Scalable Computing: Practice and Experience
Volume 13, Number 4, pp. 325–338. http://www.scpe.org
ISSN 1895-1767

INTEGRATION OF CLOUD-BASED SERVICES INTO DISTRIBUTED WORKFLOW SYSTEMS: CHALLENGES AND SOLUTIONS

PAWEL CZARNUL*

Abstract. The paper introduces the challenges in modern workflow management in distributed environments spanning multiple cluster, grid and cloud systems. Recent developments in cloud computing infrastructures are presented and are referring how clouds can be incorporated into distributed workflow management, aside from local and grid systems considered so far. Several challenges concerning workflow definition, optimisation and execution are considered. These range from configuration, integration of business and scientific services, data management, dynamic monitoring and tracking, reusable workflow patterns, semantic search and distributed execution of distributed services. Finally, the author recommends a solution to these challenges based on the BeesyCluster middleware for distributed management of services with static and dynamic rescheduling within a market of services.

Key words: workflow management, cloud computing, services on the cloud, service integration

1. Introduction. Integration of services into distributed workflow applications has been covered widely in the literature for grid based systems [4, 5]. Several solutions have been proposed for:

- a conceptual model of the workflow including DAGs with extensions,
- QoS modelling, integration and management,
- actual implementations of workflow management systems for both business and academic applications.

Since cloud-based computing including SaaS, IaaS and PaaS has become more and more popular and widely used, it is expected that knowledge and solutions to service integration developed for grid-based workflow solutions would be adopted and extended for cloud-based environments.

From the point of view of an enterprise, integration into workflows is of key importance as it allows to model processes such as processing orders, banking, payroll, B2B cooperation, flow of production processes and many others. Cloud computing adds new potential to this but at the same time several challenges arise that will be formulated in the paper and for which solutions will be presented:

- need for uniform interfaces and middleware for: service management, data transfer and handling in the cloud environment especially in the context of automatic service discovery and matching across clouds,
- uniform QoS monitoring and assurance across various cloud providers and types of services,
- algorithms for composition of workflows in the cloud environment including dynamic learning of service information including QoS,
- dynamic changes of the QoS parameters and cloud availability,
- incorporation of several clouds to avoid the vendor lock-in problem for effective implementation of workflow management in sky computing,
- integration of both ready-to-use SaaS, IaaS, PaaS as well as human interactions and decision making also in workflows spanning several enterprises,
- integration of legacy applications, SOA, grid and cloud computing into workflows effectively merging the recent paradigms for distributed computing.

In this paper, the aforementioned challenges and solutions to these will be presented from the following perspectives: a conceptual model or models proposed for the given challenge, technological aspects, APIs, implementations if already exist. Otherwise, suggestions and hints on how to adapt the existing state-of-the-art to provide future elastic, efficient and cost-effective workflow systems in clouds will be provided.

2. Related work.

2.1. Workflow definition and execution. Integration of distributed services is often modelled as workflow applications which are most often described as DAGs (Directed Acyclic Graphs). In a DAG G(V,E) a set of vertexes V corresponds to tasks that are needed to accomplish a complex scenario while the set of directed edges E corresponds to time dependencies between the tasks they connect. Such a DAG describes a recipe for a complex scenario, either a business or a scientific application. It does not yet refer to any executables that are supposed to perform the tasks thus such a workflow is called abstract. For each task, there may be several services capable of executing it, albeit at different QoS terms. Each of such services may differ in:

*Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, 11/12 Narutowicza Street, 80-233, Gdansk, Poland, (pczarnul@eti.pg.gda.pl) http://fox.eti.pg.gda.pl/~pczarnul.

execution time, cost, location which may impact the communication cost of staging in and out data, reliability, availability, conformance to standards etc. Assignment of a particular service to each workflow task results in a complete solution to how the abstract workflow is to be executed and makes the workflow concrete. Such a set of services results in the final resulting QoS, in particular the execution time and cost of the workflow. A Workflow Management System (WfMS) is a system that allows a user to:

1. define a workflow, often using a graphical editor to specify tasks, draw edges of the workflow G graph,
2. for each task:
   - specify functional requirements i.e. what the task is supposed to do or
   - assign a set of services, each of which is capable of executing the task, possibly at different QoS terms,
3. define QoS goals for the workflow in terms of a global (for the whole workflow) optimisation goal and global and/or local (for a particular task) constraints, possibly with multiple criteria [29, 25]. For instance, the optimisation goal might be to select such services (one service per task) such that the workflow execution time is minimised and the cost of the selected services is below a given threshold [36, 33],
4. perform workflow scheduling and optimisation i.e. select such services to meet the optimisation goal. In general, optimal selection will be infeasible computationally for large configurations which forces to fall back to heuristic [31] service selection and scheduling algorithms,
5. execute the workflow in a real distributed environment,
6. track the statuses of previously run workflow applications,
7. fetch results of the workflows.

Workflow applications can be categorised into the following types:

- scientific [34], characterised by:
  - structure: in which mainly compute-intensive tasks are executed on input data of large or moderate size; usually many parallel paths execute parts of computations on possibly smaller fractions of data,
  - QoS: traditionally mainly the workflow execution time and cost (corresponding to e.g. hiring HPC resources) were used.
- business, characterised by:
  - structure: usually operates on data of smaller size than scientific workflows; control flow is more complex (possibly requires more control structures than in the regular DAG) than in scientific workflows,
  - applications: document flow, processing orders in B2B scenarios, handling and processing client orders,
  - QoS: many more metrics considered for services and as the QoS goal than in scientific workflows e.g. availability, accessibility, security, reputation [24, 38].

2.2. Cloud systems. The fundamental assumption of cloud computing is outsourcing compute and storage capabilities which brings several consequences:

- the pay-as-you-go policy instead of the fixed initial cost and only running costs of on-site or grid systems [16],
- no need for maintenance and upgrades of equipment, software updates, handling of compatibility issues among software and hardware components,
- letting the cloud provider to manage computations, data and networking among the software components run on the cloud. This brings data privacy concerns for some businesses and may rule out cloud computing for some of them,
- the possibility of falling into the vendor lock-in trap if the client becomes too much invested and depending on just one provider.

Cloud systems may expose various kinds of services (each next layer makes use of the former in a layered architecture, from bottom to top closer to the cloud client):

- IaaS, Infrastructure as a Service. In this case the cloud provider exposes basic components such as computers for computing (such as virtual machines), storage, networks, load balancers, firewalls. Ex-

amples include: Google Compute Engine[1], Amazon Elastic Compute Cloud (EC2)[2], RackSpace Cloud Servers[3], Rack Space Cloud Files[4],
- PaaS, Platform as a Service. In this case, the whole operating platform is provided by the cloud provider that may include components such as an operating system, database server, web server. Clients can run software on the cloud using these components. Examples include: Aneka [23], Google AppEngine[5], Windows Azure[6], RedHat Openshift[7], RackSpace Cloud Sites[8],
- SaaS, Software as a Service. Cloud providers offer software installed on the cloud while cloud clients use the software. The software can be run transparently on virtual machines, updated and maintained by the cloud provider. Examples include: Google Apps[9], Salesforce[10].

There are several software packages that allow building of private or public clouds such as: Eucalyptus[11] (for IaaS with interfaces compatible with Amazon EC2 and S3), OpenStack[12] (for IaaS with interfaces compatible with Amazon EC2 and S3) with OpenStack Compute, OpenStack Storage, OpenStack Networking and OpenStackDashBoard, Open Nebula[13] for building IaaS datacenter vitalisation with a choice of interfaces such as AWS, OCCI and hypervisors such as Xen, KVM, VMWare, Nimbus[14] (provides an implementation of Amazon EC2 interface) for IaaS through deployment of virtual machines on resources and offering to users, Cumulus to provide storage cloud implementation (interface compatible with Amazon S3).

2.3. Workflow management in grid and cloud systems. This section presents the state-of-the-art and recent developments regarding management of distributed workflow applications on clouds, especially compared to running workflows on local and grid systems.

2.3.1. Running Workflow Applications on Cloud vs Grid Systems. Usefulness of cloud computing for large-scale workflows is evaluated against the typical use of grid systems in [16, 28]. One of the crucial differences between running in these two environments is the pay-per-use scheme in cloud computing compared to the one cost policy in grids or local cluster systems [16].

For very large workflows, it is advised to cluster smaller jobs into batches to minimise the scheduling overhead and the overhead of handling too many jobs [28]. FutureGrid was used for distributed processing of workflows across several distributed sites using Eucalyptus and Nimbus. This in fact implemented running on several clouds i.e. sky computing. Additionally, experiments were performed on three separate clouds: Magellan (with Eucalyptus), Amazon (with EC2) and FutureGrid (with Eucalyptus) with very similar results in terms of performance taking into consideration particular configurations.

Cloud-based systems offer several benefits compared to grid systems for running distributed workflow applications [17]:
- dynamic provisioning of resources that can be ordered dynamically at runtime; the Aneka Cloud [23] is able to scale horizontally to acquire more resources as these are needed. Such resources can be obtained from other clouds such as Amazon EC2 to allow the application submitted to Aneka to complete in the desired time frame. This can be especially useful for scientific workflows in which many paths and services are requested to be executed in parallel. More resources can be ordered at runtime for parallel execution, albeit at the increased cost.
- easier possibility to run legacy applications as particular hardware/software configurations of cloud resources can be booked as opposed to using the fixed set of grid computing sites and nodes.

It should be noted that wide area networks offer much larger latency than clusters which can impact workflow execution times severely [17]. However, this is more important for scientific rather than business

[1] http://cloud.google.com/products/compute-engine.html
[2] http://aws.amazon.com/ec2/
[3] http://www.rackspace.com/cloud/cloud_hosting_products/servers/
[4] http://www.rackspace.com/cloud/cloud_hosting_products/files/
[5] https://developers.google.com/appengine/
[6] http://www.windowsazure.com
[7] https://openshift.redhat.com/app/
[8] http://www.rackspace.com/cloud/cloud_hosting_products/sites/
[9] http://www.google.com/Apps
[10] http://www.salesforce.com/eu/
[11] http://www.eucalyptus.com/
[12] http://openstack.org
[13] http://opennebula.org/
[14] http://www.nimbusproject.org/docs

oriented workflow applications.

2.3.2. Systems and Environments. Many workflow management systems have been proposed, especially for grid computing. These include:

- Taverna[15], Kepler[16] [21], Triana[17] [22], Galaxy, Pegasus [11, 12], Askalon [30], Conveyor [20],
- Tavaxy [1] a system for definition and execution of workflow applications based on patterns. It integrates Taverna and Galaxy and allows to either run the whole system in a cloud or launch a part of the workflow on the cloud,
- Meandre[18] – a system for composition semantic enabled flows for data processing. It allows creation of reusable components and RDF is used to standardise publishing.

There are also other business oriented workflow solutions such as:

- Chronos Workflow Platform[19] - a system for automation of repeated business processes. It handles complex flows, parallel processes and external interfaces. It is suitable for any many types of business process including finance, CRM, HR, R&D, marketing, administration, logistics, production.
- Affinity Live[20] - a platform for managing business processes in one place in the cloud. Suitable for e.g. project management, sales, invoicing. It integrates e.g. with Google Apps, Microsoft Exchange Server.

The workflow representation can be made in various forms [15], starting from XPDL, ebXML through Petri-Nets used in Triana, BPEL in Akogrimo up to OWL-based such as OWL-WS in NextGrid. Some systems such as BeesyCluster can use proprietary representations which can be also exported to widely known formats such as BPEL. SHIWA (SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs)[21] aims at interoperability between various workflow systems and transformation of workflow representations.

Running workflow applications on top of several different, distributed resources is presented in [23]. Through an Aneka plugin the workflow engine can use the Aneka Cloud. An EC2 plugin allows to access Amazon Web Services. Additionally, a local cluster with a fixed number of resources can be utilised by the workflow engine. This in fact implements running a workflow on top of several distinct environments including clusters, grids and clouds. The workflow management system presented in [23] supports Aneka, Globus, PBS. The plugins allow to transfer data to and from resources, monitoring statuses of started workflow tasks along with energy consumption. It is demonstrated that a scientific workflow application for evolutionary multi-objective optimisation scales well in terms of the number of iterations of the algorithm when adding new virtual machines to the system.

There are several solutions for integration of software and resources on clouds, oriented on business cases and workflows rather than scientific applications:

1. Metastorm Smart Business Workspace [27] deployed on Microsofts Azure cloud as a version of the on-premise version of the Metastorm software,
2. Amazon Simple Workflow[22] that allows to define, run and control business workflows spanning cloud-based, on-premise or both types of systems focusing on the business logic. The workflow is to represent a business scenario such as processing orders on a Web site including: management of orders, various payment options including charging credit cards, notification, management of shipping items, inventory, returns etc. Several concepts are introduced including:
   - actions corresponding to workflow tasks,
   - activity workers implementing the tasks i.e. services,
   - decider that decides on the workflow logic i.e. checking if a condition is satisfied so as to process appropriate actions,
   - domain a collection of related workflows. Workflows can be managed through the AWS Management Console. The payment scheme for running workflows is pay-as-you-go with the usual AWS charges for data transferred out of the workflow.

[15] http://www.taverna.org.uk/
[16] https://kepler-project.org/
[17] http://www.trianacode.org/
[18] http://seasr.org/meandre/documentation/architecture/
[19] http://www.chronosworkflow.com/
[20] http://www.affinitylive.com/product/benefits/powerful-workflow-and-business-processes/
[21] http://www.shiwa-workflow.eu/
[22] http://aws.typepad.com/aws/2012/02/amazon-simple-workflow-cloud-based-workflow-management.html

3. RunMyProcess platform for development of business workflows using Google Apps. Several examples are cited[23]:
   - purchase-order management in aerospace consultancy,
   - incident management system for a travel agency,
   - project approval process by a bank.
4. OneSaaS[24] is a SaaS cloud integration platform for integration of separate software systems running on separate clouds or sites. It focuses on applications such as CRM, eCommerce, invoicing, email marketing, event management, project and team management, accounting.
5. Questetra BPM Suite[25] - a SaaS system for business process management with a web-based interface.

2.3.3. Scheduling algorithms. Workflow scheduling [32, 35] requires in fact two steps in order to achieve the stated QoS goal (such as minimisation of the workflow execution time) while keeping other QoS constraints (such as the total cost of selected services must not exceed the given threshold): selection of a service for each task, running the service on the given resource at a particular moment in time.

Static scheduling takes place when services are selected and scheduled before the workflow application is executed. This requires the knowledge about services capable of executing particular tasks upfront. It is often the case that some services fail during execution, become unavailable while the workflow is in progress or new, more interesting in terms of QoS optimisation, services appear. In such a case, dynamic rescheduling must be adopted that refreshes the list of available services at runtime. Consequently, execution time and cost may differ from run to run.

Since solving the workflow scheduling problem optimally is NP-hard, heuristic algorithms need to be adopted for large workflows. There is a wide spectrum of literature on workflow scheduling algorithms on grid systems [32, 4, 5, 14]. The following types of algorithms were suggested for workflow scheduling on grid systems:

1. ILP (Integer Linear Programming) methods [13],
2. genetic algorithms [34],
3. divide-and-conquer where the initial DAG is partitioned into smaller DAGs with modified constraints and solutions to smaller problems (sub-graphs) constitute the final solution,
4. GAIN [25] in which a viable solution is found first and then iteratively improved by selection of better (in terms of QoS) services.

There is a survey of workflow scheduling algorithms for cloud environments in [2]. Nine algorithms meant for cloud environments are compared in terms of scheduling methods, scheduling parameters, goals and supporting tools. It is concluded that most of the algorithms consider workflow execution time and cost (as those developed previously for grid systems), some consider resource utilisation. According to the survey, none of the algorithms take into account reliability nor availability.

Optimisation of workflow makespan on cloud systems using a traditional list ordering is proposed in [18]. First, urgency (high or low) of jobs (tasks) determines ordering of these. Secondly, importance (high or low) calculated for resources by a resource manager determines how resources are used first.

2.3.4. Applications. Apart from business workflows run in dedicated systems mentioned above, there are several scientific workflow applications which were suggested and executed in distributed environments, some involving cloud environments. Some of them are listed below:

1. protein analysis workflow (run in Taverna and Tavaxy) [1],
2. metagenomics workflow (the most compute-intensive part of the workflow was run on a cloud using the Amazon WS cloud along with the S3 storage) [1],
3. generation of periodograms that aims at identification of periodic signals in light curves that record brightness of stars over time. This can be done by handling separate frequencies in parallel [28],
4. evolutionary multi-objective optimisation: several instances of genetic algorithms can be run in parallel; it is demonstrated that adding new virtual machines (EC2 compute resources) can significantly increase the number of iterations in the simulation [23].

3. Integration of distributed business and scientific workflows using grid and cloud computing. In this section, we list both the challenges already raised in the literature as well as additional aspects as we

[23] http://www.runmyprocess.com/
[24] http://www.onesaas.com/
[25] http://store.questetra.com/en/

see to be needed for seamless integration of both scientific and business services in integrated grid and cloud computing environments.

Furthermore, solutions to these problems are suggested as extensions to the BeesyCluster[26] environment which already contains a workflow management subsystem for distributed service-based workflows with dynamic rescheduling, a market of services and a pluggable architecture for scheduling algorithms.

3.1. Challenges and Problems.

3.1.1. Preparation and Configuration for using Clouds. In case of ready to use solutions such as IaaS, one needs to be acquainted with the API offered by the cloud provider and have a client ready to use it. In case of private clouds, these require setting up.

Setting up a virtual cluster, virtual machines, preparation of images to be uploaded to cloud resources is discussed in [16]. When setting up own clouds for scientific workflows, much configuration and installation of tools must be done for running distributed workflows using virtualised cloud resources. Tools such as Virtual Workspace Service, Xen, Nimbus were used for virtualised resources. Pegasus, Condor DAGMan as well as GridFTP and GRAM were used to run workflows [16].

On the one hand, preparation of virtual machine images with necessary configuration for individual applications results in provision of necessary scalability thanks to using clouds compared to a local environment. On the other hand, though, it requires both effort in preparation of the images and overhead of setting up an environment [28]. The process of setting up, configuration and deployment of virtual clusters out of collections of virtual machines is called contextualisation [17]. It is complex to perform manually thus tools such as Nimbus Context Broker are suggested to make this process more automatic [17]. It is used to manage the virtual cluster and start appropriate services.

3.1.2. Software Stack for Distributed Workflows. There are more tools necessary to run workflows on clouds than just configuration of the virtual machines and clusters on cloud resources [17]. This involves a layer on top of clouds if workflows are to be run on geographically distributed resources, either cloud or grid based.

Firstly, if more different clouds, grids and local systems are to be used within one workflow, an integrating layer is needed with proper plugins for all underlying cloud, grid providers and local cluster systems such as PBS, LSF etc. This will allow sky computing [19] by using all clouds in a single workflow.

Secondly, in order to execute workflow applications on top of these resources, proper workflow management software communicating through the plugins with the resources are needed. Examples of such tools include:

- Pegasus and Condor DAGMan [16],
- Cloudbus WfMS (Workflow Management System) along with one or more of the following: Aneka, PBS, Globus [23],
- BeesyCluster with its WfMS and PBS, LSF on clusters [6]

Complex configuration and the lack of appropriate tools to set up and run workflows on clouds is listed as one of crucial challenges [17].

3.1.3. Efficient Data Management and Storage for Running Distributed Workflow Applications. Running workflow applications requires copying input data to locations of computing services and output data to following services. Communication cost, in case of large data sizes, can visibly impact the total workflow execution time if locations of such services are far from each other or the client. For instance, in [28] larger workflow execution time on an Amazon cloud compared to Magellan and FutureGrid located in the same state as the client is attributed to poor WAN performance apart from slightly slower processors on this cloud. Cloud systems may offer shared file systems that reduce communication costs within the cloud.

Furthermore, as we proposed in [10], software agents may be engaged for management of data at the workflow management layer and migrate to nodes closer to computing services so that communication time between services is minimised. The very same approach can be used for sky computing in case of geographically distributed clouds.

Several problems exist related to data management in distributed workflow applications run on clouds [17] [37]:

[26] https://lab527.eti.pg.gda.pl:10030/ek/AS_LogIn

- costly data movement between cloud resources, especially if more clouds are used for sky computing. It may be impossible to track the actual locations of data where it is stored. Cloud providers may charge for the amount of data transferred.
- it is not straightforward to set up a shared file system for use on a virtual cluster from a cloud provider. This may not be sufficient if more cloud providers are used. A widespread virtual file system would be an ultimate goal from the workflow perspective in terms of ease of use.
- particular clouds can offer specific APIs for submission and management of data. For instance, in Aneka [23] there is a Storage Service that allows to store input and output data of submitted tasks. Data can be obtained from the client machine, an FTP server or an Amazon S3 storage.
- the next problem related to data when running workflows on top of several clouds and in general often raised for sky computing is data exchange among several clouds in terms of coherency [23] and formats.

3.1.4. Integration of Business and Scientific Services. Most workflow management systems and solutions, as presented above, are dedicated to either scientific or business applications, not both. In some cases both scientific and business aspects appear in a single workflow. For instance, a company performing designs of buildings and bridges must cooperate with external customers and businesses (business part) as well as use HPC (High Performance Computing) services in order to perform multiple analyses of how their designed structures stand various parameters of wind, flood etc. Thus, one workflow application mixing both component types would be needed to model such a project. Such a capability would need to allow modelling and execution of the following types of services in one workflow application:

1. processing documents (e.g. orders, specifications or files),
2. involvement of human actors (e.g. to approve of or order another execution of the same service, or engage more human actors in voting on which path of the workflow should follow),
3. launching HPC long-running simulations hiding low-level details such as queueing systems, access to HPC resources etc.,
4. involvement of several distinct administrative parties (e.g. different companies trying to check if an initial task defined by a client can be solved by a design company or allowing cooperation of various partners in a consortium).

Integration of both types: scientific and business is analysed in [26]. The proposed approach is that there is a business workflow with possibly human tasks that controls the main flow and scientific, lower-level workflows are launched in certain tasks of the business workflow. Nodes of scientific workflows can spawn scientific simulations. Human tasks correspond to e.g. to provide information or make a decision. Still, there are new requirements and innovations that are suggested by the author:

1. incorporation of workflow patterns into workflows with business, scientific and human elements,
2. describing all services (business, scientific and human) using same ontologies in terms of both functional and QoS descriptions and incorporation of these into scheduling.

3.1.5. Dynamic Monitoring and History of Reliability and Availability. Other than dynamic provisioning of resources which is the advantage of cloud computing, there is the issue of QoS of the cloud services being offered: IaaS, PaaS or SaaS. According to the survey of workflow scheduling algorithms for clouds [2], there is a clear need for consideration of reliability and availability in scheduling workflows. None of the nine algorithms for clouds consider these.

The above challenge is closely related to the need for a registry of cloud services from various cloud providers [23]. Then, a proper scheduling policy considering many users who want to run multiple workflows should be developed in order to offer fair distribution of resources. This very much resembles the already known solutions in queueing systems for clusters such as PBS, LSF, LoadLeveler albeit at the higher level of the software stack. This should take into account storage and communication costs. This is again a known problem in scheduling workflows [3] but now defined higher at the multi-cloud level for sky computing [19].

Thus, this results in the need for monitoring and ranking of the following features:

- reliability,
- availability,
- rate of changes in QoS terms (such as the cost of computing and storage power in IaaS, price of access to a platform configuration in PaaS, price of access to software in SaaS, rate of software version updates in SaaS). While there exist tools for runtime monitoring and selection of best e.g. IaaS offers for desired

settings (such as required compute power, memory size, storage capacity) such as Cloudorado[27], this needs to be extended for the other metrics as well.

3.1.6. Reusable Workflow Patterns/Templates. It is much easier to construct workflow applications out of ready-to-use and reusable patterns. Most of the workflow management systems use the DAG model in which the following control statements are available:

- sequence,
- fork different ways of partitioning data among following workflow nodes are possible [1], [7] for instance, same data may be sent to each of the following nodes or data may be partitioned among following nodes.
- join.

Paper [1] suggests several additional patterns for workflow definitions, considering for control patterns also:

- multi-choice fork one of several following tasks is executed depending on the user-defined condition,
- iteration a certain task is repeated a predefined number of times or the number of iterations may also depend on the output data and a condition set on it

and for data patterns the following ones:

- list operations are performed on each of the list elements separately,
- product out of two input lists with elements, operations are performed on pairs of elements on corresponding places in the lists (dot product) or all combinations of elements in the two lists (cross product),
- data select depending on the outcome of a user-defined test, one or the other of input data is passed,
- data merge concatenation of input data lists is passed further.

Still, according to the author, more patterns are needed, especially for integration of business and scientific workflows. The author suggests addition of the following:

- consideration of a human behaviour as a service in the workflow. This would accept input data and produce output data as the other services and be accessible through various endpoints such as email, SMS etc.
- pattern: inititate_m_of_n pass through a given task if m out of n services associated with the task have fired. This can implement e.g. voting in a company if 2 out of 5 board of directors need to approve of the given resolution.

3.1.7. Distributed Management of Workflows using Clouds. Another aspect of the workflow management is how distributed the workflow execution can be. Many workflow management systems invoke distributed services but are centralised in nature. Distributing the execution engine can bring several benefits [10]:

- optimisation of data transfer costs between services/clouds as the managing party can be closer to the services/clouds if direct transfer is not possible (there are reasons for this e.g. not wanting to pass security credentials to one service/cloud to access another),
- no need for costly global synchronisation when executing the workflow ability to parallelize execution better.

3.1.8. Incorporation of Semantic Search. Semantic search for services is a challenge that is needed for automatic building of workflow applications that realise a goal defined by a user. First, out of knowledge acquired previously by the system, a graph of tasks needed to perform a given complex task (workflow) is built. Secondly, based on semantic and intelligent search methods [9], services capable of executing particular tasks are found and selected to optimise the given QoS goal and meet additional QoS constraints.

In the context of cloud utilisation (next to grid and on-site services) for workflow applications, semantic search can be useful for searching for:

- IaaS on which executables could be run alternatively to the services already assigned to workflow tasks (with proper QoS values such as compute power resulting in certain execution time of the service and the cost of the given IaaS),
- particular SaaS that might perform the given task in the workflow (with proper QoS values such as the execution time and cost).

4. Solutions and Recommendations. This section presents some solutions and recommendations on how solutions to the challenges identified in the preceding section might be designed and implemented when

[27] http://www.cloudorado.com/

running distributed workflow applications using clouds. As an example, the existing BeesyCluster middleware and the workflow management system are used as the basis and extensions suggested by the author for these.

4.1. BeesyCluster Middleware and Workflow Management System. BeesyCluster [6] is a middleware that allows distributed users to access and use distributed resources. It offers WWW (Figure 4.1) and Web Service interfaces [8]. Users can manage resources such as clusters or servers as well as develop and use software installed there through system accounts on these resources. BeesyCluster allows single sign-on to use those multiple resources. Furthermore, applications, whether run on regular servers or clusters, can be published as BeesyCluster services to which BeesyCluster users or groups are granted privileges. Such services can be incorporated into the embedded workflow management system module (WfMS) [6]. The BeesyCluster WfMS allows modelling workflow applications as DAGs defined before with assignment of services to workflow tasks (Figure 4.2). Such services may be either own services developed by the workflow author or made available by others, from either other clusters or grids. The WfMS allows monitoring statuses of previously run workflow applications (Figure 4.3). The following section shows how the aforementioned solutions, including incorporation of services from clouds can be used in this WfMS.

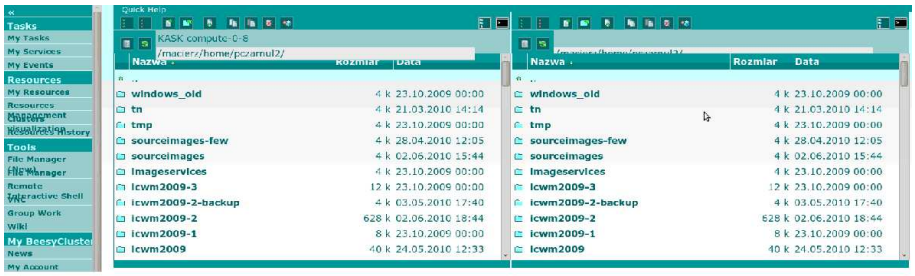

Fig. 4.1: BeesyClusters WWW interface.

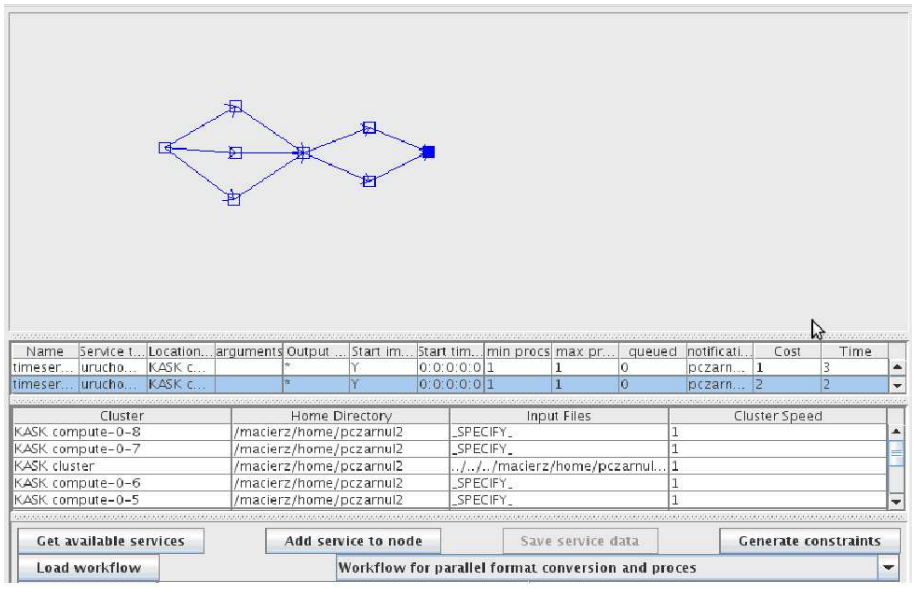

Fig. 4.2: BeesyClusters WfMS editor.

4.2. BeesyCluster Architecture and Extensions for Cloud Computing. The following extensions are proposed to solve the challenges identified in the previous section: integration of cloud-based services into

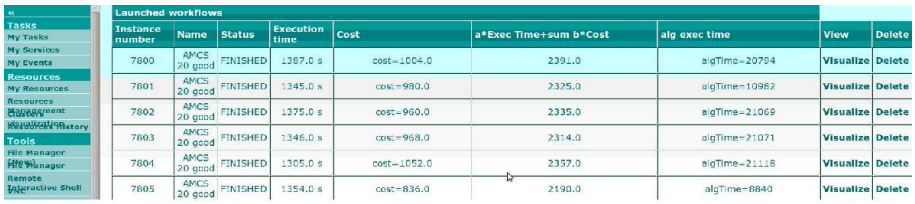

Fig. 4.3: Results of workflows runs.

workflow applications discussed as extensions of the already implemented solution (Figure 4.1):

- architecture and plugins the architecture of BeesyCluster is easily extended with support for cloud providers as follows:
  - similarly to cluster queueing systems (stored in table ra_qsystem), grid middlewares are added (stored in table ra_gmiddleware) and cloud interfaces (stored in table ra_cloudapi),
  - similarly to particular clusters (stored in table ra_cluster), grid middlewares are added (stored in table ra_gridmiddleware) and clouds (stored in tables ra_cloud _iaas, ra_cloud_paas, ra_cloud_saas). Proper interfaces are referenced in tuples from these database tables.
- workflow scheduling using clouds. In BeesyCluster, as in many other workflow management systems, services are distinguished each of which is assigned to a workflow task that it can perform. Each service has executable code associated with it along with QoS parameters at least the execution time and the cost. In BeesyCluster, a user may deploy a service on a user account of a cluster or a server available to them or make an executable available for download (which also constitutes a service i.e. that the executable can be downloaded for a fee). In the latter case, it is possible to find a matching (in terms of the architecture) environment (such as IaaS) for the executable to upload and run. In this case, two possibilities appear for clouds registered in BeesyCluster:
  - IaaS publish several services (corresponding to one executable run on the given IaaS) i.e. options of running the executable on the IaaS with various sets of parameters such as memory size, compute power and obviously corresponding execution time and cost.
  - SaaS publish a SaaS service as a BeesyCluster service with proper cloud access as defined above.
- services in BeesyCluster - in order to maintain compatibility with the existing service subsystem [6], IaaS and SaaS services are registered in the system along with standard cluster based ones. This allows incorporation of these services into the already available static and dynamic scheduling methods. Thus, the many algorithms available in BeesyCluster, in particular ILP-based, genetic algorithm, divide-and-conquer, GAIN can be used for running workflow applications on cluster, grid and cloud resources at the same time. Out of these, the genetic approach, in which a chromosome represents assignment of particular services to workflow tasks and launching services at particular moments in time, is the most general in terms of QoS goal and constraints. It does not impose e.g. linearity on constraints such as ILP. It can be noted that the algorithms developed for matching of services based on compatibility of their inputs and outputs [10] can be reused in the proposed environment as well. Information on success or failure of service invocation is used to update information of compatibility of inputs and outputs of pairs of services.
- semantic and intelligent search mechanisms developed in [9] can then be used to search for:
  - IaaS resources and registration for particular executables of interest (searching for IaaS cloud providers),
  - SaaS software and registration as BeesyCluster services (searching by name, input, output data formats). It would certainly be useful if SaaS is described in a format that makes such search easier e.g. in OWL-S with description of the service in appropriate fields.
- patterns including human interaction extension of the regular DAG:
  - control patterns with a service corresponding to a human interaction (such a service is invoked by sending an email to a specified address with attachments that correspond to input data; it then finalises its execution by invoking a proper Web Service in BeesyCluster returning data), more human actors can be registered as several services and one is selected based on the history e.g.

availability, learnt response time, cost (if defined) etc.
    * implementing the initiate_m_of_n services in this case, n services are launched in parallel and as soon as m have returned, their output is concatenated and passed further,
  – data patterns with:
    * operations on either individual data items (represented as files in BeesyCluster) or invoking the service for all data files at once,
    * data partitioning among forked, following tasks with several possibilities: send all data to all tasks, partition the data to minimise the QoS goal [6], denote what data needs to be sent to all tasks and partition the rest, denote manually what data is sent to what task.
- workflow model for both business and scientific workflow applications BeesyCluster supports this by allowing easy registration of new resources (be it university clusters, company servers or public clouds). Additionally, the basic description of each service with its execution time and cost can be easily extended by the author of the workflow with any additional QoS metrics, specifically suitable for either business or scientific uses. This is then immediately reflected in the constraint model and considered during optimisation [6]. This means that particular QoS metrics can vary from workflow to workflow. Additionally, it is worth to note that in BeesyCluster there is a natural (by design) market of services. Such services (either scientific or business) can be put on auction and BeesyCluster users can bid to win a privilege to use the given service for a fee. So, on the one hand BeesyCluster has support for low-level HPC queueing, on the other it inherently support a business oriented market of such services. As mentioned above, it is proposed by the author that human services are introduced into the workflow. However, different than in [26], the author proposes that these can be services. This means that several human services can be assigned to a workflow task, out of which one can be selected. The auction functionality applies to any type of BeesyCluster service and could also be used for human services (e.g. for consulting). Furthermore, workflow patterns can be used in such a workflow.
- distributed execution on cloud enabled systems using software agents after the various types of services have been deployed as proposed, both workflow execution methods available in BeesyCluster can be used to manage the execution of the workflow:
  – centralised using a Java EE server on which BeesyCluster and its WfMS are deployed [6],
  – distributed using software agents [10] in which agents locations are optimised to minimise communication costs and consequently the workflow execution time.
- reliability and availability and runtime monitoring of resources similarly to the work on dynamic monitoring of service availability [6], the very same method can be used in the extended version using cloud services. In this case, though, the author proposes to extend monitoring with functions that will convert the measured metrics to a normalised [0,1] quality range where 1 denotes best quality. This will consider not only the metrics, but also their derivatives. As an example:
  – lower execution time of an HPC service results in better quality,
  – higher reliability results in better quality,
  – lower fluctuations of availability of a cloud results in better quality,
  – lower fluctuations of prices of a cloud results in better quality.

Figure 4.4 presents the layered architecture of the proposed solution. Figure 4.5 presents execution of a workflow superimposed on the architecture of the integrated solution. The BeesyCluster middleware allows users (U0, U1 and U2) to define and manage workflow applications through the embedded WfMS. Execution of the workflow is delegated to a group of software agents by passing the description of the workflow in BPEL and proper credentials to access external cluster, grid and cloud systems. Agents vote which one is responsible for execution of which parallel paths of the workflow. The WfMS can execute ready-to-use services installed on clusters, grids or clouds (SaaS) and find resources (IaaS) for executables available in clusters. In one workflow, human services can allow to control flows. Furthermore, human services are treated as all others including QoS parameters, runtime monitoring and reselection. If one human service fails (the person does not respond or has failed to respond in a given time frame), control is passed to an alternative human service as shown in Figure 4.5. Agents report back to the workflow execution module on statuses and gathered QoS information for cluster, grid, cloud services (including human) which are updated in a service directory to be used in further scheduling. The embedded auction module can use this information to allow bidding and winning services by BeesyCluster users.

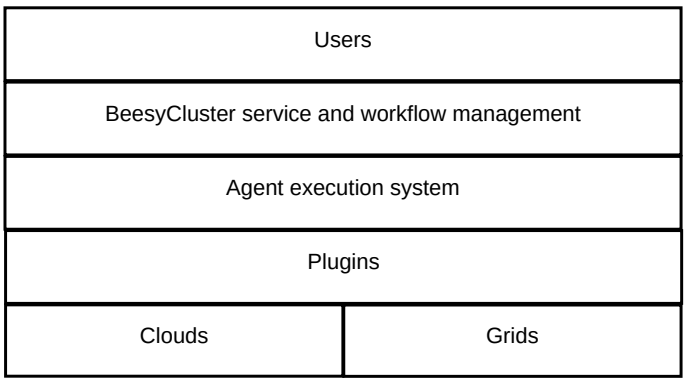

Fig. 4.4: Layers of the proposed architecture for running workflow applications on clouds and grids at the same time

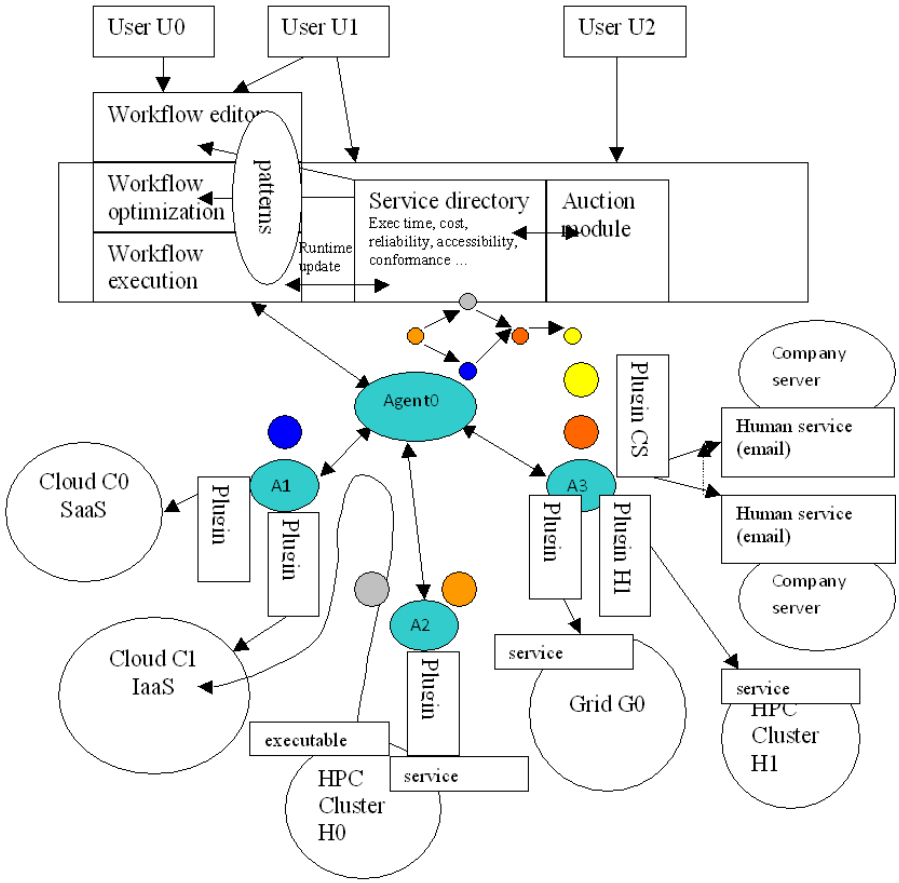

Fig. 4.5: BeesyCluster and agents executing a workflow application

5. Conclusions and Future Research Directions. The paper summarised the current developments in workflow management using on-site, grid and cloud computing with focus on challenges that appear in particular when engaging clouds in workflow management. The following challenges were discussed: preparation of using

a cloud and setting up (in case of private clouds), complex software stack for workflow management using a diverse set of resources (sky computing), data management issues, integration of features for both business and scientific services in one workflow, need for a directory of cluster, grid and cloud based services with monitoring of not only values of particular QoS metrics but also their changes in time (derivatives), reusable patterns for business and scientific uses, distributed execution of workflows composed out of distributed services, integration of semantic search into cloud-enabled workflow applications.

Secondly, suggestions and recommendations were provided on how the listed challenges can be implemented in the BeesyCluster middleware and workflow management system.

While workflow applications span more and more types of systems (local, embedded, cluster, grid and gradually cloud), the author predicts that further integration of workflow processing on these distinct types of systems will progress. For instance, dynamic ad-hoc discovery of services from mobile devices and automatic integration of services using semantic search for both business and compute-intensive tasks in one workflow is yet to follow on a global scale. This can be integrated with common uses (patterns) that should be developed for such integrated systems. A real world application could be activation of a mobile device when speaking to a foreigner in own language, uploading a translator application and voice to an IaaS cloud and conversion to a different language.

REFERENCES

[1] M. Abouelhoda, S. Issa, and M. Ghanem, *Tavaxy: Integrating taverna and galaxy workflows with cloud computing support*, BMC Bioinformatics, 13 (2012).

[2] A. Bala and I. Chana, *A survey of various workflow scheduling algorithms in cloud environment*, in IJCA Proceedings on 2nd National Conference on Information and Communication Technology NCICT, New York, USA, 2011, Foundation of Computer Science, pp. 26–30.

[3] S. Bharathi and A. Chervenak, *Scheduling data-intensive workflows on storage constrained resources*, in Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, WORKS 09, New York, NY, USA, 2009, pp. 1–10. DOI http://doi.acm.org/10.1145/1645164.1645167.

[4] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy, *Task scheduling strategies for workflow-based applications in grids*, in CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, vol. 2, May 2005, pp. 759–767.

[5] S. Chin, T. Suh, and H. Yu, *Adaptive service scheduling for workflow applications in service-oriented grid*, The Journal of Supercomputing, 52 (2010), pp. 253–283.

[6] P. Czarnul, *Modeling, run-time optimization and execution of distributed workflow applications in the JEE-based BeesyCluster environment*, The Journal of Supercomputing, (2010), pp. 1–26. 10.1007/s11227-010-0499-7, http://dx.doi.org/10.1007/s11227-010-0499-7.

[7] P. Czarnul, *Modelling, optimization and execution of workflow applications with data distribution, service selection and budget constraints in BeesyCluster*, in Proceedings of 6th Workshop on Large Scale Computations on Grids and 1st Workshop on Scalable Computing in Distributed Systems, International Multiconference on Computer Science and Information Technology, October 2010, p. 629 636. IEEE Catalog Number CFP0964E.

[8] P. Czarnul, M. Bajor, M. Fraczak, A. Banaszczyk, M. Fiszer, and K. Ramczykowska, *Remote task submission and publishing in beesycluster : Security and efficiency of web service interface*, in Proc. of PPAM 2005, Springer-Verlag, ed., vol. LNCS 3911, Poland, Sept. 2005.

[9] P. Czarnul and J. Kurylowicz, *Automatic conversion of legacy applications into services in beesycluster*, in Proceedings of 2nd International IEEE Conference on Information Technology ICIT'2010, Gdansk, Poland.

[10] P. Czarnul, M. R. Matuszek, M. Wójcik, and K. Zalewski, *Beesybees: A mobile agent-based middleware for a reliable and secure execution of service-based workflow applications in beesycluster*, Multiagent and Grid Systems, 7 (2011), pp. 219–241.

[11] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny, *Pegasus : Mapping Scientific Workflows onto the Grid*, in Across Grids Conference, Nicosia, Cyprus, 2004. `http://pegasus.isi.edu`.

[12] E. Deelman, G. Singha, M.-H. Sua, J. Blythea, Y. Gila, C. Kesselmana, G. Mehtaa, K. Vahia, G. B. Berrimanb, J. Goodb, A. Laityb, J. C. Jacob, and D. S. Katzc, *Pegasus: A framework for mapping complex scientific workflows onto distributed systems*, Scientific Programming, 13 (2005). IOS Press.

[13] A. Gao, D. Yang, S. Tang, and M. Zhang, *Web service composition using integer programming-based models*, in e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on, Sch. of Electr. Eng. & Comput. Sci., Peking Univ., Beijing, China, October 2005, pp. 603–606.

[14] S. K. Garg, R. Buyya, and H. J. Siegel, *Time and cost trade-off management for scheduling parallel applications on utility grids*, Future Generation Computer Systems, 26 (2010), pp. 1344 – 1355.

[15] S. Gogouvitis, K. Konstanteli, D. Kyriazis, G. Katsaros, T. Cucinotta, and M. Boniface, *Achieving Real-Time in Distributed Computing: From Grids to Clouds*, IGI Global, 2012, ch. Workflow Management Systems in Distributed Environments, pp. 115–132. 10.4018/978-1-60960-827-9.ch007.

[16] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, *On the use of cloud computing for scientific workflows*, in IEEE Fourth International Conference on eScience '08, 2008, pp. 640–645. doi: 10.1109/e-Science.2008.167.

[17] G. Juve and E. Deelman, *Grids, Clouds and Virtualization*, Springer, 2010, ch. Scientific Workflows in the Cloud, pp. 71–91.
[18] N. Kaur, T. Aulakh, and R. Cheema, *Comparison of workflow scheduling algorithms in cloud computing*, International Journal of Advanced Computer Science and Applications, 2 (2011).
[19] K. Keahey, M. O. Tsugawa, A. M. Matsunaga, and J. A. B. Fortes, *Sky computing*, IEEE Internet Computing, 13 (2009), pp. 43–51.
[20] B. Linke, R. Giegerich, and A. Goesmann, *Conveyor: a workflow engine for bioinformatic analyses*, Bioinformatics, 27 (2011), pp. 903–11.
[21] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao, *Scientific Workflow Management and the Kepler System*, Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, (2005).
[22] S. Majithia, M. S. Shields, I. J. Taylor, , and I. Wang, *Triana: A Graphical Web Service Composition and Execution Toolkit*, in IEEE International Conference on Web Services (ICWS'04), IEEE Computer Society, 2004, pp. 512–524.
[23] S. Pandey, D. Karunamoorthy, and R. Buyya, *Workflow Engine for Clouds*, Wiley Press, New York, USA, 2011, ch. Cloud Computing: Principles and Paradigms. ISBN-13: 978-0470887998.
[24] C. Patel, K. Supekar, and Y. Lee, *A QoS Oriented Framework for Adaptive Management of Web Service based Workflows*, in Proceedings of the 14th International Database and Expert Systems Applications Conference (DEXA 2003), LNCS, Prague, Czech Republic, September 2003, pp. 826–835.
[25] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos, *Scheduling workflows with budget constraints*, in in Integrated Research in Grid Computing, S. Gorlatch and M. Danelutto, Eds.: CoreGrid series, Springer-Verlag, 2007.
[26] M. Sonntag, D. Karastoyanova, and E. Deelman, *Bridging the gap between business and scientific workflows*, in Proceedings of e-Science 2010, Brisbane, Australia, 2010.
[27] M. Vizard, *Workflow management moves to the cloud.* IT Business Edge, Retrieved June 27, 2012, from http://www.itbusinessedge.com/cm/blogs/vizard/workflow-management-moves-to-the-cloud/?cs=42242, 2012.
[28] J. Vckler, G. Juve, E. Deelman, M. Rynge, and G. Berriman, *Experiences using cloud computing for a scientific workflow application*, in Proceedings of 2nd Workshop on Scientific Cloud Computing (ScienceCloud 2011), 2011.
[29] M. Wieczorek, A. Hoheisel, and R. Prodan, *Towards a general model of the multi-criteria workflow scheduling on the grid*, Future Generation Computer Systems, 25 (2009), pp. 237 – 256.
[30] M. Wieczorek, R. Prodan, and T. Fahringer, *Scheduling of scientific workflows in the askalon grid environment*, SIGMOD Rec., 34 (2005), pp. 56–62.
[31] Yingchun, X. Li, and C. Sun, *Cost-effective heuristics for workflow scheduling in grid computing economy*, in GCC '07: Proceedings of the Sixth International Conference on Grid and Cooperative Computing, Washington, DC, USA, 2007, IEEE Computer Society, pp. 322–329.
[32] J. Yu and R. Buyya, *A taxonomy of workflow management systems for grid computing*, Journal of Grid Computing, 3 (2005), pp. 171–200.
[33] J. Yu and R. Buyya, *A budget constrained scheduling of workflow applications on utility grids using genetic algorithms*, in Workshop on Workflows in Support of Large-Scale Science, Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC 2006), Paris, France, June 2006.
[34] ———, *Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms*, Scientific Programming Journal, (2006). IOS Press, Amsterdam.
[35] J. Yu, R. Buyya, and K. Ramamohanarao, *Workflow Scheduling Algorithms for Grid Computing*, Springer, 2008, ch. Metaheuristics for Scheduling in Distributed Computing Environments. in Metaheuristics for Scheduling in Distributed Computing Environments, ISBN: 978-3-540-69260-7, Berlin, Germany, `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.144 .7107`.
[36] J. Yu, R. Buyya, and C.-K. Tham, *Cost-based scheduling of workflow applications on utility grids*, in Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005), IEEE CS Press, Melbourne, Australia, December 2005.
[37] D. Yuan, Y. Yang, X. Liu, and J. Chen, *A cost-effective strategy for intermediate data storage in scientific cloud workflow systems*, in Proceedings of IPDPS 2010, 2010, pp. 1–12.
[38] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng, *Quality driven web services composition*, in Proceedings of WWW 2003, Budapest, Hungary, May 2003.

Edited by: Enn Õunapuu and Vlado Stankovski
Received: Dec 27, 2012
Accepted: Jan. 07, 2013