SAP HANA ™ Scalability

Design for scalability is a core SAP HANA principle. This paper explores the principles of SAP HANA's scalability, and its support for the increasing demands of data-intensive workloads.

Chaim.Bendelac@sap.com
SAP HANA Development Team

V1.1a, July 2012

Table of Contents

Legal Disclaimer

1 Introduction

1.1 SAP HANA

SAP HANA™ is an innovative 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. Research showed that in-memory data-stores show performance improvements of up to a factor of 1000 in certain analytical scenarios [Plattner 2009].

SAP HANA database systems can be distributed across multiple servers to achieve scalability in terms of both data volume and user concurrency.

The key performance indicators of SAP HANA appeal to many of our customers, and hundreds of deployments are in progress. SAP HANA has become the fastest growing product in SAP's 40-year history.

1.2 About this Document

One of the most fundamental design principles of SAP HANA is its design for scalability. Adherence to this design principle ensures that SAP HANA's performance will continue to leverage changes in hardware technology on the one hand, and keep up with the ever increasing data-set volumes and real-time demands of data-intensive applications on the other.

This paper explores the SAP HANA design as it relates to scalability and performance. It describes how SAP HANA's advanced algorithms meet the application scalability goals across a range of hardware and demand options.

2 Industry Trends

In the past, database management systems were designed and optimized to perform well on hardware with limited RAM, and with slow disk I/O as the main bottleneck. As the demand for real-time analytics on growing data volumes continued to outpace the ability of these databases, specialized database systems (often referred to as a data warehouse) for analytic processing on pre-aggregated data were introduced, at a cost of duplicate data-sets, and an expensive Extract-Transform-Load process between them.

In recent years, several related hardware trends have converged to reach a true inflection point, which is now causing a dramatic paradigm shift in database design, resulting in the development of new in-memory databases that are setting new standards in data management performance.

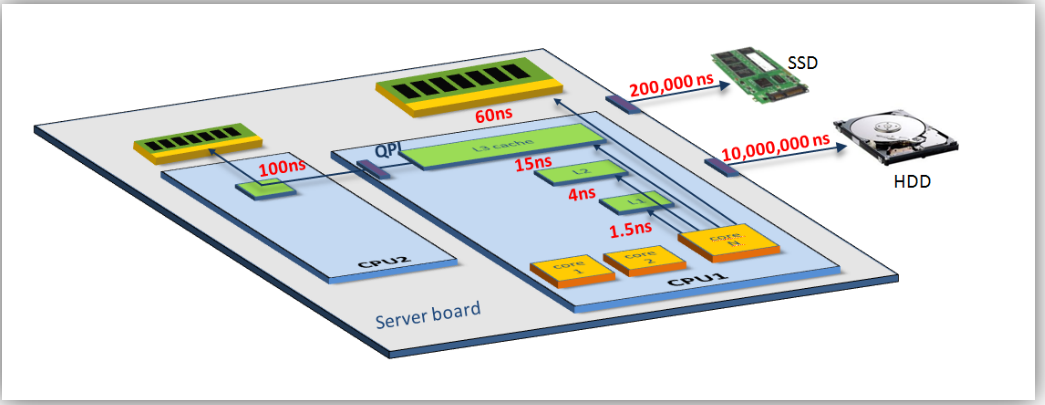
These trends are discussed here.

2.1 Micro-Architecture Trends

At the micro-architecture level, we have seen a gradual increase in computing power, no longer just by processor clock speed-up and through circuit miniaturization, but through a growing trend of parallelization. If ten years ago an integrated circuit might have one or two main computing cores, today's processors from leading vendors like Intel may include as many as ten computing cores per processor, and this number continues to grow. In fact, using a technique called hyper-threading, each core can efficiently serve two computations at once, raising the bar even further by effectively having up to 20 logical cores per processor.

A second critical micro-architecture innovation is the extension of the processor address-space. By moving to an instruction-set that supports a 64-bit address space, modern processors can efficiently address millions of terabytes of RAM, driving server designs with larger and larger on-board memory arrays.

A related trend is the ability to include much larger "cache memory" on the same processor chip. Access to data in this ultra-fast on-chip store is *much faster* than accessing conventional RAM memory. The following illustration shows the dramatic differences in access times from the on-processor caches, a few nanoseconds, compared to RAM (up to 100 nanoseconds), and further compared to traditional storage like hard drives (up to 10 million nanoseconds).



A third micro-architecture innovation is the way that processors are inter-connected. On-chip cache-coherency support and ultra-fast interconnects enable board-designs with 2, 4 and even eight processors on a single server-board, which can be treated by application software as a single computer with up to 160 logical cores!

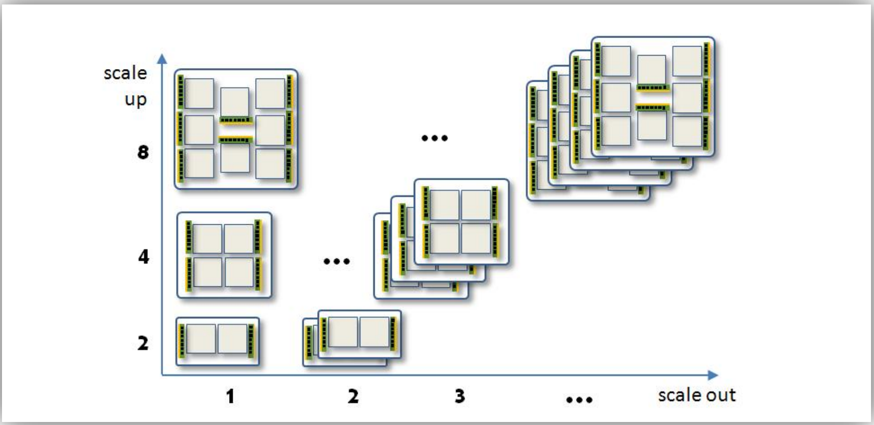
2.2 System Design

While micro-architecture innovation has enabled the construction of high-performing multi-core computers, it is the dramatic change in chip density and cost that really translated this innovation into the computer systems of today. What was recently unthinkable is now common-place: affordable, Intel-processor based servers equipped with a terabyte of RAM. And these changes will continue along a path that has become well-known as Moore's law, to yield servers with 16 terabytes and hundreds of cores in a few years.

The impact of these changes has been very deep. Databases used to run on what were known as mainframe computers, or "enterprise-class" servers. These super-expensive custom-made machines were carefully crafted to optimize rare and expensive processors, high-end storage arrays and specially-designed communication busses. Only a few years later, these expensive systems are being replaced with inexpensive, standard and modular servers, available from a variety of vendors, that can easily be deployed in data centers or in the cloud.

2.3 System Scale

The ability to build single-board computers of increasing processing power, with increased integration and performance is often referred to as a *scale-up* approach. Another approach, combining multiple independent computers in the form of a loosely-coupled networked cluster, is referred to as a *scale-out* approach. This is illustrated by the following picture:



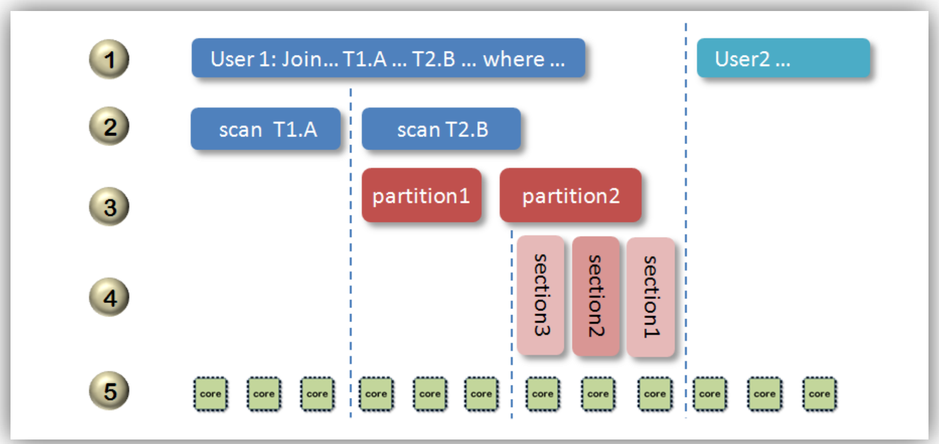
Combining both scale-up and scale-out in a single system architecture achieves huge scalability and flexibility, supporting databases with tens of terabytes of data, and various price/performance demand curves.

3 Parallelism in SAP HANA

As we have seen, the key to high scalability is the ability to efficiently utilize the computing resources of modern computers systems, namely the huge RAM array, the processor-based cache-memory, and mainly, the large and increasing number of compute cores.

This is accomplished primarily by using a columnar data organization, and by massively parallelizing the workload. A columnar data organization offers many advantages, such as higher compression rates, higher performance for column operations (like scans and aggregations), and, in particular, the ability to easily parallelize these operations.

SAP HANA takes advantage of parallelism at every level of the workload hierarchy, as is illustrated by the following figure:



1. At the connection or session level, different users can concurrently submit queries that run independent of each other and use different compute resources.
2. The SAP HANA optimizer constructs an execution plan by decomposing the query into *plan operators* (POPs), such as "aggregate", "sort", and "scan column", and executes these POPs in parallel as much as possible. For instance, scanning (find records that match a condition), aggregating or sorting different columns can be performed independently and very efficiently, due to the columnar organization of tables in the SAP HANA database.
3. Each POP in turn may be parallelized, for instance by running on distributed data partitions that may have been placed on separate servers that form a distributed SAP HANA database system.
4. In addition, a table or column may be treated as multiple "vertical" sections for the purpose of parallel execution. Each section can be scanned or aggregated in parallel in a different thread on a different compute core. For instance, a table with 64 million rows could be broken into 32 or even 192 sections, depending on available compute resources. The calculated results are combined at the completion of all the execution threads.
5. At the processor level, two execution threads are scheduled to run on a single core, achieving yet more parallelism, using hyper-threading.

And there is even more. Through a unique collaboration, SAP engineers and Intel engineers have labored side-by-side to optimize the inner loops of the SAP HANA database code for the Intel instruction set. In particular, SAP HANA was carefully tuned to optimize data access loops to maximize cache sizes and cache coherency, and its code takes full advantage of the sophisticated parallel vector operations of the SSE4 instruction set extensions.

4 Scaling SAP HANA

4.1 Understanding Workloads

SAP HANA is used in a variety of scenarios like as Data Marts, as a database for SAP Business Warehouse or SAP Business Suite, and as the general data-store for new, innovative real-time applications. Each use case and workload is different, and the scalability requirements are therefore different as well.

Generally speaking, the scalability requirements fall into two main categories:

- **Data Volume** – Since the SAP HANA System Architecture has an important and limited memory component, it is critically important to be able to estimate and understand the in-memory footprint and memory consumption of the application over time
- **Performance** – Performance scale is determined by several parameters: growing number of connections, users and query volumes, growing query complexity, and of course, the impact of the growing data sizes

Performance is fundamentally measured by two main indicators: *response-time* and *throughput*.

Response time is the time experienced by the database client, from sending a request to receiving a response. Response time is a function of the query complexity, the data-size and SAP HANA parallelization, but it is influenced by the overall load on the system as well.

Throughput is the number of "average" queries handled per unit of time, for instance queries per hour. To predict or calculate the system throughput, it is necessary to specify and measure the cost and rate of the most dominant queries.

More detailed scalability considerations are discussed below.

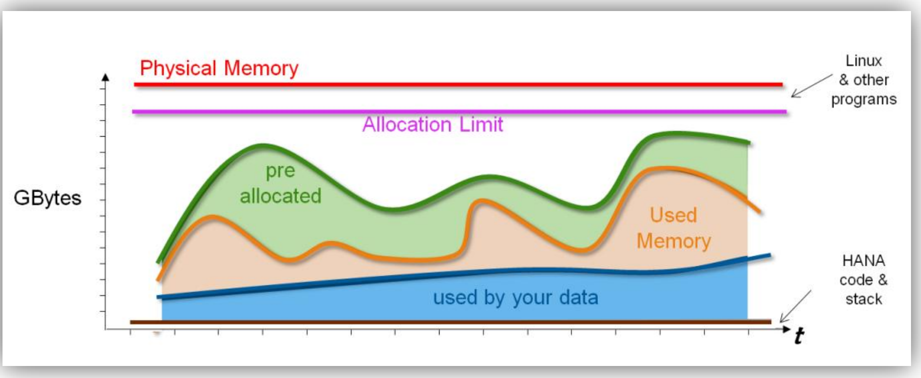
4.2 Scaling the Data

Memory is mainly used for two purposes: the database tables and as temporary storage for calculations performed by queries. The more data, and/or concurrently executed queries, the more memory is required.

The amount of RAM required to use SAP HANA will thus be one of the first scalability considerations. This is critical because of the following reasons:

- Memory is an element of the SAP HANA acquisition cost and affects run-time licenses
- SAP HANA provides advanced data compression and reduction, which makes it more difficult to calculate in advance the required physical memory footprint for a given data-set
- The amount of used memory is greatly impacted by the type and number of concurrent queries
- Data set volumes tend to grow over time, and thus memory-planning is crucial

The following figure illustrates the use of memory in a SAP HANA system over time. The top red line represents the actual amount of physical RAM in the system (one server or aggregated over multiple servers). The *allocation-limit,* by default set at 90% of RAM, corresponds to the SAP HANA runtime license. SAP HANA uses a dynamic pool to "reserve" memory in advance of its use, up to this limit. SAP recommends reserving about 50% of the used memory pool for the query-related temporary calculations, but this value is really affected by the application workload.



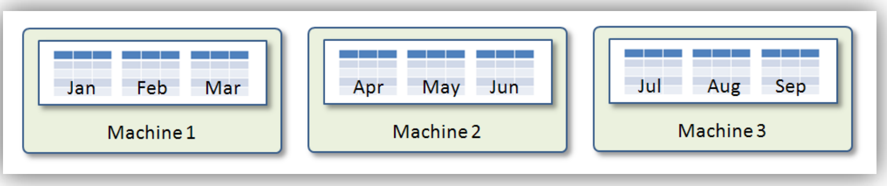
SAP HANA has several mechanisms to deal with low-memory situations. For instance, it will "unload" least-used tables or table-columns to free up memory, knowing that these columns can always be reloaded later from the persistent copy and logs. This is easy to do due to its columnar data-organization.

SAP has published several "sizing guides" to help you estimate the memory requirements of an application and its data set; see the reference section of this paper.

One technique to deal with planned expansion of the data is to purchase more physical RAM than is initially required, set the allocation-limit according to need, and increase it over time to adapt to the growing data.

Once the physical limits of a single server have been reached, it is possible to scale out over multiple machines to create a distributed SAP HANA system. One technique is to distribute different schemas and tables to different server (complete data and user separation), but that is not always possible, for instance when a single fact table is larger than the server's RAM size.

The most important strategy for data scaling is *data partitioning*. Partitioning supports the creation of very large tables (billions of rows) by breaking them into smaller chunks that can be placed on different machines, as illustrated below:



Partitioning is transparent for most SQL queries and other data manipulations. Typical uses cases for partitioning:

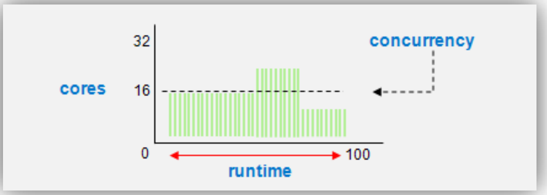
1. Overcoming the non-partitioned limitation of 2 billion rows per table
2. Load balancing: Using partitioning, a query on a table is not processed by a single server but by all servers that host partitions that are relevant for processing.
3. Parallelization: Operations are parallelized by using several execution threads per table.
4. Partition pruning: SAP HANSA analyzes queries to find a match with the partitioning specification. If found, *only* the matching machines need to be queried; this reduces the overall system load and improves response time.
5. Explicit partition handling: Applications may actively control partitions, for example by adding partitions that should hold the data for an upcoming month.

4.3 Scaling the Performance

Moving along the performance scale is a more complex topic. As discussed, SAP HANA's performance is derived from its efficient, parallelized approach. Generally speaking, the more computation cores the SAP HANA server has, the better the overall system performance.

This is a simplified view of course, and scaling performance requires a more detailed understanding of the workload and performance expectations.

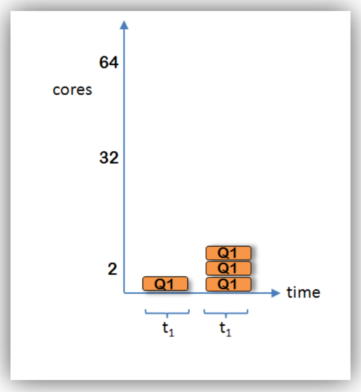
When speaking of a single query, we speak of the query's *run-time*. The *concurrency* of a query is the *average* number of cores used to execute it. This is shown in the following figure:

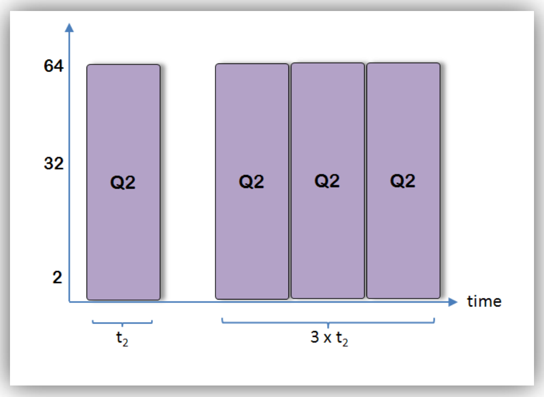


The computational cost of the query is therefore the "area" of the query (concurrency multiplied by the runtime, in units of *core-ticks*). Consider, for instance, an SQL query Q1 that performs a relatively simple transaction that needs not be parallelized, and so only uses one core for its duration. Executing, two, or twenty, or even two hundred of such queries concurrently allows the database to schedule each query on a separate compute core. Each of the concurrent queries will appear to run almost as fast as a single instance of the query by itself.

Consider, on the other extreme, a query Q2 that executes a complex analytical aggregation on a huge OLAP-structured table. The most efficient implementation of this query will use all the available resources on the machine for the entire duration of the query, resulting in an awe-inspiring performance result. What happens, if two or more similar queries are executed concurrently? Now, the machine resources must be *shared*, and each of the concurrent queries will appear to run for a much longer time.

This situation is illustrated below on a 64-core machine. The picture on the left shows how 3 concurrent instances of Q1 take the same time as one stand-alone instance. The picture on the right shows how 3 concurrent instances of Q2 take three times as much time as a single instance of this query!





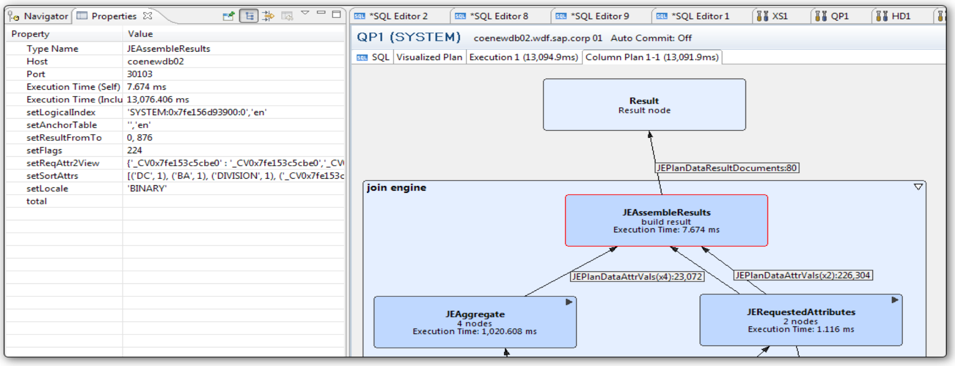
The reason is that three instances of Q1 take only 3 x t_1 core-ticks, while three instances of Q2 take 192 x t_2 core-ticks, and thus run three times longer than a single query. Users experience a longer response time.

Of course, most real queries are in between these fictional examples. Using simulations and estimations of typical customer query workloads can determine the expected load that a typical SAP HANA installation may manage comfortably.

At the workload level, a rough prediction of scalability can be established by measuring the average CPU utilization while the workload is running. For instance, an average CPU utilization of 45% may indicate that the system can be loaded 2X before showing a considerable reduction in individual query response time.

Several tools exist to measure and analyze production performance and throughput, most of which can be accessed via the SAP HANA Studio. Among these tools are various administrative perspectives, which provide a high level view of the system performance, lists of top SQL statements sorted by performance and frequency, performance graphs over time, and more.

Zooming into details, a wealth of system tables and monitoring views provide statistical insights in hundreds of performance parameters that can be also be accessed programmatically. Other tools provide deeper insights into individual query behavior, such as the Execution Plan Visualization tool, pictured below.



Property	Value
Type Name	JEAssembleResults
Host	coenewdb02
Port	30103
Execution Time (Self)	7.674 ms
Execution Time (Inclu	13,076.406 ms
setLogicalIndex	'SYSTEM:0x7fe156d93900:0','en'
setAnchorTable	'','en'
setResultFromTo	0, 876
setFlags	224
setReqAttr2View	{'_CV0x7fe153c5cbe0' : '_CV0x7fe153c5cbe0','_CV
setSortAttrs	[('DC', 1), ('BA', 1), ('DIVISION', 1), ('_CV0x7fe153c
setLocale	'BINARY'
total	

QP1 (SYSTEM) coenewdb02.wdf.sap.corp 01 Auto Commit: Off

Result
Result node

JEPlanDataResultDocuments:80

JEAssembleResults
build result
Execution Time: 7.674 ms

JEPlanDataAttrVals(x4):23,072

JEPlanDataAttrVals(x2):226,304

JEAggregate
4 nodes
Execution Time: 1,020.608 ms

JERequestedAttributes
2 nodes
Execution Time: 1.116 ms

4.4 Scaling the Application

Application developers have many ways to improve the interaction with SAP HANA, and achieve optimal performance. Existing application code can be scaled even further by judiciously pushing operations into the SAP HANA data-layer, using SQLScript and other embedded code constructs. Since this will result in a greater use of internal operators, it takes optimal advantage of the hardware and software resources of a SAP HANA system.

Modeling the data into the form of a classic OLAP star schema may result in impressive performance improvements, due to the very high parallel capabilities of the SAP HANA OLAP optimizer.

Partitioning, mentioned above as a solution for scaling out SAP HANA memory requirements, also supports ever more concurrent sessions and ever more complex analytical queries, by spreading the calculations onto multiple servers. Particular care must be taken in distributing the data, such that a majority of the queries will match partitioning *pruning* rules. This accomplishes two goals: directing different users to different servers (load balancing) and avoiding the network overhead related to frequent data joining across servers.

Other techniques to increase query throughput even more are the use of calculated columns to represent common filter conditions, the use of partitions even on a single server to exploit pruning, the deployment of load-balancers with replicated database instances, partial aggregations, and more.

4.5 Scaling the Hardware

SAP HANA is offered in two ways – in the form of appliances, delivered in a number of different configurations and “sizes” by qualified hardware partners, or as part of a cloud-based service. This creates different system design options, with respect to scale-up and scale-out variations.

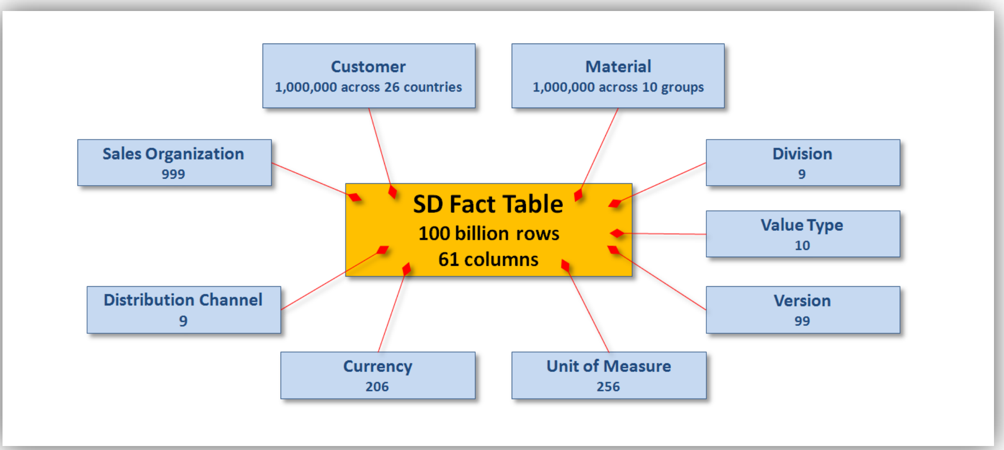
To maximize performance and throughput, SAP recommends that you scale ***up*** as far as possible (acquire the configuration with the highest processor and memory specification for the application workload), before scaling ***out*** (for deployments with even greater data volume requirements).

Note that some of the hardware vendors recommend different building blocks for their single-machine and scale-out deployments (rack servers vs. blades), so planning a scale-out strategy must take this into account.

5 Scalability Benchmark Test

A benchmark test was developed to demonstrate SAP HANA scalability and analytic performance for real-time business intelligence on a representative database, using 16 servers. The 16 servers were each equipped with 0.5TB of memory and 80 logical cores.

A 100 TB[1] Sales & Distribution data set was generated in a format as would be extracted from an SAP ERP system (e.g., data records with multiple fields) for analysis in a business warehouse.



The data consists of a hundred billion-record fact table and several smaller dimension tables, representing 5 years’ worth of sales and distribution data. The data is hash-partitioned equally across 16 servers using the customer id. Within a server, the data is further partitioned into one-month intervals. This results in 60 partitions per server and approximately 104 million records per partition.

[1] 100 TB = raw data set size before compression.

The data was loaded in parallel using SAP HANA's IMPORT command. Loading is automatically parallelized across all of the nodes. The load rate was measured at 16 million records per minute, sufficient to load 100 million records (representing one business day's activity) in just six minutes.

Data compression occurs during the data loading process. SAP HANA demonstrated a greater than 20X compression rate[2]; the 100TB data set was reduced to a trim 3.78 TB HANA database, consuming only 236 GBs of RAM on each node. The large fact table accounts for 85% of the data set, and was compressed to occupy less than half of the available RAM per node.

The query suite represents a mixed workload using the data in representative native form (i.e., not indexed, tuned or otherwise de-normalized to avoid joins, as would be customary in a conventional database). The queries cover a range of realistic BI activities including general reporting, iterative querying (drill downs), ranking, and year-over-year analysis.

Testing methodology included both baseline execution time and multi-stream throughput.

The iterative queries demonstrate SAP HANA's excellent ability to aggregate data. E.g., the longest-running query ran in just over a second but took only 2.8X longer to analyze 12X more data than its monthly equivalent. The Analytic queries efficiently performed sophisticated joins (fact-to-fact, sub queries, CSQ, union all) and analysis across a sliding time window (year-over-year). The queries with one to six-month date ranges ran in two seconds or less; the most complex query, which analyzed the entire 5-year date range, ran in under four seconds. All analytic query times are well within timeframes to support iterative speed-of-thought analysis.

Twenty-five concurrent query streams (representing about 2600 "typical" active users) ran in about 3X the time of a single query stream, demonstrating good throughput scalability.

6 In Summary

SAP HANA's design for scalability demonstrates its ability to deliver real-time BI-query performance as workloads increase in terms of capacity, complexity, and throughput. Multiple implementation techniques and deployment options ensure flexibility and choice in meeting demanding scalability requirements.

The benchmark test indicates that SAP HANA offers linear scalability with sustained performance at large data volumes. Advanced compression methods were applied directly to the columnar database without degrading the query performance. Thus, SAP HANA offers the potential to access raw transactional ERP data in virtual real-time.

7 References

- SAP HANA: http://www.sap.com/hana and https://www.experiencesaphana.com
- HANA Sizing Guides: http://help.sap.com/hana/
- SAP HANA Scale-Out Guide: http://help.sap.com/hana/hana1_imdb_scale_en.pdf
- SAP HANA
  Partioning and Distribution of Large Tables: http://help.sap.com/hana/hana_db_part_en.pdf
- SAP HANA Administration guide: http://help.sap.com/hana/hana1_imdb_studio_admin_en.pdf
- Analyzing SAP HANA's Performance Test, J. Greenbaum, *Enterprise Applications Consulting, May 2012*: https://www.experiencesaphana.com/docs/DOC-1755

[2] Compression rates depend *heavily* on characteristics of the actual data to be compressed. Individual results may vary.