Relatório Projeto Sistemas Distribuídos

Carolina Coelho (99189), Pedro Cruz (99297), Nuno Ribeiro (99293)
Grupo A52 - Abril 2023

O objetivo principal deste projeto é o desenvolvimento de um sistema sobre o qual são suportadas trocas de moeda digital - **DistLedger**.

Metodologia de Resolução

Numa primeira fase, este sistema consistia apenas numa arquitetura simples em que existia apenas um servidor a fazer leituras e escritas no sistema.
Na segunda etapa, manteve-se um servidor que permitia leituras e escritas, o primário, e adicionou-se um servidor secundário, que apenas realizava leituras.
Foi notável um melhoria ao nível das leituras, no entanto as escritas continuaram a ser o *bottleneck*. Nesta terceira entrega, de modo a resolver esse *bottleneck*, todos os servidores realizam leituras e escritas.

Arquitetura Gossip

De forma a garantir a coerência entre todas as réplicas, adotámos a *gossip architecture*, com algumas simplificações mas tendo como base o livro *Distributed Systems - Concepts and Design - 5th edition: George Coulouris and Jean Dollimore and Tim Kindberg and Gordon Blair 2011 Addison Wesley.*
A *gossip architecture* é um protocolo de coerência fraca em que as réplicas trocam mensagens *gossip* entre si periodicamente, de forma a poderem atualizar outras réplicas com updates do seu updateLog. Tendo em conta ao âmbito do projeto, foram necessárias algumas simplificações. Tal como previsto no enunciado, as mensagens *gossip* não serão trocadas periodicamente, mas sim apenas quando o administrador chamar o comando *gossip* para dado servidor, após o qual este vai dar a conhecer os seus *updates* a todas as outras réplicas. Também não foi implementada a estratégia de pedir atualizações forçadas e imediatas.

Cálculo dos IDs das réplicas

Na nossa solução assumimos que as réplicas são lançadas com qualificadores sequenciais a partir do char "A". Assim a posição de cada réplica no timestamp é calculada da seguinte maneira: qualificador - "A". Deste modo conseguimos garantir um numero máximo de até 26 réplicas ao mesmo tempo.

TableTS

Cada réplica contém uma tabela de timestamps. A entrada i desta tabela corresponde ao último timestamp recebido da réplica i. Esta entrada, permitirá estimar que updates devem ser enviados no próximo gossip à réplica.

Arranque de uma réplica nova

Quando uma réplica se junta ao sistema a meio da execução, os seus timestamps começam a zero e, à medida que os outros servidores forem fazendo gossip, a réplica acabará por estabilizar.

updateLog (LedgerState)

Consideramos o nosso ledger um histórico permanente de operações submetidas no sistema, em qualquer servidor. Em nenhum momento, uma operação é removida do ledger.
Este é constituído por um conjunto de *LedgerRecords*. Cada *LedgerRecord* é constituído pela operação em si e pelo estado da operação, isto é, se já foi executada (independentemente do sucesso da mesma) ou não.

Resposta a Updates (Escritas)

Quando um cliente faz um pedido de escrita, recebe apenas uma mensagem de sucesso que indica que o seu pedido foi registado no sistema. Esta mensagem, no entanto, não apresenta garantias da operação ter sido executada com sucesso. O cliente não vai ser notificado caso a operação falhe.

Resposta a Queries (Leituras)

Quando um cliente faz um pedido de leitura, apenas recebe uma resposta imediata com o valor solicitado se as dependências causais se encontrarem imediatamente satisfeitas. Caso contrário, o cliente entra numa espera bloqueante até a réplica receber o estado necessário através do gossip.

Propagação de Estado

Após um pedido de gossip, a réplica pede ao NamingServer uma lista de todos os qualificadores existentes associados ao serviço *DistLedger*. De seguida, verifica se esta réplica é sua conhecida. Em caso negativo, expande os seus timestamps de modo a passarem a contemplar a nova réplica. Após esta ação, a réplica de origem, com base no replicaTS fornecido pela réplica alvo no gossip anterior, estima que updates faltam à réplica e envia apenas esses pedidos. Se a lista de operações a propagar for vazia, não é feita nenhuma propagação.

Receção de Estado

Quando uma réplica recebe uma lista de operações para adicionar ao seu updateLog, percorre a lista e adiciona cada operação após verificar que ela ainda não se encontra no ledger. Após adicionar as operações em falta, percorre o updateLog e vai executando as operações que satisfazem as dependências causais. Esta ação é efetuada de forma recursiva até a lista ser percorrida sem uma única alteração.

Verificação periódica dos Stubs

Tanto no AdminService como no UserService foi adicionado uma thread que periodicamente percorre o mapa de servidores conhecidos pelo serviço e atualiza os stubs correspondentes através de consultas ao NamingServer.