The Java Executor Framework: Overview of Java Thread Pools

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
Learning Objectives in this Part of the Lesson

- Understand the purpose of the Java executor framework
- Recognize the benefits of using a thread pool
- Note a human known use of thread pools
- Know the Java Executor framework thread pools
Overview of Java Executor Framework Thread Pools
Overview of Java Executor Framework Thread Pools

- The executor framework supports several types of thread pools out-of-the-box.

See [docs.oracle.com/javase/tutorial/essential/concurrency/pools.html](docs.oracle.com/javase/tutorial/essential/concurrency/pools.html)
Overview of Java Executor Framework Thread Pools

• The executor framework supports several types of thread pools out-of-the-box

• Fixed-size pool
  • Reuses a fixed # of threads to amortize thread creation costs

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

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

See [en.wikipedia.org/wiki/Amortized_analysis](en.wikipedia.org/wiki/Amortized_analysis)
The executor framework supports several types of thread pools out-of-the-box.

**Fixed-size pool**

- Reuses a fixed # of threads to amortize thread creation costs.

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

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

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

- The executor framework supports several types of thread pools out-of-the-box
  - **Fixed-size pool**
  - Reuses a fixed # of threads to amortize thread creation costs

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

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

    Make & pass a runnable for execution by a thread in the pool
```

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 Java Executor Framework Thread Pools

• The executor framework supports several types of thread pools out-of-the-box
  • Fixed-size pool
    • Reuses a fixed # of threads to amortize thread creation costs

If a thread is somehow terminated while it is still in use, it is automatically replaced with a new thread

See docs.oracle.com/javase/tutorial/essential/concurrency/pools.html
Overview of Java Executor Framework Thread Pools

- The executor framework supports several types of thread pools out-of-the-box
- **Fixed-size pool**
  - Reuses a fixed # of threads to amortize thread creation costs
  - Compute-bound tasks on an N-core CPU run best w/an ~N thread pool

Overview of Java Executor Framework Thread Pools

- The executor framework supports several types of thread pools out-of-the-box

  - **Fixed-size pool**
    - Reuses a fixed # of threads to amortize thread creation costs
    - Compute-bound tasks on an N-core CPU run best w/an ~N thread pool
    - 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

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

- The executor framework supports several types of thread pools out-of-the-box

- **Fixed-size pool**
  - Reuses a fixed # of threads to amortize thread creation costs
  - Compute-bound tasks on an N-core CPU run best w/an ~N thread pool
  - 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 Java Executor Framework Thread Pools

• The executor framework supports several types of thread pools out-of-the-box

• **Fixed-size pool**
  • Reuses a fixed # of threads to amortize thread creation costs
  • Compute-bound tasks on an N-core CPU run best w/an ~N thread pool
  • 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

See asznajder.github.io/thread-pool-induced-deadlocks
Overview of Java Executor Framework Thread Pools

• The executor framework supports several types of thread pools out-of-the-box
  
  • Fixed-size pool
  
  • Cached
    • Create new threads on-demand in response to client workload

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

```java
void handleClientRequest(Request request) {
    mExecutor.execute(makeRequestRunnable(request));
}
```
Overview of Java Executor Framework Thread Pools

- The executor framework supports several types of thread pools out-of-the-box
  - **Fixed-size pool**
  - **Cached**
    - Create new threads on-demand in response to client workload

```
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

- Creates a new cached thread pool with 0 pre-allocated threads
The executor framework supports several types of thread pools out-of-the-box:

- **Fixed-size pool**
- **Cached**
  - Create new threads on-demand in response to client workload

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

```java
void handleClientRequest(Request request) {
    mExecutor.execute(makeRequestRunnable(request));
    // Make & pass a runnable for execution (will create or reuse a thread)
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool)
The executor framework supports several types of thread pools out-of-the-box:

- **Fixed-size pool**
- **Cached**
  - Create new threads on-demand in response to client workload

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

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

Threads are terminated if not used for a certain time.
Overview of Java Executor Framework Thread Pools

- The executor framework supports several types of thread pools out-of-the-box
  - *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 Java Executor Framework Thread Pools

- The executor framework supports several types of thread pools out-of-the-box
  - 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
The executor framework supports several types of thread pools out-of-the-box:

- **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)); ...
```
Overview of Java Executor Framework Thread Pools

- The executor framework supports several types of thread pools out-of-the-box
- **Fixed-size pool**
- **Cached**
- **Fork/join pool**
  - Supports “work-stealing” queues that maximize core utilization

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

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

Create a new pool whose size defaults to all available cores

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

- The executor framework supports several types of thread pools out-of-the-box
  - Fixed-size pool
  - Cached
  - Fork/join pool

- Supports "work-stealing" queues that maximize core utilization

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

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

Make & pass a runnable for execution in the pool (may be "stolen")

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

- The executor framework supports several types of thread pools out-of-the-box
  - 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

There are also other ways to implement thread pools.

1. **HALF-SYNC/HALF-ASYNC STRATEGY**
   - **Worker Threads**
     - 1: Select
     - 2: Dispatch
     - 3: Receive
     - 4: Enqueue
     - 5: Dequeue
     - 6: Process
   - **Message Queues**
   - **I/O Handler**
   - **Event Demuxer**

2. **LEADER/FOLLOWERS STRATEGY**
   - **Leader Thread**
     - 1: Select
     - 2: Dispatch
     - 3: Receive
     - 4: Promote Leader
     - 5: Process
   - **Follower Threads**
   - **Synchronizer**

See [www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf](http://www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf) & [www.dre.vanderbilt.edu/~schmidt/PDF/lf.pdf](http://www.dre.vanderbilt.edu/~schmidt/PDF/lf.pdf)
Other Types of Thread Pools

• There are also other ways to implement thread pools
• Moreover, you can integrate your own thread pool implementation into the Java Executor framework!

\[\text{e.g., you can extend/configure ThreadPoolExecutor, implement ExecutorService, etc.}\]
End of the Java Executor Framework: Overview of Java Thread Pools