Adaptive Watermarks: A Concept Drift based Approach for Predicting Event-Time Progress in Data Streams

Ahmed Awad, Jonas Traub*, Sherif Sakr
University of Tartu, Estonia
*Technische Universität Berlin, Germany

Introduction

Event-time based stream processing is concerned with analyzing data with respect to its generation time. In most of the cases, data gets delayed during its journey from the source(s) to the stream processing engine and might arrive out-of-order and late.

Among the different approaches for out-of-order stream processing, low watermarks are proposed to inject special records within data streams, i.e., watermarks. A ***watermark*** is a timestamp which indicates that no data with a timestamp older than the watermark should be observed later on. Any element as such is considered a late arrival.

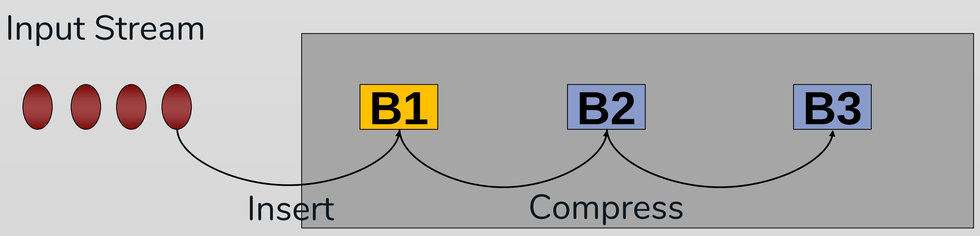
Problem

Watermark generation is usually *periodic* and *heuristic-based*. That is, application developers usually choose, heuristically, a fixed period at which a check for watermark generation is repeated. Additionally, they usually specify a fixed delay. The new watermark value is usually the maximum timestamp seen so far in the data minus the delay. Although simple, this approach is *inflexible* with respect to the *frequency of data arrival* as well as the *delay* that data may encounter. As stream processing pipelines are long-running applications, delays in data arrival may change over time and, data arrival rate may change as well. In such cases, the fixed delays may lead to either higher latency or less accuracy.

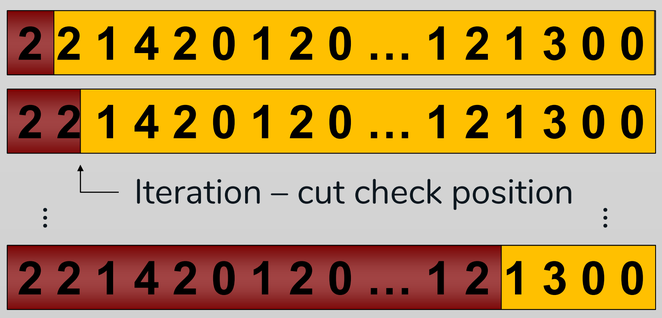
Solution

We propose an adaptive watermark generation strategy. Our strategy decides adaptively when to generate watermarks and with what timestamp without a priori adjustment. We treat changes in data arrival frequency and changes in delays as concept drifts in stream data mining. We use an **Adaptive Window (ADWIN)** as our concept drift sensor for the change in the distribution of arrival rate and delay. Moreover, we allow application developers to control the number of late arrival elements, within limits.

Adaptive Windowing (ADWIN) [Bifet '07]



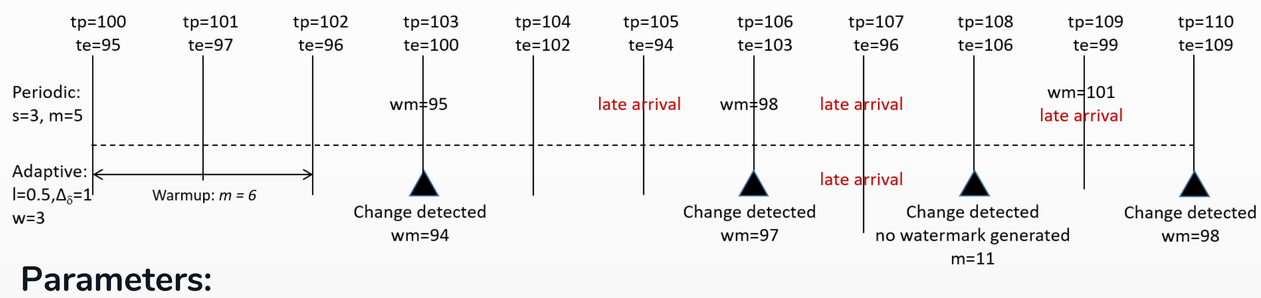
Adwin maintains an adaptive window which is the basis for computing *Concept Drift detection*. Adwin grows the window as long as there is no concept drift detected.



How Adaptive Watermark Generation Works

For the first w tuples, the adaptive watermark generator learns the average delay of incoming tuples ***m***. For up-coming tuples, the skewness between their event time and ingestion time is fed to ADWIN. Thus, the change in skewness is treated as a concept drift. ADWIN has a single parameter d which controls the sensitivity for change detection. If ADWIN detects a change, a new watermark is only generated if the rate of the late arrivals since last watermark is less than ***l*** (late arrival threshold). In case the late arrival rate is higher than ***l***, we decrease the sensitivity δ by a factor of Δ_δ. The set of tuples received between to watermarks is called a chunk. Within a chunk a new value for m is learned. The set of tuples received between to watermarks is called a chunk. Within a chunk a new value for ***m*** is learned. Whenever a new watermark the chunk is reset.

All the parameters used for either the adaptive or the periodic watermark generator are described in the table. The Figure below illustrates how adaptive watermark generation works in contrast to periodic watermark generation.



Parameters:

Parameter	Description
δ*	Sensitivity to change ∈ [0, 1]. Default is 1.
l	Late arrival threshold, l ∈ (0, 1]. Default is 1.
m*	Skewness between event time and ingestion time. (allowed lateness)
Δ_δ	Sensitivity change ratio, Ωδ ∈ (0, 1]. Default is 1.
w	Warmup tuples used to initialize m
s	The period between two watermarks

Evaluation

Setup: we have implemented adaptive watermark generation on top of Apache Flink v1.6.2, using the Source APIs to control the emission of watermarks. We compare the adaptive watermark generation against the baseline periodic generator.
Data: we use two data sets from the DEBS grand challenges of 2012 and 2015 respectively. The DEBS 2012 data set has 32,390,519 tuples and 1.5% of the elements arrive out-of-order with an average of 100 tuples per second. The DEBS 2015 data set has 14,776,616 tuples and has 78.6% of its tuples arrive out-of-order.
Metrics: We measure two metrics: **the percentage of dropped elements** and the average delay between a window end and the watermark after which the window function was triggered. We call it the **window delay** and report it in milliseconds.

Results:

Periodic generator

Data set	m	s	Win. Size	Dropped %	Avg. win. Delay (ms)
DEBS 2012	1000	200	1000	1.24	9,452,454
			100	1.24	3,083,109
	100	10	1000	1.50	713,846
			100	1.50	175,497
DEBS 2015	1000	200	1000	98.72	644,046,759
			100	98.62	648,821,195
	100	10	1000	99.93	546,682,126
			100	99.97	392,904,397

Adaptive generator

Data set	Δ^δ	l	Win. Size	Dropped %	Avg. win. Delay (ms)
DEBS 2012	1	1	1000	1.49	23,633
			100	1.49	16,796
	1	0.01	1000	1.17	2,467
			100	1.17	2,469
DEBS 2015	1	1	1000	79.60	72,863,504
			100	79.60	72,863,574
	1	0.01	1000	41.64	92,672,393
			100	41.64	92,673,050

Open Source Repository

https://github.com/DataSystemsGroupUT/Adaptive-Watermarks