MOBIUS: C# BINDING FOR SPARK

Kaarthik Sivashanmugam
Microsoft
@kaarthikss

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Quick Background

- Business Scenario: Next-gen near real-time processing of Bing.com logs
  - Size of raw logs: TBs per hour
  - C# library for processing ~ in use for several years

- Yesterday's talk "Five Lessons Learned in Building Streaming Applications at Microsoft Bing Scale" covers this scenario & challenges

C# API - Motivations

- Enable organizations invested deeply in .NET to build Apache Spark applications in C#
- Reuse of existing .NET libraries in Spark applications

Why Yet Another Language Binding

Spark Survey 2015 Results

MOST IMPORTANT ASPECTS OF SPARK

FASTEST GROWING AREAS FROM 2014 TO 2015

Popularity of C#

- StackOverflow.com Developer Survey
- RedMonk Programming Language Rankings

.NET ecosystem ~ enabling languages like F#

C# API - Goal

Make C# a first-class language for building Apache Spark applications

Word Count Example in C#

Scala

```
val textFile = spark.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                     .map(word => (word, 1))
                     .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

C#

```
var textFile = sparkContext.TextFile(@"hdfs://...");
var counts = textFile.FlatMap(x => x.Split(' '))
                     .Map(w => new KeyValuePair<string, int>(w, 1))
                     .ReduceByKey((x, y) => x + y)
                     .Map(wordCount => $"{wordCount.Key},{wordCount.Value}");
counts.SaveAsTextFile(@"hdfs://...");
```

Kafka Example in C#

```
StreamingContext sparkStreamingContext = StreamingContext.GetOrCreate(checkpointPath,
    () =>
    {
        var ssc = new StreamingContext(sparkContext, slideDurationInMillis);
        ssc.Checkpoint(checkpointPath);

        var stream = KafkaUtils.CreateDirectStream(ssc, topicList, kafkaParams, perTopicPartitionKafkaOffsets);
        var countByLogLevelAndTime = stream
                    .Map(kvp => Encoding.UTF8.GetString(kvp.Value))
                    .Map(line => line.Split(',')) //message format:[timestamp],[loglevel],[logmessage]
                    .Map(columns => new KeyValuePair<string, int>($"{columns[0]},{columns[1]}", 1))
                    .ReduceByKeyAndWindow((x, y) => x + y, (x, y) => x - y, windowDurationInSecs, slideDurationInSecs, numPartitions)
                    .Map(logLevelCountPair => $"{logLevelCountPair.Key},{logLevelCountPair.Value}");

        countByLogLevelAndTime.ForeachRDD(countByLogLevel =>
        {
            countByLogLevel.SaveAsTextFile($"{appOutputPath}/{Guid.NewGuid()}");
        });

        return ssc;
    });

sparkStreamingContext.Start();
sparkStreamingContext.AwaitTermination();
```

Initialize StreamingContext & Checkpoint

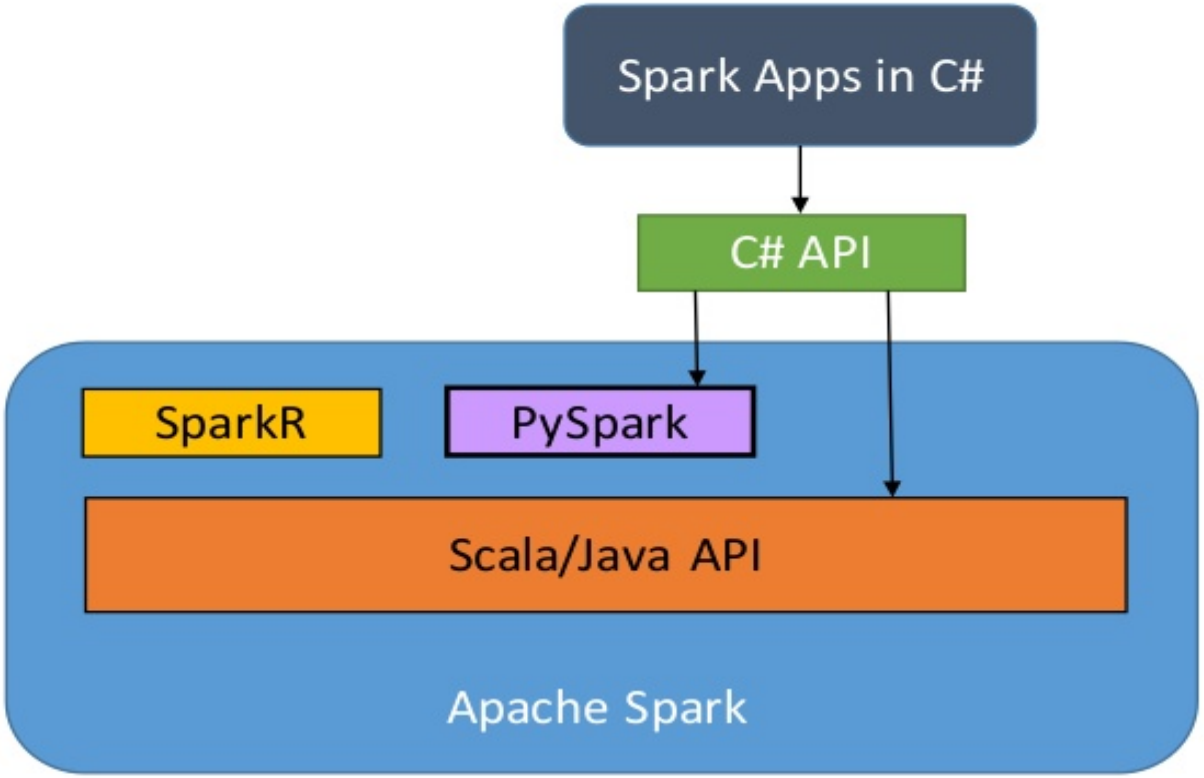
Create Kafka DStream

Use DStream transformations to count logs by loglevel within a time window

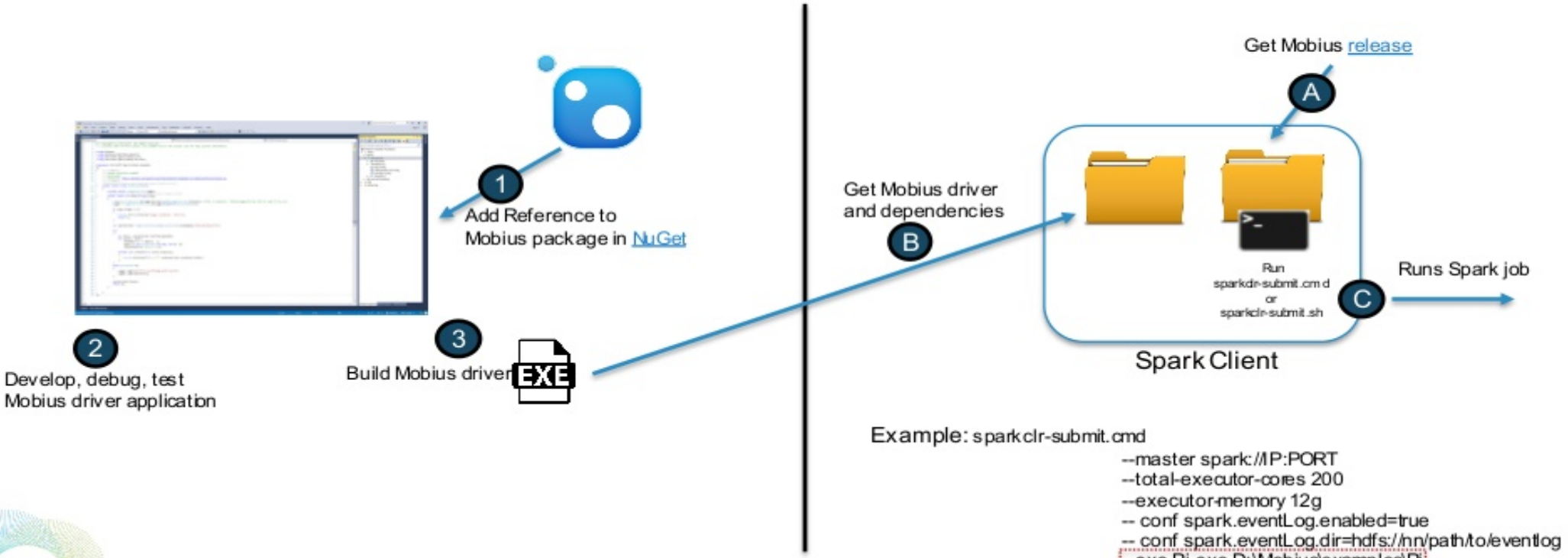
Save log count

Start stream processing

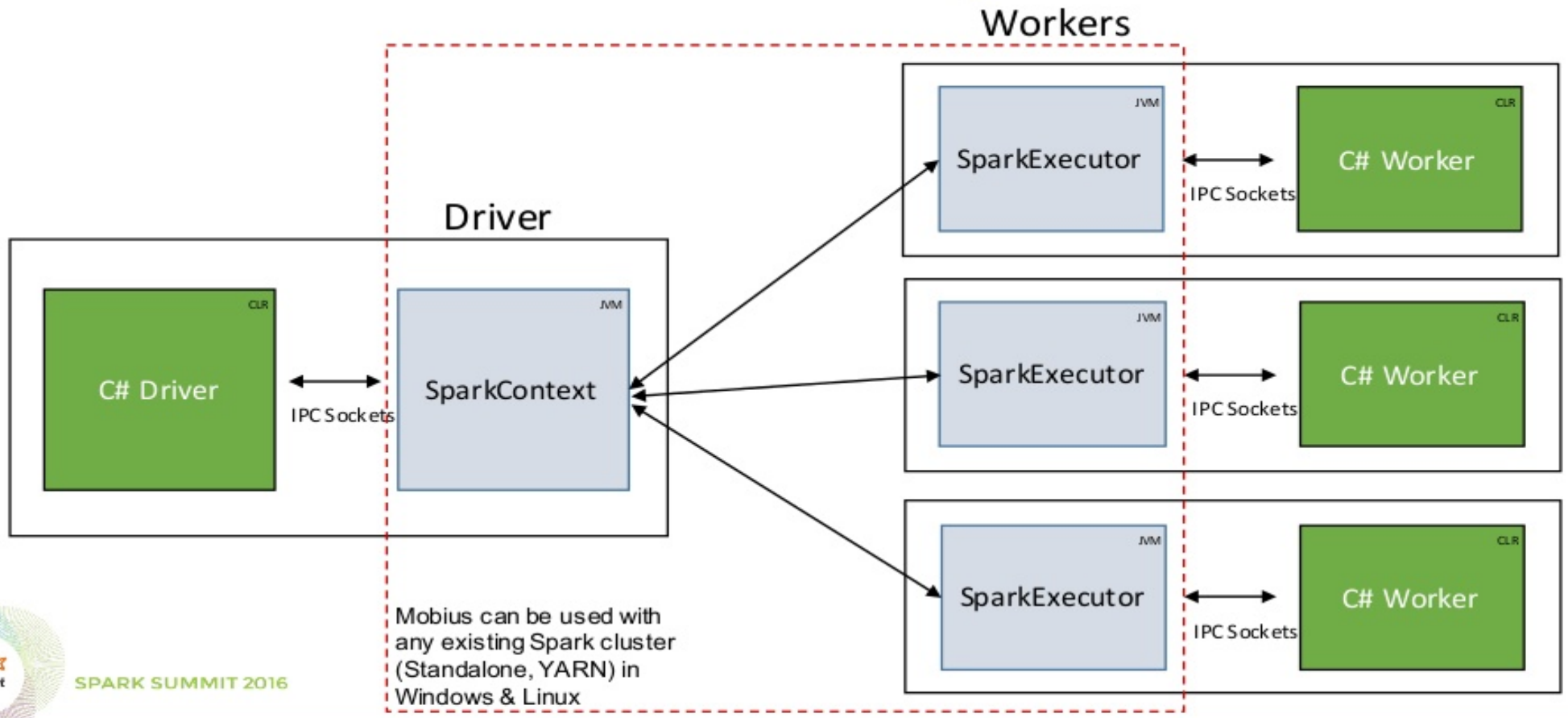
Mobius: C# API for Spark



Develop & Launch Mobius Applications



Mobius & Spark



Mobius in Linux

- Mono (open source implementation of .NET framework) used for C# with Spark in Linux
- Mobius project CI (build, unit & functional tests) in Ubuntu
- Users reported using Mobius in Ubuntu, CentOS, OSX
- Mobius validated with Spark clusters in Azure HDInsight and Amazon Web Services EMR
- More info at linux-instructions.md @ GitHub

Project Info

- https://github.com/Microsoft/Mobius
  Contributions welcome!
- MIT license
- Discussions
  - StackOverflow: tag "SparkCLR"
  - Gitter: https://gitter.im/Microsoft/Mobius
  - Twitter: @MobiusForSpark

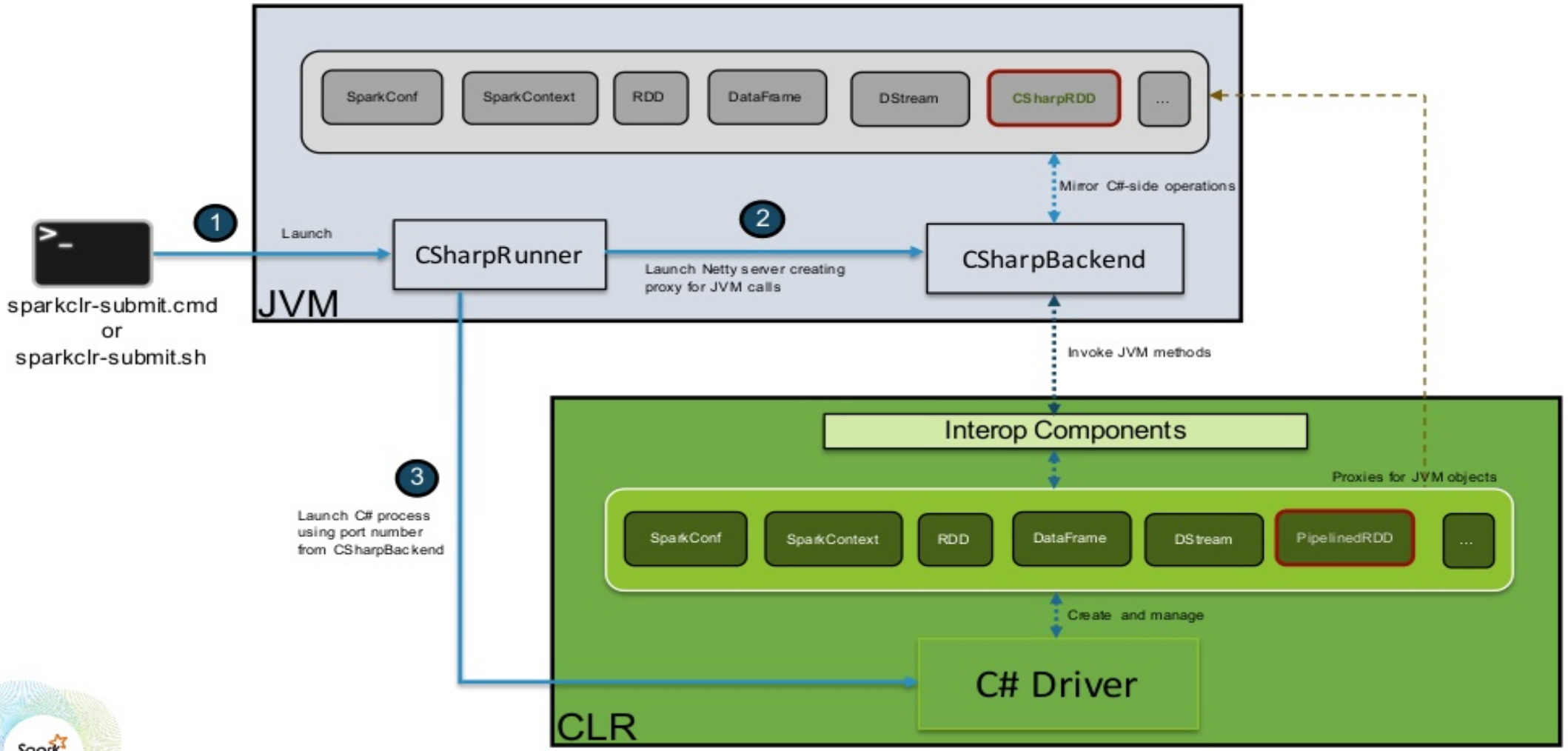
Project Status

- Past Releases
  - v1.5.200 (Spark 1.5.2)
  - v1.6.100 (Spark 1.6.1)
- Upcoming Release
  - v2.0.000 (Spark 2.0.0)
- Work in progress
  - Support for interactive scenarios (Zeppelin/Jupyter integration)
  - Exploration of support for ML scenarios
  - Idiomatic F# API

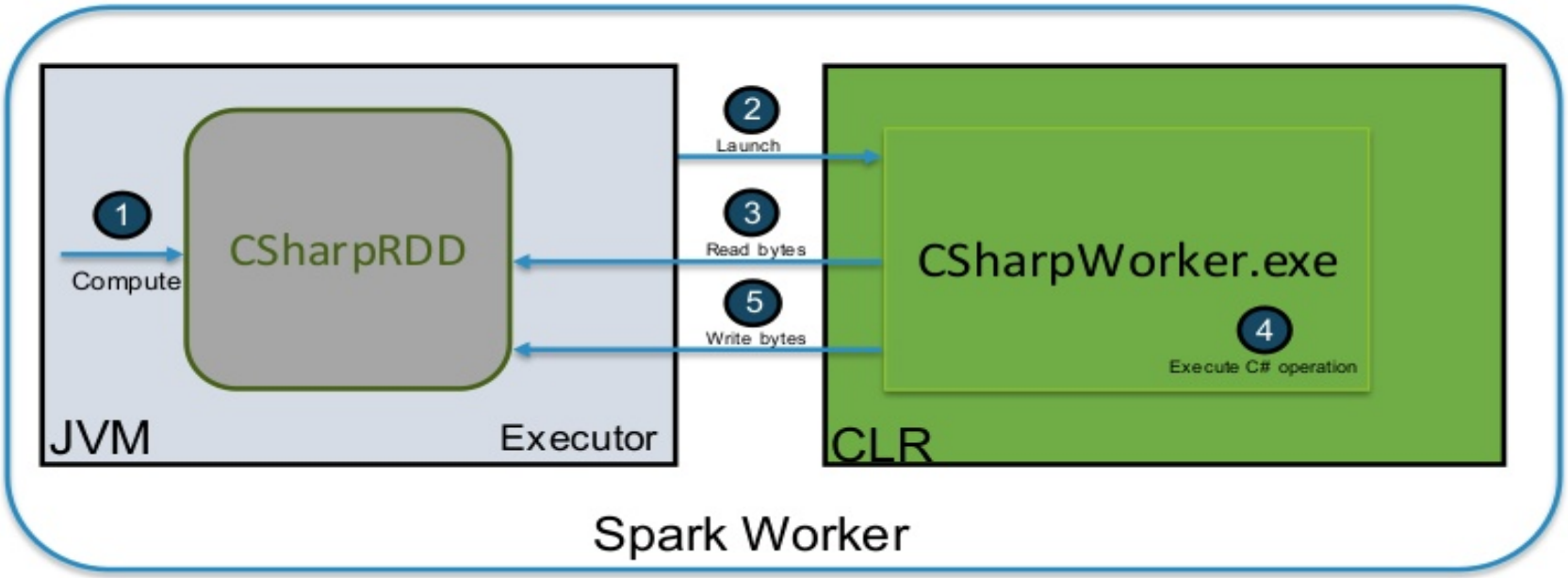
UNDER THE HOOD

CSharpRDD

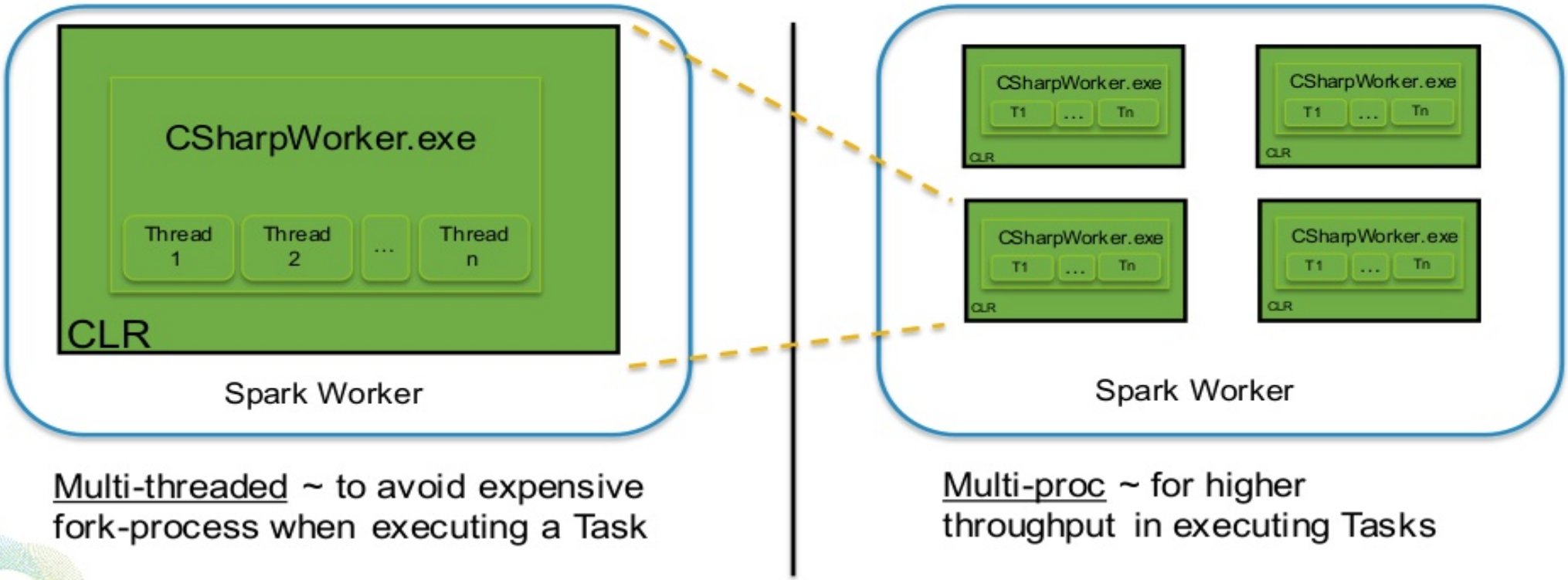
- C# operations use CSharpRDD which needs CLR to execute
  - If no C# transformation or UDF, CLR is not needed ~ execution is entirely JVM-based
- RDD<byte[]>
  - Data is stored as serialized objects and sent to C# worker process
- Transformations are pipelined when possible
  - Avoids unnecessary serialization & deserialization within a stage


SparkConf
SparkContext
RDD
DataFrame
DStream
CSharpRDD
...
Mirror C#-side operations
1
Launch
CSharpRunner
2
Launch Netty server creating
proxy for JVM calls
CSharpBackend
JVM
sparkclr-submit.cmd
or
sparkclr-submit.sh
Invoke JVM methods
Interop Components
Proxies for JVM objects
3
Launch C# process
using port number
from CSharpBackend
SparkConf
SparkContext
RDD
DataFrame
DStream
PipelinedRDD
...
Create and manage
C# Driver
CLR

Worker-side Interop



Worker Optimization Options



Multi-threaded ~ to avoid expensive fork-process when executing a Task

Multi-proc ~ for higher throughput in executing Tasks

Performance Considerations

- Map & Filter RDD operations in C# require serialization & deserialization of data ~ impacts performance
  - C# operations are pipelined when possible ~ minimizes Ser/De
  - Persistence is handled by JVM ~ checkpoint/cache on a RDD impacts pipelining for CLR operations

- DataFrame operations without C# UDFs do not require Ser/De
  - Perf will be same as native Scala-based Spark application
  - Execution plan optimization & code generation perf improvements in Spark leveraged

THANK YOU.

- Mobius is production-ready
- Use Mobius to build Apache Spark jobs in .NET
- Contribute to github.com/**Microsoft/Mobius**
- @MobiusForSpark

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO