Jmeter & gRPC

gRPC is an open source API that also falls within the category of RPC. Unlike SOAP, however, gRPC is much newer, having been released publicly by Google in 2015. (That said, the history of gRPC dates back to an internal project at Google called Protocol Buffers that started in 2001.)

Like REST and SOAP, gRPC uses HTTP as its transport layer. Unlike these other API protocols, however, gRPC allows developers to define any kind of function calls that they want, rather than having to choose from predefined options (like GET and PUT in the case of REST).

Another important advantage of gRPC, at least for many use cases, is that when you make a call to a remote system using gRPC, the call appears to both the sender and the receiver as if it were a local call, rather than a remote one executed over the network. This simulation avoids much of the coding complexity that you'd otherwise have to contend with in order for an application to handle a remote call.

The ability of gRPC to simplify otherwise complex remote calls has helped make it popular in the context of building APIs for microservices or Docker-based applications, which entail massive numbers of remote calls.

More info you can find here.

In order to execute performance tests with gRPC requests with Jmeter you can use one of the following:

1. A simpler of JMeter used to test for any gRPC server that it is not necessary to generate gRPC classes or to compile the protos binary for the service. You can find it here.

2. A JMeter plugin supports load test grpc service. You can find it here.

3. Build you own Java request sampler by following the instructions found here.

Note: we tested all three options but only the third one worked. Attached there is the code of the service and the client. The build of the client can be packaged in a jar and included in the ext folder of Jmeter. And after starting the server you can prepare the jmeter project by adding a java request, selecting the TestingServiceSampler and updating the properties accordingly.