Adaptive Temporal Verification and Violation Handling for Time-Constrained Business Cloud Workflows

Haoyu Luo[1,2], Xiao Liu[3], Jin Liu[1](✉), Bo Han[1], and Yun Yang[4]

[1] School of Computer Science, Wuhan University, Wuhan, China
{luohy, jinliu, bhan}@whu.edu.cn
[2] School of Computer Science, South China Normal University, Guangzhou, China
[3] School of Information Technology, Deakin University, Geelong, Australia
xiao.liu@deakin.edu.au
[4] School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia
yyang@swin.edu.au

Abstract. To achieve on-time completion of time-constrained business cloud workflows, a large number of parallel cloud workflow instances need to be constantly monitored so that temporal violations (namely intermediate runtime delays) can be detected and handled timely. Over the last few years, many strategies have been proposed but they are not adaptive enough to capture the dynamic behaviors of business cloud workflows. In this paper, we introduce the idea of "adaptiveness" into our strategy design. Specifically, we first present an adaptive temporal checkpoint selection strategy where the time intervals between checkpoints are adaptively determined at runtime, and then propose a matching temporal violation handling strategy which can determine the required lifecycle of cloud services. The evaluation results demonstrate that our adaptive strategy can achieve both higher efficiency and better cost effectiveness compared with conventional strategies.

Keywords: Temporal verification · Violation handling · Business workflow Adaptiveness · Cloud computing

1 Introduction

Business workflow can provide partial or even full automation of business processes in the domains of such as e-business and e-government. A notable feature of business workflow is that there is usually a large number of workflow instances running in a parallel fashion triggered by a large amount of concurrent user requests. To ensure the scalability in processing parallel workflow instances, a rapidly increasing number of business workflow applications are being deployed into the Cloud.

In business scenarios, a time-constrained workflow application needs to provide timely response to business requests [1]. Failing to deliver requested results in time may lead to the deterioration of user satisfaction, even huge financial loss. Therefore, the correctness of a business workflow application depends not only on its logical correctness, but also on its temporal correctness [2]. However, due to the dynamic nature and uncertainties that exist during the running of workflows in the cloud, temporal violations often occur which may have a serious impact on on-time completion of workflow instances. “temporal violation” means an intermediate violation of time constraints during workflow execution that can be fixed locally to achieve overall timely completion.

To ensure temporal correctness, workflow temporal behaviors need to be constantly monitored at runtime so that temporal violations can be timely detected and handled [3, 4]. When dealing with the monitoring and verification of temporal behaviors for a large number of parallel business workflow instances, throughput has been proved to be a competent measurement for the requirements of efficiency and scalability [5]. The basic idea is to select a set of time points along execution timeline as checkpoints, at which temporal behavior (represented by workflow throughput) is verified to determine whether a temporal violation occurs or not. If a temporal violation is detected, the current temporal behavior needs to be adjusted by temporal violation handling strategies.

To handle temporal violations, a temporal verification strategy and a temporal violation handling strategy are required to address the problem of “Where” and “How” respectively. In recent years, many related approaches have been proposed. However, a common problem is that they are not adaptive enough to capture the dynamic behaviors of business cloud workflows. To detect temporal violations, existing strategies verify temporal behavior at a set of temporal checkpoints which are selected from predefined system time points before workflow execution. These time points are usually equally distributed and the fixed time intervals between them are empirically set. This is inefficient as temporal behaviors constantly fluctuate at workflow runtime. In the meantime, most approaches handle the detected temporal violations by adding new resources. But the current work simply adds new resources with fixed lifecycles once a temporal violation is detected, in regardless of workflow runtime temporal behavior. Such kind of static approach may cause unnecessary cost of resource consumption.

For such an issue, we introduce the idea of “adaptiveness” into our strategy design. Two adaptive strategies are proposed to answer the questions of “Where” and “How” to handle temporal violations respectively. Specifically, we first present an adaptive temporal checkpoint selection strategy for the question of “Where”. Instead of using fixed time intervals, time interval between adjacent checkpoints is adaptive to workflow execution states. This strategy is more sensitive to the variation of temporal behaviors, which can improve the efficiency of temporal verification and decrease unnecessary resource consumption. Then we present a matching temporal violation handling strategy to answer the question of “How”. This strategy is designed to reduce or eliminate time delays by accelerating the execution of workflow activities with extra resources, e.g. service instances, provisioned to the cloud service nodes where temporal violations are detected. Specifically, it addresses three major issues for the question of “How”: (1) where to add extra resources; (2) how many resources are needed; (3) how long their lifecycles should be. Experimental results show that our adaptive strategy can achieve the target on-time completion rate with higher verification efficiency and at least 19.4% less resource consumption compared with conventional strategies.

The rest of the paper is organized as follows. Section 2 presents preliminary definitions. Section 3 presents the adaptive temporal verification and violation handling strategies. Section 4 demonstrate the experimental results. Section 5 concludes this paper.

2 Preliminary

(1) Workflow throughput

A business workflow is made up of a set of activities in partial order. We denote the i th activity of a business workflow as a_i. The mean, expected and runtime completion duration of a_i is denoted as $M(a_i)$, $E(a_i)$ and $R(a_i)$ respectively. Accordingly, WF_i is a workflow with its mean, expected and runtime completion duration denoted as $M(WF_i)$, $E(WF_i)$ and $R(WF_i)$ respectively.

Definition 1 (Workflow Throughput). Given a batch of q business workflow instances $WF\{WF_1, WF_2, \ldots, WF_q\}$ which starts at system time S_0, the completion of workflow activity a_{ij} contributes to the completion of the entire collection of workflows with a value of $M(a_{ij})/T$ where $\mathrm{T} = \sum_{i=1}^{q} M(WF_i)$. Here, we assume that at the current observation time point S_t, the set of new completed activities from the preceding nearest observation time point S_{t-1} is denoted as $a\{\}|_{S_{t-1}}^{S_t}$, then the system throughput is defined as $TH|_{S_{t-1}}^{S_t} = M(a\{\}|_{S_{t-1}}^{S_t})/T$.

Workflow throughput constraints is the expected accumulated workflow throughput (namely the percentage of completion) that should be achieved by a specific system time point. The value is decided by the throughput deadline assignment strategy [6].

(2) Queueing model for cloud services

At workflow runtime, a large number of instances are initialized in a short time. Since the number of parallel workflow instances is much more than the dedicated cloud services, workflow activities have to queue up on limited services. In this paper, queueing model is employed to depict the queueing and execution process of parallel workflow activities. We employ M/G/m/m+r model to formulate the behavior of the first queueing system and G/G/m/m+r model for the rest $k-1$ queueing systems [7]. The discussion for the rationale of the model design can be found at our previous work [8].

(3) Throughput-based temporal consistency model

Temporal verification requires a temporal consistency model which defines the relationship between the current workflow execution state and target deadline.

Definition 2 (Throughput Consistency Model). Given the same batch of workflows in Definition 1 and its final deadline $F(WF)$, at throughput checkpoint S_p, it is said to be of $\alpha\%$ consistency if:

$$F(\lambda_\alpha) = TH|_{S_0}^{S_p} + Exp\left(TH|_{S_p}^{F(WF)}\right) \quad (1)$$

where λ_α is defined as the $\alpha\%$ confidence percentile with the cumulative standard normal distribution function of $F(\mu_i + \lambda\sigma_i) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\mu_i + \lambda\sigma_i} e^{-(x-\mu_i)^2/2}\sigma_i^2 dx = \alpha\%$. $TH|_{S_0}^{S_p}$ is the current runtime throughput until S_p, $Exp\left(TH|_{S_p}^{F(WF)}\right)$ is the expected workflow throughputs during the time between the checkpoint and final deadline. $\alpha\%$ consistency is a probability confidence for on-time completion.

3 Adaptive Temporal Verification and Violation Handling

3.1 Adaptive Temporal Checkpoint Selection Strategy

Definition 3 (Candidate Throughput Checkpoints). Given the same batch of workflow instances as in Definition 1, a system time point S_t along the workflow execution timeline is a candidate throughput checkpoint if $S_t - S_{t-1} = k * bt$ (k = 1, 2, 3, …).

Algorithm 1. Throughput consistency verification strategy.

```
Require:   Target on-time completion rate $\theta\%$
           A collection of candidate checkpoints $S_p$ ($0 \le p \le n$)
Ensure:    A set of temporal checkpoints
           Throughput consistency states at checkpoints
Step1:     Temporal checkpoint selection
1:         Deficit=0; Redundancy=0; k=1; p=1;
2:         While p ≤ n
3:             If $TH|_{S_{p-k}}^{S_p} < THCons|_{S_{p-k}}^{S_p}$
4:                 Deficit= $THCons|_{S_{p-k}}^{S_p} - TH|_{S_{p-k}}^{S_p}$;
5:                 If Deficit < Redundancy
6:                     If k>1
7:                         k=⌊k/2⌋;
8:                     End if
9:                 Else
10:                    $S_p$ is selected as a checkpoint, continue to Step 2;
11:                    If $\alpha\% \ge \theta\%$
12:                        Break;
13:                    Else
14:                        k=1; Redundancy=0; Report a temporal violation;
15:                    End if
16:                End if
17:            Else
18:                k=k+1; Redundancy= $TH|_{S_{p-k}}^{S_p} - THCons|_{S_{p-k}}^{S_p}$;
19:            End if
20:            p=p+k;
21:        End while
Step 2:    Temporal verification
           Calculate workflow throughput:
             (1) the completed workflow throughput until $S_p$.
             (2) the expected remaining throughput from $S_p$ to $F(WF)$.
           Given the throughput consistency model depicted in Definition 2, calculate
           throughput consistency state $\alpha\%$ at checkpoint $S_p$
```

$k * bt$ is the time interval between adjacent candidate throughput checkpoints.bt is the minimum time interval for system monitoring. k is a variable which is decided at workflow runtime. Checkpoint selection strategy needs to figure out candidate throughput checkpoints first, then determines whether they should be selected as a checkpoint.

Algorithm 1 depicts the adaptive checkpoint selection strategy, which selects checkpoints one by one at workflow runtime. Whether a candidate checkpoint is selected as checkpoint or not depends on both the temporal behavior during the latest monitoring window $k * bt$ and the temporal behaviors of several prepositive checkpoints. Temporal consistency state of the newly selected checkpoint influences the decision of the next checkpoint. To reflect such a feedback and adjustment mechanism, we integrate the proposed checkpoint selection strategy (Step 1) with throughput-based temporal consistency model (Step 2) as an overall throughput consistency verification approach.

3.2 Temporal Violation Handling Strategy

Once a temporal violation is detected at a checkpoint, temporal violation handling strategy will be triggered to deal with the recovery of violation by accelerating the workflow execution after the current checkpoint. In the scenario of business cloud workflows, if a throughput violation is reported, temporal violation handling strategy needs to increase the system throughput in a short period of time.

We design to handle temporal violations by recruiting extra resources (namely adding more cloud service instances in queueing systems). The extra resources for violation handling are recruited only for temporary use. Specifically, our proposed temporal violation handling strategy needs to answer three major questions:

(1) **Where to add resources?**

Queueing system for a cloud service is regarded as a basic unit for violation handling. Therefore, new resources are added into the queueing systems where local throughput constraints are violated and these queueing systems are called violation handling points.

Algorithm 2 (Step 1) explains the violation handling point selection strategy. When a throughput violation is detected at checkpoint S_p (line 1), we need to calculate the average response time (ART) of workflow activities in each queueing system during S_{p-1} and S_p (lines 2–3). If the ART of a queueing system exceeds the response time constraint (RTC), the local throughput constraint will be violated inevitably, and this queueing system is selected as a handling point (lines 4–5).

(2) **How many resources are needed?**

Adding extra service instances into a queueing system can increase the throughput of cloud services and reduce the average response time of workflow activities. Here, an essential question aroused is "to timely and completely handle temporal violations, does adding more extra resources mean better effectiveness?" Our answer is "not always".

Since it is impossible to obtain a closed formula to represent the probability distribution of general distribution "G" in M/G/m/m+r and G/G/m/m+r models, the numerical relationship between the number of service instances and average response time is not clear. Therefore, we conduct testing experiments and use the results as reference.

Algorithm 2. Temporal violation handling strategy

```
Require:  – Target on-time completion rate $\theta\%$.
          – Throughput consistency state $\alpha\%$ at checkpoint $S_p$.
          – Response time constraint (RTC) of each kind of activity.
          – Time interval $k * bt$ bewteen checkpoint $S_p$ and candidate checkpoint $S_{p+1}$,
            which is achieved by the adaptive checkpoint selection strategy.
Ensure:   – Add new resources with proper lifecycles into selected handling points.
Step 1:   Temporal violation handling point selection
1:        If $\alpha\% \leq \theta\%$
2:            For i=1:n
3:                Calculate average response time (ART) of activities in each QS;
4:                If $ART_i > RTC_i$
5:                    $QS_i$ is selected as a handling point;
6:                    Continue to Step 2;
7:                End if
8:                i=i+1;
9:            End for
10:       End if
Step 2:   Violation handling at handling points
11:       If newResource=false
12:           Add one extra resource with basic lifecycle L into $QS_i$
13:       Else
14:           Check residual life (RL) of the new resource;
15:           If RL < $k * bt$
16:               RL=RL+L;            //append a basic lifecycle
17:           End if
18:       End if
```

We conduct two rounds of experiments to figure out how response time changes with the number of cloud service instances in queueing systems with the above two different queueing models. In the two queueing systems, the average execution time of workflow activities is 6.1729 s and 13.5686 s. The minimum numbers of service instances based on queueing rule are 6 and 12 (at the worst level of service). When we add the first service instance into the two queueing systems, the average response time of both systems declines dramatically. Afterwards, the average response time is gradually getting close to the average execution time but has no evident decrease despite that more service instances are being recruited. The reason is that adding new service instances can only decrease the waiting time of workflow activities in the queue.

Therefore, we only add one extra service instance into the queueing system for violation handling to achieve the best cost-effectiveness. Even if in some cases adding one service instance may not be sufficient to compensate all throughput deficit, the handling strategy will be called again timely to add another service instance as temporal behaviors are still constantly monitored by our temporal verification strategy.

(3) **How long the lifecycle of these resources should be?**

Algorithm 2 (Step 2) depicts how to determine the lifecycle of new resources for temporal violation handling. If there is currently no recruited resource in the queueing system, then we add a new resource with a basic lifecycle L (lines 11–12). The real lifecycle $T = m*L$ *(m = 1, 2, 3, ...)*. If there is already a recruited resource in the queueing system (this resource is added in the queueing system at previous checkpoints for violation handling and has not expired yet), then we check the residual time of this resource. If $RL < k * bt$ (namely the resource will expire before the next candidate checkpoint S_{p+1}), we extend its lifecycle by an extra basic lifecycle (lines 14–17).

By adaptively extending the resource's lifecycle, our strategy can make sure only one extra resource is required in the queueing system for temporal violation handling.

4 Evaluation

4.1 Experimental Settings

The simulation experiments are conducted in our cloud workflow system SwinFlow-Cloud [9]. First, we simulate a continuous running of a large number of parallel workflow instances. Basic experimental settings are similar to the settings our previous work [10]. Arrival time and execution time of activities follow general distribution which are simulated by *Simulink*[1]. Basic time unit *bt* is set as an equal interval of 30 s. Basic lifecycle of recruited resources L is set as 1 min.

We compare our approach TV_{adap} with the following two representative strategies:

- TV_{fixed}: It selects checkpoints from a collection of time points along system timeline with fixed interval. This strategy handles temporal violations by adding one extra resource with fixed lifecycle into each selected queueing system [10].
- TV_{acti}: It selects every workflow activity as candidate checkpoint [11]. If a temporal violation is detected, one resource with a fixed lifecycle is added into each selected queueing system (if there is no extra resource in the queueing system).

To get the baseline results for comparison purpose, we record the on-time completion rates of workflow instances under natural situation, i.e., without any temporal verification or violation handling strategies (denoted as **NIL**).

In business scenario, a best strategy should be the one that can reach the target on-time completion rate with high cost-effectiveness. Cost-effectiveness is measured by the average resource consumption for every 1% increment from the baseline. The formula is as follows:

$$\frac{H}{\gamma'\% - \gamma\%} \tag{2}$$

where H denotes the total number of basic lifecycle of recruited resources needed by each strategy, $\gamma\%$ is the baseline on-time completion rate and $\gamma'\%$ is the on-time completion rate achieved by each strategy.

[1] Simulink: https://www.mathworks.com/products/simulink.html.

4.2 Experimental Results

First, we compare TV_{adap} with the other two strategies under different batches of workflows. The workflow size is 25. In Fig. 1, each strategy can significantly improve on-time completion rate when compared with the baseline **NIL**, and all of them can reach the target on-time completion rate of 90%. However, in Table 1 the number of checkpoints and resources consumption are obviously different. Since TV_{acti} is working at each workflow activity, the number of checkpoints is much more than the other two strategies, and it can detect more temporal violations. But its resource consumption for violation handling is several times higher than the other two strategies. TV_{fixed} achieves a slightly higher on-time completion rate than TV_{adap}, but it consumes more resources. In contrast, TV_{adap} can reach the target on-time completion rate with the lowest resource cost. Also, it achieves the highest verification efficiency with the minimum number of checkpoints.

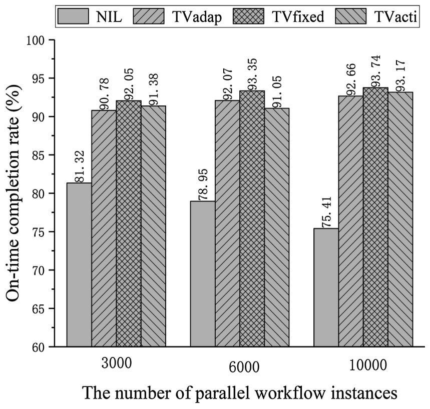

Fig. 1. On-time completion rates with different number of instances

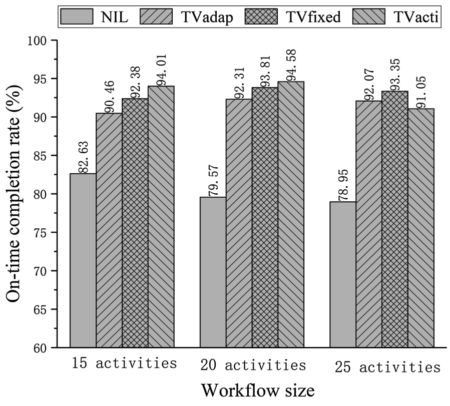

Fig. 2. On-time completion rates with different workflow sizes

Figure 2 and Table 2 show the experimental results under different workflow sizes. The number of parallel workflow instances is 6000. Similar to the above experiment results, all the three strategies can reach 90% on-time completion. Compared with TV_{fixed} and TV_{acti}, TV_{adap} is the most cost-effective, which can achieve target on-time completion with the least resource consumption.

Table 1. Experimental results with different number of workflow instances

Strategies	Checkpoints			Resource lifecycle			Average resource consumption for every 1% increment from the baseline		
	3000	6000	10000	3000	6000	10000	3000	6000	10000
TVadap	**86**	**102**	**121**	**118**	**141**	**252**	**12.47**	**10.75**	**14.61**
TVfixed	132	145	159	172	169	336	16.03	11.74	18.33
TVacti	832	1471	2984	576	902	1685	57.26	74.54	93.36

Table 2. Experimental results with different workflow sizes

Strategies	Checkpoints			Resource lifecycle			Average resource consumption for every 1% increment from the baseline		
	15	20	25	15	20	25	15	20	25
TVadap	**94**	**101**	**102**	**113**	**116**	**141**	**14.43**	**9.11**	**10.75**
TVfixed	152	159	145	141	148	169	14.46	10.39	11.74
TVacti	957	1098	1471	623	816	902	54.75	67.44	74.54

5 Conclusion and Future Work

To achieve on-time completion of time-constrained business cloud workflows, temporal violations occurred at workflow runtime need to be timely detected and handled. In this paper, we present a temporal verification strategy and a temporal violation handling strategy to tackle the problem of "Where" and "How" respectively for handling temporal violations. Considering the fluctuation of workflow temporal behaviors, the idea of "adaptiveness" is introduce into our strategy design. Compared with conventional non-adaptive strategies, our strategies can achieve both higher efficiency and better cost effectiveness.

In the future, we plan to extend the proposed strategies to a more complex environment where instances of different business workflow are mixed in the batch of parallel workflow instances.

Acknowledgement. The authors would like to acknowledge the support provided by the grants of the National Natural Science Foundation of China (61572374, 61300042, U163620068, U1531122, U1135005), the Academic Team Building Plan from Wuhan University (WHU2016012), and the Australian Research Council Discovery Project (DP180100212).

References

1. Kumar, A., Sabbella, S.R., Barton, R. R.: Managing controlled violation of temporal process constraints. In: 13th International Conference on Business Process Management, pp. 280–296 (2015)
2. Wegener, J., Grochtmann, M.: Verifying timing constraints of real-time systems by means of evolutionary testing. Real-Time Syst. **15**(3), 275–298 (1998)
3. Falcone, Y., Havelung, K., Reger, G.: A tutorial on runtime verification. J. Eng. Dependable Softw. Syst. **34**, 141–157 (2013)
4. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Exception handling patterns in process-aware information systems. BPM Center Report BPM-06-04, BPMcenter.org (2006)
5. Liu, X., Wang, D., Yuan, D., Wang, F., Yang, Y.: Workflow temporal verification for monitoring parallel business processes. J. Softw.: Evol. Process. **28**(4), 286–302 (2016)

6. Liu, X., Wang, D., Yuan, D., Yang, Y.: A novel deadline assignment strategy for a large batch of parallel tasks with soft deadlines in the cloud. In: Proceedings of 15th IEEE International Conference on High Performance Computing and Communications & 10th IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), pp. 51–58 (2013)
7. Khazaei, H., Mišić, J., Mišić, V.B.: Performance analysis of cloud computing centers using m/g/m/m+r queuing systems. IEEE Trans. Parallel Distrib. Syst. **5**, 936–943 (2011)
8. Luo, H., Liu, J., Liu, X., Yang, Y.: Predicting temporal violations for parallel business cloud workflows. Softw.: Pract. Exp. **48**(4), 775–795 (2018)
9. Cao, D., Liu, X., Yang, Y.: Novel client-cloud architecture for scalable. In: Proceedings of the 14th International Conference on Web Information Systems Engineering (WISE), pp. 270–284 (2013)
10. Luo, H., Liu, X., Liu, J., Yang, Y.: Propagation-aware temporal verification for parallel business cloud workflows. In: Proceedings of IEEE International Conference on Web Services, pp. 106–113 (2017)
11. Wang, F., Liu, X., Yang, Y.: Necessary and sufficient checkpoint selection for temporal verification of high-confidence cloud workflow systems. Sci. China Inf. Sci. **58**(5), 1–16 (2015)