Programación Distribuida

Modelo Cliente – Servidor

* Arquitectura para programación concurrente, hay un servidor que administra ciertos recursos y un número de clientes que mandan solicitudes al servidor para tener acceso a sus recursos.
* El cliente y el servidor son procesos separados, se comunican con mensajes normales de Erlang.
* Ambos pueden correr en la misma máquina – Programación concurrente.
* O en máquinas distintas – Programación distribuida.
  + Las máquinas se deben de poder ver entre sí.

Roles en el sistema

Cliente: inicia un cómputo al mandar una solicitud al servidor.

Servidor: calcula y le manda una respuesta al cliente.

* Ambos deben de conocer o mandar sus **PIDs** correspondientes.

Ejemplo: Areas

* Rpc = remote process control, se le manda el Pid del servidor

Map

Description automatically generated

1. Creación del servidor

Pid = spawn(fun áreas:ciclo/0).

1. Solicitudes de Clientes

Áreas:rpc(Pid, rectángulo, 6, 8}).

Encadenamiento de procesos

* Se utiliza cuando un proceso depende de otro
* BIF – link/1
* Ambos procesos se monitorean respectivamente.

Efecto de la señal de salida

* Si A muere, entonces B también muere.
* Pero si el receptor se convierte en un proceso del sistema, este prosigue después de la señal y puede reaccionar a la misma.

Ejemplo de encadenamiento

A picture containing text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Programación Distribuida

**Todos los primitivos en Erlang tiene la mismas propiedades en un sistema distribuido.**

Basado en el concepto **nodo**

**Nodo**

Sistema Erlang ejecutándose (**erl**) que puede tomar parte en transacciones distribuidas.

Un sistema distribuido consiste de varios nodos en una o varias computadoras conectadas.

Razones para escribir Aplicaciones distribuidas

1. Velocidad
   1. Ejecución paralela en nodos.
2. Confiabilidad y Tolerancia a Fallas
   1. Redundancia y cooperación de varios nodos.
3. Acceso a recursos que residen en otro nodo
   1. Base datos, periféricos, etc.
4. Distribución inherente de la aplicación
   1. Sistemas naturalmente distribuidos como para la reservación de vuelos.
5. Extensibilidad
   1. Escalar la capacidad del sistema agregando nuevos nodos.

Aplicaciones de Erlang

* Facebook: impulsar el backend de su servicio de chat.
* Amazon: para implementar SimpleDB.

Modelos de Programación

* **Erlang distribuido**: las aplicaciones se ejecutan en un ambiente de confianza entre computadoras fuertemente acopladas.
  + Los clientes y los servidores que se encuentran en diferentes nodos tienen acceso a todos los recursos en las otras computadoras.
  + Cualquier nodo puede realizar cualquier operación en cualquier otro nodo Erlang.
  + Típicamente, las aplicaciones se ejecutan en clústeres sobre la misma LAN detrás de un firewall.
  + Malas operaciones: borrar archivos.
* **Distribución basada en Sockets**, ambientes no confiables.

Galleta Mágica

* Es un átomo que deben compartir todos los nodos.
* Condición para que varias computadoras se puedan ver.
* Para que 2 nodos Erlang distribuidos se comuniquen deben tener la misma galleta mágica.

Métodos para utilizar la Magic Cookie

1. Almacenarla en $HOME/.erlang.cookie
2. Iniciar Erlang con erl *-setCookie Cookie* (Cuando se crea el nodo)
3. Utilizar la función *erlang:set\_cookie(Nodo, Cookie*), ya una vez creado el nodo.

Funciones predefinidas

* spawn/4
  + Nodo, Mod, Func, Args
  + Crea un proceso en un nodo remoto.
  + Deben especificar el nombre del nodo.
* spawn\_link/4
  + Nodo, Mod, Func, Args, crea un proceso remoto y lo liga al proceso.
* Monitor\_node/4
  + Nodo, Bandera
  + Si la bandera es true, monitorea al Nodo y en caso de que este falle o no exista regresa un mensaje de nodedown y el nodo al proceso.
* node/0
  + Regresa el nombre del propio nodo.
* nodes/0
  + Regresa lista de nombres de nodos conocidos
* node/1
  + Elemento
  + Regresa el nombre del Pid, referencia o puerto dado como Elemento.
* disconnect\_node/1
  + Nombre
  + Se desconecta del nodo Nombre.