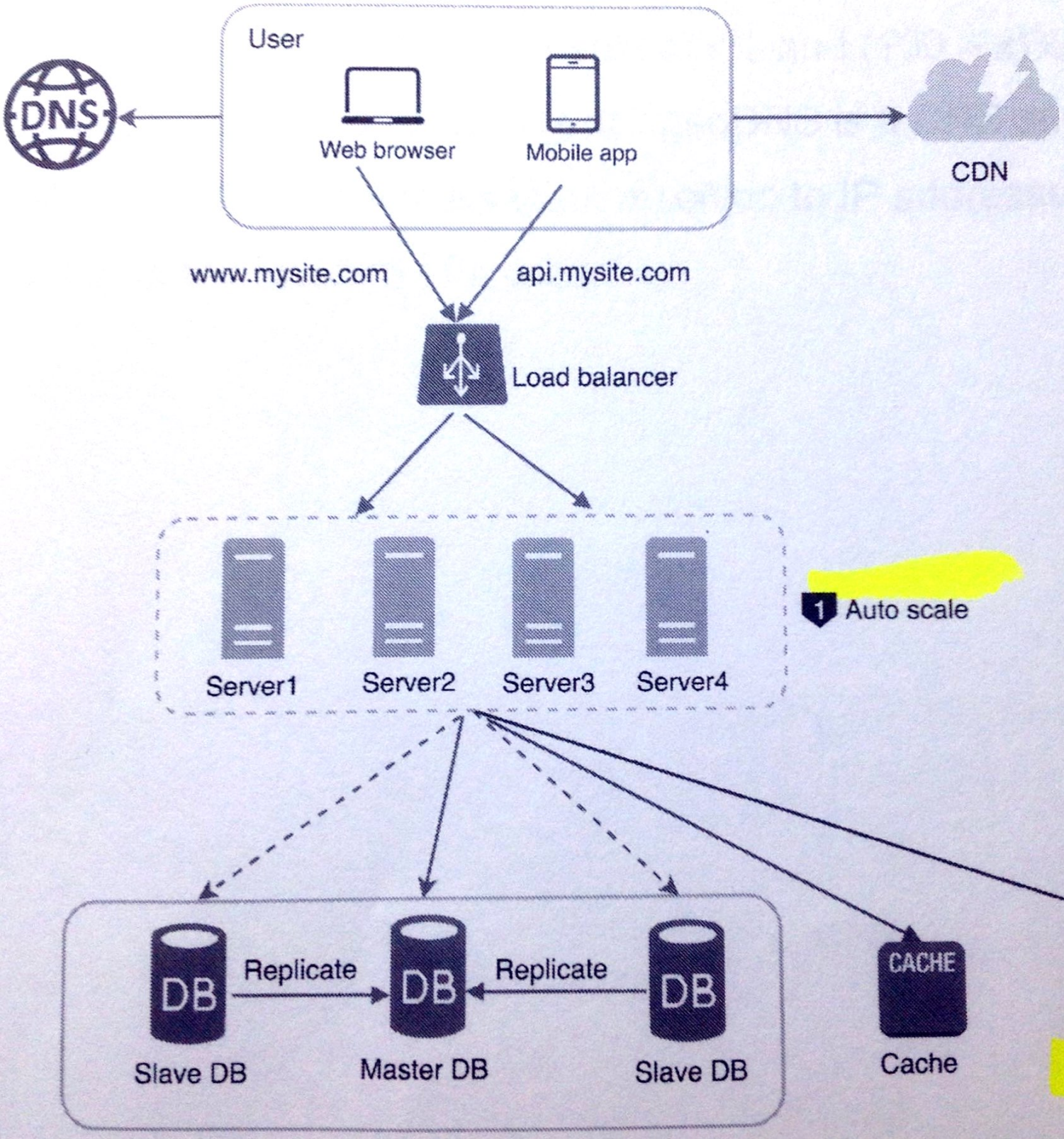
Design for one hundred thousand users

Figure 14 shows the new design.



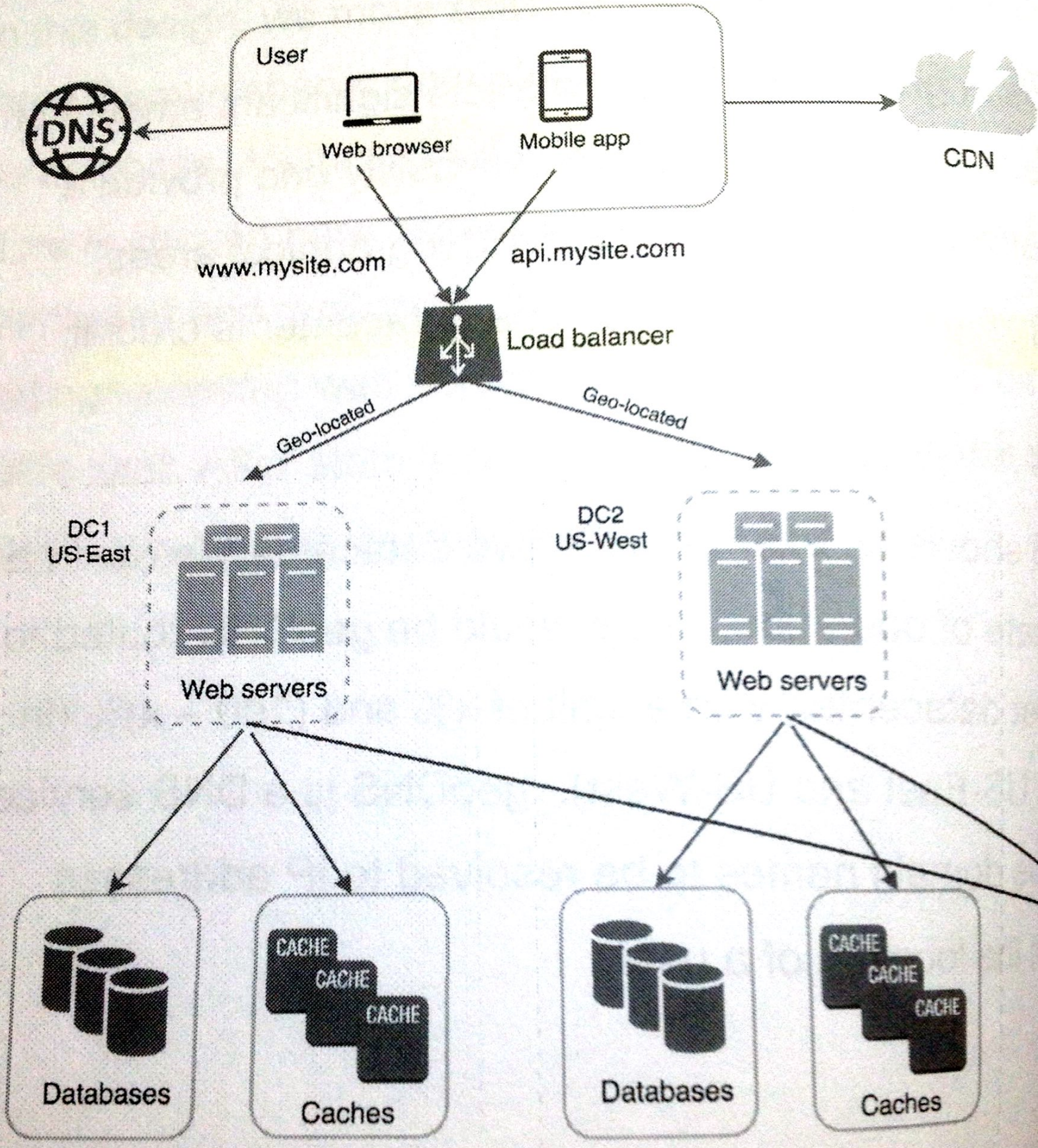

User
DNS
Web browser
Mobile app
CDN
www.mysite.com
api.mysite.com
Load balancer
Geo-located
Geo-located
DC1
US-East
Web servers
DC2
US-West
Web servers
Databases
CACHE
CACHE
CACHE
Caches
Databases
CACHE
CACHE
CACHE
Caches

Figure 15

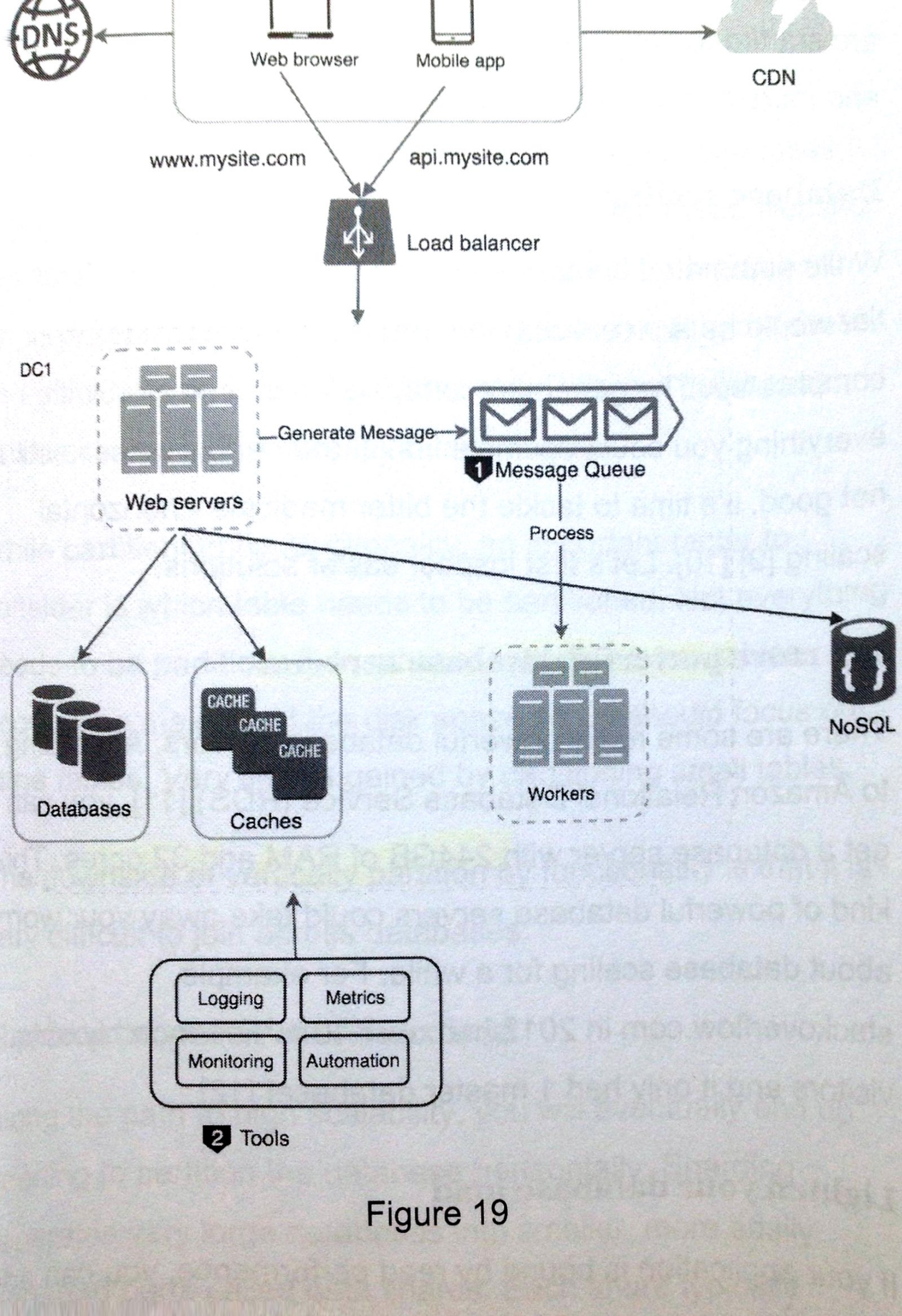

DNS
Web browser
Mobile app
CDN
www.mysite.com
api.mysite.com
Load balancer
DC1
Web servers
Generate Message
1 Message Queue
Process
Databases
CACHE
CACHE
CACHE
Caches
Workers
NoSQL
Logging
Metrics
Monitoring
Automation
2 Tools

Figure 19

Database scaling

While automated horizontal scaling (sharding) of the database tier would be an ideal solution, the implementation is complicated. The general recommendation is to start with everything you could optimize first. If the performance is still not good, it's time to tackle the bitter medicine - horizontal scaling [9] [10]. Let's first inspect easier solutions.

Get more powerful database servers.

There are some really powerful database servers. According to Amazon Relational Database Service (RDS) [11], you can get a database server with 244GB of RAM and 32 cores. This kind of powerful database servers could take away your worry about database scaling for a while. For example, stackoverflow.com in 2013 had over 10 million monthly unique visitors and it only had 1 master database! [12]

Lighten your database load

If your application is bound by read performance, you can add caches or database replicas (read from slaves). They provide

s unique to that shard. Sharding allows a database tier
scale along with its data and traffic growth. Many sharding
strategies allow additional database servers to be added.

Figure 20 shows an example of what a sharded database looks like. Each user data is allocated to a database server based on the user id. Anytime you want to access a user's data, you use a hash function to find the corresponding shard. In the following case, user_id % 4 is used as the hash function. If the result is equal to 0, shard 0 is used to fetch data. If the result is equal to 1, shard 1 is used. The same logic applies to other shards. Figure 21 shows what users table looks like in sharded databases.

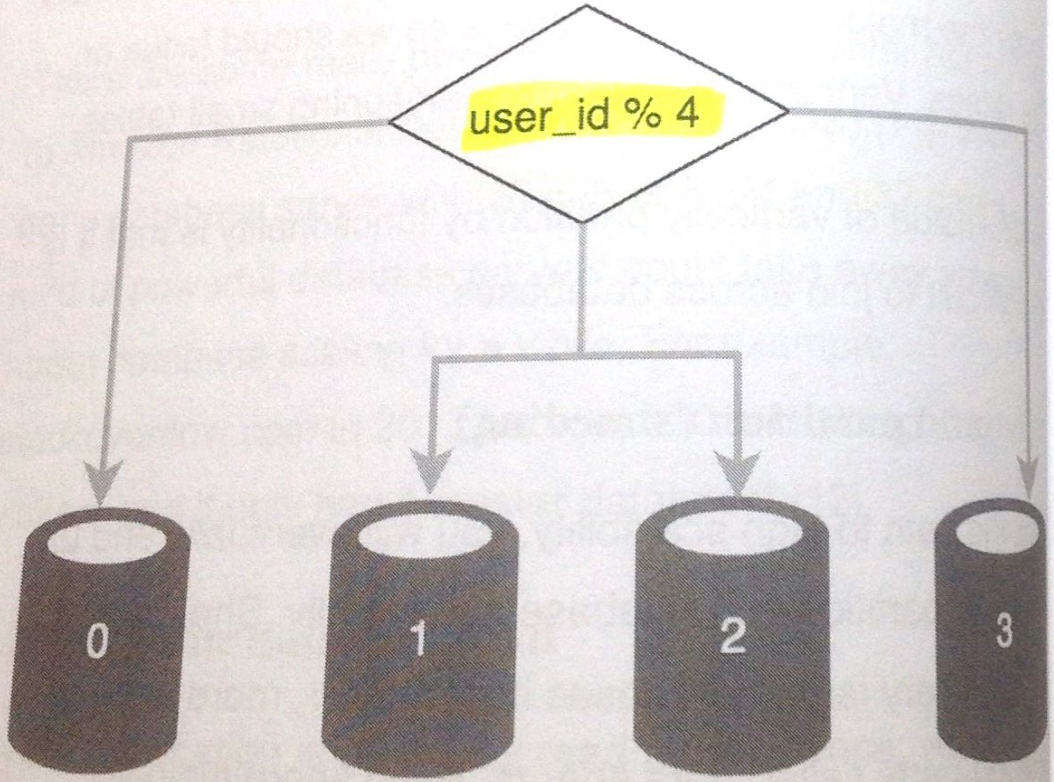
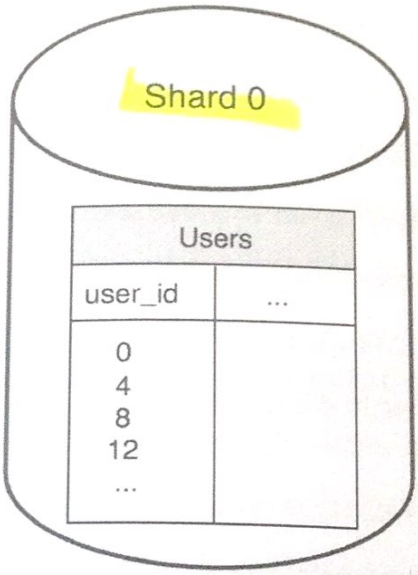


Figure 20 User



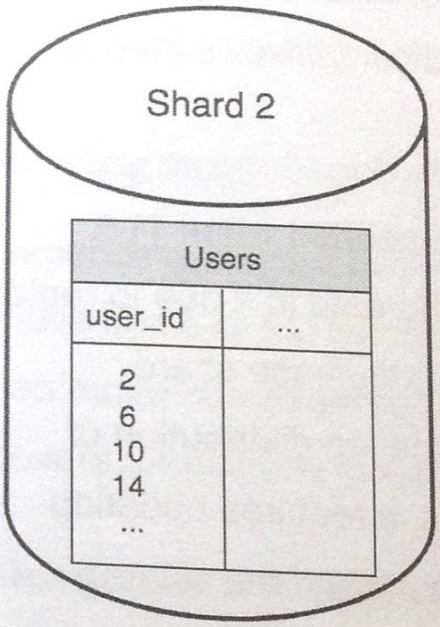


Figure 21 Users

The most important factor