PART 2.
SCALABLE FRAMEWORKS FOR REAL-TIME BIG DATA ANALYTICS

1. SPEED LAYER: APACHE STORM

Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs535

Today’s topics
- Parallelism in Storm
- Programming Assignment 2
- Lossy Counting Algorithm
- Installing Storm cluster

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

FAQs
- Term project proposal
- Feedback for the most of submissions are available
- PA2 has been posted (11/6)
Adding workers to a topology

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

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

Bolts and spouts do not have to change

Adding executors and tasks

- Specify the number of executors when defining a stream grouping
  - builder.setSpout(SENTENCE_SPOUT_ID, spout, 2);
  - Assigns two tasks and each task is assigned its own executor thread

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
  - Each executor thread will be assigned two tasks to execute
  - builder.setBolt(SPLIT_BOLT_ID, splitBolt, 2)

- Set up the Word count bolt to execute as 4 tasks each with its own executor thread
  - builder.setBolt(COUNT_BOLT_ID, countBolt, 4)

What will be the results with given parallelism?

--- FINAL COUNTS ---

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1426</td>
</tr>
<tr>
<td>ate</td>
<td>1426</td>
</tr>
<tr>
<td>beverages</td>
<td>1426</td>
</tr>
<tr>
<td>cold</td>
<td>1426</td>
</tr>
<tr>
<td>cow</td>
<td>1426</td>
</tr>
<tr>
<td>dog</td>
<td>2852</td>
</tr>
<tr>
<td>don't</td>
<td>2851</td>
</tr>
<tr>
<td>fleas</td>
<td>2851</td>
</tr>
<tr>
<td>has</td>
<td>1426</td>
</tr>
<tr>
<td>have</td>
<td>1426</td>
</tr>
<tr>
<td>homework</td>
<td>1426</td>
</tr>
<tr>
<td>i</td>
<td>4276</td>
</tr>
<tr>
<td>like</td>
<td>2851</td>
</tr>
<tr>
<td>man</td>
<td>1426</td>
</tr>
<tr>
<td>my</td>
<td>2852</td>
</tr>
<tr>
<td>the</td>
<td>1426</td>
</tr>
<tr>
<td>think</td>
<td>1425</td>
</tr>
</tbody>
</table>

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

--- FINAL COUNTS ---

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2726</td>
</tr>
<tr>
<td>ate</td>
<td>2722</td>
</tr>
<tr>
<td>beverages</td>
<td>2723</td>
</tr>
<tr>
<td>cold</td>
<td>2723</td>
</tr>
<tr>
<td>cow</td>
<td>2726</td>
</tr>
<tr>
<td>dog</td>
<td>5445</td>
</tr>
<tr>
<td>don't</td>
<td>5444</td>
</tr>
<tr>
<td>fleas</td>
<td>5451</td>
</tr>
<tr>
<td>has</td>
<td>2723</td>
</tr>
<tr>
<td>have</td>
<td>2722</td>
</tr>
<tr>
<td>homework</td>
<td>2722</td>
</tr>
<tr>
<td>i</td>
<td>8175</td>
</tr>
<tr>
<td>like</td>
<td>5449</td>
</tr>
<tr>
<td>man</td>
<td>2722</td>
</tr>
<tr>
<td>my</td>
<td>5445</td>
</tr>
<tr>
<td>the</td>
<td>2727</td>
</tr>
<tr>
<td>think</td>
<td>2722</td>
</tr>
</tbody>
</table>

Increased counts

With two workers
- 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

Stream groupings
- How a stream’s tuples are distributed among bolt tasks in a topology
  - E.g.: SplitSentenceBolt class was assigned four tasks in the topology
  - Which tuples will be processed in which task?
  - The stream grouping determines which one of those tasks will receive a given tuple

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
  - By calling the emitDirect() method
  - Only for streams that have been declared as direct streams
Custom Grouping Stream

```java
custom interface CustomStreamGrouping extends Serializable {
    void prepare(WorkerTopologyContext context,
                 GlobalStreamId stream, List<Integer> targetTasks)
    {
        List<Integer> chooseTasks = new ArrayList<>();
        values.forEach(x -> chooseTasks.add(x));
    }

    List<Integer> chooseTasks(int taskId, List<Object> values);
```
Example ($\varepsilon = 0.2$, $w = 1/\varepsilon = 5$), 1st bucket

\[ \begin{array}{c}
\text{bucket 1} & \text{bucket 2} & \text{bucket 3} & \text{bucket 4} \\
1,2,4,3,4 & 3,4,5,4,6 & 7,3,6,1 & 1,3,2,4,7 \\
\end{array} \]

[Bucket 1] $b_{max} = 1$ 1st inserted: 1,2,4,3,4

Insert phase:
- $D$ (before removing): 1,2,4,3,4
- $e, f$, $\Delta$ are deleted
- $N = 5$

Delete phase:
- elements with $f + \Delta$ is deleted
- $f$ is integer representing the estimated frequency
- $\Delta$ is maximum possible error in $f$
- $\frac{1}{w} = \frac{1}{5}$

Example ($\varepsilon = 0.2$, $w = 1/\varepsilon = 5$), 2nd bucket

\[ \begin{array}{c}
\text{bucket 1} & \text{bucket 2} & \text{bucket 3} & \text{bucket 4} \\
1,2,4,3,4 & 3,4,5,4,6 & 7,3,6,1 & 1,3,2,4,7 \\
\end{array} \]

[Bucket 2] $b_{max} = 2$ 2nd inserted: 3,4,5,4,6

Insert phase:
- $D$ (before removing): 1,2,4,3,4
- $e, f$, $\Delta$ are deleted
- $N = 5$

Delete phase:
- delete elements with $f + \Delta$ is deleted
- $f$ is integer representing the estimated frequency
- $\Delta$ is maximum possible error in $f$
- $\frac{1}{w} = \frac{1}{5}$

Example ($\varepsilon = 0.2$, $w = 1/\varepsilon = 5$), 3rd bucket

\[ \begin{array}{c}
\text{bucket 1} & \text{bucket 2} & \text{bucket 3} & \text{bucket 4} \\
1,2,4,3,4 & 3,4,5,4,6 & 7,3,6,1 & 1,3,2,4,7 \\
\end{array} \]

[Bucket 1] $b_{max} = 1$ 1st inserted: 1,2,4,3,4

Insert phase:
- $D$ (before removing): 1,2,4,3,4
- $e, f$, $\Delta$ are deleted
- $N = 5$

Delete phase:
- elements with $f + \Delta$ is deleted
- $f$ is integer representing the estimated frequency
- $\Delta$ is maximum possible error in $f$
- $\frac{1}{w} = \frac{1}{5}$

Example ($\varepsilon = 0.2$, $w = 1/\varepsilon = 5$), 4th bucket

\[ \begin{array}{c}
\text{bucket 1} & \text{bucket 2} & \text{bucket 3} & \text{bucket 4} \\
1,2,4,3,4 & 3,4,5,4,6 & 7,3,6,1 & 1,3,2,4,7 \\
\end{array} \]

[Bucket 1] $b_{max} = 1$ 1st inserted: 1,2,4,3,4

Insert phase:
- $D$ (before removing): 1,2,4,3,4
- $e, f$, $\Delta$ are deleted
- $N = 5$

Delete phase:
- elements with $f + \Delta$ is deleted
- $f$ is integer representing the estimated frequency
- $\Delta$ is maximum possible error in $f$
- $\frac{1}{w} = \frac{1}{5}$
Generating output

<table>
<thead>
<tr>
<th>Item</th>
<th>True Frequency</th>
<th>Estimated Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

When a user requests a list of items with threshold \( s \), we output those entries in \( S \) where \( f(i) \geq (s - \varepsilon)N \). For example, if a user would like to see all of the item appeared at least 5% of current total number of items in the stream with \( \varepsilon = 0.2 \), the threshold \( s \) will be 0.25.