UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE CIÊNCIAS AGRÁRIAS E DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
BACHARELADO EM ENGENHARIA DE SOFTWARE

ARILSON FERREIRA DOS SANTOS
CLAUDIO ALEX MESSIAS DA ROSA
EDUARDO LUIZ SCHADE SOARES
SILVIA TATYARA RODRIGUES LOPES

SINCRONIZAÇÃO DE PROCESSOS

PONTA GROSSA
2019

Sincronização de Processo

Na década de 60, com o surgimento dos primeiros sistemas operacionais multiprogramáveis, começaram o desenvolvimento de programas que pudessem ser executados de forma concorrente. A execução concorrente tem como base a execução de forma colaborativa de múltiplos processos ou *threads*, trabalhando na execução de uma mesma tarefa.

Como um processo é executado com tarefas concorrentes, acontece o compartilhamento de recursos e, caso não tenha gerência desses recursos, pode haver inconsistências nos dados. O compartilhamento de memória é um mecanismo de comunicação entre os processos em execução, onde há também o compartilhamento de informações em operações de escrita e leitura através do *buffer* de memória.

A Sincronização

Os mecanismos de sincronização garantem a comunicação entre os processos concorrentes e o acesso aos recursos, sendo fundamentais para garantir a confiabilidade na execução das aplicações.

Porém, durante o compartilhamento, existem problemas como compartilhamento de um arquivo de disco para alguma atualização simultânea, pois deve garantir que os processos em execução atualizem o mesmo arquivo e não criem dados inconsistentes. Outro problema é o compartilhamento de uma variável para dois processos concorrentes, garantindo que não tenha alteração do valor da variável utilizada pelo outro processo.

Exclusão Mútua

É caracterizada como um mecanismo simples para evitar compartilhamentos problemáticos, impedindo que dois ou mais processos utilizem o mesmo recurso simultaneamente.

Quando um recurso estiver em uso por um processo, todos os outros processos devem ser colocados em espera, até o recurso estar disponível, mantendo acesso exclusivo denominado de Exclusão Mútua. Essa exclusão afeta apenas os processos concorrentes que estão acessando um recurso compartilhado.

Passagem de Mensagem

Partindo do conceito de semáforos binários e mecanismos de exclusão mútua, que possuem características similares, sendo compartilhamento de variáveis, cria-se um novo paradigma de programação baseado em sincronização e comunicação de processos, chamado de Passagem de Mensagem.

Os processos realizam o envio e o recebimento de mensagens, em vez de ler e armazenar em variáveis em memórias compartilhadas. A troca de mensagem é realizada por duas primitivas chamadas de Send e Receive, que são primitivas que garantem uma sincronização e comunicação melhor entre os processos.

A sincronização entre os processos é garantida pela sincronização entre os mesmos, a fim de um processo só pode mandar uma nova mensagem caso haja uma confirmação que a mensagem foi enviada.

Com isso, criasse duas classes de primitivas, primitivas do tipo Bloqueantes (Blocking), que são executadas quando um processo quer enviar uma nova mensagem enquanto um outro processo está recebendo a mensagem, e assim que o processo recebedor confirmar que recebeu a mensagem, a primitiva destrava o processo de envio. O outro tipo de primitiva é chamada de Não-Bloqueante (Nonblocking), que ocorre quando um processo quer executar uma primitiva e não bloqueios nem no recebimento e nem no envio.

O bloqueio e não bloqueio de primitivas gera combinações a fim de haver a sincronização entre os processos, essas combinações podem ser Assíncronas, Semi-síncronas, e totalmente Síncronas.

Com as primitivas Send e Receive tende-se a ter dois tipos de comunicação entre os processos sendo Direto ou Indireto. Cada método tem suas particularidades, como, no método Direto a comunicação entre os processos é totalmente sincronizada ou simultânea, permitindo uma comunicação melhor entre

os processos. Em contrapartida no método Indireto cria-se uma Mailbox (caixa de mensagem) que atua como uma fila de mensagens, onde a primeira mensagem a ser enviada é a primeira a ser recebida ou lida. Este método é similar ao método FIFO, mas caso uma mensagem precise de elevação é criado uma fila de prioridade, enviado mensagens com base no nível de prioridade que o SO definiu (LEITE, 2005).

O problema da Transação Bancária

Um usuário deseja debitar o valor em uma conta e credita esse valor em outra conta, de forma que apenas um processo possa alterar vários itens de dados. Considerando todos as propriedades de transações, a propriedade de atomicidade caracteriza as transações para que sejam realizadas de forma indivisível; a propriedade de consistência pede para que não sejam violadas as regras do sistema; a propriedade de isolamento considera que transações concorrentes não devem interferir entre si; e a propriedade de durabilidade mantém todas as alterações após realizar o *commit*; a alteração deve ser realizada mesmo que a conexão com o sistema caia após a primeira transação.

Para que as transações requeridas sejam realizadas sem perda e inconsistência de dados, devemos considerar a propriedade de atomicidade e realizar uma transação atômica, onde todas as transações sejam realizadas, ou nada será modificado. Dessa maneira, evita que tenha perda da atualização dos dados e que não tenha uma recuperação inconsistente (PRETTI, 2009).

Referências

LEITE, Andreza. **Sistemas operacionais.** Florianópolis: IF-SC, 2009.

PRETTI, Marco. **A message-passing algorithm with damping.** Journal of Statistical Mechanics: Theory and Experiment, v. 2005, n. 11, p. P11008, 2005.