Real-time Gaming Leaderboard

xxx

April 2023

First think of it as single machine
then think how to scale it as distributed

1 Step 1

Understand the Problem and Establish Design Scope

- real-time update score once the user finishes a game
- score reflected on leaderboard
- leaderboard shows top 10 users
- user find its ranking place in the leaderboard

Estimation:
5 million DAU
QPS: around 500 at peak time

2 Step 2

Propose High-level Design
API design:

- post user score once he finishes
- get top 10 user scoreboard
- get my own score

High-level Architecture

- end user

- game service: can use a message queue if there are multiple consumers
  consumers include leaderboard service and notification service and analytics service etc.

- leaderboard service: user do not send PUT request to this directly as not secure

- leaderboard score

What database relational database: not suitable for ranking and heavy update
Redis: works good in memory
so it can do in-memory ranking, using skip list or redis sorted set

3 Step 3

Design Deep Dive

- build service on the cloud
  use AWS lambda API so it can autoscale

- storage partition
  can partition by user score [0,100],[100,200]...
  using redis replica

- using NoSQL as alternative:
  write heavy, in different partition
  to get top 10, get top 10 of each shard and merge them together