

## View Updates & Temporal Correctness in Soft Real-Time Database Systems

---

Oral Defence by  
Cheng Chun Kong (MPhil., HKUCSIS)  
*Supervised by: Dr. B.C.M. Kao*  
10<sup>th</sup> August, 2000.

## Contents

---

- ⌘ Problems Studied
- ⌘ Updates & Recomputations
- ⌘ Temporal Correctness
- ⌘ Transaction Scheduling
- ⌘ Performance Study
- ⌘ Future Work

## I Problems Studied

### Real-Time Database Systems (RTDB)

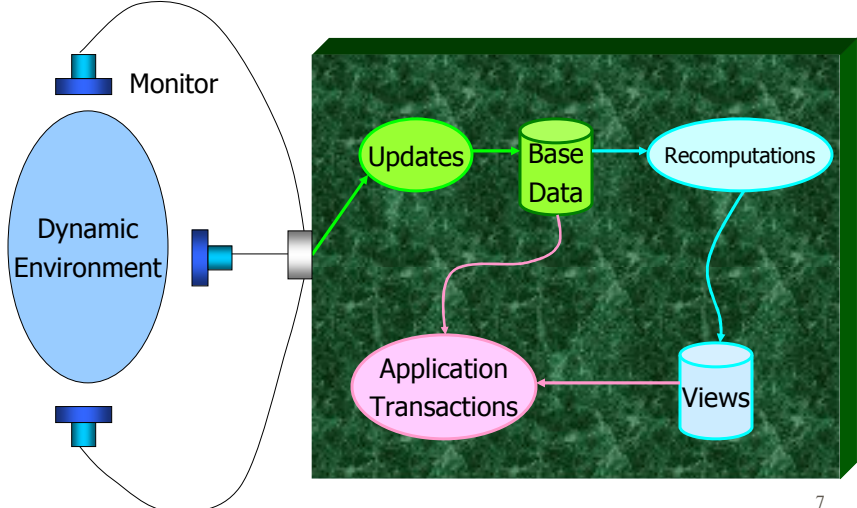
- ⌘ A database system with time constraints
- ⌘ Transactions can be associated with time constraints e.g., deadlines
- ⌘ Refresh database items to reflect the external world.

## Example: Programmed Stock Trading

- ⌘ Monitoring stocks & financial instruments
- ⌘ On discovering an opportunity, a trading transaction is performed.
- ⌘ The transaction becomes useless if it misses the deadline.

6

## System Model



7

## Timing Requirements of Application Transactions

- ⌘ **Transaction Timeliness:** How *fast* a transaction is completed
  - ❖ Can a transaction finish before its *deadline*?
- ⌘ **Data Timeliness:** How *fresh* the data is
  - ❖ Stale data is less useful to a transaction

8

## The 2 Requirements Conflict!

- ⌘ Updates & Recomputations applied promptly to *satisfy data timeliness*.
- ⌘ They are very heavy loads.
- ⌘ Transactions get little share of system resources.
- ⌘ This *hurts transaction timeliness*.

9

## Problems Studied in this Thesis

- ⌘ Understand Updates, Recons & Transactions
- ⌘ Balancing the 2 timing requirements by redefining *Temporal Correctness*
- ⌘ Design scheduling algorithms so that temporal correctness is enforced

10

## II Updates & Recomputations

## Properties of Updates

- ⌘ Based on Adelberg, Garcia-Molina & Kao's work: "*Applying Update Streams in a Soft RTDB*" in *ACM SIGMOD 1995*.

12

## Properties of Updates

- ⌘ Updates arrive at a very high rate
  - ☒ In the US market, 500 stock updates per second.
- ⌘ An update need not be given full transactional support.
- ⌘ Use a single update process to install updates

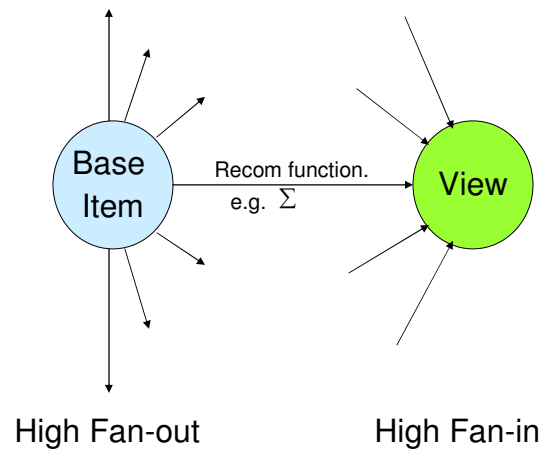
13

## Properties of Recomputations

⌘ Based on Adelberg, Garcia-Molina & Kao's work, "*Database Support for Efficiently Maintaining Derived Data*" in *EDBT 1996*.

14

## Fan-in and Fan-out



15

### III Temporal Correctness

#### Temporal Correctness

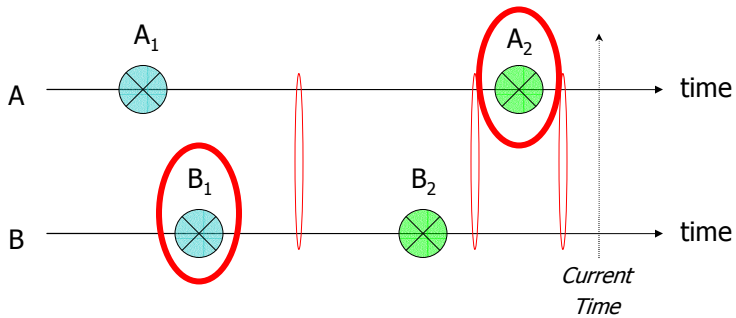
##### ⌘ Absolute Consistency (AC)

- ❖ A data item timely reflects the state of an external object.

##### ⌘ Relative Consistency (RC)

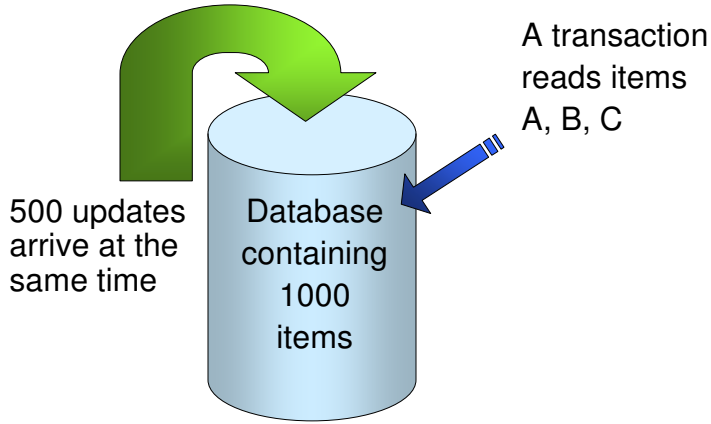
- ❖ A set of data items with values reflecting the states of the external objects at the *same time instant*.

### AC and RC: An Example



- A<sub>2</sub> and B<sub>2</sub> are AC. They are also RC.
- (A<sub>1</sub>, B<sub>1</sub>) and (A<sub>1</sub>, B<sub>2</sub>) are RC pairs.
- A<sub>2</sub> and B<sub>1</sub> are NOT consistent pairs.

### Defining AC/RC from the Perspective of a Transaction



## Instantaneous System

- ⌘ An **ideal** system which applies updates/recoms as soon as they arrive, taking **zero time** to do it.
- ⌘ A measure of correctness of real systems.

23

## Real System - ACS

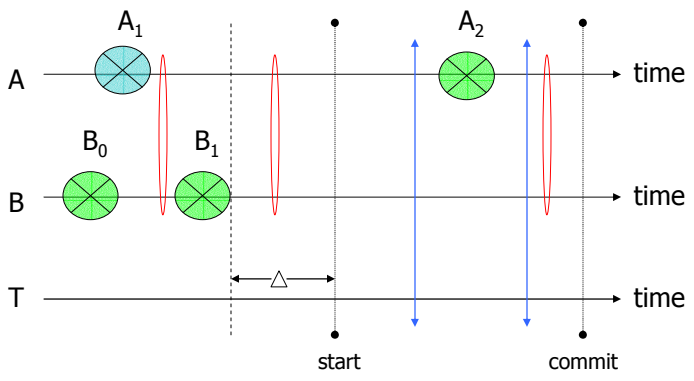
- ⌘ A transaction is given the values of all objects it reads such that:
  - ☒ these values can be found in an instantaneous system at the **commit time**.
- ⌘ No transactions can read old or inconsistent data.

24

# Real System - RCS

- ⌘ Can we *relax* data timeliness?
- ⌘ Allow transactions to read slightly old data.
- ⌘ The set of values read by a transaction can be found in an instantaneous system not older than **transaction's start time -  $\Delta$** .

# ACS vs RCS : An Example



- Readset of T in ACS:  $(A_2, B_1)$
- Readset of T in RCS:  $(A_1, B_1), (A_2, B_1)$

## IV Transaction Scheduling

### Schedulers for ACS and RCS

#### ⌘ URT

- ☒ Updates and Recons first, Transactions later

#### ⌘ OD

- ☒ On Demand Triggering of Updates & Recons

#### ⌘ OD-H

- ☒ No transactions: URT
- ☒ Otherwise, OD

29

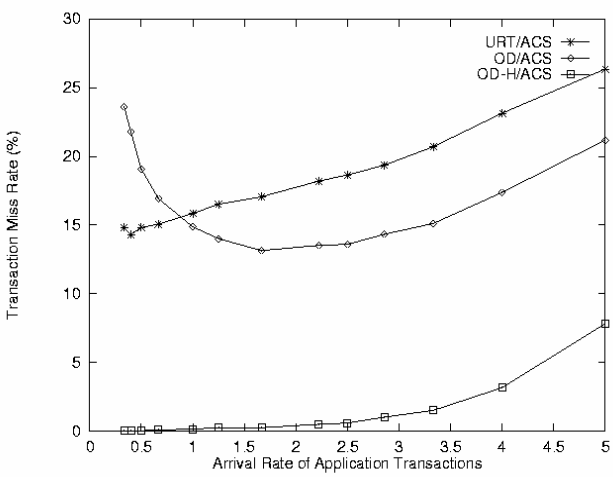
## Implementing the Schedulers

- ⌘ Concurrency control
  - ☒ ACS: *HP-2PL*
  - ☒ RCS : *Multi-version Concurrency Control*
- ⌘ OD manager for OD
- ⌘ Version manager for RCS

30

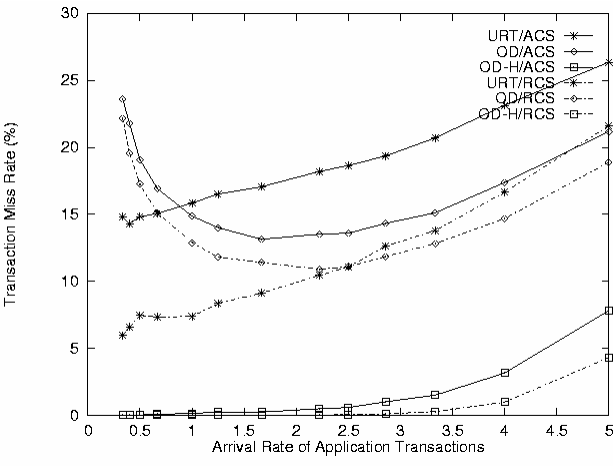
## V Performance Study

### Effect of Transaction Arrival Rate (ACS)



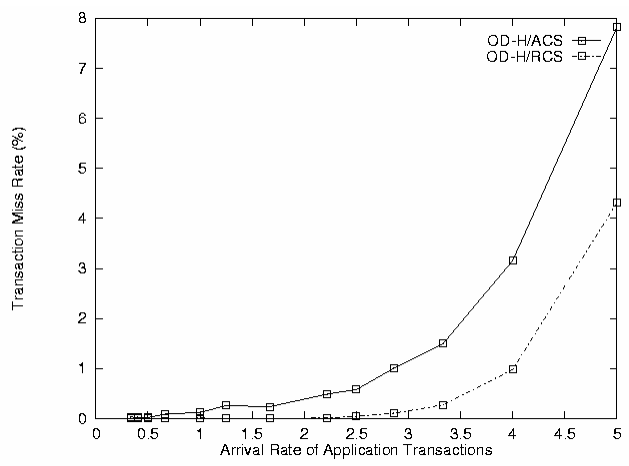
33

### Effect of Transaction Arrival Rate (ACS & RCS)



35

## Effect of Transaction Arrival Rate (OD-H/ACS vs OD-H/RCS)



36

## VI Conclusions

## Conclusions

- ⌘ We studied how temporal correctness can be incorporated into an RTDB.
- ⌘ We defined temporal consistency from the perspective of transactions.

39

## Conclusions

- ⌘ The data timeliness of ACS is too strict.
- ⌘ It has a high transaction miss rate.
- ⌘ RCS is less strict.
- ⌘ It has a smaller transaction miss rate.

40

## Conclusions

- ⌘ We studied how URT, OD and OD-H should be implemented for ACS.
- ⌘ OD-H when applied to an RCS results in the smallest transaction miss rate.

41

## Future Work

- ⌘ *Version selection problem* in an RCS
- ⌘ If the readset of a transaction is known before its execution, the performance of RCS may be improved.
- ⌘ The simulation model can be expanded from single processor and single disk, to multiple processors and multiple disks.

43

## Q & A Session

