Abstract

Existe uma procura crescente de sistemas distribuídos de tempo-real fiáveis por toda a comunidade informática. São necessárias novas técnicas para estas aplicações perpetuarem face a alterações nas condições da rede. Este artigo tem como objectivo apresentar uma concretização de um servidor replicado. A ideia do artigo é apresentar uma possível solução de um sistema tolerante a faltas que suporte leituras e escritas sobre um registo, o qual contém um inteiro, e que replique um valor, caso tenha havido uma escrita, para os outros servidores.

1 Introdução

Ao longo dos tempos tem-se verificado um aumento da necessidade de especificar sistemas em termos de requisitos temporais ditados pelo ambiente. Estes são conhecidos como sistemas de tempo-real e a sua computação definida tanto em termos dos seus resultados lógicos como do tempo no qual eles são fornecidos. Há muitos exemplos de sistemas de tempo-real: sistemas de controlo do tráfego aéreo, sistemas de reservas de bilhetes on-line ou até mesmo jogos de computador em tempo-real. [1] Com o passar do tempo o campo computacional tem mutado a sua antiga faceta da computação pessoal para uma computação multipessoal extremamente prometedora. A distribuição tem penetrado na área de tempo-real não são por esse novo desafio como também pela necessidade de descentralizar, equilibrar a carga computacional, replicar e paralelizar a computação. Estas novas necessidades levaram ao aparecimento dos sistemas distribuídos de tempo-real como são hoje conhecidos. Nestes, as garantias temporais têm de ser fornecidas sobre um sistema de máquinas ligadas pela rede. Uma vez conseguido isso, as garantias devem ser associadas a outros atributos, tais como a modularidade, a separação geográfica, a independência das faltas e o balanço da carga, entre outros. [1] Embora os sistemas distribuídos de tempo-real pareçam uma óptima solução para estes novos requisitos, seria inútil considerar a sua existência apenas em ambientes perfeitos1 ou em ambientes onde as faltas pudessem ser ignoradas. Como solução a este problema surgiram os sistemas distribuídos de tempo-real fiáveis nos quais se tenta garantir o funcionamento dos sistemas distribuídos de tempo-real mesmo sobre situações faltosas.

2 Ler e Escrever

2.1 Modelo

Temos de ter em conta que um dos objectivos do trabalho é ter um grupo com servidores replicados, onde caso haja uma falha num dos processos, é garantido que o sistema não vai abaixo. Devemos assumir que haverão muito mais leituras que escritas. Aquando uma escrita, o valor escrito é difundido para todos os outros processos conhecidos do grupo (contando com tempo de envio das mensagens, tratamento de faltas (caso ocorram), sincronização dos servidores). A uma leitura é devolvido o valor guardado no registo sem ter que ser preciso difundir a mensagem para os outros processos do grupo, fazendo com que uma operação de leitura seja mais rápida que uma operação de escrita. Na operação de leitura, teremos de nos preocupar em balancear a carga pelos vários processos do grupo. Para o projecto a desenvolver, uma das soluções possíveis seria implementar uma solução centralizada. Após a recepção da confirmação de chegada de mensagens por parte de outros processos, poderiam congestionar a ligação com o servidor central e consequentemente provocar latência na rede. Outra desvantagem seria a falha do servidor central, sem que seja possível tolerar este tipo de falta.

2.2 Propagação de Mensagens

2.2.1 Propagação Best Effort

A escrita é propagada pela rede. Caso haja um cliente que aceda a uma outra réplica do grupo, poderá ainda ler o valor antigo do registo.

2.2.2 Propagação Atómica

A difusão atómica tem vindo a dar grandes problemas ao nível do *design* e de implementação de um sistema distribuído tolerante a faltas. O problema consiste em permitir que os processos que enviem mensagens pela rede, não só os enviem como se preocupem com a ordem de chegada das mensagens. A

solução para o modelo de difusão atómica tende para a não-recuperação de falhas de processos, ou seja, se um processo for abaixo, nunca voltará acima. Neste caso, será necessário abordar a solução de um modelo *crash-recovery*. Este modelo assume que se um processo tem uma falha, poderá voltar acima. Teremos de ter em consideração que numa difusão atómica, uma mensagem só considerada entregue quando todos os processos a receberem. Para o nosso projecto, iremos adoptar a solução de difusão fiável uniforme.

2.3 Difusão em Grupo

2.4 Modelo de Faltas

Nesta secção ir-se-á abordar o tipo de faltas que o sistema poderá suportar, e as faltas que não são suportáveis. Como ter um modelo em difusão, e haver um *GossipServer*(que é um servidor externo que recebe mensagens de processos e retransmite as mensagens para todos os processos conhecidos), que desempenha funções de coordenador do grupo, se tiver uma quebra de rede, não há maneira de tratar esta falha.

3 Appia e Pilha Protocolar

4 Atenção

Existe um GossipServer a correr em cada máquina.