Transações e Controle de Concorrência em Sistemas Distribuídos

Resumo Analítico

9 de outubro de 2025

Resumo

Este artigo resume os conceitos fundamentais de transações e controle de concorrência em sistemas distribuídos (SD), focando na necessidade de garantir a integridade e a consistência dos dados em ambientes caracterizados por concorrência e falhas parciais. São abordados os mecanismos de serialização, os métodos de controle de concorrência (travamento, otimista e carimbo de tempo) e os protocolos essenciais para transações distribuídas, como o Protocolo de Confirmação de Duas Fases (2PC). Finalmente, é apresentado um cenário prático para ilustrar a aplicação e os desafios desses conceitos em sistemas de alta demanda.

1 Transações e Consistência

Em sistemas distribuídos, uma **transação** é uma sequência de operações sobre recursos compartilhados, especificadas por um cliente para serem executadas como uma **unidade indivisível** [32, 33, 36]. A função principal das transações é garantir que os dados gerenciados por um servidor sejam transformados de um **estado consistente** para outro [31].

A atomicidade da transação é crucial, garantindo que as operações sejam atômicas diante de falhas de clientes e servidores [36]. Isso implica que o resultado da transação deve ser:

1. **Serialmente Equivalente:** A execução interposta de operações feitas por clientes concorrentes deve ser equivalente a alguma ordem serial (sequencial) dessas transações [19, 33, 30, 36].

2. **Recuperável:** As operações devem ser totalmente concluídas com êxito (*commit*) e gravadas permanentemente (*durabilidade*), ou não devem ter **nenhum efeito** (*abort*) se houver falhas [33].

Os sistemas de transação lidam tipicamente com **falhas por colapso de processo** (servidores falham) e **falhas por omissão na comunicação** [32, 3]. A durabilidade e a recuperação dependem de **objetos recuperáveis**, que são aqueles cujas alterações podem ser restauradas para refletir apenas as transações confirmadas, mesmo após uma falha [4, 5].

1.1 Transações Aninhadas

Transações aninhadas (*nested transactions*) são estruturadas a partir de outras subtransações [20, 36]. Elas são particularmente úteis em sistemas distribuídos, pois permitem que as subtransações descendentes sejam executadas **concorrentemente em servidores diferentes**, proporcionando uma **maior concorrência** [20, 36].

2 Controle de Concorrência (CC)

Os protocolos de Controle de Concorrência são baseados no critério da **equivalência serial** e utilizam regras de conflito entre operações para garantir que as transações sejam serializáveis [36]. Para que isso funcione, os desenvolvedores de aplicações distribuídas devem garantir que a execução de transações seja **restrita**, o que significa atrasar operações de leitura e escrita sobre um objeto até que todas as transações que escreveram nesse objeto tenham sido confirmadas ou canceladas [6, 13].

Os três métodos de CC mais importantes são [36, 30]:

1. **Travamento (*Locking*):** É o método predominante de CC [30, 36]. Requer o uso de **travas exclusivas** para operações de escrita e travas compartilhadas para leitura [21]. Para garantir a serialização, é utilizado o **Travamento de Duas Fases Restrito** (*Restricted Two-Phase Locking*), que impõe que as travas de uma transação sejam **mantidas até que a transação seja confirmada ou cancelada** [37, 22].
   - O inconveniente principal é a possibilidade de **impasses** (*deadlocks*) [18, 36].
   - A concorrência pode ser aumentada pelo **Travamento de Duas Versões**, no qual a trava exclusiva (*commit lock*) é retardada até a confirmação da transação [7, 8].
2. **Controle de Concorrência Otimista (CCO):** Assume que os conflitos são raros [28, 34, 36]. As transações procedem até estarem prontas para a confirmação, quando passam pela **validação** [36, 38]. A validação verifica se as operações da transação (T_v) conflitam com outras transações sobrepostas (T_i) [38, 9]. Se houver conflito, a transação é cancelada (podendo ser reiniciada) [34, 23]. O CCO utiliza a **validação para trás** (em relação a transações confirmadas) ou **para frente** (em relação a transações ativas) [10].
3. **Ordenação por Carimbo de Tempo (*Timestamp Ordering*):** Atribui um carimbo de tempo (*timestamp*) exclusivo a cada transação no início, garantindo que as operações sejam executadas na ordem dos tempos iniciais [35, 36]. Se uma operação violar a ordem temporal, a transação é **cancelada** [24, 25, 27, 29]. A **Ordenação por Carimbo de Tempo de Versão Múltipla** é uma variação eficiente, pois permite manter versões antigas confirmadas de objetos, evitando que operações de leitura tardias sejam rejeitadas [11, 12].

3 Transações Distribuídas e Protocolo de Confirmação

Uma transação se torna **distribuída** se acessa objetos gerenciados por **vários servidores diferentes** [37, 1]. A atomicidade exige que todos os servidores envolvidos **confirmem-**

na ou cancelem-na [37, 36]. Para garantir o acordo, um servidor atua como **coordenador** para o **Protocolo de Confirmação Atômica** [2, 15].

3.1 Protocolo de Confirmação de Duas Fases (2PC)

O 2PC é o protocolo atômico mais comum para TDs [36, 2, 40].

1. **Fase de Votação (`canCommit?`)**: O coordenador envia a requisição `canCommit?` aos participantes [2]. Um participante só vota **Sim** se estiver **preparado para confirmar**, o que exige salvar todas as alterações da transação no armazenamento permanente (log) [38, 3]. Se o participante votar Não, ele cancela imediatamente [2, 40].

2. **Fase de Conclusão (`doCommit/doAbort`)**: O coordenador reúne os votos [40]. Se **todos** votarem Sim, o coordenador decide confirmar (`doCommit`); caso contrário, ele decide cancelar (`doAbort`) [2].

O log do servidor deve armazenar informações sobre o **status da transação** (preparada, confirmada, cancelada) e as **listas de intenções** (*intentions lists*) [17]. Isso é fundamental para a **recuperação**, garantindo que, mesmo em sistemas assíncronos e com falhas de colapso, os servidores falhos possam ser substituídos e consigam alcançar um consenso sobre o resultado das transações [19, 40].

3.2 Impasses Distribuídos

O uso de travamento em TDs pode gerar **impasses distribuídos**, que ocorrem se houver um ciclo no **grafo espera por** (*wait-for graph*) que abranja objetos em vários servidores [14, 4]. A detecção de impasses envolve métodos como o **caminhamento pelas arestas** (*edge chasing*), onde mensagens de sondagem (*probe*) seguem as arestas do grafo espera por global [16].

4 Aplicação Prática em Ambientes de Alta Demanda

Os princípios de CC e transação são essenciais para manter a **integridade dos dados** e o **desempenho** em cenários de alta concorrência [1].

4.1 Lições do Cenário de Laboratório (Rinha de Backend)

O desafio da Rinha de Backend 2025, um teste de **performance e resiliência em sistemas distribuídos** sob **limites rígidos de CPU e memória** [1], serve como um exemplo prático das dificuldades de implementar esses conceitos:

1. **Compromisso entre Consistência e Escalabilidade:** A meta de obter o "menor tempo de resposta e máxima confiabilidade"exige um equilíbrio delicado [1]. O uso de um SGBD robusto como o Postgres, que oferece fortes garantias ACID (incluindo Isolamento, geralmente via travamento), pode falhar sob alta carga devido a gargalos de memória e CPU ("uso de memória explodia") [2]. A busca por **escalabilidade** frequentemente leva a soluções que flexibilizam as garantias de consistência estrita (como o uso do Redis em memória).

2. **Controle de Concorrência e Recursos Limitados:** Em ambientes distribuídos com recursos restritos (1.5 vCPU, 350 MB), a tentativa de maximizar a concorrência pode levar a inconsistências. A otimização exige não apenas protocolos de CC eficientes, mas também o **ajuste do nível de concorrência interna** (e.g., limitando o número de *workers* ativos) para garantir que cada processo "tenha tempo e recurso suficiente para finalizar o seu trabalho", preservando a integridade dos dados.

*O cenário da Rinha demonstrou que as decisões de arquitetura e otimização são um constante *trade-off* entre o rigor teórico dos protocolos de transação e as limitações práticas de recursos e desempenho, confirmando que a implementação de sistemas distribuídos exige uma profunda compreensão das consequências dessas escolhas [1].*

Referências

[1] Excerpts from the transcript of the video "Como Venci a Rinha de Backend 2025"uploaded on the YouTube channel "Lucas Montano".

[2] Excerpts from "GitHub - zanfranceschi/rinha-de-backend-2025: Rinha de Backend - Terceira Edição".

[3] Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2013). *Sistemas Distribuídos: Conceitos e Projeto* (5ª ed.). Capítulo 16, Introdução (Modelo de Falha).

[4] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Objetos Recuperáveis.

[5] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Propriedades ACID.

[6] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Execuções Restritas de Transações.

[7] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Aumento da concorrência em esquemas de travamento.

[8] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Travamento de Duas Versões.

[9] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Regras de Conflito para Validação.

[10] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Validação Otimista (Para Trás e Para Frente).

[11] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Ordenação por Carimbo de Tempo de Versão Múltipla.

[12] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Concorrência na Ordenação por Carimbo de Tempo.

[13] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Resumo, Recuperação e Versões de Tentativa.

[14] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Impasses Distribuídos.

[15] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Transações Distribuídas (Coordenador e Participantes).

[16] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Caminhamento pelas Arestas.

[17] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Entradas no Arquivo de Recuperação.

[18] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Introdução ao Controle de Concorrência.

[19] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Serialização.

[20] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Transações Aninhadas.

[21] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Tipos de Travamento (Leitura e Escrita).

[22] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Travamento de Duas Fases Restrito.

[23] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Validação.

[24] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Carimbo de Tempo.

[25] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Ordenação por Carimbo de Tempo.

[26] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Versão Múltipla.

[27] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Cancelamento por Carimbo de Tempo.

[28] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Controle de Concorrência Otimista.

[29] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Comparação dos Métodos.

[30] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Introdução.

[31] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Objetivos das Transações.

[32] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Modelo de Falha.

[33] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Propriedades da Transação.

[34] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Controle de Concorrência Otimista.

[35] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16, Ordenação da Indicação de Tempo.

[36] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 16 (Tema 5), Visão Geral.

[37] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Transações Distribuídas (Definição e Coordenador).

[38] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Protocolo 2PC (Fases).

[39] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Log de Recuperação.

[40] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Protocolo 2PC (Exercício - Definição).

[1] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Introdução (Transações Distribuídas).

[2] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Protocolos de Confirmação Atômica.

[3] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Voto em 2PC.

[4] Coulouris, G., et al. (2013). *Sistemas Distribuídos: Conceitos e Projeto*. Capítulo 17, Impasses Distribuídos.

A Apêndices: Excertos das Fontes

Excertos da Fonte: Sistemas Distribuídos, 5ª Edição (Coulouris et al.)

[18] Todos os protocolos de controle de concorrência são baseados no critério da equi
[19] O controle de concorrência é o que é usado para garantir a serialização das tran
[20] As transações aninhadas (nested transactions) estendem o conceito de transação,
[21] O uso de travas exclusivas (para operações de escrita) e travas compartilhadas (
[22] As regras para o uso de travas em uma implementação do travamento de duas fases
[23] O controle de concorrência otimista é baseado no pressuposto de que os conflitos
[24] A ordenação por carimbo de tempo é uma alternativa para o controle de concorrênc
[25] A ordenação por carimbo de tempo usa o carimbo de tempo da transação para decidi
[26] A Ordenação por Carimbo de Tempo de Versão Múltipla melhora a concorrência e o d
[27] A requisição de uma transação para ler um objeto é aceita somente se a transação
[28] O controle de concorrência otimista é projetado para evitar as desvantagens do t
[29] O travamento de duas fases é o método de controle de concorrência mais amplament
[30] Este capítulo discute a aplicação de transações e controle de concorrência em ob
[31] O objetivo das transações é garantir que todos os objetos gerenciados por um ser
[32] O modelo de falha para transações tipicamente lida com falhas por colapso de pro
[33] Uma transação é especificada por um cliente como um conjunto de operações sobre
[34] O controle de concorrência otimista é baseado na suposição de que os conflitos e
[35] A ordenação por carimbo de tempo (timestamp) ordena as transações que acessam os
[36] Transações e Controle de Concorrência. Uma transação define uma sequência de ope
[37] O coordenador que abriu a transação torna-se o coordenador da transação distribu
[38] O teste de validação na transação Tv é baseado no conflito entre operações em pa
[39] As transações se tornam duráveis pelo estabelecimento de pontos de verificação e
[40] Um protocolo de confirmação de três fases tem as seguintes partes: Fase 1: é igu
Este capítulo se preocupa com o gerenciamento de transações que envolvem vários serv
Um servidor atua como coordenador para garantir o mesmo resultado em todos os partic
Um participante só vota Sim se estiver preparado para confirmar, o que exige salvar
O travamento de duas fases garante a serialização, mas, em transações distribuídas,

Excertos da Fonte: Vídeo "Como Venci a Rinha de Backend 2025"

[1] a rinha de backend foi um desafio de performance e resiliência em sistemas distri
[2] A Rinha de Backend é um desafio em que é necessário desenvolver uma solução backe