Exclusão mútua aplicada no controle do número de acessos em um servidor web distribuído

Matheus Santos Araújo, Gabriel Furtado Lins, Ivo Aguiar Pimenta

[1]Centro de Ciências e Tecnologia – Universidade Estadual do Ceara (UECE)
Avenida Silas Munguba – Fortaleza – CE – Brazil

Resumo. *O controle de um sistema web é um desafio que vem crescendo visto que as arquiteturas modernas estão cada vez mais distribuídas, logo são necessárias técnicas para manter consistentes as informações dos sistemas e evitar que dados de servidores diferentes de uma mesma aplicação acabe tendo dados conflitantes ou desatualizados. A partir dessa problemática, desenvolvemos uma solução de exclusão mútua em um servidor web distribuído o qual contém diversas máquinas atualizando a variável de controle de acessos em um website cliente e consequentemente gerando conflitos, os quais são resolvidos utilizando os algoritmos de Peterson, Dekker e Lamport.*

1. Introdução

Uma condição de corrida é uma falha num sistema ou processo em que o resultado do processo é inesperadamente dependente da sequência ou sincronia de outros eventos. Apesar de ser conhecido em português por 'condição de corrida' uma tradução melhor seria 'condição de concorrência' pois o problema está relacionado justamente ao gerenciamento da concorrência entre processos teoricamente simultâneos [Jr. 2006]. O fenômeno pode ocorrer em sistemas eletrônicos, especialmente em circuitos lógicos, e em programas de computador, especialmente no uso de multitarefa ou computação distribuída conforme [Tanenbaum 2006]. No nosso contexto abordamos uma simulação de uma variável global em um cliente a qual armazena uma contagem de número de acessos em um website cliente que possui diversos servidores espalhados pelo mundo os quais atualizam a todo instante o cliente.

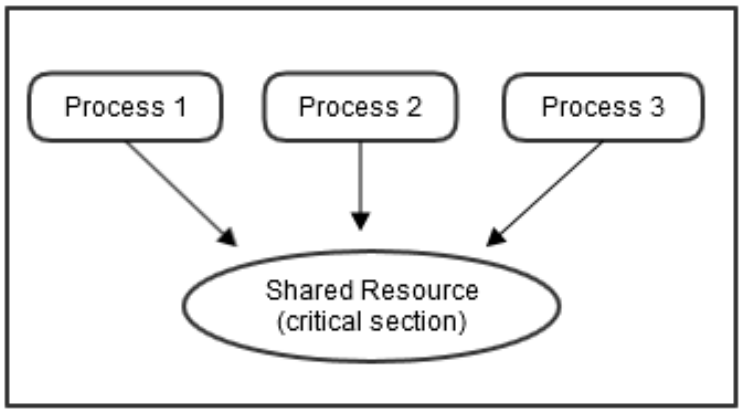

Figure 1. Representação condição de corrida

Suponha que seja retornado para o usuário ou o administrador do website determinado número de acessos em determinado instante. Como vários servidores acessam a mesma variável de contagem de acessos no lado do cliente, haveriam muitas vezes inconsistência nos resultados obtidos. Caso os donos do sistema quisessem controlar o número de acessos e seus horários para fins de marketing, por exemplo, acabariam obtendo dados conflitantes devido a condição de corrida gerada no acesso conjunto de servidores.

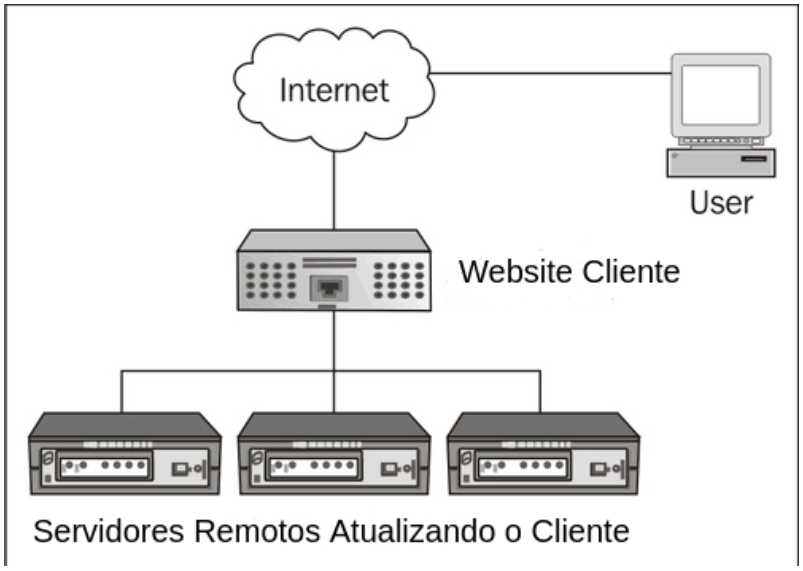

Figure 2. Representação do problema abordado

Logo, o cliente irá obter dados inconsistentes causados pela condição de corrida dos servidores que concorrem o acesso da variável do cliente de contagem de acessos.

2. Fundamentação Teórica

O algoritmo de Peterson é um algoritmo de programação concorrente para exclusão mútua, que permite a dois ou mais processos ou subprocessos compartilharem um recurso sem conflitos, utilizando apenas memória compartilhada para a comunicação. Ele foi formulado por Gary L. Peterson em 1981. Embora a formulação original de Peterson funcionasse apenas com dois processos, o algoritmo pode ser estendido para mais processos [Peterson 1981].

Na solução de Peterson, podemos observar que temos duas variáveis compartilhadas: um sinalizador booleano [i] inicializado para FALSE, ou seja, inicialmente ninguém está interessado em entrar na seção crítica e um inteiro turn o qual representa o processo cuja vez é entrar na seção crítica [Hofri 1990].

Combinando a ideia de alternar a vez (com a variável turn) com a ideia de variáveis de trava e de advertência, T. Dekker, um matemático holandês, desenvolveu uma solução de software para o problema de exclusão mútua que não requeira um chaveamento obrigatório [Tanenbaum 2014]. O algoritmo de Dekker é um algoritmo de programação concorrente para exclusão mútua, permitindo que dois processos ou threads partilhem um recurso sem conflitos. Foi um dos primeiros algoritmos mutuamente exclusivos inventados, implementados por Edsger Dijkstra. Se ambos os processos tentarem acessar a seção

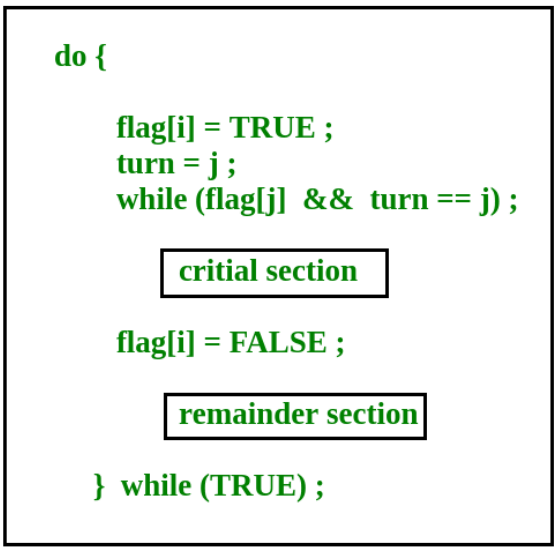

Figure 3. Representação simplificada do algortimo de Peterson

crítica simultaneamente, o algoritmo escolherá um processo de acordo com uma variável de deslocamento. Se o outro processo estiver em execução na seção crítica, você deverá aguardar sua conclusão.

O algoritmo de exclusão mútua distribuída da Lamport é um algoritmo baseado em permissão proposto por Lamport como uma ilustração de seu esquema de sincronização para sistemas distribuídos. O carimbo de data/hora baseado em permissão é usado para solicitar solicitações de seção críticas e resolver qualquer conflito entre solicitações. No algoritmo de Lamport, as solicitações de seção crítica são executadas na ordem crescente de registros de data e hora, ou seja, uma solicitação com registro de data e hora menor recebe permissão para executar a seção crítica primeiro do que uma solicitação com registro de data e hora maior.

3. Metodologia

Para execução da simulação do problema utilizamos a linguagem python (1991) em um ambiente Linux controlado com a distribuição Ubuntu versão 16.04 LST em um computador com 4GB de RAM e processador i5 onde foram implementados tanto os algoritmos, quanto as threads de simulação dos servidores em concorrência, inicialmente apenas dois, e a váriavel cliente a qual esses processos de servidores acessam e modificam no decorrer da simulação.

A quantidade de visitas que representa a contagem de acessos ao website é definida como uma váriavel global que inicia zerada e a qual é incrementada conforme executamos o método adicionar visita e representa um acesso por um servidor durante a simulação. Um método chamado rodar contador registra as visitas e gera dados reais que resultam quase sempre em falhas na consistencia dos dados as quais são corrigidas somente quando utilizada a lógica dos algoritmos para exclusão mútua.

Para a execução da exclusão mútua adicionamos visitas à contagem de acessos do website cliente a partir de um método especial para cada algoritmos e no momento de rodar o contador também é chamado um método especial de rodar contador que utiliza

Table 1. Detalhes dos Experimentos

Variável	Valor
Quantidade de servidores	3
Número de visitas	25.000
Número de experimentos	20

tanto o adicionar visitar do algoritmo quanto as visitas reais para fins de comparação da veracidade do dado.

4. Resultados

No final o programa retorna uma tabela com as visitas computadas e o número real de visitas naquele instante e se ocorreu acerto ou erro, percebe-se que sem utilizar algoritmos de exclusão mútua a tabela retorna valores errados conforme a imagem 4. É possível notar que em 15% das vezes a qual a variável visitas que representa a contagem de visitas do website cliente diferiu da contagem real, ou seja, retornou um valor errado por conta da condição de corrida que a concorrência dos servidores gerou. O tempo de experimento médio foi de apenas 0.0162 segundos apenas.

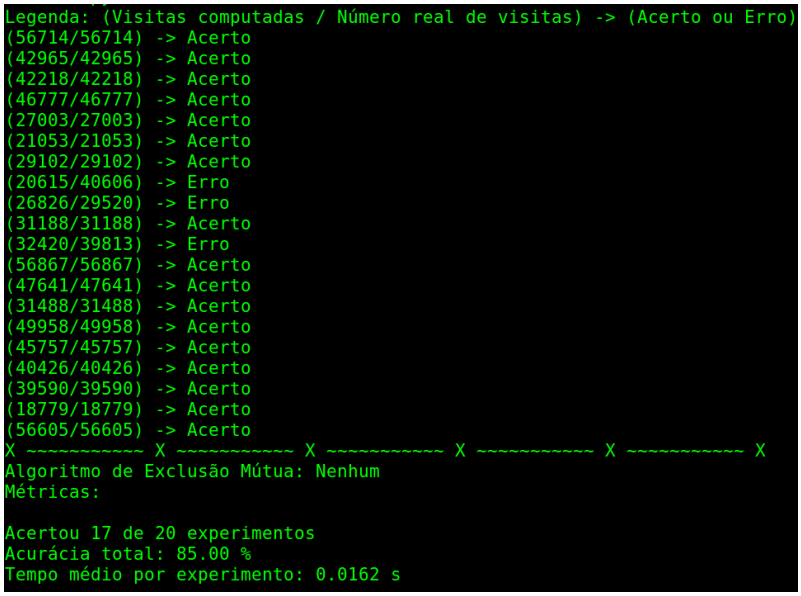

Figure 4. Exemplo de output sem utilizar algoritmos para exclusão mútua

Para solução do problema implementamos algoritmos de exclusão mútua e conforme pode ser visto na figura 5 todos os experimentos retornam acerto e a contagem correta de visitas. Tanto os algoritmos de Peterson, Dekker e Lamport obtiveram êxito de 100% nos acertos da variável de contagem de visitas, logo em nenhum momento o website apresentou uma contagem diferente da real.

No output utilizando o algoritmo de Peterson é possível observar que houve acerto em 100% dos experimentos, ou seja, de todos os acessos ao website em nenhum momento se obteve um valor errado, apesar disso o tempo médio por experimento obteve crescimento e terminou com o valor de 1.8624 segundos conforme a figura 5.

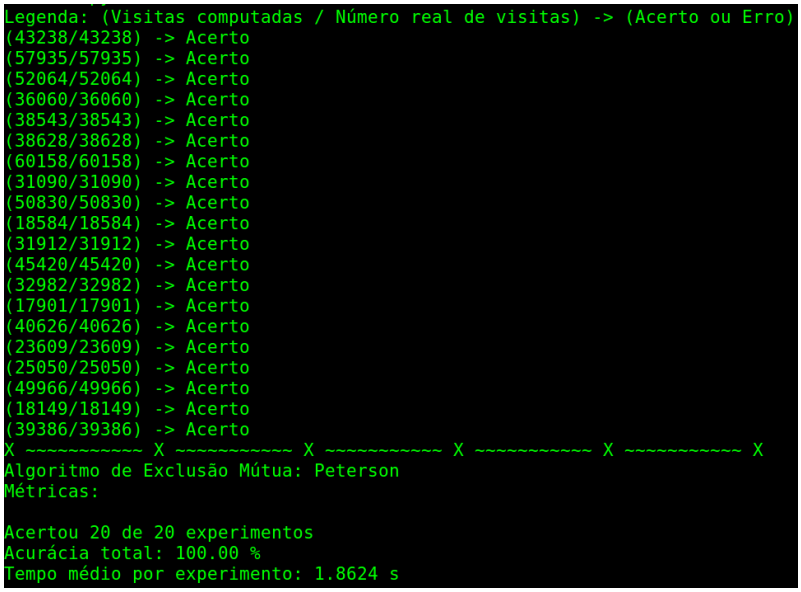

Figure 5. Exemplo de output do programa utilizando algoritmo de Peterson

O algoritmo de dekker também obteve 100% de acurácia na contagem de visitas, não obstante também foi o algoritmo de melhor desempenho registrando 0.3105 segundos de tempo médio nos experimentos. É possível ver o output geral do algoritmo de dekker na figura 6.

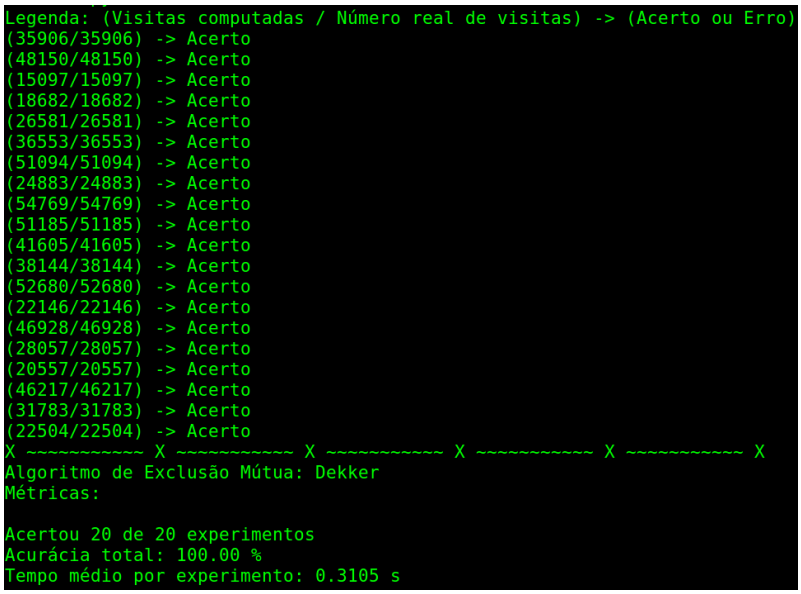

Figure 6. Exemplo de output do programa utilizando algoritmo de Dekker

O algoritmo de lamport obteve como os demais 100% de acurácia, contudo foi o algoritmo com pior performance sendo o mais demorado a executar. O tempo médio na execução dos experimentos em lamport foi de 2.0737 segundos sendo bastante maior que o tempo médio obtido no algoritmo de Dekker.

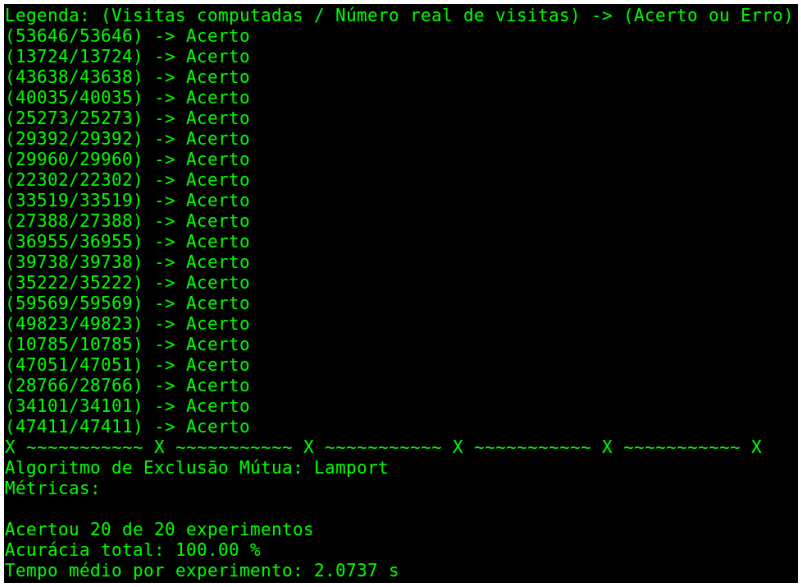

Figure 7. Exemplo de output do programa utilizando algoritmo de Lamport

Table 2. Resultados finais

Algoritmo	Acurácia	Tempo médio
Dekker	100%	0.3105 s
Peterson	100%	1.8624 s
Lamport	100%	2.0737 s
Nenhum	85%	0.0162 s

5. Conclusão

Neste trabalho apresentamos um problema relacionado a um website cliente que é alimentado por três servidores distribuídos e implementamos algoritmos de exclusão mútua para tratar o problema da condição de corrida na contagem de acessos do website. Experimentamos com e sem algoritmos e podemos afirmar que a aplicação de tais algoritmos eliminaram as inconsistências e valores errados de retorno do website gerando sempre os valores corretos da contagem de acessos e o algoritmo de Dekker foi dos três algoritmos o que levou menos tempo para a execução dos experimentos.

References

Hofri, M. (1990). *Operating Systems Review*. Prentice Hall.

Jr., L. L. (2006). *Material de aula de Sistemas operacionais - Processos e Threads*. PUCPR, 1th edition.

Peterson, G. L. (1981). *Myths About the Mutual Exclusion Problem*. Prentice Hall.

Tanenbaum, A. S. (2006). *Distributed Systems: Principles and Paradigms*. Prentice Hall, 1th edition.

Tanenbaum, A. S. (2014). *Modern Operational Systems*. Information Processing Letters, 4th edition.