微博软件架构分析

1. 分层架构(layered architecture)

在水平方向，微博采用了典型的三级分层模型，分为接口层、服务层与资源层。接口层主要实现与web页面、移动客户端的接口交互；服务层主要把核心业务模块化、服务化；而资源层主要是数据模型的存储。

优点：

1. 结构简单基础，容易理解和开发
2. 不同技术层面的工作人员可进行分工，负责不同的层，适合大多数软件公司的组织架构。
3. 每一层可独立测试，其他层的接口可通过模拟解决

缺点：

1. 软件升级时，整个服务可能都需要暂停。
2. 需要代码调整和增加功能时，通常比较麻烦和费时。
3. 微服务架构(microservices architecture)

微博将底层的东西分成了基础服务，基础服务里有分布式的存储，并做了一些去中心化、自动化的操作。

一款理想的分布式存储产品，首先它要支持海量规模、可扩展、高性能、低延迟、高可用。第二个是需要多机房分布，能够满足 国内负责的网络环境，还要具备异地容灾能力。第三个就是要调用简单，具备丰富数据库特性。因此分布式存储需要解决一个多对多的数据复制。

如其他大中型互联网应用一样，微博平台由众多的分布式组件构成，用户通过浏览器或移动客户端的每一个HTTP请求到达应用服务器后，会经过很多个业务系统或系统组件，并留下足迹（footprint）。但是这些分散的数据对于问题排查，或是流程优化都帮助有限。对于这样一种典型的跨进程/跨线程的场景，汇总收集并分析这类日志就显得尤为重要。另一方面，收集每一处足迹的性能数据，并根据策略对各子系统做流控或降级，也是确保微博平台高可用的重要因素。要能做到追踪每个请求的完整调用链路；收集调用链路上每个服务的性能数据；能追踪系统中所有的Error和Exception；通过计算性能数据和比对性能指标（SLA）再回馈到控制流程（control flow）中，基于这些目标就诞生了微博的Watchman系统。

该系统设计的一个核心原则就是低侵入性（non-invasivenss）：作为非业务组件，应当尽可能少侵入或者不侵入其他业务系统，保持对使用方的透明性，可以大大减少开发人员的负担和接入门槛。基于此考虑，所有的日志采集点都分布在技术框架中间件中，包括接口框架、RPC框架以及其他资源中间件。

**优点：**

1. 扩展性好，各个服务之间低耦合。
2. 容易部署，软件从单一可部署单元，被拆成了多个服务，每个服务都是单独的可部署单元。
3. 易于测试，可以单独测试每一个服务

**缺点：**

1. 由于强调互相独立和低耦合，服务可能会拆分得很细。这导致系统依赖大量的微服务，变得很凌乱和笨重，性能也会不佳。
2. 一些服务需要通信时（一个服务需用到另一个服务），整个架构会变得很复杂。
3. 微核架构（microkernel architecture）

微核架构又称为“插件架构”，指的是软件的内核相对较小，主要功能和业务逻辑都通过插件实现。

**优点：**

1. 良好的功能延展性，需要什么功能，开发一个插件即可
2. 功能之间是隔离的，插件可以独立的加载和卸载，容易部署
3. 可定制性搞，适应不同的开发需要
4. 可渐进式地开发，逐步增加功能