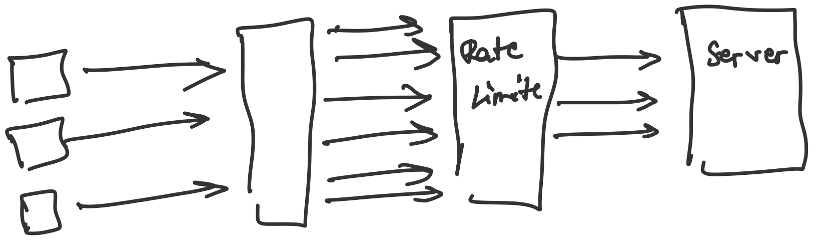
Rate limiter

Saturday, 15 November 2025 10:33 AM

Q. What is a Rate limiter?

→ A software component that can control the amount of requests coming to a system.



Q. Why do we need a rate limiter?

→ To protect attack (DDOS)
↓ Distributed ↳ Denial of Service

→ Brute force login attempts

→ Reducing infra cost

→ Help making system available to users uniformly.

→ Reduce burst traffic spikes.

256^8

$(2^8)^8$

$\boxed{2^{64}}$

Requirements

Functional

→ Limit number of request to an API within a time limit.

{ → The rate limit should be considered across different servers.

Non-functional

{ Component → separate component

→ Highly available
→ Low Latency

datastructure → map
algo

(HTTP 429 → Too many requests)

Types

→ Hard → requests cannot exceed the threshold.
↳ Banking

→ Soft → requests can exceed a certain percentage.
↓ grace cushion

100 → 10% } → 110

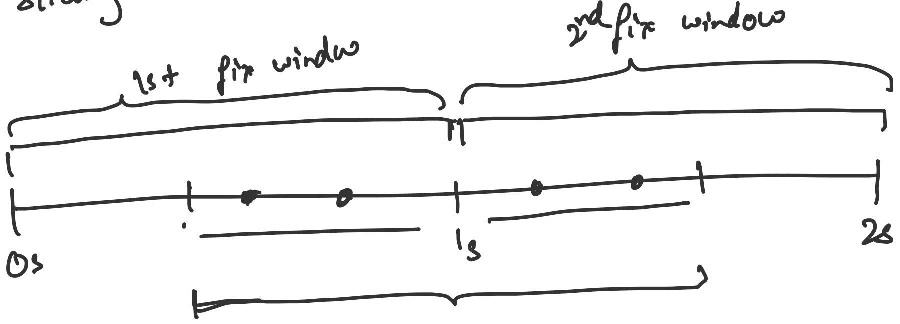
→ Dynamic → • complex throttling
• when you want the system to adapt to user.

1000/m → 100 /m /customer
900

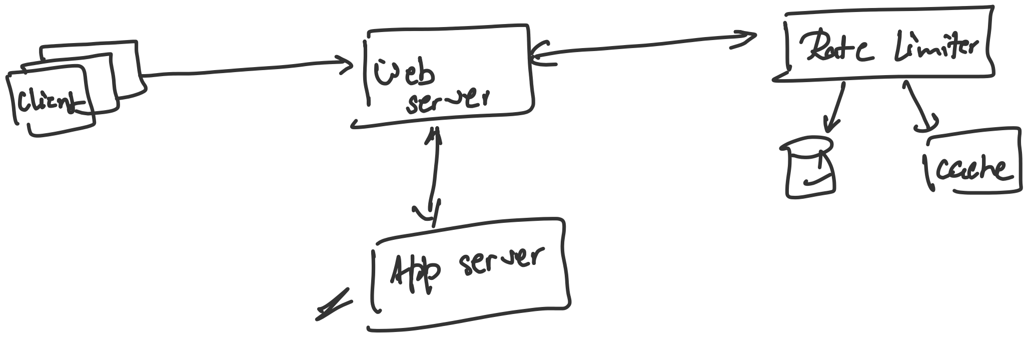
AWS S3 → 3500 /s (write)
5500/s (read)

Algorithms

→ Fixed window } → limit after n-request in a fixed time window
→ Sliding window



HLD



System design

Fixed window

Note: If
not p
address
can be

key → value.

userId : {count, startTime}

→ If userId is not present in map, set count
startTime with userId, allow the request
↳ normalize.

10s 3:00:00
3:00:05

↱ me

→ If userId is present, & currentTime - (startTi
set currentTime as start, count as 1
update the entry) → (start a new t

→ If userId is present & current - st

→ If count < threshold, • increment

→ If count >= threshold , block.

Memory

$\underbrace{\text{userId}}_{8}$: { $\underset{8}{\underset{\downarrow}{\text{count}}}$, $\underset{8}{\underset{\downarrow}{\text{timestamp}}}$ } =24 byte

overhead

↳ 20 bytes needed for overhead

Total size of 1 record = 2

for tracking 1 million users:-

44 bytes × 1M = 44 MB →

Rate limit is of 10 req/s

Traffic on system ⇒ 10 × 1M =

⇒ we need to store in a distributed cache cluster

10M queries

Sliding window (1min)

track requests in the last minute fro
request.

3:01:30
↳ see requests starting from

key → value
↳ sorted set of ti

userId → { $\underline{t_1}$, $\underline{t_2}$, t_3.... , t_n }

→ req →
↳ check the first item of set
↳ if it falls within time window
& set size >= threshold
(block the request)

→ userId is not present.
↳ add userId with current time

→ userId is present

→ remove from start till m
than (current Time - time wi

→ If size of set > threshold,

→ otherwise, add timestamp to

1 user ⇒ 10 req /min

$$\underset{20}{\boxed{\text{Map}}}\ (\underset{8}{\text{userId}} \longrightarrow \underset{20}{\boxed{\text{Set}}} < \underset{(8\times10)}{\text{Timestamp}}>$$

$$\underbrace{20}_{\text{map overhead}} + \overset{\text{userId}}{8} + \underbrace{10}_{\text{max entries in set}}(\overset{\text{set overhead}}{20} + \underbrace{8}_{\text{timestamp}})$$

28 + 280 = 308 bytes.

Tracking 1 million users would need.

$1M \times 308 = \boxed{308\ \text{MB}}$

~7x increase.

II. Partitioning

[illegible]

- → key of the map
- → range based.

What to use as a key.

→ user → can limit requests per use
multiple accounts can be

→ IP → can limit requests per dev
user can use multiple de

→ Hybrid → use combination of both.