Storage Systems

NPTEL Course

Jan 2012

(Lecture 32)

K. Gopinath

Indian Institute of Science

Consistency Mgmt

- In single-site storage systems
  - Even a single site has many components
    - upd due to link: ln /a/b/c /x/y
      - 1: incr link count of c
      - 2: add entry c in dir y
    - what is the best way to do this in presence of crashes and panics? 1,2 or 2,1?
      - Ordered write, Logging/Journalling, Transactions
  - System level or appl level view wrt consistency
  - Is there control over system after some failure?
    - Yes: on a power failure or crash (panic)
      - FS recovery on reboot
    - No: have to prevent inconsistency
      - Transactions, commit protocols
- In multi-site storage systems

2-phase commit

Transaction T initiated at site S_i and txn coord there C_i.
When subT completed at all sites (all sites inform C_i), start 2PC protocol

- **Phase 1**: C_i logs *prepare(T)*; sends *prepare(T)* to all C_k
  C_k: on receiving prepare msg, either
  does not commit: logs *no(T)* and sends *abort(T)* to C_i
  commits: logs *ready(T)* with all changes to log onto stable storage and sends *ready(T)* to C_i
- **Phase 2**: C_i receives response to prepare msg from all C_k or timeout:
  ready(T) from all: log *commit(T)*; send same to all C_k (logged)
  else: log *abort(T)*; send same to all C_k (logged)
  each C_k sends *ack(T)*

C_i receives acks from all: logs *complete(T)*
ready(T) from a site: will follow coord's order to commit or abort

Failures

- Detected by coord or sites reliably. **At coord**:
- *a site* S_k *fails* before *ready(T)*: *abort(T)* assumed

  after : continue 2PC

  recovery at S_k : *commit(T)* present in log: redo(T)

  abort(T) : undo(T)

  ready(T) : consult C_i to find out T's status

  if C_i up, easy; redo/undo as nec

  if C_i down: check with other C_k

  repeat until someone up or C_i

  nothing! : undo(T)

- *a coord C_i fails* : check if any C_k has *commit(T)* : T committed

  abort(T) : T aborted

  if some C_k does not have *ready(T)* : T aborted

  all C_k have *ready(T)*: block until C_i recovers

- network partition: map to site/coord “failure”
  - if coord and some sites in one partition: assume other sites down?
  - other sites: assume coord failed?
- For recovery, instead of *ready(T)*, log *ready(T, set of locks held)*
  - helps new txns to get going if they do not use locks held by in-doubt T's
- Optimization for RO txns:
  - a C_k responds with *RO(T)* rather than *ready(T)*; C_i need not send *commit(T)*

3-phase commit

To avoid blocking in limited cases:

- eg. no netw partition; atmost K sites can fail & atleast K+1 sites up

Provide preliminary info about fate of T thru a precommit phase

Assume failures detected by coord or sites reliably

- **Phase1**: same as 2PC
- **Phase2**: C_i receives responses to prepare msg from all C_k or timeout:
  any site *abort(T)* or no response from a site until timeout: *abort(T)* to all
  ready(T) from every site: *precommit(T)* to log & to each site to its log
  ack sent to coord from each site whether abort or precommit (logged)
- **Phase3**: only executed if precommit in Phase2
  coord waits till atleast K acks: logs *commit(T)* & sends it to each to its log

ready(T): site's promise to follow coord's decision

precommit(T): coord's promise to commit

Failure detected at coord:

a site S_k fails before *ready(T)*:

abort(T) assumed

after: continue 3PC

recovery at S_k:

commit(T) present in log: redo(T)

abort(T): undo(T)

ready(T) but no abort/precommit:

consult C_i to find T's status

if C_i up, easy: *abort(T)*: undo(T)

precommit(T): send ack & resume protocol

commit(T): redo(T)

if C_i down: execute coord failure protocol

precommit(T) but no abort/commit:

consult C_i to find T's status (same as before)

coord failure protocol

- triggered when a site does not get response from coord
- elect a new leader (C_new)
- C_new requests status of T (committed, aborted, ready, precommitted or (nothing in log) not ready) from each site (incl C_new!)
  - 1+ committed: commit
  - 1+ abort: abort
- if 0 abort but 1+ precommit: C_new resumes protocol by sending new precommit(T)
  - safe to commit but cannot do it unilaterally as blocking if C_new fails (as in 2PC)
- else abort

One scenario: no active site has precommit =>

abort T good. 3 cases:

- C_i aborts before failure
- C_i has not reached decison
- C_i cannot commit before failure.

Assume commit:

- >K precommit from sites with acks (set S); C_i has failed;
- 1+ site in S active as only max K failures and has sent precommit;
- => precommit sent to C_new. -><-
- hence, if no precommit acks to C_new: abort is safe

No netw partition: otherwise multiple coord!