MULTIGRANULARITY LOCKING IN MULTIPLE JOB CLASSES TRANSACTION PROCESSING SYSTEM

Shan-hoi Ng and Sheung-lun Hung

Department of Computer Science
City Polytechnic of Hong Kong
HONG KONG
Email: CSMAXNG@cphkvx.bitnet

Abstract

The conditions of when to apply fine and coarse granularity to different kinds of transaction are well understood. However, it is not very clear how multiple job classes using different lock granularities affect each other. This study aims at exploring the impact of multigranularity locking on the performance of multiple job classes transaction processing system which is common in multiuser database system. There are two key findings in the study. Firstly, lock granularity adopted by identical job classes should not differ from each other by a factor of more than 20; otherwise, serious data contention may result. Secondly, short job class transactions are generally benefited when its level of granularity is similar to that of the long job class since this will reduce the additional lock overhead and data contention which are induced by multigranularity locking.

1. Introduction

Locking is one of the methods used to solve the data consistency problems in an environment of concurrent multiple access to data. The term, granularity, refers to the size of a lockable data unit. There are three factors affecting the performance of different lock granularities. They are the lock overhead, data contention and resource contention. According to [Bern87], the finer the lock granularity adopted, the more the lock overhead involved and the higher is the degree of both the data contention and the resource contention. However, the multiprogramming level will also increase. It should be beneficial to short transactions. For long transactions, fine granularity does not help much since the portion of database to be locked will remain almost the same.

In [Ries77] and [Ries79], the authors concluded that coarse granularity is generally preferred except when transactions access a small part of the database randomly, in which case fine granularity is desired. [Dand91] further showed that when a system is lightly loaded, fine granularity should be used if transactions access the database randomly. However, when transactions access the database sequentially, coarse granularity is preferable for large transactions. If transactions are short or of mixed size, fine granularity is a better choice. On the other hand, if the system is heavily loaded, coarse granularity should be used.

The previous studies concentrated mostly on the effect of a single granularity on a transaction type. The conditions of when to apply different lock granularity on an individual type of transaction are well examined. However, in multiuser database system, jobs are divided into classes such as batch and query. These job classes may have different properties, such as processing demand, job length, etc. Purely applying the results from previous studies may not be appropriate since the system is complex and the interference from different classes due to the use of different lock granularities is not well examined. There are very few studies and limited knowledge on the effect of multigranularity locking in such a system. In the study, we concentrate on the effect of lock overhead and data contention due to the use of multigranularity locking in the multiple job classes system.

Initially, we observe how the effect of the lock granularity assignment of one class affects the performance of another which differs in lock granularity only. The control on multiprogramming level (MPL) is applied to vary the system workload. Then, two classes representing long and short transactions are designed respectively. They run with fine and coarse lock granularity in parallel according to some experimental arrangements. Primary result shows that the system performance is better when the lock granularity difference of both classes is small. This result suggests that the coexistence of different lock granularity may be unnecessary. Then, we further extend the experiment by varying the data processing demand and class proportions in the system. Similar results are obtained through these experiments which further confirm our initial findings.

In this paper, the system model and details of lock granularity handling mechanism for the experiments are given in section 2. Major system parameters and performance measurements are given in section 3. In section 4, experimental results and interpretations are presented. Finally, conclusions are drawn in section 5.

2. The Testbed Design

The prototype of a multiple job classes transaction processing system, composing mainly of a lock manager and a scheduler, is constructed for the study. A transaction spooler has also been implemented so that a number of job classes can submit jobs to the system in parallel. Besides maintaining transaction lock information, the lock manager supports multigranularity locking and its related activities.

The Lock Manager

A global data lock tree is designed to support multigranularity locking. Each tree node is a data granule which may further represent a number of finer granules. The major components of each tree node are execution queue, wait queue and next granular level pointer. The execution queue registers those granted transaction lock accesses which are held by some not-yet-finished transactions. The wait queue is to register those not yet granted transaction lock request. For the next granular level pointer, it points to a subtree and is null for the finest level of granule. The subtree consists of descendant nodes of the granule. Lock compatibility table and conversion table are also designed to support multiple modes of lock and lock escalation, as shown in Table 1 and Table 2. The lock types supported by this prototype are intention read, intention write, read-intention-write, read and write.

The lock hierarchy is level by level. To obtain a granule lock at level i, the ancestor granules at levels that are on top of level i will be accessed with the intentional type of such lock in top-down manner. To grant a lock access on a particular level's granule, the lock manger must check the execution queue of the granule for whether the granted access type(s) is/are compatible to the requesting type. The lock compatibility modes are shown in Table 1. If it is incompatible, the lock manager can either report lock access failed or attach the transaction's request onto the wait queue to wait for the granule access right being granted. The action chosen is determined by the scheduler according to the scheduling rule adopted. The transaction model for the above is drawn in Figure 1. Currently, the system allows a maximum of four levels of locks. Each level can allocate any number of granule. The level number chosen can be determined by specifying required parameters to the system.

The Scheduler

The scheduler contains a scheduling rule file in which the scheduling rules governing system behaviour are stored. Upon the reply from the lock manager, appropriate scheduling actions can be taken. The scheduler has another file which contains scheduling parameters such as transaction priority, wait-check interval, ..etc. which are important runtime information. In the experiment, General Waiting algorithm, which is used extensively in most commercial database systems, is chosen. Below is the description of the algorithm.

- "When a transaction T requests a lock access which results in data conflict, T will be blocked in the wait queue until the lock is granted; otherwise, it can obtain the lock and carry out further action(s)."

The Transaction Spooler

For transaction spooling, the system maintains a constant multiprogramming level by ensuring that a new transaction enters the system only after another transaction of the same class has left the system. Each transaction is given a think time delay before entering the central subsystem. The multiprogramming level is exactly equal to the total number of users in each job class.

Besides, the Dynamic Two Phase Locking (D2PL) has been chosen as the concurrency control protocol for the study. In case of a transaction being aborted, a conflict avoidance delay will be given to the aborted transaction before restarting it so as to minimize the chance of conflict again.

3. Workload Parameters and Performance Measures

There are four levels of lock granularity in the model. They represent database, area, file and record from level 1 to level 4 respectively. In the experiment, the size of each level's granule is:

Lock Level i.e. Granularity	No. of Granule
Level 1 granules in system	5
Level 2 granules per Level 1 granule	10
Level 3 granules per Level 2 granule	10
Level 4 granules per Level 3 granule	2

The resultant ratios between Level 1 : Level 2 : Level 3 : Level 4 are thus 1 : 10 : 100 : 200. The above lock granularity settings should be able to provide a fine and significant change of granularity and are reasonable setting for the experiment.

Each lock operation is divided into three components. They are the lock set operation, the data granule processing and the lock unset operation. In each lock set operation, a lock is set level by level. The lock unset operation is performed after a transaction commits or aborts. The individual lock step workload operational characteristics are:

Metrics	**Measure**
Mean lock set time (per lock operation per level)	0.01 sec
Mean lock unset time (per lock operation)	0.01 sec
Mean CPU demand (per complete data granule access[&])	0.03 sec
Mean I/O demand (per complete data granule access)	0.05 sec
Mean data processing delay[@] (per data granule)	0.01 sec

Besides, a transaction can issue a data request once on a particular data granule. As a result, no lock conversion or escalation is done. Deadlock due to lock conversion and/or escalation can be neglected.

In the experiments, the think time is exponentially distributed with a mean of 10 seconds. There are two classes of transactions in the system. In the first experiment (section 4.1), both classes are required to access 10 level 4 granules. In the second experiment (section 4.2), two job classes (long and short) with different transaction length are designed. Transaction length is defined as the number of data granule accesses required to complete a transaction. The short job class transactions will access 5 level 4 granules whereas the long job class transactions will access 15. For the long job class transactions, the data access pattern is sequential whereas for the short job class it is random. This should be a reasonable setting since queries for long range of data records are usually sequential accesses [Rodr76]. For short job class, since the number of data accesses is small, random access should be a fair assumption. The lock type issued by transactions of any class is in exclusive mode.

[&]Complete Data Granule Access = Lock set operation + Data Processing + Lock unset operation
[@]The time that does not take up any CPU or I/O Time

Throughout the sets of experiment, the primary performance measure is the system throughput which is defined as the number of committed transactions per fixed period of time.

4. Measurement Results and Interpretation

In the first section, lock granularity interference from identical job classes is investigated. Results from long and short job classes are presented in the next section.

4.1 Impact of multigranularity locking in identical job classes system

The increase in data contention can be caused by the increase in system workload, i.e. increase in MPL. In the experiments, two identical job classes are run in parallel with equal proportion. The only difference between them is the level of granularity. The aim is to observe how two identical job classes affect each other as lock granularities are varied under various degrees of data contention.

As the neighbouring class using coarse granularity lock (i.e. level 1), transaction throughput of the class using fine granularity (i.e. level 4) will experience a high degradation of performance. The effect is increasingly obvious as MPL increase. This can be observed from Figure 2. The substantial influence on the neighbouring class is due to the critical alleviation in the degree of conflict at level 1. Such alleviation has two causes. One is the large number of accesses at level 1 due to the class with level 4 granularity. Another is the lock hierarchy which starts from coarse to fine, i.e. from level 1 to level 4. Therefore, almost all of the data lock conflicts will first occur at level 1 and cause many transactions to be blocked at this level. The overall effect will intensify the degree of data contention and degrade the overall system throughput.

On the other hand, when the two job classes using comparable level of granularity, i.e. within the ratio of about 10 to 20, the throughput is not affected so severely. This is shown in Figure 3. It is expected that the number of conflicts and the degree of data contention are minimized when both classes use similar lock granularity.

As a whole, in a low data contention environment, the ratio difference between lock granule sizes of identical classes is not of primary importance to the classes' performance. In a high data contention environment, such difference should not exceed 20.

4.2 Impact of multigranularity locking in system with classes of different job size

To examine the effect of multigranularity lock in system with long and short job classes, the following experimental arrangements are being run in turn for each test case.

Run	Long job class	Short job class
1	level 2	level 4
2	level 2	level 2
3	level 4	level 2
4	level 4	level 4

In the tests, three common situations in a multiple job classes environment are chosen and listed below.

1. *Classes with different transaction length*
2. *Short job class under the influence of increasing processing demand on long job class*
3. *Different proportion of long and short job classes in system*

There are two parameters which are responsible for the modelling of the above situations along with the transaction length. They are the processing demand and the proportion of job classes in the system. Note that in observations 1 and 2, long and short job classes are run in parallel with equal proportion in the system.

Observation 1

In the experiment, both classes differ in transaction length only. Results are shown in Figures 4 and 5. It is generally valid that long job class should run in coarse granularity; otherwise, severe performance penalty will result due to the lock overhead and data contention induced. The short job class introduces minimal interference to the long job class. It can run at coarse or fine granule level.

Observation 2

In this experiment, the long job class' data processing demand per data item is varied from 1 to 3 times of that of the short job class. The aim of the experiment is to observe how the level of lock granularity affects the performance of the short job class due to the increase in processing demand of the long job class.

In general, as processing demand of long job class increases, the throughput of short job class will decrease since more system resources are utilized by the long job class. However, the performance degradation is more serious when short job class operates at a level of granularity different from the long job class. Nevertheless, the transaction's lock waiting time is more lengthy when the short job class operates at a higher level of granularity, i.e. coarser granularity, than the long job class. It causes a heavy degradation of performance, as shown in Figures 6 and 7.

The above observations can be explained as follow. When the long job class operates at level 4 granularity, it represents an environment in which short job class transactions will experience a delay not only due to the increase in data processing demand of long job class but also lock overhead. Once short job class uses any level of lock granularity above level 4, it will experience heavy conflict condition at such level since the number of granule is relatively less. As a result, the transaction's lock waiting time will be longer and lengthens the transaction's response time.

To overcome the influence of long job class, short job class should keep the same level of lock granularity as the long one.

Observation 3

The objective of this experiment is to investigate the effect of multigranularity locking to long and short job classes which have different proportions in system. The proportions of long job class and short job class transactions in the system are varied in the ratios of 8:2, 6:4, 4:6 and 2:8 respectively. Three MPL settings are tested for each of these proportions. Results are shown in Figures 8 and 9.

No significant increase in data contention due to the change of different proportion of long and short job classes occurs in the system when long job class uses level 2 lock granularity. However, as the long job class uses level 4 lock granularity, data contention arises. It is more serious when the short job class uses level 2 lock granularity under such a situation. The system performance is improved when the proportion of long job class is reduced in the system.

As a whole, when data contention is serious, the long job class may choose to change its lock granularity to a coarser level or reduce its proportion in the system. For short job class, it is generally benefited when its level of lock granularity is similar to that of the long job class.

5. Conclusions

In the study, we have examined the impact of multigranularity locking in a multiple job classes transaction processing system. Further understanding of the interference between different granularities of locking is obtained through studying them under different situations. For multiple job classes with single job type, the difference in granularity between each class should be as minimal as possible. For long and short job classes, the short job class can be highly influenced by the long job class. In the test cases, when the data processing demand per data item of the long job class is higher than the short job class, it is undesirable for the lock granularity of short job class to be different from the long one. Same conclusion can be drawn when long job class is the major cause of high data contention in the system. In the above cases, system performance can be improved as the granularity of the short job class is set close to the long one. Further study of the impact of data placement should be carried out to obtain a better understanding of this issue.

References

[Bern87] P.A. Bernstein, V. Hadzilacos and N. Goodman, "Concurrency Control and Recovery in Database System", Addison-Wesley, 1987.

[Chri86] Christos Papadimitriou, "The Theory of Database Concurrency Control", Computer Science Press, 1986.

[Dand91] S. Dandamudi and S. Au, "Locking Granularity in Multiprocessor Database Systems", Proceedings of the Seventh International Conference on Data Engineering, 1991, pp.268-277.

[Gray91] Jim Gray, "The Benchmark Handbook for database and transaction processing systems", Morgan Kaufmann Publishers.

[Ries77] D.R. Ries and M. Stonebraker, "Effects of Locking Granularity in a Database Management Systems", ACM Transactions on Database Systems, Vol. 2, No. 3, September 1977, pp.233-246.

[Ries79] D.R. Ries and M. Stonebraker, "Locking Granularity Revisited", ACM Transactions on Database Systems", Vol. 4, No. 2, June 1979, pp.210-227.

[Rodr76] J. Rodriquez-Rosell, "Empirical Data Reference Behaviour in Data Base Systems", Computer, Vol. 9 No. 11, November 1976, pp. 9-13.

[Tay85] Y.C. Tay, Nathan Goodman and Rajan Sufi, "Locking Performance in Centralized Databases", ACM Transactions on Database Systems, Vol. 10, No. 4, Dec. 1985, pp. 415-462.

	R	**W**	**A**	**r**	**w**
R	1	0	0	1	0
W	0	0	0	0	0
A	0	0	0	1	0
r	1	0	1	1	1
w	0	0	0	1	1

Table 1. Lock Compatibility Table (0 - incompatible, 1 - compatible, R - Read, W- Write, A - Read Intention Write, r - Intention Read, w - Intention Write)

	Present Lock Type				
Intended Lock Type	**R**	**W**	**A**	**r**	**w**
R	R	A	A	R	w
W	W	W	W	W	W
A	A	W	A	A	A
r	R	W	A	r	w
w	w	W	A	w	w

Table 2. Lock Conversion Table (R - Read, W- Write, A - Read Intention Write, r - Intention Read, w - Intention Write)

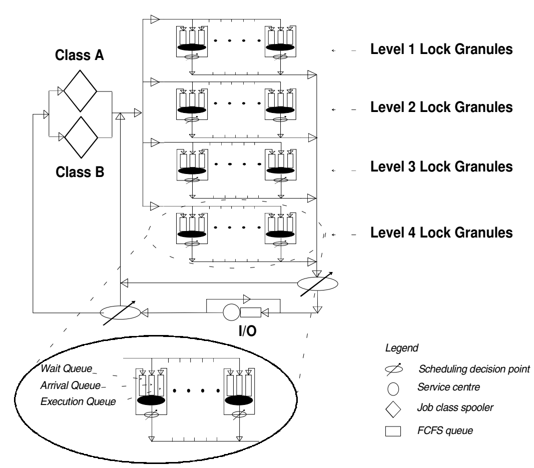

Figure 1. Multigranularity lock prototype represented in terms of a transaction model

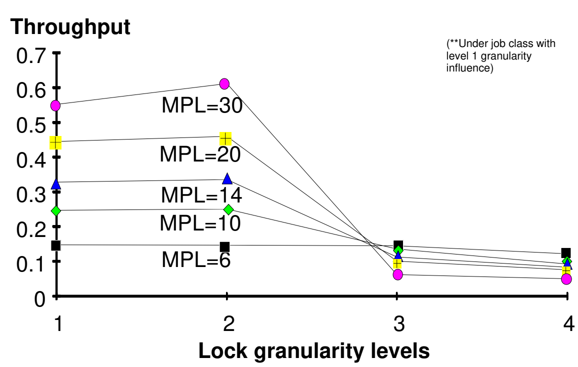

Figure 2. Single class transaction throughput under the influence of another identical job class with granularity at level 1

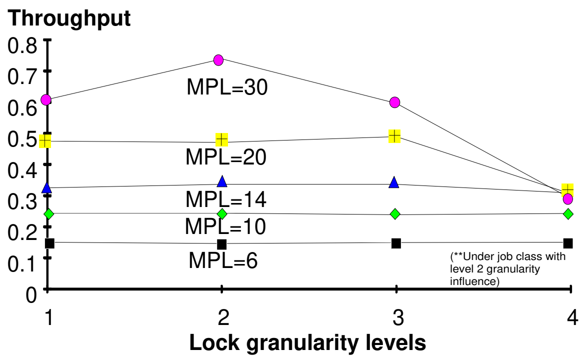

Figure 3. Single class transaction throughput under the influence of another identical job class with granularity at level 2

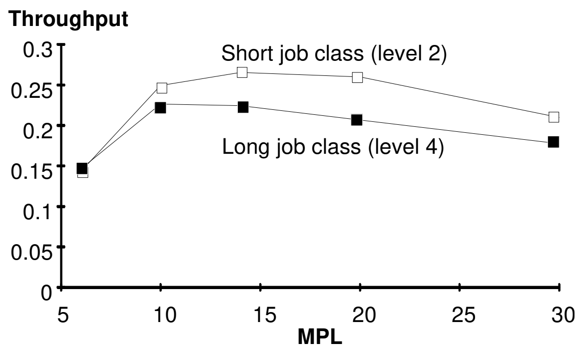

Figure 4. Throughput of long and short job classes operating at level 4 and level 2 respectively.

Figure 5. Throughput of long and short job classes operating at level 2 and level 4 respectively.

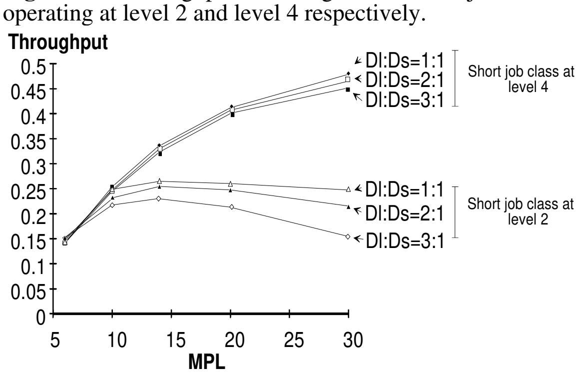

Figure 6. Throughput of short job class under the influence of long job class operating at level 4 (Note: Dl = Long job class data processing demand, Ds = Short job class data processing demand)

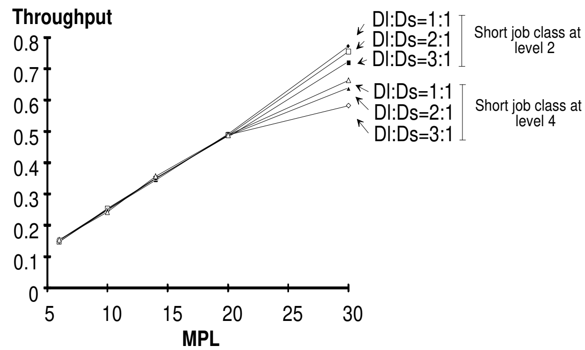

Figure 7. Throughput of short job class under the influence of long job class operating at level 2 (Note: Dl = Long job class data processing demand, Ds = Short job class data processing demand)

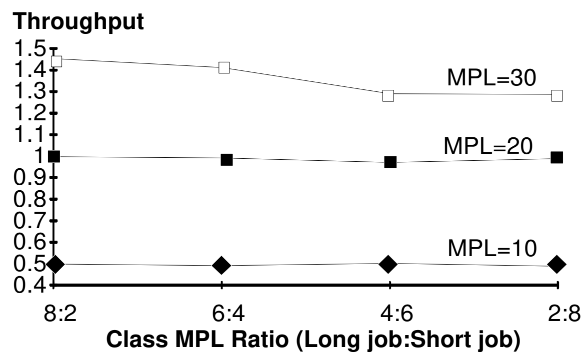

Figure 8. Effect of job class MPL ratio to system throughput as long job class using level 2's, short job class using level 4's lock granularity

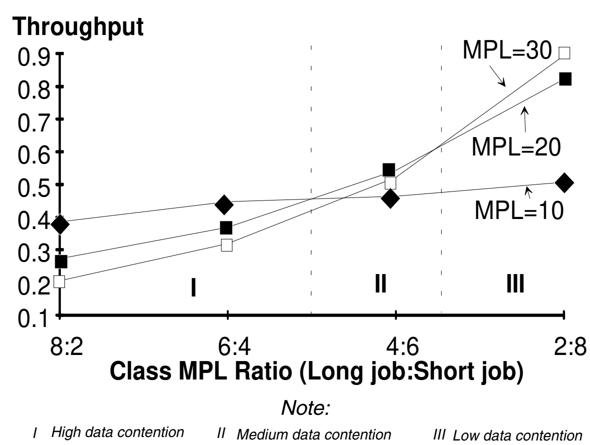

Figure 9. Effect of job class MPL ratio to system throughput as long job class using level 4's , short job class using level 2's lock granularity.