Trabalho Prático 3 (TP 03) – Disciplina de Sistemas Distribuídos

Neste trabalho, o grupo de alunos (3 no máximo) irá implementar alguma estratégia de replicação e tolerância a falhas. Não há utilização de middlewares neste TP. Qualquer linguagem será aceita. Não serão aceitas soluções apenas multicore.

Preliminares: Iremos usar o que foi feito no TP2, portanto há 5 clientes que conhecem seus elementos do Cluster Sync, totalizando 5 elementos do Cluster Sync. No TP2, quando um elemento do Cluster Sync pode entrar na seção crítica, o mesmo aguarda por 0.2 a 1 segundo, simulando estar acessando R. Já no TP3 isto ocorrerá de maneira real, portanto haverá um cluster para acessar R chamado Cluster Store. Cada elemento do Cluster Sync pode acessar qualquer elemento do Cluster Store, podendo inclusive variar de elemento do Cluster Store a cada entrada na seção crítica. O cluster Store possui 3 elementos.

Protocolo 1:

O protocolo mais simples baseado em primário e que suporta replicação é aquele em que as operações de escrita precisam ser enviadas para um único servidor fixo. Operações de leitura podem ser executadas no local. Esses esquemas também são conhecidos como **protocolos de primário e backup** (Budhiraja et al., 1993). Um protocolo de primário e backup funciona como mostra a Figura 7.19. Um processo que quer realizar uma operação de escrita, em um item de dados x, envia essa operação para o servidor de primários para x. O servidor primário executa a atualização em sua cópia local de x e, na seqüência, envia a atualização para os servidores de backup. Cada servidor de backup também efetua a atualização e envia um reconhecimento de volta ao servidor primário. Quando todos os servidores de backup tiverem atualizado sua cópia local, o servidor primário envia um reconhecimento de volta ao processo inicial.

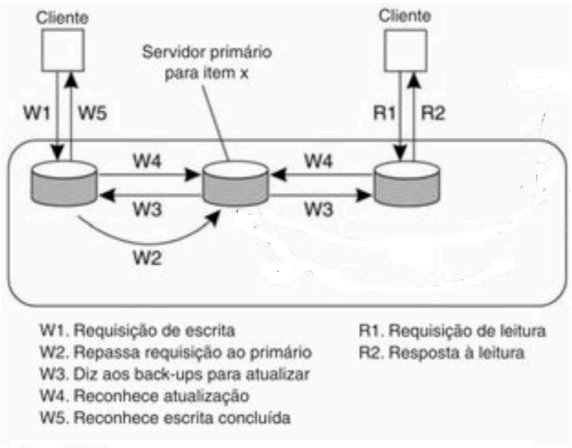

Figura 7.19 Princípio de um protocolo de primário e backup.

Protocolo 2:

Uma variante dos protocolos de primário e backup é aquela em que a cópia primária migra entre processos que desejam realizar uma operação de escrita. Como antes, sempre que um processo quer atualizar o item de dados x, ele localiza a cópia primária de x e, na seqüência, move essa cópia para sua própria localização, como mostra a Figura 7.20. A principal vantagem dessa abordagem é que múltiplas operações sucessivas de escrita podem ser executadas no local enquanto processos leitores ainda podem acessar sua cópia local. Contudo, só se pode conseguir tal melhoria se for seguido um protocolo não bloqueador pelo qual as atualizações são propagadas para as réplicas após o servidor primário ter concluído as atualizações realizadas localmente.

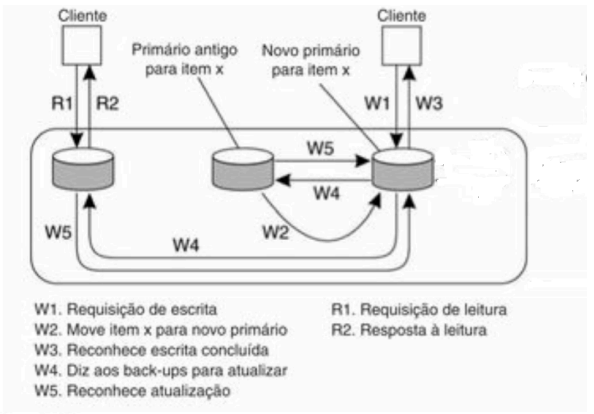

Figura 7.20 Protocolo de primário e backup no qual a cópia primária migra para o processo que quer realizar uma atualização.

Protocolo 3:

Uma abordagem diferente para suportar escritas replicadas é usar **votação** como proposto originalmente por Thomas (1979) e generalizado por Gifford (1979).

Para ver como esse algoritmo funciona, considere a Figura 7.21(a), na qual $N_R = 3$ e $N_W = 10$. Imagine que o quórum de escrita mais recente consistiu em 10 servidores, C a L. Todos eles obtêm a nova versão e o novo número de versão. Qualquer quórum de leitura subseqüente de três servidores terá de conter, no mínimo, um membro desse conjunto. Quando o cliente vir os números de versão, saberá qual é a mais recente e a adotará.

Na Figura 7.21(b) e (c), vemos mais dois exemplos. Na Figura 7.21(b) pode ocorrer um conflito escrita–escrita porque $N_W \leq N/2$. Em particular, se um dos clientes escolher $\{A,B,C,E,F,G\}$ como seu conjunto de escrita e um outro cliente escolher $\{D,H,I,J,K,L\}$ como seu conjunto de escrita, então estaremos claramente em dificuldades porque ambas as atualizações serão aceitas sem detectar que, na verdade, estão em conflito.

A situação mostrada na Figura 7.21(c) é de especial interesse porque fixa N_R em um, o que possibilita ler um arquivo replicado descobrindo e usando qualquer cópia.

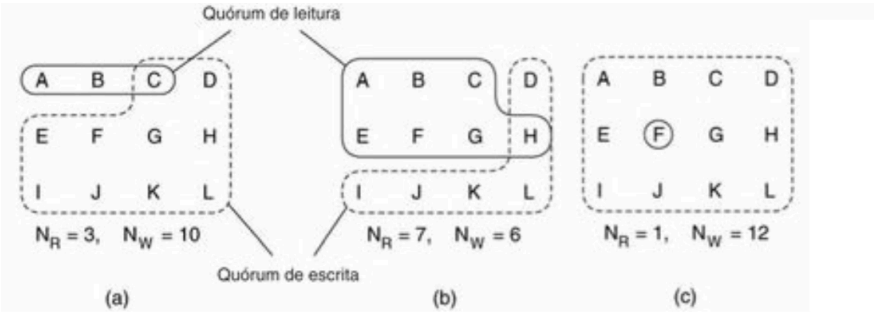

Figura 7.21 Três exemplos do algoritmo de votação. (a) Escolha correta de conjunto de leitura e de escrita. (b) Escolha que pode levar a conflitos escrita–escrita. (c) Escolha correta, conhecida como ROWA (lê uma, escreve todas).

O que ocorre se algum elemento do Cluster Sync ou Cluster Store falha?

Tipo de falha	Descrição
Falha por queda	O servidor pára de funcionar, mas estava funcionando corretamente até parar.
Falha por omissão	O servidor não consegue responder a requisições que chegam
Omissão de recebimento	O servidor não consegue receber mensagens que chegam
Omissão de envio	O servidor não consegue enviar mensagens
Falha de temporização	A resposta do servidor se encontra fora do intervalo de tempo
Falha de resposta	A resposta do servidor está incorreta
Falha de valor	O valor da resposta está errado
Falha de transição de estado	O servidor se desvia do fluxo de controle correto
Falha arbitrária	Um servidor pode produzir respostas arbitrárias em momentos arbitrários

Tabela 8.1 Diferentes tipos de falhas.

Iremos simular apenas falhas por omissão ou queda.

Cada grupo precisa implementar apenas uma das opções a seguir. Temporizadores nas comunicações podem ser necessários. Há a alternativa de envio periódico de mensagens de controle do tipo PING.

Opção 1) Elemento do Cluster Sync falha:

1.1) Elemento do Cluster Sync sem pedido de cliente

1.2) Elemento do Cluster Sync com pedido do cliente

1.3) Elemento do Cluster Sync na seção crítica

Opção 2) Elemento do Cluster Store falha:

1.1) Elemento do Cluster Store sem pedido do Cluster Sync

1.2) Elemento do Cluster Store com pedido do Cluster Sync

1.3) Elemento do Cluster Store com permissão de escrita