Review paper: Reliability in elastic distributed stream processing systems

Reviewer: Ghazale Bakhtiari Azad*

Xiaohui Wei, Yuan Zhuang, Hongliang Li, and Zhiliang Liu.
Reliable stream data processing for elastic distributed stream processing systems.
Cluster Comput 23, 555–574 (2020). https://doi.org/10.1007/s10586-019-02939-9

Abstract -

According to the expansion of processing data streams and widespread use of distributed stream processing systems, possessing a fault-tolerant stream processing system to assure correct and complete results after processing incoming data is crucial. Many methods and mechanisms claim they guarantee reliability in these systems. So we investigate and evaluate a paper that proposes one of these mechanisms and exploits upstream backup to guarantee scale-out consistency and fast recovery with low latency in elastic distributed stream processing systems.

Keywords -

Distributed stream processing systems; Reliability; Review paper;

1 Introduction

Nowadays, the growth of real-time data streams and the necessity to process and analyze them have led to the widespread use of stream processing systems. Stream processing applications are usually data-sensitive and long-running for programs that process an infinite input stream, so they may unavoidably experience various failures. Any failure in a distributed stream processing system (DSPS) may lead to incomplete and inaccurate results. So reliability in these systems is crucial to be guaranteed.

As data streams have different arrival rates, elasticity should be one of the key features in DSPSs that dynamically adjust resources to stream workloads. Furthermore, elasticity raises new issues and challenges to the reliability of DSPSs that significantly increases the requirement of fault-tolerant methods. The article *'Reliable stream data processing for elastic distributed stream processing systems'* written by Xiaohui Wei, Yuan Zhuang, Hongliang Li, Zhiliang Liu from College of Computer Science and Technology, JilinUniversity, Changchun, China, proposes a novel fault-tolerant mechanism to make elastic DSPSs reliable which guarantees consistency under auto-scaling and supports fast recovery with low overhead.

[1] ghazalebakhtiari@comp.iust.ac.ir

The rest of the paper is organized as follows: In Sect. 2, we present a summary of what does the paper talk about; in Sect. 3, we critique the paper's strengths and weaknesses; in Sect. 4, we generally evaluate the paper. We conclude the paper in Sect. 5.

2 Paper summary

2.1 Overview

First of all, the paper determines reliability problems that elastic DSPS aspects, and after defining some assumptions to simplify the problem area, it proposes the intended fault-tolerant mechanism in detail. Further, the paper presents the protocols and algorithms. It is mentioned that the algorithm has been implemented on SPATE, an experimental stream processing system prototype built on an improved combination of the Hadoop version1.1.2 and S4.

2.2 Reliability problems of elastic DSPSs

As mentioned before, elasticity in DSPSs introduce new reliability challenges that are explained as follows [1]:

(i) Consistency guarantee under the auto-scaling: The upstream backups and their data dependencies between operators are not consistent before and after auto-scaling. So backup's re-partitioning and online adjustment of their data dependencies are needed.

(ii) Fast recovery with low overhead: To achieve recovery guarantee, upstream backup operates using checkpoints to restore the state of a failed task in stateful operators. Rollback recovery undoes any recent auto-scaling adjustment that may lead to a sudden and unpredictable workload and dampen system performance seriously.

So a fault-tolerance mechanism of dynamic adjustments to support elastic DSPSs is needed.

2.3 Reliability mechanism

The paper has defined the following assumptions to simplify the problem area:

- The execution plan of a stream-processing job is deployed over multiple nodes of a shared-nothing cluster
- One or more tasks of an operator are executed in parallel over partitions on individual nodes
- Single-node failure at one time
- The approach is to support deterministic operator

By the above assumptions reliability mechanism for EDSPSs has been proposed as below:

An upstream backup is defined to preserve operator running status, keeping recovery consistency under the auto-scaling while ensuring fast recovery. Upstream nodes act as backups for downstream neighbors by preserving the output events in an output queue called QO. These backups will be removed from upstream nodes when the downstream neighbors send ACK to the upstreams after completely processing them and sending them to the successors safely.

In upstream, backtracking data dependencies between the input and output streams are crucial for efficient recomputation-based recovery. Auto-scaling can change operator topology at runtime and break the mapping consistency between upstream backups and downstream neighbors. So the buffer state should be split or merged in response to topology changes. Upstream backup usually works with online checkpointing for stateful operators to achieve a recovery guarantee to restore a state of failed task from the last checkpoint. The checkpoint mechanism should keep track of the auto-scale changes of operator tasks.

So an upstream backup model coordinating with checkpointing is proposed to control the backup process and tolerate operator failure with a minimum storage.

2.4 Results

The authors have compared their proposed mechanism to similar methods, and their metrics are (i) the runtime overheads introduced by their fault tolerance and (ii) the effectiveness of their recovery. The charts indicate that the paper's mechanism has better performance than similar runtime overhead and recovery time methods.

3 Critique

Paper's concerns and the problem indicated is a hot-topic and important issue as of today's requirements in the large-scale data processing. Paper's novelty to introduce a low overhead fault-tolerant mechanism in elastic DSPSs is noticeable as it uses adaptive upstream backup instead of replication and elastic data slice. Presenting an asynchronous checkpoint mechanism to trace operator auto-scaling changes at runtime is also appreciable. In summary, the paper mainly presents two key contributions:

- The idea of using backup streams to provide a consistency guarantee
- The idea of using checkpointing to work with upstream backups at runtime

Some strength and weakness points of the paper are indicated below:

3.1 Major Comments

Besides including comprehensive and well-defined abstract and introduction, the paper has investigated all related works precisely and mentioned their strengths and weaknesses. But the paper lacks a plan for future works. As there are many assumptions to simplify the problem area, there could be more desire to pass the assumptions and make the idea suitable for realistic environments. In particular, when the article is about 'Distributed' stream processing systems, removing the assumption that a node can host one or multiple tasks that belong to the same operator seems to be necessary.

Although there are some states that authors have talked about them in section 1 and 2, Table 2 in section 3 contains their description, to raise clarity and better understanding, it's better to show the table sooner. One question left isn't mentioned in the paper: As the paper's explanations, when a failure happens, states and data-slices will recover from queue information. So if a Level-1 ACK drops, v_{i-1} won't trim the output queue, and upon failure, the operator may roll back to a very old state depending on when failure happens and this may raise latency overhead. I wonder if there could be any mechanism to avoid this problem.

It is discussed in [2] that checkpoints impose high runtime. By considering a wide range of DSPS applications prefer quick-and-dirty results rather than precise results [3, 4], authors of the intended paper [1] have proposed the use of approximate backup [5] and make a trade-off between performance and accuracy in ICCCN 2020. This is appreciable and demonstrates that authors have enough knowledge and ability to present new and novel mechanisms based on demand in this area.

3.2 Minor Comments

Along with the clarity and well-written text of the paper, resolving some minor issues will help strengthen the paper's text. The issues are mentioned as following:

- Some grammatical problems like article missing, single subject and plural verb
- In some parts, a hyphen is missed for the word fault-tolerant
- In figure 6, space is missed for T5: Level-1 ACK

4 Evaluation

After investigating strengths and weaknesses, according to contributions of paper and novelty to guarantee consistency and fast recovery after any failure during auto-scaling, and the strong evaluations presented, the paper mechanism is acceptable. It can be inferred as a reference to improve reliability in elastic DSPSs and going through evolution of intended mechanism to pass the assumptions and make it suitable for real environments.

5 Conclusion

This review paper evaluated the article *'Reliable stream data processing for elastic distributed stream processing systems'* [1] which proposes a novel upstream backup with checkpointing to guarantee consistency and fast recovery with low overhead when failure happens in elastic DSPSs. we discussed summary of paper and Critiqued strengths and weaknesses of it. The paper makes a good effort in improving reliability in elastic DSPSs. Having a better future plan, removing some important assumptions, and fixing some grammatical problems, can make the article more credible.

References

[1] Xiaohui Wei, Yuan Zhuang, Hongliang Li, and Zhiliang Liu. Reliable stream data processing for elastic distributed stream processing systems. *Cluster Computing*, 23:555–574, 2020. doi:https://doi.org/10.1007/s10586-019-02939-9.

[2] Yu Gu, Zhe Zhang, Fan Ye, and et al. An empirical study of high availability in stream processing systems. In *proceedings of the 10th ACM/IFIP/USENIX International Conference on Middlewar*, page 23, Springer-Verlag New York, 2009.

[3] D Takao, K Sugiura, and Y Ishikawa. Approximate fault tolerance for sensor stream processing. In *proceedings of Australasian Database Conference*, pages 55–67, Springer, 2020.

[4] Z Cheng, Q Huang, and Patrick PC Lee. On the performance and convergence of distributed stream processing via approximate fault tolerance. *The VLDB Journal*, 28:821–846, 2019. doi:https://doi.org/10.1007/s00778-019-00565-w.

[5] Yuan Zhuang, Xiaohui Wei, Hongliang Li, Mingkai Hou, and Yundi Wang. Approximate fault tolerance for sensor stream processing. In *proceedings of 29th International Conference on Computer Communications and Networks (ICCCN)*, page 1, 10.1109/ICCCN49398.2020.9209717, 2020.