Big Data Visualization *using* Apache Spark and Zeppelin

Prajod Vettiyattil, Software Architect, Wipro

https://in.linkedin.com/in/prajod @prajods

saltmarch MEDIA

GREAT INDIAN DEVELOPER SUMMIT

Agenda

- Big Data and Ecosystem tools
- Apache Spark
- Apache Zeppelin
- Data Visualization
- Combining Spark and Zeppelin

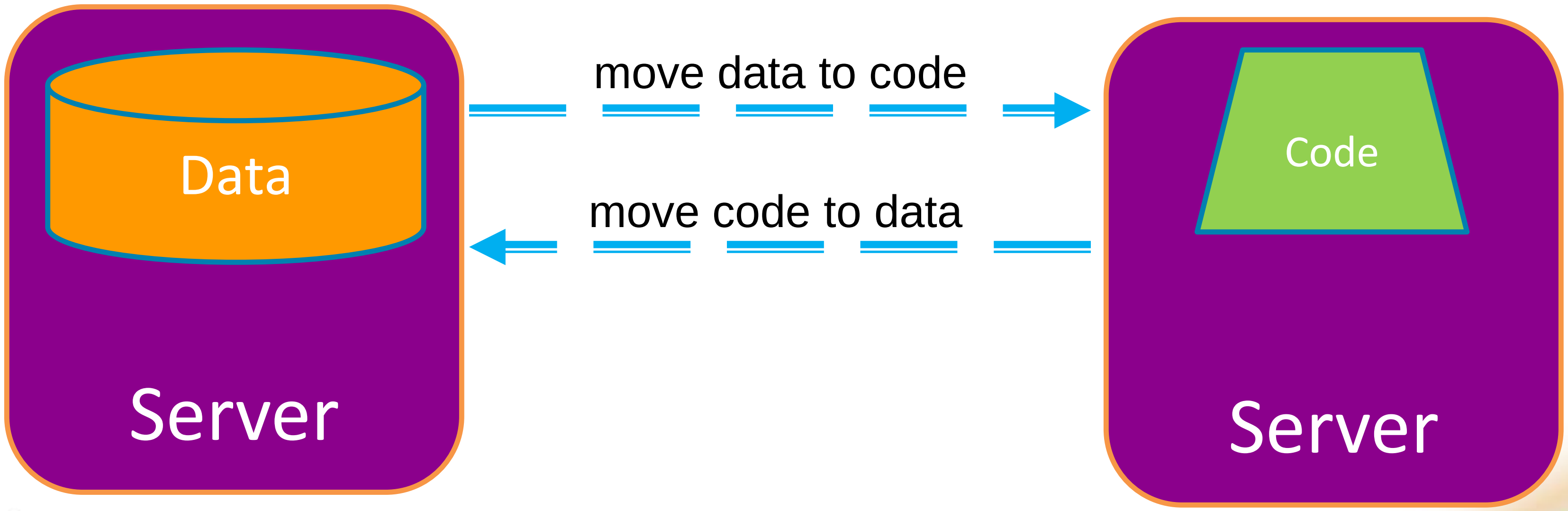
BIG DATA AND ECOSYSTEM TOOLS

Big Data

- Data size beyond systems capability
  - Terabyte, Petabyte, Exabyte
- Storage
  - Commodity servers, RAID, SAN
- Processing
  - In reasonable response time
  - A challenge here

Tradition processing tools

- Move what ?
  - the data to the code or
  - the code to the data



Traditional processing tools

- Traditional tools
  - RDBMS, DWH, BI
  - High cost
  - Difficult to scale beyond certain data size
    - price/performance skew
    - data variety not supported

Map-Reduce and NoSQL

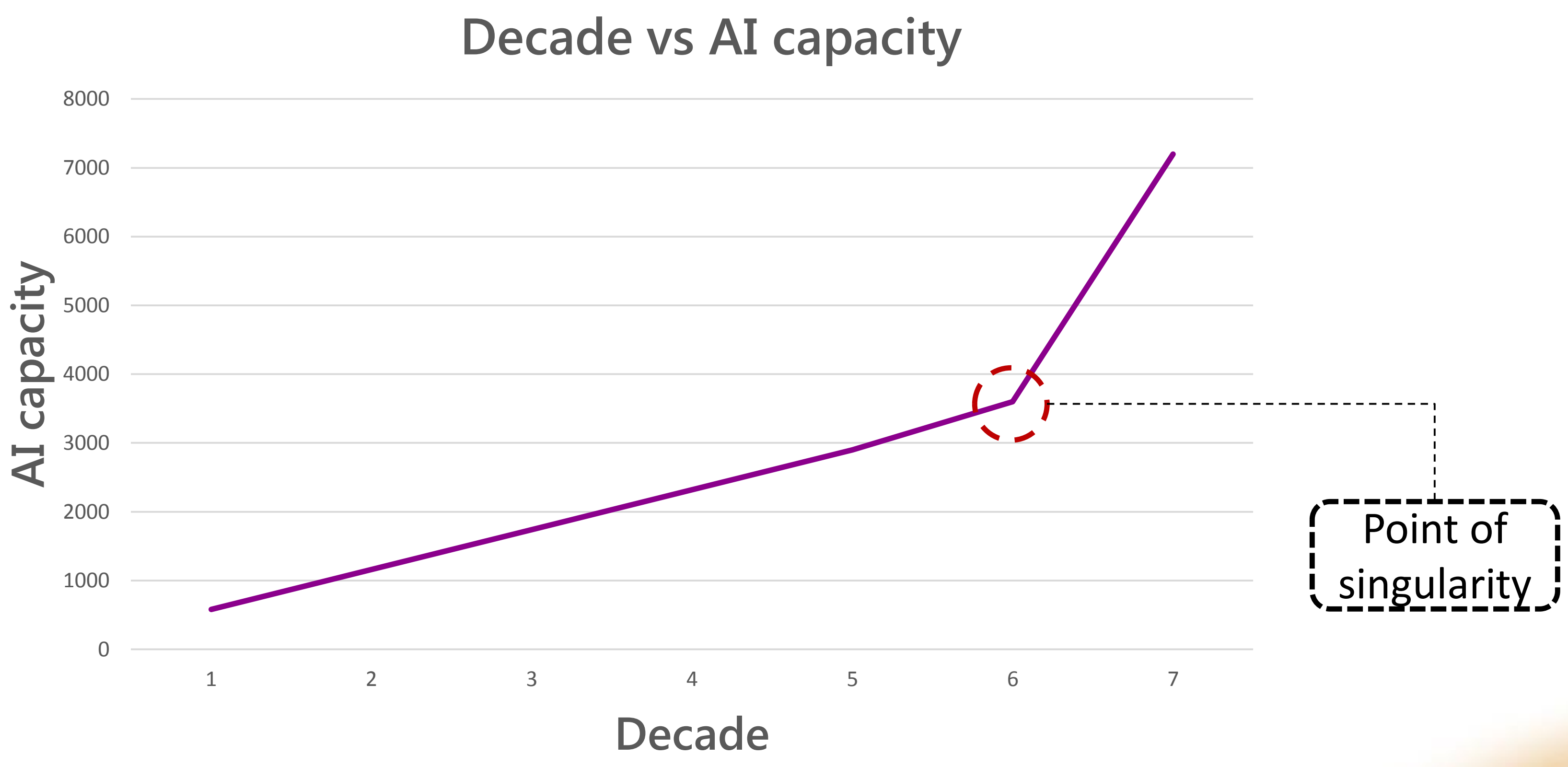
- Hadoop toolset
  - Free and open source
  - Commodity hardware
  - Scales to exabytes(10^{18}), maybe even more
- Not only SQL
  - Storage and query processing only
  - Complements Hadoop toolset
  - Volume, velocity and variety

All is well ?

- Hadoop was designed for batch processing
- Disk based processing: slow
- Many tools to enhance Hadoop’s capabilities
  - Distributed cache, Haloop, Hive, HBase
- Not for interactive and iterative

TOWARDS SINGULARITY

What is singularity ?



Technological singularity

- When AI capability exceeds Human capacity
- AI or non-AI singularity
- 2045: http://en.wikipedia.org/wiki/Ray_Kurzweil
  - The predicted year

APACHE SPARK

History of Spark

2009	2010	2013	2014	2015 March
Spark created by Matei Zaharia at UC Berkeley	Spark becomes open source.	Spark donated to Apache Software Foundation	Spark 1.0.0 released. 100TB sort achieved in 23 mins	Spark 1.3.1 released

Contributors in Spark

- Yahoo
- Intel
- UC Berkeley
- …
- 50+ organizations

Hadoop and Spark

- Spark complements the Hadoop ecosystem
- Replaces: Hadoop MR
- Spark integrates with
  - HDFS
  - Hive
  - HBase
  - YARN

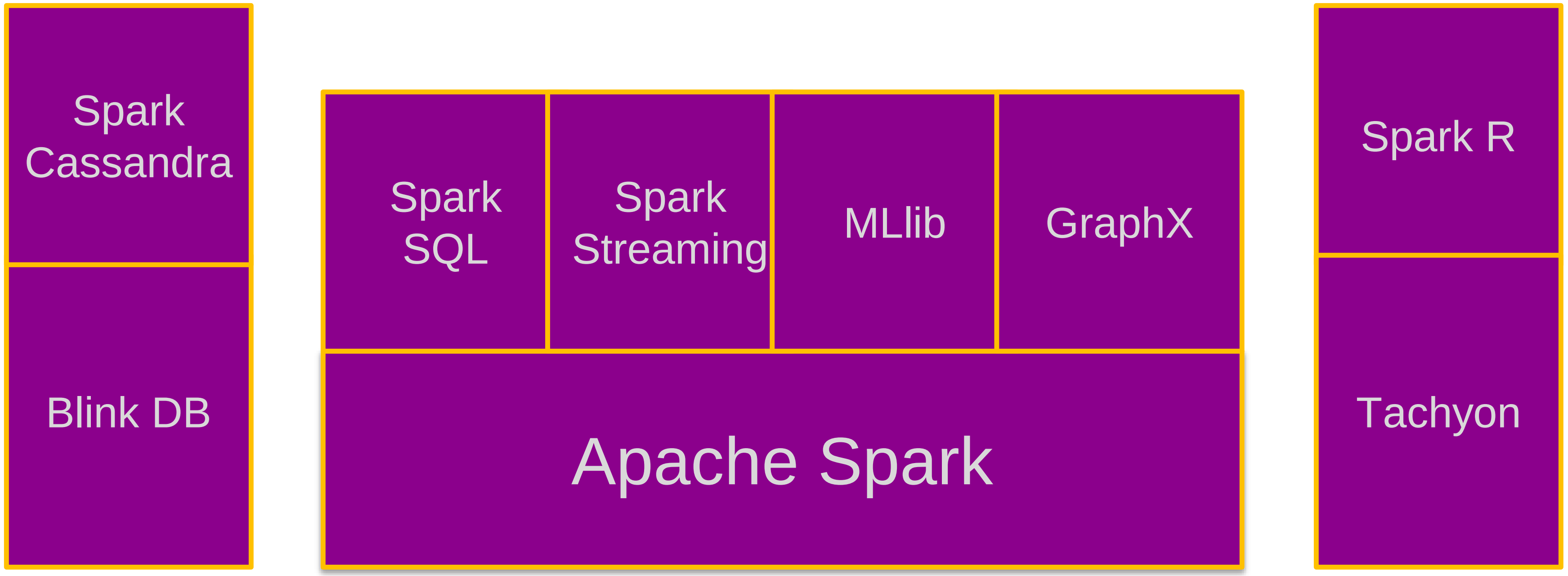
Other big data tools

- Spark also integrates with
  - Kafka
  - ZeroMQ
  - Cassandra
  - Mesos

Programming Spark

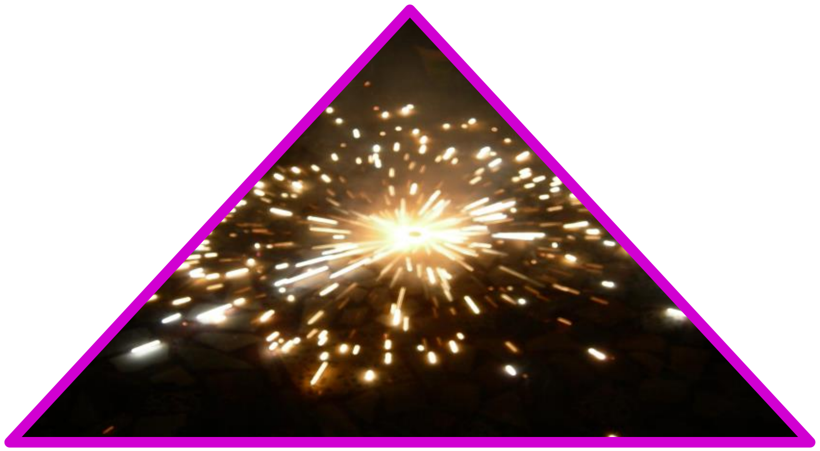
- Java
- Scala
- Python
- R

Spark toolset



What is Spark for ?

Batch

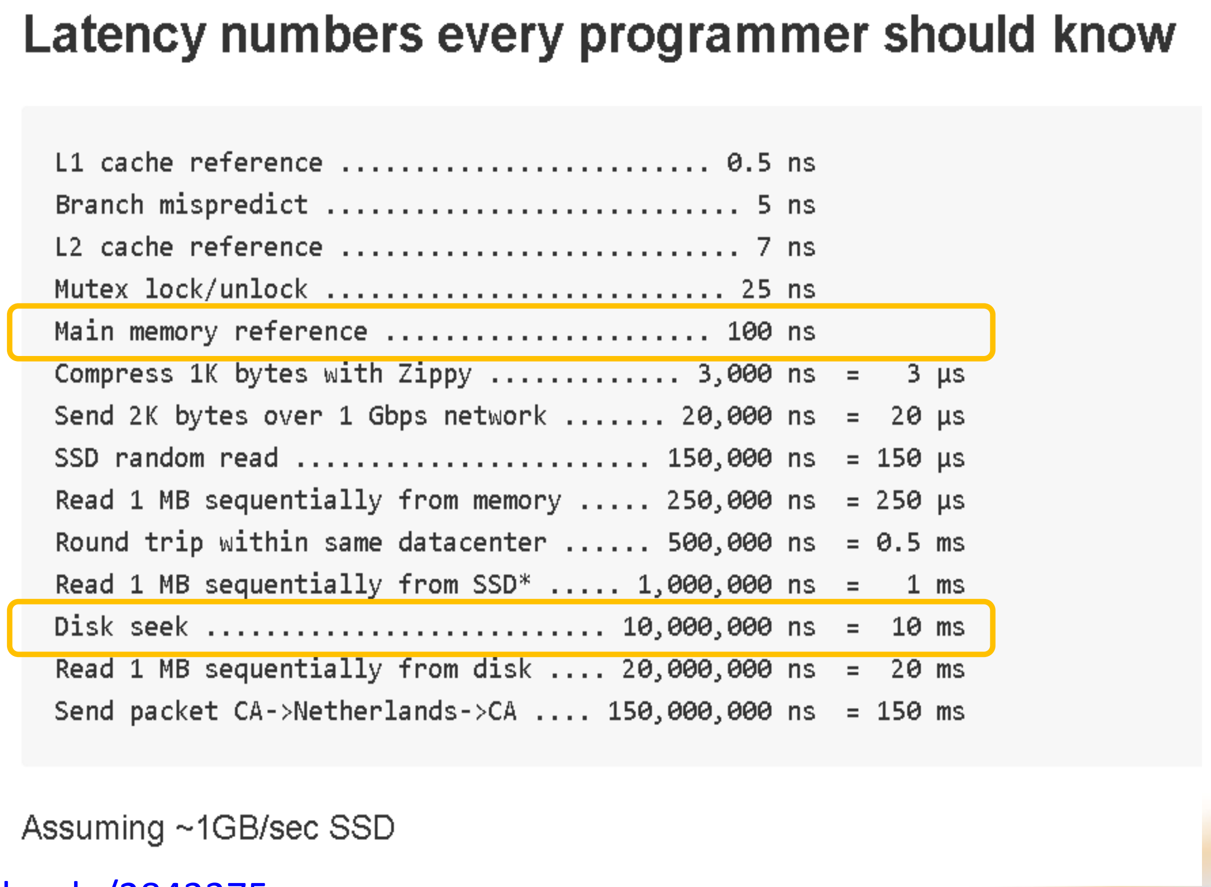


Interactive

Streaming

The main difference: speed

- RAM access vs Disk access
  - RAM access is 100,000 times faster !



Latency numbers every programmer should know

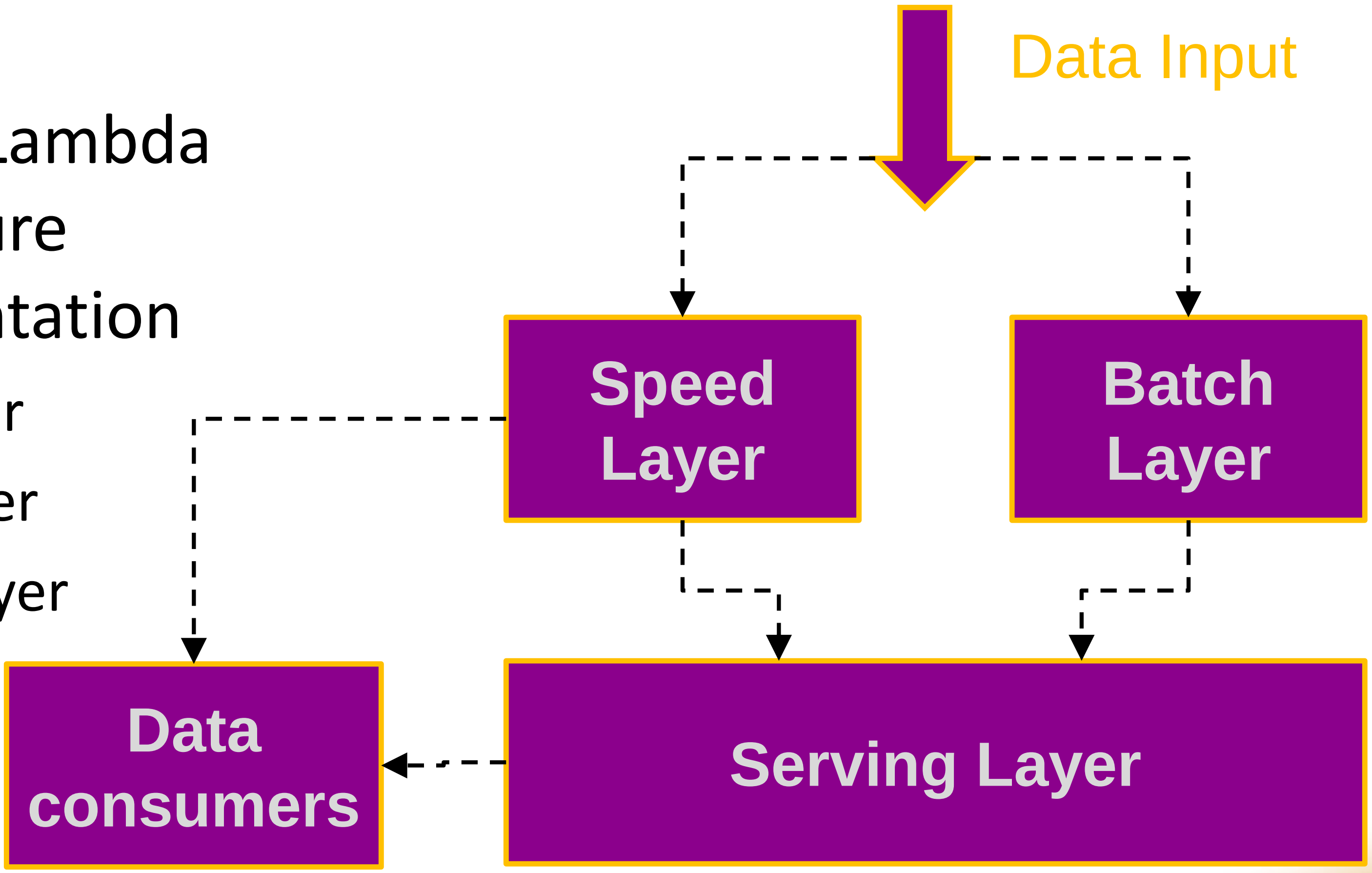
```
L1 cache reference ......................... 0.5 ns
Branch mispredict ............................ 5 ns
L2 cache reference ........................... 7 ns
Mutex lock/unlock ........................... 25 ns
Main memory reference ...................... 100 ns
Compress 1K bytes with Zippy ............. 3,000 ns  =   3 µs
Send 2K bytes over 1 Gbps network ....... 20,000 ns  =  20 µs
SSD random read ........................ 150,000 ns  = 150 µs
Read 1 MB sequentially from memory ..... 250,000 ns  = 250 µs
Round trip within same datacenter ...... 500,000 ns  = 0.5 ms
Read 1 MB sequentially from SSD* ..... 1,000,000 ns  =   1 ms
Disk seek ........................... 10,000,000 ns  =  10 ms
Read 1 MB sequentially from disk .... 20,000,000 ns  =  20 ms
Send packet CA->Netherlands->CA .... 150,000,000 ns  = 150 ms
```

Assuming ~1GB/sec SSD

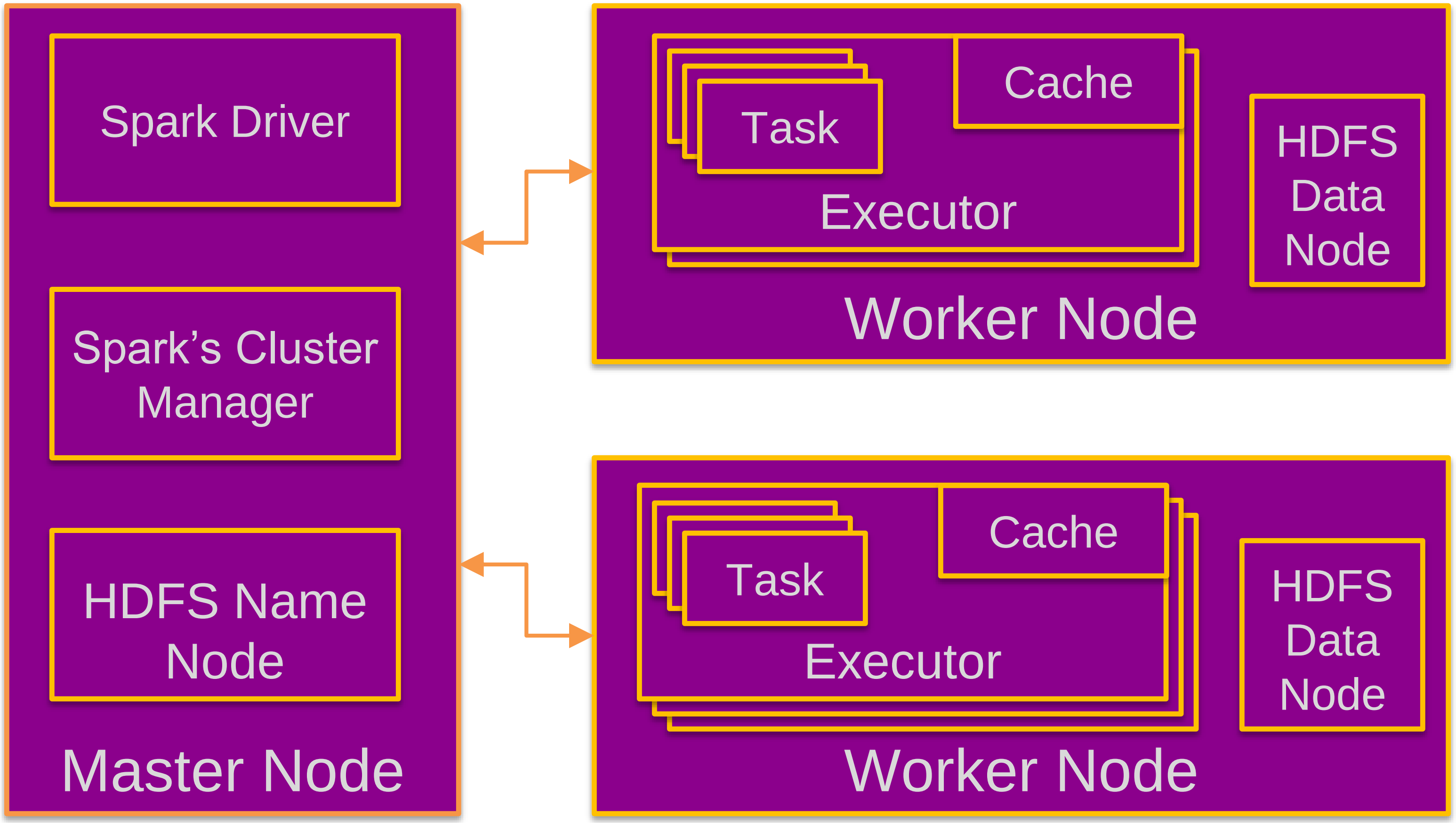
https://gist.github.com/hellerbarde/2843375

Lambda Architecture pattern

- Used for Lambda architecture implementation
  - Batch layer
  - Speed layer
  - Serving layer



Deployment Architecture



Core features of Spark

- Rich API
- RDD: Resilient Distributed Datasets
- DAG based execution
- Data caching
- Strong ecosystem tool support

Sample code in scala

- Find the top 100,000 Wikipedia pages by page views
- Log file format: code, title, num_hits
- enPages.map(l => l.split(" "))

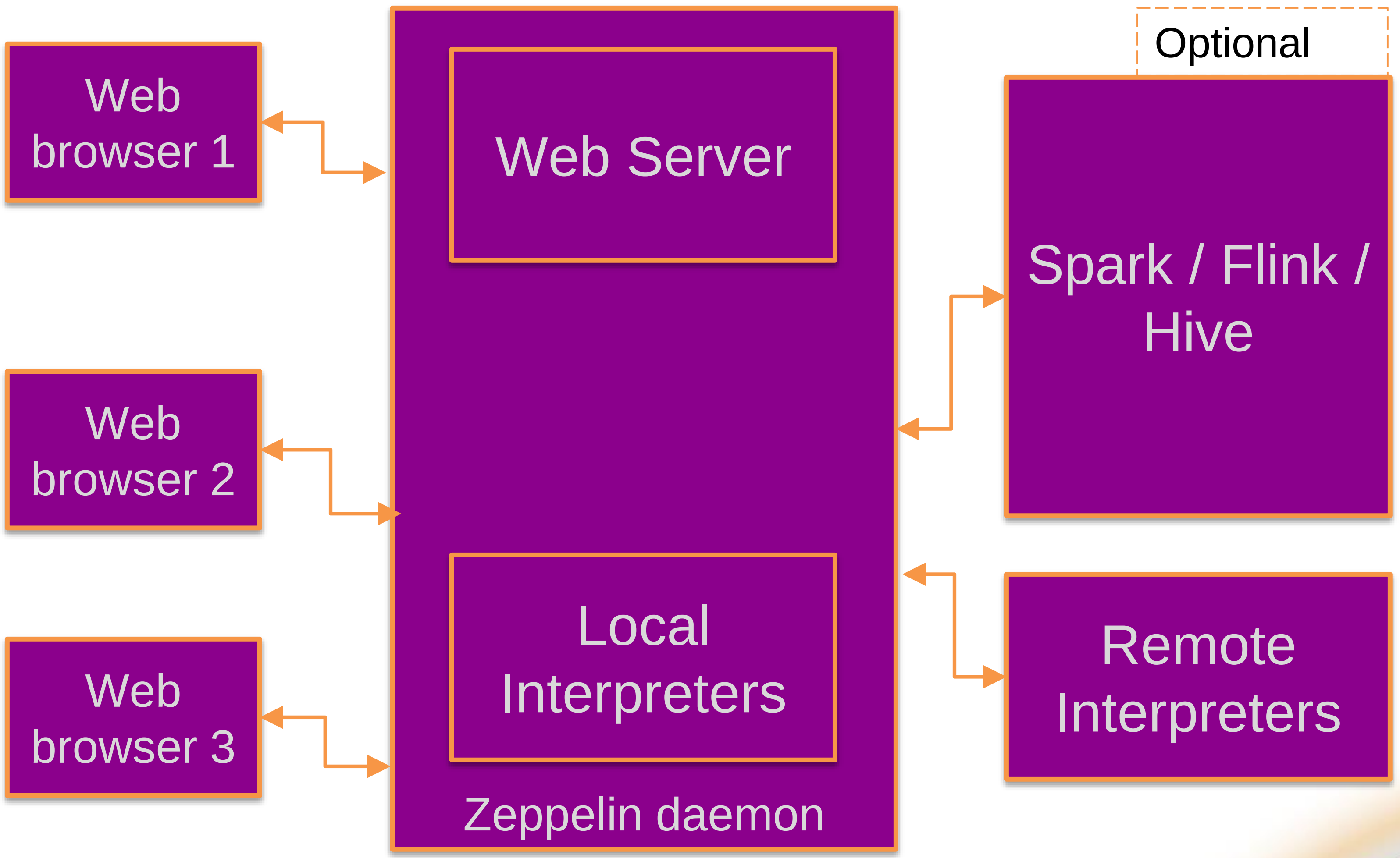
```
enPages.map(l => l.split(" "))
           .map(l => (l(1), l(2).toInt))
           .reduceByKey(_+_, 200)
           .filter(x => x._2 > 100000)
           .map(x => (x._2, x._1))
           .collect
           .foreach(println)
```

APACHE ZEPPELIN

Interactive data analytics

- For Spark and Flink
- Web front end
- At the back end, it connects to
  - SQL systems(Eg: Hive)
  - Spark
  - Flink

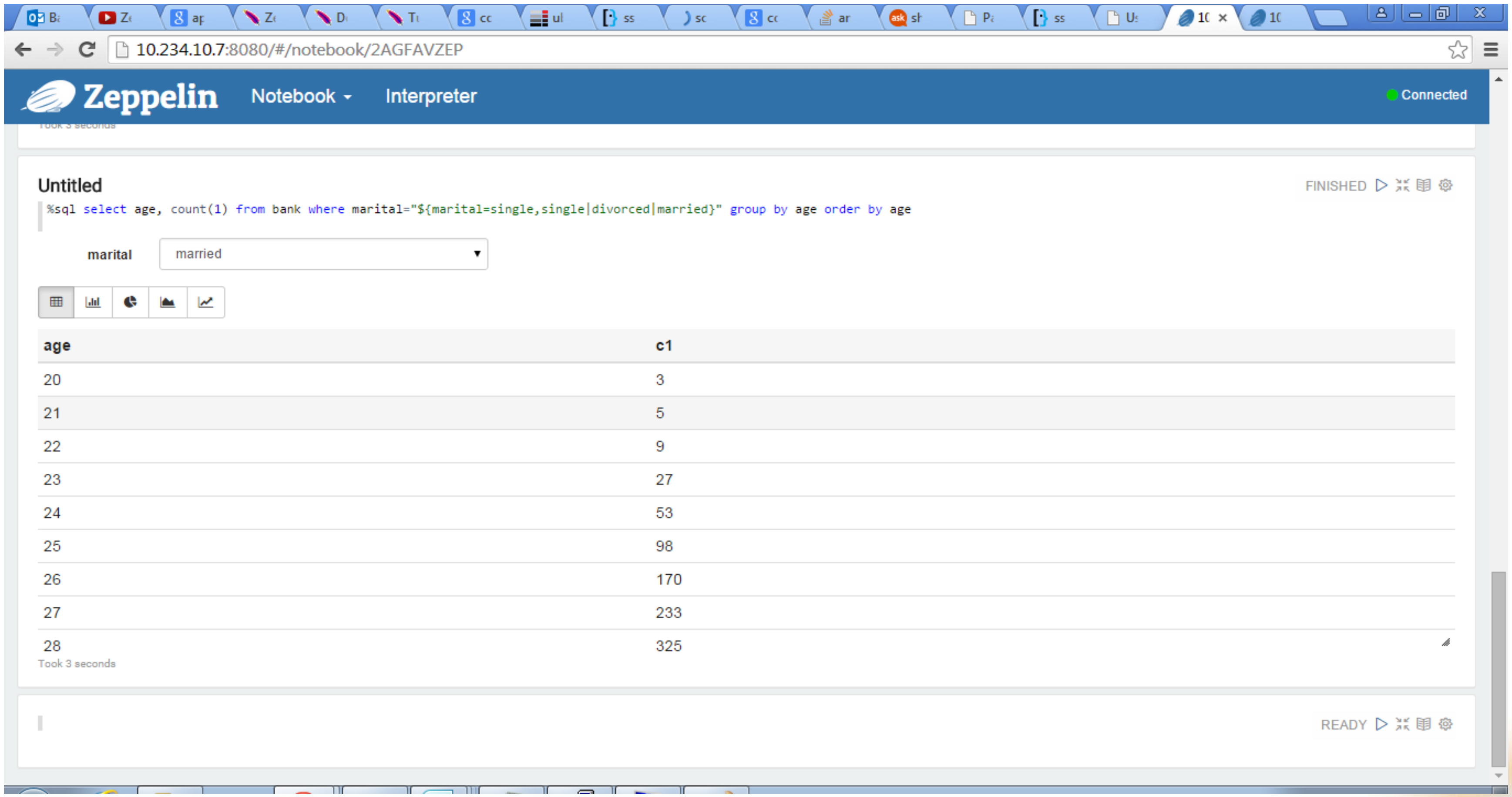
Deployment Architecture



Notebook

- Is where you do your data analysis
- Web UI REPL with pluggable interpreters
- Interpreters
  - Scala, Python, Angular, SparkSQL, Markdown and Shell

Notebook:view



Untitled

```
%sql select age, count(1) from bank where marital="${marital=single,single|divorced|married}" group by age order by age
```

marital: married

age	c1
20	3
21	5
22	9
23	27
24	53
25	98
26	170
27	233
28	325

Took 3 seconds

saltmarch MEDIA

GREAT INDIAN DEVELOPER SUMMIT

www.developersummit.com

User Interface features

- Markdown
- Dynamic HTML generation
- Dynamic chart generation
- Screen sharing via websockets

SQL Interpreter

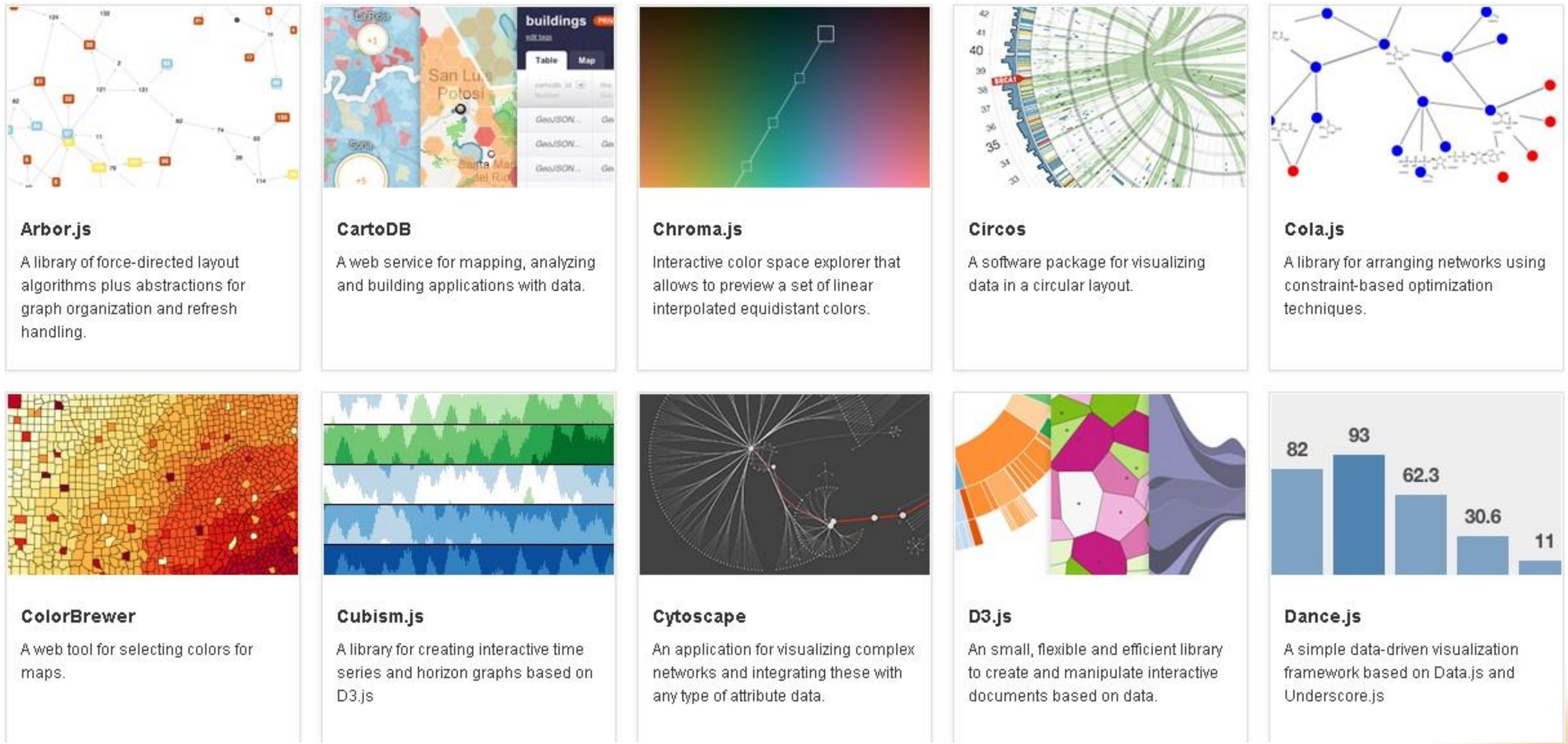
- SQL shell
  - Query spark data using SQL queries
  - Return normal text, HTML or chart type results

Scala interpreter for Spark

- Similar to the Spark shell
- Upload your data into Spark
- Query the data sets(RDDs) in your Spark server
- Execute map-reduce tasks
- Actions on RDD
- Transformations on RDD

DATA VISUALIZATION

Visualization tools



Arbor.js

A library of force-directed layout algorithms plus abstractions for graph organization and refresh handling.

CartoDB

A web service for mapping, analyzing and building applications with data.

Chroma.js

Interactive color space explorer that allows to preview a set of linear interpolated equidistant colors.

Circos

A software package for visualizing data in a circular layout.

Cola.js

A library for arranging networks using constraint-based optimization techniques.

ColorBrewer

A web tool for selecting colors for maps.

Cubism.js

A library for creating interactive time series and horizon graphs based on D3.js

Cytoscape

An application for visualizing complex networks and integrating these with any type of attribute data.

D3.js

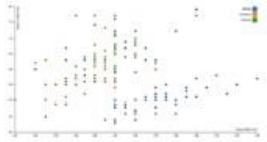
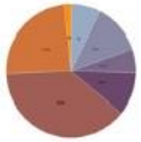
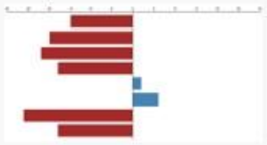
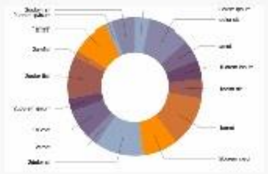
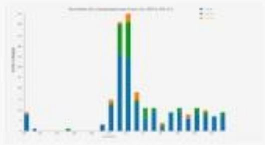
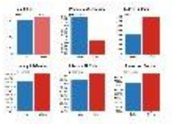
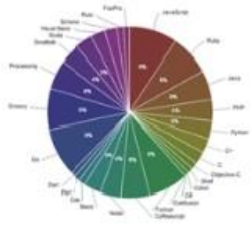
An small, flexible and efficient library to create and manipulate interactive documents based on data.

Dance.js

A simple data-driven visualization framework based on Data.js and Underscore.js

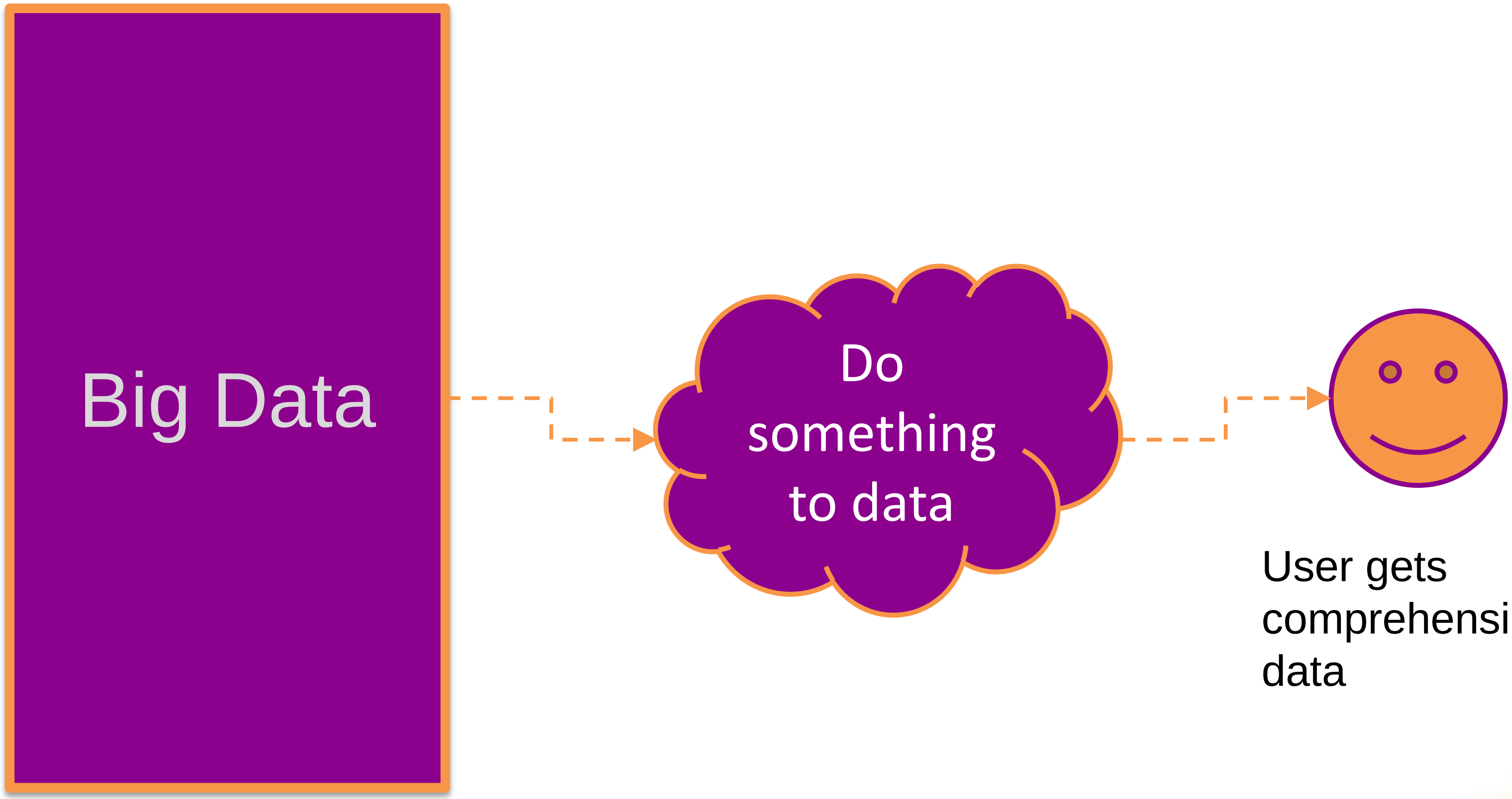
Source: *http://selection.datavisualization.ch/*

D3 Visualizations

Grouped Bar Chart	Scatterplot 	Donut Chart	Pie Chart 
Donut Multiples	Bar Chart with Negative Values 	Animated Donut Chart with Labels 	Stacked Bar Charts on time scale 
Bar Chart Multiples 	d3pie - pie chart generator and lib 	3D Donut	Gradient Pie

[Source: https://github.com/mbostock/d3/wiki/Gallery](https://github.com/mbostock/d3/wiki/Gallery)

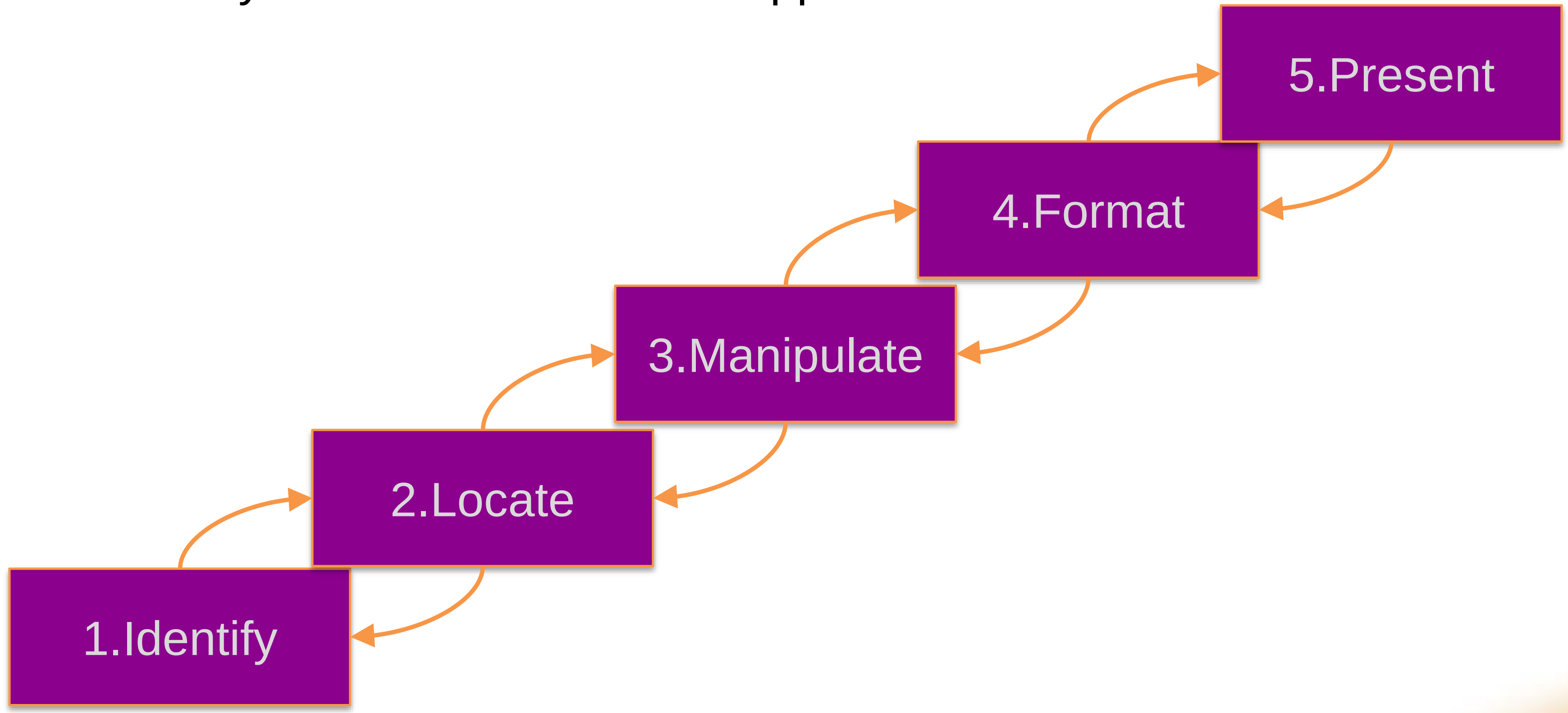
The need for visualization



User gets comprehensible data

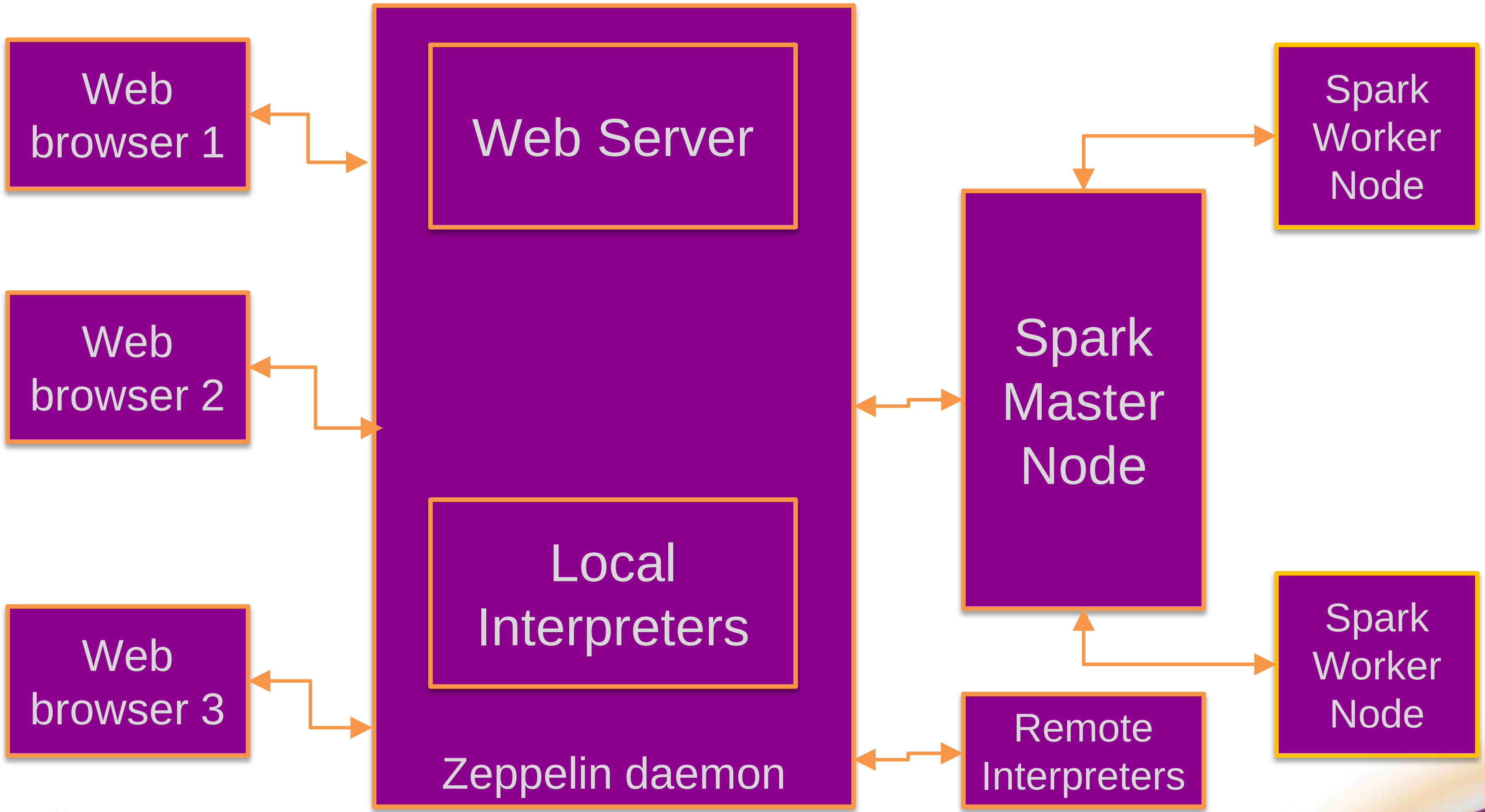
Tools for Data Presentation Architecture

A data analysis tool/toolset would support:

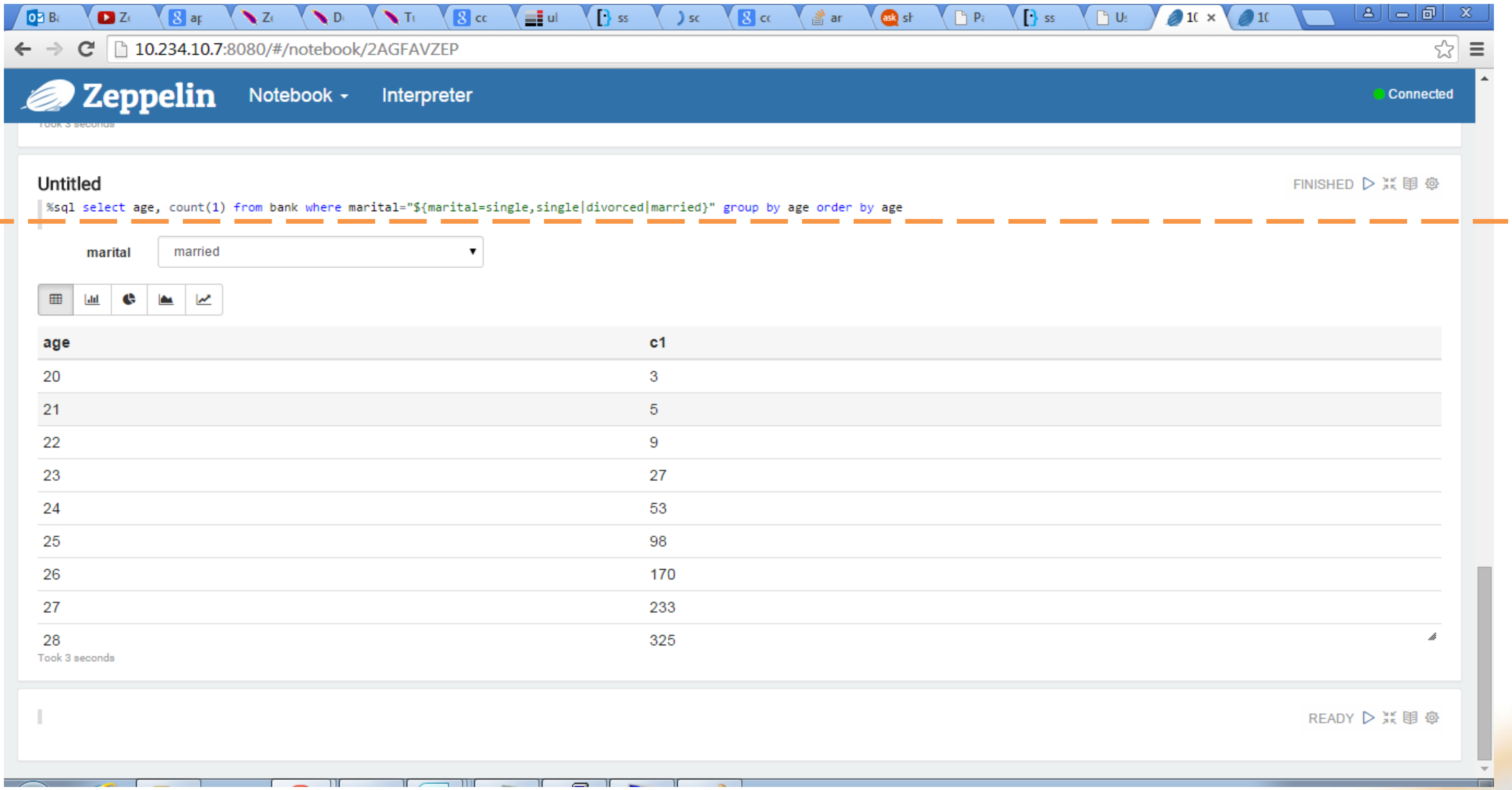


COMBINING SPARK AND ZEPPELIN

Spark and Zeppelin



Zeppelin views: Table from SQL



10.234.10.7:8080/#/notebook/2AGFAVZEP

Untitled

```
%sql select age, count(1) from bank where marital="${marital=single,single|divorced|married}" group by age order by age
```

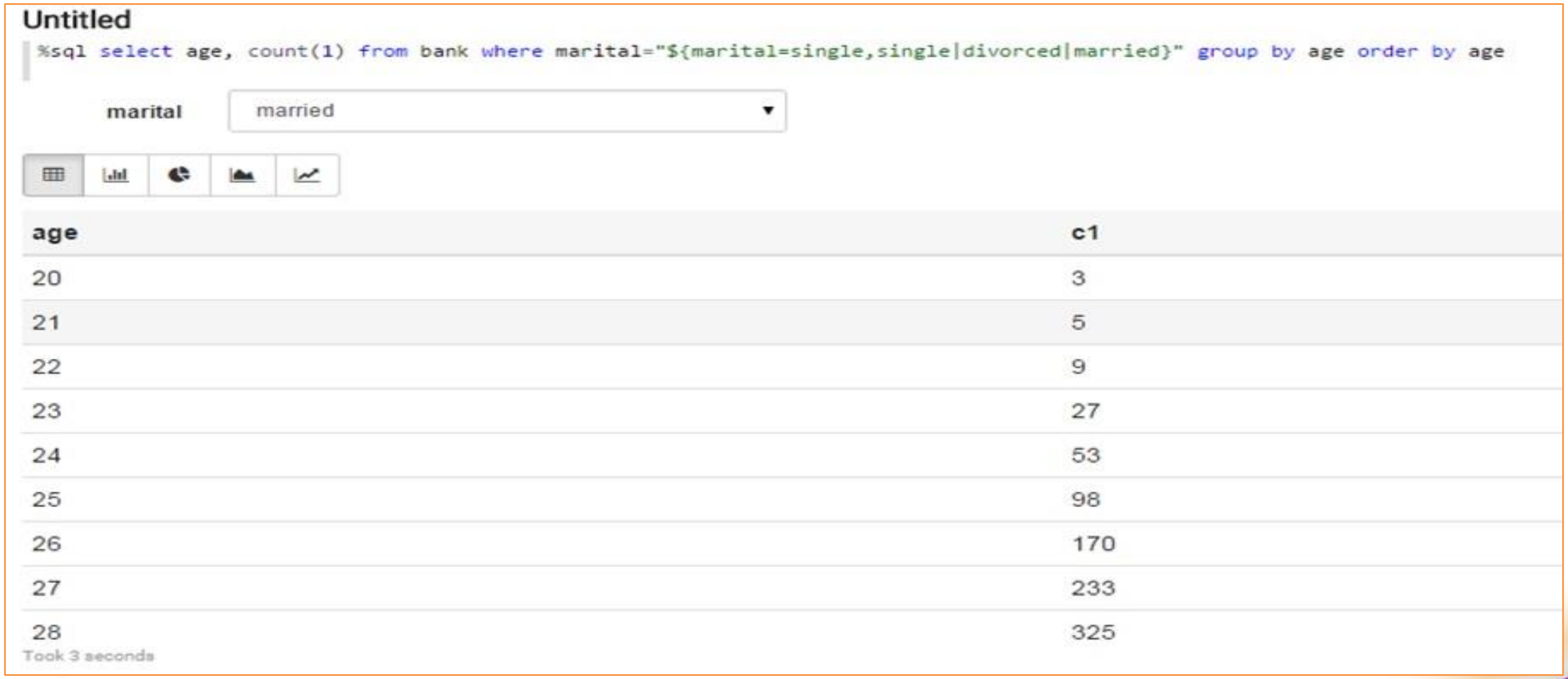
marital: married

age	c1
20	3
21	5
22	9
23	27
24	53
25	98
26	170
27	233
28	325

Took 3 seconds

Zeppelin views: Table from SQL

```
%sql select age, count(1) from bank where
        marital="${marital=single,single|divorced|married}"
        group by age order by age
```



Untitled

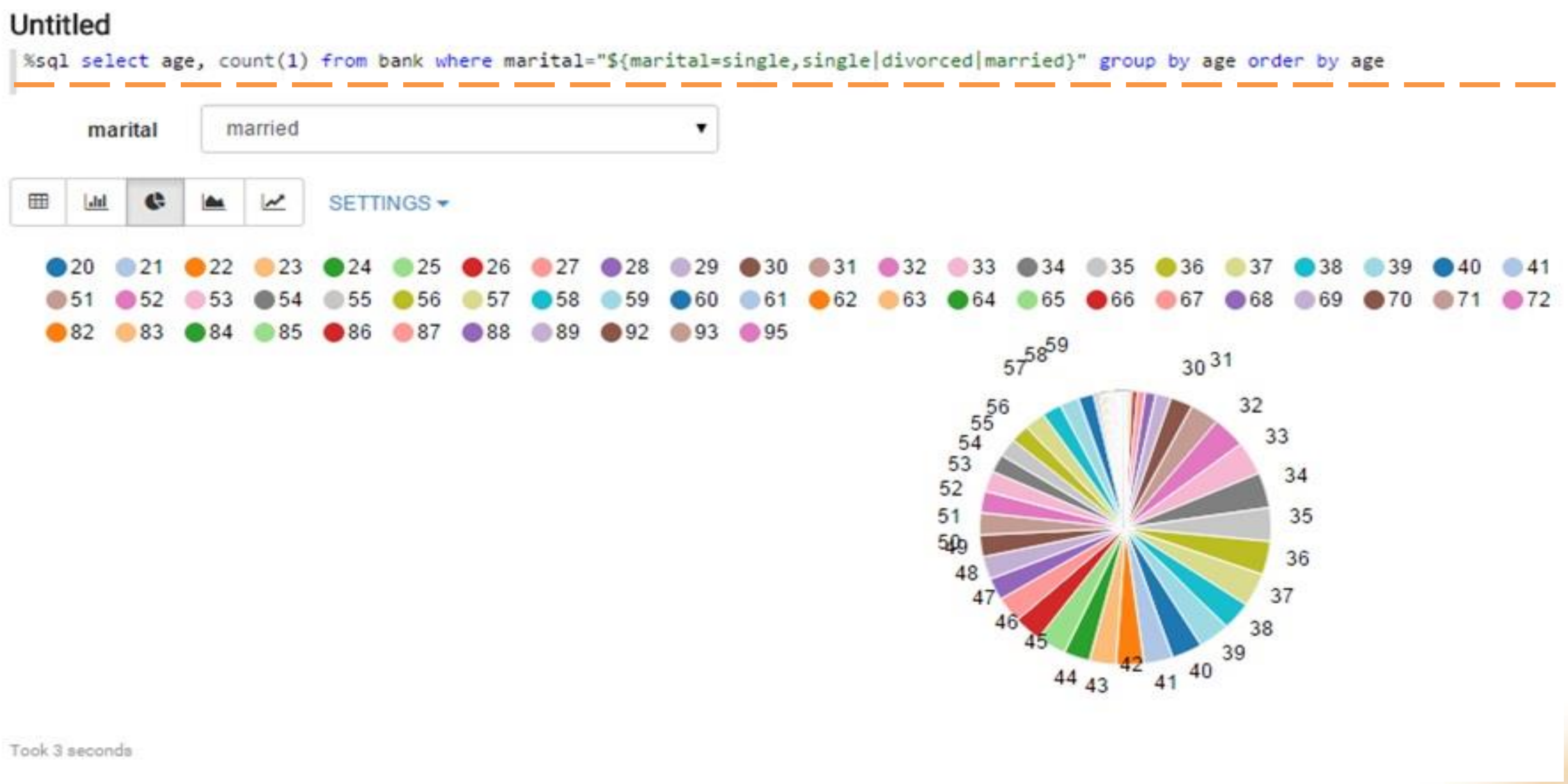
```
%sql select age, count(1) from bank where marital="${marital=single,single|divorced|married}" group by age order by age
```

marital: married

age	c1
20	3
21	5
22	9
23	27
24	53
25	98
26	170
27	233
28	325

Took 3 seconds

Zeppelin views: Pie chart from SQL



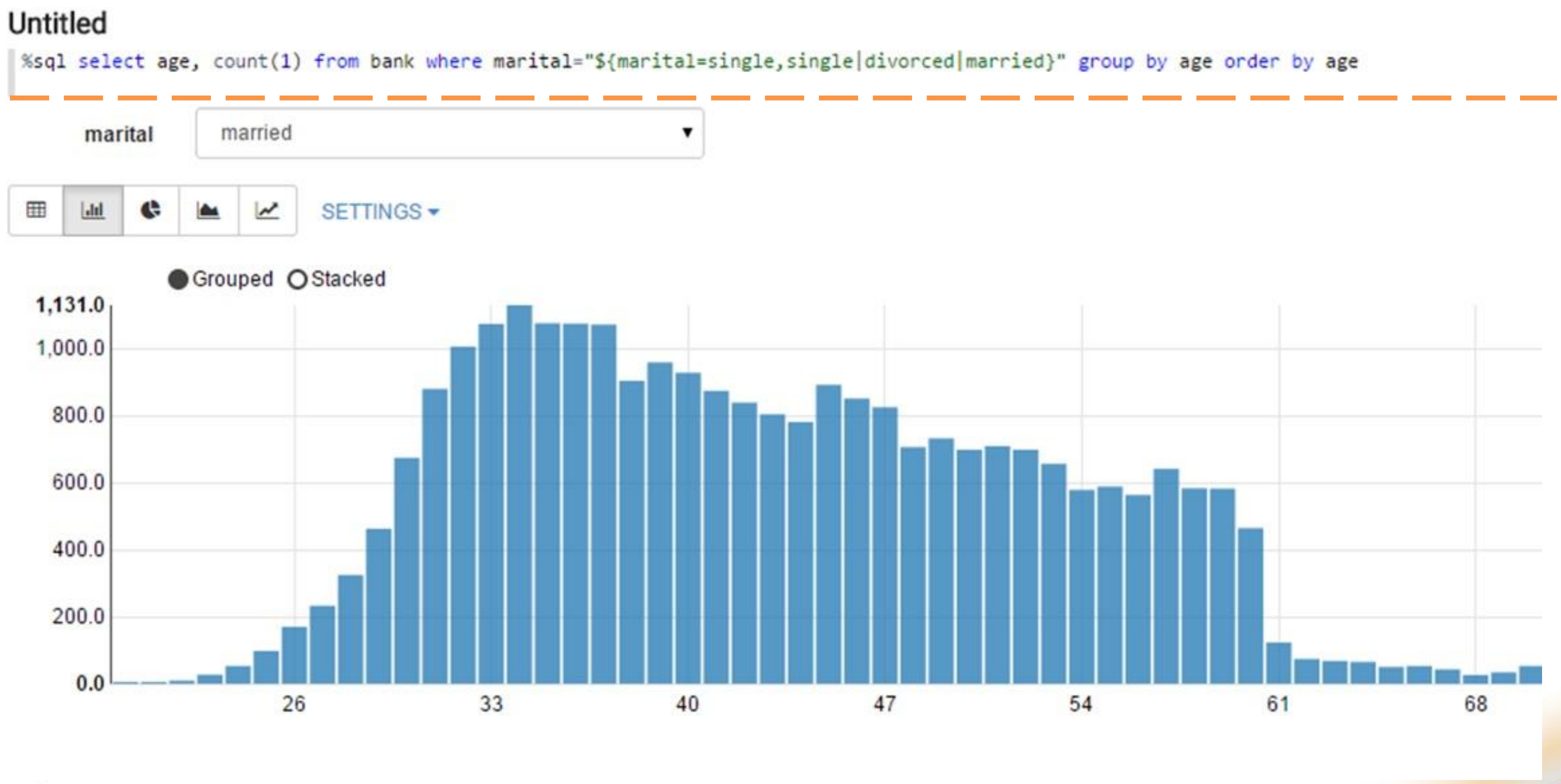
Untitled

```
%sql select age, count(1) from bank where marital="${marital=single,single|divorced|married}" group by age order by age
```

marital: married

Took 3 seconds

Zeppelin views: Bar chart from SQL



Untitled

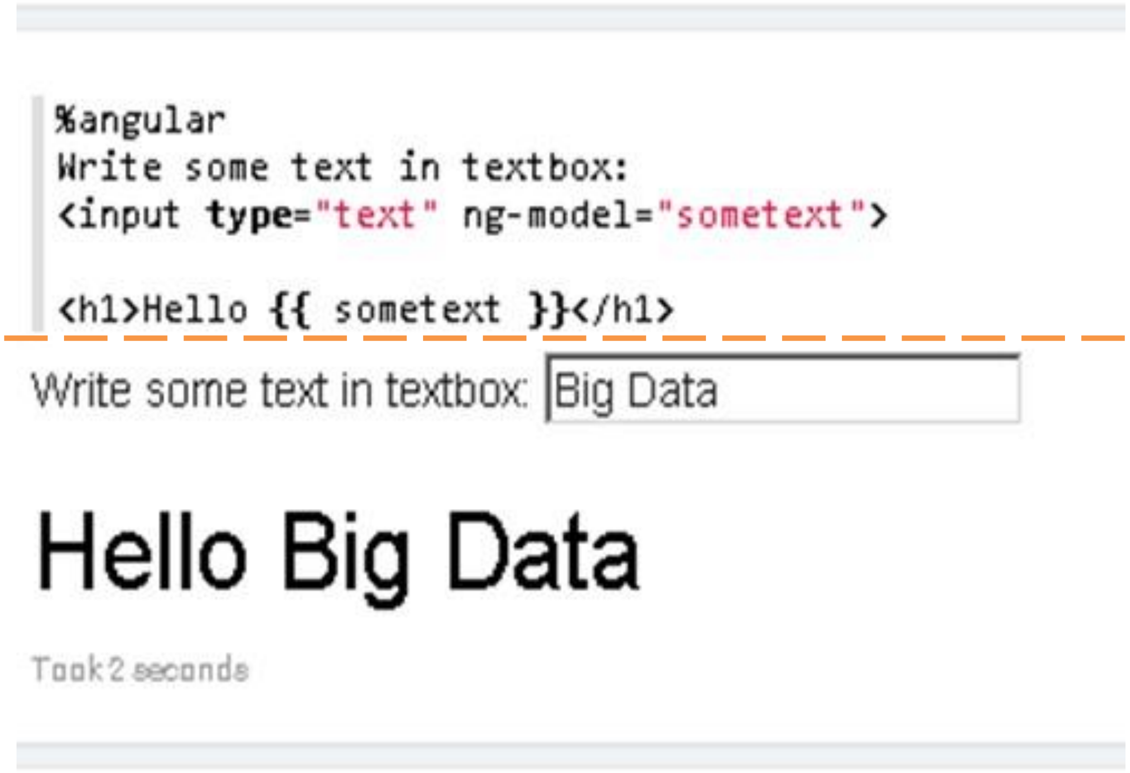
```
%sql select age, count(1) from bank where marital="${marital=single,single|divorced|married}" group by age order by age
```

marital: married

SETTINGS

● Grouped ○ Stacked

Zeppelin views: Angular



```
%angular
Write some text in textbox:
<input type="text" ng-model="sometext">

<h1>Hello {{ sometext }}</h1>
```

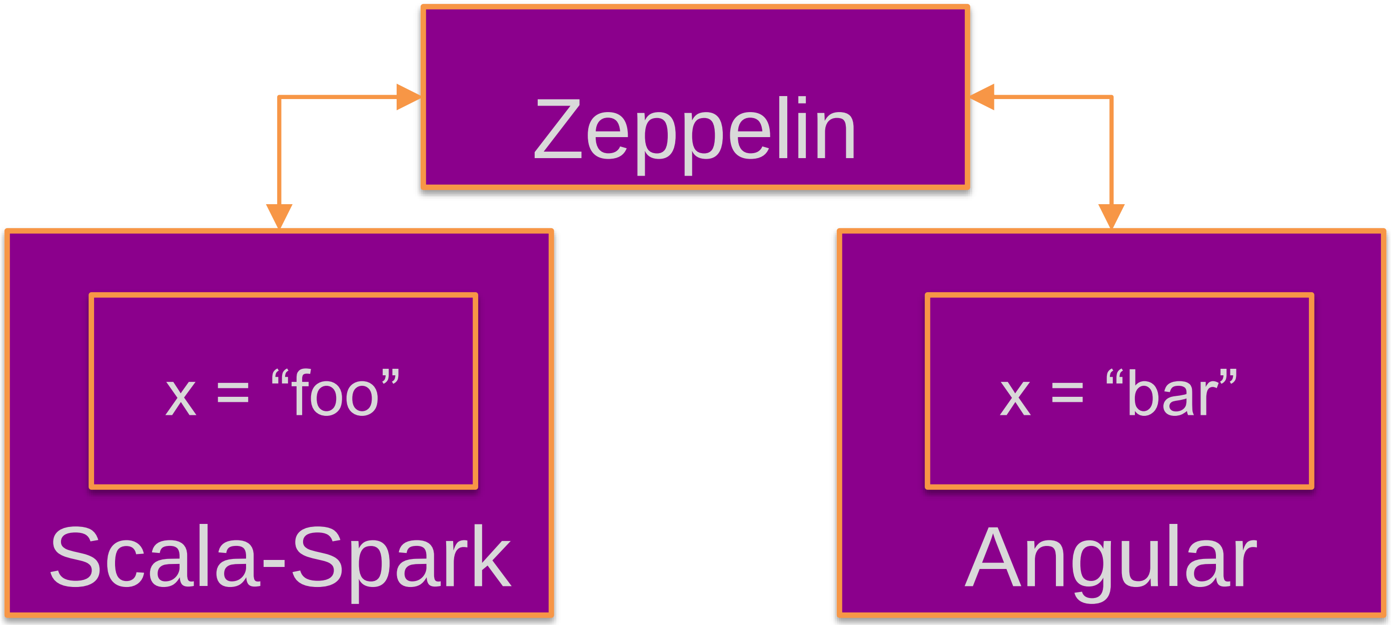
Write some text in textbox: Big Data

Hello Big Data

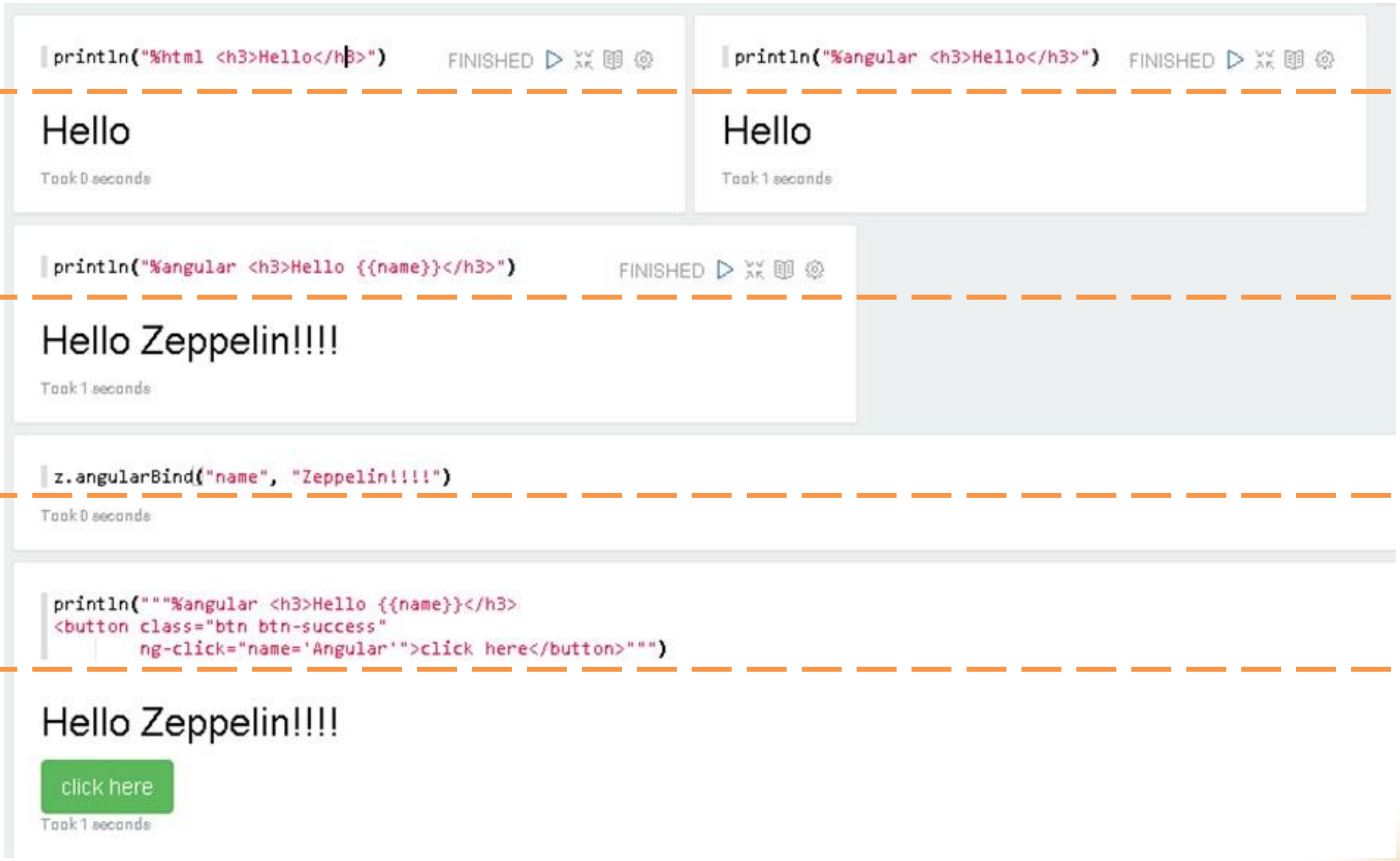
Took 2 seconds

Share variables: MVVM

- Between Scala/Python/Spark and Angular
- Observe scala variables from angular



Zeppelin views: Angular-scala binding



```
println("%html <h3>Hello</h3>")
```
FINISHED

Hello

Took 0 seconds

```
println("%angular <h3>Hello</h3>")
```
FINISHED

Hello

Took 1 seconds

```
println("%angular <h3>Hello {{name}}</h3>")
```
FINISHED

Hello Zeppelin!!!!

Took 1 seconds

```
z.angularBind("name", "Zeppelin!!!!")
```
Took 0 seconds

```
println("""%angular <h3>Hello {{name}}</h3>
<button class="btn btn-success"
        ng-click="name='Angular'">click here</button>""")
```

Hello Zeppelin!!!!

Took 1 seconds

Screen sharing using Zeppelin

- Share your graphical reports
  - Live sharing
  - Get the share URL from zeppelin and share with others
  - Uses websockets
- Embed live reports in web pages

FUTURE

Spark and Zeppelin

- Spark
  - Berkeley Data Analytics Stack
  - More source and sinks; SparkSQL
- Zeppelin
  - Notebooks for
    - Machine Learning using Spark
    - GraphX and Mllib
  - Additional interpreters
  - Better graphics, steaming views
  - Report persistence
  - More report templates
  - Better angular integration

SUMMARY

Summary

- Spark and tools
- The need for visualization
- The role of Zeppelin
- Zeppelin – Spark integration

Thank you!

@prajods

saltmarch

GREAT INDIAN DEVELOPER SUMMIT