Podsumowanie warsztatu.

1. Środowisko:
https://github.com/jpodeszwik/virtual-hadoop-cluster

Instrukcje instalacji znajdują się w readme repozytorium, oraz na stronie:
http://dandydev.net/blog/installing-virtual-hadoop-cluster

2. Generator spreparowanych logów apache'a:
https://github.com/jpodeszwik/Fake-Apache-Log-Generator
instrukcje instalacji są w pliku README.md

Polecenia:
run.sh – generuje zwykłe apache'owe logi.
run2.sh – generuje logi sprzed 2 dni symulując opóźnienie w usługach wysyłających logi.
run3.sh – generuje naprzemiennie logi poprawne z logami uszkodzonymi – linijki nie są pełne
run4.sh – generuje zduplikowane linijki loga

Przykład
./run.sh vm-cluster-node2 9999

3. Repozytorium z przykładami i zadaniami:
https://github.com/jpodeszwik/hadoop-workshop-day1

1. Komendy hdfs
Zadanie:
utwórz na hdfsie katalogi, których właścicielem będzie użytkownik vagrant:
/user/vagrant
/user/vagrant/inputs
/user/vagrant/utputs

utwórz katalog, którego właścicielem będzie użytkownik flume:
/user/flume

utwórz katalog, którego właścicielem będzie użytkownik kursant:
/user/kursant

Wrzuć pliki loremipsum i apache_logs do katalogu /user/vagrant/inputs

skorzystaj z poleceń:
hdfs dfs -mkdir <ścieżka_na_hdfs>
hdfs dfs -put <ścieżka_do_lokalnego_pliku> <ścieżka_na_hdfs>
hdfs dfs -chown <user>:<grupa> <ścieżka_na_hdfs>

Jeżeli klaster nie jest zabezpieczony kerberosem, to możesz użyć zmiennej HADOOP_USER_NAME do ustawienia użytkownika z którego będzie wykonane polecenie. Na hdfsie odpowiednikiem linuxowego roota jest użytkownik ‘hdfs’. Przykład:
HADOOP_USER_NAME=hdfs hdfs dfs -mkdir <ścieżka_na_hdfs>

Zmień prawa do odczytu do katalogu /user/kursant tak, żeby tylko kursant mógł go odczytać. Skorzystaj z komendy:
hdfs dfs -chmod <uprawnienia> <ścieżka_na_hdfs>
Spróbuj odczytać katalog jako user ’vagrant’
Spróbuj odczytać katalog jako user ’hdfs’

Dokumentacja komend:
https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html

2. Flume - agent służący do dostarczania logów na żywo na hdfs:

Zadania:
Napisz interceptor, który odkoduje datę z linijki loga i ustawi odpowiednią datę w nagłówku. Skorzystaj z nagłówka ‘timestamp’ i umieść w nim unixowy timestamp wyznaczony na podstawie daty odkodowanej z eventu.

Dokumentacja flume:
https://flume.apache.org/FlumeUserGuide.html

Zadania, których nie zdążyliśmy zrobić:
Skorzystaj z możliwości multiplexacji, żeby odfiltrować błędne linijki loga. Przekieruj błędne linijki do innego katalogu.
https://flume.apache.org/FlumeUserGuide.html#multiplexing-the-flow

3. Mapreduce – framework do rozproszonej analizy danych.
Zadania:
Policz litery w pliku loremipsum.
Oblicz sumę content_length pogrupowaną po metodzie z części logów apache'a, które dostarczyliśmy za pomocą flume'a.

Zadania, których nie zdążyliśmy zrobić:
Posortuj słowa po ilości wystąpień – trzeba uruchomić 2 joby mapreduce. Jeden liczący, a drugi sortujący przeliczone słowa.

Napisz joba mapreduce, który usunie duplikaty z logów – należy wykorzystać całą linijkę jako klucz.

Napisz joba mapreduce, który policzy ile było unikalnych słów w pliku loremipsum.

Napisz joba mapreduce, który połączy linijki loga z pliku apache_logs z imionami z tabeli ip_name (wcześniej należy wykonać import bazy sqoopem). Polem po którym należy logi połączyć jest ip. Skorzystaj z klasy MultipleInputs:
https://hadoop.apache.org/docs/r2.7.3/api/org/apache/hadoop/mapreduce/lib/input/MultipleInputs.html

4. Hadoop streaming – możliwość uruchamiania jobów mapreduce'owych w dowolnej technologii.
Zadania:
Policz litery w pliku loremipsum.
Oblicz sumę content_length pogrupowaną po metodzie z części logów apache'a, które dostarczyliśmy za pomocą flume'a.

Zadania, których nie zdążyliśmy zrobić:
Napisz joba hadoop streaming, który połączy linijki loga z pliku apache_logs z imionami z tabeli ip_name (wcześniej należy wykonać import bazy sqoopem). Polem po którym należy logi połączyć jest ip.

4. Sqoop – importer danych z relacyjnych baz za pomocą jobów mapreduce:
Przygotowanie:

- zainstaluj na vm-cluster-node1 serwer mysql 'sudo apt-get install mysql-server'. Zostaw puste hasło roota.
- wgraj dumpa bazy z folderu hadoop-workshop-day1/files/dump.sql - na vm-cluster-node1 wykonaj polecenie 'mysql -u root < dump.sql'
- ściągnij connector dla mysqla http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.39.tar.gz I rozpakuj (tar -xzf mysql-connector-java-5.1.39.tar.gz)
- W rozpakowanym archiwum znajdź mysql-connector-java-5.1.39-bin.jar i wrzuć na vm-cluster-node2 do katalogu '/var/lib/sqoop'

Szablony poleceń:

```
sqoop import --connect jdbc:mysql://<hostname:ip>/<baza> --username
<użytkownik> --password <hasło> --table <tabela> --fields-terminated-by
<delimiter> --target-dir <ścieżka_ha_hdfs>

sqoop export --connect jdbc:mysql://<hostname:ip>/<baza> --username
<użytkownik> --password <hasło> --table <tabela> --input-fields-terminated-
by <delimiter> --export-dir <ścieżka_na_hdfs>
```

Zadania:
Zaimportuj bazę ip_name na hdfs.

Wyeksportuj bazę name_useragent do sqoopa – trzeba ją najpierw utworzyć za pomocą mapreduce’a lub hive’a.

Dokumentacja sqoopa:
https://sqoop.apache.org/docs/1.4.1-incubating/SqoopUserGuide.html

5. Hive – umożliwia tworzenie na hdfsie baz sqlowych i zadawanie do nich zapytań sqlowych tłumaczonych na serię jobów mapreduce'owych.

Zadania:
Utwórz tabelę apache_logs. Wrzuć do niej dane z pliku apache_logs. Możesz skorzystać z polecenia 'SHOW CREATE TABLE', żeby namierzyć gdzie hive utworzył bazę na hdfsie i przekopiować plik do katalogu.

Utwórz tabelę typu 'external' I załaduj do niej takie same dane. Możesz skorzystać z polecenia `insert into <tabela> select ...`.

Zdropuj obie tabele I zobacz co stało się z danymi na hdfsie.

Utwórz tabelę w formacie ORC – na końcu polecenia 'CREATE' dodaj 'STORED AS ORC'. Załaduj do niej dane z tabeli apache_logs (najprościej `insert into <tabela> select ...`). Spróbuj poleceniem 'hdfs dfs -cat ...' wypisać zawartość plików I zobacz, że są binarne.

Posumuj content_length po metodzie (PUT, GET) i zobacz, które requesty zajęły najwięcej transferu. Możesz skorzystać z metody hive'owej split(<string>, <pattern>), żeby wyciągnąć metodę z pola request.

Utwórz UDF, który wyciągnie z pola request metodę i spróbuj go użyć zamiast polecenia split.

Utwórz tabelę partycjonowaną po metodzie i wrzuć do niej dane z tabeli apache_logs. Skorzystaj z polecenia SET hive.exec.dynamic.partition=true, żeby odblokować dynamiczne wyznaczanie partycji.

Stwórz tabelę name_useragent, która będzie zawierała dane zjoinowane z tabel apache_logs i ip_name (zaimportowana za pomocą sqoopa).

Dokumentacja hive:
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML

Zadania, których nie zdążyliśmy zrobić:
Stwórz tabelę partycjonowaną po dacie i godzinie i zapisz do niej dane flumem.

Utwórz funkcję UDF, która sparsuje pole request i zwróci strukturę z 3 polami. Należy wykorzystać ObjectInspector.

Utwórz funkcję UDAF, której użyjesz do zliczania ilości wystąpień metod (POST, GET). Funkcja powinna zwrócić mapę <metoda, ilość wystąpień>.

Utwórz funkcję UDTF, która zrobi split pola po spacjach i dla każdej wartości wyemituje wiersz.
W przykładach jest funkcja UDAF robiąca concat na stringach. Można jej użyć do wygenerowania tabeli, którą będziemy splitować za pomocą UDTF `select concat(ip), method(request) GROUP BY method(request)`

6. Oozie – narzędzie do tworzenia workflowów

Najpierw trzeba przekonfigurować yarna:
1. Na cloudera managerze wejdź w konfigurację yarna.
2 .Wpisz w wyszukiwarkę: yarn.nodemanager.resource.cpu-vcores I ustaw ten parametr na 2.
3. yarn.app.mapreduce.am.resource.mb: 512MiB
4. mapreduce.map.memory.mb: 512MiB
5. mapreduce.reduce.memory.mb: 512MiB
6. yarn.scheduler.minimum-allocation-mb: 512MiB
7. yarn.scheduler.increment-allocation-mb: 512MiB
8. yarn.scheduler.maximum-allocation-mb: 512MiB
9. mapreduce.map.java.opts.max.heap: 400 MiB
10. mapreduce.reduce.java.opts.max.heap: 400 MiB
11. ApplicationMaster Java Maximum Heap Size: 400 MiB
12. Kliknij save changes
13. Przy yarnie pojawiła się ikonka z tooltipem: “Stale configuration ...”. kliknij ją
14. Kliknij przycisk: “Restart stale services”

Zadania:
Utwórz workflow, który pobiera sqoopem tabelę ip_name, ładuję ją do hive’a, robi joina załadowanych danych z tabelą apache_logs, a potem eksport do mysqla do tabeli name_useragent.

Trzeba dodać mysqlowy connector do bibliotek w zakładce zaawansowane.

Utwórz koordynatora, który będzie uruchamiał poprzednio stworzony workflow co 5 minut.

Zadania, których nie zdążyliśmy zrobić:
Utwórz workflow rozgałęziony za pomocą 3 jobów streaming (można w ten sposób joina zaimplementować). Mappera i reducera należy umieścić na hdfsie i załączyć je za pomocą opcji files. Mapper i reducer powinny być podane tylko jako nazwa pliku, a nie cała ścieżka.