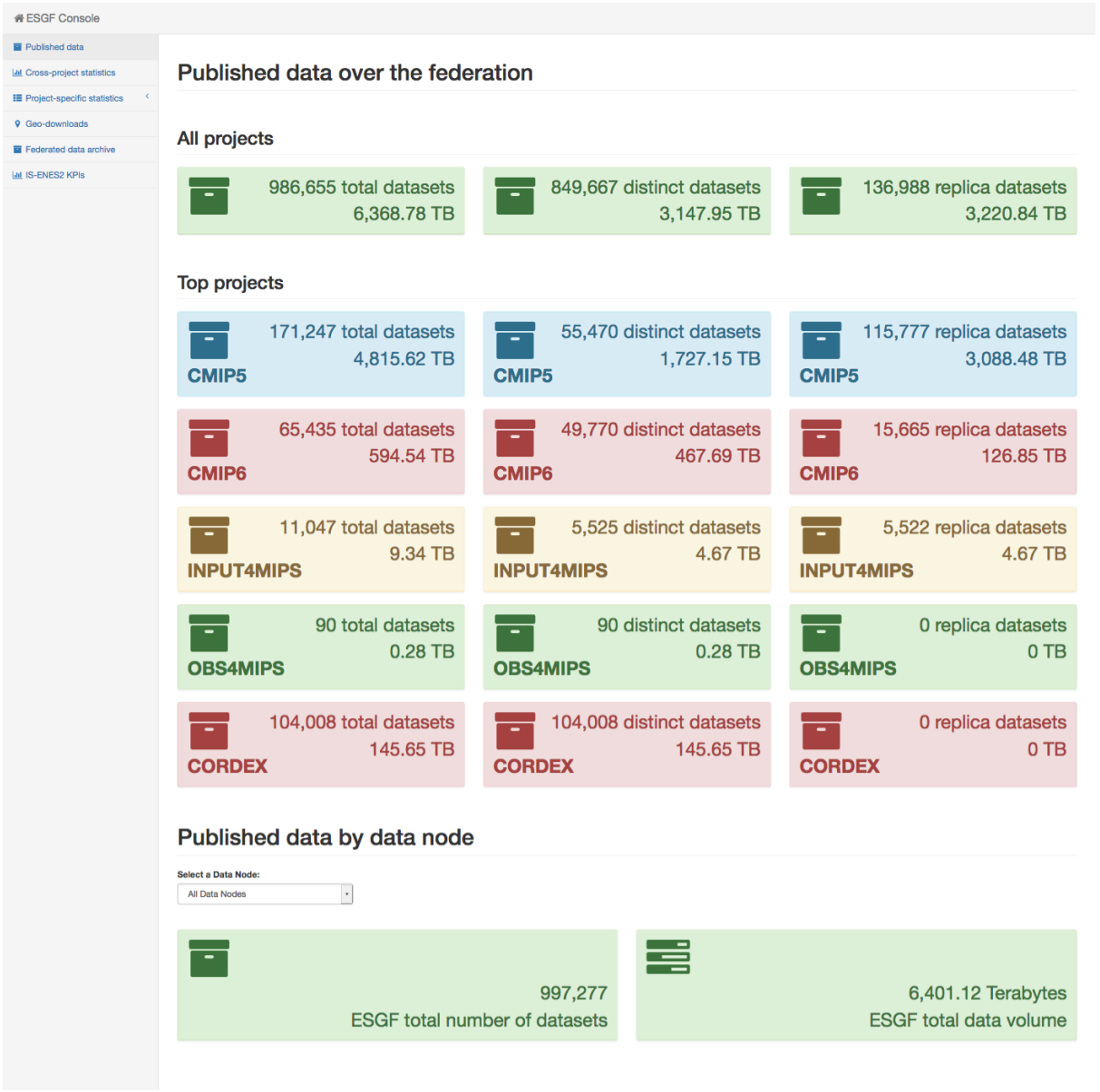

Fig. 9: Published data over federation

The Cross-project statistics section (Figure 10) incorporates a set of views related to the cross-project statistics, which show the same data download activities but are accessed through different perspectives. Different metrics such as the number of downloads, the number of successful downloads and the downloaded data in terms of Gigabytes, are visualized in several widgets grouping the information by time, host and project. For each widget, there is the possibility to download a CSV file with the data to be used for reports, documentation or further analysis.

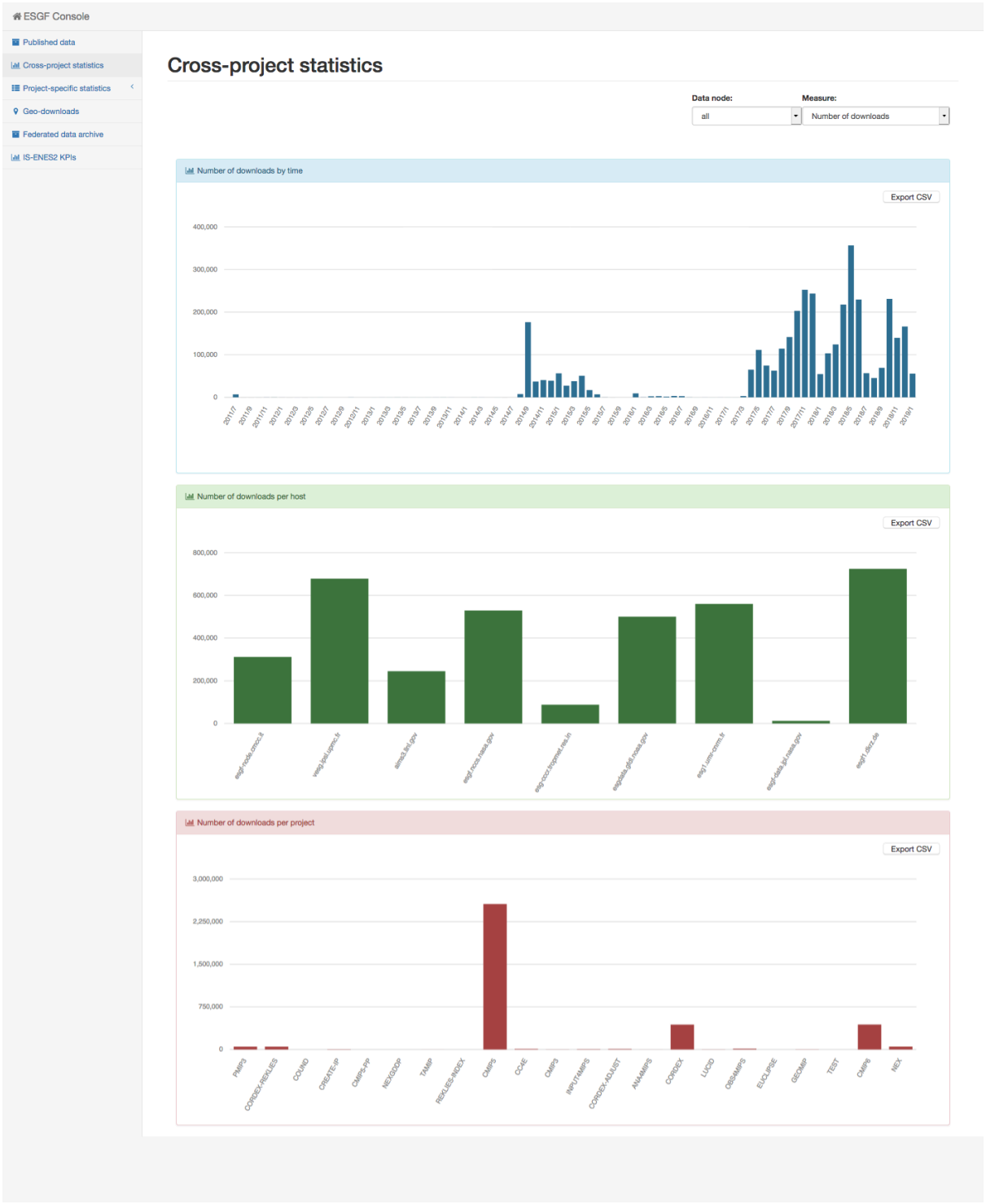

Fig. 10: Cross-project statistics by time, host and project

The Project-specific section gives a complete overview of different projects, like CMIP5, Obs4MIPs, CMIP6 and, in particular, it provides a very useful set of widgets representing specific information, such as:

- Top ten datasets;
- Top ten experiments/sources;

- Top ten variables;
- Number of downloads, number of successful downloads and downloaded data (in Gigabytes) for the top 20 downloaded variables;
- Number of downloads, number of successful downloads and downloaded data (in Gigabytes) by realm, source, model, experiment.

Figures 11 (a) and (b) show the previous views in the respective widgets.

Quite similar widgets are available for CMIP6 and Obs4MIPs projects.

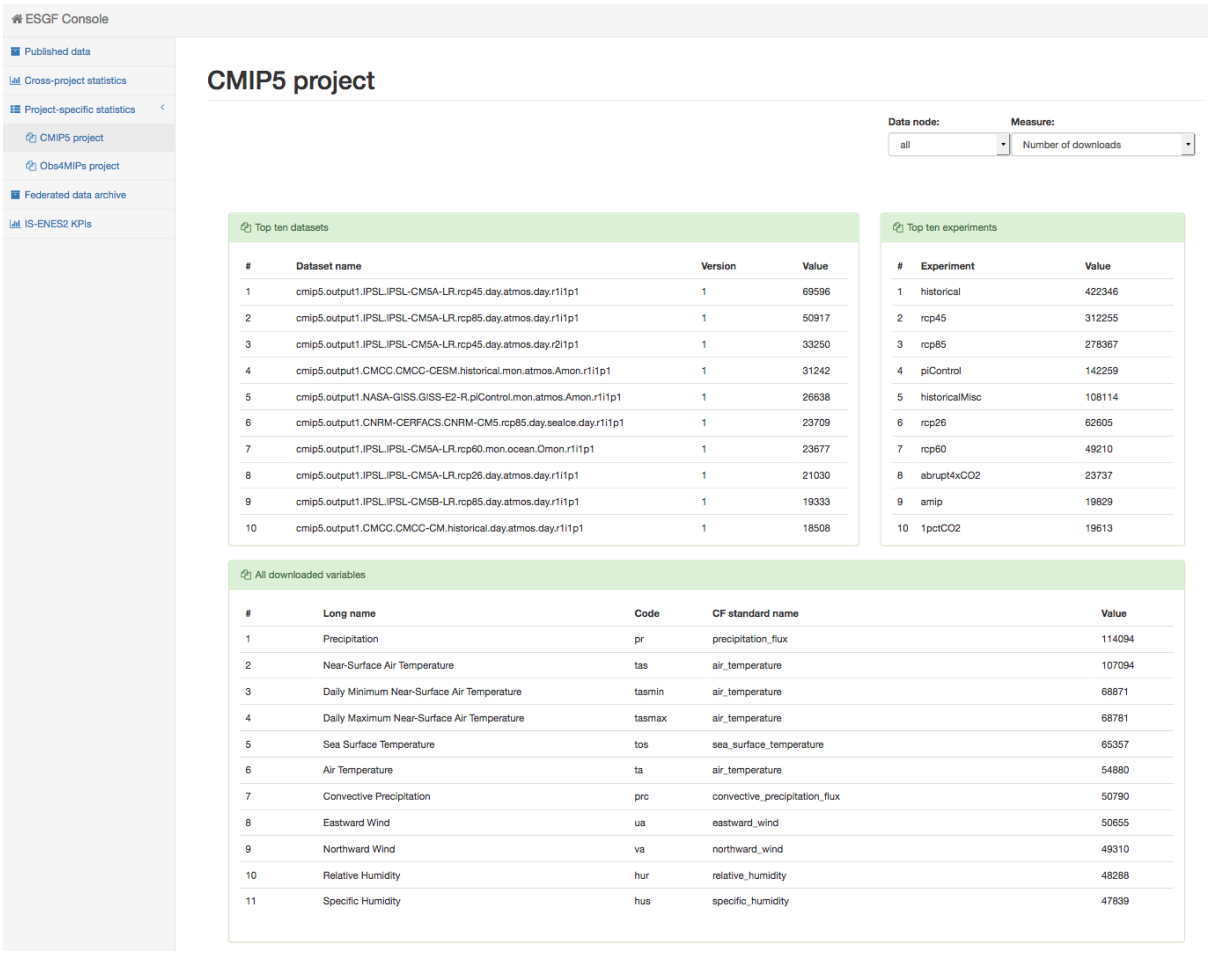

Fig. 11 (a): Project-specific statistics

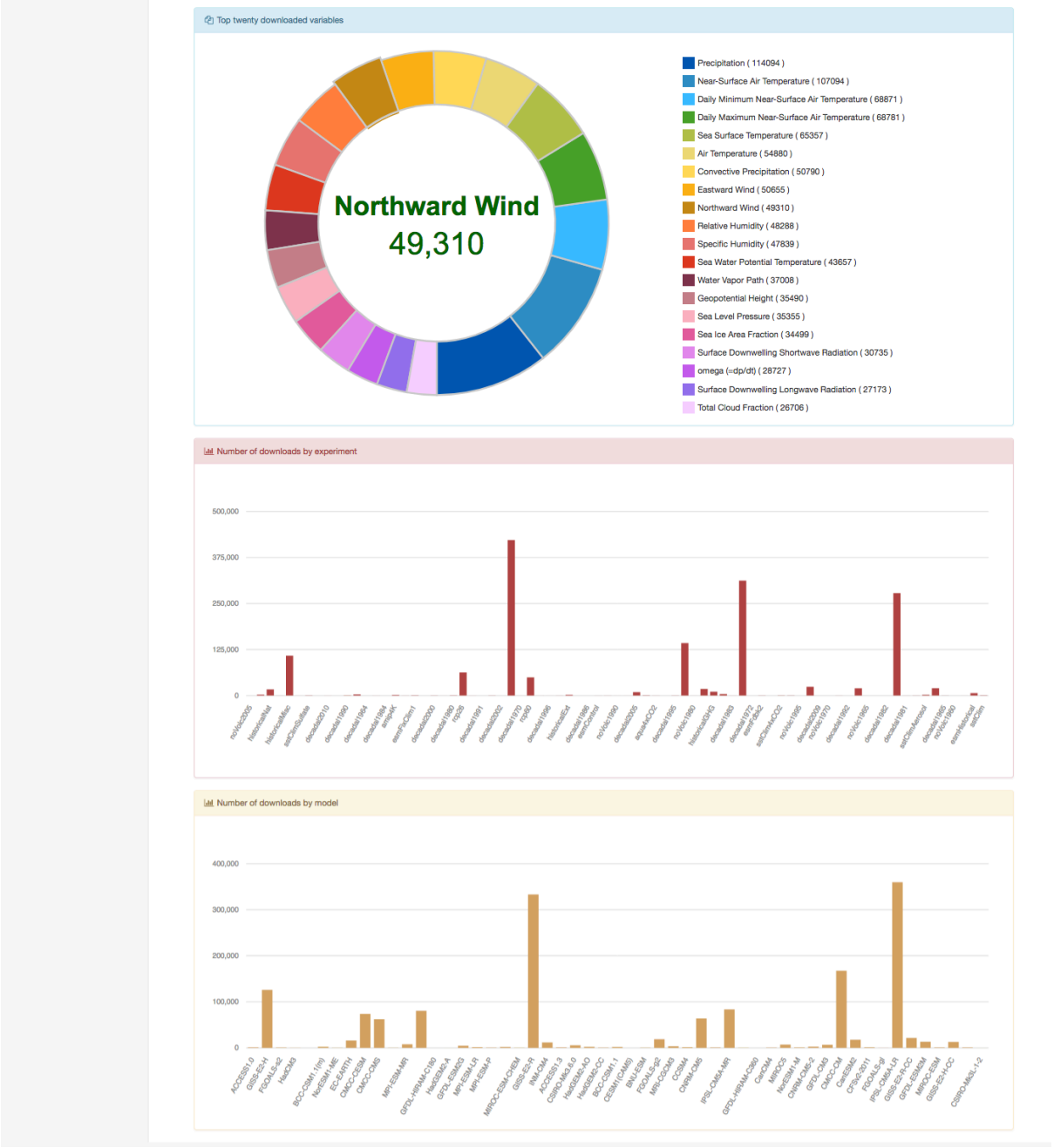

Fig. 11 (b): Project-specific statistics

The Geo-downlaods (Figure 12) provides a practical way to obtain information on the map distribution of the number of download per continent and group them by countries on different tables.

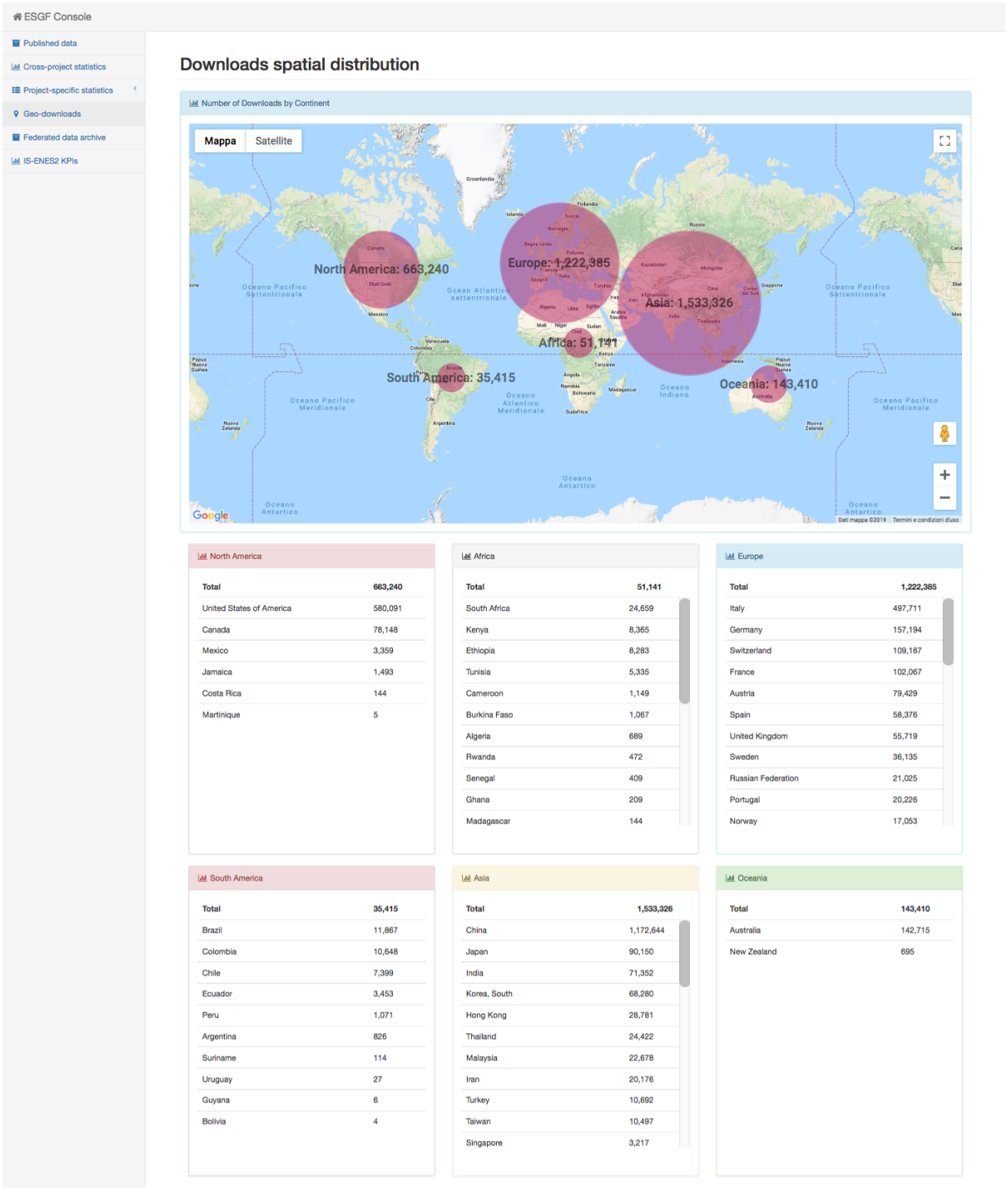

Fig. 12: Geo-downloads section

The Federated Statistics section (Figure 13) shows information about the amount of data available for download on the entire federation. In particular, at the moment two tables list the Models and the Modelling Institutes for the CMIP5 project.

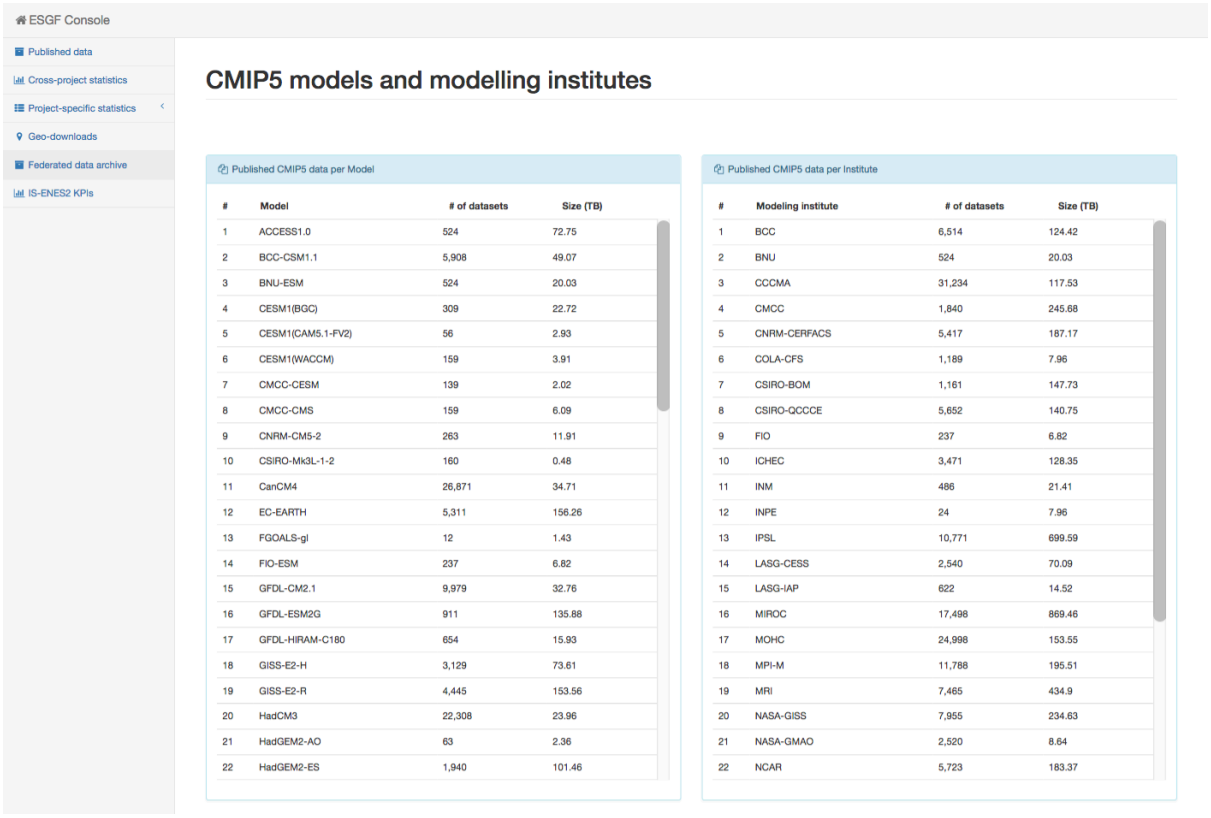

Fig. 13: Federated data archive

The last section, IS-ENES KPIs (Figure 14), shows the results of the IS-ENES2 project in terms of Successfully downloaded files, Number of active users and Downloaded data volumes, grouped by EU and non-EU users.

Fig. 14: IS-ENES2 KPIs section

Configuration

Checking the *esgf-dashboard* and *esgf-stats-api* status (from 2.6.8b-master version on)

For esgf-dashboard

Check if the dashboard is running, by executing the following command

```
ps auxwww | grep esgf-dashboard-ip
```

If the process isn't running, restart it by executing, as root,

```
/usr/local/esgf-dashboard-ip/bin/ip.service restart
```

Check if the *public.esgf_migrate_version* table is updated to the latest version (version = 11)

```
select version from public.esgf_migrate_version where repository_id='ESGF Dashboard schema migration';
```

Download a dataset from the catalog:

```
https://localhost/thredds/catalog/esgcet/catalog.htm
```

Check if the entry has been added to the *dashboard_queue* table:

```
psql -d esgcet -U dbsuper;
set search_path=esgf_dashboard;
select * from dashboard_queue;
```

For esgf-stats-api

Contact the following urls and check weather they reply correctly (insert the proper *host* and *port* for your node):

```
http(s)://<host>:<port>/esgf-stats-api/cross-project/stats-by-time/xml
http(s)://<host>:<port>/esgf-stats-api/cross-project/stats-by-space/xml
```

CMCC should be able to reach the stats-api url to get the statistics on each node and aggregate them at federation level.
Make sure that the stats API can be accessible from outside and send the url to the stats team.

Configuring the esgf-dashboard to exclude Synda downloads

Starting from the 2.6.8-master release, a new feature of the esgf-dashboard will provide the possibility to distinguish regular from replication downloads, thus providing more accurate statistics on the data usage.

To configure the dashboard, execute the following instructions:

- After the installation of the 2.6.8-master (or later) release, stop the dashboard process:

  ```
  /usr/local/esgf-dashboard-ip/bin/ip.service stop
  ```

- The node administrator can then insert the openID of the users who are supposed to be distinguished from the downloads statistics into the

  ```
  /usr/local/esgf-dashboard-ip/etc/notauthorized_openID
  ```

  configuration file. So,

  ```
  <notauthorized_openID 1>
  <notauthorized_openID 2>
  <notauthorized_openID 3>
  …
  ```

  has to be updated with the value of the *user_id* field in the *esgf_node_manager.access_logging* table (remove angled braces).

  For example:
  ```
  https://vesgint-idx.ipsl.upmc.fr/esgf-idp/openid/synda_ipsl
  https://esgf-fedtest.dkrz.de/esgf-idp/openid/synda_dkrz
  https://pcmdi11.llnl.gov/esgf-idp/openid/synda_llnl
  ...
  ```
- Finally, start the dashboard

  ```
  /usr/local/esgf-dashboard-ip/bin/ip.service start
  ```

The new downloads will be selected based on the users entered in the configuration file.