Real Alternative DBMS ALTIBASE, Since 1999

# ALTIBASE CPU과부하 현상에 대한 분석가이드

ALTIBASE

2011. 04

# Document Control

## Change Record

| Date | Author | Change Reference |
|---|---|---|
| 2011-04-01 | lim272 | Created |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Reviews

| Date | Name (Position) |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

## Distribution

| Name | Location |
|---|---|
| | |
| | |
| | |
| | |

# 목차

# 개요

시스템의 CPU부하는 기본적으로 응용프로그램(User) 및 커널(Sys/Kernel)에 의해 발생한다. 커널에 의해 발생되는 경우들은 일반적으로 시스템 리소스 사용에 대한 이벤트가 발생할 때이며 그 외의 경우들은 대부분 응용프로그램에 의해 사용된다.

ALTIBASE의 경우는 사용자가 발생 시킨 질의처리(트랜잭션)를 위해 다양한 알고리즘이 적용되어 있고 DB에 접속한 클라이언트와의 통신 및 이중화, 데이터파일의 처리에 의한 디스크I/O등 다양한 시스템 리소스 사용이 발생함으로 User영역과 Sys/Kernel영역의 CPU사용이 모두 발생하게 된다.

본 문서에서는 ALTIBASE가 점유하는 CPU사용률이 높아질 때 어떤 부분들을 살펴보고 정상/비정상 유무를 판단할 것 인지와 비정상적인 경우 어떤 형태로 해결할지에 대해 설명한다.

다음의 문서를 사전에 참고할 것을 권장한다.

1. 『ALTIBASE 개발자 가이드』
2. 『ALTIBASE SQL Tuning Guide』
3. 『ALTIBASE 모니터링 쿼리가이드』
4. 『OS별 유틸리티 사용가이드』

# CPU 과부하 현상에 대한 분석

서비스 운영 중에 갑작스런 CPU의 과부하에 의한 서비스 지연이 발생할 수 있는데 이 과정에서 지속적으로 ALTIBASE CPU 과부하에 의한 현상이 발생할 때 사용자가 확인할 부분과 해결방안에 대해 설명한다.

## 일상적 점검사항

사용자는 현재의 CPU부하의 정상유무를 판단하기 위해 다음의 이력관리가 필요하다. (본 문서를 작성하는 시점의 ALTIBASE버전은 5.5 이며 자체적인 CPU사용률 정보를 제공하는 기능은 아직 없음)

| 항목 | 수집방안 |
|---|---|
| 시간대별/일별 CPU사용률 | OS가 제공하는 프로세스의 CPU사용률 획득방법을 통해 주기적으로 기록/관리한다. |
| 시간대별/일별 트랜잭션 처리량 | 응용프로그램의 트랜잭션 처리량을 기록/관리한다.<br>이것이 어려울 경우 ALTIBASE에서 제공하는 v$sysstat을 통해 다음과 같은 정보를 주기적으로 획득하여 기록/관리한다.<br>iSQL> SELECT * FROM v$sysstat<br>WHERE name LIKE '%prepare%count%'<br>OR name LIKE '%execute%'; |
| 서비스쓰레드의 증감여부 | 평소보다 세션의 증가 혹은, 업무량의 증가로 인해 서비스쓰레드가 증가하는 경우가 있는데 이런 패턴이 발생하는지를 체크한다.<br>iSQL> SELECT COUNT(*) as Thread FROM v$service_thread;<br>iSQL> SELECT COUNT(*) as Session FROM v$session; |

위 이력정보를 통해 사용자는 다음과 같은 표 형태의 이력을 만들 수 있을 것이다.

| 시간(HH:MI:SS) | CPU (%) | Session / Thread | PRPARE (개수) | EXECUTE (개수) |
|---|---|---|---|---|
| 10:10:00 | 12.3 | 208/16 | 10 | 300,812 |
| 10:10:30 | 24.8 | 208/16 | 10 | 310,220 |
| 10:11:00 | 32.1 | 212/17 | 15 | 334,400 |

이와 같은 형태의 자료를 통해 일상적인 시점과 CPU과부하 시점의 부하차이가 트랜잭션 처리량의 증가로 인한 것인지를 먼저 판별할 필요가 있다.

## 일반적 분석절차

먼저, 아래 2 개의 상태를 점검한다.

1. 평상 시보다 더 많은 트랜잭션 처리가 발생했는가?

A. 만일, "일상적 점검" 사항에 해당하는 "세션", "서비스쓰레드", "Prepare/Execute"의 항목이 모두 증가수치를 보이면서 CPU사용량이 증가했다면 이는 정상적인 증가패턴이라 할 수 있다.

B. 다음의 사항을 정상패턴과 구별하여 주의해야 한다. 사용자의 검증되지 않은 질의(장시간 수행되거나 대량의 변경작업을 유발하는 성격) 및 서비스로 인해 서비스쓰레드의 증가가 발생할 경우가 있다.

이는 서비스쓰레드의 자체적인 load-balancing기능에 의해 증가하는 유형이다. 패턴 자체는 정상적인 경우이며 이와 같이 증가되는 질의가 수행된 원인을 찾아야 하는데 보통 사용자가 임의로 수행한 질의가 수행되어 발생한 경우는 해당 시점이 아니라면 발견하기 어렵기 때문에 오랜 시간 동안 수행되는 질의가 존재하는지 주기적으로 다음 질의를 통해 서버를 관찰할 필요가 있다.

```
iSQL> SELECT a.session_id, a.client_pid, b.query, b.execute_time, b.fetch_time
        FROM   v$session a, v$statement b
        WHERE  a.id = b.session_id
        AND    (b.execute_time > 1000000 or b.fetch_time > 1000000);
```

(실행시간 혹은, 결과 셋을 가져오는 시간이 1 초 이상인 질의를 탐색)

2. 신규로 추가된 서비스(응용프로그램)가 발생했는가?

A. 직접적인 원인일 가능성이 높음으로 새롭게 추가된 질의들에 대한 실행계획을 먼저 분석하고 실행시간과 Access-Cost가 높은 질의에 대해 적절한 튜닝작업을 진행하도록 해야 한다.

위에 언급되어 있는 악성질의 탐색방법을 통해 주기적인 관찰이 필요하다.

위 2 가지 사항은 모두 결과적으로 서비스를 수행하는 응용프로그램들의 구조나 수행하는 질의가 대부분 잘못 구현된 경우들이다. (아래 설명할 ALTIBASE의 질의처리과정의 CPU사용에 대해 참고)

그러나, (1, 2)번의 항목을 확인하면 대부분 사용자들은 "그렇지 않다."가 일반적인 답변이다. 그러면, 서비스 형상과 시스템에 아무런 변화가 없는 상태에서 ALTIBASE버전도 변경이 없었다면 CPU과부하의 원인은 어디에 있을까? 아래와 같은 확인절차가 필요하다.

3. 기 서비스중인 응용프로그램의 문제분석

어떠한 서비스 운영의 형상변경이 없음에도 불구하고 CPU과부하를 일으킨다면 이는 데이터의 증가와 같은 원인으로 기존 질의가 본래 가지고 있던 문제점이 드러나는 경우를 가정해 볼 수 있다.

예를 들어, 기존에 사용하던 인덱스가 데이터규모가 작은 상황에서는 원활히 수행되었으나 시간이 경과하면서 데이터규모가 증가하고 이런 상황에서 적절하지 않았던 기존 인덱스가 사용된다면 CPU사용률이 증가하는 원인이 될 수 있음을 의미한다.

## 질의처리 과정의 CPU비용

전반적인 질의처리과정에서 ALTIBASE가 CPU를 쓰게 되는 부분들에 대해 간략하게 설명한다.

1. ALTIBASE-응용프로그램간의 구현 형태에 따른 CPU부하

   ALTIBASE의 질의처리는 크게 아래와 같은 과정을 거쳐 수행된다.

   1.1 질의의 문장/문법 오류 체크

   1.2 질의에 사용되는 테이블/칼럼의 존재유무 체크

   1.3 최적의 실행계획 수립 (인덱스의 선택)

   1.4 물리적인 데이터영역에 접근하여 필요한 연산처리를 통해 데이터를 획득 및 변경

   이런 과정 중 (1.1~1.3)을 Prepare-Validation-Optimization (PVO)라고 지칭하며 (1.4)의 과정을 EXECUTE라고 한다. 일반적으로 PVO를 수행하는 과정이 질의전체의 처리과정에서 60~70%정도의 비중을 차지한다.

   위 가정을 전제로 "동일한 질의를 매번 PREPARE-EXCUTE" 형태로 수행하는 게 얼마나 비효율적인 CPU사용을 유발하는지 짐작할 수 있다. 따라서, 개발자나 운영자는 v$sysstat을 통해 다음의 항목이 지나치게 증가하는 경우라면 반드시 응용프로그램의 반복적인 PREPARE를 수행하지 않도록 수정하거나 혹은 ALTIBASE버전 5 부터 제공하는 PLAN-CACHE기능을 활용할 것을 권장한다.

   ```
   iSQL> SELECT * FROM v$sysstat WHERE name like '%prepare%count%';
   ```

   출력된 "value"칼럼 값이 계속 증가하는 경우라면 매번 PREPARE형태로 질의가 처리된다 볼 수 있음으로 이에 대한 수정이 가장 먼저 진행되어야 한다.

   위와 관련된 자세한 사항은 "ALTIBASE 환경의 개발 시 고려사항 가이드" 문서를 참고하도록 한다.

2. 메모리테이블의 데이터접근

   ALTIBASE DBMS의 메모리테이블은 물리적인 시스템메모리에 모두 적재된 상태로 유지된다. 저장되는 단위는 32Kbyte의 페이지 단위로 저장되고 이 페이지 내에서도 레코드 저장에 적합한 테이블 별 개별단위(Slot)로 분할되어 있다.

   문제는, 테이블에 Full-Scan형태로 접근할 경우 테이블에 존재하는 모든 데이터에 대한 접근비용이 매우 크게 발생한다. 이런 질의가 동시에 여러 세션에서 수행한다면 CPU사용률은 급증할 수 밖에 없다.

   그럼, 인덱스를 사용하는 경우는 반드시 CPU사용률은 낮아지는가?

   반드시 그렇지는 않다. 메모리테이블의 인덱스 역시 시스템메모리에 존재한다. 하지만, 메모리테이블의 인덱스에는 실제 데이터 값이 존재하지 않는다. 다만, 시스템메모리상의 데이터가 위치한 물리적 포인터 값을 가지고 있고 이 포인터 값이 인덱스 키값과 함께 연산되어 변환된 값 형태로 정렬되어 존재한다.

   따라서, 인덱스가 가리키는 실제 물리적인 데이터에 매번 접근하여 데이터 값을 획득하고 비교하는 과정을 거치게 된다. (이 과정이 불합리하게 보일지 모르지만 상대적으로 성능측면에서는 디스크I/O를 유발하지 않아 다른 DBMS가 I/O로 인한 대기시간이 요구되는 상황에서도 CPU를 점유하여 더 많은 트랜잭션의 처리가 가능하다 할 수 있다.)

따라서, 인덱스를 이용한 질의라 하여도 Access-Cost가 많을수록 CPU사용률은 상대적으로 증가하게 되기 때문에 사용자는 정기적으로 서비스 운영에 수행되는 질의의 실행계획을 모니터링 하여 비효율적인 질의를 제거하는 노력을 수행해야 한다.

3. 디스크테이블의 데이터접근

   디스크테이블에 대한 설명은 다음과 같이 I/O측면의 CPU사용에 대한 이해를 필요로 한다. 일반적으로 OS는 File-Cache라는 메모리영역을 할당하여 파일I/O가 빈번하게 발생하여도 미리 메모리상에 적재해둠으로써 I/O성능을 보장하려 한다.

   하지만, 대부분의 RDBMS는 자체적인 버퍼를 중간에 두어 이러한 효과를 얻어내기 때문에 일반적으로 파일시스템이 Direct I/O를 지원한다면 File-Cache를 사용하지 않고 Direct I/O를 이용하도록 권장하고 있다.

   Buffered I/O는 응용프로그램의 코드상으로 Read/Write System-Call처리에 대한 CPU자원을 이용하고 OS정책에 의해 File-Cache에 기록된 내역들이 디스크에 Sync되는 과정에서 CPU자원이 소모되는데 이 시점에 I/O작업이 완료될 때까지 대기가 발생하고 CPU사용률은 낮아지는 것처럼 보이지만 이때의 CPU자원을 다른 응용프로그램이 사용할 수 없음으로 서비스 처리는 감소할 가능성을 가지고 있다.

   이를 반대로 생각하면 Direct I/O를 이용할 경우 CPU는 I/O자체를 I/O전담채널에 일임하게 된다. 따라서, CPU가 I/O처리를 위해 대기할 필요가 없어지고 그 자원을 다른 응용프로그램들이 점유할 수 있으며 상대적으로 서비스 처리도 그만큼 더 수용 가능하다. 하지만, 상대적으로 Buffered I/O에 비해 CPU사용률은 마치 증가한 것과 같이 사용자가 판단할 수 있다.

   요약하면 Buffered I/O 또는, Direct I/O의 설정은 모두가 CPU사용과 성능에 대해 Trade-off를 가지기 때문에 어느 것이 적합하다는 답은 없으며 사용자의 서비스환경에 맞게 적절히 설정하도록 한다.

앞에서 설명한 몇 가지 과정들이 모두 CPU자원을 필요로 하고 결과적으로 질의처리과정 자체가 CPU를 낮게 사용하면서 빠른 성능을 내야 한다는 것은 개념적으로 불가능하다. 다만, 효율적으로 CPU를 사용하면서 요구되는 성능을 발휘하는 것은 사용자/개발자의 노력에 따라 가능한 일이라 할 수 있겠다.

## 기타 사례를 통한 CPU문제들

성능이 원하는 수준은 아니지만 CPU도 예상 수치만큼 사용하지 않는 경우들이 발생할 수 있다. 이러한 현상들은 다음과 같은 경우들을 고려해야 한다.

1. 문자 셋(NLS_USE)의 설정이 DB와 응용프로그램이 다른 경우 내부적인 문자 셋의 변환작업(Encoding)으로 인해 ALTIBASE 성능이 저하되고 CPU사용량이 줄어드는 현상이 보고된 사례가 있다.

   이 경우 DB서버와 응용프로그램간의 문자 셋을 일치시켜 문제를 해결한다.

2. 4.3.x버전대는 “Dedicated Thread”의 증가로 인한 “Select-Poll” system-call에 의한 CPU사용량이 증가하는 경우가 보고된 사례가 있다.

ALTIBASE는 세션의 접속을 "Service Thread"가 담당하여 트랜잭션을 처리하는데 Long-run질의가 발생하거나 모든 "Service Thread"가 실행(Executing)상태가 되면 새로운 접속이나 대기하는 트랜잭션들을 처리하기 때문에 별도의 새로운 "Dedicated Thread"를 실시간으로 생성하게 된다.

문제는 새롭게 생성된 "Dedicated Thread"들이 자신이 처리할 트랜잭션이 존재하는지 주기적으로 체크하게 되는데 이 과정에서 ALTIBASE는 "Select-Poll" system-call이 많아지고 CPU사용량 중 sys(%)사용량이 증가하는 패턴을 보인다.

이런 유형에 대비하여 "MULTIPLEXING_POLL_TIMEOUT"이란 속성값을 증가시키는 방안이 있는데 이것은 "Select-Poll" system-call의 수행주기를 증가시켜 system-call의 횟수를 감소시키는 것이다.

근본적으로 사용자는 "Dedicated Thread"가 발생하는 이유가 세션의 증가, 트랜잭션의 부하증가, 다수의 Long-Run질의, 락 경합 등에 의한 경우임으로 이런 상황들을 원천적으로 해소할 개선방안에 대해 고민해야 한다.

3. OS 환경변수 설정에 따른 CPU사용량의 변화가 있을 수 있는데 ALTIBASE는 멀티쓰레드 구조를 채택하기 때문에 OS의 설정에 대해 이와 관련된 설정을 적절하게 할 필요가 있다.

   해당 설정은 CPU사용량의 감소측면이 아니라 CPU를 사용하는 만큼의 성능을 나타내지 못하는 경우들에 대한 설정으로 이해하면 된다.

   자세한 사항들은 OS별로 제공되는 "ALTIBASE 설치가이드"를 참고한다.

4. ALTIBASE의 v$statement에는 현재 접속된 세션이 수행한 질의에 대한 실행정보를 가지고 있다. 따라서, Connect/Disconnect의 반복적 수행이 발생하는 구조에서는 이 정보에서 악성질의를 탐색하기가 매우 어렵다.

   더더욱, 매번 Connect/Disconnect의 비용만큼 응용프로그램의 성능도 저하될 수 밖에 없다. 따라서, 사용자는 이와 같은 응용프로그램의 구조를 DB에 연결된 상태에서 처리되도록 변경하는 것이 바람직하며 이것이 불가능할 경우 DB의 성능저하를 감수하더라도 다음과 같이 프로파일링 기능을 통해 악성질의를 탐색해야 한다.

```
iSQL> ALTER SYSTEM SET TIMED_STATISTICS=1;
iSQL> ALTER SYSTEM SET QUERY_PROF_FLAG=1; (기능 실행)
iSQL> ALTER SYSTEM SET QUERY_PROF_FLAG=0; (기능 종료)
```

   위 기능을 수행한 이후 수행된 모든 질의는 $ALTIBASE_HOME/trc/xxxxx.prof라는 확장자가 "*.prof"라는 파일명으로 기록이 된다. 이 파일은 바이너리 파일로 만들어져 사용자가 바로 확인할 수 없으며 다음과 같이 일반 텍스트형태로 변환을 해야 한다.

```
Unix Prompt> altiProfile alti-13109987-0.prof > log1.txt
```

   "altiProfile"은 ALTIBASE에서 제공되는 분석유틸리티이며 이를 이용해 분석용 파일을 만들 수 있다. 해당 파일에는 실행된 "질의문","수행시각의 상세정보" 등이 기록되어 있기 때문에 해당 내역을 기반으로 수행시각이 오래 걸린 악성질의를 찾을 수 있다.

   ※ 프로파일링 기능은 DB의 성능저하 및 로그기록으로 인한 디스크부족을 유발할 수 있음으로 사용에 신중한 주의가 필요함.

## Summary

앞에서 설명한 대로 각 항목들에 대해 사용자가 어떻게 일상적으로 점검하고 CPU과부하에 대한 원인을 찾을지에 대해 정리한다.

1. Performance View를 통한 탐색

   A. Application측면의 구조개선이 필요한지 파악

```
-- 빈번한 connect/disconnect을 수행하는지에 파악 (빈번한 connect/disconnect은 성능감소 유발)
iSQL> SELECT name, value FROM v$sysstat WHERE name LIKE '%logon cum%';

-- 반복적인 PREPARE과정을 수행함으로 CPU를 비효율적으로 사용하는지 파악
iSQL> SELECT name, value FROM v$sysstat WHERE name LIKE '%prepare%count%';
```

   B. 악성질의 여부 파악

```
-- 사용자가 설정한 적정시간을 기준으로 악성질의를 탐색
-- 자주 수행되면서 느린 질의를 탐색하거나 시간만을 기준으로 하여 탐색
iSQL> SELECT execute_time, query FROM v$statement
        WHERE execute_time >= 1000000;
        (ORDER BY execute_success DESC, execute_time DESC) - 사용자가 Ordering을 지정
```

   ※ Performance View의 각 칼럼에 대한 자세한 설명은 "ALTIBASE 모니터링 쿼리가이드" 문서를 통해 확인할 수 있다.

   ※ 발견된 악성질의에 대한 실행계획 확인 및 튜닝방안은 "ALTIBASE SQL Tuning Guide" 문서를 통해 확인할 수 있다.

   C. "Service Thread"의 증가

```
-- 버전 별로 차이는 있지만 "Shared Thread" / "Dedicated Thread"의 개수 증가 관찰이 필요
iSQL> SELECT * FROM v$service_thread;
```

2. OS 유틸리티를 통한 탐색

   A. ALTIBASE는 Multi-Thread Architecture를 채택하고 있다. 따라서, 내부의 특정 쓰레드가 비정상적으로 CPU를 많이 사용하지 않는지 파악해볼 필요가 있다. 이 경우 OS별로 제공되는 유틸리티를 통해 쓰레드별 CPU사용량 및 현재의 상태를 알아낼 수 있는데 관련된 자세한 방법은 다음 문서를 참고하도록 한다. ("문제분석을 위한 OS별 유틸리티 가이드" 문서 참고)

**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