INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Licenciatura/Mestrado de Engenharia Informática e Computadores

Unidade Curricular de Sistemas Distribuídos
(2º semestre letivo 2014/2015)

Apache ZooKeeper

Grupo Nº 2
Autores: 36907 Carlos Serra
38205 Adriano Sousa
38222 Marta Nascimento

Junho de 2015

Resumo

Um sistema distribuído apresenta diversos problemas por ser exatamente isso, distribuído. Ou seja, como é executado em vários computadores e é composto por vários componentes de *software* precisa de criar regras de cooperação para a atingir um objetivo comum. Por exemplo, a ocorrência de dois eventos no sistema podem ser conhecidos em sítios diferentes segundo uma ordem de ocorrência diferente. Isto pode acontecer, por exemplo, pela falta de sincronização dos relógios de cada máquina, atrasos na transmissão dos dados, etc.

Outro dos problemas é: pedidos simultâneos envolverem alterações de estado partilhado. Como um sistema distribuído partilha um estado global precisa de haver sincronização dos pedidos para que seja garantida a consistência dos dados.

Para resolver estes problemas foi desenvolvido o projeto Apache Zookeeper que entre outros, resolve também o problema de sincronização distribuída.

Índice

Desenvolvimento

Um sistema distribuído executa-se em vários computadores e é composto por vários componentes de *software.* Estes comunicam através de protocolos distribuídos para assistirem a execução coerente de atividades distribuídas e cujo resultado pode ser visto como se só existisse um único computador. Partilha ainda um estado global e coopera para a obtenção de um objetivo comum.

Estas características trazem algumas consequências como a concorrência, a não existência de um relógio global, falhas independentes, etc. Assim, e uma vez que os programas que precisam de cooperar entre si têm necessidade de coordenar as suas ações, surge a necessidade de criar mecanismos de sincronização distribuída.

Sincronização Distribuída

A sincronização distribuída resulta da necessidade de coordenação e sincronização da tomada de decisões num ambiente em que os múltiplos processos se executam em processadores diferentes e interligados por um sistema de comunicações que apresenta atrasos de transmissão não desprezáveis.

A coordenação entre processos distribuídos pode ser realizada com algoritmos de exclusão mútua distribuída, algoritmos de eleição, comunicação por grupos, comunicação *Multicast*, etc. Para a correta aplicação do método utilizado para realizar sincronização distribuída é necessário um conhecimento prévio de todos os tipos de algoritmos de sincronização existentes e saber aplicá-los da maneira mais correta possível. Isto faz com que possam ocorrer falhas de segurança, comunicação e até de sincronização.

Para resolver este problema e para que o utilizador não se tenha que preocupar com este tipo de problemas foi desenvolvido o ZooKepper.

ZooKeeper

ZooKeeper é um projeto de *software Open Source* da Apache Software Foundation que fornece um serviço centralizado para manter as informações de configuração, expondo serviços como a atribuição de nomes, proporciona sincronização distribuída, e fornece serviços do grupo.

Como o nome sugere, é responsável por manter organizado algo que pode ser extremamente desorganizado.

Todos esses tipos de serviços são utilizados de uma forma ou de outra por aplicações distribuídas.

A arquitetura do Zookeeper suporta alta disponibilidade através de serviços redundantes. Desta forma caso haja falha de resposta de um dos servidores, é possível que o mesmo pedido seja realizado a um servidor diferente.

Figura 1- Apache Zookeeper

Como funciona?

O Zookeeper é um serviço que mantém duplicada a sua informação por várias máquinas, desta forma caso um pedido seja feito a um servidor e este não responda, o mesmo pedido deve ser feito a outra máquina, ou seja caso algo aconteça com um dos servidores, o sistema mantem-se a funcionar com normalidade.

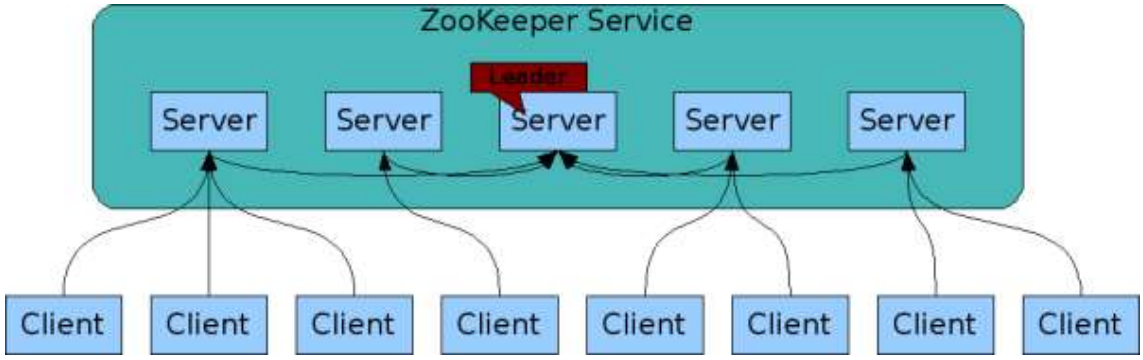

Figura 2- Representação de sistema Zookeeper

Para que a informação seja sempre coerente, é inicialmente eleito um servidor líder. A eleição é realizada por um simples processo onde cada servidor sugere um líder, quem obtiver mais votos fica então eleito.

Para que o processo de votação corra sem problemas, isto é acontecer empates, um sistema Zookeeper deve sempre arrancar com um número impar de máquinas, impossibilitando que existam dois servidores com o mesmo número de votos. Um sistema com três máquinas pode ser problemático pois caso exista falha numa das máquinas, existirá um número de servidores demasiado reduzido, como tal o número mínimo aconselhado é de cinco servidores.

A ideia geral é que todos os participantes de um processo eleitoral criam um nó efémero sequencial no mesmo caminho eleição. O nó com o menor número de sequência é o líder. Criou-se uma lista vinculada de nós, em que cada nó "seguidor" escuta o nó com o próximo número de sequência mais baixo, para minimizar a concorrência quando deixar de existir ligação com o servidor líder.

Quando o líder morre, os outros servidores entram em processo de eleição para encontrar o servidor com o menor número de sequência e tornar-se o líder.

Existem vários tipos de acções que se podem realizar sobre estes servidores, tendo o líder de saber fazer toda a gestão dos mesmos.

Caso um cliente faça um pedido de leitura, o servidor pode responder de imediato, pois sabe que a informação que este tem, também se encontra presente nos restantes. Por outro lado, operações que envolva alteração do estado do sistema têm de passar por um processo diferente, onde o servidor líder terá de fazer a gestão de forma a manter coerente toda a informação nos restantes servidores.

Modelo de Dados

O modelo de dados é feito por nós, designados de Znodes, onde é armazenada a informação. Cada Znode pode conter um ou mais nós, sendo esses nós representativos de um determinado caminho que pode ser absoluto, separado por barras ou relativo.

Cada nó também armazenada informação sobre qual o seu número de versão. Isto é útil para casos em que é necessário saber se o Znode foi alterado

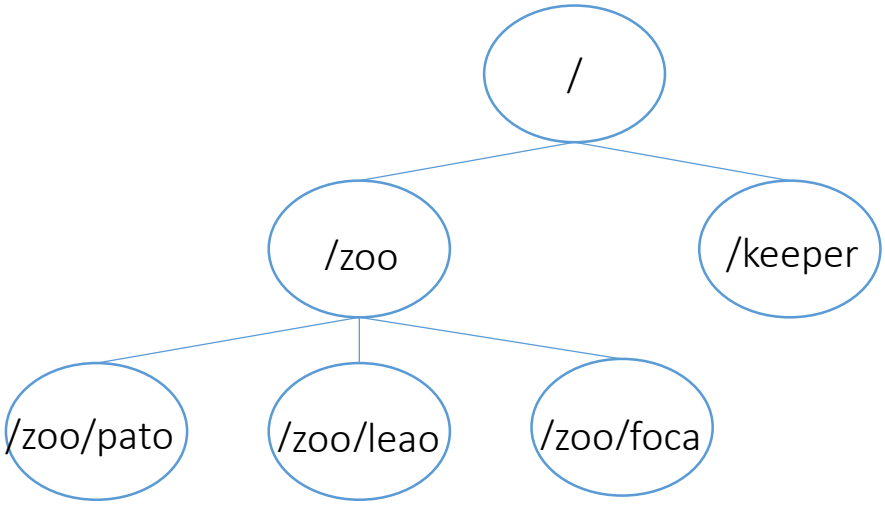

Figura 3-Representação em árvore de Znodes

Na Figura 3, podemos então ver a representação em árvore de Znodes com caminhos como:

- /
- /zoo
- /keeper
- /zoo/pato
- /zoo/leao
- /zoo/foca

Sendo que todos estes estão numa hierarquia e todos representam caminhos com possíveis acções.

Os ficheiros que estes representam não são ficheiros de dados, ou seja, são ficheiros que contêm informação útil de aplicação. Como tal são ficheiros que por norma pequenos, que nunca passam de 1 megabyte de informação.

Em resumo, um Znode é um sistema parecido ao *Filesystem*, mas que mistura o conceito de pastas e ficheiros, ou seja, um Znode é um ficheiro que pode conter informação (conceito de ficheiro do *Filesystem*) e também outros ficheiros (conceito de pasta do *Filesystem*).

Tipos de Znodes

Existem três tipos de Znodes, persistentes, efémero e sequencial.

Os nós persistentes são nós que existirão até que sejam explicitamente eliminados.

Os nós efémeros apenas irão existir durante o tempo que a sessão que o criou estiver ativa, sendo eliminado no fim desta. Este tipo de nós não pode ter filhos para garantir uma melhor estabilidade do sistema.

Já os nós sequenciais são nós compostos de um nome único acrescido de um contador. Os nós persistentes e efémeros também podem ser sequenciais.

Watches (notificações)

O ZooKeeper disponibiliza um sistema de notificações, assim um cliente pode registar-se num determinado evento de um nó e quando esse evento ocorrer o cliente será notificado. Um nó notifica os clientes por ordem de registo, ou seja, o primeiro cliente a registar é o primeiro a ser notificado. Um registo permite ter acesso a uma e uma só notificação.

Os eventos em que é possível um cliente registar-se, num nó, são:

- Evento de criação;
- Evento de alteração de estado;
- Evento de eliminação;
- Evento sobre estado dos nós filhos.

É possível remover um registo no evento em qualquer altura, deixando assim o cliente de ser notificado.

API

Um dos objetivos ao desenhar o ZooKeeper foi fornecer uma interface de programação muito simples de utilizar. Resultado disso é que apenas permite as seguintes operações:

- Create – serve para criar um nó numa localização da árvore;
- Delete - serve para eliminar um nó;
- Exists – permite testar se um nó existe numa determinada localização;
- Get data – permite obter a informação de um nó;
- Set data – serve para escrever informação num nó;
- Get children - retorna uma lista dos nós filho presentes num nó;
- Sync – permite uma espera até a informação seja propagada;

A API disponibiliza versões síncronas e assíncronas para todas as operações.

Implementação

No coração do ZooKeeper está um sistema de mensagens atómica que mantém todos os servidores sincronizados. O ZooKeeper Atomic Broadcast (Zab) usa um esquema de identificação de alterações de estado que permite que um processo consiga identificar facilmente as alterações realizadas. Esta funcionalidade é a chave para uma recuperação eficiente. Esta implementação de Zab pode atingir dezenas de milhares de transmissões por segundo, o que é suficiente para sistemas exigentes. O Zab garante ordem de entrega das mensagens e garante que se um servidor recebeu uma mensagem os outros também a receberam.

ZooKeeper implementa um esquema de *backup* em que o líder executa operações de clientes e usa Zab para propagar as correspondentes alterações de estado para os seguidores de *backup*. Foi desenhado para aplicações que exigem um alto desempenho do serviço e, por consequência, um objectivo importante é a capacidade de ter várias operações pendentes.

Funcionamento do serviço

Os clientes conectam-se a um servidor para enviar pedidos. Os pedidos são atendidos a partir do servidor ao qual está ligado, que é uma réplica do servidor principal. Os pedidos que alteram o estado da base de dados, como escritas, são processados utilizando por um protocolo de acordo.

Como parte do protocolo, todas os pedidos de escrita dos clientes são encaminhados para um único servidor, chamado de líder. Os restantes servidores do Zookeeper, os seguidores, recebem as alterações realizadas através de mensagens enviadas pelo líder e entregam as respostas aos clientes.

O ZooKeeper consegue garantir que as réplicas locais são exatamente iguais. Quando o líder recebe um pedido de escrita, calcula qual é o estado do sistema e quando a escrita é para ser aplicada e transforma a escrita numa transação que gera o novo estado.

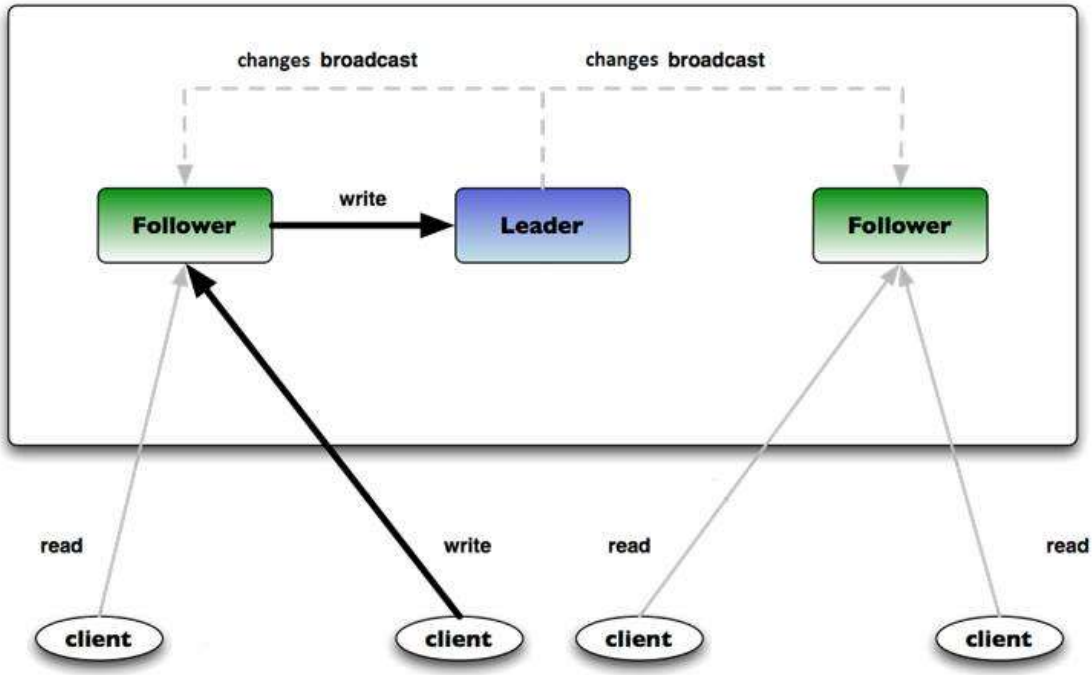

Figura 4- Percurso dos pedidos

Componentes do ZooKeeper

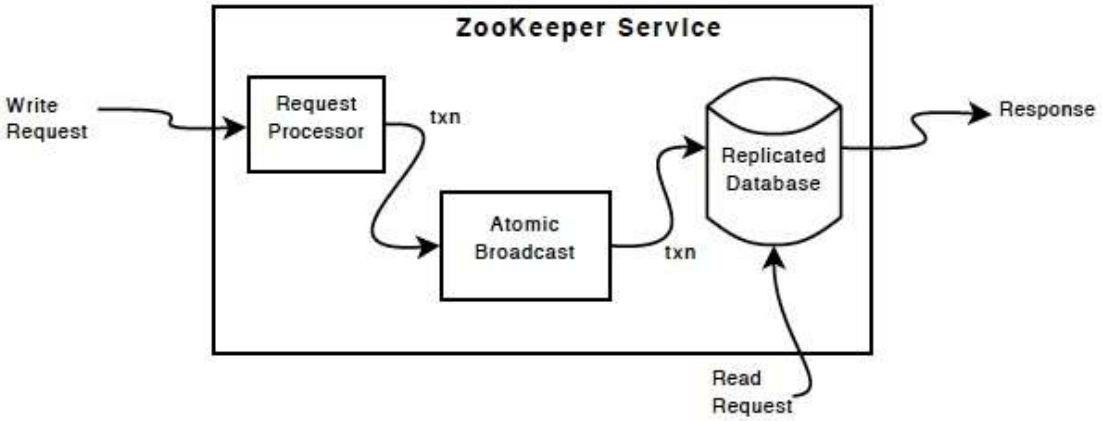

Figura 5- Componentes do ZooKeeper

Quando é feito um pedido que envolva alteração do sistema (um pedido que envolva uma escrita) este será posto numa fila FIFO (*First In First Out*) para garantir a ordem. Quando o pedido for executado será envolvido numa transação para garantir atomicidade. Este processo é representado pelo "*Request Processor*" da figura anterior.

É utilizado o Zab para informar todos os servidores seguidores que existiram alterações que recebem um conjunto de instruções para permitirem a atualização da sua imagem do sistema, representado pelo "*Atomic Broadcast"* da figura anterior.

Por fim, a "*Replicated Database*", da figura anterior, representa um servidor seguidor, que contém toda a árvore de dados. Existe um registo das atualizações que armazenado em disco para recuperação, e as escritas são serializadas e guardadas no registo das atualizações aplicadas na base de dados em memória. Para terminar o processo, o servidor seguidor que recebeu o pedido, gera uma resposta e entrega-a ao cliente que realizou o pedido.

Garantias

- Consistência sequencial - Atualizações a partir de um cliente serão aplicadas na ordem em que elas foram enviadas;
- Atomicidade - Atualizações realizadas até ao fim sem resultados parciais;
- Imagem única do Sistema - Um cliente vê o mesmo ponto de vista do serviço independentemente do servidor que se conecta;
- Fiabilidade - Após uma atualização ser aplicada irá persistir a partir daquele momento;
- Pontualidade – Todos os clientes têm a visão do sistema, atualizada, dentro de um determinado espaço temporal.

Utilização

A interface de programação do ZooKeeper é muito simples, no entanto, permite implementar aplicações de complexidade elevada sem ser necessário ter grandes cuidados com o problema da sincronização distribuída.

Por exemplo a Yahoo utiliza um serviço designado de MessageBroker que usa o Zookeeper. O *Message Broker* é um serviço de recuperação de coordenação de falhas.

Performance

ZooKeeper foi desenhado para ser altamente eficiente e os resultados da equipa de desenvolvimento do ZooKeeper indicam que é especialmente eficiente em aplicações onde o número de leituras superam as de escritas, uma vez que as escritas envolvem a sincronização do estado de todos os servidores como podemos ver na figura seguinte:

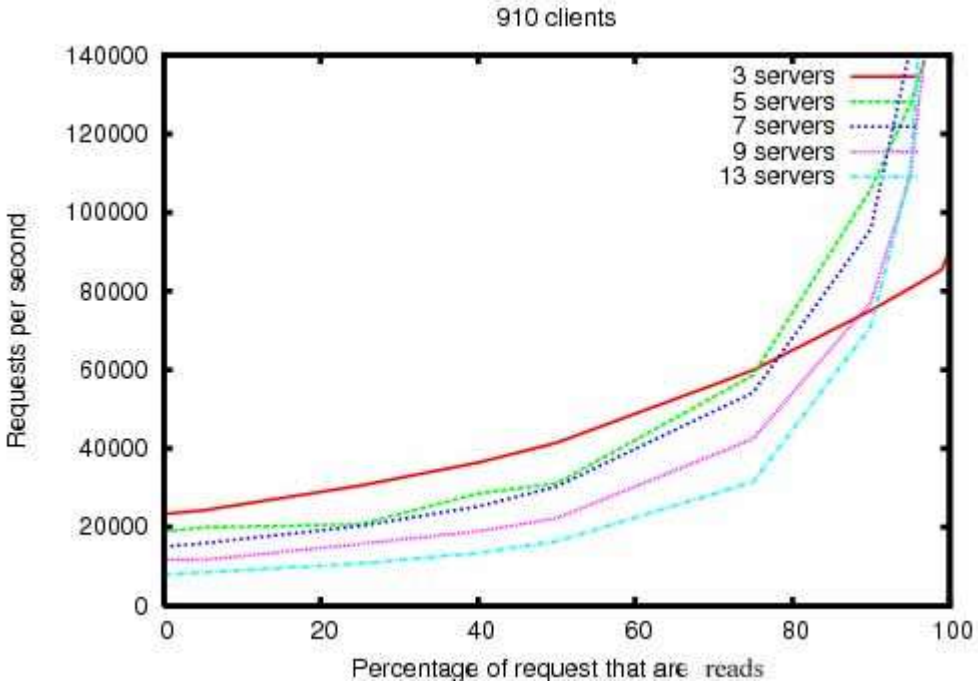

Figura 6- Resultado dos testes feitos ao ZooKeeper

Conclusão

Para tentar diminuir os problemas associados a um sistema distribuído, nomeadamente a sincronização deste, foi desenvolvido o projeto, pela Apache Software Foundation.

O ZooKeeper é um serviço distribuído centralizado de manutenção de informação de configuração. Ao proporcionar também sincronização distribuída, pretende resolver a maioria dos problemas associados a um sistema distribuído. A utilização deste serviço visa abstrair o utilizador dos problemas de sincronização de um sistema distribuído. Como existem diversas soluções que resolvem este problema, o ZooKeeper implementa um serviço que garante esta sincronização.

Assim, a implementação de um sistema distribuído torna-se mais simples, uma vez que quem o vai implementar não necessita de lidar com todos os problemas associados ao mesmo pois estas responsabilidades são atribuídas ao ZooKeeper.

Referências

https://www.youtube.com/watch?v=EPvFxuGrhtw, consultado em 12-06-2015

https://zookeeper.apache.org/, consultado em 12-06-2015

http://zookeeper.apache.org/doc/trunk/, consultado em 15-06-2015

https://en.wikipedia.org/wiki/Apache_ZooKeeper, consultado em 15-06-2015

http://br.hortonworks.com/hadoop/zookeeper/, consultado em 14-06-2015

http://stackoverflow.com/questions/3662995/explaining-apache-zookeeper, consultado em 14-06-2015