21st International Conference on Extending Database Technology (EDBT), March 26-29, 2018, Vienna

Scalable Detection of Concept Drifts on Data Streams with Parallel Adaptive Windowing

Philipp M. Grulich philipp.grulich@campus.tu-berlin.de
René Saitenmacher r.saitenmacher@campus.tu-berlin.de
Jonas Traub jonas.traub@tu-berlin.de
Sebastian Breß sebastian.bress@dfki.de
Tilmann Rabl rabl@tu-berlin.de
Volker Markl volker.markl@tu-berlin.de

Abstract

Stream analysis suffers from concept drifts such as changed user preferences or varying weather conditions. These concept drifts cause wrong predictions and lead to incorrect business decisions.

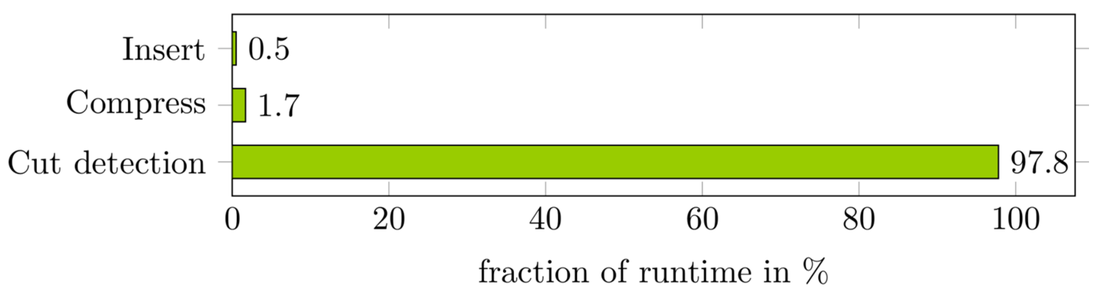
Adaptive Windowing allows for **adapting to concept drifts on the fly**.

In this paper, we examine **Adaptive Windowing (Adwin)** in detail:

1. We analyze Adwin, point out its scalability limitations, and identify its bottlenecks.
2. We parallelize Adwin to overcome its bottlenecks and to provide scalable concept drift adaptation in real-time.
3. We evaluate the latency and throughput of our solution.

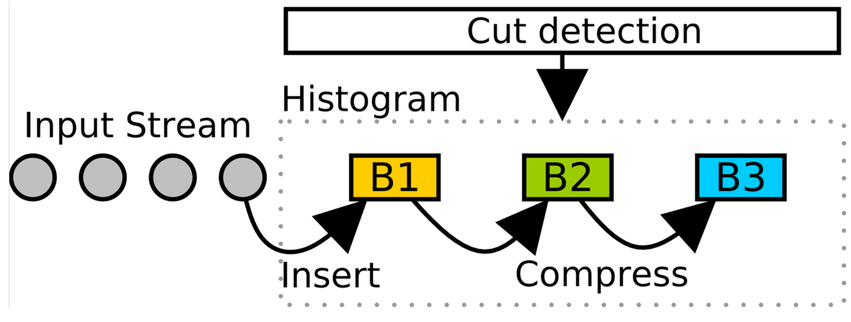
We explore parallel adaptive windowing to provide **scalable concept drift detection for high-velocity data streams.**

Initial Performance Analysis



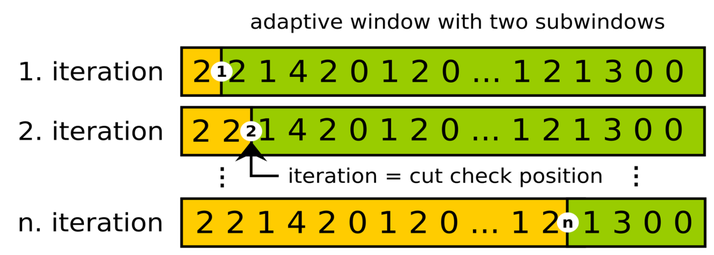
Observation: Cut detection dominates the overall workload and blocks the processing of arriving tuples.

Adaptive Windowing (Adwin) [Bifet '07]

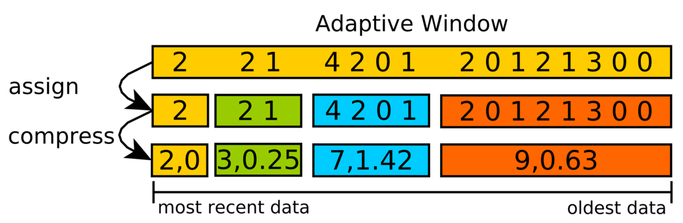


Adwin maintains an adaptive window which is the basis for computing a ML model. Adwin grows the window as long as there is no concept drift detected. As a result, the model can rely on growing training data.

Processing Tuples:
1.) Add tuple to adaptive win.
2.) Check for concept drifts.



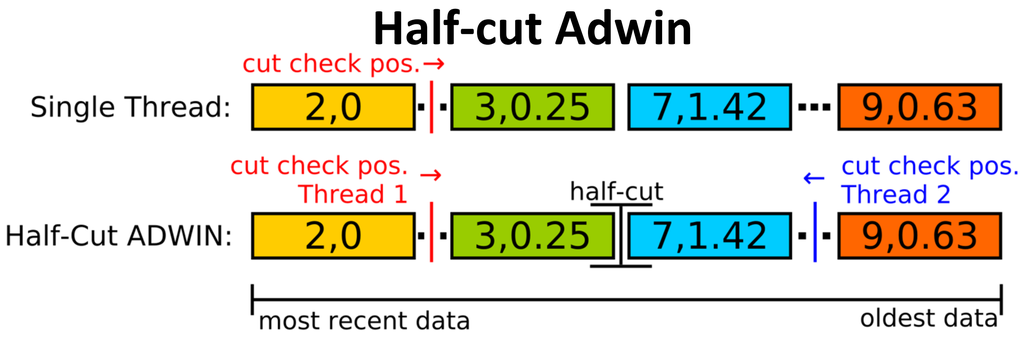
Detecting Concept Drifts:
1.) Split adaptive window in two sub-windows and compare their contents to detect drifts.
2.) If concept drift detected, remove oldest data from adaptive window (*cut* adaptive window) and restart drift detection.
3.) Iterate over all possible combinations of sub-windows.



Optimization:
Store adaptive window as exponential histogram.

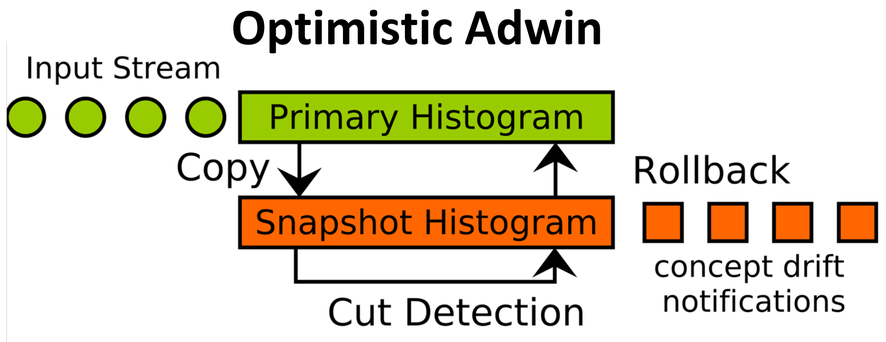
Parallel Adaptive Windowing

1) Parallelization of Cut Detection



We halve the latency of the cut detection by introducing a second thread which moves opposite to the first thread and performs half of the cut checks. This approach can be extended to more threads.

2) Decoupling of Cut Detection



We decouple cut detection from the insertion of stream tuples using multi-version concurrency control (MVCC). We execute cut detection on a snapshot histogram and roll back in case of concept drifts.

Open Source Repositories

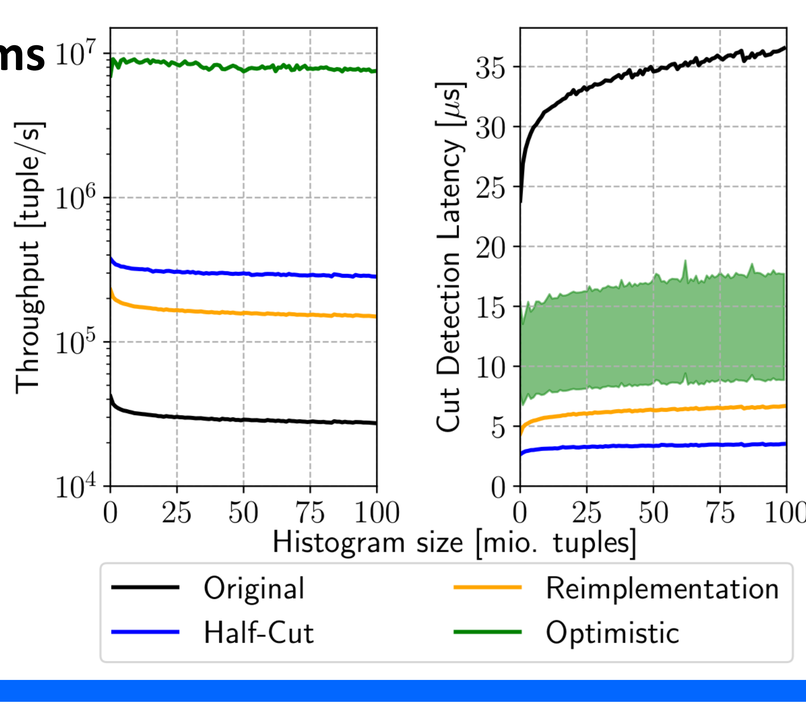
Parallel Adwin open source repository:
github.com/TU-Berlin-DIMA/parallel-ADWIN

Original Adwin open source repository:
github.com/abifet/adwin

Evaluation

Scaling to high-velocity data streams

- Optimistic Adwin is two orders of magnitude faster than the original Adwin implementation and 54 times faster than an optimized sequential implementation.
- Half-Cut Adwin is 10 times faster than the original Adwin implementation and almost 2 times faster than an optimized sequential implementation.
- Both have a low microseconds latency.



Technische Universität Berlin
German Research Center for Artificial Intelligence

This work received funding through the EU Horizon 2020 project Streamline (688191), the DFG project Stratosphere (606902) and from the German Ministry for Education and Research as Berlin Big Data Center (01IS14013A) and Software Campus (01IS12056).