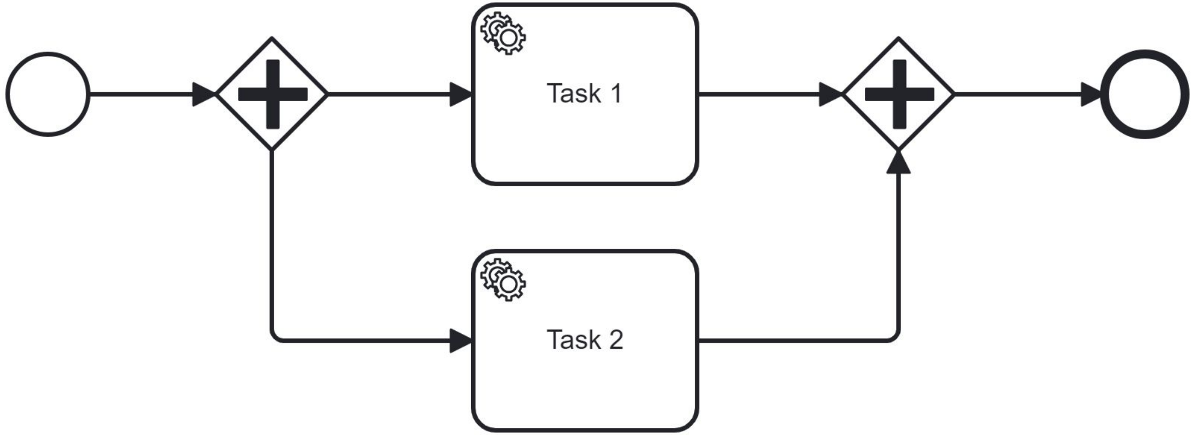
Job Executor

Parallel Join

For each branch arriving at the parallel join, we need to take a decision whether to **wait for the other executions** or whether **we can move forward.**

This requires synchronization between the branches of execution. The engine addresses this problem with **optimistic locking**



Optimistic Locking

To prevent two database transactions from overwriting each other, Camunda uses a **revision** number (like a version counter) on the execution record.

- Execution A reads parent revision = 5
- Execution B also reads parent revision = 5
- Both try to update it ➔ they each want to save revision = 6
- The first execution to commit its database transaction succeeds ➔ the parent revision goes from 5 ➔ 6.
- The second one tries to commit, but the database sees that the revision is already 6 (not 5 anymore). That means: **OptimisticLockingException is thrown**.

Exclusive Jobs

An exclusive job cannot be performed at the same time as another exclusive job from **the same process instance.**

It is actually not a performance issue:
With exclusive jobs the engine will simply distribute the load differently. Exclusive jobs means that jobs **from a single process instance** are performed by **the same thread sequentially**. But consider: you have more than one single process instance. **Jobs from other process instances are delegated to other threads and executed concurrently.** This means that with exclusive jobs the engine will not execute jobs from the same process instance concurrently but it will still execute multiple instances concurrently.

https://docs.camunda.org/manual/latest/user-guide/process-engine/the-job-executor/#exclusive-jobs

Backoff Strategy

In case there are less jobs in **ACT_RU_JOB table** than (**maxJobsPerAcquisition**) , the job acquisition waits for (**waitTimeInMillis**) before submitting jobs to the queue. If this keeps happening over and over again, the wait time will increase rapidly by multiplying it with the (**waitIncreaseFactor**), but it won't exceed the maximum waiting time (**maxWait**)

Guidelines (Parallel Execution)

Configure a save point

- **before** a **parallel join**
- and **after** every single instance of a **multi-instance activity**

To ensure that the path synchronisation will be taken care of by Camunda's internal job executor.

https://docs.camunda.io/docs/components/best-practices/development/understanding-transaction-handling-c7/#knowing-typical-dos-and-donts-for-save-points