CS601: Principles of Software Development

Pool of Threads. Work Queue.

Olga A. Karpenko

This presentation is based on the materials of Prof. Engle.

Motivation

- Web Server
  - Must handle multiple simultaneous requests
  - Must be responsive AND efficient

  (e.g. respond quickly, finish quickly)

- Implementation: Multithreading
  - Use one thread per request?

Problem

- Overhead cost to creating objects
- Overhead cost to destroying objects
- Threads are objects
  - Create one thread object per request?

Solution

- Keep threads around!
  - Create a “wise” number of threads
  - Only initialize thread objects once
  - Don’t destroy threads immediately
  - Reuse threads for other tasks
- Maintain
  - Thread pool
  - Work queue

Thread Pool

1. Create a fixed number of worker threads
2. When have work to do…
   - Get available thread from the pool
   - Assign thread a task
   - Run thread
   - Return thread to the pool

- What if out of threads?
  - Caller blocks until thread is available

ExecutorService

- Using built-in concurrent package in Java

```
ExecutorService executor =
Executors.newCachedThreadPool();
```

- Creates new threads as needed
- Reuses previously created threads
- Destroys a thread if not used for a period of time
- Other methods available such as: `newFixedThreadPool(int numThreads)`

ExecutorService

- Using the pool:

```
class Worker implements Runnable {
     @Override
     public void run() {
               // do some work
     }
}
// Somewhere else in the program:
executor.submit(new Worker());
```

Callable Interface. Future Object

- Callable interface has a call method
- Can return an value
- Can pass Callable to executorService
- The result can be obtained using Future
- See `SolvingForMaxUsingExecutor.java`

If we don't use concurrent package

Work Queue

- Add a work queue to thread pool
- Threads remove work from queue
  - Usually remove work in FIFO fashion
  - If no work, thread waits around
- Add work to queue
  - Add to queue whether or not thread is ready
  - Avoids blocking, more responsive
- Must use wait() and notify() methods…

IBM Work Queue

- Simple thread pool and work queue implementation
- Available at:

http://www.ibm.com/developerworks/library/j-jtp0730/

- Can use this for lab 3 part 2