ECMWF Feature article

from Newsletter Number 140 – Summer 2014

COMPUTING

SAPP: a new scalable acquisition and pre-processing system at ECMWF

Salem Alkait/iStock/Thinkstock

www.ecmwf.int/en/about/news-centre/media-resources

doi:10.21957/3ylomoa6

This article appeared in the Computing section of ECMWF Newsletter No. 140 – Summer 2014, pp. 37–41.

SAPP: a new scalable acquisition and pre-processing system at ECMWF

Enrico Fucile, Cristiano Zanna, Ioannis Mallas,
Marijana Crepulja, Martin Suttie, Drasko Vasiljevic

A new scalable acquisition and pre-processing system (SAPP) was introduced into operations at ECMWF on 3 June 2014, replacing the previous acquisition and pre-processing suite (HAPP) after 20 years of continuous work. The longevity of the previous processing system is both a testament to its good design and a reflection of its underlying complexity. Such a system requires great care in making any significant changes to the framework of the processing chain and, consequently, this acts as a deterrent to major upgrades or re-engineering of the system. Only a change in computing architecture and a sharp increase in satellite data in the recent years have triggered the need for re-engineering to provide a shift of the processing model from a highly available application (i.e. one that ensures a certain degree of operational continuity) to a scalable system (i.e. one that can be enlarged to handle a growing amount of work).

The function of this fundamental module of the processing chain is to:

- Acquire observations from a multitude of sources (in the order of 200).
- Decode the various formats (e.g. BUFR, GRIB, HDF, netCDF, text).
- Apply initial quality control.
- Convert to a consolidated format before delivering to the data assimilation processing.

The main design requirements for SAPP are scalability, improved monitoring and administration, continuous observation processing, and use of Commercial Off-the-Shelf (COTS) Linux boxes.

Scalability is required to cope with the increasing volume of satellite data, while improved monitoring and administration is needed for the growing variety of data coming from a multitude of remote sensing and in-situ platforms.

The requirement of building a system with COTS Linux boxes is driven by several factors:

- The need to choose a relatively cheap solution.
- The aim of minimising the diversity of systems to be maintained in-house, as this solution is suitable for many other projects.
- The portability of a system on such a platform compared with that of a system depending on special high-availability mechanisms which are tightly linked to a specific platform.

There are also drawbacks in this choice which are mostly connected with the reliability and stability of the system. To overcome these problems it has been decided that there should be enough redundancy in the system to cope with any system failures. Indeed, the operational configuration consists of two Linux clusters, one of which is operational and the other is used as a backup and for testing and other activities (e.g. the re-processing of past observations which has become more important with the growing reanalysis needs).

A project to build a Continuous Observation Processing Environment (COPE) was initiated at ECMWF with the aim of reviewing the entire observation processing chain to (a) implement continuous processing and complex monitoring and (b) build a unified framework for filtering observations. The continuous processing requirement coming from COPE was to some extent already implemented in the existing system. However, SAPP has improved this aspect by building the system around the concept of continuity rather than running quasi-continuous tasks from an external scheduling system. The immediate advantage is a ‘natural’ load balancing because most of the observations come more or less regularly throughout the day. This means that processing them as soon as they enter the system is a very efficient way of spreading the load in time as compared to processing at regular intervals which can cause resources saturation.

The aim of this article is to illustrate the design concepts of SAPP and show some of the improvements in comparison with the previous system.

 doi:10.21957/3ylomoa6

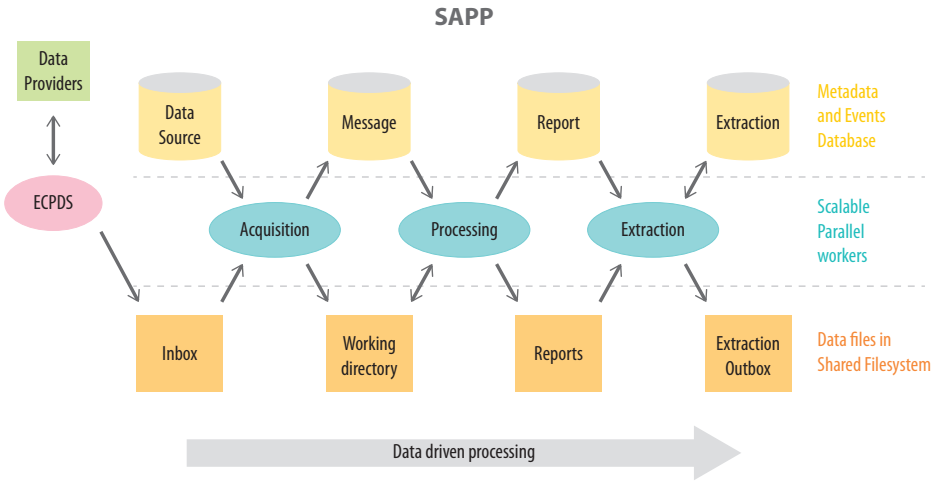

Figure 1 SAPP system design diagram.

Design

The overall system design is illustrated in Figure 1. The data is downloaded from more than 200 sources by ECPDS, which is the system providing the file moving service from and to external and internal sources or destinations. ECPDS has been developed at ECMWF and was extended so that it could handle data sources to support SAPP in the acquisition task. ECPDS is not part of the SAPP system and its development is not in the scope of this article.

The incoming data reaching the inbox directory is immediately processed by SAPP. The system design is based on a data-driven processing model in which the data is continuously pulled by parallel tasks in successive stages: acquisition, processing and extraction. The system is designed to work on a Linux cluster with multiple workers running on several nodes, spawned by a master process acting as load balancer and scheduler. A special case of workers are the processors performing the decoding, quality control and conversion of the incoming data in a standard BUFR format (*WMO*, 2013) that is then delivered to the data assimilation system through the extraction.

The process dispatcher, which is spawning the processors, can be configured to create a maximum number of processors per node. Also it assigns to each processor a maximum number of incoming messages to be processed. The dispatcher prepares the work for each processor, tagging the data in the metadata database as 'being processed' and writing on the same database the start time of the processing. At the end of the processing the result status (i.e. as 'success' or 'failure') and the time when processing finished are written on the database. This allows monitoring of the load of the system and recovery of anomalous situations by the dispatcher.

A timeout attribute is set for each processing type and the scheduler re-spawns a processor with the same data if the original processing task runs for longer than the timeout period. This allows the system to put back in the processing queue the data involved in a task which crashed or was taking longer than expected. All the timeout events are logged in the system so that anomalous behaviour of processors can be detected.

We have used a relational database to store metadata and events and a shared filesystem to make the data available to the processing nodes. The decision to keep the data on a filesystem was taken to minimise the work on the relational database engine. In addition, this ensures that the database was more manageable in terms of replication and backup, which are the main resilience mechanisms used to recover situations of database corruption. This design choice also leads to better performance and more flexibility of the system.

As an example of the benefit of the chosen design, consider the acquisition process. It is very efficient because it analyses the incoming files to insert into the database only the location and size of each message within the file. In this step, the original file remains untouched and the process dispatcher prepares the work for each processor by making a list of messages in terms of file names, position in the file and size of the message. The processor is able to access the input file which is on the shared filesystem and then a random access reading of the message that has to be processed is performed. This approach is very efficient as the preparation of the work for a single process is reduced to making a list of file name, file offsets and message sizes.

In the approach taken there is no real moving of data or preparation of an input file for the processor. This is a key aspect in the design of the system which leads to a significant improvement in performance compared with the previous HAPP system. It also has the important advantage of providing flexibility in the development phase of each processor as a processor can be developed outside of SAPP because it is working on files. The new processor can then be plugged into the system at a later stage after consolidation and when it is considered to be ready for a continuous test phase preceding the operational implementation.

SAPP versus HAPP

A summary of the comparison between the new scalable and the previous system is reported in Table 1. The improvements are mainly due to the design choices already described and satisfying the new requirements.

HAPP was not able to scale in the context of a problem that is intrinsically scalable (i.e. the processing of two different data that are not interdependent and can be performed simultaneously). The old system was able to run only one processor per type of data on one single node whereas SAPP is designed to run simultaneously several observation processors on several nodes, thereby providing the necessary scaling capability which is in the main requirements.

The old system was built around the concept of high availability of the application which is relocated to another node if there is any problem on the node on which it was running. This concept is not conveniently applied to a scalable model such as the one we were trying to design for SAPP. Therefore we decided to achieve a resilient system through an intrinsic and seamless fault tolerance. In SAPP, if a node has a problem and the processing of that node is not coming back to the process dispatcher as expected, new processes are spawned on the available nodes and the system continues to work without the need for any intervention or the use of any mechanism for relocation or recovery. The fault tolerance in this case is built into the system. Actually the system needs a highly available database server to provide enough resilience for the operational service. This is achieved with well-established replication and backup techniques which are common in the database administration domain.

Another design feature of the old system that prevented effective load balancing of the processing was the use of an SMS (Supervisor Monitor Scheduler) suite for all the activities which was designed for single process tasks. This has been addressed by designing the new system on an internal dispatcher which is used for load balancing.

One of the main requirements was the improvement of the administration of the system because the old system was based on a large set of files spread in several directories and several SMS definition files making the configuration of the system unnecessarily difficult. This requirement was implemented in SAPP by making a unique web administration interface based on an open source web framework: django.

The basic design of HAPP relied on the database not only for the metadata, but also as storage for the data, with a big overhead on the database engine both in insertion and extraction. Moreover, the processors were designed to query the database to get the data to be processed and to insert the output directly into the database. This results in a tight link between the processors and the system, and the impossibility of developing them out-of-the-box as is now the case in SAPP.

The processors are mostly written in FORTRAN and the insert and select on the database had to be performed in HAPP through a software layer, strongly dependent on the specific database engine which makes the portability to another relational database management system (RDBMS) more difficult. In SAPP, the original FORTRAN processors are used with minor modifications to read and write on files without accessing the database, while the SAPP system components are written using the Python programming language and all the accesses to the database are performed in Python, resulting in more flexibility in changing the RDBMS.

In HAPP there was only a very limited log of the activities which mainly involved error files, while SAPP logs the status and time of processing for each step as indicated in Figure 1. This gives an advantage in tracking the data through the processing steps and in building monitoring views and alarms which are extensively used in the management of daily operations.

A significant difference between SAPP and HAPP related to improving the maintainability of the system is in the use of a single component for the decoding of GTS headers (*WMO*, 2009), BUFR and GRIB (*WMO*, 2013) for the purpose of routing the messages to the appropriate processors and gathering the metadata from the produced reports. Indeed, we have developed a new decoding library (ecCodes) which is able to decode GTS headers, BUFR and GRIB with the same function calls and tools and these can be considered an important building block of SAPP. This new decoding library is described in some detail in the next section.

 doi:10.21957/3ylomoa6

HAPP	SAPP	SAPP improvements
One process per data stream on a single node	Several processes per data stream per node on several nodes	Scale out by adding more nodes to the cluster; more efficient use of resources
Highly available by relocation to another node in case of failure	Fault tolerant. Unprocessed items are made available for processing again	Seamless recovery of node or process failure
SMS suite	SAPP dispatcher	Load balancing
Configuration on files	Configuration on database managed by WEB interface	Configuration unified under unique user friendly interface
Data and metadata on database	Data on file system; metadata on database	Smaller database better performance; database backup/replication
Processors get/put data from/to database	Processors get/put data from/to files	Processors can be developed and tested as standalone and plugged into the system when ready; no need for database calls from FORTRAN
Extraction is involving retrieval of BLOB from database	Extraction relies on metadata on database and data in files	Extraction is more flexible due to the richer metadata content and faster because data are on file system, not on database
No events log, only some error files	Processing status is logged in database	Tracking data each processing step. Monitoring and live data statistics possible
No BUFR, GTS, GRIB tools used for routing	ecCodes used for routing and report metadata	Simpler use of decoding tools, more flexibility, easier maintenance
Developed with a mixture of C and FORTRAN	Developed with Python	Quicker prototyping, reduced amount of code, easier maintenance

Routing rules and ecCodes

A data acquisition and pre-processing system like SAPP has to be able to decode the incoming data in each of the processing stages, mostly because the metadata coded inside the messages is used to route the data to different processing chains; it is also used for monitoring purposes. To provide decoding capabilities to SAPP, a new decoding library has been developed based on the successful design used for grib_api *(http://software.ecmwf.int/wiki/display/GRIB/Home)*, which is the GRIB encoding/decoding software recently developed at ECMWF. This new library (ecCodes) is able to decode GTS headers, BUFR, GRIB and can be extended to decode other types of messages.

The basic principle of ecCodes (*Fucile*, 2014) is to provide access to the information items of a message through key/values pairs. Some simple functions are provided to get and set the value associated with a key, which in some cases can also be a list of values. The approach is very simple to learn and the library provides a convenient Python interface which made the implementation of the routing rules for incoming messages very simple. Since the SAPP core system is written in Python, it was particularly useful to have a decoding library available in the same language.

ecCodes is currently able to decode all the data processed in SAPP and is used to get metadata such as latitude and longitude of the station, time of observation and many other parameters from the reports produced by the processors. The metadata enables the system to provide space and time statistics in real time while the data is flowing through the processing stages.

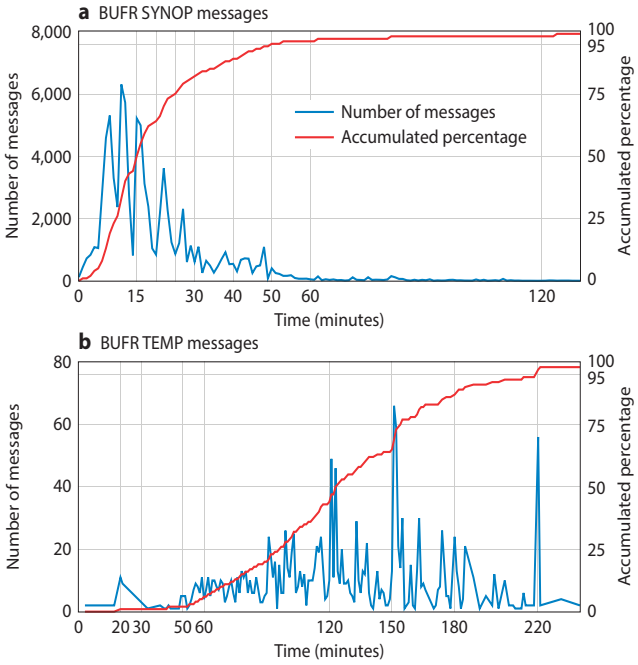

Figure 2 Number of (a) BUFR SYNOP and (b) BUFR TEMP messages as a function of the timeliness, computed as the difference between the observation and the acquisition time. The accumulated percentage of the total number of observation is also reported.

Re-processing

The activity of re-processing entire observation data streams is becoming more important with the increased need to prepare reanalyses associated with the Copernicus programme. A problem with the old HAPP system was the amount of time needed to re-process multiple years of data. Typically, for 20 years of data from a single satellite instrument, the re-processing could take more than three weeks. We estimate that the old system was processing one year of data in one day. This can appear to be a good speed, but it results in limiting the number of re-processing operations that can be completed per month to no more than one because of the limited resources available on the old system. Also, some human expertise was occasionally needed to monitor that the activity was progressing as expected.

SAPP is currently capable of processing the same data stream at a rate of one year per hour and, hence, of completing the processing of 20 years of data in two days. This will permit a significant enhancement in the re-processing throughput and satisfy the increasing need for reanalyses.

Monitoring

One of the advantages of basing the design of SAPP on a database containing metadata and events is the possibility of having several instantaneous and statistical views of the activity of the system and the data flow. A set of monitoring database queries has been developed to provide real-time alarms about events affecting the regular data processing. As an example, an alarm is sent to the console operators when a data source is not receiving data within a given period which is automatically computed by making an average of the time of inactivity for each source. Other alarms are targeted on the quality and format of the incoming data. These alarms are connected with the percentage of unsuccessful processing or rejected data per data stream. When the percentage of success goes under a fixed threshold, an alarm is sent to the console operators and the intervention of an analyst is required.

A simple use of the metadata stored in the database is shown in Figure 2 where the timeliness for BUFR SYNOP and BUFR TEMP (*WMO*, 2013) data is reported. The timeliness is computed as the difference between the observation time and the acquisition time (i.e. the time when the observation has reached SAPP). The number of messages is reported as a function of the timeliness. Also an accumulated percentage of messages received is reported which makes it easier to see when the data has reached significant thresholds in terms of percentage. In the comparison between the two plots it is clear that 95% of BUFR SYNOP data reached the processing chain 50 minutes after the observation time, while 95% of BUFR TEMP data was acquired 220 minutes after the observation time.

doi:10.21957/3ylomoa6

SAPP can provide statistics in space where map coverage can be produced in real time from the system by submitting queries based on the latitude and longitude metadata of the observation processed. This is used to plot maps of observations coverage and has been particularly useful in the development of a wiki on the data migration from Traditional Alphanumeric Codes (TAC) to BUFR (TAC to BUFR wiki) – see *http://software.ecmwf.int/wiki/display/TCBUF*. Most of the content of this wiki is produced automatically by SAPP and is, in some cases, providing a complex and precise view of the migration status. In Figure 3 we have a coverage plot on google maps of the BUFR SYNOP data produced in WMO Region VI from Italy and Greece. Other plots or tables reporting the stations producing TAC, but not yet producing BUFR, can be consulted on the wiki and are produced with relatively simple queries on the SAPP metadata database.

Administration

A web administration interface has been developed to conveniently configure the system which is processing hundreds of data sources and using hundreds of different data processors. The web administration interface (see Figure 4) is a quick and simple way of configuring the system by defining new processors with their specific properties, adding new data sources or defining new routes between existing data sources and processors. This speeds up work that is made complex by the growing number of different observations and the complexity of the data received on the GTS. Indeed, the system has the capability to define a specific routing based on the GTS header of a message. This is particularly useful when new data is introduced on the GTS or some data generated by a particular provider is affected by some problem and needs to be removed from a routing table.

Figure 3 Geographic coverage of BUFR SYNOP observations over Italy and Greece as extracted in real time from SAPP and plotted on a google map.

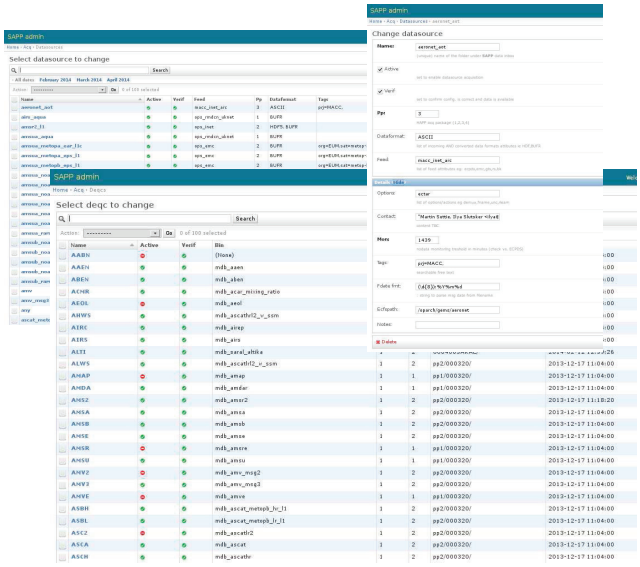

Figure 4 View of SAPP administration web interface.

Summary and outlook

The new SAPP system has improved several aspects of pre-processing observation data by providing the scalability required by future needs in data volumes and diveristy of data. It also provides improved monitoring, seamless failover recovery and a better framework to develop further observation pre-processing. This is achieved in the context of a wider project to develop a Continuous Observation Processing Environment (COPE) aimed at building a more scalable, efficient and effective system for the successive processing stages involving filtering observations and complex quality control procedures.

Further reading

WMO, 2013: Volume on Codes. *Publication N. 306.* http://www.wmo.int/pages/prog/www/WMOCodes/WMO306_vI2/ Publications/2011editionUP2013/WMO306_vI2_2011UP2013.pdf.

WMO, 2009: Manual on the Global Telecommunication System. *Publication N.386,* http://library.wmo.int/pmb_ged/ wmo_386-v1-2009_en.pdf.

Fucile, E., 2014: Semantic approach to WMO Codes. *SENSORNETS 2014 Proceedings,* 409–414.

 doi:10.21957/3ylomoa6