9

Performance Study of Shared-Nothing Parallel Transaction Processing Systems

*Jiahong Wang, Jie Li * Hisao Kameda*
Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba Science City, Ibaraki 305, Japan
Tel & Fax: +81-298-53-5521, Email: {wjh, lijie, kameda}@is.tsukuba.ac.jp

Abstract

Processing transactions in parallel brings us a new challenge: how to study the performance behavior of parallel Transaction Processing (TP) systems with dynamic Two-Phase Locking (2PL) concurrency control method analytically. In this paper, an analytic model is proposed for shared-nothing parallel TP systems with dynamic 2PL with the no-waiting policy. In this model, a flow diagram is used for characterizing the activities of transactions, and the steady-state average values of the variables are used for analyzing this flow diagram. Using this model, the performance behavior of shared-nothing parallel TP systems is studied. Analytic results are reported. Simulation experiments are performed to validate the analysis. The predictions of this model agree well with the simulation results.

Keywords

Analytic model, concurrency control, parallel databases, performance evaluation, parallel transaction processing, two-phase locking.

1 INTRODUCTION

In recent years *shared-nothing* parallel Transaction Processing (TP) systems (Stonebrak, 1986) have become increasingly popular for their cost effectiveness, scalability, and availability. Examples are DBC/1012 from Teradata (Teradata, 1983), Non-Stop SQL from Tandem (Tandem, 1987), Gamma at the University of Wisconsin (DeWitt et *al.*, 1986), and Bubba at MCC (Boral et *al.*, 1990). In a shared-nothing parallel TP system, there are numerous processors connected by an interconnection network, each of which accesses its own memory. Records of each relation in database are declustered across disk drives attached directly to each processor. A transaction is divided into subtransactions that are executed in parallel.

Two-Phase Locking (2PL) with the General Waiting (GW) policy is a widely-used Concurrency Control (CC) method in shared-nothing parallel TP systems. In 2PL with GW policy, a transaction

*Corresponding author

Performance and Management of Complex Communication Networks
T. Hasegawa, H. Takagi & Y. Takahashi (Eds.) © 1998 IFIP. Published by Chapman & Hall

is blocked if it requests a data granule locked by another transaction, and the restart is initiated only when there is a deadlock. Blocking transactions, however, has great negative effect on system performance (Franaszek et *al.*, 1985) (Tay et *al.*, 1985a) (Thomasian, 1991). Especially, as the level of data contention increases (e.g., by increasing the multiprogramming level (M), i.e., the total number of transactions running concurrently in a TP system), the interaction between data contention and 2PL with GW policy causes such a snowball effect that the blocked transactions hold locks that they have acquired and result in further blocking (Franaszek et *al.*, 1985) (Tay et *al.*, 1985b) (Thomasian, 1991). As a result, *data contention thrashing* (a sudden degradation in system performance due to the excessive data contention) may occur. For dealing with the negative effect of 2PL with GW policy, so far most of the effort has been devoted to the restart-oriented 2PL CC methods (Franaszek et *al.*, 1985) (Franaszek et *al.*, 1993) (Hsu et *al.*, 1992) (Ryu et *al.*, 1990a) (Tay et *al.*, 1985a). In restart-oriented 2PL CC methods, transactions may be restarted in resolving a lock conflict. A basic restart-oriented 2PL CC method is the 2PL with No-Waiting (NW) policy (Ryu et *al.*, 1990a) (Tay et *al.*, 1985a). In 2PL with NW policy, a transaction is restarted whenever it attempts to lock a data granule held by another transaction.

In this paper, we propose an analytic model for studying the performance behavior of shared-nothing parallel TP systems with dynamic 2PL with NW policy. All the essential factors related with parallel transaction processing, such as the subtransaction initiation, the Two-Phase Commit (2PC) protocol, the degree of system parallelism, and the access skew, are taken into consideration. In this paper, we concentrate on modeling data contention. Note that the solutions of our data contention model can be easily coupled with other standard resource contention models, as done by other studies (Ryu et *al.*, 1990b) (Tay et *al.*, 1985a) (Yu et *al.*, 1993). Simulation is used to validate the analytic model. The prediction of the analytic model agree well with the simulation result.

So far numerous analytic models (Chesnais et *al.*, 1983) (Franaszek et *al.*, 1985) (Tay et *al.*, 1985a) (Thomasian, 1993) (Yu et *al.*, 1993) have been developed for evaluating CC methods for the centralized TP systems. For the parallel TP systems, especially for the shared-nothing parallel TP systems, however, performance evaluation is performed mainly by simulation models or by database testbeds. To the best of our knowledge, no analytic performance model study has been reported that involves the evaluation of 2PL in the parallel TP systems. The lack of analytical studies may be attributed to the complexity of parallel TP systems.

This paper is organized as follows. The model of the system and the notations used in this paper are given in section 2. All the equations necessary for the analysis of this model are derived in section 3. The analysis of shared-nothing parallel TP systems is provided in sections 4 and 5. In sections 4 and 5 we also report simulation results to validate the analysis. Conclusion is presented in the last section.

2 DESCRIPTION OF THE MODEL

A closed model of shared-nothing parallel TP systems with a fixed number (M) of activated transactions is adopted. We do not use open models since open models are weak in analyzing the effect of the maximal degree of transaction concurrency. The performance of the corresponding open system can then be obtained by a hierarchical solution method (Thomasian, 1993). This is not done here for the sake of brevity. Note that in a closed model, a committed transaction is immediately replaced by a new one.

The parallel TP system (Fig. 1) is composed of $Z+1$ nodes connected by an interconnection network. Each node has its own processor, memory, disks, communication processor, and a copy of system software. One of these nodes, known as the management node, is designated to handle all such management functions as transaction initiation, commit, and restart. All these functions are implemented by the Transaction Manager (TM) that resides at management node. The other Z nodes are the data processing nodes. We refer to these data processing nodes as the node 1, node 2, ..., and node Z. The managerment node is referred to as the node $Z+1$. All the data processing nodes have the same hardware configuration, and thereby have the same processing capacity.

The database, which consists of D data granules, is partitioned among Z data processing nodes (DeWitt et *al.*, 1986), one partition per node. Z is thereby the *degree of declustering* (DD) of the database, which reflects the degree of system parallelism. A data granule, as a lockable unit of data, consists of a group of records. All data granules have the same size, i.e., the same number of records.

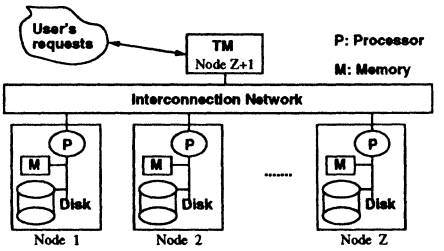

Figure 1: A shared-nothing parallel TP system.

The complexity of 2PL in shared-nothing parallel TP systems makes it practically impossible to find an exact analytic solution to its performance evaluation problem, even for extremely simplified cases. We extend the mean-value approach presented by Tay et *al.* (Tay et *al.*, 1985a) (Tay et *al.*, 1985b), where a flow diagram (Fig. 2) is used to chart the progress of transactions, and only the steady-state average values of the variables, instead of the instantaneous values, are used. By ignoring the probability distributions and avoiding the dynamics, analytic complexity is reduced. The accuracy of the results, however, may be reduced accordingly, especially for a system with much larger variations in arrivals and services. This is a weakness in our mean-value approach. It is argued, however, that the average performance of the system is thereby estimated. The flow diagram of a shared-nothing parallel TP system with dynamic 2PL with NW policy is shown in Fig. 2. In the following we explain this flow diagram. Notations used here are summarized at the end of this section.

The transactions that have been committed or aborted are said to be at the *initiating* stage, which corresponds to the management node. The transactions that are being committed are said to be at the *committing* stage. The residence time of the transactions at the initiating (resp. *committing*) stage is assumed to be T_{TM} (resp. T_{cmt}). We assume that the transactions at the committing stage can always be committed successfully.

A transaction is initiated at the initiating stage by transaction manager. Transaction manager initiates a transaction by splitting it into Z subtransactions and sending these subtransactions to the data processing nodes by a broadcast message, one subtransaction per node. A subtransaction is

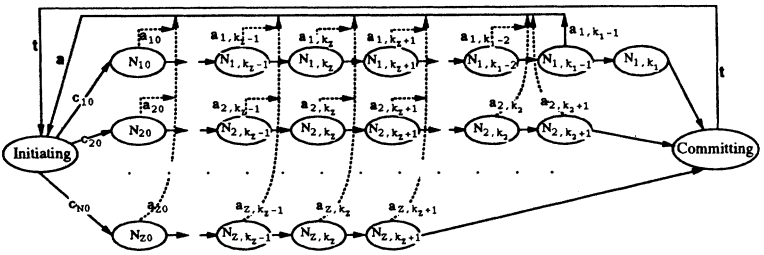

Figure 2: Flow diagram for a shared-nothing system with dynamic 2PL with NW policy. We refer to the node identified by $N_{f,i}$ in this diagram as stage (f,i). $N_{f,i}$ for $0 \leq i \leq k_f$ is the number of subtransactions holding i locks at node f. N_{f,k_f+1} is the number of subtransactions holding k_f locks and waiting for its siblings for starting 2PC protocol. $a_{f,i}$ is the abort rate of subtransactions at stage (f,i); $c_{f,i}$ is the rate at which subtransactions enter stage (f,i); t and a are transaction throughput and abort rate respectively.

called a *sibling* of the other subtransactions. Subtransaction at node f for $f = 1, 2, \ldots, Z$ is assumed to access k_f distinct granules from node f. Since we can always arrange the node number so that $k_{f_1} \geq k_{f_2}$ if $f_1 < f_2$, as a trick of the analysis, we assume that $k_{f_1} \geq k_{f_2}$ if $f_1 < f_2$. Granules are accessed uniformly and locked in exclusive mode (Ryu et *al.*, 1990b) (Tay et *al.*, 1985a) (Thomasian, 1993). Note that by distinguishing k_{f_1} from k_{f_2} we can characterize the access-skew of transactions to data processing nodes. Also note that the stage $(1, k_1+1)$ in Fig. 2 is ignored, since by the assumption that $k_{f_1} \geq k_{f_2}$ if $f_1 < f_2$, N_{1,k_1+1} is zero.

A subtransaction accessing k_f granules consists of $k_f + 2$ steps. In Fig. 2, step i at node f is represented by a graph node identified by $N_{f,i}$. The *0th* step performs subtransaction initialization and the first lock request for the first granule to be accessed. In the *ith* step $(1 \leq i \leq k_f - 1)$, the subtransaction accesses a granule to a disk, proceeds with a period of CPU usage for processing this granule, and then requests a lock for the next granule to be accessed. The k_f*th* step, after the subtransaction completes its granule accessing and processing, leads to entering the last step, i.e., the wait-for-commit step k_f+1. The last step in turn leads to starting a 2PC protocol. Note that until all the subtransactions of a transaction are completed, this transaction cannot be committed, i.e., 2PC protocol cannot be started. Therefore those subtransactions that have completed all their data-processing have to wait at their last step for their siblings to be completed.

A subtransaction acquires a lock if it is available, and proceeds with the execution of the next step; otherwise the corresponding transaction, and thus all its subtransactions are aborted. The aborted transaction is replaced immediately by a new transaction. Note that in real-life systems, the transaction that encounters a lock conflict is aborted and will be restarted after its blocker is committed or aborted by itself, and its place is taken by another transaction. This is equivalent to the situation that the aborted transaction is restarted with a new sequence of lock requests (Tay et *al.*, 1985a) (Thomasian, 1993).

In order to simplify our model, the following assumption is introduced: *If a subtransaction encounters a lock conflict, its siblings can be informed of this fact with little delay compared with the*

duration of a transaction step. This assumption seems restrictive. However, this is not really so. Let us consider a shared-nothing parallel TP system with processor speed of 100MIPS, disk service time of 13ms, and a high bandwidth interconnection network which introduces negligible delay (Franaszek et *al.*, 1993). Then the duration of a transaction step, denoted by C_s, can at least be estimated as 14ms. Since 2PL with NW policy is a restart-oriented CC method, provisions are made to reduce the overhead of restarting a transaction. We postulate that a lock conflict message is broadcasted to all the related nodes from the node where the lock conflict occurs, and such a message is assigned a high scheduling priority. Then the communication delay, denoted by C_c, is at most 0.1ms (the CPU overhead to send and receive a message is taken to be 5000 instructions (Franaszek et *al.*, 1993). Then C_c/C_s is only 0.0071 that is negligibly small. Therefore this assumption is acceptable.

The processing time of each step other than the last one is taken to be T. By T we denote the service demand for physical resources at each data processing node, such as accessing a disk, processing a granule, initiating a subtransaction, aborting a subtransaction, and sending a message. The processing time of the last step is derived from T. Given all the related service demands, T can be estimated by analytic solution or simulation.

Note that in this paper T, T_{TM}, and T_{cmt} are assumed to be given constants. It means that there are unlimited hardware resources and no software bottlenecks in the system (Franaszek et *al.*, 1985) (Tay et *al.*, 1985a) (Thomasian, 1993). This, in effect, factors out the influence of the resource contention, so that the effect of the data contention itself can be studied. Note that the resource contention is not ignored. In fact, the solution of our data contention model can be easily coupled with other standard resource contention models to obtain the overall system performance (Tay et *al.*, 1985a) (Tay et *al.*, 1985b). However, this is given here for the sake of brevity.

The notations to be used are summarized below, unless otherwise specified, $1 \leq f \leq Z$.

M	number of transactions (multiprogramming level)
M_{TM}	number of transactions at the initiating stage
M_{cmt}	number of transactions at the committing stage
M_{DP}	number of transactions at the data processing stages
Z	number of data processing nodes in the interconnection network
D	database size
D_f	size of the database partition at node f
K	transaction size, i.e., the number of lock requests per transaction
k_f	subtransaction size at node f
T_{TM}	residence time of transactions at the initiating stage
T_{cmt}	residence time of transactions at the committing stage
T	residence time of subtransactions at stage (f,i), $0 \leq i \leq k_f$
T_{f,k_f+1}	residence time of subtransactions at stage $(f, k_f + 1)$
$a_{f,i}$	abort rate of subtransactions at stage (f,i), $0 \leq i \leq k_f + 1$
$c_{f,i}$	rate at which subtransactions enter stage (f,i), $0 \leq i \leq k_f + 1$
$N_{f,i}$	number of subtransactions at stage (f,i), $0 \leq i \leq k_f + 1$
$P^c_{f,i}$	lock conflict probability when requesting the $i+1th$ lock at stage (f,i), $0 \leq i < k_f$; or the lock conflict probability at stage (f,i), $i = k_f, k_f + 1$, which are 0
$P^a_{f,i}$	abort probability of the subtransaction at stage (f,i), $0 \leq i \leq k_f + 1$
t	transaction throughput
a	transaction abort rate

3 ANALYSIS OF THE MODEL

In this section we analyze the system described in section 2. First we derive all the basic equations. On the basis of these basic equations, throughput t, abort rate a, and other performance measures can be obtained.

3.1 Basic equations

Under the assumption that the granules at a node are accessed uniformly, considering that a transaction does not request the locks that it already holds, we have

$$P_{f,i}^{c} = \begin{cases} \frac{G_f - i}{D_f - i} \simeq \frac{G_f}{D_f} & \text{for } f=1,2,\ldots,Z \quad i = 0,1,\ldots,k_f - 1, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $G_f = \sum_{j=1}^{k_f}(jN_{f,j}) + k_f N_{f,k_f+1} + k_f M_{cmt}$ is the number of locks held by the subtransactions at node f. Note that here we used a fairly standard assumption: *Compared with the number of locks held by all the subtransactions in node f, the number of locks that a subtransaction requires can be ignored* (Ryu et *al.*, 1990b) (Tay et *al.*, 1985a) (Thomasian, 1993) (Yu et *al.*, 1993). When a subtransaction at node f acquires all its k_f locks, it goes through with stage (f, k_f), and enters the waiting stage $(f, k_f + 1)$ so as to synchronize with the other siblings. Transactions at stage (f, k_f) and $(f, k_f + 1)$ no longer request any locks, and therefore they cannot encounter lock conflicts.

By (1), the probability for a transaction to encounter a lock conflict is eventually independent of the number of locks it holds. For the sake of simplicity, hereafter we use P_f^c to represent the lock conflict probability of transactions at node f.

By the Little's law we have

$$\begin{aligned} M_{TM} &= (a+t)T_{TM}, \\ M_{cmt} &= tT_{cmt}, \end{aligned} \tag{2}$$

$$N_{f,i} = \begin{cases} c_{f,i}T & \text{for } f=1,2,\ldots,Z \quad i = 0,1,\ldots,k_f, \\ c_{f,k_f+1}T_{f,k_f+1} & \text{for } f=1,2,\ldots,Z \quad i=k_{f+1}. \end{cases} \tag{3}$$

Since the number of transactions in the system is held constant (M), considering that a transaction eventually accesses each node by means of generating one and only one subtransaction at it, we have

$$M = M_{TM} + M_{DP} + M_{cmt}, \tag{4}$$

$$M_{DP} = \sum_{j=0}^{k_f+1} N_{f,j} \quad \text{for } f = 1,2,\ldots,Z. \tag{5}$$

By using (2) and (4) we have

$$M_{DP} = M - T_{TM}(a+t) - T_{cmt}t. \tag{6}$$

According to the flow conservation law, and considering that $a_{f,i} = c_{f,i}P_{f,i}^{a}$

$$c_{f,i} = c_{f,i-1} - a_{f,i-1} = c_{f,0}\prod_{j=0}^{i-1}(1 - P_{f,j}^{a}) \quad \text{for } f=1,2,\ldots,Z; \quad i=1,2\ldots,k_f+1. \tag{7}$$

By using (3) and (7) we have

$$N_{f,i} = N_{f,0} \prod_{j=0}^{i-1} (1 - P_{f,j}^a) \quad \text{for } f=1,2,\ldots,Z;\ i=1,2,\ldots,k_f. \tag{8}$$

Note that N_{f,k_f+1} remains unsolved.

Let f be a node such that $f \in \{1, 2, \ldots, Z\}$, by (5) and (8) we have $M_{DP} = N_{f,0} \sum_{i=0}^{k_f} \prod_{j=0}^{i-1} (1 - P_{f,j}^a) + N_{f,k_f+1}$, recall that $N_{1,k_1+1} = 0$. Therefore

$$N_{f,0} = \frac{M_{DP} - N_{f,k_f+1}}{\sum_{i=0}^{k_f} \prod_{j=0}^{i-1} (1 - P_{f,j}^a)}. \tag{9}$$

Hence,

$$\begin{aligned} G_f &= \sum_{i=1}^{k_f} (iN_{f,i}) + k_f N_{f,k_f+1} + k_f T_{cmt} t \\ &= N_{f,0} \sum_{i=1}^{k_f} \left[i \prod_{j=0}^{i-1} (1 - P_{f,j}^a) \right] + k_f N_{f,k_f+1} + k_f T_{cmt} t \\ &= \frac{(M_{DP} - N_{f,k_f+1}) \sum_{i=1}^{k_f} \left[i \prod_{j=0}^{i-1} (1 - P_{f,j}^a) \right]}{\sum_{i=0}^{k_f} \prod_{j=0}^{i-1} (1 - P_{f,j}^a)} + k_f N_{f,k_f+1} + k_f T_{cmt} t, \end{aligned}$$

then by (1) we have P_f^c for $f=1,2,\ldots,Z$

$$P_f^c = \frac{(M_{DP} - N_{f,k_f+1}) \sum_{i=1}^{k_f} \left[i \prod_{j=0}^{i-1} (1 - P_{f,j}^a) \right]}{D_f \sum_{i=0}^{k_f} \prod_{j=0}^{i-1} (1 - P_{f,j}^a)} + \frac{k_f N_{f,k_f+1} + k_f T_{cmt} t}{D_f}. \tag{10}$$

We next derive N_{f,k_f+1} for $f=2,3,\ldots,Z$. By (3) and (7) we have

$$N_{f,k_f+1} = c_{f,k_f+1} T_{f,k_f+1} = \frac{N_{f,0}}{T} \prod_{j=0}^{k_f} (1 - P_{f,j}^a) T_{f,k_f+1} \quad \text{for } f=2,3,\ldots,Z. \tag{11}$$

There are two unknown quantities in (11): $P_{f,j}^a$, and T_{f,k_f+1}. First we derive T_{f,k_f+1}, the residence time of a subtransaction at stage $(f, k_f + 1)$. This problem can be solved by considering both successfully completed and aborted transactions. For a successfully completed transaction, since $k_1 \geq k_f$ for any node f, T_{f,k_f+1} can be expressed as $(k_1+1)T - (k_f+1)T$, i.e., $(k_1 - k_f)T$. For an aborted transaction, the arrival time to stage $(f, k_f + 1)$ is $(k_f + 1)T$. In the time interval $\Delta T = (k_1 - k_f)T$ it is aborted. By the assumption that $k_i \geq k_f$ if $i < f$, for any given f $(2 \leq f \leq Z)$, only the subtransactions at stage (i, x) for $i < f$ and $0 \leq x < k_i$ can abort the subtransaction at stage $(f, k_f + 1)$. This is because subtransactions at other stages have finished their data accesses when subtransaction at node f enters stage $(f, k_f + 1)$. When the subtransaction at node f enters its stage $(f, k_f + 1)$, the remaining number of lock-requesting steps of the subtransaction at node i is $k_i - k_f - 1$. Therefore

$$\begin{aligned} T_{f,k_f+1} = \ & T(k_1 - k_f) \prod_{i=1}^{f-1} \prod_{j=1}^{k_i - k_f - 1} (1 - P_i^c) + \\ & T \sum_{j=1}^{k_1 - k_f - 1} \{ j [\prod_{l=1}^{j-1} \prod_{i=1}^{f-1} (1 - e_{i,k_f+l} P_i^c)] [1 - \prod_{i=1}^{f-1} (1 - e_{i,k_f+j} P_i^c)] \}, \end{aligned} \tag{12}$$

where $e_{i,j}$ is a function defined as follows

$$e_{i,j} = \begin{cases} 1 & \text{for } i=1,2,\ldots,Z;\ 0 \leq j < k_i, \\ 0 & \text{otherwise.} \end{cases}$$

We next derive the abort probability $P_{f,i}^a$ and abort rate $a_{f,i}$ for $f = 1, 2, \ldots, Z$ and $i = 0, 1, \ldots, k_f$. A transaction, and therefore all its subtransactions are aborted each time one of these subtransactions

encounters a lock conflict. For a subtransaction, it is *aborted directly* if it encounters a lock conflict; or *aborted indirectly* if one of its siblings encounters a lock conflict. The direct abort probability is just the lock conflict probability P_f^c. The indirect abort probability is calculated by $1 - \prod_{j=1, j\neq f}^{Z}(1 - e_{j,i}P_j^c)$. Then the abort probability and abort rate are respectively

$$P_{f,i}^a = 1 - \prod_{j=1}^{Z}(1 - e_{j,i}P_j^c) \quad \text{for } f=1,2,\ldots,Z \quad i = 0,1,\ldots,k_f, \tag{13}$$

$$a_{f,i} = c_{f,i}P_{f,i}^a \quad \text{for } f=1,2,\ldots,Z \quad i = 0,1,\ldots,k_f. \tag{14}$$

3.2 Transaction throughput and abort rate

Two performance measures with which we are concerned are transaction *throughput* and *abort rate* of the system. By the assumption that $k_i \geq k_f$ if $i < f$, node 1 has the maximal stage number. If a transaction is not aborted at node 1, it will be committed successfully. The transaction throughput is thereby entirely determined by node 1. Therefore

$$t = \frac{N_{1,k_1}}{T}. \tag{15}$$

Similarly, the transaction abort rate can also be determined by node 1 solely. By (3), (13), and (14) we obtain

$$a = \sum_{i=0}^{k_1-1} a_{1,i} = \sum_{i=0}^{k_1-1} c_{1,i}P_{1,i}^a = \frac{1}{T}\sum_{i=0}^{k_1-1}\left[N_{1,i}(1 - \prod_{j=1}^{Z}(1 - e_{j,i}P_j^c))\right]. \tag{16}$$

4 PERFORMANCE ANALYSIS IN THE CASE OF NO ACCESS SKEW

In this section, we examine a shared-nothing parallel TP system with the assumption that all the subtransactions of a transaction have the same size (k), i.e., no access skew occurs. A comparison between analytical results and simulation results is also included in this section. In the case of no access skew, $P_{f,k}^a$, $P_{f,k+1}^a$, and $N_{f,k+1}$ are equal to 0 for any node f; the subtransaction size is calculated by $k = K/Z$; the sizes of database partitions are calculated by $D_f = D/Z$; $P_{f_1}^c$ is equal to $P_{f_2}^c$ for any nodes f_1 and f_2.

Let $(1 - P_f^c)^Z = q_f$. Note that q_{f_1} is equal to q_{f_2} for any nodes f_1 and f_2, and all of them are represented by q. By using (5), (6), (8), (9), (10), (13), (15), and (16) we have

$$\frac{\frac{M}{D_f}\left[(1-q^{k+1})T * \frac{\sum_{i=1}^{k} iq^i}{\sum_{i=0}^{k} q^i} + kT_{cmt}(1-q)q^k\right]}{(1-q^{k+1})T + (1-q)[T_{TM} + q^kT_{cmt}]} + q^{1/Z} - 1 = 0, \tag{17}$$

$$t = \frac{M(1-q)q^k}{(1-q^{k+1})T + (1-q)[T_{TM} + q^kT_{cmt}]}, \tag{18}$$

$$a = \frac{M(1-q)(1-q^k)}{(1-q^{k+1})T + (1-q)[T_{TM} + q^kT_{cmt}]}. \tag{19}$$

About the derivation of equations (17), (18), and (19), see the appendix given at the end of this paper. Given K, M, Z, D, T, T_{TM}, and T_{cmt}, we can solve equations (17), (18), and (19) by numerical methods for q_f, t, and a. We can then compute P_f^c from $P_f^c = 1 - q_f^{1/Z}$.

4.1 Comparisons with simulation results

Now we compare the results from the analytic model with estimates from a simulator. The simulator is a preliminary version of that presented by Wang et *al.* (Wang et *al.*, 1997). For simplifying our simulator, we assumed that if a subtransaction encounters a lock conflict, its siblings can be informed of this fact with little delay compared with the duration of a transaction step. We have examined this in section 2 and found that this assumption has little effect on simulation results. In addition, the service times of CPUs and disks are taken to be constants. For each simulation result, relative half-width of 5% about the mean value were calculated using a batch means method at 95% confidence level. The results of our comparisons are given in Figs. 3(a) and 3(b). From these figures we can see that the analysis captures the characteristics (i.e., transaction abort rate and throughput) of the shared-nothing parallel TP systems well.

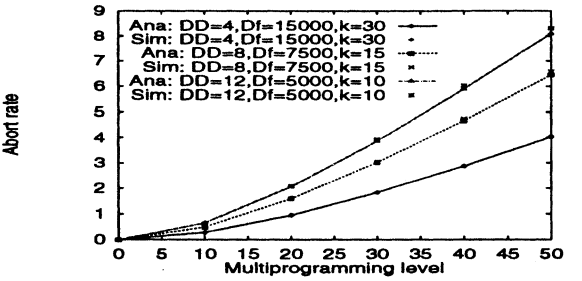

Figure 3(a): Comparisons with simulation results in the case of no access skew (Ana: analysis; Sim: simulation): Abort rate (D=60000, K=120, T=1.0, T_{TM}=1.5, T_{cmt}=2.0).

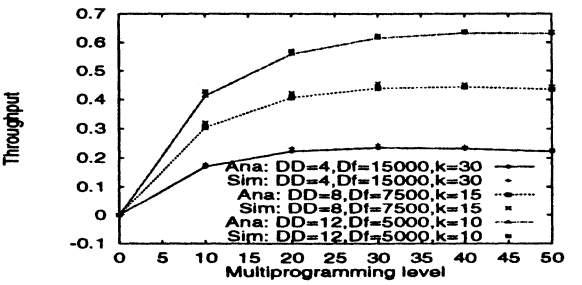

Figure 3(b): Comparisons with simulation results in the case of no access skew (Ana: analysis; Sim: simulation): Throughput (D=60000, K=120, T=1.0, T_{TM}=1.5, T_{cmt}=2.0).

4.2 Abort rate and degree of declustering

Here we examine the relation between the abort rate and the degree of declustering (DD) of the database. Figure 4 shows that abort rate increases with increasing DD. We think this is because

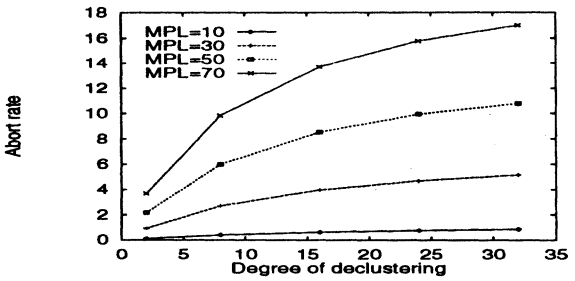

Figure 4: Abort rate vs. *DD* (D=60000, K=96, T=1.0, T_{TM}=1.5, T_{cmt}=2.0).

the parallel execution pattern (i.e., a transaction proceeds in *DD* data access flows) tends to cause a high lock conflict rate. The larger the *DD*, the higher the lock conflict rate. Because for 2PL with NW policy, a transaction is aborted whenever it encounters a lock conflict, it can be expected that a larger *DD* causes a higher abort rate. Other kinds of restart-oriented locking-based CC methods are expected to behave in the same way, since they also abort transactions on the basis of lock conflicts. From the above observation we see that a successful restart-oriented CC method for the centralized TP system is not necessarily a successful CC method for the shared-nothing parallel TP system, since it may not take the *DD* into consideration. In shared-nothing parallel TP systems, more attention should be paid to *DD*, since the abort rate is affected by it.

4.3 Throughput and degree of declustering

Furthermore, we examine the relation between the transaction throughput and the degree of declustering (*DD*) of database. Figure 5(a) shows that throughput increases with increasing *DD*. We think this is because the work per processor is reduced. Note that although increasing *DD* leads to an increase in abort rate (see Fig. 4) that tends to waste a lot of work and decrease throughput, the reduced work when increasing *DD*, however, prevails. Also note that here the effect of the resource contention is not taken into account; otherwise the results may be different. Figure 5(b) shows throughput characteristics when the additional overhead of the parallel processing is taken into account. Here T_{TM} and T_{cmt} are defined to be $\rho_1 * DD$ and $\rho_2 * DD$, respectively. ρ_1 and ρ_2 are two positive real numbers. Figure 5(b) tells us that there is a tradeoff between the benefit of a large *DD* and the additional overhead incurred by this large *DD*.

Next we derive some important conditions for preventing data contention thrashing by considering the equation (17), (18), and (19). Let Z, k, T, T_{TM}, and T_{cmt} be given constants, let $\lambda = M/D_f$, and let $\theta^{(M,D_f)}$ be the predicted value of performance measure θ given M and D_f, then $q^{(\rho M,\rho D_f)} = q^{(M,D_f)}$, $t^{(\rho M,\rho D_f)} = \rho t^{(M,D_f)}$, and $a^{(\rho M,\rho D_f)} = \rho a^{(M,D_f)}$ for any $\rho > 0$. Therefore for any given k, the predicted data contention thrashing point has the same λ value. Let it be $\lambda_{max}(k)$. Then M/D_f should be less than or equal to $\lambda_{max}(k)$, so that data contention thrashing should not occur (see Tay et *al.*, 1985a).

Furthermore, assume Z, T, T_{TM}, and T_{cmt} to be 8, 1.0, 1.5, and 2.0 respectively. We attempt to find the relationship between $\lambda_{max}(k)$ and k in this case. We can obtain the $\lambda_{max}(k)$ corresponding to every given k. Let $\lambda_{max}(k) * f(k) = 1$, where $f(k)$ is a polynomial in k. By using *the Mathematica* (see

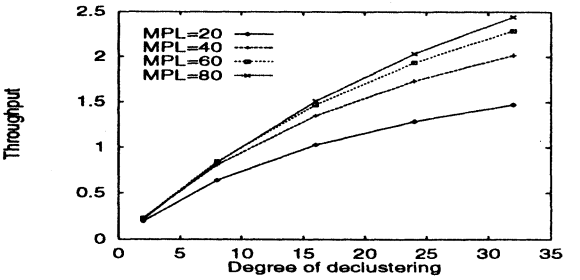

Figure 5(a): Throughput vs. DD (D=60000, K=96, T=1.0, T_{TM}=1.5, T_{cmt}=2.0).

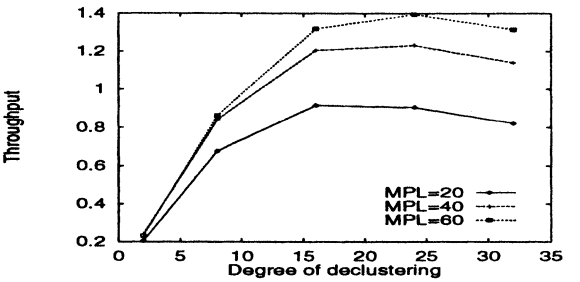

Figure 5(b): Throughput vs. DD with consideration of the additional overhead of parallel processing (D=60000, K=96, T=1.0, ρ_1=0.1, ρ_2=0.2, $T_{TM}=\rho_1 * DD$, $T_{cmt}=\rho_2 * DD$).

section 3.8 of Wolfram, 1991) we found that when $f(k)$ is $1.00388k^2$, the fit is good. Data contention thrashing tends to occur when λk^2 is greater than 1/1.00388, or 0.996135. Because $D_f = D/Z$ and $k_f = K/Z$, a shared-nothing parallel TP system in this case should satisfy inequality $\frac{MK^2}{DZ} \leq 0.996135$.

The same experiments as above have been done with T_{TM} and T_{cmt} being varied from 1.0 to 5.0 with step 1.0 respectively for $Z \in \{2, 8, 16\}$, and T being fixed at 1.0. The equation $\lambda_{max}(k) * f(k) = 1$ remains true with $f(k) = C_1(Z, T, T_{TM}, T_{cmt}) * k^2 + C_2(Z, T, T_{TM}, T_{cmt}) * k^3$, where $C_1(Z, T, T_{TM}, T_{cmt})$ and $C_2(Z, T, T_{TM}, T_{cmt})$ (hereafter we use C_1 and C_2 to represent them for simplicity) are two constants given Z, T, T_{TM}, and T_{cmt}. In fact, when T_{TM} is not greater than 1.5, it is enough for $f(k)$ to be C_1k^2. Then $M/D_f * [C_1k^2 + C_2k^3]$ should be less than or equal to 1.0. Considering that $D_f = D/Z$ and $k_f = K/Z$, the following inequality should be satisfied

$$\frac{MK^2}{DZ} * (C_1 + \frac{C_2K}{Z}) \leq 1.0. \tag{20}$$

Next we consider the case when ZP_f^c is very small. If ZP_f^c is small enough (e.g., 0.05), then $(1 - P_f^c)^Z$ approximates to $1 - ZP_f^c$. Let the latter be q, then we have (see appendix about the

derivation)

$$\frac{\frac{MZ}{D_f}\left[(1-q^{k+1})T * \frac{\sum_{i=1}^{k} iq^i}{\sum_{i=0}^{k} q^i} + kT_{cmt}(1-q)q^k\right]}{(1-q^{k+1})T + (1-q)[T_{TM} + q^k T_{cmt}]} + q - 1 = 0.$$

Let $\lambda' = MZ/D_f$, and let $\theta^{(M,D_f/Z)}$ be the predicted value of performance measure θ given M and D_f/Z, then $q^{(\rho M,\rho D_f/Z)} = q^{(M,D_f/Z)}$, $t^{(\rho M,\rho D_f/Z)} = \rho t^{(M,D_f/Z)}$, and $a^{(\rho M,\rho D_f/Z)} = \rho a^{(M,D_f/Z)}$ for any $\rho > 0$. Therefore for any given k, the predicted data contention thrashing point has the same λ' value. Let it be $\lambda'_{max}(k)$. Then MZ/D_f should be less than or equal to $\lambda'_{max}(k)$. Similarly as above, the following inequality should be satisfied in this case

$$\frac{MK^2}{D} * (C'_1 + \frac{C'_2 K}{Z}) \leq 1.0. \tag{21}$$

If T_{TM} is small enough (e.g., 1.5), it is enough for f(k) to be $C'_1 k^2$, then the inequality becomes

$$C'_1 \frac{MK^2}{D} \leq 1.0. \tag{22}$$

In brief, a shared-nothing parallel TP system using 2PL with NW policy should satisfy inequality (20). Especially, if ZP_f^c is less than 0.1, it may also attempt to maintain inequality (21). In the latter case, if T_{TM} is less than 2.0, the predicted data contention thrashing point tends to be independent of the degree of declustering, and is solely determined by M, D, and K, as shown in inequality (22). It should be noted that the effect of the resource contention is not included in the data contention thrashing; otherwise the thrashing would occur sooner than indicated by these inequalities.

From the above inequalities we know that data contention thrashing is more sensitive to k than to M. A large DD can reduce k, and therefore can relieve data contention thrashing, at least from the viewpoint of the effect of k. Furthermore, the reduced k can help to increase M without incurring data contention thrashing. In addition, the shared-nothing parallel TP system is very susceptible to access skew, since access skew tends to increase k.

5 PERFORMANCE ANALYSIS IN THE CASE OF ACCESS SKEW

In section 4 we analyzed a shared-nothing parallel TP system without access skew. In this section we examine the impact of access skew on system performance. We assume that transactions access database according to *b-20* rule. For example, by the *80-20* rule, *80%* of the accesses goes to *20%* of the database. The node number Z is fixed to be 5 (recall that Z is equal to DD in our model). A transaction consists of five subtransactions, one subtransaction per node. One of these subtransactions accesses $\lfloor K * b\% \rfloor$ granules from the first node. Each of the other subtransactions accesses a quarter of the remaining granules from the corresponding node. The database is partitioned among all these five nodes. Transaction size (K) is set to 40. b is set to 80, 60, 40, and 20 in turn. A large b means a high level of access skew. When b is 20, no access skew occurs. Database size (D) is set to be 20000. The sizes of database partitions are calculated by $D_f = D/Z$.

Let g be a node in node set {2,3,4,5}. Now we derive all the necessary equations. By (10), we have

$$\begin{aligned} P_1^c &= \frac{M_{DP}}{D_1} \frac{\sum_{i=1}^{k_1}\left[i\prod_{j=0}^{i-1}(1-P_{1,j}^a)\right]}{\sum_{i=0}^{k_1}\prod_{j=0}^{i-1}(1-P_{1,j}^a)} + \frac{k_1 T_{cmt}t}{D_1}, \\ P_g^c &= \frac{(M_{DP}-N_{g,k_g+1})\sum_{i=1}^{k_g}\left[i\prod_{j=0}^{i-1}(1-P_{g,j}^a)\right]}{D_g\sum_{i=0}^{k_g}\prod_{j=0}^{i-1}(1-P_{g,j}^a)} + \frac{k_g N_{g,k_g+1} + k_g T_{cmt}t}{D_g}. \end{aligned} \tag{23}$$

From (11) and (12)

$$N_{g,k_g+1} = \frac{N_{g,0}}{T} \prod_{j=0}^{k_g} (1 - P^a_{g,j}) T_{g,k_g+1}$$

$$= N_{g,0} \left[\sum_{j=1}^{k_1-k_g-1} \left[j(1-P^c_1)^{j-1} P^c_1 \right] + (k_1 - k_g)(1-P^c_1)^{k_1-k_g-1} \right] \prod_{j=0}^{k_g} (1 - P^a_{g,j}). \quad (24)$$

From (9)

$$N_{g,0} = \frac{M_{DP} - N_{g,k_g+1}}{\sum_{i=0}^{k_g} \prod_{j=0}^{i-1} (1 - P^a_{g,j})}. \quad (25)$$

From (13)

$$P^a_{1,i} = \begin{cases} 1 - (1-P^c_1)(1-P^c_g)^4 & i = 0, 1, \ldots, k_g - 1, \\ P^c_1 & i = k_g, k_g + 1, \ldots, k_1 - 1, \end{cases}$$

$$P^a_{g,i} = \begin{cases} 1 - (1-P^c_1)(1-P^c_g)^4 & i = 0, 1, \ldots, k_g - 1, \\ e_{1,k_g} P^c_1 & i = k_g. \end{cases} \quad (26)$$

From (8) and (9)

$$N_{1,i} = \frac{M_{DP} \prod_{j=0}^{i-1} (1 - P^a_{1,j})}{\sum_{l=0}^{k_1} \prod_{j=0}^{l-1} (1 - P^a_{1,j})} \quad i = 1, 2, \ldots, k_1.$$

Then (15) and (16) can be rewritten as

$$t = \frac{M_{DP}}{T} \frac{\prod_{j=0}^{k_1-1} (1 - P^a_{1,j})}{\sum_{l=0}^{k_1} \prod_{j=0}^{l-1} (1 - P^a_{1,j})}, \quad (27)$$

$$a = \frac{M_{DP}}{T \sum_{l=0}^{k_1} \prod_{j=0}^{l-1} (1 - P^a_{1,j})} \sum_{i=0}^{k_1-1} \left[\prod_{j=0}^{i-1} (1 - P^a_{1,j}) (1 - \prod_{j=1}^{Z} (1 - e_{j,i} P^c_j)) \right]. \quad (28)$$

Given K, M, Z, D, T, T_{TM}, and T_{cmt}, we can solve equation (23), (24), (25), (26), (27), (28) and (6) by numerical methods for p_1, p_g, t, and a.

5.1 Comparisons with simulation results

The same simulation experiments as that presented in section 4.1 were done with the parameter settings for the comparisons here. The results are given in Figs. 6(a) and 6(b). These figures show again that the analysis captures the characteristics of the data contention and the shared-nothing parallel TP system, even though the access skew occurs.

5.2 Results and discussion

Figure 7(a) shows the effect of access skew on throughput. From this figure we see that access skew degrades throughput. This result coincides with rules given in section 4.3, which states that data-contention thrashing is very sensitive to access skew, since access skew increases k. From Fig. 7(b), however, we see that access skew helps to reduce abort rate. If resource contention is taken into account, such a decrease in abort rate may help to increase throughput.

Figure 7(c) compared the throughputs obtained when considering all the nodes as a whole and when considering the hot node (i.e., node 1) only. This figure tells us that when the level of access

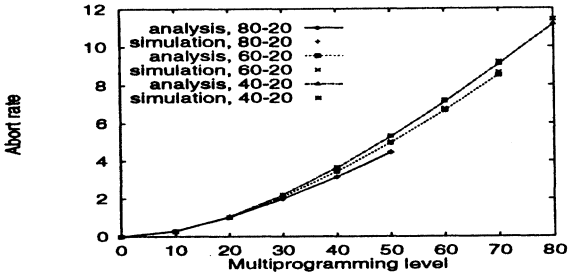

Figure 6(a): Comparisons with simulation results in the case of access skew: Abort rate (Z=5, D=20000, K=40, T=1.0, T_{TM}=1.5, T_{cmt}=2.0).

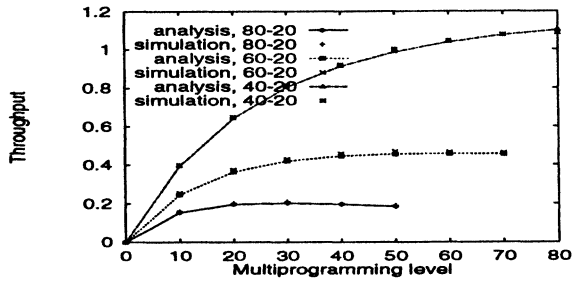

Figure 6(b): Comparisons with simulation results in the case of access skew: Throughput (Z=5, D=20000, K=40, T=1.0, T_{TM}=1.5, T_{cmt}=2.0).

skew becomes high enough, e.g., *60-20* in our experiments, throughput tends to be determined solely by the hot nodes. We think at that time, rules presented in section 4.3 are also applicable if there is no access-skew in the hot nodes. In addition, Fig. 7(c) tells us that a system with high level of access skew is equivalent in throughput to such a system that only includes the hot nodes.

6 CONCLUSION

We have proposed an analytic model for studying the performance behavior of shared-nothing parallel TP systems with dynamic 2PL with NW policy. All the essential factors of shared-nothing parallel TP systems, such as subtransaction initiation, 2PC protocol, degree of declustering, and access skew, were taken into consideration. The results from the analytic model were compared with the estimates from a simulation model. By simulation experiments we see that the analysis captures the characteristics of shared-nothing parallel TP systems well.

By applying the proposed model to analyze the performance of shared-nothing parallel TP systems,

we derived some important conditions among transaction size, database size, multiprogramming level (M), and the degree of declustering (DD) for avoiding data contention thrashing. It is shown that these conditions should be satisfied so that the system works safely. If they are violated, data contention thrashing tends to occur.

By our observation, data-contention thrashing is more sensitive to transaction size K and sub-transaction size k than to M. To keep both K and k small is important. A large DD is beneficial since it helps to reduce k. Accordingly, M can be increased as large as possible. It was shown that, however, too large a DD is unacceptable due to the additional overhead brought by it.

It was found that a larger DD leads to a higher abort rate. This means that in selecting the restart-oriented CC method for shared-nothing parallel TP systems, DD is an important performance factor. On the contrary, given a restart-oriented CC method, DD cannot be increased unlimitedly. An inadequate DD would cause a high lock conflict rate, and has negative effect on shared-nothing parallel TP systems.

We also noted that our model has some limitation. In the model, it is assumed that a transaction gets some data granules from every node. In some cases of real-life systems, however, some nodes may not accessed at all. As the future work, we intend to take this fact into our model. In addition, more accurate simulation experiments should be performed.

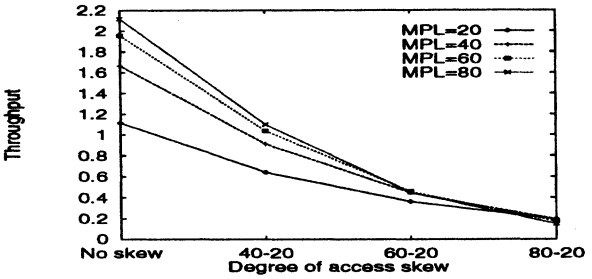

Figure 7(a): Throughput vs. access skew (Z=5, D_1=4000, D_g=4000, T=1.0, T_{TM}=1.5, T_{cmt}=2.0, K=40).

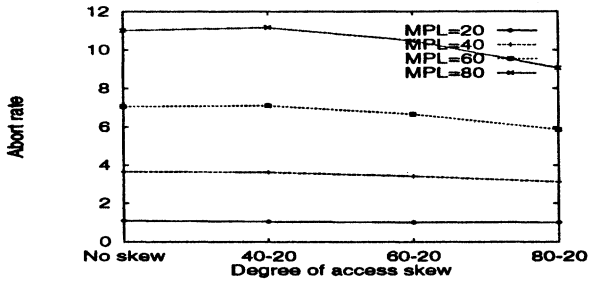

Figure 7(b): Abort rate vs. access skew (Z=5, D_1=4000, D_g=4000, T=1.0, T_{TM}=1.5, T_{cmt}=2.0, K=40).

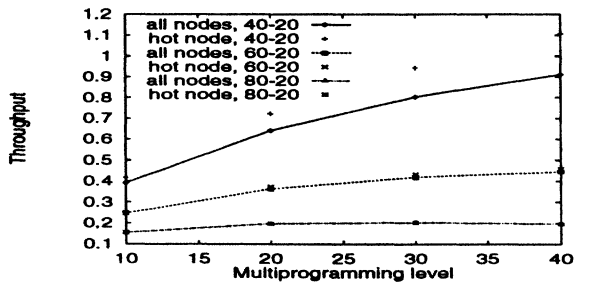

Figure 7(c): Throughput vs. access skew – considering all the nodes and the hot nodes respectively (D_1=4000, D_g=4000, T=1.0, T_{TM}=1.5, T_{cmt}=2.0, K=40).

7 REFERENCES

Boral, H., Alexander, W., Clay, L., Copeland, G., Danforth, S., Franklin, M., Hart, B., Smith, M., and Valduriez, P. (1990) Prototyping Bubba, a highly parallel database system. *IEEE Trans. Knowledge Data Eng.*, **2**, 1, 4-24.

Chesnais, A., Gelenbe, E., and Mitrani, I. (1983) On the Modeling of Parallel Access to Shared Data. *Commun. ACM*, **26**, 3, 196-202.

DeWitt, D.J., Gerber, R., Graefe, G., Heytens, M., Kumar, K., and Muralikrishna, M. (1986) GAMMA - A high performance dataflow database machine. *Proc. 12th VLDB Conf., Tokyo, Japan*, 25-28.

Franaszek, P.A., and Robinson, J.T. (1985) Limitations of Concurrency in Transaction Processing. *ACM Trans. Database Sys.*, **10**, 1, 1-28.

Franaszek, P.A., Haritsa, J.R., Robinson, J.T., and Thomasian, A. (1993) Distributed Concurrency Control Based on Limited Wait-Depth. *IEEE Trans. Parallel Distributed Sys.*, **4**, 11, 1246-1264.

Hsu, M., and Zhang, B. (1992) Performance Evaluation of Cautious Waiting. *ACM Trans. Database Sys.*, **17**, 3, 477-512.

Ryu, I.K., and Thomasian, A. (1990a) Performance Analysis of Dynamic Locking with the No-Waiting Policy. *IEEE Trans. Softw. Eng.*, **16**, 6, 684-698.

Ryu, I.K., and Thomasian, A. (1990b) Analysis of Database Performance with Dynamic Locking. *J. ACM*, **37**, 3, 491-523.

Stonebraker, M. (1986) The Case for Shared Nothing. *Database Eng. Bull.*, **9**, 1, 4-9.

Tandem Database Group (1987) NonStop SQL, A distributed, high-performance, high-reliability implementation of SQL. *Proc. 2nd Workshop on High Performance Transaction Systems, Asilomar, CA*, 60-104.

Tay, Y.C., Suri, R., and Goodman, N. (1985a) A Mean Value Performance Model for Locking in Databases: The No-Waiting Case. *J. ACM*, **32**, 3, 618-651.

Tay, Y.C., Goodman, N., and Suri, R. (1985b) Locking Performance in Centralized Databases. *ACM Trans. Database Sys.*, **10**, 4, 415-462.

Teradata (1983) DBC/1012 Data Base Computer Concepts & Facilities. Document No. C02-001-00, Teradata Corp..

Thomasian, A. (1991) Centralized Concurrency Control Methods for High-End TP *ACM SIGMOD Rec.*, **20**, 3, 106-115.

Thomasian, A. (1993) Two-Phase Locking Performance and Its Thrashing Behavior. *ACM Trans. Database Sys.*, **18**, 4, 579-625.

Wang, J., Li, J., and Kameda, H. (1997) Simulation Studies on Concurrency Control in Parallel Transaction Processing Systems. *Parallel Computing*, **23**, 6, 755-775.

Wolfram, S. (1991) *Mathematica, a system for doing mathematics by computer (second edition)*, Addison-Wesley Publishing Company, Inc..

Yu, P.S., Dias, D.M., and Lavenberg, S.S. (1993) On the Analytical Modeling of Database Concurrency Control. *J. ACM,* 40, 4, 831-872.

Appendix

In this appendix we explain briefly the derivations of the equations (17), (18), and (19) used in section 4. By (10), we have

$$P_f^c = \frac{M_{DP}}{D_f} \frac{\sum_{i=1}^{k} \left[i \prod_{j=0}^{i-1} (1 - P_{f,j}^a) \right]}{\sum_{i=0}^{k} \prod_{j=0}^{i-1} (1 - P_{f,j}^a)} + \frac{kT_{cmt}t}{D_f} \quad \textit{for } f=1,2,\ldots,Z.$$

By (13)

$$P_{f,i}^a = 1 - (1 - P_f^c)^Z \quad \textit{for } f=1,2,\ldots,Z;\ i=0,1,\ldots,k\text{-}1.$$

Therefore

$$P_f^c = \frac{M_{DP}}{D_f} \frac{\sum_{i=1}^{k} \left[i \prod_{j=0}^{i-1} (1 - P_f^c)^Z \right]}{\sum_{i=0}^{k} \prod_{j=0}^{i-1} (1 - P_f^c)^Z} + \frac{kT_{cmt}t}{D_f} \quad \textit{for } f=1,2,\ldots,Z.$$

Let $(1 - P_f^c)^Z = q_f$, note that q_{f_1} is equal to q_{f_2} for any node f_1 and f_2. By (6), the above equation can be rewritten as

$$\frac{[M - T_{TM}(a+t) - T_{cmt}t]}{D_f} \frac{\sum_{i=1}^{k} i q_f^i}{\sum_{i=0}^{k} q_f^i} + q_f^{1/Z} + \frac{kT_{cmt}t}{D_f} - 1 = 0 \quad \textit{for } f=1,2,\ldots,Z. \tag{A-1}$$

From (8) and (9)

$$N_{1,k} = \frac{M_{DP} \prod_{j=0}^{k-1} (1 - P_{1,j}^a)}{\sum_{i=0}^{k} \prod_{j=0}^{i-1} (1 - P_{1,j}^a)} = \frac{M_{DP} q_1^k}{\sum_{i=0}^{k} q_1^i} = \frac{M_{DP}(1 - q_1) q_1^k}{1 - q_1^{k+1}}.$$

From (15), (16), (5) and (6)

$$t = \frac{N_{1,k}}{T} = \frac{M - T_{TM}(a+t) - T_{cmt}t}{T} \frac{(1 - q_1) q_1^k}{1 - q_1^{k+1}}, \tag{A-2}$$

$$\begin{aligned} a &= \frac{1 - (1 - P_1^c)^Z}{T} [M_{DP} - N_{1,k}] = \frac{1 - q_1}{T} \left(M_{DP} - \frac{M_{DP}(1 - q_1) q_1^k}{1 - q_1^{k+1}} \right) \\ &= \frac{M - T_{TM}(a+t) - T_{cmt}t}{T} \frac{(1 - q_1)(1 - q_1^k)}{1 - q_1^{k+1}}. \end{aligned} \tag{A-3}$$

Then by solving equations (A-1), (A-2) and (A-3) we can derive the equations (17), (18), and (19). Note that q is used instead of q_f here. By (A-2) and (A-3) we have

$$a = \frac{1 - q^k}{q^k} t. \tag{A-4}$$

Then by (A-2) and (A-4) we have the equation (18)

$$t = \frac{M(1-q)q^k}{(1-q^{k+1})T + (1-q)[T_{TM} + q^k T_{cmt}]},$$

by (A-4) and (18) we have the equation (19)

$$a = \frac{M(1-q)(1-q^k)}{(1-q^{k+1})T + (1-q)[T_{TM} + q^k T_{cmt}]},$$

and by (18), (19) and (A-1) we have the equation (17)

$$\frac{\frac{M}{D_f}\left[(1-q^{k+1})T * \frac{\sum_{i=1}^{k} iq^i}{\sum_{i=0}^{k} q^i} + kT_{cmt}(1-q)q^k\right]}{(1-q^{k+1})T + (1-q)[T_{TM} + q^k T_{cmt}]} + q^{1/Z} - 1 = 0.$$

8 BIOGRAPHY

Jiahong Wang is now a foreign researcher in the Institute of Information Sciences and Electronics, University of Tsukuba, Japan. From 1998, he will be an Assistant Professor in the Faculty of Software and Information Sciences, Iwate Prefectural University, Japan. His research interests include concurrency control, transaction processing, distributed/parallel systems, and modeling and performance evaluation.

Jie Li has as been with the Institute of Information Sciences and Electronics, University of Tsukuba, Japan, since 1993, where he is currently an Associate Professor. His research interests are in distributed/parallel computing, mobile computing, and modeling and performance evaluation. He is a member of IEEE and ACM. He is serving as a manager of the Study Group on System Evaluation of the Information Processing Society of Japan.

Hisao Kameda received the B.Sc., M.Sc., and D.Sc. degrees all from the University of Tokyo, Tokyo, Japan. Presently he is a Professor at the Institute of Information Sciences and Electronics, University of Tsukuba, Japan. He has been conducting research on operating system design principles, operating system scheduling, performance measurement and queueing analysis of computer systems, software reliability, distributed processing, etc. He has also been continuously interested in general systems implications of computer systems. He is now the Chairman of the Study Group on System Evaluation of the Information Processing Society of Japan.