

Reliability and Workflow projects

Jim Kowalkowski

Collaboration tools

- We use a wiki at <http://whcdf03.fnal.gov/lqcd> to maintain documentation and meeting minutes concerning both the projects discussed in this talk. Feel free to look around (you will need to get the access password from me).

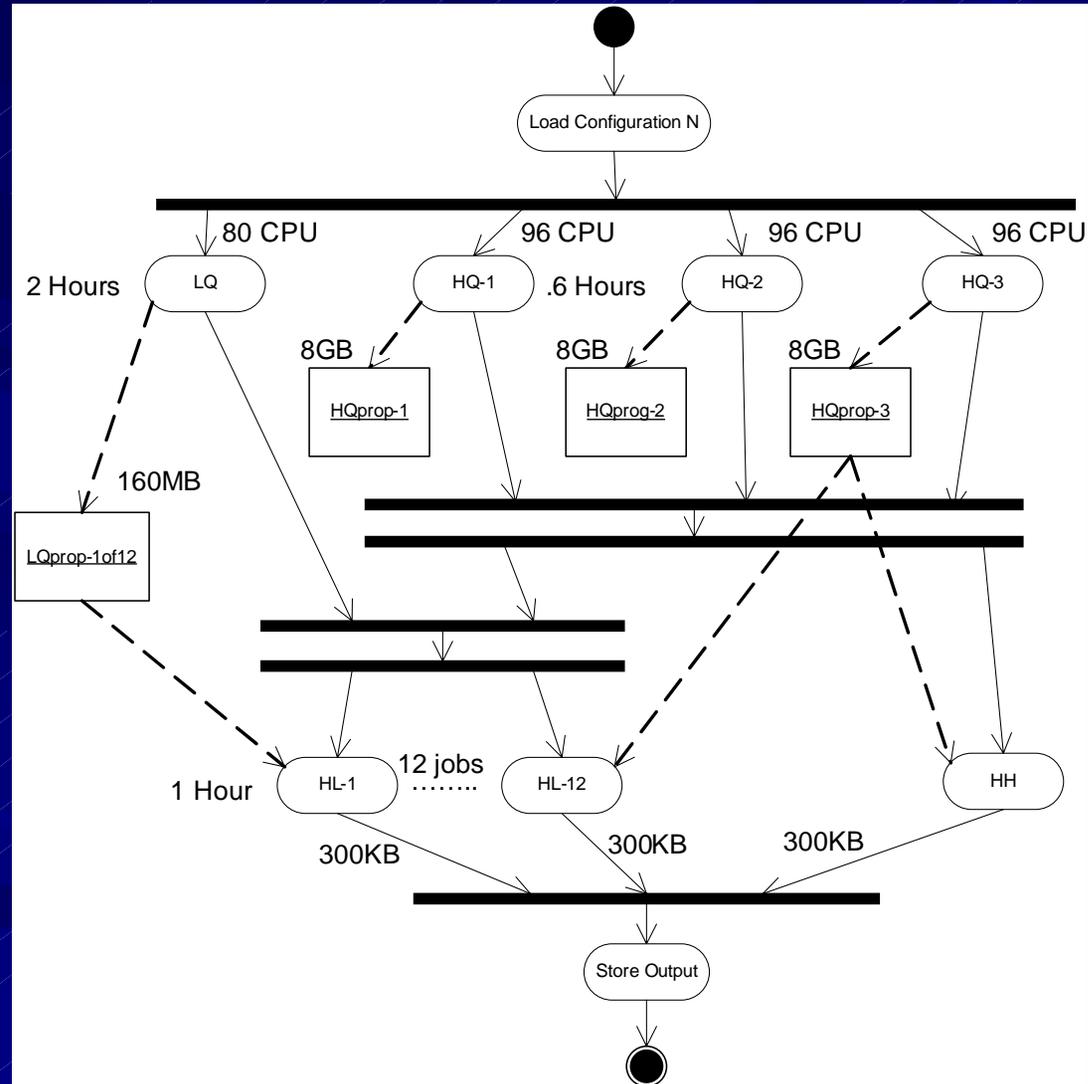
Workflow

- IIT funded through Scidac-II
- Main Participants
 - Luciano Piccoli (Fermilab & IIT)
 - Xian-He Sun (IIT & Fermilab ½ time)

Purpose and Goals

- Purpose is to create a system for
 - Designing *workflows* that comprise *campaigns*,
 - Instantiating these *workflows* with files and physics parameters,
 - And executing the instantiated *workflows* on LQCD resources
- Some of the Goals
 - Monitoring and keep execution statistics
 - Interact with provenance tracking
 - Detect and record job failure
 - Support recovery/restart of applications
 - Reuse as many existing tools as possible
- Definitions
 - ***Participant***: An object that transforms inputs into outputs. Examples include dCache, PBS, user applications, and shell scripts.
 - ***Workflow***: Procedures whereby data and control are passed among participants according to a defined set of rules (e.g. dependencies) to achieve a specific goal.
 - ***Campaign***: A coordinated set of calculations aimed at determining a set of specific physics quantities. It can be composed of many workflows.

2pt Campaign working example



Current Activities (two paths)

■ Product evaluation:

- Triana, Kepler, JBPM (graphical tool based)
- VDS, Askalon (language based)
- Organizing December workshop with Mike Wilde and Ian Foster from Argonne (let us know if you are interested in attending)

Defining terminology and requirements

- Identified some key workflow abstractions
 - Template (a pattern or description of how something is done)
 - Ideal (assumes unlimited resources)
 - Instantiated (particular input files and parameters applied by user)
 - Runnable (cluster resource usage policies applied by system)

■ Will make small, demonstrable steps towards full

Project Milestones

■ Version 1 (Jan/07)

- Extend a workflow system
 - Integration with current structure,
 - Substitute for current perl runfiles and bash scripts
- Definition of interface with cluster reliability

■ Version 2 (May/07)

- Ready to receive data from cluster reliability project
- Respond to faults
 - Workflow rescheduling
 - Restart jobs (from last completed milestone)

■ Version 3 (Aug/07)

- Campaign monitoring
- Maintain statistics on campaign execution

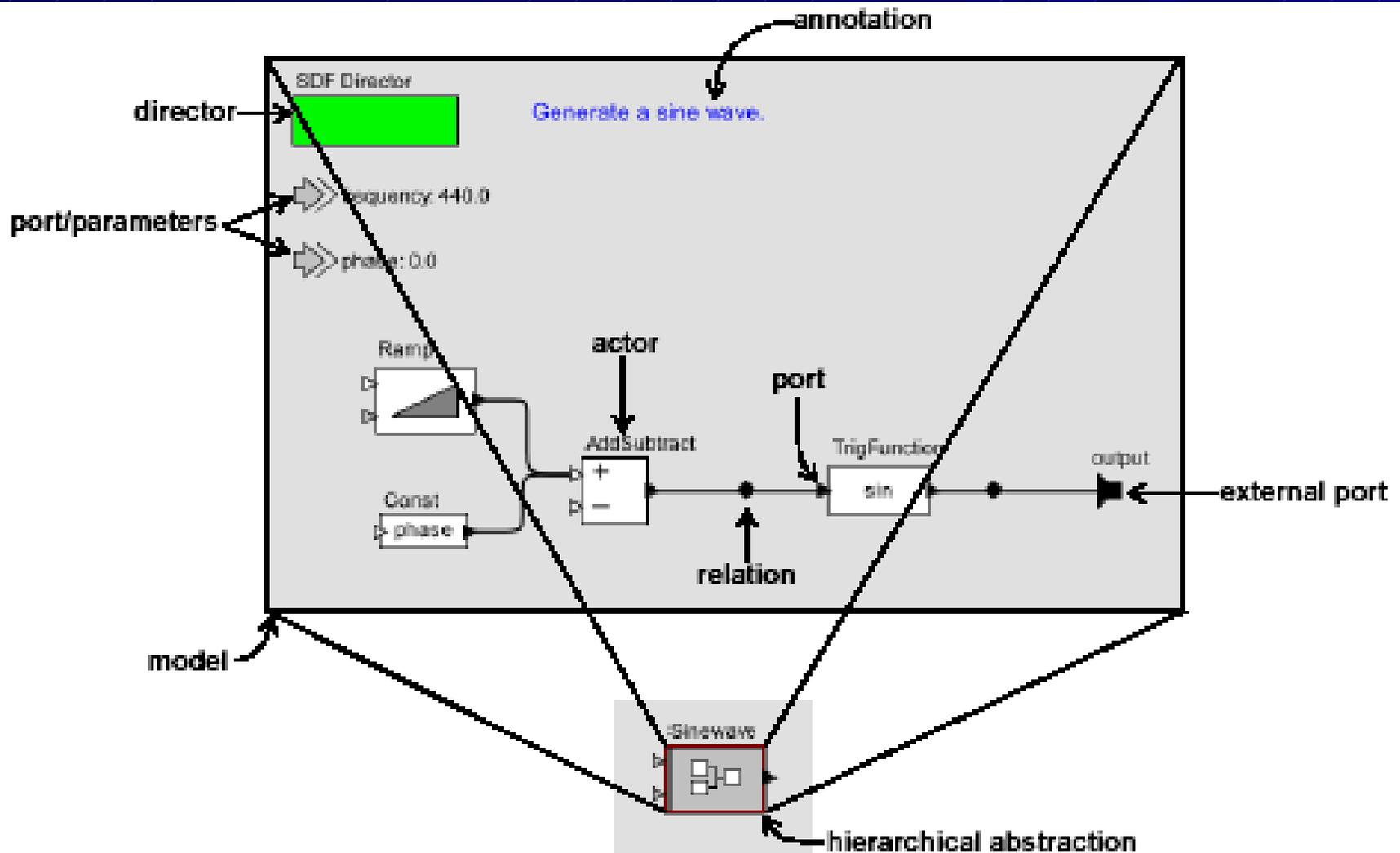
■ Version 4 (Nov/07)

- Management of intermediate files
- Keep information on data provenance

■ Version 5 (Mar/08)

- File pre-fetching
- Quality of service features (e.g. workflow deadline)

Kepler Actor-Oriented SWF



Cluster Reliability

- Vanderbilt ISIS Institute funded by SciDac—
II
- Main Participants
 - Ted Bapty
 - Sandeep Neema
 - Bunch of grad students

Purpose

- Put together a set of software tools that allows us to easily create “pluggable” components that
 - Monitor our clusters (hardware, utilization, errors, jobs, etc.)
 - Recognizes anomalous conditions (inference rules, model comparison, probabilities)
 - Take actions to correct or alleviate the problems (triggered by other components)

Goals

- Increase the availability, utilization, and reliability of the computing clusters
- Reduce the time it takes to diagnose problems
- Reduce the administrative workload associated with operating the clusters
- Automate routine administration tasks
- Monitor the use of the clusters
- Allow other systems and user scripts to easily interact with this system

Current Activities

- Underlying framework selection
 - Establish criteria and test scenario
 - Further requirements gathering
 - Product survey and evaluation (*OpenNMS, Aware*)
- Instrument current system to better understand needs (such as *data collection, storage, and organization*) and provide useful features now
 - PBS jobs database (*preload complete, active collection starts today*)
 - Standardization of job return codes and application message logging
 - Temperature/fan speed data collection next

Some basic features include:

- Identifying types of information that must be collected or communicated,
- APIs for communicating information and the creation of monitors and reactors,
- Popular programming language bindings,
- An environment for writing, testing, and releasing new reactors and monitors,
- A basic set of problems that must be addressed, the monitors, reactors and data that they will communicate,
- Recording of monitoring information and actions taken,
- Administration tools that allow for single-point release distribution and installation, and control of the runtime environment,
- A configuration system that allows for uniform parameter setting and allows for tuning to adjust the performance impact on the system.

ISIS Goals

- *Monitor the health of the system* leveraging existing tools and standards.
- *Closely monitor performance and job status*, and work together with the workflow subsystem, ensuring good progress.
- It should be *coupled to the application*. Mitigation actions depend on the properties of the application and its overall workflow.
- It should be *integrated with workflow planning*, to allow for resource optimization. This will include interactions with real-time scheduling systems.
- **Planned deliverables**
 - Customizable Monitoring & Control framework.
 - Mitigation engine.
 - Monitoring and mitigation design tool.
 - Monitoring and mitigation system generator
- **Research statement:** In this project we will address the critical difficulties in achieving a fault mitigation framework for a large cluster, which is configurable, and strives to minimally affect the performance of the cluster. For this, we will utilize a model-based design approach, that uses domain-specific modeling languages and model transformer to enable system design using domain-specific and higher-level abstractions, and uses the monitoring and control framework.

Tools evaluation chart

Tool Name	GUI and Status Reports	Service Polling, Discovery	Alarms, Sensors, Effectors	Memory/CPU requirements
OpenNMS	* * * * *	* * * * *	* * * * *	TBD
PIKT	* * *	* *	* * *	TBD
JFFNMS	* * * *	* * * *	* * *	TBD
Nagios	* * * *	* * *	* * * *	TBD
Aware	* * * *	* * * *	* * * * *	TBD

Architecture: centralized vs. hierarchical

Monitoring: available sensors as well as adding of new sensors.

Scalability: cluster size vs. resource consumption

Handling: smart data mining and virtual sensors.

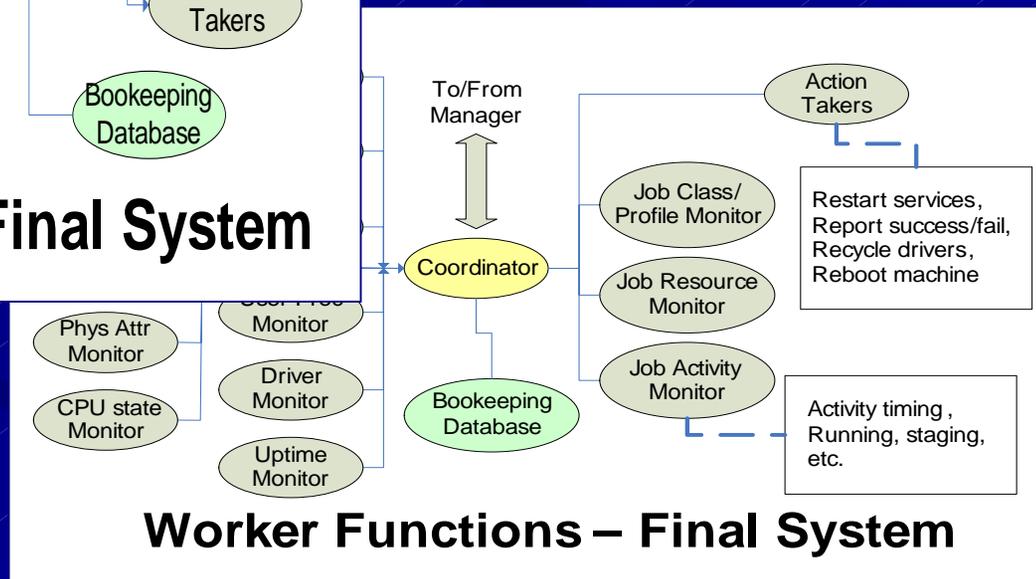
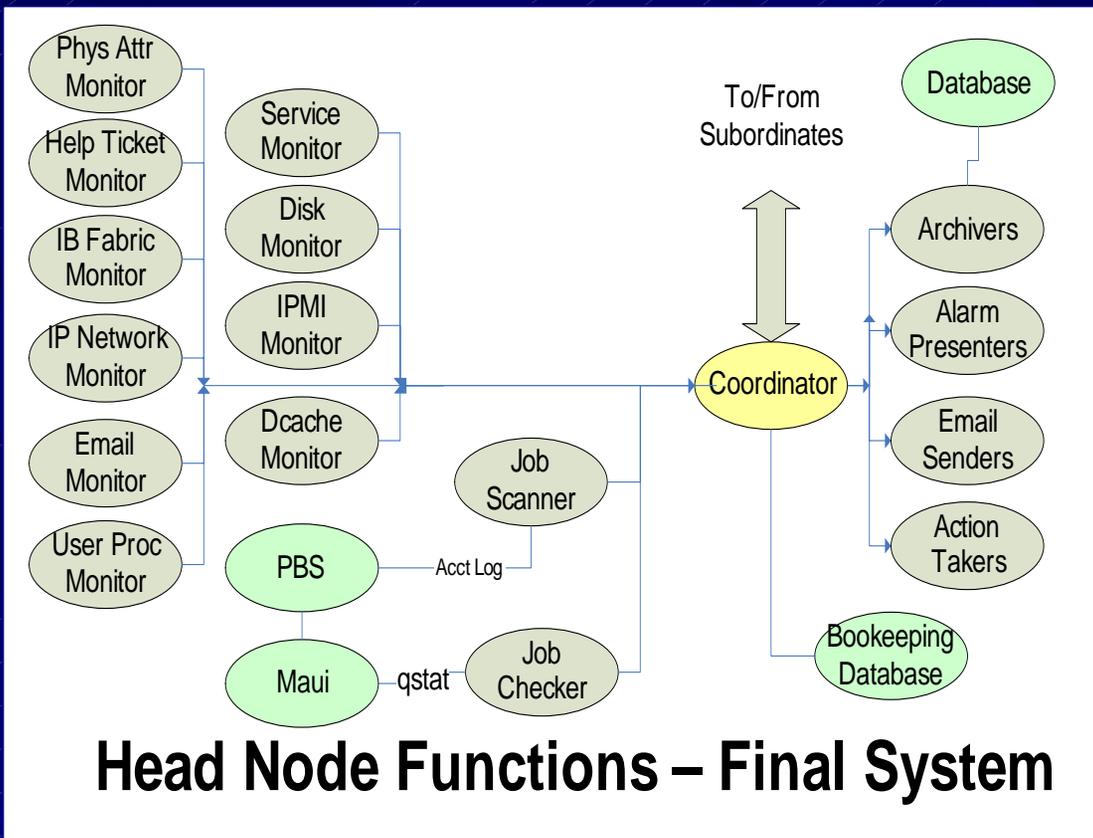
Programmability: configuration language.

Report Visualization.

Near-term major work

- Nov/Dec 06:
 - Get all job information and some currently measured attributes into a database
 - Complete evaluation of products
- Jan/Feb 07:
 - Additions to current code to record resource utilization, IMPI information, errors, and actions
 - Create test system for selected monitoring product
- Mar/Apr 07:
 - Replicating existing functionality in new framework

General architecture



QMP over IB

- This software has been on hold since September.
- A basic design is complete.
- Revamped design and changed from using C to C++ late in the summer.
- Use of higher-level libraries could simplify the management tasks
 - I need to discuss this further with LQCD folks
- Hope to continue this work as soon as the other projects start moving forward.