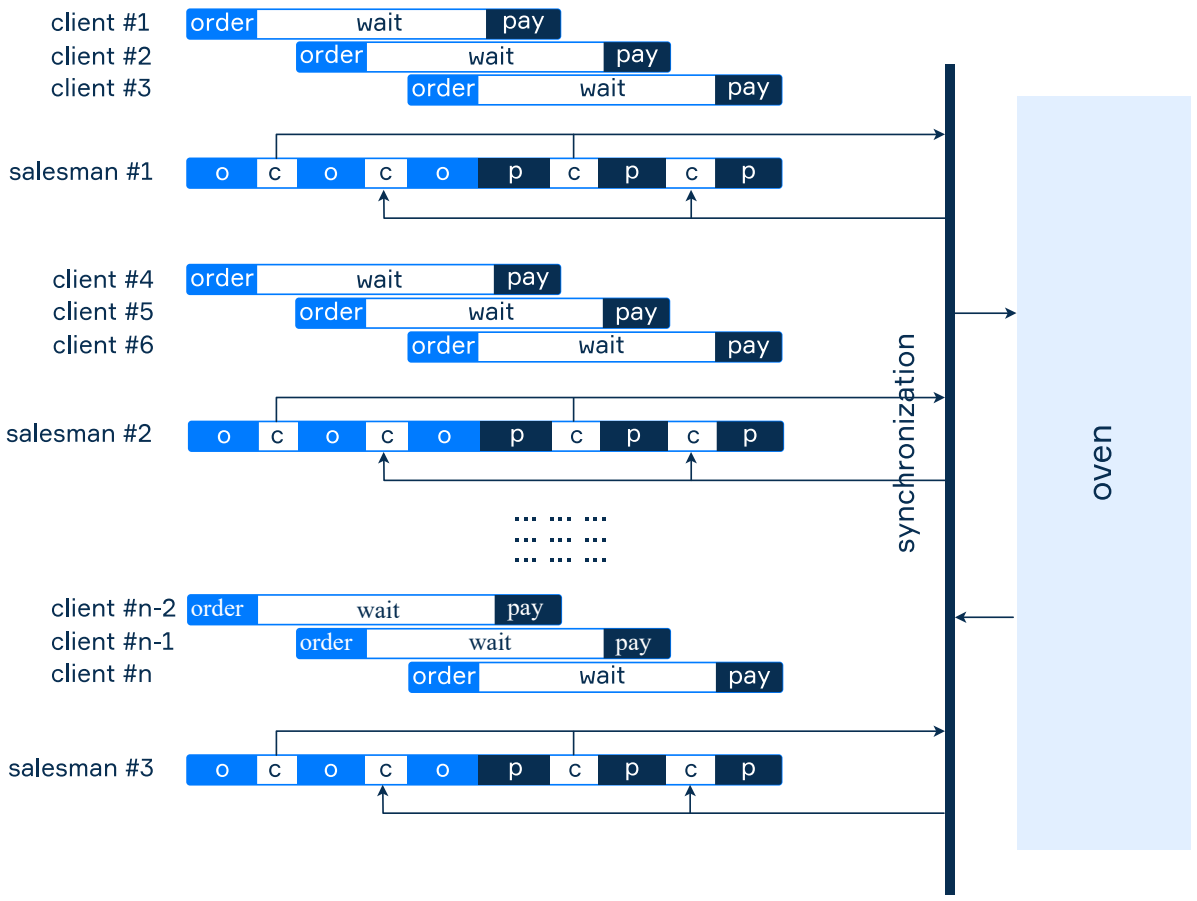
Theory: Concurrency and parallelism

15 minutes 0 / 5 problems solved

As we already know, there are three types of workflow: synchronous, asynchronous, and parallel. The synchronous workflow runs consequently step by step. Asynchronous and parallel workflows operate simultaneously. The way to perform tasks in asynchronous and parallel workflows is called concurrency and parallelism, respectively. Let's continue studying the concurrency and parallelism using an example of the pizza shop workflow.



A typical pizzeria from the real world would have large professional ovens. Several workers share them to cook many pizzas simultaneously. Large ovens are expensive but much more productive compared to small ones preparing just one pizza at a time. These ovens allow each worker to serve several clients simultaneously in overlapping periods.

§1. Critical section and synchronization

Let's look at the oven. It's roomy but not infinite. Sometimes there is no free space, and workers can't put more pizzas inside, so they have to take place in the queue and wait until it appears. This situation is an example of synchronizing the access to the oven's places. The oven in this context represents a critical section.

Critical section is a limited resource required for task execution. It's *critical* because the process can't go further without passing it, so the worker *must* get access to this resource to accomplish the task.

Managing access to the critical section for concurrent tasks is called synchronization because it's blocking and consecutive or, in other words, *synchronous*.

§2. Concurrency

Each seller takes an order and puts a pizza in the oven. Then, while pizza is cooking, the worker takes another order or gives out ready pizzas. Each seller serves several customers and plays different roles (cashier, baker, seller) simultaneously but by parts at different points in time. This approach is called a time-slicing, and this whole image is an example of concurrency or multitasking with time-slicing.

Current topic:

Concurrency and parallelism

Topic depends on:

× Synchronous, asynchronous, parallel

Topic is required for:

Threads as objects

Table of contents:

Concurrency is an ability to execute different tasks simultaneously in overlapping periods and no predefined order in an environment with some scarce shared resources required for task execution.

Concurrency appears when there is a scarce shared resource (critical section), and workers compete for access to it. For concurrency, it's necessary to have:

- more than one task in operation
- at least one critical section for these tasks

There is no such thing as concurrency for one task, it takes two to tango, so the second name of concurrency is multitasking.

§3. Parallelism

Workers in a pizzeria do similar work, but each with their clients. They perform it independently, and that's how an example of parallelism could be presented.

Parallelism is an ability to process tasks simultaneously by different executors. These can be either several independent tasks or one big task split into smaller subtasks and divided between workers.

Parallelism is, in some sense, a concurrency without a critical section. If shared resources are sufficient for all participants, processes can run in parallel. They don't need to compete for resources and get access to it by turn. If we distribute tasks among several executors so that everyone has only one, then there are no more critical sections. Concurrency disappears, and parallelism takes its place. Tasks run simultaneously, and executors don't compete for critical resources.

The presence of multiple executors is an essential condition for parallelism. There are many executors processing tasks in parallel, so the second name of parallelism is multiprocessing.

§4. Concurrency and parallelism: together and apart

Concurrency can appear with one executor and with multiple. In contrast, parallelism appears only when there are multiple executors. Processes with running concurrent tasks inside can be parallel. Parallel tasks can become concurrent and so on. Practically, every combination of concurrency and parallelism is possible.

For example, when you need to combine the results of several parallel tasks into a single whole, a critical section appears. It's represented by a procedure that uses the results of all parallel tasks at once. After each calculation of those tasks has been finished, it gives out the result in a concurrent manner with other tasks.

§5. Conclusion

As you can see, concurrency and parallelism are not the same.

- Concurrency is about dealing with multiple tasks at once in the environment with shared resources.
- Parallelism is about processing a lot of work by several executors in parallel.
- Concurrency and parallelism often intersect and appear together. They can mutate one to another.
- Practically, all combinations of concurrency and parallelism are possible.

Start practicing

Comments (9) Hints (0) Useful links (2) Show discussion