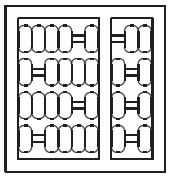

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Um Controle de Concorrência Híbrido para Adaptação de Transações em Ambientes Móveis: provas de corretude

Tarcisio da Rocha
Maria Beatriz Felgar de Toledo

Technical Report - IC-07-008 - Relatório Técnico

March - 2007 - Março

Um Controle de Concorrência Híbrido para Adaptação de Transações em Ambientes Móveis: provas de corretude

Tarcisio da Rocha Maria Beatriz F. de Toledo*

Resumo

Mecanismos de controle de concorrência têm sido de grande importância para mediar o acesso concorrente a recursos computacionais compartilhados. Esses mecanismos podem ser categorizados na abordagem otimista ou pessimista. Tais abordagens possuem vantagens e desvantagens que podem ser melhor exploradas quando se trata de ambientes de comutação móvel sujeitos a desconexões, largura de banda variável e alto custo da comunicação. Este relatório apresenta um controle de concorrência híbrido que busca reunir as vantagens das abordagens otimista e pessimista de forma a melhor lidar com as características de ambientes móveis. Como enfoque, esse relatório apresenta provas de corretude do controle de concorrência apresentado.

1 Introdução

Mecanismos de controle de concorrência têm sido de grande importância para mediar o acesso concorrente a recursos computacionais compartilhados. Em particular, com relação ao acesso a dados compartilhados entre aplicações, mecanismos de controle de concorrência possuem a função de evitar que a consistência dos dados seja comprometida quando acessados concorrentemente.

Considerando sua importância, diversos mecanismos de controle de concorrência foram propostos na literatura. No geral, esses mecanismos podem ser enquadrados em duas categorias básicas: os otimistas e os pessimistas. Os mecanismos de controle de concorrência pessimistas são aqueles que tentam evitar de antemão os tipos de acesso concorrente a dados que podem gerar inconsistências. Isso normalmente é feito bloqueando temporariamente o acesso a dados a algumas aplicações enquanto uma outra os acessa. Já os mecanismos de controle de concorrência otimistas, ao invés de tentar evitar antecipadamente acessos inconsistentes aos dados, permitem o livre acesso. Porém, no final da execução das aplicações, é instaurado um processo de validação que verifica se houve inconsistência nos dados causado pelo acesso concorrente das mesmas.

Essas duas abordagens de controle de concorrência possuem cada uma vantagens e desvantagens. Uma vantagem da abordagem pessimista seria a de justamente detectar e evitar

*Ambos do Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
Pesquisa desenvolvida com apoio financeiro da CAPES

antecipadamente acessos inconsistentes. Uma desvantagem dessa abordagem seriam os atrasos na execução das aplicações causados pelos bloqueios impostos ao acesso a dados. Em contrapartida, uma vantagem da abordagem otimista seria a de permitir que as aplicações possam executar independentemente uma das outras, ou seja, sem sofrerem atrasos causados pelo acesso concorrente das demais. Porém, uma desvantagem seria com relação ao risco que cada aplicação corre te ter a sua execução invalidada na fase de validação.

Um dos ambientes onde mecanismos de controle de concorrência são utilizados é o da computação móvel. Nesse ambiente, computadores portáteis podem fazer parte da computação distribuída independentemente se suas localizações ou características de movimento. Devido à série de restrições existentes em ambientes de computação móvel (por exemplo, desconexões, largura de banda variável e alto custo da comunicação), mecanismos de controle de concorrência puramente otimistas ou puramente pessimistas se tornam inadequados.

Tendo em vista as características de ambientes móveis, esse relatório apresenta um mecanismo de controle de concorrência híbrido que mescla as abordagens otimista e pessimista. O objetivo desse mecanismo é prover às aplicações um leque de opções que melhor se adequem ao estado corrente do ambiente bem como aos seus requisitos. Como enfoque, esse relatório apresenta provas de corretude do controle de concorrência apresentado.

2 Fundamentos

2.1 A Computação Móvel

A computação móvel é um paradigma que permite que um computador portátil equipado com uma interface de comunicação sem fio possa participar de uma computação distribuída independentemente da sua localização física ou estando em movimento. Isto se tornou possível graças aos avanços nas tecnologias de telecomunicação, redes e dispositivos de computação portáteis.

A arquitetura de apoio à computação móvel normalmente apresentada pela literatura [3, 6, 5] é composta por unidades fixas e unidades móveis. Uma Unidade Móvel é um computador móvel capaz de se comunicar com a rede fixa através de um meio de comunicação sem fio. As unidades fixas da rede são máquinas fixas ou estações de apoio que se comunicam entre si através de uma rede fixa de alta velocidade.

Uma Máquina Fixa é um computador da rede fixa que normalmente não possui interface de comunicação sem fio. Uma Estação de Apoio pode ser definida como um computador que possui interface de comunicação sem fio e é, portanto, capaz de manter uma conexão sem fio com unidades móveis. As estações de apoio agem como interfaces entre as unidades móveis e os demais componentes da rede fixa.

Cada estação de apoio possui uma área de cobertura chamada Célula Sem Fio[1]. Essa é a área por onde uma unidade móvel pode se mover e manter a conexão com a estação de apoio correspondente. Ao se mover, uma unidade móvel poderá sair de uma célula e entrar em uma outra mantendo a comunicação com a rede fixa através de outra estação de apoio.

[1] Wireless Cell

A computação móvel tem exigido várias mudanças no desenvolvimento de soluções no que diz respeito a circuitos integrados, processamento de sinais, projeto de rede e projeto de sistemas computacionais. Essas mudanças são necessárias porque a computação móvel introduziu uma série de novos obstáculos normalmente não existentes em ambientes tradicionais. Alguns desses obstáculos são citados a seguir [4, 5, 9]:

Baixa largura de banda. Redes sem fio possuem uma largura de banda bem inferior a das redes com fio. Técnicas como caching e compressão de dados têm sido usadas para reduzir os impactos da baixa largura de banda.

Custo da comunicação. O custo da comunicação sem fio pode ser alto a depender da tecnologia que se esteja utilizando (por exemplo, via celular). Isso pode exigir o uso de técnicas que reduzam o tempo de alocação do canal de comunicação como a *operação desconectada*.

Desconexões freqüentes. A susceptibilidade a desconexões freqüentes e inesperadas das redes sem fio requer dos dispositivos móveis um certo nível de autonomia, ou seja, requer o poder de continuar a operar enquanto desconectados da rede fixa. Para tanto, são usadas técnicas de operação assíncrona como *busca antecipada*[2] e *escrita com atraso*[3]. A *busca antecipada* é a ação de fazer uma cópia dos prováveis objetos que serão usados por um usuário em seu dispositivo móvel de forma que eles estejam disponíveis localmente quando a desconexão ocorrer. A *escrita com atraso* é uma técnica utilizada para atualizar os objetos remotos da rede fixa aproveitando os momentos em que haja largura de banda adequada para a transferência.

Escassez de recursos dos dispositivos móveis. Recursos como energia, espaço em disco e capacidade de processamento podem ser escassos em dispositivos móveis. Em particular, o suprimento de energia dos dispositivos móveis normalmente é feito por baterias que possuem uma capacidade limitada de suprimento. Isso pode exigir técnicas para a redução do consumo de energia por parte dos elementos de hardware e software.

Segurança. A segurança da comunicação sem fio é bem mais vulnerável do que na comunicação com fio, pois o sinal transmitido através da rede sem fio pode ser interceptado por outros dispositivos de captação intrusos. As soluções adotadas para lidar com esse problema são a criptografia e a autenticação feitas por software ou hardware especializado.

Maior dinamismo dos dados. Informações que são consideradas estáticas para um computador estacionário poderão passar a ser dinâmicas para um computador móvel. Um exemplo desse dinamismo é o endereço de rede do computador móvel que pode mudar cada vez que houver mudança de uma célula para outra.

[2] Prefetching
[3] Delayed write-back

2.2 Controle de Concorrência

Nesta seção, serão discutidos mecanismos de controle de concorrência aplicados a transações. Antes de iniciar a discussão a respeito desses mecanismos, observemos que tradicionalmente uma transação possui as seguintes características:

(a) realiza um conjunto operações sobre uma base de dados transformando-a de um estado consistente para um outro estado consistente;

(b) é permitido que uma transação deixe, durante a sua execução, o estado da base de dados temporariamente inconsistente contanto que o item anterior seja satisfeito.

Levando-se em conta essas duas características, o meio mais simples de implementar um controle de concorrência para transações é através da execução serial. Com a execução serial, uma transação só iniciaria a sua execução quando nenhuma outra transação estivesse em execução. Observando-se a característica *(a)* pode-se inferir que uma execução serial de transações consegue manter a consistência da base de dados.

A execução serial, apesar de garantir a consistência da base de dados, anula as vantagens da multiprogramação (onde várias transações poderiam executar em paralelo) trazendo grandes perdas de desempenho. Para contornar esse problema, estudos mostraram que é possível permitir a execução paralela de um conjunto de transações sobre uma mesma base de dados e que essa execução seja equivalente a uma execução serial do mesmo conjunto de transações – serialização [1] . Isso garantiria a consistência da base de dados e ao mesmo tempo acabaria com a restrição da execução serial.

Esse é um dos grandes desafios dos mecanismos de controle de concorrência de transações: permitir execução de transações em paralelo e ao mesmo tempo garantir a serialização. Para que isso seja possível, alguns algoritmos de controle de concorrência como o *two-phase locking* (2PL) [1] tentam evitar que uma transação acesse os resultados intermediários de uma outra transação (ver característica *(b)* acima).

No geral, mecanismos de controle de concorrência podem ser enquadrados em duas grandes classes: os controles otimistas e os pessimistas. Os controles de concorrência pessimistas são aqueles que evitam antecipadamente que um item de dado compartilhado se torne inconsistente devido ao acesso de transações concorrentes. Para tanto, controles de concorrência pessimistas normalmente bloqueiam o acesso ao recurso quando a operação é conflitante com as operações dos demais processos concorrentes. Um exemplo de controle de concorrência pessimista é o *two-phase locking*.

Ao contrário dos pessimistas, controles de concorrência otimistas são tão permissivos quanto possível no que diz respeito a não bloquear o acesso a um recurso compartilhado quando processos concorrentes tentam acessá-lo. Esses controles de concorrência são ditos otimistas porque confiam na hipótese de que o acesso concorrente não tornará inconsistente o recurso compartilhado. Como, na prática, essa hipótese nem sempre é verdadeira, os mecanismos de controle de concorrência otimistas provêem mecanismos para devolver a consistência ao recurso que se tornou inconsistente.

2.2.1 Controle de Concorrência Otimista

Esta seção apresenta um controle de concorrência otimista que pode ser usado por um sistema de transações para garantir a consistência dos dados acessados por transações concorrentes [2]. Esse controle de concorrência é estruturado em três fases:

Fase de leitura. nessa fase, cada transação recebe uma cópia do item de dado que deseja acessar chamada versão de tentativa. É sobre sua versão de tentativa que uma transação possui permissão para livremente realizar operações de leitura e/ou de escrita. Quando existem várias transações concorrendo ao mesmo item de dado, várias versões de tentativa desse mesmo item de dado podem coexistir. Durante essa fase, cada transação só acessa as suas próprias versões de tentativa, ou seja, as modificações feitas por uma transação não são visíveis a outras transações. As versões de tentativa de uma transação são ainda divididas em dois sub-conjuntos: um conjunto de leitura que agrupa os itens de dados que são utilizados pela transação somente para operações de leitura e um conjunto de escrita que agrupa aqueles itens de dados que receberão operações de escrita por parte da transação.

Fase de validação. depois que uma transação realizou todas as suas operações sobre as suas versões de tentativa, os itens de dados originais a partir dos quais as versões de tentativa foram copiadas têm que ser atualizados. Porém, como podem existir várias versões de um mesmo item de dado mantidas por diferentes transações, faz-se necessário um processo de validação que detecte e resolva os conflitos existentes. Se uma transação for validada com sucesso ela pode efetivar suas operações em um processo de commit. Senão, algum mecanismo de resolução de conflitos deve ser usado ou a transação conflitante deve ser abortada.

Fase de escrita. se a transação foi validada com sucesso, os itens de dados correspondentes às versões de tentativa da transação devem ser atualizados e se tornar permanentes, isto é, armazenadas em disco. Isso garante que, depois que uma transação é finalizada com sucesso, seus dados não serão mais perdidos em decorrência de falhas.

2.2.2 Controle de Concorrência Pessimista

Um controle de concorrência pode ser considerado pessimista quando evita de antemão a ocorrência de inconsistências geradas pelo acesso concorrente. A idéia principal de um controle de concorrência pessimista é bloquear o acesso ao item de dado a transações que desejam realizar operações que são conflitantes com as operações de uma transação que está utilizando o mesmo item de dado.

Um mecanismo de controle de concorrência baseado em trancas[4] muito utilizado é o *two-phase locking*. Com esse mecanismo, são definidos dois tipos básicos de trancas: de escrita e de leitura. Antes que uma transação realize uma operação de leitura ou de escrita sobre um determinado item de dado, ela terá que adquirir uma tranca de leitura ou de escrita, respectivamente. A Tebela 1 mostra a relação de conflito entre esses dois tipos de trancas ($*$ indica que os modos de tranca são conflitantes).

[4] Locks

	Leitura	Escrita
Leitura		*
Escrita	*	*

Tabela 1: Relação de conflito entre trancas

Analisando a relação de conflito entre trancas mostrada na Tabela 1, pode-se afirmar que: *(i)* a tranca de escrita é exclusiva, ou seja, uma transação só consegue adquirir uma tranca de escrita quando nenhuma outra transação possui tranca de leitura ou escrita sobre o item de dado; *(ii)* duas ou mais transações podem adquirir simultaneamente trancas de leitura sobre o mesmo item de dado contanto que nenhuma outra transação possua um tranca de escrita sobre o mesmo item de dado.

O algoritmo *two phase locking* é baseado nas seguintes regras:

(1) Quando uma transação T_1 deseja realizar uma operação sobre um item de dado, verifica-se se a operação desejada (leitura ou escrita) é conflitante com alguma tranca que já tenha sido adquirida por uma outra transação T_2 sobre o mesmo item de dado. Se a operação for conflitante, a transação T_1 tem que esperar a liberação da tranca da transação T_2. Quando não existirem mais conflitos com as trancas de outras transações, a transação T_1 poderá adquirir a sua tranca sobre o item de dado e assim acessá-lo.

(2) Durante a sua execução, uma transação T_1 pode adquirir trancas sobre diferentes itens de dados de acordo com a regra *(1)*, porém, depois que uma dessas trancas for liberada, a transação não poderá mais adquirir trancas.

Esse algoritmo é chamado *two-phase locking* porque é dividido em duas fases: fase de crescimento e fase de decrescimento. A fase de crescimento e onde a transação adquire as suas trancas e a fase de decrescimento é onde as trancas são liberadas. Pode-se observar que a regra *(2)* garante que uma vez liberado uma tranca de uma transação, essa não mais poderá adquirir trancas. essas regras do método *two-phase locking* garantem a serialização de transações [1].

3 A Plataforma de Gerenciamento de Transações

Esta seção apresenta uma Plataforma de Gerenciamento de Transações Adaptáveis (PGTA) [7, 8] para a qual o controle de concorrência que será apresentado foi proposto. A PGTA utiliza esse controle de concorrência como um mecanismo de adaptação para transações em ambientes de computação móvel.

3.1 Visão Geral

A PGTA é uma plataforma que provê apoio transacional sob a abordagem de adaptação colaborativa – dividindo as responsabilidades do processo de adaptação entre as transações e a plataforma:

PGTA. É responsável pelo monitoramento de recursos e por notificar transações quando a disponibilidade dos recursos (ex: largura de banda, custo da comunicação, espaço em disco e energia) sofre alguma mudança significativa. A PGTA também é responsável por prover mecanismos de adaptação que podem ser usados por transações para se adaptar a mudanças no ambiente.

Transação. É individualmente responsável por requisitar o monitoramento de recursos de acordo com seus interesses. A transação também é responsável por manter a sua política de adaptação a ser adotada como reação às mudanças do ambiente. O modelo de adaptação provido pela plataforma pode ser resumido nos seguintes itens:

(i) a transação requisita o monitoramento de recursos especificando uma janela de tolerância (limites inferior e superior) para cada recurso do seu interesse;

(ii) a PGTA registra a requisição da transação e inicia o monitoramento do recurso requisitado. Se a disponibilidade do recurso requisitado extrapolar os limites especificados na janela de tolerância, a PGTA envia uma notificação para a respectiva transação;

(iii) quando a transação recebe a notificação, ela pode reagir à mudança usando mecanismos de adaptação;

(iv) a transação pode voltar ao item (i) e especificar novos limites de tolerância dos recursos.

Um dos mecanismos de adaptação para transações providos pela plataforma é a de escolher entre executar suas operações sobre objetos em cache da unidade móvel ou executar sobre objetos remotos da rede fixa. Essa flexibilidade é provida através de três diferentes modos de operação dentre os quais uma transação pode optar por executar.

3.2 Arquitetura

A arquitetura da PGTA é composta por *(i)* repositórios de objetos localizados em máquinas da rede fixa, *(ii)* Plataforma de Gerenciamento de Transações Adaptáveis (PGTA) localizada em cada estação de suporte e em cada unidade móvel e *(iii)* Transações (Trn) que podem executar tanto em unidades móveis quanto em estações de suporte. Nesse modelo, uma transação que executa em uma unidade móvel ou em uma estação de suporte pode acessar tanto objetos locais (em relação à máquina onde a transação está executando) quanto remotos (nos repositórios de objetos da rede fixa). A PGTA é composta basicamente pelos seguintes componentes: Monitor de Recursos, Gerenciador de Cache, Repositório de Objetos e Gerenciador de Mobilidade.

Monitor de Recursos (MR). agrega módulos responsáveis pelo monitoramento de recursos como, por exemplo, largura de banda da comunicação sem fio, reserva de energia armazenada nas baterias do dispositivo móvel, custo da comunicação sem fio e espaço em disco do dispositivo móvel. Cada transação pode requisitar ao MR o monitoramento de um ou mais recursos. Em cada requisição, a transação deve informar o recurso a ser monitorado e a janela de tolerância (limites superior e inferior toleráveis

pela transação). Se, durante o monitoramento, o nível do recurso extrapolar um dos limites especificados pela transação, o MR envia uma notificação à mesma. Uma vez notificada, a transação poderá tomar as medidas de adaptação de acordo com a sua política de adaptação.

Gerenciador de Cache (GC). responsável por copiar objetos remotos para a cache da máquina onde a transação está executando. A cópia local de objetos aumenta a sua disponibilidade em caso de falhas e desconexões, reduz a necessidade de uso constante da comunicação sem fio e aumenta a autonomia das transações que executam em unidades móveis. Quando uma transação acessa cópias em cache, ela é posteriormente submetida a um processo de validação para garantir sua serialização.

Repositório de Objetos (RO). componente responsável por gerenciar a persistência de objetos. O RO provê mecanismos usados pela PGTA para a garantia da propriedade da durabilidade de transações e da recuperação do estado consistente de objetos em caso de falhas. O RO é um componente que faz parte da PGTA, porém pode ser mantido em servidores na rede fixa.

Gerenciador de Mobilidade (GM). responsável por dar a uma transação a capacidade de se transformar em um agente móvel. O GM age como uma agência que provê o apoio necessário para que uma transação possa se mover de uma máquina para outra. Isso pose ser usado como mecanismo de adaptação para transações que buscam máquinas que possuam mais recursos.

4 O Controle de Concorrência

O controle de concorrência da PGTA mescla as abordagens otimista e pessimista como mecanismo de adaptação aos obstáculos de ambientes de computação móvel. Para isso, esse controle de concorrência categoriza três diferentes modos dentre os quais uma transação pode ser executada. O primeiro modo, chamado *remoto*, provê controle de concorrência numa abordagem totalmente pessimista (baseado em trancas [5]). O segundo modo, chamado *local*, provê controle de concorrência numa abordagem totalmente otimista (baseada em números de versão). O último modo, chamado local-remoto, provê controle de concorrência que combina as abordagens otimista e pessimista.

4.1 Os Modos de Operação

O controle de concorrência apresentado pode ser utilizado por transações em três diferentes modos de operação: modo remoto (R), modo local-remoto (LR) e modo local (L). Cada um desses modos será apresentado como a seguir.

[5] Em inglês: locks

4.1.1 Modo Remoto (R)

Quando uma transação executa no modo de operação R todas as operações da mesma sobre os objetos participantes são executadas na(s) máquina(s) remota(s) onde os objetos se encontram. O acesso aos objetos remotos estará sujeito ao controle de concorrência *two-phase locking*.

O *two-phase locking* é um controle de concorrência pessimista que utiliza trancas para garantir a serialização de transações concorrentes. Esse controle de concorrência tem sido largamente usado para garantir regras de consistência de diversos sistemas de gerenciamento de banco de dados.

4.1.2 Modo Local-Remoto (LR)

Quando uma transação executa no modo de operação LR, os objetos participantes são copiados para a cache da máquina onde a transação está executando pela chamada transação de caching. Com isso, as operações da transação correspondente passam a ser realizadas sobre as cópias em cache. Isso possibilita que a transação execute mesmo enquanto desconectada do restante da rede.

A transação no modo LR recebe acesso exclusivo aos objetos participantes da rede fixa. O acesso exclusivo garante que as operações realizadas localmente pela transação sejam posteriormente efetivadas nos servidores de objetos da rede fixa. Porém, esse modo de operação impõe um prazo de expiração para a transação. Se a transação no modo LR não for finalizada dentro desse prazo, ela será forçada a abortar. Isso impede que transações no modo LR bloqueiem os objetos da rede fixa por tempo indeterminado impossibilitando que outras transações os utilizem.

Quando uma transação no modo LR chega à fase de efetivação, as modificações feitas sobre as cópias dos objetos mantidas em cache são propagadas para as suas correspondentes nos servidor es da rede fixa.

4.1.3 Modo Local (L)

No modo L, assim como no modo LR, a transação de caching copia os objetos remotos para a máquina onde a transação se encontra. Porém, a transação que executa no modo L não obtém acesso exclusivo aos objetos remotos nem recebe um prazo de expiração. Quando a transação entra na fase de efetivação, o Protocolo de Validação do Modo Local (PVML) verifica se a transação é válida ou não. Se a transação for considerada válida, ela será efetivada nos servidores da rede fixa, senão ela será abortada.

O PVML é baseado na verificação do número de versão dos objetos. Nesse protocolo, cada objeto matriz na rede fixa possui um número de versão. O número de versão de um objeto só é incrementado quando uma transação que modificou o estado do objeto é efetivada. Quando um objeto matriz é copiado para a máquina onde a transação se encontra, o seu número de versão também é copiado. Os números de versão das cópias em cache permanecerão os mesmos até que a transação que está acessando as cópias seja efetivada nos servidores de objetos da rede fixa.

O algoritmo usado pelo PVML verifica se cada objeto em cache utilizado pela transação no modo L continua sendo válido. Um objeto em cache é válido quando seu número de versão é igual ao número de versão do objeto matriz da rede fixa. O algoritmo de validação utiliza as seguintes definições:

(i) T_x é a transação que está sendo validada;

(ii) $WriteSet$ é o conjunto de objetos participantes de T_x que tiveram os seus estados modificados pela mesma;

(iii) $ReadSet$ é o conjunto de objetos participantes de T_x que não tiveram os seus estados modificados durante a execução da mesma;

(iv) $ValidationSet$ - conjunto dos objetos em cache que precisarão ser validados;

(v) Vo_i é um objeto pertencente ao conjunto $ValidationSet$;

(vi) Ro_i é um objeto matriz da rede fixa a partir do qual Vo_i foi originado quando copiado para a cache;

(vii) $Card(A)$ é a cardinalidade do conjunto A;

Protocolo de Validação do Modo Local (PVML):

```
ValidationSet ← ReadSet ∪ WriteSet;
i ← 1;
n ← Card(ValidationSet);
valid ← true;

Enquanto (valid = true) and (i ≤ n) faça
    Se (Ro_i ∈ ReadSet) Então
        Ro_i.setLock(read);
    Senão (Ro_i ∈ WriteSet)
        Ro_i.setLock(write);
    Fim Se
    Se (Vo_i.version ≠ Ro_i.version) Então
        valid ← false;
    Fim Se
    i ← i + 1;
Fim Enquanto
```

No algoritmo PVML, o resultado final da variável *valid* indicará se a transação passou ou não no processo de validação. Em suma, os modos de operação provêem um leque de alternativas para transações que executam em ambientes de computação móvel que vão desde o acesso remoto tradicional (modo R) com o uso de controle de concorrência pessimista ao acesso local com controle de concorrência otimista (modo L).

4.1.4 Aplicabilidade

A seguir são descritos alguns exemplos em que o uso dos diferentes modos de operação traria benefícios às transações:

Modo R – quando as características da conexão entre a máquina onde a transação se encontra e os servidores de objetos forem satisfatórias em questões como, por exemplo, largura de banda, custos e taxas de falhas; e/ou quando os requisitos de consistência da transação requerem acesso direto aos objetos remotos com controle de concorrência *two-phase locking*.

Modo LR – quando as características da conexão não forem satisfatórias como, por exemplo, alto custo da comunicação, desconexões freqüentes ou para economizar o uso de energia usada na comunicação sem fio; e/ou quando os requisitos da transação exigem a garantia de que as operações executadas localmente sejam sempre efetivadas nos servidores de objetos.

Modo L – quando as características da conexão não forem satisfatórias e para reduzir custos com comunicação ou reduzir consumo de energia; e/ou quando é aceitável para aplicação o seu cancelamento no caso de ser considerada inválida pelo PVML; e/ou quando a transação não tem como honrar o prazo de expiração imposto pelo modo LR. A mudança entre modos de operação pode ocorrer durante a execução da transação como medida de adaptação a mudanças ocorridas no ambiente.

5 Provas de Corretude

Esta seção apresenta provas de corretude para cada um dos modos de operação do controle de concorrência apresentado. Essas provas demonstram que a flexibilidade provida por esse controle de concorrência não compromete as regras de serialização de transações.

5.1 Modo Remoto

O *Teorema 1* mostra que o modo Remoto (R) provido pelo controle de concorrência apresentado garante a serialização de transações.

Teorema 1. Transações que executam sob o modo de operação R são serializáveis.

Prova) As transações que executam sob o modo de operação R estão sujeitas ao controle de concorrência *two-phase locking*. Sendo assim, esse teorema é comprovado pelo *Lema 1*.

Lema 1. Transações que executam sob o controle de concorrência *two-phase locking* são serializáveis.

Prova) A prova desse lema se encontra em [1].

5.2 Modo Local-Remoto

O *Teorema 2* mostra que o modo Local-Remoto (LR) provido pelo controle de concorrência apresentado garante a serialização de transações. A demonstração desse teorema utiliza a

Definição 1.

Definição 1. O processo de cópia de objetos remotos para a cache é realizado como uma transação somente leitura chamada transação de caching (TC). Como uma TC executa sob o controle de concorrência *two-phase locking*, ela sempre é serializável.

Teorema 2. Transações que executam sob o modo de operação LR são serializáveis.

Prova) Assuma que existe uma transação T_1 não serializável que executou sob o modo LR. Isso implica que T_1 se envolveu em um ciclo no grafo de serialização (GS) [1], ou seja, as arestas $T_1 \rightarrow T_2$ e $T_2 \rightarrow T_1$ estão presentes em GS. Considere que o item de dado x foi acessado por T_1 durante a sua execução. *Caso 1* – Considere que $T_1 \rightarrow T_2$ existe, então T_2 só acessou o item de dado x depois da execução de T_1, pois o acesso a x é exclusivo de T_1 durante toda a sua execução. Para que a aresta $T_2 \rightarrow T_1$ também exista no grafo, T_1 teria que executar uma operação conflitante depois do início da execução de T_2, o que é impossível, pois T_1 finalizou a sua execução antes de T_2 iniciar. *Caso 2* – Considere que $T_2 \rightarrow T_1$ existe, então T_2 já terminou sua execução, pois só assim T_1 consegue obter acesso exclusivo aos itens de dados acessados por T_2. Para que a aresta $T_1 \rightarrow T_2$ também exista é necessário que T_2 tenha executado uma operação conflitante depois do início da execução de T_1, o que é impossível, pois T_2 finalizou a sua execução antes do início de T_1. Os casos 1 e 2 contradizem a afirmação de que existe uma transação T_1 no modo LR em um ciclo no *GS*.

5.3 Modo Local

O *Teorema 3* mostra que o modo Local (L) provido pelo controle de concorrência apresentado garante a serialização de transações. A prova do *Teorema 3* utiliza o *Lema 2*.

Lema 2. Se uma transação T_1 executa sob o modo L, não existe ciclo no grafo de serialização tal que $T_2 \rightarrow T_1 \rightarrow T_2$.

Prova) Suponhamos que T_1 executou sobre o modo de operação L e existe um ciclo $T_2 \rightarrow T_1 \rightarrow T_2$. Como a aresta $T_2 \rightarrow T_1$ está presente no grafo de serialização então também existe a aresta $T_2 \rightarrow TC_1$ já que TC_1 é a transação de caching de T_1. Como TC_1 é somente leitura, a aresta $T_2 \rightarrow TC_1$ só pode ter sido originada do conflito $w_{T_2}(x) - r_{TC_1}(x)$. Segundo o algoritmo *two-phase locking*, $r_{TC_1}(x)$ só pode suceder $w_{T_2}(x)$ se T_2 já tiver sido efetivada. Como T_2 foi efetivada antes da execução de TC_1 e T_1 só executa depois de TC_1, não pode haver nenhuma operação de T_2 durante a execução de T_1. Sendo assim $T_1 \rightarrow T_2$ não existe e isso é uma contradição à afirmação de que existe um ciclo do tipo $T_2 \rightarrow T_1 \rightarrow T_2$.

Teorema 3. Se uma transação T_1 que executou sob o modo de operação L for considerada válida pelo algoritmo $PVML$ então T_1 é serializável.

Prova) Vamos supor que T_1 foi considerada válida pelo algoritmo $PVML$, porém T_1 não é serializável. Isso implica que T_1 se envolveu em um ciclo no grafo de serialização. Ciclos do tipo $T_2 \rightarrow T_1 \rightarrow T_2$ são descartados pelo Lema 2, então só resta observar os do tipo $T_1 \rightarrow T_2 \rightarrow T_1$. Como T_1 executa sob cópias dos dados em cache, a aresta $T_1 \rightarrow T_2$ só pode ter sido originada das operações conflitantes do tipo $r_{T_1}(x) - w_{T_2}(x)$, em que $r_{T_1}(x)$

representa a leitura do item de dado x quando copiado para a cache. Depois que x foi copiado para a cache, a operação $w_{T_2}(x)$ foi executada nos servidores da rede fixa. Depois que T_1 finaliza a sua execução sobre os itens de dados em cache, a única forma de gerar uma aresta $T_2 \rightarrow T_1$ é se T_1 tiver modificado x em cache. Assim, essa modificação tem que ser propagada para o servidor gerando um conflito do tipo $w_{T_2}(x) \rightarrow w_{T_1}(x)$. Porém, se a operação $w_{T_2}(x)$ foi realizada enquanto T_1 executava sobre as cópias em cache, isso significa que o número de versão de x foi mudado na fase de efetivação de T_2. O algoritmo de validação verificará que o número de versão da cópia de x em cache utilizado por T_1 é diferente do número de versão de x no servidor da rede fixa invalidando assim T_1. Portanto não haverá o ciclo $T_1 \rightarrow T_2 \rightarrow T_1$ no grafo de serialização e uma transação considerada válida pelo *PVML* é sempre serializável.

6 Conclusão

Esse relatório apresentou um mecanismo de controle de concorrência híbrido que oferece três diferentes modos de operação dentre os quais uma transação pode optar por executar. Esse mecanismo mescla as abordagens otimista e pessimista de forma a melhor atender os requisitos das transações bem como melhor lidar com as características dos ambientes de computação móvel.

Foram propostos três diferentes modos de operação para esse controle de concorrência: local, remoto, local-remoto. O modo local oferece às transações um mecanismo de controle de concorrência totalmente otimista baseado em cópias de dados que são trazidas para a cache da máquina móvel. Nesse modo, transações podem executar sobre as cópias em cache independentemente da rede fixa. O modo remoto oferece a abordagem pessimista baseado no controle de concorrência *two-phase locking*. Nesse modo, o acesso a dados é feito diretamente onde eles estão localizados (rede fixa). O modo local-remoto oferece uma abordagem híbrida onde as transações acessam os dados em cópias em cache, porém os dados correspondentes na rede fixa são trancados para evitar inconsistências.

Cada um desses modos oferecem soluções de controle de concorrência que podem ser empregadas em diferentes situações dentro da computação móvel. Por exemplo, os modos local e local-remoto oferecem a possibilidade de execução de transações em unidades móveis desconectadas da rede fixa. Como diferença, transações no modo local passam por um processo de validação no final das suas execuções enquanto que no modo local-remoto não há necessidade de validação. Em contrapartida, no modo local-remoto há um prazo de expiração que deve ser seguido para que as transações não sejam abortadas. O modo remoto oferece a abordagem tradicional de controle de concorrência que é mais adequada quando o estado do ambiente móvel oferecem recursos mas sólidos como, conectividade estável e largura de banda favorável.

Por fim, foram apresentadas provas de corretude para cada um dos modos de operação propostos. Essas provas demonstram que transações que executam nesses modos de operação podem ser serializadas em uma ordem bem definida.

Referências

[1] Bernstein, P. A., Hadzilacos, V. and Goodman, N. *Concurrency Control and Recovery in Database Systems*, Addison Wesley, 1987.

[2] Coulouris, G., Dollimore, J., Kindberg, T. *Distributed Systems: Concepts and Design*, Addison-Wesley, 1994.

[3] Dunham, M. H. and Helal, A. *Mobile Computing and Databases: Anything New?*, ACM SIGMOD Record, Vol. 24, No. 4, December, 1995.

[4] Forman, G. H. and Zahorjan, J. *The Challenges of Mobile Computing*, IEEE Computer, Vol. 27, April, 1994.

[5] Imielinski, T. and Badrinath, B. R. *Mobile Wireless Computing: Challenges in Data Management*, Communications of ACM, Vol. 37, No. 10, October, 1994.

[6] Pitoura, E. *Data Management for Mobile Computing*, Summer School on Mobile Computing, Jyvaskyla, 1998.

[7] Rocha, T. *Um Sistema de Transações Adaptável para o Ambiente de Comunicação Sem Fio*, Tese de Mestrado, Instituto de Computação-UNICAMP, 2004.

[8] Rocha, T. and Toledo, M. B. F. *A System of Adaptable Transactions for the Mobile Computing Environment*, In Proceedings of ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brasil, 2003.

[9] Satyanarayanan, M. *Fundamental Challenges in Mobile Computing*, Fifteenth ACM Symposium on Principles of Distributed Computing, Philadelphia, PA, May, 1996.