如何使用AWR报告发现

低效的SQL



华为技术有限公司

2012年03月

修订记录Revision record

|  |  |  |  |
| --- | --- | --- | --- |
| 日期  Date | 修订版本Revision version | 描述Description | 作者Author |
| 2012-03-05 | 1.0 | 初始版本 | 马辉（00133547） |
|  |  |  |  |
|  |  |  |  |

# 前言

目前发现很多局点在上线以后有很多低效的sql，导致cpu繁忙、磁盘I/O繁忙、效率低下，影响系统的正常运行。如何在测试环节或上线以后发现这些低效的sql呢？ Oracle AWR报告可以很好的帮我们解决这个问题。

awr报告是oracle 10g以上的版本提供的一种性能收集和分析工具，它能提供一个时间段内整个系统资源使用情况的报告，通过这个报告，我们就可以了解一个系统的整个运行情况，这就像一个人全面的体检报告。

90%的数据库性能问题是由于低效的SQL导致的，所以本文将主要介绍如何根据AWR报告查找低效的SQL。如果想深入学习，可以参考附件《Oracle数据库性能分析专题V1.0-20040930-B.zip》。

下面首先来看怎样获取AWR报告：

# 收集AWR报告

1. 登陆数据库:

sqlplus / as sysdba

(2)输入：

@?/rdbms/admin/awrrpt.sql

(3)提示输入awr报表的类型:

屏幕显示:

Enter value for report\_type:

请输入html或者直接按enter(默认为html)

(4)提示输入几天内的报表:

屏幕显示:

Enter value for num\_days:

输入1

(5)提示输入起始snap id和终止snapid(snap id对应时间请看Snap Started),起始时间和结束时间之间应该为业务高峰期.

屏幕显示: Enter value for num\_days: 1

Listing the last day's Completed Snapshots

Snap

Instance DB Name Snap Id Snap Started Level

------------ ------------ --------- ------------------ -----

mdspdb MDSPDB 3330 17 Dec 2010 00:00 1

3331 17 Dec 2010 01:00 1

3332 17 Dec 2010 02:00 1

3333 17 Dec 2010 03:00 1

3334 17 Dec 2010 04:00 1

3335 17 Dec 2010 05:00 1

3336 17 Dec 2010 06:00 1

3337 17 Dec 2010 07:00 1

3338 17 Dec 2010 08:00 1

3339 17 Dec 2010 09:00 1

3340 17 Dec 2010 10:00 1

3341 17 Dec 2010 11:00 1

3342 17 Dec 2010 12:00 1

3343 17 Dec 2010 13:00 1

3344 17 Dec 2010 14:00 1

3345 17 Dec 2010 15:00 1

Specify the Begin and End Snapshot Ids

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter value for begin\_snap:

输入一个id(如3309)后,屏幕提示输入截止id:

Enter value for end\_snap:

输入截止id（如3340）

(6)提示输入生成报表的名字（后缀名一定为.html），也可不输入，使用默认名称。

屏幕提示：

The default report file name is awrrpt\_1\_3339\_3340.html. To use this name,

press <return> to continue, otherwise enter an alternative.

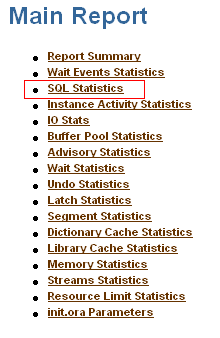
Enter value for report\_name:

(7)退出sqlplus后在当前目录下会看到生成的awr报表。

# 查看AWR报告

使用ftp工具将生成的AWR报告取下来（ASCII码方式下载），就可以查看报表了。报表是HTML格式的，直接打开即可。

打开报告以后，可以看到报告的组成部分的链接：



点击SQL Statistics就可以直接跳转到SQL 执行统计部分。

## SQL ordered by Elapsed Time部分：

在这个部分，主要统计的是整体执行时间的top sql。排列顺序是按照sql总体执行时间排序，其中：

Elapsed Time (s)= Executions\* Elap per Exec (s)

说明

Elapsed Time (s) = sql 整体运行时间

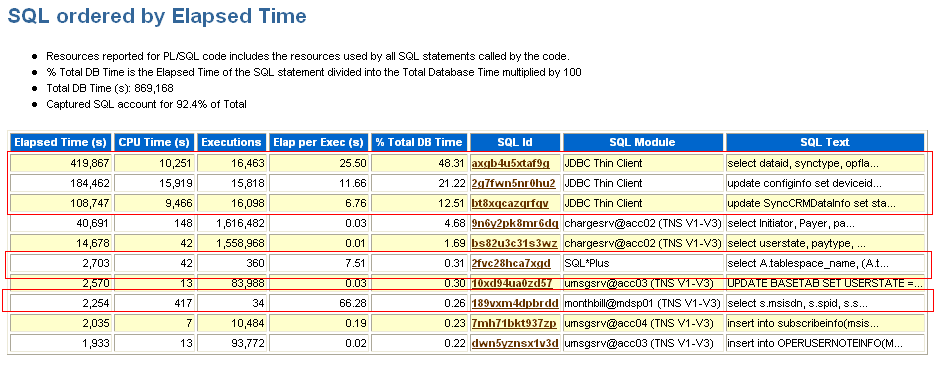
Executions = sql执行次数

Elap per Exec (s) = sql单次执行时间

我们应该重点关注“Elap per Exec (s)”列，因为sql总体执行时间对于我们来说没有太大意义，有可能此sql执行次数很多，单次执行并不长，从而整体执行时间较长。

如果想查看此sql的具体语句，可以点击“SQL Id”进行查看。

下面我们来看一个例子，



一般在OLTP系统中，普通的业务处理的SQL 单次执行在几毫秒就会返回，但是上面用红色标注的sql都达到几秒或者几十秒，所以这些sql是重点优化对象。

说明

当然也有一些特殊的语句执行时间教长，这些sql很多都是批量处理的，比如删除过期数据之类的。

## SQL ordered by CPU Time部分：

在这个部分，主要统计的是SQL占用cpu时间的top sql。排列顺序是按照sql占用CPU时间排序，其中：

CPU Time (s)= Executions\* CPU per Exec (s)

说明

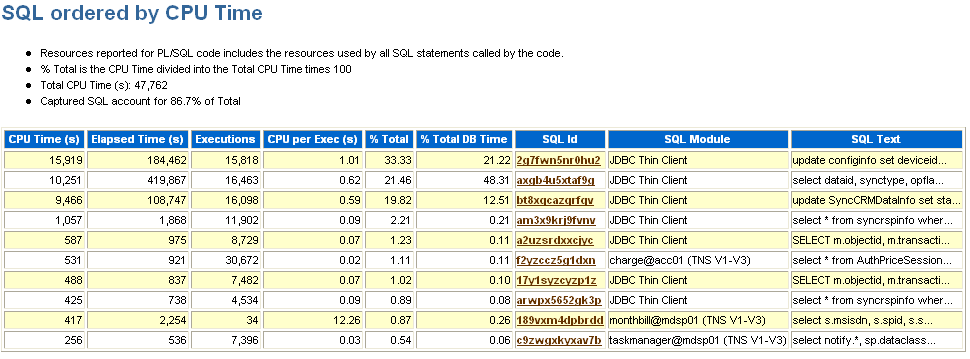
CPU Time (s) = sql 占用cpu时间

Executions = sql执行次数

CPU per Exec (s) = sql单次执行占用CPU时间

我们应该重点关注“CPU per Exec (s)”列看是否单次执行sql占用CPU时间过长。

如下图所示：



在实际现网问题处理过程中，可以不用关注此统计项，可以用“SQL ordered by Gets”统计项代替。

## SQL ordered by Gets部分：

在这个部分，主要统计的是SQL逻辑读的次数。啥是逻辑读呢？我们执行查询的时候，频繁查询的数据会缓存到数据库的高速缓存区中，再次查找这些数据的时候就直接从内存中读取，不需要从物理文件中读取，这样会大大提高查询的速度。在内存中查找数据的时候，如果能走正确的索引，则在3-100个左右的逻辑I/O之内就会将数据查询出来，如果系统中出现单次逻辑读远远超过100的SQL，这些SQL都需要进行优化。如果逻辑I/O次数过多，则会消耗更多的CPU， 一般“SQL ordered by CPU Time”和“SQL ordered by Gets”是一一对应的，是一个因果的关系，所以CPU过高，则重点关注SQL ordered by Gets部分即可。

其中：

Buffer Gets = Executions\* Gets per Exec

说明

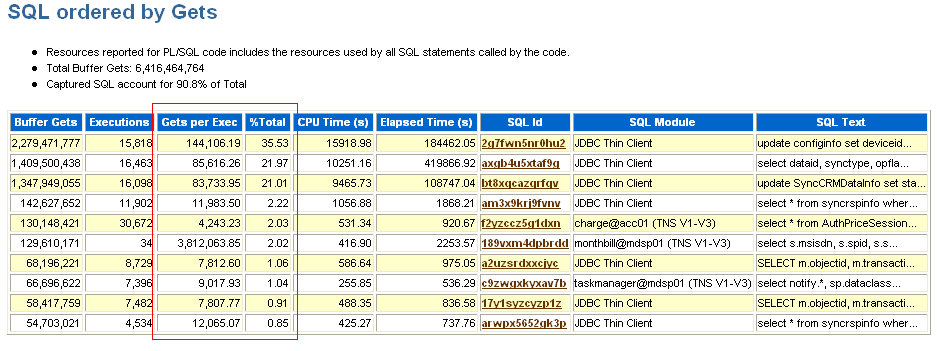
Buffer Gets = sql 总共所用逻辑I/O

Executions = sql执行次数

Gets per Exec = sql单次执行逻辑I/O次数

我们应该重点关注“Gets per Exec”列看是否逻辑读过高。

如下图所示：



可以看到所有的sql都有些问题，但是我们应该优先优化单次逻辑读过高，而且占用整体逻辑读百分比（%Total）的这些sql，也就是前三个SQL。

可以看到“SQL ordered by Gets”和 “SQL ordered by CPU Time”的两个图都差不多，所以一般只需要看一个“SQL ordered by Gets”即可。

## SQL ordered by Reads：

在这个部分，主要统计的是SQL物理读的次数。啥是物理读呢？我们执行查询的时候，如果在内存中找不到所需的数据，则会直接在物理文件中读取，则在3-100个左右的物理I/O之内就会将数据查询出来，如果系统中出现单次物理读远远超过100的SQL，这些SQL都需要进行优化。如果物理I/O次数过多，则会导致SQL执行效率非常低下，同时会导致磁盘I/O过高影响系统问题运行。磁盘I/O过高，则需重点关注SQL ordered by Reads部分。

其中：

Physical Reads = Executions\* Reads per Exec

说明

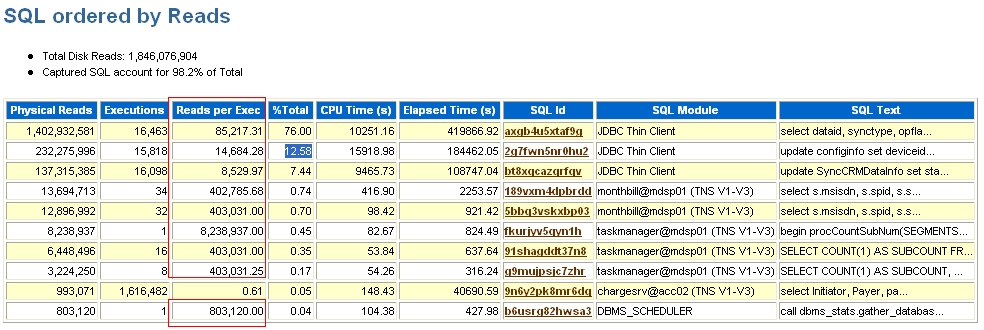
Physical Reads = sql 总共所用物理I/O

Executions = sql执行次数

Reads per Exec = sql单次执行物理I/O次数

我们应该重点关注“Reads per Exec”列看是否单次物理读过高。

如下图所示：



可以看到大部分sql都有些问题，但是我们应该优先优化单次物理读过高，而且占用整体物理读百分比（%Total）的这些sql。可以看到第一个sql单次物理读很高，而且占所有物理读的76%， 分析后发现所查的表没有加相应的索引。加上索引以后整个系统的I/O立马就下来了，得到了质的提升。

# 附录

一般介绍的AWR的可参考数据库TMG出的文档：（awr是statpack的升级版，内容基本一样，可查看第四章的分析）：

