# LQCD Workflow Execution Framework:
## Models, Provenance, and Fault-Tolerance

Luciano Piccoli[1,3], Abhishek Dubey[2], James N. Simone[3], James B. Kowalkowski[3]

[1]*Illinois Institute of Technology, Chicago, IL, USA*

[2]*Vanderbilt University, Nashville, TN, USA*

[3]*Fermi National Accelerator Laboratory, Batavia, IL, USA*

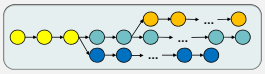
## Motivation

Lattice Quantum Chromo Dynamics (LQCD) is an identified grand challenge scientific application employing large-scale numerical calculations to extract predictions of the standard model of high-energy physics.

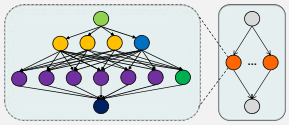
**Properties:**

- Tens of users running hundreds of complex workflows (campaigns) concurrently.
- Campaigns are composed of identical embarrassingly parallel sub-workflows running on different sets of inputs.
- Campaign running time may span several months.
- Campaigns execute thousands of MPI jobs.
- Large input and output files, from hundreds of MBytes to a several GigaBytes in size.
- Campaigns run on dedicated clusters with Infiniband and Myrinet (qcd, kaon and pion at Fermilab).

**Typical workflows:**



Configuration generation



Two-point analysis

Diagnosing job problems and faults leading to eventual failures in this complex environment is difficult, specifically when the success of whole workflow might be affected by a single job failure.

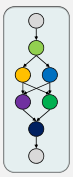
## Approach

Create a reliable execution framework that is model-based and hierarchical. The execution framework encompasses workflow specification, data provenance, execution tracking and online monitoring of each workflow task. Workflow tasks are plugins to this framework that we call participants. The system supports instantiated parameterized workflows. The relationships among participants are described using types and parameters. The dependencies in concrete workflow are translated based on the types and parameter values.

### Workflow Specification

Workflows are specified by users in parameterized abstract views. Abstract views define generic workflows. For the configuration generation workflow, where all participants are the same except for the arguments, the abstract workflow can be defined as single participant within a loop.

Abstract workflows are better illustrated by two-point analysis campaigns. The number of participant instances on concrete executable workflows depend on user specified parameters.

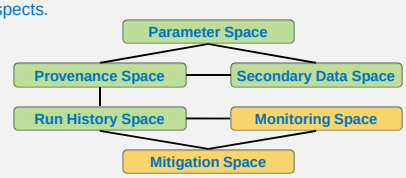


The concrete workflow for a combination of three heavy quarks and six particle masses is shown above. The number of sub-workflows on the left side is equivalent to the number of configuration files used for the analysis.

We use the Pegasus workflow management system to specify the abstract workflows and perform the mapping to concrete workflows. The latter are defined as Condor DAGMan.

## Data Model and Provenance

Workflow managed data are divided into four spaces for tracking and to minimizing cross-references. The two additional spaces are defined for the cluster reliability system, being used for monitoring and fault-tolerance aspects.



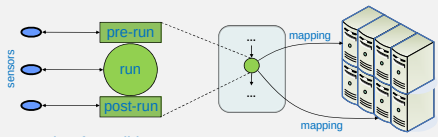

- Parameter Space: Archives all parameters used as input for a workflow, including physics parameters (e.g. quark masses), algorithmic parameters (e.g. convergence criteria) and execution parameters (e.g. number of nodes used).
- Provenance Space: Keeps information regarding inputs and outputs for each workflow participant.
- Secondary Data Space: Used for storing secondary information generated by workflow participants. (e.g. plaquette values from each iteration).
- Run History Space: Archives detailed information about execution, including algorithm versions used and outputs generated.
- Monitoring Space: Contains monitoring sensor configuration (CPU, memory, disk space) and historical values.
- Mitigation Space: Defines the failure recovery strategies and compound temporal properties used by participants.

## Execution Tracking

As participants belonging to a workflow are mapped onto machines and executed, periodic and on-demand monitoring of vital health parameters on allocated nodes is enabled according to pre-specified rules. These rules define conditions that must be true pre-execution, during execution and post-execution.

Conditions are used to define **preconditions**, **postconditions** and **invariants**. Pre and postconditions are evaluated before and after a participant is started. Invariants are monitored during the participant life time.

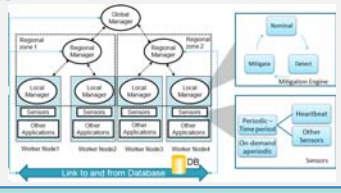
Conditions refer to sensor values, which are periodically checked. Values are verified according to the scope defined by the condition, insuring the monitored values fall within the expected range.




Example of conditions:

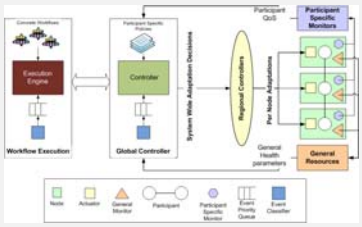
- Precondition: dCache pool manager (pm) must be available before the participant is started. *H(pm)(t)>0.*
- Postcondition: generated output file. *H(file) = 1.*
- Invariant: host is available. *H(host) = 1.*

Monitored sensor values for running participant are propagated upwards through the reflex and healing architecture, which consist of hierarchical network of decentralized fault management entities - the reflex engines.



## Integration

The integrated workflow, monitoring and mitigation system is shown below. The left side contains the workflow execution engine, which schedules participants as dependencies are met. Components on the right side are part of the cluster reliability system.



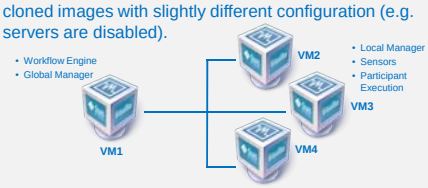
The workflow execution engine provides an interface for submitting concrete workflows. Multiple concrete workflows can be handled by multiple execution engine threads.

As participants are ready to run the workflow engine contacts the global manager. Events are exchanged asynchronously between the global manager and the workflow engine. The centralized controller processes the events and then sends required commands using events to the local managers and regional managers. Information regarding the participant conditions are used to start participant specific monitors in the worker nodes.

## Virtual Setup

In order to avoid the use of actual cluster resources and competing with production runs, we have prepared a virtual environment for testing the system prototype.

The virtual cluster uses Sun's Virtual Box running Ubuntu Linux. The workflow engine, the global manager, and the database run on VM1 head node. The worker nodes run cloned images with slightly different configuration (e.g. servers are disabled).

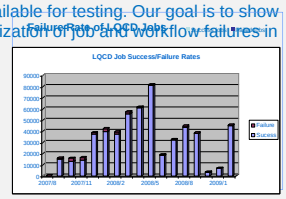



Virtual machines communicate via a private virtual network. Tests using this environment currently run on a dual quad-core host machine with Scientific Linux 5.2.

## Current and Future Work

The combination of the cluster reliability with the workflow execution framework is currently under development on the virtual setup shown above. This environment allows full control over the resources, including injection of failures and observation of system recovery behavior.

Currently only simple workflows, such as the configuration generation, are available for testing. Our goal is to show the potential minimization of job and workflow failures in the presence of injected faults. The graph on the right illustrates actual job failure rates collected from the LQCD clusters.




This work was supported in part by Fermi National Accelerator Laboratory, operated by Fermi Research Alliance, LLC, under contract No. DE-AC02-07CH11359 with the United States Department of Energy (DoE), and by DoE SciDAC program under the contract No. DOE, DE-FC02-06 ER41442.