Streaming Machine Learning Algorithms with Big Data Systems

Vibhatha Abeykoon, Supun Kamburugamuve, Kannan Govendrarajan, Pulasthi Wickramasinghe, Chathura Widanage, Niranda Perera, Ahmet Uyar, Gurhan Gunduz, Selahattin Akkas and Gregor Von Laszewski

Indiana University Bloomington

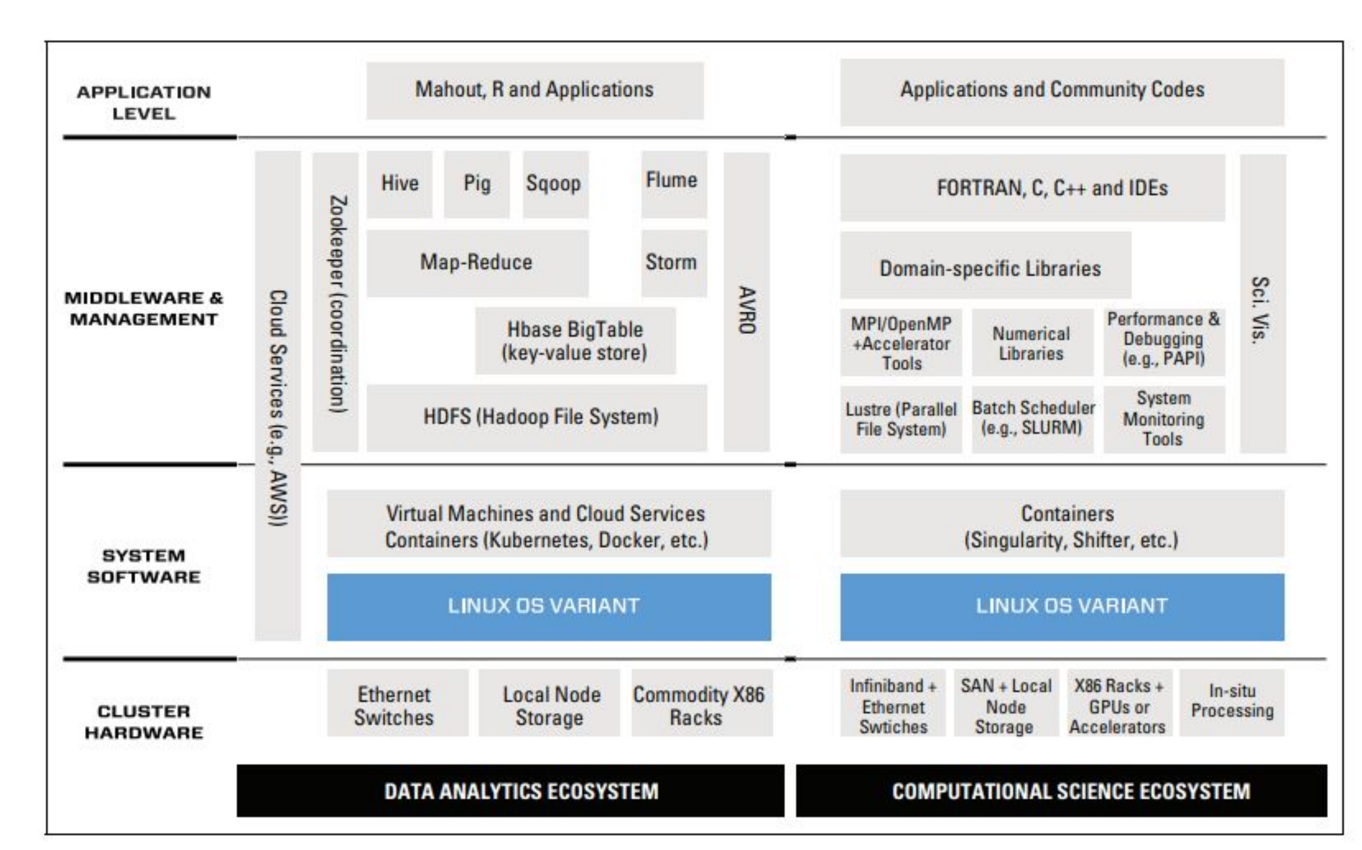
Motivation

- Data volume generated per day is increasing in a very high rate.
- Low latency is a must for increasing consumer demand on various services.
- Existing batch algorithms need to be optimized for online learning.
- Machine learning algorithms has become very important when formulating most of the supervised learning problems with less computing power.

How to design Streaming Machine Learning algorithms?

- Simply need to do train a machine learning algorithm in real-time without storing a large batches of data.
- Some algorithms can be trained by just observing a datapoint only once.
  - ***Initialization stage:*** Observe a number of data points (K elements at least if it is a clustering problem, depending on the algorithm this must be well-defined).
  - ***Model Evaluation:*** Calculate a gradient or model value for the observed elements.
  - ***Model Synchronization:*** Synchronize the model value across all the processes when using distributed training.
  - Re-do the whole process per element after the ***initialization stage***.
- Some algorithms need an iterative streaming algorithm to ensure the accuracy to be in an expected level.
  - ***Model evaluation:*** Here we observe ***w*** number of elements by formulating a window in a stream and do an iterative computation on it for ***t*** iterations. Here ***t <<< T***, ***T*** refers to the number of iterations required in batch mode to compute the optimum model.

Convergence of HPC and Big Data



Reference: https://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/bdec_pathways.pdf

Objective

- Design low-latency training on big data systems and identifying effective systems for online training
- Provide API solutions to design streaming applications on both HPC and dataflow programming models.
- Evaluate the importance of HPC frameworks for strengthening the big data stack for intensive computations.

Streaming Machine Learning Algorithms

- Non-Iterative Setting
  - KMeans Clustering
- Iterative Setting
  - Support Vector Machine (Linear Kernel for Binary classification)

Streaming SVM

Algorithm 1 Iterative SGD SVM

INPUT: $[x, y] \in S, w \in R^d, t \in R^+$
OUTPUT: $w \in R^d$
procedure ISGDSVM(S, w, t)
    for $i = 0$ to n **do**
        if $(g(w; (x_i, y_i)) == 0)$ **then**
            $\nabla J^t = w$
        else
            $\nabla J^t = w - C x_i y_i$
        $w = w - \alpha \nabla J^t$
    return w

Algorithm 2 Iterative Streaming SVM

INPUT: $X_\infty, Y_\infty \in S_\infty, w \in R^d, l \in R^+, s \in R^+, m < K, m \in R^+$
OUTPUT: $w \in R^d$
procedure ISSVM($\bar{S}_i, w, T, l, s$)
    In Parallel K Machines $[\bar{S}_1, ..., \bar{S}_b] \subset S_\infty$
    procedure WINDOW($\bar{S}_m, w, l, s$)
        for $t = 0$ to T **do**
            procedure ISGDSVM($\bar{S}_m, w, t$)
        All_Reduce(w)
    return w

Streaming KMeans

Algorithm 3 Online KMeans

INPUT: $X = \{x_1, ..., x_m\}, x_i \in \mathbf{R^m}$

$V = \{v_1, ..., v_k\} v_i \in \mathbf{R^m}, k \leq n$

OUTPUT: V

procedure STREAMING-KMEANS(X, V)

    procedure WINDOW($\bar{X}, \bar{V}$)

        for x_j in $\bar{X}$ **do**

            if $j \leq k$ **then**

                $v_i = x_j$

                $k_i = 1$

                $i = i + 1$

            else

                $v_i = argmin_i ||x_j - v_i||$

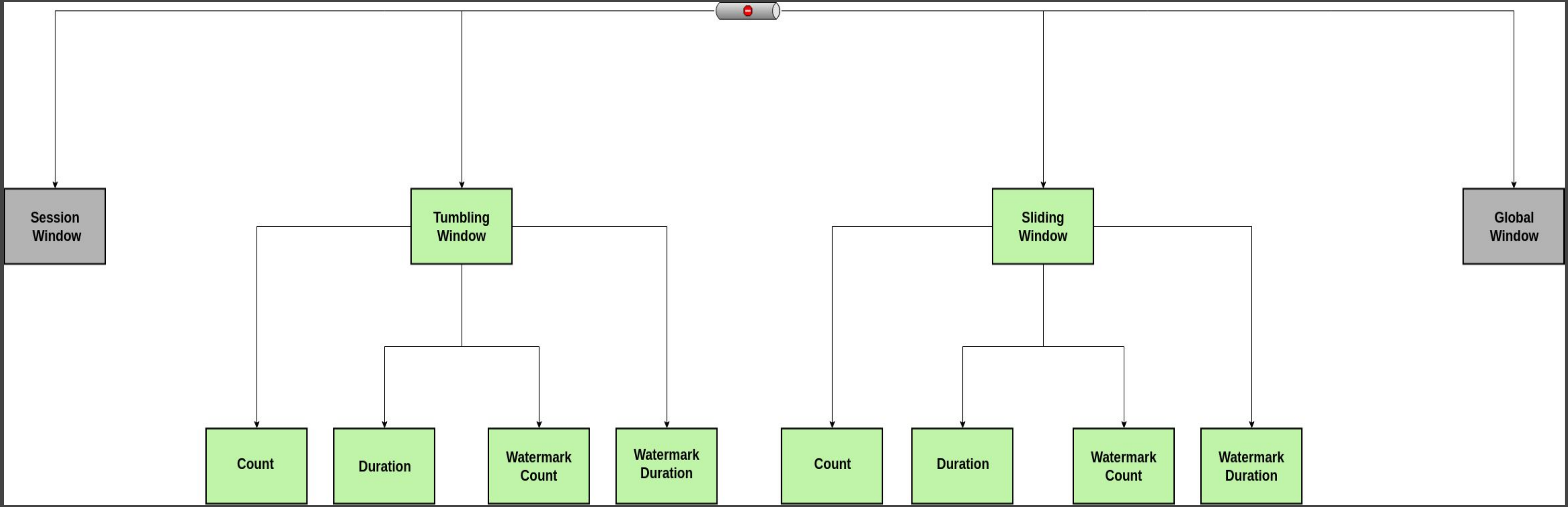
                $v_i = v_i + \frac{1}{n_i + 1}[x_j - v_i]$

                $n_i = n_i + 1$

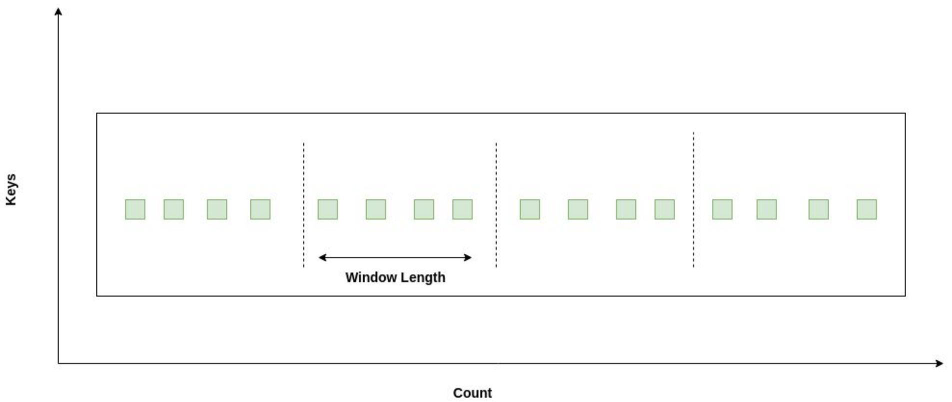
        All_Reduce(V)

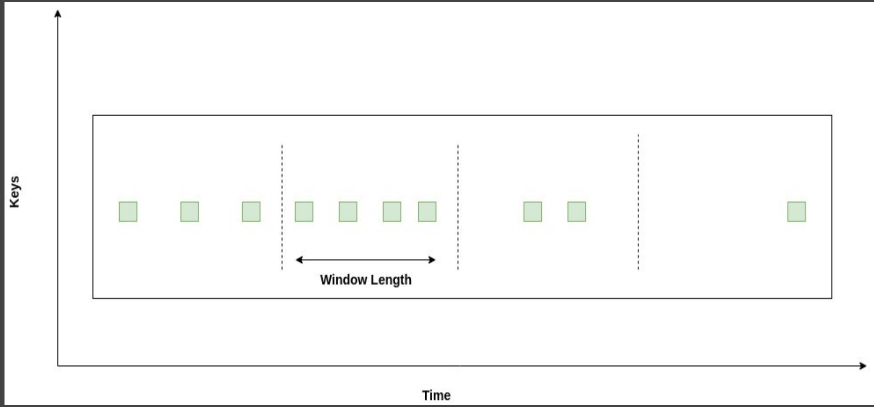
    return V

Discretization of a Stream

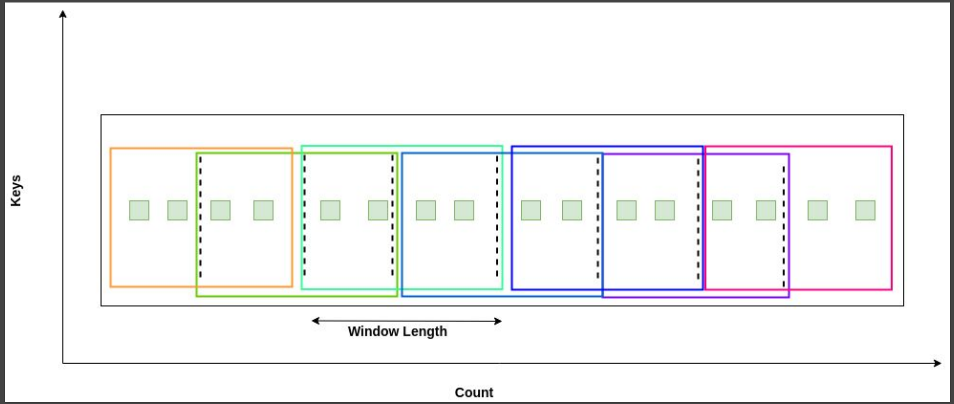


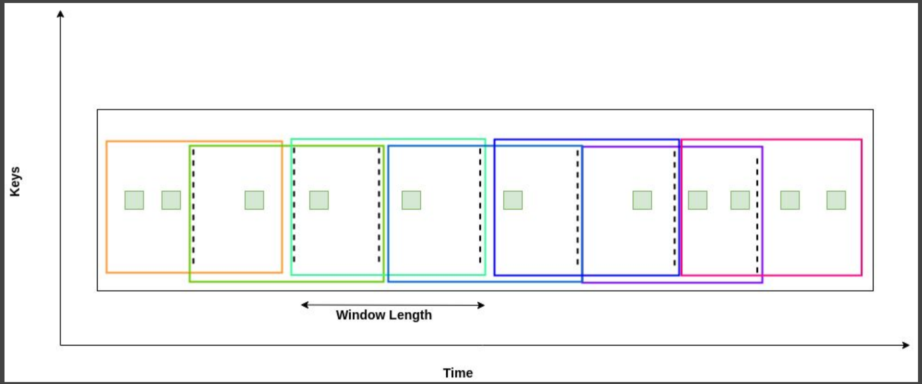
Tumbling Windows



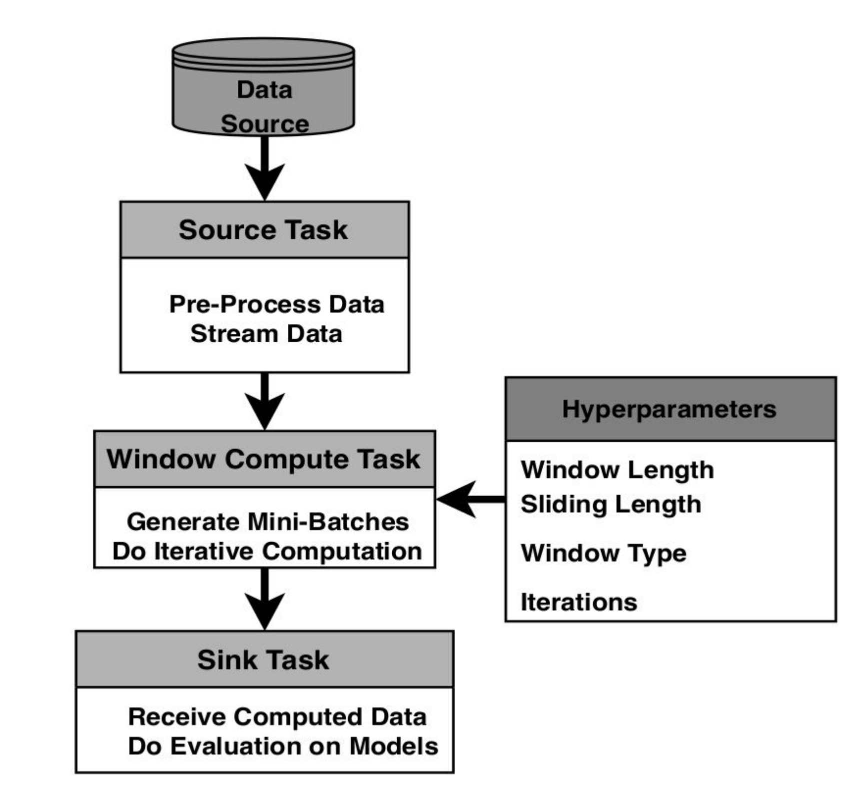


Sliding Windows





Workflow of a Streaming ML Algorithm



Streaming Platforms

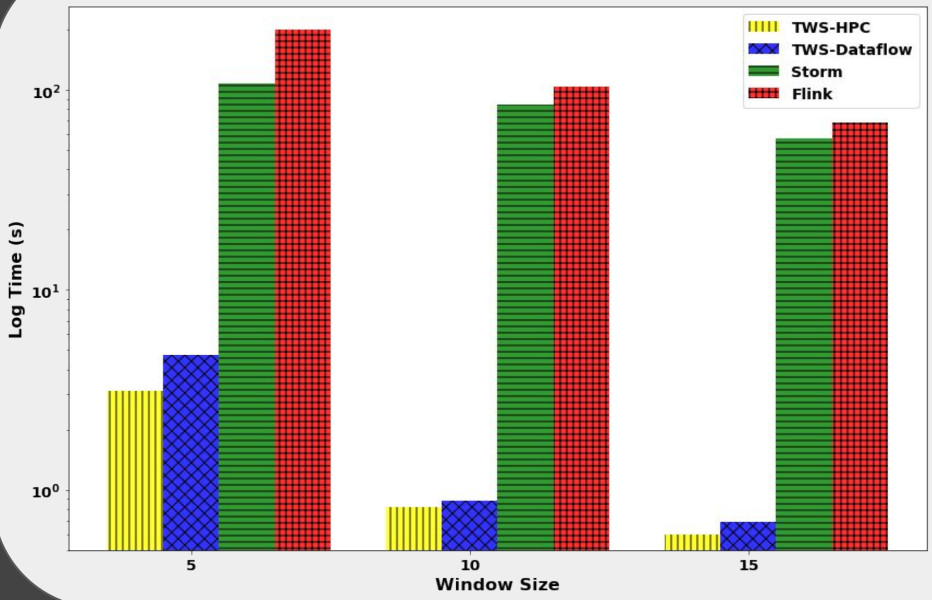
	APACHE STORM	Flink	Twister2
Iterative Streaming Support	✓	✓	✓
Dataflow Model	✓	✓	✓
HPC Model	✗	✗	✓

Apache Storm v1.2.8
Apache Flink v1.9.0
Twister2 v0.3.0

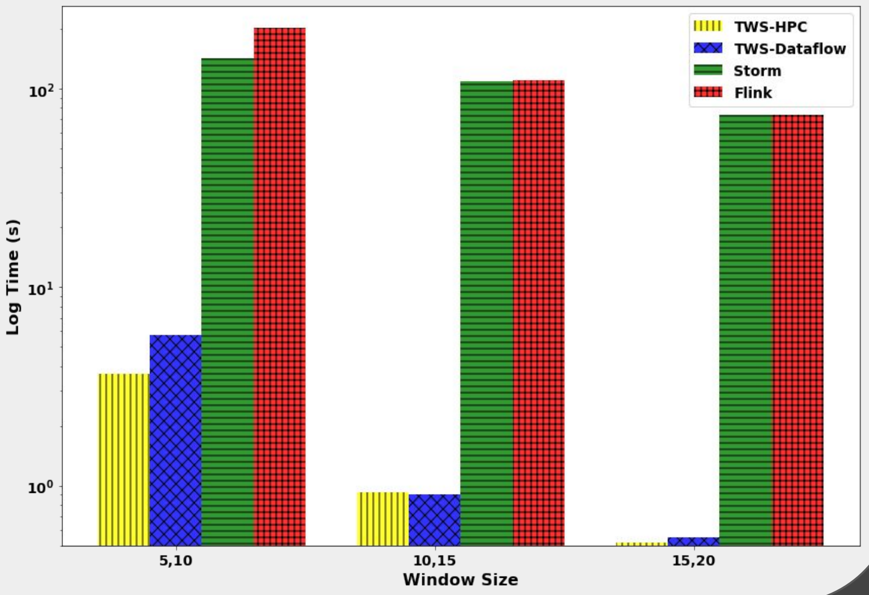
Experiment Configuration

- Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10 GHz (250 GB RAM)
- Streaming SVM :Binary Classification on 49K long stream for training and 90K sample for model testing.
- Streaming KMeans: Clustering 1000 centroids, 49K long stream for training)
- 8 Physical nodes each with 16 processes (128 parallelism).
- Use count-based window setting to do a stress test on each big data framework used.

Streaming SVM



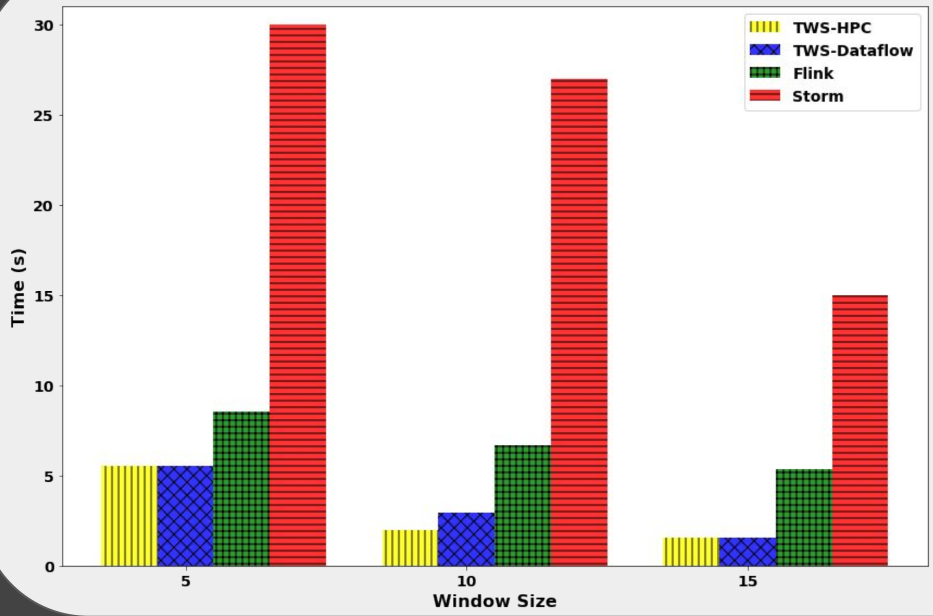
Tumbling Windowing



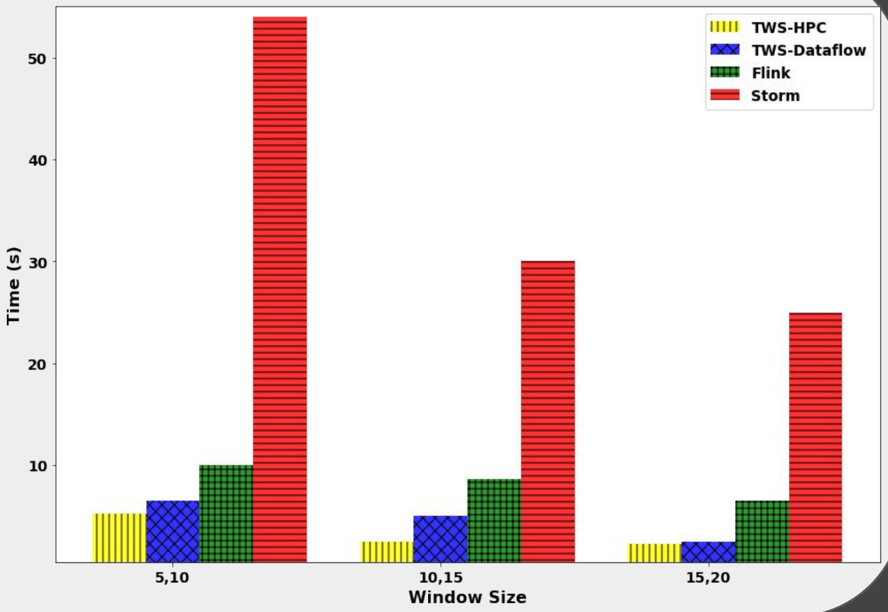
Sliding Windowing

*5,10 refers to sliding length,window length.
Obtained after experimenting with different configs towards optimum results obtained in batch mode.

Streaming KMeans



Tumbling Windowing



Sliding Windowing

*5,10 refers to sliding length,window length.
Obtained after experimenting with different configs towards optimum results obtained in batch mode.

Conclusions and Future Work

- Windowing APIs are vital for designing iterative streaming applications.
- High performance computing model can be adopted in Big Data frameworks to provide better performance for streaming applications.
- *Experimenting with a larger data stream (minimum of 1 Million of more data points per a job)*
- *Structured data streaming with stream discretization.*
- *Expanding experiment configurations for testing window config sensitivity on algorithm convergence.*
- *Scaling for a bigger experiment setting (1024+ cores)*
- *Extending experiments for more machine learning algorithms.*

Thank you

- NSF
- Future Systems Team @ IU (Allan Streib et. al)
- Digital Science Center