# 【第06章-编解码技术】【01-Java序列化的缺点】

基于J ava 提供的对象输入输出流ObjectInputStream 和ObjectOutputStream ， 可以直接把Java 对象作为可存储的字节数组写入文件，也可以传输到网络上。对程序员来说，基于JDK 默认的序列化机制可以避免操作底层的字节数组，从而提升开发效率。

J ava 序列化的目的它要有两个：

* 网络传输
* 对象持久化

## Java 序列化的缺点

Java 序列化从JDK 1.1版本就己经提供，它不需要添加额外的类库，只需实现java. io.Serializable 并生成序列ID 即可，因此，它从诞生之初就得到了广泛的应用。但是在远程服务调用（RPC）时，很少直接使用Java 序列化进行消息的编解码和传输，主要是因为Java序列缺陷导致。

### 无法跨语言

无法跨语言，是Java 序列化最致命的问题。对于跨进程的服务调用，服务提供者可能会使用C＋＋或者其他语言开发，当我们需要和异构语言进程交互时， Java 序列化就难以胜任。

由于J ava 序列化技术是Java 语言内部的私有协议，其他语言井不支持，对于用户来说它完全是黑盒。别于Java 序列化后的字节数组，别的语言无法进行反序列化，这就严重阻碍了它的应用。

事实上，目前几乎所有流行的Java RCP 通信框架，都没有使用Java 序列化作为编解码框架，原因就在于在无法跨语言． 而这些RPC 框架往往需要支持跨语言调用。

### 序列化后的码流太大

评判一个编解码框架的优劣时，往往会考虑以下儿个因素。

* 是否支持跨语言， 支持的语言种类是否丰富；
* 编码后的码流大小：
* 编解码的性能：
* 类库是否小巧， API 使用是否方便；
* 使用者需要手工开发的工作量和难度。

在同等情况下， 编码后的字节数组越大， 存储的时候就越占空间， 存储的硬件成本就越高，并且在网络传输时更占带宽，导致系统的吞吐量降低。J ava 序列化后的码流偏大也一直被业界所诟病，导致它的应用范围受到了很大限制

### 序列化性能太低

无论是序列化后的码流大小，还是序列化的性能， JDK 默认的序列化机制表现得都很差。因此，我们通常不会选择Java 序列化作为远程跨节点调用的编解码框架。