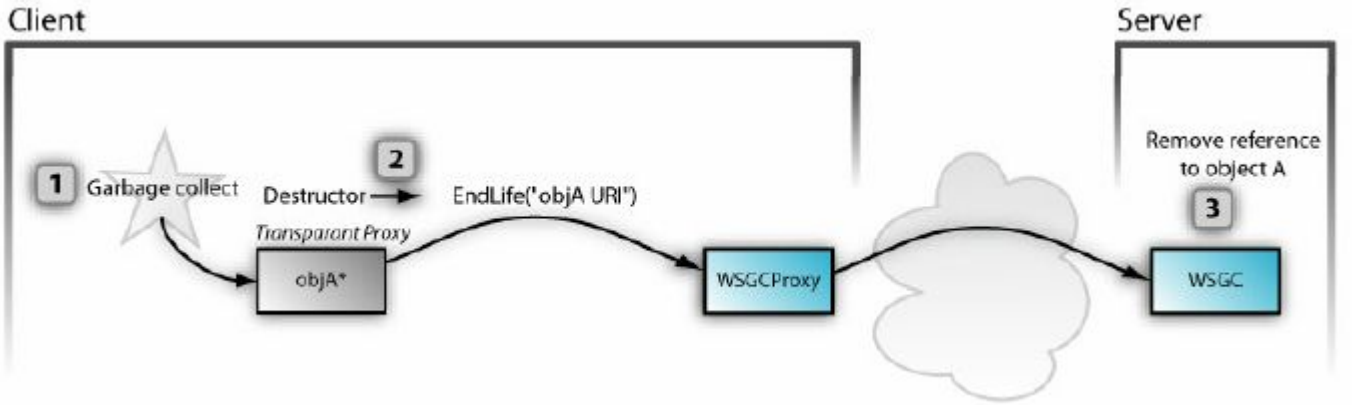
INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Distributed Garbage Collection

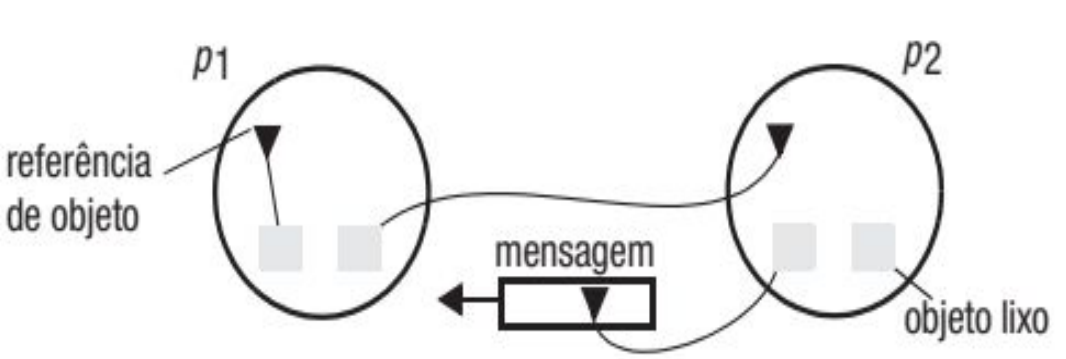
O que é?

Coleta de Lixo Distribuída

É uma técnica utilizada em sistemas distribuídos para gerenciar a memória e recuperar o espaço ocupado por objetos que não estão mais sendo referenciados e, portanto, podem ser considerados lixo.



Exemplo



Processo p1 tem dois objetos, ambos com referências – um deles é referenciado localmente pelo próprio p1 e o outro é referenciado remotamente por p2.

O processo p2 tem um objeto que é lixo, sem referências à ele em nenhuma parte do sistema. Ele também tem um objeto para o qual nem p1 e nem p2 tem uma referência, mas existe uma referência a ele em uma mensagem que está em trânsito entre os processos. Isso mostra que, quando consideramos as propriedades de um sistema, devemos incluir o estado dos canais de comunicação, assim como o estado dos processos.

INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Distribuída e Local

Distribuída vs Local

Na coleta distribuída os coletores locais devem ser coordenados para acompanhar de forma **consistente** as referências em mudança entre os espaços de endereço. Com isso, as falhas comuns em sistemas distribuídos *(mensagens perdidas, duplicadas ou atrasadas e falhas de espaço individuais)* tornam o processo mais complexo do que o realizado na coleta local.

Assim, temos um problema desafiador: recuperar todos os tipos de estruturas de dados enquanto alcançamos eficiência, escalabilidade e tolerância a falhas.

Várias propostas tentaram projetar um Garbage Collector distribuído que atendesse a todos esses requisitos. O grande número de propostas incompletas reflete a dificuldade do desafio.

INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Algoritmos iniciais de coleta de lixo distribuída

Contagem de Referências Distribuídas

Funcionamento: Num sistema fracamente acoplado, a criação de uma nova referência para um objeto requer que uma mensagem seja enviada para esse objeto a fim de que seu contador de referências seja incrementado. Da mesma forma, caso uma referência remota seja removida, uma mensagem deve ser enviada informando ao objeto que seu contador deve ser decrementado.

Problemas: Deve-se prevenir que um objeto seja coletado enquanto ainda existe uma referência remota para ele.

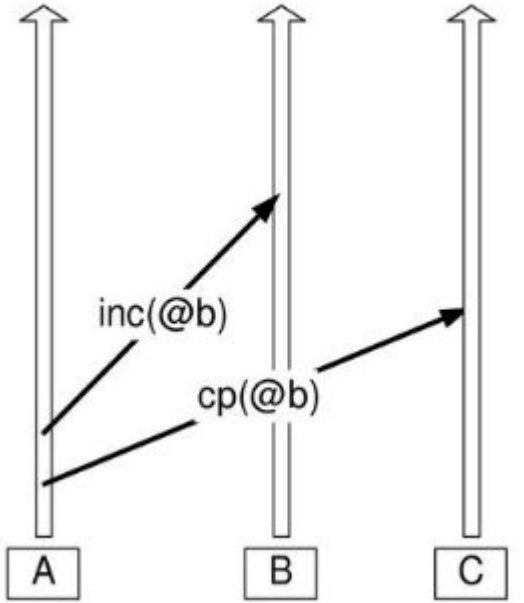
Contagem de Referências Distribuídas

Essa situação de risco pode acontecer quando as mensagens são recebidas em uma ordem diferente da esperada. Por exemplo, se uma mensagem informando a remoção da última referência para um objeto chega antes de uma mensagem informando a duplicação da referência.

Solução: Protocolo de comunicação desenvolvido no trabalho de Lerman e Maurer [Lermen, 1986] em que as mensagens entre pares de objetos são entregues na ordem em que foram enviadas. As mensagens devem ser confirmadas e um objeto só pode ser coletado se um número igual de mensagens de duplicação, remoção e confirmação foi recebido por esse objeto.

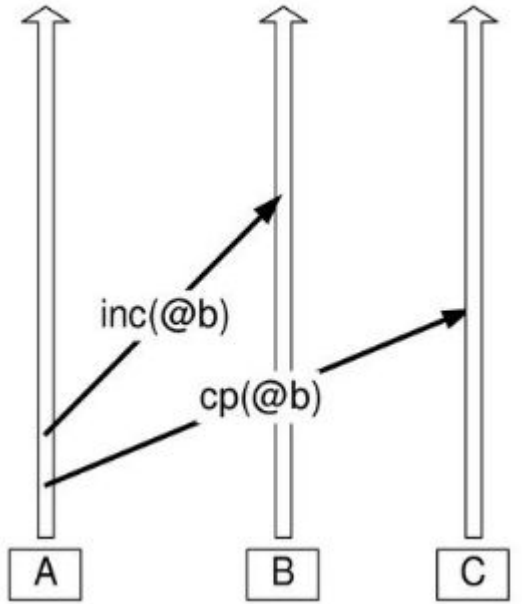
DGC baseada em Contagem de Referências

- Cada objeto tem um contador de referências
- Um comando inc(@x) e dec(@x) controlam o aumento do número de referências.
- Tal controle é feito por envio de mensagens à CPU que controla o objeto alvo.
- A remoção é feita quando o número de referências for igual a 0.



DGC baseada em Contagem de Referências

Principal problema: quando há ciclos isolados, os objetos envolvidos possuem referências circulares uns para os outros. Mesmo que esses objetos não sejam mais acessíveis por nenhum outro ponto do programa, seus contadores de referência nunca alcançarão zero. Isso acontece porque a contagem de referência não detecta ciclos; ela apenas rastreia as referências atuais para cada objeto.



Mark-sweep Distribuído

Funcionamento: Combina as coletas locais e independentes com um coletor de lixo global. Durante a fase de detecção de lixo, os coletores locais enviam e recebem mensagens indicando que determinados objetos devem ser marcados. Sendo assim, os espaços estão cooperando alternadamente para a detecção de lixo global. A fase de detecção de lixo está completa quando todos os objetos públicos alcançáveis foram marcados e não existem mais mensagens de marcação ou confirmação em trânsito. Em seguida, os coletores locais iniciam individualmente a fase de desalocação de memória a fim de se desfazer dos objetos públicos e locais que não serão mais utilizados.

Mark-sweep Distribuído

Problemas: Dificuldade em sincronizar a fase global de detecção de lixo com as fases locais de desalocação de memória.

Solução: Algoritmo descrito por [Hughes, 1985] que substitui os bits referentes à marcação de um objeto por timestamps, assumindo a existência de um relógio global e que a comunicação é feita de modo instantâneo, evitando problemas com mensagens em trânsito. Um coletor de lixo global marca os objetos com o tempo em que eles foram iniciados, caso o tempo atribuído ao mesmo não seja menor do que o tempo global atual. Os coletores de lixo locais propagam os tempos dos objetos do conjunto raiz para suas folhas, executando parte da fase de detecção de lixo. Ao final da coleta de lixo local, mensagens de atualização são enviadas aos objetos remotos alcançáveis para que estes aumentem suas marcas de tempo.

Preocupações atuais

Os algoritmos citados anteriormente são considerados algoritmos de coleta de lixo distribuída para redes locais, com isso é possível considerarmos que há uma comunicação confiável e assumir a existência de algum tipo de sincronização global.

Algoritmos que pretendem rodar em WANs (Wide Area Networks) não podem fazer tais suposições, tendo enfoque diferente dos iniciais como, por exemplo, não se preocuparem em determinar quando iniciar a coleta de lixo e resolver o problema da sincronização, não assumindo a existência de um relógio global ou uma troca de mensagens confiável.

Coleta de Lixo Assíncrona para Protocolos de Checkpointing

Funcionamento: De tempos em tempos, cada processo salva o estado local da aplicação como um checkpoint. Quando uma falha ocorre, o programa distribuído retoma a sua execução a partir do checkpoint global consistente mais recente, também chamado de linha de recuperação.

Esse rollback é organizado por meio de protocolos de checkpointings de comunicação induzida que inserem nas mensagens da aplicação informações de controle e ordenam aos processos a criação de checkpoints forçados.

Problemas: Criação de checkpoints obsoletos que não serão mais usados em linhas de recuperação futuras, ocupando espaço de armazenamento.

INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Importância de coletar o tempo global

Coleta do Tempo Global

Em um sistema distribuído, não há memória compartilhada, então cada processo tem seu próprio estado local. Isso torna difícil determinar se um objeto não é mais referenciado por nenhum processo e, portanto, pode ser coletado como lixo.

Um estado global é uma ‘’imagem instantânea’’ de todo o sistema em um determinado ponto no tempo. Inclui o estado local de todos os processos, bem como as mensagens que estão em trânsito no momento. Essas informações podem ser usadas para determinar se um objeto ainda é referenciado por algum processo.

Pontos positivos

- Pode ajudar a evitar vazamentos de memória.
- Ele pode melhorar o desempenho do sistema liberando memória que não é mais necessária.
- Pode simplificar a implementação da coleta de lixo.

Desafios

- Pode ser difícil obter um estado global consistente.
- Pode ser caro transmitir o estado global para todos os processos.
- Pode ser difícil garantir que o estado global esteja atualizado.

Conclusão

A coleta de lixo distribuída é um problema complexo, mas usar um estado global pode facilitar a solução. Ao rastrear as referências a todos os objetos no sistema, podemos determinar quais objetos ainda são necessários e quais podem ser coletados como lixo.

Referências Bibliográficas

[1] BARROS, Alexandra. Monografia de Seminários em Sistemas Distribuídos: Uma análise da evolução da coleta de lixo distribuída. Departamento de Informática - PUC Rio.

[2] TANENBAUM, Andrew S.; VAN STEEN, Maarten. Sistemas Distribuídos: Princípios e Paradigmas. 2. ed. São Paulo: Pearson Addison Wesley, 2007.

Obrigado !