Tools, methods and services enhancing the usage of the Kepler-based scientific workflow framework

Presenter: Marcin Płóciennik

Poznan Supercomputing and Networking Center, Poland

ICCS 2014, Crains, 10.06.2014
Agenda

- Use cases: Nuclear Fusion (EFDA ITM), Nanotechnology, Astronomy,
- Tools:
  - Actor generators
  - C
    - Binding physics code: FC2K, WS2K, HPC2K (ITM)
  - Central Kepler installation (ITM)
  - Provenance enabled monitoring and profiling (ITM)
  - Sharing workflows
  - GUI for running workflows in distributed environment
Use cases
EFDA ITM-TF work on validated suite of simulation tools for ITER and DEMO plasmas

Whole simulation platform that allow cross validation between different fusion devices (Kepler as w-f engine)

Individual codes in Fortran, C++, C, Python, Java and also Matlab

Many very complex multi level workflows has been developed in Kepler: European Transport Solver (ETS), Turbulence-transport, Equilibrium reconstruction and MHD

ETS: the workflow integrates more than 1000 entries among which there are more than 150 actors and has multiple layer structure, which goes up to 15 levels deep, a number of convergence and time loops implemented at different levels of the workflow

the data structure based on standardized generic objects, called Consistent Physical Objects (CPO), and the Universal Access Layer (UAL) for these data objects

90+ users- workflow developers and support team
European Transport Simulator

**Workflow parameters**

**Start up**
- Initialization, sets new time and run numbers.
- Reads profiles of the input files from the data bank.
- Sets up the ETS grid.
- Interpolates starting profiles on the ETS grid.
- Initialization: equilibrium-transport coupling.

**Time loop**
- Computes transport of impurities.
- Computes new profiles of transport.
- Computes new equilibrium profiles.
- Checks convergence.
- Checks if the time increment is within the limits.
- Checks if the run is completed.

**Finalizing the run**
- Necessary post-processing steps.
- Closes the file, closes the SOL, and stop the run.

**Convergence loop**
- This step is updating the SOL.
- Goes to next time step.
- Goes to next iteration.

**ETS dimensions**
- TRANSPORT:
  - Modes:
    - SOL_ID:
    - MODE:

- EQUILIBRIUM:
  - Modes:
    - SOL_ID:
    - MODE:

- MHD:
  - Modes:
    - SOL_ID:
    - MODE:

**Example workflow**
Fusion - requirements

- Automatically include the physics codes written in different languages
- Run codes in the context of different computational resources,
- Management of the platform releases (including customised Kepler),
- Maintenance and sharing of the physics actors
- User friendly interface for run configuration
- Easy exchange of modules of the same kind, in order to optimize the physics complexity versus the performance (CPU time)
Requirements - Open points

- Many issues solved, but still some challenges are ahead
  - How to provide user friendly interface for a large sets of Kepler workflow parameters?
  - Often the exchange of actors are done at the level of the composite actors, which already includes a number of individual physics actors and substructures. number of repetitive parts in different places of the workflow. User preferences would be to synchronize the settings for repetitive components. What would be the best strategy?
  - It would be great to have the possibility (standard) of exporting of the actors but not from GUI mode, like save as:
    - function of API of kepler.jar library?
    - as an argument of the Kepler in non-gui?
  - User friendly GUI to steer the workflow execution/provide interaction (standalone)
Atomistic simulation methods such as molecular dynamics are used to study materials. Output from the simulation (atoms within the simulation box), needs further data processing e.g., the calculation of radial and angular distribution functions. Significant difficulties arise when disordered systems (e.g. glasses) are investigated. In such cases, in order to exhaustively characterise the structure of the system, one needs to take into account the medium-range order.

A software suite - ANELLI has been designed with the aim of performing ring analysis (medium-range order in computer-simulated solid state)

Requirements:

Creation phase:
- Easy addition of new C- actors
- Easy parameterisation and triggering the components
- Possibility of easy update of the codes on the fly

Execution phase:
- Running in distributed computing environment
- Workflow should be transparent to user
ANELLI Workflow
Astronomy

Several workflow use cases implemented (DIAPL, BF) – more on next presentation

Based on *IVOA* standards

The goal is to create universal modules that could be used by different astronomy groups

Requirements:

- Interoperability with VO services and applications (mainly SAMP, ds9, Topcat)
- Interactivity in the workflows
- Integration with iRods
- Integration of different codes
- User friendly GUI to run w-f
Example - DIAPL
Tools
Physics actors generator

• wrapping the Fortran or C++ source code into Kepler actor:
  • FC2K (Fortran Code to Kepler),
  • HPC2K (HPC to Kepler),
  • WS2K (Webservices to Kepler)

• run in different computational context

• The precondition for using these tools is that the codes are using CPOs as input/output
WS2K & HPC2K & FC2K

- **WS2K**:
  - transforms the FORTRAN or C++ code into a web service,
  - adds data wrapper for accessing remote data structures,
  - installs web service inside a web server,
  - adds Java actor into Kepler
  - Using WS2K, the codes are stored separately and possibly remotely in a HTTP server

- **HPC2K**
  - automatically handles the jobs on the HPC or HTC resources.
  - the code (Fortran, C,C++) must initially be compiled as a library with a main, single function that corresponds to the execution of the code.
  - Uses Serpens suite

- **FC2K**
  - Addresses local and batch execution
The automatically generated actors make use of dynamically loaded libraries which remain in Kepler process space throughout the workflow execution.

- The input arguments -> actor port inputs
- Automatic generator of 3 output ports:
  - Output (of C program); StdoutPort; StderrPort
- Actor generation procedure
  - Preparing actor port description in XML
  - Actor generation
    - Make depend
    - Make all
    - Make INSTALL=/path/to/kepler/plgrid install
- Adding actors to Kepler
  - Tools->Instantiate Component
  - Class name: pl.psnkepler.jna.c-app-name
Central Kepler installation

• **Use case:**
  - Common core installation for all users (90+)
  - Personal set of actors/modules and specific settings
  - So far each user had copy of the Kepler in home directory

• **Solution**
  - Scripts creating a shadow copy of central installation in the user area
  - Only working directories specific for each user installed in the user area

• **Easy management and maintenance**
Provenance enabled monitoring and profiling

- Large workflows may trigger issues with proper understanding of data and execution flow. Having complex flows including multiple and nested composite actors (that wrap native code), makes it hard to determine how workflow is actually executed.

- ITM Workflow Profiler that is based on standard Provenance Recorder, however, it adds some modifications into it.

- Assumptions:
  - internal loop counters should be reported,
  - global parameters should be stored in the output,
  - each actor during its execution should store input and output value
  - execution of native codes should be reported in the output.
  - data transfer details and execution time included
  - Interactive analysis (during run)

- We have decided to use quasi XML format that can be analysed by external tool, however, it is not required to read the whole, complete XML output in order to analyse it.
MWRKF browser visualises the output of the Provenance recorder (Workflow Profiler's result).

person executing workflow can analyse the results live and determine elements that are most time consuming.

An actor can be a composite actor. In this case the elapsed time pasted is the sum of elapsed time related to all of its children.

Clicking on a composite actor tab gives access to the content of this composite actor.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tstop</td>
<td>11.0</td>
</tr>
<tr>
<td>dt</td>
<td>0.002</td>
</tr>
<tr>
<td>tstart</td>
<td>10.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actor</th>
<th>Fire</th>
<th>Elapsed time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ualinit</td>
<td>1</td>
<td>00:00:01.722</td>
</tr>
<tr>
<td>shot</td>
<td>2</td>
<td>00:00:00.000</td>
</tr>
</tbody>
</table>
Sharing workflows

- Web based GUI for saving/sharing workflows (.xml/.kar) with the metadata and additional files (based on Liferay)
- Possibility of the workflow parameters modification
- Downloading and sharing of the workflows between users
- Searching through the workflows (name, metadata, etc)
Sharing workflows – group management

- Private group for each user
- Creation of the new groups
  - Rights management
  - Adding/removing users
GUI for running workflows in distributed environment

- Liferay portlet developed, using VineToolkit/QCG Grid middleware
- Central installation concept used
- Kepler transparent from user perspective
Summary
Further developments

- Improvement of the provenance in order to report the firing of composite actors.
- The exposure of the composite actors as classes instead of instances in an automatic way.
- Address a need for automated tools could perform export/import functionality while using the API.
- User friendly GUI to steer the workflow execution/provide interaction
Summary

Presented tools, methods and services enhancing the usage of Kepler-based scientific workflow framework.

This research has been conducted by scientific communities that make intensive usage of Kepler framework.

Users' requirements have been addressed by development of tools and services that could be mostly reused by different user communities.
Acknowledgment

- Co-authors:
  - Marcin Plóciennik, Michal Owsiak, Piotr Spyra, Bartek Palak, Tomasz Żok, PSNC, Poland
  - Szymon Winczewski, Jarosław Rybicki, TASK, Poland
  - Pawel Ciecielag, Wojciech Pych, CAMK, Poland
  - Frederic Imbeaux, Bernard Guillerminet, Philippe Huynh, Thierry Aniel, CEA France

- This research has been partially supported by the European Regional Development Fund program no. POIG.02.03.00-00-096/10 as part of the PL-Grid PLUS project.

- Work in the EFDA ITM-TF was funded by EURATOM through the European Fusion Development Agreement and the Participating Countries and received support by financial resources for science in 2013 and 2014 awarded for the implementation of the international project co-funded by the Ministry of Science and Higher Education in Poland.