Lernzettel

Data-Streams und Stream-Processing: Fenstertechnik, Zustand-Verwaltung, Konsistenz

Universität: Technische Universität Berlin
Kurs/Modul: Informationssysteme und Datenanalyse
Erstellungsdatum: September 19, 2025

Lernzettel: Data-Streams und Stream-Processing: Fenstertechnik, Zustand-Verwaltung, Konsistenz

(1) Grundbegriffe und Ziele. Ein Data-Stream ist eine unendliche, zeitlich sortierte Folge von Ereignissen. Jedes Ereignis trägt typischerweise ein Zeitstempel-Attribut und sonstige Nutzdaten. Ziel des Stream-Processing ist es, kontinuierlich oder fast-continuously Ergebnisse zu erzeugen, z. B. Aggregationen, Mustererkennung oder Verknüpfungen mit Speichersystemen.

(2) Fenstertechnik. Fenster definieren, welche Ereignisse in eine bestimmte Verarbeitungseinheit fallen. Typische Fensterarten:

- Tumbling Window (W) – disjunkte, sich nicht überschneidende Fenster der Dauer W: $[t, t + W)$.
- Sliding Window (W, S) – überlappende Fenster der Breite W mit Schrittweite S: $[t, t+W)$, dann $[t + S, t + S + W)$, ...
- Session Window – basierend auf Aktivitätspausen; Fenster endet nach einer Inaktivitätszeit.

Für jedes Fenster berechnen wir typischerweise eine Aggregation, z. B. die Summe, den Durchschnitt oder die Maximalwerte der im Fenster enthaltenen Werte:

$$\text{Summe}_{[a,b)} = \sum_{i:t_i \in [a,b)} v_i, \quad \text{Durchschnitt}_{[a,b)} = \frac{1}{n} \sum_i v_i.$$

(3) Zustand-Verwaltung (State) in Streaming-Operatoren. Zustand erlaubt Operatoren, sich über aufeinander folgende Datenströme hinweg zu merken:

- Keyed State: pro Schlüssel k eine lokale Zustands-Instanz.
- Operator State: zustände, die Operator-Instanzen betreffen, unabhängig von Schlüsselwerten.

Wichtige Konzepte:

- Checkpoints und Wiederherstellung: bei Ausfall wird der Zustand aus dem letzten konsistenten Snapshot wiederhergestellt.
- Backups, Snapshots, Streams als Quelle und Senke des Zustands.

(4) Konsistenz-Modelle in Stream-Verarbeitung. Gängige Semantiken und deren Konsequenzen:

- At-least-once: Events können mehrfach verarbeitet werden; Reduktion über Idempotenz.
- At-most-once: kein Mehrfachverarbeiten, aber potenzielle Verlustfälle.
- Exactly-once: jedes Event wird genau einmal verarbeitet, oft mittels Transaktions-Logik über Zustand und Out-of-Order-Schutz.

Wichtige Mechanismen:

- Idempotente Operatoren
- Persistente Logs und Semantiken zur Konsistenz
- Watermarks und late data handling

(5) Konsistenz in Fensterung und Late Data. Herausforderungen:

- Out-of-Order-Ereignisse: erst später eintreffende Events beeinflussen laufende Fenster.
- Late Data: nachträgliche Daten können bereits abgeschlossene Fenster korrigieren.

Maßnahmen:

- Allowed Late Arrival Time
- Trigger-Strategien (Time-based, Count-based)
- Watermarks: progressiver Fortschritt der Zeit, der optimistisch Fenster schließt

Beispiel: Für eine Fenstergröße $W = 10\,\text{s}$, Allowed-Lateness=5s; ein Event mit Zeitstempel t wird bis $t + 5$ im Fenster berücksichtigt, danach wird das Fenster finalisiert.

(6) Architekturüberblick: Streaming vs. relationale Modelle. Streaming-Systeme koordinieren kontinuierliche Eingaben, speichern Zwischenzustände persistent und liefern Ergebnisse kontinuierlich. Im Gegensatz dazu arbeiten relationale Datenbanken mit Transaktionen und stabilen Sichten.

Verknüpfungsmuster:

- Streaming-Quellen (Klicks, Sensoren) → Stream-Prozessoren → State Stores → Speichersysteme
- Stream-Processing als Vorverarbeitung für Data-Warehouse-ETL oder konsistente Sicht auf Rohdaten

(7) Typische Algorithmen und Operatoren im Data-Stream-Processing.

- Fensterbasierte Aggregationen: Summe, Durchschnitt, Min/Max, Count
- Zeitbasierte Joins zwischen Streams oder zwischen Stream und gespeicherten Tabellen (Windowed Joins)
- Pattern-Detection: Sliding-Window-Templates, Ereignissequenzen
- Anomalieerkennung auf Streams

(8) Formeln und Beispiele. Beispiel 1 – windowed Average: Für ein Tumbling Window der Größe W berechnen wir den Mittelwert der Werte $\{v_i\}$ innerhalb des Fensters:

$$\text{avg}_{[t,t+W)} = \frac{1}{n} \sum_{i:t_i \in [t,t+W)} v_i$$

Beispiel 2 – Exactly-Once Semantik durch idempotente Verarbeitung: Bei jeder Re-Emission eines Events wird das Ergebnis durch eine idempotente Operation so erzeugt, dass wiederholte Ausführung das Endresultat nicht ändert.

(9) Praxis-Checkliste.

- Welche Fensterart ist sinnvoll?
- Welche Konsistenz-Semantik ist akzeptabel?
- Wie wird Zustand gespeichert und wie robust ist das System gegen Ausfälle?