1. How many socket connections are used by game servers? How many of these nodes are needed to handle peak load? Assume 1 million daily active users.

1 million active users daily gives us active users per minute as:
10^6 / (24 *60) = 900 users/minute.

Since games last for roughly five minutes, we can assume a load of 5000 connected users at any point in time.

But the peak time loads could be higher (India/US night time). We can assume this to be roughly 2x of average load = 10k connected users.

Also, if there is a special event like the World Championship or a chess meme going viral, we can expect a surge of new users. This may double the load to 20k connected users.

So our servers should be able to handle 20k connected users.

20k connections can be stored in a **single large instance** of a typical cloud service like AWS or GCP (to be safe, we may provision a 32 GB RAM instance with 8 cores).

However, for low latency and redundancy, we may want to have multiple instances of the game service running on different machines in different availability zones. **3 servers** deployed in different regions like US, India and Europe would keep latency low in all regions.

2. How much compute capacity will be needed by the matching engine? How much capacity is needed to analyse past user games?

We have at most 20k users at any given point in time. That means we can have roughly 20k match requests in the waiting stage. If we assume each request to have size 1KB, the total memory required for these requests is approximately 20K * 1KB = 20 MB.

Using a balanced BST datastructure to find matches takes O(logN) time.

That's 20k * log (20k) ~= 20k * 15 = 300k instructions.

This will take less than a second on a cloud instance.

Analysing past games is trickier. We can try to do this in batch, using a data pipeline architecture similar to that discussed in the previous chapter (Design an analytics engine).

With 1 million active users, let us assume we have 5 million active games. If a game has 30 moves on average, that's 5*30 million = 150 million moves per day.

These are **too many moves to analyse**. To reduce the load on the server, we could offer the following choices:

a. Analyse on the client (run engine on browser on client request)
b. Low resolution analysis on server (preferably on request)
c. Analyse only premium member games.
d. Analyse only high rated member games.

Let's assume we choose the top 0.1% of the games to be analysed. That's 150M / 1000 = 150K moves / day.

Each move is examined for 1 second on a single computer core.

The number of moves per second are:
= 150K / 86400 moves per second.
~= 2 moves per second.

So we need a dual core machine running at 100% capacity to analyse 0.1% of all games.

If you want to analyse games in a stream (where peak load varies), using multiple dual core computers is a faster and safer option.

3. How much time does it take for a cache server replacement to come online? Could this result in a negative user experience?

If a cache server goes down, a replacement server will have to be arranged and get ready as soon as possible.

Assume it takes us 3 seconds to identify a dead cache using heartbeats, and replace it with a new server.

This server will pull all relevant data (active games IDs that it must serve) from the database. The information of which game IDs it must pull is provided to it by the cache load balancer/coordinator. It mentions the range of keys this replacement cache must handle during its startup.

Assume it takes 2 seconds to load all keys from the database.

By the time this cache is ready, but there may be a large pile of waiting requests for it. Each of these requests can be answered with a single lookup in the cache, so assume maximum response time to be 1 second.

Total time to respond for the cache = 3 + 2 + 1 = 6 seconds.

In a chess game, this is a significant amount of time and **will affect user experience**.

4. Calculate the difference in network and compute power required by game servers forcing client-server validation vs. allowing peer-peer game moves.

To ensure all moves made by our clients are valid, we run them through the server. This means roughly 20k moves are validated every minute.

Assuming a validation operation take 100 instructions, that's 2M instructions being executed every second. This would take approximately **0.1 seconds / second** to run. The compute power to validate moves is **manageable**.

Each move is sent to the server for validation through a network call. Assume a move request is 1 KB in size. Thats roughly 1KB * 20K requests = **20 MB / second** will be sent to the server. Hence, the additional network requirement to validate moves is also **manageable**.