

Visual Analytics Milestone 2

Robin Ellerkmann
Jan-Christopher Pien
& Andreas Wegge

Aufgabe 1

Implementierung Datentyp – Tupel

- Gründe für die Nutzung von Tupeln:
 - Saubere Trennung der einzelnen Messpunkte
 - Einfacher Zugriff
 - Gründe für die ausgewählte Implementierung:
 - Entwicklung einer DataPoint-Klasse unter Nutzung des “Tuple”-Klassenbaums von Flink
 - Problematik von Tupeln in Java
 - Optionals innerhalb des Tupels
-

Aufgabe 2

Funktion `read_data` implementieren

- mittels Java `BufferedReader`
 - --> while loop mit parsing
 - Einlesen als `LinkedList` und dann weitergabe als Java 8 Stream
-

Aufgabe 3

Selection, Projection & Aggregation implementieren

- Interface DataSource
- Implementierung StreamDataSource kann:
 - Datei einlesen oder zufällige Daten generieren
 - Den Stream “persistent” als List speichern.
 - Die ersten N Ergebnisse einer Operation ausgeben
 - Die geforderten Operationen selection, projection und aggregation mit Komparatoren (atLeast, Same, LessThan) und Aggregatoren (avg, max, min)

OVERVIEW PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

de.hu.flinkydata.data

Interface DataSource<T>

All Known Implementing Classes:
StreamDataSource

```
public interface DataSource<T>
```

Basis-Interface für die drei Operationen, die auf einer Data Source möglich sein sollen. Created by Jan-Christopher on 09.11.2016.

Method Summary

All Methods	Instance Methods	Abstract Methods	Default Methods
Modifier and Type			
Method and Description			
default	DataSource<T>	aggregation(AggregatorFunction<T> aggregator)	Aggregiert alle Datensätze mit der angegebenen Aggregationsfunktion.
default	DataSource<T>	aggregation(AggregatorFunction<T> aggregator, int count)	Aggregiert die gewünschte Anzahl Datensätze mit der angegebenen Aggregationsfunktion.
	List<T>	collect()	Gibt alle Datensätze dieser DataSource als Collection aus.
	DataSource<T>	firstN(int count)	Gibt die ersten n Datensätze dieser DataSource zurück.
	void	print()	Gibt die Elemente der DataSource auf standard out aus.
<R>	DataSource<R>	projection(Function<? super T,? extends R> projector)	Projiziert die Datensätze in dieser Datenquelle auf einen neuen Datentyp.
	DataSource<T>	reduce(T identity, BinaryOperator<T> reducer)	Reduziert die Datensätze in dieser Datenquelle auf einen einzigen Datensatz.
	DataSource<T>	selection(Predicate<? super T> predicate)	Wählt Datensätze aus der Datenquelle aus, die ein gegebenes Prädikat erfüllen.

Aufgabe 3

Spezifikation von Selection, Projection & Aggregation

- Selection (Filter)

$$R \times f \rightarrow [\{r | f(r) = \text{true}\}]$$

- Projection (Map)

$$R \times f \rightarrow [f(r_1), f(r_2), \dots, f(r_n)]$$

- Aggregation (Reduce)

- Für einfache Aggregationen (Min, Max, Sum, etc.)

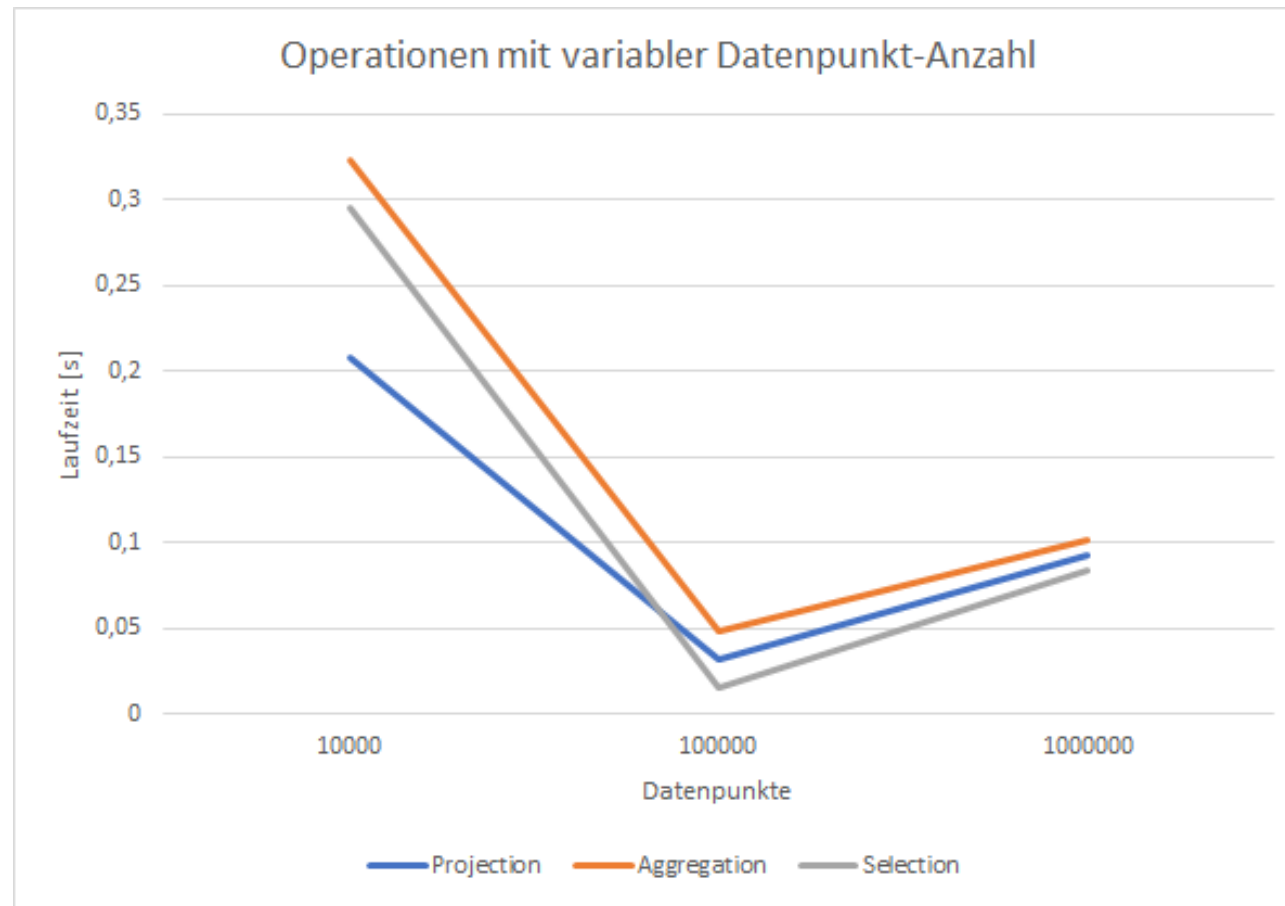
$$R \times f \times I \rightarrow f\left(r_n, f\left(r_{n-1}, f\left(\dots, f(r_1, I)\right)\right)\right)$$

- Für kompliziertere Aggregationen (Avg) müssen mehrere Map-Reduce-Schritte hintereinandergeschaltet werden
 - Implementierung Average Aggregation:

```
Average(Stream s, Feld f):  
    s.projection(tuple -> (tuple, 1))  
    s.reduce((tuple1, tuple2) -> (tuple1[1][f] + tuple2[1][f], tuple1[2] + tuple2[2]))  
    s.projection(tuple -> tuple[1]; tuple[1][f] = tuple[1][f] / tuple[2])
```

Aufgabe 4

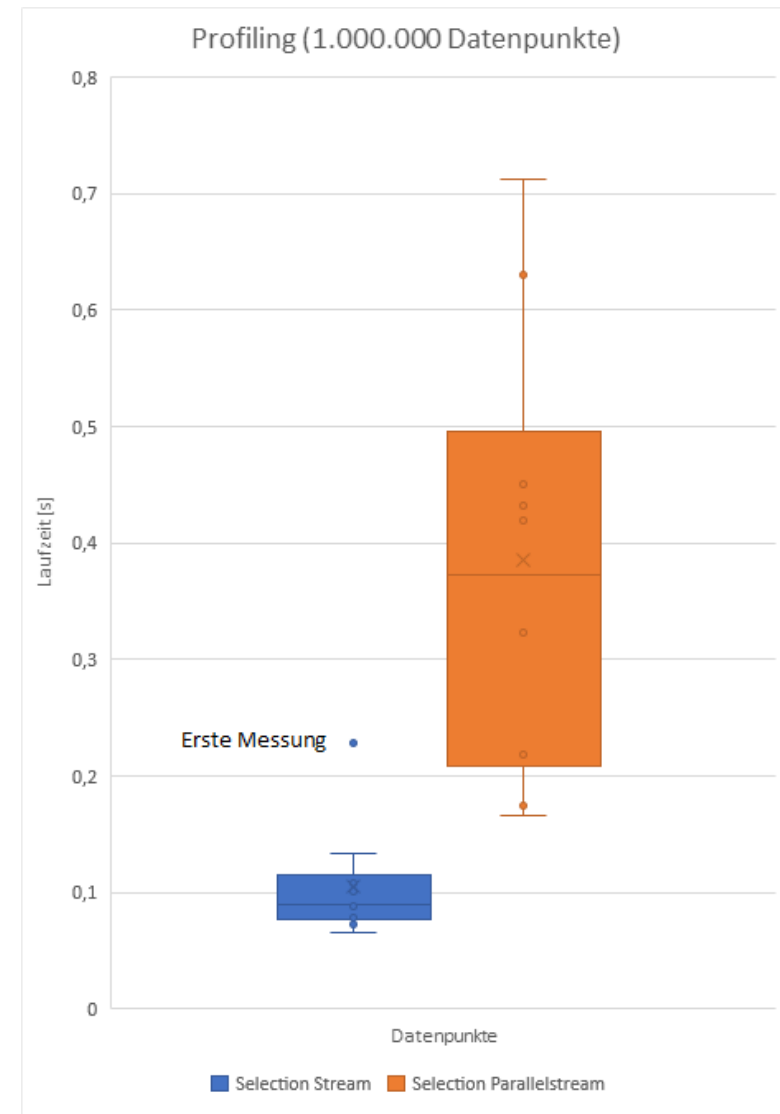
Benchmarking durch “Profile”-Funktion



Aufgabe 5

Skalierbarkeit prüfen

- 1.000.000 Datenpunkte
- Ausreißer bei erster Messung
 - Vermutlich wegen Speicherallokation
- Testweise Implementierung mit parallelen Streams
 - Langsamer, da Overhead für das Synchronisieren von Threads nötig



Aufgabe 6

Aufgabe: Zeige, dass ein Datentyp mit den Operatoren σ (Selektion), π (Projektion), γ (Aggregation) und \times (Kartesisches Produkt) die folgenden Tasks unterstützt...

1. Identifizieren:

$A = \{a_1, \dots, a_n\}$ sei eine Menge von n Objekten. Identifiziere Objekte in A die eine bestimmte Bedingung erfüllen.

Definition σ : Seien D_1, \dots, D_n Domänen und sei $R \subseteq D_1 \times \dots \times D_n$ mit $R\{A_1 : D_1, \dots, A_n : D_n\}$ eine n -stellige Relation auf diesen Domänen. Sei c eine Selektionsbedingung, d. h. ein Boolescher Ausdruck aus Attributen (A_1, \dots, A_n) , Operatoren $(=, \neq, \geq, \leq, <, >)$ und logischen Junktoren (\wedge, \vee) . Dann ist die Selektion wie folgt definiert:

$$\sigma_c(R) := \{\mu : (c[\mu] = \text{true}) \wedge (\mu \in R)\}$$

wobei μ die Tupel der Relation sind.

Die Datenstruktur enthält eine Menge A von Objekten. Die Objekte sind gleichförmige Elemente der Extension einer Relation, d.h. jedes Objekt ist ein Tupel einer bestimmten Relation R . Die Datenstruktur unterstützt weiterhin den Operator Selektion. Die Selektion σ ist äquivalent zu dem Task "Identifizieren", soweit sich die geforderten Bedingungen als Boolescher Ausdruck beschreiben lassen. Somit lässt sich der Task "identifizieren" durch den Operator σ realisieren.

2. Vergleichen:

$A = \{a_1, \dots, a_n\}$ sei eine Menge von n Objekten und $C^k = A \times_1 \dots \times_k A$ eine beliebige Relation. Vergleiche Objekte $\{a_1, \dots, a_k\}$ in A um geordnete Paare $a_{\pi(1)} C a_{\pi(2)} C \dots C a_{\pi(k-1)} C a_{\pi(k)}$ zu erkennen die C^k erfüllen (π ist eine valide Permutation der Indizes).

3. Merkmale erkennen:

$A = \{a_1, \dots, a_n\}$ sei eine Menge von n Objekten und F_l eine Familie von Funktionen. Erkenne alle Untermengen $\{a_1, \dots, a_k\}$, die eine Funktion $F \in F_l$ zu true auswerten.

Aufgabenaufteilung

- Jan: Einrichten des Projekts, der Interfaces und der meisten Klassen (Komparatoren, Aggregatoren und DataSource) für Flink (1. Implementation) und dann auf Basis von Java8 Streams (2. Implementation – in Nutzung)
 - Robin: Entwicklung des Profiling/Benchmarking und der zufälligen Datengenerierung, Testen und Messen, Verfeinerung der Implementierung, Präsentation und Dokumentation
 - Andreas: Schreiben der Projektion und des avg-Aggregators, Verfeinerung der Implementierung, Beweisskizze (unfertig), Präsentation und Dokumentation
-