Over Load Detection and Admission Control Policy in DRTDBS

Nuparam[1] and Udai Shanker[2]

[1] Computer Science and Engineering Department, FGIET,
Raebareli, 229001, U.P. India
[2] Computer Science and Engineering Department, MMMEC,
Gorakhpur, 273010, U.P. India
{nrcua80,udaigkp}@gmail.com

Abstract. Today's Real Time System (RTS) is characterized by managing large volume of distributed data making real time distributed data processing a reality [1, 2]. The demand for real time data services is increasing in many large scale distributed real time applications. The transaction work load in DRTDBS may not be balanced and the transaction access pattern may be time varying and skewed. Hence, computation workload on large scale distributed system can lead to large number of transaction deadline misses [3, 4]. Hence, efficient database management algorithm and protocol for accessing and maintaining data are required to satisfy timing constraints of transaction supported applications. In this paper, an algorithm has been proposed for admission control in DRTDBS consisting of local controller and global load balancer working at each site, which decide whether to admit or reject the newly arrived transaction. The simulation results show that the new algorithm successfully balances the workload in DRTDBS [5, 6].

Keywords: Admission control, local controller, global load balancer, distributed data processing, computation workload.

1 Introduction

In recent year, we have seen the emergence of large scale distributed real time database system, which embedded in advance traffic control, factory automation, global environment control and nation-wide electrical power grid control. DRTDBS can relieve the difficulty of developing data intensive real time applications by supporting the logical and temporal consistence of interrelated database distributed over a computer network via transaction management. They support transaction that has explicit timing constraints which is expressed in the form of dead line. A transaction is considered to have finished executing if exactly one of two things occurs: either its primary task is completed (success fully completed) or its recovery block is completed (safe termination). Committed transaction brings a profit to the system whereas a terminated transaction brings no profit. The goal of over load detection and admission control policy employed in the system is to maximize the profit [7, 8, 9]. The presence

V.V. Das, G. Thomas, and F. Lumban Gaol (Eds.): AIM 2011, CCIS 147, pp. 45–54, 2011.

of multiple sites in the distributed environment raises issue that are not present in centralized system. In large scale distributed environment it is a challenging to provide data services with guarantees, while still meeting temporal requirement of transaction. One main difficulty lies in long and highly variable remote data access delays. Large scale distributed real time database system utilizing wide geographical area have to use a network that share by many participant for cost effectiveness. Second major challenge involves the complex interaction among a large number of nodes, which can incur unpredictable work load for each node. Transaction work load fluctuation causes uneven distribution of workload among the sites even if on the average, all sites receive similar amount of workload. The third challenge is the data dependent nature of transaction. End to end transaction access pattern may be time varying and skewed. This can be achieved only by timely access to remote data and timely processed of centralized data.

The system architecture consists of overload detection and admission control of scheduling transaction which provide early notification of failure to submitted transaction that are deemed not valuable or in capable of completing in time, when transaction is submitted to the system. An admission control policy is employed to decide whether to admit or reject that transaction. Once admitted, a transaction is guaranteed to finish executing before its dead line. When any site is detecting overload then it distributes the load to other site. In DRTDBS there are two type of transaction, global and local. The global transactions are distributed real time transaction executed at more than one site whereas the local transaction executes at generation site only.

2 Distributed Real Time Database Model

2.1 Real Time Database Model

We focus our study on medium scale distributed database since the load balancer need full information from every site to make accurate decision. Several applications that required distributed real time data services fall in that range for example a ship board control system which control navigation and surveillance consist of 6 distributed control unit and 2 general control consoles located throughout the platform and linked together via a ship wide redundant Ethernet to share distributed real time data and coordinate the activity [10].

2.2 Distributed Real Time Database Model

The performance of the system is evaluated by developing two simulation models for DRTDBS. The first one is for main memory resident DRTDBS which eliminate the impact of different disk scheduling algorithm on the performance. Since main memory database system, we have also developed another model followed by description of various components such as system model, network model, cohort execution model, database model [11]. In our system model, data object are divided into two type namely temporal data and non temporal data. Temporal data are the sensor data from physical world. Each temporal data object has a validity interval and is updated by periodic sensor update transaction. Non –Temporal data object do not have validity intervals and therefore there are no periodic system updates with them.

3 A New Admission Control Policy Architecture

A database system can be overloaded if many user transactions are executed concurrently. As a result computational resources such as CPU cycle and memory spaces can be exhausted. More ever many transactions can be blocked or aborted and restarted due to data contention also. Fig. 1 shows the architecture of a new admission control policy in DRTDBS. The architecture has 4 layer, remote data access layer, QoS enforcement layer, real time database management layer and DRTDBS layer. In DRTDBS layer does exact work as to admitted or reject the transaction All the transaction are submitted in arrival queue and overload admission controller check the incoming transaction as the system resources is already hold by other transaction.

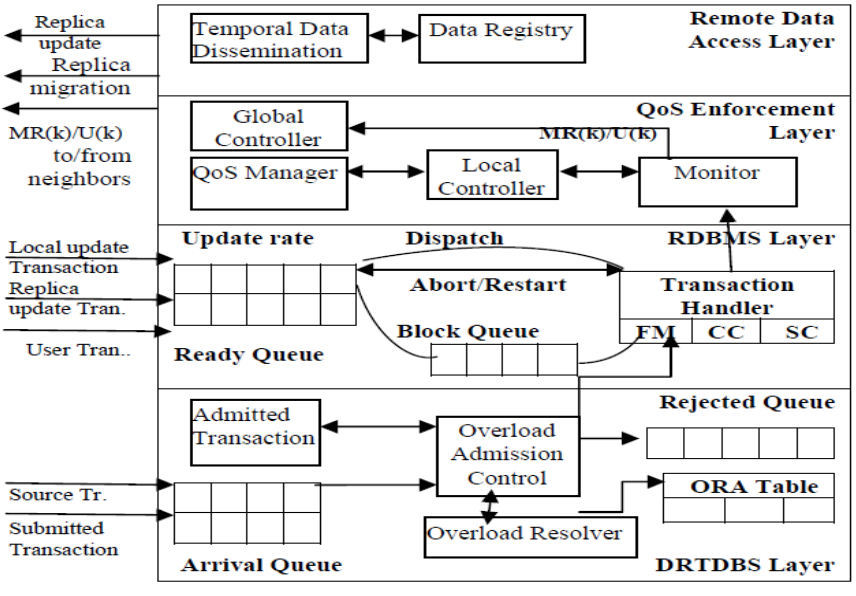

Fig. 1. The System Model for New ACP Protocol

Overload Resolver send current transaction to the ORA (Overload Resolver Array) table. After getting the response from the cohort the overload admission controller interact with the transaction handler (TH) and rejected queue.

The real time database layer does typical real time transaction handling; the incoming transaction are dispatched and processed by transaction handler. The transaction handler consists of a concurrency controller (CC) a freshness manager (FM) and a scheduler (SC). In the SC, update transaction are scheduled in the low priority queue. Update transaction are either updates from local sensors to local data objects. Within each queue, transactions are scheduled with Earliest Deadline First (EDF)[12, 13].

The third layer is guaranteed the desired miss ratio even in the presence of unpredictable workload; QoS enforcement layer exploits two feedback control loops. It has local controller (LC) which operates on the local transaction and global controller (GL) which operate on the global transaction.

The remote data access layer enables transparent access to remote data within a bounded communication time. Remote temporal data are replicated locally to provide timely access to them.

4 Algorithms for Centralized Admission Control in DRTDBS

In each node, there are a local miss ratio controller and local utilization controller. The local miss ratio controller takes the miss ratio from latest sampling period, compare them with serial execution time and compute the local miss ratio control signal δL_{misr} used to adjust the target utilization at the next sampling period. The equation used to derive δL_{misr} is as follows

$$\delta L_{misr} = \sum_{i=0}^{n} \left(I_p^{mr}\right) \times (M_{ri} - M_{rs}) + \sum_{i=0}^{n} \left(I_p^{mr}\right) \times (L_{tmri} - M_{rsi}) \quad \text{---------- (1)}$$

M_{ri} is the miss ratio of class i transaction of last period and L_{tmri} is the long term average miss ratio of class i transaction. M_{rsi} is the specified miss ratio requirement by the serial execution specification. And n is the specified serial execution level.

L_p^{mr} and L_p^{mr} are two controller parameter. In order to prevent under-utilization a utilization feedback loop is added. At each sampling period, the local utilization controller compares the utilization and generates the local utilization control signal δL_{util} using equation.

$$\delta L_{util} = I^{util} \times (L_{util} - L_{utilpset}) + J^{util} \times (LT_{util} - L_{utilpset}) \quad (2)$$

L_{util} is the CPU utilization of last sampling period and LT_{util} is the long term average CPU utilization of the system. $L_{utilpset}$ is the preset CPU utilization threshold. I^{util} & J^{util} are controller parameter.

4.1 Load Balancing Factor

The load sharing process is guided by the load balancing factor (LBF). The LBF at each node is an array of real number which denotes the amount of workload the local node transfer to other node during the next sampling period.

$$T_{remain\ work\ (a)} \geq T_{trsnsfer}\ (a,i,j) \quad (3)$$

The left expression stands for remaining executing (predicted) time of transaction ‘a in the original node and its value is the difference between the transaction ‘average executing time and executing time of transaction ‘a’. The value of right expression stands for the time cost of transferring from node i to node j of transaction ‘a’ which is determined by:

$$T_{transfer}\ (a, i , j) = T_{code}\ (a, i, j) + T_{data}\ (a, i, j) \quad (4)$$

There into, $T_{code}\ (a, i, j) = \text{sizeof\_code(a)}/R$, $T_{data}\ (a, i, j) = \text{sizeof\_data(a)}/R$.

The value of sizeof_code(a) stands for the total size of executing environment parameter and log of transaction ‘a’. The value of sizeof_data(a) stands for the size of fetched data and immediate results by transaction ‘a’ in the current database server node i. R stands for the average network transferring rate.

5 Algorithm for Global Load Balancing in Decentralized DRTDBS

5.1 Work Load Transfer Test

The first step is to test whether there exist load transfer between nodes. To do that we calculate the mean deviation of M_{ri} from different node

$$\text{Mean Deviation} = \sum_{k=1}^{n} \text{ABS}(M_{ri}) - \text{Mean}(M_{ri}) \quad \text{---------- (5)}$$

Where M_{ri} is the Miss ratio of node k. ABS(M_{ri}) returns the absolute value of M_{ri} and Mean(M_{ri}) returns the mean of $M_{ri.}$, n is the nodes in the system. The mean deviation of M_{ri} is a measure for workload balance in the system.

5.2 LBF Adjustment

The LBF adjustment is divided into two cases, depending on whether there is a load transfer among the node.

Load Imbalance -: When there is load transfer in the system i.e the mean deviation of M_{ri} is larger than the threshold, it is necessary to share the load between nodes. The load balancing algorithm at the overloaded nodes will shift some workload to the less loaded nodes. A node i is considered to be overloaded compared to other nodes if and only if the difference between its MRI and MRI mean is larger than the present mean deviation threshold. i.e. as follow

a. When the difference of MRI and MRI Mean (T_1) ≥Mean Deviation Threshold (T_2)
b. When the difference of MRI and MRI Mean < 0
c. 0≤When the difference of MRI and MRI Mean(T_1)≤Mean Deviation Threshold (T_2)

5.3 Algorithm Description

In order to realize the algorithm we introduce three data structures to record the node's states namely R_{queue}, S_{queue}. and $O_{queue.}$ and several other variables such as the maximum value of probing time to avoid too probing effect to the system performance.

5.3.1 Knocking State

A node i is considered overloaded if the difference of MRI and MRI Mean ≥ Mean Deviation Threshold T_2. It divides into two parts: sender side and receiver side. The sender side execute the following operation when a new transaction is created in node i namely T_i (m+1), which cause the node i to be sender.

I. Then select a node j from LBF R_{queue} which satisfies:
$T_{transfer}(a,i,j) = \text{Min}(T_{transfer}(a,i,j))$,
Therein j=1,2,3,4……n. n is the recorded receiver number in structure LBF R_{queue}. It is necessary to judge if the node can be receive in terms of the probing resut.

II. IF (node j is the receiver)
Puts the transaction $T_i(m+1)$ into the node j LFB queue T_j and transfer the log, then the algorithm terminate
ELSE
Remove the node I from LBF R_{queue} and put it into the queue of LBF head of sender queue S_{queue}

III. Select another node from LBF to repeat the process until one of the following condition satisfied. If then terminate the algorithm
III.I R_{queue} is empty
III.II Probing time is beyond the maximum time
III.III Node I is no longer a sender
If receiving probe information then the receiver's side executes the following operation.

IV. Remove the node from the current LBF queue to the head of the S_{queue}
Send a message about node j's states to node i.

5.3.2 Responding State

A node i is considered less overloaded if the difference of MRI and MRI Mean < 0. It also divides into two parts: receiver's side and sender's side, the first part executes the following operation if the node i becomes a receiver when a transaction, $T_{ik,}$ finished or be removed.

I. The node i sends the probing message to node j which is in the head of S_{queue} . And judge if node j is a sender.

II. IF (node j is a sender)
IF (Exists a transaction that can be transferred to node j)
Transfer the transaction T_{jr} which resides in node j before to the transaction LBF queue of node i then executes the T_{jr} from the beginning in the node i. The log will be sent i at the same time. After that, the algorithm terminates.
ELSE (Remove the node j to the tail of S_{queue})
ELSE
Removes the node j from S_{queue} and put into the queue's head of R_{queue} or O_{queue} Select another new node to repeat I and II until one of the following conditions satisfied, if then, the algorithm terminates.
a. S_{queue} is empty
b. The probing time beyond the limit
c. Node i is no longer a receiver.

When a node j receives the probe information from node i, then sender's side execute the following action:

III IF (node j is a sender)
Node j sends a notice message about the node j's states of node i, and evaluate the value of $T_{transfer}(a,i,j)$ by the equation 3 & 4 which is the transferring cost of the transaction. And judge if it is worthy of the transferred.
IF (exists a transaction is worthy of the transferred)
Node j sends a message to indicate there is an available transaction to be transfers the most cost-efficient transaction and corresponding logs. After that the algorithm terminates.

ELSE

Send a message indicating there is no available transaction can be transferred to node i.

ELSE

Remove the node i from the current queue to the queue's head of R_{queue}, and send the current states of node j and i.

5.3.3 Balance State

A node i is considered balanced if the Mean Deviation of MRI is less than the specified threshold, the LBF will reduce the load transferring factors.

$$LBF_{queue(i,j)} = LBF_{queue(i,j)} \times \mu \quad (6)$$

Where $0 < \mu < 1$. μ is called the LBF Regression Factor which regulates the load transferring factors regression process. After reducing LBF, if a LBF becomes sufficiently small(less than 0.005), it is reset to 0.

6 Performance Evaluation

In this section, we show the value of admission control policy by overload detection comparing the performance achievable through workload admission control policy. The transaction inter-arrival rate, which is drawn from an exponential distribution, is varied from 50 transactions per second up to 300 transactions per second in increments of 50, which represents light-to-medium loaded system. Each simulation was run three times, each time with a different seed, for 20000 ms. the results depicted is the average over the three runs. The settings for the user transaction workload are given in table 1. A user transaction consists of operations on both local data object and global data object to 1000 microseconds. At each node, the transaction workload consists of many periodic transactions and the average arrival rate and throughput shown in the simulation result.

Table 1. Baseline Workload Parameter

Parameter	Meaning	Value
CPUTime	CPU time per page access	2.5 ms
DBsize	Database size in pages	1,000
ArrivalRate	Transaction arrival rate	5-100 TPS
CTComp Time	Mean Compensating Task Time	10 ms
CTStdDev	St. Dev. of CT Time	0.5 T CompTime
SlackFactor	Slack Factor	2
RegFactor	Regression Factor	0 to .005
TaskSchd	Task scheduling protocol	EDF
CTSchd	CT scheduling protocol	FF, LF, LMF
Thrsh	CT computation Threshold	0.125
CCntrl	Concurrency Control Protocol	OCC-BC

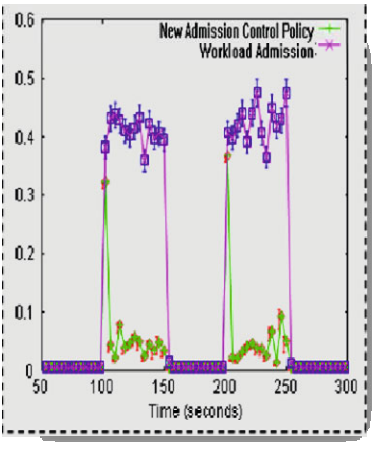

Fig. 2. Mean Deviation throughput

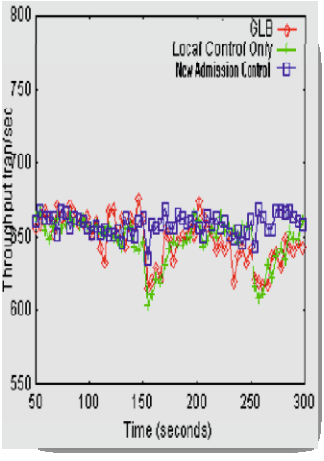

Fig. 3. Global Load Balancer with Throughput

As shown in Fig. 2 the system running best effort algorithm keeps unbalanced throughout the workload burst periods; with admission control policy the system workload become balanced (mean deviation of *MRI* becomes less than 0.1) within 5 seconds. The miss ratios at overloaded nodes are shown in Fig. 3 As we can see, for the best-effort algorithm, QoS requirements are violated and the miss ratio of class i transactions remains over 90%.

7 Related Work

This work differs from previous research in that our transaction model incorporates not only primary tasks, with unknown WCET, but also compensating tasks. The new admission control mechanism used admits transaction into the system with the absolute guarantee that either the primary task will successfully commit or the compensating task safely terminate. Distributed Real Time Database System (DRTDBS) have drawn research attention in recent years [14, 15]. Instead of providing strong logical consistency, DRTDBS focus on data freshness and timeliness of transaction. However most previous DRTDBS work targeted small scale system, but we extended it to large scale system in wide area network environment.

A New Admission Control Policy (NACP) and feedback mechanism could employ in variety of DRTDBS component: Transaction scheduling [15], Memory Allocation for Query Management [16], Concurrency Admission Control Management in ACCORD [8], and an Efficient Call Admission Control for Hard Real Time Communication in Differentiated Service Network [17].The main idea of the admission control policy is to associate an local update to each submitted transaction in order to favors, when the system is overloaded, the executions of the most important transactions according to the application-transactions set. In overload conditions, each sub-transaction of the global

transaction is executed on a site that has the lowest workload among those sites that have executed the data items needed by the sub transaction.

8 Conclusion

Most previous DRTDBS studies have assumed that the only possible outcome of a transaction execution is either the commitment or the abortion of transaction. In many systems a third outcome of an outright rejection may be desirable. A process control application the outright rejection of a transaction may safer than attempting to execute that transaction only to miss its deadline. Out system allows the system to reject a transaction because the admitted transaction holds some resources, thus making it possible utilization by other transaction, to be taken in timely fashion. Also this flexibility allows the system to relate its resources in the most profitable way, by only admitting high value transaction when the system is overloaded while being less choosy when the system is under loaded.

Our current research efforts focus on evaluating the performance of pessimistic as well as speculative CACM techniques. Moreover, our work to date has concentrated on uniprocessor systems. We are currently investigating the extension of our admission control and scheduling protocols to multiprocessor systems. A number of challenging questions arise. How are transactions, both their primary tasks and compensating tasks allocated to processors? What type of CPU scheduling discipline should be used? How valuable is the use of the WACM in a multiprocessor system? How the concurrency will maintained in CACM?.

References

[1] Aldarmi, S.A.: Real Time Database System, Concept and design. Department of computer science, University of York (April 1998)

[2] Kim, Y., Son, S.: Supporting predictability in real time database system. In: Proc. 2nd IEEE Real Time Technology and Application Symposium (RTAS 1996), Boston, pp. 38–48 (1996)

[3] Abbott, R., Garcia-Molina, H.: Scheduling real time transaction: A performance evaluation. In: Proceeding of the 14th International Conference on very large Data Bases, Los Angeles, CA, pp. 1–12 (1988)

[4] Kang, W., Son, S.H., Stankovic, J.A.: Managing deadline miss ratio and sensor data freshness in real time databases. IEEE Transactions on Knowledge and Data Engineering (October 2004)

[5] Stankovic, J.A., He, T., Abdelzaher, T., Marley, M., Tao, G., Son, S., Lu, C.: Feedback Control Scheduling in Distributed Real Time Systems Symposium (RTSS 2001), Washington, DC, USA, p. 59 (2001)

[6] Lam, K.W., Lee, V.C.S., Hung, S.L.: Transaction scheduling in distributed real time system. Int. J. Time – Crit. Comput. Syst. 19, 169–193 (2000)

[7] Lee, V.C.S., Lam, K.-W., Hung, S.L.: Concurrency control for mixed transactions in real-time data-bases. IEEE Trans. Comput. 51(7), 821–834 (2002)

[8] Nagy, S., Bestavros, A.: Concurrency Admission Control Management in ACCORD. Ph.D Thesis at Bostons Univesity (1997)

[9] Bestavros, A., Nagy, S.: Value-congnizant admission control for rtdb systems. In: RTSS 1996 the 17th Real Time System Symposium, Washington DC (December 1996)
[10] Saab Systems Pty Ltd. “Ship Control System” in Saab System Website, http://www.saabsystems.com.au
[11] Shanker, U.: Some performance issues in Distributed Real Time Database Systems. PhD Thesis, Department of Electronics & Computer Engineering, IIT Roorkee (December 2005)
[12] Chetto, H., Chetto, M.: Some results of the earliest deadline scheduling algorithm. IEEE Transaction on Software Engineering (October 1989)
[13] Liu, C.L., Layland, J.: Scheduling algorithms for multiprogramming in hard real time environments. Journal of the Association of Computing Machinery (January 1973)
[14] Kang, W., Son, S.H., Stonkovic, J.A., Amirijo, M.: I/O aware deadline miss ratio management in real time embedded database. In: 28th IEEE Real Time System Symposium (RTSS) (December 2007)
[15] Lee, V.C.S., Lam, K.W., Hang, S.L.: Transaction Scheduling in Distributed Real Time System. Intr. J. Time Crit. Computer System (2000)
[16] Pang, H., Carey, M.J., Livny, M.: Managing memory for real time queries. In: Proceedings of the 1994 ACM SIGMOD Conference on Management of Data, pp. 221–232 (1994)
[17] Baronia, P., Sahoo, A.: An efficient Call Admission Control for Hard Real Time Communication in Differentiated Services Network. Proc. IEEE (2003)