



UNIVERSITAT DE
BARCELONA

Master in Fundamental Principles of Data Science

Dr Rohit Kumar



UNIVERSITAT DE
BARCELONA

Spark SQL

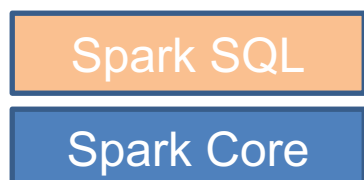
Why Spark SQL

- SQL is the language everyone understand in data analytics.
- Built on top of Spark provides all advantages of Spark.

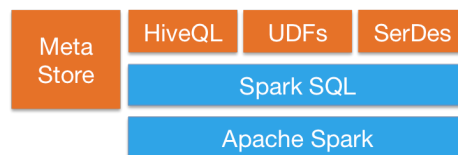
Why Spark SQL



Integrated



Unified Data Access



Hive Compatibility



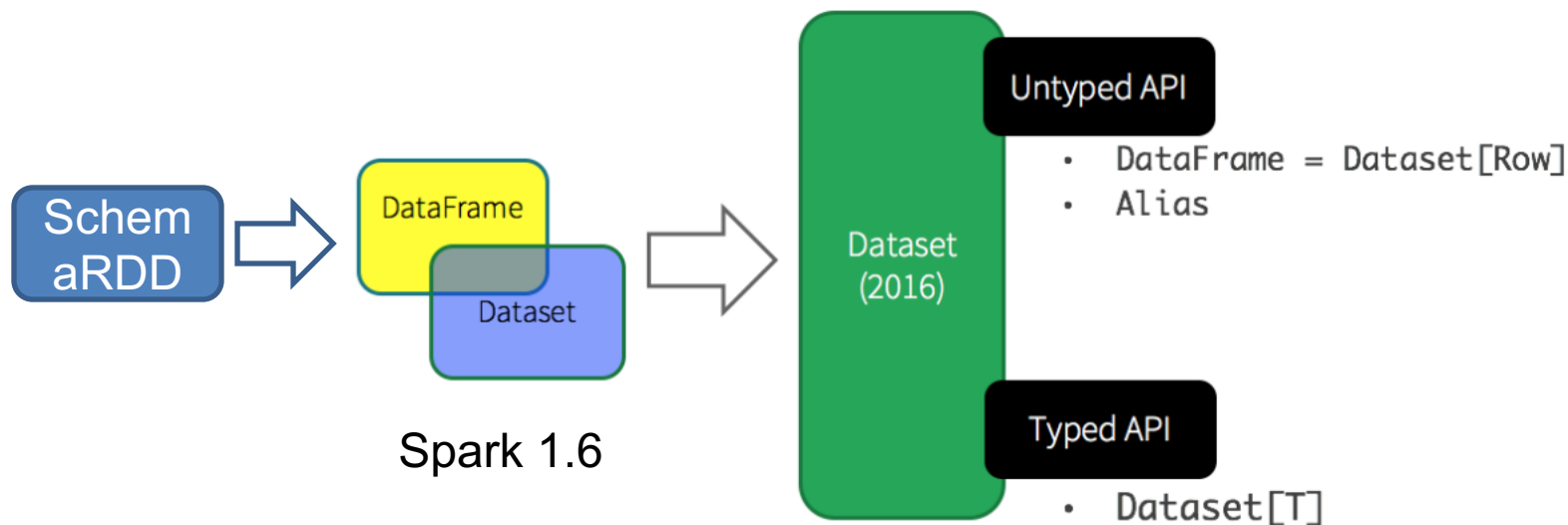
**Standard
Connectivity**



Scalability

Datasets and DataFrames

Unified Apache Spark 2.0 API



UnTyped vs typed

Employee Table

Name	Age	Joining Date
Rohit	32	12-08-2012
Rahul	35	17-09-2014

C = Employee.Name/4

Run time error

Name (char(20))	Age(Int)	Joining Date(Date)
Rohit	32	12-08-2012
Rahul	35	17-09-2014

C = Employee.Name/4

Compile time error


Typed and Un-typed APIs

Language	Main Abstraction
Scala	Dataset[T] & DataFrame (alias for Dataset[Row])
Java	Dataset[T]
Python*	DataFrame
R*	DataFrame

**Python and R have no compile-time type-safety, we only have untyped APIs, namely DataFrames.*

Benefits of Dataset APIs

Static-typing and runtime type-safety

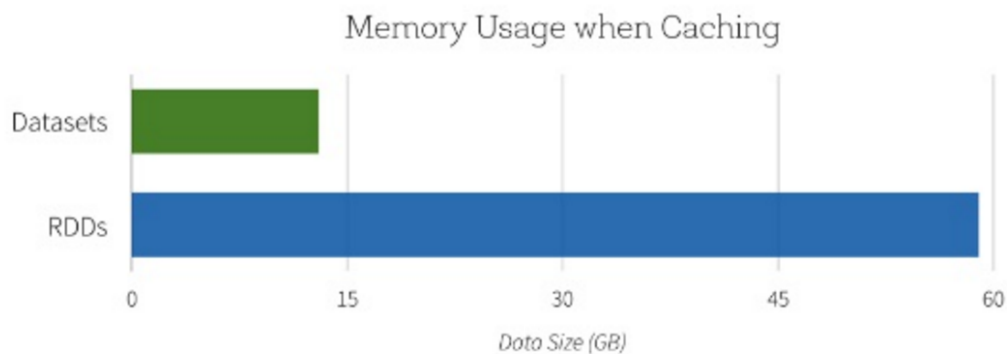


	SQL	DataFrames	Datasets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time

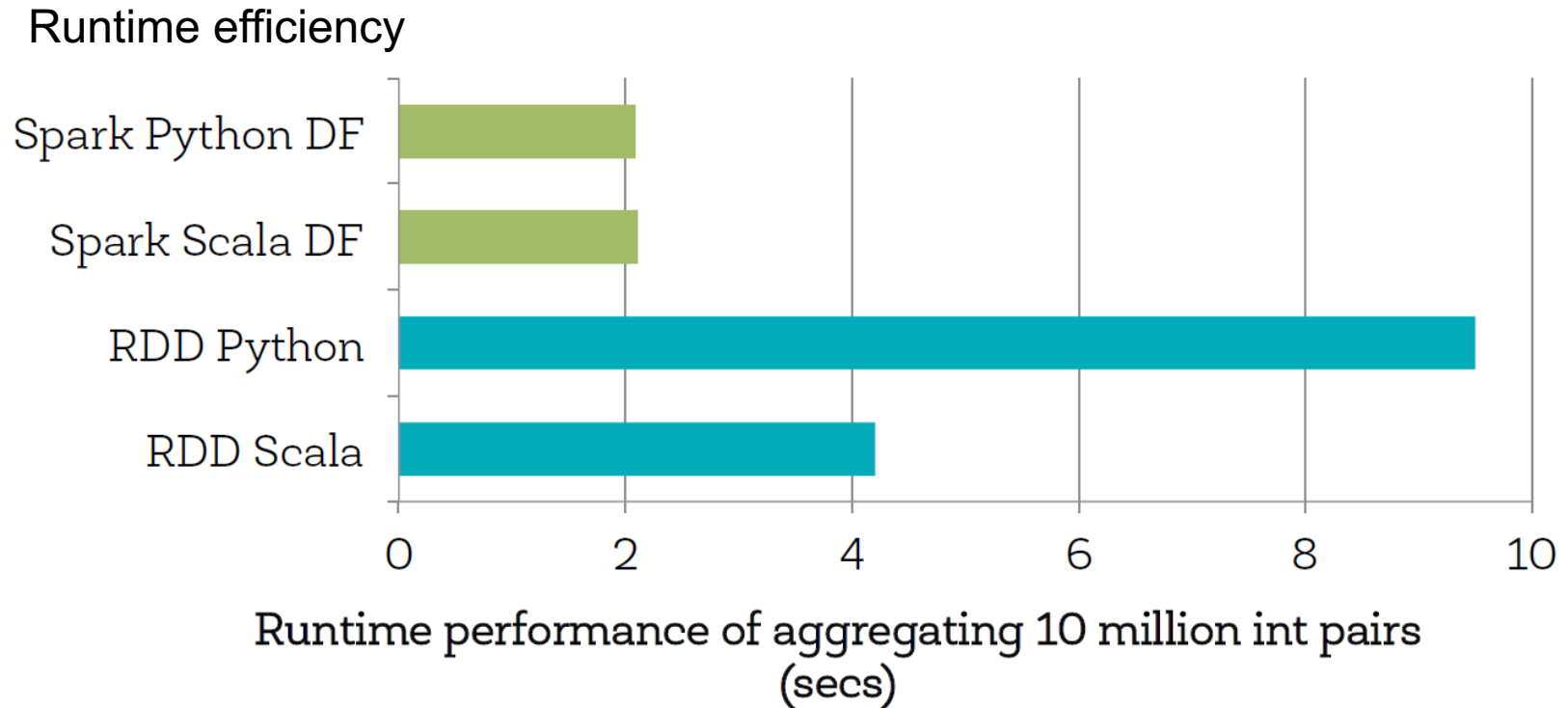
Benefits of Dataset APIs

Better Performance and Optimizations.

Space Efficiency



Benefits of Dataset APIs



Spark Session

Spark Sessions are the new entry point to spark.

```
from pyspark.sql import SparkSession
```

```
spark = SparkSession \
```

```
.builder \
```

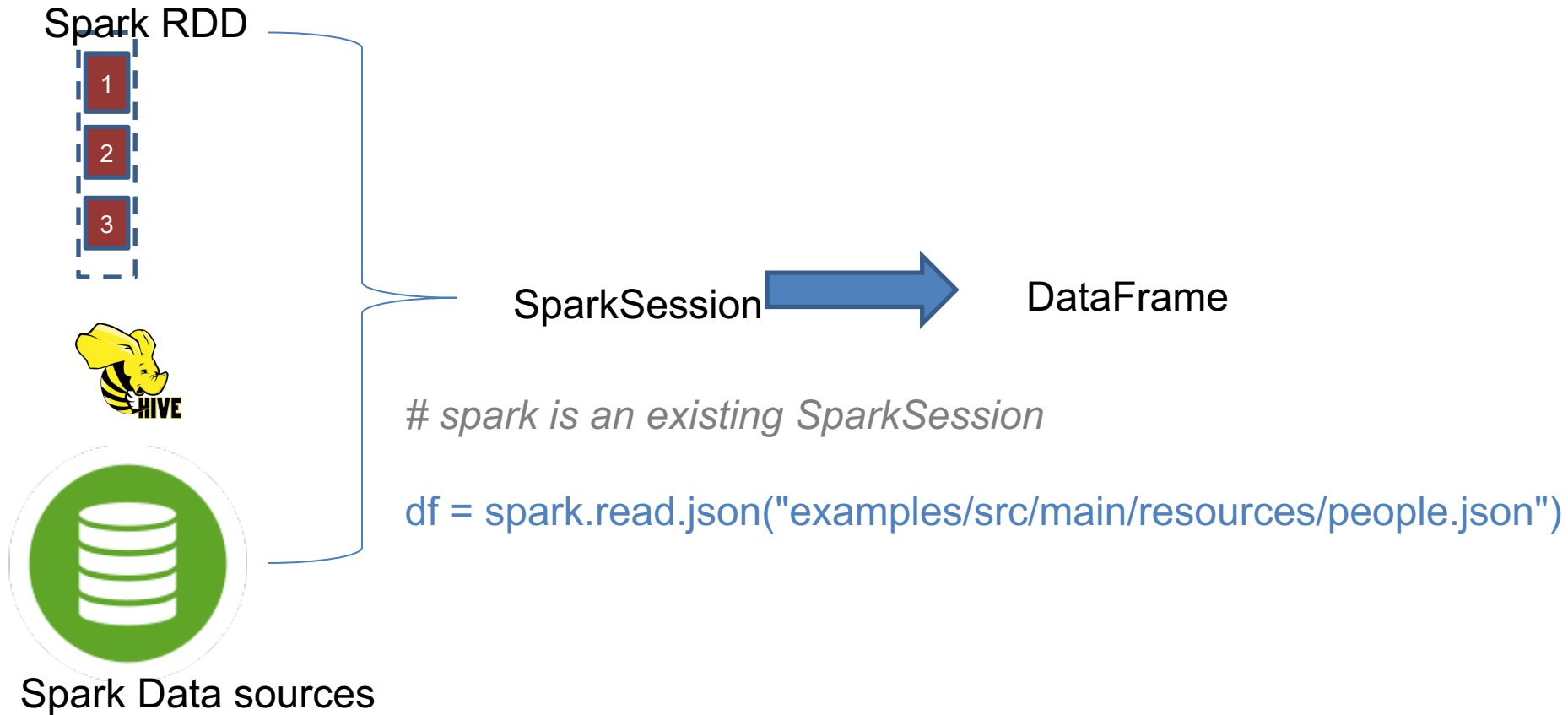
```
.master("local[*]") \
```

```
.appName("Python Spark SQL basic example") \
```

```
.config("spark.some.config.option", "some-value") \
```

```
.getOrCreate()
```

Creating Dataframes



Creating Datasets

Datasets are same as RDD in concept except!



Similar to standard serialization, *encoders* are responsible for turning an object into bytes. Encoders use a format that allows Spark to perform many operations like filtering, sorting and hashing without **deserializing** the bytes back into an object.

DataFrame operations

Think of DataFrame as a table!

Name	Age	Joining Date
Rohit	32	12-08-2012
Rahul	35	17-09-2014

In Python it's possible to access a DataFrame's columns either by attribute (`df.age`) or by indexing (`df['age']`)

Running SQL Queries Programmatically

- The `dataFrame` can be registered as temporary view and used to run SQL queries.

Open the notebook **Spark DataFrames & SQL - Part 2**

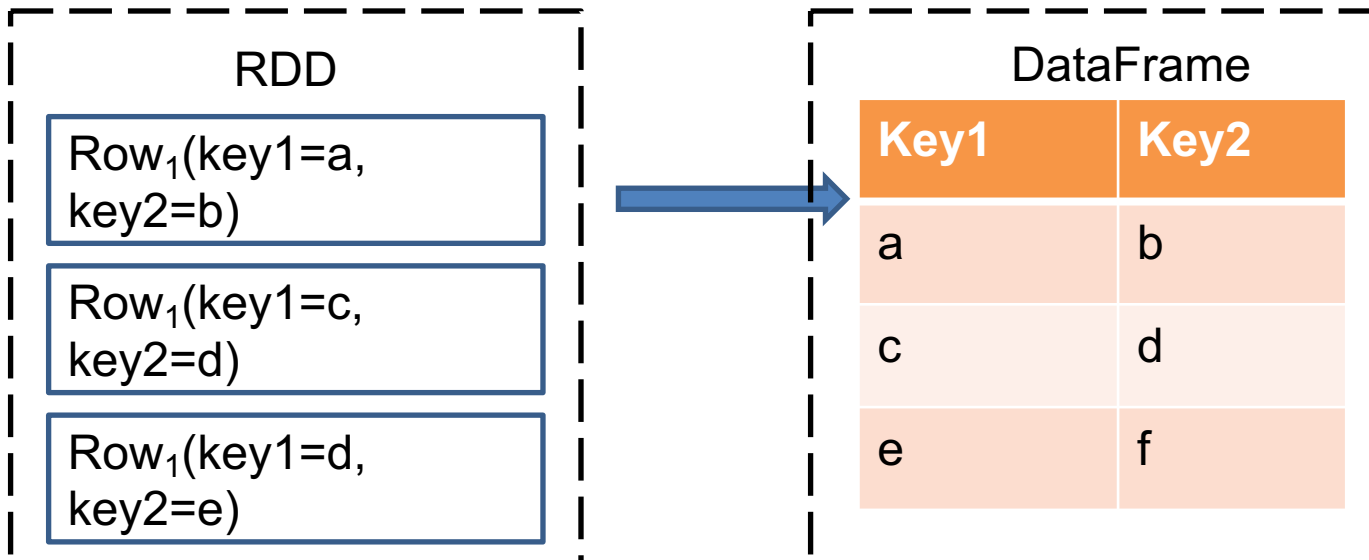
<https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3040011785658046/2693060268285012/5812217827524789/la test.html>

Interoperating with RDDs

Inferring the Schema Using Reflection



Rows are constructed by passing a list of key/value pairs as kwargs to the Row class



Type is inferred
using
sampling



Programmatically Specifying the Schema

DataFrame can be created programmatically with three steps:

- Create an RDD of tuples or lists from the original RDD;
- Create the schema represented by a **StructType** matching the structure of tuples or lists in the RDD created in the step 1.
- Apply the schema to the RDD via createDataFrame method provided by SparkSession.

Let's check section 2.0 in notebook!

Aggregations

- `count()`
- `countDistinct()`
- `avg()`
- `max()`
- `min()`
- ... many more .. [Function List](#)

User-Defined Aggregate Functions

- Create the function
- Register the function as UDF
- Call the UDF in Spark SQL

Let's check section 3.0 in notebook!

Data Sources

- Spark SQL supports operating on a variety of data sources through the DataFrame interface.
- We will see some basic concepts and then more specifically on Json dataset.

Generic Load/Save Functions

Default data source -



Apache Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language.

You can change the default by setting **spark.sql.sources.default**

Manually Specifying Options

You can also manually specify the data source that will be used along with any extra options that you would like to pass to the data source.

Such as : path, format, schema, SaveMode and other data source specific options.

Run SQL on files directly

- Instead of using read API to load a file into DataFrame and query it, you can also query that file directly with SQL.
- Spark uses a new feature call predicate push down which helps to filter the data at source before loading it in memory. This is done automatically as part of performance tuning by spark.



Save Modes

Scala/Java	Any Language	Meaning
SaveMode.ErrorIfExists(default)	"error"(default)	When saving a DataFrame to a data source, if data already exists, an exception is expected to be thrown.
SaveMode.Append	"append"	When saving a DataFrame to a data source, if data/table already exists, contents of the DataFrame are expected to be appended to existing data.
SaveMode.Overwrite	"overwrite"	Overwrite mode means that when saving a DataFrame to a data source, if data/table already exists, existing data is expected to be overwritten by the contents of the DataFrame.
SaveMode.Ignore	"ignore"	Ignore mode means that when saving a DataFrame to a data source, if data already exists, the save operation is expected to not save the contents of the DataFrame and to not change the existing data. This is similar to a CREATE TABLE IF NOT EXISTS in SQL.

Persistent Tables

DataFrame

Name	Age	Joining Date
Rohit	32	12-08-2012
Rahul	35	17-09-2014

saveAsTable



local Hive metastore

Apache **Derby** 

Persistent Tables

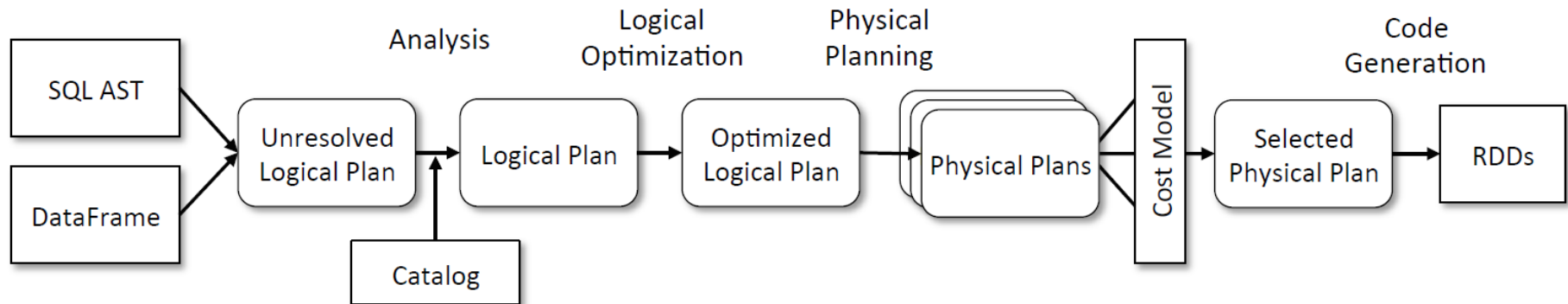
```
df.write.saveAsTable("t")
```

Stores in default location and will be removed if the table is dropped.

```
df.write.option("path", "/some/path").saveAsTable("t")
```

Stores in given path and data will not be removed even if the table is dropped.

Under the hood



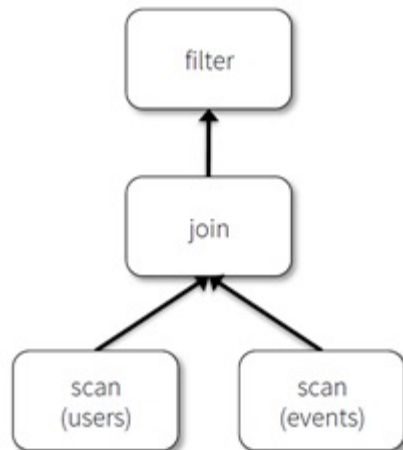
DataFrames and SQL share the same optimization/execution pipeline

Simple example

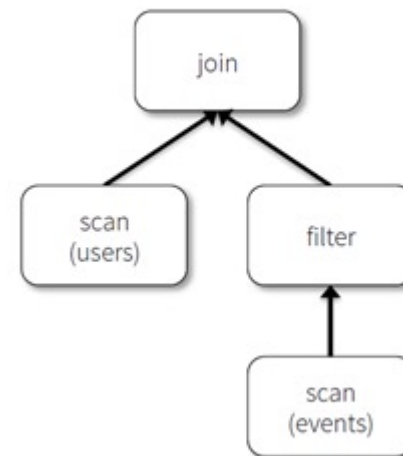
```
users = spark.read.json("users.json")
events =
  spark.read.json("events.json")
joined = users.join(events, users.id ==
  events.uid)
filtered = joined.filter(events.date >= "2015-
  01-01")
```

This join is expensive

logical plan



physical plan



Reference

- <https://docs.databricks.com/spark/latest/spark-sql/index.html>
- <https://spark.apache.org/docs/latest/sql-getting-started.html>
- Learning Spark by *Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia*