Dokumentacja projektu z przedmiotu Składowanie I przetwarzanie danych w Systemach Big Data

Przemysław Olender

Dominik Pawlak

Piotr Piątyszek

Link do repozytorium:

https://github.com/przemekolender/BigDataProject

1. Cel projektu

Celem projektu jest obserwacja informacji o pogodzie w różnych częściach Polski w czasie rzeczywistym oraz porównywanie obecnych warunków atmosferycznych z danymi z lat poprzednich. Obserwacja różnic dostarcza ciekawych informacji na temat obecnej pogody i pozwala zauważyć odchylenia od normy, a w dłuższym czasie może pokazywać zmiany klimatyczne zachodzące w danym regionie. Możliwe jest także przewidywanie postępujących zmian klimatycznych oraz stanu pogody. Potencjalni beneficjenci to firmy turystyczne, deweloperzy budujący mieszkania I ośrodki wczasowe, urbaniści planujący rozwój miast, rolnicy, ogrodnicy.

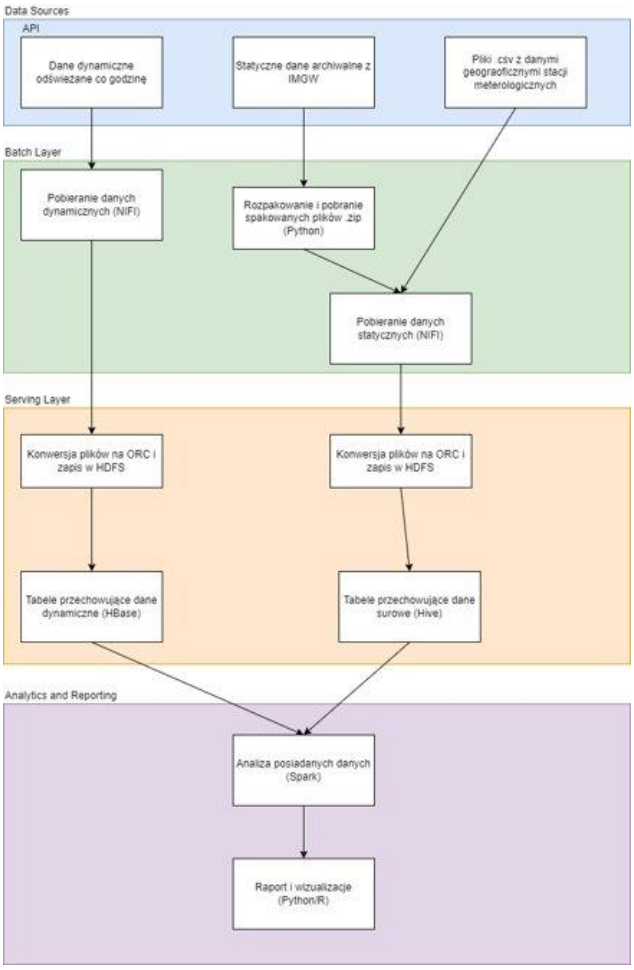
2. Źródła danych

Przetwarzamy dane z trzech różnych źródeł:

- Dane archiwalne – historyczne dane o pogodzie w Polsce w latach 1951 – 2022 z różnych stacji meteorologicznych w całej Polsce. Pochodzą ze strony Instytutu Meteorologii i Gospodarki Wodnej, są podzielone na dane z poszczególnych miesięcy (lub lat, jeśli dane są starsze) i przechowywane w osobnych plikach csv.
- Słownik stacji – plik csv z danymi na temat położenia stacji meteorologicznych.
- Dane dynamiczne - pochodzą z API Instytutu Meteorologii i Gospodarki Wodnej. API jest bardzo proste w użyciu, pod adresem https://danepubliczne.imgw.pl/api/data/synop znajdują się dane w formacie JSON, do ich pobrania nie jest potrzebne logowanie ani klucz. Dane są odświeżane co godzinę, dostępne są jedynie najnowsze dane. Udostępniane informacje to: id stacji, nazwa stacji, data pomiaru, godzina pomiaru (pełna

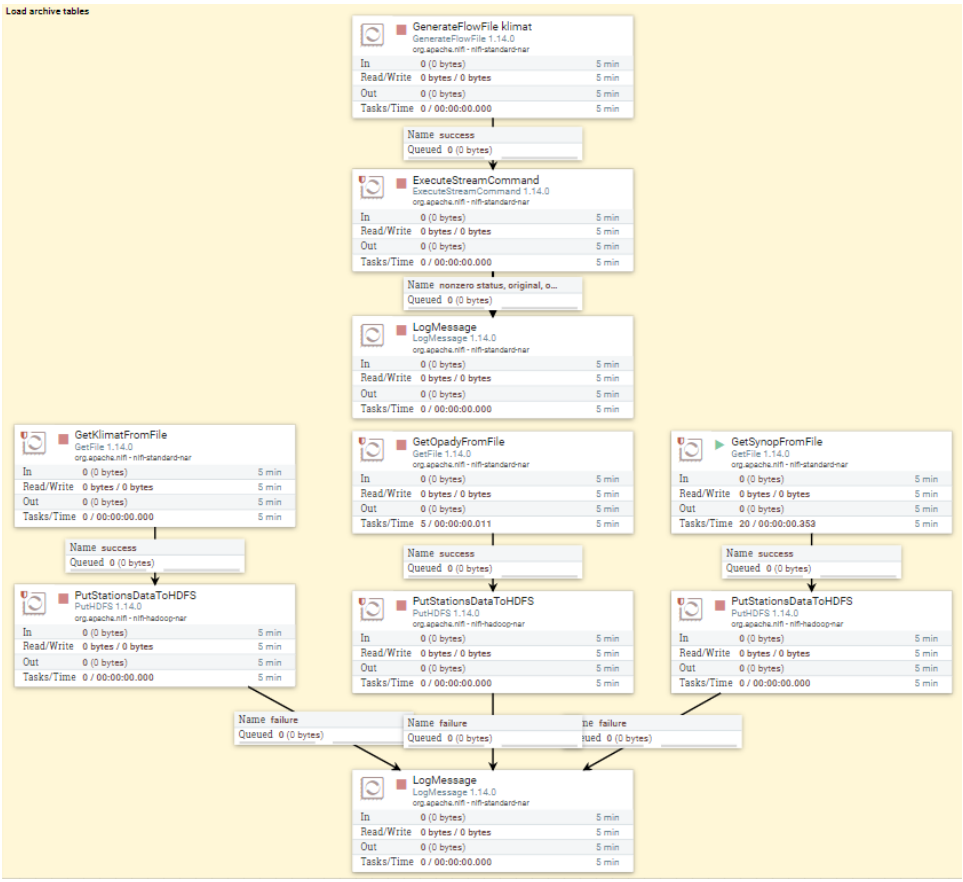
godzina), temperatura, prędkość wiatru, kierunek wiatru, wilgotność względna, suma opadu oraz ciśnienie.

3. Diagram architektury



Dane dynamiczne i archiwalne są automatycznie pobierane ze stron internetowych za pomocą Nifi, plik ze słownikiem stacji również jest automatycznie przenoszony na hdfs, następnie dane są składowane w tabelach i plikach – dane dynamiczne w tabeli Hbase i plikach parquet, a pozostałe dane w tabelach Hive i plikach ORC. Agregacja i analiza danych jest przeprowadzona za pomocą PySparka, w jej wyniku utworzone zostają niewielkie pliki csv z danymi wykorzystywanymi bezpośrednio do wizualizacji za pomocą R.

4. Ładowanie danych archiwalnych

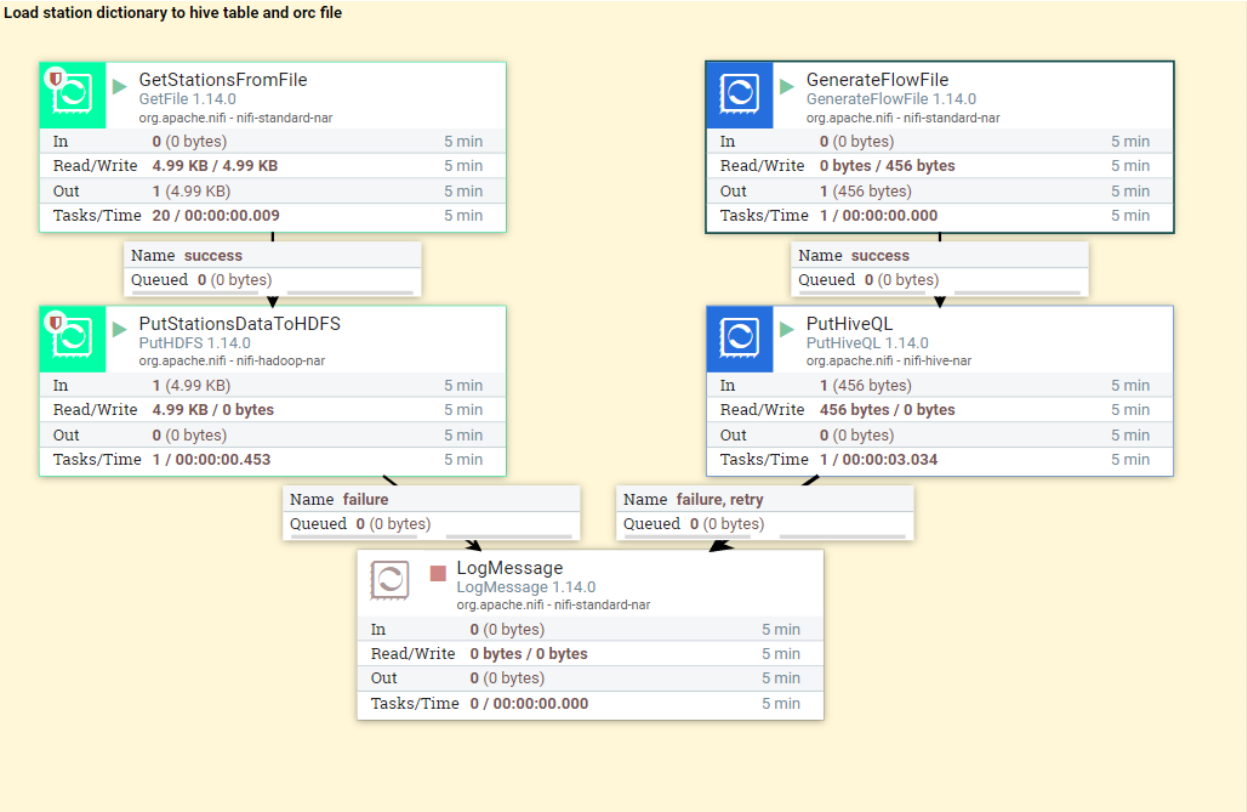


Dane archiwalne znajdują się w wielu katalogach na stronie IMGW dzielą się na 3 główne kategorie – dane dotyczące kliamtu, dotyczące opadów i synoptyczne. W celu zautomatyzowania ich pobierania napisaliśmy skrypt w Pythonie, który przechodzi przez wszystkie podstrony strony źródłowej i pobiera pliki w formacie zip do odpowiednich katalogów lokalnych. Następnie rozpakowuje pobrane pliki. Skrypt jest wywoływany poprzez Nifi w procesorze “executeStreamCommand”.

Kolejnym krokiem jest przeniesienie plików z katalogu lokalnego na hdfs – dzieje się to w 3 analogicznych procesorach w zależności od rodzaju danych (klimatyczne, opadowe, synoptyczne) ponieważ nie zdecydowaliśmy się użyć wszystkich możliwych plików w procesorach typu GetFile wybierane są tylko pliki z odpowiednimi nazwami.

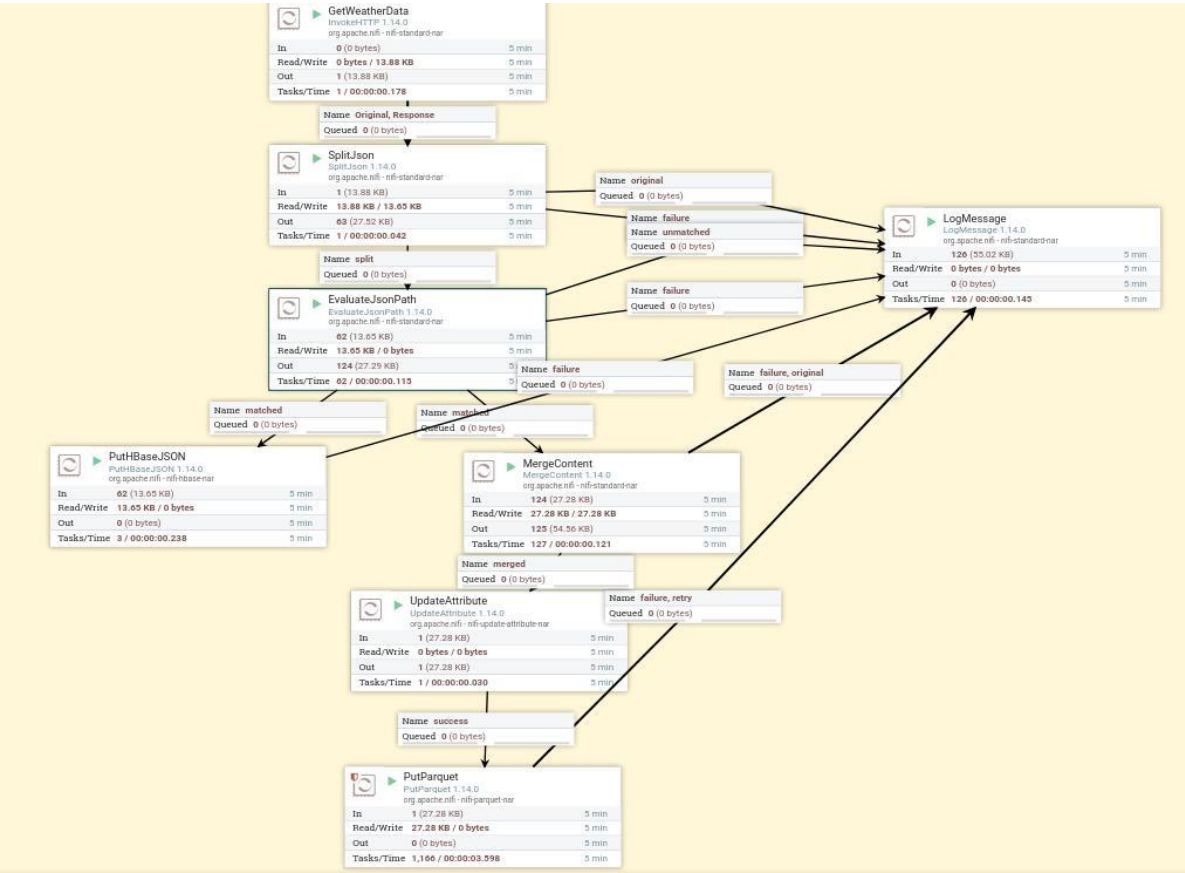
Następnie w skrypcie pythonowym zostają iteracyjnie wczytane z HDFS do pamięci, delikatnie wyczyszczone, dodane są nazwy kolumn i przy użyciu sparka załadowane do tabeli Hive. Ponieważ pliki źródłowe są podzielone względem daty, to ich iteracyjne ładowanie zapewnia optymalne podzielenie na dysku względem daty out-of-the-box.

5. Ładowanie słownika stacji



Słownik stacji to plik csv. Znajduje się w nim kod i nazwa stacji oraz jej długość i szerokość geograficzna. W pierwszej kolejności plik zostaje przekopiowany z środowiska lokalnego na hdfs. Następnie za pomocą skryptu Hive utworzona zostaje tabela, która jest wypełniana danymi z pliku. Jest ona przechowywana w formacie ORC na hdfsie.

6. Ładowanie danych dynamicznych



Najbardziej skomplikowany jest data flow do pobierania danych dynamicznych. Źródło aktualizuje się co godzinę, a więc również z taką częstotliwością dane są pobierane. Są pobierane w postaci JSONów, które następnie rozdzielna są na poszczególne rekordy. Rekordy są sprawdzane pod względem poprawności danych, jeśli tak, to następuje rozgałęzienie. Na lewej gałęzi na powyższym diagramie rekordy są ładowane do tabeli HBase z jedną rodziną kolumn a na prawej są łączone są w plik, zaktualizowana jest jego nazwa – doklejana jest do niej godzina pobrania, wreszcie plik jest zapisywany w hdfsie jako parquet.

7. Agregacja danych historycznych

W celu zrobienia wizualizacji, wpierw musieliśmy przetworzyć dane znajdujące się w Hive przy użyciu sparka. Do tego celu napisaliśmy kilka zapytań w języku HQL, które filtrują i agregują dane. Takie małe tabelki są zapisywane do formatu csv.

Wszystkie zapytania znajdują się w pliku make_stats.py. Tutaj pokażę przykładowe dla klimatu, które są dość czytelne. Zapytanie do danych synoptycznych jest bardziej skomplikowane z uwagi, że w danym pomiarze (wiersz) nie każdy parametr (kolumna) był zmierzony i informacja o tym znajduje się w jeszcze innej kolumnie.

```
month_temp_stats = sc.sql("""
    select
        Rok,
        `Miesiąc`,
        avg(`Średnia temperatura dobowa [°C]`) as avg_temp,
        avg(`Maksymalna temperatura dobowa [°C]`) as max_temp,
        avg(`Minimalna temperatura dobowa [°C]`) as min_temp
    from klimat group by Rok, `Miesiąc`
""").toPandas()
month_temp_stats.to_csv('stats/month_temp.csv')


month_rain_stats = sc.sql("""
    select
        Rok,
        `Miesiąc`,
        sum(`Suma dobowa opadów [mm]`) as sum_rain
    from klimat group by Rok, `Miesiąc`
""").toPandas()
month_rain_stats.to_csv('stats/month_rain.csv')

station_temp_stats = sc.sql("""
    select
        `Nazwa stacji`,
        avg(`Średnia temperatura dobowa [°C]`) as avg_temp
    from klimat
    where Rok > 2010
    group by `Nazwa stacji`
""").toPandas()
station_temp_stats.to_csv('stats/station_temp.csv')
```

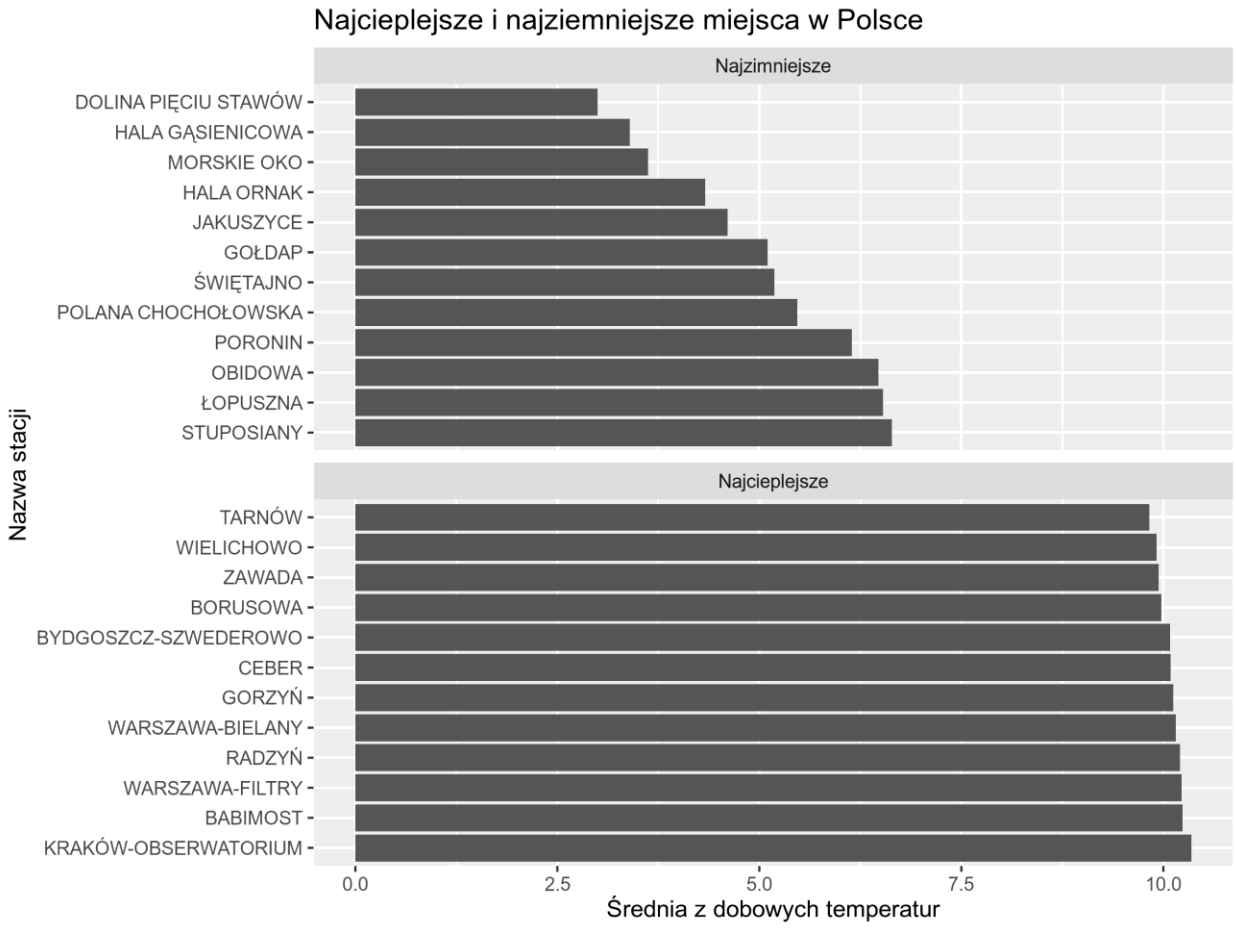
8. Generowanie wizualizacji

Wizualizacje są generowane przy użyciu skryptu w języku R. Wykorzystujemy bibliotekę ggplot do stworzenia wykresów. Tutaj zamieszczam przykładowy wykres, a wszystkie znajdują się w pliku make_vis.R.

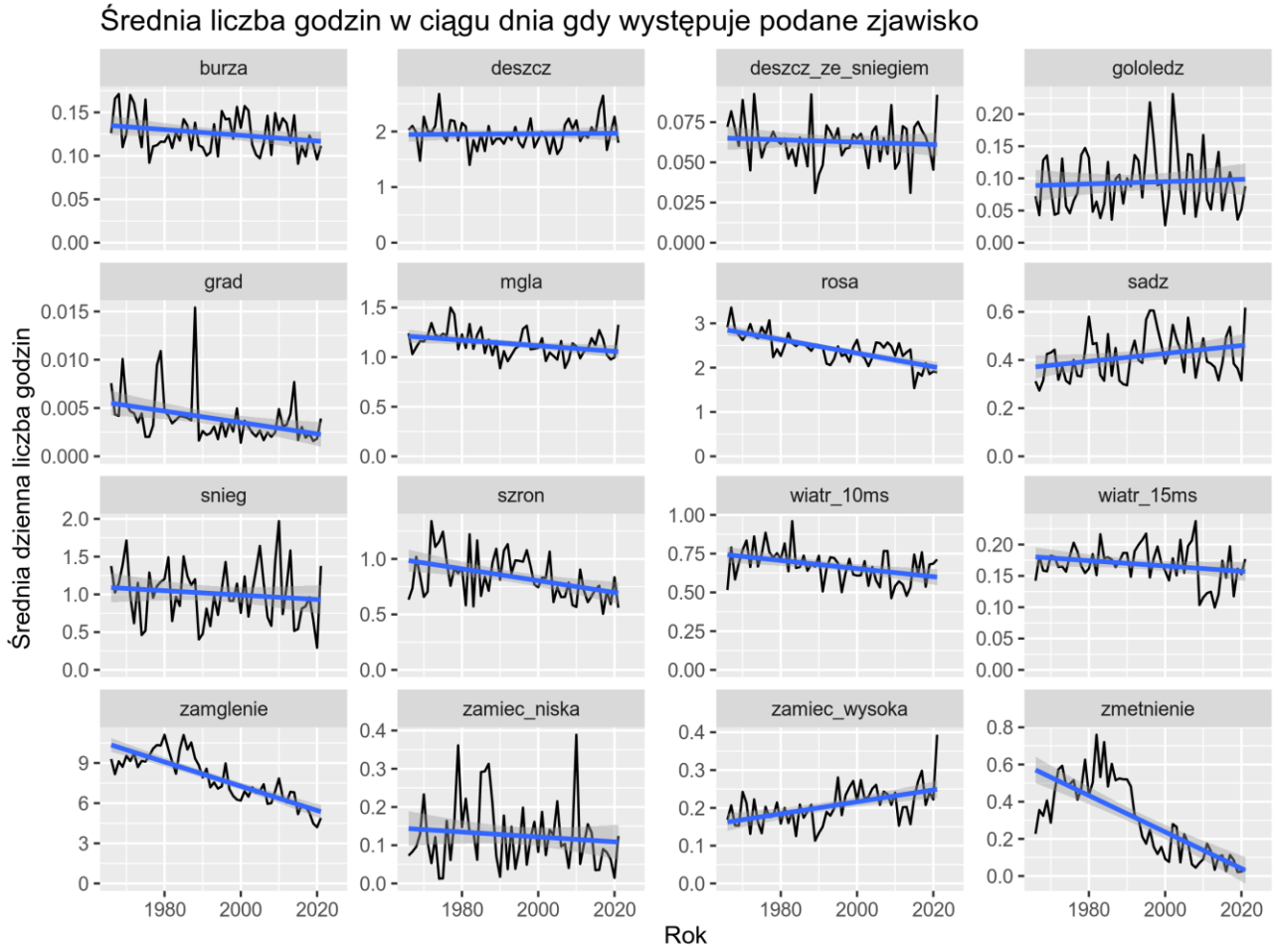
```
month_temp %>%
  pivot_longer(names_to="agg_type", values_to="temp", cols=ends_with('_temp')) %>%
  group_by(Miesiąc, agg_type) %>%
  mutate(temp = temp - mean(temp)) %>%
  ungroup %>%
  mutate(Miesiąc = str_pad(Miesiąc, width=2, pad="0")) %>%
  ggplot(aes(x=Rok, y=temp, color=agg_type)) + geom_smooth(method="lm", se=FALSE) + facet_wrap(~Miesiąc, nrow=3) +
  xlab('Rok') + ylab('Wycentrowane średnie dobowe temperatury') +
  ggtitle("Zmiany dobowych temperatur w podziale na miesiące i rodzaj agregacji") +
  scale_color_discrete(name = "Temperatura", labels = c("Średnia dobowa", "Maksymalna dobowa", "Minimalna dobowa"))
ggsave('month_temp_trends.png', width=8, height=6)
```

9. Wyniki

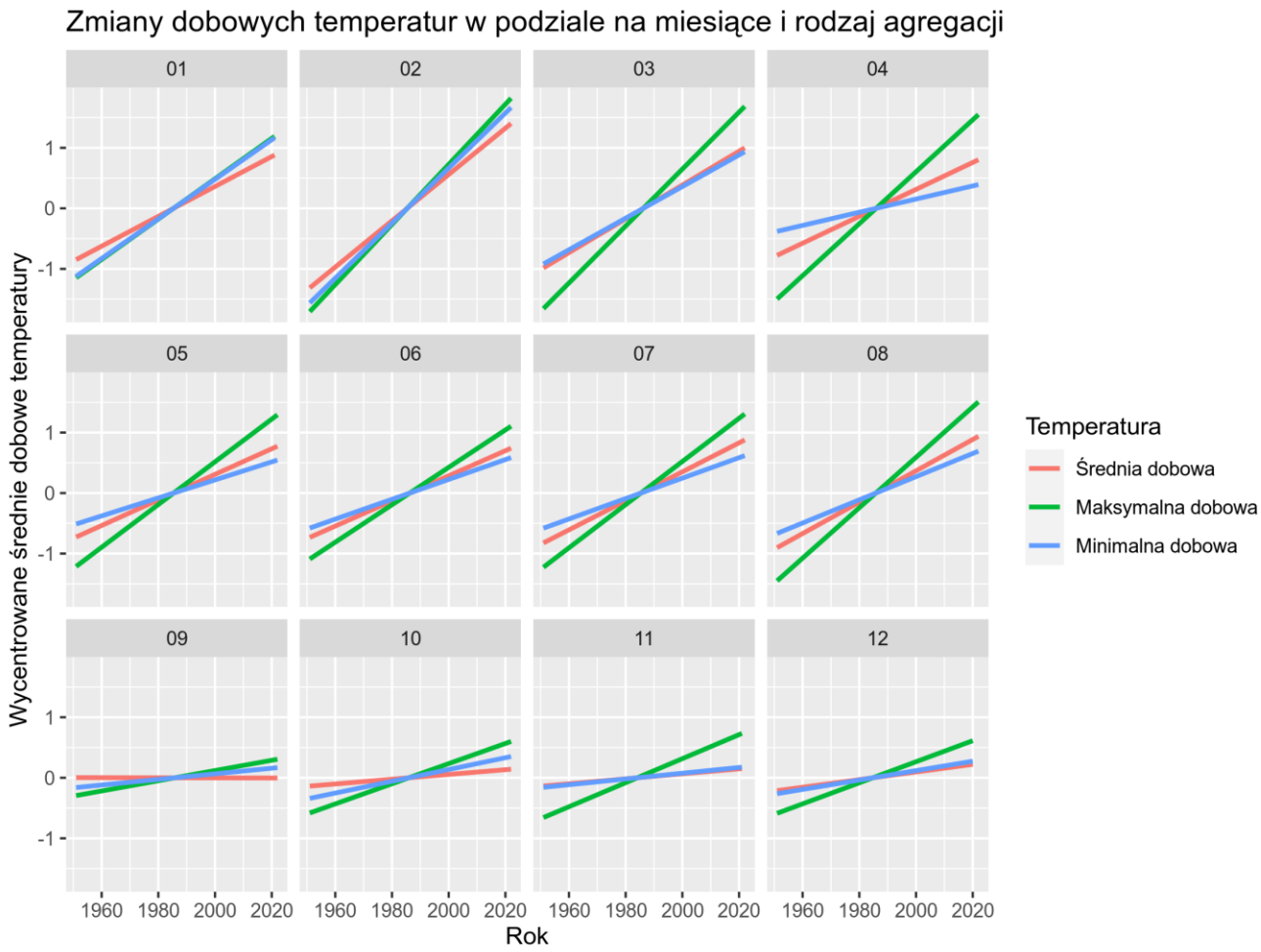
Wykres przedstawiający miejsca w Polsce ze skrajnymi temperaturami. Wśród najzimniejszych dominują miejsca górskie, a najcieplejszych miasta.

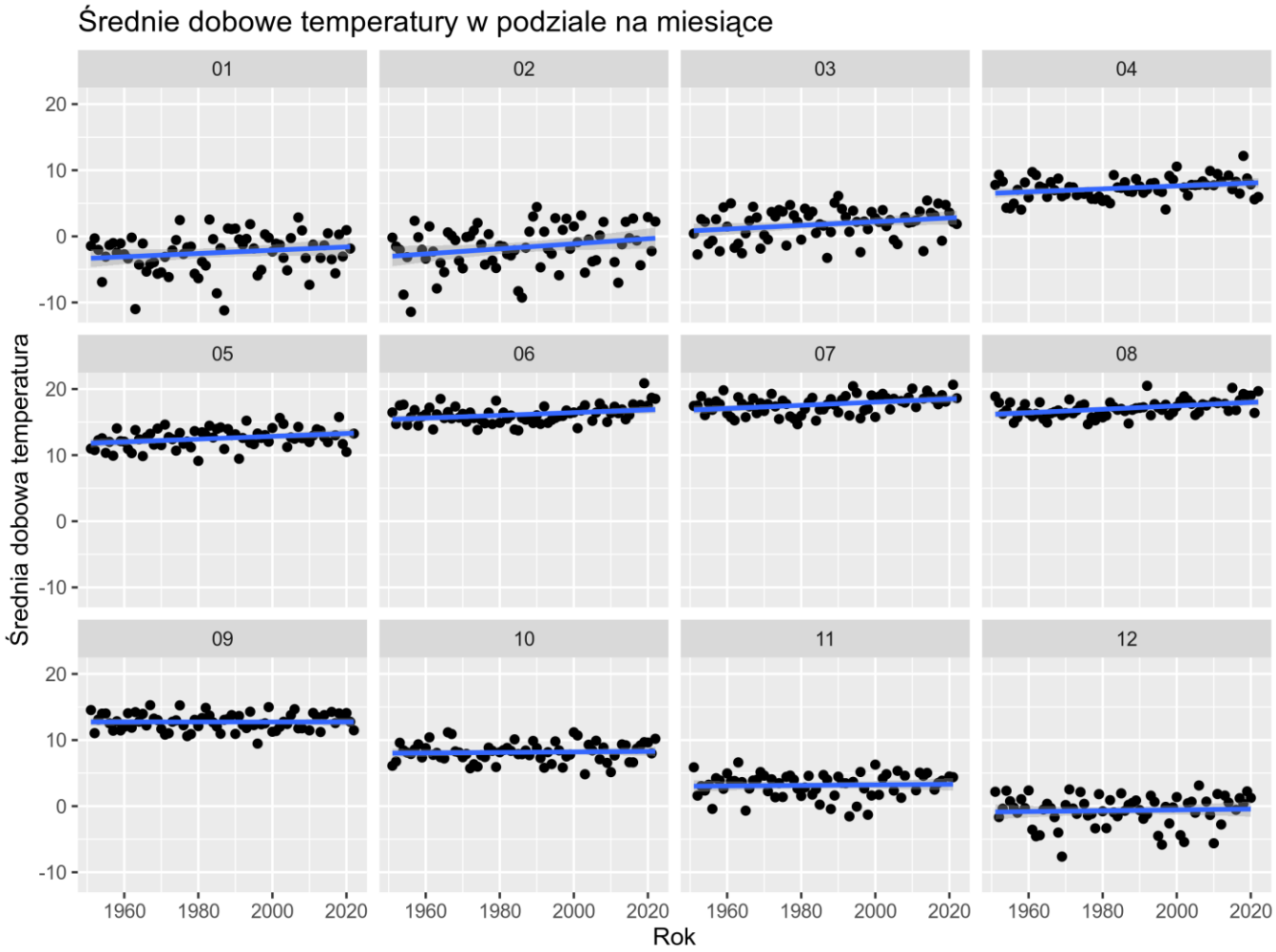


Wykres przedstawiający, ile godzin średnio występuje dane zjawisko w ciągu dnia w danym roku. Interesujący jest fakt, że spadają wartości zmętnienia, czyli mgły z zawieszonym brudem i pyłami.

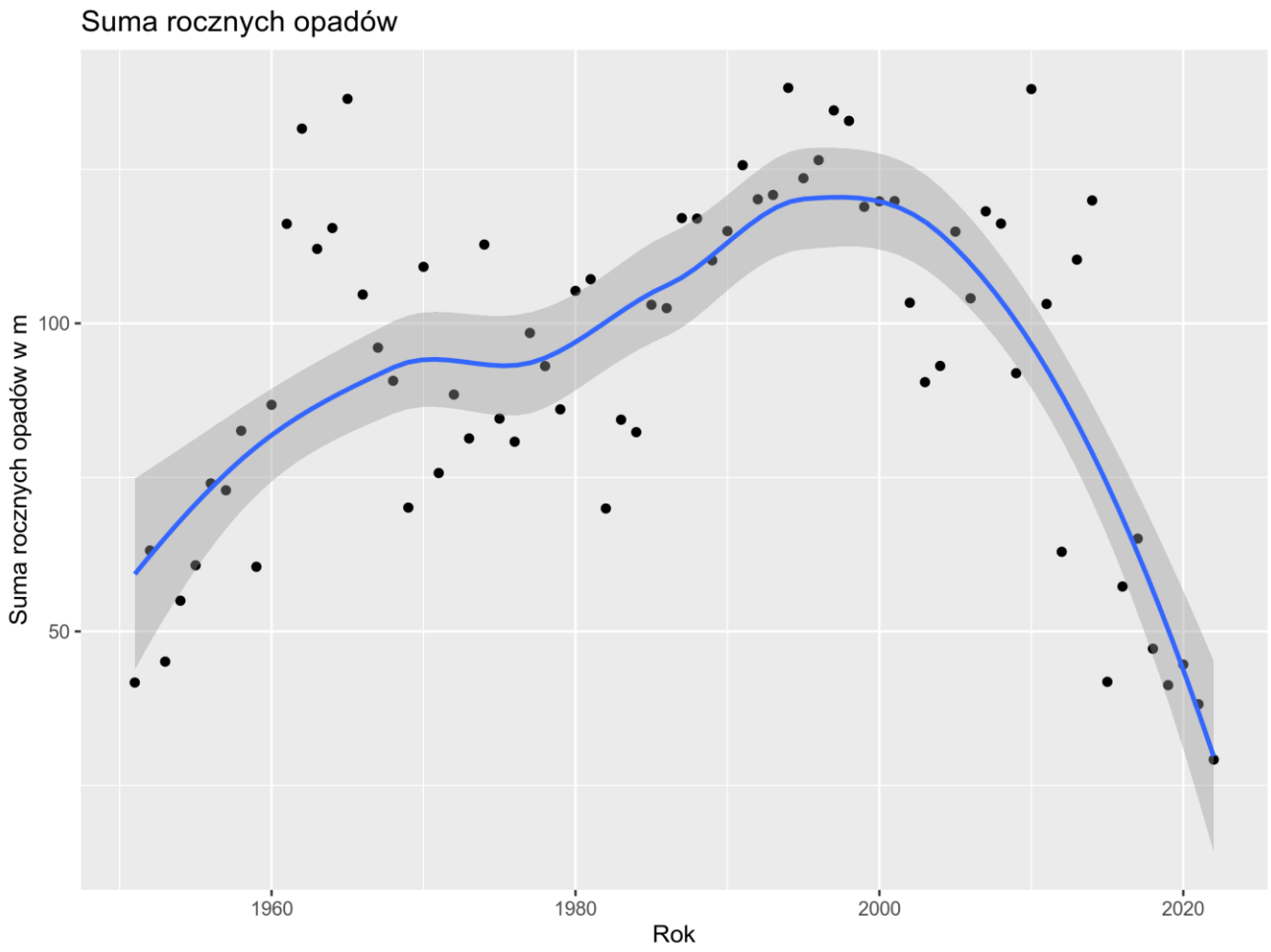


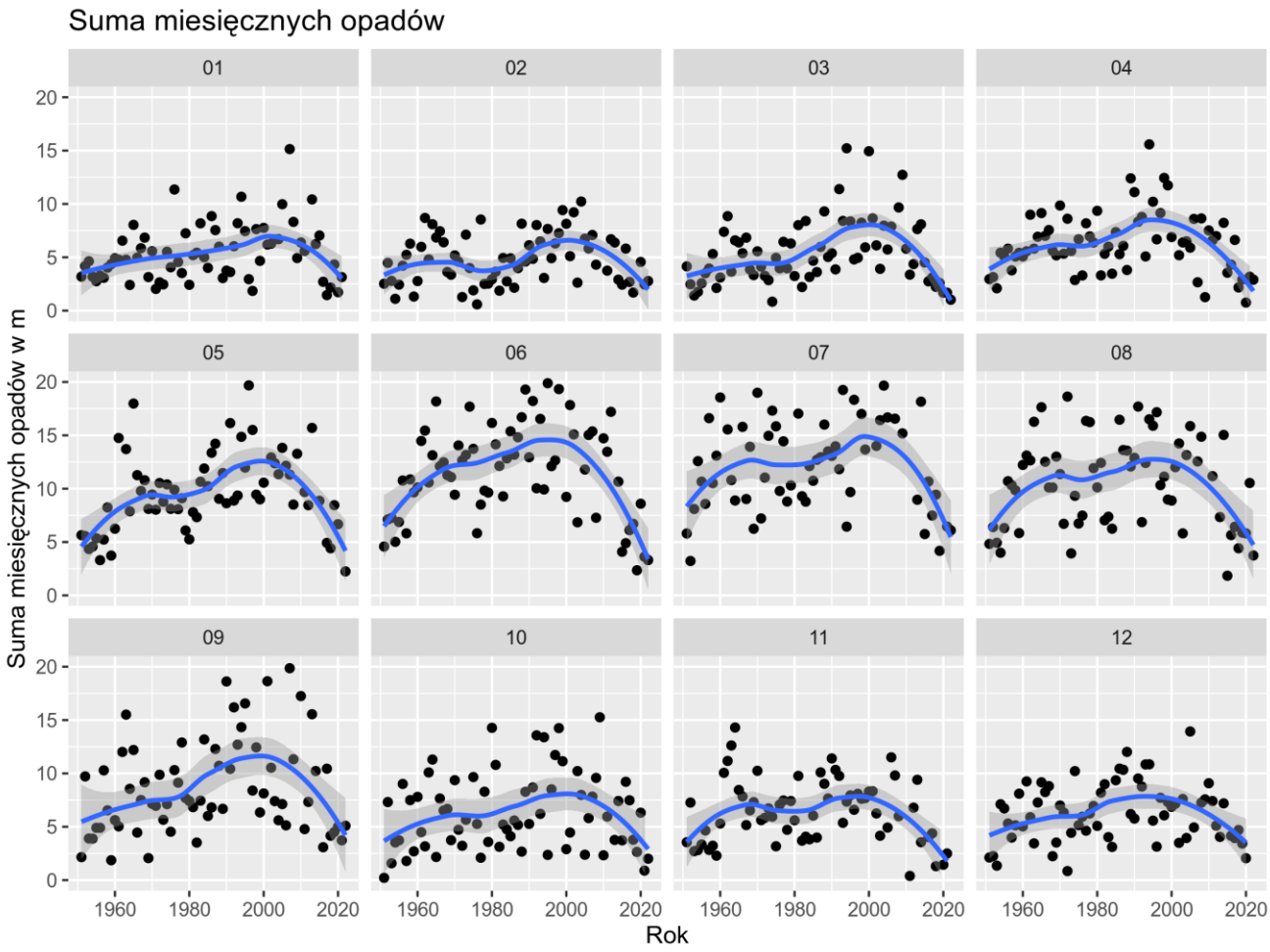
Poniższe wykresy dobitnie pokazują, że zmiany klimatu są faktem. Dla czytelności są pozostawione tylko regresje liniowe z rzeczywistych danych. Widać, że maksymalne dobowe temperatury rosną znacznie szybciej niż średnie i minimalne.


Zmiany dobowych temperatur w podziale na miesiące i rodzaj agregacji
01
02
03
04
05
06
07
08
09
10
11
12
Wycentrowane średnie dobowe temperatury
1
0
-1
1960
1980
2000
2020
Rok
Temperatura
Średnia dobowa
Maksymalna dobowa
Minimalna dobowa



Na wykresach opadów możemy zaobserwować, że susza też jest faktem i w ostatnich latach mamy trend coraz mniejszych opadów deszczu.


Suma rocznych opadów
Suma rocznych opadów w m
100
50
1960
1980
2000
2020
Rok


Suma miesięcznych opadów
01
02
03
04
05
06
07
08
09
10
11
12
Suma miesięcznych opadów w m
0
5
10
15
20
1960
1980
2000
2020
Rok

10. Testy

1. Słownik stacji
   a. Test odkładania pliku do HDFS
      i. Założenia testu:

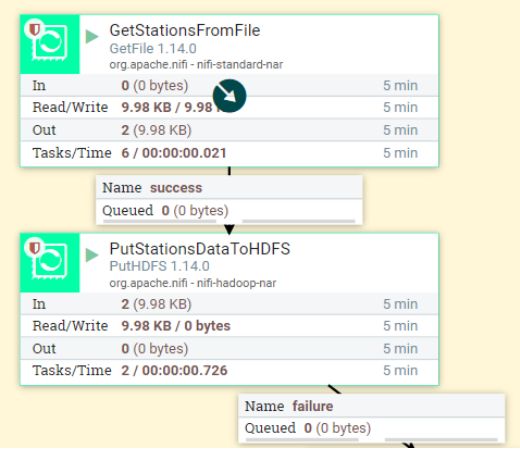
      Pierwszy procesor powinien odczytać I usunąć plik z podanego katalogu, drugi procesor powinien umieścić plik w katalogu hdfs.

      ii. Oczekiwany rezultat:
      Usunięcie pliku z lokalnego katalogu ‘project/nifi/input’. Pojawienie się pliku stations.csv w katalogu ‘/user/project/nifi_in_test’.

      iii. Faktyczny rezultat
      Przed włączenie procesorów:
      W katalogu ‘project/nifi/input’ znajduje się plik stations.csv, katalog ‘/user/project/nifi_in_test’ jest pusty.

```
vagrant@node1:~/project$ ls nifi/input/
stations.csv
```

      Działanie procesorów:



      Po zakończeniu działania:
      W hdfsie pojawił się plik stations.csv

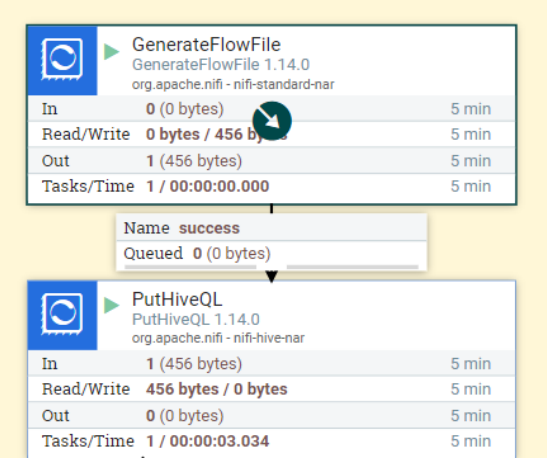
```
vagrant@node1:~/project$ hadoop fs -ls /user/project/nifi_in_test
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/im
pl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLog
gerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
-rw-r--r--   1 root supergroup       5109 2023-01-08 21:38 /user/project/nifi_in_test/stations.csv
```

   b. Test stworzenia tabeli
      i. Założenia testu:

      Pierwszy procesor zawiera skrypt hql do stworzenia tabeli, drugi procesor powinien wykonać skrypt.

ii. Oczekiwany rezultat:
W hive zostanie utworzona tabela stations_orc, powinna zawierać dane z pliku stations.csv.

iii. Faktyczny rezultat:
Działanie procesorów:



W hive pojawiła się nowa tabela, jest wypełniona poprawntmi danymi.

```
vagrant@node1:~/project$ hive -e 'show tables;'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.3.8-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/Stati
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLo
gerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/i
pl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://
23/01/08 21:46:27 [main]: WARN session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authori
ation.manager is set to instance of HiveAuthorizerFactory.
Connected to: Apache Hive (version 2.3.8)
Driver: Hive JDBC (version 2.3.8)
Transaction isolation: TRANSACTION_REPEATABLE_READ
OK
employees
external_table_trams
external_table_trams_part
klimat
salaries
stacje_slownik
stacje_slownik_orc
stations
stations_orc
stations_orc_test
```

```
vagrant@node1:~/project$ hive -e 'select * from stations_orc_test;'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.3.8-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLog
gerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/im
pl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://
23/01/08 21:49:06 [main]: WARN session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authoriz
ation.manager is set to instance of HiveAuthorizerFactory.
Connected to: Apache Hive (version 2.3.8)
Driver: Hive JDBC (version 2.3.8)
Transaction isolation: TRANSACTION_REPEATABLE_READ
23/01/08 21:49:13 [55dd1483-47f0-4cc4-811d-f8f3655e899b main]: ERROR hdfs.KeyProviderCache: Could not find uri with key
[dfs.encryption.key.provider.uri] to create a keyProvider !!
OK
 EPSK Slupsk Poland 54-28N 017-01E 25 ----
12001 ---- Petrobaltic Beta Poland 55-28N 018-10E 46 ----
12100 ---- Kolobrzeg Poland 54-11N 015-35E 3 ----
12105 EPKO Koszalin Poland 54-12N 016-09E 32 ----
12106 ---- Koszalin Zegrze Pom. Poland 54-02N 016-18E 72 ----
12115 ---- Ustka Poland 54-35N 016-52E 6 ----
12120 ---- Leba Poland 54-45N 017-32E 2 ----
```

2. Dane archiwalne

a. Pobieranie zipów z danymi

i. Założenia testu:

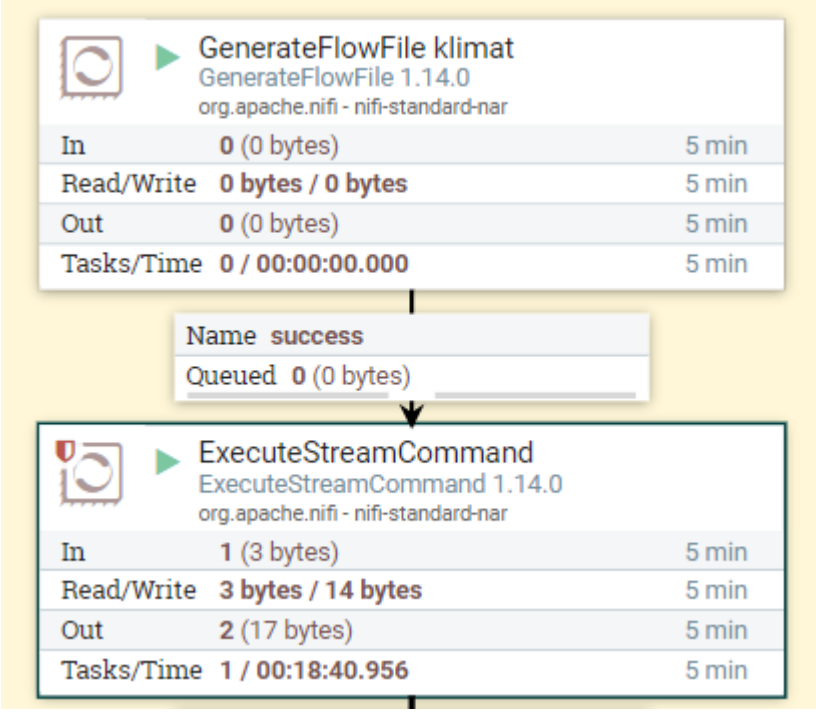
Po uruchomieniu procesor ExecuteStreamCommand powinien uruchomić skrypt python i pobrać dane ze strony IMGW oraz zapisać je w odpowiednich katalogach.

ii. Oczekiwany rezultat:

Utworzone zostaną nowe katalogi na pliki zip i rozpakowane dane, dane zostaną pobrane i rozpakowane do odpowiednich katalogów.

iii. Faktyczny rezultat

Działanie procesorów:



W trakcie pobierania, widać, że przybywa plików:

```
vagrant@node1:~/project/archive_data/test$ ls klimat/zips/
1951_k.zip  1961_k.zip  1971_k.zip  1981_k.zip  1991_k.zip  2001_01_k.zip  2001_11_k.zip  2002_09_k.zip  2003_07_k.zip
1952_k.zip  1962_k.zip  1972_k.zip  1982_k.zip  1992_k.zip  2001_02_k.zip  2001_12_k.zip  2002_10_k.zip  2003_08_k.zip
1953_k.zip  1963_k.zip  1973_k.zip  1983_k.zip  1993_k.zip  2001_03_k.zip  2002_01_k.zip  2002_11_k.zip  2003_09_k.zip
1954_k.zip  1964_k.zip  1974_k.zip  1984_k.zip  1994_k.zip  2001_04_k.zip  2002_02_k.zip  2002_12_k.zip  2003_10_k.zip
1955_k.zip  1965_k.zip  1975_k.zip  1985_k.zip  1995_k.zip  2001_05_k.zip  2002_03_k.zip  2003_01_k.zip  2003_11_k.zip
1956_k.zip  1966_k.zip  1976_k.zip  1986_k.zip  1996_k.zip  2001_06_k.zip  2002_04_k.zip  2003_02_k.zip  2003_12_k.zip
1957_k.zip  1967_k.zip  1977_k.zip  1987_k.zip  1997_k.zip  2001_07_k.zip  2002_05_k.zip  2003_03_k.zip  2004_01_k.zip
1958_k.zip  1968_k.zip  1978_k.zip  1988_k.zip  1998_k.zip  2001_08_k.zip  2002_06_k.zip  2003_04_k.zip  2004_02_k.zip
1959_k.zip  1969_k.zip  1979_k.zip  1989_k.zip  1999_k.zip  2001_09_k.zip  2002_07_k.zip  2003_05_k.zip  2004_03_k.zip
1960_k.zip  1970_k.zip  1980_k.zip  1990_k.zip  2000_k.zip  2001_10_k.zip  2002_08_k.zip  2003_06_k.zip  2004_04_k.zip
vagrant@node1:~/project/archive_data/test$ ls klimat/zips/
1951_k.zip  1962_k.zip  1973_k.zip  1984_k.zip  1995_k.zip     2001_06_k.zip  2002_05_k.zip  2003_04_k.zip  2004_03_k.zip
1952_k.zip  1963_k.zip  1974_k.zip  1985_k.zip  1996_k.zip     2001_07_k.zip  2002_06_k.zip  2003_05_k.zip  2004_04_k.zip
1953_k.zip  1964_k.zip  1975_k.zip  1986_k.zip  1997_k.zip     2001_08_k.zip  2002_07_k.zip  2003_06_k.zip  2004_05_k.zip
1954_k.zip  1965_k.zip  1976_k.zip  1987_k.zip  1998_k.zip     2001_09_k.zip  2002_08_k.zip  2003_07_k.zip  2004_06_k.zip
1955_k.zip  1966_k.zip  1977_k.zip  1988_k.zip  1999_k.zip     2001_10_k.zip  2002_09_k.zip  2003_08_k.zip  2004_07_k.zip
1956_k.zip  1967_k.zip  1978_k.zip  1989_k.zip  2000_k.zip     2001_11_k.zip  2002_10_k.zip  2003_09_k.zip  2004_08_k.zip
1957_k.zip  1968_k.zip  1979_k.zip  1990_k.zip  2001_01_k.zip  2001_12_k.zip  2002_11_k.zip  2003_10_k.zip
1958_k.zip  1969_k.zip  1980_k.zip  1991_k.zip  2001_02_k.zip  2002_01_k.zip  2002_12_k.zip  2003_11_k.zip
1959_k.zip  1970_k.zip  1981_k.zip  1992_k.zip  2001_03_k.zip  2002_02_k.zip  2003_01_k.zip  2003_12_k.zip
1960_k.zip  1971_k.zip  1982_k.zip  1993_k.zip  2001_04_k.zip  2002_03_k.zip  2003_02_k.zip  2004_01_k.zip
1961_k.zip  1972_k.zip  1983_k.zip  1994_k.zip  2001_05_k.zip  2002_04_k.zip  2003_03_k.zip  2004_02_k.zip
vagrant@node1:~/project/archive_data/test$
```

Nowo utworzone katalogi:

```
vagrant@node1:~/project/archive_data/test$ ls
archive_data.py  klimat  opady  synop
```

Pobrane zipy:

```
vagrant@node1:~/project/archive_data/test$ ls klimat/zips/
1951_k.zip  1986_k.zip  2002_09_k.zip  2005_08_k.zip  2008_07_k.zip  2011_06_k.zip  2014_05_k.zip  2017_04_k.zip  2020_03_k.zip
1952_k.zip  1987_k.zip  2002_10_k.zip  2005_09_k.zip  2008_08_k.zip  2011_07_k.zip  2014_06_k.zip  2017_05_k.zip  2020_04_k.zip
1953_k.zip  1988_k.zip  2002_11_k.zip  2005_10_k.zip  2008_09_k.zip  2011_08_k.zip  2014_07_k.zip  2017_06_k.zip  2020_05_k.zip
1954_k.zip  1989_k.zip  2002_12_k.zip  2005_11_k.zip  2008_10_k.zip  2011_09_k.zip  2014_08_k.zip  2017_07_k.zip  2020_06_k.zip
1955_k.zip  1990_k.zip  2003_01_k.zip  2005_12_k.zip  2008_11_k.zip  2011_10_k.zip  2014_09_k.zip  2017_08_k.zip  2020_07_k.zip
1956_k.zip  1991_k.zip  2003_02_k.zip  2006_01_k.zip  2008_12_k.zip  2011_11_k.zip  2014_10_k.zip  2017_09_k.zip  2020_08_k.zip
1957_k.zip  1992_k.zip  2003_03_k.zip  2006_02_k.zip  2009_01_k.zip  2011_12_k.zip  2014_11_k.zip  2017_10_k.zip  2020_09_k.zip
1958_k.zip  1993_k.zip  2003_04_k.zip  2006_03_k.zip  2009_02_k.zip  2012_01_k.zip  2014_12_k.zip  2017_11_k.zip  2020_10_k.zip
```

Rozpakowane pliki:

```
vagrant@node1:~/project/archive_data/test$ ls klimat/files
k_d_01_2001.csv  k_d_05_2003.csv  k_d_09_2005.csv  k_d_1959.csv  k_d_t_03_2005.csv  k_d_t_07_2007.csv  k_d_t_11_2009.csv
k_d_01_2002.csv  k_d_05_2004.csv  k_d_09_2006.csv  k_d_1960.csv  k_d_t_03_2006.csv  k_d_t_07_2008.csv  k_d_t_11_2010.csv
k_d_01_2003.csv  k_d_05_2005.csv  k_d_09_2007.csv  k_d_1961.csv  k_d_t_03_2007.csv  k_d_t_07_2009.csv  k_d_t_11_2011.csv
k_d_01_2004.csv  k_d_05_2006.csv  k_d_09_2008.csv  k_d_1962.csv  k_d_t_03_2008.csv  k_d_t_07_2010.csv  k_d_t_11_2012.csv
k_d_01_2005.csv  k_d_05_2007.csv  k_d_09_2009.csv  k_d_1963.csv  k_d_t_03_2009.csv  k_d_t_07_2011.csv  k_d_t_11_2013.csv
k_d_01_2006.csv  k_d_05_2008.csv  k_d_09_2010.csv  k_d_1964.csv  k_d_t_03_2010.csv  k_d_t_07_2012.csv  k_d_t_11_2014.csv
k_d_01_2007.csv  k_d_05_2009.csv  k_d_09_2011.csv  k_d_1965.csv  k_d_t_03_2011.csv  k_d_t_07_2013.csv  k_d_t_11_2015.csv
k_d_01_2008.csv  k_d_05_2010.csv  k_d_09_2012.csv  k_d_1966.csv  k_d_t_03_2012.csv  k_d_t_07_2014.csv  k_d_t_11_2016.csv
k_d_01_2009.csv  k_d_05_2011.csv  k_d_09_2013.csv  k_d_1967.csv  k_d_t_03_2013.csv  k_d_t_07_2015.csv  k_d_t_11_2017.csv
k_d_01_2010.csv  k_d_05_2012.csv  k_d_09_2014.csv  k_d_1968.csv  k_d_t_03_2014.csv  k_d_t_07_2016.csv  k_d_t_11_2018.csv
k_d_01_2011.csv  k_d_05_2013.csv  k_d_09_2015.csv  k_d_1969.csv  k_d_t_03_2015.csv  k_d_t_07_2017.csv  k_d_t_11_2019.csv
k_d_01_2012.csv  k_d_05_2014.csv  k_d_09_2016.csv  k_d_1970.csv  k_d_t_03_2016.csv  k_d_t_07_2018.csv  k_d_t_11_2020.csv
k_d_01_2013.csv  k_d_05_2015.csv  k_d_09_2017.csv  k_d_1971.csv  k_d_t_03_2017.csv  k_d_t_07_2019.csv  k_d_t_11_2021.csv
k_d_01_2014.csv  k_d_05_2016.csv  k_d_09_2018.csv  k_d_1972.csv  k_d_t_03_2018.csv  k_d_t_07_2020.csv  k_d_t_12_2001.csv
k_d_01_2015.csv  k_d_05_2017.csv  k_d_09_2019.csv  k_d_1973.csv  k_d_t_03_2019.csv  k_d_t_07_2021.csv  k_d_t_12_2002.csv
k_d_01_2016.csv  k_d_05_2018.csv  k_d_09_2020.csv  k_d_1974.csv  k_d_t_03_2020.csv  k_d_t_07_2022.csv  k_d_t_12_2003.csv
```

b. Test przenoszenia plików do hdfsu

i. Założenia testu:

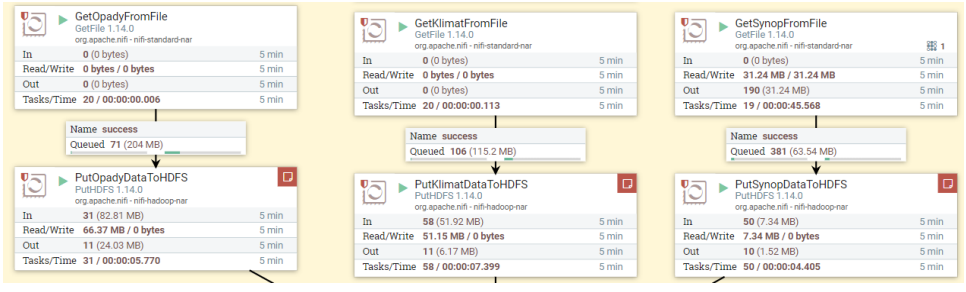
Rozpakowane lokalnie pliki powinny być przeniesione do wyznaczonego katalogu w hdfsie.

ii. Oczekiwany rezultat:

Pliki zostaną usunięte z katalogów lokalnych i przeniesione do katalogów w hdfsie (np. Dla danych o klimacie do katalogu ‘/user/project/archive_klimat_test’.

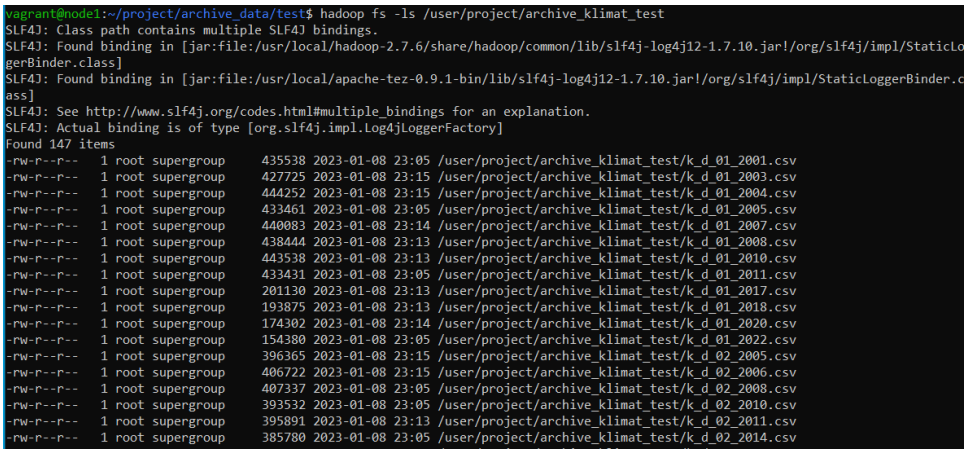
iii. Faktyczny rezultat

Działanie procesorów:



(Tutaj błędy w procesorach są spowodowane problemem z działaniem hdfsu na maszynie wirtualnej, nie przeszkodziły jednak w zapisaniu plików w odpowiednich katalogach w hdfsie)

Pliki w hdfs po zakończeniu przenoszenia:



3. Źródło dynamiczne
   a. Test stworzenia tabeli
      i. Założenia testu:

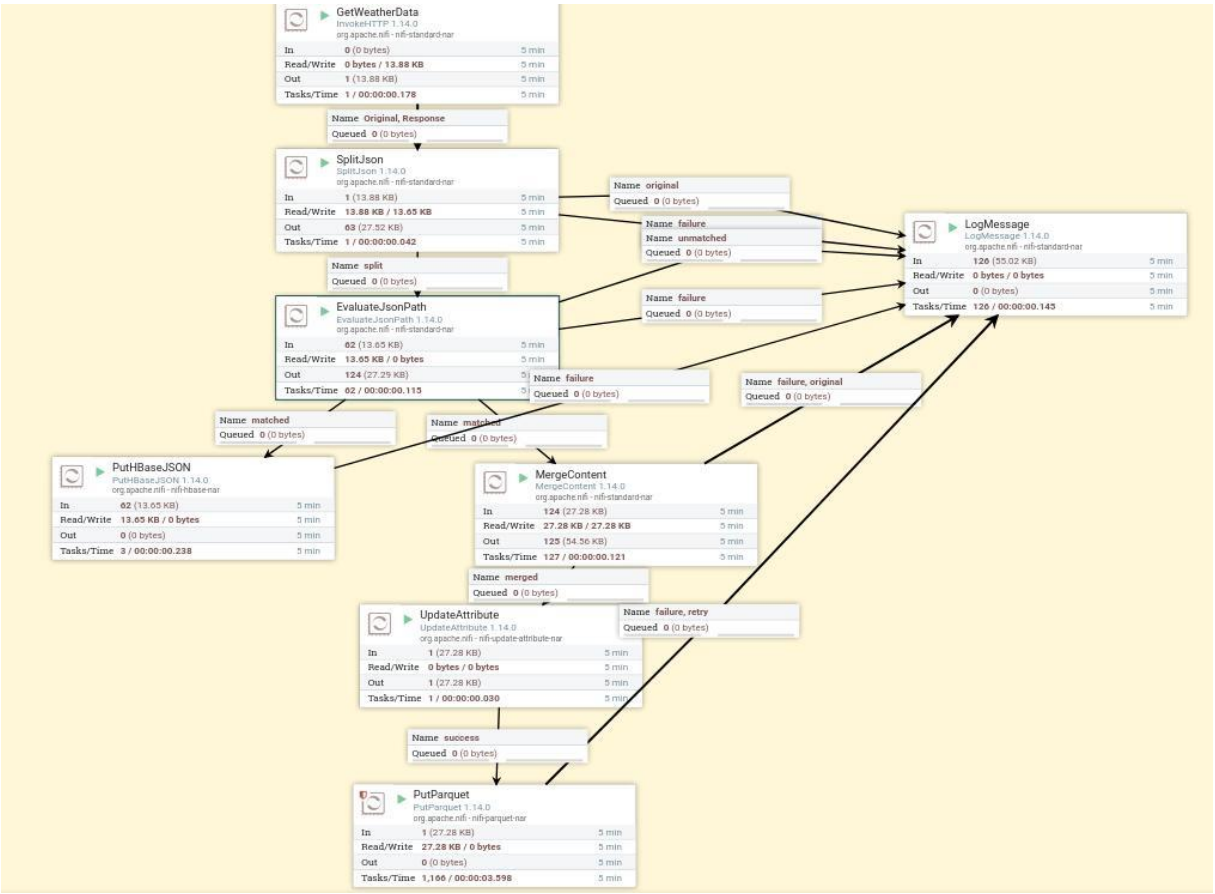
         Stworzenie i cogodzinna aktualizacja tabeli z najnowszymi danymi. Następnie tworzymy nowy plik z odpowiednią sygnaturą czasową.

      ii. Oczekiwany rezultat:

         Stworzenie nowych plików z danymi. Nowy plik powinien pojawiać się co godzinę, w nazwie pliku powinny zostać zapisane godziny pobrania.

      iii. Faktyczny rezultat

         Działające procesory:



Stworzone pliki, w nazwach widać godzinę pobrania:

Po włączeniu

```
^Cvagrant@node1:~/project/archive_data/test$ hadoop fs -ls /user/project/nifi_out_test
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLog
gerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.cl
ass]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
-rw-r--r--   1 root supergroup       5579 2023-01-08 23:43 /user/project/nifi_out_test/api_2023-01-08-234321335Z.parquet
vagrant@node1:~/project/archive_data/test$
```

Po godzinie:

```
vagrant@node1:~$ hadoop fs -ls /user/project/nifi_out_test
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLog
gerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/apache-tez-0.9.1-bin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.cl
ass]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 2 items
-rw-r--r--   1 root supergroup       5579 2023-01-08 23:43 /user/project/nifi_out_test/api_2023-01-08-234321335Z.parquet
-rw-r--r--   1 root supergroup       5608 2023-01-09 00:43 /user/project/nifi_out_test/api_2023-01-09-004311375Z.parquet
vagrant@node1:~$
```

b. Test odkładania danych do tabeli HBase

i. Założenia testu:

Po włączeniu procesorów dane dynamiczne powinny być pobierane z API i ładowanie do tabeli w HBase.

ii. Oczekiwany rezultat:

W tabeli w systemie HBase powinny zostać załadowane dane z API.

iii. Faktyczny rezultat

Do tabeli ‘weather’ zostały załadowane dane.