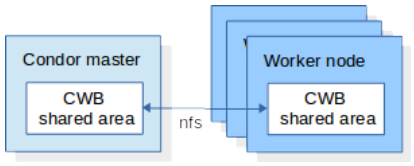
CWB Workflow Management System Architectural Design Document

Lisa Zangrando

Introduction

Coherent Wave Burst (CWB) is a joint LSC-Virgo project which implements coherent pipeline based on the constraint likelihood method for detection of gravitational wave (GW) bursts in interferometric data (for more details please see the reference [1]). By design the CWB architecture has been conceived to be executed expressly in the LSC (i.e. ATLAS and CIT) computing environments based on Condor clusters. Moreover its setup as well as its data model requires that all configuration files, several libraries and local data (i.e. frame files) must be shared among the compute nodes via a shared file system (e.g. nfs).



Although such design constraints limit considerably the capability to make CWB running in the external resources to those of LSC, the need to find a new and possibly more flexible and standardized approach is raised by the Virgo data analysis model which evolved over time and it will require more computing resources than in the past.
The intent of this document is to propose a possible solution named CWB Workflow Management System (from now on CWB WMS). In particular, starting from the list of the selected requirements the CWB WMS has to satisfy, will be described the high level architecture of the service, focusing on the main integration aspects with the existing CWB version.

Requirements

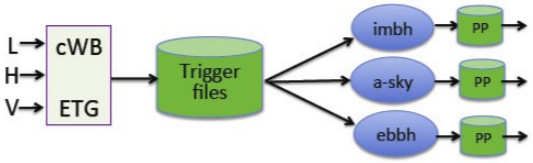
The list of the selected requirements that CWB WMS should satisfy:

- as far as possible not to require any modifications to the current CWB architectural model
- allow CWB to run in a flexible and standardized way on computing resources different to those of LSC
- allow the user to describe and execute CWB workflows, monitor and stop their execution, retrieve the produced output data at runtime (not at the completion of the whole workflow), trace the workflow status, oversee the workflow execution by accessing at runtime the CWB log file and in case of failure, recover the workflow execution
- hide the complexity to the Virgo user

The CWB data analysis model

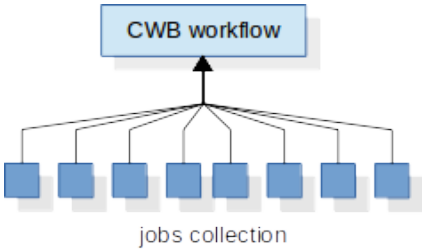
Data analysis time window is split into sub-periods analyzed in parallel by separate tasks executed on Condor Pool. In particular two specific execution modes are provided: single and double stage

- *single stage*: all-in-one task, small in/out data transfer (~200KB)
- *double stage*: two steps task, medium in/out data transfer (~15MB); the first stage pre-processes the input data and produces a reusable output needed for different and specific analysis while the second stage does specific analysis processing



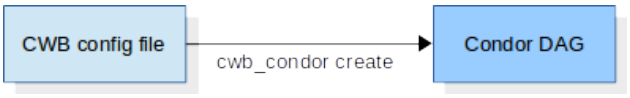
The CWB workflow

Typically the CWB workflow is expressed as Condor Directed Acyclic Graph (DAG) which describes the list of jobs to be executed in the Condor pool. Although the DAG paradigm allows the user to describe workflow even complex, composed of multi-step tasks, their relationships and dependencies in terms of data or time, CWB workflow is a flat list of simple jobs without any kind of dependency. This aspect is relevant for its design phase.



The CWB user interface

In CWB the user doesn't define the analysis workflow directly as Condor DAG but she has to setup a proper configuration file required by CWB for generating the Condor DAG. In particular the command to be invoked for the DAG generation is *cwb_condor create,* while the command *cwb_condor submit* allows the user to run the workflow on local Condor Pool (ATLAS, CIT).



Moreover the monitoring of the workflow status can be done only by using the condor tools whereas to oversee the workflow execution progress is possible by accessing at runtime the CWB log files or by analyzing the output files produced by single jobs when terminated successfully (not at the completion of the whole workflow).

Which technology?

Today there are at least two different alternatives for enabling the Virgo data analysis software to run on resources external to LSC: GRID and CLOUD. The first is a consolidated and reliable technology designed by physics for physics. CLOUD, by contrary, is an emerging and more flexible technology designed by enterprises. Although both technologies are not free of

drawbacks, they provide very attractive functionalities. Anyway being GRID a more mature project, it appears as a natural choice for the CWB.
By the user point of view, the GRID allows to spread her data analysis computing activities on different GRID sites at the same time without any specific knowledge on the GRID sites setup (unlike the CWB that run on ATLAS and CIT). Therefore GRID users belonging to Virgo Virtual Organization may access to as many Virgo GRID resources (storage and computing) as needed in a standardized way.

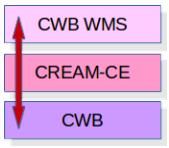
Enabling CWB to run in GRID

On GRID, the user computational activities are called grid jobs. Typically, they are independent and isolated tasks which elaborate their input data and produce the related output. Usually, the data analysis input and output files are stored in a Storage Element (SE), a GRID service which allows GRID users to store and manage files on heterogeneous storage systems. Complex analysis workflows can be implemented by adopting the Workload Management Systems (WMS), advanced frameworks which map and execute abstract application workflows over a wide range of execution environments, including GRID and CLOUD.
By trying to think CWB as GRID application, the analogy becomes evident when comparing the grid job type with the flat collection of independent jobs composing the CWB workflow. Nevertheless the CWB data model is not well suitable to the GRID mode and this behaviour will require the development of ad hoc solutions to fit the CWB structure.

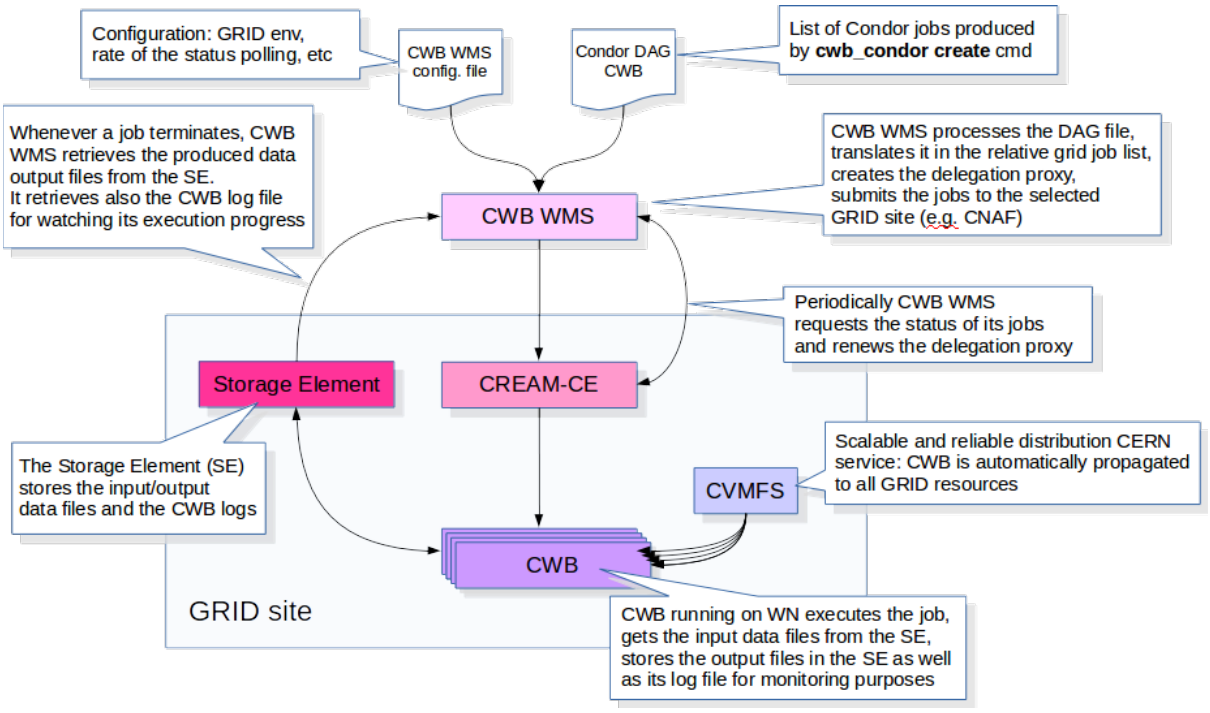
The CWB WMS high level architecture

Although CWB WMS has been conceived for enabling the CWB data analysis to be executed in GRID environments, it may be extended so that it can be used for executing pipelines different to those of CWB. Anyway, its high level architecture is split in three layers:

- client layer (CWB WMS): preparation, submission and monitoring of grid jobs
- server layer (CREAM-CE [2]): handles the user (i.e. CWB WMS) requests
- core layer (CWB): involves the process(es) to be run on a Worker Node



All workflow jobs pass through those three layers and the produced output data files will be uploaded directly to the selected storage area which is typically a Storage Element. Its conceptual design is shown in the next Figure. By design, CWB WMS will be a thin daemon fully implemented in Java which will access the Virgo GRID resources (e.g. CNAF) based on CREAM-CE. Unlike other WMSs (like Pegasus) that have to invoke third parts GRID commands for interacting with the related GRID environments (e.g. glite-ce-job-submit) and consequently have to parse the standard output (i.e. text) of the invoked commands, CWB WMS will access the Virgo GRID resources by invoking directly the CREAM Web-Service interfaces (WSDL-SOAP). This approach will improve considerably the communication layer in terms of performance and reliability. Moreover tools other than those GRID are not required (e.g. external libs, SW, etc)



All steps involved in the CWB WMS business logic are fully described in the Figure. Starting from the Condor DAG file produced by the *cwb_condor create* command, CWB WMS will create the related grid jobs list to be submitted to the selected CREAM-CE specified in its configuration file. To allow CREAM to execute actions (e.g. data staging) on behalf of the user, it requires a delegation explicitly created by the user itself. After the delegation and job submission phases, CWB WMS will monitor periodically the status of its jobs. Especially it will take care of retrieving the output data files produced by the CWB processes whenever they terminate successfully. In case of failure, it will try to re-submit the failed jobs by reporting to the user the reasons of the observed problems. Moreover, to oversee the workflow execution progress, CWB WMS will provide to the user the CWB log files produced at runtime. Finally, since CWB and its auxiliary libraries are too large (~200MB) to be transferred along with the GRID job submission, all required software is automatically propagated to all GRID sites by the CVMFS service. This approach guarantee all resources are synchronized and updated along with the latest CWB release version.

What is missing in CWB?

Some significant changes that result from a previous attempt to make CWB able to run in GRID have been already implemented in CWB such as the support of the Storage Element as well as the propagation mechanism realized with CVMFS. What is still missing is the trick of the CWB log file transfer for tracking the workflow progress.

References

[1] https://atlas3.atlas.aei.uni-hannover.de/~waveburst/doc/cwb/man/What-is-CWB.html#What-is-CWB
[2] https://wiki.italiangrid.it/twiki/bin/view/CREAM