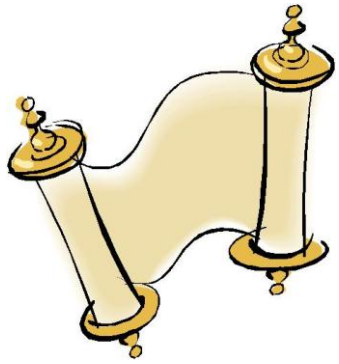
FINDING HISTORICAL FILE I/O



LOOKING INTO THE PAST

THE NEED FOR HISTORY

I recently had a request from an in-house customer to analyze the total amount of disk i/o performed by our application. My customer wanted to see if changes to the app had increased the demands on the disk. Fortunately for me, we had a pretty easy way to do this.

AWR NOT GOOD ENOUGH

If you rely totally on the AWR tables, you will not have enough history to get meaningful answers. Depending on your retention settings, you will likely have about 30 days of history. Normally, that's just fine, but not so good for historical comparisons.

We have found it useful to save a subset of the AWR tables for a much longer period. We don't bother with the massive ASH tables, since they aren't so vital for runtimes. So, everyday we save the all the entries from just five AWR tables using a daily PL/SQL job. If you restrict your long-term retention to just the tables below, the added storage will be small.

We save the data in these slightly-renamed tables, and put them in their own schema:[1]

- da_dba_hist_snapshot
- da_dba_hist_sqlstat
- da_dba_hist_sysstat
- da_dba_hist_system_event
- da_dba_hist_service_stat

Once you have the mechanism in place to save this critical information, there are lots of ways to use it. Let's take a look at just one way.

[1] Thanks to Richard Flores for creating the original script that flawlessly gathered this information.

FINDING HISTORICAL FILE I/O

A USEFUL SCRIPT

Focusing on our customer request to show historical disk i/o, we can use our special table, Da_Dba_Hist_Sysstat, to retrieve the necessary metrics. I decided to roll-up the data to a week, since the daily variation may be large. Also, we have eight nodes in a RAC cluster, so I will sample one busy node, node 4. This script finds the historical number of disk reads, and groups by week number:

```
With S1 As (
Select /*+Parallel(X 10) */
Snap_Id, (Sum(Value/1000000000)) Gb
From  Da.Da_Dba_Hist_Sysstat X
Where Instance_Number = 4
And Stat_Name In('physical read bytes')
Group By Snap_Id),
--
S2 As (Select Snap_Id,
Gb - Lag(Gb,1) Over(Order By Snap_Id) Totrds From S1 )
--
Select
To_Char(Begin_Interval_Time, 'YYYY') YYYY,
To_Char(Begin_Interval_Time, 'WW') WW,
Round(Sum(Totrds)) GB_READS From S2, Da.Da_Dba_Hist_Snapshot S
Where S.Snap_Id = S2.Snap_Id
And S.Instance_Number = 4
And Totrds > 0
Group By
To_Char(Begin_Interval_Time, 'YYYY'),
To_Char(Begin_Interval_Time, 'WW')
Order By 1,2;
```

```
YYYY WW   GB_READS
---- -- ----------
2012 13      44451
2012 14      43499
2012 15      29380
2012 16      39722
2012 17      46479
2012 18      45285
2012 19      47711
```

FINDING HISTORICAL FILE I/O

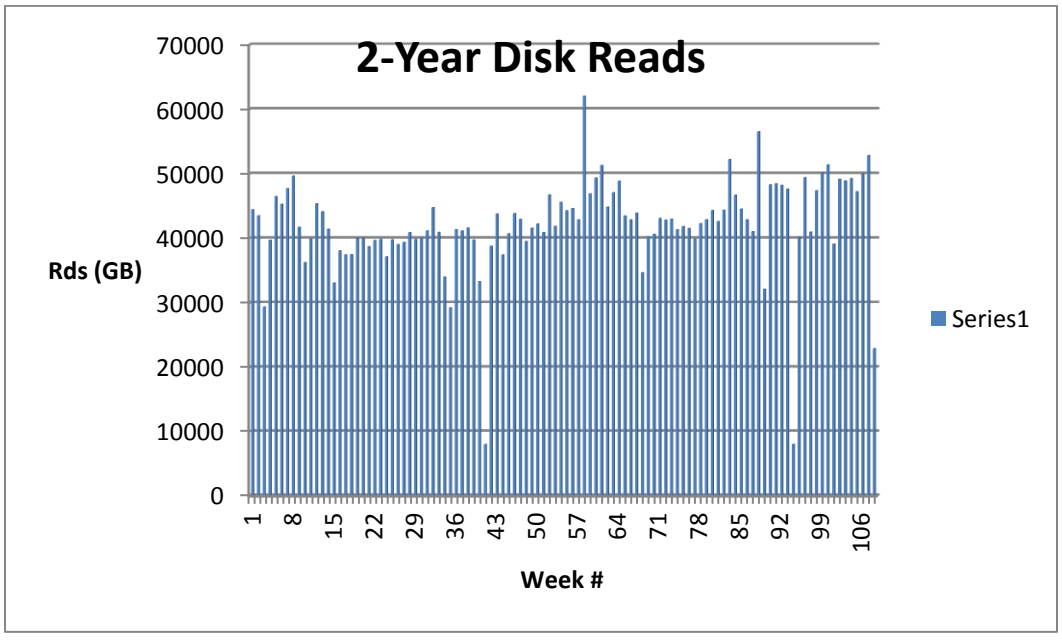
```
2012 20       49652
2012 21       41708
2012 22       36269
2012 23       39911
2012 24       45352
2012 25       44132
2012 26       41426
2012 27       33086
2012 28       38059
```

I decided to focus on the total amount of bytes read from disk. We use the "Lag" analytic function to convert cumulative statistics to delta values by Snap_Id. Also, note that we specific the Instance_Number. Otherwise, the Lag will not be mixing up different instances.

Observe that the statistic name, "physical read bytes" is in lower case, like the other metrics. This is an odd way of formatting things, so be careful. When formatting the code, I accidentally changed the case; then, of course, nothing worked.

DISPLAY RESULTS

As always, I find it best to display the results graphically. In the graph below, one can see that there has only been a gradual increase in the total amount of disk reads over two years. The two large dips are apparently due to a system outage.



FINDING HISTORICAL FILE I/O

ISSUES

I notice that the very first entry in my results is often not correct. I believe this has something to do with the first use of the lag function. For my purposes, I just removed the first, erroneous entry.

SUMMARY

By using our "extended" AWR tables, it's possible to get some very useful information on historical I/O. This paper illustrates just one benefit of keeping these extended tables.

Chris Lawson is an Oracle *Ace* and performance specialist in San Francisco. He is the author of *The Art & Science of Oracle Performance Tuning*, as well as *Snappy Interviews: 100 Questions to Ask Oracle DBAs*. When he's not solving performance problems, Chris is an avid geocacher, where he is known as *Bassocantor*. Chris can be reached at: Chris@OracleMagician.com