Managing Large Numbers of Business Processes with Cloud Workflow Systems

Xiao Liu¹, Yun Yang¹, Dahai Cao¹, Dong Yuan¹, Jinjun Chen¹
¹Faculty of Information and Communication Technologies
Swinburne University of Technology, Melbourne, Australia
²Faculty of Engineering and Information Technology
University of Technology Sydney, Sydney, Australia

{xliu, yyang, dcao, dyuan}@swin.edu.au, jijun.chen@uts.edu.au

Abstract
With the emergence of cloud computing which can deliver on-demand high-performance computing resources over the Internet, cloud workflow systems offer a competitive software solution for managing large numbers of business processes. In this paper, we first analyse the basic system requirements through a motivating example, and then, the general design of a cloud workflow system is proposed with the focus on its system architecture, functionalities and QoS (quality of service) management. Afterwards, the system implementation of a peer-to-peer based prototype cloud workflow system is demonstrated to verify our design. Finally, experimental results show that with the dynamic resource provisioning, conventional violation handling strategies such as workflow local rescheduling can ensure the on-time completion of large numbers of business processes in a more cost-effective way.

Keywords: Business Process Management, Workflow System, Cloud Computing, Cloud Workflow System

1 Introduction

With the rapid development of e-business and e-government in the global economy, both enterprises and government agencies are facing large numbers of concurrent business processes from the private and public sectors [1, 21]. For example, a federal government taxation office receive millions of tax declaration requests at the beginning and end of the tax return period each year; a banking enterprise often needs to process millions of transactions including cheques everyday; and an insurance company may need to process over thousands of claims on a daily basis which may peak by a factor of tens or hundreds when some natural disasters happen, e.g. the Melbourne hailstorm in March 2010 results in 79,000 claims which worth A$491 million¹. Failure of completing these process instances in time is not acceptable and will often results in significant loss. For example, the Australian federal government taxation office has to pay a large amount of interest to tax payers for the delay; the time delays in stock exchange may result in significant loss to both sellers and buyers in the stock market.

For time constrained business processes, software performance (e.g. response time and throughput), as one of the basic dimensions of software quality, is very important [24]. To ensure satisfactory performance, enterprises and government agencies often need to invest a huge amount of money on their self-owned and self-maintained IT (Information Technology) infrastructures which are normally designed to have the capability to meet either the maximum or at least the average needs of computing resources. However, for the option to meet the average needs, the software performance during peak time can be significantly deteriorated. As for the option to meet the maximum needs, since the number of process instances during peak time can often be much larger than the average, such a design will often result in largely idle of computing resources, which means a huge waste of financial investment and energy consumption. In general, the running of larger numbers of business processes usually require powerful, on-demand and elastic computing resources. Specifically, the basic system requirements for business software can include: 1) scalable computing resource provision; 2) elastic computing resource delivery; 3) efficient process management and 4) effective QoS (quality of service) monitoring and control. Detailed analysis will be presented in Section 2.

Cloud computing, an exciting and promising new computing paradigm, can play an important role in this regard. In late 2007, the concept of cloud computing was proposed. Cloud computing, nowadays widely considered as the “next generation” of IT, is a new paradigm offering virtually unlimited, cheap, readily available, "utility type" scalable computing resources as services via the Internet [4, 6]. As very high network bandwidth becomes available, it is possible to envisage all the resources needed to accomplish IT functions as residing on the Internet rather than physically existing on the clients’ premises. With effective facilitation of cloud computing, many sophisticated software applications can be further advanced to stretch their limits and yet with reduced running costs and energy consumption. The advantages of cloud computing, especially its utility computing and SaaS (software as a service), enable entirely new innovations to the design and development of software applications [1, 18]. It is generally agreed among many researchers and practitioners that cloud applications are the future trend for business software applications since utility computing can provide unlimited on-demand and elastic computing...
Section 3 proposes the design of a novel cloud workflow system. Section 4 describes the prototype. Section 5 demonstrates the evaluation results. Section 6 addresses requirements. Section 7 introduces some related work. Finally, Section 8 concludes and presents the future work.

The remainder of this paper is organised as follows. Section 2 presents a motivating example and system requirements. Section 3 introduces a novel cloud workflow system with the focus on its system architecture, basic functionalities and QoS management. Afterwards, Section 4 employs a securities exchange business process as a motivating example to analyse the system requirements. Section 5 demonstrates the evaluation results. Section 6 addresses requirements. Section 7 introduces some related work. Finally, Section 8 concludes and points out the future work.

The motivation of this project stems from the needs of a securities exchange business process. Due to the space limit, we only introduce the main facts here while leaving details in [15].

Workflow systems, with the benefits of efficient and flexible process modelling and process automation, have been widely used for managing business processes, or cloud workflow systems for short, can be a suitable solution for managing large numbers of time-constrained business processes.

Given the recent and rapid growth of cloud computing, we can envisage that cloud computing based workflow systems can be deployed as a part of cloud workflow systems. The design of a cloud workflow system is demonstrated. Finally, simulation experiments evaluate the effectiveness of the system implementation of our SwinDeW-C prototype in the running of large numbers of simulation experiments.

21. Motivating Example

Securities exchange in the stock market is a typical instance-intensive business process which involves a large number of transactions between different organisations. The example illustrated in Figure 1 is a securities exchange process for the Chinese stock market. There are more than one hundred securities corporations and their branches, and the third level of the procedure is between Clearing Corporation and securities corporations, the second one is between branches and clients, and the third one is between branches and clients. For example, in the Shanghai A-Shares Stock Market, the size of the output file is about 50M with tens of millions of transactions and the duration of the procedure is about 1.5 hours. All the clearing results of the third level should be completed by securities corporations in the second level of clearing. The clearing result of each level should match with each other.

The input of the clearing process is the money due between branches and clients. The output of the clearing process is the money due between branches and clients. The clearing result of each level should match with each other.

To illustrate the system, we employ a securities exchange business process as a motivating example. This business process has six major stages (sub-processes) in the securities exchange process. Due to the space limit, we only introduce the main facts here while leaving details in [15].

Fig. 1. A Typical Securities Exchange Business Process

There are more than one hundred securities corporations in the stock market. The size of the output file is about 50G with tens of millions of transactions and the duration of the procedure is about 1.5 hours. All the clearing results of the third level should be completed by securities corporations in the second level of clearing. The clearing result of each level should match with each other.

The output of the clearing process is the money due between branches and clients. The output of the clearing process is the money due between branches and clients. The clearing result of each level should match with each other.

The output of the clearing process is the money due between branches and clients. The output of the clearing process is the money due between branches and clients. The clearing result of each level should match with each other.
transferring details for each client who made deals during the day. It is a 20M size data file with about 200k transactions and it should be sent to the designated banks. The designated banks check the bills in about 30 minutes at both the client level and the branch level to ensure each entity has enough money to pay for the shares. The money is then transferred between banks and clients, and between banks and the Clearing Corporation, which takes around 50 minutes.

(6) The last stage is “produce clearing files” (Step 13 to Step 14). Both securities corporations and designated banks should produce the clearing files for the Clearing Corporation. The balance of all the capital transferred should be zero at the Clearing Corporation level. Otherwise, exception handling should be conducted with manual intervention. The whole securities exchange workflow is ended afterwards.

To summarise, the securities exchange is a typical business process which involves many parallel process instances with strict performance requirements such as fast response time and high throughput. Failures of meeting these performance requirements could result in serious financial loss to both clients and securities corporations.

2.2 System Requirements

Based on the above motivating example, we can identify the following four basic system requirements for managing large numbers of business processes.

1) Scalable computing resource provision. The running of large numbers of business processes requires powerful computing resources. To deal with millions of concurrent requests (e.g. the first and second stage) and processing these transactions after trading hours (e.g. the fifth and sixth stage), computing resources with high processing power and fast IO speed is required. Only in such a case, the satisfactory performance of the system such as short response time for each request and high throughput for processing massive transactions can be achieved.

2) Elastic computing resource delivery. The amount of computing resources required at peak-time (e.g. over millions of requests per second at the beginning and the end of the trading hours) is much higher than the average (e.g. thousands of requests per second in off-peak time). To ensure satisfactory system performance, huge capital investment is often spent on the IT infrastructure to meet the resource requirement during peak-time. However, this will result in large idle of computing resources and a huge waste of energy. Therefore, the elasticity in resource delivery, i.e. the resource pool can easily increase its size when necessary and decrease immediately after use, is very important for reducing the system running cost.

3) Efficient process management. Process automation is the key to improve the performance of running business processes. Besides, in the real world, the specific process structures may be subject to changes. For example, the introduction of new products and the practice of new market regulations may result in some process changes from the third stage to fifth stage. Therefore, the software system needs to have some flexibility for business process change, as well as some new functional and non-functional (quality) requirements coming with it [19]. To this end, efficient process management (e.g. process modelling, process redesign, service selection, and task coordination) plays a significant role in process automation.

4) Effective QoS monitoring and control. Since a business software system needs to deal with massive processes with flexible business requirements, how to ensure that all the processes are running with satisfactory QoS requirements is a challenge. For example, if the response time for the second stage (fit and make deal) in the securities exchange process is over the time constraints, e.g. 5 minutes, it will probably result in the failure of the client’s requests, and thus will bring substantial loss to both the client and the securities corporation. Therefore, effective QoS monitoring and control is essential. Specifically, QoS monitoring is to constantly observe the system execution state and detect QoS violations while QoS control is to tackle detected QoS violations so as to ensure the specified QoS constraints can be satisfied.

3 The General Design of a Cloud Workflow System

Given the four basic system requirements discussed in Section 2.2, in this paper, we propose that a cloud workflow system is a competitive solution for managing large numbers of business processes. Naturally, a cloud workflow system is running in a scalable and elastic cloud computing environment (satisfying the first and second system requirements), and it is generally designed to have the basic system components for process modelling, resource management, runtime workflow monitoring and control (satisfying the third and fourth system requirements). Our strategy is to start with prototyping a core cloud workflow system, and then extend its structure and capabilities to meet the requirements for managing large numbers of business processes. In this section, we focus on the general system architecture, functionalities and QoS management, while leaving the details in the system implementation to be demonstrated in Section 4.

3.1 System Architecture

As depicted in Figure 2, the general cloud system architecture consists of four basic layers from the top to bottom: application layer, platform layer, unified resource layer, and fabric layer.

As shown in Figure 2, the general cloud workflow architecture can be a mapping of the general cloud system architecture [10]. Specifically, the application layer consists of cloud workflows (workflow applications for real-world business processes), the platform layer is the cloud workflow system which provides a development and deployment platform for cloud workflows. All the system functionalities of a cloud workflow system such as workflow management, cloud resource management and QoS management are included. The application layer and the platform layer are usually self-maintained by the business organisation1. The unified resource layer consists of both software services and hardware services that are required for the running of cloud workflows. Specifically, SaaS (software as a service) can provide massive number of software capabilities for processing different business

---

1 A cloud workflow system can be encapsulated as a platform service, i.e. PaaS (platform as a service). In such a case, the platform layer is maintained by external cloud service providers.
tasks, while IaaS (infrastructure as a service) can provision on-demand and elastic computing power to meet the resource requirements for processing business activities. In practice, software and hardware services can also be integrated together and encapsulated to be delivered as VMs (virtual machines). The fabric layer is composed of low level hardware resources such as computing, storage and network resources. The unified layer and fabric layer are often maintained by external cloud service providers.

3.2 System Functionalities

A cloud workflow system is the combination of workflow system and cloud services. The workflow reference model [2] suggested by WfMC (workflow management coalition, http://www.wfmc.org/) defines the general components and interfaces of a workflow system. Therefore, instead of building from the scratch, we can design the basic system functionalities of a cloud workflow system by extending the workflow reference model with functionalities required for the integration of cloud services, such as cloud resource management and QoS management components. Given its critical importance in cloud workflow systems, the QoS management components will be introduced separately in Section 3.3.

3.3 QoS Management

Due to the dynamic nature of cloud computing, effective QoS management in a cloud workflow system is very important. Specifically, for managing large numbers of business processes, service performance (e.g. short response time for every client request and high throughput for processing massive concurrent client requests), service reliability (e.g. minimal failure rate for activity execution) and service security (e.g. stringent policies for the lifecycle protection of client data in its storage, transfer and destroy), are among the most important QoS dimensions which should be given higher priority in cloud workflow QoS management [17-19, 24]. Meanwhile, since a cloud workflow instance needs to undergo several stages before its completion, a lifecycle QoS management needs to be established.

In general, a lifecycle QoS management consists of four basic steps, viz. QoS requirement specification, QoS-aware service selection, QoS monitoring and QoS violation handling [16]. As depicted in Figure 3, as part of workflow built-time functionalities, QoS requirement specification and QoS-aware service selection are mainly interacted with the workflow modelling tool. The QoS requirement specification component would generate the QoS constraints, which are part of the workflow specification and the basic criteria for QoS-aware service selection. The QoS-aware service selection component will return the available (best and backup) software services satisfying the QoS constraints, through the cloud resource brokers. After the workflow specifications are submitted to the workflow enactment services, workflow instances can be executed by invoking software services which are managed by the tool agents. During workflow runtime, the workflow execution state will be constantly observed by the QoS monitoring component. The workflow execution state can be displayed to the client and the cloud system administrator by a watch list which contains runtime information such as time submitted, time finished, percentage of completion, service status and many other real-time and possible statistic data. When the QoS violations are detected, alert messages would be sent to invoke the QoS violation handling component. This

The fabric layer can also be a virtual collection of local computing infrastructure (i.e. private cloud) and the commercial computing infrastructure (i.e. public cloud), i.e. hybrid cloud.

---

Fig. 2. Cloud Workflow Architecture

Fig. 3. System Functionalities and QoS Management in Cloud Workflow System
component will analyse the workflow execution state and the QoS requirement specification to decide further actions. Generally speaking, for QoS violation handing, firstly, we should try to minimise the existing loss through compensation, and secondly, we should prevent similar violations from happening in the subsequent workflow as much as possible [16, 19]. It is evident that due to the complexity of instance intensive business processes and the dynamic nature of cloud computing, satisfactory service quality of a cloud workflow system can only be achieved through such a lifecycle QoS management.

4 System Implementation: A Prototype P2P based Cloud Workflow System

Based on the general design of a cloud workflow system presented in Section 3, this section demonstrates the implementation of a prototype cloud workflow system. SwinDeW-C (Swinburne Decentralised Workflow for Cloud) [17] is running in SwinCloud which is built on the computing facilities in Swinburne University of Technology and takes advantage of the existing SwinGrid infrastructure, a grid computing test bed [17].

Fig. 4. SwinCloud Infrastructure

The migration of SwinGrid to SwinCloud is achieved in two steps. First, VMWare (http://www.vmware.com/) is installed in existing SwinGrid nodes so that they can offer unified computing and storage resources. Second, we set up data centres on the groups of SwinGrid nodes which can host different cloud services. In each data centre, Hadoop (http://hadoop.apache.org/) is installed to facilitate Map-Reduce computing paradigm and distributed data management. Different from SwinGrid, SwinCloud is a virtualised computing environment, where cloud services run on unified resources. By dynamically acquiring computing and storage units from VMWare, cloud services can flexibly scale up and down according to system requirements. SwinDeW-C inherits many features of its ancestor SwinDeW-G [23] but with significant modifications in its functionalities to accommodate the cloud computing paradigm and the system requirements for managing instance intensive business processes. Figure 4 depicts the SwinCloud infrastructure. More details about the system environment can be found in [17].

4.1 Architecture of SwinDeW-C

In order to overcome the problems of centralised management such as performance bottleneck, lack of scalability and single point of failure, SwinDeW-C is designed in a decentralised, or more specifically, structured peer-to-peer fashion where all the workflow data and control flows are transferred among SwinDeW-C peers. Such a design can greatly enhance the performance and reliability of a workflow system in managing large numbers of workflows since all the system functionalities are implemented with distributed SwinDeW-C peers which will be introduced in Section 4.2.

The architecture of SwinDeW-C is depicted in Figure 5 [17]. Clients can access SwinDeW-C Web portal via any electronic devices such as PC, laptop 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 client-side programs, SwinDeW-C Web portal can greatly improve the usability. Here, we describe the lifecycle of an abstract workflow application through its modelling stage, instantiation stage and execution stage to illustrate the system architecture.

At the modelling stage, given the cloud workflow modelling tool provided by the Web portal on the application layer, workflow applications are modelled by clients as cloud workflow specifications (consisting of such as task definitions, process structures and QoS constraints). After workflow specifications are created, they will be submitted to 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 its computing power can scale dynamically. As for the SwinDeW-C coordinator peers, they are super nodes which are equipped with additional management functionalities.

At the instantiation stage, the cloud workflow specification is submitted to one of the SwinDeW-C coordinator peers. A coordinator peer conducts an evaluation process on the submitted cloud workflow instance to determine whether it can be accepted or not given the workflow specification, the available cloud resources, and the resource prices. It is generally assumed that functional requirements can normally be satisfied given the unlimited scalable computing resources and software services in the cloud. In the case where clients need to run their own special programs, they can upload...
them through the Web portal and these programs can be automatically deployed in the cloud data centre. However, the QoS requirements may not be always satisfied. Due to the natural limitations of cloud service quality and the unacceptable offers on budgets, a negotiation process between the client and the cloud workflow system may be conducted. The final negotiation result can be either the compromised QoS requirements or a failed submission of the cloud workflow instance. If it is successful, the workflow activities will be assigned to suitable SwinDeW-C peers through p2p based communication. The peer management such as peer join, peer leave and peer search, as well as the p2p based workflow execution mechanism, are the same as in SwinDeW-G system environment which are detailed in [23]. After all the workflow activities are successfully allocated (i.e. confirmation messages are sent back to the coordinator peer from all the allocated peers), a cloud workflow instance is successfully instantiated.

Finally, at the execution stage, each workflow activity is executed by a SwinDeW-C peer. Clients can get access to the final results as well as the running information of their submitted workflow instances through the SwinDeW-C Web portal. Each SwinDeW-C peer utilises the computing power provided by its virtual machine which can easily scale up and down according to the requests of workflow activities. As can be seen in Figure 4, 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.

4.2 Functionalities of SwinDeW-C Peers

The architecture and functionalities of SwinDeW-C peers are depicted in Figure 6. As mentioned above, the system functionalities of SwinDeW-C are distributed to its peers. SwinDeW-C is developed based on SwinDeW-G, where a SwinDeW-C peer has inherited most of the functionalities in a SwinDeW-G peer, including the components of task management, flow management, data management, and the group management [23]. Hence, a SwinDeW-G peer plays as the core of a SwinDeW-C peer. A SwinDeW-G peer is developed by Java with the Globus toolkit (http://www.globus.org/toolkit/) and JXTA (http://www.sun.com/software/jxta/).

Fig. 6. Architecture and Functionalities of SwinDeW-C Peers

To accommodate cloud resources and the system requirements for instance intensive business processes, a coordinating peer is introduced to the SwinDeW-C system and significant modifications also have been made in other normal peers. Besides the those functionalities inherited from SwinDeW-G peers, some new cloud resource management components are developed for SwinDeW-C peers based on the APIs offered by VMWare and Hadoop, and some existing components such as QoS management are further enhanced. Specifically:

First, a resource provisioning component is added to every SwinDeW-C peer. In SwinDeW-C, to meet the scalable and elastic resource requirement, a SwinDeW-C peer can scale up or down with more or fewer computing units. Meanwhile, through the SwinDeW-C coordinate peer, it can also scale out or in if necessary, i.e. to request the distribution of workflow activities to more or fewer SwinDeW-C peers in the same group. This is mainly realised through the APIs of VMWare management tools.

Second, the resource pricing and auditing components are equipped in SwinDeW-C coordinator peers. Since different cloud service providers may offer different prices, during the instantiation stage, a coordinator peer needs to have the pricing component to negotiate the prices with external service providers and set its own offered prices to its clients. Meanwhile, since the cloud workflow system needs to pay for the usage of external cloud resources, at the execution stage, an auditing component is required to record and audit the usage of cloud resources. These functionalities are mainly realised through the APIs of resource brokers and the external service provider’s monitoring services such as the Amazon CloudWatch (http://aws.amazon.com/cloudwatch/).

Third, QoS management components in a SwinDeW-C coordinator peer have been extended to support for multiple QoS dimensions, viz. performance, reliability and security, which are regarded as three major QoS dimensions for running business processes. Specifically, performance management is mainly for the response time and throughput of business processes, reliability management is mainly for the reliability and cost of data storage services, and security management is mainly for transaction security and the protection of client privacy data. According to the lifecycle QoS management introduced in Section 3.3, these components need to interact with many other system build-time and runtime functional components which are implemented as parts of the SwinDeW-C coordinator peers.

4.3 QoS Management in SwinDeW-C

The major workflow QoS dimensions supported in SwinDeW-C are performance, reliability and security. Details can be found in [17]. In this section, we take the performance management component as an example. In our previous work, the performance management only focuses on the response time of a single workflow instance. In this paper, for the running of large numbers of time-constrained business processes, we also focus on the throughput of the cloud workflow system. Specifically, there are four basic tasks for delivering lifecycle performance management in SwinDeW-C:

Temporal Constraint Setting: In SwinDeW-C, temporal constraints consist of two types, viz. constraints for workflow response time, and constraints for system throughput. A probabilistic strategy is designed for setting constraints for workflow response time in SwinDeW-C.
Specifically, one overall deadline and several milestones are assigned based on the negotiation result between clients and cloud workflow service providers. Afterwards, fine-grained constraints for individual workflow activities can be derived automatically [13]. Besides the constraints for the workflow response time, we also need to setup some throughput constraints to monitor system throughputs along the workflow execution. Currently, we adopt a setting strategy where throughput constraints are defined as the percentage of completion and assigned at pre-defined time points with fixed equal time intervals. For example, given a set of 1,000 business processes (each with 10 activities) start at 10:00am and have an overall deadline by 12:00pm, the throughput constraints can be specified as at 10:30am, 25% of the total business processes should be finished, and 50% of them should be finished by 11:00pm, and so forth. Note that here 50% completion does not necessarily mean a total of 500 processes should be finished but rather mean a total of 5,000 activities are completed since significant delays often occur in the running of some business processes.

**Temporal-Aware Service Selection:** Given the fine-grained constraints for response time assigned in the first step, a set of candidate services which satisfy the constraints can be searched by the cloud resource broker from the cloud [6, 24]. Meanwhile, since different service providers may offer different prices, and there are often other QoS constraints such as reliability and security to be considered at the same time, a ranking strategy is designed to determine the best candidate for runtime execution. Furthermore, considering to the dynamic nature of cloud computing as well as the performance and reliability requirements for managing large numbers of business processes, a set of backup/redundant services should also be reserved during service selection. In fact, many cloud service providers such as Amazon provides special discount price for reserved instances, which can be used as a source of reliable standby capacity.

**Temporal Checkpoint Selection and Verification:** During workflow runtime, the workflow execution state should be monitored against the violation of temporal constraints. Temporal verification is to check the temporal correctness of workflow execution, i.e. to detect temporal violations of workflow response time and system throughput. The verification of workflow response time constraints is conducted at the activity level and the verification of system throughput is conducted at the workflow level. In SwinDeW-C, a minimum time redundancy based checkpoint selection strategy [8] is employed which selects only necessary and sufficient checkpoints to detect the violations of workflow response time. Here, necessity means only the activity points with temporal violations are selected, and sufficiency means there are no omitted ones, hence the strategy is highly efficient for the monitoring of large numbers of workflow activities. As for throughput verification, it is conducted at the pre-defined time points which are specified at the constraint setting stage, hence in a static fashion.

**Temporal Violation Handling:** After a temporal violation is detected, violation handling strategies are required to recover the error states such as the larger response time and lower system throughput. In SwinDeW-C, to decrease the overall violation handling cost on workflow response time, a three-level temporal violation handling strategy is designed. Specifically, for minor temporal violations, the TDA (time deficit allocation) strategy [7] is employed which can remove the current time deficits by borrowing the time redundancy of the subsequent activities. For moderate temporal violations, the ACOWR (ant colony optimisation based two stage workflow local rescheduling) strategy [12] is employed which can decrease the execution time of the subsequent workflow activities through the optimisation of resource allocation. As for major temporal violations, the combined strategy of TDA and ACOWR is employed which conducts TDA in the first step and followed by several iterations of ACOWR until the temporal violation is recovered. Based on such a design, the overall violation handling cost on workflow response time can be significantly reduced compared with a single expensive exception handing strategy [12]. However, since it has been well observed that short response time does not necessarily guarantee an overall high system throughput, we still need some violation handling strategies to recover throughput violations. Meanwhile, since most violation handling strategies such as TDA and ACOWR target the reduction of the response time of a single workflow instance, it may not be directly effective for the increase of system throughput. One of the options is to conduct these strategies repeatedly for many business processes so that the system throughput can be increase by the reduction of the average workflow response time. But this option is evidently very expensive. Currently, in SwinDeW-C, we adopt a simple elastic resource provision strategy which is to dynamically provision the reserved resources when throughput violations are detected, and release these resources when the system throughput is back to normal. In such a case, since many awaiting workflow activities will be processed immediately, the system throughput can be increased in a short period time. Details will be further illustrated in our experiments demonstrated in Section 5.

5 Evaluation

Based on the SwinDeW-C prototype system, the general design of a cloud workflow system proposed in Section 3 is successfully implemented to satisfy the basic system requirements discussed in Section 2.2. Specifically, the four-layer cloud workflow system architecture and the structured p2p based decentralised workflow management ensures efficient provision of scalable and elastic cloud computing resources for running instance intensive business processes (for the first and second system requirements); the visual modelling tool for workflow specification, the workflow enactment service and the application provision service, can effectively support the efficient process management (for the third system requirement); and the QoS management components can facilitate the effective QoS monitoring and control (for the fourth system requirement).

At the moment, to evaluate and improve its performance, a number of test cases with simulated large scale instance intensive workflows are designed and being

http://aws.amazon.com/ec2/reserved-instances/
tested in SwinDeW-C, including the securities exchange workflow and some large scale high performance applications with larger number of sub-processes such as a weather forecast workflow [13] and a pulsar searching workflow in Astrophysics [14].

On the setting of business processes and resources: In order to evaluate the performance of SwinDeW-C, we have simulated a large number of business processes running in parallel. The total number of business processes is 10K, which is similar to the total number of securities corporation branches nation wide. In our experiments, we focus on the offline processing part, i.e. from the step 4 to step 14, where all the daytime transaction data are to be batch-processed over night at the stock depository and clearing corporation. For the ease of simulation, we assume that each process has 20 activities to represent the basic batch processing steps, and correspondingly there are 20 types of cloud services in charge of running these activities. The total number of cloud service instances is set as 200, i.e. 10 instances for each type of service. Additionally, there is 1 reserved instance for each type of service to handle temporal violations. As for the resource price, we adopt the Amazon EC2 price model as a reference (http://aws.amazon.com/ec2/pricing/). The price for the primary services (similar to the EC2 Quadraple Extra Large Hi-Memory On-Demand Instances) is $2.00 per hour, and the price for the reserved services (similar to the EC2 Large Standard Reserved Instances for 1 year fixed term) is about $0.12 per hour.

The simulation will start from the parallel running of 100 business processes, i.e. the maximum workload for each service instance is set as 10. The activity durations are generated based on the statistics and deliberately extended by a mixture of representative distribution models such as normal, uniform and exponential to reflect the performance of different cloud services. The mean activity durations are randomly generated in a wide range of 30 milliseconds to 30 seconds. Meanwhile, some noises are also added to a random selected activity in each business process to simulate the effect of system uncertainties such as network congestion and performance down time. Different ratio of noises (the added time delays divided by the activity durations) from 5% to 30% are implemented. The process structures are specified as DAG graphs similar to the securities exchange business process.

On the setting of temporal constraints and monitoring strategies: For each business process, an overall temporal constraint is assigned. The strategy for setting temporal constraint is adopted from the work in [13] where a normal percentile is used to specify temporal constraints and denotes the expected probability for on-time completion. Here, we specify the normal percentiles as 1.28 which denotes the probability of 90.0% for on-time completion if without any temporal violation handling. This setting can be regarded as the norm, i.e. the satisfactory performance for most clients and service providers. We employ the state-of-the-art checkpoint selection strategy introduced in [8] as the strategy for detecting the violations on workflow response time. As for the monitoring of system throughput, we pre-define a set of time points with the equal fixed time interval as introduced in Section 4.3. Specifically, the fixed time interval in our experiments is set as 60 seconds, i.e. around 20% of the average duration of a business process. Therefore, at the first 60 seconds, the system will verify whether 20% of the total activities (i.e. 20%*100*20=400) have been finished, and at the next time point, i.e. at the time points for 120 seconds, the system will verify whether 40% (i.e. 800) of the total activities have been finished, and so on so forth until the completion of all the 10,000 business processes. The throughput verification will be conducted at every pre-defined time points.

On the setting of temporal violation handling strategies: For the comparison purpose, we record the global violation rates under natural situations, i.e. without any handling strategies (denoted as NIL). The violation handling strategies we implemented including the standalone Workflow Local Rescheduling strategy, the standalone Extra Resource Recruitment strategy, and the combined of the two strategies. The Workflow Local Rescheduling strategy is based on ACOWR and the Extra Resource Recruitment strategy is based on the simple elastic resource provision strategy (denoted as SERP), as introduced in Section 4.3. For the standalone ACOWR or SERP, the same strategy will be applied both to the violations of response time and system throughput. As for the combined strategy (denoted as ACOWR+SERP), ACOWR will handle the violations of response time and SERP will handle the violations of system throughput respectively. The parameter settings for ACOWR are same as in [14]. As for SERP, we employ one additional instance for each type of service when a throughput violation is detected, and immediately release them when the system throughput is back to normal at the next throughput constraint. Based on the resource settings mentioned above, the average cost for ACOWR is $3.08*10^{-3} per time, which is mainly the computation cost for running the rescheduling strategy. Note that the cost for the re-allocation of workflow activities after rescheduling is not accounted here since the data transfer within a data centre is free in Amazon cloud. As for SERP, the cost is $16.7 per round where 20 cloud services are reserved and dedicated for the entire running period, i.e. an average of 8 hours per day.

<table>
<thead>
<tr>
<th>TABLE 1. Numbers of Temporal Violations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounds</td>
</tr>
<tr>
<td>R1</td>
</tr>
<tr>
<td>R2</td>
</tr>
<tr>
<td>R3</td>
</tr>
<tr>
<td>R4</td>
</tr>
<tr>
<td>R5</td>
</tr>
<tr>
<td>R6</td>
</tr>
<tr>
<td>R7</td>
</tr>
<tr>
<td>R8</td>
</tr>
<tr>
<td>R9</td>
</tr>
<tr>
<td>R10</td>
</tr>
</tbody>
</table>

Based on the above experimental settings, 10 rounds of experiments are implemented and each runs for 100 times to get average values. Table 1 shows the number of temporal violations recorded in each round of experiment. Clearly, the number of response time violations for workflow instances and the number of throughput violations for the workflow system both increase rapidly with the increase of noise, i.e. the embedded time delays to represent various system uncertainties. For example, with the every 5% increase of noise, the average increase of response time violations is around 200. The distribution of
service performance seems to have less effect on the temporal violations. For example, given the same noise, the average difference of throughput violations (e.g. R4 and R5, R6 and R7, R8 and R9) is around 20.

Figure 7 depicts the temporal violation rates (the unsuccessful rate for on-time completion of the entire 10K business processes) with different violation handling strategies. For comparison purpose, the results of NIL represent the natural condition, i.e. without any handling strategies, where the violation rates increase from 37% to 84% with an average of 65%. Both the combined strategy of ACOWR+SERP and the standalone ACOWR can ensure a very close to 0% violation rate. The standalone SERP strategy can also maintain a very low violation rate, i.e. with an average around 4%. The reason for such a difference is mainly because of the different granularity between ACOWR and SERP. Since ACOWR is triggered at every necessary and sufficient checkpoint while SERP only take place at pre-defined time points (a ratio around 13% according to the results shown in Table 1), there are some chances that significant time delays cannot be handled in time by SERP, and thus result in some unsuccessful on-time completion of business processes.

Figure 8 demonstrate the total temporal violation handling cost for each type of handling strategies. The cost for the standalone SERP is static with $16.7 in each round because only the reserved service instances are used. The cost for the combined strategy of ACOWR+SERP is also very stable from $31.7 to $35.1 with an average of $33.7. In contrast, the cost for the standalone ACOWR increases significantly from $16.2 to $56.8 with an average of $32.1. This is mainly because the standalone ACOWR needs to run many times (from 2 to 10 times in our experiments) to handle throughput violations, and hence the cost increases rapidly with the number of throughput violations.

In summary, according to the experimental results presented above, we can see that the combined strategy of ACOWR+SERP is the best one which can maintain the close to 0% violation rate while having the moderate cost among the three. Meanwhile, since our experimental settings actually allow for a probability of 10% violations (i.e. 90.0% for on-time completion), if a small violation rate is tolerable by the clients, e.g. below 5%, the standalone SERP strategy is also applicable and can significantly reduce the violation handling cost. In general, we can see that thanks to the dynamic and elastic resource provision provided by cloud computing, conventional violation handling strategies such as ACOWR can ensure the on-time completion of large numbers of business processes in a more cost-effective way.

6 Related Work

Traditional workflow systems are normally designed to support the business processes in a specific domain such as bank, hospital and school. Therefore, they can only invoke existing software applications (or software components) which have already developed and stored in the local repository, which limits the flexibility in the support of general and agile business processes [2, 18]. In the last decade, with the rapid development of Web services, the employment of remote software services from external service providers becomes possible. Therefore, the design and application of Web service based workflow systems starts to attract most of the attention from both researchers and practitioners, for example, the Windows Workflow Foundation (http://www.windowsworkflowfoundation.eu/) and the Kepler project (http://kepler-project.org/). In recent years, with the fast growth of high performance computing (HPC) and high throughput computing (HTC) such as cluster and grid, workflow systems are also being used as a type of middleware service which often underlies many large-scale complex e-science applications such as climate modelling, astrophysics, and chemistry [9, 20].

The work in [24] proposes a taxonomy and summaries a number of grid workflow systems such as Condor (http://www.cs.wisc.edu/condor/), Gridbus (http://www.gridbus.org/), Pegasus (http://pegasus.isi.edu/), and Triana (http://www.trianacode.org/). However, they mainly target at processing data and computation intensive activities for a single scientific workflow instance, rather than massive concurrent workflow instances for business processes.

The design of a cloud workflow system, as the combination of workflow system and cloud computing, comes from the natural needs for efficient and effective management of large numbers of business processes. However, as a cutting-edge research issue, the investigation on cloud workflow systems is so far still in its infancy. Besides SwinDeW-C, there are currently a few existing grid workflow systems investigating the migration from grid to cloud such as Pegasus in the cloud (http://pegasus.isi.edu/pegasus_cloud.php) and GridBus to CloudBus (http://www.cloudbus.org/cloudbus_flyer.pdf), but most of them are for scientific applications. Response time and system throughput are the most important measurements for the performance analysis of cloud workflow systems [3]. The work in [11] proposes a throughput maximisation strategy for transaction intensive cloud workflows. The work in [22] investigates the dynamic resource allocation for efficient parallel data processing in the cloud.

To the best of our knowledge, this is the first paper that proposes the solution of a cloud workflow system to address the management of running large numbers of time-constrained business processes.
7 Conclusions and Future Work

The concurrency of large numbers of client request has been widely seen in today’s e-business and e-government systems. Based on the analysis of a securities exchange business process, we have identified four basic system requirements for managing large numbers of business processes, viz. scalable computing resource provision, elastic computing resource delivery, efficient process management, and effective QoS monitoring and control. Based on that, the cloud workflow system is proposed as a competitive solution. We first present the general design of a cloud workflow system with the focus on its system architecture, basic functionalities and QoS management. Afterwards, based on such a general design, we have implemented a peer-to-peer based cloud workflow system prototype, SwinDeW-C. The architecture of SwinDeW-C (four-layered architecture), the system functionalities (realised in the functional components of SwinDeW-C coordinator and ordinary peers), and the QoS management (with the illustration of the performance management components) have been demonstrated to verify the effectiveness of our system design. The experimental results for the evaluation of system performance have shown that satisfactory on-time completion rate and better cost-effectiveness can be achieved with the dynamic and elastic provision of cloud resources.

In the future, the monitoring and violation handling strategies for the system throughput will be further enhanced. Meanwhile, the SwinCloud test bed will be extended in its size and capacity so that real world large scale business processes can be tested to further evaluate and improve our system design.

Acknowledgments. We are grateful for the discussions with Mr. Zhangjun Wu and his colleagues on the securities exchange business process. This work is partially supported by Australian Research Council under Linkage Project LP0990393.

References