Big Data e Machine Learning com Hadoop e Spark

Conteúdo

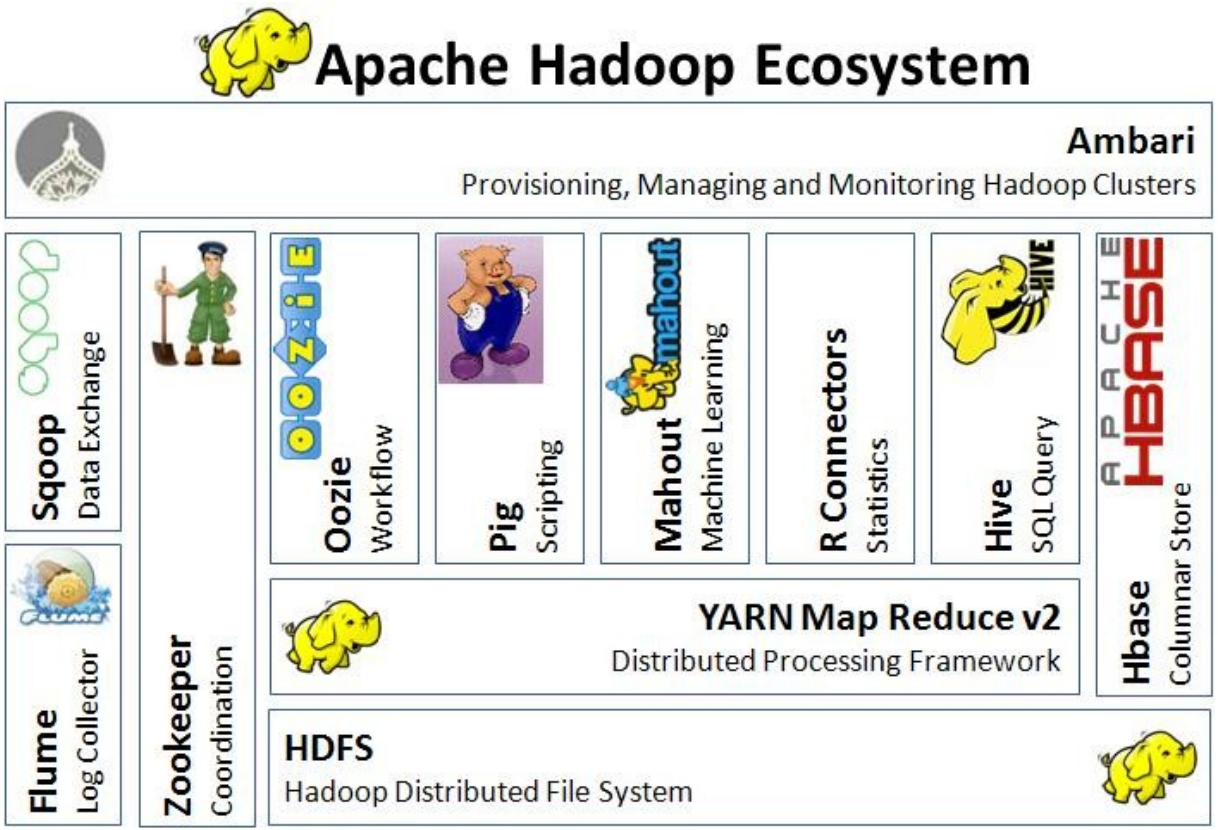
CONTEÚDO PROGRAMÁTICO

- 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

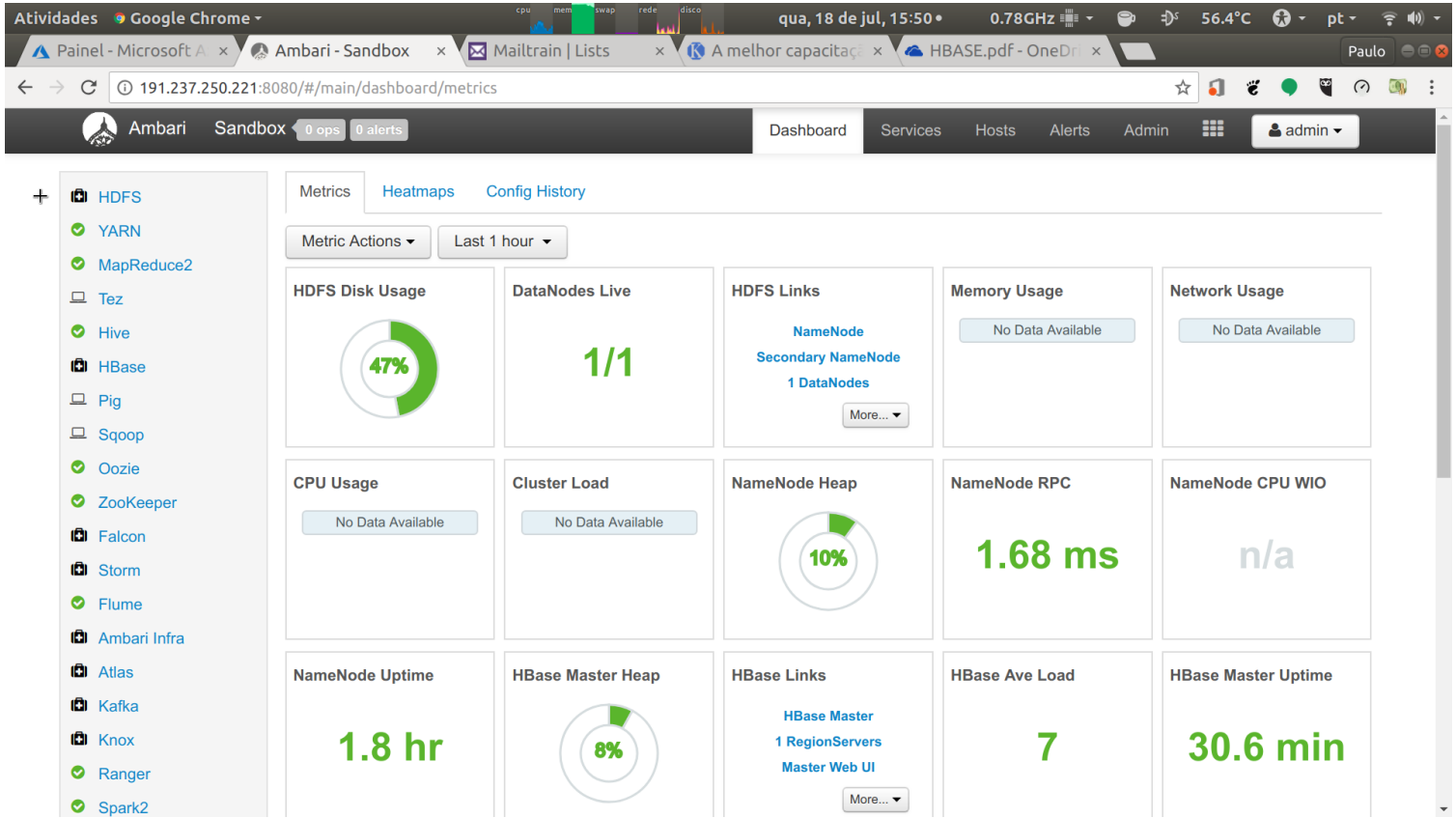
MATERIAL DIDÁTICO

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

Apache Software Foundation



Ambari


Atividades
Google Chrome
qua, 18 de jul, 15:50
0.78GHz
56.4°C
pt
Painel - Microsoft A
Ambari - Sandbox
Mailtrain | Lists
A melhor capacitaç
HBASE.pdf - OneDr
Paulo
191.237.250.221:8080/#/main/dashboard/metrics
Ambari
Sandbox
0 ops
0 alerts
Dashboard
Services
Hosts
Alerts
Admin
admin
HDFS
YARN
MapReduce2
Tez
Hive
HBase
Pig
Sqoop
Oozie
ZooKeeper
Falcon
Storm
Flume
Ambari Infra
Atlas
Kafka
Knox
Ranger
Spark2
Metrics
Heatmaps
Config History
Metric Actions
Last 1 hour
HDFS Disk Usage
47%
DataNodes Live
1/1
HDFS Links
NameNode
Secondary NameNode
1 DataNodes
More...
Memory Usage
No Data Available
Network Usage
No Data Available
CPU Usage
No Data Available
Cluster Load
No Data Available
NameNode Heap
10%
NameNode RPC
1.68 ms
NameNode CPU WIO
n/a
NameNode Uptime
1.8 hr
HBase Master Heap
8%
HBase Links
HBase Master
1 RegionServers
Master Web UI
More...
HBase Ave Load
7
HBase Master Uptime
30.6 min

Yet Another Resource Negotiator

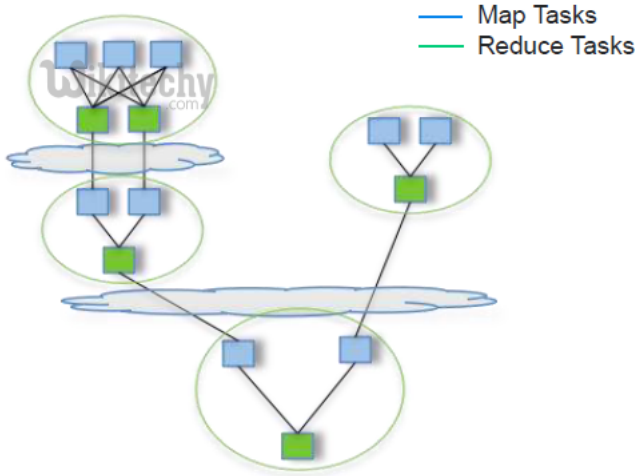
- Introduzido no Hadoop 2
- Isola o problema de gerenciar recursos no cluster de MapReduce
- Habilita alternativas MapReduce (Spark, Tez) construídas no topo de YARN

TEZ

- Directed Acyclic Graph Framework
- Isola o problema de gerenciar recursos no cluster de MapReduce
- Habilita alternativas MapReduce (Spark, Tez) construídas no topo do YARN
- Torna seus trabalhos Hive, Pig ou MapReduce mais rápidos!
- É um framework de aplicativo para a qual os clientes podem codificar como substitutos para MapReduce
- Constrói gráficos acíclicos direcionados (DAGs) para um processamento mais eficiente de trabalhos distribuídos
- Depende de uma visão mais holística do seu trabalho; elimina etapas desnecessárias e dependências.
- Otimiza o fluxo de dados físicos e o uso de recursos

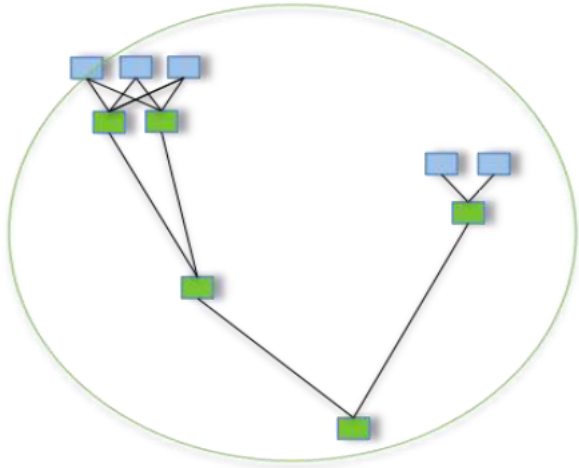
TEZ

MAPREDUCE



- Mapper and Reducer phases
- Shuffle between mapper and reducer tasks
- JobControl to run group of jobs with dependencies

TEZ



- Directed Acyclic Graph (DAG) with vertices
- Shuffle/One-One/Broadcast between vertex tasks
- Whole plan runs in a single DAG

HIVE

- Traduz consultas SQL para trabalhos MapReduce ou Tez no seu cluster

Por que usar?

- Usa sintaxe familiar de SQL (HiveQL)
- Interativo
- Escalável - funciona com “big data” em um grupo
  - Realmente mais apropriado para aplicações de armazém de dados
- consultas fáceis OLAP - maneira mais fácil do que escrevendo MapReduce em Java
- Altamente otimizado
- Altamente extensível
  - Funções definidas pelo usuário
  - Servidor Thrift
  - Driver JDBC / ODBC

Por que NÃO usar?

- Alta latência – não apropriada para OLTP
- Armazena dados desnormalizados
- SQL é limitada no que pode fazer – PIG, Spark permitem coisas mais complexas
- Nenhuma transação
- Sem atualizações, inserções, exclusões em nível de registro

Banco de dados escalável e não relacional construído em HDFS

- CRUD
- Não tem linguagem de query, só uma API de CRUD

Hbase Data Model

- Acesso rápido a qualquer ROW
- Uma linha é referenciada por uma única chave
- Cada ROW tem um pequeno número de COLUMN FAMILIES
- UMA FAMÍLIA DE COLUNA pode conter COLUNAS arbitrárias
- Você pode ter um número muito grande de COLUMNS em uma COLUMN FAMILY
- Cada CELL pode ter muitas VERSÕES com determinados timestamps
- Dados esparsos são A-OK - as colunas ausentes em uma linha não consomem armazenamento.

Algumas maneiras de acessar o HBase

- HBase shell
- Java API
- – Wrappers para Python, Scala, etc.
- Spark, Hive, Pig
- REST service
- Thrift service
- Avro service

Por que o Pig?

- Escrever mappes e reducers à mão leva muito tempo.
- Pig apresenta Pig Latin, um script linguagem que permite usar sintaxe SQL-like para definir seu mapa e reduzir passos.
- Altamente extensível com user-defined funções (UDFs)

Exemplo

```
ratings = LOAD '/user/maria_dev/data/u.data' AS
(userID:int, movieID:int, rating:int, ratingTime:int);
```

- Isso cria uma ***relation*** chamada "ratings" com um determinado esquema.

```
(660,229,2,891406212)
(421,498,4,892241344)
(495,1091,4,888637503)
(806,421,4,882388897)
(676,538,4,892685437)
(721,262,3,877137285)
```

Exemplo

Use PigStorage se você precisar de um delimitador diferente.

```
metadata = LOAD '/user/maria_dev/data/u.item' USING
        PigStorage('|')AS (movieID:int, movieTitle:chararray,
        releaseDate:chararray, videoRelease:chararray,
        imdbLink:chararray);
DUMP metadata;
```

```
(1,Toy Story (1995),01-Jan-1995,,http://us.imdb.com/M/title-exact?Toy%20Story%20(1995
(2,GoldenEye (1995),01-Jan-1995,,http://us.imdb.com/M/title-exact?GoldenEye%20(1995)
(3,Four Rooms (1995),01-Jan-1995,,http://us.imdb.com/M/title-exact?Four%20Rooms
%20(1995))
(4,Get Shorty (1995),01-Jan-1995,,http://us.imdb.com/M/title-exact?Get%20Shorty%20(1
(5,Copycat (1995),01-Jan-1995,,http://us.imdb.com/M/title-exact?Copycat%20(1995))
```

Exemplo

Criando uma ***relation*** de outra ***relation***; **FOREACH / GENERATE**

```
metadata = LOAD '/user/maria_dev/ml-100k/u.item' USING PigStorage('|')
        AS (movieID:int, movieTitle:chararray, releaseDate:chararray,
        videoRelease:chararray, imdbLink:chararray);

nameLookup = FOREACH metadata GENERATE movieID, movieTitle,
        ToUnixTime(ToDate(releaseDate, 'dd-MMM-yyyy')) AS releaseTime;
```

(1,Toy Story (1995),01-Jan-1995,,http://us.imdb.com/M/title-exact?Toy%20Story%20(199

↓

(1,Toy Story (1995),788918400)

Exemplo

Group By

```
ratingsByMovie = GROUP ratings BY movieID;

DUMP ratingsByMovie;
```

```
(1,{(807,1,4,892528231),(554,1,3,876231938),(49,1,2,888068651), ... }
(2,{(429,2,3,882387599),(551,2,2,892784780),(774,2,1,888557383), ... }
```

Exemplo

```
ratingsByMovie = GROUP ratings BY movieID;

avgRatings = FOREACH ratingsByMovie GENERATE group AS movieID,
AVG(ratings.rating) AS avgRating;

DUMP avgRatings;
```

```
(1,3.8783185840707963)
(2,3.2061068702290076)
(3,3.033333333333333)
(4,3.550239234449761)
(5,3.302325581395349)
```

Exemplo

```
DESCRIBE ratings;
DESCRIBE ratingsByMovie;
DESCRIBE avgRatings;
```

```
ratings: {userID: int,movieID: int,rating: int,ratingTime: int}
ratingsByMovie: {group: int,ratings: {(userID: int,movieID: int,rating: int,ratingTime: int)}}
avgRatings: {movieID: int,avgRating: double}
```

Exemplo

```
DESCRIBE ratings;
DESCRIBE ratingsByMovie;
DESCRIBE avgRatings;
```

```
ratings: {userID: int,movieID: int,rating: int,ratingTime: int}
ratingsByMovie: {group: int,ratings: {(userID: int,movieID: int,rating: int,ratingTime: int)}}
avgRatings: {movieID: int,avgRating: double}
```

Exemplo

FILTER

```
fiveStarMovies = FILTER avgRatings BY avgRating > 4.0;
```

```
(12,4.385767790262173)
(22,4.151515151515151)
(23,4.120879120879204)
(45,4.05)
```

Exemplo

JOIN

```
DESCRIBE fiveStarMovies;
DESCRIBE nameLookup;
fiveStarsWithData = JOIN fiveStarMovies BY movieID, nameLookup BY movieID;
DESCRIBE fiveStarsWithData;
DUMP fiveStarsWithData;
```

```
fiveStarMovies: {movieID: int,avgRating: double}
nameLookup: {movieID: int,movieTitle: chararray,releaseTime: long}
fiveStarsWithData: {fiveStarMovies::movieID: int,fiveStarMovies::avgRating: doub
nameLookup::movieID: int,nameLookup::movieTitle:
chararray,nameLookup::releaseTime: long}
```

```
(12,4.385767790262173,12,Usual Suspects, The (1995),808358400)
(22,4.151515151515151,22,Braveheart (1995),824428800)
(23,4.120879120879120 4,23,Taxi Driver (1976),824428800)
```

Exemplo

ORDER BY

```
oldestFiveStarMovies = ORDER fiveStarsWithData BY
nameLookup::releaseTime;

DUMP oldestFiveStarMovies;
```

```
(493,4.15,493,Thin Man, The (1934),-1136073600)
(604,4.0123456790123456,604,It Happened One Night (1934),-1136073600)
(615,4.050847457627119,615,39 Steps, The (1935),-1104537600)
(1203,4.047619047619048,1203,Top Hat (1935),-1104537600)
```

Pig

Hey, ho, let's go!

Exemplo

ORDER BY

```
oldestFiveStarMovies = ORDER fiveStarsWithData BY
nameLookup::releaseTime;

DUMP oldestFiveStarMovies;
```

```
(493,4.15,493,Thin Man, The (1934),-1136073600)
(604,4.0123456790123456,604,It Happened One Night (1934),-1136073600)
(615,4.050847457627119,615,39 Steps, The (1935),-1104537600)
(1203,4.047619047619048,1203,Top Hat (1935),-1104537600)
```

Obrigado!!!

Nos vemos amanhã!!!

Bom descanso!