World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:4, No:4, 2010

Effect of Network Communication Overhead on the Performance of Adaptive Speculative Locking Protocol

Waqar Haque, Pai Qi

***Abstract*—**The speculative locking (SL) protocol extends the two-phase locking (2PL) protocol to allow for parallelism among conflicting transactions. The adaptive speculative locking (ASL) protocol provided further enhancements and outperformed SL protocols under most conditions. Neither of these protocols consider the impact of network latency on the performance of the distributed database systems. We have studied the performance of ASL protocol taking into account the communication overhead. The results indicate that though system load can counter network latency, it can still become a bottleneck in many situations. The impact of latency on performance depends on many factors including the system resources. A flexible discrete event simulator was used as the testbed for this study.

***Keywords*—**concurrency control; distributed database systems; speculative locking

I. Introduction

REAL-TIME database systems are an extension of traditional database systems that are designed to process workloads whose state is constantly changing [1]. Such systems are more realistic because data acquiring and updating often occur in parallel in many real-time applications such as e-commerce, stock trading, radar tracking, and mobile computing. Because data in a real-time database system changes rapidly, one of the top priorities in its design is to respond to those changes quickly. This may mean that updates should be immediately available to all subsequent requests for this data. A transaction in a real-time database system usually has a deadline, which makes the system response time as critical as data integrity. As a real-time database system grows larger, more transactions arriving at geographically distributed nodes need to be processed in the same period of time thus degrading system performance. Distributed real-time database system (DRTDBS) attempt to address this problem by either replicating or partitioning the data across multiple sites.

A number of techniques have been proposed to increase the reliability and performance of a DRTDBS. For instance, the most extensively studied area is transaction scheduling which focuses on managing and ordering transactions in a specific way so that the processing becomes optimal. This also extends into sub-fields such as concurrency control, priority control, and deadlock resolution. Other areas include synchronization of replicated data, fault tolerance and failure recovery, which keeps the database in a stable state when unexpected situations occur. A concurrency control protocol is a data-access scheduling policy that preserves the database's integrity by resolving non-serial concurrent executions, in a manner that includes a serialization order among the conflicting transactions [2]. Examples of concurrency control protocols include the widely-used 2PL (two-phase locking) protocol [3] and speculative locking (SL) protocols. Studies have shown that SL protocols yield better performance than 2PL protocol in the distributed database systems [4]. The Adaptive Speculative Locking (ASL) protocol was introduced as an extension of SL protocols; it improves system performance and utilizes less system resources than the proposed SL protocols [5], [6]. More details of the ASL protocol are provided in the next section.

One attribute that could have a significant impact on the performance of a DRTDBS is network communication. It is normal for a DRTDBS to have data on nodes that span a geographically distributed area. The transaction will thus have to acquire this data from several nodes in order to complete the required processing. In such a scenario, the communication overhead cannot be ignored and in fact could become high enough to seriously effect the performance. The results reported from previous experiments using ASL protocol assumed a fully-connected network over which messages transferred instantly. Though unrealistic, this assumption allowed to focus the study on effects of underlying system configuration and concurrency control protocols instead of the communication overhead. In this paper, we study a more realistic situation which gives due consideration to the network communication overhead of DRTDBS while using the ASL protocol for concurrency control. The message transfers are no longer assumed to be instant but take a finite amount of time depending on the network topology.

A distributed real-time transaction processing simulator (DRTTPS) is used to measure the performance of database system in this study. DRTTPS is a full-featured, discrete event, simulation system that is capable of simulating a wide range of protocols within an environment that represents real-world scenarios. The simulator is briefly described in the next section.

W. Haque is Professor, Computer Science Program, University of Northern British Columbia, 3333 University Way, Prince George, BC V2N 4Z9, Canada (Tele: 1-250-960-6522; e-mail: haque@unbc.ca).

P. Qi is a student in the Computer Science Program at University of Northern British Columbia, Canada (e-mail: qipai@qipai.ca).

This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant 18943-98.

II. Literature Review

This section gives a brief overview of DRTTPS and the ASL protocol that is used for our study. We also provide review of other related work that has been done to address the concurrency control issues in real-time database systems.

A. DRTTPS

DRTTPS is a discrete event, highly configurable simulator that provides a distributed environment to simulate a real-time transaction-based database system. The main components of this simulator are a configuration tool, an interactive runtime environment and a reporting tool. It provides a testbed with plug-and-play ease to implement and test the performance of new protocols. These include concurrency control protocols, priority protocols, resource allocation protocols, and network topology protocols. The new protocols can be configured easily without modifying the simulator itself. Different kinds of real-time database simulators have been built for previous studies [7]–[11]. Some are designed for centralized database systems [11]; some lack flexibility of network topology setting [7], [8]; some are only designed to test countable protocols [10]. DRTTPS was designed to provide users a flexible way to deploy different protocols, policies and parameters. A detailed description of the underlying model of the simulator can be found in [12]. In DRTTPS, events are processed in a sequential manner and are driven by the next event. The event clock is measured by a unit called a tick. A tick is a discrete amount of time in which one or more events can be executed. Samples of events in DRTTPS include: sending a message from one node to another, a page being processed, queueing for resources, a transaction arriving at a node, and transaction commit. Such events are inserted into the execution queue of simulator based on the order in which they need to be executed [5]. The network consists of one or more sites with nodes within each site. Each node represents several hardware and software components and consists of processor manager, disk manager, buffer, and (optionally) a workload generator (transaction generator). A node is also characterized by policies that provide the basis for transaction processing, such as concurrency control protocol, preemption protocol, deadlock resolution protocol, and replication protocol. Nodes within a site are connected by a local area network. A site represents a work site in real situation. Sites are connected by wide area networks. Each site contains a virtual router and is responsible for all network connections within this site. All network connections in the simulator are configurable. Attributes that can be set include latency, bandwidth, and external use. The percentage of transactions that complete on time (PCOT) is calculated by dividing the total transactions that have completed before their deadlines by the total number of transactions that have been generated till that moment. In most of the experiments, PCOT has been chosen as a metric to measure system performance. As mentioned before, time variables in DRTTPS, such as disk access time and network latency, are all in the unit of ticks. Using tick as time unit instead of units such as seconds provide a more general meaning to the study. This also makes the results machine-independent unlike those reported by other studies.

B. The ASL and other concurrency control protocols

In comparison with traditional 2PL protocol, SL protocols decrease the time spent to execute a transaction when some of the required data has been locked by other transactions. This is achieved by running multiple speculative executions on combinations of all, or some possible data outcomes after the locks are released; one of these outcomes survives. The Adaptive Speculative Locking (ASL) protocol further extends the SL protocols. While SL restricts the number of speculative executions without enforcing a limitation on the number of previously aborted transactions, ASL provides the enhancement by using techniques such as hyper-threading, memory management and transaction queue management. In previous studies, it was shown that ASL protocol outperforms SL protocols in most situations, especially in systems with higher system loads [5]. However, several assumptions were made in previous experiments. *Protocol Assumptions:* When a transaction is created, the resources required (pages) are known. No rollbacks can occur after a transaction commits. Local deadlocks cannot occur, however global deadlocks can. A transaction has the ability to move any of its held pages into sub-transactions. *System Assumptions:* No hardware failures can occur. The network is fully connected. The network is connected with zero latency. Buffer access is instantaneous. Disks and Swap Disk are distinct resources. [5] Since the networks being tested were always assumed to be fully connected without latency, it left the performance of system using ASL protocol with network communication overhead unknown.

In recent years, various concurrency control protocols for real-time database systems have been proposed. [13] examined the performance of two alternative locking mechanism of the 2PL protocol, namely static locking and dynamic locking, under various system conditions. [14] discussed systems using a nested transaction model. A lock mechanism (2LP-NT-HP) is implemented to solve the data conflict between nested transactions. A timestamp vector based optimistic concurrency control (OCC) protocol is proposed in [15]. It is said to reduce unnecessary transaction restarts and yields better performance compared to traditional time interval based OCC protocols. A role ordering (RO) scheduler that serializes transaction conflicts is discussed in [16]. The study shows that that the RO scheduler outperforms the traditional 2PL protocol. Flexible High Reward (FHR) is another concurrency control protocol which gives a better consideration to the impact of communication delay in transferring data over network than other traditional concurrency control protocols such as earliest deadline, has a lower miss ratio and thus yields a better performance than those protocols [17].

III. Effect of System Parameters on DRTDBS with Varying Network Latency

To study the effect of specific attributes on system performance, most of the control parameters are fixed for each

TABLE I
BASIC PARAMETER SETTINGS FOR MOST EXPERIMENTS

Parameter	Value
Node Count	7
Connection Topology	Tree
Bandwidth	100 units per tick
Concurrency Control Protocol	ASL
Priority Protocol	EDF
Preempt Protocol	EDF
Max Active Transaction Count	30
Disk Count per Node	1
Max Page Count per Disk	100
Disk Access Time	35 ticks
Cache Size	40 pages
Swap Disk Access Time	35 ticks
Page Process Time	15 ticks
Pages per Transaction	7 - 21
Slack Time	720 - 2160 ticks
Inter Arrival Time	100 ticks
Page Update Rate	75 percent
Transaction Count	300

simulation. Parameters used in all experiments, if not mentioned separately, are configured as in Table I. The network topology used for this study is shown in Fig. 1 is used for this set of experiments. Each node is labeled with its ID. All network connections are full-duplex and assumed to have the same bandwidth and latency. Without loss of generality, we assume all nodes within the network to have identical settings. One of the leaf nodes is selected to contain the only transaction generator in the network. For reference, nodes are labeled starting from the root (node 0). A baseline study is first conducted on a simple network topology, to identify the attributes that could have major impact on system performance as network parameters change.

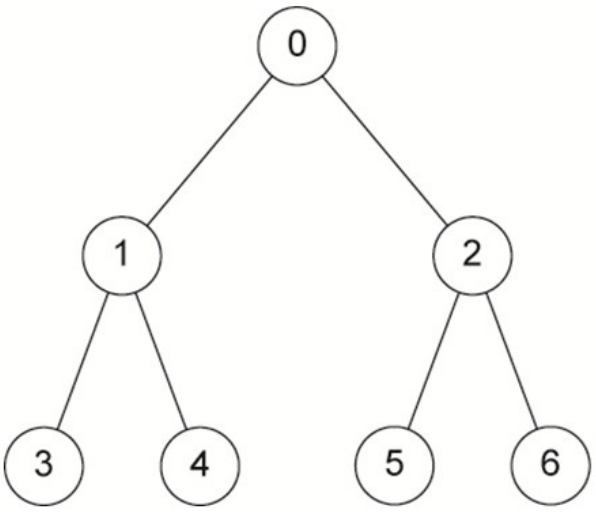

Fig. 1. Network Topology

Experiment A. Effect of cache size under varying network latency

In this experiment, we study the impact of cache size on system performance. The percentage of transactions meeting their deadlines (PCOT) is measured for various cache sizes and network latency. The network latency is specified in ticks. The system performance is shown in Fig. 2. Notice that in systems with low network latency (10 and 35), an increasing cache size clearly contributes to the increase in number of transactions that meet their deadlines. In contrast, in situations with high network latency (greater than 60) increasing cache size does not help improving system performance even when all pages stored in hard disks are cached (that is, cache size of 100). This confirms that high latency systems do not benefit from a boost in cache size.

It is believed that under overload condition, the reason why systems using traditional locking protocols exhibit poor performance in high-latency condition is that the earliest deadline first (EDF) priority protocol progressively increases the average completion time [18], [19]. This is because in an overloaded system, more and more newly generated transactions wait in the queue as time progresses. Thus, the average time need for a transaction to complete will keep increasing until most transactions become tardy, ultimately resulting in a very low number of transactions completing before their deadlines. Other studies have also confirmed that EDF is not the desirable protocol under heavily loaded systems [18], [19]. We chose to use EDF in our study because of its better performance over a wide range of system loads.

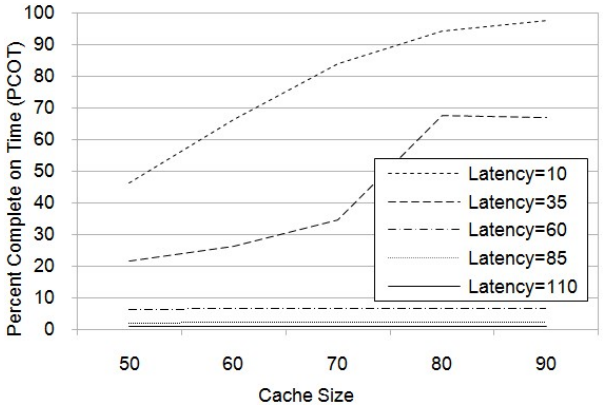

Fig. 2. Effect of cache size under varying network latency

Fig. 3 shows similar results from a different perspective using three different cache sizes. Once again, it is observed that for low network latency, the cache size has a large impact on the system performance. However, when latency becomes large, increasing the cache size does not contribute to performance enhancement. These results are consistent with the observations made above.

Experiment B. Effect of inter-arrival time under varying network latency

In this experiment, we study the effect of inter-arrival time on system performance as the network latency varies. The

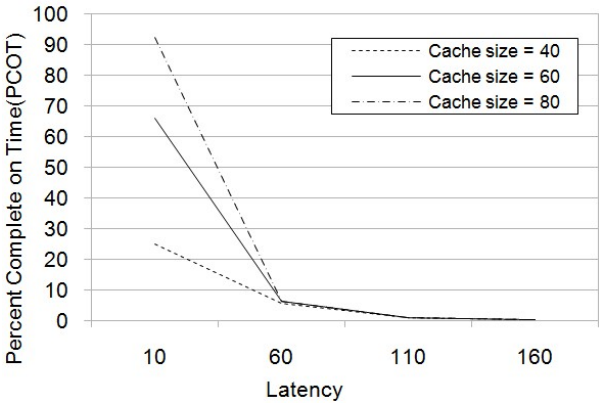

Fig. 3. Effect of Network Latency on Systems with Different Cache Size

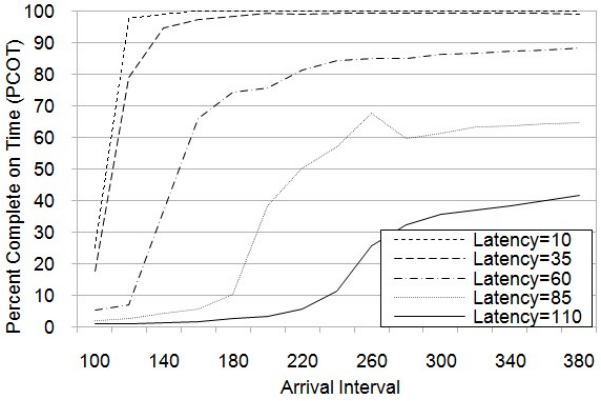

Fig. 5. Effect of inter-arrival time under varying network latency (extended range)

percent of transactions that complete on time (PCOT) is once again used as the performance metric. We vary the inter-arrival time from 100-380 ticks using the same set of network latencies as in the previous experiment. All other parameters are kept fixed as noted in Table I.

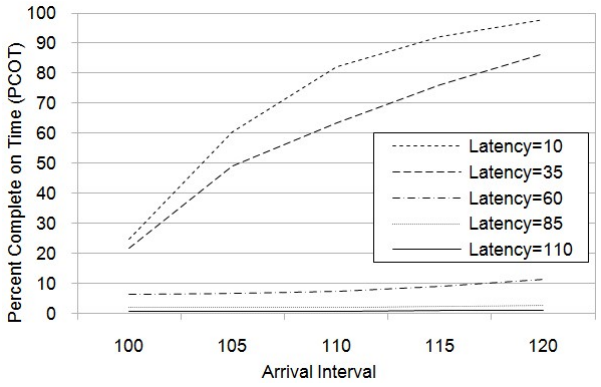

Fig. 4. Effect of inter-arrival time under varying network latency

The results show that in low-network-latency system (latency less than 60), the PCOT increases rapidly as inter-arrival time increases from 100 to 120 ticks (Fig. 4). Although from this figure it seems that a further increase in the inter-arrival time will not provide a better performance in a system with high network latency, Fig. 5 shows a significant performance improvement under all system loads even under high latency conditions. The typical S-shaped performance, demonstrating a sharp increase in performance with less dense arrivals, is observed under all tested latency values. A latency-related inter-arrival time can thus be determined for optimal performance of any system.

Looking at these results from a different perspective, Fig. 6 shows the effect of latency using three different inter-arrival times. At an inter-arrival time of 100, the arrival of transactions becomes so dense that regardless of network latency, the number of transactions that complete on time remains very low. On the other hand, in systems with a longer inter-arrival time, a (mirrored) S-shape trend is observed.

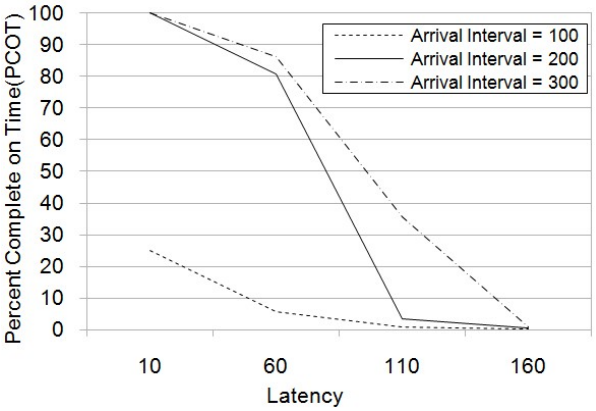

Fig. 6. Effect of Network Latency on Systems with Varying Inter-Arrival Time

Experiment C. Effect of number of disks under varying network latency

This experiment studies whether increasing the number of disks on each node will affect system performance in high latency situation. By adding more disks to nodes, the number of pages that can be read or written simultaneously is also increased. We use up to four disks while keeping the total disk capacity per node unchanged.

Fig. 7 shows a significant performance gain when the number of disks changes from 1 to 2 in low latency situations. For higher latencies, there is no system performance improvement when more disks are added and PCOT remains under 10 percent. The relationship between maximum disk utilization and disk count per node is shown in Fig. 8. For the low latency situation where there are only one or two disks per node, the reason of the dramatic improvement in performance can be attributed to the high disk utilization resulting in queue buildup when a single disk is used. Adding more disks under such situations lowers disk utilization implying some idle time and no waiting in the queues. Beyond a certain point, however, the bottleneck shifts from the disks to the latency and no further improvement is observed with additional disks.

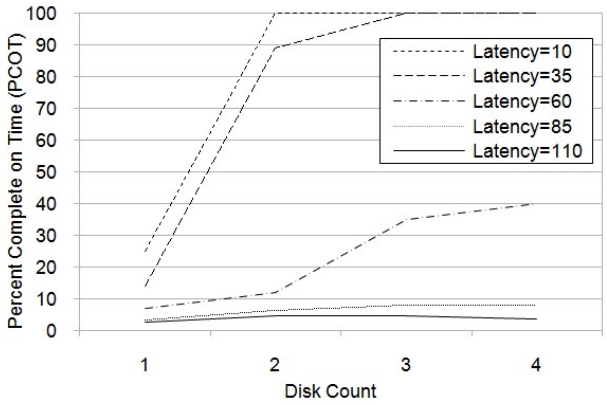

Fig. 7. Effect of disk count on systems with varying network latency

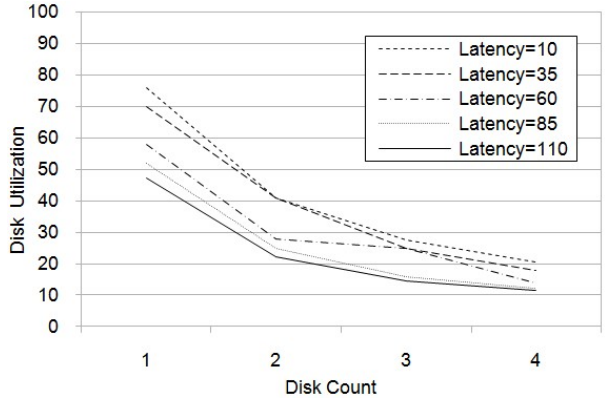

Fig. 8. Effect of disk count on disk utilization under varying network latency

Experiment D. Effect of update rates under varying network latency

The update rate of a transaction has significant impact on performance because it reflects the need for resources. A value of zero implies read-only transactions whereas an update rate of 100 means that every page that is read will also be written back to the disk. In this experiment, we observe how network latency influences transaction completion rate (PCOT) for various update rates. As shown in Fig. 9, systems with a high update rate will have an overall low PCOT. There are two things to be noticed in this figure. First, for low latencies, there is a significant difference on performance in systems with an update rate of 0-50 percent and 75 percent. This indicates that update rate plays a significant role in the performance when latency is low. Second, when the network latency rises above 50, it becomes the dominating factor influencing the performance; the update rate becomes irrelevant beyond this point. In other words, in all systems, the network latency will eventually become the bottleneck of the system.

IV. Summary

In this study, the communication overhead of DRTDBS using ASL protocol is examined using our distributed real-time transaction processing simulator. Distributed real-time database systems with different network latencies are simulated, and the effect of network communication on system performance is analyzed. Though some assumptions were made to model the system, these do not effect the generality of results. Some of the interesting observations are: Network latency has very significant effect on system performance under all conditions. The effect of network latency can be overcome by increasing inter-arrival time, which decreases the transaction density in system. Whether it can be overcome by improving system resources, such as increasing cache size and number of disks, depends on the specific situation for each system. Although disk accesses can be decreased by lowering transaction update rate, it does not always overcome the effect of network latency on system performance.

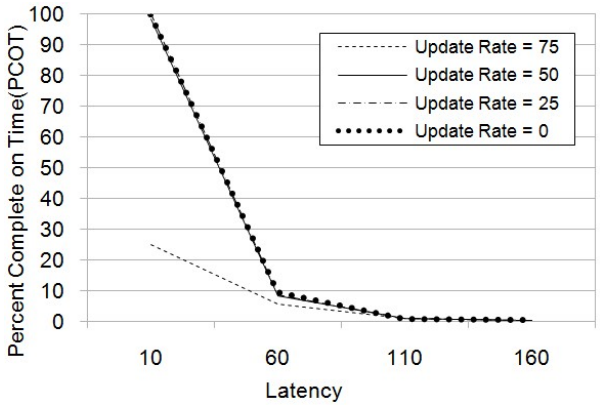

Fig. 9. Effect of page update rate under varying network latency

References

[1] A. Buchmann, "Real time database systems," in *Encyclopedia of Database Technologies and Applications*, L. C. Rivero, J. H. Doorn, and V. E. Ferraggine, Eds. Information Science Reference, 2005.
[2] S. A. Aldarmi, "Real-time database systems: Concepts and design," Master's thesis, The University of York, April 1998.
[3] K. Eswaran, J. Gray, R. Lorie, and I. Traiger, "The notions of consistency and predicate locks in a database system," *Comm. ACM*, vol. 19, no. 11, pp. 624–633, 1976.
[4] P. K. Reddy and M. Kitsuregawa, "Speculative locking protocols to improve performance for distributed database systems," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 16, no. 2, p. 154, February 2004.
[5] P. R. Stokes, "Design and simulation of an adaptive concurrency control protocol for distributed real-time database systems," Master's thesis, University of Northern British Columbia, 2007.
[6] W. Haque and P. R. Stokes, "Adaptive speculative locking protocol for distributed real-time database systems," in *Proceedings of the 19th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2007, pp. 382–390.
[7] H.-R. Chen and Y. H. Chin, "Scheduling value-based nested transactions in distributed real-time database systems," *Real-Time Systems*, vol. 27, pp. 237–269, September 2004.
[8] S. Kim, S. H. Son, and J. A. Stankovic, "Performance evaluation on a real-time database," in *Proceeding of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, September 2002, p. 253.
[9] J. Lindstrom, "Extensions to optimistic concurrency control with time intervals," in *Seventh International Conference on Real-Time Computing Systems and Applications (RTCSA'00)*, 2000, p. 108.
[10] J. R. Haritsa and S. Seshadri, "Real-time index concurrency control," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 12, no. 3, pp. 429–447, May/June 2000.
[11] V. Kanitkar and A. Delis, "Real-time processing in client-server databases," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 269–288, March 2002.

[12] W. Haque, "Simulating concurrency control with deadlock avoidance in real-time transaction processing," *International Journal of Modelling and Simulation*, vol. 27, no. 2, pp. 131–142, 2007.
[13] A. Mittal and S. Dandamudi, "Dynamic versus static locking in real-time parallel database systems," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, April 2004, p. 32.
[14] M. Abdouli, B. Sadeg, and L. Amanton, "Scheduling distributed real-time nested transactions," in *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, May 2005, pp. 208–215.
[15] T. Bai, Y. Liu, and Y. Hu, "Timestamp vector based optimistic concurrency control protocol for real-time databases," in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, Oct 2008, pp. 1–4.
[16] T. Enokido and M. Takizawa, "Concurrency control on distributed objects using role ordering (ro) scheduler," in *Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on*, Feb 2005, pp. 66–73.
[17] H.-R. Chen and Y. H. Chin, "An adaptive scheduler for distributed real-time database systems," *Information Sciences*, vol. 153, no. 1, pp. 55–83, July 2003.
[18] R. Abbott and H. Garcia-Molina, "Scheduling real-time transactions," *ACM SIGMOD Record*, vol. 17, no. 1, pp. 71–81, March 1988.
[19] J. Huang, J. Stankovic, D. Towsley, and K. Ramamritham, "Experimental evaluation of real-time transaction processing," in *Proceedings of the Tenth Real-Time Systems Symposium*, December 1989, pp. 144–153.