Comparação de Utilização da CPU em Diferentes Mecanismos de Sincronização

Sistemas Operacionais

27 de junho de 2025

1 Introdução

Este documento apresenta uma representação visual da utilização da CPU quando diferentes mecanismos de sincronização são utilizados na resolução do problema do produtor-consumidor. As visualizações demonstram como cada abordagem afeta o comportamento dos núcleos da CPU, trocas de contexto, interrupções e utilização geral dos recursos.

2 Comparação Visual de Utilização da CPU

2.1 Mutex com Busy Waiting

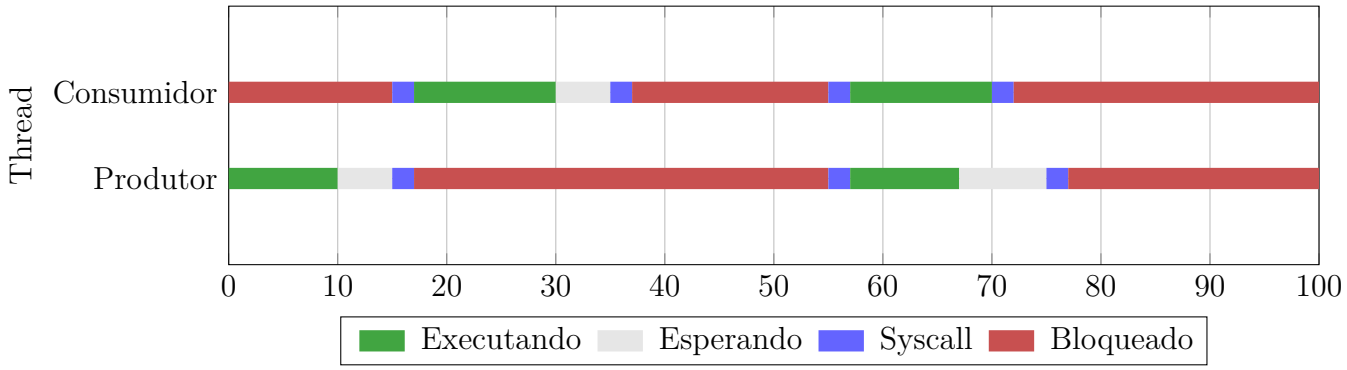

Figura 1: Mutex com Busy Waiting: Alto número de chamadas de sistema e transições entre estados

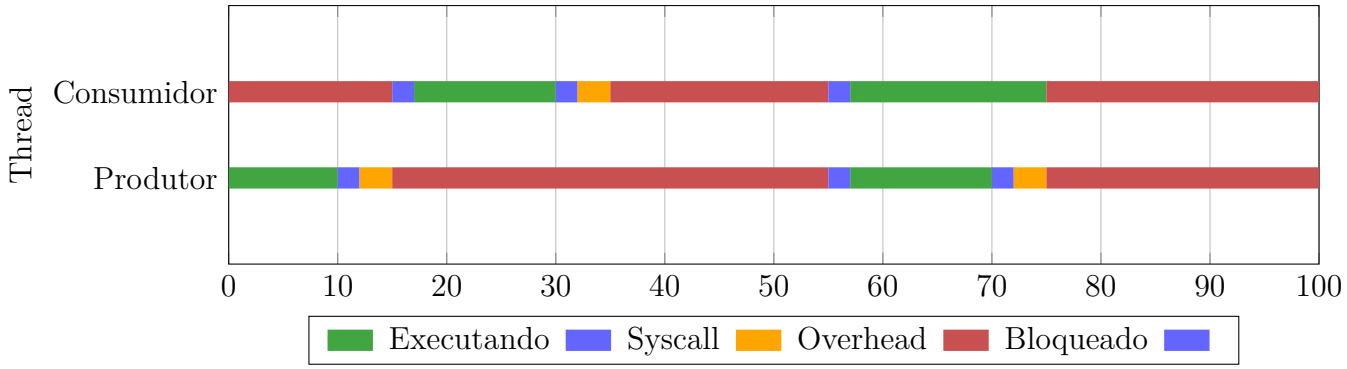

Figura 2: Mutex com Sleep: Menos chamadas de sistema, mas ainda tem overhead de verificação periódica

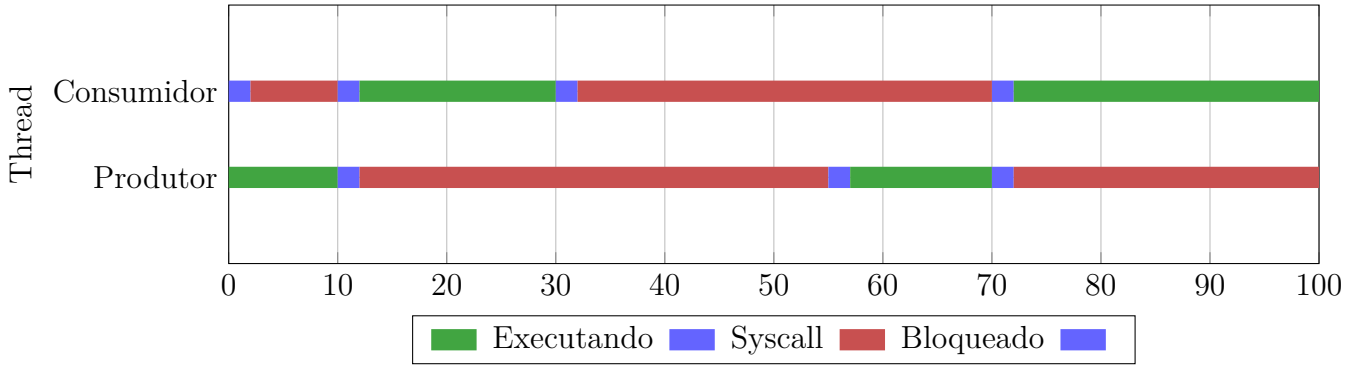

Figura 3: Semáforos: Bloqueio eficiente com sinalização direta quando recursos estão disponíveis

2.2 Mutex com Timed Sleep

2.3 Semáforos

2.4 Variáveis de Condição

3 Comparativo de Interrupções e Trocas de Contexto

4 Utilização de CPU por Tipo de Operação

5 Análise de Impacto no Cache

6 Conclusão

As representações visuais apresentadas neste documento ilustram claramente por que abordagens como semáforos e variáveis de condição geralmente oferecem melhor desempenho do que soluções baseadas apenas em mutex, especialmente em cenários com alta contenção:

- **Mutex com busy waiting**: Gera alto número de interrupções, chamadas de sistema e desperdício de ciclos de CPU.

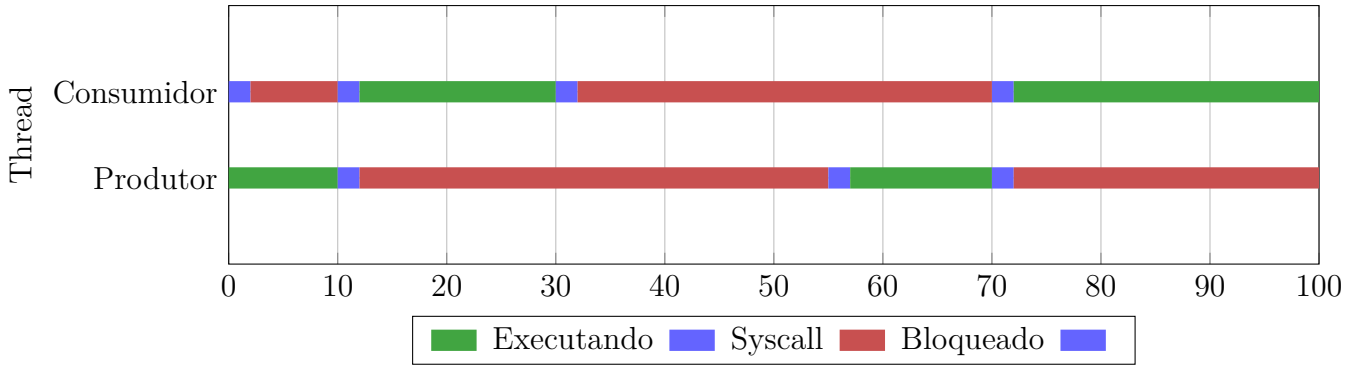

Figura 4: Variáveis de Condição: Padrão similar aos semáforos, mas com maior flexibilidade para condições complexas

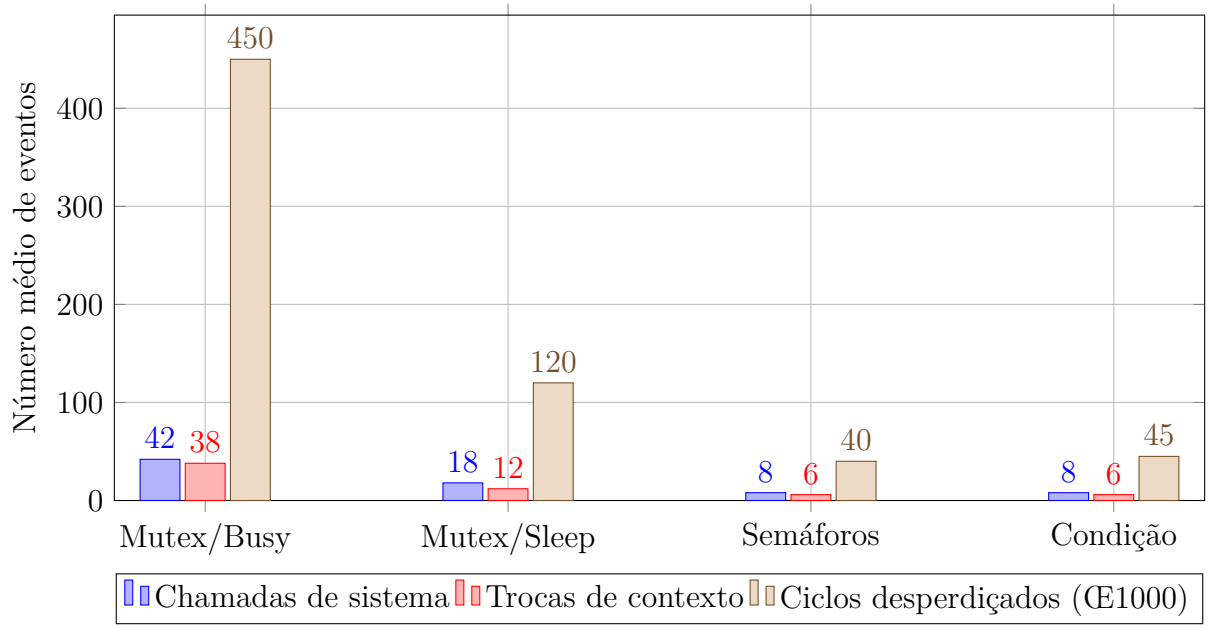

Figura 5: Comparativo de eventos de sistema entre as diferentes abordagens de sincronização

- **Mutex com sleep**: Reduz o desperdício de CPU em comparação ao busy waiting puro, mas ainda tem overhead significativo de verificação periódica.
- **Semáforos e variáveis de condição**: Minimizam chamadas de sistema e trocas de contexto através de bloqueio eficiente e sinalização direta.

No problema do produtor-consumidor, onde frequentemente threads precisam esperar que outras completem certas operações, os mecanismos que permitem bloqueio eficiente e sinalização direta entre threads oferecem vantagens significativas de desempenho e utilização de recursos.

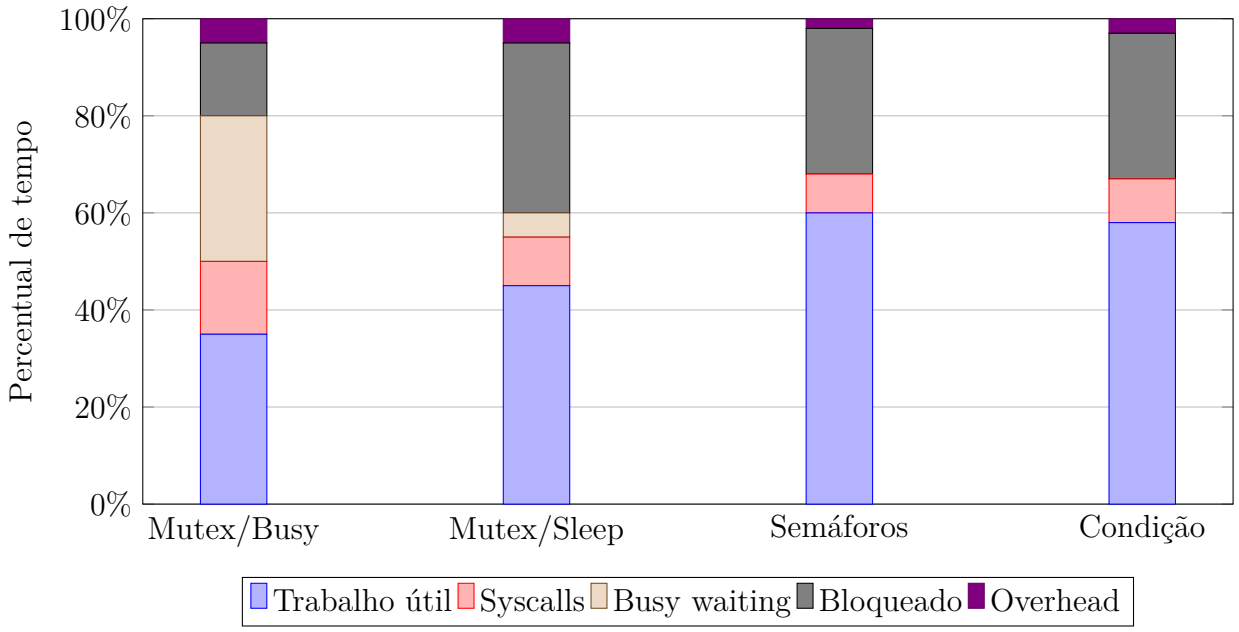

Figura 6: Distribuição do tempo de CPU entre diferentes tipos de operações

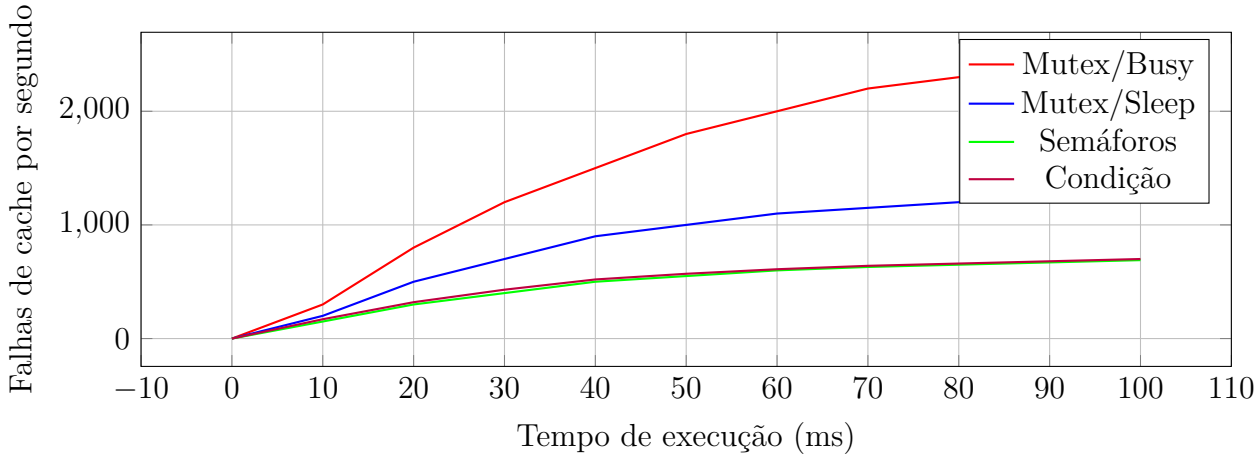

Figura 7: Falhas de cache ao longo do tempo para diferentes mecanismos de sincronização