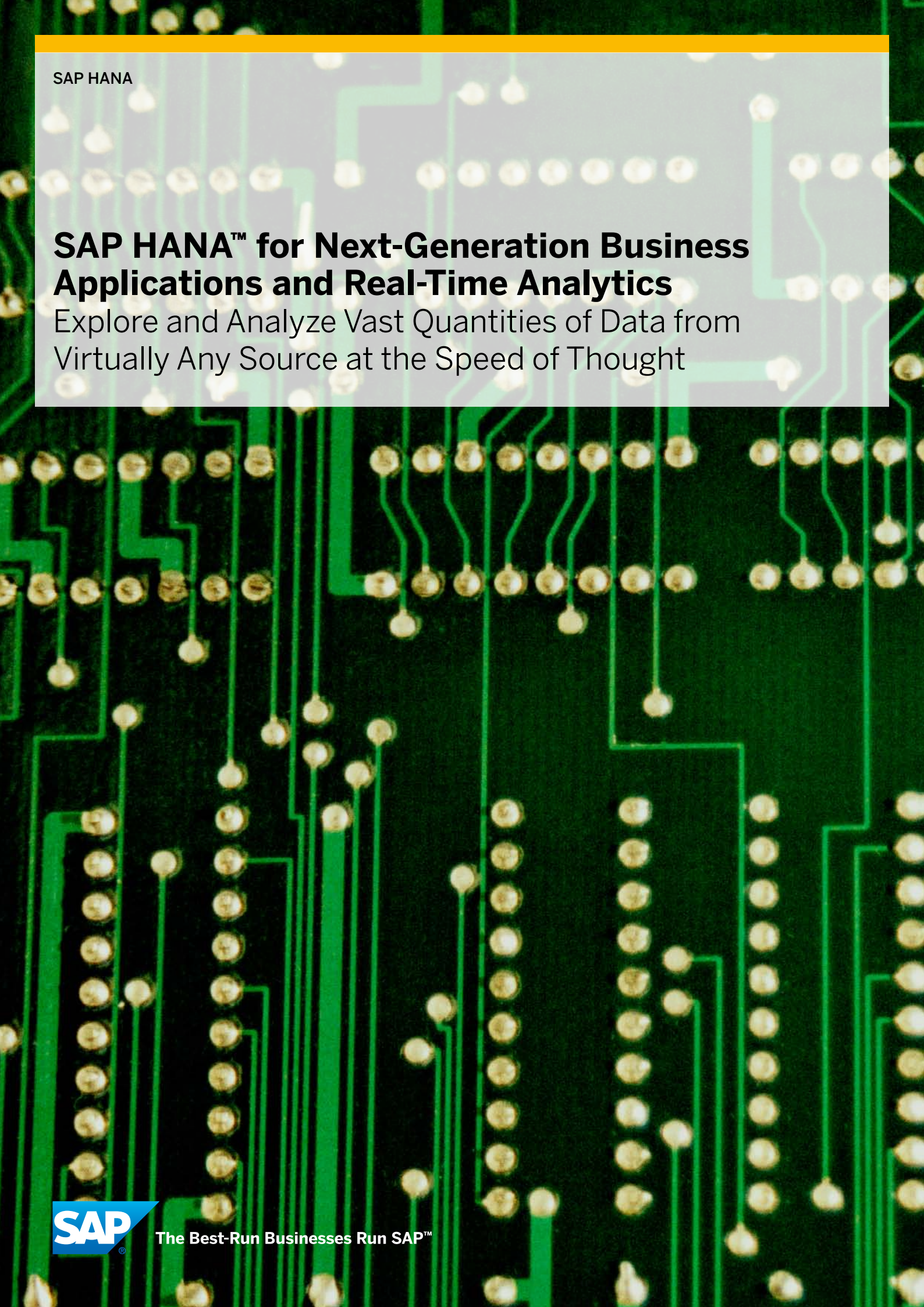
SAP HANA

SAP HANA™ for Next-Generation Business Applications and Real-Time Analytics

Explore and Analyze Vast Quantities of Data from Virtually Any Source at the Speed of Thought



SAP

The Best-Run Businesses Run SAP™

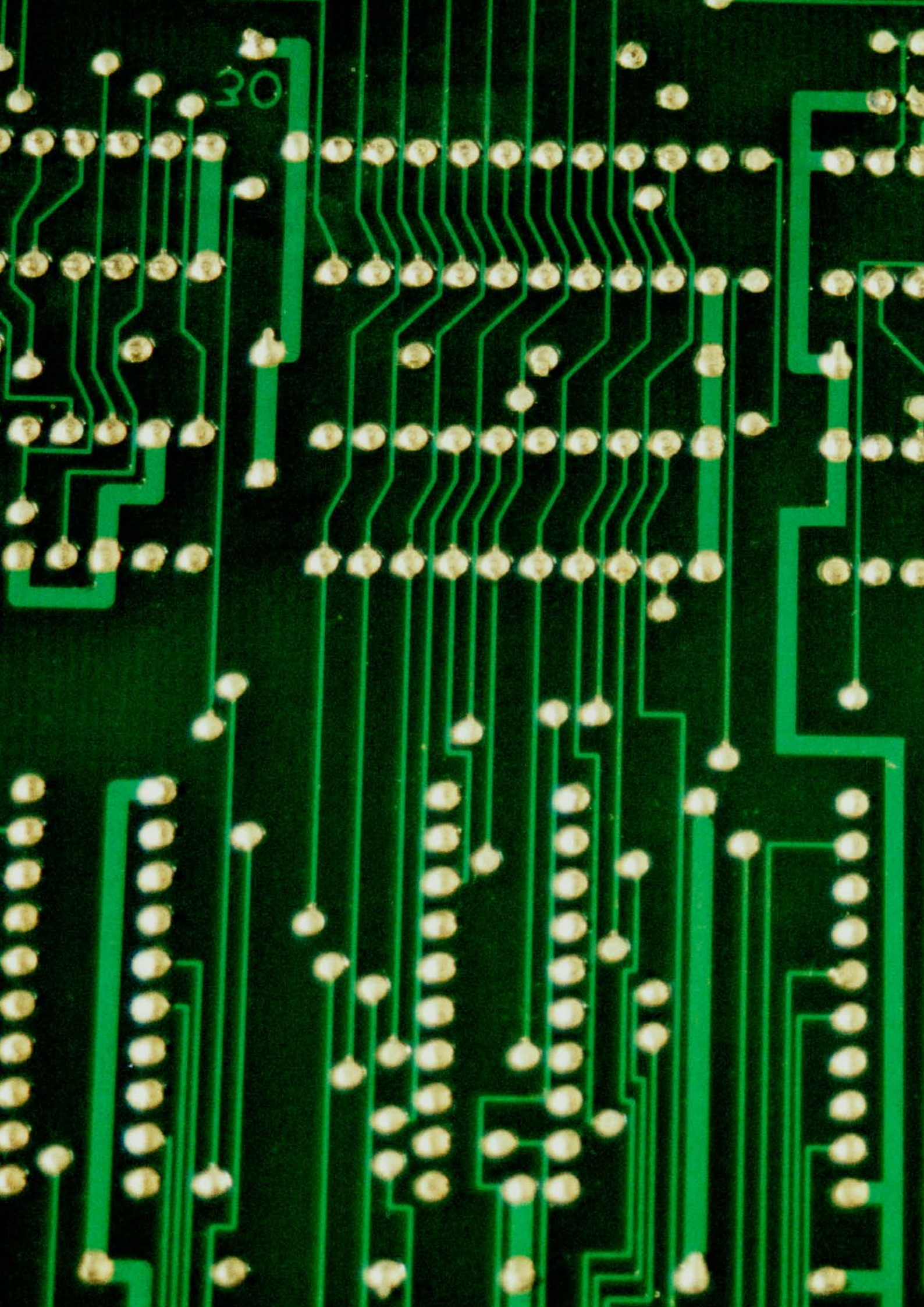

30

Table of Contents

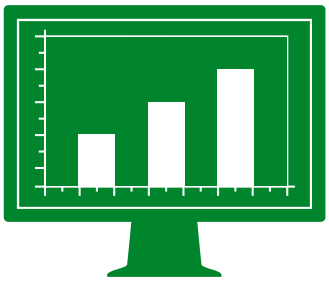
Revolutionize the Way You Run Your Business

SAP has introduced a new class of solutions that powers the next generation of business applications. The SAP HANA™ database is an in-memory database that combines transactional data processing, analytical data processing, and application logic processing functionality in memory. SAP HANA removes the limits of traditional database architecture that have severely constrained how business applications can be developed to support real-time business.

In recent years we have observed interrelated technological and social revolutions. Computer systems have a continuously increasing number of processing cores with large integrated caches. Main memory space has become practically unlimited with the ability to hold all the business data of enterprises of every size. With the exploding number of mobile devices, information for decision making needs to be available in seconds. Changes to the business, risks, and opportunities need to be tracked in real time to keep the business competitive.

Leaning on long-term business experience and visionary research, SAP has created a new paradigm of building business applications that takes advantage of these advancements in technology. The paradigm is realized in the SAP HANA platform. SAP HANA enables you to perform real-time online application processing (OLAP) analysis on an online transaction processing (OLTP) data structure. As a result, you can address today's demand for real-time business insights by creating business applications that previously were neither feasible nor cost-effective.

SAP HANA introduces a different way of thinking from both the technical side of how to construct applications and the business side of how to exploit the new functionality. SAP offers an incremental road map that enables your organization to safely explore the technology today, seize opportunities in side-by-side scenarios, and get prepared for the increasing rate of doing business.



SAP HANA enables you to perform real-time OLAP analysis on an OLTP data structure. As a result, you can address today's demand for real-time business insights by creating business applications that previously were neither feasible nor cost-effective.

Current Generation of Enterprise Solutions

The architectures of current-generation business systems reflect the technological conditions during the long evolution of business solutions:

- Database layer: Database management systems were designed for optimizing performance on hardware with limited main memory and with the slow disk I/O as the main bottleneck. The focus was on optimizing disk access, for example, by minimizing the number of disk pages to be read into main memory when processing a query.
- Business application layer: Business software was built with a sequential processing paradigm. Data tables for the current scenario were fetched from the database, processed on a row-by-row basis, and pushed back to the database.

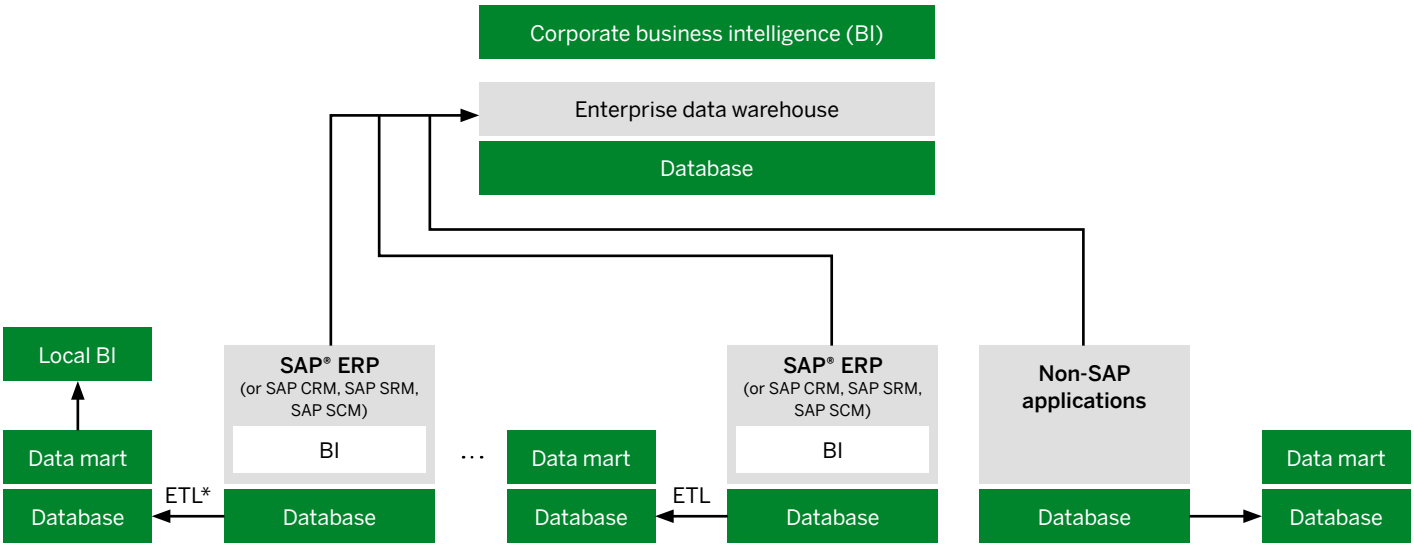
As discussed by Plattner and Zeier in their recent book on in-memory data management, technological database limitations have forced a separation of the data management solution into transactional and analytical processing:

- Online transaction processing (OLTP) systems are highly normalized to data entry volume and to speed up inserts, updates, and deletes. This high degree of normalization is a disadvantage when it comes to retrieving data, as multiple tables may have to be joined, which severely impacts performance.
- Online application processing (OLAP) systems were developed to address the requirements of large enterprises to analyze their data in a timely manner. These systems exploit specialized data structures designed to optimize read performance and provide quick processing of complex analytical data.

Data needs to be transferred out of an enterprise's transactional system into an analytical system and is then prepared for predefined reports.[1]

Figure 1 illustrates a typical enterprise software situation in current-generation solutions: Large organizations have multiple enterprise resource planning (ERP) systems, each with its own database for operational data. Analytical data is consolidated in an enterprise data warehouse in a batch offline process and consumed by business users via corporate business intelligence (BI) solutions. Lines of business that need custom reporting on more recent data (but not real time) use additional data marts and local BI clients.

Figure 1: Typical Enterprise Software Situation



*ETL = extract, transform, load

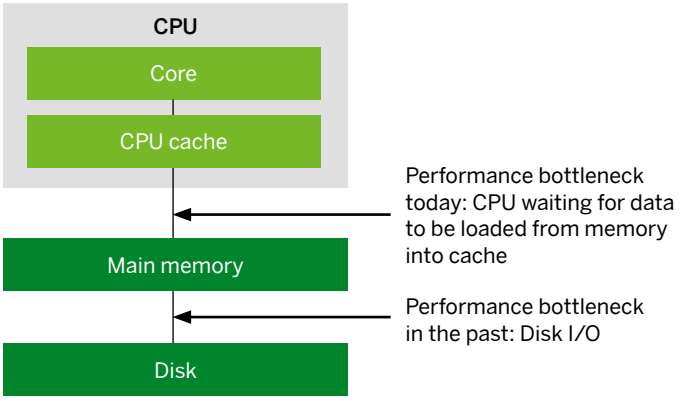
Technological Transition

Computer architecture has changed. Today's multicore, multi-CPU server provides fast communication between processor cores via main memory or shared cache. Main memory is no longer a limited resource. In 2012 servers with more than 2 terabytes of RAM are available.

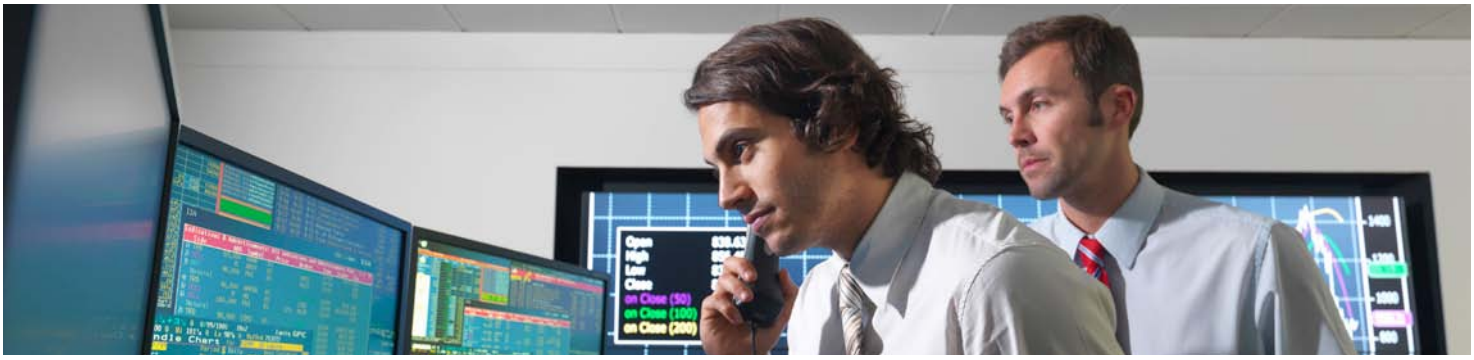
Modern computer architectures create new possibilities but also new challenges. With all relevant data in memory, disk access is no longer a limiting factor for performance. In 2012 server processors have up to 80 cores, and 128 cores will come in the near future. With the increasing number of cores, CPUs will be able to process more and more data per time interval. That means the performance bottleneck is now between the CPU cache and main memory (see Figure 2). An optimized database technology should focus on optimizing memory access by the processing cores. Simple disk access optimization by caching data in memory may not yield breakthrough performances.

To provide an idea about sizes and access speeds of a current memory hierarchy, the table below compares the different layers in this memory hierarchy (CPU characteristics for Intel's Nehalem architecture).

Figure 2: Hardware Architecture: Current and Past Performance Bottlenecks



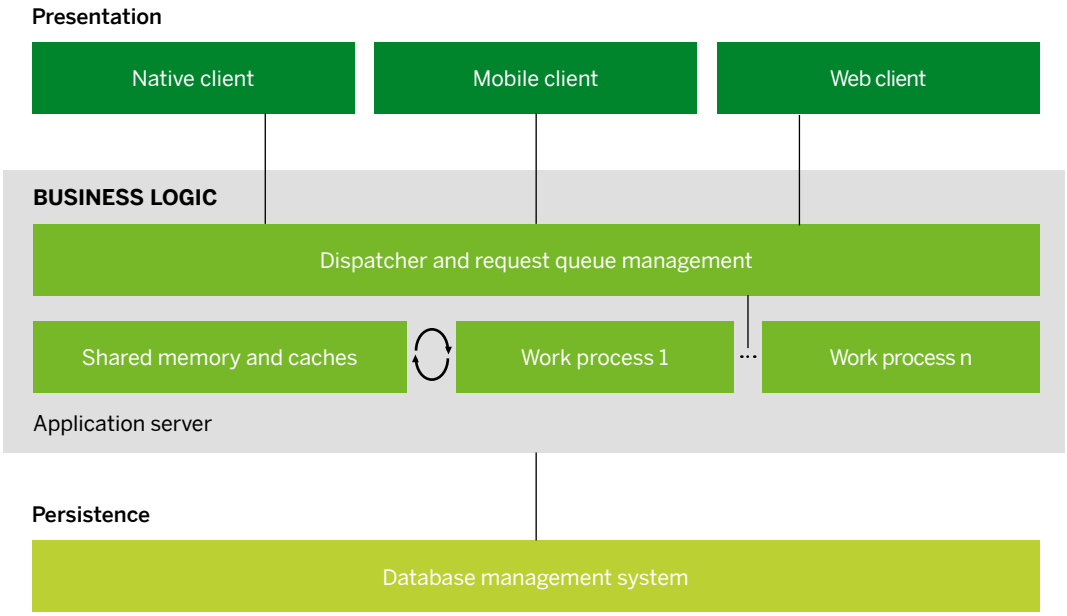
Type of Memory	Size	Latency
L1 cache	64 KB	~4 cycles [2 ns]
L2 cache	256 KB	~10 cycles [5 ns]
L3 cache (shared)	8 MB	35–40+ cycles [20 ns]
Main memory	GBs up to terabytes	100–400 cycles
Solid state memory	GBs up to terabytes	5,000 cycles
Disk	Up to petabytes	1,000,000 cycles



Since traditional business solutions lean on OLTP databases, they do not use current hardware efficiently. Figure 3 illustrates the architecture of the SAP® ERP application.

Data is transmitted from the database tier to the application logic tier for processing. As illustrated in Figure 2, the performance bottleneck is between the main memory and the CPU. Simple memory-resident caching with a traditional database system is not the solution. The CPU spends half of the execution time in wasted stalls for two reasons: waiting for data being loaded from main memory into the CPU cache, and the fact that sequential processing of the business application cannot properly utilize the increasing number of processing cores. A new computing model, focused on exploiting the CPU cache and scale in parallel to multicore processing, needs to be developed.

Figure 3: Three-Tier Architecture of SAP® ERP[2]



SAP HANA Platform Overview

TAKING A NEW APPROACH TO BUSINESS DATA PROCESSING

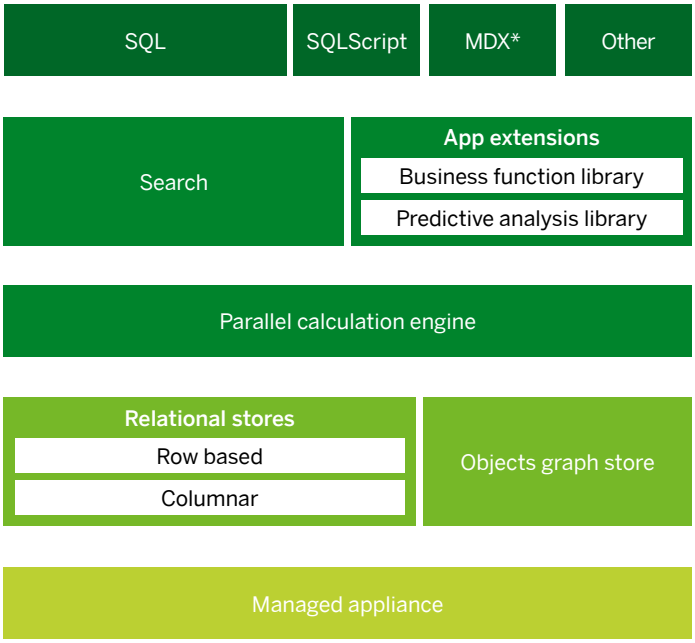
The SAP HANA platform implements a new approach to business data processing. In fact, it is much more than the traditional definition of a database. And the in-memory attribute is much more than a naïve caching of disk data structures in the main memory.

A conceptual view of SAP HANA is illustrated in Figure 4. While many of the concepts discussed below may be known in the industry, the specific synergy of SAP HANA, which leverages SAP expertise in different domains, creates a new class of solutions.

Complete DBMS As Its Backbone

SAP HANA, first and foremost, incorporates a full database management system (DBMS) with a standard SQL interface, transactional isolation and recovery (ACID [**a**tomicity, **c**onsistency, **i**solation, **d**urability] properties), and high availability. SAP HANA supports most entry-level SQL92. SAP applications that use Open SQL can run on the SAP HANA platform without changes. SQL is the standard interface to SAP HANA. Additional functionality, such as freestyle search, is implemented as SQL extensions. This approach simplifies the consumption of SAP HANA by applications.

Figure 4: Conceptual View of SAP HANA™ Platform



*MDX = multidimensional expression

Analytical and Special Interfaces

In addition to SQL, SAP HANA supports business intelligence clients directly using multidimensional expressions (MDX) for products such as Microsoft Excel and business intelligence consumer services (BICS), an internal interface for SAP BusinessObjects™ solutions. For analytical planning, the user may iterate values on the aggregated analytical reports. With SAP HANA, a single value is transmitted for immediate recalculation by the in-memory planning engine.

Parallel Data Flow Computing Model

To natively take advantage of massively parallel multicore processors, SAP HANA manages the SQL processing instructions into an optimized model that allows parallel execution and scales incredibly well with the number of cores. The optimization includes partitioning the data in sections for which the calculations can be executed in parallel. SAP HANA supports distribution across hosts. Large tables may be partitioned to be processed in parallel by multiple hosts.

Figure 5 summarizes the results of a scale test that was performed by an Intel team in collaboration with SAP. The test demonstrates near-linear scale. The processing time is 16.8 seconds using 2 cores and improves to 1.4 seconds using 32 cores. Hyperthreading adds an additional 20% improvement.

Application Logic Extensions

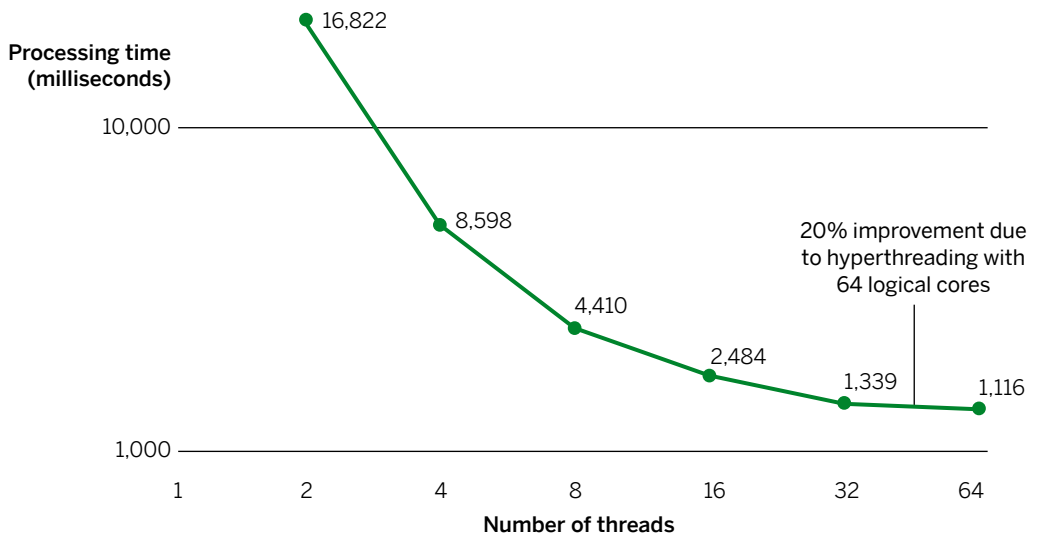
The parallel-data-flow computing model is extended with application-specific logic that is executed in processing nodes as part of the model. Support includes SQLScript as a functional language and "L" as an imperative language,[3] which can call upon the prepackaged algorithms available in the predictive analysis library of SAP HANA to perform advanced statistical calculations. The application logic languages and concepts are evolving as a result of collaboration with internal and external SAP developer communities.

Business Function Library and Predictive Analysis Library
SAP leverages its deep application expertise to port specific business application functionality as infrastructure within SAP HANA to natively take advantage of in-memory computing technologies by optimizing application and calculation processing directly within main memory. Examples include currency conversion, a fundamental first step for a global company in which many reports, which otherwise may have utilized plain SQL, utilize parallel processing well. Another example is converting business calendars: different countries use different civilian or business calendars and have different definitions of a fiscal year.

Multiple In-Memory Stores Optimized Per Task
Native in-memory storage does not leverage the processing power of modern CPUs well. The major optimization goal of SAP HANA is to achieve high hit ratios in the different caching layers of the CPU. This is done by data compression and by adapting the data store for the task. For example, when the processing is done row by row and consumes most of the fields within a row, a **row store** in which each row is placed in memory in sequence provides the best performance. If calculations are executed on a single column or several columns only, these are scanned or aggregated into a **column store** in which each column is stored in sequence as a (compressed) memory block, providing much better results. An **objects graph store** may benefit from a structure in which each object body is stored in sequence and the graph navigation is stored as another sequence to support unstructured and semistructured data storage.

Appliance Packaging
SAP HANA applies optimizations that take CPU utilization to the extreme. Appliance packaging enables full control of the resources and a certification process for hardware configurations for best performance and reliability. For example, SAP HANA includes automatic recovery from memory errors without system reboot. Systems with high memory are statistically more sensitive to such error. In addition to a beneficial total cost of ownership of the appliance packing model, it is a fundamental part of the SAP HANA design concept. Additionally, SAP is actively investigating virtual appliances and cloud templates for SAP HANA to validate additional performance use cases.

Figure 5: Example for Near-Linear Scale
Joining TPC-H Data set (120 million records) in SAP HANA™ on 4S Nehalem-EX (2.26 GHz) with 64 logical cores



Technology Foundation

EXECUTION CONTROL

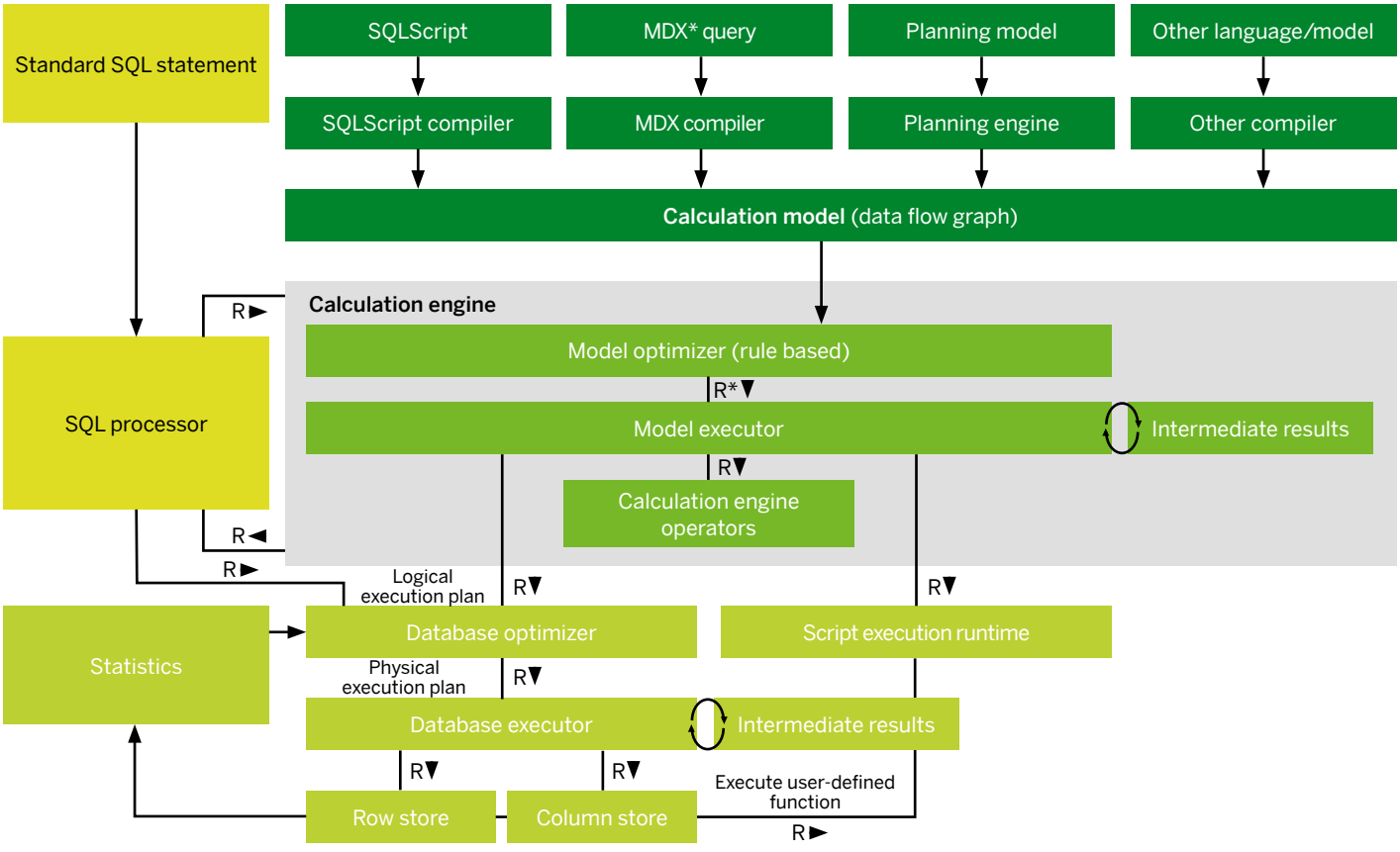
Execution control is illustrated in Figure 6. SQLScript, MDX, and the planning engine interface can be seen as domain-specific programming languages or models that can be used to interact with SAP HANA. The artifacts in the different domain-specific languages are translated by their specific compilers into a common representation called a "calculation model."

A calculation model is a directed acyclic graph with arrows representing data flows and nodes that represent operations. This approach and the exclusion of loops and recursion enable automatic massive parallelism of the processing.

The easiest way to think of calculation models is to see them as data flow graphs, where the modeler defines data sources as inputs and different operations (join, aggregation, projection, and so on) on top of them for data manipulation.

The calculation engine automatically breaks up a model into operations that can be processed in parallel (model optimizer). These operations are passed to the database optimizer, which determines the best plan for accessing row or column stores, leveraging cost-based optimizations and database statistics. Parallel process paths are illustrated in Figure 7.

Figure 6: Processing Chain (Conceptual)



*MDX = multidimensional expression; *R = request

CALCULATION MODELS FOR SQLSCRIPT TABLE FUNCTIONS

Coding Layers

As shown in Figure 8, the upper code layer within SAP HANA is enabled through SQLScript, which is the rich stored-procedure language of the SAP HANA database. SQLScript procedures may contain SQL statements and can call other procedures. It is used to write procedural orchestration logic and to define complex data flows. SQLScript is compiled first into "L" (restricted subset of C++), which is then compiled into native code. SAP has developed, directly in C++, a business function library (BFL) that includes functionality for performing business processing at the database layer. BFL can be consumed by the layers above.

SQLScript

SQLScript functions composed of SQL queries and function calls can be represented as acyclic data flow graphs. These functions, implemented in "L" language, are typically transformed into calculation models that contain only one transformation node of type "L script." In some cases a more complex graph may be created with SQL nodes for embedded SQL queries and nodes for imperative code sequences.

Each node has a set of inputs and outputs and an operation that transforms the inputs into outputs. In addition to their primary operation, each node can also have a filter condition for filtering the result set. The inputs and the outputs of the operations are table-valued operands. Inputs can be connected to SAP HANA tables or to the outputs of nodes.

Calculation models support a variety of node types:

- Nodes for set operations such as projection, aggregation, join, union, minus, and intersection
- SQL nodes that execute an SQL statement that is an attribute of the node
- Scripting nodes for describing complex operations that cannot be described with a graph of data transformations. The function of such a node is described by a procedural script.

To enable parallel execution, a calculation model may contain split and join operations. A split operation is used to partition input tables for subsequent processing steps based on partitioning criteria. Operations between the split and join operation may then be executed in parallel for the different partitions.

Figure 7: Parallel Process Paths

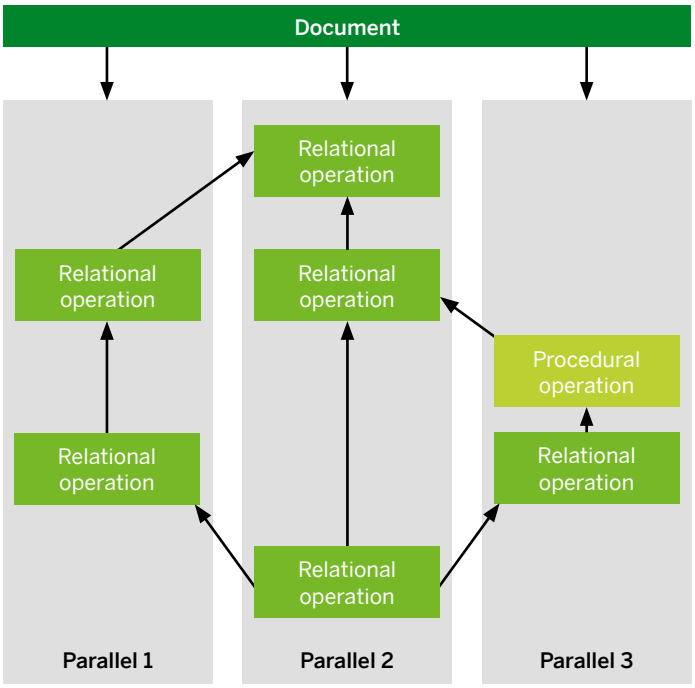
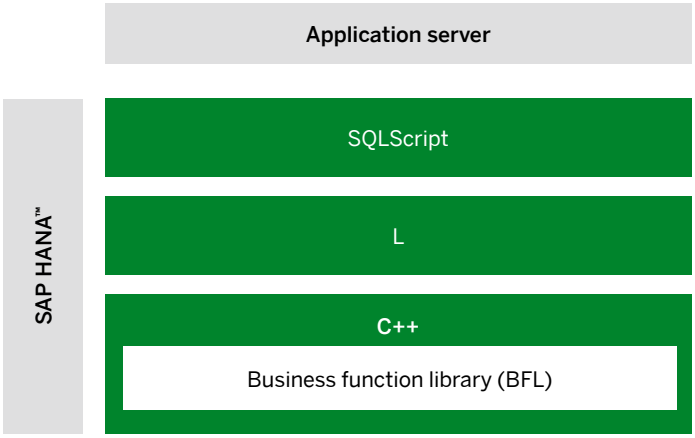


Figure 8: Coding Layers in SAP HANA



The compilers that create the calculation model try to transform as much of the input program as possible into a calculation model that consists only of set operation nodes and SQL nodes. However, this is not possible in all cases. For example, loops where an iteration depends on the results of previous iterations cannot be transformed into data flow graphs. In those cases, the parts of the domain-specific model that cannot be transformed into data flow transformations are translated into scripting nodes with procedural code in "L" language.

Calculation models are more powerful than traditional SQL queries or SQL views for two reasons:

- These models offer the possibility to define specialized parameterized calculation schemas when the actual query is issued. Unlike an SQL view, a calculation model does not describe the actual query to be executed. Instead, it describes the structure of the calculation. Additional information is supplied when the calculation model is executed. Calculation engines use the actual parameters, attribute lists, grouping attributes, and so on supplied with the invocation to instantiate a query-specific calculation model. This "instantiated model" is optimized for the actual query and does not contain attributes, nodes, or data flows that are not needed for the specific invocation.
- These models allow more flexible, scripted operations, via SQLScript or imperative scripts in "L" language.

PARALLEL EXECUTION

SAPA HANA is engineered for parallel execution that scales well with the number of available cores and across hosts when distribution is used. Specifically, optimization for multicore platforms accounts for the following two key considerations:

- Data is partitioned wherever possible in sections that allow calculations to be executed in parallel.
- Sequential processing is avoided, which includes finding alternatives to approaches such as thread locking.

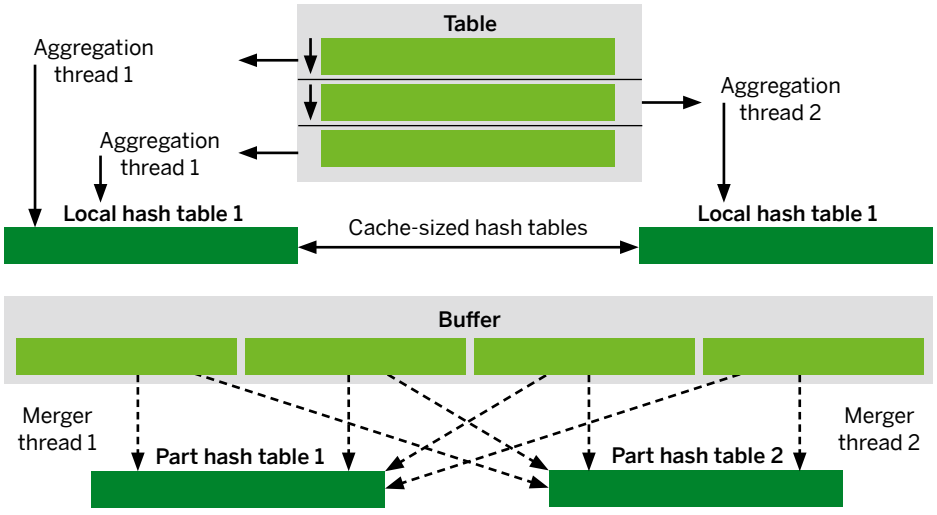
Parallel Aggregation

In the shared-memory architecture within a node, SAP HANA performs aggregation operations by spawning a number of threads that act in parallel, each of which has equal access to the data resident on the memory on that node. As illustrated at the top of Figure 9, each aggregation functions in a loop-wise fashion as follows:

1. Fetch a small partition of the input relation
2. Aggregate that partition
3. Return to step 1 until the complete input relation is processed

Each thread has a private hash table where it writes its aggregation results. When the aggregation threads are finished, the buffered hash tables are merged by merger threads using range partitioning. Each merger thread is responsible for a certain range.[4]

Figure 9: Use of Parallel Aggregation by SAP HANA™ to Divide Work Among Threads[5]



Columnar and Row-Based Data Storage

One of the differentiating attributes of SAP HANA is having both row-based and column-based stores within the same engine.

Conceptually, a database table is a two-dimensional data structure with cells organized in rows and columns. Computer memory, however, is organized as a linear sequence. For storing a table in linear memory, two options can be chosen, as shown in Figure 10. Row storage stores a sequence of records that contain the fields of one row in the table. In a column store, the entries of a column are stored in contiguous memory locations.

CHOOSING BETWEEN COLUMN AND ROW STORE

With SAP HANA you can specify whether a table is to be stored by column or by row.

Row store is recommended if:
- The table has a small number of rows, such as configuration tables.
- The application needs to process only a single record at a time (many selects or updates of single records).
- The application typically needs to access the complete record.
- The columns contain mainly distinct values so the compression rate would be low.
- Aggregations and fast searching are not required.

Row store is used, for example, for SAP HANA database metadata, for application server internal data such as ABAP™ server system tables, and for configuration data. In addition, application developers may decide to put application tables in row store if the criteria given above are matched.

Column store is recommended if:
- Calculations are executed on a single column or a few columns only.
- The table is searched based on the values of a few columns.
- The table has a large number of columns.
- The table has a large number of rows, and columnar operations are required (aggregate, scan, and so on).
- The majority of columns contain only a few distinct values (compared to the number of rows), resulting in higher compression rates.

Figure 10: Row- and Column-Based Storage for a Table

Table

Country	Product	Sales
US	Alpha	3,000
US	Beta	1,250
JP	Alpha	700
UK	Alpha	450

Row store

Row 1	US
	Alpha
	3,000
Row 2	US
	Beta
	1,250
Row 3	JP
	Alpha
	700
Row 4	UK
	Alpha
	450

Column store

Country	US
	US
	JP
	UK
Product	Alpha
	Beta
	Alpha
	Alpha
Sales	3,000
	1,250
	700
	450

SAP HANA allows the joining of row-based tables with columnar tables. However, it is more efficient to join tables that are located in the same store. Therefore, master data that is frequently joined with transaction data is put in a column store.

ADVANTAGES OF COLUMNAR TABLES

When the criteria listed above are fulfilled, columnar tables have several advantages.

Higher performance for column operations. Operations on single columns, such as searching or aggregations, can be implemented as loops over an array stored in contiguous memory locations. Such an operation has high spatial locality and can efficiently be executed in the CPU cache.

Higher data compression rates. Columnar data storage allows highly efficient compression. Especially if the column is sorted, there will be ranges of the same values in contiguous memory, so compression methods such as run-length coding or cluster coding can be used. This is especially promising for SAP business applications as they have many columns containing only a few distinct values compared to the number of rows. Examples are country codes or status codes as well as foreign keys.

This high degree of redundancy allows for effective compression of column data. In row-based storage, successive memory locations contain data of different columns, so compression methods such as run-length coding cannot be used. In column stores a compression factor of 10 can be achieved compared to traditional row-oriented database systems.

Columnar data organization also allows highly efficient data compression. This not only saves memory but also increases speed for the following reasons:

- Compressed data can be loaded into CPU cache more quickly. As the limiting factor is the data transport between memory and CPU cache, the performance gain will exceed the additional computing time needed for decompression.
- With dictionary coding, the columns are stored as sequences of bit-coded integers. Checks for equality can be executed on the integers (for example, during scans or join operations). This is much faster than comparing string values.
- Compression can speed up operations such as scans and aggregations if the operator is aware of the compression. Given a good compression rate, computing the sum of the values in a column will be much faster if the column is run-length coded and many additions of the same value can be replaced by a single multiplication.

Elimination of additional indexes. Columnar storage eliminates the need for additional index structures in many cases. Storing data in columns works like having a built-in index for each column: the column scanning speed of the in-memory column store and the compression mechanisms, especially dictionary compression, already allow read operations with very high performance. In many cases, additional index structures will not be required. Eliminating additional indexes reduces complexity and eliminates effort for defining and maintaining metadata.

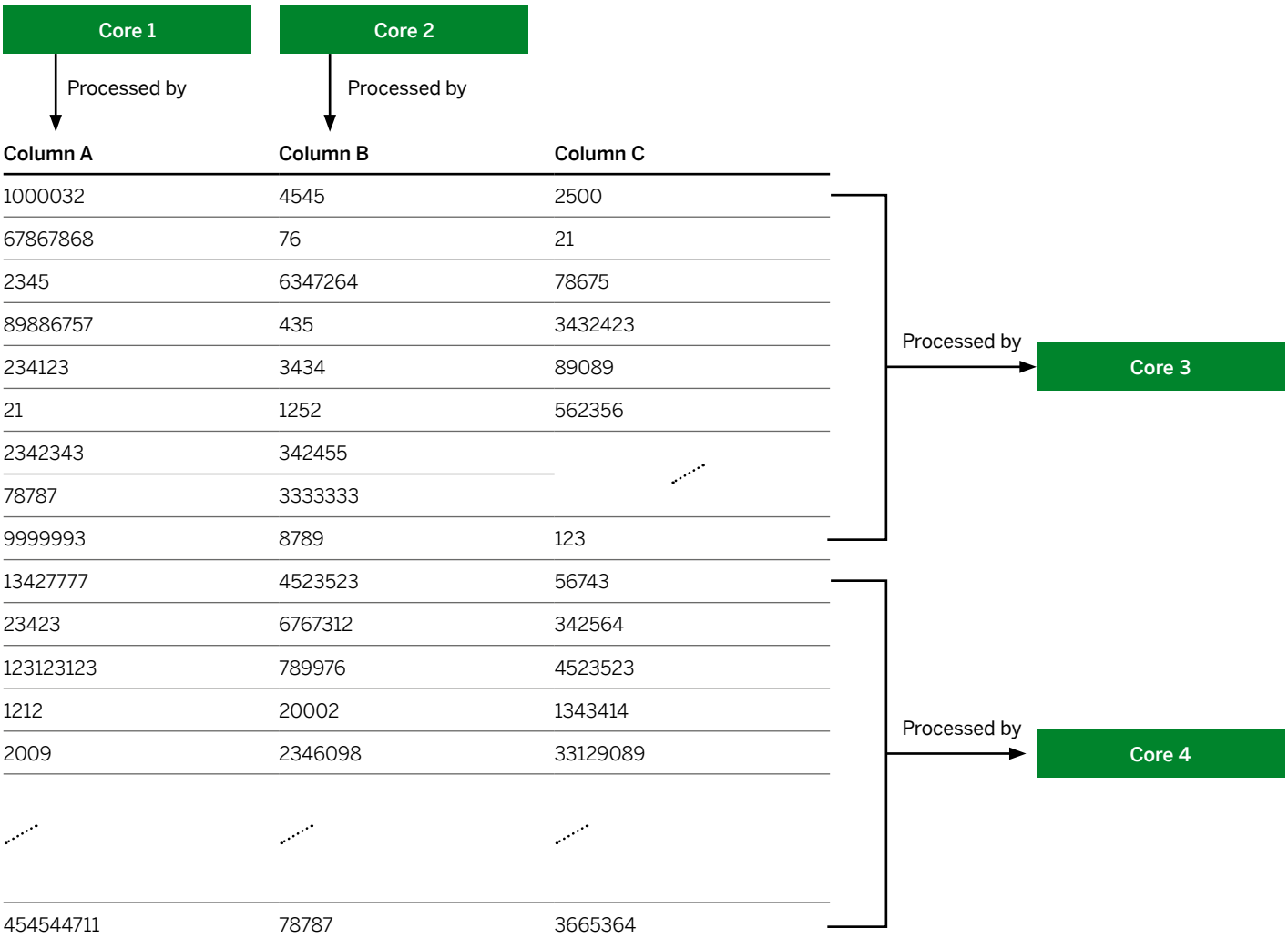
Parallelization. Column-based storage also makes it easy to execute operations in parallel using multiple processor cores. In a column store, data is already vertically partitioned. That means operations on different columns can easily be processed in parallel. If multiple columns need to be searched or aggregated, each of these operations can be assigned to a different processor core. In addition, operations on one column can be parallelized by partitioning the column into multiple sections that are processed by different processor cores. Figure 11 shows both options. Columns A and B are processed by different cores while column C was split into two partitions that are processed by two different cores.

Higher performance for column operations. With columnar data organization, operations on single columns, such as searching or aggregations, can be implemented as loops over an array stored in contiguous memory locations. Such an operation has high spatial locality and can efficiently be executed in the CPU cache.

TEMPORAL TABLES

As mentioned above, SAP HANA supports temporal tables that allow queries against a historical state of the data. Write operations on temporal tables do not physically overwrite existing records. Instead, write operations always insert new versions of the data record into the database. The different versions of a data record have timestamps indicating their validity. Applications can tell SAP HANA that subsequent queries are to be processed against a historical view of the database.

Figure 11: Example for Parallelization in a Column Store



SAP HANA Benefits for Business Applications

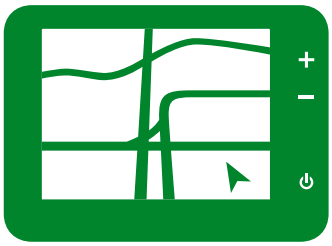
Traditional business applications use materialized aggregates to increase read performance. That means that the application developers define additional tables in which the application redundantly stores the results of aggregates, such as sums, computed on other tables. The materialized aggregates are computed and stored either after each write operation on the aggregated data or at scheduled times. Read operations read the materialized aggregates instead of computing them each time.

With scanning speed of several megabytes per millisecond, in-memory column stores make it possible to calculate aggregates on large amounts of data on the fly with high performance. This is expected to eliminate the need for materialized aggregates in many cases and thus eliminate up to 30% of the required tables.

In financial applications, different kinds of totals and balances are typically persisted as materialized aggregates for the different ledgers: general ledger, accounts payable, accounts receivable, cash ledger, material ledger, and so on. With an in-memory column store, these materialized aggregates can be eliminated as all totals and balances can be computed on the fly with high performance from accounting document items.

Eliminating materialized aggregates has several advantages:

- Simplified data model: With materialized aggregates, additional tables are needed, which make the data model more complex. In the financials data model for the SAP Business ByDesign™ solution, for example, persisted totals and balances are stored with a star schema. Specific business objects are introduced for totals and balances, each of which is persisted with one fact table and several dimension tables. With SAP HANA, all these tables can be removed if totals and balances are computed on the fly. A simplified data model makes development more efficient, removes sources of programming errors, and increases maintainability.
- Simplified application logic: The application either needs to update the aggregated value after an aggregated item was added, deleted, or modified, or special aggregation runs need to be scheduled that update the aggregated values at certain time intervals, such as once a day. By eliminating persisted aggregates, this additional logic is no longer required.
- Higher level of concurrency: With materialized aggregates, a write lock needs to be acquired after each write operation for updating the aggregate. This limits concurrency and may lead to performance issues. Without materialized aggregates, only the document items are written. This can be done concurrently without any locking.
- "Up-to-datedness" of aggregated values: With on-the-fly aggregation, the aggregate values are always up-to-date, while materialized aggregates are sometimes updated only at scheduled times.



SAP offers an incremental road map that enables your organization to safely explore the technology today, seize opportunities in side-by-side scenarios, and get prepared for the increasing rate of doing business.

Paradigm Shift in Approach to Data Management Technologies

SAP is leading a paradigm shift in the way we think about data management technologies. This rethinking affects how we construct business applications and our expectations in consuming them. Businesses that adopt this new technology will have the potential for sharpening their competitive edge by dramatically accelerating not only data querying speed but also business processing speed. SAP is working closely with customers, consultants, and developers for both immediate business gain and a longer-term perspective and road map.

As a line-of-businesses driver in a large enterprise, you may work in the short term with consultants to develop a business case for real-time reporting on operational data. You may focus on two or three specific business scenarios and deploy SAP HANA in a side-by-side, nondisruptive manner. A team of your IT experts will gradually gain experience working with SAP HANA, learn to exploit its flexibility, and further extend the utilization within your business-user community. Your organization will become in-memory-computing ready and gain immediate business benefits at the same time.

FIND OUT MORE

To learn more about SAP HANA, visit www.experiencesaphana.com and www.sap.com/hana.

ADDITIONAL READING

J. Krueger, M. Grund, C. Tinnefeld, J. Schaffner, S. Mueller, and A. Zeier, "Enterprise Data Management in Mixed Workload Environments" (Hasso Plattner Institute for IT Systems Engineering, 2009) http://ares.epic.hpi.uni-potsdam.de/apps/static/papers/2009_JK_Enterprise_Data_Management_in_Mixed_Enviroments.pdf

H. Plattner, "A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database" (SIGMOD'09, June 29–July 2, 2009) http://www.sigmod09.org/images/sigmod1ktp-plattner.pdf

H. Plattner, "Enterprise Applications – OLTP and OLAP – Share One Database Architecture" (Hasso Plattner Institute for IT Systems Engineering, 2010) http://epic.hpi.uni-potsdam.de/pub/Home/InMemoryDataProcessing2010/ShareOneDB.pdf

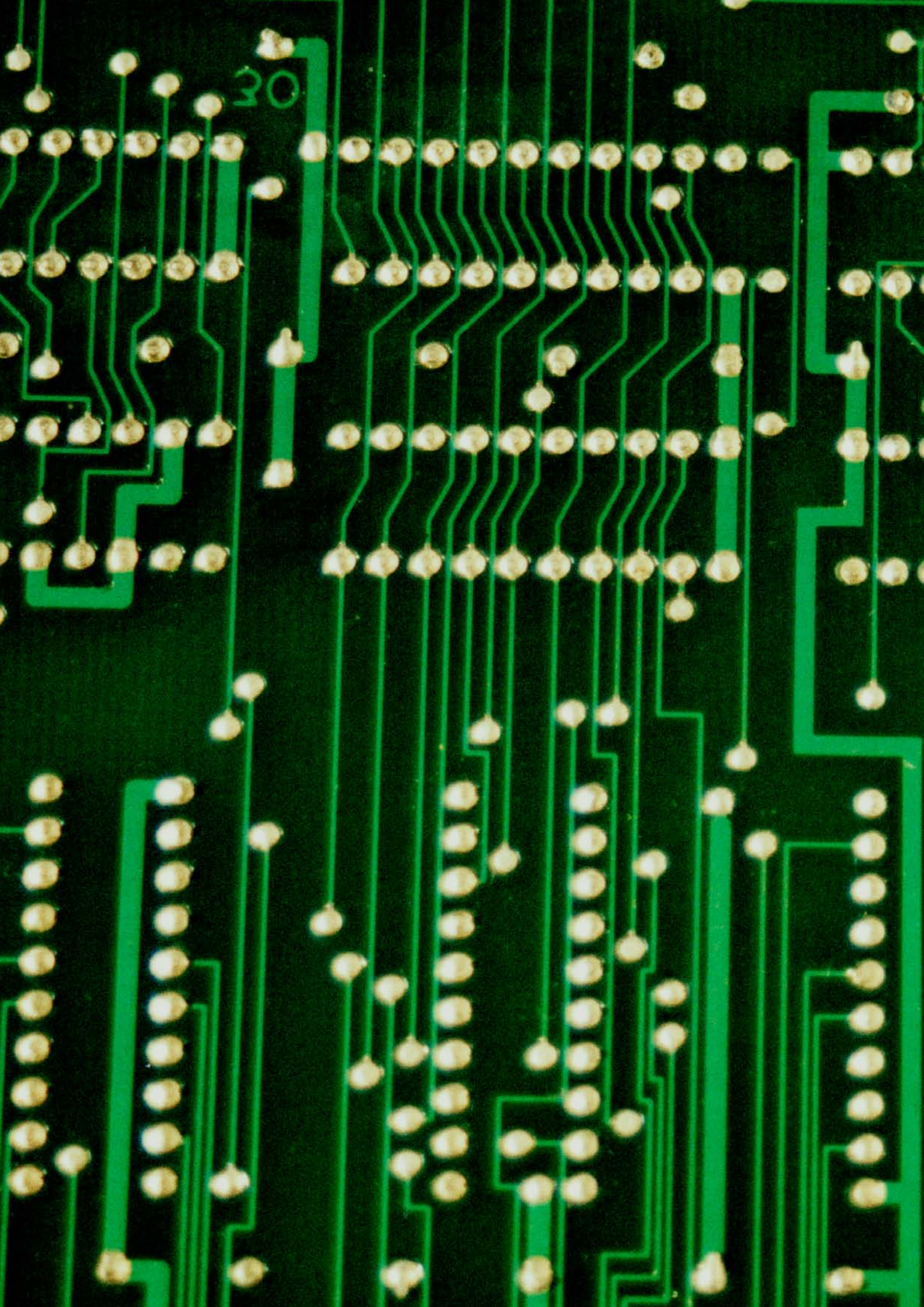
H. Plattner and A. Zeier, *In-Memory Data Management: An Inflection Point for Enterprise Applications* (Springer-Verlag, April 2011)

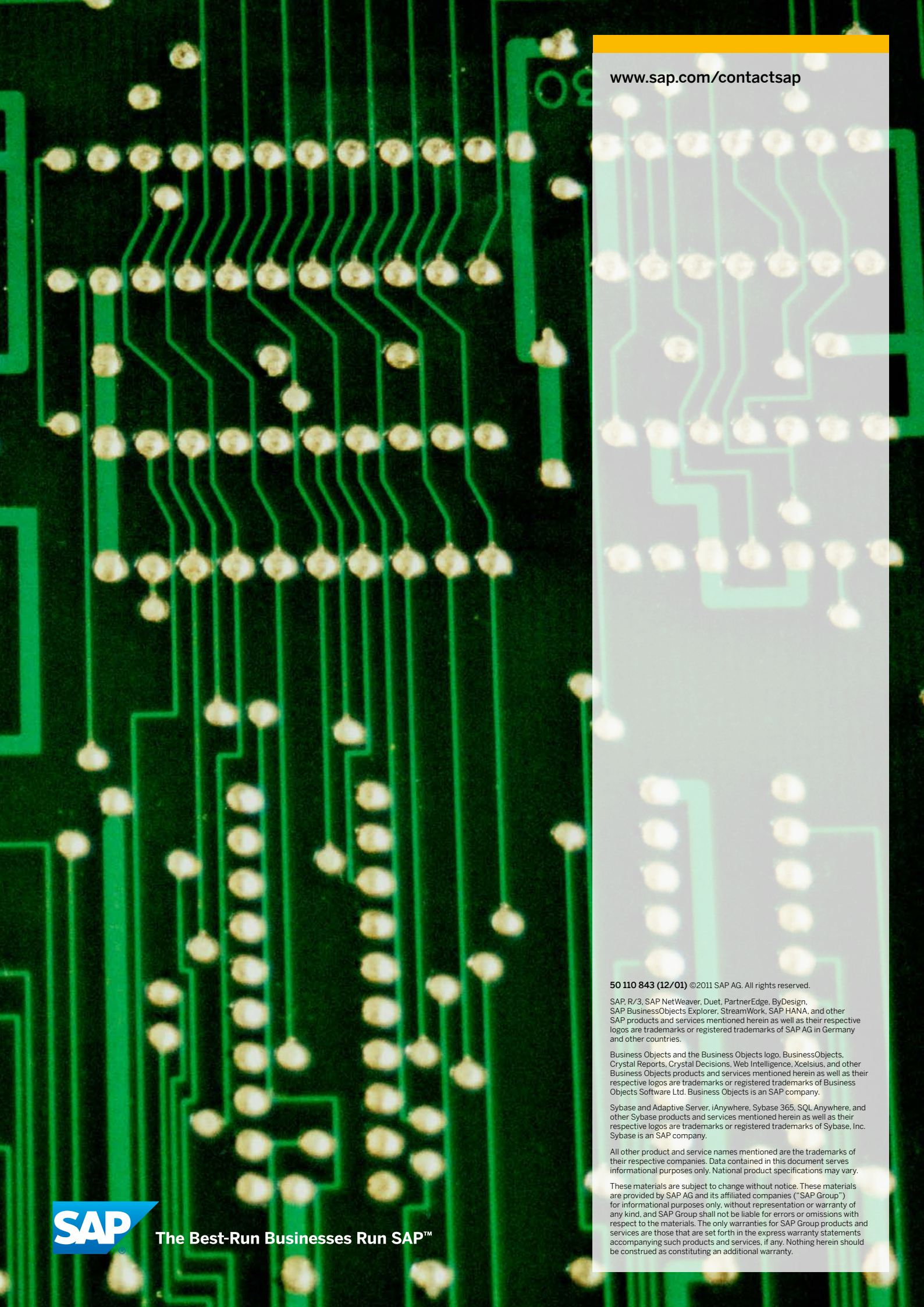
"Analyzing Business as It Happens: SAP In-Memory Appliance Software (SAP HANA™) runs on the Intel® Xeon® processor to generate superior, real-time business intelligence" (Intel|SAP, April 2011) http://www.intel.com/en_US/Assets/PDF/whitepaper/mc_sap_wp.pdf

D. Mettica and S. Lucas, "Prepare for the Quantum Leap in Real-Time Analytics. How in memory analytics is going to change everything about your enterprise" (IBM|SAP, June 2011) http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/d023f28a-6f9b-2e10-ec83-d84e518c3629?QuickLink=index

FOOTNOTES

1. Source: Plattner, Hasso and Zeier, Alexander, *In-Memory Data Management: An Inflection Point for Enterprise Applications* (Springer-Verlag, April 2011).
2. Ibid.
3. Restricted subset of C++ used internally at SAP.
4. Source: "Analyzing Business as It Happens: SAP In-Memory Appliance Software (SAP HANA™) runs on the Intel® Xeon® processor to generate superior, real-time business intelligence" (Intel|SAP, April 2011).
5. Ibid.


30



www.sap.com/contactsap

SAP

The Best-Run Businesses Run SAP™