Overview of 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 features & benefits of thread pools
Learning Objectives in this Part of the Lesson

• Understand the purpose of the Java executor framework
• Recognize the features & benefits of thread pools
• Note a human known use of thread pools
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.
- It often incurs excessive processing overhead.

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

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

• However, spawning a thread per client doesn’t scale
  • It often incurs excessive processing overhead
  • An excessive amount of memory is also needed to store all the threads
• Even if it’s possible to spawn many threads, it usually means that “ya got trouble”..

See www.jstorimer.com/blogs/workingwithcode/7970125-how-many-threads-is-too-many
Overview of Thread Pools

- A thread pool is often a better way to scale performance

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

- A thread pool is often a better way to scale performance
- Amortizes thread memory/processing overhead

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

- A thread pool is often a better way to scale performance.
- Amortizes thread memory/processing overhead, e.g.
  ```java
  new Thread(makeRequestRunnable(request)).start();
  ```
  can often be replaced with a more efficient thread pool

```java
Executor executor = makeExecutor(...);
...
executor.execute(makeRequestRunnable(request));
```
Overview of Thread Pools

- A thread pool is often a better way to scale performance
- Amortizes thread memory/processing overhead
- Pool size determined by various factors
  - e.g., # of CPU cores, compute-bound vs. I/O-bound tasks, etc.

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

- A thread pool is often a better way to scale performance
- Amortizes thread memory/processing overhead
- Pool size determined by various factors
- A thread pool is tightly bound to a work queue of tasks awaiting execution
Overview of Thread Pools

- A thread pool is often a better way to scale performance
  - Amortizes thread memory/processing overhead
  - Pool size determined by various factors
  - A thread pool is tightly bound to a work queue of tasks awaiting execution
- Worker threads are like “hungry puppies”
Human Known Uses of Thread Pools
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 Thread Pools