25-snowflake

- Our Product
  - The Snowflake Elastic Data Warehouse, or "Snowflake"
    - Built for the cloud
    - Multi-tenant, transactional, secure, highly available, elastic
    - Implemented from scratch (no Hadoop, Postgres etc.)
  - Currently runs on AWS and Azure
  - Serves tens of millions of queries per day over hundreds petabytes of data
  - 1000+ active customers, growing fast
- Motivation and Vision
  - Why Cloud?
    - Amazing platform for building distributed systems
      - Virtually unlimited, elastic compute and storage
      - Pay-per-use model (with strong economies of scale)
      - Efficient access from anywhere
    - Software as a Service (SaaS)
      - No need for complex IT organization and infrastructure
      - Pay-per-use model
      - Radically simplified software delivery, update, and user support
        - See "Lessons Learned"
  - Data Warehousing in the Cloud
    - Traditional DW systems pre-date the cloud
      - Designed for small, fixed clusters of machines
      - But to reap benefits of the cloud, software needs to be elastic!
    - Traditional DW systems rely on complex ETL (extract-transform-load) pipelines and physical tuning
      - Fundamentally assume predictable, slow-moving, easily categorized data from internal sources (CSTP, ERP, CRM, ...)
      - Cloud data increasingly stems from changing, external sources
        - Logs, click streams, mobile devices, social media, sensor data
        - Often arrives in schema less, semi-structured form (JSON, XML, Avro)
  - What about Big Data?
    - Hive, Spark, BigQuery, Impala, Blink
    - Batch and/or stream processing at datacenter scale
      - Various SQL/except front-ends
      - Increasingly popular alternative for high-end use cases
    - Drawbacks
      - Lack efficiency and feature set of traditional DW technology
        - Security? Backups? Transactions?
      - Require significant engineering effort to roll out and use
  - Our Vision for a Cloud Data Warehouse
    -
- Storage vs. Compute or the Perils of Shared-Nothing
  - Shared-nothing Architecture
    -
    - Tables are horizontally partitioned across nodes
    - Every node has its own local storage
    - Every node is only responsible for its local table partitions
    - Elegant and easy to reason about
    - Scales well for star-schema queries
    - Dominant architecture in data warehousing
      - Teradata, Vertica, Netezza...
  - The Perils of Coupling
    - Shared-nothing couples compute and storage resources
    - Elasticity
      - Resizing compute cluster requires redistributing (lots of) data
      - Cannot simply shut off unused compute resources => no pay-per-use
    - Limited availability
      - Maintenance, changes (failures, upgrades) significantly impact performance and may cause downtime
    - Homogeneous resources vs. heterogeneous workload
      - Ad-hoc loading, reporting, exploratory analysis

Architecture

Multi-cluster, shared data architecture

-
  - No data silos
    - Storage decoupled from compute
  - Any data
    - Native for structured & semi-structured
  - Unlimited scalability
    - Along many dimensions
  - Low cost
    - Compute on demand
  - Instantly cloning
    - Isolate production from DEV & QA
  - Highly available
    - 11 9's durability, 4 9's availability
-
-
  - Data Storage Layer
    - Stores table data and query results
      - Table is a set of immutable micro-partitions
    - Uses tiered storage with Amazon S3 at the bottom
      - Object store (key-value) with HTTPGET/PUT/LIST/DELETE interface
      - High availability, extreme durability (11 9)
    - Some important differences w.r.t. local disks
      - Performance (seek...)
      - No update in-place, objects must be written in full
      - But: can read parts (byte ranges) of objects
    - Strong influence on table micro-partition format and concurrency control
  - Table Files
    -
    - Snowflake uses PAX (Ailamaki01) aka hybrid columnar storage
    - Tables horizontally partitioned into immutable micro-partitions (~16 MB)
      - Updates add or remove entire files
      - Values of each column grouped together and compressed
      - Queries read header + columns they need
  - Other Data
    - Tiered storage also used for temp data and query results
      - Arbitarily large queries, never run out of disk
      - New forms of client interaction
        - No server-side cursors
        - Retrieve and reuse previous query results
    - Metadata stored in a transactional key-value store (not S3)
      - Which table contents of which S3 objects
      - Optimizer statistics, lock tables, transaction logs etc.
      - Part of Cloud Services layer (see later)
  - Virtual Warehouses
    - warehouse = Cluster of EC2 instances called worker nodes
    - Pure compute resources
      - Created, destroyed, resized on demand
      - Users may run multiple warehouses at same time
      - Each warehouse has access to all data but isolated performance
      - Users may shut down all warehouses when they have nothing to run
    - T-Shirt sizes: XS to 4XL
      - Users do not know which type or how many EC2 instances
      - ernsie and pricing can evolve independent of cloud platform
  - Worker Nodes
    - Worker processes are ephemeral and idempotent
      - Worker node forks new worker process when query arrives
      - Do not modify micro-partitions directly but queue removal or addition of micro-partitions
    - Each worker node maintains local table cache
      - Collection of table files i.e. S3 objects accessed in past
      - Shared across concurrent and subsequent worker processes
      - Assignment of micro-partitions to nodes using consistent hashing, with deterministic ideality
  - Execution Engine
    - Columnar (MonetDB, C-Store, many more)
      - Effective use of CPU caches, SIMD instructions, and compression
    - Vectorized (Zukowski05)
      - Operators handle batches of a few thousand rows in columnar format
      - Avoids materialization of intermediate results
    - Push-based (Neumann11 and many before that)
      - Operators push results to consumers instead of pulling (no Volcano iterators)
      - Removes control logic from tight loops
      - Works well with DAG-shaped plans
    - No transaction management, no buffer pool
      - But, most operators (join, group by, sort) can spill to disk and recurse
  - Cloud Services
    - Collection of services
      - Access control, query optimizer, transaction manager etc.
    - Heavily multi-tenant (shared among users) and always on
      - Improves utilization and reduces administration
    - Each service replicated for availability and scalability
      - Hard state stored in transactional key-value store
  - Concurrency Control
    - Designed for analytic workloads
      - Large reads, bulk or trickle inserts, bulk updates
    - Snapshot Isolation (SI) (Berenson95)
    - SI based on multi-version concurrency control (MVCC)
      - DML statements (insert, update, delete, merge) produce new table versions of tables by adding or removing whole files
      - Natural choice because table files on S3 are immutable
      - Additions and removals tracked in metadata (key-value store)
    - Versioned snapshots used also for time travel and cloning
  - Pruning
    - Database adage: The fastest way to process data? Don't.
      - Limiting access only to relevant data is important of query processing
    - Traditional solution: B+-trees and other indices
      - Poor fit for us: random accesses, high load time, manual tuning
    - Snowflake approach: pruning
      - ABA small materialized aggregates (Moerkotte98), zone maps (Netezza), data skipping (IBM)
      - Per file min/max values, #distinct values, #nulls, bloom filters etc.
      - Use metadata to decide which files are relevant for a given query
      - Smaller than indices, more load friendly, no user input required

Feature Highlights

- Pure SaaS Experience
  - Self Tuning & Self Healing
    -
    - Adaptive
    - Self-tuning
    - Do no harm!
    - Automatic
    - Default
    - Example: Automatic Skew Avoidance
      -
  - Support for various standard interfaces and third party tools
    - ODBC, JDBC, Python PEP-0249
    - Tableau, Informatica, Looker
  - Feature-rich web UI
    - Worksheet, monitoring, user management, usage information etc.
    - Dramatically reduces time to unbound users
  - Focus on ease-of-use and service map
    - No tuning knobs
    - No physical design
    - No storage growing
- Continuous Availabilt
  - Storage and cloud services replicated across datacenters
    - Snowflake remains available even if a whole datacenter fails
  - Weekly Online Upgrade
    - No downtime, no performance degradation!
    - Tremendous effect on pace of development and bug resolution time
  - Major seure- stateless services
    - All state is versioned and stored in common key-value store
    - Multiple versions of a service can run concurrently
    - Load balancing layer reuses new queries to new service version, until old version finished all its queries
- Semi-Structured and Schema-Less Data
  - Three new data types: VARIANT, ARRAY, OBJECT
    - VARIANT: holds values of any standard SQL type + ARRAY + OBJECT
    - ARRAY: offset-addressable collection of VARIANT values
    - OBJECT: dictionary that maps strings to VARIANT values
      - Like JavaScript objects or MongoDB documents
  - Self-describing, compact binary serialization
    - Designed for fast key-value lookup, comparison, and hashing
  - Supported by all SQL operators (joins, group by, sort...)
- Post-relational Operations
  - Extraction from VARIANTs using path syntax
    -
  - Flattening (pivoting) a single OBJECT or ARRAY into multiple rows
    -
- Schema-Less Data
  - Cloudera Impala, Google BigQuery/Dremel
    - Columnar storage and processing of semi-structured data
    - But: full schema required up front!
  - Snowflake introduces automatic schema inference and storage for schema-less data (VARIANT)
    - Frequently common paths are detected, projected out, and stored separately from other compressed columns in table file
    - Collect metadata on these columns for use by optimizer => pruning
    - Independent for each micro-partition => schema evolution
- Automatic Columnarization of semi-structured data
  -
- Schema-Less Performance
  -
- ETL vs. ELT
  - ETL = Extract-Transform-Load
    - Classic approach: extract from source systems, run through some transformations (perhaps using Hadoop), then load into relational DW
  - ELT = Extract-Load-Transform
    - Schema-later or schema-never: extract from source systems, load into Hadoop or JSON or XML, load into DW, transform those if desired
    - Decouples information producers from information consumers
  - Snowflake: ELT with speed and expressiveness of RDBMS
- Time Travel and Cloning
  -
  - Previous versions of data automatically retained
    - Same metadata as Snapshot Isolation
  - Accessed via SQL extensions
    - UNDROP recovers from accidental deletion
    - SELECT AT for point-in-time selection
    - CLONE (AT) to recreate past versions
- Security
  - Encrypted data import and export
  - Encryption of table data using NIST 800-57 compliant hierarchical key management and key lifecycle
    - Root keys stored in hardware security module (HSM)
  - Integration of S3 access policies
  - Role-based access control (RBAC) within SQL
  - Two-factor authentication and federated authentication
- Post SIGMOD '16 Features
  - Data sharing
  - Serverless ingestion of data
  - Reclustering of data
  - Spark connector with pushdown
  - Support for Azure Cloud
  - Lots more connectors

Lessons Learned

- Building a relational DW was a controversial decision in 2012
  - But turned out correct; Hadoop did not replace RDBMSs
- Multi-cluster, shared data architecture gave changes for big
  - Business units can provision warehouses on demand
  - Fewer data silos
  - Dramatically lower load times and higher load frequency
- Semi-structured extensions were a trigger for them expected
  - People use Snowflake to replace Hadoop clusters
- SaaS model dramatically helped speed of development
  - Only one platform to develop for
  - Every user running the same version
  - Bugs can be analyzed, reproduced, and fixed very quickly
- Users love "no tuning" aspect
  - But creates continuous stream of hard engineering challenges
- Core performance less important than anticipated
  - Elasticity matters more in practice
- Ongoing Challenges
  - SaaS and multi-tenancy are big challenges
    - Support tens of thousands of concurrent users, some of which do weird things, and need protection for themselves.
    - Metadata layer has become huge
    - Categorizing and handling failures automatically is hard, but
    - Automation is key to keeping operations lean
  - Lots of work left to do
    - SQL performance improvements, better skew handling etc.
    - Cloud platform enables a slew of new classes of features
  - Future work
    - Advisors
    - Materialized Views
    - Stored procedures
    - Data Lake support
    - Streaming
    - Time series
    - Multi-cloud
    - Global Snowflake
    - Replication