GPars - Groovy Parallel Systems

Jon Kerridge

Version 1.0, 2015-10-29

Table of Contents

Client-Server: Deadlock Avoidance by Design

`Chapter 7` demonstrated, with two examples, one obvious and the other less so, that deadlocked systems can be constructed quite easily even if the thought given to the design would suggest otherwise.

A design pattern is required that ensures **deadlock** and also **livelock** freedom. *Brinch Hansen* formulated a design approach for operating systems in the 1970s based upon a client-server architecture. It is a slightly updated version of that design approach that is presented here as the client-server design pattern. It is captured in two simple rules, together with a method for analysing a network.

1. A client process that issues a request to a server process guarantees to accept any response from that server immediately. A `client – server` interaction requires a client request upon the server but it is not necessary for there to be a communication from the server to the client process.
2. A server process that accepts a request from a client process guarantees to return a response to the client process within finite time. In addition, a server process will never send a message to any of its clients without having first received a request from a client. A server process can behave as a client to another server process.
3. Deadlock and livelock will not occur in such a network of client and server processes, provided a labelling of the client and server ends of the interactions between processes, does not result in a completed circuit of clients and servers.