基于分区表的 RAC 优化技术应用①

唐世伟[1]，许璟龙[1]，刘万伟[2]，荣海亮[2]

[1](东北石油大学 计算机与信息技术学院，大庆 163000)
[2](大庆油田有限责任公司 勘探开发研究院，大庆 163000)

摘 要：OracleRAC 架构是从 Oracle9i 开始采用的一项新技术，是高可用性的一种，也是 Oracle 数据库支持网格计算环境的核心技术。RAC 的性能主要取决于 Cache Fusion 的性能。通过分析 Cache Fusion 的原理以及产生等待事件的原因，利用分区技术分散实例的数据请求，减少 Cache Fusion 的压力，充分发挥了 RAC 的性能。

关键词：高可用集群；性能调优；复合分区表；等待事件

Application of Optimization Technology of the RAC Based on Partition Table

TANG Shi-Wei[1], XU Jing-long[1], LIU Wan-wei[2], RONG Hai-liang[2]

[1](College of Computer and Information Technology, Northeast Petroleum University, Daqing 163000, China)
[2](Exploration and Development Institute, Daqing Oilfield Company Ltd., Daqing 163000, China)

Abstract: Oracle RAC architecture is a new technology which began to be used from Oracle9i. It is a kind of high-availability technology, and also is the core technology of Oracle database support for grid computing environment. The performance of RAC mainly depends on the performance of Cache Fusion. Through analyzing the principle of Cache Fusion and the reasons of generated wait events, and taking the advantage of Partition technology to scatter data request of instance, it can reduce the pressure of Cache Fusion, and fully play the performance of the RAC.

Key words: high availability clusters; performance tuning; composite partitioned table; wait events

目前，随着企业信息化的不断发展和集群技术的广泛应用，更多的用户选择了 Oracle 实时应用集群（RAC），对于数据库管理员来讲，如何有效的管理，优化数据库，成为越来越艰巨的挑战。Oracle RAC 所有的节点都共享一份磁盘数据,实例间通过高速缓存合并(Cache Fusion)机制进行数据同步，所以 RAC 的性能在很大程度上受限于 Cache Fusion 的性能[1]。如果过多的数据块通过 Cache Fusion 在实例间传递，就会产生 global cache cr request 等待事件。本文通过研究 Cache Fusion 机制，利用“分散数据”的思想，设计出一种分区表结构，以达到减少节点间数据传递的目的，从而提高了 RAC 的性能。

1 Cache Fusion 的原理

在 Oracle9i 之前，RAC 的名称是并行服务器(OPS Oracle parallel Server)。RAC 与 OPS 之间的一个较大区别是，RAC 采用了 Cache Fusion 技术[2]。在 OPS 中，节点间的数据请求需要先将数据写入磁盘，然后发出请求的节点才可以读取该数据。而使用 Cache Fusion 时，RAC 的各个节点的数据缓冲区通过高速、低延迟的内部网络进行数据块的传输。Cache Fusion 是实现 RAC 的根本技术，是一种协议，能够使各实例将他们的数据缓存合并为一个共享的全局缓存，由全局缓存服务（GCS Global Cache Service）协调共享。Cache fusion 就是通过高速的内网互联，在实例间进行数据块传递，它是 RAC 最核心的工作机制，它把所有实例的系统全局区系统全局区（SGA System Global Area）虚拟成一个大的 SGA 区。每当不同的实例请求相同的数据块时，这个数据块就需要在实例间进行传递。因为通过内网互联的数据传递要快于磁盘写，因此 RAC 的性能和伸缩性得到了极大的提高[3]。Cache Fusion 的一个最常见的读操作场景的流程（见图 1）。

① 实例 1 想要读取某个不在本地数据块时，就会向 GCS 提出请求；

① 收稿时间:2011-07-05;收到修改稿时间:2011-07-28

② GCS 会将请求发送给数据块的持有者实例 2；

③ 实例 2 接收到信息，锁管理服务器进程（LMS）将数据块传递给实例 1 并且自己保留一个残像（PI past image）；

④ 实例 1 收到数据块后，通知 GCS。

Cache Fusion 的一个最常见的写操作场景的流程（见图 2）

① 实例 2 向 GCS 提出写数据请求；

② GCS 将请求发送给持有当前版本数据块的实例 1；

③ 实例 1 收到请求，将数据写入磁盘；

④ 通知 GCS 写操作结束；

⑤ GCS 通知实例 2 删除持有的 PI。

通过图 1-2 可以看出，写请求发起实例并不一定是将 cache 写入到 disk 的实例,真正写入到 disk 的实例是持有当前版本的实例。

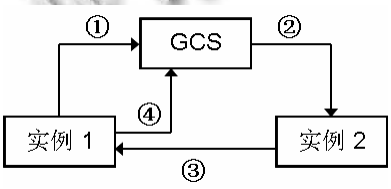

图 1 读数据图

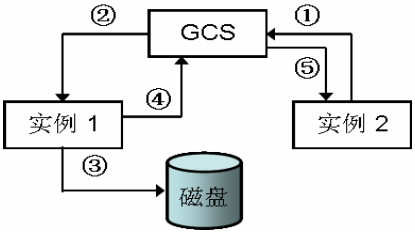

图 2 写数据

Cache Fusion 虽然提供全局数据协调共享的功能，但不能解决所有 RAC 的优化性能。在 RAC 中访问数据块时，先从本地的数据缓存中寻找，然后利用 Cache Fusion 在其他实例中寻找，最后才从磁盘读取[4]。如果经常全表扫描，势必导致数据块在实例间频繁传递，增大了 Cache Fusion 的压力，从而给系统造成了额外的开销。如果是 Insert 操作密集型的应用，当实例 2 占某据数据块时，如果此时实例 1 也要用到该数据块，就必须等待。此时就产生了等待事件 global cache buffer busy,而大量的 global cache buffer busy 导致了大量的 global cache cr request 等待[5]。

2 利用分区表分散数据

适当并正确的使用分区技术可以减少 RAC 节点间的通讯量及大表的全表扫描开销，通过分区表来实现数据块被某个实例独立使用，从而减少热块的争用[6]。这种方法是针对 RAC 进行的应用优化，让不同的应用运行在不同的节点上，也是在 RAC 上最底层的优化方法。

2.1 分区的优势及时机

首先通过使用数据分区技术可以大大提高访问速度，分区可以显著提高访问大表时的性能，并且分区的存在对应用系统是透明的[7]。其次可以提高关键数据的可用性，在数据库正常运行时保持高可用性，按分区备份。快速灾难恢复，首先恢复最关键的数据分区。

并非所有的表都适合做分区处理，Oracle 建议当表的数据量超过 2G 时就应该考虑进行分区，主要还是要分析应用层面，是不是适合做分区[8]。

2.2 Oracle 提供的基本分区方法

2.2.1 范围分区（range partitioning）

范围分区是依据用户创建分区时设定的分区键值（partition key value）范围将数据映射到不同范围。范围分区是比较常用的分区方式，通常针对日期数据使用。

2.2.2 列表分区（list partitioning）

可以使用户明确地控制将数据行映射到各个分区。用户在各分区的定义中指定一个分区键离散值的列表，从而实现列表分区。

2.2.3 哈希分区（hash partitioning）

将不适于采用范围分区和列表分区的数据进行分区[9]。

2.3 复合分区方法

由于实际环境的复杂，如果单一使用上述的分区方式，很难在系统中发挥其作用，综合考虑各项因素，以复合分区（composite partitioning）最为优越。复合分区先根据范围方式进行分区，再使用哈希或列表方式创建子分区（subpartition）[10]。范围-哈希复合分区既能发挥范围分区的可管理优势，也能发挥哈希分区的数据分散、条带化、并行化优势。

而之前的应用并没有针对 RAC 进行优化，所以决定从底层机构上进行调整，在表里面加一个字段，并且每次进行插入操作的时候给这个字段赋一个常量，并且重新对表进行分区，使之成为一个复合分区表。

在下面这张表上加入一个字段 Node_id，也就是实例编号，这个字段作为主分区字段，按照这个分区做范围分区后，再设置主键 WELL_ID 的 hash 分区，一共 16 个分区。

调整后的表结构如下：

```
CREATE TABLE WELL_PARTITION
(WELL_ID    VARCHAR2(20) not null,
PROD_DATE    DATE not null，
OIL_PROD_METHOD    VARCHAR2(20),
UP_CURRENT NUMBER(5,1),
DOWN_CURRENT VARCHAR2(20),
WATER_MIX_TEMP VARCHAR2(20),
RETU_OIL_TEMP VARCHAR2(20),
AVG_CASING_PRES VARCHAR2(20),
TUBING_PRES VARCHAR2(20)
Node_id        number
) initrans 20 pctfree 20 tablespace    TS_WELL
PARTITION BY RANGE(Node_id)
SUBPARTITION    BY    HASH(WELL_ID)SUBP
ARTITIONS 16
(PARTITION p1 VALUES LESS THAN (2),
PARTITION p2 VALUES LESS THAN (3))
;
```

Node_id 的值就是插入进程连接到的实例的 ID，这样调整后，在 1 号节点上插入的数据将全部被插入到 Node_id =1 的分区，这样的设计是为了避免产生大量的 global cache cr request，从而降低 Cache Fusion 的压力，因此，该方法是在 rac 架构上解决 GC BUFFER BUSY 的有效方法。

3　RAC实例测试

以油田数据维护管理应用为例，该应用主要是对油田的数据进行管理，因此会产生很多的插入性操作，选择单井文档维护管理表进行测试，该表数据量达到 3.4GB，并且范围查询的比例较小，且无法有效划分范围，以上这些情况很符合范围-哈希分区的标准。

通过使用 ASH（Active Session History）得到Oracle 活动会话的历史信息：

表 1　使用复合分区前的 Top Events

Event	Event Class	% Activity	Avg Active Sessions
buffer busy waits	Concurrency	71.48	9.23
enq: HW - contention	Configuration	14.31	1.82
gc buffer busy	Cluster	8.42	0.89
CPU + Wait for CPU	CPU	3.13	0.37

表 2　使用复合分区后的 Top Events

Event	Event Class	% Activity	Avg Active Sessions
CPU + Wait for CPU	CPU	78.45	1.02
db file scattered read	User I/O	7.02	0.09
enq: TX - row lock contention	Application	2.59	0.03
log file sync	Commit	2.01	0.03

通过对比表 1 与 2 可以明显的看出，优化后系统的 gc buffer busy 等待事件已经不在 Top Events 里，系统运行平稳。

4　结论

随着大庆油田勘探开发信息技术的不断进步，数据量的急剧增长，针对基于 RAC 的勘探开发数据库系统中的优化具有重大意义。本文提出的基于 Oracle 分区技术的 RAC 优化方案,大大提高了系统性能及可维护性。通过对表结构的修改，采用复合分区方法，降低了等待事件，减少了数据块的争用，达到了优化的效果。

参考文献

1 Oracle Real Application Clusters 10g. 2006–01. http://www.oracle.com/lang/cn/technologies/grid/index.html.

2 张晓明.大话 Oracle RAC 集群高可用性备份与恢复.北京:人民邮电出版社,2009.

3 姜召凤.Oracle RAC 数据库缓存优化方法研究.大连海事大学,2009.

4 付社良,田斌.Oracle RAC 10g 系统高可用性测试及分析.武汉理工大学学报,2007,(2).

5 刘晓辉,姚惠东.RAC 技术在医院信息系统中的应用研究.中国医疗器械杂志,2010,(4).

6 袁韬.RAC 技术在零售业企业信息系统中的应用研究.现代计算机,2010,(12).

7 刘让国,蒲宝明,杜圣东,王守能.基于海量空间数据库的高可用性研究与应用.计算机工程,2008,(6).

8 Kevin L. Oracle Database 10g: the Complete Reference. USA: Osborne Oracle Press Series, 2004.

9 王晗.基于 Oracle 的 XX 公安厅数据平台性能优化和研究.长春理工大学,2010,(8).

10 Oracle.Oracle metalink site.http://metalink. oracle. com.