

# A novel statistical time-series pattern based interval forecasting strategy for activity durations in workflow systems<sup>☆</sup>

Xiao Liu<sup>a,\*</sup>, Zhiwei Ni<sup>b</sup>, Dong Yuan<sup>a</sup>, Yuanchun Jiang<sup>b,c</sup>, Zhangjun Wu<sup>b,a</sup>, Jinjun Chen<sup>a</sup>, Yun Yang<sup>a</sup>

<sup>a</sup> Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn, Melbourne 3122, Australia

<sup>b</sup> School of Management, Hefei University of Technology, Hefei, Anhui 230009, China

<sup>c</sup> Joseph M. Katz Graduate School of Business, University of Pittsburgh, Pittsburgh, PA 15260, USA

## ARTICLE INFO

### Article history:

Received 2 March 2010

Received in revised form 1 September 2010

Accepted 29 November 2010

Available online 15 December 2010

### Keywords:

Workflow system

Activity duration

Interval forecasting

Statistical time series

Time-series patterns

## ABSTRACT

Forecasting workflow activity durations is of great importance to support satisfactory QoS in workflow systems. Traditionally, a workflow system is often designed to facilitate the process automation in a specific application domain where activities are of the similar nature. Hence, a particular forecasting strategy is employed by a workflow system and applied uniformly to all its workflow activities. However, with newly emerging requirement to serve as a type of middleware services for high performance computing infrastructures such as grid and cloud computing, more and more workflow systems are designed to be general purpose to support workflow applications from many different domains. Due to such a problem, the forecasting strategies in workflow systems must adapt to different workflow applications which are normally executed repeatedly such as data/computation intensive scientific applications (mainly with long-duration activities) and instance intensive business applications (mainly with short-duration activities). In this paper, with a systematic analysis of the above issues, we propose a novel statistical time-series pattern based interval forecasting strategy which has two different versions, a complex version for long-duration activities and a simple version for short-duration activities. The strategy consists of four major functional components: duration series building, duration pattern recognition, duration pattern matching and duration interval forecasting. Specifically, a novel hybrid non-linear time-series segmentation algorithm is designed to facilitate the discovery of duration-series patterns. The experimental results on real world examples and simulated test cases demonstrate the excellent performance of our strategy in the forecasting of activity duration intervals for both long-duration and short-duration activities in comparison to some representative time-series forecasting strategies in traditional workflow systems.

© 2010 Elsevier Inc. All rights reserved.

## 1. Introduction

Given the recent popularity of high performance computing for e-science and e-business applications, and more importantly the appearing applicability of workflow systems to facilitate high performance computing infrastructures such as grid and cloud computing, there is an increasing demand to investigate non-traditional workflow systems (Deelman et al., 2008; Yu and Buyya, 2005a; Yuan et al., 2010). One of the research issues is the interval forecasting strategy for workflow activity durations (Liu et al., 2008b; Nadeem and Fahringer, 2009a,b; Smith et al., 2004). It is

of great significance to deliver satisfactory QoS (Quality of Service) in workflow systems where workflow applications are often executed repeatedly. For example, grid workflow systems can support many data/computation intensive scientific applications such as climate modelling, disaster recovery simulation, astrophysics and high energy physics, as well as many instance intensive business applications such as bank transactions, insurance claims, securities exchange and flight bookings (Taylor et al., 2007; Wang et al., 2009). These scientific and business processes are modelled or redesigned as workflow specifications (consisting of such as workflow task definitions, process structures and QoS constraints) at the build-time modelling stage (Chen and Yang, 2010; Hsu and Wang, 2008; Liu et al., 2008c; Zeng et al., 2008). The specifications may contain a large number of computation and data intensive activities and their non-functional requirements such as QoS constraints on budget and time (Chen and Yang, 2008; Li et al., 2004; Zhao et al., 2004; Martinez et al., 2007). Then, at the run-time execution stage, with the support of workflow execution functionalities such as workflow scheduling (Stavrinides and Karatza, 2010; Neng and Zhang,

<sup>☆</sup> The initial work was published in Proc. of 4th IEEE International Conference on e-Science (e-Science08), pp. 23–30, Indianapolis, USA, December 2008.

\* Corresponding author at: Department of Information and Communication Technologies, Swinburne University of Technology, 1 Alfred Street, Hawthorn, Melbourne 3122, Australia. Tel.: +61 3 9214 8699.

E-mail address: [xliu@swin.edu.au](mailto:xliu@swin.edu.au) (X. Liu).

2009), load balancing (Taylor et al., 2007) and temporal verification (Chen and Yang, 2007), workflow instances are executed by employing the computing and data sharing ability of the underlying computing infrastructures with satisfactory QoS (Son and Kim, 2001). Interval forecasting for workflow activity durations, i.e. forecasting the upper bound and the lower bound of workflow activity durations, is required both at build time for creating workflow QoS specifications and at runtime for supporting workflow execution functionalities (Liu et al., 2008c; Zhuge et al., 2001). Therefore, effective interval forecasting strategies need to be investigated for workflow systems.

Traditional workflow systems are often designed to support process automation in specific application domains. For example, Staffware (van der Aalst and Hee, 2002) and SAP workflow system (SAP, 2010) are designed mainly to support business workflow applications for enterprises while GridBus Project (2010) and Kelper Project (2010) are designed mainly to facilitate scientific workflow applications for research institutes. However, with the emerging of high performance computing infrastructures, especially cloud computing which is becoming a type of computing utility to provide the basic level of computing services (Buyya et al., 2009), workflow systems should be able to support a large number of workflow applications regardless of their specific application domains. In the real world, most workflow applications can be classified as either scientific workflows or business workflows (van der Aalst and Hee, 2002; Barga and Gannon, 2007; Yu and Buyya, 2005b). Therefore, in this paper, we focus on two representative workflow applications: data/computation intensive scientific workflows (Deelman et al., 2008) and instance intensive business workflows (Liu et al., 2008a). In data/computation intensive scientific workflows, there is a large number of workflow activities which normally occupy some fixed resources and run continuously for a long time due to their own computation complexity, for example, the *De-dispersion* activity in a pulsar searching activity which takes around 13 h to generate 90 GB of de-dispersion files (NMDC, 2010). On the contrary, in instance intensive business workflows, there is a large volume of concurrent relatively simple workflow activities with fierce competition of dynamic resources, for example, the *Clearing* activity in a securities exchange business workflow which checks the balance between branches and clients with 50,000 transactions in 3 min (Liu et al., 2010). More examples are provided in Section 2.1. Therefore, it is important that the forecasting strategy in a workflow system can effectively adapt to the requirements of different workflow applications. Specifically, as will be discussed in Section 2.2, the execution time of a long-duration activity is mainly decided by the average performance of the workflow system over its lifecycle while the execution time of a short-duration activity is mainly dependent on the system performance at the current moment where the activities are being executed. However, in a traditional workflow system, a particular forecasting strategy is normally designed for a specific type of workflow applications and applied uniformly to all of its workflow instances (regardless of the differences between long-duration and short-duration activities). Therefore, new forecasting strategies in workflow systems need to be investigated.

Interval forecasting for activity durations in workflow systems is a non-trivial issue. On one hand, workflow activity durations consist of complex components. In workflows, activity durations cover the time intervals from the initial job submission to the final completion of each workflow activity. Hence, besides the exact running time on allocated resources, they also consist of extra time, i.e. workflow overheads. As introduced in (Prodan and Fahringer, 2008), there exist four main categories of workflow overheads in scientific workflow applications including middleware overhead, data transfer overhead, loss of parallelism overhead and activity related overhead. For example, in grid workflows, activity dura-

tions involve much more affecting factors than the running time of conventional computation tasks which are dominated by the load of high performance computing resources. On the other hand, the service performance is highly dynamic. For example, grid workflows are deployed on grid computing infrastructures where the performance of grid services is highly dynamic since they are organised and managed in a heterogeneous and loosely coupled fashion. Moreover, the workload of these shared resources, such as the computing units, the storage spaces and the network, is changing dynamically. Therefore, many traditional multivariate models which consist of many affecting factors such as CPU load, memory space, network speed (Dinda and O'Hallaron, 2000; Wu et al., 2007; Zhang et al., 2008) are either not satisfactory in performance or too complex to be applicable in practice, let alone the fact that it is very difficult, if not impossible, to measure these affecting factors in the distributed and heterogeneous computing environments such as grid and cloud (Dobber et al., 2007; Glasner and Volkert, in press). There are also many strategies which define a type of “templates” like models (Nadeem and Fahringer, 2009b; Nadeem et al., 2006). These models may include workflow activity properties such as workflow structural properties (like control and data flow dependency, etc.), activity properties (like problem size, executables, versions, etc.), execution properties (like scheduling algorithm, external load, number of CPUs, number of jobs in the queue, free memory, etc.), and then use “similarity search” to find out the most similar activity instances and predict activity durations (Nadeem and Fahringer, 2009b). However, they also suffer the same problems faced by traditional multivariate models mentioned above.

In this paper, we utilise time-series based forecasting models. In both scientific and business fields, time-series models are probably the most widely used statistical ways for formulating and forecasting the dynamic behaviour of complex objects (Chatfield, 2004; Liu et al., 2008b). A time series is a set of observations made sequentially through time. Some representative time series, including marketing time series, temperature time series and quality control time series, are effectively applied in various scientific and business domains (Huang et al., 2007). Similarly, a workflow activity duration time series, or duration series for short, is composed of ordered duration samples obtained from workflow system logs or other forms of historical data. Here, the samples specifically refer to the historic durations of the same workflow activity instances. Therefore, duration series is a specific type of time series in the workflow domain. In this paper, the term “time series” and “duration series” can be used interchangeably in most of the cases. In this paper, instead of applying traditional multivariate models, we conduct univariate time-series analysis which analyses the behaviour of duration series itself to build a model for the correlation between its neighbouring samples, or in other words, forecasting the future activity durations only based on the past activity durations (Chatfield, 2004). Unlike “template” based strategies which need to identify and collect various data about workflow activity properties, all the information are embedded implicitly in duration series. Therefore, the problems faced by both multivariate models and “template” based strategies, e.g. the complexity of models and the difficulty of data collection, can be overcome by time-series based forecasting strategies. Meanwhile, we focus on workflow systems in this paper since one of the fundamental requirements for time-series based forecasting strategies is the sufficient amount of samples. Clearly, such a requirement can normally be satisfied in workflow systems since the same workflow instances are frequently (or periodically) executed in order to realise some scientific or business processes.

Current forecasting strategies for computation tasks mainly reside on the prediction of CPU load (Akioka and Muraoka, 2004;

Zhang et al., 2008). However, this is quite different from the prediction of workflow activity durations in high performance computing environments due to the reasons mentioned above. Besides, most forecasting strategies also neglect another important issue that is the difference between long-duration activities (mainly in the data/computation intensive scientific applications) and short-duration activities (mainly in the instance intensive business applications). Typically for time-series forecasting strategies, the major requirements for long-duration activities are to handle the problems of limited sample size and frequent turning points in the duration series, while the major requirements for short-duration activities are to identify the latest system performance state and predict the changes (as detailed in Section 2.2). However, forecasting strategies in traditional workflow systems, either focusing on the prediction of the system performance state of the next moment or the prediction of the average system performance over a long period, but overlook the different requirements of long-duration activities and short-duration activities by treating them in an identical way. Therefore, there is a space to improve the accuracy of prediction if we differentiate long-duration activities from short-duration activities and investigate effective forecasting strategies which can adapt to their different requirements.

In this paper, a novel non-linear time-series segmentation algorithm named *K-MaxSDev* is proposed to facilitate a statistical time-series pattern based forecasting strategy for both long-duration activities and short-duration activities. For long-duration activities, two problems of the duration series which seriously hinder the effectiveness of conventional time-series forecasting strategies are limited sample size and frequent turning points. Limited sample size impedes the fitting of time-series models and frequent turning points where dramatic deviations take place significantly deteriorate the overall accuracy of time-series forecasting. To address such two problems, we utilise a statistical time-series pattern based forecasting strategy (denoted as *K-MaxSDev(L)* where *L* stands for long-duration). First, an effective periodical sampling plan is conducted to build representative duration series. Second, a pattern recognition process employs our *K-MaxSDev* time-series segmentation algorithm to discover the minimum number of potential patterns which are further validated and associated with specified turning points. Third, given the latest duration sequences, pattern matching and interval forecasting are performed to make predictions based on the statistical features of the best-matched patterns. Meanwhile, concerning the occurrences of turning points, three types of duration sequences are identified and then handled with different pattern matching results. As for short-duration activities, the key is to identify the system performance state at the current moment and detect possible changes. Therefore, with fewer steps, a more efficient time-series pattern based forecasting strategy (denoted as *K-MaxSDev(S)* where *S* stands for short-duration) is adapted to apply on the latest duration sequences for short-duration activities. Similarly, the advantages of pattern based forecasting strategy such as accurate interval and the handling of turning points are still well retained.

Both real world examples and simulated test cases are used to evaluate the performance of our strategy in *SwinDeW-G* (*Swinburne Decentralised Workflow for Grid*) workflow system. The experimental results demonstrate that our time-series segmentation algorithm is capable of discovering the smallest potential pattern set compared with three generic algorithms: Sliding Windows, Top-Down and Bottom-Up (Keogh et al., 2001). The comparison results further demonstrate that our statistical time-series pattern based strategy behaves better than several representative time-series forecasting strategies including MEAN (Dinda and O'Hallaron, 2000), LAST (Dobber et al., 2007), Exponential Smoothing (ES) (Dobber et al., 2007), Moving Average (MA) (Chatfield, 2004), and Auto Regression (AR) (Chatfield, 2004) and Network

Weather Service (NWS) (Wolski, 1997), in the prediction of high confidence duration intervals and the handling of turning points.

The remainder of the paper is organised as follows. Section 2 presents two motivating examples and problem analysis. Section 3 defines the time-series patterns and overviews our pattern based time-series forecasting strategy for both long-duration and short-duration activities. Section 4 proposes the novel time-series segmentation algorithm which is the core of the whole strategy. Sections 5 and 6 describe the detailed algorithms designed for long-duration activities and short-duration activities respectively. Section 7 demonstrates comprehensive simulation experiments to evaluate the effectiveness of our strategy. Section 8 discusses the related work. Finally, Section 9 addresses our conclusions and points out the future work.

## 2. Motivating examples and problem analysis

In this section, we first introduce two motivating examples, one for data/computation intensive scientific workflow and another for instance intensive business workflow. Afterwards, the problems with the forecasting of both long-duration and short-duration activities are analysed.

### 2.1. Motivating examples

#### 2.1.1. Example data/computation intensive scientific workflow: a pulsar searching workflow

Swinburne Astrophysics group has been conducting pulsar searching surveys using the observation data from Parkes Radio Telescope, which is one of the most famous radio telescopes in the world (<http://www.parkes.atnf.csiro.au/>). Pulsar searching is a typical data and computation intensive scientific application. It contains complex and time consuming activities and needs to process terabytes of data. Fig. 1 depicts the high level structure of a pulsar searching workflow, which is running on Swinburne high performance supercomputing facility (<http://astronomy.swin.edu.au/supercomputing/>).

There are three major steps in the pulsar searching workflow:

- (1) Data recording and initial processing. In Parkes Radio Telescope, there are 13 embedding beam receivers, by which signal from the universe are received. The raw signal data are recorded at a rate of one gigabyte per second by the ATNF (<http://www.atnf.csiro.au/>) Parkes Swinburne Recorder (<http://astronomy.swin.edu.au/pulsar/?topic=apsr>). Depending on different areas in the universe that the scientists want to conduct the pulsar searching survey, the observation time is normally around 1 h. The raw observation data recorded from the telescope includes the data from multiple beams interleaved. They are processed by a local cluster in Parkes in real time, where different beam files are extracted from the raw data files and compressed. The size of each beam file ranges from 1 GB to 20 GB depending on the observation time. The beam files are archived in tapes for permanent storage and future analysis.
- (2) Data preparation for pulsar seeking. Scientists analyse the beam files to find potentially contained pulsar signals. However, the signals are dispersed by the interstellar medium, where scientists have to conduct a *De-dispersion* step to counteract this effect. Since the potential dispersion source is unknown, a large number of de-dispersion files need to be generated with different dispersion trials. For one dispersion trial of one beam file, the size of de-dispersion file is 4.6 MB to approximately 80 MB depending on the size of the input beam file. In the current pulsar searching survey, 1200 is the minimum number of the

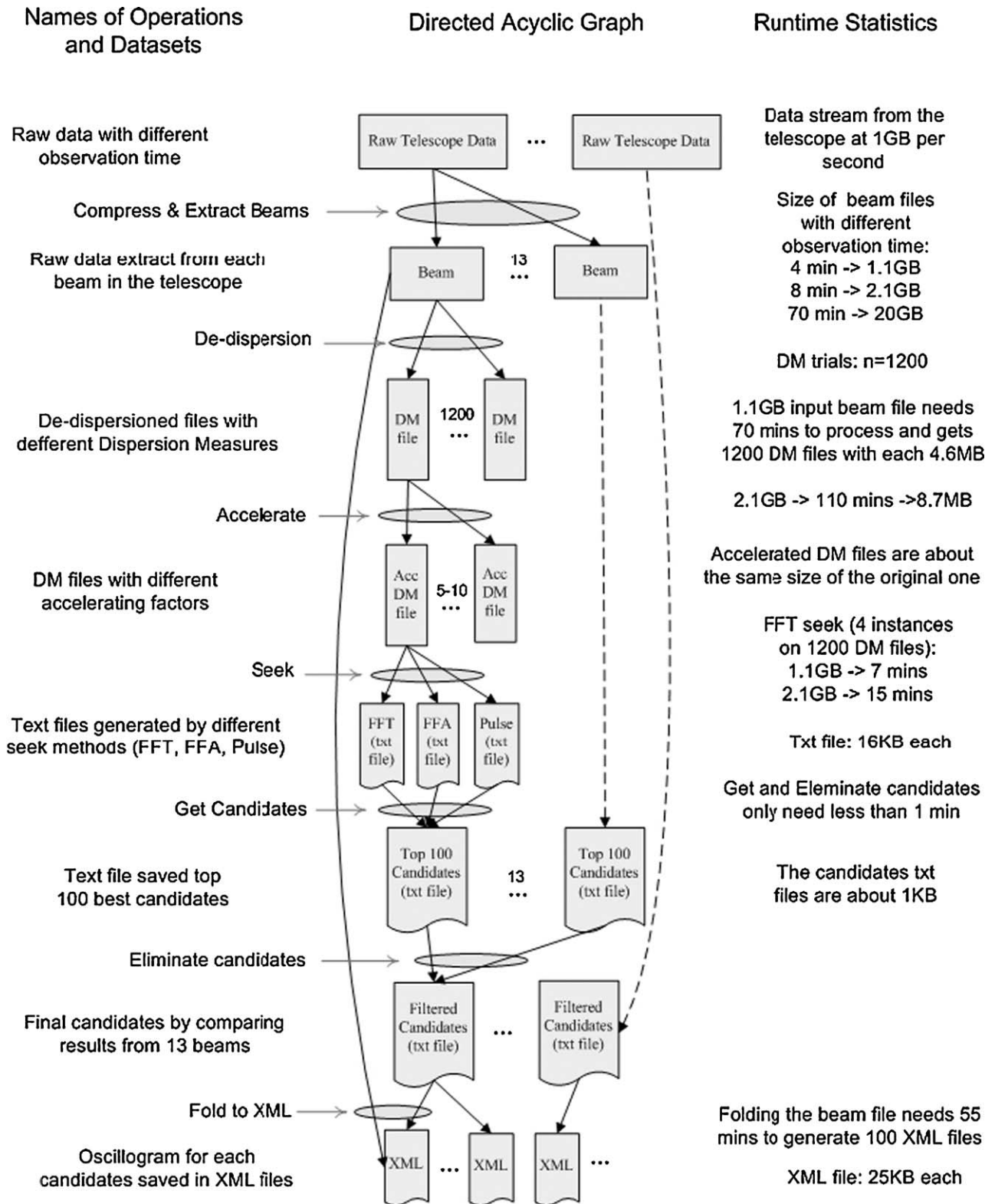


Fig. 1. A pulsar searching workflow in astrophysics.

dispersion trials, where the *De-dispersion* activity takes around 13 h to finish and generate 90 GB of de-dispersion files. Furthermore, for binary pulsar searching, every de-dispersion file needs another step of processing named *Accelerate*. This step generates the accelerated de-dispersion files with the similar size in the last de-dispersion step.

(3) Pulsar seeking process. Based on the generated de-dispersion files, different seeking algorithms can be applied to search pulsar candidates, such as *FFT* (Fast Fourier Transform) *Seeking*, *FFA* (Fast Fold Algorithm) *Seeking*, and *Single Pulse Seeking*. For example, the *FFT Seeking* takes 80 min to seek the 1200 de-dispersion files with the size of 90 GB. A candidate list of pulsars

is generated after the seeking step which is saved in a 1 kb text file. Furthermore, by comparing the candidates generated from different beam files in the same time session, interference may be detected and some candidates may be eliminated. With the final pulsar candidates, we need to go back to the de-dispersion files to find their feature signals and fold them to XML files. Each candidate is saved in a separated XML file about 25 kb in size. The *Fold to XML* activity takes close to 1 h depending on the number of candidates found in this searching process. At last, the XML files are visually displayed to scientists for making decisions on whether a pulsar has been found or not.

### 2.1.2. Example instance intensive business workflow: a securities exchange workflow

The securities exchange workflow is a typical instance-intensive workflow process which involves a large number of transactions and each of them is a relatively short workflow instance with only a few steps. Some steps of the workflow instance are executed concurrently. The example illustrated in Fig. 2 is the securities exchange workflow for the Chinese Shanghai A-Share Stock Market (<http://www.sse.com.cn/sseportal/en/>). There are more than one hundred securities corporations in this market and each corporation may have more than one hundred branches.

There are six major steps in the securities exchange workflow:

- (1) The first stage is “client entrustment” (Step 1). Every trading day, there are millions of clients online at the same time and everyone is a potential deal maker. The number of transactions can reach several millions per second at the peak time and the average is around several thousands. Entrustments are processed concurrently in more than 4000 branches scattered across the country.
- (2) The second stage is “fit and make deal” (Step 2 to Step 3). The raw entrustment data from clients are first validated at the corporation level to check whether the clients have enough money to make the deal and whether the deal is feasible (Step 2). After validation, the entrustments will be sent to the stock market where all the entrustments are fit to form the final valence and clinch a deal according to the trading rules. The fitting results are recorded into the database in the securities corporation and fed back to the clients (Step 3). The trading process is completed in several minutes. However, so far, the deal is technically completed but the share is not legally owned by the client until the completion of the entire workflow.
- (3) The third stage is “register shares variation and calculate capital variation” (Step 4 to Step 6). After 3:00 pm of the trading day, the market is closed to all clients. At this time, all the completed deals need to be archived and to be summed up by securities corporations for clearing. The size of the output file is about 50 G with tens of millions of transactions and the duration of the procedure is about 1.5 h (Step 4). After that, the corresponding fees and the credit and debit balance of every corporation are calculated. The size of the output file is about 50 M with several hundred thousands transactions and the duration is about 0.5 h (Step 5). Now all the trading data are transferred to Shanghai Stock Depository and Clearing Corporation of China (<http://www.chinaclear.cn/>). Registration shares variation of every client and calculation of the capital variation of every client are fulfilled at this stage. Registration ensures that the exchange occurred during the day is legal and the varied amount is registered on the client’s account. Calculation of the capital variation ensures that every corporation has the legal evidence to transfer money between firms and branches, or between branches and clients (Step 6). These are the two main data flow of stock exchange, one is the shares flow, and the other one is the capital flow.
- (4) The fourth stage is “settle the trades” (Step 7 to Step 9). The output files of the last step are divided by corporation ID and delivered to the securities firms concurrently (Step 7). The subsequent clearing steps are executed at the corporation level which can be processed concurrently among all the corporations. There are three levels of clearings: the first level clearing refers to the clearing between Clearing Corporation and securities firms, the second one refers to the clearing between securities firms and branches, and the third one refers to the clearing between branches and clients. After the corporation receives the clearing data files, there are two copies of the trading data in the clearing system. One comes from the database in the securities corporation and the other is from the Clearing Corporation. First, the *Pre-process* should be done to check whether there is any difference between these two files and ensures the first level clearing is correct. If they match, the clearing procedure can be started, otherwise, the problem caused this difference should be fixed before the next level of clearing. The *Pre-process* deals with a 50 M size data file with about 500k transactions in roughly 10 min (Step 8). Clearing is fulfilled in the clearing system, and it deals with a 50 M size data file with 500k transactions in 2 h; meanwhile, it checks the balance between branches and clients with a 5 M size data file, 50k transactions in 3 min. There is a check point between the second level clearing and the third level clearing. The share and capital balance of each branch in the data file from Clearing Corporation should agree with the sum of clients’ transactions in the recorded database. Otherwise, manual intervention will be conducted after the whole clearing process is completes (Step 9).
- (5) The fifth stage is “transfer capital” (Step 10 to Step 12). The output of the second and third level clearing is the money transfer details for each client who made deals during the day. It is a 20 M size data file with 200k transactions and it should be sent to the designated banks. Meanwhile, this file can also be used to produce the capital transfer details on the corporation level, i.e., which designated bank should transfer how much money to the Clearing Corporation. This procedure lasts for around 10 min (Step 10). The designated banks check the bills in about 30 min on both the client level and the branch level to ensure each entity has enough money to pay for the shares. The input data file is 20 M in size with 200k transactions. Generally speaking, clients will not be lack of money for the bills because they have been checked before the exchanges occurs (Step 11). If the branch is lack of money, certain amount of money is transferred from its deposit automatically. Then the capital is transferred between banks and clients and between banks and the Clearing Corporation. The input file for this step is a 5 M data file with 200k transactions and it will take around 50 min to be completed (Step 12).
- (6) The last stage is “produce clearing files” (Step 13 to Step 14). Both securities firms and designated banks should produce the clearing files for the Clearing Corp. This file includes the transferred details and some statistics. It is about 2 M in size with 20k records and takes around 10 min (Step 13). Clearing Corporation receives the clearing data file from the securities firms and banks concurrently. The balance of all the capital transferred is zero at the Clearing Corporation level. Otherwise, exception handling should be conducted with manual intervention (Step 14). The whole securities exchange workflow is completed afterwards.

### 2.2. Problem analysis

From the above two representative example workflows, we can obtain at least the following three observations.

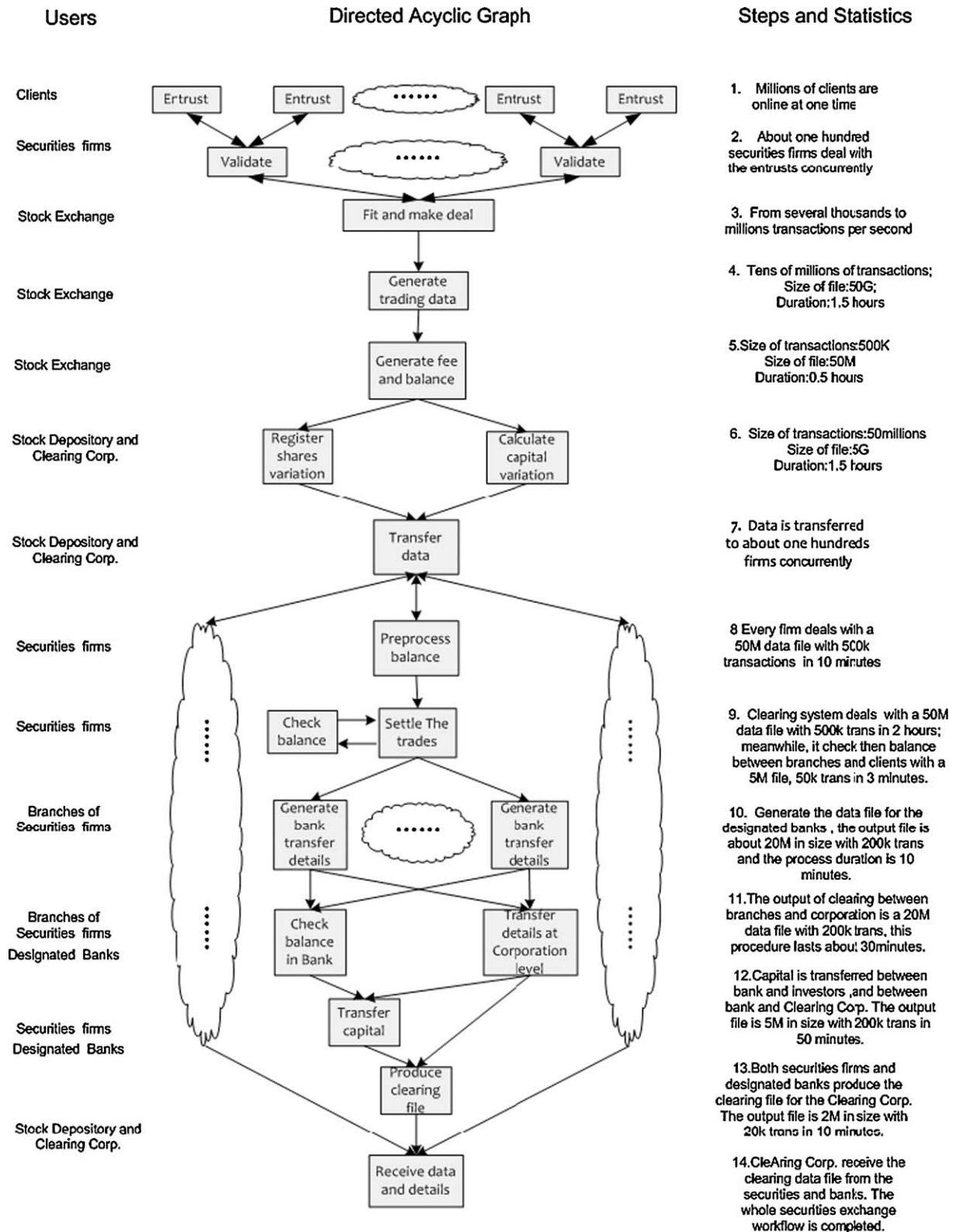


Fig. 2. A securities exchange workflow.

(1) Most activities in data/computation intensive scientific applications are long-duration activities. In the pulsar searching example, the *De-dispersion* activity requires around 13 h to generate 90 GB de-dispersion files; the *FFT Seeking* takes around 80 min to seek 1200 de-dispersion files; the *Fold to XML* activity takes close to 1 h to generate 100 XML files. These long-duration

activities are usually executed sequentially on a set of fixed high performance real or virtual machines.

(2) Most activities in instance intensive business applications are short-duration activities. In the securities exchange example, millions of client entrustments can be requested by clients all over the country and all of them are processed concurrently in

more than 4000 branches. Many activities such as *Validate, Fit and Make Deal*, requires prompt response time, i.e. very short durations; the *Pre-process* (Step 8) deals with 500k transactions in roughly 10 min, i.e. 1.2 s every 1k transactions; the clearing system checks the balance between branches and clients with 50k transactions in 3 min. These short-duration activities are executed concurrently by distributed high performance computing resources across different organisations.

- (3) *There are also some short-duration activities in scientific applications and some long-duration activities in business applications.* Although long-duration activities are the majority in scientific applications (similarly, short-duration activities are the majority in business applications), there are some short-duration activities in scientific applications (similarly, long-duration activities in business applications). For example, in the pulsar searching example, the top 100 best candidates are first selected from the FFT Seeking results and restored in a 1 kb file, and those invalid candidates are eliminated. The whole Get and Eliminate Candidates process is finished in 1 min, i.e. 0.6 s per candidate; in the securities exchange example, after the closing of the market, the Stock Depository and Clearing Cooperation needs to process 50 G size of transaction data and the duration of the procedure is about 1.5 h.

Therefore, the forecasting strategies of workflow systems should be able to meet the requirements of both long-duration and short-duration activities. In this paper, we focus on univariate time-series models. More specifically, we discuss the requirements for time-series based forecasting strategies, given both long-duration and short-duration activities. Most traditional forecasting strategies do not differentiate long-duration activities from short-duration activities. However, due to their different characteristics, specific requirements need to be identified and met.

For long-duration activities in data/computation intensive scientific applications, two problems of the duration series are limited sample size and frequent turning points. In workflow systems, long-duration activities are mainly those data/computation intensive activities which require constant processing of specific resources for a certain period of time. Therefore, an effective forecasting strategy is required to tackle these two problems which are explained as follows.

- (1) Limited sample size. Current work on the prediction of CPU load such as AR(16) demands a large sample size to fit the model and at least 16 prior points to make one prediction. However, this becomes a real issue for the duration series of long-duration workflow activities. Unlike CPU load which reflects an instant state and the observations can be measured at any time with high sampling rates such as 1 Hz (Zhang et al., 2008), the duration series of long-duration workflow activities denotes discrete long-term intervals since activity instances occur less frequently and take long time to be completed. Meanwhile, due to the heavy workload brought by these activities and the limits of available underlying resources (given requirements on the capability and performance), the number of concurrent activity instances is usually constrained in the systems. Therefore, the duration sample size is often limited and linear time-series models cannot fit effectively.
- (2) Frequent turning points. A serious problem which deteriorates the effectiveness of linear time-series models is the frequent occurrence of turning points where large deviations from the previous duration sequences take place (Zhang et al., 2008). This problem resides not only in the prediction of CPU load, but also in the forecasting of activity durations as observed and discussed in the many literatures (Dobber et al., 2006; Zhang et al., 2008). Actually, due to the large durations and the uneven

distribution of activity instances, the states of the underlying system environments are usually of great differences even between neighbouring duration-series segments. For example, in our Swinburne high performance supercomputing facility, the average number of job submissions in (11:00 am–12:00 noon) may well be bigger than that of (10:00–11:00 am) due to the normal schedule of working hours. Therefore, activity instances are unevenly distributed and frequent turning points are hence often expected. This further emphasises the importance of effective turning points discovery and handling in the forecasting of scientific workflow activity durations.

Compared with long-duration activities, the execution time of short-duration activities is normally much smaller. In workflow systems, short-duration activities are mainly those light-weight activities which normally need to be processed promptly but having a large number of instances in business applications. For example, millions of entrustments are processed distributed and concurrently in more than 4000 branches. These branches all have their own clusters of high performance machines which are managed by load balancing (workflow scheduling) strategies. Therefore, even at the peak time, every entrustment can be handled and responded within few seconds. For short-duration activities, the two basic requirements are the identification of the system performance state at the current moment and the detection of possible changes as detailed below.

- (1) Efficient identification of current performance state. Since short-duration activities are only executed for a small amount of time, their durations are strongly dependent on the system performance at those moments or specifically the performance of the specific underlying resources where these activities are being executed. For example, the number of client entrustments during (2:30–3:00 pm) is normally much bigger than that of (2:00–2:30 pm) since it is approaching to the close of the trading day. Therefore, clients may often experience longer system response time during (2:30–3:00 pm). Generally speaking, the system performance state is relatively stable during a very short period of time, e.g. a minute. Therefore, to estimate the durations of short-duration activities, the current system performance state needs to be identified in a very efficient way. Otherwise, the forecasting results will probably be less accurate due the performance state changes during the forecasting process.
- (2) Effective detection of possible changes. The problem of turning points also exists in the duration series of short-duration activities. The neighbouring samples in the duration series of short-duration activities have stronger correlations due to their temporal closeness. Therefore, compared with that of long-duration activities, the deviation of the next value from its latest neighbour is normally smaller. But, if the next value is on the edge of two different system performance states, the problem of turning points, similar to that of “level switches” introduced in Dobber et al. (2007), will harm the effectiveness of forecasting as well. Therefore, the possible changes of underlying system performance states should also be detected in order to maintain consistent forecasting performance.

To conclude, the deficiency of most traditional forecasting strategies lies in their ignorance of different problems faced by long-duration and short-duration activities. In this paper, to address the above issues, we propose a statistical time-series pattern based forecasting strategy which can achieve better and maintain consistent performance in the interval forecasting of both long-duration and short-duration activities in workflow systems. To the best of our knowledge, this is the first paper that system-

atically analyses and applies time-series patterns in the interval forecasting of both long-duration and short-duration activities in workflow systems.

### 3. Time-series pattern based forecasting strategy

As the problems analysed above, rather than fitting conventional linear time-series models in this paper which is not ideal, we investigate the idea of pattern based time-series forecasting. In this section, we first introduce the definition of statistical duration-series patterns and then we present the overview of our statistical time-series pattern based forecasting strategy.

#### 3.1. Time-series patterns

The motivation for pattern based time-series forecasting comes from the observation that for those duration-series segments where the number of concurrent activity instances is similar, these activity durations reveal comparable statistical features. It is obvious that when the number of concurrent activity instances increases, the average resources that can be scheduled for each activity decrease while the overall workflow overheads increase, and vice versa. Similar observation is reported in Dobber et al. (2006) between the task running time and the resource load in a global-scale (grid) environment. Accordingly, the duration series behaves up and down though it may not necessarily follow a linear relationship. This phenomenon, for example, is especially evident when workflow systems adopt market-driven or trust-driven resource management strategies that give preferences to a limited range of resources (Yu and Buyya, 2005a). Furthermore, if these segments reappear frequently during a duration series, they can be deemed as potential patterns which represent unique behaviour of the duration series under certain circumstances. Therefore, if we are able to define and discover these typical patterns, the intervals of future durations can be estimated with the statistical features of the closest patterns by matching the latest duration sequences. Based on this motivation, here, we present the definition of our statistical time-series patterns as follows.

**Definition 1 (statistical time-series patterns).** For a specific time series  $T = \{X_1, X_2, \dots, X_n\}$  which consists of  $n$  observations, its pattern set is  $Patterns = \{P_1, P_2, \dots, P_m\}$  where  $\sum_{i=1}^m Length(P_i) \leq n$ . For pattern  $P_i$  of length  $k$ , it is a reoccurred segment which has unique statistical features of sample mean  $\mu$  and sample standard deviation  $\sigma$ , where  $\mu = \sum_{i=1}^k X_i / k$  and  $\sigma = \sqrt{(\sum_{i=1}^k (X_i - \mu)^2) / (k - 1)}$ .

Note that in this paper, since we use statistical features to describe the behaviour of each time-series segment in an overall sense, the order of patterns is important for time-series forecasting while the order of sample points within each pattern is not. Meanwhile, for the purpose of generality, sample mean and standard deviation are employed as two basic statistical features to formulate each pattern. However, other criteria such as the median value, the trend and the length of each segment (the number of samples it consists of) can also be considered to make fine-grained definitions (Stroud, 2007).

Furthermore, for the ease of discussion in this paper, normal distribution model is selected. The rationale for choosing the normal distribution model mainly lies in its popularity in both research and practice. Without any priori knowledge, it is difficult, if not impossible, to find out a distribution model which can fit all the samples. Therefore, normal distribution model is often used in practice for both scientific and business applications. Meanwhile, as will be explained in Section 4, the basic theory employed in this paper is the “3 $\sigma$ ” rule which is a special case of Tchebysheff’s theorem (Stroud, 2007) when the samples follow the normal distribution

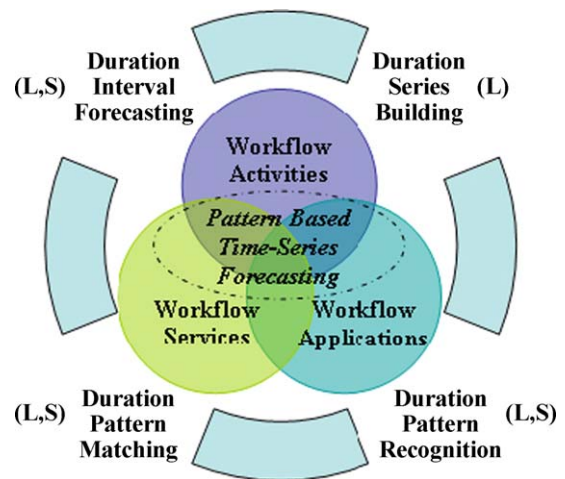


Fig. 3. Overview of statistical time-series pattern based forecasting strategy.

model. However, since Tchebysheff’s theorem can be applied to all distribution models, the work presented in this paper can also be applied to all distribution models as long as their sample mean and sample variance are available.

#### 3.2. Overview of statistical time-series pattern based forecasting strategy

Our statistical time-series pattern based forecasting strategy consists of two different versions. The first version is for long-duration activities in data/computation intensive scientific applications which includes an offline (build-time) time-series pattern discovery process and an online (runtime) forecasting process while the second version is for short-duration activities in instance intensive business applications where the pattern discovery and the interval forecasting are both conducted in an online (runtime) fashion. The overview of the strategy is presented as follows.

As depicted in the outlier of Fig. 3, our statistical time-series pattern based forecasting strategy consists of four major functional components which are duration series building, duration pattern recognition, duration pattern matching and duration interval forecasting. The inner three circles stand for the three basic factors concerned with activity durations: characteristics of the workflow activities (e.g. functional requirements, problem size, etc.), specifications of workflow applications (structure definitions, QoS constraints, etc.), and performance of workflow services (e.g. computing and storage services). As marked in Fig. 3 with L (Long-Duration) and S (Short-Duration), the complex version for long-duration activities includes all the four functional components while the simplified version for short-duration activities does not include the first functional component, i.e. duration series building.

Here, we illustrate the process of each functional component while leaving the detailed algorithms to Section 4 to Section 6.

- (1) Duration series building. Duration series building is designed to tackle the problem of limited sample size for long-duration activities. Therefore, it is only applied to the duration series of long-duration activities. Building duration series is actually a sampling process where the historical durations for a specific activity are sampled at equal observation unit from workflow system logs or other historical data sources, and then they are arranged according to the order of their submission time. In our strategy, we adopt a periodical sampling plan where the samples having their submission time belonging to the same observation unit of each period are treated as samples for the

same unit. Afterwards, a representative duration series is built with the sample mean of each unit. This periodical sampling plan effectively solves the problem of limited sample size and uneven distribution, since samples are assembled to increase the density of each observation unit. Meanwhile, recurrent frequencies of duration-series segments are statistically guaranteed. Therefore, we only need to analyse the representative duration series, and if a pattern is identified for this representative duration series, it is also statistically a pattern for the whole historic duration series. Note that there are many existing techniques to determine the specific value of the period (Law and Kelton, 2007) and it is strongly dependent on the real world situation of the workflow system. However, it can also be estimated by system managers from the main source of system workload. For example, many workflow systems especially those mainly dealing with heavy-weight scientific processes such as pulsar searching by a specific group of users, the system work load normally behaves periodically according to the lifecycle of these workflow processes. Specifically, if these processes are conducted routinely, say once every day, the system workload probably also behaves similarly with a period of 1 day in a rough sense. Therefore, the value of the period can be first chosen as 1 day and adjusted according to historic data.

- (2) Duration pattern recognition. Duration pattern recognition is to discover duration-series patterns by an effective pattern recognition process. It is applied to both long-duration and short-duration activities. However, for long-duration activities, the pattern recognition process contains two steps which are time-series segmentation and pattern validation, while for short-duration activities, only the first step applies. The first step is to identify a potential pattern set. For this purpose, we design a novel non-linear time-series segmentation algorithm named *K*-MaxSDev to achieve better performance in the discovery of the potential duration-series patterns compared with three generic algorithms. These duration-series patterns reflect the fundamental behaviour patterns of duration series. The second step is to check the validity of these segments by the minimum length threshold and specify the turning points for each valid pattern. The minimum length threshold is set to filter those segments which have no enough samples to make effective pattern matching. Meanwhile, we emphasise the proactive identification of turning points so as to eliminate large prediction errors. In our strategy, turning points are defined as those on the edge of two adjacent segments where the testing criterion is violated. Note that, only the time-series segmentation without pattern validation step is applied for short-duration activities. For short duration activities, the duration series only consists of a small buffer for the latest samples. Therefore, we only apply time-series segmentation here to identify the latest system performance patterns without the step for pattern validation and turning points identification. The reason is that the second step can only be carried out based on the representative duration series which is only applicable to long-duration activities.
- (3) Duration pattern matching. Duration pattern matching is applied to both long-duration and short-duration activities. Given the latest duration sequence, duration pattern matching is to find out the closest pattern with similarity search. For long-duration activities, the latest duration sequence is compared with valid duration-series patterns obtained through pattern recognition. Moreover, due to the occurrences of turning points, three types of duration sequences including the type where the sequence contains internal turning points, the type where the next value of the sequence is probably a turning point and the type for the remainder of the sequences, are first identified and then specified with different means and standard deviations.

For long-duration activities, the distance in similarity search between two sequences is defined as the absolute differences of their sample means. However, for short duration activities, the process of pattern matching and turning points discovery is conducted with different measurements. Since for short-duration activities, we only have a small number of patterns obtained from latest duration sequences. These patterns represent the latest system performance states and their lengths denote the corresponding duration where the system performance is within the same performance states, i.e. the same distribution models. The motivation here is that the length of the current segment should be similar to that of its latest pattern. Therefore, for the next value, its mean is defined as the sample mean of the previous duration segment. As for the turning points discovery, the heuristic rule here is that if the length of the current segment is larger than that of the last segment, then the next value is considered as a turning point. Otherwise, the next value is considered as still within the same distribution model as the previous duration segment.

- (4) Duration interval forecasting. Duration interval forecasting is applied to both long-duration and short-duration activities. With the pattern matching results (the estimated mean value and associated turning points, if any), duration intervals are specified accordingly with a given confidence under the assumption that the samples within a pattern follow the normal distribution (Law and Kelton, 2007).

Note that in this paper, our strategy focuses on one-step-ahead prediction which demonstrates excellent outcomes. The forecasting process can repeat itself by taking the predicted value as a new sample to the latest duration sequence, and hence it can perform multi-step-ahead prediction as well. However, the possible error margin could be huge, especially in highly dynamic system environments. Therefore, the specific strategies for multi-step-ahead prediction will be investigated in the future.

This section briefly describes the process of our statistical time-series pattern based forecasting strategy. To our best knowledge, this is the first work to investigate statistical time-series patterns to predict workflow activity durations. Furthermore, it is also the first work to investigate the different requirements and propose corresponding forecasting strategies within a consistent framework for both long-duration and short-duration activities in workflow systems. The technique details will be illustrated in the following sections.

#### 4. Novel time-series segmentation algorithm: *K*-MaxSDev

In this section, we propose *K*-MaxSDev, a non-linear time-series segmentation algorithm which is designed to formulate the duration series with minimal number of segments, i.e. potential duration-series patterns. Here, *K* is the initial value for equal segmentation and MaxSDev is the Maximum Standard Deviation specified as the testing criterion. The intuition for the initial *K* equal segmentation is an efficiency enhancement to the generic Bottom-Up algorithm which normally starts from the finest equal segments with the length of 2. We modify the initial equal segments with a moderately larger *K* which can be either user specified or with the empirical function proposed later in this section. This modification can save significant computation time at the initial stage. MaxSDev comes directly from the requirement for precise interval forecasting where the variances in each segment need to be controlled within a maximum threshold in order to guarantee an accurate small bound of the predicted durations, i.e. the duration intervals.

**Table 1**  
Notations used in *K*-MaxSDev.

$Seg_j$	The $j$ th segment of time series
$SDev(Seg_p, Seg_q)$	The standard deviation of all the samples in the $p$ th and $q$ th segments
InitialSeg( $K$ )	The initial $K$ equal segmentation for time series
Split( $Seg_j$ )	Split the $j$ th segment into two equal parts
Merge( $Seg_p, Seg_q$ )	Merge the $p$ th and $q$ th segments
Delete( $Seg_j$ )	Delete the $j$ th segment

The pseudo-code for *K*-MaxSDev algorithm is presented in Algorithm 1 and its notations (operations) are listed in Table 1.

**Algorithm 1.** *K*-MaxSDev time-series segmentation.

```

Algorithm: K-MaxSDev
Input: Duration series  $T = \{X_1, X_2, X_3, \dots, X_n\}$ , Const  $K$ , Const MaxSDev, Const  $R$ 
Output: Segmented duration series  $D = \{Seg_1, Seg_2, Seg_3, \dots, Seg_m\}$ 
InitialSeg( $K$ ); //Bottom-Up
Do
   $R = R - 1$ ; // maximum repetition time -1
  for ( $j=2, j \leq K-1, j++$ )
    if  $SDev(Seg_{j-1}, Seg_j) \leq MaxSDev$  //Sliding Windows-Backward
       $Seg_j = Merge(Seg_{j-1}, Seg_j)$ ;
      Delete( $Seg_{j-1}$ );  $K = K - 1$ ; Update the index;
    end
    if  $SDev(Seg_j, Seg_{j+1}) \leq MaxSDev$  //Sliding Windows-Forward
       $Seg_{j+1} = Merge(Seg_j, Seg_{j+1})$ ;
      Delete( $Seg_j$ );  $K = K - 1$ ; Update the index;
    end
    if  $SDev(Seg_{j-1}, Seg_j) > MaxSDev$  and  $SDev(Seg_j, Seg_{j+1}) > MaxSDev$ 
      Split( $Seg_j$ );  $K = K + 1$ ; Update the index; //Top-Down
    end
  end
Until ( $R = 0$  || no individual segments can be merged with their neighbours)

```

*K*-MaxSDev is a hybrid algorithm which can be described as follows. It starts from Bottom-Up with  $K$  initial equal-segmentation, followed by the repetitive process of Sliding Window and Top-Down with the testing criterion of MaxSDev. Here, the basic merging element of Sliding Window is not a single point but a segment. Meanwhile, the windows can slide both forward and backward to merge as many segments as possible. As for Top-Down, it equally splits those segments which cannot be merged with any neighbours. This repetitive process stops when either the maximum repetition time  $R$  is reached or all individual segments cannot be merged with their neighbours any more. Since the condition of “all individual segments cannot be merged with their neighbours any more” requires additional programming (or memory space) to store the segmentation results of the previous repetition, in practice, to set the maximum repetition time is a more efficient way to stop the repetition process. Meanwhile, the maximum repetition time can prevent the occurrence of endless loop (e.g. the deadlock of *Split* and *Merge*). According to our experiments, for most of the duration series, the repetition process (i.e. the do-until part as shown in Algorithm 1) will be repeated for no more than 3 times in most cases. Therefore, the maximum repetition time  $R$  is set as 5 in this paper (including the experiments demonstrated in Section 7) but may be changed according to different system environments. Evidently, the aim here is to discover the minimum number of segments. The purpose of designing *K*-MaxSDev rather than employing existing algorithms is to take full advantage of these three generic algorithms to achieve a better performance. Specifically, *Split* is to divide

the current segment into two equal parts. In the case where the current segment has odd number of sample points, the point at

the middle position will be included into the left segment. For example, if we *Split* a segment of  $\{p_1^1, p_2^2, p_3^3, p_4^4, p_5^5, p_6^6, p_7^7\}$ , it then becomes  $\{p_2^1, p_3^2, p_4^3, p_5^4, p_6^5, p_7^6\}$ . Its subsequent sample needs to update their index accordingly. *Merge* is to merge two neighbouring segments into one. For example, if we *Merge* two segments of  $\{p_1^1, p_2^2, p_3^3, p_4^4, p_5^5\}$ , it then becomes  $\{p_1^1, p_2^2, p_3^3, p_4^4, p_5^5\}$ . Its subsequent sample needs to update their index accordingly. *Delete* is to “delete” the previous segment after *Merge*, i.e. update the index of the sample points. Like in the above *Merge* operation, the fourth segment is deleted, i.e. the previous fifth segment now becomes the new fourth segment, so does the subsequent segments in the time series. Section 7.2.1 will demonstrate a detailed example process.

The empirical function for choosing  $K$  is based on the equal segmentation tests. The candidates of  $K$  are defined to be in the form of  $2i$ , e.g. 4, 6, 8 or 10, and should be a value no bigger than  $n/4$  where  $i$  is a natural number and  $n$  is the length of the duration series. Otherwise, *K*-MaxSDev is turning into Bottom-Up with low efficiency. The process is that we choose the candidates in a descending order, e.g. from  $n/4, n/8, \dots$  to 4, 2, and perform equal segmentation. We calculate the mean for each segment and rank each candidate according to the average of the absolute differences between the means of neighbouring segments. Finally, the candidate ranked first will be selected. The intuition is that initial segmentation should ensure large differences between the means of segments, so as to guarantee the efficiency of the segmentation algorithm. Note that different  $K$  values may result in the differences for the efficiency of the segmentation algorithm, but the final results will not be affected by the selection of different  $K$  values.

The formula for the empirical function is denoted as Formula (1):

$$\text{Rank}(k) = \frac{\sum_{i=1}^{k-1} |\text{Mean}_{i+1} - \text{Mean}_i|}{k-1} \quad (1)$$

The theoretical basis for MaxSDev is Tchebysheff's theorem (Stroud, 2007) which has been widely used in the statistics theory. According to the theorem, given a number  $d$  greater than or equal to 1 and a set of  $n$  samples, at least  $[1 - (1/d)^2]$  of the samples will lie within  $d$  standard deviations of their mean, regardless what the actual probability distribution is. For example, 88.89% of the samples will fall into the interval of  $(\mu - 3\sigma, \mu + 3\sigma)$ . The value of  $\mu$  and  $\sigma$  can be estimated by the sample mean and sample standard deviation respectively. If it happens to be a normal distribution, Tchebysheff's theorem is turning into one of its special cases, i.e. the "3 $\sigma$ " rule which means with a probability of 99.73% that the sample is falling into the interval of  $(\mu - 3\sigma, \mu + 3\sigma)$  (Stroud, 2007). Therefore, if we control the deviation to be less than  $m\%$  of the mean, then  $3\sigma \leq m\% \times \mu$  is ensured. We can thus specify MaxSDev by Formula (2):

$$\text{MaxSDev} = \frac{m\%}{3} \times \mu \quad (2)$$

For other non-normal distribution models,  $d$ , i.e. the value 3 in Formula (2), can be replaced by a larger value such as 4, 5, or 6 according to general Tchebysheff's theorem. For example, if

complexity of  $K$ -MaxSDev is in a magnitude of  $R \times n$ , i.e. a linear time complexity  $O(n)$ .

## 5. Interval forecasting for long-duration activities in data/computation intensive scientific applications

As presented in Section 3, the interval forecasting strategy for long-duration activities in data/computation intensive scientific applications are composed of four functional components: duration series building, duration pattern recognition, duration pattern matching, and duration interval forecasting. In this section, we will propose the detailed algorithms for all the functional components. Note that since pattern matching and interval forecasting are always performed together, we illustrate them within an integrated process.

### 5.1. Duration series building

As mentioned in Section 3.2, building the representative duration series is to address the problem of limited sample size for long-duration activities. The representative duration series is built with a periodical sampling plan where the samples having their submission time belonging to the same observation unit of each period are treated as samples for the same unit. Afterwards, a representative duration series is built with the sample mean of each unit. The pseudo-code for the algorithm is presented in Algorithm 2. Here, the specific value of the period can be either user defined or obtained by existing techniques.

**Algorithm 2.** Representative time-series building algorithm (L).

<b>Algorithm: Representative Time-Series Building</b>
<b>Input:</b> $m$ Duration series $t^i = \{x_1^i, x_2^i, x_3^i, \dots, x_n^i\}$ with the same length of a period
<b>Output:</b> Representative Time Series $T = \{X_1, X_2, X_3, \dots, X_n\}$
<pre> While not end of the time series   for (j=1, j ≤ m, j++)     for (k=1, k ≤ n, k++)       <math display="block">X_k = \frac{\sum_{i=1}^m x_k^i}{m}</math>     end   end end </pre>

### 5.2. Duration pattern recognition

As introduced in Section 3.2, for long duration activities, the duration pattern recognition process consists of two steps which are time-series segmentation and pattern validation. The time-series segmentation step is to discover the minimal set of potential patterns with our novel  $K$ -MaxSDev non-linear time-series segmentation algorithm as proposed in Section 4. Therefore, its pseudo-code is omitted here. The pseudo-code for the second step of pattern validation together with turning points discovery is presented in Algorithm 3. With the segmented duration series, namely the discovered potential duration-series patterns, we further check their validity with the specified minimum length threshold defined as *Min.pattern.length* which should be normally larger than 3 in order to get effective statistics for pattern matching and interval forecasting. Afterwards, we identify those turning points associated with each valid pattern. Specifically, turning points are defined as the points where the testing criterion of MaxSDev is violated. Since our  $K$ -MaxSDev segmentation algorithm actually ensures that the violations of MaxSDev only occur on the edge of two adjacent seg-

we want the interval to cover around 99.73% of the samples like the normal distribution model,  $d$  should be larger than 6 since  $[1 - (1/6)^2] = 97.22\%$ . In this paper, since we focus on the normal distribution model,  $d$  is specified as 3.

$K$ -MaxSDev is the core algorithm for pattern recognition in our strategy. For long-duration activities, it is applied to discover the minimal set of potential duration-series patterns. For short-duration activities, it is applied to discover the latest duration-series patterns and the current duration-series segment. The basic computations in  $K$ -MaxSDev are the calculation of statistic values such as sample means and standard deviations. The computation time for  $K$ -MaxSDev mainly depends on the length of the time series (i.e. the number of sample points in the duration series). Given a duration series of ' $n$ ' sample points and the maximum repetition time of a constant  $R$  (no more than 5), the time

ments, as shown in Algorithm 3, turning points are either specified by the mean of the next invalid pattern or the first value of the next valid pattern. Evidently, this specification of turning points captures the locations where large prediction errors mostly tend to happen.

**Algorithm 3.** Pattern validation and turning points discovery algorithm (L).

<p><b>Algorithm: Pattern validation and turning points discovery</b>  <b>Input:</b> Segmented duration series <math>D = \{Seg_1, Seg_2, \dots, Seg_m\}</math>,  Const Min_pattern_length  <b>Output:</b> Duration patterns Pattern[] and associated turning points</p> <pre> Pattern[] = {}; for (i=1, i ≤ D.length, i++) // Checking Patterns     Pattern[i] = Seg<sub>i</sub>;     Pattern[i].mean = Seg<sub>i</sub>.mean, Pattern[i].sdev = Seg<sub>i</sub>.sdev;     if Seg<sub>i</sub>.length &lt; Min_pattern_length         Pattern[i].valid = false;     else         Pattern[i].valid = true;     end end for (j=1, j ≤ Pattern.length, j++) // Specifying Turning Points     if Pattern[i].valid = true         if Pattern[i+1].valid = true             Pattern[i].turningpoint = Pattern[i+1].firstvalue         else             Pattern[i].turningpoint = Pattern[i+1].mean         end     end end end </pre>
--

### 5.3. Pattern matching and interval forecasting

As mentioned in Section 3.2, the most important issue for pattern matching and interval forecasting is to identify the type of the given latest duration sequence. Specifically, three types of duration sequences are defined in this paper including the type where the sequence contains internal turning points, the type where the next

value of the sequence is probably a turning point and the type for the remainder of the sequences. As presented in Algorithm 4 the types of sequences are identified based on standard deviations.

For the first type of duration sequence which has a standard deviation larger than MaxSDev, it cannot be matched with any valid patterns and must contain at least one turning point. Therefore, we need to first locate the turning points and then conduct pattern matching with the remainder of the duration sequence. However, to ensure the effectiveness of the prediction, if the length of the remainder duration sequence is smaller than the minimum pattern length, the duration interval is specified with the mean of the duration sequence while its standard deviation is substituted by MaxSDev. Note that the matched pattern is defined as the valid pattern of which the statistical mean has the minimum absolute difference with that of the latest duration sequence. As for the second type, it is identified when the standard deviation of the latest duration sequence is larger than that of the matched pattern but smaller than MaxSDev. In this case, the next value of the sequence will probably be a turning point since it is on the edge of two different patterns according to our pattern recognition process. Therefore, the mean of the next value is specified with the associated turning point of its matched pattern and the standard deviation is specified with MaxSDev. For the third type of sequence where its standard deviation is smaller than its matched pattern, the next value is predicted with the statistical features of the matched patterns as a refinement to that of the latest duration sequence. As presented in Algorithm 4, we illustrate the specification of duration intervals with normal distribution which is a kind of symmetric probability distribution. For example, given  $\lambda$  which is the  $\alpha\%$  confidence percentile for normal distribution, the predicted  $\alpha\%$  confidence interval is  $(\mu - \lambda\sigma, \mu + \lambda\sigma)$ , where  $\mu$  and  $\sigma$  stands for the statistical sample mean and standard deviation respectively. In our strategy,  $\mu$  and  $\sigma$  are specified according to different types of duration sequences. However, if the duration samples of a pattern follow non-normal distribution models, such as Gamma distribution, log-normal distribution or Beta distribution, changes can be made to the predicted intervals accordingly (Law and Kelton, 2007).

**Algorithm 4.** Pattern matching and duration interval forecasting algorithm (L).

<p><b>Algorithm: Pattern matching and duration interval forecasting</b>  <b>Input:</b> Latest duration sequence DS, Duration patterns Pattern[], Const ms=MaxSDev,  Const Min_pattern_length, Const <math>\lambda</math> (<math>\alpha\%</math> confidence percentile)  <b>Output:</b> Matched pattern MP and duration interval DI(min, max)</p> <pre> if DS.sdev ≥ ms // The latest duration sequence contains turning point in itself     While (DS.sdev ≥ ms) // remove the part before and include the turning point         DS = DS - DS.firstvalue;     end     DI(min, max) = (DS.mean - <math>\lambda</math> * ms, DS.mean + <math>\lambda</math> * ms); else // <math>\lambda</math> is the <math>\alpha\%</math> confidence percentile for normal distribution     if DS.length ≥ Min_pattern_length // ensure a valid pattern matching         MP = Min (Abs(Pattern.mean - DS.mean))         // Min() returns the pattern with minimum absolute difference of mean         if MP.sdev &lt; DS.sdev &lt; ms             // The next value is highly possible to be a turning point             DI(min, max) = (MP.turningpoint - <math>\lambda</math> * ms, MP.turningpoint + <math>\lambda</math> * ms);         else if DS.sdev ≤ MP.sdev             DI(min, max) = (MP.mean - <math>\lambda</math> * MP.sdev, MP.mean + <math>\lambda</math> * MP.sdev)         end     else // Sample size is too small to match a valid pattern         DI(min, max) = (DS.mean - <math>\lambda</math> * ms, DS.mean + <math>\lambda</math> * ms);     end end end </pre>
--

## 6. Interval forecasting for short-duration activities in instance intensive business applications

In this section, we present the detailed algorithms for the interval forecasting for short-duration activities in instance intensive business applications which consists of three functional components including duration pattern recognition, pattern matching, and interval forecasting. Similarly, since pattern matching and interval forecasting are always performed together, we illustrate them with an integrated process.

### 6.1. Duration pattern recognition

Different from long-duration activities where the duration pattern recognition process is conducted on the representative duration series, for short-duration activities, the duration pattern recognition process is conducted on the latest duration sequence. The latest duration sequence is acted as a buffer which records the latest historic durations for a specific activity and its size should be normally large enough to create about 5 or 6 segments as suggested in Keogh et al. (2001) to maintain a good performance for online segmentation algorithms. For short-duration activities,

obvious that the average length of patterns should be large when the behaviour of the system performance is stable. On the contrary, if the current system performance is highly dynamic, the length of each pattern is normally very small. Therefore, based on the observation of the pattern length, we can determine whether the current behaviour of system performance is stable or not. Obviously, the pattern matching process here for short duration activities is actually a comparison process with the length of current segment and its previous neighbouring segment, i.e. the second rightmost segment. The pseudo-code is presented in Algorithm 5. As can be seen, if the length of current segment is smaller than that of its previous neighbour, this implies that the next value is probably still within the current segment, or in another word, the next value is not a turning point. Therefore, the duration interval is defined based on the mean value and the standard deviation of the current segment. However, if the length of current segment is equal to or larger than that of its previous neighbour, this implies that the next value is probably a turning point. Therefore, in this case, the duration interval is defined based on the mean value of the current segment and the value of MaxSDev.

**Algorithm 5.** Pattern matching and duration interval forecasting algorithm (S).

<p><b>Algorithm: Pattern matching and duration interval forecasting</b>  <b>Input:</b> The latest segmented duration series <math>D = \{ Seg_1, Seg_2, \dots, Seg_m \}</math>,  Const <math>ms = MaxSDev</math>, Const <math>\lambda</math> (<math>\alpha\%</math> confidence percentile)  <b>Output:</b> Duration interval <math>DI(\min, \max)</math></p> <p><math>\mu = \mu_m</math>; (the mean value of the rightmost segment)</p> <p><b>If</b> <math>Seg_m \text{ length} &lt; Seg_{m-1} \text{ length}</math> //the length of the current segment is smaller than its previous one, which implies that the next value is probably still within the last segment, namely not a turning point  <math>DI(\min, \max) = (\mu_m - \lambda * \sigma_m, \mu_m + \lambda * \sigma_m)</math></p> <p><b>else</b> //the next value is probably a turning point, the standard deviation is changed by MaxSDev to increase the interval  <math>DI(\min, \max) = (\mu_m - \lambda * ms, \mu_m + \lambda * ms)</math></p> <p><b>end</b></p>
--

$K$ -MaxSDev is conducted to discover the current system performance state, i.e. the duration distribution model of the rightmost segment. Since our  $K$ -MaxSDev segmentation algorithm ensures that the final segments cannot be merged with any of its neighbours, it can accurately divide different system performance states from each other. Therefore, the rightmost segment which contains the latest historic durations can represent the current system performance state with its distribution model. Meanwhile, the other segments in the latest duration series which represent the latest system performance states can be regarded as the latest system performance patterns. Since this algorithm is very similar to that for long-duration activities except that the input is the latest duration sequence instead of representative duration series, the pseudo-code is omitted here.

### 6.2. Pattern matching and interval forecasting

After the latest system performance patterns are identified with the duration pattern recognition process for short-duration activities, the next step is for pattern matching and interval forecasting. However, different from long-duration activities, what we focus here is the length of these patterns instead of their distribution models. The motivation here is that the length of a pattern reflects its lifecycle, i.e. the lasting time for a specific system performance state. Since the duration points are sampled at equal length, it is

## 7. Evaluation

In this section, both real world examples and simulated test cases are used to evaluate the performance of our forecasting strategy. The real world historic data employed are collected from the workflow system logs and other sources of historic data for three example workflow applications, viz. the pulsar searching workflow and the securities exchange workflow as demonstrated in Section 2.1, and an additional weather forecast scientific workflow introduced in Liu et al. (2010a), are employed for training and testing. The weather forecast workflow is also a typical data and computation intensive scientific workflow which deals with the processing of large-size meteorology data collected from geographically distributed equipments such as radars and satellites (<http://data.cma.gov.cn/index.jsp>). The forecasting processes for two example activities (one for a long-duration activity and another for short-duration activity) are demonstrated. Meanwhile, simulation experiments are conducted in our SwinDeW-C cloud workflow system. In order to simulate more general system environments, the mean values for the simulated activity durations are specified according to real statistics but extended deliberately with different variances. The simulation data and related materials can be found online.<sup>1</sup>

<sup>1</sup> <http://www.ict.swin.edu.au/personal/xliu/doc/IntervalForecasting.rar>.

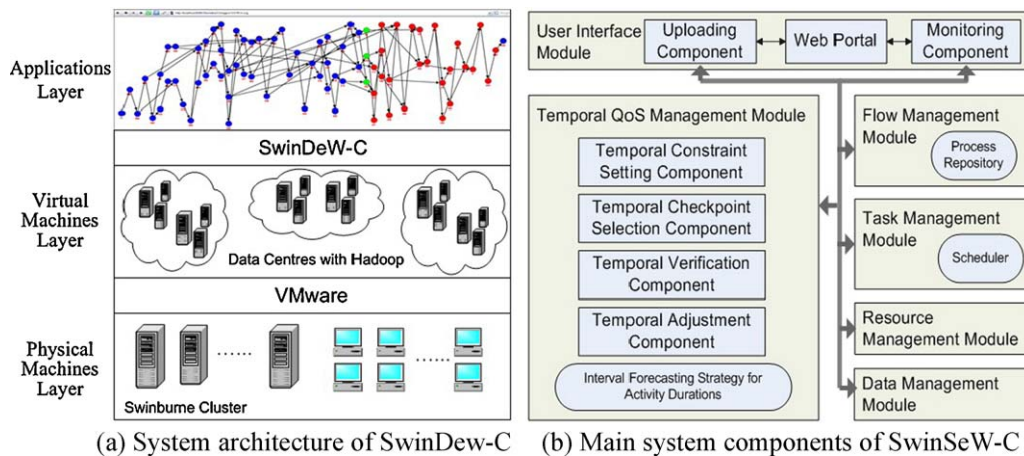


Fig. 4. Simulation environment of SwinDeW-C: (a) system architecture of SwinDeW-C and (b) main system components of SwinDeW-C.

### 7.1. Simulation environment

SwinDeW-C (Swinburne Decentralised Workflow for Cloud) (Yang et al., 2008) is a cloud workflow system prototype developed based SwinCloud, a cloud simulation test bed built on SwinDeW (Yan et al., 2006) and SwinDeW-G (Swinburne Decentralised Workflow for Grid) (Yang et al., 2007). SwinCloud currently consists of hundreds of virtualised nodes based on the physical machines of 10 servers and 10 high-end PCs in the Swinburne Astrophysics Supercomputer Node (<http://astronomy.swin.edu.au/supercomputing/>) of Swinburne Cluster. The Swinburne Astrophysics Supercomputer Node comprises 145 Dell Power Edge 1950 nodes each with: 2 quad-core Clovertown processors at 2.33 GHz (each processor is 64-bit low-volt Intel Xeon 5138), 16 GB RAM and  $2 \times 500$  GB drives. To simulate the cloud computing environment, we set up VMware (2010) software on the physical servers and create virtual clusters as data centres. Fig. 4 shows the simulation environment of SwinDeW-C which is sufficient for the simulation purpose including the running of the forecasting strategies. For more details about SwinDeW-C, please refer to Liu et al. (2010b).

As depicted in Fig. 4(a), every data centre created is composed of 8 virtual computing nodes with storages, and we deploy an independent Hadoop file system (Hadoop, 2010) on each data centre. SwinDeW-C runs on these virtual data centres that can send and retrieve data to and from each other. Through a user interface at the applications layer, which is a Web based portal, we can deploy workflows and upload application data. SwinDeW-C is designed for large scale workflow applications in the cloud computing environments. In Fig. 4(b), we illustrate the key system components of SwinDeW-C.

**User Interface Module:** The cloud simulation test bed is built on the Internet and a Web browser is normally the only software needed at the client side. This interface is a Web portal by which users can visit the system and deploy the applications. The *Uploading Component* is for users to upload application data and workflows, and the *Monitoring Component* is for users, as well as system administrators to monitor workflow execution.

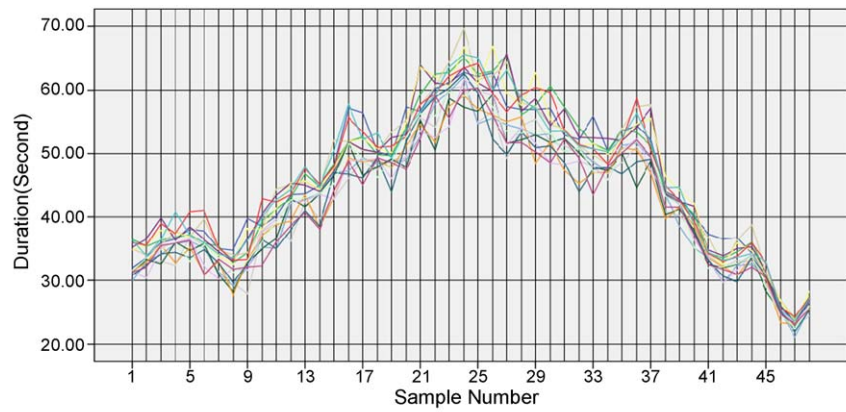
**Temporal QoS Management Module:** As an important reinforcement for the overall workflow QoS, a temporal QoS management module is being implemented in SwinDeW-C. The module consists of four basic components: temporal constraint setting which specifies temporal constraints in cloud workflow specifications at build time; temporal checkpoint selection which dynamically selects activity points along workflow execution for temporal verifica-

tion (Chen and Yang, 2009), temporal verification which checks the current temporal consistency state according to temporal consistency models (Chen and Yang, 2008), and temporal adjustment which handles detected temporal violations to ensure satisfactory temporal QoS. All the components in the temporal QoS management module are highly dependent on the effective prediction of workflow activity durations. Therefore, our interval forecasting strategy is included and serves as a basic functionality to support temporal QoS management in SwinDeW-C.

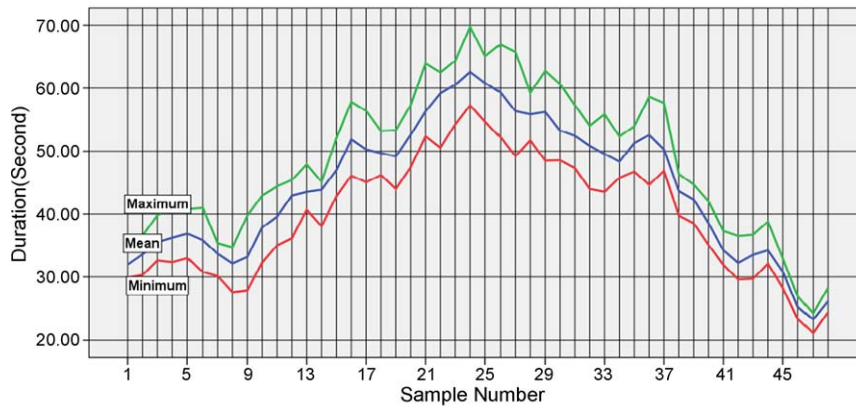
**Other Modules:** The *Flow Management Module* has a *Process Repository* that stores all the workflow instances running in the system. The *Task Management Module* has a *Scheduler* that schedules ready tasks to data centres during the runtime stage of the workflows. Furthermore, the *Resource Management Module* keeps the information of the data centres' usage, and can trigger the adjustment process in the data placement strategy. The *Data Management Module* includes the strategies for data placement, the data catalogue for service registry and lookup, and other components for cloud data management.

### 7.2. Simulation experiments and results

In this section, two examples, one for data and computation intensive activity and another one for instance intensive activity, are first presented to illustrate the forecasting process in detail. To ensure the representativeness, a selection process is first conducted in workflow specifications to search for a group of activities which are typically data and computation intensive, and a group of activities which are typically instance intensive. After that, only those activities with moderate mean values (the mean values are close to the mean values of all samples in the group) are selected as candidates. In Sections 7.2.1 and 7.2.2, we randomly choose one of the candidates from each group to demonstrate the detailed forecasting process. Afterwards, the results of a large scale comparison experiment are demonstrated. Specifically, on the performance of time-series segmentation, the three generic time-series segmentation algorithms, Sliding Window, Top-Down and Bottom-Up, are compared with our novel hybrid algorithm *K-MaxSDev*. On the performance of interval forecasting, 6 representative time-series forecasting strategies widely used in traditional workflow systems including LAST, MEAN, Exponential Smoothing (ES), Moving Average (MA), Autoregression (AR) and Network Weather Service (NWS) are compared with our statistical time-series pattern based interval forecasting strategy based on the samples for both long-duration activities and short-duration activities.



(a) 15 historical duration series



(b) The representative duration series

Fig. 5. Building duration series: (a) 15 historical duration series and (b) the representative duration series.

7.2.1. Example forecasting process for a data and computation intensive activity

Here, we demonstrate an example forecasting process for a data and computation intensive activity (long-duration activity) with each step addressed in Section 5.

(1) *Duration series building.* The first step is to build the duration series. Here, as shown in Fig. 5, we set the basic observation time unit as 15 min and the overall observation window as a typical working day of (8:00 am–8:00 pm).

As depicted in Fig. 5(a), with the period of a day, 15 duration series are built by random sampling without replacement in each observation unit. Here, null values are replaced by the unit sample means plus white noise (Law and Kelton, 2007), i.e. a random number with the mean of 0 and standard deviation of 1. Eventually, the representative duration series is built by the composition of the sample mean in each observation unit sequentially through the time. As depicted in Fig. 5(b), the representative duration series which is to be further analysed lies between the upper and the lower duration series which are composed of the maximum duration and the minimum duration of each observation unit respectively. In Fig. 5, the vertical axis stands for the activity durations with the basic time unit of second.

(2) *Duration pattern recognition.* In this step, we first conduct the  $K$ -MaxSDev time-series segmentation algorithm to discover the potential pattern set. Here, we need to assign the value for two parameters:  $K$  and MaxSDev. As explained in Section 4, the maximum repetition time  $R$  is set as a fixed value of 5 in our work since the repetition time in  $K$ -MaxSDev is below 3 in most cases. Based on our empirical function, we choose

the candidates of 2, 4, 8, and 12. With Formula (1) in Section 4, 4 is ranked the best. As for MaxSDev, based on Formula (2) in Section 4 with the overall sample mean  $\mu$  as 44.11 and having the candidates of  $m$  as 15, 20, 25, and 30, they are specified as 2.21, 2.94, 3.68 and 4.41 respectively. We conduct  $K$ -MaxSDev and compare with the three generic segmentation algorithms. With different MaxSDev, the number of segments for the four rounds of test is presented in Fig. 6. Evidently, with 11, 8, 7 and 5 as the minimum number of segments in each round of test,  $K$ -MaxSDev achieves the best performance in terms of discovering the smallest potential pattern set.

Moreover, with MaxSDev as 2.21 and Min.pattern\_length as 3, the result for segmentation and pattern validation is presented as an example in Table 2 where 8 valid patterns are identified from a total of 11 segments. The valid patterns and their associated turning points are listed in Table 3. Note that since our observation period is a whole working day, the turning point

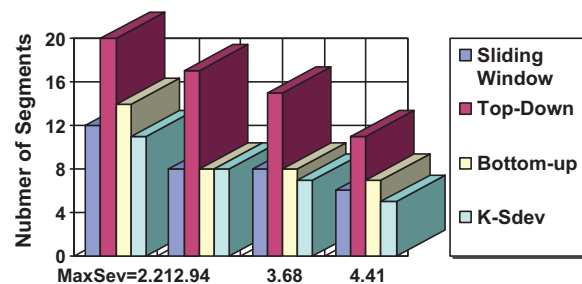


Fig. 6. Comparison with three generic segmentation algorithms.

**Table 2**  
Results for segmentation and pattern validation (long-duration activity).

	Segments description			Covered observation	Pattern validation
	Mean	SDev	Length		
Segment 1	34.7	2.05	10	(8:00–10:30 am)	Valid
Segment 2	39.58	0	1	(10:30–10:45 am)	Invalid
Segment 3	44.47	0.47	3	(10:45–11:30 am)	Valid
Segment 4	50.10	2.00	6	(11:30 am–1:00 pm)	Valid
Segment 5	59.81	2.10	6	(1:00–2:30 pm)	Valid
Segment 6	55.43	1.45	4	(2:30–3:30 pm)	Valid
Segment 7	50.75	1.51	7	(3:30–5:15 pm)	Valid
Segment 8	43.00	0.94	2	(5:15–5:45 pm)	Invalid
Segment 9	38.55	0	1	(5:45–6:00 pm)	Invalid
Segment 10	33.01	1.45	5	(6:00–7:15 pm)	Valid
Segment 11	24.88	1.49	3	(7:15–8:00 pm)	Valid

**Table 3**  
Pattern description and associated turning points.

Patterns	Patterns description			Concurrent activities
	Mean	SDev	Turning points	
Pattern 1	34.7	2.05	39.58	(2,4)
Pattern 2	44.47	0.47	46.98	(3,6)
Pattern 3	50.10	2.00	56.33	(4,8)
Pattern 4	59.81	2.10	56.35	(7,11)
Pattern 5	55.43	1.45	52.42	(6,10)
Pattern 6	50.75	1.51	43.00	(4,9)
Pattern 7	33.01	1.45	25.22	(2,5)
Pattern 8	24.88	1.49	31.99	(1,3)

of Pattern 8 is actually defined as the first value of Pattern 1 to make it complete as a cycle. Here, we also trace back the number of concurrent activities according to the observation units covered by each pattern. The result shows that when the average number of concurrent activities increases with the schedule of working hours, the sample means of activity durations also follow an increasing trend. However, they are evidently not in a linear relationship. Therefore, this observation indeed supports our motivation.

(3) *Duration pattern matching* and (4) *Duration interval forecasting*. Since these two steps are highly correlated, we demonstrate them together. Here, we randomly select 30 duration sequences from the system logs where their lengths range from 4 to 6 and the last value of each sequence is chosen as the target to be predicted. In our experiment, we adopt the normal distribution model to specify the duration intervals with a high confidence of 95%, i.e. a confidence percentile of 1.65 (Stroud, 2007). As shown in Fig. 7, for 25 cases, the target value lies within the predicted duration intervals, i.e. an accuracy rate of 83.3%. As for the rest of the 5 cases, the target values deviate only slightly from

the intervals. Therefore, the performance of interval forecasting is satisfactory.

Section 7.2.3 will demonstrate more test cases and the comparison results with other forecasting strategies on long-duration activities.

7.2.2. *Example forecasting process for an instance intensive activity*

As explained, the forecasting process for short-duration activities is a simplified version of the process for long-duration activities. Specifically, in the forecasting process for short-duration activities, there is no requirement for building representative duration series and pattern validation. For short-duration activities, only the duration series segmentation results by *K*-MaxSDev is required for the pattern matching and interval forecasting. Here, we demonstrate an example forecasting process for an instance intensive activity (short-duration activity) with each step addressed in Section 6.

Fig. 8 depicts the duration series of a short-duration activity which contains the latest 60 samples.

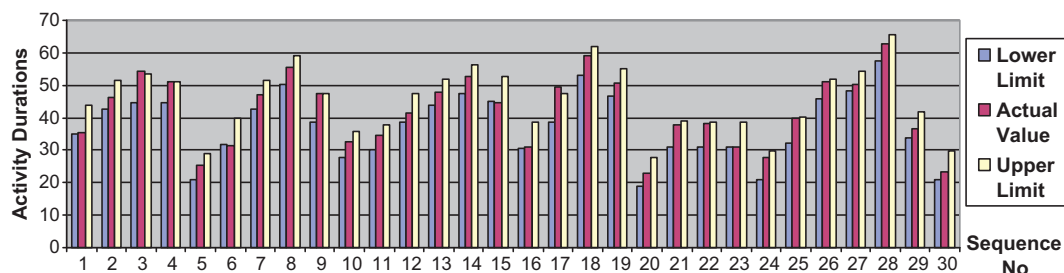


Fig. 7. Predicted duration intervals.

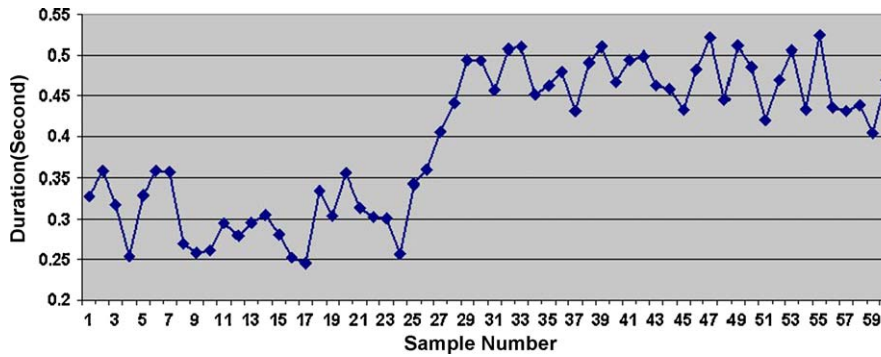


Fig. 8. Duration series of a short-duration activity.

- (1) *Duration pattern recognition.* Table 4 depicts the segmentation results. Similar to the example for the long-duration activity above, the mean of the duration series is 0.398, the initial  $K$  is 4, and we set  $m\%$  as 20%, hence we get the MaxSDev as 0.027 according to Formula (2). The segmentation results are presented in Table 4.
- (2) *Pattern matching* and (3) *duration interval forecasting.* Here, we aim to forecast the interval for the next point, i.e. the 61st point. The last segment, i.e. Segment 14, is the current duration sequence. Here, we set the confidence as 95%, i.e. the confidence percentile is 1.65 (Stroud, 2007). The MaxSDev is equal to 0.027, the length of Segment 13 is 1 and the length of Segment 14 is 5. Therefore, according to Algorithm 5, since the length of the current segment (i.e. Segment 14) is larger than that of its previous segment (i.e. Segment 13), the predicted interval for the next point should be defined as  $(0.436 - 1.65 \times 0.027, 0.436 + 1.65 \times 0.027) = (0.391, 0.481)$ .

Section 7.2.3 will demonstrate more test cases and the comparison results with other forecasting strategies on short-duration activities.

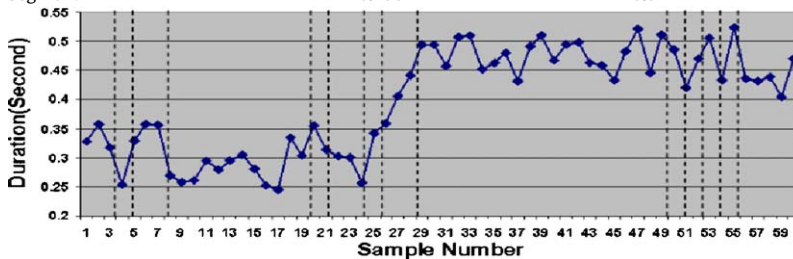
7.2.3. Comparison results

Besides real world examples, simulation experiments are conducted to evaluate the performance of our strategy in more general system environments. For the comparison purpose, test cases for 100 long-duration activities and 100 short-duration activities are generated. Meanwhile, for each independent long-duration or short-duration activity, 10 separate duration series are generated. Therefore, a total of 1000 simulated duration series for long-duration activities and another 1000 for short-duration activities are generated for our experiments. As mentioned earlier, the mean values for the simulated activity durations are specified according to real statistics but extended deliberately with different variances. These variances are randomly specified from a range of 5% to 30% of their corresponding mean values. Since 33% is the largest rate for normal distribution models according to the “ $3\sigma$ ” rule (Stroud, 2007), such settings can ensure that the test cases are able to represent general workflow applications.

Here, we first demonstrate the comparison results on the performance of time-series segmentation. The results for 100 duration series are randomly selected from the total of 2000 simulated duration series (either for long-duration or short-duration activities).

Table 4 Results for segmentation (short-duration activity).

Duration series	Mean = 0.398		
K-MaxSDev	MaxSDev = 0.027 (we set $m\%$ = 20% according to Formula (2))		
Segments	Segments description		
	Mean	SDev	Length
Segment 1	0.335	0.020	3
Segment 2	0.254	0	1
Segment 3	0.348	0.016	3
Segment 4	0.282	0.026	12
Segment 5	0.356	0	1
Segment 6	0.294	0.025	4
Segment 7	0.351	0.012	2
Segment 8	0.424	0.025	2
Segment 9	0.480	0.026	22
Segment 10	0.420	0	1
Segment 11	0.488	0.024	2
Segment 12	0.433	0	1
Segment 13	0.524	0	1
Segment 14	0.436	0.024	5



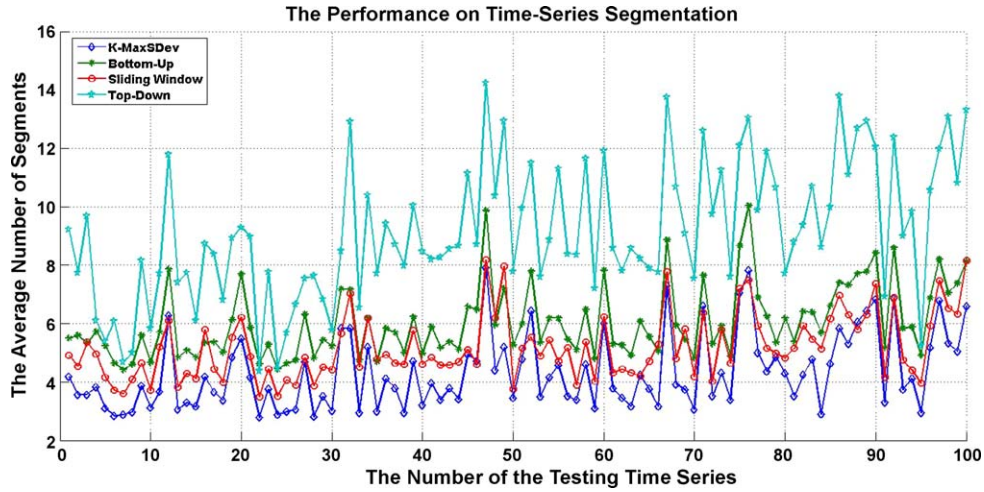


Fig. 9. Performance on time-series segmentation.

As discussed in Section 4, given the same testing criterion, i.e. with the maximum standard deviation, the smaller the number of segments, the better the segmentation performance is. The results are presented in Fig. 9.

As can be seen in Fig. 9, in most cases, our novel hybrid time-series segmentation algorithm *K*-MaxSDev behaves the best and achieves the minimal number of segments which build up the effective basis for time-series pattern recognition. Sliding Window ranks in the second place, but since it lacks the ability of looking backwards, it tends to over segment when the system performance is stable (i.e. the number of segments is small) such as the test cases of 11, 33, 48 and 53. Bottom-Up ranks the third place but since it lacks the ability of splitting, it tends to over segment when the system performance is dynamic such as on the test cases of 12, 20, 47 and 76. When the system performance is less dynamic (e.g. the number of segments is around 4–5 as depicted in Fig. 9), its performance is close to that of Sliding Window. Top-Down behaves the worst in most cases due to its nature for over-segmenting. Since the only action for Top-Down is splitting and it stops the segmentation process as long as the testing criterion is met, Top-Down ignores the probability of merging small segments into large ones. Therefore, Top-Down usually has more segments than others.

Before we demonstrate the comparison results on the accuracy of interval forecasting, we briefly review the 6 representative forecasting strategies and introduce the setting of their parameters. Here, the actual value of the  $k$ th point is denoted as  $Y_k$  and the predicted value for the  $k$ th point is denoted as  $P_k$ . The length of the latest duration sequence is randomly selected from 4 to 8.

- (1) LAST. LAST takes the last value of the time series as the predicted value for the next point. Therefore, the formula for LAST is defined as follows:

$$P_k = Y_{k-1} \quad (k \geq 1) \quad (3)$$

In the experiments, LAST takes the last point of the latest duration sequence as the predicted value.

- (2) MEAN. MEAN takes the mean value of all its previous points as the predicted value for the next point. Therefore, the formula for LAST is defined as follows:

$$P_k = \frac{1}{k} \sum_{i=0}^{k-1} Y_i \quad (k \geq 1) \quad (4)$$

In the experiments, MEAN takes the mean value of the entire time series where the latest duration sequence belongs to.

- (3) Exponential Smoothing (ES). ES makes the prediction for the next value based on both the actual value and the predicted value of its previous point. ES does not react directly to peaks while it reacts fast enough to significant changes of the mean values (e.g. the changes from one segment to another segment). The parameter  $\alpha$  depends on the characteristics of the dataset. The formula for ES is defined as follows:

$$P_k = Y_{k-1} + (1 - \alpha)P_{k-1} \quad (0 < \alpha < 1) \quad (5)$$

In the experiments, ES applies to the latest duration sequence with  $\alpha$  as 0.5 (a common choice given no priori knowledge).

- (4) Moving Average (MA). The generic MA predicts the next value based on the average value of the latest  $K$  points, denoted as  $MA(K)$ . There are also many other modified versions of MA such as the Weighted MA and Exponential MA (Dobber et al., 2007). The formula for the generic  $MA(K)$  is defined as follows:

$$P_{i+k} = \frac{1}{k} \sum_{j=i}^{i+k-1} Y_j \quad (i \geq 0, k \geq 1) \quad (6)$$

In the experiments, MA applies to the latest duration sequence and  $K$  is set randomly from 4 to 8, i.e. the average length of the duration sequences.

- (5) Autoregression (AR). AR multiplies previous data points with some parameters between 0 and 1 to predict the next value. These parameters can be deemed as weights assigned for each point where normally the closer the time, the larger the weight is. The formula for the generic  $AR(K)$  is defined as follows:

$$P_{i+k} = \alpha_i Y_i + \alpha_{i+1} Y_{i+1} + \dots + \alpha_{i+k-1} Y_{i+k-1} \quad \left( \sum_{j=i}^{j=i+k-1} \alpha_j = 1 \right) \quad (7)$$

In the experiments, the value of  $\alpha_i$  is defined as follows

$$\alpha_i = \begin{cases} 1 - \sum_{i=2}^K \alpha_i & , \quad i = 1 \\ \frac{1}{Dis^2 \times 0.9} & , \quad 2 \leq Dis \leq K \end{cases} \quad (8)$$

Here, the distance of the points, denoted as  $Dis$ , is defined based on the positions, i.e. for the last point in the latest duration sequence, its  $Dis$  is defined as 1, and for the second last, its  $Dis$  is

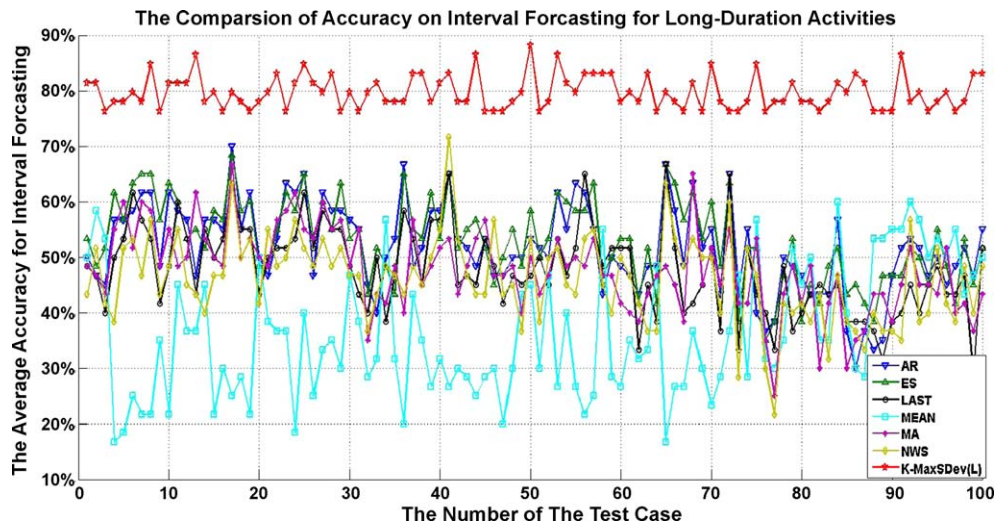


Fig. 10. Accuracy of interval forecasting for long-duration activities.

defined as 2, so on and so forth. Obviously, as defined in Formula (8), the weight of each point decreases as its distance increases.

- (6) Network Weather Service (NWS). NWS conducts postcasters using different windows of previous data and records the “winning” forecaster for each window size, i.e. the one with the minimal prediction errors on the last point afterwards, the predicted value will be the one predicted by the winning forecaster. The common forecasters included are Mean/Median, Moving Average/Median, LAST and ES. Therefore, the formula for the NWS can be described using the following formula:

$$P_k = P_k \{ \text{forecaster}_i \mid \text{Min}(\text{Err}(P_{k-1} \mid \text{forecaster}_i)) \} \quad (k \geq 1) \quad (9)$$

In the experiments, the forecasters included in NWS are all the other 5 forecasting strategies listed here. Therefore, based on the current performance, NWS dynamically selects the best forecasting strategies among LAST, MEAN, ES, MA, and AR.

The comparison results on the accuracy of interval forecasting for long-duration and short-duration activities are presented in Figs. 10 and 11 respectively. In this paper, since we focus on interval forecasting, the accuracy here means that the actual value is within the predicted interval.

The results for 100 long-duration activities (with the mean durations around 20–40 min) are demonstrated in Fig. 10. Since despite our *K*-MaxSDev, all the other forecasting algorithms are all dedicated to point forecasting, in the experiments, their upper bounds and lower bounds are defined as the predicted value plus 10% of itself and the predicted value minus 10% of itself respectively. Therefore, an interval with the length of 20% of the predicted value is created. The forecasting process for *K*-MaxSDev(L) is the same as the one demonstrated in Section 7.2.1. For each test case, we randomly select 30 duration sequences (with lengths range from 4 to 6) from the 10 duration series for the same activity and then choose the last value of each sequence as the target to be predicted. The accuracy of the test case is obtained from the average accuracy of the 10 duration sequences. For example, for the last value of each duration sequence, if 25 out of 30 of them are within the predicted intervals, then the accuracy of the test case is 25/30 = 83.3%. As can be seen in Fig. 10, our pattern based forecasting strategy (denoted as *K*-MaxSDev(L)) gains a large improvement on the average accuracy of interval forecasting for long-duration activities over the other representative forecasting strategies. The average accuracy of our *K*-MaxSDev(L) is 80% while that of the other strategies are mainly around 40–50%. Among the other 6 strategies, AR behaves

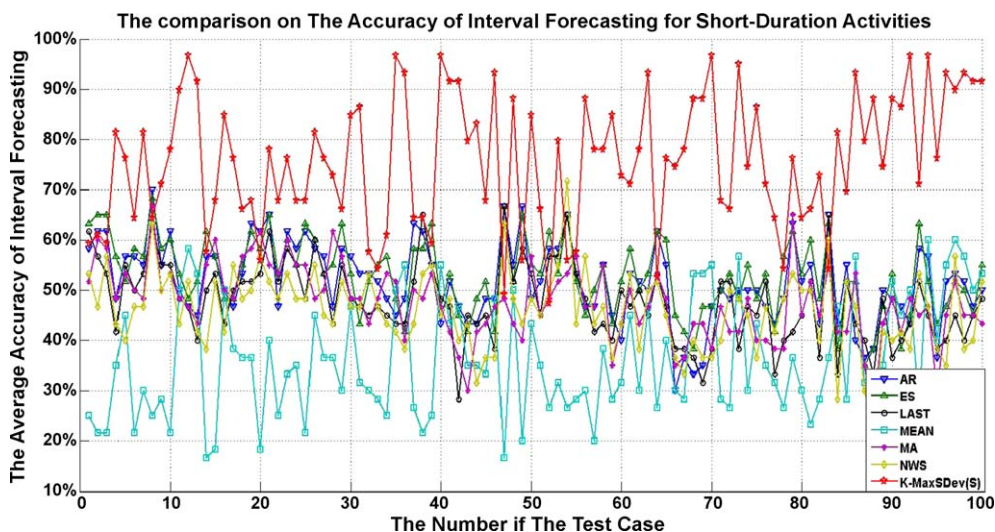


Fig. 11. Accuracy of interval forecasting for short-duration activities.

the best while the others have similar performance except MEAN which behaves poorly compared with the others. However, the results actually demonstrate the highly dynamic performance of the underlying services in workflow systems.

The results for 100 short-duration activities (with the mean durations around 0.2–0.5 s) are demonstrated in Fig. 11. The forecasting process of  $K$ -MaxSDev(S) is the same as the one demonstrated in Section 7.2.2 and the average accuracy is also obtained through the same process as  $K$ -MaxSDev(S).  $K$ -MaxSDev(S) behaves better than other strategies. The average accuracy of  $K$ -MaxSDev(S) is 76%, close to that of  $K$ -MaxSDev(L) which is 80%. However, it is less stable compared with the performance of  $K$ -MaxSDev(L). This is normal since the execution time for a short-duration activity is easily affected by the dynamic changes of system performance at the current moment and thus its duration is more difficult to be predicted than a long-duration activity. For all the other strategies, their performance on short-duration activities is actually very similar to that on long-duration activities since they do not differentiate these two types of activities.

To conclude, in comparison to other representative forecasting strategies in traditional workflow systems, our forecasting strategy can effectively adapt to both data/computation intensive scientific applications and instance intensive business applications in workflow systems. Specifically, our strategy can address the problems for the duration series of long-duration activities: limited sample size and frequent turning points, and meet the requirements for short-duration activities: efficient identification of current performance state and effective detection of possible changes.

## 8. Related work

Generally speaking, there are five major dimensions of workflow QoS constraints including time, cost, fidelity, reliability and security (Yu and Buyya, 2005a). In this paper, we focus on the dimension of time. In fact, since time is the basic measurement of performance, temporal QoS is one of the most important issues in many service oriented/enabled processes such as service composition (Zhao et al., 2004). Real world scientific or business processes normally stay in a temporal context and are often time constrained to achieve on-time fulfilment of scientific/business goals. Temporal correctness, i.e. whether a scientific (and business) workflow can be completed on time, is critical for the usefulness of workflow execution results (Chen and Yang, 2008). For example, timely completion of a scientific climate modelling workflow is a must so that the execution results are useful for weather forecast. Therefore, in order to maintain a satisfactory temporal QoS, many time related functionalities such as workflow scheduling and rescheduling (Yu and Buyya, 2006), temporal verification and validation (Chen and Yang, 2007), need to be implemented in workflow systems. However, no matter which functionality, one of its key requirements is an effective forecasting strategy for workflow activity durations. Besides, since accurate point forecasting, i.e. forecasting a specific value point, for activity durations is very difficult, if not impossible, in dynamic and complex scenarios such as computing environments, the forecasting of duration intervals, i.e. forecasting the upper bound and the lower bound of activity durations, is much more preferable than point forecasting (Law and Kelton, 2007). Furthermore, for the practical purpose, accurate estimation of upper bounds and lower bounds can help decision makers to evaluate the system performance states and design corresponding strategies.

In this paper, we focus on time-series based interval forecasting strategies. Five service performance estimation approaches including simulation, analytical modelling, historical data, online learning and hybrid, are presented in Yu and Buyya (2005a). In general, time-series forecasting belongs to historical data based approaches.

Currently, there is a lot of work which utilises multivariate models and focuses on the prediction of CPU load since it is the dominant factor for the durations of computation intensive activities. Typical linear time-series models such as AR, MA and Box-Jenkins fit to perform host load prediction in Dinda and O'Hallaron (2000) where it claims that most time-series models are sufficient for host load prediction and recommends AR(16) which consumes miniscule computation cost. The work in Yang et al. (2003) proposes a one-step-ahead and low-overhead time-series forecasting strategy which tracks recent trends by giving more weight to recent data. A hybrid model which integrates the AR model with confidence interval is proposed in Wu et al. (2007) to perform  $n$ -step-ahead load prediction. A different strategy proposed in Smith et al. (2004) predicts application duration with historical information where search techniques are employed to determine those application characteristics that yield the best definition of similarity. The work in Akioka and Muraoka (2004) investigates extended forecast of both CPU and network load on computation grid to achieve effective load balancing. The authors in Wu et al. (2007) mention the problem of turning points which causes large prediction errors to time-series models. However, few solutions have been proposed to investigate more affecting factors and handle turning points. Different from the above work, there are a few literatures utilise univariate models. NWS (Wolski, 1997) implements three types of univariate time-series forecasting methods including mean-based, median-based and autoregressive methods, and it dynamically chooses the one exhibiting the lowest cumulative error measure at any given moment to generate a forecast. Hence, NWS automatically identifies the best forecasting methods for any given resource. The work in Dobber et al. (2007) proposes a Dynamic Exponential Smoothing (DES) method which forecasts the future duration only based on its past values. The basic motivation for DES is to handle the problems of sudden peaks and level switches. DES computes the optimal parameter for different classes of fluctuations and it can consistently outperform the common prediction methods, such as the NWS predictor, AR(16), and ES (Dobber et al., 2007) in a grid environment. However, to the best of our knowledge, no existing work has been proposed so far to systematically address the forecasting of long-term as well as short-term durations of activities in workflow systems.

Time-series patterns are those time-series segments which have high probability to repeat themselves (similar in the overall shape but maybe with different amplitudes and/or durations) in a large time-series dataset (Chatfield, 2004; Lai et al., 2004; Huang et al., 2007; Liu et al., 2008b). Time-series patterns can be employed for complex prediction tasks, such as the trend analysis and value estimation in stock market, product sales and weather forecast (Han and Kamber, 2006). The two major tasks in pattern based time-series forecasting are pattern recognition which discovers time-series patterns according to the pattern definition, and pattern matching which searches for the similar patterns with given query sequences. To facilitate complex time-series analysis such as time-series clustering and fast similarity search, time-series segmentation is often employed to reduce the variances in each segment so that they can be described by simple linear or non-linear models (Keogh et al., 2001). In general, most segmentation algorithms can be classified into three generic algorithms with some variations which are Sliding Windows, Top-Down and Bottom-Up. The process of Sliding Window is like sequential scanning where the segments keep increasing with each new point until some thresholds are violated. The basic idea of the Top-Down algorithm is the repetition of the splitting process where the original time-series is equally divided until all segments meet the stopping criteria. In contrast to Top-Down, the basic process of the Bottom-Up algorithm is merging where the finest segments are merged continuously until some thresholds are violated. The main prob-

lem for Sliding Window is that it can only “look forward” but not “look backward”. The main problem for Top-Down is that it lacks the ability of merging while the main problem for Bottom-Up is that it lacks the ability of splitting. Therefore, hybrid algorithms which take advantages of the three generic algorithms by modifications and combinations are often more preferable in practice (Han and Kamber, 2006; Keogh et al., 2001; Tirthapura et al., 2006).

## 9. Conclusions and future work

Interval forecasting for activity durations in workflow systems is of great importance since it is related to most of workflow QoS and non-QoS functionalities such as load balancing, workflow scheduling and temporal verification. However, to predict accurate duration intervals is very challenging due to the dynamic nature of computing infrastructures. Most of the recent studies focus on the prediction of CPU load to facilitate the forecasting of computation intensive activities. Meanwhile, another problem which affects the performance of existing forecasting strategy is that they do not differentiate long-duration activities in data/computation intensive scientific applications from short-duration activities in instance intensive business applications which are the two major types of workflow applications. In fact, the durations of long-duration activities are normally dominated by the average performance of the workflow system over their lifecycles while the durations of short-duration activities are mainly dependent on the system performance at the current moment where the activities are being executed. Therefore, the forecasting strategies in workflow systems should be able to adapt to different workflow applications.

In this paper, rather than fitting conventional linear time-series models in this paper which are not ideal, we have investigated the idea of pattern based time-series forecasting. Specifically, based on a novel non-linear time-series segmentation algorithm named  $K$ -MaxSDev, a statistical time-series pattern based forecasting strategy which consists of four major functional components: duration series building, duration pattern recognition, duration pattern matching, and duration interval forecasting, have been proposed. Meanwhile, two different versions of the strategy have been proposed to facilitate the interval forecasting for long-duration activities ( $K$ -MaxSDev(L)) and short-duration activities ( $K$ -MaxSDev(S)).  $K$ -MaxSDev(L) includes an offline time-series pattern discovery process and an online forecasting process while  $K$ -MaxSDev(S) conducts both the pattern discovery and the interval forecasting in an online fashion. The simulation experiments conducted in our SwinDeW-G workflow system have demonstrated that our time-series segmentation algorithm is capable of discovering the smallest potential pattern set compared with three generic algorithms: Sliding Windows, Top-Down and Bottom-Up. The large scale comparison results have further demonstrated that our statistical time-series pattern based strategy behaves better than other 6 representative time-series forecasting strategies including LAST, MEAN, Exponential Smoothing, Moving Average, Autoregression and Network Weather Service in the prediction of high confidence duration intervals and the handling of turning points. To the best of our knowledge, this is the first paper that has systematically investigated the problems for interval forecasting of activity durations in workflow systems and proposed with a solution of statistical time-series pattern based forecasting strategy.

In the future, some more sophisticated hybrid time-series forecasting strategies will be investigated and compared with our pattern based forecasting strategy. Meanwhile, some existing strategies which can efficiently detect system performance changes may be utilised to further improve the detection of turning points in the duration series. We also plan to investigate the strategies for

multi-step-ahead prediction which can achieve satisfactory accuracy in workflow systems.

## Acknowledgments

This work is partially supported by Australian Research Council under Linkage Project LP0990393, the National Natural Science Foundation of China project under Grant No. 70871033.

## References

- Akioka, S., Muraoka, Y., 2004. Extended forecast of CPU and network load on computational Grid. In: Proc. IEEE International Symposium on Cluster Computing and the Grid, pp. 765–772.
- Barga, R., Gannon, D., 2007. Scientific versus business workflows. *Workflows for e-Science*, 9–16.
- 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.
- Chatfield, C., 2004. *The Analysis of Time Series: An Introduction*, Sixth Edition. Chapman and Hall/CRC.
- 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., 2009. Temporal dependency based checkpoint selection for dynamic verification of temporal constraints in scientific workflow systems. *ACM Transactions on Software Engineering and Methodology*, <http://www.ict.swin.edu.au/personal/yyang/papers/ACM-TOSEM-checkpoint.pdf>.
- Chen, J., Yang, Y., 2010. Localising temporal constraints in scientific workflows. *Journal of Computer and System Sciences* 76 (6), 464–474.
- 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.
- Dinda, P.A., O'Hallaron, D.R., 2000. Host load prediction using linear models. *Cluster Computing* 3 (4), 265–280.
- Dobber, M., van der Mei, R., Koole, G., 2006. Statistical properties of task running times in a global-scale grid environment. *Proc. Sixth IEEE International Symposium on Cluster Computing and the Grid* 1, 150–153.
- Dobber, M., van der Mei, R., Koole, G., 2007. A prediction method for job runtimes on shared processors: survey, statistical analysis and new avenues. *Performance Evaluation* 64 (7–8), 755–781.
- Glaser, C., Volkert, J., in press. Adapts—a three-phase adaptive prediction system for the run-time of jobs based on user behaviour. *Journal of Computer and System Sciences*.
- GridBus Project, <http://www.gridbus.org>, accessed on 1st August 2010.
- Hadoop, <http://hadoop.apache.org/>, accessed on 1st August 2010.
- Han, J.W., Kamber, M., 2006. *Data Mining: Concepts and Techniques*, Second Edition. Elsevier.
- Hsu, H.J., Wang, F.J., 2008. An incremental analysis for resource conflicts to workflow specifications. *Journal of Systems and Software* 81 (October (10)), 1770–1783.
- Huang, Y., Hsu, C., Wang, S., 2007. Pattern recognition in time series database: a case study on financial database. *Expert Systems with Applications* 33 (1), 199–205.
- Kepler Project, <http://kepler-project.org/>, accessed on 1st August 2010.
- Keogh, E., Chu, S., Hart, D., Pazzani, M., 2001. An online algorithm for segmenting time series. In: Proc. 2001 IEEE International Conference on Data Mining, pp. 289–296.
- Lai, C.F., Chung, F.T., Ng, V., Luk, R.W.P., 2004. An evolutionary approach to pattern-based time series segmentation. *IEEE Transactions on Evolutionary Computation* 8 (5), 471–489.
- Law, A.M., Kelton, W.D., 2007. *Simulation Modelling and Analysis*, Fourth Edition. McGraw-Hill.
- Li, H., Yang, Y., Chen, T.Y., 2004. Resource constraints analysis of workflow specifications. *Journal of Systems and Software* 73 (2), 271–285.
- Liu, K., Chen, J., Yang, Y., Jin, H., 2008a. A throughput maximization strategy for scheduling transaction-intensive workflows on SwinDeW-G. *Concurrency and Computation: Practice and Experience* 20 (5), 1807–1820.
- Liu, X., Chen, J., Liu, K., Yang, Y., December 2008. Forecasting duration intervals of scientific workflow activities based on time-series patterns. In: Proc. 4th IEEE International Conference on e-Science, Indianapolis, USA, pp. 23–30.
- Liu, X., Chen, J., Yang, Y., September 2008. A probabilistic strategy for setting temporal constraints in scientific workflows. In: Proc. 6th International Conference on Business Process Management, Milan, Italy, pp. 180–195.
- Liu, X., Ni, Z., Chen, J., Yang, Y., 2010a. A probabilistic strategy for temporal constraint management in scientific workflow systems. *Concurrency and Computation: Practice and Experience*, Wiley, <http://www.ict.swin.edu.au/personal/yyang/papers/CCPE-ConstraintManagement.pdf>, accessed on 1st August.
- Liu, X., Yuan, D., Zhang, G., Chen, J., Yang, Y., 2010b. SwinDeW-C: a peer-to-peer based cloud workflow system. In: Furht, B., Escalante, A. (Eds.), *Handbook of Cloud Computing*. Springer.

- 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*, <http://www.ict.swin.edu.au/personal/yyang/papers/IJHPCA2010.pdf>, accessed on 1st August.
- Martinez, A., Alfaro, F.J., Sanchez, J.L., Quiles, F.J., Duato, J., 2007. A new cost-effective technique for QoS support in clusters. *IEEE Transactions on Parallel and Distributed Systems* 18 (12), 1714–1726.
- Nadeem, F., Fahringer, T., 2009a. Predicting the execution time of grid workflow applications through local learning. In: *Proc. 2009 ACM Conference on High Performance Computing Networking, Storage and Analysis*, Portland, Oregon, pp. 1–12.
- Nadeem, F., Fahringer, T., 2009b. Using templates to predict execution time of scientific workflow applications in the grid. In: *Proc. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 316–323.
- Nadeem, F., Yousaf, M., Prodan, R., Fahringer, T., 2006. Soft benchmarks-based application performance prediction using a minimum training set. In: *Proc. the Second IEEE International Conference on e-Science and Grid Computing*, p. 71.
- Neng, C., Zhang, J., 2009. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 39 (1), 29–43.
- National Meteorological Data Centre, 2010. China Meteorological Data Sharing Service System, <http://cdc.cma.gov.cn/index.jsp>, accessed on 1st August.
- Prodan, R., Fahringer, T., 2008. Overhead analysis of scientific workflows in grid environments. *IEEE Transactions on Parallel and Distributed Systems* 19 (March (3)), 378–393.
- SAP Library, 2010. Workflow System Administration, [http://help.sap.com/saphelp\\_nw2004s/helpdata/en](http://help.sap.com/saphelp_nw2004s/helpdata/en), accessed on 1st August.
- Smith, W., Foster, I., Taylor, V., 2004. Predicting application run times with historical information. *Journal of Parallel Distributed Computing* 64 (9), 1007–1016.
- Son, J., Kim, M.H., 2001. Improving the performance of time-constrained workflow processing. *Journal of Systems and Software* 58 (3), 211–219.
- Stavrinides, G.L., Karatza, H.D., 2010. Scheduling multiple task graphs with end-to-end deadlines in distributed real-time systems utilizing imprecise computations. *Journal of Systems and Software* 83 (6), 1004–1014.
- Stroud, K.A., 2007. *Engineering Mathematics*, Sixth Edition. Palgrave Macmillan, New York.
- Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M., 2007. *Workflows for e-Science: Scientific Workflows for Grids*. Springer.
- Tirthapura, S., Xu, B., Busch, C., 2006. Sketching asynchronous streams over a sliding window. In: *Proc. the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, Denver, CO.
- van der Aalst, W.M.P., Hee, K.M.V., 2002. *Workflow Management: Models, Methods, and Systems*. The MIT Press, Cambridge.
- VMware, <http://www.vmware.com/>, accessed on 1st August 2010.
- Wang, L., Jie, W., Chen, J., 2009. *Grid Computing: Infrastructure, Service, and Applications*. CRC Press, Taylor & Francis Group.
- Wolski, R., 1997. Forecasting network performance to support dynamic scheduling using the network weather service. In: *Proc. 1997 IEEE International Symposium on High Performance Distributed Computing*, pp. 316–325.
- Wu, W., Yuan, Y., Yang, G., Zheng, W., 2007. Load prediction using hybrid model for computational grid. In: *Proc. 2007 IEEE/ACM International Conference on Grid Computing*, pp. 235–242.
- Yan, J., Yang, Y., Raikundalia, G.K., 2006. SwinDeW-a p2p-based decentralized workflow management system. *IEEE Transactions on Systems, Man and Cybernetics, Part A* 36 (5), 922–935.
- Yang, Y., Foster, I., Schopf, J.M., 2003. Homeostatic and tendency-based CPU load predictions. In: *Proc. 2003 International Parallel and Distributed Processing Symposium*, p. 9.
- Yang, Y., Liu, K., Chen, J., Lignier, J., Jin, H., December 2007. Peer-to-Peer based grid workflow runtime environment of SwinDeW-G. In: *Proc. 3rd International Conference on e-Science and Grid Computing*, Bangalore, India, pp. 51–58.
- Yang, Y., Liu, K., Chen, J., Liu, X., Yuan, D., Jin, H., 2008. An algorithm in SwinDeW-C for scheduling transaction-intensive cost-constrained cloud workflows. In: *Proc. 4th IEEE International Conference on e-Science*, Indianapolis, USA, December, pp. 374–375.
- Yu, J., Buyya, R., 2005a. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing* (3), 171–200.
- Yu, J., Buyya, R., 2005b. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record* 34 (3), 44–49.
- Yu, J., Buyya, R., 2006. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming* 14 (December (3.4)), 217–230.
- Yuan, D., Yang, Y., Liu, X., Chen, J., 2010. A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems* 26 (6), 1200–1214.
- Zeng, Q.T., Wang, H.Q., Xu, D.M., Duan, H., Han, Y.B., 2008. Conflict detection and resolution for workflows constrained by resources and non-determined durations. *Journal of Systems and Software* 81 (September (9)), 1491–1504.
- Zhang, Y., Sun, W., Inoguchi, Y., 2008. Predict task running time in grid environments based on CPU load predictions. *Future Generation Computer Systems* 24 (6), 489–497.
- Zhao, Z., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H., 2004. QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering* 30 (5), 311–327.
- Zhuge, H., Cheung, T., Pung, H., 2001. A timed workflow process model. *Journal of Systems and Software* 55 (3), 231–243.



**Xiao Liu** received his master degree in management science and engineering from Hefei University of Technology, Hefei, China, 2007. He is currently a PhD student in Centre for Complex Software Systems and Services in the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Australia. His research interests include workflow management systems, scientific workflow, business process management and data mining.



**Zhiwei Ni** received his master degree from the Department of Computer Science and Engineering, Anhui University, Hefei, China, 1991 and a PhD degree from the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2002, all in computer science. He is currently a full Professor in the School of Management and also the Director for the Institute of Intelligent Management in Hefei University of Technology, Hefei, China. His major research interests include Artificial Intelligence, Machine Learning, Intelligent Management and Intelligent Decision-making Techniques.



**Dong Yuan** was born in Jinan, China. He received the B.Eng. degree in 2005 and M.Eng. degree in 2008 both from Shandong University, Jinan, China, all in computer science. He is currently a PhD student in the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Vic., Australia. His research interests include data management in workflow systems, scheduling and resource management, grid and cloud computing.



**Yuanchun Jiang** received his bachelor degree in management science and engineering from Hefei University of Technology, Hefei, China. He is a PhD student in Institute of Electronic Commerce in the School of Management at Hefei University of Technology. He is currently a visiting PhD student in the Joseph M. Katz Graduate School of Business at University of Pittsburgh. His research interests include decision science, electronic commerce and data mining. He has published papers in journals such as *Decision Support Systems*, *Expert System with Applications*, and *Knowledge-Based Systems*.



**Zhangjun Wu** received his master degree in Software Engineering from University of Science and Technology of China in 2005, Hefei, China. He is currently a Ph.D. student in Hefei University of Technology, Hefei, China. From March to September, 2010, he has been visiting the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Australia. His research interests include evolutionary algorithms, workflow scheduling and cloud computing.



**Jinjun Chen** received his Ph.D. degree in Computer Science and Software Engineering from Swinburne University of Technology, Melbourne, Australia in 2007. He is currently a Lecturer in Centre for Complex Software Systems and Services in the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Australia. His research interests include: Scientific Workflow Management and Applications, Workflow Management and Applications in Web Service or SOC Environments, Workflow Management and Applications in Grid (Service)/Cloud Computing Environments, Software Verification and Validation in Workflow Systems,

QoS and Resource Scheduling in Distributed Computing Systems such as Cloud Computing, Service Oriented Computing (SLA, Negotiation, Engineering, Composition), Semantics and Knowledge Management, Cloud Computing.



**Yun Yang** was born in Shanghai, China. He received a Master of Engineering degree from The University of Science and Technology of China, Hefei, China, in 1987, and a PhD degree from The University of Queensland, Brisbane, Australia, in 1992, all in computer science. He is currently a full Professor and Associate Dean (Research) in the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Australia. Prior to joining Swinburne as an Associate Professor in late 1999, he was a Lecture and Senior Lecturer at Deakin University during 1996–1999. Before that, he was a Research Scientist at DSTC-Cooperative Research Centre for Distributed Systems Technology during 1993–1996.

He also worked at Beihang University in China during 1987–1988. He has published more than 160 papers on journals and refereed conferences. His research interests include software engineering; p2p, grid and cloud computing based workflow systems; service-oriented computing; Internet computing applications; and CSCW.