Analyse von Flugdaten mit Hadoop MapReduce:
Big Data Storage

vorgelegt am 9. April 2024

Fakultät Wirtschaft und Gesundheit
Studiengang Wirtschaftsinformatik
Kurs WWI2022F

von

Abdulaziz Al-Surabi | 5438747

Contents

1 Einleitung **3**
- 1.1 Hintergrund . . . 3
- 1.2 Ziel der Dokumentation . . . 3
- 1.3 Aufgabenstellung . . . 3

2 System-Vorbereitung **4**
- 2.1 Systemvoraussetzungen . . . 4
- 2.2 Hadoop Konfiguration . . . 4

3 Schritte zur Vorbereitung des MapReduce-Jobs **5**
- 3.1 Kompilieren und Packen der Java-Datei mit Maven . . . 5
  - 3.1.1 Öffnen der Maven-Projektansicht in IntelliJ IDEA . . . 5
  - 3.1.2 Ausführung der Maven Lifecycle-Phasen . . . 5

4 Job 1: Kundenpunkte Berechnung **6**
- 4.1 Code-Überblick und Funktion . . . 6
- 4.2 Job-Konfiguration und -Ausführung . . . 7
- 4.3 Ausführung des MapReduce-Jobs . . . 8
- 4.4 Überprüfung der Ergebnisse des ersten Jobs . . . 8
  - 4.4.1 Alternative Überprüfung im HDFS Web Interface . . . 8

5 Job 2: Gruppierung der Kundenpunkte **9**
- 5.1 Code-Überblick und Funktion . . . 9
- 5.2 Kompilieren und Packen der Java-Datei für Job 2 mit Maven . . . 10
- 5.3 Ausführen des zweiten MapReduce-Jobs . . . 10
- 5.4 Überprüfung der Ergebnisse des zweiten Jobs . . . 10
  - 5.4.1 Alternative Überprüfung im HDFS Web Interface . . . 11
- 5.5 Bereinigung des HDFS . . . 11

6 Quellen **12**

1 Einleitung

1.1 Hintergrund

In der heutigen Zeit, in der die Menge an verfügbaren Daten exponentiell wächst, ist es von entscheidender Bedeutung, effektive Werkzeuge und Methoden für die Datenverarbeitung und -analyse zu entwickeln. Hadoop MapReduce stellt eine solche leistungsstarke Technologie dar, die es ermöglicht, große Datenmengen effizient über verteilte Systeme zu verarbeiten. Die Fähigkeit, aus großen Datensätzen wertvolle Einsichten zu gewinnen, hat Hadoop zu einem unverzichtbaren Werkzeug in vielen Branchen gemacht, von der Finanzwelt bis hin zur wissenschaftlichen Forschung.

Die Aufgabenstellung, die in dieser Dokumentation behandelt wird, wurde im Rahmen des Moduls "Big Data Storage" gestellt. Dieses Modul konzentriert sich auf die Speicherung, Verwaltung und Analyse von Big Data, wobei ein besonderes Augenmerk auf Technologien wie Hadoop gelegt wird. Die spezifische Herausforderung besteht darin, eine Datenanalyse der Datei fluege.csv durchzuführen, die Flugdaten enthält. Ziel ist es, durch die Anwendung von Hadoop MapReduce-Techniken tiefergehende Einblicke in das Kundenverhalten und die Nutzung von Flugpunkten zu erlangen.

1.2 Ziel der Dokumentation

Das primäre Ziel dieser Dokumentation ist es, den Prozess der Durchführung der MapReduce-Jobs, von der Vorbereitung der Daten bis hin zur Auswertung der Ergebnisse, detailliert zu beschreiben. Die Dokumentation soll dabei als eine umfassende Anleitung dienen, die es dem Leser ermöglicht, die einzelnen Schritte der Datenverarbeitung nachzuvollziehen und die durchgeführten MapReduce-Jobs zu verstehen.

Des Weiteren dient diese Dokumentation dazu, die technische Umsetzung der spezifizierten Aufgabenstellung zu erläutern und dabei die Designentscheidungen und Implementierungsdetails zu beleuchten. Sie soll auch als Nachweis der individuellen Leistung im Rahmen des Moduls dienen und dabei sicherstellen, dass die Lösung der Aufgabenstellung nachvollziehbar und reproduzierbar ist.

Schließlich wird die Dokumentation als ein Instrument der Reflexion über die verwendeten Methoden und Technologien verwendet. Sie bietet die Möglichkeit, über die Effektivität der gewählten Ansätze zu reflektieren und potenzielle Verbesserungen oder Alternativen zu diskutieren, die in zukünftigen Projekten angewendet werden könnten.

1.3 Aufgabenstellung

- **Summe der Punkte pro Kunde:** Für jeden Kunden sollen die insgesamt erworbenen Punkte sowie die Anzahl der eingelösten Punkte berechnet werden. Diese Berechnung dient als Grundlage, um das Kundenverhalten hinsichtlich der Nutzung von Flugpunkten zu verstehen.

- **Kundensegmentierung nach Punkten:** Die Kunden sollen auf Basis der erworbenen Punkte in drei unterschiedliche Gruppen eingeteilt werden:

  1. Gruppe 1: Kunden mit höchstens 2000 erworbenen Punkten.
  2. Gruppe 2: Kunden, die zwischen 2001 und 10000 Punkte erworben haben.
  3. Gruppe 3: Kunden mit mindestens 10001 erworbenen Punkten.

  Diese Segmentierung ermöglicht eine differenzierte Betrachtung der Kunden nach ihrem Punkteaufkommen und bietet Ansätze für zielgerichtete Marketingmaßnahmen oder Kundenbindungsprogramme.

- **Ausgabe von Gruppeninformationen:** Für jede der definierten Gruppen sollen folgende Informationen ausgegeben werden:
  - Die Anzahl der Kunden in der Gruppe.
  - Die Gesamtsumme der von dieser Gruppe erworbenen Punkte.
  - Der prozentuale Anteil der eingelösten Punkte im Verhältnis zu den insgesamt erworbenen Punkten.

  Diese Informationen geben Aufschluss über das Engagement und die Aktivität der Kunden in den jeweiligen Segmenten.
- **Numerische Werte ohne Nachkommastellen:** Sämtliche Berechnungen und Ausgaben sollen ohne Nachkommastellen erfolgen, d.h., alle numerischen Werte sind entsprechend zu runden.
- **Implementierung von MapReduce-Programmen:** Zur Bewältigung dieser Aufgabenstellung sollen zwei MapReduce-Programme implementiert werden. Beide Programme sollen jeweils einen Combiner-Schritt beinhalten, um die Effizienz der Datenverarbeitung zu optimieren.
- **Definition von HDFS-Verzeichnissen:** Es sind drei Verzeichnisse im Hadoop Distributed File System (HDFS) zu definieren, um den Prozess zu strukturieren:
  - Ein Verzeichnis für die Eingabedaten (`input`).
  - Ein Verzeichnis für die Zwischenergebnisse (`intermediate-output`).
  - Ein Verzeichnis für die Endresultate (`final-output`).

2 System-Vorbereitung

Bevor Sie die Hadoop MapReduce-Jobs ausführen, müssen einige vorbereitende Schritte durchgeführt werden, um sicherzustellen, dass Ihr System richtig konfiguriert ist und alle notwendigen Dienste ausgeführt werden.

2.1 Systemvoraussetzungen

Überprüfen Sie, ob Ihr System die folgenden Anforderungen erfüllt:

- Hadoop 3.2.2 ist installiert und konfiguriert.
- Java Development Kit (JDK) ist installiert und die Umgebungsvariablen sind gesetzt.
- Maven ist installiert und konfiguriert, um das Projekt zu bauen.

2.2 Hadoop Konfiguration

1. **Wechseln zum Hadoop-Benutzer**

   Um Hadoop-Befehle ausführen zu können, müssen Sie sich als Hadoop-Benutzer anmelden:

   ```
   su - hdoop
   ```

   Wenn Sie dazu aufgefordert werden, geben Sie das entsprechende Passwort `student` für den Benutzer `hdoop` ein.

2. **Starten der Hadoop-Dienste**

   Um alle erforderlichen Hadoop-Dienste zu starten, führen Sie das Startskript aus:

```
./start-hadoop.sh
```

Dieses Skript startet alle notwendigen Dienste, darunter den Namenode, den Datanode, den ResourceManager und den NodeManager.

3. **Überprüfung der Hadoop-Dienste**

   Stellen Sie sicher, dass alle Dienste korrekt gestartet wurden. Sie können dies überprüfen, indem Sie die Dienststatus oder das Web-UI von Hadoop besuchen. Die Standard-URLs für das Hadoop Web-UI sind:

   - Namenode: `http://localhost:9870/`
   - ResourceManager: `http://localhost:8088/`

4. **Erstellung eines HDFS-Eingabeverzeichnisses**

   Das HDFS (Hadoop Distributed File System) muss für die Eingabedaten konfiguriert werden. Verwenden Sie den folgenden Befehl, um ein Eingabeverzeichnis zu erstellen:

   ```
   hdfs dfs -mkdir -p /user/hdoop/input
   ```

5. **Hochladen der Daten ins HDFS**

Nachdem das Eingabeverzeichnis erstellt wurde, müssen Sie die zu verarbeitenden Daten in das HDFS hochladen. Verwenden Sie den Befehl `hdfs dfs -put`, um Ihre Datei `fluege.csv` ins HDFS zu kopieren:

```
hdfs dfs -put /path/to/local/fluege.csv input
```

Ersetzen Sie `/path/to/local/fluege.csv` durch den tatsächlichen Pfad Ihrer lokalen Datei `fluege.csv`.

In meinem Fall:

```
hdfs dfs -put /home/student/PERSISTENT/Bigdata_abgabe/Hadoop_Abgabe/data/
    ↪ fluege.csv input
```

3 Schritte zur Vorbereitung des MapReduce-Jobs

3.1 Kompilieren und Packen der Java-Datei mit Maven

Bevor der MapReduce-Job ausgeführt werden kann, müssen die Java-Dateien kompiliert und in eine ausführbare JAR-Datei gepackt werden. Dieser Prozess wird mit Maven durchgeführt, einem Software-Management- und Automatisierungstool, das speziell für Java-Projekte entwickelt wurde.

3.1.1 Öffnen der Maven-Projektansicht in IntelliJ IDEA

Um Maven in IntelliJ IDEA zu verwenden, navigieren Sie zu `View > Tool Windows > Maven`. Siehe Abbildung 1.

3.1.2 Ausführung der Maven Lifecycle-Phasen

Im Maven-Projekte-Fenster führen Sie die Lifecycle-Phasen `compile` und `package` aus. Dies kompiliert Ihr Projekt und packt es in eine JAR-Datei. Überprüfen Sie nach Abschluss, ob die JAR-Datei im `target`-Verzeichnis Ihres Projekts erstellt wurde. Siehe Abbildung 2.

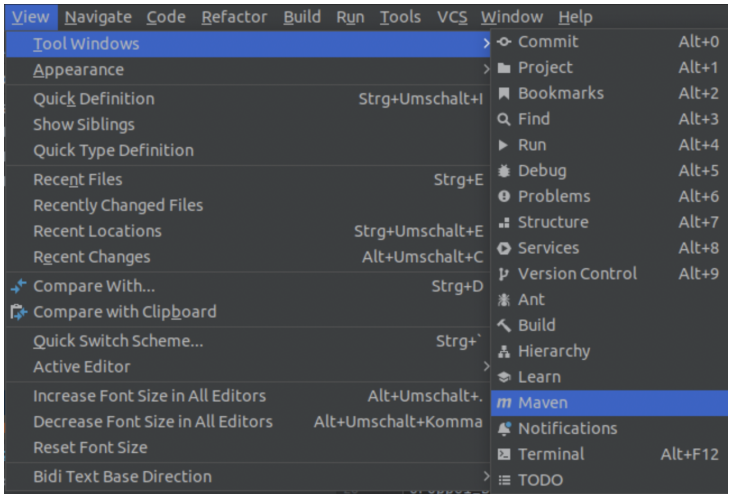

Figure 1: Maven-Projekte-Fenster in IntelliJ IDEA

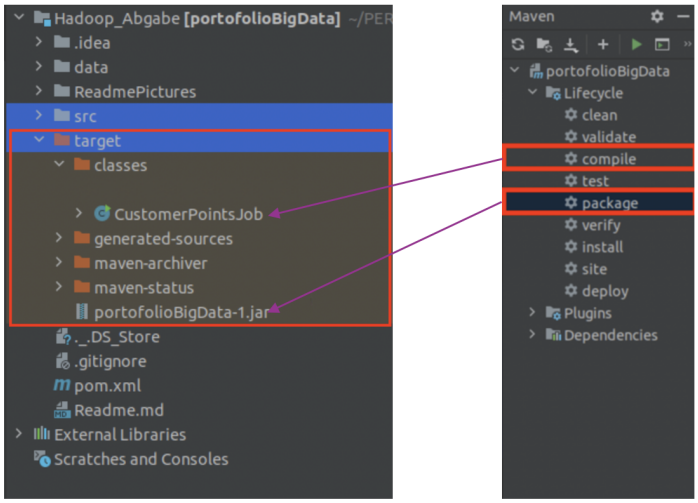

Figure 2: Ausführen der Maven-Phasen `compile` und `package`

4 Job 1: Kundenpunkte Berechnung

Dieser Abschnitt erläutert den Hadoop MapReduce-Job ‘CustomerPointsJob‘, der für die Berechnung der Summe der erworbenen und eingelösten Punkte jedes Kunden zuständig ist.

4.1 Code-Überblick und Funktion

Der Job implementiert benutzerdefinierte Klassen und Methoden, um die Kundendaten effektiv zu verarbeiten und zu gruppieren.

Die `PointsTuple` Klasse

Die `PointsTuple` Klasse ist ein benutzerdefinierter `Writable`-Datentyp, der für die Übertragung der Punktedaten zwischen dem Mapper und dem Reducer verwendet wird. Sie enthält zwei Felder: `accumulatedPoints` für die Summe der erworbenen Punkte und `redeemedPoints` für die Summe der eingelösten Punkte.

- `write(DataOutput out)` und `readFields(DataInput in)` Methoden dienen der Serialisierung bzw. Deserialisierung der Objekte, was für die Datenübertragung im Hadoop-Ökosystem erforderlich ist.
- Über die Getter- und Setter-Methoden können die Punktwerte gesetzt und abgerufen werden.

Der Mapper

Der `CustomerPointsMapper` analysiert jede Eingabezeile (CSV-Format), extrahiert die Kunden-ID und die Punktedaten und überspringt dabei die Kopfzeile. Für jede gültige Zeile erzeugt der Mapper ein Schlüssel-Wert-Paar, wobei der Schlüssel die Kunden-ID ist und der Wert ein `PointsTuple`-Objekt, das die erworbenen und eingelösten Punkte enthält.

Der Reducer

Der `CustomerPointsReducer` empfängt alle `PointsTuple`-Objekte, die zu einer bestimmten Kunden-ID gehören, und summiert die erworbenen und eingelösten Punkte. Das Ergebnis ist ein neues `PointsTuple`-Objekt pro Kunde, welches die Gesamtsummen enthält und in das Ausgabeverzeichnis geschrieben wird.

Job-Konfiguration und -Ausführung

Der Job wird durch die `main`-Methode konfiguriert und gestartet:

- Die `Configuration` und `Job`-Instanz wird erstellt und mit der `CustomerPointsJob` Klasse verknüpft.
- Mapper und Reducer Klassen sowie die Ein- und Ausgabeformate werden festgelegt.
- Der Input-Path (Quelldaten) und der Output-Path (Zielverzeichnis für die Ergebnisse) werden definiert.
- Schließlich wird der Job gestartet und auf seine Beendigung gewartet.

Durch die Ausführung dieses Jobs erhalten wir eine detaillierte Übersicht über die Punkteaktivitäten jedes Kunden, was die Grundlage für weitere Analysen oder Geschäftsentscheidungen bietet.

4.2 Job-Konfiguration und -Ausführung

Zur Ausführung des Jobs wird die Konfiguration wie folgt festgelegt:

- Der Job verwendet die `CustomerGroupMapper`- und `CustomerGroupReducer`-Klassen.
- Der Output-Key ist vom Typ `Text`, der die Gruppen-ID darstellt.
- Der Output-Value ist vom Typ `Text`, der die Zusammenfassung der Gruppenstatistiken enthält.
- Die Pfade für die Ein- und Ausgabedaten werden durch die Argumente beim Start des Jobs festgelegt.

Die Ausführung dieses Jobs ermöglicht es, tiefergehende Einblicke in die Verteilung der Kundenpunkte zu gewinnen und die Kunden basierend auf ihrem Punktestand effektiv zu segmentieren.

4.3 Ausführung des MapReduce-Jobs

Die Ausführung des Jobs umfasst die Initialisierung der Job-Konfiguration, die Festlegung der Ein- und Ausgabepfade sowie die Definition der Mapper- und Reducer-Klassen. Der Job wird mit folgendem Befehl gestartet:

```
hadoop jar <pfad-zur-jar-datei> <Main-Klasse> <Eingabeverzeichnis> <
    ↪ Ausgabeverzeichnis>
```

in meinem Fall:

```
hadoop jar /home/student/PERSISTENT/Bigdata_abgabe/Hadoop_Abgabe/target/
    ↪ portofolioBigData-1.jar CustomerPointsJob input intermediate-output
```

Dies startet den MapReduce-Prozess, der die Daten verarbeitet und die Ergebnisse im spezifizierten Ausgabeverzeichnis speichert.

4.4 Überprüfung der Ergebnisse des ersten Jobs

Um die Ergebnisse des ersten Jobs zu überprüfen, können Sie den Inhalt des Ausgabeverzeichnisses listen und anzeigen:

```
hdfs dfs -ls /user/hdoop/intermediate-output
hdfs dfs -cat /user/hdoop/intermediate-output/part-r-00000
```

Diese Befehle zeigen die Dateien im Ausgabeverzeichnis an und geben den Inhalt der Datei `part-r-00000` aus, die das Ergebnis des Jobs enthält.

4.4.1 Alternative Überprüfung im HDFS Web Interface

Eine alternative Möglichkeit, die Ergebnisse zu überprüfen, bietet das HDFS Web Interface:

1. Klicken Sie oben auf den Reiter `Utilities`.
2. Wählen Sie `Browse the File System`.
3. Navigieren Sie zum gewünschten Verzeichnis, z.B. `/user/hdoop/intermediate-output` für den ersten Job.

Die folgende Abbildung zeigt, wie das Ausgabeverzeichnis im HDFS Web Interface aussieht:

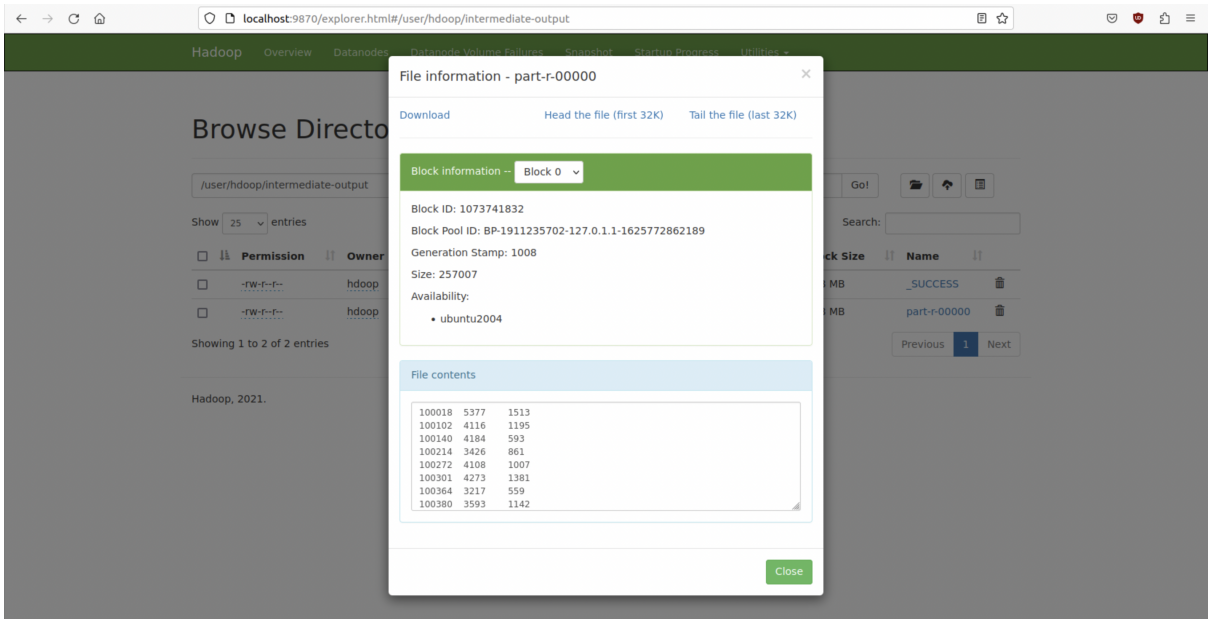

Figure 3: Überprüfung der Ausgabe im HDFS Web Interface

5 Job 2: Gruppierung der Kundenpunkte

Dieser MapReduce-Job klassifiziert Kunden basierend auf ihren akkumulierten Punkten in drei Gruppen und berechnet wichtige statistische Informationen für jede Gruppe. Die Eingabedaten für diesen Job sind die Ergebnisse des ersten Jobs.

5.1 Code-Überblick und Funktion

Der Job implementiert benutzerdefinierte Klassen und Methoden, um die Kundendaten effektiv zu verarbeiten und zu gruppieren.

Die `PointsTuple`-Klasse

- Diese Klasse ist ein benutzerdefinierter `Writable`-Typ, der für die Speicherung und Übertragung von Punktedaten zwischen den Map- und Reduce-Phasen verwendet wird.
- Sie speichert zwei Attribute: `accumulatedPoints` für die Summe der erworbenen Punkte und `redeemedPoints` für die Summe der eingelösten Punkte, beide vom Typ `int`.
- Methoden `write(DataOutput out)` und `readFields(DataInput in)` dienen der Serialisierung und Deserialisierung der Daten.

`CustomerGroupMapper`

- Der Mapper liest die Zwischenergebnisse des ersten Jobs und ordnet jeden Datensatz einer spezifischen Gruppe zu, basierend auf der Anzahl der akkumulierten Punkte.
- Die Zuordnung zu einer der drei Gruppen erfolgt durch die `getGroupId`-Methode, die basierend auf der Punktzahl einen Gruppenidentifikator zurückgibt:
  - Bis zu 2000 Punkte: `"Gruppe1_Bis2000Punkte"`.
  - Zwischen 2001 und 10000 Punkten: `"Gruppe2_2001Bis10000Punkte"`.
  - Über 10000 Punkte: `"Gruppe3_Ueber10000Punkte"`.

- Für jeden Kunden erzeugt der Mapper ein Schlüssel-Wert-Paar, bestehend aus dem Gruppenidentifikator und einem `PointsTuple`-Objekt.

`CustomerGroupReducer`

- Der Reducer fasst die Daten für jede Gruppe zusammen, indem er die Anzahl der Kunden, die Gesamtsumme der erworbenen und eingelösten Punkte zählt und den Prozentsatz der eingelösten Punkte berechnet.
- Das Ergebnis wird für jede Gruppe als Text zusammengefasst und beinhaltet die Anzahl der Kunden, die Summe der Punkte und den Prozentsatz der eingelösten Punkte.

Job-Konfiguration und -Ausführung

- Der Job wird durch die Definition der Mapper- und Reducer-Klassen, der Ein- und Ausgabeformate sowie der Pfade für die Ein- und Ausgabedaten im HDFS konfiguriert.
- Die Ausführung erfolgt durch den Aufruf der `main`-Methode, welche die Job-Konfiguration initialisiert und den Job auf dem Hadoop-Cluster ausführt. Die Eingabedaten für diesen Job sind die Ausgabe des ersten Jobs, die im Pfad `"intermediate-output"` gespeichert sind. Die Ergebnisse werden im Pfad `"final-output"` abgelegt.

Diese Struktur ermöglicht eine effektive Segmentierung der Kunden basierend auf ihrem Punktestand und bietet Einblicke in das Verhalten verschiedener Kundengruppen hinsichtlich ihrer Punktenutzung.

5.2 Kompilieren und Packen der Java-Datei für Job 2 mit Maven

Genau wie beim ersten Job muss die Java-Datei für den zweiten Job kompiliert und in eine ausführbare JAR-Datei verpackt werden.3

5.3 Ausführen des zweiten MapReduce-Jobs

Mit der kompilierten JAR-Datei kann der zweite Job ausgeführt werden. Dieser Job liest die Zwischenergebnisse des ersten Jobs, teilt die Kunden basierend auf ihren erworbenen Punkten in Gruppen und berechnet zusammenfassende Statistiken für jede Gruppe.

```
hadoop jar <pfad-zur-jar-datei> <Main-Klasse> <Eingabeverzeichnis> <
    ↪ Ausgabeverzeichnis>
```

In meinem Fall:

```
hadoop jar /home/student/PERSISTENT/Bigdata_abgabe/Hadoop_Abgabe/target/
    ↪ portofolioBigData-1.jar CustomerGroupPointsJob intermediate-output final
    ↪ -output
```

Ersetzen Sie `<pfad-zur-jar-datei>` mit dem tatsächlichen Pfad zu Ihrer JAR-Datei. Dieser Befehl nutzt die Daten im Verzeichnis `/user/hdoop/intermediate-output`, die vom ersten Job generiert wurden, und schreibt die Ergebnisse in das Verzeichnis `/user/hdoop/final-output`.

5.4 Überprüfung der Ergebnisse des zweiten Jobs

Nach Abschluss des Jobs können Sie die Ergebnisse überprüfen, um die Gruppierung der Kunden und die zugehörigen Statistiken einzusehen. Verwenden Sie folgende Befehle, um die Dateien im Ausgabeverzeichnis zu listen und die Ergebnisse anzuzeigen:

```
hdfs dfs -ls /user/hdoop/final-output
hdfs dfs -cat /user/hdoop/final-output/part-r-00000
```

Diese Befehle zeigen Ihnen, wie Kunden basierend auf der Anzahl der gesammelten Punkte in Gruppen unterteilt und die entsprechenden Statistiken für jede Gruppe berechnet wurden.

5.4.1 Alternative Überprüfung im HDFS Web Interface

Für eine benutzerfreundlichere Überprüfung der Ergebnisse können Sie das HDFS-Webinterface nutzen:

1. Gehen Sie zum Reiter `Utilities`.
2. Wählen Sie `Browse the File System`.
3. Navigieren Sie zum Verzeichnis `/user/hdoop/final-output`, um die Ergebnisse des zweiten Jobs zu sehen.

Die Ergebnisse liefern Einsichten in die Verteilung der Kundenpunkte und ermöglichen weitere Analysen bezüglich des Kundenverhaltens.

Die folgende Abbildung zeigt, wie das Ausgabeverzeichnis im HDFS Web Interface aussieht:

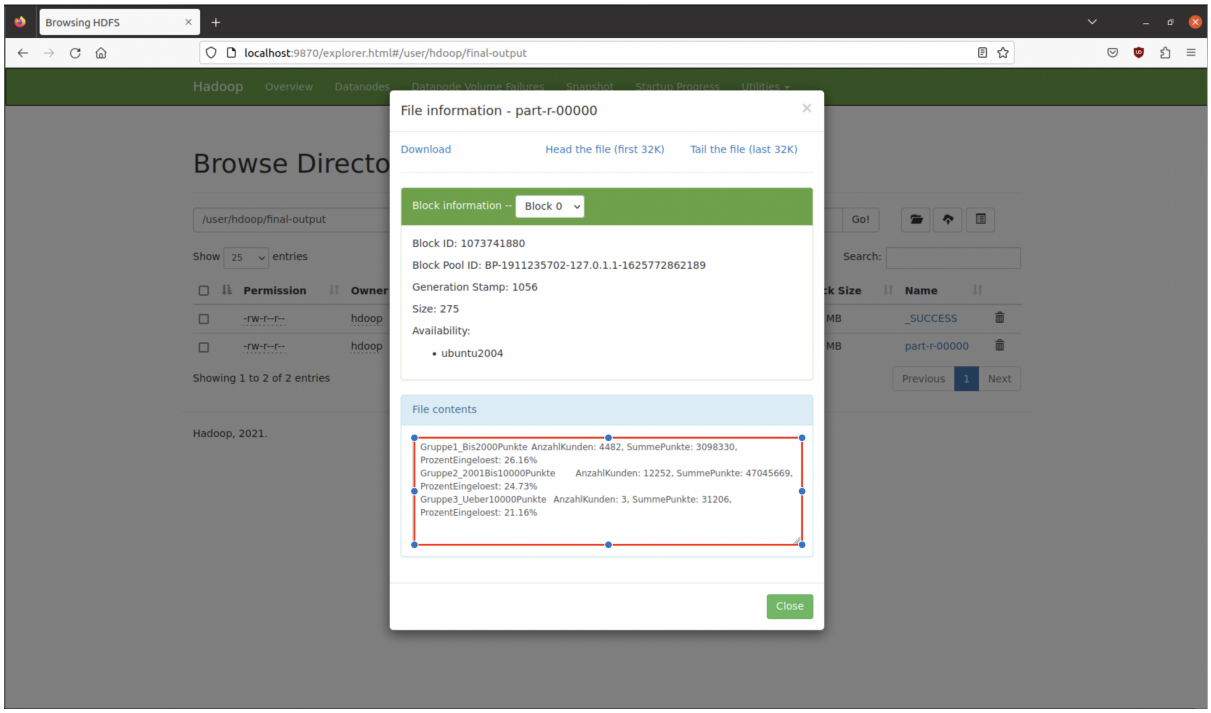

Figure 4: Überprüfung der Ausgabe im HDFS Web Interface

5.5 Bereinigung des HDFS

Nach Abschluss der MapReduce-Jobs und Analyse der Ergebnisse kann es notwendig sein, das Ausgabeverzeichnis im HDFS zu löschen, um Speicherplatz freizugeben oder um die Umgebung für zukünftige Jobs vorzubereiten. Dieser Abschnitt beschreibt, wie Sie das Endverzeichnis sicher aus dem HDFS entfernen können.

Löschen des Ausgabeverzeichnisses

Um das Endverzeichnis, welches die Ergebnisse der MapReduce-Jobs enthält, zu löschen, verwenden Sie den folgenden Befehl in der Hadoop-Befehlszeile:

```
hdfs dfs -rm -r /pfad/zum/ausgabeverzeichnis
```

Ersetzen Sie `/pfad/zum/ausgabeverzeichnis` mit dem tatsächlichen Pfad des Verzeichnisses, das Sie entfernen möchten. Zum Beispiel, um das Verzeichnis `final-output` zu löschen, das die Ergebnisse des zweiten Jobs enthält, führen Sie aus:

```
hdfs dfs -rm -r /user/hdoop/final-output
```

Achtung: Dieser Befehl löscht das Verzeichnis und seinen gesamten Inhalt unwiderruflich. Stellen Sie sicher, dass Sie alle notwendigen Daten gesichert haben, bevor Sie den Löschvorgang durchführen.

Diese Bereinigung hilft, das Hadoop-Dateisystem übersichtlich und verwaltbar zu halten und bereitet die Umgebung für zukünftige Datenverarbeitungsprojekte vor.

6 Quellen

- Für eine Einführung und ein grundlegendes Verständnis von Hadoop MapReduce, einschließlich der Erklärung des MapReduce-Programmiermodells und seiner Anwendung in Hadoop für die Verarbeitung großer Datenmengen. `https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html`
- Zur Vertiefung der Funktionsweise von MapReduce und Beispielen für MapReduce-Jobs, die zeigen, wie Daten analysiert und verarbeitet werden können. `https://www.ibm.com/cloud/learn/mapreduce`
- Diese Quelle bietet einen Überblick über die Einsatzmöglichkeiten von Hadoop im Big-Data-Umfeld und beschreibt, wie MapReduce zur Lösung komplexer Datenanalyse-Aufgaben beiträgt. `https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm`