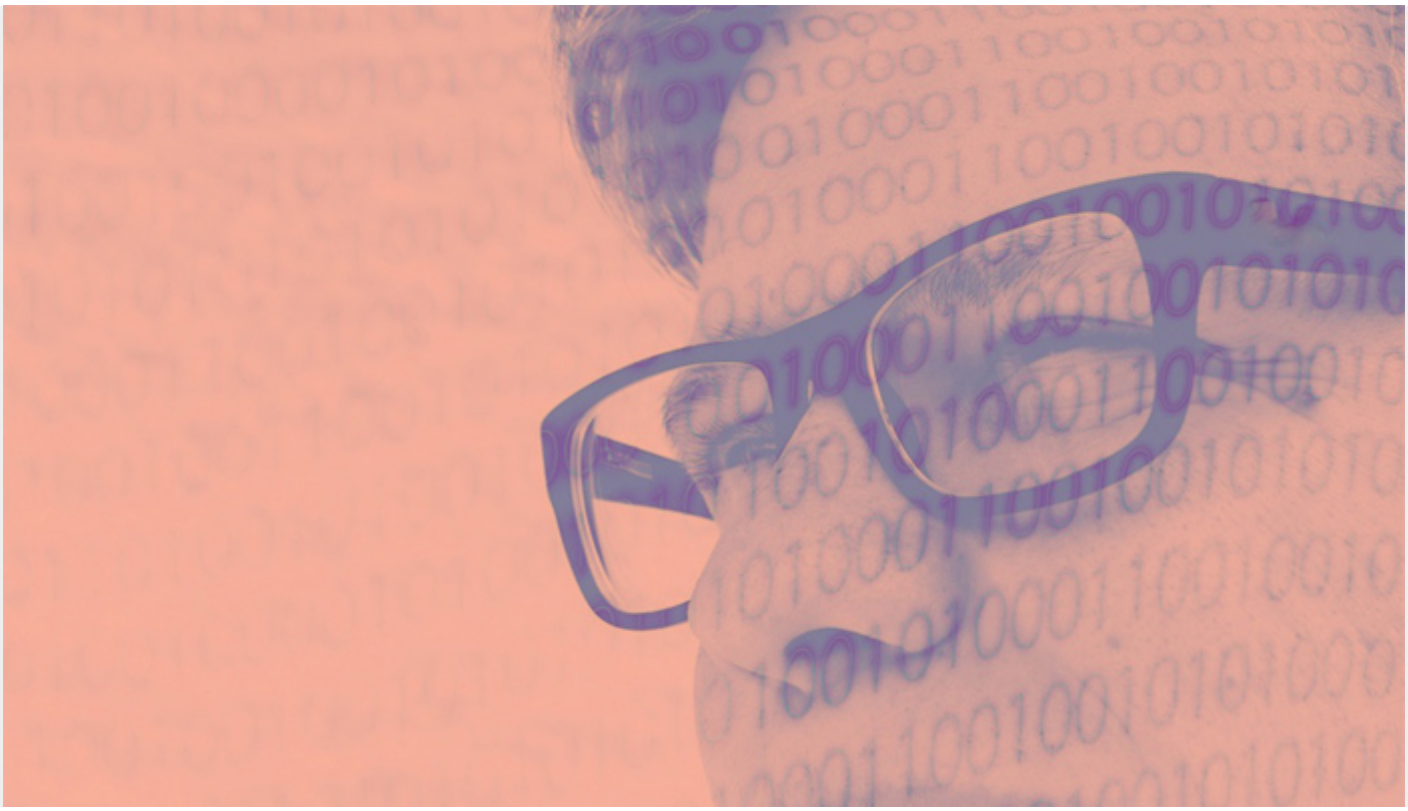
BASICS OF GRAPHING STREAMING BIG DATA



Imagine creating a live chart that updates as data flows in. With this you could watch currency value fluctuations, streaming IOT data, application performance, cybersecurity events, or other data in real time.

It is not so hard to create Spark Streaming data. We give an example below. But creating any graphs more elaborate than simple SQL charts is quite complex. The problem is you either need to know Matplotlib (works with Python) or Highcharts. Highcharts requires that you know AngularJS, which is a JavaScript framework for creating web pages using JavaScript.

To make live charts using JavaScript you need to add watch variables and pass data to them, which Apache Spark explains here. Apache Zeppelin supports that with Spark and Scala. It does not support live charts with anything but Scala, yet.

Here we show how to pass variables to AngularJS below.

Matplotlib is more of a graphing package for data science, Python, and R programmers. It does not support live updates. But, on the positive side, it is not complicated. To create charts by yourself in JavaScript would be quite complex, which is why you need a framework, like HighCharts.

HighCharts is free for non-commercial use. You could also look at Google Charts for the R programming language, which is called googleVis. Regarding Google Charts, it is designed for Google's own cloud and Google Big Tables rather than open source Apache Spark.

One thing that makes graphing difficult in any language is understanding the different kinds of

charts. There are dozens. You can learn something about that by studying these design principles on the HighCharts website.

Install Zeppelin

So let's get going. Here we are going to give the simplest possible example of passing data to an web page (i.e., AngularJS). Then we give the simplest possible example of Spark Streaming.

First, you need to install Zeppelin. The easiest way to do that is to use Docker:

```
docker pull dylanmei/zeppelin
docker run --rm -p 8080:8080 dylanmei/zeppelin
```

Binding

Zeppelin lets you pass variables to Angular using z-commands, like z.put, z.run, and z.angularBind. Here we make the simplest possible example. (The problem with most examples on the internet is they are too complicated. No one starts with a very simple example that you can easily understand. So we do.)

Below we make an Array of 1 element. Then we create an RDD and map over it 1 time to pass the value 1 to the HTML table element "1" shown below.

```
val data = Array(1)
val distData = sc.parallelize(data)
distData.map( l=> z.angularBind(l.toString, 1))
```

This is an HTML table. We use **%angular** to tell Zeppelin that this code is JavaScript and not Spark. Note that we put the variable name that we want to pass data too in double curly braces {{}}. We do not actually use any JavaScript, just simple HTML.

```
%angular
<html>
<table>
<tr>
<td>value={{1}}</td>
</tr>
</table>
```

So when you run that and the Spark Scala code it will output:

```
value=1
```

Streaming code

Below we give an example of Spark Streaming code. In part II of this blog post we will show how to make a graph using this data and HighCharts. But for now just get a feel of how to create streaming code. We also wrote another, more complex example of streaming Twitter Tweets here.

Know that when you run streaming code in Zeppelin it will freeze up your browser, since it is streaming. So you have to kill the Docker process like this:

docker stop $(docker ps -aq)

If you kill it any other way you will have errors because you are trying to run to SparkContexts at the same time.

What this example does is listen on port 80 for traffic on your laptop, i.e., web page traffic. If you were to dump that as text you could see data packets. Here we just create a Dstream and use the info() method to echo some of that. It does not show the whole packet.

To prove that you have data coming into port 80 on your laptop, you could install nmap and then run:

while true;do nmap --packet-trace -A localhost -p 80;sleep 1;done

Here is the code.

```
import spark.implicits._
import org.apache.spark.sql.types._
import org.apache.spark.sql.Encoders
import org.apache.spark.streaming._
val ssc = new StreamingContext(sc, Seconds(5))
val lines = ssc.socketTextStream("localhost", 80)
lines.print()
ssc.start()
ssc.awaitTermination()
```

It should output something like this is a continuous loop:

Time: 1499497850000 ms

Next steps

The next steps would be to make JavaScript AngularJS variables listen for updates from the Spark bound variables and then use HighCharts, the JavaScript framework for graphs. That is far more complicated than what we just showed.