Sistemas Distribuídos
Transações Atômicas

Disciplina: Sistemas Distribuídos
Prof.: Edmar Roberto Santana de Rezende

Faculdade de Engenharia de Computação
Centro de Ciências Exatas, Ambientais e de Tecnologias
Pontifícia Universidade Católica de Campinas

Transações Atômicas

Introdução

- Abstração em alto nível para ocultar:
  - uso de semáforos para controle de concorrência
  - prevenção de deadlocks
  - recuperação de falhas
- Vantagem:
  - programadores concentram-se nos algoritmos das aplicações
- Sinônimos:
  - atomic transaction, transaction, atomic action.

Transações Atômicas

Exemplos de Transações

- Um cliente, em um PC ligado por modem, faz transferência de fundos de uma conta bancária para outra, em dois passos:

  (1) Saque(quantia, conta1)

  (2) Deposite(quantia, conta2)

- Se a ligação telefônica cair entre os passos (1) e (2):
  - o dinheiro desaparece!

- Solução:
  - passos (1) e (2) devem ocorrer como uma transação atômica (como se fosse um único passo)
  - se a ligação telefônia cair entre os passos (1) e (2), os efeitos do passo (1) devem ser cancelados

Transações Atômicas

Armazenamento Estável

- Informação armazenada em RAM é perdida se faltar energia ou se a máquina falhar
- Informação armazenada em disco é perdida se a cabeça do disco falhar
- Informação armazenada em memória estável sobrevive
- Implementação típica:
  - disco replicado

Transações Atômicas

Primitivas Transacionais

- BEGIN_TRANSACTION:
  - marca o início da transação
- END_TRANSACTION:
  - termina a transação e tenta fazer o commit
- ABORT_TRANSACTION:
  - destrói a transação;
  - restaura os valores anteriores (do início da transação)
- READ:
  - lê dados de um objeto (por exemplo, um arquivo)
- WRITE:
  - escreve dados em um objeto

Transações Atômicas

Primitivas Transacionais

```
BEGIN_TRANSACTION
        reserve São Paulo - Salvador
        reserve Salvador - Brasília
        reserve Brasília - São Paulo
END_TRANSACTION

BEGIN_TRANSACTION
        reserve São Paulo - Salvador
        reserve Salvador - Brasília
        reserve Brasília - São Paulo => ABORT_TRANSACTION
```

Transações Atômicas

Propriedades das Transações

- Atômica:
  - para o mundo externo, a transação ocorre de forma indivisível
- Consistente:
  - a transação não viola invariantes de sistema
- Isolada:
  - transações concorrentes não interferem entre si (*serializable*)
- Durável:
  - os efeitos de uma transação terminada com *commit* são permanentes

Transações Atômicas

Isolamento

```
BEGIN_TRANSACTION
        x = 0;
        x = x + 1;
END_TRANSACTION
```

```
BEGIN_TRANSACTION
        x = 0;
        x = x + 2;
END_TRANSACTION
```

```
BEGIN_TRANSACTION
        x = 0;
        x = x + 3;
END_TRANSACTION
```

Transações Atômicas

Escalonamentos

Escalon. 1	Escalon. 2	Escalon. 3
x = 0;	x = 0;	x = 0;
x = x + 1;	x = 0;	x = 0;
x = 0;	x = x + 1;	x = x + 1;
x = x + 2;	x = x + 2;	x = 0;
x = 0;	x = 0;	x = x + 2;
x = x + 3;	x = x + 3;	x = x + 3;
Legal	Legal	Ilegal

Transações Atômicas

Transações Aninhadas

- A transação top-level cria sub-transações que executam em paralelo, em processadores distintos:
  - melhor desempenho
  - programação mais simples

- Se uma transação top-level abortar:
  - todas as suas sub-transações também devem abortar

- Uma sub-transação
  - herda todos os objetos controlados pela transação top-level
  - faz cópia local de todos os objetos herdados e só repassa os novos valores destes objetos à transação top-level em caso de *commit* da sub-transação

Transações Atômicas

Implementação

- Métodos de controle sobre modificações:
  - Espaço de trabalho privado
  - Log

- Protocolo de *commit* em duas fases

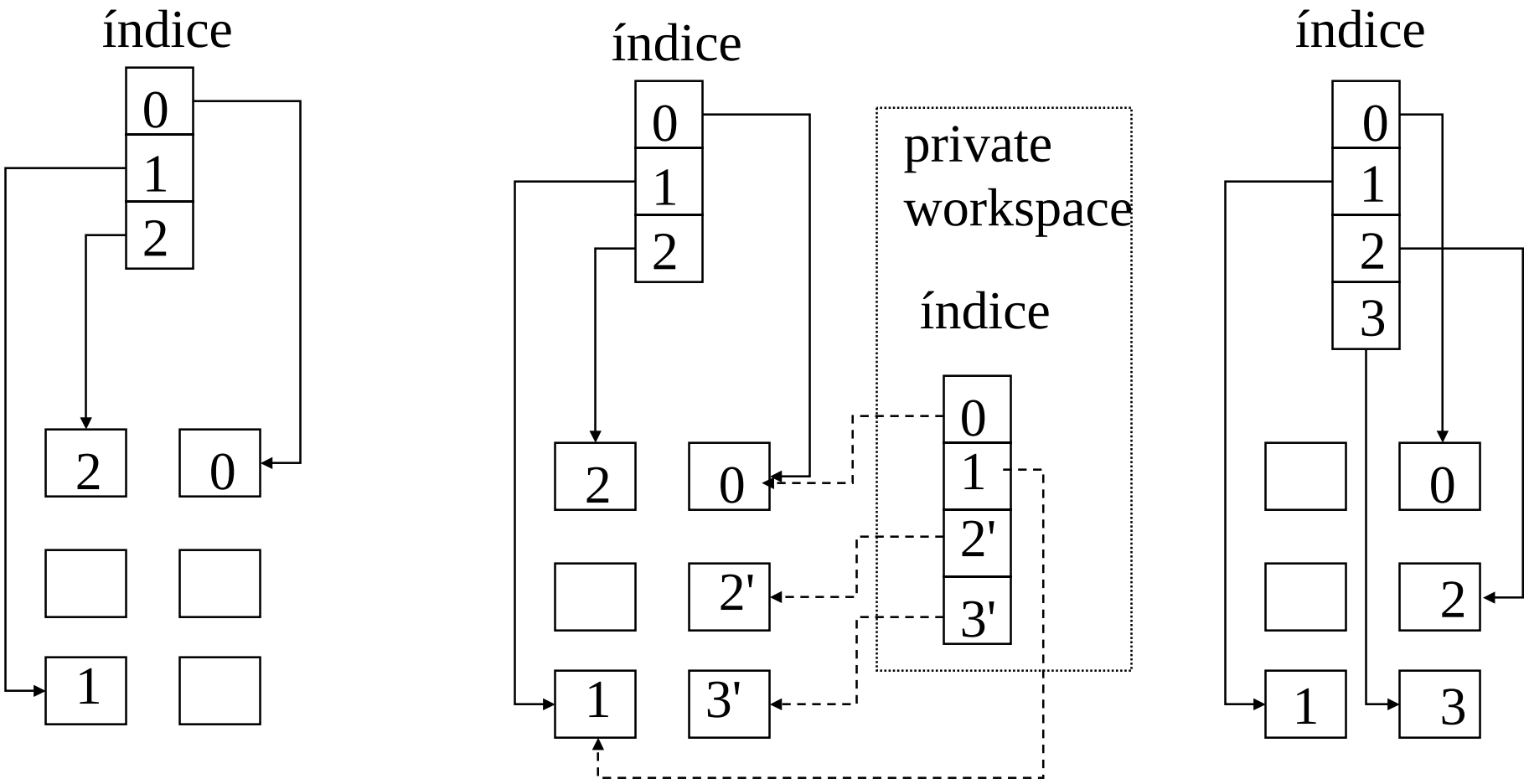
Transações Atômicas

Área de Trabalho Privada

- Um processo que começa uma transação
  - cria um espaço contendo cópias de todos os objetos manipulados pela transação
- Se ocorrer *commit*:
  - a transação repassa os novos valores dos objetos para os seus originais
- Problema:
  - alto custo
- Otimização:
  - *shadow blocks*

Transações Atômicas

Área de Trabalho Privada



Transações Atômicas

Log

- *Writeahead log* ou *intentions list*
  - Os objetos originais são modificados durante a transação
  - Antes de cada modificação, um registro é escrito em um arquivo de log (em memória estável)
  - Cada registro de log informa o valor anterior e o valor novo de um objeto, além de informar que transação fez a modificação no objeto
  - Se ocorrer *commit* um registro apropriado é inserido no log
  - Se ocorrer *abort* todas as operações efetuadas pela transação são desfeitas com base no log, começando pelo último registro (*rollback*)

Transações Atômicas

Log

```
x = 0;
y = 0;
BEGIN_TRANSACION
      x = x + 1;
      y = y + 2;
      x = y * y;
END_TRANSACION
```

Log

x = 0/1
y = 0/2
x = 1/4

Transações Atômicas

Protocolo de *Commit* em Duas Fases

- A ação de *commit* deve ser “instantânea” e indivisível
- Pode ser necessária a cooperação de muitos processos, em máquinas distintas
  - cada qual com um conjunto de objetos envolvidos na transação
- Um processo é designado como o coordenador
  - normalmente o próprio cliente que inicia a transação
- Os demais processos:
  - são designados como subordinados
- Toda ação é registrada em log:
  - armazenado em memória estável, para o caso de falha durante a transação

Transações Atômicas

Protocolo de *Commit* em Duas Fases

- Fase 1:
  - O coordenador registra “prepare” em log e envia a mensagem “prepare” para os subordinados
  - Um subordinado registra “ready” / “abort” em log e envia a mensagem “ready” / “abort” para o coordenador
  - O coordenador coleta todos as mensagens “ready”

- Fase 2:
  - O coordenador registra a decisão em log e envia mensagem “commit” / “abort” para os subordinados
  - Um subordinado registra “commit” / “abort” em log, toma a ação correspondente e envia mensagem “finished” ao coordenador

Transações Atômicas

Controle de concorrência

- Locking
  - Um gerente centralizado ou distribuído registra todos os locks e rejeita pedidos de lock em objetos já alocados a outros processos
    - lock para escrita deve ser exclusivo, mas lock para leitura pode ser compartilhado
    - quanto menor a granularidade do lock maior a chance de paralelismo, mas também maior é a chance de deadlock
  - Lock em duas fases:
    - *growing*: todos os locks são adquiridos
    - *shrinking*: todos os locks são liberados
  - *Strict two-phase locking*:
    - a fase *shrinking* ocorre “instantaneamente” (previne *cascade aborts*)

Transações Atômicas

Controle de concorrência

- Controle otimista
  - Os objetos são modificados sem preocupação com concorrência até o fim da transação
  - Quando chegar o momento de *commit*:
    - a transação verifica se outra transação modificou os mesmos objetos que ela tenha modificado
  - Se não há conflito:
    - o *commit* é feito (repasse de objetos do espaço de trabalho privado)
    - senão é feito um *abort*

Transações Atômicas

Controle de concorrência

- Timestamps
  - Cada transação recebe um timestamp único em seu início
  - Cada objeto no sistema tem um read timestamp e um write timestamp, dizendo que transação fez a operação
  - Se transações são “curtas” e “espaçadas” no tempo, normalmente, quando uma transação fizer acesso a um objeto, os timestamps do objeto serão mais velhos que o timestamp da transação
    - caso contrário a transação está “atrasada” e deve ser abortada