Ensemble Broker Service Oriented Architecture for LEAD

Jay Alameda[1], Shawn Hampton[1], Brian Jewett[1,2], Albert Rossi[1], Bob Wilhelmson[1,2]

[1]National Center for Supercomputing Applications, University of Illinois
[2]Department of Atmospheric Sciences, University of Illinois

Abstract

We will be demonstrating a set of services designed to broker large number of computational jobs, as described in requirements for the Linked Environments for Atmospheric Discovery (LEAD) ITR project. This broker system, in particular, will have the ability to broker on-demand jobs of various urgency levels, through an on-demand service being developed in collaboration with Cluster Resources, Inc. The broker is designed to support the launching of a multiplicity of ensembles of workflows. An ensemble, for our purpose, is a collection of workflows; each workflow is comprised of nodes of a variety of flavors. These flavors could include remote invocations of our compute-node-local workflow engine, the Open Grid computing environments Runtime Engine (OGRE), a service invocation, or even a plain script. Within a workflow description, we can describe properties and constraints – for example, we may want to place groups of nodes on the same computational resource, or a group of nodes (or workflows, or an ensemble) may need to be run in some on-demand fashion. We will be demonstrating early capabilities of the broker system by executing a subset of the full LEAD capabilities, as motivated by ensemble requirements from the atmospheric science community, on production NCSA and TeraGrid resources.

Research Objectives

- Broker large numbers of computational jobs onto production resources
- Broker on-demand jobs of various urgency levels.

Discussion

In the context of LEAD, we are developing an ensemble broker to respond to the need to manage a large number of simulation ensembles on production resources. In distinct contrast with the core orchestration thrust within LEAD, namely, direct service orchestration (which in some ways is analogous to the phone system, ie, one can dial a number (in this case a service node) and get an immediate response), we need to work in the production batch

computing world, which is much more akin to the postal service, ie, one makes requests and it is not known when one will get a response, not to mention what is the nature of the response.

In order to work with the postal service model, our broker needs to be able to launch an ensemble of workflows, each workflow having a *variety* of node types possible. These types could include remote invocations of our compute-node-local workflow engine, the Open Grid computing environments Runtime Engine (OGRE), a service invocation, or even a plain script. Within a workflow description, we can describe properties and constraints – for example, we may want to place groups of nodes on the same computational resource, or a group of nodes (or workflows, or an ensemble) may need to be run in some on-demand fashion.

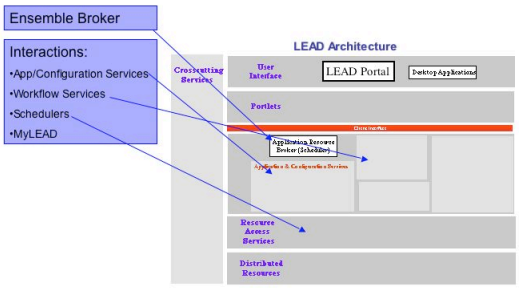

Figure 1: Ensemble Broker in the context of the LEAD Architecture

In Figure 1, one can see the service context and service interactions for the Ensemble Broker. Note that it interacts with many of the core LEAD configuration and execution services, as well as reaching down into lower-level resource access services, which include GRAM, Gridftp, and ssh.

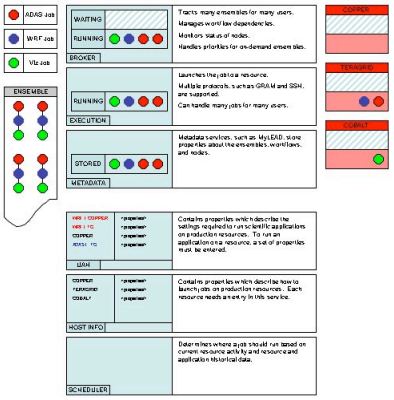

Figure 2: Ensemble Broker functional diagram

In order to understand how the broker works, consider Figure 2. In this figure, we have a 4 member ensemble of workflows that we would like to broker to the production world, represented schematically as Copper, TeraGrid and Cobalt on the right of the figure. Note that each workflow is composed of 3 node types (ADAS, WRF, Vis), represented with colors for each type.

In order to understand how the broker works, consider Figure 2. In this figure, we have a 4 member ensemble of workflows that we would like to broker to the production world, represented schematically as Copper, TeraGrid and Cobalt on the right of the figure. Note that each workflow is composed of 3 node types (ADAS, WRF, Vis), represented with colors for each type.

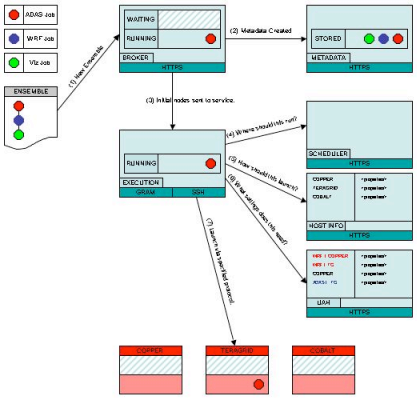

Figure 3: Ensemble Broker service interactions

In Figure 3, we demonstrate how an ensemble is processed through the brokering system. Note that metadata is created and stored during the first interaction with the broker. The Execution Service is responsible for interacting with the scheduler, host information, and UAH services, in order to be able to successfully place a job on a particular computational resource.

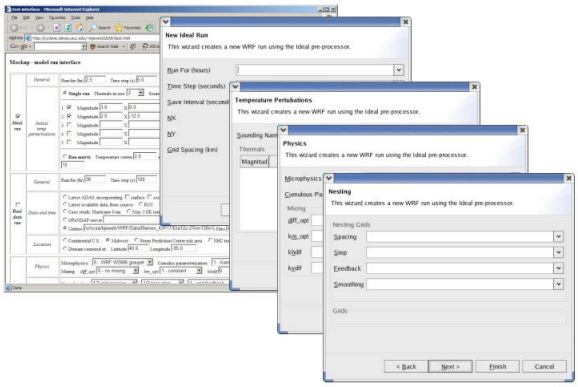

Figure 4: User interface development:
from web mockup (left) to eclipse-based wizard interface (right).

Figure 4 illustrates our process for turning user requirements, in this case expressed as a web page mockup of parameters that are scientifically interesting from the standpoint of describing ensemble calculations, to an interactive desktop client which leads the user to set up their parameter via a familiar wizard. This work leverages the Eclipse platform as a rapid development environment for user interfaces

Future Work

A key capability of the ensemble broker is to be able to broker on-demand nodes. Two services, the MOAB-data service, and the MOAB-on-demand service, will be developed to support this need in particular. The information service, MOAB-data, is a web service back-ended with a relational database which will support data pushed from a resource-local instance of the MOAB scheduler at regular intervals regarding the load characteristics of the particular resource in question.

The MOAB-on-demand service is a web service which allows authenticated connections by other services to determine which level of on-demand service is both allowed, and feasible, by virtue of a fine grained authorization scheme to be implemented within this service. Some of the on-demand scenarios that we will investigate with this service include a N-processor wide reservation starting by a certain time, that will allow the computational task to finish by the desired time (as indicated by the vGRADS scheduler), to preemption of jobs marked as preemptible to create a reservation of the proper processor count, to killing and requeueing running jobs – with policy decisions such as a preference for recently started jobs over jobs near the end of their runtime as a tunable parameter for the MOAB scheduler.

Acknowledgements

This work is supported primarily by the National Science Foundation under the following cooperative agreements: ATM03-31574, 31578, 31579, 31480, 31586, 31587, 31591, and 31594, as well as SCI03-30554, SCI04-38712, and SCI96-19019. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.