Procedia Computer Science

Volume 29, 2014, Pages 1733–1744

ICCS 2014. 14th International Conference on Computational Science

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

Marcin Płóciennik[1], Szymon Winczewski[2,5], Paweł Ciecieląg[3]
Frederic Imbeaux[4], Bernard Guillerminet[4], Philippe Huynh[4]
Michał Owsiak[1], Piotr Spyra[1], Thierry Aniel[4], Bartek Palak[1]
Tomasz Żok[1], Wojciech Pych[3], and Jarosław Rybicki[2,5]

[1] Poznań Supercomputing and Networking Center, Poznań, Poland
{marcinp, pspyra, michalo, bartek,tzok}@man.poznan.pl
[2] Faculty of Applied Physics and Mathematics, Gdańsk University of Technology Narutowicza 11/12, Gdańsk, Poland
swinczew@mif.pg.gda.pl, ryba@pg.gda.pl
[3] N. Copernicus Astronomical Center, Polish Academy of Sciences, Warsaw, Poland
{pci, pych}@camk.edu.pl
[4] CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France
{frederic.imbeaux,bernard.guillerminet,philippe.huyhn,thierry.aniel}@cea.fr
[5] TASK Computer Centre, Gdańsk University of Technology, Narutowicza 11/12, Gdansk, Poland

Abstract
Scientific workflow systems are designed to compose and execute either a series of computational or data manipulation steps, or workflows in a scientific application. They are usually a part of a larger eScience environment. The usage of workflow systems, however very beneficial, is mostly not irrelevant for scientists. There are many requirements for additional functionalities around scientific workflows systems that need to be taken into account, like the ability of sharing workflows, provision of the user-friendly GUI tools for automation of some tasks or submission to distributed computing infrastructures, etc. In this paper we present tools developed in response to the requirements of three different scientific communities. These tools simplify and empower their work with the Kepler scientific workflow system. The usage of such tools and services is presented on Nanotechnology, Astronomy and Fusion scenarios examples.

Keywords: Kepler

1 Introduction

Scientists face nowadays complex problems that usually require using a variety of analysis and simulation tools. In many cases scientists are provided with the whole scientific workflow environment that is a traceable and reproducible tool for solving their scientific challenges.

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

© The Authors. Published by Elsevier B.V.
doi:10.1016/j.procs.2014.05.158

Scientific workflow systems, in general, are designed to compose and execute either a series of computational or data manipulation steps, or workflows in a scientific application. The usage of workflow systems, though very beneficial, is mostly not irrelevant for scientists. There are many requirements for simplification and additional functionalities around scientific workflows systems (like the ability of sharing the workflows, provision of user-friendly GUI tools for automation of some tasks). In this paper we present a set of tools and services developed in response to the requirements of three different scientific communities that have been using one of the most common scientific workflow systems called Kepler [12]. These developments enhanced standard capabilities of the Kepler framework. The exploitation of tools and services are presented on Nanotechnology, Astronomy, and Fusion scenarios examples.

The rest of the paper is composed as following: section 2 presents the requirements coming from different user communities on the scientific workflow system in the context of eScience environments. Section 3 presents the tools and services developed to fulfill various requirements of the user communities using the Kepler workflow system. In section 4 we present different use cases coming from three scientific fields, that make use of the developed tools. In section 5 the future work is discussed and we summarise the performed work.

2 Requirements

There is a number of common requirements collected by us, raised by user communities of Kepler workflow system (we recognise here different roles of users: end user, workflow developer, platform provider).

The first one concerns the need for a user friendly GUI tools that would hide the complexity of the workflows and the framework itself. Users that are not developers of the workflows, would mostly like just to be able to specify initial parameters, execute workflow and retrieve results. Users expect also the whole preconfigured environment which minimises the time needed on configuration. Such a requirement concerns both the local runs as well as the cluster or distributed computing infrastructures. Depending on the cases users prefer either local tools or web based solutions. Some of the scientific workflow systems like [1] provide web based forms that are simplifying the proccess of running the workflows. In our case we needed tools and services very specific for the Kepler system.

The next request refers to collaboration possibilities like sharing, tracking and versioning Kepler actors and workflows. Workflow developers that are building the scenarios composed of native codes written in many different languages require the possibility of easy generation of Kepler actors that run these applications. Working with large workflows may trigger issues of proper understanding of data and control flow in order to find the bottlenecks and places where optimisation should be applied. In this kind of situation one would typically use profiler tool in order to locate places where time consumption has the highest values. There are several tools used together with various scientific workflow systems analised, as an example using Kickstart [15] profiling tool, which collects performance statistics and provenance information from the execution and was used with the Pegasus [16, 15]. Kickstart does not collect however very detailed information.

Many systems have been integrated with performance monitoring tools [17, 4] ,however, these tools focus rather on the high level information like the runtime tasks or data size. Paratrac [5] system is very detailed, but based on the FUSE filesystem, that is not good for large I/O. None of these systems that we have analised could provide detailed information on the Kepler actors execution level.

MOST WIEDZY Pobrano z mostwiedzy.pl

Another requirement is related with the eScience platform providers that would like to have the ability to have an easy way to maintain the software installation (like Kepler).

3 Tools and services

3.1 Sharing the Kepler workflows

As part of dedicated eScience portal services for Astronomy, so called "pipeline sharing portlet" has been developed. Pipeline sharing portlet is a pluggable user interface software that is managed and displayed in a web portal based on Liferay [3]. The key feature of this application is to give users the ability to collaborate on the same Kepler workflows, to speed up the process of solving common issues. A user can upload the workflows and then, through the web interface, modify workflow's global parameters, save changes and download modified workflow. A user is also able to share the workflows with another user or a group of users. A group administrator can modify members' permissions, for example to restrict the rights 'only to read' for some groups of users. There is also a possibility to describe every workflow, add some metadata tags to simplify searching process and also add some extra files, like screenshots etc. Figure 1 shows main features of this web portlet. The first tab is used to upload/download workflows in XML and KAR format. After uploading workflow parameters are detected and can be changed from the portlet. A user can also share the workflow and metadata within working groups. The groups can be created and managed via the group manager.

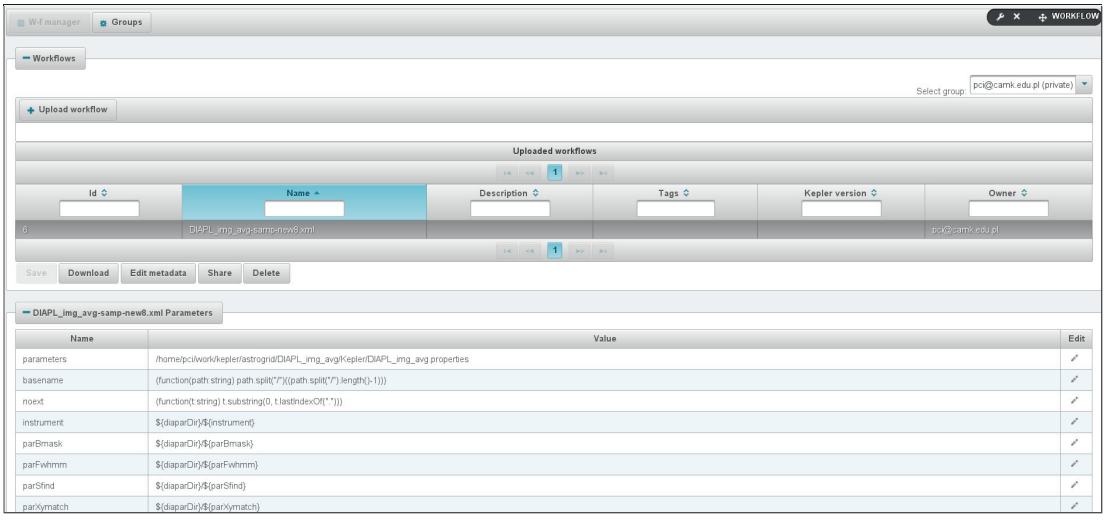

Figure 1: Web applet for sharing Kepler workflows.

3.2 Actor generators

3.2.1 C actor generator for general purpose codes

We have also developed the tool for generating the Kepler actors from any C code that runs as a standalone executable. It is a real integration, not a replacement for Kepler Exec actor which runs binaries. Exec actor creates a new process upon each firing (with a new context,

MOST WIEDZY Pobrano z mostwiedzy.pl

environment and other OS-related overhead associated with process spawning). On the other hand, the automatically generated actors make use of dynamically loaded libraries which remain in Kepler process space throughout the workflow execution. This allows to reuse a once established context and may speed up the workflow execution in scenarios with a huge number of executions of the C code. T his tool generates an actor with ports mapped to code arguments. This actor uses Kepler tokens (e.g. `DoubleToken`).

Technically, codes automatically becomes a shared library loaded by the actor on request. The principle is to take an existing code (with `int main()` function) and use our tool to generate an equivalent in the form of Kepler actor. The only extra effort is to prepare a short XML of the following form:

```
<actor>
<inputs>
<in type="String" name="myArgument1"/>
<in type="Integer" name="myArgument2"/>
<in type="Double" name="myArgument3"/>
</inputs>
</actor>
```

It defines the type and name of the expected inputs. Let's say there is a program used like this: `myapp test 123 1.0`. With the above XML, an actor will be generated with three typed ports. The above program execution would be equivalent to passing `StringToken` *test* to port `myArgument1`, `IntToken` 123 to `myArgument2` and `DoubleToken` 1.0 to `myArgument3`. Since the common practice in C codes is to explicitly call `void exit(int status)` instead of making `int main()` return the exit code, we also handle such a situation without changes in codes itself.

3.2.2 Generation of components from their physics codes (FC2K, HPC2K, WS2K)

FC2K(Fortran Code to Kepler), HPC2K (HPC to Kepler), WS2K (Webservices to Kepler) are a family of tools developed for wrapping the Fortran or C++ source code into Kepler actor that can run in different computational context. Such an actor contains the data wrapper used to access locally or remotely standardized data structures used as interface between coupled physics components that is discussed in more details in section 4. The precondition for using these tools is that the codes are using CPOs as input/output. WS2K addresses the distributed resources that can be accessed via web services. The Web service infrastructure allows to see physics models in fusion simulations as components described by a WSDL file and stored in a Web server. Most of the fusion simulation codes are written in FORTRAN and the development of web services in FORTRAN is not straightforward so the WS2K tool has been developed. It transforms the FORTRAN or C++ code into a web service, adds data wrapper for accessing remote (typicaly complex) data structures, installs web service inside a web server, adds Java actor into Kepler. The generated actor is able to access web service prepared by WS2K. The codes included in the actor are a part of the Java code within KEPLER. Using WS2K, the codes are stored separately and possibly remotely in a HTTP server. FC2K gives similar functionality but addresses the local execution or the local batch system. The goal of the HPC2K is to convert a physics code written in languages like Fortran or in C++ to the Kepler actor that automatically handles the jobs on the HPC or HTC resources. The code must initially be compiled as a library with a main, single function that corresponds to the execution of the code. From a user point of view, HPC2K provides simple graphical user interface that allows

MOST WIEDZY Pobrano z mostwiedzy.pl

to define parameters such as the code arguments, the location of the code library, the remote target system, etc. The composite Kepler actor generated by HPC2K sends the input files to the execution host, prepares a description of the job and resources required for execution. After a job is executed, it monitors job status, and collects the outputs after a job execution.

This composite actor is based on the Serpens [13] suite modules. These modules provide a remote execution and job handling mechanism. The general idea for the composite Kepler actors generated by FC2K, HPC2K or WS2K is that they allow to implement a special paradigm of code interoperation. Physics code developers are required to make the I/O of their code compliant with the standardized data model and then using Kepler it is possible to design complex workflows with sophisticated dependencies.

3.3 Central installation of the Kepler

In order to simplify management of the Kepler framework maintenance and usage, we have introduced the concept of central installation of the Kepler. The basic idea is to have one installation of the Kepler, and the scripts enabling to use just a shadow copy of that installation with only working directories (.kepler, KeplerData with user actors and workflows) specific for each user installed in the user area. Each of the user besides links to the common actors have their own generated actors in the local directories (so users share only the common set of actors and libraries, and the rest are unique per user).

Since each of our users has had a different set of the actors so far, each of the users had its own installation of Kepler (standard+own actors) which was hard to maintain in a longer perspective.

Central installation of Kepler (common for many users) is very beneficial: in the shared filesystem/computing environment where many users run the Kepler (in particular the cluster environments). Central installation decreases space used, since the Kepler is installed only in one place and users have only symbolic links and their local folders like .kepler and KeplerData, it allows to control more easily the Kepler version used by users and manage the updates and fixes. For this purpose we have developed and deployed scripts that make use of these concepts. The detailed usage scenario is described in section 4

3.4 Provenance enabled monitoring and profiling

Kepler provides provenance framework that can be used as execution storage and later on used to re-execute the workflow with the same parameters. However, thanks to the way the Kepler Provenance Recorder is designed, it is also possible to use it in the context of workflow profiling. Working with large workflows may trigger issues with proper understanding of data and execution flow. Having complex flows including multiple and nested composite actors, actors that wrap native code, makes it hard to determine how workflow is actually executed. Finding bottlenecks and places where optimisation should be applied might be a challenging task. In this kind of situation one would typically use the profiler tool in order to locate places where time consumption has the highest values. In order to deal with large workflows, we have developed ITM Workflow Profiler that is based on standard Provenance Recorder, however, it adds some modifications into it. A few assumptions had been made before the final solution was provided. These were: reporting iteration of the internal loop counters should be reported, global parameters should be stored in the output, each actor during its execution should store input and output values, execution of native codes should be reported in the output. Additionally, there were solution specific requirements, bound to the data transfer specifics. Eventually, as the monitoring of the workflow was supposed to be done live, there

MOST WIEDZY Pobrano z mostwiedzy.pl

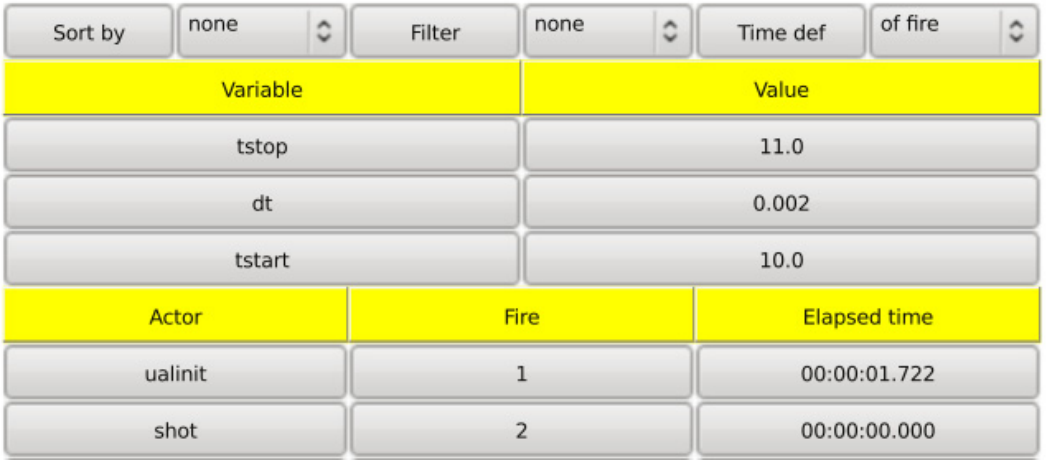

Figure 2: MWRKF profiling tool

were requirements to the output format as well. 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. The external tool can read input data and analyse it as well formed XML element, while at the same time it doesn't require the whole document in order to parse it. The data read by an analyser are reported within a presentation of a part of the tool (called MWRKF) browsable from the web. This way, person executing workflow can analyse the results live and determine elements that are most time consuming. MWRKF browser visualises the output of the Provenance recorder (Workflow Profiler's result). MWRKF is able to read and visualise the XML output during the workflow execution and, of course, after completion of it. The MWRKF output is refreshed periodically during the execution. The GUI is a CGI application written in C-langage. The first webpage permits a user to select the output file of the Provenance recorder to be visualised. The monitor tab gives access to the second webpage which presents the content of global parameters and the list of actors of the workflow as in the figure 2. For each actor the number of actual fires and the elapsed time of the last fire (the tab following « Time def » permits the user to paste cumulated elapsed time instead of elapsed time of the last fire) are given. 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 such a composite actor tab gives access to the content of this composite actor. Clicking on its child pastes the values of input and output ports (and multi ports if any) for the current fire. ITM Workflow Profiler has been tested with some ITM workflows (e.g. for workflow of integrated modelling of Heating and Current Drive [2]), and is currently under deployment phase on the ITM infrastructure. Full utilisation and possible workflows optimisation (thanks to profiling) is expected in the next step.

3.5 Running the Kepler workflows in Grid environment via user friendly GUI

For the purpose of fulfilling the Nanotechnology requirements we have developed portlets base managed and displayed in a liferay web portal. The key feature of this application is to give a researcher the opportunity to perform multiple similar tasks simultaneously with different

MOST WIEDZY Pobrano z mostwiedzy.pl

parameters on remote environments. Computing power delivered by the grid infrastructure significantly enhances a single user capabilities of scientific research. The portlet is composed of 3 main parts, which define a typical use case: case list, description form, monitoring tab. It has been written in Adobe Flex and is embedded in the Liferay portal. All components are connected by Vine Toolkit – a modular, extensible Java library that offers high-level API for grid-enabling applications. When the job is submitted a server receives the parameters, and composes the QCG job description. QCG is one of the grid middleware which offers an advanced job and resource management capabilities including mapping, execution and monitoring capabilities. The job profile describes: where the input files are (configuration files, setup scripts, etc); how to start the execution environment (multiple-kepler.sh bash script); where to transfer the output files. After the job is submitted QCG finds the most appropriate site in the grid, where the job can be executed (and Kepler is installed). In the next stage QCG executes the bash script multiple-kepler.sh. Kepler must be installed on the site to run the application. We use here the concept of the central Kepler installation on clusters described in the section 3.3, that results in making creating links.

The application is executed as a Kepler workflow, which is composed of a set of actors, based on C/C++ or other languages applications connected with each other. Because users are able to submit many of such jobs simultaneously, we give a possibility to run multiple instances of Kepler at the same time on the same nodes, taking care of customising the .kepler and home directories (customised in multiple-kepler.sh script). Since applications can end up in different running environments, each time the multiple-kepler.sh script regenerates all the native codes, creating and installing new actors required by the workflow in the shadow copy of the Kepler on each node, using the mechanism described in the section 3.3.

4 Use cases

4.1 Nanotechnology: Anelli

Atomistic simulation methods such as molecular dynamics (MD) are nowadays routinely used to study materials. As an output from the simulation programs one usually obtains the positions of atoms within the simulation box. Such information on the system, although theoretically complete, requires further data processing, e.g., the calculation of radial and angular distribution functions. Significant difficulties arise when disordered systems (e.g. glasses and metallic 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. Ring analysis is one of the approaches for analysing medium-range order in computer-simulated solid state. The principal aim of ring analysis is to obtain information on the rings and chains presented in the system, which is thus viewed as a graph. The atoms comprising the system correspond to nodes in the graph, while the bonds between them correspond to edges. The ring analysis consists of identifying all rings and chains in the graph. Even though the ring analysis is relatively easy to formulate, its practical implementation is highly nontrivial and it employs a number of specific algorithms and computational strategies. A software suite called ANELLI has been designed with the aim of performing ring analysis. It has been described in detail in [10]. The suite is composed of eight individual codes: fnlg, ggsplit, gbi, anelli, frs, recover, geom and vis. The dependencies between those codes are schematically shown in figure 3. The individual codes are briefly described below with the aim to characterise the approach as a whole. Structural analysis starts with the calculation of the adjacency matrix (i.e. the identification of the edges connecting the nodes). This is done by means of the fnlg program, which constructs the neigh-

MOST WIEDZY Pobrano z mostwiedzy.pl

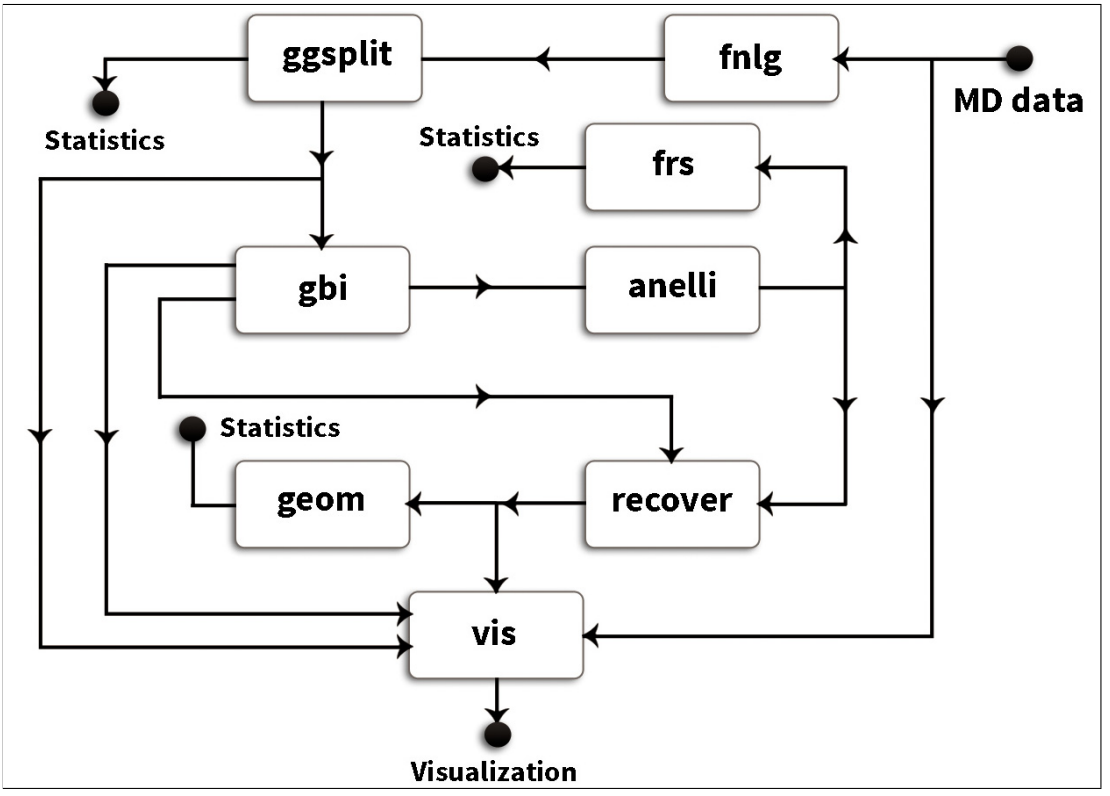

Figure 3: Anelli suite flow chart.

bour lists for all the atoms in the system. In the next step the full bonding graph is split into subgraphs by means of the ggsplit program. Subsequently, the gbi program generates binary input files for the anelli program. Since the execution time and memory requirements of the anelli program depends exponentially on the subgraph size, the elimination of all dangling structures (nodes which do not belong to any ring or any path between rings) is carried out at this stage. The main part of the analysis is performed by means of the anelli program, which finds all the basal rings using the Balducci-Pearlman-Mancini algorithm. It is important to note that ring detection is performed separately for each subgraph identified by the ggsplit program. This introduces the need to calculate full (i.e. global) ring statistics, which is done by the frs program. Since the anelli program operates on reduced-graph node numbers, it also becomes necessary to restore the original atom indices. This is done by the recover program. At the very end of the analysis, the geometrical properties of the detected rings are computed and the data files for visualisation programs are produced. This is done by the geom and vis programs respectively.

It is apparent that performing ring analysis can be complicated and tedious with the original approach, especially for new users, due to the need for repeated manual executions of individual programs comprising the suite in order to carry out the analysis for subsequent subgraphs, before the final results can be extracted. This process can be, however, vastly simplified and automated using scientific workflows like Kepler, with only the following tasks remaining for the end user to carry out: specifying the configuration to be analysed (providing an input configuration file

MOST WIEDZY Pobrano z mostwiedzy.pl

that specifies the number of atoms in the system and their positions), specifying the parameters for the calculation (such as a cutoff radius, i.e. the maximum distance that can separate two nodes connected by an edge), starting calculations, obtaining results. In addition, in order to study different analysis of the usage of larger computing facilities like grid can speed up the process for end users. This certainly needs a user friendly environment that will hide both the scientific workflow layer as well as the grid layer.

The workflow that has implemented the above mentioned dependencies, has been developed using the Kepler workflow system. In addition, this use case fully utilised the tools and development mentioned in the previous section, namely: C actor generators, for embedding the codes into Kepler, the central installation of the Kepler on cluster inside the PL-GRID infrastructure, and the usage of the portlets for specifying parameters, submission, monitoring of the jobs and data in/out handling, providing easy to use environment for the end users, that hid all technical details. The technical implementation of the use case for Anelli has been described in the section 3.5.

4.2 Astronomy: DIAPL

DIAPL [18] is an astronomical package devoted to photometry using a differential image analysis method. It is particulary useful for photometry of variable stars in crowded fields, where it's hard to discriminate between objects. The basic idea of the method is to create an averaged image from a series of input images and then substract it from each of them. By reducing random fluctuations and removing constant background one is able to detect even faint, variable stars which would be missed while using other methods. The DIAPL package consists of a number of command line programs in C language which are typically invoked from shell scripts. The scripts can be used in various configurations depending on the problem. In typical applications, users have to spend substantial amount of time handling intermediate steps manually. We used the Kepler environment to support them in this work. We have implemented package programs as library of Kepler actors and prepared a few ready-to-use workflows. The programs are converted into Kepler actors with the 'C actor generator' described in section 3.2.1. This approach is very flexible since the DIAPL package is under constant development and a user is able to easily upgrade Kepler actors. For example, the DIAPL program performing actual image averaging is called with arguments as follow: `'template image_names_file output_image x_nim y_nim'`. In order to generate a Kepler actor with corresponding parameters one has to provide only simple xml file:

```
<actor>
<inputs>
<in type="String" name="imageNamesFile"/>
<in type="String" name="outputImage"/>
<in type="Integer" name="xNim"/>
<in type="Integer" name="yNim"/>
</inputs>
</actor>
```

Figure 4 displays the most basic workflow which is typically the first step in analysis with DIAPL. It takes a series of images of roughly the same sky region, transforms them to common grid (non-linear effects are accounted for) and averages. The purpose of this procedure is to increase signal to noise ratio with respect to a single frame and to remove bad pixels (camera hot pixels or traces of cosmic rays) from the image. There are two important aspects of this workflow: interactivity and support for external, domain-specific software. For example,at the beginning a user is presented with input images for visual inspection and has an option to

MOST WIEDZY Pobrano z mostwiedzy.pl

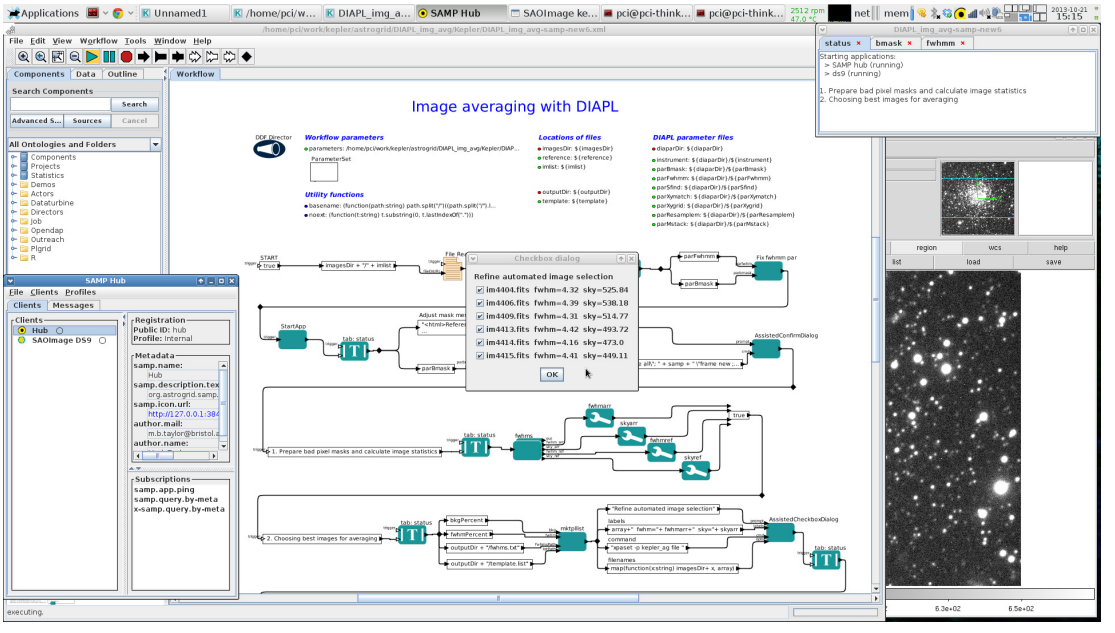

Figure 4: Astronomical image averaging in Kepler using DIAPL package.

refine their selection and mark additional regions of bad pixels. The selection is implemented as a standard checkbox list, but, in addition, any change on the list loads corresponding image into external viewer application - ds9. ds9 is a popular astronomical program supporting asynchronous control over SAMP protocol (Virtual Observatory standard). Thanks to this solution, astronomers can use familiar software to inspect the image and mark regions of bad pixels. After the checkbox selection is acknowledged the coordinates of marked region are read into Kepler via the SAMP protocol. All interactive features are optional since a typical usage pattern is to process a small subset of images manually in order to adjust many parameters and then run the rest in a batch mode.

4.3 Nuclear Fusion: Integrated Tokamak Modelling workflows

The EFDA Integrated Tokamak Modelling Task Force (ITM-TF) was "coordinating the development of a validated suite of simulation tools for ITER and DEMO plasmas" [6, 14, 11] . ITER is the next generation of fusion devices intended to demonstrate scientific and technical feasibility of fusion as a sustainable energy source for the future. To exploit full potential of the device and to guarantee optimal operation for the device, a high degree of physics modelling and simulation is needed even in the current construction phase of the ITER project. The modelling tools are aimed at general use, in the sense of a device independent approach to data structures, and data access in order to allow cross validation between different fusion devices. The community has developed standardised descriptions of the data called Consistent Physical Objects [9]. In addition, a set of libraries, called Universal Access Layer (UAL) used for accessing and exchanging the CPOs has been developed. UAL API is accessible via different programming languages like Fortran, C++, C, Python, Java and also Matlab. In order to standardise environment for the codes that are in different languages and enable easier con-

MOST WIEDZY Pobrano z mostwiedzy.pl

struction of more complex dependencies, the Kepler workflow orchestration system has been chosen to construct the number of physics workflows. A number of very complex workflows has been developed: European Transport Solver (ETS) [7, 8], Turbulence-transport, Equilibrium reconstruction and MHD stability chain and many others that are described in more details in [11] . Workflows have been created using the presented tools including HPC2K, FC2K to include the physics codes written in different languages and run them in the context of different computational resources. A very low granularity has been envisaged for the workflows: as an example the ETS exposes the internals of transport solver algorithms (not detailed calculations but convergence and time evolution loops), which resulted in multi level, workflows with hundreds of actors. For further development and management of such complex workflows the usage of the profiling tool described in section 3.4 is of great importance. Also, since there is a large number of workflows and users, the whole platform is moving currently towards the central Kepler installation. Supplemented with additionally developed tools described above, Kepler provides a dynamic flexible and extensible modelling environment for fusion modelling.

5 Conclusions and future work

In this paper we have 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. A number of users' requirements have been identified and addressed by development of general purpose tools and services that could be mostly reused by different user communities. These tools include actor generators (HPC2K, FC2K, WS2K, C generator), web based service for sharing workflows and running them in a distributed environment, provenance enabling monitoring and profiling and Central installation of Kepler. All the tools have been deployed in production environments like the ITM-TF platform or PL-GRID infrastructure and provided to user communities, being a base for application deployments. Furthermore, the scenarios (Nanotechnology, Astronomy and Nuclear Fusion) making use of the tools and services have been described. There are still many requirements around Kepler scientific workflows that have not been solved yet and require further development. One example is improvement of the provenance in order to report the firing of composite actors. Another aspect is related to the exposure of Kepler and composite actors as classes instead of instances in an automatic way. There is also a need for automated tools could perform export/import functionality while using the API.

Acknowledgments

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.

References

[1] Apache Airavata. `http://airavata.apache.org/`. [Online;].

[2] IMPhcd workflow. `http://www.efda-itm.eu/ITM/html/imp5_imp5hcd.html`. [Online;].

[3] Liferay. `http://www.liferay.com`. [Online;].

MOST WIEDZY Pobrano z mostwiedzy.pl

[4] S.M.S. da Cruz, F.N. da Silva, L.M.R. Gadelha, M.C. Reis Cavalcanti, M.L.M. Campos, and M. Mattoso. A lightweight middleware monitor for distributed scientific workflows. In *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pages 693–698, May 2008.

[5] Nan Dun, Kenjiro Taura, and Akinori Yonezawa. Paratrac: A fine-grained profiler for data-intensive workflows. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 37–48, 2010.

[6] A. Bécoulet et al. The way towards thermonuclear fusion simulators. *Computer Physics Communications*, 177(1-2):55–59, 2007.

[7] Coster et al. The european transport solver. *IEEE Transactions on Plasma Science*, 38, 2010.

[8] D. Kalupin et al. Numerical analysis of jet discharges with the european transport simulator. *Nuclear Fusion*, 53, 2013.

[9] F. Imbeaux et al. A generic data structure for integrated modelling of tokamak physics and subsystems. *Computer Physics Communications*, 181(6):987–998, 2010.

[10] G. Bergmański et al. A new program package for structural analysis of computer simulated solids. *TASK Quarterly : scientific bulletin of ACC in Gdansk*, Vol. 4, No 4:555–573, 2000.

[11] G. Falchetto et al. The european integrated tokamak modelling (itm) effort: achievements and first physics results. *Nuclear Fusion submitted*, 2014.

[12] Ludäscher et al. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[13] M. Płóciennik et al. Approaches to distributed execution of scientific workflows in kepler. *Annales Societatis Mathematicae Polonae. Fundamenta Informaticae*, Vol. 128, nr 3:281–302, 2013.

[14] P.I. Strand et al. Simulation and high performance computing—building a predictive capability for fusion. *Fusion Engineering and Design*, 85(3–4):383 – 387, 2010.

[15] Y. Zhao E. Deelman M. Wilde J. Voeckler, G. Mehta. Kickstarting remote applications. 2006.

[16] Gideon Juve, Ann L. Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Characterizing and profiling scientific workflows. *Future Generation Comp. Syst.*, 29(3), 2013.

[17] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, Thomas Fahringer, and Alexandru Iosup. Workflow monitoring and analysis tool for askalon. In Ramin Yahyapour, Domenico Talia, and Norbert Meyer, editors, *Grid and Services Evolution*, pages 1–14. Springer, 2009.

[18] W. Pych. DIAPL. `http://users.camk.edu.pl/pych/DIAPL/`. [Online;].

MOST WIEDZY Pobrano z mostwiedzy.pl