**Library Cache에서의 대기이벤트들**

**- latch: shared pool**

shared pool 래치는 Shared Pool의 기본 메모리 구조인 힘을 보호하는 역할을 한다. 프리 청크를 찾기 위 해 프리리스트를 탐색하고, 적절한 청크를 할당하고, 필요한 경우 프리 청크를 분할하는 일련의 작업들은 모두 shared pool 래치를 획득한 후에만 가능하다. 주로 발생하지는 않으나, 동시에 여러 세션이 청크를 할당 받아야하는 경우, 혹은 하드파싱이 심한 경우 청크를 쪼개는 현상이 자주 발생하고 이로 인해 프리리스트에 수많은 작은 크기의 프리 청크들이 붙는 현상이 발생한다. 이 현상을 단편화라고 부르며, 단편화로 인해 프리리스트를 탐색하는 시간이 길어지고, 그만큼 shared pool 래치를 보유하는 시간이 늘어난다. 이로 인하여 shared pool 래치 경합이 발생한다.

하드파싱이 발생하면 새로운 SQL 정보를 담을 청크를 할당 받기 위해 프리리스트를 탐색해야 한다. 이러한 일련의 행위 동안 오직 하나의 프로세스만이 shared pool 래치를 독점해야 하기 때문에 동시에 많은 세션이 하드파싱을 수행하는 경우 latch: shared pool 대기가 많이 발생하게 된다. 따라서 하드파싱이 과도하게 발생하는 시스템에서 shared pool 래치 대기를 줄이기 위해 Shared Pool의 크기를 늘려주는 것은 대단히 위험한 발상이라고 할 수 있다. Shared Pool의 크기가 커지면 그만큼 프리리스트의 개수와 프리리스트에서 관리해야 할 프리 청크의 개수도 증가한다.

따라서 프리리스트를 탐색하기 위해 shared pool 래치를 잡고 있어야 하는 시간이 늘어난다. 그와비례해서 latch: shared pool 대기시간도 증가하게 된다. 해결책으로는 오라클은 shared pool을 여러 개의 서브풀로 최대 7개까지 나누어서 관리할 수 있다. \_KGHDSIDX\_COUNT 히든 파리미터를 이용하면 서브풀의 개수를 관리할 수 있다. 다른 방안으로는 하드파싱에 의해 shared pool 래치 경합이 발생하는 경우, shared pool의 크기를 줄이는 것이다. 하지만 이 경우엔 ORA-4031에러가 발생할 확률이 높아지며, 부가적인 하드파싱이 유발될 수도 있다. 마지막으로는 Cursor Sharing 기법을 사용하는 것이다. 즉, 상수를 사용한 SQL 문장을 자동으로 바인드 변수를 사용하게끔 치환해서 커서가 공유되도록 해주는 기능이다.

추가적으로, 새로운 SQL 수행으로 하드파싱이 발생하는 경우 오라클은 먼저 library cache 래치를 획득한 상태에서 library cache 영역을 탐색하고 이 상태에서 shared pool 래치를 획득하게 된다. 따라서 library cache 래치 획득 단계에서 경합이 발생해서 병목현상이 생기는 경우에는 상대적으로 latch: shared pool 대기가 적게 나타날 수 있다.

**- latch: library cache**

library cache 래치는 SQL을 수행하기 위해 Library Cache 메모리 영역을 탐색하고 관리하는 모든 작업을 보호한다. 하드파싱이나 소프트파싱이 과다한 경우, 버전 카운트가 높은 경우 library cache 래치 경합이 발생한다.

하드파싱이 많이 발생하면 library cache를 탐색하는 횟수가 늘어나므로, 그만큼 library cache 래치를 보유하는 시간과 횟수가 늘어나고, Library Cache 영역에 대한 탐색뿐만 아니라 추가적인 청크 할당이 필요하기 때문에 그만큼 library cache 래치를 보유하는 시간이 늘어난다.

소프트파싱의 경우는 하드파싱에 비해 그 비용이 매우 낮지만, 필수적인 신텍스(syntax), 시멘틱(semantic) 체크와 library cache 탐색 등의 과정은 피할 수 없다. 이에 따라 많은 세션이 동시에 소프트파싱을 수행하는 경우, library cache 래치 경합에 의한 성능 저하 현상이 발생하게 된다. 이를 방지하기 위해선, 파싱 횟수 자체를 줄이거나, Static SQL를 쓰는 것, SESSION\_CACHED\_CURSORS 파라미터(세션 커서 캐싱)를 쓰는 것이다. 버전 카운트가 많은 경우는, 자식 LCO가 많다는 것이다. 자식이 많아지므로, 탐색 시간이 길어지고, 이로 인해 library cache를 탐색하는 시간이 그만큼 길어진다는 것이다.

**- SGA 영역의 페이지 아웃이 발생하는 경우**

Shared Pool 이 디스크로 페이지 아웃된 경우, 해당 영역에 대한 스캔이 발생할 때 다시 디스크 내용을 메모리로 불러들이는 과정(페이지인)동안 대기해야 하므로 library cache 래치에 대한 대 기시간이 증가할 수 있다 만일 latch: library cache 대기가 높은 시점에，OS에서 스왑(Swap) 현상이 발생한다면 페이지 아웃에 의한 성능 저하일 확률이 높다.

**- library cache lock과 library cache pin**

library cache lock은 Library Cache 객체에 접근하거나 변경할 때 library cache 핸들에 대해 획득하는 락이다. library cache pin은 Library Cache 객체에 대한 수행이나 변경 시 Library cache object(LCO)에 대해 획득하는 락이다. Library cache lock이 LCO의 명세(specification)를 보호한다면 library cache pin은 LCO의 내용(실행정보)을 보호한다. 따라서 SQL 커서와 같이 하나의 논리적 객체에 대해 여러 개의 LCO가 생성되는 경우에 library cache 영역에 대한 접근성을 최대한으로 높일 수 있다.

library cache lock과 library cache pin에 대해 또 한가지 유념할 것은, 이 두개의 락이 비록 Enqueue 락으로는 분류되지 않지만 내부적으로는 Enqueue 락과 유사한 메커니즘을 사용한다는 것이다. library cache lock과 library cache pin에 관련된 성능 이슈를 정리해보자면 아래와 같다.

1) Library cache lock과 Library cache pin 대기에 의한 성능저하현상은 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. 하드파싱이 왕성한 시스템에서 Shared Pool 메모리 고갈을 피하기 위해(ORA-4031 에러) flush를 수행하는 경우가 있다. 하지만 이 경우 library cache pin 대기에 의한 성능 저하 현상이 생길 수 있다. 또한 flush 이후의 SQL문 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.