Contributions

We intend to contribute to the open source community by developing features for a real time data processing framework called Storm. These features have been raised by the Storm developer community. Details of the features are as follows:

1) Making spouts their own ackers:

The current implementation requires the number of ackers to be specified and internally, spout sends init message to the acker while it responds by sending the ack/fail message to the spout. Also, with the current implementation the birthday problem takes effect and occasionally two different spout tuples might have same id, causing them both to timeout and fail.

Making spouts their own ackers would resolve all of the issues mentioned above. The code given below shows the snippet of the implementation of acking system.

```clojure
(defn update-ack [curr-entry val]
  (let [old (get curr-entry :val 0)]
    (assoc curr-entry :val (bit-xor old val))
  )
)
```

In short, the system relies on the fact that a bitwise-xor applied to any list where every number appears twice (or an even number of times) always produces 0. In this way the acker can know when the tuple has been completed. A mistake could be made if the random 64-bit number causes the xor to produce 0 before the tree is completed. We are trying to accommodate this logic within the spout.

2) UI Visualizations

One of the feature that Storm community is looking forward to is the UI visualizations that would provide a visual indicator of how topologies are behaving. We plan to represent the topology and its components as a force directed graph where the size of the edges between nodes would indicate the amount of throughput passing between them. Hovering over a topology would give more detailed information about it such as total number of nodes, a visualization of incoming and outgoing throughput to each individual task in that component etc. Also if time permits we would like to indicate bottlenecks using different colors. All this information would be useful while debugging an issue in production.

As existing visualization tools have certain limitations we will utilize the D3.js library in our solution as it supports a vast array of visualization types. Since the library sits right between data and graphics, it is located in the ideal place for data visualization.
A sample visualization is given below without any data.

**Experimental Evaluation**

Enhancement 1 - Making spouts their own ackers

- Do spouts generate their own tuple ids?
- Does this improve performance by not having to send ack/init from spout to acker and ack/fail from acker to spout?

Enhancement 2 - UI Visualizations

- Is a working program implemented to quickly depict the topologies using a force-directed graph?
- Does the size of the edges between nodes indicate the amount of throughput passing between them?
- Is it possible to use colors to indicate which components are likely bottlenecks in the topology?
- Does clicking on a node show more detailed information such as throughput and latency?
Although the above questions cover significant success criteria, we will further develop a more extensive evaluation of our enhancements.

**Timeline**

We are performing analysis on both of the features listed and are working on them simultaneously. This is taking longer than expected as a result of limited documentation. In our opinion we should be done with the basic implementation of the real time visualization and the enhancement related to spouts.

**Major Difficulties**

- None of us had experience with Clojure. We had to gain fairly good understanding about Clojure before we could start the implementation
- Since, Apache Storm is a relatively new project, very limited documentation is available
- Implementing data visualization in real time.
- Performance can be an issue since the number of topologies can be time-varying
- Condensing topologies in an intuitive manner within a force-directed graph