Overview of the Java Executor Framework (Part 1)

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

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

• Understand how the Java executor framework decouples the creation & management of threads from the rest of the app logic.
Learning Objectives in this Part of the Lesson

• Understand how the Java executor framework decouples the creation & management of threads from the rest of the app logic

• Know the types of thread pools supported by the Java executor framework

Variable-sized Thread Pool

Fixed-sized Thread Pool

Work-stealing Thread Pool
Learning Objectives in this Part of the Lesson

• Understand how the Java executor framework decouples the creation & management of threads from the rest of the app logic
• Know the types of thread pools supported by the Java executor framework
• Recognize a human known use of thread pools
Overview of the Java Executor Framework
Overview of The Java Executor Framework

- The Java executor framework provides many classes & interfaces that decouple the creation & management of threads from application task logic.
Overview of The Java Executor Framework

- Access to the mechanisms defined by Java’s executor framework is mediated via the Executors class.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html
Overview of The Java Executor Framework

• Access to the mechanisms defined by Java’s executor framework is mediated via the Executors class

• This class defines a “façade” consisting of factory methods

Overview of The Java Executor Framework

- Access to the mechanisms defined by Java’s executor framework is mediated via the Executors class.
  - This class defines a “façade” consisting of factory methods.
  - These factory methods create thread pools.

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

• Concurrent programs must often handle a large # 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, e.g.
• Dynamically spawning a thread per client incurs excessive processing overhead

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

• However, spawning a thread per client doesn’t scale, e.g.
  • Dynamically spawning a thread per client incurs excessive processing overhead
  • It consumes an excessive amount of memory resources for all the threads

```java
void handleClientRequest(Request request) {
    new Thread(makeRequestRunnable(request));
    ...
```
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
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-intensive vs. compute-intensive tasks

See www.ibm.com/developerworks/library/j-jtp0730
Overview of Thread Pools

- Java’s executor service framework has several types of thread pools

A pool of worker threads
Overview of Thread Pools

- Java’s executor service framework has 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](docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newFixedThreadPool)
Overview of Thread Pools

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

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

void handleClientRequest(Request request) {
    mExecutor.execute(makeRequestRunnable(request));
    ...
```
Overview of Thread Pools

- Java’s executor service framework has 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](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool)
Overview of Thread Pools

- Java’s executor service framework has 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));
    ...
```
Overview of Thread Pools

• Java’s executor service framework has 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
Overview of Thread Pools

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

The pool size defaults to all available processor cores as its target parallelism level

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

void handleClientRequest(Request request) {
    mExecutor.execute(makeRequestRunnable(request));
    ...
```
Overview of Thread Pools

- There are also other ways of implementing 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
Human Known Uses of Thread Pools

• A human known use of a thread pool is a call center

See en.wikipedia.org/wiki/Call_centre
End of Overview of the Java Executor Framework (Part 1)