Real Alternative DBMS ALTIBASE, Since 1999

ALTIBASE Memory 사용량 증가 분석가이드

ALTIBASE

2011. 04

Document Control

Change Record

Date	Author	Change Reference
2011-04-07	lim272	Created

Reviews

Date	Name (Position)

Distribution

Name	Location

목차

개요

ALTIBASE제품을 운영하는 과정에서 정상적인 데이터 증가에 따른 메모리 사용량 증가를 제외한 경우가 있을 때 사용자가 어떠한 사항들을 확인하고 조치해야 하는가에 대해 설명한다.

아래의 문서들을 미리 참고할 것을 권장한다.

1. 『ALTIBASE 개발자 가이드』
2. 『ALTIBASE SQL Tuning Guide』
3. 『ALTIBASE 모니터링 쿼리가이드』
4. 『OS별 유틸리티 사용가이드』
5. 『ALTIBASE 메모리테이블스페이스 관리』
6. 『ALTIBASE MVCC & GC』

Memory 사용량 일상적 점검

일상적 점검사항

사용자는 ALTIBASE 메모리 사용량에 대한 주기적인 관찰을 필요로 한다. 이것은 문제상황이 발생할 때 정상 대비 증가된 부분을 분석하여 원인을 추적하는데 중요한 정보로 활용이 가능하다.

1. 일상적인 시스템의 메모리/Swap 사용량

   OS별로 제공되는 프로세스 별 메모리 사용량을 조회하여 이를 기록/관리한다. (Swap도 동일한 방법을 통해 확인)

2. ALTIBASE 내부 모듈 별 메모리 사용량

   다음의 질의를 통해 획득한다.

```
iSQL> SELECT * FROM v$memstat;
```

사용자는 이 정보를 통해 다음의 관리항목을 생성할 수 있다.

일시	Memory(M)	Swap(M)	Storage_(M)	Query_(M)	Transaction_(M)
13:10:05	12,344	0	8,949	1,359	163
13:10:35	12,344	0	8960	1,650	163

V$memstat

V$memstat은 ALTIBASE가 제공하는 내부 모듈 별 메모리 사용량에 대한 Performance View이다. 항목들이 많지만 최소한 아래의 항목들을 모니터링 할 것을 권장한다.

주요항목	설명
Query_Prepare	질의문 자체, 실행계획, 통계정보, 바인딩정보 등 질의자체의 통합적 정보를 내부적으로 관리해야 하는데 이에 대한 저장공간 확보를 위해 메모리를 필요로 한다.
Query_Execute	질의를 실행하는데 필요한 메모리 공간으로 SELECT문과 같은 질의를 통해 전송할 데이터들이 저장될 공간을 할당하기 위해 메모리를 필요로 한다.
Query_Binding	질의에 Binding되는 변수가 저장되는 공간으로 질의문에 따른 호스트변수의 개수만큼의 저장공간 확보를 위해 메모리를 필요로 한다.
Storage_Memory_Manager	메모리 데이터가 저장되는 공간
Index_Memory	메모리 인덱스가 저장되는 공간

Transaction_OID_List	메모리 Ager에 의해 삭제될 대상들의 목록이 저장되는 공간으로 Aging작업이 지연될수록 이 항목의 메모리가 증가되는 현상을 보인다.

대표적인 메모리 증가 유형

앞서 설명한 "일상적 점검"항목을 기준으로 ALTIBASE의 메모리 증가 부분은 다음과 같이 정리할 수 있다.

1. 데이터 증가에 의한 경우
2. 실행되는 질의의 개수가 증가하는 경우
3. MVCC(Multi Version Concurrency Control)기법에 의해 증가되는 경우
4. Aging대상정보의 삭제가 지연 처리되는 경우

위와 같은 대표적인 경우들에 대해 원인과 조치방법에 대해 자세히 살펴보도록 한다.

Memory 사용량 증가에 대한 원인과 조치

데이터 증가

데이터 증가에 의한 경우는 메모리 테이블에 한정되는데 다음과 같은 질의를 통해 사용량 증가 추이를 살펴볼 수 있다.

```
iSQL> SELECT     a.table_name
                 , (b.fixed_alloc_mem + b.var_alloc_mem) alloc
                 , (b.fixed_used_mem + b.var_used_mem) used
      FROM    system_.sys_tables_ a , v$memtbl_info b
      WHERE a.table_oid = b.table_oid
      AND     a.table_type = 'T'
      ORDER BY 2 DESC;
```

위 질의에서 나온 결과 중 "Alloc"의 공간은 데이터가 저장된 공간 및 사용 가능한 공간을 모두 포함한다. ("Used"는 데이터가 저장된 공간만 의미한다.)

사용자가 "1G"의 데이터를 적재한 후 "900M"정도를 삭제했다면 해당 테이블에는 "1G"의 Alloc된 공간이 존재하며 "100M"의 Used공간이 포함되어 있고 나머지 "900M"는 해당 테이블만이 사용 가능한 빈 공간으로 존재하게 된다.

ALTIBASE를 사용하는 사용자의 대다수가 혼동하는 부분은 바로 이 "900M"에 대한 부분이다. 데이터가 삭제되었으니 "ALTIBASE 혹은 OS시스템에서 이 공간이 "Free"되어 메모리 사용량이 줄어들어야 정상이 아닌가?" 라는 의문을 갖는다.

OS의 메모리관리 구조는 대부분 한번 할당된 메모리에 대해 프로그램이 Free하여도 즉시 OS로 반납하지 않는다. 실제로는 메모리상의 Reserved영역에 할당해 놓은 후 다른 응용프로그램이 메모리가 부족한 시점에 Free된 영역을 사용하는 알고리즘을 일반적으로 사용한다. 즉, 언제라도 재사용시점에 메모리를 재 할당하는 과정의 비용이 크기 때문에 빈번한 메모리 할당/해제를 가급적 하지 않는다.

이러한 알고리즘 개념은 ALTIBASE에도 마찬가지로 적용된다. ALTIBASE 역시 한 번 사용된 공간은 데이터가 삭제되어도 테이블 내 재사용할 공간으로 유지한다.

사용자가 이러한 테이블을 발견했을 경우 조치방법은 다음과 같다.

1. 해당 테이블을 Compaction한다.

   메모리 테이블은 메모리 테이블스페이스로부터 공간을 할당 받게 된다. 하지만 하나의 테이블의 대량의 데이터 적재/삭제로 공간이 증가하면 동일 메모리 테이블스페이스에 존재하는 다른 테이블들이 할당 받을 공간이 부족해질 수 있음으로 더 이상 사용하지 않는 테이블의 공간을 동일 메모리테이블스페이스 내의 다른 테이블이 가용할 수 있도록 반납할 수 있는데 이를 Compaction이라 한다.

   ```
   iSQL> ALTER TABLE TargetTable COMPACT;
   ```

2. ALTIBASE를 재 구동한다.

   위와 같은 조치를 취하면 동일 메모리테이블스페이스 내의 다른 테이블들이 공간을 할당 받을 수 있도록 1 차적인 조치가 가능하다. 그러나 여전히 OS에서 점유한 메모리 사용량은 줄어들지 않는다. 이 경우 ALTIBASE를 재 구동하게 되면 메모리 사용량을 줄이는 것과 같은 효과를 나타낼 수 있는데 이는 다음과 같은 이유로 가능하다.

   원래의 재 구동과정에서는 모든 메모리 테이블스페이스의 데이터페이지를 메모리로 적재하지만 실제 데이터가 쓰여지지 않는 빈 페이지들은 메모리로 올리지 않기 때문에 메모리 사용량을 감소시키는 효과를 볼 수 있다.

   만일, 사용자가 일일이 테이블에 대한 Compaction을 하기 힘들다면 정기적인 PM과정에서 ALTIBASE를 2 회 재 구동을 주기적으로 할 것을 권장한다.

   첫 번째 시점에는 테이블들이 사용 중이지 않은 공간을 해당 테이블이 속한 메모리 테이블스페이스로 반납하는 과정을 수행하게 되고 두 번째 시점에는 테이블스페이스에 관리되는 빈 데이터페이지들이 메모리로 적재되지 않는 효과를 볼 수 있다.

질의의 증가

주기적으로 v$statement에 기록된 질의의 개수를 통해 확인이 가능하다. v$statement에 기록된 질의들은 현재 접속된 세션에서 수행한 질의만을 기록하기 때문에 정확하지 않으나 개략적인 개수를 파악하는 것에는 도움이 된다.

이 경우 업무적으로 필요한 질의의 증가임으로 사용자가 특별히 조치할 만한 사항은 없으나 다음의 사항을 개발자와 함께 확인할 경우들이 종종 발생할 수 있다.

1. 모든 질의는 사용 후 Close되는지 여부의 확인

   일반적으로 질의가 사용된 후에는 해당 질의에 대한 객체를 종료해야 한다.

```
Ex) JAVA / JDBC
Connection          cn;
prepareStatement    ps;

ps = cn.prepareStatement ("select…");
….
ps.close();
```

   위의 짧은 코드에서 만일, "ps.close"와 같은 구문이 존재하지 않으면 ALTIBASE 내부 모듈 중 "Query_Prepare"등에 해당 질의에 대한 정보유지를 위해 메모리를 지속적으로 유지하고 이러한 상황이 반복되면 세션이 종료되지 않는 이상 메모리는 계속 사용되게 된다.

   한번 사용된 메모리는 ALTIBASE 자체적으로 Free시켜도 실제 OS에서 바로 회수하지 않기 때문에 결과적으로 메모리 사용량이 늘어난 상태로 운영되게 된다.

   따라서, 응용프로그램 수행 후 정상적으로 질의수행에 의해 사용된 응용프로그램의 객체에 대한 종료가 정상적으로 잘 수행되는지 여부를 확인하도록 해야 한다.

2. 비슷한 질의가 계속 사용되는 경우의 확인

   다음의 예를 살펴보자.

```
SELECT 1 FROM DUAL WHERE C1 = 1;
SELECT 1 FROM DUAL WHERE C1 = 2;
SELECT 1 FROM DUAL WHERE C1 = 3;
```

   위 질의는 "SELECT 1 FROM DUAL WHERE C1 = ?"과 같은 구문으로 대체하고 조회 조건의 변수 값만 바뀌는 형태로 개발하면 하나의 문장으로 해결할 수 있다.

   위와 같은 비슷한 질의들을 누적해서 실행할 경우 ALTIBASE의 현재 버전에서는 각각 다른 질의로 분석하여 메모리를 할당하게 된다.

MVCC기법에 의한 증가

MVCC기법은 조회/변경연산이 서로 대기하지 않도록 하여 성능을 높이는 DBMS의 일반적 동시성 제어기법을 의미한다. 현재 ALTIBASE 메모리 테이블은 변경연산 시점에 레코드의 복제 본을 생성하고 그 복제 본에 변경된 정보를 기록하는 형태로 동작한다.

따라서, 레코드에 대한 변경시점에는 원본과 복제 본이 함께 존재하는 형태가 된다. (삭제 시점에는 위와 같은 메모리증가는 없으며 Aging지연에 의한 증가가 발생할 수 있다.)

이는, 대량의 변경작업을 수행하면 변경작업 대상 레코드의 개수만큼 복제 본이 생성된다는 것을 의미한다. (빠른 성능만큼 리소스의 사용이 증가되는 trade-off관계를 가진다.)

이러한 복제 본의 생성은 해당 테이블의 공간에 생성하게 된다. 따라서, 대량의 변경작업이 발생하면 해당 테이블의 사용량이 증가하게 된다. 만일, 테이블에 빈 공간이 존재하면 해당 공간이 사용되게 되고 부족하거나 없다면 테이블스페이스로부터 공간을 할당 받아 사용하게 된다.

사용자는 테이블 별 사용량 결과를 이용하여 지나치게 Alloc/Used의 차이가 현격한 테이블을 선별하고 업무적으로 사용량에 대한 이상징후가 발견되면 조치하도록 해야 한다. 조치할 수 있는 방안은 앞서 설명한 테이블 Compaction이나 혹은 재 구동과 같은 방법을 사용해야 한다. 또한, 일반적으로 DBMS 작업과정에서 대량의 변경작업은 권장하는 작업방식이 아니기 때문에 일정 레코드 단위로 변경작업을 수행할 것을 권장한다.

예를 들어, 천만 건의 레코드를 변경해야 한다면 몇 만 건씩 단위작업으로 나누어 변경작업을 수행하도록 한다. (LIMIT절 사용)

Aging 지연에 의한 증가

앞서 설명한 MVCC에 의해 생성된 복제 본이 Commit시점에는 최종 레코드가 될 것임으로 원본 레코드는 삭제 대상이 된다. 만일, Rollback이 되면 복제 본은 삭제 대상이 된다. (이러한 삭제 대상 레코드를 "Old Version" 혹은 "Garbage Data"라고

하며 삭제 하는 작업을 내부적으로 Aging이라는 용어를 사용하고 GC (Garbage Collector)라는 내부 모듈이 담당)

문제는 이러한 삭제 대상을 어떠한 이유로 삭제하지 못하는 경우가 있을 수 있는데 이런 상태가 지속되면 계속 삭제 대상이 누적될 것임으로 메모리 사용량 증가를 유발할 수 있게 된다.

GC가 정상적으로 Aging을 하는지 여부는 다음의 질의를 통해 확인이 가능하다.

```
iSQL> SELECT add_oid_cnt, gc_oid_cnt FROM v$memgc;
```

"add_oid_cnt"는 트랜잭션들이 Aging대상으로 넘겨준 대상을 의미한다. 즉, Aging을 요청한 트랜잭션의 수와도 같으며 "gc_oid_cnt"는 GC가 처리한 Aging대상의 개수를 의미한다.

위 수치가 계속 변경된다면 처리속도의 지연이 있더라도 GC가 정상적으로 작업을 한다고 볼 수 있다. 그러나 특정시점에 "gc_oid_cnt"가 증가되지 않는다면 문제가 발생한 것으로 간주할 수 있다.

GC모듈은 삭제대상 레코드가 어떠한 트랜잭션에 의해서도 참고되지 않아야 삭제할 수 있으며 삭제 대상 목록이 요청된 순서대로만 삭제처리를 진행할 수 있다. 따라서, GC에 문제가 있다고 판단되면 어떤 트랜잭션이 여전히 종료되지 않은 상태로 남아 있는 유형의 문제임으로 해당 트랜잭션을 찾아 정리할 필요가 있다. 이러한 질의는 다음과 같은 질의로 찾을 수 있다.

```
iSQL> SELECT a.session_id, b.query
        FROM v$transaction a, v$statement b, v$memgc c
        WHERE a.id = b.tx_id
        And     (a.memory_view_scn = c.minmemscnintxs
                 Or a.min_memory_lob_view_scn = c.minmemscnintxs)
```

위 질의를 수행하면 해당 질의를 포함하여 문제가 되는 질의정보를 보여준다.
모니터링질의를 제외한 다른 질의가 나타난다면 해당 질의의 수행시간이 과도한 경우라면 튜닝 하거나 혹은, 사용자가 질의 수행 후 정상 종료를 하지 않았을 경우임으로 해당 프로그램의 소스를 확인하여 정상 종료가 가능하도록 조치해야 한다.

※ MVCC, GC와 관련된 메모리 증가유형은 메모리테이블에 한정된다.

알티베이스㈜
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 1008 호
02-2082-1000
http://www.altibase.com

대전사무소
대전광역시 서구 둔산동 921
주은리더스텔 901 호
042-489-0330

기술지원본부
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 908 호
02-2082-1000

기술지원센터
02-2082-1114
http://support.altibase.com