International Journal of Electronics Communication and Computer Engineering
Volume 4, Issue (6) NCRTCST-2013, ISSN 2249–071X

National Conference on Recent Trends in Computer Science and Technology (NCRTCST)-2013

Fault Tolerant Concurrent Service Composition in Service Oriented Architecture

M. Siva Mahesh
M.Tech-SE, Department of CSE, JNTUA, India
Email: perumahesh@gmail.com

***Abstract* – Web services are rapidly used in development of efficient business applications, in service transmission and integration of those on the internet. Service failure is not acceptable in an online reservation system, so the challenging issue in web services is fault tolerance. Fault tolerant service is possible by enhancing availability and reliability. This paper introduces architecture for fault tolerance in web services. The architecture focused on service transmission, application response by making request and response, multi version and dynamic reverberation methods. The developed architecture provides a fault tolerance, the requests can be possible while server is failed.**

***Keywords* – Fault Tolerant Process, Service Integration, Transmission Processing.**

I. INTRODUCTION

Web services are platform for accessing the data service in distributed form. The important distributed methods are mostly used in ecommerce, composition of various dissimilar systems; business processes management and web applications. The erroneous results in process of online reservation, banking will make the service unreliable. So the fault tolerance and recovery of failure is a important process in web services.

System dependability can be achieved in fault tolerance. It means when system is failed that shows defective impact on other systems. The service failure can be avoided by service composition techniques. System provides dependability is concerned with the aspects of Quality of Service, which contains availability and reliability. Availability means system will give response at any time. It mentions that the system is performing correctly at any time and performs its own functions.

The uninterrupted performance of a system is reliability. The more reliable system means which should not interrupt to erroneous process in making huge time based services. Fault tolerance techniques aimed to enhance the reliability and availability. A tiny work has done on the concepts of requirements of fault tolerance.

This paper developed a fault tolerant model. When the transmission in between client and server. If interruption is occurred, the process shouldn't get effect. Because Request Handler and Replication Manager will tolerate the fault occurred in process by activating the backup servers. The model has components like data process logs are used to store all the processes. When server is failed the fault detector sends message to Replication Manager then fault can be tolerated with Backup server by sending acknowledgement to Request Handler.

The paper is directed as follows: section 2 discusses related works. Section 3 describes the model and components. In section 4 is shown the evaluation and section 5 shows conclusions.

II. RELATED WORKS

The previous works based on connection establishment and maintaining the logs. The logs used only for logging and recovery shouldn't considered. Figure 1 depicts elements and message tracks. It uses two servers at each connection a primary and a backup. When the replies and requests are stored at backup TCP state is actively responded. The client transmissions all are stored at backup server before the primary server. Replies sent by the primary server are entered into the backup earlier than they enter into the client. So the backup can easily find out which acknowledgements are needed to send when the primary goes wrong.

The proposed model compared with past processes as given below.

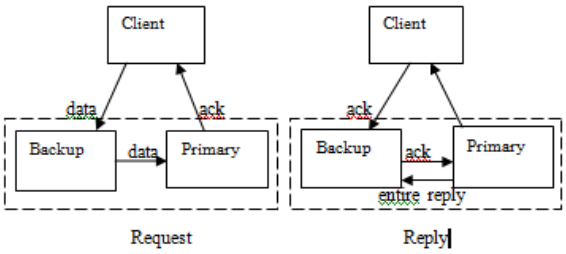

Fig.1. Message routing for connection acknowledgement and application logging

The proposed model compared with past processes as given below.

The primary or backup server is failed then acknowledgement sends to the client. In [1] comparisons and voting did not takes place but in the proposed system those can be considered as a part in the fault tolerance. In [2] and [3] client transparent are not accepted while in the developed architecture, clients make requests and wait for responses the total procedure is deal by the server. Even this model is transparent and fault tolerant is acceptable if the server is failed while requests are progressing, but it exceeds time boundary finally performance is decreased. In the fault tolerant mechanism redundancy causes the

overhead. The proposed model also has redundancy but in most of the cases overhead is allowed because erroneous process is not acceptable.

III. Fault Tolerant Model for Web Services

The components of the model expressed as follows:

A. Request Handler: Its purpose is used to handle the clients requests and transmits them to the Logger. The Logger sends the response to the Request Handler for ensuring the request logging. It is used to transmitting the requests to the servers and also for obtaining the acknowledgements.

B. Logger: It is contains four sub components described as follows.

a. *HTTP Request Logger:* Its purpose for logging HTTP requests. Application level stores the each request message by HTTP Request Logger.
b. *HTTP Reply Logger:* HTTP replies can be stored here. HTTP reply logger stores each replica message at application level.

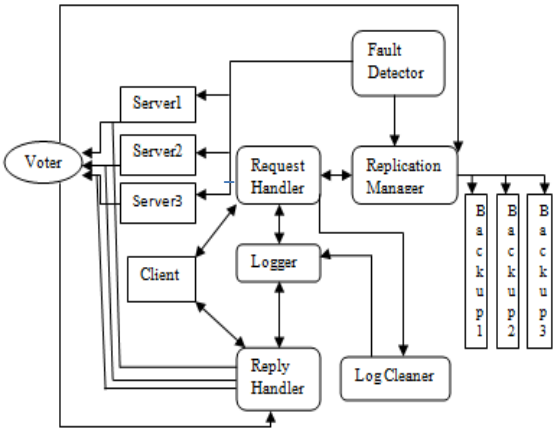

Fig.2. Components of the fault tolerance model

c. *TCP/IP Packets of Request Logger:* The requests of TCP/IP packets can be stored. It stores each request message at transport level.
d. *TCP/IP Packets of Reply Logger:* The replies of TCP/IP can be logged here. Reply Logger used to store each replication message at logger.

C. Reply Handler: The received acknowledgements from server can be transmitting to voter later to Logger. The logger sends acknowledgement to Reply Handler for storing replicas later transferring replicas to client.

D. Log Cleaner: It dumps the message logs of ended transactions like request and reply messages take over between client and the components.

E. Fault Detector: It indicates failure of software and hardware to the Replication Manager. The scanning of port used to identify the software failures. Example: to identify whether a network is failed or not by making use of Internet Control Message Protocol. Each network receives the echo requests of Internet Control Message Protocol periodically. Fault Detector holds for some period of time to receive a response After It retransmits the ICMP request and waits for a acknowledgement. If the response of the retransmit request doesn't achieve then Fault Detector determines that network gone wrong.

F. Replication Manager: It is used for managing replicated servers. Example, if the failure is achieved by Fault Detector then it alerts the Replication Manager about the failure. Replication Manager is responsible for choosing backup server to continue the process of previous failed server.

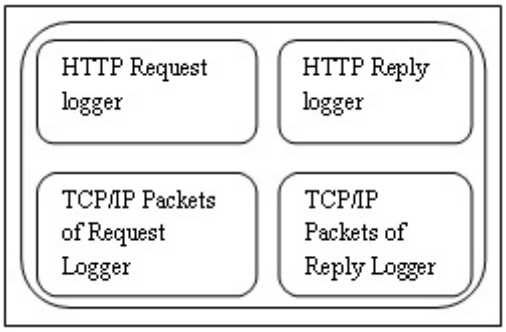

Fig.3. Elements of the logger

In the given model web services carried by various operating systems which uses various hardware. The client makes a request to the request handler which transmits to the whole server replicas. The evaluation took part at individual replicas later transmits to the voter. The voter executed the received replica messages then it sends to reply handler. The enhanced availability and reliability achieved by dynamic replication, multi version methods and maintaining the logs at transport and application level.

Operation flow:

1. First operation starts by making client's request to Request Handler.
2. Request Handler transmits to the Logger by accepting the client's request.
3. Request Logger holds TCP/IP Packets which stores the requests being performed at transport level.
4. HTTP Request Logger used to logs the request at application level.
5. Request Handler receives the acknowledgement of the request by logger.
6. Servers receive the request from the Request Handler and it makes an acknowledgment to the Client request.
7. The servers are individually proceeds the request and the reply transmit to Reply Handler.
8. The results of servers transmit over the Reply Handler to logs at Logger.
9. Reply Logger holds TCP/IP Packets which stores the reply being performed at transport level.

10. The replies stores in HTTP Reply Logger at application level.
11. The replies of reply handler stored at logger and return the acknowledgments to Reply Handler.
12. The replies stored as logs later voter receives the replies sent by servers.
13. After the voter executes and reply can be transmitted to the Reply Handler and then to Logger.
14. Reply Logger holds TCP/IP Packets which stores the reply being performed at transport level.
15. The replies stores in HTTP Reply Logger at application level.
16. The replies of reply handler stored at logger and return the acknowledgments to Reply Handler.
17. The client receives reply of the reply handler with the address of request handler. (Only client have the address of request handler).
18. After receiving the reply handler's reply at client, Request handler receives the acknowledgement of client.
19. The logged data should be removed from the Log Cleaner after receiving the notification of Request Handler.

If the Fault Detector finds out any faults then the following steps are performed:

1. The servers examined by Fault Detector to find out failure in servers.
2. After detection of the failure Replication Manager receives a message from Fault Detector.
3. Replication Manager chooses the backup server which acts as server in place of previous one to overcome the failure.
4. The Replication Manager transmits about new identity of server to Request Handler, Reply Handler and Voter.

The failure of a server can be recovered at several steps given below:

1. Replication Manager is responsible for replacing the backup server in place of failed one. The processing requests of client can be handled.
2. The client cannot transmit an acknowledgement to the Request Handler while server is failed. The request retransmission made to servers by activating logged data. The request achieved at backup server but rest of servers considered as duplicate.
3. The requests can handle at replaced server and Reply Handler receive the reply for maintaining logs and transmits to the voter.
4. At the voter it executes and transmits the result to the Reply Handler. After the other operations are performed. The described model in this paper contains both strengths and defects. Fault tolerance and client transparent is a strength, when the server is failed that doesn't shows any bad impact to its client, so high reliability and availability is possible. The weakness is redundancy, when server is failed retransmission from backup server to all servers will make redundancy.

IV. RESULTS

The results based on tolerating the fault from failed services. Each service provider provides three services are year, month, string in reverse. The execution steps represents as follows:

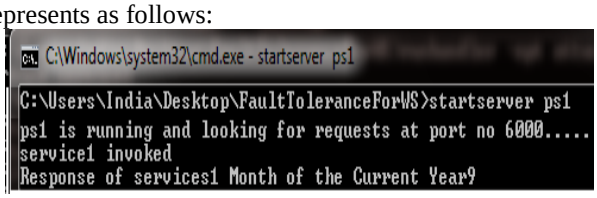

Fig.3. Response of service provider 1

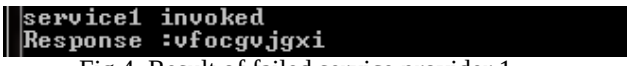

Fig.4. Result of failed service provider 1

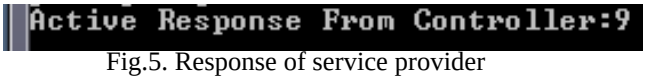

Fig.5. Response of service provider

Fig.6. Results of failed service providers 1, 2

Figure7. Fault tolerated response from backup server

1. Service one gives result as ninth month.
2. Service two shows 2013.
3. Service three gives string reverse to 'and' as 'dna'
4. While making a request to service one if service provider is failed then the most common results of other providers can be transmitted as ninth month.
5. The fault in service provider of service one tolerated by transmitting fault handling message to service provider.
6. If service one is failed in all providers then the result shown from the cache, in the same way fault can be tolerated in the other service providers.

V. CONCLUSIONS

This paper proposes fault tolerance model, when client make a request to the server if server is failed at that moment. The client received a response and process can be continued without any disruption. The model holds seven components having replication, log maintenance and backups, which tolerate the faults by logging all client server processes at logs. The failed server can be replaced with backup server. The redundancy causes overhead but efficient availability and reliability can be achieved.

REFERENCES

[1] Liusheng Huang and Mingjun Xiao, "FACTS: A Framework for Fault- Tolerant Composition of Transactional Web Services", In IEEE transactions on services computing, vol. 3, no. 1, 2010.

[2] Zibin Zheng and Michael R. LyuA. , "Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services", IEEE International Conference on Web Services, 2008.

[3] Jim Lau, Lau Cheuk Lung and Joni da S. Fraga., "Designing Fault Tolerant Web Services Using BPEL", Seventh IEEE/ACIS International Conference on Computer and Information Science, 2008.

[4] Zhao, W., "BFT-WS: a byzantine fault tolerance framework for web services", In: Proceedings of the Middleware for Web Services Workshop, Annapolis, MD, 2007.

[5] Aghdaie, N., Tamir, Y., "Performance optimizations for transparent fault tolerant web service", In: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada, pp. 29–32, 2003.

[6] Y.Xiao and S.D.Urban, "Using Rules and Data Dependencies for the Recovery of Concurrent Processes in a Service Oriented Environment," IEEE Trans. Service Computing., vol. 5, no. 1,pp. 59-71, Mar. 2012