GPars - Groovy Parallel Systems

Jon Kerridge

Version 1.0, 2015-10-29

Table of Contents

Deadlock Revisited: Circular Structures

In previous chapters, the concept of a `client-server` design pattern has been introduced and then it has been applied to a number of simple examples.

The primary requirement of the pattern is that any resulting network should not contain any circuits of client and server labels. Needless to say, that if we have a ring of processes then a circuit is inevitable. Hence, we shall investigate a ring of processes to explore how, even though the client-server pattern cannot be applied, we can construct a system that is deadlock free.

Needless to say, with a ring of processes then a circuit is inevitable

---

The aim of this application is to construct a message passing structure from one node to another by providing a set of message passing elements that connect each node to the next.

The simplest way of doing this is to create a ring of message-passing nodes to which message sender and receiver processes are attached to each node.

`Figure 10-1` shows the basic structure with a `client-server` labelling that demonstrates immediately that deadlock will occur, even ignoring the effect of the sender and receiver processes.

It is obvious that the set of channels that connect the `Ring Element` processes has to be broken in some way. **Deadlock** will occur, trivially, when every `Ring Element` attempts to either input or output a message at the same time. Thus we have to find a way of breaking the ring.

Want to read more of this chapter? Download this chapter's PDF here.
