SODX LAB1 - Preguntes

Ixent Cornella, Eric Gonzalez

INTRODUCCIÓ

El Laboratori 1 consisteix en implementar un sistema distribuït per parlar entre clients. Hi ha un servidor i múltiples clients al principi, i posteriorment, més d'un servidor i múltiples clients.

PREGUNTES UN SERVIDOR, MÚLTIPLES CLIENTS.

i) Does this solution scale when the number of users increases?

En primer lloc, l'escalabilitat, és la mesura de la capacitat d'un sistema per a incrementar o disminuir el seu rendiment i cost, en resposta als canvis en les sol·licituds de processament de l'aplicació i el sistema. Aleshores, segons aquesta definició podem afirmar que en aquest cas, la nostra aplicació no escala a mesura que el nombre d'usuaris incrementa (l'escalabilitat es inversament proporcional al nombre d'usuaris), degut a que no es un sistema distribuït, un únic servidor no serà capaç de processar tota la informació a mesura que es vagin afegint nous usuaris.

ii) What happens if the server fails?

Si el servidor falla, el que succeeix es que els paquets d'informació que l'usuari intenta enviar-li, no arribaran mai, i per tant l'usuari no rebrà cap resposta. En termes de la nostra aplicació, aparentment al client no li passarà res, però quan intentem enviar un missatge, no rebrem confirmació, ni tampoc veurem si entra o surt algun client. En definitiva, perdem la connexió al servidor.

iii) Are the messages from a single client guaranteed to be delivered to any other client in the order they were issued? (hint: search for the 'order of message reception' in Erlang FAQ)

Segons l'informació trobada a la documentació de Erlang, l'ordre de recepció dels missatges si que es garanteix pero només dins d'un procés. És a dir, que si un procés està actiu i enviem el missatge A i tot seguit el missatge B, es garanteix que si el missatge A, ha arribat, llavors el missatge B també ha arribat. Per altra banda si tenim tres processos diferents (P, S i Q) i P envia el missatge 'A' a 'Q' i déspres el missatge 'B' a 'S', no es garanteix que el missatge A arribi abans que el B.

iv) Are the messages sent concurrently by several clients guaranteed to be delivered to any other client in the order they were issued?

No, no hi ha cap garantia de que els missatges s'envien en ordre. De fet, si això fos un requisit, seria molt complicat d'implementar-ho en Erlang distribuït (font: erlang.org). Llavors si dos missatges s'envien consecutivament, no hi ha cap garantia de que arribin en aquest ordre, tot i que normalment, si tot va bé, és el resultat esperat.

v) Is it possible that a client receives a response to a message from another client before receiving the original message from a third client?

Sí, és totalment possible ja que, com hem dit abans, no hi ha garantia de que s'enviin en ordre. Un missatge pot arribar abans que un altre, i per tant, el client pot respondre abans al missatge que s'ha enviat més tard tot i no haver rebut encara el primer.

vi) If a user joins or leaves the chat while the server is broadcasting a message, will he/she receive that message?

Això depèn de la velocitat del servidor enregistrar la petició de sortida / entrada, si el servidor rep el *leave* abans del broadcast, llavors no el rebrà, però en cas contrari, el rebrà (assumint que no es perd el missatge). Pel que fa al cas de *join*, l'usuari rebrà el missatge, si el missatge de broadcast s'executa abans que el de *join*, llavors no el rebrà, però en cas contrari, llavors sí és possible.

PREGUNTES MÚLTIPLES SERVIDORS, MÚLTIPLES CLIENTS.

i) What happens if a server fails?

En aquest cas, els clients que estiguin connectats al servidor que ha fallat perdran la connexió amb aquest servidor i tots els clients conseqüentment. La novetat d'aquesta versió, és que tots els clients que estiguessin connectats a altres servidors podran continuar xatejant, ja que no hi ha dependència d'un sol servidor. Aquesta implementació és millor que l'anterior, però una millor implementació seria que al perdre un servidor, automàticament es redirígis als clients a un altre servidor i no perdessin la connexió amb els demés.

ii) Do your answers to previous questions iii, iv, and v still hold in this implementation?

Continua sent possible, ja que de nou, no tenim cap garantia de que els missatges s'enviin en l'ordre original, potser que un missatge tardi més tot i tenir més servidors, de fet, tenir més servidors és un incentiu per què passi més aquest error, més retràs als missatges si han d'anar de servidor a servidor i més possibles errors de coherència de temps.

iii) What happens if there are concurrent requests from servers to join or leave the system?

Si hi han varies sol·licituds concurrents de servidors per entrar o sortir del sistema, no podrem garantir que arribin en ordre, però si tot va bé, haurien d'arribar les dues. Per tant, no sabrem en quin ordre es registraran els servidors, però haurien de poder registrar-se sense problema..

iv) What are the advantages and disadvantages of this implementation regarding the previous one?

Com s'ha comentat al punt 1, el primer avantatge és que és més robust en quant a possibles errors d'un servidor, ja que si es perd la connexió amb un servidor, no necessàriament vol dir que tots els clients perdran la connexió, només ho faran els que estaven connectats a aquell servidor. D'altra banda, quan més servidors, més

exposats estem, i per tant, el sistema és més vulnerable. Un altre avantatge és que podem usar servidors a diferents localitzacions per disminuir la latencia, això seria un avantatge però en el nostre cas realment no ens és útil ja que la localització del servidor es normalment molt propera a la del client.