SAP HANA ™ Memory Usage Explained

Memory is a critical SAP HANA resource. This paper explains the basic memory concepts and how to explore the memory consumption of a SAP HANA system.

Chaim.Bendelac@sap.com
SAP HANA Development Team

V2, December 2013
Updated for **SAP HANA SPS7 (Rev 70+).**

Table of Contents

Legal Disclaimer

THIS DOCUMENT IS PROVIDED FOR INFORMATION PURPOSES ONLY AND DOES NOT MODIFY THE TERMS OF ANY AGREEMENT. THE CONENT OF THIS DOCUMENT IS SUBJECT TO CHANGE AND NO THIRD PARTY MAY LAY LEGAL CLAIM TO THE CONTENT OF THIS DOCUMENT. IT IS CLASSIFIED AS “CUSTOMER” AND MAY ONLY BE SHARED WITH A THIRD PARTY IN VIEW OF AN ALREADY EXISTING OR FUTURE BUSINESS CONNECTION WITH SAP. IF THERE IS NO SUCH BUSINESS CONNECTION IN PLACE OR INTENDED AND YOU HAVE RECEIVED THIS DOCUMENT, WE STRONGLY REQUEST THAT YOU KEEP THE CONTENTS CONFIDENTIAL AND DELETE AND DESTROY ANY ELECTRONIC OR PAPER COPIES OF THIS DOCUMENT. THIS DOCUMENT SHALL NOT BE FORWARDED TO ANY OTHER PARTY THAN THE ORIGINALLY PROJECTED ADDRESSEE.

This document outlines our general product direction and should not be relied on in making a purchase decision. This document is not subject to your license agreement or any other agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or to develop or release any functionality mentioned in this document. This document and SAP's strategy and possible future developments are subject to change and may be changed by SAP at any time for any reason without notice. This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP assumes no responsibility for errors or omissions in this document and shall have no liability for damages of any kind that may result from the use of these materials, except if such damages were caused by SAP intentionally or grossly negligent.

© Copyright 2013 SAP AG. All rights reserved.

1 Introduction

SAP HANA

SAP HANA™ is a leading in-memory database and data management platform, specifically developed to take full advantage of the capabilities provided by modern hardware to increase application performance. By keeping all relevant data in main memory (RAM), data processing operations are significantly accelerated.

The key performance indicators of SAP HANA appeal to many of our customers, and thousands of deployments are in progress. SAP HANA has become the fastest growing product in SAP's history.

About this Document

A fundamental SAP HANA resource is memory. Understanding how the SAP HANA system requests, uses and manages this resource is crucial to the understanding of SAP HANA. SAP HANA provides a variety of memory usage indicators, to allow monitoring, tracking and alerting. This paper explores the key concepts of SAP HANA memory utilization, and shows how to understand the various memory indicators.

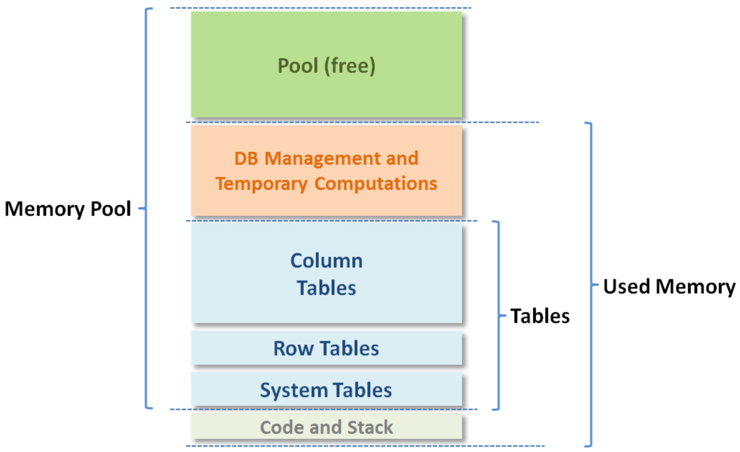
2 What is Memory used for?

SAP HANA consists of a number of processes running on the SUSE Linux operation environment. Under Linux, the operating system is responsible for the reservation of memory to all processes. When SAP HANA starts up, the OS reserves memory for the program code (sometimes called the *text),* the program stack, and static data. It then dynamically reserves additional data memory[1] upon requests from the SAP HANA memory manager.

As an in-memory database, it is particularly critical for SAP HANA to carefully manage and track its consumption of memory. For this purpose, SAP HANA manages its own data memory pool by requesting memory from the OS, possibly in advance of using it. The memory pool is used to store all the in-memory data and system tables, thread stacks, temporary computations and all the other data structures required for managing a database.

At any given point, only parts of the memory pool are really in use. SAP refers to the *total amount of memory actually in use* as the SAP HANA **Used Memory**. This is the most precise indicator of the amount of memory that SAP HANA uses.

The SAP HANA Used Memory, consisting of code, stack and data, is shown below:



Since the code and program stack size are about 6 GB, almost all of the Used Memory is in fact used for storing tables, computations and database management.

It is possible to *partition* large tables, and even distribute these partitions over multiple hosts. In such distributed scenarios, this discussion of memory usage applies to each of the hosts of an SAP HANA system, separately.

1 Dynamically allocated memory consists of *heap* memory and *shared* memory.

Memory Sizing

Memory *sizing* is the process of estimating, in advance, the amount of memory that will be required to run a certain workload on SAP HANA. To understand memory sizing, you will need to answer the following questions:

1. **What is the size of the data tables that will be stored in SAP HANA?**
   You may be able to estimate this based on the size of your existing data, but unless you precisely know the compression ratio of the existing data and the anticipated growth factor, this estimate may only be partially meaningful.
2. **What is the expected compression ratio that SAP HANA will apply to these tables?**
   The SAP HANA Column Store automatically uses a combination of various advanced compression algorithms (dictionary, LRE, sparse, and more) to best compress each table column separately. The achieved compression ratio depends on many factors, such as the nature of the data, its organization and data-types, the presence of repeated values, the number of indexes (SAP HANA requires fewer indexes), and more.
3. **How much extra working memory will be required for DB operations and temporary computations?**
   The amount of extra memory will somewhat depend on the size of the tables (larger tables will create larger intermediate result-tables in operations like joins), but even more on the expected work load in terms of the number of users and the concurrency and complexity of the analytical queries (each query needs its own workspace).

SAP HANA provides some control over compression, via the "optimize_compresssion" configuration section of the index server. More details on data compression can be found in "How to Calculate Data Compression Rates in SAP HANA" (https://cw.sdn.sap.com/cw/docs/DOC-145836).

SAP Notes 1514966, 1637145 and 1736976 provide additional tools and information to help you size the required amount of memory, but the most accurate method is ultimately to import several representative tables into a SAP HANA system, *measure* the memory requirements, and extrapolate from the results.

The next section will explain how to measure and understand SAP HANA Used Memory. This description corresponds to SAP HANA revision 70 (SAP HANA 1.0 SPS7) or later.

3 Used Memory

As explained above, **Used Memory** is the *total amount of memory currently in use* by SAP HANA. This is the most precise indicator of the amount of memory that SAP HANA requires at any time.

To display the current size of the Used Memory, you can use the following simple SQL statement (for example with the SAP HANA Studio SQL editor):

```
select HOST, round(INSTANCE_TOTAL_MEMORY_USED_SIZE/1024/1024/1024, 2) as "Used Memory GB"
from M_HOST_RESOURCE_UTILIZATION
```

This value provides a single summary result per host.

A SAP HANA system consist of multiple services that all consume some memory. Of particular interest is the "indexserver" service, the main database service. The indexserver holds all the data tables and temporary results, and therefore dominates the SAP HANA Used Memory.

You can drill down into Used Memory, and obtain the amount of indexserver Used Memory as follows[2]:

```
select HOST, round(TOTAL_MEMORY_USED_SIZE/1024/1024/1024, 2) as "Used Memory GB"
from M_SERVICE_MEMORY where SERVICE_NAME = 'indexserver'
```

This will list all the hosts in the system, and the amount of indexserver Used Memory per host. Similarly, the memory consumption of other services (like the xsengine service) can be examined.

Used Memory over time

The value of Used Memory represents a current measurement, but it is ultimately more important to understand the behavior of Used Memory over time and under peak loads.

A snapshot copy of indexserver Used Memory is periodically saved in the HOST_SERVICE_MEMORY system table, providing you with a very useful time-series.

For instance, to see the ***peak*** amount of indexserver Used Memory since the server was restarted:

```
select top 1 HOST, SERVER_TIMESTAMP, round(TOTAL_MEMORY_USED_SIZE/1024/1024/1024, 2) as
"Used Memory GB" from _SYS_STATISTICS.HOST_SERVICE_MEMORY
where SERVICE_NAME = 'indexserver' order by TOTAL_MEMORY_USED_SIZE desc
```

In a multi-host system, you can filter the query by host (`where HOST = 'myhost1' and …`).

You can use the time series to create utilization graphs over time, or to drill down into a particular period in the past. For instance, the following query shows the value of indexserver Used Memory at 7:00 AM during each of the last 30 days[3]:

```
select top 30 HOST, SERVER_TIMESTAMP, round(TOTAL_MEMORY_USED_SIZE/1024/1024/1024, 2) as
"Used Memory GB" from _SYS_STATISTICS.HOST_SERVICE_MEMORY where SERVICE_NAME = 'indexserver' and
hour(SERVER_TIMESTAMP) = 7 and minute(SERVER_TIMESTAMP) = 0 order by SERVER_TIMESTAMP desc
```

Again, you can filter by host as appropriate.

Similarly, you can use the HOST_SERVICE_MEMORY time series to examine memory usage yesterday between 9AM and 10:30 AM, or find the peak-of-the day in the last week, etc.

[2] Note that the amount of overall Used Memory is not simply the sum of used memory by all the SAP HANA services, as will be explained in section 6.

[3] By default, SAP HANA keeps history on only the last 30 days. To change the default settings, use the "new" SPS7 statistics service, see SAP Note 1917938.

© 2013 SAP AG

4 SAP HANA Tables

The dominant part of the SAP HANA Used Memory is evidently the space taken up by the data tables. SAP HANA provides separate measurements for Column Store tables and Row Store tables. These measurements are discussed below.

Column Tables

The following simple query provides a high-level overview of the amount of memory used for column tables:

```
select round(sum(MEMORY_SIZE_IN_TOTAL)/1024/1024) as "Column Tables MB Used" from M_CS_TABLES
```

Or, providing per-schema details:

```
select SCHEMA_NAME as "Schema", round(sum(MEMORY_SIZE_IN_TOTAL) /1024/1024) as "MB Used" from
M_CS_TABLES GROUP by SCHEMA_NAME order by "MB Used" desc
```

Note that SAP HANA loads column tables into memory column-by-column only upon use. This is sometimes called "lazy loading". Hence, columns that are never used will not be loaded, which avoids memory waste. When SAP HANA runs out of allocatable memory, it will try to unload unimportant data (such as caches) and even least recently used table columns to free up some memory.

Thus, if it is important to precisely measure the total, or worst-case, amount of memory used for a particular table, it is best to ensure that the table is first loaded in its entirety, by executing the following SQL statement:

```
load TABLE_NAME all
```

You can use the following technique to examine the amount of memory consumed by a specific table. This also shows which of its columns are loaded, and the compression ratio that was accomplished. For example, list all tables for schema "SYSTEM":

```
select TABLE_NAME as "Table", round(MEMORY_SIZE_IN_TOTAL/1024/1024) as "MB Used" from M_CS_TABLES where
SCHEMA_NAME = 'SYSTEM' order by "MB Used" desc
```

Or drill down into columns of a single table, for instance the table "LineItem", to view the actual size of the data, the "delta changes" and the compression ratio for each of its columns.

```
select COLUMN_NAME as "Column", LOADED, round(UNCOMPRESSED_SIZE/1024/1024) as "Uncompressed MB",
round(MEMORY_SIZE_IN_MAIN/1024/1024) as "Main MB", round(MEMORY_SIZE_IN_DELTA/1024/1024) as "Delta MB",
round(MEMORY_SIZE_IN_TOTAL/1024/1024) as "Total Used MB", round(COMPRESSION_RATIO_IN_PERCENTAGE/100, 2)
as "Compr. Ratio" from M_CS_COLUMNS where TABLE_NAME = 'LineItem'
```

In fact, the M_CS_TABLES and M_CS_COLUMNS system views contain much more information (such as cardinality, main-storage vs. delta storage and more). Another useful system view is M_CS_ALL_COLUMNS, which also exposes the internal columns (such as index columns). See the "SAP HANA SQL and System Views Reference" for more information.

Row Tables

Some system tables are in fact row store tables. To get a sense of the total amount of memory used for these row tables, you can use the following query:

```
select round(sum(USED_FIXED_PART_SIZE + USED_VARIABLE_PART_SIZE)/1024/1024) as "Row Tables MB Used"
from M_RS_TABLES
```

To examine the memory consumption of row tables of a particular schema, for instance the schema 'SYS', drill down as follows:

```
select SCHEMA_NAME, TABLE_NAME, round((USED_FIXED_PART_SIZE + USED_VARIABLE_PART_SIZE)/1024/1024, 2) as
"MB Used" from M_RS_TABLES where SCHEMA_NAME = 'SYS' order by "MB Used" desc, TABLE_NAME
```

5 Managing the Memory Pool

As mentioned, SAP HANA pre-allocates and manages its own memory pool, used for storing in-memory tables, for thread stacks, and for temporary results and other system data structures.

When more memory is required for table growth or temporary computations, the SAP HANA memory manager obtains it from the pool. When the pool cannot satisfy the request, the memory manager will increase the pool size by requesting more memory from the operating system, up to a predefined **Allocation Limit**.

By default, the allocation limit is set to 90% of the first 64 GB of physical memory on the host plus 97% of each further GB. You can see the allocation limit on the Overview tab of the Administration perspective of the SAP HANA studio, or view it with SQL:

```
select HOST, round(ALLOCATION_LIMIT/1024/1024/1024, 2) as "Allocation Limit GB"
from PUBLIC.M_HOST_RESOURCE_UTILIZATION
```

There is normally no reason to change this value, except in the case when you purchased a license for less than the total of the physical memory. In such a case, you should set the global allocation limit accordingly, to remain in compliance with the license[4]. Another case when you may want to limit the size of the memory pool is on *development systems*, where more than one SAP HANA system may be installed on a single host, to avoid resource contentions or conflicts.

Memory is a finite resource. Once the allocation limit has been reached and the pool exhausted, the SAP HANA memory manager will no longer be able to allocate memory for internal operations without first giving up something else. In fact, this is exactly what will happen: buffers and caches will be released and column store tables will be unloaded, column by column, based on a least-recently-used order, up to a preset lower-limit. When tables are partitioned over several hosts, this is managed per host; that is, column partitions will be unloaded only on hosts with acute memory shortage.

Table (column or partition) unloading is generally not a good situation, since it leads to performance degradation later, when the data will have to be reloaded later for queries that need them. You can identify pool exhaustion by examining the M_CS_UNLOADS system view. For instance, the following query will provide the number of unloads during a particular one-hour time-slot:

```
select count(*) from M_CS_UNLOADS
where UNLOAD_TIME between '19.08.2013 09:00:00' and '19.08.2013 10:00:00'
```

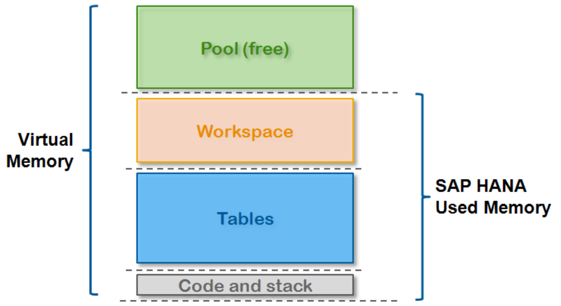
Despite all these abilities, it is theoretically possible that the memory manager will still face a need for more memory that it cannot satisfy. For instance, when too many concurrent transactions use up all memory, or when a particularly complex query performs a cross-join on very large tables, creating a huge intermediate result that exceed the available memory. Such situations can even lead to an Out Of Memory failure.

[4] Please also refer to SAP Note **1704499**, for more information on memory usage auditing related to SAP HANA licenses.

© 2013 SAP AG

6 The Operating Environment

From the Linux operating system perspective, SAP HANA is a collection of separate processes. Linux processes reserve memory for their use by requesting an allocation from the Linux operating system. The entire reserved memory footprint of a program is referred to as its **Virtual Memory**. Each Linux process has its own virtual memory, which grows when the process requests more memory from the operating system, and shrinks when the process relinquishes memory. You can think of virtual memory size as the maximal amount that the process has been allocated, including reservations for its code, stack, data, and memory pools under program control. SAP HANA's virtual memory is logically[5] shown below:



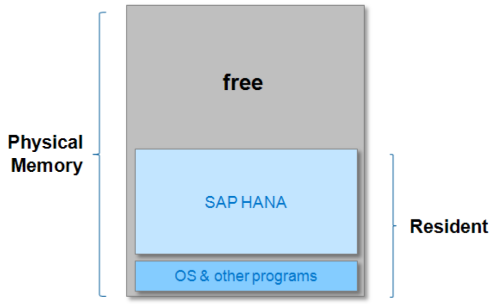
Virtual, Physical and Resident Memory

When (part of) the virtually allocated memory actually needs to be *used*, it is loaded or mapped to the real, physical memory of the host, and becomes "resident". Physical memory is the DRAM memory installed on the host. On most SAP HANA hosts, it ranges from 256 Gigabytes (GB) to multiple Terabytes (TB). It is used to run the Linux operating system, SAP HANA, and all other programs.

Resident memory is the physical memory actually in operational use by a process.

Over time, the operating system may "swap out" some of a process' resident memory, according to a least-recently-used algorithm, to make room for other code or data. Thus, a process' resident memory size may fluctuate independently of its virtual memory size. In a properly sized SAP HANA appliance there is enough physical memory, and thus swapping is disabled and should not be observed.

See the following illustration:



On a typical SAP HANA appliance, the resident part of the OS and all other running programs usually does not exceed 2GB. The rest of the memory is thus dedicated for the use of SAP HANA.

To display the size of the Physical Memory and Resident part, you can use the following SQL command:

```
select HOST, round((USED_PHYSICAL_MEMORY + FREE_PHYSICAL_MEMORY)/1024/1024/1024, 2) as
"Physical Memory GB", round(USED_PHYSICAL_MEMORY/1024/1024/1024, 2) as "Resident GB"
from PUBLIC.M_HOST_RESOURCE_UTILIZATION
```

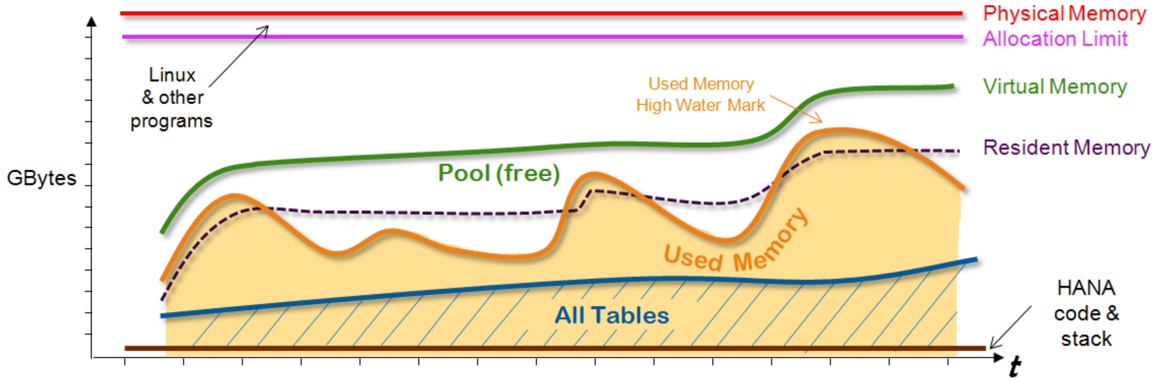
5 SAP HANA really consists of several separate processes, so this figure shows the combination of all SAP HANA processes

© 2013 SAP AG

When memory is required for table growth or for temporary computations, the SAP HANA code obtains it from the existing memory pool. When the pool cannot satisfy the request, the HANA memory manager will request and reserve more memory from the operating system. At this point, the virtual memory size of the HANA processes grows.

Once a temporary computation completes or a table is dropped, the freed memory is returned to the memory manager, who recycles it to its pool, usually without informing Linux[6]. Thus, from SAP HANA's perspective, the amount of *Used Memory* shrinks, but the process' virtual and resident sizes are not affected. This creates a situation where the Used Memory may even shrink to *below* the size of SAP HANA's resident memory, which is perfectly normal.

The following illustration shows the relationship between physical memory, Linux virtual and resident memory, and SAP HANA's pool and Used Memory indicators. Note how changes in Used Memory do not affect the processes' virtual and resident sizes.



Linux Indicators

To view the amount of physical memory on the host, and the resident part, use:

```
free -g | awk '/Mem:/ {print "Physical Memory: " $2 " GB."} /cache:/ {print "Resident: " $3 " GB."}'
```

This merely provides the host-perspective. The Linux process indicators are of course more relevant. Since the operating system treats SAP HANA as a collection of processes, we need to consider both the virtual and resident part of these processes. A process' virtual memory size is always larger than its resident size. Note also, that due to the managed memory pool, both the SAP HANA virtual size and resident sizes may appear larger than what the Used Memory indicator would lead you to expect. This is entirely normal.

Linux maintains High Water Mark (peak) indicators for the virtual and resident process sizes. In a stable system, the current virtual and resident sizes will be only slightly smaller than their respective high water marks because SAP HANA grows its pool, but does not normally relinquish unused memory. Thus, the resident size high water mark should generally track the peak Used Memory. Very large differences may indicate that parts of the SAP HANA memory pool were freed, possibly due to insufficient physical memory.

Display these process memory indicators for SAP HANA as follows (*replace "qp4adm" in this example with the username of the appropriate SAP HANA administrator*):

```
cat `ps h -U qp4adm -o "/proc/%p/status" | tr -d ' '` | awk '/VmSize/ {v+=$2} /VmPeak/ {vp+=$2} /VmRSS/ {r+=$2;} /VmHWM/ {rp+=$2} END {printf("Virtual Size = %.2f GB (peak = %.2f), Resident size = %.2f GB (peak = %.2f)\n", v/1024/1024, vp/1024/1024, r/1024/1024, rp/1024/1024)}'
```

[6] The memory manager may also choose to return memory back to the operating system, for instance when the pool is close to the allocation limit, and contains large unused parts.

© 2013 SAP AG

Process memory reports on Linux are difficult to interpret for various reasons. One of the main reasons is the way processes can share or re-use memory allocated to another process. Linux normally allocates *separate* memory areas to different processes, of course. There are two main exceptions to this rule.

The first is program code. When several processes use the same code libraries, there will only be *one* copy of that code in physical memory, re-used by all the processes.

This partially explains why it is not simple to determine the "size" of SAP HANA, whose processes highly re-use the same code. If we look at an individual process, its memory footprint will include the size of its entire code area (about 6GB for the index-server process), but if we naively sum the sizes of two processes, we *over-count* the re-used code areas. From SAP HANA SPS6, the formula used to calculate SAP HANA Used Memory incorporates a more sophisticated calculation of reused code areas.

The second exception is a technique called "shared memory". Shared memory, as its name implies, is used to "share" data between processes. Both processes define the same memory area as "shared", and they can then exchange information simply by writing into it. This (used to be, and still is somewhat) faster than the alternative of sending network or pipe-based messages between processes.

It is difficult to account for shared memory. Does it belong to one process? Both? Neither? If we naively sum the memory belonging to multiple processes, we grossly "over-count".

Linux reports shared memory inconsistently. When you ask it to report the resident memory map of a single process, it will report the shared-memory as part of its memory footprint. However, when you ask "how much *total memory* is resident", it does *not* account for shared memory.

This is usually not significant, because very few programs use large blocks of shared memory. However, SAP HANA is different. An early SAP HANA design decision was to use "shared memory" for row-store tables[7]. As a result, when you use large row-store tables, the shared-memory footprint of SAP HANA can become very large.

Thus, if you ask Linux to report the resident size of SAP HANA and the *total* resident size on the host, the size of SAP HANA may appear to exceed the total, which of course makes no sense.

To compensate, SAP HANA adds the resident size of the shared-memory part of the SAP HANA processes to the resident size reported by Linux[8]. This is another intentional reason why SAP HANA may report different values than Linux.

---

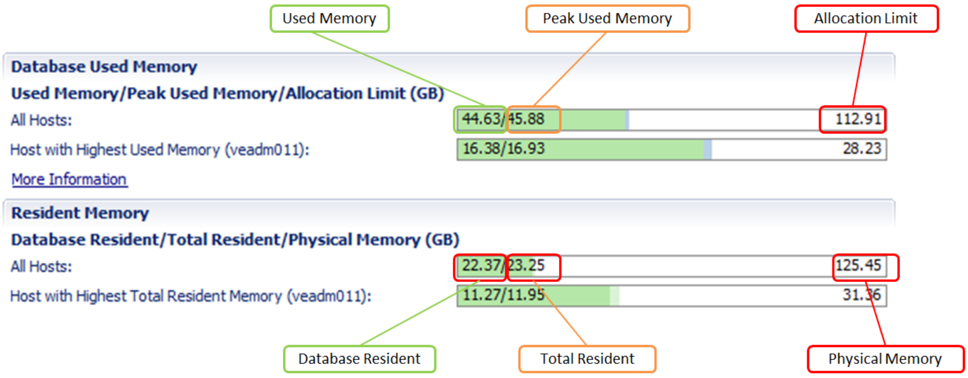
7 Incidentally, this memory is not really shared between processes; it is just defined as "shared memory".

8 This is not precise - it slightly over-counts really-shared memory and swapped-out shared memory - but is much more accurate than reporting the obviously wrong Linux value.

© 2013 SAP AG

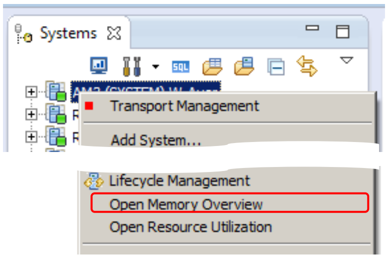
7 SAP HANA Studio

You can view some of the most important memory indicators on the Overview tab of the SAP HANA studio administrative perspective:

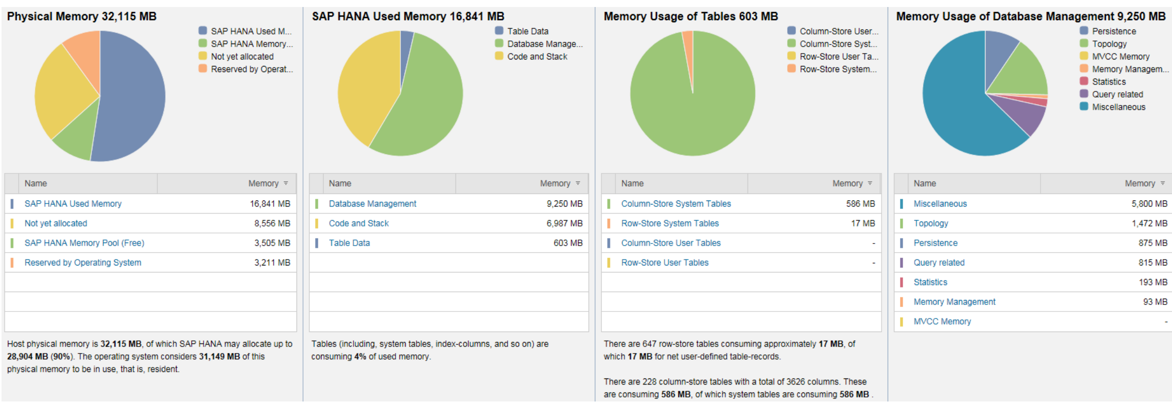


Note that the above illustration shows the memory indicators of a distributed system; the first line shows the sum over all hosts, the second line shows the status of the host with the highest Used Memory.

For even *more* details, check out the new **Memory Overview** feature of the SAP HANA studio. To access it, right click on a system in the Systems View, and select "Open Memory Overview" in the context menu, as follows:



This will open the Memory Overview[9], which looks as follows:



On a distributed system with multiple hosts, this chart will have tabs, allowing you to review the memory usage of each of the hosts.

The first pie chart describes a view of the host's Physical Memory, as divided into Used Memory (blue) and other parts (free pool, reserved, etc.). The second pie zooms into Used Memory, and shows the division of

---

[9] To view the Memory Overview, you need **Monitoring** privileges. E.g. use the following SQL statement (replace 'youruser' with the actual user name): `call GRANT_ACTIVATED_ROLE('sap.hana.admin.roles::Monitoring','youruser')`

table data vs. other parts (database management, code/stack). The third pie again zooms in to focus on the data tables, showing the breakdown between column tables and row tables, for instance. The fourth pie zooms into the data-management part of the second pie chart, providing an insight into the database internals.

The Memory Overview chart is interactive. Click on one of the pies, and you can see usage percentages. Click on the labels in the tables below the pies to see the SQL statement that was used to calculate the displayed values.

Comments at the bottom provide additional information.

8 In Summary

SAP HANA maintains many system views and memory indicators, to provide a precise way to monitor and understand the SAP HANA memory utilization. The most important of these indicators is **Used Memory** and the corresponding historic snapshots. In turn, it is possible to drill down into very detailed reports of memory utilization using additional system views, or by using the convenient Memory Overview from the SAP HANA studio.

Since SAP HANA contains its own memory manager and memory pool, external indicators, like the host-level Resident Memory size, or the process-level virtual and resident memory sizes, can be misleading when estimating the real memory requirements of a SAP HANA deployment.

© 2013 SAP AG