Системы хранения и обработки
потоков больших данных

Введение в потоковую обработку данных

Самарев Роман Станиславович, 2023

МГТУ им. Н.Э. Баумана
Кафедра Компьютерные системы и сети

План презентации

Терминология

1. Потоковая обработка (здесь) – непрерывная обработка пакетов данных с минимальной задержкой
2. Фреймворк – библиотеки, набор инструментов, методика применения

Область применения потоковой обработки

- Регистрация событий
- Мониторинг активности
- Оперативное реагирование на изменение ситуации
- Internet of Things
- Тарификация
- Регистрация продаж
- ...

Технологические предпосылки потоковой обработки

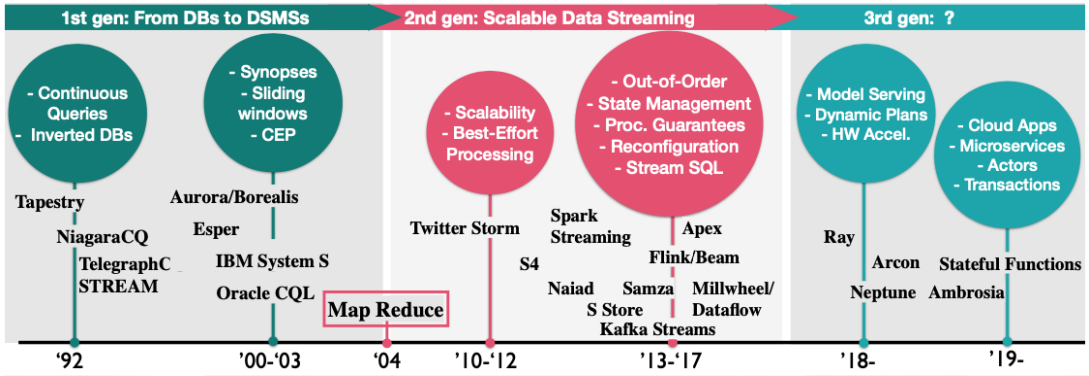
Исторические предпосылки:

- наличие сетевого взаимодействия
- наличие источников данных вне сервера
- возможность асинхронной обработки данных

Технологии передачи данных:

- очереди сообщений
- режимы pub/sub
- специализированные средства накопления событий

Эволюция потоковой обработки данных



M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos. A survey on the evolution of stream processing systems, 08 2020

Dataflow processing, Dataflow database machine, Parallel Dataflow Approach to SQL, Continues queries over data stream...

- H. C. M. Andrade, B. Gedik, and D. S. Turaga. *Fundamentals of Stream Processing: Application Design, Systems, and Analytics.*
  Cambridge University Press, New York, NY, USA, 1st edition, 2014
- L. Golab and M. T. Özsu. Issues in data stream management.
  ***SIGMOD Rec.*, 32(2):5–14, June 2003**

Gamma (DeWitt, 86) [7], Tapestry (Terry, 92) [19],

Aurora[1], Borealis, COUGAR, Gigascop, NiagaraCQ, OpenCQ, StatStream, STREAM, TelegraphCQ , Tribeca, ...

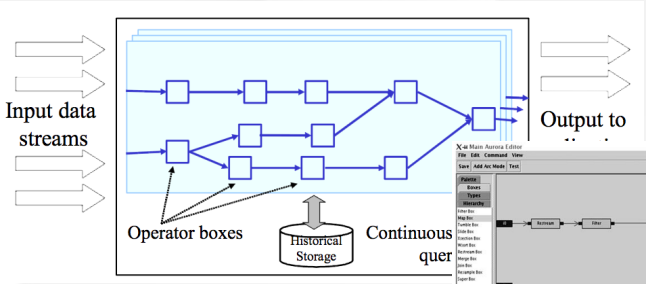
* CQ - Continuous Query

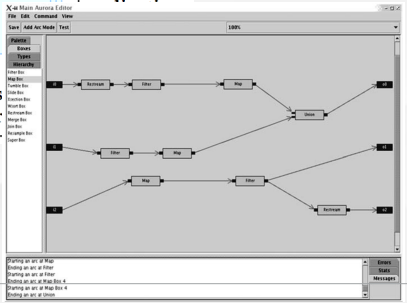
Разнообразие языков запросов и способов описания процесса обработки

Aurora: visual programming approach

D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A new model and architecture for data stream management.

***The VLDB Journal*, 12(2):120–139, Aug. 2003**





SQuAl ([S]tream [Qu]ery [Al]gebra).
Operations: filter, map, union, sort, aggregate, join, resample.

RuleML, Drools, JBoss Enterprise BRMS...

Описание правил обработки информации (сообщений в частном случае)

```
rule "When there is a fire turn on the sprinkler"
when
Fire($room : room)
$sprinkler : Sprinkler( room == $room, on == false )
then
modify( $sprinkler ) { setOn( true ) };
System.out.println( "Turn on the sprinkler for room " + $room.getName() );
end
```

https://docs.jboss.org/drools/release/6.5.0.Final/drools-docs/html/ch06.html

Подход обработки потоков как отдельных сообщений
Обобщение, обработка, порождение новых сообщений.

- SQL: TIBCO BusinessEvents, Oracle CEP, SAP ESP,...
- Java: Apache Flink, ...

```
DataStream<Event> input = ...
Pattern<Event, ?> pattern = Pattern
.begin("start").where(evt -> evt.getId() == 42)
.next("middle").subtype(SubEvent.class).where(subEvt -> subEvt.getVolume() >= 10.0)
.followedBy("end").where(evt -> evt.getName().equals("end"));

PatternStream<Event> patternStream = CEP.pattern(input, pattern);
DataStream<Alert> result = patternStream.select(pattern -> {
        return createAlertFrom(pattern);
});
```

Требования к потоковой обработке

M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 requirements of real-time stream processing.

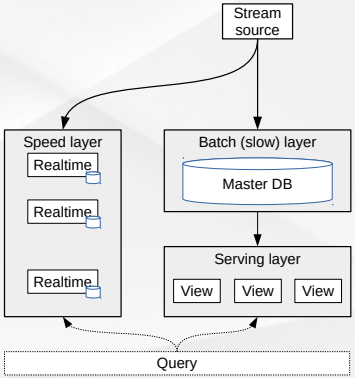
***SIGMOD Rec.*, 34(4):42–47, Dec. 2005**

Перевод `http://citforum.ru/database/articles/stream_8_req/`

1. Сохраняйте данные движущимися.
2. Формулируйте запросы с использованием SQL на потоках (StreamSQL).
3. Справляйтесь с дефектностью потоков (задержка, отсутствие и нарушение порядка данных).
4. Генерируйте предсказуемые результаты.
5. Интегрируйте хранимые и потоковые данные.
6. Гарантируйте безопасность и доступность данных.
7. Автоматически разделяйте и масштабируйте приложения.
8. Мгновенно обрабатывайте и выдавайте результаты.

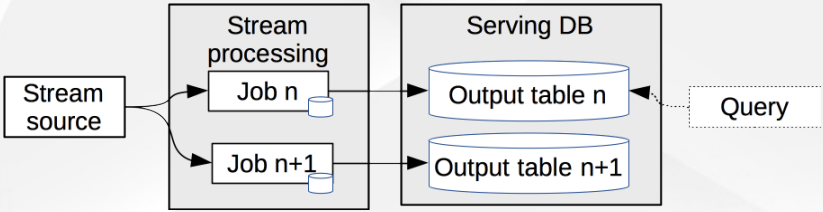
N. Marz and J. Warren. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems.*
Manning Publications Co., Greenwich, CT, USA, 1st edition, 2015

Ответ на вопрос собирается объединением из слоя потоковой (оперативной) и слоя медленной пакетной обработки.



Jay Kreps, CEO of Confluent

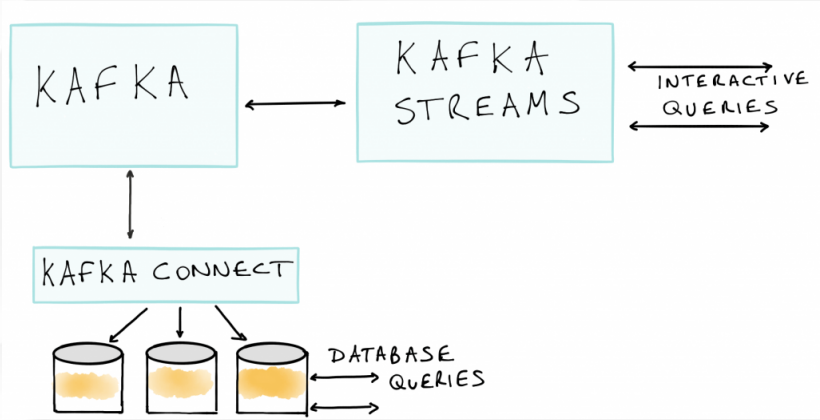
Вычисления только в тот момент, когда есть изменение данных



https://www.oreilly.com/ideas/questioning-the-lambda-architecture

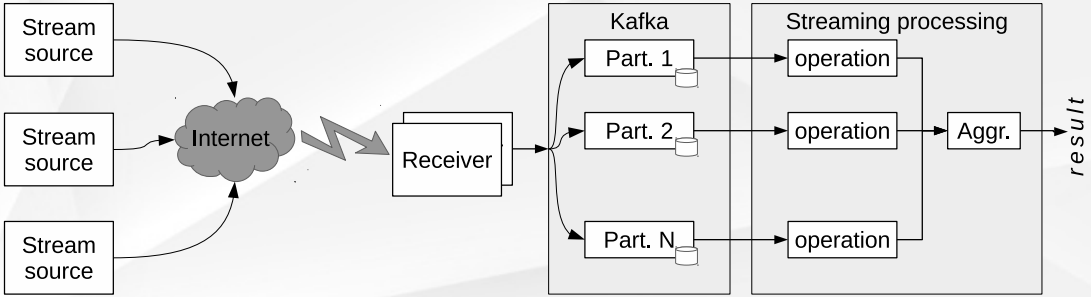
Поток как база данных

...what is the link between Interactive Queries of embedded state and traditional databases? The link is the notion of materialized views... We have made the case in the past that, for streams, materialized views can be thought as a cached subset of a log (i.e., topics in Kafka)....
Apache Kafka and Kafka Streams // Jay Kreps, Confluent



Типовой пример сбора и обработки сообщений

- Источники информации отправляют сообщения.
- Первичный приём и накопление выполняет Apache Kafka.
- Обработку данных реализуют при помощи потокового фреймворка (Apache Storm, Samza, Flink, Apex...).



Естественная потоковая обработка

Apache Storm, Apache Samza, Apache Flink
Сообщения обрабатываются индивидуально

Пакетная (micro-batches)

Apache Storm/Trident, Apache Spark Streaming
Сообщения группируются в пакет. Сообщения в пакете упорядочены. Обрабатывается весь пакет за раз.

Comparison of Apache Stream Processing Frameworks: Part 1 and Part 2:
https://cloud.tencent.com/developer/article/1088152

https://cloud.tencent.com/developer/article/1088157

Previously:

http://www.cakesolutions.net/teamblogs/comparison-of-apache-stream-processing-frameworks-part-1

http://www.cakesolutions.net/teamblogs/comparison-of-apache-stream-processing-frameworks-part-2

Порядок обработки событий

In-order data management

Способность обрабатывать события только в порядке их поступления.
Aurora, STREAMS, Timestream, Trill, Streamscope

Out-of-order data management

Обработка, детектирование и упорядочивание сообщений (при необходимости). Flink, Spark, ...
Используются механизмы slack, heartbeats, low-watermarks, pointstamps , triggers

Композиционная

Жесткая схема соединения элементарных компонентов (топология) через интерфейс сток-исток
Apache Storm, Apache Samza

Декларативная

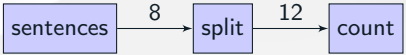
Высокоуровневая декларация логического плана обработки данных
Apache Storm/Trident, Apache Spark Streaming, Apache Flink, Apache Apex

Пример композиционной модели

Фрагмент кода для Apache Storm топологии подсчёта общего количества слов:

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("sentences", new RandomSentenceSpout(), 5);
builder.setBolt("split", new SplitSentence(), 8)
.shuffleGrouping("sentences");
builder.setBolt("count", new WordCount(), 12)
.fieldsGrouping("split", new Fields("word"));
```



http://storm.apache.org/releases/1.1.0/Tutorial.html

Пример декларативной модели

Фрагмент кода для Apache Flink процесса подсчёта количества слов за последние 5 секунд:

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<Tuple2<String, Integer>> dataStream = env
.socketTextStream("localhost", 9999)
.flatMap(new Splitter())
.keyBy(0)
.timeWindow(Time.seconds(5))
.sum(1);

dataStream.print();

env.execute("Window WordCount");
```

https://ci.apache.org/projects/flink/flink-docs-release-1.2/dev/datastream_api.html#example-program

Достоинства декларативной модели

- Возможность трансляции в физический план на этапе размещения приложения
- Возможность автоматического балансирования нагрузки по вычислительным узлам
- Возможность оптимизации операций
- Возможность описания логики на любом языке программирования (Java, Scala, Ruby, XML, SQL, ...)

Тенденции

Создание промежуточного слоя, способного транслировать логический план на разные фреймворки.

- Apache Beam – оболочка над потоковыми фреймворками.
- проект Emma [2] – DSEL на основе Scala для написания приложений Apache Flink или Apache Spark

Гарантированность обработки сообщений

Потоковый фреймворк обеспечивает безопасность и доступность данных

at most once

"Максимум один раз" – сообщение может быть доставлено ноль или 1 раз. Может быть потеряно.

at least once

"По крайней мере один раз" – сообщение гарантированно будет доставлено на обработку, но может быть с дубликатами. Проблему решает программист, использующий фреймворк.

exactly once

"Точно один раз" – сообщение будет гарантированно доставлено, и при этом исключены дубликаты.

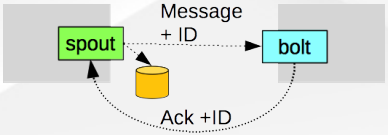
Подтверждение для каждого сообщения

Недостатки

- В распределённой среде из-за задержки потверждения появляются дубликаты
- Низкая производительность

Достоинства

- Простота реализации



spout – источник сообщений
bolt – получатель сообщений

http://storm.apache.org/releases/current/Guaranteeing-message-processing.html

http://data-artisans.com/high-throughput-low-latency-and-exactly-once-stream-processing-with-apache-flink/

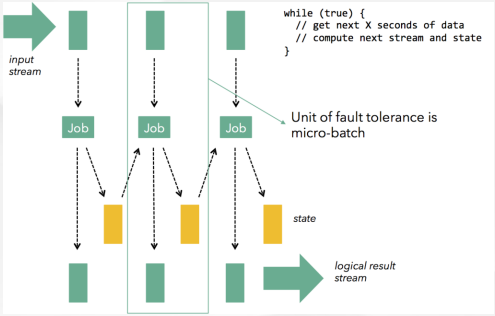
Подтверждение для каждого пакета сообщений (microbatching)

Недостатки

- Высокая задержка времени обработки и затруднён контроль обработки окна
- Возможно бездействие операторов при ожидании сохранения данных

Достоинства

- Теоретически высокая скорость



http://data-artisans.com/high-throughput-low-latency-and-exactly-once-stream-processing-with-apache-flink/

https://databricks.com/blog/2015/07/30/diving-into-apache-spark-streamings-execution-model.html

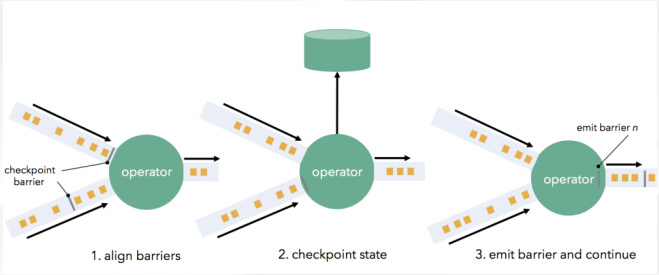
Asynchronous Barrier Snapshotting (ABS) [5]. Происходит выравнивание состояния по меткам "checkpoint barrier".

Недостатки

- Сложность реализации

Достоинства

- Высокая скорость сочетается с отсутствием простоя операторов



http://data-artisans.com/high-throughput-low-latency-and-exactly-once-stream-processing-with-apache-flink/

- Реализации операторов агрегации требует сохранения и восстановления предыдущего состояния (Apache Flink, Apache Spark, Apache Storm/Trident)
- Внутреннее накопление данных необходимо для реалиации операций с окном данных. Обработка активируется триггером

Виды окон

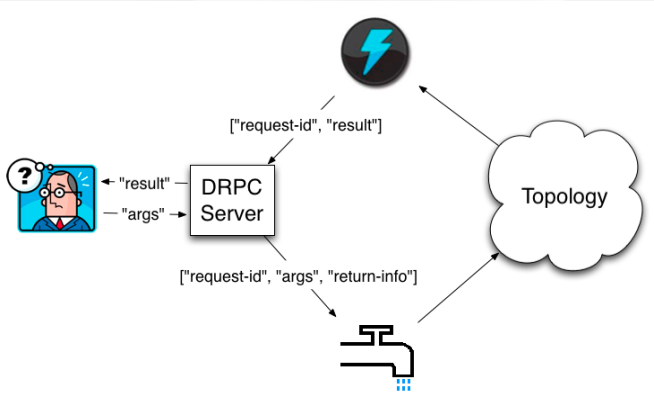
- С перекрытием (по времени или количеству сообщений)
- Последовательные

Виды триггеров

- Интервальные
  - по времени поступления
  - по времени в обработке
  - по времени пользователя
- По количеству сообщений

https://ci.apache.org/projects/flink/flink-docs-release-1.2/dev/windows.html

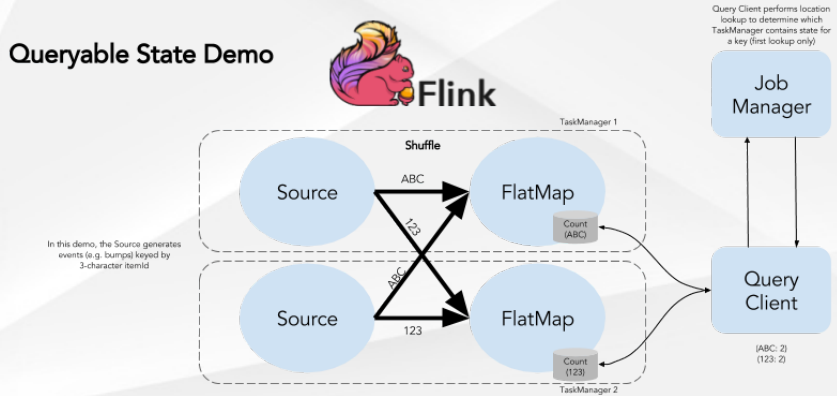
- Естественный процесс обработки – приём, обработка, отправка данных
- Модель distributed RPC (DRPC) в Apache Storm



https://storm.apache.org/releases/2.4.0/Distributed-RPC.html

Доступ к хранимому состоянию потока

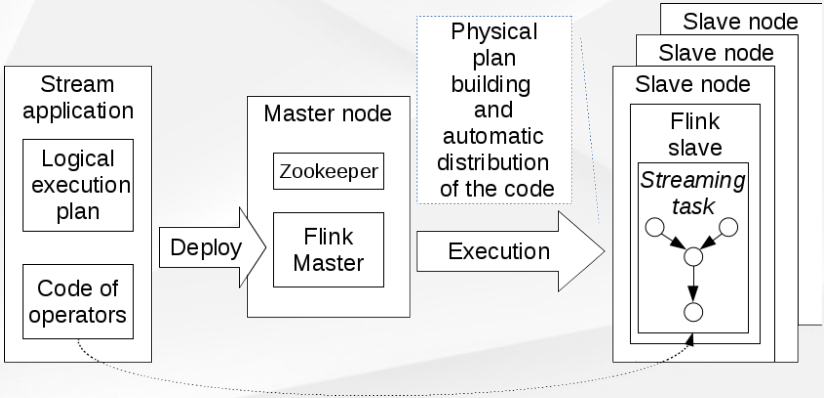
- Apache Storm Trident State
- Apache Flink Queryable State
- ...



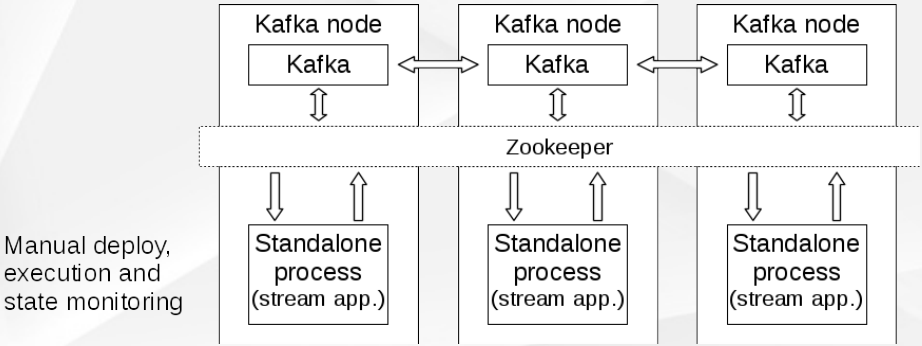
= Fault tolerant local state

Модель размещения приложений: автоматическая

Автоматическое размещение и распараллеливание на примере Apache Flink. Приложение хранит логический план обработки и набор операторов-классов. Размещение и распределение по узлам – автоматическое.



Приложение Apache Kafka Streams – автономное Java-приложение, полностью выполняющее обработку одного потока данных.
Масштабирование – за счёт запуска нескольких экземпляров приложения.



Примечание: похоже на проект Java Reactor Project

Один из первых и наиболее известный бенчмарк (для оценки Aurora project [1])

A. Arasu, M. Cherniack, E. F. Galvez, D. Maier, A. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts.

Linear road: A stream data management benchmark.

In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 480–491. Morgan Kaufmann, 2004

Linear Road: Basic principles

Basic statements of Linear City

- 100 x 100 miles area
- 10 parallel expressways
- 100 enterences/exit per each expressway
- Each express way contains 3 travel lanes and 1 entrance/exit ramp
- Every vehicle emits a position every 30 seconds.

Linear Road Requirements

- Toll Processing: Notifications, Assessments
- Accident Processing: Detection, Notification
- Historical Query Processing

Stream

(Type = 0, Time, VehicleID, Speed, XWay, Lane, Dir, Seg, Pos)

Disadvantage: implemented only for Aurora project

R. Lu, G. Wu, B. Xie, and J. Hu. Stream bench: Towards benchmarking modern distributed stream computing frameworks.

In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, UCC '14, pages 69–78, Washington, DC, USA, 2014. IEEE Computer Society

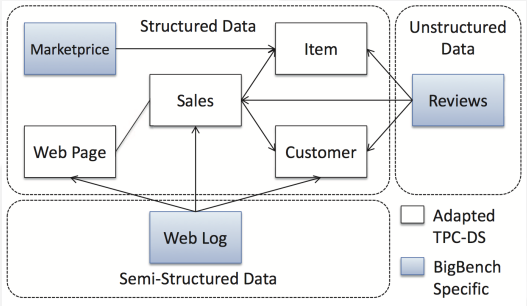
StreamBench

Datasets	AOL Search Data, CAIDA Anonymized, Internet Traces Dataset
Operations	Identity, Sample, Projection, Grep, Wordcount, DistinctCount, Statistics
Frameworks	Apache Spark, Storm

Non reproducable. Without source codes.

BigBench [10] и BigBench2 [17] (Tilmann Rabl, Kai Sachs, Meikel Poess, Chaitanya K. Baru, Hans-Arno Jacobsen)

Dataset: TPC-DS (TPC Benchmark DS: 'The' Benchmark Standard for decision support solutions including Big Data)
Control parameters: volume, variety, velocity



Модификация BigBench [16] для тестирования на некоторых запросах TPC-DS в терминах декларативной Java-модели Apache Flink и HiveQL.

S. Ekanayake. *Towards a systematic study of big data performance and benchmarking.*

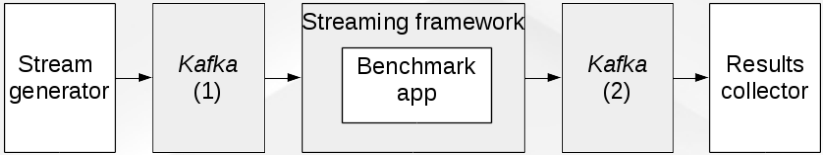
PhD thesis, the School of Informatics and Computing, Indiana University, United States – Indiana, 10 2016.

`https://www.researchgate.net/publication/308761924_Towards_a_Systematic_Study_of_Big_Data_Performance_and_Benchmarking`

- Berkeley Big Data Benchmark
- BigDataBench
- HiBench
- Graph500
- MineBench
- ...

Типовая схема тестирования фреймворка

- Stream generator обеспечивает поток сообщений заданной плотности
- Kafka(1) и Kafka(2) – входная и выходная очереди сообщений. Могут быть заменены собственными средствами фреймворков или быть исключены.
- Benchmark app – приложение, выполняющее тестовую нагрузку

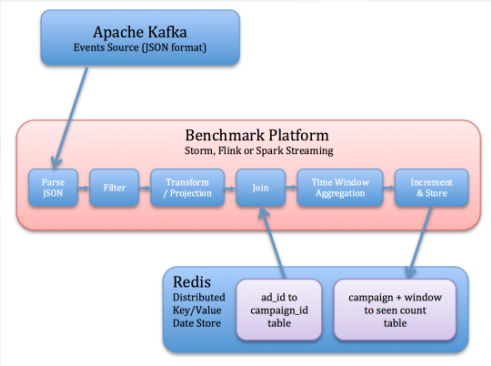


Для управления процессом тестирования использовать, например, Peel Framework - http://peel-framework.org/

Оценка задержки обработки данных

Проблема: данные обрабатываются с задержкой. Критично для приложений “реального времени“.

Цель проверки: оценить задержку обработки, вызванную передачей по конвейеру



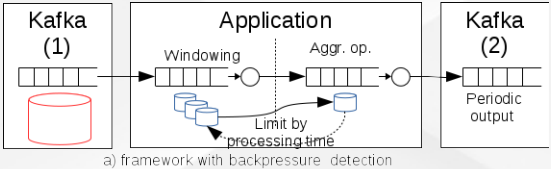
Yahoo Streaming Benchmark [6] (в статье – для Apache Storm, Spark и Flink).

`https://github.com/yahoo/streaming-benchmarks`

Формула вычисления задержки: $window.final\_event\_latency = (window.last\_updated\_at - window.timestamp) - window.duration$

Проблема: имеем операцию с окном по времени. Необходимо обеспечить стабильный выход результата

Цель проверки: оценить качество работы back pressure detector



a) Apache Flink with back pressusure detector; b) Apache Spark https://github.com/rssdev10/spark-kafka-streaming

Оценка восстановления узлов кластера после сбоя

Проблема: поток запускается однократно и обеспечивает обработку данных неограниченно долгое время. Узлы кластера могут выходить из строя. Процесс не должен деградировать со временем

Цель проверки: оценить способоность фреймворка восстанавливать связи

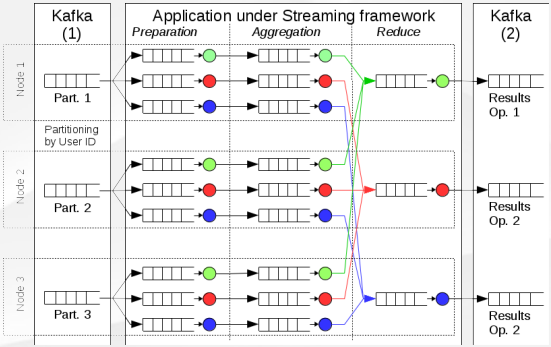
Пример: в работе Diana Matar (TU-Berlin) [15] эмулировались сбои узлов кластера и оценивалось влияние на производительность. Spark Streaming 1.3 не восстанавливает обработку данных.

Оценка способности обработки без лишних сетевых обменов

Оценка возможности организации последовательной обработки данных без лишних сетевых обменов

Проблема: сеть ограничена. Apache Kafka (или аналог) реализует распределённое хранение. Фреймворк не должен требовать данные, расположенные на других узлах

Цель проверки: оценить реализуемость и качество управления



Оценка сохранения состояния

Оценка предельных возможностей сохранения состояния (оконные операции)

Проблема: операции аггрегации (реализуются в окне данных) требуют временного хранения в оперативной памяти, которая ограничена. Временное состояния должно быть защищено от сбоя.

Цель проверки: Оценить накладные расходы фреймворка для хранения состояния в оперативной памяти и задержки сохранения и восстановления на постоянном носителе.

Проблема: фреймворки реализуют разные интерфейсы и языки запросов (понимаем этот как любой язык обработки данных).

Цель проверки:

- Оценить синтаксис языков запросов и семантика (включая совместимость SQL, особенности операций аггрегации)
- Оценить полноту реализации языка запросов
- Оценить применимость стандартизованных бенчмарков (семейство TPC)
- Оценить производительность обработки данных на данном языке запросов

Spark SQL, Flink Table API and SQL,...

M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm. A catalog of stream processing optimizations.

***ACM Comput. Surv.*, 46(4):46:1–46:34, Mar. 2014**

Проблема: план выполнения операторов зависит от данных и доступных аппаратных ресурсов. Фреймворк должен автоматически оптимизировать логический план.

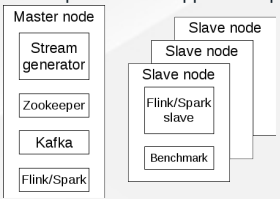
Цель проверки: выявить поддерживаемые оптимизации и оценить их качество.

- Operator reordering
- Redundancy elimination
- Operator separation
- Fusion
- Fission
- Placement
- Load balancing
- State sharing
- Batching
- Algorithm selection
- Load shedding

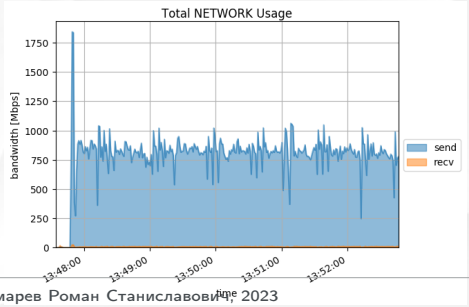
Проблема: нехватка пропускной способности сети ограничивает производительность и масштабируемость

Цель проверки: оценить характер загрузки сети и влияние на производительность

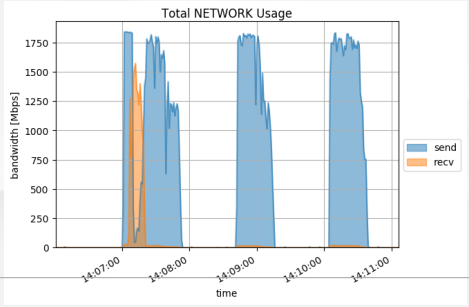
Схема организации потока данных для оценки



Apache Flink

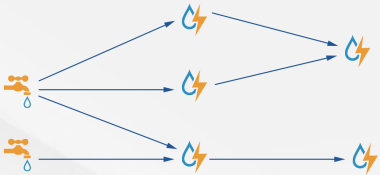


Apache Spark



Event size	single message
Delivery guarantees	at least once
Data flow	topology
Mainly powered by	Twitter, Hortonworks
Adventages	Low latency. Well known and widely used
Disadvantages	Low throughput

http://storm.apache.org/



Хранение состояния реализовано в модуле Apache Storm Trident – https://storm.apache.org/releases/2.4.0/Trident-tutorial.html

Event size	single message
Delivery guarantees	at least once
Data flow	topology
Mainly powered by	Linkedin
Specifics	Primary oriented to work with Kafka
Adventages	Low latency. High throughput
Disadvantages	Low level programming conception

http://samza.apache.org/

https://engineering.linkedin.com/performance/benchmarking-apache-samza-12-million-messages-second-single-node

Event size	microbatch
Delivery guarantees	exactly once
Data flow	Application with declarative description
Powered by	`amplab.cs.berkeley.edu/`, Databricks
Specifics	batch oriented
Adventages	Relativaly high throughput, very popular
Disadvantages	Limited support of timed windows, unworkable backpressure technique, cluster degradation (checked upto v2.0)

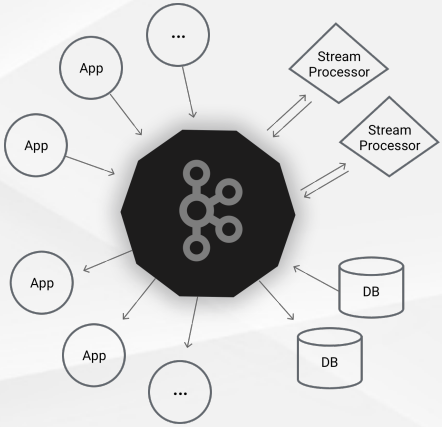
`http://spark.apache.org/`

Event size	single message
Delivery guarantees	exactly once
Data flow	Application with declarative description
Powered by	`www.dima.tu-berlin.de/`, data Artisans
Adventages	Low latency, high throughput
Disadvantages	limited ML support

`http://flink.apache.org/`

http://kafka.apache.org/

Event size	single message
Delivery guarantees	at least once
Data flow	Topology
Powered by	Confluent
Specifics	other semantic of aggregation operations vs Flink, Spark
Adventages	Low latency, high throughput
Disadvantages	Manual scalability



Event size	single message
Delivery guarantees	at least once
Data flow	Job with a declarative description
Powered by	GridGain
Disadvantages	Limited community

https://ignite.apache.org

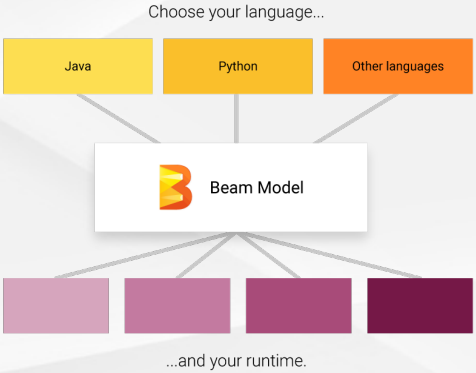
Event size	single message
Delivery guarantees	at least once
Data flow	Application with a declarative description
Powered by	DataTorrent
Adventages	Low Latency, High throughput
Disadvantages	Limited community

This project has retired since 2018.

`https://apex.apache.org`

Event size	single message
Delivery guarantees	exactly once
Data flow	Application with declarative pipline
Powered by	Google
Specifics	Wrapper over Flink, Apex, Cloud Dataflow, Spark (limited)
Adventages	Low latency, high throughput

http://beam.apache.org/



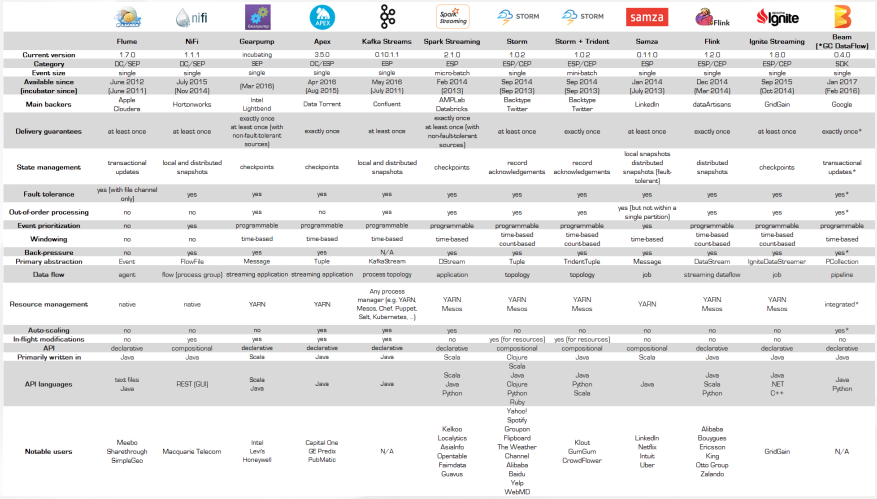
Коммерческие потоковые фреймворки

Mike, Gualtieri and Rowan, Curran and Holger, Kisker and Emily, Miller and Matthew, Izzi The Forrester Wave™: Big Data Streaming Analytics, Q1 2016

https://www.sas.com/content/dam/SAS/en_us/doc/analystreport/forrester-big-data-streaming-analytics-108218.pdf

- Cisco Connected Streaming Analytics
- Data Torrent RTS
- Esper Enterprise Edition
- IBM Streams
- Impetus Technologies StreamAnalytix
- Oracle Stream Explorer
- SAP Event Stream Processor
- Software AG Apama Streaming Analytics Platform
- SQLstream Blaze
- Striim
- TIBCO StreamBase
- WSO2 Complex Event Processor

Потоковые фреймворки под управлением фонда Apache



	Flume	NiFi	Gearpump	Apex	Kafka Streams	Spark Streaming	Storm	Storm + Trident	Samza	Flink	Ignite Streaming	Beam (*GC DataFlow)
Current version	1.7.0	1.1.1	incubating	3.5.0	0.10.1.1	2.1.0	1.0.2	1.0.2	0.11.0	1.2.0	1.8.0	0.4.0
Category	DC/SEP	DC/SEP	SEP	DC/ESP	ESP	micro-batch	ESP/CEP	ESP/CEP	ESP	ESP/CEP	ESP/CEP	SDK
Event size	single	single	single	single	single	single	single	mini-batch	single	single	single	single
Available since (incubator since)	June 2012 (June 2011)	July 2015 (Nov 2014)	(Mar 2016)	Apr 2016 (Aug 2015)	May 2016	Feb 2014 (2013)	Sep 2014 (Sep 2013)	Sep 2014 (Sep 2013)	Jan 2014 (July 2013)	Dec 2014 (Mar 2014)	Sep 2015 (Jan 2014)	Jan 2017 (Feb 2016)
Main backers	Apple Cloudera	Hortonworks	Intel Lightbend	Data Torrent	Confluent	AMPLab Databricks	Backtype Twitter	Backtype Twitter	LinkedIn	dataArtisans	GridGain	Google
Delivery guarantees	at least once	at least once	exactly once at least once (with non-fault-tolerant sources)	exactly once	at least once	exactly once at least once (with non-fault-tolerant sources)	at least once	exactly once	at least once	exactly once	at least once	exactly once*
State management	transactional updates	local and distributed snapshots	checkpoints	checkpoints	local and distributed snapshots	checkpoints	record acknowledgements	record acknowledgements	local snapshots distributed snapshots (fault-tolerant)	distributed snapshots	checkpoints	transactional updates*
Fault tolerance	yes (with file channel only)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes*
Out-of-order processing	no	no	yes	no	yes	yes	yes	yes	yes (but not within a single partition)	yes	yes	yes*
Event prioritization	no	yes	programmable	programmable	programmable	programmable	programmable	programmable	yes	programmable	programmable	programmable
Windowing	no	no	time-based	time-based	time-based	time-based	time-based counts-based	time-based counts-based	time-based	time-based counts-based	time-based counts-based	time-based
Back-pressure	no	yes	yes	yes	N/A	yes	yes	yes	yes	yes	yes	yes*
Primary abstraction	Event	FlowFile	Message	Tuple	KafkaStream	DStream	Tuple	TridentTuple	Message	DataStream	IgniteDataStreamer	PCollection
Data flow	agent	flow (process group)	streaming application	streaming application	process topology	application	topology	topology	job	streaming dataflow	job	pipeline
Resource management	native	native	YARN	YARN	Any process manager (e.g. YARN, Mesos, Chef, Puppet, Salt, Kubernetes, ..)	YARN Mesos	YARN Mesos	YARN Mesos	YARN	YARN Mesos	YARN Mesos	integrated*
Auto-scaling	no	no	no	yes	yes	yes	no	no	no	no	no	yes*
In-flight modifications	no	yes	yes	yes	yes	no	yes (for resources)	yes (for resources)	no	no	no	no
API	declarative	compositional	declarative	declarative	declarative	declarative	compositional	compositional	compositional	declarative	declarative	declarative
Primarily written in	Java	Java	Scala	Java	Java	Scala	Clojure	Java	Scala	Java	Java	Java
API languages	text files Java	REST (GUI)	Scala Java	Java	Java	Scala Java Python	Scala Java Clojure Python Ruby	Java Python Scala	Java	Java Scala Python	Java .NET C++	Java Python
Notable users	Mozilla Shareathrough SimpleGeo	Macquarie Telecom	Intel Lev's Honeywell	Capital One GE Predix PubMatic	N/A	Kellkoo Localytics AsiaInfo Openstable Famdata Guavus	Yahoo! Spotify Groupon Flipboard The Weather Channel Alibaba Baidu Yelp WebMD	Klout GumGum CrowdFlower	LinkedIn Netflix Intuit Uber	Alibaba Bouygues Ericsson Kng Otto Group Zalando	GridGain	N/A

Ian Hellström, An Overview of Apache Streaming Technologies https://databaseline.tech/an-overview-of-apache-streaming-technologies/

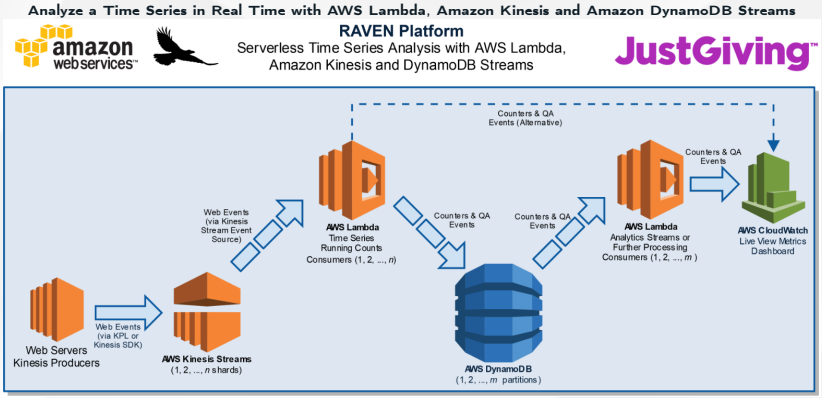
Достоинства

- Нет необходимости поддерживать собственный кластер
- Проблемы стабильности фреймворка – проблемы поставщика услуги
- Проблемы производительности решаемы за счёт избыточности оборудования

Недостатки

- Привязка к конкретному облачному поставщику услуги
- Сложность отладки
- Масштабируемость может быть ограничена особенностями реализации фреймворка
- Невозможно оценить предельные возможности фреймворка

Amazon Kinesis, `https://aws.amazon.com/kinesis`



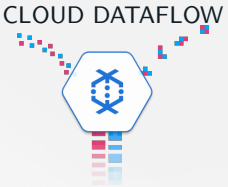
Programming Model	AWS SDKs for Java, JavaScript, .NET, Node.js, PHP, Python, and Ruby
Additional tools	Amazon Kinesis Firehose, Amazon Kinesis Analytics, Amazon Kinesis Analytics

Azure Stream Analytics `https://azure.microsoft.com/en-us/services/stream-analytics/`

Programming Model	.Net and REST API
Additional tools	Event Hubs, Machine Learning, IoT Hub

`https://cloud.google.com/dataflow/`

Programming Model	Apache Beam SDK
Additional tools	Cloud Storage, Cloud Pub/Sub, Cloud Datastore, Cloud Bigtable, and BigQuery



Same Apache Beam SDK for both local and cloud deployment.

Итоги

- Область потоковой обработки интенсивно развивается
- Определились типовые подходы построения приложений
- Отсутствуют стандарты на языки запросов или методы описания процесса обработки
- Отсутствут типовые методы оценки различных потоковых фреймворков
- Нет возможности сравнить коммерческие и облачные фреймворки
- В каждом конкретном случае разработки бизнес-приложения необходимо проверять все фреймворки-кандидаты

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, Aug. 2003.

[2] A. Alexandrov, A. Salzmann, G. Krastev, A. Katsifodimos, and V. Markl. Emma in action: Declarative dataflows for scalable data analysis. In F. Özcan, G. Koutrika, and S. Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 2073–2076. ACM, 2016.

[3] H. C. M. Andrade, B. Gedik, and D. S. Turaga. *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*. Cambridge University Press, New York, NY, USA, 1st edition, 2014.

[4] A. Arasu, M. Cherniack, E. F. Galvez, D. Maier, A. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear road: A stream data management benchmark. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 480–491. Morgan Kaufmann, 2004.

[5] P. Carbone, G. Fóra, S. Ewen, S. Haridi, and K. Tzoumas. Lightweight asynchronous snapshots for distributed dataflows. *CoRR*, abs/1506.08603, 2015.

[6] S. Chintapalli, D. Dagit, B. Evans, R. Farivar, T. Graves, M. Holderbaugh, Z. Liu, K. Nusbaum, K. Patil, B. J. Peng, and P. Poulosky. Benchmarking streaming computation engines: Storm, flink and spark streaming. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1789–1792, May 2016.

[7] D. J. DeWitt, R. H. Gerber, G. Graefe, M. L. Heytens, K. B. Kumar, and M. Muralikrishna. Gamma - a high performance dataflow database machine. In *Proceedings of the 12th International Conference on Very Large Data Bases*, VLDB '86, pages 228–237, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[8] S. Ekanayake. *Towards a systematic study of big data performance and benchmarking*. PhD thesis, the School of Informatics and Computing, Indiana University, United States – Indiana, 10 2016. `https://www.researchgate.net/publication/308761924_Towards_a_Systematic_Study_of_Big_Data_Performance_and_Benchmarking`.

[9] M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos. A survey on the evolution of stream processing systems, 08 2020.

[10] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen. Bigbench: Towards an industry standard benchmark for big data analytics. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1197–1208, New York, NY, USA, 2013. ACM.

[11] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, June 2003.

[12] M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm. A catalog of stream processing optimizations. *ACM Comput. Surv.*, 46(4):46:1–46:34, Mar. 2014.

[13] R. Lu, G. Wu, B. Xie, and J. Hu. Stream bench: Towards benchmarking modern distributed stream computing frameworks. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, UCC '14, pages 69–78, Washington, DC, USA, 2014. IEEE Computer Society.

[14] N. Marz and J. Warren. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2015.

[15] D. Matar. Benchmarking Fault-Tolerance in Stream Processing Systems. Master's thesis, TU-Berlin, 2016.

[16] G. Mazza. big data streaming processing engines under the umbrella of the apache foundation: benchmark and industrial applications. Master's thesis.

[17] T. Rabl, M. Frank, M. Danisch, H.-A. Jacobsen, and B. Gowda. The vision of bigbench 2.0. In *Proceedings of the Fourth Workshop on Data Analytics in the Cloud*, DanaC'15, pages 3:1–3:4, New York, NY, USA, 2015. ACM.

[18] M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 requirements of real-time stream processing. *SIGMOD Rec.*, 34(4):42–47, Dec. 2005.

[19] D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, SIGMOD '92, pages 321–330, New York, NY, USA, 1992. ACM.