Servicio de gestión de vistas

Sistemas distribuidos - P4

Alberto Calvo Rubió, Óscar Baselga Lahoz
Universidad de Zaragoza

8 de diciembre de 2019

Índice

1. Introducción

En esta práctica se introduce la tolerancia a fallos para nodos distribuidos con estado. El objetivo de la práctica es construir un servicio de almacenamiento clave/valor tolerante a fallos utilizando replicación Primario/Copia en memoria RAM

El gestor de vistas no estará replicado para mayor sencillez, lo que implica que no se plantea una solución completa de tolerancia a fallos como la que obtiene mediante algoritmos de consenso como Raft, Paxos o Zab.

2. Servidor gestor de vistas

El gestor de vistas tiene como objetivo actuar como el coordinador en el sistema de réplicas Primario/Copia. Este trata de detectar fallos en las réplicas, asignar a cada una si se trata del primario, copia o servidor en espera, y añade dinámicamente otras nuevas al grupo. Para poder llevar a cabo estas acciones los servidores réplicas enviarán mensajes de tipo *:latido* al servidor gestor de vistas.

2.1. Vistas

El coordinador mantendrá dos conjuntos de miembros que pertenezcan al grupo en un momento dado llamadas vistas. Estas indican que nodo es el primario y cuál es copia. En esta ocasión solo habrá un servidor copia. Además, se guarda el número de vista correspondiente, que se irá actualizando conforme se creen nuevas vistas, permitiendo diferenciarlas unas de otras.

Implementacion en Elixir:

defstruct num_vista: 0, primario: :undefined, copia: :undefined

- **vista_tentativa**: vista que se intenta establecer, pero debe ser validada por el nodo primario.
- **vista_valida**: vista confirmada por el nodo primario. Esta es la que se envía a los clientes para que accedan al servidor primario. Una vista tentativa pasará a ser válida en el momento en el que el nodo primario confirme la vista (se habrá realizado la copia de seguridad al nodo copia).

2.2. Arquitectura y comportamiento

El comportamiento del servidor gestor de vistas se ha representado mediante un diagrama de estados. En el se diferencian los estados del sistema, si es consistente, si hay algún nodo caído y como puede recuperarse, o en el peor caso terminar con un fallo crítico. A continuación se explica su funcionamiento:

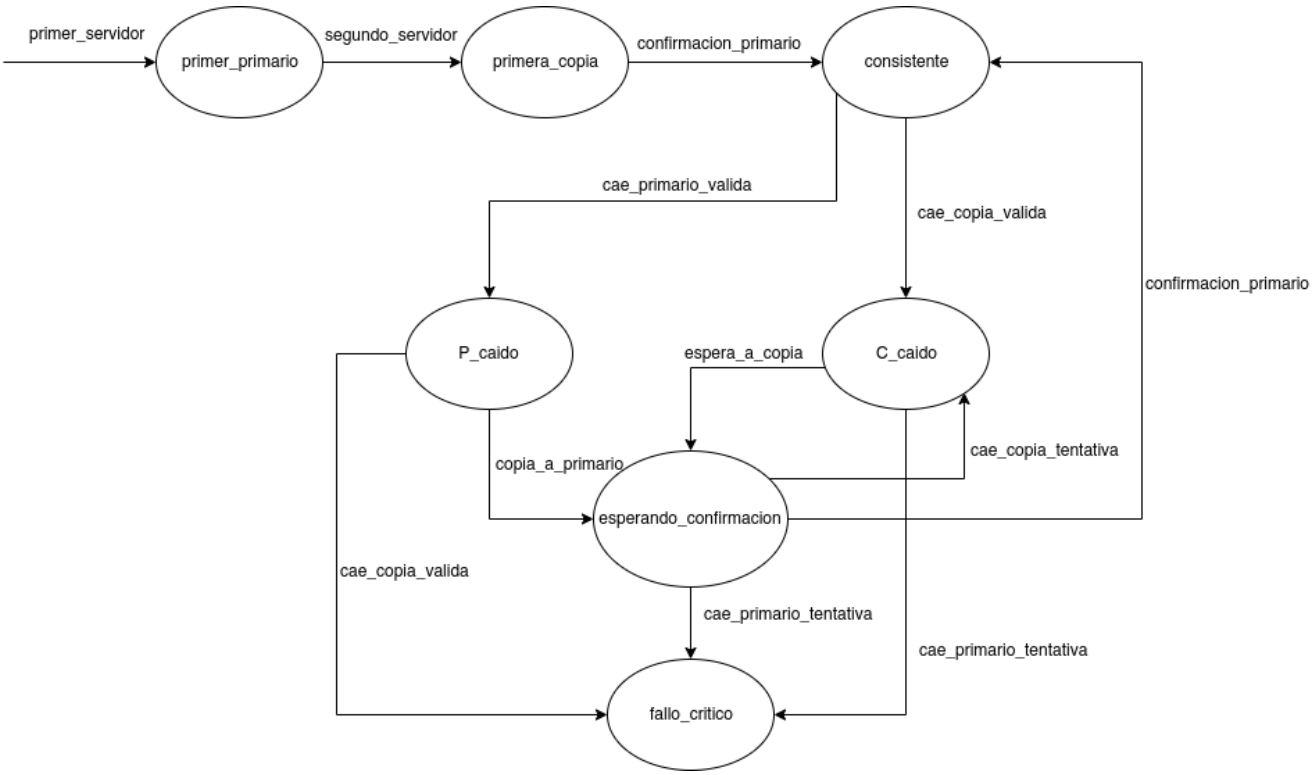

Figura 1: Diagrama estados gestor de vistas

- Los dos primeros estados corresponden al inicio del servidor gestor de vistas con la introducción del primer servidor primario y la primera copia. Una vez se han confirmado por el primario, se establece el sistema como consistente. A continuación, se puede producir tanto la caída del primario como de la copia de la vista_valida.
- En el caso de que se caiga uno de los dos servidores se pasará al estado de P_caido(primario caido) o C_caido(copia caído). Independientemente de cuál de los dos haya caído, si cae el otro servidor antes de cambiar de estado se pasa a estar en fallo_critico donde se detiene el funcionamiento.
- En el caso de que se haya caído el servidor primario y la copia no, se promociona a la copia como servidor primario y se espera en el estado de esperando_confirmacion a que el nuevo servidor primario confirme la nueva vista, significando que vuelve a haber primario y copia establecidos.
- Si es el servidor copia el que se ha caído, se intenta promocionar un nodo en espera a nuevo nodo copia y se repetiría el proceso de espera a la confirmación del primario. En caso de no existir nodos en espera, se continua en el mismo estado de C_caido.

- Finalmente, en el estado de esperando_confirmacion se espera a que el servidor primario valide la nueva vista. En este estado si cae el primario sin haber confirmado la vista_tentativa significa que se ha perdido todos los datos y por tanto se pasa al estado de fallo_critico y se para el sistema. Sin embargo, si solo cae la copia, se vuelve al estado de C_caido a la espera de promocionar otro servidor en espera.

2.3. Implementación Elixir

Principalmente la implementación en Elixir consiste en dos procesos. El principal es bucle_recepcion que recibe todos los mensajes del gestor de vistas y realiza las acciones correspondientes dependiendo del tipo de mensaje. El proceso secundario se trata de un bucle que envía mensajes al proceso principal cada @intervalo_latidos para que periódicamente vaya actualizando el estado del servidor y aumente el número de latidos que los servidores réplica

En el caso de recibir un mensaje de tipo *:latido*, se comprueba si se trata de una caída(número de latido 0) o de un latido común. Si se trata de una caída se procede a comprobar en que lugar se incorporará el servidor nuevo o en el caso de que la caída haya sido del primario o de la copia, se actualizará el estado del sistema según el diagrama de estados anterior.

Si se recibe el mensaje *:obten_vista_valida*, se devolverá la actual vista valida y si el sistema es consistente o no(vista valida igual a la vista tentativa)

Finalmente, el último tipo de mensaje :procesa_situacion_servidores, enviado por el proceso secundario nombrado anteriormente, indicará al proceso principal que debe aumentar el número de latidos de todos los servidores actuales y realizar los cambios correspondientes si se considera que alguno de ellos esta caído(si su contador de latidos es mayor que @latidos_fallidos)

3. Validación

Para comprobar el correcto funcionamiento del servidor gestor de vistas, se han utilizado los test proporcionados por el profesorado(test 1-7) y se han ampliado con los test 8 y 9 propios.

1. No debería haber primario (antes de tiempo).
2. Hay un primer primario correcto.
3. Hay un nodo copia.
4. Copia toma relevo si primario falla.
5. Servidor rearrancado se convierte en copia.
6. Servidor en espera se convierte en copia si primario falla.
7. Primario rearrancado es tratado como caido y convertido en nodo en espera.
8. Servidor de vistas espera a que primario confirme vista, pero este no lo hace.
9. Si anteriores servidores caen, un nuevo servidor sin inicializar no puede convertirse en primario.

Para realizar los test 8 y 9 se parte de los test anteriores, ejecutándose de forma secuencial. Tras caer el servidor c1 en el test 7, la nueva vista tentativa consiste en que el nuevo nodo primario sea c3 y la copia c1 tras rearrancar. Partiendo de este punto, en el test 8 se espera a que el servidor c3 confirme la vista, pero este cae y se debe comprobar que c1 no es promocionado, ya que c3 no había confirmado y por tanto, se han perdido los datos provocando un fallo crítico.

En el test 9 se parte del estado anterior y se intenta que un servidor nuevo sin inicializar se convierta en nodo primario, pero tras haber ocurrido un fallo crítico, no se permite.

Traza de los test con los avisos conforme se van perdiendo primario, copia o se pierde la consistencia del sistema y se produce un fallo crítico:

```
Tiempo puesta en marcha de nodos  : 2231
Test 1: No deberia haber primario ...
 ... Superado
.Test 2: Primer primario ...
 ... Superado
.Test 3: Primer nodo copia ...
 ... Superado
.Test4 : copia toma relevo si primario falla ...
WARNING: No hay copia
 ... Superado
.Test 5: Servidor rearrancado se convierte en copia ...
 ... Superado
.Test 6: Servidor en espera se convierte en copia ...
 ... Superado
.Test 7: Primario rearrancado tratado como caido ...
.Test 8: Servidor de vistas espera a que primario confirme ...
ERROR: Primario caido con vista_tentativa sin confirmar
END: Estado del sistema no consistente parada critica
 ... Superado
.Test 9: Sin inicializar no ...
 ... Superado
.Finalmente eliminamos nodos
```

Figura 2: Traza de los test