

Putting A Spark in Web Apps

Apache Big Data, Seville ES, 2016

David Fallside

Intro

- Web app development has moved from Java etc to Node.js and JavaScript
- JS environment relatively simple and very rich, e.g. “npm install” provides access to world’s largest OSS repo
- Node.js environment scales very well “horizontally” but large compute requirements offloaded to back-end engines
- Apache Spark is an ideal engine:
 - Optimized for multiple compute tasks: coding, SQL, streaming, ML, graph manipulation
 - All available in single environment, reduces TCO
 - Fast: parallelised operators and in-memory policies, scalable from laptop to large distributed clusters
- Native APIs for Scala, Java, Python, (R) but not JavaScript and so EclairJS

Web App Requirements

- Simplicity
 - “npm install eclairjs”
 - Familiar coding style
 - Delineation between client and server
- Enable Spark functionality: SQL, streaming, ML, graph
- Performance
 - Intrinsic (JVM, in/out-proc)
 - Support interactive applications

Code Demonstration

Code Semantics

```
var rdd = sparkSession.read().textFile(file).rdd();
```

```
var rdd2 = rdd.flatMap(function(sentence) {  
    return sentence.split(" ");  
});
```

```
var rdd3 = rdd2.filter(function(word) {  
    return word.trim().length > 0;  
});
```

```
var rdd4 = rdd3.mapToPair(function(word, Tuple2) {  
    return new Tuple2(word.toLowerCase(), 1);  
}, [spark.Tuple2]);
```

```
var rdd5 = rdd4.reduceByKey(function(value1, value2) {  
    return value1 + value2;  
});
```

```
var rdd6 = rdd5.mapToPair(function(tuple, Tuple2) {  
    return new Tuple2(tuple._2() + 0.0, tuple._1());  
}, [spark.Tuple2]);
```

```
var rdd7 = rdd6.sortByKey(false);
```

```
rdd7.take(10).then(function(val) {  
    console.log("Success:", val);  
    stop();  
}).catch(stop);
```

immutable

Spark operator

lambda
function

action vs
transformation

Spark class,
passed arg

EclairJS & Web App

Browser Script

```
function clickme() {  
  if (ws) {  
    // First clear old results then start count  
    ws.send(JSON.stringify({startCount: true}));  
    ...  
  }  
  
  window.onload = function() {  
    ws = new WebSocket("ws://" + URIPort);  
  
    // On msg from Node, display the results  
    ws.onmessage = function(e) {  
      if (e.data) {  
        var data = JSON.parse(e.data);  
        data.forEach(function(item) {  
          // display data on screen ...  
        });  
      }  
    };  
  };  
}
```

```
var wss = new WebSocketServer({  
  server: server  
});
```

```
wss.on('connection', function(ws) {  
  ws.on('message', function(message) {  
    var msg = JSON.parse(message);  
    if (msg && msg.startCount) {  
      startCount();  
    } ...  
  });  
});
```

```
var wcount = require('./wcount.js');
```

```
function startCount() {  
  var file = './data/dream.txt';  
  wcount.start(file, function(rawdata) {  
    var results = [];  
    rawdata.forEach(function(result) {  
      results.push({count: result[0],  
                    word: result[1]});  
    });  
    wss.clients.forEach(function(client) {  
      try {  
        // Send results to browser  
        client.send(JSON.stringify(results));  
      } ...  
    });  
  });  
}
```

```
// Wrapper class  
function WCount() {  
}  
  
WCount.prototype.start = function(file, callback) {  
  startWCount(file, callback);  
}  
  
function startWCount(file, callback) {  
  var rdd = sparkSession.read().textFile(file).rdd();  
  
  var rdd2 = rdd.flatMap(function(sentence) {  
    return sentence.split(" ");  
  });  
  // etc ...  
  
  rdd7.take(10).then(function(val) {  
    callback(val);  
  }) ...  
}
```

1

4

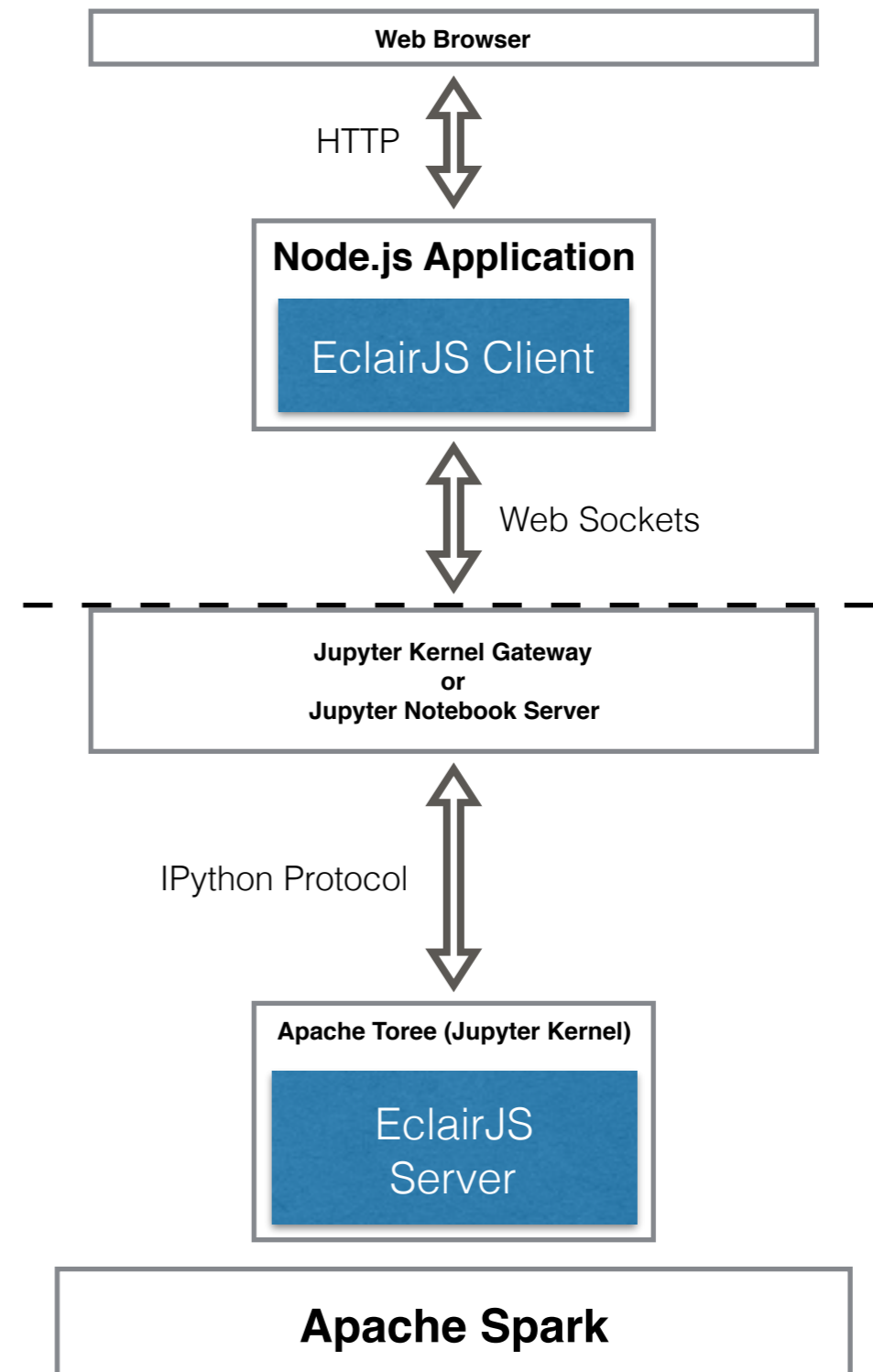
2

3

Web Application Demonstration

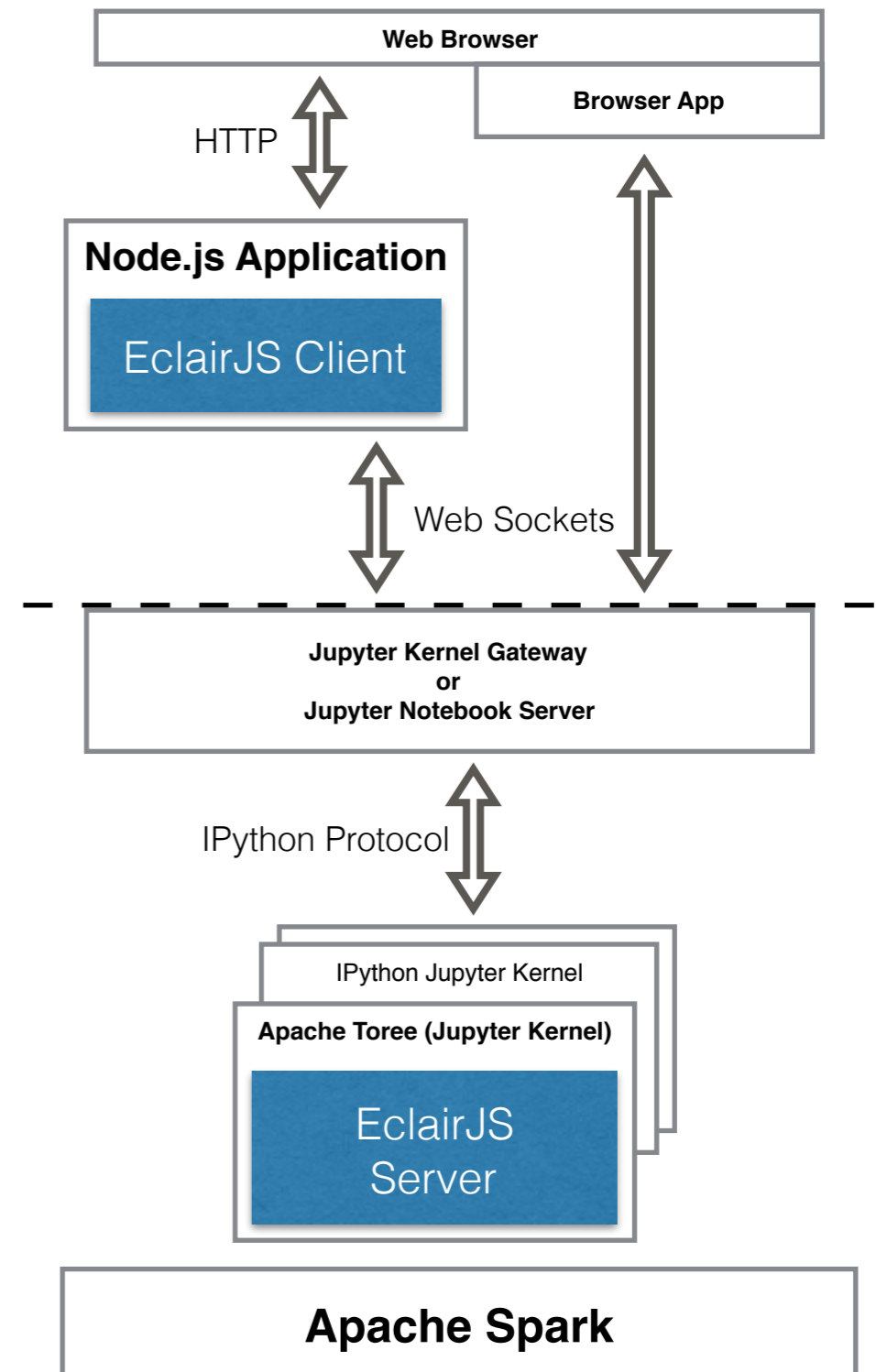
Architectural Detour

- EclairJS components
 - Client: JS API, “npm install” target
 - Server: JS - Java mappings, JVM Nashorn (v8)
- Jupyter infrastructure
 - Transport for interactive applications
 - Multi-channel IPython Protocol includes TCP- streaming support
 - JKG multiplexes WS - IPyP



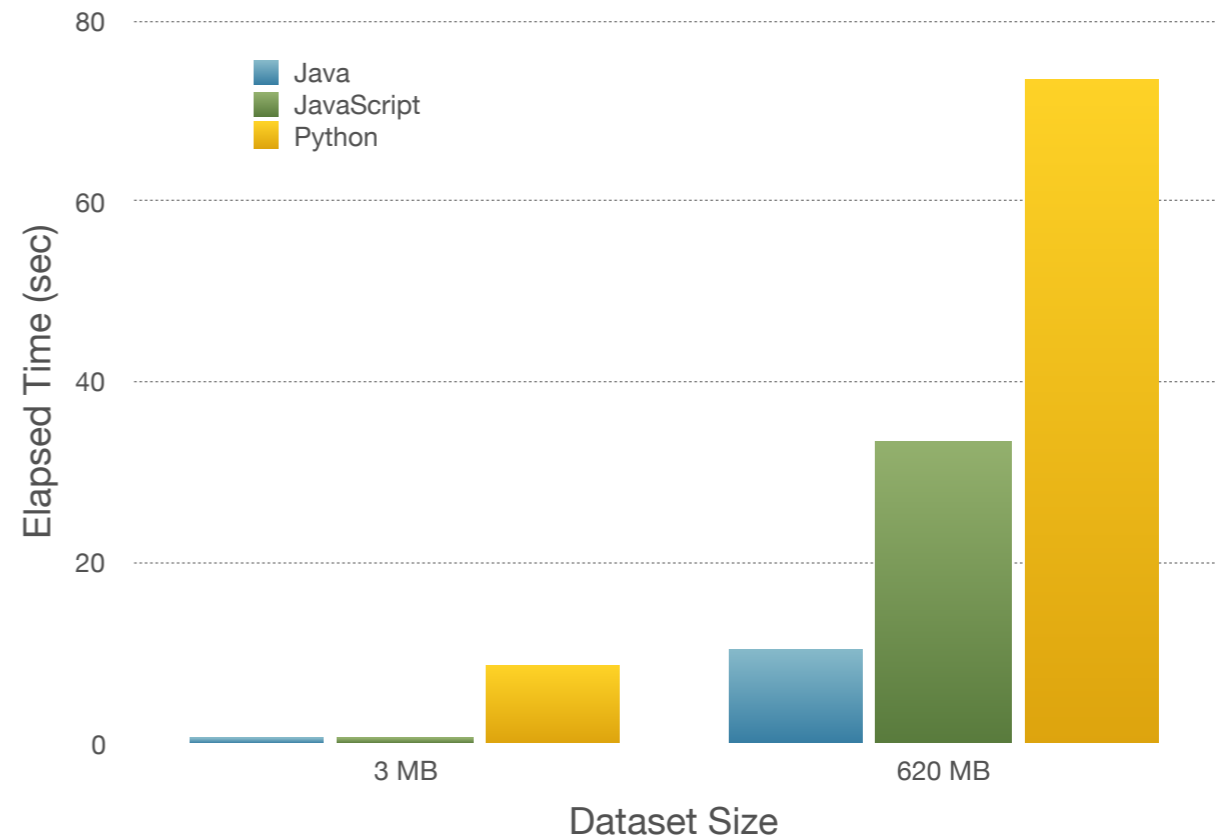
Deployment

- Server deployment expected for EclairJS Server
 - JKG + Toree + EJS Server + Spark
- E.g. IBM Bluemix: Node.js SDK, IBM Analytics for Apache Spark, Spark deployed on SoftLayer
- Jupyter for Notebooks + EclairJS = JS Notebooks



Performance

- Simple test based on MovieLens datasets
 - read, filter, map, count
 - 3MB/100k and 620MB/24m
 - Spark running local mode
- Java < JavaScript << Python
 - Difference maintained across test set sizes
 - Java and PySpark API's built-in



In Closing

- EclairJS: Simplicity, Performance & Spark Functionality
- Open source on GitHub, see eclairjs.com
- Project is active and looking for collaborators
- Gracias, merci, thank-you!