Consistência Eventual

Sistemas Distribuidos e Tolerância a Falhas

Marco Bernardo

Departamento de Informática
Universidade da Beira Interior

25 de Maio de 2009

Descrição Geral

Introdução

Problema:

O desenvolvimento de sistemas distribuídos confiáveis numa escala mundial, exige um compromisso entre consistência e disponibilidade.

Proposta de solução:

Tolerar eventuais consistências de dados em sistemas distribuídos de larga escala.

Técnica de solução:

Dado o objectivo mundial destes sistemas, usam-se técnicas de replicação de forma ubíqua para garantir performance e alta escalabilidade consistente.

Definições

Sistemas Ubíquos:

Têm como objectivo tornar a interacção pessoa-máquina invisível, ou seja, integrar a informática com as acções e comportamentos naturais das pessoas.

Amazon SimpleDB:

Um *Web-Service* que fornece as principais funções de uma base de dados, quer a indexação, quer os *query's*.

Formas de Caracterização de Consistência

De acordo com Werner, existem duas formas de caracterizar a consistência:

- **Do ponto de vista do programador/cliente** - como são observados e actualizados os dados.
- **Do ponto de vista do servidor** - como é processado o fluxo das actualizações através do sistema e que garantias o sistema pode fornecer, no que diz respeito às actualizações.

Considerações

Sob o ponto de vista do cliente, quando se define um modelo de consistência, devem-se considerar os seguintes aspectos:

- **Sistema de armazenamento** - Definido como "caixa-negra". Deve ser assumido que, "por detrás do cenário", está algo em larga escala e altamente distribuído, para garantir durabilidade e disponibilidade.
- **Processo A** - Processo que escreve e lê a partir do sistema de armazenamento.
- **Processos B e C** - Processos independentes do processo A. Escrevem e lêem a partir do sistema de armazenamento.
  É importante considerar, que estes processos são independentes e necessitam de comunicar, para partilhar informação.

Tipos de Consistências (1/2)

Considerando que o processo A efectuou uma actualização num conjunto de dados, os seguintes exemplos ilustram os diferentes tipos de consistência:

- **Consistência forte** - Após a actualização completa, qualquer acesso subsequente (por A, B ou C) fornecerá o valor actualizado.
- **Consistência fraca** - O sistema não garante que os acessos subsequentes irão dar o valor actualizado.
  Um número de condições têm de ser conhecidas antes que o valor seja devolvido.
  O período entre a actualização e o momento em que é garantido ao observador que irá sempre observar o valor actualizado, é designado de janela de inconsistência.

Tipos de Consistências (2/2)

- **Consistência eventual** - É uma forma específica de consistência fraca.
  O sistema de armazenamento garante que, se nenhuma actualização for feita ao objecto, todos os acessos irão devolver o último valor actualizado.
  Se não ocorrer nenhum erro, o tamanho máximo da janela de inconsistência pode ser determinado, com base em factores, tais como:
  - Os atrasos de comunicação;
  - A carga no sistema;
  - O número de réplicas envolvidas do esquema de replicação.

  O sistema mais popular que implementa consistências eventuais é o DNS.

Variações da Consistência Eventual (1/2)

O modelo de consistência eventual apresenta um número de variações a considerar:

- **Consistência causal** - Se o processo A comunicou ao processo B que actualizou determinada informação, um acesso subsequente pelo processo B irá devolver o valor actualizado, sendo garantida uma nova escrita para substituir a escrita anterior.
  O acesso pelo processo C, que não tem relação causal com o processo A, está sujeito às regras normais da consistência eventual.
- **Consistência leitura/escrita** - Este é um modelo importante no qual o precesso A, após ter actualizado determinada informação, considera sempre o valor actualizado e nunca o anterior.
  Este é um caso especial da consistência causal.

Variações da Consistência Eventual (2/2)

- **Consistência de sessão** - Considera-se como uma versão prática do modelo anterior, onde um processo acede ao sistema de armazenamento no contexto de uma sessão.
  Desde que exista sessão, o sistema garante consistência de leitura/escrita.
- **Consistência de leitura monótona** - Se um processo identificou um valor particular para um objecto, qualquer acesso subsequente nunca vai devolver quaisquer valores anteriores.
- **Consistência de escrita monótona** - Neste caso, o sistema garante a colocação em série da escrita pelo mesmo processo.

Combinação de Consistências

Um número destas propriedades pode ser combinado, por exemplo:

Consistência de:

- Leitura Monótona

Consistência de:

- Sessão

Vantagens da combinação de consistências:

- Propriedades desejadas num sistema de consistência eventual;
- Torna mais simples a construção de aplicações;
- Permite ao sistema de armazenamento melhorar a consistência e fornecer maior disponibilidade.

Nível de Consistência

Sob o ponto de vista do servidor, o nível de consistência depende da forma como as actualizações são propagadas entre a réplica de dados.

Isto permite:

- Melhorar a taxa de transferência;
- Fornecer escalabilidade.

Contextualização

A consistência fraca/eventual:

- Ocorre quando nem todas as réplicas de dados participam na operação de actualização.
- Este tipo de consistência verifica-se, mais normalmente, nas duas situações seguintes:

- Replicações sólidas para o escalonamento da leitura;
- Casos em que o acesso de dados é complicado.

Operação de Actualização

- O intervalo de tempo até que todas as réplicas tenham sido actualizadas, é a janela de inconsistência.
- Na maioria dos sistemas, as actualizações são propagadas de uma maneira lenta para os restantes nodos, no conjunto de réplicas.
- Todo o sistema é vulnerável para a leitura, a partir de nodos, que ainda não tenham recebido as actualizações.

Melhoria da Consistência

O nível de consistência fornecido por um servidor, pode ser melhorado, através da comunicação, entre o cliente/servidor ou pelo próprio cliente:

- Dependendo da rigidez dos clientes, para com o servidor que executa o protocolo, podem ser alcançadas as consistências de:
  - Leitura/Escrita;
  - Sessão;
  - Monótonas.

  Estas consistências são mais fáceis de garantir, se for sempre o mesmo servidor a atender determinado cliente.
- Com a melhoria das consistências anteriores, é possivel melhorar o modelo de consistência eventual, o que fornece melhores ferramentas para o programador.

Amazon's Dynamo

O Amazon's Dynamo é um sistema de armazenamento, usado internamente em muitos serviços que constituem a plataforma *e-commerce*, tal como os serviços Web da Amazon.

O objectivo da Amazon's Dynamo é:

Permitir ao proprietário do serviço de aplicação, o qual cria uma instância do sistema de armazenamento Dynamo, fazer as trocas de informação entre:

- Consistência;
- Durabilidade;
- Disponibilidade;
- Performance.

Conclusão

A inconsistência de informação nos sistemas distribuídos em larga escala, deve ser tolerada por dois motivos:

- Melhoria de performance de leitura/escrita sob condições altamente concorrentes;
- Manuseamento de casos de partição, onde o modelo de maioria deveria realizar parte do trabalho indisponível, mesmo que os nodos estejam a processar.

O programador deve:

- Estar consciente que as garantias da consistência são fornecidas pelo sistema de armazenamento.
- Ter em consideração a ideia anterior, aquando do desenvolvimento de aplicações.

FIM...

Obrigado pela atenção!