Deploy your own replication system with Wal2json

PGCONF.EU 2019

Mai PENG

17/10/2019

webedia.
MOVIES WITH PASSION

Hello

Mai Peng , DBA @webedia movies pro
Data operations : migrations
Detect bottleneck latency
Find solutions for Fast Data Processing

maily.peng@webedia-group.com

Twitter:@mlypeng

webedia.
MOVIES WITH PASSION

Webedia Movies

- ➔ WEBEDIA MOVIES is the first digital platform dedicated to cinema and series in France and 4 other countries : 14 millions visitors per month.
- ➔ Social media interactions is one of the new marketing strategy
- ➔ More interactive means : more rates on media, more reviews, links to third social media like Facebook, Instagram.
- ➔ Convert our previous social platform into a more transactional architecture, speed up the response of any interaction.

webedia.
MOVIES WITH PASSION

Why this topic ?

- ➔ Allocine is using this replication stack after months of issues
- ➔ Few people use Wal2Json : it's an opportunity to exchange about our project
- ➔ Now the solution is deployed on all over our movies websites
- ➔ It’s relevant to share our feedback and discuss!

webedia.
MOVIES WITH PASSION

Agenda

Problem statement

Capture the data Change as Near to REAL-TIME

Message Queuing: Write data events quickly to ElasticSearch

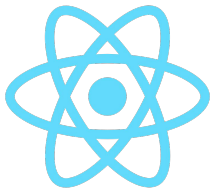
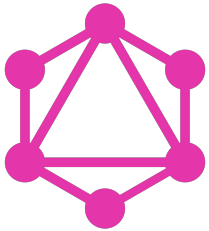
It works !

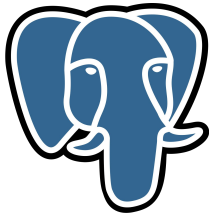
Conclusion

webedia
MOVIES WITH PASSION

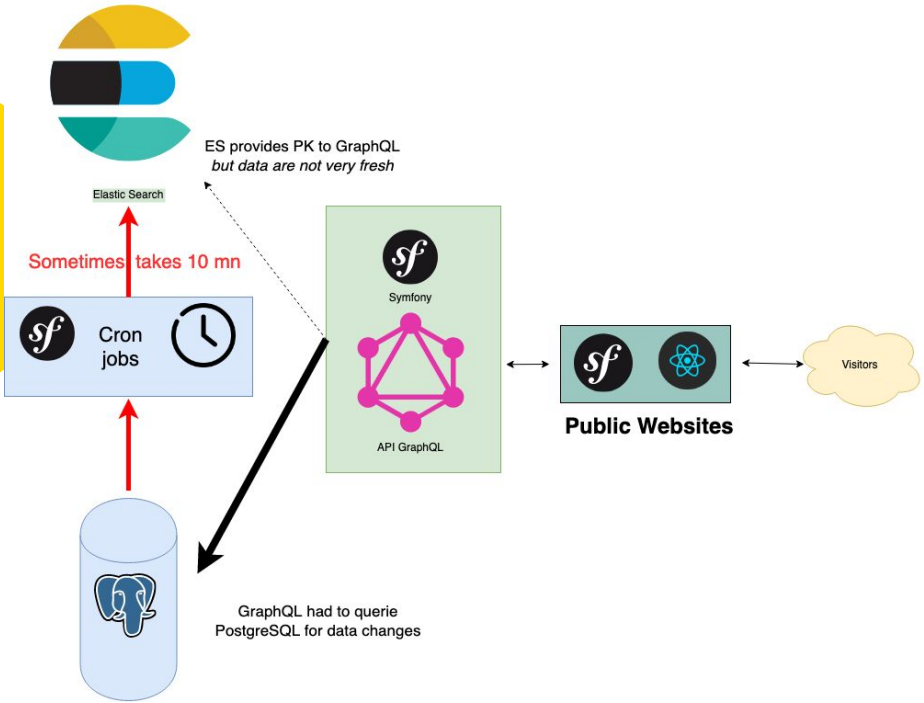
Webedia Movies : the tech

- A server side rendered Website **on premise(not in cloud)**
  Built in **Symfony** and **React**
  - Consuming a GraphQL api written in Symfony
    - Using data from PostgreSQL database
    - Using Redis for caching
    - Using Elastic Search for **filtering and ordering**



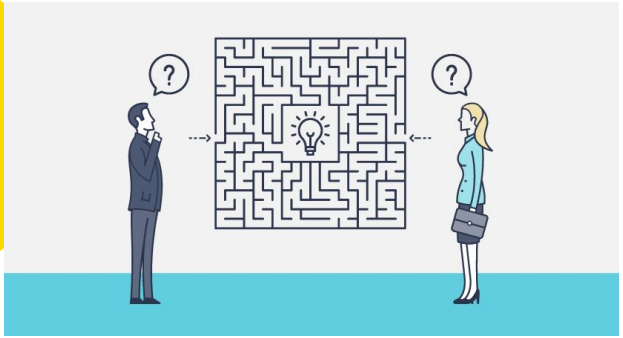

Issues: Time consuming and load



- ➔ PG and ES are not sync : new data written on PG are replicated to ES **in minutes**.
- ➔ To much queries for new data on PG
- ➔ Big transactions generate LOADS, long queries and locks on database

webedia
MOVIES WITH PASSION

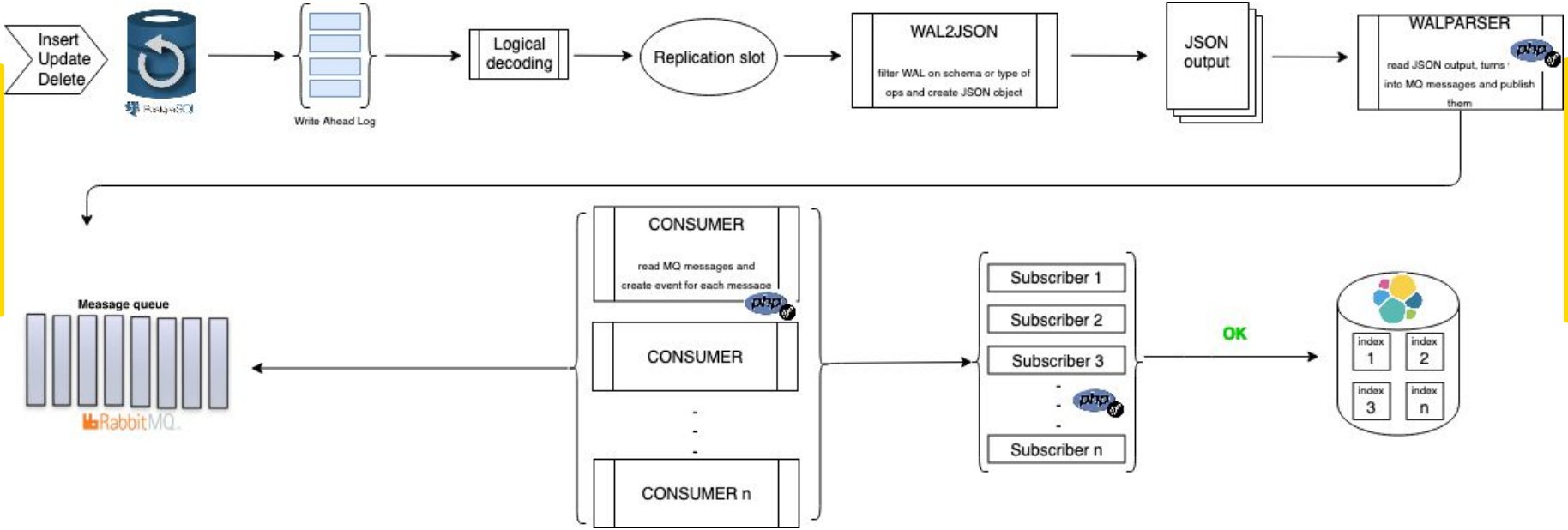
Constraints



- ➔ Every user interactions has to be written to pg and to ES in milliseconds
- ➔ We do not want performance overhead on our database: less queries, or only queries with pk=> use indexes
- ➔ Make the replication between PG to ES the more transactional as possible.
- ➔ Keep PostgreSQL and ElasticSearch in sync for coherency

webedia
MOVIES WITH PASSION

Whole system not WAL system



webedia
MOVIES WITH PASSION

Agenda

Problem statement

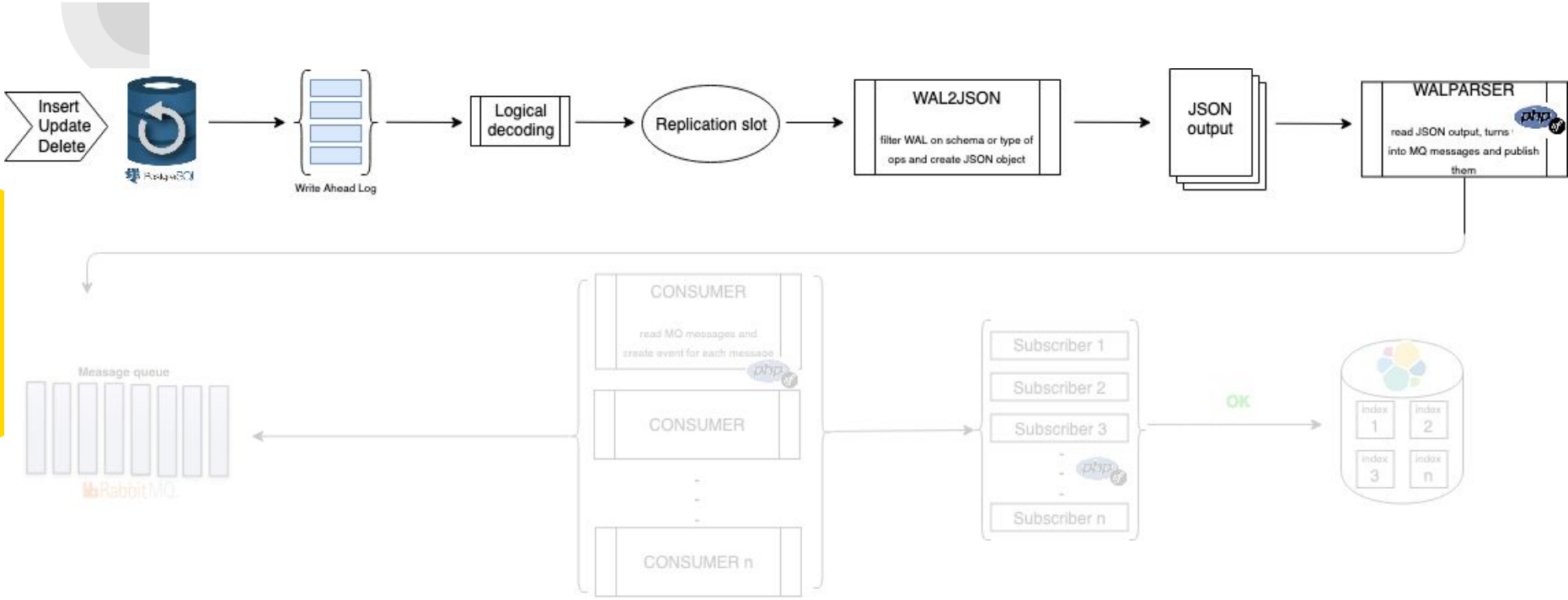
Capture the data Change as Near to REAL-TIME

Message Queuing: Write data events quickly to ElasticSearch

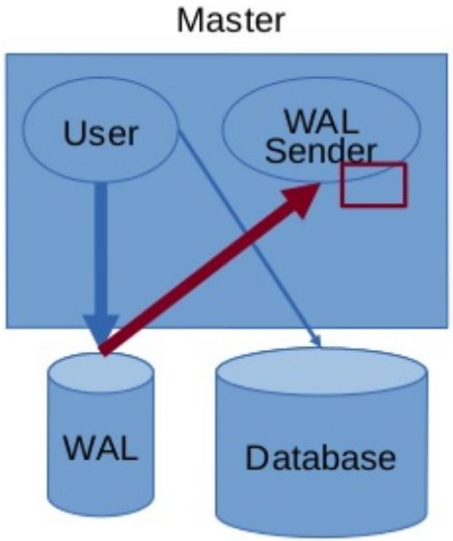
It works !!!!!!

Conclusion

webedia
MOVIES WITH PASSION



Logical decoding basis



- ➔ Logical Decoding added in PostgreSQL 9.4
- ➔ Extracts information from Write-Ahead Log into logical changes (INSERT/UPDATE/DELETE)
- ➔ Concurrent transactions are decoded in **commit order**
- ➔ Achieved by creating a replication slot with a plugin to produce data for a receiver

Logical Replication slot

- ➔ A “pipe” that give a continuous stream of logical change
- ➔ Keep track of the replication
- ➔ Changes are decoded row by row, even if they were produced by a single command
- ➔ it controls the amount of WAL to be kept at the server : Be careful !

webedia
MOVIES WITH PASSION

Once a slot is created...

- ➔ ...no WAL records are cleaned up until they are no longer required.
  This means that if you create a slot but no client ever connects…
- ➔ Or if your output plugin is crashing
  ... no WAL records are ever cleaned up

AND YOU WILL RUN OUT OF SPACE

webedia
MOVIES WITH PASSION

pg_recvlogical

- ➔ Controls logical decoding replication slots and streams data from replication slots
- ➔ It sends replay confirmations for data as it receives it
- ➔ Unnecessary changes can be filtered out

```
pg_recvlogical -h ['host'] -d ['dbname'] -p ['port'] --slot ['name_slot]  -U ['user'] --start add-tables=social.* -o include-types=0 -o include-timestamp=true
```

Wal2json the output plugin

- ➔ The plugin have access to tuples produced by INSERT and UPDATE
- ➔ UPDATE/DELETE old row versions can be accessed depending on the configured replica identity
- ➔ Produces a J**SON object per transaction**. All of the new/old tuples are available in the JSON object.
- ➔ https://github.com/eulerto/wal2json

Wal2Json set up: postgres conf

```
############ REPLICATION ##############
# MODULES
shared_preload_libraries = 'wal2json'    ①

# REPLICATION
wal_level = logical                      ②
max_wal_senders = 4                      ③
max_replication_slots = 4                ④
```

1 =>loads the wal2json logical decoding plug-in
2 =>uses logical decoding with the write-ahead log
3 =>uses a maximum of 4 separate processes for processing WAL changes
4 =>should allow a maximum of 4 replication slots to be created for streaming WAL changes

Wal2Json ready

- **Create a slot** named test_slot for the database named test, using the logical output plug-in wal2json

```
pg_recvlogical -d test --slot test_slot --create-slot -P wal2json
```

- **Begin streaming changes** from the logical replication slot test_slot for the database test

```
pg_recvlogical -d test --slot test_slot --start -o pretty-print=1 -f -
```

Wal2Json output

➔ **Perform some basic DML** operations at test_table to trigger INSERT/UPDATE/DELETE change events

```
test=# INSERT INTO test_table (id, code) VALUES('id1', 'code1');
INSERT 0 1
test=# update test_table set code='code2' where id='id1';
UPDATE 1
test=# delete from test_table where id='id1';
DELETE 1
```

➔ Wal2Json produces a Json object **per transaction** :Output for INSERT event

```
{
  "change": [
    {
      "kind": "insert",
      "schema": "public",
      "table": "test_table",
      "columnnames": ["id", "code"],
      "columntypes": ["character(10)", "character(10)"],
      "columnvalues": ["id1       ", "code1     "]
    }
  ]
}
```

webedia
MOVIES WITH PASSION

Wal2Json output

➔ Output for UPDATE event

```
{
  "change": [
    {
      "kind": "update",
      "schema": "public",
      "table": "test_table",
      "columnnames": ["id", "code"],
      "columntypes": ["character(10)", "character(10)"],
      "columnvalues": ["id1       ", "code2     "],
      "oldkeys": {
        "keynames": ["id"],
        "keytypes": ["character(10)"],
        "keyvalues": ["id1       "]
      }
    }
  ]
}
```

webedia
MOVIES WITH PASSION

Wal2Json output

➔ Output for DELETE event


```
{
  "change": [
    {
      "kind": "delete",
      "schema": "public",
      "table": "test_table",
      "oldkeys": {
        "keynames": ["id"],
        "keytypes": ["character(10)"],
        "keyvalues": ["id1       "]
      }
    }
  ]
}
```

A word of caution

- ➔ Big transactions issues (more than 1GB of memory)
- ➔ Wal2Json can not handle too big transaction unless the use of option write-in-chunks but the json is **not well formed**
- ➔ pg_recvlogical pass from streaming state to **catchup state**
- ➔ The master might run out of disk space
- ➔ NEVER use replication slots without monitoring

webedia
MOVIES WITH PASSION

Monitoring interfaces

- ➔ pg_stat_replication
- ➔ pg_replication_slots
- ➔ pg_stat_activity
- ➔ Exemple of check :

```
SELECT 1
FROM pg_replication_slots s
INNER join pg_stat_replication r on s.active_pid=r.pid
WHERE r.state='streaming'
AND s.slot_name = 'wal_parser'
AND s.active_pid is not null
AND confirmed_flush_lsn is not null;
```

WalParser command

- A service that uses pg_recvlogical to
  - Create a replication slot using the plugin output Wal2Json
  - Start streaming changes from this replication slot
- Read the Json output, and turns them into MQ messages
- Sends the message to the queue

Agenda

Problem statement

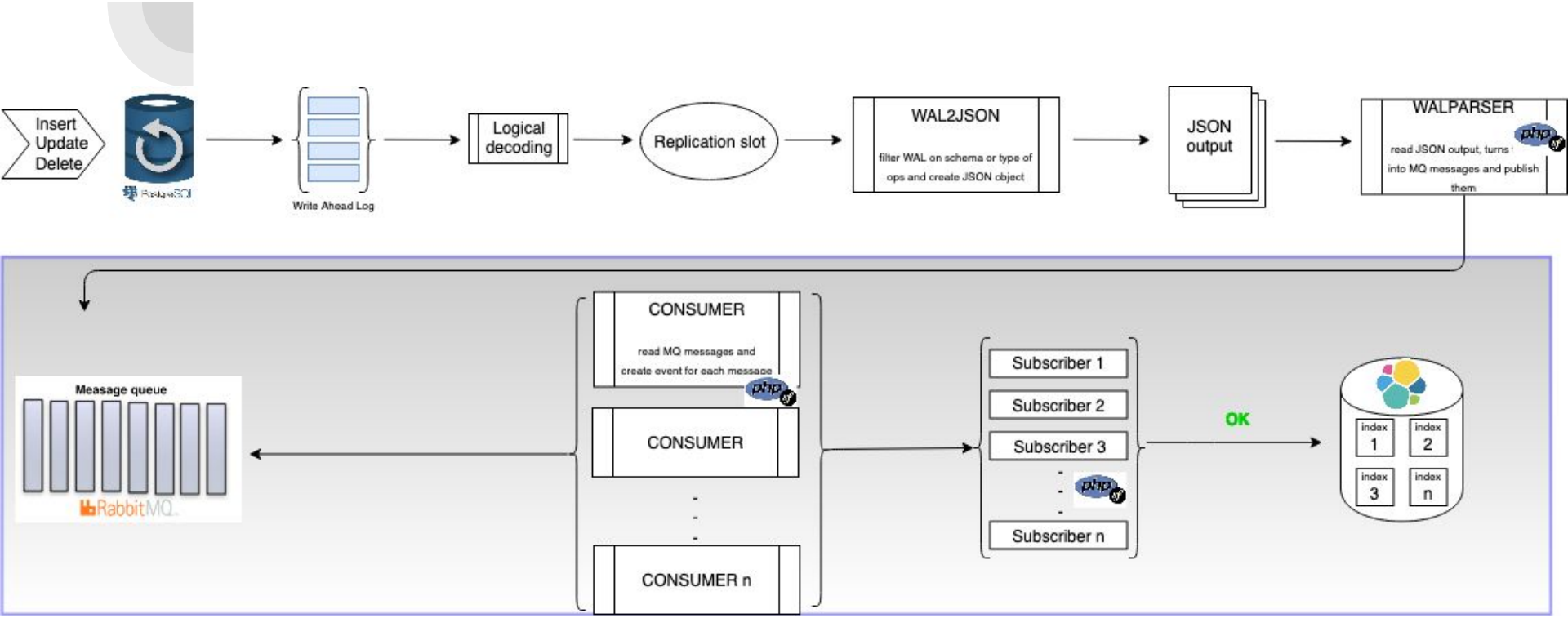
Capture the data Change as Near to REAL-TIME

Message Queuing: Write data events quickly to ElasticSearch

It works !!!!!!

Conclusion

webedia
MOVIES WITH PASSION



RabbitMq

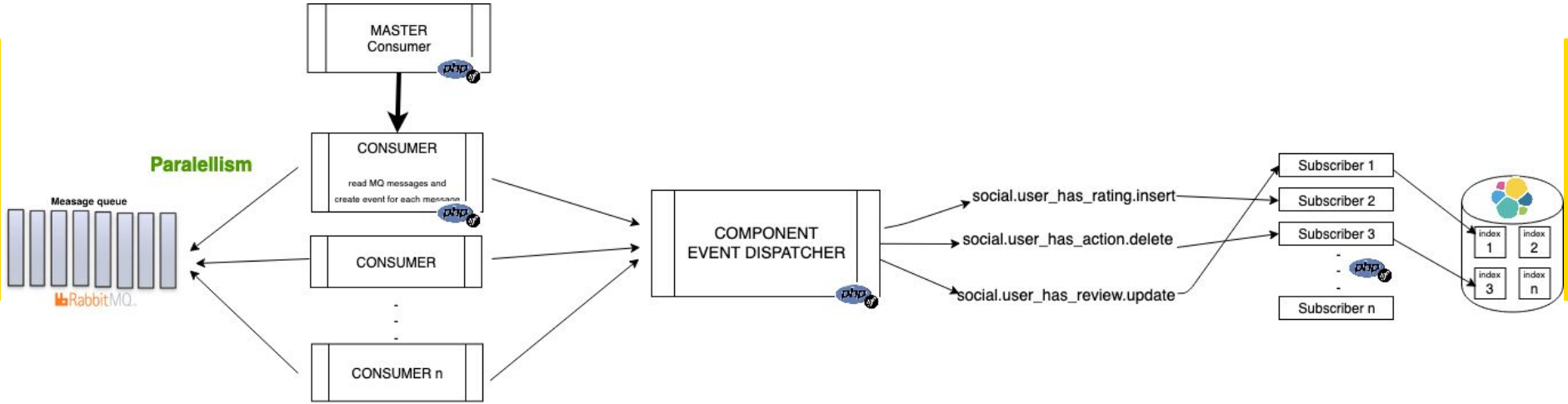
- RabbitMQ is a message broker
- It acts as a middleman
  - Reduces loads and delivery times by delegating resource-heavy tasks to a third party

- multiple consumers can retrieve the message in parallelism
- The sender and receiver have low coupling

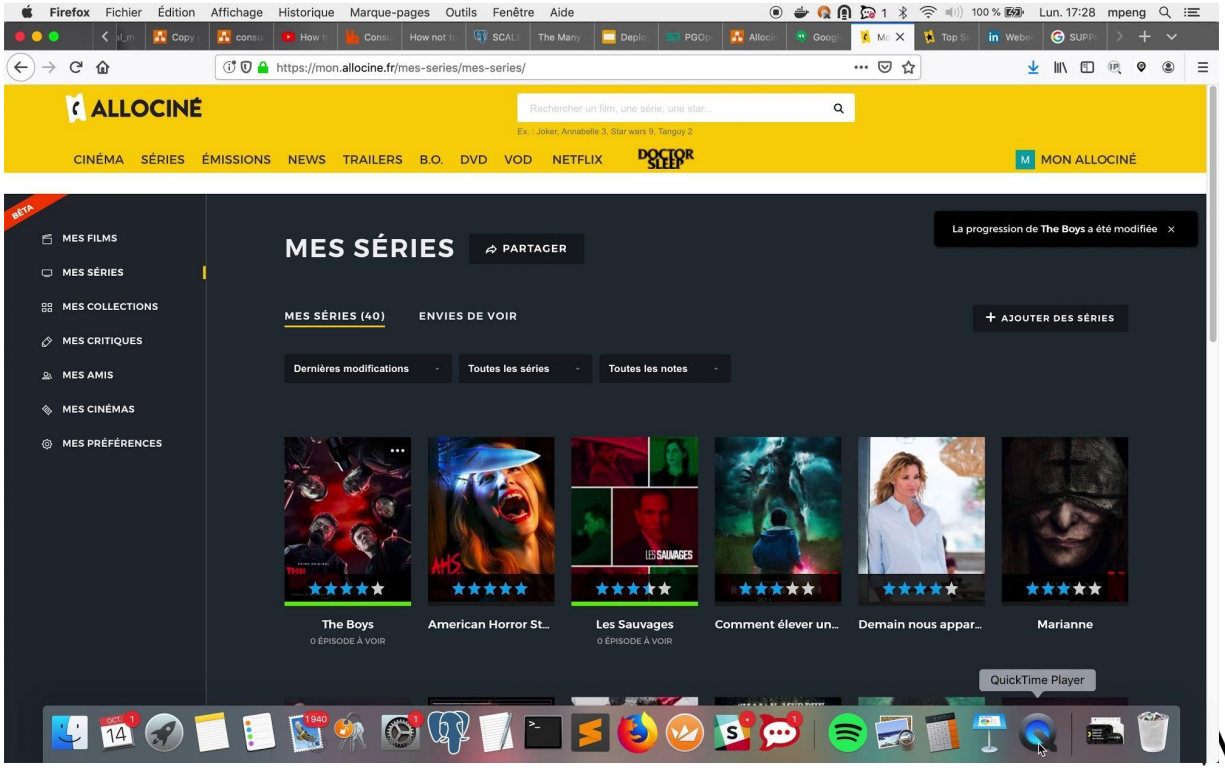
Benefits of using ElasticSearch

- ➔ Manages the huge amount of data
- ➔ Direct, Easy and Fast access
- ➔ Scalability of the search Engine

Consumers and Subscribers



It works



vebedia
MOVIES WITH PASSION

Conclusion

- ➔ Logical decoding and Wal2Json are keys:
  To output data changes from db to json objects
  To generate a message event per action (commit per row)
  To reduce database loads

- ➔ Small messages are send to an MQ:
  Queues keep the order of modifications for single p.k. values
  Enables concurrent processing to take place using parallelism

Now social events are written into Elasticsearch in milliseconds without querying the database.

webedia.
MOVIES WITH PASSION

THANK YOU

Q & A

webedia.
MOVIES WITH PASSION