Sistemas Distribuídos

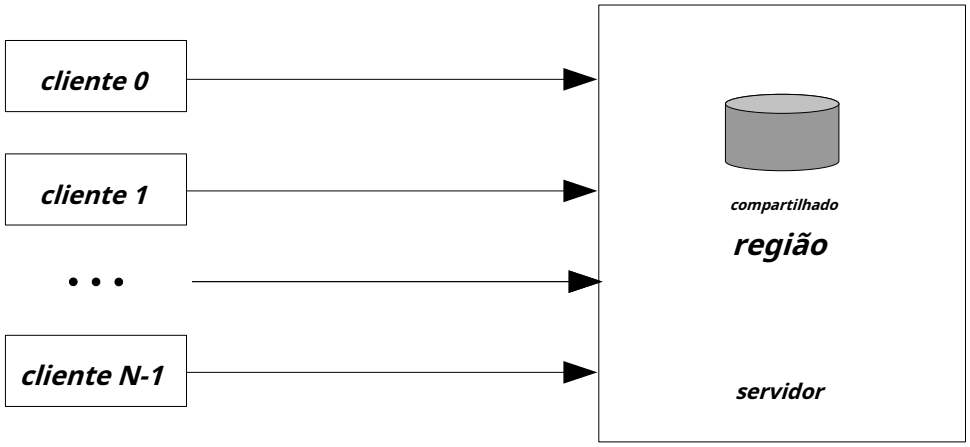
Transações Distribuídas

António Rui Borges

Resumo

- *O que é uma transação?*
  - *Caracterização*
- *Organização das operações*
  - *Problemas de simultaneidade*
- *Transações distribuídas*
  - *Compromisso em duas etapas*
  - *Compromisso em três etapas*
- *Leitura sugerida*

O que é uma transação? - 1



A *transação* pode ser pensado em um conjunto de *ler* gosto e *escrever* como operações solicitadas por um cliente para serem realizadas em registros de uma região compartilhada que o servidor que a gerencia deve realizar como uma unidade indivisível.

Assim, o servidor deve garantir que os resultados do conjunto de operações sejam salvos como um todo em armazenamento permanente, ou descartados como um todo.

O que é uma transação? - 2

As propriedades das transações podem ser descritas pela sigla ACID (Härder e Reuter)

- *atomicidade*–uma transação deve ser tudo ou nada
- *consistência*–uma transação deve levar a região compartilhada de um estado consistente para outro
- *isolamento*–uma transação deve ser realizada sem qualquer interferência de outra, ou seja, os efeitos intermediários de uma transação não podem ser vistos por outras transações que estejam ocorrendo ao mesmo tempo
- *durabilidade*–uma vez realizada com sucesso, os efeitos gerados por uma transação devem durar para sempre.

Organização das operações - 1

Uma transação é criada e gerenciada por um *coordenador* processo no lado do servidor. A pedido do cliente, uma transação é aberta pelo *coordenador* processo e um *ID da transação* lhe é atribuído.

Todos os seguintes *ler* gosto e *escrever* operações semelhantes em registros específicos emitidos pelo cliente devem ter o *ID da transação* anexado a ele e retornar a operação *status*: ou a operação foi bem-sucedida e o cliente pode prosseguir com outras operações, ou toda a transação é abortada. O próprio cliente pode, a qualquer momento, abortar a transação que está sendo executada em seu nome.

Para concluir a transação, o cliente emite um comando de fim de transação. Se o status for bem-sucedido, todo o conjunto de *ler* gosto e *escrever* como as operações são *comprometido* e as alterações produzidas nos registros endereçados da região compartilhada tornam-se permanentes; caso contrário, um status de interrupção será retornado.

Quando uma transação é abortada, o cliente pode tentar novamente mais tarde.

Organização das operações - 2

Bem-sucedido **transação**	**Abortado** **transação** **(pelo cliente)**	**Abortado** **transação** **(pelo servidor)**	**Abortado** **transação** **(pelo servidor)**
transação aberta	transação aberta	transação aberta	transação aberta
operação 1	operação 1	operação 1	operação 1
operação 2			operação 2
...	...	...	...
...	...	operação K (abortar)	...
...	abortarTransação		...
operação N			operação N
fecharTransação (sucesso)			fecharTransação (abortar)

Problemas de simultaneidade - 1

Tendo múltiplas transações realizando operações nos mesmos registros da região compartilhada potencialmente ao mesmo tempo, as condições de corrida tornam-se um problema e devem ser tratadas de forma controlada para que não sejam geradas inconsistências nos dados.

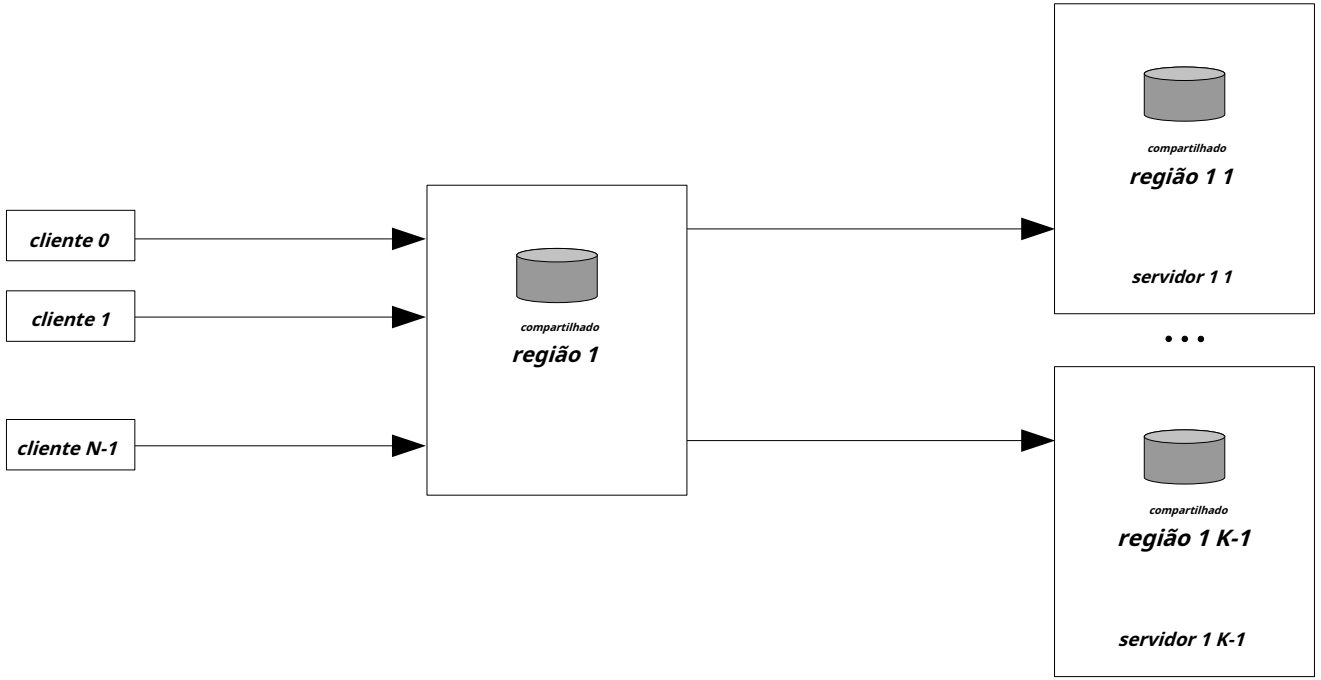
Para começar, deve-se considerar o impacto dos pares de *ler* gosto e *escrever* como operações realizadas por duas transações diferentes no mesmo registro.

Transação T1	Transação T2	Impacto
ler	ler	nenhum por si só
ler	escrever	depende da ordem de execução
escrever	escrever	depende da ordem de execução

Problemas de simultaneidade - 2

Portanto, se duas transações tentarem modificar um determinado registro, o registro deverá ser bloqueado primeiro por uma das transações e depois a modificação será realizada em uma cópia. Somente quando a transação for concluída, o conteúdo do registro será atualizado e o bloqueio será liberado. Então, a segunda transação pode prosseguir.

Transações distribuídas



Em um *transação distribuída*, a região de interesse compartilhada é dividida em diversas partes, cada uma sendo executada por um servidor diferente.

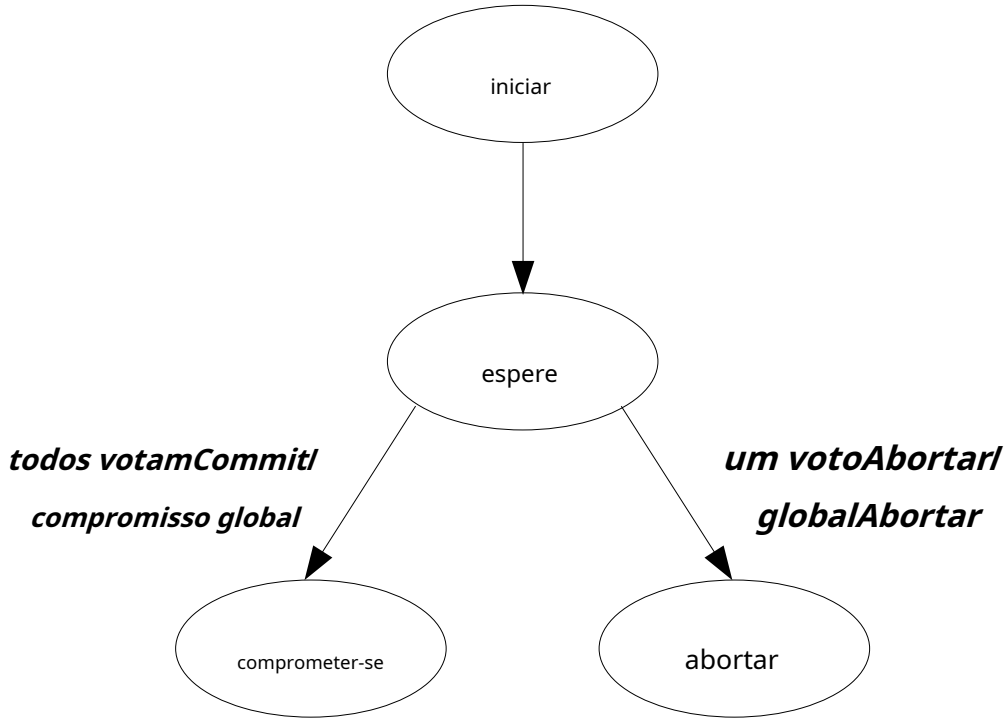
A questão chave que se coloca agora é como garantir que os resultados do conjunto de operações realizadas pelos diferentes servidores sejam guardados como um todo em armazenamento permanente, ou descartados como um todo.

Compromisso em duas etapas - 1

- o processo *coordenador* processo envia a mensagem *votarPedido* para todos os *participante* processos na transação
- quando um *participante* processo recebe a mensagem, ele responde enviando a mensagem
  - *votarCommit* para o *coordenador* processo, se estiver pronto para comprometer sua parte da transação
  - *votarAbortar* para o *coordenador* processo, caso contrário
- o processo *coordenador* coleta o *votação* mensagens e envia de volta a mensagem
  - *compromisso global* para todos os *participante* processos, se todos eles votaram *votarCommit* pela sua parte na transação
  - *globalAbortar* para todos os *participante* processos, se pelo menos um deles declarou que não pode comprometer sua parte da transação
- o *participante* processos aguardam a confirmação do *coordenador* processo, se a mensagem recebida for
  - *compromisso global*, eles comprometem sua parte na transação
  - *globalAbortar*, eles descartam sua parte da transação.

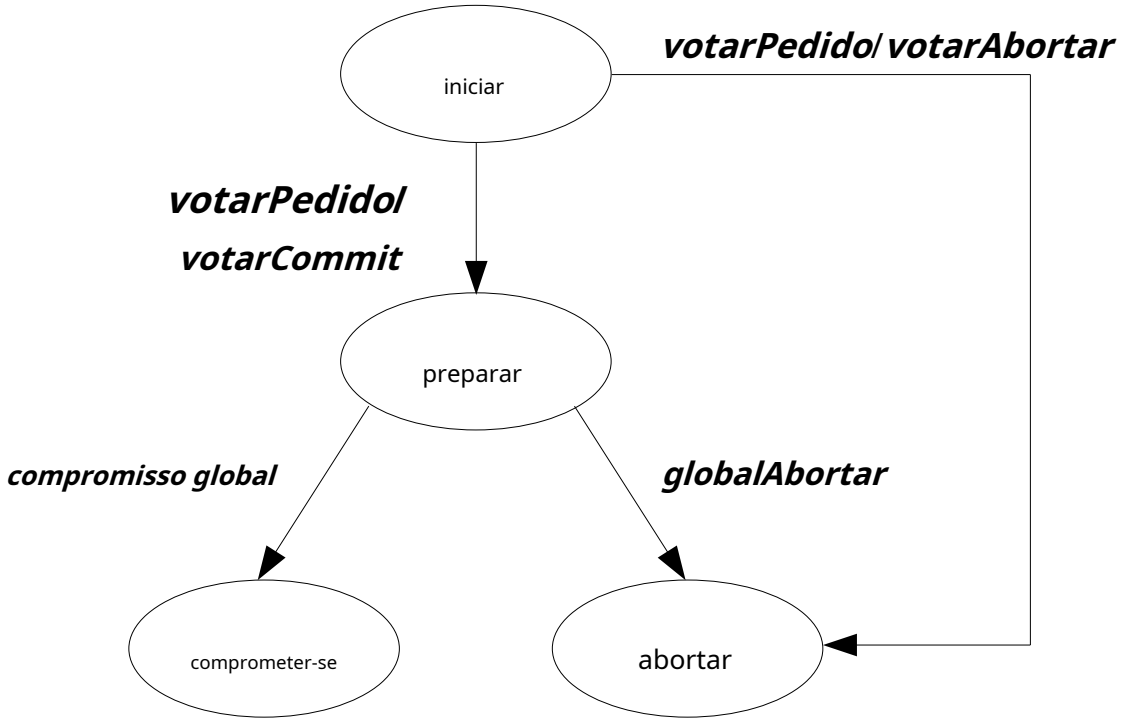
Compromisso em duas etapas - 2

processo de coordenação (nenhuma falha ocorreu)



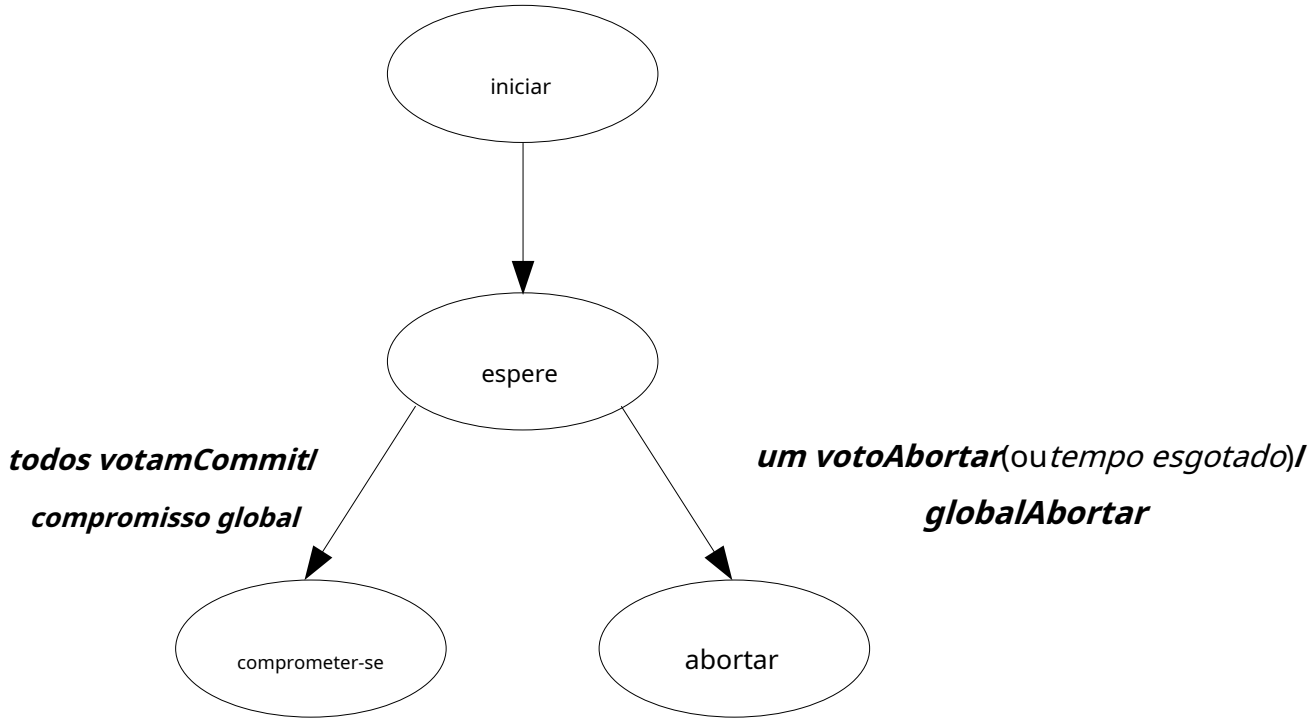
Compromisso em duas etapas - 3

processos participantes (nenhuma falha ocorreu)



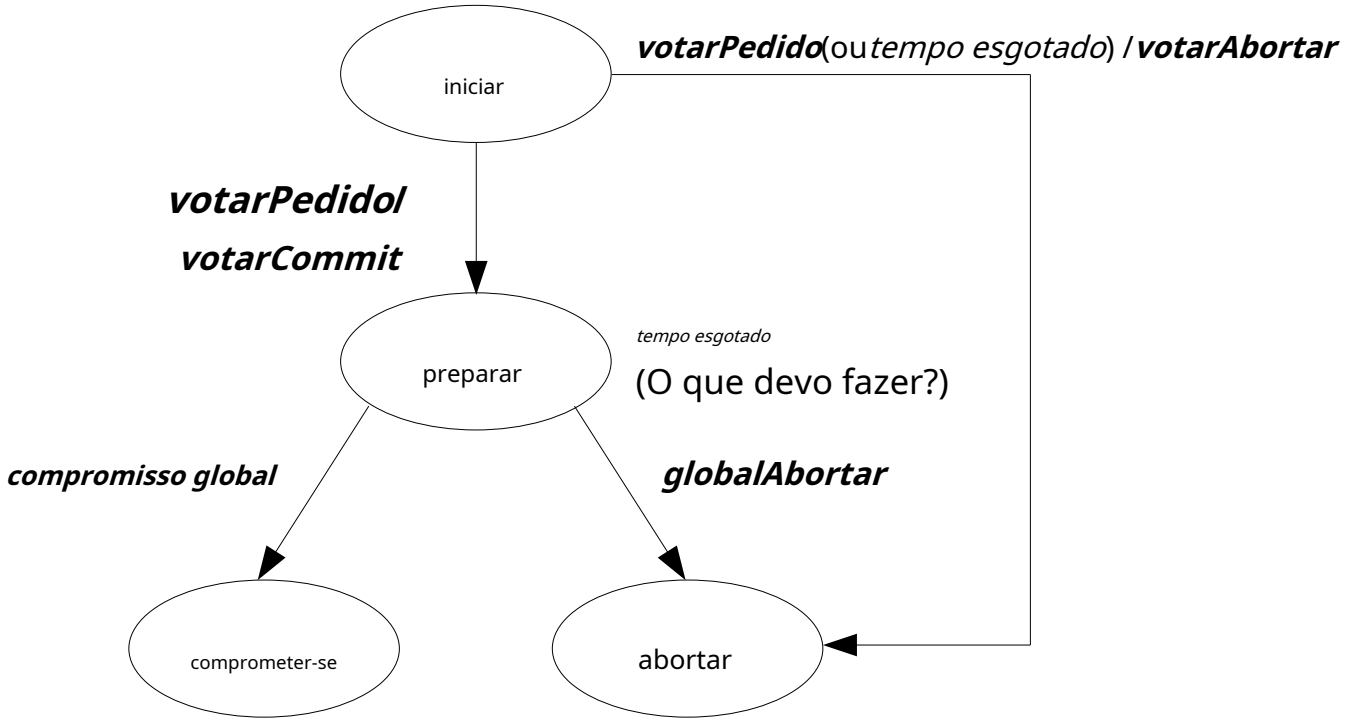
Compromisso em duas etapas - 4

processo de coordenação (ocorreram falhas)



Compromisso em duas etapas - 5

processos participantes (ocorreram falhas)



Compromisso em duas etapas - 6

o que p_{eu} deveria fazer

estado de p (*j*≠*eu*)	**ação a ser tomada por *p_{eu}***
iniciar	trânsito para abortar
preparar	entre em contato com outro participante
comprometer-se	trânsito para cometer
abortar	trânsito para abortar

Compromisso em duas etapas - 7

processo de coordenação

estado = inicialização;
writeLocalLog (estado);
númeroDeVotos = 0;
multicast (voteRequest) para todos os processos participantes; estado = esperar;
writeLocalLog (estado);
enquanto(númeroDeVotes <N)
{**Esperar por**(voto);
  se (tempo esgotado)
    { estado = abortar;
      writeLocalLog (estado);
      multicast (globalAbort) para todos os processos participantes; **retornar**;

    }
  writeLocalLog (votação);
}

Compromisso em duas etapas - 8

processo de coordenação (continuação)

```
se(todos os participantes votaram voteCommit)
      { estado = confirmar;
          writeLocalLog (estado);
          multicast (globalCommit) para todos os processos participantes;
      }
      outro{estado = abortar;
                  writeLocalLog (estado);
                  multicast (globalAbort) para todos os processos participantes;
                }
```

Compromisso em duas etapas - 9

processos participantes

```
estado = inicialização;
writeLocalLog (estado);
Esperar por(votarReq); se(
tempo esgotado)
      { estado = abortar;
          writeLocalLog (estado);
          retornar;
      }
se(participante vota voteAbort)
      { estado = abortar;
          writeLocalLog (estado);
          unicast (coord, voteAbort); retornar;

      }
```

Compromisso em duas etapas - 10

processos participantes (continuação)

```
se(participante vota voteCommit)
    { estado = pronto;
      writeLocalLog (estado);
      unicast (coord, voteCommit); Esperar
      por(decisão); se(tempo esgotado)

          multicast (decisionRequest) para todos os outros processos participantes; outro{
          writeLocalLog (decisão);
                    se(decisão == globalC                omitir)
                        { estado = confirmar;
                          writeLocalLog (estado);
                          retornar;
                        }
                        outro{estado = abortar;
                                  writeLocalLog (estado);
                                  retornar;
                                }
                  }
    }
```

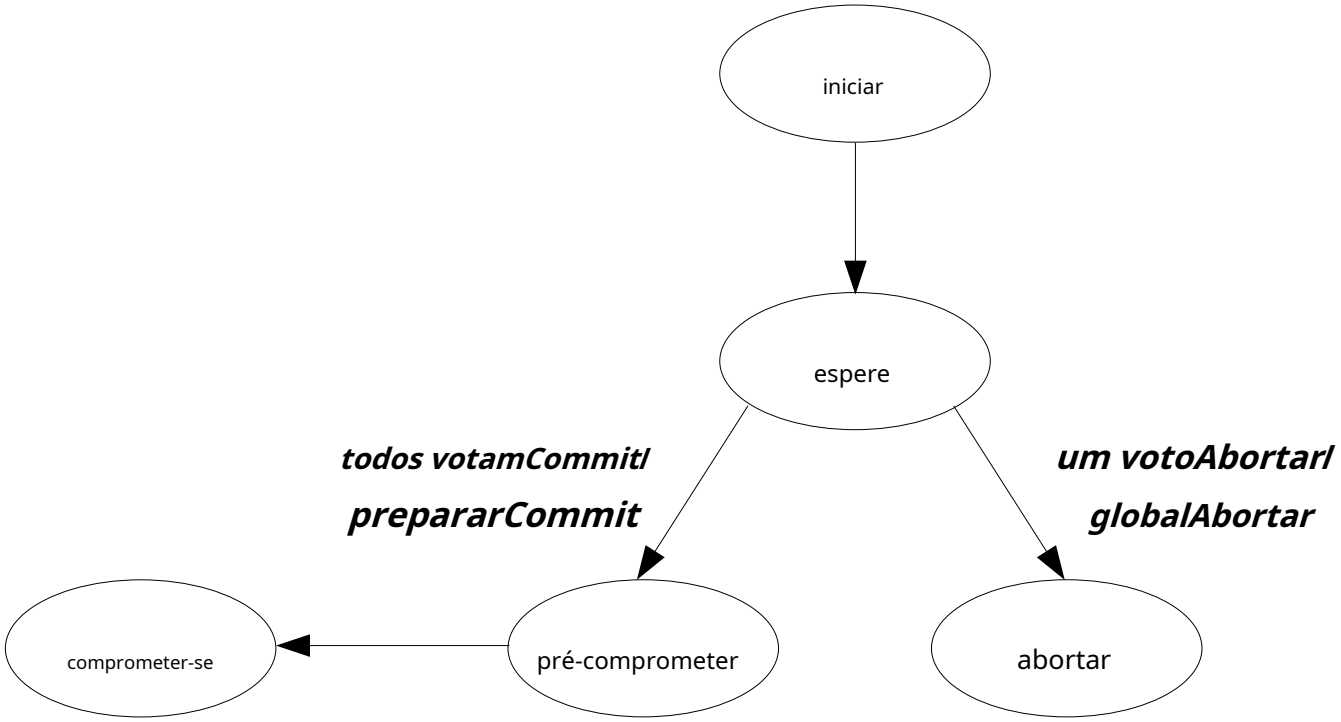
Compromisso em duas etapas - 11

processos participantes(continuação)

```
enquanto(verdadeiro)
{Esperar por(solicitação de decisão);
    se(decisãoRequest == commit)
        { estado = confirmar;
            writeLocalLog (estado);
            retornar;
        }
        senão se((decisãoRequest == init) || (decisãoSolicitação == abortar))
                    { estado = abortar;
                        writeLocalLog (estado);
                        retornar;
                    }
                    outrobloquear novamente;
}
```

Compromisso de três estágios - 1

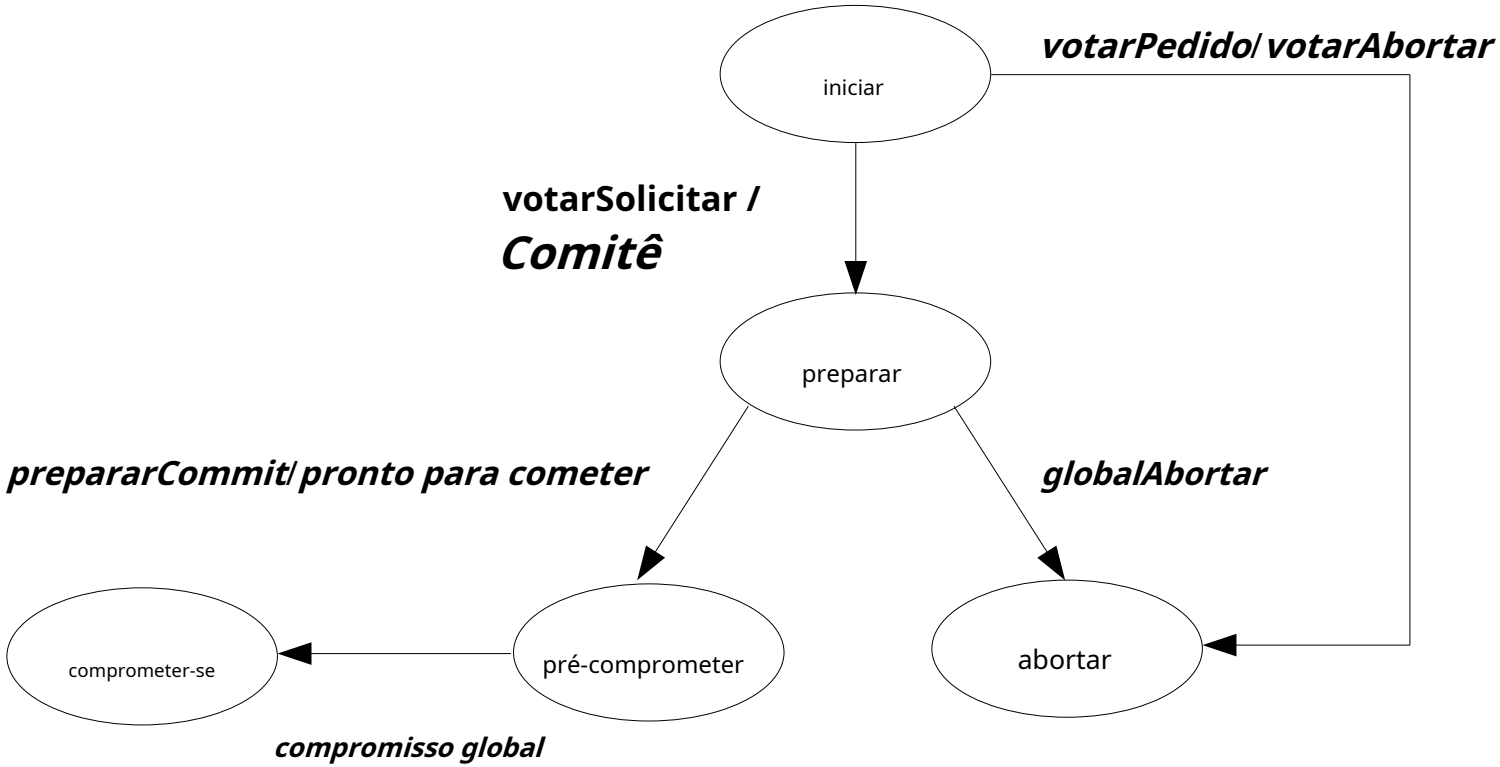
processo de coordenação (nenhuma falha ocorreu)



tudo pronto para Commit/ compromisso global

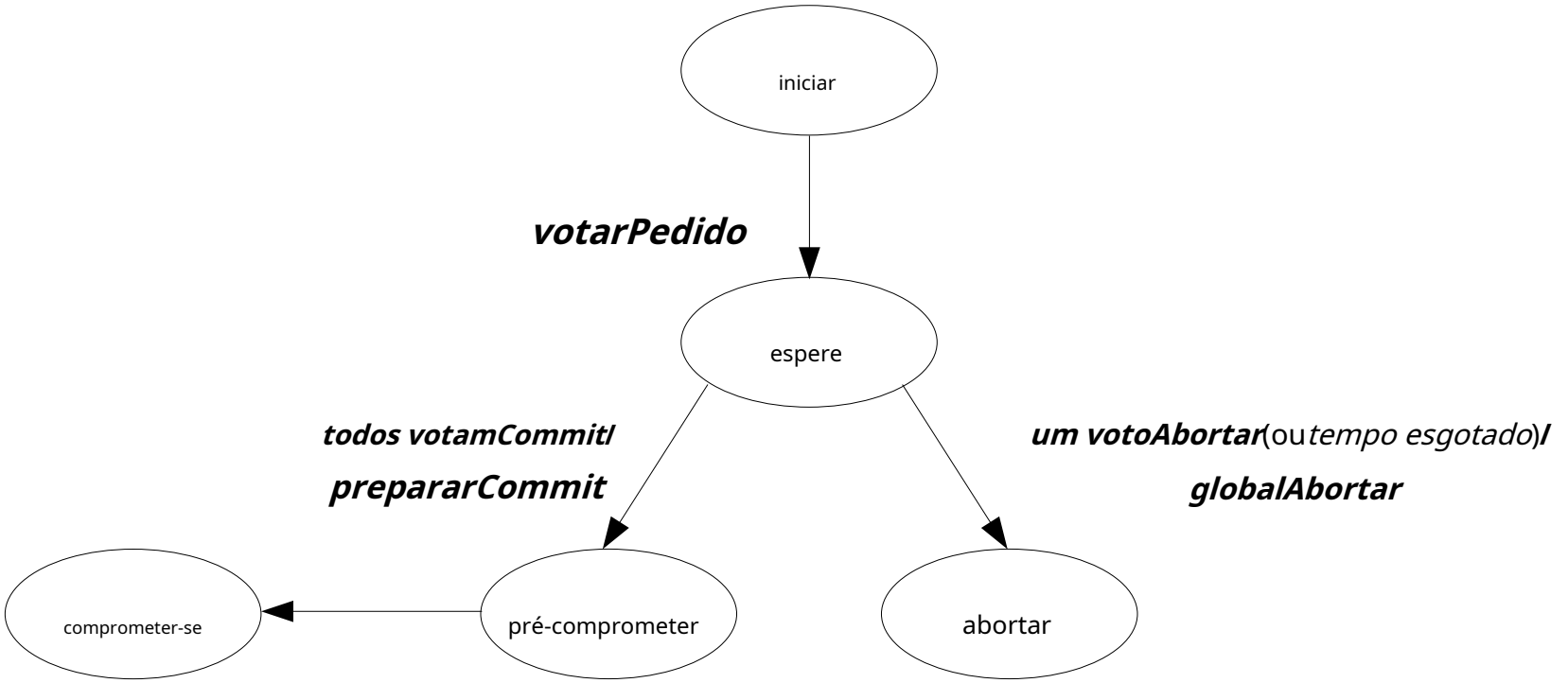
Compromisso em três etapas - 2

processos participantes (nenhuma falha ocorreu)



Compromisso em três etapas - 3

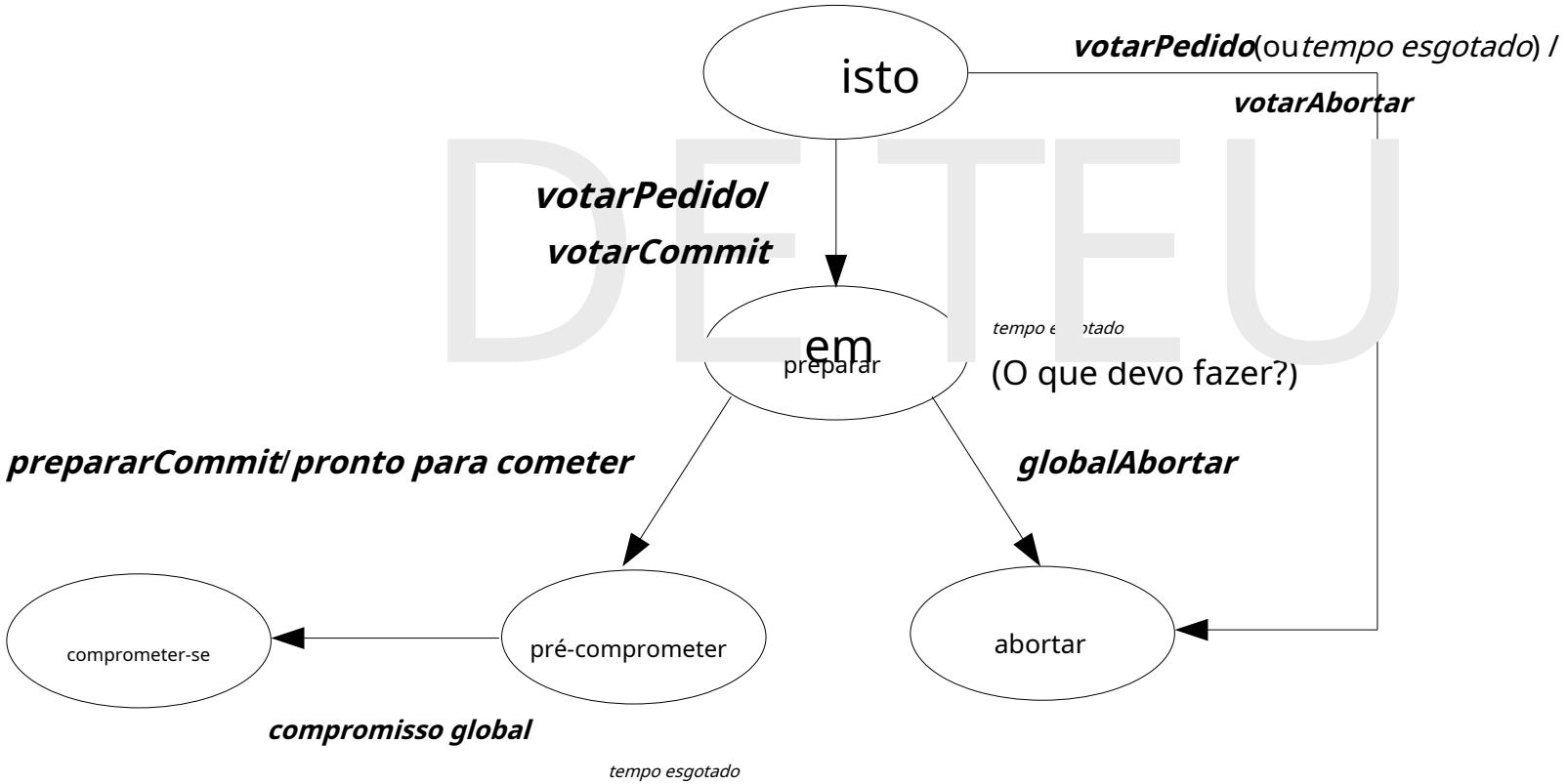
processo de coordenação (ocorreram falhas)



tudo pronto para Commit (ou *tempo esgotado*) ***/ compromisso global***

Compromisso em três etapas - 4

processos participantes (nenhuma falha ocorreu)



Leitura sugerida

- *Sistemas Distribuídos: Conceitos e Design, 4ºEdição, Coulouris*, Dollimore, Kindberg, Addison-Wesley
  - Capítulo 13: *Transações e controle de simultaneidade*
    - Seções 13.1 a 13.4
  - Capítulo 14: *Transações distribuídas*
    - Seções 14.1 a 14..3
- *Sistemas Distribuídos: Princípios e Paradigmas, 2ª Edição,* Tanenbaum, van Steen, Pearson Education Inc.
  - Capítulo 8: *Tolerância ao erro*
    - Seções 8.1 e 8.5