Oracle 10g HWM 原理及性能优化 *

蔡 焰
(广东韶关学院 图书馆，广东 韶关 512005)

摘 要：HWM(High Water Mark)是表中已经使用过的存储空间与未使用过的存储空间之间的分界线，HWM 对全表扫描的性能有非常大的影响。当全表扫描时，Oracle 会读取 HWM 下所有的块，即使这些块中有很多是空块，空块的存在，也即是表中碎片的存在，必将增加全表扫描额外的物理 I/O 开销及 CPU 开销，严重降低访问 Oracle 数据表的性能。通过对 Oracle 中关于表中 HWM 的原理及性能优化问题的讨论，针对 HWM 下的碎片问题提出相关的优化策略，并对其空间重组前后进行性能对比测试。

关键词：Oracle；HWM；性能优化
中图分类号：TP311.138 **文献标识码**：A **文章编号**：1674-7720(2013)08-0001-03

Oracle 10g HWM performance tunings

Cai Yan
(China Library of Shaoguan University, Shaoguan 512005, China)

Abstract: The high water mark (HWM) is a line separate the used blocks and free blocks in a database segment. The HWM is important because of the effect it has when a full-table scan is performed. When performing a full-table scan, Oracle will read all the blocks below the HWM, even if they are empty blocks. Having many unused blocks below the HWM (thus fragmentations) degrades performance for full-table scans with additional, yet unnecessary I/O and CPU. It also fills the buffer cache with empty blocks. This paper discussed the concept of Oracle HWM and HWM performance tunings. It focused on the data fragmentation issues when using HWM and provided perforumance optimization solutions. It also compared the performance differences before and after the fragmentation has been resolved.

Key words: Oracle; HWM; performance tuning

Web2.0 与 Web3.0 的发展都离不开后台支持数据库，数据库运行的好坏、快慢，直接影响到使用者的应用，因而本文将重点研究信息资源建设中后台数据库的优化策略。Oracle 数据库是具有高可靠性、高安全性、高兼容性的大型关系型数据库，是信息化建设的重要基础平台。网络中的信息资源数据库具有异构、数据量大、多媒体内容多、查询频繁等特点，伴随网络不断深入的应用，其存储在数据库中的数据量越来越多，而传统的数据库设计方法使得数据库随着访问数据量的增大其性能明显地降低[1]。Oracle 的逻辑空间管理是 Oracle 管理和优化的重要部分，ASSM 段空间自动管理下的 HWM 问题对 Oracle 的存储管理和性能优化有重大影响。本文在探讨 Oracle 10g 逻辑存储管理的基础上，针对 HWM 下的碎片问题提出了相关的优化策略，并对其空间重组前后进行了性能测试。

* 基金项目：广东图书馆学科研课题(GDTK1136)

1 Oracle 存储管理

1.1 Oracle 逻辑存储管理

Oracle 在逻辑存储上分 4 个粒度，如图 1 所示。

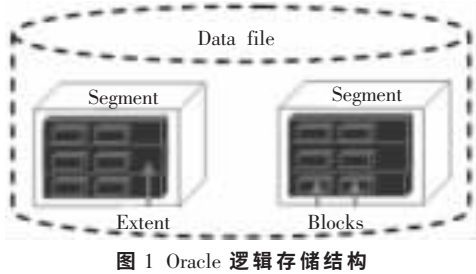

图 1 Oracle 逻辑存储结构

(1) Block(块)：粒度最小的存储单位，标准默认大小是 8 KB，Oracle 每一次 I/O 操作都是按 Block 来进行的。

(2) Extent(区)：由一系列相邻的 Block 组成，是 Oracle 空间分配的基本单位[2]，Oracle 是以 Extent 为单位进行扩展的。

(3) Segment(段):由一系列的 Extents 所组成[2],当创建一个对象时(表或索引),就会分配一个 Segment 给这个对象。

(4) Tablespace (表空间): 包括 Segment、Extent 和 Block,Tablespace 的数据物理上存储在其所在的数据文件中,一个数据库最少要有一个 Tablespace。

1.2 HWM

高水标记 HWM(High-Water Mark)这个概念在 Segment 的存储内容中是比较重要的。简单来说,HWM 代表一个表使用的最大(top limit)块(如图 2 所示),就是一个 Segment 中已使用和未来使用的 Block 的分界线[3]。图 2 显示了 HWM 首先位于新创建表的第一个块中,随着数据的插入和更新,使用了越来越多的块,当现有空间不足而进行空间扩展时 HWM 会随之向上移。如果删除一部分行数据,可能会有许多块不再包含数据,但 HWM 不会往下移,被占用的最高空间称为 HWM。

图 2 HWM 示意图

Oracle 在做全表扫描时会读取 HWM 下的所有 Blocks,即使其中不包含任何数据,Oracle 都会一一读取,这会大大影响系统的性能,特别是当 HWM 之下的大多数块都为空时。

如果一个 OLTP 系统(即联机事务处理,就是常说的关系数据库,对记录进行增、删、改、查)应用频繁地对某个表里的记录进行 DML(Data Manipulation Language)操作(即数据操纵语言,一种命令使用户能够查询数据库以及操作已有数据库中的数据的计算机语言),会造成 Block 中数据分布稀疏,导致 HWM 下存在大量的碎片,浪费大量的空间。当做全表扫描时,Oracle 会读取 HWM 之下的所有块,即使其中不包含数据[3]。对于 HWM 以下表的碎片,做全表访问时必然增加一致性读,因而影响到响应时间,降低系统性能。

2 优化策略

对于增、删、改操作比较频繁的表,尤其是删除操作比较频繁的表,一般表的高水位 HWM 值会偏高,也就是表中数据块碎片高,虽然 ASSM 的自动空间管理能提高 DML 操作并发访问的性能,但是 HWM 下高碎片的产生会大大影响访问效率,而减少碎片、降低对象的 HWM 可提高对象的访问效率,从而达到性能优化,大大提高数据的访问效率。表对象可以通过 shrink 或 move 方法实现重组、减少碎片、降低 HWM,进行性能优化;索引对象可以提供 rebuild 的方法来实现重组、减少碎片、降低 HWM,进行性能优化。当然,在对表及索引进行 shrink 或 move 及 rebuild 操作时,最好选择在非业务高峰时进行,避免影响业务的正常运转。

shrink 与 move 操作有一些不同,但都可以完成表中碎片的整理,在此可做一些比较:

(1) move 的执行效率比 shrink 高,因为 shrink 会产生 redo log、undo log;

(2) shrink 对数据的移动是从后往前的,所以 shrink 不需要使用额外的空闲空间,而 move 是需要额外空闲空间的;

(3) 对带有索引的表进行 shrink 操作时,索引是不需要重建的;而对带有索引的表进行 move 操作时,索引是需要 rebuild 重建的,否则索引不可用;

(4) 对表进行 shrink 操作时,必须打开表的行迁移属性。

shrink 和 move 都会对操作的表加表级独占锁,因此其他 session 对此表执行 DML 操作时,存在锁等待;当 shrink 或 move 操作执行完成,锁释放。

索引的 rebuild 是可以在线完成的,比较适合在高可用环境下完成。

另外,shrink 是 10g 的新特性,仅对 ASSM 管理表空间有效。

具体命令操作如下:

shrink 命令:

Alter table XXX enable row movement(打开表 XXX 的行迁移属性);

Alter table XXX shrink space(仅仅对表 XXX 进行缩小,不对表中的索引进行空间缩小);

Alter table XXX shrink space cascade(缩表的同时,也对表中的索引进行缩小);

Alter table XXX disable row movement(缩表完成后,关闭表 XXX 的行迁移属性)。

move 命令:

Alter table XXX move(对表进行 move);

Alter index YYY rebuild(如果表有索引,则需要对表的索引进行重建)。

3 性能优化测试

对于碎片较多的表,可以通过 shrink 或 move 操作降低表中 HWM 高水位的值来进行性能优化。下面以 shrink 命令为例子进行测试。

3.1 对 TEST 表进行分析

(1) 表大小

SQL> select segment_name, bytes/1024/1024 表大小 MB from dba_segments where segment_name='TEST';

SEGMENT_NAME 表大小 MB

TEST 5.632

(2) 表的实际数据大小:2.439 MB

SQL >select table_name, AVG_ROW_LEN ,NUM_ROWS, AVG_ROW_LEN*NUM_ROWS/1024/1024 表实际大小 MB, LAST_ANALYZED from dba_tables where table_name='TEST';

TABLE_NAME AVG_ROW_LEN NUM_ROWS 表实际大小 MB LAST_ANALYZED

TEST 157 16290 2.439 2012-12-01 00:13:12

(3) 根据表实际大小公式可得出该表的碎片率为:56.7%

(1-表实际数据大小/表大小)×100/%=(1-2.439/5.632)×100%=56.7%碎片率:56.7%

(4)执行 SQL 语句:select * from test;

查询表 test 中所有记录,查询所需时间为 2 s。

SQL 的解释计划如下:

Execution Plan

|Id|Operation |Name |Rows | Cost (%CPU)| Time|

|0| SELECT STATEMENT | | | 141 (0)| |

|1| TABLE ACCESS FULL|TEST|16290|141(0)|00:00:02|

从以上的 SQL 解释计划来看,SQL 采用的是全表扫描读的方式访问,SQL 将读取表的高水位 HWM 以下的所有数据块。

由上可知:(1)表 TEST 的大小为 5.632 MB,但实际数据大小约为 2.439 MB,碎片率约为 56.7%,表 TEST 中存在大量的碎片;(2)查询该表所有记录所需要的时间为 2 s。

3.2 碎片重组 优化处理

通过 shrink 方式对表 TEST 作碎片重组实现对表的优化处理。

SQL> select segment_name, bytes/1024/1024 表大小 MB from dba_segments where segment_name='TEST';

segment_name 表大小 MB

TEST 5.632

SQL>alter table test enable row movement;

(上接第 3 页)

SQL>alter table test shrink space;

SQL>alter table test disable row movement;

SQL >select segment_name,bytes/1024/1024 表大小 MB from dba_segments where segment_name='TEST';

SEGMENT_NAME 表大小 MB

TEST 3.072

通过对上对 TEST 表进行优化处理后可以看到:(1) shrink 缩表操作后 TEST 表的大小从 5.632 MB 缩小到 3.072 MB,缩小了近一半,从而降低了表 TEST 的 HWM 值;(2)再次执行全表扫描的查询 SQL:select * from test;查询时间缩短为 1 s,SQL 执行速度大大提高。

3.3 测试结论

在对高碎片表进行全表扫描读的访问方式时,碎片增加了不必要的物理读与内存读,也就增加了不必要的物理 I/O 与 CPU 的消耗,最终降低了对表数据的访问速度,即影响了SQL 语句的响应时间。通过 shrink 或者 move 操作对表碎片空间进行重组,可以有效降低表中的 HWM 值,提高表的访问效率,进而提高 block 的命中率,在一定程度上,可以起到系统优化的作用。

本文针对 HWM 下碎片问题对性能的影响,探讨减少碎片空间的优化策略,通过对碎片空间的重组来减少碎片的产生,以提高访问效率。

数据库性能优化是一项复杂的系统工程,是一个循序渐进的过程,应该针对 Oracle 运行过程中出现的各种问题,找出性能瓶颈,有针对性地对系统进行调整,保证数据库高效可靠的运行。

参考文献

[1] 高敬媛,赵克宝.校园网数据库性能优化技术[J].煤炭技术,2011,30(07):226-228.

[2] KYTE T, ORACLE E, Signature edition programming techniques and solutions for Oracle 7.3 through 8.1.7 (Expert One-On-One)[M]. New York: Apress, 2010.

[3] KYTE T. Expert Oracle database architecture: 9i and 10g programming techniques and solutions[M]. 2006, San Bernardino: Macsource press, 2006.

(收稿日期:2013-01-27)

作者简介:

蔡焰,女,1976 年生,副研究馆员,硕士,主要研究方向:数据库技术有网络信息技术。