DISTRI.C.O. an operational system for post-processing and distribution of numerical model outputs.

Enrico Fucile, Centro Nazionale di Meteorologia e Climatologia Aeronautica Aeroporto De Bernardi – Pratica di Mare (Roma) fucile@meteoam.it

The increasing volume of numerical models outputs gives rise to the need of operational systems capable of elaborating and managing large amounts of data. DISTRICO is the system realized at the Centro Nazionale di Meteorologia e Climatologia Aeronautica (CNMCA) for post-processing and dissemination of data in grib format that fulfils the need of managing large amounts of data generated by numerical models. The main ideas of its design are two: organizing data into a relational database providing an easy and quick access through Structured Query Language, and using a scripting language into which C/C++/Fortran code or libraries can be easily embedded allowing reuse of reliable software.

PHP was chosen as scripting language for its extensibility capabilities and for the easiness of the access it provides to database systems. Furthermore it was especially useful for the realization of the web interface providing management and control of DISTRICO data and tasks.

Although it was initially designed as the central system for numerical post-processing and dissemination it is operationally used at CNMCA to manage and produce all the analysis and forecast charts.

1 System description

DISTRICO is a system mainly designed for management of model outputs in grib format. In its first requirements there is the transformation of grib fields from one grid to another as needed by the end users that in many cases require fields on geographical coordinates from limited area models or local models working on rotated coordinates. Moreover the large amount of model outputs requires an optimization of the whole production cycle that must implement efficient interpolation algorithms and a smart distribution model. At this aim the system is designed with an activities model at its foundation that is described in full details in section 4.

Since a good data design is basic for maintenance and functionality of a system dealing with such data volumes, we choose to start our project designing a relational database containing both the model outputs in grib format and the information about users and fields that they require. The details of this database are shown in section 2.

As we were implementing the first system facilities it became quickly evident the need of a simplified access to be provided both to the end users and for developing and maintenance proposes. We choose PHP as the scripting language to develop the core of the DISTRICO system libraries so that we can provide easy access to the system facilities and we were able to build with few effort a web interface for administration and control of the whole system. A description of the PHP module can be found in section 3 and the web interface is introduced in section 5.

In fig. 1 a schema of DISTRICO is drawn. Its core is made by two basic elements: the relational database working as a central repository and the PHP module providing access to the system facilities. All tasks are carried on following an activities model (see sec. 4) and can be grouped into few main categories:

- GRIB loading from the file systems of the registered host machines to the database
- Elaboration of the grib files from the database to produce the fields required by the users (mainly interpolation) and in the recently upgraded version the meteorological plots
- GRIB distribution to the hosts designated by the user for the fields required.

All those activities and the Web interface completely rely on the PHP-module both to access data from the central repository and to process them before being inserted back into the database.

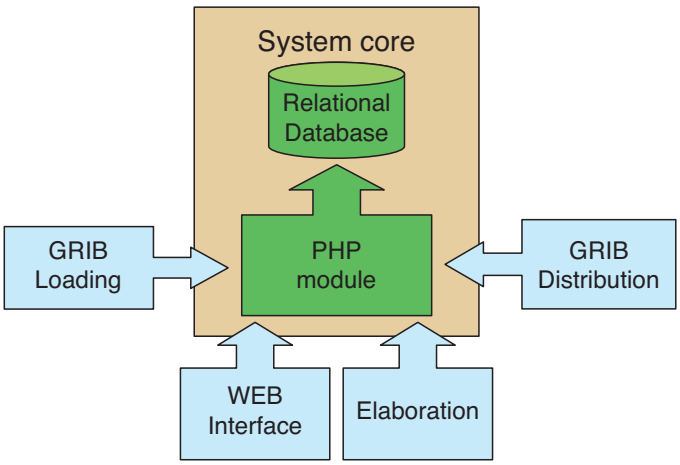

Fig. 1 DISTRICO schema. System core is made by relational database and a PHP module. Activities (loading, distribution and elaboration) and Web Interface access database through PHP module.

2 Relational Database

A system made to manage large volume of data needs a well designed data structure at its base. Thus the first step in building DISTRICO was the data design and the choice between relational database or a file system for the data repository. At a first sight it seams better a file system because it provides easy and fast access to data. Nevertheless analysing thoroughly the subject it results that to give a smart data access as it is needed when we work with model outputs a client-server application must be developed. Such an application should implement some kind of syntax to get/put data from/to the repository and should provide remote access to this one. We remark that a data server application is already available as integrating part of a relational database as the engine of the RDBMS (relational database management system). Thus we think that it is less expensive in time choosing a client-server application already tested as the one embedded into an RDBMS.

Moreover choosing a relational database has other advantages because Structured Query Language (SQL) that is used to access data through a RDBMS is an high level language very useful for complex data structures as those coming from the outputs of different models and the design of data structure itself is more rigorous and less error prone in respect to a free style file system. The data model implemented into DISTRICO is made as to separate in different tables data changing slowly in time from data that change every day. The idea is that when a grib enters into the database all its information is stored into some description tables and the grib file is stored into a separate table with a reference to its description and the date of run. Thus we have tables containing originating centre, process id, parameter id, etc. for each of the grib files ever entered into the system and the grib object itself is stored into a binary large object (BLOB) space into a different table with external references to its descriptive information. Static tables grow only if new grib files unknown to the system are loaded, whereas the table containing the grib object grows daily if older data are not deleted. New grib data are automatically classified and inserted into the database by the system without any manual operation. This database design complies normalization standards and this yields to high performances when getting and putting data.

3 PHP module

In order to provide easy access for users and to speed up development it was chosen to use a scripting language and to expose through it all the system features such as functions to get data from the database, coding and decoding tools and all interpolation and mathematical routines.

For various reasons we have chosen PHP as the scripting language for DISTRICO. Indeed it provides a lot of useful extensions ranging from the DB connectivity modules to graphical, network, and file system modules. Moreover it is easy to extend with C and Fortran code that was one of our needs because we had legacy code (mainly C) for decoding and encoding grib data and for doing interpolation and other operations. The embedding of C routines into the PHP interpreter has two advantages first provides the functions directly available as native calls from the scripting language and furthermore it is an optimization that must be done when dealing with interpolation of grib data because long loops over thousand of points can't be carried on from a scripting language without loss of performance.

PHP has other advantages as a scripting language for developing a complex system such as its availability both as command line interpreter and embedded into a web server as Apache to build the administration and control interface.

A PHP module was developed providing all the system features and its schema is reported in fig. 2 where it can be seen how a GRIB library for coding, decoding, interpolation and rotation is embedded into the PHP interpreter through an appropriate C wrapping code. The embedded functions are used to build PHP classes (made in PHP code) that provide an object oriented design and are instantiated by the Apache Web server or by command line scripts to access and to process data taken from the database or from the file system.

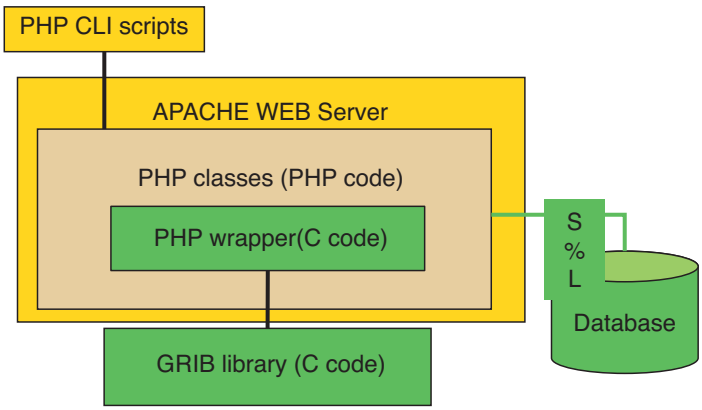

Fig. 2 PHP module is built on embedded C cod (GRIB library) and provides object oriented features accessible both from command line interpreter (CLI) scripts and from Apache web server.

As an example of the features provided by the PHP module and its classes we report the following code that after instantiating a new grib object it applies some constraints to data (With method) that are taken from the database in a while cycle. Within this cycle the values of the fields in the four nearest neighbourhood points are printed from the decoded data.

```
$grib = new GRIB;
$grib->With("id_center",80);              //  constraint id_center=80
$grib->With("id_process",2);              //  constraint id_process=2
$grib->With("ref_date","2005-11-16");     //  constraint ref_date="2005-11-16"
$grib->With("run",12);                    //  constraint run=12
$grib->With("id_parameter",11);           //  constraint id_parameter=11
...
$fh=fopen("filename","r");    // optional file handler if not
                              // given default database is used
while ($grib->GetNext( optional file handler $fh )) { // grib matching constraints are getted
        $grib->Decode();  // grib decoding excluding sec. 4
        $grib->PrintPDS(); //PDS (sec. 1) also available as $grib->pds[parm]
        $grib->PrintGDS(); //GDS (sec. 2) also available as $grib->gds[parm]
        $grib->Unpack();  // after unpack field values are available in $grib->field[i]
        $lat=40000; // latitude in millidegrees
        $lon=11000; // longitude in millidegrees
        $NearestPoints=$grib->FindNearest($lat,$lon); // find the 4 nearest neighborhoods
        foreach ($NearestPoints as $Point) {
                print $Point->latitude;
                print $Point->longitude;
                print $grib->field[$Point->index]; // field in one of the nearest points
                ...
        }
}
```

4 Activities model

A model for the activities was implemented for the system at the aim of getting control also over tasks doing very different things as elaboration and distribution. In the model developed each activity is made by a parent process always running as an high availability application on a cluster and forking a child with a configurable time interval. After forking the parent only waits for the child to exit and it doesn't know anything about the task of its child. The child does its activity that usually is triggered by a query to the database or to the file system and exits returning and exit code to its parent.

Activities are grouped in three categories: loading, elaboration and distribution.

The first activity is grib loading and it is carried on by child processes that search for new files into a list of remote directories. When a new file (not already processed or containing new data) is detected the grib data contained in it are decoded, classified (added to the static tables if not already present) and loaded into the database. As soon as the grib fields are loaded into the database by the loading activity they are immediately available for other activities.

Elaboration works as the other activities and looks for fields that can be produced (they are made starting from data that are already onto the database) and must be produced (they are not already present onto the database, i.e. not jet produced). The previous triggering conditions are implemented as a query on the database and so data present onto it are the only elements needed to start the elaboration there isn't any temporal or action trigger. The idea is simple: as soon as data are available and products were not produced jet it is time to produce them and insert into the database. The same concept is applied to distribution. If all data requested by a user are on the database, because they were loaded or produced, the distribution is triggered and if the files were not already distributed they will be.

5 Web Interface

The administration and control interface was developed using web technology and taking advantage of the high level of integration provided by PHP into a web server such as Apache that is used in operational configuration at CNMCA. Through the web interface all the system features relating to activities of the system are exposed with easy accessible and interactive graphical views provided by a model•view•controller application. Information contained into the database is presented through progress bars giving a visual meaning of the state of each activity. Moreover administration tools are provided that allows management of users and streams. As an example a new user can be added and products can be scheduled for distribution to a destination (user,host, directory) provided by the user himself and the grib data added to each stream can be chosen individually from that available on the system i.e. on the database. A geographical area can be defined as requested by the user as the system interpolate data to those areas within the elaboration activity. All those administration actions doesn't need any restart or refresh of the system and are immediately executed as soon as they are submitted.

This web interface is now used not only for monitoring or maintenance of the distribution and production activities, it is also in use for the new added activity of charts plotting. Indeed an interactive view is provided to manage plots using magics and to produce them daily.

6 Conclusions

DISTRI.C.O as is now implemented into the operational suite at CNMCA manages hundred of thousands of grib data daily providing monitoring and administration tools through its web interface. A Relational Database is efficiently used to store a large number of fields showing that it is possible to manage such data volumes using an RDBMS. Scalability for the database is assured by the implemented RDBMS (IBM Informix) up to Petabytes without loss performance providing that some fragmentation techniques are used for tables. Distributed databases can also be implemented hiding with a PHP layer multiple data sources.

The use of a scripting language as PHP embedding C code for performance and reuse of existing code gave advantages during development and operational maintenance.

The activities model and data schema implemented are now used not only for interpolation and distribution of grib fields, they are also used for other kind of daily production as that of charts.

As it is designed the activities model is scalable because it is easy to add a new activity and to monitor it without stopping the system or restarting it.

Finally DISTRI.C.O is extremely scalable and provides all the administration and monitoring features needed to a system that is integrated into an operational environment.