Storm @Twitter

Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M. Patel*, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, Nikunj Bhagat, Sailesh Mittal, Dmitriy Ryaboy

Paper Summary by Group 5:
Alok Thatikunta (111498776), Rahul Sihag (111462160), Sweta Kumari (111497926)
{athatikunta, rsihag, swkumari} @cs.stonybrook.edu

Storm is a real-time data process system aimed to handle large-scale streaming data. It offers relatively strong fault-tolerant feature as well as two processing semantic guarantees.

The paper describes the system architecture of Storm. In Storm cluster, we run topologies, which keeps on processing live data forever from different data sources. Topologies are basically connected components of *spouts* and *bolts*. Spouts are tuple sources for the topology. Typical spouts pull data from queues, such as *Kafka* or *Kestrel*. On the other hand, bolts process the incoming tuples and pass them to the next set of bolts downstream. When a topology is submitted to a Storm cluster by client, *Nimbus* service on master node consults the supervisor services on different worker nodes and submits the topology. The cluster state is maintained by *Zookeeper* Each *supervisor*, creates one or more worker processes, each having its own separate jvm . Each process runs within itself threads which is called *Executors*. The thread/executor processes the computational tasks : Spout or Bolt. The actual work is done at worker nodes. Data is shuffled from a producer spout/bolt to a consumer bolt using grouping techniques like Shuffle, FIeld, All, Local or Global grouping. Tasks provide intra-bolt/intra-spout parallelism, and the executors provide intra-topology parallelism. At Twitter, the topologies are generated using Summingbird, a general stream processing abstraction. One of the key characteristics of Storm is its ability to provide guarantees about the data that it processes. It provides two types of semantic guarantees – "*at least once,*" and "*at most once*" *semantics*. At least once semantics guarantees that each tuple that is input to the topology will be processed at least once. With at most once semantics, each tuple is either processed once, or dropped in the case of a failure.

All coordination between Nimbus and the Supervisors is done using Zookeeper. Furthermore, Nimbus and the Supervisor daemons are fail-fast and stateless, and all their state is kept in Zookeeper or on the local disk(s). This design is the key to Storm's resilience. If the Nimbus service fails, then the workers still continue to make forward progress. In addition, the Supervisors restart the workers if they fail. However, if Nimbus is down, then users cannot submit new topologies.

Storm currently runs on hundreds of servers (spread across multiple datacenters) at Twitter. Storm operations are continuously displayed using a rich visualization developed in-house, which is critical in assisting with identifying and resolving issues that have caused alarms to be triggered. The metrics is broadly - system metrics and topology metrics. The paper also discuss the operational stories of Storm at Twitter. After analyzing various configurations, they changed the KafkaSpout code to write its state to a key-value store. They also changed the Storm core to write its heartbeat state to a custom storage system designed specifically for the purpose of storing the Storm heartbeat data. The heartbeat daemon cluster is designed to trade off read consistency in favor of high availability and high write performance. They are horizontally scalable to match the load that is placed on them by the workers running in the Storm core. Storm topologies have a *max spout pending* parameter which puts a limit on how many tuples can be in flight, i.e. have not yet been acked or failed, in a Storm topology at any point of time. At Twitter, they have implemented a auto-tuning algorithm for this value to cater their need of different max spout pending for different topologies. The algorithm adjusts the value periodically to achieve maximum throughput in the topology. And the empirical evaluations and results sums up the fact that Storm is a critical infrastructure at Twitter that powers many of the real-time data driven decisions.