Configuration Tuning for SAP Sybase ESP Data loading Into SAP HANA
### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVERVIEW</td>
<td>3</td>
</tr>
<tr>
<td>SAP HANA OUTPUT ADAPTER</td>
<td>3</td>
</tr>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Tuning Properties</td>
<td>3</td>
</tr>
<tr>
<td>SAP HANA ODBC DRIVER</td>
<td>4</td>
</tr>
<tr>
<td>SAP HANA DATABASE SERVER</td>
<td>4</td>
</tr>
<tr>
<td>Table Partitioning and Multiple SAP HANA Nodes</td>
<td>4</td>
</tr>
<tr>
<td>Delta Merge</td>
<td>5</td>
</tr>
<tr>
<td>Savepoints</td>
<td>6</td>
</tr>
</tbody>
</table>
OVERVIEW

This document describes the behaviour of the SAP Sybase ESP Output Adapter for SAP HANA and what mechanisms are available to tune performance.

SAP HANA OUTPUT ADAPTER

Overview

The SAP HANA Output Adapter accepts records from the window or stream to which it is attached and passes these records to SAP HANA via the SAP HANA ODBC driver. For high volume scenarios, the adapter is attached to a stream. One adapter instance loads one database table.

Inside the adapter is a queue. Records arriving at the adapter are placed onto this queue. The purpose of the queue is to handle bursts of activity where the database may temporarily not be able to process the records at the same rate at which they are arriving. In ESP 5.1 SP02 and SP03, the size of this queue is unlimited. In ESP 5.1 SP04, the size of this queue can be configured. If the queue size is limited and the queue fills up, then just as with the queues of other operators in ESP, ESP will stop passing new data until the queues begin to clear.

There are multiple loading threads within the SAP HANA Output Adapter, each of which opens an ODBC connection to SAP HANA, prepares records and passes those records to SAP HANA. The number of threads loading SAP HANA is configurable (see the “threadCount” parameter in the next section).

Inserts are collected and sent in batches to SAP HANA. Delete and update operations are performed as single operations on SAP HANA. The insert batch size, the commit frequency, and several other parameters controlling the batch processing are configurable (see the parameters in the next section). The type of operation is indicated by the “opcode” on the record.

Tuning Properties

The following properties are available for tuning the performance of the SAP HANA Output Adapter. The property name as specified in CCL is provided in brackets.

**Bulk Batch Size (bulkBatchSize)**

The number of insert records that are loaded before a commit occurs. This setting controls the trade-off between load efficiency and the latency with which the records become available to queries.

**Bulk Insert Array Size (bulkInsertArraySize)**

The number of insert records that are batched before being passed to SAP HANA. This option must be a divisor of the bulkBatchSize property.

**Idle Buffer Write Delay in Msec (idleBufferWriteDelayMSec)**

If no new data has arrived after this period of time, then any records queued in the adapter’s internal buffers are passed along for loading even though the bulk insert array size has not been reached. This parameter is useful for systems that stop sending data for some period of time (for example, at the end of a business day).

**Buffer Age Limit in Msec (bufferAgeLimitMSec)**

If any record queued in the adapter’s internal buffers reaches this age limit, then the data is passed along for loading even if the specified bulk insert array size has not been reached. This parameter is useful for systems that at any point in the day may be trickling occasional records.
Number of Threads (threadCount)

The number of threads to load SAP HANA. If a value of 0 is set or the property is not specified, N threads are created where N is the number of partitions on the SAP HANA table. Depending on the number of SAP HANA partitions and the capacity of the ESP machine, the default value may not be appropriate. Set the value of this parameter based on the number of SAP HANA partitions and the capacity of the ESP machine.

SAP HANA ODBC DRIVER

The SAP HANA Output Adapter uses the SAP HANA ODBC driver to pass data to SAP HANA. The SAP HANA Output Adapter should be configured to use an ODBC connection that points to the SAP HANA Master node. For multi-node SAP HANA systems, the ODBC driver may connect to multiple SAP HANA nodes in parallel. The SAP HANA ODBC driver contains logic for insert statements that determines on which SAP HANA node the data should reside based on the destination table and the table partition into which the record should be placed. The ODBC driver can also determine, in most cases, the node and partition to which updates and deletes should be sent. There is no need to split data within the ESP project in order to control the connections to partitions or multiple nodes. In order to take advantage of this logic within the ODBC Driver, use Hash or Range partitioning.

SAP HANA DATABASE SERVER

Table Partitioning and Multiple SAP HANA Nodes

For large volume tables, table partitioning will provide performance improvements. For really large tables, partitioning is a must since a single non-partitioned table can contain a maximum of 2 billion rows. With partitioning, each partition can contain 2 billion rows. Using partitioning, a database operation on a table can be processed by all servers that host partitions relevant to the query. Nodes that hold partitions that are not relevant to the operation will not be queried. Partitioning is also beneficial in a single-host system. Operations will be parallelized within the node using multiple threads per table.

Partitioning also allows for query performance improvement through query pruning. Queries are analyzed to determine which partitions hold the data being queried. For example, if a table is partitioned by year, a query restricted to the data of one year is executed only on the partition with data for this year.

There are several partitioning methods available including hash, range, and round-robin. Hierarchical partitioning is also supported (for example, hash and then range).

With hash partitioning, the values of specified columns are used to calculate a hash value, and the record is placed into a partition based on this hash value. If the table has a primary key, only the columns within the key may be used for the hash function. This restriction does not hold for second level partitions in a hierarchical partition. Care must be taken to choose columns that will result in records being evenly distributed across the partitions.

Round-robin partitioning distributes records evenly across partitions in a round-robin fashion.

Range partitioning places records based on value ranges. Usually this requires in-depth knowledge of the distribution of values for the chosen partitioning column. If the table has a primary key, the partitioning columns must be part of that key. This restriction does not hold for second level partitions in a hierarchical partition. Care must be taken to choose columns that will result in records being evenly distributed across the partitions.

The best choice of partitioning method will depend on the nature of the data (for example, how well does it distribute over key values) and the nature of the queries (for example, do the queries need to access all of the partitions or only some of the partitions). For example, if the data would not distribute evenly using hash or range partitioning, then round-robin partitioning may be appropriate. However, this comes at the cost of not being able to exclude any partitions when executing a query (in other words, no query pruning).
Assuming a suitable hash function can be defined to distribute data in a fairly even manner, hash partitioning may provide better query performance than round-robin partitioning since query pruning can occur. Hash partitioning also has the advantage that checking uniqueness of a key is an operation local to the server hosting the partition. Range partitioning requires in-depth knowledge of the data. Applications may choose range partitioning when there is a need to explicitly add or drop partitions. For example, an application may create a partition for an upcoming month so that new data is inserted into that new partition.

For detailed information about partitioning, please see “Table Partitioning in SAP HANA” in the SAP HANA Administration Guide available from help.sap.com.

**Delta Merge**

The delta merge operation is performed if one of the following events occurs:

- The number of lines in delta storage exceeds the specified limit.
- The memory consumption of the delta storage exceeds the specified limit.
- The delta log exceeds the defined limit.

The delta merge can also be triggered explicitly using SQL.

To alter parameters that control when a delta merge occurs, see the section [indexing] in hdbindexserver.ini. It is recommended that delta merge configuration not be changed unless you are instructed to do so by SAP HANA engineering or support.

The delta merge performance is dependent of the size of the main storage. If data is only being modified in some partitions, there are fewer partitions that need to be delta-merged and therefore the performance of the delta merge will be better.

The delta merge may operate in parallel on all available partitions. There are three parameters that control how the threading is handled. To configure the delta merge, use the following parameters of section [indexing] in hdbindexserver.ini:

1) One thread per server (default): set parallel_merge_location to "yes".

2) Configurable amount of threads: set parallel_merge_location to "no" and parallel_merge_part_threads to the maximum number of threads you wish to use.

3) One thread per part: set parallel_merge_location to "no" and parallel_merge_part_threads to "0". Do not use this setting if the table has a large number of partitions.

There is an in-memory version of delta merge that merges in-memory structures but does not persist data to disk. This type of delta merge is not recommended. Because information is not persisted, the delta merge log file can grow quite large making the next persistent delta-merge very lengthy.

The choice of partitioning scheme will affect delta merge performance. A partitioning scheme that minimizes the number of partitions modified by incoming data will improve the overall resource consumption and performance for the delta merge process. However, the choice of partitioning scheme also affects query performance. When choosing a partitioning scheme, a balance must be found between delta merge performance (load performance) and query performance.

**Savepoints**

During the savepoint operation, modified pages are written to disk. Savepoints are written asynchronously. Savepoints are used when restoring a SAP HANA system.

The frequency at which savepoints occur can be configured in the persistence section of the global.ini file (every 5 minutes by default). Savepoints are also triggered automatically by a number of other operations such as data backup, database shutdown, and database restart. You can trigger a savepoint manually by executing the statement "ALTER SYSTEM SAVEPOINT".

For detailed information about savepoints, please see “Backing Up and Recovering the SAP HANA Database” in the SAP HANA Administration Guide available from help.sap.com