

Семинар №2

SQL и BigData

1. Инструментарий:

[Лекция](#)

[Презентация](#)

Задание:

1) Для работы нужно установить дистрибутив [Docker](#):

- <https://docs.docker.com/get-docker/>
- <https://docs.docker.com/desktop/install/windows-install/>
- <https://docs.docker.com/desktop/install/mac-install/>
- <https://docs.docker.com/desktop/install/linux-install/>

2) Выполняем команду docker compose up:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
37e0d92e8655	bde2020/hive:2.3.2-postgresql-metastore	"/entrypoint.sh /opt/..."	About an hour ago	Up About an hour	10000/tcp, 0.0.0.0:9083->9083/tcp, 10002/tcp	hadoop-mapreduce-hive-metastore-1
9f67686c6erb	bde2020/hadoop-namenode:2.0.0-hadoop2.7.4-java8	"/entrypoint.sh /run..."	About an hour ago	Up About an hour (healthy)	0.0.0.0:50070->50070/tcp	hadoop-mapreduce-namenode-1
83c790773db	bde2020/hive:2.3.2-postgresql-metastore	"/entrypoint.sh /bin/..."	About an hour ago	Up About an hour	0.0.0.0:10000->10000/tcp, 10002/tcp	hadoop-mapreduce-hive-server-1
669b60fde70	bde2020/hive-metastore-postgresql:2.3.0	"/docker-entrypoint..."	About an hour ago	Up About an hour	5432/tcp	hadoop-mapreduce-hive-metastore-postgresql-1
2abc2d9a3e0b	shanzhu/prestodb:0.181	"/bin/launcher run"	About an hour ago	Up About an hour	0.0.0.0:8080->8080/tcp	hadoop-mapreduce-presto-coordinator-1
77ec377531c	bde2020/hadoop-datamode:2.0.0-hadoop2.7.4-java8	"/entrypoint.sh /run..."	About an hour ago	Up About an hour (healthy)	0.0.0.0:50075->50075/tcp	hadoop-mapreduce-datamode-1

3) Наблюдаем и изучаем настройки и состояние NM и RM в веб-интерфейсе

Overview 'namenode:9000' (active)


Started:	Tue Oct 25 01:24:02 +0300 2022
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 18:56:00 +0300 2019 by rohitsharmaks from branch-3.2.1
Cluster ID:	CID-45d80440-b8ca-4048-8ec7-434c91ea5f28
Block Pool ID:	BP-789518686-172.18.0.2-1666650078496

Summary

Security is off.
Safemode is off.
1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).
Heap Memory used 254.18 MB of 762 MB Heap Memory. Max Heap Memory is 6.95 GB.
Non Heap Memory used 48.88 MB of 50.06 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	601.79 GB
Configured Remote Capacity:	0 B
DFS Used:	84 KB (0%)
Non DFS Used:	389.91 GB
DFS Remaining:	181.1 GB (30.09%)
Block Pool Used:	84 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion (including replicas)	0
Block Deletion Start Time	Tue Oct 25 01:24:02 +0300 2022
Last Checkpoint Time	Tue Oct 25 01:21:18 +0300 2022
Enabled Erasure Coding Policies	RS-6-3-1024k

← → 🔍 localhost:9870/cluster/scheduler

NEW,NEW_SAVING,SUBMITTED,ACCEPTED,RUNNING Applications

Cluster

- About Nodes
- Node Labels
- Applications
 - NEW
 - NEW_SAVING
 - SUBMITTED
 - ACCEPTED
 - RUNNING
 - FINISHED
 - FAILED
 - Scheduler

Cluster Metrics

Apps Submitted	0	Apps Pending	0	Apps Running	0	Apps Completed	0	Containers Running		Used Resources	<memory:0 B, vCores:0>	Total Resources	<memory:16 GB, vCores:1>	Reserved Resources	<memory:0 B, vCores:0>	Physical Mem Used %	56	
----------------	---	--------------	---	--------------	---	----------------	---	--------------------	--	----------------	------------------------	-----------------	--------------------------	--------------------	------------------------	---------------------	----	--

Cluster Nodes Metrics

Active Nodes	0	Decommissioning Nodes		Decommissioned Nodes		Lost Nodes	0	Unhealthy Nodes		Rebooted Nodes	0
--------------	---	-----------------------	--	----------------------	--	------------	---	-----------------	--	----------------	---

Scheduler Metrics

Scheduler Type	Capacity Scheduler	Scheduling Resource Type	[memory-mb (unit=M), vcores]	Minimum Allocation	<memory:1024, vCores:1>	Maximum Allocation	<memory:8192, vCores:4>	Maximum Cluster Application Priority	0
----------------	--------------------	--------------------------	------------------------------	--------------------	-------------------------	--------------------	-------------------------	--------------------------------------	---

Dump scheduler logs 1 min

Application Queues

Legend: Capacity Used Used (over capacity) Max Capacity Users Requesting Resources Auto Created Queues

[- Queue: root 0.0% used
[- Queue: default 0.0% used

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Allocated GPUs	Reserved CPU Vcores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress
No data available in table																				

Showing 0 to 0 of 0 entries

Aggregate scheduler counts

Total Container Allocations(count)	0	Total Container Releases(count)	0	Total Fulfilled Reservations(count)	0	Total Container Preempts	0
------------------------------------	---	---------------------------------	---	-------------------------------------	---	--------------------------	---

HDFS

Intro

Основные команды hdfs: <https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide>

Команды для работы с файлами:

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html>

Задание:

- 1) Загрузите датасет, доступный на kaggle.com:
<https://www.kaggle.com/pieca111/music-artists-popularity>
- 2) Положите его в hdfs
- 3) Выяснить сколько места занимают файл (hdfs dfs -du -h)
- 4) Изменить фактор репликации (hdfs dfs -setrep <path>)
- 5) Удалить файл (hdfs dfs -rm <file>). При удалении файл перемещается в корзину (~/.Trash). При этом в корзине воссоздаётся абсолютный путь к файлу (.../.Trash/user/your_user/your_dir/your_file). Для удаления в обход корзины можно использовать флаг --skipTrash

Пример решения:

```
sudo docker exec -it namenode bash
hdfs dfs -mkdir /my_dir
hdfs dfs -touchz /my_dir/another_dir/emptyfile
hdfs dfs -rm -r /my_dir
dfs dfs -copyFromLocal example.txt /
hdfs dfs -head /error.txt
hdfs dfs -tail /error.txt
hdfs dfs -cat /error.txt
```

Hive

Задание:

- 1) Сделать таблицу artists в Hive и вставить туда значения, используя датасет <https://www.kaggle.com/pieca111/music-artists-popularity>
- 2) Используя Hive найти
 - a) Исполнителя с максимальным числом скробблов
 - b) Самый популярный тэг на ластфм
 - c) Самые популярные исполнители 10 самых популярных тегов ластфм
 - d)

Пример решения:

```
docker cp artists.csv hadoop-mapreduce-hive-server-1:/
docker exec -it hadoop-mapreduce-hive-server-1 bash
/opt/hive/bin/beeline -u jdbc:hive2://localhost:10000
CREATE TABLE `artists`(`mbid` string, `artist_mb` string, `artist_lastfm`
string, `country_mb` string, `country_lastfm` string, `tags_mb` string,
`tags_lastfm` string, `listeners_lastfm` int, `scrobbles_lastfm` int,
`ambiguous_artist` boolean) row format delimited fields terminated by ',';
LOAD DATA LOCAL INPATH '/artists.csv' OVERWRITE INTO TABLE artists;

# Исполнителя с максимальным числом скробблов
select artist_mb, max(scrobbles_lastfm) max_scrobbles from artists group by
artist_mb order by max_scrobbles desc limit 10;
# +-----+-----+
# |      artist_mb      | max_scrobbles |
```

```

# +-----+-----+
# | The Beatles          | 517126254 |
# | Radiohead            | 499548797 |
# | Coldplay             | 360111850 |
# | Muse                 | 344838631 |
# | Arctic Monkeys       | 332306552 |
# | Pink Floyd           | 313236119 |
# | Linkin Park          | 294986508 |
# | Red Hot Chili Peppers | 293784041 |
# | Lady Gaga            | 285469647 |
# | Metallica            | 281172228 |
# +-----+-----+
# Самый популярный тэг на ластфм
select trim(itemsName) tag_lastfm, count(*) cnt from artists LATERAL VIEW
explode(split(tags_lastfm, ';')) itemTable AS itemsName group by
trim(itemsName) order by cnt desc limit 10;
# +-----+-----+
# | tag_lastfm           | cnt      |
# +-----+-----+
# |                      | 1084072  |
# | seen live           | 99540    |
# | rock                | 73406    |
# | electronic          | 70676    |
# | under 2000 listeners | 50827    |
# | All                 | 50166    |
# | pop                 | 48447    |
# | indie               | 47452    |
# | alternative         | 43732    |
# | experimental        | 40845    |
# +-----+-----+
# c)
with t1 as (select trim(itemsName) as tag_lastfm, sum(listeners_lastfm) as
listeners from artists LATERAL VIEW explode(split(tags_lastfm, ';'))
itemTable AS itemsName group by trim(itemsName) order by listeners desc
limit 10),
t2 as (select trim(itemsName) as tag_lastfm, listeners_lastfm as listeners,
artist_lastfm from artists LATERAL VIEW explode(split(tags_lastfm, ';'))
itemTable AS itemsName),
t3 as (select tag_lastfm, artist_lastfm, listeners, row_number()
over(partition by tag_lastfm order by listeners desc) as rn from t2 where
tag_lastfm in (select tag_lastfm from t1))
select * from t3 where rn <= 10;
d)
select artist_mb, count(*) from artists group by artist_mb limit 10;

```

MapReduce

Задание:

- 1) Загрузите датасет по ценам на жилье Airbnb, доступный на kaggle.com: <https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data>
- 2) Используя Python, реализуйте скрипт mapper.py и reducer.py для расчета. В итоге у вас должно получиться 2 скрипта: mapper и reducer для каждой величины.

Пример решения:

```
def mapper():
    for line in csv.reader(sys.stdin):
        if len(line[9]) > 0 and line[9] != 'price':
            sys.stdout.write(f'{line[9]}\t1\n')

def reduce():
    s = 0
    counter = 0
    for line in sys.stdin:
        vals = line.strip().split("\t")
        s += float(vals[0])
        counter += 1

    sys.stdout.write("{}\n".format(s / counter))

root@fb82701789af:/# hdfs dfs -cat /block3/output/part-00000
152.7206871868289
```

2. Домашнее задание

Условие:

- 1) Загрузите датасет по ценам на жилье Airbnb, доступный на kaggle.com: <https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data>
- 2) Подсчитайте среднее значение и дисперсию по признаку "price" в hive
- 3) Используя Python, реализуйте скрипт mapper.py и reducer.py для расчета
- 4) Проверьте правильность подсчета статистики методом mapreduce в сравнении со hive.