INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores

Licenciatura de Engenharia Informática e de Computadores

Unidade Curricular de Sistemas Distribuídos
2º semestre letivo 2017/2018

Modelo de Entrega de Dados para Sistemas Publish-Subscribe Baseado em Fog/Cloud

Grupo G5
Autores: 42181 Nuno Veloso, 42798 Steven Brito, 42799 Daniela Gomes

Introdução

Publish-Subscribe descreve um padrão utilizado em várias aplicações onde *subscribers* indicam o seu interesse num determinado tópico usando palavras chaves relacionadas e são notificados quando alguém publica novos dados sobre o referido tópico. Porém, o desenvolvimento de sistemas para suportar aplicações distribuídas usando este modelo em larga escala é um problema recorrente.

As arquiteturas frequentemente utilizadas para a resolução deste problema são *Peer to peer* (P2P) [1] e *Broker Overlay*. Na arquitetura P2P, um *subscriber* pode ser também um *publisher* e vice-versa. Isto permite que o sistema seja capaz de suportar uma elevada quantia de tópicos e conteúdos. Contudo, não é fiável devido à desistência de muitos participantes, podendo tornar o sistema menos eficiente.

Na arquitetura *Broker Overlay,* vários *brokers publisher/subscriber (pub/sub)* estão organizados numa *Broker Overlay Network* em que os participantes conectam-se a alguns brokers para publicar ou subscrever um tópico desejado. Embora esta arquitetura garanta tolerância a falhas e soluções economicamente viáveis não permite a adição dinâmica de brokers, tornando difícil a escalabilidade.

É com este propósito que o artigo *"A Fog/Cloud Based Data Delivery Model for Publish-Subscriber Systems"* [2] propõe uma solução que consiste numa hierarquia de *brokers* e participantes para a entrega dos eventos. Para a coordenação entre as diversas componentes, usam um conjunto de servidores *Zookeeper* [3] de forma a aprimorar a escalabilidade e o desempenho destes sistemas.

Maio de 2018

Síntese

Modelo

O modelo da solução proposta pelo artigo consiste em três componentes principais: servidores coordenadores, brokers *pub/sub* e os participantes, podendo ser subscritores ou publicadores.

Os servidores coordenadores consistem em servidores *ZooKeeper* (ZK) para coordenar os *brokers pub/sub* e os participantes. Escolheram esta tecnologia por ser capaz de eleger líderes entre grupos de *brokers*, detetar falhas, ter a associação de grupos e permitir a gerência de configurações nos sistemas distribuídos. Existem também servidores de processamento que realizam tarefas como mapeamento de participantes, deteção de novos tópicos, atribuição de *brokers* aos participantes designados, entre vários outros.

Os grupos de *brokers* são lançados em localizações geográficas de modo estratégico com o intuito de reduzir a latência entre o envio das mensagens. Estes conectam-se aos servidores ZK para coordenarem-se entre si dinamicamente e tornar a coordenação mais eficaz num ambiente distribuído.

O servidor ZK usa uma hierarquia de nós chamada *ZNodes*. Cada *ZNode* pode armazenar uma certa quantia de dados, mantém versões e *timestamps* para ler e alterar atomicamente os dados contidos.

Funcionamento

A Figura 1 descreve a hierarquia *znode* do modelo proposto no artigo [2]. Em seguida, apresenta-se em detalhe o funcionamento do sistema.

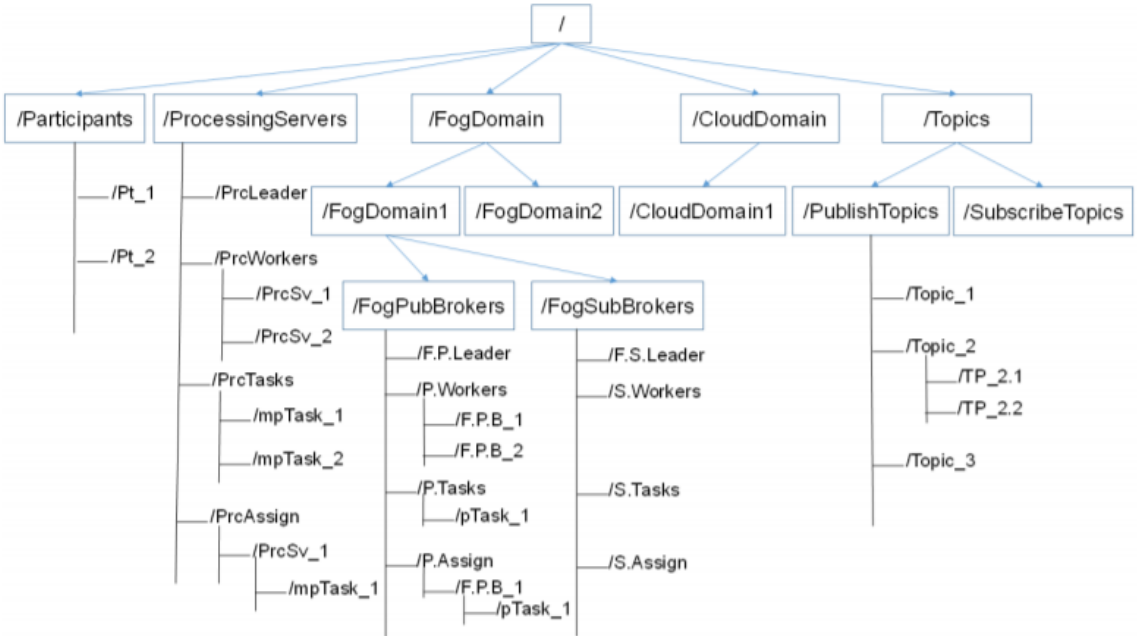

Figura 1- Hierarquia Znode do modelo

- Conectar participantes ao sistema

Um participante começa por conectar-se a um servidor ZK, criando um *znode* que representa o pedido de mapeamento, contendo o seu endereço IP e o porto. O líder dos servidores de processamento atribui a tarefa de mapeamento a um *worker* com menos carga de trabalho. O *worker* selecionado procura pela localização do participante e utiliza técnicas indicadas no artigo [4, 5, 6] para determinar os grupos de *brokers* (*fog/cloud*) mais próximos do participante. O participante ao estar inserido dentro do contexto de um grupo cria um *znode* para

representar o pedido de correspondência com brokers Pub/Sub apropriados, contendo os tópicos que deseja subscrever/publicar. O líder desse grupo atribui ao participante um *broker pub/sub* com menos carga de trabalho, baseado na correspondência das subscrições. O líder cria também um *znode* para notificar o participante. Este contem a lista dos tópicos e a localização dos *brokers pub/sub* atribuídos.

- Conectar um broker *pub/sub* ao serviço de coordenação

Um *fog broker pub/sub* conecta-se ao servidor ZK como cliente do serviço ZK, criando um *znode* para si próprio e elegendo um líder caso seja necessário. Este líder fica à escuta no *znode* das tarefas e no *znode* dos *workers* para monitorar os seus *workers*. Cada *broker pub/sub* cria um *znode* para receber atribuições e ficar à escuta de novas tarefas do líder. Para um *fog broker pub/sub* conectar-se a um *cloud broker pub/sub* é criado um *znode* para mapear aos *brokers pub/sub* adequados, contendo a lista dos tópicos *pub/sub* e o caminho do *fog broker*. O líder do grupo dos *cloud brokers* escolhe os *brokers* apropriados e cria um *znode* para notificar o cliente, contendo uma lista de tópicos e os *brokers* atribuídos. É também criado um *znode* para notificar os *brokers* atribuídos.

- Entrega de mensagens no sistema

Os *brokers pub/sub* têm a sua lista dos tópicos que subscreveram e dos que publicaram. Estes fornecem essa informação aos servidores ZK criando um *znode* que contém informação do *broker* (nome, endereço IP e porto) e do seu respetivo *cloud broker* responsável. O servidor ZK contem um *znode* "*NeedNotifyTopics*" para armazenar os *brokers* subscritores para os quais não existem *brokers* publicadores. O servidor ZK irá informar os subscritores quando um publicador desses tópicos existir.

Os participantes que subscreverem um tópico irão ser notificados por um broker subscritor assim que um broker publicador publicar esse tópico.

1) Envio de mensagens publicadores

Quando um broker publicador recebe uma mensagem verifica na sua lista local se o tópico é novo ou não. Se o tópico existir o broker envia a mensagem aos brokers registados. Caso contrário, envia para o servidor ZK a informação do tópico. Se o ZK encontrar o tópico na sua lista de "tópicos publicados", é atualizado os dados. Caso o ZK não encontre o tópico, o *broker* irá adicionar a informação do tópico ao *znode* de "tópicos publicados" e notifica o ZK para procurar no *znode* "*NeedNotifyTopics"* se existe algum *subscriber broker* interessado nesse tópico.

2) Envio de mensagens subscritoras

Quando um broker subscritor recebe uma mensagem verifica na sua lista local se o tópico é novo ou não. Se o tópico existir o broker adiciona o cliente à lista dos registados. Caso contrário, envia para o servidor ZK a informação do tópico. Se o ZK encontrar o tópico na sua lista de "tópicos publicados", o ZK fornece todos os *brokers* publicadores para o *broker* subscritor. O ZK adiciona informação do broker subscritor à sua lista de "tópicos subscritos". Caso o ZK não encontre o tópico, o ZK irá adicionar um novo *znode* à sua lista *"NeedNotifyTopics"*.

Conclusão

Devido ao facto de o artigo ser meramente teórico e não ter uma implementação concreta, ainda não é possível indicar se as ideias propostas constituem uma melhor solução do que as já existentes. Contudo, há vários pontos do artigo em que podiam detalhar mais profundamente as escolhas realizadas. A solução também não torna claro quais as decisões tomadas para garantir algumas das características de um sistema distribuído, sendo como exemplo a tolerância de falhas que é pouco mencionada. A expansibilidade é garantida, pois a solução consiste em diversos grupos de *brokers*, permitindo adicionar ou retirar dinamicamente *brokers* em diferentes regiões consoante o número de participantes e a respetiva carga de trabalho.

Os autores deste artigo pretendem implementar uma prova de conceito para o modelo proposto no futuro, logo resultados mais concretos serão obtidos posteriormente.

Referências

[1] P. Christensson, "P2P Definition," techterms, 2006. [Online]. Available: https://techterms.com/definition/p2p. [Acedido em 12 Maio 2018].

[2] V.-N. Pham e E.-N. Huh, "A Fog/Cloud Based Data Delivery Model for Pusblish-Subscribe Systems," 2017.

[3] Apache, "Apache ZooKeeper," Apache, [Online]. Available: https://zookeeper.apache.org/. [Acedido em 12 Maio 2018].

[4] S. Jafari e H. Naji, *GeoIP clustering: Solving replica server,* 2013.

[5] E. Katz-Bassett, J. John, A. Krishnamurthy, D. Wetherall, T. Anderson e Y. Chawathe, *Towards IP geolocation using delay and,* 2006.

[6] B. Eriksson, R. Nowak, P. Barford e B. Maggs, *framework for lightweight ip geolocation,* 2011.

[7] A. Mehra, "DZone," 6 Setembro 2017. [Online]. Available: https://dzone.com/articles/understanding-the-cap-theorem. [Acedido em 12 Maio 2018].