# Virtualization of Workflows for Data Intensive Computation

Sreekanth Pothanis [1,2], Arcot Rajasekar [3,4], Reagan Moore [3,4].
[1] *Center for Computation and Technology, Louisiana State University, Baton Rouge, LA.*
[2] *Department of Computer Science, Louisiana State University, Baton Rouge, LA.*
[3] *School of Information and Library Science, University of North Carolina, Chapel Hill, NC.*
[4] *Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC.*
*sreekanth@cct.lsu.edu, sekar@renci.org, rwmoore@renci.org*

## Abstract

*Many scientific computational analyses rely heavily on workflow management systems to both link and automate execution of multiple tasks. The execution time required for such workflows can span days and require storage of massive data output into data grids. Past attempts to bridge this gap between workflow systems and data grids modify the workflow system to include additional steps, making the solution very specific to the workflow system and the data grid. Our workflow-independent approach is based on a stand-alone, modular workflow virtualization server that interfaces with many workflow systems and acts as a conduit between data grids and executing workflows. A complex workflow can span many workflow systems while seamlessly supporting large-scale data management. Modularity in the server design at the workflow level accommodates customization of workflow execution and also enables additional logic such as collection of metadata, and trimming or modification of the produced data sets. We have virtualized data management across workflow systems such as Makeflow, and Pegasus using the iRODS data grid system.*

## 1. Introduction

Recent years have seen a noticeable shift towards data-driven research from conventional computational research methods. Physicists now run multiple versions of complex simulations with a large number of variables, and statistically aggregate results. Examples include the study of black hole evolution or generation of Gamma ray bursts [1]. In chemistry and biology, the results of multiple molecular simulations are managed to enable further analysis of the plethora of information that is generated. In the Ocean Observatory Initiative, real-time sensor data will be compared with prior archived data and simulation output to detect events[2]. A unifying requirement for these examples is a workflow that integrates multiple computational steps with both massive data input sets and the storage and organization of massive output data sets. Such data-driven tasks are enabled by High performance computing cyber infrastructure, like TeraGrid [3] or LONI [4] and usually are executed across distributed systems.

### 1.1 Workflows systems and Data grids

Scientific Workflows are used to manage most of these complex computations. Workflows link the scientific tasks to input data sets, manage transfer of information between tasks, and generate output data sets. A workflow system can also be considered as a composition of related sub-workflows, in which each of the tasks can be executed as an independent task or service. The integration of multiple workflow systems can take advantage of capabilities unique to a specific workflow environment and achieving seamless orchestration at sub-workflow levels across multiple workflow systems is one of the goals of our project.

Large amounts of data may be generated by these workflows or workflow systems. Typically it is not possible to bring back the entire output data sets to local machines for further analysis. More often than not, these datasets are moved to data grids to enable further use. Orchestration of data movement and integration of data organization and management with workflow systems, relieving the onus from the scientist, is a second goal of our project.

After successful execution of each workflow, the output datasets have to be moved from the compute resource to long-term storage space, need to be organized to enable collaboration and data sharing, need assignment of metadata to provide a context for discovery, should be replicated to mitigate risk of data loss, and so on. Data grids simplify access to remote input data, and simplify management of the output data sets. Data grids organize distributed data into a shareable collection and manage the properties of the collection while interacting with multiple types of storage systems.

A significant opportunity for innovation is the integration of server-side workflows for data management with traditional client-driven workflows. This paper details such integration. Additional information on the design of iRODS, micro-services and rules will be dealt with in later sections of the paper.

### 1.2 Integration of data grids and workflow systems

Apart from executing a complex workflow according to specified input parameters, scientists are also challenged with the issue of archiving the produced datasets for the purposes of further analyses and collaboration. This in itself can be extremely time consuming and require attention to data management details. There are many workflow systems that are popular in the scientific community, and though not equal in number, there are a variety of data grid middleware to manage data. This introduces an unfortunate learning curve on the part of researchers before making effective use of their chosen systems.

There have been various attempts to overcome these data management challenges. But most of them aim at modifying the workflow system, by addition of a few extra tasks that would attempt to automate the movement of data into storage areas. Such attempts are very specific to the workflow. A specific challenge is integration with the workflow system's trust environment. The transfer of files from a user account under which the workflow may be executed, to a collection account that controls a shared collection requires a change in ownership of the data. The choice of mechanism to affect this transfer can be workflow specific.

Our approach, in integrating workflows and data grids, is based on a stand-alone modular workflow virtualization server, which acts as an interface between multiple workflow systems and data grid systems. The development of a fault tolerant workflow virtualization mechanism that is integrated into the same trust environment as the data management servers, bridges the gap between the production of data and their archival storage in an efficient way. Sections 2 and 3 provide a detailed explanation of the design and implementation of such a system. Sections 4 and 5 discuss future work and conclusions.

## 2. Workflow Virtualization

Workflow virtualization is the management of the properties of a workflow while it is executing. This includes ensuring that input data sets are available on required resources, the status of the workflow is monitored, and the output data sets are appropriately disposed. An implication of workflow virtualization is the ability to run a workflow in a variety of workflow systems, and manage the interactions with each workflow system for both input and output of files. This capability can be achieved by creating a workflow manager that will act as a conduit between the workflow system and the data grid and also between various workflow systems for both input and output operations. Advantages of workflow virtualization are manifold. While it provides a tighter grasp over the execution of workflows, it also enables execution of complex workflows spanning multiple different workflow systems while automating large-scale data transfers. Moreover, the virtualization server can also be used as a central point for gathering statistics, for monitoring execution, and for launching recovery mechanisms on failures. A check-pointing framework for stopping and re-launching workflows on demand can also be implemented. The data required for execution of complex workflows usually resides on geographically distributed data grids, from where data staging can be managed by the workflow server through trusted communication.

The concept of the virtualization server is based on its ability to be external to the environment that actually executes the workflow. This decouples the workflow from the execution environment, increases the generality, and enables integration with data grids. It is also highly modular, enabling integration of additional workflow systems with minimal effort. In the later sections we will see in detail how this feature will also enable a higher degree of control by administrators and developers over the execution and management of the workflow.

### 2.1 Workflow virtualization server (WVS)

The foremost functionality of WVS is to act as an interface between the workflows, the data grid, and the workflow systems. It is proposed to be stand-alone and external to the data grid for several reasons. The most important of them being the inherent robustness it brings to the system from a security standpoint. Figure

1 shows an architectural depiction of the system. Upon the receipt of a valid request to execute a workflow, WVS would set up the environment required for the workflow, stage the required input files to the workflow execution host from the data grid, hand off the configuration and input files to the module interfacing to the workflow system, and launch the workflow process. After a successful run, depending upon whether or not the request includes execution of additional workflows, the generated data would be either transmitted to the next appropriate module or archived back into data grid. The core of the server is coupled with the data grid to be able to achieve major functionalities such as authentication and authorization, and extraction and maintenance of contextual information (metadata).

Most of the workflow systems run as an operating system user, allowing direct access to data. In such an environment, running the WVS on the same host as the data grid might create grave security loopholes. For example, the submit scripts accepted by the Makeflow engine [5] are very similar to Make files [6]. Hence executing a Makeflow workflow as an OS user on the WVS could be catastrophic. The general flow of tasks is depicted in Figure 2.

## 2.2 Authentication and context handling

Maintaining a trusted environment for an authenticated user is of the utmost importance in a distributed environment. Authentication has to be handled at multiple levels, once at the data grid level while performing operations within the data grid and again at the operating system level while executing the workflow. Authentication with a data grid has to be done according to the prescribed standards of the data grid. For example iRODS uses an administrative account to proxy as any other user while maintaining a trusted context. Hence the server can perform all of the operations for the data grid user who submitted the request, once it is configured as an administrator account within the data grid. This approach may be different for other data grids.

At the operating system level, authentication and context maintenance are can be managed using proxy accounts. When a request for execution of a workflow is received, the WVS will create a logical name space and perform all the required tasks under this space. This will be the working directory for any future tasks related to data handling. But, when concurrent workflows are executed on the same server, there is a chance of a workflow stepping on each other's data. To overcome this, the process actually executing the workflow can be circumscribed to the logical directory. Another approach to overcome this situation is to spawn operating system level users whenever a new request is received. This might be unfeasible because, it would add considerable overhead to computation and can lead to security flaws.

Context information can broadly be categorized into data grid context, and workflow execution context. Pertinent information regarding the data grid can be obtained from its context. These details could include, user information on the data grid, such as privileges, quota, and default preferences. It could also be useful to obtain information regarding the grid. Examples include, host from which the request was made, other hosts and their storage capacities available on the grid, and system polices imposed by administrators. Workflow context also needs to be generated after each request is received. Some of the information would include type of workflow or workflow systems to be executed in the request, logical working name space, list of input files and configuration files required by the workflow systems, list of output files, storage destination on the data grid, and system and user metadata, if any, to be tagged to the generated datasets. None of the information should be workflow system specific because a typical request could span multiple workflow systems.

## 2.3 Staging, pre-processing and data transfers

Most of the present day complex scientific workflows manipulate large amounts of data. Automation of staging of the data to the execution resource is highly desirable. When a request is received for execution of a particular workflow or a set of workflows, WVS will check all of the requirements and set up the environment. This not only saves time, but also lets workflow developers manage data within the context of a logical name space. They do not have to specify physical storage locations or be concerned with moving data. Staging is performed just before the interface module is called; this is particularly efficient when executing multiple workflow systems in the same request. The time required for transfer of input files for the subsequent workflow is eliminated; thereby execution of successive workflows can be started immediately.

Apart from staging, certain other pre-processing tasks are also carried out before executing the workflow. Most of the system metadata attributes are initialized at this stage, to be later added as metadata to newly produced datasets. Examples of the standard system metadata attributes could include type of workflow systems involved, starting time of the execution, input file names and sizes, logical working directory, next workflow system to be executed (if any), final logical destination on the data grid and so on.

Moving datasets to and from the data grid is achieved using the protocol defined by the data grid. Usually all data grid middleware provide a simple data transport mechanism. Since WVS is built into the trust domain of the data grid; data transfers can be handled very effectively.

### 2.4 Execution of workflow and post-processing

Workflows are integrated into the server through modules; these are independent functions that interface the workflow to the workflow system. The modules can be thought of as server building blocks. Each new type of workflow system can be integrated through a specific module that understands the required job description protocol. The modules enable high levels of customizability with the integrated data grid and workflow system. Modules can be easily developed in accordance to the requirement of each site. Each module can simply execute the workflow according to the configuration submitted by the user, or it can collect metadata, replicate certain or all of the datasets that are produced, and create derived data products before archiving in accordance with user-specified requirements. Administrators can define modules that execute specific custom scripts or checks before or after executing the specified workflow, i.e. modules can be also used to implement policy control-points specific to that installation. For example before executing a Makeflow workflow, administrators can have scripts check the consistency of the workflow configuration file to see if all datasets are accessible. Another use-case could be a module that requires execution of certain other pre-existing modules in a particular sequence to achieve the required functionality for the site.

Modularity at the level of workflow execution not only eases integration of new workflow systems into the server but also provides the administrator a tighter control over the workflows that are being executed. Administrators and developers will be able to decide the kind of variations that can be used in executing the workflow; any workflow system can easily be made inaccessible, by disabling the control module.

System metadata attributes available at the end of execution can be collected within the workflow server. Examples include the total execution time, the total size of datasets produced, or any metadata provided by the users for association with output files. Clean up of temporary files and datasets that are archived can also be carried out automatically.

## 3. Implementation with iRODS data grid

The past year has seen significant increase in the use of iRODS as a data grid throughout the world. This has generated many scenarios that provide compelling reasons to integrate multiple types of workflow systems with iRODS. The basic requirement is the automation of the execution of an external workflow in tandem with the data grid automation of administrative workflows managed within iRODS. In our implementation, clients submit requests for workflow execution through iRODS rules. We also use the rules and micro-services of iRODS to establish communication between the data grid and an external WVS. This approach is highly extensible, enables use of a wide variety of data management interfaces, and enables the automation of derived data product generation. Though many intricacies are yet to be realized, initial integrations with Pegasus, Makeflow and Taverna have provided satisfactory results.

### 3.1 Micro-services and rules

iRODS manages shared collections assembled from data distributed across multiple storage locations through use of a logical name space [7]. All operations within iRODS are performed based upon the logical name space. Every task carried out in iRODS is accomplished by executing sets of micro-services, which provides customizability and extensibility. Micro-services are well-defined functions that are created to realize a specific task; they are the building blocks of the system. Set of micro-services can be chained together to achieve a complex workflow. Micro-services are further classified into system micro-services and module micro-services. System micro-services are chained under administrative control to carry out essential system tasks, such as creation and deletion of users, creation of new resources and so on. Module micro-services allow customization of the operations performed at each storage installation to meet the needs of users. Examples include, extraction of data points from specific image datasets when uploaded, tweeting messages upon occurrence of specific events and so on. We utilize the ability of iRODS to control the execution of micro-services to develop a micro-service that is capable communicating with the WVS installation.

The execution of these micro-services is completely controlled by rules, which are defined by administrators and users. In the case of WVS, rules can be considered as the client interface through which users submit requests for execution of workflows. Each rule will specify the WVS server host, the port on which it is accepting connections, the path to the location of a client side configuration file, which is stored within the data grid, and the micro-service that will interact with the WVS installation.

## 3.2 Client design and configuration

The micro-service submitting the workflow can be thought of as the client controlling interactions with the WVS. The functioning of the client depends on the module micro-service, “msiwfSubmit”, a client configuration file and a rule that triggers the submission. The micro-service requires a host address and port number for the WVS and the location of a configuration file, which would contain all of the required details to execute the workflow through a rule. A micro-service would then contact the WVS over standard TCP/IP sockets and provide the current context structure, including the path to the configuration file. The context would also contain all of the user details, such as username, resource name, and destination collection path. A typical client configuration file is shown in figure 3.

```
WORKFLOW=MAKEFLOW
CONFIG=/tempZone/home/wfuser/test.makeflow
INPUT=/tempZone/home/wfuser/capitol.jpg
INPUT=/tempZone/home/wfuser/local.jpg
INPUT=/tempZone/home/wfuser/meta.jpg
DEST=/tempZone/home/wfuser/test_dest/
METADATA=NAME1=VAL1
METADATA=NAME2=VAL2
```

**Figure 3. Client configuration snippet**

In the above configuration file, “WORKFLOW” and “SUBMIT” are mandatory fields. If input files are required, they are also specified. If no destination is mentioned, the datasets generated after executing the workflow will be archived to a folder containing the configuration file. Any metadata that is mentioned here is attached to the generated datasets after they are archived.

## 3.3 Server design and configuration

There are four major routines that constitute the server core. They handle authentication, data transfer, metadata tagging and module execution, respectively. Authentication module is capable of connecting to a pre-defined iRODS installation similar to any other client executing under an iRODS administrative account. This enables the WVS to act as a proxy for an iRODS user for further transactions. At the operating system level, the process executing the workflow is tied to a virtual working directory. Within iRODS, the Rule Execution Information (REI) structure provides extensive information on the current state of the execution, both for system and user related attributes. This structure is preserved and is directly used in WVS to identify and keep track of the current context for all of the iRODS related information. This includes context information about the completion of the workflow that could be used globally, for managing complex interactions with other workflow systems.

The functioning of the server is controlled through a configuration file. This can be considered as an administrative interface for the initial version of the system. The configuration file contains information pertaining to each individual module. For example consider figure 4.

```
[MAKEFLOW]
path=/usr/local/cctools/redhat5/bin/makeflow
args= -T condor
[MAKEFLOW]
[MAKEFLOW1]
path=/usr/local/Makeflow/bin/makeflow
args=  -p 9876
[MAKEFLOW1]
#[KEPLER]
#path=path to kepler
#args=-t –P
#[KEPLER]
[PEGASUS]
path=/usr/local/Pegasus/Pegasus-plan
path_to_sites.xml = /usr/local/Pegasus/sites.xml
path_to_rc.data /usr/local/Pegasus/rc.data
path_to_tc.data = /usr/local/Pegasus/tc.data
[PEGASUS]
```

**Figure 4. WVS server configuration file snippet**

Module names in square braces act as delimiters to module specific information. They also act as pointers to the interfacing modules that will be invoked during execution. This design makes integration of new modules very easy. Each time a new module is added, the module name and the interfacing module function pointer have to be entered into a module map that will be referred to by the WVS at the time of execution. In addition all the information pertaining to the workflow system has to be entered into the server configuration file. The workflow system represented by the new module will be ready to use as soon as the interfacing module is compiled. Note this compilation is done only once for each new type of workflow system. The client also uses this module name to denote the type of the workflow system to be executed. This configuration file is read every time an interfacing module has to be called. Even though this increases the number of disk accesses, it provides dynamic re-configurability to the system and eliminates the need to restart the server.

After the interfacing module is executed successfully, WVS will initiate data transfer of newly generated datasets back to iRODS using the iRODS client API. Metadata will be tagged to these datasets once they are registered into the data grid. Module developers are provided with functions that can query any user metadata they wish to tag to these datasets.

## 4. Conclusion

This paper describes the design and implementation of a workflow virtualization mechanism that can be used to integrate, manage and orchestrate execution of workflows and archive the resulting large-scale data. The design implements a stand-alone, highly modular server that acts as an interface between the data grid and workflow systems. The detached hosting of such a server is vindicated by the fact that it improves the security of the system to a great extent. Our successful implementation of the concept with the data grid iRODS and workflow systems Makeflow and Pegasus, justifies the assertion that the integration of workflows and data grid trust environments can provide a generic solution to the use of distributed compute and storage resources.

## 5. Future work

The WVS presented here provides the desired workflow virtualization, but there are additional functionalities that have to be worked out. Federation with more than one data grid, of the same or different kind, will increase the usability of the system. Federation across multiple types of data grids would require modular extensions to the trust environment and efficient handling of interfacing protocols according to each grid's specifications. We will continue development of the WVS with the iRODS data grid as a benchmark. One of the major areas to be dealt with is to provide feedback to the users, in terms of status of workflow execution, logs, progress of execution, after the job is submitted. This can be easily achieved in iRODS by having a dedicated micro-service that can query WVS. A simple rule can provide access to such a service. We have discussed several authentication methodologies that can be implemented with WVS, but the preferred approach is not feasible on all systems. Hence an efficient, yet viable solution has to be developed. Further, the design has to be implemented and tested with other data grids and storage systems.

## 7. Acknowledgements

The research results in this paper and the iRODS technology development have been funded by NSF SDCI 0721400, "SDCI Data Improvement: Data Grids for Community Driven Applications" (2007-2010). We gratefully acknowledge the funding. We also thank the Numerical Relativity group at Center for Computation and Technology, Louisiana State University for providing the statistics about their simulations and data transfers and all other scientist collaborators who motivated the research described in the paper.

## 6. References

[1] HPCWire. (2010, *HPCWire:TeraGrid 2010 Keynote: The Physics of Black Holes with Cactus*. Available: http://www.hpcwire.com/features/TeraGrid-2010-Keynote-The-Physics-of-Black-Holes-with-Cactus-100463419.html

[2] *Ocean Observatory Initiative(OOI)*. Available: http://www.oceanleadership.org/programs-and-partnerships/ocean-observing/ooi/

[3] Teragrid. *Teragrid*. Available: www.teragrid.org

[4] L. O. N. Initiative. *LONI*. Available: www.loni.org

[5] U. o. N. Dame. *Makeflow*. Available: http://www.nd.edu/~ccl/software/makeflow/

[6] Douglas Thain and Christopher Moretti, "Abstractions for Cloud Computing with Condor," in *Cloud Computing and Software Services: Theory and Techniques*, S. A. a. M. Ilyas, Ed., ed: CRC Press, 2010, pp. 153-171.

[7] A. R. Reagan Moore, "White Paper: IRODS: Integrated Rule-Oriented Data System," 2008.

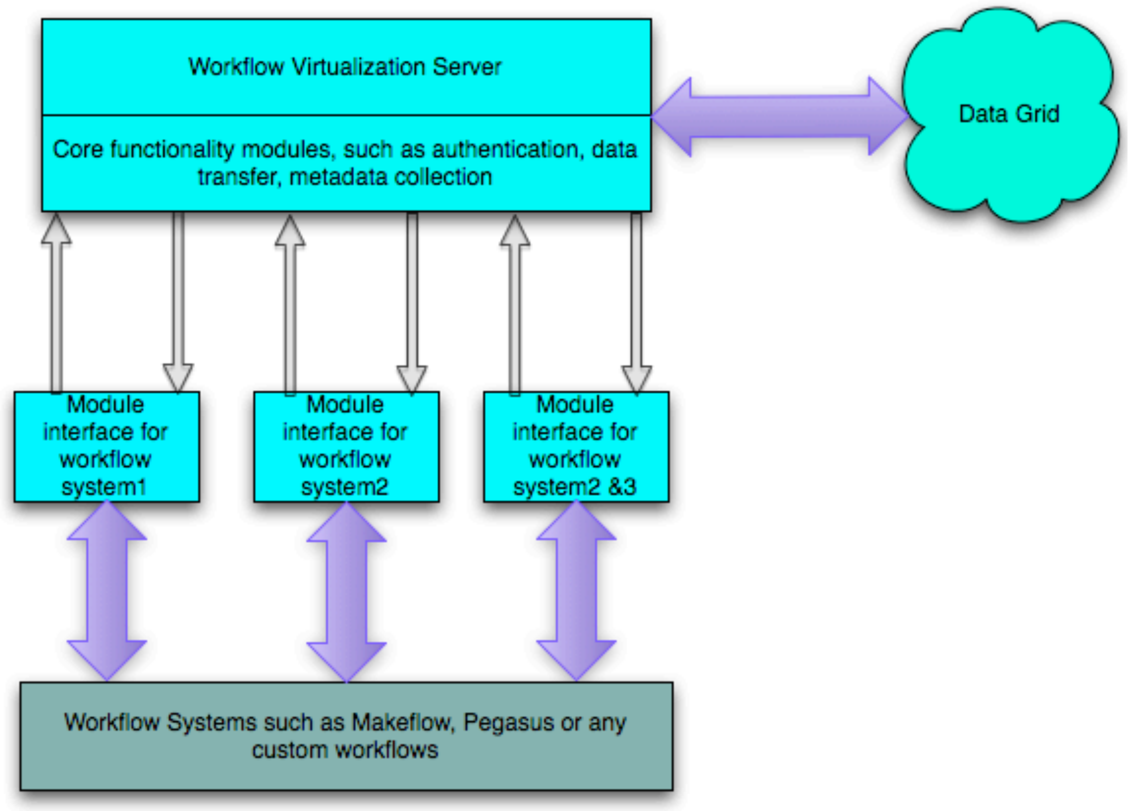
Workflow Virtualization Server
Core functionality modules, such as authentication, data transfer, metadata collection
Data Grid
Module interface for workflow system1
Module interface for workflow system2
Module interface for workflow system2 &3
Workflow Systems such as Makeflow, Pegasus or any custom workflows

**Figure 1. Workflow virtualization server architecture**

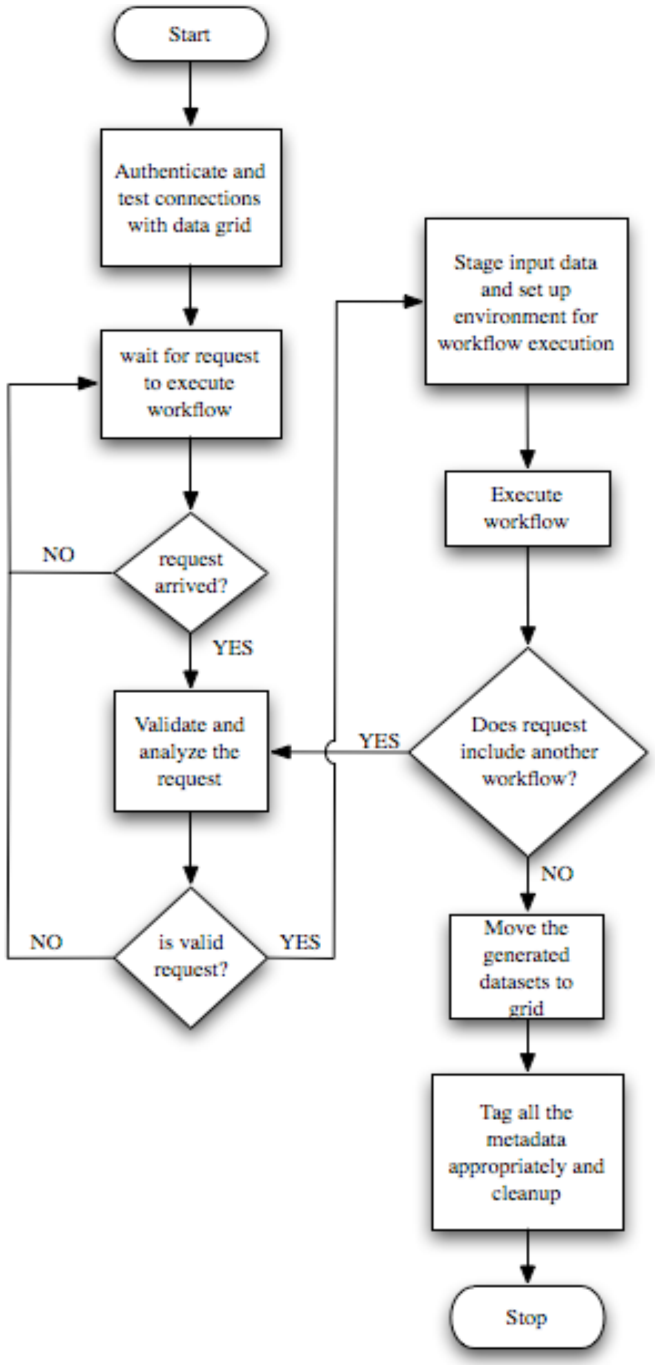
Start
Authenticate and test connections with data grid
wait for request to execute workflow
request arrived?
NO
YES
Validate and analyze the request
is valid request?
NO
YES
Stage input data and set up environment for workflow execution
Execute workflow
Does request include another workflow?
YES
NO
Move the generated datasets to grid
Tag all the metadata appropriately and cleanup
Stop

**Figure 2. WVS flowchart**