facebook

Redis @ Facebook

Guy Yonish

Production Engineering Manager

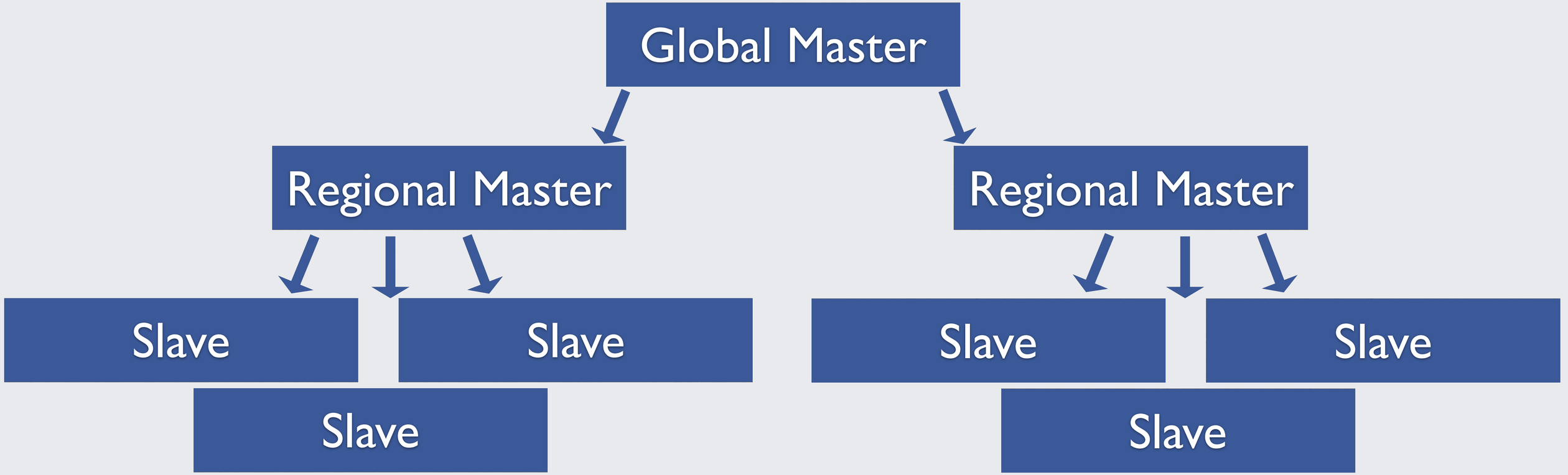
Problem #1 - User configuration

- Need to read user configs in real time for every request
- Need to update user config in some cases
- High request rate - tens of thousands per second. Some distribution is needed.
- Requests are blocked on the results. Answers are needed as quickly as possible

The Solution:

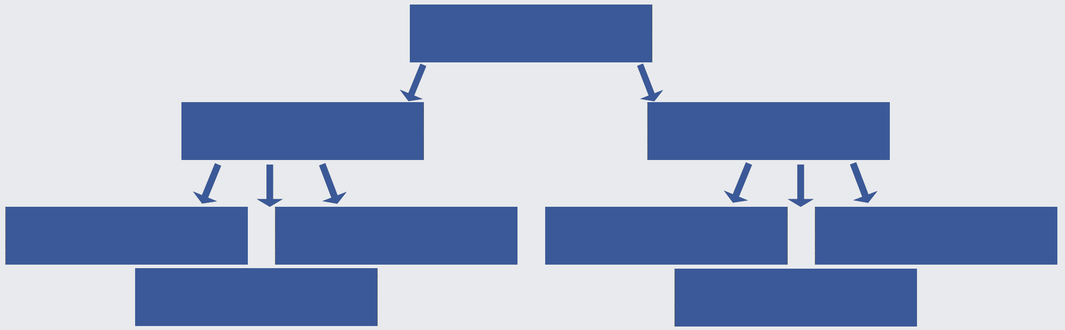
redis

Hierarchical master/slave replication



Hierarchical master/slave replication

- User configs are saved as Hashes
- The multi-level replication brings the data close to the server



- Reads are distributed across the Slaves.
- If more rps is needed - more slaves can be added - very scalable.
- Writes can be performed at every level depending on the need

Balance the load easily - hash

Easy to use in your attributes file for every server

```
# Use my fqdn as the shard key
shard_key = node[‘fqdn’]

# Returns a random position in the redis servers array
redis_shard = Digest::MD5.hexdigest(shard_key) \
[0…7].to_i(16) % redis_servers.length

my_redis = redis_servers[redis_shard]
```

Problem #2 - Data processing (stats)

- Data processing and insertion into long term storage takes time
- Stats are needed in real time for the user

The Solution:

redis

Split data processing model

- Perform daily bulk processing of that day. Can take long time.
- Tailers process stats realtime and insert them into sharded Redis cluster
- Using HINCRBY makes it easy to increment counters
- Updates are performed in bulks (batches)
- Using Redis TTL to automagically delete data after X days (while the bulk processing has finished)
- 10s of GB of data, sharded using twemproxy.

facebook