Project Proposal: Exploring Parameter Behavior in Apache Flink for Optimized Stream Processing

Group Members:
- Ojasvi Bansal (2021CS50600)
- Esha Patel (2021CS10566)
- Aaveg Jain (2021CS10073)

1. Which systems are you picking? Why are they competitors?

We are focusing solely on Apache Flink for this benchmarking study. Flink is a leading stream processing framework renowned for its robust handling of stateful stream processing and complex event processing. The primary objective of our project is to explore the behavior of various parameters within Flink, such as checkpointing frequency, state backend choice, and windowing strategies.

While we are not comparing Flink with another system in this study, our goal is to analyze how different configurations affect Flink's performance metrics, including latency, throughput, and fault tolerance. Understanding these parameters is crucial for optimizing Flink for various real-world streaming applications.

2. What difference do you expect to see between the configurations? Why does this difference matter in the real world?

We expect to see significant differences in performance metrics based on the following parameters:

- Checkpointing Frequency: Varying the frequency of checkpoints may lead to different latency and throughput outcomes. Frequent checkpoints can introduce overhead but enhance fault tolerance, while infrequent ones might lead to data loss in the event of a failure. Understanding this trade-off is vital for applications that require high availability and low latency.

- State Backend Choices: Flink supports several state backends (e.g., MemoryStateBackend, FsStateBackend, RocksDBStateBackend). We anticipate that different backends will impact memory usage, recovery time, and processing speed. This knowledge is critical for applications that manage large states, such as session management or event pattern detection.

- Windowing and Triggering Strategies: Different windowing techniques (e.g., tumbling, sliding, session windows) and triggering strategies can impact event processing latency and accuracy. We expect to observe how these strategies affect performance and how they align with business requirements for real-time analytics.

- Parallelism: The degree of parallelism can significantly influence throughput and resource utilization. We anticipate that varying task parallelism will impact performance metrics differently

for workloads that are either CPU-bound or I/O-bound. Understanding this behavior will help optimize Flink configurations for different streaming applications.

These differences matter in real-world applications because they guide developers in making informed decisions on how to configure Flink to meet specific performance requirements, enabling efficient processing of real-time data streams.

3. What benchmarks will you run? How will these benchmarks quantify these differences?

We will implement the following benchmarks to assess the performance of various configurations in Apache Flink:

- Throughput Measurement: Measure the number of events processed per second under different checkpointing frequencies and state backends.

- Event Processing Latency: Measure the time taken to process events from input to output under various configurations. This will be quantified by tracking the time between when an event is produced and when it is consumed after processing. We will analyze how varying checkpointing intervals and windowing strategies affect latency.

- State Recovery Time: Introduce controlled failures during the execution of streaming jobs and measure the time taken by Flink to recover and resume processing. This benchmark will help evaluate the impact of different state backends on recovery time and overall resilience.

- Resource Usage: Monitor CPU and memory consumption during benchmark execution to understand how different configurations affect resource utilization. Metrics will be collected to identify efficient configurations for specific workloads.

- Backpressure Analysis: Evaluate Flink's behavior under varying loads to identify when backpressure occurs and how effectively the system manages it. We will measure the response time of the system under increasing event rates and analyze how task parallelism and state management strategies impact backpressure handling.

These benchmarks will allow us to quantify the trade-offs in performance, reliability, and resource consumption resulting from different parameter configurations in Apache Flink. The findings will provide valuable insights into optimizing Flink for specific streaming applications, enhancing both performance and resource efficiency.