Introduction to Real-Time Databases

郭大維 教授
ktw@csie.ntu.edu.tw
嵌入式系統暨無線網路實驗室
(Embedded Systems and Wireless Networking Laboratory)
國立臺灣大學資訊工程學系

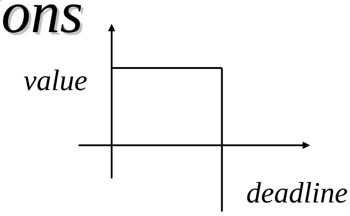
Reading:
Kam-yiu Lam and Tei-Wei Kuo, "Real-Time Database Systems: Architecture and Techniques", Kluwer Academic Publishers, 2000
Krishna and Kang, "Real-TimeSystems," McGRAW-HILL, 1997.

Introduction

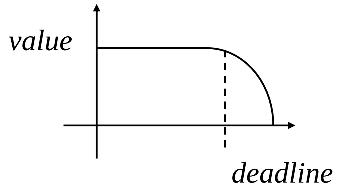
- *An Informal Definition of Real-Time Databases:*
  A real-time database system is a database system in which a timely response to a user request is needed.
- *Types of Real-Time Database Systems:*
  - *Hard real-time database systems, e.g., safety-critical system such as an early warning system, etc.*
  - *Soft real-time database systems, e.g., banking system, airline reservation system, digital library, stock market system, etc.*
  - *Mixed real-time database systems, e.g., air traffic control system, etc.*

Introduction

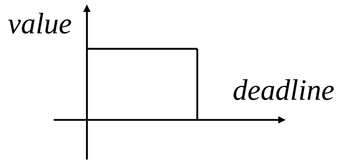
- Types of Real-Time Transactions
  - Hard real-time transactions
    - No deadline violation



  - Soft real-time transactions
    - Low miss ratio or avg/worst-case response time



  - Firm real-time transactions
    - No value after deadlines expire.



Introduction

- Design Issues
  - Real-Time Concurrency Control
    - Optimistic vs Conservative CC
    - Index
  - Run-Time System Management
    - Recovery
    - Buffer Management
    - Disk Scheduling
  - Distributed RTDBMS
    - Data Replication
    - Commit Processing
    - Mobile RTDBMS
  - etc

Introduction to Real-Time Database

- *Checklist*
  - ⊕ *What should we really know about the design issues of real-time databases?*
  - ⊕ *What is known about concurrency control of real-time data access?*
  - ⊕ *What is known about real-time recovery?*
  - ⊕ *Why is it so hard to have response-time predictability?*
  - ⊕ *What is main-memory database? Is it useful to RTDB?*
  - ⊗ *What is known about real-time query optimization?*
  - ⊗ *What is known about availability issues, real-time file systems, and disk management?*

Introduction to Real-Time Database

Time

Concurrency Control (CC)
(Son, Ramaritham, Lin, Bestavos, Wolfe, Garcia-Molina, Mok, Kuo, Lam, Zhao, Sha, etc, since early 1980)

CC Based on Simulation
.
Complex CC
.
CC Based on Application Semantics
.
CC of Mixed RT Transactions
.
CC + Recovery
.

Model and Design
(Son, Lin, Singhal, Mok, Kuo, Dayal,Ramaritham, Stankovic, since early 1980)

Weak Correctness Criteria
(Mok, Kuo, Pu, Ramaritham, Lin, etc, since mid 1980)

Fault Tolerance & Availability
(Lin, 1988 & ....)

Query Optimization
(Wolfe, etc, since early 1990)

Recovery and Logging
(Ramaritham,Lam, since 1996)

Active + RTDB
(Son, Mok, Lam, since 1996)

File Structure & Data Caching
(??)

Commercial Database & Realistic Workloads
(??)

Introduction to Real-Time Database

Real-Time vs. General Purpose Databases

- *Basic Definitions & ACID Properties*
- *Correctness Criteria*
- *Consistency Constraints*
- *Needs for Response-Time Predictability*
- *Main Memory Database for RTDB*

Basic Definitions & ACID Properties

- *A transaction is a sequence of read and write operations, i.e., r(x) and w(y). (transaction instance)*
- *A history/schedule over a set of transactions is an interleaving of the read and write operations issued by the transactions , e.g., w2(x),r1(x),w2(y),r1(y).*
- *A query transaction consists of only read operations. (vs update)*
- *A serial schedule is a sequence of operations which are issued by transactions one by one, e.g., w2(x), w2(y), r1(x), r1(y).*

Data Access versus Semaphore Locking

- *Typical Schedule*
  - *T1 (x=x-100, y=y+100)   T2*

T1	T2
r(x)	
w(x)	
	r(x)
	r(y)
r(y)	
w(y)	

Basic Definitions & ACID Properties

- *In conventional databases, transactions must satisfy the ACID properties:*
  - *Atomicity: all or nothing.*
  - *Consistency: consistent transformation of DB states.*
  - *Isolation: invisibility for dirty data. (degrees)*
  - *Durability: permanent committed updates.*
- *In real-time databases, relaxing ACID depends on application semantics.*

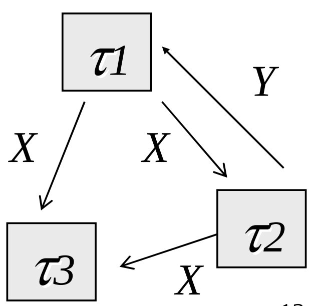
Correctness Criteria

- *Conventional Criteria:*
  - *Final-State Serializability ~ NP-hard*
    - *Generate the same final state as a serial schedule does.*
  - *View Serializability ~ NP-hard*
    - *Final-State Serializability, and*
    - *Corresponding transactions have the same view over the database.*
  - *Conflict Serializability ~ Polynomial*
    - *The order of conflicting operations is the same as that of a serial schedule.*
- *Criteria for Real-Time Databases:*
  - *Weak criteria are possible, but their definitions depend on application semantics.*

Reading: C. Papadimitriou, "The theory of Database Concurrency Control," Computer Science Press, 1986.

Examples: Serializability

- *S = R1(X)W1(X) R2(X)R2(Y)W2(Y) W1(Y)*
  - *S is final-state equivalent to S1 = $\tau 2$ $\tau 1$*
  - *S is not view equivalent to S1 because of the transaction view of $\tau 2$, which is a dead transaction.*
- *S = R1(Y) R3(W) R2(Y) W1(Y)W1(X) W2(X)W2(Z) W3(X)*
  - *S is view equivalent to S1 = $\tau 2$ $\tau 1$ $\tau 3$.*
  - *S is not conflict equivalent to S1 because of the order of the two dead W(X)'s of $\tau 1$ and $\tau 2$.*



Correctness Criteria - Relaxing...

- *An Airline Reservation Example*[1]
  - *Rules:*
    - *Reservation:*
      - *Reserve a seat.*
      - *If over 100 seats, assign 5 flight attendants to the flight; otherwise assign 3 attendants.*
    - *Cancellation*
      - *Cancel a seat on the flight.*
      - *If the number of reservations drops below 85, assign only3 flight attendants to the flight.*
    - *Hysteresis: The assigned number will not oscillate rapidly.*
  - *Scenarios: Starting from 3 attendants from TPE to LA, and LA to AUS, 99 servations on each flight.*
    - *ReserveA(TPE,LA), CancelB(TPE,LA,), CancelB(LA,AUS), ReserveA(LA,AUS)*
    - *TPE-LA: 5 attendants, LA-AUS: 3, An acceptable but non-serializable schedule!*

[1] *H. Garcia-Molina and K. Salem, "Main Memory Database Systems: An Overview," IEEE Trans. Knowledge and Data Engineering, 4(6):509-516, 1992.*

Consistency Constraints

- *In conventional databases,*
  - *Internal Consistency*
    - *Database satisfies consistency and integrity constraints, e.g., x=y.*
- *In real-time databases,timing properties of data are important, too!*
  - *Absolute/External Consistency*
    - *Data reflect the changings of the external environment.*
    - *For example, stock index.*
  - *Relative/Temporal Consistency*
    - *The ages of two data are within a tolerable length of time.*
    - *For example, the temperature and the pressure of a boiler read at time t.*

Needs for Response-Time Predictability

- *Why is it so hard to have response-time predictability for disk-based or other databases?*
  - *Blocking and transaction abortings caused by the requirement to meet the ACID properties.*
  - *Unpredictability of disk access time and page faults[2].*
  - *Data dependency of transaction executions.*
- *However, in many cases, we often only*
  - *use main memory database, or*
  - *need worst-case predictability, or*
  - *use real memory addressing, or*
  - *best effort in scheduling.*

[2] *Only bad for disk-based databases, logging purpose, or virtual memory usage.*

Main Memory Database for RTDB

- *Why main memory databases?*
  - *Improve response time.*
  - *Reduce unpredictability of response time.*
    - *Critical factors of contentions:*
      - *transaction duration and lock granularity.*
  - *Hardware technology improvements.*
- *What is the cost or research beside money?*
  - *Higher frequency in data backup.*
  - *Vulnerable to system failures - efficient logging mechanism, recoverability, and recovery time to transaction and system failures.*
  - *Different indexing schemes beside shallow B-tree.*

Introduction to Real-Time Database

Concurrency Control

- *Conservative Concurrency Control*
- *Optimistic Concurrency Control*
- *Semantics-Based Concurrency Control*
- *Concurrency Control for Mixed Transaction Systems*

Introduction to Real-Time Database

- *Issues for Real-Time Concurrency Control (RT-CC)*
  - *Data consistency and integrity.*
  - *Urgency of transaction executions.*
- *General Approaches for RT-CC:*
  - *Integrate real-time techniques, e.g., RM, EDF, and PCP, and traditional concurrency control protocols, e.g., 2PL, OCC, RWPCP, Multiversion-CC.*
  - *Utilize application semantics to improve system performance.*
  - *Adopt suitable software architectures such as an object-oriented design, etc.*

Introduction to Real-Time Database

- Classification of RT-CC protocols:
  - Syntactic-based concurrency control
    - Conservative Mechanism
      - Prevention of any serializability violation in advance.
        - conservative in resource usages.
      - Significant blocking cost
    - Optimistic Mechanism
      - Three phases for each transaction execution:
        - read, validation, write
      - Significant aborting cost
    - etc
  - Semantics-based concurrency control
    - CC with flexibility in reordering read and write events.
      - Concurrency level vs worst-case blocking time.
    - CC with reduced and simplified CC protocols, e.g., single writer.
      - Such systems which totally satisfy requirements rarely exist.
    - etc.

Syntactic-Based Concurrency Control

- Pessimistic Concurrency Control
  - Ensure that transactions will not violate serializability consistency during their executions
  - Q: How to favor high priority transactions, e.g., in the processing of locking requests?
- Optimistic Concurrency Control
  - Any violation of serializability consistency from a transaction will not be checked until its validation time.
  - Q: How to favor high priority transactions if there exist conflicts between high and low priority transactions?

Lock-Oriented Concurrency Control

- *Characteristics*
  - *A typical way for pessimistic concurrency control*
  - *Prevention of serializability violation by lock management - possibly lengthy blocking time*
- *An Example Protocol*
  - *Two-phase locking + A Priority Assignment Scheme, such as RM or EDF.*
    - *Two-phase locking – growing phase and shrinking phase*
    - *priority inheritance.*

Lock-Based Concurrency Control

- *Read/Write Priority Ceiling Protocol (RWPCP)*
- *2-Version RWPCP*
- *Aborting versus Blocking*

Lock-Based Concurrency Control

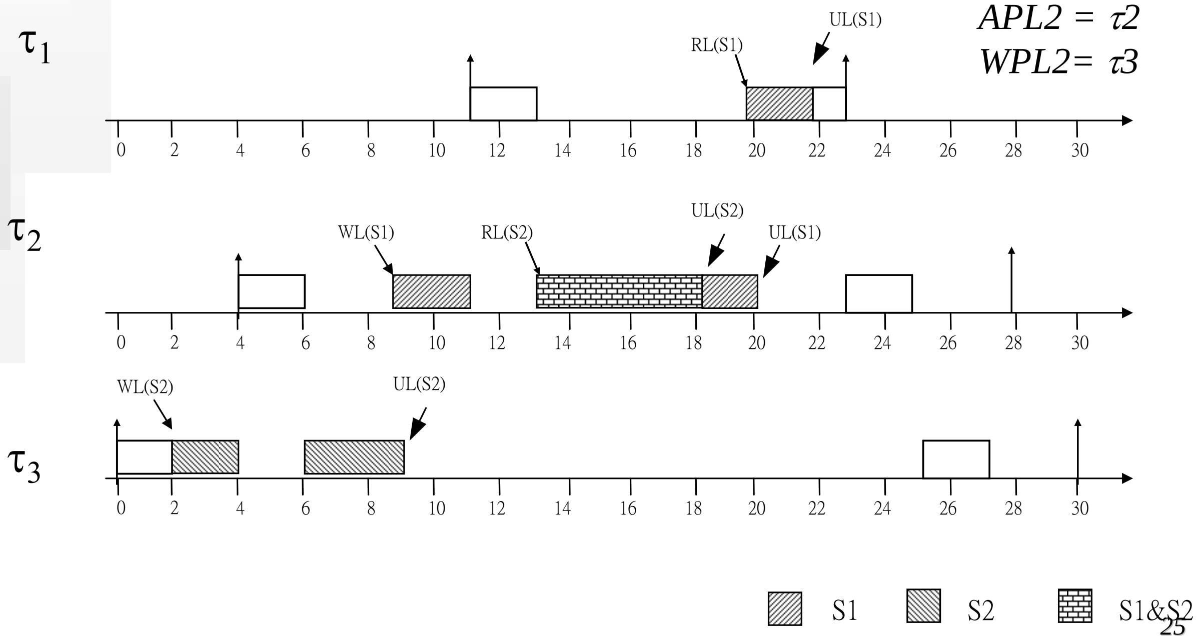
- *Read/Write Priority Ceiling Protocol (RWPCP)*
- *2-Version RWPCP*
- *Aborting versus Blocking*

Read/Write Priority Ceiling Protocol

- *Ceiling definitions of data object* $\mathbf{O_i}$
  - *Write Priority Ceiling (WPL_i) of* $\mathbf{O_i}$
  - *Absolute Priority Ceiling (APL_i) of* $\mathbf{O_i}$
  - *Read/Write Priority Ceiling ($RWPL_i$) of* $\mathbf{O_i}$
    - *WPL_i or APL_i*
- *Ceiling rule*
  - *A transaction may lock a data object if its priority is higher than the highest $RWPL_i$ of data objects locked by other transactions.*

RWPCP

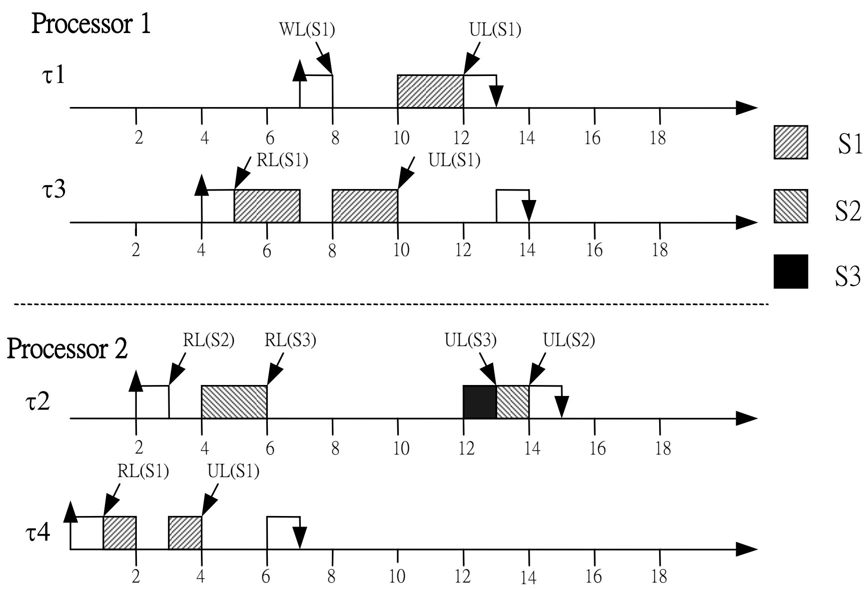
APL1 = τ1
WPL1= τ2
APL2 = τ2
WPL2= τ3



Properties of RWPCP

- *Properties in Uniprocessor Environments*
  - *Lemma1: No transitive blocking (τ_L->τ_M->τ_H)*
  - *Theorem 1: One priority inversion per transaction.*
  - *Theorem 2 : Deadlock-freeness*
  - *Theorem 4: Serializable schedules if the two-phase-locking scheme (2PL) is followed.*

RWPCP in a Multiprocessor Environment



APL1 = τ1
WPL1= τ1
APL2 = τ2
WPL2= null
APL3 = τ2
WPL3= null

Example 1 RWPCP Schedule

An Observation

The number of priority inversion may be more than one when there are more than one processor in the system!

Why?

- *The priority gap between the priority of τ_2 and the read write priority ceiling of the data objects locked by τ_2*

- *How to guarantee single priority inversion time in a multiprocessor environment ?*

Reference: Tei-Wei Kuo and Hsin-Chia Hsih, 2000, "Concurrency Control in a Multiprocessor Real-Time Database System," the 12th Euromicro Conference on Real-Time Systems, Stockholm, Sweden, June 2000.

Lock-Based Concurrency Control

- *Read/Write Priority Ceiling Protocol (RWPCP)*
- *2-Version RWPCP (2VPCP)*
- *Aborting versus Blocking*

Two-Version Read/Write Priority Ceiling Protocol

- *Objectives:*
  - *Reduce the blocking time of higher-priority transactions*
  - *Dynamic Adjustment of Serializability Order*
- *Lock Modes*
  - *Working/Consistent Versions*
    - *Writes on working versions*
    - *Reads from consistent versions*
  - *Read/Write/Certify Locks*

Two-Version Read/Write Priority Ceiling Protocol

- *Ceiling definitions of data object* $\mathbf{O_i}$
  - *Write Priority Ceiling* (WPL_i) *of* $\mathbf{O_i}$
  - *Absolute Priority Ceiling* (APL_i) *of* $\mathbf{O_i}$
  - *Read/Write Priority Ceiling* ($RWPL_i$) *of* $\mathbf{O_i}$
    - – WPL_i *for read/write locks or* APL_i *for certify locks*
- *Ceiling rule*
  - *A transaction may lock a data object if its priority is higher than the highest* $RWPL_i$ *of data objects locked by other transactions.*

Two-Version Read/Write Priority Ceiling Protocol

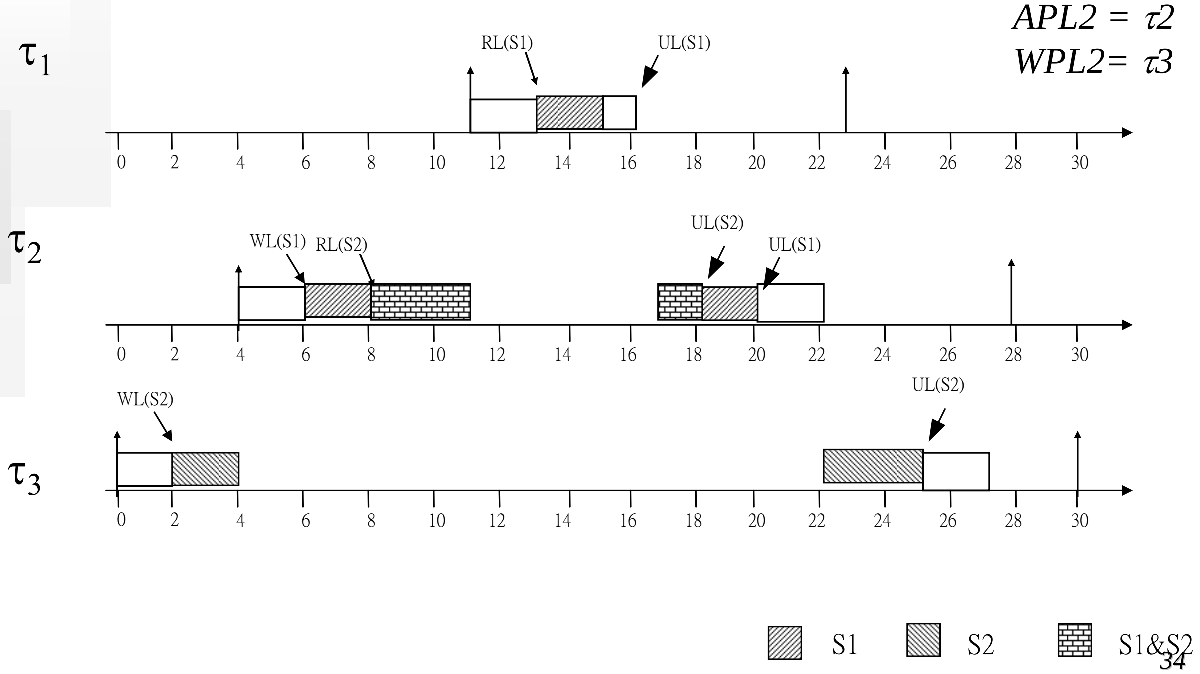
- *A compatibility table for 2VPCP:*

	Requested locks		
Lock already set	Read	Write	Certify
Read	*Granted*	*Granted*	*Blocked*
Write	*Granted*	*Blocked*	*Blocked*
Certify	*Blocked*	*Blocked*	*Blocked*

- Remark:
  - More versions?
  - Aborting allowed?

2VPCP

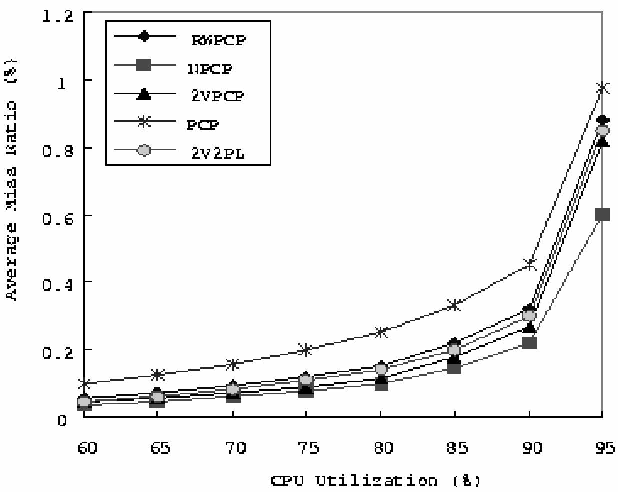
APL1 = τ1
WPL1= τ2
APL2 = τ2
WPL2= τ3



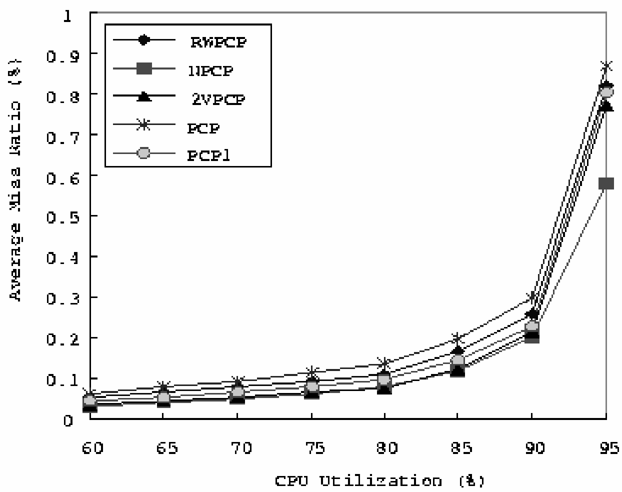
Properties of 2VPCP

- Properties
  - Lemma1: No transitive blocking ($\tau_L \rightarrow \tau_M \rightarrow \tau_H$)
  - Theorem 1: One priority inversion per transaction.
  - Theorem 2 : Deadlock-freeness
  - Theorem 4: Serializable schedules if the two-phase-locking scheme (2PL) is followed.

Simulation Results



(a) database size = 150

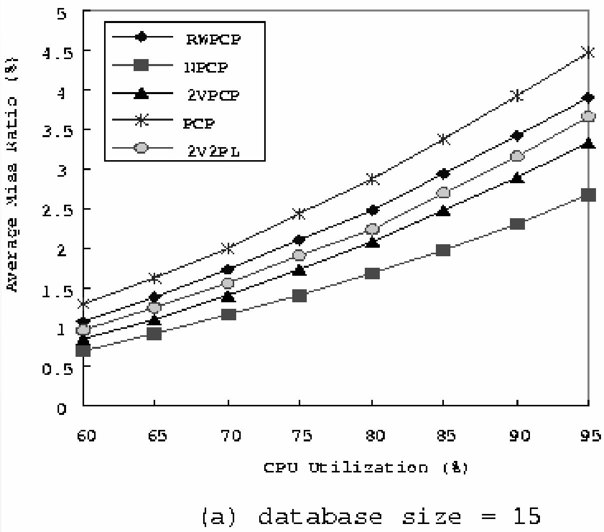


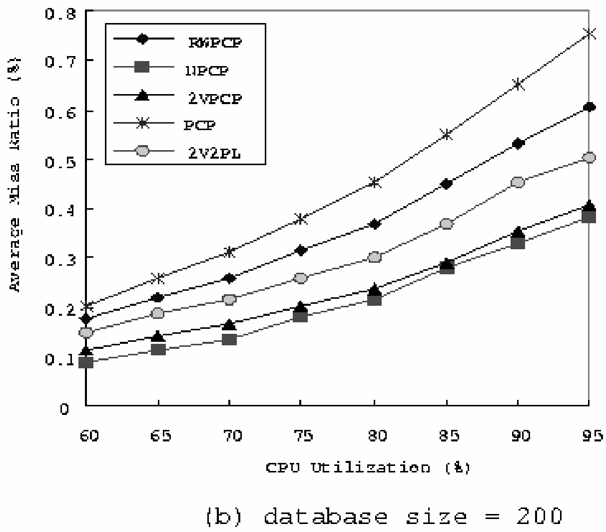
(b) database size = 200

Miss Ratios of All Transactions

* NPNP adopts multiple versions for a data object!

Simulation Results





Miss Ratios of the Top ¼ Priority Transactions

Lock-Based Concurrency Control

- *Read/Write Priority Ceiling Protocol (RWPCP)*
- *2-Version RWPCP (2VPCP)*
- *Aborting versus Blocking*

Basic Aborting Protocol (BAP)

- Main Idea:

  When a lower priority transaction introduces excessive blocking to a higher priority transaction, then higher priority transaction will abort the lower priority transaction.

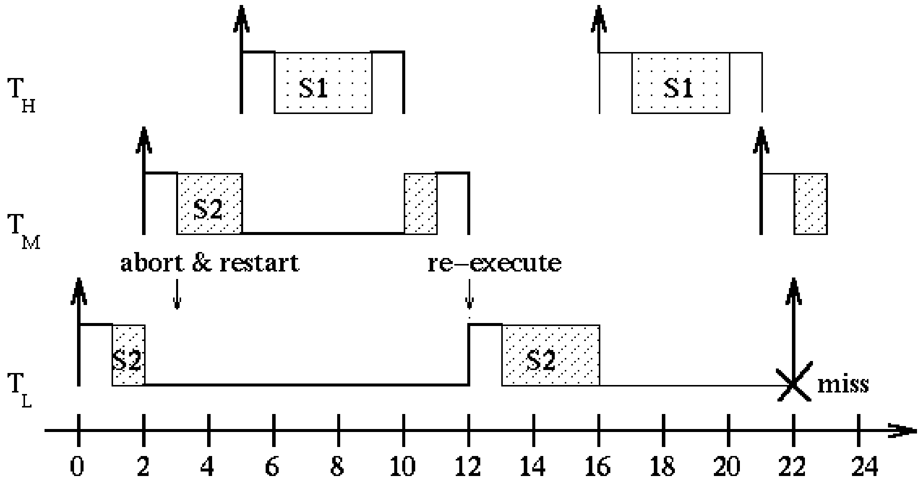
- Compatible Modules:
  - *Priority Ceiling Protocol (PCP)*
  - *2PL*
  - *A simple aborting mechanism*

Reference: Tei-Wei Kuo, Ming-Chung Liang, and LihChyun Shu, "Abort-Oriented Concurrency Control for Real-Time Databases," IEEE Transactions on Computers (SCI), Vol. 50, No. 7, July 2001, pp. 660-673.

BAP Protocol Summary

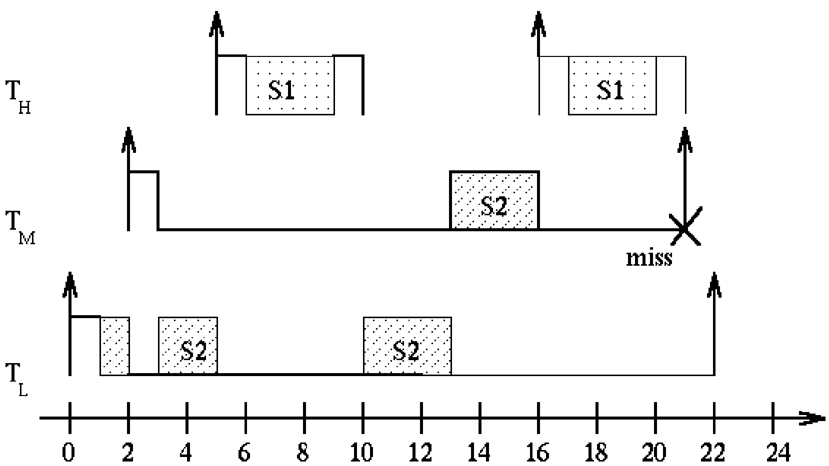
- *Transactions are classified as abortable or non-abortable in an off-line fashion.*
- *Each transaction instance τ must acquire a semaphore before access the corresponding data object.*
  - *Lock granted: when a transaction instance τ attempts to lock a semaphore, it checks whether it's priority is higher than the priority ceiling of all semaphores already locked by other transaction instances.*
  - *Blocked: if there exists any non-abortable lower priority transaction instance τ' which locked a semaphore with a priority ceiling no less than the priority of τ, then τ is blocked by τ', and τ' inherits the priority of τ.*
  - *Aborting: Otherwise, τ' is aborted, and the lock is granted.*

BAP Schedule



$T_H(5,11)$ $T_M(5,19)$ $T_L(7,22)$

PCP+2PL Schedule



$T_H(5,11)$ $T_M(5,19)$ $T_L(7,22)$

Properties

- **Lemma 1**. *BAP prevents deadlocks.*
- **Theorem 1**. *Schedules generated by BAP are logically correct (based on serializability).*
- **Theorem 3**. *No transaction instance τ scheduled by BAP directly or indirectly inherits a priority level from a transaction instance which is aborted before τ commits or is aborted.*
- **Theorem 4**. *A transaction instance can experience at most one time of priority inversion under BAP.*
- **Theorem 5**. *A higher priority transaction instance can abort at most one lower priority transaction instance under BAP.*

Schedulability Analysis

- **A-cost$_{i,j}$:** *maximum direct aborting cost of τ_j charged by τ_i*
- **α-cost$_{i,j}$** : *max*(A-cost$_{i,k}$), *where $i < k <= j$.*
- **Lemma 2.** *The worst-case aborting cost for a request of transaction τ_j between time 0 and time $t <= p_j$ is at most*

$$\sum_{\tau_i \in HPC_j} \left(\left\lceil \frac{t}{p_i} \right\rceil \times \alpha - \text{cost}_{i,j} \right)$$

- **Lemma 3.** *A transaction τ_i scheduled by BAP will always meet its deadline for all process phases if there exists a pair $(k,m) \in R_i$ such that*

$$\sum_{j \in HPC_i} \left(c_j \left\lceil \frac{mp_k}{p_j} \right\rceil \right) + c_i + b_i + ab_i \le mp_k$$

where b_i and ab_i are the worst case blocking cost and aborting cost of transaction τ_i,

$$R_i = \{(k,m) \mid 1 \le k \le i, m = 1,2,\ldots, \left\lfloor \frac{p_i}{p_k} \right\rfloor \}$$

Schedulability Analysis Procedure

- Lemma 3 shows that the maximum blocking time that transaction τ_i can tolerate is

$$MB_i = \max_{t \in SP_i}\left[t - \sum_{j \in HPC_i}\left(c_j\left\lceil \frac{t}{p_j}\right\rceil\right) - c_i - ab_i\right]$$

- Initially all transactions are non-abortable.
  - *i=1*
  - *If i > n then stop*
  - *If transaction* τ_j *has a priority ceiling no less than* τ_i *and the length of the critical section is larger than* MB_i *, then* τ_j *becomes abortable, where j > i.*
  - *i=i+1*

Extensions of BAP

- Table-Driven Aborting Protocol (TAP)
  - *Give a more fine-grained fashion of aborting relationship*
  - *An instance of transaction* τ_i *can abort an instance of transaction* τ_j *only when AB[i, j] = yes.*
  - *The rest of the TAP is the same as BAP.*
  - *The properties of BAP remain.*

Extensions of BAP

- Dynamic Aborting Protocol (DAP)
  - *Run-Time Calculation of Tolerable Blocking Time:*
    - *The blocking time that an instance of a transaction can tolerate is estimated dynamically and based on the current workload instead of the worst case situation.*
  - *Run-Time Determination of Aborting Relationship:*
    - *An instance of a higher priority transaction τ_H can abort an instance of a lower priority transaction τ_L at time t only if (1) τ_L blocks τ_H, (2) τ_L is abortable, and (3) the maximum tolerable blocking time of τ_H is less than the possible blocking time of τ_L at time t.*

DAP: Approximate Schedulability Test

- **Theorem 8.** *A transaction τ_i scheduled by DAP will always meet its deadline for all process phases if*

$$\sum_{j \in HPC_i} \left(\left\lceil \frac{d_i}{p_j} \right\rceil \times c_j \right) + c_i + b_i + ab_i \le d_i$$

- *The maximum blocking time that transaction τ_i can tolerate at time t is approximated as:*

$$AMB_i = (d_i - t) - \sum_{j \in HPC_i} \left(\left\lceil \frac{d_i - t}{p_j} \right\rceil \times c_j \right) - c_i - ab_i(t)$$

where

$$ab_i(t) = \sum_{\tau_j \in HPC_j} \left(\left\lceil \frac{d_i - t}{p_i} \right\rceil \times \alpha - \mathrm{cost}_{i,j} \right)$$

- *The rest of the DAP is the same as BAP.*
- *The properties of BAP remain.*

Performance Evaluation

- Case Study
  - Generic Avionics Platform
    - 18 periodic transactions.
    - 9 data objects.
  - Olympus AOCS
    - 10 periodic transactions.
    - 4 sporadic transactions.
    - 17 data objects.
- Simulation Experiment
  - Compare BAP, TAP, and DAP with the well known Priority Ceiling Protocol (PCP), Rate Monotonic Scheduling algorithm (RMS), and Abort Ceiling Protocol (ACP).

Case Study 1: Generic Avionics Platform

Transaction #	Period (ms)	Exec (ms)	Blocking (ms)	Read Set	Write Set
Timer_Interrupt	1	0.051	0	-	-
Weapon_Release	200	3	9	-	DB
Radar_Tracking_Filter	25	2	9	DB,N	D,DB,T
RWR_Contact_Mgmt	25	5	9	DB,N,K,W	D,DB,T
Poll_Bus_Device	40	1	9	all	-
Weapon_Aim	50	3	9	N,T	D,DB
Radar_Target_Update	50	5	9	DB,N,K	D,DB,T
Nav_Update	59	8	9	DB,K,R	D,DB,T,R,W,RW
Display_Graphic	80	9	5	all	DB
Display_Hook_Update	80	2	5	DB	-
Tracking_Target_Upd	100	5	3	DB,N,K,R,RW	D,W
Weapon_Protocol	200	1	3	k	DB
Nav_Steering_Cmds	200	3	3	D	D
Display_Stores_Update	200	1	3	W	DB
Dislpay_Keyset	200	1	3	DB	all
Display_Stat_Update	200	3	1	all	
BET_E_Status_Update	1000	1	1	-	D
Nav_Status	1000	1	0	DB	D

Table 1: Transaction set characteristics adapted from the generic avionics example

Schedulability Analysis: Generic Avionics Platform

Transaction #	Period (ms)	Exec (ms)	Abortable ?	Aborting Cost (ab_i) (ms)	Schedulability Test (Use best mp_k for MB_i)	Tolerent Blocking (MB_i)
Timer_Interrupt	1	0.051	No	0	0.051	0.949
Weapon Release	200	3.01	No	0	$\lceil 5/1 \rceil \times 0.051 + 3.01$	1.735
Radar_Tracking_Filter	25	2.03	Yes	2.03	$\lceil 25/1 \rceil \times 0.051 + \lceil 25/200 \rceil \times 3.01 + \lceil 25/25 \rceil \times 2.03 + abort$	16.655
RWR_Contact_Mgmt	25	5.03	Yes	10.06	$\lceil 25/1 \rceil \times 0.051 + \lceil 25/200 \rceil \times 3.01 + \lceil 25/25 \rceil \times 2.03 + \lceil 25/25 \rceil \times 5.03 + abort$	3.595
Poll_Bus_Device	40	1	No	15.09	$\lceil 40/1 \rceil \times 0.051 + \lceil 40/200 \rceil \times 3.01 + \lceil 40/25 \rceil \times 2.03 + \lceil 40/25 \rceil \times 5.03 + \lceil 40/40 \rceil \times 1 + abort$	4.74
Weapon Aim	50	3.02	No	15.09	$\lceil 50/1 \rceil \times 0.051 + \lceil 50/200 \rceil \times 3.01 + \lceil 50/25 \rceil \times 2.03 + \lceil 50/25 \rceil \times 5.03 + \lceil 50/40 \rceil \times 1 + \lceil 50/50 \rceil \times 3.02 + abort$	10.21

Table 2: Schedulability analysis of **BAP** for the generic avionics example

* PCP + 2PL: Only the first two transactions are schedulable.

Simulation Results

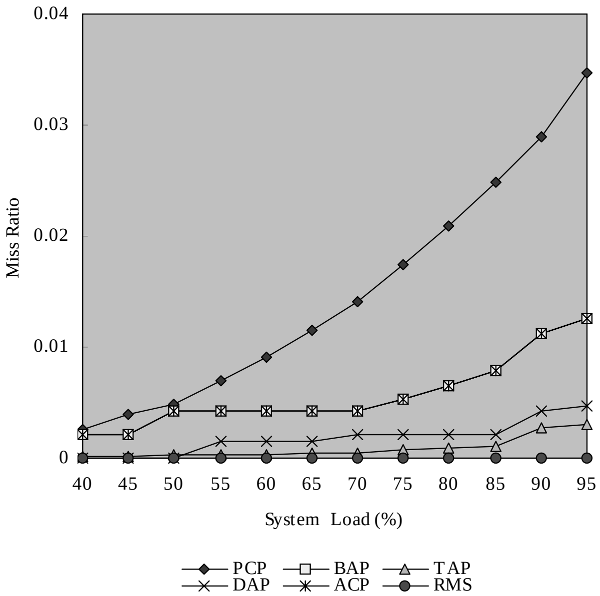


Fig 4: Top 1/4 Transactions, DB size = 25

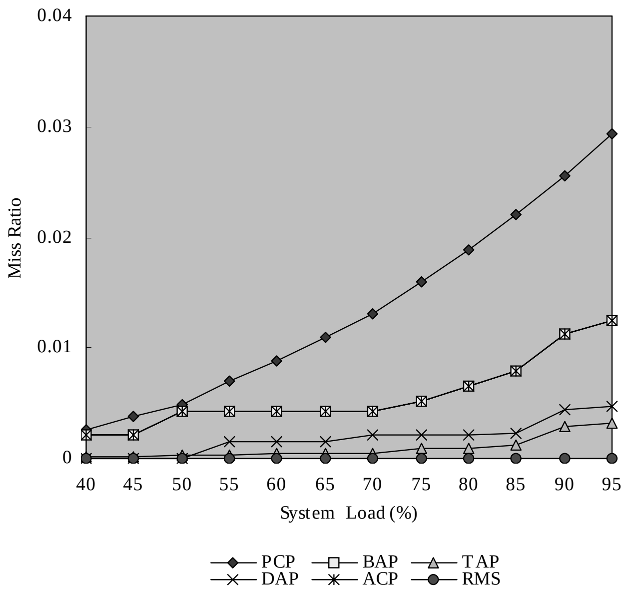


Fig 5: Top 1/4 Transactions, DB size = 50

Simulation Results

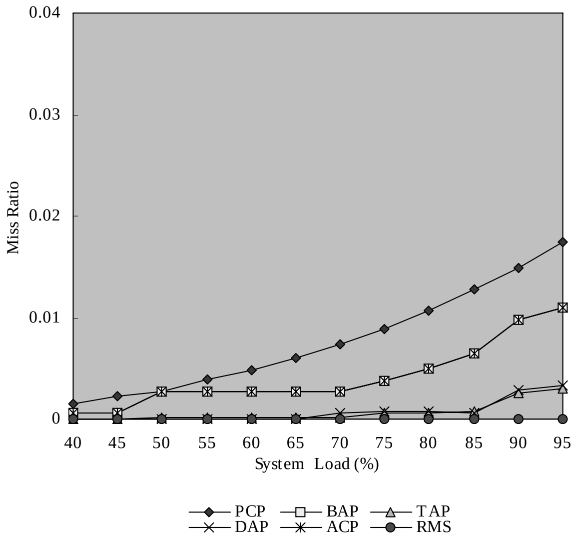


Fig 6: Top 1/4 Transactions, DB size = 100

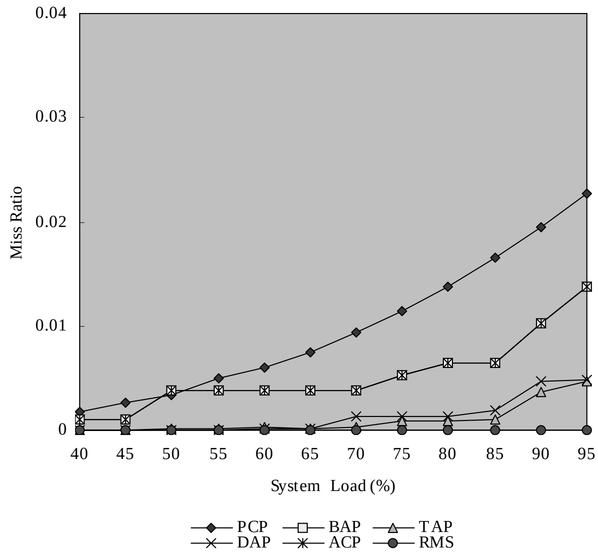


Fig 7: Top 1/4 Transactions, DB size = 150

Simulation Results

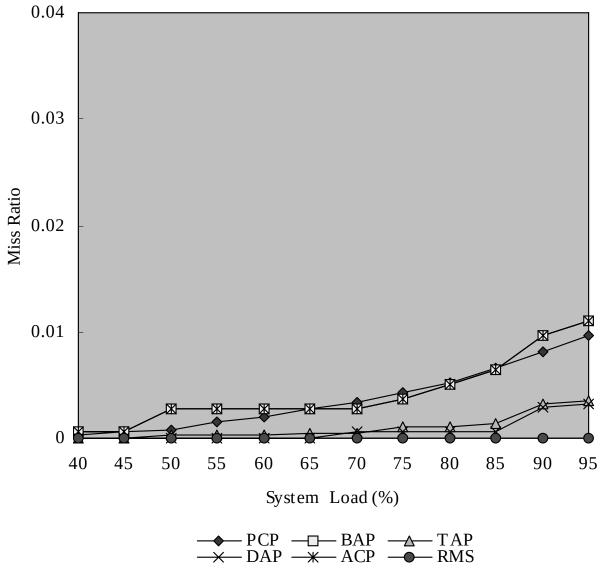


Fig 8: Top 1/4 Transactions, DB size = 200

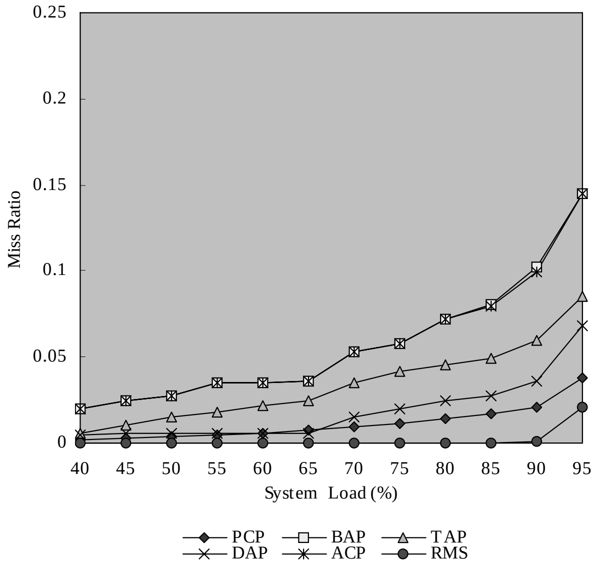


Fig 9: The Whole Transaction Set, DB siz = 100

Optimistic Concurrency Control

- *Broadcast Commit*
- *Alternation of Serializability*

Real-Time Optimistic Concurrency Control

- *Example A - A simple optimistic CC*
  - *Three execution phases: read, validation, write.*
  - *Use timestamp to validate the serializability of trans.*
  - *Let the timestamp of A be before that of T. Serializability consistency is not violated due to T* if
    - *A completed its write phase before T starts its read phase,or*
    - *The read set of A is distinct from the write set of T, and A finished its write phase before T starts its write phase, or*
    - *The write set of A is distinct from both the read and write sets of T.*
  - *Long transactions are been against because they tend to have a lot of conflict.*

Real-Time Optimistic Concurrency Control

- *Variations:*
  - *Broadcast commit protocol:*
    - *When a transaction commits, it tells all the transactions that it conflicts with so that they abort.*
  - *When priority is involved...*
    - *When T commits at its validation phase, all lower-priority transactions abort.*
    - *Any higher priority transactions H in conflict with T...*
      - *Sacrifice policy - abort T.*
      - *Wait policy - Wait until H commits. If H commits, abort T; otherwise, commit T.*
      - *Wait-X policy - T commits unless more than X% of the transactions that conflict with it are of a higher priority; otherwise, T waits… (X=50 seems very good.)*

Real-Time Optimistic Concurrency Control

- *Example B - Alternation of Serializability*
  - *Motivation: Reduce abortings by flexibly adjusting serializability order.*
  - *For example,*

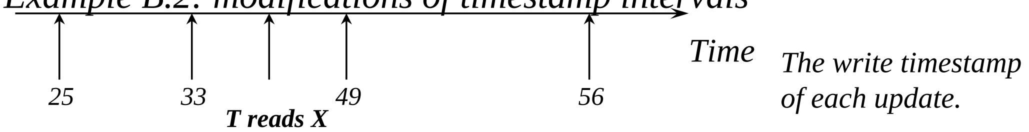
    $R_A(x), R_A(y), R_A(z), R_B(x), R_A(u), W_A(x), W_B(v)$

    An acceptable order is B, A instead of A, B!!
  - *An timestamp-based algorithm:*
    - *The system maintains a valid interval (x,y) for each transaction to assign the transaction a timestamp at its commit time.*
    - *A read timestamp and a write timestamp for each data item which are the latest timestamps of committed transactions that have read and updated it (updates done at commit times).*
    - *Updating of a data item at the commit time of a transaction is effective if the timestamp of the transaction is larger than the write timestamp of the data item; otherwise, the write timestamp is not changed and the update is simply ignored.*
  - *Example B.1:*
    - *x1(r=40,w=3), x2(r=2,w=60), timestamp(T1)=25, ReadSet(T1)= {x1}, WriteSet(T1)={x1,x2,x3}*
    - *After T1 commits, x1(r=40,w=25), x2(r=2,w=60), x1 is updated, x2 remains the same.*

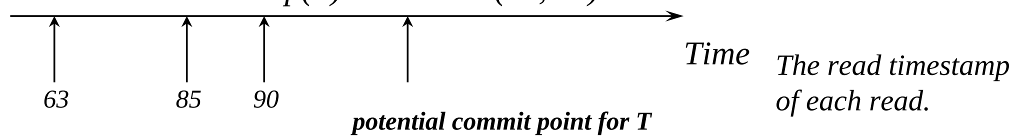
Remark: The serializability order of transactions scheduled by pessimistic CC is often determined at lock request times.

Real-Time Optimistic Concurrency Control

- *Example B.2: modifications of timestamp intervals*



Timestamp(T) must be in (33, 49)!



Timestamp(T) must be larger than 90!

- *Rules for assigning timestamps to a transaction T:*
  - *Determine the validity intervals of data read by T*
  - *Take the intersection of all these validity intervals. Let it be $I_T=(l_T, u_T)$. If the interval is empty, then abort T.*
  - *Let max_T be the maximum read timestamp of all of the data items updated by T. If $max_T >= u_T$ then abort T. Otherwise choose a timestamp for T in the interval (max_T, u_T).*

Y. Lin and S.H. Son, "Concurrency control in Real-Time Databases by Dynamically Adjustment of Serializability Order," IEEE Real-Time Systems Symposium, 1990, pp. 104-112.

Real-Time Optimistic Concurrency Control

- *The protocol shown in Example B only considers the transaction that is being validated in the context of the transactions that have already been committed.*
- *Validation Schemes[1]: (not exclusively classified)*
  - *Backward validation: The validation procedure is performed against recently committed transactions.*
    - *Ti: validating transaction, Tj: transactions commit between the time Ti starts execution and the time at which Ti comes to the validation phase.*
    - *Cond. 1: The writes of Tj should not affect the read phase of Ti.*
    - *Abort Ti if necessary.*
  - *Forward validation: The validation of a transaction is performed against concurrently executing transactions.*
    - *Ti: validating transaction, Tj: transactions which currently executes in their read phase.*
    - *Cond. 1: The writes of Ti should not affect the read phase of Tj.*
    - *Abort Ti or Tj depending on properties such as priority level.*

1. Kwok-Wa Lam, "Concurrency Control and Transaction Scheduling in Real-Time Database Systems," Ph.D. thesis, Dept. of Computer Science, City University of Hong Kong, 1997.

Real-Time Concurrency Control

- Other papers for discussion
  - R. Abbott, H. Garcia-Molina, “Scheduling Real-Time Transactions: A Performance Evaluation,” Proceedings of the 14th VLDB Conference, 1988.
  - M.-C. Liang, T.-W. Kuo, and L.C. Shu,”BAP: A Class of Abort-Oriented Protocols Based on the Notion of Compatibility,” The Third International Workshop on Real-Time Computing Systems and Applications, 1996.
  - T.-W. Kuo and A.K. Mok, “SSP: a Semantics-Based Protocol for Real-time Data access,” IEEE 14th Real-Time Systems Symposium, 1993.

Introduction to Real-Time Database

Other Issues

- Logging and Recovery
- Query Optimization
- Availability