Informe del seminari: Groupy

Alejandro Carol
Héctor Mañosas
31 d'Octubre, 2013

1 Introducció

L'objectiu de la pràctica és implementar un sistema de servei de pertinença a un grup que proporciona multidifusió atòmica.
La pràctica es divideix en diferents parts.
- En la primera part, el sistema no té en consideració quan els nodes poden aparèixer i desaparèixer (*crash*).
- En la segona part, es té en consideració les fallades dels nodes. Construirem el nostre sistema amb tolerancia de fallides, gradualment.

En primer lloc, detectarem les fallides. En segon lloc, veurem que la implementació actual no contempla que es puguin perdre missatges pel camí. En tercer lloc, per solucionar aquest problema canviarem el *multicaster basic* per un *multicaster reliable*. Per últim, veiem que Erlang no garanteix que els missatges arribin al node i suposem que el detector de Erlang és perfecte.

2 Feina feta

Hem completat part del codi que se'ns proporciona del *leader* i del *slave* del mòdul *gms1* per a la primera versió del sistema.
Per a la primera part de la millora, hem creat un nou mòdul *gms2* que utilitzant el *monitor* s'encarrega de triar un nou líder quan aquest falla.
Per a la segona, hem modificat el procediment *bcast/3* i hem afegit crashes aleatoris als processos per comprovar si es manté la sincronització.
Per a la tercera part (gms3) hem afegit identificadors de missatges, per assegurar la sincronització entre processos encara que hi hagin crashes, reenviant l'últim missatge enviat pel líder a tots els processos (per si algun no l'havia rebut).

3 Experiments

Primera part (gms1)

i) Fes alguns experiments per veure que pots crear un grup, afegir alguns fills i mantenir l'estat coordinat. Pots utilitzar el següent codi per començar o aturar tot el sistema. Nota que estàs utilitzant el nom del modul (p.e gms1) com el paràmetre *Module* per començar el procediment.
La posició d' *Sleep* per un nombre de milisegons els *workers* haurien d'esperar fins que el següent missatge és enviat.

Per executar-ho fem: *groupy:start(gms1,1000).*
Podem observar que els cinc workers estan sincronitzats.

ii) Divideix el mòdul groupy i fes el necessari per activar cada worker en diferents màquines. Recorda com registrar els mots en nodes remots i com la màquina Erlang hauria de començar a córrer els programes distribuïts.
Creem els nodes amb la comanda
erl -name <node_name>@127.0.0.1 -setcookie secret
Iniciem el líder amb: *groupy2:start(gms1, 1000, a, "1", 1, leader).*
Iniciem cada slave amb *groupy2:start(gms1, 1000, <atom_id>, <id>, <rand>, {a, 'leader@127.0.0.1'}).*
Podem observar que els workers estan sincronitzats.

Segona part (gms2)

Detector de fallides

i) Fes alguns experiments per veure si els nodes poden mantenir el seu estat coordinat inclús si un node falla.
Executem groupy:start(gms2, 1000).
No observem que els workers perdin sincronització, ja que aturant el procés a mà és força improbable que el líder es quedi a mig enviar els missatges.

Missatges perduts

ii) Repeteix els experiments i observa si els workers han perdut la sincronització.
Si el líder falla a mig enviar missatges es perd la sincronització entre els workers.

Tercera part (gms3)

Multicast Segur

i)Repeteix els experiments i observa si ara els fills poden mantenir el seu estat coordinat inclús si els nodes fallen.
Executem *groupy:start(gms3, 1000).*
En tot moment es manté la sincronització entre processos, encara que falli un node la resta mantenen la sincronització.

ii) Intenta mantenir un grup continu afegint més nodes mentre els nodes existents moren.
Hem fet servir un mètode un tant “rudimentari” per afegir nous processos, que consisteix en executar la següent línia de codi, i requereix tenir en compte no utilitzar un àtom d’un procés viu i assignar un id apropiat (una millor manera de fer-ho seria tenir algun procés que mantingués l’estat i comprovés que no s’intentin assignar dos nodes al mateix àtom).
register(<atom_id>, worker:start(<id>, gms3, <rand>, <leader_id>, 1000)).

4 Preguntes obertes

Missatges perduts

Per què està això passant?
Passa perquè el procés cau a mig enviar els missatges i alguns treballadors l'han rebut i altres no.

Què podria sortir malament?

i) Com canviaries la implementació per controlar la possibilitat de pèrdua de missatges?
Podríem enviar un missatge ack per cada missatge que es rep i reenviar si no es rep l'ack abans d'un temps t.
ii) Com impactaria això al rendiment?
Hauríem de guardar una còpia de tots els missatges pels que encara no tenim confirmació i tindríem el doble de tràfic de missatges a la xarxa, el que incrementaria l'ús de memòria i augmentaria l'ús de CPU.
iii) Què passaria si sospitem erròniament que el líder ha fallat?
Que el líder quedaria exclòs del grup i no rebria missatges enviats per altres workers (tot i que els altres nodes que ja estiguessin al sistema en el moment en que se l'exclogués seguirien rebent missatges del líder).

5 Opinió personal

La documentació proporcionada está molt ben estructurada i s'entén molt bé quina és l'evolució del sistema que estem construint.
Creiem que una millora possible, seria proporcionar el codi en un zip per a que no haguem de copiar el codi del enunciat de la pràctica, que resulta una mica farragós.
Ens ha resultat molt interessant veure com amb el monitor podem saber l'estat d'un procés i com es pot recuperar.