International Journal of Data Analytics
Volume 1 • Issue 1 • January-June 2020

SBASH Stack Based Allocation of Sheer Window Architecture for Real Time Stream Data Processing

Devesh Kumar Lal, SCSIT DAVV, Indore, India
Ugrasen Suman, SCSIT DAVV, Indore, India

ABSTRACT

The processing of real-time data streams is complex with large number of volume and variety. The volume and variety of data streams enhances a number of processing units to run in real time. The required number of processing units used for processing data streams are lowered by using a windowing mechanism. Therefore, the appropriate size of window selection is vital for stream data processing. The coarse size window will directly affect the overall processing time. On the other hand, a finely sized window has to deal with an increased number of management costs. In order to manage such streams of data, we have proposed a SBASH architecture, which can be helpful for determining a unipartite size of a sheer window. The sheer window reduces the overall latency of data stream processing by a certain extent. The time complexity to process such sheer window is equivalent to w log n w. These windows are allocated and retrieved in a stack-based manner, where stacks $\geq$ n, which is helpful in reducing the number of comparisons made during retrieval.

KEYWORDS

Big Data, Data Stream Processing, Real Time Processing, Stream Processing Engines

1. INTRODUCTION

The processing of real time streaming data in minimum latency is always an arduous job to handle. In every time interval of real time data processing, the rate of data generation whether it is structured, semi-structured, or unstructured is growing exponentially, which is directly proportional to an amount of memory required to store. Generation of unbounded streams of data may be processed by different techniques such as, interactive processing, real time processing, batch processing etc. These techniques are categorised on the basis of processing time, which is measured from the time of stream generation termed as event time. Among these techniques, real time processing provides results into minimum latency as measured from event time. Real time data stream processing depends on two major aspects; firstly, based on their availability and secondly, latency to process such data streams. The recent generated data streams consists higher value. The value can be measured from event of stream generation. In real time processing, value is always higher as compared to historical data processing. The value associated with data stream is reduces with enhancement in latency to process such data stream. Latency with respect to real time processing may have permissible factor to process data streams which comes under acceptable region, where latency constrains are application dependent. The extraction of inference from these massive unbounded datasets in real time with minimum latency requires parallelism mechanism.

DOI: 10.4018/IJDA.2020010101

Real time big data frameworks use parallelism approach for stream processing such as, Apache Spark, Storm, Flink, etc. Existing frameworks deal with most recent data streams by using techniques such as, batching, micro-batching, continuous flow etc. These approaches majorly use data flow approach, which allows high degree of parallelism. In dataflow programming, computation nodes are connected to each other, where a node is dependent on the outcomes from another node. Outcomes are propagated as soon as they are processed to the dependent nodes, triggering the computation among them (Sousa, 2012). Stream processing frameworks use the concept of operator, which applies over a fixed window size. Operator is a set of operations such as detection, correlation, aggregation, scoring etc., operate over a chunks of data streams. Where window is a fixed and finite size of data, acquired from unbounded streams by using any windowing mechanism such as sliding window, tumbling window, session window, landmark window etc. Size and organization of datasets of a window plays significant role in real time data stream processing. However, rate of change in latency to process real time streaming data is directly proportional to preferred window size. Parallelization approach manages reduction in latency by reducing window size. High degree of parallelism can minimize latency up to a certain extent. The further growth of data streams subsequently managed by applying reduction techniques such as, sketching, sampling, aggregation, etc.

We have introduced a sheer window approach, where unbounded data stream attains a sheer size of unipartite window. Sheer window supports to achieve the behaviour of sliding window and tumbling window at the same time. These techniques can be implemented with the help of stack based sheer window allocation. This allocation helps in reduction of processing cost by re-evaluating the same stream again and again. Our work aims to process data streams in minimum latency with the introduction of stack-based allocation of a sheer window (SBASH) architecture. SBASH architecture is comprised of five major components each consists of their individual functionalities. A sheer window approach is used for window generation, which enhances the performance as it accepts the window for sliding and tumbling at the same time with single computational cost. The multiple execution nodes accept sheer windows for processes from split node controllers. Stack based allocation of sheer window is introduced for faster retrieval and allocation. Such allocation inherits the principle of write once and read many.

The paper is organized as follows. Section 2 acquaints about related work on real time stream processing covering the existing stream processing engines with their limitation. Section 3 is dedicated to basic introduction of stream processing architecture and also explains the types of possible architecture. Section 4 explains SBASH architecture with its algorithms. Section 5 describes the experiment setup and implementation. The performed results and comparison are presented in section 6. Section 7 concludes the paper with future work.

2. RELATED WORK

A wide research has been performed on stream processing frameworks such as on windowing mechanism, levels of parallelism, task scheduling, multifarious operation, fault tolerance etc. We have discussed the recent research works performed on window selection and assignment to a particular node followed by the discussion of various types of stream processing engines.

Gabriele (2017) proposed agnostic and active model with count-based sliding window by using two worker nodes. They have used emitter-worker model, where emitter is responsible for data distribution, window triggering and window eviction policies. Chen (2016) uses the hashing for assigning proper node to a window. Bhatotia (2014) implemented a memorization-aware scheduler which decides the assignment of window in multiple slave nodes. Marcu (2017) implemented deduplication technique for real time data streams with the help of key-value approach to store the state. Andre (2018) implemented pilot streaming architecture which mainly focuses issues based on deploying and managing streaming frameworks.