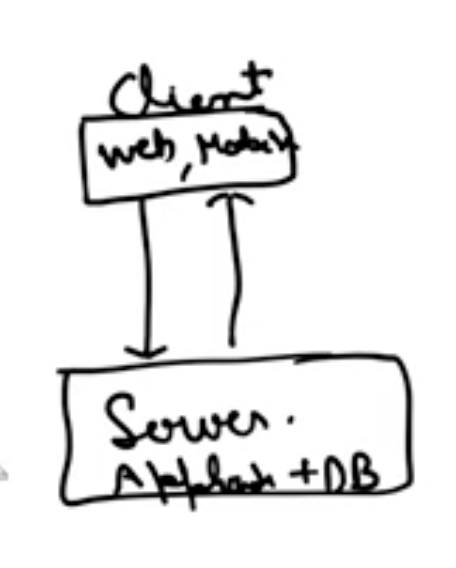
- Covers various concepts like sharding, horizontal and vertical scaling, load balancing, caching, and messaging queues.

### \*Steps for Scaling:\*

1. Single Server:

- Basic setup with a single server for the application, database, and client.

- Suitable for the initial stage with zero users.

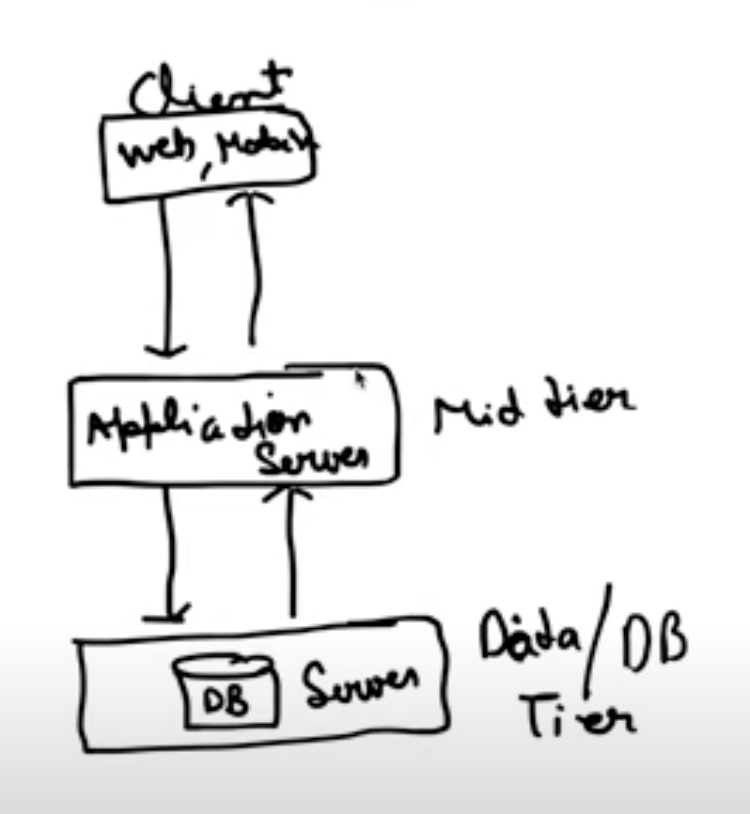


2. Application and Database Separation:

- Introduces a separate layer for the application server, handling business logic.

- Database server handles data storage and retrieval.

- Enables independent scaling of both application and database.

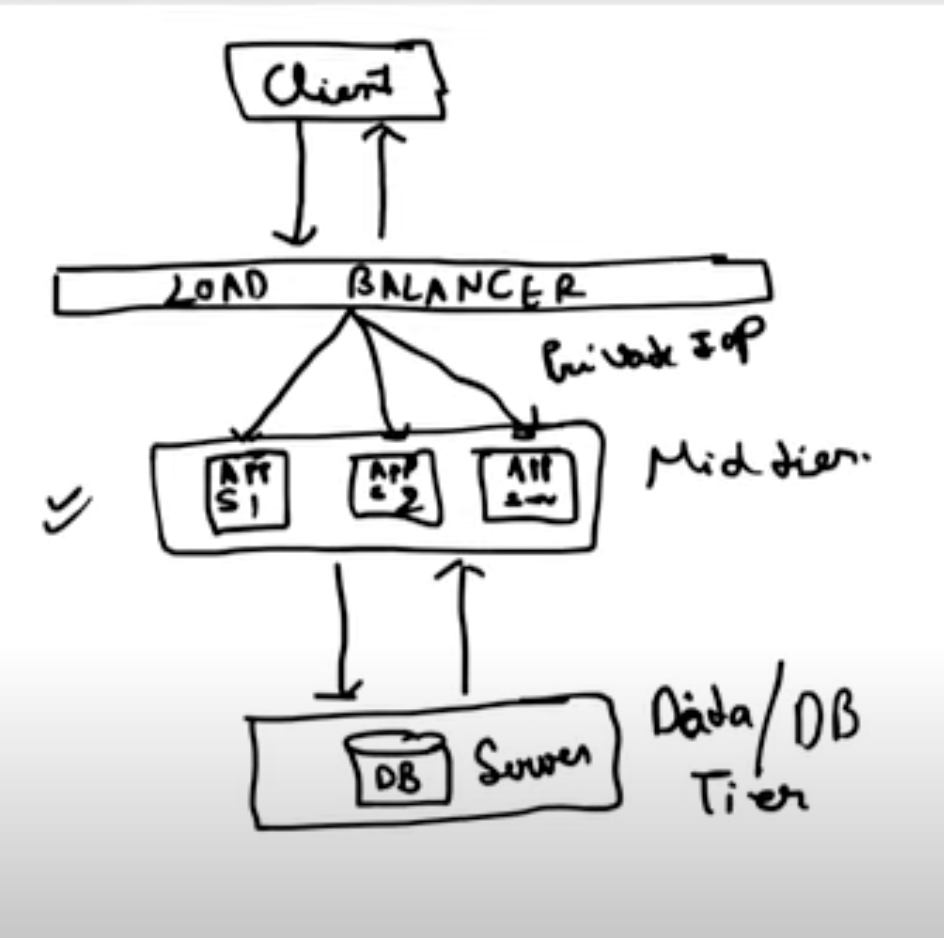


3. Load Balancing and Multiple Application Servers:

- Introduces a load balancer to distribute incoming requests across multiple application servers.

- Load balancer provides security and privacy.

- Ensures efficient handling of increased traffic by distributing workload.



4. Database Replication:

- Implements a master-slave configuration for the database.

- Master database handles write operations, while slave databases handle read operations.

- Improves performance and provides redundancy in case of database failure.

0 卪 
0 ß 

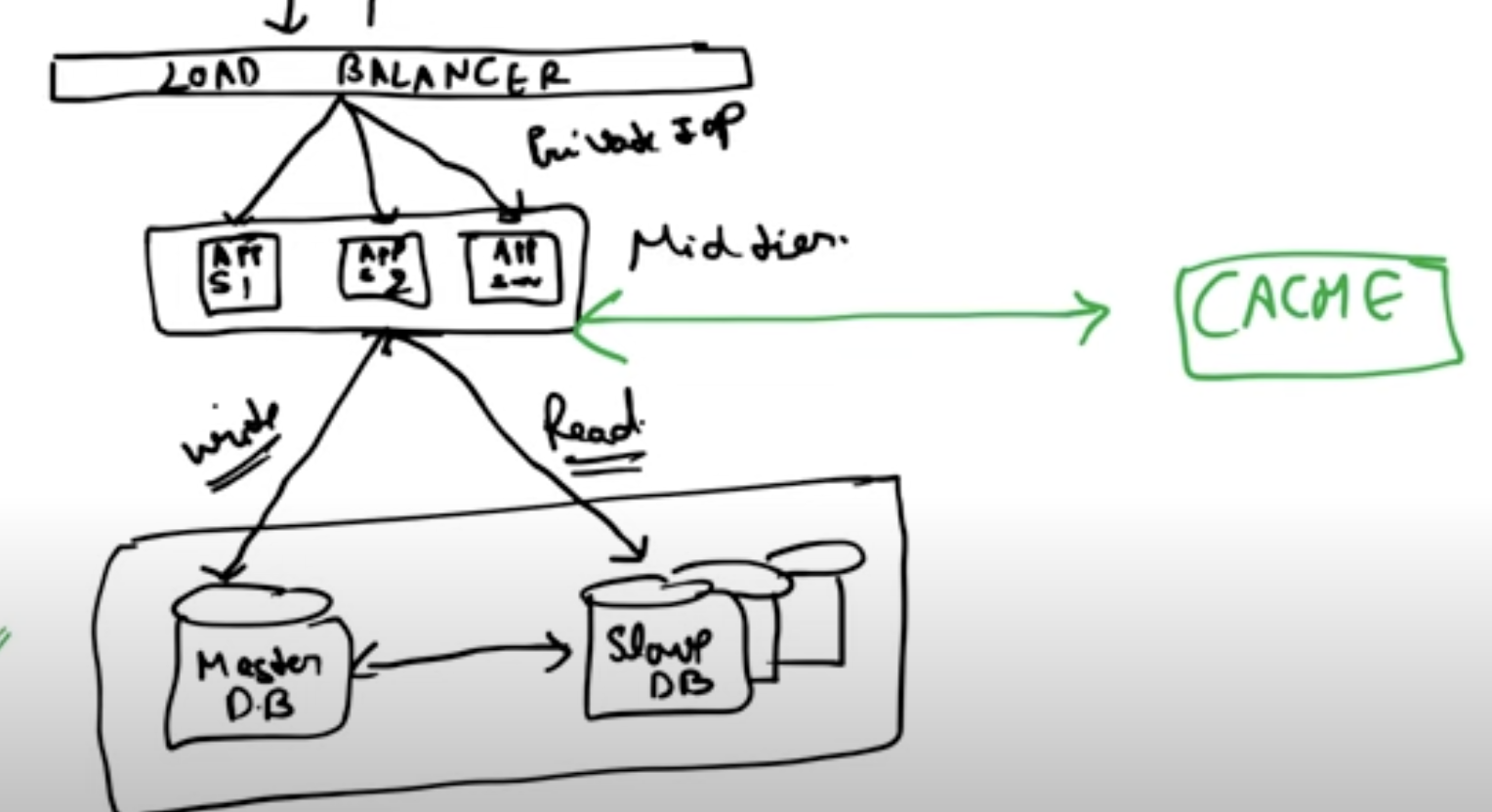
5. Caching:

- Utilizes a caching layer to store frequently accessed data in memory.

- Application server checks the cache first before accessing the database.

- Reduces database load and improves response time.

- Uses time-to-live (TTL) to manage cached data expiry.



6. Content Delivery Network (CDN):

- Uses a distributed network of servers to cache static content closer to users.

- Reduces latency and improves website performance for users worldwide.

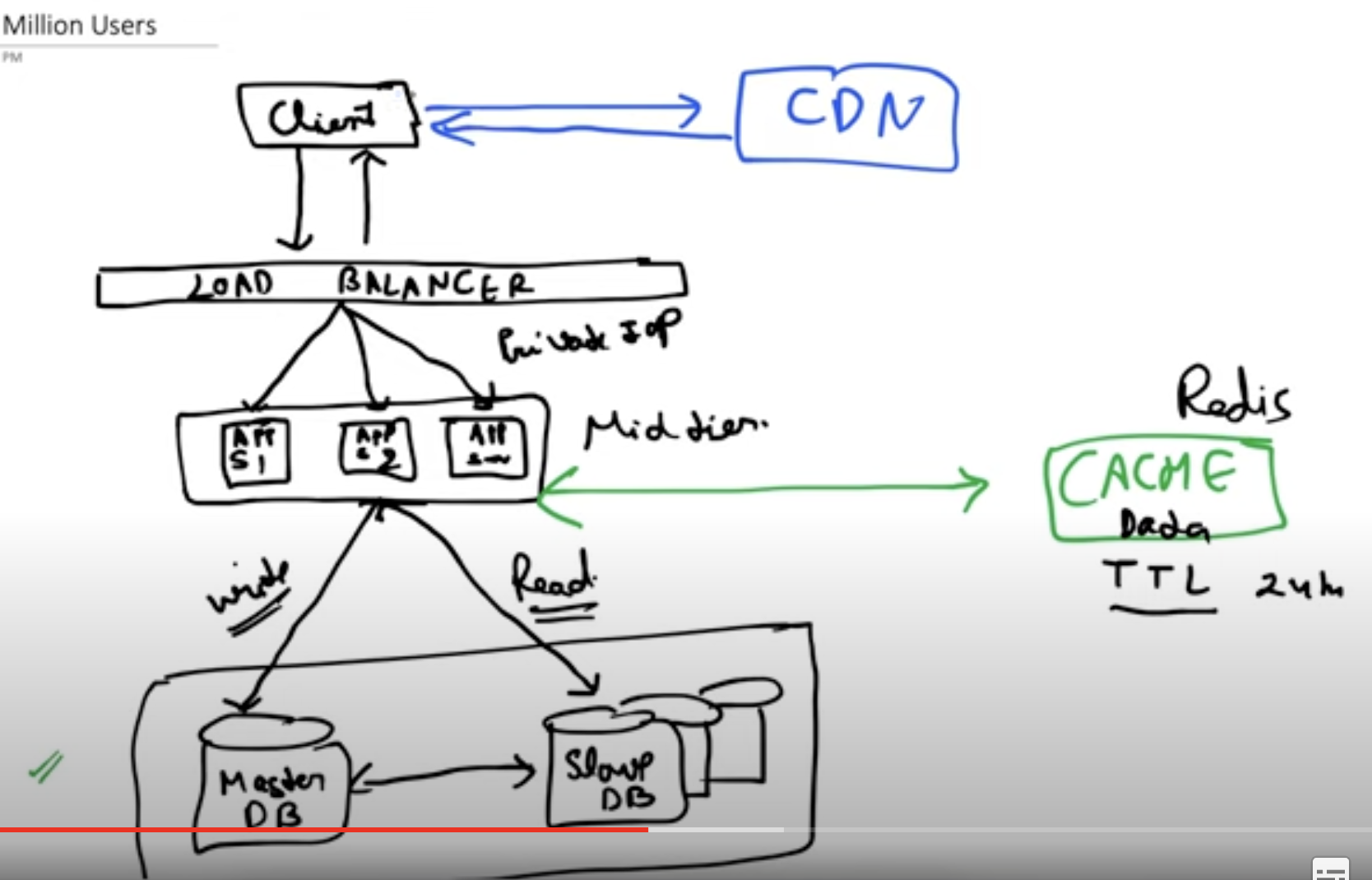
- Handles requests for static content like images, videos, and JavaScript files.

- In case of cache miss, CDN first ask neighbour CDN for the data then from origin DB.

+1 ЭА съел Ом 
Cpfv 

If that sdn doest not have have data then it ask from nearest CDN then ask from original DB

vidNT— 
NO つ 
三 ↑ 8 
0 代 工 
•JÖ つ 
ル / ・ 。 甲 へ 



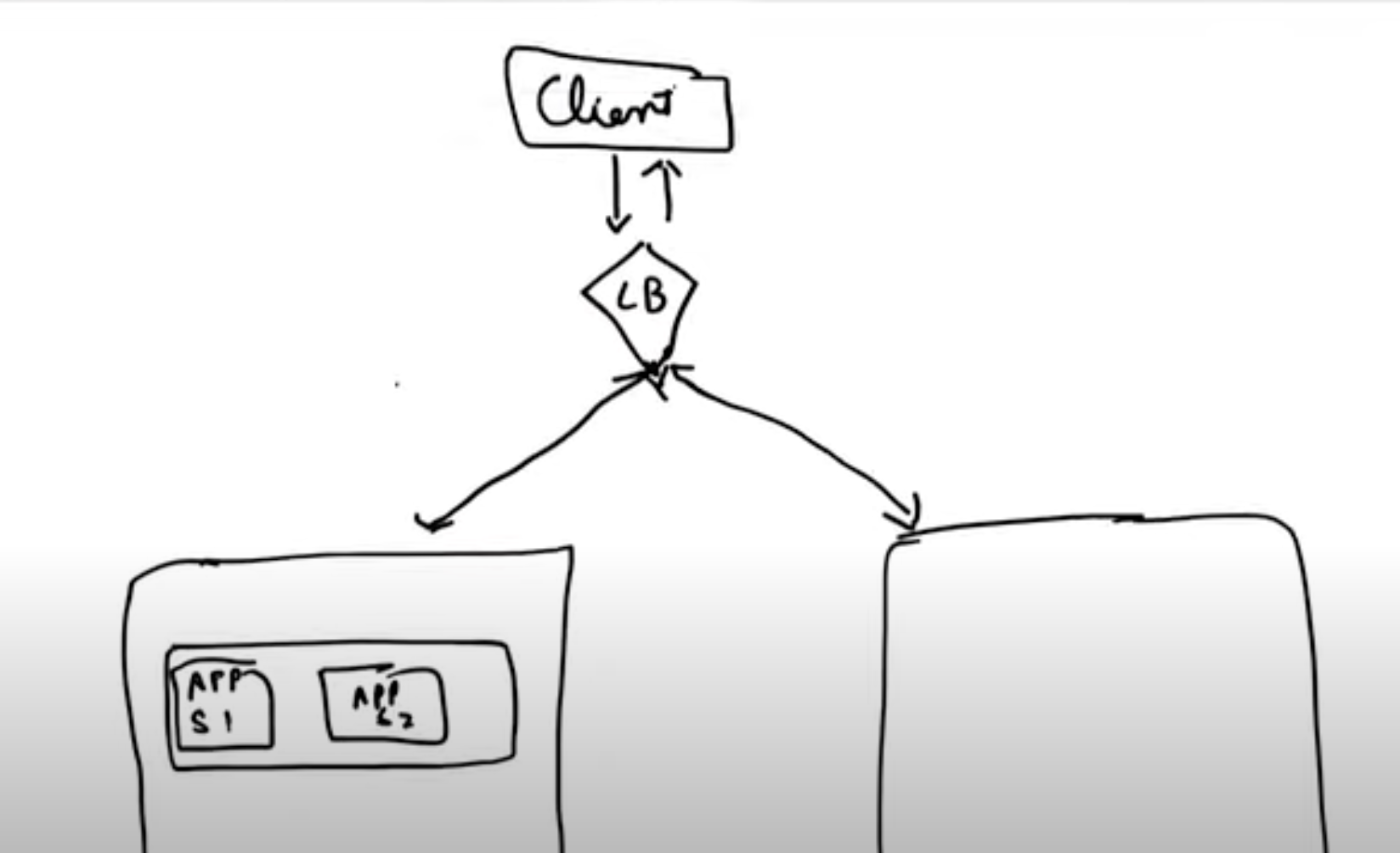
7. Multiple Data Centers:

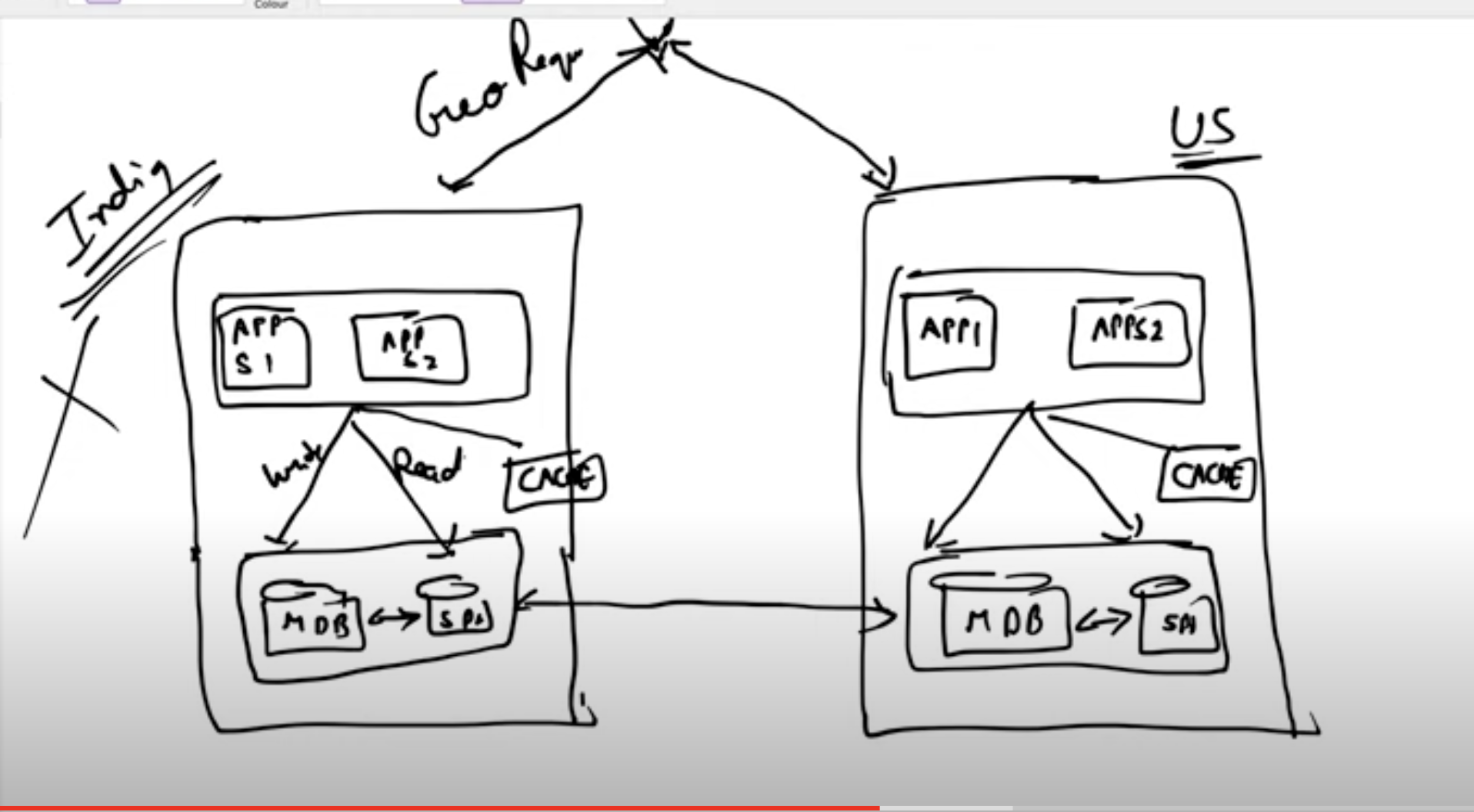
- Distributes the application and database across multiple data centers.

- Reduces load on individual data centers and improves reliability.

- Enables geographically distributed user access with minimal latency.

- Load balancer distributes requests to different data centers based on user location.





8. Messaging Queues:

- Uses messaging queues to handle asynchronous tasks like sending notifications or emails.

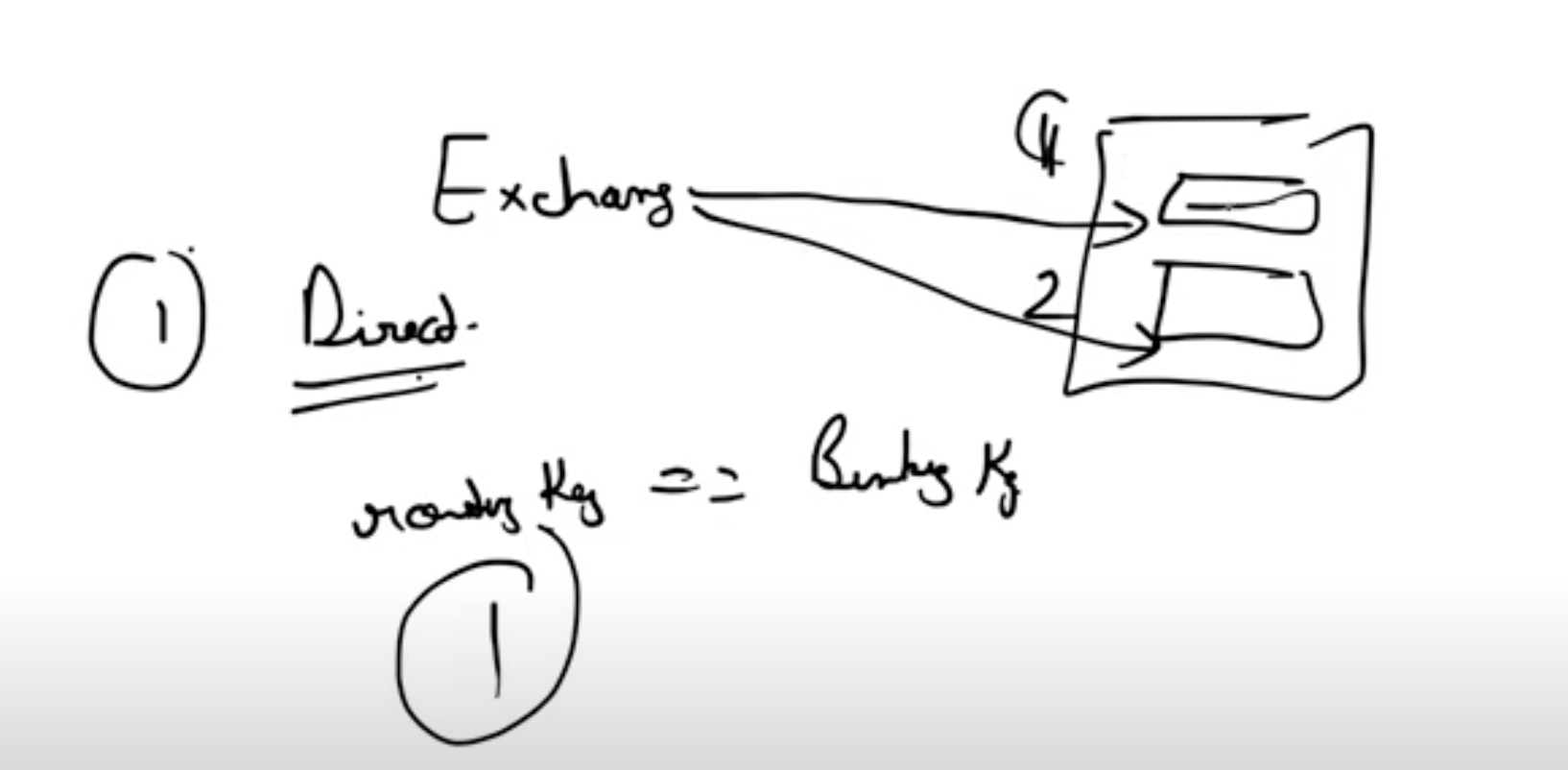
- Decouples tasks from the main application flow.

- Improves performance and reliability by handling high-volume tasks efficiently.

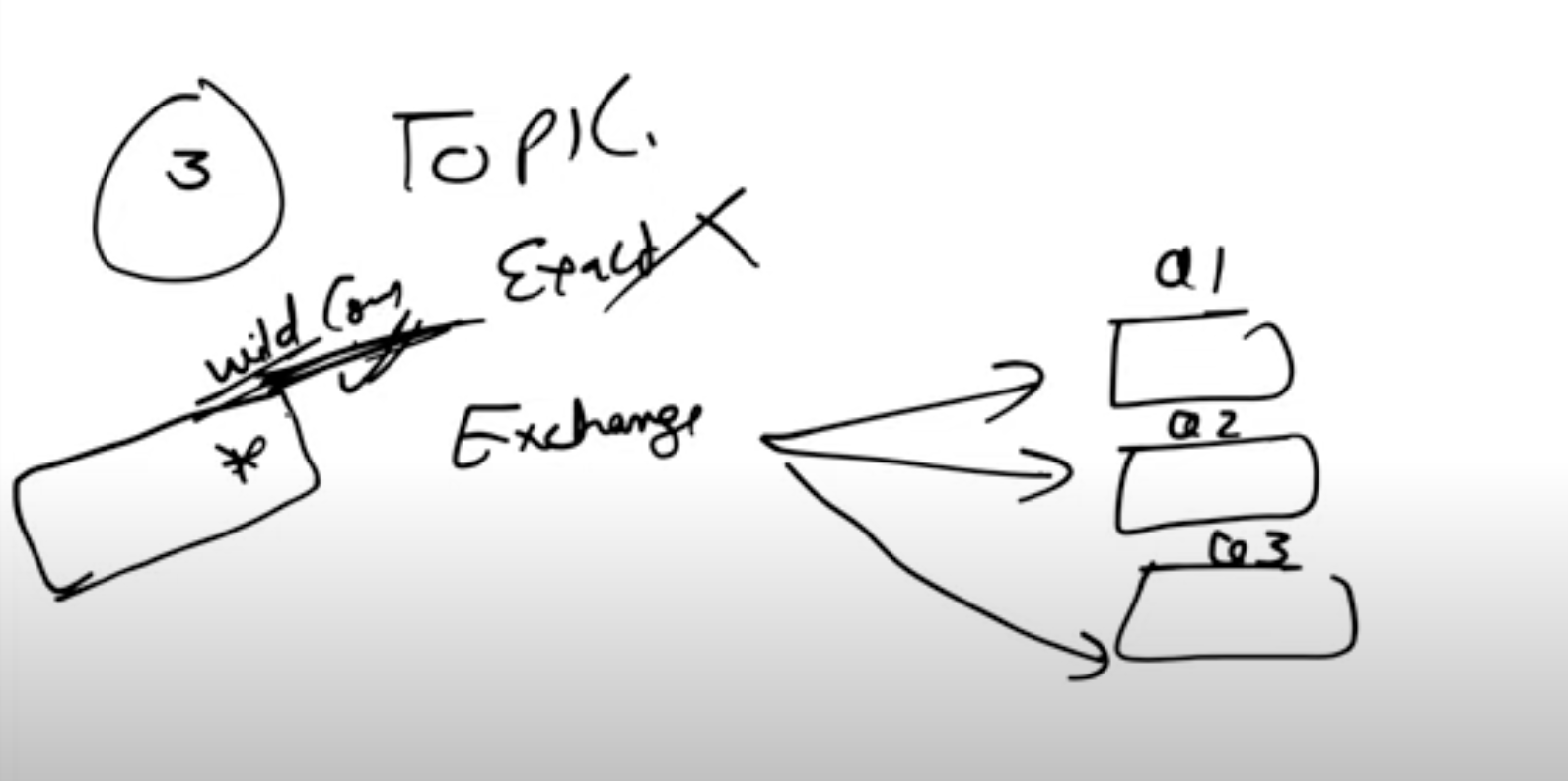
- Utilizes messaging platforms like RabbitMQ or Kafka.

دط م/ و 
حها يه ي، مسسال ك 
لسي ة 
هعله 

(ns 
zns 



) >Q へ 
し 肉 



إ٨١ 
دك 
aæ 
قع فكا 

9. \*Database Scaling:\*

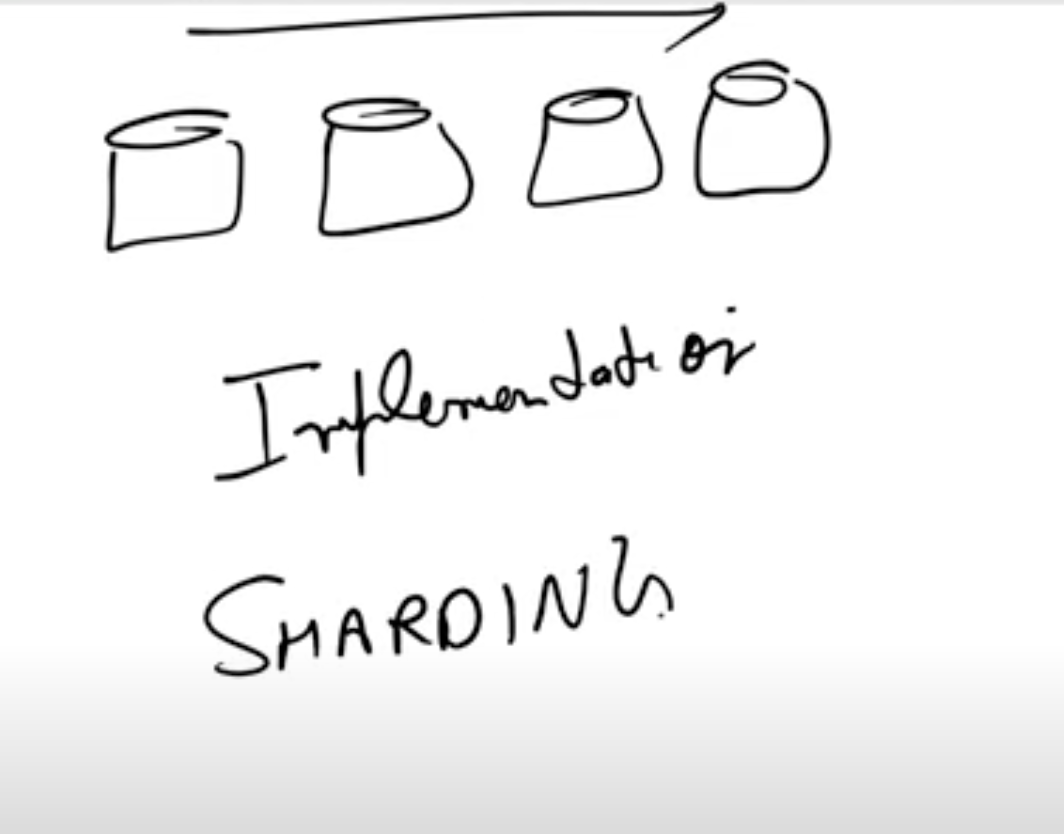
- \*Vertical Scaling:\* Increase the capacity of existing database servers (CPU, RAM). This has limitations and eventually reaches a ceiling.

- \*Horizontal Scaling / Data Sharding :\* Split the database into multiple servers or shards, distributing data across them. This offers better scalability.

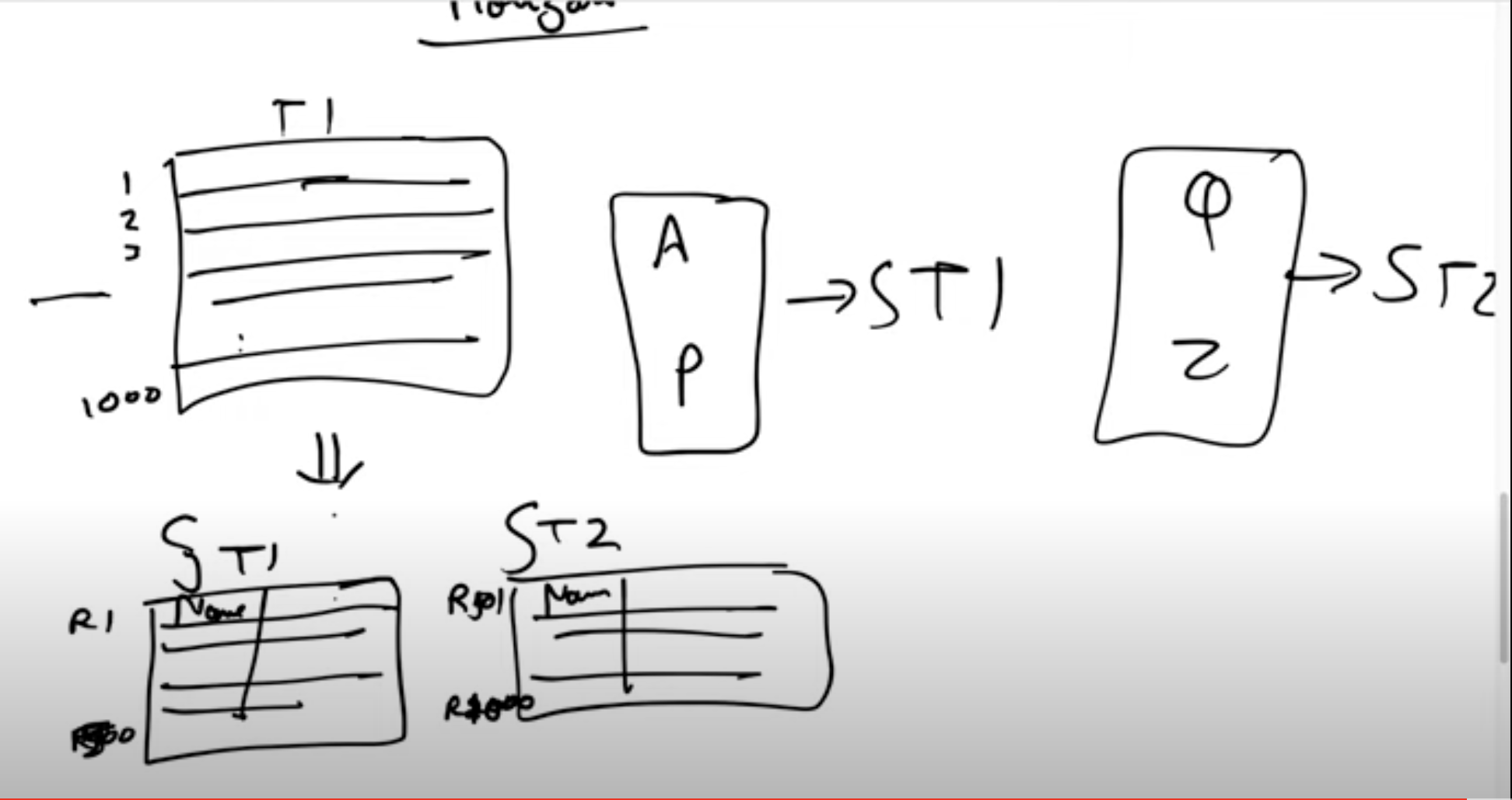
- Splits data across multiple databases or tables based on a specific key.

- Can be implemented through vertical sharding (splitting columns) or horizontal sharding (splitting rows).

معلا ث 
ن٣/ 
لا ممى 
rønoø 



مه مهله 
لعمل C بـ 



@vertical sharding

Сю 
cS 
ср 

