AWR

AUTOMATED WORKLOAD REPOSITORY

Oracle 10g provides the AWR. The AWR is a repository of performance information collected by the database to aid in the tuning process for DBAs. The main focus for the Oracle database in version 10g is self-manageability. AWR is the place where the data to aid in self-management is stored.

WHAT DOES AWR DO?

Historically people used **bstat** and **estat** to collect Oracle statistics over a time period and then compare them to each other. The bstat/estat approach was replaced with statspack available in Oracle 8i. Statspack was a package provided by Oracle that did roughly the same thing but better. Statspack has now been surpassed in functionality by AWR which is *always* collecting execution statistics for future analysis and tuning performed by all of the expert components provided by Oracle. Oracle recommends that all statspack users switch to AWR in 10g.

The statistics saved in the AWR provide full trending an analysis. In addition, database up/down events will not lose data in the AWR.

AWR collects data in the following categories:

- Base Statistics - general database performance metrics since instance start-up.
- SQL – statistics for each executed SQL statement (# executions, # physical reads, etc)
- Deltas – the rate of change of important stats over time. Similar to our collection technologies that do before and after snapshots and only show the deltas over the specified period of time.
- Expert Advice – results of the expert analysis engine provided in 10g.

The metrics collected by AWR are easily obtained from v$metricname:

```
Total Wait Counts
Total Time Waited
Number of Sessions Waiting (Event)
Total Wait Counts
Total Time Waited
Database Time Spent Waiting (%)
Average Users Waiting Counts
Host CPU Utilization (%)
Database Time Per Sec
Txns Per Logon
Executions Per Sec
Executions Per Txn
Session Limit %
Process Limit %
PGA Cache Hit %
Shared Pool Free %
Library Cache Miss Ratio
Library Cache Hit Ratio
Row Cache Miss Ratio
Row Cache Hit Ratio
Response Time Per Txn
Database CPU Time Ratio
Database Wait Time Ratio
SQL Service Response Time
User Limit %
Current Open Cursors Count
Current Logons Count
Global Cache Blocks Lost
Global Cache Blocks Corrupted
Global Cache Average Current Get Time
Global Cache Average CR Get Time
GC Current Block Received Per Txn
GC Current Block Received Per Second
GC CR Block Received Per Txn
GC CR Block Received Per Second
PX downgraded to serial Per Sec
PX downgraded Parallel Operation Per
Sec
PX downgraded 75% or more Per Sec
PX downgraded 50% or more Per Sec
PX downgraded 25% or more Per Sec
Branch Node Splits Per Txn
Branch Node Splits Per Sec
Leaf Node Splits Per Txn
Leaf Node Splits Per Sec
User Rollback Undo Records Applied Per
```

```
Txn
User Rollback UndoRec Applied Per Sec
CR Undo Records Applied Per Txn
CR Undo Records Applied Per Sec
CR Blocks Created Per Txn
CR Blocks Created Per Sec
CPU Usage Per Txn
CPU Usage Per Sec
Consistent Read Changes Per Txn
Consistent Read Changes Per Sec
DB Block Changes Per Txn
DB Block Changes Per Sec
Consistent Read Gets Per Txn
Consistent Read Gets Per Sec
DB Block Gets Per Txn
DB Block Gets Per Sec
Enqueue Requests Per Txn
Enqueue Requests Per Sec
Enqueue Deadlocks Per Txn
Enqueue Deadlocks Per Sec
Enqueue Waits Per Txn
Enqueue Waits Per Sec
Enqueue Timeouts Per Txn
Enqueue Timeouts Per Sec
Network Traffic Volume Per Sec
User Calls Ratio
Soft Parse Ratio
Execute Without Parse Ratio
Rows Per Sort
Disk Sort Per Txn
Disk Sort Per Sec
Cursor Cache Hit Ratio
Parse Failure Count Per Txn
Parse Failure Count Per Sec
Hard Parse Count Per Txn
Hard Parse Count Per Sec
Total Parse Count Per Txn
Total Parse Count Per Sec
Total Index Scans Per Txn
Total Index Scans Per Sec
Full Index Scans Per Txn
Full Index Scans Per Sec
Total Table Scans Per Txn
Total Table Scans Per Sec
Long Table Scans Per Txn
Long Table Scans Per Sec
Redo Writes Per Txn
Redo Writes Per Sec
Background Checkpoints Per Sec
DBWR Checkpoints Per Sec
Logical Reads Per Txn
Logical Reads Per Sec
Recursive Calls Per Txn
Recursive Calls Per Sec
User Calls Per Txn
User Calls Per Sec
User Rollbacks Percentage
User Rollbacks Per Sec
User Commits Percentage
User Commits Per Sec
Open Cursors Per Txn
Open Cursors Per Sec
Logons Per Txn
Logons Per Sec
Redo Generated Per Txn
Redo Generated Per Sec
Physical Writes Direct Lobs  Per Txn
Physical Writes Direct Lobs Per Sec
Physical Reads Direct Lobs Per Txn
Physical Reads Direct Lobs Per Sec
Physical Writes Direct Per Txn
Physical Writes Direct Per Sec
Physical Reads Direct Per Txn
Physical Reads Direct Per Sec
Physical Writes Per Txn
Physical Writes Per Sec
Physical Reads Per Txn
Physical Reads Per Sec
User Transaction Per Sec
Redo Allocation Hit Ratio
Memory Sorts Ratio
Buffer Cache Hit Ratio
Database Time Per Sec
Txns Per Logon
Executions Per Sec
Executions Per Txn
Shared Pool Free %
Library Cache Hit Ratio
Database CPU Time Ratio
Consistent Read Changes Per Txn
Consistent Read Changes Per Sec
DB Block Changes Per Txn
DB Block Changes Per Sec
Consistent Read Gets Per Txn
Consistent Read Gets Per Sec
DB Block Gets Per Txn
DB Block Gets Per Sec
Soft Parse Ratio
Execute Without Parse Ratio
Full Index Scans Per Txn
Full Index Scans Per Sec
Total Table Scans Per Txn
Total Table Scans Per Sec
Redo Writes Per Txn
Redo Writes Per Sec
Logical Reads Per Txn
Logical Reads Per Sec
User Calls Per Txn
User Calls Per Sec
Logons Per Txn
Logons Per Sec
Redo Generated Per Txn
Redo Generated Per Sec
Physical Reads Direct Per Txn
Physical Reads Direct Per Sec
Physical Writes Per Txn
Physical Writes Per Sec
Physical Reads Per Txn
Physical Reads Per Sec
User Transaction Per Sec
Memory Sorts Ratio
Buffer Cache Hit Ratio
```

```
Blocked User Session Count
Logical Reads Ratio (Sess/Sys) %
Physical Reads Ratio (Sess/Sys) %
Total Parse Count (Session)
Hard Parse Count (Session)
PGA Memory (Session)
Physical Reads (Session)
CPU Time (Session)
User Transaction Count (Session)
CPU Time Per User Call
Elapsed Time Per User Call
Physical Block Writes (Files-Long)
Physical Block Reads (Files-Long)
Physical Writes (Files-Long)
Physical Reads (Files-Long)
Average File Write Time (Files-Long)
Average File Read Time (Files-Long)
Tablespace Space Usage
```

AWR Installation

AWR is automatically installed and running with 10g. The new MMON process is responsible for collecting data and populating the AWR.

```
$ ps -ef | grep MID101db

  oracle  2134     1  0   Feb 16 ?        0:02 ora_cjq0_MID101db
  oracle  2136     1  0   Feb 16 ?        0:00 ora_d000_MID101db
  oracle  2138     1  0   Feb 16 ?        0:00 ora_s000_MID101db
  oracle  4984     1  1 13:51:52 ?        0:00 ora_q000_MID101db
  oracle  2150     1  0   Feb 16 ?        0:35 ora_mmon_MID101db
  oracle  2152     1  0   Feb 16 ?        0:00 ora_mmnl_MID101db
  oracle  2154     1  0   Feb 16 ?        1:57 ora_j000_MID101db
  oracle  2158     1  0   Feb 16 ?        0:01 ora_j002_MID101db
  oracle  2160     1  0   Feb 16 ?        0:00 ora_j003_MID101db
```

The MMON process takes snapshots of performance data at regular intervals and inserts that data into the AWR tables. The tables containing AWR information are stored in the SYSAUX tablespace (also new in 10G) under the SYS schema. There are a ton of tables in this tablespace – 806 to be exact. However, the AWR related tables all begin with “WR”:

```
WRM$_WR_CONTROL
WRM$_SNAP_ERROR
WRM$_SNAPSHOT
WRM$_DATABASE_INSTANCE
WRM$_BASELINE
WRI$_TRACING_ENABLED
WRI$_SQLSET_STATEMENTS
WRI$_SQLSET_REFERENCES
WRI$_SQLSET_DEFINITIONS
WRI$_SQLSET_BINDS
WRI$_SCH_VOTES
WRI$_SCH_CONTROL
WRI$_OPTSTAT_TAB_HISTORY
WRI$_OPTSTAT_OPR
WRI$_OPTSTAT_IND_HISTORY
WRI$_OPTSTAT_HISTHEAD_HISTORY
WRI$_OPTSTAT_HISTGRM_HISTORY
WRI$_OPTSTAT_AUX_HISTORY
WRI$_DBU_USAGE_SAMPLE
WRI$_DBU_HWM_METADATA
WRI$_DBU_HIGH_WATER_MARK
WRI$_DBU_FEATURE_USAGE
WRI$_DBU_FEATURE_METADATA
WRI$_ALERT_THRESHOLD
WRI$_ALERT_OUTSTANDING
WRI$_ALERT_HISTORY
WRI$_AGGREGATION_ENABLED
WRI$_ADV_USAGE
WRI$_ADV_TASKS
WRI$_ADV_SQLW_TABVOL
WRI$_ADV_SQLW_TABLES
WRI$_ADV_SQLW_SUM
WRI$_ADV_SQLW_STMTS
WRI$_ADV_SQLW_COLVOL
WRI$_ADV_SQLT_STATISTICS
WRI$_ADV_SQLT_RTN_PLAN
WRI$_ADV_SQLT_PLANS
WRI$_ADV_SQLT_BINDS
WRI$_ADV_SQLA_TMP
WRI$_ADV_SQLA_STMTS
WRI$_ADV_SQLA_MAP
WRI$_ADV_SQLA_FAKE_REG
WRI$_ADV_REC_ACTIONS
WRI$_ADV_RECOMMENDATIONS
WRI$_ADV_RATIONALE
WRI$_ADV_PARAMETERS
```

```
WRI$_ADV_OBJECTS
WRI$_ADV_MESSAGE_GROUPS
WRI$_ADV_JOURNAL
WRI$_ADV_FINDINGS
WRI$_ADV_DIRECTIVES
WRI$_ADV_DEF_PARAMETERS
WRI$_ADV_DEFINITIONS
WRI$_ADV_ACTIONS
WRH$_WAITSTAT_BL
WRH$_WAITCLASSMETRIC_HISTORY
WRH$_UNDOSTAT
WRH$_THREAD
WRH$_TEMPSTATXS
WRH$_TEMPFILE
WRH$_TABLESPACE_STAT_BL
WRH$_TABLESPACE_SPACE_USAGE
WRH$_SYS_TIME_MODEL_BL
WRH$_SYSTEM_EVENT_BL
WRH$_SYSSTAT_BL
WRH$_SYSMETRIC_SUMMARY
WRH$_SYSMETRIC_HISTORY
WRH$_STAT_NAME
WRH$_SQL_WORKAREA_HISTOGRAM
WRH$_SQL_SUMMARY
WRH$_SQL_PLAN
WRH$_SQLTEXT
WRH$_SQLSTAT_BL
WRH$_SQLBIND_BL
WRH$_SHARED_POOL_ADVICE
WRH$_SGASTAT_BL
WRH$_SGA
WRH$_SESSMETRIC_HISTORY
WRH$_SERVICE_WAIT_CLASS_BL
WRH$_SERVICE_STAT_BL
WRH$_SERVICE_NAME
WRH$_SEG_STAT_OBJ
WRH$_SEG_STAT_BL
WRH$_ROWCACHE_SUMMARY_BL
WRH$_RESOURCE_LIMIT
WRH$_PGA_TARGET_ADVICE
WRH$_PGASTAT
WRH$_PARAMETER_NAME
WRH$_PARAMETER_BL
WRH$_OSSTAT_NAME
WRH$_OSSTAT_BL
WRH$_OPTIMIZER_ENV
WRH$_MTTR_TARGET_ADVICE
WRH$_METRIC_NAME
WRH$_LOG
WRH$_LIBRARYCACHE
WRH$_LATCH_PARENT_BL
WRH$_LATCH_NAME
WRH$_LATCH_MISSES_SUMMARY_BL
WRH$_LATCH_CHILDREN_BL
WRH$_LATCH_BL
WRH$_JAVA_POOL_ADVICE
WRH$_INSTANCE_RECOVERY
WRH$_FILESTATXS_BL
WRH$_FILEMETRIC_HISTORY
WRH$_EVENT_NAME
WRH$_ENQUEUE_STAT
WRH$_DLM_MISC_BL
WRH$_DB_CACHE_ADVICE_BL
WRH$_DATAFILE
WRH$_CURRENT_BLOCK_SERVER
WRH$_CR_BLOCK_SERVER
WRH$_CLASS_CACHE_TRANSFER_BL
WRH$_BUFFER_POOL_STATISTICS
WRH$_BG_EVENT_SUMMARY
WRH$_ACTIVE_SESSION_HISTORY_BL
```

The third letter of each table name signifies the type of data that it contains.

- I – advisory functions (SQL Advice, Space Advice, etc)
- M – metadata information
- H – historical data

Historical data is already populated in the tables. For instance, a query against WRH$_TABLESPACE_SPACE_USAGE yields 1032 rows even though only 6 tablespaces exist in the database. This is due to data being captured and logged at regular intervals. Any tool would have everything it needs to chart tablespace space usage over time.

Oracle also adds views on top of these base tables. The views all begin with DBA_HIST:

```
DBA_HIST_DATABASE_INSTANCE
DBA_HIST_SNAPSHOT
DBA_HIST_SNAP_ERROR
DBA_HIST_BASELINE
DBA_HIST_WR_CONTROL
DBA_HIST_DATAFILE
DBA_HIST_FILESTATXS
DBA_HIST_TEMPFILE
DBA_HIST_TEMPSTATXS
DBA_HIST_SQLSTAT
DBA_HIST_SQLTEXT
DBA_HIST_SQL_SUMMARY
DBA_HIST_SQL_PLAN
DBA_HIST_SQLBIND
```

```
DBA_HIST_OPTIMIZER_ENV
DBA_HIST_EVENT_NAME
DBA_HIST_SYSTEM_EVENT
DBA_HIST_BG_EVENT_SUMMARY
DBA_HIST_WAITSTAT
DBA_HIST_ENQUEUE_STAT
DBA_HIST_LATCH_NAME
DBA_HIST_LATCH
DBA_HIST_LATCH_CHILDREN
DBA_HIST_LATCH_PARENT
DBA_HIST_LATCH_MISSES_SUMMARY
DBA_HIST_LIBRARYCACHE
DBA_HIST_DB_CACHE_ADVICE
DBA_HIST_BUFFER_POOL_STAT
DBA_HIST_ROWCACHE_SUMMARY
DBA_HIST_SGA
DBA_HIST_SGASTAT
DBA_HIST_PGASTAT
DBA_HIST_RESOURCE_LIMIT
DBA_HIST_SHARED_POOL_ADVICE
DBA_HIST_SQL_WORKAREA_HSTGRM
DBA_HIST_PGA_TARGET_ADVICE
DBA_HIST_INSTANCE_RECOVERY
DBA_HIST_JAVA_POOL_ADVICE
DBA_HIST_THREAD
DBA_HIST_STAT_NAME
DBA_HIST_SYSSTAT
DBA_HIST_SYS_TIME_MODEL
DBA_HIST_OSSTAT_NAME
DBA_HIST_OSSTAT
DBA_HIST_PARAMETER_NAME
DBA_HIST_PARAMETER
DBA_HIST_UNDOSTAT
DBA_HIST_SEG_STAT
DBA_HIST_SEG_STAT_OBJ
DBA_HIST_METRIC_NAME
DBA_HIST_SYSMETRIC_HISTORY
DBA_HIST_SYSMETRIC_SUMMARY
DBA_HIST_SESSMETRIC_HISTORY
DBA_HIST_FILEMETRIC_HISTORY
DBA_HIST_WAITCLASSMET_HISTORY
DBA_HIST_DLM_MISC
DBA_HIST_CR_BLOCK_SERVER
DBA_HIST_CURRENT_BLOCK_SERVER
DBA_HIST_CLASS_CACHE_TRANSFER
DBA_HIST_ACTIVE_SESS_HISTORY
DBA_HIST_TABLESPACE_STAT
DBA_HIST_LOG
DBA_HIST_MTTR_TARGET_ADVICE
DBA_HIST_TBSPC_SPACE_USAGE
DBA_HIST_SERVICE_NAME
DBA_HIST_SERVICE_STAT
DBA_HIST_SERVICE_WAIT_CLASS
```

The frequency of data collection is 30 minutes by default but that can be adjusted. All functionality for driving the workload repository is done via the Oracle supplied package DBMS_WORKLOAD_REPOSITORY. The pacjkage header spec is below:

```
  -- ************************************ --
  --  DBMS_WORKLOAD_REPOSITORY Constants
  -- ************************************ --

  -- Minimum and Maximum values for the
  -- Snapshot Interval Setting (in minutes)
  MIN_INTERVAL    CONSTANT NUMBER := 10;                       /* 10 minutes */
  MAX_INTERVAL    CONSTANT NUMBER := 52560000;                  /* 100 years */

  -- Minimum and Maximum values for the
  -- Snapshot Retention Setting (in minutes)
  MIN_RETENTION   CONSTANT NUMBER := 1440;                          /* 1 day */
  MAX_RETENTION   CONSTANT NUMBER := 52560000;                  /* 100 years */


  -- *********************************** --
  --  DBMS_WORKLOAD_REPOSITORY Routines
  -- *********************************** --

  --
  -- create_snapshot()
  --   Creates a snapshot in the workload repository.
  --
  --   This routine will come in two forms: procedure and function.
  --   The function returns the snap_id for the snapshot just taken.
  --
  -- Input arguments:
  --   flush_level               - flush level for the snapshot:
  --                               either 'TYPICAL' or 'ALL'
  --
  -- Returns:
  --   NUMBER                    - snap_id for snapshot just taken.
  --
```

```
PROCEDURE create_snapshot(flush_level IN VARCHAR2 DEFAULT 'TYPICAL'
                          );

FUNCTION create_snapshot(flush_level IN VARCHAR2 DEFAULT 'TYPICAL'
                         )  RETURN NUMBER;

--
-- drop_snapshot_range()
-- purge the snapshots for the given range of snapshots.
--
-- Input arguments:
--   low_snap_id              - low snapshot id of snapshots to drop
--   high_snap_id             - high snapshot id of snapshots to drop
--   dbid                     - database id (default to local DBID)
--

PROCEDURE drop_snapshot_range(low_snap_id      IN NUMBER,
                              high_snap_id     IN NUMBER,
                              dbid             IN NUMBER DEFAULT NULL
                              );


--
-- modify_snapshot_settings()
-- Procedure to adjust the settings of the snapshot collection.
--
-- Input arguments:
--   retention                - new retention time (in minutes). The
--                              specified value must be in the range:
--                              MIN_RETENTION (1 day) to
--                              MAX_RETENTION (100 years)
--
--                              If ZERO is specified, snapshots will be
--                              retained forever. A large system-defined
--                              value will be used as the retention setting.
--
--                              If NULL is specified, the old value for
--                              retention is preserved.
--
--   interval                 - the interval between each snapshot, in
--                              units of minutes. The specified value
--                              must be in the range:
--                              MIN_INTERVAL (10 minutes) to
--                              MAX_INTERVAL (100 years)
--
--                              If ZERO is specified, automatic and manual
--                              snapshots will be disabled.  A large
--                              system-defined value will be used as the
--                              interval setting.
--
--                              If NULL is specified, the
--                              current value is preserved.
--
--   dbid                     - database identifier for the database to
--                              adjust setting. If NULL is specified, the
--                              local dbid will be used.
--
--  For example, the following statement can be used to set the
--  Retention and Interval to their minimum settings:
--
--    dbms_workload_repository.modify_snapshot_settings
--                (retention => DBMS_WORKLOAD_REPOSITORY.MIN_RETENTION
--                 interval  => DBMS_WORKLOAD_REPOSITORY.MIN_INTERVAL)
--
--  The following statement can be used to set the Retention to
--  7 days and the Interval to 60 minutes:
--
--    dbms_workload_repository.modify_snapshot_settings
--                (retention => 10080, interval  => 60);
--

PROCEDURE modify_snapshot_settings(retention  IN NUMBER DEFAULT NULL,
                                   interval   IN NUMBER DEFAULT NULL,
                                   dbid       IN NUMBER DEFAULT NULL
                                   );
```

```
  --
  -- create_baseline()
  --   Routine to create a baseline.  A baseline is set of
  --   of statistics defined by a (begin, end) pair of snapshots.
  --
  --   This routine will come in two forms: procedure and function.
  --   The function returns the baseline_id for the baseline just created.
  --
  -- Input arguments:
  --   start_snap_id            - start snapshot sequence number for baseline
  --   end_snap_id              - end snapshot sequence number for baseline
  --   baseline_name            - name of baseline (required)
  --   dbid                     - optional dbid, default to Local DBID
  --
  -- Returns:
  --   NUMBER                   - baseline_id for the baseline just created
  --

  PROCEDURE create_baseline(start_snap_id  IN NUMBER,
                            end_snap_id    IN NUMBER,
                            baseline_name  IN VARCHAR2,
                            dbid           IN NUMBER DEFAULT NULL
                            );

  FUNCTION create_baseline(start_snap_id  IN NUMBER,
                           end_snap_id    IN NUMBER,
                           baseline_name  IN VARCHAR2,
                           dbid           IN NUMBER DEFAULT NULL
                           )  RETURN NUMBER;

  --
  -- drop_baseline()
  -- drops a baseline (by name)
  --
  -- Input arguments:
  --   baseline_name            - name of baseline to drop
  --   dbid                     - database id, default to local DBID
  --   cascade                  - if TRUE, the range of snapshots associated
  --                              with the baseline will also be dropped.
  --                              Otherwise, only the baseline is removed.
  --
  PROCEDURE drop_baseline(baseline_name IN VARCHAR2,
                          cascade       IN BOOLEAN DEFAULT false,
                          dbid          IN NUMBER DEFAULT NULL
                          );


  -- ***********************************************************
  --  awr_report_text and _html (FUNCTION)
  --    This is the table function that will display the
  --    AWR report in either text or HTML.  The output will be
  --     one column of VARCHAR2(80) or (150), respectively
  --
  --    The report will take as input the following parameters:
  --      l_dbid     - database identifier
  --      l_inst_num - instance number
  --      l_bid      - Begin Snap Id
  --      l_eid      - End Snapshot Id
  -- ***********************************************************
  FUNCTION awr_report_text(l_dbid     IN NUMBER,
                           l_inst_num IN NUMBER,
                           l_bid      IN NUMBER,
                           l_eid      IN NUMBER,
                           l_options  IN NUMBER DEFAULT 0)
  RETURN awrrpt_text_type_table PIPELINED;

  FUNCTION awr_report_html(l_dbid     IN NUMBER,
                           l_inst_num IN NUMBER,
                           l_bid      IN NUMBER,
                           l_eid      IN NUMBER,
                           l_options  IN NUMBER DEFAULT 0)
  RETURN awrrpt_html_type_table PIPELINED;

END dbms_workload_repository;
```

Manual snapshots are also available by using the *create_snapshot* function within the package. The execution of this package produces the following TKPROF results:

```
OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse      127      0.10       0.10          0          0          0           0
Execute    214      1.93       2.40         43       1510       7022        2213
Fetch      229      0.03       0.04          2        471          0         212
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      570      2.06       2.55         45       1981       7022        2425
```

Now we can see that the total number of statements on an AWR snapshot refresh:

```
      1  session in tracefile.
      7  user  SQL statements in trace file.
    127  internal SQL statements in trace file.
    134  SQL statements in trace file.
    113  unique SQL statements in trace file.
   4057  lines in trace file.
     31  elapsed seconds in trace file.
```

134 SQL statements submitted. Some statements even use the RULE hint - go figure. This is the exact same architecture of DBXray - a package that executes at regular intervals to insert data into tables. I'm not sure why MMON process is needed unless it only does the scheduling.

LIMITATIONS

It can only be assumed that this data capture and storage utility provided by the MMON process utilizes CPU. However, during my tests of manually creating snapshots, I could never get MMON to use more the .5% of the CPU although this is a test database.