How SAH eats 2 billion rows for lunch

Vince Kellen, PhD, Judy White, Kenny Valdivia, Mohamed Al-Omar, Brett Pollak, Kevin Chou JD, Dan Suchy

September 20, 2023

UC San Diego
INFORMATION TECHNOLOGY SERVICES

Agenda

1. Intro – What is the Student Activity Hub
2. SAP HANA basics – in-memory, columnar store, parallel architecture, heavy compression
3. SAP HANA optimizations
4. SAH architecture basics –activity tables, 4 view levels, ETL vs EL
5. SAH architecture performance optimizations – view materializations, table partitioning
6. Questions

SAH: A mission-driven, multi-institution collaboration

MISSION

Advance the state of student data management and student analytics in order to achieve our institutional goals, as diverse as they may be, while protecting institutional autonomy and control over all data.

PROBLEM

The SAH tackles the student data management data and analysis problems directly, giving control back to the institution. Think of SAH as a rich and high performance 'transmission.' You can drive it anywhere you like.

SOLUTION

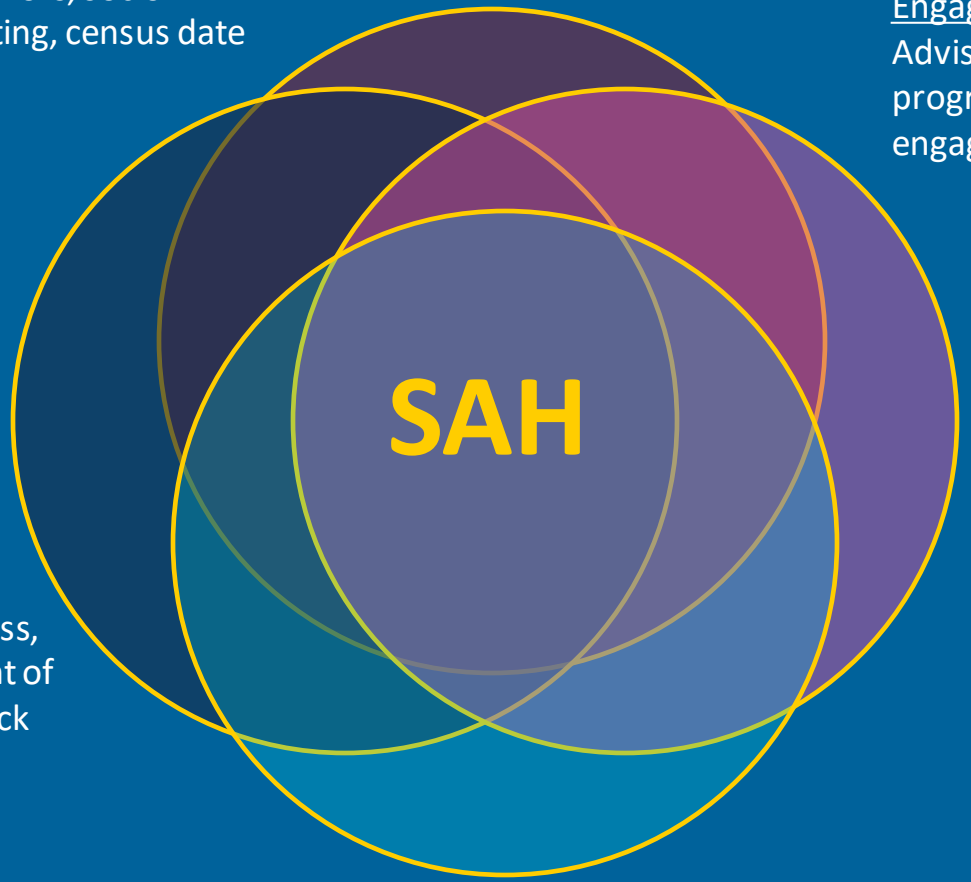
SAH allows for the merging of all kinds of data in one solution. Each institution has its own high-speed, in-memory server environment. With its security, scalability and sophistication, we can integrate any and all student data.

OUTCOME

The goal is modest. We want help institutions who might to leverage a common, but easily tailored or customized solution. Our goal is not to "sell" large numbers of SAH. We just want to make a difference where we can and collaborate with peers.

The Student Activity Hub (SAH) can support various needs

Institutional analytics:
Graduation rates, retention rates, enrollments, demographic, lists of majors/minors, socio-economic analysis, IPEDS reporting, census date frozen reporting etc.



Engagement analytics:
Advising interactions, co-curricular activities, degree progress tool use, mobile app interactions, student engagement impact on progression or retention, etc.

Operational analytics:
Entrance test scores, satisfactory progress, term and course grades, commencement of academic activity, financial aid, bottleneck course, major/minor switching etc.

Learning analytics:
Course engagement, engagement by hour/day/week of term, submissions, within-course or between-course assignment grades, assignments overdue, discussion participation, clickstream, page views, video views, etc.

SAH collaboration values

- **Service** to the community. Help improve student data management in higher education
- Be **transparent** with everything, including pricing
- Be frugal and **affordable**. SAP HANA and SAH are extremely affordable for everyone!
- Use partners! We know we can't do it alone so we are **partner-friendly**
- Use **leading edge** and wickedly fast technology (SAP HANA)
- Embrace **data standards** (e.g., 1EdTech standards)
- Use a very rigorous, disciplined, and **clever software engineering** approach

SAH was designed to give institutions full control

SaaS or IaaS: You can establish the level of control you need. We can operate in a full SaaS or in a full IaaS mode and adjust fees as needed. Items of control include:

- Each institution gets its own environment: This enables full institutional control, easy tailoring, new data integrations. Each institution can control its own change management process, adding new views, new data
- Data integration platform: We use Apache Kafka, Apache NiFi, Go Anywhere and WSO2 API manager. Institutions are free to choose their own integration tools and operate them or let us do it for them
- Custom view construction: The core SAH views are easily 'forkable' enabling institutions to develop their own solutions. We can perform the customization work or the institution can. Either way! All views are 100% ANSI SQL (2016)
- New activity tables: Institutions are free to add their own activity tables (a type of data lake), provided they do not alter the delivered activity tables. Views can freely access data from delivered activity tables or institution customized activity tables
- Metadata management and daily operations: As views get created and modified, we have a metadata administrative console (AH-MAC) tool that enables 'materializations', controls API access for downstream applications, and creation of data groups (Group Builder). These two tools are available to institutions that want full control over their environment. These two tools are written in Python. Institutions can 'fork' their own tools, but will need to manage the change process for new server console tools themselves. Institutions can administer their environment or let us do it for them
- Report building: At the moment, SAH does require each institution to have a reporting strategy. We have a large collection of workbooks in Tableau and Cognos we make available

Student Activity Hub Partners

Our partners are an integral part of Student Activity Hub development and adoption

Moran Technology Consulting

NTT DATA

Slower, Inc.

SAP

ERP Associates

Instructure

InvenioLSI

SAP HANA BASICS: Key Differentiators

- **In-memory compute engine** – tunable, lower latency for instant access to huge data pools e.g.; memory data in 5 nanoseconds as compared to conventional databases that take 5 milliseconds.
- **Massively Parallel Processing** - designed to perform its basic calculations, such as analytic joins, scans and aggregations in parallel. Often it uses hundreds of cores at the same time, fully utilizing the available computing resources of distributed systems.
- **Column store** - column storage uses contiguous memory locations which eliminates the need for additional index structures. Storing data in columns is functionally similar to having a built-in index for each column. Column scanning speed of the in-memory column store and the compression mechanisms – especially dictionary compression – allow read operations with very high performance.
- **Compression** – efficient compression of data makes it less costly to keep data in main memory while speeds up searches and calculations. SAP HANA can deliver 4:1 compression ratios or higher.

See backup slides for details

Optimizations – How we bend the OPEX curve

- **Benchmarking** – total cost of ownership target versus comparable technologies for TCO targets to bend the cloud 1 to 4 cloud Capex/Opex ratio.
- **Workload Profiling** – profile system under load for query performance and operational duty cycle to establish baseline targets and tuning techniques for optimization e.g.; partitioning, data slicing, data overhead and "*budgetdust*".
- **Performance Profiling** – determining the right mix of memory, cache, disk and instance sizing to deliver optimal performance at affordable price with no disruption to user experience or refactoring the AH code base.
- **Data Tiering** – profiling the age of the business data to determine proper placement between hot (in-memory), warm (cache/SSD) and cold (disk) to match performance based on the "user defined" value of the data.

See backup slides for details

SAH view architecture

Activity Table

All data streamed in via Apache NiFi / Kafka

Data replication services via SAP SDI

Activity hubs can have many activity tables

Activity table match common ingestion patterns

Three types of activity tables:

1. IoT style (e.g., Caliper event steams)
2. Table replication (e.g., Canvas Batch)
3. Table incremental replication (e.g., ERP transactional systems)

Base Views (BVs)

Marks and/or remove duplicates

If the activity table is incremental replication, removes deletes

Manages type conversions as needed

If needed, renames columns that reflect source system

Creates reusable column segments used more widely within IVs or CVs

Contains view-localized column segments that are not shared

Can reference other BVs

Intermediate Views (IVs)

Combines data from other BVs or IVs

Are typically either wide (repeated columns) or narrow (repeated rows instead of repeated columns)

Adds in more extended calculations, aggregations, complex where clauses, complex joins (e.g., business logic, or logic to enhance materialization, snapshot performance)

Can perform type conversions as needed

Can rename columns to user-friendly and highly conformed names

Curated Views (CVs)

Combines data from other BVs and IVs

Normally do not reference each other, but can if needed

Transforms column names into user-friendly names, replaces underscores in column names with spaces

Can filter data through WHERE or JOIN clauses

Can integrate Group Builder groups

Final Curated Views (FCVs)

Combines data from CVs, IVs or BVs as needed, fulfilling on an analysis 'vignette' or common need

Serve as Tableau and Cognos data sources

Normally used to service API requests via SAH API architecture

Types of “Curated views” of the data

Demographics
Residency, SAT/ACT and other entrance test scores, academic status, etc.

Enrollment
Enrollment counts by class, departments, schools, colleges, including course grades. Census and operational metrics

Major/Minors (wide and narrow)
Degrees, Programs, switching of majors, etc. Census and operational metrics

Student Statistics Per Term
Dozens of common student statistics, term-by-term for examining progression. Census and operational versions

Retention
Cohort, retention and graduation rates, etc. Census and operational metrics

Class and Section Stats Per Term
Dozens of class and section statistics, term by term for course and section planning, instructor load, etc. Census and operational metrics

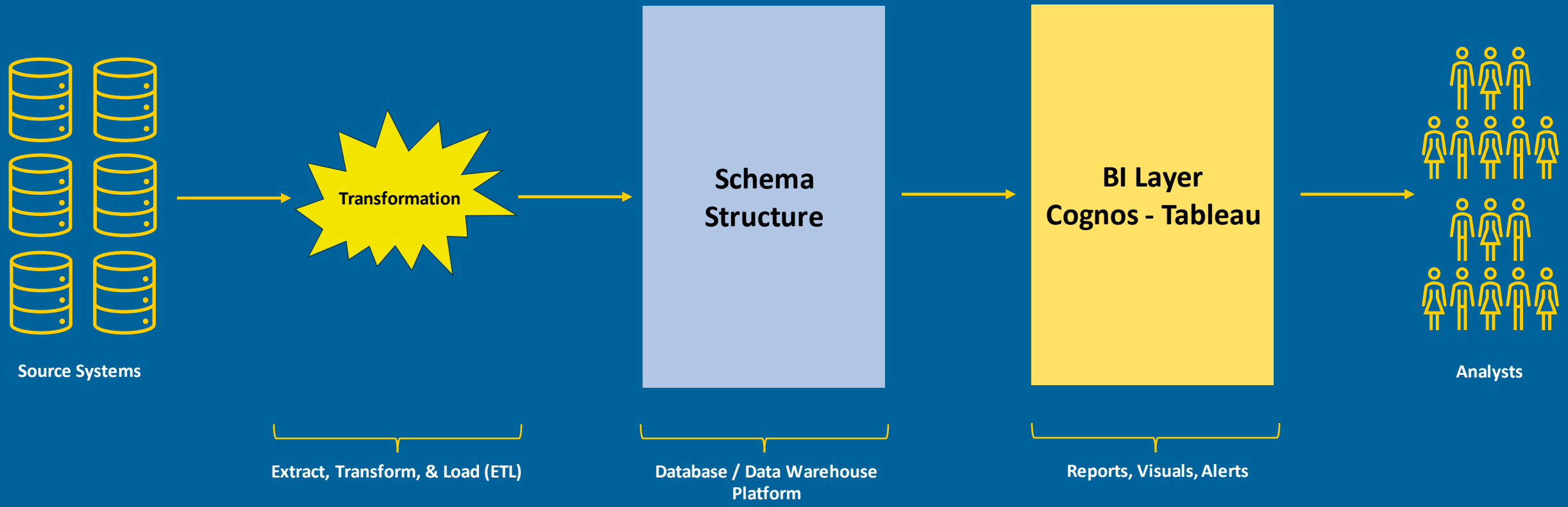
Admissions and financial aid
Applicants, applications, test scores, scholarships, financial aid forms and awards

Continuing education students (Extension, other)
Demographics, enrollment, credentials

LMS and other learning analytics
Canvas, OpenEdX, Kaltura. Canvas specific views and general learning event views

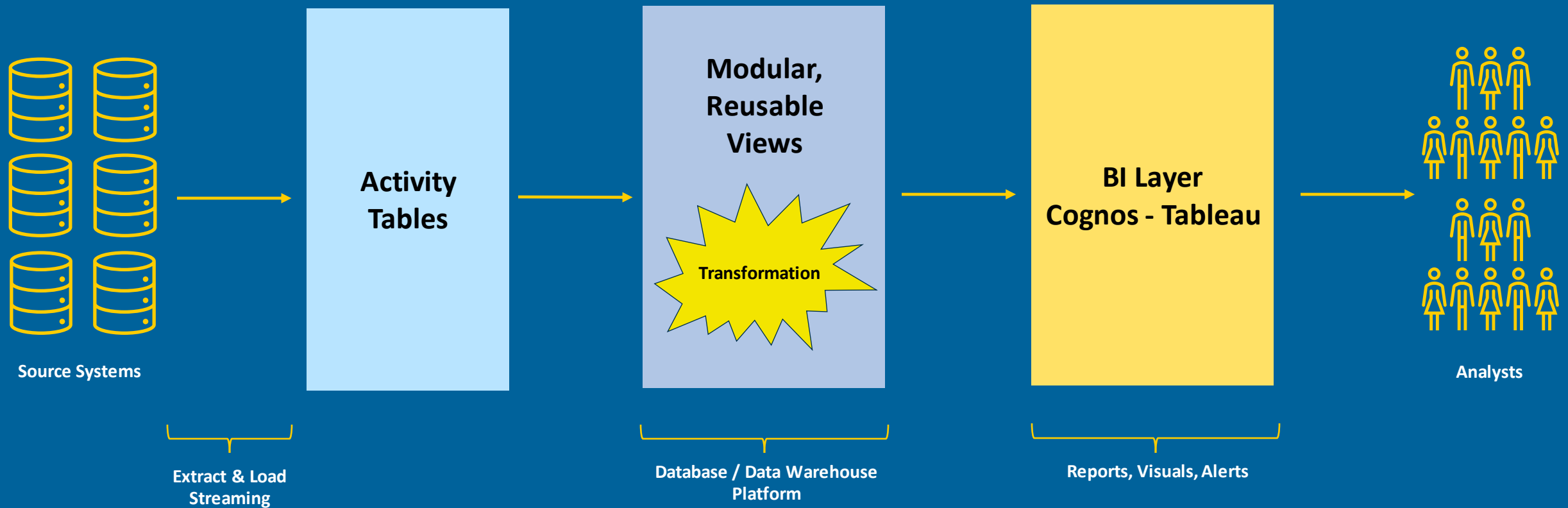
Institution extensions
Any source system data can be added and additional views created by the institution or partners for customized analytics

Data integration – typical approach

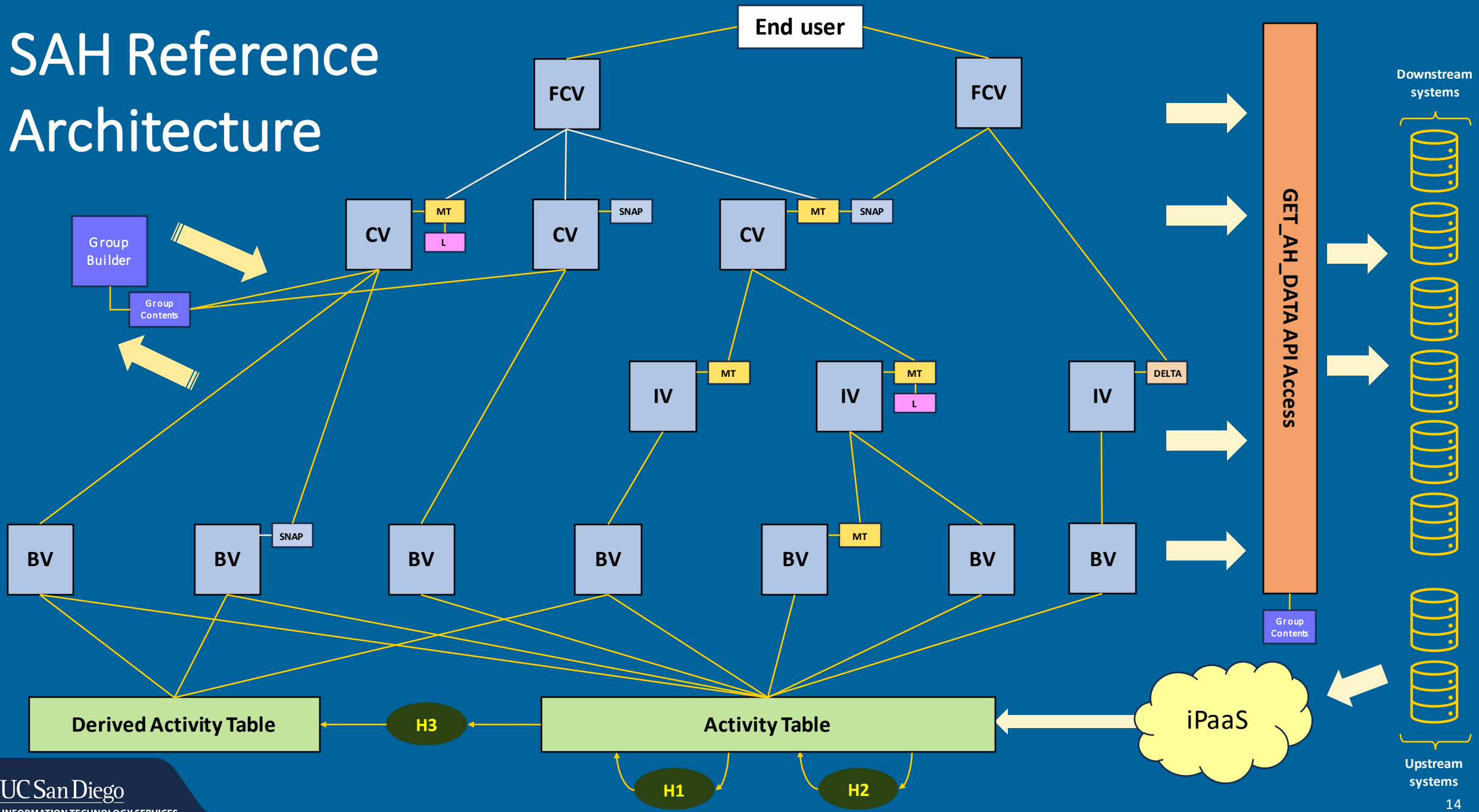


SAH simplifies data integration

Data integration is decoupled from view design and can be implemented ahead of actual view design and use. Activity tables serve as a data lake and can hold billions of rows of data



SAH Reference Architecture



View materialization

- Why?
  - To stabilize data for a period of time (e.g., for 1 minute, 3 hours, daily, or any unit of time).
  - To enhance view retrieval performance by ‘memorializing’ joins and calculations
- Materialization methods
  - Full, Full in slices, Full in half slices
  - Delta, Delta in slices, Delta with logs, Delta in slices with logs
  - Snapshots
- Materialization partitioning
  - We can add SQL optimization hints in any part of the materialization process, which can dramatically speed up some unique cases
  - Any materialization with slices can create slice partitions using any valid SQL where clause
  - Partitions can be established before hand from fixed table entries, useful for the largest of tables
  - A subset of partitions can be made active for ‘top slice’ materialization. Useful for old data that can never be updated. Active slices can be one (slightly faster) or can be any number of slices that meet a criteria (any valid SQL where clause)
- Delta Views (coming Q1 2024)
  - These are not the same as delta materializations. These are views that have, in each row, a column that indicates all the columns that have been changed since the prior row. These views are automatically created based on configuration items. Delta views can be materialized
  - These are useful for downstream application integration or BI visualization purposes to identify rows that have a change in a one columns (e.g., Last Name) or any combination of columns

Table hardening

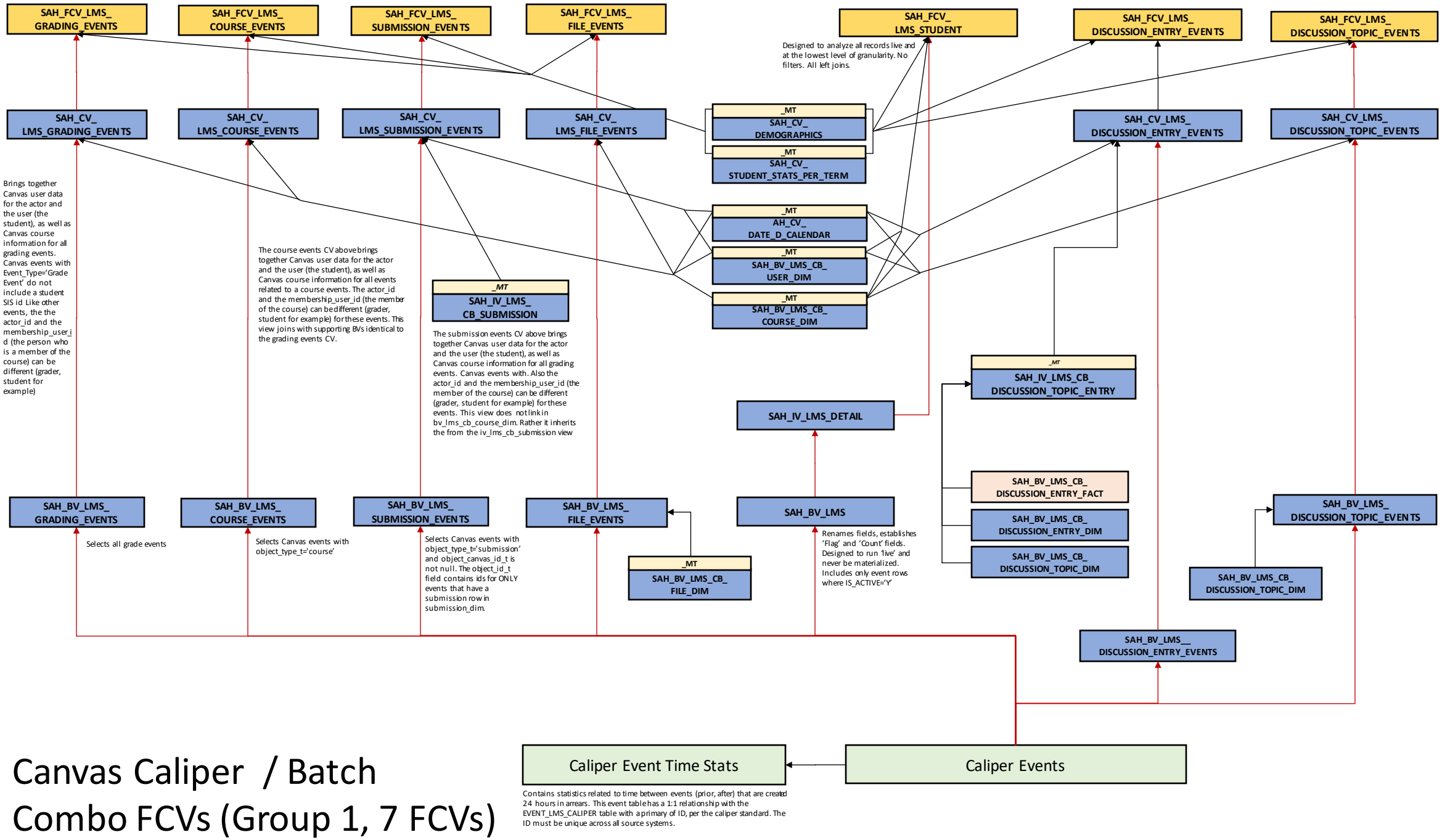
- Why?
  - When the SAH receives rows from source systems, some calculations or joins operations done once and never again. Caliper event data and other logging, CDC or IOT style data often meet this criteria
  - Some examples include:
    - Light transformations (splitting JSON, trimming columns)
    - Bringing additional values to the activity row via joins with other data
  - The hardening routines can be run at any schedule: seconds or minutes after row is received or hours
- Derived activity tables
  - These are tables designed to hold rows that have a perfect 1:1 match with the activity table, but can contain columns that have more computationally expensive calculations such as lag() or lead() functions
  - The same concept of write once and never again applies

Table partitioning

- Table partitioning has many performance benefits. Under the hood and hidden from the developer, HANA treats one table that is partitioned into 30 partitions as 30 separate tables
- This enables much more parallelism to queries that process it, providing significant speedups and load balancing, distributing tables over multiple hosts
- Each partition can contain 2 billion rows, each table can have 16,000 partitions, which means tables can have 32,000 billion rows! Upon startup, HANA only loads into memory partitions that are accessed. This typically allows is to keep only 50% of the data in memory.
- Many types of table partitioning
  - Round robin – rows are assigned automatically to different partitions
  - Range – rows are assigned to different partitions based on a range of values in a table, e.g., a partition for each year/month
  - Hash – rows are assigned different partitions based on a hash of one or more columns, typically for primary key distribution
  - Multi-level – 2 combinations of the three options above (hash-range, round robin-range, hash-hash, range-range)
  - Heterogeneous – more options for range-range and range-hash 2-level partitions
  - And more options!
- We typically use round-robin and range partitioning on SAH tables. We partition activity tables and larger MTs

Table tiering

- We can automatically or manually place data into a less expensive tier
- For the SAH, the Caliper activity table is a candidate. We maintain a copy of the original JSON received in a cold tier (Google Big Query) via our integration platform and remove it from HANA as it is redundant
- Older Caliper data (events 2 years or older) can be placed into a warm or cold tier. Our analysis shows that an SSD (warm) tier is the right blend of cost control and performance (although disk is 6-10x slower generally than in-memory)
- This tiering is transparent to the SAH developer and the end-user
- Table tiering can be applied to other activity tables that grow large and have rows that are infrequently accessed



Canvas Caliper / Batch Combo FCVs (Group 1, 7 FCVs)

Summary

- Real-time and non-real-time use case support. We can mix and max columns that real-time or stabilized (materialized) in a view. For example, we do this liberally in the Instructure Canvas LMS views with the Caliper Live Event data
- We have tremendous flexibility to configure both SAP HANA and the Student Activity Hub to handle the largest of data needs
- Query response time for very large tables is frequently in the 30-150 ms range. Real-time columns from our Caliper table (~2 billion rows) can be as fast for simpler queries of 150-1000 ms range for more complicated queries
- While end-users appreciate the speed, development time is greatly sped up as development can work on production-sized data environments
- Oh, BTW, HANA is a fantastic transaction environment as well that is used by the largest corporations in the world. We have some applications that write directly to HANA for both transaction management and analysis in the same platform

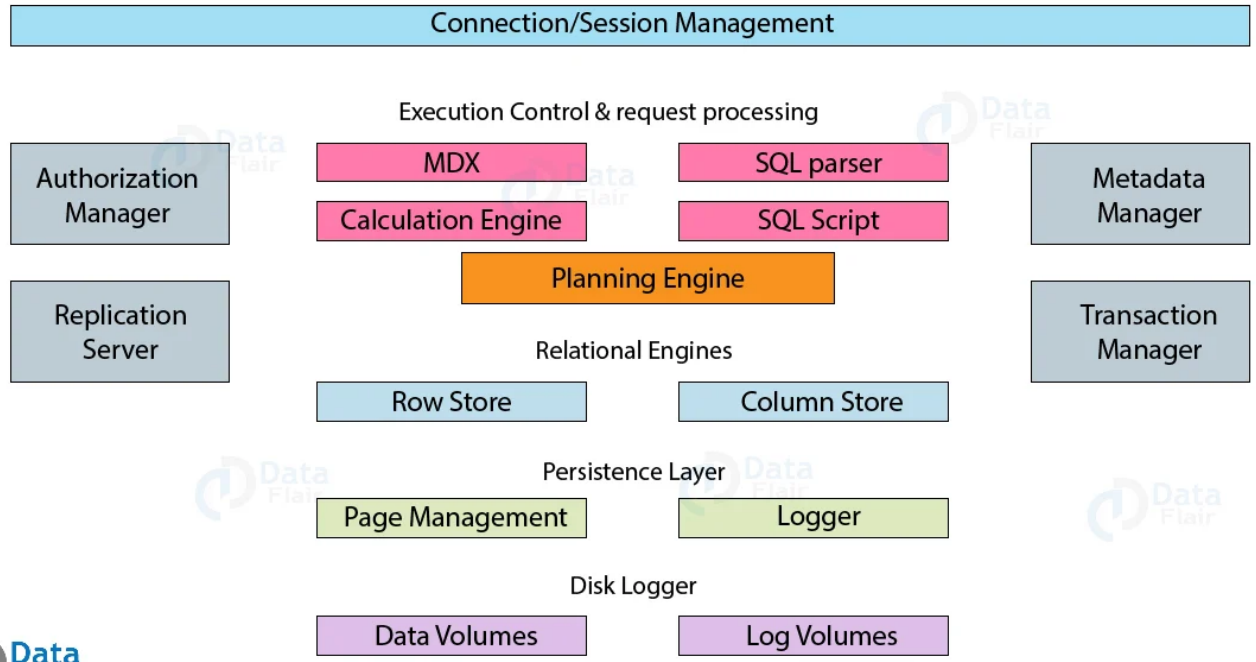
Q&A
What's on your mind?

Thank you!

Website: studentactivityhub.com

Backup Slides

SAP HANA IN-MEMORY COMPUTE ENGINE OVERVIEW

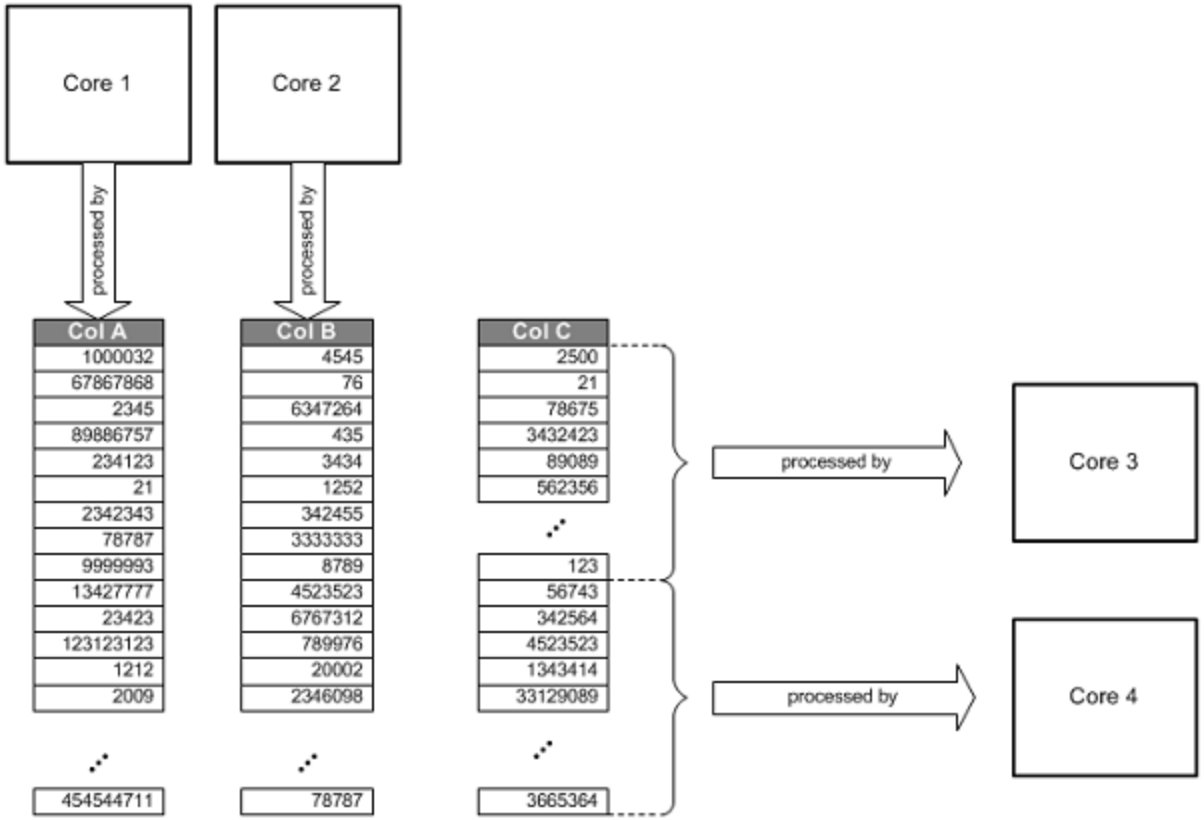


SAP HANA In-Memory Computing Engine (IMCE) /Index Server

- **Connection and Session Management:** Creates and manages the connection session between clients and SAP HANA database. These sessions establish for the user to communicate with the database using different query languages.
- **Authorization Manager:** This component allows only authorized users with a legible user ID. It makes sure that the users access, manipulate and share only that data which they are allowed to access.
- **Replication Server:** This is responsible for managing the replication of table data as well as metadata from the data source.
- **Planning Engine:** This creates an execution plan to apply on the database depending on the query sent to the computing engine.
- **Metadata Manager:** It stores all the information, i.e., metadata about the data table structures, views, data types, field descriptions, etc.
- **Transaction Manager:** It manages data transactions and keeps track of Commits and Rollbacks.
- **Request Processing and Execution Control:** This component receives queries or requests from client applications and directs it towards the respective component in the SAP HANA environment. It consists of an SQL Parser, SQL Script, MDX and Calculation Engine.
- **SQL Processor:** It processes the incoming queries or **SQL statements** and manipulates (insert, delete, update) data accordingly.
- **Persistence Layer and Disk Logger:** In in-memory computing, RAM stores data which makes it volatile (can erase due to system malfunctioning). Thus, the persistence layer is responsible for taking data backups periodically and store it permanently. It is known as **Savepoints** and by default, the savepoint frequency is in every 5 minutes. The data stores as log volumes and data volumes.

Source: https://data-flair.training/blogs/sap-hana-in-memory-computing/

SAP HANA MASSIVELY PARALLEL PROCESSING (MPP)



- SAP HANA was designed to perform its basic calculations, such as analytic joins, scans and aggregations in parallel. Often it uses hundreds of cores at the same time, fully utilizing the available computing resources of distributed systems.
- With columnar data, 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. With row-oriented storage, the same operation would be much slower because data of the same column is distributed across memory and the CPU is slowed down by cache misses.
- Compressed data can be loaded into the CPU cache faster. This is because the limiting factor is the data transport between memory and CPU cache, and so the performance gain exceeds the additional computing time needed for decompression.
- Column-based storage also allows execution of operations in parallel using multiple processor cores. In a column store, data is already vertically partitioned. This means that 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 can be processed by different processor cores.

Source: https://help.sap.com/docs/SAP_HANA_PLATFORM/fc5ace7a367c434190a8047881f92ed8/e468034561fe4ee6bb85b00b38e14712.html

SAP COLUMN STORE

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

A database table is conceptually a two-dimensional data structure organized in rows and columns. Computer memory, in contrast, is organized as a linear structure. A table can be represented in row-order or column-order. A row-oriented organization stores a table as a sequence of records. Conversely, in column storage the entries of a column are stored in contiguous memory locations. SAP HANA supports both, but is particularly optimized for column-order storage.

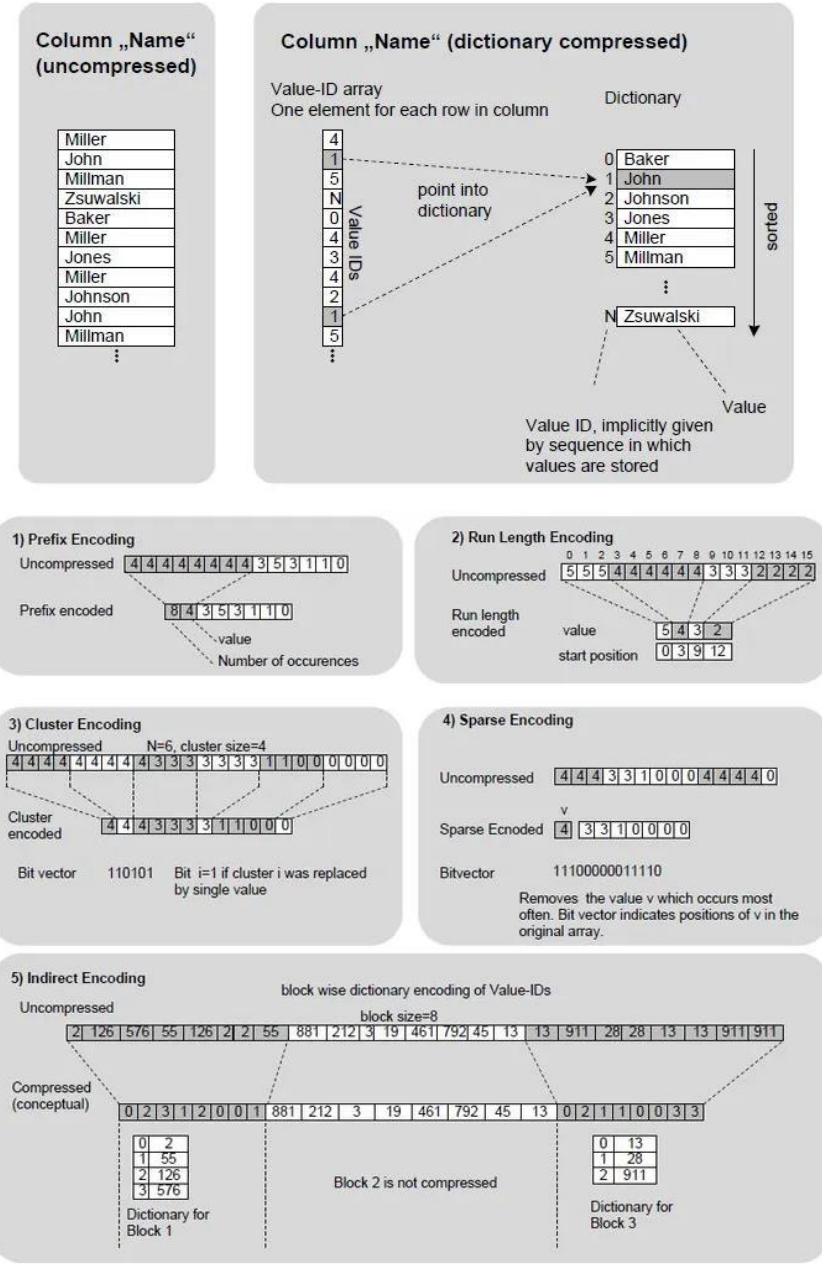
Columnar data storage allows highly efficient compression. If a column is sorted, often there are repeated adjacent values. SAP HANA employs highly efficient compression methods, such as run-length encoding, cluster coding and dictionary coding. With dictionary encoding, columns are stored as sequences of bit-coded integers. That means that a check for equality can be executed on the integers; for example, during scans or join operations. This is much faster than comparing, for example, string values.

Columnar storage, in many cases, eliminates the need for additional index structures. Storing data in columns is functionally similar to 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 – allow read operations with very high performance. In many cases, it is not required to have additional indexes. Eliminating additional indexes reduces complexity and eliminates the effort of defining and maintaining metadata.

Source: https://help.sap.com/docs/SAP_HANA_PLATFORM/fc5ace7a367c434190a8047881f92ed8/8c1fb4ff2f9640ee90e2dccea49c1739.html

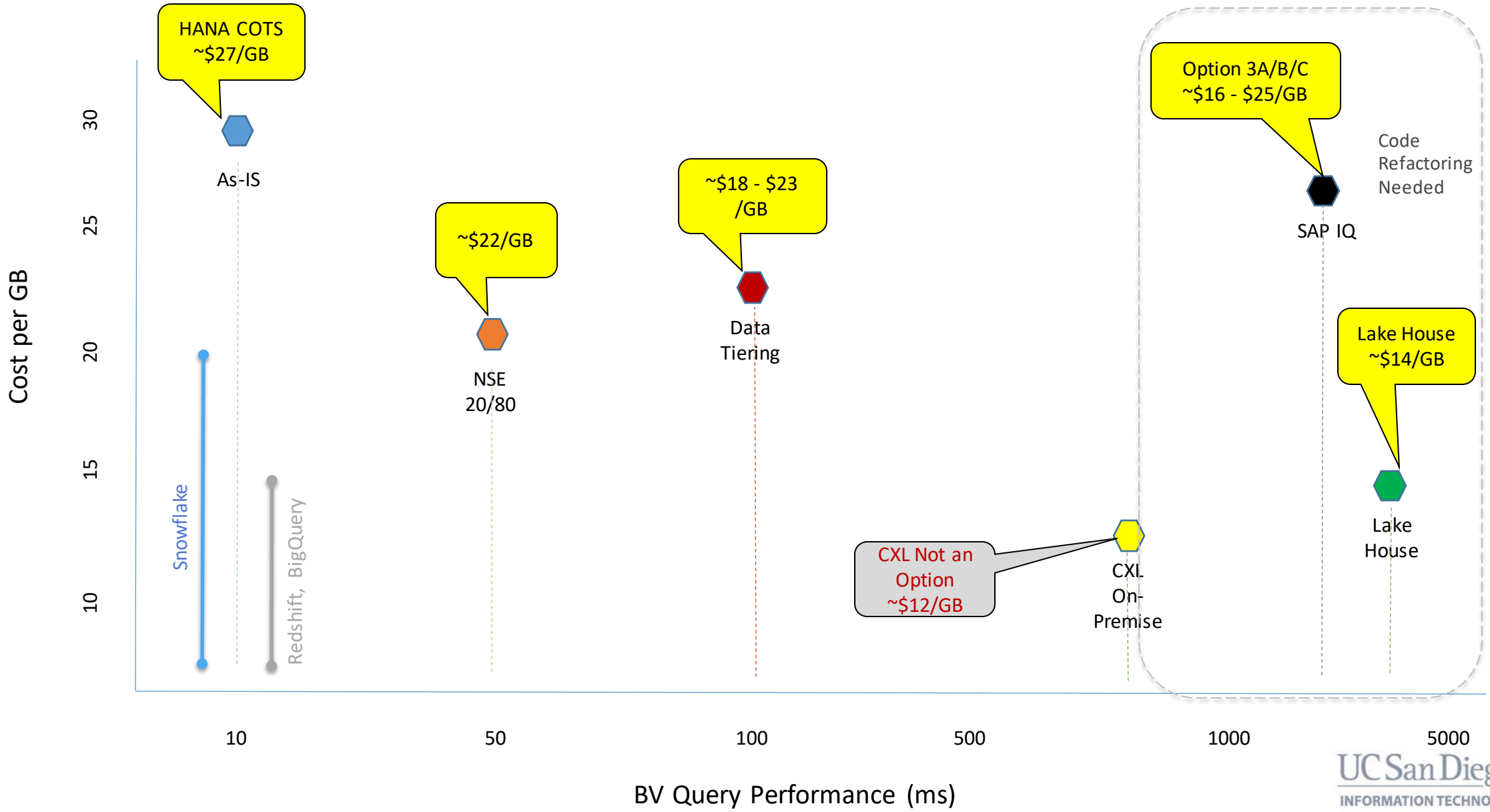
SAP HANA COMPRESSION OPTIONS

Compression type	Valid for	Details	Typical Scenario
Dictionary	main delta	The standard column store dictionary approach already provides a significant space reduction, because the distinct column values are mapped to value ID numbers which typically require much less space in memory. Dictionary compression is always used. Additionally any one of the other compression techniques mentioned below can be in place.	generally
Prefix encoding	main	Identical values at the beginning of the value ID array are stored only once, together with the number of occurrences.	single predominant column value
Run-length encoding	main	Consecutive identical value IDs are replaced with a single instance of this value ID and its start position.	several frequent column values
Cluster encoding	main	The value ID array is cut into clusters of 1024 elements. If a cluster contains only occurrences of a single value, the cluster is replaced by a single occurrence of that value.	several frequent column values
Sparse encoding	main	The most popular value is removed from the value ID array. A bit vector indicates at which positions the value was removed.	single predominant column value, value ID array not well clustered
Indirect encoding	main	The value ID array is cut into clusters of 1024 elements. If a cluster contains only a few distinct value IDs, a cluster specific dictionary is created, so that each value ID is represented with even fewer bits.	several frequent column values



Source: https://www.stechies.com/which-compression-types-exist/

BENCHMARKING: SAP HANA TCO COMPARATIVE ANALYSIS

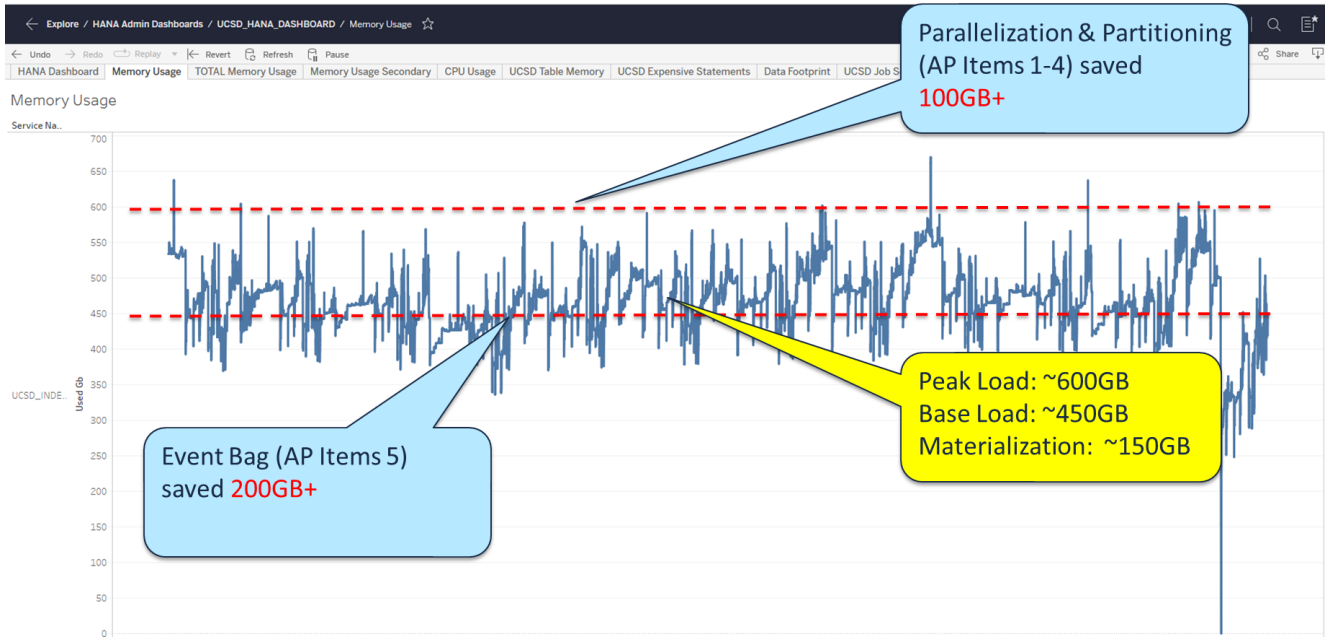


UC San Diego
INFORMATION TECHNOLOGY SERVICES

WORLKOAD PROFILING for ACTIVITY HUBS on SAP HANA

Prior State

Revised State



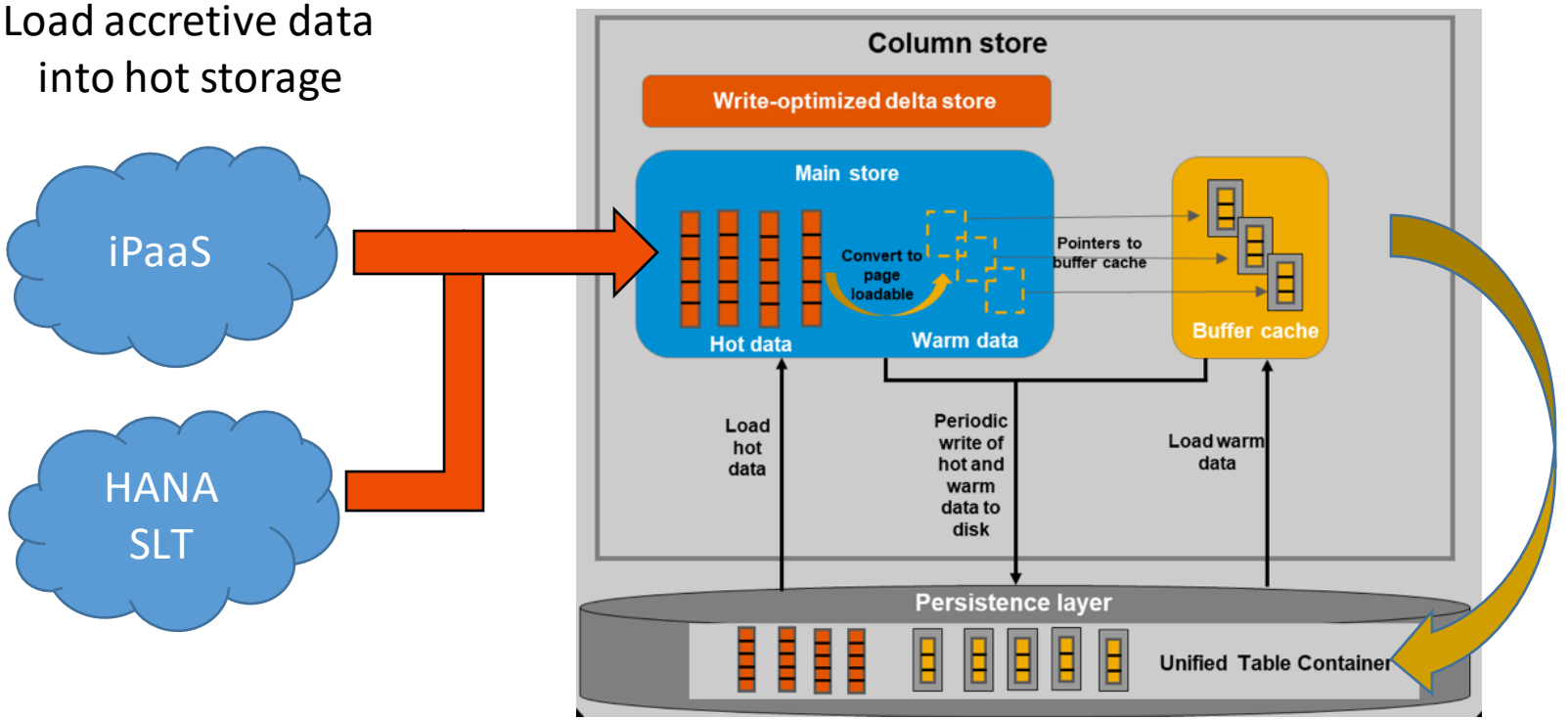
Continuous Action Plan

1. **Partition new activity tables** based on logical analysis of the data (date, subregion based partitions) for hot/warm/cold (HWC)
2. **Partition large always hot MTs** round robin 4-6 partitions to improve within-node parallelization, unless HWC is warranted
3. **Change materialization of large or slow processing MTs** to use full in half slices, with a date or modulo partition
4. **Use 'top slice' partitioning in MTs that have stale data**
5. **Remove event bag from LMS Caliper** activity table to a cold tier (1.4 TB, 200 GB Memory footprint)
6. **Test out a hardened column hash** within NiFi or AH hardening for improved delta materializations
7. **Apply the hash method for delta materialization** for MTs that have fewer changes to rows, reducing rows involved in DM
8. **Add warm and/or cold tiers** to the API log table and the LMS caliper activity table (500 GB data, memory?)
9. **Adjust the AH materialization procs** to include partitioning clause statements to MT table creation, and snag peak mem
10. **Test out the application of workload classes** to materialization runs, perhaps in conjunction with smart Delta Merge

UC San Diego
INFORMATION TECHNOLOGY SERVICES

PERFORMANCE PROFILING: BALANCING SAP HANA MEMORY, CACHE & DISK

Load accretive data into hot storage



Partition & write it to warm storage as it ages.

Buffer cache must grow as warm storage gets larger

Cost = ~5.25% performance reduction

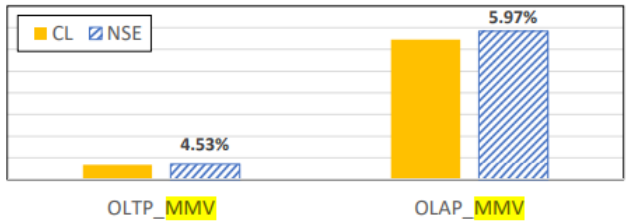


Figure 5: Normalized runtime (mixed workload)

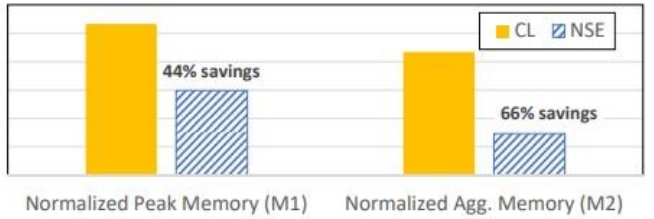
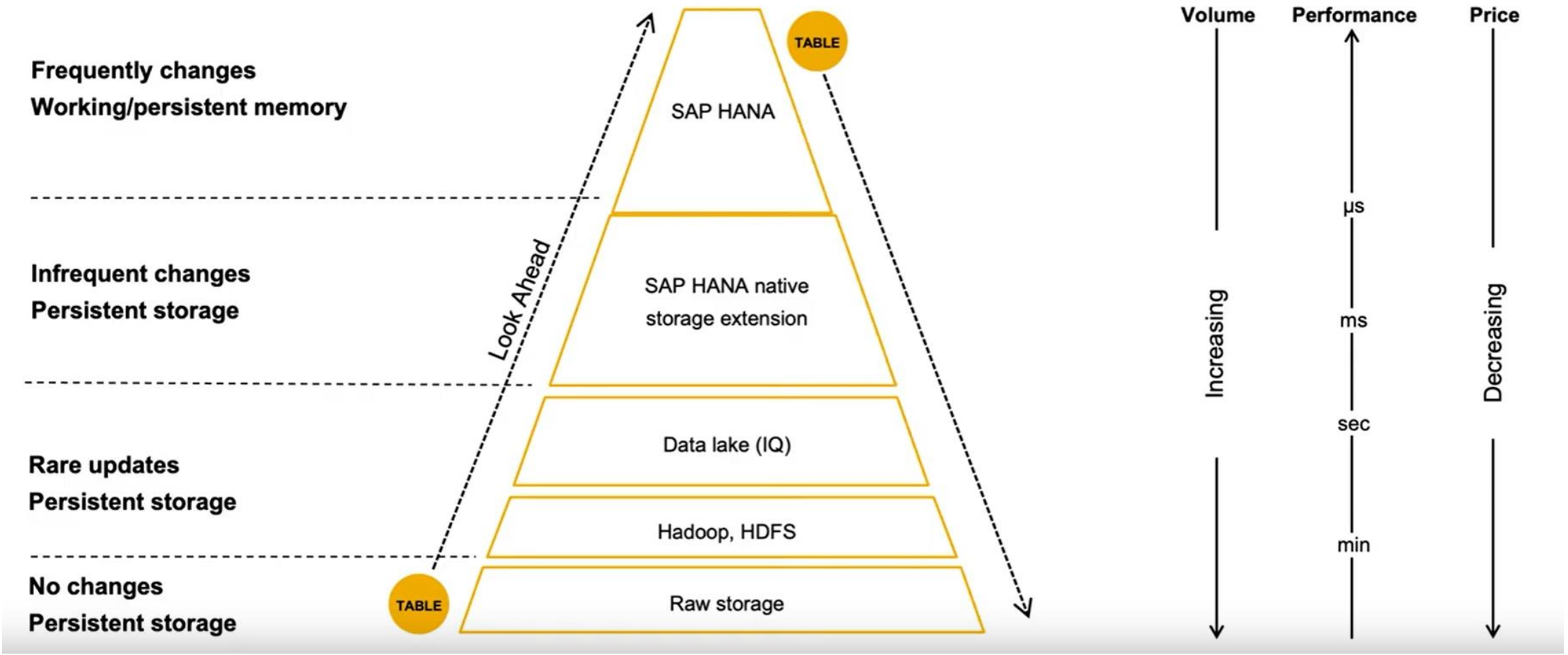


Figure 6: Normalized memory (mixed workload)

Benefit = ~55% memory reduction ($)

Source: Sherkat et al. 2019. Native store extension for SAP HANA. Proc. VLDB Endow. 12, 12 (August 2019), 2047–2058. https://doi.org/10.14778/3352063.3352123

DATA TIERING: INTELLIGENT PLACEMENT OF ACTIVITY DATA



Source: SAP