Cross Experiment Workflow Management: The Runjob Project

D. Evans[1], G.E. Graham[2], P. Love[3]

[1]Imperial College, London, UK
[2]Fermi National Accerator Laboratory, Batavia IL, USA
[3]Lancaster University, Lancaster, UK

Abstract
Building on several years of success with the MCRunjob projects at DZero and CMS, a joint Runjob project aims to provide a Workflow description language common to three experiments: DZero, CMS and CDF. This project will encapsulate the remote processing experiences of the three experiments in an extensible software architecture using web services as the communication medium. The core of the Runjob project will be the Shahkar software packages that provide services for describing jobs and targeting them at different execution environments. A common interface to multiple storage and compute grid elements will be provided, allowing the three experiments to share hardware resources in a transparent manner. Several tools provided by Shahkar are discussed including FileMetaBrokers, which provide a uniform way to handle files and metadata over a distributed cluster, the ShREEK runtime execution environment that allows executable jobs to provide a real time monitoring and control interface to any system, and the ScriptObject generic task encapsulation package and XMLProcessor object persistency tool.

Introduction
The MCRunjob project has been the DZero experiment's Monte Carlo production workflow planner since 1999. The main tasks for MCRunjob was to wrap multiple processing executables to be submitted as batch jobs, management of metadata associated with each job step, provide communication between the job steps, and provide tools for jobs to interact with external databases and data handling systems. In 2002 the CMS experiment started using MCRunjob for regular production operations and independently developed the code from DZero. A joint Runjob [1] project was started by building a common codebase to be used by each experiment in order to reduce development and maintenance costs, this package has been established and is called Shahkar [1]. In addition to DZero and CMS, Fermilab's CDF experiment will also utilize the Runjob system as a workflow manager.

Runjob architecture
Runjob architecture is dominated by three main classes as shown in Figure 1. These constitute the core code in the base Runjob package and build upon the legacy MCRunjob experience.

The Framework - The Framework generates framework calls which are sent to Configurators and control their actions at various stages of the job lifecycle. Typical series of calls would be: Reset, MakeJob, MakeScript, RunJob. The Framework structure is flexible enough to allow user-defined cycles.

Configurators - Configurators are objects which contain keyword fields describing a particular application or process and respond to framework calls using registered framework handlers. Configurators may or may not respond to particular framework calls allowing various types of Configurator to operate. For example, we may have ScriptGen Configurators which only respond to Reset and MakeScript calls, whereas a ForkConfigurator would only respond to a RunJob call.

The Linker - The Linker is a container object keeping an ordered list of Configurators defining the workflow. Communication between Configurators is managed via the Linker and it is the Linker which executes the framework calls on its collection of Configurators.

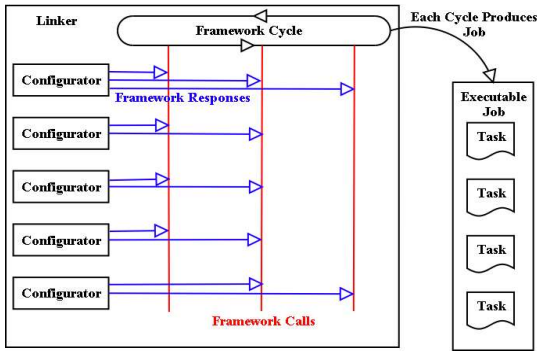

Figure 1. Core Runjob architecture with the Linker containing multiple Configurators which each respond to Framework calls.

Runjob sub-packages

Runjob is written in Python and contains several auxillary subpackages designed to be independent from the core code. These packages are described below.

XMLProcessor - The XMLProcessor package is a toolkit for transforming python objects into an XML representation, for persistancy or network transport purposes, and then transforming the XML back into python objects. Most native python types and user-defined classes are supported, with complex user-defined types supported via registration of handler methods. Conversion and Extraction is recursive so that nested data structures are handled automatically. This subpackage is deemed essential for future development of Runjob as a Webservice.

FileMetaBrokers - The FileMetaBroker (FMB) class abstracts the handling of data files and their associated metadata. This class is derived from a python dictionary and contains key-value data describing various aspects of the file and its metadata. Example fields include FileName, PathName, HostName, TransportMethod, DescFiles, etc. Transformation methods are included in a supporting library which allow FMBs to encapsulate the information needed by a particular transportation protocol. FMBs may be saved in persistent XML form via the Shahkar XMLProcessor.

ShREEK - The Shahkar Runtime Execution Environment Kit is a thread based execution manager for executables, giving real-time monitoring and intelligent error handling logic. This wrapping mechanism is essential in order to protect the workflow from the vagaries of the highly complex and sometimes problematic applications seen in high energy physics. The implementation consists of a python package which is shipped with a job to the execution node, whereupon it launches each execution task with an associated monitoring thread. An RPC interface for the job is also provided via XML-RPC and HTTP if required.

ScriptObjects - ScriptObjects provide information about an executable task from which scripts may be generated capable of being run in various environments. These objects are produced by Configurators with the effect that Configurators don't need to know about how the task will be run. Specialized ScriptGenerators will produce concrete job scripts for various Batch, Grid, MOP, ImpalaLite environments.

Conclusion

The development of the Shahkar core and satellite experiment specific extension packages are enabling simple description of jobs and rigorous metadata and provenance cataloging for the three involved experiments. In addition, advances made by any of the experiments can be swiftly shared with the others, thus enabling fast adaptation to new Grid Environments.

References

[1] Runjob Project webpage *http://projects.fnal.gov/runjob/*