Thread Pool Architecture

Daniel Zappala

CS 360 Internet Programming
Brigham Young University

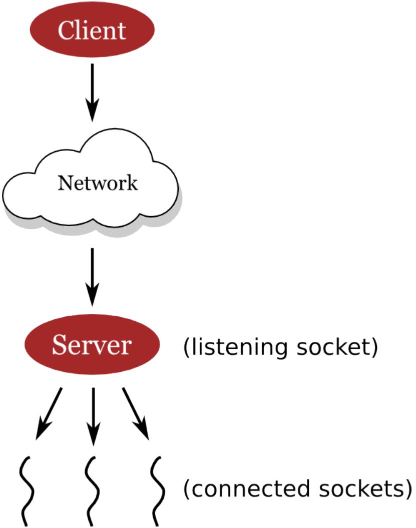
Goals for a Scalable Server

- handle many clients simultaneously
- provide fast response time to each client

How NOT to Build a Scalable Server

Do NOT Do This!

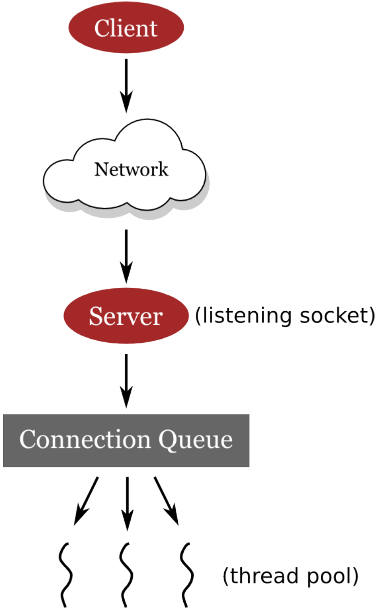
- create a new thread for each incoming connection
  - operating system overhead to create and switch among them
  - high CPU utilization from many simultaneous threads or processes
  - limit on the number of threads or processes you can create



Building a Scalable Server

Thread Pool

- server creates a fixed size thread pool
- listener thread (main program) enqueues incoming connections
- worker threads dequeue connections
    - handle one request and then return connection to queue, or
    - handle all requests until client done



Other Resource Pools

- one bottleneck to server performance is the operating system
  - system calls to allocate memory, access a file, or create a child process take significant amounts of time
  - as with many scaling problems in computer systems, caching is one solution
- resource pool: application-level data structure to allocate and cache resources
  - allocate and free memory in the application instead of using a system call
  - cache files, recent responses
  - limit critical functions to a small, well-tested part of code

Any Shared Resource Needs Synchronization