Batch Processing - Cheat Sheet

Einführung in Batch Processing

Batch Processing umfasst die Ausführung von Datenverarbeitungsaufgaben, die große Datenmengen in Batches sog. Stapeln bearbeiten. Diese Methode ist besonders geeignet für Prozesse, die nicht in Echtzeit ablaufen müssen, jedoch hochvolumige oder komplexe Datenmengen handhaben.

Definition:
Batch Processing bezieht sich auf das Sammeln von Daten in Gruppen oder Batches für die Verarbeitung. Ein typischer Batch-Prozess durchläuft mehrere Phasen – Sammeln, Verarbeiten, Ausgeben.

Beispiel für Batch-Verarbeitung:

- Finanzbuchhaltung: Automatische Durchführung von Buchungen am Monats- oder Quartalsende.
- Telekommunikationsindustrie: Verarbeitung von Anrufdatensätzen (CDRs) zur Rechnungserstellung am Ende eines Abrechnungszyklus.

Vorteile von Batch Processing:

- **Effizienzsteigerung:**
  Optimale Nutzung der Rechnerkapazitäten in Zeiten geringerer Nachfrage (z.B. nachts).
- **Skalierbarkeit:**
  Einfache Skalierung der Datenverarbeitung mit zunehmender Datenmenge.
- **Kostenreduktion:**
  Minimierung der Kosten durch Automatisierung und Optimierung der Ressourcennutzung.

Anwendungsgebiete:
Finanzbuchhaltung, ETL-Prozesse

Einführung in die Linux/Unix Shell

Die Shell in Unix/Linux-Systemen ist eine Befehlszeilen-Schnittstelle (CLI), die als mächtiges Werkzeug zur Verwaltung von Systemressourcen, zur Dateimanipulation, zur Prozesssteuerung und zur Automatisierung von Aufgaben genutzt wird.

Grundlegende Shell-Befehle:
Die Shell ermöglicht Interaktionen mit dem Betriebssystem über einfache Textbefehle.

- **ls:** Listet Dateien und Verzeichnisse auf.
- **cd:** Wechselt das Verzeichnis.
- **mkdir:** Erstellt ein Verzeichnis.
- **rm:** Entfernt Dateien oder Verzeichnisse.
- **cp:** Kopiert Dateien oder Verzeichnisse.
- **mv:** Verschiebt oder benennt Dateien oder Verzeichnisse um.

Navigieren und Verwalten von Dateien:
Effektives Navigieren und Verwalten von Dateisystemen ist eine grundlegende Fähigkeit für jeden Unix/Linux-Nutzer.

```
cd Dokumente
ls
mkdir neues_verzeichnis
rm alte_datei.txt
```

Bedeutung von Shell-Skripting:
Shell-Skripting kombiniert mehrere Shell-Befehle in einer Datei zur automatischen Ausführung und erhöht die Effizienz bei wiederkehrenden Aufgaben.

```
#!/bin/bash
echo "Starte die Aufgabe"
cp /pfad/zum/quellverzeichnis/* /pfad/
    zum/zielverzeichnis/
echo "Aufgabe abgeschlossen"
```

Batch Processing mit UNIX-Tools

Einführung in UNIX-Tools für Batch-Processing:

- **Cron:**
  Cron ist ein Planer, der automatisch Aufgaben zu festgelegten Zeiten ausführt. Du kannst es benutzen, um Dinge wie nächtliche Datensicherungen ohne dein Zutun durchzuführen.
- **Awk:**
  Awk ist ein Tool zur Datenverarbeitung, das hilft, Textdateien Zeile für Zeile zu analysieren und zu bearbeiten, etwa um Berichte zu erstellen.
- **Sed:**
  Sed ist ein Text-Editor, der direkt in der Kommandozeile funktioniert. Er ist nützlich, um schnelle Änderungen an Textdateien vorzunehmen, wie zum Beispiel das Ersetzen von Wörtern.
- **Grep:**
  Grep ist ein Suchwerkzeug, das dir hilft, spezielle Zeilen in Textdateien zu finden, die bestimmte Wörter oder Phrasen enthalten.

Automatisierung mit Cron:
Cron hilft dir, Aufgaben zu einem festgelegten Zeitplan auszuführen, was nützlich ist, um wiederkehrende Jobs zu planen.

Einfaches Beispiel für einen Cron-Job:

```
# Oeffnen der Crontab-Datei zur
   Bearbeitung:
crontab -e

# Ein Beispiel fuer einen Cron-Job,
   der taeglich um Mitternacht
   ausgefuehrt wird:
0 0 * * * /pfad/zum/script.sh
```

Datenverarbeitung mit Awk:
Awk kann genutzt werden, um schnell Informationen aus Daten zu extrahieren.

```
# Hier ein einfacher Befehl, um die
   erste Spalte jeder Zeile in einer
   Datei auszugeben:
awk '{print $1}' datei.txt
```

Textmanipulation mit Sed:
Mit Sed kannst du Text in Dateien einfach ändern.

```
# Ersetzt 'alt' mit 'neu' ueberall in
   'datei.txt':
sed 's/alt/neu/g' datei.txt
```

Suchoperationen mit Grep:
Grep hilft dir, schnell Zeilen zu finden, die bestimmte Muster enthalten.

```
# Zeigt Zeilen in 'datei.txt', die '
   Muster' enthalten:
grep 'Muster' datei.txt
```

MapReduce

MapReduce ist ein Verfahren/Programmiermodell, das die Verarbeitung umfangreicher Datenmengen über ein Netzwerk verteilter Computer ermöglicht. Dieses Modell eignet sich hervorragend für Batch Processing, indem es komplexe Verarbeitungsoperationen in kleinere, unabhängige Einheiten unterteilt, die simultan auf verschiedenen Systemen ausgeführt werden können. Diese Methode ist besonders effektiv für Szenarien, in denen große Datenbestände in einem durchgehenden Prozess ohne direkte Interaktion des Endbenutzers bearbeitet werden.

MapReduce Konzept
MapReduce organisiert die Datenverarbeitung durch zwei sequentielle Phasen – ***Map*** und ***Reduce***:

Batch Processing - Cheat Sheet

- **Map-Phase:** Hier werden die Eingabedaten in Schlüssel-Wert-Paare umgewandelt. Jeder Eingabedatensatz wird unabhängig verarbeitet, was eine hohe Parallelisierung ermöglicht.
- **Reduce-Phase:** Nach der Map-Phase werden alle Zwischenschlüssel-Wert-Paare, die denselben Schlüssel haben, gruppiert und an die Reduce-Funktion weitergeleitet, die sie zu einem einzelnen Ausgabewert zusammenfasst.

Einfaches Python-Beispiel für MapReduce:
In diesem Beispiel wird die Häufigkeit jedes Wortes in einem Textdokument gezählt:

```python
from mrjob.job import MRJob

class MRWordFrequencyCount(MRJob):
  def mapper(self, _, line):
    # Split the line into words
    words = line.split()
    # Emit each word with a count of 1
    for word in words:
      yield word, 1

  def reducer(self, key, values):
    # Sum all the counts for each word
    total = sum(values)
    yield key, total

if __name__ == '__main__':
  MRWordFrequencyCount.run()
```

Hadoop Distributed File System (HDFS)

Das Hadoop Distributed File System (HDFS) ist ein System, das entwickelt wurde, um sehr große Mengen von Daten über viele Computer in einem Netzwerk zu verteilen und zu speichern. Es ist ein wesentlicher Bestandteil von Hadoop, einer Software, die große Datenmengen verarbeiten kann.

Was macht HDFS besonders?
- **Skalierbarkeit:** HDFS kann sehr groß werden, indem es mehr Computer in das Netzwerk einbindet, sodass es mit wachsenden Datenmengen mithalten kann.
- **Fehlertoleranz:** Es kopiert Daten automatisch auf mehrere Computer, so dass keine Daten verloren gehen, selbst wenn einige Computer ausfallen.
- **Hohe Verfügbarkeit:** Die Daten sind jederzeit verfügbar, auch wenn Teile des Systems ausfallen, dank der Kopien auf verschiedenen Computern.
- **Kosteneffizienz:** Es kann auf normaler, nicht spezialisierter Hardware betrieben werden, was Kosten spart.

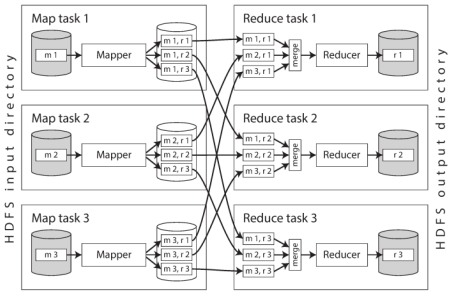
Wie funktioniert HDFS?
- **NameNode:** Dies ist der Hauptcomputer, der überwacht, wo alle Datenstücke im Netzwerk gespeichert sind.
- **DataNode:** Diese Computer speichern und verwalten die eigentlichen Daten. Sie nehmen Anweisungen vom NameNode entgegen, um Daten zu lesen, zu schreiben und zu löschen.

Zusammenarbeit mit MapReduce:
HDFS arbeitet eng mit MapReduce zusammen, einem weiteren Tool von Hadoop, das dazu dient, die Daten zu analysieren und aufzubereiten. Zusammen ermöglichen sie es, komplexe Analysen mit großen Datenmengen effizient durchzuführen.

Anwendungen von HDFS:
HDFS wird in vielen verschiedenen Bereichen eingesetzt, darunter in der Web-Indexierung, in sozialen Netzwerken und in der Datenanalyse, wo große Mengen von Daten gespeichert und schnell verarbeitet werden müssen.

Prozessveranschaulichung

Das folgende Diagramm veranschaulicht den typischen Ablauf eines MapReduce-Jobs innerhalb eines Hadoop Distributed File System (HDFS). Es zeigt die parallele Natur der Map- und Reduce-Aufgaben sowie den Datenfluss zwischen diesen Komponenten.



Beschreibung des Diagramms:

- **Map Tasks:** Jede Map Task verarbeitet einen Teil der Eingabedaten unabhängig von den anderen. Beispielsweise liest Map Task 1 Daten aus dem ersten Block des HDFS-Inputverzeichnisses.
- **Reducer Tasks:** Jede Reduce Task sammelt und verarbeitet die Zwischenergebnisse, die von den Map Tasks erzeugt wurden, um das Endresultat zu erstellen. Reducer 1 beispielsweise aggregiert Ergebnisse, die den gleichen Schlüssel (z.B. r1) teilen.

Nachteile von Batch Processing / MapReduce und HDFS

Nachteile von Batch Processing:
Batch Processing, obwohl effizient für große Datenmengen, hat einige Einschränkungen:

- **Zeitverzögerungen:** Da Prozesse in Batches ausgeführt werden, kann es zu Verzögerungen kommen, bevor Ergebnisse verfügbar sind.
- **Ressourcenintensiv:** Große Datenmengen erfordern signifikante Rechenressourcen, was insbesondere bei begrenzter Infrastruktur zu Problemen führen kann.
- **Komplexität:** Die Verwaltung und Überwachung von Batch Jobs kann komplex sein, insbesondere wenn Fehlerbehandlung und Wiederanlauf von Prozessen notwendig werden.

Nachteile von MapReduce und HDFS:

- **Skalierbarkeitsgrenzen:** Obwohl Hadoop hervorragend skalierbar ist, kann die Leistung bei extrem großen Datenmengen oder sehr schnellen Datenströmen nachlassen.
- **Komplexität der Verwaltung:** HDFS erfordert eine aufmerksame Verwaltung der DataNodes und NameNodes, was die Systemkomplexität erhöht.
- **Langsamere Verarbeitung:** MapReduce ist nicht optimal für Prozesse, die Echtzeitanalysen benötigen, da die Verarbeitung von Daten in Batches erfolgt.

Batch Processing vs. Stream Processing