SPARKLYR: RECAP, UPDATES AND USE CASES

JAVIER LURASCHI

SPARK SUMMIT 2017

SCHEDULE

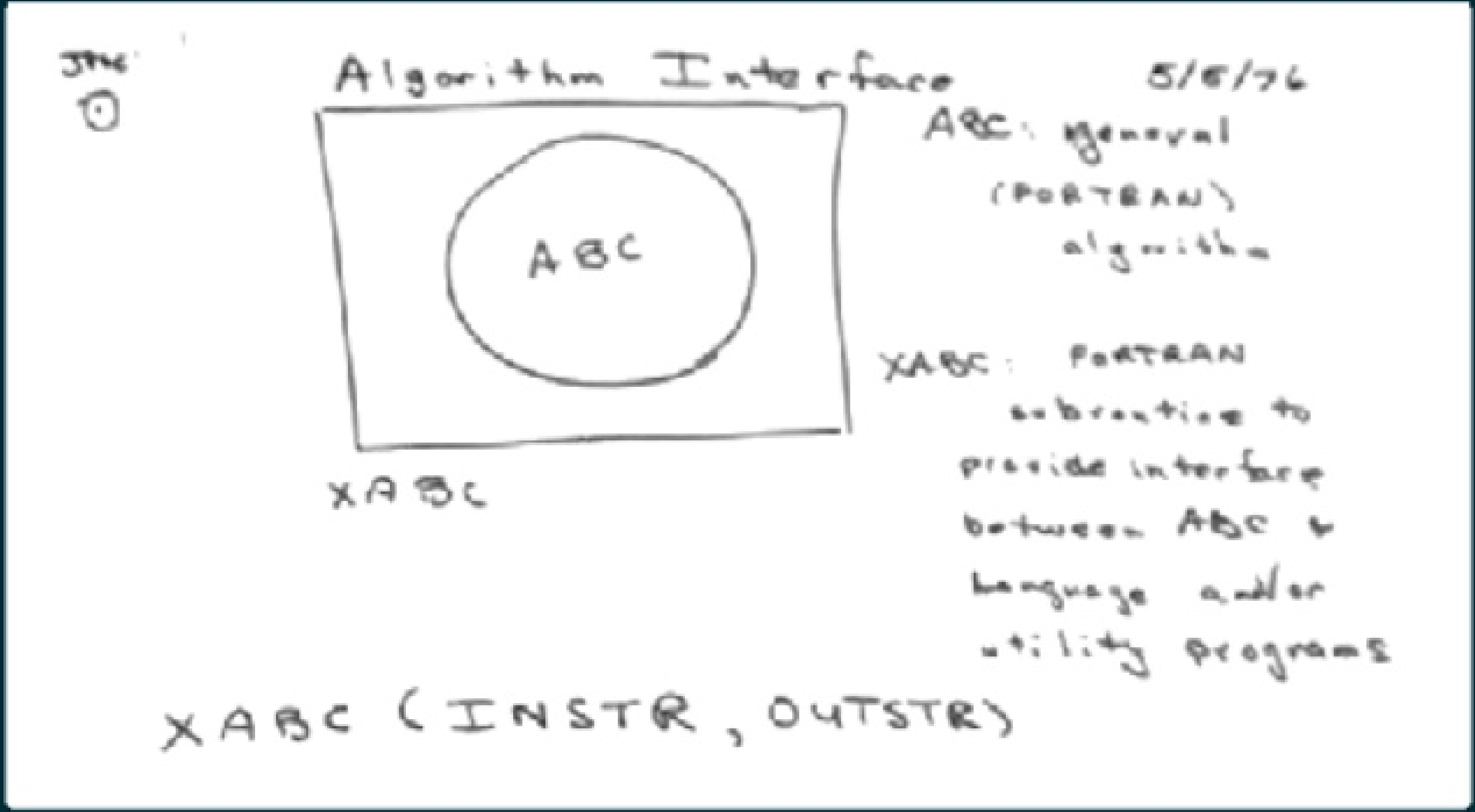
2:00-2:30 sparklyr: Recap and Updates

2:40-3:10 sparklyr: Architecture and Use Cases

RECAP

S - LANGUAGE FOR STATS COMPUTING

Designed at Bell Laboratories by John Chambers, where computing was done by calling Fortran subroutines.



JMC
1

Algorithm Interface

5/6/76

ABC: general (FORTRAN) algorithm

XABC: FORTRAN subroutine to provide interface between ABC & language and/or utility programs

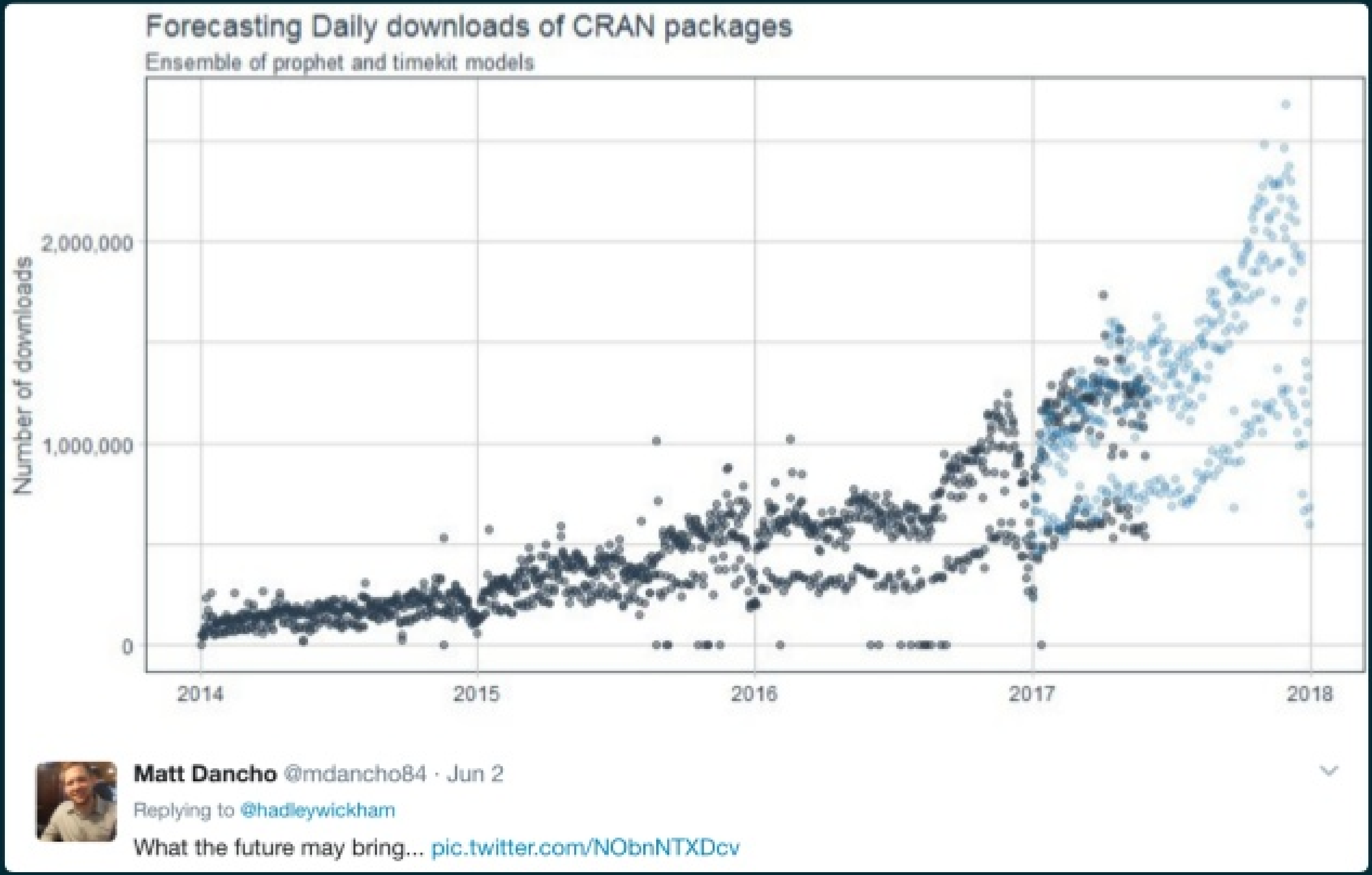
XABC (INSTR, OUTSTR)

S - LANGUAGE FOR STATS COMPUTING

> *"S is great but serious data analysis will always be done in Fortran" - Bell Labs Management*

R - MODERN S

R community is noted for its active package contributions.
CRAN R's package manager with ~10K packages.


Forecasting Daily downloads of CRAN packages
Ensemble of prophet and timekit models
Number of downloads
2,000,000
1,000,000
0
2014
2015
2016
2017
2018

Matt Dancho @mdancho84 · Jun 2

Replying to @hadleywickham

What the future may bring... pic.twitter.com/NObnNTXDcv

R - DESIGN PRINCIPLES

1. Everything that exists is an object
2. Everything that happens is a function call
3. R is built on interfaces to many (R and non-R) algorithms

SPARKLYR - R INTERFACE FOR SPARK

```r
library(sparklyr)                      # Load sparklyr
```

```r
spark_install()                        # Install Apache Spark
```

```r
sc <- spark_connect(master = "local")  # Connect to local instance
```

```r
library(dplyr)                         # Data Manipulation Grammar
mtcars_tbl <- copy_to(sc, mtcars)      # Copy mtcars into Spark
count(mtcars_tbl)                      # Count records
```

```r
ml_linear_regression(mtcars_tbl,       # Perform linear regression
  response = "mpg",                    # Response vector
  features = c("wt", "cyl"))           # Features for the model fit
```

```r
library(DBI)                           # R Database Interface
dbGetQuery(sc, "SELECT * FROM mtcars") # Run SQL query in Spark
```

```r
invoke(spark_context(sc), "version")   # Run sc.version in Scala
```

```r
compile_package_jars()                 # Compile Scala code
```

UPDATES

SPARKLYR 0.4

KICKOFF: April, 2016

ANNOUNCED: June, 2016

RELEASED: September, 2016

CLOUDERA CERTIFIED: October, 2016

NEW FEATURES: Install, connection, backend, data, DataFrame, DBI, dplyr, MLlib, extensions

SPARKLYR 0.5

RELEASED: January 2017

MINOR: 0.5.2, 0.5.3, 0.5.4 and 0.5.5

NEW CONNECTIONS: Gateway, Livy and Databricks

IMPROVEMENTS: MLlib, DataFrame, compatibility and dplyr

SPARKLYR 0.6 (DEVEL)

RELEASED: Soon

NEW FEATURES: Distributed R

IMPROVEMENTS: Data, dplyr, databases, DataFrames, MLlib, broom, compatibility, connections, extensions and backend

SPARK-INSTALL

Cross-platform installer for Apache Spark.

```r
library(sparkinstall)
spark_install(version = "1.6.2")    # Install Spark from R
```

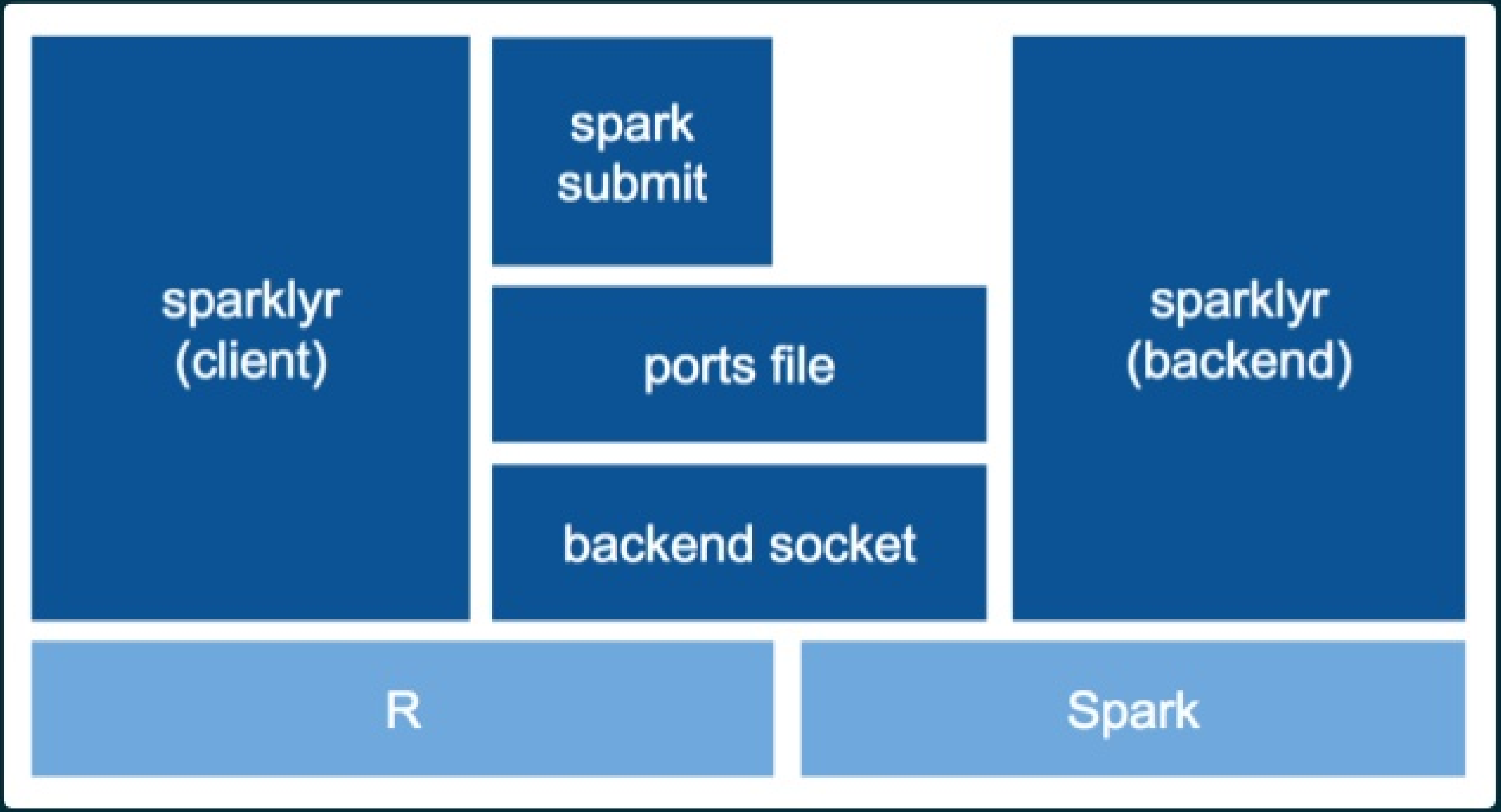
```python
from spark_install import *
spark_install(version = "1.6.2")    # Install Spark from Python
```

“This project provides a cross-platform installer for Apache Spark designed to use system resources efficiently under a common API. This initial version commes with support for R and Python that arose from a collaboration between RStudio and Microsoft” - github.com/rstudio/spark-install

ARCHITECTURE

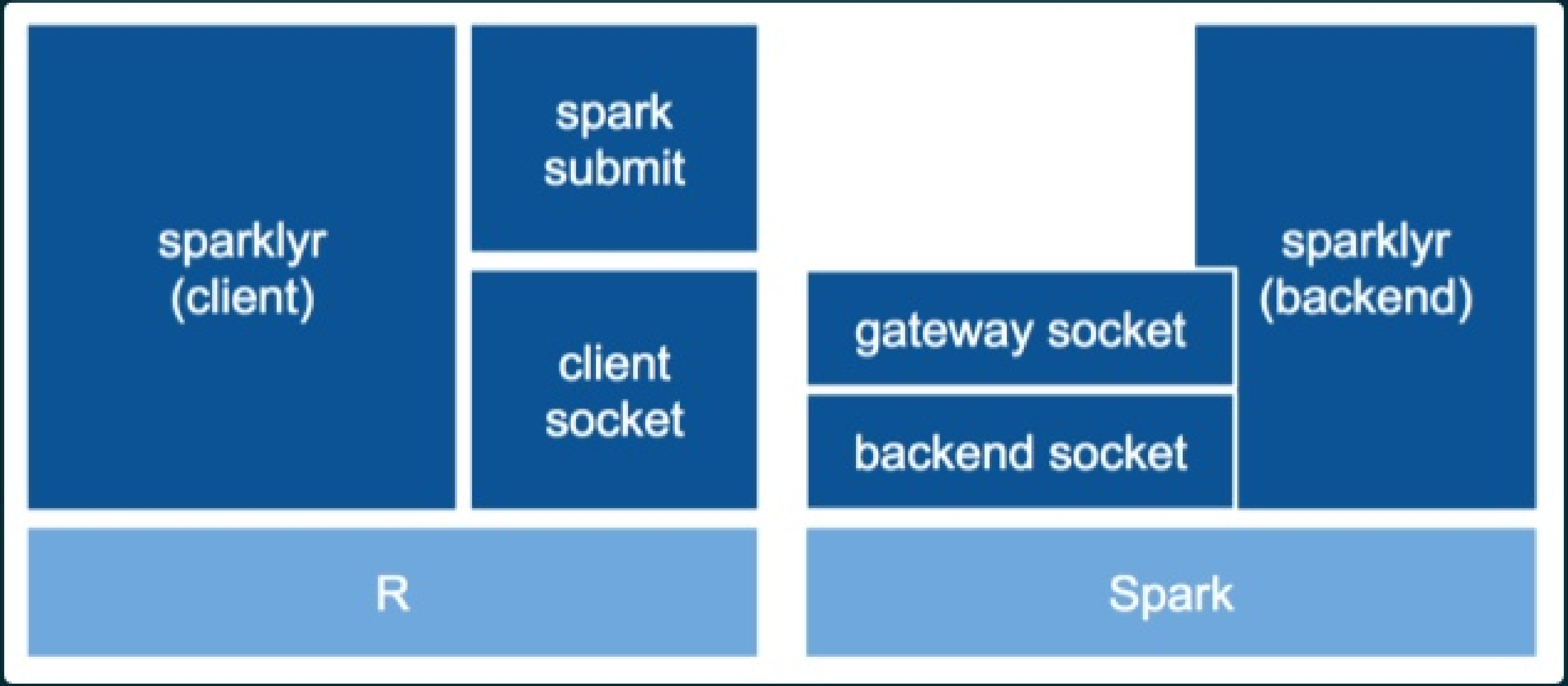
BACKEND

SPARKLYR 0.4



GATEWAY

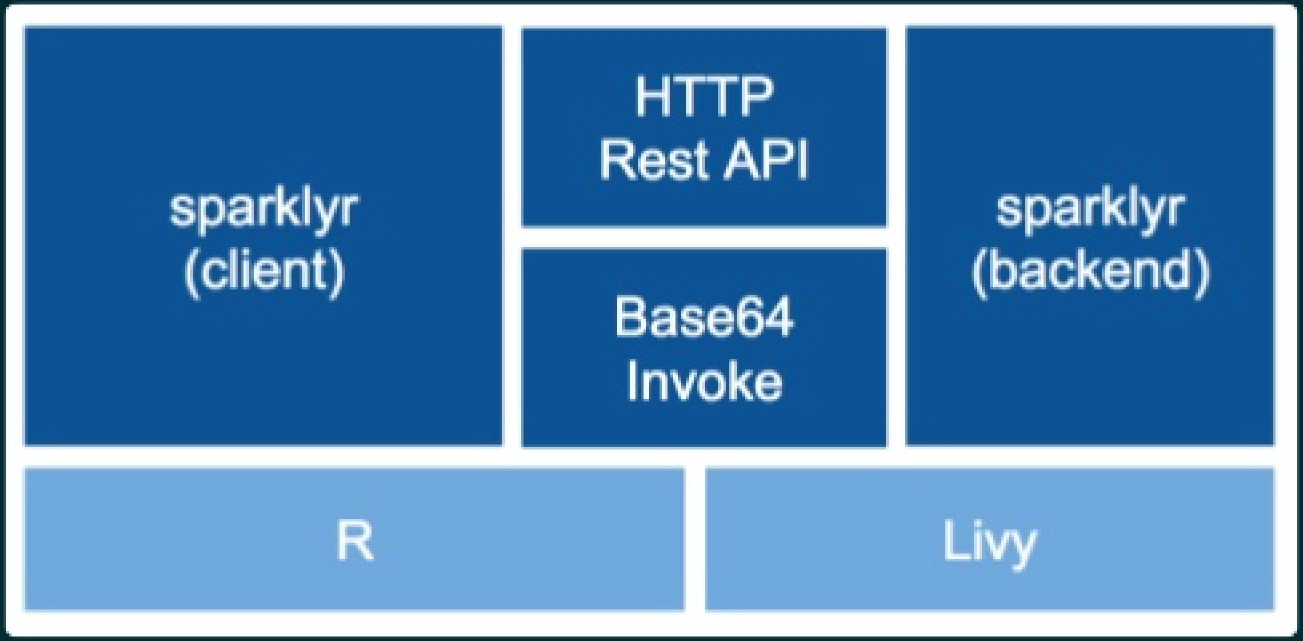
SPARKLYR 0.5



Replace ports file with gateway socket

LIVY

SPARKLYR 0.5

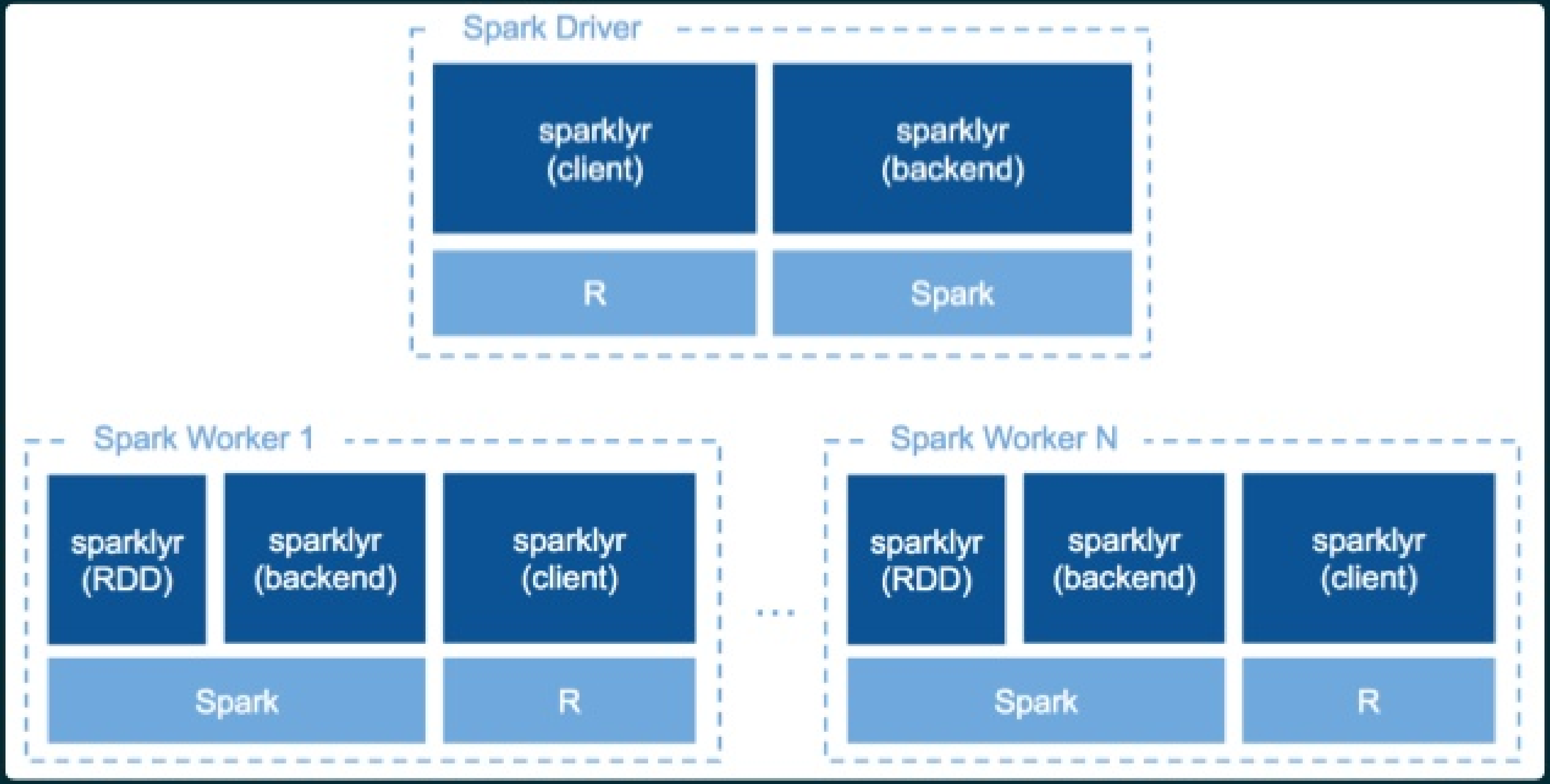


```
var sparklyrRetVar_0 = LivyUtils.invokeFromBase64(
  "AAAAHm9yZy5hcGFjaGUuc3BhcmsuU3BhcmtDb250" +
  "ZXh0AAAAAAEAAAAMZ2V0T3JDcmVhdGUAAAAAAAA=="
)
```

Implement Livy connections

WORKER

SPARKLYR 0.6



Implement R Workers

USE CASES

ANALYSIS WITH SQL AND DPLYR

```r
delay <- flights_tbl %>%
  group_by(tailnum) %>%
  summarise(count = n(), dist = mean(distance), delay = mean(arr_delay)
  filter(count > 20, dist < 2000, !is.na(delay)) %>%
  collect

# plot delays
library(ggplot2)
ggplot(delay, aes(dist, delay)) +
  geom_point(aes(size = count), alpha = 1/2) +
  geom_smooth() +
  scale_size_area(max_size = 2)
```

```r
library(DBI)
dbGetQuery(sc, "SELECT * FROM flights LIMIT 100")
```

MACHINE LEARNING WITH MLLIB

```
# transform our data set, and then partition into 'training', 'test'
partitions <- mtcars_tbl %>%
  filter(hp >= 100) %>%
  mutate(cyl8 = cyl == 8) %>%
  sdf_partition(training = 0.5, test = 0.5, seed = 1099)

# fit a linear model to the training dataset
partitions$training %>%
  ml_linear_regression(response = "mpg", features = c("wt", "cyl"))
```

MACHINE LEARNING WITH RSPARKLING

EXTENSION BY H2O (Navdeep Gill)

```r
library(rsparkling)
library(sparklyr)
library(dplyr)
library(h2o)

sc <- spark_connect(master = "local")
```

```r
mtcars_h2o <- as_h2o_frame(sc, mtcars_tbl,
                           strict_version_check = FALSE)

h2o.glm(x = c("wt", "cyl"),
        y = "mpg",
        training_frame = mtcars_h2o,
        lambda_search = TRUE)
```

```r
h2o_flow(sc, strict_version_check = FALSE)
```

GRAPHFRAMES WITH SPARKLYGRAPHS

EXTENSION BY Kevin Kuo

```r
spark_disconnect(sc)
```

```r
library(sparklygraphs)
library(sparklyr)
library(dplyr)

sc <- spark_connect(master = "local", version = "2.1.0")
highschool_tbl <- copy_to(sc, ggraph::highschool, "highschool")

# create a table with unique vertices using dplyr
vertices_tbl <- sdf_bind_rows(
  highschool_tbl %>% distinct(from) %>% transmute(id = from),
  highschool_tbl %>% distinct(to) %>% transmute(id = to)
)

# create a table with <source, destination> edges
edges_tbl <- highschool_tbl %>% transmute(src = from, dst = to)

# calculate PageRank over the highschool dataset
gf_graphframe(vertices_tbl, edges_tbl) %>%
  gf_pagerank(reset_prob = 0.15, max_iter = 10L, source_id = "1")
```

DISTRIBUTED EXECUTION

SPARKLYR 0.6

```r
spark_apply(highschool_tbl, function(x) {
  x + rgamma(1, 2)
})
```

THANK YOU!

QUESTIONS?

@javierluraschi

javier@rstudio.com

```
spark_disconnect(sc)
```