A Generic Model of Consistency Guarantees for Replicated Services

Szymon Francuzik
Institute of Computing Science
Poznań University of Technology
Poznań, Poland
szymon.francuzik@put.edu.pl

Cezary Sobaniec
Institute of Computing Science
Poznań University of Technology
Poznań, Poland
szymon.francuzik@put.edu.pl

Dariusz Wawrzyniak
Institute of Computing Science
Poznań University of Technology
Poznań, Poland
szymon.francuzik@put.edu.pl

Abstract

High availability, scalability, and reliability of services can be provided by replication. However, distributed systems suffer from network partitioning, which reduces availability and/or consistency. The choice between availability and consistency boils down to distinguishing between pessimistic and optimistic approaches to replication. This paper proposes a new model of coexistence of pessimistic and optimistic replication, enabling the user to balance between availability and consistency. In this approach a client can decide on the degree of optimism appropriate for the application. This is achieved by specifying operation modes which are intended to describe client expectations.

Categories and Subject Descriptors C.2.4 [*Computer Systems Organization*]: Computer-Communication Networks—Distributed applications

Keywords optimistic replication, weak consistency, strong consistency

1. Introduction

High availability, scalability, and reliability of services can be provided by replication [6]. However, distributed systems suffer from network partitioning, which cease communication between nodes, thereby reducing availability. It has been proved that it is impossible to ensure both consistency and availability in systems where network partitioning is possible [4]. The choice between consistency and availability boils down to distinguishing between pessimistic and optimistic approaches to replication [5]. Pessimistic replication ensures strict consistency but precludes from performing some operations when the communication between replicas is impossible. In contrast, optimistic replication allows execution of operations on a local replica without communication with other replicas, but may reveal inconsistency of replicas states. Thereby we have to rely on weak guarantees like eventual consistency. This paper proposes a new model of coexistence of pessimistic and optimistic replication, enabling the user to balance between availability and consistency. It is an abstract of a revised version of our previous paper [1]. In this approach a client can decide on the degree of optimism appropriate for the application. This is achieved by specifying operation modes which are intended to describe client expectations. Optimistic operations can be executed during network partitioning, whereas pessimistic operations require connectivity but offer strict consistency guarantees.

PaPEC'14, April 13–16 2014, Amsterdam, Netherlands.

ACM 978-1-4503-2716-9/14/04. . . $15.00.
http://dx.doi.org/10.1145/2596631.2596643

2. System Model

We assume a client-server processing model. Clients connect to selected servers, and issue requests. Requests are of two distinguished types: read-only requests (or just *reads*), and modifying requests (also called *writes*). The receiving server propagates modifying requests to other replica servers, i.e. it uses operation transfer to synchronize replica states, which is known as semi-active replication. The services are deterministic: a sequence of writes performed on a replica should produce a consistent state provided that the initial state was consistent. We do not assume the internal structure of the service and its interface. The replication mechanism may be built into the service or can have a form of proxy servers intercepting communication with the service.

3. Operation Processing

Operations are accepted at servers regardless of the state of communication links. The lack of connectivity means that the replica cannot decide on the final ordering of operations, because at the same time other requests (possibly conflicting) may be processed at other servers. As a result the re-

Tab. 1. Classification of submission modes of operations

READ	**pessimistic**	**optimistic**
synchronized	writes received before the read must be committed	writes received before the read must be applied
immediate	uncommitted operations must be retracted	the current state
WRITE	**pessimistic**	**optimistic**
synchronous	the write must be committed	writes received before the write must be applied
asynchronous	the write is submitted and acknowledged	

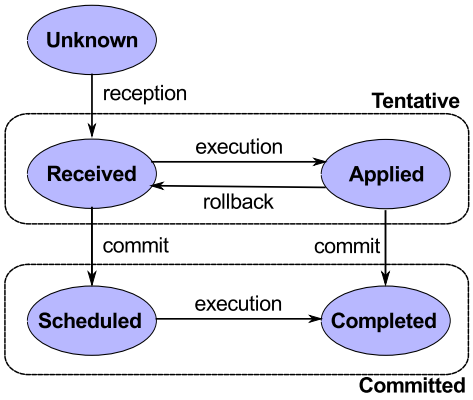

Fig. 1. States of processing of writes

quest is accepted *tentatively*. Optimistic approaches to replication assume that conflicts are rare, and that the local order of operations finally will be accepted. Otherwise, a rollback is necessary and conflicting operations must be (re)executed in a proper order. Moreover, the same operation may be applied and rolled back several times before establishing the final order because of the concurrent operations at other servers. After establishing the final global order of writes the operation becomes *committed*. Not applied operations that are committed become *scheduled*, and after execution they become *completed*. Optimistically applied operations in case of no conflicts may change its state directly to *completed* after committing. The state diagram of processing phases of modifying operations is presented in Fig. 1. For reads, the diagram is very similar, without rollback transitions, and with *applied* state treated as an alternative terminating state.

The decision concerning consistency vs. availability trade off can be made by the client by choosing one of the predefined operation submission modes. A mode can be chosen individually for every operation. We have proposed two orthogonal criteria for classification of operation modes. The first distinction is between *optimistic* and *pessimistic* operation modes. Pessimistic operations are supposed to give globally consistent view of the data and reflect committed states of replicated data, but usually require coordination with other replicas. Optimistic operations can complete even when other replicas are unavailable but the results are based on tentative states.

The second distinction depends on the type of operation. In case of reads we distinguish: *synchronized* and *immediate* modes. The synchronized mode is supposed to provide as up to date state as possible, which means applying all writes known to the replica at the time the request has been issued. The immediate mode tries to produce the answer as quickly as possible relying on the current state of the server. In the case of writes we distinguish *synchronous* and *asynchronous* modes. Synchronous writes—similarly to synchronized reads—require that all previous writes are applied. Asynchronous writes do not expect any outcome, thus can be performed any time after submitting the request.

Table. 1 presents all possible submission modes for operations. In the case of reads: the pessimistic synchronous mode requires that all preceding writes are in completed state, i.e. they are scheduled and applied. Moreover, all tentative writes following the read that could influence the read have to be rolled back. The pessimistic synchronous mode guarantees that the replication is transparent to the client due to strict consistency, but the response may require a great deal of time necessary for the communication between replicas. The synchronized optimistic mode is less restrictive: all preceding writes must be applied, but not necessarily scheduled, which is possible to achieve without communication with other servers. Immediate pessimistic reads do not have to commit all previous writes: they return a stable state resulting from all previously committed writes with all tentatively applied writes rolled back. Finally, immediate optimistic reads return just the current state of the server, without applying or rolling back any writes.

In the case of writes: once again the strongest consistency guarantees come from synchronous pessimistic writes. The current write must be committed, which means that it has to be scheduled and applied along with all preceding writes, and the following tentatively applied writes have to be rolled back. Synchronous optimistic writes require execution of all previous writes with no prior scheduling. Asynchronous

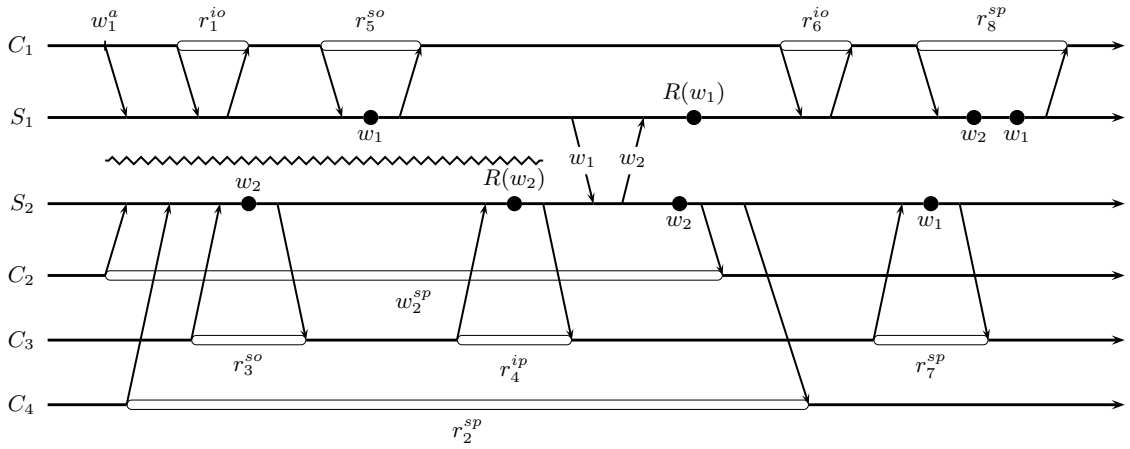

Fig. 2. Example execution of operations using different modes

pessimistic and optimistic writes are processed in the same manner from the client point of view: they are just submitted and get scheduled/applied later. The exact moment depends on the following operations and their submission modes.

It is worth noting that the execution of optimistic operations is not blocked by pessimistic ones. Preceding pessimistic operations may be executed tentatively without returning results to the issuing process. In fact the modes of operations focus on the view of states observed by clients rather than the states of replicas themselves.

Execution of operations can be optimized by applying multi-versioning to data objects for the purpose of producing tentative states of objects. The stable version could represent states resulting from committed operations, and the other version (or versions) could represent tentative states. After scheduling the tentative version can be promoted to a stable state, or be discarded in case of wrong ordering.

4. Example

Fig. 2 presents an example execution in a system supporting combined pessimistic and optimistic replication. The notation is following: C_i denotes a client, S_j a replica server; r denotes a read operation, w—a write; the submission mode of operations is given in superscript (e.g. w^{so} denotes a synchronous optimistic write), subscript of operations just identifies it; $R(w)$ denotes a rollback of write w. The processing starts with an asynchronous write w_1^a submitted at server S_1 by client C_1. The write is being queued and not applied, therefore the following immediate optimistic read r_1^{io} does not reflect any change. The next read r_5^{so} of C_1 due to its synchronicity forces execution of w_1. At the same time another replica S_2 receives a synchronous pessimistic write w_2^{sp} from client C_2. The write cannot be completed due to communication problems between S_2 and S_1. Similarly, a synchronous pessimistic read r_2^{sp} of client C_4 has to wait for the completion of buffered write w_2. However, a synchronous optimistic read r_3^{os} of client C_3 can proceed, and forces tentative execution of w_2 at S_2. The next immediate pessimistic read r_4^{ip} of the client forces rollback of w_2. Finally, the connectivity is restored and servers inform each other about new writes. Additionally, they decide that the global ordering of operations will be w_2 followed by w_1. As a consequence server S_1 has to rollback w_1, which is observed by an immediate optimistic read r_6^{io} of C_1. The following synchronized pessimistic read r_8^{sp} of that client forces execution of (already scheduled) writes w_2 and w_1. At the same time server S_2 has finally performed w_2 and completed pending operations w_2^{sp} and r_2^{sp}. The last synchronous pessimistic read r_7^{sp} of client C_3 forces execution of write w_1 at server S_2.

5. Related Work

The idea of issuing operations in different modes appeared in hybrid consistency for distributed shared memory [3]. Essential to that approach is the distinction between strong and weak operations that apparently resemble pessimistic and optimistic ones. However, the dissimilarity between hybrid consistency and our model lies in the effect of operations on the system state. In our model, the effect is local — it concerns de facto the view of the system state for the issuing process. The specification of operations in hybrid consistency determines the order of their execution on every replica, thereby influences the view for other processes.

6. Implementation

The proposed replication model has been realized by developing a series of consistency protocols. We have also built an infrastructure for replication of RESTful web services [2]. The infrastructure comprises of reverse proxy servers intercepting invocations of services. The proxies communicate with each other and disseminate modifying requests in order to keep replicas consistent. The user can specify submission modes of requests by adding extra headers to HTTP requests (by default the system process requests in sync. pessimistic

mode). We are currently conducting a series of experiments demonstrating overall performance and availability of the system.

7. Acknowledgments

The project was funded from National Science Centre funds granted by decision No. DEC-2012/07/B/ST6/01230.

References

[1] M. Bazydło, S. Francuzik, C. Sobaniec, and D. Wawrzyniak. Combining optimistic and pessimistic replication. In *Proc. of the 9th Int. Conf. on Parallel Processing and Applied Mathematics (PPAM 2011)*, volume 7203 of *Lecture Notes in Computer Science*, pages 20–29, Toruń, Poland, 2012. Springer-Verlag.

[2] M. Bazydło, S. Francuzik, C. Sobaniec, and D. Wawrzyniak. Replication infrastructure for RESTful web services. In *Proc. of Int. Conf. on Advanced Computing, Networking and Security*, volume 7135 of *Lecture Notes in Computer Science*, pages 28–37, Mangalore, India, Apr. 2012. Springer-Verlag.

[3] R. Friedman. *Consistency Conditions for Distributed Shared Memories*. PhD thesis, Computer Science Department, Technion–Israel Institute of Technology, June 1994.

[4] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[5] Y. Saito and M. Shapiro. Optimistic replication. *ACM Computing Surveys*, 37(1):42–81, Mar. 2005.

[6] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. Understanding replication in databases and distributed systems. In *Proc. of the 20th Int. Conf. on Distributed Computing Systems (ICDCS 2000)*, pages 464–474, Taipei, Taiwan, R.O.C., Apr. 2000.