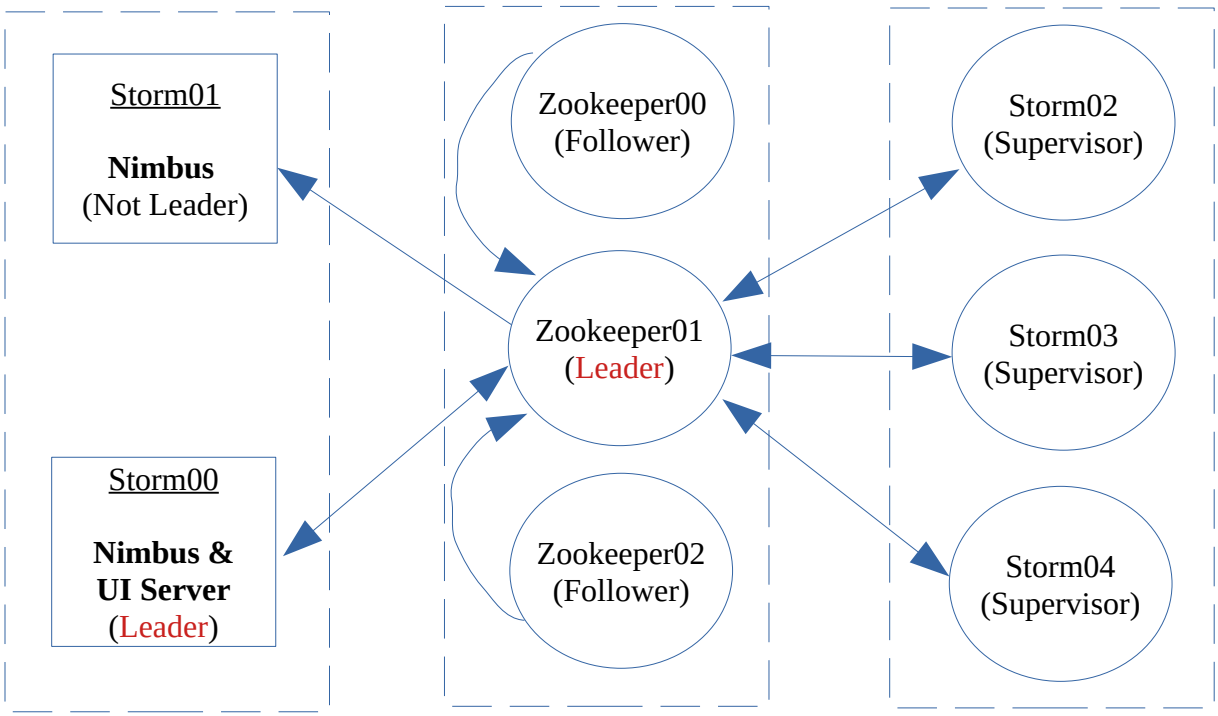
Ciclo de Vida Analítico del Dato
Práctica 2

Daniel Pérez Efremova

Enero de 2022

Ejercicio I: Una imagen que represente el clúster de Storm utilizado en los laboratorios: (3 puntos)



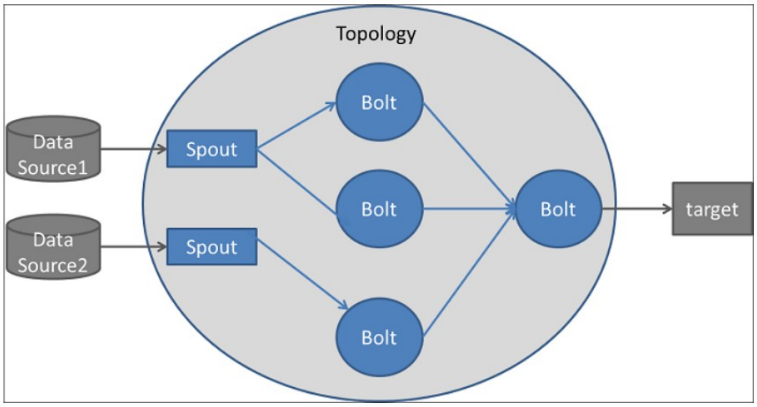
Preguntas:

1. ¿Qué es una topología, un Spout y un Bolt en Storm? (0.5 puntos)

Una **topología** de Apache Storm consiste en un grafo de computación que establece cómo se procesan determinados datos en tiempo real. Cada nodo de la topología contiene sus propias instrucciones de procesamiento y las aristas del grafo indican cómo se distribuyen los datos por la topología. Dichos nodos pueden ser de tipo Spout o Bolt.

Los nodos **Spout** se encargan de leer tuplas de la fuente que transmite los datos y las distribuye por la topología para ser procesadas. Los nodos **Bolt** se encargan de procesar las tuplas que generan los Spouts.

En la imágen inferior se muestra un ejemplo de una topología.



2. ¿Qué tipos de procesamiento puedo ejecutar con Storm en relación con la entrega de mensajes? ¿Podría proporcionar una breve descripción y un caso de uso para cada uno de ellos? (0.5 puntos)

Storm ofrece tres opciones de procesamiento garantizado para cada tupla dentro del circuito de procesamiento.

1. At-Most-Once. Cada tupla es procesada a lo sumo una vez por los nodos Bolt, pero estos no se comunican con el Spout que ha enviado la tupla (no hay acuse de recibo). De esta manera, no se vuelve a procesar una tupla fallida.

- Ventajas: Alto rendimiento y bajo esfuerzo de implementación.

- Desventajas: Posibilidad de perder mensajes e integridad de la información comprometida.

- Propuesta de caso de uso: Análisis de sentimiento en tiempo real de un chat de Twitch. Por lo general se producen cientos de interacciones por segundo pero se repiten interacciones o no todas son analizables. Es un caso de uso interesante porque se requiere un alto rendimiento y el riesgo de perder información relevante no es significativo dado el volumen de interacciones.

2. At-Least-Once. Cada tupla es procesada como mínimo una vez por los nodos Bolt (hay acuse de recibo). En caso de que haya una tupla fallida se vuelve a mandar el mensaje. Existe riesgo de duplicar mensajes, pero se garantiza que ninguno se pierde.

- Ventajas: Alta integridad de la información.

- Desventajas: Bajo rendimiento.

- Propuesta de caso de uso: Análisis en tiempo real del funcionamiento de una central nuclear. Por lo general se miden muchas variables acerca de los procesos y toda la información es relevante para los sistemas de control. Es un caso de uso interesante porque la integridad de la información es vital para su funcionamiento, lo cual justifica un menor rendimiento del sistema o un gasto en recursos de computación muy alto.

3. Exactly-One. Cada tupla se envía una única vez y no se pierden.

- Ventajas: Alta integridad y eficiencia en la información

- Desventajas: Alto coste de implementación.

- Propuesta de caso de uso: Análisis en tiempo real del precio de acciones para microtrading. Este sistema consiste en pequeñas transacciones en cuestión de segundos de valores bursátiles. La integridad de la información para la correcta interpretación del valor de un activo es vital y dada la complejidad del análisis, debe garantizarse que se dispone de toda la información y de que no hay mensajes duplicados que distorsionen los resultados. Es un caso de uso interesante porque exige gran precisión e integridad en la información que se maneja, lo cual justifica el alto coste en la implementación.

3. ¿Una tarea ejecuta executors o un executor ejecuta tareas?(0.5 puntos)

Un proceso worker ejecuta una parte de la topología a la cual pertenece y puede crear uno o más executors. A su vez, un executor es un hilo creado por el proceso worker y puede crear una o más

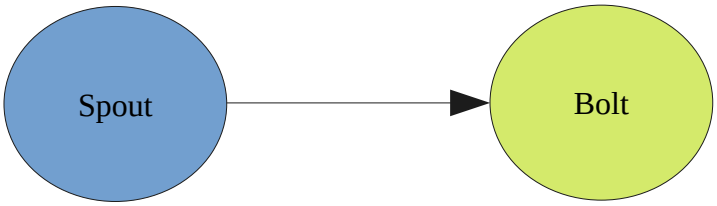
tareas para el mismo componente (Bolt o Spout) [1]. Estos tres aspectos pueden configurarse de antemano o rebalancearse sin parar el cluster.

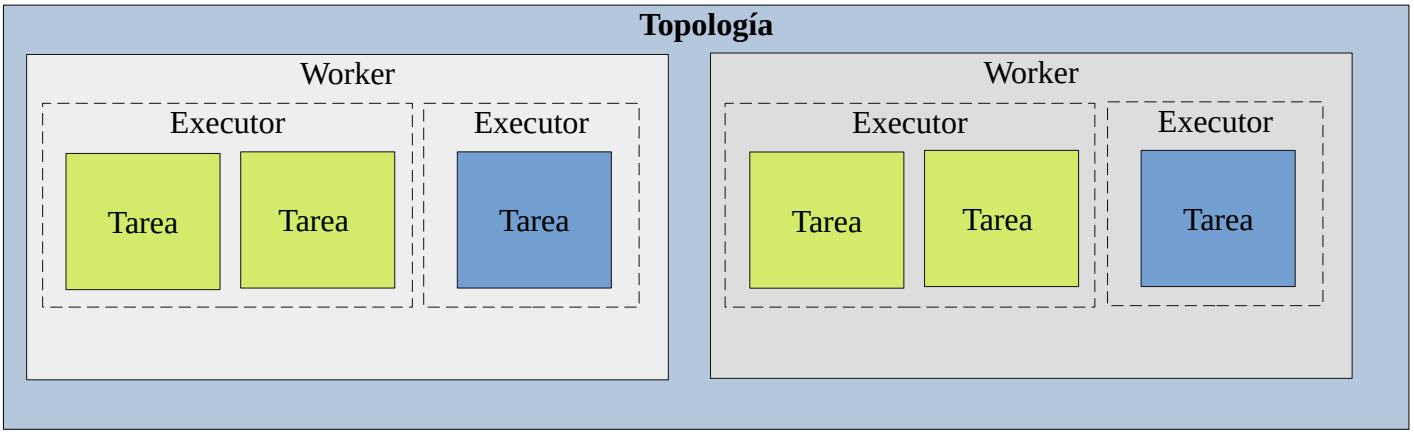
Supongamos que tenemos una máquina con 4 hilos disponibles y que tenemos una topología con un nodo Spout y otro Bolt. La topología se configura de tal manera que hayan dos procesos worker, cuatro tareas para el nodo Bolt y dos tareas para el Spout.

Al haber 4 hilos, cada worker creará a lo sumo dos executors. Cada executor se ocupará de las tareas de un nodo. Para el caso del nodo Spout se realizan dos tareas, cada una de ellas creadas por un executor en cada proceso worker. En el caso del nodo Bolt se realizan cuatro tareas, repartidas en dos workers. Al haber solo dos hilos restantes, un executor en cada worker se encargará de dos tareas del nodo Bolt. El número de tareas por componente es siempre el mismo durante la vida de una topología, pero el número de executors puede variar siguiendo la condición:

#hilos <= #tareas

Cabe señalar que si hubiera más maquinas, cada máquina corre múltiples workers pertenecientes a una o varias topologías, pero dentro de un worker se crean executors de una única topología. Los executors corren una o varias tareas siempre del mismo nodo (Bolt o Spout), no pueden mezclarse.





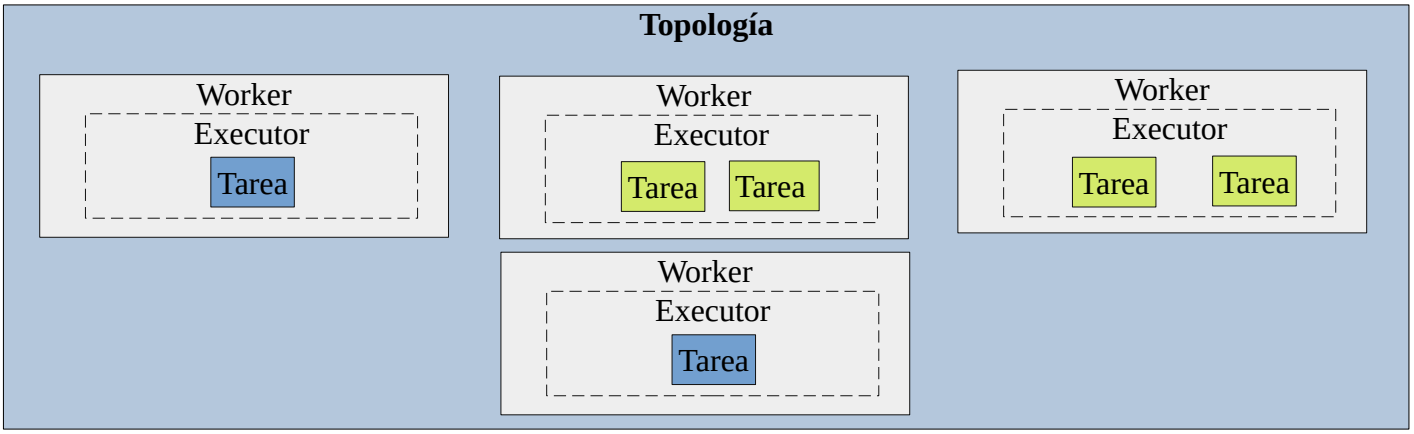
4. ¿Qué comando debo ejecutar para cambiar el paralelismo de una topología en funcionamiento? (0.5 puntos)

Siguiendo con el ejemplo anterior, supongamos que se quieren usar cuatro workers y cuatro executors con el mismo número de tareas. Este rebalanceo se puede hacer mediante la UI o mediante la herramientaCLI de Storm:

$ storm rebalance mytopology -n 4 -e blue-spout=2 -e yellow-bolt=2

1 **Recurso utilizado:** *storm.apache.org/releases/2.2.0/Understanding-the-parallelism-of-a-Storm-topology.html*

Con este rebalanceo del paralelismo, la ejecución quedaría como en la imagen siguiente.



5. Si uno de los nodos que forman mi clúster de Storm es de alto rendimiento (ej GPU), ¿qué planificador debería utilizar para extraer todo su valor y optimizar los recursos? (0.5 puntos)

La mejor opción desarrollar un scheduler propio. En este blog se explica un caso particular similar al propuesto: https://inside.edited.com/taking-control-of-your-apache-storm-cluster-with-tag-aware-scheduling-b60aaaa5e37e.

En el caso expuesto en el blog, se quiere aumentar el rendimiento del cluster para procesar imágenes. Esto se consigue asignando las tareas de un Bolt de la topología a Workers localizados en una máquina del cluster dotada de GPU.

Para conseguirlo, hay que poner etiquetas (**tags**) en la configuración de los objetos de la topología y los nodos supervisor del cluster. Finalmente, hay que diseñar una clase que mapee las etiquetas de cada nodo supervisor y las compare con las etiquetas de los Bolts, para poder asignar las tareas del Bolt encargado de procesar imágenes a un executor localizado en el nodo del cluster con GPU (Nótese que los detalles técnicos son más complicados).