A New Methodology to Evaluate Locking Protocols

Yin-Fu Huang and Yeh-Hao Chin

Abstract—In this paper, the average lock range (ALR) is proposed as an evaluation factor for measuring the strengths and weaknesses of locking-based concurrency control methods, for both structural and nonstructural locking. The new methodology provides a simple and general way to analyze the performance of any locking method, and requires no queuing model. Based on the concept of the ALR, two popular locking protocols, the 2PL protocol and the tree protocol, are analyzed, and a simulation is done to validate the correctness of the ALR model.

Index Terms—Concurrency control, locking protocol, mathematical analysis, performance evaluation, simulation.

I. INTRODUCTION

In a database management system (DBMS), users can access shared data under the assumption that the data satisfy certain consistency assertions. A concurrency control mechanism is required to maintain database consistency, when a set of transactions is running concurrently against a shared database. In general, locking is a common technique used in synchronizing a set of concurrent transactions. Among the locking protocols having been proposed, the 2PL protocol [2] and the tree protocol [5] are the most popular ones.

As the number of methods is growing, it is important to evaluate each method's strengths and weaknesses. In addition to the simulation approach [1], [3], [8], [9], many researchers have investigated the performance of concurrency control methods by using the mathematical analysis [6], [7], [10]. For ease of analysis, a mathematical approach usually makes simplifying and impractical assumptions. Even so, the analysis is still often complicated and intricate due to having to build a queuing model, to derive the blocking ratio and the waiting time formulas, and others [10].

In this paper, the performance of locking protocols is studied mathematically based on an evaluation factor called the "average lock range (ALR)." ALR does not require one to build a queuing model and does not require the derivation of other relevant measurable parameters such as the blocking ratio, the waiting time, and the throughput; hence, ALR provides a simple way to determine the performance of either a structural or nonstructural locking-based concurrency control method such as the tree protocol or the 2PL protocol. In general, the larger an ALR is, the worse the performance is. Section II gives the definition of ALR and an analysis. In Section III, ALR is shown to correlate with the throughput by using simulation. From the simulation, the relationship between ALR and the system's throughput is verified as an inverse proportion. Finally a conclusion is made in Section IV.

II. THE DEFINITION OF ALR AND THE ANALYSIS OF LOCKING PROTOCOLS

Before giving the definition of ALR, the following assumptions associated with the operational requests of a transaction are listed.

1) A locking unit is a granule such as a disk block; it contains a number of accessible data objects.
2) For a locked granule, each access action made on it, such as a retrieval, insertion, deletion, or modification, could be viewed as one ACCESS request.
3) After making a LOCK request, a transaction should make the appropriate ACCESS request as soon as possible, but this does not mean that the locking sequence is the same as the access sequence.
4) Similar to assumption 3), a transaction should make the appropriate UNLOCK request as soon as possible when an ACCESS request is completed.
5) All operational requests have the same processing time cost since a LOCK request (an UNLOCK request) involves a disk read (write) time, and an ACCESS request involves a number of memory access time by assumption 1) and 2).

In general, the total time between a LOCK request and an UNLOCK request on a granule consists of processing time and waiting time, and the waiting time is a function of average conflict ratio and average processing time of a granule. In the model, the average conflict ratio is unpredictable; therefore, the average processing time becomes a dominated factor of the total time. Thus, the lock range of a granule $i$ could be defined as a function of the total processing time between a LOCK request and an UNLOCK request on the granule $i$. By assumption 5), a lock range may be viewed as the number of operational requests enclosed between a pair of (LOCK, UNLOCK) operations. As for the ALR, it is defined as the average number of operational requests for a lock range in a transaction. For example, the following operational request sequence of a transaction under the 2PL discipline

$$\text{LockRange} = \frac{\sum_{i=1}^{n} \text{LockRange}(i)}{n}$$

has the ALR $\frac{\sum_{i=1}^{n} \text{LockRange}(i)}{n} = \frac{(5 + 4 + 3)}{3} = 4$. The symbols $L_i$, $A_i$, and $U_i$ represent the LOCK, ACCESS, and UNLOCK requests on granule $i$.

In general, the larger an ALR is, the worse the performance is. This fact can be found from the simulation shown in Section III; when the ALR of 2PL increases from 7.00 to 74.50, the throughput decreases from 1.93 to 0.17.

A. ALR of the 2PL Protocol

Regardless of different access sequences, the ALR of a 2PL-based transaction is a linear function of $n$, where $n$ is the number of granules. Let $n$ be a transaction's number of granules. Following the principles of 2PL, the standard operational sequence for a transaction should be $L_i A_i L_i A_i \ldots L_i A_i U_i U_i \ldots U_i$. The subscripts of operational requests represent the access sequence of required granules in a transaction. For UNLOCK requests, the sequence of subscripts $i_1, i_2, \ldots, i_n$ may be different from the sequence $1, 2, \ldots, n$ for LOCK requests. It could be one of the $n!$ permutations.

Theorem 1: Let $T$ be a transaction running under the 2PL policy where $T$ requires $n$ granules. The ALR of $T$, $\text{LockRange}_T$, is $(3 \cdot n - 1)/2$.

Proof: The lock range of a granule can be expressed as $U(i) - L(i) - 1$ where the values of $U(i)$ and $L(i)$ represent the execution steps of LOCK and UNLOCK requests for the $i$th granule in $T$. Therefore, the ALR could be formulated as below:

$$\text{LockRange}_T = \sum_{i=1}^{n} (U(i) - L(i) - 1)/n$$

$$= \frac{n \cdot (5 \cdot n + 1)/2 - n^2 - n}{n}$$
E. The Workload Parameters

A. The Queueing Model

An ACCESS request is put into the granule can be accessed by the ACCESS request. At first, the tent to do the service. From the ready queue into the CC queue and await the CC server are then accessed by the TRANSACTION (BOT), LOCK, ACCESS, UNLOCK, and END-OF-TRANSACTION (EOT). When a transaction enters into the queueing model through a terminal as shown in Fig. 1, its BOT makes a starting signal for a service request and enters into the concurrency control (CC) queue. The EOT signals the completion of the transaction execution, and the transaction commits. Before the data objects can be accessed by the ACCESS request, the LOCK request must be granted by the concurrency control (CC) server. When a LOCK request is made, there are three possible outcomes. The first outcome is that the LOCK request is granted, and the data objects in the locked granule could then be accessed by the transaction. The second outcome is that the granule to be locked is unavailable, and the transaction is blocked. The blocked transaction enters into the blocked queue until the blocked granule is unlocked by the UNLOCK request, and these blocked transactions resident in the blocked queue are woken up by the UNLOCK. The third outcome is that a deadlock is detected and the transaction is chosen as the victim. After waiting for a while, called restart time, the victim enters into the ready queue. Then the victim and the newcomer are moved from the ready queue into the CC queue and await the CC server to do the service.

When the LOCK request is granted, the data objects in the locked granule can be accessed by the ACCESS request. At first, the ACCESS request is put into the object queue, and the data objects are then accessed by the object server.

B. The Workload Parameters

The fairness of the evaluation results depends heavily on the contents of the workload parameters. In the simulation, the workload parameters represent the most important three categories. The three categories are 1) transaction behavior, 2) system environment, and 3) the size of the database. These three categories have been recognized as the major influence over the behavior of a multitransaction system [7].

For the simulation, the workload parameters are divided into three categories of workload parameters are included for the completeness; they are 1) transaction-oriented parameters, 2) system environment, and 3) the size of the database. Moreover, the performance indexes are 1) throughput, and 2) conflict-ratio.

In the simulation, we have two experiments to show the relationship between the theoretical ALR and the throughput. The first one is done by changing the transaction size, while the second one is done by changing the database size.

A. The Queueing Model

Assume each transaction has the requests of BEGINNING-OF-TRANSACTION (BOT), LOCK, ACCESS, UNLOCK, and END-OF-TRANSACTION (EOT). When a transaction enters into the queueing model through a terminal as shown in Fig. 1, its BOT makes a starting signal for a service request and enters into the concurrency control (CC) queue. The EOT signals the completion of the transaction execution, and the transaction commits. Before the data objects can be accessed by the ACCESS request, the LOCK request must be granted by the concurrency control (CC) server. When a LOCK request is made, there are three possible outcomes. The first outcome is that the LOCK request is granted, and the data objects in the locked granule could then be accessed by the transaction. The second outcome is that the granule to be locked is unavailable, and the transaction is blocked. The blocked transaction enters into the blocked queue until the blocked granule is unlocked by the UNLOCK request, and these blocked transactions resident in the blocked queue are woken up by the UNLOCK. The third outcome is that a deadlock is detected and the transaction is chosen as the victim. After waiting for a while, called restart time, the victim enters into the ready queue. Then the victim and the newcomer are moved from the ready queue into the CC queue and await the CC server to do the service.

When the LOCK request is granted, the data objects in the locked granule can be accessed by the ACCESS request. At first, the ACCESS request is put into the object queue, and the data objects are then accessed by the object server.

B. ALR of the Tree Protocol

The tree protocol [5] developed by Kedem and Silberschatz is a deadlock-free locking protocol, and is applied to a database which has a hierarchical organization. A transaction following the tree protocol has a different ALR for different access sequences. According to the tree protocol’s concepts, the granules in a database are organized hierarchically, and a transaction must lock the required granules according to the hierarchical structure. Due to this structural locking, the analysis of ALR on tree protocol becomes very difficult. To simplify the problem, we assume the granules in database are organized in a complete, balanced binary tree as below:

1) The total number of granules is $N$ which may be expressed as $2^i - 1$, where $i \geq 1$.
2) For any node in the binary tree, the height of its left subtree is always equal to that of its right subtree.

For such a tree, the ALR can be obtained through a complicated and lengthy analysis. For detailed results, see [4].

III. THE SIMULATION

To make the evaluation as fair and open-minded as possible, a general and closed queueing model is designed. The queueing model can reflect the behavior of a transaction very close to a real database management system’s run-time environment. Three categories of workload parameters are included for the completeness; they are 1) transaction-oriented parameters, 2) system environment, and 3) the size of the database. Moreover, the performance indexes are 1) throughput, and 2) conflict-ratio.

In the simulation, we have two experiments to show the relationship between the theoretical ALR and the throughput. The first one is done by changing the transaction size, while the second one is done by changing the database size.

A. The Queueing Model

Assume each transaction has the requests of BEGINNING-OF-TRANSACTION (BOT), LOCK, ACCESS, UNLOCK, and END-OF-TRANSACTION (EOT). When a transaction enters into the queueing model through a terminal as shown in Fig. 1, its BOT makes a starting signal for a service request and enters into the concurrency control (CC) queue. The EOT signals the completion of the transaction execution, and the transaction commits. Before the data objects can be accessed by the ACCESS request, the LOCK request must be granted by the concurrency control (CC) server. When a LOCK request is made, there are three possible outcomes. The first outcome is that the LOCK request is granted, and the data objects in the locked granule could then be accessed by the transaction. The second outcome is that the granule to be locked is unavailable, and the transaction is blocked. The blocked transaction enters into the blocked queue until the blocked granule is unlocked by the UNLOCK request, and these blocked transactions resident in the blocked queue are woken up by the UNLOCK. The third outcome is that a deadlock is detected and the transaction is chosen as the victim. After waiting for a while, called restart time, the victim enters into the ready queue. Then the victim and the newcomer are moved from the ready queue into the CC queue and await the CC server to do the service.

When the LOCK request is granted, the data objects in the locked granule can be accessed by the ACCESS request. At first, the ACCESS request is put into the object queue, and the data objects are then accessed by the object server.

B. The Workload Parameters

The fairness of the evaluation results depends heavily on the contents of the workload parameters. In the simulation, the workload parameters represent the most important three categories. The three categories are 1) transaction behavior, 2) system environment, and 3) the size of the database. These three categories have been recognized as the major influence over the behavior of a multitransaction system [7].

The parameters used to describe the transaction behavior are 1) the transaction size and 2) the transaction’s mean time of arrival. The parameters used to describe the system’s environment are 1) the level of multiprogramming and 2) the number of terminals linked with the underlying operating system.

The tran-size parameter represents the number of data objects accessed by a transaction. This number is an integer, and it is the mean value of an exponential distribution. The parameter tran-arr-time represents the arrival interval between a newcomer and the latest committed transaction at a terminal. The time is also the mean value of an exponential distribution. The level of multiprogramming has directly affected the performance of a transaction system; hence, the parameter lev-mul-prg represents the level of multiprogramming (i.e., the maximum number of active transactions in the system) and the parameter no-of-terms describes the number of terminals linked with the underlying operating system. The parameter db-size indicates the number of data objects in the database.

C. Performance Indexes

The factors to be evaluated in the simulation are throughput and conflict ratio. The throughput measures the number of committed transactions per unit of time. The conflict ratio estimates the number of conflict locking requests against the total number of locking requests. The two indexes truly reflect strengths and weaknesses on the degree of concurrency for each protocol, since a good concurrency control method shall have a higher throughput, lower conflict ratio. In fact, the throughput is more relevant than the conflict ratio as a performance index. Moreover, the occurrence of deadlock in 2PL is also shown for reference only.

D. Performance Experiments

The simulation was done on an IBM 4341-M02 machine with the simulation language GPSS/H. Each test took about 1 800 000
Fig. 2 (a) ALR in the average case from formula (total granules = 1023). (b) Throughput in the average case from simulation (total granules = 1023). (c) Conflict ratio in the average case from simulation (total granules = 1023).

TABLE I

<table>
<thead>
<tr>
<th>WORKLOAD PARAMETERS FOR EXPERIMENT 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction behavior:</td>
</tr>
<tr>
<td>tran-size:</td>
</tr>
<tr>
<td>5, 10, 20, 30, 40, and 50 data objects</td>
</tr>
<tr>
<td>tran-arr-time:</td>
</tr>
<tr>
<td>5 s</td>
</tr>
<tr>
<td>system environment:</td>
</tr>
<tr>
<td>lev-mul-prg:</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>no-of-terms:</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>database:</td>
</tr>
<tr>
<td>db-size:</td>
</tr>
<tr>
<td>1023 data objects</td>
</tr>
</tbody>
</table>

*A transaction accessing more than 10 data objects can be viewed as a large one.

simulation time units (30 min). In our simulation for the tree protocol, we assume that the structure of the database is a balanced binary tree.

1) Experiment 1: Transaction Size: This experiment observes 1) the throughput and 2) the conflict ratio of each protocol under different transaction sizes. The workload parameters for this experiment are listed in Table 1.

Fig. 2(a) shows the computed ALR of the 2PL and tree protocol, according to the formula derived in the previous section (the formula for the tree protocol is omitted). Fig. 2(b) and (c) show the throughput and the conflict ratio of them, respectively. From the experiment, we can have the following observations.

1) The larger the computed ALR is, the lower the throughput is in the simulation.
2) For the 2PL protocol, the conflict ratio is gradually higher when the computed ALR is getting larger. On the contrary, the conflict ratio is getting lower for the tree protocol. The reason is that although more locking requests are made when transaction size increases, the increased number of conflict locking requests is far less than that of total locking requests.

2) Experiment 2: Database Size: This experiment observes 1) the throughput of each protocol and 2) the number of restarts for the 2PL under different database sizes. The workload parameters for this experiment are listed in Table II.
Fig. 3 (a) ALR in the average case from formula (accessed granules = 10). (b) Throughput in the average case from simulation (accessed granules = 10). (c) Restart in the average case from simulation (accessed granules = 10).

TABLE II

<table>
<thead>
<tr>
<th>WORKLOAD PARAMETERS FOR EXPERIMENT 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction behavior:</td>
</tr>
<tr>
<td>tran-size 10 data objects</td>
</tr>
<tr>
<td>tran-arr-time 5 s</td>
</tr>
<tr>
<td>system environment:</td>
</tr>
<tr>
<td>lev-mul-prg 5</td>
</tr>
<tr>
<td>no-of-terms 10</td>
</tr>
<tr>
<td>database:</td>
</tr>
<tr>
<td>db-size 15, 31, 63, 127, 255, 511 data objects</td>
</tr>
</tbody>
</table>

1) For the tree protocol, the larger the computed ALR is, the lower the throughput is in the simulation.
2) For the 2PL protocol, although the computed ALR is not varied, the throughput decreases gradually when the database size is much smaller. The reason is that there are deadlocks occurring in the environment and the deadlock factor is not involved in the ALR.

IV. CONCLUSIONS

To verify the mathematical results, a simulation is done for the 2PL and tree protocol in the average case. From these figures, we have observed that the relationship between the ALR and the system’s throughput is verified as an inverse proportion, with the only exception that the database size is relatively small and deadlock occurs frequently in 2PL. Therefore, the definition of ALR provides a very general evaluation factor, and it can be used to measure the strengths and weaknesses of any locking based concurrency control method. Basically, the new methodology has the following advantages.
1) ALR can be used to evaluate any locking protocol since it requires no queueing model. In general, one particular queueing model can only describe one locking protocol unless a general queueing model is proposed; however, it is very difficult to build a general queueing model.

2) Due to ALR’s simplicity property, a tree-protocol based transaction can be mathematically studied, and such a study cannot be analyzed under a conventional queueing model since a queueing model based on the tree protocol is very difficult to build.

3) The new methodology can analyze the best, the worst, and the average cases of a transaction (see [4] for a structural or a nonstructural locking protocol analysis). Such a merit cannot be achieved through the queueing model based analysis. In a queueing model based analysis, the results can only be viewed as an average case because a queueing model based analysis could generate much detailed information such as the blocking ratio, the waiting time, the throughput, and the response time, while this information is very difficult to derive in the best case (or the worst case).

In short, the following guideline is used to decide whether an ALR or a queueing model based analysis should be applied. If several locking protocols are compared to each other, the ALR can be used to determine the best one easily, while a queueing model based analysis should be adopted if detailed information within a locking protocol is required.

REFERENCES