

# LQCD Workflow Management Proposal

Jim Kowalkowski, Luciano Piccoli

## 1 Introduction

Data processing within the LQCD project is carried out as *analysis campaigns*. Any analysis campaign consists of an input *dataset* and a set of interdependent processing steps that can be expressed as a directed acyclic graph (DAG). This DAG can be considered to be the *workflow* specification for an analysis campaign. Given the size and complexity of the LQCD compute cluster, it is now necessary to more closely manage these workflow specifications, and use them to automate many aspects of executing an analysis campaign.

The purpose of this paper is to define a subsystem that allows workflow to be specified in domain-specific way and later be turned into a set of instructions that can be carried out or executed on a compute cluster. Execution includes configuration, submission, progress tracking, and accounting of an analysis campaign. It also includes input staging and storage of results. This paper contains a description of the problem and a working set of requirements for this subsystem. It also includes a description of how the problem is currently being addressed and suggestions on how it might be solved within this new subsystem.

### 1.1 Scope

We are currently assuming that the underlying computing infrastructure has a batch job submission system with a scheduler and that we are able to communicate with these systems. We are also assuming that a software system will exist for monitoring all interesting aspects of a job and that that system will provide a framework for reacting to recognized important conditions.

### 1.2 Rationale

The task of monitoring an ongoing campaign becomes too large for a person to do once a compute cluster becomes very large. As the number of concurrently executing tasks becomes very large, the person monitoring the jobs becomes the bottleneck in the system.

Diagnosing job problems and failures, and taking corrective actions is a highly repetitive and algorithmic task. Much of this work is better suited for a computer than for a person.

For large installations, tracking progress and knowing what has been done and what needs to be done is necessary to optimize resource utilization, to meet deadlines, and to do planning for the future.

### 1.3 Terminology

*Configuration*: A four dimensional wave function. It represents a snapshot of space-time. It lives in a file that is about 2GB (probably will grow to about 8GB). Each file can be thought of as a single event in a large ensemble.

*Dataset*: A collection of configuration files (the ensemble) that meet some criteria.

*U-file*: A number identifying a configuration file.

*Propagator*: A function that evolves a configuration from some initial state to a final state

*Workflow specification*: A formal description of dependencies and parameters amongst processing steps in an analysis campaign.

*Configuration generation*: The process of creating configurations.

*Analysis Campaign*: Applying everything described in a workflow to all configurations in a dataset.

### 1.4 Basic feature overview

Some basic features of this subsystem include:

- Definition of a workflow language that it useful for describing an analysis campaign.
- A set of tools that aid in the writing and validation of documents that conform to the language.
- Management (e.g. storage and retrieval) of workflow specifications.
- A set of tools that transform the workflow specifications into instruction that can be scheduled (e.g. Maui) and executed (e.g. PBS, Condor) within the various computing environments.
- Monitoring the progress of the activities and progress of work being carried out within a workflow and store metrics related to jobs.
- A set of components that are capable of reacting (taking action in response) to progress reports and measurement data from the monitoring system.
- Lifetime management of temporary or intermediate results in order to maximize resource utilization (e.g. disk space, network bandwidth, memory, CPU).
- A set of planning tools to help with the scheduling of analysis campaigns.

## 2 An Example Campaign

### 2.1 What needs to be done

An example analysis campaign description will be used to demonstrate the type of processing that occurs on the LQCD cluster. Below is the definition of 2-pt analysis for processing one configuration file. The ovals represent work being done in the form of jobs run on demand or under a batch system. The horizontal bars represent fork (parallel execution starting) or join (completion synchronization) conditions. The rectangles represent products (files in this case).

## LQCD Workflow Management Proposal

There are approximately 600 configuration files that live on RAID disk, on tape, or somewhere within Dcache. For many analysis campaigns, the processing of one configuration file is independent of any other configuration file, so many can be processed at the same time.

The first step in processing one configuration file is to make the file available to all the jobs that must access it directly. This step is likely to be done outside the domain of the job scheduler/batch queuing system. The amount of work done here depends on how much of the data is already cached in acceptable places and how much must be transferred from long-term storage or from remote locations. It may also depend on the architecture of the cluster.

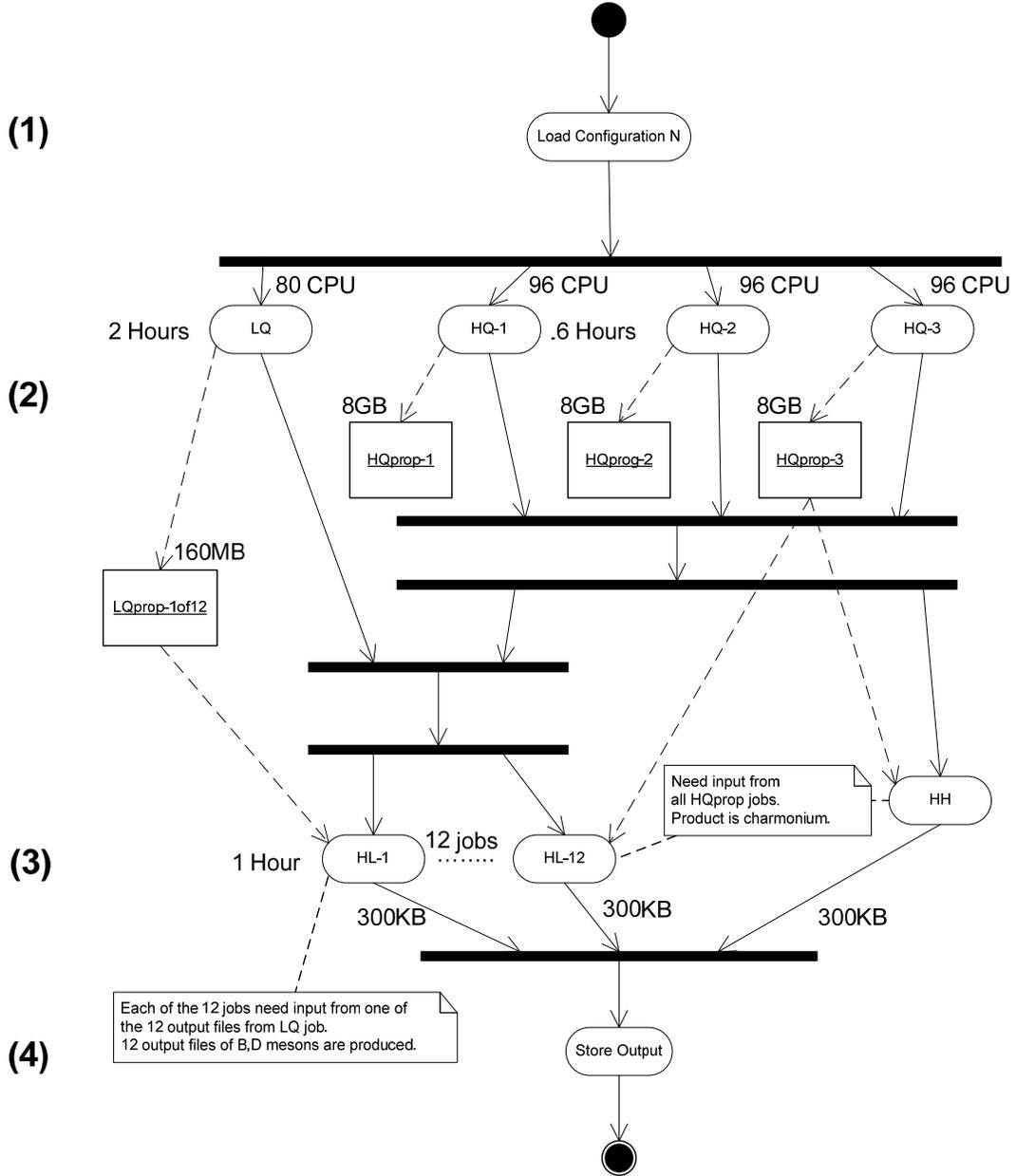
Step two is to run all the propagators. The input of these jobs is a configuration file and a file of job parameters. The output of this stage is large intermediate result files that currently live in a large LRU-based disk buffer. Gathering of these intermediate results can use significant disk and network bandwidth.

The last step uses the intermediate results to form summaries useful in a final analysis.

Executing the workflow in the diagram for a given configuration file generates one row in the above results table (example). Each cell holds the results from the HL/HH jobs in the workflow. The goal is to fill out the entire table by running the workflow on each configuration file. Later analysis is typically carried out using columns of this table i.e. the same thing being calculated using each configuration and averaged across all the configurations for a final measurement.

Currently the programs that run in the various stages three different software environments or software frameworks: MILC (LQ jobs), FermiQCD (HQ jobs), and Canapy (HH/HL jobs).

Example workflow: 2-pt analysis for processing configuration N



CONFIG N	D(0)	D(P <sub>1</sub> )	D(P <sub>2</sub> )	D(P <sub>3</sub> )	D(P <sub>4</sub> )	D(P <sub>5</sub> )	D(P <sub>6</sub> )	D(P <sub>7</sub> )	D(P <sub>8</sub> )	D(P <sub>9</sub> )
198										
199										
200										
201										
202										

## **2.2 *Some interesting aspects***

How should staging of input configurations be overlapped with processing of them?

How many configurations can be processed at the same time?

How is process tracked so that we know how to restart in case of failure and so we know that all is well the procedure?

How are the job parameter files stored and retrieved and what information needs to be recorded to fully describe the job, its inputs, and its outputs?

How long will it take for the job to complete? How is the time a job takes affected by changing the number of CPUs used to solve the problem? Many of the current values were determined by trial and error.

## **2.3 *How the process is currently managed***

To be filled in.

# **3 Requirements**

## **3.1 *Workflow specification and manipulation tools***

Workflow specification documents: what tools are necessary to store, retrieve, and manipulate them?

## **3.2 *Execution***

What is required to convert workflow documents into instructions and to execute them?

## **3.3 *History***

What do we need to record about analysis campaigns that are planned, complete, or executing? In what ways will the information be stored and accessed?

## **3.4 *Performance, status, and recovery***

What does it mean to measure the performance and status of jobs within a campaign and to report their progress? What types of recovery actions need to be carried out when things fail?

## **3.5 *Resource Utilization***

What are rules for managing intermediate output?

## **3.6 *Planning***

What are some of the questions that will be asked of a campaign planning system? What information is necessary to answer the questions?

### **3.7 *Unclassified at this time***

Enable users to describe the dependency relations among the steps of an analysis campaign.

The framework should make “runnable” job steps as soon as their dependencies are successfully completed. Users should be able to control the maximum number of running job steps.

The framework should pre-stage input data files and store output data files.

The framework should be able to plan the execution order in an analysis campaign so that delays due to data access latencies are minimized.

Maintain a persistent history for the state of an analysis campaign. Use the saved state to resume from the last successful step in case of an error or when restarting a campaign at a later date.

Provide a framework to monitor the performance, error rates and resource requirements for a campaign.

Provide a uniform framework for interacting with batch systems, grid portals and UNIX system processes.

Describe how this subsystem will be used by outside actors. Include constraints imposed by outside systems and any other important factors governing its design and implementation.

### **3.8 *Actors***

Describe here the set of characters that will be interacting with this system. This list should not include internal actors.

### **3.9 *The Major Inputs and Output***

Describe what the system needs to do its work and what it produces.

### **3.10 *Behavioral requirements (use cases)***

List the steps necessary to perform each important task associated with this system. Do not include any implementation details here, only brief statements of actions and responses for each one. Each use case should only cover one specific task. Below is a template for use cases.

#### **3.10.1 Task name**

Task	Name of this task
Level	Summary/user goal/sub-function
Goal	Stated as a short active verb phrase
Actor	Who does this

Trigger	Why this is happening
Preconditions	Things which must exist before the use case can start, and any particular state the overall system must be in to allow performance of this case
Post-conditions	New state which exists at successful completion of use case
Description	Steps (numbered) to complete the task. This should include a narrative description of the manageable series of steps that make up the use case
Nonstandard Flow	Exceptions (error conditions) to the standard flow or alternative success routes.
Comments	Link to other information regarding with use case

### **3.11 Constraints**

List any additional non-functional requirements here or reference them here. Examples include known external interfaces or protocols or performance constraints.

### **3.12 Failure modes**

List problems that this system may encounter.

## **4 Architectural Overview**

This section can contain a series of diagram illustrating parts of the subsystem and their relationship with other parts.

Do all the following subsections make sense? Does it make sense to distinguish between a function or role and components?

### **4.1 Roles**

The roles or functional units are defined and described in this section.

### **4.2 Functional unit or Component block diagram**

Include here a picture of functional units within the system and their relationship to one another.

### **4.3 Physical unit block diagram**

Show here the known hardware configuration that is to be built or is available for use.

#### **4.4 Deployment scenario**

How do functional units and components map to physical devices?

### **5 Component Interfaces**

Expand the interfaces of the components shown in the previous section. Include relationships to other components here.

### **6 Protocols**

Describe known elements of any protocol involved in data exchanges, external or internal to this subsystem, and the types of messages or data that may be exchanged. **This is distinguished from component interfaces right now – should it be?**

Show important invocation or message exchange timing sequences here. Show what parts of the interfaces are used by other components.

### **7 Discussion**

This section captures discussions and information that lead to the current architecture view and component organization.

#### **7.1 Decisions and Choices**

Other solutions that were considered and rejected should be briefly summarized here along with arguments for and against them. The purpose of doing this is so old arguments do not continually resurface.

#### **7.2 Rationale**

Why the current architecture and component interfaces are appropriate for the problem.

#### **7.3 Implications resulting from Choices**

Additional constraints that are imposed on the whole system or this system as a result of choices made here.

#### **7.4 Resulting rules**

Include all things that must be true in the system and rules that must be followed while the system is in operation. An example is that one worker node will only be assigned to one partition.

#### **7.5 Constraints imposed on other systems**

List what constraints this system imposed on other systems.

## **8 Testing considerations**

Explain any load testing that must be performed to evaluate the performance of this system as a whole or parts within it. Suggestions for how to test (verification and validation) this system should be included. Ways of evaluating the performance of this system should be included here.