ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

Load Balancing in Distributed Database System using Resource Allocation Approach

S. Jagannatha[1], D.E. Geetha[2], T.V. Suresh Kumar[3], K. Rajani Kanth[4]

Associate Professor, Computer Applications, MS Ramaiah Institute of Technology, Banagalore, INDIA[1]

Associate Professor, Computer Applications, MS Ramaiah Institute of Technology, Banagalore, INDIA[2]

Professor, Computer Applications, MS Ramaiah Institute of Technology, Banagalore, INDIA[3]

Professor, Computer Applications, MS Ramaiah Institute of Technology, Banagalore, INDIA[4]

ABSTRACT: Distributed database applications needs to replicate data to improve data availability and query response time. Performance is improved because the fragment replica is stored at the nodes where they are frequently needed. Load balancing by proper allocation of transaction, and replicas by sharing of resources for performance analysis is important consideration during early stages. In this paper it is proposed a methodology for performance analysis of load balancing by sharing of resources, allocation of fragment replicas and transaction in distributed database system. The proposed methodology is simulated and results are validated using case study.

I. INTRODUCTION

Distributed Database is spread across the network of computers where application needs. There are two major phase database design: fragmentation and allocation. Dividing the Global schema into a set of non overlapping portion called fragments. Horizontal, vertical, and mixed fragmentation is performed based on application requirements. Allocate these fragments into nodes where the application needs. Performance can be improved for the transaction processing locally rather than remote node. Availability and performance is important non functional requirement should be consider during early design stages. Replica improves the performance and availability by concurrent processing for read only transaction. Determine the number of replica is important consideration for assessing the performance and availability. Replica helps to reduce the waiting time; as a result better performance can be achieved. During data gather stage it is necessary to identity the required number of fragment replica in order to meet performance objective.

In replication control methods, the objective is to provide a high degree of concurrency and thus faster average response time without violating data consistency. Both aborting and blocking mechanism are used for concurrent access transaction and update of database. The number of replicas are depending the query allocation in resources in work load sharing. We assume that all replicas are stored in the same server. During the updating of transaction, it waits for the entire replica available and holds all the transaction until it finishes updates.

The remaining part of this paper is organized as follows. In Section 2 we proposed to study the related work. In Section 3 we proposed methodology and Analysis model using UML 2.0. We proposed and algorithm in section 4. In section 5 we presented the validity of our algorithm using case study. Finally a conclusion and future work is given in Section 6.

II. RELATED WORK

Many researches made significant contribution of replication and load balancing issues in distributed database system. In IV.A.[1] this paper, author address the performance comparison of database replication techniques based on total order broadcast technique based on classical replication scheme like distributed locking. Performance has little influence in a LAN setting; total order broadcast-based techniques are very promising as they minimize synchronization between replicas. In IV.A.[2] author describes design and analyze the data replication strategies with the model of Dynamic Window Mechanism algorithm jointly implemented with different types of object replacement strategies with limited buffer

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

capacities. In IV.A.[3] author present concurrency control in replicated real-time systems using Static Two Phase Locking for deadlock free environment. It also decreases the waiting time of transactions in wait queue. Author provides framework designed to emphasize the role played by different mechanisms and to facilitate comparisons. The framework is to provide all replicated real-time database and maintain all timing constrains and general outline that consider improving the scalability by using segmentation algorithm applied on the whole database, with the intent to lower the degree of replication, purpose of avoiding excessive resource usage, solving the scalability problem IV.A.[4]. In IV.A.[5] author presented a new mechanism on how to handle the fragmented database replication through the Binary Vote Assignment on Grid Quorum and it also address how to build reliable system by using the proposed BVAGQ for distributed database fragmentation. Author describes in IV.A.[6] data replication and maintains consistency in mobile computing environment. He propose the solid infrastructure for distributing replicas and updates propagation protocol to propagate recent updates between the components of the replication architecture in a manner that achieves the consistency of data. The proposed replication strategy is compared with a baseline replication strategy and shown that it achieves updates propagation delay reduction, less communication cost, and load balance as important requirements for maintaining consistency in large scale environments with large number of replicas and highly mobile users. In IV.A.[7]authors surveys performance models for distributed and replicated database systems. Author classify the different alternatives and modeling assumptions, and discuss their interdependencies and expressiveness for the representation of distributed databases. He used proven modeling concepts and gives an example how to compose a balanced analytical model of a replicated database. In IV.A.[8] author develop a consistent mutable replication extension to meet the rigorous demands of large scale data sharing in global collaborations and uses a hierarchical replication control protocol that dynamically elects a primary server at various granularities. Experimental evaluation indicates a substantial performance advantage over a single server system. With the introduction of the hierarchical replication control, the overhead of replication is negligible even, when applications mostly write and replication servers are widely distributed. In IV.A.[9] address this problem with the GORDA architecture and programming interface (GAPI) that enables different replication strategies to be implemented once and deployed in multiple DBMSs. They propose a reflective model of transaction processing. Then we implement and evaluate the proposed architecture in three representative DBMS architectures. Author present in IV.A.[10] replication-based data availability mechanism designed for a large-scale cluster file system prototype named LionFS. Replica updates, relaxed consistency model to enable concurrent updating all replicas for a mutation operation, greatly reducing the latency of operations. LionFS ensures replica consistency if applications use file locks to synchronize the concurrent conflict mutations. Recovery of replica consistency needs not stop the file system services and running applications. Performance evaluation shows that our solution achieves 50-70% higher write performance than serial replica updates. In IV.A.[11] author used the concept of deferred updates propagation enhances update performance and reduces transaction response time for software-based replication in distributed databases. They investigate the replication models and propose one vector-based atomic broadcast protocol for deferred updates, which is superior to the global sequencer-based atomic broadcast protocol in robustness and efficiency. In IV.A.[12] author introduces a model for evaluating the performance of data allocation and replication algorithms in distributed databases. Their limits on the minimum number of copies of the object because of communication cost and I introduce an algorithm for automatic dynamic allocation of replicas to processors. They compare the performance of the traditional read-one write-all static allocation algorithm to the performance of the dynamic allocation algorithm and it shows that communication cost and I/O cost for which static allocation is superior to dynamic allocation. In IV.A.[17] this paper surveys performance models of replicated distributed database. The model uses the concepts of replication, communication, non uniform data access using proven modeling concepts. Finally, author compares the analytical results to measurements in a distributed database system

III. METHODOLOGY

The performance of Distributed database is strongly related to the concurrent execution, fragment replication, allocation transaction, load sharing of resources in the nodes of computer network. A multiple query of different

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

application tries to access same piece of data in the same server. Data replicas are stored in the local servers. Since the data replicas are stored in the same server, update of replicas is done using lock and release method. Number of replica are depend on the number of query tries to access the same piece of data in the same server. The Cost of computing is based on Load balancing, concurrent execution by resource allocation and reduces the waiting time of the query by placing the replicas in the server. The goal is to load sharing by use of replicas by computing resource usage cost for transaction processing. The cost is to minimize by proper allocation of resources in load sharing, data availability in distributed environment.

A. Proposed Methodology

Suppose that n be the number of use cases share m resources. Assuming that application is model using use case diagram using UML. Each use case U_i consisting of k parallel and dependent sub queries, some queries are assigned same resources for load sharing and replica available. For example: use case with-draw has withdrawn, checks balance, overdraw, and are parallel sub queries. These tasks are sharing R_j resources and share the R_j capacity and expense P_j according to its processing capacity. These tasks are allocating R_j resources for load sharing. Each sub query may have to share R_j and may be waiting to share R_j. The assigning these queries into these resources based on the data replica available. Number of replica based on the query waiting for resources. A solution of the scheduling problem is a non-negative matrix i.e. query allocation matrix a of n rows one for each use case and m columns one for each resource. The entry a_{ij} is the allocation of query in the U_i use case to the R_j resources based data availability.

The final allocation matrix is obtained based on allocation of monitor server. It monitors the load between and the replicas of the corresponding server. It is called as the dispatcher. Dispatcher determines the query to be assigned to which replica and responsible for balancing the load. The selection of resource and data replica availability by estimation of weight of the resource. The weight is obtained by average service time for the serviced the query in each resource and number of requests in the queue. This estimation is approximate time required t_i process the query. The weight if the i^{th} resource is defined as

$$W_{Ri} = AST_{Ri} + Q_{Ri,t} \quad (1)$$

Where AST_{Ri} average service time of the i^{th} resource

$Q_{Ri,t}$ number of requests waiting in the queue of the i^{th} replica at time t

The dispatcher of the server estimates the weight of the resource in two steps: i) when query arrives at the server, the dispatcher send the message to all the resources to collect the current status of the weight. Based on the weight of the resources, transaction is assigned the resource which is having least weight. The allocation matrix and cost of computation is estimated based on the resource allocation in transaction processing system for validation of our result. Using allocation matrix another two matrices are obtained: Completion time matrix T, and final execution matrix E. Let t_{ij} of T be the turnaround time it takes resource R_j to complete a_{ij} queries of the use case U_i. U_i queries are parallel and dependent, the completion time of use case U_i is given by $\max \{t_{ij} \mid t_{ij} \in t_i\}$, where t_i denotes the vector of the i^{th} row of matrix T. The entry E_{ij} of the matrix E is the network cost for using resource Rj to complete a_{ij} queries. So the total cost of usage of U_i is $\sum_{j=1}^{m} eij$. In general there is a tradeoff between completion time and cost of usage of resources of each use case U_i. Let w_t and w_e be the weight of completion time and network cost respectively. Therefore the

Total cost of $Ui = wt \cdot maxtij \in ti\{tij\} + we \cdot \sum j\, eij$

$1/Ui$ Denote the utility of use case U_i.

IV. ALGORITHM

Step 1: The key scenarios of the software system are identified.

Step 2: The use cases representing the key scenarios are identified to develop the use case model using UML.

Step 3: Let m be the no of resources(R) are used in the proposed architecture.

Step 4: Identify the subtask in each use case share m resources (nodes)

Step 5: Each resource R_j has fixed price P_j according to the capacity..

Step 6: Identify the vector p = (p1,p2, ..,pm) satisfies p1 < p2 < · · · < pm, and the corresponding execution time of any subtask of an arbitrary task Si satisfies ˆti1 > ˆti2 > · · > ˆtim.

Step 7: Identify the multiple sub queries in each use case, they share Rj capacity and expense.

Step 8: The dispatcher of the server estimates the weight of the resource in two steps:

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

i) When query arrives at the server, the dispatcher sends the message to all the resources and collects the current status of the weight. Weight is computed using the equation (1)

ii) Based on the weight of the resources, transaction is assigned the resource which is having least weight

Step 9: Let ˆtij denote the execution time it takes for resource Rj solely to complete one subtask of a use case Ui without consideration of communication time.

Step 10: Let ai represents the amount of sub query allocated to the resource Rj.

Step 11: Derived from a another two n x m matrix are obtained

i.) Compute completion time matrix T. The entry tij of T is the turnaround time it takes for Resource Rj to Complete aij sub queries of a use case Ui i.e. tij = ∑j aij * ˆtij where ˆtij execution matrix

ii) Expense matrix E. The entry eij of E is the expense Ui pays for Resource Rj to Complete aij sub queries, i.e. eij = aij * ˆtij * pj when only one sub task assigns to one resource

Step 12: Let wt and we denote the weights of completion time and expense, respectively.

We assumed wt = we = 0.5;

Step 13: Therefore, the total "cost" of a query is

Ui is wt • maxtij * ti {tij} + we *∑j eij.

Step 14: Compute utility of task

ui (ai) = 1/ (wt * maxtij * ti {tij} + we *∑j eij.)

Step 15: Repeat step 9 through 12 for a different allocation strategy until it reaches the optimum allocation i.e. least cost by predicts the performance.

V . ILLUSTRATED THE PROBLEM MODEL WITH CASE STUDY

In this section we proposed a case study, Airline reservation application IV.A.[15]IV.A.[16] that are highly distributed in nature for apply the proposed methodology, to illustrate and validate. The prediction of performance for this problem is discussed using different approach in IV.A.[15]IV.A.[16]. The required data are fragmented and distributed in various nodes of the system. Description of the Case Study:

Nachtfliegen airline system has major functions: Flight Booking, Login, Abandon, Get Flight planning, Find Flights, Select Seat, Get Fare, Purchase Itinerary, and Store Itinerary, The database is fragmented and deployed in a given architecture of 8 nodes. All servers connected by a LAN. The details can be found in IV.A.[16] the typical requests from the users for the application may be authenticating the user, getting the page of the application, searching for the appropriate flights, selecting the desired seat, purchase the ticket, store the details about the flight for later reference, and abandon the Itinerary. The use cases have taken in IV.A.[16], actors identified for the scenarios are presented in the use case model in Figure 1.

A. Estimation of the cost matrix of a given use cases.

In this section, we illustrate the proposed methodology for the case study. Our objective is to estimate the cost of the given use case by maximum utilization of servers and minimize the cost by proper allocation strategy. We estimate (i) find allocation strategy (ii) compute turnaround time (iii) compute Expense matrix E (iv) compute total cost of each query (v) Compute utility of resources by each query. Let w_t and w_e denote the weights of completion time and expense, respectively. We assume $w_t = w_e = 0.5$. Optimum allocation strategy is obtained by different runs and using probable execution time matrix.

The use case model given in Figure 1 consists of use cases namely, Login, GetFlightPlanningPage, Plan Itinerary, Get Fares, Select Segment, Select Seat, Purchase Itinerary, Store Itinerary and Abandon(U1 - U7) the actors namely, customer, user and frequent flyer. The terminals of the nodes are connected to their respective computers.

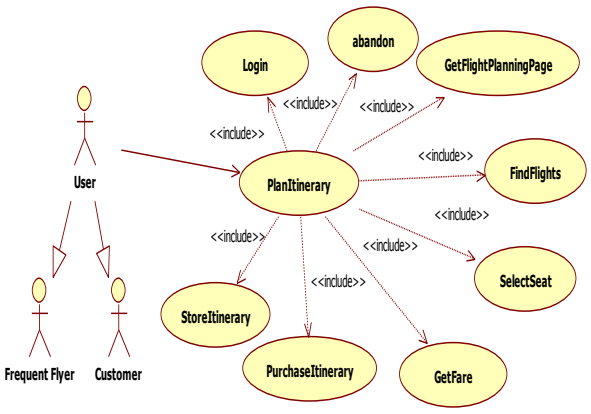

Figure 1 Use case model of the case study

VI. RESULTS AND GRAPHS

If the price vector of each resources(R1 - R8) is ranging from 1 to 10 by taking low level is 1 middle level is 5 and

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

high level is 10, then the execution matrix i.e. final execution time matrix, completion time matrix and expenses matrix is obtained in table 1, table 3 and table 4 respectively. The price vector 2 The available resources (R1 – R8) and their configuration as Intel Core 2 Duo, 3.0 GHz. The description of use cases (U1 – U8) as described above.

Table 1: The execution time matrix

8	7	6	5	4	3	2	1
7	6	5	4	3	2	1	.5
6	5.5	5	4.5	4	3.5	3	2.5
5	4.5	4	3.5	3	2.5	2	1.5
4.5	4	3.5	3	2.5	2	1.5	1.0
4	3.5	3	2.5	2	1.5	1	.5
3.5	3	2.5	2	1.5	1.0	0.5	0.2
3	2.8	2.5	2	1.5	1.2	1.0	.5

Table 2 Allocation matrix

0	1	0	1	1	1	0	0
0	1	1	0	0	1	1	1
1	0	1	1	0	1	1	0
1	0	1	0	1	1	1	1
1	1	0	1	1	0	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	1	1
0	0	1	1	1	1	1	0

The use case U1 chooses { R2, R4, R5, R6,} and U2 chooses {R1, R2, R3, R7, R8} U3 chooses (R1, R3, R4, R7, R8) U4 chooses (R1, R3, R5, R6, R7, R8) U5 chooses (R1, R2, R4, R5, R8) U6 chooses (R1, R2, R4, R5, R8) U7 chooses (R1, R2, R3, R7, R8) and U8 chooses (R3, R4, R5, R6, R7,) as mentioned in the allocation matrix is in table 2. The use cases contains sub tasks share the capacity and expense of allocated resources. The task of each use case share the resources based on the data availability. The data is distributed based on the requirement of the applications. The resources are sharing by parallel tasks. The completion and expense matrix are obtained and the results are shown in table 3 and table 4 respectively.

Table 3 The completion time matrix

0	35	0	25	20	15	0	0
0	30	25	0	0	10	5	2.5
30	0	25	22	0	17.5	15	0
25	0	20	0	15	12.5	10	7.5
22.5	20	0	15	12.5	0	0	5.0
20	17.5	0	12.5	10	0	0	2.5
17.5	15	12.5	0	0	0	2.5	1.0
0	0	12.5	10	7.5	6	5.0	0

Table 4: The expense matrix/network usage matrix

9	16	21	30	0	0	0	0
0	0	0	25	8	3	14	8
0	0	15	0	10	4.5	28	28
6	11	0	22	0	0	21	20
5	9	12	17	6	25	14	12
4.5	8	0	0	5	20	10.5	8
0	0	11	17.5	6.4	3.0	19.6	20
0	0	9	12.5	4.4	2.0	12.6	12

The total cost of each use case and its utilities mentioned in table5. We observed from the shaded part in table 5 that the cost of using resources by the use case U3 is highest compared to others use case and the corresponding utility of resources is least. Hence we conclude that more and more utility of resources effectively the cost of paying is least and vice versa. In use case U6 least cost but the utility of resources is highest not properly utilized. Hence we deduce from the results that more and more usage of resources the expense of network usage is least and we obtain optimum allocation strategy.

Table 5: Utility of a given matrix

Use case	Cost of each Use case	Utility of Resource
U1	35	0.023925
U2	30	0.030257
U3	30	0.023310
U4	25	0.027778
U5	22	0.034118
U6	20	0.040404
U7	17	0.050787
U8	12	0.057637

For illustrative purpose we have taken seven price vector P from 2 to 3.4 Each resources has taken different values which are uniformly distributed with the minimum value 2 average 2.7 and the maximum range is 3.4, the corresponding utility of resources and the cost of each use case mentioned in table 5. The results show the some resources 5, 6, 7, and 8 are used most of the time in computation. The load sharing is uniform in number of tasks are assigned but load is not uniform in computation time. Hence the load balancing is determined in terms of number of tasks and computation time. In figure 6 shows the cost of computation of each use case.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

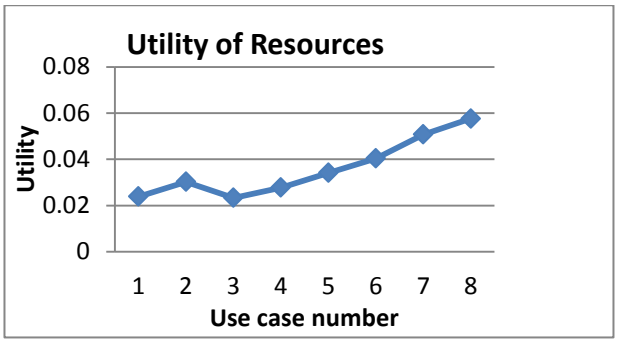

Figure 6: Utility of Resources

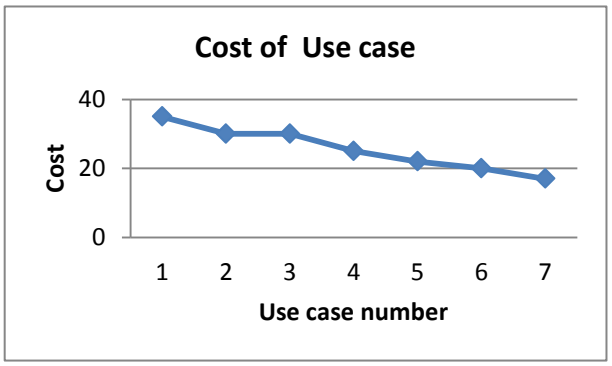

Figure 7: Cost of use case

In the figure 6 shows the Utility of resources of eight use cases. The cost of Use case 1 utilizes 0.023925 Use case 2 utilizes 0.030257 etc. as shown in table 5. The utility of resources in use case 8 is high and least cost with allocation matrix as in graph 2. The results show that the utility of resources increases the cost is proportionally decreases and vice versa as shown in the figure 6 and figure 7. Cost vs. utility, price vector of resources are represented in figure 8

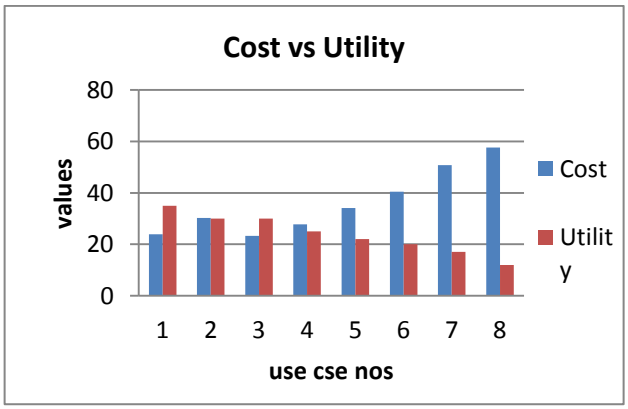

Figure 8: Cost Vs Utility of resources

VII CONCLUSION

In this paper we propose load balancing strategy by allocation of resources during early stages of SDLC. The cost of allocation strategy of each resource is computed and the results are obtained. We can determine the optimum load balancing strategy by knowing the uniform cost of the resources sharing in computation. We propose an algorithm for load balancing resource allocation strategy that optimum utilization of resources and minimize the cost of computation and load sharing based on the number of tasks and not the duration of the task. Data fragment replication is allocated into these set resources in the proposed architecture for data usage. The fragment allocation, resource allocation strategy, load balancing is considered in minimize the cost computation and uniform workload distribution.

REFERENCES

[1] Matthias Wiesmann and Andre´ Schiper, Member, IEEE Comparison of Database Replication Techniques Based on Total Order Broadcast, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 17, NO. 4, APRIL 2005

[2] Xin Gu, Wujuan Lin, and Bharadwaj Veeravalli, Member, IEEE Practically Realizable Efficient Data Allocation and Replication Strategies for Distributed Databases with Buffer Constraints IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 17, NO. 9, SEPTEMBER 2006

[3] Vishal Pathak, Ajay Pratap Rabin Kr. SinghAbhishek Kr. SinghCIRS: A State-Conscious Concurrency ControlProtocol for Replicated Real-Time Databases Vishal Pathak et al. / International Journal on Computer Science and Engineering (IJCSE) ISSN : 0975-3397 Vol. 3 No. 1 Jan 2011

[4] SANJAY KUMAR TIWARI, A .K. SHARMA, VISHNU SWAROOP DISTRIBUTED REAL TIME REPLICATED DATABASE: Concept and Design Sanjay Kumar Tiwari et al. / International Journal of Engineering Science and Technology (IJEST) ISSN : 0975-5462 Vol. 3 No. 6 June 2011.

[5] Ainul Azila Che Fauzi, A. Noraziah, Noriyani Mohd Zain, A.H. Beg "Handling Fragmented Database Replication through Binary" Vote Assignment Grid Quorum Journal of Computer Science 7 (9): 1338-1342, 2011, ISSN 1549-3636 © 2011 Science Publications,

[6] ASHRAF AHMED, DANAPAL DURAI DOMINIC*, AZWEEN ABDULLAH, A NOVEL REPLICATION STRATEGY FOR LARGE-SCALE MOBILE DISTRIBUTED DATABASE SYSTEMS Journal of Engineering Science and Technology Vol. 6, No. 3 (2011) 268 - 299 © School of Engineering, Taylor's University

[7] Matthias Nicola* Matthias Jarke (Senior Member, IEEE)* Performance Modeling of Distributed and Replicated Databases

[8] Jiaying Zhang and Peter Honeyman Hierarchical Replication Control in a Global File System Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07) on Cluster Computing and the Grid(CCGrid'07) 0-7695-2833-3/07 $20.00 © 2007

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

[9] Alfrânio Correia, Jr.U. Minho, José Pereira, U. Minho, Luís Rodrigues, U. Lisboa, Nuno Carvalho, U. Lisboa, Ricardo Vilaça, U. Minho GORDA: An Open Architecture for Database Replication Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007) 0-7695-2922-4/07 $25.00 © 2007

[10] Jin Xiong, Jianyu Li, Rongfeng Tang, Yiming Hu "Improving Data Availability for a Cluster File System through Replication 978-1-4244-1694-3/08/$25.00 ©2008 IEEE.

[11] Xiangzhong Xu, Jingye Wang, and Lijun Pan A Vector-based Atomic Broadcast Protocol for Deferred Updates in Replicated Distributed Databases 978-1-4244-1787-2/08/$25.00_c 2008 IEEE

[12] Ouri Wolfson, Member, IEEE Computer Society, and Yixiu Huang Competitive Analysis of Caching in Distributed Databases IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 9, NO. 4, APRIL 1998 391

[13] ISHFAQ AHMAD, YU-KWONG KWOK, SIU-KAI SO Evolutionary Algorithms for Allocating Data in Distributed Database Systems Distributed and Parallel Databases, 11, 5–32, 2002

[14] Leon ¸Tâmbulea, Manuela Horvat-Petrescu Redistributing Fragments into a Distributed Database Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. III (2008), No. 4, pp. 384-394

[15] Evangelin Geetha, D., Suresh Kumar, T. V., and Rajani Kanth, K.: 'Predicting the Software Performance during Feasibility Study.

[16] Connie, U. Smith, and Lloyd G. Williams: 'Performance Solutions' (Addison-Wesley, 2000).

[17] Matthias Nicola and Matthias Jarke, Senior Member, IEEE Performance Modeling of Distributed and Replicated Databases IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 12, NO. 4, JULY/AUGUST 2000

[18] Danilo Ardagnaa, Sara Casolari b, Michele Colajanni b, Barbara Panicucci a, Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems J. Parallel Distrib. Comput. 72 (2012) 796–808

[19] Javier David Conchaa A tenant-based resource allocation model for scaling Software-as Espadas a, Arturo Molina b, Guillermo Jiménez a, Martín Molina b, Raúl Ramírez a, -a-Service applications over cloud computing infrastructures 0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved. doi:10.1016/j.future.2011.10.013A.