Grid Services Adaptation in a Grid Workflow*

Wencai Guo, Yang Yang, and Zhengli Zhai

School of information engineering, University of Science and Technology Beijing,
Beijing 100083, P.R. China
wc_guo@yahoo.com.cn

Abstract. At present the grid workflow is a focus topic in the research on grid technologies, but few approaches are provided for designing and checking the grid services adaptation in a workflow, especially in a formal way. As a mobile process algebra, Pi-Calculus is suitable tool for modeling and analyzing distributed, concurrent and dynamic system, such as grid workflow. Addressing the protocol level deadlock, a grid service adaptor is proposed with the Pi-Calculus based Grid Services Deadlock Eliminating algorithm. Through analyzing the experiment results the algorithm proves to be available and useful for service adaptation. As a conclusion the future research work is also discussed.

1 Introduction

In the recent years, Grid technologies are getting increasingly popular and have developed quickly to a degree of maturity. Grid is a distributed environment that enables flexible, secure, coordinated resource sharing, among dynamic collections of individuals and institutions. The benefits of embracing Web services on Grid have been recently realized in the Open Grid Services Architecture [1], which defines a service-oriented architecture for Grid systems. One of the compelling reasons for the acceptance of the web service model as the basis of Grid services is their ability to combine existing services into new ones, i.e. through workflow, in which its participant is usually a Web service or Grid service and its engine coordinates the execution order of tasks.

In the studies of grid workflow, some architectures have been provided, such as GSFL [2], OCGSA [3] and CCA [4]. But a major limitation of these architectures is that they do not present suitable means for describing and reasoning on the concurrent behavior of interacting service oriented systems. Indeed while these platforms provide convenient ways to describe the service interfaces and service data, they offer a quite low-level support to describe the concurrent behavior of service. For instance, system developers would like to determine beforehand whether the inclusion of a third-party service may introduce a deadlock into the application under development. In order to rigorously verify properties of system consisting of large numbers of dynamically interacting services, a formal description of the interactive behavior of services is needed.

Our present goal is to make use of a fair amount of work [5][6][7][8] currently available on the behavioral descriptions of components and web services at the

* This work has been supported by National Natural Science Foundation of China (No.90412012).

H. Zhuge and G.C. Fox (Eds.): GCC 2005, LNCS 3795, pp. 172 – 177, 2005.

protocol level, in order to apply many of such theoretical results to the practical field of the grid services. In particular, in this paper we show how the grid workflow can be checked whether two or more grid services are compatible to interoperate or not, and if not, whether the specification of adaptors that mediate between them can be automatically generated.

2 Protocol Level Grid Services Adaptation

In a grid workflow application, multi grid services have to coordinate to work according to certain rules, but all the service compositions are based on the successful message exchange between them. Otherwise, the service deadlock will happen. For instance, when a grid service S_1 sends a message to invoke a synchronous method on another service S_2, S_1 will block waiting for the method to complete. If S_2 attempt to call S_1 (invoke a callback) while S_1 is waiting for the original method to complete, then each party will be waiting for the other. This is a kind of deadlock. The same situation could occur if a callback handler attempted to call a synchronous method on its caller.

Protocol level adaptation [13] can be seen as an interoperability adjustment that deals just with the partial order between the services' exchanged methods, and the blocking conditions that rule the availability of the services. Of course it does not cover all the semantic aspects of services, but it is not weighed down with the heavy burden of semantic checks. At the protocol level the problems can be more easily identified and managed, and practical (tool-based) solutions can be proposed to solve them.

In OGSI [12], although any grid service has the common NotificationSource/ NotificationSink interface, its message sequence order is distinct due to its inherent behaviors or semantics, which results in the possible protocol mismatching between interactive grid services. As a solution, we propose a grid service adaptor (also a grid service instance) to realize the mapping between heterogeneous service protocols and then to eliminate the possible deadlocks.

3 Grid Services Adaptor

In figure 1, a grid service adaptor is located between two services. It is created by grid workflow coordinator and responsible for receiving or sending all the required messages of these two services in a deadlock free way. As a critical component, adaptor specification plays important role in protocol level grid service adaptation.

In order to describe an adaptor specification, a grid service is re-defined in following style:

GridService *ServName* = {
    signature   *input and output actions*
    protocol   *messages sequencing*
}

The **signature** interface of a grid service declares a set of input and output actions, that is, the set of messages sent and received by the service, representing the methods that the service offers and invokes, the values or exceptions returned, etc. Both input and output actions may have parameters, representing the data exchanged in the communication.

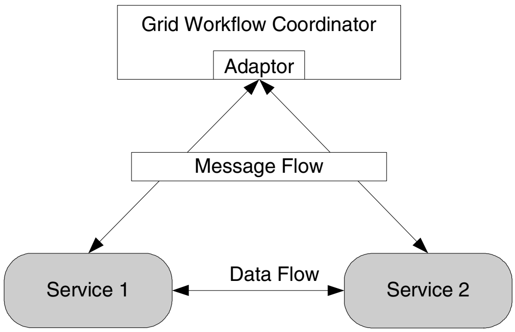

Fig. 1. Grid Services Adaptor

The **protocol** description of a service consists of sequenced messages, which reflects the essential aspects of the finite interactive behavior that a service may show to its environment in part. Compared with CCS[10], CSP[11] or other process algebra theories, the pi-calculus[9], allowing link names to be sent and received as values, is a very expressive notation for describing the behavior of grid services in applications with changing interconnection topology.

As an example, we assume a book borrowing service, *Library*, has been deployed in a library web and one student broker service, *Student*, sends a request for borrowing some book after passing the login authentication. Via Pi-Calculus, the service specification can be described as following:

GridService *Student* = {
  signature $Book\_request!()$, $reply?(answer)$;
  protocol $Book\_request!().reply?(answer).0$
}
GridService *Library* = {
  signature $Lib\_Query?()$, $Ack!()$, $Confirmation!()$;
  protocol $Lib\_Query?().\ Ack!().\ Confirmation!().0$
}

where the parameter *answer* is a data value either representing a confirmation or a refusal from a borrowing service.

Hence, we represent an adaptor specification by a mapping that establishes a number of rules relating actions and data of two services. For instance, the mapping expressing the intended adaptation for the previous example consists of the following three rules:

$$\begin{aligned} MA = \{ request!() &\quad \diamond \quad Query?(), Ack!(); \\ reply?("Confirmed") &\quad \diamond \quad Confirmation!(); \\ reply?("Refused") &\quad \diamond \quad Refusal!() \} \end{aligned}$$

where the messages in the left refer to the *Student* service, while those in the right refer to the *Library* service. The intended meaning of the first rule is that whenever the student outputs a request message, the borrowing service must eventually input one *Query* message, and output one *Ack* message. The rest rules have the similar meaning.

Given a mapping *MA* and the interaction pattern *P* and *Q* of two services, a concrete adaptor is generated by means of a fully automated procedure. Intuitively speaking, such an adaptor will be a component-in-the-middle *A* such that: 1) the parallel composition $P|A|Q$ will not deadlock, and 2) *A* will satisfy all the action correspondences and data dependencies specified by *MA*.

For such goals, the adaptor *A* is formed by eliminating gradually all the possible deadlocks that may occur in the evolution of the system.

4 A Pi-Calculus Based Grid Service Deadlock Eliminating Algorithm (PGSDE)

For removal of grid service deadlocks in a grid workflow, a Pi-Calculus based grid service deadlock eliminating algorithm is proposed in this section. The description of the algorithm is as following:

```
Input: a grid workflow W ;
Output: the complete adaptor specifications;
List all involving grid services of W, the amount of
them is n;
Adaptor A= 0;  bool isDeadlock = false; int i=0;
While (n>0){
  Select service(i);
  List all the messages required by activities;
  Perform the service protocol via pi-calculus;
  Find its partner services and produce all mappings MA
between them;
  while ( isDeadlock) {
    Reconstruct and expanded A[i] ;
    To check its correctness by theorems
    of Pi-calculus;
  }
  Output the adaptor A[i];
  n = n-1; i = i+1;
}
return all the adaptors
```

In this algorithm, all the adapters among the involved grid services in workflow *W* are produced by exhaustively eliminating possible deadlocks.

5 Performance of PGSDE Algorithm

Up to now, fewer studies on grid service adaptation are performed and less experiment results are reported in literatures. As an optional comparison, we choose web service composition method, Keyword-based Service Composition Method (KSCM) [14], to compare with PGSDE algorithm.

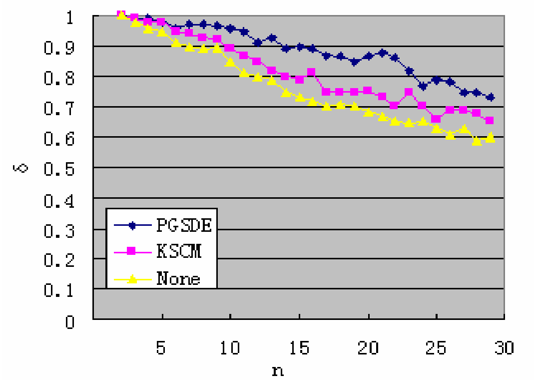

Fig. 2. Performance Comparison among PGSDE, KSCM and none adaptation

We define successful service adaptation rate δ as performance data. δ is defined as:

$$\delta(w) = \frac{Succ\_Message(n)}{Total\_Message(n)} \tag{1}$$

where w is the tested workflow application which is executed by the services numbered n, range over 2 to 30 , and messages numbered range over 10 to 300 . *Succ_Message(n)* and *Total_Message(n)* are defined as the number of successful message and the total service message required by n services in w.

Figure 2 gives a performance comparison among PGSDE, KSCM and none adaptation in grid workflow applications, which indicates that PGSDE is obviously better than the other two methods when the number of grid services in a workflow is more than 10. And we can also conclude that the number of failure requests increases with the increasing service number in a same workflow.

6 Conclusion

In order to reduce the deadlocks among services when a heterogeneous service is included, it's necessary to adapt the protocol interaction between the service and the existing ones. So a service adaptor in this paper is presented to meet such demands in concurrent, distributed and dynamic grid environment through eliminating protocol mismatching between services with a Pi-Calculus based deadlock eliminating algorithm (PGSDE). The analysis of PGSDE performance proves the adaptor is available and effective.

But in real world, such as in e-Commerce or e-Government fields, the service workflow process is more complicated, which results in more difficulties to describe and refine it in process modeling. Such complexities are from two basic properties of grid service: 1) the status of any service is dynamic during its execution, and 2) a service may play multiple roles in practice. Moreover, as stated in [15], flows in the future Internet environment can involve more rich semantics. Therefore, addressing these issues the future work should be extended to establish a flexible and robust semantics-rich adaptor, such as adaptor reconstruction once it fails to work properly.

References

1. Foster, I., Kesselman, C., Nick, J. and Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002. http://www.globus.org /research/papers /ogsa.pdf.
2. Wagstrom, P., Krishnan, S. and von Laszewski:, G.: GSFL: A Workflow Framework for Grid Services. In Supercomputing Conference (SC 2002).
3. Amin, K., Nijsure, S., von Laszewski, G.: Open Collaborative Grid Service Architecture (OCGSA). http://www1.bcs.org.uk/ DocsRepository/03700/3789/amin.htm, December 2002.
4. Govindaraju, M., Krishnan, S., Kenneth Chiu, Slominski, A. etc.: Merging the CCA Component Model with the OGSI Framework. Proceedings of CCGrid2003. The 3rd International Symposium on Cluster Computing and the Grid, http://www.extremeindiana.edu/ xcat/publications/xcatogsi.pdf, May 2003.
5. Bracciali, A., Brogi, A., Turini, F.: 2001. Coordinating interaction patterns. In: ACM Symposium on Applied Computing (SAC'2001). ACM Press.
6. Brogi, A., Canal, C., Pimentel, E., Vallecillo, A.: Formalizing Web Service Choreographies. WS-FM 2004 Preliminary Version. 2004.
7. Zhuge, H.: Component-based Workflow Systems Development. Decision Support Systems, 35(4)(2003)517-536.
8. Bracciali, A., A. Brogi and C. Canal, A formal approach to component adaptation. Journal of Systems and Software, 74(2005)45-54.
9. Milner, R., Parrow, J., and Walker, D.: A calculus of mobile processes, parts i and ii. Journal of Information and Computation, 100:1-77,1992.
10. Milner, A.J.R.G.: A Calculus of Communicating Systems. Lecture Notes in Computer Science, Volumn 92, Springer-Verlag,1980.
11. Hoare. C. A. R.: Communication Sequential Processes. International Series in Computer Science. Prentice Hall, 1985.
12. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., Maguire, T., Sandholm, T., Vanderbilt, P. and Snelling, D.: Open Grid Services Infrastructure (OGSI) version 1.0. Technical report, Open Grid Services Infrastructure WG, Global Grid Forum, 2002.
13. Yellin, D.M., Strom, R.E.: Protocol specifications and components adaptors. ACM Transactions on Programming Languages and Systems 19 (2), 292–333.
14. Thakkar S.: Dynamically composing Web services. Proceedings of AAAI Workshop on Intelligent Service Integration , Edmonton , Alberta , Canada , 2002 , 1-7
15. Zhuge, H.: The Future Interconnection Environment. IEEE Computer, 38 (4)(2005) 27-33