# 00 | 设计秒杀系统时应该注意的5个架构原则

**秒杀情景：**双十一购物、春节抢红包，12306抢火车票

**秒杀其实主要解决两个问题，一个是并发读，一个是并发写。**并发读的核心优化理念是：尽量减少用户到服务端来“读”数据，或者让他们读更少的数据；并发写的核心优化理念是：在数据库层面独立出来一个库，做特殊的处理。另外，我们还要针对秒杀系统做一些保护：针对意料之外的情况设计兜底方案，以防止最坏的情况发生。

**其实，秒杀的整体架构可以概括为“稳、准、快”几个关键字。**

* **稳**：就是整个系统架构要满足高可用，保证秒杀活动顺利完成，即秒杀商品顺利地卖出去，这个是最基本的前提。
* **准**：就是秒杀 10 台 iPhone，那就只能成交 10 台，多一台少一台都不行。所以“准”就是要求保证数据的一致性。
* **快**：它就是说系统的性能要足够高，否则你怎么支撑这么大的流量呢？

所以从技术角度上对应了我们架构上的高可用、一致性和高性能的要求：

* **高性能。**秒杀涉及大量的并发读和并发写，主要从设计数据的动态分离方案、热点的发现与隔离、请求的削峰与分层过滤、服务端的极致优化入手。
* **一致性。**
* **高可用。**极致的优化，但还要设计一个 PlanB 来兜底，以便在最坏情况发生时仍然能够从容应对。

# 01 | 设计秒杀系统时应该注意的5个架构原则

**架构原则：**4要1不要，即要保证用户请求的数据尽量少、请求数尽量少、路径尽量短、依赖尽量少，并且不要有单点。

**1. 数据要尽量少**

首先是指用户请求的数据能少就少。请求的数据包括上传给系统的数据和系统返回给用户的数据（通常就是网页）。其次，还要求系统以来的数据能少就少，包括系统完成某些业务逻辑需要读取和保存的数据，和数据库打交道䧂越好，数据越简单、越小则越好。

**2. 请求数要尽量少**

浏览器渲染这个页面包含其他的额外请求要尽可能少。

例如，减少请求数最常用的一个实践就是合并 CSS 和 JavaScript 文件，把多个 JavaScript 文件合并成一个文件，在 URL 中用逗号隔开，这种方式在服务端仍然是单个文件各自存放，只是服务端会有一个组件解析这个 URL，然后动态把这些文件合并起来一起返回。

**3. 路径要尽量短**

路径，就是用户发出请求到返回数据这个过程中，需求经过的中间的节点数。要缩短访问路径有一种方法，就是多个相互强依赖的应用合并部署在一起，把远程调用（RPC）变成JVM内部之间的方法调用。

**4. 依赖要尽量少**

依赖，指的是要完成一次用户请求必须依赖的系统或者服务，这里的依赖指的是强依赖。对于不是非要不可的信息，这些依赖在紧急情况下可以去掉。

**5. 不要有单点**

单点，指的是没有备份，风险不可控，我们设计分布式系统最重要的原则就是“消除单点”。

那如何避免单点呢？关键点是避免将服务的状态和机器绑定，即把服务无状态化。

**不同场景下的不同架构案例**