Scalable Web Systems

Other Decompositional Techniques

Shared Libraries

A very standard decompositional technique that is built into virtually any language is breaking down a codebase into multiple libraries. These libraries may be provided by third parties, or created by a team or organization.

But, there are problems with this: you lose true technology heterogeneity. The library typically has to be in the same language, or at the very least run on the same platform.

Microservices can and should make heavy use of third-party libraries to reuse common code.

But libraries do not get us all the way there.

Modules

Some languages provide their own modular decomposition techniques that go beyond simple libraries. They allow some lifecycle management of the modules, such that they can be deployed into a running process, allowing you to make changes without taking the whole process down.

While modular decomposition within a process boundary may be something you want to do as well as decomposing your system into services, **by itself it won't help solve everything.**

Modules offer the same sorts of benefits as shared libraries.

So, microservices solve the scalability problem?

Like all approaches to building robust and scalable software, microservices are no free lunch or silver bullet, and make for a bad choice as a golden hammer.

Microservices have all the associated complexities of distributed systems.

From a monolithic system point of view, you'll have to get much better at handling **deployment**, **testing**, and **monitoring** to unlock the benefits.

You'll also need to think differently about how you **scale** your systems and ensure that they are **resilient**.

Summary

Main Challenge: microservices introduce a shift in the role of those who often guide the evolution of our systems: the architects.

So, as we progress through this course it will require you to think differently about how to design and implement web applications.

It will also require a more focused effort on the design of a web system with many more moving parts: load balancing, redundancy, databases, monitoring, and microservices.