NewSQL Databases and Time Series Databases

Corso di Sistemi e Architetture per Big Data
A.A. 2018/19
Valeria Cardellini

Laurea Magistrale in Ingegneria Informatica

The reference Big Data stack

- High-level Interfaces
- Data Processing
- Data Storage
- Resource Management

Support / Integration
Relational database services

• RDBMS pros:
  – ACID transactions
  – Relational schemas (and schema changes without downtime)
  – SQL queries
  – Strong consistency

• RDBMS cons:
  – Lack of horizontal scalability, to hundreds or thousands of servers

NewSQL databases

• How to build a relational database service that is both ACID compliant and horizontally scalable?
  – i.e., how to make ACID scale?

• NewSQL: a class of modern RDBMSs
• Goals
  – Provide the same scalable performance of NoSQL systems for OLTP read-write workloads while maintaining ACID guarantees of traditional RDMSs
  – Support SQL
NewSQL examples

- Google’s Spanner
  - Also available as cloud service in Google Cloud Platform: Cloud Spanner [https://cloud.google.com/spanner/](https://cloud.google.com/spanner/)
  - CockroachDB is a clone
- Google’s F1
  - Relational distributed transactional database built on top of Spanner
- VoltDB
  - And H-Store, its research prototype predecessor [http://hstore.cs.brown.edu](http://hstore.cs.brown.edu)
  - Developed by M. Stonebraker (2015 ACM Turing award)
- Clustrix
- NuoDB
- Note: most of them closed source

Replication in NewSQL

- Hot to scale? Multi-master or master-less schemes
  - Any node can receive update statements
    - Google Spanner
      - Uses Paxos state machine replication to guarantee that a sequence of commands will be executed in the same order in all the replica nodes
    - VoltDB
      - A transaction/session manager receives the updates, which are forwarded to all replicas and executed in parallel
Spanner: why

• Google’s motivations:
  – “We provide a database instead of a key-value store to make it easier for programmers to write their applications”
  – “We consistently received complaints from users that Bigtable can be difficult to use for some kinds of applications”

Spanner

• **Wide-area distributed multi-version** database
  – General-purpose transactions (ACID)
    • Including strong consistency
  – SQL query language (slightly modified)
  – Schematized semi-relational tables
    • A hierarchical approach to grouping tables that allows Spanner to co-locate related data into directories that can be easily stored, replicated, locked, and managed

• Running in production
  – Storage for Google’s ads data
  – Replaced a sharded MySQL database
Spanner: overview

- **Feature:** lock-free distributed read-only transactions
  - Lock-free: no need of locking to only read any data items
  - But of course lock on read/write transactions!
- **Property:** external consistency of distributed transactions
  - External consistency: strictest concurrency-control guarantees for transactions (more than linearizability)
    - System behaves as if all transactions were executed sequentially
  - In a **globally distributed system**
- **Implementation:** integration of concurrency control, replication, and 2PC
  - Correctness and performance
- **Enabling technology:** a new API Time called **TrueTime**
  - Used to generate monotonically increasing timestamps and assign them to transactions

Spanner: Google’s TrueTime (TT)

- **Distributed synchronized clock with bounded non-zero error**
  - It returns a time interval that is guaranteed to contain the clock’s actual time for some time during the call’s execution
  - TT relies on a well engineered tight clock synchronization available at all servers thanks to GPS clocks and atomic clocks
  - **Cons:** TT requires special hardware and a custom-build tight clock synchronization protocol, which is infeasible for many systems
    - Spanner is run over Google’s private global network (not over public Internet), which is very high throughput, global fiber optic network linking its data centers
Spanner: concurrency control

• Hybrid approach
  – Read-write transactions are implemented through read-write locks, but read-only transactions are lock-free

• Why is it possible?
  – To read without blocking writes, Spanner (and other database systems) keep multiple immutable versions of data, often called multi-version concurrency control (MVCC)
    • A write creates a new immutable version whose timestamp is that of the write’s transaction
    • A read at a timestamp returns the value of the most recent version prior to that timestamp, and does not need to block writes
  – That is, Spanner stores multiple versions of data, and a read transaction is basically a read at a “safe” timestamp
  – Proper timestamping is achieved by using TrueTime

VoltDB

• In-memory, partitioned, distributed, ACID-compliant database based on shared nothing architecture
  – Open source community edition

• How it began
  – Open source RDBMS ran on memory-based file system
    • Over 80% of time spent on page buffer management, index management, and concurrency management
    • Only 12% of time spent doing the real work
  – Lead to H-Store

• Features
  – Horizontal scale-out on commodity hardware
  – High concurrency thanks to data partitioning
  – Each partition is single-threaded
  – Durability and high availability: asynchronous and synchronous command logging, database snapshot, replication
VoltDB: partitioning

- Tables are automatically partitioned over multiple servers, and clients can call any server
  - Transparent distribution but the user can choose how to partition the table by specifying the partitioning column
  - If a table is partitioned, each time you insert a row into that table, VoltDB decides which partition the row goes into based on the value of the partitioning column

- Selected tables can be replicated over servers, e.g. for fast access to read-mostly data

VoltDB: concurrency control

- Alternative design based on two assumptions
  - Assumption 1: total available memory is large enough to store the entire data store
  - Assumption 2: all user transactions are short-lived and can be very efficiently executed over in-memory data

- All transactions are executed sequentially in a single-threaded, lock-free environment
Time series data base (TSDB)

• How to analyze DevOps monitoring, application metrics, sensor data from smart factories, smart cities, or smart vehicles?

**Time series databases (TSDBs)**
  – A possible solution, not the only one!

• Optimized for handling high-volume time series data
  – *Time series*: sequence of data points (arrays of numbers) indexed by time (a date time or a date time range), e.g.:
    • Stock prices (price curve)
    • Energy consumption (load profile)
    • Temperature values (temperature trace)

• Optimized for providing complex logic to analyze time series data
  – Queries for historical data, replete with time ranges and roll ups and arbitrary time zone conversions are difficult in DBMS

TSDB: overview

• Create, enumerate, update and destroy various time series and organize them in some fashion
  – Series may be organized hierarchically and have companion metadata
  – Provide basic calculations on a series as a whole (e.g., multiplying, adding, or combining various time series into a new time series)
  – Filter on arbitrary patterns (e.g., day of the week, low value, high value)
  – Provide additional statistical functions that are targeted to time series data
TSDB: some products

• Some open-source products
  – CrateDB https://crate.io
  – Chronix http://www.chronix.io
  – Graphite https://graphiteapp.org
    • Stores numeric time-series data and render graphs of this data on demand
  – InfluxDB https://www.influxdata.com
  – KairosDB https://kairosdb.github.io
    • Stores its time series in Cassandra
  – OpenTSDB http://opentsdb.net
    • Stores its time series in HBase
    • NoSQL key-value store optimized for time series data with masterless architecture (similar to Riak-KV)

InfluxDB

• Written in Go
• Supports high write loads and large data set storage
• Conserves space through downsampling
  – By automatically expiring and deleting unwanted data as well as backup and restore
• Provides easy-to-use SQL-like query language for interacting with data
• Provides simple, high performing write and query HTTP(S) APIs, e.g.:
  – To create a database
    curl -i -XPOST http://localhost:8086/query --data-urlencode "q=CREATE DATABASE mydb"
  – To write data
    curl -i -XPOST 'http://localhost:8086/write?db=mydb' --data-binary 'cpu_load_short,host=server01,region=us-west value=0.64 1434055562000000000'
InfluxDB: time series

- Data organized by time series, which contain a measured value, like “cpu_load” or “temperature”
- Time series have zero to many points, one for each discrete sample of the metric
- Points consist of:
  - time (a timestamp)
  - a measurement (e.g., “cpu_load”)
  - at least one key-value field (the measured value itself, e.g. “value=0.64”, or “temperature=21.2”)
  - and zero to many key-value tags containing any metadata about the value (e.g. “host=server01”, “region=EMEA”, “dc=Frankfurt”)

General format of points:

`<measurement>[,<tag-key>=<tag-value>...] <field-key>=<field-value>[,<field2-key>=<field2-value>...] [unix-nano-timestamp]`

Examples of points:

```
cpu,host=serverA,region=us_west value=0.64
payment,device=mobile,product=Notepad,method=credit
  billed=33,licenses=3i 1434067467100293230
stock,symbol=AAPL bid=127.46,ask=127.48
temperature,machine=unit42,type=assembly
  external=25,internal=37 1434067467000000000
```
InfluxDB data store

- A measurement is like a SQL table, where the primary index is **time**
- With respect to DBMS:
  - No need to define schemas up-front
  - Null values are not stored

- InfluxDB limitation
  - Horizontal scalability: clustered installation available only as enterprise product

InfluxDB stack

- Integrated with Telegraph, Chronograf and Kapacitor (**TICK stack**)
- To realize a MAPE control loop

See [https://www.influxdata.com/time-series-platform/](https://www.influxdata.com/time-series-platform/)
InfluxDB stack

- Telegraf: plugin-driven server agent for collecting and reporting metrics and events
  - Input plugins or integrations to source a variety of metrics
  - Output plugins to send metrics to a variety of other data stores, services, and message queues (InfluxDB, Graphite, OpenTSDB, Kafka, MQTT, …)

- Chronograf: administrative user interface and visualization engine
  - To build dashboards with real-time visualizations of data and to create alerting and automation rules

- Kapacitor: native data processing engine
  - To process both stream and batch data from InfluxDB
  - E.g., to perform specific actions (e.g., dynamic load balancing) based on alerts (e.g., above load threshold)

References
