# Persistent Transaction Context

# Enhancement proposal

Change History:

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Name | Description | Distributed to |
| 12/2/2015 | Ping Zhou | Initial draft |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Persistent Transaction Context 1](#_Toc417913361)

[Enhancement proposal 1](#_Toc417913362)

[1. Background 1](#_Toc417913363)

[2. Design 2](#_Toc417913364)

# Background

  In order to calculate the new truncation point, we need check the open transaction list and persist it, but we cannot trigger such check every time when each new transaction come in, we need choose a check point(P) to trigger the open transaction list check , previously we chose the check point according to the transaction count , but when in P66/Ariba, we found when the transaction blooms, it is too frequently to persistence a transaction even every 30s, but when database is not so busy, it might take long time to trigger the persistence, In order to trigger the persistence regardless the system is busy or not, we had thought we’d better try to trigger it according to time gap.

# Design

## **1.1 Choose the check point according to time gap**

We can choose the check point for each (t1-t0) = (t2 – t1) = time gap (for example, 1 hour etc)

Start P1 P2

--------|-------------------|-------------------|--------------------------🡪 Time

t0 t1 t2

### Adjust the time gap during rescan

T2

Start T1 P1 P2

--------|--------------|----|-------------------|-------------------------------🡪 Time

t0 t1\* t1 t2

But during the rescan, the problem become complex, this time, given during initial scan when transaction T2 come in, the time gap for the check point matched, but the first long running transaction might be T1 on time t1\* instead of T2, during rescan, RepAgent will start from T1 at t1\*, but we still expect choose the same check point to guarantee that the same truncation point during rescan, so we need ignore the check point selection until the nearest check point P1 at t1 come in, and then we can continue to choose the check point according to the time gap, in this way, we can make sure the check point keep consistent between scan and rescan.

Similar to current tx processed count, We need to persist the time of persistent point in persisted tx. Thus when rescan, we can use it to calculate the time gap.

Even the current time gap could also be persisted, then we can configure a different time gap, effective from next persist point.

LSN make sense in log truncation, when LSN change, we can trigger the tx persistence checking in processing the first tx begin op.

1, last persist point is the version of persisted tx.

2, when LSN changed, in first tx begin, check tx.begin time with last persist time.

3, if tx.begin time satisfied the time gap after the last persist time, it is the current persist point.

4, open tx begin time and time of current persist point determine whether it is a long-running tx.

To reduce calculate count, we can only check the condition when lsn change.

If(lsn change) {

If(lr is tx begin && lr.time - last persist time >= time gap ) {

For(tx in open tx list){  
 if(lr.time – tx.begin.time >= long running tx criteria){

Persiste tx; // including current persist point time.

}else{

Break;

}  
 }

}

Start T1 P1 P2

--------|--------|------**|**----|---------|----------|-------------------------------🡪 Time

t0 t1\* t1 t2

The red | indicate the first tx begin op for a new LSN

Or maybe we don't need a LSN change, the first tx begin satisfy the time gap should be the persist point. If redo log size is big and PDB is not in a busy state, we might be in trouble again?

But if we don’t use LSN change for checking, maybe it is too frequent to check the time?

------------OR------------------

Could we configure a self-adaptive tx gap? It would change the value according to how busy the system are.

And we persist the last persisted point time (T0) and the tx gap. The P1 locator is already persisted as version.

In P1, adjust the tx gap,

G2 = G1/2 if time gap between P1 time and T0 is too long.

G2 = G1\*2 if time gap is too short.

Persist the TXs and Gap.

When rescan:

Before P1, no change,

In P1, calculate P2 = P1 + G2

## **1.2 Choose the check point according to specific ticket**

The first solution require us to guarantee the check point keep consistence between scan and rescan during the check point automatically choosing.

After discuss with Aaron, we think about what if we just choose the check point according to the specific ticket sent by user?

Ticket1 Ticket2 Ticket3

--------|-------------------|------------|--------------------------🡪 Time

t0 t1 t2

Given user can use some programs to send such ticket according to a gap, we also can start a dedicate thread to send ticket also. In this way, we even do not need to make sure the gap between those ticket keep same, and those ticket in log(check point) also will not be changed during scan and rescan. I will think more about the details.

The downside is we had to guarantee that user really had sent the ticket in period. Because we have no chance to update the already existed archive log file. How about user forget to sent ticket?

1, by oracle scheduler to send ticket?

Trigger the scheduler in ra\_admin init and delete it in ra\_admin deinit.

Use rs\_marker procedure

We can also change the interval.

2, by RA timer?