Prototype DieselNet Controller Module

Contents

1. Introduction

An overview of a prototype DieselNet Controller module for ORCA has completed by Brian Lynn and David Irwin. They have also provided notes to the Kansei team on how to adapt it to the Kansei project.

There are two files attached:

dieselnet-controller.tar.gz - The controller source and build.
dieselnet-python.tar.gz - Simple python scripts for calling
into the controller via XML-RPC.

2. Summary

Provided by Brian Lynn on May 6, 2009:
Attached is the DieselNet controller for ORCA. The controller is the module responsible for obtaining a lease from ORCA to use resources provided by the DOME testbed. For example, the controller provides the interface that our DOME portal will use to integrate with ORCA.

I do not yet have are the handlers, which will do the DOME-specific functions implementing the ORCA join and leave events. That will be coming out later.

Comments for adapting the controller to Kansei, should you care to, are near the end of this email.

The DieselNet controller is based on code that David Irwin wrote and he provided much guidance. If you have questions, please direct them to both David and me.

There are two files attached:

dieselnet-controller.tar.gz - The controller source and build.
dieselnet-python.tar.gz - Simple python scripts for calling

into the controller via XML-RPC.

For my testing, I built ORCA as described by the 4/20 ORCA release:

https://geni-orca.renci.org/trac/wiki/instructions

I used the tomcat provided by the RENCI January 29 "go release:"

https://geni-orca.renci.org/trac/
https://geni-orca.renci.org/trac/wiki/Deploying

To build the DieselNet controller:

```
$ mkdir controllers
$ cd controllers
$ tar zxf dieselnet-controller.tar.gz
$ cd dieselnet
$ source envvars
  [you may need to edit envvars]
$ mvn clean
$ mvn package
```

To install the controller:

```
start tomcat
  - it's best to always start clean:
     $ cd <tomcat directory>
     $ ./stop.sh
     [ you may need to "pkill java" to really stop it ]
     $ cd webapps
     $ rm -rf orca
     $ cd ../logs
     $ rm *
     $ cd ..
     $ ./start.sh
login to the ORCA administration:
  - http://localhost:8080/orca
  - user: admin  password:
install the dieselnet controller:
  - select admin tab
  - select Install Package
  - browse to controllers/dieselnet/target/
  - choose orca.controllers.dieselnet.tar.gz
  - click install
  - select user tab
  - select Start Controller
```

- choose DieselNet controller
- click create
- it should be "running" if you select View Controllers

select View Reservations

[there should be nothing, but as you request a lease it will show up here]

If you want to test the DieselNet controller:

```
$ tar zxf dieselnet-python.tar.gz
$ cd dometest
$ python reqexp.py
  [ refresh View Reservations and it should show up ]
$ python leases.py
  [ requests status from the controller ]
```

Should you want to adapt the controller to Kansei:

- Copy the files into a kansei directory under controllers
- Generate an ORCA guid for kansei. Replace all references to the DOME guid (de7e9e4d-5dc4-4a7f-8180-7d437c50d634) with your guid.
- You'll need to go through the various files and replace references to DieselNet or DOME with kansei.
- You'll need to rename the dieselnet subdirectories to kansei, and rename the source files (e.g. DieselnetController.java to KanseiController.java)
- Once it builds as kansei, you can change the code to be specific to your requirements. The source files that you would need to customize are DieselnetController.java (which does all of the actual work), and DieselnetHandlers.java (which receives the XML-RPC calls).

- Brian

Provided by David Irwin on May 7, 2009:
I thought I'd elaborate a little bit on how Brian is going about integration. DOME's testbed management software (like nearly all other testbed management software) combines the functions of all three GENI entities---the experimenter control tools, the clearinghouse, and the aggregate manager---in a single software artifact.

The experimenter control tool is the part of the DOME web portal where users go to upload their experiments (in the case of DOME, entire VM images) and request resources (in the case of DOME, nodes in the bus network) for some duration (with some start and end time). The aggregate manager is the part of the DOME testbed that goes out and "pushes the buttons" to satisfy approved requests by either uploading an experiment and starting VMs or halting an experiment by shutting down VMs. The Clearinghouse is the part of DOME that approves research requests (i.e., authorizes users, approves/modifies/fails requests based on resource availability, user priority, or user privileges).

Brian is interfacing his experiment control tool (the web portal) with Orca's implementation of the GENI slice controller and his aggregate manager with Orca's implementaion of the GENI aggregate manager. Once this is done, DOME will be capable of using the Orca implementation of the GENI Clearinghouse; importantly, it will still use its own DOME-specific researcher web portal and DOME-specific aggregate manager functions. For now, these two entities, as before, will function as part of the same software artifact (although they are logically separate and in the future should be capable of being physically separate). So DOME users will go to the same web portal and use the same tools as they did before; there should be little change in their end-user experience and minimal impact on his existing system. Brian is using the Orca web portal primarily as an administrative console, which users won't necessarily interact with.

This initial integration requires minimal changes to the DOME software, since Brian just has to interpose on (1) the place where his software is issuing a user request to a policy that approves/schedules it and (2) the place where approved requests get satisfied by re-imaging nodes and executing researcher code. The slice controller integration he sent you is (1)---it uses a simple XML-RPC interface that Brian's web portal uses to pass requests to the Orca slice controller.

XML-RPC is simply an easy way to integrate two bodies of code written in two different languages; these calls are not meant to traverse a network and are not meant to be "public" GENI methods (which in Orca use SOAP instead of XML-RPC). You can think of these calls as being essentially local procedure calls (if Brian had written his other code in Java he would have just used local procedure calls), and Orca's slice controller as being a logical extension of Brian's researcher web portal. Brian's requests for resources are pretty simple right now (they could get more complex later): each experiment gets the entire network of buses and Brian's aggregate manager identifies the experiment using an ID assigned by the web portal. The code that the experiment should run is identified by another ID. These are set as "configuration properties" in Brian's controller, and they will be sent both to the Clearinghouse (which doesn't do anything with them) and then to Brian's aggregate manager.

Brian's aggregate manager will know what to do with these properties to instantiate the right experiment with the right code. For now, I believe his aggregate manager assumes that the code is pre-staged on the nodes out-of-band, so it's only function is to start and stop the experiment. As Brian articulated, he is working on a "handler package" for Orca that specifies what the functions should do when an experiment starts and stops----for Brian these functions will invoke XML-RPC calls to his equivalent of an aggregate manager and pass the appropriate IDs for the specific experiment.

My feeling is that Kansei's structure is probably similar (although I don't exactly know). You have some portal where a user uploads code and requests resources (in the form of collections of sensor nodes) for some duration; you have some scheduler that applies a policy to determine who/when/where someone should get resources; you have something that takes approved requests and "pushes the buttons" to (re)image nodes and start/stop an experiment. The first step in integrating with Orca (and GENI) is logically (if not physically) dividing your code so that you can offload the authorization/scheduling policy to a GENI Clearinghouse. Thus, the web portal passes requests (via XML-RPC or local procedure calls) to the Orca slice controller (which then forwards them to the Clearinghouse at the right time), and at the appropriate time (before the experiment should start) the Orca slice controller passes approved requests to an Orca aggregate manager. The Orca aggregate manager calls the aggregate manager part of your system when experiments should start and stop----and then your system does whatever it needs to do to start/stop an experiment.

For now, anything beyond separating out a Clearinghouse's authorization/scheduling policy can be done out-of-band (e.g., pre-staging researcher code) using your existing tools to keep things simple, as Brian as done. In the simple case (and you should start with the simple case), Orca's existing Clearinghouse policy is enough to get you started; it approves requests for collections of "things" (e.g., sensor nodes) over time (given the start and end time) and space. Right now it will approve any request as long as the nodes are available during the specified time period (i.e., no priorities, no sophisticated per-resource authorization policies, just first-come-first-served). These policies can be changed, but right now I don't see a pressing need to change them.

Let me know if you have questions.

-David