Algoritmos de consenso e sua aplicação em bancos chave/valor

Vitor Guidi

Proposta de Trabalho de Conclusão de Curso
apresentado à disciplina
MAC0499
(Trabalho de Formatura Supervisionado)

Orientador: Prof. Dr. Alfredo Goldman
Coorientador: Renato Cordeiro Ferreira

São Paulo, Maio de 2022

Conteúdo

Capítulo 1

Introdução

Com a popularização da computação em nuvem, sistemas modernos tornaram-se distribuídos, o que implica que as máquinas onde programas executam podem parar de funcionar a qualquer momento, de forma intermitente ou permanente. A falha de um dado servidor é inevitável, estratégias são necessárias para contornar esses problemas, mantendo alta disponibilidade e ocultando as falhas dos usuários.

Quanto trabalhamos na indústria, nos deparamos com soluções que abstraem a complexidade de lidar com sistemas distribuídos e oferecem um conjunto de garantias. Tais abstrações são úteis da perspectiva de quem desenvolve um produto, porém pouco se estuda na universidade sobre a complexidade que é escondida em soluções tradicionais na indústria.

Decisões fundamentais de engenharia de software, como por exemplo a escolha de um sistema mais apropriado de banco de dados para um determinado projeto, exigem conhecimentos mais profundos de sistemas distribuídos. Pode ser mais interessante escolher um banco de dados relacional para lidar com informações bancárias, mas a mesma decisão não se aplica tão bem a um sistema de armazenamento de imagens.

O presente projeto se propões a discutir conceitos fundamentais de sistemas distribuídos e modelos de falha, de forma a dar subsídio para a análise de requisitos em projetos reais de engenharia. Ademais, se propõe a discutir modelos de consistência de sistemas distribuídos de forma geral.

Dados os conhecimentos adquiridos de modelos de consistência, o presente projeto tem como objetivo final discutir o algoritmo de consenso RAFT e implementá-lo em Go. Em seguida, nos valeremos do algoritmo para construir um banco chave-valor com consistência forte, abrindo a caixa preta das abstrações industriais que geralmente estão presentes no contexto de desenvolvimento de aplicações comerciais.

Por fim, pretende-se que o projeto aqui realizado sirva como referência para que qualquer interessado possa aprender mais sobre sistemas distribuídos, dissipando a névoa que paira sobre as abstrações comerciais geralmente presentes na indústria.

Capítulo 2

Revisão da literatura

Nos baseando no livro *Designing data intensive applications*(Kleppmann , 2017), exploraremos alguns conceitos básicos. Definiremos sistemas distribuídos, falhas, reordenamento e perda de mensagens, modelos de consistência e, por fim, consenso.

Para as discussões a seguir, os termos técnicos foram usados em português de forma liberal, já que por não há versão traduzida do livro referência.

2.1 Sistemas distribuídos

No contexto usual de computação em nuvem, sistemas distribuídos são sistemas que operam em diferentes máquinas, reais ou virtuais. Mais especificamente, são sistemas "share-nothing", cuja memória, processador e disco são exclusivos, e qualquer troca de informação entre máquinas distintas é realizada por meio da rede.

2.2 Escalabilidade

Escalabilidade é a capacidade de um sistema de suportar carga. No contexto de sistemas distribuídos, pode ser quantificado como o número de requisições que pode ser atentida por segundo.

Há duas formas de escalar um sistema: de forma vertical, aumentando a potência do hardware, ou horizontal, aumentando o número de máquinas que serve as requisições.

A grande vantagem de sistemas distribuídos é que podemos associar um número arbitrário de máquinas para responder a mesma requisição. Há, porém, um preço que se paga: a necessidade de coordenar as múltiplas máquinas em uma rede física.

2.3 Falhas

Há dois tipos distintos de falha: fail-stop e bizantinas.

Falhas fail-stop consistem em uma unidade de computação parar de funcionar, assumindo que o comportamento da execução de código não desvia do esperado.

Falhas bizantinas consistem no desvio do comportamento esperado de um sistema, uma das possibilidades sendo ataques de agentes maliciosos. Tal modo de falha não será explorado no presente projeto.

2.4 Perda e reordenamento de mensagens

Os elementos constituintes de um sistema distribuído em arquitetura "share nothing"se comunicam por meio de uma rede. A complexidade de administrar tais sistemas vem do fato que a própria rede não é determinística: uma mensagem pode ser perdida, e um conjunto de mensagens pode chegar a um destinatório numa ordem diferente que foi emitida.

2.5 Partição na rede

Outro problema que pode surgir em sistema distribuído é a perda de comunicação entre diferentes partes de um sistema. Quando construímos um sistema distribuído, partimos da premissa que todas as máquinas são capazes de mandar mensagens entre si.

Uma partição consiste na perda da capacidade de uma parte das máquinas se comunicar com a outra, gerando duas partições desconexas

2.6 Tolerância a falhas

Quando queremos que um sistema tolere falhas, nos valemos de replicá-lo em várias máquinas distintas, de forma que a falha de uma não impossibilite que requisições sejam servidas.

Há, porém, o problema da coordenação entre as máquinas, que deve levar em consideração que as máquinas podem parar de funcionar e serão posteriormente reiniciadas, e que a rede se comporta de forma não determinística, com perda e reordenamento de mensagens enviadas, além da possibilidade de partição na rede.

Um modelo de consistência consiste em restrições na ordem que as mensagens são recebidas por diversos agentes de um sistema distribuído, de forma a garantir um dado comportamento do sistema.

Não existe uma solução única de tolerância a falha. Tudo depende da aplicação em questão, e do quão rigorosa deve ser a consistência entre as diferentes máquinas no sistema.

O presente projeto discutirá diferentes modelos de consistência, e dará especial atenção ao modelo de consistência forte, aquele em que todos os membros de um sistema recebem mensagens em uma mesma ordem. Em tal modelo, o sistema se comportaria da mesma forma que um sistema equivalente de uma única máquina.

2.7 Consenso

Algoritmos de consenso são protocolos em sistemas distribuídos que garantem a consistência forte de uma sequência de mensagens. Mais especificamente, garante que cada agente atribua a mesma ordem para um conjunto de mensagens.

O presente projeto, por facilidade de implementação, explorará o algoritmo RAFT para implementar consistência forte.

Capítulo 3

Proposta e cronograma

O presente projeto tem como objetivos:

1. Explorar a teoria dos sistemas distribuídos
   (a) Conceitos básicos
   (b) Modelos de consistência
   (c) Consistência forte e consenso
2. Apresentar o algoritmo RAFT
   (a) Discussão teórica.
   (b) Implementação no ambiente MIT.
3. Construir banco de dados chave/valor
   (a) Discutir técnicas de replicação e sharding
   (b) Implementação no ambiente MIT.
4. Migrar o projeto para ambiente cloud
   (a) Proposta da arquitetura
   (b) Implementação dos componentes
   (c) Teste das garantias de consistência

Tarefa	Entrega
Implementação do RAFT no ambiente MIT	30/06
Implementação do banco chave/valor no ambiente MIT	30/07
Arquitetura do banco chave/valor em ambiente cloud	30/09
Teste das garantias de consistência do banco em ambiente de produção	30/10
Monografia	30/11

Bibliografia

Kleppmann (2017) Martin Kleppmann. **Designing data intensive applications**. O'Reilly Media, 1st ed. Citado na pág. 2