We can use technologies such as Node.js, Redis, MongoDB / MySQL, AWS EC2, Message Broker and ELB to achieve millions of concurrent requests. We can have many EC2 instances, and each instance is running one or more Node.js programs. Use load balancing to distribute user requests.

The reason why I choose Node.js is that it is non-blocking, lightweight and flexible that makes building robust APIs and middleware very easy. Compare to using Spring framework where we can easily have hundreds of memory in use, Node.js has low memory utilization. Use Redis to store commonly used data in the cache: parse JSON data and store it in Redis, where Key is the time period in hours, and Value is the restaurant opened at that time. In this way, users can get the data users want without hitting the database, without latency. If the user wants to update the data in the database, simply use Mongoose to update the data into the MongoDB. If there are many user update requests, you can add Message Brokers to the system, such as RabbitMQ, Kafka, to store these requests, reducing the pressure on the database during peak hours. With the expansion of business, the database may need to do index optimization, database shard, etc. to meet the requirement of storing massive data, improve the performance and scalability.