RESEARCH ARTICLE OPEN ACCESS

Enhancement in Web Service Architecture

Prof. Seema Patil [1], Prof. Suman Tanwar[2], Prof. Manisha Tijare[3]
[1,2,3] (Department of Computer Science, Symbiosis Institute of Technology, Pune.

ABSTRACT
Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are increasingly used to integrate and build business application on the internet. Failure of web services is not acceptable in many situations such as online banking, so fault tolerance is a key challenge of web services. This paper elaborates the concept of web service architecture and its enhancement. Traditional web service architecture lacks facilities to support fault tolerance. To better cope with the fundamental issues of the traditional client-server based web service architecture, peer to peer web service architecture have been introduced. The purpose of this paper is to elaborate the architecture, construction methods and steps of web services and possible weaknesses in scalability and fault tolerance in traditional client server architecture and a solution for that, peer to peer web service technology has evolved.
Keywords – Web services, fault tolerance, client-server, peer to peer

I. INTRODUCTION

Web Service is a software system designed to support interoperable machine to Machine interaction over a network. Web Services can convert your application into a Web-application, which can publish its function or message to the rest of the world. Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the Word Wide Web.

The architecture of a Web Services stack varies from one organization to another. The number and complexity of layers for the stack depend on the organization. Each stack requires Web Services interfaces to get a Web Services client to speak to an Application Server, or Middleware component, such as Common Object Request Broker Architecture (CORBA), Java 2 Enterprise Edition (J2EE), or .NET.

Although there are variety of Web Services architectures, Web Services can be considered a universal client/server architecture that allows disparate systems to communicate with each other without using proprietary client libraries. This architecture simplifies the development process typically associated with client/server applications by effectively eliminating code dependencies between client and server" and "the server interface information is disclosed to the client via a configuration file encoded in a standard format (e.g.WSDL)." Doing so allows the server to publish a single file for all target client platforms.

Related Paper

The Web Services architecture describes principles for creating dynamic, loosely coupled systems based on services. There are many ways to instantiate a Web Service by choosing various implementation techniques for the roles, operations, and so on described by the Web Services architecture.

A mechanism, called CoRAL, provides high reliability and availability for Web service. CoRAL is client-transparent and provides fault tolerance even for requests being processed at the time of server failure. CoRAL does not require deterministic servers and can thus handle dynamic content. For dynamic content, the throughput of a server cluster is increased by distributing the primary and backup tasks among the servers. For static content, that is deterministic and readily generated, the overhead is reduced by avoiding explicit logging of replies to the backup. In the event of a primary server failure, active client connections fail over to a spare, where their processing continues seamlessly. [6]

The model in [7] describes a model by extensions of the SOAP standard and passive replication technique to achieve fault tolerance. This model carries out alterations on the WSDL document inserting information related to the primary replica and the backup replicas. It uses interceptors in the SOAP layer for redirecting of the requests to replicas in case of fault in the primary. of faults and replica management. This In the infrastructure of model, using interceptors is limited to fault detection in the infrastructure itself, allowing in case of faults on the primary WS Dispatcher Engine requests can be referred to a WS Dispatcher Engine backup.

The Construction of Web Service

Several essential activities need to happen in any service-oriented environment:

1. A Web service needs to be created, and its interfaces and invocation methods must be defined.
2. A Web service needs to be published to one or more intranet or Internet repositories for potential users to locate.
3. A Web service needs to be located to be invoked by potential users.
4. A Web service needs to be invoked to be of any benefit.
5. A Web service may need to be unpublished when it is no longer available or needed.

Web Services architecture then requires three fundamental operations: publish, find, and bind. Service providers *publish* services to a service broker. Service requesters *find* required services using a service broker and *bind* to them.

Web Service is a service-oriented architecture which is based on the interaction among service provider, service broker (service register center) and service requester. This interaction involves publishing, query and binding operation. The architecture of Web Service is shown Figure 1.

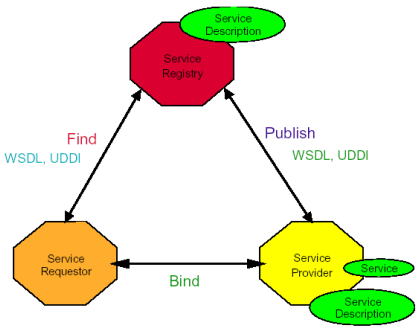

Figure 1 the architecture of Web service

Two main compositions of Web Service are:

Service:

Web Service is an interface described by service description and the implementation of service description is just the service. Service is software module and it is deployed on the platform which is accessed through Internet and provided by service provider. The purpose of service is to be called by service requesters or interacted with them. During the implementation of the service, if it calls other Web Services, it can be considered as a service requester.

Description of service:

Service description includes the interface of services and the details of their implementation such as the data type, the operation, the binding information and the network position of Services. It may also include the classification which is convenient for service requesters to discover and use as well as other metadata. Service description can be published to service requesters or service register centers. The construction of Web Service includes three steps:

1. The first thing is to construct the software modules which can provide services, which is usually implemented through component such as COM Component and .NET Component. These components are the core of Web Service and the implementation part of the services provided by Web Service.
2. Defining the description of service interface. The service description is regarded as the WSDL contract which is the most important information that can be provided by Web Service. The client application locates the Web method of some specific Web Service through WSDL document.
3. Web Service publishing. The process of publishing is similar to that of publishing a web site. In Windows operation System, Web Service is usually deployed on IIS (Internet Information server).

The construction of the components which can provide services is the key step. It determines the actual function provided by Web Service. However, viewed from the essence, the components in Web Service have no difference from those in ordinary programs. Therefore, the previous methods and technologies of component construction such as the object oriented technology; interface-oriented programming can be used for constructing the components in Web Service. As Web Service is independent on concrete programming language, so COM component developed with VB, VC can be considered as component of Web Service and .NET component developed with the programming language of .NET such as VB.NET, C# can also be implementation of services provided by Web Service. At the same time, tools can be used for defining the description of service interfaces, in other words, creating WSDL documents. For example, WSDL document of COM component can be generated by SOAP Toolkit, while .NET component's can be created by Visual Studio .NET integrated development environment. Finally, IIS virtual directory can be created on Web server to deploy Web Service. At this time, client can test and use the functions provided by Web Service.

Fault Tolerance Architecture For Web Services

In many situations such as online banking, stock trading, reservation processing, and shopping erroneous processing or outages are not acceptable, so fault tolerance is a key issue of web services. Fault tolerance makes to achieve system dependability. Dependability is related to some QoS aspects provided by the system, it includes the attributes like availability and reliability. Fault tolerance techniques are often used to increase the reliability and availability.

In client-transparent fault tolerance, the client sends the request and waits for the reply and the logic is handled on the server side. The proposed architecture is client transparent; clients communicate with Request Handler and are unaware of server replication. This architecture is involved six components and is fault tolerance even for requests being processed at the time of server failure. It is based on N-Version, active replications and logging request and reply messages at application and transport levels.

The proposed fault tolerant architecture for web services consists of six components: (1) Request Handler, (2) Logger, (3) Reply Handler, (4) Log Cleaner, (5) Fault Detector, (6) Replication Manager. Figure 2 illustrates the high level components of the architecture.

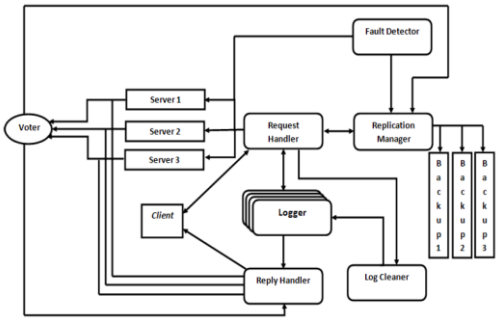

Figure 2 the fault tolerant architecture and its components

Request Handler:

This component is responsible for receiving requests from clients and transfers them to Logger. Request Handler will be sure of request logging by receiving an acknowledgment from Logger. It is responsible for sending requests to Servers after receiving the acknowledgment, too.

Logger:

It is consisted of four sub components: HTTP Request Logger, HTTP Reply Logger, TCP/IP Packets of Request Logger and TCP/IP Packets of Reply Logger. Figure 3 illustrates the sub components of Logger.

Reply Handler:

This component is responsible for transferring received replies from Servers and voter to Logger. Reply Handler will be sure of logging replies by receiving an acknowledgment from Logger, then sending replies to Client.

Log Cleaner:

This component removes the logged messages of requests that were terminated and their replies were sent to the clients.

Fault Detector:

It detects software and hardware failures and notifies to Replication Manager appropriately. The software failures can be detected by port scanning. For example, by checking whether a particular port is active (if an application server is up and running). The hardware or network failure is detected by using the Internet Control Message Protocol (ICMP protocol). ICMP echo requests are sent to each machine periodically. Fault Detector waits for a certain time period to receive a reply. It then resends the ICMP request and waits for a reply. If it does not get a reply to the resent request, Fault Detector decides that particular machine has failed.

Replication Manager:

It is responsible for maintaining replicated servers. For example, when Fault Detector notifies a failure to Replication Manager, it selects one of the backup servers instead of failure server.

This architecture carries out a set of web services. The web services run on different hardware, using different operating systems and different web servers. When Request Handler receives a request from a client, it is sent to all of the server replicas. Each replica computes the result independently and sends it to the voter. After the execution of a voting scheme, the reply is transferred to Reply Handler. Active replication, NVersion techniques and logging request and reply messages at application and transport level cause high availability and reliability. Operations of receiving, processing and sending reply of request are shown below.

Step Operation:

1 Client sends a request to Request Handler.
2 Request Handler accepts the request and transfers it to Logger.
3 TCP/IP Packets of Request Logger logs the request at transport level.
4 HTTP Request Logger logs the request at application level too.
5 Logger sends the acknowledgment of request to Request Handler.
6 Request Handler forwards the request to the servers and an acknowledgment to Client.
7 The servers process the request independently and forward the reply to Reply Handler.
8 Reply Handler sends replies of Servers to Logger.
9 TCP/IP Packets of Reply Logger logs the reply at transport level.
10 HTTP Reply Logger logs the reply at application level too.
11 Logger sends the acknowledgments of replies to Reply Handler.
12 After logging of replies, Servers send back replies to Voter.

13 After the execution of a voting scheme, Voter transfers the reply to Reply Handler.
14 Reply Handler sends the reply to Logger.
15 TCP/IP Packets of Reply Logger stores the reply at transport level.
16 HTTP Reply Logger logs the reply at application level too.
17 Logger sends the acknowledgment of reply to Reply Handler.
18 Reply Handler forwards the reply to Client with Request Handler's Address (Client only knows the address of Request Handler).
19 Client sends the acknowledgment to Request Handler.
20 Request Handler notifies Log Cleaner to removes the logged data about the request.
21 Log Cleaner removes the logged data about the terminated request.

If any fault or failure is detected by Fault Detector, the following operations are done:

- Fault Detector detects a failure (software or hardware failure) in one or more servers.
- Fault Detector notifies Replication Manager about the failure.
- Replication Manager selects one of the backup servers instead of failure server.
- Replication Manager notifies Request Handler, Reply Handler and Voter about the address of new server.

  The mechanism used to recover from server failures that occur during different phases, is as follows:
- In the event of a server failure, the backup server replica must be replaced by Replication Manager and continue providing service to Client, including handling of inprogress requests.
- When Server (or Servers) is failed, Request Handler does not receive acknowledgment from Client, Request Handler retransmits the request to Servers by using logged data of Logger. The replaced server receives the request but other servers ignore retransmitted request as duplicate.
- The replaced server processes the request and transfers the reply to Reply Handler for logging and next forwards it to Voter.
- Other servers ask Reply Handler for sending themselves logged replies and then forward them to Voter.
- After the execution of a voting scheme, Voter transfers the reply to Reply Handler and other described operations are done.

Peer To Peer Web Service Architecture

To better cope with the fundamental issues of the traditional client-server based web service architecture, peer to peer web services have been introduced and it has become a new research area in web service arena. There is no clear cut distinction between service providers and service consumers in peer to peer approach. This make the peer service model more dynamic and resilient. The fully distributed nature of the peer to peer architecture leads to minimize the central point of failure and each peer can communicate and provide services directly with each other without any centralized control.

This architecture allows flexible interaction among peer web services using the mobility behavior of the channels in MoCha (MOBILE CHANNEL)based peer to peer environment. The entire peer to peer architecture is built on top of MoCha middleware and peers communicate with each other through mobile channels in case of joining the network, service discovery and service invocation. Another desirable property that is to be achieved with this study is to introduce fault tolerance in order to deal with peer failure in web service invocation time.

Here the main focus is to introduce a distributed approach for web service publishing, discovery and use mobile channels to build up a flexible web service interaction mechanism.
Figure 3 depicts the layered view of the peer to peer web service architecture.

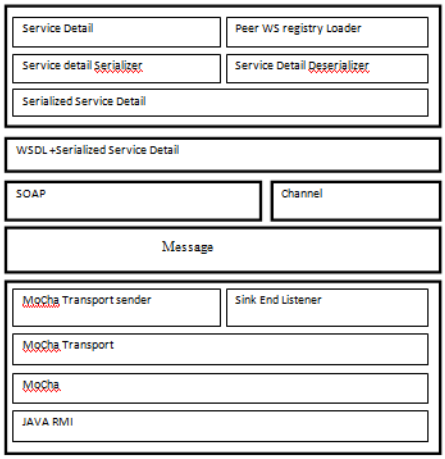

Figure 3 Layered view of MoCha based peer to peer web service architecture

1) Service Publish and Discovery*:*

Each service provider peer in the network maintains their own registry which contains their own services as well as services provided by their neighbors (any connected peer to a particular peer at a moment). The registry is a directory of the peers local file system. Each service is associated with a service detail file (Serialized Service Detail) which is stored in the service directory of the peers local file system. This file contains service specific information related to a service in XML format. The Service Detail Serializer creates this file when a

service is published. Each service provider peer maintains two data structures. One data structure contains the services provided by that peer. The other data structure is used to maintain the services provided by neighbor peers. A peer can request the service lists from other connected peer and store them in that data structure. This approach is efficient the service discovery process.

Any potential service consumer peer may first connect to a service provider peer and then it can request services from it. When a service provider peer gets a service lookup request, it first queries the data structure which contains the services provided by that peer. If the requested service is found within that peer the public interface (WSDL file) related to that service is sent to the service requester. If the requested service is unavailable within that peer, it queries the data structure which contains the services provided by the neighbor peers. If the service is found at any neighbor peer, the service lookup request is directed to that particular peer as it can respond to that request. This kind of behavior can be achieved due to the mobility nature of the connections among peers.

2) Service Description:

This specifies the description of a web service in XML format. As usual WSDL is used to describe the operations, data types and parameters related to a web service. The Serialized Service Detail for a given web service is used to persists the details such as service name, service class name, WSDL document name, WSDL document location of the peers local file system and access details of the service in XML notations. The Serialize Service Details will be also used in service deployment as well as in service invocation.

3) Messaging:

Here MoCha has been used as the transport medium for web service related interactions. The current version of MoCha supports any serialized object to be passed through the channels. Therefore Channel Message has been introduced as it provides serialized messaging format that can be transmitted through MoCha channels. SOAP is the most widely using messaging standard for sending web service invocation requests and responses. Therefore SOAP has been combined with Channels Messages in case of web service invocations.

4) Transport:

The peer to peer web service architecture is built on top of this MoCha based network infrastructure. So MoCha has to be used as the transport medium of the web service architecture, a depicted Figure 5.2. The MoCha Transport Sender is responsible in sending service invocation messages to the service provider in case of web service invocation. The Sink End Listener always listens to the sink channel end in order to read messages from its channel. The MoCha Transport defines the mobile channel based transport for web service invocations.

Web Service Replication For Fault Tolerance

In a peer to peer environment peers may not be up and running all the time. Unavailability of a particular peer may lead to unavailability of whole set of services provided by that peer. In order to deal with peer failures and unavailability of peers a web service can be replicated and deployed among several peers. The fault tolerance mechanism proposed here increases the availability of a service in service invocation time in the presence of peer failures and unavailability of peers.

When a consumer invokes a replicated service, first it invokes the service at the master service provider as depicted in Figure 4 If the master service provider has crashed or unavailable, then the consumer may iteratively invoke the service at replicated peers until it finds a live peer.

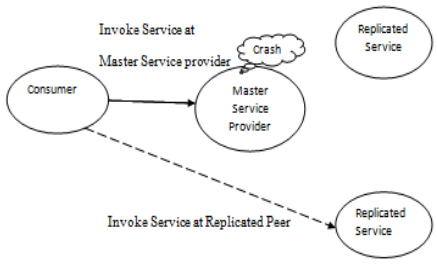

Figure 4 Web service replication and invocation

II. CONCLUSION

The increasing adoption of the web services needs efficient, scalable alternatives to the traditional client-server model for service discovery and invocation. As a solution for that client-transparent fault tolerance architecture for web services have evolved that correctly handles client requests, including those in progress at the time of server failure. It can recover requests being processed at the time of server failure by logging of request and reply messages Because of redundancy, it has overhead but provides high availability and reliability for web services. Peer to Peer web service architecture enables deploying, publishing, discovering and invoking web services in MoCha based peer to peer network infrastructure.

REFERENCES

[1]. Xuelei Wu, Jia Chen, Bilan Rong. "Web Service Architecture and Application Research", E-Business and Information System Security, International Conference, Page number 1 - 5, 23-24 May 2009.

[2]. Aghaei S., Khayyambashi M.R., Nematbakhsh M.A. "A Fault Tolerant Architecture for Web Services", Innovations in Information Technology (IIT), 2011 International Conference, Page number 53 - 56, 25-27 April 2011.

[3]. Sanjeewa P.A.A., Ranasinghe D. N. "A peer to peer web service architecture based on MoCha", Information and Automation for Sustainability (ICIAFs), 0 5th International Conference, Page number 207- 212, 17-19 Dec. 2010.

[4]. Zimmermann,M.; "A Web Service Architecture for VoIP Services", Internet and Web Applications and Services (ICIW), Fifth International Conference, Page number 445-450, 9-15 May 2010.

[5]. Ajlan Al-Ajlan, Hussein Zedan. "The Extension of Web Services Architecture to Meet the Technical Requirements of Virtual Learning Environments (Moodle)", Page number 27- 32, 2008.

[6]. Aghdaie N., Tamir Y., "CoRAL: A transparent fault-tolerant web service", The Journal of Systems and Software, 2008.

[7]. Wei Chuyuan, "The Exploration of Web Services Architecture and Implementation Mechanism", Journal, Aeronautical Computer Technique, Page number 103-104, 2003.