#ASLI ENGINEERING

GitHub Outage Downtime due to ALTER TABLE

SWIPE

BY

ARPIT BHAYANI

Dissecting GitHub Outage

Handling large schema migrations

November 2021. GitHub experienced major outage that affecked their core services

GitHub actions, API requests, Codespaces, Git Operations, Issues, Packages, WebHooks, Pull requests

The outage was due Large Schema Migration

Insight 1: Schema migration can take weeks to complete

Imagine, ALTER TABLE command taking weeks to complete

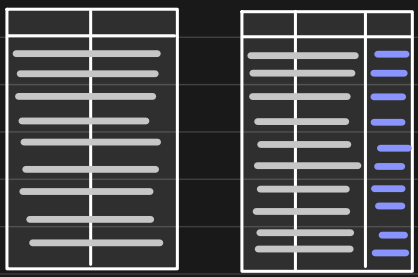
Why schema migration takes so long?

ALTER TABLE is too slow for large tables

Both COPY and INPLACE

Data to be locked and copied with new schema

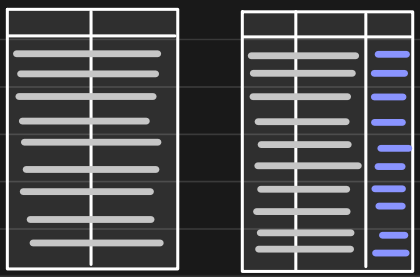
* The key concern is table locking during alteration



lock and rewrite the table with the new schema

Larger the table, larger time it will take for migration

Insight 2: last step of schema migration is RENAME



How migration happens:

1. New ghost table is created
2. schema is altered
3. data is copied
4. table is renamed

```
RENAME TABLE repositories_1 TO repositories;
```

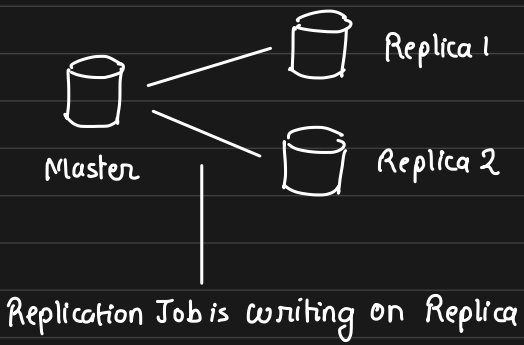
↑

During this step MySQL read replicas entered deadlock

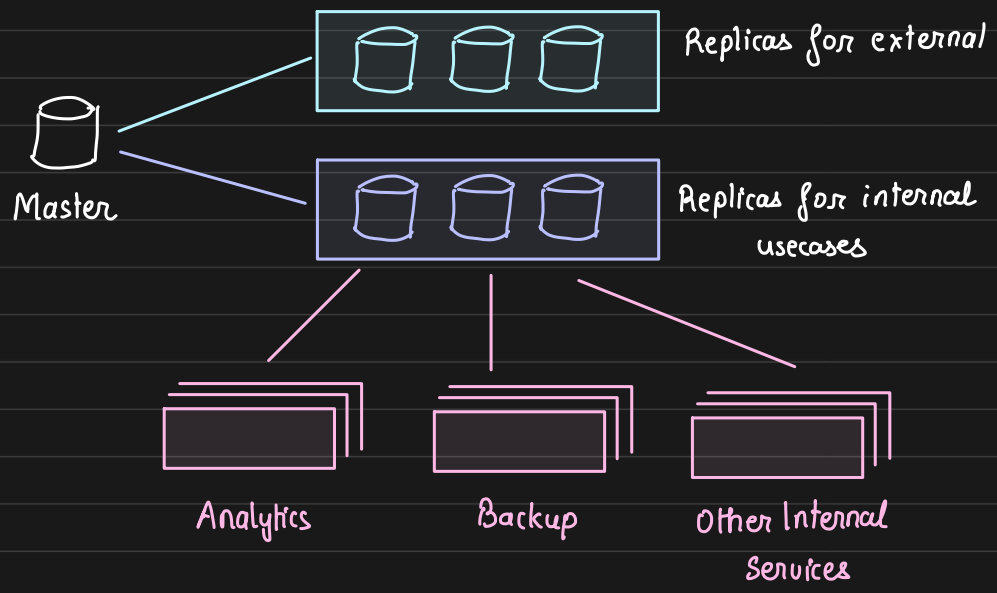
Insight 3: Deadlock on replicas

It is really strange to think that replicas can be in deadlock!

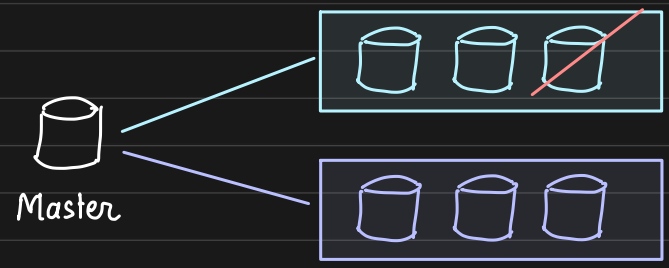
Deadlock → locks → write, But who's writing on replica?



Insight 4: Separate fleet of replicas for internal traffic



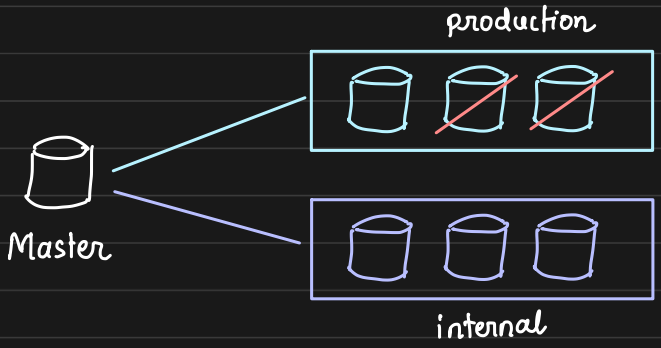
Insight 5: Database failures cascade



if one replica goes down, the load gets on other replicas and they crash

if other replicas cannot bear the load,
... failures cascades

How to mitigate such an outage?



Because there are not enough replicas to bear the load, to mitigate the issue we have to add more replicas.

But because spinning up new replica takes time, what can be a quick hack here?

> Promote the replica of internal traffic to production

But it did not work....

Because of heavy load,

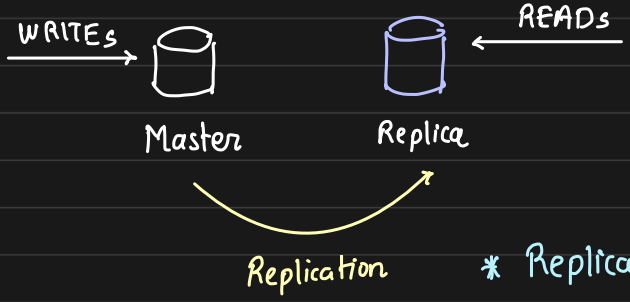
- the newly added replicas also crashed
- the crash replicas recovered & crashed again

CRASH-RECOVERY LOOP

Prioritizing Data Integrity over Availability

Data integrity was at stake purely because the crash could corrupt the data

But..... aren't these read replicas? Whose writing?.....



Data on the replica could corrupt because the DB could crash while replicating the data from the master

* Replicas are in crash-recovery loop

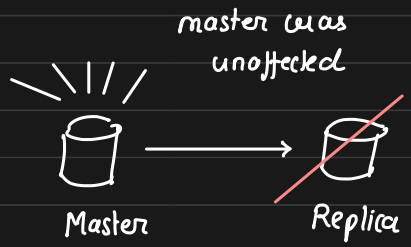
So, how to handle this?

- let the replica crash
- let the crashed replica not handle any traffic
- let the replica complete its schema migration
- once completely recovered, add it to handle production

and slowly, let the entire capacity restore

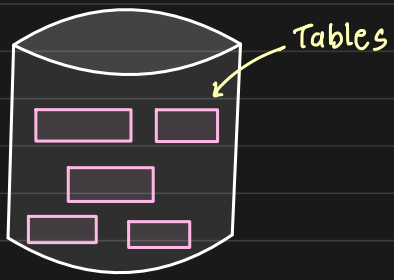
* A key detail to note here,

During this entire outage, the WRITEs were all healthy and fine!



Long Term Fix

- Continue to prioritize Functional Partition



all tables in one DB

create multiple DBs each holding a few tables [vertical partitioning]

Advantages:

- migrations can be run on a canary [small Database server] before prod
- if one DB goes down it will NOT affect others