Getting Started with the Kepler Master-Slave Module

Master-Slave Suite includes:

Master-Slave Module and Kepler Suite

Master-Slave 2.2.0

Mar 24, 2011

Getting Started with the Kepler Master-Slave Module

The *Getting Started with the Kepler Master-Slave Module* guide is for scientists who would like to use the *Master-Slave* module in Kepler. The Master-Slave module was created to distribute parts of a workflow to be remotely executed on other machines.

Table of Contents

1. Introduction

This guide introduces the main components and functionality of the Master-Slave module. The Master-Slave module is an add-on module suite for Kepler, a software application for workflow distributed execution on multiple independent machines in either local-area network (LAN) or wide-area network (WAN). A paper on this module and application is: Jianwu Wang, Ilkay Altintas, Parviez R. Hosseini, Derik Barseghian, Daniel Crawl, Chad Berkley, Matthew B. Jones. Accelerating Parameter Sweep Workflows by Utilizing Ad-hoc Network Computing Resources: an Ecological Example. In Proceedings of IEEE 2009 Third International Workshop on Scientific Workflows (SWF 2009), 2009 Congress on Services (Services 2009), pages 267-274.

2. Downloading and Installing the Master-Slave Module

From the Kepler application menu select Tools => Module Manager. From the Module Manager dialog, select the Available Modules tab. Select 'Master-Slave-2.2.0' from the list of Available Suites, then click the right arrow button to move the Master-Slave-2.2.0 suite to the list of Selected Modules. Click the 'Apply and Restart' button to retrieve the Master-Slave suite and restart Kepler.

3. Master-Slave Architecture

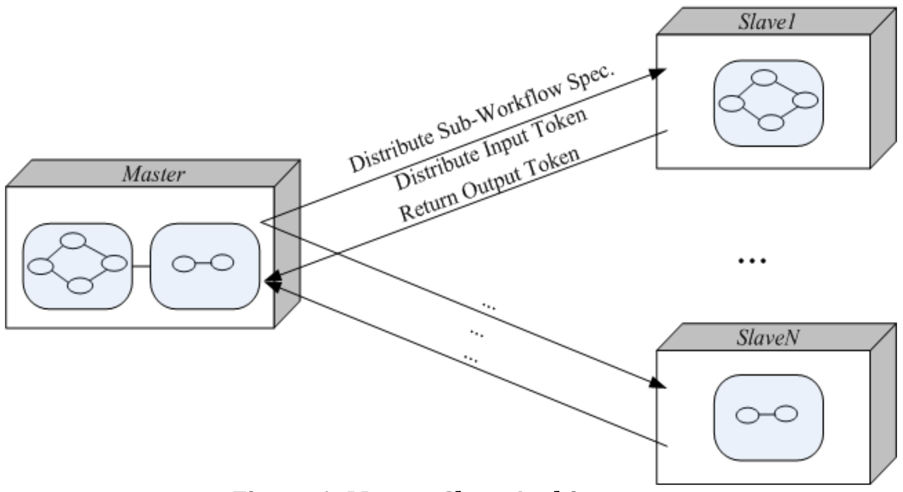


Figure 1: Master-Slave Architecture

In this distributed framework, illustrated in Figure 1, each computing node runs an instance of the workflow execution engine and is assigned one or more roles. Workflow execution is initiated by a *Master* node that performs overall coordination of an arbitrary number of *Slave* nodes that execute sub-workflow tasks. A distributed composite actor was designed, called *DistributedCompositeActor*, to act as the role of Master. Each input data received by the DistributedCompositeActor is distributed to a Slave node, and the result data is returned from the Slave node to the Master node after the sub-workflow is executed on the Slave node.

For the Slave role of our Master-Slave architecture, we implemented a Java Remote Method Invocation (RMI) service that wraps the Kepler execution engine and that communicates with the Master through the underlying RMI infrastructure. For each Slave node, a Slave package and the domain specific programs to be invoked, such as the Fortran and Batch applications, must be deployed on the Slave node beforehand. During the initial phase of the workflow, DistributedCompositeActor will transfer its specification to each Slave. Then the Slaves are ready for receiving input data and execution. The output data will be transferred back to the Master node once the current iteration is finished.

To manage available Slave information, a centralized Web service, called the *Registration Service,* is provided. When a Slave RMI service is started, it can invoke the Registration Service to register this node as a Slave. This allows the Master node to get a listing of all available Slave nodes by querying the Registration Service.

4. Building Workflows with Master-Slave Module

To build workflows utilizing Master-Slave module, users need to use DistributedCompositeActor. Since DistributedCompositeActor inherits the same interfaces of other actors in Kepler, it is very easy to switch between the regular composite actor and DistributedCompositeActor. For workflow users, this actor is very similar to the regular composite actor except a few additional configuration parameters for the Slaves, such as the Slave host URL. From a workflow specification perspective, the only change is the actor class name and some attributes. All other workflow specification details, such as inner actors and links between actors, are still the same. We note that a director has to be used in DistributedCompositeActor to make it work properly.

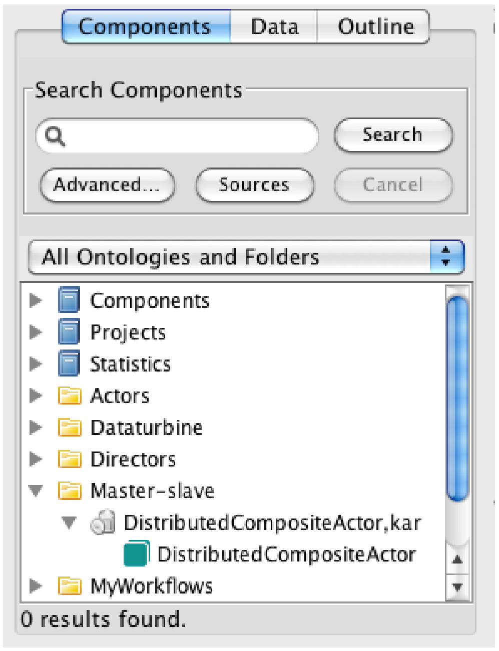


Figure 2: DistributedCompositActor

DistributedCompositeActor can be gotten by either dragging and dropping from actor component panel (see Figure 2), or distribute existing normal composite actors. To distribute a normal composite actor to be a DistributedCompositeActor, users just need right click the actor and use 'Distribute This Actor' option (see Figure 3). Similarly, a DistributedCompositeActor can also be transformed to be normal actor using 'UnDistribute This Actor'.

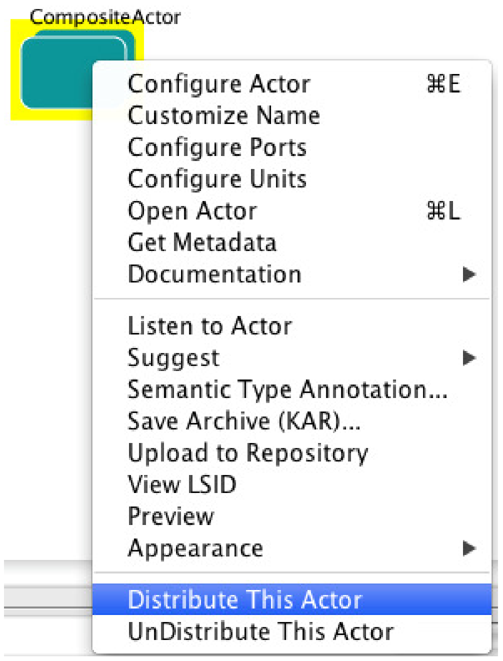


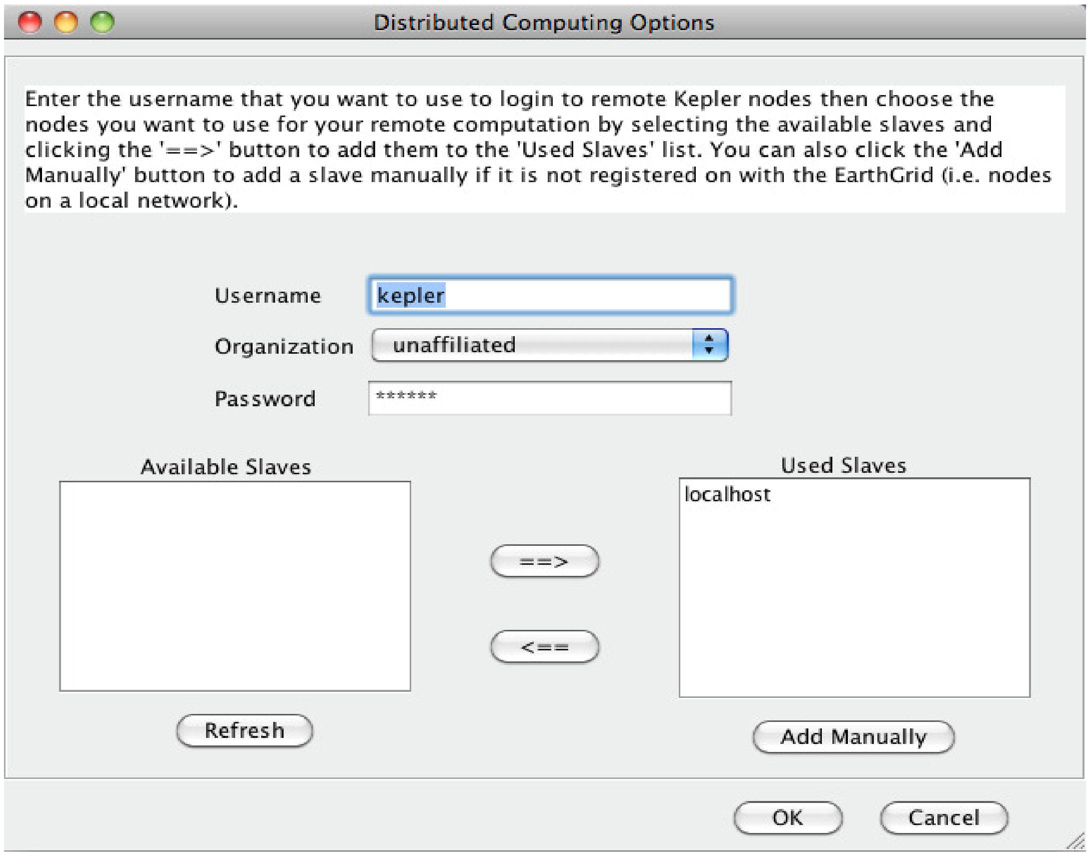
Figure 3: Distribute a normal composite actor to be DistributedCompositActor

5. Distribution Configuration Options

There are several options can be configured in Master-Slave module, which can be categorized into software level and actor level. The options for the software level will be effective for the whole Kepler, namely all workflows built in Kepler will use the configurations. The options for the actor level have to be configured for each actor and therefore only effective for the actor.

5.1. Software Level Distributed Options Configuration

The first main aspect of software level distribution configuration options is the "Distributed Computing Options" dialogue which can be gotten by clicking menu option 'Tools-> Distributed Computing Options' and shown in Figure 4. The hostname or IP addresses of Slaves that are registered to the Kepler repository will be shown in the 'Available Slaves' box. Users can select which Slaves they want to use by moving them to 'Used Slaves' box. If a Slave is not registered, it can still be used by using 'Add Manually' button and put its hostname or IP address.



Distributed Computing Options

Enter the username that you want to use to login to remote Kepler nodes then choose the nodes you want to use for your remote computation by selecting the available slaves and clicking the '==>' button to add them to the 'Used Slaves' list. You can also click the 'Add Manually' button to add a slave manually if it is not registered on with the EarthGrid (i.e. nodes on a local network).

Username: kepler

Organization: unaffiliated

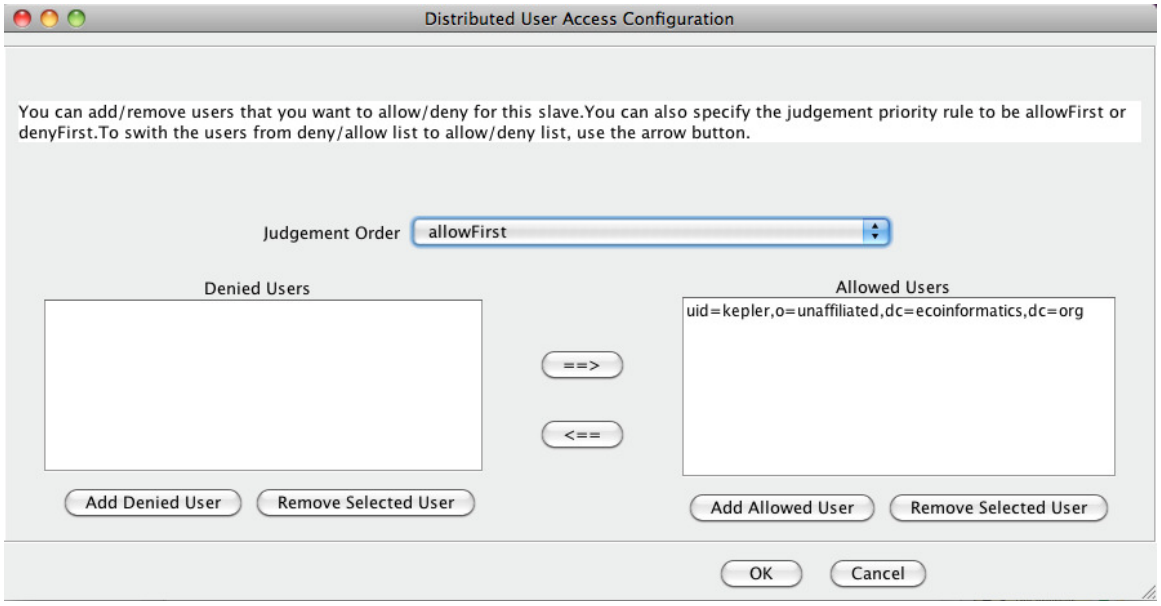
Password: ******

Available Slaves:

Used Slaves: localhost

Figure 4: Distributed Computing Options

The second main aspect of software level distribution configuration options is the “Distributed User Access Configuration” dialogue which can be gotten by clicking menu option ‘Tools-> Distributed User Access Configuration’ and shown in Figure 5. This dialogue manages which users can use the Kepler instance as a Slave. Since we are using the LDAP user management service provided by Knowledge Network for Biocomplexity (http://knb.ecoinformatics.org), each user should have his/her distinguished name, such as 'uid=kepler,o=unaffiliated,dc=ecoinformatics,dc=org'.


You can add/remove users that you want to allow/deny for this slave.You can also specify the judgement priority rule to be allowFirst or denyFirst.To swith the users from deny/allow list to allow/deny list, use the arrow button.

Judgement Order: allowFirst

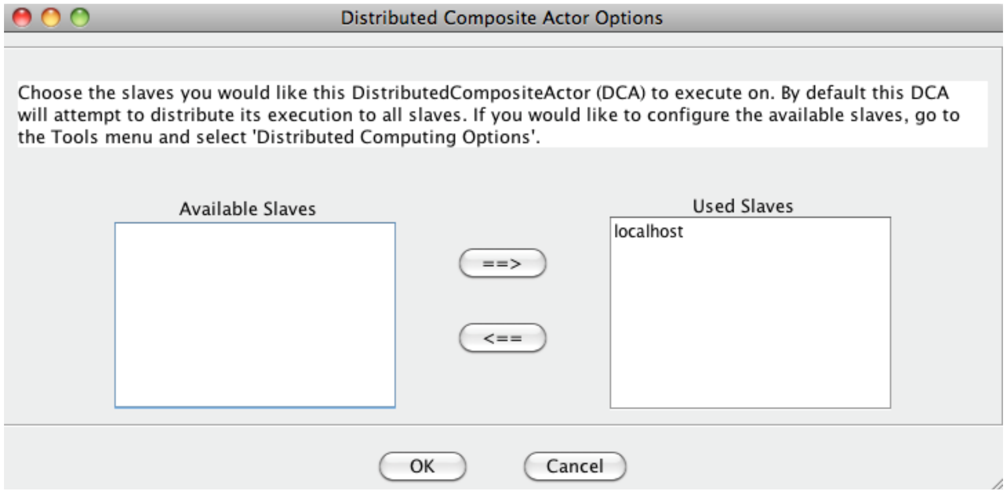
Allowed Users: uid=kepler,o=unaffiliated,dc=ecoinformatics,dc=org

Figure 5: Distributed User Access Configuration

Besides the above two main aspects, more configuration options can be done by editing the configuration file of this module, which is located at $Kepler/master-slave-2.2/resources/configurations/configuration.xml. The options include whether register this Kepler instance, where to register, user information and so on.

5.2. Actor Level Distributed Options Configuration

When double clicking a DistributedCompositeActor or distribute a normal composite actor, a dialogue (shown in Figure 6) called "Distributed Composite Actor Options" will show up. This dialogue will show or specify which Slaves will be used for this particular actor. By this way, different DistributedCompositeActor actors might use different Slaves in one workflow.


Choose the slaves you would like this DistributedCompositeActor (DCA) to execute on. By default this DCA will attempt to distribute its execution to all slaves. If you would like to configure the available slaves, go to the Tools menu and select 'Distributed Computing Options'.

Used Slaves: localhost

Figure 6: Distributed Composite Actor Options

6. Start Kepler as a Master/Slave Node

By default, when Kepler is started with Master-Slave module, this instance can be used as both Master and Slave node. Also a Master node does not need to be started separately, it will communicate with Slave nodes when a workflow executes. Yet a Slave node has to be started beforehand so that it can handle requests from a Master node. A machine may only be accessed by batch mode, where Kepler graphical user interface cannot show up. So we also provide a set of commands to start Slave separately, which can be found in $Kepler/master-slave-2.2/ directory. The 'startSlave' command will start a Slave process at the node, display execution messages, and will not quit until explicitly typing 'X' in the console. The 'startSlaveAsyn' command will start a Slave process asynchronously and quit immediately, leaving the Slave process running in the background. The 'startSlaveWithSecurityPolicy' command is to start Slave in a more secure way whose policy is specified in the 'master-slave.policy' file in the same directory. The 'stopSlave' command is to explicitly stop a Slave process which is started by the above commands.