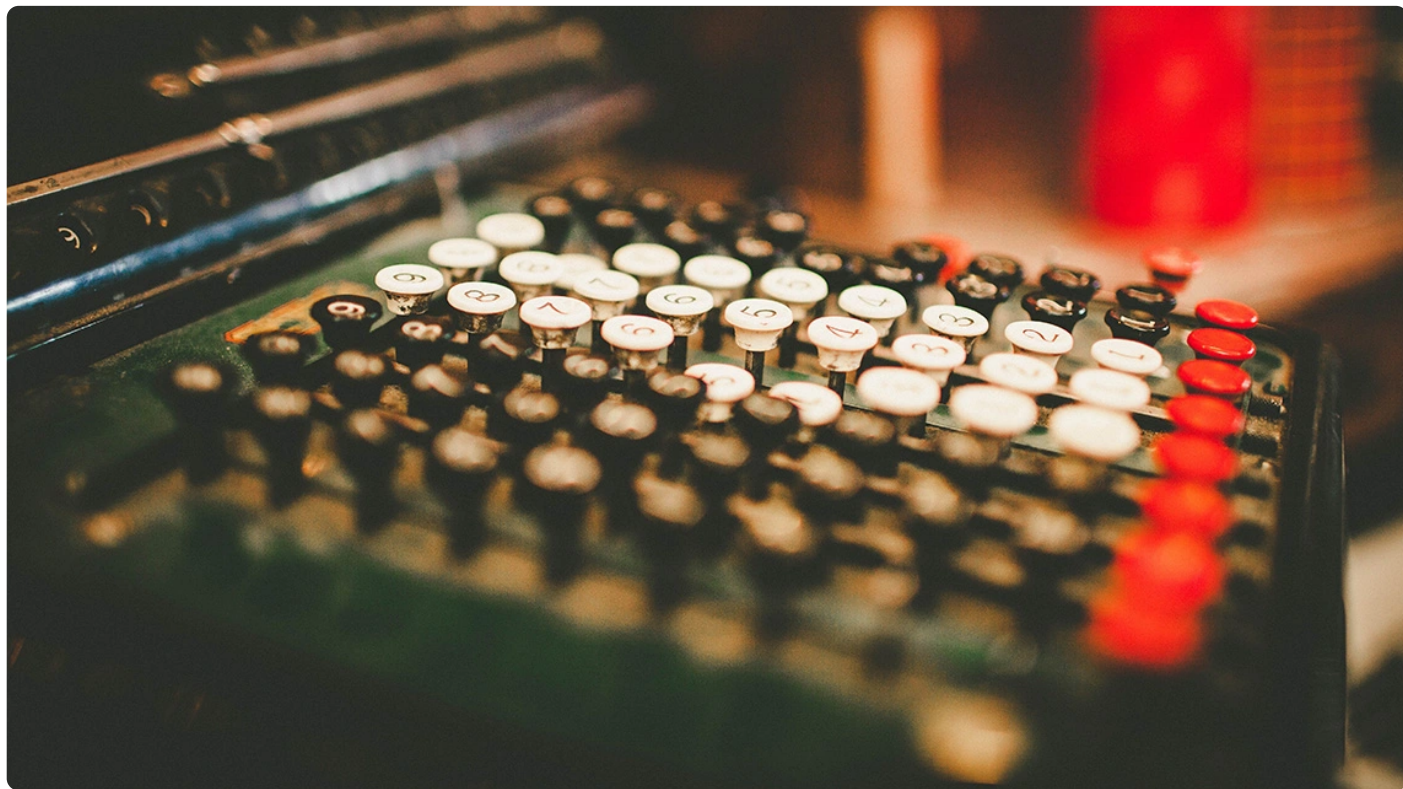


## 12 | 序列化与反序列化：如何通过网络传输结构化的数据？

李玥 · 消息队列高手课



你好，我是李玥。

最近有一些同学留言说，感觉这几节课讲的内容和消息关系不大。这里我解释一下，因为我们课程其中的一个目的，是让同学们不仅会使用消息队列，还可以通过对消息队列的学习，在代码能力上有一个提升，具备“造轮子”的能力。这样，你对消息队列的理解才能足够的深入，而不只是浮于表面。如果你细心可能也会发现，很多大厂在面试时，提到消息队列的问题，也不会仅仅局限在消息队列的使用上，他更多的会考察“你为什么这么实现”。

所以在进阶篇的上半部分，我会把开发一个消息队列需要用到的一些底层的关键技术给大家讲解清楚，然后我们再来一起分析消息队列的源代码。

在上节课中，我们解决了如何实现高性能的网络传输的问题。那是不是程序之间就可以通信了呢？这里面还有一些问题需要解决。

我们知道，在 TCP 的连接上，它传输数据的基本形式就是二进制流，也就是一段一段的 1 和 0。在一般编程语言或者网络框架提供的 API 中，传输数据的基本形式是字节，也就是 Byte。一个字节就是 8 个二进制位，8 个 Bit，所以在这里，二进制流和字节流本质上是一样的。

那对于我们编写的程序来说，它需要通过网络传输的数据是什么形式的呢？是结构化的数据，比如，一条命令、一段文本或者是一条消息。对应到我们写的代码中，这些结构化的数据是什么？这些都可以用一个类（Class）或者一个结构体（Struct）来表示。

那显然，**要想使用网络框架的 API 来传输结构化的数据，必须得先实现结构化的数据与字节流之间的双向转换**。这种将结构化数据转换成字节流的过程，我们称为序列化，反过来转换，就是反序列化。

序列化的用途除了用于在网络上传输数据以外，另外的一个重要用途是，将结构化数据保存在文件中，因为在文件内保存数据的形式也是二进制序列，和网络传输过程中的数据是一样的，所以序列化同样适用于将结构化数据保存在文件中。

很多处理海量数据的场景中，都需要将对象序列化后，把它们暂时从内存转移到磁盘中，等需要用的时候，再把数据从磁盘中读取出来，反序列化成对象来使用，这样不仅可以长期保存不丢失数据，而且可以节省有限的内存空间。

这节课，我们就来聊聊，怎么来实现高性能的序列化和反序列化。

## 你该选择哪种序列化实现？

如果说，只是实现序列化和反序列的功能，并不难，方法也有很多，比如我们最常使用的，把一个对象转换成字符串并打印出来，这其实就是一种序列化的实现，这个字符串只要转成字节序列，就可以在网络上传输或者保存在文件中了。

但是，你千万不要在你的程序中这么用，这种实现的方式仅仅只是能用而已，绝不是一个好的选择。

有很多通用的序列化实现，我们可以直接拿来使用。Java 和 Go 语言都内置了序列化实现，也有一些流行的开源序列化实现，比如，Google 的 Protobuf、Kryo、Hessian 等；此外，像 JSON、XML 这些标准的数据格式，也可以作为一种序列化实现来使用。

当然，我们也可以自己来实现私有的序列化实现。

面对这么多种序列化实现，我们该如何选择呢？你需要权衡这样几个因素：

1. 序列化后的数据最好是易于人类阅读的；
2. 实现的复杂度是否足够低；
3. 序列化和反序列化的速度越快越好；
4. 序列化后的信息密度越大越好，也就是说，同样的一个结构化数据，序列化之后占用的存储空间越小越好；

当然，**不会存在一种序列化实现在这四个方面都是最优的**，否则我们就没必要来纠结到底选择哪种实现了。因为，大多数情况下，易于阅读和信息密度是矛盾的，实现的复杂度和性能也是互相矛盾的。所以，我们需要根据所实现的业务，来选择合适的序列化实现。

像 JSON、XML 这些序列化方法，可读性最好，但信息密度也最低。像 Kryo、Hessian 这些通用的二进制序列化实现，适用范围广，使用简单，性能比 JSON、XML 要好一些，但是肯定不如专用的序列化实现。

对于一些强业务类系统，比如说电商类、社交类的应用系统，这些系统的特点是，业务复杂，需求变化快，但是对性能的要求没有那么苛刻。这种情况下，我推荐你使用 JSON 这种实现简单，数据可读性好的序列化实现，这种实现使用起来非常简单，序列化后的 JSON 数据我们都可以看得懂，无论是接口调试还是排查问题都非常方便。付出的代价就是多一点点 CPU 时间和存储空间而已。

比如我们要序列化一个 User 对象，它包含 3 个属性，姓名 zhangsan，年龄：23，婚姻状况：已婚。

[复制代码](#)

```
1 User:
2   name: "zhangsan"
3   age: 23
4   married: true
```

使用 JSON 序列化后：

[复制代码](#)

```
1 {"name":"zhangsan","age":"23","married":"true"}
```

这里面的数据我们不需要借助工具，是直接可以看懂的。

序列化的代码也比较简单，直接调用 JSON 序列化框架提供的方法就可以了：

[复制代码](#)

```
1 byte [] serializedUser = JsonConvert.SerializeObject(user).getBytes("UTF-8");
```

如果 JSON 序列化的性能达不到你系统的要求，可以采用性能更好的二进制序列化实现，实现的复杂度和 JSON 序列化是差不多的，都很简单，但是序列化性能更好，信息密度也更高，代价就是失去了可读性。

比如我们用 Kryo 来序列化 User 对象，它的代码如下：

[复制代码](#)

```
1 kryo.register(User.class);
2 Output output = new Output(new FileOutputStream("file.bin"));
3 kryo.writeObject(output, user);
```

在这段代码里，先要向 Kryo 注册一下 User 类，然后创建一个流，最后调用 writeObject 方法，将 user 对象序列化后直接写到流中。这个过程也是非常简单的。


## 实现高性能的序列化和反序列化

绝大部分系统，使用上面这两类通用的序列化实现都可以满足需求，而像消息队列这种用于解决通信问题的中间件，它对性能要求非常高，通用的序列化实现达不到性能要求，所以，很多的消息队列都选择自己实现高性能的专用序列化和反序列化。

使用专用的序列化方法，可以提高序列化性能，并有效减小序列化后的字节长度。

在专用的序列化方法中，不必考虑通用性。比如，我们可以固定字段的顺序，这样在序列化后的字节里面就不必包含字段名，只要字段值就可以了，不同类型的数据也可以做针对性的优化：

对于同样的 User 对象，我们可以把它序列化成这样：

 复制代码

```
1 03 | 08 7a 68 61 6e 67 73 61 6e | 17 | 01
2 User | z h a n g s a n | 23 | true
```

我解释一下，这个序列化方法是怎么表示 User 对象的。

首先我们需要标识一下这个对象的类型，这里面我们用一个字节来表示类型，比如用 03 表示这是一个 User 类型的对象。

我们约定，按照 name、age、married 这个固定顺序来序列化这三个属性。按照顺序，第一个字段是 name，我们不存字段名，直接存字段值“zhangsan”就可以了，由于名字的长度不固定，我们用第一个字节 08 表示这个名字的长度是 8 个字节，后面的 8 个字节就是 zhangsan。

第二个字段是年龄，我们直接用一个字节表示就可以了，23 的 16 进制是 17。

最后一个字段是婚姻状态，我们用一个字节来表示，01 表示已婚，00 表示未婚，这里面保存一个 01。

可以看到，同样的一个 User 对象，JSON 序列化后需要 47 个字节，这里只要 12 个字节就够了。

专用的序列化方法显然更高效，序列化出来的字节更少，在网络传输过程中的速度也更快。但缺点是，需要为每种对象类型定义专门的序列化和反序列化方法，实现起来太复杂了，大部分情况下是不划算的。

## 小结

进程之间要通过网络传输结构化的数据，需要通过序列化和反序列化来实现结构化数据和二进制数据的双向转换。在选择序列化实现的时候，需要综合考虑数据可读性，实现复杂度，性能和信息密度这四个因素。

大多数情况下，选择一个高性能的通用序列化框架都可以满足要求，在性能可以满足需求的前提下，推荐优先选择 JSON 这种可读性好的序列化方法。

如果说我们需要超高的性能，或者是带宽有限的情况下，可以使用专用的序列化方法，来提升序列化性能，节省传输流量。不过实现起来很复杂，大部分情况下并不划算。

## 思考题

课后，你可以想一下这个问题：在内存里存放的任何数据，它最基础的存储单元也是二进制比特，也就是说，我们应用程序操作的对象，它在内存中也是使用二进制存储的，既然都是二进制，为什么不能直接把内存中，对象对应的二进制数据直接通过网络发送出去，或者保存在文件中呢？为什么还需要序列化和反序列化呢？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。



小二

2019-08-20

尝试回答一下问题：

内存里存的东西，不通用，不同系统，不同语言的组织可能都是不一样的，而且还存在很多引用，指针，并不是直接数据块。

序列化，反序列化，其实是约定一种标准吧，大家都按这个标准去弄，就能跨平台，跨语言。

作者回复：这个总结好，简单明确，赞！



👍 234



游弋云端

2019-08-17

需要面临的问题：

- 1、网络字节序与主机字节序问题，业务要感知和处理大小端问题；
- 2、平台差异，各平台对基本数据类型的长度定义不一致、结构体对齐策略不一致，无法实现平台兼容；
- 3、连续内存问题，一个对象可能引用，指向其他对象，指针就是一个地址，传输后在另外的设备上无效值；

如果解决这些问题了，也变相的实现了自己的序列化框架了。

共 1 条评论 >

👍 58



ly

2019-08-19

个人对今天的内容进行简单的描述：

1. 序列化：是一种规则，它定义了数据表达的规则；
2. 反序列化：依靠给定的规则，还原数据。
3. 今天的问题：

内存中的对象数据应该具有语言独特性，例如表达相同业务的User对象(id/name/age字段),Java和PHP在内存中的数据格式应该不一样的，如果直接用内存中的数据，可能会造成语言不通。通常两个服务之间没有严格要求语言必须一致，只要对序列化的数据格式进行了协商，任何2个语言直接都可以进行序列化传输、接收。

作者回复：👍👍👍





24



a、

2019-08-17

- 1.因为应用程序里的对象，除了属性和属性值以外，还有一些其他的信息，比如jdk编译的版本，类的全限定名，类继承的父类和实现的接口等信息。如果服务端是jdk1.8编译的对象，发给客户端，客户端用的是jdk1.7，肯定会报错。
- 2.这些其他的信息是多余的，传输中会增加网络负担



23



钱

2019-08-23

### 课后思考及问题

1: 课后，你可以想一下这个问题：在内存里存放的任何数据，它最基础的存储单元也是二进制比特，也就是说，我们应用程序操作的对象，它在内存中也是使用二进制存储的，既然都是二进制，为什么不能直接把内存中，对象对应的二进制数据直接通过网络发送出去，或者保存在文件中呢？为什么还需要序列化和反序列化呢？

非常赞，老师的这个问题太经典了，可以拓宽认知边界。

具有细节，我不清楚，不过大概知道原因，比如：

1-1: JAVA说01表示TRUE，PHP说01表示我是世界上最好的语言

1-2: Mac说01表示1，Windows说10表示1

1-3: 虽然0/1在计算机的世界里，可以组合表示万事万物，比如：文字、图片、音频、视频、数据、指令等等，但是不同的语言、操作系统、硬件体系并没有被一个唯一的皇帝统一，他们的标准和存储方式都是有差异性的。所以，字符集有N多种就是这样，同样一个字符0在不同王国有不同的含义。

2: 看了老师的讲解，感觉自己也能自定义一个序列化和反序列化的框架，可能性能、通用性待优化。我的问题是，如果我想自己实现一个这样的框架，该怎么思考才能实现的更好？Jd自研的是怎么思考的？或者换个问法，老师在自研时是怎么考虑和设计的，有什么坑没？性能如何？关键想知道怎么自研的？

作者回复: 近期会发一个加餐来解答你的问题。

共 2 条评论 >

8



许童童

2019-08-17



内存在每个平台的分布都是不一样的，一个对象不光有用户定义的属性，还包括平台定义的属性，如果不经过序列化就传输过去，一方面会浪费大量的带宽，另一方面还可能因为平台不同等问题导致不兼容，从而无法解析。



8



星愿

2019-10-21

Jmq用的啥序列化框架呢

作者回复: 和其它的MQ的实现是类似的，自身的命令用的专用序列化，消息本身如何序列化是由业务代码决定的。

共 3 条评论 >



7



oscarwin

2019-08-19

虽然都是二进制的数，但是序列化的二进制数据是通过一定的协议将数据字段进行拼接。第一个优势是：不同的语言都可以遵循这种协议进行解析，实现了跨语言。第二个优势是：这种数据可以直接持久化到磁盘，从磁盘读取后也可以通过这个协议解析出来。如果是内存中的数据不能直接存盘的，直接存盘后再读出来我们根本无法辨识这是个什么数据。

作者回复: 👍

共 2 条评论 >



6



梵高

2020-06-01

老师，您好。咨询一个问题，序列化与反序列化是如何实现跨语言的。例如，服务端是java实现的，请求端是php的

作者回复: 其实序列化和反序列化是无所谓“跨语言”问题的。

序列化之后的数据，就是一段二进制数据，只要知道序列化的规则，用任何语言都可以反序列化。

如果是跨语言来传递对象，需要注意的就是不同语言之间数据类型的对应问题，特别是数据类型的长度和高低位问题。



楚翔style

2019-08-30

序列化:把对象转成通用格式数据(byte json)

反序列化:通用格式转成服务端能认的对象,比如json->Java Object

共 1 条评论 >



you-aholic

2020-07-21

张三23岁就结婚了，而我25的单身狗还在学消息队列。😂😂😂

共 3 条评论 >



淡定的、王先森

2019-10-27

想问下老师，在项目开发中，什么时候需要显式的实现对象的序列化，或者用到序列化？感觉日常开发中并没有涉及，或者应用的框架给做了我并没有察觉到？大多数项目都是怎么设计序列化的呢？请老师提示或指教下，不胜感激

作者回复: 你在使用消息队列的时候，发消息之前就需要对消息进行序列化，收到消息再反序列化。

可能你日常发送的都是String类型的消息，没有意识到？

比如，你要通过消息发送一个有结构的对象到对端，就需要把这个对象序列化。



Switch

2019-10-12

在内存中，存放的对象也是有相应的结构的。如果直接保存，直接读取，那么也需要有相应的转换器，将数据转换为内存中的对象。而且这样的对象，不通用，换一种语言或者实现就不能通用了，并且这样的对象不一定是节省空间的。

我们损失一部分性能，换取跨语言性、节省存储空间也是一种两者取较好的选择。



南辕北辙

2019-08-21

个人理解，序列化与反序列化的通用思想和通信协议挺类似的，只要通信双方约好什么样的格式去定义数据，想怎么玩就怎么玩。



👍 3



**zhoufeng**

2022-07-23

问个小白问题，既然序列化是将结构化数据转换成字节流，那为什么json序列化后是字符串形式的

共 2 条评论 >

👍 1



**William Ning**

2022-02-23

每天都在学习，感觉有收获不断。



👍 1



**凯文小猪**

2021-11-22

二刷尝试回答下老师的问题：

以java为例，内存中对象存储格式是：对象头+this指针+payload+对齐数据

1.所以对象头、this指针、对齐数据对实际传输是多余的 当然通常他们也就不过几十到几百个字节不等。但是设计一个协议 脏数据总是要考虑移除的。

2.内存中数据是按照class文件定义存放的 比方说我定义了 `class User{ String name; int age}`

那么 第一个出现的必然是name 第二个出现的就是age 。但是直接将该数据传送出去 其他语言框架未必就是按照这个格式来读取的 所以就不具备通用性

3.现代语言通常为了节省内存 对于确定性常量都会构造常量池 这部分拆解可想而知非常麻烦。

4.反序列化原生java内存对象 还需要将class定义也一起传送 故又多了一个成本

综上 序列化需要一个通常的协议 即使是常用的json也比直接用内存对象好



👍 1



**Trident**

2020-02-22

自己实现序列化反序列化，虽然可以节省一点存储成本或者网络传输带宽，但是牺牲了可读性，对于一般应用来说，没有必要为了性能的提升而降低数据的可阅读性



1

**业余爱好者**

2019-11-24

从语义上理解序列化与反序列化，序列化就是把内存中结构化的对象转化为跨平台的可以在网络上传输的二进制“流”。关键词是流，也就是说，它是一个线性的数据结构。而在内存中的数据是结构化的，非线性的，而且各个平台，各种语言的存在方式还不一样。

序列化就是把要传输的信息从专用的表示方式转换为通用的二进制流的表示方式。反序列化的过程刚好相反。

虽然都是二进制，不过信息的编码方式不同。



1

**小红帽**

2023-02-28 来自广东

为什么不能直接把内存中，对象对应的二进制数据直接通过网络发送出去，或者保存在文件中呢？为什么还需要序列化和反序列化呢？

首先这些落地的数据我们肯定要看的，可读性我们有要求；

其次，像java 虚拟机，他内部操作的也是字节，只不过我们代码书写的时候不可能针对字节来编写代码；

综上所述，我们的业务不需要如此高的性能，宁愿多付出一点序列化，反序列化的时间，也要提高代码和数据的可读性；

