CSE 530 Adv. ES

Atomicity (Cont.)

Recap:
- Basic def. two types: alternating before or after
  
  - AoN \rightarrow AoN PUT - single cell update; build upon:
    - Journal Storage System, bootstrap
    - arbitrary data structure from.

Post is poor! lots of n.v. r/w are involved

Obj: A

- interface layer
  - obtain new action id
  - read / write new volume

Outcome record

Commit
Recovery
- configure ARGOS server
- make system recover by necessity

No free lunch!

Optional for common core.

Principle: Make common core four!

How can this improve pay?

Update logs to change to cell舒服.

General info
- logs the change to F: 5.

Automatic Log:

Method to improve performance:
log (change my id)

log (change, my id)

crvalue = new credit old + amount

crvalue. old = new (credit account)
devvalue. old = devvalue. old - amount

devvalue. old = get (debit account)

my. initlog (begin - transaction)

ex. transfer (debit account, credit account, amount)

3. undo action
2. redo action
1. add or note action
0. print record

in memory db.
Log (END, TRANSACTION, myid)

else

Log (commit, commit, myid)

if duplicate new > 0 then

//increment

n inertia

Put (debit = account, credit = new)
then break repeat

! if log-record

! log-record

! then perform undo-action of

! if log-record

! there = change

! if log-record

! there = outcome

! if log-record

! there = outcome

! if log-record

! tcl = action-id

! then

! log-record = previous record of

! log

! log

! start of and to log repeat until beginning

! procedure A800R (action-id) L150 scan

! L141
Then perform log-rollback. 

end (outcome, rollback, status = committed) 

in outcome 

(rollback, action id) 

and (out come, rollback) 

find record 

if (log-rollback, type = change) 

log-rollback = next log-rollback of log 

stopping at beginning of log repeat until end 

then outcome = outcome of (log-rollback) 

if (log-rollback, type = outcome) 

log-rollback = previous record of log 

stopping at end of log repeat until beginning 

outcome = null 

proceed (recovery) // in memory db.
- Builds the in-memory db by performing all the committed action in order in which they were logged.

- idempotent
  - recovery op: only modifies V.M.
  - V.M is reset at each failure.

⇒ it is safe to restart it again and again after each failure.
And the in-memory db will be constructed correctly if the recovery op finishes.
- before or after activity
- other 10-30m cost
- total 7th tier
Next time