Simulação de falhas de um sistema de descoberta de serviços

Rubens Plentz Gonçalves[1], **Rafael Augusto Rost**[1]

[1]Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Faculdade de Informática (FACIN)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Porto Alegre, RS - Brasil

{rubenspg, rafarost}@gmail.com

Abstract. *This article presents a study of the Service Discovery system's behavior in failure scenario. The experiment has used the tool Consul [HashiCorp 2016] and its behavior was simulated using the Core Network simulator [USNavalResearch 2016]. The article describes the use case of Consul in a simulated environment, presenting its behavior of fault tolerance when server or slaves fail during the execution.*

Resumo. *Este artigo apresenta um estudo do comportamento de um sistema Service Discovery em caso de falhas. Foi realizado experimentos utilizando a ferramenta Consul [HashiCorp 2016] e seu funcionamento para o estudo foi simulado atráves do simulador Core Network [USNavalResearch 2016]. Posteriormente o artigo apresenta um estudo de caso com o uso deste serviço em dois cenários, mostrando o comportamento em caso de falhas do servidor e dos nós que compõe o serviço.*

1. Introdução

Sistema de descoberta de serviços é um padrão arquitetural e um componente chave dos sistemas distribuídos e arquitetura orientada a serviços. O paradigma da Arquitetura Orientada a Serviços (SOA) é um modelo que propõe a construção de sistemas modulares permitindo a reutilização de sistemas de terceiros em grande escala. Com isso facilita a padronização e reutilização, modificando a forma de desenvolvimento, implantação e arquitetura dos sistemas.

2. Consul

Consul é um sistema de configuração e descoberta de serviços distribuídos e com alta disponibilidade. Permite que o usuário apresente novos serviços e nodos de maneira flexível e sua poderosa interface oferece aos clientes uma visão atualizada da infraestrutura que abrange. [HashiCorp 2016]

Consul provê diferentes características que são utilizadas para prover informações consistentes sobre a infraestrutura de quem o utiliza. Essas características incluem mecanismos de descoberta de serviços (microserviços), sistema de *tags*, monitoramento de serviços, múltiplos *datacenters* e sistema de armazenamento de dados baseado em chave/valor.

2.1. Arquitetura e componentes

Consul é um sistema complexo que possui diferentes componentes que juntos formam a solução. Sua arquitetura é simples e composta pelas seguintes partes:

- Agente - O agente do Consul é um *daemon* que roda nos nodos de cada um dos membros do cluster. Ele é requerido para ser um membro do cluster, pois são os agentes que mantém a comunicação essencial para o funcionamento do cluster e seus serviços. Ele pode ser executado em modo cliente ou servidor. Se for cliente, este será a um direcionador de RPCs (acrônimo de Remote Procedure Call, em português Chamada de Procedimento Remoto) para o servidor. Este cliente não guarda estado e basicamente sua única atividade é fazer o encaminhamento das chamadas e se manter ativo no *LAN gossip pool*. Não é esperado alto consumo de recursos durante seu funcionamento. O agente também pode ser um servidor. Neste caso, sua função é diferente do cliente. Ele deve manter o estado do cluster, responder às chamadas *RPCs*, trocar *WAN gossip* com outros *datacenters* e encaminhar chamadas e fazer a comunicação com outros *datacenters*.

A figura a seguir mostra a arquitetura em alto nível do Consul 2.

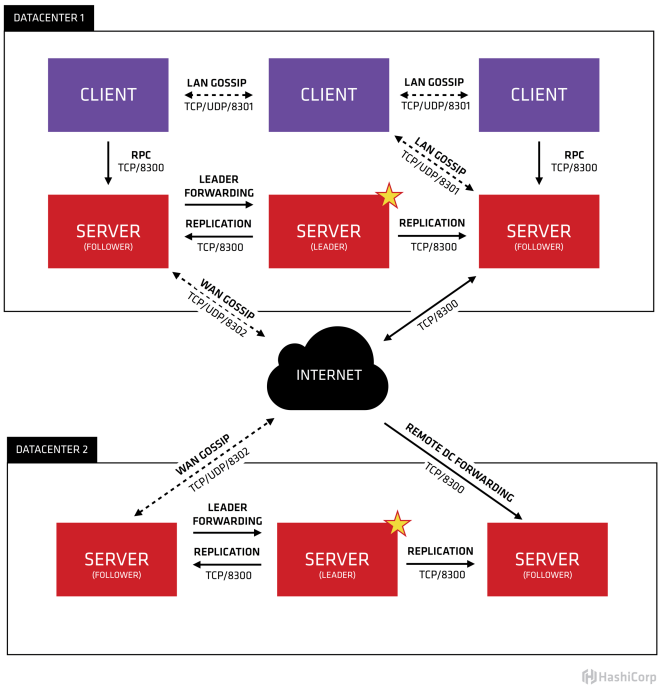

Figura 1. Arquitetura do Consul

3. Protocolo Gossip

O Consul utiliza em sua base o protocolo gossip [Wikipedia 2016] (traduzindo para o Português significa *fofoca*). Esse nome se dá devido ao seu simples funcionamento de troca de mensagens entre si com informações de outros nodos. É um protocolo simples, leve (em relação a utilização de recursos) e rápido.

O protocolo gossip foi criado com base no protocolo SWIM [Abhinandan Das 2012], com algumas mudanças e adaptações na velocidade de propagação e taxa de convergência.

Seu funcionamento se baseia na seguinte lógica. Quando um novo nó é criado em um *cluster*, este passa seu endereço a outro nó qualquer do *cluster*. O outro nó por sua vez, deverá validar se o nó que requisitou entrada é válido, através do protocolo TCP, e então propagar a informação para outros nós do *cluster*.

Gossip utiliza o protocolo UDP com porta e intervalos fixos. Isto assegura que o uso de rede será constante independente do número de nós. A troca de mensagens para averiguação do estado dos nós entre eles é feita periodicamente através do protocolo TCP. Porém a troca de mensagens gossip é feita com muito mais frequência. Essa frequência de trocas de mensagens para verificação do estado pode ser configurável aumentando sua frequência ou até desabilitando-a.

A detecção de falhas é feita através da sondagem aleatória entre os nodos em um certo intervalo. Se o nó não responder em um certo período de tempo, em seguida, uma verificação indireta é tentada. Esta verificação indireta pede para um número de nós aleatórios verificarem se aquele nó que não respondeu (no caso de haver problemas de rede que causam o próprio nó falhar essa verificação). Se tanto a própria verificação quanto a verificação dos outros nós falharem, o nó é marcado como ”suspeito”e essa informação é propagada para todos os nós do *cluster*. Um nó suspeito ainda é considerado um membro de *cluster*. Se o membro suspeito não contesta a suspeita dentro de um período de tempo, o nó é finalmente considerado morto, e esta informação com o novo estado do nó é novamente repassado entre os membros do *cluster*.

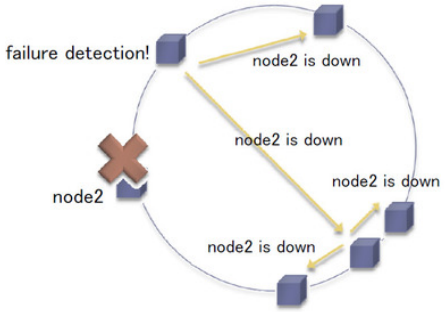

Figura 2. Detecção de falhas no protocolo gossip

4. Experimento

Para averiguarmos o comportamento da ferramenta Consul em casos de falhas, realizamos cenários de testes distintos capturando dados e analisando o comportamento nas diferentes execuções. A topologia de rede e a criação de serviços foram simuladas utilizando o simulador CORE (acrônimo de *Common Open Research Emulator*), o que facilitou a execução dos serviços utilizando Consul, bem como testes de escala, testes de falhas e testes com variações de rede. A ferramenta CORE utiliza em sua base Linux Containers (LXC) [Cannonical 2016], combinando *namespaces* com *Linux Ethernet bridging* para criar as redes que são utilizadas por cada componente do experimento.

O experimento proposto busca simular a execução de um sistema web real. Para isso, foi desenhado dentro da ferramenta CORE um ”datacenter”que expõe um endereço IP chamado por um agente externo.

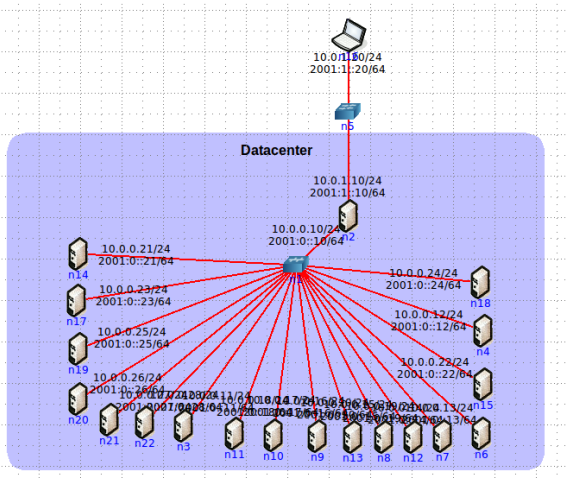

Figura 3. Experimento proposto

O *"datacenter"* possui dezenas de servidores que formam o *cluster* de Consul. Dezoito membros são servidores web que rodam NGINX e retornam uma página HTML quando requisitados. Estes estão ligados a um *switch* com um servidor sendo *Load Balancer* (e também roda NGINX), criando uma porta de entrada para as requisições que o agente fora deste *datacenter* virtual realiza.

4.1. Configurações

Foram criados serviços para o CORE, possibilitando a criação de hosts e configurando-o para executar determinados serviços. A ferramenta CORE por padrão oferece alguns serviços como Docker, SSH, FTP, Firewall, etc. Foram criados quatro novos serviços para o experimento:

- ConsulServer - é o serviço do Consul que roda em modo servidor.
- ConsulClient - é o serviço do Consul que roda em modo cliente.
- NginxWSService - é o serviço do NGINX que roda em modo Web Server. Ele possui uma página HTML padrão que é customizada no momento em que o serviço sobe, desta maneira cada serviço retorna uma página HTML diferente, facilitando a diferenciação dos servidores web na execução dos testes.
- NginxLBService - é o serviço do NGINX que roda em modo Load Balancer. As verificações de status dos servidores foram desabilitadas, deixando esta função para o Consul. Assim, o NGINX é atualizado constantemente pelo Consul através da sua extensão Consul Template a lista de servidores web que estão disponíveis. Desta forma, o NGINX apenas redireciona as chamadas para os servidores que estão presentes nesta lista.

O Consul é distribuído como um arquivo binário e pode ser encontrado na página oficial da ferramenta (www.consul.io). O NGINX pode ser instalado via gerenciador de pacotes na maioria das distribuições Linux, ou obtido através do site oficial (nginx.org).

4.1.1. ConsulServer

Após a instalação do Consul, deve-se criar um arquivo de configuração que é o descritor do serviço. Este é um arquivo JSON que contém algumas informações importantes como

nome do serviço, *tags*, porta e um comando para *health check*, bem como seu intervalo.

O conteúdo do arquivo é algo semelhante a:

```
{"service": {"name": "web", "tags": ["nginx"], "port": 80,
"check": {"script": "curl localhost >/dev/null 2>&1", "interval": "1s"}
```

Para iniciar o serviço em modo agente é necessário rodar o seguinte comando:

```
consul agent -server -bootstrap-expect 1 -data-dir /tmp/consul -bind=10
```

-data-dir: Local onde o Consul vai persistir informações. -config-dir: Indica o local e nome do arquivo JSON criado no passo anterior. -bind: O endereço IP local (geralmente ele pega automaticamente, porém usando com o simulador CORE tivemos que passar manualmente pois em alguns experimentos o endereço IP não estava sendo atribuído). -server: Indica que deve iniciar o agente em modo servidor. -bootstrap-expected: Indica quantos agentes em modo servidor é esperado no Consul cluster. Para o experimento foi utilizado apenas um.

4.1.2. ConsulClient

Uma vez configurado o servidor e o Consul, para iniciar outros nós como clientes, basta rodarmos o seguinte comando:

```
consul agent -data-dir /tmp/consul -bind=10.0.0.2 -config-dir /etc/cons
```

Desta forma este novo nó do *cluster* conectará com o servidor e outros clientes e o *cluster* estará formado. Para conferir se foi criado com sucesso, basta rodar em qualquer um dos agentes o seguinte comando:

```
consul members
```

Este comando retornará a lista de agentes do cluster.

4.1.3. NGINX

4.2. Resultados

4.2.1. Cenário 1: Verificação do tempo de atualização do template no *Load Balancer*

Para o experimento, foi utilizado o NGINX para distribuição de carga porém o serviço de *health check* foi desabilitado, pois o mesmo foi configurado no Consul, deixando o próprio Consul monitorar seu *cluster* e responsividade dos membros.

Foi configurado também o Consul Template, utilizado para deixar a lista de servidores responsivos sempre atualizada no NGINX. Com isso, o NGINX funcionou apenas como redirecionador das requisições que chegaram.

Quando realizado experimento com mais de mil requisições sequenciais ao servidor web, o NGINX respondeu rapidamente a essas requisições sem perda de performance. Também foi notado que ao parar o serviço do servidor web nos nós do *cluster* do Consul, a lista é atualizada no *load balancer* automaticamente, porém houve casos em que o *load balancer* acreditava que um nó ainda estava respondendo porém o serviço já havia sido

parado. Então ele redirecionou para o próximo nó da lista. Durante a primeira e segunda tentativa, foi notado que a lista então foi atualizada. Não se sabe precisar o tempo de atualização da lista de nós responsivos no *cluster*, porém é algo em torno de um segundo (para um experimento com 4 membros).

4.2.2. Cenário 2: Verificação do tempo de resposta em caso de falha de servidores web

Conforme descrito no cenário anterior, não houve perda significativa de desempenho após serviços serem parados durante a execução. Quando o *load balancer* não conseguiu encaminhar uma requisição a um dos membros, ele prontamente encaminha para o próximo da lista. A lista por sua vez, é constantemente atualizada pelo Consul, tornando o serviço confiável e seguro.

5. Conclusão

Neste artigo mostramos como configurar o Consul e como usá-lo com o emulador de redes CORE. Foram executados experimentos simulando falhas e monitorando os serviços, para auxilar nas conclusões sobre como é o comportamento da ferramenta Consul em cenários de falhas.

Além da ferramenta, o artigo aborda o funcionamento do protocolo Gossip. Foi analisado durante os testes o comportamento do protocolo, sua confiabilidade e rapidez na troca de informações entre os nós do *cluster*.

Com este trabalho concluímos que o Consul é um serviço fácil de configurar, leve e que fornece as funcionalidades necessárias para descoberta de serviços, *health check* e tolerância a falhas. Também notamos que ele se integra facilmente a outros softwares como NGINX, por exemplo, utilizado nos experimentos. Nos casos de falhas, não houve indisponibilidade do serviço web que estava sendo testado mostrando mais uma vez que ele pode ser utilizado em ambientes de alta performance com confiabilidade e alto desempenho.

Referências

Abhinandan Das, Indranil Gupta, A. M. (2012). Swim: Scalable weakly-consistent infection-style process group membership protocol.

Cannonical (2016). LXC Documentation @ONLINE https://linuxcontainers.org/.

HashiCorp (2016). Consul Documentation @ONLINE https://www.consul.io/docs/index.html.

USNavalResearch (2016). Core Documentation @ONLINE http://www.nrl.navy.mil/itd/ncs/products/core.

Wikipedia (2016). Gossip Documentation @ONLINE https://en.wikipedia.org/wiki/gossip$_p rotocol$.