INF550 - Computação em Nuvem I

Apache Spark

Islene Calciolari Garcia
Instituto de Computação - Unicamp
Julho de 2018

Programação

16/06 **MapReduce** (Islene)

23/06 **Virtualização** (Luiz)

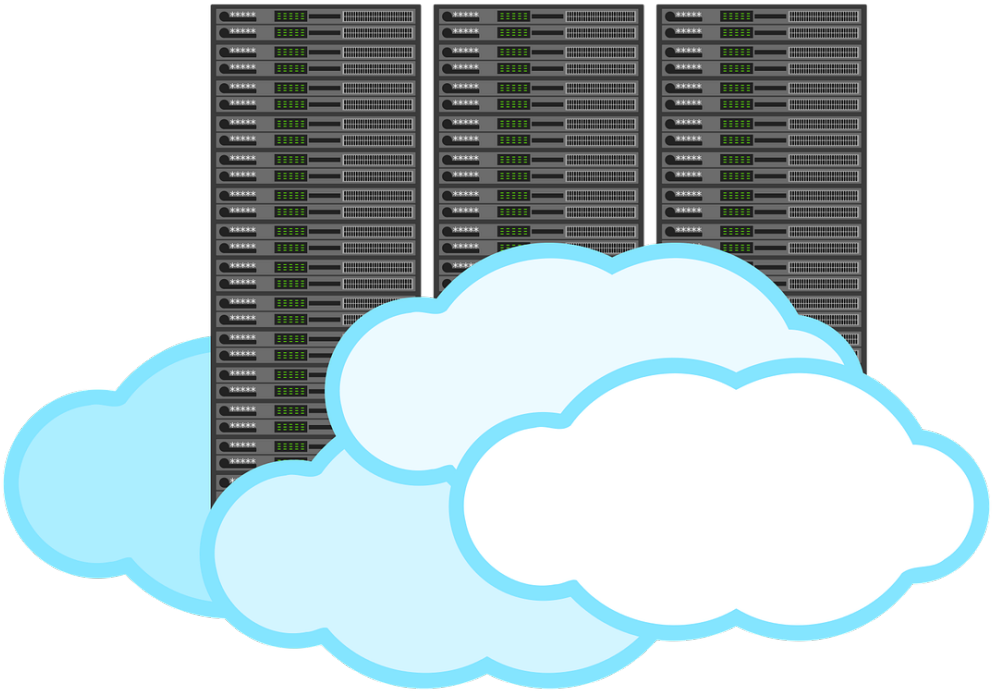
30/06 **Computação em nuvens** (Luiz)

07/07 **Spark** (Islene)

- Revisão: MapReduce
- Resilient Distributed Datasets (RDDs)
- Transformações e Ações
- Exercício em laboratório

Computação em Nuvem e Big Data

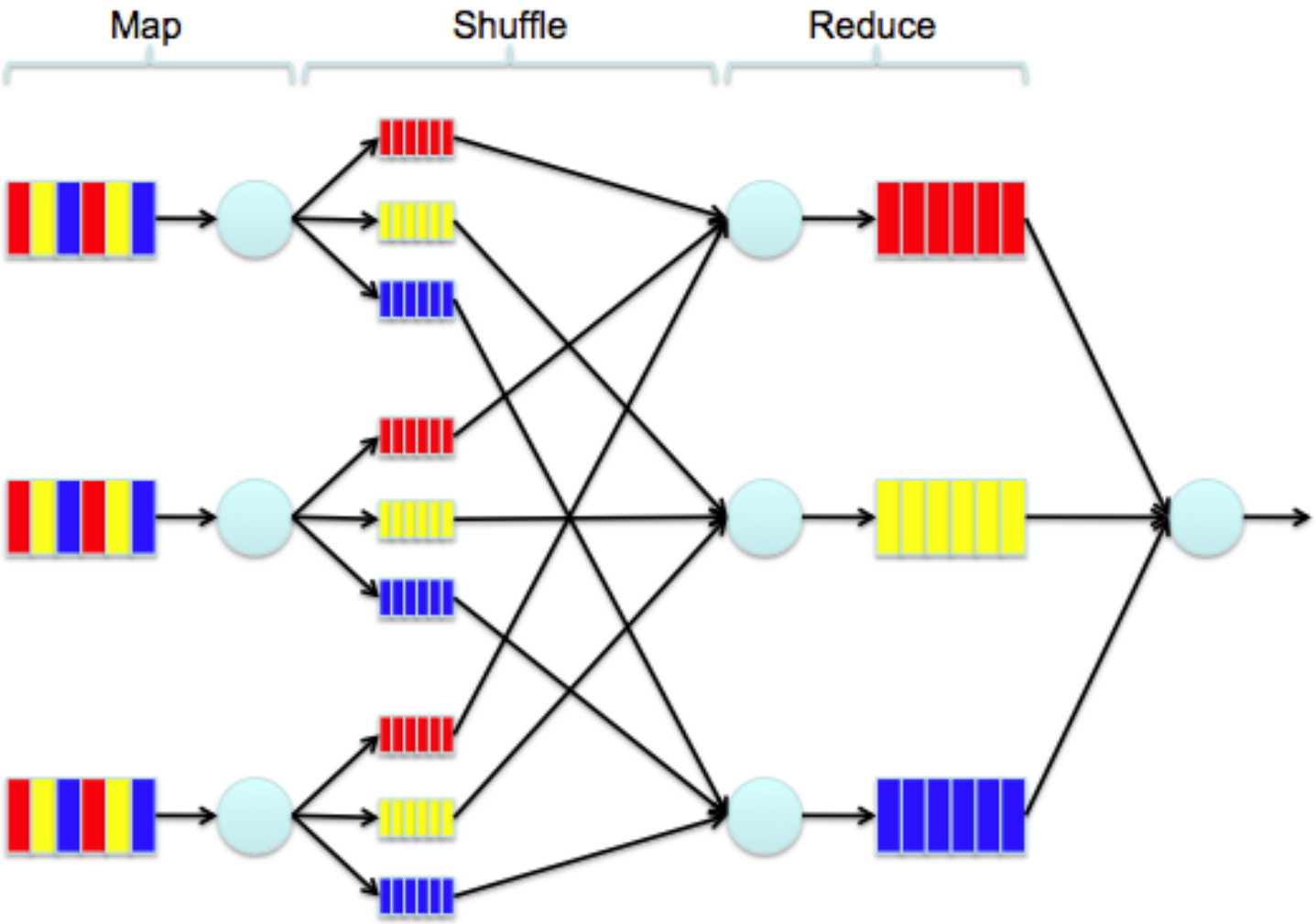
- Foco da aula de hoje:
  Processamento de grandes massas de dados na nuvem utilizando Apache Spark



Fonte: `https://blog.jejualan.com/wp-content/uploads/2018/03/cloud-computing-1924338_1280.png`

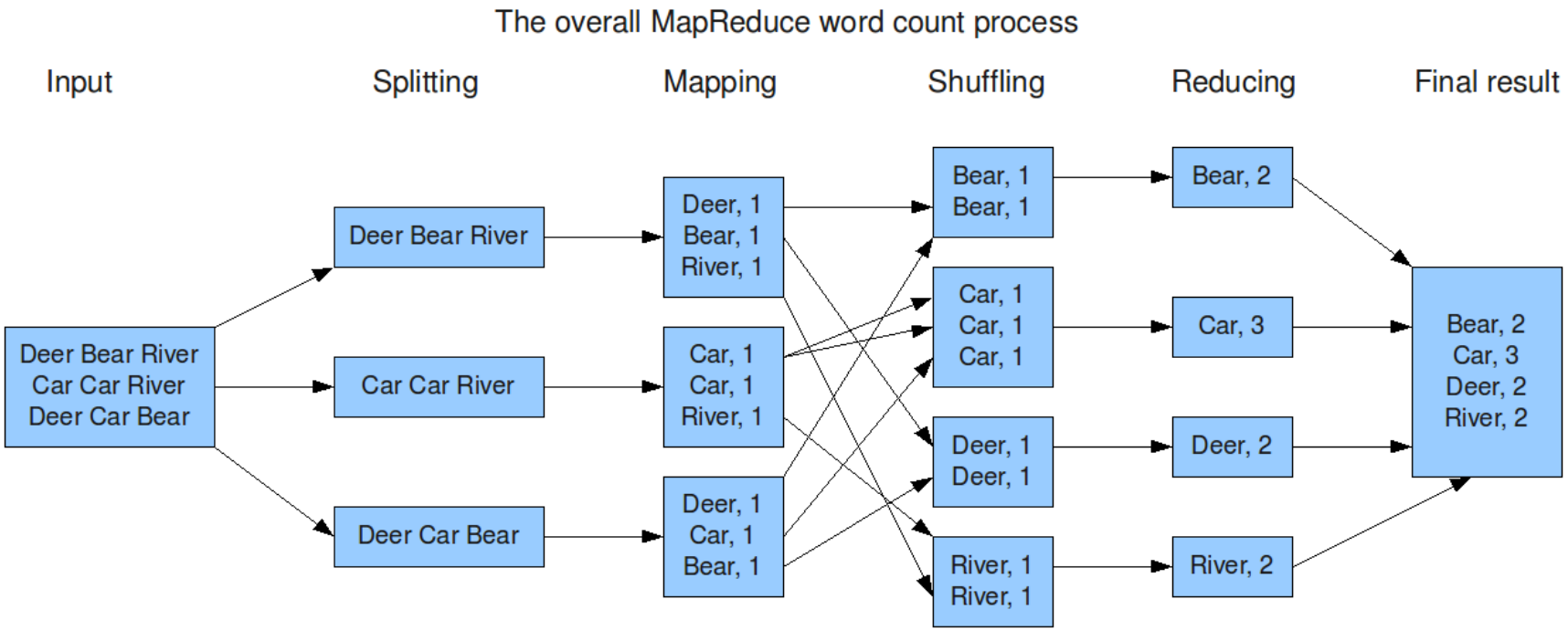
MapReduce

Visão colorida



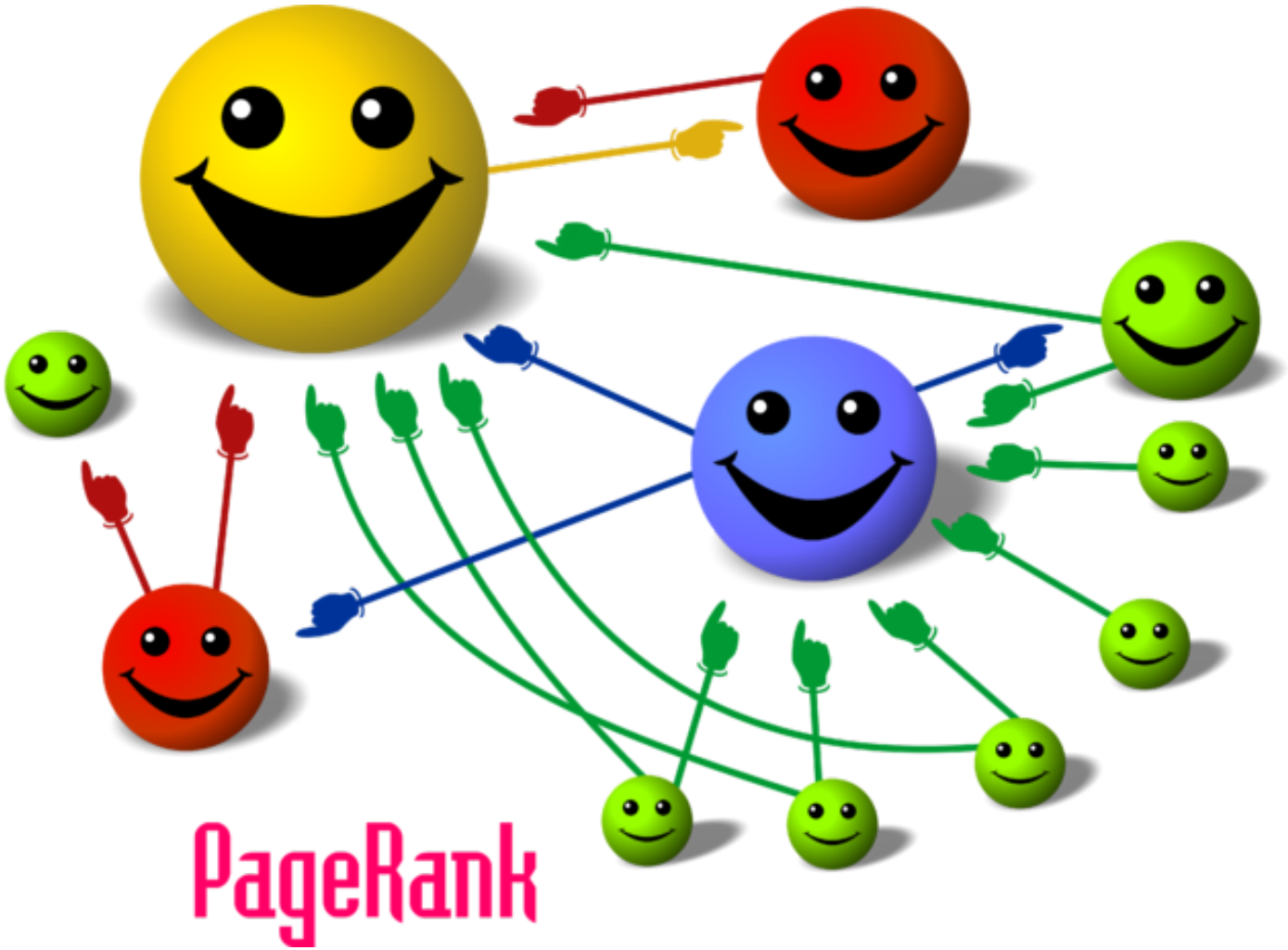
http://www.cs.uml.edu/~jlu1/doc/source/report/MapReduce.html

Word Count



http://www.cs.uml.edu/~jlu1/doc/source/report/img/MapReduceExample.png

Page Rank



https://commons.wikimedia.org/w/index.php?curid=2776582

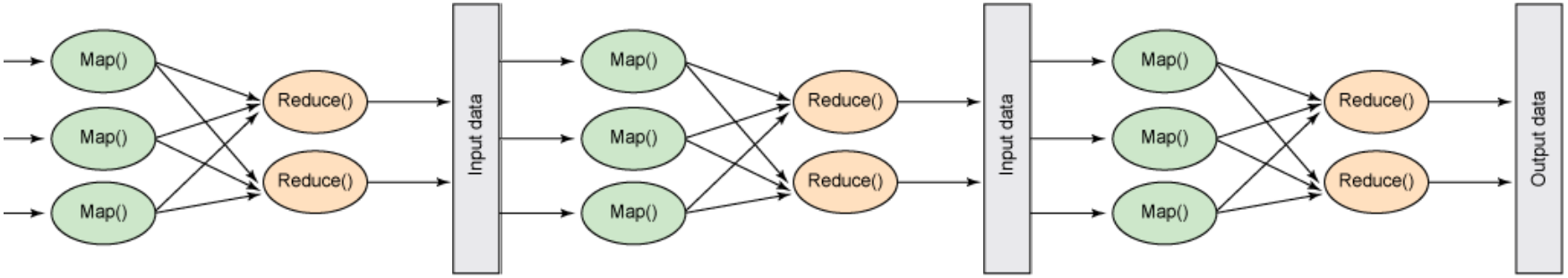
Page Rank

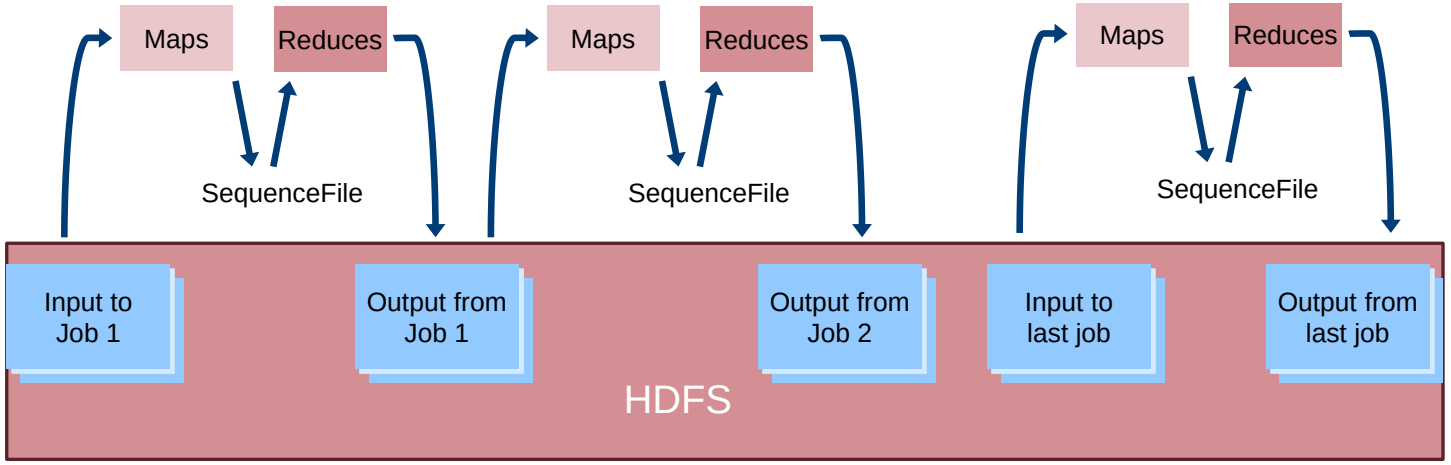
- Simula o comportamento de um *random sufer*
  - digita urls de tempos em tempos
  - segue *links* aleatoriamente

$$PR(x) = (1 - d) + d \sum_{i=1}^{N} PR(t_i)/L(t_i)$$

- $PR(x)$ page rank da página x
- d fator de amortecimento
- N número de páginas que apontam para a página x
- $L(t_i)$ número de links distintos que uma página aponta
- Várias iterações até a convergência

MapReduce: várias interações





Carol McDonald: An Overview of Apache Spark

Apache Spark™ - Lightning-Fast Cluster Computing - Mozilla Firefox

Apache Spark™ - Lightr

spark.apache.org 120% Search

APACHE Spark™

Lightning-fast cluster computing

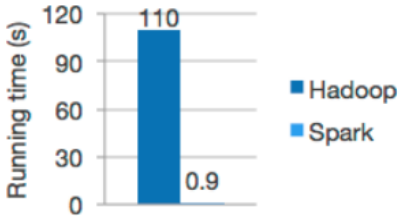
Download | Libraries | Documentation | Examples | Community | Developers | Apache Software Foundation

Apache Spark™ is a fast and general engine for large-scale data processing.

Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Apache Spark has an advanced DAG execution engine that supports acyclic data flow and in-memory computing.



Logistic regression in Hadoop and Spark

Latest News

Spark 2.1.1 released (May 02, 2017)

Spark Summit (June 5-7th, 2017, San Francisco) agenda posted (Mar 31, 2017)

Spark Summit East (Feb 7-9th, 2017, Boston) agenda posted (Jan 04, 2017)

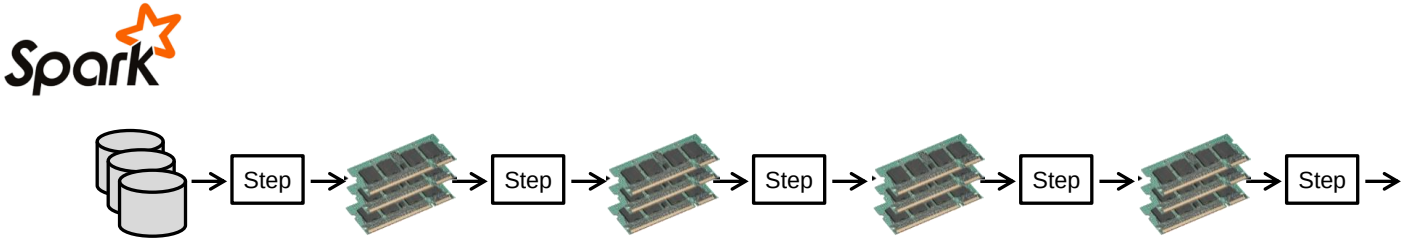
Spark 2.1.0 released (Dec 28, 2016)

Archive

Download Spark

Como o Spark consegue ser tão mais rápido?

Resilient Distributed Datasets

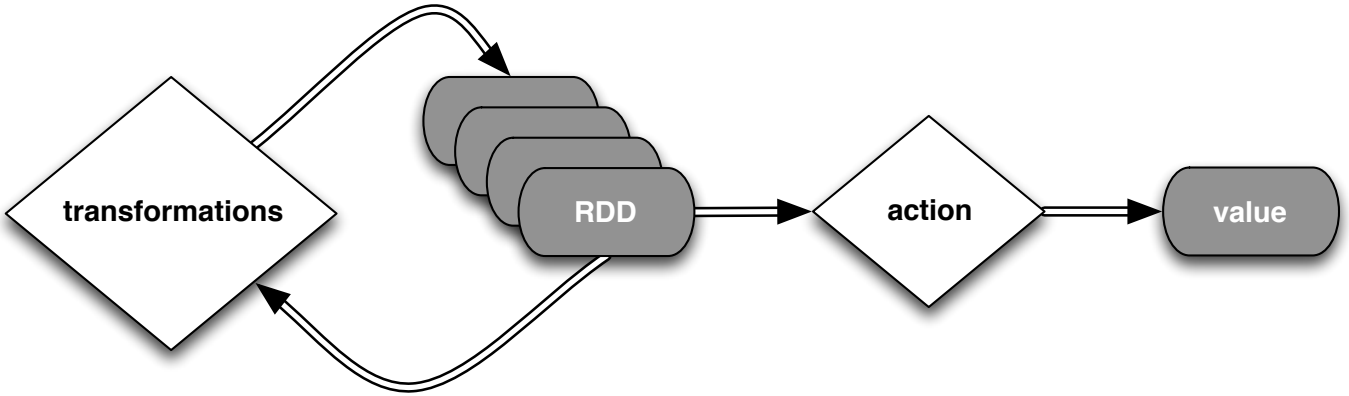


Carol McDonald: An Overview of Apache Spark

- RDD: principal abstração em Spark
- Imutável
- Tolerante a falhas

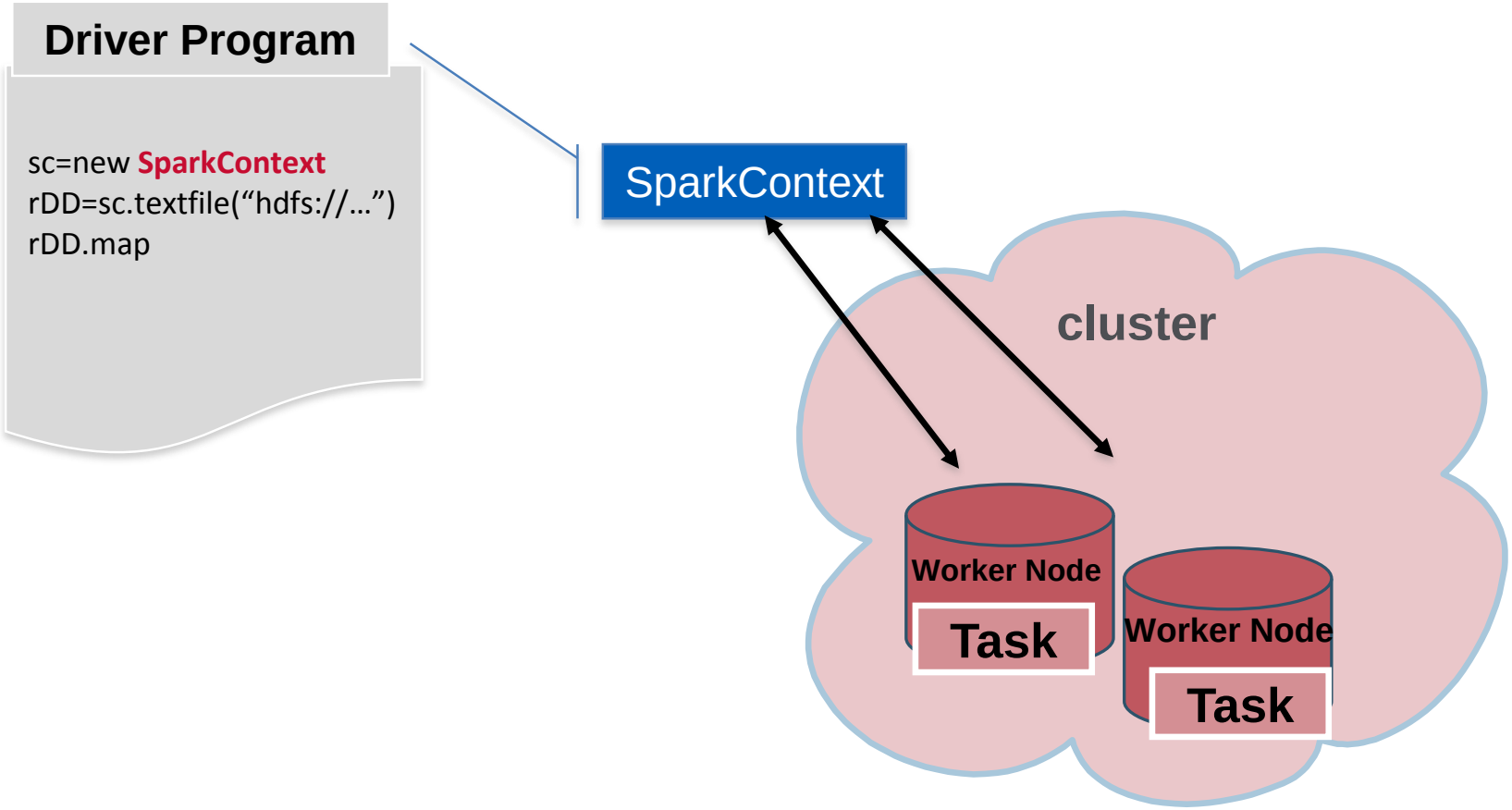
Operações com RDDs

- Muito mais do que Map e Reduce
- Transformações e Ações



http://databricks.com

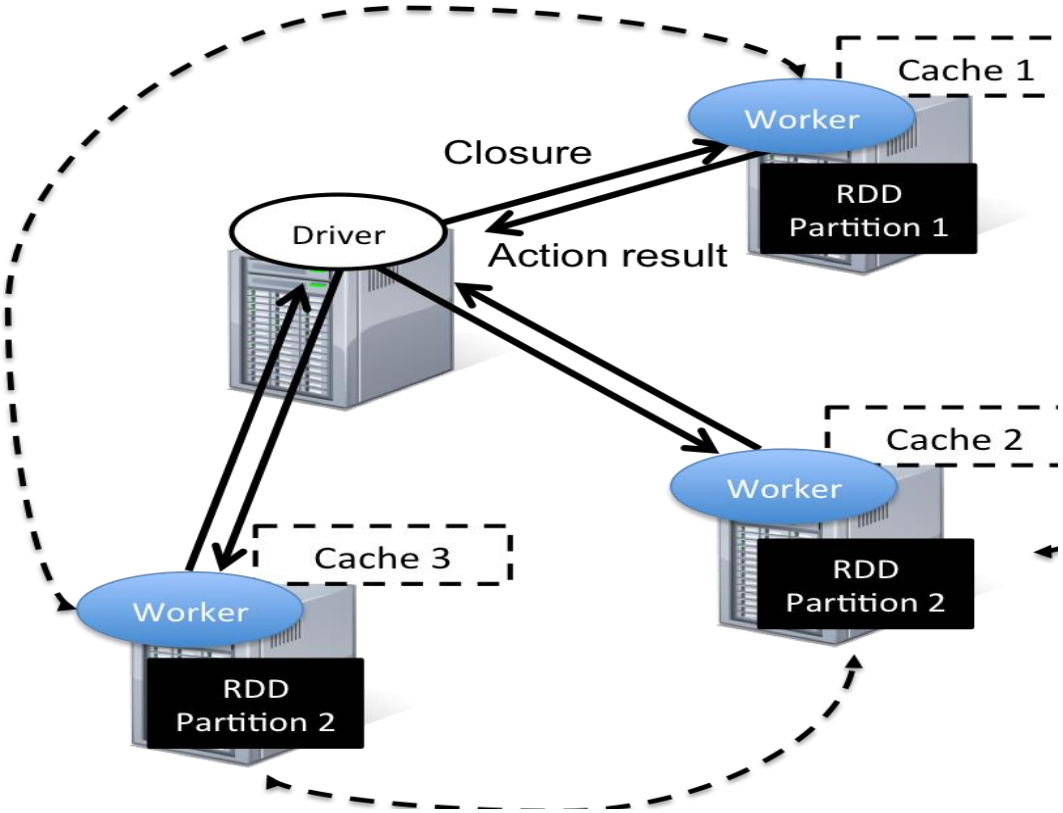
SparkContext



Carol McDonald: An Overview of Apache Spark

SparkContext

RDDs e partições



http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

Algumas transformações simples

`map(func)`	todo elemento do RDD original será transformado por *func*
`flatmap(func)`	todo elemento do RDD original será transformado em 0 ou mais itens por *func*
`filter(func)`	retorna elementos selecionados por *func*
`groupByKey()`	Dado um dataset (k, v) retorna (k, Iterable<v>)
`reduceByKey(func)`	Dado um dataset (k, v) retorna outro, com chaves agrupadas por func
`sortByKey(ascending)`	Dado um dataset (k,v) retorna outro, ordenado em ordem ascendente ou descendente

Veja mais em Spark Programming Guide: Transformations

Algumas ações

`count()`	retorna o número de elementos no dataset
`collect()`	retorna todos elementos do dataset
`take(n)`	retorna os n primeiros elementos do dataset

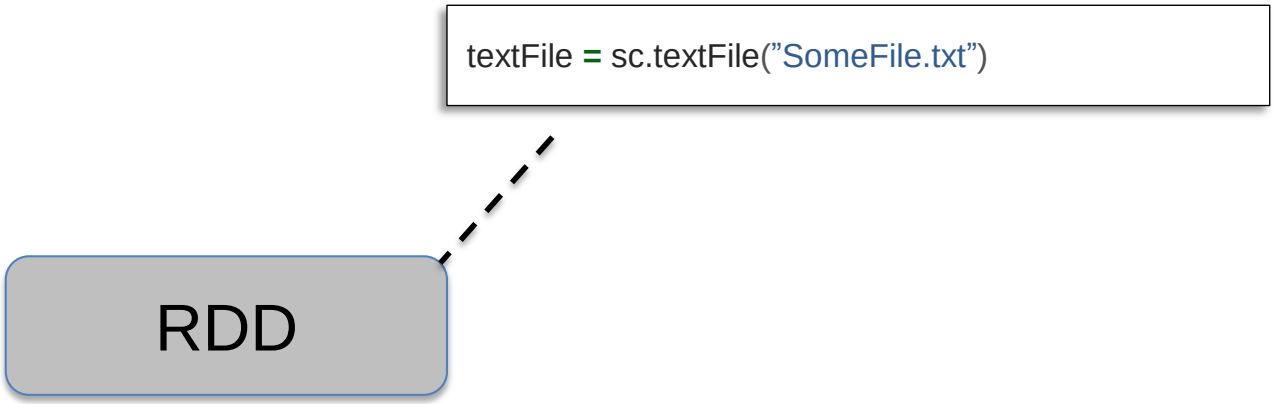
Veja mais em Spark Programming Guide: Actions

PySpark

- Spark pode ser utilizado com Scala, Java ou **Python**
  - Veja Spark Quick Start
- Pode ser mais fácil aprender com *shells*...
  - `python shell`
  - `pyspark`
- Instalação (bem simples!!!)

```
$ wget http://ftp.unicamp.br/pub/apache/spark/spark-2.3.1/spark-2.3.1-bin-hadoop2.7.tgz
$ tar spark-2.3.1-bin-hadoop2.7.tgz
$ cd spark-2.3.1-bin-hadoop2.7
$ bin/pyspark
```

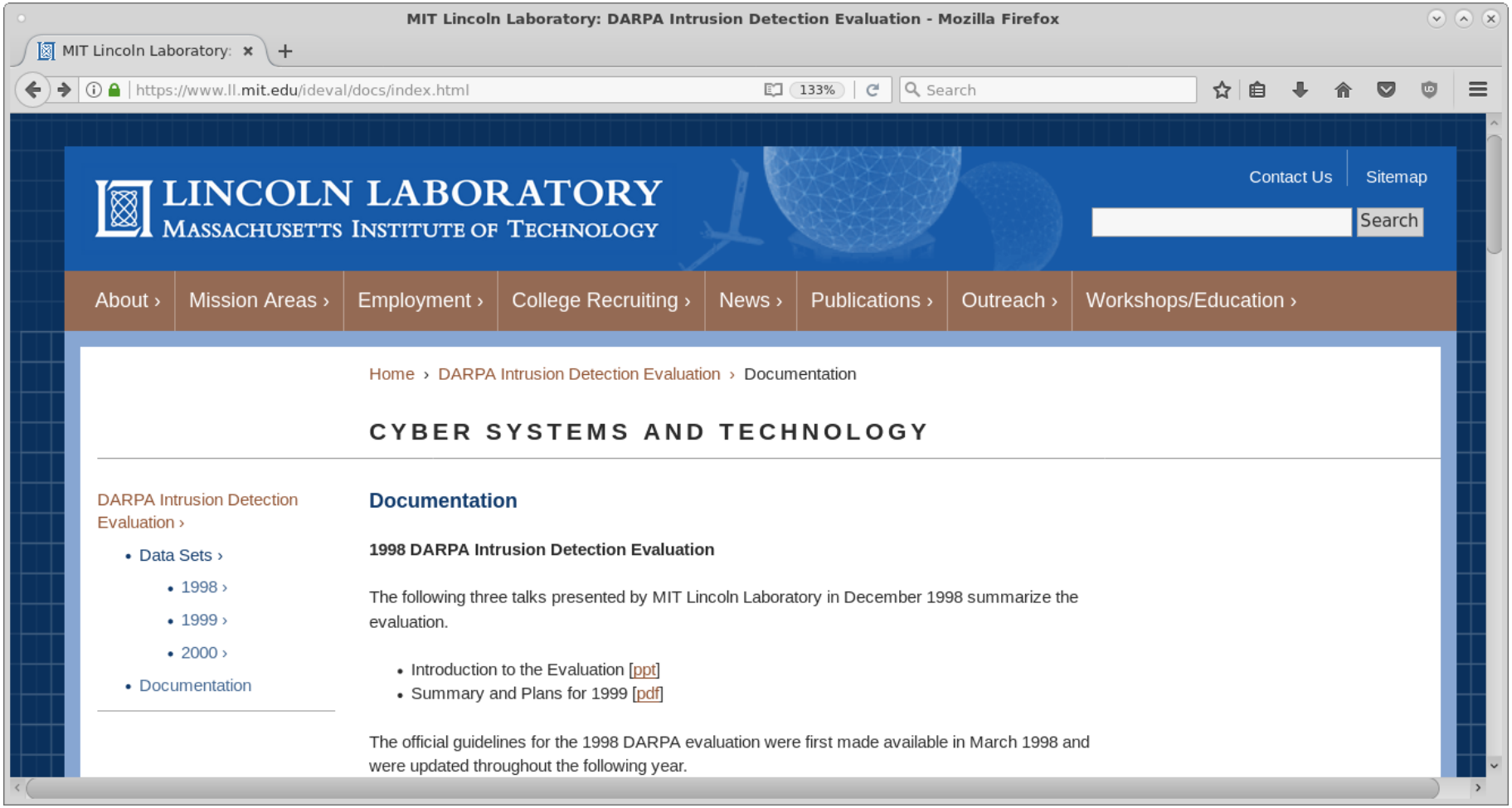
Primeiro RDD



Carol McDonald: An Overview of Apache Spark

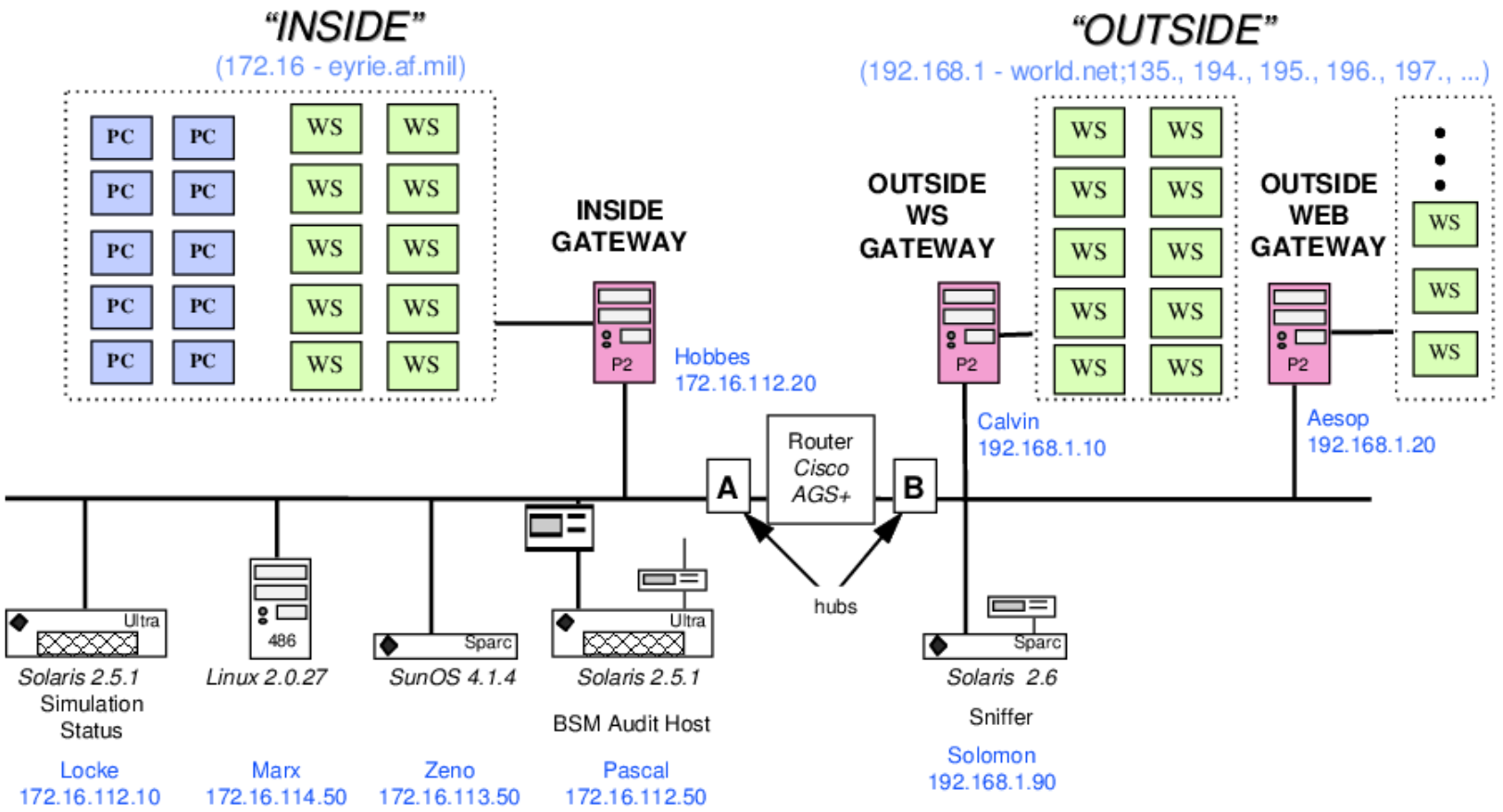
```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.1
      /_/

Using Python version 2.7.15 (default, May 16 2018 17:50:09)
SparkSession available as 'spark'.
>>> lines = sc.textFile("tcpdump.list");
```

DARPA Intrusion Detection Evaluation



Vários conjuntos de dados, com ataques documentados

DARPA Intrusion Detection Evaluation



Ethernet Hub Info
A: AsanteHub 1012 (no ip)
B: HP EtherTwist Hub PLUS (ip = 192.168.1.5)

Router Interface Info
A: 172.16.112.1
B: 192.168.1.1

https://www.ll.mit.edu/ideval/docs/index.html

DARPA Intrusion Detection Evaluation

	Start Date	Start Time	Duration	Serv	Src Port	Dest Port	Src IP	Dest IP	Attack Score	Name
1	01/27/1998	00:00:01	00:00:23	ftp	1755	21	192.168.1.30	192.168.0.20	0.31	-
2	01/27/1998	05:04:43	67:59:01	telnet	1042	23	192.168.1.30	192.168.0.20	0.42	-
3	01/27/1998	06:04:36	00:00:59	smtp	43590	25	192.168.1.30	192.168.0.40	12.0	-
4	01/27/1998	08:45:01	00:00:01	finger	1050	79	192.168.0.40	192.168.1.30	2.56	guess
5	01/27/1998	09:23:45	00:00:01	http	1031	80	192.168.1.30	192.168.0.40	-1.3	-
7	01/27/1998	15:11:32	00:00:12	sunrpc	2025	111	192.168.1.30	192.168.0.20	3.10	rpc
8	01/27/1998	21:53:17	00:00:45	exec	2032	512	192.168.1.30	192.168.0.40	2.95	exec
9	01/27/1998	21:58:21	00:00:01	http	1031	80	192.168.1.30	192.168.0.20	0.45	-
10	01/27/1998	22:57:53	26:59:00	login	2031	513	192.168.0.40	192.168.1.20	7.00	-
11	01/27/1998	23:57:28	130:23:08	shell	1022	514	192.168.1.30	192.168.0.20	0.52	guess
13	01/27/1998	25:38:00	00:00:01	eco/i	-	-	192.168.0.40	192.168.1.30	0.01	-

Como verificar as primeiras linhas de um RDD

Ação `take(n)`

```
>>> lines = sc.textFile("tcpdump.list")
>>> lines.take(5)
[u'1  06/02/1998 00:00:07 00:00:01 http 2127 80 172.016.114.207 152.163.214.011 0 -', u'2
06/02/1998 00:00:07 00:00:01 http 2139 80 172.016.114.207 152.163.212.172 0 -', u'3  06/02/1998
00:00:07 00:00:01 http 2128 80 172.016.114.207 152.163.214.011 0 -', u'4  06/02/1998 00:00:07
00:00:01 http 2129 80 172.016.114.207 152.163.214.011 0 -', u'5  06/02/1998 00:00:07 00:00:01
http 2130 80 172.016.114.207 152.163.214.011 0 -']
```

```
>>> for x in lines.take(5) :
...     print x
...
1  06/02/1998 00:00:07 00:00:01 http 2127 80 172.016.114.207 152.163.214.011 0 -
2  06/02/1998 00:00:07 00:00:01 http 2139 80 172.016.114.207 152.163.212.172 0 -
3  06/02/1998 00:00:07 00:00:01 http 2128 80 172.016.114.207 152.163.214.011 0 -
4  06/02/1998 00:00:07 00:00:01 http 2129 80 172.016.114.207 152.163.214.011 0 -
5  06/02/1998 00:00:07 00:00:01 http 2130 80 172.016.114.207 152.163.214.011 0 -
>>>
```

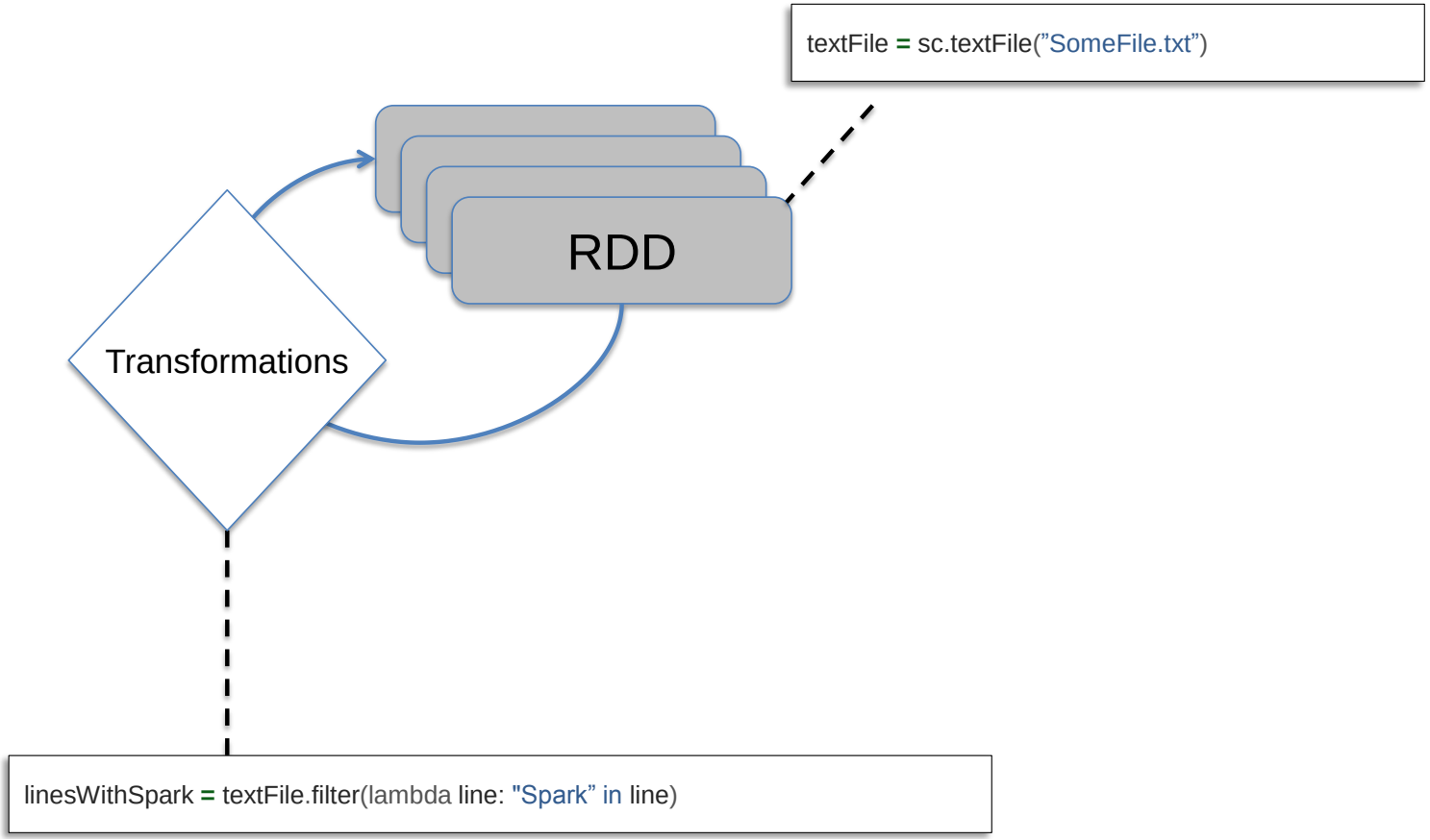
Como listar um RDD inteiro

Ação `collect()`

```
>>> for x in lines.collect() :
...     print x
...
1  06/02/1998 00:00:07 00:00:01 http 2127 80 172.016.114.207 152.163.214.011 0 -
2  06/02/1998 00:00:07 00:00:01 http 2139 80 172.016.114.207 152.163.212.172 0 -
3  06/02/1998 00:00:07 00:00:01 http 2128 80 172.016.114.207 152.163.214.011 0 -
4  06/02/1998 00:00:07 00:00:01 http 2129 80 172.016.114.207 152.163.214.011 0 -
5  06/02/1998 00:00:07 00:00:01 http 2130 80 172.016.114.207 152.163.214.011 0 -
6  06/02/1998 00:00:07 00:00:01 http 2131 80 172.016.114.207 152.163.214.011 0 -
7  06/02/1998 00:00:07 00:00:01 http 2132 80 172.016.114.207 152.163.214.011 0 -
8  06/02/1998 00:00:07 00:00:01 http 2136 80 172.016.114.207 152.163.214.011 0 -
9  06/02/1998 00:00:07 00:00:01 http 2137 80 172.016.114.207 152.163.212.172 0 -
10  06/02/1998 00:00:07 00:00:01 http 2138 80 172.016.114.207 152.163.212.172 0 -
11  06/02/1998 00:00:07 00:00:01 http 2140 80 172.016.114.207 152.163.214.011 0 -
12  06/02/1998 00:00:07 00:00:01 http 2141 80 172.016.114.207 152.163.214.011 0 -
13  06/02/1998 00:00:07 00:00:01 http 2177 80 172.016.114.207 152.163.212.172 0 -
14  06/02/1998 00:00:07 00:00:01 http 2178 80 172.016.114.207 152.163.214.011 0 -
15  06/02/1998 00:00:07 00:00:01 http 2242 80 172.016.114.207 152.163.214.011 0 -
16  06/02/1998 00:00:59 00:00:01 ntp/u 123 123 172.016.112.020 192.168.001.010 0 -
17  06/02/1998 00:01:01 00:00:01 eco/i - - 192.168.001.005 192.168.001.001 0 -
18  06/02/1998 00:01:21 00:00:01 http 2305 80 172.016.114.207 207.077.090.015 0 -
19  06/02/1998 00:01:22 00:00:01 http 2306 80 172.016.114.207 207.077.090.013 0 -
20  06/02/1998 00:02:32 00:00:01 http 2307 80 172.016.114.207 152.163.214.011 0 -
21  06/02/1998 00:02:33 00:00:01 http 2376 80 172.016.114.207 152.163.214.011 0 -
22  06/02/1998 00:02:33 00:00:01 http 2314 80 172.016.114.207 152.163.214.011 0 -
23  06/02/1998 00:02:33 00:00:01 http 2590 80 172.016.114.207 152.163.212.172 0 -
24  06/02/1998 00:02:33 00:00:01 http 2377 80 172.016.114.207 152.163.214.011 0 -
25  06/02/1998 00:02:33 00:00:01 http 2378 80 172.016.114.207 152.163.214.011 0 -
26  06/02/1998 00:02:33 00:00:01 http 2441 80 172.016.114.207 152.163.214.011 0 -
27  06/02/1998 00:02:33 00:00:01 http 2505 80 172.016.114.207 152.163.214.011 0 -
```

Como filtrar um RDD

Transformação `filter()`



Carol McDonald: An Overview of Apache Spark

Revisão rápida de Python

Funções lambda e `filter()`

Funções lambda: funções que não recebem um nome em tempo de execução

```
>>> def impar(x) :
...     return x % 2 != 0

>>> lista = range(1,10)
>>> print lista
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> filter (impar, lista)
[1, 3, 5, 7, 9]

>>> filter (lambda x: x % 2 != 0, lista)
[1, 3, 5, 7, 9]
```

Como filtrar um RDD

```
>>> lines = sc.textFile("tcpdump.list")

>>> telnet = lines.filter(lambda x: "telnet" in x)
>>> for x in telnet.collect():
...     print x

>>> http = lines.filter(lambda x: "http" in x)
>>> http.count()
```

Revisão rápida de Python

Operações com Strings

```
Comandos                        Saída
astring = "Spark"
print astring                   Spark
print len(astring)              5
print astring[0]                S
print astring[1:3]              pa
print astring[3:]               rk
print astring[0:5:2]            Sak
print astring[::-1]             krapS
```

Python: Mais operações com Strings

Comandos	Saída
`uline = u" GNU is not Unix.  "`	
`l = [uline]`	
`print l`	`[u'  GNU is not Unix  ']`
`line = str(l[0])`	
`l = [line]`	
`print l`	`['  GNU is not Unix  ']`
`line = line.strip()`	
`print line`	`GNU is not Unix.`
`words = line.split()`	
`print words`	`['GNU', 'is', 'not', 'Unix']`
`print words[1]`	`is`

Como trabalhar com pares `(chave, valor)`

Como encontrar o serviço mais utilizado

```
>>> pairs = lines.map(lambda x: (str(x.split()[4]), 1))

>>> totalByService = pairs.reduceByKey(lambda a,b: a + b)

>>> inverted = totalByService.map(lambda (k,v) : (v,k))

>>> sortedPairs = pairs.sortByKey(False)
```

Escrevendo scripts para execução fora da Shell

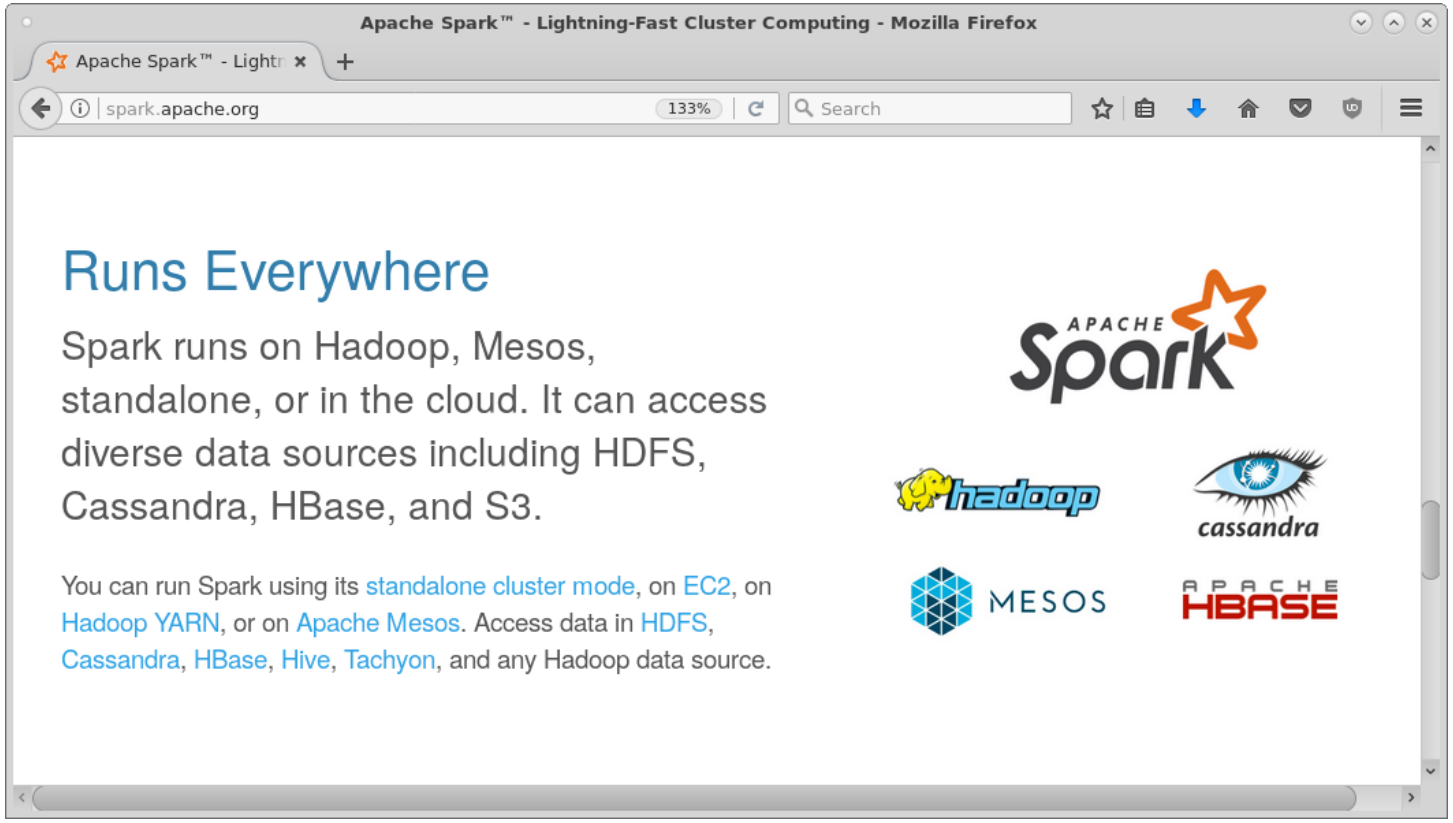
```
# -*- coding: utf-8 -*-
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName("Pyspark Pgm")
sc = SparkContext(conf = conf)

lines = sc.textFile("tcpdump.list")

for x in lines.collect() :
   print x

$ bin/spark-submit script.py
```

Detecção de intrusões

Quais dados seriam interessantes?

- Tentativas de acesso a serviços pouco seguros?
- Muitos acessos por hora de um dado serviço?
- Lista de conexões de curta duração?
- ...

Qual a plataforma mais adequada?



Databricks

- Empresa fundada pelo time que criou o Spark
- Community edition
  - Ambiente para experimentos iniciais
  - Uso gratuito
  - Mini 6GB cluster
  - Confira esta opção em
    `https://databricks.com/try-databricks`

Laboratório

- Instale o Spark
- Obtenha uma versão do darpa dataset ou outro arquivo de complexidade similar
- Elabore questões interessantes e opere com os dados
  - Uso de transformações: `map`, `reduceByKey`, `sortByKey`
- Entrega de código e relatório via Moodle
- Veja mais intruções em `http://www.ic.unicamp.br/~islene/2018-inf550/explorando-spark.html`

Referências

- Python Tutorial
- Apache Spark
  - Spark Programming Guide
- Clash of Titans: MapReduce vs. Spark for Large Scale Data Analytics, Juwei Shi e outros, IBM Research, China