Apache Flink

Abstract

zum Abschluss des Moduls

Oberseminar Datenbanksysteme – Aktuelle Trends, SS18

in der Studienrichtung Informatik

Eingereicht von: Michael Kriegs
Phillip-Reis-Straße 97
04179 Leipzig

SG: IMN17
MatrNr: 70055

Leipzig, 12. Juni 2018

Inhaltsverzeichnis

1. Überblick

„Apache Flink® is an open-source stream processing framework for distributed, high-performing, always-available, and accurate data streaming applications.“[Fou14]
Apache Flink ist ein Streaming Framework was an der TU Berlin entwickelt wurden ist und seit 2014 im Apache Incubator als Top Level Projekt aufgenommen wurde. Es stellt APIs zur Verfügung für die Verarbeitung von endliche und unendlichen Daten. Die APIs bedienen die Programmiersprachen Java , Scala und Python (nur endliche Daten sogenannte Batches). Selbst ist das Framework in Java implementiert worden. Im folgenden Abbildung 1 werden typische Anwendungsfälle für diese Framework dargelegt. Im Kern des

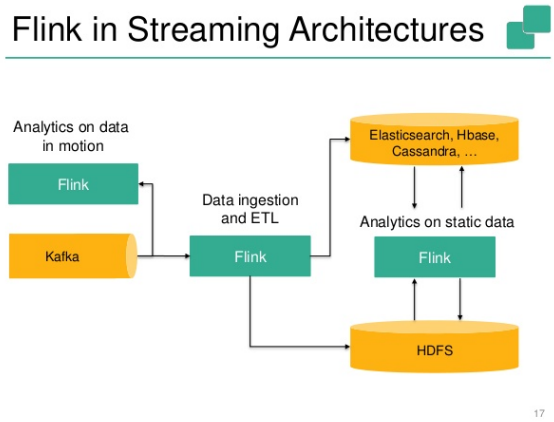

Abbildung 1: Usecase Apache Flink

Frameworks befindet sich die Runtime Engine "Destributed Streaming Dataflow"welche die Steuerung und Optimierung der Anwendung vor nimmt. Aufsetztend auf diesen Kern befinden sich die zwei Basis APIs für endliche (DataSet API [Fou18a]) und unendliche (DataStream API [Fou18b]) Daten, mehr zu den Grundsätzen in 1.1.. On Top sind die Module bzw Bibliotheken für die speziellen Anwendungen zu finden.

Apache Flink bietet zwei relationale APIs - die Table API und SQL - für die vereinheitlichte Stream- und Batch-Verarbeitung. Die Tabellen-API ist eine sprach integrierte Abfrage-API für Scala und Java, die die Zusammenstellung von Abfragen von relationalen Operatoren wie Auswahl, Filter und Join auf sehr intuitive Weise ermöglicht. Die SQL-Unterstützung von Flink basiert auf Apache Calcite, das den SQL-Standard implementiert. Abfragen, die in einer der Schnittstellen angegeben sind, haben dieselbe Semantik

und geben dasselbe Ergebnis an, unabhängig davon, ob die Eingabe eine Batch-Eingabe (DataSet) oder eine Stream-Eingabe (DataStream) ist.

Gelly ist eine Graph API für Flink. Es enthält eine Reihe von Methoden und Utilities, die die Entwicklung von Anwendungen zur Graphenanalyse in Flink vereinfachen sollen. In Gelly können Graphen mit High-Level-Funktionen transformiert und modifiziert werden, die denen ähneln, die von der Stapelverarbeitungs-API bereitgestellt werden. Gelly bietet Methoden zum Erstellen, Transformieren und Modifizieren von Graphen sowie eine Bibliothek von Graphalgorithmen.

FlinkML ist die Machine Learning (ML) Bibliothek für Flink. Es ist eine neue Bemühung in der Flink Gemeinschaft, mit einer wachsenden Liste von Algorithmen und Mitwirkenden. Mit FlinkML wollen wir skalierbare ML-Algorithmen, eine intuitive API und Werkzeuge bereitstellen, die helfen, den Klebstoff-Code in End-to-End-ML-Systemen zu minimieren. Viele Funktionen erfreuen sich einer erhöhten Beliebtheit, dadurch werden hier viele Algorithmen implementiert und durch eine Roadmap kontinuierliches Wachstum initialisiert.

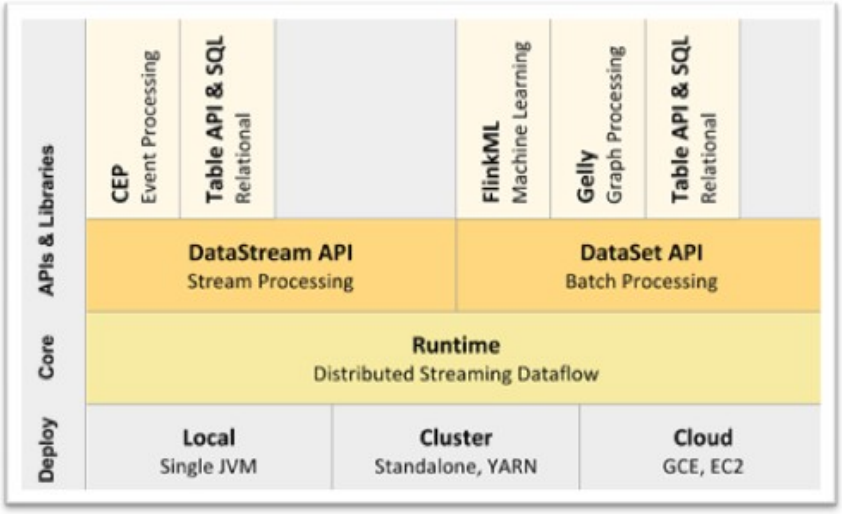

Abbildung 2: The “What”: Flink from the bottom-up

1.1. Kontinuierliche Verarbeitung für unbeschränkte Datenmengen

Bevor wir auf Flink eingehen, betrachten wir auf einer höheren Ebene die Arten von Datensätzen, die bei der Verarbeitung von Daten auftreten können, sowie Typen von Ausführungsmodellen, die Sie zur Verarbeitung auswählen können. Diese beiden Ideen sind oft miteinander verschmolzen, und es ist sinnvoll, sie klar voneinander zu trennen.

1. 2 Arten von Datensätzen

Unbegrenzt: Unendliche Datensätze, die kontinuierlich angehängt werden
Begrenzt: Endliche, unveränderliche Datensätze
Viele reale Datensätze, die traditionell als beschränkte oder „Batch"-Daten betrachtet werden, sind in der Realität unbegrenzte Datensätze. Dies gilt unabhängig davon, ob die Daten in einer Reihe von HDFS-Verzeichnissen oder in einem protokollbasierten System wie Apache Kafka gespeichert sind.

Beispiele für unbegrenzte Datensätze beschränken sind aber nicht auf:
Endbenutzer, die mit mobilen oder Webanwendungen interagieren
Physikalische Sensoren liefern Messungen
Finanzmärkte
Maschinenprotokolldaten

2. 2 Arten von Ausführungsmodellen

Streaming: Verarbeitung, die kontinuierlich ausgeführt wird, solange Daten erzeugt werden
Batch: Verarbeitung, die ausgeführt wird und in einer begrenzten Zeit zur Vollständigkeit abläuft, wodurch Rechenressourcen freigegeben werden, wenn sie beendet sind
Es ist möglich, obwohl nicht unbedingt optimal, beide Arten von Datensätzen mit beiden Arten von Ausführungsmodellen zu verarbeiten. Zum Beispiel wurde die Batch-Ausführung schon lange auf unbeschränkte Datensätze angewendet, trotz möglicher Probleme mit Windowing, Zustandsverwaltung und Out-of-Order-Daten.

Flink setzt auf ein Streaming-Ausführungsmodell, das sich intuitiv für die Verarbeitung unbegrenzter Datensätze eignet: Die Streaming-Ausführung ist eine kontinuierliche Verarbeitung von Daten, die kontinuierlich produziert werden. Und die Ausrichtung zwischen dem Typ des Datensatzes und der Art des Ausführungsmodells bietet viele Vorteile in Bezug auf Genauigkeit und Leistung.

1.2. Features

Zuvor haben wir besprochen, wie Sie den Dateityp (bounded vs. unbounded) mit dem Typ des Ausführungsmodells (Batch vs. Streaming) abstimmen. Viele der unten aufgeführten Flink-Funktionen, die Verwaltung von Out-of-Order-Daten und flexible Fensterfunktionen sind für die Berechnung genauer Ergebnisse in unbegrenzten Datensätzen unerlässlich und werden vom Streaming-Ausführungsmodell von Flink aktiviert.

1.2.1. Stateful Operation

Flink garantiert genau einmal Semantik für Stateful-Berechnungen. „Stateful“ bedeutet, dass Anwendungen eine Aggregation oder Zusammenfassung von Daten aufrechterhalten können, die im Laufe der Zeit verarbeitet wurden, und der Checkpointing-Mechanismus von Flink stellt eine genau einmalige Semantik für den Zustand einer Anwendung im Falle eines Fehlers sicher.[Fou18c]

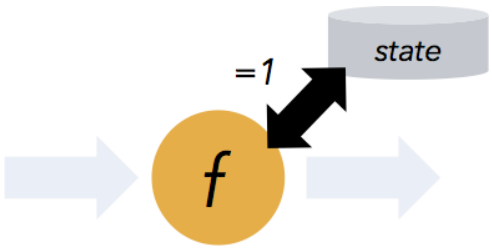

Abbildung 3: Stateful Operation

1.2.2. Semantik der Ereigniszeit

Flink unterstützt Stream-Verarbeitung und Windowing mit Ereigniszeitsemantik. Ereigniszeit macht es einfach, genaue Ergebnisse über Streams zu berechnen, bei denen Ereignisse außerhalb der Reihenfolge ankommen und Ereignisse verzögert ankommen können.

Hier kommt das Modul CEP „Complex event processing for Flink“ zum Einsatz welches von Alibaba Group entwickelt wurde und an die Community zurück gegeben wurde.[Fou18d]

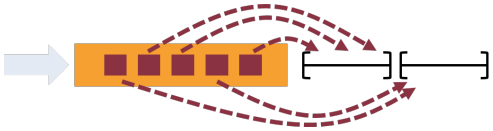

Abbildung 4: The “What”: Flink from the bottom-up

1.2.3. flexible Windowing

Flink unterstützt flexible Windowfunktionen basierend auf Zeit, Anzahl oder Sitzungen zusätzlich zu datengesteuerten Fenstern. Windows kann mit flexiblen Triggerbedingungen angepasst werden, um anspruchsvolle Streaming-Muster zu unterstützen. Flink's Windowing ermöglicht es, die Realität der Umgebung zu modellieren, in der die Daten erzeugt werden.[Fou18e]

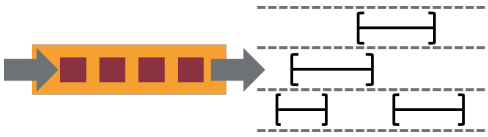

Abbildung 5: Window Feature

1.2.4. Fehlertoleranz und Savepoints

Die Fehlertoleranz von Flink ist gering und ermöglicht es dem System, hohe Durchsatzraten aufrechtzuerhalten und gleichzeitig genau einmal Konsistenzgarantien zu liefern. Flink erholt sich von Fehlern mit null Datenverlust, während der Kompromiss zwischen Zuverlässigkeit und Latenz vernachlässigbar ist. Die DataStream API unterstützt die Checkpoint Funktionalität hingegen die DataSet API nicht. Hier wird eine Fehlertoleranz mit einem erneuten Senden des Datensatzes realisiert.

Die Savepoints von Flink bieten einen State-Versions-Mechanismus, der es ermöglicht, Anwendungen zu aktualisieren oder historische Daten ohne verlorenen Zustand und minimale Ausfallzeiten erneut zu verarbeiten.[Fou18c] Savepoints werden in sogenannten state

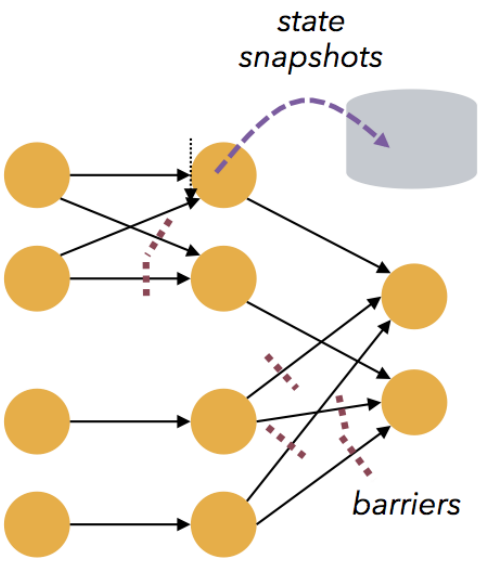

Abbildung 6: Checkpoints, Snapshot

Backends gespeichert.

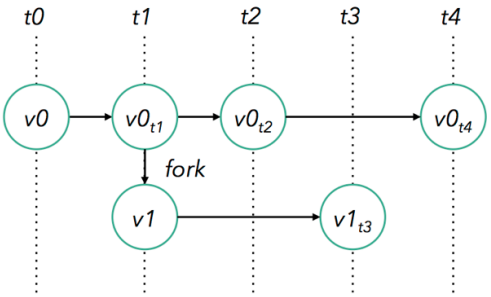

Abbildung 7: Savepoint

1.2.5. hoher Durchsatz und geringe Latenz

Flink ist in der Lage, einen hohen Durchsatz und eine niedrige Latenzzeit (schnelle Verarbeitung von vielen Daten) zu erreichen. Die folgende Abbildung 8 zeigen die Leistung von Apache Flink und Apache Spark bei der Ausführung einer Aufgabe zum Verteilen von verteilten Elementen, die Streaming-Datenmischungen erfordert. Hierzu wurden verschiedene Workloads getestet.

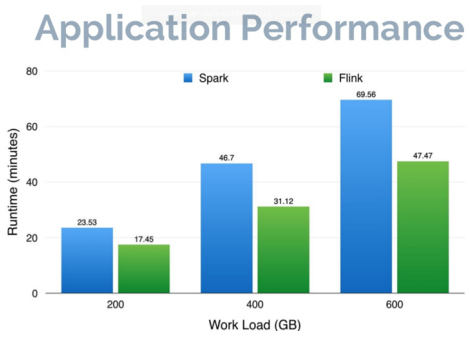

Abbildung 8: Vergleich Spark vs Flink

1.2.6. hohe Skalierbarkeit

Flink ist für den Betrieb in großen Clustern mit vielen tausend Knoten ausgelegt. Zusätzlich zu einem eigenständigen Cluster-Modus bietet Flink Unterstützung für YARN und Mesos. Durch die Core Engine, in 1. beschrieben, lassen sich die Prozesse sehr gut parallelisieren und verteilen. Die Daten fließen von Operator zu Operator als Stream. So können die Operatoren auf verschiedenen Knoten im Cluster ausgeführt und unabhängig voneinander Arbeiten.

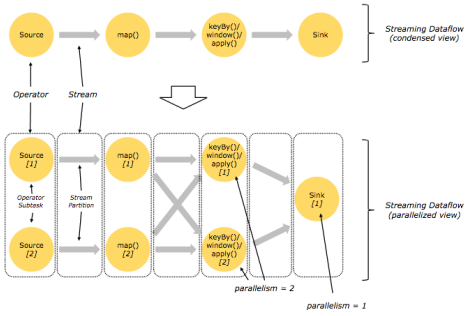

Abbildung 9: Parallelisierung des Jobgraphs

2. Anwendungsbeispiel

Uber ermöglicht nahtlose und angenehmere Benutzererfahrungen, indem es Daten aus einer Vielzahl von Echtzeitquellen kanalisiert. Diese Erkenntnisse reichen von aktuellen Verkehrsbedingungen, die eine Orientierungshilfe für Reiserouten bieten, bis zur geschätzten Zustellungszeit (ETD) einer UberEATS-Bestellung - und jeder dazwischen liegenden Metrik.

Das schiere Wachstum von Ubers Geschäft erforderte eine Datenanalyse-Infrastruktur,

die in der Lage ist, eine breite Palette von Erkenntnissen aus der ganzen Welt und zu jeder Zeit, wie z. B. stadtspezifische Marktbedingungen, zu globalen Finanzschätzungen zu streamen. Mit mehr als einer Billion Echtzeit-Nachrichten, die jeden Tag durch unsere Kafka-Infrastruktur geleitet werden, musste die Plattform (1) für alle Nutzer leicht zugänglich sein, unabhängig von technischem Fachwissen, (2) skalierbar und effizient genug, um Echtzeit-Ereignisse zu analysieren. und (3) robust genug, um hunderte, wenn nicht tausende von kritischen Jobs kontinuierlich zu unterstützen.
Wir haben AthenaX, unsere hauseigene Streaming-Analyse-Plattform, entwickelt und eröffnet, um diese Anforderungen zu erfüllen und barrierefreie Streaming-Analysen für jedermann zugänglich zu machen. AthenaX ermöglicht unseren technischen und nichttechnischen Benutzern, umfassende Streaming-Analysen in Produktionsqualität mit Structured Query Language (SQL) durchzuführen. SQL macht die Verarbeitung von Ereignisströmen einfach - SQL beschreibt, welche Daten analysiert werden sollen, und AthenaX legt fest, wie die Daten analysiert werden (z. B. durch Auffinden oder Skalieren der Berechnungen). Unsere Praxiserfahrung zeigt, dass AthenaX es Benutzern ermöglicht, große Streaming-Analyse-Workloads innerhalb von Stunden im Vergleich zu Wochen in Produktion zu bringen.[Hao17]

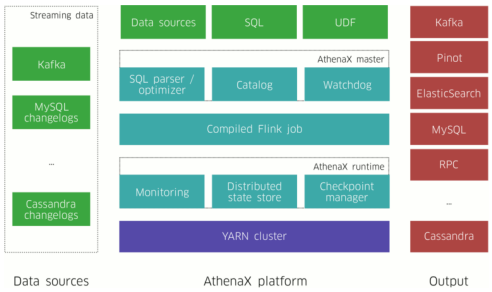

Abbildung 10: AthenaX nimmt Streaming-Daten und Abfragen als Eingaben auf, berechnet die Ergebnisse und leitet sie dann an eine Vielzahl von Ausgaben weiter.

Weitere Anwendungsfälle für Flink:
Zalando, eines der größten E-Commerce-Unternehmen in Europa, nutzt Flink für Echtzeit-Prozessüberwachung und ETL.

Telefónica NEXTs TÜV-zertifizierte Daten-Anonymisierungsplattform wird von Flink betrieben.

Otto Group, der weltweit zweitgrößte Online-Händler, nutzt Flink für die Business-Intelligence-Stream-Verarbeitung.

King, die Entwickler von Candy Crush Saga, nutzt Flink, um Data-Science-Teams ein Echtzeit-Analyse-Dashboard zur Verfügung zu stellen.

Ericsson nutzte Flink, um einen Echtzeitanomaliedetektor mit maschinellem Lernen über große Infrastrukturen zu bauen.

ResearchGate, ein soziales Netzwerk für Wissenschaftler, nutzt Flink für die Netzwerkanalyse und nahezu doppelte Erkennung.

3. Zusammenfassung

Im Gegensatz zu Apache Storm (das ebenfalls einem Datenflussmodell folgt) bietet Flink eine extrem einfache API auf hohem Niveau in Form von Map / Reduce, Filtern, Window, GroupBy, Sort und Joins. Dies gibt dem Entwickler viel Flexibilität und beschleunigt die Entwicklung beim Schreiben neuer Jobs. Zusammenfassend ist Apache Flink ein Open-Source-Stream-Verarbeitungs-Framework, das die „Performance vs. Reliability“-Konkurrenz eliminiert, die oft mit Open-Source-Streaming-Engines einhergeht und in beiden Kategorien konsistent funktioniert.

A. Abbildungsverzeichnis

Literatur

[Fou14] The Apache Software Foundation. *'Homepage'*. 2014. URL: https://flink.apache.org/index.html (besucht am 14. 04. 2018).

[Fou18a] The Apache Software Foundation. *Flink DataSet API Programming Guide*. 2018. URL: https://ci.apache.org/projects/flink/flink-docs-master/dev/batch/ (besucht am 15. 04. 2018).

[Fou18b] The Apache Software Foundation. *Flink DataStream API Programming Guide*. 2018. URL: https://ci.apache.org/projects/flink/flink-docs-master/dev/datastream_api.html (besucht am 14. 04. 2018).

[Fou18c] The Apache Software Foundation. *State & Fault Tolerance*. 2018. URL: https://ci.apache.org/projects/flink/flink-docs-master/dev/stream/state/ (besucht am 14. 04. 2018).

[Fou18d] The Apache Software Foundation. *Complex event processing for Flink*. 2018. URL: https://ci.apache.org/projects/flink/flink-docs-master/dev/libs/cep.html (besucht am 14. 04. 2018).

[Fou18e] The Apache Software Foundation. *Windows*. 2018. URL: https://ci.apache.org/projects/flink/flink-docs-master/dev/stream/operators/windows.html (besucht am 14. 04. 2018).

[Hao17] & Naveen Cherukuri Haohui Mai Bill Liu. *Introducing AthenaX, Uber Engineering's Open Source Streaming Analytics Platform*. 2017. URL: https://eng.uber.com/athenax/ (besucht am 17. 04. 2018).