Time Series Database (InfluxDB)

CIS 6930 Advanced Databases

(Group 8)
Nagapandu Potti
Aman Raj Singh
Tarun Gupta Akirala
Rakesh Dammalapati
What is time series data?
<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Oct 25 - Close</td>
</tr>
<tr>
<td>Close Price</td>
<td>33.15</td>
</tr>
<tr>
<td>Change</td>
<td>+0.08</td>
</tr>
<tr>
<td>Change %</td>
<td>0.24%</td>
</tr>
<tr>
<td>Range</td>
<td>33.01 - 33.49</td>
</tr>
<tr>
<td>52 Week Range</td>
<td>29.52 - 36.43</td>
</tr>
<tr>
<td>Open</td>
<td>33.21</td>
</tr>
<tr>
<td>Volume</td>
<td>19.82M</td>
</tr>
<tr>
<td>Average</td>
<td>22.63M</td>
</tr>
<tr>
<td>Market Value</td>
<td>154.01B</td>
</tr>
<tr>
<td>Shares</td>
<td>4.65B</td>
</tr>
<tr>
<td>Beta</td>
<td>1.08</td>
</tr>
<tr>
<td>Industry</td>
<td>61%</td>
</tr>
<tr>
<td>P/E</td>
<td>14.29</td>
</tr>
<tr>
<td>Dividend</td>
<td>0.12/1.45</td>
</tr>
</tbody>
</table>

**Compare:** Enter ticker here

**Zoom:**


**Volume (mil / 2min):**

- 2.5
- 1.25

**Yearly Range:**

- **2012:**
- **2013:**

**Settings | Technicals | Link to this view**

*Volume delayed by 15 mins.*
What about...
“...order by some_time_col”
Why a database for time series?
Events

Measurements

Exceptions

Page Views

User actions

Commits

Things happening in time...
Billions of data points
Scale horizontally.
Example from DevOps

2000 servers, VMs, containers, or sensor units

200 measurements per server/unit

Every 10 seconds

= 3,456,000,000 distinct points per day
Sharding Data
(Usually requires application level code)
Existing Tools

RRDTool (metrics)

Graphite (metrics)

OpenTSDB (metrics + events)

Kairos (metrics + events)
Something missing...
InfluxDB

Written in Go

Self Contained binary

No external dependencies

Distributed
Features

HTTP native API to build on

Automatically clear out old data if we want - Data Retention

Continuous queries (for rollups and aggregation)

Process or monitor data as it comes in

Built in tools for downsampling and summarizing

Sharding data
Applications - IoT
Applications - Custom DevOps Monitoring

More Metrics and Monitoring Mean More Problems

Edge
- Mobile
- Sensors
- Web
- API

Applications
- Traditional
- SaaS
- PaaS
- CaaS
- Dev / QA

Network
- Server
- Network
- Storage

Infrastructure Mgmt.
- APM Tools (JVM, CLR...)
- Cloud Watchers
- Log Parsers

Legacy Petrolers
- Packet Analyzers

Synthetics / RUM
- Mobile & Crash Analytics
- API Analytics

Network
- Location 1
- Location 2
- ... Location N

MOM
- Aggregate
- Alerting
- Reporting

Service Desk
- BI & Reports
Applications - Real time Analytics

Real-Time Analytics Geolocation Application with InfluxDB

Web / Mobile Client
OpenLayers, SocketIO

Net

HTTP Accelerator
Varnish (80)

WebSockets
Node.js + SocketIO

App Tier
Nginx, Gunicorn (Django), TileStache, Memcached, Mapnik

Messaging
Celery + RabbitMQ

Visualization
Grafana

Data Storage / Analytics
InfluxDB
Applications - Cloud and OpenStack

OpenStack Alerts, Log and Metrics Management with InfluxDB
SCHEMA & DATA MODEL
Sample Data

<table>
<thead>
<tr>
<th>time</th>
<th>butterflies</th>
<th>honeybees</th>
<th>location</th>
<th>scientist</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-08-18T00:00:00Z</td>
<td>12</td>
<td>23</td>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T00:00:00Z</td>
<td>1</td>
<td>30</td>
<td>1</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
<td>11</td>
<td>28</td>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td><strong>2015-08-18T00:06:00Z</strong></td>
<td><strong>3</strong></td>
<td><strong>28</strong></td>
<td><strong>1</strong></td>
<td><strong>perpetua</strong></td>
</tr>
<tr>
<td>2015-08-18T05:54:00Z</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T06:00:00Z</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T06:06:00Z</td>
<td>8</td>
<td>23</td>
<td>2</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T06:12:00Z</td>
<td>7</td>
<td>22</td>
<td>2</td>
<td>perpetua</td>
</tr>
</tbody>
</table>
Fields

Field keys store meta data

Field values are your data

<table>
<thead>
<tr>
<th>name: census</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
</tr>
<tr>
<td>2015-08-18T00:00:00Z</td>
</tr>
<tr>
<td>2015-08-18T00:00:00Z</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
</tr>
<tr>
<td>2015-08-18T05:54:00Z</td>
</tr>
<tr>
<td>2015-08-18T06:00:00Z</td>
</tr>
<tr>
<td>2015-08-18T06:06:00Z</td>
</tr>
<tr>
<td>2015-08-18T06:12:00Z</td>
</tr>
</tbody>
</table>

Field Sets
Tags

Tag keys and values record metadata

Indexed

Tags are optional

<table>
<thead>
<tr>
<th>location</th>
<th>scientist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td>1</td>
<td>perpetua</td>
</tr>
<tr>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2</td>
<td>perpetua</td>
</tr>
<tr>
<td>2</td>
<td>perpetua</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>butterflies</th>
<th>honeybees</th>
<th>location</th>
<th>scientist</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-08-18T00:00:00Z</td>
<td>12</td>
<td>23</td>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T00:00:00Z</td>
<td>1</td>
<td>30</td>
<td>1</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
<td>11</td>
<td>28</td>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
<td>3</td>
<td>28</td>
<td>1</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T05:54:00Z</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T06:00:00Z</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T06:06:00Z</td>
<td>8</td>
<td>23</td>
<td>2</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T06:12:00Z</td>
<td>7</td>
<td>22</td>
<td>2</td>
<td>perpetua</td>
</tr>
</tbody>
</table>
Measurement

Container for tags, fields & time column

Conceptually similar to SQL table

A “measurement” can belong to multiple “retention policies”
Retention Policy (RP)

- Life of the data [DURATION]
- Copies stored in cluster [REPLICATION]
- “shard group” duration [SHARD DURATION]
- RP is unique per DB
- `autogen` is the default RP
  - DURATION = INF
  - REPLICATION = 1
  - SHARD DURATION = 7d
Retention Policy

InfluxQL command:

CREATE RETENTION POLICY <retention_policy_name> ON <database_name> DURATION <duration> REPLICATION <n> [SHARD DURATION <duration>] [DEFAULT]

Example:

CREATE RETENTION POLICY "one_day_only" ON "NOAA_water_database" DURATION 1d REPLICATION 1 SHARD DURATION 1h DEFAULT

# of data nodes that have the copy
**Measurement** *(Coming back!)*

Container for tags, fields & time column

Conceptually similar to SQL table

A “measurement” can belong to multiple “retention policies”

<table>
<thead>
<tr>
<th>name: census</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------------</td>
</tr>
<tr>
<td><strong>time</strong></td>
</tr>
<tr>
<td>2015-08-18T00:00:00Z</td>
</tr>
<tr>
<td>2015-08-18T00:00:00Z</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
</tr>
<tr>
<td><strong>2015-08-18T00:06:00Z</strong></td>
</tr>
<tr>
<td>2015-08-18T05:54:00Z</td>
</tr>
<tr>
<td>2015-08-18T06:00:00Z</td>
</tr>
<tr>
<td>2015-08-18T06:06:00Z</td>
</tr>
<tr>
<td>2015-08-18T06:12:00Z</td>
</tr>
</tbody>
</table>

Duration + Replication + Shard Duration
Series

- Collection of data that share retention policy, measurement & tag set

<table>
<thead>
<tr>
<th>Arbitrary series number</th>
<th>Retention policy</th>
<th>Measurement</th>
<th>Tag set</th>
</tr>
</thead>
<tbody>
<tr>
<td>series 1</td>
<td>autogen</td>
<td>census</td>
<td>location = 1, scientist = langstroth</td>
</tr>
<tr>
<td>series 2</td>
<td>autogen</td>
<td>census</td>
<td>location = 2, scientist = langstroth</td>
</tr>
<tr>
<td>series 3</td>
<td>autogen</td>
<td>census</td>
<td>location = 1, scientist = perpetua</td>
</tr>
<tr>
<td>series 4</td>
<td>autogen</td>
<td>census</td>
<td>location = 2, scientist = perpetua</td>
</tr>
</tbody>
</table>
Point

- Field set in the same series for a given timestamp
- Conceptually similar to an SQL record

<table>
<thead>
<tr>
<th>time</th>
<th>butterflies</th>
<th>honeybees</th>
<th>location</th>
<th>scientist</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-08-18T00:00:00Z</td>
<td>12</td>
<td>23</td>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T00:00:00Z</td>
<td>1</td>
<td>30</td>
<td>1</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
<td>11</td>
<td>28</td>
<td>1</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T00:06:00Z</td>
<td>3</td>
<td>28</td>
<td>1</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T05:54:00Z</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T06:00:00Z</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>langstroth</td>
</tr>
<tr>
<td>2015-08-18T06:06:00Z</td>
<td>8</td>
<td>23</td>
<td>2</td>
<td>perpetua</td>
</tr>
<tr>
<td>2015-08-18T06:12:00Z</td>
<td>7</td>
<td>22</td>
<td>2</td>
<td>perpetua</td>
</tr>
</tbody>
</table>
Tricky!

Retention Policy = A bucket of, \textbf{duration (life)} + [replication factor + shard duration]

Series = Retention Policy + Measurement + Tag Set

Point = Timestamp + Field Set + Series
DISTRIBUTION

“InfluxDB is distributed by design”
Why distributed database?

- Provides reliability
  - Data is located in multiple nodes in the cluster

- Offers scalability
  - For both write and query load
Sharding

- Lets us scale out
- Improves query and write performance
- Splits data on time field
Shard

- Contiguous block of time
- Represented by file on disk
- Contains a specific set of “series” for a given time duration
Shard Group

- Just logical containers → Duration + Replication Factor + Shard Duration
- Organized by “retention policy”
- Contains 1 or more shards
Each shard stores a specific set of series.

All points in *same series* are stored in *same shard*.
Replication

- Redundancy to prevent data loss
- Retention Policy determines the replication factor

Duration + Replication Factor + Shard Duration
Pros

● InfluxDB is a schema-less DB. Tags and Fields can be added on the fly!
● Optimized for high volume of reads and writes
● Writing of data in time ascending order is super fast

Cons

● No table joins due to schema-less design
● Updates and deletes are significantly restricted
● Writing of data with random times is slow
Influx QL
Comparing to SQL

Timing is everything!

Dynamic schema

Not CRUD

<table>
<thead>
<tr>
<th>measurement</th>
<th>SQL Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>point</td>
<td>Rows</td>
</tr>
<tr>
<td>fields</td>
<td>Unindexed columns</td>
</tr>
<tr>
<td>tags</td>
<td>Indexed columns</td>
</tr>
<tr>
<td>Retention policies / Continuous Queries</td>
<td>Stored Procedures / Materialized views</td>
</tr>
</tbody>
</table>

### Example Data

<table>
<thead>
<tr>
<th>park_id</th>
<th>planet</th>
<th>time</th>
<th>#_foodships</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Earth</td>
<td>142918560000000000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Earth</td>
<td>142918560100000000</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Saturn</td>
<td>142918560200000000</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Saturn</td>
<td>142918560300000000</td>
<td>14</td>
</tr>
</tbody>
</table>

```plaintext
name: foodships
tags: park_id=1, planet=Earth
time #_foodships
----- ------------------
2015-04-16T12:00:00Z 0
2015-04-16T12:00:01Z 3

name: foodships
tags: park_id=2, planet=Saturn
time #_foodships
----- ------------------
2015-04-16T12:00:02Z 10
2015-04-16T12:00:03Z 14
```
InfluxQL

```
SELECT * FROM "foodships" WHERE "planet" = 'Saturn' AND time > '2015-04-16 12:00:01'
```

Tags are strings

Relative or Absolute
SELECT

Regex - For tags.

```
SELECT "level" + 2 FROM "h2o_feet" WHERE location !~ ./.
```

Tag and Field Value

```
SELECT * FROM "h2o_feet" WHERE "location" = 'gainesville'
AND "level" + 2 > 10

SELECT "level"::field, "location"::tag FROM "h2o_feet"
```

Comparators

- = equal to
- != not equal to
- > greater than
- < less than
- =~ matches against
- !~ doesn’t match against
GROUP BY

Tags

```
SELECT MEAN("level") FROM "h2o_feet" GROUP BY "location"
```

Time Interval

```
SELECT MEAN("level") FROM "h2o_feet" WHERE time > now() - 2w GROUP BY "location",time(3d, -1d)
```

- u Microseconds
- ms Milliseconds
- s Seconds
- m Minutes
- h Hours
- d Days
- w Weeks

`now() → current time`
SELECT MEAN("level") FROM h2o_feet WHERE time >= '2015-08-18' AND time < '2015-09-24' GROUP BY time(10d) fill(-100)

Options in fill clause:

None

Previous

Null

Any numerical value!
Regex

SELECT * FROM /.*/ LIMIT 1

SELECT * FROM /.*temperature.*/ LIMIT 5

SELECT * FROM "h2o_feet" WHERE "location" !~ /.*a.*/ LIMIT 4

Tags are Strings!

Cast Fields

SELECT "level"::float FROM "h2o_feet" LIMIT 4
DB - RP - Series

- Database
- Retention Policy
  - Series
  - Series
  - Series
Downsampling

INTO

Relocate Data to another database, retention policy, measurement.

SELECT "level" INTO "h2o_feet_copy" FROM "h2o_feet" WHERE "location" = 'coyote_creek'

Downsample

SELECT MEAN("level") INTO "average" FROM "h2o_feet" WHERE "location" = 'santa_monica' AND time >= now() - 2w GROUP BY time(12m)
LIMIT and SLIMIT

Return 3 from each series.

```
SELECT "level" FROM "h2o_feet" GROUP BY * LIMIT 3
```

Return 3 oldest from one of the series

```
SELECT "level" FROM "h2o_feet" GROUP BY * LIMIT 3 SLIMIT 1
```
Continuous Queries

Humongous Data!

```
CREATE CONTINUOUS QUERY "mean" on "h2o_db" BEGIN

SELECT min("level") INTO "min_level" FROM "h2o_feet" GROUP BY time(1d)
END

CREATE CONTINUOUS QUERY "mean_sample" ON "h2o_db"

RESAMPLE EVERY 15m FOR 60m BEGIN

SELECT min("level") AS "miniscule", max("h2o_feet") AS "monstrous" INTO "h2o_feet_min" FROM "h2o_feet" GROUP BY time(30m)
END
```
Continuous Query

CREATE CONTINUOUS QUERY [name_of_continuous_query] ON [name_of_db] [RESAMPLE [EVERY interval] [FOR interval]] BEGIN

    SELECT [inner_part_of_select] INTO [new_measurement] FROM [measurement]

    GROUP BY time([frequency]), [tags]

END

EVERY Clause specified how frequently the CQ will run. FOR Clause specifies how far back the CQ resample.
Design Policies

Do’s

Commonly queried → Tags.
Group By → Tags
InfluxQL function → Fields
Non string data → Field

Don’ts

Series cardinality
Kinds of info in a single tag
Database Management

Create db

```
curl -i -XPOST http://localhost:8086/query --data-urlencode "q=CREATE DATABASE pirates"
```

CREATE DATABASE pirates

Insert Data

```
INSERT INTO treasures,captain_id=pirate_king value=2

curl -i -XPOST 'http://localhost:8086/write?db=pirates' --data-binary 'treasures,captain_id=pirate_king value=2'
```
Database Management

Writing from a file

curl -i -XPOST 'http://localhost:8086/write?db=pirates' --data-binary @treasures.txt

Querying Data using http

curl -GET 'http://localhost:8086/query?pretty=true' --data-urlencode "db=pirates" --data-urlencode "q=SELECT "value" FROM "treasures" WHERE "captain_id"='pirate_king'"

epoch=[h,m,s,ms,u,ns]

Chunk size

Max row limit
TSM - InfluxDB Storage Engine

- TSM - Time Structured Merge
- Very Similar to LSM - Log Structured Merge
  - Cassandra, LevelDB
Requirements

High Write throughput

Data Compression

Simultaneous reads and writes without blocking

Columnar format

No limit on number of fields.
TSM Components & Data Flow

Time Series Data

- WAL
- In memory cache
- Index files
WAL - Write ahead log

Append only file

Success message sent to the user

First entry point from which data corruption handled

Fsync data with In memory cache
In memory cache

2 Internal Cache components

- Regular write cache
- Flush Cache

Memory Threshold

Data is split and stored.

Map[String] Values
In memory cache

temperature,device=dev1,building=b1 internal=80,external=18 1443782126

Measurement → Tags → Fields → Timestamp

temperature,device=dev1,building=b1#internal → 1

1 → (1443782126,80)

temperature,device=dev1,building=b1#external → 2

2 → (1443782126,18)
Index/Data files - on Disk

Contiguous Data blocks.

Can overlap on time - but a series within cannot.

<table>
<thead>
<tr>
<th>DF - 1</th>
<th>Min: 10000</th>
<th>Max: 25000</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF - 2</td>
<td>Min: 15000</td>
<td>Max: 30000</td>
</tr>
<tr>
<td>DF - 3</td>
<td>Min: 35000</td>
<td>Max: 60000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Series A</th>
<th>Min: 10000</th>
<th>Max: 14000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Series A</td>
<td>Min: 15000</td>
<td>Max: 30000</td>
</tr>
<tr>
<td>Series A</td>
<td>Min: 35000</td>
<td>Max: 60000</td>
</tr>
</tbody>
</table>
Properties:

Read only

Periodic Compaction

Compressed data

<table>
<thead>
<tr>
<th>DF - 1</th>
<th>DF - 2</th>
<th>DF - 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min: 10000</td>
<td>Min: 15000</td>
<td>Min: 35000</td>
</tr>
<tr>
<td>Max: 25000</td>
<td>Max: 30000</td>
<td>Max: 60000</td>
</tr>
</tbody>
</table>
The Index

Data File
Min Time: 10000
Max Time: 29999

Data File
Min Time: 30000
Max Time: 39999

Data File
Min Time: 70000
Max Time: 99999

they periodically get compacted (like LSM)
Data File Layout

- 4 byte magic number
- Data block 1
- Data block n...
- Index Block
- 8 byte min time
- 8 byte max time
- 4 byte series count
- 8 byte uint64 ID
- 4 byte uint32 length
- 8 byte min timestamp
- Compressed block
TICK Stack

**Telegraf**
Collects time-series data from a variety of sources

**InfluxDB**
Delivers high performance writes and efficiently stores time-series data.

**Chronograf**
Visualizes and graphs the time-series data stored in InfluxDB.

**Kapacitor**
Provides alerting, ETL and detects anomalies in time-series data.
References

- https://www.influxdata.com/
Thank You!