**Machine Programming 4 – Distributed Stream Processing System**

**Yunsheng Wei (wei29) Neha Chaube (nchaub2)**

**Design:**

In MP4, we built a distributed stream processing system called Crane. We will briefly describe some of the key designs in the following sections:

1. The system is composed of two components: master node called Nimbus, and slave nodes called supervisors. Nimbus is responsible for receiving jobs from clients, and assigning tasks to supervisors. Once being assigned tasks by Nimbus, supervisors will respond by starting separate threads for every task.
2. Nimbus maintains a map between supervisors and their assigned tasks. Once Nimbus detect a supervisor’s failure, it will attempt to assign tasks on the failed supervisor to other live supervisors.
3. We use the Gossip group membership service in MP2 for failure detection, and use Observer pattern for the communication between group membership service and Nimbus.
4. In order to ensure the at least once semantics for tuple, we adopt the ack mechanism like the one adopted in Apache Storm. We have a separate acker thread listening for ack message, and every time a tuple is sent from spout to bolt, or bolt to bolt, it will also send an ack message to acker. When the acker knows a tuple has been finished, it will notify spout.
5. The communication between Nimbus and supervisors, between clients and Nimbus are using Java RMI. The communication between bolts and bolts, bolts and ackers are using UDP. And in order to reduce packet loss, we allow clients to specify the rate for spout and bolts to emit tuples.
6. Our implementation supports parallelism for bolts. Also in our implementation, we assume Nimbus never dies, and it can support arbitrary failures of supervisors.
7. The whole project is implemented in Java.

**Applications:**

We developed three applications for both Crane and Storm. The applications are based on the Tweet Trends dataset used in the paper [Real-Time Classification of Twitter Trends](used%20in%20the%20paper%20Real-Time%20Classification%20of%20Twitter%20Trends). The three applications are:

1. Find all users who tweet about a trending topic which belongs to the class “ongoing-event”. The topology is: Spout -> Filter bolt -> Join bolt -> Sink bolt

From the plot, we can see. When use Crane, when use Storm.

1. Count the number of tweets for trending topic class “meme”, “onging-event”, and “news”. The topology is: Spout -> Filter bolt -> Transform bolt -> Sink bolt

From the plot, we can conclude. When use Crane, when use Storm.

1. Filter all tweet ids for trending topic class “meme”, “ongoing-event”, and “news”. The topology is: Spout -> Filter bolt -> Sink bolt

From the above plot, we can see. When use Crane, when use Storm.

**Comparison- Storm vs Crane**

1. Using Transform

**Comparison- Storm vs Crane**