RMI Object Consistency Maintenance Techniques at Distributed Computing

Seong Eun Chu
Department of Computer Science
Chonnam National University
300 Yongbong-Dong
Bug-Gu
500-757, Gwangju, Korea
sechu@jnu.ac.kr

Jae Nam Kim
Department of Digital Animation
Kwangju Women's University
165 Sanjeong-Dong
Gwangsan-Gu
506-713, Gwangju, Korea
jnkim@mail.kwu.ac.kr

Dae Wook Kang
Department of Computer Science
Chonnam National University
300 Yongbong-Dong
Bug-Gu
500-757, Gwangju, Korea
dwkang@jnu.ac.kr

ABSTRACT

Various object caching techniques between a client and a server have been proposed at distributed computing. However, these techniques have handled data consistency only. They have not handled object own consistency. In this paper, we proposed two techniques for object consistency maintenance. The first technique makes it possible that the client confirms the object update time in the server based on RMI (Remote Method Invocation). The second is that the server broadcasts invalid message to the clients. Both techniques are evaluated experimentally, and results show that they could be applied selectively at the distributed applications considering object update frequency.

Keywords

RMI (Remote Method Invocation), Consistency Maintenance, Distributed Computing.

1. INTRODUCTION

New objects are created in server and methods of the objects are invoked by clients of the different memory address in distributed computing. There are several techniques such as RMI, CORBA, DCOM, EJB for the development of object oriented distributed applications. RMI does not need to install separately the middleware for supporting information communication between objects or components because it has been implemented already at JVM. Therefore, RMI has been used widely at distributed computing environments as the most simple object communicational model [Dow98]. A data caching at distributed computing environments has been used efficiently when the frequency of data access is high. Generally, caching is simple, but many elements should be considered to maintain consistency of caching.

Existing techniques about consistency maintenance have been classified by detection-based and avoidance-based categories. Detection-based algorithms permit stale data in local, examine the validity of a server when read or write operations are performed. These algorithms maintain the consistency by sending many messages to ensure the validity of data, therefore bandwidth is wide. Avoidance-based algorithms don't allow a reference opportunity about the stale data to maintain consistency. Because this consistency checks are delayed to the last, the decrease of the number of message transmission and rapid response are possible, but conflict is found late, which is increasing aborting rate. [Fra97, Fra96]. However, it is difficult for these techniques to apply to objects that have complex characteristics such as abstraction, encapsulation, polymorphism, inheritance [Wol96]. Therefore, the researches to maintain object consistency are need.

In this paper, we proposed two techniques for the maintenance of an object caching consistency based on RMI. The first technique is Time Stamp (TS) technique that compares update time in a server with cache time in a client when a client asked a server of consistency check to use a cached object. We applied TS to RMI. This modified RMI is called TS-RMI. The second is Invalid Message (IM) technique that broadcasts object updating message to all clients using this cached object when object is changed in server. This modified RMI which applies IM to RMI is called IM-RMI. We measured average response time of method invocation according to object update frequency under equal computing environments to compare TS-RMI with IM-RMI.

The rest of the paper is organized as follows. In section 2, we describe related work. The proposed consistency maintenance techniques are explained in

section 3, and experimental results are presented in section 4. Finally, we conclude our work and suggest future work in section 5.

2. RELATED WORK

In fixed computing environment, data consistency maintenance techniques have been classified by detection-based and avoidance-based. At mobile computing environments, detection-based algorithms such as NWL-NH are used between a mobile host and a base station, avoidance-based algorithms such as O2PL are used between fixed hosts [Jin95], techniques such as AT, SIG periodically broadcast a client the fact that database is updated for consistency maintenance in disconnection [Bar94].

Detection-based algorithms permit stale data in local, examine the validity of a server when read or write operations are performed. In C2PL, synchronization has been needed. A server doesn't send updating message to a client and takes the responsibility for all locking and deadlock detection. 2PL techniques are expanded from client/server environments to centralized control database environments, whereas implementation is simple, Each time they make a server perform validity check process at read or write operations, therefore, message transmission frequency is high and bandwidth is narrow [Car91, Wan91]. NWL is an asynchronous technique. Message transmission frequency is low in NWL. It is different from C2PL in write operations, and it makes server check validity. NWL-NH is adapted at mobile environments. After server broadcasts updated data items to a client periodically, a client removes invalid data items in cache [Jin95]. AOCC is a detection-based optimistic technique that allows a transaction to access cached data. It defers validity check until the transaction commits phase. In AOCC, if a transaction aborts locally, the server need not be notified as all of the transactions are performed locally until the commit. Because it is incurred primarily in the client, it makes abort cost low and performance high [Bod04].

Avoidance-based algorithms don't allow a reference opportunity about the stale data to maintain consistency. CBL is locally cached page copies are always guaranteed to be valid, so transactions can read them without contacting the server (i.e., only a local read lock is required). On a cache miss, the client sends a page request message to the server. The server returns a valid copy of the requested page when it determines that no other active clients believe they have write permission for the page. In callback locking, write intentions are declared synchronously, a client must have write permission on a page before it can grant a local write lock to a transaction. Because write permissions are obtained during transaction execution, transactions can commit after completing their operations without performing any additional consistency maintenance actions [Fra97]. ACBL is a synchronous avoidance-based algorithm as it uses lock-escalation messages in a synchronous manner, it sends a request for lock-escalation and waits for a reply before proceeding [Bod04, Gru97, and Zah97]. AACC technique is that all client/server manage lock with page and object unit, read-lock divide by private-read lock and shared-read lock. Private-read lock is cached to one client and shared-read lock is cached to several clients. AACC has high performance than ACBL, and low aborting rate than AOCC [Tam98]. O2PL acquires read and write lock locally until transaction completion, examines accuracy to a server when there is completion. It shows that transmission messages are reduced as compared with C2PL [Fra96, Car91].

3. PROPOSED TECHNIQUES

Object caching consistency means that an original object in a server is equal to a cached object in a client. The proposed techniques applied object caching consistency maintenance problem to RMI. These modified RMI that apply proposal techniques in general RMI is called TS-RMI and IM-RMI. We need some hypotheses as follows. First, objects of old version may be in cache basically. Second, Remote method invocation is happening from many clients to optimum level frequently. Also, we put Cache Manager and Consistency Manager commonly in modified RMI and take charge processing about caching and consistency maintenance respectively.

3.1 Time Stamp Technique (TS-RMI)

This technique adds to general RMI time stamp function to compare client's cache time to server's update time for consistency. Client's cached object is changed into a new object that reflects server's update time recently. Time comparison is processed by client-initiated which try to use object. Consistency is maintained, when cache time is the greater equal than update time (⑧**Valid** of [Figure 1]). Therefore, Client's Cache Manager invokes a local object method. This process shows ①-⑩ in [Figure 1].

In other case, the object was stored in caching table before a remote object is changed. That is, it is state that consistency doesn't maintain (⑦ ***Invalid*** of [Figure 1]). At this time, Server's Consistency Manager invokes remote method, then it makes Skeleton transmit result and an object to Stub through network.

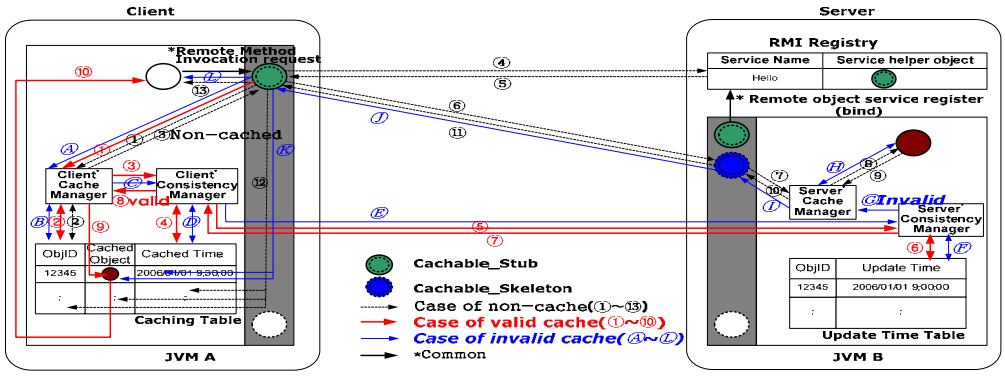

Figure 1. System Model of TS-RMI

Stub changes contents of caching table with a transmitted object (Update on Caching Table Ⓚ), returns result to a client object. This process shows Ⓐ-Ⓛ marked in [Figure 1].

When the remote method is invoked for the first time, there is no cached object in the caching table (③ Non-cached of [Figure 1]). In this case, consistency check is not necessary. Stub stores result and an object that transmitted from Skeleton in caching table (Add to Caching Table ⑫), returns result to client object. This process shows ①-⑬ marked in [Figure 1].

An advantage of TS-RMI is the fact that the consistency checked simply, the communication bandwidth is decreased by handling client-initiated consistency check. Also, server's responsibility for consistency maintenance is decreased because it is not necessary for server to send object update message.

3.2 Invalid Message Technique (IM-RMI)

In IM-RMI, as soon as a caching object is changed, a server broadcasts invalid message to clients which have ever used the remote object caching (*1-3* of [Figure 2]). Therefore, it must have information about all clients that possesses a cached object on broadcasting message table in a server. Whenever caching happens, Server's Cache Manager keeping client's IP-Address in own broadcasting message table.

On the other hand, a client maintains consistency by checking transmitted state of invalid message. Consistency is maintained when client did not receive invalid message (⑤ **Valid** of [Figure 2]), therefore, Client's Cache Manager invokes a local object method. This process shows ①-⑦ in [Figure 2].

In the case of receiving a invalid message, consistency is not maintained (Ⓔ ***Invalid*** of [Figure 2]), then Client's Consistency Manager makes Stub invoke a remote method, gets result and a object by Skeleton from Server's Cache Manager, stores updated object in client's caching table, and sets invalid message field as default value (zero) for next invocation (Update on Caching Table Ⓜ). This process shows Ⓐ-Ⓝ marked in [Figure 2].

When the remote method is invoked for the first time, there is no cached object in the caching table (③ Non-cached of [Figure 2]). In this case, consistency check is not necessary. Stub stores result and an object that is transmitted from Skeleton in caching table (Add to Caching Table ⑬), returns result to a client object. This process shows ①-⑭ marked in [Figure 2].

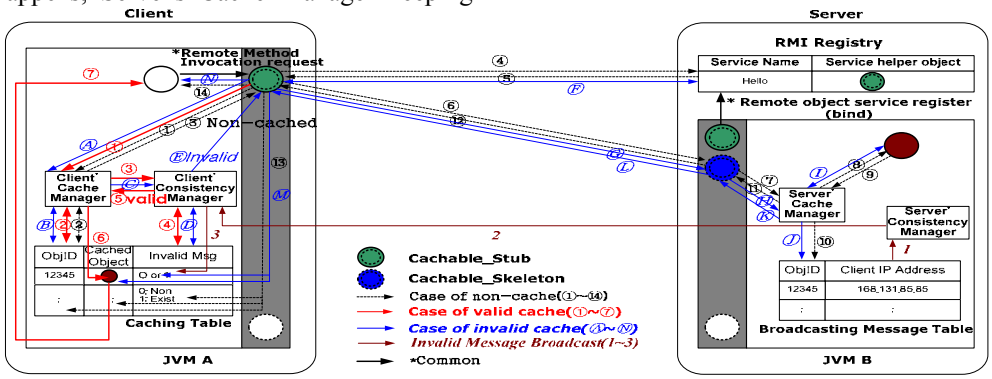

Figure 2. System Model of IM-RMI

An advantage of IM-RMI is the fact that the consistency check around time is decreased because client access to server only when client has received invalid message. Therefore, this RMI can get fast response time.

4. PERFORMANCE COMPARISON

We experimented with server such as Pentium IV 3.0GHz, Main Memory 2GB, Windows XP, Desktop Computer, Marvell Yukon 88E8001 PCI Gigabit Ethernet Controller, LAN 100Mbps, and client such as Pentium IV 1.6GHz, Main Memory 256MB, Windows XP, IBM Laptop Computer, Orinoco Wireless LAN PC Card (5volt), WaveLAN 11Mbps. Both a server and a client are installed in JDK 1.5.0_06.

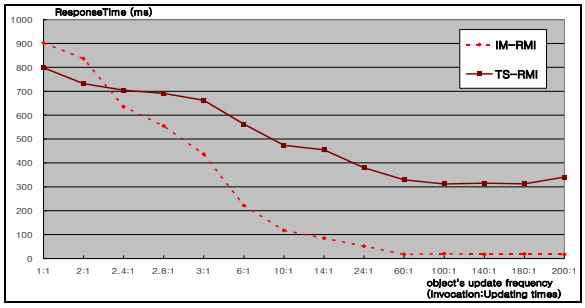

Figure 3. Average Response Time Comparison TS-RMI with IM-RMI

We measured response time respectively, while a method is invoked 30 times, object updated intervals are from 500 ms to 100,000 ms, and then compared TS-RMI with IM-RMI under equal computing environments. Elapsed time of remote method takes about 500 ms. In case an object update cycle takes 500 ms, whenever a method invoked, an object was changed (1:1). In case an object update cycle takes 100,000 ms, when a method is invoked 200 times, an object was changed only one (200:1).

As experimental results, the more object update is, TS-RMI is faster than IM-RMI in response time. Otherwise, IM-RMI is faster than TS-RMI in response time as shown in [Figure 3].

5. CONCLUSIONS

In this paper, we proposed two techniques for object consistency that has complex characteristics such as abstraction, encapsulation, polymorphism, inheritance. We experimented based on RMI. One is TS-RMI that compares update time of a server with cache time of a client when client checks consistency to server for using cached object. The other is IM-RMI that broadcasts an object updating message to all clients using this cached object when object is changed in server. As results, TS-RMI was efficient when the frequency of server object update is high, and IM-RMI was efficient that server's object update is infrequent. Therefore, modified RMI could apply selectively at the distributed applications considering object update frequency.

We have presented mechanisms for efficient caching consistency of objects for RMI applications. Using these mechanisms, caching can be easily and transparently added to existing RMI applications, while preserving RMI compatibility. The central mechanism includes the ability to support the Cache Manager and Consistency Manager. Stub and Skeleton class that generated by RMIC are modified by CRMIC. No changes are required to existing Client and Server applications. Also existing RMI performance can be enhanced without losing backward-compatibility. The future work of this study will be additional technique for mobile computing environments.

6. REFERENCES

[Bar94] Barbara D., Imielinski T., ″Sleepers and Workaholics: Caching Strategies in Mobile Environments″, ACM SIGMOD, pp. 1-12, 1994.

[Bod04] Bodorik P., Jutla D., Lu Y., ″Interoperable Server-based Cache Consistency Algorithm″, IEEE Database Engineering and Applications Symposium, 2004.

[Car91] Carey M., Franklin M., Livny M., Shekita E., ″Data Caching Tradeoffs in Client-Server DBMS Architectures″, ACM SIGMOD, pp. 357-366, 1991.

[Dow98] Downing T. B., ″Java RMI: Remote Method Invocation″, IDG Books, pp. 13-15, 1998.

[Fra96] Franklin M., Carey M., ″Client Data Caching: A Foundation for High Performance Object Database System″, Kluwer Academic Publishers, 1996.

[Fra97] Franklin M., Carey M., Livny M., ″Transactional Client-Server Cache Consistency: Alternatives and Performance″, ACM TODS, pp. 315-363, 1997.

[Gru97] Gruber R., ″Optimism VS. Locking: A Study of Concurrency Control for Client Server Object-Oriented Databases″, Ph.D Thesis, MIT, 1997.

[Jin95] Jin Jing et al., ″Distributed Lock Management for Mobile Transaction″, IEEE Distributed Computing System, 1995.

[Tam98] Tamer M., Kaladhar M., ″An Asynchronous-Based Cache Consistency Algorithm for Client Caching DBMSs″, VLDB, pp. 440-451, 1998.

[Wan91] Wang Y., Rowe L., ″Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture″, ACM SIGMOD, pp. 367-376, 1991.

[Wol96] Wollrath A., Riggs R., Waldo J., ″A Distributed Object Model for the Java System″, 2nd Conference on Object-Oriented Technologies, 1996, Toronto, Ontario, Canada.

[Zah97] ZahariousDakis M., Franklin M., ″Adaptive, Fine Grained Sharing in a Client-Server OODBMS: A Callback-Based Approach″, ACM TODS, pp. 570-627, 1997.