Associating Replication and Recovery Protocols For Replicated Databases *

Luis H. García-Muñoz, J. Enrique Armendáriz-Iñigo, Francisco D. Muñoz-Escoí
Instituto Tecnológico de Informática, Valencia, España
{lgarcia, armendariz, fmunyoz}@iti.upv.es
Technical Report ITI-ITE-06/08

Database Replication consists in storage copies of the data in multiple sites for increasing availability and performance. However, it is necessary to maintain copies updated to ensure consistency. The recovery task basically consists in transferring the information lost during the failure interval, from one or more replicas that have been active to recovering replicas, without altering the service capability.

A good replication system must use simple technics, tolerate work overload, keep consistency, provide operability and avoid transactions rollback [6]. In a similar way, a good recovery protocol must use simple technics, distribute in an efficient way the recovery work among available sites and allow the concurrency.

The main goal of this work is to present a revision of the replication and recovery protocols developed in the last years so that the reader can observe the different alternatives and strategies used. Besides, the concepts related to Group Communication Systems (GCS) and virtual synchrony [4] are also reviewed. We will present a schematic comparison through a table that summarizes the characteristics found.

In order to compare the replication protocols we will consider [10]: 1)Server architecture(SA). Where are the transactions executed in the first place, it can be primary copy (PC) or update everywhere(UE). 2)Server interaction(SI). Interaction between servers that contain replicas. Constant interaction(C) or linear interaction(L). 3)Transaction termination(TT). The way transactions terminate: voting termination (V) or non-voting termination(NV). 4)The way data is updated(U): eager update(E) or lazy update(L).

For recovery protocols we consider: 1)Transference model (TM). How the actual state of the database is transferred: Full database transfer (FDB), object version based (VB) or resend lost messages (LB). 2)How concurrency control is made during the recovery: in an optimistic(O) or pessimistic(P) way, with a unique(U) or distributed(D) manager and if it is version based(V),(Y)es/(N)o. 3)The way to distribute the recovery work(WD): centralized(C) or distributed(D).

*This work has been funded by the MCYT and the MEC under projects: TIC2003-09420-C02 and TIN2006-14738-C02.

When the primary target is to assure the replica consistency, generally the eager update algorithms are used with better results; when the primary target is performance, better results with the lazy update algorithms are obtained.

If both the replication and recovery protocols are managed at the middleware layer they can be easily adapted to a variety of database management systems [8].

Considering the analysis made in [5] and this work, in general we can say that older protocols do not consider broadcast protocols based on GCS, and were made based on optimistic or pessimistic concurrency control with voting transaction termination. Recently developed algorithms propose to make widely use of GCS to implement replica control, such as the use of total order multicast for writeset delivery and used for conflict resolution too, membership service for provide to system with a list of all active sites, and handling the primary partition model and virtual synchrony. This provides better performance than traditional eager replica control mechanisms. The use of virtual synchrony and Enriched View Synchrony (EVS) aids the system to have a more realistic state of updated members encapsulating the reconfiguration process [9].

We can observe that the use of loggers can simplify the replica and recovery task [6], but may increase the work in the sites or the amount of messages and information to store and send for achieving consistency. Other ambiguous things remains for study, such as the use or not of a site as primary logger for recovery: how big will the log become? or how long must the loggers record the lost messages for a failed site waiting for its recovery?.

The proposals for doing the recovering task in a parallel form are a good idea for load balancing and time optimization as in [8], but in that case, they are restricted to a particular modeling for the database.

To combine the advantages of several replication and recovery algorithms without their disadvantages would be desirable but in some cases as in [7] and [2] is not fully possible. In [7] a configurable replication protocol is presented. This protocol can be configured for eager, lazy or combined eager and lazy update. In addition to this, the re-

covery protocol is implicit in the replication algorithm. This provides several advantages, such as a reduction in the abortion rate, do not suspend the system activity during site recovery, avoid the necessity of log maintenance and no more code for recovery algorithm must be added, but the transaction service time is usually longer than that of pure lazy database replication protocols. In the framework presented in [2] we could take the advantages for version-based and log-based recovery protocols.

Minimizing the amount of information being transferred with version-based strategies or combining it with log-based strategies, distributing the recovery work for balance, execution of transactions during the recovery time and low space requirements for lost objects or messages are the desirable characteristics for replication and recovery database protocols. Several of these characteristics can be found in [3], where only a more generalized application for the protocol would be desirable and a possibility for performance decrement exists when we need to explore the writeset for recording the identifiers of the objects when there are some failed sites. Elsewhere, in [1] these disadvantages are discarded and some tests will be done to confirm its advantages and provide performance measurements.

References

[1] J. E. Armendáriz. *Design and Implementation of Database Replication Protocols in the MADIS Architecture*. PhD thesis, Univ. Pública de Navarra, Pamplona, Spain, Feb. 2006.

[2] F. Castro, J. Esparza, M. I. Ruiz, L. Irún, H. Decker, and F. D. Muñoz. Clob: Communication support for efficient replicated database recovery. In *PDP*, pages 314–321, 2005.

[3] F. Castro, L. Irún, F. García, and F. D. Muñoz. Fobr: A version-based recovery protocol for replicated databases. In *PDP*, pages 306–313, 2005.

[4] G. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. In *ACM Computing Surveys 33(4)*, pages 1–43, 2001.

[5] L. H. García-Muñoz, J. E. Armendáriz, and F. D. Muñoz. Recovery protocols for replicated databases - a minimal survey. Technical Report ITI-ITE-06/07, Inst. Tecnológico de Informática, Valencia, Spain, Oct. 2006.

[6] J. Holliday. Replicated database recovery using multicast communication. In *NCA*. IEEE-CS Press, 2001.

[7] L. Irún, F. Castro, F. García, A. Calero, and F. Muñoz. Lazy recovery in a hybrid database replication protocol. In *XII Jornadas de Concurrencia y Sistemas Distribuidos*, 2004.

[8] R. Jiménez, M. Patiño, and G. Alonso. An algorithm for non-intrusive, parallel recovery of replicated data and its correctness. In *SRDS*, pages 150–159. IEEE-CS Press, 2002.

[9] B. Kemme, A. Bartoli, and Ö. Babaoglu. Online reconfiguration in replicated databases based on group communication. In *DSN*, pages 117–130. IEEE-CS Press, 2001.

[10] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. Database replication techniques: a three parameter classification. In *SRDS*, pages 206–215. IEEE-CS Press, 2000.

Table 1. Classification of replication and recovery protocols

		Replication				Recovery				
		SA	SI	TT	U	TM	O/P	D/U	V	WD
[9]	Full DB Transfer	UE	C	NV	E	FDB	P	U	N	C
	Version number	UE	C	NV	E	VB	P	U	N	C
	Restrict set of objs.	UE	C	NV	E	VB	P	U	N	C
	Log Filter	UE	C	NV	E	VB	P	U	Y	C
	Lazy data Transf.	UE	C	NV	L	VB	P	U	N	C
[6]	Bcast writes Log upd.	UE	L	NV	E	LB	P	U	N	C
	Bcast writes Augm. bcast	UE	L	NV	E	LB	P	U	N	C
	Delayed bcast Log upd.	UE	C	NV	E	LB	P	U	N	C
	Delayed bcast Augm. bcast	UE	C	NV	E	LB	P	U	N	C
	Single bcast	UE	C	NV	E	1	P	U	N	C
[8]		PC	C	NV	E	LB	P	U	N	D
[7]		UE	C	V	2	VB	O	D	Y	D
[2]	CLOB	UE	C	NV	E	3	O	D	Y	C
[3]	FOBr	UE	C	V	E	VB	O	D	Y	D
[1]	BRP	UE	C	V	E	VB	O	D	Y	D
	ERP	UE	C	NV	E	VB	O	D	Y	D
	TORPE	UE	C	NV	E	VB	O	D	Y	D

1. Considers full database transfer, is needed if a site is new or if it is not in the record of views in the logger.
2. It is configurable, and may be hybrid.
3. VB for long-term failures, and LB for short-term ones.