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

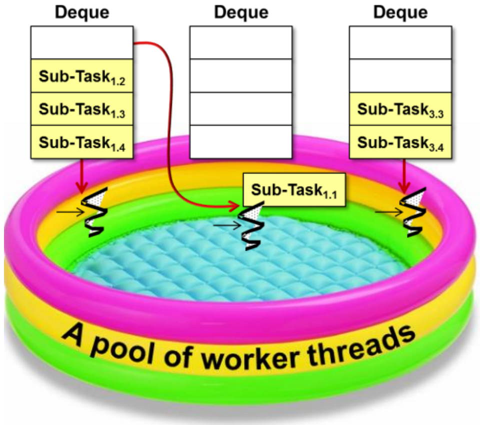
Cached (Variable-sized) Thread Pool

A pool of worker threads

Fixed-sized Thread Pool

A pool of worker threads

Work-stealing Thread Pool



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

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

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

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

See en.wikipedia.org/wiki/Amortized_analysis

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

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

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

Pre-allocate a pool of sMAX_THREADS

See 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

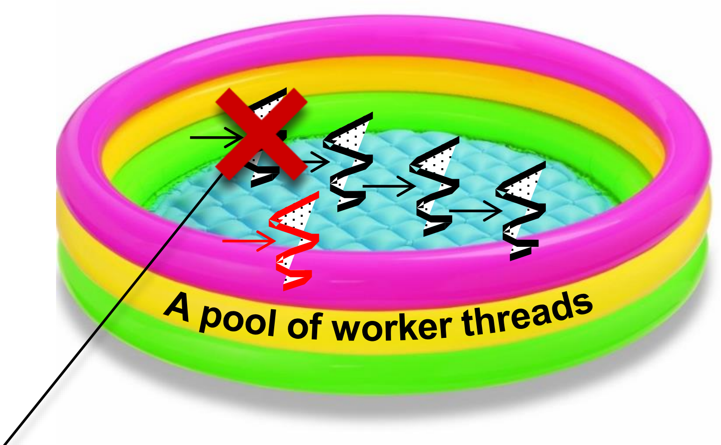
```
mExecutor = Executors
     .newFixedThreadPool
        (sMAX_THREADS);
...

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

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

See www.ibm.com/developerworks/library/j-jtp0730

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
      - WT = wait time & ST = service time

The goal is to keep the cores fully utilized

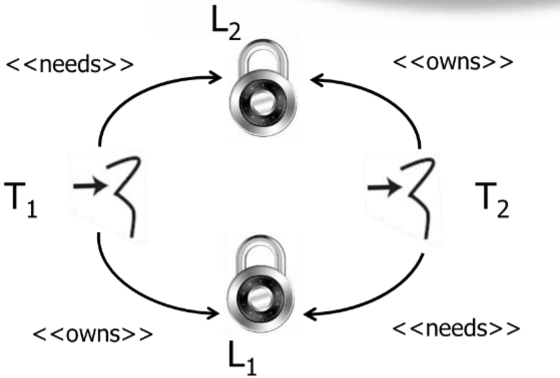
Overview of Java Executor Framework Thread Pools

- 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*(1+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

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

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

```
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();
...
```

Creates a new cached thread pool with 0 pre-allocated threads

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

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool

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));
```

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

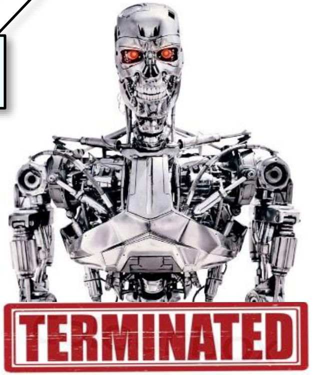
Overview of Java Executor Framework Thread Pools

- 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));
```

A pool of worker threads

Threads are terminated if not used for a certain time

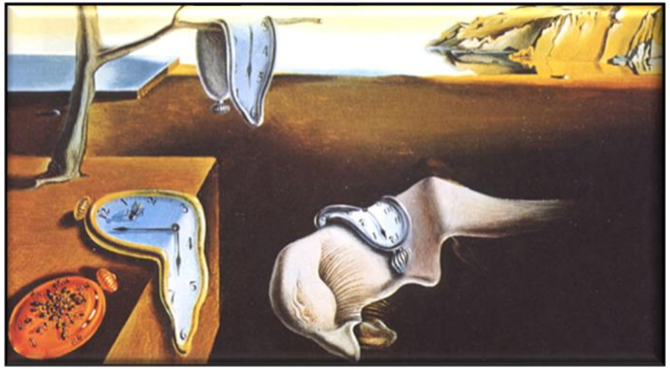


Overview of Java Executor Framework Thread Pools

- 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

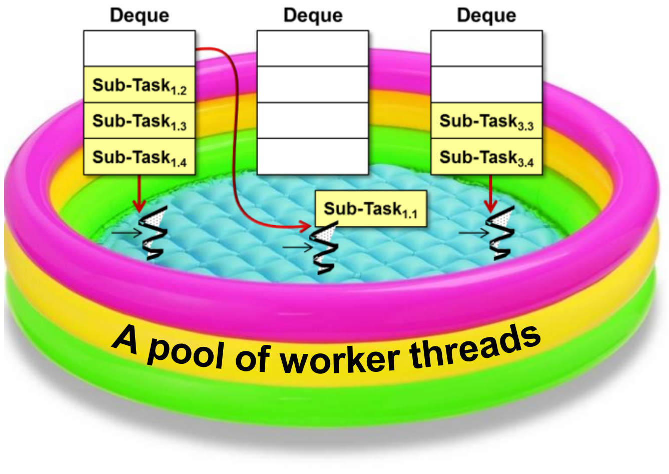
Overview of Java Executor Framework Thread Pools

- 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



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

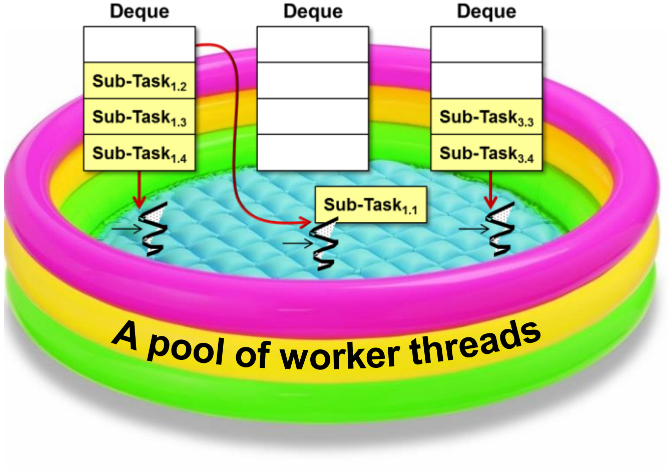


```
mExecutor = Executors
     .newWorkStealingPool();
...

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

See en.wikipedia.org/wiki/Work_stealing

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



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

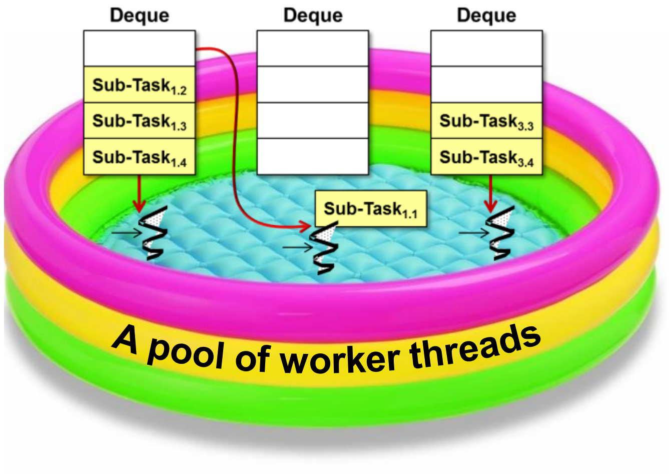
Create a new pool whose size defaults to all available cores

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

See 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



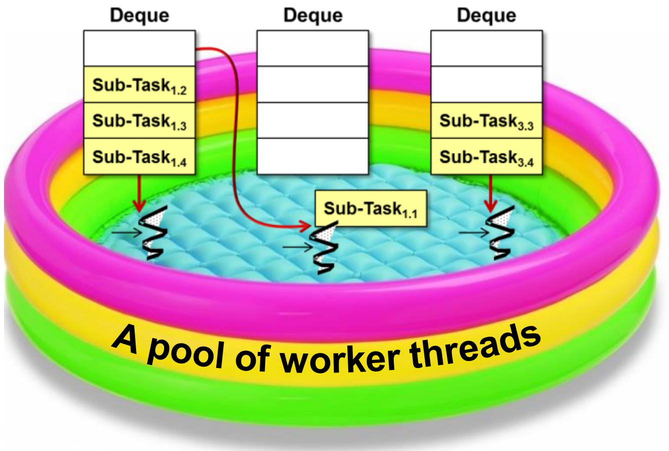
```
mExecutor = Executors
    .newWorkStealingPool();
...

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

Overview of Java Executor Framework Thread Pools

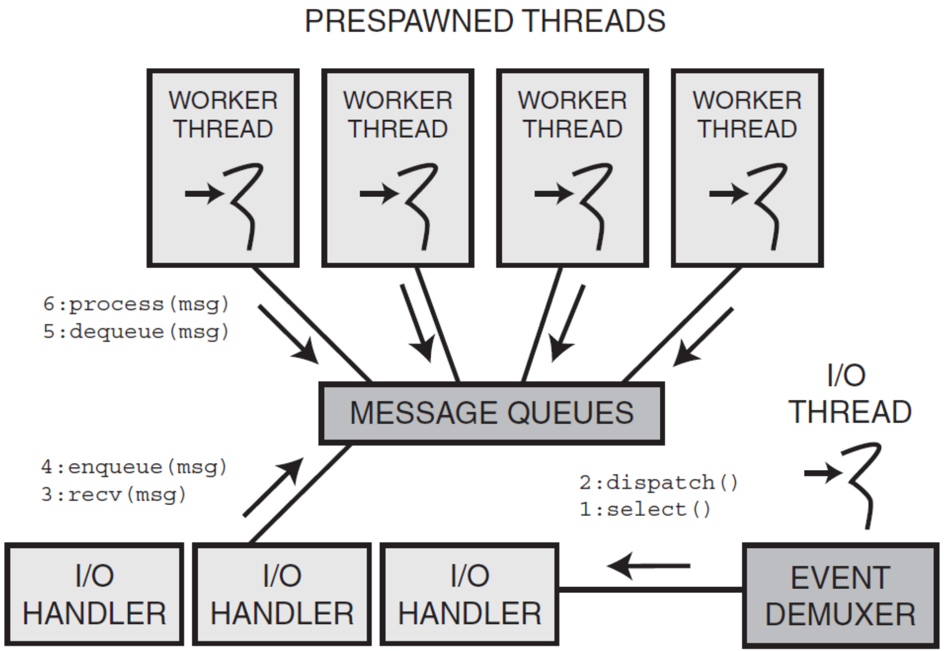
- The executor framework supports several types of thread pools out-of-the-box
  - *Fixed-size pool*
  - *Cached*
  - *Fork/join pool*
    - Supports "work-stealing" queues that maximize core utilization
    - Strike a balance between a fixed- & variable-# of threads in the pool



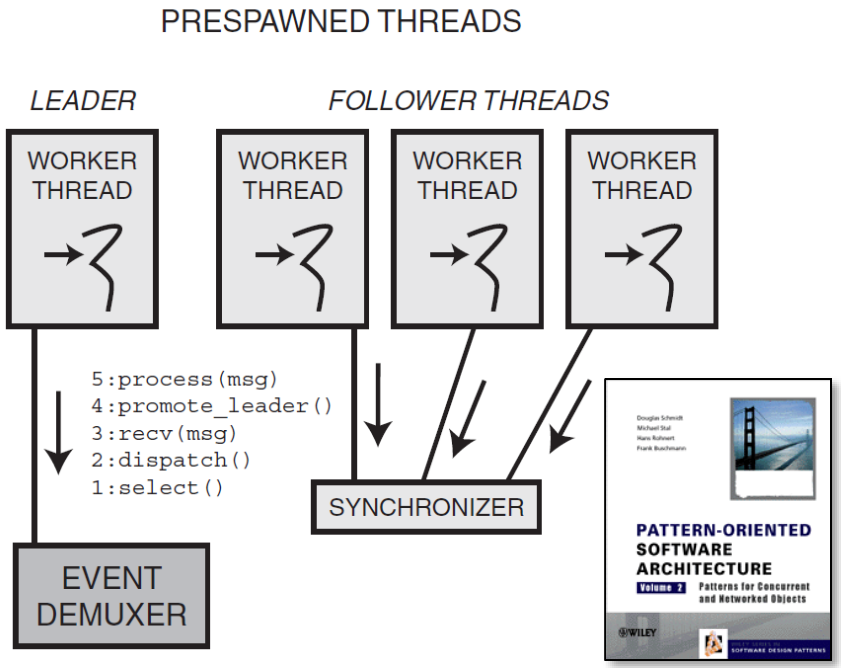
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.ManagedBlocker.html

Other Types of Thread Pools

- There are also other ways to implement thread pools



(1) HALF–SYNC/HALF–ASYNC STRATEGY



(2) LEADER/FOLLOWERS STRATEGY

See www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf & 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 you own thread pool implementation into the Java Executor framework!

e.g., you can extend/configure ThreadPoolExecutor, implement ExecutorService, etc.

End of the Java Executor Framework: Overview of Java Thread Pools