Sistemas Distribuídos e Tolerância a Falhas

Consistência

Trabalho realizado por:

Gonçalo Dias, Nº. 14638
João Tavares, Nº 14888
Rui Brás, Nº 14820

segunda-feira, 28 de Abril de 2008

Consistência

Índice

- Consistent Global States;
- Distributed Consensus;
- Agreement on Membership;
- View Synchrony;
- Atomic Broadcast;
- Replica Determinism;
- Primary Partition;
- Weak Consistency.

Consistência

Definição

- Um sistema considera-se consistente quando o mesmo não viola as restrições de integridade impostas na sua especificação.

- Isto previne uma possível falha, já que verifica se as condições que levam a essa falha não se verificam.

Consistência

Consistent Global States

- Em Sistemas Distribuídos, é por vezes necessário obter uma **visão global (cut)** instantânea do sistema.

- Obter essa Visão global é uma tarefa de difícil concretização, já que estes sistemas estão em constante modificação (Dinâmicos).

Consistência

Consistent Global States

Caso prático:

- Encontro de coleccionadores de pacotes de açúcar;
- Tarefa: contar o n.º de pacotes de açúcar no encontro;
- Dificuldade: conseguir fazer uma contagem precisa do n.º de pacotes, tendo em conta que os coleccionadores transaccionam pacotes entre si.

Consistência

Consistent Global States

- Obter esta contagem é uma tarefa complexa pois pode haver transacções de pacotes, o que leva a que um determinado pacote pode ser contado mais que uma vez, ou mesmo nenhuma vez.

Consistência

Consistent Global States

Transpondo este exemplo para um sistema informático, consideramos que:

- Pessoas presentes na convenção -> nodos;
- Pacotes de açúcar -> mensagens;
- Troca de pacotes de açúcar ->troca de mensagens entre os nodos;
- Contagem do n.º de pacotes de açúcar -> Visão Global do sistema.

Consistência

Consistent Global States

Ao tentarmos obter uma visão global do sistema sem termos em conta as transacções efectuadas, dá-nos uma visão global inconsistente, pois os dados obtidos não correspondem à realidade!

Para tentar responder a este problema, Chandy e Lamport desenvolveram um protocolo, o "Snapshot protocol".

Consistência

Consistent Global States

Protocolo de *Chandy e Lamport*

- Utilização de canais FIFO que conectam os vários processos.
- Utilização de um MARKER (controlador de mensagens) para distinguir as mensagens enviadas antes e depois do pedido de snapshot.

Consistência

Consistent Global States

Snapshot global possui:

- estado local de cada processo;
- estado de cada canal.

Cada processo é responsável por capturar o estado de todos os seus canais de entrada.

Consistência

Consistent Global States

Algoritmo executado em cada processo:

- Salva o seu estado actual;
- Envia um MARKER por todos os canais de saída;
- Continua as suas execuções;
- Recebe MARKER de todos os canais de entrada;
- Processa a informação contida no seu estado guardado no 1º passo e a das mensagens recebidas que não continham MARKER;

Consistência

Consistent Global States

Todos os processos executam o algoritmo anterior.

O processo considera-se terminado quando o estado de todos os nodos foi capturado (o algoritmo descrito anteriormente foi executado).

Consistência

Distributed Consensus

Objectivo:

- Fazer com que haja um consenso entre os processos para chegarem a um resultado que dependa do valor inicial de cada um dos participantes.

Consistência

Distributed Consensus

Exemplo:

- Empacotamento de itens através de uma máquina.
- Os itens chegam em fila à máquina. Para acelerar o processo, existe um conjunto de máquinas disponível para empacotar.

Consistência

Distributed Consensus

- Quando duas ou mais máquinas estão livres, têm que decidir que máquina apanha o 1º item da fila.

- Uma solução é criar um centro de despacho, com o intuito de seleccionar a máquina que vai receber o item.

Consistência

Distributed Consensus

- Este exemplo pode ser realizado com uma via descentralizada, executando protocolos nas próprias máquinas.

- Problema: quando chega um novo item, as máquinas têm que chegar a um consenso sobre qual delas o trata.

Consistência

Distributed Consensus

- Solução: Se uma máquina está livre quando um novo item chega, propõe-se para servir esse item. Caso contrário, fica à espera de:
  - ficar livre para servir esse item;
  - receber alguma proposta de outra máquina.

Consistência

Distributed Consensus

- O consenso é atingido quando:
  - juntamos todas as propostas;
  - retiramos os duplicados;
  - ordenamos as propostas de acordo com o número da máquina;
  - retiramos a primeira proposta desta lista;
  - todas as máquinas recebem o mesmo número de propostas;

o algoritmo de selecção é determinístico.

Consistência

Agreement on Membership

- Em sistemas distribuídos ocorre frequentemente processamento cooperativo, em que cada membro executa uma parte da tarefa, tendo em conta o número, e por vezes outras características, de membros.
- Estes grupos são geralmente dinâmicos, ou seja, os membros podem sair ou juntar-se ao grupo.

Consistência

Agreement on Membership

- Para manter a informação actualizada é criado um serviço com informação dos membros, que fornece aos mesmos informação actualizada sobre o grupo.
- A lista de membros activos denomina-se de "*view*".

Consistência

Agreement on Membership

- Para se tentar manter esta lista actualizada é utilizado um algoritmo semelhante ao descrito no ponto anterior (consenso).
- O método utilizado é enviar uma view a todos os participantes sempre que haja uma alteração na mesma.

Consistência

Agreement on Membership

- A actualização da view pode ser feita da seguinte forma:
  - Quando um novo processo pretende ser incluído ou excluído do grupo, envia essa informação aos membros da view actual;
  - Ao receberem a informação de actualização, cada membro faz uma proposta para a nova view;
  - A nova view é definida usando uma regra (por exemplo, a proposta com maior número de membros).

Consistência

Agreement on Membership

- Esta solução é exemplificada no seguinte esquema:

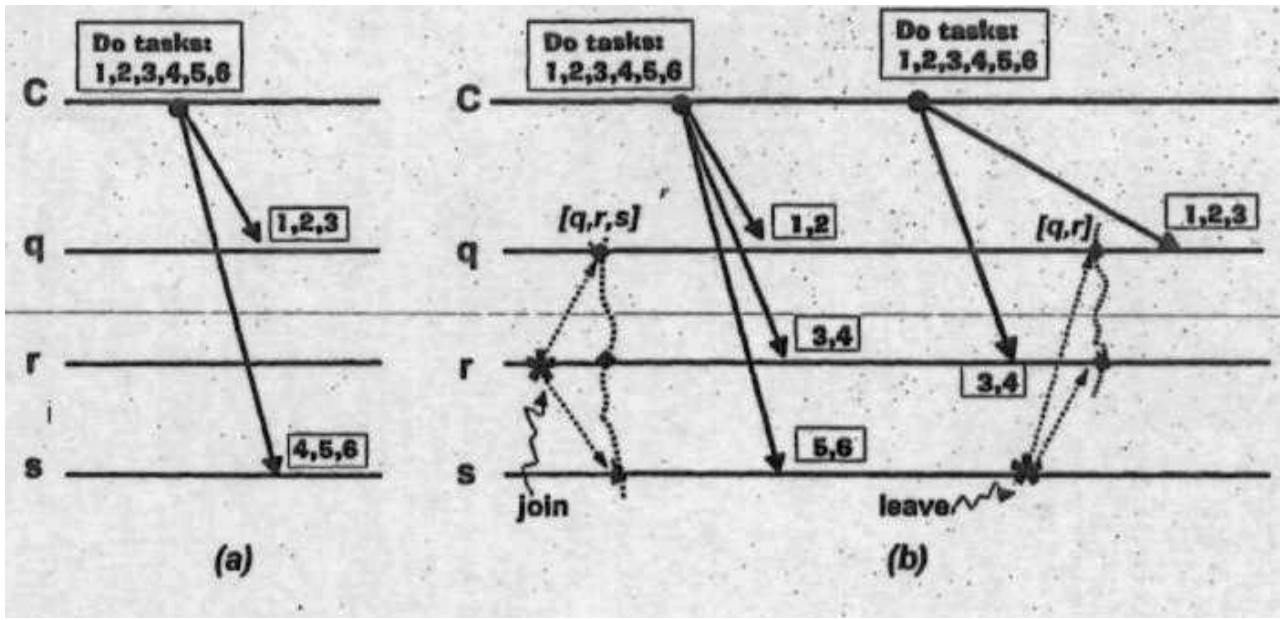

Fig. 1: Alteração da View

Consistência

Agreement on Membership

- Na figura anterior, no caso a), há apenas 2 membros no grupo, sendo o trabalho distribuído de forma uniforme por cada membro (3 tarefas);

- No caso b) estão na view 3 membros. Durante a distribuição das tarefas, o membro s abandona o grupo, mas a sua mensagem cruza-se com a mensagem das tarefas, fazendo com que r receba a informação da view incorrecta, e q a view actualizada, fazendo com que a distribuição não seja feita de forma correcta.

Consistência

View Synchrony

- O exemplo anterior demonstra inconsistência na actualização das views, já que não há garantia que o trabalho é distribuído de forma correcta devido á sincronização das views.

- Para resolver este problema, tem que se garantir a entrega a todos os membros da view antes das restantes mensagens.

Consistência

View Synchrony

- Para tal garantimos que o fluxo de mensagens é entregue a todos os membros da view actual antes de os mesmos receberem a nova view.

- Uma forma, apesar de ineficiente de garantir essa entrega é segundo um algoritmo de sincronização virtual desenvolvido por Birman e Joseph em 1987.

Consistência

View Synchrony

- Mensagens são enviadas em multicast;
- Quando um membro recebe uma mensagem entrega automaticamente essa mensagem;
- Quando há uma nova view, todos os membros param de enviar e entregar mensagens;
- Cada membro envia para os restantes elementos a lista de mensagens já entregues nessa view;
- Cada um guarda a lista de mensagens dos restantes dessa view;

Consistência

View Synchrony

- Após todas as listas estarem recebidas, cada processo actualiza a sua view da seguinte forma:
  - Todas as mensagens da lista são entregues;
  - A nova view é aceite;
  - O membro volta a enviar e aceitar novas mensagens.

- Com este algoritmo obtêm-se um atraso na implementação na nova view de forma a todos os membros receberem as mensagens pendentes.

Consistência

Atomic Broadcast

- Em alguns casos a resposta não depende apenas dos restantes membros, mas também de mensagens anteriores.
- Apesar de a sincronização de views garantir a entrega, não garante a ordenação das mesmas.
- Este protocolo garante a entrega ordenada das mensagens, e/ou a entrega das mensagens a todos os membros.

Consistência

Atomic Broadcast

- Este protocolo pode ser interpretado como um problema de consenso, já que os membros tem que concordar em:
  - Mensagem entregue;
  - A ordem da mensagem relativamente ás restantes.

Um algoritmo que pode resolver este possível estado de consistência é descrito de seguida.

Consistência

Atomic Broadcast

- Ao receber uma mensagem ele apenas armazena esta sem a entregar;
- Quando recebe uma view os processos trocam as mensagens para um "saco" comum, e após as mesmas estarem ordenadas segundo um regra, sendo distribuídas de seguida pelos destinatários.

Consistência

Atomic Broadcast

- De notar que o algoritmos apresentado apenas envia as mensagens quando há uma actualização da view, podendo haver um período demasiado longo sem a troca de mensagens.

- De forma a resolver isto, pode-se "forçar" uma actualização da view frequentemente.

Consistência

Replica Determinism

- Réplica consiste em manter cópias idênticas de um processo ou informação em diferentes localizações.
- Mesmo que uma das réplicas falhe as outras estarão disponíveis para fornecer o serviço.
- Quando uma réplica executa uma dada instrução, todas as réplicas executam essa instrução, e assim sucessivamente.

Consistência

Replica Determinism

- Deste modo, o estado de cada réplica pode ser comparado a qualquer momento e todas as réplicas consomem inputs e produzem outputs aproximadamente ao mesmo tempo.

- Uma das réplicas tem de assegurar que as outras recebam os mesmos inputs e as mesmas respostas.

Consistência

Replica Determinism

- Se as réplicas são executadas em nodos diferentes e os inputs mudam usando mensagens multicast, é natural que ocorra um certo grau de dessincronização.

- A quantidade de dessincronização permitida entre duas réplicas depende de factores relacionados com as limitações de tempo da aplicação.

Consistência

Primary Partition

- De modo a manter as réplicas num estado consistente, é necessário assegurar que a comunicação seja possível de modo que as réplicas possam ser coordenadas.

- Por vezes por causa de falhas de ligação ou crashes de routers, a rede é dividida em duas ou mais partições.

- Os nodos da mesma partição podem comunicar uns com os outros mas não com os nodos das outras partições.

Consistência

Primary Partition

- Exemplo1:

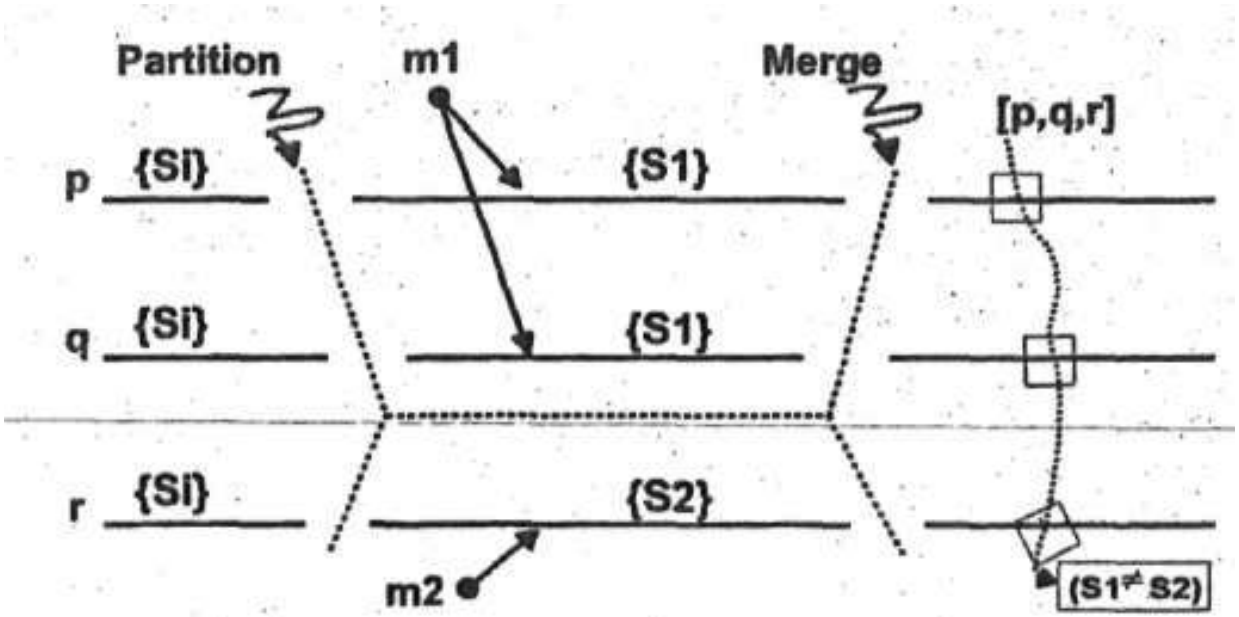

Figure 2.33. State Divergence with Partitioning

Consistência

Primary Partition

- O estado das réplicas em diferentes partições é provável que seja divergente;
- Para prevenir isso deve-se seleccionar uma das partições como "primary partition", activa e bloqueando a actividade das outras partições;
- A primary partition deve ser aquela que tem mais nodos pois permite analisar localmente sem comunicar com as outras partições.

Consistência

Primary Partition

- Exemplo2:

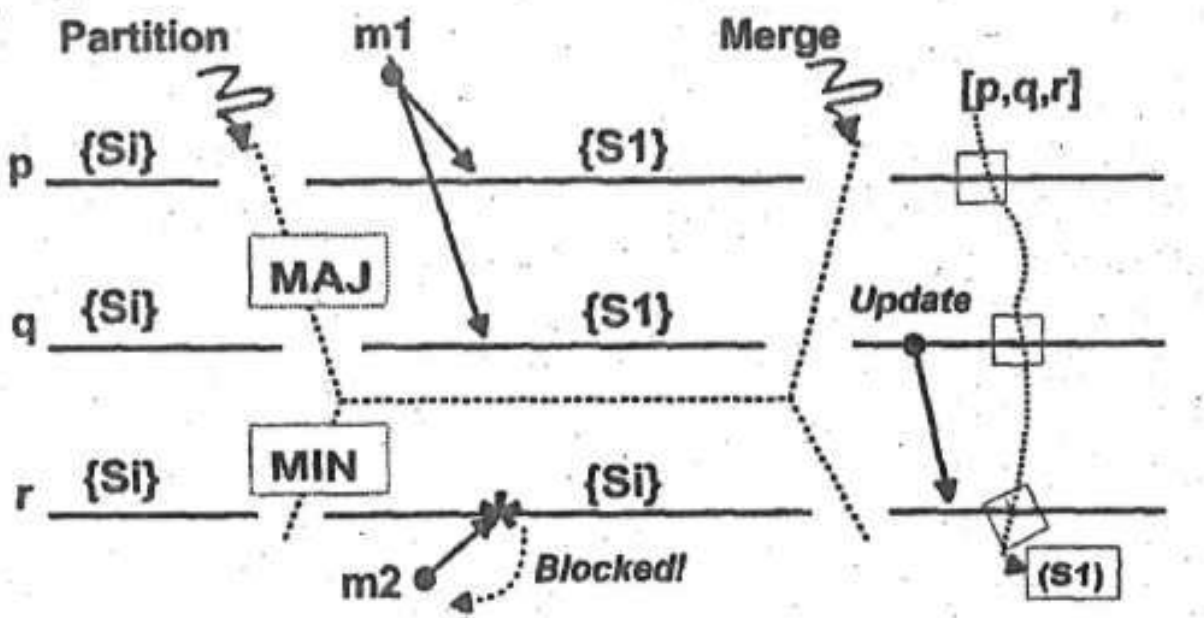

Figure 2.34. Primary Partition

Consistência

Weak Consistency

- Mantendo uma forte consistência ou, por outras palavras, evitando inconsistência, simplifica a programação concorrente distribuída;
- Em algumas aplicações, a capacidade de progredir é mais relevante do que assegurar que as coisas nunca divirjam;
- Isto torna-se relevante em ambientes onde as partições são frequentemente ou até mesmo forçadas a uma computação móvel, onde o utilizador pode desligar a sua máquina da rede mas continuando a querer trabalhar.

Consistência

Weak Consistency

- Considere-se um ambiente de computação móvel: muitos dos ficheiros que são actualizados por utilizadores são ficheiros pessoais;
- Neste caso, é vantajoso permitir que o processo seja feito em qualquer partição;
- Por outro lado, se o mesmo ficheiro for actualizado simultaneamente por mais do que um utilizador, vai ser muito difícil juntar ambas as actualizações num novo ficheiro de maneira automática.

Consistência

Questões???

Consistência

FIM