CS5052 – Data Intensive Systems

220031985

University of
St Andrews

23rd March 2023

"Twitter Heron: Stream Processing at Scale"

Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel*,1, Karthik Ramasamy, Siddarth Taneja

INTRODUCTION :

Twitter's quantity of data processed in real-time has grown along with its *diversity* and *use cases*. The limitations of Storm, which had long been *Twitter's primary platform for real-time data*, became evident.

Storm was getting harder and harder to use at its current scale due to issues with **scalability, debug-ability, management, and efficient cluster resource sharing** with other data services. There were several debuggability-related challenges when using Storm.

Simply put at the scale Twitter was operating, any improvement in performance translates into a significant reduction in infrastructure costs and also into a significant increase in the productivity of our end users. Therefore, the authors needed a system that *scales better*, has *better debug-ability*, *has better performance*, and is *easier to manage* – all while working in a shared cluster infrastructure.

The authors present the design and implementation of a new real-time design stream data processing system which meets these needs, called Heron. It is now the *de facto stream of data processing engine* inside Twitter and in the paper, the authors share their experiences running Heron in production.

BACKGROUND :

Storm relied on a centralized master node to coordinate the distribution of computation tasks, which became a bottleneck as the system grew. Additionally, Storm's approach to fault tolerance involved **restarting failed tasks**, which resulted in significant latency spikes.

As an alternative to Storm, Twitter developed Heron to tackle these issues. Heron's distributed architecture *eliminates* away with the requirement for a centralised master node, making it highly scalable and fault tolerant.

Heron also incorporates several performance-enhancing enhancements, such as a new data model and more effective resource management. Storm customers can easily switch to Heron because it is **API-compatible** with Storm.

Heron's architecture is based on distributed graph processing. Heron uses a *directed acyclic graph (DAG)* to describe data flow, with the vertices being processing tasks and the edges denoting data flow between jobs. Heron can manage both batch and stream processing workloads since the DAG is dynamically built based on the incoming data.

The usage of a pluggable scheduler is one of the notable features of Heron architecture. Depending on the workload and resource availability, the scheduler can utilise several scheduling algorithms to distribute computing tasks across the cluster.

A *round-robin scheduler* and a *locality-aware scheduler*, that aims to reduce data movement between nodes, are just two of the **built-in schedulers** that Heron supports.

Heron also **has several optimizations** to boost resource usage. For instance, the pre-emptive scheduling method used by Heron enables it to effectively handle burst workloads without wasting resources.

To grow to tens of thousands of processing jobs per node, Heron also makes advantage of a lightweight process model.

NOVELTY:

The paper proposes a new solution to the problem of efficiently processing large-scale data streams in real time. Heron is a **major shift** from earlier stream processing systems like Apache Storm, making the work discussed in the paper particularly novel.

The system's *new features* include a distributed graph processing paradigm, a pluggable scheduler, shared memory communication, pre-emptive scheduling, and fault tolerance optimizations.

Several big companies, including Twitter, Yahoo, and Airbnb, have adopted Heron. These organisations employ Heron to process massive data streams in real-time, giving them the ability to act decisively based on recent information.

EVALUATION:

According to the authors, Heron has outperformed Apache Storm in terms of performance, processing up to **14 million messages per second on a 1000-node cluster**, or nearly 10 times as many messages as Strom on the same hardware. Additionally, it has been shown to have lower end-to-end latency than Storm, **averaging 1.5 seconds as opposed to Storm's 2.3 seconds.**

The evaluation, however, might be more thorough because it mostly concentrates on synthetic workloads and might not accurately represent the system's performance in real-world scenarios.

Although the proposed solution offers several advantages over Storm, including higher performance, new stream processing systems have likely been developed since this paper's publication that offers performance that's on par with or even better.

CONCLUSION :

To conclude, the paper "*Twitter Heron: Stream Processing at Scale*" presents a highly novel and innovative solution to the problem of efficiently processing large-scale data streams in real time. The authors propose Heron, a stream processing engine designed to handle large-scale data processing workloads with improved performance over Apache Storm.

A distributed graph processing architecture, a pluggable scheduler, shared memory connectivity, pre-emptive scheduling, and fault tolerance optimizations are just a few of the characteristics the system features.

The authors **present solid evidence** that Heron outperforms prior stream processing systems like Apache Storm, even though the system's evaluation could be more rigorous. Heron has been implemented by several significant organisations, demonstrating its *viability* in real-life scenarios.

QUESTIONS FOR THE AUTHORS :

- Synthetic workloads are the main focus of the system evaluation. Have any studies been done to assess the system's performance in actual-world scenarios?
- Are there any restrictions on the system's ability to scale, and if so, how do you intend to fix them in upcoming iterations of Heron?
- How does Heron stack up against other stream processing systems developed since this paper's publication?
- Is there any intention to include machine learning features in Heron, such as the capacity to develop and deploy models in real time?

TOTAL WORDS: 902