Scheduling Real-time Multi-Granularity Locks in Object-Oriented Database Systems

QIN XIAO*, PANG LI-PING*, LIU JIE* and LI SHENG-LI*

*Huazhong University of Sci. & Tech., Department of Computer Science, Wuhan Hubei, 430074, P.R.China

Abstract. This paper presents a real-time multi-granularity locking model(RTMGL) for concurrency control in object-oriented database system and proposes real-time scheduling algorithms for RTMGL are proposed. As shown in the experiments, real-time scheduling algorithms behave better than non real-time algorithms in real-time environment.

Key words. Real-time scheduling; multi-granularity lock; concurrency control; object-oriented database

1. INTRODUCTION

Real-time database systems(RTDB) handle transaction with time constrains. Real-time transactions' execution must complete by its deadline, and the data it reads must be sufficiently current(Liu, 1988). Recently, object-oriented database system (OODB) has become popular and many papers have discussed about what object-orientation means and what an object-oriented database is(Kim, 1990). A new transaction model for object-oriented database system was proposed by Shah et al. (1994). Object-oriented real-time database system (OORTDB) is becoming an interesting research area, for it is not only as flexible as OODB is, but also has real-time properties as RTDB has. In this paper, we study a real-time muti-granularity locking model for the concurrency control(CC) in OORTDB. A lot of papers have explored the CC in RTDB (Son, 1989). The CC in OODB also has been widely examined and the CC model for OORTDB presented in this paper is an extended model of the MGL model proposed by Lee et al. (1996).

2. EXTENDED MGL MODEL

For MGL model and locking scheduling do not take any timing constrains into consideration, an extended model is proposed formally.

Definition 1: MGL model is defined as a triple, MGL = (LM, CM, SCH), where LM is a set of lock modes, CM is a compatibility matrix, SCH is the scheduling strategy. CM is defined as:CM = (cm_{ij}), ι, j symbol 206 \f "Symbol" \s 10.5∈ } LM where $cm_{ij} = cm_{ji}$ and cm_{ij} = N (symbol 105 \f "MS LineDraw" \s 10.5ι is incompatible with j), cm_{ij} = Y (symbol 105 \f "MS LineDraw" \s 10.5ι is compatible with j); Function cmp(lk_1, lk_2) is used to check whether lk_1 and lk_2 are compatible or not, if they are compatible cmp will return Y, otherwise N is returned.

Definition 2: Object-Oriented database is as a pair, OODB = (AT, OS), where AT is a set of active transactions, OS is a set of objects in the database.

Definition 3: A lock request is a tuple, lr = (t, o, lk, dl), where t symbol 206 \f "Symbol" \s 10.5∈ } AT, o symbol 206 \f "Symbol" \s 10.5∈ } OS, lk symbol 206 \f "Symbol" \s 10.5∈ } LM, lr is required by active transaction t, and o is an object that is accessed by t, lk is the mode of the lock, dl is the deadline of this lock request.

Definition 4: Transaction in AT is abstracted as a duple, t = (st, LL), where t symbol 206 \f "Symbol" \s 10.5∈ } AT, st is the status of transaction, and LL is a list of the lock requests, all requests in LL are required by t; st = BLK|RUN, if a lr symbol 206 \f "Symbol" \s 10.5∈ } t.LL can not be served, t will be blocked, otherwise t' s status is RUN.

Definition 5: Object is modeled as: o = (ST, TC, GB, RQ, DQ), where osymbol 206 \f "Symbol" \s 10.5∈ }OS, GB is a granted buffer, and all lock requests in GB are mutually compatible. RQ and DQ denote request queue and delay queue; ST = SRQ|SDQ, where SRQ means that RQ is being served and SDQ indicates that delay queue is being scheduled. TC is a timer used to determine whether the it' s the time to schedule DQ.

Definition 6: Compatible requests set of RQ is defined as: CRS-RQ(o) = {rq | symbol 34 \f "Symbol" \s 10.5∀r symbol 206 \f "Symbol" \s 10.5∈ } o.GB (cmp(r.lk, rq.lk) = Y), rq symbol 206 \f "Symbol" \s 10.5∈ } o.RQ }. Compatible requests set of DQ is defined as: CRS-DQ(o) = {rq | symbol 34 \f "Symbol" \s 10.5∀r symbol 206 \f "Symbol" \s 10.5∈ } o.GB (cmp(r.lk, rq.lk) = Y), rq symbol 206 \f "Symbol" \s 10.5∈ } o.DQ }.

3. LOCKING SCHEDULING

FCFS is a traditional policy that is simple and fair for requests. Single queue (FCFSS) and dual queue scheduling(FCFSD) are proposed by Qin et al.

(1998). FCFSD performs better than FCFSS. In FCFSD algorithm, requests for a given object are appended to RQ first, if the request in the front of RQ is incompatible with GB, it will be moved from RQ into DQ, the scheduler will switch to DQ periodically.

Lock requests are ordered in RQ or DQ according to their deadlines, and the requests with the earliest deadline are granted first. The locking algorithm of EDF is the same as that of FCFS(Qin, 1998). As the space limitation, we only present EDFS and FEDFS.

EDFS unlock(lr) /* EDF for Single Queue */

(1) o = lr.o; Delete lr from o.GB;
(2) If CRS-RQ(o)symbol 185 \f "Symbol" \s 10.5≠}symbol 198 \f "Symbol" \s 10.5∅} then do next step; else do (6)
(3) Find rsymbol 206 \f "Symbol" \s 10.5∈}CRS-RQ(o) where symbol 34 \f "Symbol" \s 10.5∀rsymbol 162 \f "Symbol" \s 10.5′}symbol 206 \f "Symbol" \s 10.5∈}CRS-RQ(o) (r.dl symbol 163 \f "Symbol" \s 10.5≤} rsymbol 162 \f "Symbol" \s 10.5′}.dl);
(4) Move r from o.RQ into o.GB; r.t.st=RUN;
(5) Recalculate CRS-RQ(o); repeat (2);
(6) End of EDFS unlock;

The feasible earliest deadline first (FEDF) algorithm is similar to EDF except that the request chosen by FEDF not only has the earliest deadline but also has the feasible deadline. If the deadline of the request lr can be met, this deadline is feasible, that is: lr.dl is feasible symbol 171 \f "Symbol" \s 10.5↔} lr.dl symbol 179 \f "Symbol" \s 10.5≥} t + hold(lr), where t denotes current time, hold(lr) means how long object lr.o will be hold by the lock request.

Definition 7: CRS-RQ has two subsets, feasible requests set(FRS-RQ) and unfeasible requests set(UFS-RQ), which defined as: FRS-RQ(o) = {rq | rq.dl is feasible, rq symbol 206 \f "Symbol" \s 10.5∈} CRS-RQ(o) }, UFS-RQ(o) = CRS-RQ(o) - FRS-RQ(o) = {rq | rq symbol 206 \f "Symbol" \s 10.5∈} CRS-RQ(o) and rq symbol 207 \f "Symbol" \s 10.5∉} FRS-RQ(o) }.

We can similarly define CRS-DQ, FRS-DQ and UFS-DQ for delay queue. We also design two kinds of FEDF. In FEDFS, if the FRS-RQ is not empty, a request with the feasible earliest deadline will be chosen from FRS-RQ, when FRS-RQ is empty, FCFS algorithm will be used to schedule set UFS-RQ. Algorithm FEDFD use the same method to schedule both RQ and DQ.

FEDFS unlock(lr) /* FEDF for Single Queue */

(1) o = lr.o; Delete lr from o.GB;
(2) If FRS-RQ(o) symbol 185 \f "Symbol" \s 10.5≠}symbol 198 \f "Symbol" \s 10.5∅} then do next step; else do (5)
(3) Find rsymbol 206 \f "Symbol" \s 10.5∈}FRS-RQ(o) where symbol 34 \f "Symbol" \s 10.5∀rsymbol 162 \f "Symbol" \s 10.5′}symbol 206 \f "Symbol" \s 10.5∈}FRS-RQ(o) (r.dl symbol 163 \f "Symbol" \s 10.5≤} rsymbol 162 \f "Symbol" \s 10.5′}.dl).
(4) Move r from o.RQ into o.GB; r.t.st=RUN; Recalculate FRS-RQ(o); repeat (2);
(5) If UFS-RQ(o) symbol 185 \f "Symbol" \s 10.5≠}symbol 198 \f "Symbol" \s 10.5∅} then use FCFS to schedule it.
(6) End of FEDFS unlock;

4. SIMULATION RESULT

We assume the lock requests are generated according to a poisson distribution with mean inter-arrival 1/symbol 108 \f "Symbol" \s 10.5λ, and the hold time is modeled by using exponential distribution with mean serving rate symbol 109 \f "Symbol" \s 10.5μ. The lock modes lk is modeled by using uniform distribution, P{lk = x} = 1/16, xsymbol 206 \f "Symbol" \s 10.5∈}[1,16]. Switching period is 10ms, length of DQ = symbol 165 \f "Symbol" \s 10.5∞}. Slack-time is generated uniformly from the range [10ms, 50ms]. The deadline is defined as: Deadline = Arrival - time + hold + slack-time. Lock requests are independent of each other. Percentage of missed deadlines(PMD) is presented as: PMD% = Missed requests * 100 / Total requests.

We only exam PMD because of the space limitation. For single queue scheduling, FEDFS is the best algorithm, and FCFSS is the worst one (symbol 108 \f "Symbol" \s 10.5λ = 50No. /Sec., FEDFS misses 27.2% and 51.5% fewer deadlines than EDFS and FCFSS respectively). For dual queue scheduling, the performance of FCFSD and EDFD are nearly the same, FEDFD is the best of three(symbol 108 \f "Symbol" \s 10.5λ = 60No./Sec., FEDFD misses 19.6% fewer deadlines than EDFD and FCFSD). Among all the algorithms, FEDFS has the fewest PMD across all load settings, especially when arrival rate of requests is high(symbol 108 \f "Symbol" \s 10.5λ = 70No./Sec., FEDFS misses 32.4%, 51.1% and 47.5% fewer deadlines than EDFS, EDFD and FEDFD respectively).

5. CONCLUSION

This paper proposes a RTMGL model and two real-time locking scheduling algorithms. Simulation

experiment shows that single queue is better than dual queue in real-time environments. We can employ FEDFS in MGL for OORTDB to make more real-time transactions meet their deadlines, for FEDFS is the best algorithm.

6. REFERENCES

J.W.S.Liu, K.J.Lin and X.Song. (1988). "A position paper on scheduling hard real-time transactions," Proc. IEEE Workshop Real-Time Operating Systems and Software.

W.Kim. (1990). "Object-oriented database: Definition and research directions," IEEE Trans. Knowledge and Data Engineering, Vol. 2, No. 3, pp.327-340, Sept. 1990.

Shah.P and Wong.J. (1994). "Transaction Management in an object-oriented Data Base System," Journal of Systems and Software, Vol. 34, No. 2, pp.115-124.

S.H.Son and R.P.Cook. (1989) "Scheduling and consistency in real-time database systems," Proc. Sixth IEEE Workshop Real-Time Operating Systems and Software, pp.42-45.

S.Y.Lee and R.L.Liou. (1996). "A multi-Granularity Locking Model for Concurrency Control in Object-Oriented Database Systems," IEEE Trans. On Knowledge and Data Engineering, Vol.8, No.1, pp144-156.

Qin Xiao, Han Zong-Fen and Pang Li-ping. (1998) "Study of the Model for Dual Queue Scheduling of Multi-Granularity Lock in Objected-Oriented Database" J. of HUST. ready to appear.