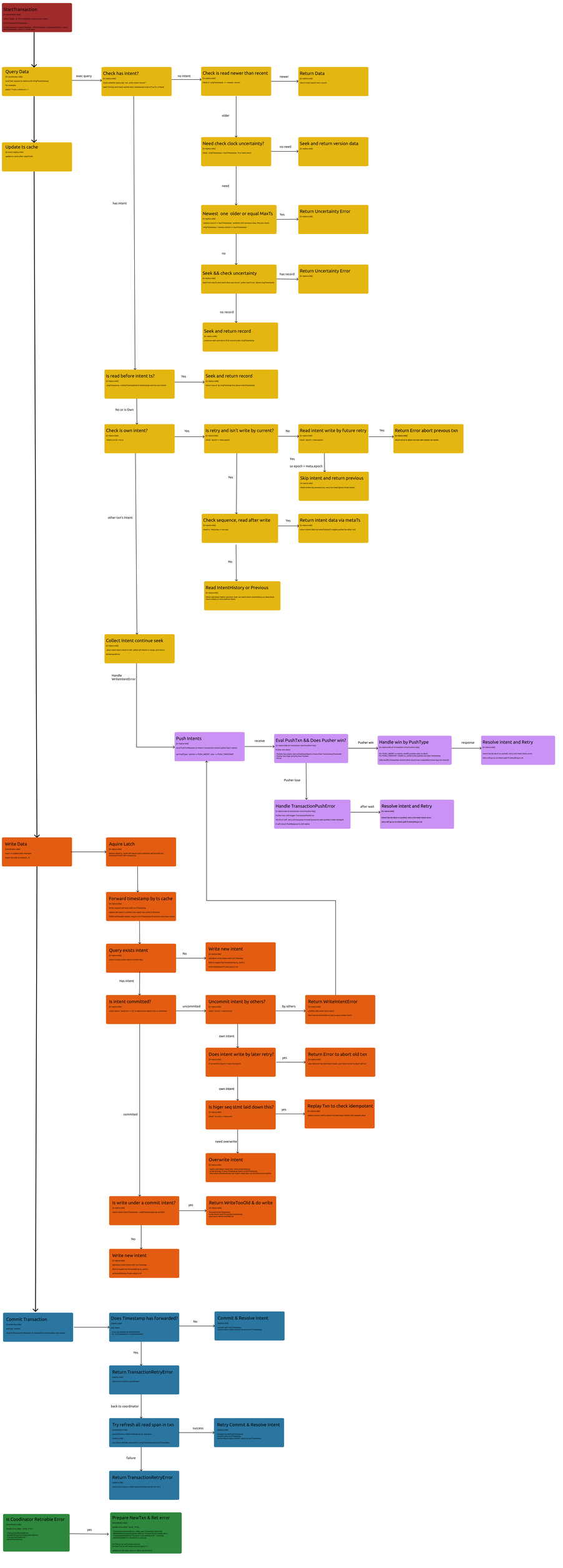
StartTransaction
(in raftor side)
after begin txn, first stmt after autocommit = false
after begin txn, transactions
txn.Timestamp = meta.Timestamp = ctx.Timestamp = ( ctx.HasClockTs = true) ? ctx.ClockTs : meta.Timestamp
txn.Status = Pending
Query Data
(in coordinator side)
send Get request to raftor with GrpTimeoutMap
For examples:
select * from t where id = 1
exec query
Check has intent?
(in raftor side)
check whether query key has write intent record ?
seek first key and check wether key's timestamp is (lare-if so it's a intent
no intent
Check is read newer than recent
(in raftor side)
check is: txn.Timestamp >= newest record
newer
Return Data
(in raftor side)
return seek result (key: record)
older
Need check clock uncertainty?
(in raftor side)
check: txn.Timestamp < maxTimestamp - if un-need check
no need
Seek and return version data
(in raftor side)
need
Update ts cache
(in every raftor side)
update ts cache after read finish
has intent
Newest one older or equal MaxTs
(in raftor side)
newest record >= maxTimestamp ( combine with previous step, this just check txn.Timestamp < newest record >= maxTimestamp)
Yes
Return Uncertainty Error
(in raftor side)
no
Seek && check uncertainty
(in raftor side)
seek from maxTs and check does any record under maxTs but above orgTimestamp
has record
Return Uncertainty Error
(in raftor side)
no record
Seek and return record
(in raftor side)
continue seek and return first record under orgTimestamp
Is read before intent ts?
(in raftor side)
orgTimestamp < meta(TimestampIntent timestamp) and not own intent
Yes
Seek and return record
(in raftor side)
return record by orgTimestamp but above meta(Timestamp)
No or is Own
Check is own intent?
(in raftor side)
check txn.id == meta.id
Yes
Is retry and isn't write by current?
(in raftor side)
check: epoch == meta.epoch
No
Read intent write by future retry
(in raftor side)
check: epoch > meta.epoch
Yes
Return Error abort prevous txn
(in raftor side)
return error to abort txn due new restart txn works.
Yes
so epoch > meta.epoch
Skip intent and return previous
(in raftor side)
intent written by previous txn, retry txn need ignore those intent.
Yes
other txn's Intent
Check sequence, read after write
(in raftor side)
check is: txn.seq >= rec.seq
Yes
Return intent data via metaTs
(in raftor side)
return intent data via meta(Timestamp/s maybe pushed by other txn)
No
Read IntentHistory or Previous
(in raftor side)
- read out down higher seq (for non-read, we need check intentHistory to determinate read's value or not, partial intent
Collect Intent continue seek
(in raftor side)
after meet ylew intent it will collect all clients in range, and return errIntentsError
Handle WriteIntentError
Push Intents
(in raftor side)
send PushTxnRequest to raftor's transaction record author key's raftor
txn.PushType: Abortw -> PUSH_ABORT, else -> PUSH_TIMESTAMP
receive
Eval PushTxn && Does Pusher win?
(in raftor side at transaction record author key)
- Pusher wins when:
- Pushee has higher, lock acquiredTime(pt) is more than Txn.Lockwait(Threshold) - Pusher has higher priority than Pushee
Pusher win
Handle win by PushType
(in raftor side at transaction record author key)
txn.PUSH_ABORT or expiration record (pusher lose) is abort
txn.PUSH_TIMESTAMP: modify ts_cache, let pusher use new timestamp
only modify transaction record when record was (metadata/ lose key) txn record
response
Resolve intent and Retry
(in raftor side)
intent has be abort or pushed, retry end meet intent error.
retry will go to no intent path if everything is ok
Pusher lose
Handle TransactionPushError
(in raftor side at transaction record author key)
Pusher lose, will trigger TransactionRetryError
this Error will retry end pusher forward (Lockwait Queue to wait pusher's state changed
& will return PushRequestw to end raftor
after wait
Resolve intent and Retry
(in raftor side)
- txn.Ts has be abort or pushed, retry end meet intent error.
- retry will go to no intent path if everything is ok
Write Data
(in coordinator side)
Send Put/ Cpdate/Delete request
- meet txn.id, ts values 0, 0
Aquire Latch
(in raftor side)
before Put txn, Cpdate and aquire latch and latch will be hold all
- meet txn.id, ts values 0, 0
Forward timestamp by ts cache
(in raftor side)
Write request will start with txn.Timestamp
raftor will check ts cache's new read new write to forward
- CHECK: write(table lock), require: txn.Timestamp forward to rect. read, raftor
Query exists intent
(in raftor side)
check is exists write intent in written key
No
Write new intent
(in raftor side)
- lock down write intent with txn.Timestamp
- Write is maybe has Forwarded by ts_cache /
- write tombstone if val.value is nil
Has intent
Is intent committed?
(in raftor side)
check: txn.id == meta.Txn.id
uncommited
Uncommit intent by others?
(in raftor side)
check: txn.id != meta.Txn.id
by others
Return WriteIntentError
(in raftor side)
- Conflict with other txn's intent
- Abort txn uncommitted or txn pu query read intent
own intent
Does intent write by later retry?
(in raftor side)
if (curreTxnTs.Epoch < meta.Txn.Epoch)
yes
Return Error to abort old txn
(in raftor side)
- raise retry txn has laid down intent , just return error to abort old txn
own intent
Is higer seq stmt laid down this?
(in raftor side)
check txn.seq >= meta.seq
yes
Replay Txn to check idempotent
(in raftor side)
replay txn from end by check is or not value match txn's expected value
need overwrite
Overwrite intent
(in raftor side)
- read old intent value (ctx: meta.IntentHistory)
- lock down write intent with txn.Timestamp (ctx: txn.Timestamp)
- raise error (if meta.txn.id == txn.id) meet error, we will leave old value and overlap
commited
Is write under a commit intent?
(in raftor side)
check intent meta.Timestamp > maxTimestamp(orgTs/MaxTs)
yes
Return WriteTooOld & do write
(in raftor side)
- Return to end of Forward
- Set Pushee.Timestamp = Pusher.Timestamp
- set curent write Pusheed to ts
No
Write new intent
(in raftor side)
- lock down write intent with txn.Timestamp
- Write is maybe has Forwarded by ts_cache /
- write tombstone if val.value is nil
Commit Transaction
(in coordinator side)
send EndTransactionRequest to transaction record author key raftor
Does Timestamp has forwarded?
(in raftor side)
check:
txn.orig.Sandwich && txn.Ts.Eq(txn.Ts.Prev())
&& txn.Timestamp == txn.orig.Timestamp
No
Commit & Resolve Intent
(in raftor side)
- Commit with txn.Timestamp
- Write intent made commit, resume all txn.Timestamp
Yes
Return TransactionRetryError
(in raftor side)
- return error back to coordinator
back-to-coordinator
Try refresh all read span in txn
(in coordinator side)
send RefreshRequestw to all read span
handle rules:
- check whether any intents in orgTimestamp and txn.Timestamp
success
Retry Commit & Resolve Intent
(in raftor side)
- Forward txn.RefreshTimestamp
- Reset txn.Timestamp
- Write intent made commit, resume all txn.Timestamp
failure
Return TransactionRetryError
(in raftor side)
- return error back to client and need client to do txn retry
Is Coodinator Retriable Error
(in coordinator side)
handle error after "send" if it's
- TransactionRetryWithProtoRefreshError
- ReadWithinUncertaintyIntervalError
- RefreshAfterCommitError
- MixedSuccessError
for first error will create new txn
- Epoch = Epoch + 1, Sequence = Sequence + 1
- update txn.Ts, make sure txn.Ts > Read/ wrtaiid ctime
yes
Prepare NewTxn & Ret error
(in coordinator side)
handle error after "send" if it's
- TransactionAbortedError
- TransactionPushFailedError
- TransactionStatusUnknownError
- IntentMissingError
- InternalSafeSynthetic
- UnhandledRetryableError
for first error will create new txn
- Epoch = 0, Sequence = 0
- update txn.Ts, make sure txn.Ts > Read/ wrtaiid ctime
- update txn.Ts, make sure txn.Ts > Read/ wrtaiid ctime