Advanced OWI in Oracle 10g - Library Cache에서의 대기이벤트들

Library Cache에서의 대기이벤트들
latch: shared pool
shread pool 래치는 - Shared Pool의 기본 메모리 구조인 힙을 보호
-> 프리 청크를 찾기위해 프리리스트 탐색 - 청크 할당 - 청크 분달등 모두 shared pool 래치를 획득한 후에만 작업 가능
경합시 latch: shared pool 대기 이벤트
일반적으로 힙 안에서 동작하는 작업들은 매우 빠른 시간 안에 완료되기 때문에 shared pool 래치 경합이 나타나지 않는다
그런데도 발생하는 경우라면
- shared pool 래치는 기본적으로 전체 인스턴스에 하나만이 존재한다. 즉 하나의 Shared Pool에 하나의 shared pool 래치가 사용된다. 이것은 Shared Pool의 기본메모리 구조인 힙의 아키텍처에 기인하는 것이다. 따라서 동시에 여러 세션이 청크를 할당받아야 하는 경우 하나의 shared pool 래치를 획득하기 위해 경쟁하기 때문에 경합이 발생하게 된다.
- 하드파싱이 심한 경우 청크를 쪼개는 (Split) 현상이 자주 발생하고 이로 인해 프리리스트에 수많은 작은 크기의 프리 청크들이 붙는 현상이 발생한다. 이러한 현상을 흔히 Shared Pool의 단편화(fragmentation)라고 부른다. Shared pool 래치를 보유하는 시간이 늘어나게 된다. Shared Pool 단편화는 shared pool 래치 경합을 발생시키는 근본적인 원인이다. ORA-4031 에러 또한 단편화로 인해 발생한다.
하드파싱이 발생하면 새로운 SQL정보를 담을 청크를 항당받기 위해 프리리스트를 탐색해야함 -> 이 과정에 오직 하나의 프로세스만이 shared pool 래치를 독점하게된다 -> 따라서 많은 세션이 동시에 하드 파일을 한다면 latch: shared pool 대기 이벤트가 발생하게 된다.
그렇다고 Shared Pool의 사이즈를 키운다면 -> 프리리스트의 개수와 프리리스트가 관리해야할 청크의 수가 증가한다 -> 따라서 탐색하는 시간동안 shared pool 래치를 보유하고 있어야한다. -> 그만큼 latch: shared pool이 증가하게 된다. => 하드파싱에 의해 latch: shared pool 대기가 증가하는 경우 Shared Pool의 크기를 늘려주는 것은 오히려 문제를 악화시킨다.
- 최선의 해결책은 바인드 변수사용하여 하드파싱 줄이기
하드파싱 때문에 shared pool 래치 경합 클 때 해결법
1. 9i 이상부터는 Shared Pool을 최대 7개 까지의 서브 풀로 나누어 관리할 수 있다. CPU가 4개 이상이고, Shared Pool의 크기가 250M 이상인 경우 _KGHDSIDX_COUNT의 값만큼 서브풀 생성 관리한다.
2. Shared Pool의 크기를 줄인다 -> 검색해야 하는 프리청크 감소시켜서 -> 프리리스트 탐색 소요시간을 줄인다 || 단 shared pool 공간이 줄어서 하드파싱이 유발될 수도 있다 -> dbms_shared_pool.keep 프로시져를 이용해 shared pool에 영구히 상주시키는 등의 해결책이 있다 - 이 경우 alter system flush shared_pool 명령으로도 내려가지 않는다.
3. Cursor Sharing 기법 -> 상수를 사용한 SQL 문장을 자동으로 바인드 변수를 사용하게끔 치환해서 커서가 공유되도록 해주는 기능 -> 테스트 하고 사용하도록하라. 실행계획등이 변경될수도 있다
주의사항
하드 파싱 발생시 -> library cache 래치 획득 -> library cache 영역 탐색 -> 이 상태에서 shared pool 래치 획득 -> 즉, library cache 래치 획득할때 벌써 경합이 발생하여 병목 현상이 발생한다면, share pool 래치 경합은 되려 줄어드는 경우도 볼 수 있다.

latch: library cache
SQL을 수행하기 위해 Library Cache 메모리 영역을 탐색하고 관리하는 모든 작업을 보호한다.
library cache 래치를 획득하는 과정에서 경합 발생시 latch: library cache 대기 이벤트
library cache 래치는 CPU count보다 큰 소수 중 가장 작은 수만큼 생성된다
주요 발생 원인
- 하드파싱이나 소프트파싱이 과다한 경우
- 버전 카운트(Version count)가 높은 경우
- SGA 영역의 페이지 아웃(Page out)이 발생하는 경우

하드파싱이나 소프트파싱이 과다한 경우
하드파싱 발생시 -> Library Cache 탐색 회수 증가로 library cache 래치 보유 시간과 회수 증가하게 됨 || 추가적인 청크 할당도 필요하므로 그만큼 libaray cache 래치 보유 시간 증가
하드파싱과 래치 경합 상관관계는 shared pool 래치의 경우와 비슷 -> 헌데 library cache 래치 경합은 소프트 파싱일 때에도 발생할 수 있다.
소프트파싱이라 하더라도 -> syntax체크, semantics체크, library cache 탐색 등의 작업은 해야하기 때문이다.
소프트파싱으로 인한 library cache 래치 경합 해결법
1. 파싱회수 자체를 줄인다 -> 어플리케이션 수정, Static SQL사용
2. SESSION_CACHED_CURSORS파라미터 이용 -> 이 값이 세팅되어 있으면, 세번 이상 수행된 SQL 커서에 대한 정보를 PGA 내에 보관한다. -> PGA에 캐시된 정보가 있다면 library cache 탈 필요없이 바로 지정된 버퍼 주소로 접근하게된다.
이 기능은 세션단위로만 동작함 -> Connection Pool 같은 기법을 이용하여 세션 접속을 유지시켜준다면 성능이 더욱 좋아질 것이다.

버전 카운트(Version Count)가 높은 경우
동일한 문장을 서로 다른 유저가 실행했드라~ -> 문장은 같지만 스키마가 다르기땜에 이들은 같으면서 다른문장 -> 이 경우 오라클은 부모 LCO에 문장을 저장하고 각각의 자식 LCO를 만들어 개별 SQL정보를 저장한다 -> 각각에대한 자식 LCO를 가지므로 V$SQLAREA의 VERSION_COUNT컬럼 수치가 높다 -> 즉, 버전카운트가 높다는 것은 자식 LCO 탐색 작업으로 인해 library cache를 탐색하는 시간이 그만큼 증가한다는 것.

SGA 영역의 페이지 아웃(Page Out)이 발생하는 경우
- > Shared Pool이 디스크로 페이지 아웃 되었다면, 해당 부분을 다시 스캔할때 -> 해당 내용을 다시또 메모리로 불러들이는 과정 -> 이 과정동안 대기해야한다 -> 따라서 library cache 래치에 대한 대기 시간이 증가할 수 있다.
latch: library cache 대기가 높은 상태에서 O/S에서 스왑 현상이 발생한다면, 페이지 아웃 때문일 가능성 높음

library cache lock과 library cache pin
library cache lock -> Library Cache 객체(SQL커서, 프로시져, 패키지 등)에 접근하거나 변경할때 library cache 핸들에 대해 획득하는 락. (경합시 library cache lock 대기 이벤트)
SQL문 수행시 : library cache 객체에 대해 파싱을 수행하는 동안에는 library cache lock을 Shared모드로 획득 -> 파싱이 끝나면 library cache lock을 Null모드로 변환 후 계속 보유
SQL문이 참조하고 있는 모든 객체(테이블, 뷰, 시노님등)에 대해서도 동일한 모드로 library cache lock획득
-> 즉 여러 프로세스가 동시에 같은 SQL문 수행 가능 ->

프로시저 생성 변경시 : 프로시저에 해당하는 library cache 객체에 대해 library cache lock을 Exclusive하게 획득.
테이블 변경(alter)시 : 테이블에 해당하는 library cache 객체에 대해 library cache lock을 Exclusive하게 획득.
SQL 문이나 프로시저 수행시 library cache lock을 Null모드로 바꾸고 계속 보유하는 이유 : library cache 객체의 무효화를 자동화하기 위함

library cache pin -> Library Cache 객체에 대한 수행이나 변경시 library cache object (LCO)에 대해 획득하는 락.
library cache lock이 LCO의 명세를 보호한다면, library cache pin은 LCO의 내용을 보호한다.
library cache pin : library cache lock을 획은한 후에 library cache 객체에 대해 추가적인 작업이 필요할 때 획득
SQL문 수행 : library cache lock Shared모드로 획득 후, library cache pin도 Shared모드로 획득
프로시저 컴파일 : library cache pin을 Exclusive하게 획득
pin이라는 용어의 의미는 LCO에 핀을 꽂는다는 것 -> 그 동안 LCO의 값이 변동되지 않도록 보장받는 역할

library cache lock은 핸들에 대해 획득, library cache pin은 LCO에 대해 획득

1. Syntadx 체크 (문법체크)
2. Semantic 체크 (객체체크)
3. 권한체크
4. Shared Pool의 동일 SQL 검색 (해시 값을 이용해 Library Cache 영역 검색 + SQL Text 비교 + 동일 오브젝트 비교 등)
여기까지가 소프트파싱
5. Parse Tree 생성
6. Execution Plan 생성
5~6번까지 수행하는 것을 하드파싱

library cache 객체에 대해 library cache lock을 획득하는 것은 소프트파싱까지의 단계
5,6번 하드파싱 단계 동안에는 library cache lock을 Null모드로 변환하고 library cache pin을 Exclusive하게 획득 -> 실행 계획을 세우는 동안 LCO에 대해 변동이 발생하는 것을 막기 위함
하드 파싱이 끝나면 -> library cache pin을 Shared로 바꾸고 Execute단계로

library cache lock과 library cache pin은 Enqueue락은 아니지만 Enqueue락과 유사한 메커니즘을 사용한다.

1. 두개의 세션에서 계속해서 같은 CREATE OR REPLACE PROCEDURE XXX를 호출하면 어떤 이벤트를 대기하는가? -> 프로시저 등의 오브젝트를 새로 생성할때는 해당 library cache 객체에 대해 library cache lock을 exclusive하게 획득해야 한다.
2. 두개의 세션에서 계속해서 같은 프로시저를 컴파일한다면 어떤 이벤트를 대기하는가? -> 프로시저를 컴파일 하는 경우에도 library cache lock을 exclusive하게 획득함
3. 한 세션에서 프로시저를 수행하는 동안 다른 세션에서 해당 프로시저를 컴파일한다면 어떤 이벤트를 대기하는가?
수행하는 쪽 : library cache lock=Shared로 파싱, 파싱 끝나면 Null +library cache pin = shared로 획득
컴파일 쪽 : library cache lock=exclusive + library cache pin=exclusive
-> library cache lock의 경우 수행쪽에서 파싱만 끝나면 바로 Null로 바꾸기 때문에 대기 시간이 길지 않지만, library cache pin의 경우 수행 내내 shared로 보유하고 있기에 대기시간 길게된다. || 좌우간 둘다 대기
4. 하드 파싱에 아주 많은 시간이 걸리는 SQL 문을 두개의 세션에서 동시에 수행한다면 어떤 이벤트를 대기하는가?
먼저 파싱한 쪽 : SQL커서에 대해 library cache pin을 Exclusive하게 획득 -> 나중 파싱한 쪽 : SQL커서에 대해 Shared하게 library cache pin락을 획득하기 위해 대기한다.
-> 하드파싱이 심할땨 주로는 library cache래치-shared pool래치 경합이 심한데 -> 가끔 library cache pin 대기가 발생한다는 것은, SQL 커서및 관련 객체에 library cache pin을 exclusive하게 획득하기때문
5. 두개의 세션에서 계속해서 같은 테이블의 정의를 변경한다면 어떤 이벤트를 대기하는가?
-> 테이블의 속성을 변경하려면, 테이블에 해당하는 library cache 객체에 library cache lock을 exclusive하게 획득 -> library cache lock 경합
6. 한 세션에서 alter table ...을 이용해 테이블 정의를 변경하는 동안 다른 세션에서 같은 테이블에 대해 select를 수행하면 어떤 이벤트를 대기하는가?
-> library cache lock을, 변경은 exclusive, 조회는 shared -> 둘사이에 호환성은 없으니 대기현상 발생.
7. 한 세션에서 select ... from xxx 를 수행하는 동안 다른 세션에서 alter system flush shared_pool을 수행하면 어떤 이벤트를 대기하는가?
-> 둘 모두 library cache pin 대기 이벤트 -> flush하는 쪽은 딕셔너리 이용하려고 대기 - select쪽은 사라진 딕셔너리 정보들이 다시 메모리에 올라올때까지 대기

정리
1. library cache lock과 library cache pin 대기에 의한 성능 저하 현상은 대부분 부적절한 DDL(create, alter, compile, flush, ...)에 의해 발생한다. 따라서 트랜잭션이 왕성한 시스템에 대해서 DDL을 수행할 때는 위의 내용을 충분히 고려한 후 수행하도록 해야 한다. 간혹 하드파싱이 왕성한 시스템에서 Shared Pool 메모리 고갈을 피하기 위해 (즉 ORA-4031에러를 피하기 위해) flush를 수행하는 경우가 있다. 하지만 이 경우 library cache pin 대기에 의한 성능 저하 현상이 생길 수 있다. 또한, flush 이후의 SQL 문 수행은 모두 하드파싱을 수반하게 되므로 이에 의한 성능 문제가 야기될 수 있다.
2. 하드파싱이 왕성한 시스템에서도 library cache pin 대기가 나타날 수 있다. library cache pin 뿐만 아니라 library cache와 관련된 대기의 대부분이 과도한 하드파싱에 의해 유발된다. 최선의 해결책은 바인드 변수를 적절히 사용하는 것이며, Cursor sharing과 같은 기법을 검토할 수도 있다. 단, Cursor sharing은 충분한 테스트를 통해서만 실제 시스템에 적용할 것을 권장한다.