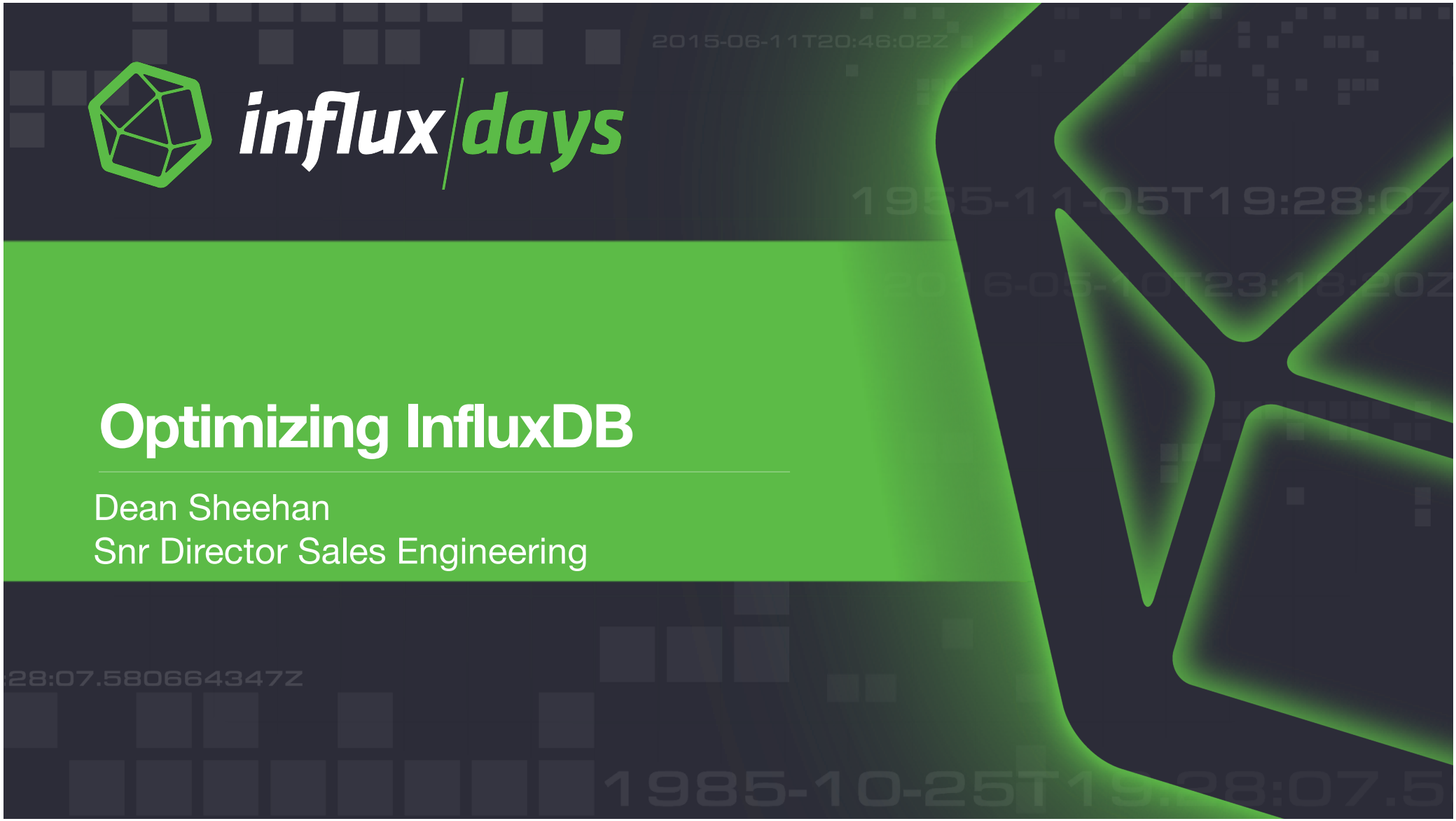

influx|days
2015-06-11T20:46:02Z
1955-11-05T19:28:07
1985-10-25T19:28:07.5
Optimizing InfluxDB
Dean Sheehan
Snr Director Sales Engineering

Agenda

- Optimizing
  - Data Modelling (Schema)
  - Ingesting Data (Writing)
  - Excreting Data ☺ (Querying)
  - Configuration
- Q&A

Data Modelling (Schema)

Line Protocol

Format: `measurement,tagset fieldset timestamp`

Measurement: i.e., `stock_prices`

Tagset:

-comma-separated set of key-value pairs in-mem

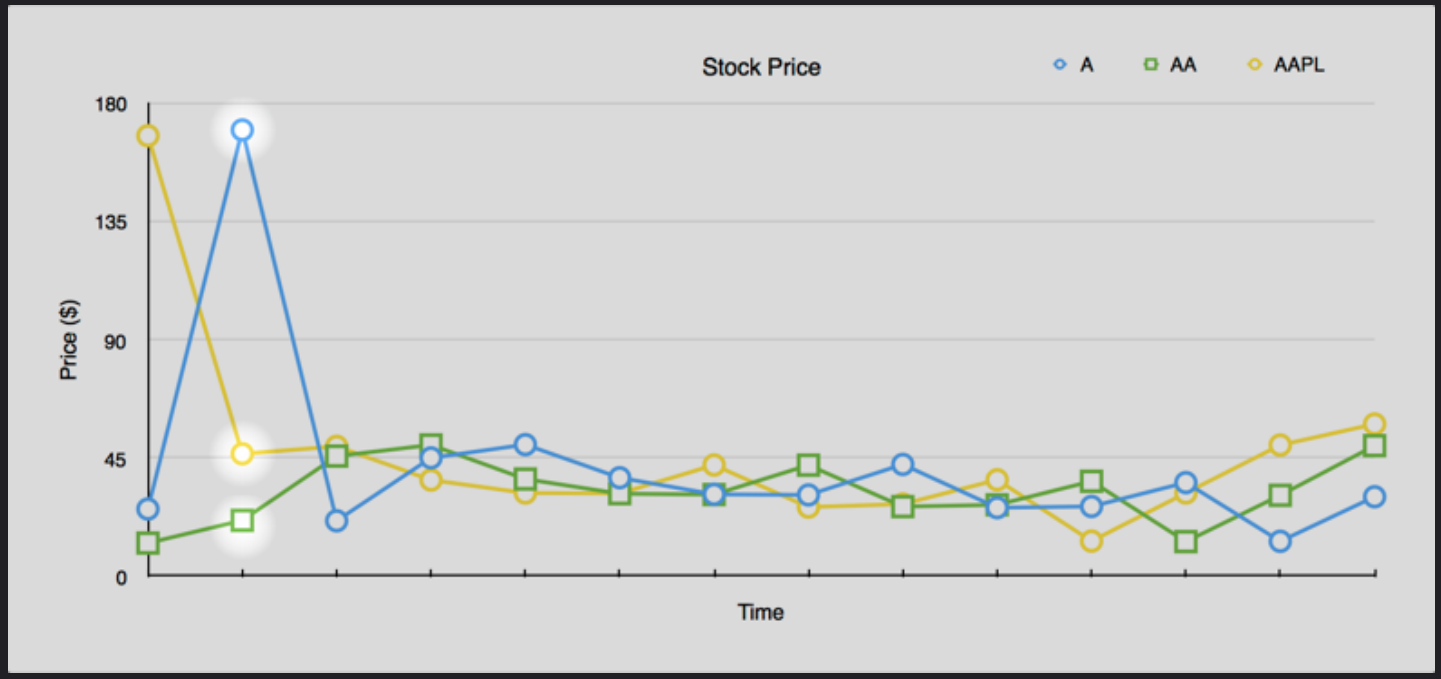
i.e., `ticker=GOOG,exchange=NYSE`

Fieldset:

-comma-separated set of key-value pairs eg:

`close=45.00,open=43.00,high=48.50,low=38.50`

Points in InfluxDB look like…




```
stock_price,ticker=A,exchange=NASDAQ close=177.03 1445299200000000000
stock_price,ticker=AA,exchange=NYSE close=32.10 144529920000000000
stock_price,ticker=AAPL,exchange=NASDAQ close=45 1445299200000000000a
```

Schema Design Goals

- Minimizing:
  - Cardinality
  - Information-encoding
  - Key lengths

- Maximizes:
  - Write performance
  - Query performance
  - Readability

Schema Insights

- A Measurement is a namespace for like metrics (SQL table)
- What to make a Measurement?
  - Logically-alike metrics; categorization
  - I.e., CPU has metrics has many metrics associated with it
  - I.e., Transactions
    - “usd”,”response_time”,”duration_ms”,”timeout”, whatever else…
- What to make a Tag?
  - Metadata; what asset these metrics belong to
  - Often this translates to “*things you need to `GROUP BY`*”
- What to make a Field?
  - Actual metrics
  - More specifically?
    - Things you need to do math on or other operations
    - Things that have high value variance...*that you don’t need to group*

DON'T ENCODE DATA INTO THE MEASUREMENT NAME

Measurement names like:

```
cpu.server-5.us-west.usage_user value=20.0 14442349820000000000

cpu.server-6.us-west.usage_user value=40.0 14442349820000000000

mem-free.server-6.us-west.usage_user value=25.0 14442349820000000000
```

Encode that information as tags:

```
cpu,host=server-5,region=us-west usage_user=20.0 14442349820000000000

cpu,host=server-6,region=us-west usage_user=40.0 14442349820000000000

mem-free,host=server-6,region=us-west usage_user=25.0 1444234982000000
```

DON'T OVERLOAD TAGS

BAD

```
cpu,server=localhost.us-west.usage_user value=2 1444234982000000000

cpu,server=localhost.us-east.usage_system value=3 1444234982000000000
```

GOOD: Separate out into different tags:

```
cpu,host=localhost,region=us-west usage_user=2.0 1444234982000000000

cpu,host=localhost,region=us-east usage_system=3.0 1444234982000000000
```

In practice, use something like Telegraf Graphite parser

Feed graphite like: `cpu.usage.eu-west.idle.percentage 100` into Telegraf configured like:

```
[[inputs.http_listener_v2]]
data_format = “graphite”
separator = "_"
templates = [
    "measurement.measurement.region.field*"
]
```

Results in following Graphite -> Telegraf transformation:

`cpu_usage,region=eu-east idle_percentage=100`

Also smaller payloads:

From:

```
cpu,region=us-west-1,host=hostA,container=containerA usage_user=35.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_system=15.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_guest=0.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_guest_nice=0.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_idle=35.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_iowait=0.2 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_irq=0.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_nice=1.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_steal=2.0 <timestamp>
cpu,region=us-west-1,host=hostA,container=containerA usage_softirq=2.5 <timestamp>
```

To:

```
cpu,region=us-west-1,host=hostA,container=containerA
usage_user=35.0,usage_system=15.0,usage_guest=0.0,usage_guest_nice=0.0,usage_idle=35.0,usage_iowait=0.2,usage_irq=0.0
,usage_irq=0.0,usage_nice=1.0,usage_steal=2.0,usage_softirq=2.5 <timestamp>
```

Cardinality per database

- The number of unique measurement, tag set, and field key combinations in an InfluxDB instance.
- In practice we generally just care about the tagset

For example:

Measurement: http

Tags:

- host -- 35 possible hosts
- appName -- 10 possible apps
- datacenter -- 4 possible DCs

Fields:

- responseCode
- responseTime

Assuming each app can reside on each host, total possible series cardinality for this measurement is
1*(35 * 10) = 350

Note 1: data center is a dependent tag since each host can only reside in 1 data center therefore adding data center as a tag does not increase cardinality

Note 2: all hosts probably don't support all 10 apps. Therefore, "real" cardinality is likely less than 350.

Ingesting Data (Write)

Write Method

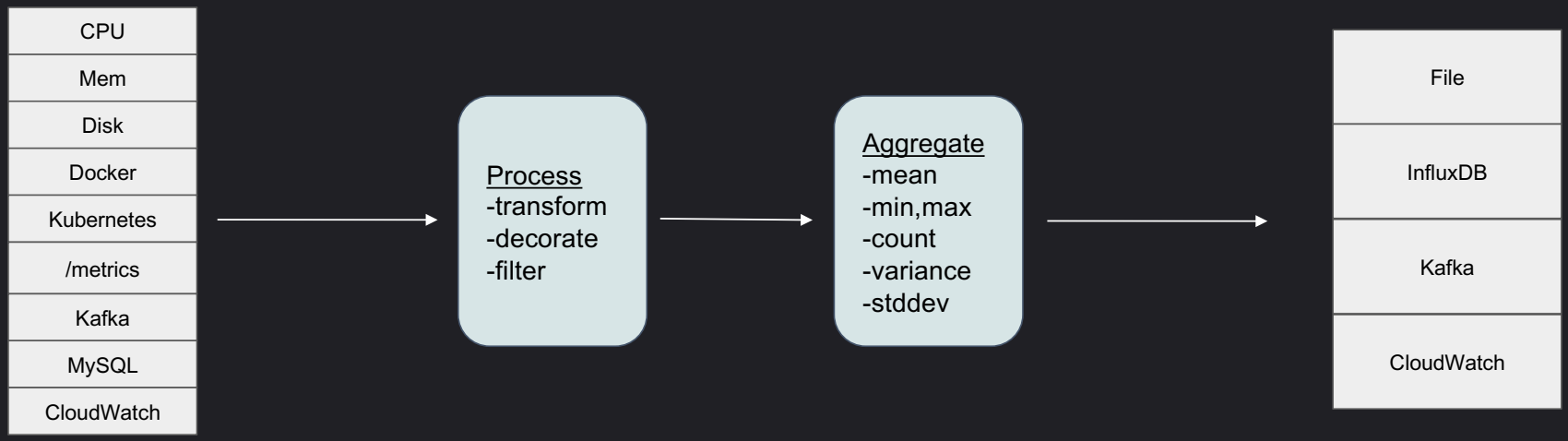
Telegraf

- Lightweight Agent/Gateway written in Go
- Plug-in architecture
- Optimized for writing to InfluxDB
  - Formatting
  - Retries
  - Modifiable batch sizes and jitter
  - Tag sorting
- Preprocessing
  - Converting tags to fields, fields to tags
  - Regex transformations
  - Renaming measurements, tags
  - Aggregate
    - Mean, Min, Max, Count, Variance, Stddev
    - Histogram
    - ValueCounter (i.e., for HTTP response codes)

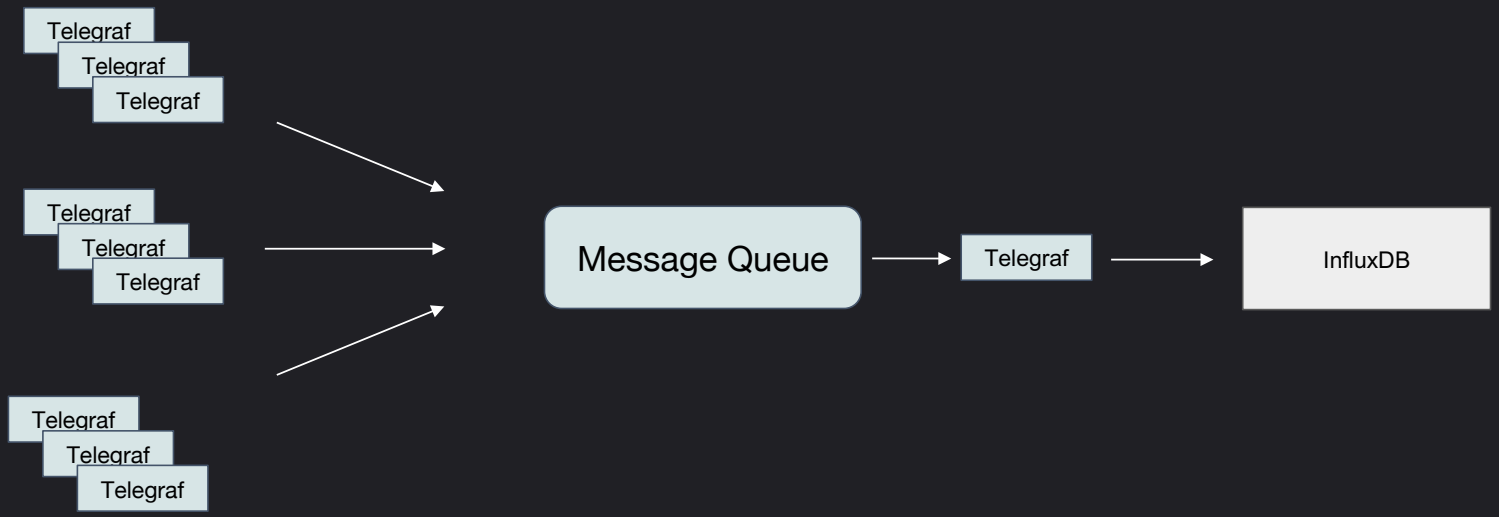
Telegraf

Out-of-the-box	Custom
Kubernetes (kubelet)	HTTP_listener_v2 (push)
Kube_inventory (apiserver)	HTTP (formatted endpoints)
Kafka (consumer)	Prometheus (/metrics)
RabbitMQ (consumer)	Exec
AMQP (mq metadata)	StatsD
Redis	
Nginx	
HAproxy	
Jolokia2	

Telegraf

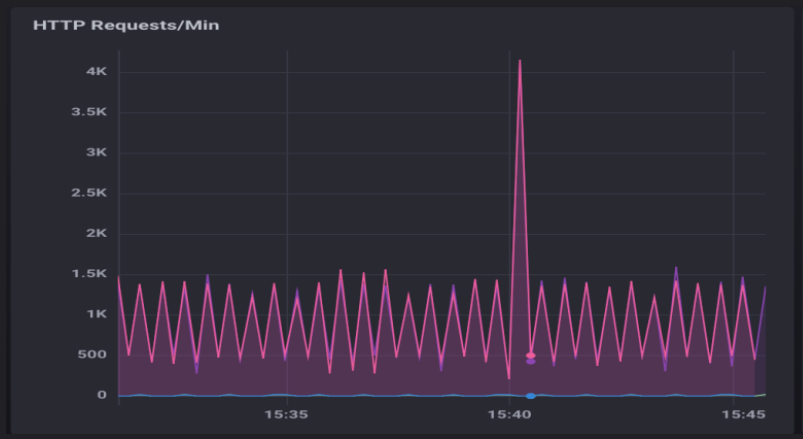

CPU
Mem
Disk
Docker
Kubernetes
/metrics
Kafka
MySQL
CloudWatch
Process
-transform
-decorate
-filter
Aggregate
-mean
-min,max
-count
-variance
-stddev
File
InfluxDB
Kafka
CloudWatch

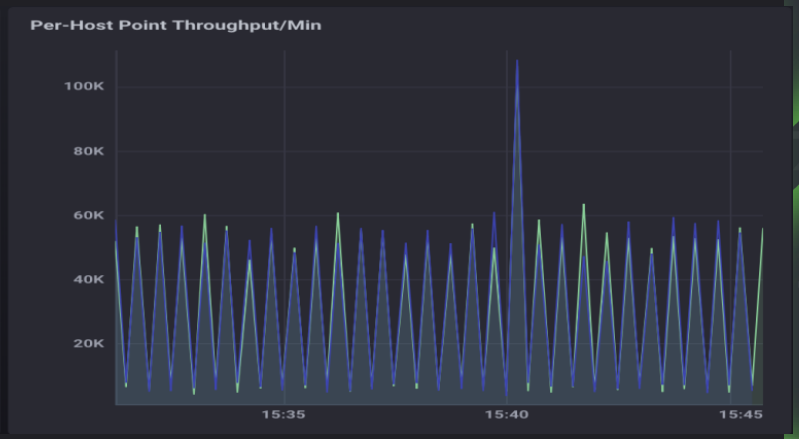
Telegraf

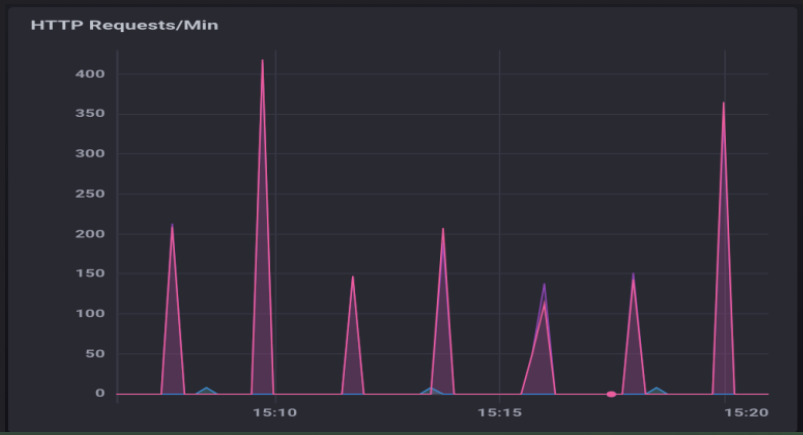


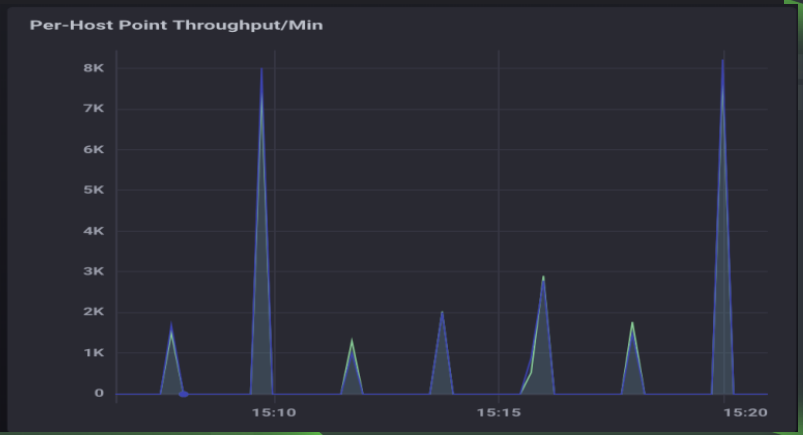
Smoothing the ingestion helps....

Which is better?

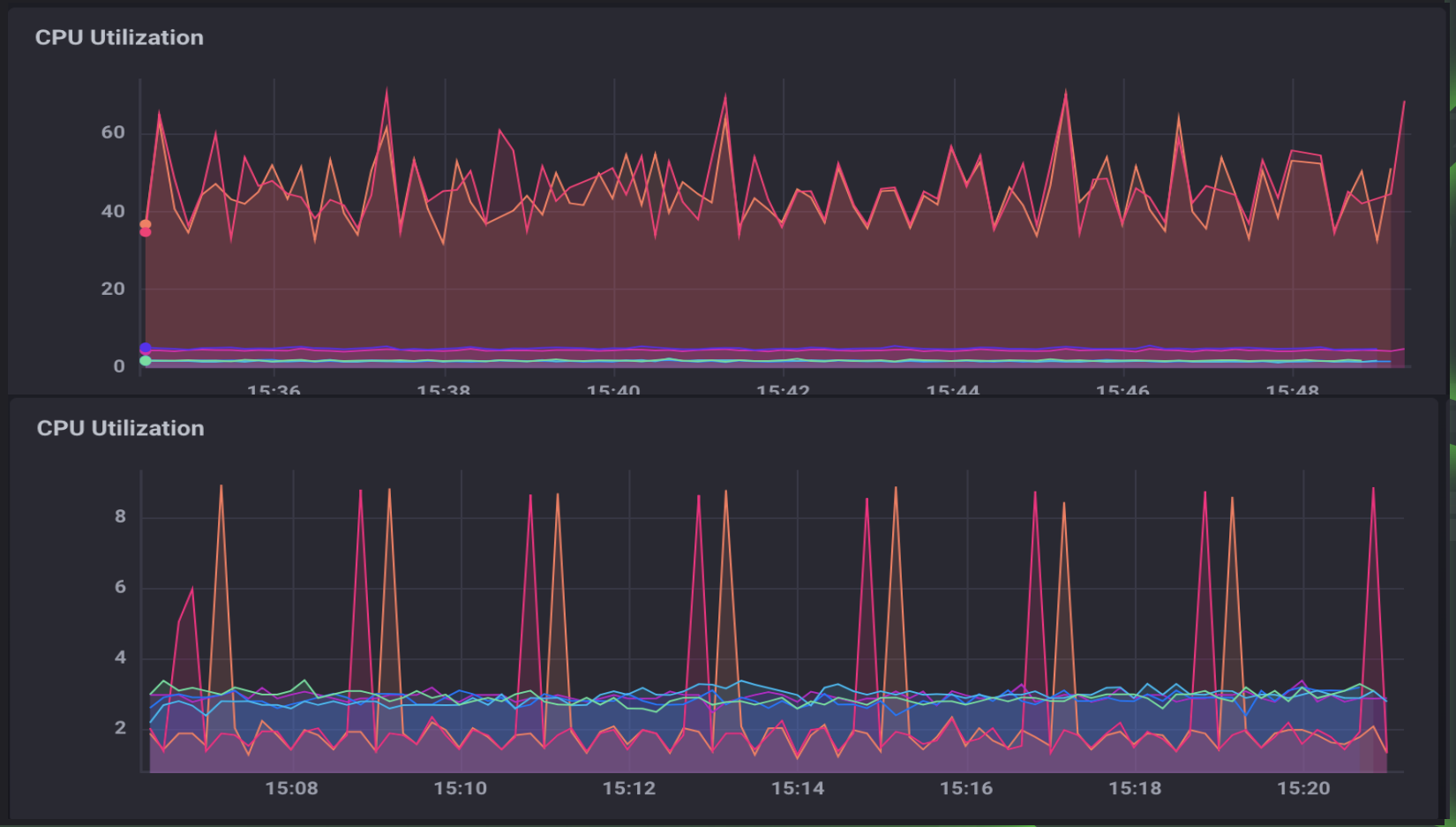








Which is better?



Excreting Data ☺ (Queries)

Query Performance

- Boundaries!
  - Time-bounding and series-bounding with `WHERE` clause
  - `SELECT *` generally not a best practice
- Agg functions instead of raw queries
  - `SELECT mean(<field>)` **>** `SELECT <field>`
- Subqueries
  - When appropriate, process data from an already processed subset of data
  - `SELECT min("max") FROM (SELECT max("usage_user") FROM cpu WHERE time > now() - 5d GROUP BY time(5m))`

Query Performance

- Streaming functions & batch functions
  - Batch funcs
    - percentile(), spread(), stddev(), median(), mode(), holt-winters
  - Stream funcs
    - mean(),bottom(),first(),last(),max(),top(),count(),etc.
- Distributed functions (clusters only) & local functions
  - Distributed
    - first(),last(),max(),min(),count(),mean(),sum()
  - Local
    - percentile(),derivative(),spread(),top(),bottom(),elapsed(),etc.

Configuration

Configuration

TSI - How does it help

- Memory use:
  - TSI-->30M series = 1-2GB
  - In-mem-->30M series = *inconceivable*
- Startup performance. Startup time should be insignificant, even for very large indexes.
- The tsi1 index has its own index, while the inmem index is shared across all shards.
  - Write performance--tsi1 index will only need to consult index files relevant to the hot data.

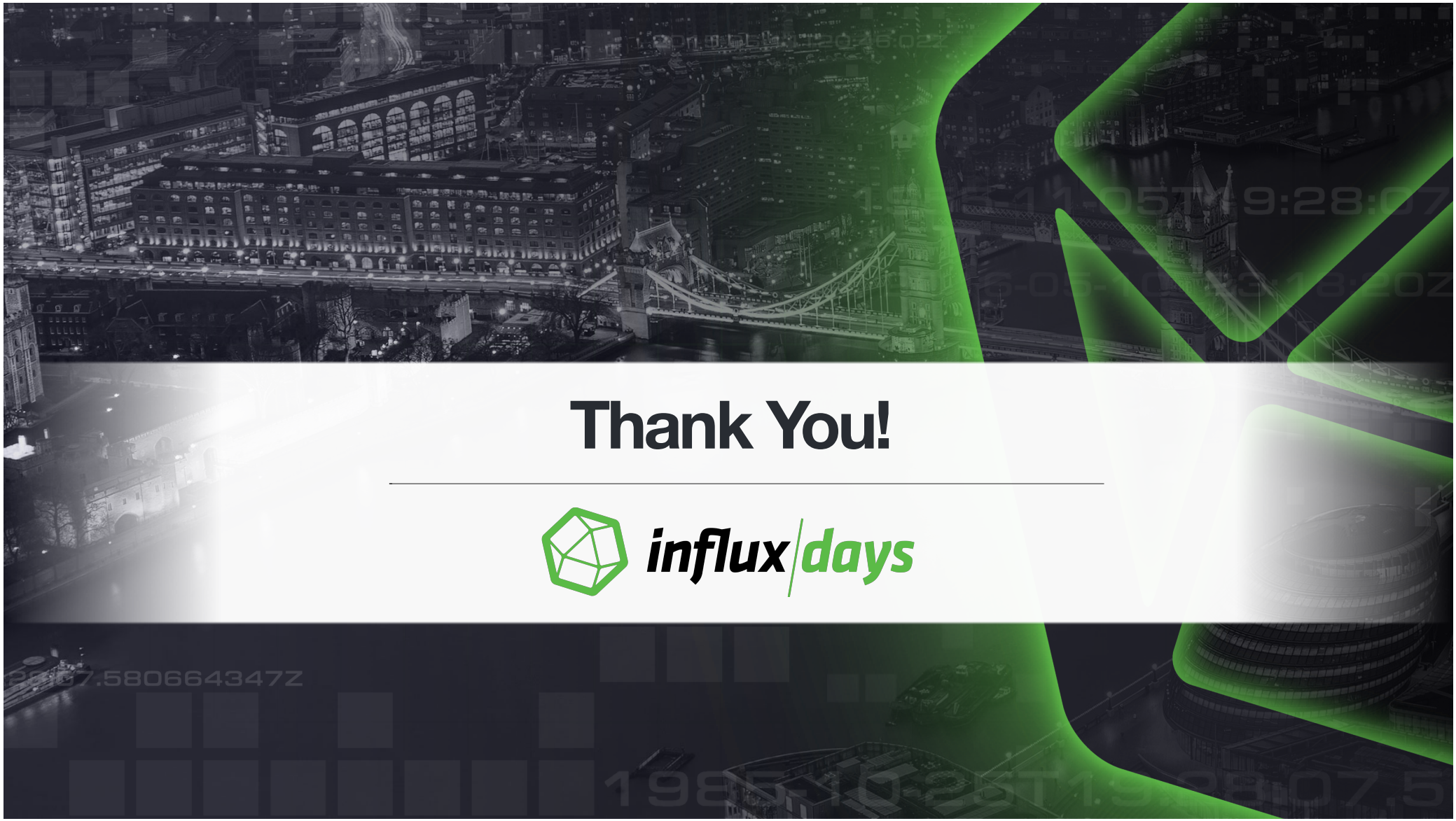
Tuning Parameters

- Max-concurrent-queries
- Max-select-point
- Max-select-series
- Rate limiting compactions
  - Max-concurrent-compactions
  - Compact-full-write-cold-duration
  - Compact-throughput (*new in v1.7!*)
  - Compact-throughput-burst (*new in v1.7!*)

Tuning Parameters

- Cache-max-memory-size
- Cache-snapshot-...
- Max-series-per-database
- *Max-values-per-tag*

- *Fine Grained Auth instead of multiple databases*


Thank You!
influx|days


2015-06-11T20:46:02Z
1955-11-05T19:28:07
28:07.580664347Z
1985-10-25T19:28:07.5
influx/days