UNIVESIDADE POSITIVO

PÓS-GRADUAÇÃO EM BIGDATA

Contagem das palavras em Python e ShellScript em base de dados lit2go

Alunos:

Anderson Bertling

Marcos Estevão Hadiak

Disciplina:

Hadoop e MapReduce

Professor:

Alessandro Binhara

Curitiba, 20/04/2016

Proposta de análise

Uma análise de base de dados literário, do site <https://etc.usf.edu/lit2go/>, com exploração de conceitos de sistema linux para baixar bases novas e refinar dado melhorando assim dados obtidos em quaisquer análises análise.

Orientação para execução do script

Devido a fatores de obtenção de dados para a elaboração da análise, constatamos a necessidade de refinar a base, optamos por utilizar aplicações básicas do linux como os comandos sed, wget e outros que veremos detalhadamente em breve.

Os comandos usados para coletar informações novas e refinar esse dados são:

Comando- lynx converte arquivos html em um formato mais fácil de se trabalhar, exemplo lynx --dump index.html > index.txt

Scripts

Requisitos

yum -y install dialog

yum -y install lynx # caso conversão seja feito

yum install poppler-utils # conversão de pdf para txt usado no exemplo

Obtenção e refino de base de dados

$ vim obtencao.sh

-----------------------------------------------------------------------

#/bin/bash

LIVRO=$(dialog --stdout --inputbox "Digite o nome do livro em caixa baixa" 0 0 )

WEB1=$(dialog --stdout --inputbox "Cole o endereço web do livro em html" 0 0 )

mkdir -p $HOME/$LIVRO

mkdir -p $HOME/$LIVRO/txt

mkdir -p $HOME/$LIVRO/htmls

mkdir -p $HOME/$LIVRO/pdf

cd $HOME/$LIVRO/

wget $WEB1

fgrep $LIVRO index.html | cut -d \" -f 2 > indice.txt

sed -i '/faux/d' indice.txt

sed -i '/subject/d' indice.txt

wget -i indice.txt -P $HOME/$LIVRO/htmls

grep .pdf $HOME/$LIVRO/htmls/\* > PDF.txt

fgrep .pdf PDF.txt | cut -d \" -f 2 | sort > pdfok.txt

wget -i pdfok.txt -P $HOME/$LIVRO/pdf

ls $HOME/$LIVRO/pdf | sort > $HOME/$LIVRO/indice\_ordenado.txt

sed -i 's/^/pdftotext /' $HOME/$LIVRO/indice\_ordenado.txt

#sed -i 's/$/' >> livro.txt'/' $HOME/$LIVRO/indice\_ordenado.txt

sed -e '1i\' -e '#!/bin/bash' $HOME/$LIVRO/indice\_ordenado.txt >> $HOME/$LIVRO/pdf/agrupar.sh

chmod 777 $HOME/$LIVRO/pdf/agrupar.sh

cd $HOME/$LIVRO/pdf/

./agrupar.sh

ls | grep txt| sort > agrupado.sh

sed -i 's/^/cat /' agrupado.sh

sed -i 's/$/ >> livro.txt/' agrupado.sh

sed -e '1i\' -e '#!/bin/bash' agrupado.sh >> agrupaok.sh

chmod 777 agrupaok.sh

./agrupaok.sh

sed -i 'y/ABCDEFGHIJKLMNOPQRSTUVWXYZÀÁÂÃÉÊÍÓÔÕÚÇ/abcdefghijklmnopqrstuvwxyzàáâãéêíóôõúç/' livro.txt

sed -i 's/\[0-9]//g' livro.txt

sed -i 's/[[:punct:]]//g' livro.txt

sed -i '/litgo/d' livro.txt

cd ..

cd ..

#wordcount

echo "#########################################" > baseresultado.txt

echo "wordcout-result" >> baseresultado.txt

wc -w $HOME/$LIVRO/pdf/livro.txt >> baseresultado.txt

--------------------------------------------------------------------------

$vim mapper.py

#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

# remove leading and trailing whitespace

line = line.strip()

# split the line into words

words = line.split()

# increase counters

for word in words:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

#

# tab-delimited; the trivial word count is 1

print '%s\t%s' % (word, 1)

-------------------------------

$vim reducer.py

#!/usr/bin/env python

from operator import itemgetter

import sys

current\_word = None

current\_count = 0

word = None

# input comes from STDIN

for line in sys.stdin:

# remove leading and trailing whitespace

line = line.strip()

# parse the input we got from mapper.py

word, count = line.split('\t', 1)

# convert count (currently a string) to int

try:

count = int(count)

except ValueError:

# count was not a number, so silently

# ignore/discard this line

continue

# this IF-switch only works because Hadoop sorts map output

# by key (here: word) before it is passed to the reducer

if current\_word == word:

current\_count += count

else:

if current\_word:

# write result to STDOUT

print '%s\t%s' % (current\_count, current\_word)

current\_count = count

current\_word = word

# do not forget to output the last word if needed!

if current\_word == word:

print '%s\t%s' % (current\_count, current\_word)

-----------------------------------------------------------------------

# comando para copiar o arquivo livro.txt para o hdfs

hadoop fs -put livro.txt /user/root/MapReduce/

# comando que executa o Map e o Reduce

hadoop fs -cat /user/root/MapReduce/livro.txt | python mapper.py | sort | python reducer.py | sort -g > resultado.txt

Execução via Streaming

Comando:

yarn jar /usr/lib/hadoop-mapreduce/hadoop-streaming-\*.jar \

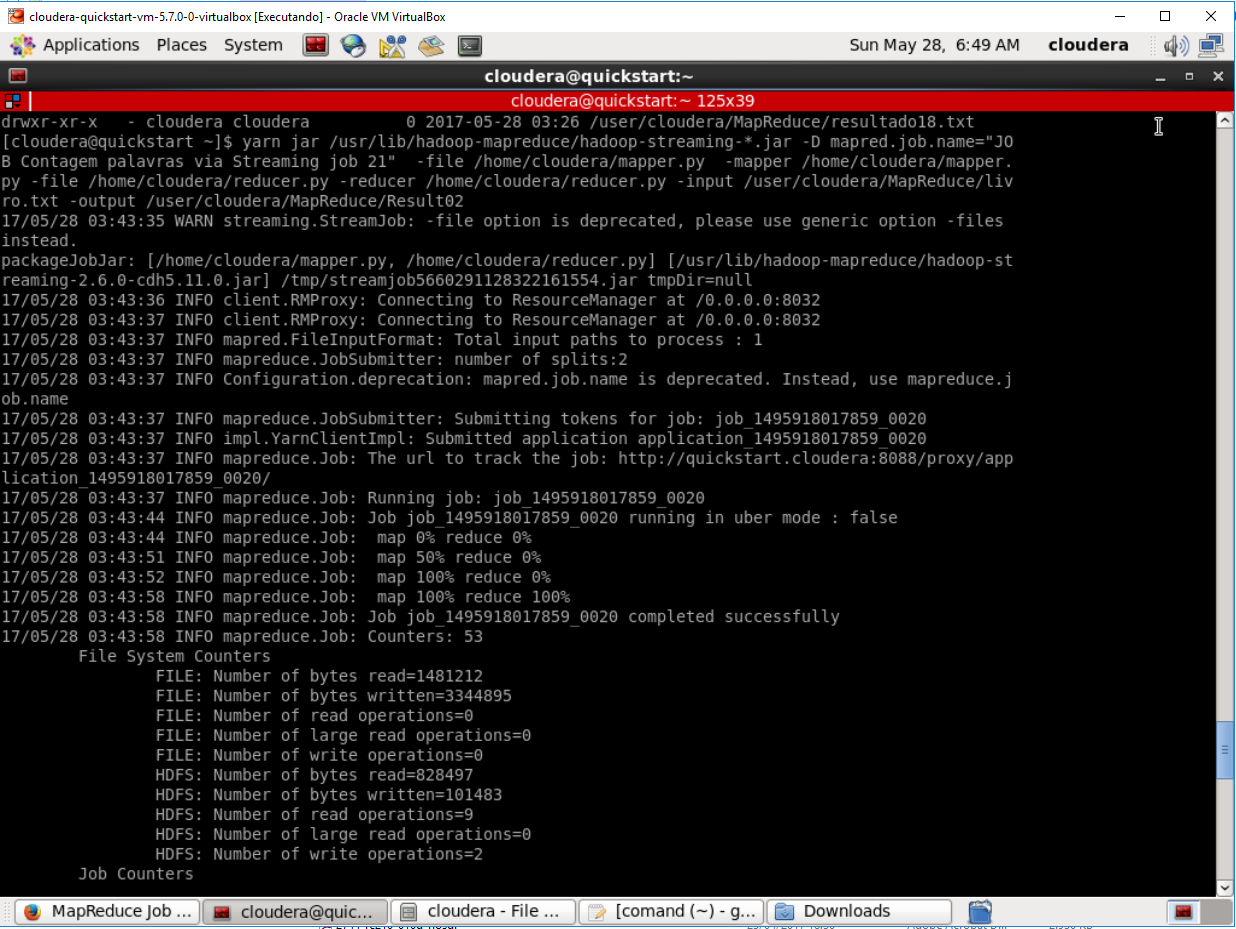
-D mapred.job.name="JOB Contagem palavras via Streaming job 21"  \

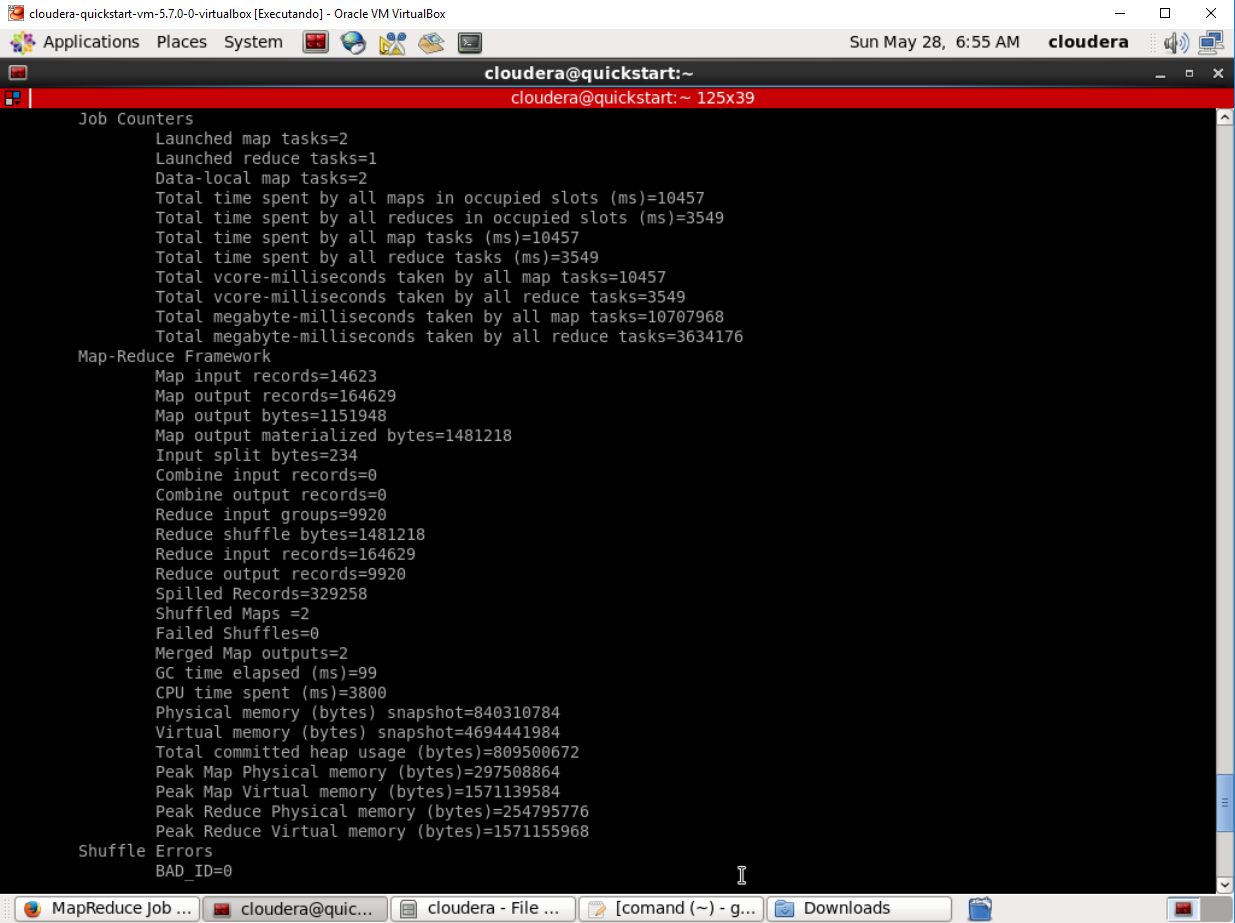
-file /home/cloudera/mapper.py  -mapper /home/cloudera/mapper.py \

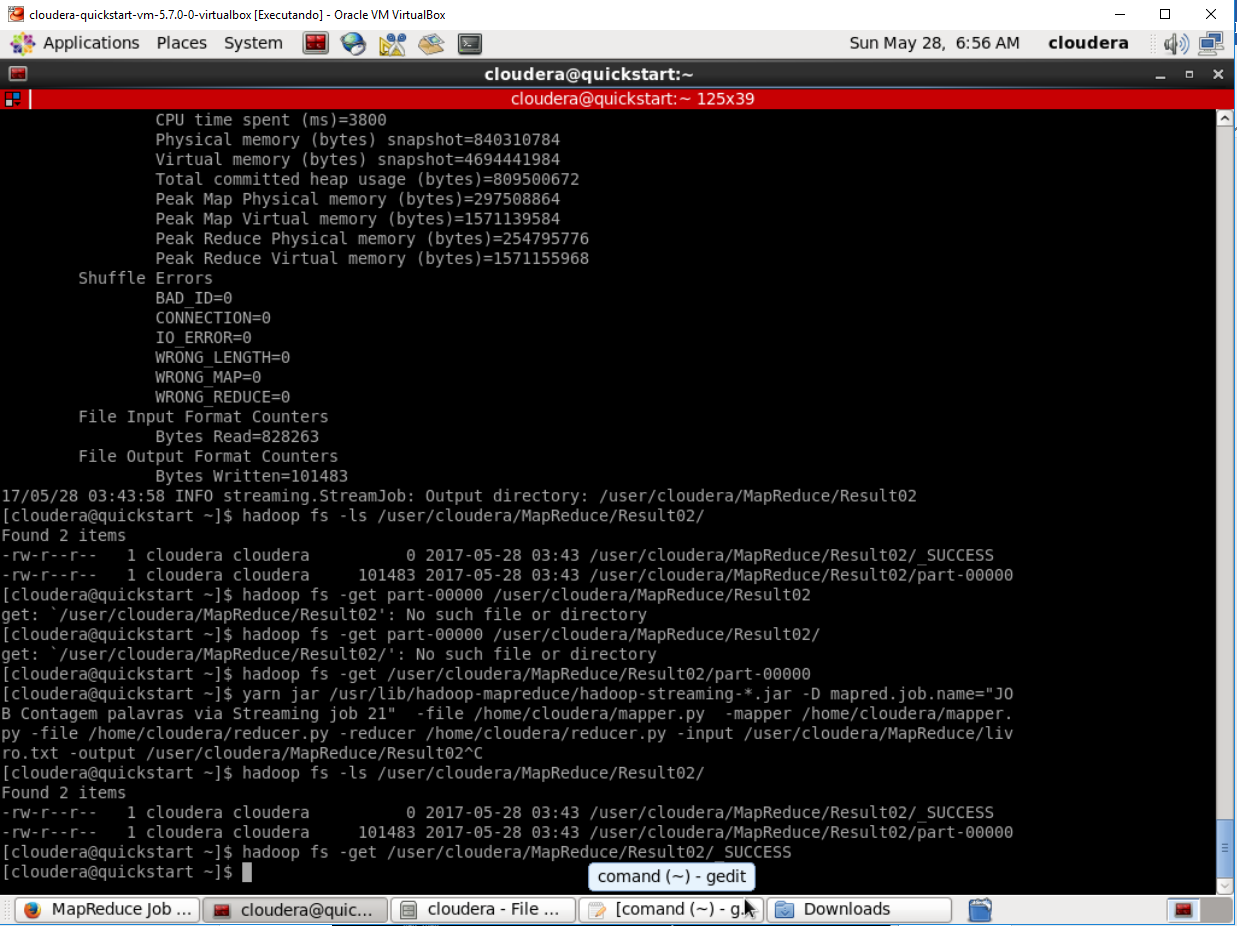
-file /home/cloudera/reducer.py -reducer /home/cloudera/reducer.py \

-input /user/cloudera/MapReduce/livro.txt -output /user/cloudera/MapReduce/Result02

Telas de execução do Streaming







## Base de dados

Site: http://etc.usf.edu/lit2go/