Improving queries over streaming data

Ahmed Amr

German University in Cairo

June 9, 2019

Overview

Introduction

Definition

Data Stream Management System

A data stream management system (DSMS) is a computer software system to manage continuous data streams. It is similar to a database management system (DBMS), which is, however, designed for static data in conventional databases. A DSMS also offers a flexible query processing so that the information need can be expressed using queries

Functional principle

DBMS	**DSMS**
Persistent data	volatile data streams
Random access	Sequential access
unlimited secondary storage	limited main memory
relatively low update rate	potentially extremely high update rate

Table: DSMS vs DBMS

Background

Background

- Stream Query Processing consist of a set of operators with queues over multiple sources.
- Stream Query Optimization is the process of modifying a stream processing query
- Sliding Window technique is most popular techniques for rolling data to DSMS
- Apache Storm is most popular streaming management system platform. and, Streaming Typologies can be implemented with *Java*

Aim of the project

The aim of the project is to design and implement Data Streaming Topology that can classify top N words from continuous streaming data on Twitter API. apply ranking techniques and compare standard implementation with modified one

Motivation

Motivation

Problem Statement

There are **M** tuples of data enter to streaming system.
What is the time consumption to update top **N** Objects ?

Methodology

Architecture Overview

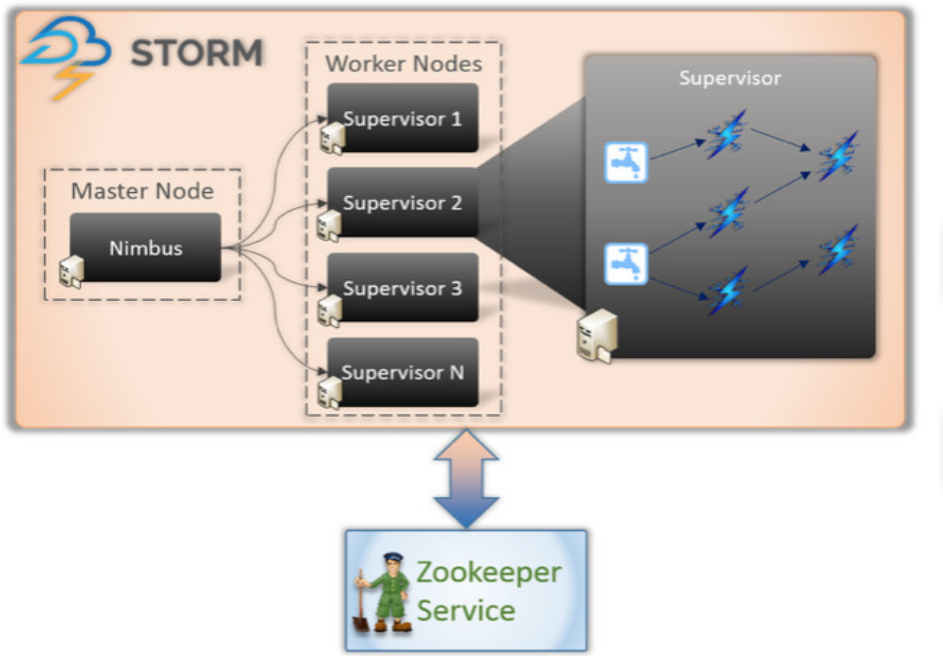

Figure: Node Services

Architecture Overview

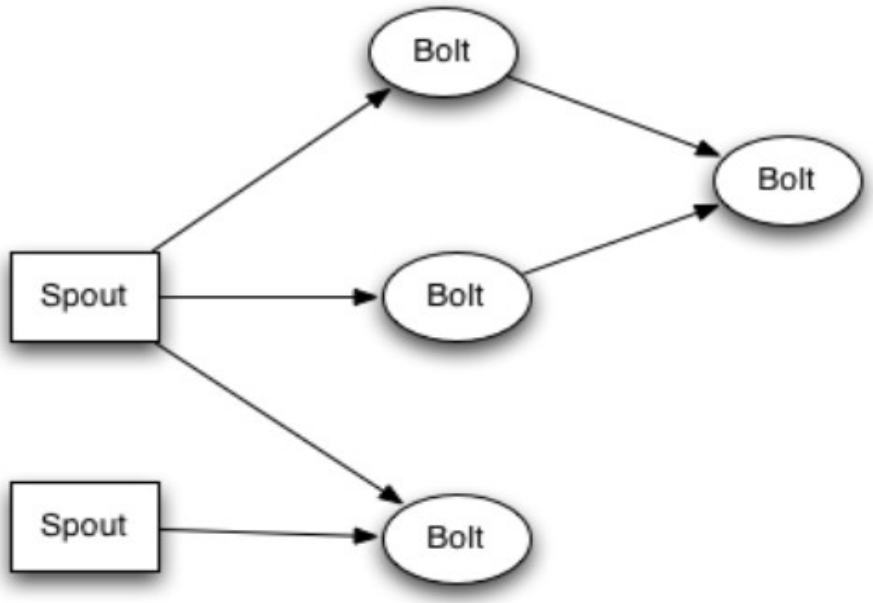

Figure: Task Services

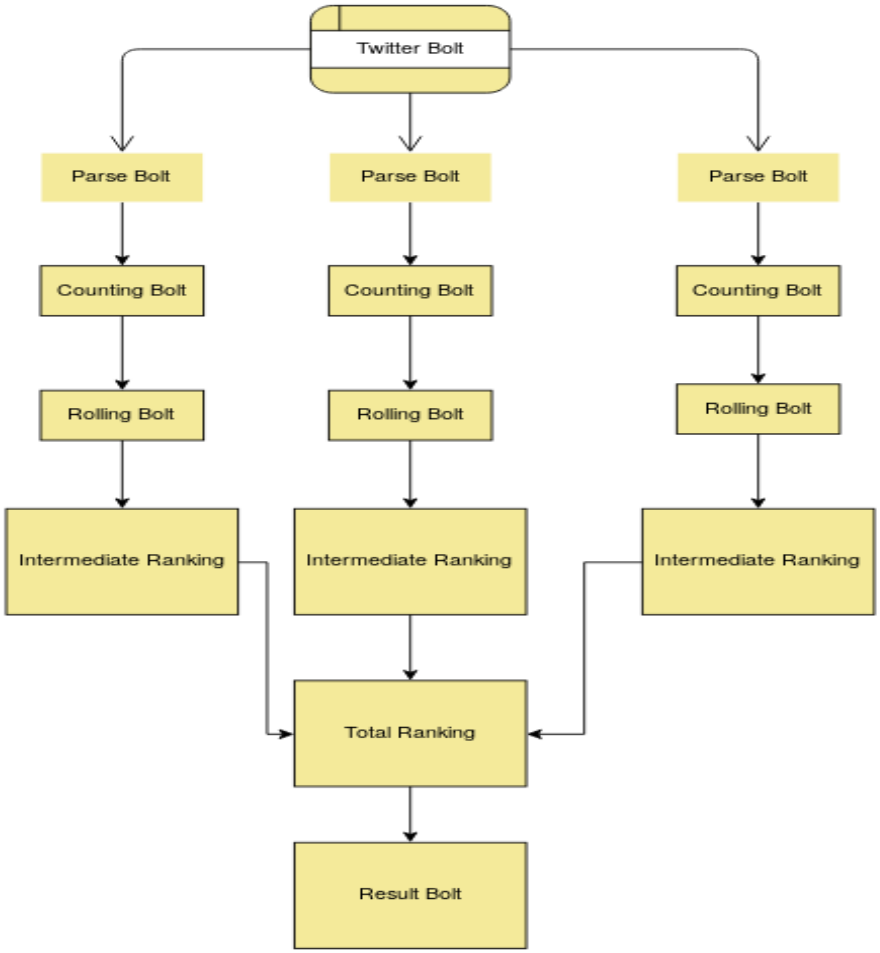

Figure: Storm Topology Architecture

Sliding Window

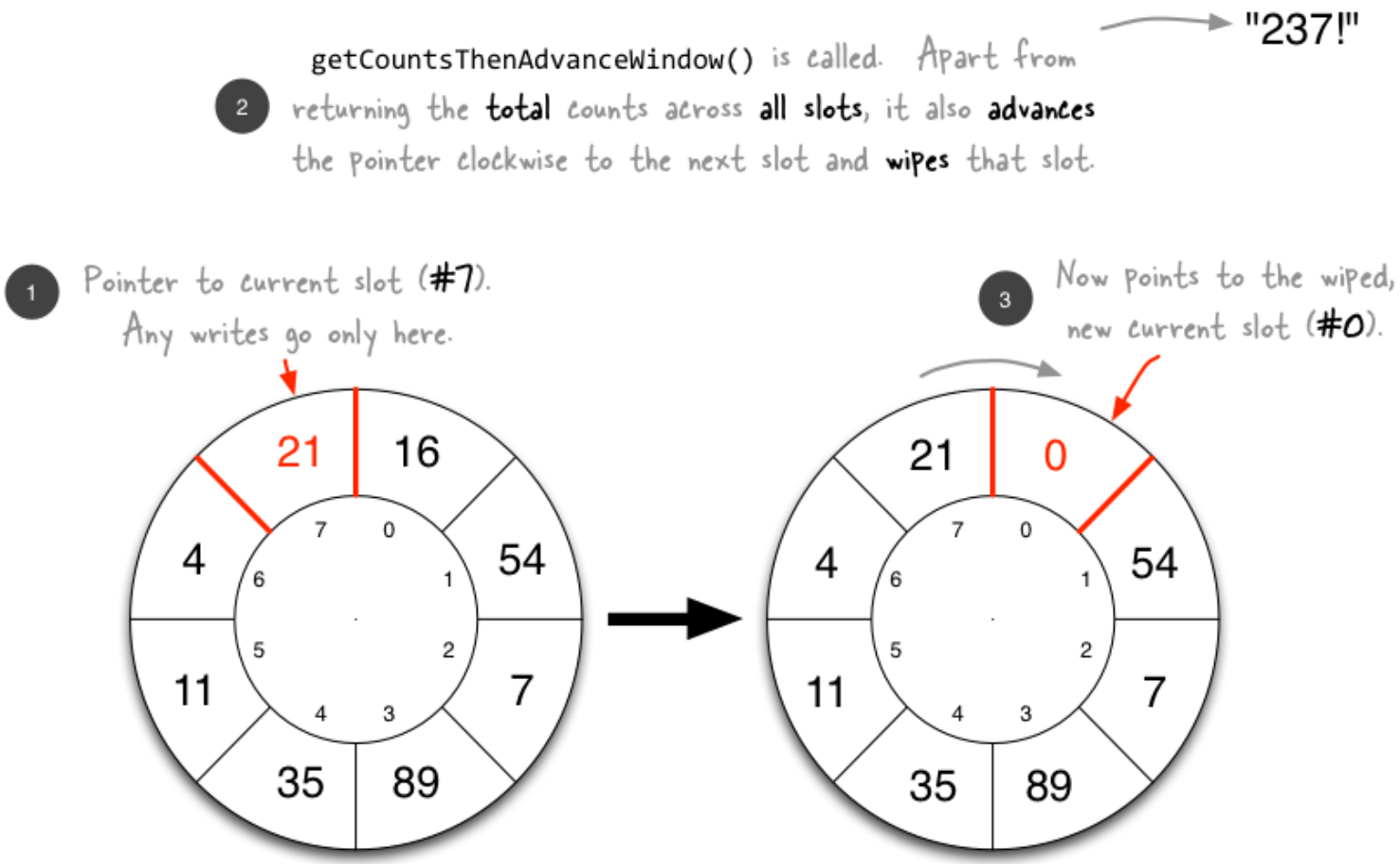

Figure: The Sliding Window Counter

Sliding Window

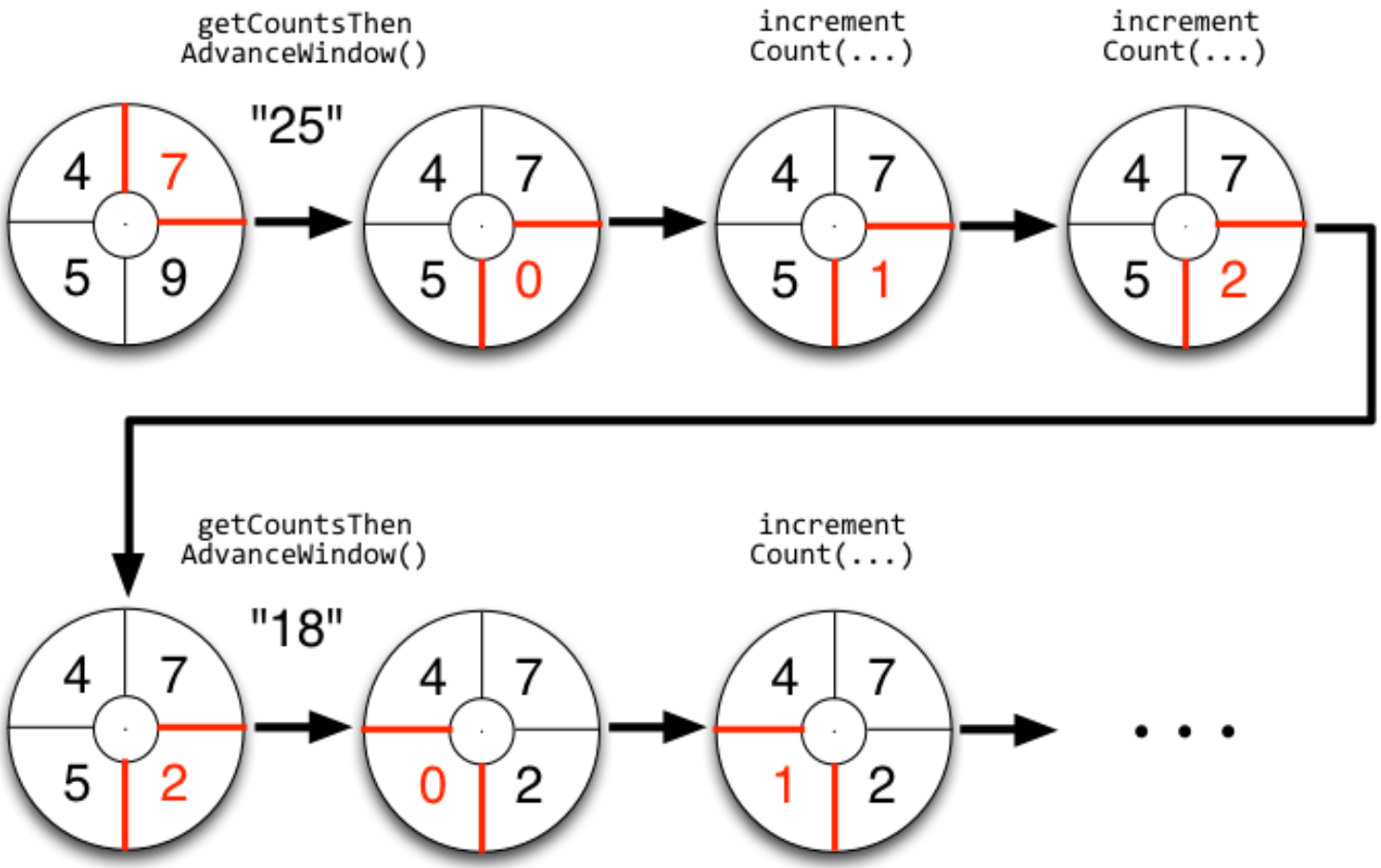

Figure: Example for Sliding Window behavior

Ranking Technique

Ranking Technique

There is one method which can affect more significantly in time complexity which is updateWith() method.

Standard Implementation (Intermediate Ranking)

```
// Worst Case Complexity(O(N))
public void updateWith(Rankable r)
{
   // O(N)
   Integer rank = rankElements.indexOf(r) ;
   if(rank == -1) // O(1)
      rankElements.add(r)
   else // O(1)
      rankElements.set(rank , r) ;

}
```

Standard Implementation (Total Ranking)

```
// Worst Case Complexity(O(M * N log N))
for(Rankable r : merge.getRanking()){ //O(M)
   // O(N)
   Integer rank = rankElements.indexOf(r) ;
   if(rank == -1)
      rankElements.add(r) ; // O(1)
   else
      rankElements.set(rank , r) ; // O(1)
   Collections.sort(rankElements,
   Collections.reverseOrder()); // O(N log N)
   if(rankElements.size() > topN)
      rankElements.remove(0) ;
}
```

Modified Implementation (Data Structures)

- AVL Tree with time complexity for insertion and deletion O(log n).
- Priority Queue with time complexity for insertion and keep all elements sorted in O(log n).

Modified Implementation (Intermediate Ranking)

```
private final AVL_Tree<Rankable> rankElements ;
```

```
public void updateWith(Rankable r)
// Worst Case Scenario O(log N)
{
   rankElements.add(r) ; //log(N)
}
```

Modified Implementation (Total Ranking)

```
private final PriorityQueue<Rankable> queue;
```

```
public void updateWith(Ranking merge) //O(M log N)
{
   for(Rankable o : merge.rankElements) // O(M)
   {
      queue.add(o); // O(log N)
      if(queue.size() > topN)// O(1)
         queue.poll() ; // O(log N)
   }
}
```

Evaluation and Results

	Standard	**Modified**
Intermediate Ranking	O(N)	O(log N)
Total Ranking	O(M * N * log(N))	O(M * log(N))

Table: Standard Implementation vs Modified Implementation

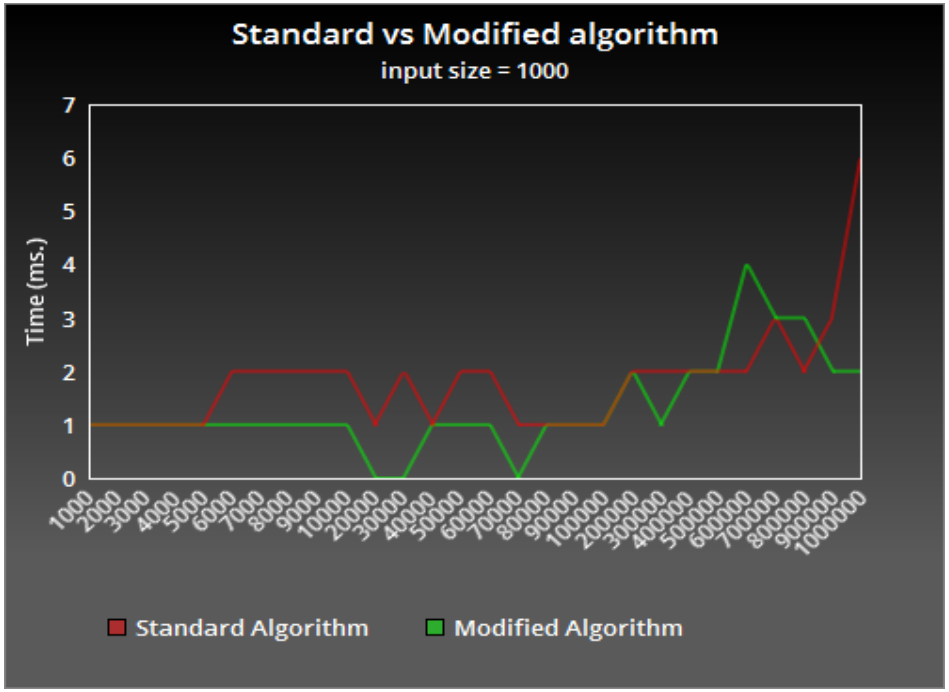

Figure: input size 10^3

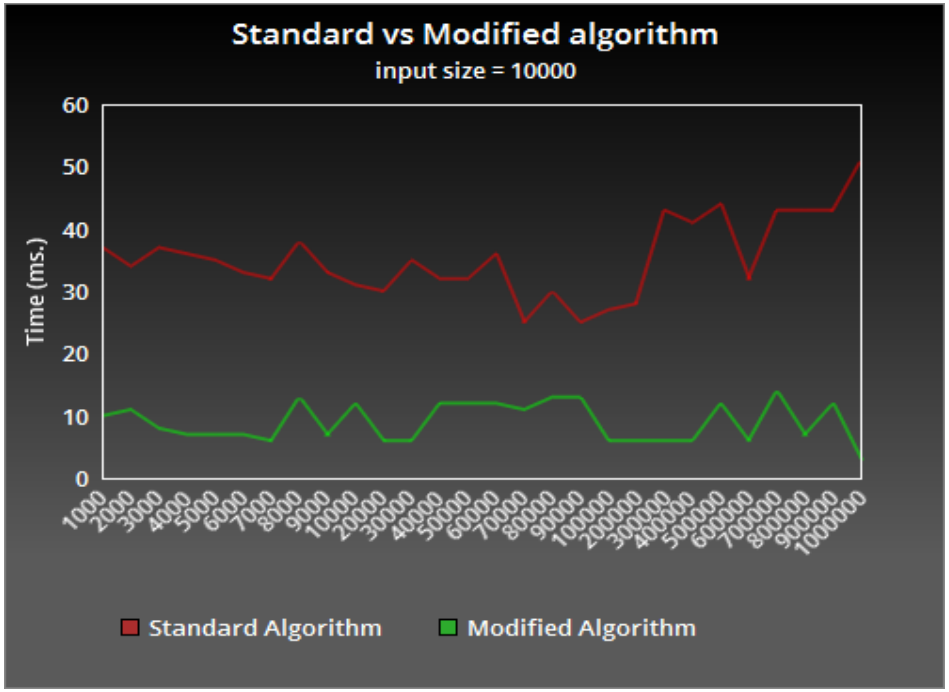

Figure: input size 10^4

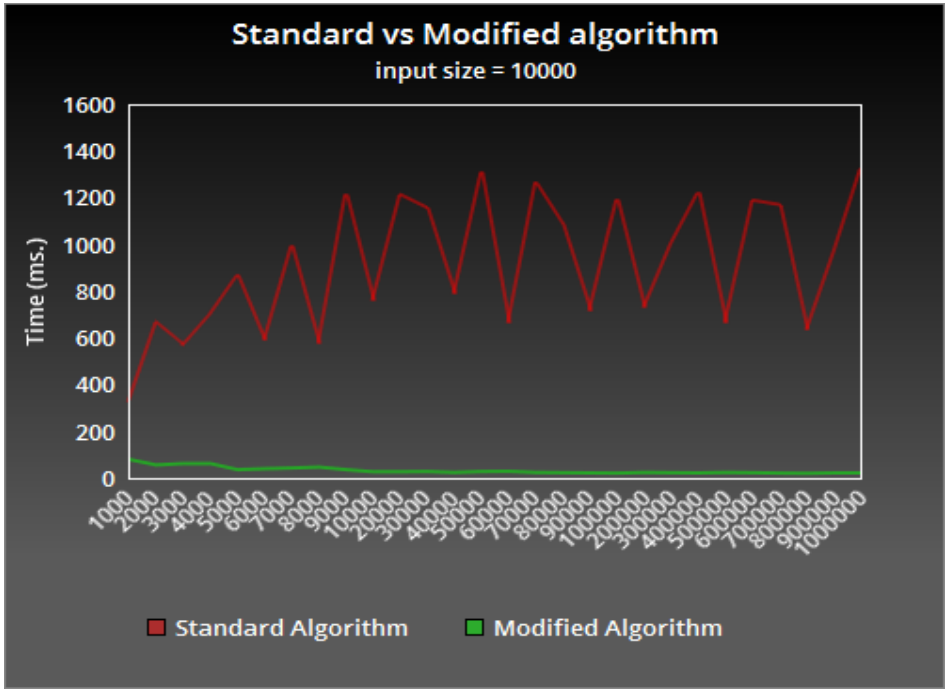

Figure: input size 10^5

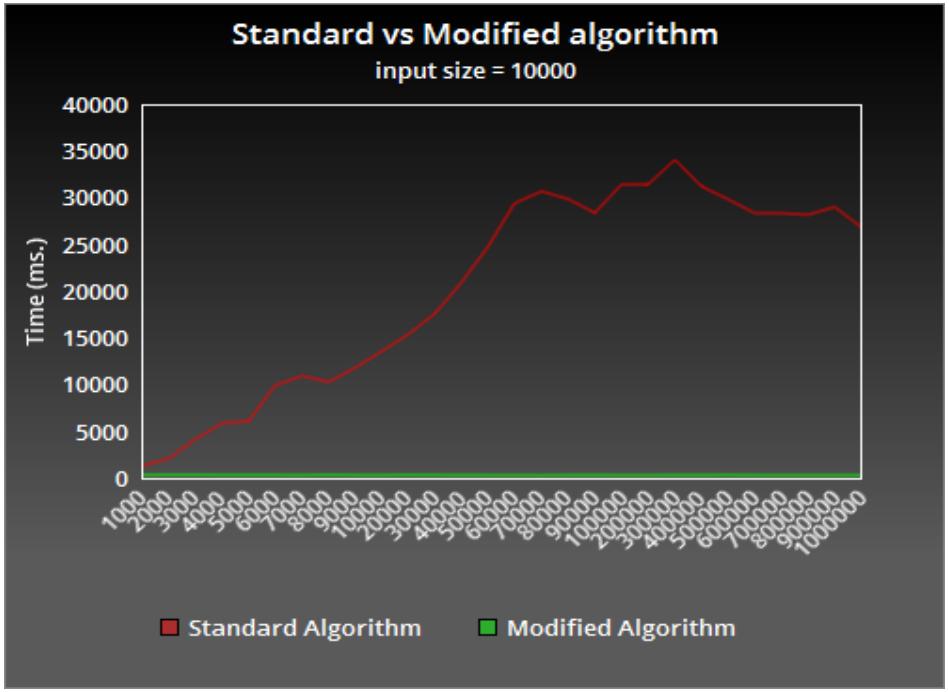

Figure: input size 10^6

Conclusion

Conclusion

- We identify streaming systems with continuous streaming data .
- We use apache storm to implement data streaming management system.
- We implement standard algorithm that already predefined in Twitter to get top N words.
- We identify a new data structures AVL Tree and Priority Queue to modify the standard algorithm .

Future Works

Future Works

Hashing Technique

We suggest Rabin Karp algorithm which is popular hashing techniques can be used instead of AVL Tree .

The End