UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

INE5645 - Programação Paralela e Distribuída

Profº Dr. Odorico Machado Mendizabal

Alan Vinicius Cezar Ensina (16206354)

A.4 Problemas de sincronização

A análise a seguir mostra a definição e a solução do problema de sincronização do Jantar dos Filósofos.

Para mais informações sobre o código acesse o repósitório no github: https://github.com/alanensina/INE5645-ParallelAndDistributedProgramming/tree/master/Tarefa3

DEFINIÇÃO DO PROBLEMA

O Jantar dos Filósofos foi proposto por Dijkstra em 1965 para tratar de um problema de sincronização. Desde então, grande parte dos algoritmos de sincronização acabam sendo relacionados ao problema proposto por Dijkstra em seu Jantar dos Filósofos.

O problema consiste em cinco filósofos sentados em uma mesa para jantar. Cada filósofo possui um prato de espaguete à sua frente e apenas um garfo, porém o espaguete é muito escorregadio, onde que para conseguir comer, serão necessários 2 garfos, um para enrolar o espaguete e outro para segurar.

Sendo assim, cada filósofo alterna entre suas duas tarefas: pensar e comer. Quando um dos filósofos fica com fome, ele pega um garfo da esquerda e outro da direita para que assim consiga comer o espaguete. Caso ele consiga pegar os dois garfos, ele come durante um certo período de tempo e em seguida devolve os garfos para a mesa, em seguida ele volta a pensar.

O problema em questão é elaborar um algoritmo onde os filósofos possam fazer as duas tarefas sem que fiquem travados.

POSSÍVEL SOLUÇÃO E SUAS CARACTERÍSTICAS

Uma das primeiras alternativas para solução é a que cada que o algoritmos seja sequencial, sem que nenhuma das tarefas sejam executadas de modo paralelo. Porém essa solução não é muito viável nos dias de hoje, onde se formos fazer uma analogia aos sistemas de hoje em dia, isso seria muito custoso, pois imagine um sistema bancário onde diariamente ele realiza centenas de milhares de transações financeiras entre contas.

Imagine se cada transação só pudesse ser feita se nenhuma outra estivesse sendo feita naquele exato momento. Isso poderia acarretar um enorme prejuízo tanto para o banco quanto para o usuário do sistema.

A solução em si foi desenvolvida para que o algoritmo realizasse suas tarefas de modo paralelo, porém com alguns cuidados a serem seguidos, a fim de evitar problemas de *deadlocks* e *starvation.*

Deadlocks é o conceito onde dois ou mais processos ficam travados e impossibilitados de realizarem suas tarefas, já *starvation* é quando um processo nunca é executado, pois processos de prioridade maior sempre o impedem de ser executado.

No caso do Jantar dos Filósofos, o *deadlock* ocorreria caso todos os filósofos pegassem um garfo e ficassem em posse do garfo aguardando outro outro garfo ser liberado, sendo assim o jantar nunca chegaria ao fim. Já o cenário para ocorrer *starvation* seria se um ou mais filósofos ficassem esperando que 2 garfos estivessem disponíveis para jantar, porém os garfos nunca ficassem, sendo assim o filósofo nunca jantaria.

Para que esses problemas não ocorresse, o algoritmo utilizou de semáforos.

Os semáforos são uma alternativa para cenários de concorrência que suportam o acesso concorrente a zonas de memória partilhada.

Para isso, o cenário será implementado usando uma *thread* para representar cada um dos filósofos e os semáforos para representar cada garfo. No momento em que um filósofo tenta pegar um garfo ele realizará uma operação espera no semáforo,

já quando o filósofo terminar de comer e largar o garfo, ele envia um sinal ao semáforo informando o fim da refeição.

```
23
24  void pegarGarfo(int nfilosofo){
25      sem_wait(&sem);
26      estado[nfilosofo] = FOME;
27      printf("Filósofo %d está com fome.\n", nfilosofo+1);
28      verificar(nfilosofo);
29      sem_post(&sem);
30      sem_wait(&S[nfilosofo]);
31      sleep(1);
32  }
```

Cada um dos filósofos (threads) seguirá o mesmo caminho, tomando as mesmas ações. Porém estamos seguros que os sinais e espera que damos ao semáforo nos protege de *deadlocks*. O único risco que passamos é se caso mais do que um filósofo ficar com fome ao mesmo tempo e ficassem em espera por muito tempo, causando assim um *starvation.*

Sendo assim devemos ter total atenção ao desenvolvermos algoritmos paralelos que necessitem de sincronização, pois devemos conhecer muito bem o cenário em que estamos lidando para que não ocorra nenhum desses problemas citados anteriormente.