The LQCD Workflow Experience:

What Have We Learned

Luciano Piccoli[1,2], Xian-He Sun[1,2], James N. Simone[2], Donald J. Holmgren[2], Hui Jin[1,2], James B. Kowalkowski[2], Nirmal Seenu[2], Amitoj G. Singh[2]

[1]*Illinois Institute of Technology, Chicago, IL, USA 60616*

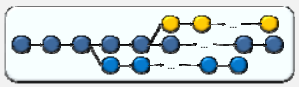
[2]*Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL, USA 60510*

Introduction

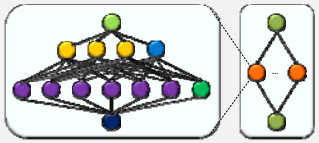
- QCD (Quantum Chromo Dynamics): theory of the strong force that describes the binding of quarks by gluons to make particles such as neutrons and protons.
- LQCD (Lattice QCD): computation and data intensive numerical simulation of QCD using a discrete space-time lattice. Its calculations allow us to understand the results of particle and nuclear physics experiments in terms of QCD. Representative of large scale scientific computing.

The Environment

- Tens of users running several complex workflows (campaigns) concurrently.
- Campaigns are composed of identical embarrassingly parallel sub-workflows running on distinct inputs.
- Campaign running time may span several months.
- Hundreds of running campaigns.
- Typical workflows:
  - Configuration Generation workflow.



  - Two-point analysis campaign sub-workflow.



- Tasks composed of parallel MPI jobs requiring tens to hundreds of processors.
- Dedicated clusters with Infiniband and Myrinet (qcd, kaon and pion at Fermilab).
- Large input and output files, from hundreds of MBytes to a few GigaBytes in size.

Challenges

- Create an effective model and a set of tools to deal with LQCD workflows.
- Clear definition of responsibilities for each participant.
- The experience with LQCD gives us insights to understand where the boundaries should be drawn between workflow management systems, web services, task schedulers and other subsystems.

Requirements

- Templates: recipe for solving a LQCD problem with parameterized physics values (e.g. particle masses)
- Instances: a template with validated physics parameters
- Execution: schedule each workflow task (participant) upon resolution of control and data dependencies, by mapping it to available resources
- Monitoring: ability to monitor the current status of a workflow instance
- History: for accounting and prediction for future workflow executions
- Multiple Instances: support multiple campaign execution
- Stage in files: ability to pre-fetch workflow input files
- Fault Tolerance: recovery from hardware and software failures along a workflow execution
- Management of intermediate files: track generated files, optimize file re-usage among workflows
- Campaign execution: ability to execute long-term workflows composed of identical embarrassingly parallel sub-workflows running on distinct input configurations
- Campaign dispatching: submission of campaigns (workflow instances) to the system. New campaigns may extend ongoing campaigns by adding new inputs, participants and dependencies.

Experience with Existing Systems

- Variety of systems targeting scientific workflows available (e.g. Askalon, Swift, Kepler and Triana).
- All systems provide integration with the Grid.
- Very active research area.

- Lack of data provenance support, which is critical for most scientific workflows.
- Some systems require moderate to advanced programming knowledge to create workflows.
- Steep learning curve for domain scientists, difficult to migrate from original batch scripts into workflow specifications.
- Lack of common language between workflow systems
- Abstraction of physics parameters from workflow. template is not straight forward (sometimes not possible).
- Limited quality of service features.
- No complete solution available yet.

Scientific Workflows vs. Service-Oriented Workflows

- **Service-oriented architecture:**
  - Well defined and modularized architecture.
  - Decouples service providers and users.
- **Service-oriented workflows:**
  - Business-oriented workflows are usually implemented as service-oriented workflows.
- **Service-oriented workflows:**
  - Participants are black-boxes represented by remote services.
  - Participants can be easily replaced/replicated by services (as long as the interface remains the same).
  - Fault-tolerance at participant level.
- **LQCD workflows:**
  - Scientific applications requiring dedicated/predefined hardware.
  - Software fine tuned for specific platforms.
  - Large input and output files, including intermediate results.
  - Need for data provenance.

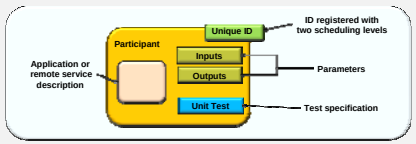
Scientific Workflows vs. Conventional Batch Scheduler

- **Batch scheduling:**
  - Independent jobs.
  - Primitive support for job dependencies through digraphs.
  - No fault tolerance.
- **Scientific workflows:**
  - Control and data dependencies between jobs define execution order.
  - The result of one job could determine further execution of the workflow.
  - Each job instance could require tightly coupled parallel execution.
  - Number of jobs and inputs may be determined by the outputs generated by previous jobs.

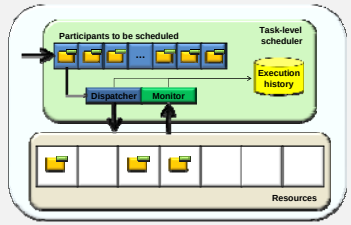
Two-Level Workflow Scheduling

Service-level scheduling and task-level scheduling, where service-level scheduling supports both control and data dependency.

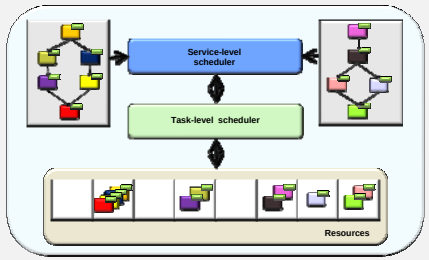
- **Participant**



- **Task-level scheduling (participant-level)**
  - Support execution of participants from multiple workflows: accept participant submissions from multiple workflow instances and execute them.
  - Monitor execution: of uniquely identified participants and report failures to the service-level scheduler.
  - Record execution times: keep records of execution times for participants, which can be used for predictions and accounting.
- **Task-level scheduling (participant-level)**
  - Estimate execution time: based on the recorded history and cluster status the task-level scheduler can provide the service-level scheduler with data for Quality of Service purposes.
  - Resource reservation: for participants based on data and control dependencies from the workflow.



- **Service-level scheduling (workflow-level)**
  - Track dependencies: is the basic function of a workflow system; it must enforce control and data dependencies of the workflow instances.
  - Submit participants: as dependencies get resolved participants are submitted to the task-level scheduler for execution.
  - Estimation of workflow run time: based on participants run time estimates from the task-level scheduler, report the workflow instance expected run time.



Conclusion

- Current workflow systems do *not* support most LQCD requirements.
- Systems *can* meet requirements, provided that the underlying architecture is modularized and expandable.

Future Work

- Prototype the two-level workflow proposal by extending current systems.
  - Can currently available languages adequately express the LQCD workflows?
- Meet requirements posed by LQCD workflow problems.
- Apply solution to similar workflow problems.

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.