Commit Wait Class 대기시간 감소 방안

㈜엑셈 컨설팅본부/DB컨설팅팀 박 준연

개요

Wait Class 중 Commit 카테고리에 해당하는 Wait Event 에 의한 대기현상으로 DB 시스템의 성능 저하 현상이 발생하는 것은 종종 경험할 수 있다. 그 중 대표적인 Wait Event 는 Log File Sync 이다. 실제로 대부분의 DB 시스템의 Top 5 Wait Event 를 조사해 보면, Log File Sync 가 이에 속해 있는 경우가 대부분이다.

하지만 Log File Sync 를 포함하여 Log File Parallel Write 등의 Commit Wait Class 에 해당하는 Wait Event 가 발생하는 것 자체가 문제가 되진 않는다.

이는, Oracle 은 Commit 이나 Rollback 명령이 요청되면 이를 LGWR 에게 요청을 전달하며, LGWR 은 Redo Buffer 에서 가장 마지막에 기록이 이루어진 이후 시점부터 Commit 또는 Rollback 지점까지의 모든 Redo Entry 를 Redo Log File 에 기록하는 메커니즘을 가지고 있다. 따라서, 언급한 Wait Event 들은 이와 같은 과정에서 필연적으로 발생하는 Transaction 의 부가적인 현상일 뿐 그 자체로 문제가 된다고 볼 수 없다.

다만, 해당 Wait Event 를 대기하는 시간이 과도하게 긴 경우, 예를들어 과도한 Commit 이 발생하는 경우 등에 따라서는 이 대기시간을 감소시켜 전반적으로 DB 시스템의 최적화된 성능을 보장할 수도 있다. 이 문서는 Commit_wait , Commit_Logging 파라미터 값의 수정을 통해 Log File Sync 의 대기시간을 경감할 수 있는 방안과 이에 대한 주의 사항에 대한 정보를 알아보는 것을 목적으로 한다.

Commit_Wait / Commit_Logging

두 파라미터를 통해 Log File Sync 대기 시간을 감소시킬 수 있다는 것은 놀라움과 의아함을 동시에 느낄 수 있다. 단지 파라미터의 수정을 통해 당연히 대기해야 하는 시간을 감축한다는 것은 분명 성능을 개선해야 하는 입장에서는 놀라운 일이 될 것이다.

반면, 그에 따르는 비용도 감수해야 할 것이다. 이에 대한 자세한 내용은 마지막 부분에 언급하는 것으로 하고, 두 파라미터에 대한 설명과 수정 가능한 값들에 대한 의미를 알아보자.

Commit_Wait

Commit_Wait 파라미터는 Server Process 가 Redo Data 를 Redo Log File 에 기록을 완료할 때까지 대기할 것인지 말것인지를 결정하는 파라미터이다.

- Wait : Wait 값은 위 파라미터의 기본값이다. 이는 LGWR 의 작업이 완료되는 순간까지 Server Process 가 대기함을 의미한다.
- Nowait : 말그대로 Server Process 는 해당 작업에 대해 대기하지 않음을 의미한다. 이 경우, 시스템의 ACID 의 특성 중 지속성이 침해될 수 있으므로 일반적으로는 Nowait 으로 설정하는 것을 권고하지는 않는다.
- Force_Wait : Wait 설정 값과 유사한 의미를 갖는다. 다만, 시스템 레벨에서 설정한 경우, 세션 레벨에서 이를 무시할 수 있으며, 그 반대의 경우도 가능하다. 즉, 낮은 수준에서의 Wait 설정이라고 이해하면 된다.

Commit_Logging

Commit_Logging 파라미터는 LGWR 가 Redo Data 를 일괄로 기록할지의 여부를 결정하는 파라미터이다.

- Immediate : 이 값이 기본 값이며, 각 Commit 마다 LGWR 가 쓰기 작업을 진행함을 의미한다.
- Batch : 이는 LGWR 가 일괄로 Redo Data 를 기록함을 의미한다. 작은 단위의 Transaction 의 경우 이 방법으로 기록하면 보다 효율적인 방법이 될 것이다.

위에서 설명한 바와 같이 이 두 파라미터의 설정 값의 변경만으로도 Log File Sync 를 포함한 Commit Class 의 대기 이벤트들의 대기 시간이 획기적으로 감소할 수 있다는 가능성이 있음을 알 수 있다. 사실 이 기능이 처음 소개된 것은 10G 때부터 인데, 10G 에서는 파라미터가 Commit_Write 라는 파라미터만 존재한다.

설정 값을 Wait , Immediate 나 Nowait, Batch 등으로 설정한다. 11G 에 와서 파라미터가 위와 같이 두 가지로 분리되어 적용할 수 있도록 변경되었다.

Parameter 값 변경에 따른 성능 테스트

```
DECLARE
  l_dummy INTEGER;
BEGIN
  FOR i IN 1..1000
  LOOP
    INSERT INTO t VALUES (i, rpad('*',100,'*'));
    COMMIT;
    SELECT count(*) INTO l_dummy FROM dual;
END LOOP;
END;
```

본격적으로 두 파라미터의 값의 변경에 따른 위의 PL/SQL Block 과 같이 건건이 Commit 을 수행하는 트랜잭션의 성능차이가 어느 정도인지 테스트를 통해 알아보도록 하자.

참고 : 아래 결과는 STRACE 를 이용하여 서버 프로세스와 LGWR 의 System Call 횟수를 보여주는 것으로 별도 첨부된 Script 를 참조할 것.

WAIT / IMMEDIATE

```
***** Server Process *****
```

```
  % time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.069561          69      1005         1 semtimedop
------ ----------- ----------- --------- --------- ----------------
100.00    0.069561                  1005         1 total

***** Log Writer *****

  % time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.013919          14      1016           io_submit
------ ----------- ----------- --------- --------- ----------------
100.00    0.013919                  1016           total
```

NOWAIT / IMMEDIATE

```
***** Server Process *****

  % time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.002195         439         5         1 semtimedop
------ ----------- ----------- --------- --------- ----------------
100.00    0.002195                     5         1 total

***** Log Writer *****

  % time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.010073          10      1015           io_submit
------ ----------- ----------- --------- --------- ----------------
100.00    0.010073                  1015           total
```

NOWAIT / BATCH

```
***** Server Process *****

  % time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.002132         533         4         1 semtimedop
```

```
------ ----------- ----------- --------- --------- ----------------
100.00    0.002132                     4         1 total

***** Log Writer *****

% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.000533          36        15           io_submit
------ ----------- ----------- --------- --------- ----------------
100.00    0.000533                    15           total
```

두 파라미터의 값이 Default 값인 경우(Wait/Immediate), Server Process 와 LGWR 의 System Call 횟수는 Commit 횟수와 비슷한 것으로 나타났다. Nowait/Immediate 로 설정한 경우에는 Server Process 의 Call 횟수는 급감하였지만 LGWR 의 경우는 여전히 Commit 횟수와 비슷한 수치를 유지하고 있다.

반면에 Nowait/Batch 로 설정한 경우, 두 프로세스 모두 시스템 Call 횟수가 눈에 띄게 감소한 것으로 나타났다. 이것이 원인이 되어 Log File Sync 등의 Wait Event 대기 시간이 감소하여 전반적인 성능 향상의 결과로 나타난다.

주의 사항 및 결론

앞서 소개한 빈번한 Commit 에 의한 대기현상을 감소하는 방안은 분명 매력적인 방법이다. 하지만 이 역시 공짜는 아니다. ACID(Atomicity, Consistency, Isolation, Durability)라는 Transaction 의 특성 중 Durability(지속성)가 위배되는 상황이 발생할 수 있기 때문이다. 파라미터 값의 변경의 의미는 단순히 Commit 시 마다 LGWR 가 Redo Log File 에 기록하지 않고 일정량을 한 번에 기록하여 대기 시간을 감소시키려는 목적인데, 예기치 않은 Instant Failure 가 발생한다거나 DB Shutdown 상황이 발생하면 Transaction 의 완전한 복구는 사실상 불가능하다.

따라서, 금융 시스템과 같은 중요한 시스템에 해당 파라미터의 변경 적용은 고려사항이 아니다. 즉, DB 시스템의 성격과 업무의 특성을 고려하여 선별적인 적용이 필요함을 의미한다.

예를 들어, Data Migration 시에 세션 단위로 파라미터 변경 값을 적용하여 작업을 수행한다던지, 빈번한 Commit 이 발생하는 특정 업무(단, Transaction 의 지속성에 큰 영향을 받지 않는 Data 에 한함.)에 선별적으로 적용하는 등의 적절한 대처가 필요하다.

상황에 따라 영리하게 적용할 수 있다면 Commit Class 에 해당하는 Wait Event 들의 대기 시간을 감소시켜 성능을 개선할 수 있는 확실한 카드로 사용될 수 있다고 믿는다.

별도첨부 Script

Script 1. Comml.sql

```
SET DEFINE ON VERIFY OFF FEEDBACK OFF PAGESIZE 0 TERMOUT OFF ECHO OFF

ALTER SESSION SET commit_wait = &1;

ALTER SESSION SET commit_logging = &2;

CREATE TABLE t (n NUMBER, pad VARCHAR2(1000));

SPOOL servprc.pid
SELECT p.spid
FROM v$process p, v$session s
WHERE p.addr = s.paddr
AND s.sid = sys_context('userenv','sid');
SPOOL OFF

execute dbms_lock.sleep(2)

DECLARE
  l_dummy INTEGER;
BEGIN
  FOR i IN 1..1000
  LOOP
    INSERT INTO t VALUES (i, rpad('*',100,'*'));
    COMMIT;
    SELECT count(*) INTO l_dummy FROM dual;
  END LOOP;
END;
/

DROP TABLE t PURGE;

EXIT
```

Script 2. Commit.sh

```
#/bin/bash

user=$1
password=$2
commit_wait=$3
commit_logging=$4

lgwr_pid=`pgrep -f lgwr_$ORACLE_SID`

strace -p $lgwr_pid -c -e io_submit -o lgwr.log &

strace_pid=`pgrep -f strace`

sqlplus -s $user/$password @commit.sql $commit_wait $commit_logging &

sleep 1

strace -p `head -1 servprc.pid` -c -e semtimedop -o servprc.log
rm servprc.pid

kill -1 $strace_pid

echo
echo "***** Server Process *****"
echo
cat servprc.log
echo


echo "***** Log Writer *****"


echo


cat lgwr.log
```