한국시뮬레이션학회 논문지
Vol. 16, No. 4, pp. 77-86 (2007. 12)

WORKGLOW: A P2P-based Web Service Orchestration Supporting Complex Workflow Patterns

Tran, Doan Thanh[1] · Hoang, Nam Hai[1] · Eunmi Choi[1†]

복잡한 워크플로우 패턴들을 지원하는 P2P 기반 웹 서비스 오케스트레이션

쩐도안타인 · 호앙남하이 · 최은미

ABSTRACT

Web services are considered as the critical component in the business plans of corporations as they offers the potential for creating highly dynamic and versatile distributed applications that span across business boundaries. Web Service Orchestration studies composition of already-existing web services to create new value-added services. The composite web services could be executed in a centralized or peer-to-peer(P2P) orchestration model. Compared with centralized-orchestration model, the P2P-based orchestration model provides better scalability, reliability, and performance for the overall services. However, recent P2P-orchestration solutions have limitation in supporting complex workflow patterns. Therefore, they could not effectively handle sophisticated business workflow, which contains complex workflow patterns. In this paper, we propose the WORKGLOW system, which can deal with complex workflow patterns while it is able to perform composite services in P2P orchestration manner. Comparing with centralized orchestration systems, the WORKGLOW brings up more business logic advantages, better performance, and higher flexibility with only a little overhead.

Key words : Web Service Orchestration, Workflow, Complex workflow patterns

요 약

웹 서비스는 비즈니스 분야의 경계를 가로지르는 매우 동적이고 다면적인 분산 어플리케이션 생성의 가능성을 제공하며, 차세대 기업의 다양한 비즈니스 계획에서 중요한 요소로 고려된다. 웹 서비스 오케스트레이션은 새로운 부가가치의 서비스를 생성하기 위해 이미 존재하는 웹 서비스와의 합성을 연구한다. 복합적인 웹 서비스는 중앙 집중형이나 P2P 오케스트레이션 모델에서 실행될 수 있다. 중앙 집중형 오케스트레이션 모델에 비하여, P2P기반 오케스트레이션 모델은 전체 서비스를 위해 더 나은 확장성, 신뢰성, 성능을 제공한다. 그러나 최근 P2P 오케스트레이션 솔루션은 복잡한 워크플로우 패턴을 지원하기에는 한계를 가진다. 그러므로 복잡한 워크플로우 패턴을 가지는 정교한 비즈니스 워크플로우에서는 효과적으로 다루어질 수 없는 한계가 있게 된다. 본 논문에서는, P2P 오케스트레이션 방법을 이용하여 복잡한 서비스 수행을 가능하게 하여 복잡한 워크플로우 패턴을 다루는 WORKGLOW 시스템을 제안한다. 중앙 집중형 오케스트레이션 시스템과 비교하여, WORKGLOW 시스템은 비즈니스 로직에 이점을 가져다 주고, 약간의 오버헤드만을 감안하며, 향상된 성능과 더 높은 유연성을 제공한다.

주요어 : 웹 서비스 오케스트레이션, 워크플로우, 복잡한 워크플로우 패턴

1. Introduction

Today's business is constructed from many business workflow processes. These business processes are chains of activities that are involved in delivering products or services. The activities chains may span across organi-

* 이 연구는 국민대학교 연구지원과 BK21 지원 연구결과임.
2007년 12월 3일 접수, 2007년 12월 15일 채택
[1] 국민대학교 비즈니스IT학부
주 저 자 : Tran, Doan Thanh
교신저자 : Eunmi Choi(최은미)
E-mail; emchoi@kookmin.ac.kr

zations resided all over the world. As the consequence of the mature of information technology industry, people are finding solutions that help enterprise automate these business processes not only within an enterprise scope but also across enterprises. In this situation, Web Service Composition technology appears as a promising solution. One topic of Web Service Composition research field is Web Service Orchestration, which focuses on business process creation based on orchestration. Orchestration always represents control from one party's perspective. The Orchestration term refers to an executable business process that can interact with both internal and external web services. The interactions, which include business logics and task execution orders, occur at the message level. They may span across applications and organizations to define a long-lived, transactional, and multi-step process model. There are two approaches to execute composite web services: centralized orchestration and P2P-based orchestration (P2P-orchestration). Currently, most of the commercial workflow management systems such as MQSeries, SAP R/3, Staffware, and MS Biztalk, use the centralized orchestration model. Compared with the P2P-based orchestration model, the centralized model is popular because it is easier to manage and develop, despite of the fact that the distributed orchestration style brings more benefits on scalability, fault tolerance, security, and the distribution of workload. Former research works of P2P-based orchestration have been studied. However, the prior proposed P2P solutions expose the lack of supporting for three groups of complex workflow patterns: (1) advanced branching and synchronization patterns; (2) multiple instances patterns; (3) cancellation patterns. In this paper, we propose a workflow execution that supports these three complex patterns. We also experiment this workflow in heterogeneous use-cases to verify the practical usage of our solution.

This paper is organized as follows: Section 2 reviews some concepts of workflow patterns and P2P-orchestration. Section 3 describes the WORKGLOW, the proposed workflow execution system that is capable of executes complex workflow patterns via the P2P-orchestration style. Section 4 presents the analysis on performance evaluation of the proposed solution. Conclusion is discussed in Section 5.

2. Related Work

This section reviews some concepts of workflow patterns and P2P-orchestration by showing two parts: the first part is about workflow pattern categorization; the second part is about SELF-SERV, which is the pioneer research project to execute composite Web Service in a P2P-orchestration manner.

2.1 Categorization of Workflow Patterns

According to the research work of W.M.P. van der Aalst[3,4,5,6], workflow patterns address business requirements in an imperative workflow style expression regardless of specific workflow languages. There are 20 commonly used workflow patterns, which can be categorized into six groups as shown in Figure 1. In these six categories, there are three pattern categories that are usually not supported by P2P-orchestration sys-tem: (1) Advanced branching and synchronization patterns; (2) Patterns involving multiple instances; and (3) Patterns involving multiple instances.

Advanced branching and synchronization patterns: These patterns transcend the basic patterns to allow more advanced types of splitting and joining behavior. An example is the Synchronizing merge (Pattern 7), which behaves like an AND-join or XOR-join depending

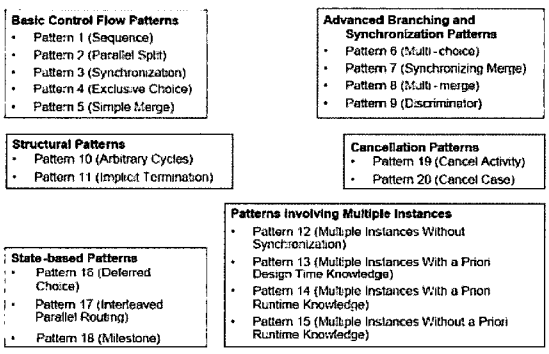

Fig. 1. Twenty most commonly used workflow patterns categorized in six groups

on the context.

Patterns involving multiple instances: Within the context of a single case (i.e., workflow instance) sometimes parts of the process need to be instantiated multiple times. For example, within the context of an insurance claim, multiple witness statements need to be processed.

Cancellation patterns: The occurrence of an event (e.g., a customer canceling an order) may lead to cancellation of activities. In some scenarios, events may even cause a withdrawing process of the whole rest of cases.

2.2 P2P-Orchestration Mechanism

The SELF-SERV[1,2,7] is the pioneer framework for dynamic and peer-to-peer provisioning of web services. In SELF-SERV system, composite services are executed in a P2P-style, which is coordinated by several peer software components called coordinators. Coordinators are attached to every task of a composite service and they are in charge of initiating, controlling, monitoring the associated tasks, and collaborating with their peers to manage service execution.

Knowledge required at runtime by each coordinator involving in a composite service (e.g., location, peers, and control flow routing policies) is statically extracted from a service operation's state chart at design time. After that, the knowledge is stored in a simple tabular form, called routing tables. Routing tables contains preconditions and post-processing information. Preconditions information is used to figure out when a service should be executed. Post-processing information is used to determine what should be done after service execution. Routing tables, which contain preconditions and post-processing data, are the heart of SELF-SERV system. Calculating routing tables does not require severe time consuming and is not too difficult to handle. However, preconditions and post processing information contained in routing tables is not enough for SELF-SERV to deal with three categories of complex workflow patterns: (1) advanced branching and synchronization patterns; (2) patterns involving multiple instances; (3) cancellation patterns.

This limitation results in the low flexibility in workflow design. For instance, let us consider a Travel Planning composite Web Service. In this service, the task Pay will be invoked when all the three booking services (FlightBooking, HotelBooking, and CarRental) return positive resulting values. When the FlightBooking Web Service returns negative values (it means that there is no available flight ticket), the execution of CarRental and HotelBooking web services need to be canceled as soon as possible. The travelers want to pay only for the full plan; it wastes money and is meaningless if they spend money on hotel booking without reserving the flight ticket for traveling. This is a common scenario that usually met in the business world. Unfortunately, SELF-SERV does not provide a direct way to cancel a part of the workflow at runtime as in this example.

The YAWL (Yet Another Workflow Language)[4] is the first solution that fully implements all prior-mentioned 20 common workflow patterns in a formal way. At first, the YAWL was developed by taking Petri nets as a starting point and adding mechanisms to allow more direct and intuitive supports of the work-

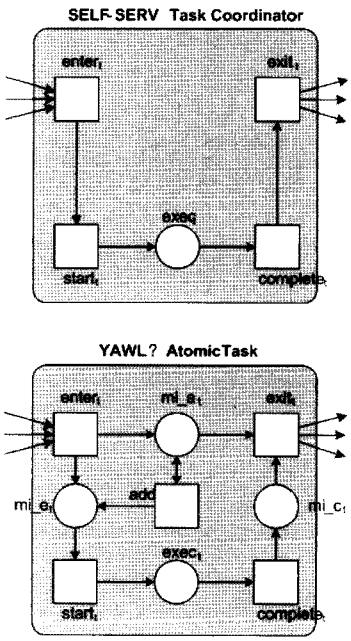

Fig. 2. Task-coordinator structure of SELF-SERV solution and YAWL solution

flow patterns identified. While most of the popular workflow languages support only 10 to 14 out of 20 most-popular identified workflow patterns, YAWL directly supports the full range, 20 patterns.

After all, YAWL is not designed to execute the composite web service in the P2P-orchestration style. Therefore, like others contemporary workflow system, YAWL system uses the centralized orchestration model which is easier to manage but has lower expansibility and higher workload at the orchestration point when compared with P2P-orchestration approach.

3. The WORKGLOW Solution

In this section, we present the WORKGLOW, which is a system that can deal with complex workflow patterns while it is able to perform composite services in a P2P orchestration manner. The WORKGLOW is built by synthesizing task's structure of SELF-SERV and YAWL system together. Besides, we work on additional mechanism to control execution of a workflow task in the distributed environment at runtime.

This powerful feature of YAWL comes from the innovation in task structure design and the Reset nets theory[8]. Figure 2 represents the structural differences between SELF-SERV's task coordinator and YAWL's atomic task. Compare with the SELF-SERV's, YAWL's atomic task is more complex; it has one "add transition" and three arrays more (mi_at, mi_et, and mi_ct), which are responsible for keeping information of all job instances. Descriptions of these arrays are listed in Table 1.

Table 1. Workload characteristics of application requests

SELF-SERV Task Coordinator	YAWL's atomic task
exect: keep track of the task is being executed	exect: keep track of the task is being executed
	mi_at: keep track of job instances already created
	mi_et: keep track of job instances that are waiting for execution
	mi_ct: keep track of job instances already completed

By inheriting this structure, the WORKGLOW task coordinator can perform multi-instance workflow patterns. However, because of the differences between centralized execution and P2P execution environments, WORKGLOW cannot inherit the ability to disable a part of workflow at runtime from YAWL. The proposed mechanism to disable a part of the workflow at runtime of WORKGLOW is described as follows:

1. While executing:

- Receiving control flow notifications from other task coordinators. There could be two types of notification message:
  - task complete
  - task cancel
- Whenever there is a job in the mi_et waiting list, the transition start will be enabled. When this transition fires, it consumes a token from mi_et and produces a token for exect.
- Transition complete occurs when the execution is completed. When this transition fires, it consumes a token from mi_et and produces a token for mi_ct.
- Transition exit occurs when all instances corresponding to the same parent have completed. The task coordinator also removes tokens from selected parts of the specification as indicated in its removing list. The number of tokens produced depends on the number of connections and the behavior of the split output specified in the workflow specification.

2. When "task complete" notification is received:

- Check with information from the workflow specification to determine whether it is time to execute the currently waiting task or not
- If it is time to execute the task, task coordinator will create a job instance and put it in mi_at and mi_et

3. When "task cancel" notification is received:

- Stop the current execution task
- Remove all the tokens from the input place

- Propagate the "task cancel" notification to children task coordinators which are executing

Aimed with the new added components variable inside and the new set of executing task-coordinator, the orchestration system now can execute all 20 commonly-used workflow pattern of six categories identified in Figure 1. As mentioned earlier, in P2P orchestration system, besides the executing the composed web service mission, the task coordinator need to handle the entire job that the central workflow engine in the centralized model usually do. This means in the real system, we need to add more components and functions for the task-coordinator. Implementation detail of the task-coordinator is coming next.

Figure 3 depicts the components inside the task-coordinator of the proposed orchestration system. For easy to refer, we call the proposed system as WORKGLOW. The coordinator has new components compares with the structure described in Figure 2. These new added components will make the task coordinator work with other application tools (such as: workflow definition tools, monitoring and administration tools, etc, showed in figure 3 as number one to five).

With this addition cancellation part for YAWL mechanism to execute composite web service based on P2P-orchestration, we have a complete WORKGLOW system that can support 20 commonly used patterns on P2P-ochestration based execution.

4. System Evaluations

This section presents the performance assessment of the WORKGLOW solution described in Section 3. In order to perform the experiments, we setup a prototype of the WORKGLOW system and use a business scenario called Complete Travel Service (CTS). There are two workflows that fulfill all the business requirements that CTS expected. The CTS version 1 uses pure basic workflow patterns, and it is very similar to the version 2 except for one difference.

As illustrated in Figure 4, the only difference between those two workflows is that version 2 workflow contains a cancellation pattern while the other does not. The semantic description of CTS is described as follows:

- The CTS is built based on 9 tasks: Register, FlightBooking, HotelBooking, CarRental, TravelInsurance, SpecialEquipmentService, AirportPickup, PaymentService, and Cancel.
- To use CTS, at first, travelers have to register their personal information by using Register service. The information should include name, address, phone number and credit card information, etc.
- There are three vital things to a travel plan: (1) the Flight ticket, (2) the Hotel Reservation confirmation, and (3) the Car Rental confirmation. If one or more of these things could not be arranged, the whole plan should be canceled.

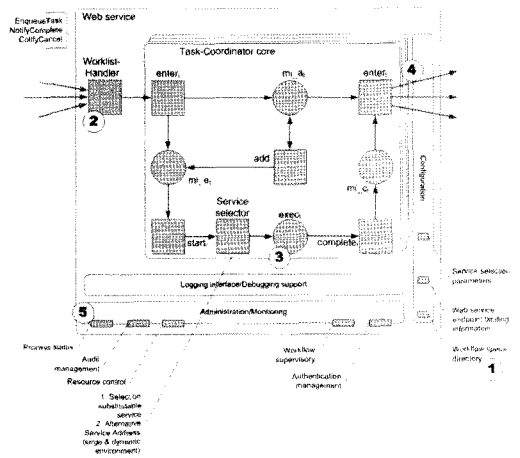

Fig. 3. Implementation structure of Task Coordinator

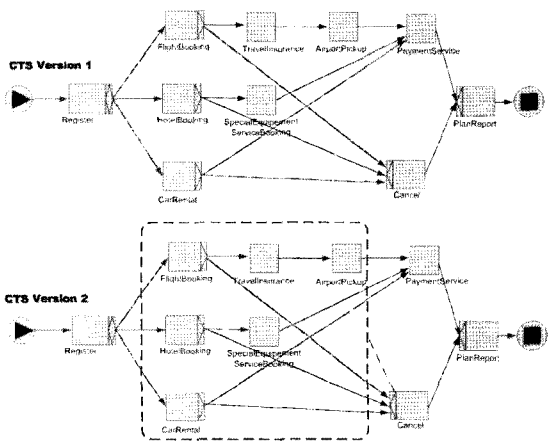

Fig. 4. Two versions of the CTS's workflow

- The TravelInsurance agreement, AirportPickup service, and the confirmation of Special Equipment Service should be considered. However, these things are not critical; they are just options of preference in a travel plan.
- The PaymentService task is called on demand. Only three tasks inside the CTS workflow could invoke the Payment service: (1) AirportPickup, (2) SpecialEquipmentService, and (3) CarRental tasks.
- The task Cancel should be executed at the moment the first booking task fails and it withdraws executing items in the remaining booking tasks.
- Canceling a booking that has completed successfully need to be compensated. For more specific, let us suppose that the penalty fee of canceling each booking service is 10%.

4.1 Experimental results of using complex workflow patterns

Because the only difference between two versions of CTS workflows is that one workflow contains a cancellation workflow pattern while the other does not. Therefore, by running these two workflows in the same hardware environment and configuration conditions, we can measure the effect of the cancellation mechanism used in the WORKGLOW on the overall performance of the system.

In order to conduct the performance test, we run the CTS version 1 and CTS version 2 workflows 100 times and measure log information of processing time, communication time, waiting time and message exchanged. In every test case:

- The processing time of each task is assigned randomly in the range of 650 to 750 milliseconds.
- All web services involved in the workflows are installed on 3 machines.
- All the branching conditions are set randomly.

The average processing time and the average number of messages exchanged when executing the CTS workflows using P2P orchestration style are summarized in Figure 5. As plotted in the figure, when executing the CTS without any complex workflow patterns (i.e., CTS version 1), the average number of messages used per

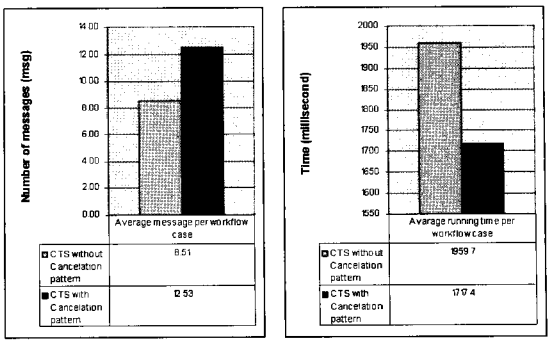

Fig. 5. Performance comparison of the CTS workflows w/ and w/o cancellation pattern

case is 8.51 and the average processing time is 1,959 milliseconds. When running execute the CTS with cancellation pattern (CTS version 2) in the same conditions, the average number of messages and average processing time per workflow cases are 12.53 and 1,717, respectively. As described before, the only difference between these two workflows is that one version contains a cancellation pattern while the other does not. The CTS workflow with cancellation pattern achieves about 14% better performance although it uses about 47% more message than the CTS without cancellation pattern. When executing the CTS workflow with cancellation pattern, if every time the Cancel task is invoked, it spreads out cancellation notifications to other peers which were listed in its removing list. Because the cancellation message is so small that it makes just a little computing overhead and networking traffic. In addition to this kind of direct cancellation, there are many cases that the task-coordinators were informed about cancellation before calling the elementary web services and invoking unnecessary tasks.

4.2. Experiment of homogeneous composite web services

In this experiment we measure running time, waiting time, communication time, and number of messages in a homogeneous system called Inter-Library search engine.

The results are measured in two cases: With Cancellation pattern and without Cancellation pattern. In this system, a number of universities' libraries join together and form an inter library search engine.

Through this system, a user can search a book from different databases of all the universities. When receiving a search request, the system searches in local university's databases and inquires to other universities to get the results. Figure 6 illustrates the workflow specification of this system.

The experimental environment is setup as follows:

- 100 tasks are executed simultaneously at the beginning of the experiment.
- The processing time of each task is assigned randomly in the range of 250 to 450 milliseconds.
- All web services involved in the workflows are installed on 3 machines.
- All machines are using Microsoft Windows 2000 server, Internet Information Server version 5.0, dot Net framework version 1.1 (Asp.net version 1.1).
- All machines use CPU Intel Pentium IV 3GHz.
- The random access memory of the machines is 512 MB RAM.
- Each machine is connected to a LAN through the 100 Mbits/sec Ethernet card.

Figure 7 shows that even in homogeneous cases, web service execution with Cancellation pattern achieved better performance in running time, communication time, and waiting time, compared to web service execution without Cancellation pattern while consuming just little overhead in number of transmitted messages.

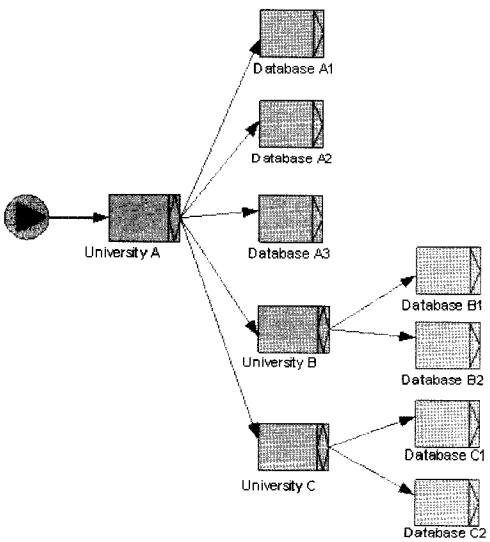

Fig. 6. A workflow specification for Inter-Library Search Engine

4.3 Experimental results of performance

WORKGLOW inherits the layered architecture and the P2P orchestration feature from SELF-SERV system. Therefore, when compared any system that uses the centralized orchestration style, WORKGLOW always has more advantage in term of system scalability, flexibility and workload distribution.

Normally, when creating a composite service, the address and syntax of the elementary web service must be explicitly known. Hence, the service that could be used in composition is restricted to existing ones. By inheriting Service Container and Membership modes concepts from SELF-SERV, WORKGLOW is able to select the most suitable web service to execute a task at run time. This means the web service used to execute a WORKGLOW's task could be add after the workflow design time and changing in the elementary web service is separated from the workflow specification.

A Service Container is a service that aggregates several other substitutable services - those that provide a common capability (the same set of operations) either directly or through an intermediary mapping. Service

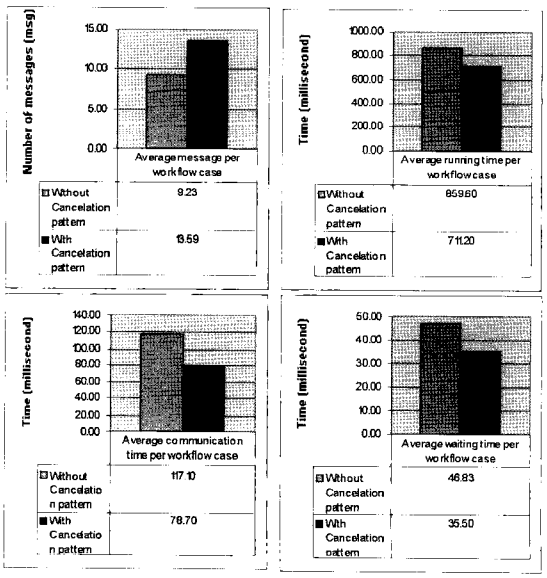

Fig. 7. Performance comparison in homogeneous cases

Container exists independently with the service that they aggregate. In essence, a Service Container also is a web service so that it can be invoked.

Take the workflow in Figure 8 as example. The HotelBookingContainer in this scenario aggregates the HotelBooking1, HotelBooking2, and HotelBooking3 service. The HotelBooking task has two options: (1) invoke a specific HotelBooking service that available or (2) invoke the HotelBookingContainer and let the Container service take responsible for selecting the most suitable service. HotelBookingContainer will process this multi-attribute dynamic selection by using the membership mode concept and a scoring service.

The purpose of this subsection's experiment is to compare the execution time and distribution of the P2P orchestration model with those of the centralized one. We measured the average processing time for a workflow case and the number of overall physical message exchanged across participant tasks. We setup the experimental environment to run the CTS with cancellation pattern (CTS version 2) in two situations. In both situations:

- The processing time of each task is assigned randomly in the range of 650 to 750 milliseconds.
- All web services involved in the workflows are installed on 3 machines.
- All machines are using Microsoft Windows 2000 server Internet Information Server version 5.0, dot Net framework version 1.1 (Asp.net version 1.1).
- All machines use CPU Intel Pentium IV 3GHz.
- The random access memory's configurations of the machines are different: one machine has 1GB while the other two have 512 MB RAM.
- Each machine is connected to a LAN through the 100 Mbits/sec Ethernet card.
- All the branching conditions for all workflow tasks are assigned randomly without considering dependency.

In the situation A, the workflow is executed using centralized orchestration style. The situation B is configured to run within the same condition of the situation A except that the workflow is executed using a P2P-orchestration style. For each of situations A and B, we run 100 workflow cases continuously. All information of execution time and the number of message processed at each point of the workflow are recorded.

As the results, the bar graph in Figure 9 shows the average processing times for a workflow case when

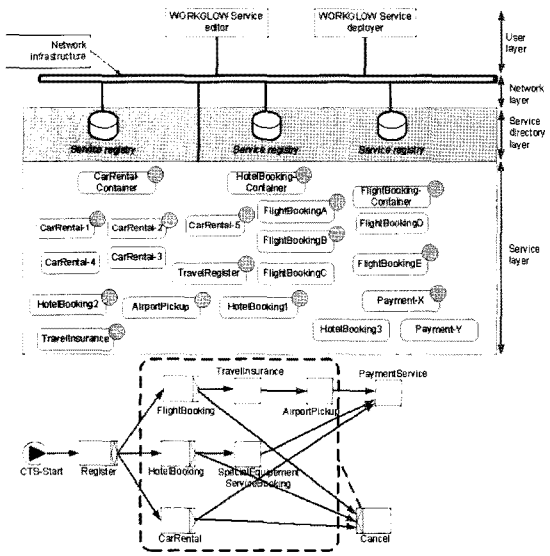

Fig. 8. WORKGLOW system architecture for executing CTS service

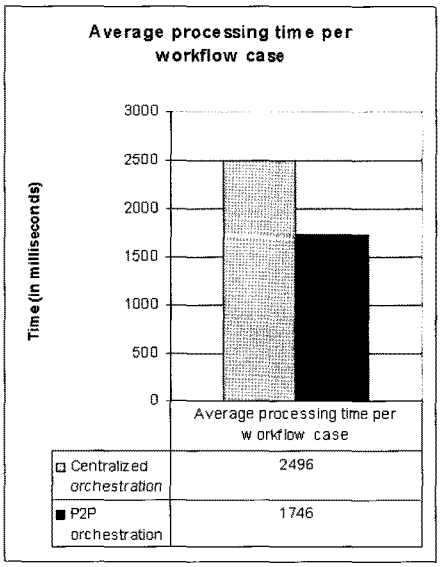

Fig. 9. Performance comparison of centralized orchestration and P2P orchestration styles.

performing the workflow using centralized orchestration style and P2P orchestration style. The average processing times in centralized orchestration style and P2P orchestration style are 2,496 and 1,746 milliseconds, respectively. With the benefit of P2P orchestration manner, the more computers participate in running the distributed workflow, the shorter the average execution time is.

5. Conclusion

This paper introduces basic concepts about web service composition. There are two ways to execute the composite web service: centralized and P2P approaches. Although the P2P execution approach provides more advantages over the centralized style in terms of scalability, availability, and security, current P2P execution solutions do not support three groups of complex workflow patterns (1) branching and synchronization group, (2) multiple-instance group, and (3) cancellation group. Thus, we propose the WORKGLOW system as a proper solution for this problem, which is built by combining the good ideas in system architecture design of two related works (YAWL and SELF-SERV) and defining a mechanism for canceling a part of the workflow specification in a distributed environment at runtime. As the result, the WORKGLOW system can handle all 20 identified common workflow patterns (including three complex pattern groups mentioned above), while it still preserves the scalability and the distributed execution features. The side effect of this cancellation mechanism is that the system created little overhead because of more notification messages generated. Nevertheless, choosing to have a clearer, more flexible and easier to adapt workflow and trade-off a little computation time is always a good option to consider.

References

1. Boualem Benatallah, Marlon Dumas, Quan Z. Sheng, Anne H. H. Ngu, "Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services", Proceedings of the 18th International Conference on Data Engineering (ICDE 02), 2002, pp. 297.
2. Marlon Dumas, Boualem Benatallah and Quan Z. Sheng, "The Self-Serv Environment for Web Services Composition", IEEE Internet Computing, 2003, pp. 40-48.
3. Wil van der Aalst and Kees van Hee, "Workflow Management Models, Methods, and Systems", MIT Press Cambridge 2002, 2002.
4. W. M. P. van der Aalst, L. Aldred , M. Dumas, and A. H. M. ter Hofstede, "Design and implementation of the YAWL system", Technical Report FIT-TR-2003-07, Centre for IT Innovation, QUT, 2003, 142-159.
5. W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: Yet Another Workflow Language (Revised version)"
6. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns", Distributed and Parallel Databases, 2003, pp. 5-51.
7. Boualem Benatallah, Quan Z. Sheng, Marlon Dumas "The SELF-SERV Environment for Web Service Composition", IEEE Internet Computing, 2003, pp. 40-48.
8. Moe Thandar Wynn, David Edmond, W. M. P. van der Aalst and A. H. M. ter Hofstede, "Achieving a General, Formal and Decidable Approach to the OR-join in Workflow using Reset nets", ICATPN 2005, 2005, pp. 423-443.
9. W. M. P. van der Aalst "The Application of Petri Nets to Workflow Management", Journal of Circuits, Systems and Computers, 1998, pp. 21-66.

Tran, Doan Thanh (thanhtd@kookmin.ac.kr)

2002 Hochiminh National University of Science, Telecommunication and Networking, B.S.
2006 Kookmin University, School of Business IT, M.S.
2006~Now Kookmin University, School of Business IT, Ph.D. Candidate

Areas of Interest: Grid computing, Ubiquitous Computing, Ubiquitous Sensor Network, Web Service Orchestration

Hoang, Nam Hai (emchoi@kookmin.ac.kr)

2002 Hanoi National University of Technology, Computer Science, B.E.
2006 Kookmin University, School of Business IT, M.S.
2004~Now IT Expert, UNIDO and Ministry of Planning and Investment of Vietnam

Areas of Interest: Web Service Orchestration, Message Queueing Services, Enterprise Architecture

최 은 미(Eunmi Choi) (emchoi@kookmin.ac.kr)

1988 고려대학교 컴퓨터학과 학사
1991 Michigan State University, Computer Science, M.S.
1997 Michigan State University, Computer Science, Ph.D.
1998~2004 한동대학교 전산전자공학부 조교수
2004~현재 국민대학교 비즈니스IT학부 부교수

관심분야 : 분산시스템, 미들웨어, 유비쿼터스 컴퓨팅, 시스템 아키텍처 모델링, 그리드 컴퓨팅