Programação distribuída em Erlang

Paulo Ferreira
paf@dei.isep.ipp.pt

Fevereiro de 2006

Distribuição

Introdução

- Em vez de termos um nó computacional (uma instância do Erlang) podemos ter vários nós
- O que podemos fazer?
- Para quê?

Introdução

- Em vez de termos um nó computacional (uma instância do Erlang) podemos ter vários nós
- O que podemos fazer?
  - Paralelismo entre nós diferentes
  - Lançar processos em nós remotos
  - Comunicar com processos em nós remotos
  - Monitorizar nós remotos
- Para quê?
  - Fiabilidade e tolerância a falhas
  - Acesso a recursos remotos
  - Escalabilidade do sistema

Detalhes

- Diferentes nós podem coexistir no mesmo computador
- Os nós podem estar em máquinas diferentes (óbvio)
- Os nós podem estar em máquinas diferentes de arquitecturas e sistemas operativos diferentes
- O mecanismo de comunicação entre nós diferentes é normalmente feito por cima do TCP/IP
- Podem-se usar outros canais para comunicação (há o *source code*)

- Em Windows alguns antivírus/firewalls dão problemas
- A comunicação não é encriptada (mas pode ser[1])

[1] ver este ponto...

Tipos

Tipos de agrupamentos de nós Erlang

- Nós de nome curto
- Nós de nome comprido

Tipos

Tipos de agrupamentos de nós Erlang

- Nós de nome curto
  - Apenas comunicam dentro da mesma rede local
- Nós de nome comprido
  - Comunicam na Internet

Tipos

Tipos de agrupamentos de nós Erlang

- Nós de nome curto
  - Apenas comunicam dentro da mesma rede local
  - Exemplo: `no1@maquina`
- Nós de nome comprido
  - Comunicam na Internet
  - Exemplo: `no1@maquina.dominio.net`

Tipos

Tipos de agrupamentos de nós Erlang

- Nós de nome curto
  - Apenas comunicam dentro da mesma rede local
  - Exemplo: `no1@maquina`
  - Arranque: `erl -sname no1`
- Nós de nome comprido
  - Comunicam na Internet
  - Exemplo: `no1@maquina.dominio.net`
  - Arranque: `erl -name no1`

Cookies

Servem para a autenticação

- Dois sistemas comunicam se tiverem os mesmos *cookies*

Cookies

Servem para a autenticação

- Dois sistemas comunicam se tiverem os mesmos *cookies*
- O que se entende por *cookie* neste caso?

Cookies

Servem para a autenticação

- Dois sistemas comunicam se tiverem os mesmos *cookies*
- O que se entende por *cookie* neste caso?
  - Um átomo colocado no ficheiro `.erlang.cookie` no directório `$HOME` do utilizador
  - O ficheiro deve estar acessível apenas para o seu dono
  - Podemos usar na linha de comandos o argumento `-setcookie Cookie` ou o mais perigoso `-setcookie nocookie`

Funções

- `erlang:get_cookie()` – devolve o cookie do nó onde estamos
- `set_cookie(Node,Cookie)` – define o cookie a usar na comunicação com um determinado nó
  - Se `Node` for o nó onde estamos, então estamos a definir o cookie que será usado na comunicação com todos os outros nós
- `node()` – devolve o nó onde estamos
- `node(Arg)` – devolve o nó onde se encontra `Arg` que pode ser um Pid, uma referência ou um port
- `is_alive()` – devolve `true` se estamos num nó, ou `false` se estamos numa instância isolada do Erlang
- `nodes()` – devolve uma lista dos nós visíveis aos quais este nó está ligado
- `nodes(Arg)` – `Arg` pode ser `hidden` ou `connected`

Nós escondidos

Qual a ideia?

- Os nós em Erlang são transitivos
- Se A está ligado a B e B está ligado a C então A está ligado a C
- Um nó que esteja ligado a outro está ligado a todos os nós
- E se quisermos observar um sistema sem o perturbar?

Nós escondidos

Qual a ideia?

- Os nós em Erlang são transitivos
- Se A está ligado a B e B está ligado a C então A está ligado a C
- Um nó que esteja ligado a outro está ligado a todos os nós
- E se quisermos observar um sistema sem o perturbar?
  - Usamos nós «escondidos»!
  - Argumento `-hidden` na linha de comandos

Monitorização

- `monitor_node(Node,true|false)` – uma mensagem `{nodedown,Node}` é recebida se o nó ficar inactivo
- Desta forma podemos monitorizar outros nós

Mudanças

Em isolamento	Num grupo de nós
`Pid ! Mensagem`	`Pid ! Mensagem`
`nome ! Mensagem`	`{nome,No} ! Mensagem`
`receive`	`receive`
`link(Pid)`	`link(Pid)`
`unlink(Pid)`	`unlink(Pid)`
`spawn(Fun)`	`spawn(No,Fun)`
`spawn(Modulo,Funcao,Arg)`	`spawn(No,Modulo,Funcao,Arg)`
`spawn_link(Fun)`	`spawn_link(No,Fun)`
`spawn_link(Modulo,Funcao,Arg)`	`spawn_link(No,Modulo,Funcao,Arg)`

Mais coisas

- Ler os manuais

Mais coisas

- Ler os manuais
- Nomes globais em vez de ser em cada nó $\Rightarrow$ módulo `global`
- Em Windows há o `erl` funciona na linha de comandos e o `werl` que funciona numa janela própria
- Não é possível registar localmente processos remotos