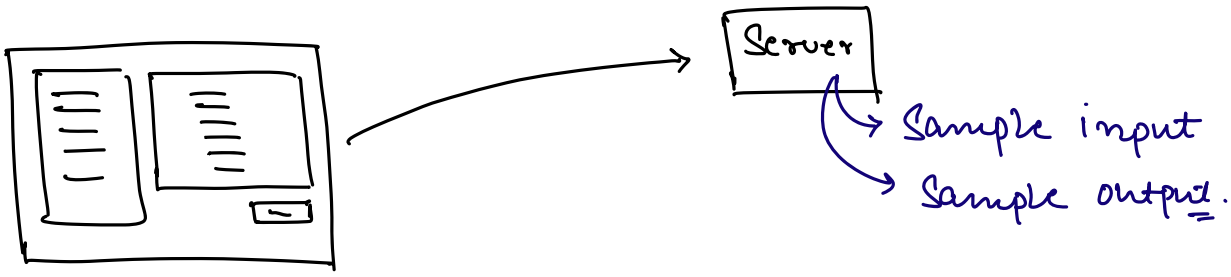
Agenda.

→ Scaler Code Judge.

→ Scaler Contest Leaderboard.

#



Sample Input file:

(100 TC) * (10^6 Array Elements) * 8 Bytes.

800 MB ~ 0.8 GB.

~1 GB.

Output file ~1 GB.

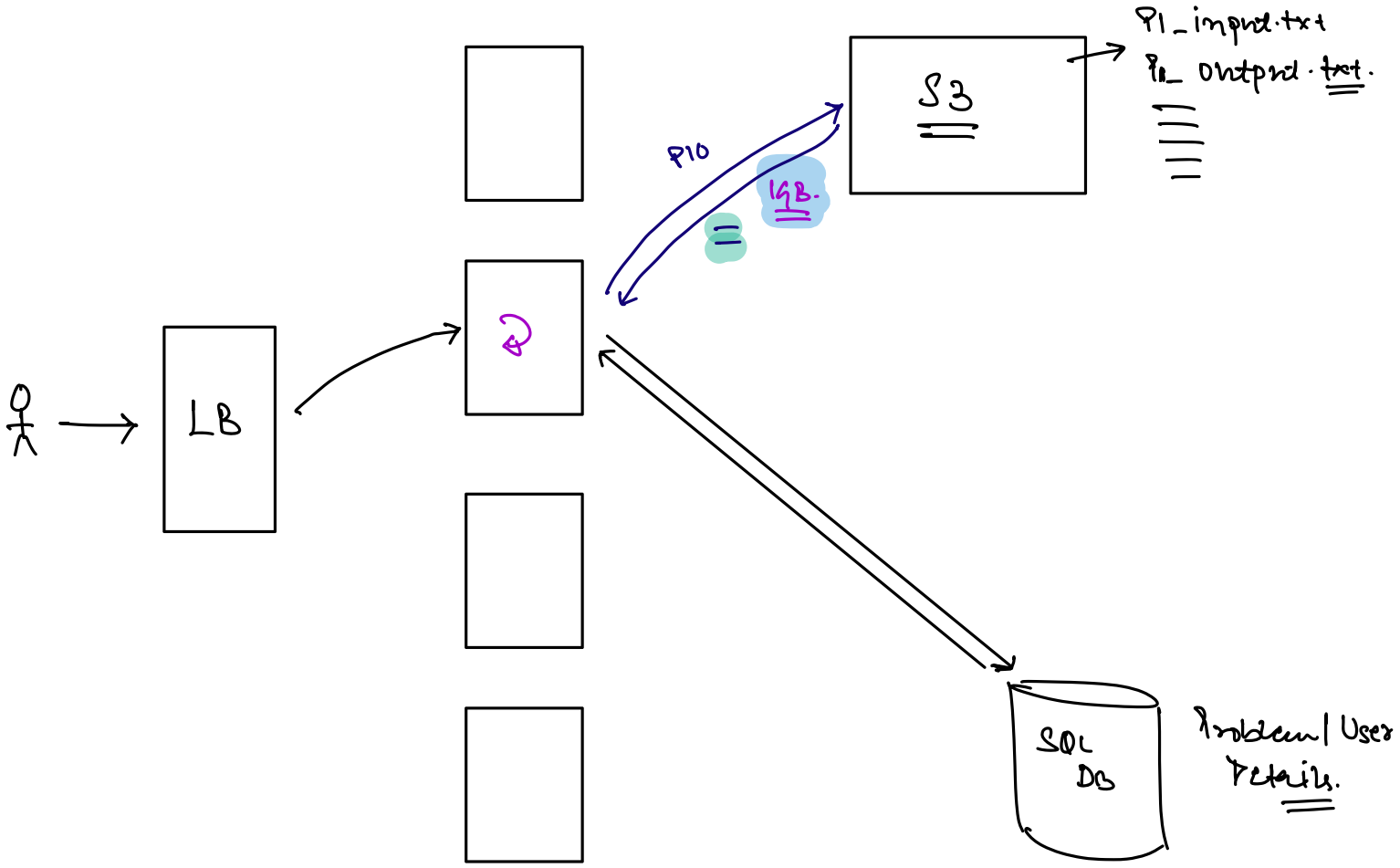
For 1 problem : ~2 GB.

Total # of problems : 2000

Total space = 2000 x 1 GB

= 2 TB.

AWS S3 | Azure Blob Storage.



Need of Caching. ✓

↳ lot of data trasfer is happening over the n/w.

latency ↑

N/w Bandwidth ↑

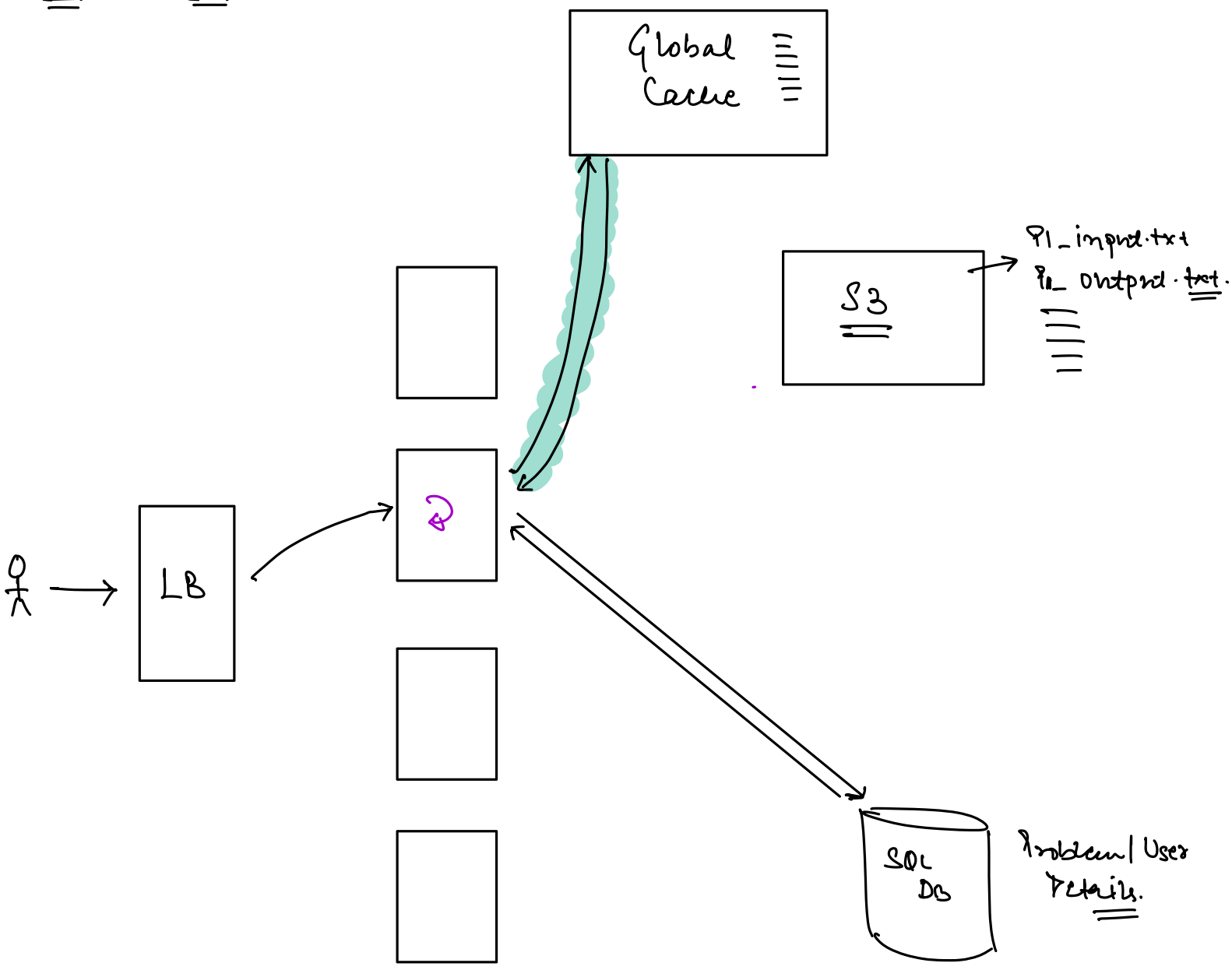
CDN. → ✗

↳ 3rd party service for Client side caching.

Backend Cache.

Global Local.

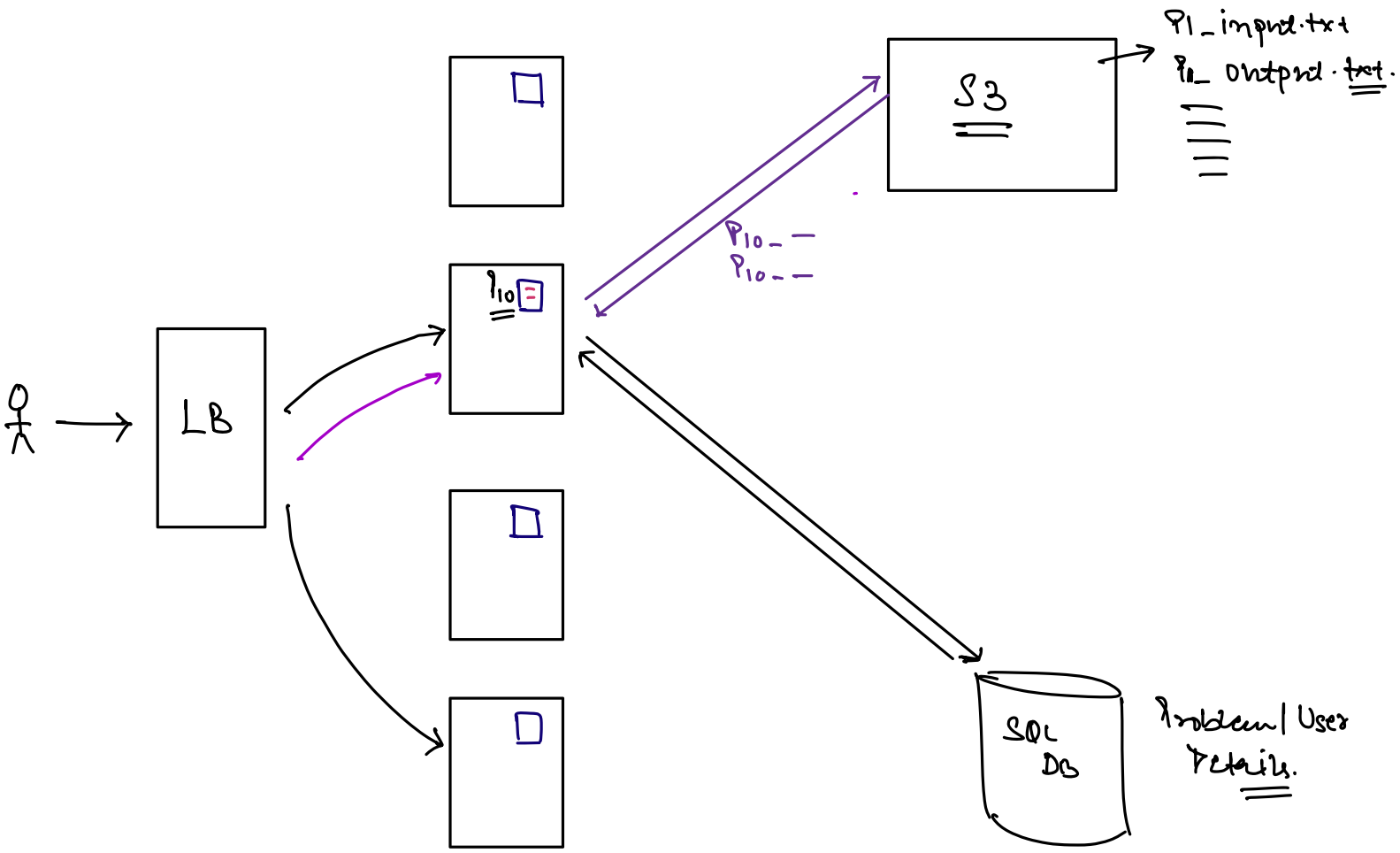
Global Cache



Global Cache isn't solving the problem of huge amount of data transfer over the n/w.

Local Cache.

→ Cache the TC files in the local HDD.



Local Cache

↳ by default distributed.

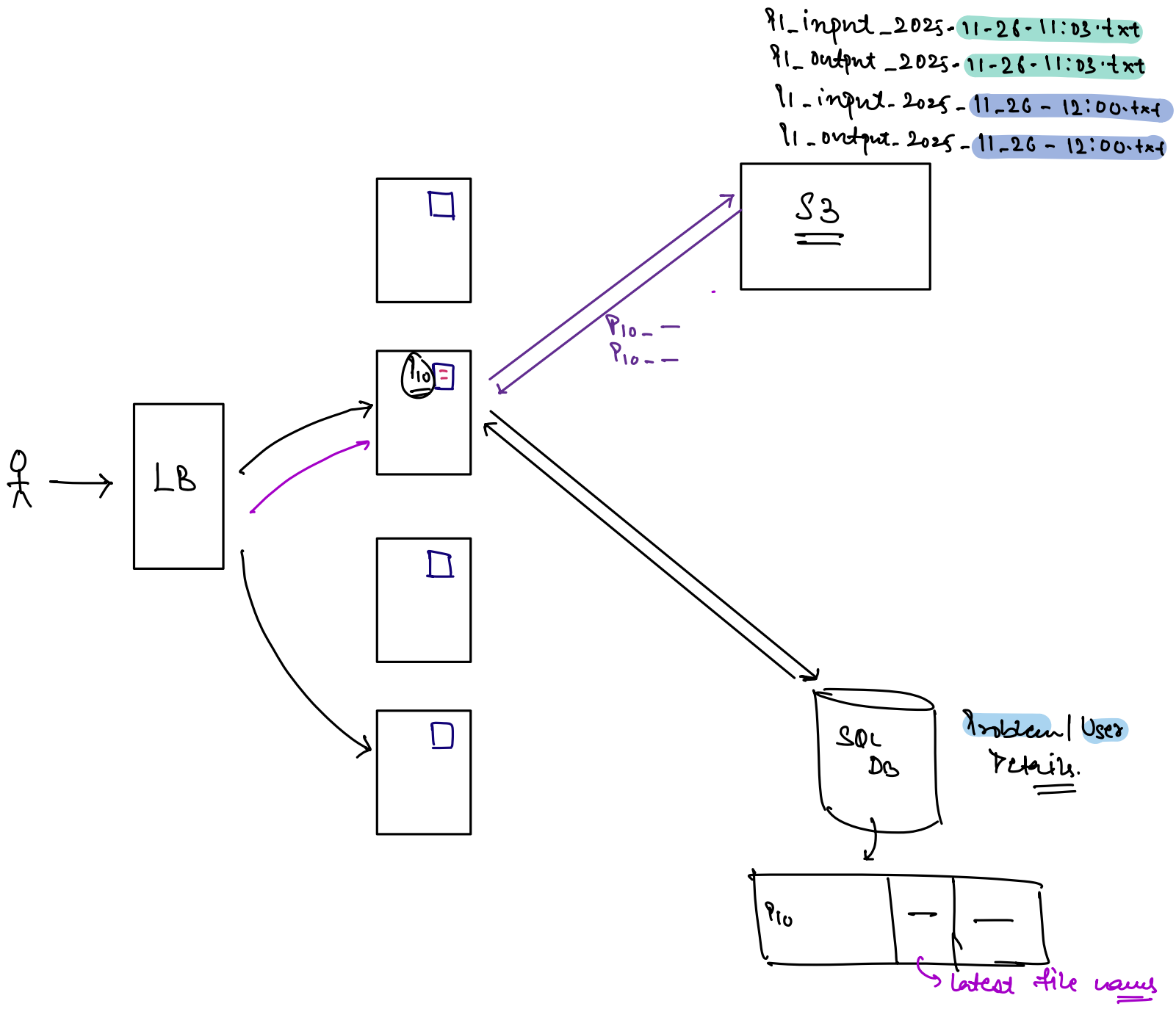
Cache Invalidation

Immediate (VS) Eventual Consistency.

TTL.

- 1D
- 2D
- 10 mins
- 10sec → Too many n/w calls.
- 1hr

Write Around | Back | Through Cache.



Eviction Strategy.

↳ LRU.

Load Balancer.

↳ Based on Problem Id

↳ Some of the m/c might get overloaded during the contest.

→ Based on userId. → ✗

→ Round Robin ✓.

Scaler Contest Leaderboard.

↳ 100K participants.

→ Paginated

→ username | rank | total score | score for each problem | time.

Usecase

→ 3 hours contest

→ 100K users

→ 5 problems.

Avg # of submissions / user / problem = 2

Total # of submissions = $2 \times 5 \times 100K$

$= 1M.$

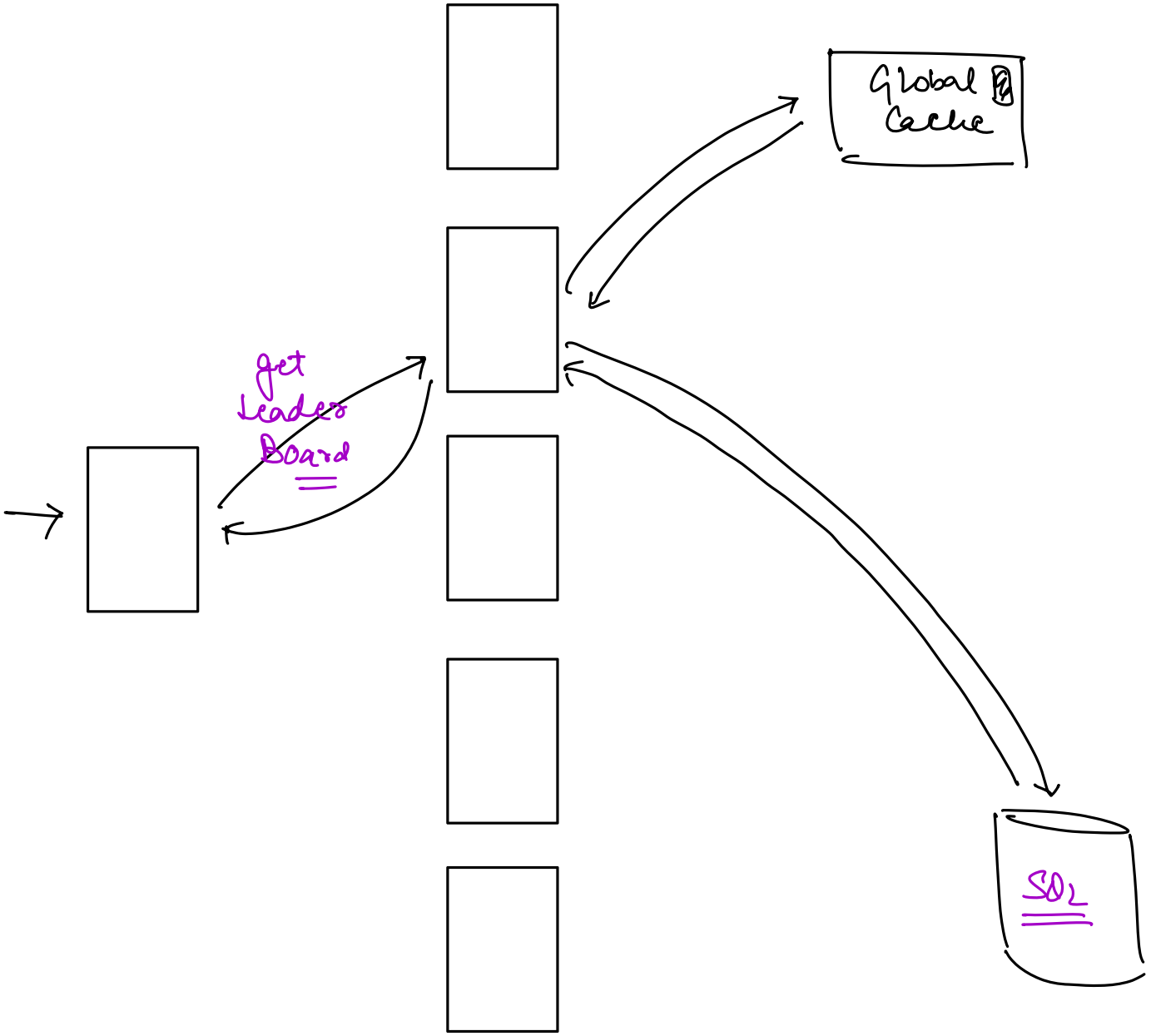
# of submissions / sec $= \frac{10^6}{3 \times 60 \times 60}$

$= \frac{10^6}{10^4} \approx 100$

100 submissions / sec

Peak traffic = 2 × Avg traffic

= 200 submissions / sec



⇒ Generating a leaderboard is time consuming process.

↳ Can take easily ~ 2-3 seconds.

Need for Caching

Local (VS) Global Cache.

Size of Data = (200 B) × 100×10^3

(Per user)

= 20 MB.

# of read requests = 20 views/user for 3 hours

$$= \frac{20 \times 100K}{3 \times 60 \times 60} \text{ views/sec}$$

$$= \frac{20 \times 100 \times 1000}{10000} \text{ views/sec}$$

= 200 views/sec.

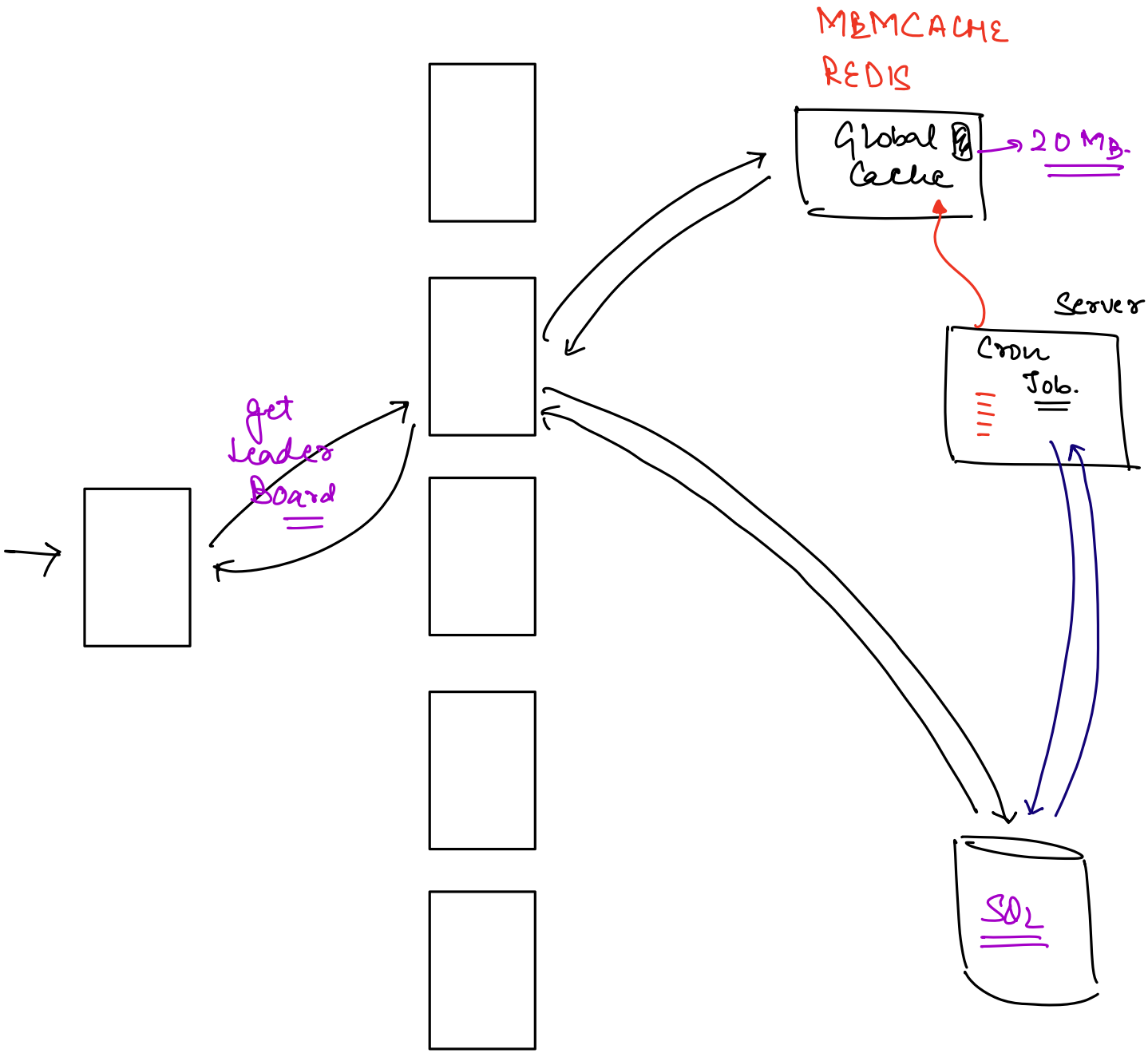
Distribute.

↳ No.

Cache Invalidation.

↳ (TTL) ⇒ 10 mins.

Scaler has TTL of 30 mins.



Eviction Strategy.

↳ We don't really need to evict as data is just few MBs.

LB.

↳ Round Roubin for App Servers.

Points to note while designing a Cache

- Need for caching
- Local (vs) Global
- Distributed
- Invalidation, Writing & Eviction Strategies.
- Load Balancing.