Google protobuf and large binary blobs

Asked 10 years, 9 months ago Modified 8 years ago Viewed 23k times

10

I'm building a software to remotely control radio hardware which is attached to another PC.

I plan to use ZeroMQ for the transport and an RPC-like request-reply with different messages on top of it which represent the operations.

While most of my messages will be just some control and status information, there should be an option to set a blob of data to transmit or to request a blob of data to receive. These data blobs will usually be in the range of 5-10MB but it should be possible to also use larger blobs up to several 100MB.

For the message format, I found the google protocol buffers very appealing because I could define one message type on the transport link which has optional elements for all the commands and responses. However, the protobuf FAQ states that such large messages will negatively impact performance.

So the question is, how bad would it actually be? What negative effects are there to expect? I don't really want to base the whole communications on protobuf only to find out that it doesn't work.

protocol-buffers rpc

Share

Improve this question

Follow

asked Mar 10, 2014 at 18:52

jan

987 ● 2 ● 9 ● 21

2 Answers

Sorted by: Highest score (default)

19

Frankly, it's not so much performance *per se* as that the library is not designed in a way you might want it to be for dealing with large messages. For example, you have to parse a message all at once, and serialize it all at once. So if you have a message containing a 100MB blob, you can't read any part of the message unless you read in the entire 100MB and block the calling thread while it parses. Also problematic is the fact that the 100MB blob will be allocated as one gigantic flat byte array. On 64-bit systems this may be fine but on 32-bit you may have address space fragmentation issues. Finally, there is a hard message size limit at 2GB.

If you are OK with these sorts of issues, then you can pretty much just do it. You will have to manually override the message size limit which for security purposes defaults to 64MB. To do this, you need to construct a `CodedInputStream` manually and call `SetTotalBytesLimit()` on it before parsing the message from it.

But personally I'd recommend trying to design your system such that big blobs can be split up into small chunks.

Share Improve this answer Follow

edited Dec 14, 2016 at 5:21

answered Mar 12, 2014 at 9:53

Kenton Varda
44.9k ● 9 ● 129 ● 112

2 Chunking is the recommended approach: *Protocol Buffers are great for handling individual messages within a large data set. Usually, large data sets are really just a collection of small pieces, where each small piece may be a structured piece of data. Even though Protocol Buffers cannot handle the entire set at once, using Protocol Buffers to encode each piece greatly simplifies your problem: now all you need is to handle a set of byte strings rather than a set of structures.* - developers.google.com/protocol-buffers/docs/… – S Balough Oct 19, 2015 at 21:12

16 @SBalough Indeed, I wrote that text. :) – Kenton Varda Oct 20, 2015 at 5:46

I'd like to send images over the protocol buffer. They aren't big, maybe 50kb to 100kb (2MB max). Is protobuf an inappropriate tool for this? – speedplane Mar 23, 2017 at 2:26

2 @speedplane It should be fine if you put them in a `bytes`-typed field. – Kenton Varda Mar 26, 2017 at 23:43

I don't have time to do this for you, but I would browse the Protobuf source code. Better yet, go ahead and write your code using a large `bytes` field, build protobuf from source, and step through it in a debugger to see what happens when you send and receive large blobs.

From experience, I can tell you that large `repeated Message` fields are **not** efficient unless they have the `[packed=true]` attribute, but that only works for primitive types.

My gut feeling is that large `bytes` fields will be efficient, but this is totally unsubstantiated.

You could also bypass Protobuf for your large blobs:

```
message BlobInfo {
    required fixed64 size;
    ...
}

message MainFormat {
    ...
    optional BlobInfo blob;
}
```

then your parsing code looks like:

```
...
if (message.has_blob()) {
    uint64_t size = msg.blob()->size();
    zmqsock.recv(blob_buffer, size);
}
```

Share Improve this answer

Follow

answered Mar 10, 2014 at 19:16

japreiss
11.3k ● 2 ● 41 ● 78