Job Submitting Tool

Ricardo Graciani Díaz, Rubén Vizcaya Carrillo
Universidad de Barcelona
Juan Saborido Silva, Manuel Sánchez García
CERN

26 January 2004

Abstract

This document describes the functionality expected from a “Job Submitting Tool” to be developed to handle the job submission for DC’04 production. This tool is called from the “Production Management Interface” to handle the job instantiation and submission to DIRAC.

1. Introduction

For LHCb DC'04, the Production Manager, PM, is responsible for the execution of few hundred thousand Jobs. To help him in this task, the "Production Management Interface" is expected to give him access to different tools. The relevant ones for the purpose of this document are the "Workflow Editor", WE, and the "Job Submitting Tool", JST.

The DC'04 is organized in different **Productions**, corresponding to different event types and processing algorithms. Different options may also be used for a given algorithm, giving rise to new Productions. The description of the productions is done via **Workflows**, and the Workflows are used to instantiate a number of Jobs that are submitted for execution via **DIRAC** "Workload Management System", WMS.

The WE allows for a easy definition of the Workflows that are structured into one or more Steps that connect one to another via output and input data files. Each Step is made out of different modules that describe, not only the execution of the major Gaudi Applications, but also the other necessary actions like copying input or output data, inform the Monitoring System,… The complete description of the Workflow is made persistent using an XML description[1].

The JST receives the description of a Workflow together with a request to launch the corresponding Production. The JST must then instantiate and submit to the WMS a number of jobs that would execute the requested task.

Two different Use Cases are presented in Section 2 to help focussing the discussion. For the sake of clarification, a possible file naming convention is presented in Section 3, that slightly differs from the one used in DC'03. The required functionality coming out of these Use Cases is introduced in Section 4.

2. DC'04 Production Use Cases

Joel made a clear presentation on his e-mail from Jan 19th, of three Use Cases related to DC'04 production. For the purpose of this document they are reduced to two by combining the second and third into a single one. The can be called "Production with No Input Data Use Case" and "Production with Input Data Use Case".

2.1 Production with No Input Data Use Case

As the name says, they correspond to self contained workflows, in the sense that their jobs required no data files from other productions to be executed. This was the case in DC'03:

1. The workflow defines a number of output data files that, if there is more than one step may be later used as input data file for a later step.

2. The PM uses the WET to define the desired Workflow.

3. Then he may first want to launch a small test production with few thousand events.

4. Once this has been checked he decides to launch the requested amount, let's say one million events.

[1] For the purpose of this document an XML description of the Workflow similar to the Job descriptions in DC'03 is assumed. Other descriptions could also be used.

5. And later on, because there is a new request and there are still available computing resources he decides to launch another five hundred thousand events.

6. It may happened that, after the DC'04 production phase, during the analysis of the data it is decided to double the statistics and a further one and a half million event production needs to be launched.

In each of the successive cases, the PM would like to use the same production ID (except perhaps the initial test production), and the number of wanted events; passing these data to the JST and get newer jobs instantiated and submitted using correlative run numbers.

2.2 Production with Input Data Use Case

Assume now that the DC'04 has started, at some point the stripping algorithms get fixed and the PM wants to launch a production to run the stripping. He would proceed as follows:

1. First, he creates the corresponding workflow, that now needs to reflect the fact the that its is intended to process a very specific type data files.

2. He now wants to run a small test on a small data sample, so he queries the Bookkeeping DB to get the corresponding list of files and passes it together with the workflow description to the JST.

3. Once this test production has been checked, he wants to launch the rest of the productions, that is, jobs that will processed all the data available in the BKDB of the type specified in the workflow, plus those to process all the data that is still to become available as the DC'04 goes on.

4. Further more, if the production of the input data is extended the PM (as in 2.1) will like that the stripping is also run on the new data.

As in the previous case, the PM would like all the production to share the same production ID, and all jobs to have correlative run numbers (except may be the first test production).

3. File Naming Convention

In order to fulfil the above use cases, the JST must be able to understand the description of the input and output data files included in the description of the Workflow. We proposed this to be based on the LFN of the files, they should included all the necessary information for an unique identification of the Job that produced them. The following is a possible naming convention for this files:

ProductionID_FileID_RunNo.Extenxion[2]

This LFN will allow to use a simple matching algorithm in the job instantiation procedure. Using this convention, data files internal to the workflow can be described for instance:

$ProductionID$_1_$RunNo$.ooSim,
$ProductionID$_2_$RunNo$.ooSim,
$ProductionID$_3_$RunNo$.ooSim,
$ProductionID$_4_$RunNo$.ooSim,
$ProductionID$_5_$RunNo$.ooDst,
$ProductionID$_6_$RunNo$.ooDst,

[2] For a simpler matching pattern a change in the order of the FileID and the RunNo is proposed.

$ProductionID$_7_$RunNo$.ooDst,
$ProductionID$_8_$RunNo$.ooDst

At the time of instantiating the job, ProductionID and RunNo are defined as RUNPARAMETERS and their values substituted whenever $ ProductionID$ or $RunNo$ appears.

For workflows with input data, each input file is file is declared in the RUNPARAMETERS list:

InputNType='ProductionID_FileID'

Where N varies from 1 to the number of input file types needed. These parameters are used to define new parameters, **Input1, Input2,…, InputM**, when a matching is done between the corresponding type and name in the list of input LFN. These parameters are then used as in the case above, $InputM$, for a parameter substitution in the STEPPARAMETER and STEPOPTION sections of the job description.

4. Required functionality of the JST

After the above conventions are assumed, the functionality required to the JST is the following:

1. Assign a new ProductionID to a Workflow description, determine if the workflow correspond to Use Case 1 (No Input Data) or 2 (With Input Data), keep a persistent description of the Workflow (by populating the Production DB??), and return the ProductionID to the Client.

2. Handle explicit requests for either of the Use Cases. In all cases, the ProductionID must be send as parameter, others parameters are:

   Use Case 1:

   a. Njobs (Integer): total number of jobs to be submitted. The jobs are immediately instantiated and submitted, the total number of submitted jobs for the production is return. A Query and Update of the PDB is necessary to set the current RunNo.

   Use Case 2:

   b. Njobs (Integer): total number of jobs to be submitted, zero means no limit.

   c. List of LFN (optional): if a list is passed, jobs are instantiated using this list to match the input files types, as described above. As many jobs as possible (or at most Njobs if specified above) are submitted.

      If no list is given the JST interrogate the BKDB for LFN of each of the different Input Files Types defined in the Workflow and instantiates as many jobs as possible (or as many jobs as defined by the Njobs parameter). The total number of submitted jobs for the production is return.

   For each ProductionID, a table of used files is to be created, and checked for every new LFN. Where this table is kept is to be defined[3].

   A new RUNPARAMETER may be introduced in the workflow description to explicitly declare whether all input files for a given job must be already replicated at the same SE. This condition is then applied previous to the LFN matching, by splitting the list of files return from

[3] Notice that this table of used LFN is per Production, it is a responsibility of the PM to determine if a given data is to be reused again (i.e., reprocessing) or not.

the query to the BKDB by site if necessary (a given LFN may be on more that one of this sub-lists, after being used once it will be written to the table of used LFN and is not used again).

In order to extend the production for new files produced there are several possibilities:

3.a) The PM sends new requests after a certain time, that triggers a new query to the BKDB as above. The table of used LFN prevents duplicated use of files.

3.b) The JST, and until the PM explicitly stop the production, periodically queries the BKDB instantiating and submitting new jobs (probably by calling itself with some cron mechanism).

3.c) The JST is informed, by the Production System, as new files are ready and this information can be used to instantiate and submit new jobs, until the PM explicitly stops the production.

4. Inform the Production DB about the submitted jobs.

5. Wait for new requests.

The 3.a) option above is the easier to implement but requires continuous human intervention from the PM. Option 3.b) requires to implement a stop method for the productions (Use Case 2 only) to avoid useless queries to the BKDB once the production of input files is completed. The production can be restarted at any moment using the procedure described in step 2.

Option 3.c), is clearly more difficult to implement providing the benefit of avoiding extra queries to the BKDB at the price of having to introduce extra DB's to keep the information of ready input files are not yet used since there are not enough files to instantiate a new job. Furthermore it implies modifications on the Production System and, eventually, on the Data Management Tool in order to notify the JST about new files or replicas available. It creates more interdependences with other parts of the system, and is now not considered as baseline.

5. Requirements to WMS

In order to take advantage of the JST being able to submit jobs with all their input files on the same SE, the WMS must be able to submit it to the corresponding CE.

If production jobs corresponding to Use Case 2 (merging, digitisation + reconstruction, stripping) are to be run as soon as enough input data is available, some priority mechanism should be implemented in the WMS, otherwise they will be in the queue until all previously submitted jobs from Use Case 1 (simulation) are executed[4]. This priority may be part of the workflow definition as a new RUNPARAMATER or it may be implemented on the WMS interface.

Terminology

DB	Data Base
DC	Data Challenge
JST	Job Submitting Tool
PDB	Production DB
PM	Production Manager
WFE	Workflow Editor
WMS	Workload Management System

[4] Remember that for DC'04 they correspond to 95% of the total CPU.