Big Data e Machine Learning com Hadoop e Spark

Conteúdo

CONTEÚDO PROGRAMÁTICO

- Visão geral da ciência de dados e aprendizado de máquina em escala
- Visão geral do ecossistema do Hadoop
- Instalação de um Cluster Hadoop
- Trabalhando com dados do HDFS e tabelas do Hive usando o Hue
- Visão geral do Python
- Visão geral do R
- Visão geral do Apache Spark 2
- Leitura e gravação de dados
- Inspeção da qualidade dos dados
- Limpeza e transformação de dados
- Resumindo e agrupando dados
- Combinando, dividindo e remodelando dados
- Explorando dados
- Configuração, monitoramento e solução de problemas de aplicativos Spark
- Visão geral do aprendizado de máquina no Spark MLlib
- Extraindo, transformando e selecionando recursos
- Construindo e avaliando modelos de regressão
- Construindo e avaliando modelos de classificação
- Construindo e avaliando modelos de cluster
- Validação cruzada de modelos e hiperparâmetros de ajuste
- Construção de pipelines de aprendizado de máquina
- Implantando modelos de aprendizado de máquina

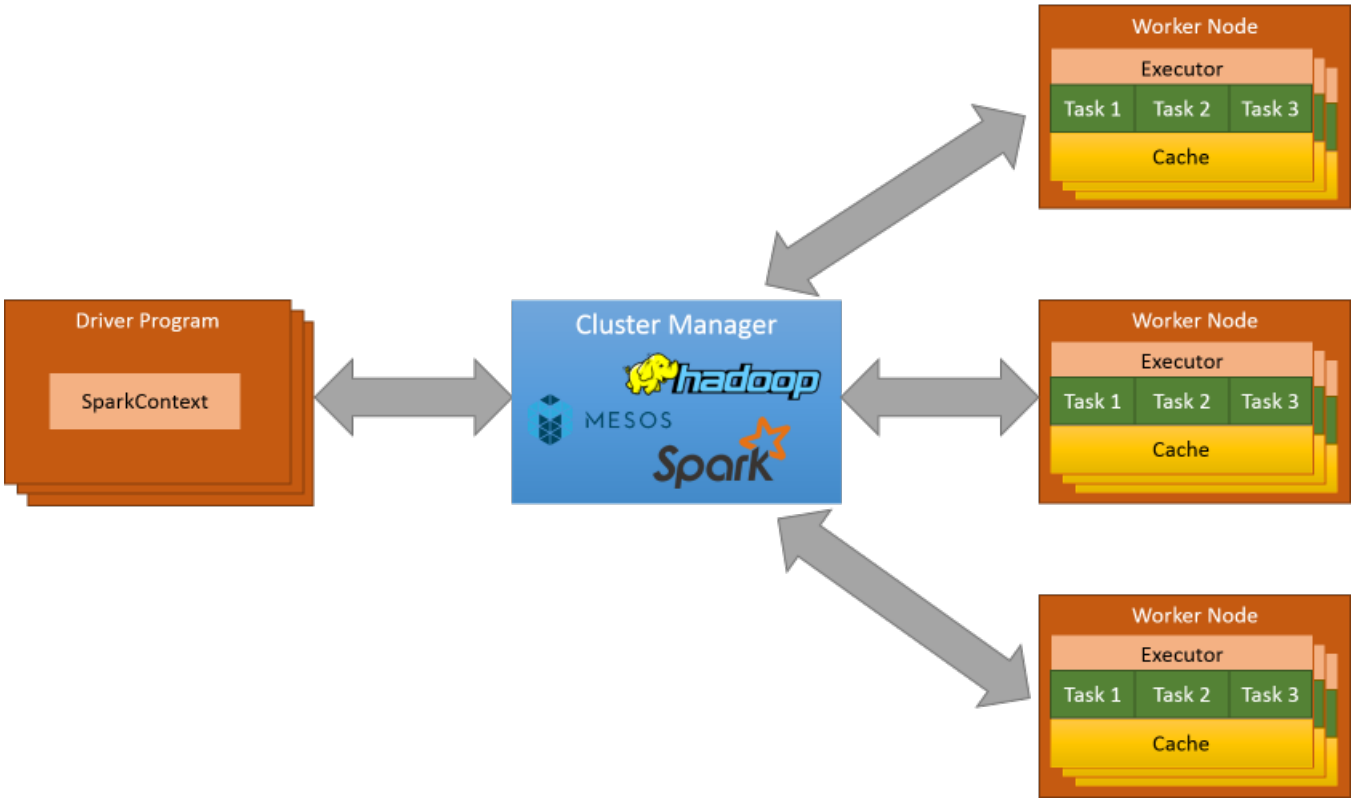
MATERIAL DIDÁTICO

- Slides do treinamento em PDF
- GitHub com exercícios e códigos exemplo
- Máquinas virtuais para exercícios simulados
- Gravação das aulas disponível durante 3 meses

O que é o Spark?

- "Um mecanismo rápido e geral para processamento de dados em larga escala"

Escalável


Worker Node
Executor
Task 1
Task 2
Task 3
Cache
Driver Program
SparkContext
Cluster Manager
hadoop
MESOS
Spark
Worker Node
Executor
Task 1
Task 2
Task 3
Cache
Worker Node
Executor
Task 1
Task 2
Task 3
Cache

Rápido

- "Executa programas até 100x mais rápido que o Hadoop MapReduce na memória ou 10x mais rápido no disco. "
- O mecanismo DAG (gráfico acíclico direcionado) otimiza os fluxos de trabalho

Preferido por muitos

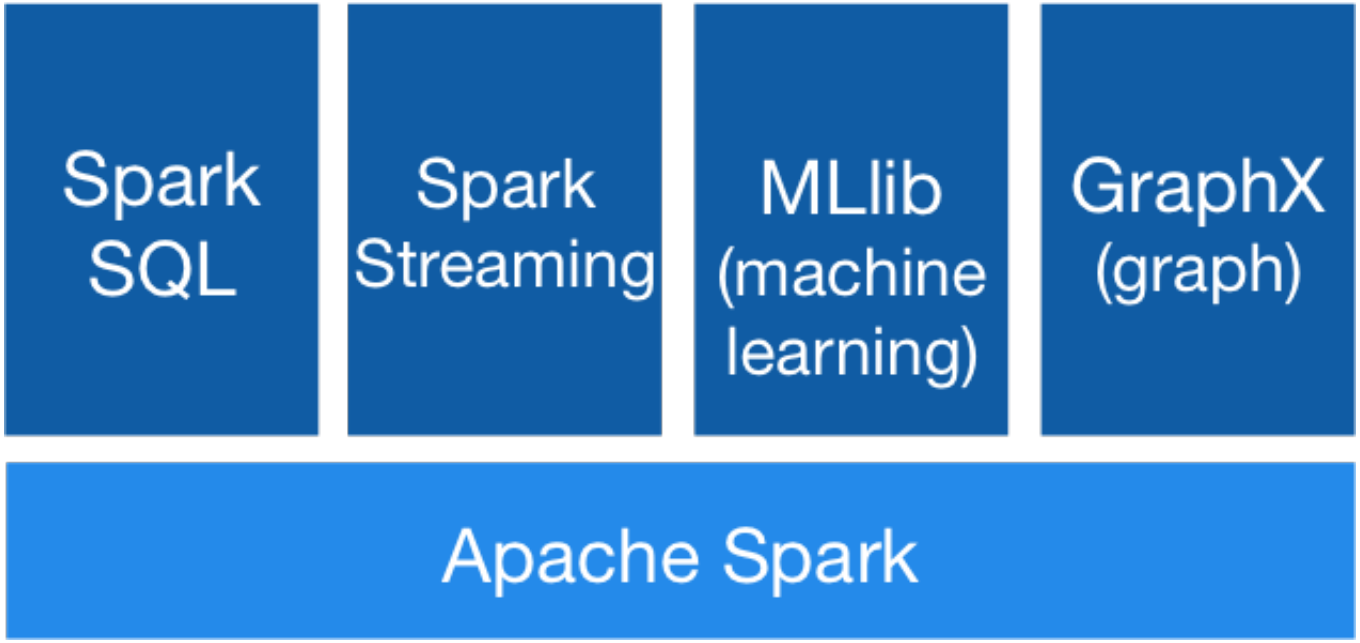
- Amazon
- Ebay: log analysis and aggregation
- NASA JPL: Deep Space Network
- Groupon
- TripAdviser
- Yahoo
- Outros:

https://cwiki.apache.org/confluence/display/SPARK/Powered+By+Spark

Não é tão complicado...

- Código em Python, Java ou Scala
- Construído em torno de um conceito principal: o Conjunto de dados distribuído resiliente (RDD)

Componentes



Vamos usar Python

- Por que Python?

- É muito mais simples e isso é apenas uma visão geral.
- Não precisa compilar nada, lidar com JARs, dependências, etc.

- Mas ...

- O própria Spark está escrito em Scala
- O modelo de programação funcional do Scala é adequado para distribuir em processamento
- Oferece desempenho rápido (Scala compila para bytecode Java)
- Menos código e material do que Java
- Python é lento

Sem medo...

- O código Scala no Spark parece muito com o código Python.
  - Código Python para números quadrados em um conjunto de dados:

```
nums = sc.parallelize ([1, 2, 3, 4])
quadrado = nums.map (lambda x: x *
x) .collect ()
```

  - Código Scala para números quadrados em um conjunto de dados:

```
val nums = sc.parallelize (List (1, 2, 3, 4))
val squared = nums.map (x => x * x) .collect ()
```

O que é o RDD?

- Resilient
- Distributed
- Dataset

O Contexto Spark (SparkContext)

- Criado pelo seu programa de driver
- É responsável por tornar resiliente e distribuído o RDD!
- Cria RDD's
- O shell Spark cria um objeto "sc" para você

Criando RDD's

- nums = parallelize ([1, 2, 3, 4])
- sc.textFile ("file: ///c: /users/frank/gobs-o-text.txt")
  - ou s3n: //, hdfs: //
- hiveCtx = HiveContext (sc) rows = hiveCtx.sql ("SELECT nome, idade FROM usuários")
- também pode criar a partir de:
  JDBC
  Cassandra
  HBase
  Elastisearch
  JSON, CSV, arquivos de sequência, arquivos de objetos, vários formatos compactados

Transformando RDD's

- map
- flatmap
- filter
- distinct
- sample
- union, intersection, subtract, cartesian

Exemplo map

- rdd = sc.parallelize([1, 2, 3, 4])
- squaredRDD = rdd.map(lambda x: x*x)
- Resultando em 1, 4, 9, 16

O que é essa tal de lambda???

Muitos métodos RDD aceitam uma função como um parâmetro

```
rdd.map (lambda x: x * x)
```

É a mesma coisa que

```
def squareIt (x):
    return x * x
rdd.map (squareIt)
```

Assim, você agora entende de programação funcional!!!!

Ações RDD

- collect
- count
- countByValue
- take
- top
- reduce
- ... e outras ...

Lazy evaluation

- Nada realmente acontece no seu programa de driver até que uma ação seja chamada!

Obrigado!!!

Nos vemos amanhã!!!

Bom descanso!