Sistemas Distribuídos e Tolerância a Falhas

Fault Tolerance Middleware for Cloud Computing

Docente:
Prof. Drª Paula Prata

2010-2011
Universidade da Beira Interior

Paula Freire M3841
Tiago Machado M3863

Plano da Apresentação

- **Introdução**
- ***Low Latency Fault Tolerance middleware***
- **Conceitos**
  - Modelo de Falhas
  - Tipos de replicação
  - Grupos de serviço e grupos de processos
  - Ligações virtuais

Plano da Apresentação (cont.)

- **Low Latency Messaging Protocol**
  - Serviço de entrega fiável
  - Serviço de entrega totalmente ordenada
- **Leader-Determined Membership Protocol**
  - Mudança da réplica primária
  - Mudança de um backup

Plano da Apresentação (cont.)

- **Virtual Determinizer Framework**
  - Modelo de Threading
  - Algoritmo genérico
  - Serviço consistente de Multi-Threading
- **Desempenho do LLFT**
- **Considerações Finais**

Introdução

- O objectivo do cloud computing é fornecer serviços que são entregues aos seus utilizadores de uma forma transparente, fiável e segura.

- Uma das dificuldades é assegurar que as aplicações corram de forma a que os utilizadores não sejam afectados por falhas no sistema.

Low Latency Fault Tolerance Middleware

- O LLFT é um middleware que fornece tolerância a falhas para aplicações distribuídas, no contexto do *cloud computing*.

- Replica os processos das suas aplicações, através de uma abordagem de replicação líder/seguidor.

Low Latency Fault Tolerance Middleware (cont.)

- Os processos replicados integram um grupo de processos, e destes grupos os que fornecem serviços aos utilizadores constituem grupos de serviços.

- O *middlware* é constituído por:
  1. **LLMP** - *Low Latency Messaging Protocol*
  2. **LDMP** - *Leader-Determined Membership Protocol*
  3. **VDF** - *Virtual Determinizer Framework*

Conceitos – Modelo de falhas

- O *middleware* replica processos da aplicação para proteger a mesma contra vários tipos de falhas, em particular:
  - "Crash faults"
  - Falhas de sincronismo / "Timing faults"

Conceitos – Tipos de replicação

O LLFT suporta dois tipos de replicação:

- Replicação semi-activa
  - A réplica primária ordena as mensagens que recebe, executa as operações correspondentes, e fornece informações sobre operações não determinísticas aos backups.
  - O *backup* recebe e regista as mensagens recebidas, executa as operações de acordo com essas informações e regista as mensagens de saída.

Conceitos – Tipos de replicação (cont.)

- Replicação semi-passiva
  - A réplica primária ordena as mensagens que recebe, executa as operações correspondentes, e fornece informações sobre operações não determinísticas aos backups.
  - Adicionalmente, comunica actualizações de estados aos backups, bem como actualizações de ficheiros e bases de dados.
  - Um backup recebe e regista as mensagens recebidas e actualiza os seus estados mas não executa as operações e não produz mensagens de saída.

Conceitos – Grupos de serviços e grupos de processos

- Um **grupo de serviços** é um conjunto de processos interativos que de uma forma colectiva fornecem serviços a um utilizador.

- Um **grupo de processos** é um conjunto de réplicas de um determinado processo.

  - Tipicamente um conjunto de processos correm em diferentes processadores – um dos elementos é o processo primário, e os restantes são os backups.

Conceitos – Ligações virtuais

- Com o LLFT é introduzido o conceito de ligação virtual, que resulta de uma extensão da ligação ponto-a-ponto do TCP, baseando-se contudo em alguns princípios do UDP.

- É uma ligação entre dois pontos de comunicação, em que cada um destes pontos é preenchido por um grupo de processos.

  - As mensagens ocorrem de um ponto para outro, sendo a sua origem/destino definidas por um porto virtual.
  - Fornece um canal de comunicação N-N.
  - Os grupos de processos/serviços necessitam de saber os *ids* dos portos virtuais para interagir (à semelhança do TCP)

Conceitos – Ligações virtuais (cont.)

- Os utilizadores acedem ao grupo de serviço através do porto virtual, através do *gateway* definido na *cloud*.

- Tipicamente cada ligação virtual é constituída por um (ou mais) grupos de processos, e os seus membros são participantes nas mesmas.

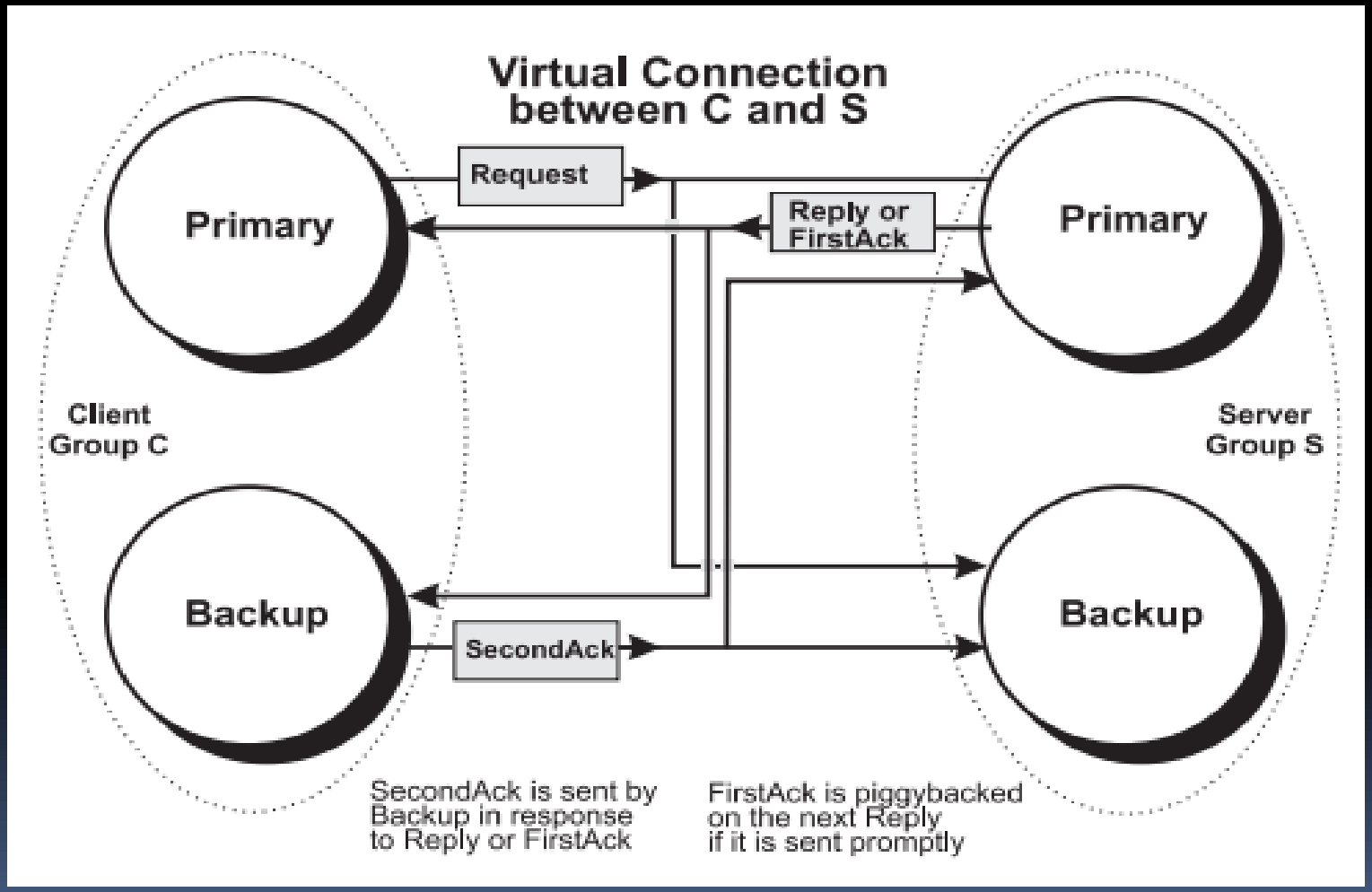
Low Latency Messaging Protocol

Este protocolo fornece os seguintes serviços para a comunicação por mensagens:

- **Entrega fiável**
- **Ordenação total**

- O objectivo é converter o serviço de entregas de mensagens *multicast* pouco fiável do UDP num serviço fiável e ordenado entre dois pontos de comunicação.

Serviço de entrega fiável



Serviço de entrega totalmente ordenada

- A réplica primária comunica as suas informações de uma forma ordenada para as réplicas de segurança (backups) para que os mesmos possam reproduzir as acções da réplica primária em caso de falha.

  - Se a réplica primária falhar, uma forte consistência de réplica é mantida.

Leader-Determined Membership Protocol

- O protocolo LDMP assegura que os membros de um grupo de processos tenham um acesso, uma adesão consistente à replica primária do grupo.
- A réplica é escolhida de uma forma determinística – maior eficiência.
- A réplica primária determina a adição/remoção das réplicas clonadas no grupo, baseando-se na sua precedência e na sua classificação.

Leader-Determined Membership Protocol (cont.)

- A precedência é determinada pela ordem em que o mesmo se juntou ao grupo.

- O nível de classificação de um membro de um grupo de processos é 1, e os seus backups têm valores incrementais. (2,3,4,5..)

Mudança da réplica primária

A alteração da réplica primária num grupo é gerida **em duas fases:**

- Determinação do novo membro: numa primeira fase é definina a réplica primária, em que esta determina os backups que são incluidos.
  - Se um backup detectar que a R.P. está "defeituosa", envia uma mensagem "ProposePrimary", em que denuncia o seu novo estado – como réplica primária.

➢ Quando um pedido de nova réplica primária é concordado por todos as réplicas constituintes, dá-se inicio à segunda fase.

Mudança da réplica primária (cont.)

Recuperação da alteração do estado:

- A nova réplica primária inicia a sincronização de todos os portos virtuais, de forma a assinalar o seu novo estado, e o antigo estado da antiga réplica primária.

- Quando a nova réplica termina as operações de sincronia de acordo a informado determinada pela antiga réplica primária, conclui-se a segunda fase.

  - É restabelecida a numeração sequencial e as classificações das réplicas através da geração de informação ordenada.

Mudança de um *backup*

- Adição de um backup
  - O backup envia uma mensagem "ProposeBackup";
  - A réplica primária classifica o novo de backup, enviando de seguida a mensagem "AcceptBackup" para o grupo (que contêm agora o novo membro);
  - Os backups responde com uma notificação de mensagem recebida – o novo membro é aceite;
  - Se a réplica primária falhar antes do novo backup ter recebido a verificação do seu estado, um dos outros backups propõe-se a ser a nova réplica primária.

Mudança de um *backup*

- Remoção de um backup
  - A réplica primária modifica a classificação dos backups no grupo, e é enviada uma mensagem "RemoveBackup".
  - Um backup ao receber esta mensagem em que não é incluindo no grupo de membros resultante, o backup restabelece o seu estado e envia uma mensagem "ProposeBackup" pedindo para ser readmitido.

Virtual Determinizer Framework

- As aplicações existentes na nuvem geralmente envolvem várias fontes não determinísticas.

- Para manter uma forte consistência das réplicas, é necessário "limpar" ou "mascarar" essas fontes, de forma a obter um funcionamento da aplicação virtualmente determinístico.

- O LLFT introduz uma *framework* para a "higienização" dessas fontes, de forma a permitir um funcionamento *multi-threading determinista*.

Modelo de Threading

- Cada thread num processo tem um identificador único;
- Uma variável partilhada por mais de uma thread é protegida por um *mutex* – podem ser criados dinamicamente.
- Cada réplica num grupo de processos executa o mesmo conjunto de threads.
- Para que um réplica permaneça consistente é necessário que as actualizações sobre dados partilhados ocorram simultaneamente e similarmente, e cada thread actualize esses dados na mesma ordem.

Algoritmo genérico

- Regista a informação de chamadas de sistema não determinísticas de uma forma ordenada e devolve a mesma, de forma que todas as réplicas obtêm os resultados na mesma ordem.

- Para cada operação não deterministica, o algoritmo genérico regista a seguinte informação:

  - O identificador da thread;
  - O identificador da operação;
  - O contador de operações;
  - Os metadados da operação.

Serviço consistente de Multi-Threading

- É uma instância especifica do algoritmo genérico, em que é usado um mutex "M" como identificador de operação.

- O serviço permite concorrência de threads que não adquirem simultaneamente o mesmo mutex – alcançando-se desta forma máximo possível de concorrência e consistência

Serviço consistente de Multi-Threading (cont.)

- Quando é necessário realizar o lock de uma thread, o serviço examina os metadados e verifica se existe algum código de erro.

- **Caso exista**: é devolvido o controlo da thread que solicita o lock.

- **Caso não exista**: é delegado o lock para a chamada de sistema fornecida pelo sistema operativo.

  - se o mutex não estiver adquirido por outra thread, a thread adquire o lock imediatamente;
  - Caso contrário, a thread que solicite o stock é suspensa, e resumida assim que S.O. liberte o lock da thread que o detém.

Desempenho do LLFT

- Recuperação de falha na réplica primária – quase inteiramente dependente do tempo de detecção de falhas e da latência do lado do cliente.

- Recuperação de falha num backup – a réplica primária remove o backup, não tendo impacto na latência no lado do cliente.

- Adição de uma réplica(nova ou reiniciada) – é adicionado um novo backup. Apesar de se ter de fazer a actualização do estado do novo backup esta actualização não interfere no tempo de latência no lado do cliente.

Considerações Finais

- O middleware LLFT (Low Latency Fault Tolerancy) fornece tolerância a falhas para aplicações distribuídas num ambiente de *cloud computing* ou *data center*.

- Vários cálculos demonstram o LLFT alcança baixos e competitivos valores de latência nas comunicações.

- Aplicações desenvolvidas com recurso a APIs de sockets, ou Web Services, podem ser replicados através deste *middleware* – sem modificações nas próprias aplicações.

  - A sua genericidade, transparência, e baixo custo de integração torna-o apropriado.

Questões?