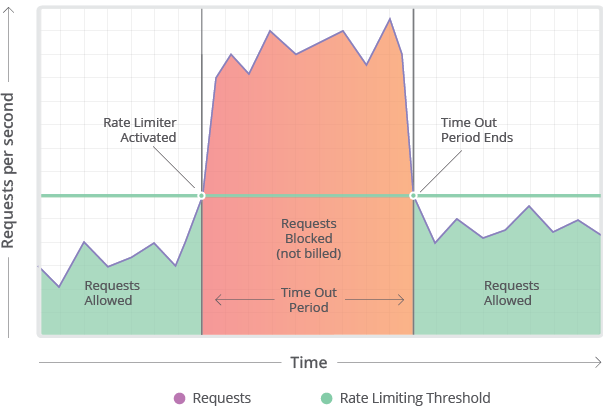


Rate Limiter?

* Refers to preventing the frequency of an operation from exceeding a defined limit.
* Rate limiter is used to protect underlying services and resources.
* Protect your API from unintended or malicious overuse by limiting the number of requests that can reach your API in given period of time.



Why rate limiting?

1. Preventing Resource Starvation :

Improve the availablility of API-based services by avoiding resource starvation.

1. Security :

Rate limiting prevents brute forcing of security intensive functionalities like login , promo code etc.Number of request to these features is limited on user level so the brute force algorithm don’t work in this scenarios.

1. Preventing operational costs:If rate limiting is not employed , exposed to exponential bills

Rate limiting strategies :

1. User : User based rate limiting. No. of request allowed for user in given period of time .
2. Concurrency : No. of parallel sessions allowed for a user in given timeframe. Helps in migigate DDOS attacks as well.
3. Location / ID : Request not from the targer demography can be rate limited so as to increase availability in the target regions.
4. Server : Server based rate limiting is niche strategy .Servers are stringly coupled with specific functions.

Rate Limiting Algorithms :

1. Leaky Bucket :
   * + Simple intuitive algorithm.
     + Queue with finite capacity.
     + All the request beyond the capacity of queue are splilled off.
     + It smoothens out bursts of request and processes them in constant Rate.

Drawback : burst of requests can be fill up the bucket leading to starving of new requests.

1. Token Bucket :
   * + Assign token at user level.
     + Fetch token : The current number of tokens for the user is fetched.If it is greater than the limit defined then the request is dropped.
     + Update token : Ifthe fetched token is less than the limit for the duration d , then the request is accepted and token is appended.
     + This algorithm is memory efficient . Problem here is it can cause race conditions in the distributed environment.
     + When 2 request from 2 different application servers trying to fetch the token at the same time.
2. Fixed window Counter :
   * + Keep counter for given duration of time .keep incrementing it for every request we get.once limit is reched , we drop all further request till the time duration is reset.
3. Sliding Log:
   * + This alogorithm involved maintaining a time stamped log of request at the user level.
     + It discards all the request within timestamps beyond the threashold.
     + Request rate can be calculated by summing the logs.
     + System tracks the sliding log for wach consumer.
     + It avoid starvation.

Rate Limiting in distributed Systems

* + Works very well for single server application.
  + 2 problems are inconsistency and race conditions
* Inconsistency :
  + Different regions having own rate limiters . Need to dine global rate limiter.
  + The grater the number of noes , more likely the user will exceed the global limit.
  + 2 ways to solve the problem
* Sticky session : sticky session in load alancers

Centralised Data store : Use centralized data store like Redis or Cassandra to handle counts for each window and consumer.

Race Conditions:

* Each request gets the value of counter then tries to increment it.
* While write operation is completed , several other request have read the value .
* Very large number of request are sent than what was intended.
* Intorducting locks on write-read operation. Making it atomic
* Thus increases performance cost and become a bottleneck casuing more latency.

Throttling

* Its process of controlling the usage of API by customers during a given period.
* Throttliing can be defined at application level. When throttling limit is crossed , the server returns 429 – too many request .

Type of throttling :

1. Hard Throttling : The number of API request can wexceed the threottle limit.
2. Soft Throttling :

We set the API request limit to exceed a certain percentage.

Say we increase to send 110 messages instead of 00 messages.

1. Elastic or Dynamic throttling:

Request form the user can go beyond the threshold ifthe system has some resource available .