5M-3

Concurrency Control of Bulk Access Transactions: a Performance Evaluation

OHMORI, Tadashi TANAKA, Hidehiko

Department of Electrical Engineering, The University of Tokyo

1 Introduction

A *Bulk Access Transaction* (BAT) refers to a transaction which accesses large bulk of data. When scheduling BATs, the high contentions of both data and resources degrade the performance largely. This paper describes the performance evaluation of the two new schedulers of BATs, which we have proposed in [Ohm89].

2 Simulation model

We test the performance of both Chain-WTPG scheduler (Chain) and K-conflict WTPG scheduler of $K = 2$ (K2), in [Ohm89]. Furthermore we test the performance of Cautious Two Phase Lock (C2PL), Atomic Static Lock (ASL) and NODC (no data contention). NODC grants any lock at any time. C2PL is a variant of the two phase lock in the cautious schedulers [Nis87].

Figure 1 displays the simulation-model of a shared nothing database machine. Its parameters are listed in Table 1.

We assume the range-partitioning of relations. Partitions are located at data-nodes such that *data-node id* = (*partition-id modulo NumNodes*). Among *NumParts* partitions, the first *NumHots* partitions are designated as a hot set if necessary. A transaction is modeled by a sequence of read/write steps. `r(P:C)` (or `w(P:C)`) refers to the read (or write) step to a partition `P` with its I/O demand `C`. `C` is the number of *objects* the step accesses. An *object* is the unit of data in the bulk-data processing, such as a cylinder of a disk in the sequential scan.

In each experiment, a transaction is notated by *Tr-pattern*: *step1* →...→ *stepN*. This means the sequential execution of steps from *step1* to *stepN*. A read-step (or a write-step) requires a shared (S) (or exclusive (X)) lock, and the locks are held until the commitment.

The control node (CN) manages the centralized concurrency control of the partition locking-granules. Once a lock is granted to a transaction T, CN sends T to a data-node. In Table 1, *kwtpgtime* is the CPU time of CN for computing $\mathcal{E}(q)$ of a lock-request q in K2, and *reducetime* is that for computing the optimized serializable order in Chain [Ohm89].

A data-node (DN) is modeled by *ObjTime*: the time to process an object at the node. On a data-node, a step switches to the next waiting one after processing each object.

Note that the values of both *NumNodes* and *NumParts* in Table 1 are much smaller than those in short-term transaction simulations. The performance metrics we use is the throughput θ at the saturated point. θ is the number of committed transaction per second (TPS). It is measured by increasing λ with $mpl = \infty$, and the saturated point is when $\theta = 0.9\lambda$. For each measured point, the simulation runs in 1,000,000 clock (1 clock = 1 milli-second).

3 Experiments and Discussion

The thrashing by the high data contention is caused either by chains of blocking or by a hot set. We test the performance of the above protocols on these thrashings.

Experiment1: Test on Chains of Blocking.

NumNodes = 8, *NumParts* =8, 16, 24, or 32. Tr-pattern1: `r(F1:1)`→ `r(F2:5)` → `w(F1:0.2)` → `w(F2:1)`.

In Tr-pattern1, the first two read-steps require X-locks and cause chains of blocking. `F1` and `F2` are randomly chosen among *NumParts* partitions.

Figure 2 shows the throughputs versus *NumParts*. K∞ is K-WTPG of $K = \infty$. ASL, Chain, and K2 have 80% - 100% higher throughput than C2PL in the figure. Since ASL has no blocking, we can see that Chain and K2 avoid chains of blocking successfully. Comparing K2 with K∞, we can see that K-WTPG of a lower value of K performs better.

Experiment2: Test on a Hot Set.

NumNodes = 8, *NumParts* = 32, *NumHots* = 4, 8, 16, or 24. Tr-pattern2: `r(B:5)` → `w(F1:1)` → `w(F2:1)`.

These read/write steps request S/X-lock respectively. `B` is chosen randomly among 8 'read-only' partitions spread over all the data-nodes. `F1` and `F2` are chosen randomly among the other *NumHots* partitions.

Figure 3 displays the throughputs as a function of *NumHots*. Since ASL permits the smallest number of active transactions, ASL has the lowest throughput in the figure. At *NumHot* = 4, the 'chain-form' constraint in [Ohm89] also degrades the throughput of Chain. K2 performs best because it allows any topology of a WTPG [Ohm89]. At *NumHots* = 8 and 16, both K2 and Chain outperform C2PL because of chains of blocking in C2PL.

we have seen that the available utilization of data-nodes are relatively low (about 60 %) in these experiments. It is because the high data contention has limited the inter-transaction parallelism of BATs. So the intra-transaction parallelism should be highly utilized

in the BAT processing.

Next we test the performance of both Chain and K2 when transactions declare the erroneous I/O demands.

Experiment3: Sensitivity Test.
NumNodes = 8, *NumParts*=16, Tr-pattern1 in Experiment 1. The I/O demand of each step is estimated at C by the formula: $C = C_0 \times (1 + x)$, where C_0 is the exact I/O demand of the step. x is the error ratio, given by the normal distribution of the average 0 and the standard deviation σ. (when $x \leq -1$, $C = 0$).

We also use the protocol Chain-C2PL (or K2-C2PL) as the lower bound of Chain (or K2), They are C2PL except that a WTPG must keep the constraint of 'chain-form' and that of 'K-conflict' in [Ohm89] respectively.

Figure 4 displays the throughputs as a function of σ. At $\sigma = 0.5$, Chain and K2 still keep 70% higher throughput than C2PL. K2 is more sensitive to the error ratio than Chain. Since Chain-C2PL has the high throughput, we can see that Chain is insensitive because of its chain-form constraint. In contrast, K2 avoids chains of blocking mainly using the weights in the WTPG.

4 Summary

This paper has presented the performance of the two new schedulers for scheduling Bulk Access Transactions. In the simulation results, these schedulers achieve stable and much higher performance than the two phase lock and the atomic lock, even when transactions declare erroneous I/O demands. In our future works, it is important to clarify the effects of the intra-transaction parallelism in the BAT-processing.

References

[Ohm89] Ohmori,T. et al., Technical Report, DE-89-21, IEICE, July, '89.

[Nis87] Nishio,S. et al., in Proc. 5th Int'l Workshop Database Machines '87.

Table1: Table of system/workload parameters

Parameter	Meaning	Value
NumNodes	number of data-nodes	4 ~ 24
NumParts	number of partitions	8 ~ 32
NumHots	number of hot partitions	4 ~ 24
mpl	multi programming level	infinite
λ	trans. arrival rate	0 ~ 3 TPS
CPUspeed	CPU speed of the control node	4 MIPS
netdelay	network delay time	negligible
msgtime	message send/receive CPUtime	2 ms
ddtime	deadlock detection CPUtime	1 ms
kwtpgtime	weight evaluation CPUtime	10 ms
reducetime	WTPG resolution CPUtime	40 ms
startuptime	trans. startup CPUtime	5 ms
committime	commitment CPUtime	7 ms
toptime	chain topology test CPUtime	5 ms
ObjTime	1 object processing time at a data-node	1000 ms
sleeptime	delay time before restart	5000 ms
keeptime	period of control-saving	5000 ms

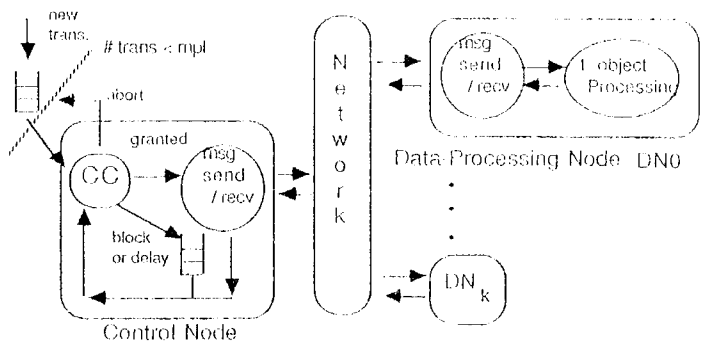

Figure1: Simulation Model

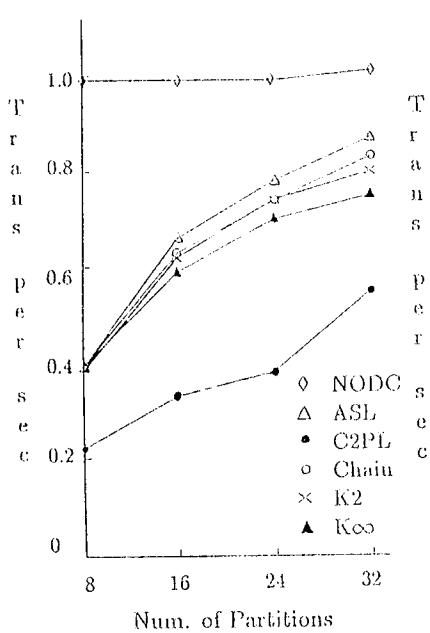

Figure2: Test on Chains of Blocking

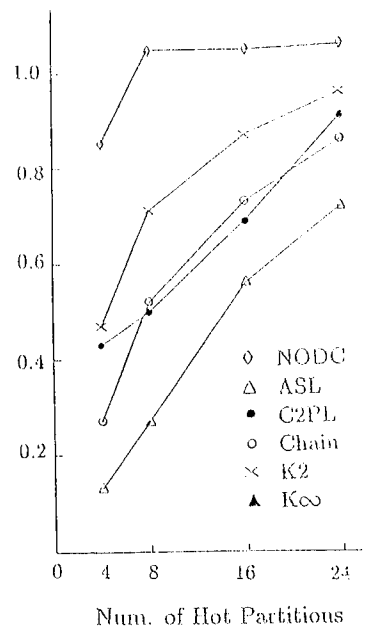

Figure3: Test on a Hot Set

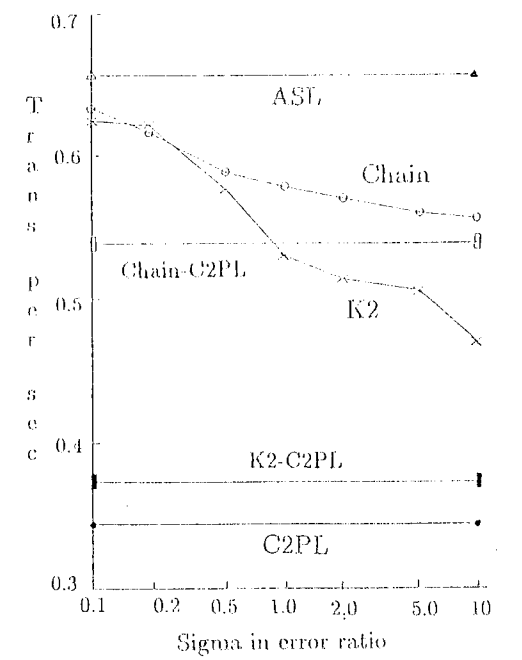

Figure4: Test on Sensitivity