Protocol Buffers: Marshalling

Vojtěch Horký Petr Tůma

2010 – 2022

This work is licensed under a "CC BY-NC-SA 3.0" license. Created to support the Charles University Performance Evaluation lecture. See http://d3s.mff.cuni.cz/teaching/introduction-to-middleware for details.

Contents

1 Technology Overview 1

2 Assignment Part I 2

3 Message Encoding 2

4 Message Specification 3

5 Message Manipulation 4

6 Assignment Part II 6

1 Technology Overview

Goals
Provide platform independent structured data serialization framework.

Features

– Platform independent data description language.
– Serialization code generation for multiple languages (C++, Java, Python, Go, Ruby, JavaScript, Objective C, C# ...).
– Binary transport format with compact data representation.
– Textual transport using JSON.

... http://developers.google.com/protocol-buffers

Examples To Play With ...

> git clone http://github.com/d-iii-s/teaching-introduction-middleware.git

C

> cd teaching-introduction-middleware/src/protocol-buffers-basic-usage/c
> cat README.md

Java
Python

```python
message AnExampleMessage {
  uint32 some_integer = 1;
  sint32 another_integer = 2;
  string some_string = 8;
  repeated string some_more_strings = 11;
}

message MoreExampleMessages {
  repeated AnExampleMessage messages = 1;
}
```

2 Assignment Part I

Assignment

Server
Implement a server that will provide information on current time.
- The server should accept a spec of what fields to return.
- Fields should be standard YYYY-MM-DD HH:MM:SS.

Client
Implement a client that will query server time:
- Pick a random combination of fields.
- Query information on current time.
- Print the time.

Interoperability
Implement compatible clients and servers in two languages.

3 Message Encoding

Goals
Compact structure with support for field removal and addition.

Features
- Sequence of field key value pairs.
- Key is field index and type indication.
  - One of variable integer, explicit length, fixed length.
– Not enough to tell the exact field type!
– Primitive repeated fields packed.
– Total length not included!

**Variable Length Encoding**

**Goals**
Support integers clustered around zero more efficiently.

**Features**
– Integer stored as variable number of 7 bit values.
– High bit set to zero for last byte.
– Little endian byte order.
– Signed variant.

4 Message Specification

**Primitive Field Types**

**Integer Types**

(s)fixed(32|64) Integers with fixed length encoding.
(u)int(32|64) Integers with variable length encoding.
sint(32|64) Integers with sign optimized variable length encoding.

**Floating Point Types**

float IEEE 754 32 bit float.
double IEEE 754 64 bit float.

**Additional Primitive Types**

bool Boolean.
bytes Arbitrary sequence of bytes.
string Arbitrary sequence of UTF-8 characters.

**More Field Types**

**Oneof Type**

message AnExampleMessage {
  oneof some_oneof_field {
    int32 some_integer = 1;
    string some_string = 2;
  }
}

**Enum Type**

enum AnEnum {
  INITIAL = 0;
  RED = 1;
  BLUE = 2;
  GREEN = 3;
  WHATEVER = 8;
}
More Field Types

Any Type

```cpp
import "google/protobuf/any.proto";
message AnExampleMessage {
  repeated google.protobuf.Any whatever = 8;
}
```

Map Type

```cpp
message AnExampleMessage {
  map<int32, string> keywords = 8;
}
```

5 Message Manipulation

### C++ Message Basics

**Construction**

```cpp
AnExampleMessage message;
AnExampleMessage message (another_message);
message.CopyFrom (another_message);
```

**Singular Fields**

```cpp
cout << message.some_integer ();
message.set_some_integer (1234);
```

**Repeated Fields**

```cpp
int size = messages.messages_size ();
const AnExampleMessage &message = messages.messages (1234);
AnExampleMessage *message = messages.mutable_messages (1234);
AnExampleMessage *message = messages.add_messages ();
```

### C++ Message Serialization

**Byte Array**

```cpp
char buffer [BUFFER_SIZE];
message.SerializeToArray (buffer, sizeof (buffer));
message.ParseFromArray (buffer, sizeof (buffer));
```

**Standard Stream**

```cpp
message.SerializeToOstream (&stream);
message.ParseFromIstream (&stream);
```

### Java Message Basics

**Construction**
AnExampleMessage.Builder messageBuilder;
messageBuilder = AnExampleMessage.newBuilder ();
messageBuilder = AnExampleMessage.newBuilder (another_message);
AnExampleMessage message = messageBuilder.build ();

Singular Fields
System.out.println (message.getSomeInteger ());
messageBuilder.setSomeInteger (1234);

Repeated Fields
int size = messages.getMessagesCount ();
AnExampleMessage message = messages.getMessages (1234);
List<AnExampleMessage> messageList = messages.getMessagesList ();
messagesBuilder.addMessages (messageBuilder);
messagesBuilder.addMessages (message);

Java Message Serialization

Byte Array
byte [] buffer = message.toByteArray ();
try {
  AnExampleMessage message = AnExampleMessage.parseFrom (buffer);
} catch (InvalidProtocolBufferException e) {
  System.out.println (e);
}

Standard Stream
message.writeTo (stream);
AnExampleMessage message = AnExampleMessage.parseFrom (stream);

Python Message Basics

Construction
message = AnExampleMessage ()
message.CopyFrom (another_message)

Singular Fields
print (message.some_integer)
message.some_integer = 1234

Repeated Fields
size = len (messages.messages)
message = messages.messages [1234]
message = messages.messages.add ()

Python Message Serialization

Byte Array
buffer = message.SerializeToString ()
message.ParseFromString (buffer)
message = AnExampleMessage.FromString (buffer)
Standard Stream

```python
file.write(message.SerializeToString())
message.ParseFromString(file.read())
AnExampleMessage.FromString(file.read())
```

Code Now …

Show Your Code …

Query Host Name

```bash
> hostname
u1-22
```

Run Screen Sharing

```bash
> x11vnc -viewonly
```

6 Assignment Part II

Assignment

Performance

Measure the performance of your implementation.

Experiment Design

Stick to the following, or provide arguments for why not:
- Random field mix, each field with probability 1/2.
- Measure at least two minutes long traffic.

- Report average invocation throughput.
- No printing during measurement.
- Compare with past assignments.

**Submission**

**GitLab**
Use your personal GitLab repository under [https://gitlab.mff.cuni.cz/teaching/nswi163/2022](https://gitlab.mff.cuni.cz/teaching/nswi163/2022)

**Requirements**
- Use the assignment subdirectory.
- Write brief report in SOLUTION.md.
- Include build scripts with instructions.
- Do not commit binaries or temporary build artifacts.
- Tag your solution with task-02 and push the tag.