MapReduce w Apache Spark I

18 listopada 2018

Opis pliku z zadaniami

Wszystkie zadania na zajęciach będą przekazywane w postaci plików `.pdf`, sformatowanych podobnie do tego dokumentu. Zadania będą różnego rodzaju. Za każdym razem będą one odpowiednio oznaczone:

- Zadania do wykonania na zajęciach oznaczone są symbolem $\triangle$ – nie są one punktowane, ale należy je wykonać w czasie zajęć.
- Punktowane zadania do wykonania na zajęciach oznaczone są symbolem $\diamond$ – należy je wykonać na zajęciach i zaprezentować prowadzącemu, w wypadku nie wykonania zadania w czasie zajęć lub nieobecności, zadania staje się zadaniem do wykonania w domu ($\star$).
- Zadania do wykonania w domu oznaczone są symbolem $\star$ – są one punktowane, należy je dostarczyć w sposób podany przez prowadzącego i w wyznaczonym terminie (zwykle przed kolejnymi zajęciami).
- Zadania programistyczne można wykonywać w dowolnym języku programowania, używając jedynie biblioteki standardowej dostępnej dla tego języka.

1 Pierwsze kroki z Apache Spark △

Treść

Na zajęciach będziemy wykorzystywać Apache Spark, w którym będziemy stosować obliczenia oparte na paradygmacie MapReduce.

Do rozpoczęcia pracy należy wykonać następujące kroki:

1. Zapoznać się z najważniejszymi informacjami dotyczącymi Apache Spark: https://spark.apache.org/docs/latest/index.html
2. Pobrać paczkę instalacyjną ze strony: http://spark.apache.org/downloads.html
3. Rozpakować pobraną paczkę, np.: `tar xvfz spark-2.2.0-bin-hadoop2.7.tar`
4. Rozpocząć pracę z systemem poprzez konsolę: `./bin/spark-shell`

2 Proste zadania w Apache Spark △

Treść

W tym zadaniu należy wykonać następujące zadania:

- `WordCount`: Należy napisać program tworzący plik zawierający listę słów wraz z liczbą ich wystąpień w dziełach Szekspira (odpowiedni plik został załączony na stronie przedmiotu). Lista słów powinna zostać posortowana zgodnie z liczbą ich wystąpień. Program powinien wykorzystywać następujące instrukcje: `textFile`, `FlatMap`, `map`, `reduceByKey`, `sortBy(_._2)`, `saveAsTextFile`. Rozwiązanie można znaleźć w materiałach z wykładu. Po uruchomieniu programu należy sprawdzić zapisany wynik i odpowiedzieć na następujące pytania:
  - Jak wyglądają pliki wynikowe?
  - Ile jest takich plików? Dlaczego tak jest?
  - Jak działa `reduceByKey`? Co się stanie, jak zamienimy '`+`' na '`*`'?
- `MatrixVectorMultiplication`: Należy napisać program obliczający iloczyn macierzy i wektora zapisanych w sposób relacyjny (odpowiednie pliki zostały dołączone do strony przedmiotu). Zakładamy, że wektor może zostać rozesłany do wszystkich komputerów w klastrze. Program powinien wykorzystywać następujące instrukcje: `textFile`, `map`, `toInt`, `toDouble`, `collect`, `FlatMap`, `reduceByKey`, `broadcast`, `toDF`, `orderBy`, `show`. Adresowanie elementów listy odbywa się w następujący sposób: `nazwa_listy(i)`, gdzie `i` jest liczbą całkowitą równą lub większą od zera. Rozwiązanie można znaleźć w materiałach z wykładu.
- `ApproximatePI`: Należy napisać program, który przybliża liczbę π metodą Monte Carlo. Program powinien wykorzystywać następujące instrukcje: `parallelize`, `map`, `1 to 10`, `math.random`, `if (.) . else .`, `reduceByKey`. Adresowanie pól krotek odbywa się w następujący sposób: `nazwa_krotki._1`, gdzie `_1` jest pierwszym polem.