SOA Distributed Bundles with QoS-aware

Sutheera Puntheeranurak
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
kpsuthee@kmitl.ac.th

Aekkawit Chanpen
Dept. of Information Engineering,Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
ekawit.ch@gmail.com

***Abstract*— Service replication is leveraged in a Service-Oriented Architecture (SOA) system for distributing services to different servers. It prevents the system from bottle neck and overload problem to improve the availability and reliability of the system. For this work, we focus on dynamic replication that needs an intelligent agent to control the replication procedure. It doesn't only improve the availability and reliability of the system, but it also optimizes resource usage on each server. In this paper, we discuss and propose the QoS-aware platform that aims to monitor the quality of services and decide to start the distributing function service. It makes the replica is deployed properly when a system's load is exceeded the capability of system service.**

Keywords- Service-Oriented Architecture (SOA), Distributed bundle, QoS-awareness

I. INTRODUCTION

The availability and reliability of Service-oriented Architecture (SOA) is an important indicator to measure performance of a business process. There are a lot of approaches that propose a fault-tolerant system by preparing alternative services for the consumer's invocation to handle centralized conflict. They discussed in a B2B model [13] and self-serve [16]. It also needs several services to replicate for improving availability of service composition. The self-organization platform [1] proposes a fault-tolerant solution to avoid vulnerabilities or single points-of-failure in the system. This platform introduces a solution by extending functions for service replication and deploying services on the core system of SOA in each node. Thus, the system has various services to handle consumer invocation, and the business process keeps going on even a service fails. In addition, the Double Redundancy [7] proposes a system which provides two instances for each service. This platform chooses two instances from the service list to handle routing request. The results from these services are compared to make sure that the operation is correct. The QoS-aware service aggregation model [2], [12] postpones the service selection until runtime, and the service consumer will acquire the most proper service that fits to the QoS requirements. Similarly, the service selection algorithm will be evaluated and assigned. It depends on QoS requirement.

In the previous work, A distributed bundle platform [3] propose to replicate required service bundles and deploy them to local communities. It provides server registration function to keep file access information. Then service bundle downloading process will be manipulated automatically when a consumer on the local community tries to invoke the remote service from another community. Eventually, the service lookup function is evaluated real-time and the service that has the good response time is sent to the consumer. However, the discussion is not covered the solution to assess the proper solution to start replication process and how to find the best server and service for the replication.

This paper proposes an extension of the distributed bundle platform that provides additional functionality to manipulate QoS, detect crisis of service in real-time and handle the system automatically. Our proposed platform uses the QoS information to detect critical situations of each service and start to download the service bundle if the processing time of service is greater than the defined threshold. In addition, it prepares for replication by checking the most famous service and deploy on a server that has least service replicas and a little workload. Therefore, the service selection brings load balancing technique for sharing the system's load on a different server equally.

II. RELATED WORK

Service based architecture is a famous model for software design and implementation, and OSGi is a service component based technology. This framework provides an execution platform for the service component, called bundles. And it enables the platform to download, install, start, stop and update service bundles at runtime without impact to the rest of the system. The Home Networking Service works [4], [5], [6] leverage the OSGi platform with a home network to monitor and control home's devices. The device in this system allows downloading, installing and sharing services. Nevertheless, this OSGi technology focuses only on services on device environment. Furthermore, it doesn't have a mechanism to evaluate and manipulate the replication mechanism.

In the web service environment, there are some replication solutions to distribute the service on difference server machines that will improve the availability and reliability of the system. In a data distributed work [9], it presents an advantage of replication that is the most replication and the less service latency. This work also discusses various strategies and shows the Fast Spread strategy which replicates the data to all nodes without time

and space consideration, and it uses the smallest latency. While the Cascade strategy is a replication that replicates the data on a server that has the shortest path. And this solution is the most efficient replication strategy with resource consideration to reduce latency. Our approach creates an algorithm to consider machine resource and evaluates service's loads on each node before the start replication process that's similar to Cascade strategy. However, we use QoS and service queue information instead of hop count.

In Distributed Replication and Caching [8], this system employs Java Data Object (JDO) to store persistent information that avoids accessing to the database, and it improves significant application performance. Moreover, it comes with a replication strategy to deploy service replica on multi-node clustering system. It then presents several service selection algorithms to handle the consumer request and optimize the response time of the invocation. However, this proposed doesn't present an algorithm to estimate service QoS and replicate the service dynamically. While a Dynamic Service Replication [10] proposes an advance algorithm that is leveraged in the distributed data mining grid. Service in this system is selected and replicated independently across the server machine to improve service's QoS with load balancing and achieve fault-tolerance requirement. In service selection, this platform uses Listing Table and Waiting Table to manipulate load balancing. And service that has the shortest service queue is offered to consumers. In addition, this replication process, they defined a threshold for operation time of service. If service executed, and its operation time is over the defined threshold value, then the service replication process is activated. Even so, this approached proposes a system that has only a central server to assess service QoS and process the service selection mechanism that may be a cause of points-of-failure.

A QoS-Aware Load Balancing [14] proposes a service scheduler with a simple efficient load balancing algorithm. The scheduler manages the job queue for service requests, updates QoS of each service and provides services selection function. Job service will be added to the job queue at the beginning of the invoking process. Then that job is removed after the service operation finished. In service selection aspect, it will choose a service by balancing service invocation to each server. While our proposed platform employs QoS-aware and load balancing to manipulate the service selection process. And this technique is also leveraged in service replication as well to separate service instances to the different server properly.

III. OUR APPROACH

Many replication works with SOA system propose solutions, which distribute a service on different servers to improve availability and reliability of the service. It shows that the more replicas have the best response time, and it also prevents the single point-of-fails problems. However, in the real system, the replication mechanism consumes a lot of network and machine resources, while these resources are limited, and it will impact to the system if resource usage excess the system capacities. Thus, the platform offers a solution to start the replication process only if current service instants are not enough to handle the consumer requests, and the system is in crisis. On the other hand, if service replication is started improperly that may cause of the problem as follows.

- In an SOA system consists of a lot of services, while the real machine has limited a storage resources so it cannot replicate all services on the single machine without proper consideration.
- Each server machine has limited processing capacities. So a machine that contains too many services and service selection algorithm doesn't work properly, may bring the bottleneck problem.
- The replication mechanism uses a lot of network resources, and it may effects to general services.

Our proposed solution focuses on above problems. We offer QoS functions manage QoS threshold of each service, balance service's load on different servers and evaluate the system's situation before starting service replication. And we offer to follow functions on this our proposed.

- Service registration with QoS definition. The administrator needs to estimate service processing time and assign a reasonable QoS threshold.
- Service updated function. This function is invoked when service processing time excess the registered processing threshold to inform critical situation to the central server.
- Service selection queue length and critical status of each server to choose the best service for invocation.

IV. SYSTEM ARCHITECTURE

This section describes the proposed platform which optimizes service selection and replication algorithm to improve service performance and resource usage in service distributed environment. The system needs a central server that providing databases to store file access detail of each server machine, service information, service queue and service usage history that is applied in service selection and replication process.

A. SOA community with central server

An SOA-distributed model [15] suggests that the SOA system that have multiple communities need a central server to manage and share registration information with all system's communities. To prevent bottlenecks or centralized to fail problems, the proposed platform leverages QoS and load balancing algorithm to service selection and offer proper service instance for consumer so it will separate service invocations to different providers equally. And this platform leverages service replication algorithms to replicate an appropriate service on a suitable provider machine.

While the service replication process uses service usage history to identify the most famous service on the busy provider. Then it will replicate that service on an idler provider. Finally, after the central server finished replication process, the selected server will register automatic the new

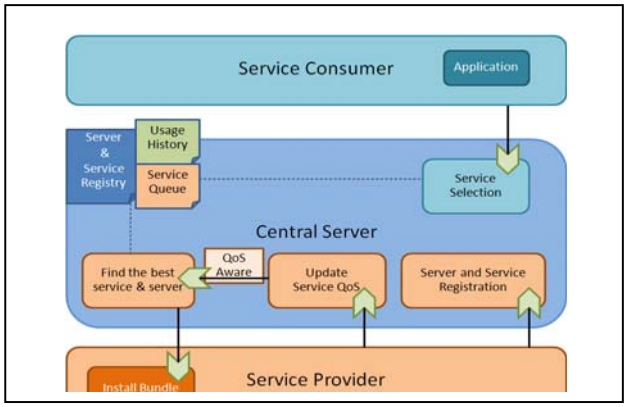

Figure 1. An experimental environment

service to the central server. And this service registration is shared by other communities in the whole system.

B. *Platform component operation*

We propose a service distributed platform that consists of general functions in an SOA environment (e.g. Service registration, Service discovery) and extended function (e.g. Server registration, Install the bundle, Find the best server for replication). In this work, it doesn't only need a database for service registration, but it also desires additional databases for server registration, service usage history and service queue. In Fig. 1 depicts platform component interoperation that includes three roles (service provider, central server and consumer). This platform needs administrator to evaluate and approximate QoS threshold value in the service registry. Then a QoS-aware procedure in this platform applies the threshold value to detect the request overload problem.

Aspect of service invocation, our platform offers service discovery function to retrieve registry information of available service. Furthermore, it will balance consumer's requests to the different providers equally. After the function found the list of available providers, it will evaluate service loads on each provider by using a service queue database. It will choose a service replica from the provider who has the shortest service queue. Then it adds preferred service to the service queue of the selected provider.

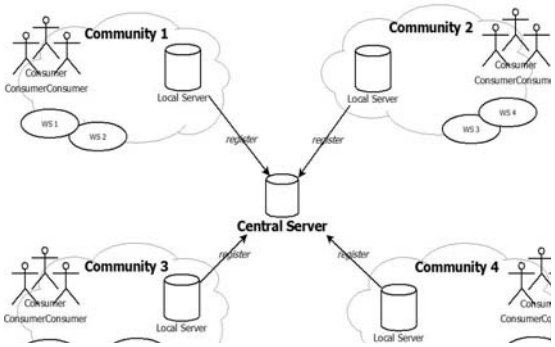

Figure 2. Obtained result from a simple replication system

This platform provides service situation updating function. Service consumer uses this function for updating critical situation to a central server if the invoked service uses greater time than the defined threshold. Then the local server uses service usage history to find the busiest service on that busy provider and replicate that service on the idlest server inside their community by. In case that the server cannot find another machine on the local community for replication, it will forward this replication process to the central server.

V. EXPERIMENTAL EVALUATION

In order to present the different performance between the old simple service distribution platform and our proposed platform, we establish an SOA environment that contains the services and consumers. This scenario generates several of consumer requests on each community that are enough to exhaust the service providers. And we expect that our proposed will be able to prevent bottlenecks and centralize to fail problem.

The scenario consists of four communities and one central server. Our proposed platform is installed on a central server and server machines in every community. The beginning of this scenario, ten services (WS1 – WS10) are distributed to different communities as Fig. 2. Then we generate 1000 consumer requests respectively with a 200 ms interval time in each community, and the requests are dispatched to each community with distinct probability (7% of community 1, 13% of community 2, 27% of community 3 and 53% on community 4).

This first scenario is working on the old distributed platform. The server in each community will communicate with the central server then it downloads and installs the desired service bundle. Finally, each community server handles the service requests by using services in the local community. Fig. 3 shows the response time results. At the beginning of the experiment, response time is high. Furthermore, sometimes it is zero that means the server is too busy and time out problem is occurring.

In another scenario, our proposed platform is installed in this environment. The smart agent can evaluate service situation. And it will start service replication when the system is in critical only. In addition, the smart agent can dispatch service requests across the community. Therefore, service requests are separate from different providers. In Fig. 4 shows that our proposed has good efficiency performance.

TABLE I. COMPARISON OF DISTRIBUTED SYSTEM SCENARIOS

Experimental Evaluation	***Simple Replication***	***Replication with QoS-Aware***
Average processing time (ms)	6750	2103
Availability (%successful)	94.8	99.8
Resource usage (replicas)	40	18

Table 1 shows the advantage from our proposed that has been significantly better than the legacy system. Our platform uses smaller processing time, and consumes fewer

system resources than the old one that doesn't have proper selection and replication algorithm.

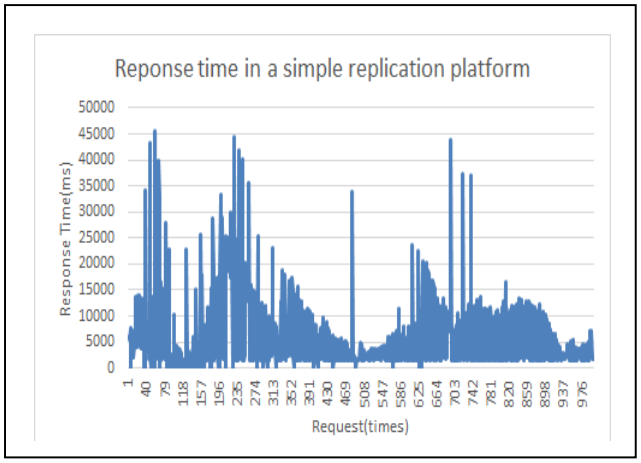

Figure 3. Obtained result from a simple replication system

VI. CONCLUSION

This paper discussed problems in dynamic replication for service bundle that does not have any algorithm to manipulate this replication process. We overview different approached in this area and propose a replication strategy to improve system performance and optimize resource usage. Then we described our proposed that leverage QoS-aware and the load balancing algorithms to monitor service situation and control the replication process. Finally, we create an experimental and show the difference between the simple replication and the replication with QoS-aware As the result, we can show that our approach optimizes system usage and also improves system performance.

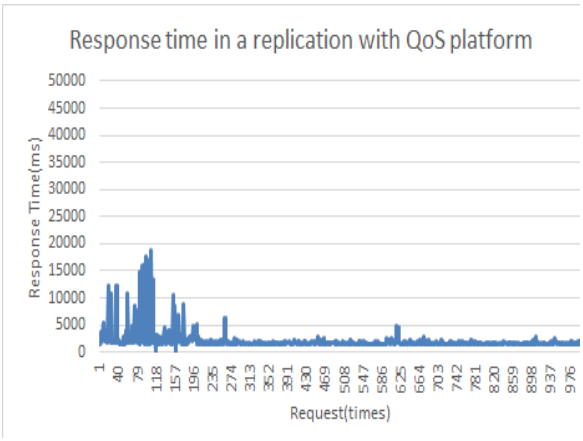

Figure 4. Obtained result from replication with QoS

REFERENCES

[1] A. Papageorgiou, T. Krop, S. Ahlfeld, S. Schulte, J. Eckert, R. Steinmetz, "Enhancing Availability with Self-Organization Extensions in a SOA Platform,". IEEE International Conference on Internet and Web Applications and Services (ICIW 2010), May 2010, pp. 161-166

[2] X. Gu, K. Nahrstedt, "A scalable QoS-aware service aggregation model for peer-to-peer computing grids,". IEEE International Symposium on High Performance Distributed Computing (HPDC-11 2002), 2002, pp 73-82.

[3] Sutheera Puntheeranurak, Aekkawit Chanpen, "A Distributed Bundle SOA for Fault Tolerant Improvement," The International Conference on Engineering, Applied Sciences, and Technology (ICEAST2012), November 2012, pp. 328-333.

F. Yang, U. Da-Yeh, T. Chunghua, "Design and Implement of the Home Networking Service Agent Federation Using Open Service Gateway,". IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems, Sep-Oct 2003, pp 628-633.

[4] C. Fiehe, A. Litvina, I. Luck, I. L. O. Dohndorf, J. Kattwinkel, F. J. Stewing, J. Kruger, H. Krumm, "Location-Transparent Integration of Distributed OSGi Frameworks and Web Services,". IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA '09), May 2009, pp 464-469.

[5] Y. G. Ha, U. Konkuk Univ, "Dynamic Integration of Zigbee Home Networks into Home Gateways Using OSGi Service Registry," IEEE Transactions on Consumer Electronics, May 2009, pp 470-476.

[6] B. Wu, S. J. Liu, W. Cui, "Double Redundant Fault-Tolerance Service Routing Model in ESB,". IEEE ChinaGrid Annual Conference (ChinaGrid '09.), Aug 2009, pp 22-27.

[7] V. I. S. Wietrzyk, R. Lawson, R. Schmid, V. Khandelval, M. Takizawa, T. Enokido, "Distributed Replication and Caching: A Mechanism for Architecting Responsive Web Services," IEEE International Conference on Advanced Information Networking and Applications (AINA 2004), March 2004, pp 289-292

[8] X. Dong, J. Li, Z. F. Wu, D. Zhang, J. Xu, "On Dynamic Replication Strategies in Data Service Grids,". IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), May 2008, pp 155-161.

[9] L. Huo, Y. Fang, H. P. Hu, "Dynamic Service Replica on Distributed Data Mining Grid,". International Conference on Computer Science and Software Engineering, Dec 2008, pp 390-393.

[10] Z. Zheng, M. R. T. Lyu, "A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services,". IEEE International Conference onWeb Services (ICWS '08), Sept 2008, pp 145-152.

[11] Z. Zheng, Zibin M. R. T. Lyu, "A QoS-Aware Middleware for Fault Tolerant Web Services,". International Symposium on Software Reliability Engineering (ISSRE 2008), Nov 2008, pp 97-106

[12] Z. Maamar, U. Zayed, D. Abu P. Thiran, N. C. Narendra, S. Subramanian, "A Framework for Modeling B2B Applications,". International Conference on Advanced Information Networking and Applications (AINA 2008), March 2008, pp 12-19.

[13] M. Zhang, H. Xie, A. F. Boukerche, "A QoS-Aware Load Balancing Algorithm for P2P Based Large-Scale Distributed Virtual Environment,". IEEE International Workshop on Haptic Audio visual Environments and Games (HAVE 2009), Nov 2009, pp 192-196.

[14] S. Kanaparti, "A Distributed-SOA model for Unified Communication Services,". IEEE International Conference on Web Services (ICWS '08), Sept 2008, pp 529-536.

[15] B. Benatallah, Q. Z. Sheng, M. Dumas, "The Self-Serv Environment for Web Services Composition," IEEE Internet Computing, Jan-Feb 2003, pp 40-48.