To test my Pig installation I extracted source IP addresses from raw IP packets. I used the tcpdump capability of Wireshark to create a CSV file. Every row was in a form similar to 01d0,e2,b6,49,80,84,9e,62,96,49,1e,e1,28,6,5c,c4,78. The first hexadecimal value is the row offset that represents that row's position within the packet. The next 16 values are 16 of the many bytes per packet. This gave me a two-dimensional challenge. I had to get the right columns from the right rows. I know from the standard IPv4 header format that the source IP address would be in the row with a 0x0010 offset, and be the 11ththrough 14th value on this row. I used WinSCP to copy my packet\_bytes.csv file to my CentOS installation, and copyFromLocal to move it into HDFS. Next, I created an alias "packets" for the command to load the CSV file.

[hadoop@pig ~]$ hadoop fs -copyFromLocal ~/packet\_bytes.csv wk5

[hadoop@pig ~]$ hadoop fs -ls wk5

Found 3 items

-rw-r--r--   1 hadoop supergroup   2893177 2015-09-19 12:27 wk5/movies\_data.csv

-rw-r--r--   1 hadoop supergroup     43653 2015-09-26 12:38 wk5/packet\_bytes.csv

-rw-r--r--   1 hadoop supergroup   3927127 2015-09-19 13:31 wk5/wireshark.txt

grunt> packets = LOAD 'wk5/packet\_bytes.csv'

USING PigStorage(',')

AS (offset:chararray, b0:chararray, b1:chararray, b2:chararray, b3:chararray, b4:chararray, b5:chararray, b6:chararray, b7:chararray, b8:chararray, b9:chararray, b10:chararray, b11:chararray, b12:chararray, b13:chararray, b14:chararray, b15:chararray);

grunt> DUMP packets;

(0,80,19,34,78,88,6d,60,e3,27,52,8e,c1,8,0,45,40)

(10,0,83,0,0,40,0,38,11,c0,6d,c0,a8,0,1,c0,a8)

(20,0,6b,0,35,f1,c2,0,6f,86,9c,63,dc,81,83,0,1)

(30,0,0,0,1,0,0,3,32,32,39,3,32,31,31,2,39)

(40,33,3,32,30,37,7,69,6e,2d,61,64,64,72,4,61,72)

(50,70,61,0,0,0c,0,1,c0,10,0,6,0,1,0,0,4)

(60,39,0,2e,3,6e,73,31,3,69,63,67,3,6e,65,74,0)

(70,3,64,6e,73,6,6c,65,76,65,6c,33,c0,41,77,dd,eb)

(80,b4,0,0,1c,20,0,0,2,58,0,27,8d,0,0,0,0e)

(90,10,.,,,,,,,,,,,,,,)

Note that this is just one of hundreds of packets in the CSV file. My next step was to filter on only rows that started with an offset of 0x0010. This hexadecimal value was represented in my CSV file with the leading zeros truncated.

grunt> offset\_10 = FILTER packets BY offset == '10';

grunt> DUMP offset\_10;

(10,0,c9,0,0,40,0,40,11,79,7b,c0,a8,0,1,ff,ff)

(10,0,28,70,3b,40,0,78,6,2e,1e,cf,5d,d3,e5,c0,a8)

(10,0,28,70,3c,40,0,78,6,2e,1d,cf,5d,d3,e5,c0,a8)

(10,0,49,3a,9d,0,0,80,11,7e,4a,c0,a8,0,6b,c0,a8)

(10,0,83,0,0,40,0,38,11,c0,6d,c0,a8,0,1,c0,a8)

The final step was to iterate through each row on my filtered table and extract the source IP addresses in hexadecimal form:

grunt> src = FOREACH offset\_10 GENERATE b10, b11, b12, b13;

grunt> DUMP src;

(c0,a8,0,1)

(c0,a8,0,6b)

(c0,a8,0,1)

(c0,a8,0,1)

(c0,a8,0,6b)

(c0,a8,0,6b)

(c0,a8,0,1)

(cf,5d,d3,e5)

(cf,5d,d3,e5)

(c0,a8,0,6b)

(c0,a8,0,1)

grunt>

The following table converts the IP addresses from hexadecimal to decimal format. 192.168.0.1 is my Virtual Box host adapter. 192.168.0.107 is my CentOS guest. 207.93.211.229 is registered through ISP Level 3 Communications to Regis College. That packet was most likely a Keep-Alive packet because I had WorldClass open and connected when I captured my packets.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Hexadecimal | | | | Decimal | | | |
| c0 | a8 | 0 | 1 | 192 | 168 | 0 | 1 |
| c0 | a8 | 0 | 6b | 192 | 168 | 0 | 107 |
| cf | 5d | d3 | e5 | 207 | 93 | 211 | 229 |

If anyone wants to build on this, I'd be interested in seeing what you did. The destination addresses are b14 and b15 on row 10, and b0 and b1 on row 20.

You would need

grunt> offset\_10 = FILTER packets BY offset == '10';

grunt> offset\_20 = FILTER packets BY offset == '20';

grunt> dest1 = FOREACH offset\_10 GENERATE b14, b15;

grunt> dest2 = FOREACH offset\_20 GENERATE b0, b1;

How do you combine dest1 and dest2?