Analytics on streaming data from Environmental Sensors

Rajrup Ghosh (M Tech CDS)

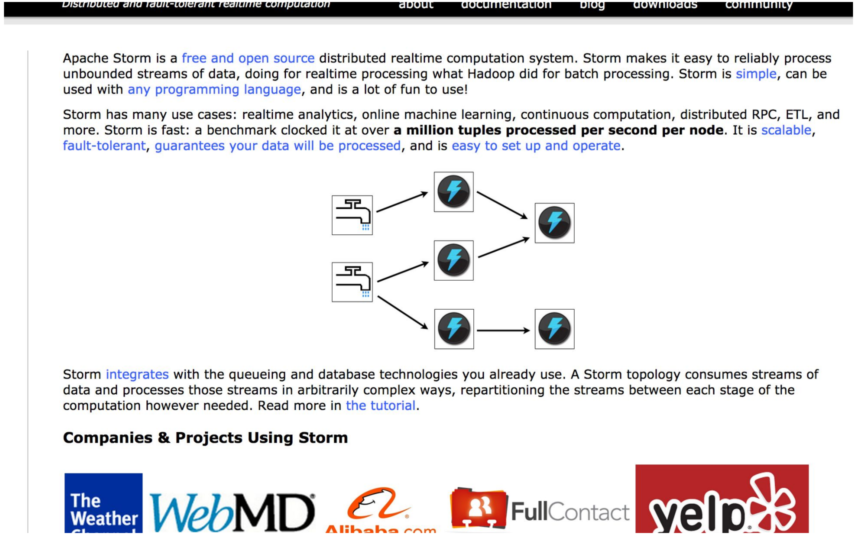
Shib Shankar Das (M Tech ECE)

Streams Everywhere

- Internet and Social Networks generate streams of posts, hashtags, videos, etc
  - Twitter, Facebook, Internet packets
- Business
  - Stock market prediction
  - Monetary Transactions
- Cybersecurity
  - Telecom call logs,
  - financial transactions, Malware
- Internet of Things
  - Smart Transport, Power and smart grids
  - Weather forecast and Pollutant monitoring
  - Smart phone, Health appliances, TV

Apache Storm

- Originally developed by Nathan Marz at Backtype/Twitter
- Distributed, fault-tolerant stream-processing platform



http://storm.apache.org/

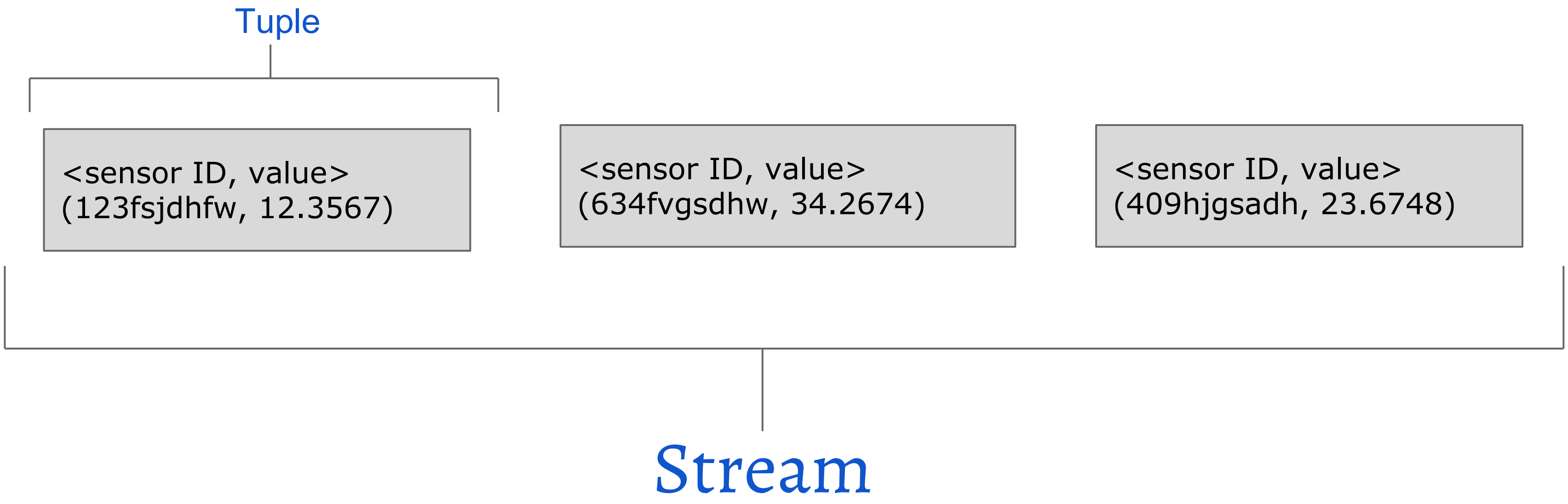
Storm Cocepts

- Tuples and Streams
- Spouts, Bolts, Topologies
- Tasks and Workers
- Stream Grouping

http://storm.apache.org/releases/current/Tutorial.html

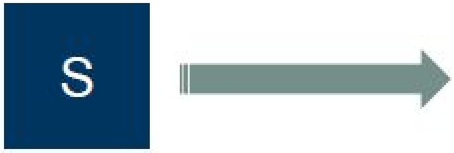
Tuples and Streams

- Tuple: ordered list of elements
- Stream: unbounded sequence of tuples



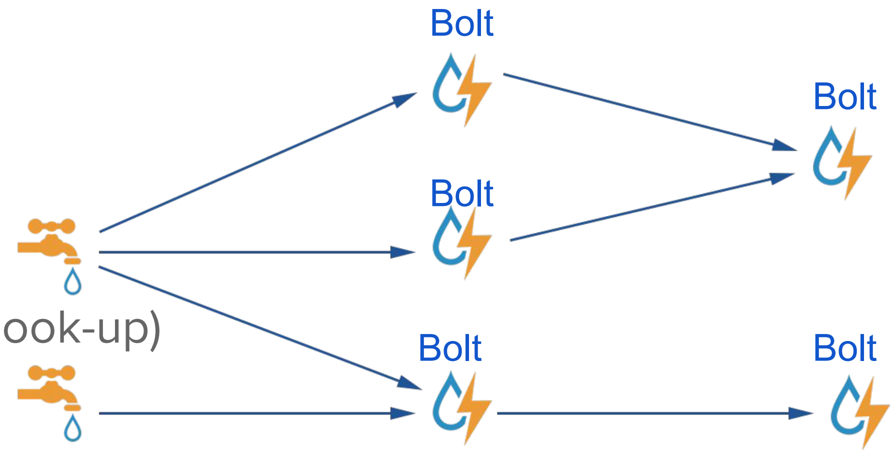
Spouts

- The sources of streams
- Can talk with
  - Queues (Kafka, Kestrel, etc.)
  - Web logs
  - API calls
  - Filesystem (MapReduce-FS / HDFS)



Bolts

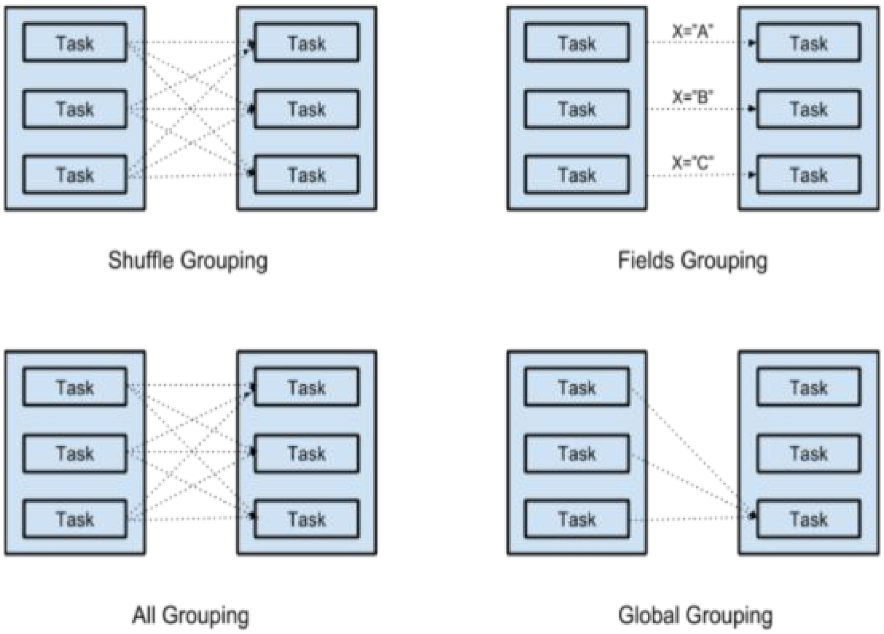
- Process tuples and create new streams
- Implement business logic via ...
  - Transform
  - Filter
  - Aggregate
  - Join
  - Access datastores & DBs
  - Access APIs (e.g., geo location look-up)



TOPOLOGY - Directed graph of spouts and bolts

Stream Grouping

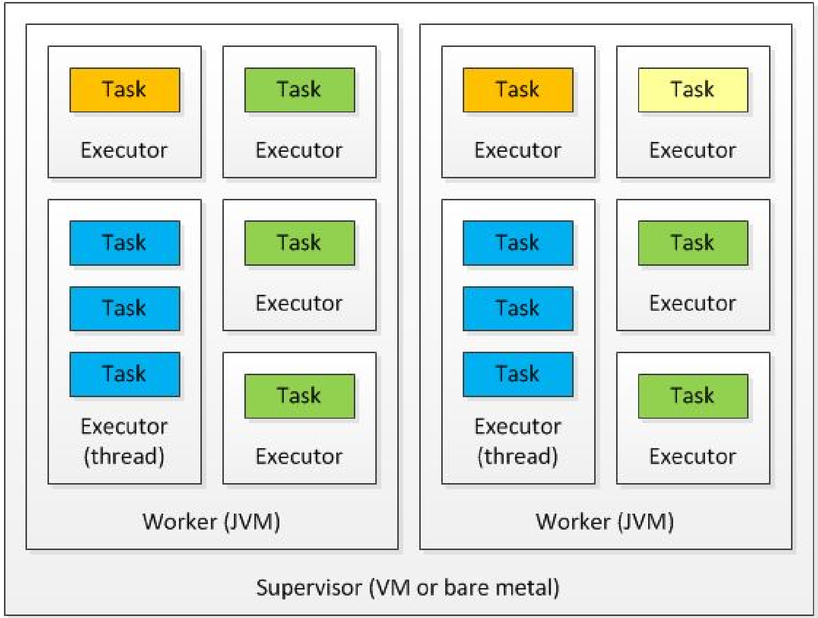
- Shuffle grouping: tuples are randomly distributed across all of the tasks running the bolt
- Fields grouping: groups tuples by specific name field and routes to the same task



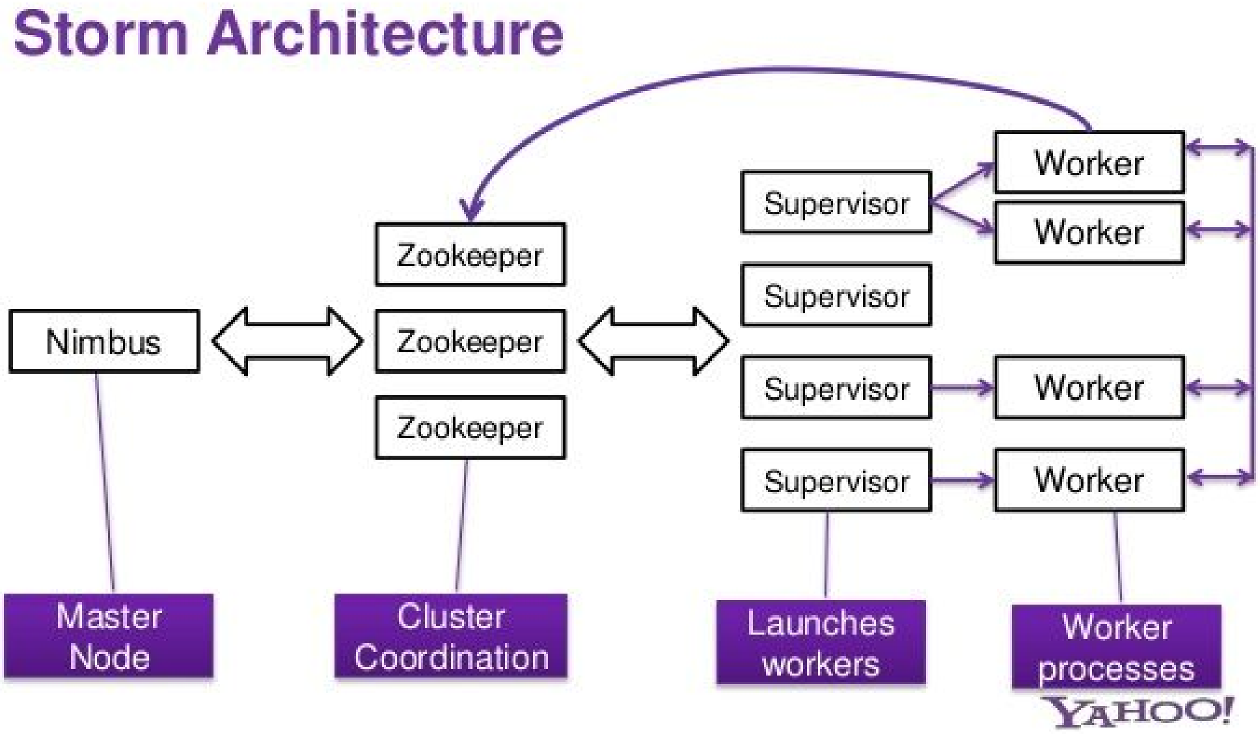
Storm Groupings

Tasks and Workers

- **Task:** each spout/bolt executes as many threads of execution across the cluster
- **Worker:** a physical JVM that executes a subset of all the tasks for the topology



Storm Architecture



Storm Architecture

DataStreamGeneratorSpout

FilterBolt : Kalman Filter

BuildParameterBolt (mean,standard deviation,second moment,third moment)

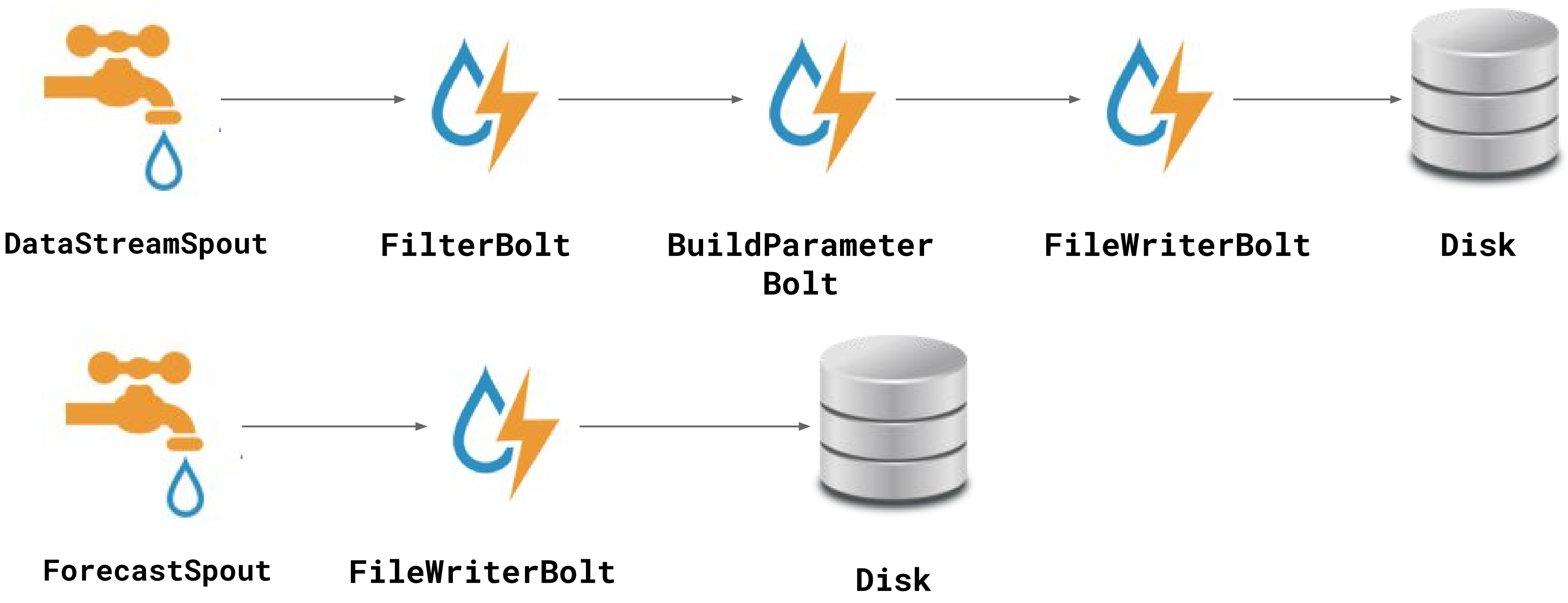
- Used Alon-Matias-Szegedy Algorithm for moment estimation

FileWriterBolt

DataEstimatorSpout

DataWriterBolt

Topology


DataStreamSpout
FilterBolt
BuildParameter
Bolt
FileWriterBolt
Disk
ForecastSpout
FileWriterBolt
Disk

Alon-Matias-Szegedy Algorithm

Pick random element

Store the element with a count 1

If existing element found in further data increase count

Estimated Second moment = n*(2*X.value-1)

Estimated Third moment= n*(3*v^2-3*V+1)

Forecasting

- Forecasting is a process of estimation of an unknown event/parameter such as demand for a product, amount of rainfall, etc.
- Forecasting is commonly used to refer timeseries data.
- Time Series is a sequence of data points measured at successive time intervals.
- Time series analysis helps to identify and explain:
  - Any systematic variation in the series of data which is due to seasonality.
  - Cyclical pattern that repeat.
  - Trends in the data.
  - Growth rates in the trends.

Additive and Multiplicative Models

1. Additive Forecasting Model

$$Y_t = \underbrace{T_t}_{\text{Trend}} + \underbrace{S_t}_{\text{Seasonality}} + \underbrace{C_t}_{\text{Cylical}} + \underbrace{R_t}_{\text{Random}}$$

2. Multiplicative Forecasting Model

$$Y_t = \underbrace{T_t}_{\text{Trend}} \times \underbrace{S_t}_{\text{Seasonality}} \times \underbrace{C_t}_{\text{Cylical}} \times \underbrace{R_t}_{\text{Random}}$$

Time Series Techniques

- Moving Average.
  - Simple moving average
  - Weighted moving average

$$F_{t+1} = \frac{1}{n} \sum_{i=t-n+1}^{t} Y_i$$

F_{t+1} = Forecast for period $t+1$

Y_i = Data corresponding to time period i

- Exponential Smoothing.

$$F_t = \alpha Y_{t-1} + \alpha(1-\alpha) Y_{t-2} + \alpha(1-\alpha)^2 Y_{t-3} + \ldots$$

- Auto-regression Models (AR Models)
- ARIMA (Auto-regressive Integrated Moving Average) Models

AR(p), MA(q) and ARMA(p, q)

- Auto-Regressive Process: AR(p) process models each future observation as a function "p" previous observations. If $\{Y_t\}$ is purely random with mean zero and constant standard deviation σ (White Noise). Then the autoregressiv AR(p) process:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t$$

- Moving Average Process: MA(q) models each future observation as a function of "q" previous errors. If $\{Y_t\}$ is a moving average process of order q (MA(q)) if for some constants $\beta_0, \beta_1, \ldots, \beta_q$

$$Y_t = \beta_0 + \beta_1 \varepsilon_t + \beta_2 \varepsilon_{t-2} + \ldots + \beta_q \varepsilon_{t-q} + \varepsilon_t$$

- ARMA(p,q) model is a combination of AR(p) and MA(q) process, given by -

$$\begin{aligned} Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \ldots + \alpha_p Y_{t-p} + \\ + \beta_0 + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \ldots + \beta_q \varepsilon_{t-q} + \varepsilon_t \end{aligned}$$

ARIMA (p, d, q)

ARIMA has the following three components:

- Auto-regressive component (p): Function of past values of the time series.
- Integration Component (d): Differencing the time series to make it a stationary process.
- Moving Average Component (q): Function of past error values.
- The q and p values are identified using auto-correlation function (ACF) and Partial auto-correlation function (PACF) respectively. The value d identifies the level of differencing.
- Box-Jenkins Methodology is used to identify the model parameters.

Measures of aggregate error

Mean absolute error **MAE**	$MAE = \frac{1}{n}\sum_{t=1}^{n}\lvert E_t \rvert$
Mean absolute percentage error **MAPE**	$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left\lvert \frac{E_t}{Y_t} \right\rvert$
Mean squared error **MSE**	$MSE = \frac{1}{n}\sum_{t=1}^{n} E_t^2$
Root mean squared error **RMSE**	$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} E_t^2}$

$E_t = F_t - Y_t$

What we have used?

- R forecast package has been used (developed by Hyndman & Khandakar (JSS, 2008). We have used JRI for R function calls from Java.
- We have used auto.arima( ) function to estimate ARIMA(p, d, q) parameters, with learning seasonal parameters as well
- We have trained a part of the dataset, and used multi-step forecast with re-estimation for the test data in streaming environment.

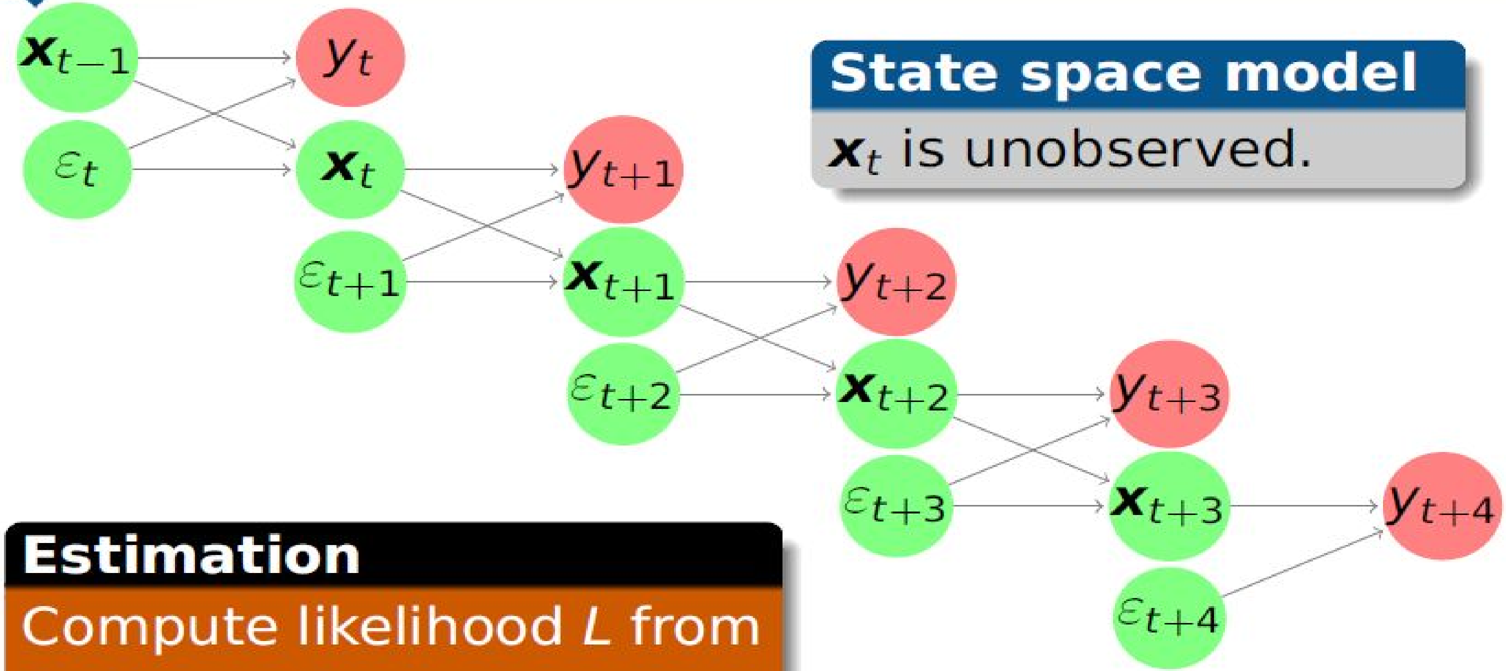
Test Code

```
t <- ts(temperature, start= start_time, deltat = 1)
train <- window(t, end=end_train)
test <- window(t, start=start_test)
fit <- auto.arima(train)
order <- arimaorder(fit)
```

Train Code

```
fc <- ts(numeric(length(test)), start=start_test, deltat = 1)
for(wind in 1:num_retrain)
{
  x <- window(temperature, end=end_train + wind_size*(wind-1))
  refit <- Arima(x, model = fit)
  temp_forecast <- forecast(refit, h=10)$mean
  for(i in 1:wind_size)
  {
    fc[wind_size*(wind-1)+i] <- temp_forecast[i]
  }
}
```

Time series forecasting



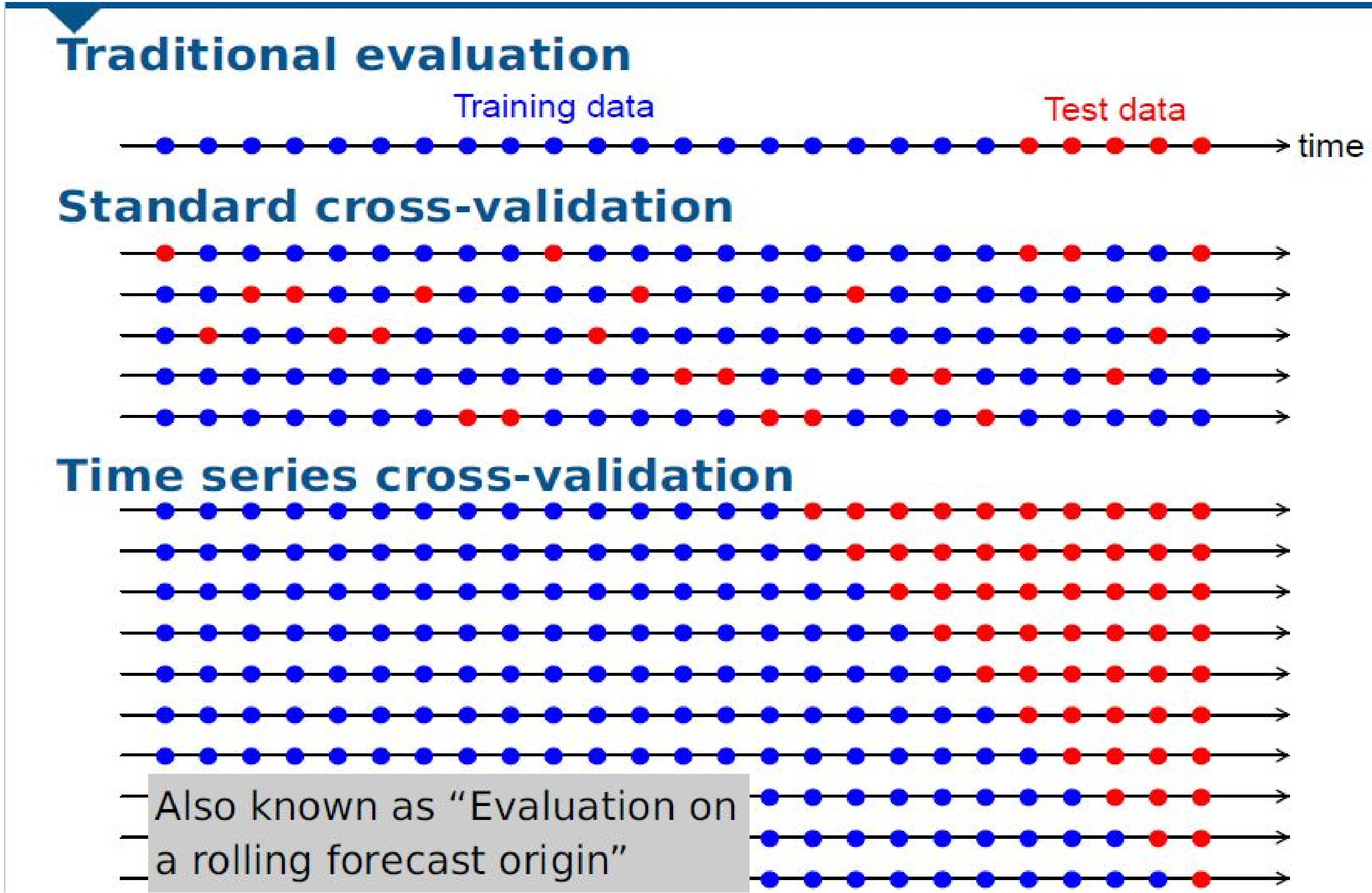
State space model

$\mathbf{x}_t$ is unobserved.

Estimation

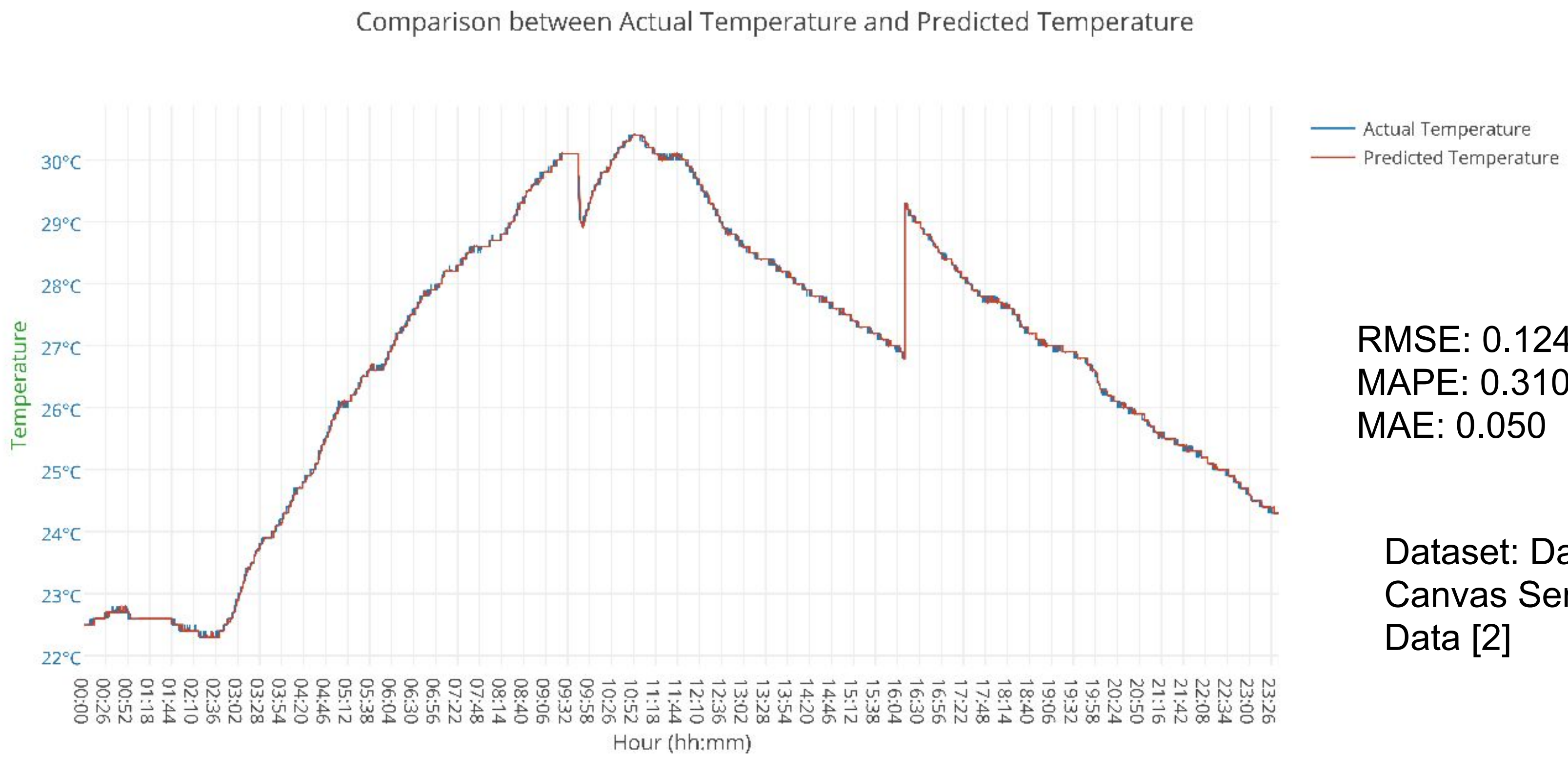
Compute likelihood L from $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_T$.
Use optimization algorithm to maximize L.

Time Series Cross-validation



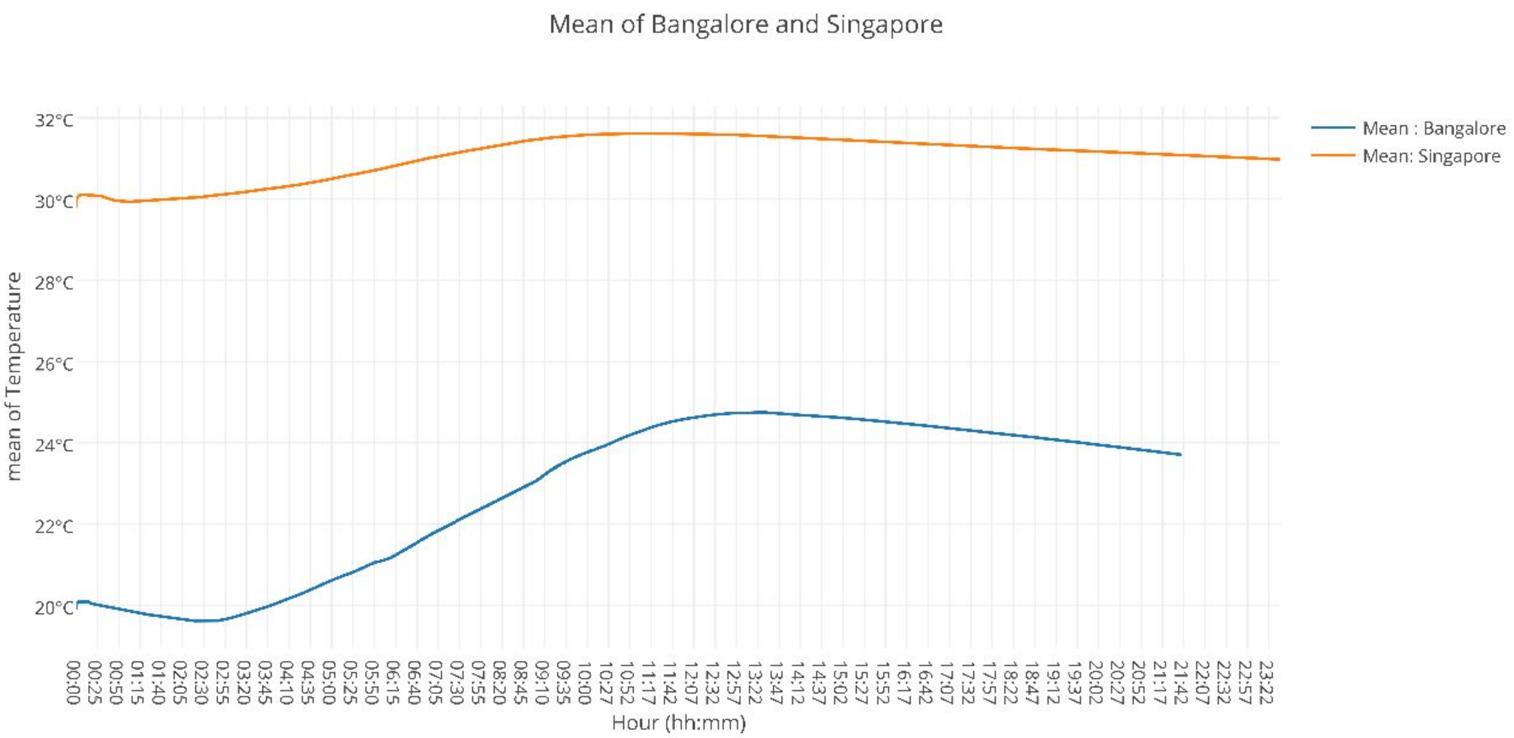
Also known as “Evaluation on a rolling forecast origin”

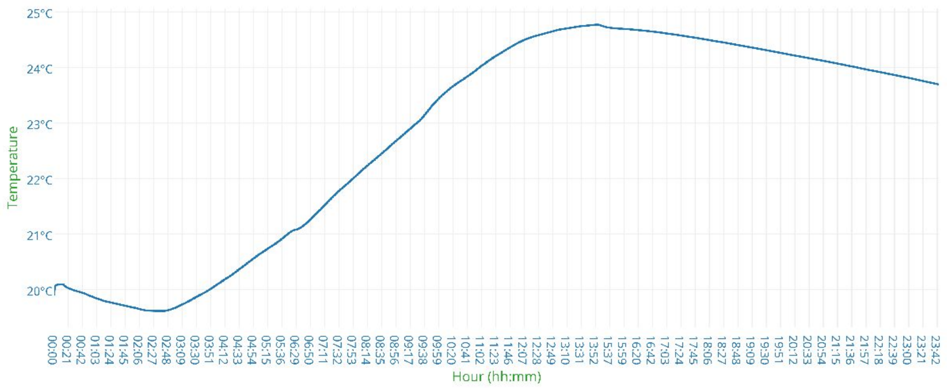
Results

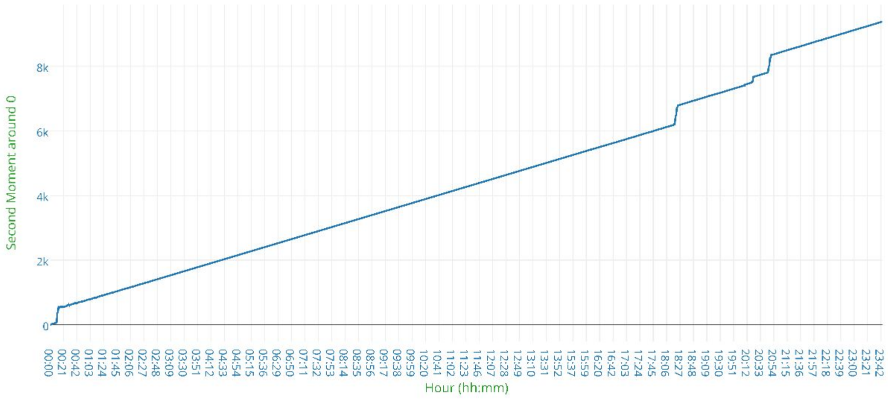


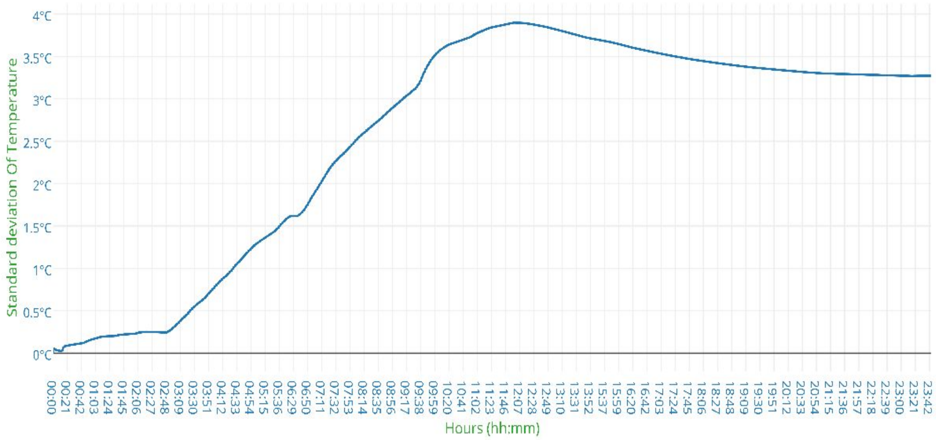
RMSE: 0.124
MAPE: 0.310
MAE: 0.050

Dataset: Data Canvas Sensor Data [2]


Mean of Bangalore and Singapore
Mean : Bangalore
Mean: Singapore
mean of Temperature
32°C
30°C
28°C
26°C
24°C
22°C
20°C
Hour (hh:mm)
00:00
00:25
00:50
01:15
01:40
02:05
02:30
02:55
03:20
03:45
04:10
04:35
05:00
05:25
05:50
06:15
06:40
07:05
07:30
07:55
08:20
08:45
09:10
09:35
10:00
10:27
10:52
11:17
11:42
12:07
12:32
12:57
13:22
13:47
14:12
14:37
15:02
15:27
15:52
16:17
16:42
17:07
17:32
17:57
18:22
18:47
19:12
19:37
20:02
20:27
20:52
21:17
21:42
22:07
22:32
22:57
23:22







Comparison of Mean, Standard deviation, second moment about 0 (in degree Celsius) for temperature in Bangalore

Conclusion

- In this project we have learned real-time analytics over high velocity streaming data
- Apache Storm scaled nicely with different input stream rates for performing statistical analysis on the sensor data.
- It can be observed from the results obtained that forecasting for weather data worked very well using auto.arima( ) with periodic retraining.
- All these study has a huge application in weather stations and pollution monitoring centers for dealing with numerous sensor data and is of a growing importance in developing future “SMART CITIES”.

Future Work

- Due to lack of JAVA based implementation of ARIMA, we used R packages which have some overhead on end to end latency for streaming applications.
- ARIMA and auto.arima( ) can be implemented in JAVA so the strom application can better scale with forecasting, decreasing the end to end latency as a whole.
- Another important study may involve comparison with different forecasting methods and how they scale with streaming applications.
- Deploy a statistical package (maybe query based) for real-time analytics of streaming data coming from weather stations and pollution monitoring centers.

References

1. A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy, “Storm@twitter,” in Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data
2. D. Canvas, “Data Canvas Dataset,” url: http://datacanvas.org/sense-your-city/, 2015..
3. A. Rajaraman and J. D. Ullman, Mining of Massive Datasets. New York, NY, USA: Cambridge University Press, 2011.
4. G. E. P. Box and G. Jenkins, Time Series Analysis, Forecasting and Control. Holden-Day, Incorporated, 1990.
5. R. J. Hyndman and Y. Khandakar, “Automatic time series forecasting: the forecast package for R,” Journal of Statistical Software, vol. 26, no. 3, pp. 1–22, 2008.
6. Y. Sakamoto, M. Ishiguro, and G. Kitagawa, Akaike information criterion statistics, ser. Mathematics and its applications. Japanese series. Tokyo: KTK Dordrecht, 1986, includes index.