TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HCM

KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

---------- oOo ----------

Báo cáo môn Kỹ Thuật Dữ Liệu

Real-Time Analytic Data with Apache Beam and Google Cloud Platform

Giảng viên: Giảng viên: PGS.TS Trần Minh Quang - PGS.TS Đặng Trần Khánh

Học Viên: Nhóm 10

Phạm Linh Sơn (1970596) - Tô Ngọc Long Hồ (1970588)

Nguyễn Xuân Vĩnh Hưng (1970589) - Trần Đình Lai (1970591)

TP. Hồ Chí Minh, năm 2020

MỤC LỤC

I. Mở đầu

1.1 Giới thiệu bài toán

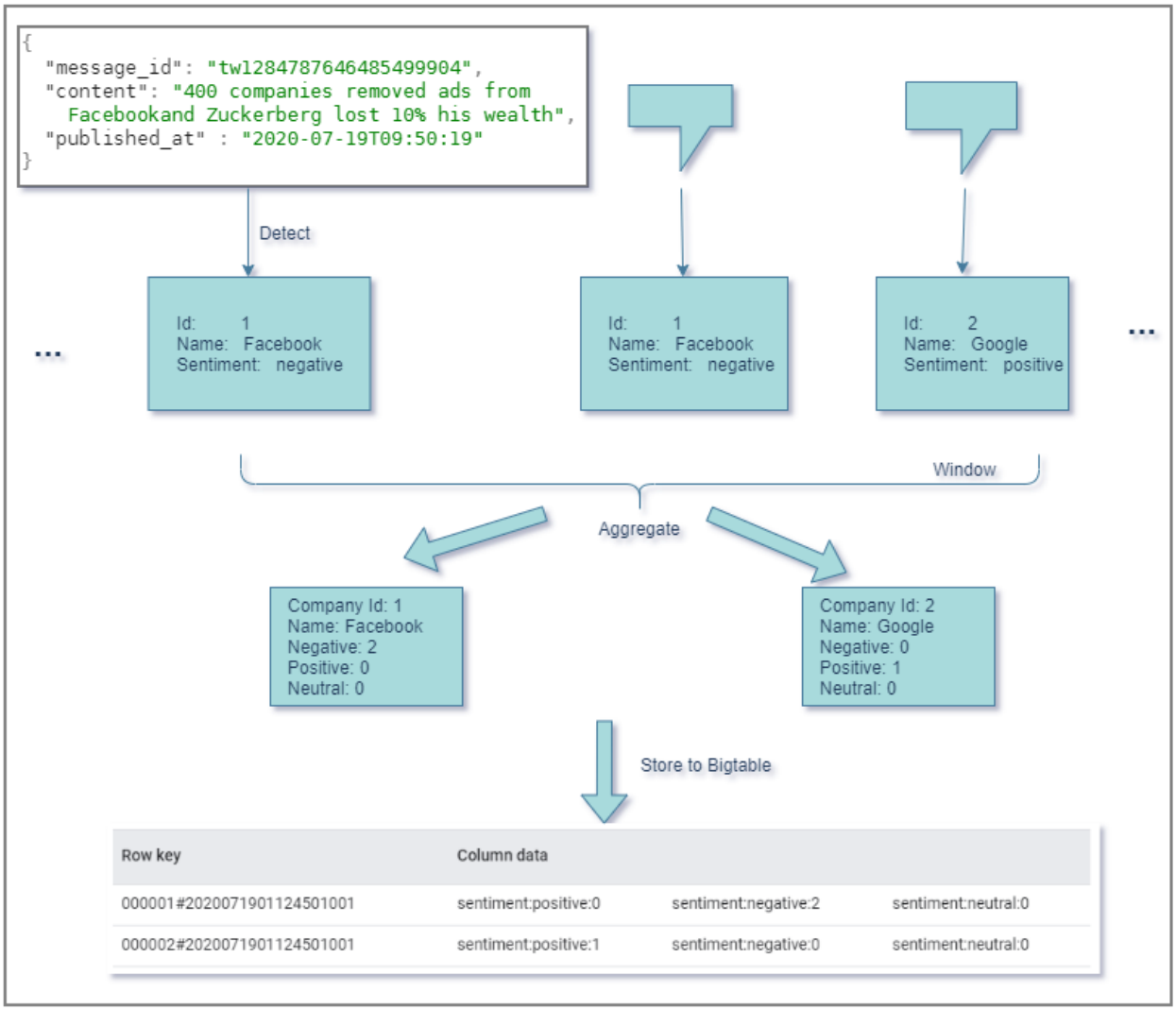
Yêu cầu bài toán là phân tích, tổng hợp dữ liệu từ mạng xã hội (social media) nói về các công ty. Những dữ liệu đó có thể được dùng để dự báo giá chứng khoán của các công ty.

Yêu cầu đặt ra:

- ➔ Phân tích dữ liệu trong thời gian thực (real-time).
- ➔ Dữ liệu streaming (dữ liệu không giới hạn đến liên tục theo thời gian).
- ➔ Yêu cầu đáp ứng truy xuất dữ liệu sau khi xử lý với độ trễ thấp.

Nguồn dữ liệu đầu vào dữ liệu không cấu trúc (unstructured data) từ các nguồn như tin tức (news), blogs, tweets (twitter), vv...

Qua hệ thống phân tích (analytics), tổng hợp (aggregate) ra dữ liệu những dữ liệu sentiment có liên quan tới công ty. (tổng hợp rất nhiều message từ social media nói về công ty đó).

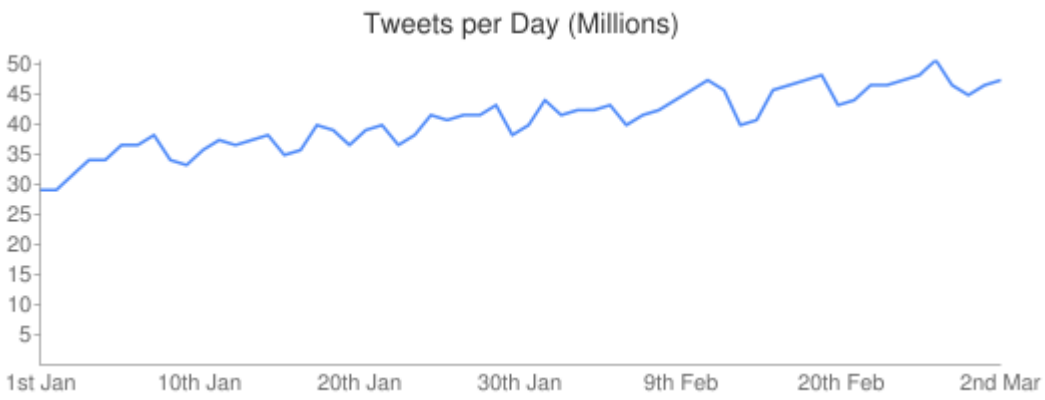


Hình: Phân tích, tổng hợp dữ liệu về các công ty từ tin nhắn

1.2 Vấn đề kỹ thuật

Vấn đề công nghệ cần giải quyết:

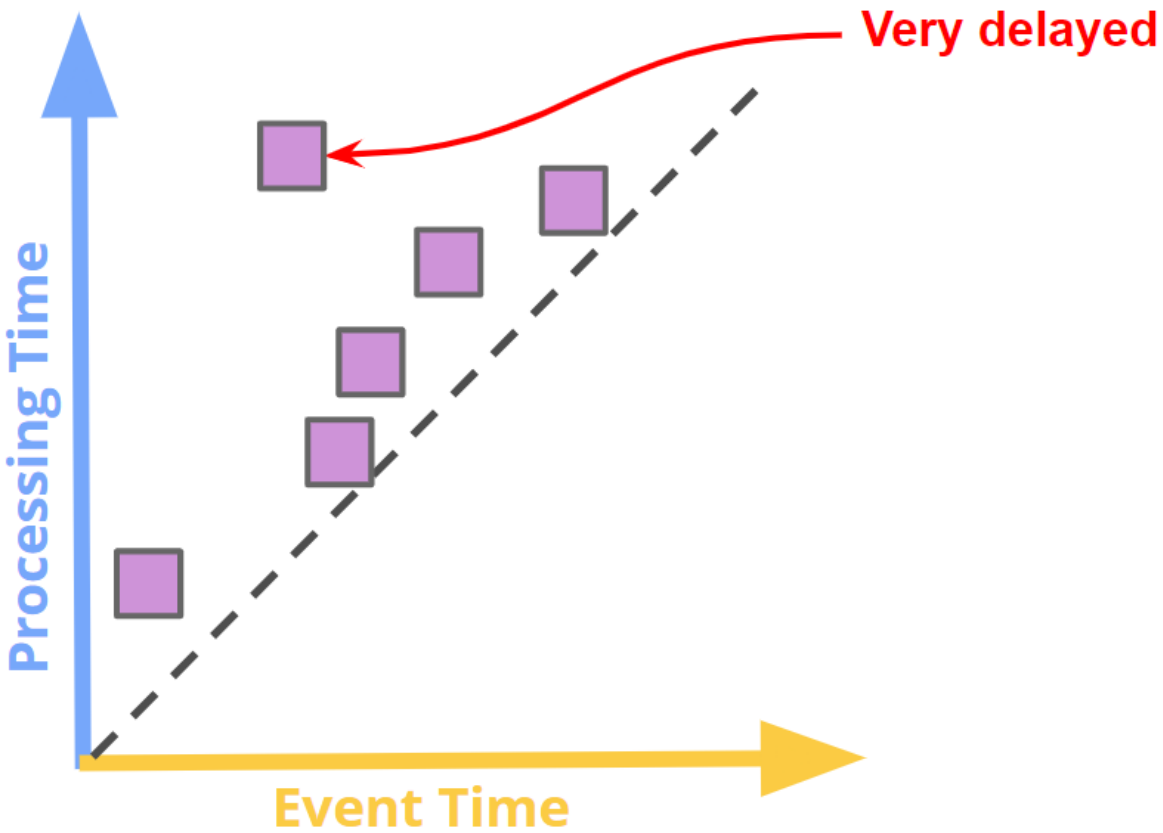
- Big Data: dữ liệu lớn từ mạng xã hội, cùng nói về các công ty. Twitter có hơn 50 triệu tweet (tin) mỗi ngày.



Nguồn: https://www.dsayce.com

- Dữ liệu thời gian thực và không giới hạn kích thước (unbounded data) và không kết thúc (stream data). Nên cần công nghệ xử lý streaming data.

- Dữ liệu đến trễ, đôi khi rất trễ và không xác định trước được thời gian trễ.



- Yêu cầu phân tích dữ liệu và đáp ứng trong thời gian thực. Độ trễ thấp.
- Hệ thống cần có khả năng tự mở rộng (auto scale) khi cần thiết, để đáp ứng xử lý lượng lớn dữ liệu.

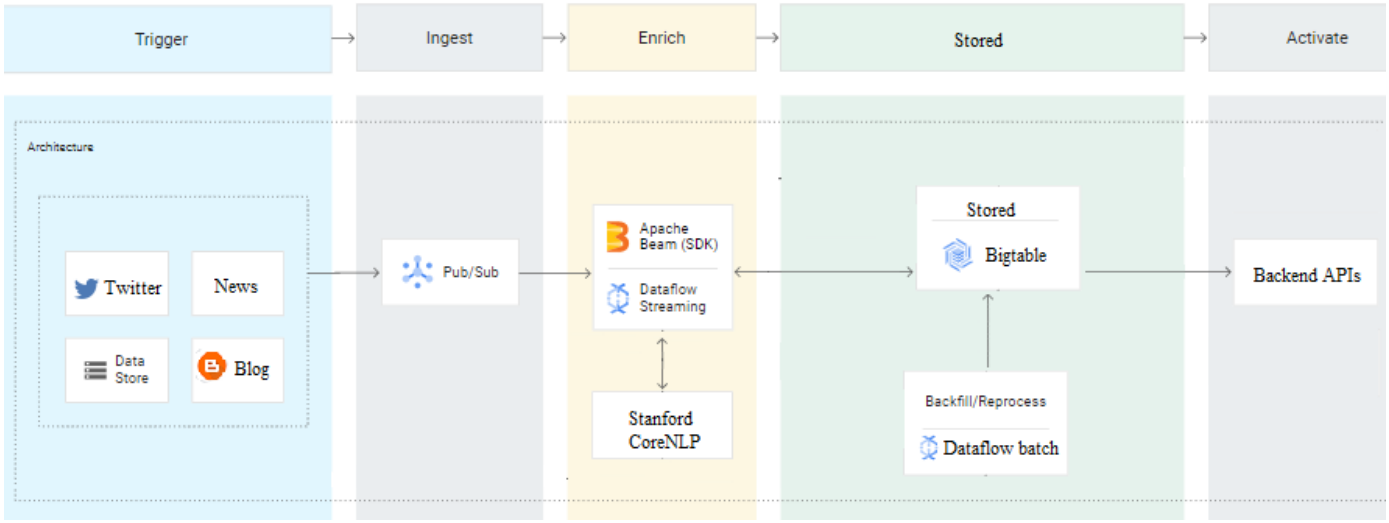
II. Hiện thực

2.1 Tổng quan

2.1 a) Áp dụng Google Cloud Platform.

- Sử dụng **Google Cloud Pub/Sub** làm message queue để nhận dữ liệu đầu vào.
- Sử dụng streaming processing framework: **Apache Beam** chạy trên nền **Google Dataflow** để xử lý dữ liệu streaming.
- Sử dụng thư viện / API để phân tích dữ liệu (trong nội dung project này, nhóm không nghiên cứu về việc phân tích dữ liệu dùng NLP, chỉ dùng thư viện của Stanford CoreNLP (opensource).
- Lưu trữ dữ liệu đã phân tích vào **Cloud Bigtable**, để phục vụ cho nhu cầu truy vấn thời gian thực (dự báo giá chứng khoán, áp dụng vào các công thức).

2.1 b) Kiến trúc hệ thống:



Hình: kiến trúc hệ thống

2.1 c) **Các bước (step) xử lý dữ liệu:**

- Bước 1, tiền xử lý dữ liệu: Dữ liệu được lấy về từ nhiều nguồn khác nhau như Twitter, các trang báo, blog, hay từ các file đã lưu trữ từ trước. Dữ liệu lọc để loại bỏ dữ liệu rác, thiếu thông tin cần thiết, đưa về một cấu trúc chung là json thống nhất.

  Cấu trúc của message:

```
{
    "id": "<ID của messae>",
    "content": "Nội dung của message ",
    "published_at" : "Thời gian tạo message"
}
```

- Bước 2: Nhận dữ liệu, dữ liệu được đưa vào Cloud Pub/Sub để thực hiện các bước xử lý tiếp theo.
- Bước 3: Xử lý dữ liệu streaming.

  Xử lý dữ liệu được chia ra nhiều bước nhỏ:

  1. Lọc những dữ liệu rác, dữ liệu thiếu các thuộc tính, dữ liệu đến quá trễ….

2. Detect ngôn ngữ của message, chỉ giữ lại tiếng anh.

3. Gọi Stanford CoreNLP API để “detect company, sentiment”.

4: Xử lý dữ liệu theo streaming. (group dữ liệu theo window, group dữ liệu theo kỹ, xử lý dữ liệu đến trễ).

5. Lưu trữ dữ liệu đã được phân tích, tổng hợp vào Bigtable. Để hỗ trợ truy vấn theo time-series về sau.

6. Triển khai pipeline lên Dataflow, thực hiện auto-scale, monitor...

2.2 Google Pub/Sub

Pub/Sub là một messaging service bất đồng bộ, lợi ích chính của Pub/Sub là tách biệt các services trong dự án.

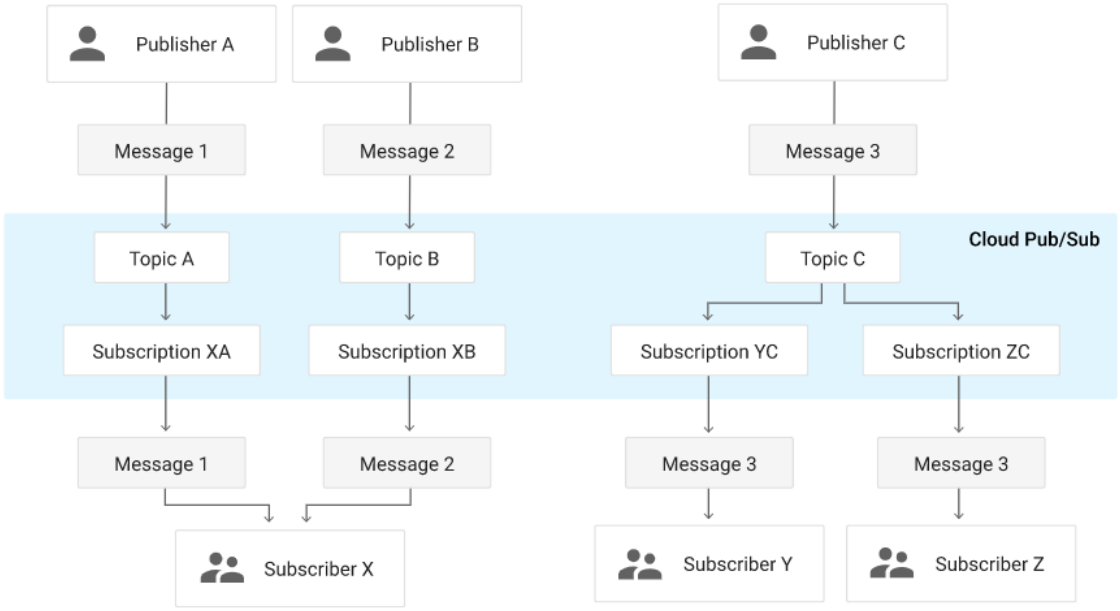
Để thiết kế ứng dụng hướng đến xử lý messaging hoặc events chúng ta hoàn toàn có thể sử dụng Pub/Sub để tạo một Middleware. Dữ liệu được gửi và nhận thông qua streaming pipelines.

Pub/Sub đảm bảo lưu trữ và gửi dữ liệu real-time, high performance và high availability.

Các khái niệm chính trong mô hình hoạt động của Pub/Sub

- **Topic:** Publishers gom nhóm các message theo tên nhất định.
- **Subscription:** Chỉ định các topic cụ thể được gửi đến subscribing.
- **Message:** Gồm có dữ liệu gửi và các thuộc tính của publisher
- **Message attributes:** Có cấu trúc <key, value>. Ví dụ: *<Language, ‘japan’>*

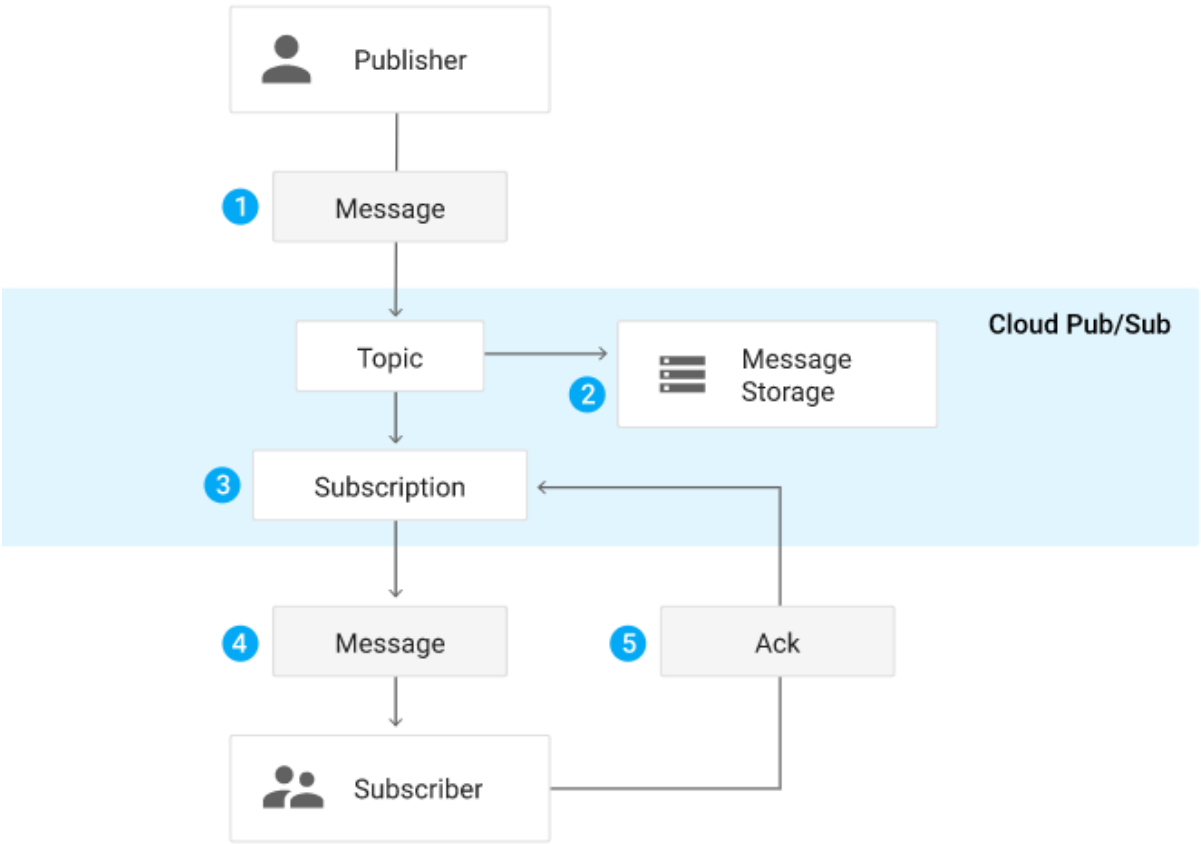
Mối quan hệ giữa Subscribers và Publishers



Hình ảnh từ cloud.google.com

Publishers tạo message và gửi đến topic, subscribers khởi tạo đăng ký đến các topic để nhận messages từ đó. Mối quan hệ này bao gồm: one-to-many, many-to-many, many-to-one.

Message flow



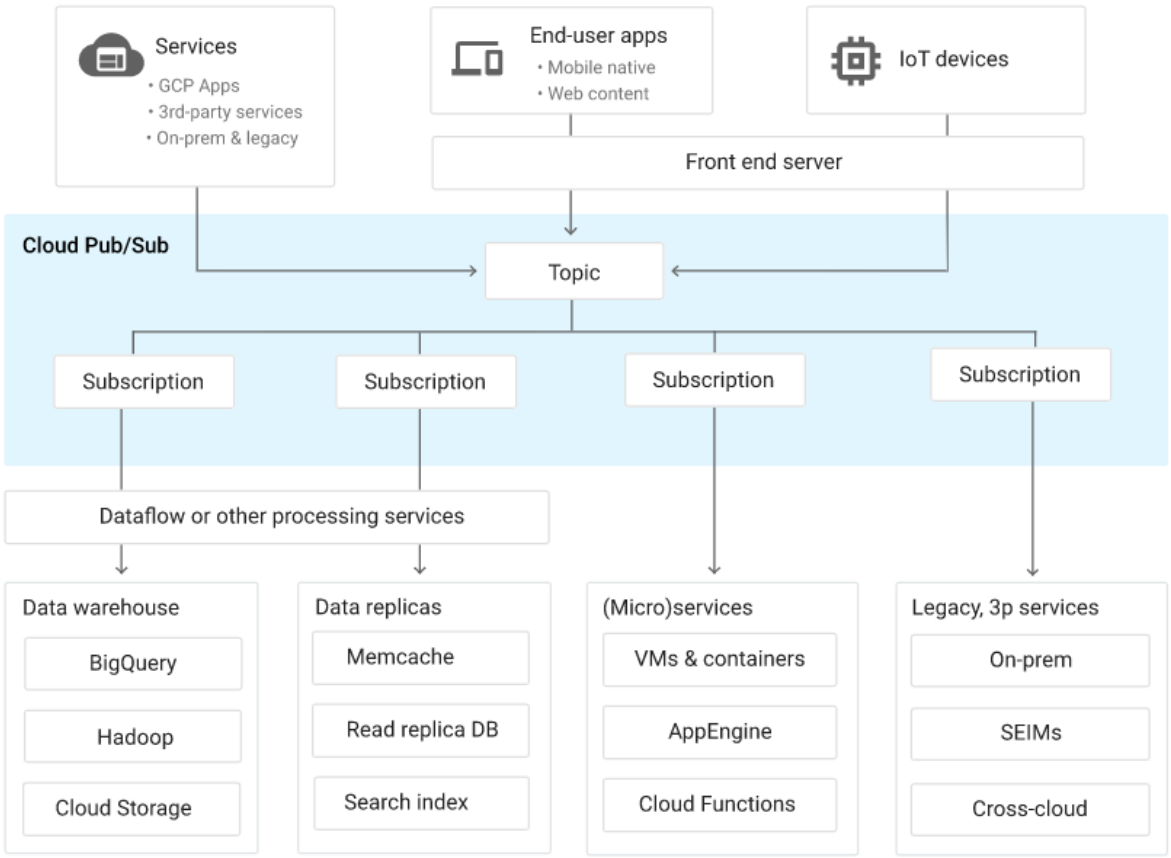
Hình ảnh từ cloud.google.com

1. Tạo một topic và gửi message đến topic đó.
2. Pub/Sub lưu lại messages trong khi subscribers tiến hành đăng ký với Subscriptions.
3. Topic gửi messages đến Subscriptions chờ được push đến từng subscriber.
4. Pub/Sub lựa chọn subscriber phù hợp để gửi mesage.
5. Subscriber gửi ack đến Pub/Sub sau mỗi lần nhận được message.

Sử dụng Pub/Sub

Publishers có thể là bất kỳ ứng dụng nào sử dụng giao thức HTTP để gửi request đến pubsub.googleapi.com. Ví dụ: Website services, desktop app, mobile app, browser events…

Tương tự, Subscribers cũng là các ứng dụng có thể tạo HTTP request đến pubsub.googleapi.com.



Hình ảnh từ cloud.google.com

Trường hợp thực tế:

- Balancing workloads in network clusters: Phân tán tác vụ cho nhiều workers cùng xử lý.
- Implementing asynchronous workflows: Giúp ứng dụng thực hiện các tác vụ theo thứ tự.
- Distributing event notifications
- Refreshing distributed caches
- Logging to multiple systems
- Data streaming from various processes or devices
- Reliability improvement

Phần bổ sung thêm https://cloud.google.com/pubsub/architecture

2.3 Stream Processing

Stream Processing là một công nghệ xử lý dữ liệu lớn. Stream yêu cầu xử lý nguồn dữ liệu liên tục, nhanh chóng trong một khoảng thời gian rất nhỏ kể từ khi nhận được dữ liệu. Stream processing được sử dụng nhiều trong các ứng dụng quản lý ví dụ: Phát hiện lỗi, hệ thống thương mại, hệ thống nhà máy…

Xử lý stream thông thường chúng ta phải đối mặt với thời gian ngắn và đặc biệt là khi dành cho mục đích phân tích. Trong thực tế khi có rất nhiều bài toán cần xử lý nhanh chóng, có độ trễ cực kỳ thấp, đơn cử như phân tích có bao nhiêu user đã login vào hệ thống trong 5 phút vừa qua.

So sánh với batch process thì batch quá trình xử lý một yêu cầu phải thông qua một tập dữ liệu lớn. Quá trình xử lý phải duyệt qua toàn bộ dữ liệu đã xảy ra, trong thực tế chúng ta phải tốn nhiều thời gian để đọc qua toàn bộ dữ liệu của một năm vừa qua, chính vì thế hệ thống xử lý không chỉ tốn một vài phút mà có thể tốn hàng giờ đồng hồ để thực thi.

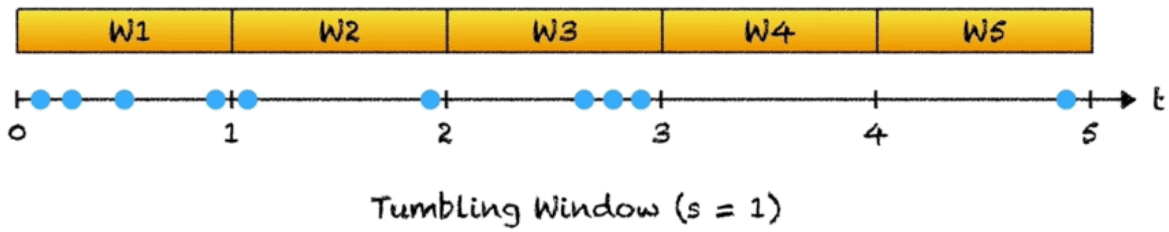
Trong thực tế, dữ liệu stream đến liên tục cũng có thế bị lỗi bởi vấn đề về internet, dữ liệu bị lỗi. Chính vì vậy để khắc phục lỗi do đọc dữ liệu từ stream thì có nhiều phương pháp để đọc dữ liệu, một trong đó Windowing by processing time là một cách cực kỳ hiệu quả.

Types of windows

Mỗi windows sẽ được xác định dựa vào khoảng thời gian xảy ra các events. Dưới đây là các loại windows phổ biến:

- Tumbling window

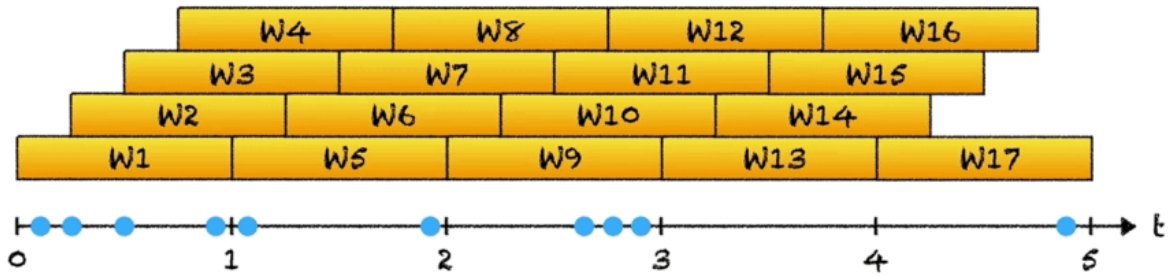
Tumbling window là loại window có độ dài cố định, mỗi event xảy ra chắc chắn sẽ nằm trong một window. Mỗi lần đọc dữ liệu, cửa sổ sẽ luôn đọc phần kế tiếp với cửa sổ hiện tại.



Tumbling Window (s = 1)

- Hopping window

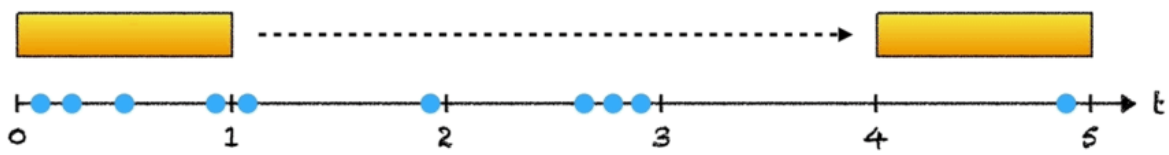
  Hopping window là loại window có độ dài cố định, nhưng mỗi lần đọc dữ liệu thì windows sẽ dịch chuyển đến các event kế tiếp nhưng vẫn overlap một phần với window trước đó.



Hopping Window (s = 1, h = 0.25)

- Sliding window

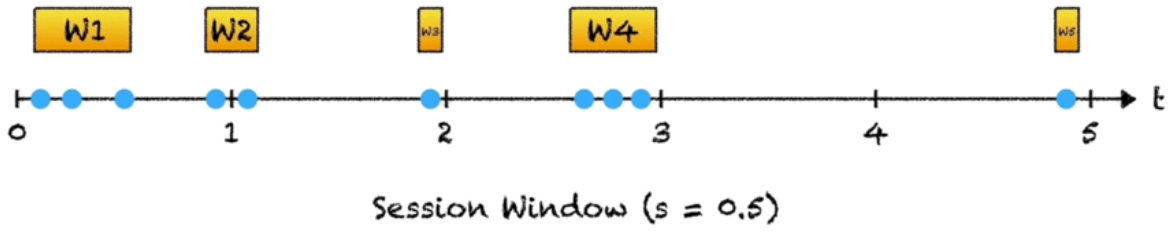
  Sliding window không phụ thuộc vào độ dài, mà phụ thuộc vào số phần tử events đang được đọc trong cửa sổ. Lần dịch chuyển tiếp theo của window sẽ tiếp tục đọc các events tiếp theo.



Sliding Window (s = 1)

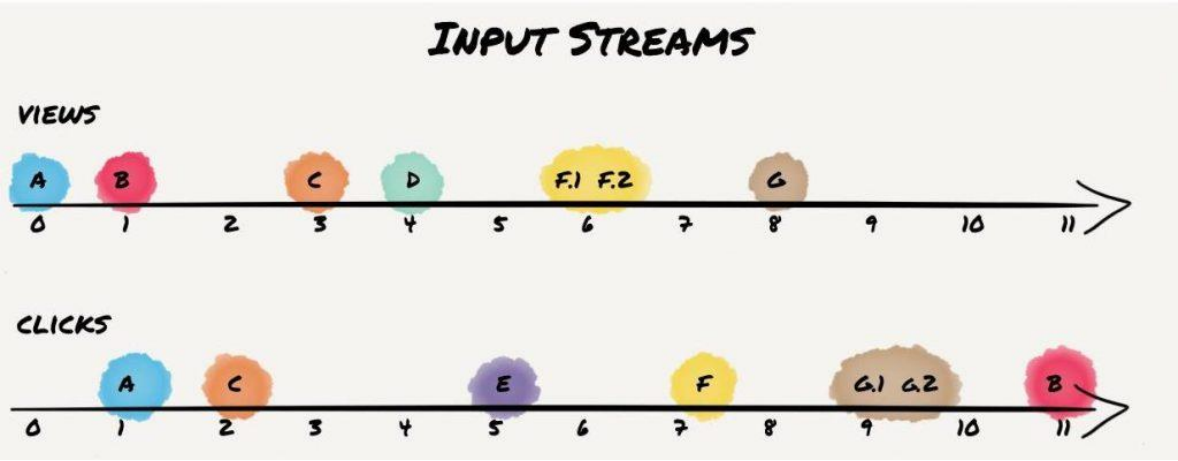
- Session window

  Không giống như các loại cửa sổ khác, session window không cố định về kích thước. Thay vào đó mục đích của session window sẽ gom nhóm các events. Ví dụ: gom nhóm các events của một user xảy ra cách nhau trong 5 phút trở lại đây.



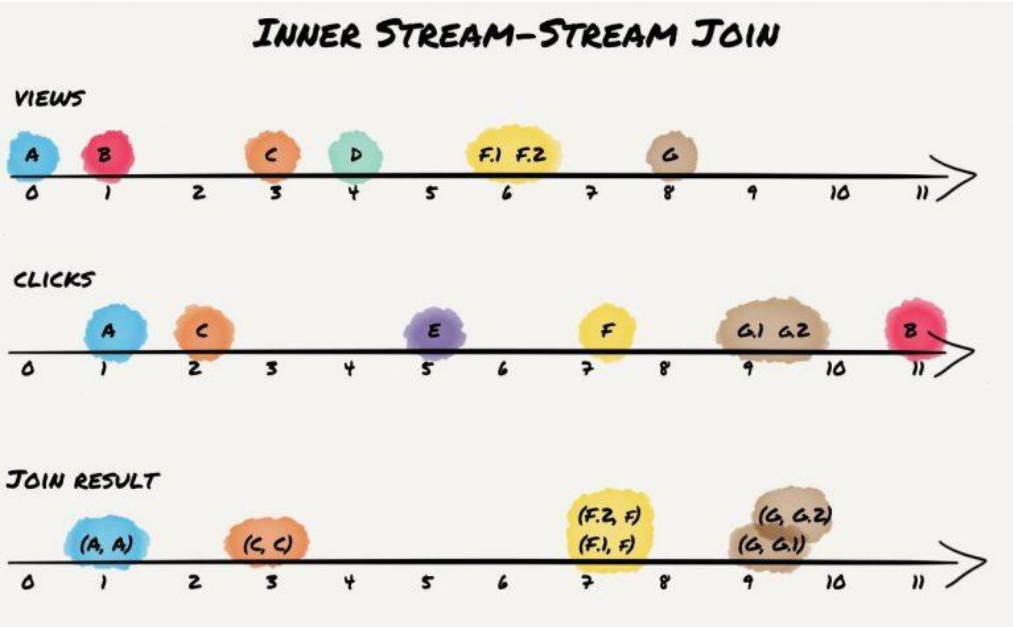
Stream Joins

Như đã giới thiệu ở trên, chúng ta đã hiểu được stream events được đọc và xử lý như thế nào. Nhưng trong thực tế có rất nhiều events khác có thể xuất hiện bất cứ lúc nào, yêu cầu chúng ta xử lý đồng thời và cùng lúc, đó thật sự là thử thách rất lớn cho việc xử lý. Sau đây là kỹ thuật stream joins: stream - stream joins, stream - table joins và table - table joins.

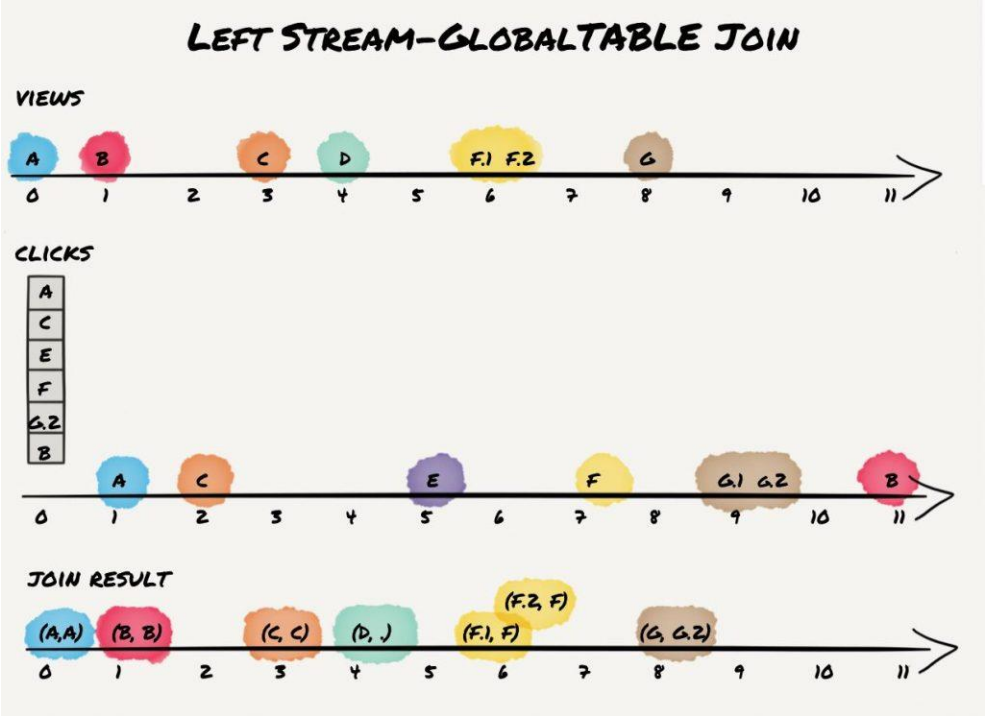


- Stream - stream join (window join)

Đây là kỹ thuật join 2 window với nhau, để hiện thực được join giữa các window yêu cầu hệ thống phải duy trì state của window trên memory.

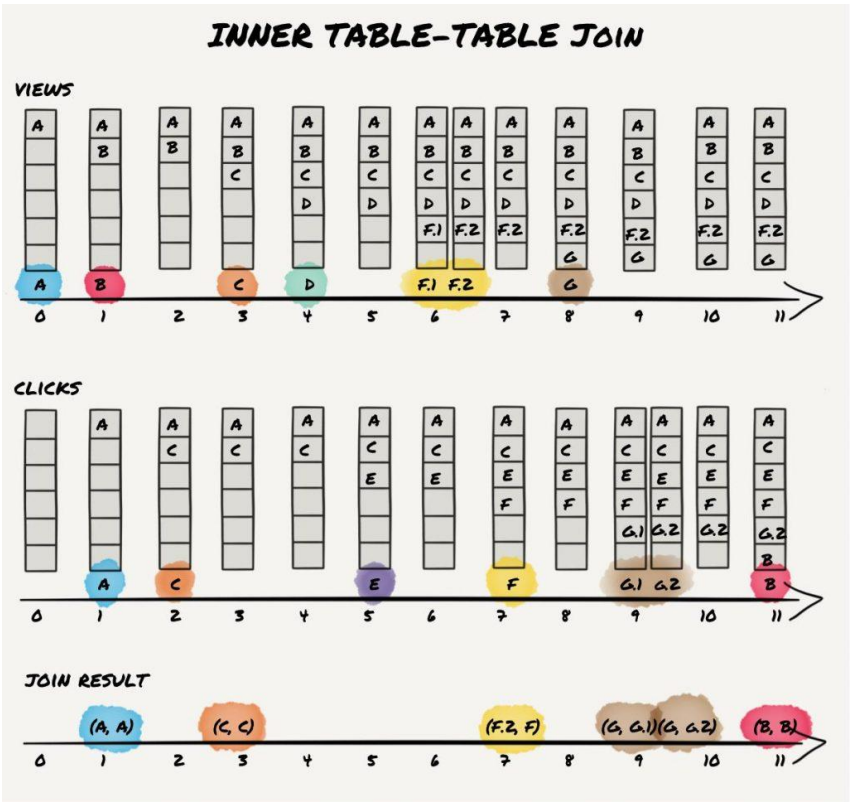


- Stream - table join (stream enrichment)
  Đây là kỹ thuật join stream và dữ liệu được lưu trữ dưới database. Để hiện thực được điều này yêu cầu hệ thống phải theo dõi events của một stream kết hợp với query dữ liệu từ database.
  Một cách tiếp cận khác là query dữ liệu từ database ngay từ khi khởi tạo. Sau đó copy dữ liệu vào một stream processor, vì thế sẽ tiết kiệm được thời gian truy vấn dữ liệu database về sau. Nhìn chung stream - table join khá tương đồng với stream - stream jon.



- Table - table join (Materialized view maintenance)

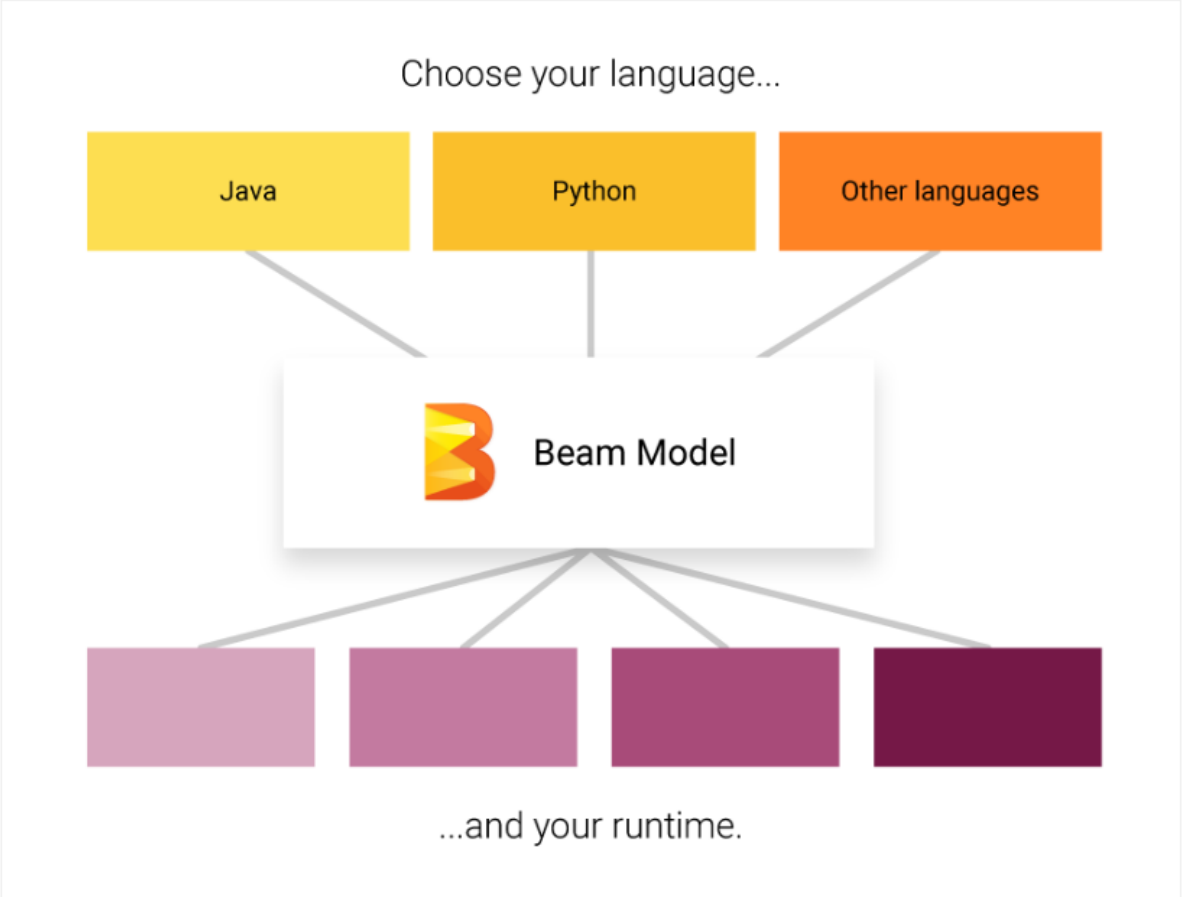
  Để hiện thực được kiểu join này, yêu cầu chúng ta phải cache dữ liệu trước khi thực hiện join để tiết kiệm chi phí truy vấn.



2.4 Apache Beam

2.4.1 Giới thiệu Beam Model

Apache Beam là một mô hình lập trình hợp nhất xử lý dữ liệu dạng batch và streaming. Cho phép hiện thực việc xử lý dữ liệu batch và streaming để chạy trên nhiều nền tảng thực thi (execution engine) khác nhau. Các nền tảng thực thi như: Flink, Spark, Google Cloud Dataflow, Samza…
Người dùng có thể hiện thực trên nhiều ngôn ngữ khác nhau như: Java, Python, Go, Scala...



2.4.2 Pipeline

Một pipeline gói gọn toàn bộ quá trình xử lý dữ liệu của bạn, từ bắt đầu đến

kết thúc. Bao gồm đọc dữ liệu từ đầu vào (input data), biến đổi (transforming) dữ liệu đó, và ghi dữ liệu ra. Tất cả các chương trình Beam đều cần tạo một Pipeline. Khi bạn tạo Pipeline, bạn cũng cần phải chỉ rõ các tùy chọn thực thi, cho biết nơi nào và làm sao để chạy Pipeline đó.

2.4.3 PCollection

Một PCollection đại diện cho một tập dữ liệu phân tán mà Beam pipeline của bạn thi hành trên đó. tập dữ liệu có thể là có giới hạn (bounded), tức là nó đến từ một nguồn dữ liệu cố định như file, hoặc không có giới hạn (*unbounded*), tức là nó đến từ một nguồn đang tiếp tục cập nhật thông qua một subscription hoặc một cơ chế nào đó.
Pipeline của bạn thường tạo ra một khởi tạo PCollection bằng cách đọc dữ liệu từ một nguồn bên ngoài, nhưng cũng có thể tạo một PCollection từ bộ nhớ bên trong chương trình.
Từ đó những PCollection là đầu vào và đầu ra cho mỗi step trong pipeline của bạn.

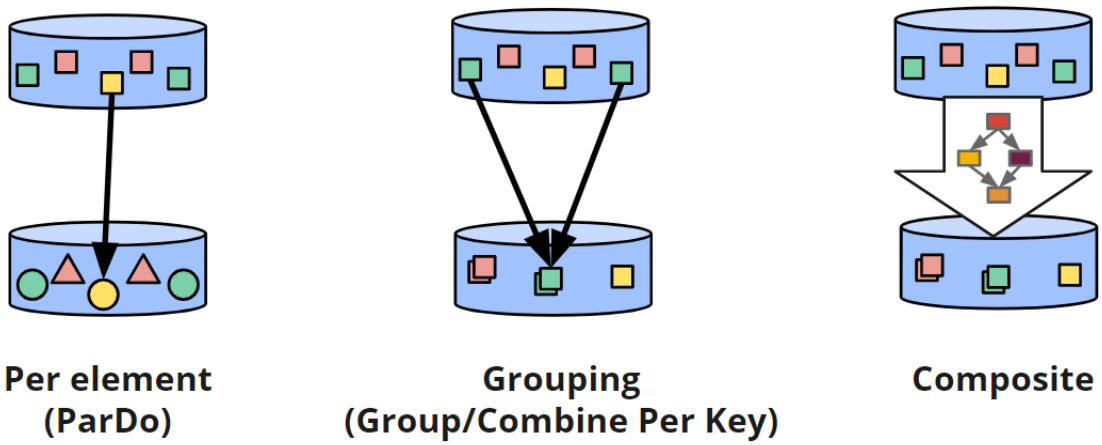
2.4.4 PTransform

Một PTransform là một phép tính biến đổi dữ liệu trong Pipeline. Đầu vào của mỗi PTransform một hoặc nhiều PCollection, sau đó PTransform áp dụng phép tính logic để biến đổi dữ liệu và xuất ra một hoặc nhiều PCollection mới, những PCollection mới sẽ là dữ liệu đầu vào cho một PTransform khác trong Pipeline.
Core Beam transforms
ParDo: cho phép áp dụng một phép tính cùng lúc lên nhiều phần tử của PCollection đầu vào
GroupByKey: cho phép chọn và tập hợp những phần từ trong PCollection có chung đặc tính, thường được gọi là từ khóa (key).
CoGroupByKey: cho phép kết hợp kết quả của 2 tập hợp GroupByKey

Combine: cho phép áp dụng những phép tính toán học trên một thuộc tính của các phần tử của PCollection, các phép toán bao gồm : sum, count, min, max...

Flatten: cho phép kết hợp giữa hai hoặc nhiều PCollection

Partition: cho phép chia 1 PCollection thành nhiều PCollection nhỏ hơn.



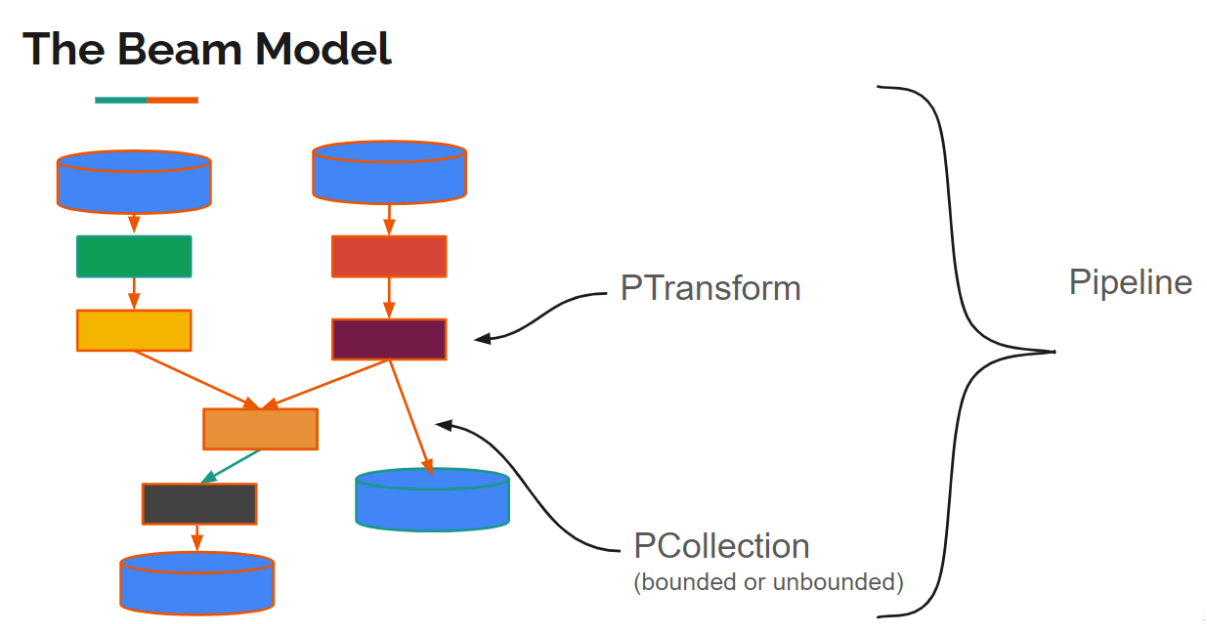
Hình: PTransform trong Beam

2.4.5 Pipeline I/O

Cung cấp APIs để đọc dữ liệu vào PCollection từ: message (streaming), tệp, cơ sở dữ liệu, google storage, hoặc các nguồn dữ liệu khác nhau.

Ví dụ ở đây project sử dụng API PubSubIO của Beam để đọc message từ Google Cloud Pub/Sub:

```
PubsubIO.readStrings().fromTopic(options.getPubsubReadTopic())
```



Hình: Beam model

2.4.6 Hiện thực Beam trên Java

```
PipelineOptions options = PipelineOptionsFactory.fromArgs(args).create();
Pipeline p = Pipeline.create(options);
PCollection<SentimentMessage> sentimentMessagePC = ReadMessage.fromPubSub(p)
    .apply(ParDo.of(DetectCompanyLabelFn.of()));

//apply window
final PCollection<SentimentMessage> windowedMessages = sentimentMessagePC.apply(
    String.join( delimiter: "_",  ...elements: String.valueOf(2), "Minutes", "Window"),
    assignWindows( windowSizeMinutes: 2,  allowLatenessHours: 48));

//write to file
windowedMessages.apply(MapElements.via(new FormatAsTextFn()));

//group by topic
windowedMessages.apply(ParDo.of(MapTopicToSentimentLabelFn.of()))
    .apply(AggregateSentimentLabels.of())
    .apply(new SinkRowToBigTable());

p.run();
```

Hình: Hiện thực Beam pipeline bằng ngôn ngữ Java

2.5 Google Cloud Dataflow

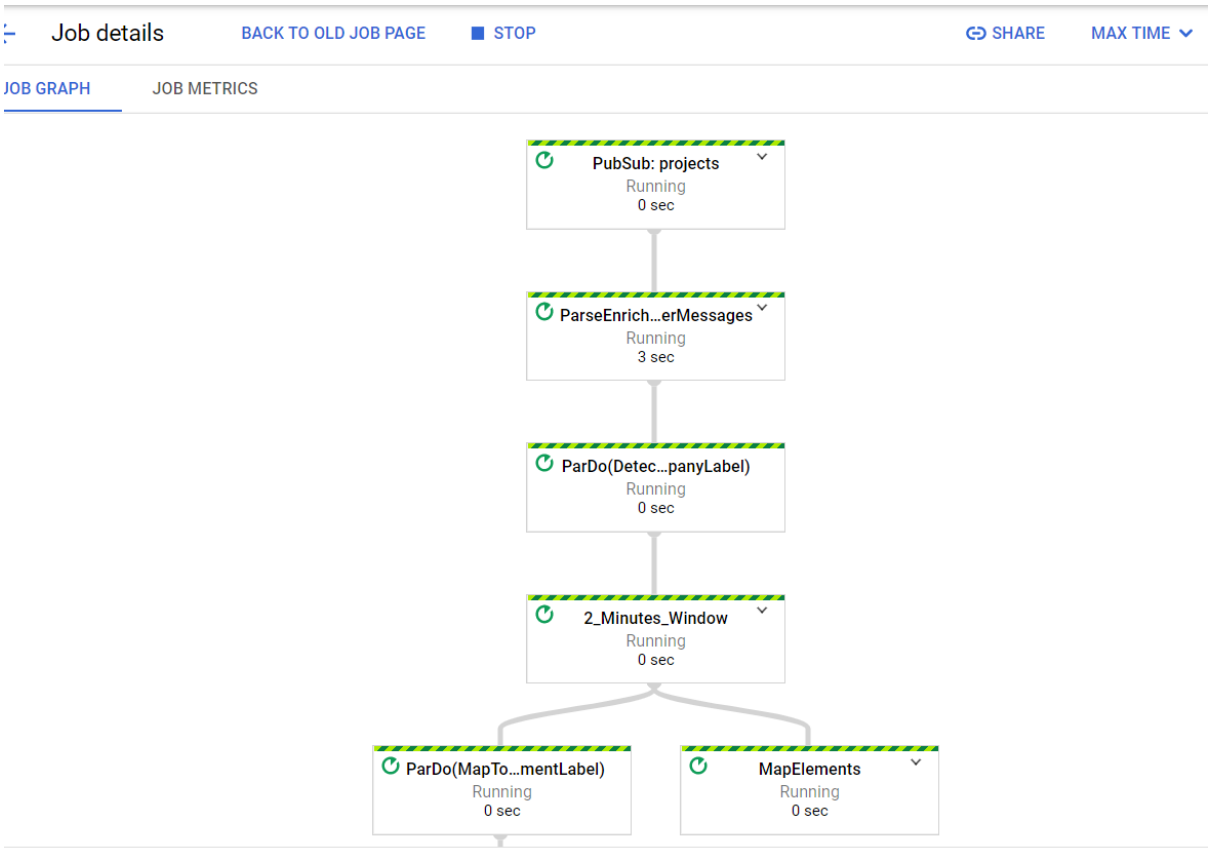
Project sử dụng Google Cloud Dataflow làm runner để chạy Beam model.

Google Cloud Dataflow là một dịch vụ được quản lý hoàn toàn để chuyển đổi và làm phong phú dữ liệu trong chế độ Stream và Batch với độ tin cậy và hiệu quả. Không cần giải pháp phức tạp. Với cách tiếp cận không cần máy chủ để cấp phép và quản lý tài nguyên, bạn có quyền truy cập vào khả năng vô hạn để giải quyết những vấn để xử lý dữ liệu lớn của bạn và chị chi trả cho những gì bạn sử dụng.

Tính năng của Cloud Dataflow:

- **Quản lý tài nguyên tự động**: Cloud Dataflow cung cấp và quản lý tài nguyên xử lý một cách tự động để giảm thiểu độ trễ và tối đa hóa việc sử dụng tài nguyên
- **Tái cân bằng công việc**: Tự động và tối ưu hóa phân vùng làm việc. Không cần phải …. Hoặc xử lý trước dữ liệu đầu vào
- **Xử lý chính xác và nhất quán**: Cung cấp built-in support để thực thi khả năng chịu lỗi phù hợp và chính xác bất kể kích thước dữ liệu, kích thước cluster, pattern xử lý hay pipeline phức tạp
- **Horizontal Auto-scaling**: tự động mở rộng tài nguyên làm việc cho kết quả tối ưu dẫn đến hiệu suất tổng thể tốt hơn.
- **Mô hình lập trình thống nhất**: Apache Beam SDK cung cấp các hoạt động giống như MapReduce, cửa sổ dữ liệu mạnh mẽ và kiểm soát độ chính xác chi tiết cho dữ liệu streaming và batch giống nhau.
- **Cải tiến theo hướng cộng đồng**: Các nhà phát triển muốn mở rộng mô hình lập trình Cloud Dataflow có thể fork và / hoặc đóng góp cho Apache Beam.

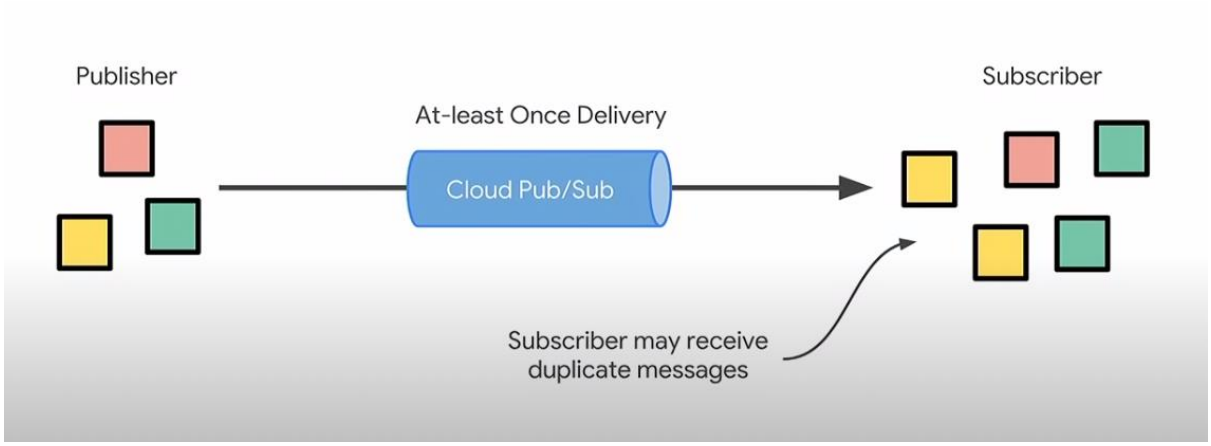
Chạy Pipeline trên nền Google Data Flow:



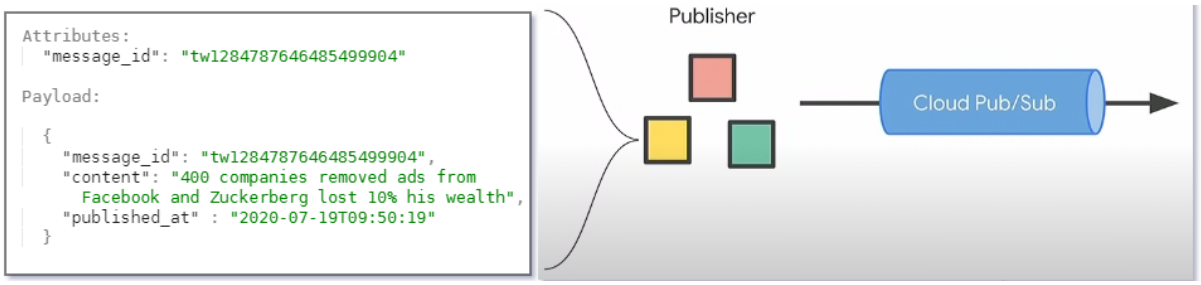
Hình: Chạy Beam pipeline trên Dataflow

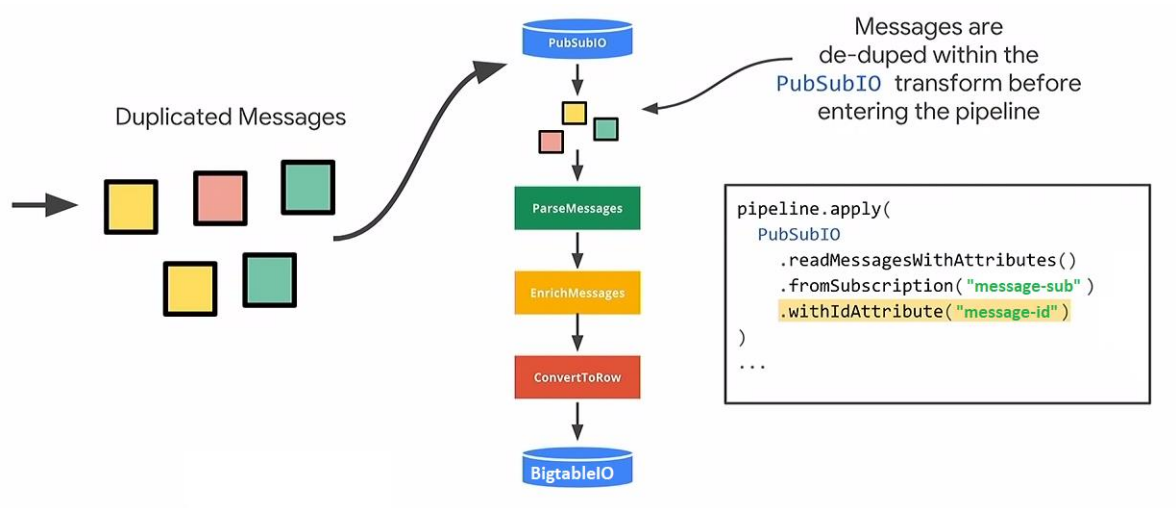
Kết hợp giữ Google Cloud Pub/Sub và Beam (Dataflow runner) để hiện thực Exactly-Once processing (xử lý chính xác một lần).

Cloud Pub/Sub đảm bảo At-least Once (xử lý ít nhất một lần)



Beam cung cấp API giúp việc kiểm tra trùng lặp (duplicate) một cách đơn giản, hiệu quả. Do đó ta dễ dàng hiện thực Exactly-Once processing (xử lý chính xác một lần) trên Streaming sử dụng Google Cloud và Apache Beam.





Ta chỉ cần chỉ ra thuộc tính Id để sử dụng tính năng này.

2.6 Bigtable

2.6.1 Giới Thiệu Bigtable

Bigtable là một hệ thống lưu trữ phân tán được thiết kế dành cho việc quản lý dữ liệu có cấu trúc, rất lớn có thể mở rộng lên tới petabytes trên cả ngàn máy chủ.

Lợi Thế của Bigtable là tính ứng dụng cao, khả năng mở rộng, hiệu suất cao và tính sẵn sàng. được sử dụng bởi hơn mười sáu ứng dụng của Google: Google Analytics, Google Finance, Orkut, Personalized Search, Writely, and

Google Earth. Những sản phẩm này dùng Bigtable từ việc chạy theo lô với throughput cao hoặc phục vụ với độ trễ thấp tới người dùng

2.6.2 Mô Hình Dữ Liệu của Bigtable

Một Bigtable là một map thưa, phân tán, nhiều chiều được sắp xếp. Map được đánh chỉ mục bởi một khóa hàng (Row key), khóa cột (Column Key), và một dấu thời gian (Timestamp); mỗi giá trị trong map là một mảng của bytes.

```
(row:string, column:string, time:int64)  =>  string
```

Hàng (Row)
Hàng là một chuỗi bất kỳ có size có thể lên tới 64KB) . Việc đọc hay ghi vào 1 dòng duy nhất là nguyên tử (atomic). Bigtable sắp xếp dữ liệu theo khóa cột (Row key) của hàng theo từ điển.
Cột (Column)
Khóa Cột (Column key) được nhóm lại theo Cột gia đình (Column Families). Dữ Liệu trong cột gia đình đều thường chung một kiểu data. Và Cột gia đình phải được tạo trước khi dữ liệu được chứa ở cột nào trong gia đình đó.
Dấu Thời Gian (Timestamp)
Mỗi ô trong Bigtable có chứa nhiều phiên bản của dữ liệu. Những phiên bản đó được đánh chỉ mục theo dấu thời gian. Bigtable dấu thời gian là kiểu số nguyên 64 bit và biểu diễn cho thời gian thực ở millisecond. Bigtable dấu thời gian được tự gán hoặc do người dùng gắn một rõ ràng bằng mã.

2.6.3 Áp Dụng Vào Dự Án

BigTable được áp dụng vào dự án vì những lợi thế không thể bàn cãi mà nhóm cũng đã trình bày ở trên.
Thứ 1 Google Bigtable với lợi thế là là tính ứng dụng cao, khả năng mở rộng, hiệu suất cao và tính sẵn sàng, có thể chịu tải với đầu ra từ hệ thống xử lý theo thời gian thực.

Thứ 2 Bigtable nằm trong bộ Google Cloud Service, dễ dàng để tích hợp với dịch vụ khác của Google, ở đây là Dataflow.
Thứ 3 Google thích hợp cho dữ liệu chuỗi thời gian (time serie data). Theo như phần mô hình dữ liệu, Bigtable sắp xếp dữ liệu theo khóa hàng (Row key) theo từ điển. Và Bigtable cho chúng ta 2 cách để lấy dữ liệu theo 2 cách

- Lấy một dòng bằng cách xác định khóa dòng(row key).
- Lấy nhiều dòng bằng cách truyền một dãy cái khóa dòng (row key).

Thiết Kế Dữ Liệu

Thiết kế Khóa Dòng: ID, Id của công ty (Company Id) + “#” + Nhãn Thời Gian (Timestamps)

Thiết kế Cột: Bảng sẽ có một Cột gia đình. Trong cột gia đình có 3 cột

- Cột cho sentiment positive
- Cột cho sentiment negative
- Cột cho sentiment neutral

Row key	Column data		
000001#2020071901124501001	sentiment:positive:0	sentiment:negative:2	sentiment:neutral:0
000002#2020071901124501001	sentiment:positive:1	sentiment:negative:0	sentiment:neutral:0

III. Nguồn dữ liệu

Streaming data từ Twitter

Nền tảng twitter cung cấp rất nhiều API, công cụ giúp cho nhà phát triển tận dụng được sức mạnh của nguồn thông tin lớn, mạng kết nối thời gian thực.

Đối với twitter APIs bao gồm 2 loại như REST APIs và Streaming APIs. Dưới đây là danh sách một số API thường được sử dụng nhất:

- Create, retrieve, like
- Search, get users
- Retrieve trends
- Advances search
- Real time tweets and public account information
- Twitter data

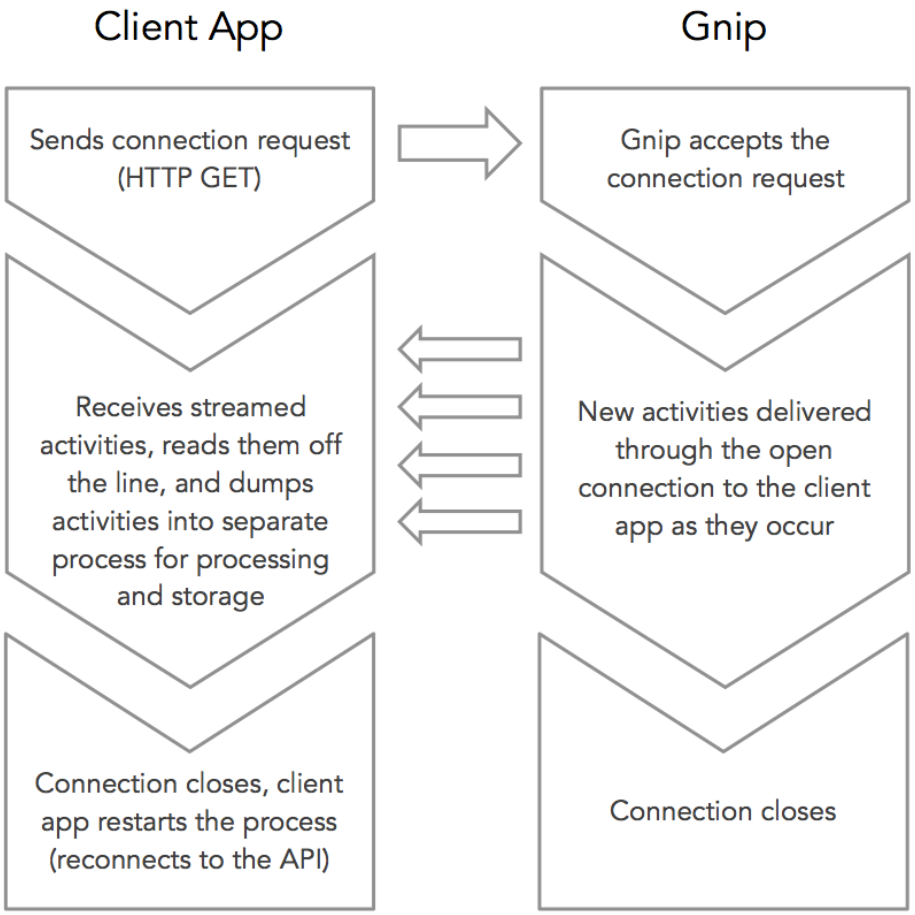
Như vậy, Trong dự án này nhóm sẽ sử dụng Streaming API để lấy dữ liệu từ twitter. Sau đây nhóm xin giới thiệu chi tiết hơn về Streaming APIs.

Streaming API

Giao thức Streaming HTTP được ứng dụng trong một số loại streaming của twitter như: PowerTrack, Volume and Replay. Khác với kỹ thuật truyền tải dữ liệu trong batching, streaming không cần phải gửi request nhiều lần đến server, thay vào đó kỹ thuật streaming sẽ mở một connection giữa App và API. Như vậy, bất cứ khi nào API có cập nhập dữ liệu mới sẽ đồng thời gửi đến App.

Chúng ta dễ thấy rằng, streaming có thể xử lý được một lưu lượng (throughput) cao bởi vì độ trễ trong gửi/nhận dữ liệu là cực kỳ thấp.

Thông qua sơ đồ bên dưới để làm rõ hơn mối quan hệ giữa App và API:



Hình ảnh từ developer.twitter.com

Giải thích sơ đồ:

1. Khởi tạo kết nối: App sẽ gửi request đến API để yêu cầu mở một kết nối, trong mô hình này thì 2 bên sẽ mở một pipeline để truyền tải dữ liệu. Kết nối chỉ tồn tại trong một khoảng thời gian nhất định.
2. Gửi dữ liệu: khi có một action cụ thể xảy ra trên API, thì dữ liệu đó sẽ được gửi thông qua pipeline. App có thể đọc và xử lý nhiều loại dữ liệu khác nhau.
3. Ngắt kết nối: Bất cứ lúc nào kết nối bị ngắt thì khả năng dữ liệu bị mất trong quá trình đó là rất cao, chính vì thế để tránh mất dữ liệu sẽ có cơ chế để App khởi tạo lại kết nối trong quá trình đó (ví dụ: Replay …).

Xử lý nhận dữ liệu

Sau khi pipeline được khởi tạo, thì App bắt đầu nhận dữ liệu dưới 3 dạng như sau: JSON, system message và blank lines.
Về nguyên lý, nguồn dữ liệu là không bao giờ có điểm dừng, nên cách xử lý dữ liệu nhận được phải liên tục (read line by line). Tránh trường hợp chờ đợi toàn bộ dữ liệu được gửi về rồi mới xử lý.

Xử lý mất kết nối

Tình trạng mất kết nối có thể xảy ra tại bất kỳ thời điểm nào, để nhận lại được lượng dữ liệu bị mất. Twitter cung cấp nhiều giải pháp như: Redundant connections, Backfill, Replay.

- Redundant connections: tổng hợp stream từ nhiều server để lấy missing data.
- Backfill: kết nối lại sau mỗi 5 phút đồng thời gửi request để lấy missing data.
- Replay: Sử dụng một stream khác để gửi lại missing data trong vòng 5 ngày trước đó.

IV. Kết Quả

Trong hoàn cảnh bùng nổ dữ liệu liên tục như hiện tại, nhiều nguồn dữ liệu mới xuất hiện với độ đa dạng và kích thích lớn, nếu tích hợp và khai thác kịp thời sẽ mang lại giá trị kinh tế vô cùng lớn. Vì vậy, chọn Google Cloud Platform và Apache Beam trong việc xây dựng hệ thống xử lý dữ liệu liên tục theo thời gian mang lại những ưu điểm:

1. Dễ dàng tích hợp với bất kỳ nguồn dữ liệu mới nào do APIs được cung cấp sẵn từ Apache Beam.
2. Hệ thống được triển khai trên nền tảng Google cloud nên đảm bảo sự ổn định và dễ dàng quản lý, không tốn nhiều chi phí cho đội ngũ IT
3. Hỗ trợ nhiều ngôn ngữ lập trình.
4. Việc chuyển đổi nền tảng đơn giản. Apache Beam có thể chạy trên nhiều nền tảng khác nhau (Không nhất thiết phải là Google DataFlow).

Dự án đã triển xây dựng một kiến trúc hệ thống dựa trên nền tảng Google Cloud Platform. Đáp ứng khả năng xử lý dữ liệu theo thời gian thực. Dữ liệu streaming, dữ liệu lớn từ nguồn mạng xã hội. Hệ thống có khả năng dễ mở rộng dựa trên nền tảng Google Cloud. Có khả năng xử lý dữ liệu từ nhiều nguồn khác nhau.

Hệ thống có khả năng tích hợp với nhiều nền tảng, nhiều công cụ xử lý dữ liệu hiện đại. Ta có thể sử dụng các thư viện, model mã nguồn mở về xử lý ngôn ngữ tự nhiên. Để rút ngắn thời gian hiện thực. Sau này nếu có để nguồn lực, có thể tự thiết kế model để tăng tính chính xác cho từng mục đích sử dụng khác nhau.

Nền tảng xử lý, phân tích streaming theo thời gian thực hiện đang được ứng dụng rộng rãi. Và phát triển rất nhanh trong thời gian gần đây. Do những giá trị to lớn nó đem lại với việc xử lý dữ liệu lớn trong thời gian thực. Ứng dụng được trong rất nhiều lĩnh vực như: game, quảng cáo online. công nghệ tài chính, IOT, shopping,...

Tài liệu tham khảo

1. Akidau, Tyler;Chernyak, Slava;Lax, Reuven. Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing, O'Reilly Media, 2018
2. Martin Kleppmann. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, 2017
3. Apache Beam, https://beam.apache.org
4. Google Dataflow, https://cloud.google.com/dataflow
5. Google Bigtable, https://cloud.google.com/bigtable
6. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data, Google Inc., 2006
7. Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J.Fernandez-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, Sam Whittle. The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing, Google Inc., 2015
8. Woodie, Alex. Apache Beam's Ambitious Goal: Unify Big Data Development, Datanami, 2016
9. Google Team. Data Processing with Apache Beam and Google Cloud Dataflow, 2016