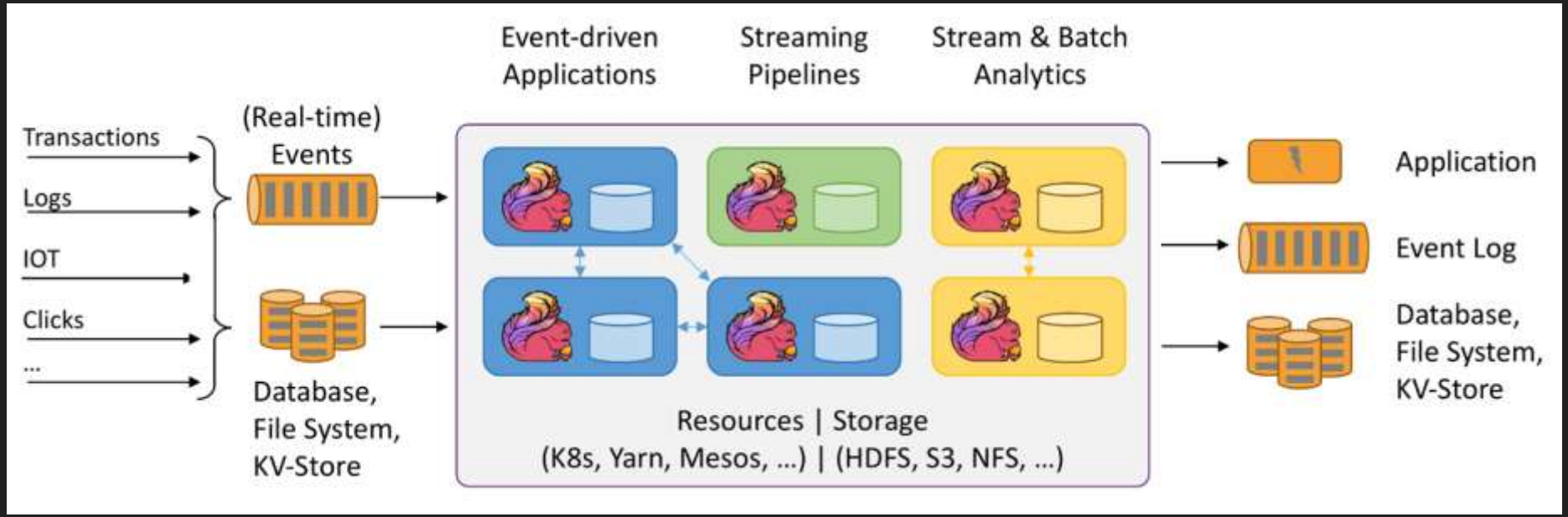
Flink

Background

Flink is what is known as a stream processing framework. It is not a message broker, but instead rather software run on a cluster of nodes used as consumers of some message broker.

Flink is useful as it allows distributed processing of a message queue (in real time as opposed to micro batching), while also persisting state about the seen messages amongst the nodes in a cluster in order to make meaningful aggregations or joins on incoming data.

Use Case Visualized



State in Stream Processing

Why is it so important to keep state if each event should be processed independently?

- Grouping events by time
- Holding a local copy of a database table for stream enrichment
  - Table updated over time by ingesting changes from a stream
- Tinder use case example
  - Every like placed into a stream, Flink processes likes and stores them in state, if a like comes in with the opposite directional like already in state, we have a match
  - Flink forwards match data into a second stream

Fault Tolerance

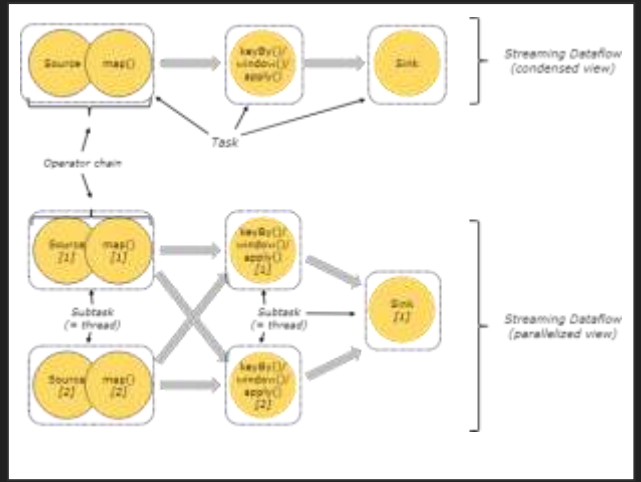
Each node has its own built up local state that it computes based on the messages that it uniquely sees (messages are hashed such that they only go to one node in the cluster, it is the developer's job to make sure they are partitioned correctly).

So each node has its own local state, and we want to make sure that if one of the nodes crashes, we can recover its local state as well as process any messages that it had yet to process on a new node. It does so by occasionally storing checkpoints of node state to durable storage such as HDFS or S3!

Flink Architecture

Obviously, in order to increase performance, Flink tries to parallelize computation as much as possible.

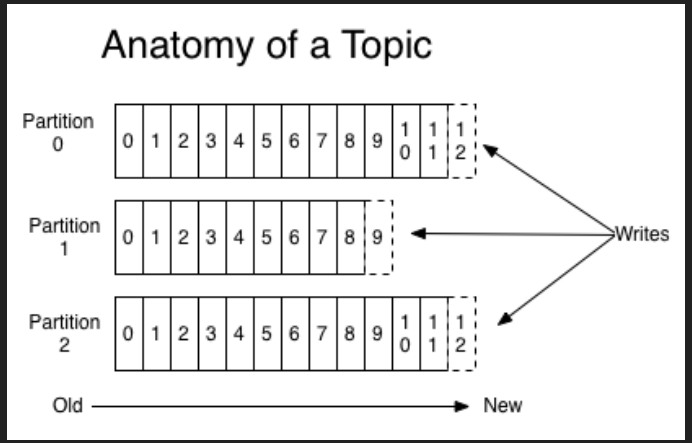
- Single Job Manager node
  - Responsible for parallelizing a given job into work for each Task Manager using dataflow graph
  - Responsible for initiating checkpoints, and storing where they are located in distributed file system
  - Can achieve high availability on job manager by using a coordination service for keeping track of which nodes are up and which are not, as well as status of which checkpoints have worked



Log Based Message Brokers

Recall: Log Based Message Brokers are partitioned and persistent message queues that keep track of the offset of the last message that a consumer node has seen. Messages are delivered in order to a single consumer.

Flink uses log based message brokers such as kafka as its data source.



Checkpointing

- Job Manager occasionally sends “checkpoint barrier” message to all nodes in cluster
  - Checkpoint barrier delivered in order to all consumers, since it is treated like any other message in the log based message queue (which delivers messages in order)
  - Upon processing checkpoint barrier consumers note their current offset in the log based message broker, and persist this along with their local state to distributed storage
    - Using Chandy Lamport distributed snapshots (consistent with causality)
- If a node crashes, restore all nodes to a checkpointed state, and start processing messages from corresponding offsets
  - May result in certain messages being processed more than once!
- Snapshots created in the background (non-blocking), considered successful if all nodes report successfully storing local snapshot to job manager

Changing Cluster Size

If we want to change the size of our cluster, we have to redistribute state over the nodes. Similarly to consistent hashing, we want to ensure that as little state as possible is redistributed over the network while ensuring that each node processes a similar amount. To do this, Flink has made an adjustment such that checkpoints are just lists of keys and values (as opposed to individual objects representing the state on each node), so that this way they can be easily re-partitioned using some sort of range schema.

Conclusion

Unlike a Spark Streaming, which processes messages in microbatches, Flink is a fully real time stream processing engine that provides the ability to make stateful computations with high availability. In comparison to other stream processing engines like Apache Storm or Samza, Flink makes life easier for the developer by automatically calculating the dataflow graph, and parallelizing work for you.