Transaction Overhead Reduction by Server Localization in Bank Database Management Systems

K. Sriraghav
Student
SSN College of Engineering

R. Vijayaraghavan
Student
SSN College of Engineering

S. Shriram
Student
SSN College of Engineering

Shomona Gracia Jacob, PhD
Associate Professor
SSN College of Engineering

ABSTRACT
Server localization refers to introduction of local servers that are connected to a main centralized server. The banking system currently prevailing in the country maintains two databases: one for transaction management while the other is maintained for backup. The transactional data from various branches across the country is maintained by a centralized server. In this scenario, given one centralized server, the access overhead becomes too high since all the branches access the main server only. As a part of day-end closing, the database in the main server is duplicated. The proposed system introduces an algorithm named Transaction Overhead Reduction by Localization of Servers (TORLS) – where servers are locally placed - region wise and they contain local databases pertaining to that region. Hence, for intra-regional transactions, it is sufficient that the local servers alone are accessed. For inter-regional transactions, the two region-based local servers are accessed via the main server. The measure of decrease in the overhead is calculated as the number of intra-regional transactions. The duplication server is optional since the integration of local server databases will constitute the main database. The main database is updated at the end of each day thus alleviating the need for duplication.

General Terms
Transaction management, Banking data

Keywords
Server localization, access overhead, inter-regional transactions, intra-regional transactions, replication

1. INTRODUCTION
There are two kinds of databases used in general - centralized and distributed databases. A centralized database is generally located and maintained in a single location which is a computer system server in most of the cases. Banking systems use a server CPU. All of the information stored on the centralized server is accessible from a large number of different locations. Centralized Database (CDB) is easily accessible to the end-user due to the simplicity of having a single database design. Also, data kept in the same location can easily be changed, re-organised, mirrored, or analysed. But the main disadvantage of CDB is that a large number of simultaneous transactions access the server data leading to server trafficking.

On the other hand, a distributed database system (DDB) consists of a number of sites that have minimal or no knowledge about the other sites in the network. These sites usually share no physical components. The distributed databases store data across multiple computers and hence they improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to a single site. DDB allows transparency to be achieved across various levels and hence it promotes increased reliability and availability. Distributed query processing has proved to improve the performance of the system. It also ensures ACID properties that any database must support. Distributed databases usually rely on replication to support increased availability and reliability metrics.

Taking into account the pros and cons of the centralized and distributed databases; the idea of server localization is proposed in this paper that combines the features of both CDB and DDB. The proposed idea reduces the server overload and increases the throughput while maintaining concurrency and parallelism.

1.1 Existing Banking System:
The hierarchy of organization present in a general banking system in India is as given below:

There will be a Head Office present in any of the main cities in the nation. A main centralized server will be located in this Head Office. There will be many regions based on the geographic zones. Each region will have many branches under its control.

The banking operations can be related to the database operations as follows:

1. CREATE – Create a table for each branch in the database server.
2. INSERT – Create a new tuple in the branch table feeding in the customer details, after he opens an account in that particular branch.
    a. A sample SQL query can be:

INSERT INTO branchname VALUES (ac_no, cust_id, custname, age, gender, bal, passwd);

3. UPDATE – For each transaction, the values in the database has to be updated.
    a. If an amount ‘x’ has to be transferred from ‘ac_A’ of ‘branch_from’ to ‘ac_B’ of ‘branch_to’ , the following queries have to written in the banking software:
    i. UPDATE branch_from set bal = bal- x where ac_no=”ac_A”;

ii. UPDATE branch_to set bal = bal + x where ac_no=”ac_B”;

4. RETRIEVE – For processing customer queries which include balance enquiry,

    a. The banking software implements the query.

        i. SELECT bal from branchname where ac_no= account_number;

5. DELETE – Delete the whole tuple of the customer after he closes the account in a particular branch.

    a. A sample SQL query will be:

i. DELETE FROM branchname where ac_no= account_number;

A transaction in a general banking system can be broadly classified as –

1. Inter-regional transactions – transactions involving branches from different regions
2. Intra-regional transactions - transactions involving branches from the same region.

2. CURRENT SCENARIO

The software used in most of the banking organizations currently has two main servers - one for access and other for duplication - they use centralized server. The replication server replicates the database available in the main server as a backup, at the end of each day. Several branches across the country may access this centralized server for each and every transaction. As every transaction requires the execution of three sub operations for its processing, the access overhead on the server is usually high. So if the connection is maintained with the main server, other transactions will be queued to get processed until the current transaction is completed. This occurs irrespective of the region from which the transaction occurs. Consequently, the overload on the server and the waiting time of the other transactions becomes very high.

The access overhead in the main server due to the transactions has to be computed for both intra-regional and inter-regional transactions. Note that there are no local servers in each region to store the transaction data pertaining to that region. So for each transaction, the main server is accessed.

Irrespective of intra-regional or inter-regional transactions, the banking software at one branch updates the main server with the changes in its table. For a transaction involving more than one region, the databases of the other branch in the main server has to be updated, as shown in the UPDATE operation above.

The diagram depicts the structure of the existing system. All the regions have to access the main server for all types of transactions.

An important thing to note is that if the banking systems use a centralized server, there will be no difference between inter-regional and intra-regional transactions, as all the transactions invariably access the main server.

2.1 Pitfalls in the existing system:

Usage of centralized servers for banking software suffers from the following disadvantages.

- Even for intra-regional transactions, the centralized server has to be accessed.
- When there are many transactions queued at the server to be processed, the waiting time of the transactions become high.
- Access overhead on the server whenever multiple simultaneous transactions try to access transactional data on the server.
- Bottlenecks occur as a result of high network traffic.

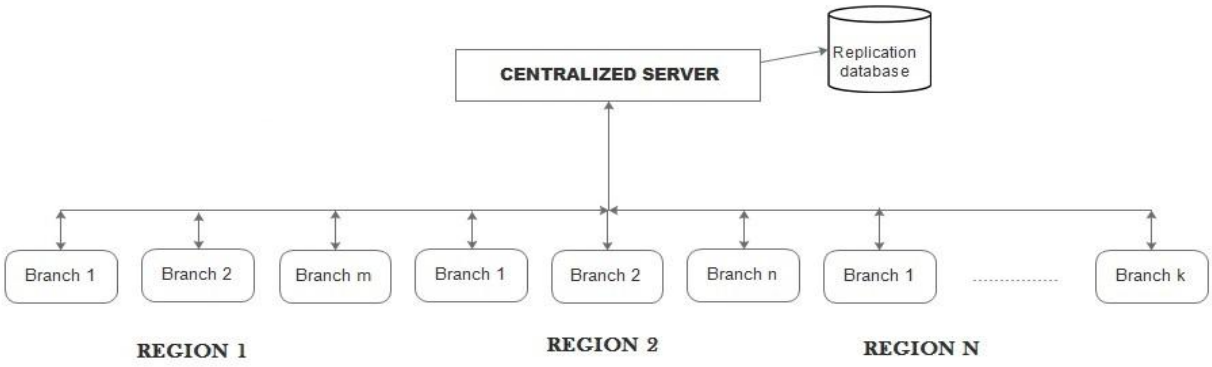

Fig 1: Structure of the existing banking system

3. MOTIVATION

A banking transaction consists of a series of operations to be performed on a database. The important issue in transaction management is that if a database was in a consistent state prior to the initiation of a transaction, then the database should return to a consistent state after the transaction is completed. This should be done irrespective of the fact that transactions were successfully executed simultaneously or there were failures during the execution. Above all this, it must be ensured that a transaction does not lead to inconsistency of data.

Usually, a banking transaction involves three operations namely, retrieval of account information from the server, manipulation of data depending upon the type of transaction

and updating the changes to the server. For every transaction that is processed in any branch the centralized server is accessed. Since a large number of transactions try to access the server simultaneously, it adds to a large increase in the overhead thereby causing server overload.

4. PROPOSED SYSTEM

The existing system has many disadvantages that open a wide area for improvement. The proposed idea focuses on reducing the overhead access of the main server by introducing localised servers, i.e., placing servers locally (region-wise).The introduction of servers locally means that a second tier in the hierarchy of the existing system is inserted.

4.1 Local Servers

The local server placed in each of the regions has the database pertaining to that region alone. The transactions which happen in that region update the corresponding local server which in turn updates the main server at the end of the day. When the databases of each of the local servers are integrated, the original databases are retrieved. Hence the duplication server is an optional server.

Since the database pertaining to that region is stored in that local server, it is sufficient if the intra-regional transactions access the local servers alone, hence avoiding the accessing of main server as in the existing system, whereas in the case of inter-regional transactions the main server is accessed via the local servers.

4.2 Operations in the proposed system:

1. The branch initiating the transaction will first check if the transaction is inter-regional or intra-regional.
2. If the transaction is intra-regional
    a. The regional server is accessed and the tables of ‘m’ branches involved in the transaction are updated.
    b. As a part of day-end closing, check these transactions and make an entry of these transactions in the main server.
3. If the transaction is inter-regional
    a. The banking software in the branch initiating transaction will first find the region in which the other branch is located.
    b. The local server is accessed through the main server, i.e. the tuples are retrieved from each of the local server.
    c. After the transaction is completed, each of the local servers is updated.
4. As a part of day-end closing, the databases from each of the local server are updated on to the main server. The duplication server is optional.

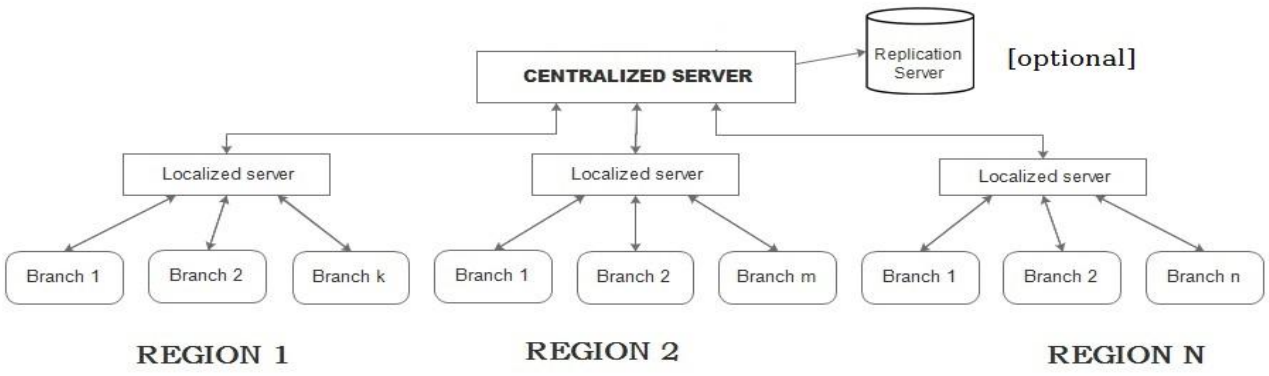

Fig 2: Localization of servers in the structure of the banking system

The following algorithm is used to process both inter-regional and intra-regional transactions in the proposed system:

4.3 Algorithm

TORLS - Perform transaction initiated by the source branch

Input: Transaction details - Source_Account_no A1, Destination_Account_ no A2, amount Amt

Values known: Branch code of the source branch B

Procedure

Obtain the values of b1 and b2 as branch codes of A1 and A2 respectively;

if (b1==B&&b2==B)

goto intra;

else

goto inter;

intra: do

(i) Obtain the account details of the account no A1 and A2 from the regional server

(ii) Update the balance values of both the accounts as

UPDATE B set bal = bal- x where ac_no=”A1”;

UPDATE B set bal = bal + x where ac_no=”A2”;

Where B is a table on the regional server.

inter: do

(i) Obtain the account details of the account no A1 from the current regional server

(ii) Using the branch code of the destination branch A2, find out the region to which the account belongs. Let the region code be R2 and branch code isB2.

(iii) Initiate a tuple_retrieval_request command to the main server.

(iv) The main server fetches the current value of R2.B2 which will be updated as:

UPDATE B set bal = bal- x where ac_no="A1";

UPDATE R2.B2 set bal = bal + x where ac_no="A2";

5. MATHEMATICAL MODEL

5.1 Existing system

In the existing system, the main server has to be accessed in case of both inter-regional and intra-regional transactions. The access overhead in the main server due to the transactions has to be computed for both intra-regional and inter-regional transactions. Note that there are no local servers in each region to store the transaction data pertaining to that region.

Case 1: For intra-regional transactions

A single intra-regional transaction needs 'm' main server accesses.

If there are t_{intra} intra-regional transactions totally,

The total number of main server accesses = t_{intra}*m --- [1]

Here an access denotes an atomic transaction involving data in the access server.

Case 2: For inter-regional transactions

An inter-regional transaction involving 'm' branches from different regions requires 'm' main server accesses.

If there are t_{inter} inter-regional transactions totally,

The total number of main server accesses = t_{inter}*m --- [2]

Summing up the intra-regional and inter-regional transactions in the bank,

The access overhead of the main server will be = t_{intra}*m + t_{inter}*m

$$=m (t_{intra} + t_{inter}) \text{ ---------- [3]}$$

5.2 Proposed system:

In the proposed system, there are local servers located at each regional office. The data and the transactions of the region are stored in the regional server.

The access overhead of the main server in case of both intra-regional and inter-regional transactions is computed as follows:

Case 1: For intra-regional transactions

The local server located at each region will have transactional databases of each and every branch under its control. A copy of the same data will be maintained in the main server.

However, banking software need not access the main server for the processing of an intra-regional transaction.

So, the total number of main server accesses = 0 --------- [4]

Case 2: For inter-regional transactions

If an inter-regional transaction involves 'm' branches across different regions, the data in the regional server of the branch initiating the transaction is not sufficient. Further, it does not have the knowledge of the data and the transactions stored in other regional servers. The system does not allow local server – local server communication. So the main server access is mandatory in this scenario.

So for a single inter-regional transaction to be processed, 'm' tables in the main server have to be processed.

Hence if there are t_{inter} inter-regional transactions totally,

The total number of main server accesses = t_{inter}*m --- [5]

Summing up the intra-regional and inter-regional transactions in the bank,

The access overhead of the main server will be = 0 + t_{inter}*m

$$= t_{inter} * m \text{ ---------- [6]}$$

5.3 Increase in efficiency:

The decrease in the number of access overhead values of the main server in the existing system and the proposed system, 'd' is to be calculated.

From equations [3] and [6],

$$d= m (t_{intra} + t_{inter}) - t_{inter} * m = t_{intra}*m$$

$$\% \text{ increase in efficiency} = \frac{d}{\text{number of access overhead values of the main server in the existing system}} * 100$$

$$= [(t_{intra}*m) / m (t_{intra} + t_{inter})] * 100$$

$$= [t_{intra} / (t_{intra} + t_{inter})] * 100 \text{ ---[7]}$$

In most of the cases, it is likely that m takes a value of 2.

In the following situations, the efficiency will be increased:

1. Whenever t_{intra} is higher and t_{inter} value is lower
2. Difference between Numerator and denominator in equation [7] is as low as possible

6. EXPERIMENTAL SETUP AND RESULTS

For various values of t_{intra} and $t_{inter,}$ the number of main server accesses is computed and hence the % reduction in transaction overhead is calculated as shown in the table below:

Table 1: Test cases for main server accesses

TEST CASES FOR MAIN SERVER ACCESSES				
Inter-regional **Transactions**	**Intra-regional transactions**	**Main Server access before localization**	**Main Server access after localization**	**% reduction**
75	25	100	75	25
65	35	100	65	35
55	45	100	55	45
45	55	100	45	55
35	65	100	35	65
25	75	100	25	75
15	85	100	15	85

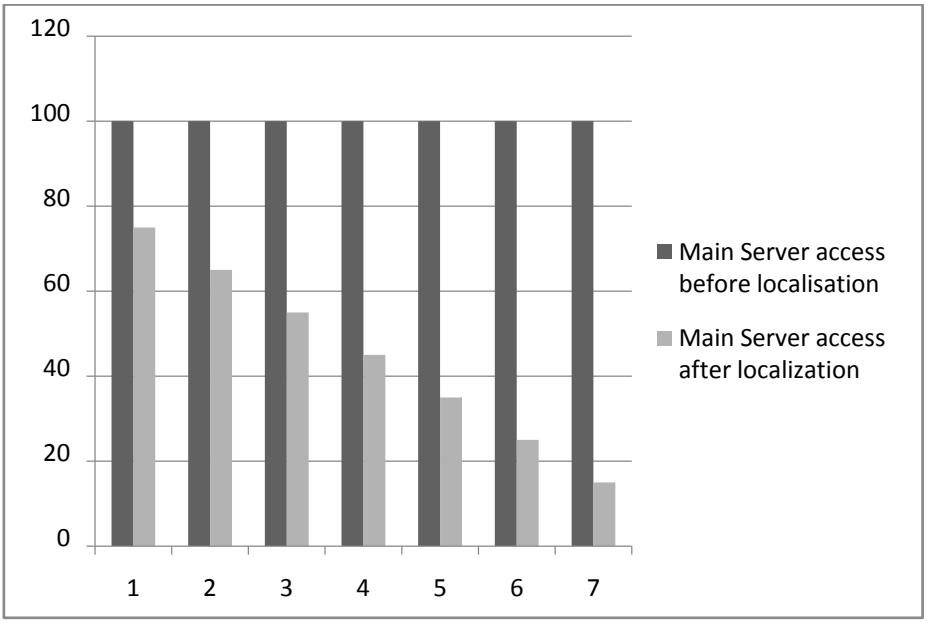

Fig 3: Count of Main Server Access for various test case values

A graph is plotted based upon the above test cases and the results obtained. The following points are evident from the graph:

1. Irrespective of the presence or absence of regional servers in the banking system, the banking software deployed in each branch accesses the main server for the inter–regional transactions.
2. The localization of server decreases the number of accesses in the centralized server and thereby reduces the transaction overhead in the main server.
3. Localization of servers renders the use of replication server as optional, which in the case of existing system is mandatory.

7. DISCUSSION AND SCOPE FOR FUTURE WORK

The advantages of the proposed system over the existing system include: Foremost, the overhead in accessing the main server is reduced. Since only the inter-regional transactions needs to access the main server, the number of transactions by which the overhead gets reduced is equal to the number of intra-regional transactions. Secondly, data integrity is not lost as the databases are duly updated after every transaction. Moreover, the backup of original data exists i.e., the main server database is updated at the end of each day and that constitutes the integration of all local databases thus alleviating the need for a duplication server. In case of failure of a local server, the previously updated database is obtained from the main server and the transactions are executed again. One point to note is that though the installation of the servers

locally is expensive, it is a one-time investment. Here the trade-off is between efficiency in network bandwidth utilisation and the cost of server localization.

8. CONCLUSION

The existing banking system in India uses a centralized server where the transactional data of all the branches are maintained. The access overhead on the main server becomes very high. Also whenever there are multiple transactions accessing the data on the server simultaneously, the waiting time of the transactions in the queue maintained at the server also increases. So the idea of localisation of servers is proposed where the banking software in the branch initiating the transaction accesses the regional server for intra-regional transactions. But for inter-regional transactions, the branch accesses the regional server via the main server. So the server trafficking is considerably reduced. The theoretical ideas presented in this article have been applied to real-world data modelling. But the trade-off between efficiency in network bandwidth utilisation and cost of server localisation has to be considered.

9. REFERENCES

[1] Srivastava, A., Shankar, U. and Tiwari, S.K., 2012. Transaction Management in Homogenous Distributed Real-Time Replicated Database Systems. *International Journal of Advanced Research in Computer Science and Software Engineering*, *2*(6), pp.190-196.

[2] Corbett, J.C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J.J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P. and Hsieh, W., 2013. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, *31*(3), p.8.

[3] Elmasri, R., 2008. *Fundamentals of database systems*. Pearson Education India.

[4] Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., & Price, T. G. (1979, May). Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data* (pp. 23-34). ACM.

[5] Astrahan, M. M., Blasgen, M. W., Chamberlin, D. D., Eswaran, K. P., Gray, J. N., Griffiths, P. P., ... & Putzolu, G. R. (1976). System R: relational approach to database management. *ACM Transactions on Database Systems (TODS)*, *1*(2), 97-137.

[6] Haag, S., Cummings, M., & Dawkins, J. (1998). Management information systems. *Multimedia systems*, *279*, 280-297.

[7] A book - Laudon, K. C., & Laudon, J. P. (2000). *Management information systems* (Vol. 6). Upper Saddle River, NJ: Prentice Hall.

[8] Moorman, C., & Miner, A. S. (1998). Organizational improvisation and organizational memory. *Academy of management review*, *23*(4), 698-723.

[9] Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases* (Vol. 8). Reading: Addison-Wesley.

[10] Lenzerini, M. (2002, June). Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (pp. 233-246). ACM.

[11] Özsu, M. T., & Valduriez, P. (2011). *Principles of distributed database systems*. Springer Science & Business Media.

[12] Sheth, A. P., & Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, *22*(3), 183-236.

[13] Graefe, G. (1993). Query evaluation techniques for large databases. *ACM Computing Surveys (CSUR)*, *25*(2), 73-169.

[14] Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., DeWitt, D. J., & Naughton, J. F. (1999, September). Relational databases for querying XML documents: Limitations and opportunities. In *Proceedings of the 25th International Conference on Very Large Data Bases* (pp. 302-314). Morgan Kaufmann Publishers Inc.

[15] Kossmann, D. (2000). The state of the art in distributed query processing. *ACM Computing Surveys (CSUR)*, *32*(4), 422-469

[16] Carey, M. J., Ceri, S., Bernstein, P., Dayal, U., Faloutsos, C., Freytag, J. C., & Valduriez, P. (2006). Data-Centric Systems and Applications.

[17] Wolfson, O., Jajodia, S., & Huang, Y. (1997). An adaptive data replication algorithm. *ACM Transactions on Database Systems (TODS)*, *22*(2), 255-314.

[18] Weikum, G., & Schek, H. J. (1992). Concepts and applications of multilevel transactions and open nested transactions.

[19] Valduriez, P. (1993). Parallel database systems: open problems and new issues. *Distributed and parallel Databases*, *1*(2), 137-165.

[20] Clement, T. Y., & Meng, W. (1998). Principles of database query processing for advanced applications.

[21] Taniar, D., Leung, C. H., Rahayu, W., & Goel, S. (2008). *High performance parallel database processing and grid databases* (Vol. 67). John Wiley & Sons.