UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

MARCUS FARIAS E
GUILHERME IEPSEN

COMPARAÇÃO DE DESEMPENHO ENTRE A BIBLIOTECA PTHREAD E O ALGORÍTIMO DA PADARIA DE LAMPORT

Porto Alegre, Rio Grande do Sul, Brasil
22 de abril de 2025

O presente trabalho tem como objetivo comparar a eficiência e a correção de primitivas de exclusão mútua implementadas em software, utilizando o algoritmo da padaria, com as primitivas fornecidas pela biblioteca POSIX Threads (pthread). Para isso, desenvolvemos uma biblioteca contendo as funções de *lock* e *unlock* baseadas no algoritmo da padaria e utilizamos o mesmo programa principal que utiliza tanto essa biblioteca quanto as funcionalidades de exclusão mútua da pthread para proteger o acesso a uma variável compartilhada por múltiplas threads. A análise comparativa foi realizada através da observação dos resultados da execução dos programas e da medição do tempo de execução.

O algoritmo da padaria garante a exclusão mútua através da atribuição de "tickets" aos processos que desejam acessar a seção crítica. Cada processo escolhe um número que é maior que todos os números atualmente em uso e, em caso de empate, o processo com o menor identificador tem prioridade. Um programa principal em C foi desenvolvido para criar múltiplas threads. Cada thread executa um loop que incrementa uma variável global compartilhada muitas vezes. Duas versões desse programa foram criadas: uma utilizando as funções lock e unlock da nossa biblioteca baseada no algoritmo da padaria, e outra utilizando a biblioteca pthread para proteger o acesso à variável compartilhada.

Como resultado, esperávamos que a variável compartilhada tivesse o mesmo valor final para ambas as execuções. Entretanto, notamos que houve uma leve diferença na soma da variável global, conforme os prints abaixo:

```
----------------------------
| EXECUTING PTHREAD PROGRAM |
----------------------------

Hello! I'm thread 1, id 135253909833280!
Hello! I'm thread 2, id 135253899347520!
Hello! I'm thread 3, id 135253888861760!
Joined with thread 1, id 135253909833280
Joined with thread 2, id 135253899347520
Joined with thread 3, id 135253888861760
Global var: 9000000
0.59user 0.59system 0:00.42elapsed 280%CPU (0avgtext+0avgdata 1536maxresident)k
0inputs+0outputs (0major+79minor)pagefaults 0swaps
----------------------------
| EXECUTING LAMPORT PROGRAM |
----------------------------

Hello! I'm thread 0, id 132881806526016!
Hello! I'm thread 1, id 132881796040256!
Hello! I'm thread 2, id 132881785554496!
Joined with thread 0, id 132881806526016
Joined with thread 1, id 132881796040256
Joined with thread 2, id 132881785554496
Global var: 8970398
3.56user 0.00system 0:01.18elapsed 299%CPU (0avgtext+0avgdata 1536maxresident)k
0inputs+0outputs (0major+151minor)pagefaults 0swaps
```

TIME (s)	PROGRAMA PTHREAD	PROGRAMA LAMPORT
USER	0.59	3.56
SYSTEM	0.59	0.00
ELAPSED	0.42	1.18

Observamos uma diferença significativa no tempo de execução entre os dois programas. O programa que utiliza as primitivas de exclusão mútua da biblioteca pthread demonstrou um tempo decorrido (ELAPSED) consideravelmente menor (0.42 segundos) em comparação com o programa que utiliza a nossa implementação do algoritmo da padaria (1.18 segundos). Isso indica que as primitivas de sincronização fornecidas pela biblioteca pthread são muito mais eficientes. Essa diferença de desempenho pode ser atribuída às otimizações realizadas a nível de kernel e ao gerenciamento eficiente de recursos pelo sistema operacional, que geralmente superam implementações em espaço de usuário como a do algoritmo da padaria, que pode envolver um maior número de operações e comparações. Além disso, ao analisar os valores finais da variável global, notamos uma inconsistência no programa que utiliza a implementação do algoritmo da padaria (897098) em comparação com o programa que utiliza a pthread (90000). Embora o valor esperado fosse um múltiplo do número de threads multiplicado pelo número de incrementos por thread, a discrepância sugere que, apesar da tentativa de implementar a exclusão mútua, ocorreram condições de corrida que levaram à perda de algumas atualizações na variável compartilhada.

Os resultados demonstram a clara superioridade da biblioteca pthread em termos de desempenho e confiabilidade para exclusão mútua, especialmente em arquiteturas *multi-core* onde o gerenciamento de concorrência é crucial. A implementação do algoritmo da padaria em software apresentou ineficiência no tempo de execução e falhas na garantia da exclusão mútua devido à falta de controle a nível de kernel sobre o acesso à memória, evidenciadas pela inconsistência no valor final da variável compartilhada. Isso reforça a importância de utilizar as soluções nativas do sistema operacional, como a pthread, que são otimizadas para lidar com a complexidade do paralelismo moderno.