Some aspects of using RPC

Sandor Kiraly[a], Szilveszter Szekely[b], Roland Kiraly[c], Tamas Balla[d]

[a]Eszerhazy Karoly University
kiraly.sandor@uni-eszterhazy.hu

[b]Imperial College London
szekelyszilv@gmail.com

[c]Eszterhazy Karoly University
kiraly.roland@uni-eszterhazy.hu

[d]Eszterhazy Karoly University
balla.tamas@uni-eszterhazy.hu

Abstract

In a client-server network, sockets provide a mechanism for a program to establish a connection to another program and send messages back and forth. This interface underlies the working of a mechanism that allows a program running as a process on computer A to call a procedure or a function on computer B, pass parameters to it and have the result returned. After the call, the caller process on A is suspended and execution continues on B. When the callee procedure or function on computer B finishes and produces its results, it is passed back to the calling environment on computer A. Then the process on A continues the execution from where it was suspended. This mechanism is called the Remote Procedure Call (RPC). The question is how long must the process wait for the answer to arrive from computer B and why? To answer the question, this paper describes the structure of third generation RPCs and analyses them, putting the focus on the way of marshalling parameters, and performance. To facilitate the choice between them this paper represents the results of performance tests carried out by the authors. The tests are implemented in Java, Ruby, and C++ using GRPC, XML-RPC, and JSON-RPC with calls between the languages to create a broader picture of their performance characteristics.

Keywords: Remote Procedure Call, Google Protocol Buffer, marshalling

MSC: 68N19, 68M14, 68M12