Kepler - Bug #3739

Issues in inferring data lineage via the provenance recorder

01/10/2009 10:09 AM - Shawn Bowers

Status:	Resolved	**Start date:**	01/10/2009
Priority:	Normal	**Due date:**	
Assignee:	Daniel Crawl	**% Done:**	0%
Category:	provenance	**Estimated time:**	0.00 hour
Target version:	3.X.Y	**Spent time:**	0.00 hour
Bugzilla-Id:	3739		

Description

I am trying to figure out how to infer data/token dependencies (more generally, data lineage information) via the current Provenance Recorder component/implementation in Kepler. It looks as though the recorder is providing the following high-level information (shown as relational tables):

Fire(Actor, StartTime, EndTime)
Write(Actor, Port, Channel, DataValue, WriteTime)
Read(Actor, Port, Channel, DataValue, ReadTime)

If we assume that actor invocations do not maintain state, and that all data read by an actor invocation prior to writing a data value were used to derive the written value, then one could derive simple data dependencies using the following high-level query (or view). Note that in general, these are severe restrictions, e.g., many Kepler/Ptolemy actors maintain state and do not use all input data to derive all output data.

DependsOn(DataValue2, DataValue1) :-
Fire(Actor, StartTime, EndTime),
Read(Actor, _, _, DataValue1, ReadTime),
Write(Actor, _, _, DataValue2, WriteTime),
StartTime <= ReadTime <= WriteTime <= EndTime.

This query says that if an actor fires from StartTime to EndTime, and during this time range it read DataValue1 before it wrote DataValue2, then implicitly DataValue2 depended on (or was derived by) DataValue1.

Here are three problems (there may be more) that seem to prevent these simple types of dependencies, and in general data lineage, from being inferred from the information provided by the Provenance Recorder.

1. The granularity of timestamps recorded for fire, read, and write events is too coarse (I believe at the granularity of seconds). For many Kepler/Ptolemy workflows this means that every such event has the same timestamp (thus, timestamps in this case become "meaningless"). A simple fix for this issue is for the Provenance Recorder to generate timestamps at a finer granularity.

2. To compute data lineage information---the transitive closure of these simple dependencies, e.g., to infer which input data items the output of a workflow were derived from---instead of recording data values above, we should be recording token ids. This is because the same data value can be produced and consumed by multiple invocations in a workflow, which confuses the inference of data lineage information. A possible approach for obtaining an object id for Ptolemy tokens would be to use the System.identityHashCode(Object) method provided by Java. Other alternatives would be to add an "id" field to Token (that could be auto generated on token creation), but this would present additional difficulties (e.g., it would require each token to be "larger", possibly decreasing efficiency).

3. The Provenance Recorder does not record the read, write, and fire events of all actors. In particular, very basic actors like ArrayToSequence are not recorded. I'm not really sure why this is the case, but unless each invocation has its events recorded, in general, it will not be possible to infer data lineage via the Provenance Recorder.

While I am not sure what information specifically the new workflow execution reporting tools are proposing to track and present to users, basic data dependencies and lineage information (e.g., to determine which input data produced output data) to me seems like essential information. So, figuring out how to record provenance events to infer these dependencies generally for all Kepler workflows seems crucial.

Thoughts?

Thanks,
Shawn

History

#1 - 04/27/2009 12:01 PM - Daniel Crawl

1, 2: version 8 of the schema explicitly includes token dependencies, and can be queried using the Queryable interface.

3: I am unable to reproduce this. There have been several bug fixes since this bug was created, so perhaps one of these has fixed the problem. If not, please attach a workflow demonstrating it.

#2 - 03/27/2013 02:24 PM - Redmine Admin

Original Bugzilla ID was 3739