UP.TIME DATA MANAGEMENT PACK:

ORACLE 11G TABLE PARTITIONING

TABLE OF CONTENTS

REVISION HISTORY

Rev	Date	Author	Description
1	July. 10, 2009	Joel Pereira	Initial Publication
2	Sep. 24, 2009	Joel Pereira	Added more info on partitions

INTRODUCTION

About up.time

up.time is an On-Line Transaction Processing (OLTP) application; meaning it inserts a huge amount of (small) transactions into the database at a constant and high rate. This can be an intensive process on a database as it is constantly writing new data.

After adding many systems to up.time and having it collect performance data for a lengthy period of time there will be a point where we are no longer interested in data beyond a certain point in time (example: anything older than 6 months, or 1 year, etc). This is where built-in archiving is used to remove data older than the specified amount of time (number of months).

Limitations of Archiving on Very Large Databases (VLDBs)

up.time archiving works by first extracting any old data (older than the archiving policy set) from the database, dumping the data into archive files (zipped XML files, "*.xml.gz") and then going back and removing all the archived data out of the database.

The problem is that this is a very intensive process on the tables that are also being constantly written to with new performance data. This can cause major issues having 2 intense processes both fighting to work on the same large tables in the database.

Issues Caused by Archiving on Very Large Databases (VLDBs)

For most users, the archiving process should work without a problem, but for users with a larger setup (monitoring over 500 systems on one monitoring station) they may experience some of the issues below:

- Performance degradation
- Oracle Temp space filled (during archiving)
- Oracle Undo space filled (during deletion)
- Oracle Redo log performance issues

The Solution: Oracle Table Partitioning

To get around the limitation of having to manage the huge amounts of data directly, Oracle table partitioning allows us to eliminate the archiving process entirely by segmenting the large tables that hold performance data into smaller "partitions". Each “partition” holds one month of data, which allows us to easily and quickly manage older data by swapping it out of the database without having to deal with long wait times and performance degradation of the built-in archiving method.

Selecting the Appropriate Partition Type

With the combination of Range + Reference + Interval table partitioning types in Oracle 11g we can convert the up.time tables to partitioned tables without adding any new columns to any of the tables.

A current limitation of 11g is that Reference Table Partitioning cannot be used with Interval Partitioning, so the "performance_sample" table is the only one that must be manually managed by the DBAs (Interval Partitioning would fully automate the entire partitioning procedure on all tables).

Link: http://download.oracle.com/docs/cd/B28359_01/server.111/b32024/partition.htm

TECHNICAL INFORMATION

Requirements

- Oracle 11g Database Enterprise Edition
- Licensed to use Oracle Table Partitioning

Limitations

- Only works on Oracle 11g (and up)
- **Partitions need to be manually added to the "performance_sample" table on a monthly scheduled basis. If a row is attempted to be inserted into a partition that does not exist you will receive the following error: "ORA-14400: inserted partition key does not map to any partition" error. The row will not be inserted, and any new rows will fail with the same error until a new partition is added.**

Table Partition Types Used

The table below indicates the table partitioning type used for each table in the up.time schema that requires partitioning.

Table Name	Partition Key	Type of Partitioning
performance_sample	sample_time	Range Partitioning
performance_aggregate	Referenced (performance_sample)	Reference Partitioning
performance_cpu	Referenced (performance_sample)	Reference Partitioning
performance_disk	Referenced (performance_sample)	Reference Partitioning
performance_esx3_workload	Referenced (performance_sample)	Reference Partitioning
performance_fscap	Referenced (performance_sample)	Reference Partitioning
performance_lpar_workload	Referenced (performance_sample)	Reference Partitioning
performance_network	Referenced (performance_sample)	Reference Partitioning
performance_nrm	Referenced (performance_sample)	Reference Partitioning
performance_psinfo	Referenced (performance_sample)	Reference Partitioning
performance_vxvol	Referenced (performance_sample)	Reference Partitioning
performance_who	Referenced (performance_sample)	Reference Partitioning
erdc_int_data	sampletime	Interval Partitioning
erdc_decimal_data	sampletime	Interval Partitioning
erdc_string_data	sampletime	Interval Partitioning
ranged_object_value	sample_time	Interval Partitioning

We will recreate the tables listed above with partitions and then perform a mass conversion/copy using the built-in Oracle tools (Dbms_Redefinition). Thanks to the new partitioning features of Oracle 11g we do not need to add any new columns or make any adjustments to application functionality to get partitioning working properly with up.time. After partitioning is complete, accessing the data works the exact same way as before as well as inserting new rows. The only thing that is greatly enhanced is the fact that we'll be able to make modifications (move/drop) to large chunks of data without experiencing any issues on either the database or application side. Changes to the tables are seamless and do not impact anything being done on the application side, so changes can be done while the application is online.

The tables will be setup with monthly partitions all based on the date field in each table. For the list of partition keys for each table please reference the table above.

PRE-SETUP INSTRUCTIONS

Oracle User Permissions

The user that up.time uses to connect to Oracle (in the "<uptime_dir>/uptime.conf" file) must have the following permissions granted to it:

- Connect
- Create Sequence
- Create Session
- Create Table
- Unlimited Tablespace
- DBA

Note: The "DBA" role is a very powerful yet necessary privilege for initially setting up table partitioning. It will be revoked as soon as the table partitioning process is complete. The uptime user should keep most of the other permissions for the application to function properly.

Example commands to create the user and grant the necessary permissions:

```
CREATE USER "UPTIME" PROFILE "DEFAULT" IDENTIFIED BY "PASSWORD" DEFAULT TABLESPACE
"UPTIME" TEMPORARY TABLESPACE "TEMP" ACCOUNT UNLOCK
/
GRANT CREATE SEQUENCE TO "UPTIME"
/
GRANT CREATE SESSION TO "UPTIME"
/
GRANT CREATE TABLE TO "UPTIME"
/
GRANT UNLIMITED TABLESPACE TO "UPTIME"
/
GRANT "CONNECT" TO "UPTIME"
/
GRANT "DBA" TO "UPTIME"
/
```

Stop All up.time Services

After verifying that all checks passed, we can go ahead and stop all the up.time instances and start the conversion process. Any and all up.time data collectors, UI, reporting instances that access the database should be stopped.

How to stop up.time on Windows:

- Start > Run > services.msc
- Locate and stop the service "up.time Data Collector"

How to stop up.time on Linux/Solaris:

- /etc/init.d/uptime_core stop

Running Pre-Conversion Checks

Before executing the commands to convert the tables into partitioned tables, we need to run some checks to verify if everything is setup properly and we are ready to do the conversion. The script “pre_checks.sql” will run some checks to make sure Oracle is ready for the conversion process.

To execute the script, extract all the scripts from the zip file into a location on the Oracle server. For simplicity we will use the following directory for all examples: “/uptime-ora-scripts/”. Also, make sure the “$ORACLE_HOME/bin” directory is in your $PATH so that we can execute the “sqlplus” utility. Please contact your DBA and/or uptime support on how to do this.

Note: You can use any directory you wish; the following is just a recommendation as it avoids certain potential permission issues.

Setting up the directory to contain the scripts :

```
mkdir /uptime-ora-scripts
chmod 777 /uptime-ora-scripts
```

Extract scripts into this directory (/uptime-ora-scripts/)

Help executing SQLPlus from the command line:
sqlplus user/password@hostname/SID
Example:
sqlplus uptime/uptime@localhost/orcl

After connecting to Oracle as the uptime user with sqlplus, run the following command to start the pre checks:

```
Spool on
Set heading off
Spool /uptime-ora-scripts/pre_checks.log
@/uptime-ora-scripts/pre_checks.sql
Spool off
```

You should see the output listing all tables as well as everything “PASSED.” If there is anything that mentions “FAILED”, please contact support immediately with the output to get the issue looked at before proceeding to the next step.

CONVERSION STEPS

Backup the Database

Important Note: Before we proceed to start converting the up.time performance tables to partitioned tables, we HIGHLY recommend taking a full backup of the database. The next few steps will perform large changes to (potentially) very large tables in the database, which means that unexpected events can happen that may corrupt the database. There may also be certain limitations that cause issues while running the conversion that may result in corrupted data. If anything unexpected should happen, we may rely on the database backup to be able to revert back to the previous point. If there is no database backup then it is possible that all historical performance data may be lost if there is an issue.

Pre-Requisite(s)

The most important thing to do before running any of the scripts is to make sure that you have enough free space to perform all the conversion(s).

The scripts will create newly partitioned tables and then copy all the data from the original tables into the new ones, so it's important that there is enough free space in the database for this to work properly. It will run through this process one table at a time so it should be sufficient to have enough free space for a copy of the largest performance table in the database plus a little bit extra. Generally if you have enough free space for a copy of the largest 2 tables in your database it should be more than safe.

If you aren't sure or have any doubts about sizing issues, feel free to contact uptime support (support@uptimesoftware.com) with any questions.

Executing the Conversion Scripts

After creating the database backup, the actual steps to convert to partitioned tables simply involve running one script that will perform the entire conversion. The steps to do so are below:

- Use "sqlplus" to login to the Oracle database.
- Run the following commands to convert all performance tables to partitioned tables:

```
Spool on
set heading off
Spool /uptime-ora-scripts/conversion.log
@/uptime-ora-scripts/run_conversion.sql
Spool off
```

It is important to note that this process may take a lengthy period of time depending on how much data is in the database.

POST CHECKS

Now that the tables have all been converted to partition tables, it is time to run the post-checks to make sure they were converted over properly. We will also revoke the "DBA" role we previously granted for the Oracle user as we no longer need it.

Running Post-Conversion Checks

After connecting to Oracle as the uptime user with sqlplus, run the following command to start the post checks:

```
Spool on
Set heading off
Spool /uptime-ora-scripts/post_checks.log
@/uptime-ora-scripts/post_checks.sql
Spool off
```

You should see the output listing all tables as well as everything "PASSED." If there is anything that mentions "FAILED", please contact support immediately with the output to get the issue looked at.

Adding Indexes

After confirming that all the tables have been properly converted to partitioned tables we can create the extra indexes that uptime will use when querying the tables.

To add the indexes just run the following script:

```
@/uptime-ora-scripts/create_indexes.sql
```

Revoke DBA Role

After the conversion process has been completed successfully, we can go ahead and remove the "DBA" role from the uptime Oracle user. Login to the Oracle database with sqlplus and run the following command to remove the role:

```
REVOKE "DBA" FROM "UPTIME"
/
```

To keep the Oracle database as locked down as possible you can also revoke the following permissions for normal use of the application:

- Create Sequence
- Create Table
- Create Trigger

Note: When installing a Service Pack update you are required to grant at least the "Create Table" permission, but it can be removed once the upgrade is completed. Contact support (support@uptimesoftware.com) with any questions on necessary Oracle permissions for service packs.

MANAGING THE PARTITIONS

Adding a New Partition

After setting up the performance tables as Oracle partition tables some will need to be maintained to make sure that there is always a partition there for the upcoming month so that we do not get an "ORA-14400" error. Due to the partition types we selected for each table, only one table needs to be actively managed; the "*performance_sample*" table.

Format of adding a partition to the "performance_sample" table:

```
ALTER TABLE performance_sample
ADD PARTITION ups_<YEAR>_<MONTH>
VALUES LESS THAN (TO_DATE('1-<MONTH + 1>-<YEAR>', 'DD-MM-YYYY'))
/
```

Example adding a partition to hold data for January 2010:

```
ALTER TABLE performance_sample
ADD PARTITION ups_ 2010_01 VALUES LESS THAN (TO_DATE('1-2-2010', 'DD-MM-YYYY'))
/
```

The name of the partition (ups_2010_01) tells us it contains data for January 2010, and the partition is setup to hold data until February 1, 2010, which is correct.

Moving a Partition

You can use the ALTER TABLE MOVE PARTITION statement to move a partition. For example, a DBA wishes to move older data onto a slower/cheaper disk.

Format:

```
ALTER TABLE aaa MOVE PARTITION bbb
TABLESPACE ccc NOLOGGING
/
```

We recommend moving older data that is not needed for reporting but may be needed for auditing purposes (depending on your corporate auditing policy) onto older/cheaper disks. This allows your SAN to contain only the data needed for generating reports, while older data that may only be needed for auditing to be stored on cheap storage. Then when the older data is no longer needed (beyond your auditing policy) it can be dropped from the database.

Dropping a Partition

You can use the ALTER TABLE DROP PARTITION statement to drop a partition. This will free up the space being used by the older partition back to the database for new data.

Format:

```
ALTER TABLE aaa DROP PARTITION bbb
/
```

For data that is older than your corporate archiving policy, we recommend deleting or backing up on tape. This allows your SAN to contain only the latest data for generating reports, older data that may only be needed for auditing is stored on cheap storage, and anything outside the archiving policy can be deleted or backed up onto tapes.