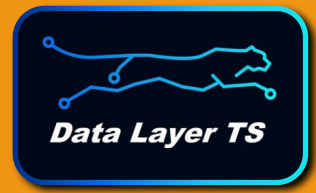
Data Layer TS

The data lake for time series data.

DLTS at One Glance



- Store & retrieve equidistant numerical time series data
- Manage and share millions of time series effortlessly
- High performance access
- Aggregation Functions
- Simple RESTful API

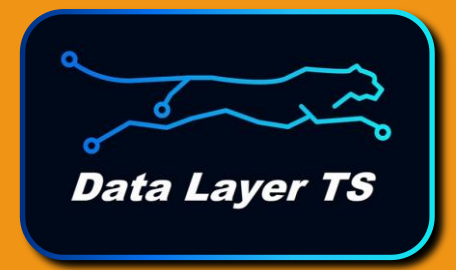
Sections


Functionality


Scalability


70.111
69.928
48.991
26.417
Use Cases

Functionality



Data Structure

Frequency

All timestamps inside a time series follow a specific frequency

Values

For every timestamp inside a time series a numeric value can be stored

Creation Timestamp

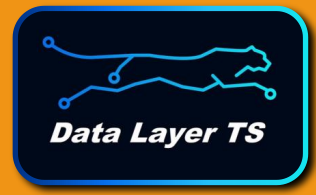
Optional: Store an additional creation timestamp for every value

Data Structure Examples

Timestamp	Max Price
2020-01-01T00:01:00Z	221.19912127
2020-01-01T00:02:00Z	225.49124811
2020-01-01T00:03:00Z	231.52498213

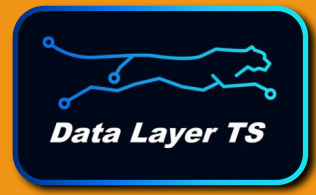
Timestamp	Temperature	Creation Timestamp
2020-01-01T01:00:00Z	18.446	2020-01-01T01:01:22Z
2020-01-01T02:00:00Z	19.647	2020-01-01T02:01:23Z
2020-01-01T03:00:00Z	19.146	2020-01-01T03:01:99Z

Core Functionality



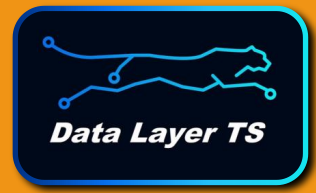
- Retrieve data from any specific time span from any specific time series
  - Raw data
  - Aggregated over time
  - Aggregated over multiple time series
  - Aggregated to another time resolution
- Insert or update data in specific time series for any specific time
  - Only insert missing values
  - Overwrite all values
  - Overwrite values where creation timestamp is equal or higher

Included Usability Features



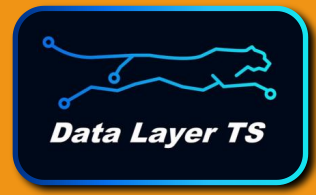
- Data models
  - Point model for easy integration
  - Vector model for high performance and matrix scenarios
- Error response model
  - Reference to the affected time series
  - Message containing problem details
- ACID compliant time series operations
  - Transactional time series manipulations

Included Usability Features



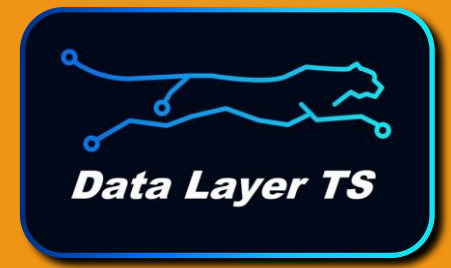
- Metadata management
  - Access rights to time series
  - Retention policies for time series
  - Get, create or delete time series
- Auto time series creation on first insert
- Auto deletion of expired data
- Auto background maintenance jobs

API Details

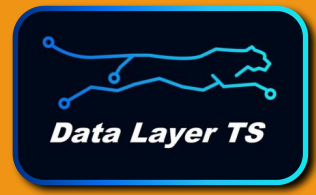


- HTTP RESTful API without query language
- Supported content formats
  - JSON
  - CSV
  - MessagePack
- Optional content compression
- Swagger API documentation

Scalability



Cloud Scalability



- Vertical VM scalability is supported
  - Horizontal disk scaling is enabled through an internal disk load balancer
- Public cloud providers support big VM sizes
  - Hundreds of cores
  - Multiple TB of RAM
- Horizontal VM scalability is not supported
  - Includes cluster communication overhead
  - A VM cluster costs the same as one vertically scaled VM with the total cluster performance

Vertical Scalability

Data Layer TS

More Cores
+ Parallel requests
+ Overall speed

More IOPS
+ Insert performance
+ Maintenance speed

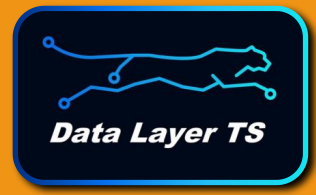
More RAM
+ In-memory data capacity
+ Buffer capacity

Faster Network
+ Throughput limit

Better SIMD Support
+ Aggregation speed

Higher Clock Rate
+ Overall speed

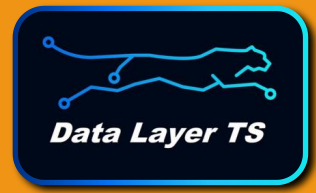
Scaling Disk Performance



- Horizontal scaling partitions with disk load balancing solution
  - Balances the load equally on all configured partitions

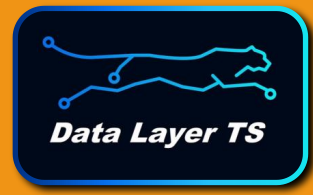
- Constant maximum insert performance through scalable binary file formats
  - Header files contain the recent part of a time series
  - History files contain the historical part of a time series
  - The content move from the header files into the history files runs decoupled from the user requests in the background maintenance

Scaling Memory Performance



- In-Memory database solution for time series content management
  - Content is stored in memory using a data structure optimized for fast parallel data access
  - Tracks the time series usage for inserts and selects
  - Keeps recently needed parts of time series when the in-memory capacity is reached
- Memory pooling solution for low memory pressure
  - Maximum internal buffer reuse

Load Testing



- Free load-testing tool
  - Source code available on GitHub
  - Can be used to find the right VM size for a specific workload
  - Continuous time series deliveries can be simulated
  - Scenarios can include all formats, models and aggregations

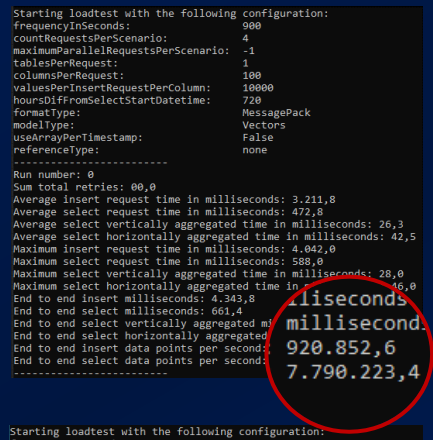
- Results show unmatched performance

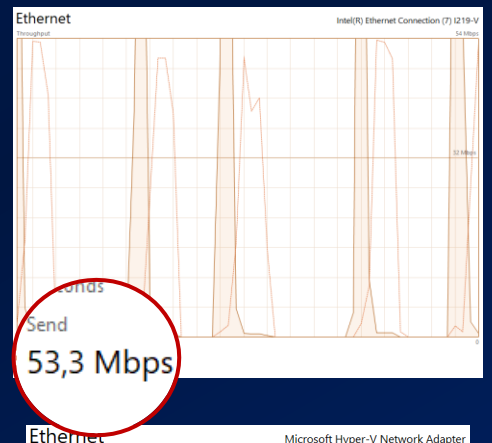
Load Testing Examples

Data Layer TS

Test client 1:

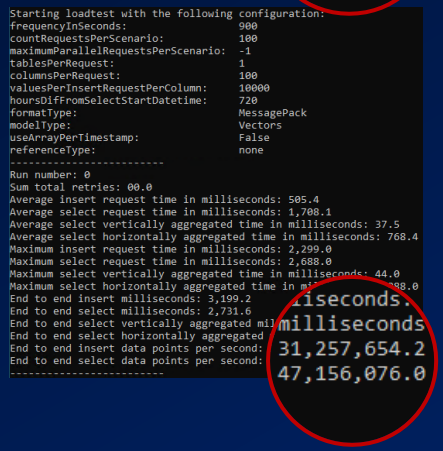
- 8 Cores
- 16 GB RAM
- ~1 Gbps download
- ~50 Mbps upload

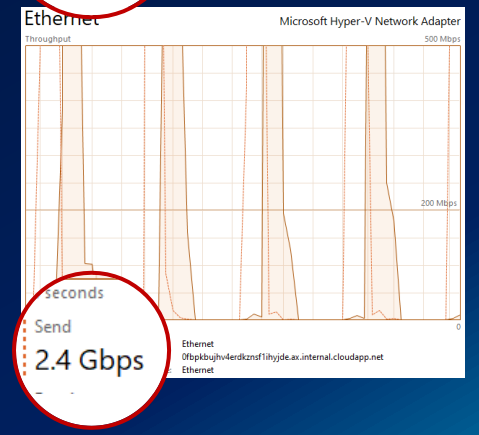




Test client 2:

- 8 Cores
- 16 GB RAM
- ~3.5 Gbps sync





Test server:

- 10 Cores
- 64 GB RAM
- ~2.5 Gbps sync
- 10.000 IOPS

Use Cases

Data Layer TS

Use Case Examples

Financial Data

Open-high-low-close prices

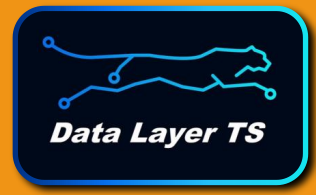
Weather Data

Grid based weather forecasts

IoT Data

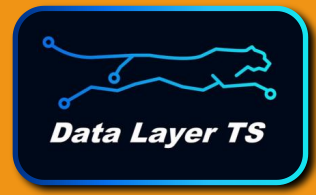
Sensor measurements

Financial Data Example



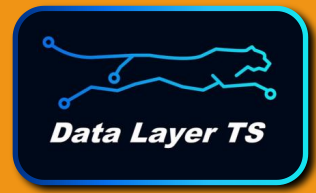
- Save data at the highest possible resolution
  - Open-high-low-close-volume values in separated time series
  - Highest available or needed frequency of aggregated trades
- Aggregate to any other time resolution dynamically
  - Use first, maximum, minimum, last and sum aggregations to aggregate all open-high-low-close-volume values to any time resolution in milliseconds
- Aggregate over any time span dynamically
  - Select first, maximum, minimum, last and sum aggregations from any large historical time period in milliseconds

Weather Data Example



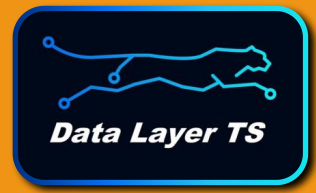
- Save data at the highest possible resolution
  - One time series per grid point per variable
  - Highest available frequency
- Always keep the most recent forecasts in the time series
  - Override old data based on the forecast timestamp of the data to ensure the most recent forecast for each time interval is stored
- Aggregate grid point time series dynamically
  - Aggregate hundreds of grid points inside a postal code and select the minimum, maximum and average value from all time series for each time interval in milliseconds

IoT Data Example



- Save data at the highest possible resolution
  - One time series for each sensor measurement
  - Highest available frequency
- Aggregate millions of historical data points dynamically
  - Select aggregations over one large time interval or change the resolution and get the timestamp of the maximum and minimum in addition to the maximum and minimum itself
- Aggregate thousands of sensor time series dynamically
  - Aggregate any cluster of sensors to the average time series of all contained sensor values
  - Get a time series reference out of the cluster to the time series, which has the maximum or minimum value stored for each time interval

Why choose DLTS?



- Data must be shared with many concurrent users
- High performance data access is needed
- Data should be monetized through API marketplaces
- Centralized time series store for all available series is needed
- Data lake for time series data is needed

Thank You

Data Layer TS - It’s about time.

API: https://datalayerts.com

Mail: info@clapsode.com