Scientific Research Publishing

Journal of Software Engineering and Applications, 2019, 12, 339-364
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Optimizing a Long-Lived Transaction with Verification Function

Shinji Kikuchi, Subhash Bhalla

The University of Aizu, Aizu-Wakamatsu City, Japan
Email: shinji.kikuchi@ieee.org, bhalla@u-aizu.ac.jp

How to cite this paper: Kikuchi, S. and Bhalla, S. (2019) Optimizing a Long-Lived Transaction with Verification Function. *Journal of Software Engineering and Applications*, **12**, 339-364.
https://doi.org/10.4236/jsea.2019.129021

Received: July 15, 2019
Accepted: September 21, 2019
Published: September 24, 2019

Abstract

We have considered a method called Enhanced Rollback Migration Protocol, which potentially has the effects of compressing the period of compensations in a long-lived transaction, since before. In general, a compensation transaction can recover an irregular status of a long-lived transaction into the original status without holding unnecessary resources by making its consistency tentatively loose. However, it has also been pointed out that there is a difficulty of maintaining the isolation between a pair of transactions when executed in parallel. In particular, this could be more prominent under modernized scalable cloud environments. Thus, there is a proposal for concurrency control for the service level. However, there is still another risk that more computer resources will be consumed than actually necessary and an unnecessary stagnation of the processing will be caused if concurrency control is naively applied without careful consideration. Therefore, we need to implement a functionality which can optimize the processing of a long-lived transaction by selecting a suitable method between concurrency control and compensation transactions. In this paper, we propose a method in which optimistic concurrency control is applied for long-lived transactions. Furthermore, a pair of verification phases is carried out. At the beginning from a safe point, an attempt of verification is done. Then if the difficulty of isolation on a long-lived transaction executed under a competitive situation is estimated, concurrency control for the service level is applied. Alternatively, a long-lived transaction without any concurrency control is executed. At the next reachable safe point, another attempt of verification is performed. Then if a failure of serialization is detected, a set of compensation transactions is invoked to recover the original long-lived transaction by returning to the first safe point. We evaluated this approach by using numerical simulations and confirmed the basic features. This approach can realize optimizing and enhancing the performance of a long-lived transaction. We regard this approach applicable even to the modernized scalable cloud environments.

Keywords

Transaction Management, Optimistic Concurrency Control, Compensation, Service Level Agreement

1. Introduction

Due to the current paradigm shift to cloud computing especially Business Process as a Service (BPaaS) or cloud workflow, the complicity of transaction processing and requirements of scalability have been continuously maintained and grown. However, handling these transactions has remained immature and become vaguer according to the complicity itself. Since before, we have defined an abstract model and framework that gives a compensation transaction a centralized role in order to cope with the predictive exponential amount of transaction demands [1]. We proposed the Enhanced Rollback Migration Protocol and its performance evaluation [1] [2]. This protocol supports the conservative backward approach. However, it also relies on less dependency on the data status which must be considered in executing a compensation transaction, because we expect the temporal data management to be implemented due to the emergence of a huge amount of cloud storage. This approach includes an algorithm to reduce the number of procedures for compensation transactions between safe points belonging to an instance of a long-lived transaction. Compared with existing approaches, this approach potentially has advantages with regard to the performance.

However, there is another risk about difficulties in maintaining isolation when executing multiple instances of long-lived transaction in parallel. In particular, this could be more prominent under these scalable environments. Accordingly, there has been a proposal to carry out concurrency control for the service level instead of applying compensation [3]. Our aforementioned approach using compensation has relied on traditional backward recovery. Therefore, an issue arises in regard to violating isolation under a competitive execution. However, if concurrency control is applied naively, computer resource lockups will occur more frequently. Furthermore, the negative impacts that compensation tries to relieve will be made even worse. In order to solve this conflict, it is obviously required to implement a mechanism that optimizes the choice of concurrency control or compensation during the execution.

In this paper, we propose a new approach with the optimization, in which optimistic concurrency control is applied at the level of long-lived transaction and has two verification phases. This approach is not limited only to the modernized RESTful style; however, we demonstrate the renewal approach of optimization with clarifying the applicable conditions as our contributions. In this approach, the first verification is carried out at the initial safe point, and concurrency control is applied at the service level when a high probability of violating isolation

under a competitive execution is predicted. Otherwise, further verification is executed at another later safe point, and the corresponding compensation process is invoked to recover the failures or unsuitable states. Accordingly, it is possible to avoid needless resource lockups and to optimize the long-lived transactions. In general, transaction processing in the restful style as the current major approach is assumed with hesitating to deploy a transaction coordinator or adopting variations such as an agent system because of demands in regard to efficiency for the scalability. However, we demonstrate that it is potentially responsible to the demands for the scalability by applying the optimized approach with suitable operational conditions. Through evaluation using numerical simulation, we clarify the applicable conditions for the different choices and critical factors for the performance in executing the long-lived transactions.

The remainder of this paper is organized as follows. In Section 2, we provide an overview, features of our proposal, definition of the components and outline of their behaviors. We also define the conceptual correspondence among them. In Section 3, we discuss in detail the behaviors including optimistic concurrency control. In Section 4, we provide the results of our evaluation, which contains a quantitative evaluation related to SLA (Service Level Agreement). Section 5 presents a brief explanation of related work followed by the conclusion in Section 6.

2. Architecture Model

2.1. Functional Layers and Features

Firstly, the overview of functional layers will be defined, then, the characteristics of our proposal will be explained. By defining the functional layers, it is possible to clarify locations of the corresponding functions by mapping them with layers. Figure 1 shows the outline of the function layer model consisting of the five

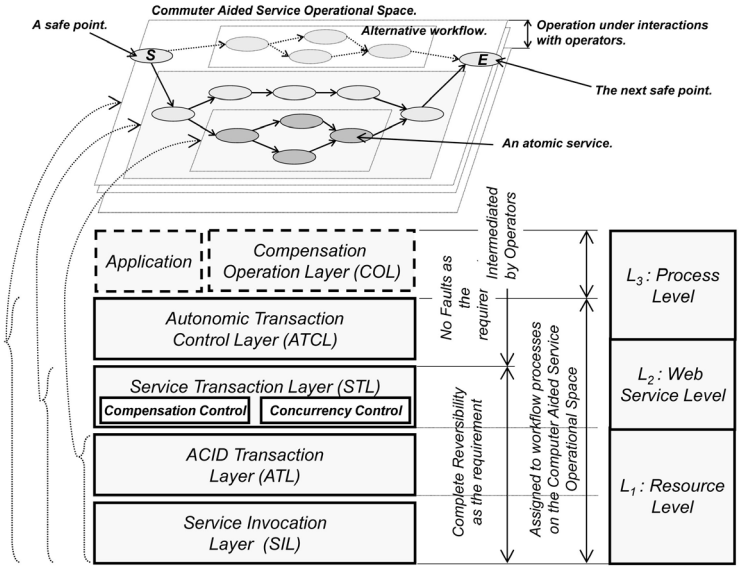

Figure 1. Function layer model including long-lived transaction.

sub-function layers. This is an enhanced version of the original model [1]. It also shows the correspondence between the sub-function layers and the granularity of the activities network. Here, we assume that the four layers from the bottom will automatically be executed along the specified workflows, whereas the fifth layer at the top must be executed by inter-mediation of human operations, or applications.

1) The first layer is the Service Invocation Layer (SIL), which corresponds to the invoked elemental activities as services and is regarded as linked resources. We will use the term "service" to mean "activity" in this paper.

2) The second layer is the ACID Transaction Layer (ATL), which corresponds to the ACID transactions group and the set of elemental activities and guarantees the ACID properties.

3) The third layer is the Service Transaction Layer (STL), which corresponds to an instance of the long-lived transaction such as Saga [4] and others. In the previous work in [1], this was named the Compensational Transaction Layer. This is the advance transaction using the elemental ACID transactions and is also mapped into concurrency control and compensation transactions. Therefore, we consider two sub-functions named Compensation Control and Concurrency Control. In particular, under the case of using Compensation Control, we define the obvious constrained features to this layer, which differ from the fourth layer. That is complete-recovery (or equivalent-recovery in looser cases) to the original status at the passed safe point whenever a fault occurs.

4) The fourth layer is the Autonomic Transaction Control Layer (ATCL), in which the compensation procedures will not be limited within a transactional manner, but a more generalized and non-transaction approach as well. In previous work [1], this was named the Automatic Compensation Layer. However, this is extended because of the existing non-compensation cases. In this layer, there is an obvious feature, which is "no faults". This feature was originally pointed out in [5]. In particular, the fourth layer should be treated with all of the possible procedures featured as fault tolerant.

5) The fifth layer at the top is the Compensation Operation Layer (COL). This must be run as applications or as inter-mediation of human operations. In later cases, semantically equivalent operations for fault tolerant will be executed under the operators' judgments.

Considering the relationship with the granularity of the activities' network, the ATCL should be defined including all paths of processes between safe points. The definition of a safe point is the same as that in [6]. Therefore, a path between safe points corresponds to an instance of workflow on the STL. If there is an alternative path having the same start and end points, another instance of workflow on this layer must be invoked. Any part of the above path consisting of a set of element activities should be mapped to an instance of the ATL, and elemental invocation should correspond to an invocation on the SIL. The three levels at the right side of **Figure 1** correspond to the functional levels defined in

[3]. L2 corresponds to STL and L3 corresponds to a combination of ATCL and COL.

Our proposal in this paper is a method of the STL, in order to maintain isolation and consistency of the long-lived transactions under the predefined Service Level Agreement (SLA). The characteristic features are defined as follows;

1) Safe points must be explicitly defined [6]. Therefore, an instance of a long-lived transaction is divided into multiple preferable scopes according to these safe points, and the multiple fragmental scopes are sequentially executed.

2) An independent monitor is implemented, which tracks the execution states of individual instances of the long-lived transactions. By using this monitor, the evaluation of the SLA is carried out. The individual fragmental instances of long-lived transactions select the optimized method of execution according to the results of the evaluation and constraints derived from the set of instances of the SLA.

3) The actual statistic of results in the previous executions should be referenced when selecting the optimized method of execution. When there are high probabilities of collisions or interferences among multiple instances of the long-lived transactions due to a high density of accessing the services, it is potentially difficult to maintain isolation for the individual instances. In this case, the concurrency control method (CCM), in particular, the one specified in [3] should be applied. Conversely, in the cases of maintaining isolation, the compensation approach defined in [1], [2] should be adopted as the Full-Compensation Method (FCM). These selections must consider resource consumption and unnecessary overheads. Accordingly, it is possible to realize optimized executions of the long-lived transactions at every transactional step.

4) In the FCM, a verification process related to the competitive accessing of the subordinate resources and the interferential collisions of the multiple instances of the long-lived transactions must be demonstrated. In this process, Backward-Oriented Optimistic Concurrency Control (BOCC) should be applied [7]. Once a collision occurs, the compensation transactions are launched in order to cancel the effects of the identified instance of the long-lived transaction. Furthermore, the optimized approach for compensation should be applied as mentioned in [8].

2.2. Entire Configuration

In this section, we explain about configurative model of the logical architecture, depicted in **Figure 2**. This configurative model involves the functional components defined in [3] and [8]. Additionally, it is partly harmonized with legacy industrial standards such as Web Service Atomic Transaction (WS-AT), Web Service Business Activities (WS-BA), and Web Services Business Process Execution Language (WS-BPEL) [9] [10] [11]. However, these referred elemental concepts are not completely harmonized within the original specifications. Therefore, some conceptual adjustments were applied and renamed accordingly. These are defined below.

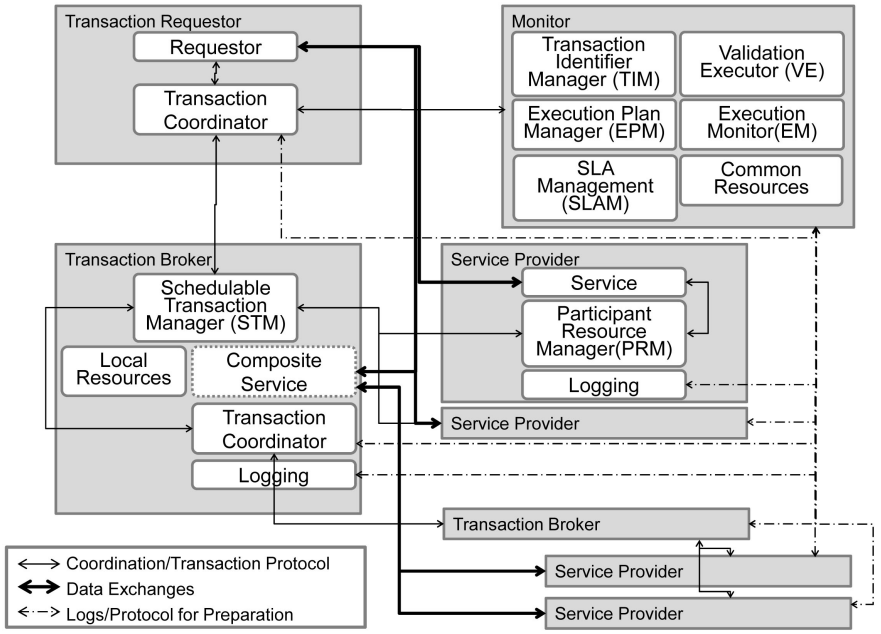

Figure 2. Configurative model of logical architecture.

When viewing the configurative model at the macro level, there are four definable logical elements: Transaction Requestor, Transaction Broker, Service Provider and Monitor. They remain as logical entities, and it is possible and generally accepted to have several practical configurations mapped into the actual physical implementations. There are no explicit logical components corresponding to the Transaction Broker in the specifications of WS-AT and WS-BA. This Transaction Broker is explicitly defined as an extension of the WS-scheduler function defined in [3] in order to realize various composite services. The Transaction Broker and the Service Provider are individually definable as multiple entities, and multiple instances of them could be connected to make various network structures, regardless of the topological aspects such as opened and closed.

The Transaction Requestor element contains a component named Transaction Coordinator which corresponds to the Coordinator of WS-AT [9]. In this Transaction Requestor, the Requestor component is an application regardless of type and category. Thus, it might be a composite service or a workflow.

In the Transaction Broker, the multiple components are included. The component named Schedulable Transaction Manager (STM) corresponds to the functionality of the WS-scheduler in [3]. The component named Local Resources contains various metadata such as the Conflict Matrix defined in [3]. The component named Composite Service corresponds to the generalized implemented instance of the composite service which utilizes another instance of the Transaction Broker and multiple instances of the Service Provider. It is reasonable that the Transaction Coordinator component should be included inside the Transaction Broker in order to realize the Composite Services. Finally, the Logging component is explicitly implemented in order to track the executional

states of the individual instances of the long-lived transactions.

The Service Provider represents the entities of the Service Provider on the edges, and includes a Participant component defined in WS-AT [9]. However, we define this component as the Participant Resource Manager (PRM) in order to emphasize the coordination with the Resource Manager. However, this does not mean a strongly tied implementation. Finally, Logging component is explicitly implemented in the Service Provider in order to track the executional states of individual instances of the long-lived transactions.

The Monitor element consists of the following two sets of components: the set of common utilities and the set of utilities for only the FCM. The former consists of the SLA Management (SLAM) component, the Common Resources component and the Transaction Identifier Manager (TIM) component. The later consists of the Validation Executor (VE) component, the Execution Plan Manager (EPM) component and the Execution Monitor (EM) component.

The SLAM component is used for evaluating the states with regards to SLA according to the extracted actual statistic of results through the monitoring processes. The detailed configuration should be inherited from the architecture mentioned in [8]. According to the results of the evaluation, selecting the optimal method of the execution is done at every fragment instance of the long-lived transactions. The component of the Common Resources is used to manage the specific items of the SLA, and other metadata. Inside the TIM component, the identifiers tagged to the transaction instances such as the Coordinator Context should be defined and managed. A detailed explanation is omitted from this paper.

Inside the procedure of the FCM, there are three phases as explained in Section 3. The first Preparation Phase is the phase for registration with regards to the set of planned entries in the Monitor element. That set consists of multiple Transaction Brokers, multiple composite services and multiple elemental services, all of which actively run during the next Execution Phase. The EPM component should be implemented to manage their registration. The EM component is used for monitoring the running states of the individual instances of the long-lived transaction during the second Execution Phase. The final VE component is utilized for detecting the existence of failures and any gaps between the actual tracked data acquired during the Execution Phase and the planned data registered during the Preparation Phase. Furthermore, it is also used for verifying whether competitive accessing of the subordinate resources and interfering collisions occur among the multiple instances of the long-lived transactions. In particular, it is done by using BOCC [7]. There is an inquiry step in the SLAM component at the beginning just after taking a new safe point. However, as there is no verification during the inquiry, it is not suitable to regard this approach as one of Forward-oriented optimistic concurrency control (FOCC).

As for relationships among elements, there are roughly three categories. The first category corresponds to the Coordination/Transaction Protocol expressing as thin real arrows. This category is the procedures for transaction control which

are almost the standardized protocol, rather than exchanging data among applications. Almost standardized means "Not completely the same and includes some partial modifications", otherwise, mapping into another style with preserving the original features. The actual data exchanged among applications as the second category are expressed as bold arrows. The remaining thin dashed arrows correspond to the proposed protocols such as Logging and the first Preparation Phase as the third category.

2.3. Overview of System Behaviors

Figure 3 depicts the algorithm for the Transaction Coordinator component of the Transaction Requestor element. Some parts of error handling are omitted here. Part of Lines 2 - 8 corresponds to the definition of instances of the handled objects.

As Lines 10 - 12 show, an instance of a long-lived transaction is divided into multiple preferable scopes according to multiple safe points. The multiple fragment scopes are sequentially executed as shown on Lines 13 - 58. In this case, the multiple instances of several long-lived transactions could be executed in parallel as the suffix *i* shows. An identifiable transaction context is commonly shared over multiple safe points.

As Lines 14 - 16 show, the category class of the required instance of the long-lived transaction is identified at the beginning just after taking a new safe point. Then, an inquiry with regards to the optimal method is sent to the SLAM component in order to identify the best selection of the individual fragment in the instance of the long-lived transaction. The best selection is responsible for the constraints derived from the particular SLA for the identified category class. As an SLA is usually decided with dependency on a customer and a use case,

```
01: //Input;
02: Ti ;      // Requested transaction tagged with 'i'.
03: class ;   // ClassType of the requested transaction.
04: methodN ; // Optimized normal transaction method .
05: methodC ; // Optimized compensation transaction method.
06: recieveMsg[ ] ;  // Set of the replied messages.
07: safepoint[ ] ;   // Set of the Safe points.
08: tci ;     // Transaction Context for the requested transaction.
09: //Start;
10: tci = createCoordinationContext( Ti );
11: safepoint[ ] = identifyScope( tci );
12: MAX = count( safepoint[ ] );
13: for( j=0 ; j < MAX ; ){
14:   class   = identifyClassTransaction( Ti );
15:   methodN = identifyOprimizedNormalTransaction( class );
16:   methodC = identifyOprimizedCompensationTransaction( class );
17:   if(identify( methodN , 'ConcurrencyControl' )){
18:     doNegotiation( tci , safepoint[j] );
19:     while( ){
20:       setCompensate( state , Failure );
21:       recieveMsg[ ] = executeTransaction( tci , safepoint[j] , methodN );
22:       for( k=0 ; k < count(recieveMsg[ ] ) ; k++ ){
23:         if( identify( recieveMsg[k] , 'Wait' )){
24:           if( !doDetectAcycle( tci , safepoint[j] ) ){
25:             setCompensate( state, True );
26:             doCompensate( tci , safepoint[j] ); //As a Backward Recover.
27:           }
28:         }
29:         else if( identify( recieveMsg[k] , 'Failure' )){
30:           setCompensate( state, True );
31:           doCompensate( tci , safepoint[j] ); //As a Backward Recover.
32:         }
33:       }
34:       if( ! checkCompensate( state ) ) break;
35:     }
36:   }
37:   else{
38:     doTransactionPrepare( tci , safepoint[j] );
39:     while( ){
40:       setCompensate( state, Failure );
41:       recieveMsg[ ] = executeTransaction( tci , safepoint[j] , methodN );
42:       for( k=0 ; k < count( recieveMsg[ ] ) ; k++ ){
43:         if( identify( recieveMsg[k] , 'Failure' )){
44:           setCompensate( state, True );
45:           doCompensate( tci, safepoint[j], methodC ); //As a Backward Recover.
46:         }
47:       }
48:       if( !verification(tci, safepoint[j], class ) ){
49:         setCompensate( state, True );
50:         doCompensate(tci, safepoint[j], methodC ); //As a Backward Recover.
51:       }
52:       if( ! checkCompensate( state ) ) break ;
53:     }
54:   }
55:   doClose(tci, safepoint[j]);
56:   doSafePoint(tci, safepoint[j]);
57:   j++ ;
58: }
59://End;
```

Figure 3. Algorithm for the transaction coordinator component.

elemental information such as arguments needs to be specified in the inquiry. However, a simplified model is depicted here. The actual statistical results in the executions should be referenced when selecting the optimal method of the execution. When there are high probabilities with regards to the interfering collisions among multiple instances of the long-lived transactions due to a high density of accessing services, it is potentially difficult to maintain isolation for the individual instances from the others. In this case, the CCM should be applied. Conversely, in the case of maintaining the isolation, the FCM should be adopted. In the FCM, the method of compensation is also specified as shown in [8].

Part of Lines 17 - 36 corresponds to the CCM and the flow of this part almost follows the description in [3]. In particular, at line 18, negotiation with the STM component is carried out in order to ensure serialization, and the actual invocation of the transaction is caused at line 21. Then, the post-procedure is executed to restrain the invocations due to transaction dependencies through the concurrency control. Specifically, the post-procedure corresponds to the detection of no global waiting cycle on dependencies in order to maintain serialization in Lines 24 - 26. When detecting a cyclic case, a partial compensation process occurs. We assume that the compensation process is a backward recovery process rather than the forward recovery process defined in [12]. Thus, the equivalent set of elemental transactional services that are canceled by the partial compensation process must be rerun. Accordingly, the re-execution of the transactions is done at line 21.

Part of Lines 37 - 54 corresponds to the FCM. During the first Preparation Phase, registration with regards to the set of planned resources such as multiple Transaction Brokers, multiple composite services and multiple elemental services in the Monitor element are executed at line 38. These planned resources actively run for the instance of the long-lived transaction during the next Execution Phase at line 41. If an inconsistency due to a failure is detected, a compensation process sequentially occurs. During the commitment phase, a one phase commitment (1PC) is applied instead of a two-phase commitment (2PC). This is done to relieve the overhead inside the protocol because the compensation transaction is executed for recovering from failures.

Part of Lines 48 - 50 corresponds to the verification of whether competitive accessing of the subordinate resources and the interfering collisions occur among the multiple instances of the long-lived transactions by using BOCC [7]. In the case of detecting a collision, corresponding compensation transactions are sequentially executed to replace the inconsistent state with the original one before executing the fragment instance of the long-lived transactions at line 41. In the FCM, the optimal method of compensation is selected according to the conditions specified in the SLA at line 16. The detailed procedures are given in [1], [8]. The post-procedure at Lines 55 - 57 is demonstrated to complete the fragment instance of the long-lived transactions. In particular, durability is ensured by taking a safe point at line 56. Then, all of the steps are repeated until all fragment instances of the long-lived transaction are completed.

3. Procedures and Protocols

3.1. Concurrency Control Sensitive Protocol

Figure 4 depicts the procedures of the CCM under the configurative model of the logical architecture shown in Figure 2. The symbols described as <X> correspond to the order number X in steps as Step X. In Figure 4, we do not explicitly utilize the Composite Service. Therefore, it is not presented in this figure. Accordingly, we assume that only the Transaction Coordinator component inside the Transaction Requestor element directly accesses the multiple PRMs. However, it is expected to access the STM component for controlling the transactions. The basic procedure should obey the protocol specified in [3]; however, we include several enhancements and clarifications. For example, the multiple instances of the service invocation are assumed to be the long-lived transactions. Therefore, in their processes, the context creations and registrations by the coordinator must be included. However, there are few explicit descriptions of these processes in [3].

The listed characteristics of the CCM are as follows:

1) The procedures from Step 1 to Step 15 are common in any cases and those from Step 16.1 to Step 20.1 are applied in normal cases. Otherwise, the procedures specified from Step 16.2 to Step 27.2 are used in the failure cases.

2) In many cases, the long-lived transactions specified as the extended WS-BA could be carried out based on the executions of multiple instances of the elemental transaction such as WS-AT. However, there are no explicit figures about the commitment phase of the transaction in Figure 4. Thus, some procedures related to Complete and Close are running from Step 13 to Step 18 after executing Data Exchanges (Response) at Step 11. However, we implicitly assume that

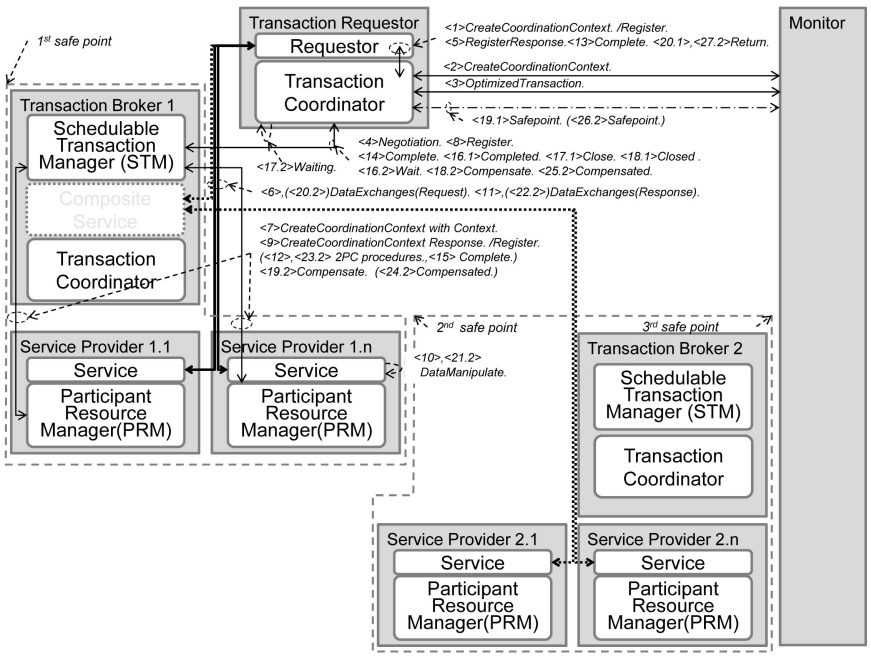

Figure 4. Procedure of CCM on the configurative model.

the 2PC procedure would be executed in the background. So, we need both models: one that uses the representative transaction such as WS-AT implicitly and the other that does not apply these transactions. As an additional assumption here, it should also be considered that an equivalent protocol such as in Json expression, which entirely preserves the corresponding states defined in the original, could be sufficiently applied.

3) When carrying out compensation transactions at Step 18.2 and 19.2, these should be regarded as the partial BOCC. Therefore, both Data Exchanges at Step 6 and Step 11 must be re-executed later. However, these are also not explicitly expressed in **Figure 4**.

4) The corresponding procedures at Steps 2, 3, 19.1 and 26.2 are not explicitly mentioned in [3]. They are derived from our original proposal.

5) As for the achieved results related to the actual responsibility and practical reliability of the individual services, they should be reported to the Monitor in an independent method, although no corresponding depictions are shown in **Figure 4**.

6) During the negotiation between the Transaction Coordinator and the STM components at Step 4, the values of "the expected time for acquiring the exclusive lock" and "the deadline for releasing the exclusive lock" must be negotiated. Further, decisions should be made for each specified Web Service based on the individual SLA instance. However, these procedures can have high communication costs.

7) From Step 14 to Steps 16.1 and 16.2, the verification process must be carried out which is equivalent to the specified item at line 24 in **Figure 3**. The OCC should be applied here [3]. However, in this approach, what is detected is only at verification of the no global cyclic waiting situation before commitments, and has different features from the BOCC approach defined in the FCM. And, it is also predictive that this procedure takes high communication costs in actual cases.

3.2. Reservation Protocol in Non-Concurrency Control

Figure 5 depicts the procedures of the FCM under the model of the logical architecture shown in **Figure 2**. In **Figure 5**, unlike **Figure 4**, we explicitly assume to utilize the Composite Service. The reason is that the Monitor must have the ability to trace the interactions among services of any types, even complicated structures consisting of multiple composite services. Furthermore, the Monitor must grasp the actual configurations of planned resources in advance, and monitor the current status of a flow as a time series of events. In **Figure 5**, we assume that the Transaction Coordinator component inside the Transaction Requestor element directly accesses the multiple PRMs. However, it is also expected to access the STM component for controlling the transactions by the Transaction Coordinator component not shown explicitly in the figure. The characteristics of the FCM are as follows:

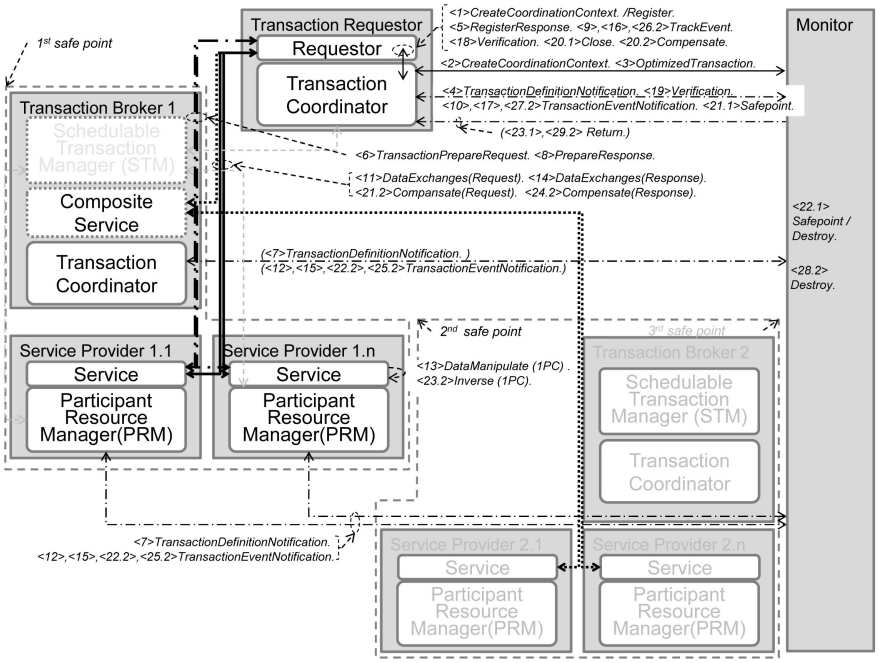

Figure 5. Procedure of FCM on the configurative model.

1) The FCM is broken down into three phases. The first is the Preparation Phase in which the registration of the set of planned resources such as the multiple Transaction Brokers, multiple composite services and multiple elemental services with the Monitor element is carried out. This corresponds to the partial procedure from Step 4 to Step 9 in **Figure 5**. Therefore, the registration information as "Transaction Definition Notification" is sent to the Monitor element at Step 4 and Step 7.

2) The second is the Execution Phase in which 1PC procedures are applied because the compensation transaction will be executed in the failure cases, and it is required to minimize the overhead of the protocol. This corresponds to the partial procedure from Step 10 to Step 17 in **Figure 5**. If a failure is detected, the compensation will be undertaken, but in **Figure 5** we merge it with procedures after Step 20.2, which belong to the third Adjustment Phase. The ambiguity of the border between the Preparation Phase and the Execution Phase we regard as a post-timing of Step 9, in which the Requestor requests the Transaction Coordinator to invoke Step 10. The synchronicity of them is not specified in this paper, because of a macro view level.

3) The third is the Adjustment Phase in which the BOCC is applied. This phase corresponds to the steps from Step 18 to the end in **Figure 5**. Based on the procedures mentioned in 2 above, the verification might be embedded into the second Execution Phase. However, we regard to define the verification as part of this Adjustment Phase. After carrying out the verification, if there are no errors and no detected competitions with another instance of the long-lived transaction, the procedures from Step 20.1 to Step 24.1 will run in order to close the transaction. (Step 24.1 is not drawn in **Figure 5**). In particular, at Step 22.1, the

next safe point is taken, and then, the set of the registration information as a "Transaction Definition Notification" generated in the Preparation Phase is cleared.

4) When detecting competitions among instances of the long-lived transaction or an occurrence of the errors after Step 18, the compensation is sequentially undertaken. In this case, the set of procedures from Step 20.2 to the end are applied to the sacrificed transaction. We do not specify which instance should be aborted in this paper. The undertaken compensation at Step 23.2, 1PC procedures to the set of the subordinate PRMs is also applied. Therefore, there are several possibilities that the compensations themselves could fail. In these cases, the retries should be adopted. However, this is not depicted in **Figure 5**.

5) In the case of a retry of the failed long-lived transaction from the beginning with a successful result of the compensation, the first Preparation Phase is not re-executed as shown in **Figure 3**. This lack of execution relies on the fact that the Preparation Phase could be costly. However, if considering availability of the subordinate resources, the planning of the consumed resources such as the Transaction Brokers, composite services and elemental services should be reorganized. For this, we need to carry out a quantitative evaluation, which is linked to defining the allowable maximum number retrying the compensations.

6) The procedures at Step 2 and Step 3 should be common and sharable with the CCM.

4. Evaluation

4.1. Overview

It is preferable to compare our proposal to existing approaches in order to verify the validity. However, as our proposal is a method to optimize a long-lived transaction by embedding the existing approaches, it is possible to show the validness indirectly by clarifying the guidelines for adopting the different conditions for the given cases. In this way, it is required to execute our evaluation process consisting of two stages. At the first stage, we clarify the characteristic features of the CCM and the FCM individually. Then, at the second stage, based on the results of the first stage, we define formulas that determine which method should be adopted for the given cases, based on the specified conditions in the SLA. In the first stage, we evaluate the features by using numerical simulation. We apply a model that has simplified aspects for the following reasons. It is predictable that some parts of the procedure in the CCM might suffer from degraded performance due to the complicity of topological relationships in invoking the services. However, our central issue is a comparison between the CCM and the FCM instead of the evaluation of the complicity of the topological relationships. Accordingly, we primarily deal with the basic features in this paper. The issues caused by the complicity of topological relationships are regarded as future work. In the second stage, we carry out a qualitative evaluation instead of a quantitative evaluation because an SLA consists of various factors.

In Section 4.2, we explain the evaluation of the first stage. Then, we discuss the evaluation of second stage in Section 4.3. In the final Section 4.4, we consider the relationship between the current transaction management and related standards.

4.2. Comparison between the CCM and the FCM

The procedures depicted as the CCM in Figure 4 and as the FCM in Figure 5 do not contain complicated services' invocations in a hierarchy and correspond to the most primitive and simplified topological relationships in invoking services. We digitize these procedures by using a metric of the process cost which is an abstract concept of the processing time. Accordingly, this process cost is a number greater than zero without any unit and becomes worse with larger values. We assume a set of values of this process cost depends on the individual steps of the process; therefore, we express the individual process cost by using variables given as (1), (2), (3), (4) and (5).

$$C_{inter} \tag{1}$$

$$C_{cm} \tag{2}$$

$$C_{dm} \tag{3}$$

$$C_{trans} \tag{4}$$

$$C_{nego} = k \cdot C_{trans} \tag{5}$$

The variable defined by (1) represents the average cost for the process within a physical site without any communication with the outside. The variable defined by (2) represents the average cost for accessing the Monitor element with communication. The variable defined by (3) expresses the average cost for writing data within a physical site without any communication with the outside. The variable defined by (4) represents the average cost for primitive interactions of a transaction using communication. These primitive interactions cover any application data and native procedures of the protocols. Even for an instance of exchanging a huge amount of data, the corresponding procedure should be expressed by using this variable (4), because of the native aspect of the average. Finally, the variable defined by (5) expresses the average execution cost in the negotiation process between the Transaction Coordinators and the STM components in the CCM. This variable could originally and directly be affected by the complicity of topological relationships in the invoking services; however, it is expressed as the variable (5), which we regard as k times ($k > 0$) the variable (4).

Table 1 contains the results of mapping between the individual procedures depicted in Figure 4 and variables expressed in (1), (2), (3), (4) and (5). Furthermore, Table 2 shows the mapping results in Figure 5. The procedures from <16.1> to <20.1> in Table 1 correspond to the case where the CCM is applied and finishes successfully. The procedures from <16.2> to <27.2> in Table 1 correspond to the case that ended in a failure and applied a compensation process. On the other hand, the procedures from <20.1> to <24.1> in Table 2 correspond to the case where the FCM is applied and finished successfully. The

Table 1. Mapping results between procedures in CCM and individual costs.

Step	Title	Category	Symbol (Times)
<1>	Create Coordination Context/Register	Internal Process at Transaction Requestor	$C_{inter}(1)$
<2>	Create Coordination Context	Communication with Monitor.	$C_{cm}(1)$
<3>	Optimized Transaction	Communication with Monitor.	$C_{cm}(1)$
<4>	Negotiation	Communication with Transaction Broker	$C_{nego}(1)$
<5>	Register Response	Internal Process at Transaction Requestor	$C_{inter}(1)$
<6>	Data Exchanges (Request)	Communication with Service Providers	$C_{trans}(1)$
<7>	Create Coordination Context with Context	Communication between Transaction Broker and Service Providers	$C_{trans}(2)$
<8>	Register	Communication with Transaction Broker	$C_{trans}(2)$
<9>	Create Coordination Context Response/Register	Communication between Transaction Broker and Service Providers	$C_{trans}(2)$
<10>	Data Manipulate	Internal Process at Service Providers	$C_{dm}(1)$
<11>	Data Exchanges (Response)	Communication with Service Providers	$C_{trans}(1)$
(<12>)	2 PC Procedures		$C_{trans}(4)$
<13>	Completed	Internal Process at Transaction Requestor	$C_{inter}(1)$
<14>	Completed	Communication with Transaction Broker	$C_{trans}(1)$
<15>	Completed	Communication between Transaction Broker and Service Providers	$C_{trans}(1)$
<16.1>	Completed	Communication with Transaction Broker	$C_{trans}(1)$
<17.1>	Closed	Communication with Transaction Broker	$C_{trans}(1)$
<18.1>	Closed	Communication with Transaction Broker	$C_{trans}(1)$
<19.1>	Safepoint	Communication with Monitor.	$C_{cm}(1)$
<20.1>	Return	Internal Process at Transaction Requestor	$C_{inter}(1)$
<16.2>	Wait	Communication with Transaction Broker	$C_{trans}(1)$

Continued

<17.2>	Waiting		$C_{nego}(1)$
<18.2>	Compensate	Communication with Transaction Broker	$C_{trans}(1)$
<19.2>	Compensate	Communication with Service Providers	$C_{trans}(1)$
(<20.2>)	Data Exchanges (Request)	Communication with Service Providers	$(C_{trans}(1))$
<21.2>	Data Manipulate	Internal Process at Service Providers	$C_{dm}(1)$
(<22.2>)	Data Exchanges (Response)	Communication with Service Providers	$(C_{trans}(1))$
(<23.2>)	2 PC Procedures		$C_{trans}(4)$
<24.2>	(Compensated)	Communication with Service Providers	$C_{trans}(1)$
<25.2>	Compensated	Communication with Transaction Broker	$C_{trans}(1)$
<26.2>	(Safepoint)	Communication with Monitor	$C_{cm}(1)$
<27.2>	Return	Internal Process at Transaction Requestor	$C_{inter}(1)$

Table 2. Mapping results between procedures in FCM and individual costs.

Step	Title	Category	Symbol (Times)
<1>	Create Coordination Context/Register	Internal Process at Transaction Requestor	$C_{inter}(1)$
<2>	Create Coordination Context	Communication with Monitor	$C_{cm}(1)$
<3>	Optimized Transaction	Communication with Monitor	$C_{cm}(1)$
<4>	Transaction Definition Notification	Communication with Monitor	$C_{cm}(1)$
<5>	Register Response	Internal Process at Transaction Requestor	$C_{inter}(1)$
<6>	Transaction Prepare Request	Communication with Service Providers	$C_{trans}(1)$
<7>	Transaction Definition Notification	Communication between Service Providers and Monitor	$C_{cm}(1)$
<8>	Prepare Response	Communication with Service Providers	$C_{trans}(1)$
<9>	Track Event	Internal Process at Transaction Requestor	$C_{inter}(1)$
<10>	Transaction Event Notification	Communication with Monitor	$C_{cm}(1)$
<11>	Data Exchanges (Request)	Communication with Service Providers	$C_{trans}(1)$
<12>	Transaction Event Notification	Communication between Service Providers and Monitor	$C_{cm}(1)$

Continued

<13>	Data Manipulate (1 PC)	Internal Process at Service Providers	$C_{dm}(1)$
<14>	Data Exchanges (Response)	Communication with Service Providers	$C_{trans}(1)$
<15>	Transaction Event Notification	Communication between Service Providers and Monitor	$C_{cm}(1)$
<16>	Track Event	Internal Process at Transaction Requestor	$C_{inter}(1)$
<17>	Transaction Event Notification	Communication with Monitor	$C_{cm}(1)$
<18>	Verification	Internal Process at Transaction Requestor	$C_{inter}(1)$
<19>	Verification	Communication with Monitor	$C_{cm}(2)$
<20.1>	Close	Internal Process at Transaction Requestor	$C_{inter}(1)$
<21.1>	Safepoint	Communication with Monitor	$C_{cm}(1)$
<22.1>	Safepoint/Destroy	Internal Process at Monitor	$C_{dm}(1)$
<23.1>	Return	Communication with Monitor	$C_{cm}(1)$
<24.1>	Return	Internal Process at Transaction Requestor	$C_{inter}(1)$
<20.2>	Compensate	Internal Process at Transaction Requestor	$C_{inter}(1)$
<21.2>	Compensate (Request)	Communication with Service Providers	$C_{trans}(1)$
<22.2>	Transaction Event Notification	Communication between Service Providers and Monitor	$C_{cm}(1)$
<23.2>	Inverse (1 PC)	Internal Process at Service Providers	$C_{dm}(1)$
<24.2>	Compensate (Response)	Communication with Service Providers	$C_{trans}(1)$
<25.2>	Transaction Event Notification	Communication between Service Providers and Monitor	$C_{cm}(1)$
<26.2>	Track Event	Internal Process at Transaction Requestor	$C_{inter}(1)$
<27.2>	Transaction Event Notification	Communication with Monitor	$C_{cm}(1)$
<28.2>	Destroy	Internal Process at Monitor	$C_{dm}(1)$
<29.2>	Return	Communication with Monitor	$C_{cm}(1)$
<30.2>	Return	Internal Process at Transaction Requestor	$C_{inter}(1)$

procedures from <20.2> to <30.2> in Table 2 correspond to the case that ended in a failure and applied a compensation process.

In particular, as mentioned in Section 3.1, we have introduced several clarifications in Table 1 because the manner specified in [3] contains some ambiguous

points. As far as applying a coordination protocol such as WS-Coordination, the set of procedures consisting of <1>, <5>, <7>, <8>, <9> is required. For instance, <1> corresponds to the procedure that should be executed first in registration. <5> means the completeness of the registration procedure at the requester side. <7> is one of the procedures of the registration at the resource side and corresponds to informing the context after receiving an application dependent message at the PRM. In <7>, there is a paired procedure consisting of creating a context and registering it in the detailed definition. <8> and <9> are also elemental procedures for registering the context. As STM is the function of an intermediator, the created context should be forwarded to the coordinator. Then, the acknowledgement should be returned to the STM and the PRM as a confirmation after successful execution. There is also a paired procedure consisting of creating a context and registering when considering the exchange procedures in detail.

Furthermore, there are a few explicit descriptions of the execution of the protocol corresponding to WS-AT in [3]. It is possible to interpret the usage of WS-BA in [3] in two different ways. Without the particular approach of Saga [4], it is possible to execute an elemental transaction in one phase commitment. In this case, the procedures of two-phase commitment do not occur. Whereas using the advanced layered approach of Saga, it is appropriate to assume an execution of the elemental transaction in the two-phase commitment. In this case, there are explicitly the procedures of 2PC corresponding to the status: prepare, prepared, commit and committed.

If we express the probability of a transaction failure with probable variable f, the mathematical expectation of the total cost is expressed as (6) in the case of executing the CCM without two-phase commitment (2PC) under the most primitive environment in invoking services. Then, as for the expected total cost (ETC) by normalization of the variable (4) as the average cost for primitive interactions of a transaction using communication, we derive (7). Furthermore, by differentiating (7) with respect to f, we derive (8).

$$\begin{aligned}\mathrm{ETC}_{\mathrm{CCM/NO2PC}} &= (1-f)\cdot\left(4C_{inter}+3C_{cm}+C_{dm}+(k+13)\cdot C_{trans}\right)\\ &\quad + f\cdot\left(4C_{inter}+3C_{cm}+2C_{dm}+(2k+17)\cdot C_{trans}\right).\end{aligned} \tag{6}$$

$$\mathrm{ETC}^{\mathrm{normalized}}_{\mathrm{CCM/NO2PC}} = 4\frac{C_{inter}}{C_{trans}}+3\frac{C_{cm}}{C_{trans}}+(1+f)\cdot\frac{C_{dm}}{C_{trans}}+(k+13+k\cdot f+4f). \tag{7}$$

$$\frac{\mathrm{d}\left(\mathrm{ETC}^{\mathrm{normalized}}_{\mathrm{CCM/NO2PC}}\right)}{\mathrm{d}f} = \frac{C_{dm}}{C_{trans}}+(k+4). \tag{8}$$

Conversely, the mathematical expectation of the total cost is expressed as (9) in the case of executing the CCM with a two-phase commitment (2PC) under the most primitive environment. Then, as for ETC by normalization of the variable (4) we derive (10). Finally, the mathematical expectation of the total cost is expressed as (11) in the case of carrying out the FCM under the most primitive environment. Then, as for ETC by normalization of the variable (4) as the aver-

age cost for primitive interactions of a transaction using communication, we derive (12). Furthermore, by differentiating (12) with respect to f, we derive (13).

$$\begin{aligned} \mathrm{ETC}_{\mathrm{CCM/2PC}} &= (1-f)\cdot\left(4C_{inter}+3C_{cm}+C_{dm}+(k+17)\cdot C_{trans}\right) \\ &\quad + f\cdot\left(4C_{inter}+3C_{cm}+2C_{dm}+(2k+25)\cdot C_{trans}\right). \end{aligned} \tag{9}$$

$$\mathrm{ETC}_{\mathrm{CCM/2PC}}^{\mathrm{normalized}} = 4\frac{C_{inter}}{C_{trans}}+3\frac{C_{cm}}{C_{trans}}+(1+f)\cdot\frac{C_{dm}}{C_{trans}}+(k+17+k\cdot f+8f). \tag{10}$$

$$\begin{aligned} \mathrm{ETC}_{\mathrm{FCM}} &= (1-f)\cdot\left(7C_{inter}+12C_{cm}+2C_{dm}+4C_{trans}\right) \\ &\quad + f\cdot\left(8C_{inter}+14C_{cm}+3C_{dm}+6C_{trans}\right). \end{aligned} \tag{11}$$

$$\begin{aligned} \mathrm{ETC}_{\mathrm{FCM}}^{\mathrm{normalized}} &= (7+f)\cdot\frac{C_{inter}}{C_{trans}}+(12+2f)\cdot\frac{C_{cm}}{C_{trans}} \\ &\quad +(2+f)\cdot\frac{C_{dm}}{C_{trans}}+(4+2f). \end{aligned} \tag{12}$$

$$\frac{\mathrm{d}\left(\mathrm{ETC}_{\mathrm{FCM}}^{\mathrm{normalized}}\right)}{\mathrm{d}f} = \frac{C_{inter}}{C_{trans}}+2\frac{C_{cm}}{C_{trans}}+\frac{C_{dm}}{C_{trans}}+2. \tag{13}$$

Our evaluation was carried out using the above set of formulas and applying the values of the variables given in Table 3. The ratio between the cost variable expressed in (1) and the cost variable expressed in (4) is tentatively set as 0.11 because the average cost for the process within a physical site without any communication is much smaller than the average cost for primitive interactions of a transaction using communication. The ratio between the cost variable expressed in (2) and another cost variable expressed in (4) is less than one and remains within several times at the maximum. Because, the average cost for accessing the monitor with communication is smaller than the average cost for primitive interactions of a transaction. Finally, the ratio between the cost variable expressed in (3) and another cost variable expressed in (4) is less than one and remains around one at the maximum, due to the similar reasons of the average cost for writing data within a physical site with the cost variable expressed in (3). As mentioned before, the variable k ($k > 0$) could directly be affected by the complicity of topological relationships in invoking services due to aspects of the CCM. Accordingly, we assume that the range of this variable should be specified as Table 3 shows.

Table 3. Specified values to individual variables.

Variables	Range	Representative Value
C_{inter}/C_{trans}	0.05 - 0.25	0.11
C_{cm}/C_{trans}	0.2 - 2.2	0.5
C_{dm}/C_{trans}	0.05 - 1.05	0.75
k	0.5 - 3.0	1.2
f	0.01 - 0.2	0.05

Figure 6 depicts the result of the evaluation with respect to the dependency of the ETC on the probability of transaction failure. According to (8) and (13), they are linear functions. From this figure, we extract three insights: The first is that there are very small influences on the costs by the growth of the probability due to whether a transaction is aborted in both the CCM and FCM cases. It is impossible to grasp the tendency in the case of a larger probability of transaction failure through Figure 6. However, empirically it looks like a rare case that the probability of an abort is more than 20 percent, and it might not be a general case. Based on this thought, the influences of the probability of a transaction failure might be limited. The second is that the cost of the FCM tends to be lower under the specified conditions. This means that the FCM has an advantage in resolving an uncompleted state caused by an abort in shorter time under the given conditions. The third is that the existence of 2PC procedure in the CCM obviously impacts the ETC.

Figure 7 depicts the result of the evaluation with respect to the dependency of the ETC on the ratio between the C_{cm} and the C_{trans}. According to this figure, as far as applying fewer accesses to the monitor, the FCM could show its advantage under certain conditions. However, the more expensive the accessed cost is to the Monitor element, the more disadvantage cost is consumed in the ETC. Furthermore, this factor might bring a worse result than that of increasing k, which means degradation of the executed process during the negotiation in the CCM. This also suggests that the degraded performance at the Monitor element in terms of scalability might contribute to more stability of the CCM. More analyses might be meaningless because the value of k depends on the topological relationships in invoking the services.

Figure 8 depicts the result of the evaluation with respect to the dependency of the ETC on the ratio between C_{dm} and C_{trans}. One of the major insights from Figure 8 is that there are no critical influences of the ETC by writing data within

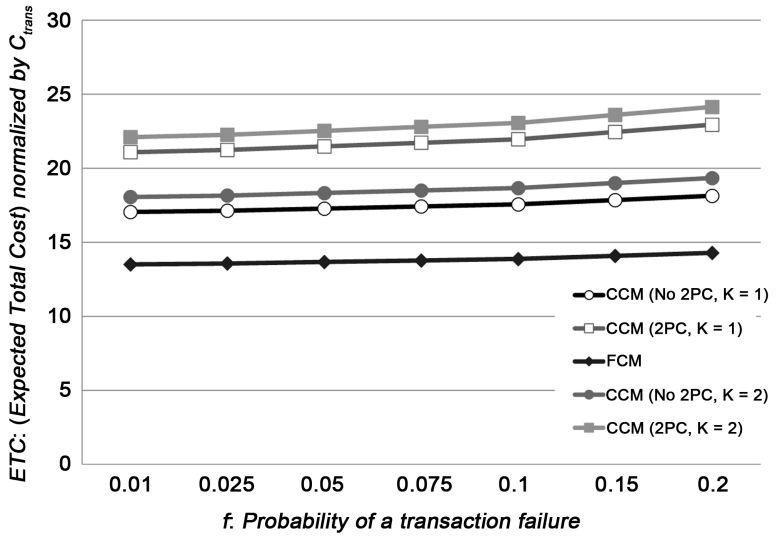

Figure 6. Dependency of ETC on probability of a transaction failure.

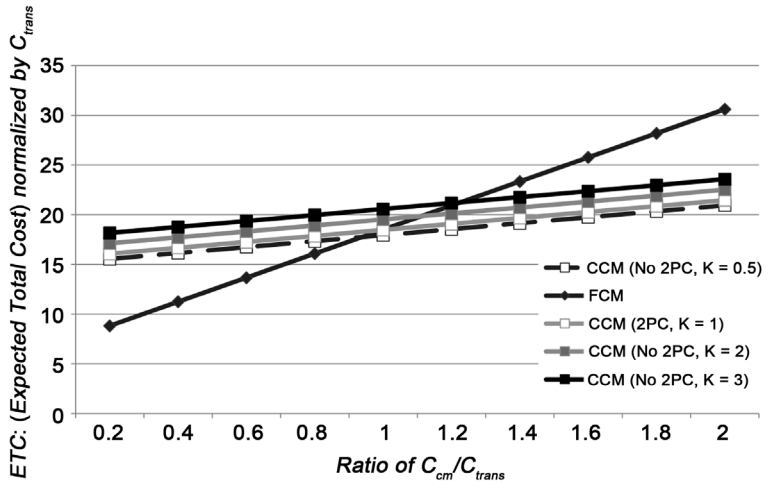

Figure 7. Dependency of ETC on ratio between C_{cm} and C_{trans}.

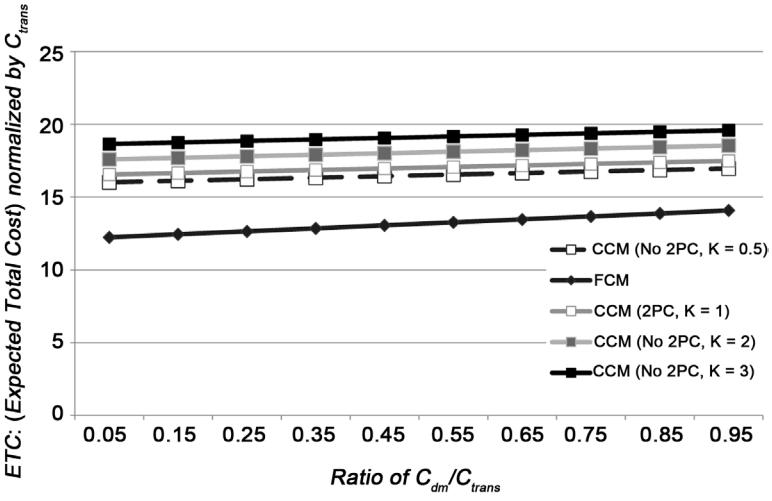

Figure 8. Dependency of ETC on ratio between C_{dm} and C_{trans}.

a physical site without any communication, regardless of CCM and FCM under the given condition. Furthermore, the above suggests two subsidiary insights. The first is that there are stronger factors that dominate the cost than writing data within a physical site. These might potentially be accessing the Monitor element in Figure 7 and the related cost of executing the negotiation process in the CCM. The second is that the overhead to update a replica on a remote side might be limited compared to the impacts of the mentioned stronger factors. This feature suggests that making distributed replicas is sufficiently possible and advantageous for maintaining reliability. This is important when thinking about the items in the SLA.

4.3. Qualitative Evaluation Based on SLA

Because a SLA will be decided as the agreement between the service provider and the user, it should be possible to contain some items despite guidelines [13]. However, it might be appropriate to involve items related to security, main-

tained maximum response time, and availability and reliability in regard to error occurrences. Security is out of the scope of our study.

Considering the response time of calling a service merely according to the evaluation results in the previous session, there are several cases where it is preferable to select FCM at the initial safe point if maintaining the scalability in accessing the Monitor element. However, as higher density of invocation of transactions leads to greater difficulty in maintaining the isolation among the transactions, applying the CCM is definitely required. However, the CCM is also the approach in which we are forced to use compensation. Consequently, it is probable that the FCM could show an advantage under the conditions where scalable accessing of the Monitor element could be implemented.

To make the error rate of transaction processing less than specified values, using the alternative invocation of services becomes an available candidate, especially writing data within multiple physical sites. In particular, as **Figure 8** shows, it does not impact the ETC much compared to the other negative potential factors. Furthermore, as **Figure 6** shows, it could be denied mentioning that the sensitivity of the ETC against the probability of a transaction failure is high. Accordingly, it suggests that the requirements on availability and reliability are not the most critical factors when selecting the approach at the initial safe point. This insight could lead to the following: Selecting the CCM or FCM as the optimized approach based on the SLA should primarily rely on the density of invocation of transactions, but this does not naturally mean that availability and reliability are ignorable factors. In that case, we need to take two other items into account other than the SLA. The first is the capability of the Monitor element in regard to the scalable access as mentioned previously, and the second is the complicity of service invocations. The former item is one of the typical issues at the providers and the latter is a matter that the user should manage. Therefore, both items are not related to the SLA directly, because they are not common for both sides.

In spite of being independent from the SLA, there is one more point to be considered when selecting the approach. That is flexibility in invoking services in order to realize the required performance. This means a dynamic change at run time. As the FCM has the preparation phase before executing a transaction, the FCM tends to have less dependency on the specific protocol and to be enhanced more by loose coupling. The above feature might be included in the list of requirements for supportable software in spite of not major points in the SLA. We should regard the above features of the FCM more positive.

4.4. Relationships with Current Standards of Transaction Processing

To the best of our knowledge, the standards of Web Service Transactions are limited by only OASIS WS-AT and WS-BA based on SOAP messaging. Moreover, there are just few practical activities about the standardization for RESTful services, in spite of several efforts [9] [10] [14] [15]. Furthermore, there are cur-

rently no standardized protocols to support FCM. As the features of the FCM, there are three phases named as Preparation, Execution and Adjustment phases. This approach is responsible for negotiating the execution plan of a transaction prior to the actual execution between the opposite sides, whenever the complicated relationships in service invocation should dynamically be changed, or whenever there is insufficient information about supportable protocols by the opposite sides. In this sense, it is difficult to map these phases with existing standard protocols, and it is also expected to apply the RESTful style, which relies on the HTTP protocol directly, because of fewer mismatches.

Finally, CCM explained in [3] relies on WS-BA with the specific extension. However, as mentioned before, there are few explicit descriptions about execution of WS-AT. The existence of 2PC which is supported by WS-AT, in the CCM obviously impacts the ETC. Currently, there might be potential requirements to modernize these protocols as CCM.

5. Related Work

In this section, we briefly discuss the relationships with other works. Presume abort, Presume commit, One phase and Read-Only transactions are some traditional and well-known techniques of transaction optimization [16] [17]. However, we will not deal with them here. The area which we have focused on in service computing is more complicated. Therefore, the related work can mainly be categorized into the following two areas: The first is optimization of the transactions with respecting the SLA. The second is purely about the protocols used in the transactions.

As for the first area, when considering optimization of the transactions, we need to take into account the SLA-Aware and QoS-Aware approaches. In [18], the authors propose an integrated algorithm using both Transaction-Aware and QoS-Aware approaches. Based on their orientation, their model is expressed as an automaton about selectable Transactional Web Services and an algorithm for it is specified. However, their major concern remains around the algorithm at design time, and there is insufficient explanation of their architectural aspects at run time.

In regard to QoS-Aware, there is another approach related to the selection of services in Composite Web Service (CWS). This area has a relatively long history of research. For instance, in [19], the method of QoS-Aware CWS selection at run time is enhanced in order to realize global optimization combined with local selection technique. They propose a hybrid approach to reduce the cost of calculations for QoS optimization. However, selecting the approach while maintaining specified constraints from the SLA is treated as an indirect sub-thesis, although the selection of services in CWS is remarkable. Finally, the proposal of this paper is an extension of our previous work [8], in which a concrete architecture for integrating Transaction-Aware with the mechanism for maintaining the SLA categorized as QoS-Aware was proposed. Monitoring the current status of a transaction has a crucial role in the entire approach. However, that previous

work remains as a discussion how to select a reasonable approach for carrying out compensations instead of the whole of the processing. This is one of the main differences.

As for the second area, we need to limit the scope of our consideration because of the existence of a huge number of studies. Regarding the recovering failures during running long-lived transactions, there is a comprehensive survey by Colombo and Pace [20]. However, it already becomes outdated, because of emergences of RESTful and microservice. As mentioned before, our FCM has three phases named as Preparation, Execution and Adjustment. Substantially, it could be categorized as a reservation-based protocol. As for this area the study by Zhao, *et al.* is remarkable [21]. However, their study remains at the analysis level in demonstrating the advantages of the reservation based protocol. There are several factors that may impact the performance of the transactions from an architectural aspect, and our approach can give further insight.

Finally, we will touch on the several recent works including the microservice and elemental techniques in our approach. As for the compensation for a distributed transaction in microservice, there is a study by Limón, *et al.* [22]. They adopted Multi-Agent System to coordinate the compensation processes when a failure occurs. In spite of considering how to handle the complicated transactions with scalability, there are few explanations about how to maintain the isolation between transactions. The approach by Ding *et al.* includes the common elemental techniques with our proposal, such as OCC [23]. However, their focus is on the operation reordering rather than selecting the protocol for optimization as our major concern.

6. Conclusion

In this paper, we have proposed a new approach in which optimistic concurrency control is adopted for long-lived transactions with two verification phases. By using numerical simulation, it revealed the applicable conditions of the elemental transaction approaches and critical factors that dominate the performance in executing long-lived transactions. As for future work, we aim to define the protocol in a formal way and modernize them. Furthermore, we will clarify the influences of the topological relationships in service invocations.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Kikuchi, S. (2010) On Realizing Quick Compensation Transactions in Cloud Computing. In: Kikuchi, S., Sachdeva, S. and Bhalla, S., Eds., *Databases in Networked Information Systems. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 46-64. https://doi.org/10.1007/978-3-642-12038-1_5

[2] Kikuchi, S. and Bhalla, S. (2014) Quick Compensation Technique for Long-Lived

Transactions in Cloud Based Database Environment. IEICE Technical Report SC2013-19.1-5.

[3] Alrifai, M., Dolog, P., Balke, W.T. and Nejdl, W. (2009) Distributed Management of Concurrent Web Service Transactions. *IEEE Transactions on Service Computing*, **2**, 289-302. https://doi.org/10.1109/TSC.2009.29

[4] Garcia-Molina, H., Gawlick, D., Klein, J., Kleissner, K. and Salem, K. (1991) Modeling Long-Running Activities as Nested Sagas. *IEEE Data Engineering*, **14**, 14-18.

[5] Reuter, A., Schneider, K. and Schwenkreis, F. (1997) Contracts Revisited. Advanced Transaction Models and Architectures. In: Jajodia, S. and Kerschberg, L., Eds., *Advanced Transaction Models and Architectures*, Springer, Boston, MA, 127-151. https://doi.org/10.1007/978-1-4615-6217-7_5

[6] Grefen, P., Vonk, J. and Apers, P. (2001) Global Transaction Support for Workflow Management Systems: From Formal Specification to Practical Implementation. *The VLDB Journal*, **10**, 316-333. https://doi.org/10.1007/s007780100056

[7] Lee, J. and Son, S.H. (1993) Using Dynamic Adjustment of Serialization Order for Real-Time Database Systems. *Proceedings of the* 14*th IEEE Real-Time Systems Symposium*, Raleigh-Durham, NC, 66-75.

[8] Kikuchi, S. (2013) Architectural Design of a Compensation Mechanism for Long Lived Transactions. In: Madaan, A., Kikuchi, S. and Bhalla, S., Eds., *Databases in Networked Information Systems*. *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 49-66. https://doi.org/10.1007/978-3-642-37134-9_4

[9] OASIS WS-TX TC (2009) OASIS Standard Web Services Atomic Transaction (WS-Atomic Transaction) Version 1.2. http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec-os/wstx-wsat-1.2-spec-os.html

[10] OASIS WS-TX TC (2007) OASIS Standard Web Services Business Activity (WS-Business Activity) Version 1.1. http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-os/wstx-wsba-1.1-spec-os.html

[11] OASIS WSBPEL TC (2007) OASIS Standard Web Services Business Process Execution Language Version 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[12] Schäfer, M., Dolog, P. and Nejdl, W. (2008) An Environment for Flexible Advanced Compensations of Web Service Transactions. *ACM Transactions on the Web*, **2**, Article No. 14. https://doi.org/10.1145/1346337.1346242

[13] Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benrernou, S., Brandic, I., Kertész, A., Parkin, M. and Carro, M. (2013) A Survey on Service Quality Description. *ACM Computing Surveys*, **46**, Article No. 1. https://doi.org/10.1145/2522968.2522969

[14] Mihindukulasooriya, N., Esteban-Gutierrez, M. and Garcia-Castro, R. (2014) Seven Challenges for Restful Transaction Models. *Proceedings of the* 23*rd International Conference on World Wide Web*, Seoul, Korea, 7-11 April 2014, 949-952. https://doi.org/10.1145/2567948.2579218

[15] Pardon, G. and Pautasso, C. (2014) Atomic Distributed Transaction: A Restful Design. *Proceedings of the* 23*rd International Conference on World Wide Web*, Seoul, Korea, 7-11 April 2014, 943-948. https://doi.org/10.1145/2567948.2579221

[16] Mohan, C. and Lindsay, B. (1985) Efficient Commit Protocols for the Tree of Processes Model of Distributed Transactions. *ACM SIGOPS Operating Systems Review*, **19**, 40-52. https://doi.org/10.1145/850770.850772

[17] JBoss Transaction Service 4.2.3. (2006) Web Service Transactions Programmers Guide,

JBoss.

[18] Hadded, J.E., Manouvrier, M. and Rukoz, M. (2010) TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition. *IEEE Transaction on Service Computing*, **3**, 73-85. https://doi.org/10.1109/TSC.2010.5

[19] Alrifai, M., Risse, T. and Nejdl, W. (2012) A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints. *ACM Transactions on the Web* (*TWEB*), **6**, Article No. 7. https://doi.org/10.1145/2180861.2180864

[20] Colombo, C. and Pace, G.J. (2013) Recovery within Long-Running Transactions. *ACM Computing Surveys*, **45**, Article No. 28. https://doi.org/10.1145/2480741.2480745

[21] Zhao, W., Moser, L.E. and Melliar-Smith, P.M. (2008) A Reservation-Based Extended Transaction Protocol. *IEEE Transactions on Parallel and Distributed Systems*, **19**, 188-203. https://doi.org/10.1109/TPDS.2007.70727

[22] Limón, X., Guerra-Hernández, A., Sánchez-García, A.J. and Arriaga, J.C.P. (2018) SagaMAS: A Software Framework for Distributed Transactions in the Microservice Architecture. 2018 6*th International Conference in Software Engineering Research and Innovation* (*CONISOFT*), San Luis Potosí, Mexico, 24-26 October 2018, 50-58. https://doi.org/10.1109/CONISOFT.2018.8645853

[23] Ding, B., Kot, L. and Gehrke, J. (2018) Improving Optimistic Concurrency Control through Transaction Batching and Operation Reordering. *Proceedings of the VLDB Endowment*, **12**, 169-182. https://doi.org/10.14778/3282495.3282502