# OPTIMISTIC CONCURRENCY CONTROL WITH PARTIAL REPLICATION DESIGN

August Climent, Joan Navarro, Miquel Bertran, Francesc Babot
*Research Group of Distributed Systems*
*La Salle, Ramon Llull University*

## ABSTRACT

This paper presents a performance comparison of optimistic concurrency control algorithms using a database design with partial replication. Most studies always assume total replication design, but this scenario is not valid when the network latency is high and with a high number of system sites. So, this paper presents the optimistic algorithms and the simulation results under different scenarios of network latency, number of sites and system load. Finally, the paper shows that partial replication design is a good alternative that improves system throughput.

## KEYWORDS

Database Systems, Parallel and Distributed Systems.

## 1. INTRODUCTION

A current trend in distributed database systems is the increasing demand for large queries in a transactional approach [7,20] with replicated data and mobile users [6,19]. To execute these queries preserving data consistency keeping the good system performance is an open research area [1,18].

The concurrency control algorithms try to improve the system throughput, keeping the information consistent. The most widely used concurrency control algorithm is the pessimistic strict two phase locking s2pl [3,10], based on locking the data to preserve their consistency, but these locks can produce an increase in data contention [2,11] and a performance degradation. An alternative to pessimistic algorithms are optimistic algorithms such as Optimistic Two Phase Locking or Basic Optimistic [5,16,17]. In this type of algorithms all operations are assumed to be compatible so they are executed when requested and never delayed. Each transaction makes its updates on all replicas of data items and at the end of the transaction it checks if the updates would maintain the database consistency [13,15] assuming that conflicts between operations are unusual.

Optimistic algorithms present three problems in front of pessimistic ones: a higher storage cost, starvation and a higher restart ratio. The restart ratio can be improved preventing the abort transactions when the transaction finishes. This is done by the brp protocol [4,12] when the site checks the transaction correctness at commit time. In most protocols there is an increase of network messages and communication overhead when we scale the number of servers. The brp protocol only sends the WriteSet propagation messages at the end of the transaction and minimizes the number of messages of all the protocol.

With this approach the global serializability is guaranteed and so at commit time the ROWA model is applied and the updates are propagated to all system sites (n) with (4n) messages for every transaction, independently of the number of updates. In a s2pl algorithm, for every update the algorithm needs (4n) messages because the transaction coordinator manages the data consistency among the sites.

The main contribution of this work is to study the performance of these optimistic concurrency control algorithms using a database design with partial replication. Most studies assume total replication design, but this scenario is not valid when the network latency is high and with a high number of system sites.

Section 2 presents and describes the system model, section 3 presents the algorithms and section 4 shows the experiments and performance evaluation. Finally section 5 concludes the paper with a summary of the contributions and future work.

## 2. SYSTEM MODEL

We have developed a single Distributed Database based on a set of servers, one scheduler and one network. The scheduler sends and coordinates the database operations to the servers. The servers execute the operations and return to the scheduler the execution result. Depending on the concurrency control algorithm used, the server decides where and when to execute the operation and the process of operation validation. When the transaction coordinator receives a termination operation executes the termination protocol based on 2pc protocol [14]. The server sends a Vote frame to all the transaction participants and when receives all the responses then decides and coordinates the execution of the termination operation.

As shown in Figure 1, we have implemented a full mesh network, where each site has an input queue fmNet() and an output queue toNet(). These queues dispatch the operations based on a network delay. At the end of simulation, the scheduler writes the system and network metrics with the simulation results into the output file Out.txt. This architecture has been developed with the PADD/RALE simulation tool [8,9].

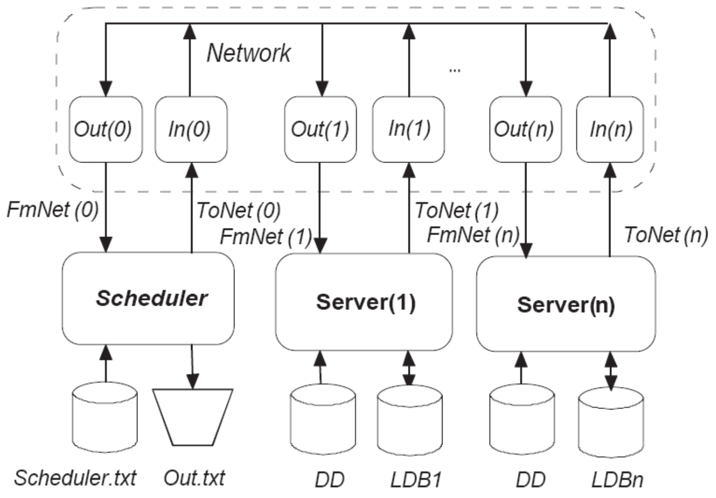

Figure 1. System architecture.

To compare the behavior of the concurrency control algorithms used in our simulations, we use the Throughput metric. It measures the mean average committed transaction for an interval of time (one second).

If we compare two algorithms with the same system parameters, the algorithm that has a high value of Throughput means that commits more transactions by unit of time, and so it has a better performance. The system parameters define the disk, network and load model used in the simulations [Table 1].

Table 1. Simulation Model Parameters.

| Parameter | Description | Domain |
|---|---|---|
| IO | Disk access time | 1..10 mseg |
| COM | Time to send or receive a message | 0, 0.1..1 mseg |
| NS | Number of sites | Low(10),Medium(20),High(50) |
| NT | Number of transactions in schedule | Low(5),Medium(10),High(20) |
| CR | % of operations with conflict | Low(5),Medium(15),High(30) |
| RL | % of sites with replication | No(0),Low(25),Medium(50),High(75),Full(100) |
| TT | Type of transaction | Short(10,40,30),Long(30,40,30),Query(30,0,0) |

IO is the disk access time. COM is the time to send a frame between two sites. The NT are the transactions that the scheduler sends to the servers. CR is the % of operations that produce a conflict. The RL defines the % of sites that have a replica for every data. Transaction Type TT defines the different type of transactions and its based on three parameters: Transaction size, Transaction Write Access TWA and Transaction Read-Write Dependency TRWD. We use three different transaction types. Short for transactions with TS of 10 operations, TWA of 40 and TRWD of 30. Long for transactions with TS of 30 operations, TWA of 40 and TRWD of 30. Transaction Type Query is a read-only transaction with 30 operations.

## 3. ALGORITHMS DESCRIPTION

In our simulations we have studied three algoritms: ndcc, bocc and brp. The **ndcc algorithm** executes all the operations when arrive in a Read-One Write-All model. The CCM manager of this algorithm does not make any validation and so we use this algorithm as a reference model.

The **bocc algorithm** assumes that the conflicts among transactions are unusual. When the CCM receives an operation always sends the operation to the data processor. When the Transaction Manager receives a commit operation, all the transaction participants must validate if all the operations assumed to be compatible are actually compatible. When a transaction has to decide whether all the operations done are correct, it has to check whether one of the next two conditions are true. First, whether or not the transaction began later than all the others which have already finished. Otherwise the transactions would be concurrent. If there are concurrent transactions and all the operations executed are compatible, the transaction can be finished. So, if one of the two conditions is satisfied, the transaction can commit, otherwise it is aborted.

The **brp algorithm** executes locally the transaction operations. So, throughout the operation execution, the information is not propagated to the replica sites. If the transaction coordinator or Master Site receives an abort, executes the abort locally. In this situation, since the operations have not been propagated to all the sites, it is not necessary to undo these operations.

If the Master Site receives a commit, then it sends a vote message to all the sites (participant) in order to decide the final operation. Then, it waits for the response from all the participants. If all the participant sites answer with an OK message, then the Master Site commits the transaction since all the participants can commit the transaction. But if at least one participant site answers with a KO message, the transaction is aborted. In order to commit or abort a transaction, the Master Site executes the termination operation locally and then sends the operation to all the participant sites.

The details of this validation phase are in [4,12]. So, the brp algorithm executes the participant operations at the end of the transaction and it saves a lot of messages throughout the transaction execution.

## 4. ALGORITHMS EVALUATION

This section introduces three experiments showing the algorithms performance.

### 4.1 Experiment 1. Communication Cost

This experiment is performed to analyze the network behavior below the three algorithms. In order to analyze the network bandwidth impact, we have left the COM as variable, taking values of 0.1, 0.2, 0.4, 0.6, 0.8, 1 ms. Figure 2 shows the system Throughput and we can see that brp's throughput is better than bocc regardless the communication cost.

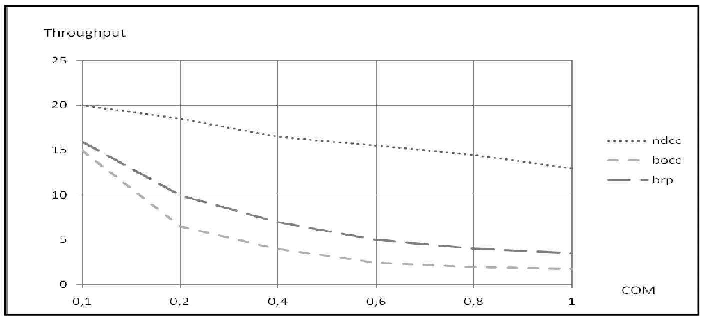

Figure 2. Throughput variation in function of Communication Cost.

## 4.2 Experiment 2. System Scalability

This experiment studies the effect of the number of sites to the system performance and shows that the best performance takes place when the number of servers is large. Depending on the number of sites of the distributed database, the system metrics vary on a considerable way. The metric used is the Throughput.

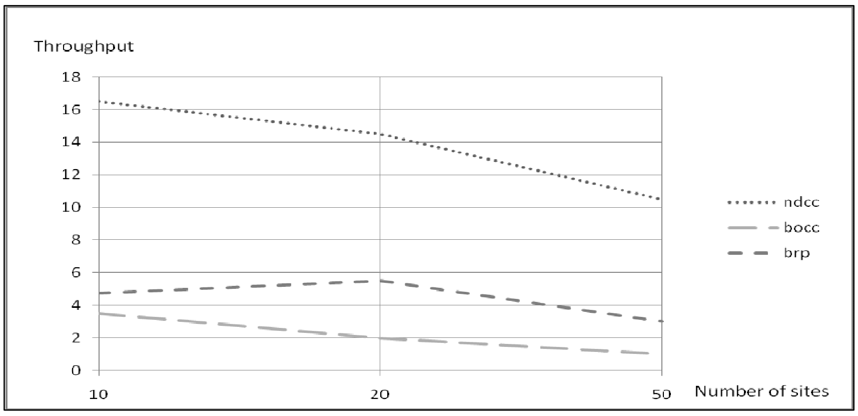

Figure 3. Throughput variation in function of Number of Sites.

We can see that brp algorithm is always better than bocc regardless the number of sites in the system.

## 4.3 Experiment 3. Level of replication

The objective of this experiment is to analyze the system throughput with different levels of replicacion and to show that partial replication design is a good alternative that improves system throughput with optimistic concurrency control algorithms. We show the simulation results with 50 sites and COM = 0.5 ms.

We can see that brp's throughput is better than bocc for high values of replication level, but when there is not replication or with low values of replication level (25%), the best algorithm is bocc.

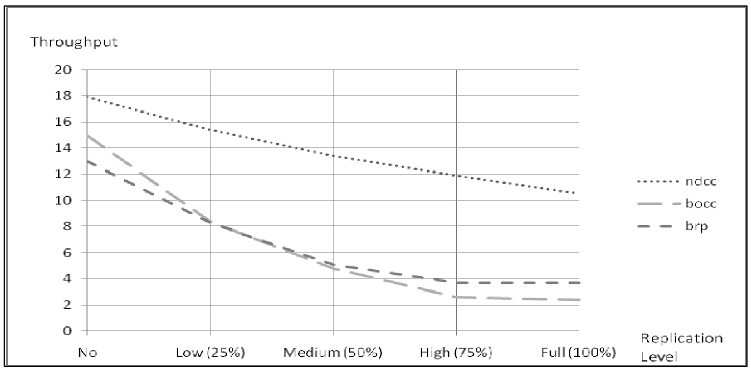

Figure 4. Throughput variation in function of Replication Level with 50 sites.

## 5. CONCLUSIONS AND FUTURE WORK

We have introduced the brp algorithm and shown that the performance is better than the standard optimistic algorithm. This improvement is based on two facts. First, as a result of propagating operations only at commit time, it saves a lot of network frames and input/output operations. Second, aborting conflictive transactions prematurely prevents the execution of unnecessary operations.

In our simulations we also show these improvements under different environment configurations. The system throughput improves with low level of replication independently of communication cost and the number of system sites. The brp's throughput is better than bocc for high values of replication level and with low levels of replication the best algorithm is bocc.

A further analysis would be required in order to study the system throughput improving the network model architecture. We also think that this work could be completed studying total order broadcast networks.

## ACKNOWLEDGEMENT

Whe recognise the support of the Ministerio de Educación y Ciencia, TIN2006-14738-C02-02 and the grant from Enginyeria La Salle, URL to the Research Group of Distributed Systems GRSD-2008/01.

## REFERENCES

[1] Abdelguerfi, M. and Kam-Fai Wong, K.F., 1998. *Parallel Database Techniques*. IEEE Computer Society Press, Los Alamitos, USA.

[2] Agrawal, D. et al., 1994. The Performance of Protocols Based on Locks with Ordered Sharing. *IEEE Transactions on Knowledge and Data Engineering*, Vol.6, No.5.

[3] Al-Jumah, N.B. et al, 2000. Implementation and modeling of two-phase locking concurrency control, a performance study. *Information and Software Technology,* Vol.42, pp.257-273, 2000.

[4] Armendáriz-Íñigo, J.E., 2006. Design and Implementation of Database Replication Protocols in the MADIS Architecture, *PhD Thesis*, Universidad Pública de Navarra, Spain.

[5] Armendáriz-Iñigo, J.E. et al, 2006. Proof and Evaluation of a 1CS Middleware Data Replication Protocol Based on O2PL. *4th Int. Symposium Parallel and Distributed Processing and Applications*. LNCS(4330), pp.524-537.

[6] Barbara, D., 1999. Mobile Computing and Databases - A Survey. *IEEE Transactions on Knowledge and Data Engineering*, Vol.11, No.1, pp.108-117.

[7] Barghouti, N.S. and Kaiser, G.E., 1991. Concurrency control in advanced database applications. *ACM Computing Surveys*, Vol.23, No.3, pp.269-317.

[8] Bertran, M., et al., 2009. PADD Reference Manuals, http://www.gsystemsd.com/paddrale.htm, Barcelona, Spain.

[9] Bertran, M. et al, 1999. The PADD/RALE environment for parallel-distributed software development. *Actas de las VII Jornadas de Concurrencia,* Gandia, Spain.

[10] Climent, A. et al., 2000. Control de concurrencia en bases de datos distribuidas. *Actas de las VIII Jornadas de concurrencia,* Cuenca, Spain.

[11] Franaszek, P.A., et al, Concurrency Control for High Contention Environments. *ACM Transactions on Database Systems*, Vol.17, No.2, pp.304-345, June 1992.

[12] Juárez-Rodríguez, J.R. et al, 2006. Implementing O2PL Protocols in a Middleware Architecture for Database Replication. *In 14th Euromicro Conference on Parallel, Distributed and Network-Based Processing.*

[13] Kung, H.T. and Robinson, J.T., 1981. On Optimistic Methods for Concurrency Control. *ACM Trasactions on Database Systems*, pp.213-226.

[14] Stonebraker, M. and Skeen, D., 1983. A formal model of crash recovery in a distributed system. *IEEE Transactions on Software Engineering*, Vol.9, No.3, pp.219-228.

[15] Thomas, R.H. 1979. A Majority Consensus Approach to Concurrency Control for Multimple Copy Databases. *ACM Transactions on Database Systems*, pp.180-209.

[16] Thomasian, A., 1998. Concurrency Control: Methods, Performance, and Analysis. *ACM Computing Surveys*, Vol.20, No.1, pp.70-119.

[17] Thomasian, A., 1998. Distributed Optimistic Concurrency Control Methods for High-Performance Transaction Processing, *IEEE Transactions on Knowledge and Data Engineering*, Vol.10, No.1, pp.173-188.

[18] Valduriez, P. and Ozsu, M.T., 1999. *Principles of Distributed Database Systems*. Prentice-Hall, New Jersey, USA.

[19] Vandermeer, D.E., 1999. Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users. *ACM Transactions on Database Systems,* Vol.24, No.1, pp.1-79.

[20] Wieczerzycki, W., 1995. Long-duration transaction support in design databases. *Proceedings of the fourth international conference on Information and knowledge management*, pp.362-369.