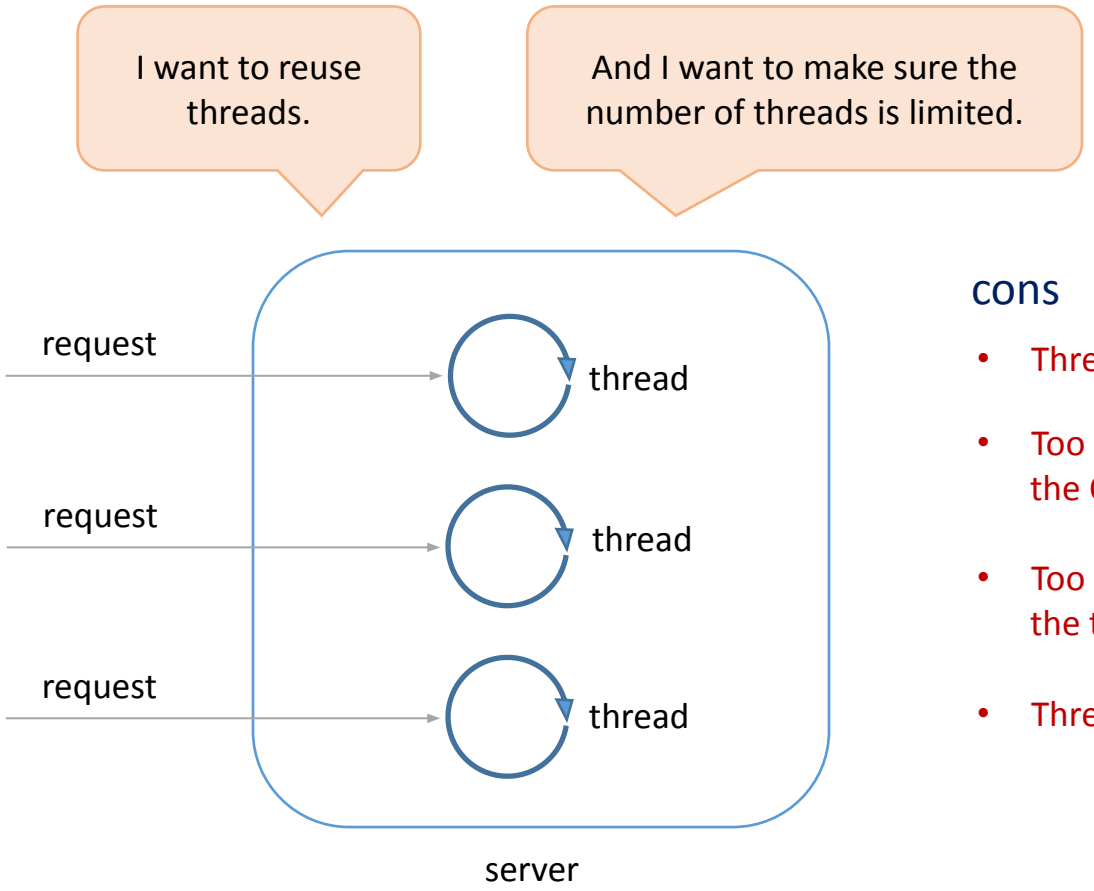
Thread pool



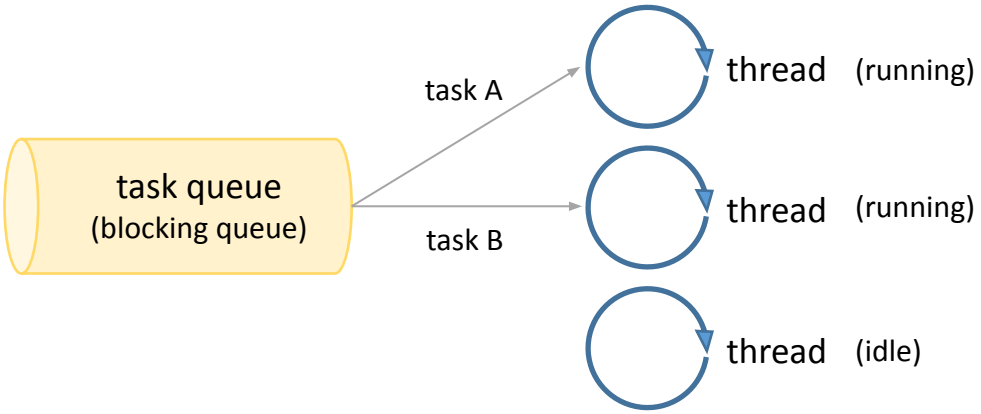
pros

- Threads increase throughput.

cons

- Thread creation consumes resources.
- Too many concurrent requests may lead to the OutOfMemory problem.
- Too many concurrent requests may lead to the thread starvation problem.
- Thread creation takes time.

Thread pool



pros

- Increase performance (decrease latency).
- Make applications more stable and predictable.
- Simplify coding (engineers think in tasks, not threads).

cons

- Sizing the thread pool can be difficult.
- Long-running tasks can clog the thread pool. (mitigated by timeouts)

Thread pool

How to size a thread pool?

CPU-bound task

compute-intensive

I/O-bound task

large number of input/output operations

in theory

size = number of CPU cores + 1

size = number of CPU cores * (1 + wait time / service time)

wait time - time spent waiting for IO operations to complete (CPU idle time)

service time - CPU busy time

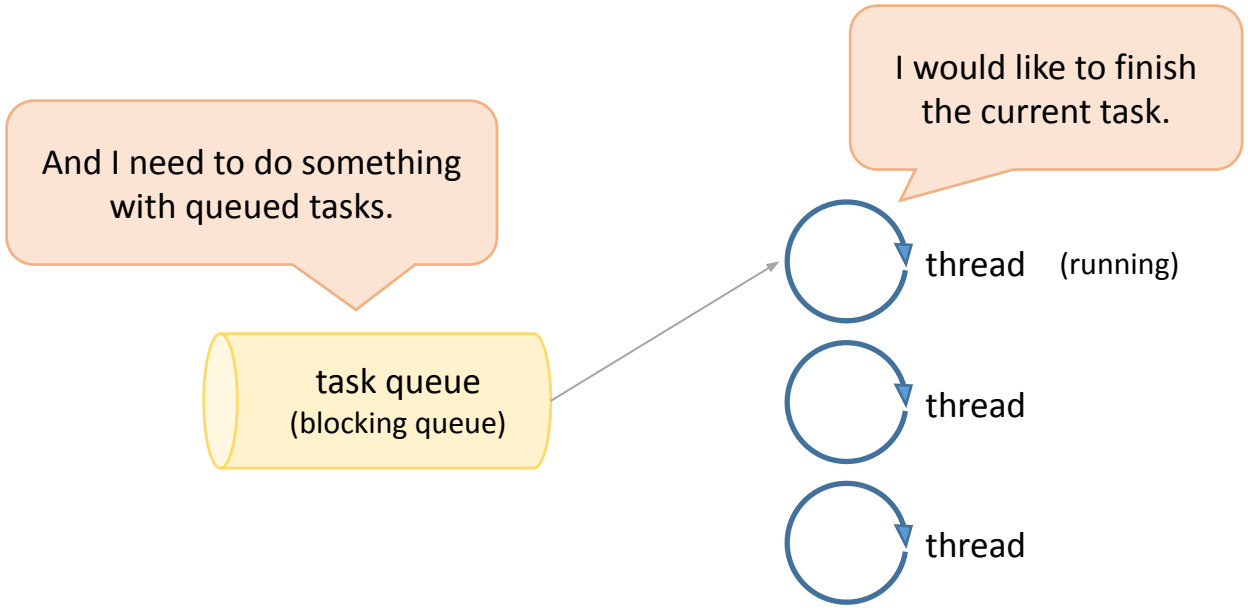
in practice

run a load test and and observe the level of CPU utilization

run a load test and and observe the level of CPU utilization

Thread pool

graceful shutdown



typical shutdown algorithm

1. Thread pool stops accepting any new tasks.

2. Thread pool waits for the previously submitted tasks to execute (e.g. several seconds).

3. Remaining tasks are cancelled (be careful here as we need to make sure no tasks are lost).

4. Thread pool terminates.