Assignment 7: Review of concurrent languages
ETH Zurich

1 Distributed random adder

1.1 Background
For this task, you will work with X10. To install the X10 compiler, please visit the X10 website [1]. To install the command line compiler, you can find instructions in the section For Users. To install the Eclipse plugin, you can find instructions in the section X10DT Installation.

In the For Users section you can find a number of X10 tutorials. Another useful resource is the X10 library API description [2].

Consider an adder that computes the sum of distributed random numbers in parallel. The adder distributes the random numbers into fragments; for each fragment, the adder computes the local sum as the sum of the numbers in that fragment. The adder then computes the global sum as the sum of all local sums.

1.2 Task
Implement this adder in X10. You might find the classes Dist, DistArray, and Random useful. An object of type Dist distributes points across places. You can use the expression $d \mid \text{here}$ to restrict the distribution $d$ to the current place; you will get a distribution with only the points that are mapped to the current place. An object of type DistArray is an array whose elements are distributed according to a given distribution.

1.3 Solution
The solution can be found in the source code that comes with this solution.

References

2 Playing Telephone

2.1 Background
This exercise uses the Erlang programming language, available from the Erlang website at [1], which also includes documentation. Once you have the Erlang VM, you can start immediately using erl, the Erlang emulator. If you use Windows, you may want to use werl instead.

There is fairly comprehensive tutorial available at [2]. It is not necessary to read the entire document, as it is quite large. The introductory sections on syntax, basic values, types, and the available data-structures are useful to get started on basics of the language. For more specific
information on concurrency, “The Hitchhiker’s Guide to Concurrency” (only the last section) [3] and “More on Multiprocessing” [4] cover the necessary material.

2.2 Task

The task you are given is to implement a simple distributed chat server. Every member of the chat is represented by an Erlang process.

The requirements are as follows:

• Every process knows only about one other process, its neighbour.

• To join the chat, a process can ask any current member of the chat to participate.

• To send a message, a member-process can ask its neighbour to forward the message. Try to detect if the message cannot reach its destination.

• A process should also be able to leave the chat, without permanently breaking the chat for the remaining members.

Please note that the system does not have to be robust while members are being added or removed; for example, a simultaneous removal and send operation.

References


