射电望远镜控制系统中的数据传输序列化分析

李 军[1,2]，刘志勇[1,3]，王 娜[1,3]，宋祎宁[1,2]，杨 垒[1,2]，王吉利[1]

（1. 中国科学院新疆天文台，乌鲁木齐830011;

2. 中国科学院大学，北京100049;

3. 中国科学院射电天文重点研究室，南京 210008）

摘 要：控制系统能衔接、集成和管理射电望远镜的软硬件系统。控制系统中的序列化工具可将射电望远镜中不同设备、操作系统、编程语言和网络之间传输的信息进行编码和解码，增强系统之间数据的传输效率。本文分析和比较了三款二进制序列化工具Msgpack、Protobuf和Flatbuffers的编码原理及特性，并通过一个实例测试了它们的序列化数据大小、序列化时间和CPU利用率。结果表明，Msgpack的综合性能优于Protobuf和Flatbuffers，适用于周期长、需求易变的射电望远镜系统之间传输信息的编码和解码。

关键词：射电望远镜；二进制序列化工具；控制系统；编码；解码

中图分类号：TP311　　**文献标识码：**A　　**文章编号：**

0 引言1

射电望远镜是射电天文研究的基石，它由天线、接收机、终端、监测和控制等系统组成。它们中具有连接、集成和管理功能的控制系统为射电望远镜的重要组成部分[1]。其中，数据交换是控制系统的基本功能之一，在望远镜控制与多终端数据交换中，在保证稳定可靠的同时兼备高效与通用。对于将在新疆奇台县建设的QTT（奇台110米射电望远镜），各设备之间通信数据大小由观测波段和观测模式决定。它们之间的单次数据通信量一般小于1KB。天线伺服控制的数据通信最频繁，它的数据交换频率约为20Hz，单次数据交换大小约200B；其它设备的数据交换频率约为1Hz。主动面运行时，它的数据通信量约10KB；电磁监测的数据通信量一般为10KB～100KB。QTT控制系统拟采用分布式架构，各子系统之间的数据交换包含多种方式，如，Linux、Windows、VxWorks、Unix和嵌入式等系统之间的信息传输；网络的大端模式与机器的小端模式之间的信息传输；C++与Python之间的信息交换等。其中，大端模式的机器与小端模式的机器传输信息时，long类型数据的前后字节会互换。为了解决系统之间传输信息的数据交换格式问题，序列化工具将射电望远镜系统之间的传输信息编码为统一格式。因此，序列化工具作为控制系统传输信息格式的基础，可实现射电望远镜软硬件系统之间的信息传输。

现有的射电望远镜控制系统大多使用序列化技术[2]。如，ALMA的控制系统架构结合文本序列化工具XML作为控制系统传输信息编码和解码的基础[3]；ASKAP监视和控制系统使用网络通信引擎（ICE）提供的序列化技术，完成不同射电望远镜硬件、软件等系统之间传输信息格式[4]。GMRT监视和控制系统以Tango控制系统为基础，结合XML实现系统传输信息的编码和解码[5]。为了解决望远镜控制系统之间的信息传输格式问题，邓辉等人提出了以通信中间件ZeroMQ和文本序列化工具JSON为望远镜自动控制系统的通信框架[6]。

对于序列化工具中，XML和JSON是文本型序列化工具，它们被广泛应用于互联网软件系

***基金项目：**国家重点研发计划资助(2018YFA0404603)；中国科学院天文台站设备更新及重大仪器设备运行专项经费支持；中国科学院西部之光项目(XBQN-A-1)

收稿日期：xxxx-xx-xx；修订日期：xxxx-xx-xx

作者简介：李军，男，博士研究生，研究方向：大口径射电望远镜相关技术. Email:lijun@xao.ac.cn

通信作者：刘志勇，男，正高级工程师，研究方向：射电天文、大口径射电望远镜相关技术. Email:liuzhy@xao.ac.cn

统，以及早期射电望远镜控制系统的应用层与服务层之间的数据交换。在射电望远镜控制系统的使用中，发现XML和JSON存在一些不足，如内存使用率高、数据类型精度易缺失、难以实现底层设备驱动程序与服务之间的数据交换[7]。于是实验物理装置、射电望远镜等底层与服务层之间的通信逐渐被二进制序列化工具Msgpack、Protobuf、Flatbuffers[8-9] 所替代，它们可以更好地解决数据精度缺失、底层与服务层之间的数据交换效率等问题。本文着重分析Msgpack、Protobuf和Flatbuffers三款二进制序列化工具的编码原理、特性，通过测试、比较和分析它们的序列化数据大小、序列化时间和CPU利用率，兼顾底层、服务层和应用层，选择适合射电望远镜控制系统的序列化工具，以提高控制系统的信息传输效率，保证射电望远镜系统信息传输格式的统一性和兼容性。

1 序列化工具

序列化工具由编码（又称序列化）和解码（又称反序列化）构成。序列化是将结构化数据（或对象）编码为字节流；反序列化则是将字节流还原成原始的结构化数据（或对象）。

使用序列化工具构建射电望远镜控制系统时，需分析它的编码原理和特性。不同的编码方式会影响序列化数据大小、序列化时间、CPU利用率等。本章后续部分将以图1的JSON数据为例来分析Msgpack、Protobuf和Flatbuffers它们的编码原理。

```
{
    "names" : "zhang",
    "num" : 1331,
    "descript" : "inthefirstnicks"
}
```

图 1 JSON 格式示例

Fig. 1 An example of JSON schema.

1.1 Msgpack

Msgpack（又称MessagePack）是一款支持多语言、跨平台，具有动态编译的二进制序列化工具，编码对象（或结构化数据）之后的字节流具有紧凑、简洁的特点。字节流由头字节、前缀字节和数据字节构成。头字节表示之后紧跟的数据类型和类型个数；前缀字节表示其后的数据类型；数据字节表示对象的内容，如基本类型bin、float、uint，结构化类型str、array、map和扩展类型ext、fixext等。其中，字符串str不使用任何标记（或任何转义字符）表示内容结束。Msgpack的编码方式包括两种：

第一种方式使用Msgpack编码key-value值（这种方式简写为MSGP-M），需先对key值编码，再编码value值。

图2和图3分别表示MSGP-M编码图1的JSON数据之后的逻辑图和字节流图。逻辑图为编码之后的数据表示形式；字节流图则是编码之后各字节的先后顺序，如83为第一个字节。字节流由7部分组成：1）第1个字节（83）的前四位（1000）表示编码的数据类型为map，后四位（0011）表明之后包含3个map对象；2）第2个字节为第一个map对象中key值的前缀字节（A5），表明后续包含5个str对象。第3至7个字节以ASCII码表示map对象中的key值“names”；3）第8个字节（A5）表示后续包含5个字符串。第9-13个字节使用ASCII码表示字符串“zhang”；4）第14个字节表示第二个map对象中key值的前缀字节（A3），表明后续包含3个字符串。第15-17个字节以ASCII码表示字符串“num”；5）第18个字节为第二个map对象value值的前缀字节（CD）表明其后紧跟2个字节的无符号整数。第19-20个字节则以大端模式表示数字1331；6）第21个字节表示第三个map对象的key值前缀字节（A8），表明紧跟8个字符串。第22-29字节使用ASCII表示字符串“descript”；7）第30个字节（AF）为第三个map对象的value值的前缀字节（A5），表明后续紧跟15个字符串。第31-45字节表示字符串“inthefirstnicks”。

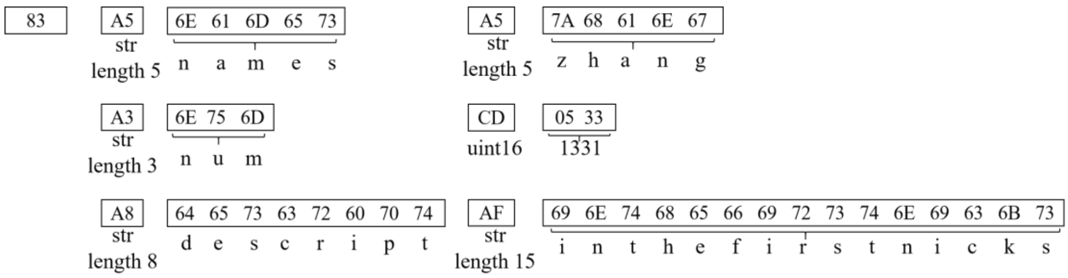

图 2 MSGP-M 对 JSON 格式中 key-value 值编码之后的逻辑图

Fig. 2 Logic diagram after MSGP-M encodes the key-value in JSON format

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
83	A5	6E	61	6D	65	73	A5	7A	68	61	6E	67	A3	6E	75	6D	CD	05	33	A8	64	65

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
73	63	72	60	70	74	AF	69	6E	74	68	65	66	69	72	73	74	6E	69	63	6B	73

图 3 MSGP-M 对 JSON 格式中 key-value 值编码之后的字节流图

Fig. 3 MSGP-M encodes byte stream after the key-value in the JSON format

第二种使用Msgpack编码JSON格式中的value值——序列化结果以数组表示。这种方式的Msgpack可缩写成MSGP-D。

MSGP-D编码图1的JSON格式后，分别对应图4的逻辑图和图5的字节流图，包括4部分：1）第1个字节（0X93）表示后续包含3个array对象；2）第2个字节（0XA5）表示其后有5个字符串。第3-7字节使用ASCII表示字符串“zhang”；3）第8个字节（0XCD）表明其后包含一个16位无符号整数。第9-16字节以大端模式的二进制形式表示数字1331；4）第11字节（0XAF）后续紧跟15个字符串。第12-26字节则以ASCII表示字符串“inthefirstnicks”。

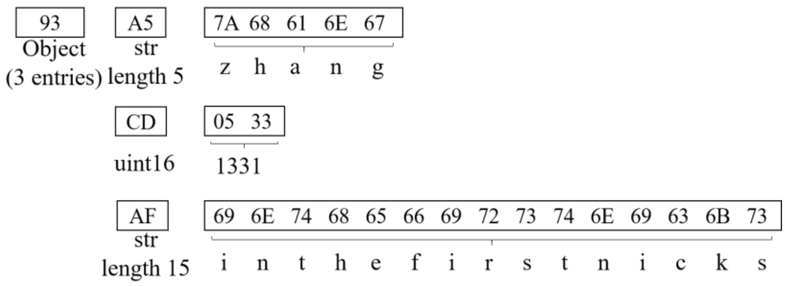

图 4 MSGP-D 对 JSON 格式中 value 值编码后的逻辑图

Fig.4 Logic diagram after MSGP-D encodes value in JSON format

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
93	A5	7A	68	61	6E	67	CD	05	33	AF	69	6E	74	68	65	66	69	72	73	74	6E	69	63	6B	73

图 5 MSGP-D 对 JSON 格式中 value 值编码后的字节流图

Fig.5 MSGP-D encodes byte stream of the value in the JSON format

1.2 Protobuf

Protobuf（PB）是一款开源，支持多语言、跨平台，提供接口描述语言（IDL）的二进制序列化工具。PB编码传输信息时，需定义IDL的键和字段（数据类型），以生成指定编程语言代码，如，C++、Python等。使用PB编码数据后，得到的字节流由键、前缀字节和数据字节组成。键分为标记数字和标记类型。标记数字将常用（或重复）元素标记为1~15，而不常用元素标记为16~2047；标记类型包括string、float等。键可对IDL文件中的数据类型进行唯一标记——标记后的数据类型不能更改。

图6和图7分别为PB编码图1的JSON数据之后得到的逻辑图和字节流图。其中，字节流占

用空间为27字节，由3个步骤组成：1）第1个字节（0A）中的第2位到第5位（0001）为数字标记，后三位（010）表示数据类型为string；第2个字节（05）表示后续包含5个string对象。第3-7字节以ASCII的形式表示字符串“zhang”；2）第8个字节（10）表示后续包含一个整型的数据。第9-10个字节是以小端模式表示16位的无符号整数1331；3）第11个字节（10）表示后续包含字符串。第12个字节（0F）则表示后续有15个字符串。第13-27字节表示字符串“inthefirstnicks”。

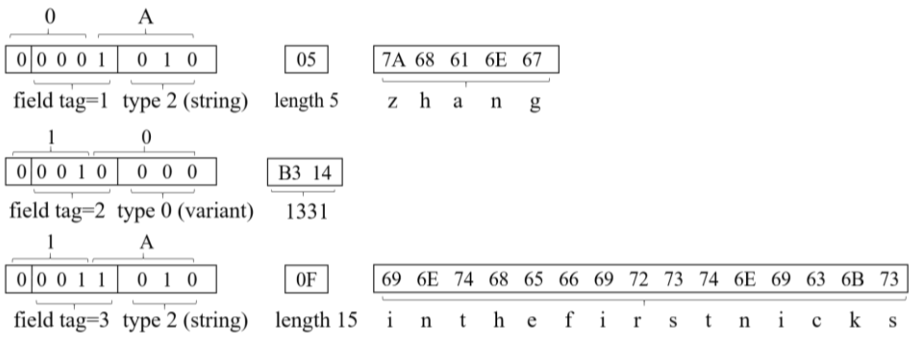

图 6 Protobuf 对 JSON 格式中 value 值编码后的逻辑图

Fig. 6 Logic diagram after Protobuf encodes value in JSON format

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0A	05	7A	68	61	6E	67	10	B3	14	1A	0F	69	6E	74	68	65	66	69	72	73	74	6E	69	63	6B	73

图 7 Protobuf 对 JSON 格式中 value 值编码后的字节流图

Fig. 7 Protobuf encodes the result of value in JSON format

1.3 Flatbuffers

Flatbuffers（FB）是一款支持多语言、跨平台，提供IDL的二进制序列化工具。FB具备良好的兼容性，如，系统添加新功能时，新字段只能在IDL文件末尾添加，且旧字段仍会正常读取；数据在内存中的格式与编码格式一致；反序列化过程支持“零拷贝”，便于快速读取数据。FB序列化字节流包括int、string等标量和struct、table等矢量。标量由固定长度的以小端模式表示的整型（8位~64位）和浮点型构成；矢量则由字符串和数组构成，开头必须是一个32位长度的VECTOR SIZE来指明矢量长度——不包括‘\0’和本身占用空间大小。其中，字符串和数组的唯一区别是字符串包含一个结束符‘\0’。

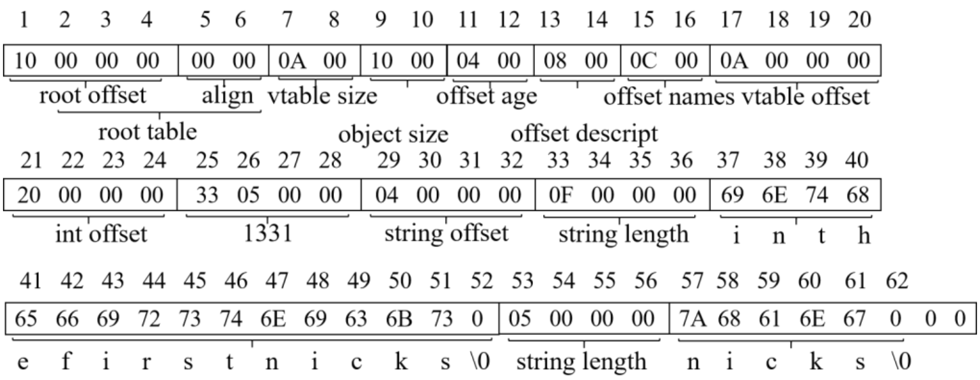

图 8 Flatbuffers 对 JSON 格式中 value 值编码后的字节流图

Fig. 8 Flatbuffers encodes the byte stream of the value in JSON format

图8为FB编码图1中数据的结果，字节流占用空间大小为62字节，由7部分组成：1）第1~4字节root offset（10 00 00 00为根偏移量，它偏移16个字节之后为编码数据。第5-6字节align（00 00）具有填充作用；2）第7~8字节vtable size（0A 00）为表vtable的字节大小。它包括vtable size、object size、offset num、offset descript和offset names占用的空间大小。第9~10字

节object size（10 00）表示在表中存储数据占用的空间偏移大小，它包括vtable offset、int offset、1331和string offset。第11~12字节offset num（04 00）表示num在字节流中的位置， offset只需移动4个字节便能找到num的偏移量。第13~14字节offset descript（08 00）为descript的位置，通过vtable offset和int offset便能找到descript的位置。第15~16字节offset name（0C 00）表示vtable offset，int offset和string offset之后为name对象；3）第17~20字节vtable offset（0A 00 00 00）与vtable size具有相同的大小，唯一区别是后者占2个字节。第21~28字节分别表示num的前缀和以小端模式表示的数字1331。第29~32字节（0F 00 00 00）表示后续类型为string。第33~36字节（0F 00 00 00）表明后续包含15个字符串。第37-52字节中包含15个以ASCII表示的“inthefirstnicks”和一个结束字符“\0”。第53-62字节与第33~52字节的编码原理相同。

1.4 编码原理分析与对比

对于三款二进制序列化工具，Msgpack不使用IDL来预先设置数据结构，可手动编写字段，具有两种编码方式；Protobuf和Flatbuffers在IDL中定义传输的信息字段，使用IDL编译器生成对应的编程语言接口，且只有一种编码方式。三款二进制序列化工具的编码原理不同，编码之后的字节流占用空间大小不同。Msgpack编码的字节流只有头字节，以及一一对应的前缀字节和数据字节。Protobuf的字节流中包含一一对应的键、前缀字节和数据字节。其中，只有数据类型使用键和数据字节。Flatbuffers序列化之后的字节流与数据在内存中的存储格式一致。Flatbuffers字节流中不仅包括前缀字节和数据字节，还包括root offset，object size，vtable offset等不能表示内容的字节。对同一数据格式编码，MSGP-M序列化JSON格式的全部数据，占用空间大。MSGP-D编码之后的字节流占用空间最小，字节流不用于表示信息的只有头字节和前缀字节。Protobuf占用的空间稍大，字节流中不能表达数据信息只包含一一对应的键和前缀字节。Flatbuffers占用空间大，字节流中包含大量不能表达数据的信息。

2 实验结果与分析


```
{
    "TelStatus": {
        "AzFlag": 0,  "ElFlag": 0, "TimeFlag": 0, "ServerFlag": 0, "PositionFlag": 0, "Az":
        54.321345,  "El": 44.321345, "AzRotation": 1,   "ElRotation": 1,
        "AzSpeed": 0.4234, "ElSpeed": 0.2345, "AzDiff": 1.23455, "ElDiff": 2.12456,
        "AzLimit": 0, "ElLimit": 0, "Track": 0 },
    "Weather": {
        "Status": 1,
        "Date": "2020-10-01 12:23:34.2000",
        "WindTower": {"one": [0.232, 230],"two":[0.3321, 230], "three": [0.4332, 230],"four":
        [0.4921, 230], "five":[0.6112, 230],"six":[0.7021, 230], "seven": [0.832, 230],
        "eight":[0.9431, 230], "nine": [1.232, 23],"ten":[1.5321, 230], "eleven": [1.862, 230]
        ,"twelve":[2.3321, 230]},
        "MeteoroInstrument": {
            "Temperature": 23.43,  "Pressure": 840.543, "Humidity": 0.3567, "Speed": 3.21,
            "Dir": 32.3213},
        "Plot": {
            "SpeedRose": [1.322689, 0.241007, 2.210769, 3.381295, 1.353786, 0.224426, 0.
            064588, 1.285891, 0.436123, 1.435216, 0.180438, 1.169811, 1.194245, 0.230022, 3.
            083807, 0.183219],
            "DirRose": [0.122689, 0.041007, 0.010769, 0.081295, 0.053786, 0.024426, 0.064588,
            0.085891, 0.036123, 0.035216, 0.080438, 0.069811, 0.094245, 0.030022, 0.083807, 0.
            03219]}
    }
}
```


图 9 射电望远镜系统之间的数据传输格式

Fig. 9 Data transmission format between radio telescope systems

Msgpack、Protobuf和Flatbuffers不仅运行于Linux、Windows等操作系统，还支持C、C++、 Python等编程语言。然而，控制系统开发往往使用多种编程语言，其中，C、C++用于底层驱动程序开发和通信；Python用于服务端的开发以及天文学家对数据进行处理等。因此，对于三款二进制序列化工具的测试，测试环境的CPU为2.0GHz的Intel Core i7-4750，内存为8GB，操作系统为Ubuntu 16.04。编译环境的GCC版本为5.3.1，Python版本为3.7.3。Msgpack、Flatbuffers和Protobuf的版本分别为1.2.1、1.1.0和3.7.1。

图9为一个数据交换格式的展示，所传输的信息为控制系统中射电望远镜的状态信息编码。“TelStatus”为射电望远镜的状态信息，主要包括天线方位标签“AzFlag”、天线俯仰标签“ElFlag”、子系统时间标签“TimeFlag”、望远镜状态标签“ServerFlag”、天线所处位置标签“PositionFlag”等。“Weather”表示天文台站周围的气象信息，如，气象设备状态、日期、风塔、气象仪器和风玫瑰图。“WindTower”阐述气象仪器的温度、气压、湿度、风速、风向等五要素。“Plot”表示风玫瑰图用以统计台站周围一段时期内风向、风速等。控制系统只对传输的数据进行一次编解码。由于传输的数据中包含浮点型和双精度型数据，编码之后的数据在控制系统中不能以ASCII编码传输。是因为下面以图9为例，使用C++和Python测试三款二进制序列化工具的序列化数据大小、序列化时间和CPU利用率。

2.1 序列化数据大小

Msgpack有两种方式的编解码，既能编码和解码JSON格式的key-value值，又能编码和解码JSON格式中的value值。使用三款二进制序列化工具分别测试图9的数据，得到MSGP-M、MSGP-D、Protobuf和Flatbuffers的字节流大小分别为713B、460B、520B、794B。因此，序列化数据大小与编码原理密切相关。MSGP-M较MSGP-D占用空间大，是因为MSGP-D只编码图1中的key值。MSGP-D的字节流表示为一个头字节、一一对应的前缀字节和数据字节，而Protobuf的字节流包含一一对应的键、前缀字节和数据字节。Flatbuffers占用空间大是因为其不仅编码key-value中的value值，还包括非数据值，如root offset、int offset、float offset等。因此，MSGP-D比Protobuf和Flatbuffers输出格式更紧凑、占用空间更小。

2.2 序列化时间

Msgpack、Protobuf和Flatbuffers的编解码原理不同，使序列化时间和反序列化时间存在差异。以图9为例，三款二进制序列化工具迭代100,000次之后，得到它们的单次平均序列化时间详见图10。MSGP-M（C++）的序列化时间为22.425微秒，反序列化时间为52.491微秒；Python的序列化时间为15.566微秒，反序列化时间为10.896微秒。其中，C++的序列化时间比反序列化时间短，Python的序列化时间比反序列化时间短，是因C++的基本数据类型多，解码时间长；而Python基本类型少、能更好匹配key-value值，解码时间短。MSGP-M（C++）的序列化时间为22.425微秒较MSGP-D（C++）的15.566微秒时间长，是因为MSGP-M需要编码key-value，而MSGP-D只需编码Value值。同理，解码与编码的原理相似。Protobuf（C++）序列化时间为10.514微秒，反序列化时间为17.163微秒；Python序列化时间为86.506微秒，反序列化的时间为62.431微秒，两种编程语言各自的序列化和反序列化时间接近，是由PB的IDL决定。Flatbuffers（C++）序列化时间为5.446微秒，反序列化时间为0.344微秒；Python的序列化时间为203.718微秒，反序列化时间为2.130微秒。Flatbuffers的序列化时间比反序列化时间长，是因为编码之后的字节流与其在内存中的数据格式一致，解码不需要时间，只需读取IO的时间。

从图10中可知，对于同一种二进制序列化工具的不同编程语言，Flatbuffers（C++）的序列化时间比Python的序列化时间快40倍。Protobuf（Python）的序列化时间是C++的序列化时间的8倍以上。MSGP-M或MSGP-D同一种编码方式的C++和Python的序列化时间与反序列化时间接近，不会因编程语言的不同造成序列化和反序列时间的不平衡。

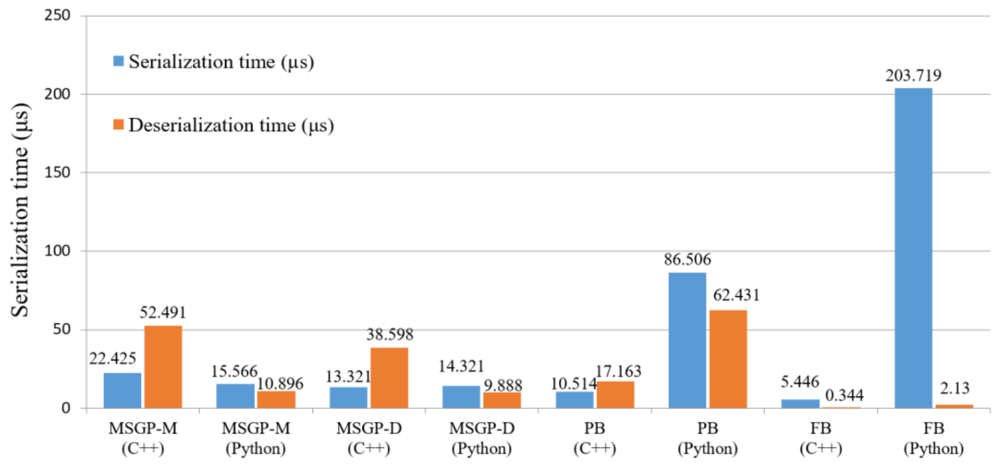

图 10 三款二进制序列化工具的执行时间

Fig. 10 Execution time for three binary serialization tools

2.3 CPU利用率

程序运行时CPU利用率的高低会影响程序运行。以图9展示的数据为例，测试三款二进制序列化工具得到表1的结果。从表中可知，无论是C++还是Python，Msgpack编解码的CPU利用率均在12.4%左右。Protobuf在C++和Python的CPU利用率也是12.4%。然而，Flatbuffers编码和解码的CPU利用率却存在差异。在编码时，Flatbuffers的Python的CPU利用率达到25.9%，远高于Msgpack和Protobuf编码时C++和Python的CPU利用率；而解码时，Flatbuffers的CPU利用率相比Msgpack和Protobuf略低。因此，Msgpack和Protobuf适用于服务端和客户端内存充足的场景，而Flatbuffers可应用于服务端内存充足、客户端内存不足的场景。然而，射电望远镜控制系统在实际应用中的服务端和客户端内存相似，在CPU利用率方面，Msgpack和Protobuf的总体性能明显优于Flatbuffers。

表 1 三款二进制序列化工具的 CPU 利用率

Table 1 CPU utilization of three binary serialization tools

Name	MSGP-M		MSGP-D		Protobuf		Flatbuffers	
	C++	Python	C++	Python	C++	Python	C++	Python
encode CPU usage	12.45%	12.44%	12.38%	12.36%	12.37%	12.57%	12.48%	25.9%
decode CPU usage	12.5%	12.37%	12.48%	12.4%	12.47%	12.49%	11.82%	12.21%

3 总结

本文分析和比较了Msgpack、Flatbuffers和Protobuf的编码原理和特性，并对它们进行了测试和分析。Msgpack不需要IDL，只需开发人员编写代码实现编解码的功能；而Flatbuffers和Protobuf使用IDL对传输的信息进行编解码。它们对同一信息编解码时，MSGP-D字节流的大小和多语言的序列化时间优于Protobuf，且明显优于Flatbuffers。MSGP-M对需求变化大的小数据编码具有优势，它可对同一数据以任意顺序的key-value数据进行编解码，但Protobuf和Flatbuffers却不能对这种方式进行解码。根据射电望远镜控制系统的开发情况，当通信的数据格式确定时，可使用MSGP-D；而通信的数据格式变化较大时，可使用MSGP-M。总之，通过分析三款二进制序列化工具，Msgpack更适合射电望远镜控制系统的信息传输，有助于射电望远镜的硬件系统、软件系统、操作系统、编程语言和网络之间的信息交换，使系统的扩展性好，移植性强等特性。

Serialization analysis of data transmission in control system of radio telescope

Li Jun[1,2], Na. Wang[1,3], Liu Zhiyong[1,3], Song Yining[1,2], Yang Lei[1,2], Wang Jili[1]

（1. Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011, China;

2.University of Chinese Academy of Sciences, Beijing 100049, China; 3. 100049, China;

3. Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing 210008, China）

Abstract: The control system can connect, integrate and manage the software and hardware systems of the radio telescope. Serialization tool in the control system encodes and decodes the information transmitted between different devices, operating systems, programming languages, and networks in the radio telescope, enhancing the efficiency rate of data transmission between systems. This article analyzes and compares the coding principles and characteristics of the three binary serialization tools Msgpack, Protobuf and Flatbuffers, and tests their serialized data size, serialization time, and CPU utilization through an example. The results show that the overall performance of Msgpack is better than that of Protobuf and Flatbuffers, and it is suitable for encoding and decoding of transmission information between radio telescope systems with long periods and variable requirements.

Key words: Radio Telescope; Binary serialization tool; Control system; Encode; Decode

参考文献:

[1] Wang Jian , Liu Jiajing, Tang Pengyi, et al. a study on generic models of control systems of large astronomical telescopes[J]. Publications of the Astronomical Society of the Pacific, 2013.

[2] A Götz, E.T. Taurel, P.V. Verdier. TANGO - Can ZMQ Replace CORBA ?[C]. Proceedings of the 14th International Conference on Accelerator & Large Experimental Physics Control Systems. California. 2013.

[3] Heiko Sommer and GianlucaChiozzi , et al. Transparent XML Binding using the ALMA Common Software（ACS）Container/Component Framework[J]. 2004.

[4] Juan C. Guzman，Ben Humphreys. The Australian SKA Pathfinder（ASKAP）software architecture[C]// Spie Astronomical Telescopes + Instrumentation. International Society for Optics and Photonics, 2010.

[5] KodilkarJ ,Uprade R , Nayak S , et al. Developments of next generation monitor and control systems for radio telescopes[J]. Iop Conference, 2013, 44:012026.

[6] 邓辉,钟文杰,付映雪, 等. 基于ZeroMQ的新一代望远镜自动控制系统的通信框架设计[J].天文研究与技术，2018，15（03）：308-314.

Deng Hui, ZhongWenjie , Fu Yingxue, et al. The Design of Communication Framework for a New Generation of Telescope Autonomous Control System Based on ZeroMQ[J]. Astronomical Research &Technology, 2018,15（03）:308-314.

[7] Daradkeh T , Agarwal A, Goely N , et al. Real Time Metering of Cloud Resource Reading Accurate Data Source Using Optimal Message Serialization and Format[C]// 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). IEEE, 2018.

[8] Marassi A . Status of the Local Monitor and Control System of SKA Dishes[C]// ICALEPCS. 2015.

[9] Clarke M.J., Akeroyd F.A., Arnold O., et al. Live Visualisation of Experiment Data at ISIS and the ESS [C]// ICALEPCS 2017 - International Conference on Accelerator and Large Experimental Physics Control Systems. 2017.