Task Scheduler

Carlos Silva A75107
Catarina Cardoso A75037
José Silva A74601

Universidade do Minho, Braga, Portugal

1 Introdução

Este projecto baseia-se na implementação em Java, usando o protocolo de comunicação em grupo Spread, de um serviço distribuído de escalonamento de tarefas que seja tolerante a faltas. Existem duas entidades principais: os clientes, que submetem e processam tarefas, e os servidores, que armazenam informações sobre as tarefas a executar e fazem a sua atribuição aos clientes.

O serviço deve assegurar a conclusão de todas as tarefas introduzidas (apesar da falha de f servidores e desde que exista pelo menos um cliente), a transferência de estado para novas réplicas e que a falha de um cliente não resulte na não conclusão das tarefas a este atribuídas.

2 Decisões

Para assegurar que o serviço é tolerante a faltas, optou-se por implementar o protocolo de replicação activa. Comparando os dois protocolos[1], a maior vantagem da replicação activa é a sua simplicidade de implementação (possibilidade de reutilização do mesmo código para todas as réplicas).

Para além da replicação passiva não ser transparente para o cliente em caso de falha da réplica primária, o processo de reconfiguração de uma nova primária exige um grande custo. Contrariamente, na alternativa activa, quando uma réplica falha, as restantes processam o pedido e respondem ao cliente, não transparecendo qualquer falha ao cliente.

A maior desvantagem da replicação activa é a obrigatoriedade do uso de um algoritmo determinístico. Como o algoritmo utilizado é executado de forma homogénea nas diferentes réplicas, tal limitação não se aplica nesta situação.

Para a comunicação em grupo, existem dois grupos: o grupo dos servidores, a que pertencem todos os servidores, e o grupo dos clientes, a que pertencem todos os servidores e todos os clientes. Desta forma, os servidores receberão *Membership Info* sobre todos os intervenientes do serviço, quer sejam servidores ou clientes.

3 Implementação

3.1 Tarefas

Uma tarefa é identificada por uma URL única e uma descrição textual. As tarefas são atribuídas de acordo com a ordem que são adicionadas, embora não se possa concluir sobre a ordem pela qual são concluídas.

3.2 Comunicação

Para a comunicação entre servidores e também entre clientes e servidores, foi utilizado o *Spread Toolkit* que fornece comunicação em grupo com *Extended Virtual Synchrony (EVS)* [2]. Com este *toolkit* é facilitado o desenvolvimento do protocolo de replicação, utilizando primitivas como *Total Order Multicast*. Fica também mais fácil o problema da eleição de um líder, que pode ser útil em diversas situações.

3.3 Clientes

Os clientes, ao conectarem-se, juntam-se ao grupo de clientes (a que pertencem clintes e servidos, para que os servidores recebam notificações relativas à saída de clientes). Quando um cliente pretende efectuar um pedido ao serviço, envia uma mensagem para o grupo de servidores. O serviço oferece três operações: introduzir nova tarefa, obter próxima tarefa e assinalar conclusão de uma tarefa.

Visto tratar-se de um serviço que usa replicação activa, todos os servidores disponíveis respondem ao cliente e este receberá várias respostas para o mesmo pedido. Como as mensagens estão identificadas com o id do pedido, o cliente saberá quando se trata de uma mensagem repetida e, consequentemente, ignorá-la-á.

3.4 Servidores

Os servidores, ao conectarem-se, juntam-se ao grupo de servidores e ao grupo de clientes. Na classe da lógica de negócio de cada servidor existem três variáveis principais: um contador (que se incluí no URL de cada nova tarefa, incrementado-o), um *set* com as tarefas introduzidas mas não atribuídas e um *map* que, para cada cliente, mapeia as suas tarefas atribuídas.

3.4.1 Falha do Cliente

De forma a que a falha de um cliente não resulte em tarefas inacabadas, as tarefas atribuídas a um cliente que falhe voltam a estar disponíveis e são atribuídas a outros clientes assim que possível. Para que estas tarefas tenham prioridade sobre outras ainda não atribuídas, a colecção que armazena estes dados é ordenada pela ordem que as tarefas são introduzidas no sistema.

Na eventualidade de um cliente falhar, os servidores, como partilham de um grupo de comunicação com todos os clientes, recebem *Membership Info* sobre a ligação e desconexão de um cliente. Visto que o grupo de comunicação pelo qual os servidores recebem os pedidos dos clientes é diferente do grupo pelo qual recebem as notificações de desconexão, não é assegurada ordem total entre pedidos e notificações.

Para prevenir erros associados à ordenação desigual destas mensagens nas diferentes réplicas, uma das réplicas é nomeada (*leader election*) para reenviar estes avisos, mas desta vez para o grupo de servidores. Desta forma é garantido que todas as réplicas processam a saída do cliente pela mesma ordem. Embora seja apenas um servidor a enviar estas mensagens para o grupo, todos guardam estas mensagens até receberem o aviso de cada uma delas dentro do grupo de servidores (momento em que as vão apagando). Isto porque, se o líder falhar, o novo líder sabe que mensagens deste tipo ainda faltam enviar para o grupo de servidores.

3.4.2 Transferência de Estado

Quando um novo servidor se conecta é iniciado o processo de transferência de estado. Todas as réplicas enviam o seu estado para a réplica que se iniciou. Para que os dados transferidos não sejam inconsistentes, as réplica que enviam o estado não devem processar pedidos de clientes, mantendo assim o mesmo estado durante o processo de transferência. Para que os clientes não recebam respostas diferentes/erradas, a réplica que recebe o estado também não deve processar novos pedidos. Assim, a nova réplica deve arquivar as mensagens oriundas dos clientes pela ordem de chegada, até que termine a transferência de estado (também guarda mensagens provenientes relativas a *membership* e novos servidores *up to date* (referidos mais à frente) também pela ordem de chegada). No final deve processar estas mensagens e chegará ao mesmo estado. As outras réplicas, enquanto estão a enviar o estado, ocupam a *thread* (do *SingleThreadContext*) e por isso não precisam de guardar os pedidos dos clientes, que serão executados posteriormente.

Também a nível da transferência de estado e se o estado for maior que 100KB (tamanho máximo para uma mensagem no Spread), os dados devem ser enviados em vários fragmentos. Para uma utilização controlada dos fragmentos, a descrição associada a cada tarefa não deve ser maior que 100KB. Foi feita uma aproximação e com essa aproximação não se ultrapassam nunca os 50KB para que qualquer erro na aproximação não cause problemas. Ainda assim, como 50KB ainda deixa uma margem grande, anexa-se sempre ao primeiro fragmento os membros da eleição de líder (abordado na próxima secção).

A lista de membros poderá sofrer possíveis alterações após a transferência de estado, quando se executam os pedidos e mensagens guardados. Tudo será executado tal como foi nos outros servidores, ficando com os mesmos membros e com o mesmo líder. Estas mensagens que não são pedidos dos clientes podem ser informações de *membership* e mensagens também relacionadas com *membership* com mais garantias. Por exemplo a mensagem que informa que um cliente saiu,

já reenviada pelo servidor líder para o grupo dos servidores com *Total Order Multicast*. Existe outra mensagem "*UpToDate*"que tem o objetivo de ajudar na eleição de líder e é explicada na próxima secção. Para efeitos de terminação da transferência, no último fragmento deve haver uma indicação da finalização do envio, de forma a não bloquear a réplica nova.

3.4.3 Eleição de Líder

Como referido anteriormente, a existência de um servidor líder é essencial para a assegurar a ordem total entre pedidos e notificações de desconexão de clientes, assim como a transferência de estado para uma réplica que se tenha conectado (não foi implementado, explicado no trabalho futuro). A nomeação é feita através de uma eleição em que se seleciona alfabeticamente o servidor com o nome mais baixo. A eleição é feita em duas situações distintas: quando um servidor se desconecta e quando um servidor se conecta (não é assim tão simples). Neste último caso, a eleição só é feita após a conclusão da transferência de estado, de modo a evitar que a réplica nova seja eleita enquanto incoerente com as restantes.

Para conseguir ter a eleição apenas quando a transferência de estado termina, existe uma lista dos membros em cada servidor. Quando um servidor recebe uma transferência de estado e fica *up to date*, envia um aviso para todos. Quando este aviso é recebido (incluindo por ele próprio), ele é adicionado à lista de membros e o líder é reeleito. No caso de um servidor falhar, remove-se da lista de membros e o líder é reeleito.

4 Conclusões e Trabalho Futuro

O desenvolvimento deste projecto iniciou-se com uma implementação base do protocolo de replicação activa, a par da construção da camada de negócio. A comunicação entre clientes e servidores foi feita utilizando protocolo de comunicação em grupo Spread, fazendo uso dos diferentes grupos de comunicação criados. Seguiu-se o desenvolvimento de afinações ao protocolo de forma a tolerar faltas de clientes e transferências de estado de tamanho muito elevado.

A nível do *back-end*, um optimização da transferência de estado das réplicas é necessária: ao invés de todas as réplicas enviarem o seu estado para a nova réplica, apenas uma realizaria essa transmissão. Desta forma, quando um novo servidor se conecta, a réplica líder envia o seu estado para a réplica que se iniciou, enquanto as restantes (se existirem) asseguram o serviço sem interrupções.

No que toca a *front-end* e do lado do cliente, o desenvolvimento de uma interface gráfica seria uma mais valia para todo o tipo de utilizadores, devido à sua facilidade de utilização.

Referências

1. Wiesmann, Matthias and Pedone, Fernando and Schiper, André and Kemme, Bettina e Alonso, Gustavo *Understanding Replication in Databases and Distributed Systems*. 2000.
2. Jonathan R. Stanton *A Users Guide to Spread Version 0.11*. http:www.spread.orgdocsguideusers_guide.pdf