DOI: 10.14529/jsfi170102

Workflows for Science: a Challenge when Facing the Convergence of HPC and Big Data

Rosa M. Badia[1,2]***, Eduard Ayguade***[1,3]***, Jesus Labarta***[1,3]

Workflows have been traditionally a mean to describe and implement the computing experiments, usually parametric studies and explorations searching for the best solution, that scientific researchers want to perform. A workflow is not only the computing application, but a way of documenting a process. Science workflows may be of very different nature depending on the area of research, matching the actual experiment that the scientist want to perform. Workflow Management Systems are environments that offer the researchers tools to define, publish, execute and document their workflows.

In some cases, the science workflows are used to generate data; in other cases are used to analyse existing data; only in a few cases, workflows are used both to generate and analyse data. The design of experiments is in some cases generated blindly, without a clear idea of which points are relevant to be computed/simulated, ending up with huge amount of computation that is performed following a brute-force strategy.

However, the evolution of systems and the large amount of data generated by the applications require an in-situ analysis of the data, thus requiring new solutions to develop workflows that includes both the simulation/computational part and the analytic part. What is more, the fact that both components, computation and analytics, can be run together will enable the possibility of defining more dynamic workflows, with new computations being decided by the analytics in a more efficient way.

The first part of the paper will review current approaches that a set of scientific communities follow in the development of their workflows. This paper does not intent to be exhaustive in the compilation of different approaches available to develop and deploy workflows. We focus on the Workflow Management Systems used by a set of scientific communities and their representative use cases, with the objective of understanding their different needs and requirements. The second part of the paper proposes a new software architecture to develop a new family of end-to-end workflows that enables the management of dynamic workflows composed of simulations, analytics and visualization, including inputs/outputs from streams.

Keywords: workflows, scientific applications, Big Data.

Introduction

Workflows appeared last century and have been used in the manufacturing industry as a mean to optimize their processes. Examples of traditional (non-IT) workflows can be found in the assembly lines, i.e., the Ford Model T assembly line standardized the production processes and was the first continuous delivery pipeline for the automotive industry. This process reduced the costs of manufacturing from $850 to $260 in 1924.

The time and motion studies defined by Taylor [52] and Gilbreth [41] had significant impact in the manufacturing processes. These studies proposed to break manufacturing activities into small, simple steps, to determine with accuracy the amount of time required to perform each of the steps. Then, the sequence of movements taken by the employee has to be carefully observed to detect and eliminate redundant or wasteful motion, and the precise time invested for each correct movement is measured. From these measurements, production and delivery times and

[1]Barcelona Supercomputing Center (BSC), Barcelona, Spain
[2] Consejo Superior de Investigaciones Científicas (CSIC), Madrid, Spain
[3] Universitat Politècnica de Catalunya (UPC), Barcelona Spain

prices can be computed and incentive schemes devised. Methods used in these early times were: Flow diagrams, Gantt charts, and ERT charts.

Although the term workflow was not used at that time, the same concept is used in current Workflow Management Systems, a software system that is able to orchestrate a set of tasks. The tasks show dependencies between them, which can be of data or control, forming a task graph or workflow. The concept of workflow is used extensively in a large number of scientific communities.

Scientific users have a plethora of Workflow Management Systems available for their needs. Traditionally different communities stick to a system or to a set of systems for different reasons: due to the needs of the community applications, due to the popularity of given systems, due to historical reasons, to availability of domestic systems that are adopted by others and later extended, due to the possibility of sharing, avilability of specific functionalities that are needed by the community applications not present in others, etc. However, we believe that aspects such as modularity and elegance of the design, portability, genericity of the systems; should be given more attention.

The paper takes into account a set of Workflow Management Systems used by given scientific communities to implement their workflows: life science (genomics), earth-science (climate), fusion, and astrophysics. For each of them an example of how the workflows are defined and the specific features they have is described.

It is very usual that these scientific applications generate a large amount of data, and this is in-crescendo. Also, the use of parallel systems and High Performance Computing (HPC) is every time more usual. Traditionally, the phases of computation/simulation of these workflows have been decoupled from the phases of data analysis. Also, traditionally workflows are defined quite statically, even loops are possible, but no margin for dynamicity on the decision of what computations should be performed is left.

Taking into account potential users of next coming exascale architectures, workflow management systems that support the convergence of the computation and data analysis parts are a must. Even more, those workflows should support in-situ data-analysis and dynamism, in such a way that results from previous analysis determine the next steps of the workflow, i.e., which computation to trigger, searching for new alternatives or going in-depth into a more detailed simulation.

In section 1 we give an overview of the alternatives in the implementation of a Workflow Management System. Then, the paper is organized around the different cases that have been chosen: section 2 describes Kepler, and its usage by the fusion community; section 3 describes Pegasus, and the case of the LIGO collaboration that has been using this system for more than 10 years; section 4 describes Galaxy and its use in the framework of the Life Sciences community; section 5 describes the workflow management systems used by the Earth Science (climate) community; and section 6 describes Taverna and its use by the astrophysics community. Section 7 proposes a new architecture of end-to-end workflows with dynamic management, orchestrating the computation and analytics of the experiments. Final section concludes the paper.

1. Workflow Management Systems: an Overview

A Workflow Management System can be defined as a software environment able to orchestrate the execution of a set of interdependent computing tasks that exchange data between them with the objective of solving a given experiment. A workflow can be graphically described as

a graph, where the nodes denote the computations and the edges data or control dependencies between them.

Workflow Management Systems became very popular with the appearance of Grid computing, since they offered the possibility of exploiting this distributed infrastructure. Papers [14, 57] present taxonomies of Workflow Management Systems from that period. Some of the systems developed at that time are still alive projects used in current distributed computing platforms (either High Performance Computing (HPC) clusters, High Throughput Computing (HTC) platforms, clouds or combination of several of these options).

Workflows can be described graphically, with a drag and drop interface where the workflow is totally specified with a graphical interface by the user like in Kepler [6],Taverna [26], or Galaxy [3]. It can be described textually, by specifying the graph in a textual mode, indicating the nodes and its interconnections like in Pegasus [15] or ASKALON [21]. It can also be described programmatically, using all the flexibility of a programming language to describe the behaviour of the workflow that is dynamically built depending on the actual dependencies found by the workflow system like in PyCOMPSs/COMPSs [32] or Swift [55]. A particular case of this is the use of simple tagged scripts that are processed by the actual engine, like with Cylc [39], Autosubmit [34], or ecFlow [33]. Another alternative is to describe the workflow through a set of commands with a command interface, like with Copernicus [42]. With the objective of offering a single syntax to describe workflows the initiative of the Common Workflow Language [7] has appeared. The Common Workflow Language (CWL) is a working group consisting of various organizations with interest in portability of data analysis workflows, mostly oriented to bioinformatics tools and with an emphasis on systems enabled with Docker. CWL offers a syntax to connect command line tools in order to create workflows that can be used by multiple platforms. CWL follows JASON or YAML syntaxes, or a mixture of the two.

Some systems orchestrate already deployed web services (Taverna), others compose external binaries or tools (Galaxy), and a few are able to interoperate directly with methods described in programming languages (PyCOMPSS/COMPSs). The data exchanged between the computation nodes of the workflows is typically a file, although in some cases can be objects in memory (like in PyCOMPSs/COMPSs).

A key component in a Workflow Management System is its engine. The engine is the responsible for coordinating the execution of all the tasks, scheduling them in the available computing resources and storage devices, transferring the data between distributed storage systems, monitoring the execution of the tasks, etc. The information that can be obtained about the engine in the literature is very variable: while for some systems (i.e. Pegasus, PyCOMPSs/COMPSs or Swift) the bibliography details sophisticated engines that implement various optimizations, either to schedule in parallel the workflow to be executed, to improve data locality, to be able to exploit heterogeneous computing platforms, ...; for others the information is very scarce and difficult to find.

On the user side, aspects that are valued by the scientific community are the possibility of sharing their workflows and data, and the support for workflow provenance. Several systems report the existence of repositories for workflows or experiments, like the myExperiment [23] repository, which currently supports inputs from several systems (Taverna, Galaxy and Kepler), or HUBzero [37] a software platform to support collaborations that is able to launch Pegasus workflows.

Another characteristic of these systems is the computing platform where the workflows are executed. As said before, many systems began their developments with the Grid as a computing platform, and still are able to run in this type of platforms, like the OSG [40] or EGI [30]. Most systems can execute in distributed environments (either composed of regular servers/clusters or HPC systems), also support for Clouds is common, and some systems are starting to support containers. While in most scientific communities the workflow tasks have been mostly sequential, the trend in general is to take benefit of current multicore architectures and accelerators such as GPGPUs or FPGAs, including tasks in the workflows that require some level of parallelism although with a low degree and only intranode (up to a few threads), other communities have been using large clusters or supercomputers for part of their workflow tasks (like in the climate or fusion communities). The trend in general is to take benefit of current multicore architectures, including tasks in the workflows that require some level of parallelism.

Given the amount of systems available, there have been some interoperability initiatives, like the European project SHIWA [49] and its continuation ER-flow [18] that dealt with interoperability of a dozen of workflow management systems existent at that time. The SHIWA simulation platform consists of a repository that supports the storage of workflows and metadata and of a portal that includes a workflow engine able to orchestrate workflows from different systems.

2. Kepler - Fusion community

The Kepler system [6] is free and open source, and developed, supported and maintained by the Kepler Project [29]. Kepler is a successor of the Ptolemy II system [43], and was designed to help users to create workflows, to perform analysis, to share and reuse workflow components, models and data between scientists. It was not designed to fulfill the needs of a specific community. With regard to data, Kepler is interoperable with a variety of formats, and supports local and remote data-access. The Kepler Project claims that the system is an effective environment to integrate software components of different nature, such as “R” scripts and compiled “C” code, or to facilitate the remote, distributed execution of models. This is done through the Java Native Interface and by using specific “Actors” (see below).

Kepler is based on a graphical user interface, where users select and connect the elements that will conform their scientific workflows, from computation, analysis and data sources.

Workflow components in Kepler are called Actors. Actors may contain a hierarchy of Actors, and in this case are called Composites. The Ports are the elements in the Actors that can receive Tokens. Tokens may include single or multiple data or messages. The execution of workflows is controlled by Directors in Kepler. Typically, a Director manages the execution of a set of actors. Actors can be tuned with Parameters. Kepler was extended to be able to access streaming sensor data and archived historical data [8]. In fig. 1 we can see a sample Kepler workflow that accesses sensor data.

Kepler actors are executed as local Java threads, but can also spawn distributed execution threads via web services or through the Java Native Interface (JNI). The actual execution model of the workflow depends on the nature of the director: for example, an SDF director will imply a synchronous execution of the workflow, where each computation node is processed one after the other; a PN Director will imply an execution of the workflow actors in parallel.

Kepler is a Java-based application that is maintained for the Windows, OSX, and Linux operating systems.

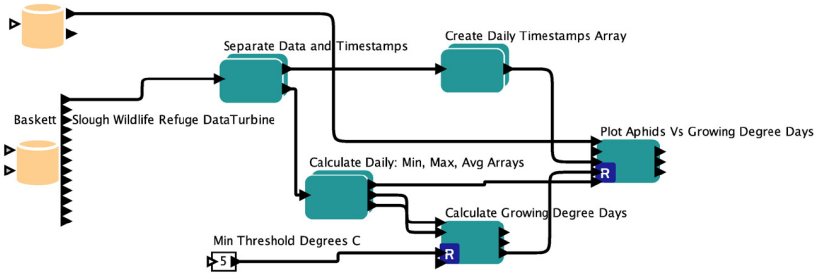

Figure 1. Sample Kepler analysis workflow which includes sensor data (taken from [8])

2.1. Use of Kepler in the Fusion Community

The fusion community in Europe is organized around EUROfusion, the European consortium for the development of Fusion Energy [20]. In the framework of the EUROfusion project, the European Integrated Modelling (EU-IM) team has as objective the development of a tokamak simulator that considers both the physics and all the machine related data, applicable to any fusion device. The simulation platform has been designed to be modular, flexible, and independent of a programming language. In 2011, the community evaluated different existing workflow engines and selected Kepler for the development of their workflows [27].

With this objective, they built a modelling infrastructure with a generic data structure that integrates both simulated and experimental data. The elements of this data structure are identified as "Consistent Physical Objects" (CPO). Thanks to this standardization of elements as CPOs, modules that solve the physics can be coupled into different integrated simulations (workflows). Also, modules describing the same physics can be interchanged within the same workflow. Physics modules are mapped as actors of a Kepler workflow and the data transfer among actors are performed through CPOs. Thanks to the semantic types that can be defined in Kepler, different CPOs can be distinguished and it can be verified if the different actors are correctly connected between them. Another feature interesting to this community is the functionality that Kepler allows for interactive steering of simulations, enabling to pause the simulation and reconfigure it, as well as the possibility of visualizing the present state of a simulation with specific actors.

The applications of the EU-IM require to execute from simple orchestration of workflows without convergence loops to tightly coupled workflows, involving mutual interactions among different codes.

An example of tightly coupled workflow has been built by the EU-IM [22] (formerly, EFDA ITM-TF). The European Transport Simulator (ETS) workflow [13], which couples different codes and will enable an entire discharge simulation from the start up until the current termination phase, including controllers and sub-systems. This workflow includes parallel components, like the GEMHPC one, which is run in 1024 cores. GEMHPC is based in GEM, which is written in MPI [48].

Within the project EUFORIA, the joint usage of different computing infrastructures (both HPC and HTC) in the Fusion community was considered. The solution derived by this project leverages and integrates different existing middleware: Kepler, as a workflow engine, which accesses the infrastructure using the Roaming Access Server (RAS). RAS provides access to the different underlying infrastructures using two alternative middlewares: gLite [31] and UNICORE [19]. Also, interactive access to the resources is supported with i2glogin [11].

The use of the different type of resources in this project took into account that, generally, simulations generating large amount of data will require large computing power only found in HPC systems, while the data analysis phase can be performed as independent tasks in HTC servers.

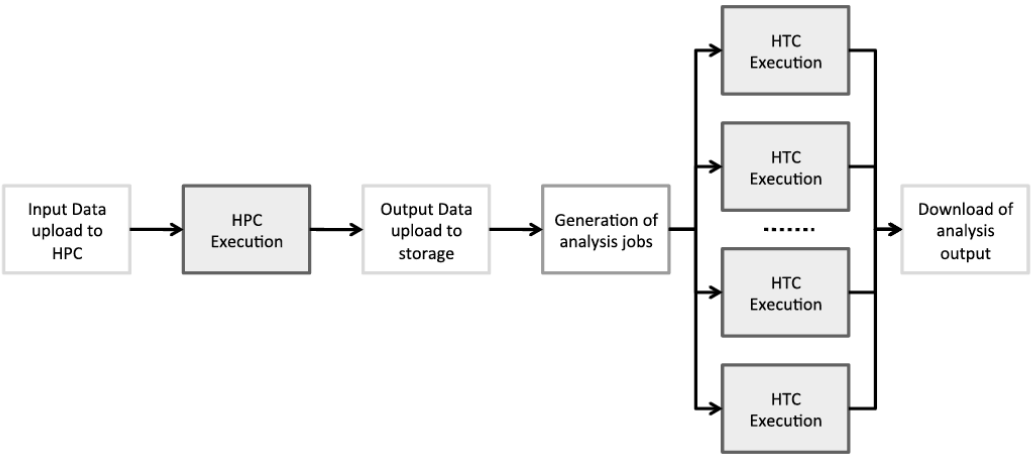

Figure 2. Sample Hybrid workflow from fusion community

The scenario considered in this project consisted of large simulations performed in HPC systems and the data produced was transferred to a storage system (see fig. 2). This transfer can be concurrent to the simulation, thus allowing to begin some of the following steps of the workflow.

3. Pegasus and the LIGO Collaboration

Pegasus [15] is a system based on the idea that users should define abstract workflows that include computations and information about the data without a direct mapping into the actual compute and storage resources. In that system, workflows are described as Direct Acyclic Graphs (DAGs), where nodes represent computational tasks and the edges represent data and control dependencies between the tasks. The data is exchanged between tasks in the form of files. From the abstract workflow, the actual physical location of data and executables is decided by the *Mapper*, which converts the abstract workflow into an executable one, with all the information about the location of the data files, resources where to execute the computations, etc. To locate the resources and files, catalogs are used that contain all the information.

Pegasus Mapper modifies the initial DAG before execution, therefore, statically. Nodes can be removed if there is data that is already available. Another optimization that Pegasus performs is task clustering, that merges a set of short duration nodes into a single one to reduce overhead. Some of these clustering strategies are guided by the users. The Mapper also associates jobs with workflow engines (again statically).

The workflow can be submitted to a local computing environment, a remote physical cluster or grid, or a virtual computing environment like the cloud.

In Pegasus, the workflow management system takes care of all the activities related to the execution of the workflow, from job and data management, monitoring and failure handling.

Pegasus provides textual interfaces in different programming languages, such as Python, Java and Perl, but what is described in these languages is the explicit abstract workflow, with

information about the nodes and their interconnections. This textual input, its translated into an XML description of the abstract workflow (DAX) which is then executed by the Pegasus engine. The Pegasus team advocates the use of textual workflows versus graphical workflows, since they consider that complex patters are easier to describe in this way.

Pegasus workflows can be defined in a hierarchical way, with nodes representing another workflow. This also helps to improve scalability of the system, since Pegasus needs to parse the whole XML file describing the DAX and for large cases this will not fit in memory. However, each DAX is managed by a different instance of the Pegasus engine. The hierarchy is used also in cases where the location of input files is unknown.

Pegasus has a set of execution engines with different features: single-core, which runs a single task at a time; non-shared file system, which stages in and out the files required by each computation; and Pegasus MPI Cluster (PMC) which is able to execute a DAG in Petascale systems by means of running them in an architecture based in a single master and several workers. PMC handles multi-core tasks but only within a node.

3.1. The LIGO Scientific Collaboration

Pegasus is a workflow environment that has been used for many different applications from various fields, from genomics, climate modeling, generation of sky mosaics, neuroscience, etc. The large collaborations where Pegasus has participated and has been key in their workflow developments are the LIGO Scientific Collaboration [1], the Southern California Earthquake Center (SCEC) [46] , and the National Virtual Observatory [38].

The Laser Interferometer Gravitational Wave Observatory (LIGO) is a network of gravitational-wave detectors, with observatories in Livingston, LA and Hanford, WA. The purpose of this collaboration is to prove the existence of the gravitational waves predicted by Einstein's General Theory of Relativity. To try to detect these waves, the scientists use kilometer-scale interferometric detectors.

The data generated by the instruments is distributed on the partners' sites, and then workflows are executed on the resources of their sites. Pegasus discovers the required data for each workflow and feeds the sites with them. One example of such an application is a workflow that searches for compact binary inspiral signals. The LIGO workflows are complex in the number of tasks (over 1.5 million jobs) and their dependencies, and the size of the datasets being analyzed (approximately 10 TB).

A characteristic of these workflows is that sometimes the granularity of the tasks is too small: in these cases, the workflows benefit of the feature of Pegasus that can cluster multiple tasks into one. Also, sometimes, some parts of the input data is recallibrated, requiring to recompute the workflow. However, recomputing the whole workflow is very expensive and what Pegasus offers is the possibility of registering data already being produced with the objective of not reproducing the part of the workflow that already generated it.

While LIGO workflows were initially (2002) deployed on the LIGO Data Grid, more recently have been extended to compute in the Open Science Grid and XSEDE. In September 2015 the LIGO collaboration detected gravitational waves. This detection was verified by processing roughly five terabytes of data by the LIGO workflows, generating many petabytes of exported data and executed in a distributed computing infrastructure composed of multiple HPC sites. The PyCBC search pipeline used for this validation is composed of hundreds of thousands tasks. Although some of the tasks are threaded (like calls to FFTW library), most of them are

sequential and short tasks. To reduce the overhead of small tasks in a large HPC cluster, the Pegasus MPI Cluster execution engine was used to submit sub-workflows as monolithic jobs.

4. Life-sciences Community - Galaxy

Galaxy [3] is a web-based platform initially designed for life sciences workflows. It offers a public service and a collaborative environment which enables to share, through internet, analysis tools, genomic data, tutorial demonstrations, persistent workspaces, and publication services, all available through internet in public repositories.

Through a web browser interface, Galaxy users can edit their workflows in a graphical editor where workflows are created by connecting tools. A Galaxy workflow is a reusable template analysis that a user can run repeatedly on different data; each time a workflow is run, the same tools with the same parameters are executed. Interoperability with different programming languages is done by invoking binaries: Galaxy supports any tool or piece of software for which a command line invocation can be constructed. Besides the graphical interface, the users can use BioBlend [51] a Python programmatic API to define their workflows in a textual form and supporting more complex formats difficult to deal in a graphical way.

Galaxy has a significant community of users and developers. Galaxy pages are the principal means to communicate research performed using Galaxy. Pages are interactive, web-based documents that users can create to describe a complete genomics experiment. This allows users to document and publish their experiments with computational outputs, allowing others to view the experiment with all the details and enable total reproducibility.

Galaxy enables the users to import datasets from many data warehouses. It relies on the concept of Object Store, a file interface that acts as a layer between Galaxy and user datasets. The Object Store supports distributed datasets and the application can exploit data locality and submit jobs to the resource closer to the data. Also, it automatically generates and maintains metadata about the different aspects of each analysis: input datasets, tools used, parameter values, and output datasets.

Users can import existing *histories*[4] and workflows, and rerun them. Also, they can modify or extend the analysis. Galaxy's public web server processes about 5,000 jobs per day and there is a large number of groups not affiliated with the Galaxy team that have been using the system to perform different types of genomic research and have published their results in prominent journals as Science or Nature. Besides the public server, a local instance can also be deployed in the user premises. Additional to the Galaxy server, Galaxy workflows can be executed in the cloud through the CloudMan platform [4].

One of the drawbacks reported by Galaxy users, is the challenge of installing a Galaxy instance [25]. This has been recently fixed by making available a Docker image.

Galaxy team is also collaborating in the definition of the Common Workflow Language support.

4.1. Galaxy Workflows in Life Science

Galaxy is very popular for Next-Gen Sequencing data analysis since it has available a large collection of tools for genomics and sequence analysis. Galaxy repositories [54] list on the order of thousand tools, most of them specific to genomics and sequence analysis that are used to

[4] A history is a series of analysis steps

compose the workflows. Most of these tools are sequential and parallelism is only exploited at very low levels (for example, up to 16 cores in Stampede [5]).

With regard the data used in these workflows, it can involve many datasets of variable size. For example, input data sets with 1 - 10 large files of 1 - 10 GB each, during the analysis other datasets are referenced (genome datasets of 1-40 GB) and although several intermediate datasets are produced, the final results require a relatively small amount of storage size (<100MB).

ELIXIR [17], the distributed infrastructure for life-science information partly funded by the European Commission within the Research Infrastructures programme of Horizon 2020, is an example of usage of Galaxy in this community. Due to the large interest of their scientific community in Galaxy, they established a Galaxy Working Group to evaluate the technical strategy for Galaxy within the context of ELIXIR. Between the activities performed by this group (meetings, surveys and discussions), they generated a report with recommendations on the use of Galaxy [12].

According to this report, Galaxy is used in that community for data intensive analysis from different domains: Genomics, Transcriptomics, Proteomics, Systems Biology, Metabolomics, but also metagenomics, imaging, small RNAs, etc. The popularity of Galaxy in that area is due to the possibility that offers to users with limited or no knowledge of command line to perform data-intensive analyses.

The users of Galaxy in this context can execute their workflows in the global Galaxy server or in local instances of Galaxy installed in the users institutions' or partner institutions. Most of the institutions reported in the survey the use of a compute cluster to host the Galaxy server (52.63%) but the amount of cores available for Galaxy jobs is surprisingly small (for most cases, from 10 - 49 cores, and only 9% of cases more than 100 cores).

BioExcel [10], the Center of Excellence for Computational Biomolecular Research funded by the European Commission, also has Galaxy as one of the workflow managers considered to be used in their activities. However, in this case, other systems have been considered due to the expertise of the partners: Taverna and PyCOMPSs/COMPSs, or the combination of two of them, for example, using Galaxy to compose coarse grain workflows and PyCOMPSs/COMPSs for a finer grain workflows that better exploit the parallelism of the system. Other workflow management systems considered by this community are KNIME [9] and Copernicus [42].

5. Cylc, Autosubmit and ecFlow and the Earth Science Community

The Earth Science community (climate) is another community case considered in this paper. Three different workflow management systems were compared by this community in the European project IS-ENES2 [28] which involved the stakeholders in Europe in that topic. The community considered: Cylc [39], Autosubmit [34] and ecFlow [33]. Although the community does not have a clear winner, the Met Office is using Cylc and received funding to continue development of Cylc.

Cylc is a Python based workflow engine and meta-scheduler. According to the developers, it specialises in continuous workflows of cycling tasks such as those used in weather and climate forecasting and research (i.e. workflows that show iterative patterns). Cylc is also easy to use with non-cycling systems. Cylc was created at the National Institute of Water and Atmospheric Research (NIWA, New Zealand) and is free software under the GNU GPL v3 license.

Cylc was developed to offer a workflow management system for the weather and climate community which based its studies on the use of complex scripts. Cylc is widely used by the community from research to real-time operations including ensemble prediction systems.

To provide robustness when executing the workflows, Cylc dumps state files and writes information into SQLite databases about the state of the execution. Cylc tasks can be configured to retry a number of times on failures.

Autosubmit is a solution created at IC3's Climate Forecasting Unit (CFU) to manage and run the research group's experiments. The development of this tool was a result of the lack of in house HPC facilities that led to a software design with very minimal requirements on the HPC that will run the jobs. Autosubmit, written in Python, provides a simple workflow definition and is capable to run experiments on remote clusters or supercomputers and on any GNU/Linux or Unix host. Autosubmit is currently being developed at the BSC Computational Earth Sciences group [35].

It has some fault tolerance features, based on check-pointing the tasks that have been finished: it keeps a list of completed tasks, and if the scheduler does not respond properly, when restarting the experiment the process will continue from the same point. A number of given retrials can also be defined for the different jobs that compose an experiment.

ecFlow is a workflow package that enables users to run a large number of programs (with dependencies on each other and on time) in a controlled environment. It is used at ECMWF to manage around half of their operational suites across a range of platforms. ecFlow checkpoint file allows it to restart at the last checkpoint before a failure. Also, a number of retries are supported on job failure.

The three systems have a similar input interface, based on scripts with tags, which seem to fulfill the needs of the community to describe their workflows. Cylc and ecFlow have also graphical interfaces to monitor the evolution of the execution of the experiments. While Cylc and ecFlow have a Graphical User Interface, Autosubmit only has some visual features through its monitor command.

5.1. Multi-member Climate Experiments with Autosubmit

The experiments that this community run are (multi-model) multi-member ensemble experiments. These experiments are traditionally organized in multiple simulations executed for given start dates (the purpose is to simulate weather or climate conditions on that period of time). The complexity of each experiment can be defined by different axes: number of start dates, number of members within a start date and number of chunks within a member.

For example, in [34] the authors present two experiments performed with Autosubmit. The experiments involved three type of resources: the local machine where the whole experiment is submitted, the MareNostrum3 supercomputer where the parallel simulations were run, and a post-processing fat node. In this case, each experiment consisted of 10 members of 4-month length for 34 start dates between 1993 and 2009 (only one chunk per member in this case). The total experiment consisted of 340 independent cases of 4 months, which is equivalent in cost to running a single simulation of approximately 113 years. This information is registered in an input configuration file which is provided as input to Autosubmit – this is how the user specify the workflow in this system.

Each simulation itself consists of several tasks: input data transfer, compilation, initialisation, chunk simulation, chunk post-processing, cleaning, and results data transfer. The user

needs to supply for each of these tasks the corresponding run scripts and the definition of how these tasks are to be executed (execution script, reference to computing resource to execute the task, dependencies with other tasks, number of processors required, etc).

At execution time, Autosubmit takes all the information from the configuration file and builds a task graph that takes into account the dependencies. Autosubmit is able to execute in parallel several tasks if dependencies between them allow it, although these type of experiments tend to be highly sequential since results from previous simulations are used as input for subsequent ones.

For each of these two experiments, Autosubmit ran 2381 jobs: 341 jobs were run in the local machine, 1360 in MareNostrum3 and 680 in the post-processing fat node. Some of the jobs in MareNostrum are MPI simulations using between 300-400 processors each.

6. Astrophysics - Taverna and COMPSs

Taverna [56] is another alternative Workflow Management System. Developed and maintained by the University of Manchester, it is currently used by several scientific communities.

Written in Java, it is composed of the Taverna Engine (used for enacting workflows), the Taverna Workbench (a graphical desktop client application, although a command line interface is also offered) and the Taverna Server (which supports the execution of remote workflows). Taverna supports local and remote services, and has been used in several domains: from biology, chemistry and medicine to music, meteorology and social sciences. The system is open source and it is offered for windows, linux and Mac OS.

Taverna [26] was initially designed as an application to ease the use of molecular biology tools and databases available on the web, especially web services. Taverna was designed with the philosophy that scientists could develop their workflows of webservices already published and then save the workflow in a repository, in such a way that the workflow can be reused and shared. The workflows are published in a public repository in http://www.myexperiment.org [23]. The myExperiment workflows repository does not only contain Taverna workflows, but also Galaxy or Kepler workflows.

New workflows are built with the Taverna Workbench in a graphical way, dragging and dropping new services in the workflow diagram and connecting their inputs and outputs. Taverna workflows are traditionally a mixture of web services, scripts (in R, for example) and other type of services.

In 2013 the Taverna engine was improved in order to be able to support scalable processing of large data sets, and to be capable of performing implicit iteration, looping and streaming of data. It was also at that time that the Taverna server was introduced, in order to support distributed execution.

The workflows can be executed on local machines or in distributed computing infrastructures (supercomputers, Grids or cloud environments), through the Taverna Server. An installation of the Server provides access to a collection of workflows (normally through a web interface, called the Taverna Player). However, in this execution mode users cannot edit the published workflows in the Server, neither add new workflows to the set of workflows deployed in the Server.

Another feature of Taverna is the possibility of tracking provenance: the Taverna engine records service invocations, intermediate and final workflow results. Also, Taverna supports nested workflows.

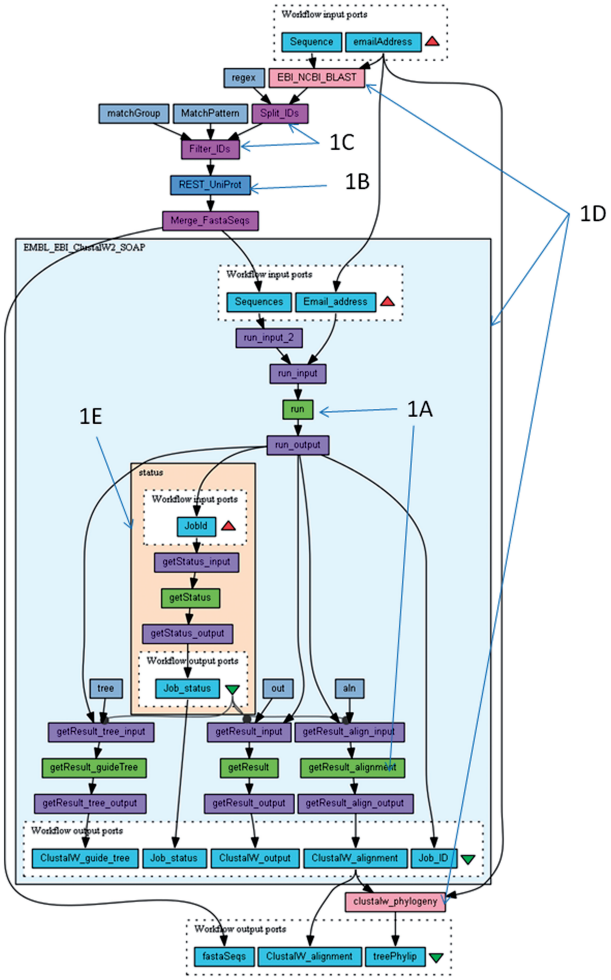

Figure 3. Sample Taverna workflow that implements Blast_Align_and_Tree (taken from [56])

6.1. Implementing Two-level Workflows for Astronomy with Taverna and COMPSs

The astrophysics community is facing a huge challenge both in terms of computing and data with the Square Kilometer Array (SKA [50]), where they expect to reach data rates in the exascale domain. They expect up to 10 exabytes of data per day, and are planning to build an exascale computing platform that can deal with this amount of data and that it is able to process it, sometimes in near real-time.

Taverna has traditionally been used in this area, however, since recently the exploitation of distributed computing infrastructures with Taverna was quite limited, and in general the exploitation of the parallelism is not the strong point of the environment. An alternative implementation of workflows, was considered in [45], with the combination of workflows at two levels: first level driven by Taverna and a second level driven by COMPSs.

COMPSs [32, 53] is a framework which aims to ease the development and execution of parallel applications for distributed infrastructures, such as Clusters and Clouds. A COMPSs application is composed of tasks, which are annotated methods. At execution time, the runtime builds a task graph that takes into account the data dependencies between tasks, and from this graph schedules and executes the tasks in the distributed infrastructure, taking also care of the required data transfers between nodes. COMPSs is written in Java, and supports applications in Java, Python and C/C++. Between the features of COMPSs, we find that the workflow can

be composed of tasks that are regular methods or web services, and that the whole COMPSs application can be published as a web service.

Another feature of COMPSs is that their applications are agnostic of the actual computing infrastructure where they are executed. This is accomplished through a component that offers different connectors, each bridging to each provider API. COMPSs can run in different Cloud providers and federation of them, and in clusters and supercomputers. COMPSs runtime also supports elasticity in clouds and federated clouds.

COMPSs applications can be exposed as web services, and internally these web services are task-based applications able to run in parallel in distributed computing platforms. These web services can then be combined with the Taverna Workbench into graphical workflows. This approach results in a two-level workflow system: at the user level, workflows are built upon web services, while those services turn out to be workflows as well at the infrastructure level. The architecture of this solution is shown in fig. 4.

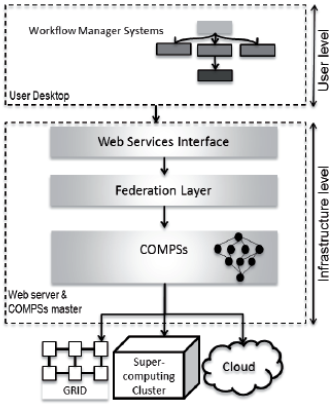

Figure 4. Two-level workflow system architecture for a astrophysics use case (taken from [45])

The cases considered compose a set of of analysis tasks of interest for some user applications of the SKA community. The focus is on the kinematical modelling of galaxies, which is applied in the study of galaxies evolution.

A common practice is to run the set of tasks with different parameters, in order to generate several models. Therefore, several workflows are executed, and later there is a manual phase from the astronomer to choose the optimal generated model. Given the large amount of data that it is foreseen to be generated by SKA, the workflows have been designed to execute the processing tasks where the data is stored.

The web services were deployed in a supercomputing cluster and in a distributed computing infrastructure (IBERGRID). The COMPSs services were configured to receive either individual sets of parameters to run a single combination of data or a list of sets of parameters in order to run multiple time the same workflow. This is easy to be implemented in COMPSs, since offers a programmatic interface, and it is also executed very fast in its runtime, while in other systems like the same Taverna Workbech, either was difficult to specify since it is not that simple to specify a loop in the graphical interface or it was not as efficient as expected.

Taverna Workbench Astronomy 2.5 was used to edit the graphical workflows, a special edition of the Taverna Workbench that includes support for building and executing astronomy workflows based on VO services through the Astrotaverna plug-in [44]. The workflows have been published in the myExperiment repository and can be accessed by the community.

7. Intelligent Workflows

Previous sections have described current approaches to manage scientific workflows and successful use cases deployed in distributed computing. With the Exascale era around the corner, the community faces a unique opportunity of implementing a new generation of intelligent workflows, which involve large simulations together with data analytics.

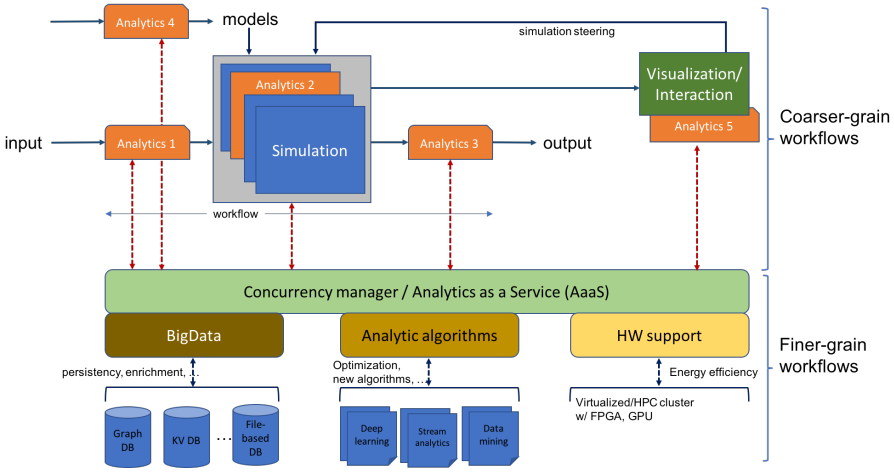

Figure 5. Architecture of new intelligent workflows

Such workflows (see fig. 5) will be composed of HPC simulations (a single task or node in the workflow may be a large MPI+X simulation involving several computing nodes), data analytics (which can be both at the input, interleaved with computation, or at the output) and visualization. The actual workflow should not be static, but dynamically instantiated according to the needs of the overall application objective. This will prevent *brute force* execution of large simulations, otherwise enabling the dynamic deployment of new simulations or computations in order to, for example, enact a finer simulation cycle since previous analytics cycle detect a given anomaly in the previous results.

At a higher level, the system should provide an end-to-end coordination layer that enables the management of dynamic workflows composed of simulations, analytics and visualization, including inputs/outputs from streams. Since graphical interfaces usually lack of enough tools to express dynamicity, a programmatic interface would be probably more appropriate. Programmatic interfaces does not only support the description of iterative constructions like conditional loops, but offer the whole expressiveness of the programming language to express complex algorithms, like optimization searches, etc. For example, PyCOMPSs [53] or Swift [55] offer programmatic/scripting interfaces.

Additionally, taking into account that some application areas may require the possibility of accepting streamed input data (from sensors or other sources of dynamic data) and streamed output data (visualization, monitoring, etc) the system should support this type of data acquisition.

This first coarser grain level of workflows will include a set of analytics, implemented as fine grain workflows. These analytics can be provided as a layer of Analytics as a Service that can be used by the workflows depending on their requirements. The analytics may implement algorithms that can be parallelized as well, but usually this type of algorithms does not show a parallelism easy to deal with traditional parallel programming models such as MPI, that is why task-based programming models seem to be a better approach to implement such services.

The services will be executed in a set of nodes of the same computing infrastructure, showing an inherent parallelism described in the form of a finer grain workflow or task-graph.

Alternatives to implement these services can be traditional Big Data programming models, such as Spark [58] or Hadoop [24], although these systems sometimes lack of the expected performance in HPC systems [16], and environments that include runtimes with more performance may be required for this purpose. Of special interest to implement these analytics can be the use of GPUs or accelerators, that have proven to be key in the implementation of fast Neural Networks used in Deep Learning [47].

The amount of data received, processed and generated with these workflows will require new solutions for storage that go beyond the traditional file systems. New storage devices such as Non-volatile RAM and storage class memories support data persistency and byte addressable access, with a performance between memories and SSDs. These devices enable the availability of data while being generated, without the need of writing the data to disk. A new type of consumer-producer applications can be designed, where the data can be stored in these persistent storage and be accessed during the execution of the producer application or after. While this data can be stored in files or databases, both are designed to use block devices, while this type of storage supports other alternatives, as direct object storage [36].

In the environment described before, persistent storage can be used to store the results of simulations. The data can be consumed by the analytic services as soon as it has been produced and the results of the analytic steps can also be stored in persistent storage, in order to be used in visualization steps or in future queries. New challenges that appear are decisions on which data should be stored in each level of the storage hierarchy, since probably the persistent layer would have less capacity, or how to perform garbage collection in such memories (since data is persistent after the execution of the applications), and its integration with the programming models, since a clean interface should be provided to the programmers.

7.1. Summary and Systems Comparison

As a summary of the paper, this subsection discusses the main features of the Workflow Management Systems (WFS) described in this paper in comparison with the new WFS architecture proposed in this section. This comparison is shown in Tab. 1.

Table 1. Workflow Management Systems features comparison

Feature / WMS	Galaxy	Kepler	Autosubmit	Taverna	Pegasus	(COMPSs	Int. Workflows
Interface	Graphical	Graphical	Script	Graphical	Textual	Programmatic	Programmatic
Parallel tasks	Limited	Yes	Yes	Limited	Yes	Yes	Yes
Dynamic workflow	No	No	No	No	Somehow	Yes	Yes
Hierarchy	No	Yes	No	Yes	Somehow	Somehow	Yes
Support for streams	No	Yes	No	Yes	No	No	Yes
Support for visualization	Yes	Yes	No	No	No	No	Yes
Support for new stor. tech.	No	No	No	No	No	Yes	Yes
Support for accelerators	No	No	No	No	Somehow	Yes	Yes

As a general comment, we believe that WMS should be generic enough to cover the requirements of different scientific communities. It is reasonable that exists solutions home-made or ad-hoc which are later adopted by more users and extended, but we consider that this should not be the best practice.

A graphical interface is sometimes preferred by non expert programmers. However, drawing large workflows that include conditional and loops can be a difficult task. Programmatical interfaces offer the flexibility and expressiveness of the programming model: the behaviour of a

complex workflow can be described with a few lines of code. These interfaces can be supported with graphical tools that visually show the obtained workflow. Also, this interface is able to naturally support dynamic workflows.

The support for parallel tasks is, in most of the current systems, limited to multi-threaded intranode tasks. Also, in some systems, although MPI tasks are supported, this support is through its interaction with the batch system sending the whole task to the queuing system. However, this is a key feature when considering workflows of HPC applications, with tasks that are MPI applications executed across multiple nodes of a cluster.

The support for hierarchy is in a similar situation in the existing systems, sometimes supported through invocation of new instances of the engine, like in Pegasus or in the current version of PyCOMPSs. This feature is very relevant in order to compose sub-workflows semantically different into larger workflows. For example, a sub-workflow may compose a set of HPC simulations, while another sub-workflow implement analytics of the results of these simulations.

Support for streaming and visualization are related features with limited support in current systems (only Kepler supports both of them), but that are key for the support of end-to-end workflows which involve inputs and outputs from multiple sources.

The support for new storage technologies and new architectures like accelerators have not been considered so far by most of the current systems, but as technology evolve the WMS should also consider them to improve performance and functionality.

Conclusions

While the scientific community has a unified view of what is a workflow, the different instances of Workflow Management Systems available for researchers have large variety: options for the interface, views on what can be a workflow tasks, types of data being exchanged by the tasks, engine complexity, computing platform, etc.

This different nature is sometimes explained by the best practices of specific communities and by the type of workflows each community requires. For example, for some communities, having an intuitive graphical interface with the possibility of editing their workflows with a simple drag and drop is essential, while for others, simplifying the access to large supercomputers where they can run large parallel applications is a must.

However, within a given scientific community WMS with similar characteristics or interoperable between them are used. This is the case of Galaxy and Taverna, for example, largely used in bioinformatics research, for which even exists a system, Tavaxy [2], that supports both systems' workflows.

While offering a single workflow management system for all scientific communities does not seem possible, we believe that interoperability between similar systems should be promoted, through common workflow description languages or interoperable interfaces. What is more, a new family of workflow management systems that enable better integration between the computation and analytics of the workflows should be designed. These systems should enable a smarter definition of the workflows, which will be more efficient in the usage of computing and storage resources, and more effective on performing the required computations and analysis that are required by the scientists. With regard the computing infrastructures, new architectures that include new computing devices (GPUs, FPGAs and other accelerators), and new storage hierarchies and technologies should be considered.

Acknowledgments

This work has been supported by the Spanish Government (SEV2015-0493), by the Spanish Ministry of Science and Innovation (contract TIN2015-65316-P), by Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272). This work is also supported by the Intel-BSC Exascale Lab. The Human Brain Project receives funding from the EU's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 604102.

References

1. Abbott, B., Abbott, R., Adhikari, R., Ajith, P., Allen, B., Allen, G., Amin, R., Anderson, S., Anderson, W., Arain, M., et al.: Ligo: the laser interferometer gravitational-wave observatory. Reports on Progress in Physics 72(7), 076901 (2009), DOI: 10.1088/0034-4885/72/7/076901

2. Abouelhoda, M., Issa, S.A., Ghanem, M.: Tavaxy: Integrating taverna and galaxy workflows with cloud computing support. BMC bioinformatics 13(1), 77 (2012)

3. Afgan, E., Baker, D., van den Beek, M., Blankenberg, D., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Eberhard, C., et al.: The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic acids research p. gkw343 (2016)

4. Afgan, E., Chapman, B., Taylor, J.: Cloudman as a platform for tool, data, and analysis distribution. BMC Bioinformatics 13(1), 315 (2012), DOI: 10.1186/1471-2105-13-315

5. Afgan, E., Coraor, N., Chilton, J., Baker, D., Taylor, J., Team, T.G.: Enabling cloud bursting for life sciences within galaxy. Concurrency and Computation: Practice and Experience 27(16), 4330–4343 (2015), cPE-15-0018.R1

6. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: an extensible system for design and execution of scientific workflows. In: Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on. pp. 423–424. IEEE (2004)

7. Amstutz, P., Crusoe, M.R., Tijanić, N., Chapman, B., Chilton, J., Heuer, M., Kartashov, A., Leehr, D., Mnager, H., Nedeljkovich, M., Scales, M., Soiland-Reyes, S., Stojanovic, L.: Common Workflow Language, v1.0. Tech. rep. (3 2016), `https://figshare.com/articles/Common_Workflow_Language_draft_3/3115156`, DOI: 10.6084/m9.figshare.3115156.v2

8. Barseghian, D., Altintas, I., Jones, M.B., Crawl, D., Potter, N., Gallagher, J., Cornillon, P., Schildhauer, M., Borer, E.T., Seabloom, E.W., Hosseini, P.R.: Workflows and extensions to the kepler scientific workflow system to support environmental sensor data access and analysis. Ecological Informatics 5(1), 42 – 50 (2010), `http://www.sciencedirect.com/science/article/pii/S1574954109000673`, special Issue: Advances in environmental information management

9. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007). Springer (2007)

10. BioExcel website. Web page at http://www.bioexcel.eu, accessed: 2017-02-15

11. Cabellos, L., Campos, I., del Castillo, E.F., Owsiak, M., Palak, B., Pciennik, M.: Scientific workflow orchestration interoperating htc and hpc resources. Computer Physics Communications 182(4), 890 – 897 (2011), http://www.sciencedirect.com/science/article/pii/S0010465510005096

12. Coppens, F., Corpas, M.: Recommendation for actions on Galaxy for ELIXIR HoNs. available at https://www.elixir-europe.org/about/groups/galaxy-wg, accessed: 2017-02-15

13. Coster, D.P., Basiuk, V., Pereverzev, G., Kalupin, D., Zagorksi, R., Stankiewicz, R., Huynh, P., Imbeaux, F., et al.: The European Transport Solver. IEEE Transactions on Plasma Science 38(9), 2085–2092 (2010)

14. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25(5), 528 – 540 (2009), http://www.sciencedirect.com/science/article/pii/S0167739X08000861

15. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., da Silva, R.F., Livny, M., et al.: Pegasus, a workflow management system for science automation. Future Generation Computer Systems 46, 17–35 (2015)

16. Ekanayake, S., Kamburugamuve, S., Wickramasinghe, P., Fox, G.C.: Java thread and process performance for parallel machine learning on multicore hpc clusters. In: Proceedings of the 2016 IEEE International Conference on Big Data (2016)

17. Elixir website. Web page at https://www.elixir-europe.org, accessed: 2017-02-15

18. Building an European Reseach Community through Interoperable Workflows and Data. Web page at http://www.erflow.eu, accessed: 2017-02-15

19. Erwin, D.W., Snelling, D.F.: Unicore: A grid computing environment. In: European Conference on Parallel Processing. pp. 825–834. Springer (2001)

20. European Consortium for the Development of Fusion Energy. Web page at https://www.euro-fusion.org, accessed: 2017-02-15

21. Fahringer, T., Prodan, R., Duan, R., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.L., Villazon, A., Wieczorek, M.: Askalon: A grid application development and computing environment. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing. pp. 122–131. IEEE Computer Society (2005)

22. Falchetto, G.L., Coster, D., Coelho, R., Scott, B., Figini, L., Kalupin, D., Nardon, E., Nowak, S., Alves, L.L., Artaud, J.F., et al.: The european integrated tokamak modelling (itm) effort: achievements and first physics results. Nuclear Fusion 54(4), 043018 (2014)

23. Goble, C.A., Bhagat, J., Aleksejevs, S., Cruickshank, D., Michaelides, D., Newman, D., Borkum, M., Bechhofer, S., Roos, M., Li, P., et al.: myexperiment: a repository and social network for the sharing of bioinformatics workflows. Nucleic acids research 38(suppl 2), W677–W682 (2010)

24. Apache Hadoop. Web page at http://hadoop.apache.org/ ((Date of last access: 15th November, 2016))

25. Hospital, A., Montras, A., Soiland-Reyes, S., Bonvin, A., Melquiond, A., Gelpí, J.L., Lezzi, D., Newhouse, S., Dianes, J.A., Abraham, M., Apostolov, R., Ippoliti, E., Carter, A., White, D.J.: D2.1 State of the art and gap analysis. Tech. rep., BioExcel deliverable (2016)

26. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. Nucleic acids research 34(suppl 2), W729–W732 (2006)

27. Imbeaux, F., Pinches, S., Lister, J., Buravand, Y., Casper, T., Duval, B., Guillerminet, B., Hosokawa, M., Houlberg, W., Huynh, P., Kim, S., Manduchi, G., Owsiak, M., Palak, B., Plociennik, M., Rouault, G., Sauter, O., Strand, P.: Design and first applications of the iter integrated modelling & analysis suite. Nuclear Fusion 55(12), 123006 (2015), http://stacks.iop.org/0029-5515/55/i=12/a=123006

28. InfraStructure for the European Network for the Earth System Modelling. Web page at https://is.enes.org, accessed: 2017-02-15

29. The Kepler Project. Web page at https://kepler-project.org, accessed: 2017-02-15

30. Kranzlmüller, D., de Lucas, J.M., Öster, P.: The european grid initiative (egi). In: Remote Instrumentation and Virtual Laboratories, pp. 61–66. Springer (2010)

31. Laure, E., Edlund, A., Pacini, F., Buncic, P., Barroso, M., Di Meglio, A., Prelz, F., Frohner, A., Mulmo, O., Krenek, A., et al.: Programming the grid with glite. Tech. rep. (2006)

32. Lordan, F., Tejedor, E., Ejarque, J., Rafanell, R., Alvarez, J., Marozzo, F., Lezzi, D., Sirvent, R., Talia, D., Badia, R.M.: ServiceSs: An Interoperable Programming Framework for the Cloud. Journal of Grid Computing 12(1), 67–91 (2014)

33. Manubens-Gil, D., Vegas-Regidor, J., Matthews, D., Shin, M.: Assesment report on autosubmit, cylc and ecflow. Tech. rep. (2016), https://earth.bsc.es/wiki/lib/exe/fetch.php?media=tools:isenes2_d93_v1.0_mp.pdf

34. Manubens-Gil, D., Vegas-Regidor, J., Prodhomme, C., Mula-Valls, O., Doblas-Reyes, F.J.: Seamless management of ensemble climate prediction experiments on hpc platforms. In: High Performance Computing & Simulation (HPCS), 2016 International Conference on. pp. 895–900. IEEE (2016)

35. Manubens-Gila, D., Vegas-Regidora, J., Acostaa, M.C., Prodhommea, C., Mula-Vallsa, O., Serradell-Marondaa, K., Doblas-Reyes, F.J.: Autosubmit: a versatile tool for managing Earth system models on HPC platforms. Future Generation Computer Systems submited (2016)

36. Marti, J., Gasull, D., Queralt, A., Cortes, T.: Towards DaaS 2.0: Enriching data models. In: Proceedings - 2013 IEEE 9th World Congress on Services, SERVICES 2013. pp. 349–355. IEEE, IEEE (jun 2013), DOI: 10.1109/SERVICES.2013.59

37. McLennan, M., Clark, S., Deelman, E., Rynge, M., Vahi, K., McKenna, F., Kearney, D., Song, C.: Hubzero and pegasus: integrating scientific workflows into science gateways. Concurrency and Computation: Practice and Experience (2014), DOI: 10.1002/cpe.3257

38. National Virtual Observatory. Web page at `http://us-vo.org`, accessed: 2017-02-15

39. Oliver, H.J.: Cylc (the cylc suite engine). Tech. rep. (2016)

40. Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Würthwein, F., et al.: The open science grid 78(1), 012057 (2007)

41. Price, B.: Frank and lillian gilbreth and the manufacture and marketing of motion study, 1908-1924. Business and economic history pp. 88–98 (1989)

42. Pronk, S., Larsson, P., Pouya, I., Bowman, G.R., Haque, I.S., Beauchamp, K., Hess, B., Pande, V.S., Kasson, P.M., Lindahl, E.: Copernicus: A new paradigm for parallel adaptive molecular dynamics. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 60:1–60:10. SC '11, ACM, New York, NY, USA (2011), DOI: 10.1145/2063384.2063465

43. Ptolemaeus, C. (ed.): System Design, Modeling, and Simulation using Ptolemy II. Ptolemy.org (2014), `http://ptolemy.org/books/Systems`

44. Ruiz, J., Garrido, J., Santander-Vela, J., Sánchez-Expósito, S., Verdes-Montenegro, L.: Astrotavernabuilding workflows with virtual observatory services. Astronomy and Computing 7, 3–11 (2014)

45. Sánchez-Expósito, S., Martín, P., Ruíz, J.E., Verdes-Montenegro, L., Garrido, J., Sirvent, R., Falcó, A.R., Badia, R., Lezzi, D.: Web services as building blocks for science gateways in astrophysics. Journal of Grid Computing 14(4), 673–685 (2016)

46. Southern California Earthquake Center. Web page at `http://scec.org/`, accessed: 2017-02-15

47. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks 61, 85 – 117 (2015), `http://sciencedirect.com/science/article/pii/S0893608014002135`

48. Scott, B.D., Weinberg, V., Hoenen, O., Karmakar, A., Fazendeiro, L.: Scalability of the plasma physics code gem. arXiv preprint arXiv:1312.1187 (2013)

49. SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs. Web page at `http://www.shiwa-workflow.eu/`, accessed: 2017-02-15

50. Square Kilometre Array. Web page at `https://www.skatelescope.org`, accessed: 2017-02-15

51. Sloggett, C., Goonasekera, N., Afgan, E.: Bioblend: automating pipeline analyses within galaxy and cloudman. Bioinformatics 29(13), 1685–1686 (2013)

52. The Principles of Scientific Management. The Mathematics Teacher 4(1), 44–44 (1911), http://www.jstor.org/stable/27949698

53. Tejedor, E., Becerra, Y., Alomar, G., Queralt, A., Badia, R.M., Torres, J., Cortes, T., Labarta, J.: Pycompss: Parallel computational workflows in python. International Journal of High Performance Computing Applications (2015)

54. Galaxy Tool Sheed. Web page at https://toolshed.g2.bx.psu.edu, accessed: 2017-02-15

55. Wilde, M., Hategan, M., Wozniak, J.M., Clifford, B., Katz, D.S., Foster, I.: Swift: A language for distributed parallel scripting. Parallel Computing 37(9), 633–652 (2011)

56. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., et al.: The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic acids research p. gkt328 (2013)

57. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. Journal of Grid Computing 3(3-4), 171–200 (2005)

58. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster Computing with Working Sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. HotCloud'10, USENIX Association, Berkeley, CA, USA (2010)