Meerkat: Anomaly Detection as a Service

Julien Herzen

joint work with

Khue Vu & Big Data Network Intelligence Group

Big Data @ Swisscom

- About **75 billions events/day** (~860K event/s) ingested in Apache Kafka

- 800 cores, 3 TB memory, 1 PB HDFS storage

- **Technologies:** Apache Hadoop, HDFS, Kafka, Spark, Spark Streaming, Cassandra, Druid, ELK, …

- **Applications:** Network & business intelligence, mobility insights, customer care, …

Meerkat

Goal: Have a system always **on the lookout for things out of the ordinary**, in order to increase engineers' QoL (Quality of Life).

Overview

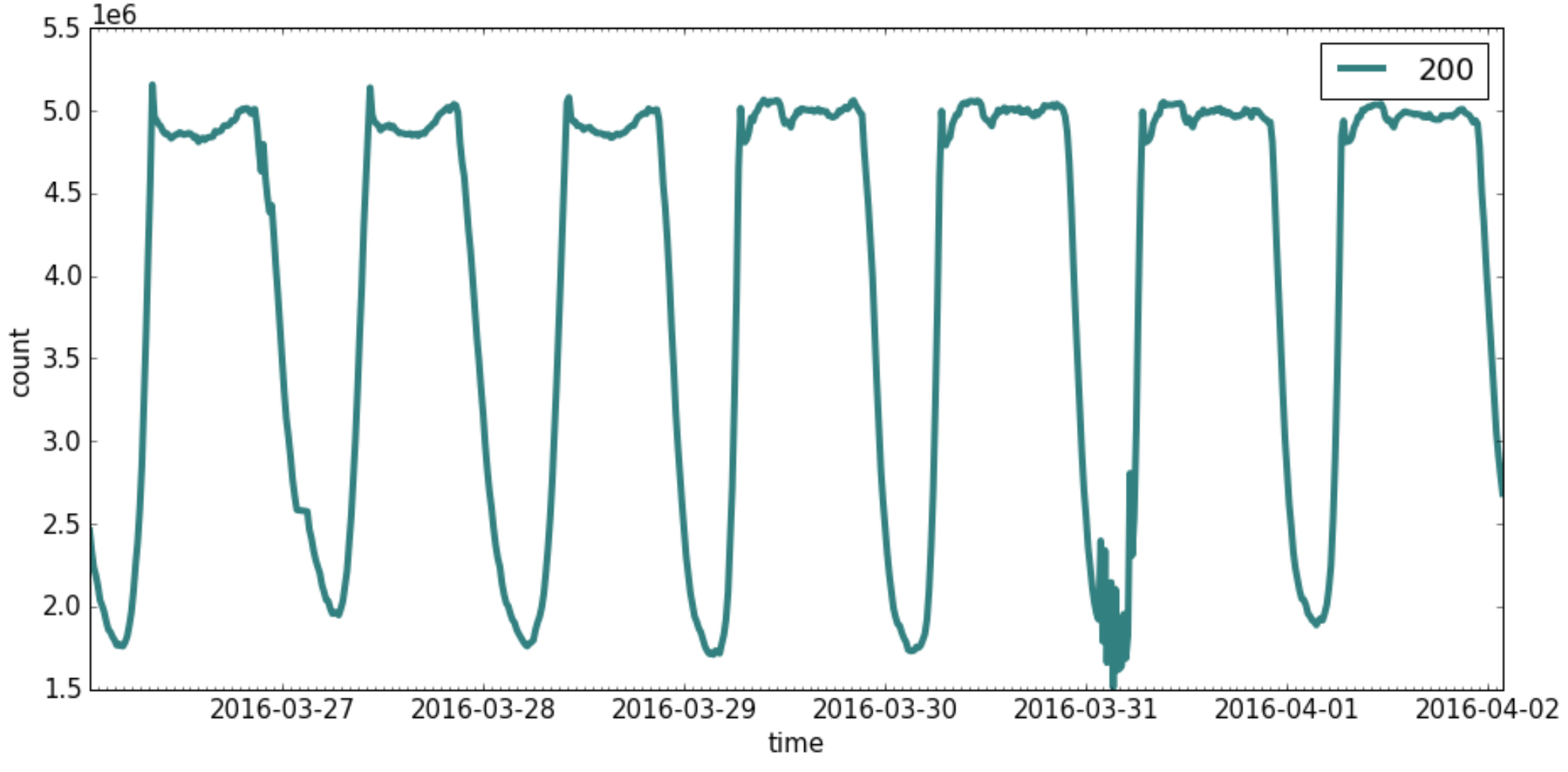
Telco data can often be seen as **time series**:

- Nr. phone calls
- Nr. failures
- Occurrences of different network protocol codes
- …

Approach: learn what time series should look like when things are normal. Quickly send an alert when things don’t look normal

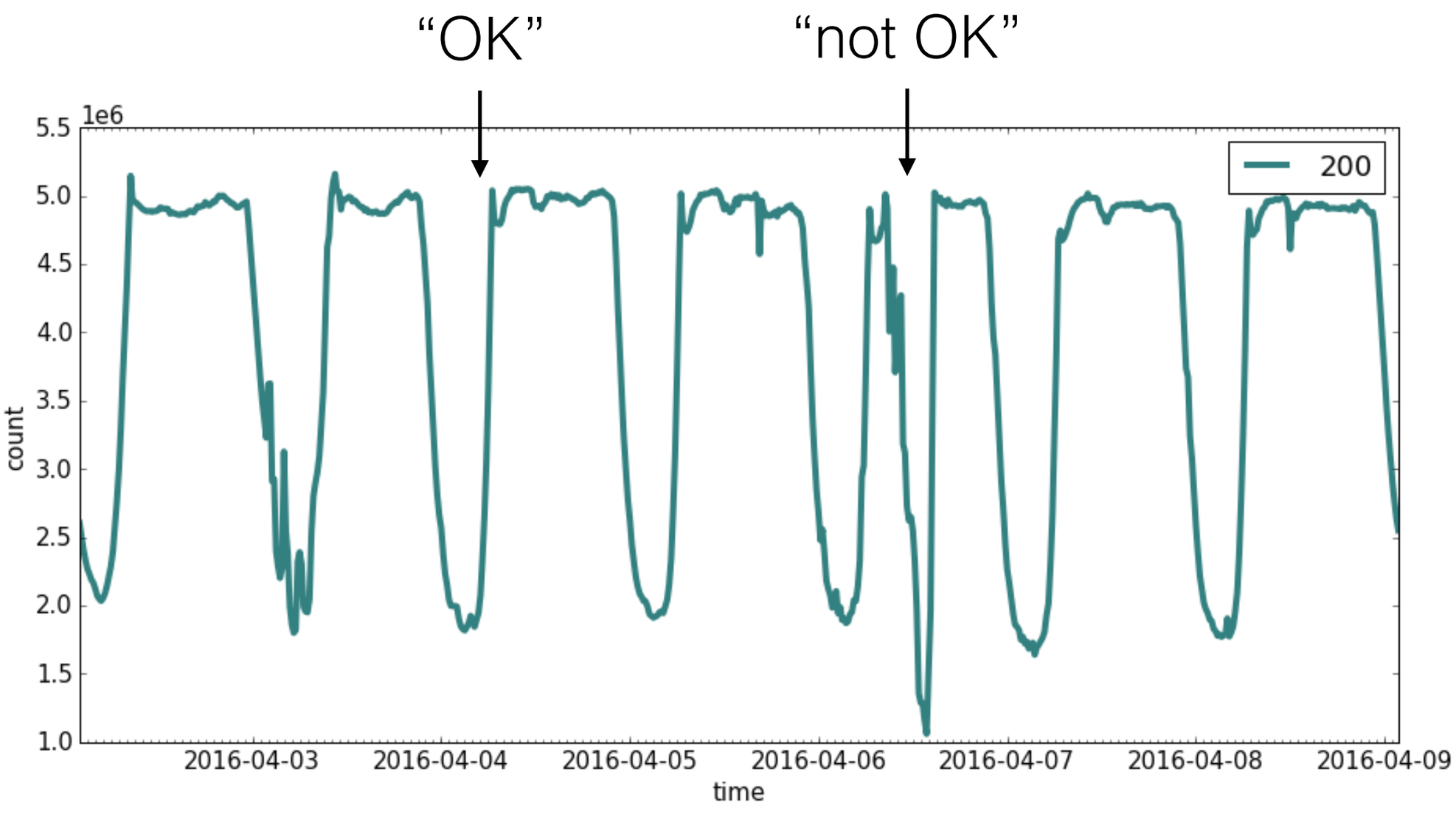
Example

- VoIP calls (VoLTE & WiFi): few millions calls/day
- SIP (Session Initialisation Protocol) cause code 200 (**OK**)

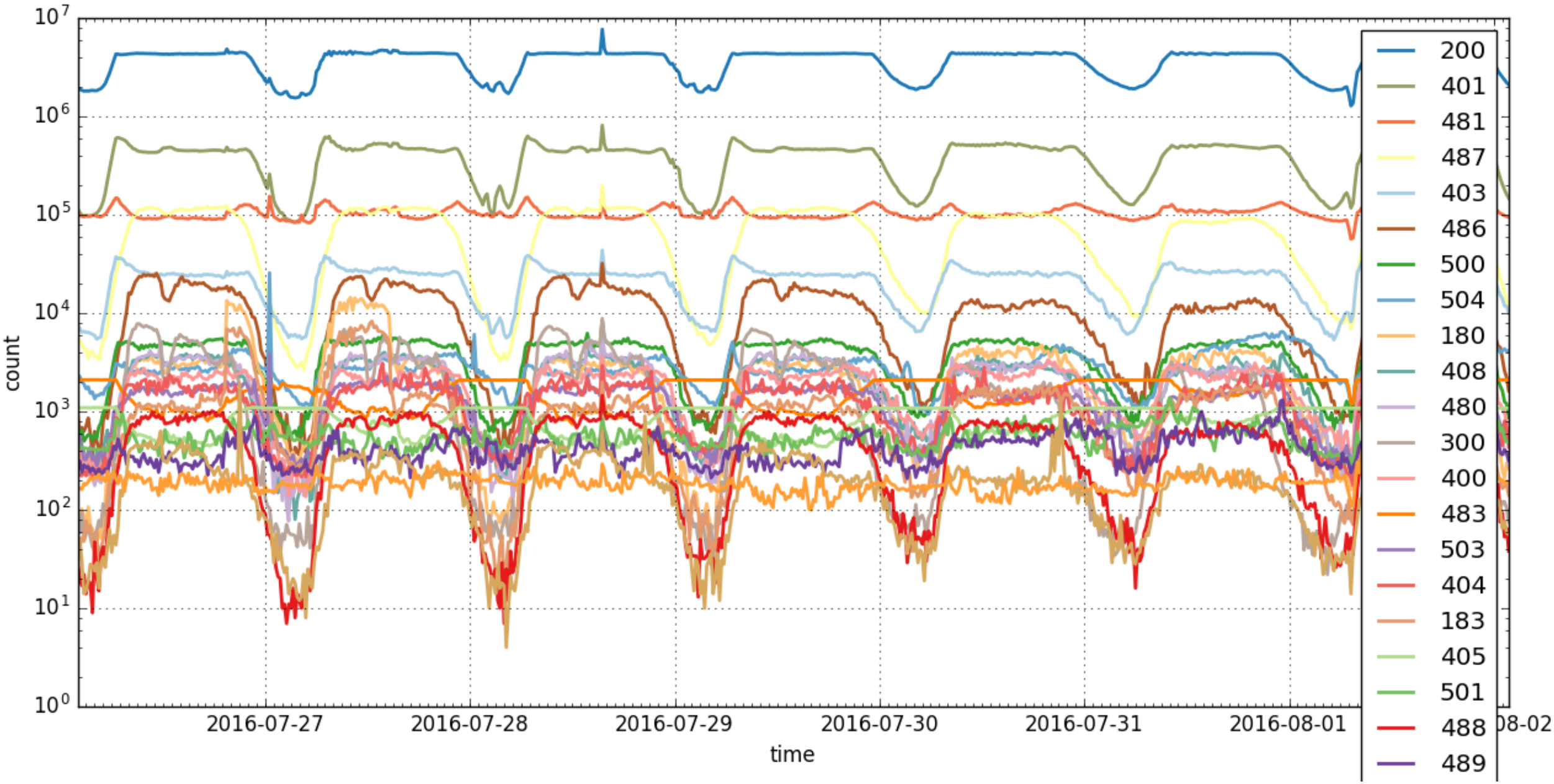


Example

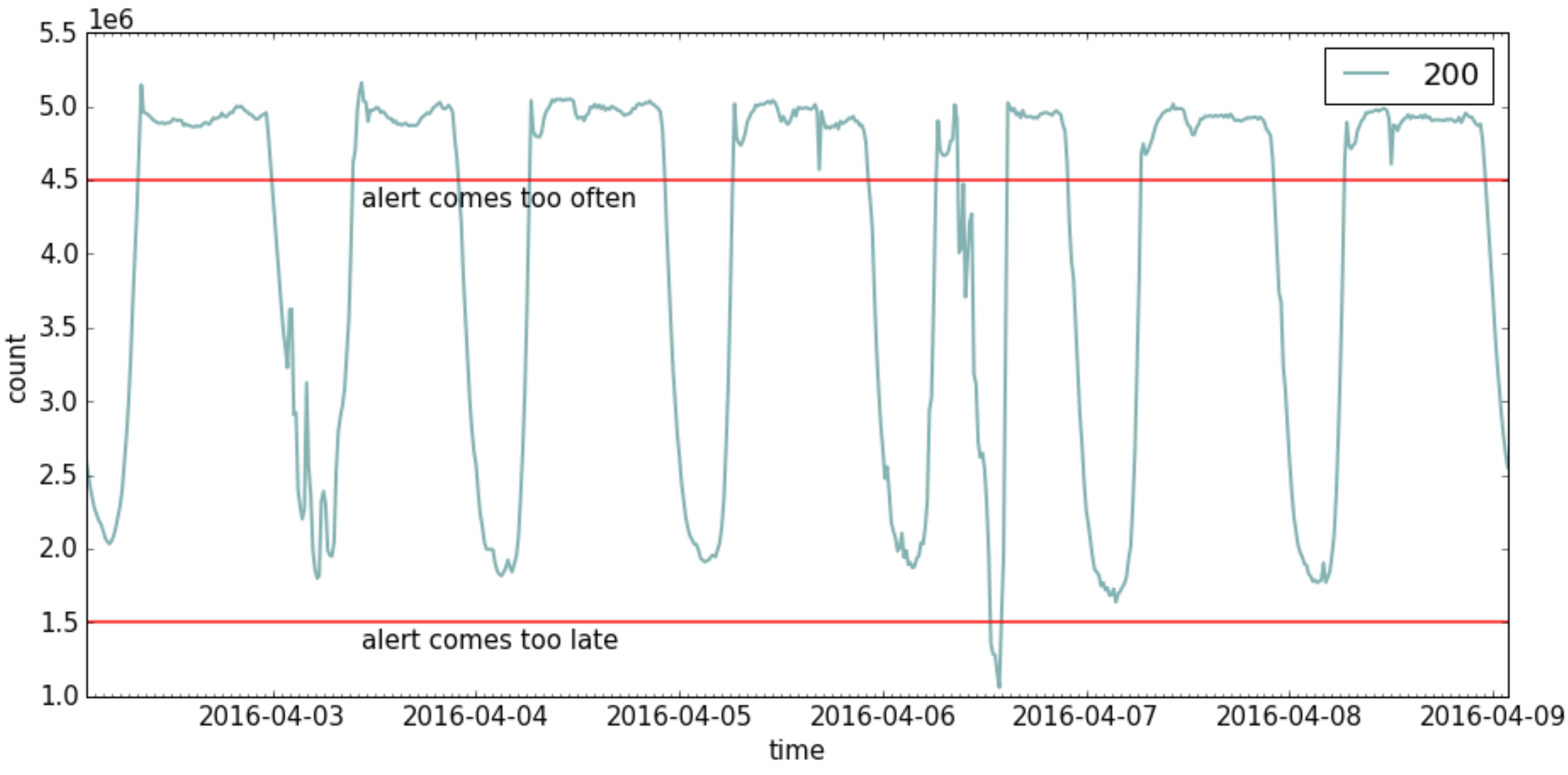
- VoIP calls (VoLTE & WiFi): few millions calls/day
- SIP (Session Initialisation Protocol) cause code 200 (**OK**)



In General

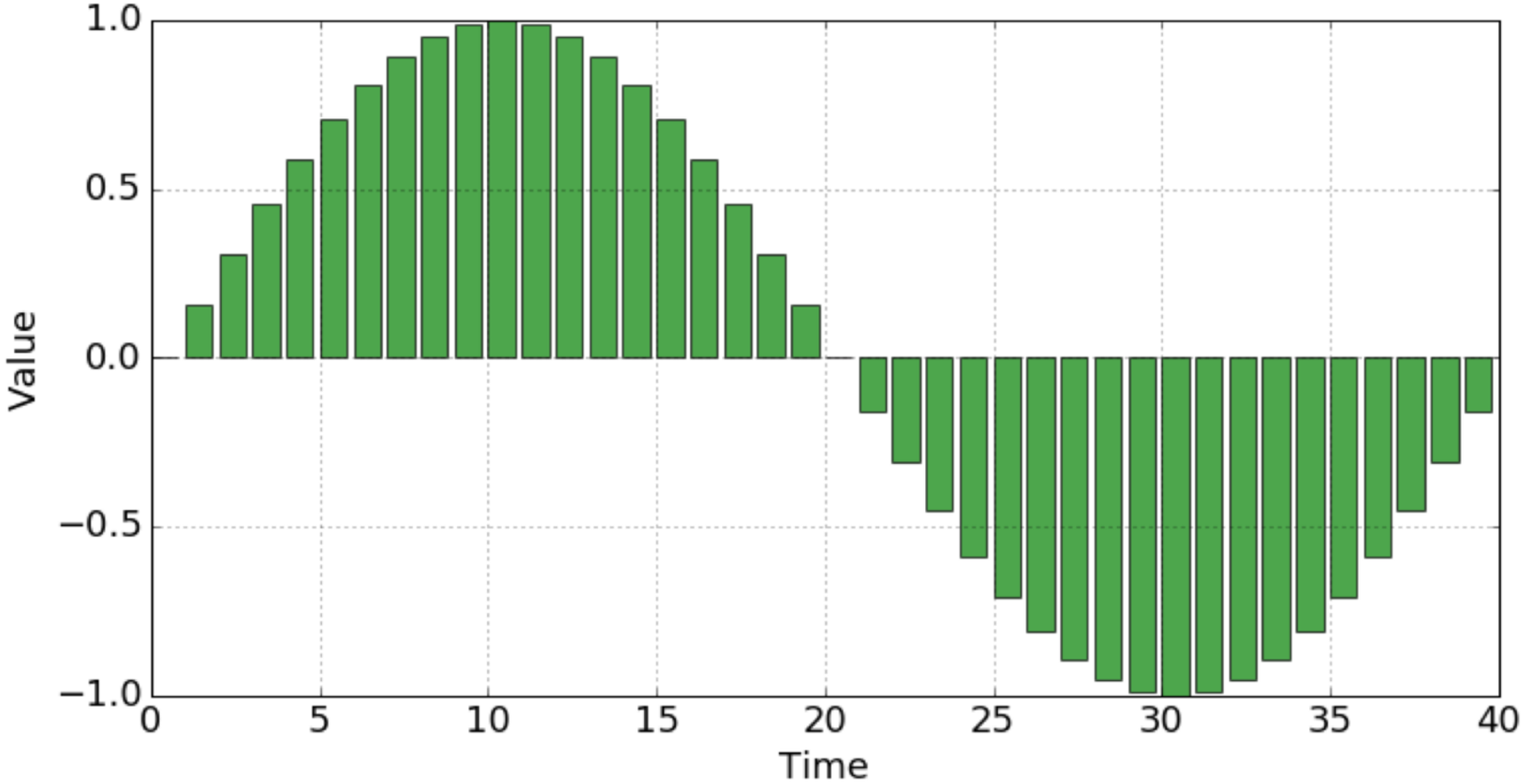

count
time
2016-07-27
2016-07-28
2016-07-29
2016-07-30
2016-07-31
2016-08-01
200
401
481
487
403
486
500
504
180
408
480
300
400
483
503
404
183
405
501
488
489

Thresholds are not always useful



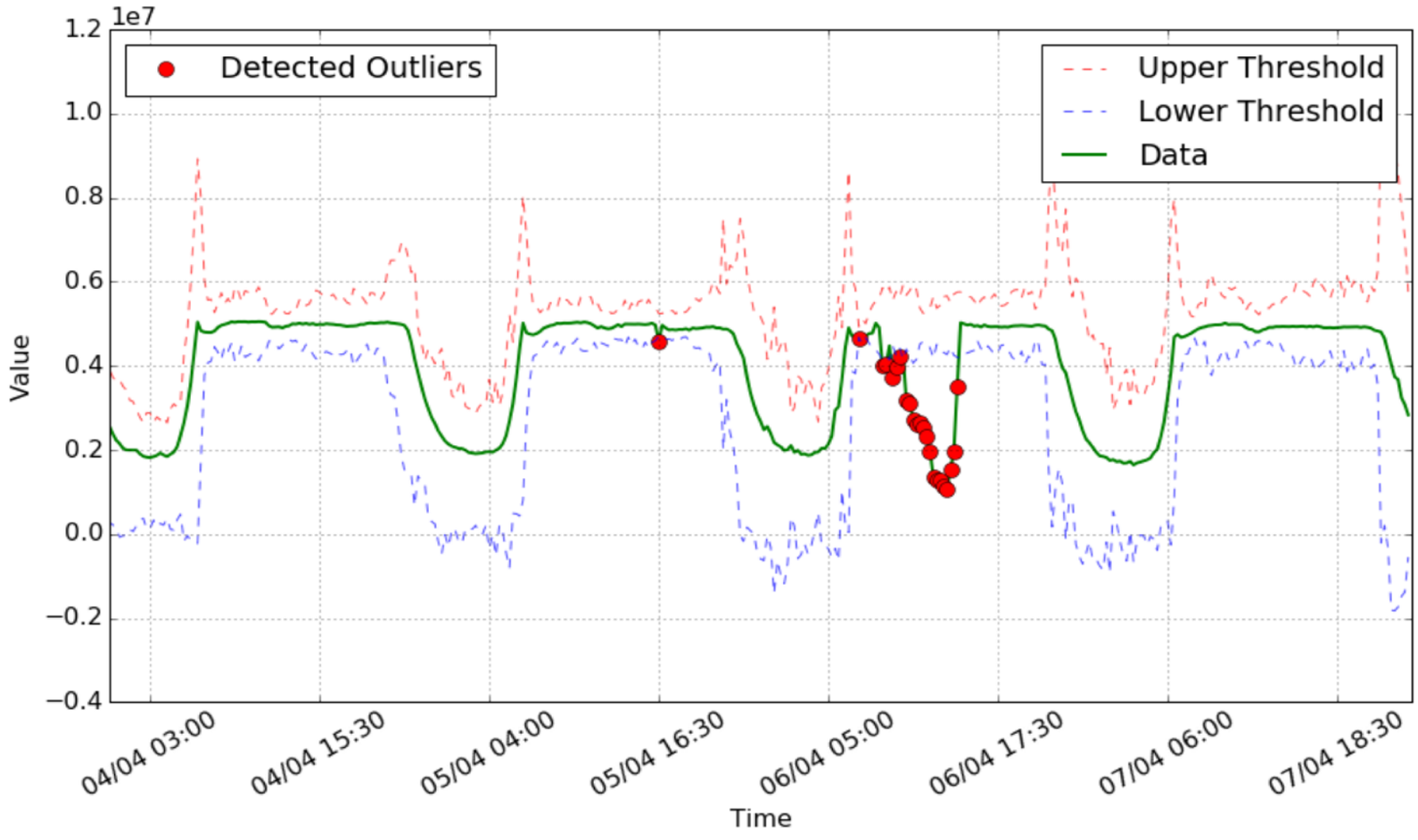
First Approach: Statistical Modeling

- Naive “**bucketing**” strategy
- Divide time series into e.g., 15-mins buckets
- Compute simple statistics, e.g., mean and stdev — or percentiles for robustness
- `isOutlier(x) = |x - mean| > k * stdev`



Pros:

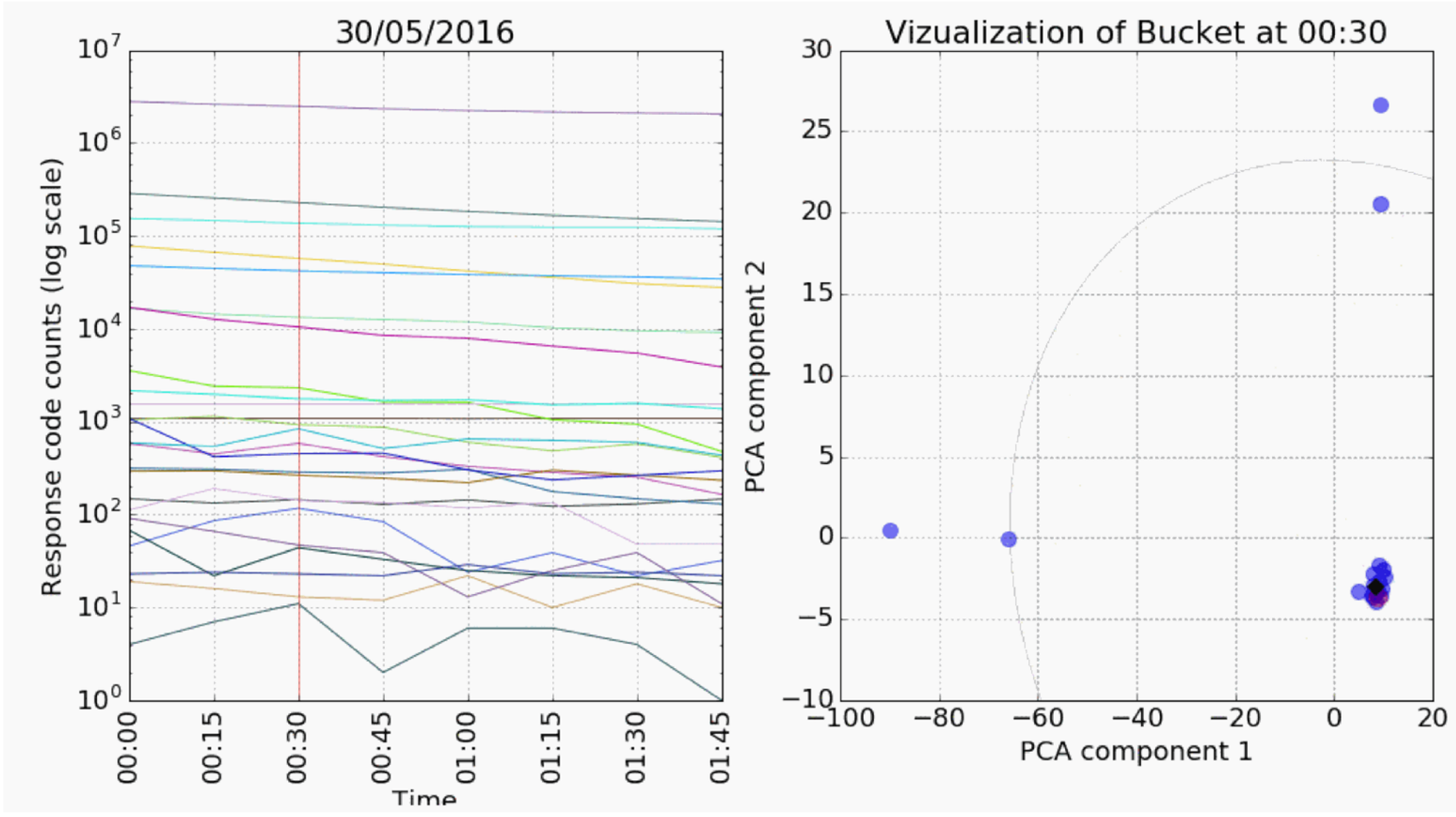
- Simple & computationally efficient
- Captures periodicity
- Can be adapted to learn trend (using e.g., linear regression on the mean) and multi-variate data (using e.g., PCA)



Con: Takes a long time to adapt to permanent changes

Multivariate extension

- Observations: $\vec{v}$
- Concatenate in a matrix $V_i := [v_i, v_{i+T}, v_{i+2T}, \ldots]$
- PCA on V_i for all i
- outlier if point falls outside of confidence region

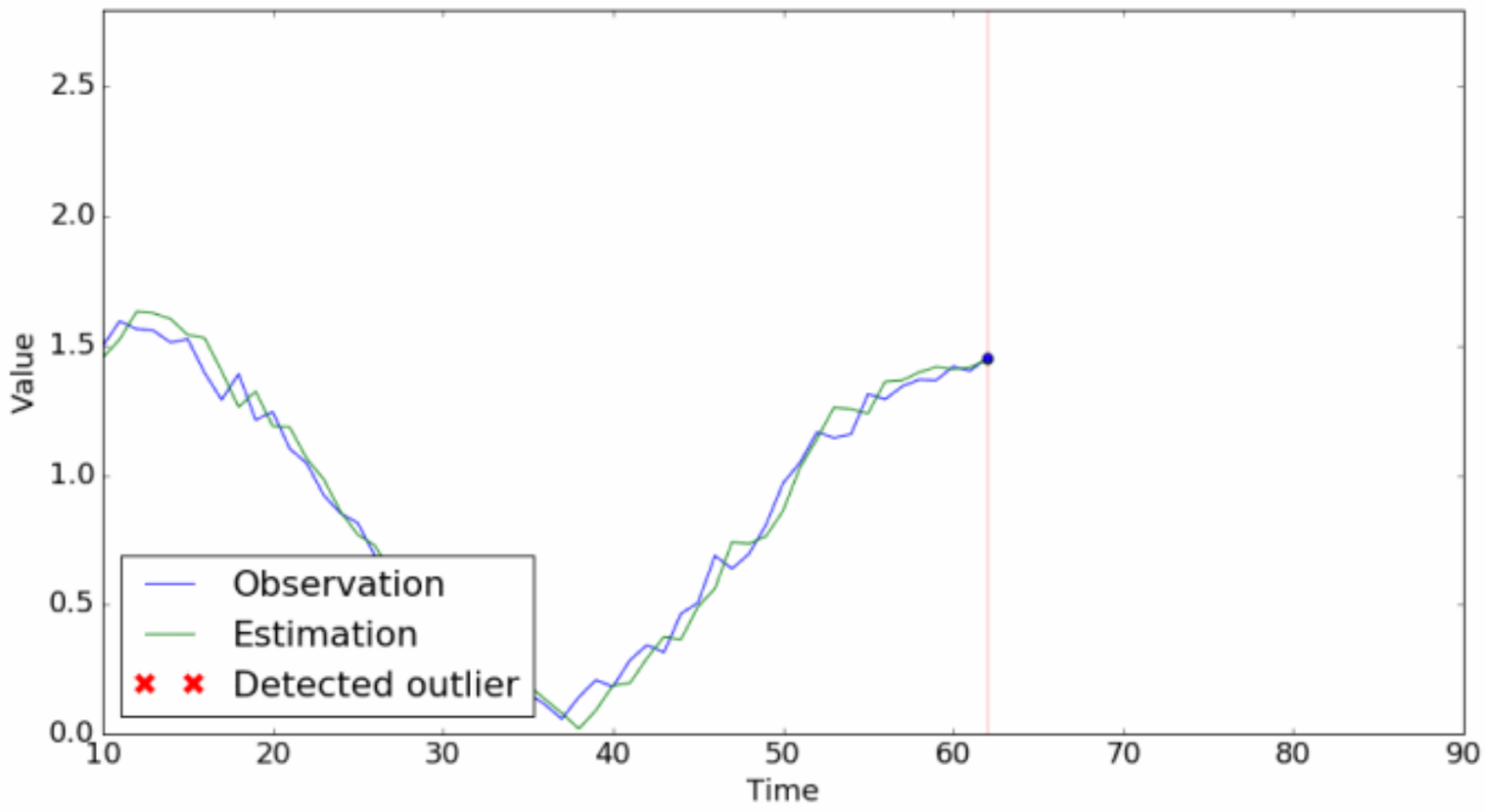


Second Approach: Prediction

- Consider generative models that can produce **next-point predictions**
- E.g.: **AR(I)MA, Kalman filtering**
- **Idea:** predict next point before observing it. If the observation is very different from prediction, decide that the point is an outlier

```
isOutlier(x) = |x - prediction| > k * stdev(recent prediction errors)
```

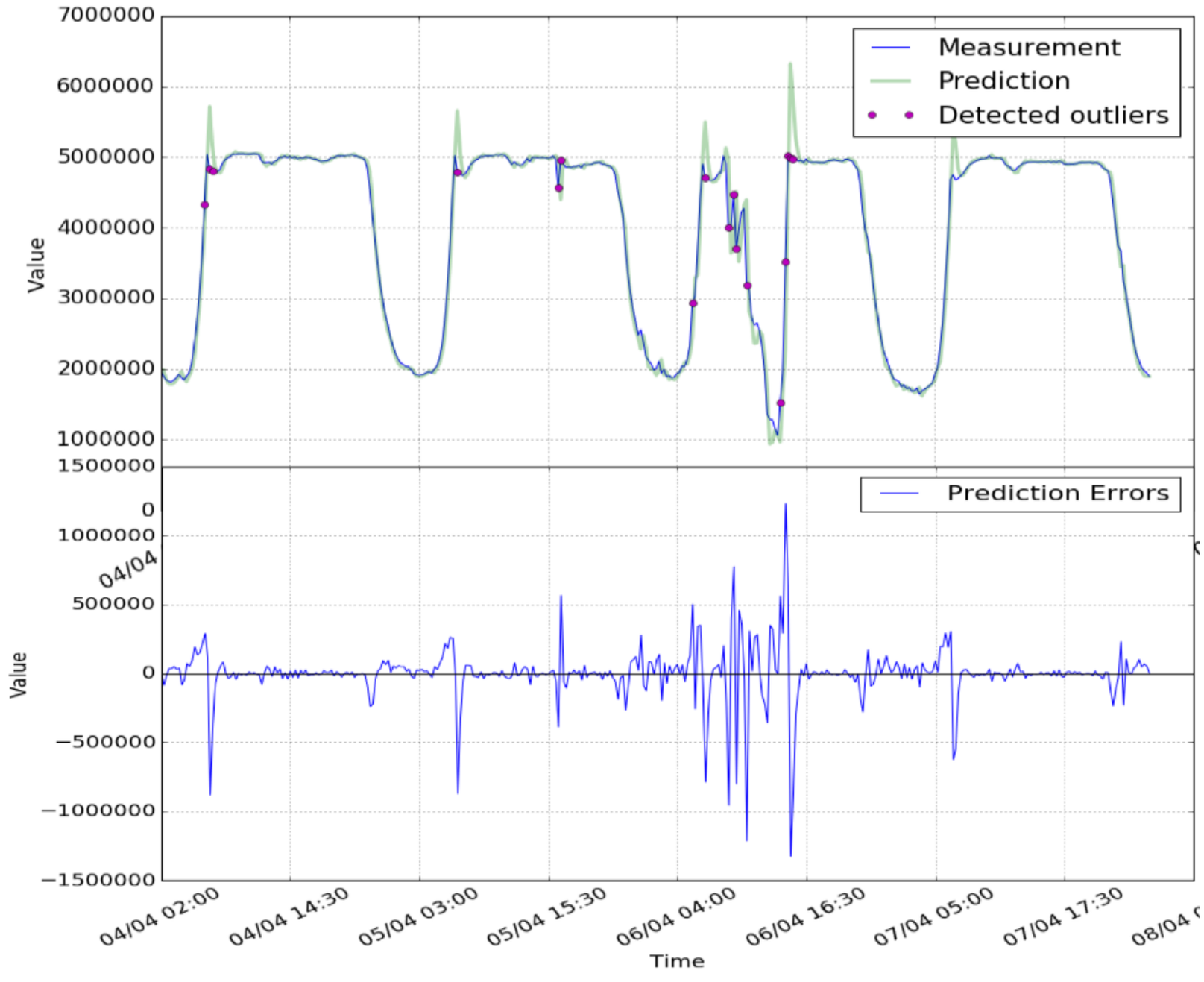
Example animation of one-step-ahead prediction with a Kalman filter:



Pros:

- Can be built to immediately react to permanent changes
- Good at detecting sudden spikes or drops

Con: Not good at capturing periodicity (in computationally-efficient way)



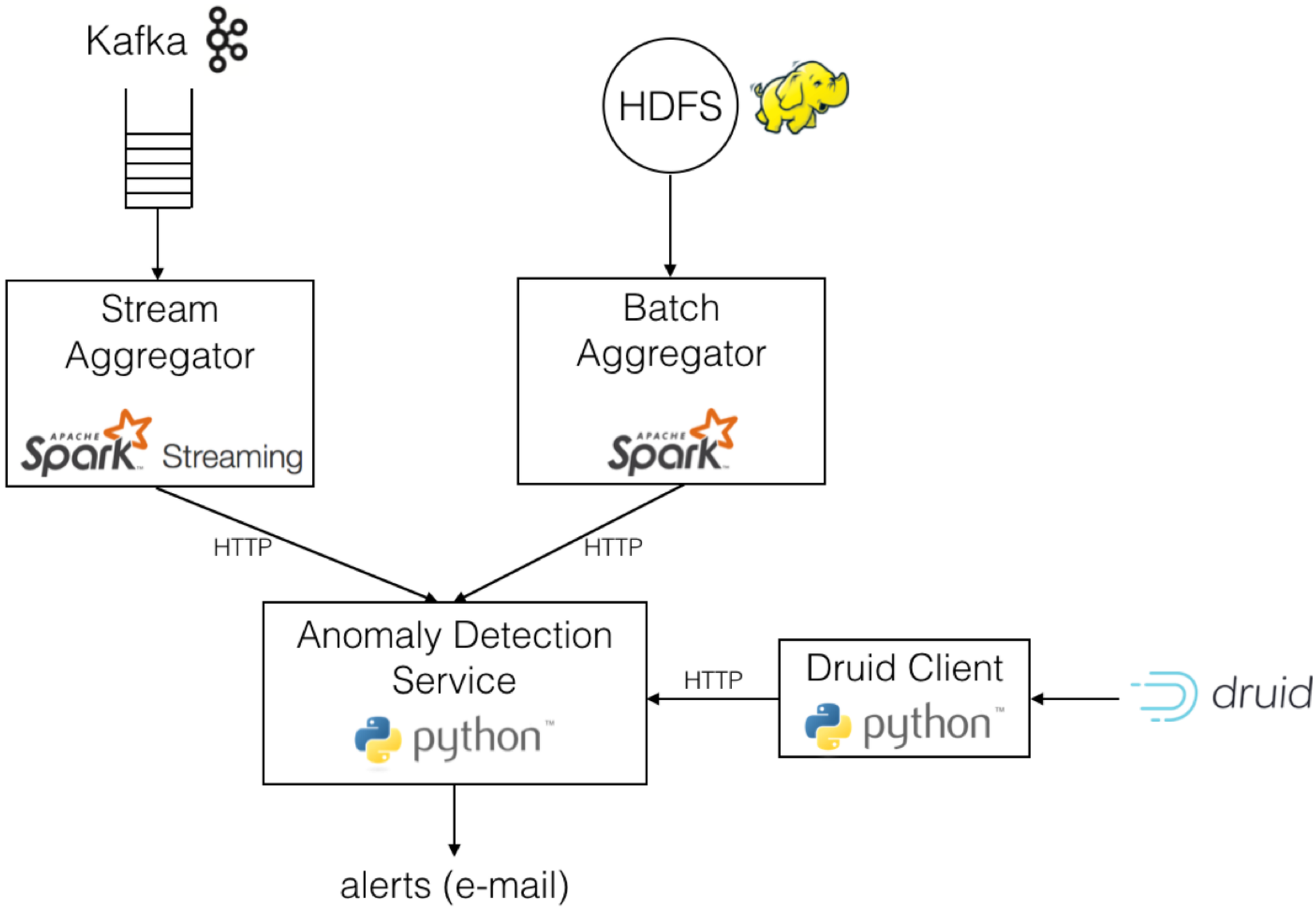
Combining Models

Our combined models currently use a trivial voting strategy:

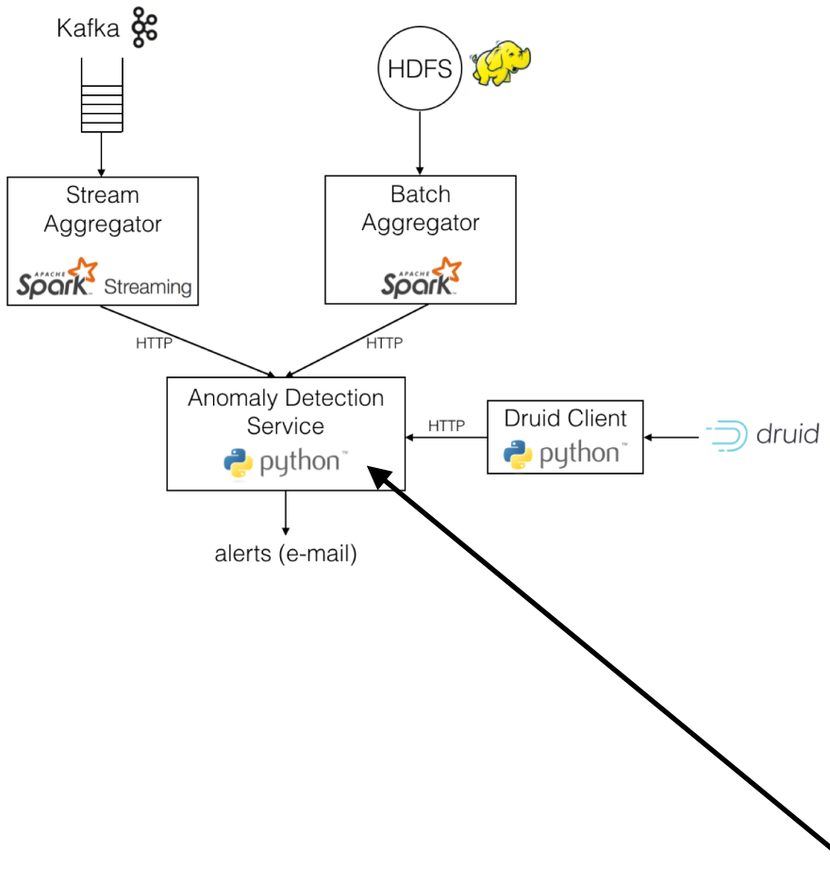
```
isOutlier(x) = models.forall(_.isOutlier(x))
```

- Conservative combining rule => individual models can be tuned more aggressively, yet:
  - Statistical model **avoid periodic false positives** of prediction-based model
  - Prediction-based model **avoids repeated false positives after permanent change** of bucketing-based model

Architecture


Kafka
HDFS
Stream
Aggregator
Streaming
Batch
Aggregator
HTTP
HTTP
Anomaly Detection
Service
python
HTTP
Druid Client
python
druid
alerts (e-mail)

Univariate & Multivariate Models




```python
class Model(object):
    """
    Base class for univariate and multivariate models
    """
    __metaclass__ = ABCMeta

    @abstractmethod
    def build(self):
        pass

    @abstractmethod
    def update(self, timestamp, observation):
        pass

    @abstractmethod
    def detect_outlier(self, timestamp, observation):
        pass

    @abstractmethod
    def update_and_detect(self, timestamp, observation):
        pass

class UnivariateModel(Model):
    """
    Base class for all univariate models
    """
    def __init__(self, train_start_time, granularity, name=None, train_weeks=4,
                 rebuild_period=None, first_build_after=None, is_eligible=None):
        ...

class MultivariateFromUnivariateModel(Model):
    """
    This class represents a multivariate model that bundles several univariate models together.
    """
    def __init__(self, granularity, model_name, features, train_weeks, is_eligible):
        ...
```

Anomaly Detection as a Service

Step 1: specify monitoring parameters:

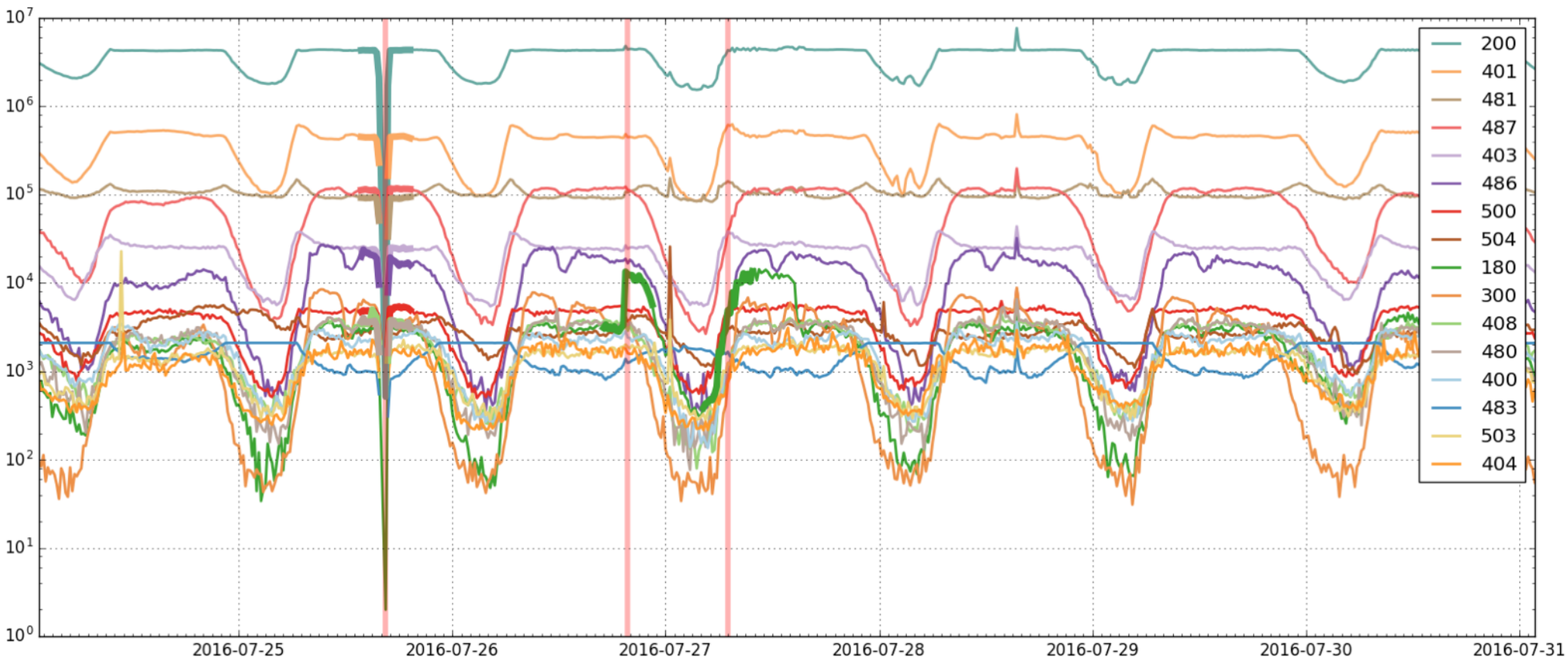
```
{
    "name": "causecode_bucket_arma",
    "mail_tos": [
        "khue.vu@swisscom.com",
        "julien.herzen@swisscom.com"
    ],
    "models": [
        {
            "type": "bucket",
            "parameters": {
                ...
            }
        },
        {
            "type": "arma",
            "parameters": {
                ...
            }
        }
    ]
}
```

Step 2: send data (in any order):

```
timestamp, 200, 401, 481, ...
1472040000, 4978367, 477813, 129010, ...
1472040900, 5113918, 517215, 127873, ...
...
```

Spamming Colleagues: How to do it Right

- **Principle:** Decouple outlier detection from alert generation
- Notion of **ongoing anomaly:** send an initial alert, then disable alerting until the anomaly is “over”
- Generate (simple) human-readable message




```
Anomalies detected for timeseries test-series at:
-- Time: 2016-07-26 19:45
---- Reason: "180" is significantly higher than usual.
```

Conclusions & Future Work

- Meerkat already generated a handful of alerts which generated positive feedback from engineering team
- Next steps:
  - UI for easy model selection
  - Include human feedback in the loop to improve voting semantics across models
  - Open source?