Spring Boot Microservices

Beginner to Guru

HTTP Clients

Communication Layers

- **HTTP - Application Layer** - ie how the client is communicating with the server
- **TCP - Transmission Control Protocol** - Transport Layer
  - How data is moved in packets between client and server
  - Server listens on a port (ie 80, 443), an ephemeral port is used on client to communicate back
  - Data is divided up into packets, transmitted, then re-assembled
- **IP - Internet Protocol** - Internet Layer
  - Specification of how packets are moved between hosts - just one packet

Java Input Output - IO

- Network communication in Java is done via java.io packages
- These are the low level libraries used to communicate with the host operating system
- TCP/IP connections are made via sockets
  - Light weight, but there is a cost to establish
- Early Java used one thread for each connection
  - Threads are much more costly
  - Modern OSs can support 100s of thousands of sockets, but only ~10,000 threads

Blocking and I/O

- Pre Java 1.4 threads would get blocked. One thread per connection.
  - Thread sleeps while IO completes
- Java 1.4 added non-blocking IO a.k.a. NIO - which allows for I/O without blocking the thread
  - Sets of sockets now can be used by a thread
- Java 1.7 added NIO.2 with asynchronous I/O
  - Networking tasks done completely in the background by the OS
- Non-Blocking is central to Reactive Programming

HTTP Client Performance

- Not uncommon for microservices to have many many client connections
- Non-blocking clients typically benchmark much higher than blocking clients
- Connection pooling can be used to avoid cost of thread creation and establishment of connections
  - Non-blocking and connection pooling can have a significant difference in the performance of your application
- As will all benchmarks - Your mileage may vary!!

Blocking Clients

- JDK - Java's implementation
- Apache HTTP Client
- Jersey
- OkHttp - may be changing version 4 under development

NIO Clients

- Apache Async Client
- Jersey Async HTTP Client
- Netty - Used by Reactive Spring

HTTP/2

- HTTP/2 is more performant than HTTP 1.1
- HTTP/2 uses the TCP layer much more efficiently
  - Multiplex streams
  - Binary Protocols / Compression
  - Reduced Latency
  - Faster Encryption
- To the REST API Developer, Functionally the Same
  - Both server and client need to support HTTP/2

HTTP/2 HTTP Clients

- Java 9+
- Jetty
- Netty
- OkHttp
- Vert.x
- Firefly
- Apache 5.x (In beta as of August 2019)

SPRING FRAMEWORK

GURU