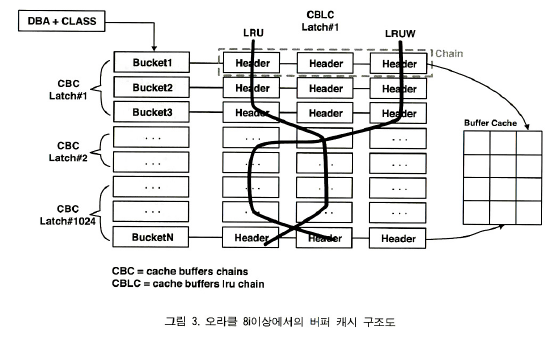
**오라클 내부구조와 OWI**

**버퍼캐시와 OWI**

- 버퍼캐시 구조

최근에 사용된 블록에 대한 정보를 메모리의 일정 부분에 저장해 놓은 것을 버퍼 캐시라고 한다. Database Buffers에 해당하는 값이 현재 인스턴스의 버퍼 캐시의 크기이다. 오라클은 버퍼 캐시를 효과적으로 관리하기 위해 해시 체인 구조를 사용한다. 버킷 -> 체인 -> 헤더의 구조를 사용한다. 해시 체인 구조는 cache buffers chains 래치를 이용해 보호된다.



- Working Set

오라클은 버퍼 캐시를 효율적으로 사용하기 위해 두 종류의 LRU 리스트를 사용한다. LRU 리스트는 프리 버퍼, 사용 중이거나 사용된 버퍼, 더티 버퍼 등을 포함한다. LRUW 리스트는 아직 디스크에 기록되지 않은 변경된 버퍼들의 리스트를 관리한다.

이 리스트들은 다시 메인 리스트와 보조 리스트로 나누어 관리한다. LRU 리스트는 사용된 버퍼들의 리스트인 메인 리스트, 프리 버퍼들의 리스트인 보조 리스트가 있다. LRUW리스트는 변경된 버퍼들의 리스트인 메인 리스트, 현재 DBWR에 의해 기록된 버퍼들의 리스트인 보조 리스트가 있다. LRU와 LRUW 리스트는 항상 짝으로 존재하며, 이 짝을 Working Set이라고 부른다.

하나의 Working Set을 하나의 cache buffers lru chain 래치가 관리한다. 따라서 동시에 많은 프로세스가 LRU 리스트나 LRUW리스트를 탐색하고자 할 경우에 cache buffers lru chain 이벤트를 대기한다.

오라클에는 다양한 종류의 버퍼 풀이 존재하며 각 버퍼 풀들이 이들 래치를 골고루 사용한다. 첫째, 버퍼는 크게 Default 버퍼 풀, Keep 버퍼 풀, Recycle 버퍼 풀로 나뉘어 진다 둘째， Default 버퍼 풀은 다시 블록 크기 별로 표준 블록 사이즈, 2K, 4K, 8K, 16K, 32K 버퍼 풀로 나뉘어 진다. 각각의 버퍼 풀은 각각 독립적인 cache buffers lru chain 래치를 사용한다. 따라서 래치의 최소 개수는 8개가 된다.

서버 프로세스가 스캔하는 모든 버퍼들이 LRU 리스트에 등록되기 때문에 LRU 리스트를 효율적으로 관리하는 것이 매우 중요하다. 특히 불필요하게 많은 량의 블록을 스캔하는 프로세스에 의해 중요한 버퍼들이 버퍼 캐시에서 밀려나는 것을 최소화할 수 있어야 한다. 오라클은 8i 이후의 버전부터 LRU 리스트를 효율적으로 관리하기 위해 Touch count에 기반한 LRU 알고리즘을 사용한다.

Touch count 기반의 LRU 알고리즘은 다음과 같은 방식으로 작동한다.

1) LRU 리스트의 메인 리스트는 크게 핫 영역 과 콜드 영역으로 나누어진다 자주 사용되는 블록은 핫 영역에 머무르며, 사용빈도가 낮은 블록은 콜드 영역에 머무른다 오라클은 개별 버퍼마다 Touch count를 관리하며 프로세스에 의해 스캔이 이루어질 때마다 Touch count를 1씩 증가시킨다.

2) 프리 버퍼를 찾을 때는 우선 LRU 리스트의 보조 리스트에서 미 사용된 버퍼를 찾는다. 만일 보조 리스트가 비어 있다면， 메인 리스트의 콜드 영역의 꼬리에서부터 프리 버퍼를 찾는다. 메인 리스트의 꼬리에 있으면서 Touch count가 1 이하인 버퍼가 프리 버퍼로 사용된다 프리 버퍼를 찾는 과정에서 Touch count가 2 이상인 블록을 만나면 핫 영역의 머리로 옮기고 해당 버퍼의 Touch count를 0으로 초기화 시킨다.

3) 싱글 블록 I/O에 의해 읽힌 블록은 Mid-point에 삽입되며 Touch count는 1의 값을 지닌다. Mid-point가 가리키는 위치는 콜드 영역의 머리이다. 싱글 블록 I/O에 읽힌 블록은 콜드 영역의 머리에 위치함으로써 버퍼 캐시에 머무를 확률이 높아진다.

4) 멀티 블록 I/O에 의해 읽힌 블록들은 Mid-point에 삽입된 후 콜드 영역의 제일 뒤로 옮겨진다. 풀 테이블 스캔이나 인덱스 풀스캔으로 읽힌 블록들은 콜드 영역의 꼬리에 위치함으로써 버퍼 캐시에 머무를 확률이 낮아진다.

5) Keep 버퍼 풀과 Recycle 버퍼 풀은 Default 풀과는 달리 영역의 구분이 불필요하므로 핫 영역을 가지지 않는다. Recycle 버퍼 풀은 핫 영역을 가지지 않는다는 점을 제외하면 Default 버퍼 풀과 완전히 통일한 방식으로 작동한다. 하지만 Keep 버퍼 풀의 경우에는 FTS로 읽히는 작은 크기의 테이블을 메모리에 상주시키기 위해 고안된 공간이기 때문에 멀티 블록 I/O로 읽은 블록들을 싱글 블록 I/O로 읽은 블록과 동일하게 콜드 영역의 제일 앞에 위치시키도록 구현되었다.

- Buffer Lock

버퍼 락은 버퍼 자체를 보호하는 역학을 한다. 버퍼의 내용을 변경하거나 읽으려는 프로세스는 변경 작업이나 읽기 작업을 완료할 때까지 해당 버퍼에 대해서 버퍼 락을 Exclusive 혹은 Shared 모드로 획득해야 한다. 동시에 두 프로세스가 버퍼의 내용을 변경하는 것을 막기 위해서다.

- 버퍼 탐색

1) 사용자가 요청한 블록에 대해 해시 함수를 이용해서 해시 값을 생성하고 해시 값에 해당하는 버킷을 찾는다.

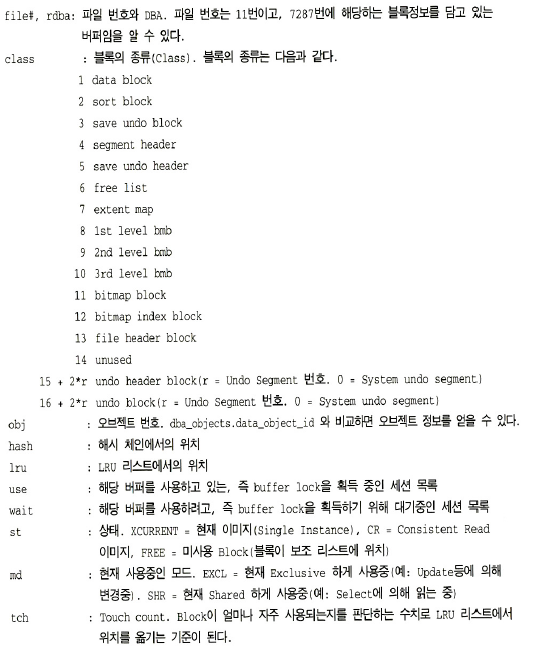
2) 해시 버킷을 보호하는 cache buffers chain 래치를 획득한다. 해시 버킷에 딸려 있는 체인을 탐색해서 블록에 해당하는 버퍼 헤더가 존재하는지 확인한다. 있다면 그것을 이용하고 이 과정을 Logical reads라고 부른다.

3) 버퍼 캐시에 블록이 존재하지 않으면 working set을 관리하는 cache buffers lru chain 래치를 획득한다. 래치를 획득한 후 LRU 리스트의 보조 리스트에서 프리 버퍼를 찾는다. 프리 버퍼를 찾게 되면 해당 버퍼에 대해 버퍼 락을 Exclusive하게 획득하고 데이터 파일로부터 블록을 해당 버퍼로 읽어 들인다. 이 일련의 과정을 Physical Reads라고 부른다.

4) LRU 리스트에서 프리 버퍼를 찾을 때 \_DB\_BLOCK\_SCAN\_MAX\_PCT 파라미터의 값만큼 스캔을 하고도 프리 버퍼를 찾지 못하면 서버 프로세스는 LRU 리스트의 스캔을 멈춘다. 서버 프로세스는 DBWR에 더티 버퍼를 파일에 기록하고 프리 버퍼를 확보할 것을 요청한다. 디스크에 기록할 버퍼를 찾게 되면 buffer lock을 획득한 후 버퍼를 디스크에 기록한다. 디스크에 기록된 버퍼는 프리 버퍼로 변경되고 LRU 리스트로 옮겨진다.

- 버퍼 캐시 덤프

1) alter session set events 'immediate trace name buffers level 1'; / oradebug dump buffers 1



3) X$BH 뷰

