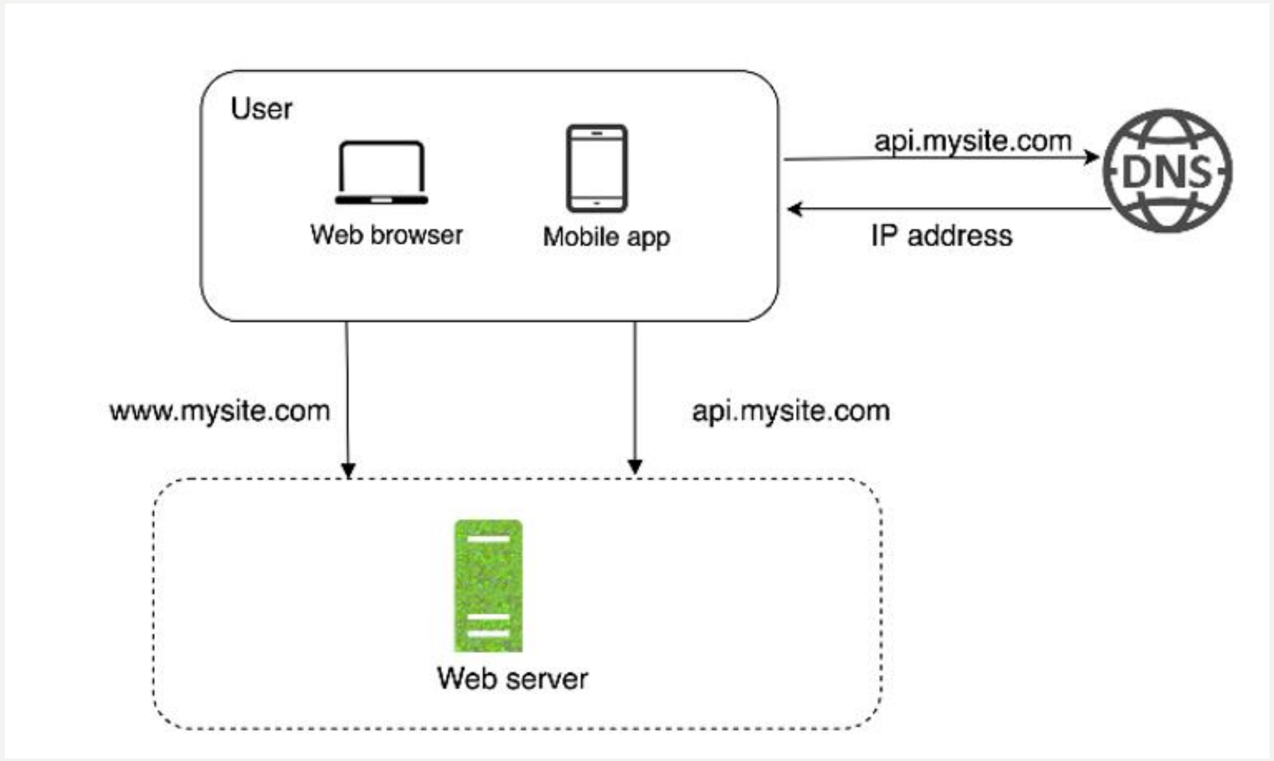
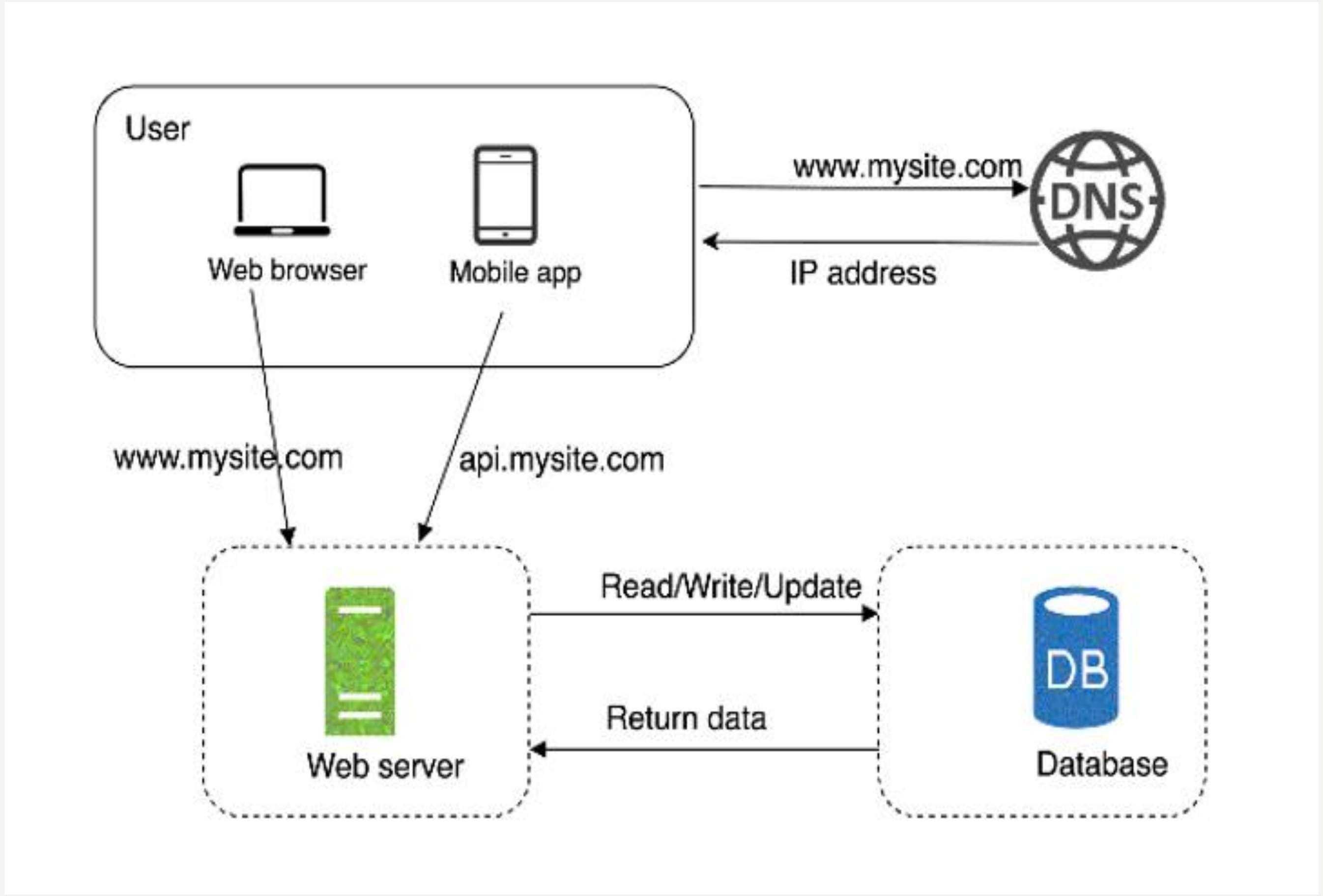
SCALE FROM 0 TO 10M USERS

SINGLE SERVER SETUP



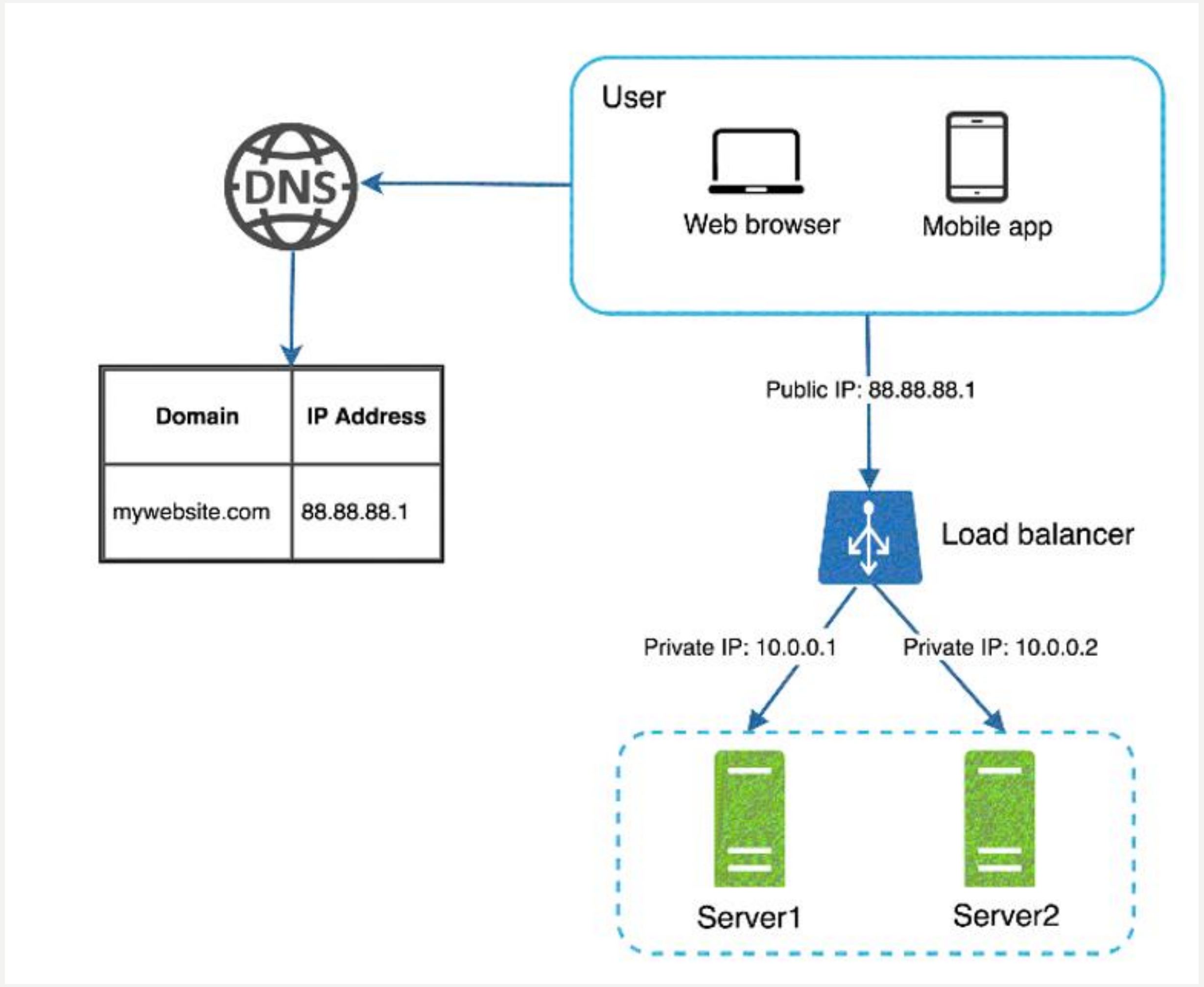
- Everything (Web Server, DB Server, Content) in single physical box
- Usually for web browser the server will generate the HTML output
- For mobile app the server will usually generate JSON response.

100 USERS



- **Split the servers** when user base grows
- Here we split the web tier and data tier
- When traffic is low **vertical scaling** is a good option
- Vertical scaling won't support failover and redundancy

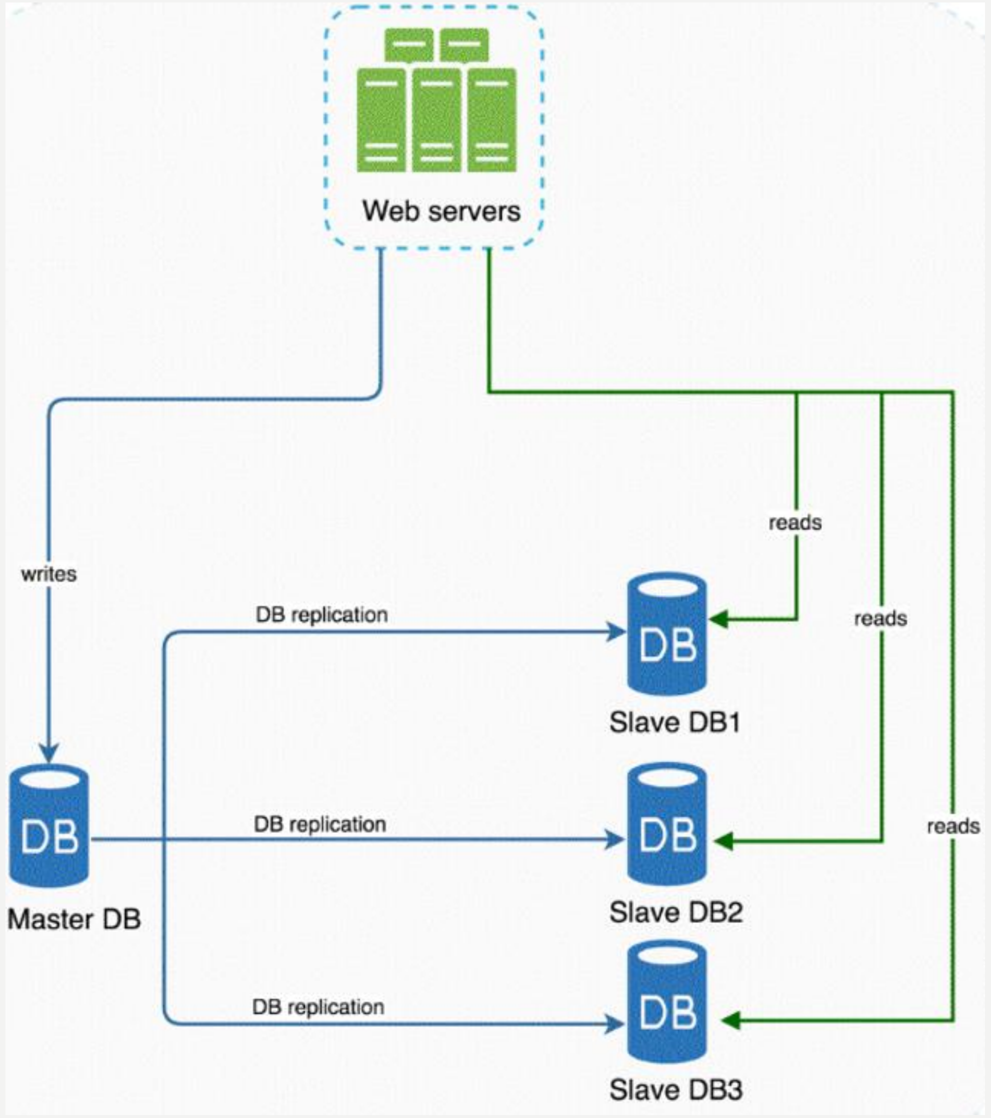
1000 USERS



Domain	IP Address
mywebsite.com	88.88.88.1

- Introduce **Load Balancer**
- Go for **horizontal scaling**
- For security the communication between the servers are always private and not accessible from internet

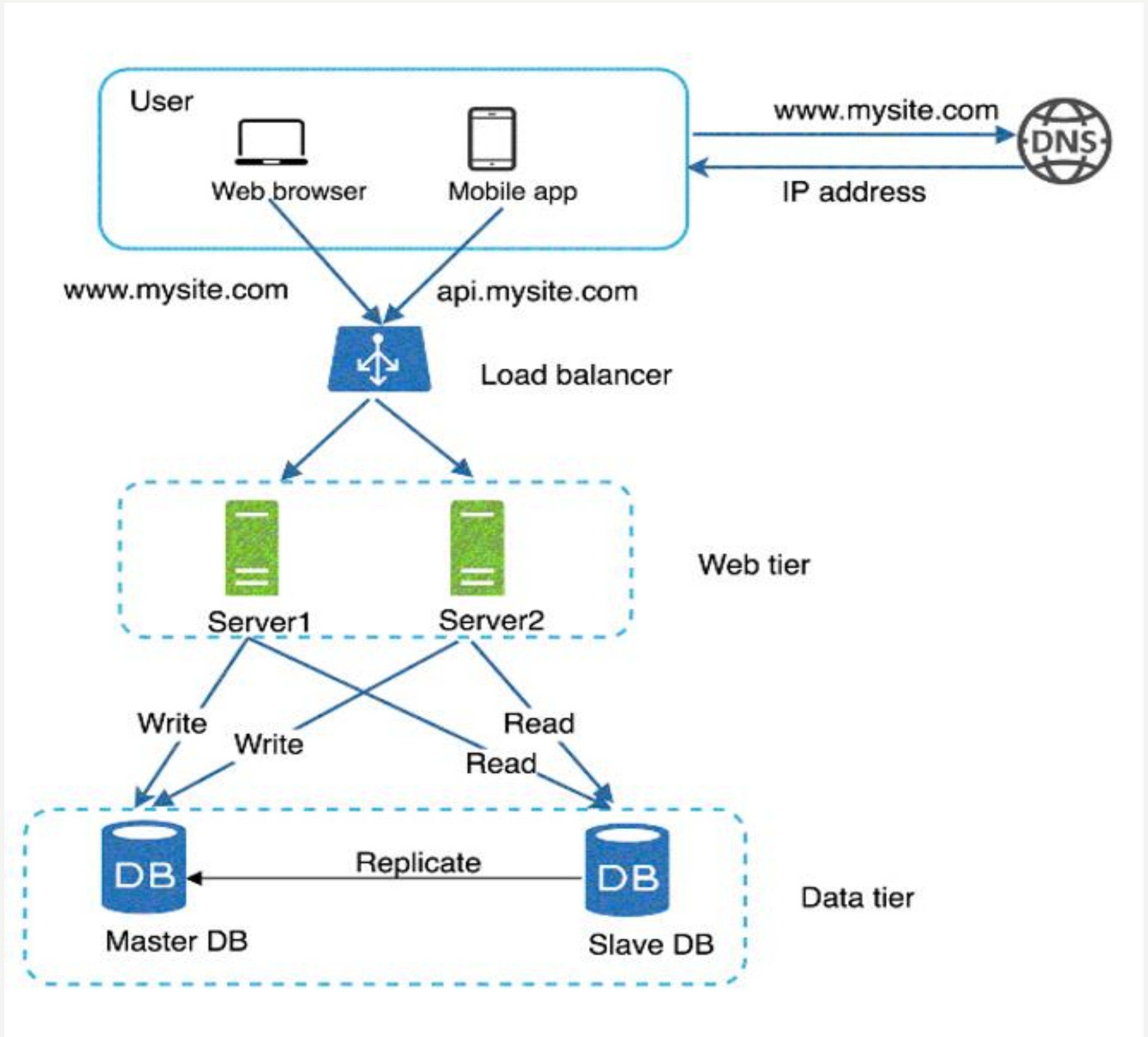
1000 USERS



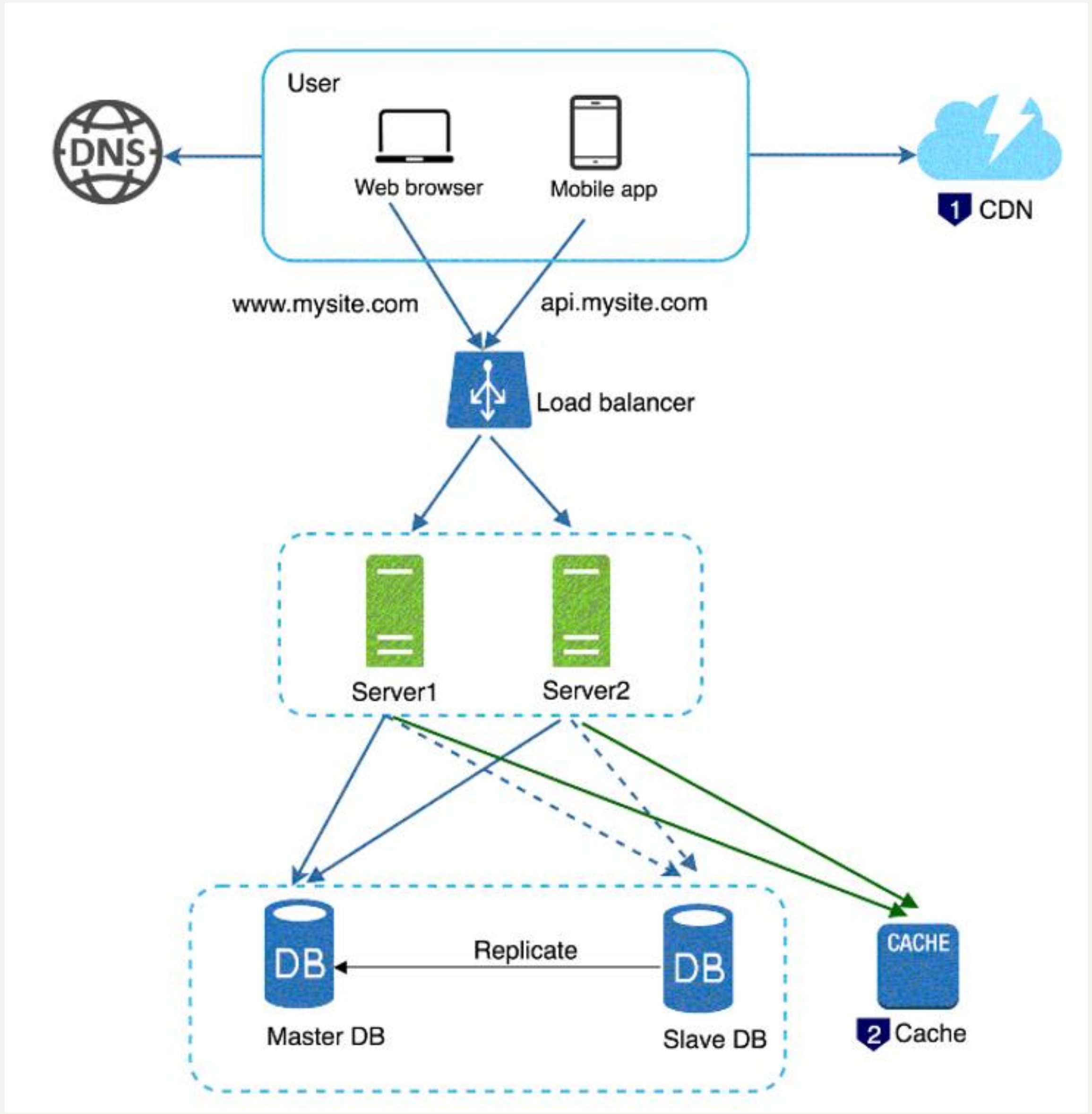
- **DB Replication**
- Supports failover and redundancy
- Usually done with **Master and Slave setup**
- All writes redirected to Master
- All reads redirected to Salves
- If Master goes offline slave could be promoted as a Master
- If there is only one slave and that goes offline Master will handle both reads and writes

Other replication options are Multi Masters, circular replication

1000 USERS FINAL SETUP

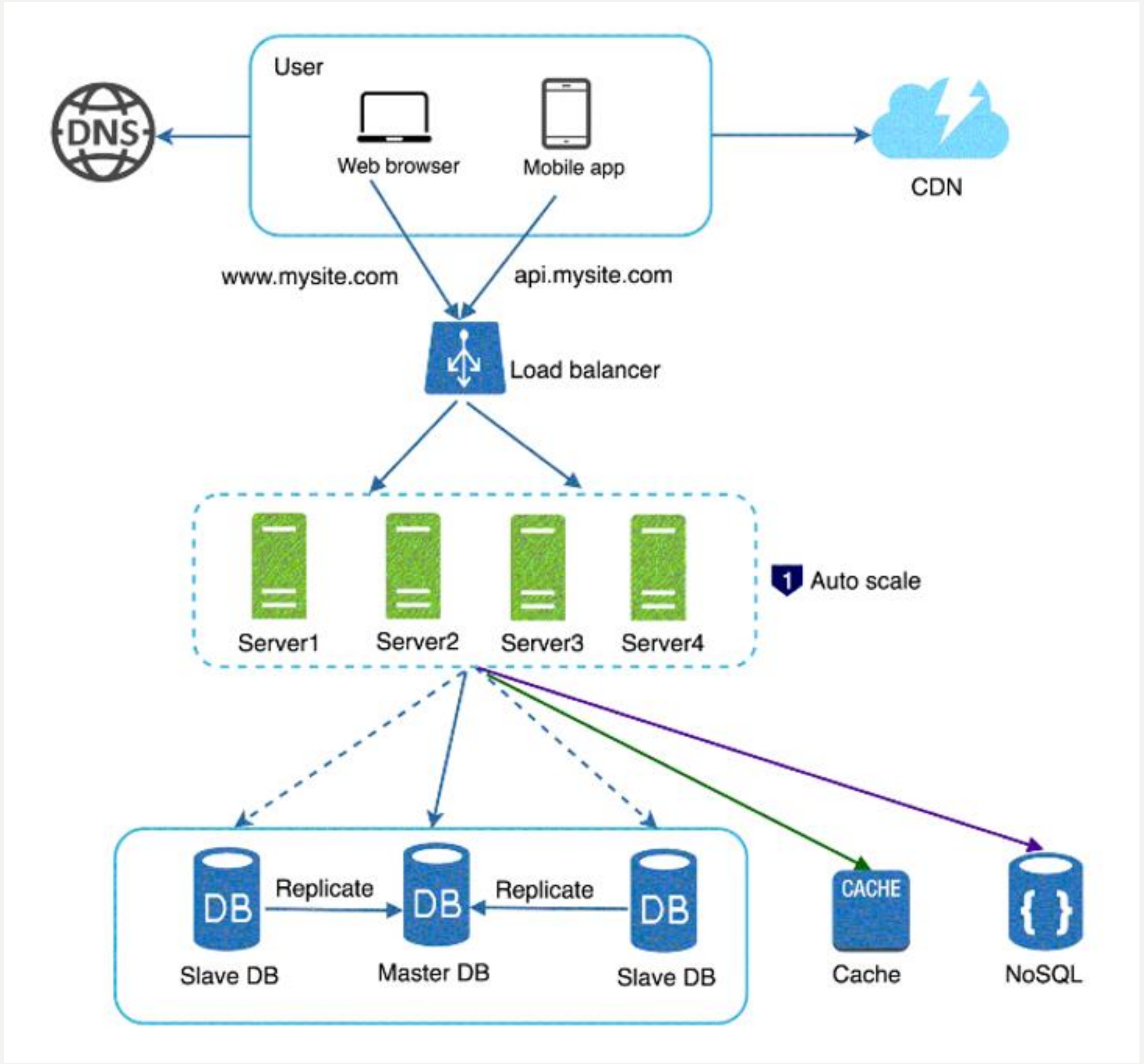


10,000 USERS



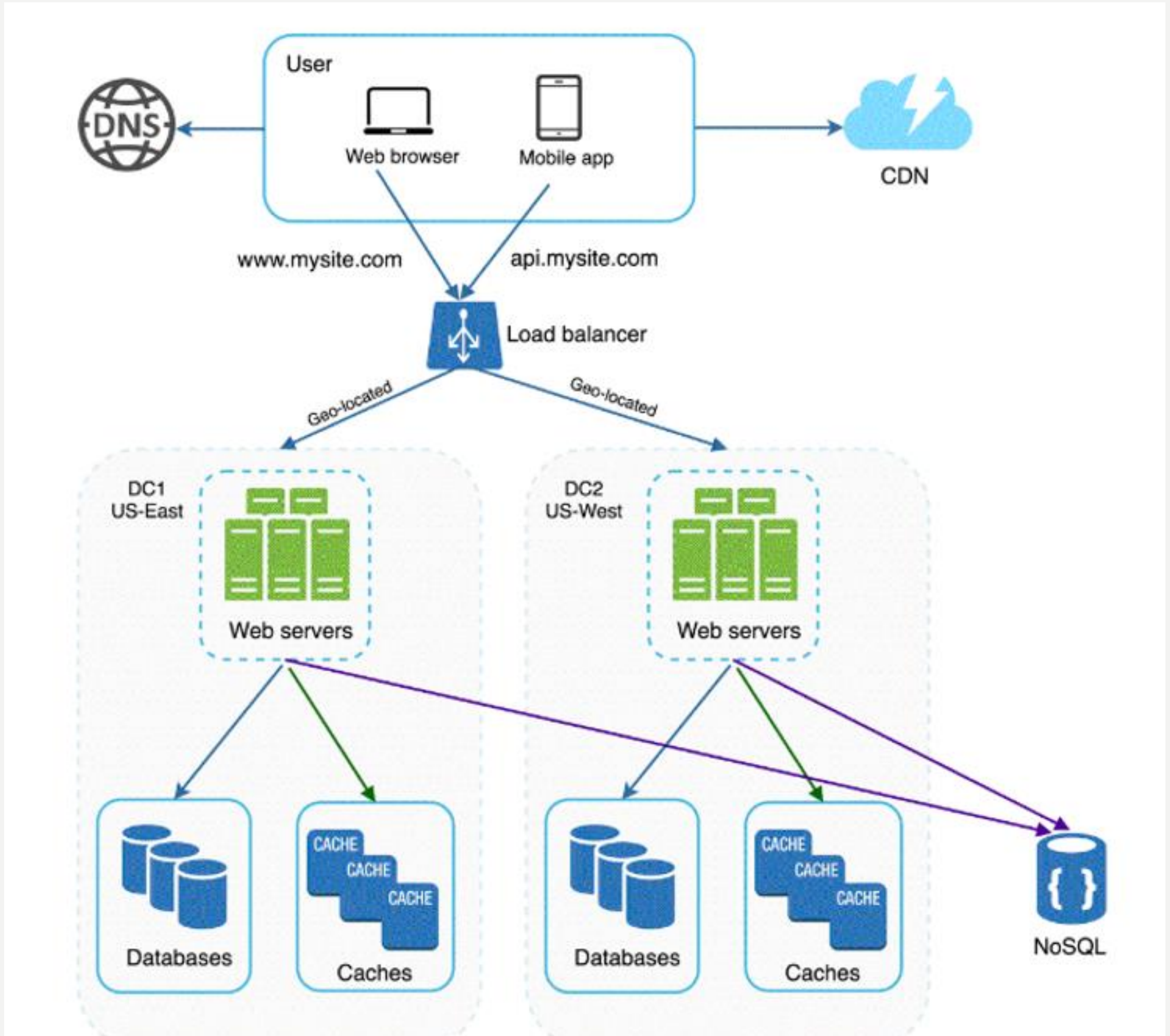
- Introduce **Cache**
  - Life time of the cache
  - Consistency
  - Mitigating failures in case of cache server failure
  - Evicting Data
- Introduce **CDN**
  - Server closes to the user will deliver the static content to the users
  - Cost
  - Invalidating files
  - CDN failover

1,00,000 USERS



- Embrace stateless architecture
- Move state data out of web tier and store in **separate shared data store**
- Auto scale the web tier

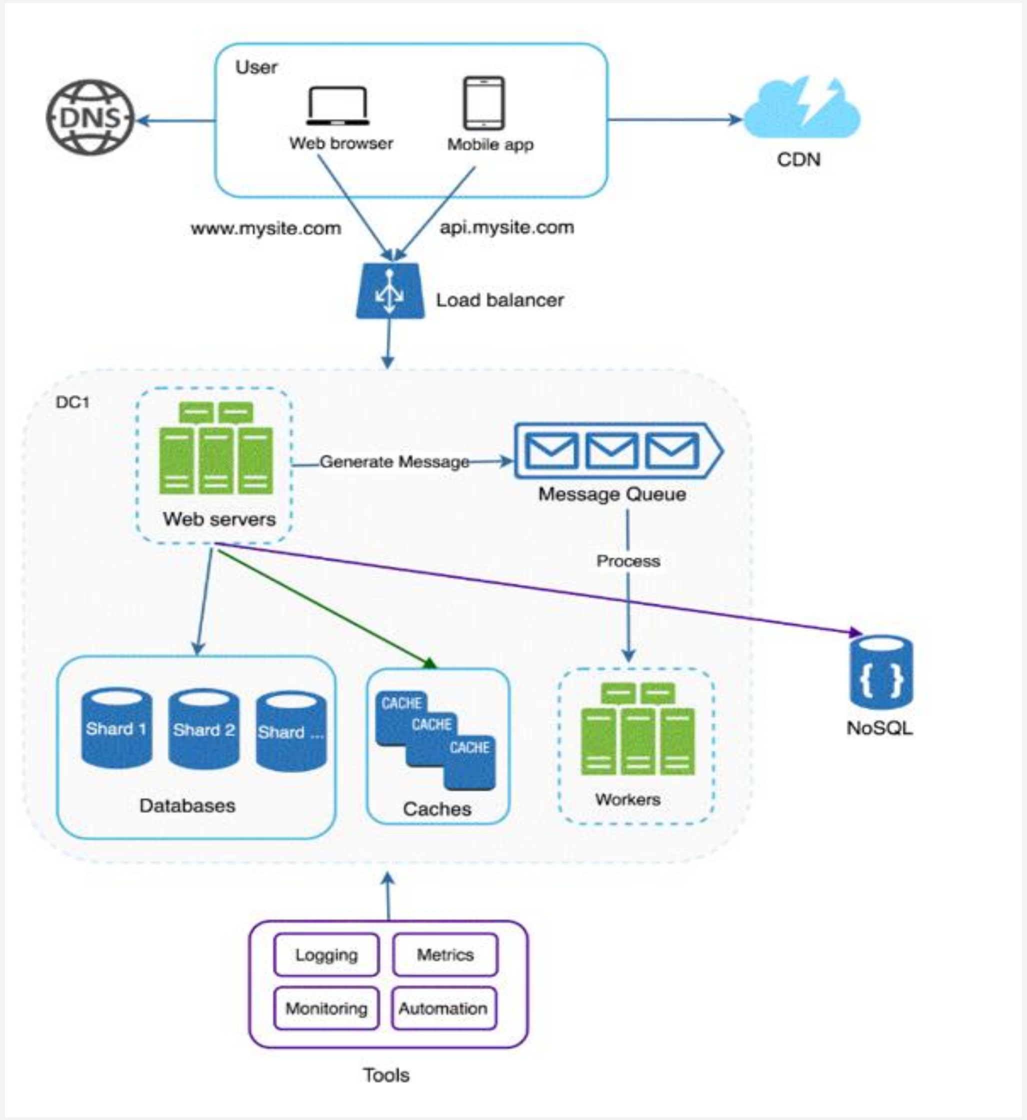
5,00,000 USERS



Note : Data synchronization across DCs are not shown

- Introduce **geo spread data centres**
- Use geo DNS to direct traffic to the nearest data centre
- Data synchronization would be the challenge across cache and DBs

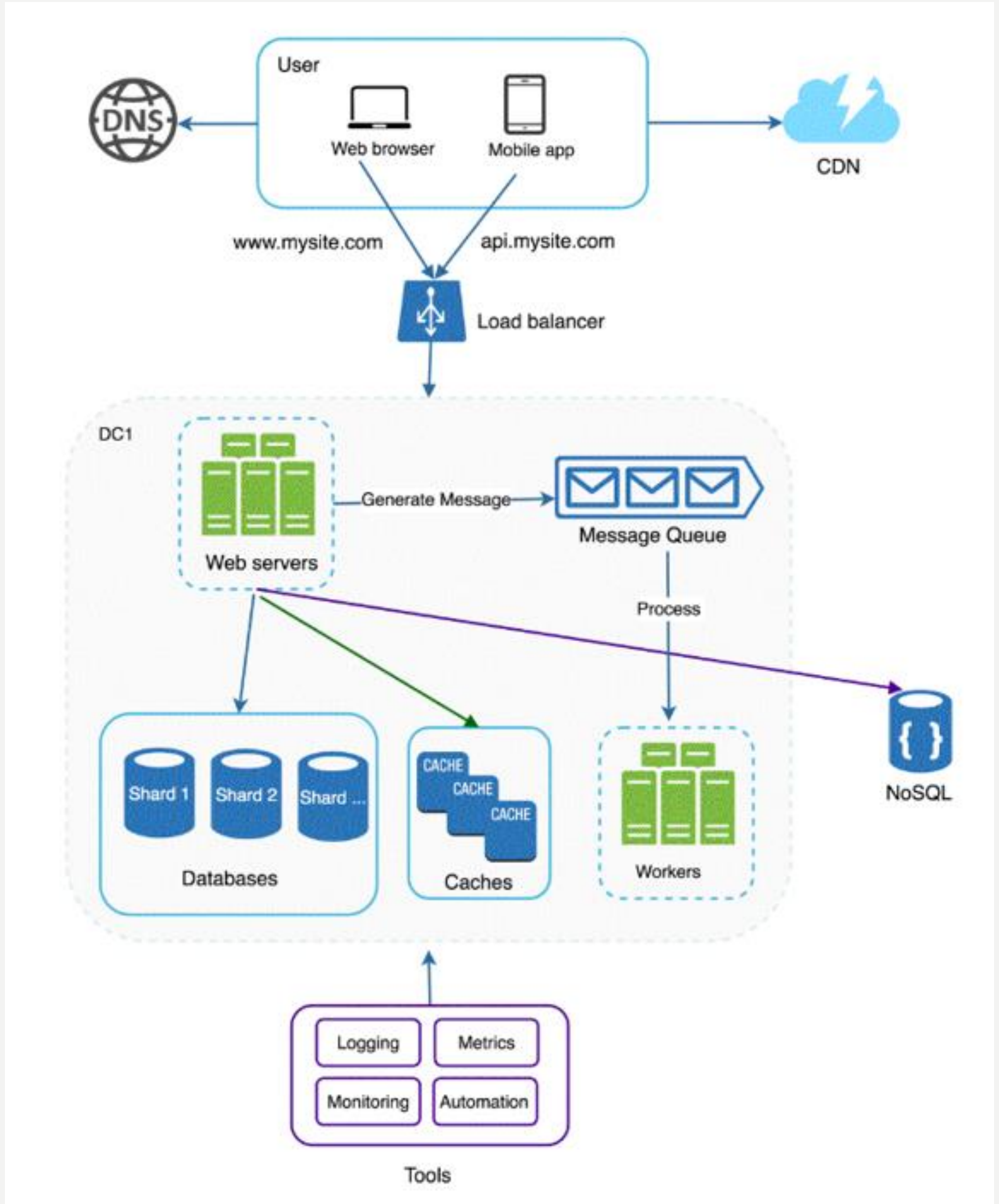
1 MILLION USERS



Note: Setup has to be replicated to multi DCs

- Embrace **Message Queue** for loose coupling
- Use different queue for different type of workloads
- Use centralized logging, metrics and monitoring tools

5 MILLION USERS



- Shard the DB (Horizontal Partition)
  - Hash is used to know which shard to read from
  - Selection of Shared key is very important
  - Problems
    - Resharding
    - Celebrity Problem
    - Joins and Denormalization

- Split DB based on functionality (Vertical Partition)
  - Joins will be challenging
- Get more powerful DB servers (Vertical Scaling)
- Lighten the DB load by using caches for read operation

10 MILLION USERS

- Keep Web tier stateless
- Build redundancy at each tier
- Cache Data as much as possible
- Support multiple Data Centres
- Host static contents in CDN
- Scale the DB tier using sharding
- Split tiers into individual services
- Centralize monitoring, logging

FURTHER READING TOPICS

- DB replication
- https://dev.mysql.com/doc/refman/8.0/en/replication.html
- https://en.wikipedia.org/wiki/Multi-master_replication
- https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-replication-multi-master.html
- https://medium.com/netflix-techblog/active-active-for-multi-regional-resiliency-c47719f6685b
- http://highscalability.com/blog/2010/10/15/troubles-with-sharding-what-can-we-learn-from-the-foursquare.html
- https://www.percona.com/blog/2009/08/06/why-you-dont-want-to-shard/

REFERENCE

- “Systems design interview an insiders guide” by Alex