新型序列化类库MessagePack，比JSON更快、更小的格式

作者：heyue 时间：2013-07-26 分类：[LAMP](http://www.heyues.com/category/lamp/" \o "查看 LAMP 中的全部文章) [Linux C开发](http://www.heyues.com/category/linux-c/) [互联网技术](http://www.heyues.com/category/%e4%ba%92%e8%81%94%e7%bd%91%e6%8a%80%e6%9c%af/) [未分类](http://www.heyues.com/category/%e6%9c%aa%e5%88%86%e7%b1%bb/) 评论：[4条评论](http://www.heyues.com/messagepack/" \l "comments" \o "《新型序列化类库MessagePack，比JSON更快、更小的格式》上的评论) 浏览：20240

MessagePack 是个什么东东？先来看一段官方的解释：

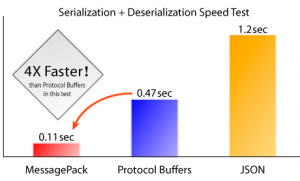
MessagePack is an efficient binary serialization format. It lets you exchange data among multiple languages like JSON. But it’s faster and smaller. Small integers are encoded into a single byte, and typical short strings require only one extra byte in addition to the strings themselves.

MessagePack  是一个高效的二进制序列化格式。它让你像JSON一样可以在各种语言之间交换数据。但是它比JSON更快、更小。小的整数会被编码成一个字节，短的字符串仅仅只需要比它的长度多一字节的大小。

官方用一句话总结了这个东东：

It’s like JSON.  
but fast and small.

最初研究MessagePack 大概是两年前了，还开了个讲座给大家讲MessagePack是个什么东西，大概用在什么场合，它是不是给Javascript用的之类的。但是两年过去了，由于博客平台老系统太多，以至于这个协议一直没有能推进使用。后来，redis宣布支持MessagePack格式，以及pintrest等公司，也在积极得使用这个协议进行开发，说明这个格式确实有很多先进性。

[](http://www.heyues.com/wp-content/uploads/2013/07/afb35d36-7aa3-3bc8-af7f-0d4360c096de.png)

MessagePack、protocol buffers、json的速度对比

这张图片是以前MessagePack 官方网站的首页图片，数字对比确实很能反映问题，笔者不是很了解protocol buffers，XML又太老土了，就跳过他们俩了，只讨论JSON和MessagePack了。

为啥会小？

先大概说下MessagePack 为啥会比JSON小吧，先来段json:

{“name“:”heyue“,”sex“:”\u7537“,”company“:”sina“,”age“:30}

这个json长度为57字节，但是为了表示这个数据结构（所有标红色的地方就是他为了表示这个数据结构而不得不添加的），它用了23个字节（就是那些大括号、引号、冒号之类的，他们是白白多出来的）。大家可以去<http://json.org/> 上看看json的数据标示定义。

换成MessagePack，我只能给大家贴代码和结果了，38字节：

[?](http://www.heyues.com/messagepack/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <?php  $arr = array('name'=>"heyue",'sex'=>'男','company'=>'sina','age'=>30);  echo "Json:".strlen(json\_encode($arr))."\n";  echo "Messagepack:".strlen(msgpack\_pack($arr))."\n";    $str = "何跃新浪";  echo json\_encode($str)."\n";  echo 'json\_str:'.strlen(json\_encode($str))."\n";  echo 'MessagePack\_str:'.strlen(msgpack\_pack($str))."\n";  $str = "sina china";  echo json\_encode($str)."\n";  echo 'json\_str:'.strlen(json\_encode($str))."\n";  echo 'MessagePack\_str:'.strlen(msgpack\_pack($str))."\n";    ?>    Json:57  Messagepack:38 //从这里可以看出MessagePack比json少了好多  "\u4f55\u8dc3\u65b0\u6d6a"  json\_str:26  MessagePack\_str:13 //在UTF-8多字节字符中，MessagePack采用原生态存储，4个汉字，只用了13字节，比原始的只多了1字节  "sina china"  json\_str:12  MessagePack\_str:11 //英文字符呢？这个仅仅是比json少了一个引号的大小。 |

我不能给大家算比例，因为这个得看MessagePack的压缩算法，MessagePack的核心压缩方式：

**1.true、false 之类的**：这些太简单了，直接给1个字节，（0xc2 表示true，0xc3表示false）

**2.不用表示长度的**：就是数字之类的，他们天然是定长的，是用一个字节表示后面的内容是什么东东，比如用（0xcc 表示这后面，是个uint 8，用oxcd表示后面是个uint 16，用 0xca 表示后面的是个float 32).

**3.不定长的：比如字符串、数组**，类型后面加 1~4个字节，用来存字符串的长度，如果是字符串长度是256以内的，只需要1个字节，MessagePack能存的最长的字符串，是(2^32 -1 ) 最长的4G的字符串大小。

**4.ext结构：**表示特定的小单元数据。

**5.高级结构：MAP结构**，就是key=>val 结构的数据，和数组差不多，加1~4个字节表示后面有多少个项。

这个是官方的数据表示结构文档：<https://gist.github.com/frsyuki/5432559>

总的来说，MessagePack对数字、多字节字符、数组等都做了很多优化，减少了无用的字符，二进制格式，也保证不用字符化带来额外的存储空间的增加，所以MessagePack比JSON小是肯定的，小多少，得看你的数据。如果你用来存英文字符串，那几乎是没有区别….

为啥会快？

先说说JSON怎么解析吧，我们开发中一般都用cJSON这个库，cJSON存储的时候是采用链表存储的，其访问方式很像一颗树。每一个节点可以有兄妹节点，通过next/prev指针来查找，它类似双向链表；每个节点也可以有孩子节点，通过child指针来访问，进入下一层。问题就是首先，构造这个链表的时候，得一个字符一个字符地匹配过去吧，得判断是不是引号、括号之类的吧…

但是MessagePack 则简单多了，直接一遍遍历过去了，从前面的数据头，就可以知道后面的是什么数据，指针应该向后移动多少，比JSON的构建链表少了很多比较的过程。

来计算个数据吧，把刚才的数组，encode、decode重复1000万次：

msgpack\_unpack(msgpack\_pack($arr));

json\_decode(json\_encode($arr));

**Json：37.099s**

**MessagePack：22.050s**

大概是快这么多吧，如果数组更大，理论上，MessagePack比Json快更多。

MessagePack的常用的地方：

MessagePack 不是给JS用的，虽然它有JS的库，但是用浏览器来解析MessagePack是一件很悲剧的事情，我曾经测试过（如果我还能找到，我会提供代码），在低端浏览器下，JS计算MessagePack会卡死在那里，毕竟JSON是javascript亲生的，用起来自然比MessagePack要容易。

MessagePack主要用于结构化数据的缓存和存储：

1.存在Memcache中，因为它比json小，可以省下一些内存来，速度也比json快一些，页面速度自然快一个档次。当然，也有一种情况，我在mc中存json，然后直接出来就是页面可用的json，都不用解析json了（当然这个在实际开发中比较少见）。

2.存在可以持久化的Key-val存储中。

MessagePack的现状：

我就说PHP吧，因为C、C++的没啥好说的，就是解包、打包，速度比JSON快一些，但是业务逻辑的数据太多，还是先考虑上层的吧。

PHP的MessagePack的扩展的安装：

[?](http://www.heyues.com/messagepack/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 可以用PECL的安装方式：  pecl install msgpack  也可以编译源码安装：  $/path/to/phpize  $./configure  $make && make install |

使用方法：

[?](http://www.heyues.com/messagepack/)

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?php  $data = array(0=>1,1=>2,2=>3);  $msg = msgpack\_pack($data);  $data = msgpack\_unpack($msg);  ?> |

这个MessagePack的PHP扩展，是传说中的[鸟哥Laruence](http://www.laruence.com/" \o "鸟哥的博客" \t "_blank)开发维护的，在鸟哥的[Yar](http://www.laruence.com/2012/09/15/2779.html" \o "鸟哥的Yar" \t "_blank)中，也使用了MessagePack 作为打包协议之一。

从现状看来，MessagePack目前还很少有公司大规模使用？这是为什么呢？由于没有读过MessagePack的相关的源码，所以在这个范畴，鸟哥最有发言权…

后来，redis 2.6支持了MessagePack…

先写到这里了，有空了，再补充一些，比如MessagePack 和 protocol buffer的异同之类的，洗洗睡了…

来源： <http://www.heyues.com/messagepack/>