Overview of the Java Executor Framework

(Part 1)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
• Understand the purpose of the Java executor framework

Decouples thread creation & management from the rest of the app logic
Learning Objectives in this Part of the Lesson

- Understand the purpose of the Java executor framework
- Know the types of thread pools supported by the framework
Learning Objectives in this Part of the Lesson

• Understand the purpose of the Java executor framework
• Know the types of thread pools supported by the framework
• Recognize a human known use of thread pools
Overview of the Java Executor Framework
Overview of The Java Executor Framework

- Java’s executor framework provides many classes & interfaces

Decouples thread creation & management from the rest of the app logic
The Executors utility class provides access to key mechanisms in the Java executor framework.

Overview of The Java Executor Framework

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html
The Executors utility class provides access to key mechanisms in the Java executor framework.

A utility class is a final class having only static methods, no state, & a private constructor.

Overview of The Java Executor Framework

<table>
<thead>
<tr>
<th>Java Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executors</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>newFixedThreadPool(int): ExecutorService</td>
</tr>
<tr>
<td>newWorkStealingPool(int): ExecutorService</td>
</tr>
<tr>
<td>newWorkStealingPool(): ExecutorService</td>
</tr>
<tr>
<td>newFixedThreadPool(int, ThreadFactory): ExecutorService</td>
</tr>
<tr>
<td>newSingleThreadExecutor(): ExecutorService</td>
</tr>
<tr>
<td>newSingleThreadExecutor(ThreadFactory): ExecutorService</td>
</tr>
<tr>
<td>newCachedThreadPool(): ExecutorService</td>
</tr>
<tr>
<td>newCachedThreadPool(ThreadFactory): ExecutorService</td>
</tr>
<tr>
<td>newSingleThreadScheduledExecutor(): ScheduledExecutorService</td>
</tr>
<tr>
<td>newSingleThreadScheduledExecutor(ThreadFactory): ScheduledExecutorService</td>
</tr>
<tr>
<td>newScheduledThreadPool(int): ScheduledExecutorService</td>
</tr>
<tr>
<td>newScheduledThreadPool(int, ThreadFactory): ScheduledExecutorService</td>
</tr>
<tr>
<td>defaultThreadFactory()</td>
</tr>
<tr>
<td>privilegedThreadFactory()</td>
</tr>
<tr>
<td>callable(Runnable, T): Callable&lt;T&gt;</td>
</tr>
<tr>
<td>callable(Runnable): Callable&lt;Object&gt;</td>
</tr>
<tr>
<td>callable(PrivilegedAction&lt;?&gt;): Callable&lt;Object&gt;</td>
</tr>
<tr>
<td>callable(PrivilegedExceptionAction&lt;?&gt;): Callable&lt;Object&gt;</td>
</tr>
<tr>
<td>privilegedCallable(Callable&lt;T&gt;): Callable&lt;T&gt;</td>
</tr>
<tr>
<td>privilegedCallableUsingCurrentClassLoader(Callable&lt;T&gt;): Callable&lt;T&gt;</td>
</tr>
</tbody>
</table>

Overview of The Java Executor Framework

• The Executors utility class provides access to key mechanisms in the Java executor framework
  • A utility class is a final class having only static methods, no state, & a private constructor
  • Its factory methods create various types of thread pools

See [en.wikipedia.org/wiki/Thread_pool_pattern](en.wikipedia.org/wiki/Thread_pool_pattern)
Overview of Thread Pools
Overview of Thread Pools

• Concurrent programs must often handle a large number of clients

*e.g., consider a web server that must handle thousands of client requests simultaneously*
Overview of Thread Pools

• However, spawning a thread per client doesn’t scale
Overview of Thread Pools

• However, spawning a thread per client doesn’t scale
• Dynamically spawning a thread per client incurs excessive processing overhead

```java
void handleClientRequest(Request request) {
    new Thread(makeRequestRunnable(request));
    ...
}
```
Overview of Thread Pools

- However, spawning a thread per client doesn’t scale
- Dynamically spawning a thread per client incurs excessive processing overhead
- An excessive amount of memory is also needed to store all the threads
Overview of Thread Pools

- A pool of threads is often a better way to scale concurrent app performance

See en.wikipedia.org/wiki/Thread_pool_pattern
Overview of Thread Pools

- A pool of threads is often a better way to scale concurrent app performance
- Amortizes memory/processing overhead associated with spawning threads

See cs.stackexchange.com/a/25899
Overview of Thread Pools

• A pool of threads is often a better way to scale concurrent app performance

• Amortizes memory/processing overhead associated with spawning threads

• Pool size determined by factors like # of cores, I/O-bound vs. compute-bound tasks, etc.

Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
  - **Fixed-size pool**
    - Reuses a fixed # of threads to amortize creation overhead

```java
mExecutor = Executors
    .newFixedThreadPool
    (sMAX_THREADS);
...

void handleClientRequest(Request request) {
    mExecutor.execute(makeRequestRunnable(request));
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newFixedThreadPool](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newFixedThreadPool)
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
- **Fixed-size pool**
  - Reuses a fixed # of threads to amortize creation overhead

\[
\text{mExecutor} = \text{Executors}.\text{newFixedThreadPool}(\text{sMAX\_THREADS});
\]

\[
\text{void \ handleClientRequest(Request request) \{}
   \text{mExecutor.execute(makeRequestRunnable(request))};
\]

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newFixedThreadPool
Overview of Thread Pools

• Java’s executor framework supports several types of thread pools
  • **Fixed-size pool**
    - Reuses a fixed # of threads to amortize creation overhead
    - Compute-bound tasks on an N-core CPU run best with a pool of ~N threads

Overview of Thread Pools

- Java’s executor framework supports several types of thread pools

- **Fixed-size pool**
  - Reuses a fixed # of threads to amortize creation overhead
  - Compute-bound tasks on an N-core CPU run best with a pool of ~N threads
  - I/O-bound tasks on an N-core CPU run best with N*(1+WT/ST) threads
    - WT = wait time & ST = service time

The goal is to keep the cores fully utilized
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
  - **Fixed-size pool**
    - Reuses a fixed # of threads to amortize creation overhead
    - Compute-bound tasks on an N-core CPU run best with a pool of ~N threads
    - I/O-bound tasks on an N-core CPU run best with \( N \times (1 + \frac{WT}{ST}) \) threads
      - \( WT = \) wait time & \( ST = \) service time
      - You can estimate the ratio for a typical request using profiling

See [www.baeldung.com/java-profilers](http://www.baeldung.com/java-profilers)
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools

- **Fixed-size pool**
  - Reuses a fixed # of threads to amortize creation overhead
  - Compute-bound tasks on an N-core CPU run best with a pool of ~N threads
  - I/O-bound tasks on an N-core CPU run best with N*(1+WT/ST) threads
  - Deadlock can be a problem with fixed-size thread pools that use bounded queues

Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
  - Fixed-size pool
  - Cached
    - Create new threads on-demand in response to client workload

```java
mExecutor = Executors.newCachedThreadPool();

... void handleClientRequest(Request request) {
    mExecutor.execute(makeRequestRunnable(request));
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool)
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
  - **Fixed-size pool**
  - **Cached**
    - Create new threads on-demand in response to client workload

```java
mExecutor = Executors.newCachedThreadPool();
...

void handleClientRequest(Request request) { 
    mExecutor.execute(makeRequestRunnable(request));
```

*Threads are terminated if not used for a certain time*
Overview of Thread Pools

• Java’s executor framework supports several types of thread pools
  • Fixed-size pool
  • Cached
    • Create new threads on-demand in response to client workload
  • There’s no need to estimate the size of the thread pool
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
  - *Fixed-size pool*
  - *Cached*
    - Create new threads on-demand in response to client workload
    - There’s no need to estimate the size of the thread pool
    - However, performance may suffer due to overhead of creating new threads
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
  - Fixed-size pool
  - Cached
  - Fork/join pool
    - Supports “work-stealing” queues that maximize core utilization

```java
mExecutor = Executors.newWorkStealingPool();
...

void handleClientRequest(Request request) {
  mExecutor.execute(makeRequestRunnable(request));
  ...
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newWorkStealingPool](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newWorkStealingPool)
Overview of Thread Pools

- Java’s executor framework supports several types of thread pools
  - Fixed-size pool
  - Cached
  - Fork/join pool
    - Supports “work-stealing” queues that maximize core utilization

```java
mExecutor = Executors.newWorkStealingPool();
...

void handleClientRequest(Request request) {
    mExecutor.execute(makeRequestRunnable(request)); ...
```

The pool size defaults to all available cores as its target parallelism level
Overview of Thread Pools

• Java’s executor framework supports several types of thread pools
  • Fixed-size pool
  • Cached
  • Fork/join pool
    • Supports “work-stealing” queues that maximize core utilization
    • Strike a balance between a fixed- & variable # of threads in the pool
Overview of Thread Pools

- There are also other ways to implement thread pools

---

See [www.dre.vanderbilt.edu/~schmidt/PDF/Lf.pdf](http://www.dre.vanderbilt.edu/~schmidt/PDF/Lf.pdf) & [www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf](http://www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf)
Human Known Uses of Thread Pools
A “call center” is a human known use of a thread pool. See en.wikipedia.org/wiki/Call_centre
End of Overview of the Java Executor Framework (Part 1)