SISTEMAS DE RECUPERAÇÃO - CHECKPOINTS

Sérgio Mergen

Checkpoints

- Problemas da recuperação:
  - Busca no log inteiro é custosa
  - Pode-se desnecessariamente refazer transações que já foram gravadas no banco.

- Uma solução envolve o uso de checkpoints periódicos

Checkpoints

- Ações tomadas durante um checkpoint
  - Gravar todos registros de log em memória para o meio físico.
  - Gravar todos blocos em memória modificados no banco.
  - Escrever o registro de log **<checkpoint>** no disco.

- Obs. Nenhuma atualização ocorre enquanto o checkpoint está sendo realizado

Recuperação com Checkpoints – Caso simples

- Inicialmente, considere que as transações executem serialmente
  - Ou seja, uma após a outra

- Durante a recuperação é preciso considerar apenas
  - a operação mais recente Ti que iniciou antes do checkpoint
  - as transações que iniciaram depois de Ti.

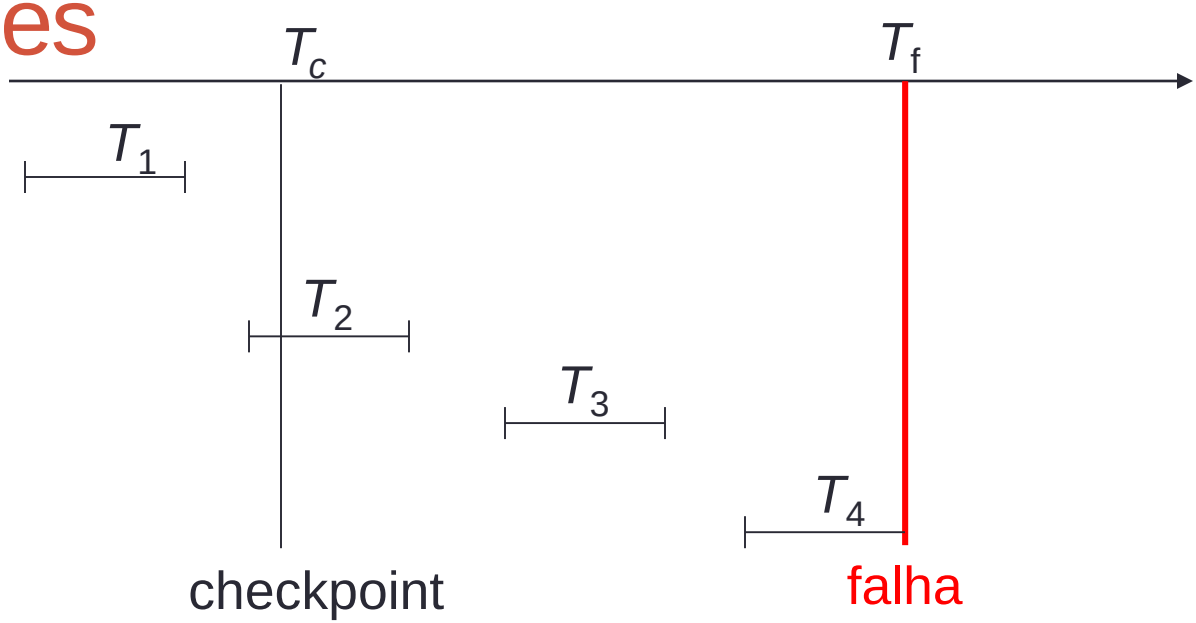
Recuperação com Checkpoints – Caso simples

- Procedimento de recuperação:
  - Varrer o log de trás para frente até encontrar o registro de **<checkpoint>** mais recente
  - Continuar a varrer para trás até encontrar o registro **<Ti start>**.
    - É preciso considerar apenas a parte do log abaixo desse registro
    - O log mais antigo pode ser ignorado e apagado quando desejado

Recuperação com Checkpoints – Caso simples

- Procedimento de recuperação:
  - Para todas transações Tx (a partir de Ti)
    - Com <Tx commit>,
      - executar redo(Ti). (apenas para instruções abaixo do checkpoint)
    - Sem <Tx commit>
      - executar undo(Ti). (esquema de modificação imediata)

Recuperação com Checkpoints – Caso simples



- T_1 pode ser ignorado (atualizações já estão no banco)
- T_2 e T_3 devem ser refeitas (redo).
- T_4 deve ser desfeita (undo)

Recuperação com Transações Concorrentes

- Pode-se modificar o esquema baseado em log para permitir que transações executem concorrentemente (e satisfaçam as propriedades ACID).

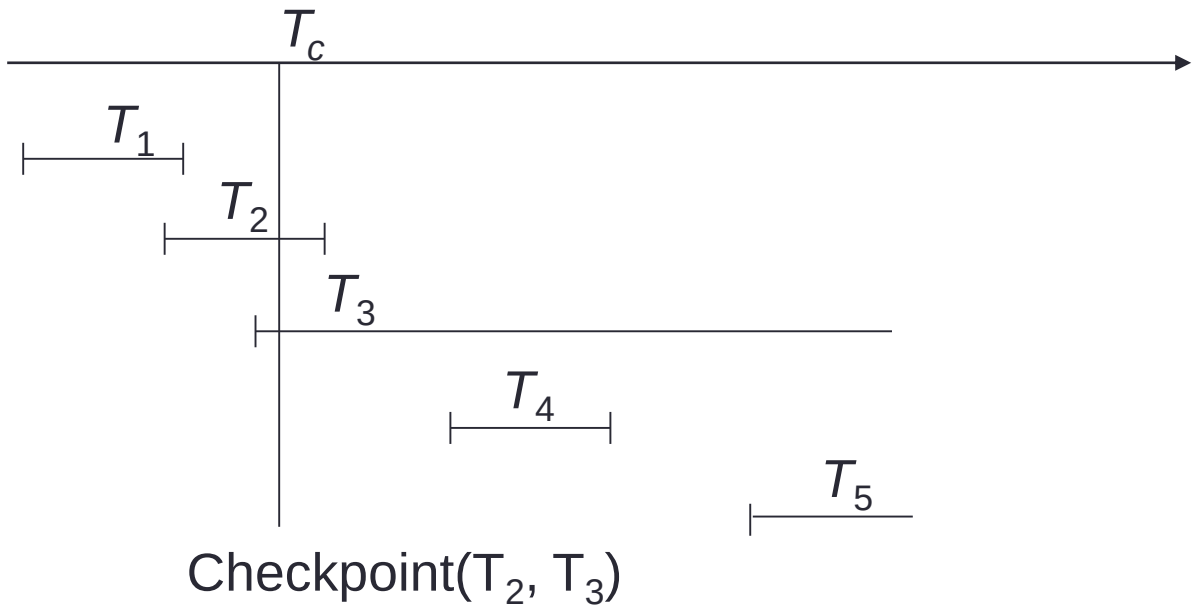
- O log é produzido como já descrito
  - Sendo que registros de transações diferentes podem aparecer juntos no log.

- Mas as técnicas de checkpoint e as ações devem mudar
  - Já que muitas transações podem estar ativas quando um checkpoint é realizado

Recuperação com Transações Concorrentes

- Novo formato do checkpoint
  - < checkpoint L>
  - onde **L** é a lista de transações ativas no momento do checkpoint

Recuperação com Transações Concorrentes



- Duas transações ativas no momento do checkpoint
  - T_2 e T_3

Recuperação com Transações Concorrentes

- Procedimento de recuperação:
  - Inicializar vazias as listas undo-list e redo-list
  - Varrer o log para trás a partir do fim, parando ao encontrar o primeiro **<checkpoint L>**
  - Todas transações encontradas são analisadas, inclusive aquelas que estão no checkpoint
  - Para as transações encontradas
    - Se tiverem commit no log
      - Vão para a redo-list
    - Caso contrário
      - Vão para a undo-list

Recuperação com Transações Concorrentes

- Procedimento de recuperação:
  - A undo-list consiste de transações incompletas que precisam ser desfeitas
  - A redo-list consiste de transações finalizadas que precisam ser refeitas.

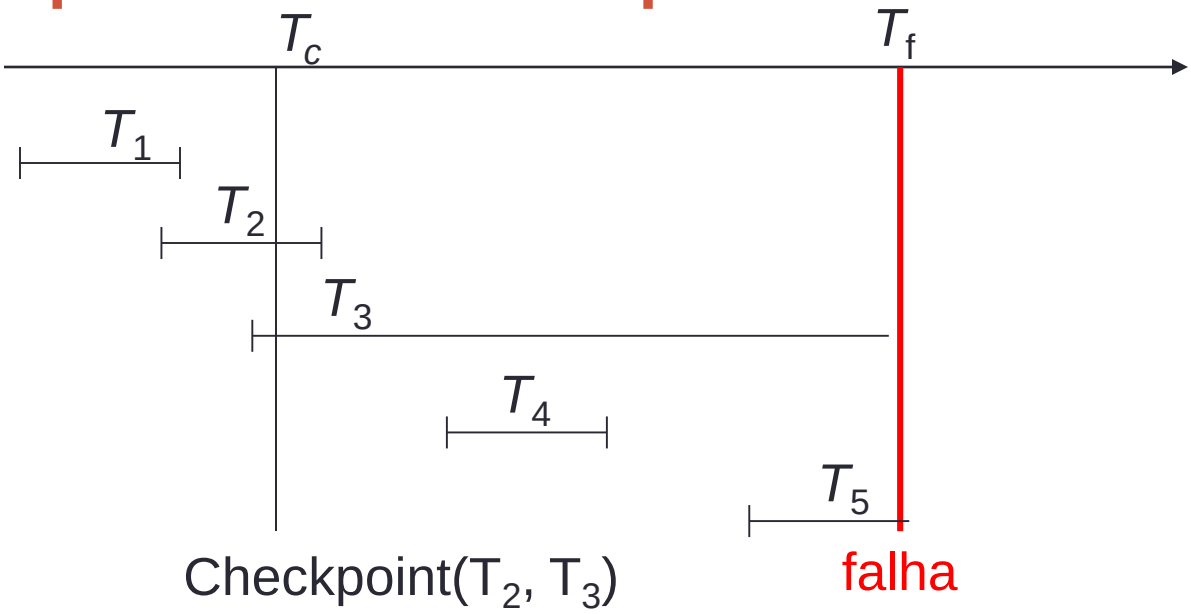
Recuperação com Transações Concorrentes

- Procedimento de recuperação:
  - Varrer novamente o log para trás a partir do fim:
    - Durante a varredura
      - Realizar **undo** para cada registro de log que pertença a uma transação da *undo-list*.
    - Parar a varredura quando
      - O registro **<T_i start>** for encontrado
        - Para todas transações da *undo-list*

Recuperação com Transações Concorrentes

- Procedimento de recuperação:
  - Localizar o registro **<checkpoint L>** mais recente
  - Varrer o log para frente a partir desse ponto:
    - Durante a varredura
      - Realizar redo para cada registro de log que pertença a uma transação da redo-list
    - No final do log
      - Parar a varredura

Exemplo de Checkpoints



- T_1 pode ser ignorado (atualizações já estão no banco)
- T_2 e T_4 devem ser refeitas (redo).
- T_3 e T_5 devem ser desfeitas (undo)

Exemplo de Recuperação 1

- Quais seriam os passos para a recuperação a partir do log abaixo?

<T_0 start>
<T_0, A, 0, 10>
<T_0 commit>
<T_1 start>
<T_1, B, 0, 10>
<T_2 start>
<T_2, C, 0, 10>
<T_2, C, 10, 20>
<checkpoint {T_1, T_2}>
<T_3 start>
<T_3, A, 10, 20>
<T_3, D, 0, 10>
<T_3 commit>

Exemplo de Recuperação 1

- Quais seriam os passos para a recuperação a partir do log abaixo?

<T_0 start>
<T_0, A, 0, 10>
<T_0 commit>
<T_1 start>
<T_1, B, 0, 10>
<T_2 start>
<T_2, C, 0, 10>
<T_2, C, 10, 20>
<checkpoint {T_1, T_2}>
<T_3 start>
<T_3, A, 10, 20>
<T_3, D, 0, 10>
<T_3 commit>

Undo T1, T2
C = 10
C = 0
B = 0

Redo T3
A = 20
D = 10

Exemplo de Recuperação 2

- Quais seriam os passos para a recuperação a partir do log abaixo?

<T_0 start>
<T_0, A, 0, 10>
<T_0 commit>
<T_1 start>
<T_1, B, 0, 10>
<T_2 start>
<T_2, C, 0, 10>
<T_2, C, 10, 20>
<T_3 start>
<T_3, A, 10, 20>
<checkpoint {T_1, T_2}>
<T_3, D, 0, 10>
<T_3 commit>

Exemplo de Recuperação 2

- Quais seriam os passos para a recuperação a partir do log abaixo?

<T_0 start>
<T_0, A, 0, 10>
<T_0 commit>
<T_1 start>
<T_1, B, 0, 10>
<T_2 start>
<T_2, C, 0, 10>
<T_2, C, 10, 20>
<T_3 start>
<T_3, A, 10, 20>
<checkpoint {T_1, T_2, T_3}>
<T_3, D, 0, 10>
<T_3 commit>

Undo T1, T2
C = 10
C = 0
B = 0

Redo T3
D = 10

Exemplo de Recuperação 3

- Quais seriam os passos para a recuperação a partir do log abaixo?

<T_0 start>
<T_0, A, 0, 10>
<T_0 commit>
<T_1 start>
<T_1, B, 0, 10>
<T_2 start>
<T_2, C, 0, 10>
<T_2, C, 10, 20>
<T_3 start>
<T_3, A, 10, 20>
<**checkpoint** {T_1, T_2, T_3}>
<T_3, D, 0, 10>
<T_3 commit>
<T_2 commit>

Exemplo de Recuperação 3

- Quais seriam os passos para a recuperação a partir do log abaixo?

<T_0 start>
<T_0, A, 0, 10>
<T_0 commit>
<T_1 start>
<T_1, B, 0, 10>
<T_2 start>
<T_2, C, 0, 10>
<T_2, C, 10, 20>
<T_3 start>
<T_3, A, 10, 20>
<**checkpoint** {T_1, T_2, T_3}>
<T_3, D, 0, 10>
<T_3 commit>
<T_2 commit>

Undo T1
B = 0

Redo T2, T3
D = 10

CONTROLE DOS BUFFERS

Buffer de Registros de Log

- Registros de log são mantidos em um buffer em memória
  - Em vez de descarregados diretamente para o meio físico.

- A ideia é descarregá-los juntos
  - Em uma única operação de saída
- Ao invés de fazê-lo um registro por vez

- Isso reduz custo de I/O.

Buffer de Registros de Log

- As regras abaixo devem ser seguidas para a descarga de registros de log
  - Os registros devem ser descarregados para o meio físico
    - Na ordem em que foram criados.
  - A transação T_i entra no estado de commit
    - Apenas quando o registro de log **<T_i commit>** for descarregado.
  - Antes de um bloco de dados modificado ser copiado para o banco
    - Todos registros de log referentes a esse bloco devem ter sido descarregados (Regra WAL)

Buffer de Registros de Log

- Os registros de log são descarregados para o meio físico
  - Quando a operação log force for executada.

- Log force pode ser executado quando
  - Uma transação comitar
  - Motivo: Para evitar que a aplicação precise ficar esperando pelo commit

- Log force deve ser executado quando
  - O gerenciador de buffer decidir enviar um bloco de dados modificado ao meio físico.
  - Motivo: para preservar a regra WAL (Write-ahead log)

Buffer de Blocos de Dados

- Até então vimos que itens de dados são bloqueados pelas transações
- Blocos de dados também podem ser bloqueados
  - Esse tipo de bloqueio é chamado de latch
- Um latch ocorre quando um bloco deve ser enviado para o meio físico
  - Nenhuma escrita pode estar em andamento no bloco enquanto essa transferência ocorre
- Os latches são liberados assim que a transferência for concluída