Optimizing Paxos with Request Exchangeability for Highly Available Web Services

Fenglin Zhang
School of Computer Science and Engineering
Beihang University
Beijing,China
15810542756
zhangfl@act.buaa.edu.cn

Xu Wang
School of Computer Science and Engineering
Beihang University
Beijing,China
13426013850
wangxu@buaa.edu.cn

ABSTRACT
There are a variety of factors such as massive commodity computers, storage, network devices and complex management tasks running behind web services which make web services not available. Replication is an effective solution in that case. There are a variety of replication protocols, and which has been used most is called Paxos. For high available web services, we introduce Paxos as replication protocol and have done a series of optimization work. Firstly, we find that there exists exchangeability between requests to web services and the exchangeability can be used to improve the performance of Paxos. Secondly, in the light of the characteristics of web services with business processes, we find the exchangeable relationship between operations. The exchangeability between operations and the exchangeability between requests can be one-to-one correspondence. Thirdly, we apply the exchangeability to the optimization of Paxos for high available web services. Finally, we have implemented some experiments and experimental results show its effectiveness and performance advantages compared with other replication methods.

Categories and Subject Descriptors
H.4.3 [Information Systems Applications]: Communications Applications.

General Terms
Application, Experimentation.

Keywords
web service, business process, exchangeable relationship, Paxos.

1. INTRODUCTION
In recent years, web services are widely used in Internet and cloud computing [1] applications. With the rapid development of cloud computing and big-data [2], the data-centric services become more popular in many big data clouds. The key of the scalability and availability of services is replication.

In this paper, we introduce the Paxos [5] algorithm as the base of our replication protocol. In this situation, a replica will be elected as the leader to issue a request, and all the other replicas will be followers. Under the constraints of replication protocol, all the replicas will be submitted in the same order and this ensures consistency. This process includes two phases [5, 6], the request sorting phase and the request executing phase. This paper focuses on the request sorting phase and some improvement will be made to enhance the performance, including the introduction of pipeline mechanism and exchangeability between requests. Some later request can be submitted earlier without influencing the consistency of the web service. For example, there often exist concurrent requests A and B (i.e., the corresponding operation of A and B can be concurrent) in a web service and the exchange for the order of submitting A and B will not influence the consistency.

Rep4WS [12] has directly used RDG (Request Dependence Graph), the extended results of exchangeability between requests, to improve the performance in execution phase. Here, we address the characteristics of web service, utilize the structural features of operation protocol for web service, make a definition of cell structures, and finally obtain exchangeability between requests via the thought of super-graph.

The main contributions of the paper include: (1) According to the characteristics of web services, through the analysis of business processes' structure, we find exchangeable relationship between requests; (2) We describe how to use pipeline concurrency and exchangeability between requests to improve the Paxos's performance; (3) We have implemented some experiments and experimental results show its effectiveness and performance advantages compared with other replication methods.

The remainder of this paper is structured as follows. Section 2 introduces related work. Section 3 describes the key algorithm that how to obtain the exchangeability between requests. The optimization of replication protocol is presented in Section 4. Section 5 shows our experimental results. Finally, Section 6 concludes.

2. WEB SERVICE REQUEST EXCHANGEABILITY

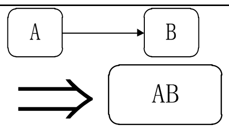
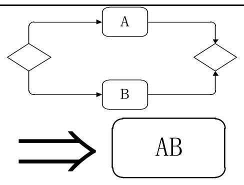
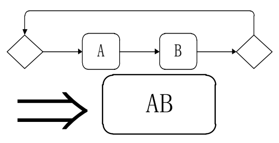
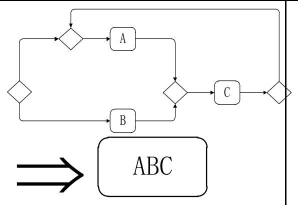
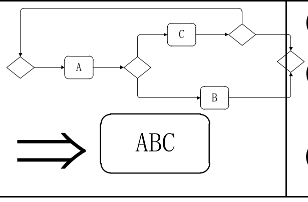
2.1 Exchangeability and Consistency
As a general rule, in order to ensure consistency of replicas, the request $n + i$ cannot be executed before the executions of the requests from 1 to $n + i - 1$ has be executed. But for web services, some requests don't need to meet this condition, and the order of requests is not the only. For instance, Twitter's web services have many information checking and searching operations. The order

of requests to these operations is exchangeable without influencing the consistency. We can implement optimization on Paxos if we know the exchangeability of requests.

2.2 Business Process and Exchangeability

As to web services with business process, the business process provides us with good conditions to analysis the exchangeability from its structure.

We introduce Business Process Model and Notation, BPMN to describe the business process of web services. BPMN process is composed of BPMN elements, which has the ability of process description and supports multiparty collaboration. We focus on the executable elements of BPMN, such as Flow Objects and Connecting Objects. An activity usually indicates a specific work link, which corresponds to an operation of a web service. Gateway describes the branch and summary of the process, including fork, join, decision and merge. Connecting Objects include sequence flow and message flow. We don't consider the message flow temporarily.

No.	Rule	Exchangeable Relationship
1		$(A, B) \notin OET$
2		$(A, B) \in OET$
3		$(A, B) \notin OET$
4		$(A, B) \in OET$, $(B, C) \in OET$ $(A, C) \in OET$
5		$(A, B) \in OET$, $(B, C) \in OET$ $(A, C) \in OET$

Definition 1.Cell-Structure

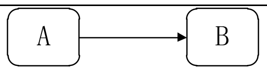
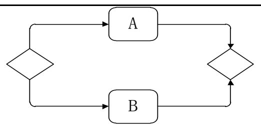
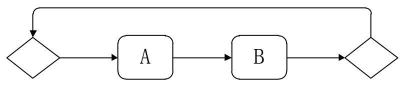
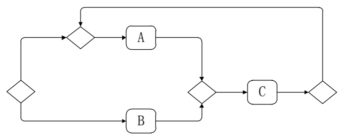
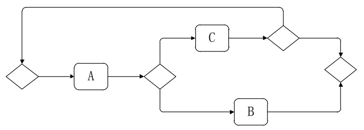
There are five Cell-Structures in Table 1, in which the A, B, C are Primitives defined in the following.

Definition 2.Primitive

(1) An activity is a Primitive;

(2) A Primitive can be transferred from a Cell-Structure, and involves all Primitives included in this Cell-Structure.

Through the above two definitions, a business process in fact can be seen as combination with Primitives and Cell-Structure in limited times

Cell-Structure	
Sequence	
Concurrent and Branch	
General-loop	
Branch-loop	
Loop-branch	

2.3 Operation Exchangeable Table

As our work is partly based on the exchangeable relationship between requests, and Rep4WS [14] has done some work about this, we introduce the concept of Operation Exchangeable Table.

At first, a web service can be seen as a state machine, which generates a new state and an output after executing a request. This can be described by a formula:

$$WS(s_n, r_n) = (s_{n+1}, o_n)$$

This formula means that when a web service is in the state s_n and after it has executed a request r_n, the web service will move to a new state s_{n+1} and generate an output o_n. On the basis of the above, we use an Operation Exchangeable Table of a web service to define the exchangeable relationship between two web services.

Definition 3 [Operation Exchangeable Table] $T = \{(x, y) | x \in P, y \in P, x \text{ and } y \text{ is exchangeable}\}$, where P is the collection of operations in a web service. For any state s and two requests τ_x and τ_y which invoke operation x and y respectively, if $WS(s, \tau_x) = (s_x, o_x), WS(s_{x,}\tau_y) = (s_y, o_y), WS(s,, \tau_y) = (s'_y, o'_y), WS(s'_x, \tau_x) = (s'_x, o'_x)$, operation x and y are exchangeable if and only if $s'_x = s_y,\ o'_x = o_x,\ and\ o'_y = o_y$.

In order to get the complete OET, we need to define a set of transformation rules based on Cell-Structure and Primitive, used for digging exchangeable relationship between requests.

According to the following steps, we generate the OET

1. Initialization transformation recording stack that is empty at the

very start, which is used to record the used transformation rules and related Primitives this time;

2. By single or multiple using rule one, we combine the activities or Primitives that are connected with sequence flow into one or multiple Primitives including all the related activities and Primitives. Then we push these transformations into stack. In the end of this step, we get a business process without directly connected by sequence flow.

3. Along the stream direction, which means from start event to end event, and the order of rule five, rule four, rule three, rule two, we get every Cell-structure transformed into a Primitive. Then we push these transformations into stack.

4. At last we get one Primitive which includes all the activities and a stack which record all the transformations.

5. Pull the stack, and restore the business process, and get the OET. According to the following rules, we can reverse exchangeable relationship.

(1)If Primitive T and Primitive F are exchangeable, all the activities included in Primitive T and all the activities included in Primitive F are pairwise exchangeable, which means if $(T, F) \in OET$, then $\forall A_i \in T, \forall B_j \in F, (A_i, B_j) \in OET$

(2)If Primitive T and Primitive F are not exchangeable, all the activities included in Primitive T and all the activities included in Primitive F are pairwise not exchangeable, which means if $(T, F) \notin OET$, then $\forall A_i \in T, \forall B_j \in F, (A_i, B_j) \notin, OET$

6. Through the two rules, once we pull a record from the stack, we can get the exchangeable relationships between all the related activities. Until the stack become empty, we get all exchangeable relationships between all activities.

At this point, we generate a complete OET, and this can be used to optimize the Paxos in the commit phase.

The following Figure 1 is an example that simulate the transformation process

1. According to Rule one, we combine B and D into a Primitive BD, and push this into the transformation record stack;

2. According to Rule five, we combine A, BD, C into a Primitive ABCD, and update the stack;

3. According to Rule one, we combine ABCD and E into a Primitive ABCDE, and update the stack;

4. In the end, we get a Primitive including A, B, C, D, E.

5. As mentioned above, we can reverse exchangeable relationship as following:

$(E, ABCD) \notin OET$, that is $(E, A) \notin OET, (E, B) \notin OET, (E, C) \notin OET, (E, D) \notin OET$

$(A, BD) \in OET$, $(C, BD) \in OET, (A, C) \in OET$, that is $(A, B) \in OET$, $(A, D) \in OET$, $(A, C) \in OET$, $(C, B) \in OET$, $(C, D) \in OET$.

$(B, D) \notin OET$

Through these steps, we generate the whole OET of ABCDE.

The exchangeability between operations and the exchangeability between requests can be one-to-one correspondence. So we can obtain the exchangeability of requests from OET to improve the performance of Paxos.

3. OPTIMIZATION OF PAXOS: OET-Paxos

This section will come up from two ideas for optimization.

3.1 Optimization with Pipeline Concurrency

The first phase of Paxos can be concurrent with pipeline technique. Pipeline allows us commit requests from *n* to *n*+*α* in parallel as long as requests before *n* have been executed. Pipeline concurrency can improve the performance of Paxos especially in high-latency networks, as the leader can commit more requests during the time of waiting to receive messages.

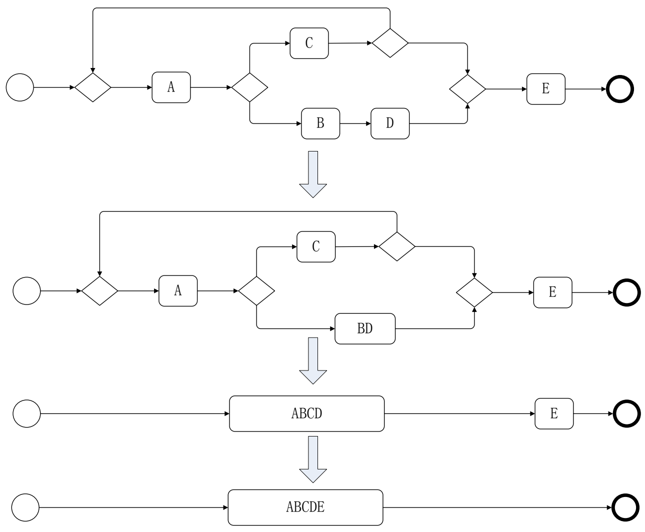


Figure 1.Transformation process

3.2 Optimization based on OET

At the committing phase of request, Paxos is to sort all the received requests, which ensures that all replicas have the same order to commit and execute these requests and ensure the consistency of replicas.

However, as mentioned above, there are exchangeable relationships between requests, and we can get the exchangeable relationship through technological means. In this case, we don't need to pay time to sort these requests that have exchangeable relationship. So we can reduce the response time and improve performance by leaving out this part of time to sort these exchangeable requests.

For example, the request sequences are {A, B, C, D, E}. B and C are exchangeable; we can ignore the order between B and C. The more exchangeable relationships we find, the more time we can save.

4. EXPERIMENTAL RESULTS

This section describes the experimental results. The experiments are performed to compare our replication protocol which we name OET-Paxos with conventional replication approaches such as basic Paxos, Two-Phase Commit (2PC), and Group Communication (GC).

We design some experiments to prove the performance of OET-Paxos is better than other conventional replication protocols, such as 2PC, GC, and the basic Paxos which doesn’t have pipeline and OET optimization. For 2PC, we use the standard 2PC protocol. For GC, we use JGroups 3.0 that is an open source system published on Git. In our experiments, we use JPaxos1.0 as experiment and optimization object and JGroups3.0 which base on TCP and total order protocol stack.

Figure 2 is the business process we use in our experiment, which is shown in the following.

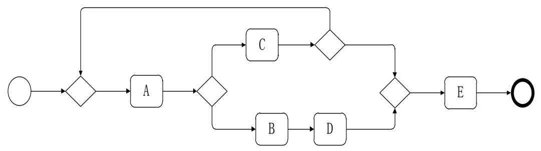


Figure 2.business process in experiment

4.1 Performance Comparison

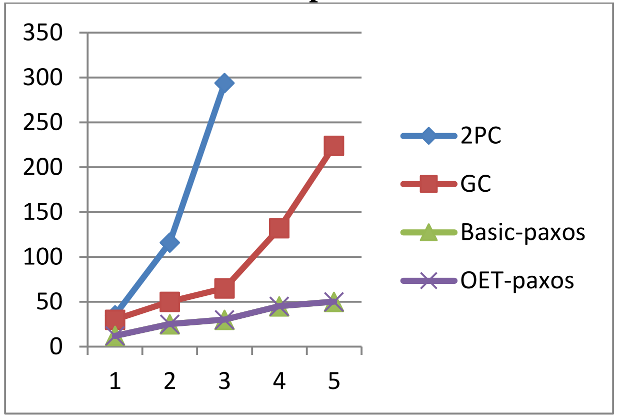


Figure 3.Consecutive Requests

Figure 3 shows that response time of algorithms increase as the number of replicas increased. This is because that with the increase in the number of replicas, the overhead of maintaining consistency also increases. As shown, the average response time of OET-Paxos is respectively 45.3 percent of 2PC and 28.1 percent of JGroups. In such a serial requesting case, the performance between OET-Paxos and Basic-Paxos are the same. In addition, the response time rises as the replica number is increased. This is because new replicas will incur extra overheads in message transport, thread synchronization and so on.

Figure 4 shows that response time changes as the load increases when there are a large number of concurrent requests. As shown, OET-Paxos curve changes less dramatic than the other three. OET-Paxos performs well when there are a large number of concurrent requests.

5. CONCLUSION

In this paper, we complete the exchangeable relationship of requests analyzing algorithm designing work based on business process. In our replication protocol, OET-Paxos, pipeline concurrency and OET are used to improve its performance. Our experiments show that OET-Paxos is effective and outperforms conventional ones.

This paper mainly embarks from the structure of business process due to mining exchangeable relationship. In the next work, we are going to join the influence of input and output parameters, and further explore exchangeable relationship between requests, and handle the conflicts between them by comparing them with the results of structure. In addition, OET-Paxos can be extended for more widely web-based services, such as RESTful services, data services and so on.

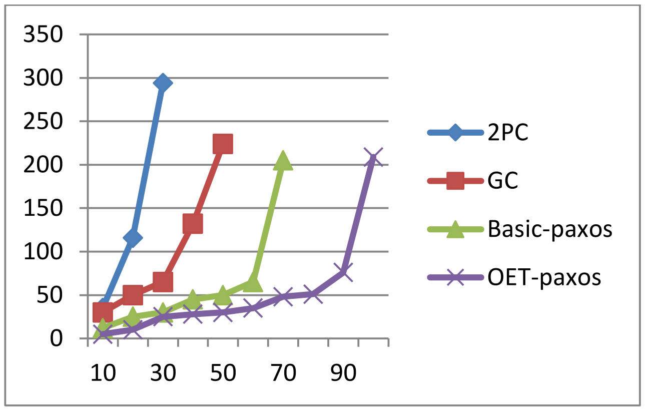


Figure 4.Concurrent Requests

6. ACKNOWLEDGMENTS

This work was supported by China 863 program (No. 2012AA011203).

7. REFERENCES

[1] WIKIPEDIA: Cloud http://en.wikipedia.org/wiki/Cloud_computing

[2] Big data: The next frontier for innovation, competition, and productivity. http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_frontier_for_innovation

[3] A Greenberg, etc. The cost of a cloud: research problems in data center networks. ACM SIGCOMM 2008.

[4] State of the Data Center 2011 http://www.emersonnetworkpower.com/en-US/About/NewsRoom/Pages/2011DataCenterState.aspx

[5] L. Lamport. Paxos Made Simple. ACM SIGACT News, 32(4):18-25,December 2001

[6] L. Lamport. The Part-Time Parliament. In ACM Trans. On Computer Systems, pages 133-169, 16(2), 1998.

[7] F. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. ACM Computing Surveys, 22(4), 1990

[8] Kończak J, Santos N, Żurkowski T, et al. JPaxos: State machine replication based on the Paxos protocol. Faculté Informatique et Communications, EPFL, Tech. Rep, 2011, 167765.

[9] X. Zhang, F. Junqueira, etc. Replicating Nondeterministic Services on Grid Environments. In HPDC 2006

[10] Jacob R. Lorch,etc. The SMART Way to Migrate Replicated Stateful Services. ACM SIGOPS 2006.

[11] William J. Bolosky,etc. Paxos Replicated State Machines as the Basis of a High-Performance Data Store. In NSDI 2011

[12] Xu Wang, Hailong Sun, etc Rep4WS: A Paxos based Replication Framework for Building Consistent and Reliable Web Services. IEEE ICWS, 19, pp. 355–369, 2012