# Replicação de servidores Memcached com Repcached

Arquitetura master/slave em servidores memcached com libevent

Fabiano Araujo
*Universidade La Salle*
*UNILASALLE*
Canoas, Brasil

***Abstract*—TODO**
***Index Terms*—TODO**

## I. Introdução

## II. Referencial Teórico

### A. Tolerância a falhas

### B. Servidores cache

### C. Arquitetura Master/Slave

## III. Implementação

O experimento realizado para demonstração da ferramente se dispôs entre dois *containers* Docker [1], obtidos de uma imagem com todos as bibliotecas necessárias para execução de dois servidores Repcached [2] isolados.

A replicação fica por parte da definição cíclica dos IPs de servidor *master* e *slave*. Por vezes a definição da dependência ficou confusa e foi obtido o entendimento de que o a definição real entre *master* e *slave*, nesta ocasião, refere-se especificamente à escolha do desenvolvedor ou cliente de utilização de um IP.

Utilizando imagens pré definidas dos *containers*, algumas modificações foram realizadas para que pudesse ser exibida de forma coerente a comunicação entre os servidores nos *containers*. Isto porque os exemplos encontrados e fundamentados nas próprias imagens utilizadas como exemplo utilizam do próprio *host* de um *container* Docker com o próprio *container*, não deixando clara as limitações da comunicação entre dois *containers*.

Da imagem *yrobla/docker-repcached* [3], foram retirados os arquivos que criavam a possibilidade de logins com administradores, por não ser relevante ao experimento e, principalmente, foi postergada a inicialização do *daemon* Memcached. Mesmo que o projeto possua um nome diferente, Repcached, o mesmo é uma adaptação do Memcached adicionando apenas um parâmetro para identificação do IP do servidor cujo terá replicação dos dados.

A postergação da inicialização se fez necessária pelo contexto de utilizar dois *containers* Dockers. Em sua natureza, cada *container* possui um IP local cujo, por padrão, é definido por um dispositivo virtual de rede. Este dispositivo funciona como um DHCP entre *containers* atribuindo os IPs assim que o *container* for iniciado.

No código inicial no entanto era necessário definir o IP do servidor *slave* antes da inicialização. Porém como a inicialização é o evento que é atribuído um IP na rede virtual interna de *containers* Docker, não era possível definir a atribuição, gerando uma falha no procedimento pois o mesmo adotava o IP 127.0.0.1 caso não fosse especificado.

Exposta a porta 11211, padrão do servidor memcached, então, é inicializado o *container* e acessado o mesmo utilizando as diretivas *exec -it –entrypoint /bin/bash* para que se tenha acesso à um *shell* onde é possível iniciar o servidor memcached.

Iniciando os dois *containers* foi então verificado os IPs internos definidos pelo comando *docker network inspect bridge*. Tomando nota dos IPs atribuídos, os serviços memcached foram iniciados em ambos os *containers* apontando o parâmetro *-x* para o IP do *container* oposto. Por isto a atribuição de *master* e *slave* se tornou cíclica.

Tal comportamento permite que caso o *slave* caia e retorne em um momento posterior, o mesmo possua todos os dados de *master* e vice-versa. Diferente de estruturas como Redis que possui *slaves* como apenas leitura e propagação direta para *master*, da forma como o Repmemcached é organizado ambos possuem a mesma atribuição

Omitir a definição do IP ou tornar a atribuição do IP pelo parâmetro *-x* inviabiliza o fluxo, sendo necessário reiniciar todo o processo em *master* caso o *slave* caia.

## IV. Resultados

## V. Conclusão

## Referências

[1] Docker. Docker. [Accessed: 15- Nov- 2018]. [Online]. Available: https://www.docker.com/
[2] K. Inc. Repcached. [Accessed: 15- Nov- 2018]. [Online]. Available: http://repcached.lab.klab.org/
[3] yrobla. Base docker image to run a repcached server. [Accessed: 10- Nov- 2018]. [Online]. Available: https://github.com/yrobla/docker-repcached