Advanced Usage of the AWR Warehouse

Author: Kellyn Pot'Vin-Gorman, Consulting Member of Technical Staff, SCP Team

Introduction

My initial introduction to the concept of an AWR Warehouse occurred back in 2009. The company I worked for had a very unique situation, where once per week, a datamart was dropped and a new image was created from the ever-growing data warehouse. Ensuring consistent performance in a single database is difficult enough, the challenge of guaranteeing this in a new database each week is a whole new demand on an optimization database administrator. Due to this challenge, I took a set of scripts that Karl Arao and I had enhanced over a period of a couple years via email into a full AWR Warehouse. This performance data repository offered me answers to many of the company's performance questions time and time again, so when Oracle introduced its impressive offering, I was sold.

Since then, I've had the opportunity to work with both the Oracle EM12*c* Cloud Control AWR Warehouse interface, as well as work with the repository via SQL*Plus. I've dug in deep to understand what options and performance advantages exist with the AWR Warehouse repository and played devil's advocate when the chance has arisen. I'm pleased to talk about all the impressive features that a centralized AWR Warehouse offers the IT business.

Design and Function

The AWR Warehouse is set up with the same objects as you are accustomed to in a standard AWR schema of any Oracle database. The enhancement lies in the partitioning, (either by DBID, SNAP_ID or a combination of both) that allows for quick loads, efficient querying and when requested, effective purging of unwanted data.

The jobs to both extract data from the source target, as well as load into the AWR repository is designed with "throttles" to ensure that if historical loads are being performed, no impact to user performance is felt.

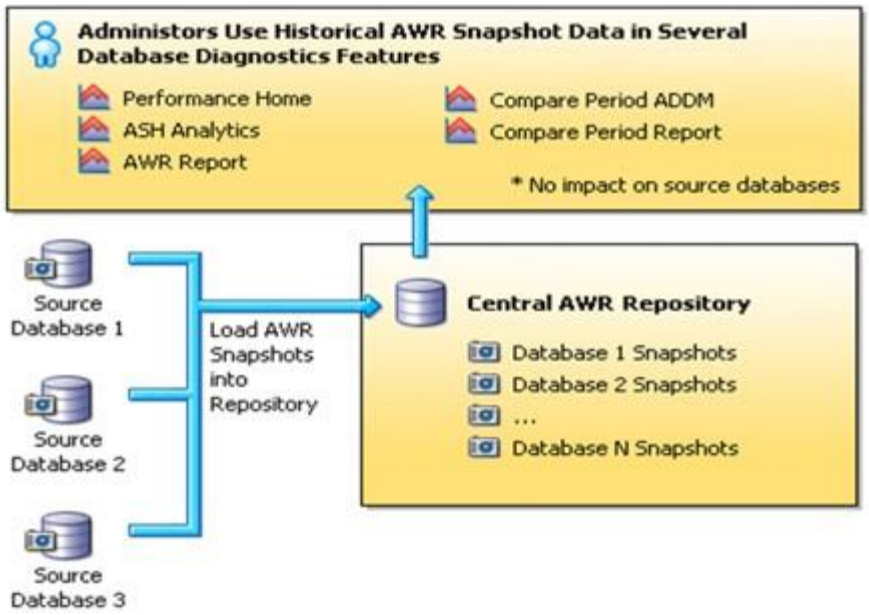


Figure 1.1 The AWR Warehouse Architecture and ETL Load process.

If for some reason the database was unavailable for uploads to the AWR Warehouse, due to maintenance or other outage, there are added "throttles" to ensure an ETL load is never too large and that oldest data is always loaded first to ensure retention times don't impede the ability to offload the valuable AWR data to the repository.
During a "catch-up" period, the ETL jobs are run once every three hours instead of the once every 24hr. standard interval. The files are transferred from the source target file system to the AWR Warehouse server via a "Agent to Agent" push. This limits the pressure on the network and at no time, touches the EM12c Oracle Management Service, (OMS) and/or Oracle Management Repository, (OMR) server.

Sizing the AWR Warehouse Properly

When sizing the AWR Warehouse, one of my requirements is to have certain reports for a sampling of databases that will source the AWR Warehouse. This report provides me the right information to create the correct sizing requirements vs. any assumptions done with other choices. The report is the General AWR Information report and can be found in the $ORACLE_HOME/rdbms/admin directory. It provides the following information:

- Time of Report
- AWR Retention and interval settings, including if non-default setting.
- Schema percentage of space used in the SYSAUX tablespace.
- Objects and space consumption in the SYSAUX tablespace, including breakdown of the AWR space usage by type.

- Automated tasks scheduled.
- ASH Usage Info
- Foreground vs. Background processing info.

I've now run into a few projects where questions were raised on why so much space was required and there is sometimes an issue with the data being retained in the AWR that must be identified or addressed that will impact the long term size demand on the warehouse before the data is brought over from the source database, (target) via the ETL to the AWR Warehouse. Knowing how to identify this is crucial, but many folks aren't looking at AWR data space usage regularly, so how would you know? Well, that's what this post is about and hopefully will save me time with how much is on my plate these days…

Executing the AWR General Info Report

To run the AWR Info report, log into the host of the database in question as a user that has rights to AWR reports. From the directory you would like to have the report housed, run the following:

```
SQL> $ORACLE_HOME/rdbms/admin/awrinfo.sql;
```

Once the report is generated, simply open it in a text editor on the host or FTP it to your workstation.

We'll start with what a standard AWR Info report with "normal" space usage looks like and use it as a baseline. This will help you, as a DBA understand what is common for AWR space consumption.

Our Standard retention and interval is 8 days and 60 minute intervals on snapshots and the space usage for the SYSAUX tablespace looks like this:

```
**************************************************
(1a) SYSAUX usage - Schema breakdown (dba_segments)
**************************************************
|
| Total SYSAUX size       2,993.9 MB ( 74% of 4,046.0 MB MAX with
AUTOEXTEND OFF )
|
| Schema SYS occupies     2,750.9 MB ( 91.9% )
| Schema XDB occupies     67.3 MB ( 2.2% )
| Schema AUDSYS occupies  65.4 MB ( 2.2% )
| Schema MDSYS occupies   61.7 MB ( 2.1% )
| Schema ORDDATA occupies 16.1 MB ( 0.5% )
| Schema SYSTEM occupies  15.7 MB ( 0.5% )
| Schema WMSYS occupies   7.1 MB ( 0.2% )
| Schema EXFSYS occupies  3.7 MB ( 0.1% )
| Schema CTXSYS occupies  3.7 MB ( 0.1% )
```

Non-Default ASH Settings

If the settings have been changed from the default, the AWR info report will display the settings and let you know they aren't the default. Space consumption will change vs. what you see in our first example, too.
Warning: Non Default AWR Setting!
————————————————————————————

Snapshot interval is 30 minutes and Retention is 8 days (5.2 GB, vs. 3GB for a 60 min. interval.)

OR

Snapshot interval is 60 minutes and Retention is 42 days (10.5 GB)

With an increase in interval or retention, an increase in space consumption will result, but it won't be 1:1. There are two features that impact what space is required- rollup, which saves space, then partitioning, that requires a bit more, so approximately 70% increase on average with the interval to every 30 minutes.

When an increase in retention of AWR data is implemented, then you should calculate about 2.5G of data for each one week of AWR data retained, at 1hr interval on snapshots, (this depends on version of the database, too. AWR in 10g is much smaller than 11g, which is also smaller than 12c…)

AWR Components

We also need to look at the consumption used by a standard AWR schema breakdown to understand WHAT components are using the space:

```
**********************************
(3a) Space usage by AWR components (per database)
**********************************
COMPONENT MB % AWR KB_PER_SNAP MB_PER_DAY MB_PER_WEEK TABLE% :
INDEX%
--------- --------- ------ ------------ ---------- ----------- -
---------------
FIXED         1,559.1 63.3 7,750 181.6 1,271.5 45% : 55%
EVENTS        489.9 19.9 2,435 57.1 399.5 40% : 60%
SQL           238.3 9.7 1,184 27.8 194.3 64% : 36%
SPACE         111.1 4.5 552 12.9 90.6 63% : 37%
ASH           35.3 1.4 175 4.1 28.7 83% : 17%
SQLPLAN       11.0 0.4 55 1.3 9.0 64% : 36%
SQLTEXT       0.9 0.0 5 0.1 0.8 87% : 13%
SQLBIND       0.6 0.0 3 0.1 0.5 50% : 50%
RAC    0.6 0.0 3 0.1 0.5 50% : 50%
```

Note that fixed objects are at the top of the list, followed by events, SQL, space and then, ASH. This is how the flow of greatest to least should commonly be displayed.

Now lets looks at an AWR Info report where the data consumption is experiencing an issue:

```
**********************************
(3a) Space usage by AWR components (per database)
**********************************

COMPONENT MB % AWR KB_PER_SNAP MB_PER_DAY MB_PER_WEEK TABLE% :
INDEX%
--------- --------- ------ ------------ ---------- ----------- -
---------------
ASH           2,410.3 42.5 1,494 70.0 490.2 89% : 11%
FIXED         2,149.7 37.9 1,332 62.5 437.2 48% : 52%
EVENTS        489.7 8.6 304 14.2 99.6 43% : 57%
SPACE         224.4 4.0 139 6.5 45.6 58% : 42%
SQL           160.6 2.8 100 4.7 32.7 55% : 45%
SQLPLAN       82.0 1.4 51 2.4 16.7 67% : 33%
RAC           58.3 1.0 36 1.7 11.8 70% : 30%
SQLTEXT       7.3 0.1 5 0.2 1.5 96% : 4%
SQLBIND       6.0 0.1 4 0.2 1.2 33% : 67%
```

Note that the ASH data is the first component listed and the size is extensively larger than the FIXED, EVENTS, etc. There are numerous reasons for this to have occurred, so we'll investigate what could have caused the increase in space consumption, as over time, the extended retention into the AWR Warehouse will consume more space on the destination side, increasing requirements for the AWR Warehouse.

When ASH is Extensive

First, we'll check to see what the minimum and maximum snap_id's from both the dba_hist_snapshot in comparison to the AWR:

```
select min(snap_id),MAX(snap_id) from dba_hist_snapshot;

MIN(SNAP_ID) MAX(SNAP_ID)
------------ ------------
       15027        15189

select min(snap_id),MAX(snap_id) from
WRH$_ACTIVE_SESSION_HISTORY;

MIN(SNAP_ID) MAX(SNAP_ID)
------------ ------------
           1        15189
```

As you can see, the AWR contains ASH data from the first snap_id when the dba_hist snapshot shows that only data from 15027 on should exist.

We'll next check for orphaned rows of ASH data in the AWR:

```
SELECT COUNT(*) FROM wrh$_active_session_history a
  WHERE NOT EXISTS
```

```
(SELECT 1
FROM wrm$_snapshot
WHERE snap_id       = a.snap_id
AND dbid            = a.dbid
AND instance_number = a.instance_number
);
```

If this exists, follow the steps from Oracle to **Manually Purge the Optimizer Statistics & AWR Snaphots to Reduce Space Usage of SYSAUX Tablespace (Doc ID 1965061.1)** to split the partitions and purge the data manually from the AWR from the SOURCE DATABASE, (target) to address before the ETL extracts and load the data to the AWR Warehouse.

Understanding ASH Parameters

The next reason for the extensive ASH data in the AWR could result in a change to the parameters involving how ASH data is written to the AWR. I've only recently heard that some shops are doing this as a way to "audit" the SQL happening in their databases. I have to admit, I would prefer to see DBAs use auditing features vs. use ASH samples to track this, but it is happening and they should expect the following:

1. Extensive space usage by the AWR

2. Inaccurate results in ASH and AWR reports due to Oracle expecting only 1:10 samples existing in the AWR and having 10:10 will impact the results.

The parameters controlling this feature are underscore parameters and should only be changed under the guidance of Oracle.

_ash_sampling_interval = 100 The interval that ASH samples, lessened, causing samples to be created more often than the default of 1000.
_ash_sample_all = TRUE True results in samples of even inactive sessions to be created, increasing the amount of ASH data by 10X or more.
_ash_disk_filter_ratio = 1 Would result in ASH writing all samples to the AWR instead of 1:10.

Once you've addressed any issues in storage of the AWR and loaded all snapshots to your new AWR Warehouse, also remember to "dial down" the retention in the source database to a the default of 8 days, (or something close) and shrink the SYSAUX tablespace to reallocate the space back to the database, having no longer need of the space it once consumed.

The AWR Warehouse does require considerable storage for the performance data housed within the Automatic Workload Repository, but with the right process to inspect what is kept in your AWR before building out your environment, you can avoid having to allocate more storage than you really need to.

Once you know what is in your AWR in your source databases are prepared to bring into an AWR Repository, then you can set up the repository database to load this data into.

Requirements for the AWR Warehouse

The repository for the AWR Warehouse should be an 11.2.0.4 database or above and with the tuning and diagnostics pack, a limited EE license is available to use for the AWR Warehouse repository. Don't attempt to use your EM12c repository, (OMR) for the repository. Considering the amount of data that will be housed here and use type, the two repository use would be highly incompatible long-term. There are patches and other requirements, so see MOS note 1907335.1 for the complete list and detailed steps of installation. For general introduction and set up instructions, see the AWS Warehouse section of the Oracle Documentation set here.

We are going to proceed onto more important things, like how to query the AWRW directly!

Why Mapping is Important

If you were to take your AWR queries "as is" and run them in the AWR Warehouse, you can almost guarantee inaccurate results. To demonstrate this, we can take a specific AQL_ID: "d17f7tgcaa416" to clarify why.

In the following query, using SQL_ID, **'d17f7tqcaa416'** as an example, you quickly realize that the algorithm used to create the SQL_ID is not unique to the database, but is assigned viato the Oracle software and would be assigned to that query no matter what database it was run in. This is easily recognized as a feature if one were to trouble shoot performance from production to test to development or reporting where having a uniform generation of a unique identifier for a specific statement is valuable.

```
SQL>  select distinct(dbid) from dba_hist_sqlstat
  2  where sql_id='d17f7tqcaa416';

      DBID
----------
1912296936
2089309222
3373983684
1940814992
 287602056
815540361
1119635158
…
18 rows selected.
```

Due to this, any AWR query that is modified to run against the AWR Warehouse must have a join added to map the DBID so as to limit the results to the source target in question.

To map this data, we then inspect the AWR Warehouse DBSNMP schema and a very important table to the repository that is part of the AWR Warehouse:

```
DBSNMP.CAW_DBID_MAPPING
MAPPING_ID                    NUMBER(38)
EM_ID                         NUMBER
TARGET_NAME                   VARCHAR2(266)
TARGET_TYPE                   VARCHAR2(64)
OLD_DBID                      NUMBER
NEW_DBID                      NUMBER
```

This table has a simple, but effective design and is used to map data as part of ETL loads and will be used by EM12c to provide reports via the user interface against the AWR Warehouse and also by anyone wanting to query the AWR Warehouse efficiently.

```
SQL> select target_name, new_dbid from dbsnmp.caw_dbid_mapping;

TARGET_NAME          NEW_DBID
-------------------- ----------------------------------
dbm02                815540361
sid1                 1940814992
cawr                 2700230493
db311                1813086947
dbm01                1048341949
db313                1890415439
db305                1912296936
……
```

We can now easily add this table to our queries, join on the NEW_DBID, (if you rename your DBID, then understand why the OLD_DBID may be important for some historical queries….) and add the TARGET_NAME to your where clause.

Querying the AWR Warehouse

To update a query, we'll start with a simple query to inspect information about a particular SQL_ID and the CPU usage per execution plan.

```
SQL> select SQL_ID
, PLAN_HASH_VALUE
, sum(EXECUTIONS_DELTA) EXECUTIONS
, sum(ROWS_PROCESSED_DELTA) CROWS
, trunc(sum(CPU_TIME_DELTA)/1000000/60) CPU_MINS
, trunc(sum(ELAPSED_TIME_DELTA)/1000000/60)  ELA_MINS
from DBA_HIST_SQLSTAT S, DBSNMP.CAW_DBID_MAPPING M
where LOWER(M.TARGET_NAME) = '&dbname'
and M.NEW_DBID = S.DBID
and S.SQL_ID in ('&sqlid')
group by SQL_ID , PLAN_HASH_VALUE
order by SQL_ID, CPU_MINS;

Enter value for dbname: db305
old   9: where LOWER(M.TARGET_NAME) = '&dbname'
new   9: where LOWER(M.TARGET_NAME) = 'db305'
Enter value for sqlid: d17f7tqcaa416
old  11: and S.SQL_ID in ('&sqlid')
new  11: and S.SQL_ID in ('d17f7tqcaa416')
```

SQL_ID	PLAN_HASH_VALUE	EXECUTIONS	CROWS	CPU_MINS	ELA_MINS
d17f7tqcaa416	3035679780	1	1	0	0
d17f7tqcaa416	8947562239	1	1	2.21	1.79

With just a few, simple changes, I now can see that I have seen a change in plan values for the SQL_ID **d17f7tqcaa416** for the **db305**database.

We can then build out on this and add a second database for comparison:

```
SQL> select M.TARGET_NAME dbname
,  SQL_ID
, PLAN_HASH_VALUE
, sum(EXECUTIONS_DELTA) EXECUTIONS
, sum(ROWS_PROCESSED_DELTA) CROWS
, trunc(sum(CPU_TIME_DELTA)/1000000/60) CPU_MINS
, trunc(sum(ELAPSED_TIME_DELTA)/1000000/60)  ELA_MINS
from DBA_HIST_SQLSTAT S, DBSNMP.CAW_DBID_MAPPING M
where LOWER(M.TARGET_NAME) in('&dbname','&dbname2')←second DB
and M.NEW_DBID = S.DBID
and S.SQL_ID in ('&sqlid')
group by M.TARGET_NAME, SQL_ID , PLAN_HASH_VALUE
order by SQL_ID, CPU_MINS;

Enter value for dbname: db305
Enter value for dbname2: sid.us.oracle.com
old  10: where LOWER(M.TARGET_NAME) in('&dbname','&dbname2')
new  10: where LOWER(M.TARGET_NAME) in('db305','sid.us.oracle.com')
Enter value for sqlid: d17f7tqcaa416
old  12: and S.SQL_ID in ('&sqlid')
new  12: and S.SQL_ID in ('d17f7tqcaa416')

DBNAME     SQL_ID PLAN_HASH_VALUE EXECUTIONS  CROWS CPU_MINS   ELA_MINS
------------------------------------------------------------------------
sid.us.oracle.com   d17f7tqcaa416  1435720983   1     1      0      0
db305               d17f7tqcaa416  3035679780   1     1      0      0
db305               d17f7tqcaa416  8947562239   1     1    2.21   1.79
```

We've now demonstrated how a simple join offers performance data for the same query across more than one database.

The next query pulls more information, but still only requires the request for a DBNAME, (or two if you wish to compare or view more than one as we did in the previous query…) and then the join on the DBID to NEW_DBID.

```
column sample_end format a21
select to_char(min(s.end_interval_time),'DD-MON-YYYY DY HH24:MI')
sample_end
, q.sql_id
, q.plan_hash_value
, sum(q.EXECUTIONS_DELTA) executions
, round(sum(DISK_READS_delta)/greatest(sum(executions_delta),1),1)
pio_per_exec
, round(sum(BUFFER_GETS_delta)/greatest(sum(executions_delta),1),1)
lio_per_exec
,
round((sum(ELAPSED_TIME_delta)/greatest(sum(executions_delta),1)/1000)
,1) msec_exec
from dba_hist_sqlstat q, dba_hist_snapshot s, dbsnmp.caw_dbid_mapping
m
where m.target_name= upper('&dbname')
and q.SQL_ID=trim('&sqlid.')
and s.snap_id = q.snap_id
and s.dbid = q.dbid
and q.dbid = m.new_dbid
and s.instance_number = q.instance_number
and s.end_interval_time >= to_date(trim('&start_time.'),'dd-mon-yyyy
hh24:mi')
and s.begin_interval_time <= to_date(trim('&end_time.'),'dd-mon-yyyy
hh24:mi')
and substr(to_char(s.end_interval_time,'DD-MON-YYYY DY HH24:MI'),13,2)
like '%&hr24_filter.%'
group by s.snap_id
, q.sql_id
, q.plan_hash_value
order by s.snap_id, q.sql_id, q.plan_hash_value;
```

```
SAMPLE_END   SQL_ID    PLAN_HASH_VALUE EXECUTIONS PIO_PER_EXEC
LIO_PER_EXEC  MSEC_EXEC
-------------------- ------------- --------------- ----------
19-JUN-2010 6gvch1xu9ca3g  0    50            .4          906       54.5
19-JUN-2010 6gvch1xu9ca3g  0    50             0          541       42.8
19-JUN-2010 6gvch1xu9ca3g  0    50             0        353.1       49.1

SAMPLE_END   SQL_ID    PLAN_HASH_VALUE EXECUTIONS PIO_PER_EXEC
LIO_PER_EXEC  MSEC_EXEC
-------------------- ------------- --------------- ---------- -------
28-JUN-2014 3hih3b8drup2x  0    30             0        435.6       20.9
28-JUN-2014 3hih3b8drup2x  0    30            .2        434.9       21.2
28-JUN-2014 3hih3b8drup2x  0    30           1.2        823.8       26.3
28-JUN-2014 3hih3b8drup2x  0    30           1.5        369.3       20.5
```

I can now query differences in plans, IO information, etc. and compare mid-year executions in June 2010 vs. June 2014. We can use this information to answer very specific business questions, performance changes or pull execution plans for comparison, as all of the DBA_HIST_XX data exists in the AWR Warehouse. With all of this data from the source, now offloaded to the AWR Warehouse available, you are able to perform full analysis against all history for the database. The AWR Warehouse is designed for advanced reporting vs. previous AWR repository that resided with the production environment and may have impacted production use if advanced analysis was performed on the source database.

Across Multiple Databases on One Host

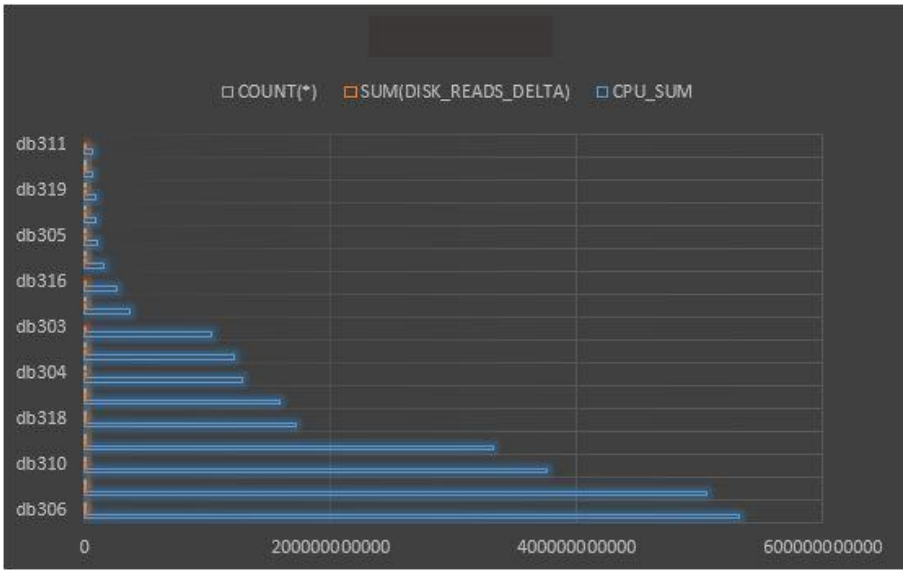
As demonstrated with previous queries, we'll now demonstrate results across more than just one database, but focus on values for an entire host and/or engineered system.

```
select * from (
select
m.target_name,
sum(CPU_TIME_DELTA),
sum(DISK_READS_DELTA),
count(*)
from
DBA_HIST_SQLSTAT a, dba_hist_snapshot s, dba_hist_database_instance
di, dbsnmp.caw_dbid_mapping m
where di.host_name='&host'
and di.dbid in m.new_dbid
and m.new_dbid = a.dbid
and a.snap_id = s.snap_id
and s.begin_interval_time > sysdate -120
group by m.target_name
order by
sum(CPU_TIME_DELTA) desc)


Enter value for host: host687
old   9: where di.host_name='&host'
new   9: where di.host_name='host687'
```

```
TARGET_NAME

        CPU_SUM SUM(DISK_READS_DELTA)    COUNT(*)
---------------------- -------------------- ----------
db306
531404950000              37830984     235318

db308
      505794340000        35797879     190897

db310
375487020000               4531907      91725

db314
331511340000              66608306     692015

db318
171840830000              27431935     310404

db307
159150740000              11246128     116810
```

```
db304
128096640000               55548946     243479

db317
122228340000               17706738     241942

db303
103659670000                3850941      38092

db309
36379820000                 2246820      81611

db316
25804930000                 1818186      68747

db301
16172050000                13471177     250871

db305
9797020000                 17900962      81353

db313
8574870000                 15296613     273022

db319
8480480000                16084847     274796

db302
6411920000                  9206788      89097

db311
5978100000                   359376      10944


17 rows selected.
```

Displayed above are a high level view of CPU usage, disk reads and quantity of executions for the last 120 days across this host for all the databases that reside on it. We could also take this data and create a graph to give a visual view of this data for the business to understand the demands of one database over another:



What Can't I do with the AWR Warehouse?

As the examples above demonstrate, there are very few performance issues that can't be identified with the AWR Warehouse repository. The infinite retention and advanced warehouse features are only limited by the user's vision to answer all the questions of Information Technology about a business' database universe.

Enhancements and new ways of using this invaluable data arrives every day and more and more people are going to embrace AWR Warehouse in the year to come. Build the database, install the AWR Warehouse and start to use it -- The sky's the limit.