linkedin.com

Google Docs High-Level System design

Murat Atak

9-12 minutes

Multiple users work on the same copy of the document. The document could be text, rich text, painting, drawing. Let multiple users collaborate on the same copy of the document.

Functional Requirement

- Users can change a document at the same time without any conflict
- Users can give permission(view, editor) for a document
- Users can make import and export a document
- A user can add a comment to the document
- Could be a notification (Email, GCM notification)

Design Constraint

- **Concurrency**: A lot of people are working on the same document
- **Latency**: People are working in different places and they connect throughout the internet, so there is a latency between each and every know clients or users when they are sharing the same document.
- 200M User, Daily active % 10 = 20M User
- Average 1 MB usage ~ 1MB x 20M user = 20 BP per day usage

  Google Docs users get a lot of storage space with their accounts, but it's not unlimited. Each account can have up to:
- 5,000 documents of up to 500 kilobytes each
- 1,000 spreadsheets of up to 1 megabyte each
- 5,000 presentations of up to 10 megabytes each

First, we need to understand some logic that a lot of users can change a document at the same time without any conflict.

Operational Transformation

Operational Transformations as an algorithm for automatic conflict resolution.

OT, at its core, is an optimistic concurrency control mechanism. It allows two editors to modify the same section of a document at the same time without conflict. Or rather, it provides a mechanism nor sanely resolving those conflicts so that neither user intervention nor locking becomes necessary.

Our operation like :

INSERT(char, position) like (A, 1)

DELETE(char, position)

UPDATE(char, new char, position)

RETAIN(char, position)

is the basic operation. But there can be numerous operations in Google Docs. Change font, color, font family, font size, etc ...

Logic

The basic idea of OT can be illustrated by using a simple text editing scenario as follows. Given a text document with a string "abc" replicated at two collaborating sites, and two concurrent operations:

1. O1 = Insert[0, "x"] (to insert character "x" at position "0")
2. O2 = Delete[2, "c"] (to delete the character "c" at position "2")

generated by two users at collaborating sites 1 and 2, respectively.

Suppose the two operations are executed in the order of O1 and O2 (at site 1).

After executing O1, the document becomes "xabc".

To execute O2 after O1, O2 must be transformed against O1 to become: O2' = Delete[3, "c"], whose positional parameter is incremented by one due to the insertion of one character "x" by O1.

Executing O2' on "xabc" deletes the correct character "c" and the document becomes "xab". However, if O2 is executed without transformation, it incorrectly deletes character "b" rather than "c".

The basic idea of OT is to transform (or adjust) the parameters of an editing operation according to the effects of previously executed concurrent operations so that the transformed operation can achieve the correct effect and maintain document consistency.

No alt text provided for this image

Another example:

No alt text provided for this image

OT is not stable/reliable in the face of conflicts. Too many messages to handle on the server-side

High-Level Design

No alt text provided for this image

Flow

We have 3 clients.

API GATEWAY :

Is an API management tool that sits between a client and collection of backend service.

An API Gateway takes all the API calls from clients, then routes them to the appropriate microservice with request routing, composition, and protocol translation.

In this case, clients interact with the API GATEWAY for any operation to be performed. The gateway can connect first Authenticate and permission service whether understanding you have a permit or not in the current documents.

Advantages

- Single entry point & API composition
- Security
- Dynamic Service Discovery
- Service Partition HIdden

- Hiding MS
- Circuit Breaking

Comments Service

We might have a lot of comments about sharing documents. We can choose a line or position or a section and we can put a comment.

The dataType could be K-V.

K: document_id, V: Json object relevant to the comment. {line:1, position: 123, comment: “hey”, user_id: 123}

And we can use a NoSQL DB for comment service. Because we have a lot of huge data and we don’t need serious relations. K-V is enough. (Mongo, DynamoDB, etc.)

Authentication Service

Provides our permission. Could be view just to the current document or editor. We can use this information User and User Authorization DB.

Export-Import Document Service

If you want to export or import a document this service will be in handle over here. Connecting NoSQL and RDBMS DB. It could need comments while export documents.

We have RDBMS DB. Because we need to be consistent and we need ACID property there.

Like users, even the Document ID and they refer to the document and all the different resources.

If we have some change in a document we need **ACID** (Atomicity, Consistency, Isolation, Durability) or a transactional way to ensure change is completed. It didn’t complete this transaction will be very confusing matching to the diff for Operational Transformation.

Like we want to back former date documents. We make sure this transaction is complete. And we can continue some updates on the former documents. Otherwise, we can not update these former documents.

Time Series DB

Time-series data is a sequence of data points collected over time intervals, giving us the ability to track changes over time. Time-series data can track changes over milliseconds, days, or even years.

There are many other kinds of time-series data, but regardless of the scenario or use case, all time-series datasets have 3 things in common:

1. The data that arrives is almost always recorded as a new entry
2. The data typically arrives in time order
3. Time is a primary axis (time-intervals can be either regular or irregular)

In other words, time-series data workloads are generally “append-only.”

Simply put: time-series datasets track changes to the overall system as INSERTs, not UPDATEs.

Some DBs: InfluxDB, ElastichSearch

We have 3 clients and one sharing document. A user wants to reach the former 3 days ago document.

You always make sure that there is a state which is maintained on the server because this client goes offline anytime, other clients can diverge at any time. We need to make sure we are actually maintaining the state of the document and all the histories or the updates are kind of saved in someplace. In the Time Series DB, we can get a timeline of operations, and also we can track what user has modified what particular part of the document.

No alt text provided for this image

Websockets - TCP connection

1-) Real-Time

2-) Light Weight

HTTP, Long Pooling, AJAX not efficient. Because takes little time.

We don’t need to ask the server ‘Are there any changes.’

We just send and receive.

When the first time a user opens the document, we actually hit the

API gateway as far as if the has permission, the user will get a session established once he/she goes to the edit mode and he gets Websocket. WE can use **NodeJs WebSocket** which is established from the browser to the server always on so now the user can efficiently keep on sending small operations as when the user wants then receive the updates from other clients from the server immediately as soon as they broadcast it.

We are using the Operational Transformation design in our case. We are expecting the message size to be very small but numerous events so Websockets are best for those kinds of operations even in Node Js hands as well.

Operation Queue

All these operations should be ordered in a **queue** because 2 guys have to send the same operation on the same line or same char at the same time you still or the service should still prioritize one over the other so we need to it's better to put it in the queue and then keep on in just as in when the operations from all the clients for that particular document keeps coming in.

Session or Authorative MS

We have an operations queue here where all the operations are in the Queue over here and the **session server** will keep on interesting both operations and it keeps its own state of the document. That acts as a single source of truth.

Time Series DB

Also, it keeps on saving a copy of the history of our operations into the **Time Series DB**. Want to deliver it back to a particular version or if you want to check who edited this particular line or this particular word we can easily get that information Time Series DB. We are using Time Series DB but if you want you can actually use SQL or another like Tree kind of representation a graph representation of how this particular line changes or document changed you can actually use graph Db as well.

Cloud Storage

If you want to convert a document to PDF or HTML or any format or if you want to share a link obviously you download it you need to talk bout it to cloud storage and that link will be provided to the user via email for them to download so you can use the cloud storage there or save that in and exporting document.

Microservices Structure

- Simplicty / modularity
- Faster to develop & understand
- Different tech stack
- Scaling easy

Sources :

High-Level System Design

https://www.youtube.com/watch?v=2auwirNBvGg&ab_channel=TechDummiesNarendraL

https://www.youtube.com/watch?v=U2lVmSIDJhg&ab_channel=TechDummiesNarendraL

Detail information about the OT :

https://medium.com/coinmonks/operational-transformations-as-an-algorithm-for-automatic-conflict-resolution-3bf8920ea447

https://en.wikipedia.org/wiki/Operational_transformation

Time Series DB :

https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/

API GATEWAY

https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do#:~:text=An%20API%20gateway%20is%20an,and%20return%20the%20appropriate