Ecosistema de soporte a proyectos Big Data

Práctica 2 - Análisis de datos en tiempo real con Apache Storm

José Manuel Bustos Muñoz

Índice

1. Storm Cluster

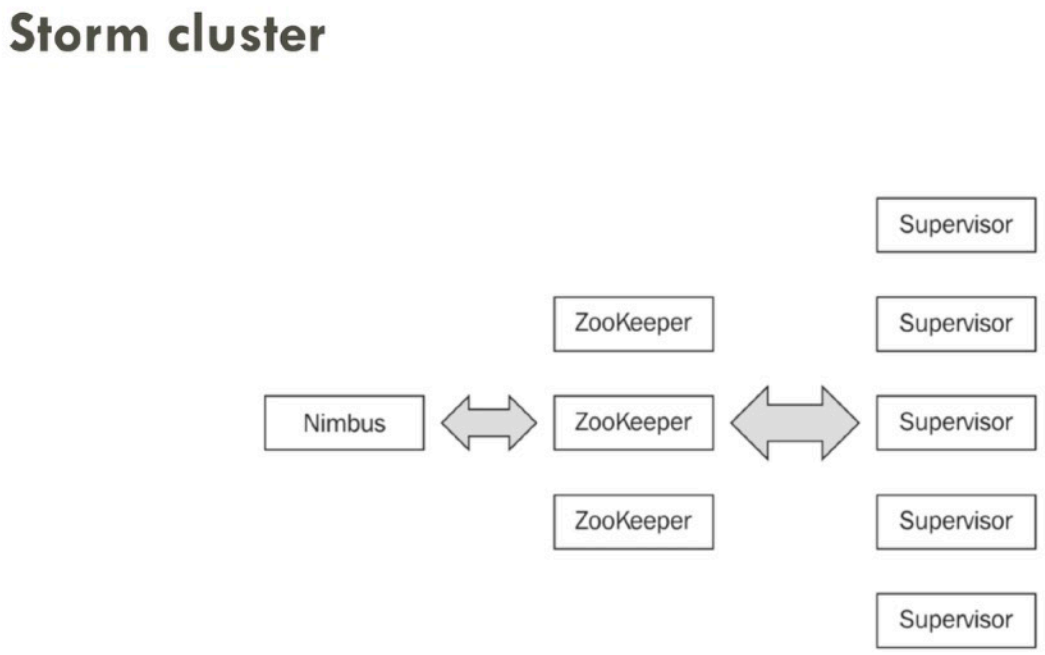
Provide a picture about the Storm cluster that we configured on previous laboratories. You should link nodes and tag following:

- Zookeeper
- Nimbus
- Supervisor

La arquitectura de Storm es bastante sencilla, se divide en los siguientes componentes:

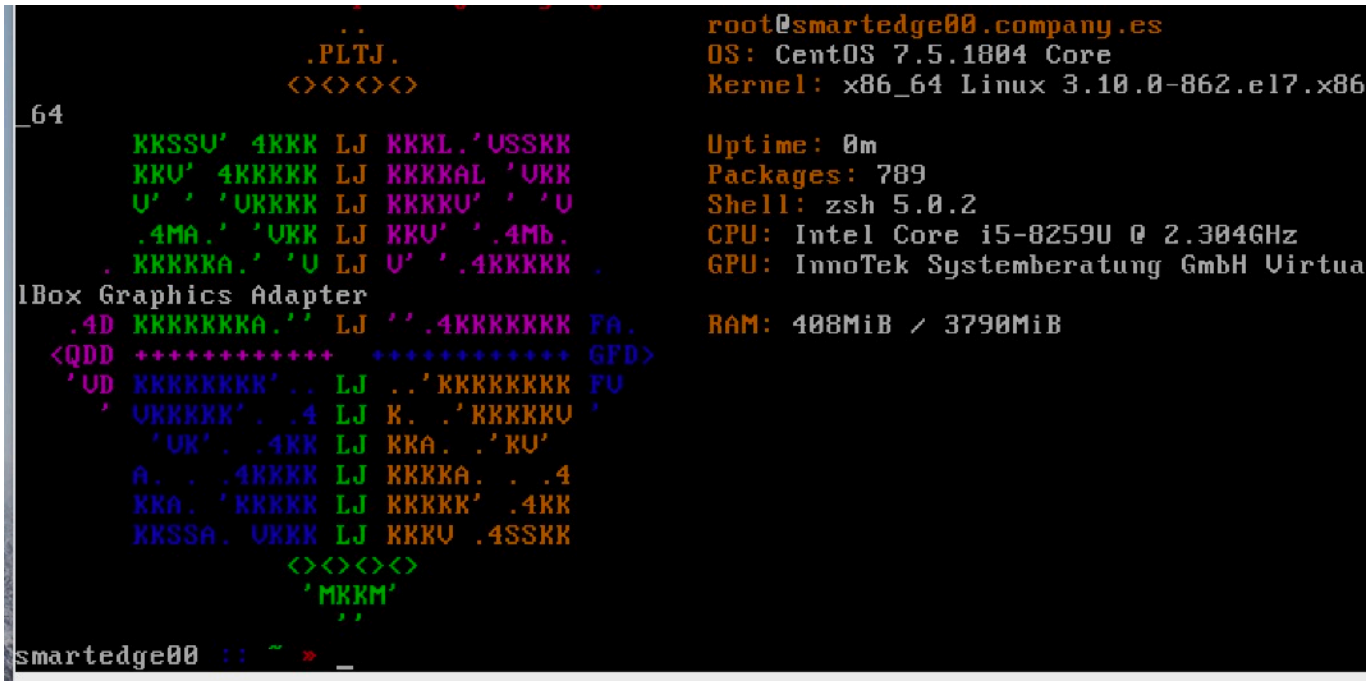
- El master node ejecuta el demonio llamado **Nimbus**, responsable de distribuir el código a través del cluster (similar al JobTracker de Hadoop). Realiza también la asignación y monitorización de las tareas en las distintas máquinas del cluster.
- Los worker nodes ejecutan el demonio **Supervisor** encargado de recoger y procesar los trabajos asignados en la máquina donde corre. Estos nodos ejecutan una porción de la topología para que así se puedan distribuir los trabajos a lo largo del cluster. Si fallara un worker node el demonio Nimbus se daría cuenta y redirigiría el trabajo a otro worker node.
- **Zookeeper**: aunque no es un componente como tal de Storm, si que es necesario montar un Apache Zookeeper ya que será el encargado de la coordinación entre el Nimbus y los Supervisors. También es el encargado de mantener el estado ya que el Nimbus y los Supervisors son stateless.

Imagen ejemplo de cluster Storm:



En la práctica en clase se ha configurado un cluster Storm similar, con 8 nodos. Para ello se ha utilizado una máquina virtual con contenedores dockers, y en ella se han realizado los pasos para dejarlo configurado.

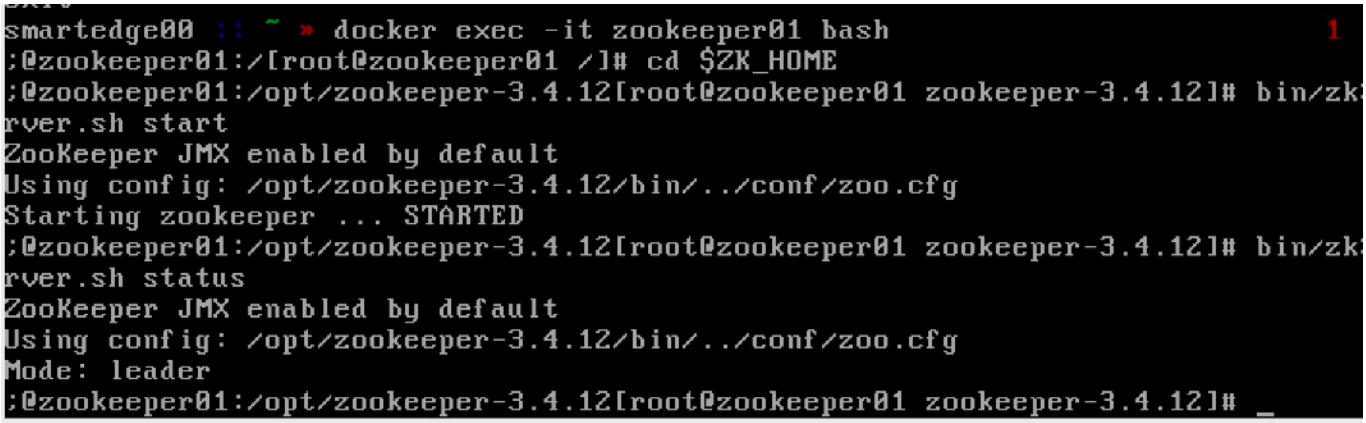
Máquina virtual:



Se listan los dockers. Se tienen 3 nodos zookeeper (zookeeper00, zookeeper01, zookeeper02) y 5 de Storm (storm00, storm01, storm02, storm03, storm04).

```
77b6ed67a98c        centos:centos7.5.1804   "/bin/bash"         3 months ago
    Up 57 seconds       2181/tcp, 3772-3773/tcp, 6627/tcp, 8000/tcp, 8080/tcp
storm04
4481f8f056be        centos:centos7.5.1804   "/bin/bash"         3 months ago
    Up 57 seconds       2181/tcp, 3772-3773/tcp, 6627/tcp, 8000/tcp, 8080/tcp
storm03
e32ed9f3ed34        centos:centos7.5.1804   "/bin/bash"         3 months ago
    Up 57 seconds       2181/tcp, 3772-3773/tcp, 6627/tcp, 8000/tcp, 8080/tcp
storm02
09296a4f7a20        centos:centos7.5.1804   "/bin/bash"         3 months ago
    Up 57 seconds       2181/tcp, 3772-3773/tcp, 6627/tcp, 8000/tcp, 8080/tcp
storm01
67de7555d7f4        centos:centos7.5.1804   "/bin/bash"         3 months ago
    Up 56 seconds       2181/tcp, 3772-3773/tcp, 6627/tcp, 8000/tcp, 8080/tcp
storm00
3bb49b8799c6        centos:centos7.5.1804   "/bin/bash"         4 months ago
    Up 56 seconds       2181/tcp, 2888/tcp, 3888/tcp
zookeeper02
b611ef2afaa8        centos:centos7.5.1804   "/bin/bash"         4 months ago
    Up 56 seconds       2181/tcp, 2888/tcp, 3888/tcp
zookeeper01
a2cd44748722        centos:centos7.5.1804   "/bin/bash"         4 months ago
    Up 56 seconds       2181/tcp, 2888/tcp, 3888/tcp
zookeeper00
smartedge00 :: ~ »
```

Dentro de los 3 zookeeper, el nodo maestro es el Zookeeper01, los otros dos son esclavos. Al lanzar la sentencia para ver el status sale el mensaje de “mode: leader”.

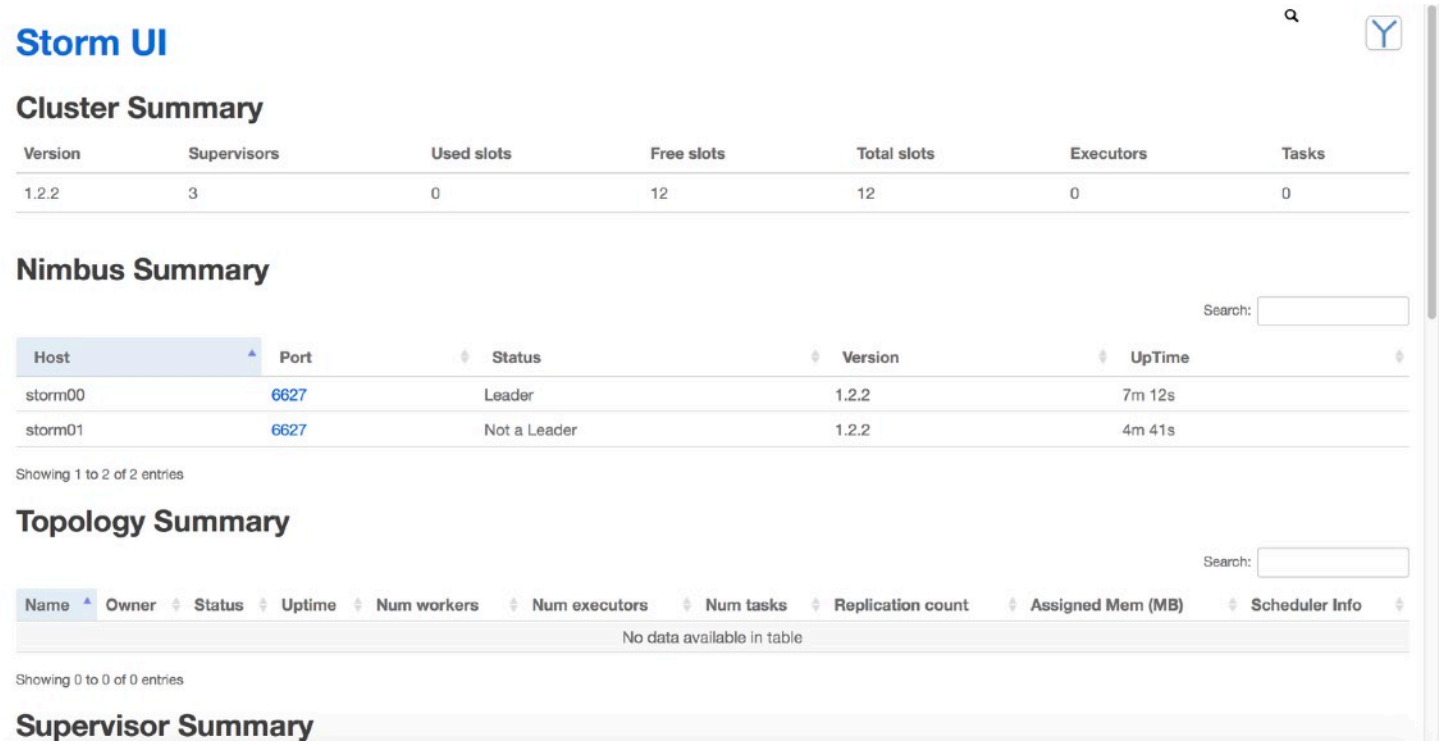


Luego se va accediendo a los nodos Storm, con secuencia de comandos como la siguiente:
“
docker exec -it storm01 bash
cd $STORM_HOME
bin/storm nimbus &
exit
“

Los nodos Storm se reparten del siguiente modo:
- Storm00 - Nimbus
- Storm01 - Nimbus
- Storm02 - Supervisor
- Storm03 - Supervisor
- Storm04 - Supervisor

2. Storm Topology

Una vez arrancados y configurados los distintos contenedores, obtenemos la ip y podemos desde el navegador local acceder a la interfaz de Storm.



Evaluamos el cluster Storm:

```
smartedge00 :: ~ » docker exec -it storm00 jps
133 core
26 nimbus
283 Jps
smartedge00 :: ~ » docker exec -it storm01 jps
195 Jps
24 nimbus
smartedge00 :: ~ » docker exec -it storm02 jps
24 Supervisor
120 Jps
smartedge00 :: ~ » docker exec -it storm03 jps
119 Jps
25 Supervisor
smartedge00 :: ~ » docker exec -it storm04 jps
119 Jps
25 Supervisor
smartedge00 :: ~ »
```

En local tenemos el código java para la topología, con maven intentamos generar el .jar para subirlo a la máquina virtual y realizar los siguientes pasos.

Descargamos maven y exportamos al path para poder utilizarlo:

```
bash: mvn: command not found
MBP-de-Jose:~ josemanuel$ export PATH=/Users/josemanuel/server/apache-maven-3.6.0/bin:$PATH
MBP-de-Jose:~ josemanuel$ mvn --version
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-24T20:41:47+02:00)
Maven home: /Users/josemanuel/server/apache-maven-3.6.0
Java version: 1.8.0_162, vendor: Oracle Corporation, runtime: /Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/jre
Default locale: es_ES, platform encoding: UTF-8
OS name: "mac os x", version: "10.13.6", arch: "x86_64", family: "mac"
MBP-de-Jose:~ josemanuel$
```

Lanzamos las sentencias de maven para generar el directorio del código, con el fichero pom correspondiente:

```
MBP-de-Jose:~ josemanuel$ mvn archetype:generate -DgroupId=com.storm.learn -DartifactId=storm-first -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.3
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.0.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.0.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.0.1:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[INFO] Using property: groupId = com.storm.learn
[INFO] Using property: artifactId = storm-first
Define value for property 'version' 1.0-SNAPSHOT: : 1.0
[INFO] Using property: package = com.storm.learn
Confirm properties configuration:
```

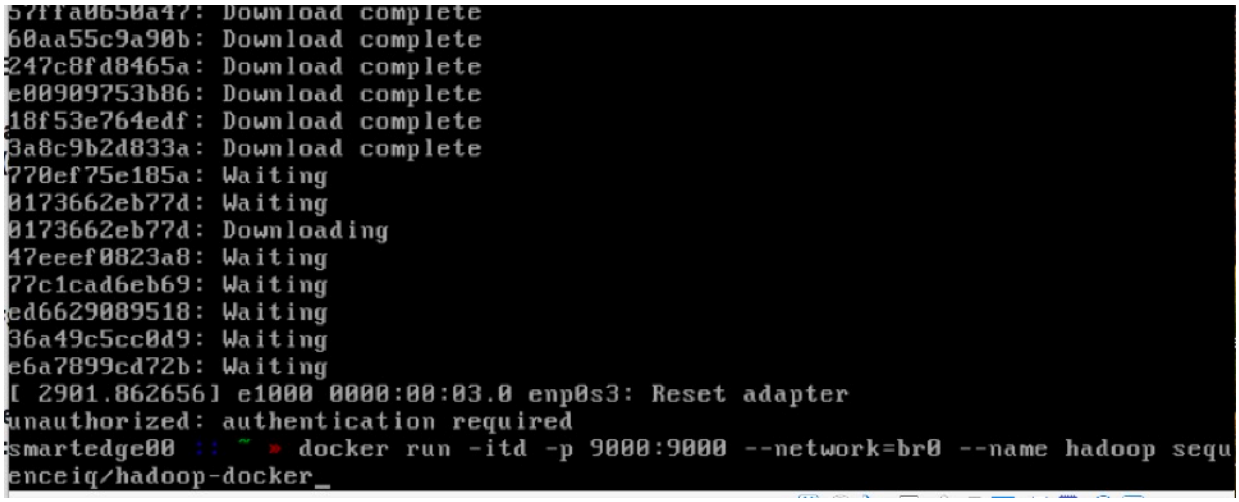
```
[MBP-de-Jose:~ josemanuel$ ls                                                   ]
Applications            Downloads               Public
Biblioteca de calibre   Library                 VirtualBox VMs
Calibre Library         Movies                  server
Desktop                 Music                   storm-first
Documents               Pictures
[MBP-de-Jose:~ josemanuel$ cd storm-first                                       ]
[MBP-de-Jose:storm-first josemanuel$ ls                                         ]
pom.xml src
```

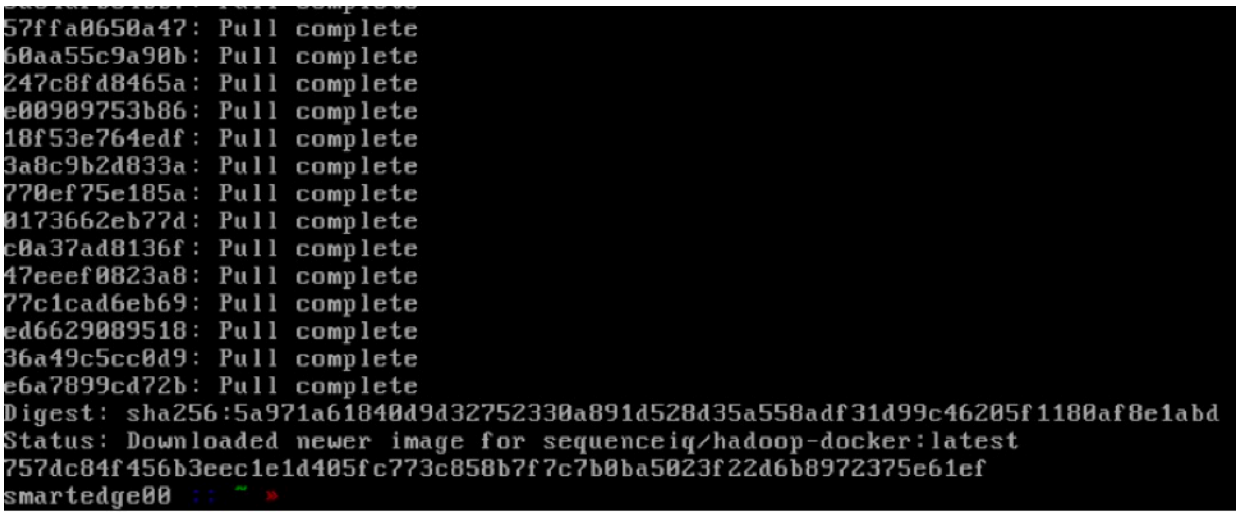
Al intentar generar el .jar con maven: "*mvn compile exec:java -Dexec.classpathScope=compile -Dexec.mainClass=com.storm.learn.FirstStormTopology*", se obtienen errores en la compilación y no se ha conseguido solucionar para generar correctamente el .jar y subirlo a la máquina virtual para generar la topología.

Accedemos al storm03 y vamos a la ruta de los logs:

```
;@storm03:/opt/apache-storm-1.2.2/logs/workers-artifacts[root@storm03 workers-ar
tifacts]# pwd
/opt/apache-storm-1.2.2/logs/workers-artifacts
;@storm03:/opt/apache-storm-1.2.2/logs/workers-artifacts[root@storm03 workers-ar
tifacts]# ls
FirstStormClusterTopology-25-1537717737   storm-apache-log-12-1535036164
apache_log_error_filtering-18-1535371296  storm-apache-log-13-1535037139
apache_log_error_filtering-19-1535371514  storm-apache-log-14-1535038008
apache_log_error_filtering-20-1535372099  storm-apache-log-15-1535038161
apache_log_error_filtering-21-1535372294  storm-apache-log-16-1535097814
apache_log_error_filtering-22-1535372578  storm-apache-log-17-1535098095
apache_log_error_filtering-23-1535372804  storm-hadoop-5-1534252023
apache_log_error_filtering-24-1535372957  storm-hadoop-6-1534253613
example-2-1532951337                      storm-hadoop-7-1534254246
example-3-1532952371                      storm-starter-1-1532694422
example-4-1532952748
;@storm03:/opt/apache-storm-1.2.2/logs/workers-artifacts[root@storm03 workers-ar
tifacts]# _
```

En la máquina virtual descargamos el contenedor de Hadoop y comprobamos que se ha instalado con las sentencias:
docker pull sequenceiq/hadoop-docker
docker run -itd -p 9000:9000 --network=br0 --name hadoop sequenceiq/hadoop-docker
docker ps





cd $HADOOP_PREFIX
bin/hdfs dfs -ls /

A partir de aquí con los siguientes pasos se han encontrado errores y no se ha podido seguir, entiendo que al no haber realizado correctamente el lab2 con la topología, no se puede realizar bien esta parte tampoco.

3. Preguntas

a. What is a Spout? And a bolt?

Un Spout se encarga de recoger el flujo de datos de entrada, de la ingesta de datos. Un Bolt se encarga del procesado o transformación de los datos, consumiendo las tuplas emitidas por el Spout.

En la documentación oficial se representan los Spouts con grifos simulando la entrada de un stream de datos al sistema, y a los Bolts con un rayo que es donde se realizan las acciones pertinentes con los datos de entrada.

Algunas características más:

- Spout:
    - El Spout es una fuente de streams en un topology.
    - Generalmente leen tuplas de la fuente externa y los emiten en la topología.
    - Un spout es una secuencia de streams.
- Bolt:
    - En ellos se realiza el procesamiento del topology. Pueden realizar operaciones de filtrado, funciones, agregados, conexiones con BBDD, etc.
    - Consumen cualquier número de streams de entrada, realizan el procesamiento y posiblemente emiten nuevos streams.
    - Las transformaciones complejas requieren múltiples pasos y bolts.

b. What types of processing can I execute with Storm regarding message delivery?

Storm ofrece varios niveles diferentes de procesamiento de mensajes garantizados, incluido el mayor esfuerzo, al menos una vez y exactamente una vez a través de Trident.

Storm garantiza que cada tupla se procesará completamente. Las abstracciones básicas de Storm proporcionan una garantía de procesamiento de al menos una vez.

"*At most once processing*", las tuplas fallidas no son reintentadas. El Spout no espera un reconocimiento.

"*At least once processing*", las tuplas fallidas se reintentan en el proceso de procesamiento. Garantiza que cada tupla que ingresa al proceso de procesamiento debe procesarse al menos una vez.

c. Which sentence is the right one?: 'A task runs executors' or 'A executor runs tasks'.

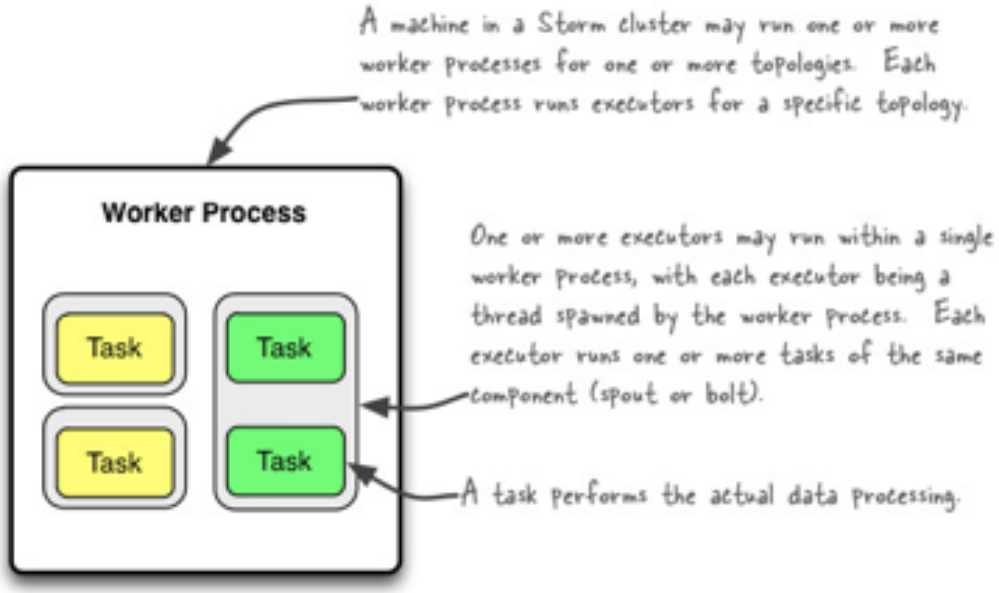
La correcta sería "A executor runs tasks".

Un proceso Worker ejecuta un subconjunto de una topología, y puede ejecutar uno o más Executors para uno o más componentes de la topología.

Un Executor es un subproceso generado por un proceso Worker, y puede ejecutar/correr una o más tareas para el mismo componente (Spout o Bolt).

Por lo tanto los Executors ejecutan las Tasks. El número de tareas para un componente es el mismo durante la vida útil de una topología, pero el número de Executors para un componente puede variar, por ello se cumple la condición: Executors <= Tasks.

De forma predeterminada se establece el mismo número de tasks y de executors.



d. Which command should I execute to change the parallelism of a running topology?

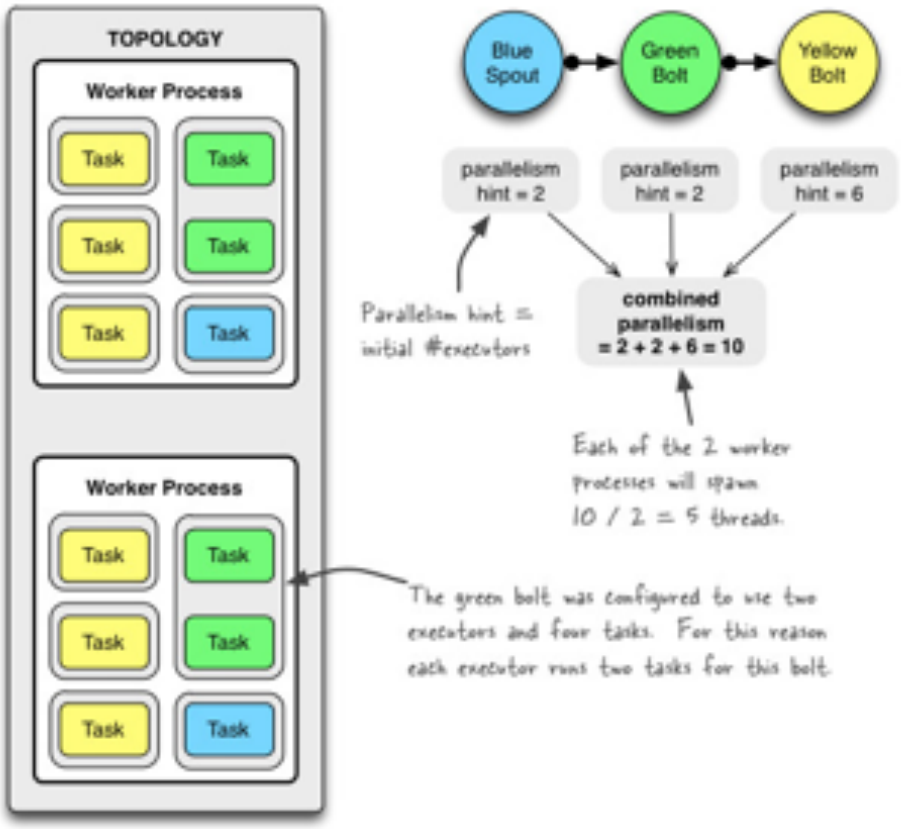
Para cambiar el paralelismo en una topología se puede hacer un rebalanceo, incrementando o disminuyendo el número de procesos worker y/o executors, sin ser necesario el reinicio del cluster de la topología.

Hay 2 opciones para rebalancear una topología:

1. Usar la UI de Storm.
2. Usar la herramienta CLI de Storm.

Un ejemplo de la herramienta CLI:

$ storm rebalance mytopology –n 5 –e blue-spout=3 –e yellow-bolt=10



e. If I have a single high performance node in my cluster, Which scheduler I will should to set?

El Scheduler ayuda a Nimbus a decidir la distribución de los workers de cualquier topología dada.
Storm tiene 4 tipos de Schedulers:

- "*DefaultScheduler*" asigna a los workers de componentes de la forma más equitativa posible.
- "*IsolationScheduler*" ayuda a asignar/reservar los conjuntos dedicados de nodos Storm para las topologías dentro del cluster.
- "*MultitenantScheduler*" aisla las topologías entre sí.
- "*ResourceAwareScheduler*" especifica la cantidad de recursos necesarios para una sola instancia de cualquier componente.

Para este caso se configuraría el "ResourceAwareScheduler".