Chapter 13
SwinDeW-C: A Peer-to-Peer Based Cloud Workflow System

Xiao Liu, Dong Yuan, Gaofeng Zhang, Jinjun Chen, and Yun Yang

13.1 Introduction

Workflow systems are designed to support the process automation of large scale business and scientific applications. In recent years, many workflow systems have been deployed on high performance computing infrastructures such as cluster, peer-to-peer (p2p), and grid computing (Moore, 2004; Wang, Jie, & Chen, 2009; Yang, Liu, Chen, Lignier, & Jin, 2007). One of the driving forces is the increasing demand of large scale instance and data/computation intensive workflow applications (large scale workflow applications for short) which are common in both eBusiness and eScience application areas. Typical examples (will be detailed in Section 13.2.1) include such as the transaction intensive nation-wide insurance claim application process; the data and computation intensive pulsar searching process in Astrophysics. Generally speaking, instance intensive applications are those processes which need to be executed for a large number of times sequentially within a very short period or concurrently with a large number of instances (Liu, Chen, Yang, & Jin, 2008; Liu et al., 2010; Yang et al., 2008). Therefore, large scale workflow applications normally require the support of high performance computing infrastructures (e.g. advanced CPU units, large memory space and high speed network), especially when workflow activities are of data and computation intensive themselves. In the real world, to accommodate such a request, expensive computing infrastructures including such as supercomputers and data servers are bought, installed, integrated and maintained with huge cost by system users. However, since most of these resources are self-contained and organised in a heterogeneous way, resource scalability, i.e. how easily a system can expand and contract its resource pool to accommodate heavier or lighter work loads, is very poor. Due to such a problem, on one hand, it requires great cost, if not impossible, to recruit external resources to address "resource insufficiency" during peak periods; on the other hand,

X. Liu (✉), D. Yuan, G. Zhang, J. Chen, and Y. Yang
Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn, Melbourne, Australia 3122
e-mails: {xliu@groupwise.swin.edu.au; dyuan@groupwise.swin.edu.au; gzhang@groupwise.swin.edu.au; jchen@swin.edu.au; yyang@groupwise.swin.edu.au}

it cannot provide services to others during off-peak periods to make full advantage of the investment. In current computing paradigms, workflow systems have to maintain their own high performance computing infrastructures rather than employ them as services from a third party according to their real needs. Meanwhile, most of the resources are idled except for bursting resource requirements of large scale workflow applications at peak periods. In fact, many workflow systems also need to deal with a large number of conventional less demanding workflow applications for large proportion of the time. Therefore, resource scalability is becoming a critical problem for current workflow systems. However, such an issue has not been well addressed by current computing paradigms such as cluster and grid computing.

In recent years, cloud computing is emerging as the latest distributed computing paradigm and attracts increasing interests of researchers in the area of Distributed and Parallel Computing (Raghavan, Ramabhadran, Yocum, & Snoeren, 2007), Service Oriented Computing (Ardagna & Pernici, 2007) and Software Engineering (SECES, 2008). As proposed by Ian Foster in (Foster, Zhao, Raicu, & Lu, 2008) and shared by many researchers and practitioners, compared with conventional computing paradigms, cloud computing can provide "a pool of abstracted, virtualised, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet". Therefore, cloud computing can provide scalable resources on demand to system requirement. Meanwhile, cloud computing adopts market-oriented business model where users are charged according to the usage of cloud services such as computing, storage and network services like conventional utilities in everyday life (e.g. water, electricity, gas and telephony) (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009). Evidently, it is possible to utilise cloud computing to address the problem of resource scalability for managing large scale workflow applications. Therefore, the investigation of workflow systems based on cloud computing, i.e. cloud workflow systems, is a timely issue and worthwhile for increasing efforts.

Besides scalable resources, another principal issue for large scale workflow applications is decentralised management. In order to achieve successful execution, effective coordination of system participants (e.g. service providers, service consumers and service brokers) is required for many management tasks such as resource management (load management, workflow scheduling), QoS (Quality of Service) management, data management, security management and others. One of the conventional ways to solve the coordination problem is centralised management where coordination services are set up on a centralised machine. All the communications such as data and control messages are transmitted only between the central node and other resource nodes but not among them. However, centralised management depends heavily on the central node and thus can easily result in the performance bottleneck. Some others common disadvantages also include: single point of failure, lack of scalability and the advanced computation power required for the coordination services. To overcome the problems of centralised management, decentralised management where the centralised data repository and control engine are abandoned, and both data and control messages are transmitted between all the nodes through general broadcast or limited broadcast communication mechanisms. Thus the performance bottlenecks are likely eliminated and the system scalability can be greatly enhanced. Peer to Peer (p2p) is a typical decentralised architecture. However, without any centralised coordination, pure p2p (unstructured decentralised) where all the peer nodes are communicating with each other through complete broadcasting suffers from low efficiency and high network load. Evidently, neither centralised nor unstructured decentralised management is suitable for managing large scale workflow applications since massive communication and coordination services are required. Therefore, in practice, structured p2p architecture is often applied where a super node acts as the coordinator peers for a group of peers. Through those super nodes which maintain all the necessary information about the neighbouring nodes, workflow management tasks can be effectively executed where data and control messages are transmitted in a limited broadcasting manner. Therefore, structured decentralised management is more effectively than other for managing workflow applications.

Based on the above analysis, it is evident that cloud computing is a promising solution to address the requirement of scalable resource, and structured decentralised architecture such as structured p2p is an effective solution to address the requirement of decentralised management. Therefore, in this chapter, we present SwinDeW-C (**Swin**burne **De**centralised **W**orkflow for **C**loud), a peer to peer based Cloud workflow system for managing large scale workflow applications. SwinDeW-C is not built from the scratch but based on our existing SwinDeW-G (Yang et al., 2007) (a peer-to-peer based grid workflow system) which will be introduced later in Section 13.3. As agreed among many researchers and practitioners, the general cloud architecture includes four basic layers from top to bottom: application layer (user applications), platform layer (middleware cloud services to facilitate the development/deployment of user applications), unified resource layer (abstracted/encapsulated resources by virtualisation) and fabric layer (physical hardware resources) (Foster et al., 2008). In order to support large scale workflow applications, a novel SwinDeW-C architecture is presented where the original fabric layer of SwinDeW-G is inherited with the extension of external commercial cloud service providers. Meanwhile, significant modifications are made to the other three layers: the underlying resources are virtualised at the unified resource layer; functional components are added or enhanced at the platform layer to support the management of large scale workflow applications; the user interface is modified to support Internet (Web browser) based access.

This chapter describes the novel system architecture and the new features of SwinDeW-C. Specifically, based on a brief introduction about SwinDeW-G, the architecture of SwinDeW-C as well as the architecture of SwinDeW-C peers (including both ordinary peers and coordinator peers) is proposed. Meanwhile, besides common features for cloud computing and workflow systems, additional new functional components are enhanced or designed in SwinDeW-C to facilitate large scale workflow applications. In this chapter, three new functional components including QoS Management, Data Management and Security Management are presented as the key components for managing large scale workflow applications. The SwinDeW-C prototype system is demonstrated to verify the effectiveness of

SwinDeW-C architecture and the feasibility of building cloud workflow system based on existing grid computing platform.

The remainder of the paper is organised as follows. Section 13.2 presents the motivation and system requirements. Section 13.3 introduces our SwinDeW-G grid computing environment. Section 13.4 proposes the architecture for SwinDeW-C as well as SwinDeW-C peers. Section 13.5 presents the new components in SwinDeW-C for managing large scale workflow applications. Section 13.6 presents SwinDeW-C system prototype. Section 13.7 introduces the related work. Finally, Section 13.8 addresses the conclusion and feature work.

13.2 Motivation and System Requirement

In this section, we first present some examples to illustrate the motivation for utilising cloud computing to facilitate large scale workflow applications. Afterwards, based on the introduction of our existing SwinDeW-G grid computing environment, system requirements for cloud workflow systems are presented.

13.2.1 Large Scale Workflow Applications

Here, we present two examples, one is from the business application area (insurance claim) and the other one is from the scientific application area (pulsar searching).

Insurance claim: Insurance claim process is a common business workflow which provides services for processes of such as insurance under employee benefits including, for example, medical expenses, pension, and unemployment benefits. Due to the distributed geographic locations of a large number of applicants, the insurance offices are usually deployed at many locations across a wide area serving for a vast population. Despite the differences among specific applications, the following requirements are often seen in large/medium sized insurance companies: (1) supporting a large number of processes invoked from anywhere securely on the Internet, the privacy of applicants and confidential data must be protected; (2) avoiding management of the system at many different locations due to the high cost for the setting and ongoing maintenance; (3) being able to serve for a vast population involving processes at the minimum scale of tens of thousands per day, i.e. instance intensive; and (4) for better quality of service, needing to handle workflow exceptions appropriately, particularly in the case of instance-intensive processes.

Pulsar searching: The pulsar searching process is a typical scientific workflow which involves a large number of data intensive and computation intensive activities. For a single searching process, the average data volume (not including the raw stream data from the telescope) can be over terabytes and the average execution time can be about one day. In a single searching process, many parallel execution paths need to be executed for the data collected from different beams of the telescope, and each execution path includes four major segments: data collection, data pre-processing, candidate searching and decision making. Take the operation of de-dispersion in the data pre-processing segment as an example. De-dispersion is to generate a large number of de-dispersion files to correct the pulsar signals which are dispersed by the interstellar medium. A large number of de-dispersion files need to be generated according to different choices of trial dispersion and normally take many hours on high performance computing resources. After the generation of large volume of de-dispersion files, different pulsar candidate seeking algorithms such as FFT seek, FFA seek, and single pulse seek will be further implemented. Finally, the results will be visualised to support the decision of human experts on whether a pulsar has been found or not. Generally speaking, the pulsar searching process often has the following requirements: (1) easy to scale up and down the employed computing resources for data processing at different stages; (2) for better QoS, especially efficient scheduling of parallel computing tasks so that every pulsar searching process can be finished on time; (3) decreasing the cost on data storage and data transfer, specific strategies are required to determine the allocation of generated data along workflow execution; (4) selecting trustworthy service nodes, ensuring the security of data storage, especially for those need to be stored for long term.

13.2.2 System Requirements

Based on the introduction about the above two examples, here, we present the system requirements for managing large scale workflow applications. Besides the two fundamental requirements, namely scalable resource and decentralised management, which have been discussed in the introduction section, there are three important system requirements, including: QoS Management, Data Management, and Security Management.

13.2.2.1 QoS Management

It is critical to deliver services with user satisfied quality of service (QoS) in cloud computing environment, otherwise, the reputation of the service providers will be deteriorated and finally eliminated from the cloud market. Generally speaking, there are 5 major dimensions of QoS constraints including time, cost, fidelity, reliability and security (Yu & Buyya, 2005). Among them, time, as the basic measurement for system performance, is probably the most general QoS constraint in all application systems. Especially for large scale workflow applications, temporal QoS is very important since any large delays may result in poor efficiency or even system bottlenecks. Therefore, in this paper, we mainly focus on temporal constraints as the example for QoS management.

For a general cloud workflow application, both global and local temporal constraints are assigned at workflow build time through service level agreement (SLA) (Erl, 2008). Then, at workflow run time, due to the highly dynamic system performance (activity durations with large deviations (Liu, Chen, Liu, & Yang, 2008)), workflow execution is under constant monitoring against the violations of these temporal constraints (Chen & Yang, 2008a, 2010, 2008b). If a temporal violation is detected (i.e. the workflow execution time exceeds the temporal constraint), exception handling strategies will be triggered to compensate the occurring time delays. Therefore, to deliver satisfactory temporal QoS (as well as other QoS constraints), a set of strategies should be adopted or designed to facilitate at least the following three tasks: the setting of QoS constraints, the monitoring of workflow execution against QoS constraint violations, and the handling of QoS constraint violations.

13.2.2.2 Data Management

Large scale workflow applications often come along with data intensive computing (Deelman, Gannon, Shields, & Taylor, 2008), where workflow tasks will access large datasets for query or retrieving data, and during the workflow execution similar amounts or even larger datasets will be generated as intermediate or final products (Deelman & Chervenak, 2008). Data management in cloud workflow systems has some new requirements, which becomes an important issue. Firstly, new data storage strategy is required in cloud workflow systems (Yuan, Yang, Liu, & Chen, in press). In a cloud computing, theoretically, the system can offer unlimited storage resources. All the application data can be stored, including the intermediate data, if we are willing to pay for the required resources. Hence, we need a strategy to cost-effectively store the large application data. Secondly, new data placement strategy is also required (Yuan, Yang, Liu, & Chen, 2010). Cloud computing platform contains different cloud service providers with different pricing models, where data transfers between service providers also carry a cost. The cloud workflows are usually distributed, and the data placement strategy will decide where to store the application data, in order to reduce the total system cost. Last but not least, new data replication strategy should also be designed for cloud workflow systems (Chervenak et al., 2007). A good replication strategy can not only guarantee the security of application data, but also further reduce the system cost by replicating frequently used data in different locations. Replication strategy in the cloud should be dynamic based on the application data's usage rate.

13.2.2.3 Security Management

Security always plays an important role in distributed computing systems (Lin, Varadharajan, Wang, & Pruthi, 2004). To ensure high QoS of these systems, we focus on the security problems brought by different types of components, large volume of heterogeneous data, and unpredictable execution processes. Since some general aspects of system security such as service quality and data security are partially included in the previous QoS and data management components, this chapter emphasises the trust management which plays an important role in the management of SwinDeW-C peers. In the large scale workflow applications, to meet the high requirements of quality and scalability, an efficient and adaptive trust management is an indispensable part of the SwinDeW-C platform (Bhargav-spantzel, Squicciarini, & Bertino, 2007; Winsborough & Li, 2006). Besides, User management is essential to guarantee system security and avoid illegal usage. Facing the complex network structures in the cloud environment, we also need encryption technology to protect privacy, integrity, authenticity and undeniableness.

Given these basic system requirements, cloud computing is a suitable solution to address the issue of resource scalability and p2p is an effective candidate for decentralised management. Meanwhile, to adapt the system requirements of large scale workflow applications, functional components for Data Management, QoS Management and Security Management are required to be designed or enhanced to guarantee satisfactory system performance.

13.3 Overview of SwinDeW-G Environment

Before we present SwinDeW-C, some background knowledge about SwinDeW-G needs to be introduced. SwinDeW-G (**Swin**burne **De**centralised **W**orkflow for **G**rid) is a peer-to-peer based scientific grid workflow system running on the SwinGrid (Swinburne service Grid) platform (Yang et al., 2007).

An overall picture of SwinGrid is depicted in Fig. 13.1 (bottom plane). SwinGrid contains many grid nodes distributed in different places. Each grid node contains many computers including high performance PCs and/or supercomputers composed of significant numbers of computing units. The primary hosting nodes include the

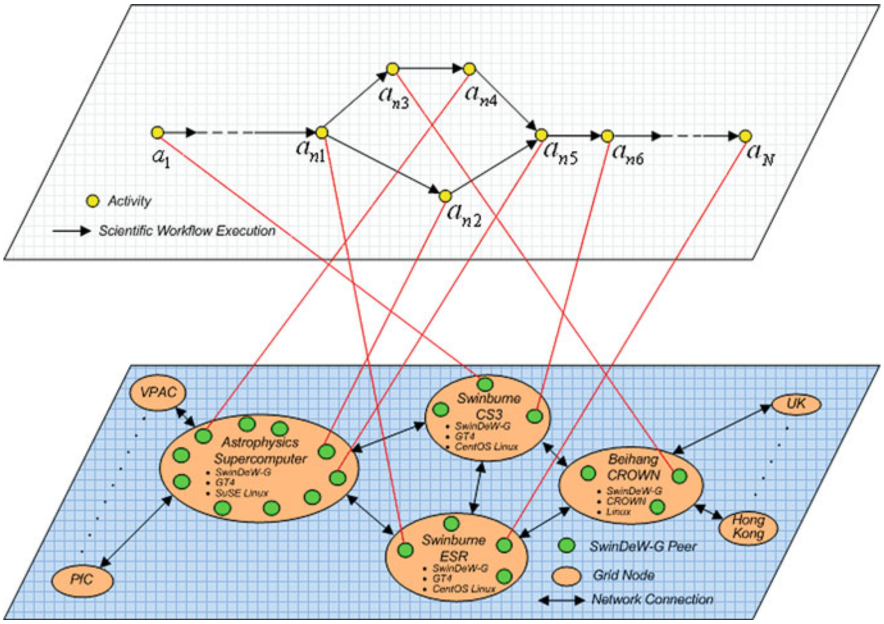


Fig. 13.1 SwinDeW-G environment

Swinburne CS3 (Centre for Complex Software Systems and Services) Node, the Swinburne ESR (Enterprise Systems Research laboratory) Node, the Swinburne Astrophysics Supercomputer Node, and the Beihang CROWN (China R&D environment Over Wide-area Network) Node in China. They are running either Linux, GT4 (Globus Toolkit) or CROWN grid toolkit 2.5 where CROWN is an extension of GT4 with more middleware, and thus is compatible with GT4. The CROWN Node is also connected to some other nodes such as those at the Hong Kong University of Science and Technology, and at the University of Leeds in the UK. The Swinburne Astrophysics Supercomputer Node is cooperating with the Australian PfC (Platforms for Collaboration) and VPAC (Victorian Partnership for Advanced Computing). Currently, SwinDeW-G is deployed at all primary hosting nodes as exemplified in the top plane of Fig. 13.1. In SwinDeW-G, a scientific workflow is executed by different peers that may be distributed at different grid nodes. As shown in Fig. 13.1, each grid node can have a number of peers, and each peer can be simply viewed as a grid service. In the top plane of Fig. 13.1, we show a sample of how a scientific workflow can be executed in the grid computing environment.

The basic service unit in SwinDeW-G is a SwinDeW-G peer which runs as a grid service along with other grid services. However, it communicates with other peers via JXTA (http://www.sun.com/software/jxta/), a platform for p2p communication. As Fig. 13.2 shows, a SwinDeW-G peer consists of the following components:

The Task Component manages the workflow tasks. It has two main functions. First, it provides necessary information to the Flow Component for scheduling and stores received tasks to Task Repository. Second, it determines the appropriate time to start, execute and terminate a particular task. The resources that a workflow task instance may require are stored in the Resource Repository.

The Flow Component interacts with all other modules. First, it receives the workflows definition and then creates the instance definition. Second, it receives tasks from other peers or redistributes them. Third, it decides whether to pass a task to

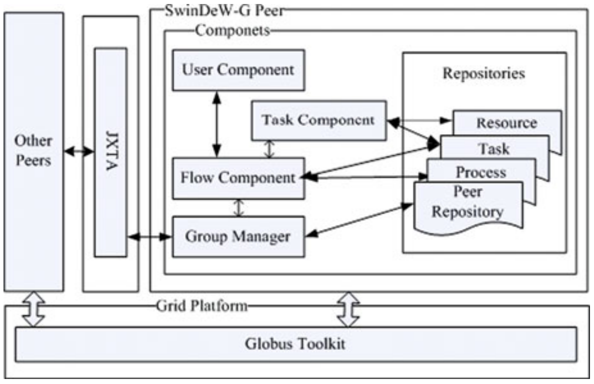


Fig. 13.2 Architecture of a SwinDeW-G Peer

the Task Component to execute locally or distribute it to other peers. The decision is made according to the capabilities and load of itself and other neighbours. And finally, it makes sure that all executions conform to the data dependency and control dependency of the process definitions which are stored in the Process Repository and the Task Repository.

The Group Manager is the interface between the peer and JXTA. In JXTA, all communications are conducted in terms of peer group, and the Group Manager maintains the peer groups the peer has joined. The information of the peer groups and the peers in them is stored in the Peer Repository. While a SwinDeW-G peer is implemented as a grid service, all direct communications between peers are conducted via p2p. Peers communicate to distribute information of their current state and messages for process control such as heartbeat, process distribution, process enactment etc.

The User component is the interface between the corresponding workflow users and the workflow environment. In SwinDeW-G, its primary function is to allow users to interfere with the workflow instances when exceptions occur.

Globus Toolkit serves as the grid service container of SwinDeW-G. Not only a SwinDeW-G peer itself is a grid service located inside Globus Toolkit, the capabilities which are needed to execute certain tasks are also in forms of grid services that the system can access. That means when a task is assigned to a peer, Globus Toolkit will be used to provide the required capability as grid service for that task.

13.4 SwinDeW-C System Architecture

In this section, the system architecture of SwinDeW-C is introduced. SwinDeW-C (**Swin**burne **De**centralised **W**orkflow for **C**loud) is built on SwinCloud cloud computing infrastructure. SwinDeW-C inherits many features of its ancestor SwinDeW-G but with significant modifications to accommodate the novel cloud computing paradigm for managing large scale workflow applications.

13.4.1 SwinCloud Infrastructure

SwinCloud is a cloud computing simulation environment, on which SwinDeW-C is currently running. It is built on the computing facilities in Swinburne University of Technology and takes advantage of the existing SwinGrid systems. We install VMWare (VMware, 2009) on SwinGrid, so that it can offer unified computing and storage resources. Utilising the unified resources, we set up data centres that can host applications. In the data centres, Hadoop (2009) is installed that can facilitate Map-Reduce computing paradigm and distributed data management. The architecture of SwinCloud is depicted in Fig. 13.3.

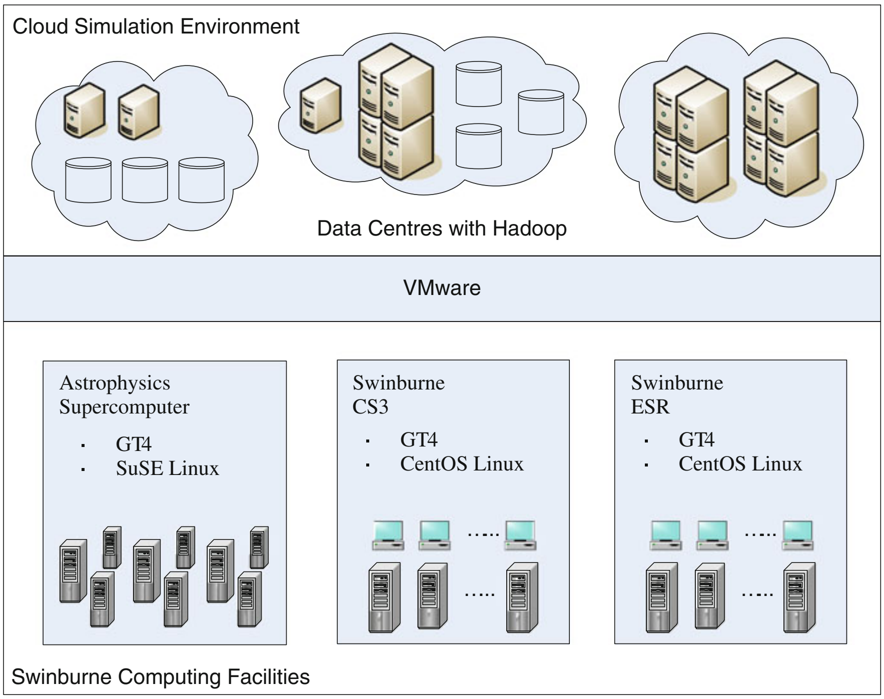


Fig. 13.3 SwinCloud Infrastructure

13.4.2 Architecture of SwinDeW-C

The architecture of SwinDeW-C is depicted in Fig. 13.4. As discussed earlier, the general cloud architecture includes four basic layers from top to bottom: application layer (user applications), platform layer (middleware cloud services to facilitate the development/deployment of user applications), unified resource layer (abstracted/encapsulated resources by virtualisation) and fabric layer (physical hardware resources). Accordingly, the architecture of SwinDeW-C can also be mapped to the four basic layers. Here, we present the lifecycle of an abstract workflow application to illustrate the system architecture. Note that here we focus on the system architecture, the introduction on the cloud management services (e.g. brokering, pricing, accounting, and virtual machine management) and other functional components are omitted here and will be introduced in the subsequent sections.

Users can easily get access to SwinDeW-C Web portal (as demonstrated in Section 13.6) via any electronic devices such as PC, laptop, PDA and mobile phone as long as they are connected to the Internet. Compared with SwinDeW-G which can only be accessed through a SwinDeW-G peer with pre-installed programs, the SwinDeW-C Web portal has greatly improved its usability. At workflow build-time stage, given the cloud workflow modelling tool provided by the Web portal on the application layer, workflow applications are modelled by users as

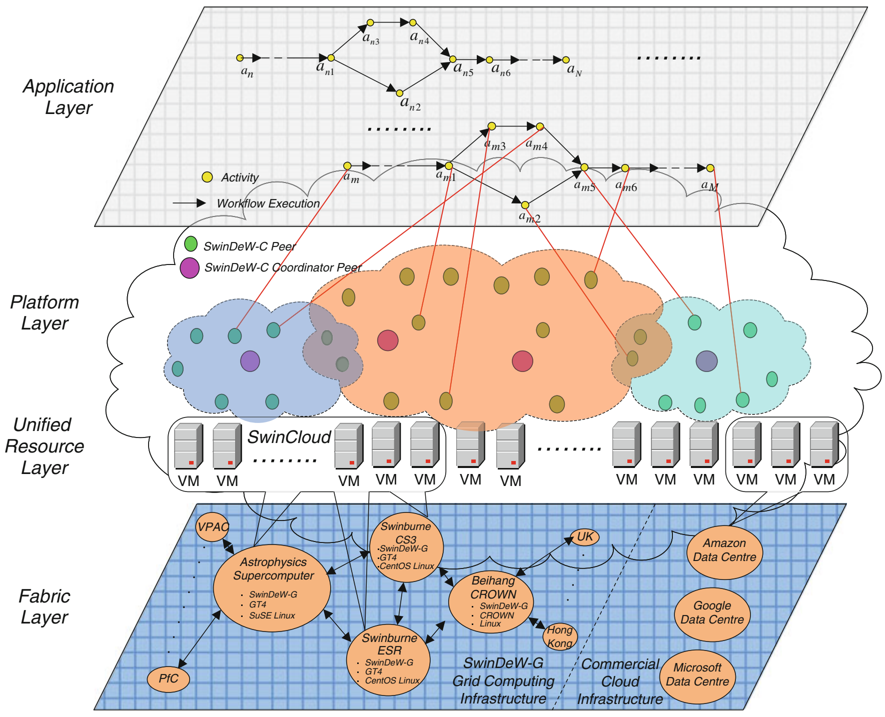


Fig. 13.4 Architecture of SwinDeW-C

cloud workflow specifications (consist of such as task definitions, process structures and QoS constraints). After workflow specifications are created (static verification tools for such as structure errors and QoS constraints may also be provided), they will be submitted to any one of the coordinator peers on the platform layer. Here, an ordinary SwinDeW-C peer is a cloud service node which has been equipped with specific software services similar to a SwinDeW-G peer. However, while a SwinDeW-G peer is deployed on a standalone physical machine with fixed computing units and memory space, a SwinDeW-C peer is deployed on a virtual machine of which the computing power can scale dynamically according to task request. As for the SwinDeW-C coordinator peers, they are super nodes equipped with additional workflow management services compared with ordinary SwinDeW-C peers. Details about SwinDeW-C peers will be introduced in the next section.

At the run-time instantiation stage, the cloud workflow specification can be submitted to any of the SwinDeW-C coordinator peers. Afterwards, the workflow tasks will be assigned to suitable peers through peer to peer based communication between SwinDeW-C peers. Since the peer management such as peer join, peer leave and peer search, as well as the p2p based workflow execution mechanism, is the same as in SwinDeW-G system environment. Therefore, the detailed introduction is omitted here but can be found in (Yang et al., 2007). Before workflow execution, a coordinator peer will conduct an evaluation process on the submitted cloud workflow instances to determine whether they can be accepted or not given the specified non-functional QoS requirements under the current pricing model. It is generally assumed that functional requirements can always be satisfied given the theoretically unlimited scalability of cloud. In the case where users need to run their own special programs, they can upload them through the Web portal and these programs will be automatically deployed in the data centre by the resource manager. Here, a negotiation process between the user and the cloud workflow system may be conducted if the user submitted workflow instance is not acceptable to the workflow system due to the unacceptable offer on budgets or deadlines. The final negotiation result will be either the compromised QoS requirements or a failed submission of the cloud workflow instance. If all the task instances have been successfully allocated (i.e. acceptance messages are sent back to the coordinator peer from all the allocated peers), a cloud workflow instance may be completed with satisfaction of both functional and non-functional QoS requirements (if without exceptions). Hence, a cloud workflow instance is successfully instantiated.

Finally, at run-time execution stage, each task is executed by a SwinDeW-C peer. In cloud computing, the underlying heterogeneous resources are virtualised as unified resources (virtual machines). Each peer utilises the computing power provided by its virtual machine which can easily scale according to the request of workflow tasks. As can be seen in the unified resource layer of Fig. 13.4, the SwinCloud is built on the previous SwinGrid infrastructure at the fabric layer. Meanwhile, some of the virtual machines can be created with external commercial IaaS (infrastructure as service) cloud service providers such as Amazon, Google and Microsoft. During cloud workflow execution, workflow management tasks such as QoS management, data management and security management are executed by the coordinator peers in order to achieve satisfactory system performance. Users can get access to the final results as well as the running information of their submitted workflow instances at any time through the SwinDeW-C Web portal.

13.4.3 Architecture of SwinDeW-C Peers

In this section we will introduce the architecture of a SwinDeW-C peer. As we described above, SwinDeW-C is developed based on SwinDeW-G, where a SwinDeW-C peer has inherited most of the SwinDeW-G peer's components, including the components of task management, flow management, repositories, and the group management. Hence the SwinDeW-G peer plays as the core of a SwinDeW-C peer, which provides the basic workflow management components and communication components between peers. However, some improvements are also made on SwinDeW-C peers to accommodate the cloud computing environment. The architecture of the SwinDeW-C peers is depicted in Fig. 13.5.

Firstly, different from a SwinDeW-G peer, a SwinDeW-C peer runs on the cloud platform. The cloud platform is composed of unified resources, which means the

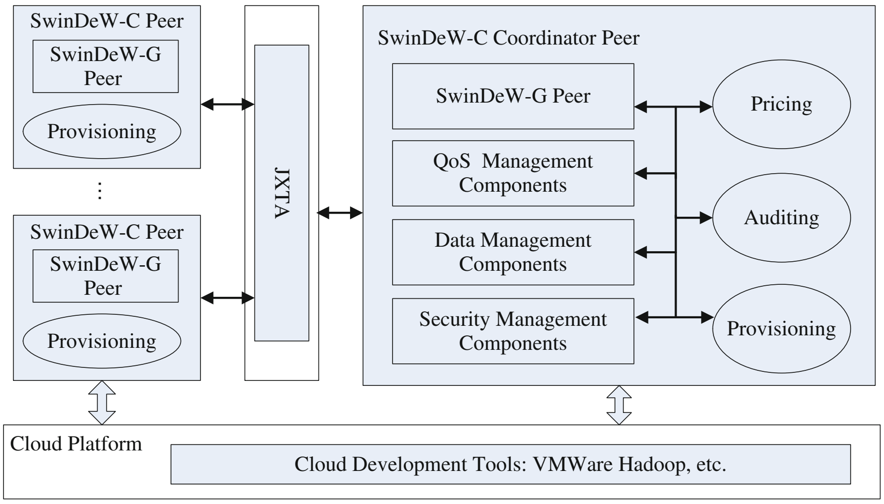

SwinDeW-C Peer
SwinDeW-G Peer
Provisioning
SwinDeW-C Peer
SwinDeW-G Peer
Provisioning
JXTA
SwinDeW-C Coordinator Peer
SwinDeW-G Peer
QoS Management Components
Data Management Components
Security Management Components
Pricing
Auditing
Provisioning
Cloud Platform
Cloud Development Tools: VMWare Hadoop, etc.

Fig. 13.5 Architecture of SwinDeW-C peers

computation and storage capabilities a SwinDeW-C peer can dynamically scale up or down based on the applications' requirements. Unified resources are offered by cloud service providers and managed in resource pools, hence every SwinDeW-C peer has a provisioning component to dynamically apply and release the cloud resources.

Secondly, in cloud computing environment, different cloud service providers may have different cost model, hence we have to set up a coordinator peer within every cloud service provider. The coordinator peer has the pricing and auditing components, which can coordinate the resource provisioning of all the peers that reside in this service provider.

Last but not least, the coordinator peer of SwinDeW-C also has new functional components related to cloud workflow management. As introduced in Section 13.2.2, the system has new requirements for handling the large scale workflow applications. To meet these new requirements, components of QoS management, data management and security management are added to the SwinDeW-C coordinator peer. More detailed descriptions of these components will be given in the following section.

13.5 New Components in SwinDeW-C

In this section, we introduce the new components in SwinDeW-C. As the three system requirements presented in Section 13.2.2, the three new functional components including QoS Management, Data Management and Security Management are introduced.

13.5.1 QoS Management in SwinDeW-C

The basic requirement for delivering satisfactory temporal QoS (as well as other QoS constraints) includes three basic tasks: the setting of QoS constraints, the monitoring of workflow execution against QoS constraint violations, and the handling of QoS constraint violations. Here, take temporal QoS constraints for example, the new QoS management component in SwinDeW-C is introduced.

Temporal Constraint Setting: In SwinDeW-C QoS management component, a probabilistic strategy is designed for setting temporal QoS constraints at workflow build time (Liu, Chen, & Yang, 2008). Specifically, with a probability based temporal consistency model, the one global or several coarse-grained temporal constraints are assigned based on the negotiation result between clients and service providers. Afterwards, fine-grained temporal constraints for individual workflow activities can be derived automatically based on these coarse-grained ones.

Checkpoint Selection and Temporal Verification: At workflow run time, a checkpoint selection strategy and a temporal verification strategy are provided to monitor the workflow execution against the violation of temporal constraints. Temporal verification is to check the temporal correctness of workflow execution states (detecting temporal violations) given a temporal consistency model. Meanwhile, in order to save the overall QoS management cost, temporal verification should be conducted only on selected activity points. In SwinDeW-C, a minimum time redundancy based checkpoint selection strategy (Chen & Yang, 2010, 2007b) is employed which can select only necessary and sufficient checkpoints (those where temporal violations take place).

Exception Handling: After a temporal violation is detected, exception handling strategies are required to recover the error states. Unlike functional errors which are normally prevented by duplicated instances or handled by roll back and re-execution, non-functional QoS errors such as temporal violations can only be recovered by compensation, i.e. to reduce or ideally remove the current time delays by decreasing the durations of the subsequent workflow activities. Since the previous activities have already been finished, there is no way in the real world that any action can reduce their running time. In SwinDeW-C, for minor temporal violations, the TDA (time deficit allocation) strategy (Chen & Yang, 2007a) is employed which can remove the current time deficits by borrowing the time redundancy of the subsequent activities. As for major temporal violations, the ACOWR (ant colony optimisation based two stage workflow local rescheduling) strategy (Liu et al., 2010) is employed which can decrease the duration of the subsequent workflow segments through ant colony optimisation based workflow rescheduling.

In SwinDeW-C, by constant monitoring of the workflow instance and effective handling of temporal violations along workflow execution, satisfactory temporal QoS can be delivered with low violation rates of both global and local temporal constraints. Similar to temporal QoS management, the management tasks for other QoS constraints are being investigated. Meanwhile, since some of them such as cost and security are partially addressed in the data management and security management components, some functions will be shared among these components.

13.5.2 Data Management in SwinDeW-C

Data management component in SwinDeW-C consists of three basic tasks: data storage, data placement and data replication.

Data Storage: In this component, a dependency based cost-effective data storage strategy is facilitated to store the application data (Yuan et al., 2010). The strategy utilises the data provenance information of the workflow instances. Data provenance in workflows is a kind of important metadata, in which the dependencies between datasets are recorded (Simmhan, Plale, & Gannon, 2005). The dependency depicts the derivation relationship between the application datasets. In cloud workflow systems, after the execution of tasks, some intermediate datasets may be deleted to save the storage cost, but sometimes they have to be regenerated for either reuse or reanalysis (Bose & Frew, 2005). Data provenance records the information of how the datasets have been generated. Furthermore, regeneration of the intermediate datasets from the input data may be very time consuming, and therefore carry a high computation cost. With data provenance information, the regeneration of the demanding dataset may start from some stored intermediated datasets instead. In a cloud workflow system, data provenance is recorded during workflow execution. Taking advantage of data provenance, we can build an Intermediate data Dependency Graph (IDG) based on data provenance (Yuan et al., 2010). All the intermediate datasets once generated in the system, whether stored or deleted, their references are recorded in the IDG. Based on the IDG, we can calculate the generation cost of every dataset in the cloud workflows. By comparing the generation cost and storage cost, the storage strategy can automatically decide whether a dataset should be stored or deleted in the cloud system to reduce the system cost, no matter this dataset is a new dataset, regenerated dataset or stored dataset in the system.

Data Placement: In this component, a data placement strategy is facilitated to place the application data that can reduce the data movement during the workflows' execution. In cloud computing systems, the infrastructure is hidden from users (Weiss, 2007). Hence, for application data, the system will decide where to store them. In the strategy, we initially adapt the k-means clustering algorithm for data placement in cloud workflow systems based on data dependency (Yuan et al., in press). Cloud workflows can be complex, one task might require many datasets for execution; furthermore, one dataset might also be required by many tasks. If some datasets are always used together by many tasks, we say that these datasets are dependant on each other. In our strategy, we try to keep these datasets in one data centre, so that when tasks were scheduled to this data centre, most, if not all, of the data needed are stored locally. Our data placement strategy has two algorithms, one for the build-time stage and one for the run time stage of scientific workflows. In the build-time stage algorithm, we construct a dependency matrix for all the application data, which represents the dependencies between all the datasets. Then we use the BEA algorithm (McCormick, Sehweitzer, & White, 1972) to cluster the matrix and partition it that datasets in every partition are highly dependent upon each other. We distribute the partitions into k data centres, which are initially as the partitions of the k-means algorithm at run time stage. At run time, our clustering algorithm deals with the newly generated data that will be needed by other tasks. For every newly generated dataset, we calculate its dependencies with all k data centres, and move the data to the data centre that has the highest dependency with it.

Data Replication: In this component, a dynamic data replication strategy is facilitated to guarantee data security and the fast data access of the cloud workflow systems. Keeping some replicas of the application data is essential for data security in cloud storage. Static replication can guarantee the data reliability by keeping a fixed number of replicas of the application data, but in a cloud environment, different application data have different usage rates, where the strategy has to be dynamic to replicate the application data based on their usage rates. In large scale workflow applications, many parallel tasks will simultaneously access the same dataset on one data centre. The limitation of computing capacity and bandwidth in that data centre would be a bottleneck for the whole cloud workflow system. If we have several replicas in different data centres, this bottleneck will be eliminated. Hence the data replication will always keep a fix number of copies of all the datasets in different data centres to guarantee reliability and dynamically add new replicas for each dataset to guarantee data availability. Furthermore, the placement of the replicas is based on data dependency, which is the same as the data placement component, and how many replicas a dataset should have is based on the usage rate of this dataset.

13.5.3 Security Management in SwinDeW-C

To address the security issues for the safe running of SwinDeW-C, the security management component is designed. As a type of typical distributed computing system, trust management for SwinDeW-C peers is very important and plays the most important role in security management. Besides, there are some other security issues that we should consider from such as user and data perspective. Specifically, there are three modules in the security management component: trust management, user management and encryption management system.

Trust management: The goal of the trust management module is to manage the relations between one SwinDeW-C peer and its neighbouring peers. For example, to process a workflow instance, a SwinDeW-C peer must cooperate with its neighbouring peers to run this instance. Due to the high level QoS requirements of large scale workflow applications, peer management in SwinDeW-C should be supported by the trust management during workflow run time. The trust management module acts like a consultant. This module can evaluate some tasks and give some advices about the cooperated relation between one peer and other peers for each instance of a specific task. Firstly, peer evaluation makes trust assessment of other neighbouring peers. Secondly, task evaluation makes assessment of re-assignment of the task to other peers. Then the two evaluation scores will be combined by the trust evaluation to reach the conclusion whether this neighbouring peer has adequate trust to take this task. Besides, we design a rule base. For instance, a specific task must not be assigned to one specific neighbouring peer, and this is a simple rule. The rule base is a complement to the previous value-based trust evaluation to fit the real situation (Hess, Holt, Jacobson, & Seamons, 2004).

User management: the user management module is an essential piece in every system. In SwinDeW-C, a user base is a database which stores all user identity and log information that submit service requests. In addition, an authority manager controls the permissions for users to submit some special service requests.

Encryption management System: Given SwinDeW-C peers are located within different geographical local networks, it is important to ensure the data security in the process of data transfer by encryption. In SwinDeW-C, we choose the PGP tool GnuPG (http://www.gnupg.org) to ensure secure commutation.

To conclude, besides the above three new functional components, SwinDeW-C also includes the common cloud functional components such as brokering, pricing, auditing and virtual machine management. Detailed description can be found in (Calheiros, Ranjan, De Rose, & Buyya, 2009) and hence omitted in this paper.

13.6 SwinDeW-C System Prototype

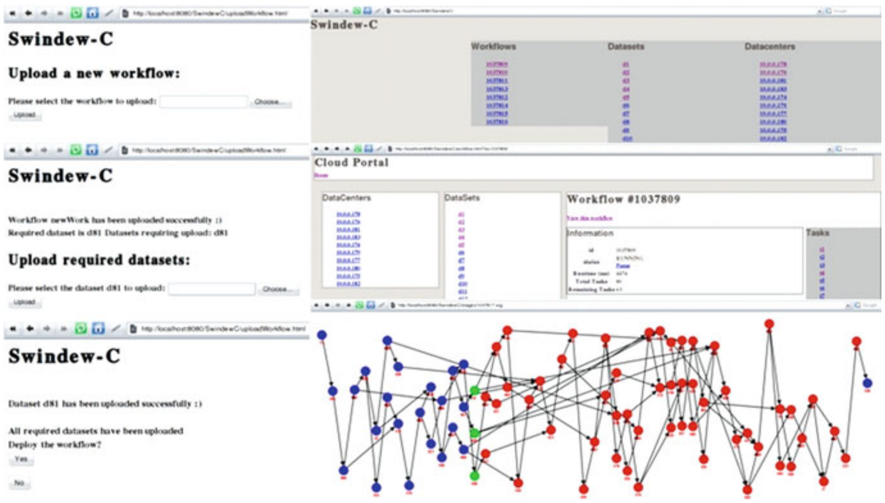
Based on the design discussed above, we have built a primitive prototype of SwinDeW-C. The prototype is developed in Java and currently running on the SwinCloud simulation environment. In SwinDeW-C prototype, we have inherited most of SwinDeW-G functions, and further implemented the new components of SwinDeW-C, so that it can adapt to the cloud computing environment. Furthermore, we have built a Web portal for SwinDeW-C, by which users and system administer can access the cloud resources and manage the applications of SwinDeW-C. The Web portal provides many interfaces to support both system users and administrators with the following tasks, specifically for the system user:

(a) browse the existing datasets that reside in different cloud service providers' data centres;
(b) upload their application data to and download the result data from the cloud storage;
(c) create and deploy workflows to SwinDeW-C system using the modelling tools;
(d) monitor the workflows' execution.

For system administers:

(a) coordinate the workflows' execution by triggering the scheduling strategies;
(b) manage the application datasets by triggering the data placement strategies;
(c) handle the execution exceptions by triggering the workflow adjustment strategies.

Some interfaces of the Web portal are shown in Fig. 13.6.


Swindew-C

Upload a new workflow:

Please select the workflow to upload: Choose... Upload

Swindew-C

Workflow newWork has been uploaded successfully :)
Required dataset is d81 Datasets requiring upload: d81

Upload required datasets:

Please select the dataset d81 to upload: Choose... Upload

Swindew-C

Dataset d81 has been uploaded successfully :)

All required datasets have been uploaded
Deploy the workflow?
Yes
No

Fig. 13.6 SwinDeW-C web portal

13.7 Related Work

Since the research on cloud workflow management systems is at its initial stage, it is difficult to conduct direct comparison between SwinDeW-C with others at present. Most of the current projects are either on the general implementation of cloud computing or focus on some specific aspects such as data management in the cloud. There exists some research into data-intensive applications on the cloud (Moretti, Bulosan, Thain, & Flynn, 2008), such as early experiences like Nimbus (Keahey, Figueiredo, Fortes, Freeman, & Tsugawa, 2008) and Cumulus (Wang, Kunze, & Tao, 2008) projects. Comparing to the distributed computing systems like cluster and grid, a cloud computing system has a cost benefit (Armbrust et al., 2009). Assunção et al. (2009) demonstrate that cloud computing can extend the capacity of clusters with a cost benefit. Using Amazon clouds' cost model and BOINC volunteer computing middleware, the work in (Kondo, Javadi, Malecot, Cappello, & Anderson, 2009) analyses the cost benefit of cloud computing versus grid computing. In terms of the cost benefit, the work by Deelman, Singh, Livny, Berriman, & Good (2008) shows that cloud computing offers a cost-effective solution for data-intensive applications, such as scientific workflows (Hoffa et al., 2008). The work in (Hoffa et al., 2008) explores the use of cloud computing for scientific workflows, focusing on a widely used astronomy application-Montage. The Cloudbus project (http://www.gridbus.org/cloudbus/) being conducted in the CLOUDS Laboratory at the University of Melbourne are working on a new generalised and extensible cloud simulation framework named CloudSim (Calheiros et al., 2009) which can enable seamless modelling, simulation, and experimentation of cloud computing infrastructures and management services.

With the existing projects for many grid workflow systems developed in recent years, it is agreed by many researchers and practitioners that cloud workflow systems might be built on grid computing environments rather than from scratch. For example, the CloudSim toolkit used in the Cloudbus project is implemented by programmatically extending the core functionalities exposed by the GridSim used in the Gridbus project (2010). Therefore, in this chapter, we review some representative grid workflow system and focus on the related features discussed in this paper such as workflow scheduling architecture, QoS, data and security management. Specifically, we investigate Gridbus (2010), Pegasus (2010), Taverna (2010), GrADS (2010), ASKALON (2010), GridAnt (2010), Triana (2010), GridFlow (2010) and Kepler (2010). For the architecture of the workflow scheduling, Pegasus, Taverna, GrADS, and Kepler use a centralised architecture; Gridbus and GridFlow use a hierarchical architecture; ASKALON and Triana use a decentralised architecture. It is believed that centralised schemes produce more efficient schedules and decentralised schemes have better scalabilities, while hierarchical schemes are their compromises. Similar to SwinDeW-G, SwinDeW-C uses a structured decentralised scheme for workflow scheduling. SwinDeW-G aims at using a performance-driven strategy to achieve an overall load balance of the whole system via distributing tasks to least loaded neighbours.

As far as QoS (quality of service) constraints are concerned, most grid workflow systems mentioned above do not support this feature. Gridbus supports QoS constraints including task deadline and cost minimisation, GrADS and GridFlow mainly use estimated application execution time, and ASKALON supports constrains and properties specified by users or predefined. Right now, SwinDeW-C supports QoS constraints based on task deadlines. When it comes to fault tolerance, at the task level, Gridbus, Taverna, ASKALON, Karajan, GridFlow and Kepler use alternate resource; Taverna, ASKALON and Karajan use retry; GrADS uses rescheduling. At the workflow level, rescue workflow is used by ASKALON and Kepler; user-defined exception handling is used by Karajan and Kepler. Pegasus, GridAnt and Triana use their particular strategies respectively. As a comparison, SwinDeW-C uses effective temporal constraint verification for detecting and handling temporal violations.

As for data management, Kepler has its own actor-oriented data modelling method that for large data in the grid environment. It has two Grid actors, called FileFetcher and FileStager, respectively. These actors make use of GridFTP to retrieve files from, or move files to, remote locations on the Grid. Pegasus has developed some data placement algorithms in the grid environment and uses the RLS (Replica Location Service) system as data management at runtime. In Pegasus, data are asynchronously moved to the tasks on demand to reduce the waiting time of the execution and dynamically delete the data that the task no longer needs to reduce the use of storage. In Gridbus, the workflow system has several scheduling algorithms for the data-intensive applications in the grid environment based on a Grid Resource Broker. The algorithms are designed based on different theories (GA, MDP, SCP, Heuristic), to adapt to different use cases. Taverna proposed a new process definition language, Sculf, which could model application data in a dataflow. It considers workflow as a graph of processors, each of which transfers a set of data inputs into a set of data outputs. ASKALON is a workflow system designed for scheduling. It puts the computing overhead and data transfer overhead together to get a value "weight". It dose not discriminate the computing resource and data host. ASKALON also has its own process definition language called AGWL. Triana is a workflow system which is based on a problem-solving environment that enables the data-intensive scientific application to execute. For the grid, it has an independent abstraction middleware layer, called the Grid Application Prototype (GAP), enables users to advertise, discover and communicate with Web and peer-to-peer (p2p) services. Triana also uses the RLS to manage data at runtime. GridFlow is a workflow system which uses an agent-based system for grid resource management. It considers data transfer to computing resources and archive to storage resources as kinds of workflow tasks.

As for security management, Globus uses public key cryptography (also known as asymmetric cryptography) as the basis for its security management, which represents the main stream in the grid security area. Globus uses the certificates encoded in the X.509 certificate format, an established standard data format. These certificates can be shared among public key based software, including commercial Web browsers from Microsoft and Netscape. The International Grid Trust Federation (IGTF) (http://www.igtf.net/) is a third-party grid trust service provider which aims to establish common policies and guidelines between its Policy Management Authorities (PMAs) members. The IGTF does not provide identity assertions but ensures that within the scope of the IGTF charter, the assertions issued by accredited authorities of any of its PMAs member can meet or exceed an authentication profile relevant to the accredited authority. The European GridTrust project (http://www.gridtrust.eu/gridtrust/) is a novel and ambitious project, which provides new security services at the GRID middleware layer. GridTrust is developing a Usage Control Service to monitor resource usage in dynamic Virtual Organisations (VO), enforce usage policies at run-time, and report usage control policy violations. This service brings dynamic usage control to Grid security in traditional, rigid authorisation models. Other services of the security framework include a Grid Security Requirements editor to allow VO owners and users to define security policies; a Secure-Aware Resource Broker Service to help create VOs based on services with compatible security policies; and a sophisticated Reputation Manager Service, to record past behaviour of VO owners and users as reputation credentials.

13.8 Conclusions and Feature Work

Large scale sophisticated workflow applications are commonly seen in both e-Business and e-Science areas. Workflow systems built on high performance computing infrastructures such as cluster, p2p and grid computing are often applied to support the process automation of large scale workflow applications. However, two fundamental requirements including scalable resources and decentralised management have not been well addressed so far. Recently, with the emergence of cloud computing which is a novel computing paradigm that can provide virtually unlimited, easy-scale computing resources, cloud workflow system is a promising new solution and thus deserves systematic investigation. In this chapter, SwinDeW-C, a novel peer-to-peer based cloud workflow system has been presented. SwinDeW-C is not built from scratch but on its predecessor SwinDeW-G (a p2p based grid workflow system). In order to accommodate the cloud computing paradigm and facilitate the management of large scale workflow applications, significant modifications have been made to the previous SwinDeW-G system. Specifically, the original fabric layer of SwinDeW-G is inherited with the extension of external commercial cloud service providers. Meanwhile, the underlying resources are virtualised at the unified resource layer; functional components including QoS management, data management and security management are added or enhanced at the platform layer to support the management of large scale workflow applications; the user interface is modified to support Internet (Web browser) based access.

This chapter has described the system architecture of SwinDeW-C and its new features for managing instance and data/computation intensive workflow applications. The SwinDeW-C prototype system has been demonstrated but still under further development. In the future, more functional components will be designed and deployed to enhance the capability of SwinDeW-C. Meanwhile, comparison will also be conducted between SwinDeW-C and other workflow systems based on the statistics of performance measurements such as success rate, temporal violation rate, system throughput and others.

Acknowledgment This work is partially supported by Australian Research Council under Linkage Project LP0990393.

References

Ardagna, D., & Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering, 33*(6), 369–384.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., et al. (2009). *Above the clouds: A Berkeley view of cloud computing* (Tech. Rep., University of California, Berkeley).

de Assuncao, M. D., di Costanzo, A., & Buyya, R. (2009). Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, Garching, Germany*, 1–10.

Bhargav-spantzel, A., Squicciarini, A. C., & Bertino, E. (2007). Trust negotiation in identity management. *IEEE Security & Privacy, 5*(2), 55–63.

Bose, R., & Frew, J. (2005). Lineage retrieval for scientific data processing: A survey. *ACM Computing Surveys, 37*(1), 1–28.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599–616.

Calheiros, R. N., Ranjan, R., De Rose, C. A. F., & Buyya, R. (2009). *CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services* (Tech. Rep., Grid

Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering,The University of Melbourne).

Chen, J., & Yang, Y. (2007). Multiple states based temporal consistency for dynamic verification of fixed-time constraints in grid workflow systems. *Concurrency and Computation: Practice and Experience (Wiley), 19*(7), 965–982.

Chen, J., & Yang, Y. (2008). A taxonomy of grid workflow verification and validation. *Concurrency and Computation: Practice and Experience, 20*(4), 347–360.

Chen, J., & Yang, Y. (2010). Temporal dependency based checkpoint selection for dynamic verification of temporal constraints in scientific workflow systems. *ACM Transactions on Software Engineering and Methodology*, to appear. Retrieved 1st February 2010, from http://www.swinflow.org/papers/TOSEM.pdf.

Chen, J., & Yang, Y. (2007). Adaptive selection of necessary and sufficient checkpoints for dynamic verification of temporal constraints in grid workflow systems. *ACM Transactions on Autonomous and Adaptive Systems, 2*(2), Article 6.

Chen, J., & Yang, Y. (2008). Temporal dependency based checkpoint selection for dynamic verification of fixed-time constraints in grid workflow systems. *Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany*, 141–150.

Chervenak, A., Deelman, E., Livny, M., Su, M. H., Schuler, R., Bharathi, S., et al. (2007). Data placement for scientific applications in distributed environments. *Proceedings of the 8th Grid Computing Conference,* 267–274.

Deelman, E., Gannon, D., Shields, M., & Taylor, I. (2008). Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems, 25*(6), 528–540.

Deelman, E., & Chervenak, A. (2008). Data management challenges of data-intensive scientific workflows. *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, 687–692.

Deelman, E., Singh, G., Livny, M., Berriman, B., & Good, J. (2008). The cost of doing science on the cloud: The montage example. *Proceedings of the ACM/IEEE Conference on Supercomputing, Austin, TX*, 1–12.

Erl, T. (2008). *SOA: Principles of service design*. Upper Saddle River, NJ: Prentice Hall.

Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud computing and grid computing 360-degree compared. *Proceedings of the Grid Computing Environments Workshop, 2008, GCE′08,* 1–10.

Hadoop (2009). Retrieved 1st September 2009 from http://hadoop.apache.org/.

Hess, A., Holt, J., Jacobson, J., & Seamons, K. E. (2004). Content-triggered trust negotiation. *ACM Transactions on Information and System Security, 7*(3), 428–456.

Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., et al. (2008). On the use of cloud computing for scientific workflows. *Proceedings of the 4th IEEE International Conference on e-Science,* 640–645.

Keahey, K., Figueiredo, R., Fortes, J., Freeman, T., & Tsugawa, M. (2008). Science clouds: Early experiences in cloud computing for scientific applications. *Proceedings of the First Workshop on Cloud Computing and its Applications (CCA′08),* 1–6.

Kondo, D., Javadi, B., Malecot, P., Cappello, F., & Anderson, D. P. (2009). Cost-benefit analysis of cloud computing versus desktop grids. *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing, IPDPS′09,* 1–12.

Lin, C., Varadharajan, V., Wang, Y., & Pruthi, V.t. (2004). Enhancing grid security with trust management. *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC04)*, 303–310.

Liu, K., Chen, J. J., Yang, Y., & Jin, H. (2008). A throughput maximization strategy for scheduling transaction-intensive workflows on SwinDeW-G. *Concurrency and Computation: Practice and Experience, 20*(15), 1807–1820.

Liu, K., Jin, H., Chen, J., Liu, X., Yuan, D., & Yang, Y. (2010). A compromised-time-cost scheduling algorithm in SwinDeW-C for instance-intensive cost-constrained workflows on cloud computing platform. *International Journal of High Performance Computing Applications*.

Liu, X., Chen, J., Liu, K., & Yang, Y. (2008). Forecasting duration intervals of scientific workflow activities based on time-series patterns. *Proceedings of the 4th IEEE International Conference on e-Science (e-Science08), Indianapolis, IN, USA,* 23–30.

Liu, X., Chen, J., Wu, Z., Ni, Z., Yuan, D., & Yang, Y. (2010). Handling recoverable temporal violations in scientific workflow systems: A workflow rescheduling based strategy. *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid10), Melbourne, Australia*.

Liu, X., Chen, J., & Yang, Y. (September 2008). A probabilistic strategy for setting temporal constraints in scientific workflows. *Proceedings of the 6th International Conference on Business Process Management (BPM08), Lecture Notes in Computer Science, Vol. 5240, Milan, Italy,* 180–195.

McCormick, W. T., Sehweitzer, P. J., & White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research, 20*, 993–1009.

Moore, M. (2004). An accurate parallel genetic algorithm to schedule tasks on a cluster. *Parallel Computing, 30*, 567–583.

Moretti, C., Bulosan, J., Thain, D., & Flynn, P. J. (2008). All-Pairs: An abstraction for data-intensive cloud computing. *Proceedings of the IEEE International Parallel and Distributed Processing Symposium, IPDPS′08,* 1–11.

Askalon Project (2010). Retrieved 1st February 2010, from http://www.dps.uibk.ac.at/projects/askalon.

GrADS Project (2010). Retrieved 1st February 2010, from http://www.iges.org/grads/.

GridBus Project (2010). Retrieved 1st February 2010, from http://www.gridbus.org.

Kepler Project (2010). Retrieved 1st February 2010, from http://kepler-project.org/.

Pegasus Project (2010). Retrieved 1st February 2010, from http://pegasus.isi.edu/.

Taverna Project (2010). Retrieved 1st February 2010, from http://www.mygrid.org.uk/tools/taverna/.

Triana Project (2010). Retrieved 1st February 2010, from http://www.trianacode.org/.

Raghavan, B., Ramabhadran, S., Yocum, K., & Snoeren, A. C. (2007). Cloud control with distributed rate limiting. *Proceedings of the 2007 ACM SIGCOMM, Kyoto, Japan,* 337–348.

SECES (May 2008). *Proceedings of the 1st International Workshop on Software Engineering for Computational Science and Engineering, in conjuction with the 30th International Conference on Software Engineering (ICSE2008), Leipzig, Germany*.

Simmhan, Y. L., Plale, B., & Gannon, D. (2005). A survey of data provenance in e-Science. *SIGMOD Rec. 34*(3), 31–36.

VMware (2009). Retrieved 1st September 2009, from http://www.vmware.com/.

Wang, L. Z., Kunze, M., & Tao, J. (2008). Performance evaluation of virtual machine-based grid workflow system. http://doi.wiley.com/10.1002/cpe.1328, 1759–1771.

Wang, L. Z., Jie, W., & Chen, J. (2009). *Grid computing: Infrastructure, service, and applications*. Boca Raton, FL: CRC Press, Talyor & Francis Group.

Weiss, A. (2007). Computing in the cloud. *ACM Networker, 11*(4), 18–25.

Winsborough, W. H., & Li, N. H. (2006). Safety in automated trust negotiation. *ACM Transactions on Information and System Security, 9*(3), 352–390.

Yang, Y., Liu, K., Chen, J., Lignier, J., & Jin, H. (December 2007). Peer-to-peer based grid workflow runtime environment of swinDeW-G. *Proceedings of the 3rd International Conference on e-Science and Grid Computing (e-Science07), Bangalore, India,* 51–58.

Yang, Y., Liu, K., Chen, J., Liu, X., Yuan, D., & Jin, H. (December 2008). An algorithm in swinDeW-C for scheduling transaction-intensive cost-constrained cloud workflows. *Proceedings of the 4th IEEE International Conference on e-Science (e-Science08), Indianapolis, IN, USA,* 374–375.

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

Yuan, D., Yang, Y., Liu, X., & Chen, J. (2010). A cost-effective strategy for intermediate data storage in scientific cloud workflow systems. *Proceedings of the 24th IEEE International*

Parallel & Distributed Processing Symposium, Atlanta, GA, USA, to appear. Retrieved 1st February 2010, from http://www.ict.swin.edu.au/personal/yyang/papers/IPDPS10-IntermediateData.pdf.

Yuan, D., Yang, Y., Liu, X., & Chen, J. A data placement strategy in cloud scientific workflows. *Future Generation Computer Systems,* in press. http://dx.doi.org/10.1016/j.future.2010.02.004.