Aufgabenblatt 6
Web-Scale Data Management

- Abgabetermin: **Sonntag, 07.02.2021 (23:59 Uhr)**
- Zur Prüfungszulassung muss ein Aufgabenblatt mit mind. 25% der Punkte bewertet werden und alle weiteren Aufgabenblätter mit mindestens 50% der Punkte.
- Die Aufgaben sollen in Zweiergruppen bearbeitet werden.
- Abgabesystem unter `http://www.dcl.hpi.uni-potsdam.de/submit`
  - für die ersten beiden Aufgaben: jeweils eine pdf-Datei, bitte mit Namen beschriftet
  - Bei der Programmieraufgabe: bitte nur die ensprechende Scala-Klassendatei (Exercise_3x.scala), bzw falls Sie weitere Hilfsklassen erzeugt haben, alle Klassen als .zip Archiv (das sollte aber eingentlich nicht nötig sein)

Aufgabe 1: Apache Spark - Einführung

In diesem Aufgabenblatt befassen wir uns mit Apache Spark, einem Framework für verteilte Ausführung, bzw. Cluster Computing. Unter `https://github.com/HPI-Information-Systems/spark-tutorial` (Branch: DBS_II_Excercise_7) finden Sie in dem ausführbaren Objekt `de.hpi.spark_tutorial.SparkIntroduction` eine allgemeine Einführung zu Apache Spark. Klonen Sie das Repository und führen die Anweisungen in der Readme aus, sodass Sie den Code *SparkIntroduction* lokal bei sich ausführen können. Schauen Sie sich anschließend die dort gezeigten Beispiele an.

Beantworten Sie die folgenden Wissensfragen zur Wiederholung und Vertiefung. Alle Fragen beziehen sich auf die konkrete Umsetzung in Apache Spark und nicht auf allgemeine Konzepte zum Thema Map-Reduce.

a) Was ist die Map-Funktion, worauf wird sie ausgeführt und was erhält sie als Parameter? **1 P**

b) Was ist die Reduce-Funktion, worauf wird sie ausgeführt und was erhält sie als Parameter? Verwenden Sie ggf. ein kleines Beispiel zur Erklärung. **2 P**

c) Was ist der Unterschied zwischen Dataframes und Datasets? **1 P**

d) Mit welcher Funktion gruppiert man in Apache Spark Datasets nach Schlüsseln? **1 P**

e) Was ist der Unterschied zwischen Transformations und Actions? Nennen Sie für beides jeweils eine Beispiel-Methode aus der Spark Library. **2 P**

Aufgabe 2: Typische Programmier- und Verständnisfehler in Apache Spark

Da Apache Spark die verteilte Ausführung sehr gut versteckt und diverse Optimierungen vornimmt, welche die Ausführung des Codes beeinflussen, gibt es einige typische Programmierfehler, denen wir mit den folgenden Aufgaben vorbeugen wollen:

a) Beschreiben Sie kurz, was der Driver-Prozess ist und was der Unterschied zu den Worker-Prozessen ist. **2 P**

b) Geben Sie für das folgende Spark-Programm für folgende Code-Zeilen jeweils an, ob die Anweisung im Driver-Prozess oder im Worker Prozess ausgeführt wird: Zeilen 11,13,18,23

```
10  def execute(dataset: Dataset[String]): Unit = {
11    println("Executing a new spark program")
12    val keyToSize = dataset
13      .map(string => string.take(10))
14      .groupByKey(identity)
15      .mapGroups{case (prefix,occurrences) => (prefix,occurrences.size)}
16    keyToSize
17      .filter{ keyValuePair => keyValuePair._2 > 100}
18      .foreach{case (prefix,occurrenceCount) => doSomething(prefix,occurrenceCount)}
19    val collected = keyToSize
20      .filter(keyValuePair => keyValuePair._1.startsWith("Test"))
21      .collect()
22    collected
23      .foreach{case (prefix,occurenceCount) => doSomethingElse(prefix,occurenceCount)}
24    println("Finished Executing a new spark program")
25  }
```

2 P

c) Betrachten Sie folgendes Spark Programm. Welches Problem gibt es mit diesem Programm? Begründen Sie ihre Antwort.

```
12  def execute(dataset: Dataset[String]): Unit = {
13    var numStringsCointaingA = 0
14    dataset
15      .foreach(string => {
16        if(string.contains("A")){
17          numStringsCointaingA += 1
18        }
19      })
20    println(numStringsCointaingA)
21  }
```

2 P

d) Betrachten Sie folgendes Spark Programm (die Markierung von *size* in Zeile 36 hat nichts zu bedeuten). Nehmen Sie an, dass die Methode *callMyWebsite* die Webseite aufruft, `www.myCountingWebsite.com` alle Aufrufe zählt und die Methode *getWebsiteCallCount* den Wert dieses Counters zurückgibt. Welches Problem gibt es mit diesem Programm? Begründen Sie ihre Antwort.

```
32  def execute(dataset: Dataset[String]): Unit = {
33    val responses = dataset
34      .map(param => callMyWebsite( str = "www.myCountingWebsite.com", param))
35      .collect()
36    val responseCount = responses.size
37    val websiteCallCount = getWebsiteCallCount( str = "www.myCountingWebsite.com")
38    assert(websiteCallCount == responseCount)
39  }
```

2 P

e) Betrachten Sie folgendes Spark Programm. Welches Problem gibt es mit diesem Programm? Begründen Sie ihre Antwort.

```scala
def measureExecutionTime(dataset: Dataset[String]):Long = {
  val timeBefore = System.currentTimeMillis()
  val result = dataset
    .map(string => superLongAndComplexMethod(string))
  val timeAfter = System.currentTimeMillis()
  val executionTime = timeAfter-timeBefore
  println(s"Wow, the execution only took ${timeAfter-timeBefore}ms")
  println("Now all that remains is writing the results, this should be super fast:")
  result.write.csv( path = "myResultFile")
  return executionTime
}
```

2 P

Aufgabe 3: Change Exploration mit Apache Spark

Für die folgende Programmierübung nutzen wir das package `de.hpi.dbsII_exercises` aus dem gleichen Repository, in dem auch die Einführung zu finden ist (`https://github.com/HPI-Information-Systems/spark-tutorial`, Branch: DBS_II_Excercise_7). Das Main-Object `de.hpi.dbsII_exercises.DBSIISparkExerciseMain` führt nacheinander alle im folgenden beschreibenen Teilaufgaben aus. Eure Aufhabe ist es in den entsprechenden benannten Klassen jeweils die *execute()* Methode zu vervollständigen, sodass das gewünschte Ergebnis zurückgegeben wird (der geforderte Rückgabetyp ist in der Methodensignatur definiert). Ändern Sie sonst nichts an den bestehenden Klassen. Die Main-Klasse erwartet zwei Parameter: den Pfad zu den Daten, sowie die Anzahl an Cores, die für die lokale Ausführung genutzt werden sollen.

Einen Auszug der Input-Daten finden Sie unter `https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/lehre/WS2020/DBS_II/Test_Input_Spark.zip`. Wir bezeichnen diese Daten als Change-Records. Diese beschreiben Änderungen, welche in relationalen Tabellen gemacht wurden. Ein einzelner ChangeRecord beschreibt die Änderung des Wertes eines Attributes für eine Entität auf einen neuen Wert zu einem bestimmten Zeitpunkt. Beipiel: In Tabelle *DBSII-Teilnehmer* hat sich am *20.02.2021* in Entität *35* der Wert für das Attribut *Note* auf *1,7* geändert. Die Scala Case-Class `de.hpi.dbsII_exercises.ChangeRecord` bildet diese Daten ab. **Hinweis: Die Felder *entityID*, sowie *attributeName* sind immer nur innerhalb einer Tabelle (*tableID*) eindeutig.**

Zur Selbstkontrolle prüft *DBSIISparkExerciseMain* nach der Ausführung aller Aufgaben die zurückgegebenen Ergebnisse und gibt auf der Konsole aus, ob diese korrekt ausgeführt wurden (im Bezug auf die Daten unter `https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/lehre/WS2020/DBS_II/Test_Input_Spark.zip`).

Gegeben ist jeweils ein Spark Dataset mit ChangeRecords. Lösen Sie folgende Aufgaben:

a) Geben Sie die IDs aller Tabellen zurück (Jede ID nur einmal). Die IDs sollen alpha-numerisch sortiert sein. **1 P**

b) Geben Sie für jede Tabelle die Anzahl der Attribute zurück. **3 P**

c) Machen Sie einen Datenqualitätscheck: Wir würden annehmen, dass es in jeder Tabelle zu jedem Zeitpunkt für jedes Feld (identifiziert durch die Kombination aus *entityID* und *attributeName*) nur einen einzigen Wert geben kann. Geben Sie für alle Felder aller Tabellen, bei denen dies nicht so ist für jeden Zeitpunkt alle Werte aus, die zu diesem Zeitpunkt vorkommen. **3 P**

d) Bestimmen Sie die Change-Signatures aller Attribute. Die Change Signature ist eine sortierte Liste an Zeitpunkten, zu denen viele Änderungen an dem Attribut stattfanden. Ein Zeitpunkt soll genau dann in der Change Signature eines Attributes *A* auftauchen, wenn mindestens 100 entitities zu diesem Zeitpunkt im Attribut *A* ihren Wert geändert haben. Geben Sie für jede Change-Signature alle Attribute (Identifiziert durch die Kombination aus *tableID* und *attributeName*) zurück, die diese Change-Signature aufweisen. **7 P**