

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?

33.15 +0.08 (0.24%)

Oct 25 - Close
NYSE real-time data - Disclaimer
Currency in USD

Range	33.01 - 33.49	Div/yield	0.12/1.45
52 week	29.52 - 36.43	EPS	2.32
Open	33.21	Shares	4.65B
Vol / Avg	19.82M/22.63M	Beta	1.08
Mkt cap	154.01B	Inst. own	61%
P/E	14.29		

199

Compare: Dow Jones S&P 500 HPQ IBM MSFT CRM INTC CA SI

Zoom: [1d](#) [5d](#) [1m](#) [3m](#) [6m](#) [YTD](#) [1y](#) [5y](#) [10y](#) [All](#)

Oct 21, 2013 - Oct 25, 2013 +0.25 (0.76%)



[Settings](#) | [Technicals](#) | [Link to this view](#)

Volume delayed by 15 mins.

Audience Overview

Nov 4, 2013 - Nov 11, 2013

Email Export Add to Dashboard Shortcut



All Visits
100.00%

Overview

Visits vs. Select a metric

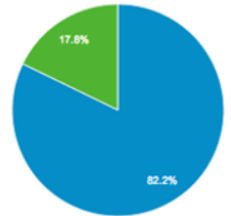
Hourly Day Week Month



15,886 people visited this site



■ New Visitor ■ Returning Visitor



64.242.88.10 - - [07/Mar/2004:16:05:49 -0800] "GET /twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVariables HTTP/1.1" 401 12846
64.242.88.10 - - [07/Mar/2004:16:06:51 -0800] "GET /twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1" 200 4523
64.242.88.10 - - [07/Mar/2004:16:10:02 -0800] "GET /mailman/listinfo/hadivision HTTP/1.1" 200 6291
64.242.88.10 - - [07/Mar/2004:16:11:58 -0800] "GET /twiki/bin/view/TWiki/WikiSyntax HTTP/1.1" 200 7352
64.242.88.10 - - [07/Mar/2004:16:20:55 -0800] "GET /twiki/bin/view/Main/DCCAndPostFix HTTP/1.1" 200 5253
64.242.88.10 - - [07/Mar/2004:16:23:12 -0800] "GET /twiki/bin/coops/TWiki/AppendixFileSystem?template=oopsmore%ml=1.12%fm2=1.12 HTTP/1.1" 200 11382
64.242.88.10 - - [07/Mar/2004:16:24:16 -0800] "GET /twiki/bin/view/Main/PeterThoeny HTTP/1.1" 200 4924
64.242.88.10 - - [07/Mar/2004:16:29:16 -0800] "GET /twiki/bin/edit/Main/Header_checks?topicparent=Main.ConfigurationVariables HTTP/1.1" 401 12851
64.242.88.10 - - [07/Mar/2004:16:30:29 -0800] "GET /twiki/bin/attach/Main/OfficeLocations HTTP/1.1" 401 12851
64.242.88.10 - - [07/Mar/2004:16:31:48 -0800] "GET /twiki/bin/view/TWiki/WebTopicEditTemplate HTTP/1.1" 200 3732
64.242.88.10 - - [07/Mar/2004:16:32:50 -0800] "GET /twiki/bin/view/Main/WebChanges HTTP/1.1" 200 40520
64.242.88.10 - - [07/Mar/2004:16:33:53 -0800] "GET /twiki/bin/edit/Main/Smtppd_etrn_restrictions?topicparent=Main.ConfigurationVariables HTTP/1.1" 401 12851
64.242.88.10 - - [07/Mar/2004:16:35:19 -0800] "GET /mailman/listinfo/business HTTP/1.1" 200 6379
64.242.88.10 - - [07/Mar/2004:16:36:22 -0800] "GET /twiki/bin/rdiff/Main/WebIndex?rev1=1.2&rev2=1.1 HTTP/1.1" 200 46373
64.242.88.10 - - [07/Mar/2004:16:37:27 -0800] "GET /twiki/bin/view/TWiki/DontNotify HTTP/1.1" 200 4140
64.242.88.10 - - [07/Mar/2004:16:39:24 -0800] "GET /twiki/bin/view/Main/TokyoOffice HTTP/1.1" 200 3853
64.242.88.10 - - [07/Mar/2004:16:43:54 -0800] "GET /twiki/bin/view/Main/MikeMannix HTTP/1.1" 200 3686
64.242.88.10 - - [07/Mar/2004:16:45:56 -0800] "GET /twiki/bin/attach/Main/PostfixCommands HTTP/1.1" 401 12846
64.242.88.10 - - [07/Mar/2004:16:47:12 -0800] "GET /robots.txt HTTP/1.1" 200 68
64.242.88.10 - - [07/Mar/2004:16:47:46 -0800] "GET /twiki/bin/rdiff/Know/ReadmeFirst?rev1=1.5&rev2=1.4 HTTP/1.1" 200 5724
64.242.88.10 - - [07/Mar/2004:16:49:04 -0800] "GET /twiki/bin/view/Main/TWikiGroups?rev=1.2 HTTP/1.1" 200 5162
64.242.88.10 - - [07/Mar/2004:16:50:54 -0800] "GET /twiki/bin/rdiff/Main/ConfigurationVariables HTTP/1.1" 200 59679
64.242.88.10 - - [07/Mar/2004:16:52:35 -0800] "GET /twiki/bin/edit/Main/Flush_service_name?topicparent=Main.ConfigurationVariables HTTP/1.1" 401 12851
64.242.88.10 - - [07/Mar/2004:16:53:46 -0800] "GET /twiki/bin/rdiff/TWiki/Registration HTTP/1.1" 200 34395
64.242.88.10 - - [07/Mar/2004:16:54:55 -0800] "GET /twiki/bin/rdiff/Main/NicholasLee HTTP/1.1" 200 7235
64.242.88.10 - - [07/Mar/2004:16:56:39 -0800] "GET /twiki/bin/view/Sandbox/WebHome?rev=1.6 HTTP/1.1" 200 8545
64.242.88.10 - - [07/Mar/2004:16:58:54 -0800] "GET /mailman/listinfo/administration HTTP/1.1" 200 6459
lordgun.org - - [07/Mar/2004:17:01:53 -0800] "GET /razor.html HTTP/1.1" 200 2869
64.242.88.10 - - [07/Mar/2004:17:09:01 -0800] "GET /twiki/bin/search/Main/SearchResult?scope=text&ex=on&search=Joris%20*Benschop[^A-Za-z] HTTP/1.1" 200 4284
64.242.88.10 - - [07/Mar/2004:17:10:20 -0800] "GET /twiki/bin/coops/TWiki/textFormattingRules?template=oopsmore%ml=1.37%fm2=1.37 HTTP/1.1" 200 11400
64.242.88.10 - - [07/Mar/2004:17:13:50 -0800] "GET /twiki/bin/edit/TWiki/DefaultPlugin?t=1078688936 HTTP/1.1" 401 12846
64.242.88.10 - - [07/Mar/2004:17:16:00 -0800] "GET /twiki/bin/search/Main/?scope=topic&ex=on&search=^g HTTP/1.1" 200 3675
64.242.88.10 - - [07/Mar/2004:17:17:27 -0800] "GET /twiki/bin/search/TWiki/?scope=topic&ex=on&search=^d HTTP/1.1" 200 5773
l1j036.inktomisearch.com - - [07/Mar/2004:17:18:36 -0800] "GET /robots.txt HTTP/1.0" 200 68
l1j090.inktomisearch.com - - [07/Mar/2004:17:18:41 -0800] "GET /twiki/bin/view/Main/LondonOffice HTTP/1.0" 200 3860
64.242.88.10 - - [07/Mar/2004:17:21:44 -0800] "GET /twiki/bin/attach/TWiki/TablePlugin HTTP/1.1" 401 12846
64.242.88.10 - - [07/Mar/2004:17:22:49 -0800] "GET /twiki/bin/view/TWiki/ManagingWebs?rev=1.22 HTTP/1.1" 200 9310
64.242.88.10 - - [07/Mar/2004:17:23:54 -0800] "GET /twiki/bin/statistics/Main HTTP/1.1" 200 808
64.242.88.10 - - [07/Mar/2004:17:26:30 -0800] "GET /twiki/bin/view/TWiki/WikiCulture HTTP/1.1" 200 5935
64.242.88.10 - - [07/Mar/2004:17:27:37 -0800] "GET /twiki/bin/edit/Main/WebSearch?t=1078669682 HTTP/1.1" 401 12846
64.242.88.10 - - [07/Mar/2004:17:28:45 -0800] "GET /twiki/bin/coops/TWiki/ResetPassword?template=oopsmore%ml=1.4%fm2=1.4 HTTP/1.1" 200 11281
64.242.88.10 - - [07/Mar/2004:17:29:59 -0800] "GET /twiki/bin/view/TWiki/ManagingWebs?skin=print HTTP/1.1" 200 8806
64.242.88.10 - - [07/Mar/2004:17:31:39 -0800] "GET /twiki/bin/edit/Main/UvscanAndPostFix?topicparent=Main.WebHome HTTP/1.1" 401 12846
64.242.88.10 - - [07/Mar/2004:17:35:35 -0800] "GET /twiki/bin/view/TWiki/KlausWriessnegger HTTP/1.1" 200 3848
64.242.88.10 - - [07/Mar/2004:17:39:39 -0800] "GET /twiki/bin/view/Main/SpamAssassin HTTP/1.1" 200 4081
64.242.88.10 - - [07/Mar/2004:17:42:15 -0800] "GET /twiki/bin/coops/TWiki/RichardDonkin?template=oopsmore%ml=1.2%fm2=1.2 HTTP/1.1" 200 11281
64.242.88.10 - - [07/Mar/2004:17:46:17 -0800] "GET /twiki/bin/rdiff/TWiki/AlWilliams?rev1=1.3&rev2=1.2 HTTP/1.1" 200 4485
64.242.88.10 - - [07/Mar/2004:17:47:43 -0800] "GET /twiki/bin/rdiff/TWiki/AlWilliams?rev1=1.2&rev2=1.1 HTTP/1.1" 200 5234
64.242.88.10 - - [07/Mar/2004:17:50:44 -0800] "GET /twiki/bin/view/TWiki/SvenDowideit HTTP/1.1" 200 3616
64.242.88.10 - - [07/Mar/2004:17:53:45 -0800] "GET /twiki/bin/search/Main/SearchResult?scope=text&ex=on&search=Office%20*Locations[^A-Za-z] HTTP/1.1" 200 7771

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

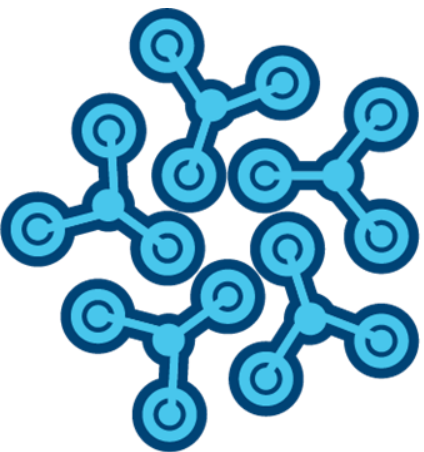
RRDTool (metrics)

Graphite (metrics)

OpenTSDB (metrics + events)

Kairos (metrics + events)

Something missing...



InfluxDB

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

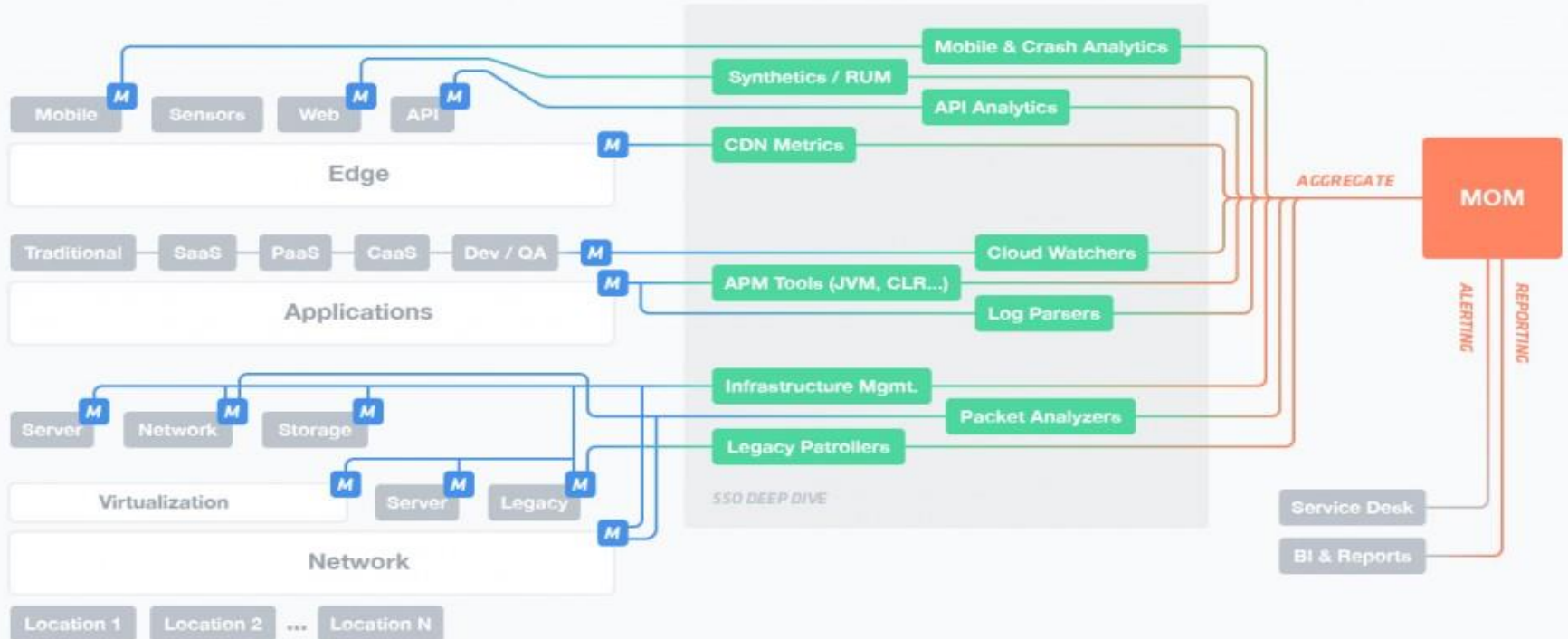
Applications - IoT

Example IoT Smart Utilities Application with InfluxDB



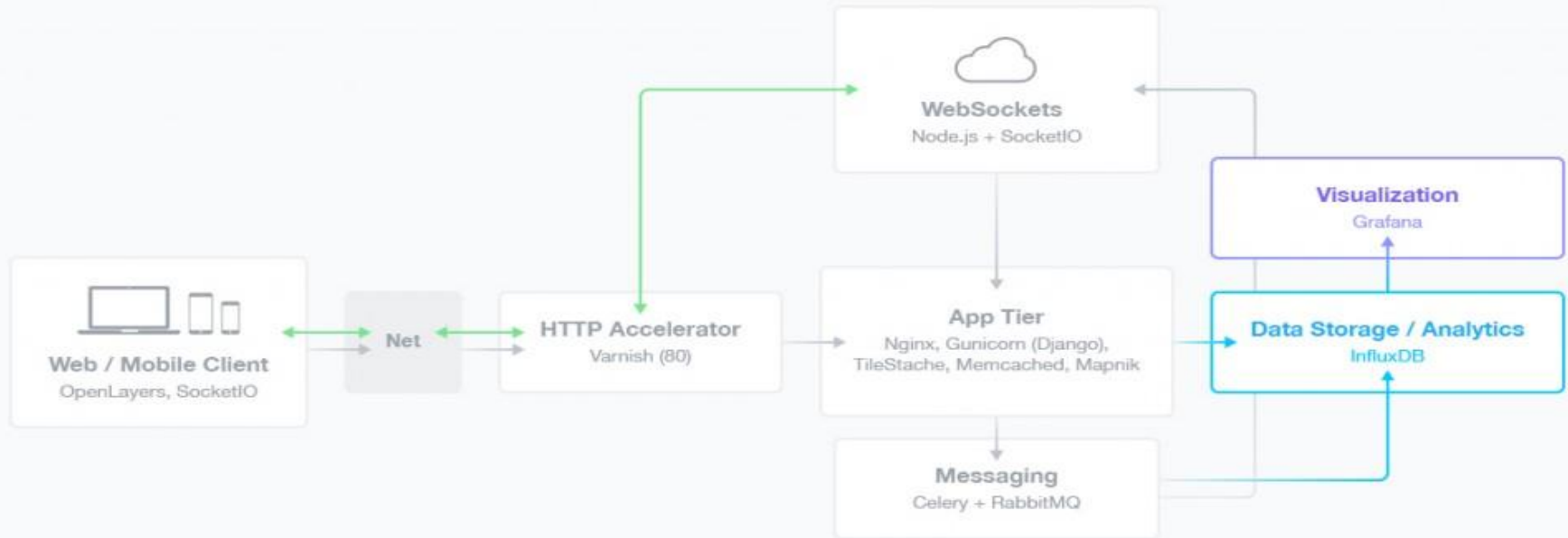
Applications - Custom DevOps Monitoring

More Metrics and Monitoring Mean More Problems



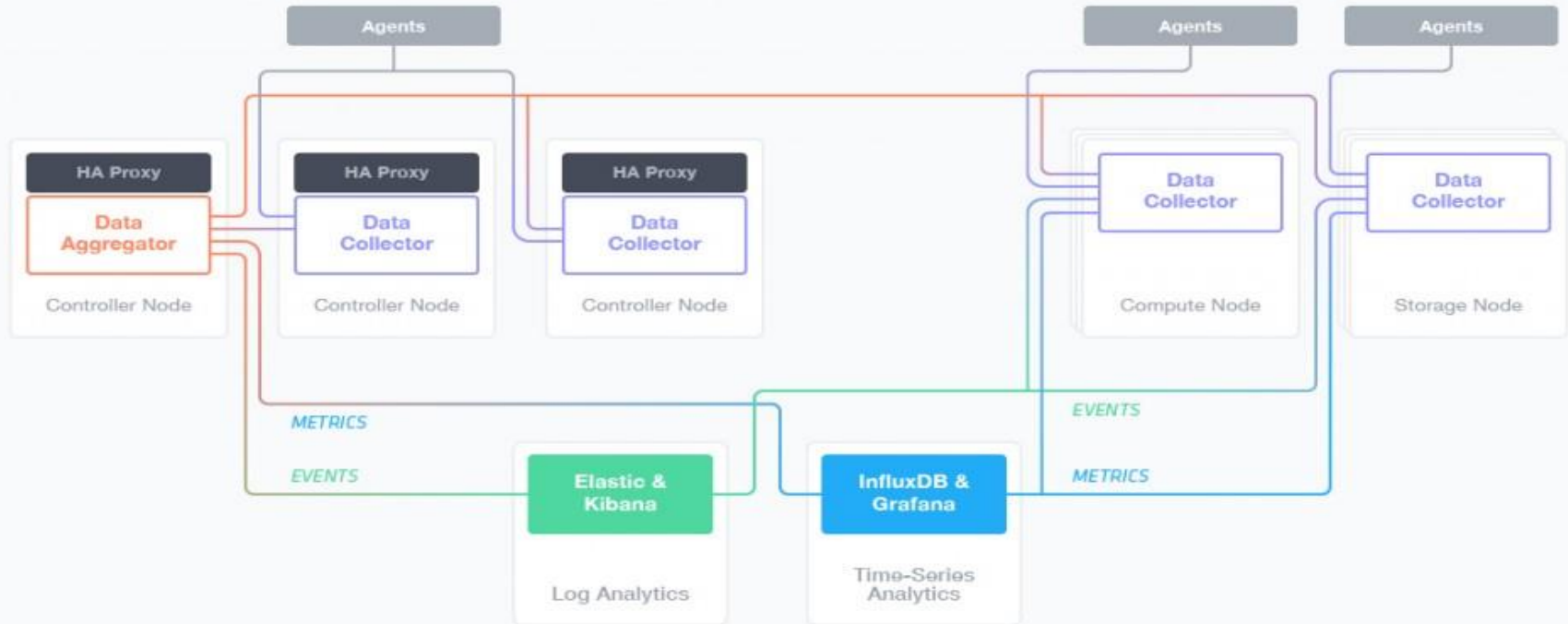
Applications - Real time Analytics

Real-Time Analytics Geolocation Application with InfluxDB



Applications - Cloud and OpenStack

OpenStack Alerts, Log and Metrics Management with InfluxDB



SCHEMA & DATA MODEL

Sample Data

Measurement (SQL table)

Fields (not indexed)

Tags (indexed)

name: **census**

time	butterflies	honeybees	location	scientist
2015-08-18T00:00:00Z	12	23	1	langstroth
2015-08-18T00:00:00Z	1	30	1	perpetua
2015-08-18T00:06:00Z	11	28	1	langstroth
2015-08-18T00:06:00Z	3	28	1	perpetua
2015-08-18T05:54:00Z	2	11	2	langstroth
2015-08-18T06:00:00Z	1	10	2	langstroth
2015-08-18T06:06:00Z	8	23	2	perpetua
2015-08-18T06:12:00Z	7	22	2	perpetua

Point (SQL record)



Fields

Field keys store meta data

Field values are your data

```
butterflies = 12 honeybees = 23
butterflies = 1 honeybees = 30
butterflies = 11 honeybees = 28
butterflies = 3 honeybees = 28
butterflies = 2 honeybees = 11
butterflies = 1 honeybees = 10
butterflies = 8 honeybees = 23
butterflies = 7 honeybees = 22
```

Field Sets

name: **census**

time	butterflies	honeybees	location	scientist
2015-08-18T00:00:00Z	12	23	1	langstroth
2015-08-18T00:00:00Z	1	30	1	perpetua
2015-08-18T00:06:00Z	11	28	1	langstroth
2015-08-18T00:06:00Z	3	28	1	perpetua
2015-08-18T05:54:00Z	2	11	2	langstroth
2015-08-18T06:00:00Z	1	10	2	langstroth
2015-08-18T06:06:00Z	8	23	2	perpetua
2015-08-18T06:12:00Z	7	22	2	perpetua

Fields

Tags

Tag keys and values record metadata

Indexed

Tags are optional

name: **census**

time	butterflies	honeybees	location	scientist
2015-08-18T00:00:00Z	12	23	1	langstroth
2015-08-18T00:00:00Z	1	30	1	perpetua
2015-08-18T00:06:00Z	11	28	1	langstroth
2015-08-18T00:06:00Z	3	28	1	perpetua
2015-08-18T05:54:00Z	2	11	2	langstroth
2015-08-18T06:00:00Z	1	10	2	langstroth
2015-08-18T06:06:00Z	8	23	2	perpetua
2015-08-18T06:12:00Z	7	22	2	perpetua

Tags

location = 1, scientist = langstroth
location = 2, scientist = langstroth
location = 1, scientist = perpetua
location = 2, scientist = perpetua

Tag Set

Measurement

Container for tags, fields & time column

Conceptually similar to SQL table

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

name: **census**

```
-----
```

time	butterflies	honeybees	location	scientist
2015-08-18T00:00:00Z	12	23	1	langstroth
2015-08-18T00:00:00Z	1	30	1	perpetua
2015-08-18T00:06:00Z	11	28	1	langstroth
2015-08-18T00:06:00Z	3	28	1	perpetua
2015-08-18T05:54:00Z	2	11	2	langstroth
2015-08-18T06:00:00Z	1	10	2	langstroth
2015-08-18T06:06:00Z	8	23	2	perpetua
2015-08-18T06:12:00Z	7	22	2	perpetua


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]
```



of data nodes that
have the copy

Example:

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

Measurement (Coming back!)

Container for tags, fields & time column

Conceptually similar to SQL table

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



Duration + Replication + Shard Duration

```
name: census
```

```
-----
```

time	butterflies	honeybees	location	scientist
2015-08-18T00:00:00Z	12	23	1	langstroth
2015-08-18T00:00:00Z	1	30	1	perpetua
2015-08-18T00:06:00Z	11	28	1	langstroth
2015-08-18T00:06:00Z	3	28	1	perpetua
2015-08-18T05:54:00Z	2	11	2	langstroth
2015-08-18T06:00:00Z	1	10	2	langstroth
2015-08-18T06:06:00Z	8	23	2	perpetua
2015-08-18T06:12:00Z	7	22	2	perpetua

Series

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

Arbitrary series number	Retention policy	Measurement	Tag set
series 1	autogen	census	location = 1, scientist = langstroth
series 2	autogen	census	location = 2, scientist = langstroth
series 3	autogen	census	location = 1, scientist = perpetua
series 4	autogen	census	location = 2, scientist = perpetua

Point

(Measurement + Tag Set) remember?

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

name: **census**

```
-----  
time          butterflies  honeybees  location  scientist  
2015-08-18T00:00:00Z  12          23         1         langstroth  
2015-08-18T00:00:00Z  1           30         1         perpetua  
2015-08-18T00:06:00Z  11          28         1         langstroth  
2015-08-18T00:06:00Z  3           28         1         perpetua  
2015-08-18T05:54:00Z  2           11         2         langstroth  
2015-08-18T06:00:00Z  1           10         2         langstroth  
2015-08-18T06:06:00Z  8           23         2         perpetua  
2015-08-18T06:12:00Z  7           22         2         perpetua
```

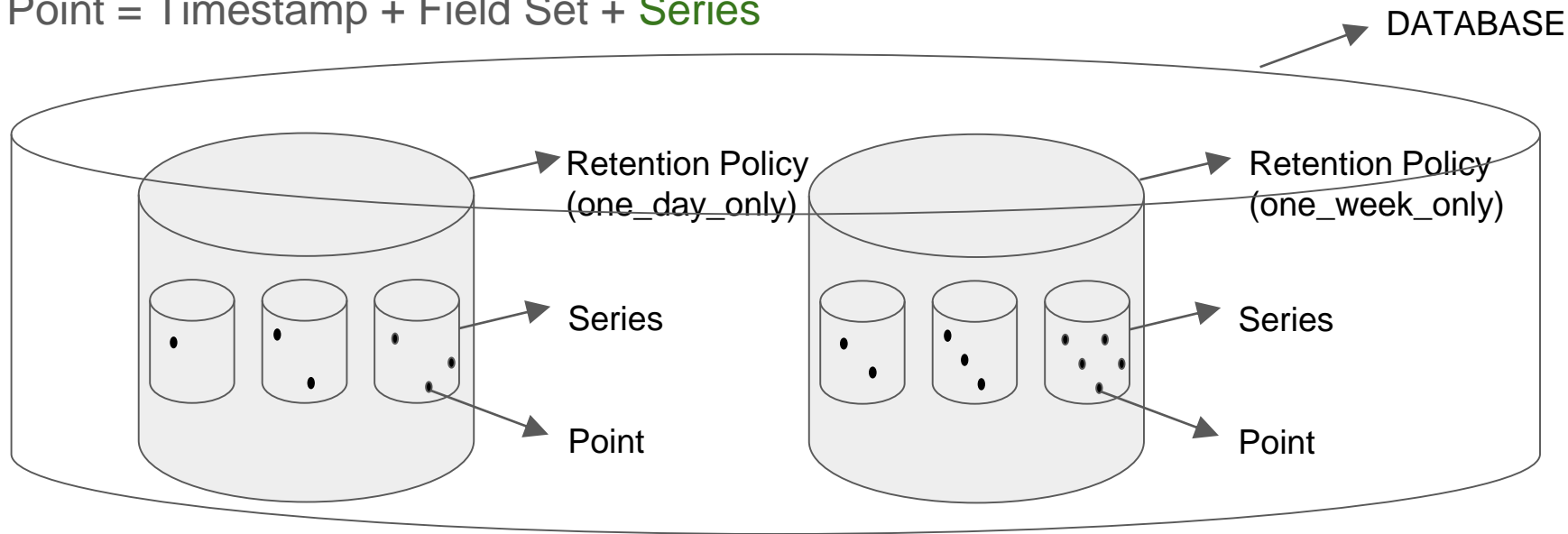
Point

Tricky!

Retention Policy = A bucket of, 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

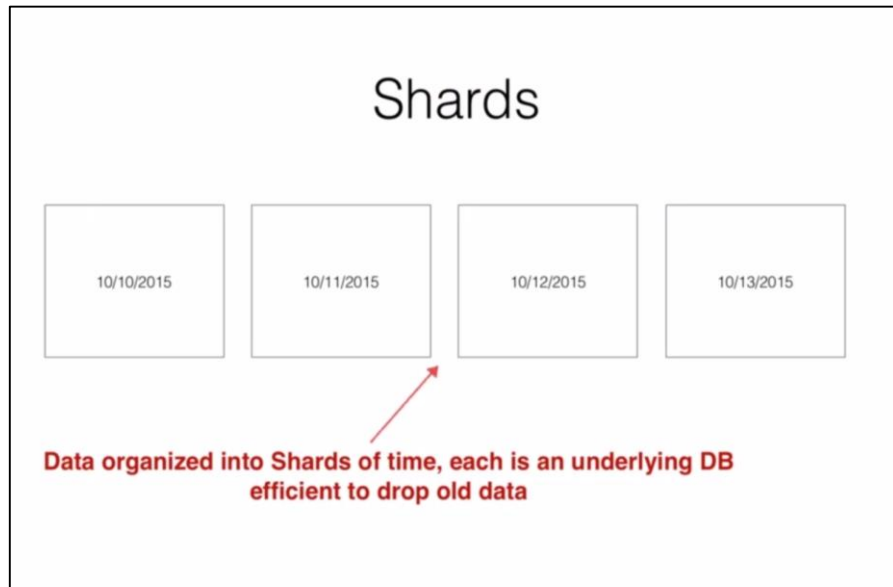
- Offers scalability
 - For both write and query load

Sharding


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

Shard

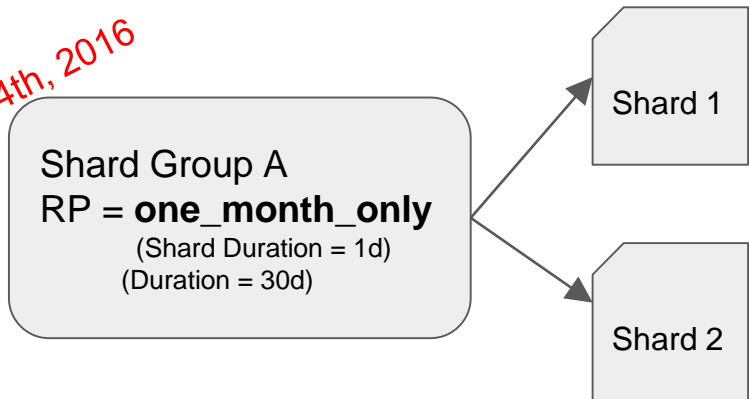
- Contiguous block of time
- Represented by file on disk
- Contains a specific set of “series” for a given time duration



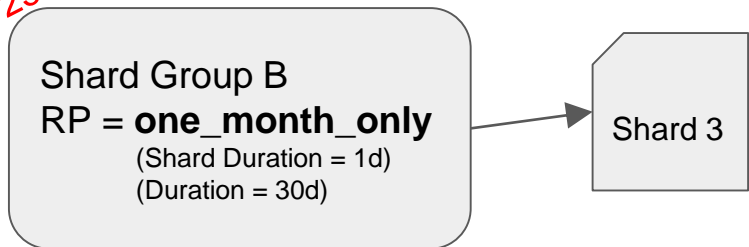
Shard Group

- Just logical containers  Duration + Replication Factor + **Shard Duration**
- Organized by “retention policy”
- Contains 1 or more shards

Sept 24th, 2016



Sept 25th, 2016

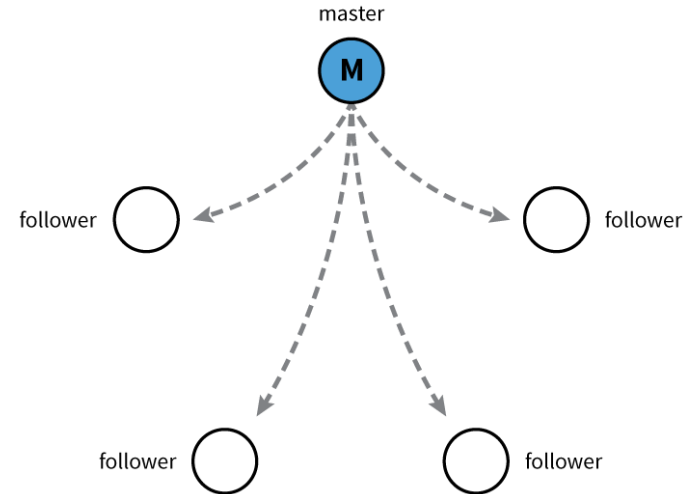
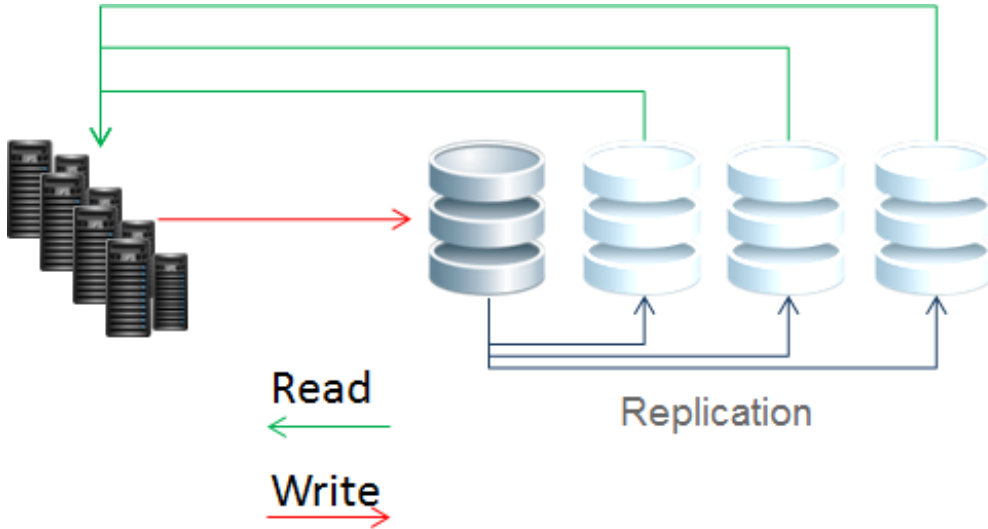


- Each shard stores a specific set of series
- All points in **same series** are stored in **same shard**.

Replication

Duration + **Replication Factor** + Shard Duration

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



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

measurement	SQL Table
point	Rows
fields	Unindexed columns
tags	Indexed columns
Retention policies / Continuous Queries	Stored Procedures / Materialized views

```
+-----+-----+-----+-----+
| park_id | planet | time | #_foodships |
+-----+-----+-----+-----+
| 1 | Earth | 142918560000000000 | 0 |
| 1 | Earth | 142918560100000000 | 3 |
| 2 | Saturn | 142918560200000000 | 10 |
| 2 | Saturn | 142918560300000000 | 14 |
+-----+-----+-----+-----+
```

```
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

GROUP BY

Fill

```
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

Regex

```
SELECT * FROM ./ */ LIMIT 1
```

```
SELECT * FROM /*temperature */ LIMIT 5
```

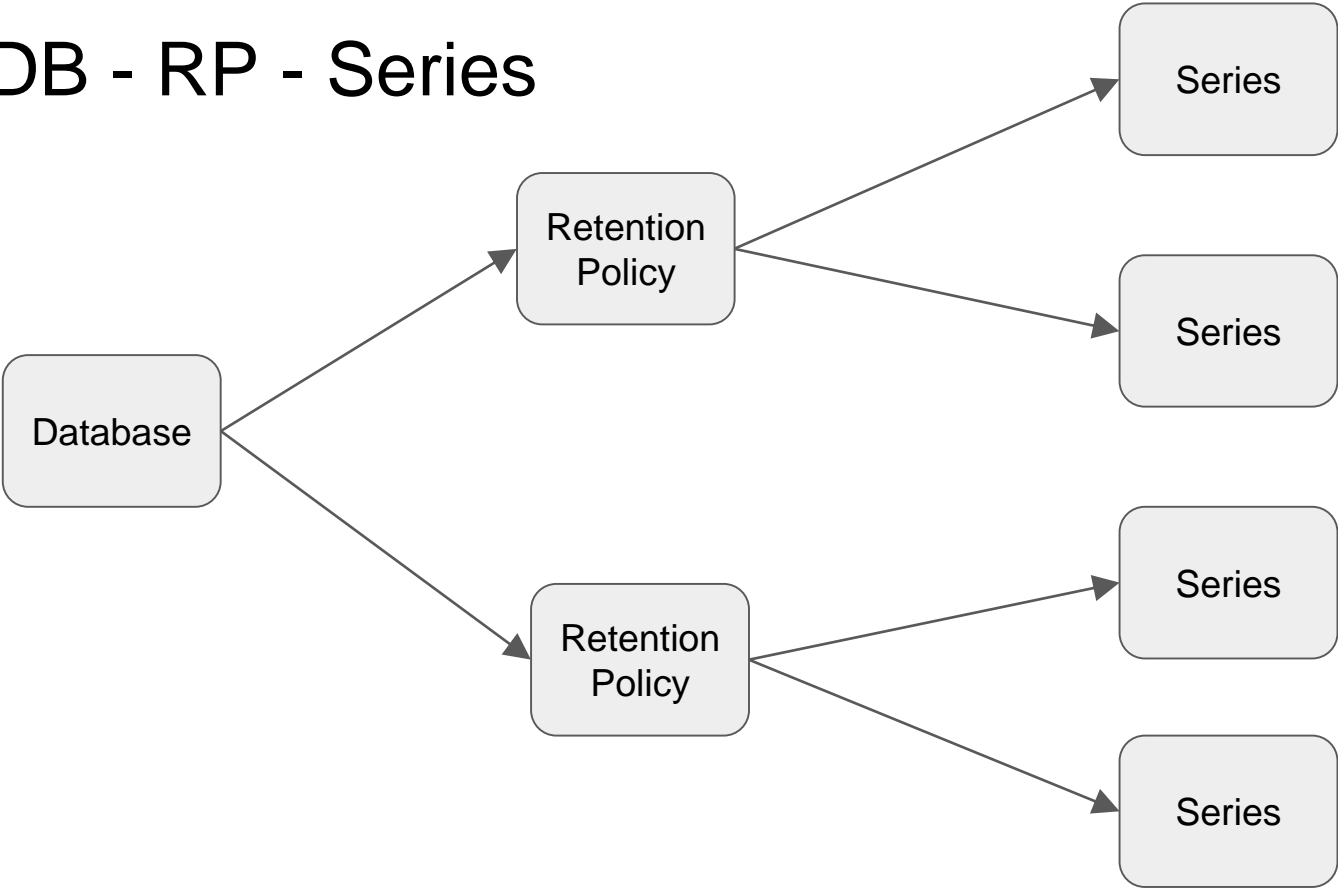
```
SELECT * FROM "h2o_feet" WHERE "location" !~ /*a */ LIMIT 4
```

Tags are Strings !

Cast Fields

```
SELECT "level"::float FROM "h2o_feet" LIMIT 4
```

DB - RP - Series



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

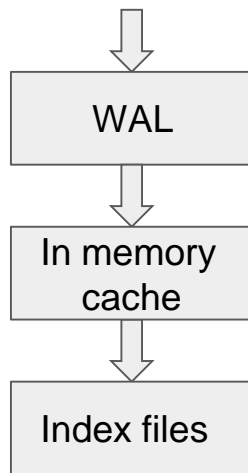
Simultaneous reads and writes without blocking

Columnar format

No limit on number of fields.

TSM Components & Data Flow

Time Series Data



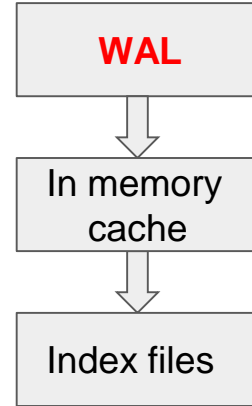
WAL - Write ahead log

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

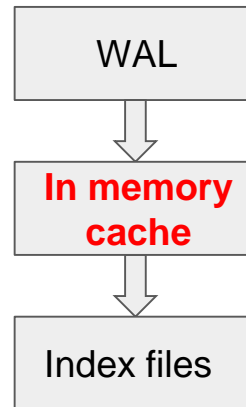
Regular write cache

Flush Cache

Memory Threshold

Data is split and stored.

Map[String] Values



In memory cache



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

DF - 1
Min: 10000
Max: 25000

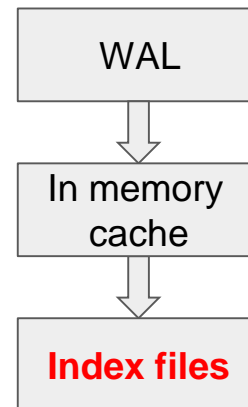
DF - 2
Min: 15000
Max: 30000

DF - 3
Min: 35000
Max: 60000

Series A
Min: 10000
Max: 14000

Series A
Min: 15000
Max: 30000

Series A
Min: 35000
Max: 60000



Properties:

Read only

Periodic Compaction

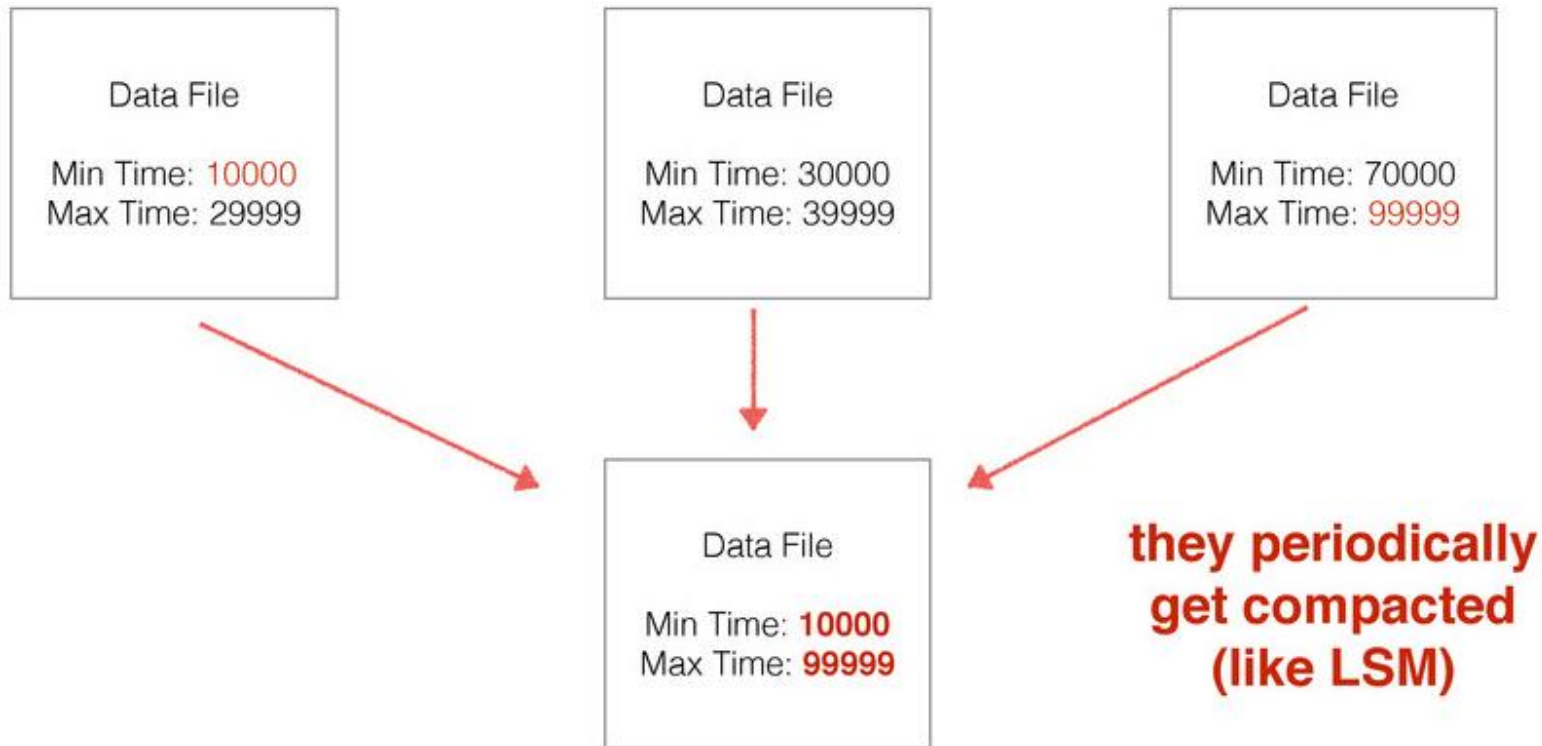
Compressed data

DF - 1
Min: 10000
Max: 25000

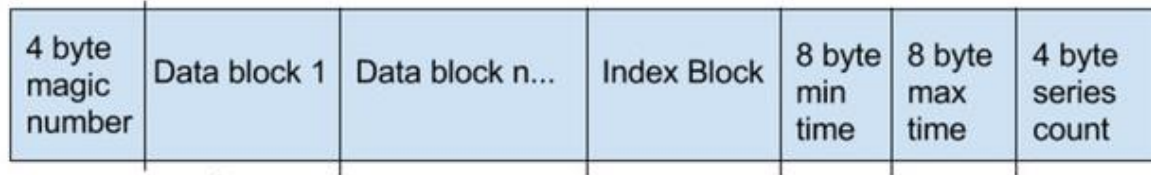
DF - 2
Min: 15000
Max: 30000

DF - 3
Min: 35000
Max: 60000

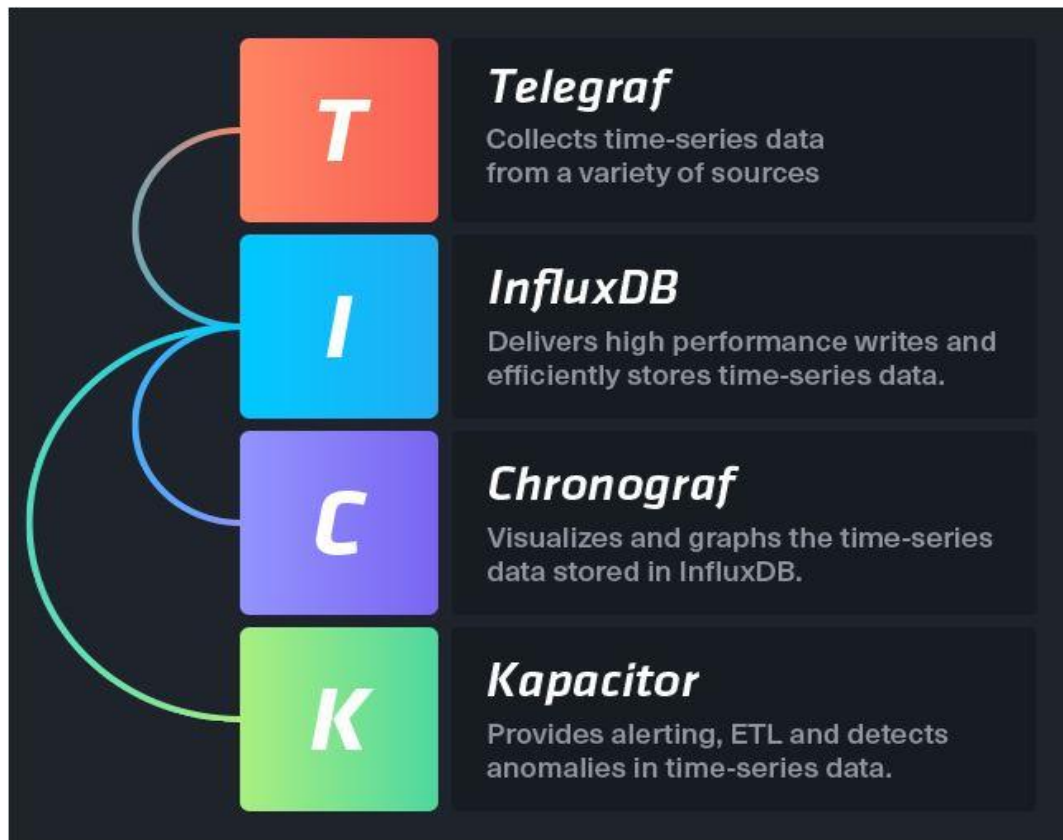
The Index

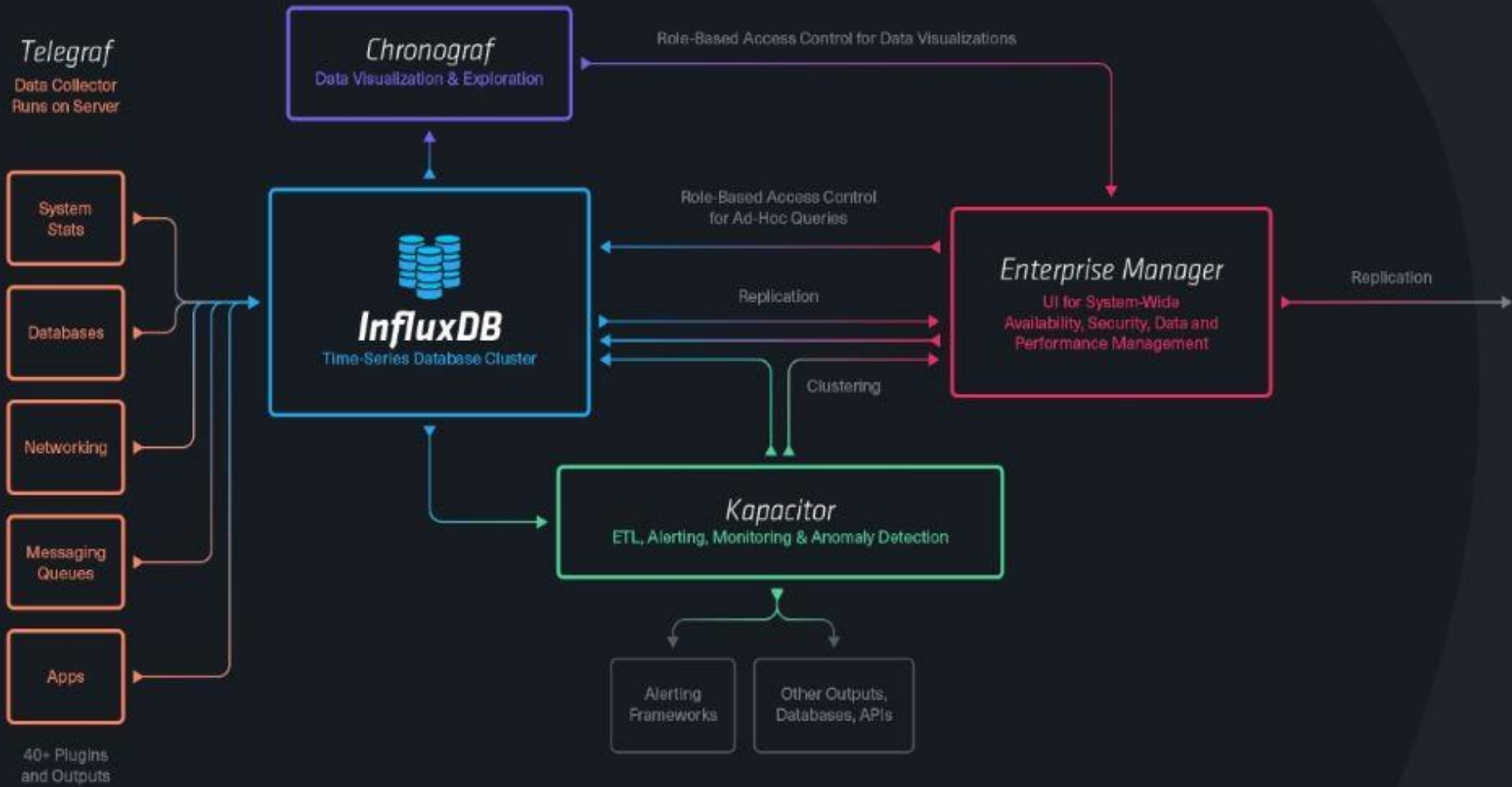


Data File Layout



TICK Stack





Testimonials

All

IoT and Sensor Data

Custom Monitoring

Real Time Analytics

OpenStack, Docker and Virtualization



References

- <https://www.influxdata.com/>
- https://en.wikipedia.org/wiki/Time_series_database

Thank You!