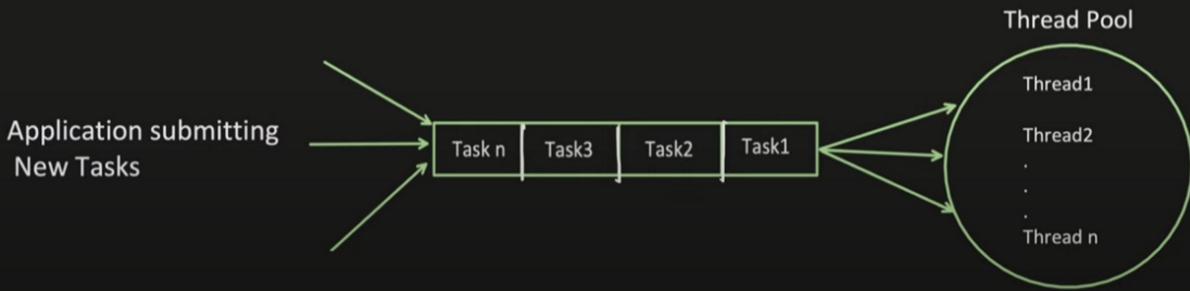


Lesson 6 : ThreadPool and ThreadPoolExecutor

Sunday, May 5, 2024 10:42 AM

What is ThreadPool:

- It's a collection of threads (aka workers), which are available to perform the submitted tasks.
- Once task completed, worker thread get back to Thread Pool and wait for new task to assigned.
- Means threads can be reused.



Tasks are added to the queue and picked up by the threads as they get free.

What's the Advantage of Thread Pool?

Thread Creation time can be saved:

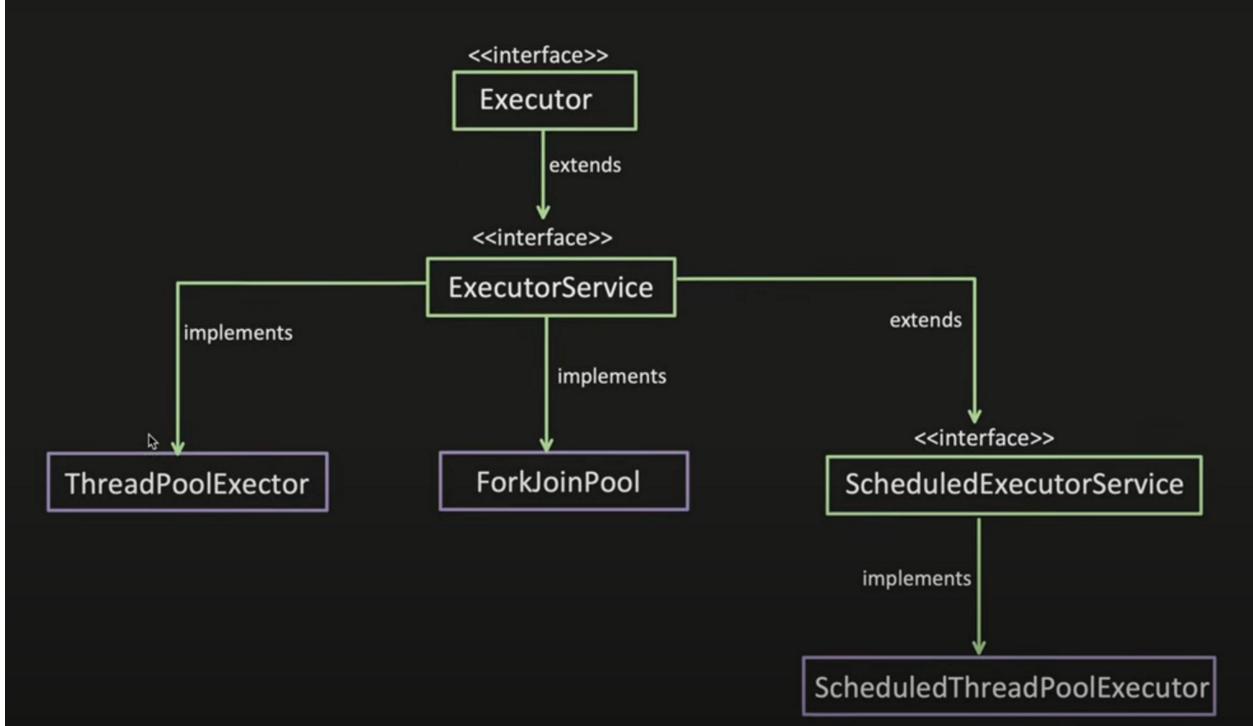
- When each thread created, space is allocated to it (stack, heap, program counter etc..) and this takes time.
↳ *new Thread*
- With thread, this can be avoided by reusing the thread.

Overhead of managing the Thread lifecycle can be removed:

- Thread has different state like Running, Waiting, terminate etc. And managing thread state includes complexity.
- Thread pool abstract away this management.

Increased the performance:

- More threads means, more Context Switching time, using control over thread creation, excess context switching can be avoided.



ThreadPoolExecutor

It helps to create a customizable ThreadPool.

ThreadPool is the collection of threads that are ready to pick up tasks from queue (diag above). ThreadPoolExecutor accepts params like `corePoolSize`, `maxPoolSize`. So if all the threads in the thread pool are busy and even the queue is full then thread pool assigns one more thread upto `maxPoolSize` limit and assigns the new task to it. If both `threadPool` and `Queue` have reached max limit then unfortunately task will have to be rejected.

ThreadPoolExecutor Constructor

```

public ThreadPoolExecutor(
    int corePoolSize,
    int maximumPoolSize,
    long keepAliveTime,
    TimeUnit unit,
    BlockingQueue<Runnable> workQueue,
    ThreadFactory threadFactory,
    RejectedExecutionHandler handler)
  
```

- **corePoolSize**:

Number of threads are initially created and keep in the pool, even if they are idle.

- **allowCoreThreadTimeOut**:

If this property is set to TRUE (by default its FALSE), idle thread kept Alive till time specified by 'KeepAliveTime'.

As carPoolSize has threads even if they are idle, so if we want to terminate those threads to save memory, we can set allowCoreThreadTimeOut as true, by default it is false. keepAliveTime is set that terminates the thread if it is idle for that time.

- **maxPoolSize**:

Maximum number of thread allowed in a pool.

If no. of thread are == corePoolSize and queue is also full, then new threads are created (till its less than 'maxPoolSize').

Excess thread, will remain in pool, this pool is not shutdown or if allowCoreThreadTimeOut set to true, then excess thread get terminated after remain idle for KeepAliveTime.

- **TimeUnit**:

TimeUnit for the keepAliveTime, whether Millisecond or Second or Hours etc.

- **BlockingQueue**:

Queue used to hold task, before they got picked by the worker thread.

Bounded Queue: Queue with FIXED capacity.

Like: *ArrayBlockingQueue*

Unbounded Queue: Queue with NO FIXED capacity.

Like: *LinkedBlockingQueue*

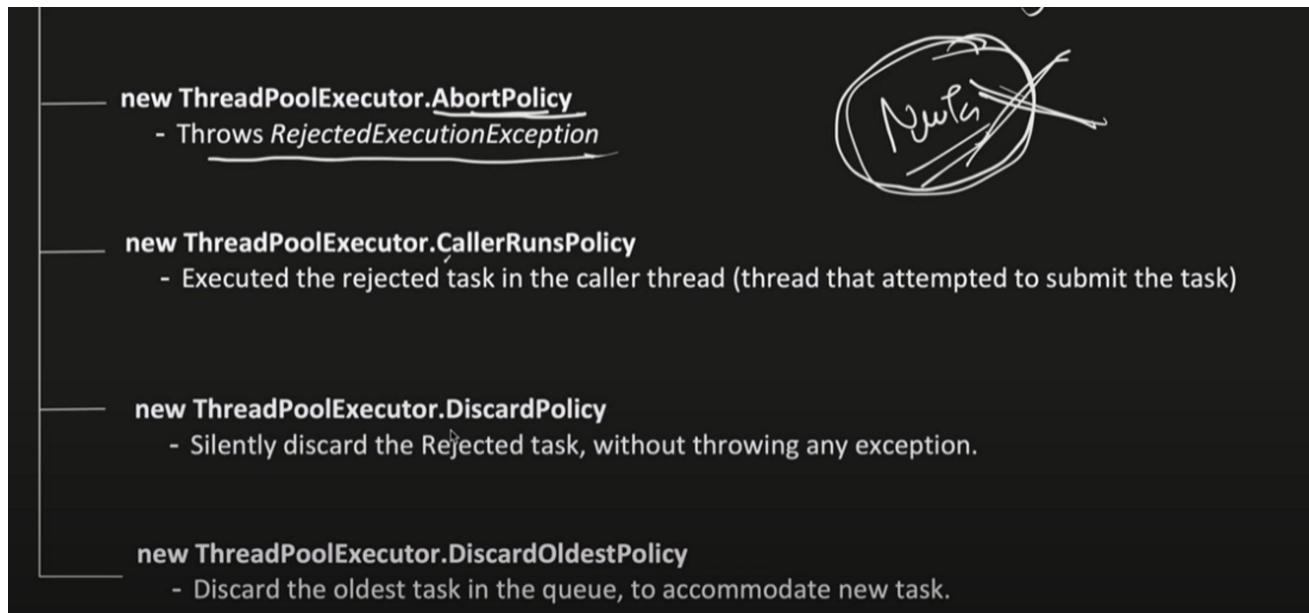
- ThreadFactory:

Factory for creating new thread. ThreadPoolExecutor use this to create new thread, this Factory provide us an interface to:

- To give custom Thread name
- To give custom Thread priority
- To set Thread Daemon flag etc.

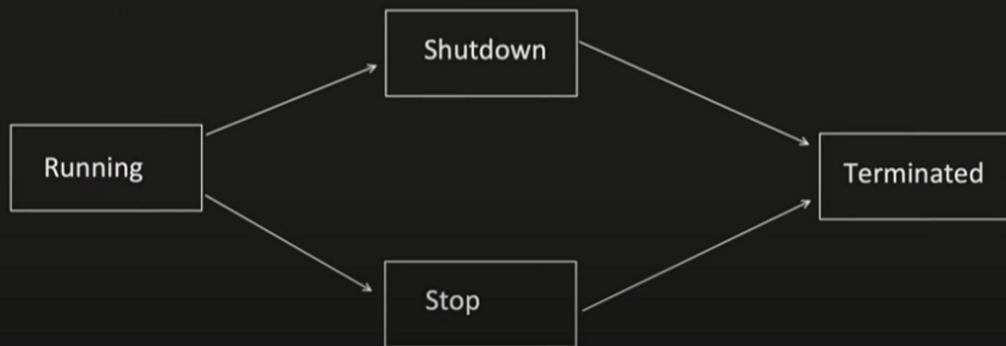
Whenever all the threads in threadPool are busy and even the queue is full, the upcoming tasks will fail.

RejectedExecutionHandler is the handler for tasks that can not be accepted by thread pool. Generally logging logic can be put here for debugging purpose.



Lifecycle of a ThreadPoolExecutor

Lifecycle of ThreadPool Executor



Running:

Executor is in running state and submit() method will be used to add new task.

Shutdown:

- Executor do not accept new tasks, but continue to process existing tasks, once existing tasks finished, executor moves to terminate state.
- Method used shutdown()

Stop (force shutdown):

- Executor do not accept new tasks.
- Executor forcefully stops all the tasks which are currently executing.
- And once fully shutdown, moves to terminate state.
- Method used shutdownNow()

Terminated:

- End of life for particular ThreadPoolExecutor.
- isTerminated() method can be used to check if particular thread pool executor is terminated or not.

Stop (force shutdown):

- Executor do not accept new tasks.
- Executor forcefully stops all the tasks which are currently executing.
- And once fully shutdown, moves to terminate state.
- Method used shutdownNow()

Terminated:

- End of life for particular ThreadPoolExecutor.
- isTerminated() method can be used to check if particular thread pool executor is terminated or not.

Formula to find the no. of thread:

$$\text{Max No of thread} = \text{No. of CPU Core} * (1 + \frac{\text{Request waiting time}}{\text{processing time}})$$

But this formula, do not consider Memory yet, which need to be consider...

{
JVM : 2 GB
 (Heap space : 1GB ✓
 ✓Code Cache space: 128MB ✓
 per Thread space: 5MB * N no. of threads (includes Thread Stack space)
 JVM Overhead: 256MB
)
}