A WSRF Based Shopping Cart System

Maozhen Li[1,2], Man Qi[3,2], Masoud Rozati[4], and Bin Yu[1]

[1] School of Engineering and Design,
Brunel University,
Uxbridge, UB8 3PH, UK
{Maozhen.Li, Bin.Yu}@brunel.ac.uk
[2] Dept. of Computer Science,
Guangxi University of Technology,
Liuzhou, Guangxi, 545006, P.R.China
[3] Dept of Computing,
Canterbury Christ Church University College,
Canterbury, Kent, CT1 1QU, UK
[4] ANSARI GmbH,
Friedrich-Ebert-Damm 160,
D-22047 Hamburg, Germany
rozati@gmx.net

Abstract. Web Services Resource Framework (WSRF) is a set of specifications that represents a convergence point of the Web services and the Grid services communities. This paper presents our early experience with WSRF. A shopping cart system has been implemented with WSRF supported Globus toolkit 3.9.2 (GT3.9.2). Based on the system, the performance of the WSRF core in GT3.9.2 has also been evaluated.

1 Introduction

Based on Web services [1], Open Grid Services Architecture (OGSA) [2] becomes the de facto standard for building services-oriented Grid systems [3]. In the context of OGSA, a Grid service is a Web service with the following major extensions:

- A Grid service can be a transient service that can be dynamically created and explicitly destroyed.
- A Grid service is a stateful service that is associated with service data.
- Clients can subscribe to Grid services for notifications of events of interest.

OGSA merely defines what interfaces are needed, but does not specify how these interfaces should be implemented. It is the Open Grid Service Infrastructure (OGSI) [4] that defines how to technically implement these interfaces.

However, the Web services community has recently criticized the work on the extension of standard Web services in OGSI mainly because the OGSI specification is too heavy with everything in one specification, and it does not work well with

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 993 – 1001, 2005.

existing Web services and XML tooling [5]. In January 2004, the Globus Alliance [6] and IBM in conjunction with HP introduced the WSRF [7] to resolve this issue.

WSRF represents a convergence point of the Web services (WS) and Grid services communities. It introduces the concept of WS-Resource to model Web services with stateful resources [8]. WSRF is a set of specifications of which the following ones have been drafted:

- *WS-ResourceLifetime* [9] defines mechanisms for service requestors to request Web services to destroy associated WS-Resources immediately or after certain time.
- *WS-ResourceProperties* [10] defines the means by which the definition of the properties of a WS-Resource may be declared as part of a Web service interface.
- *WS-Notification* [11] defines mechanisms for event subscription and notification using a topic-based publish/subscribe pattern.
- *WS-BaseFaults* [12] defines an XML Schema for base faults, along with rules to specify how these faults types are used and extended by Web services.
- *WS-ServiceGroup* [13] defines a means by which Web services and WS-Resources can be aggregated or grouped together.

WSRF has been receiving more and more attentions from both Web services and Grid services communities since its first draft drew up in Jan. 2004. Work to fully implement WSRF specifications is still ongoing. For example, it is expected that a WSRF based Globus toolkit 4 (GT4) [14] will be available in Jan. 2005.

In this paper, we present our experience with WSRF based on the implementation of a shopping cart system with GT3.9.2 [15], an early work of GT4. Based on the system, the performance of the WSRF core in GT3.9.2 has also been evaluated.

The remainder of the paper is organised as follows. Section 2 presents the shopping cart system. Section 3 evaluates the performance of WSRF from the aspects of WS-Resource creation/destruction, service invocation, and WS-Resource property access. The evaluation has been performed using one computer and using two computers connected by a local area network. Section 4 concludes the paper and gives a discussion on the reliability of GT3.9.2.

2 A WSRF Based Shopping Cart System

In this section, we present a shopping cart system implemented with GT3.9.2. This prototype system is used for the performance evaluation of the WSRF core in GT3.9.2.

The shopping cart system supports multiple clients to access services. As shown in Fig. 1, a client uses SOAP [16] to access a shopping cart service through its interface defined in WSDL [17]. The cart is a stateful WS-Resource. The main components implemented in the system are described below.

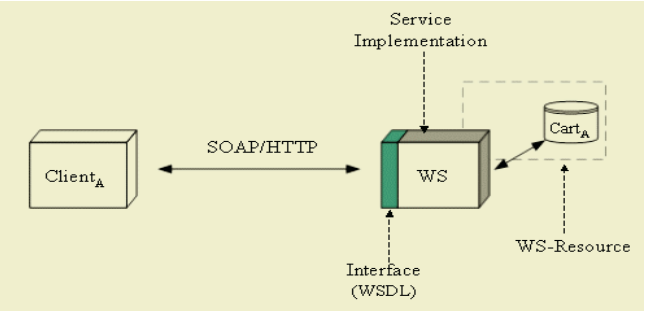

Fig. 1. A WSRF based shopping cart system

2.1 Cart

The Cart is a stateful WS-Resource that keeps the identifier of picked articles as well as the desired quantity of them. It has two properties: total price of the articles and the number of articles. We store the Cart and its properties in a file to make it a persistent WS-Resource (PersistentCart) in case of a failure. The cart ID, which is transparent to the client, is embedded in the WS-Resource qualified endpoint reference of the Cart and will be carried along in all message exchanges between the client and the server.

```
public class Cart
      implements ResourceLifetime, ResourceIdentifier, ResourceProperties,
TopicListAccessor
    {
        . . .
        protected Hashtable articles;
        protected ResourceProperty totalPrice;
        protected ResourceProperty articleCount;
        ...
    }
```

Fig. 2. The Cart class

As shown in Fig. 2, the Cart class implements *org.globus.wsrf.ResourceProperties* interface for an access of resource properties, *org.globus.wsrf.ResourceIdentifier* interface for resource identification, *org. globus.wsrf.ResourceLifeTime* interface for resource lifetime management. In addition, the Cart also implements *org.globus.wsrf.TopicListAccessor* interface for resource notification.

2.2 Cart Service

The Cart service (cartService) is the actual Web service component in the system. The *cartService* exposes two operations through its public interface:

- *addArticle*
- *removeArticle*

In addition, cartService provides two methods for creating and destroying the Cart WS-Resource and works as a factory service for the WS-Resource.

2.3 Cart Home

The stateful resources are deployed in GT3.9.2 container as Java Naming and Directory Interface (JNDI) [18] resources. A JNDI resource has a home class that is responsible for the creation and location of that resource. The Cart WS-Resource home class (CartHome) should implement the *org.globus.wsrf.impl.ResourceHomeImpl* interface. The *create()* method of the CartHome class is called whenever a request to create a new instance of this resource is received. Fig. 3 shows the sequences to create a resource.

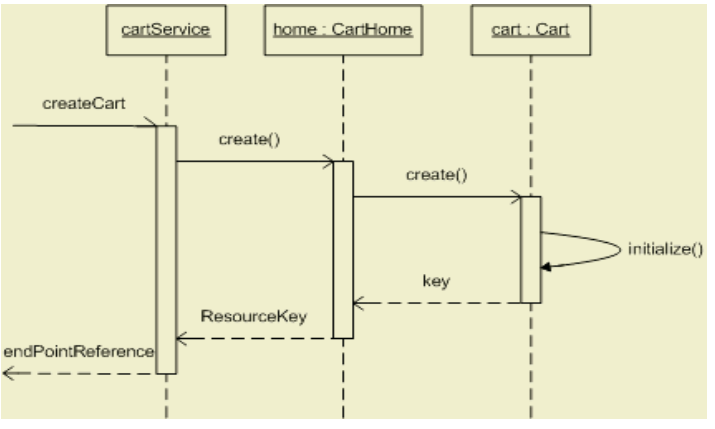

Fig. 3. Sequences to create a resource

2.4 Cart Client

A client uses the automatic generated stubs to locate the Cart Web service and send requests to create a WS-Resource and invoke operations upon it. GT 3.9.2 generates helper classes such as *ServiceNameAddressingLocator* that finds and returns the service proxy object using the service address (URI). In the shopping cart system, the CartPortType class is the proxy of cartService at the client side. Fig. 4 shows how a client uses *CartServiceAddressingLocator* to find the CartPortType and then request it to create a Cart WS-Resource.

To subscribe to a notification, the client should implement the *org.globus.wsrf.NotifyCallback* interface. The notification workflow at the client side is described as follows:

- The client receives an instance of *NotificationConsumerManager* and then invokes it. This object listens for incoming notifications.
- The client calls the *subscribe()* method of the Cart Web service. The subscription request contains the topic name of interest and the topic dialect the name uses. (GT 3.9.2 currently only supports *SimpleTopicDialect*)
- The public method *deliver()* will be called back whenever a notification is received.

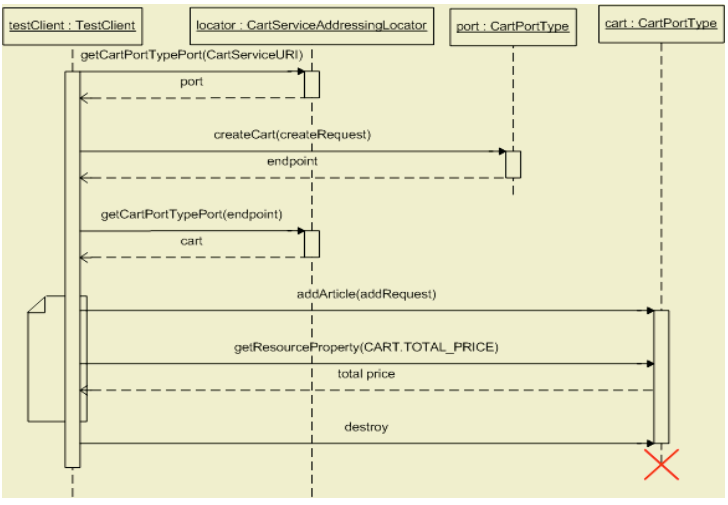

Fig. 4. Sequences for requesting a Cart service

3 Performance Evaluation

We have constructed two experimental environments for the evaluation of the WSRF based shopping cart system implemented with GT3.9.2. In the first environment, the Cart client and Cart service were deployed on the same computer (Intel Pentium III 850MHz, Windows 2000 Professional, 512 MB RAM). In the second environment, the Cart client and Cart service were deployed on two computers connected by a local area network with a bandwidth of 100Mbps. The client computer is an Intel Pentium III 550MHz with a memory of 256MB running Windows 2000 Professional. The server computer is an Intel Pentium III 850MHz with a memory of 512MB running Windows 2000 Professional. For our tests, we considered the single-server/multiple-clients situation as shown in Fig.5.

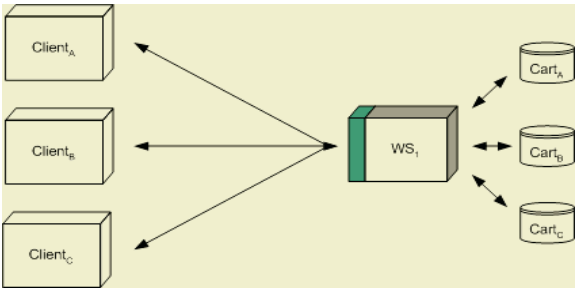

Fig. 5. Multiple clients access a single Cart service

The response time of the following four operations have been measured using variable number of clients:

- The creation of the Cart WS-Resource (CreateCart method)
- The destruction of the Cart WS-Resource (Destroy method)
- The invocation of the Cart service (addArticle method)
- The access of the Cart WS-Resource property (GetProperty method)

Clients can subscribe to the Cart service for notification of events of interest. The Cart service published two notification topics: ARTICLE_COUNT and TOTAL_PRICE. Clients who were interested in getting notified of changes of these values subscribed to these topics.

3.1 Performance Evaluation on One Computer Without Notifications

In this evaluation, the clients and the Cart service were deployed on one computer. The clients did not subscribe to the Cart service for any notifications.

As shown in Fig. 6, the creation of the Cart WS-Resource is the most time consuming step in the shopping cart system. This is because the initiations of WS-Resource need more time than other operations. Another reason is that we created a file for each Cart WS-Resource to make it persistent. When the number of clients tested is below 100, the time taken to invoke the Cart service and the time taken to access the WS-Resource property is roughly the same. However, as number of clients goes beyond 100, the time taken to invoke the Cart service is getting longer than that of the access of the WS-Resource property. Again, this can be explained due to file access in the AddArticle() method. The destruction of the Cart WS-Resource needs the least time among the four operations.

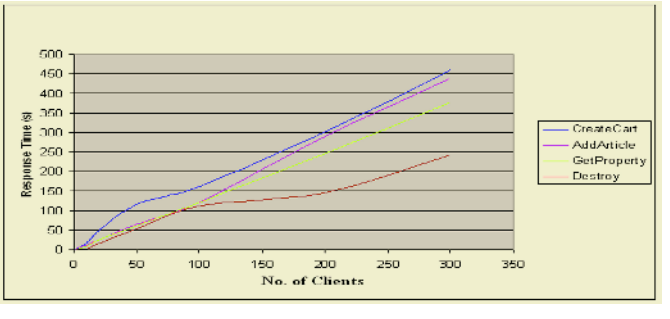

Fig. 6. Performance evaluation on one computer without notifications

3.2 Performance Evaluation on One Computer with Notifications

In this evaluation, the clients and the Cart service were deployed on one computer. The clients subscribed to the Cart service for notifications of events of interest.

As shown in Fig. 7, the use of notifications increases the overhead of the four operations. Again the creation of the cart WS-Resource is the most time-consuming operation among the four operations. However, compared with the evaluation de-

scribed in Section 3.1, the major difference here is that for a large number of clients, the destruction of the Cart WS-Resource needs significantly more time than before. Unlike previous tests without notification, we have experienced out of memory errors for 100+ clients. For example, in the test of 200 clients, 66% of the destruction requests were failed either due to the socket timeout or out of memory errors happened at the client side. Almost all tests with more than 200 clients in this evaluation were failed.

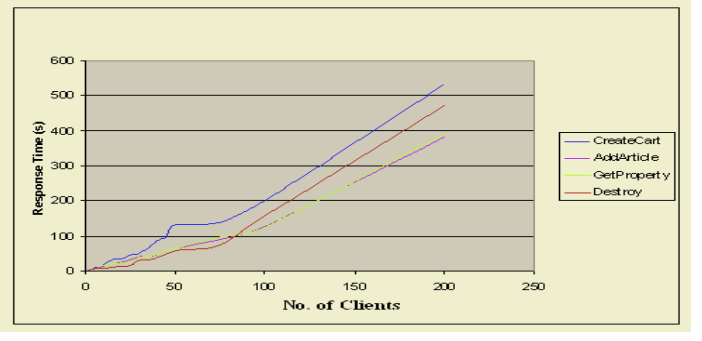

Fig. 7. Performance evaluation on one computer with notifications

3.3 Performance Evaluation in a LAN Without Notifications

In this evaluation, the clients and the Cart service were deployed on two computers connected by a 100Mbps local area network (LAN). The clients did not subscribe to the Cart service for any notifications.

As shown in Fig. 8, Similar to the evaluation described in Section 3.1, the time to create the Cart WS-Resource is the most time-consuming step among the four operations, and the destruction of the Cart WS-Resource needs the least time. However, compared with the evaluation described in Section 3.1, the time taken for each of the four operations performed in the LAN is less because the clients and the Car service were deployed on two computers using more resources than using only one computer.

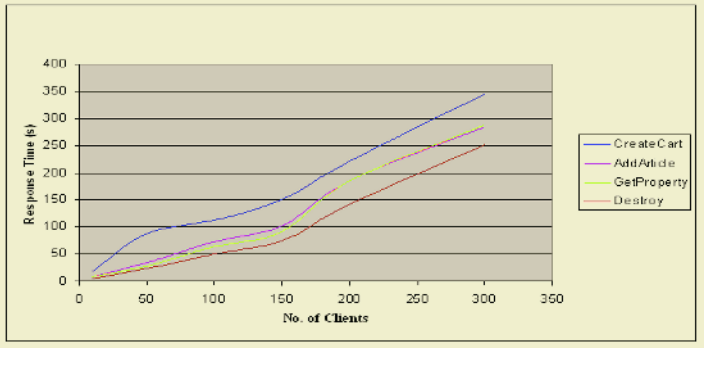

Fig. 8. Performance evaluation in a LAN without notifications

3.4 Performance Evaluation in a LAN with Notifications

In this evaluation, the clients and the Cart service were deployed on two computers connected by a 100Mbps local area network (LAN). The clients subscribed to the Cart service for notifications of events of interest.

As shown in Fig. 9, the use of notifications increases the overhead of the four operations. However, compared with the tests performed on one computer with notifications as described in Section 3.2, the overhead incurred by notifications in a LAN is less in each of the four operations.

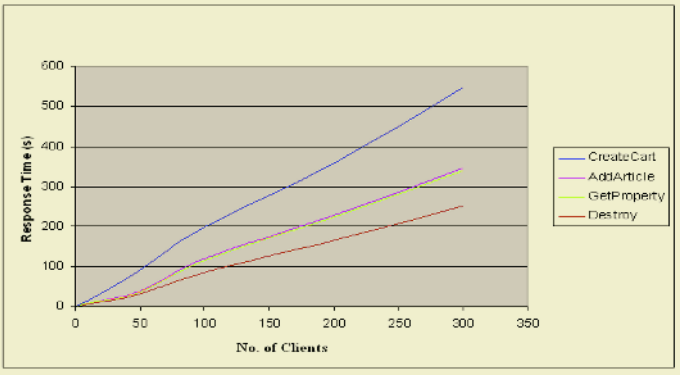

Fig. 9. Performance evaluation in a LAN with notifications

4 Conclusions

In this paper, we have presented a WSRF based shopping cart system implemented with GT3.9.2. Based on the system, we have evaluated the performance of WSRF in terms of WS-Resource creation and destruction, WS-Resource property access and Web services invocation. From the tests we know that, among the four operations, the creation of WS-Resource is the most time-consuming step, as it needs more time for initiations. Tests performed in a LAN have a better performance than that of the tests performed on one computer. The use of notifications increases the overhead of the four operations performed on one computer and in a LAN as well; however, the overhead incurred in a LAN is less than the overhead incurred on one computer as more computing resources are used.

The evaluation results have also shown a reliable container in GT3.9.2. Almost none of the tests performed at the server side crashed or incurred any errors. However, the client side incurred some errors when it had high load for opening connections for a large number of clients. When the number of clients went beyond 500, almost every destruction request ended up with a “socket read timeout” error at the client side. On the other hand, when the server had a high load, the notification callbacks were rarely received by clients even though they have subscribed to the events of interest.

References

1. Web Services, http://www.w3.org/2002/ws/
2. Foster, I., Kesselman, C., Nick, J., and Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, June 2002, http://www.globus.org/research/papers/ogsa.pdf
3. Foster, I. and Kesselman, C. (ed.): The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999.
4. OGSI Working Group, http://www.Gridforum.org/ogsi-wg/
5. Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Maguire, T., Snelling, D., Tuecke, S.: From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution, Version 1.0, Feb. 2004. http://www-106.ibm.com/developerworks/library/ws-resource/gr-ogsitowsrf.html
6. Globus, http://www.globus.org
7. Czajkowski, K., Ferguson, D. F., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W.: The WS-Resource Framework, Version 1.0, March 2004, http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf
8. Foster, I., Frey, J., Graham, S., Tuecke, S., Czajkowski, K., Ferguson, D., Leymann, F., Nally, M., Sedukhin, I., Snelling, D., Storey, T., Vambenepe, W., Weerawarana, S.: Modelling Stateful Resources with Web Services, Version 1.1, March 2004. http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf
9. Frey, J., Graham, S., Czajkowski, K., Ferguson, D. F., Foster, I., Leymann, F., Maguire, T., Nagaratnam, N., Nally, M., Storey, T., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W., Weerawarana, S.: Web Services Resource Lifetime, Version 1.1, March 2004, http://www-106.ibm.com/developerworks/library/ws-resource/ws-resourcelifetime.pdf
10. Graham, S., Czajkowski, K., Ferguson, D. F., Foster, I., Frey, J., Leymann, F., Maguire, T., Nagaratnam, N., Nally, M., Storey, T., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W., Weerawarana, S.: Web Services Resource Properties, Version 1.1, March 2004, http://www-106.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf
11. Web Services Notification, http://www-106.ibm.com/developerworks/library/specification/ws-notification/
12. Tuecke, S., Czajkowski, K., Frey, J., Foster, I., Graham, S., Maguire, T., Sedukhin, I., Snelling, D., Vambenepe, W.: Web Services Base Faults, Version 1.0, March 2004, http://www-106.ibm.com/developerworks/library/ws-resource/ws-basefaults.pdf
13. Graham, S., Maguire, T., Frey, J., Nagaratnam, N., Sedukhin, I., Snelling, D., Czajkowski, K., Tuecke, S., Vambenepe, W.: WS-ServiceGroup Specification, Version 1.0, March 2004, http://www-106.ibm.com/developerworks/library/ws-resource/
14. GT4, http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/GT4Facts/index.html
15. GT3.9.2, http://www-unix.globus.org/toolkit/downloads/development/
16. SOAP, http://www.w3.org/TR/soap
17. WSDL, http://www.w3.org/TR/wsdl
18. JNDI, http://java.sun.com/products/jndi/