TP5: From Batch processing to Stream data processing: Getting Started with *Storm*

Alessio Pagliari
March 31, 2022

The goal of this TP is to study the implementation and the operation of the Storm stream processing platform. Deploy and configure a cluster and run simple a benchmark to understand the DAG application paradigm.

Exercise 1: Deploy Storm

The goal of this exercise is to deploy a Storm cluster on Grid 5000 and start with an initial exploration of Web UI.

Question 1.1

To quick deploy Storm you can retrieve the image available at `/home/alpagliari/public/storm_deploy`, you will find three files: `storm-image.tgz`, `storm-env.yaml` and `DeployStorm.sh`.

Copy those files to your home folder, **edit the path to the img** in the environment file, then deploy it:

- Reserve 3-4 nodes
- Deploy the image with:

```
$ kadeploy3 -f $OAR_NODE_FILE -a storm-env.yaml -k
```

- Finally, run the script `./DeployStorm.sh` to apply some basic configurations and start the Storm cluster.

Question 1.2

To check that everything is working:

- Create a bridge to the Nimbus node with port 8081.
- Connect to Storm Web UI at `localhost:8081`.

How many nodes do you see? How many available slots?

Exercise 2: Running your first Streaming application

In this exercise you will deploy a first streaming application, understand how to access the log files and the placement of the tasks in the workers.

Question 2.1

Copy the file `storm-starter-2.4.0.jar` from `/home/alpagliari/public/storm_deploy` to your nimbus node.

Run the RollingTopWords application through the command:

```
# storm jar storm-starter-2.4.0.jar org.apache.storm.starter.RollingTopWords
```

Check that the deployment is correct from Storm UI. You should see the topology running under the section "Topology Summary".

Question 2.2

From the Web UI open the running topology details:

- under "Topology Visualization" you can click "Show Visualization", you will see a draw of the application graph. Can you guess what the application does?
- under the section "Worker Resources" you can see how the tasks are placed in your available workers. How many executors per task do you see?
- in the topology panel you can see the processing latency of the tuples and the number of emitted tuples. Can you guess how you could compute an approximate average of the throughput?

Question 2.3

Always from the Web UI check where the last bolt (finalRanker) has been placed (node and worker port), as well as the Topology ID.

Connect to that node and check the `worker.log` file in:

```
# cd $STORM_HOME/logs/workers-artifacts/<topology-id>/<worker-port>/
```

What can you see in the log file, do you see any output?

Question 2.4

Kill the topology from the Web UI using the "kill" button in the topology page. The topology is killed when is no more visible in the "Topology Summary".

Exercise 3: Let's do some tuning

In the previous exercise the application where run with a parallelism level of 1 for every task. Now we will try to increase it and extend our cluster.

Question 3.1

Now let's run the application by specifying the number of worker and the parallelism of each task, like the example:

```
# storm jar storm-starter-2.4.0.jar org.apache.storm.starter.WordCountTopology 1 4 4 4
```

The specified values are: number of workers the topology will require, parallelism for each of the components. Note that the last component perform a `globalGrouping` to aggregate the final result, thus it is not parallelized.

Now deploy with three different configurations:

- Try deplying requiring the maximum amount of workers you have with a parallelism of 1 for each component. How is the application placed? What do you observe in terms of latency and throughput?
- With the same workers, increase the parallelism to 4 for each component. What do you observe in terms of palcement and preformance?
- Last one, try requiring more workers than the ones available. Is the application deployed anyway? If yes, do you observe difference in placement from the previous configuration?

Question 3.2

Now we increase the number of slots per node.

- Stop Storm cluster by killing the Nimbus and the Supervisors in each of the nodes, or you can send the command through ssh from Grid'5000 frontend.
- Only in the supervisor nodes modify the file `$STORM_HOME/conf/storm.yaml` by adding more slots for a total of 3-4 per supervisor, you could also try to assign different number of workers per node:

```
supervisor.slots.ports:
    - 6700
    - 6701
    - 6702
```

- Restart the Storm cluster. On the Nimbus, run:

```
# storm nimbus &
#storm ui &
```

  on the supervisors run:

```
# storm supervisor &
```

- Check that all the configured slots are available in the Web UI.

Question 3.3

Run again the RollingTopWords application with different configurations. Test different number of workers (e.g. 1, 3, max), with different parallelism levels for the applications.

What do you observe in the placement? What results do you get in latency and throughput?