MAC5701 - Tópicos em Ciência da Computação

Técnicas de Replicação de Dados

Marcelo Brito dos Santos
brito@ime.usp.br
Orientador: Markus Endler

11 de julho de 2000

Sumário

Capítulo 1

Introdução

Criar e manter multiplas cópias de um dado vem se tornando uma necessidade dos sistemas distribuídos. A replicação é a chave para a disponibilidade e maior desempenho. Um backup do home dos usuários de uma rede é uma réplica, uma página no cache do seu browser é uma réplica, existem inúmeras facetas da replicação. Vamos nos deter na replicação que pode contribuir na tolerância a falhas de servidores de informação, como técnicas que possibilitem o mascaramento de falhas por meio da redundância dos dados.

Existem inúmeras formas de realizar replicação. O objetivo deste trabalho é apresentar um resumo das técnicas de replicação de dados, classificando-as segundo a forma de manutenção da consistência. Dividimos essas técnicas em consistência forte, relaxada ou fraca. Pretende-se com isto investigar e identificar as técnicas que melhor podem ser aplicadas ao projeto SIDAM(Sistemas de Informação Distribuídos para Agentes Móveis)[7].

Esta monografia está organizada da seguinte forma: no capítulo dois encontramos algumas definições utilizadas. O capítulo três aborda as técnicas de consistência forte. O capítulo quatro apresenta as técnicas de consistência fraca. O capítulo cinco apresenta as técnicas de consistência relaxada. Por fim, uma breve conclusão sobre os protocolos estudados e a bibliografia utilizada.

Capítulo 2

Replicação

A replicação de dados pode prover tolerância a falhas. A replicação de um dado pode mascarar falhas de réplicas e possibilita a reconfiguração do sistema para melhor atender a estas ocorrências. Replicação também provê maior desempenho do sistema, distribuindo-se o acesso ao dado entre as réplicas permite-se uma maior disponibilidade do dado. Protocolos de replicação são utilizados para manter a consistência entre as réplicas que representam um mesmo dado. No entanto, possui limitações quando se diz respeito a propagação de estados, pois o custo é alto na manutenção da consistência entre as réplicas.

Chamamos de réplicas consistentes aquelas que estão de acordo com a representação consistente do dado. Na teoria de serialização após qualquer operação, um dado deve ser levado de um estado consistente para outro. Essa propriedade nos leva a considerar que qualquer execução sequencial de operações sobre este dado é correta. Então uma forma de mantermos réplicas consistentes é serializar as operações sobre elas.

Um protocolo de replicação com *consistência forte* entre réplicas garante que todas as cópias (de um mesmo item de dado) estão sincronizadas entre si[1]. Este protocolo, que garante que as réplicas são cópias do dado, normalmente também acrescenta o custo da sincronização das réplicas. Portanto, a consistência forte entre as réplicas pode comprometer o desempenho do sistema.

Protocolos de consistência fraca não utilizam sincronização, deixando as réplicas divergirem do dado representado. Com isto, há maior disponibilidade de réplicas sem comprometimento do desempenho do sistema. Em computação móvel a consistência fraca é uma característica necessária, devido a eventual existência de réplicas temporariamente "desconectadas".

A *Consistência relaxada* aceita um certo grau de inconsistência, como em consistência fraca, mas possibilita a partir de critérios de consistência, a reconciliação entre réplicas trazendo-as a um estado consistente, como em consistência forte.

Uma réplica pode ser vista como uma cópia ou uma representação do dado. *Cópias* são todas as réplicas de um dado que representam o estado atual do dado replicado. Uma *representação* é uma réplica que semanticamente representa um dado, mas pode não estar no mesmo estado que ele[1].

Uma representação é dita inconsistente, ou divergente, se não está no mesmo estado que o objeto representado. Em operações não coordenadas, ou seja, não serializadas a manutenção de réplicas consistentes é bem mais complexa. Chama-se então de *convergência* a garantia de que todas as operações, feitas sobre as réplicas, serão executadas e as réplicas serão levadas a um estado consistente.

A disponibilidade de um recurso no sistema é medida pela possibilidade de acesso a este recurso. A replicação possibilita o aumento da disponibilidade, que atinge seu grau máximo nos protocolos de consistência fraca. Os protocolos de consistência forte bloqueiam as cópias durante as operações, o que diminui a disponibilidade do dado, mas garante a consistência das réplicas. É necessário ponderar sobre disponibilidade e consistência visando as reais necessidades da aplicação.

Protolocos de controle de concorrência são utilizados para restringir a execução concorrente, nas réplicas, de operações que podem levar o dado representado a um estado inconsistente. Os protocolos de controle de concorrência são divididos em duas categorias, otimistas e pessimistas. Os protocolos otimistas não se preocupam com a manutenção de consistência, deixando certa inconsistência, postergam a reconciliação das réplicas e apostam que não será complexa a reconciliação. Já os protocolos pessimistas, prevendo que tudo vai gerar inconsistência e que a reconciliação será NP-Completa, não permitem execuções não serializáveis.

A utilização de protocolos de consistência forte garante que as réplicas são cópias do dado representado. Com isso temos a idéia de *visão global*, onde as réplicas são representações lógicas de um item de dado. As operações são feitas neste dado lógico e não nas réplicas, assim todas continuam consistente com o dado representado. Em contraste, a utilização de protocolos de consistência fraca introduz réplicas inconsistentes e o conceito de *visão local* do dado. Na consistência eventual, qualquer réplica é uma representação local do dado, podendo responder por ele, mas diferir de outra visão local. Este conjunto de visões locais pode gerar inconsistências globais.

Capítulo 3

Consistência Forte

Uma regra é primordial na manutenção de consistência forte entre réplicas: cada execução concorrente deve equivaler a uma serial. A idéia de *one-copy seriability*(1-SR) faz com que todas as cópias sofram execução serializada para um único item. Assim, é garantida a consistência de todas as cópias tendo-se o resultado propagado entre elas. Toda operação de leitura e escrita em um dado consistente é atomica e leva a um estado consistente.

As técnicas de consistência forte podem ser classificadas em três abordagens: Cópia corrente, votação e híbrida[1]. Em cópia corrente atualizações são aplicadas em todas as réplicas acessíveis, enquanto que a leitura pode ser feita em apenas uma réplica. Nas técnicas de votação, cada réplica possui o poder de voto. Cada operação recolhe os votos e realiza um consenso dos votos. Por fim, temos técnicas que são híbridos dessas duas.

Muitas dessas técnicas não são escaláveis para um grande número de réplicas requerendo muita comunicação, sendo inviáveis em grandes sistemas distribuídos. A cópia corrente requer participação total e a votação requer muita comunicação entre as réplicas nas operações. É importante salientar que todo protocolo de consistência forte bloqueia as réplicas envolvidas em operações de escrita.

3.1 Cópia Corrente

Cópia corretente tem como vantagem a alta disponibilidade em transações de leitura. Como desvantagem não tolera falhas de comunicação e nem partições de rede. Necessitando de redes de alto desempenho e comunicação confiável.

Este protocolo se utiliza de um coordenador para operações de escrita.

As técnicas classificadas como cópia corrente são também chamadas de *Read-One-Write-All*(ROWA). A ROWA é a mais genérica das abordagens. As demais abordagens baseadas em cópia corrente são variações deste modelo. Este algoritmo converte uma operação de leitura no dado em uma única operação de leitura em qualquer uma das cópias. Sendo n o número de cópias, uma operação de escrita é convertida em n escritas, uma em cada cópia. Durante a escrita todas as cópias são bloqueadas, em caso de falha de um site, todas ficam bloqueadas até a recuperação da falha no site. Baseado neste caso, surgiu uma variação, o ROWA-A.

3.1.1 Read One/Write All Available

Nesta abordagem, uma operação de escrita já não requer que todas as cópias do dado respondam, mas somente as cópias acessíveis. Um site que falhou, quando volta, fica fora do algoritmo até que copie o estado atual de alguma réplica participante. Neste momento, a cópia que falhou se torna novamente *available*. As escritas em cópias que não respondem, chamadas de *Missing Writes*(MW), são armazenadas pelo coordenador. Antes de confirmar uma operação, o coordenador inicia um protocolo de validação em duas etapas:

1. *Missing Writes Validation*, onde determina se todas as cópias que causaram MW estão inacessíveis. O coordenador envia um UNAVAIL para todos os sites que tem MW. Se algum responder, a operação é abortada por inconsistência de uma cópia. Caso contrário é realizado o próximo passo.

2. *Access Validation*, determina se todas as cópias que foram lidas, ou escritas, estão acessíveis. O coordenador envia um AVAIL para todos os sites que contenham cópias acessíveis. Caso todos respondam, a operação é terminada com sucesso.

Esta abordagem tem algumas desvantagens. Em repetidas tentativas de escrita em sites com falhas gera-se um grande volume de requisições armazenadas no MWL.

3.1.2 Primary-Copy

Primary-copy é a mais utilizada das abordagens ROWA. O modelo básico garante que toda leitura e escrita é feita em uma cópia do dado, chamada *cópia corrente* ou *primary*, as demais cópias são chamadas de *backups*. Ou seja, as operações sofrem serialização na cópia corrente. A escrita é feita na cópia corrente e é propagado imediatamente para as backups e só então é dado o sucesso da operação. Quando a cópia corrente falha, uma backup assume seu papel, por meio de votação ou por pré-determinação.

3.1.3 True Copy Token ROWA

Aqui temos um ROWA com a idéia que a cópia corrente é a cópia que possue o token, ou seja, quem conseguir o token é a primary e as demais são backups, seguindo uma analogia a primary-copy. Nesta abordagem, cada conjunto de réplicas que representam um dado lógico possui um token exclusivo e vários compartilhados. Uma operação de escrita requer um token exclusivo, enquanto que as de leitura utilizam os compartilhados. Quando uma cópia recebe uma operação de escrita, ela localiza e solicita o token exclusivo. Em caso de falha no site que contenha o token exclusivo, todas as réplicas ficam indisponíveis para escrita. Uma alternativa é a utilização de um protocolo de regeneração de token.

3.2 Votação

Uma vantagem dos protocolos de votação é possibilidade de reconfiguração dinâmica e a tolerância a falhas de comunicação e de site, pois não precisa de intervenções pra tratar particionamento da rede ou junção de partições. Para manter esta tolerância é necessária muita comunicação para validação das operações, o que pode vir a ser uma desvantagem.

Votação, também conhecido como *Quorum Consensus*(QC), garante a escrita em apenas um subconjunto (quorum de escrita) dos sites acessíveis, bem como leituras em um subconjunto (quorum de leitura) que tem sobreposição de réplicas com o quorum de escrita. Esta interseção garante que a leitura sempre responderá com o resultado da última escrita. Assumindo que as réplicas estão divididas em quoruns, elas entram em consenso para responder perguntas (operações) feitas a estes quoruns.

3.2.1 Weight Majority QC

Nesta abordagem, no lugar de associarmos um simples voto por site, cada cópia de um dado tem um peso não-negativo. A soma do peso de todas as cópias é u. Há um valor mínimo de leitura r e um mínimo de escrita w, tal que $r + w > u$ e $w > u/2$. Assim um quorum de leitura é qualquer conjunto de cópias com peso maior ou igual a r, ou w. Neste protocolo é utilizado versões para determinar a cópia corrente. Cada leitura é convertida em um conjunto de leituras no quorum. Cada cópia retorna a versão e o resultado. A cópia de maior versão será a cópia corrente e o resultado da operação será o resultado desta cópia. Uma cópia que falhou, mesmo após sua recuperação, não retorna ao quorum até que tenha sido atualizada com o último valor do quorum.

3.2.2 QC de Objetos

Esta abordagem difere das baseadas em valores, uma vez que é necessário o conhecimento da semântica do dado replicado. Algumas cópias do objeto não necessariamente tem o mesmo valor, por isso não chamaremos de cópias e sim representações do objeto. Utilizando-se de votação, temos que uma operação afeta um subconjunto das réplicas, onde estas guardam log das operações. Estas representações do objeto, quando juntas, geram um log completo com todos os eventos que afetaram o objeto. Daí todas sofrem as operações levando a uma completa representação do estado atual do objeto.

3.2.3 QC com Estruturas

Alguns algoritmos enfatizam a diminuição na comunicação entre as réplicas, além da tolerância a falhas. Uma forma de fazer isto é diminuindo o número de sites que devem manter consistência mútua. Estes novos protocolos tentam reduzir pela metade o número de participantes em votações, para isto é necessário alocar os sites em estruturas lógicas, como árvores. Neste modelo existem vários quoruns mínimos de leitura e escrita. Um conjunto destes quoruns é dita *quorum set*. Ainda vale, aqui, a restrição que os quoruns de escrita e leitura devem possuir interseções não nulas. As mensagens são entregues percorrendo a estrutura lógica de localização da réplica. Em caso de falha de um site há uma restruturação.

3.2.4 QC em Grades

Este algoritmo mapeia os sites em uma estrutura lógica de grades, ou matriz. Conjuntos de sites que armazenam réplicas são logicamente alocados em uma grade com M linha de N sites, o que configura uma grade(Fig. 3.1). Neste método, o quorum set de leitura contém exatamente um dos sites de cada coluna, ou um dos sites de cada linha. O quorum de escrita consiste em um grupo de sites onde há representantes de cada linha e coluna da grade. Com isto, garantimos que as alterações atingirão todos os quoruns e ainda assim temos alta disponibilidade, pois há membros do quorum (linha ou coluna) que não foram bloqueados para escrita.

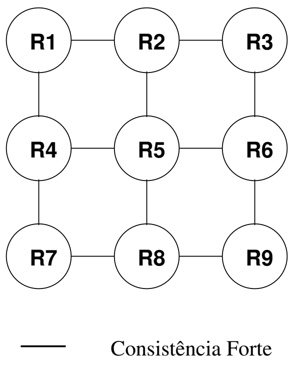

Figura 3.1: Um exemplo de grade com 9 sites de réplicas.

3.3 Híbrido ROWA/QC

Mostrou-se que o quorum consensus é caro para redes confiáveis, pois suas trocas de mensagens são desnecessárias neste cenário. Mostrou-se ainda que a abordagem de cópia corrente é eficiente, mas precisa de redes sem falhas, o que não é garantido na maioria das aplicações. O híbrido de cópia corrente e votação propõe um método para reduzir este custo, adotando ROWA durante períodos confiáveis de operação e chaveando para QC durante períodos onde existam falhas de sites ou comunicação. As réplicas trabalham em ROWA-A no *modo normal* e quando percebem que uma réplica não responde chaveiam para o *modo de falha*. Neste modo, o método usado é um baseado em votação. A presença de MW, operação não efetuada na réplica, indica que alguma cópia não está consistente com as demais. É o momento de chavear para QC. Para armazenar o MW, uma lista(MWL) de MW com versões é associada para cada réplica. Quando uma réplica falha, o MW fica não nulo em todas as réplicas participantes, a partir daí todas guardam os MWs e o coordenador faz o chaveamento para o modo de falha. Quando a réplica for recuperada, há uma transação de cópia, onde qualquer uma das réplicas repassa as operações que esta não sofreu e eliminasse a MWL. Isto é possível pelo uso de versões. Quando a lista é nula o coordenador faz o chaveamento para o modo normal e voltasse à técnica de cópia corrente.

Capítulo 4

Consistência Fraca

Replicação é utilizada em sistemas distribuídos para aumentar a disponibilidade e performance. Tradicionalmente, os requisitos de consistência em banco de dados distribuídos são muito rigorosos, o que é um resultado da transparência da distribuição e da transferência do paradigma de transações ACID para o ambiente distribuído. Com o passar dos anos, muitas abordagens vem propondo o relaxamento dos requisitos de consistência forte, comuns até então. O termo consistência fraca pode ser usado para qualquer abordagem que relaxa as restrições de consistência forte[2]. Neste trabalho, chamaremos de consistência fraca apenas protocolos que não especificam consistência e protocolos que fazem propagação epidêmica[3], que garantem consistência eventual. Vamos comentar sobre o protocolo de atualizações independentes, o TSAE e o *identity connections*, que é uma tentativa de consistência global, com as vantagens da consistência fraca.

4.1 Atualizações Independentes

Permitir atualizações independentes garante maior autonomia para a réplica e por consequência o máximo em disponibilidade. No entanto as aplicações que podem se valer desta abordagem são limitadas por fatores como:

- Inconsistências globais não são resolvidas por rotinas de resolução de conflitos locais.
- Geralmente, uma aplicação que necessite de reconciliações exigiria um protocolo robusto e caro de reconciliação de réplicas, tornando inviável

esta alternativa.

4.2 Timestamped Anti-Entropy(TSAE)

É um dos mais sedimentados protocolos de consistência fraca. Provê consistência eventual, confiável e garantias de ordem nas operações[4]. O que significa que todas as réplicas que representam o dado logicamente irão receber todas as operações. Só é garantido que receberão em um tempo finito, indeterminado.

O protocolo TSAE faz com que pares de réplicas troquem mensagens periodicamente, chama-se este procedimento de *sessão de anti-entropia*. A sessão entrega conjuntos de mensagens que seguiram imediatamente após a última sessão recebida, então não há "*gaps*" na seqüência de mensagens. Esta propriedade é muito útil para difusão de mensagens em grupos de réplicas.

Um protocolo de consistência eventual é confiável se toda réplica eventualmente recebe uma cópia de toda mensagem enviada para o grupo[3]. O TSAE garante esta restrição. Ainda possui um algoritmo simples para remover mensagens do registro de mensagem, baseado nos *acknowlegments*, o que não interfere com a propagação da mensagem.

O protocolo mantém dois vetores de *timestamp*. Um guarda quais mensagens foram recebidas, enquanto que o outro acompanha os acknowlegments dos outros processos. Os vetores contêm um timestamp para cada operação nas réplicas. Os vetores possibilitam uma forma compacta de determinar o quanto divergente está uma replica da outra. Isso permite a transmissão somente de mensagens não recebidas, apenas uma vez para cada réplica. O vetor de acknowlegments provê um jeito eficiente de checar se todas as réplicas receberam a mensagem. De tempos em tempos é realizada a sessão de anti-entropia. Os envolvidos detectam as mensagens que o outro detem e que não recebeu e trocam mensagens. Ao final do processo, o vetor de timestamp é ajustado para refletir a atualização. O vetor de acknowlegments é atualizado implicitamente.

4.3 Identity Connections

O conceito de *identity connections*(IC) relaciona atributos pertencentes a diferentes relações ou fragmentos alocados em diferentes sites. O termo IC é usado para caracterizar dados que são armazenados em bancos de dados diferentes, mas relacionados por uma restrição de integridade que especifica consistência e dependência entre os dados. Em replicação uma IC entre diferentes réplicas significa que elas são réplicas idênticas, cópias. A noção de primary/backup copy é usada para especificar as restrições de consistência. Uma atualização na cópia corrente é propagada para as cópias secundárias. Este método fornece limitada tolerância a falhas, pois só é confiável utilizando primary copy, o que seria um método de consistência forte, inviável para esta abordagem que visa consistência fraca e eventual das réplicas. Uma forma de garantir boa consistência é utilizandar *Epsilon-serializability*(Seção 5.2) propiciando uma certa inconsistência e restauração das réplicas. Vale comentar que, sendo uma abordagem para redes de larga escala, uma abordagem de consistência forte é muito cara. A proposta do modelo de utilizar consistência fraca gera inúmeros problemas, pois busca manter uma visão global do dado, mas não quer "pagar" pelo protocolo que garante uma visão global. Uma falha é a possibilidade de duas cópias sofrerem alterações e propagarem ao mesmo tempo. Para ambas, elas são primárias e sobrescreverão sobre as propagações da outra, gerando inconsistência real nas réplicas. Isto se deve ao fato de não utilizar, por exemplo, uma escrita exclusiva o que levaria esta abordagem de volta a um modelo de consistência forte.

Capítulo 5

Consistência Relaxada

Um meio termo, ou híbrido das técnicas de consistência forte e fraca é o uso de consistência relaxada, ou seja, relaxa-se a consistência mútua entre as réplicas. Protocolos de consistência relaxada[1] são caracterizados pela especificação de momentos de reconciliação, relaxamento na seriação das operações ou uso de consistência insular.

Os gatilhos para reconciliações podem ser divididos em duas categoria, de acordo com a especificação da dimensão temporal ou espacial de consistência da réplica[2]. A *dimensão espacial* descreve o grau de consistência ou a qualidade do dado de uma certa réplica. A *dimensão temporal*, por outro lado, especifica quando essa qualidade do dado é necessária. As restrições temporais podem ser eventos ou intervalos de tempo. Os eventos indicam um certo ponto no tempo e são responsáveis por checar a consistência e atualizar as réplicas. Quanto as restrições espaciais, elas podem ser classificadas em consistentes, fracas e sem restrições. As consistentes garantem que a réplica é uma cópia do dado representado, enquanto que a fraca tolera uma distância máxima do dado representado. A distância do dado logicamente representado pode ser especificada de diferentes formas, usando distância entre versões ou valor de divergência.

A seguir temos alguns protocolos de consistência relaxada, descritos em [3].

[1] Alguns autores, como Lenz[2], não utilizam o termo consistência relaxada. Considerando protocolos de consistência relaxada também como protocolos de consistência fraca.

5.1 Quasi-Copies

O protocolo Quasi-Copies permite especificar limites de divergência entre as réplicas. A convergência das réplicas é garantida desde que escritas sejam feitas apenas na cópia corrente. Através de um predicado de convergência há uma propagação contínua para as réplicas que divergem.

5.2 Epsilon Seriability

Epsilon Seriability é uma técnica de controle de concorrência, não foi desenvolvida especificamente para sistemas distribuídos. É uma extensão da seriabilidade estrita que permite controlar inconsistências. Ao invés de reforçar requisitos da seriabilidade em todas as transações, como se usasse um protocolo de travamento bifásico, a serialização é aplicado somente às operações de escrita.

Em se tratando de protocolo de controle de réplicas, utiliza propagação assíncrona. Pode ser classificado como consistência relaxada por especificar operações a serem serializadas e operações não coordenadas. As escritas são realizadas na mesma ordem em todas as réplicas. Cabe a aplicação limitar o quanto inconsistente poderá ficar uma réplica. Como utiliza-se de bloqueios para serialização, não é uma boa abordagem para serviços com grande número de réplicas.

5.3 Lazy-Replication

Em Lazy-Replication as aplicações podem especificar exatamente quais as relações causais que devem ser reforçadas, consistência fraca é conseguida não oferecendo especificações. Pode-se ainda especificar relações causais com eventos de fora do grupo de réplicas. Há maior comunicação neste protocolo, pois as escritas são realizadas em *one-way* com mensagens *gossip*, semelhante a sessão de anti-entropia, mas de desempenho inferior, por não ser bi-direcional sendo redundante em operações já recebidas pela réplica.

5.4 Virtual-Primary-Copy

Diferente do modelo de cópia corrente padrão, introduz-se aqui a abstração de *primary copy virtual*[2]. Termo utilizado para o conjunto de cópias que estão consistentes com o dado logicamente representado pelas réplicas do grupo, também chamado de ilha de consistência, ou consistência insular.

Réplicas podem entrar ou sair da ilha de consistência de acordo com o requerido pela aplicação. A réplica que entra na ilha recebe o valor da cópia corrente. Todos os outros membros da ilha são informados do novo membro, pois ele participará das sincronizações na ilha.

As operações realizadas na ilha de consistência são serializadas de acordo com 1-SR[1], mas especificamente seguindo o protocolo *ROWA-Available* (Seção 3.1.1), por possibilitar leituras locais em qualquer cópia, enquanto a ilha está estável. Ocorrendo uma falha em um membro da ilha o algoritmo chaveia para um protocolo de votação (Seção 3.2). Se uma escrita é feita em um membro da ilha, todos os integrantes da ilha são sincronizados com a cópia corrente. Aplicações que acessam as cópias inconsistentes podem requisitar escritas, também. No entanto atualizações fora da ilha, são chamadas *blind writes*. Blind writes são escritas onde o valor corrente da cópia é desconhecido, uma atualização não é um blind writes, pois depende do valor corrente do dado. O blind write é redirecionado para a ilha de consistência, onde a operação é realizada como se fosse requisitada diretamente a ilha. Operações bem sucedidas na ilha são propagadas para as cópias de consistência fraca de acordo com a especificação de consistência nessas cópias. Assim, a ilha de consistência funciona como uma primary-copy das réplicas de consistência fraca(Fig. 5.1).

Vale ressaltar que é notório a impossibilidade de permitir a existência de ilhas de consistência vazias, nesta abordagem. Uma estratégia simples para manter a ilha não nula é impedir que uma réplica deixe a ilha, se esta for a única integrante da ilha.

5.5 Consistência Relaxada: Forte e Fraca

Algumas pesquisas visam integrar consistência forte e fraca em um mesmo framework[1]. Noha Adly propõe em seu framework operações $fast\_read$ e $fast\_write$ para consistência fraca, enquanto que as operações $slow\_read$ e $slow\_write$ são para consistência forte. Conflitos de consistência gerados por

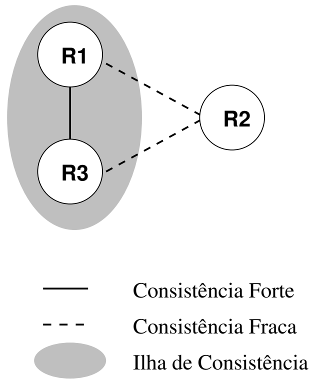

Figura 5.1: Um exemplo de Virtual-Primary-Copy com 3 réplicas.

operações "fast" são resolvidas em "off-line" e as aplicações devem tolerar algumas inconsistências neste caso.

Sean Liu amplia o trabalho de Adly, propiciando quatro formas de consistência para as aplicações. Podendo ser consistência forte(ou estrita), consistência insular, consistência fraca ou sem manutenção de consistência. A consistência insular mantêm convergencia de réplicas, mas permite leituras em réplicas que não estão na ilha consistente, ou seja, em réplicas inconsistentes. A consistência insular esta classificada entre a consistência fraca e a forte. Neste framework as réplicas são organizadas de forma hierárquica em uma estrutura de multiversionamento. Um meta-sistema processa as requisições e define a versão das réplicas que melhor atende. Atualizações são re-integradas a versão ao qual pertence e propagadas a cada nível, garantindo que as operações levam a réplica de um estado consistente para outro.

Capítulo 6

Conclusão

A consistência forte apresenta melhores resultados em pequenos grupos de réplicas, porque os algoritmos para manutenção da consistência global tem custo proporcional ao número de réplicas envolvidas[5]. A consistência fraca apresenta bons resultados em grandes grupos de réplicas, pelo fato de não sincroniza-las, nos garantindo apenas uma consistência eventual.

No projeto SIDAM, acreditamos que a utilização de consistência relaxada, garante um maior domínio na manipulação de dados que são atualizados constantemente, mas podem ser representados por réplicas divergentes durante um espaço de tempo, possibilitando níveis de consistência e maior disponibilidade do dado. A associação conseguida por esta técnica garante a manipulação de visões locais e, quando necessária, a reconciliação das réplicas em uma visão global. Dentre as abordagens de consistência relaxada, a abordagem de "virtual-primary-copy"[2] nos mostra as vantagens da utilização de consistência insular e um protocolo de interação entre a ilha e réplicas de consistência fraca.

No SIDAM esta sendo desenvolvido o Juggler[6], um gerenciador de grupos de réplicas, que possibilita a escolha dinâmica do tipo de consistência que será mantida entre as réplicas de um grupo. Este estudo propõe um protocolo de consistência relaxada, semelhante ao visto em [2], para ser inserido no serviço de grupos de réplicas gerenciado pelo Juggler.

Referências Bibliográficas

[1] A. A. Helal, A. A. Heddaya and B. B. Bhargava. *Replication Techniques in Distributed Systems*, Kluwer Academic Publishers, 1996.

[2] Lenz, Richard. *The "Virtual-Primary-Copy Approach" Compared to Other Approaches with Weak Consistent Data Replication*, University of Erlangen-Nuremberg, 1996.

[3] R. A. Golding. *Weak-Consistency Group Communication and Membership*, Ph.D. Thesis, University of California, december 1992.

[4] R. A. Golding, and D. D. E. Long. *Desing Choices for Weak-Consistency Group Communication*, UCSC-TR-92-45, University of California, october 1992.

[5] K. P. Birman. *A Review of Experiences with Reliable Multicast*, Cornell University, 1999.

[6] M. A. de Moura and M. Endler, *Juggler: A Configuration Management Service for CORBA Object Groups*, SBRC 2000, May 2000.

[7] D. Silva, M. D. Gubitoso and M. Endler. *Sistemas de Informação Distribuídos para Agentes Móveis*, SEMISH'98, August 1998 .