# 2) SYSTEM DESIGN OF PASTE BIN - CONTENT SHARING SERVICE

**1) Requirements & Clarifications:**
(i)Design content upload and sharing service that provides URL with expiry date

Use Case: It is mainly used for sharing code, log files etc

(ii) It can optionally have custom alias
(iii) Read : Write ratio is 5 : 1
(iv) Per day 1M Posts
(v) It can have expiry date

**2) Capacity Estimation :**

(i) Traffic / Query Per Second (QPS) ??
Write(QPS) :
1M / 24(hours) * 3600(seconds) = 12 / seconds

Read(QPS):
5M / 24 * 3600
= 58 / seconds

(ii) Storage Required ??
Assuming 10 years and each object is of 10KB
10 * 12 ( months ) * 10 KB = 36 TB

(iii) Network Bandwidth ??
Write :
12 (URL per second) * 10 (KB)
= 120 KB / sec

Read :
58 ( URL per second ) * 10 ( KB )
= 580 KB / sec

(iv) Memory Estimation ( caching ) ??
Considering 80:20 rule , that is 20% Redirecting generating 80% traffic

0.2 * 5M * 10 ( KB )
= approx 10 GB

note: less than 10GB will be needed, as majorly requests wiill be duplicates

**3) System Interface Definition**

(i)createPaste( api_dev_key, content, custom_alias=None,expiry_Date=None)

(ii)getURL( api_dev_key, content_url)

(iii)deleteURL( api_dev_key, content_url)

**4) Define Data Modeling**
observations are as below
(i) Read heavy
(ii) billions of records to be stored
(iii) no major relationship other than which user has created the url
(iv) object size average 10KB

we can have NO-SQL database considering above.

USER
userid int 10 (pk)
username varchar 50
creationDate dateTime
lastLogin dateTime

PASTE
hashCodeURL varchar 6 (pk)
ContentKey varchar 500
expiryDate dateTime
userid int 10 (fk)

**5) High Level Design**
Creating Paste Logic:
We can use Key Generation Service (KGS) that will generate unique keys before hand and store in key DB.

key DB will have NotUsedUrl and UsedUrl columns.
We can periodically load some keys in memory of KGS and application servers.

Application server will transfer data into Object storage (e.g amazon s3) and then insert into meta-data store ( e.g MySql or Cassandra) and return the hashContentUrl

**6) Detailed Design**
We can add load balancer before application server.
We can add 2 Cache server after application server. one for object storage and one for meta-data storage
Load Balancer can be placed in between cache servers and Databstores.
Cleanup service can be added to delete the expied contents from meta-data store and Key Db.

**7) Identification and resolving bottlenecks**

**Single point of failure?**

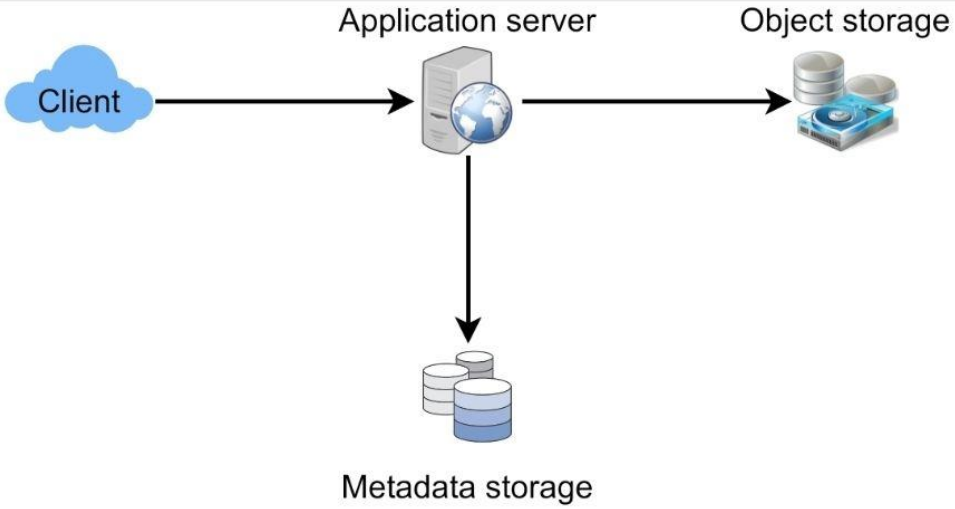
We can add standby load balancer and standby key Db, standby KGS service to avoid single point of failure

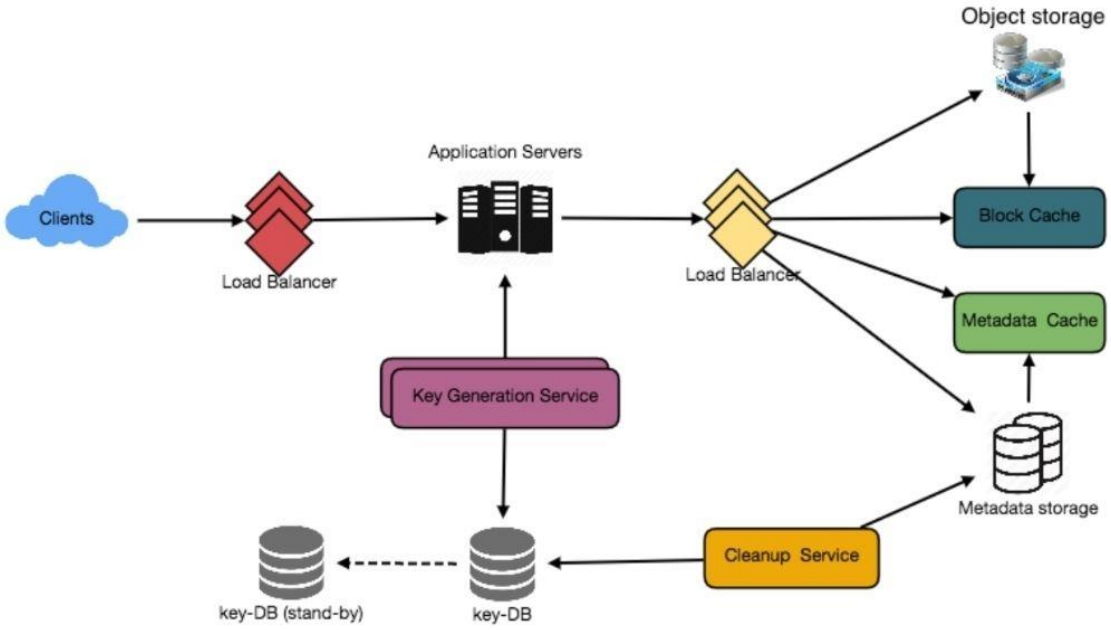
**Data partitioning?**

We can use consistent hashing for data partitioning

**Data Replication Services?**

As our system is READ heavy and AVAILABILITY is important, so we can add Replication services at application server level to avoid total system shutdown.




Detailed component design for Pastebin