Replication: Analysis & Tackle in Distributed Real Time Database System

SANJAY KUMAR TIWARI, A.K.SHARMA, V.SWAROOP
Department of Computer Sc. & Engg. M.M.M. Engineering College, Gorakhpur, U.P., India.

ABSTRACT
Replication is a basic concept of distributed database systems, which has been concentrate studied for a very long period. Many research and active usage as made data replication a well-understood and usually applied method for different purposes like increasing scalability, availability, fault tolerance and responsiveness of distributed systems. But when we implement it for distributed real time system the complication has been increased more about using replication of real time application data, a synchronization mechanism has to be incorporated into the distributed real time system in order to assurance a particular consistency model of the data copies. Particular distributed real time applications differ in several characteristics like, time constraints, the size of the distributed system and the database the overall update rate of data or the ratio of read and write accesses, such that a variety of replication concepts has been developed: active or passive replication, with eager or lazy synchronisation. When implementing a particular distributed application or a distribution concept for a particular class of applications, there are several specialized replication concepts available to choose from. And researches must be analyzed for real time application. In addition, consistency of the replicated state is much more important than responsiveness of the distributed application. In this paper we raise all the analysis & tackle in Replicated data management for distributed real time system to raise the issues for work done more for new researchers.

KEYWORDS: *Data Replication, Consistency, Scalability, Eager and Lazy Replication, Correctness, Replicated Distributed Real Time Databases, Synchronisation.*

I. INTRODUCTION

In the available replication approach, all dynamic entities of the virtual environment are replicated at all participating servers, allowing parallelising their processing. The issue regarding the replication for distributed real time system is time constraints for Hard, Soft and Firm type transactions. The available techniques for data replication are analyzed for these transactions. More research works are carried out in this direction. The processing of the virtual environment state of the corresponding region is parallelized among participating servers, which is possible in a novel way because each server has the full application state available due to its replication. The replicated states are synchronized between the servers in order to have a consistent copy of the application state at all servers. This parallel processing and synchronisation is integrated into the tick-based processing scheme of real time interactive applications: Each server in the replication approach runs a conventional real-time update loop, enriched with additional functionality for inter-server synchronisation. Instead of parallelizing "horizontally" over the virtual environment as in the instancing and zoning approach, this replication concept parallelises "vertically" over the entities in the replicated environment. Thus, one replication at a single server can be imagined as a parallel layer of the virtual world. A client participating in a session using this replication is connected to one of the servers. The complete environment state is fully replicated at each server, such that any of them can in principle support any client regardless of the client's avatar position in the virtual environment.

Most performance evaluations assume full replication (e.g., [1], [2], [3], [4], [5], [6]), i.e., all data objects are replicated to all sites so that each site holds a complete copy of the distributed database. This is an extreme case of replication and it has been recognized that, for many applications, neither full nor no replication is the optimal configuration [7], [8], [9]. Some authors argue that full replication is an acceptable assumption for worst-case considerations [10].

In particular, the four main aspects of distributed real-time processing have to be addressed in detail-

Consistency- The servers have to synchronise their replicated copies of the virtual environment following a suitable scheme, which determines a particular consistency model for the distributed application state. In general, there is a trade-off between the degree of consistency of the replicated state and the level of responsiveness of the interactive application, because stronger consistency model implementations introduce a higher coordination overhead which lessens the responsiveness. As a compromise suitable to satisfy the real-time requirements of the continuous application, this paper presents a fast and highly responsive synchronisation mechanism which still offers a suitable degree of consistency. A systematic analysis of different consistency models and synchronization mechanisms regarding their suitability for the "vertical" replicated parallelization of interactive real-time environments is required.

Responsiveness- Responsiveness denotes the reaction time of the application to inputs by the users to control their avatars and interact with each other. Especially the controlling in action, sports and racing games requires a fast feedback denotes the reaction time of the application to inputs by users to control their avatars and loop of issuing a command and perceiving the results like, for example, position changes of entities. Depending on the actual application, reaction times may have to be below 50 to 100ms for optimal performance. The challenge for the replication concept in terms of responsiveness is two fold:

1. The replication approach has to offer synchronization mechanism between servers which has minimal waiting times and can be pipelined.
2. The operational multi-server network architecture for running replicated applications has to allow responsive setups.

Scalability- The replication parallelization approach has been developed with the goal to scale the density of entities in a virtual environment. In the layered, "vertical" parallelisation concept, this requires to efficiently distribute the real-time processing among the participating servers and to minimize synchronization overhead between the replication layers. For this purpose, different types of virtual world entities are distinguished and a load balanced distribution scheme for the processing of these entities among the servers is developed. The usage of computation and communication resources in the replication approach has been analytically analysed. The resulting Game Scalability Model allows comparing different distribution concepts for interactive applications and provides a cost and load prediction model for optimising the setup of a replicated session.

Correctness- Ensure correctness in the replication approach is a challenging task due to the inherently parallel processing of entities in different replication Parallelisation and Scalability of Entity Processing layers. The work on analyzing the correctness issues in the replication approach and devising two different mechanisms for ensuring correctness. Furthermore, a correctness control server has been designed and implemented for the Proxy-Server network to quantify incorrect input processings in one of the demonstrator games, allow analysing the correctness problems as well as the implemented solution at runtime.

II. PROCESSING MODEL AND SCALABILITY CONSIDERATION

The computations of the replicated layers are not independent from each other, but require synchronising the entities each time a new state is computed, i.e., each tick. As a general non-functional requirement of distributed virtual environments, the application clients have to immediately and frequently display the changes of other entities, like for example another avatar walking around. It would be not acceptable and not considered a real-time application anymore if the other entities were only updated once in a while, resulting in unresponsive movements and interactions. As a result, the processing model and synchronisation mechanism introduced in detail in the following has to introduce as little extra computation and communication time as possible in

order to enable fast responses for user inputs and the general virtual environment processing. The worst case for the computations in a single update loop occurs when every entity has to be processed. In contrast to the fixed available computation time of a single-server setup, the replication approach allows to add several servers to a single session. Let us assume that the n entities are equally distributed on l servers hosted on server machines of the same processing power. As the result, each server has to process only n/ l entities in each update loop. However, each server has to synchronise the entities processed at other servers: let tsync be the required time for updating the local replicated state of a remotely processed entity. Using these coefficients, the processing time for a single overall state update at a server in the parallel replication approach can be estimated as-

$$T_{replicationserver} = n/l.t_{proc}+(n-n/l).t_{sync}$$

constitutes a perfectly parallelizable part of the computation of the new state. The total number of entities n can be increased without increasing the processing time at each server by adding additional replication servers, thus increasing l in the equation. However, besides the perfectly parallelized processing of entities, each server has to synchronise all entities maintained at the other servers requiring a computation time of (n - n / l) tsync. This synchronization overhead grows with the total number of entities n, such that the parallelisation is not embarrassingly, but, depending on the particular synchronization mechanism implemented, may require notable computation time for the synchronisation.

III. REPLICATED STATE SYNCHRONIZATION AND CONSISTENCY

Data replication is a fundamental concept of distributed systems, which has been intensively studied for a long time. Various research and active usage has made data replication a well-understood and commonly applied technique for different purposes like increasing scalability, availability, fault tolerance and responsiveness of distributed systems [11]. When using replication of application data, a synchronisation mechanism has to be incorporated into the distributed system in order to guarantee a particular consistency model of the data copies. Particular distributed applications differ in several characteristics like, for example, the size of the distributed system and the database, the overall update rate of data or the ratio of read and write accesses, such that a variety of replication concepts has been developed: for example, active or passive replication, with eager or lazy synchronisation. Each of them behaves differently regarding characteristics like scalability, fault-tolerance or responsiveness. When implementing a particular distributed application or a distribution concept for a particular class of applications, there are several specialized replication concepts available to choose from. An overview of the different available replication concepts can be found in [12] or [13]. Most of the replication concepts known in the field of distributed computing target discrete, transaction-oriented applications. Such applications, like distributed databases or information management systems, change their state only upon explicit input. Furthermore, consistency of the replicated state is much more important than responsiveness of the distributed application. For example, in the case of bank account operations, the distributed replications of an updated account object have to be synchronised before any other operation may take place, regardless of the time required. Responsiveness is of much more importance in continuous applications [14] which require adjusted or completely novel synchronisation mechanisms for ensuring consistency of the replicated application state [15]. However, there are no off-the-shelf responsive synchronisation mechanisms which can be used in the replication approach. It a challenge to evaluate existing synchronization mechanisms developed for discrete applications regarding their suitability to ensure an adequate level of consistency in the replication approach. Foreclosing the result of the analysis of existing synchronisation mechanisms in the following sections, a lazy, primary copy synchronisation mechanism ensuring eventual consistency of the replicated application state has been chosen to be used in the replication approach. This mechanism fits the particular needs of the fast-paced and highly responsive virtual environment applications like action games which are aimed at with the replication concept. In the following, the different available alternatives for synchronizing replicated objects are discussed and the chosen mechanism is justified and motivated.

IV. ANALYSIS OF EXISTING REPLICATION AND SYNCHRONISATION MECHANISM

In support of developing an appropriate synchronisation mechanism for the replication approach, the following sections systematically analyse existing synchronization mechanisms for different replication concepts for discrete, transaction based applications. Since the main goal of using replication in interactive applications is to increase scalability, the entity synchronisation discussion focuses on minimising the processing time tsync for scalability reasons. The analysis of synchronisation mechanisms presented below follows the general classification scheme found in the research of replication concepts [12, 13]. The two main classification criteria of replication concepts are-

• The Update propagation scheme defining how objects copies are being updated, and
• The Update location scheme determining where, i.e., at which servers, updates for what objects can be initialized.

Eager vs. Lazy Replication

In the following, two main update propagation concepts of eager replication and lazy replication are generally presented and, in particular, discussed regarding their suitability for being used in the novel entity replication approach for interactive virtual environments presented in this paper. Both update propagation concepts are introduced by discussing how the replication servers handle a general request send by an application client to a particular server and how they update the replicated application state accordingly. When adopting such a propagation concept in a virtual environment application, the client requests would be interactive inputs like moving avatars and interacting with other users, which require fast responses for providing a fluent and immediately reacting real-time application.

Eager Replication- The concept of eager replication updates all replicated copies of the particular object being changed as part of the original processing of the received client request. An update operation, therefore, is executed on all distributed copies of a particular object before a notification is returned to the client. This synchronisation scheme requires using nested transactions with a distributed commit protocol like the two-phase commit [16] or a reliable atomic multicast [17] for ensuring atomicity of the distributed update. The servers maintaining the copies have to communicate in several steps before the operation can be performed and the response can be send to the client. Following the general processing scheme of eager replication presented in [12], the different stages of synchronising replicated objects eagerly. As the main property of eager replication, that a client initiating an update does not receive any response until all the processing steps of executing and coordinating the object update are performed by the servers. Therefore, although a high degree of consistency can be ensured due to the atomicity of the distributed replication updates, the eager synchronization scheme introduces a substantial responsiveness overhead due to the communication required between servers.

Lazy Replication- Lazy replication does not immediately update all distributed replications, but only changes the state of the local copy at the replication server receiving the original request before sending the response to the client. The propagation of the new state of the object to the other copies is done eventually some time later. The lazy replication provides a fast response to the client, because the server receiving the original request already sends a notification response after the local update execution. However, this approach requires an additional coordination mechanism for ensuring consistency of the replication application state. Without such an subsequent mechanism, replicated object copies would diverge because each server changes objects' state independently of each other when executing the original client request. Possible mechanisms for preventing state divergence include ordering the updates by using Lamport timestamps [18] or a primary copy mechanism where a single server coordinates the updates for a particular object. Anyway, this agreement coordination for the distributed execution of an object update is performed after the client already received the notification response. Thus, lazy replication provides high responsiveness because the distributed inter-server communication is done after the actual request execution.

V. CHALLENGES IN REPLICATED DATABASES

In support of the replication approach for interactive environments presented in this paper, the primary copy approach has been chosen as its main update location scheme. In fact, the general disadvantage that the primary copy mechanism requires to forward actions affecting remote objects to another server does only apply to a subset of the possible user inputs. Most of the user inputs received by a particular server affect only the users' own entities. These entities are maintained by the same server according to the parallelisation concept and, therefore, are primary copies which can immediately be updated. By using the primary copy scheme, the replication approach takes advantage of the fact that most of the objects in virtual worlds are "owned" by a particular user who is the main source of state change actions like moving around and executing user commands. However, while a lot of the typical user actions only affect objects directly controlled by the user, there are situations where objects maintained as primary copies at other servers are affected.

VI. CONCLUSION

The scheme discussed here for distributed real time systems are analyzed more and more to conclude the problems and the results. More research work is required for implementing in time constraints for real time systems to execute the available technique for data replication in distributed system. Using a lazy primary copy synchronisation scheme concept results in data convergence of the distributed application state following the eventual consistency model due to the laziness of the replica synchronisation, there is no a priori assertion possible about when all copies will be finally updated. In contrast to a synchronous eager replication where the client can be sure that all replications are synchronised when the response arrives, the chosen asynchronous lazy scheme only guarantees that the updates happen at some point in time, thus eventually, after the client received the response. Due to the chosen primary copy update location mechanism, concurrent write access conflicts are prevented; the changes to particular objects are serialized by the single process maintaining the primary copy.

References

[1] E.G. Coffmann, E. Gelenbe, and B. Plateau, Optimization of the Number of Copies in a Distributed System, IEEE Trans. Software.

[2] H. Garcia-Molina, ªPerformance of the Update Algorithms for Replicated Data in a Distributed Database, PhD dissertation, revised, Computer Science Dept., Stanford Univ., 1982.

[3] M. Singhal, Concurrency Control Algorithms and Their Performance for Replicated Database Systems,º PhD dissertation, Dept.of Computer Science, Univ. of Maryland, 1986.

[4] W. Mariasoosai and M. Singhal, A Concurrency Control Algorithm for Replicated Database Systems, Proc. Iint'l Symp. Memory Management, pp. 143-147, Oct. 1990.

[5] A. Kumar and A. Segev, Cost and Availability Tradeoffs in Replicated Data Concurrency Control, ACM Trans. Database Systems, vol. 18, no. 1, pp. 102-131, Mar. 1993.

[6] S.H. Son and F. Zhang, Real-Time Replication Control for Distributed Database Systems: Algorithms and Their Performance, º Proc. Fourth Int'l Conf. Database Systems for Advanced Database Applications, pp. 214-221, Apr. 1995.

[7] B. Ciciani, D.M. Dias, and P.S. Yu, Analysis of Replication in Distributed Database Systems, IEEE Trans. Knowledge and Data Eng., vol. 2, no. 2, pp. 247-261, June 1990.

[8] R. GallersdoÈ rfer and M. Nicola, Improving Performance in Replicated Databases through Relaxed Coherency,º Proc. 21st Conf. Very Large Databases, pp. 445-456, Sept. 1995.

[9] G. Alonso, Partial Database Replication and Group Communication Primitives,º Proc. Second European Research Seminar Advances in Distributed Systems (ERSADS '97), Mar. 1997.

[10] T. Anderson, Y. Breitbart, H. Korth, and A. Wool, Replication, Consistency, and Practicality: Are These Mutually Exclusive, Proc. ACM SIGMOD Int'l Conf. Management of Data, June 1998.

[11] Andrew Tanenbaum and Marten van Steen. Distributed Systems: Principles and Paradigms. Prentice Hall, 2002.

[12] Matthias Wiesmann, Fernando Pedone, Andr´e Schiper, Bettina Kemme, and Gustavo Alonso. Understanding replication in databases and distributed systems. In Proceedings of 20th International Conference on Distributed Computing Systems (ICDCS), pages 464–474, 2000..

[13] Jim Gray, Pat Helland, Patrick E. O'Neil, and Dennis Shasha. The dangers of replication and a solution. In H. V. Jagadish and Inderpal Singh Mumick, editors, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996, pages 173– 182. ACM Press, 1996.

[14] Jouni Smed and Harri Hakonen. Algorithms and Networking for Computer Games. John Wiley & Sons, 2006.

[15] Martin Mauve, J¨urgen Vogel, Volker Hilt, and Wolfgang Effelsberg. Local-lag and timewarp: Providing consistency for replicated continuous applications. IEEE Transactions on Multimedia, 6(1):47–57, February 2004.
[16] Tim Gray and Andreas Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 1993. [Hal02] Pete Hallenberg. Of Internet servers and SQL databases: Designing the backend for power and performance,September2002.http://www.gamasutra.com/resource guide/ 20020916 /hallenberg 01.htm.
[17] Vassos Hadzilacos and Sam Toueg. Fault-tolerant broadcasts and related problems, pages 97–145. ACM Press/Addison- Wesley Publishing Co., New York, NY, USA, 1993.
[18] Leslie Lamport. Time, clocks and the ordering of events in a distributed system. Communications of the ACM, 21, July 1978.

Authors

Sanjay Kumar Tiwari received his Master degree in Computer Application in year 2007 presently he is working as Instructor in NSIC, Computer Centre, Ujjarpar, Gorakhpur. He has more than 4 years teaching and professional experience. His area of interest includes DBMS & Networks. He is pursuing his PhD in Computer Science on "Relative Performance Issue in Distributed Real Time Database For Replicated Data". He has published several papers in National and International conferences and Journals.

Dr. A.K. Sharma received his Master degree in Computer Science in year 1991 and PhD degree in 2005 from IIT, Kharagpur. Presently he is working as Associate Professor in Computer Science and Engineering Department, Madan Mohan Malaviya Engineering College, Gorakhpur. He has more than 23 years teaching experience. His areas of interest include Database Systems, Computer Graphics, and Object Oriented Systems. He has published several papers in National & International conferences & journals.

Vishnu Swaroop received his Master degree in Computer Application in year 2002 presently he is working as Computer Programmer in Computer Science and Engineering Department, Madan Mohan Malaviya Engineering College, Gorakhpur. He has more than 22 years teaching and professional experience. His area of interest includes DBMS, & Networks; he is pursuing his PhD in Computer Science on Data Management in Mobile Distributed Real Time Database. He has published several papers in several National, International conferences and Journals.