NSF/IUCRC CAC PROJECT

Experience of Storing and Querying Monitoring Data of Large-scale High Performance Computing Platforms

Jie Li
Doctoral Student, TTU
04/15/2020

Advisors:
Mr. Jon Hass, SW Architect, Dell Inc.
Dr. Alan Sill, Managing Director, HPCC, TTU
Dr. Yong Chen, Associate Professor, CS Dept, TTU

AGENDA

- Time Series, Time Series DBs, InfluxDB
- Problems & Challenges
- Hands-on Experience & Efforts
- Summary & Demo

TIME SERIES

Definition:

Time Series is an ordered sequence of values of a variable (e.g. temperature) at equally spaced time intervals (e.g. hourly)

Uses:

- Time Series Analysis: explore how a given variable changes over time
- Regression Analysis: examine how the changes associated with a specific variable can cause shifts in other variables over the same time period
- Time Series Forecasting: use information regarding historical values and associated patterns to predict future activity

Naqvi, Syeda Noor Zehra, Sofia Yfantidou, and Esteban Zimányi. "Time series databases and influxdb." *Studienarbeit, Université Libre de Bruxelles* (2017).

TIME SERIES DBS

Definition:

A Time Series Database (TSDB) is a database type which is optimized for time series or time-stamped data.

Properties:

- Data Location: co-locate chucks of data within the same time range on the same physical part of the database cluster
- Fast, Easy Range Queries: keep the co-related data together to ensure that the range queries are fast
- High Write Performance: ensure high availability and high performance for both read and write operations during peak loads

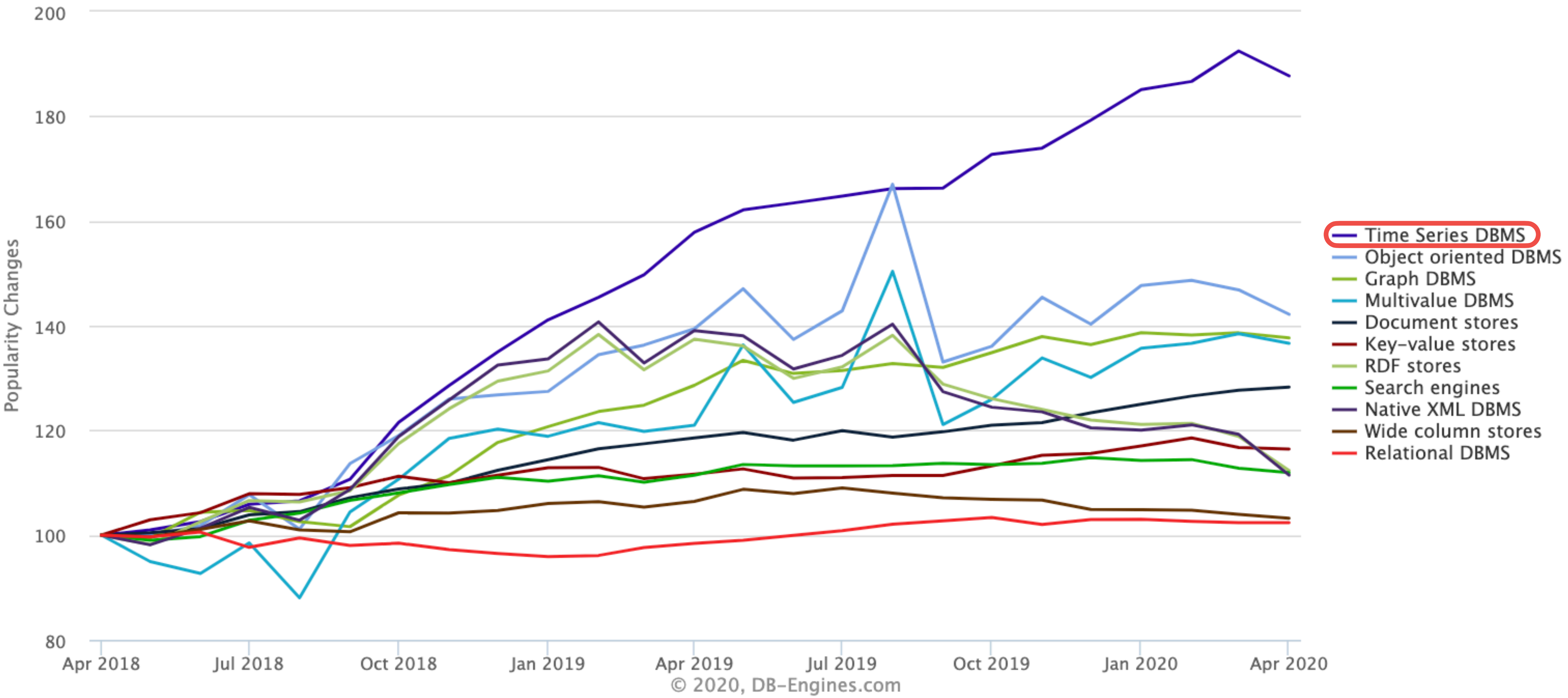
Naqvi, Syeda Noor Zehra, Sofia Yfantidou, and Esteban Zimányi. "Time series databases and influxdb." *Studienarbeit, Université Libre de Bruxelles* (2017).

Properties(cont.):

- Data Compression: provide functionality to perform roll-ups in such scenarios for data compaction
- Scalability: take care of scale by introducing functionalities that are only possible when treat time as first concern
- Usability: include functions and operations that are common to time series data analysis
  - data retention policies
  - continuous queries
  - flexible time aggregations
  - range queries etc.

Naqvi, Syeda Noor Zehra, Sofia Yfantidou, and Esteban Zimányi. "Time series databases and influxdb." *Studienarbeit, Université Libre de Bruxelles* (2017).

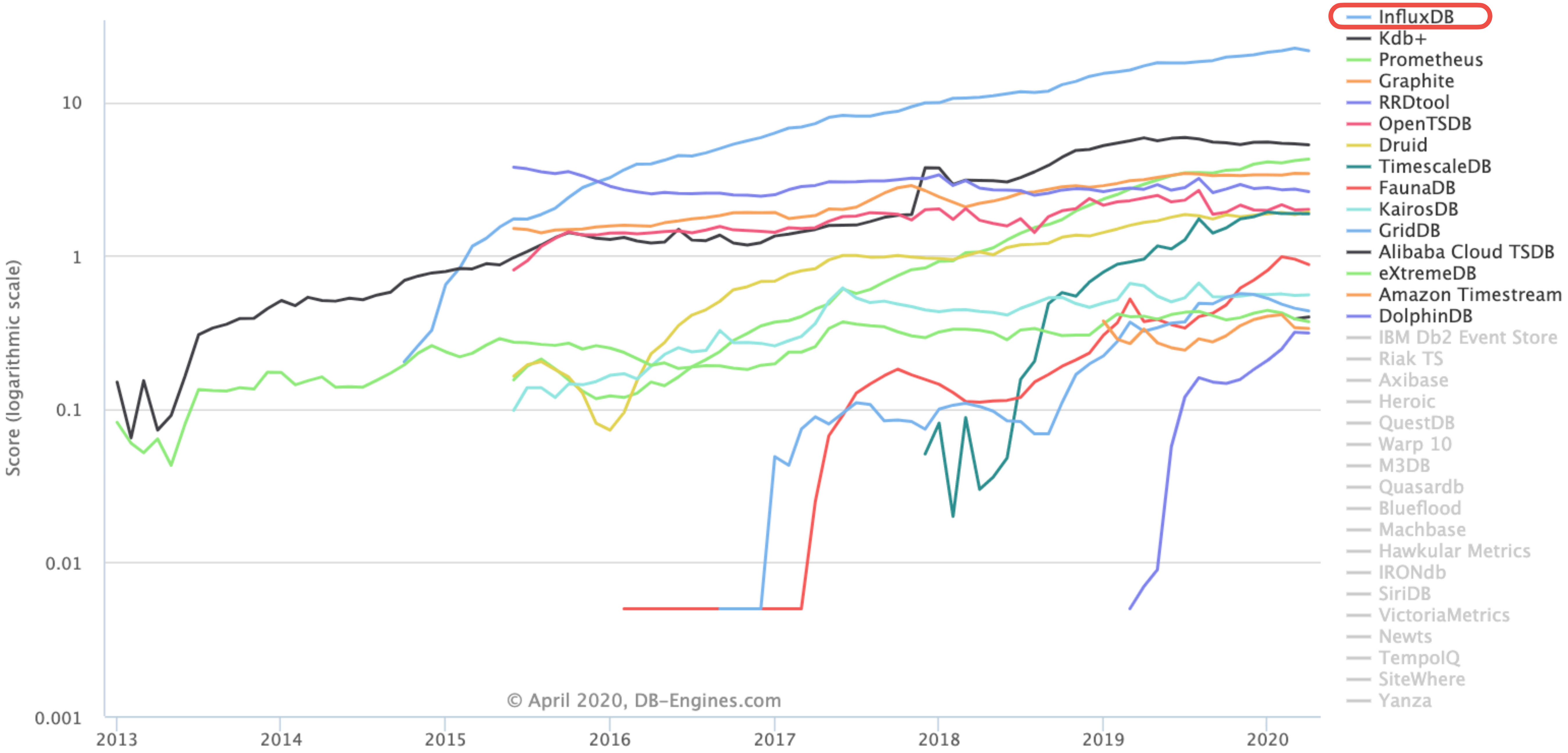
TIME SERIES DBS



DBMS (Database Management System) Popularity broken down by database model

https://db-engines.com/en/ranking_categories

TIME SERIES DBS



Ranking of Time Series DBMS

https://db-engines.com/en/ranking_trend/time+series+dbms

INFLUXDB

InfluxDB:

- Open-source schemaless time series database
- Written in Go and optimized for fast, high-availability storage and retrieval of time series data
- Provides an SQL-like query language

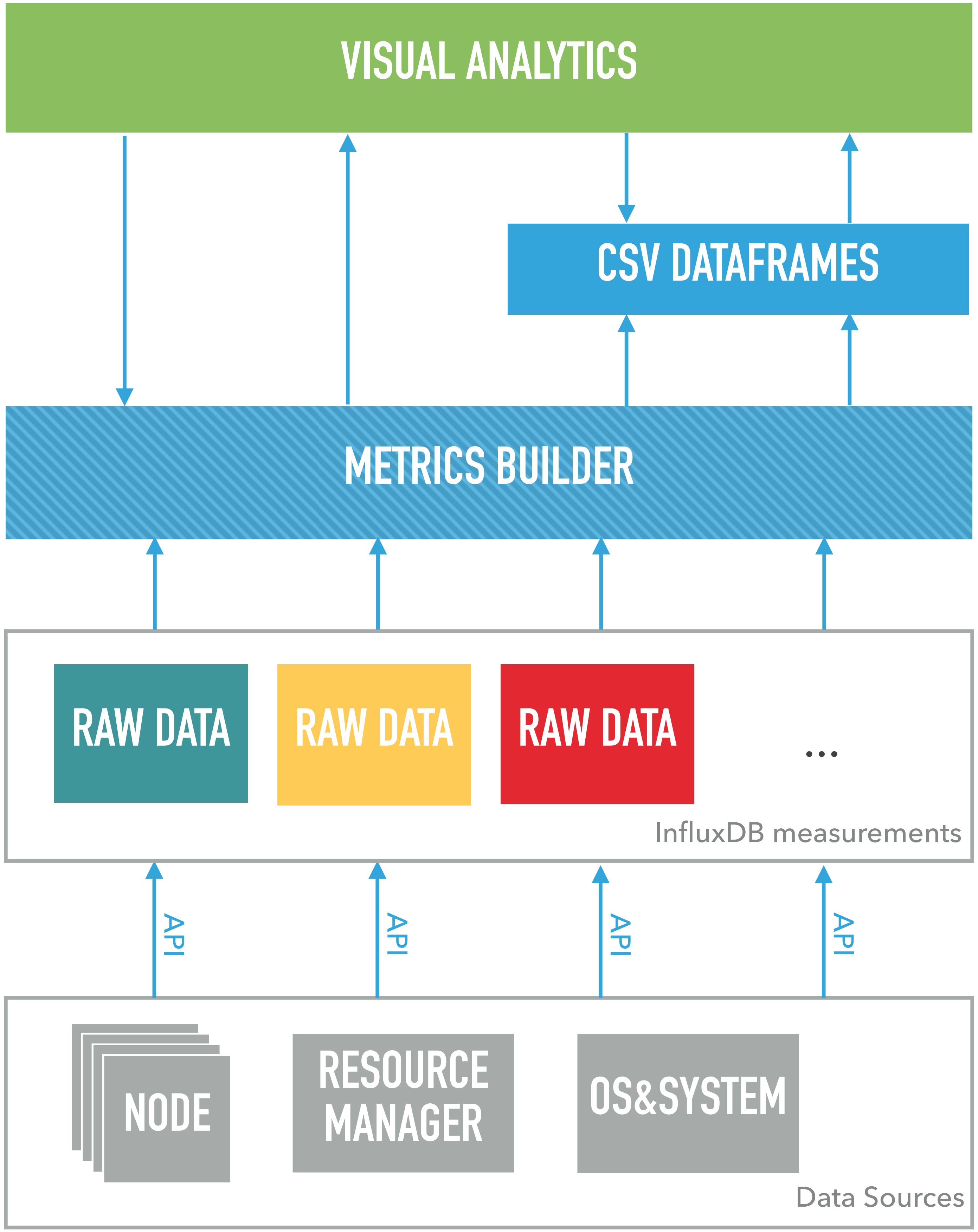
Data model:

- measurement
- timestamp
- fields
- tags

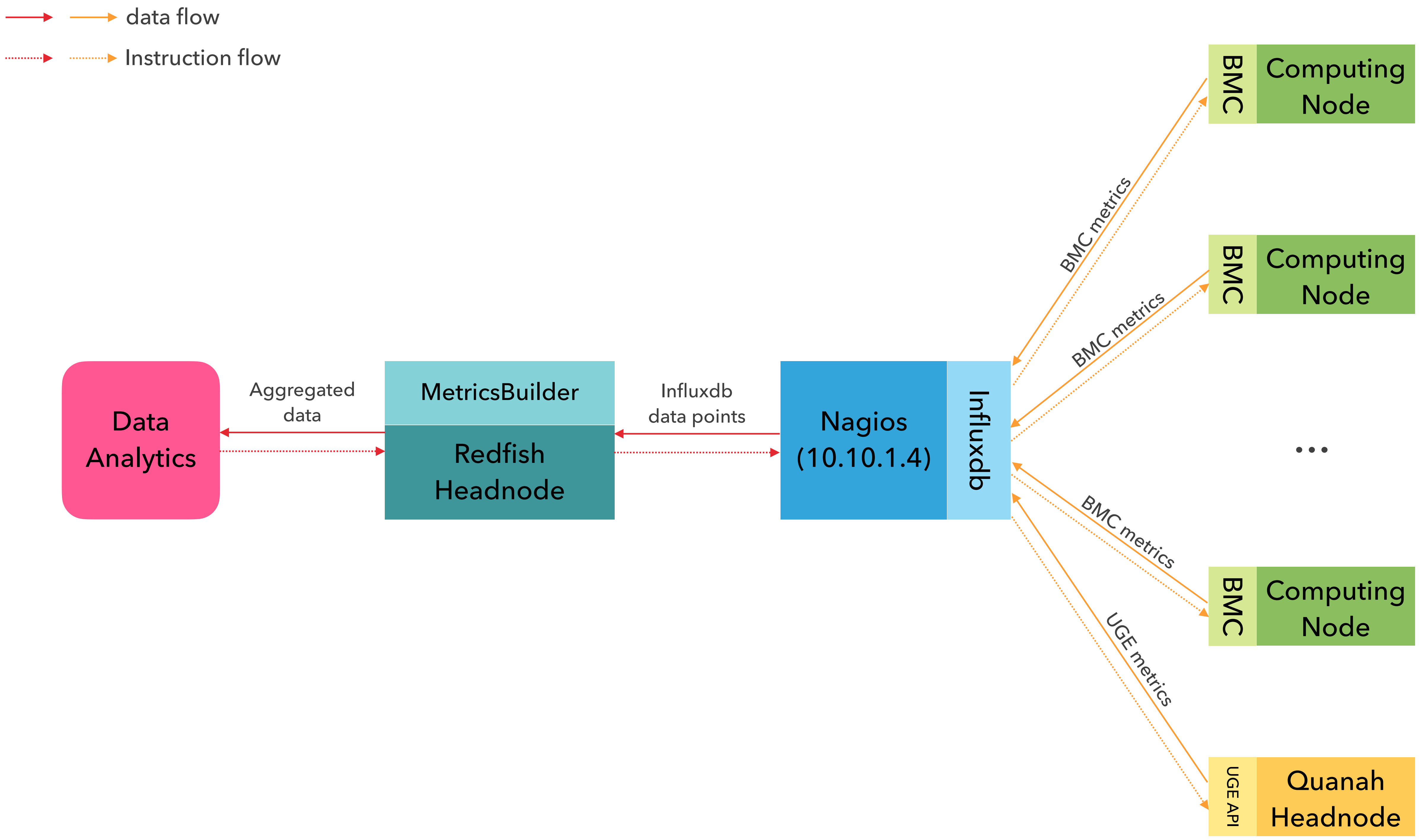
```
<measurement>[,<tag_key>=<tag_value>[,<tag_key>=<tag_value>]]
<field_key>=<field_value>[,<field_key>=<field_value>] [<timestamp>]
```

https://en.wikipedia.org/wiki/InfluxDB
Salmaan, Shabana, and Danish Amjad. "INFO-H415–Advanced Databases Project: Time Series Database with InfluxDB Academic Year 2018-2019."

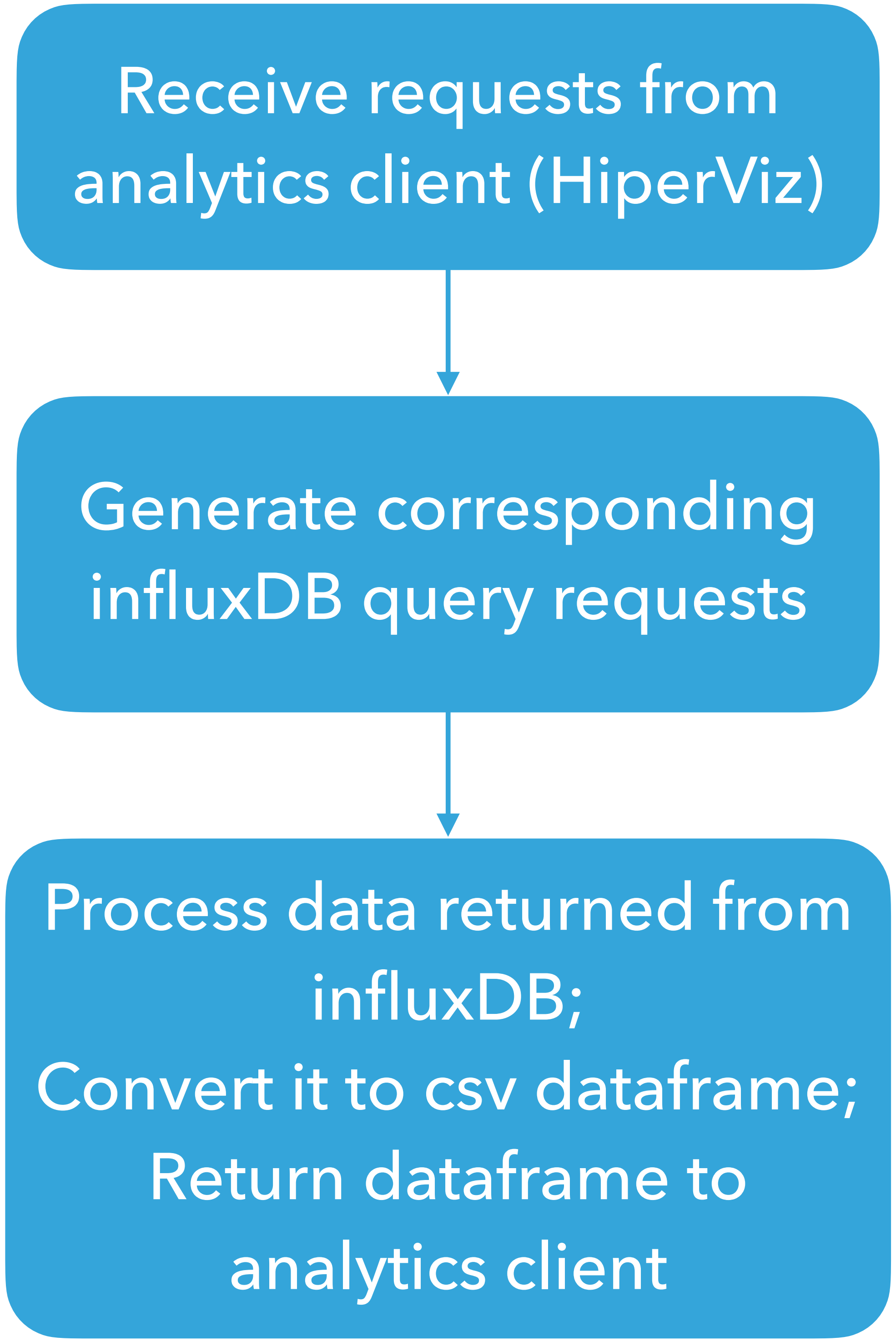
MONITORING FRAMEWORK



MONITORING FRAMEWORK



METRICS BUILDER WORKFLOW



Receive requests from analytics client (HiperViz)

Generate corresponding influxDB query requests

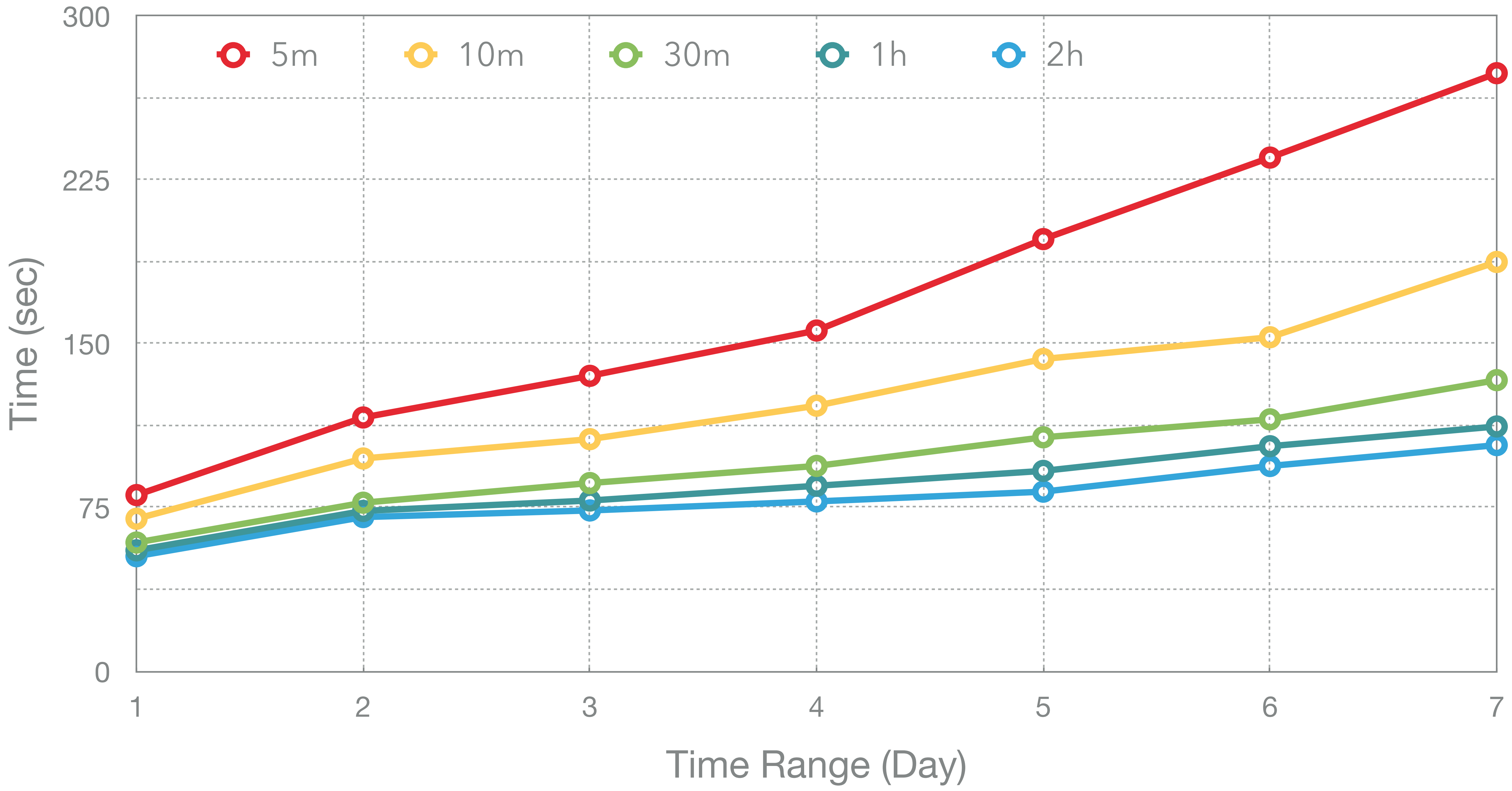
Process data returned from influxDB;
Convert it to csv dataframe;
Return dataframe to analytics client

time range:
e.g. 2019-04-20T00:00:00Z, 2019-04-21T00:00:00Z
time interval: e.g. 30m
data type: e.g. Max, Min, Average

```
SELECT max(CPU_Usage) FROM CPU_Usage WHERE
host='10.101.1.1' AND time >= '2020-04-10T00:00:00Z' AND
time <= '2020-04-11T00:00:00Z' GROUP BY time(30m)
```

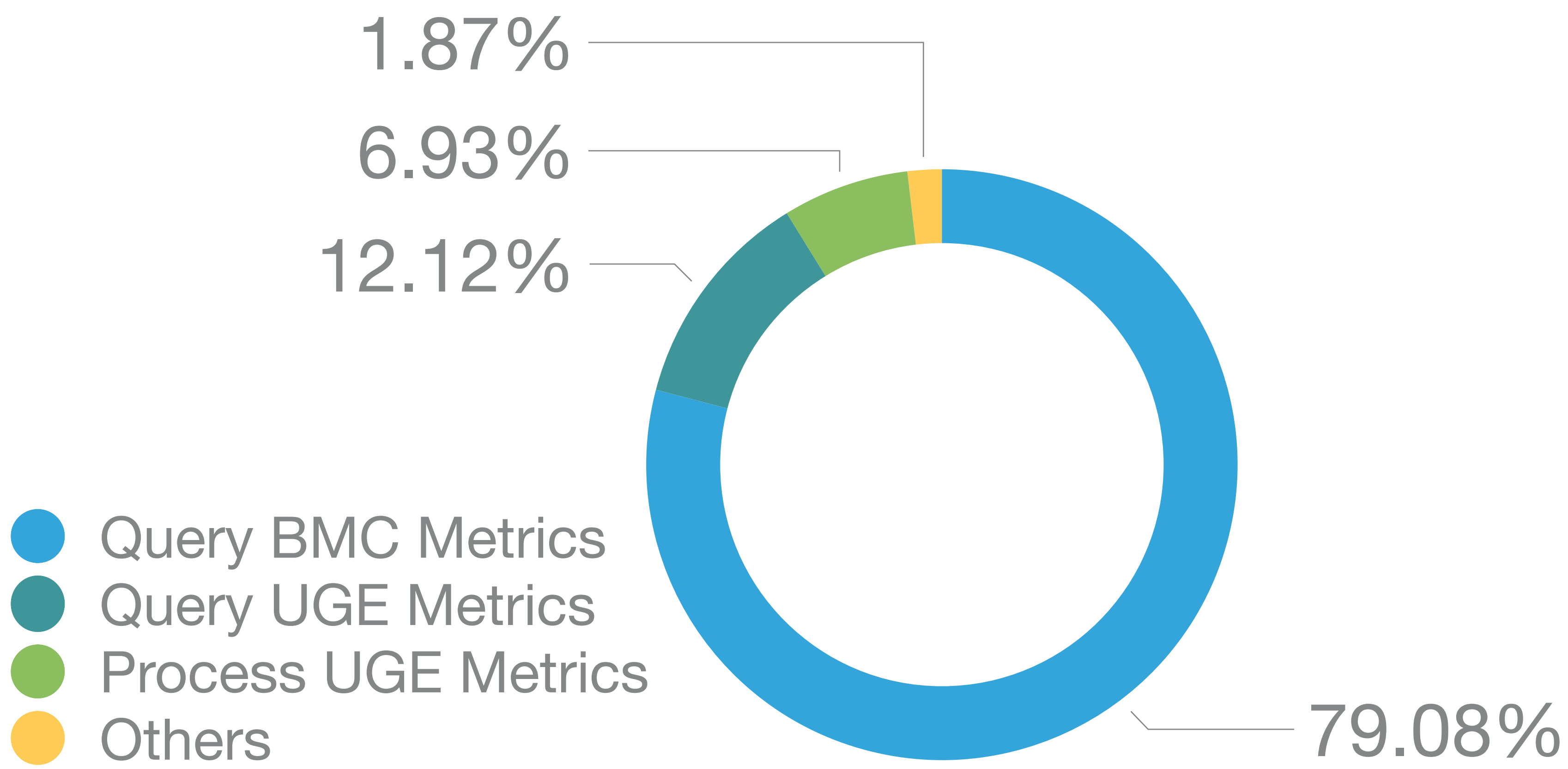
- Move data processing from front end to back end
- Provide a uniform API to analytics client
- Act as a middleware that deals with different database design

PERFORMANCE



Query and processing time of MetricsBuilder

ANALYSIS



Time Occupation of Query and Processing Metrics

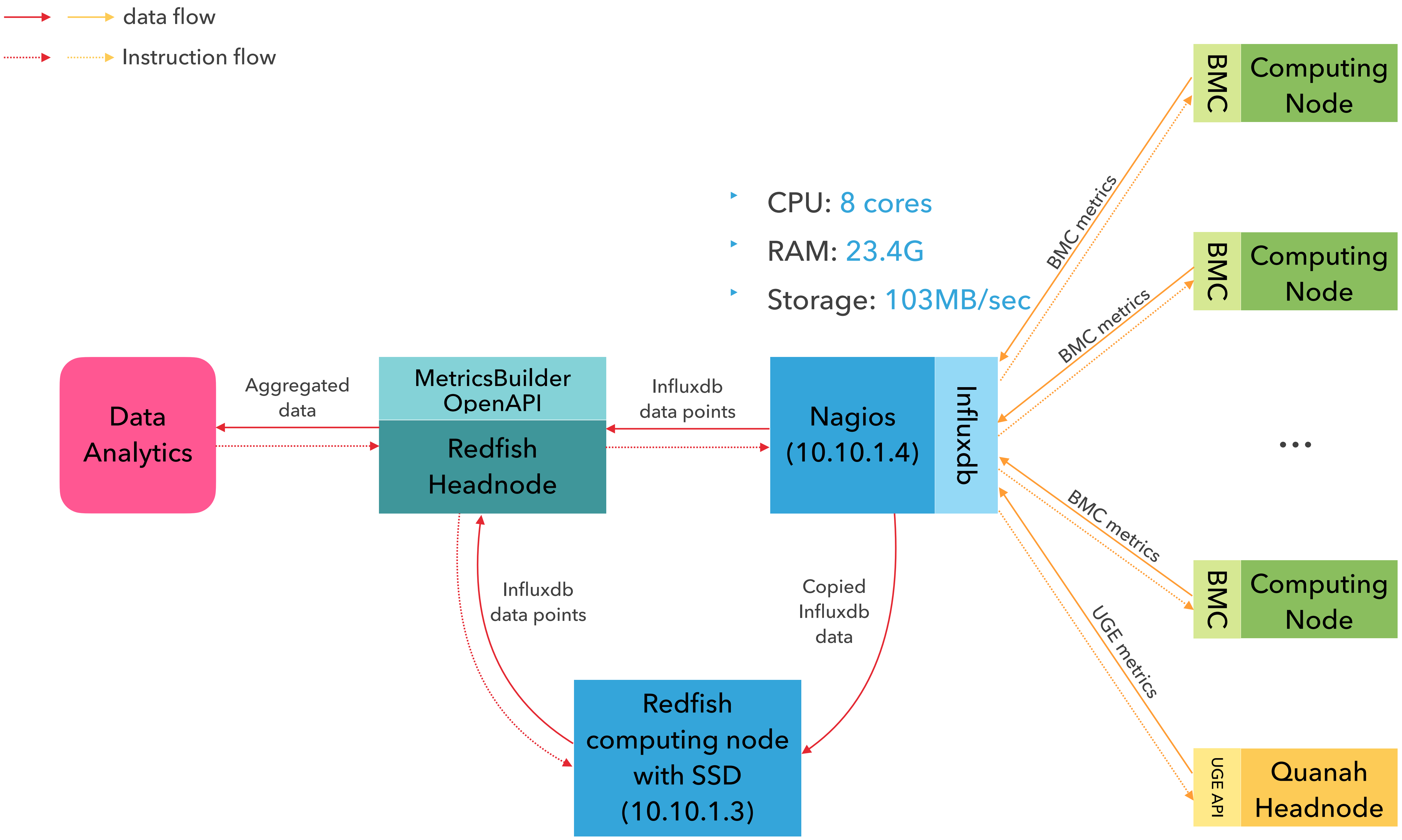
Queuing database occupies ~90% of total running time.

Profiling Python script using **cProfile**

APPROACH-1

Switch from HDD to SSD

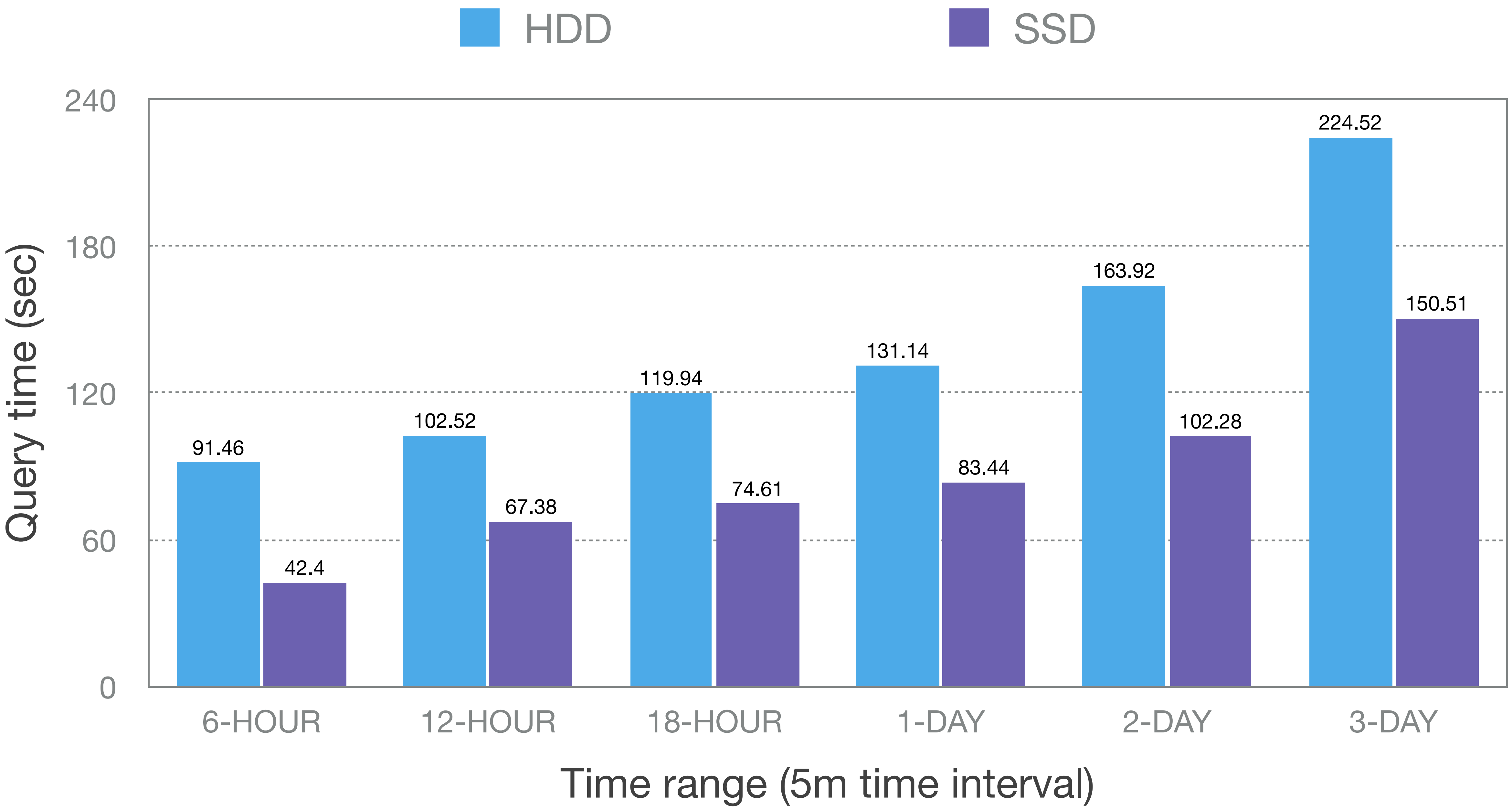
EFFORTS - USE SSD



- CPU: 8 cores
- RAM: 23.4G
- Storage: 103MB/sec

- CPU: 16 cores
- RAM: 94.2G
- Storage: 391MB/sec

IMPROVEMENT - USE SSD



Query data from SSD performs 1.5x ~ 2.1x faster than HDD

APPROACH-2

Redesign Schema

EFFORTS - REDESIGN SCHEMA

Metrics Saved in different measurements(tables):

- **BMC**:
  - CPU_Temperature
  - Inlet_Temperature
  - Fan_Speed
  - Node_Power_Usage
- **UGE**:
  - Job_Info
  - CPU_Usage
  - Memory_Usage

Mar. 14, 2019, 11:44 PM

- **BMC**:
  - cluster_unified_metrics
- **UGE**:
  - Current_Jobs_ID
  - qu_1236124 etc.
  - …

Oct. 17, 2019, 9:09 PM

EFFORTS - REDESIGN SCHEMA

Previous Schema:

Measurement: CPU_Temperature

Query: select * from CPU_Temperature WHERE host='10.101.1.1' limit 1

time	tags	fields
(epoch time)	cluster host location	CPU1 Temp CPU2 Temp GET_proessing_time Inlet Temp cpuLowerThresholdCritical cpuLowerThresholdNonCritical cpuUpperThresholdCritical cpuUpperThresholdNonCritical error

EFFORTS - REDESIGN SCHEMA

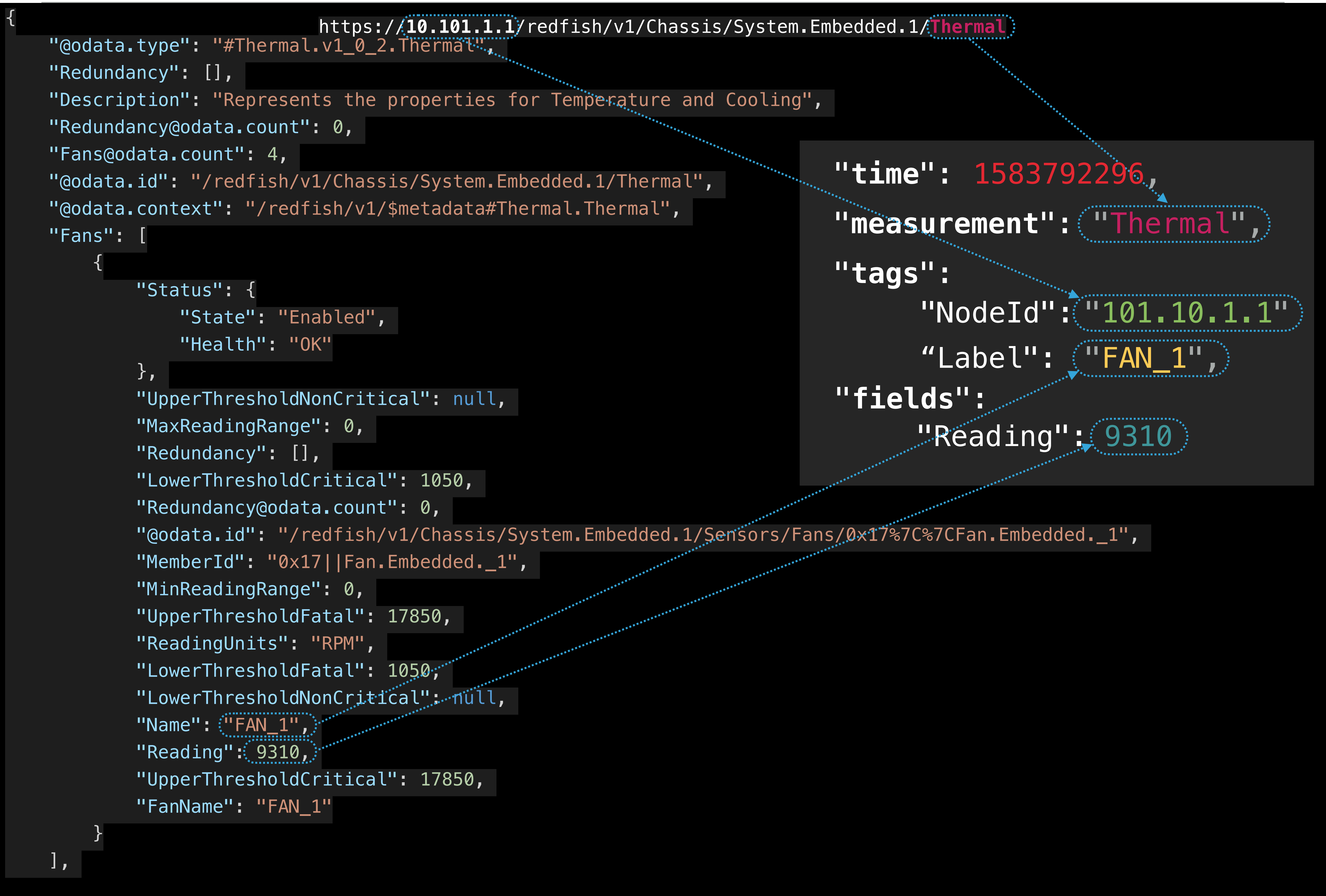
Previous Schema:

Measurement: qu_1236124

Query: select * from qu_1236124 limit 1

time	tags	fields
(epoch time)	cluster location	CPUCores app_name error id nodes_address startTime "Thu Apr 02 22:04:19 CDT 2020" 1585883059 state submitTime "Thu Apr 02 15:08:42 CDT 2020" 1585858122 total_nodes user

EFFORTS - REDESIGN SCHEMA



```
https://10.101.1.1/redfish/v1/Chassis/System.Embedded.1/Thermal
```

```
{
    "@odata.type": "#Thermal.v1_0_2.Thermal",
    "Redundancy": [],
    "Description": "Represents the properties for Temperature and Cooling",
    "Redundancy@odata.count": 0,
    "Fans@odata.count": 4,
    "@odata.id": "/redfish/v1/Chassis/System.Embedded.1/Thermal",
    "@odata.context": "/redfish/v1/$metadata#Thermal.Thermal",
    "Fans": [
        {
            "Status": {
                "State": "Enabled",
                "Health": "OK"
            },
            "UpperThresholdNonCritical": null,
            "MaxReadingRange": 0,
            "Redundancy": [],
            "LowerThresholdCritical": 1050,
            "Redundancy@odata.count": 0,
            "@odata.id": "/redfish/v1/Chassis/System.Embedded.1/Sensors/Fans/0x17%7C%7CFan.Embedded._1",
            "MemberId": "0x17||Fan.Embedded._1",
            "MinReadingRange": 0,
            "UpperThresholdFatal": 17850,
            "ReadingUnits": "RPM",
            "LowerThresholdFatal": 1050,
            "LowerThresholdNonCritical": null,
            "Name": "FAN_1",
            "Reading": 9310,
            "UpperThresholdCritical": 17850,
            "FanName": "FAN_1"
        }
    ],
```

```
"time": 1583792296,
"measurement": "Thermal",
"tags":
      "NodeId": "101.10.1.1"
      “Label": "FAN_1",
"fields":
      "Reading": 9310
```

EFFORTS - REDESIGN SCHEMA

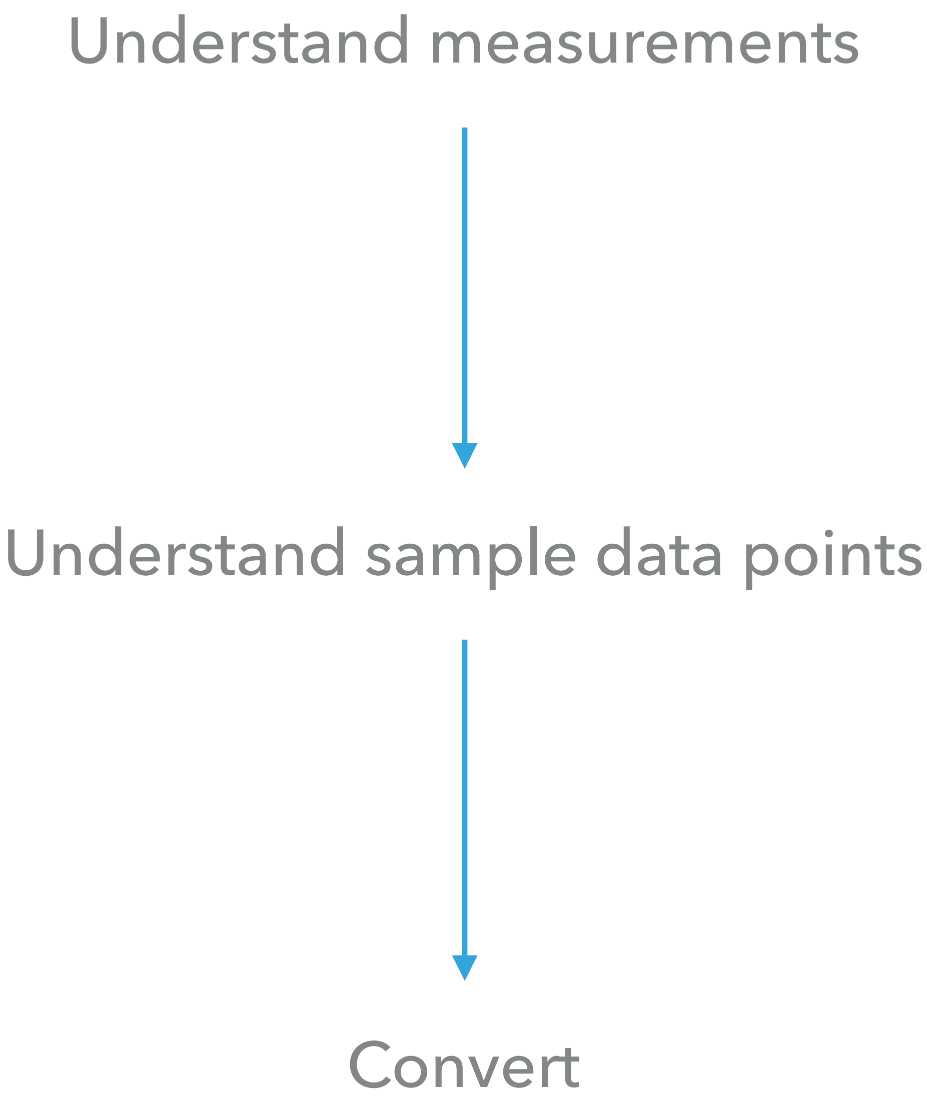
```
"time": 1583792296,
"measurement": "UGE",
"tags":
     “NodeId": "101.10.1.1"
     “Label”: "CPUUsage",
"fields":
     “Reading": 0.50
```

```
"time": 1583792296,
"measurement": "NodeJobs",
"tags":
     "NodeId": "101.10.1.1"
"fields":
     "JobList": ["123456",
                 "123457"]
```

```
"time": 1583792296,
"measurement": "JobsInfo",
"tags":
     "JobId": "123456"
     "Queue": "quanah"
"fields":
     "StartTime": 1583792200
     "SubmitTime": 1583792200
     "TotalNodes": 1
     "NodeList": ["101.10.1.1"]
     "CPUCores": 10
     "JobName": "test"
     "User": "abc"
```

Only update when a new job is submitted

EFFORTS - REDESIGN SCHEMA



Understand measurements

Understand sample data points

Convert

All measurements :845,241

- Numerical measurements :10
- Job measurements :845,217
- Other measurements :14

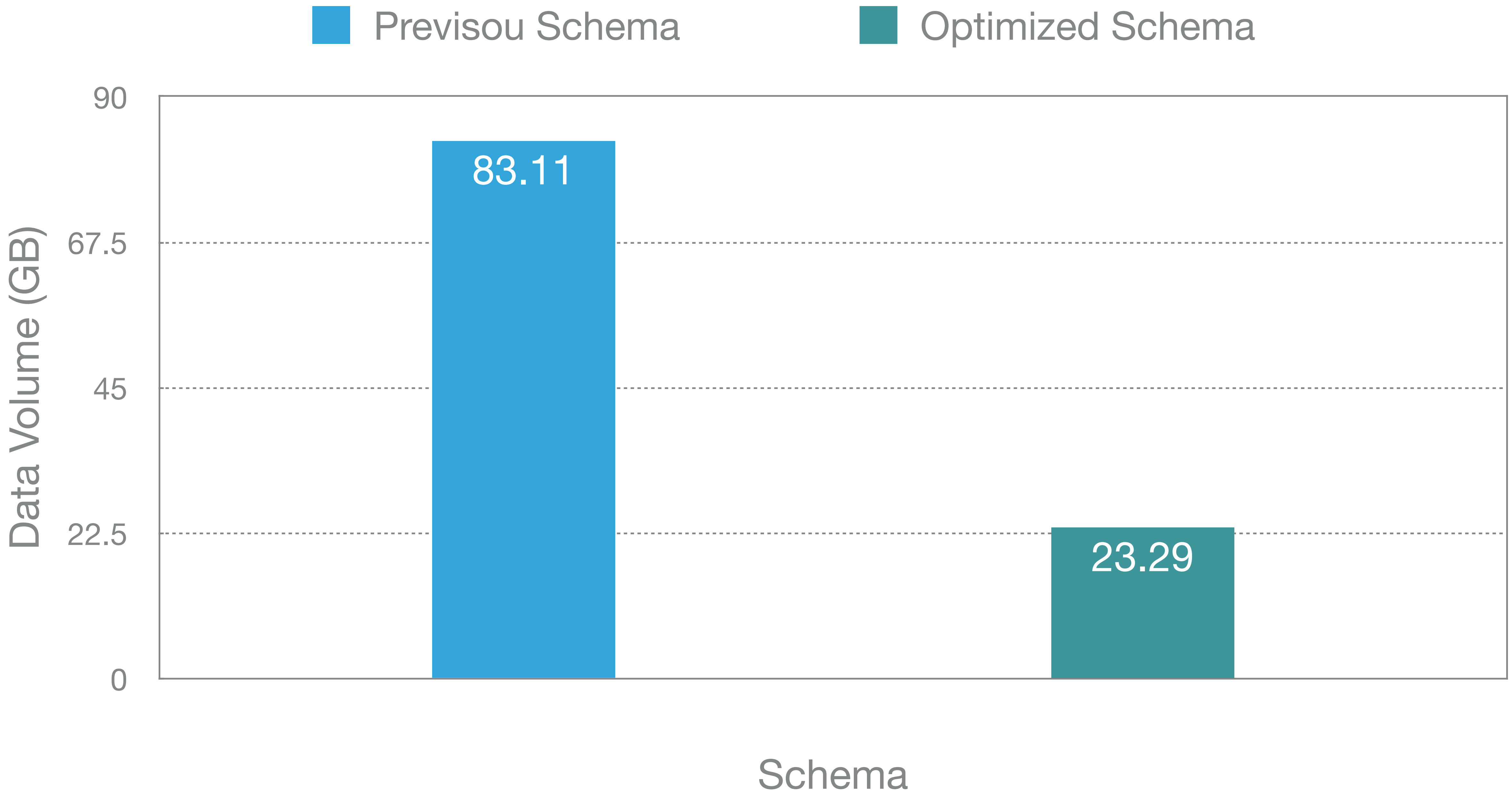
(As of Mar. 13, 2020)

All measurements : 5

JobsInfo	NodeJobs	Power	Thermal	UGE

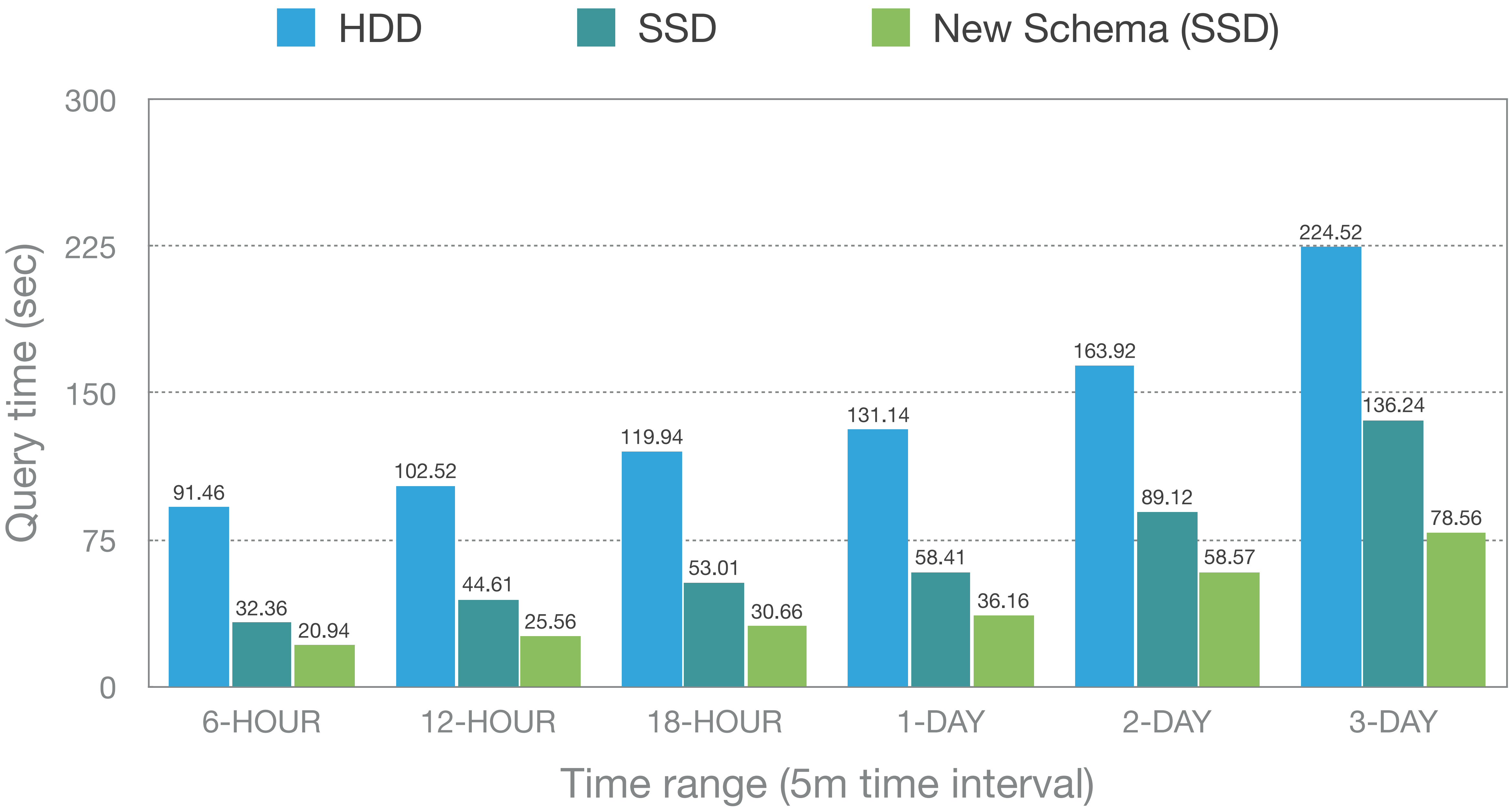
EFFORTS - REDESIGN SCHEMA

March 14, 2019 12:00:00AM – April 10, 2020 12:00:00AM



Data volume in Optimized Schema is 28.02% of the one in previous schema

IMPROVEMENT - REDESIGN SCHEMA



Query data from New Schema performs 1.6x ~ 1.76x faster than previous schema

Query data from New Schema performs 2.8x ~ 4.3x faster than previous schema on HDD

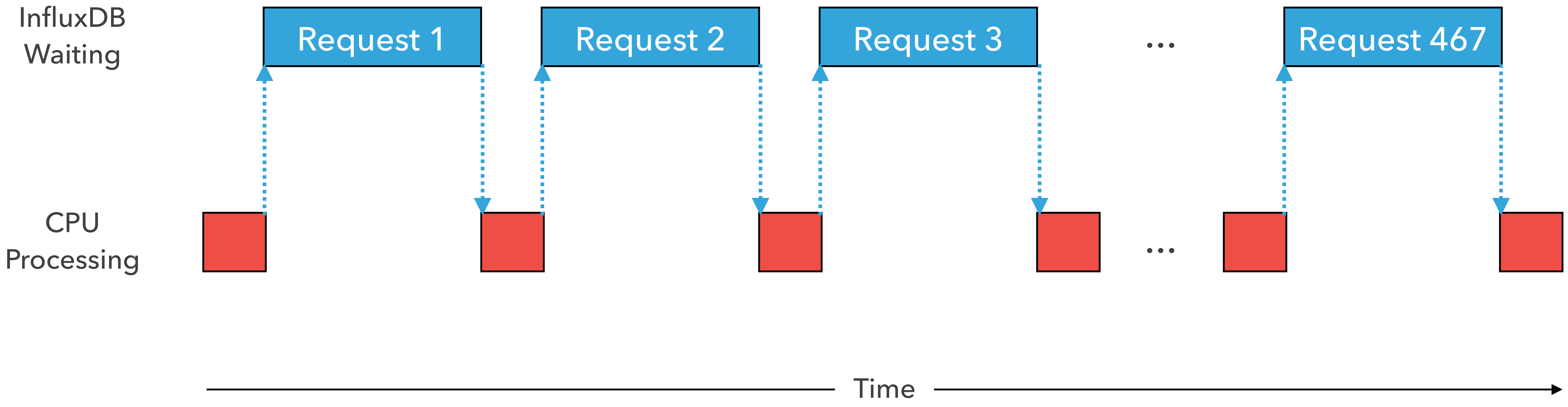
APPROACH-3

Concurrent Processing

EFFORTS - CONCURRENT PROCESSING

Request

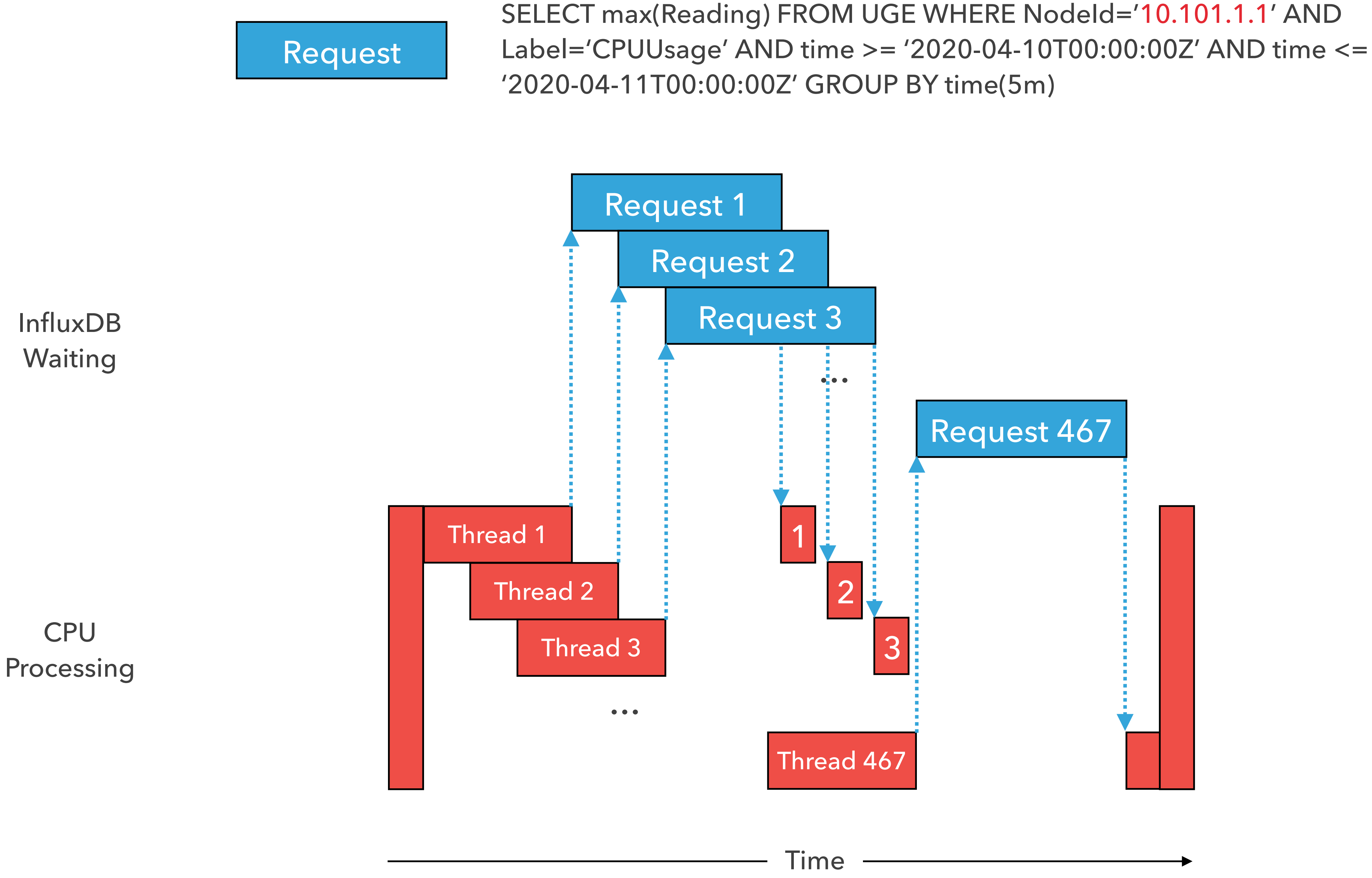
```
SELECT max(Reading) FROM UGE WHERE NodeId='10.101.1.1' AND
Label='CPUUsage' AND time >= '2020-04-10T00:00:00Z' AND time <=
'2020-04-11T00:00:00Z' GROUP BY time(5m)
```



Execution Timing Diagram of Previous Implementation

https://realpython.com/python-concurrency/

EFFORTS - CONCURRENT PROCESSING

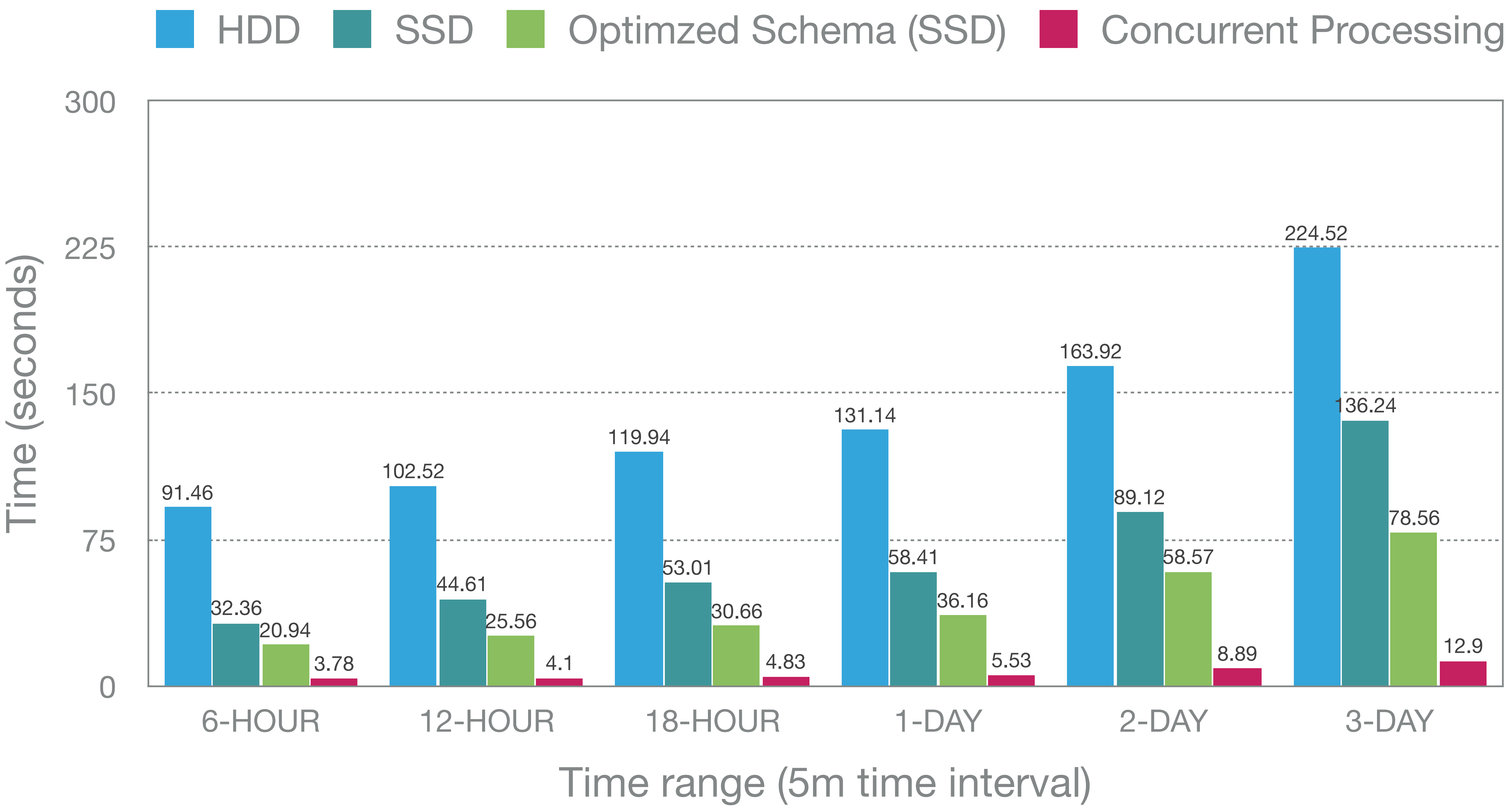


Request
SELECT max(Reading) FROM UGE WHERE NodeId='10.101.1.1' AND Label='CPUUsage' AND time >= '2020-04-10T00:00:00Z' AND time <= '2020-04-11T00:00:00Z' GROUP BY time(5m)
InfluxDB Waiting
CPU Processing
Request 1
Request 2
Request 3
...
Request 467
Thread 1
Thread 2
Thread 3
...
1
2
3
Thread 467
Time

Execution Timing Diagram of Current Implementation

https://realpython.com/python-concurrency/

IMPROVEMENT - CONCURRENT PROCESSING



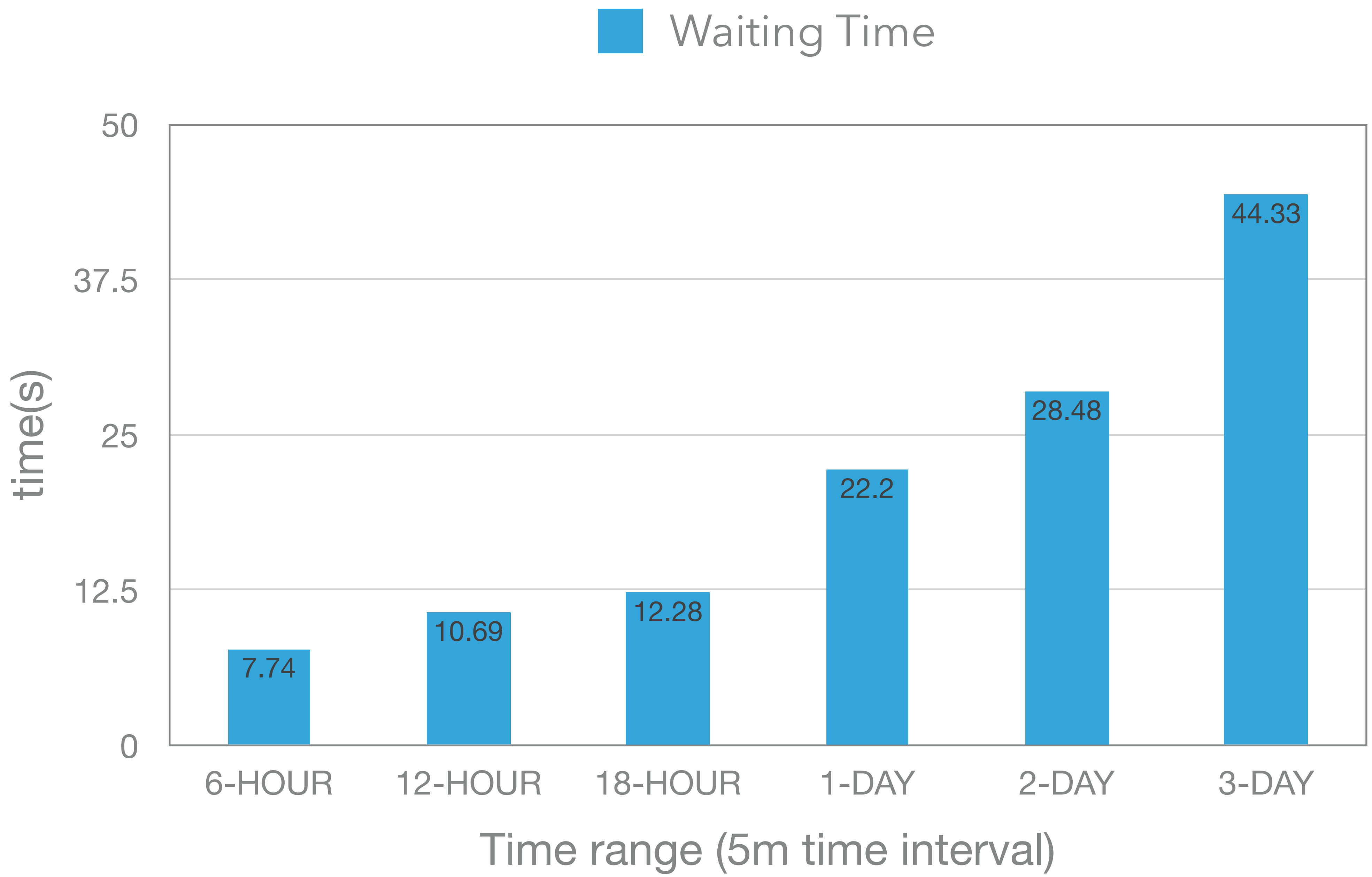
Query data in Concurrent from New Schema performs 5.5x ~ 6.5x faster than in sequence

Query data in Concurrent from New Schema performs 17x ~ 25x faster than previous schema on HDD

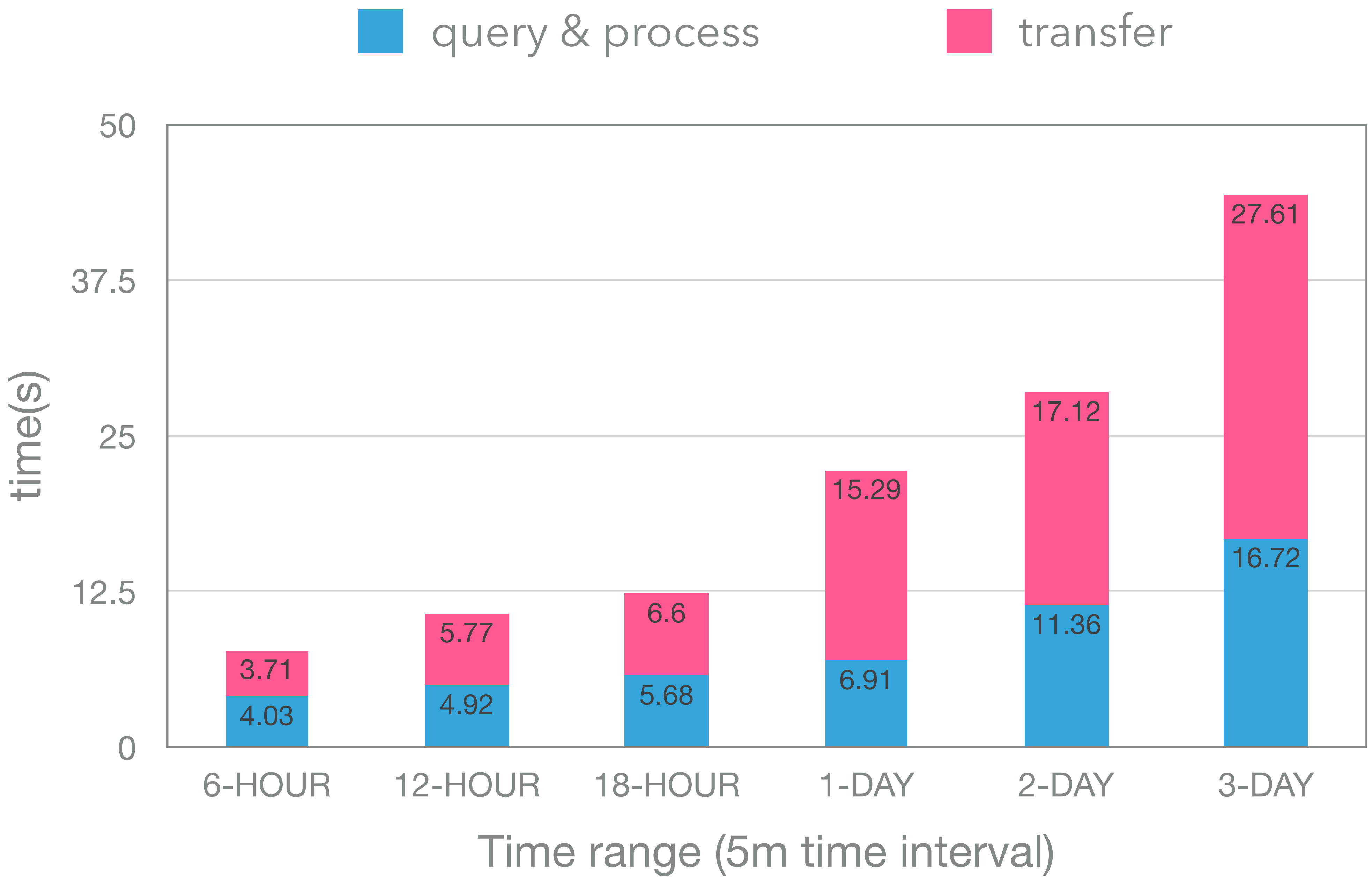
ONE MORE THING

One More Thing…

WAITING TIME



WAITING TIME DECOMPOSITION



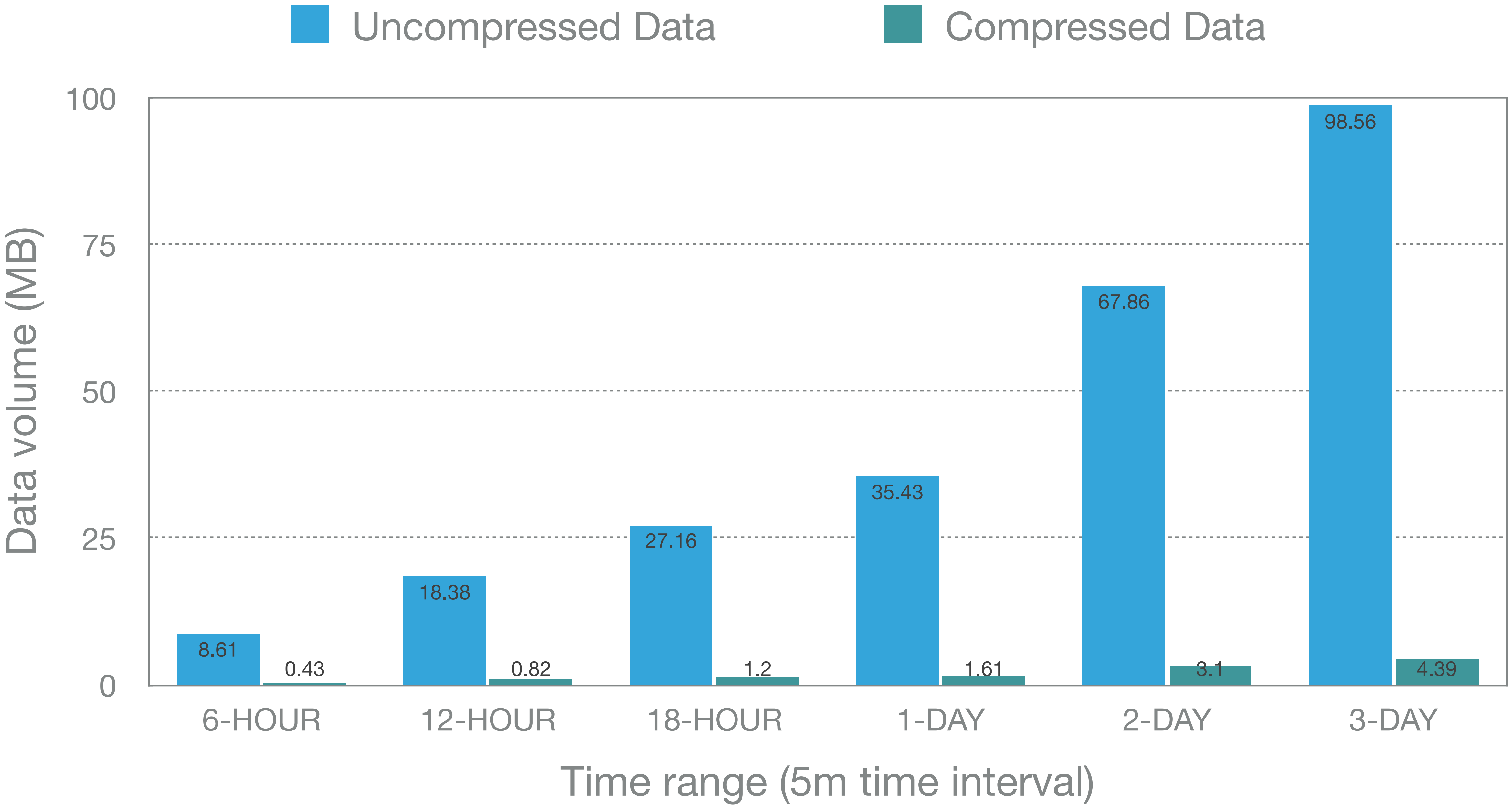
Transfer time is as much as 1.65x of query & process time

APPROACH-4

Transfer **Compressed** Data

EFFORTS - COMPRESSION

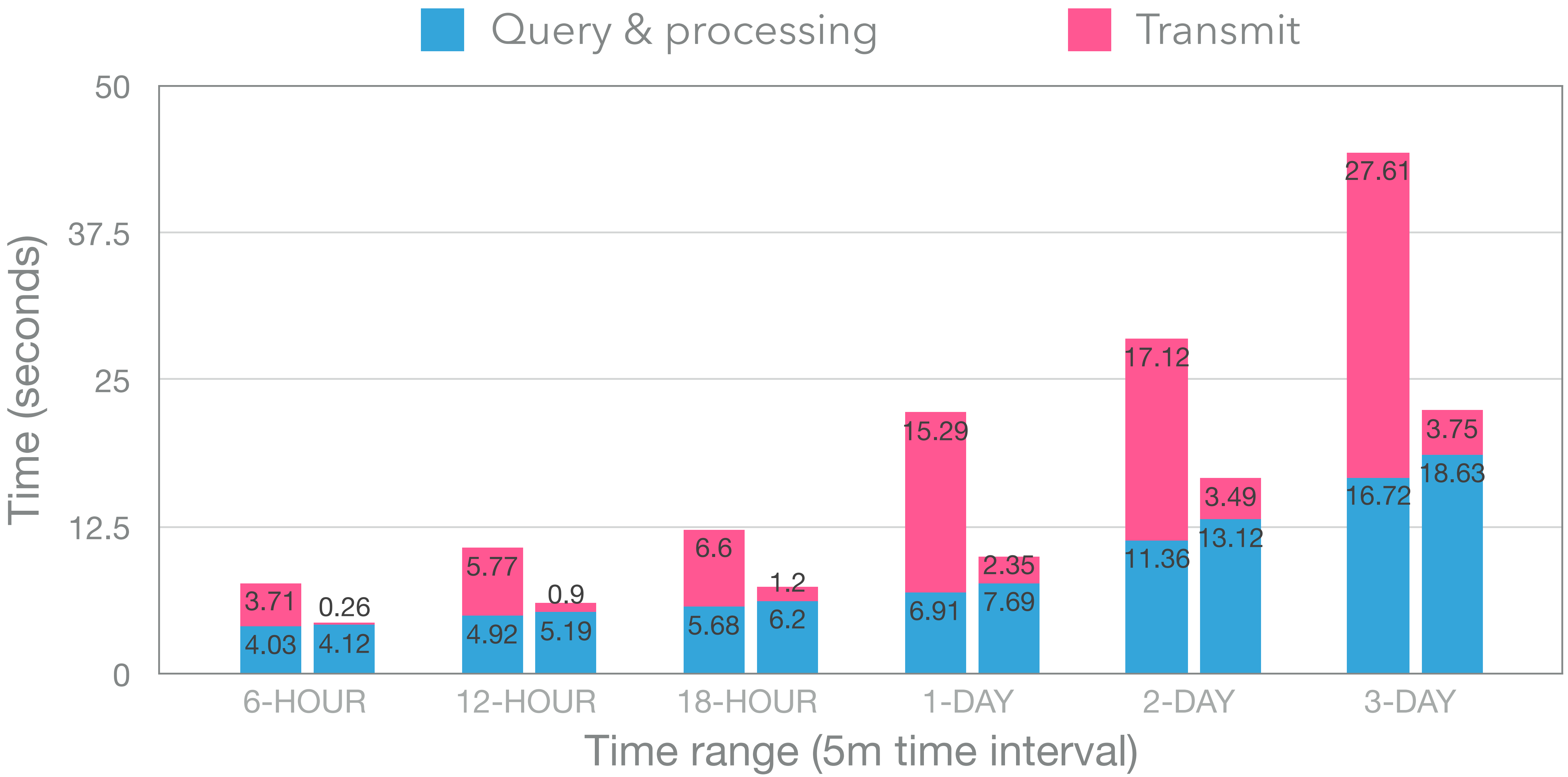
Use **zlib library** for data compression



Compressed data volume is only 4.45%~5.0% of uncompressed data

https://medium.com/@busybus/zipjson-3ed15f8ea85d

IMPROVEMENT - COMPRESSION



Using compressed data is 1.8x~2.16x faster than using uncompressed data

De-compress 6 hours of data only takes about 0.144 seconds.

SUMMARY

1. Switch to **SSD**
2. Redesign **Schema**
3. **Concurrent** Processing
4. Transfer **Compressed** Data

25x Speed Up

2x Speed Up

DEMO