4. Real-time Streaming Computing Models: Apache Storm and Twitter Heron

Apache Storm Model

FAQs

- HITS Algorithm
- How do we limit the base-set?

This material is built based on

- Storm programming guide
  - http://storm.apache.org/releases/2.0.0-SNAPSHOT/index.html

Storm Model

- One-at-a-time stream processing
- Represents the entire stream processing pipeline as a graph of computation called a topology
  - A single program is deployed across a cluster
  - A stream is represented an infinite sequence of tuples
    - A tuple: a named list of values
Why use Storm?

- Distributed real-time computation system
- Real-time analytics
- Online machine learning
- Continuous computation
- Distributed RPC
- ETL

Use Case

- Twitter
- Amazon Kinesis and Storm
  - KinesisSpoutConfig
  - KinesisSpout

Spout in the Storm model

- Spout
  - A source of streams in a topology
  - A spout can read from a Kealet or Kafka queue
  - Turns the data into a tuple stream
  - Timer spout could emit a tuple into its output stream every 10 seconds

Bolt in the Storm model

Bolt

- Performs actions on streams
- Takes any number of streams as input and produces any number of streams as output
- Runs functions, filters data, computes aggregations, does streaming joins, updates database, etc.

Topology in the Storm model

- Topology
  - A network of spouts and bolts with each edge representing a bolt that processes the output stream of another spout or bolt
- Task
  - Each instance of a spout or bolt

Word count topology: Sentence Spout

Sentence Spout

- Emits a stream of single-value tuples continuously with the key name "sentence" and a string value ("sentence":"my dog has fleas")
**Word count topology: Split Sentence**

- Split Sentence Bolt
  - Subscribes to the sentence spout’s tuple stream
    - \{"word":"my"
    - \{"word":"dog"
    - \{"word":"has"
    - \{"word":"fleas"

**Word count topology: Word Count**

- Word count bolt
  - Subscribes to the output of the SplitSentenceBolt class
  - Keeps a count of how many times it has seen a particular word
  - Whenever it receives a tuple, it will increment the counter and emit
    - \{"word":"dog", "count":5

**SentenceSpout.java**

```java
public class SentenceSpout extends BaseRichSpout {
    private SpoutOutputCollector collector;
    private String[] sentences = {
        "my dog has fleas",
        "i like cold beverages",
        "the dog ate my homework",
        "don't have a truck",
        "i don't think i like fleas"
    };
    private int index = 0;
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("sentence"));
    }
    public void open(Map config, TopologyContext context, SpoutOutputCollector collector) {
        this.collector = collector;
    }
    public void nextTuple() {
        this.collector.emit(new Values(sentences[index]));
        index ++;
        if (index >= sentences.length) {
            index = 0;
        }
        Utils.sleepInMilli();
    }
}
```

**SplitSentenceBolt.java**

```java
public class SplitSentenceBolt extends BaseRichBolt {
    private OutputCollector collector;
    public void prepare(Map config, TopologyContext context, OutputCollector collector) {
        this.collector = collector;
    }
    public void execute(Tuple tuple) {
        String sentence = tuple.getStringByField("sentence");
        String[] words = sentence.split(" ");
        for(String word : words) {
            this.collector.emit(new Values(word));
        }
    }
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("word"));
    }
}
```

**SentenceSpout.java: Continued**

```java
public void open(Map config, TopologyContext context, SpoutOutputCollector collector) {
    this.collector = collector;
}
public void nextTuple() {
    this.collector.emit(new Values(sentences[index]));
    index ++;
    if (index == sentences.length) {
        index = 0;
    }
    Utils.sleepInMilli();
}
```

**Spring 2019 Colorado State University, page 3**
What if we have a large number of "sentence"s?

- How many "Sentence Spout" can you have during the computation?
- How many "Split Sentence Bolt" can you have during the computation?
- How many "Word Count Bolt" can you have during the computation?
- How many "Report Bolt" can you have during the computation?

http://www.cs.colostate.edu/~cs535
4. Real-time Streaming Computing Models: Apache Storm and Twitter Heron

Parallelism in Storm

Components of the Storm cluster

- **Nodes** (machines)
  - Executes portions of a topology
- **Workers** (JVMs)
  - Independent JVM processes running on a node
  - Each node is configured to run one or more workers
  - A topology may request one or more workers to be assigned to it
- **Executors** (threads)
  - Java threads running within a worker JVM process
  - Multiple tasks can be assigned to a single executor
  - Unless explicitly overridden, Storm will assign one task to each executor
- **Tasks** (bolt/ spout instances)
  - Instances of spouts and bolts whose nextTuple() and execute() methods are called by executor threads

Parallelism in the WordCount topology

- In our example, we have NOT used any of Storm's parallelism
  - Default setting is a factor of one
  - Topology execution flow

Adding workers to a topology

- Through configuration
- Through APIs
  - Passing Config object to the submitTopology() method

- Bolts and spouts do not have to change

Adding executors and tasks

- Specify the number of executors when defining a stream grouping

```java
Config config = new Config();
config.setNumWorkers(2);
```

Two spout tasks (if we are using one worker)
In `SplitSentenceBolt` and `WordCountBolt`,

- Set up the split sentence bolt to execute as 4 tasks and 2 executors
  - Parallelism hint
  - Storm will run 2 tasks per executor (thread)
  - Each executor thread will be assigned two tasks to execute

```
builder.setBolt("SPLIT_BOLT_ID", splitBolt, 2)
  .setNumTasks(4)
  .shuffleGrouping("SENTENCE_SPOUT_ID");
```

- How many workers will work for this example?
  - Answer: 2 workers (JVMs)
  - 2 executors per worker

In `SplitSentenceBolt` and `WordCountBolt`,

- Set up the split sentence bolt to execute as 4 tasks and 2 executors
  - Parallelism hint
  - Storm will run 2 tasks per executor (thread)
  - Each executor thread will be assigned two tasks to execute

```
builder.setBolt("SPLIT_BOLT_ID", splitBolt, 2)
  .setNumTasks(4)
  .shuffleGrouping("SENTENCE_SPOUT_ID");
```

- How many workers will work for this example?
  - Answer: 2 workers (JVMs)
  - 2 executors per worker

- Set up the `WordCountBolt` to execute as 4 tasks each with its own executor thread

```
builder.setBolt("COUNT_BOLT_ID", countBolt, 4)
  .fieldsGrouping("SPLIT_BOLT_ID", new Fields("word"));
```

- How many workers will work for this example?
  - 1 workers (JVMs)
  - 4 executors per worker

http://www.cs.colostate.edu/~cs535
What will be the results with given parallelism?

--- FINAL COUNTS ---
- a : 1426
- ate : 1426
- beverages : 1426
- cold : 1426
- cow : 1426
- dog : 2852
- don't : 2851
- fleas : 2851
- has : 1426
- have : 1426
- homework : 1426
- i : 4276
- like : 2851
- man : 1426
- my : 2852
- the : 1426
- think : 1425

-------------------

--- FINAL COUNTS ---
- a : 2726
- ate : 2722
- beverages : 2723
- cold : 2723
- cow : 2726
- dog : 5445
- don't : 5444
- fleas : 5441
- has : 2722
- have : 2722
- homework : 2722
- i : 8175
- like : 5449
- man : 2722
- my : 5445
- the : 2727
- think : 2722

--------------

Increased counts

Spout emits data indefinitely
- Stops when the topology is killed

Having multiple workers has no effect when running a topology in local mode
- Only task and executor parallelism settings have effect
- A topology running in local mode always runs within a single JVM process
- Use your application in a cluster for true parallelism

4. Real-time Streaming Computing Models:
   Apache Storm and Twitter Heron

Stream Groupings

Seven built-in stream groupings (1/3)
- **Shuffle grouping**
  - Randomly distributes tuples across the target bolt’s tasks

**Fields grouping**
- Routes tuples to bolt tasks based on the values of the fields specified in the grouping
  - Grouped on the “word” field
    - Tuples with the same value for the “word” field will always be routed to the same bolt task

- **All grouping**
  - Replicates the tuple stream across all bolt tasks

Seven built-in stream groupings (2/3)
- **Global grouping**
  - Routes all tuples in a stream to a single task
  - Chooses the task with the lowest task ID value

- **None grouping**
  - Functionally equivalent to the shuffle grouping
  - Reserved for future use

- **Direct grouping**
  - The source stream decides which component will receive a given tuple
    - Only for streams that have been declared as direct streams

http://www.cs.colostate.edu/~cs535
Seven built-in stream groupings (3/3)

- Local or shuffle grouping
  - Shuffles tuples among bolt tasks running in the same worker process, if any
  - Otherwise, performs shuffle grouping
  - Depending on the parallelism of a topology, the local or shuffle grouping can increase topology performance by limiting network transfer

Custom Grouping Stream

```java
public interface CustomStreamGrouping extends Serializable {
    void prepare(WorkerTopologyContext context, GlobalStreamId stream, List<Integer> targetTasks);
    List<Integer> chooseTasks(int taskId, List<Object> values);
}
```

Example of grouping (1/2)

```java
public void nextTuple() {
    if(index < sentences.length) {
        this.collector.emit(new Values(sentences[index]));
        index ++;
    }
    Utils.waitForMillis(1);
}
```

Example of grouping (2/2)

- Now change the grouping on the CountBolt parameter to a shuffle grouping and rerun the topology:

```
Builder.setBolt(COUNT_BOLT_ID, countBOLT, 4).shuffleGrouping(SPLIT_BOLT_ID);
```

Why?

- The CountBolt parameter is stateful
  - It maintains a count for each word it’s seen