Cache Buffer Chain

par

Date de publication : 28/07/2005

Dernière mise à jour : 28/07/2005

Introduction
Latch free
Cache buffer chain (CBC) issue du wait event ' Latch Free"
A - v$session_wait
B - v$latch_children
Dictionnaire

Introduction

Avant tout je tiens à préciser que cette note technique n'a pas pour but de vouloir être une référence technique concernant la résolution d'une anomalie de type :

<< LATCH FREE et CACHE BUFFER CHAIN>> .

Elle a un but plus modeste qui consiste à apporter les outils nécessaires pour effectuer une analyse descendante : Drill and Down . J'espère que les requêtes fournies vous permettront de faire ressortir deux informations primordiales pour la résolution de votre anomalie :

- Analyser le type de latch free
- Déterminer le segment concerné

Environnement Oracle : Oracle 8i

Latch free

Les latchs sont des mécanismes bas niveau protégeant les blocs de données en SGA .Bien que de bas niveau , les latchs à contrario des Verrous , ne possèdent que deux états possibles : libre ou occupé .Un verrou quant à lui possède plusieurs niveaux . Ainsi un process ayant obtenu le latch pour la modification d'une zone mémoire a l'assurance qu'aucun autre process ne peut modifier la même zone mémoire pendant que celui ci << travaille >> .

Un latch peut être demandé de deux manières :

- *Willing to wait* : Le latch est demandé jusqu'a obtention .Si à sa demande , il n'est pas libre , le process attend (spining) un laps de temps puis le redemande à intervalle régulier .A chaque échec d'obtention un compteur est incrémenté (P 3 de v$session_wait) .
- *No wait* : Le latch est demandé , s'il est indisponible la transaction est abandonnée .

Cache buffer chain (CBC) issue du wait event ' Latch Free"

Les blocs sont placés dans le buffer cache dans une liste chaînée (cache buffer Chain) .Chaque chaîne est protégée par un seul verrou (ou latch) . Un processus doit avoir ce verrou s'il veut consulter les données .

Ce type d'évènement : CBC n'est pas présent en tant que tel dans les vues d'observation V$session_wait , v$session_event ou même v$system_event .

- V$session_wait : Evènement d'attente à un instant T (exécution de la requête) pour une session (ou SID, SERIAL#)
- v$session_event : Evènements d'attentes cumulés depuis le début de la session (SID, SERIAL#) .
- V$system_event : Evènements d'attentes cumulés depuis le démarrage de la base et pour toutes les sessions :

A - v$session_wait

```
SQL> SELECT sid ,event , p1text , p1 p1raw , p2 , p3  FROM v$session_wait w
```

```
  2  WHERE sid IN (17) ;

       SID EVENT                P1TEXT          P1RAW                     P2         P3
---------- -------------------- ----------        ----------                    ----------
----------
        34 latch free  file#      address       C0000001053EC720          66      0
```

Lorsqu'on observe dans la vue v$session_wait un évènement de type latch free on peut collecter un certain nombre d'informations très importantes :

Tout d'abord dans la colonne P2 donne le numéro de latch , le nom du latch peut être retrouvé ainsi :

```
SQL> SELECT * FROM  v$latchname
  2  WHERE latch#=66 ;

    LATCH# NAME
---------- ----------------------------------------------------------------
        66 cache buffers chains
```

Ensuite la colonne P1raw va nous permettre d'isoler la ressource qui est sujette à ce bloc chaud (Hot bloc):

Qu'est ce que le P1RAW : c'est une valeur hexadécimale spécifiant l'adresse du latch .Nous allons nous servir de cette valeur pour requêter sur la vue sys.x$bh

Pour information P3 est un compteur qui va comptabiliser le nombre de fois ou le latch a été demandé et n'a pas été obtenu .

En effet sous SYS :

```
SQL> SELECT FILE# , dbablk, class, state, tch FROM x$bh WHERE hladdr='C0000001053EC720' ORDER BY tch
 ;

     FILE#     DBABLK      CLASS      STATE        TCH
---------- ---------- ---------- ---------- ----------

(###.
voir la dernière ligne )

        40      41359          1          1         94
```

Et grâce à la valeur de file# et de DBABLK (block) nous allons pouvoir retrouver le segment (table ou index) qui est sujet à contention .Plus précisément la colonne FILE# est le numéro de fichier et le DBABLK est le numéro de bloc.

toujours sous le compte sys ou internal (version > 9i) :

```
SQL> SELECT * FROM dba_extents WHERE 41359 BETWEEN dba_extents.block_id AND dba_extents.block_id+blo
cks-1   AND dba_extents.file_id = 40 ;

OWNER
------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME                 SEGMENT_TYPE       TABLESPACE_NAME
------------------------------ ------------------ ------------------------------
 EXTENT_ID    FILE_ID   BLOCK_ID      BYTES     BLOCKS RELATIVE_FNO
---------- ---------- ---------- ---------- ---------- ------------
PSHR83
PS_GP_RTO_PRC_WRK
                               INDEX              PSINDEX
```

```
           0         40      41357      40960          5          40
```

B - v$latch_children

Nous venons de nous intéresser à une vue micro (sur une requête à un instant T) , maintenant nous allons nous intéresser à une vue macro , à savoir tout les évènements d'attente depuis le démarrage de la base , pour ce faire la vue latch_children et la requête suivante vont nous aider .

Cette démarche est plus dans une optique d'optimisation de l'instance que d'une requête ou d'un traitement .

```
SQL> SELECT addr, latch#, gets, misses, sleeps, ROUND (((misses*100)/gets),6) AS ratio
  2      FROM v$latch_children
  3     WHERE sleeps>0
  4       AND latch# = 66
  5     ORDER BY sleeps DESC ;

ADDR                 LATCH#       GETS     MISSES     SLEEPS      RATIO
---------------- ---------- ---------- ---------- ---------- ----------
C0000001053EC720         66   70591975    1976853       2877   2,800393
```

La colonne RATIO est une colonne issue d'un calcul , déterminant le ratio qui existe entre misses et gets , :

Détaillons cette requête ,chaque fois qu'un verrou est demandé sur un block de données , si oracle nous l'accorde de suite la colonne GET est incrémenté , sinon MISSES est incrémentée .

Donc le ratio revient à poser la question suivante :

Combien de fois un latch a été demandé et combien de fois il a été octroyé de suite ?

Oracle recommande d'avoir un ratio inférieur à 1% , ici nous voyons que le ratio avoisine les 3 % ...

Cependant comme tout ratio , cela doit être pondéré par le contexte de l'instance .

Avec la colonne ADDR et en reprenant le même principe que vu précédemment , on peut retrouver le segment concerné

A part les conseils classique de tuning concernant le code , je vous déconseille fortement de " tuner " les paramètres non documentés et cachés d'oracle :_db_block_hash_buckets et

_db_block_hash_latches sans le support Oracle .

Pour une table , on conseille en règle générale le partitioning et pour un index , « le reverse index » peut obtenir de bons résultats mais attention , si cet index est utilisé ailleurs en range scan , l'optimiseur COST peut dans ce cas choisir le FTS (Full Table Scan) .D'où la nécessité de construire des Histogrammes pour les colonnes indexées

Lorsqu'on met en place un reverse index , deux méthodes permettent de diminuer les probabilités que l'optimiseur COST ou CHOOSE préfère un FTS au lieu d'un accès indexé :

1-Construire des histogrammes : La commande suivante permet de le faire :

```
analyze table schéma.table compute statistics for all indexed columns
```

Cependant pour les versions 8i ou supérieures préférez le paquetage : DBMS_STATS

```
exec DBMS_STATS.GATHER_TABLE_STATS(OWNNAME => NULL, TABNAME => 'DVP', CASCADE => FALSE, METHOD_OPT
=> 'FOR ALL INDEXED COLUMNS');
```

Suite à « la mort » du process , c'est PMON (process monitor) qui libère le latch .Notamment lors d'un « SNAPSHOT TOO OLD ».

Cependant , la construction des histogrammes ne renseignent que sur la distributivité des données , donc a un effet limité .

Remarque : attention le latch free n'est pas en corrélation avec un cache hit ratio faible .Il ne sous-entend nullement une augmentation du cache size .

Dictionnaire

Wait event : Evènement d'attente que l'on peut observer dans les vues systèmes

Latch : Verrou

Range Scan : Lecture par intervalle d'un index , cela se produit par exemple lorsque nous avons dans une clause where un between

Reverse index : Index inversé , à savoir Oracle distribue les différentes valeurs d'un index afin d'avoir un Hot block , par exemple si nous avons deux valeurs : 1009 et 1010 qui sont fortement sollicitées , il y a un fort risque d'avoir un hot block , cette probabilité est réduite lorsque les valeurs sont inversées : 1010 devient 0101 et 1009 devient donc 9001 .

Hot block : Bloc de données fortement accédé

FTS ou Full table SCAN : Oracle va parcourir toute la table sans passer par un index .En règle générale l'événement d'attente associé est le DB_FILE_SCATTERED_READ . A contrario, une lecture d'index se traduit pas un événement DB_FILE_SEQUENTIAL_READ.

Partitioning : Découper un Index ou une table sur plusieurs axes .

Cost : L'optimiseur va calculer et choisir son chemin , en fonction de coût recueillis grâce notamment aux statistiques .

CHOOSE : Choix entre COST et RULE.

Rule : le mode RULE indique à Oracle de choisir le plan d'exécution en fonction de règles pré-établies par Oracle (ordre des tables dans la clause FROM par exemple). Ce mode est également utilisé lorsque aucune statistique n'est calculée sur les objets de la requête analysée par Oracle.

Tuning : Littéralement réglage mais ici nous parlerons plus dans le sens d'optimisation (bien qu'un bon réglage apporte également une optimisation) .

X$tables : Ces tables existent directement dans la SGA , donc on consulte la SGA lorsque l'on ces interrogent ces tables .En effet ces tables existent uniquement lorsque l'instance est démarré .Pour voir ces tables il est préférable d'être connecté en SYS .La x$BH est une vue sur les cache buffer chain accédés .