WORK REPORT:

SCADA system for Offline Data Processing in the LHCb experiment.

Author: Manuel Mazo Jr. <manuelme@ieee.org>
Finishing Date: 24/8/2001

Abstract: This system aim is to provide a common user interface to control and follow the Offline Data Processing in farms of computers. It gives the user information around the stage of the Data Analysis in each computer, as well as information on the environment of each process and the state of the processing queues in each farm. All the system has been build based on the DIM communications package and the PVSS SCADA system.

1.- Motivations:

After collecting data from an experiment, and after passing some previous tests, it comes a data processing stage. Due to the great amount of data collected from the experiments, this data processing and analysis must be done in big farms of computers distributed all around the world. One desirable thing when doing this kind of distributed processing is knowing in which state of processing is each job (packet of data being analyzed). It is also interesting having a way to decide in "real time" (while processing) if there is any problem, and by that mean, you can have the possibility of deciding to stop certain jobs that will not give correct information.

If we have all this facilities, when we get new data to be processed we could do a simple "computational charge distribution", and send, in a pseudo-automatic way, that data to the farm, or computer, with less charge in each moment, or just randomly. Then we could control all the data processing from a single computer.

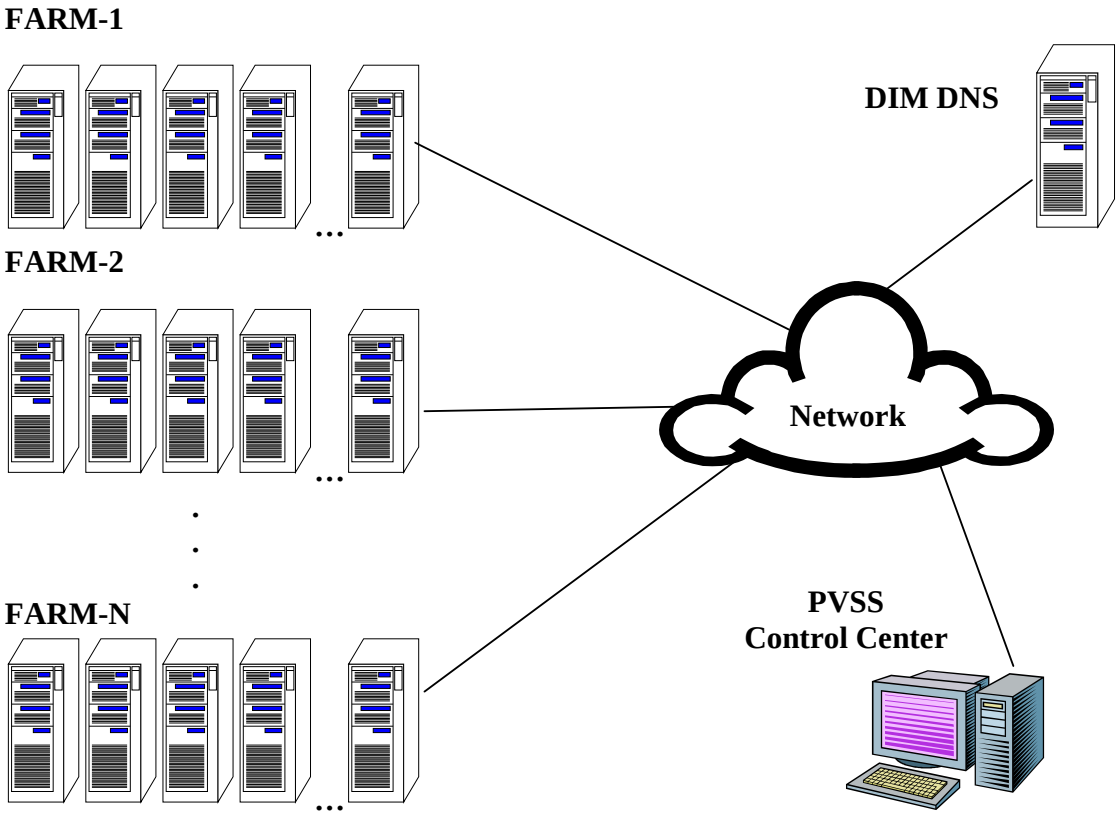
Another desired facility is being able to have different control centers for different experiments data analysis, even if they are using the same farms of computers to do their computation.

Using PVSS as SCADA system gives some extra possibilities as the possibility of distributing the control system in a small group of computers to have a better response and enlarge the number of jobs that it could manage to monitorize.

The possibilities of the system build will be discussed in the following points.

2.- General Overview:

The system build can be divided at least in two big subsystems: the server that allows the data processing scripts send information to the control center, and the SCADA system running in the control center.



The figure above represents the general structure of the system. We have different farms of computers that will process the information, a machine that will be the intermediary between the farms and the Control Center. The basic idea on how this system works is: the farms, when one of its machines start processing a data processing job, always starts a server that will publish some information and advise the DIM DNS of that, that way the Control Center with a simple query will know how many jobs are running and will be able to get the information that each of those jobs are publishing.

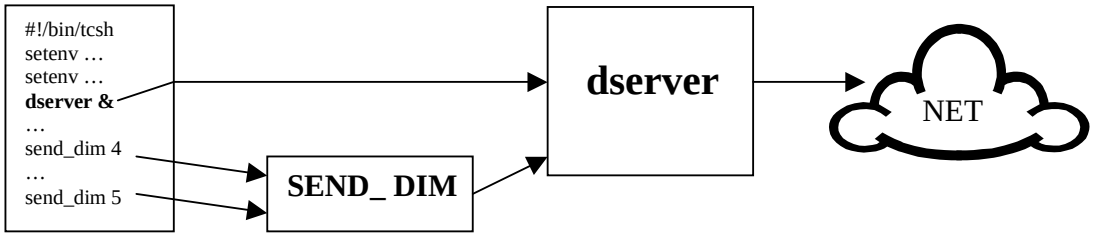
The PVSS system can be just one computer, or a distributed system running in different computers the different processes that compose the PVSS system.

3.- Subsystems:

As we have said before this system has two different parts, one that publish information and the other that collects that information and present it in a user interface. The first subsystem, the one publishing the information, is basically a server that we have called *"dserver"*. The second subsystem is the PVSS SCADA system, in which some panels have been built to display the different information around each job.

3.1.- DServer:

The publishing part of the system is composed by: the "dserver", the shell script, and "send_dim". These blocks are related as it can be seen in the figure:



We have a shell script that launches the different programs involved in the data processing, that shell script, just after setting all the environment variables it uses, has to launch the "*dserver*", and afterwards, each time the script need to update it status in the server, it uses the *"send_dim"* script, as a command in which its second argument has to be a number related with the actual status.

Internally, ***"send_dim"*** acts just as a redirection, passing the value received to the input of a pipe created by *"dserver"* according to the structure: **/tmp/*hostname_ppid*_DIM**, where PPID is the parent process ID of dserver, it is, the PID of the shell script.

The "***dserver"*** just get all the environment variables set up in the script just before launching the "dserver", and get the information contained on them. This is possible cause "dserver" as it is launched by the script, it inherits the environment of its parent process, the script.

The ***"dserver"*** has three different ways of collecting information:

1) The first way is through the environment variables that have been previously set up in the script.
2) Through the "pipe" created to collect the information sent by the shell script, using "send_dim".

3) And the last way, via the execution of some shell commands redirecting their outputs to a pipe previously created to read the output of the commands through it.

Apart from the dserver, there is also another server that publishes the jobs queues state of each center, and that actualize that values when it receives a command from the PVSS control center. That server has been called **comrsh.**

To actualize the values **comrsh**, do *"rsh"* to the different centers and get the values executing the correspondent command in each center.

This way we have to different channels that give us information around the queues state; one is this comrsh server, and the other one is the dserver of each job that publishes the queue status of the center in which that job is running, when it is started and when it finishes.

3.2.- PVSS/Control Center:

The control center provides the user a graphical user interface, with a ***main panel*** in which the new jobs appear as they start publishing information on the DNS. From this panel the user can also access to the complete information available about the job selected, in the ***All_info*** panel. There is a third panel that displays information on the number of jobs in the queues of each processing center (farm of computers).

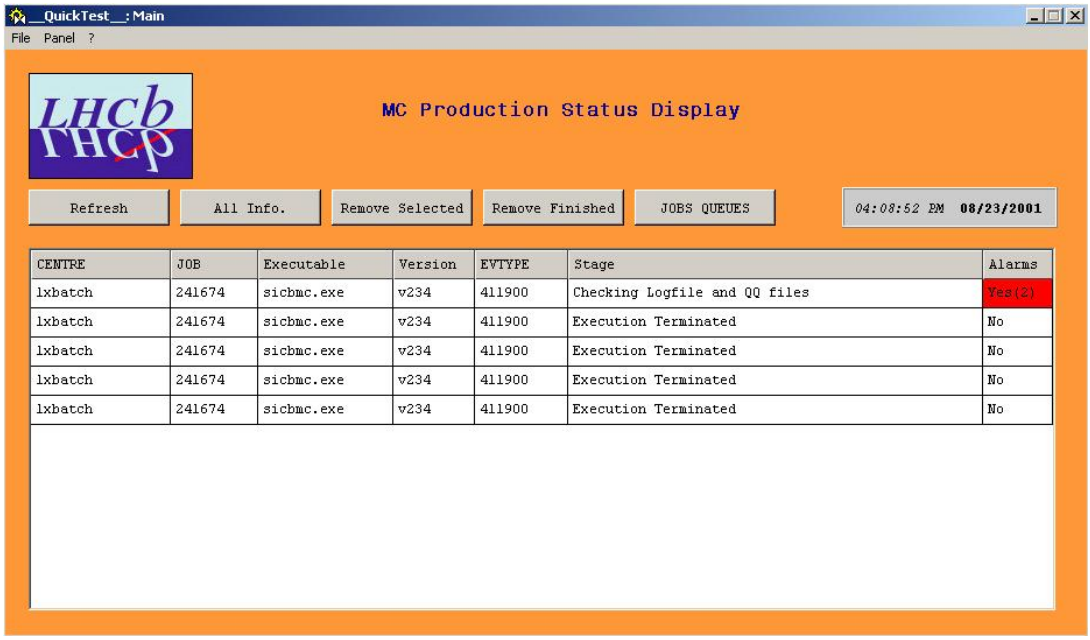
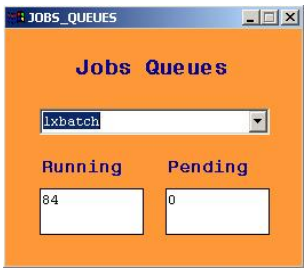


fig.3 Main Panel



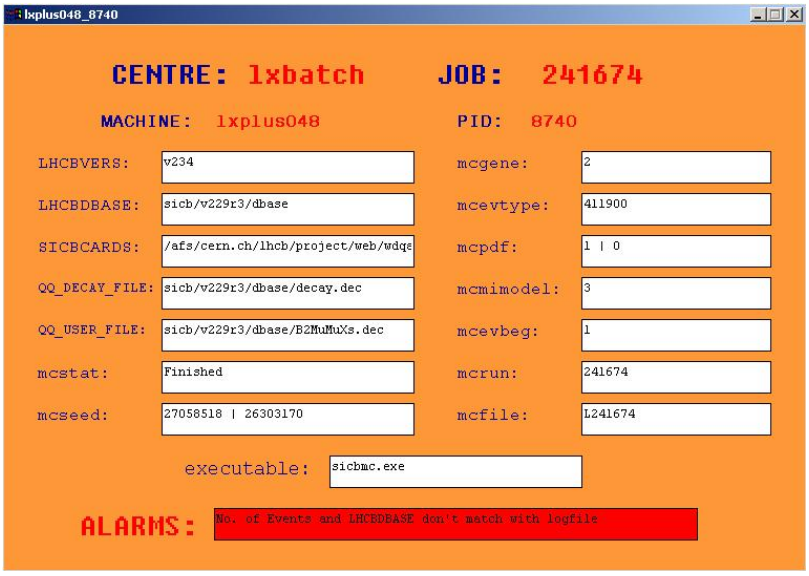


fig.4 Jobs Queues Panel **fig.5 All information panel**

The PVSS system build basically uses the DIM package (PVSSdim.exe) for PVSS, for communications with the DIM DNS, and a "ctrl script" written to create new DPEs (Data Pointer Elements) of type **ProcStat** when new jobs appear published in the DNS.

The system uses four different DPs (Data Pointers, data structures used by PVSS), the ones related with DIM: ***FwDimConfig***, another to store information on each job: ***ProcStat***, a third one to store the job's queues state of each center: ***CentresJobs***, and the last one, called ***QJobsRSH***, that is used to send DIM commands to the **comrsh** server.

4.- Running the System

Whenever you have to start the complete system, the two first things that must be launched before any other subsystem are the **DNS DIM server**, and **comrsh.** Then afterwards the PVSS system can be started, just taking in mind that the ctrl script: **subscribe_cpus**, has to be the last manager to be launched. It is specially important that the *subscribe_cpus* script is run after *comrsh*, cause otherwise the complete PVSS system won't be able to get the jobs queues status using this server; that happens because *subscribe_cpus*, just looks for the *comrsh* services once, just when it is started this script.