Kepler Scientific Workflow Engine

What is Kepler ?

The purpose of the Kepler Project is develop a software package to model scientific workflows that can be run repetitively without need for human mediation. The Kepler software package is based on Ptolemy II system for modeling, simulation, and design of concurrent, real-time, embedded systems. This rich set of tools provided by these two projects gives users with the ability to create and re-use sophisticated data analysis pipelines.

How does it work ?

Kepler begins with a graphical user interface for composing workflows and editing the workflow environment. The resulting workflows connect a series of data processing elements such that the output of one element is input to the next. Workflows are saved from the GUI to XML files that contain an executable representation of all components. In addition, Kepler provides a sophisticated run-time engine that can execute workflows either from within the GUI or from XML files via command line or a batch process.

Kepler uses the Actor/Director paradigm. It's data processing elements are know as actors. Actors communicate with each other through well-defined interfaces called Tokens.

Directors control the order and timing of execution of each actor. Directors communicate with Actors through well-defined interfaces, giving each one just enough information to successfully perform it's task. Directors also control whether actors are executed sequentially, in parallel or using time models.

Key Features

- Kepler is a java-based application that is maintained for the Windows, OSX, and Linux operating systems.
- Python actor can be used to create domain- or content-specific scripts for accessing, merging and manipulating data.
- R and Matlab actors can be used to easily perform complex statistical analyses.
- A WebSer-vice actor can be used to access and execute WSDL-defined Web services from within a workflow.
- An ExternalExecution actor can be used to execute command line applications from within a workflow.
- Groups of Actors can be saved as a CompositeActor, thus allowing a com-plex series of tasks to appear to the user as a single component.
- A number of grid technology actors provide access to web-accessible data repositories and support for parallel processing.
- Kepler workflows can be exchanged in XML using the Modeling Markup Language (MoML) described at http://ptolemy.eecs.berkeley.edu /publications/papers/00/moml/moml_erl_memo.pdf .

Issues

While Kepler does provide an effective environment for integrating a variety of software components into a coherent, reusable pipeline. it does have some limitations.

The GUI

The Kepler visual interface is both simple and hard. It is simple in that you can drag and drop components onto a canvas and graphically connect them in the order you want them to execute. It is also easy to run a workflow or to go to the File menu and save the workflow. Beyond this, it is difficult for the average scientist with limited software development experience.

The rest of Kepler's UI is analogous to a programming IDE. It is an excellent tool for building and testing workflows, but it is not always clear what a user must tweak in order to get the desired results from a wokflow. Of particular concern, input parameters to both actors and directors need to be entered as strings in non-intuitive interfaces.

When it comes to deploying the workflows to users, simply setting them up with a copy of Kepler and expecting them to run their workflows, which may require tweaking of the actors properties for different runs,
is like expecting a user to modify a program's source code each time they want to run the program.

Another problem is that, with the exception of workflows that use grid job submission actors, all workflow execution is performed on the user's desktop. This is not an ideal environment for jobs that require heavy computation or long running jobs.

Any validation of input parameters must be scripted via a Python or R Actor or included in the Java code of a supported Actor.

When dealing with a repetitive workflow such as spreadsheet row/column manipulation, there is no support for conditional interaction during a sequential workflow run.

Originally, Kepler was designed to download data sets into a cache on the machine where it is running, so Kepler actors run as local Java threads. However, in order to provide access to web-based resources, actors have been implemented that spawn distributed execution threads to access distributed resources.

Using Jython in Kepler

Python scripts run in a Jython interpreter within the same JVM as Kepler. This interpreter is only instantiated once so it does not need to be reloaded by each Python actor. On the downside, a global variable set in one actor can cause unexpected side effects if it is reset in another actor. We discovered that if a function is defined as a global (i.e. outside of a class), the method name cannot be reused. The Jython interpreter will always persist the code from the first instantiation of the method.

PythonActors use the Jython 2.2.1 interpreter which implements a subset of the Python 2.2 language. See http://www.jython.org/archive/22/index.html for details.

Add-on packages such as FCRepoKepler must be put in the user's home directory under the Lib directory. For example, on a Windows box, this would be C:\Documents and Settings\username\Lib.

Kepler 2.0 (dev)

PythonActors use the Jython 2.5.1 interpreter which implements a most of the Python 2.5 language. See Is Jython the same language as Python for details. This is a significant improvement over the version of Jython supplied with Kepler 1.0.

Add-on packages such as FCRepoKepler must be put in the Kepler installation directory under the core/lib/jar/Lib directory.

Using R and MATLAB in Kepler

R and Matlab actors require that those applications be installed outside of Kepler. When an associated script is run, the code and data are passed to an external process running R or Matlab. The actor waits for that process to finish before proceeding. Thus workflows with multiple R/Matlab steps will incur the corresponding application startup overhead for each instance of the actor. This can significantly affect workflow performance. Kepler is a very good tool for building scientific workflows for those who are familiar with software development. However, it is not particularly friendly to those who are used to
. (Google Code project hydrant-kepler discussion of this issue)

Issues with Kepler 1.0

The PythonActor is not in the list of availbale actors. In order to use it you must open one of the Python demo worflows and copy one of those actors to your new workflow.

PythonActors have a tendency to lose their scripts if they contain a coding error when the workflow is saved. On occasion, we have seen similar behavior when copying PythonActors from one workflow to another. Be sure to save a copy of Jython scripts for **ALL** PythonActors in a file external to Kepler.

It should be possible to save fully scripted Python Actors for reuse. So I did some research and found a poorly documented feature that sometimes saves scripted PythonActors in the local Kepler component library. I say sometimes because it may take a few tries to overcome the PythonActor's tendency to lose track of it's script along the way

These irregularities in behavior apparently occur because the PythonActor hasn't been integrated into Kelper's Actor repository yet. The only reason we can use copied PythonActors is because the base class is available in the underlying Ptolemy II modeling engine.

New Java actors cannot be dynamically loaded to an installed instance of Kepler 1.0. Incorporating a new Java actor requires a rebuild of the entire application and consequent rebuild of the actor database. Thus, a small change to an actor under development requires a complete rebuild of Kepler.

Issues with Kepler 2.0 (development version)

Kepler 2.0 is very promising with a lot of features that will make life easier for projects that need to extend Kepler's functionality. Most important is the updated module management system that allows new features to be loaded at runtime rather than requiring complete rebuilds from source.

There are several new actors available and the PythonActor is fully integrated into the Kepler actor database. This has improved much of the flakiness seen when using PythonActors in Kepler 1.0. **NOTE** using Kepler 1.0 to open a workflow that was saved in Kepler 2.0 can cause PythonActors to lose their scripts. Reopening these workflows in Kepler 2.0 will NOT fix them, the scripts must be re-entered from some external source.

Kepler 2.0 is still in development and changes daily, so it has proven to be very buggy at times. On the other hand, the development team does a very good job of letting the community know about critical bugs and then fixing them quickly.

The targeted release date now appears to be early 2010. The Kepler 2.0 Release Roadmap discusses details of what will be included in that release.

Downloads

Kepler 1.0

Download Kepler 1.0 at https://kepler-project.org/users/downloads

Kepler 2.0 (development version)

Download source code from the SVN trunk and use the Kepler build system to run it. See Kepler Build System Instructions in documentation section below for details.

Documentation

Kepler's documentation is somewhat scattered through it's web site so it can be difficult to find just the document that you need. Here is a list of key documents to get you started.

Getting Started Guide - PDF

Kelper User Manual - PDF

Kepler Actor Reference - PDF

Kepler Developer's Reference - wiki page

Kepler Build System Instructions - wiki page

Kepler and Eclipse - wiki page

NOTES

Python actors were heavily used to do the proof of concept because they provide a way to rapidly prototype functionality. Python actors use the Jython version of Python (http://www.jython.org). In addition to providing most standard python functionailty, the Jython interpreter provides access to any Java class or class library available to your JVM.

To learn more about Jython:

- Jython Users Guide - http://wiki.python.org/jython/UserGuide
- "Jython Essentials", by Samuele Pedroni & Noel Rappin - http://www.oreilly.com/catalog/jythoness/ - this is a bit outdated but still useful
- Online training course by Dave Kuhlman http://www.rexx.com/~dkuhlman/jython_course_03.html

---