7º ATIVIDADE de CES-27 / 2018

CTA - ITA - IEC

Prof Juliana e Prof Vitor

Importante: É permitido fazer essa atividade individualmente ou em dupla.

Objetivo: Compreender a essência dos algoritmos de consenso Paxos e Raft. Além de verificar a aplicabilidade de algoritmos de consenso em sistemas reais (como os baseados em Blockchain).

Entregar (através do TIDIA): Relatório com as respostas das questões abaixo. É permitido utilizar outras referências além dos slides da aula.

****

Paxos

****

Obs: Algumas questões pedem exemplos. Se precisar, você pode considerar falha de processos e/ou troca de ordem no recebimento de mensagens.

1) Considere um cluster com 8 *acceptors*. Houve uma partição na rede e temos agora dois grupos isolados: 4 *acceptors* de um lado e 4 *acceptors* do outro lado. Suponha que 4 *acceptors* concordam com 'cat' e 4 *acceptors* concordam com 'dog'. Consenso não é atingido. Lembre-se que Paxos se fundamenta no conceito de maioria!
   a) O que há de errado com esse cenário?
   b) Paxos consegue trabalhar com partição de rede? Discuta um exemplo.

2) Explique por que Paxos não tolera falhas em m nodes quando o cluster tem menos de $2m+1$ nodes.

3) Mostre um exemplo (com esquema ilustrativo) em que dois *proposers* recebam a *Promise* da maioria dos *acceptors*. Explique.

4) No Paxos, *proposers* usam diferentes *proposal number* (n) ao iniciarem um novo *round* (com mensagem *Prepare*).
   Suponha esse novo modo de operação para o algoritmo:
   - Diferentes *proposers* podem usar o mesmo *proposal number* (n).
   - *Acceptors* só rejeitam a mensagem recebida caso o *proposal number* (n) seja estritamente menor do que os anteriores recebidos.

   Mostre, através de um exemplo (com esquema ilustrativo), o problema de consenso que isso pode gerar.

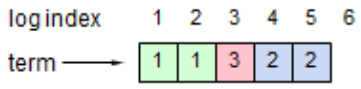
****

Raft

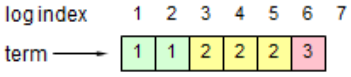
****

5) Cada figura abaixo mostra uma configuração de *log structure* do Raft Server. A figura não mostra dado guardado no *log*. A figura só possui *index* e *term.* Explique por que somente os logs das figura ‘a’ e ‘c’ fazem sentido numa implementação de Raft.

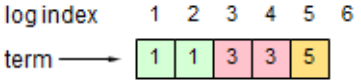
a)



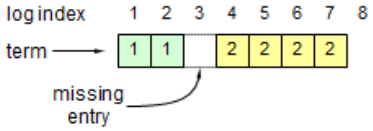
b)



c)



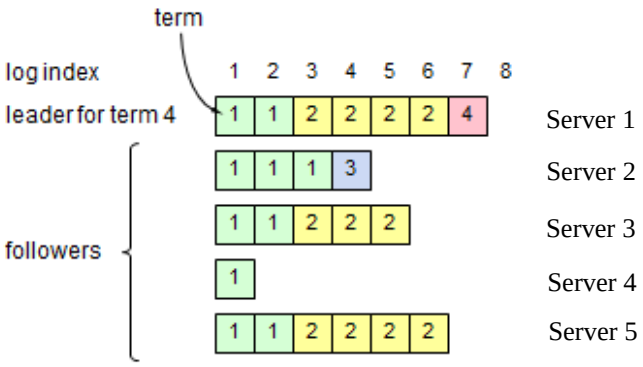
d)



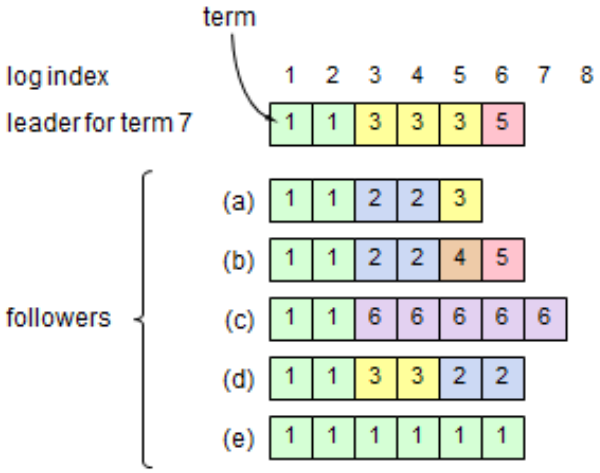
6) A figura abaixo mostra os *logs* dos 5 *servers* de um cluster. Novamente temos indicado somente *index* e *term.*

Podemos pensar em atualizar as máquinas de estados com as seguintes entradas: <1,1>, <2,1>, <3,2>, <4,2>, <5,2>. Isso porque temos maioria para atingir consenso nesses casos.

Porém somente as entradas <1,1> e <2,1> são realmente seguras. Explique por que. Discuta, por exemplo, o caso do *server* 2 assumir a liderança.



7) A figura abaixo mostra os *logs* dos 6 *servers* de um cluster. Novamente temos indicado somente *index* e *term.* Numa correta implementação do Raft, explique porque somente os logs ‘c’ e ‘e’ fazem sentido. Tenha em mente a propriedade “*log matching property*” do Raft (slide 43 da aula).



8) Suponha que você implementou Raft com todos os *servers* num mesmo datacenter. Agora você quer fazer *deploy* desse sistema sendo cada *server* em um datacenter distinto. Seria necessário alterar o *timeout* relacionado ao início de eleição. Explique (lembre-se de indicar potenciais problemas).

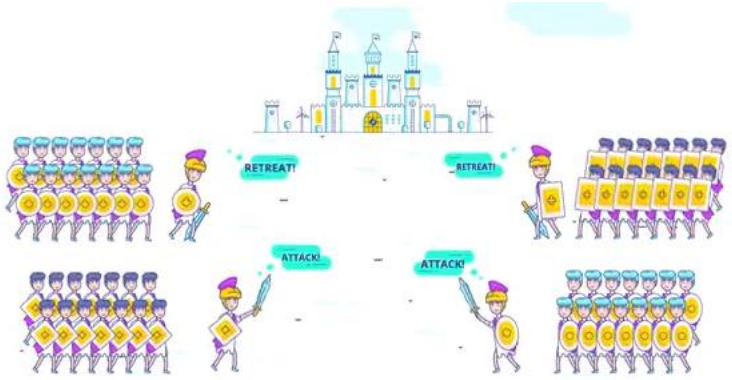
****

Outros tópicos

****

Sobre o Problema dos Generais Bizantinos

Trata-se do problema de se obter consenso na presença de incertezas. Sistemas computacionais confiáveis devem lidar com a falha de um ou mais componentes. Um componente falho pode exibir um tipo de comportamento confuso e/ou conflitante. O problema pode ser abstraído como o Problema dos Generais Bizantinos.



https://www.youtube.com/watch?v=A-mNgqJETQg

Cada exército tem um general. Os generais podem ser leais ou traidores. É requerido que todos os generais leais concordem num mesmo plano de ação (atacar ou recuar) apesar da presença de traidores. Os generais podem comunicar apenas por passagem de mensagens. Os generais traidores podem alterar as mensagens que são passadas.

Suponha que um dos generais seja o comandante e os outros sejam tenentes. O general comandante dever enviar uma ordem para os seus n-1 generais tenente tal que:

- CI1: Todos os generais tenentes leais obedecem à mesma ordem.
- CI2: Se o general comandante for leal então todos os generais tenentes leais obedecem à ordem que ele envia.

CI1 e CI2 são conhecidas como as condições de consistência interativa. Note que se o comandante é leal, CI1 segue de CI2. Contudo o comandante pode ser um traidor.

Existem soluções clássicas para este problema, por exemplo, o artigo:

> **Lamport** L, Shostak R, Pease M, The Byzantine Generals Problem, ACM TOPLAS 4(3) (July **1982**).

9) Explique a relação entre Blockchain e o Problema dos Generais Bizantinos.

10) Explique a relação entre Raft e Blockchain.

11) Explique os dois algoritmos de consenso mais conhecidos para Blockchain: PoW (*proof-of-work*) e PoS (*proof-of-stake*). Use ilustrações para facilitar a explicação.

Bom trabalho!