UNIVERSIDAD DE GRANADA

Big Data II

máster ciencia de datos e ingeniería de computadores

Ejecución paralela en Pig

Práctica sobre ETL

Autor
Ignacio Vellido Expósito
ignaciove@correo.ugr.es

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2020-2021

1. Experimento

Dataset con medidas de peticiones en la red de una universidad, publicado en la página de la UCI (`https://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data`). Cuenta con 65.532 instancias y los siguientes 12 atributos:

- Source Port
- Destination Port
- NAT Source Port
- NAT Destination Port
- Action (cuatro tipos: allow, action, drop y reset-both)
- Bytes
- Bytes Sent
- Bytes Received
- Packets
- Elapsed Time (sec)
- pkts_sent
- pkts_received

1.1. Pasos para el desarrollo del experimento

Tras descargar el .csv, debemos cargarlo en HDFS.

```
$ mv Downloads/log2.csv /var/tmp/materialPig/log2.csv
```

Creamos el directorio en HDFS:

```
$ hdfs dfs -mkdir input
```

Cargamos los datos en el directorio:

```
$ hdfs dfs -put /var/tmp/materialPig/log2.csv input
```

Entramos en Pig:

```
$ pig
```

Creamos esquema del flujo del que leer los datos:

```
> measure = load 'input/log2.csv'
            using PigStorage(',')
            as (
                SourcePort:int ,
                DestinationPort:int ,
                NATSourcePort:int ,
                NATDestinationPort:int ,
                Action:chararray,
                Bytes:int ,
                BytesSent:int ,
                BytesReceived:int ,
                Packets:int ,
                ElapsedTime:int ,
                pktsSent:int ,
                pktsReceived:int
            );
```

Aplicamos la consulta: **Calcular la cantidad máxima de bytes enviados y la carga media por paquete (en cualquier puerto de destino) para aquel puerto con más solicitudes al puerto HTTPS (443)**

1. Seleccionamos las peticiones a HTTPS:

```
> all_ports = foreach measure generate DestinationPort, SourcePort;
> https_ports = filter all_ports by DestinationPort==443;
```

2. Agrupamos por puerto de origen y calculamos el que ha realizado más peticiones:

```
> ports = group https_ports by SourcePort ;
> groups = foreach ports generate group, COUNT(https_ports) as count;
> max_port = limit (order groups by count desc) 1;
```

```
il.MapRedUtil - Total input paths to process : 1
(59779,7)
```

(Hace falta renombrar COUNT para hacerle referencia en la expresión siguiente)

3. Filtramos peticiones por puerto de origen y calculamos la información solicitada

```
> petitions = filter measure by SourcePort==max_port.group;
> out = foreach (group petitions by SourcePort)
        generate group,
                 MAX(petitions.BytesSent),
                 (float)(SUM(petitions.BytesSent)/SUM(petitions.Packets));
```

(Aunque todas las peticiones sean del mismo puerto, agrupamos para poder mostrarlo con facilidad)

4-1. Podemos mostrar el resultado con:

```
> dump out;
```

4-2. O guardarlo con:

```
> store out into 'pigResults/firewall' using PigStorage(',');;
```

```
il.MapRedUtil - Total input paths to process : 1
(59779,3602,100.0)
```