

# Scientific Workflows and Components: Together at Last!

Kostadin Damevski, Ayla Khan and Steven Parker  
Scientific Computing and Imaging Institute  
University of Utah

(presentated by Boyana Norris)

# Scientific Workflows

- Scientific workflows consist of one or more actors that commonly execute in a dataflow fashion
- Each actor represents a relatively complex execution step (e.g. fetch data from web service, put job on cluster's queue)
- Actors are loosely coupled
- Temporal decomposition

# Scientific Workflows

- Actors are connected to form an application and a token (string) is given to each actor when it begins execution
- The Kepler (<http://www.kepler-project.org>) system, is used to create, coordinate and execute scientific workflows
- Several “mature” workflows have been created for a wide range of scientific fields (e.g. ecology, biology, astronomy)

file:/Users/aykhan/workspace/kepler/workflows/SC06-Tutorial/JobSubmission.xml

File Edit View Workflow Tools Window Help

file:/Users/aykhan/workspace/kepler/w. . . tutorial/JobSubmission.xml#SubmitSimJob

Component

Search

Components Data

Search

Search repository

Search Reset

Search Results

- Components
- Projects
- Disciplines
- Statistics

General Purpose

Job Command

JobSubmitter

0 results found.

submitFile: \$JobScript  
 remoteWorkDir: \$RemoteDir  
 logFile: \$LogFile  
 logFormat: \$LogFormat

CreateJob jobID

JobSubmitter jobIn jobManager jobOut succ log

Logger

job

ThrowException

Configure ports for JobSubmitter

Name	Input	Output	Multiport	Type	Direction	Show Name	Hide	Units
jobIn	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		DEFAULT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
jobManager	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		DEFAULT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
jobOut	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		DEFAULT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
succ	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		DEFAULT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
log	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		DEFAULT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Commit Apply Add Remove Help Cancel

file: /Users/aykhan/workspace/kepler/workflows/SC06-Tutorial/RunSimulation.xml

File Edit View Workflow Tools Window Help

Components Data

Search  
exec

Search repository

Search Reset

23 results found.

Search Results

- Components
  - Data Input
  - Data Output
  - General Purpose
    - Grid Function
    - SSH
      - SSH Execute cmd
      - Unix Command
      - Web Service
  - Projects

SDF Director

Workflow version \$Revision: 1.6 \$ on \$Date: 2006/10/29 02:40:37 \$

```

graph LR
    SimTarget[SimTarget] -- target --> Exec[Exec]
    SimCmd[SimCmd + " " + SimArgs] -- command --> Exec
    Exec -- stdout --> Display[Display]
    Exec -- stderr --> Display
    Exec -- exitcode --> Expr2[Expression2]
    Expr2 -- "Exit Code: " + exitcode --> Display
    Exec -- errors --> Expr[Expression]
    Expr -- "Errors: " + err --> Display
  
```

Simulation machine? [user@]host :

Simulation program start cmd:

Simulation program start args:

- SimTarget: "norbert@sdm5.csc.ncsu.edu"
- SimCmd: "/home/sdm/bin/TutorialBigSimulation.sh"
- SimArgs: "-f 2500 -l 2520 -s 1 -o simout -v"

# Scientific Components

- Common Component Architecture (<http://www.cca-forum.org>) is a well-established standard for scientific components
- Scientific IDL to define each component's uses/provides ports
- Several CCA framework implementations exist:
  - We use SCJump for this work

# Scientific Components

- In contrast to Kepler, CCA is:
  - Tightly coupled components
  - NOT temporally decomposed
  - Usually, expresses simulations on at a finer grain than Kepler
    - i.e. a greater number of components than workflow actors would be used in order to accomplish a similar task

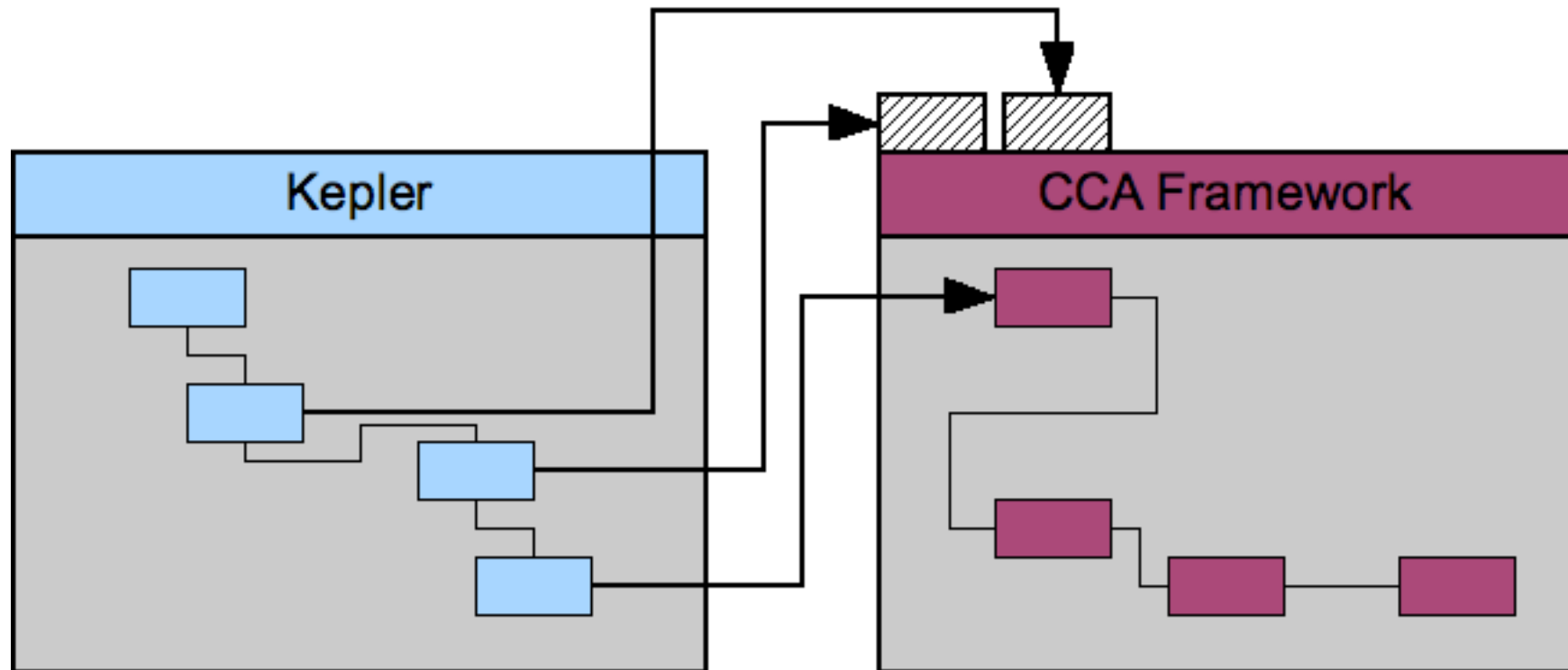
# Component - Workflow Interoperability

- Why?
  - Both technologies have a lot of traction in the scientific community and have a large set of interesting components, actors and applications
  - Apart from re-use, a user may benefit creating an application using both technologies
    - leveraging the best tool for each application part
    - the workflow model is limited in expressing control-flow (e.g. web-service retry)

# Interoperability Design

- Workflow decomposition is more coarse-grained, and they are intended to provide top-level control of a scientific application
  - Our system allows workflows to be in charge of the interaction with components. However, communication flows in both directions
- Our design plan: Create a set of Kepler actors that enable interacting with a CCA framework and individual components.
  - we call these CCA actors in this presentation

# Interoperability Design



# Interoperability Design

- These new CCA actors will:
  - Communicate with framework services in order to stage a CCA application (create and connect components)
  - Communicate with individual CCA components in order begin the execution, or transfer parameters
  - Communicate with either framework services or individual components to get intermediate values and provide computational steering

# Implementation

- The SCIJump CCA framework v.1.3 is based on the Babel SIDL compiler
- Leverage the Babel SIDL compiler to generate stubs for our CCA actors
  - Gain painless multi-language communication (SCIJump is written in C++, Kepler in Java)
  - Gain distributed ability (using Babel Remote Method Invocation)

# Proof of Concept Application

- To demonstrate our design we implemented a proof of concept application that integrates SCIJump and Kepler
  - Implements a “minimal” set of three CCA actors
  - Uses the CCA tutorial as an application in SCIJump
    - Estimates PI using a few components

# “Minimal” Set of CCA Actors

- ***CCAEventListenerActor*** -- connects to a framework’s event service and subscribes to event(s), reacts when something is published
- ***CCAInvokePortMethodActor*** -- when fired executes a port’s method (given a port name, method name, and arguments to that method)
  - Relies heavily on Java reflection

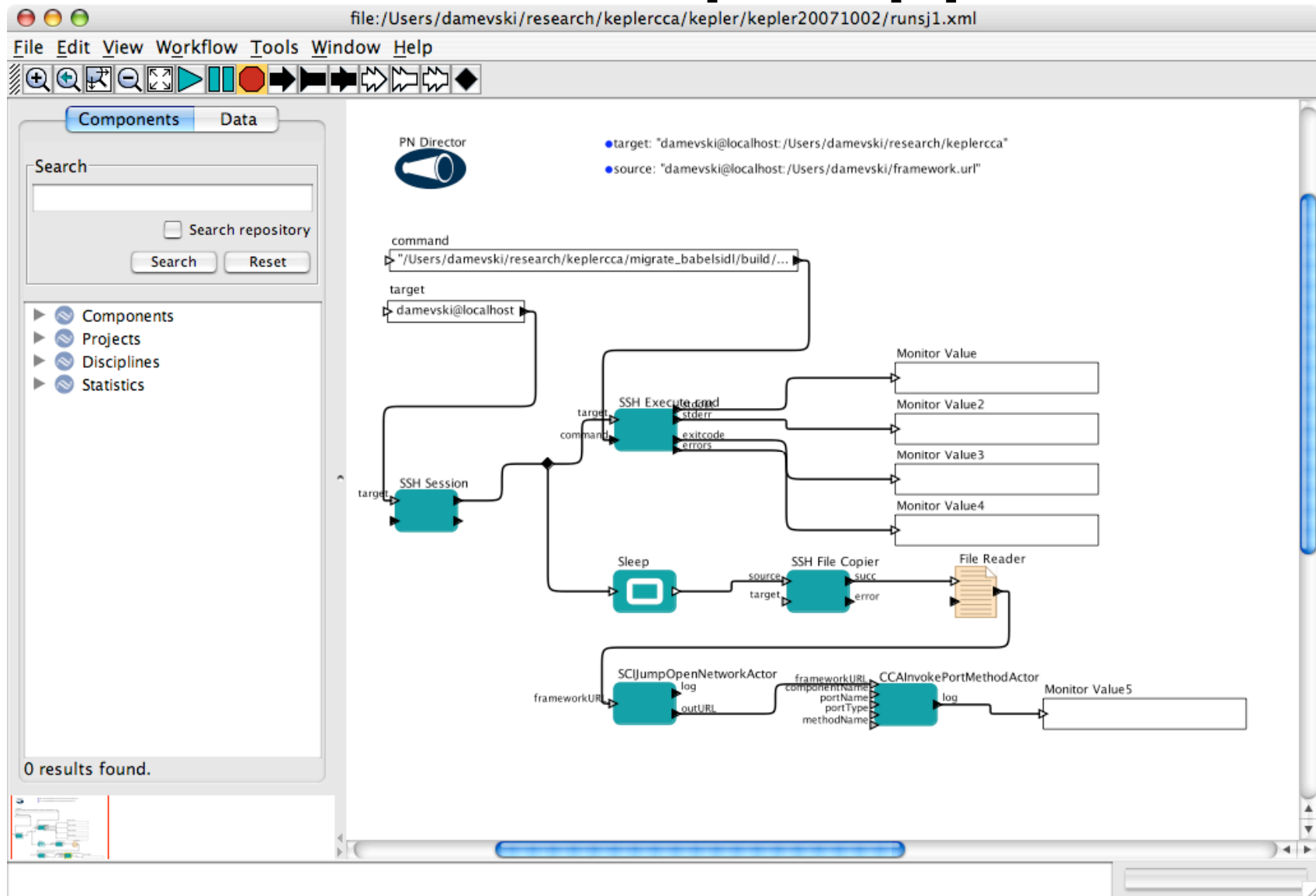
# “Minimal” Set of CCA Actors

- ***SCIJumpOpenNetworkActor*** -- the functionality to open a saved application file in SCIJump and automatically get all the instantiations and connections. This uses our `ApplicationLoaderService` that other frameworks could adopt.
  - This actor is SCIJump specific, however, by exposing the `Builder Service` we could have achieved the same goal, though by taking a few additional steps

# P. of C. Application Revisited

- Kepler workflow starts, actors set up environment variables and start *SCIJump*, grabbing its URL
- Using *SCIJump*'s URL, the *SCIJumpOpenNetworkActor* loads the CCA tutorial components and connections
- The *CCAEventListenerActor* is registered to listen for the completion of the simulation and the results
- *CCAInvokePortMethodActor* is used to execute the component application
- Result arrives to the *CCAEventListenerActor*

# Proof of Concept Application



# Future Work

- A larger, more “realistic” example
- Through this example, identify other useful CCA-Kepler interoperability actors:
  - CCAEventPublisherActor
  - CCABuilderServiceActor
  - etc.

# Questions?

- Thanks to the CCA-Forum and DOE's SCIDAC program
- Note: The implementation of the three CCA actors (in pre-alpha, no guarantees) should be available with the CBHPC proceedings