The way I see it, the biggest problem with doing too many synchronous writes in the database is ends up creating a big traffic jam. This is what people call **database I/O contention**.

Now what does that mean?

When a write is synchronous, our app sends a command to the database and then just waits. It won't do anything else until the database says, "Okay, I've saved it to disk." So during that time, the app is just sitting idle.

Let's assume the database is like a single cashier in a busy shop. Every write request is a customer. The cashier can only handle one person at a time. If suddenly 50 people come, everyone else has to wait in line. That's what contention is.

From here, a few problems start:

1. **Long Queue:** Requests stack up since only one can be processed at a time.
2. **Locking:** To keep data safe, the database may lock rows or tables while writing. Like the cashier saying "hold on, don't touch anything until I'm done." Even reads get blocked.
3. **Lag:** Because of all this waiting, everything feels slow for users. That usual "app is too slow" feeling is mostly database bottleneck.
4. **System Slowdown:** Since everything depends on writes finishing, our system's overall speed drops. It doesn't matter how good my code is. Performance depends on how fast the disk can write.

So, in short, too many synchronous writes make our database the poor cashier who's overloaded. And if we want to scale, that cashier becomes our biggest roadblock.