# Threads

Prof. Msc. Rodrigo D. Malara

# THREADS

Os primeiros sistemas operacionais suportavam apenas uma tarefa por processo.

À medida em que as aplicações se tornavam mais complexas, essa limitação se tornou um claro inconveniente.

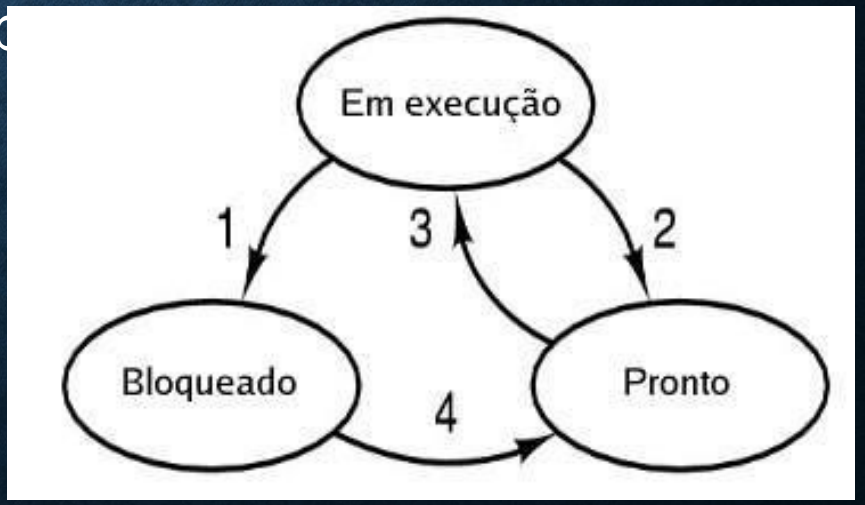
Demandas atuais evidenciaram a necessidade de suportar mais de uma tarefa operando no mesmo contexto, ou seja, dentro do mesmo processo.

É uma forma de um ***processo*** dividir a si mesmo em duas ou mais tarefas que podem ser executadas de forma concorrente.

# THREADS

## Tarefas ou processos leves

- Desacoplam os recursos da execução;
- Várias linhas de execução num único processo,  compartilhando os recursos;
- Vários threads cooperam para a execução mais  rápida do processo;
- Baixo custo computacional p
  criados;
- Possui vários estados;
  - Pronto;
  - Suspenso;
  - Em execução;



# THREADS

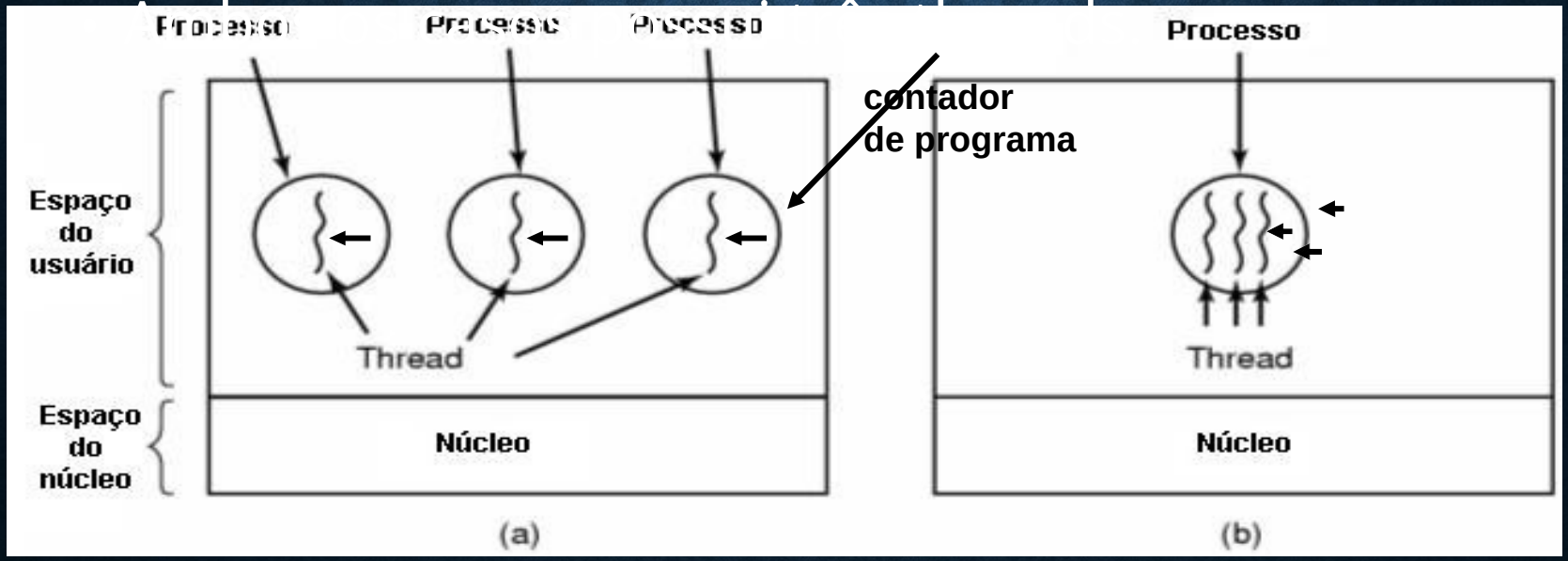
## O que são?

- É uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.
- Uma thread permite, por ex., que o usuário de um programa utilize uma funcionalidade do ambiente enquanto outras linhas de execução realizam outros  cálculos e operações

# THREADS

## Três processos tradicionais

- Cada processo tem seu próprio espaço de endereço  e uma única linha de controle, vemos também um  único processo com três linhas de controle.



# THREADS

| Recursos compartilhados | Recursos Individuais |
| --- | --- |
| Espaço de endereçamento<br>Variáveis globais<br>Arquivos abertos<br>Alarmes pendentes<br>Sinais e tratadores<br>Informações de contabilidade | Contador de programa<br>Registradores<br>Pilha<br>Estado |

- Recursos compartilhados
  - Todos os threads tem acesso a esses recursos
- Recursos individuais
  - Cada thread possui o seu

# MODELO DE THREADS

É denominado **thread**, cada fluxo de execução do sistema que é associado a um processo ou executa no interior do núcleo do SO.

Threads executando dentro de um processo são chamados de **threads de usuário**
(***user-level threads*** ou simplesmente ***user threads***).

Cada thread de usuário corresponde a uma tarefa a ser executada dentro de um processo.

Os fluxos de execução reconhecidos e gerenciados pelo núcleo do sistema operacional são chamados de ***threads de núcleo*** (***kernel-level threads*** ou ***kernel threads***).

Os threads de núcleo representam tarefas que o núcleo deve realizar.

Essas tarefas podem corresponder à execução dos processos no espaço de usuário, ou a atividades internas do próprio núcleo, como drivers de

# MODELO DE THREADS

Os sistemas operacionais mais antigos não ofereciam suporte a threads para a construção de aplicações.

Sem poder contar com o sistema operacional, os desenvolvedores de aplicações contornaram o problema construindo bibliotecas que permitiam *criar e gerenciar threads dentro de cada processo*, sem o envolvimento do núcleo do sistema.

Usando essas bibliotecas, uma aplicação pode lançar vários threads conforme sua necessidade, mas o núcleo do sistema irá sempre perceber (e gerenciar) apenas **um** fluxo de execução dentro de cada processo.

Por essa razão, esta forma de implementação de threads é nomeada **Modelo de Threads N:1,** N threads no processo, mapeados em um único thread de núcleo.

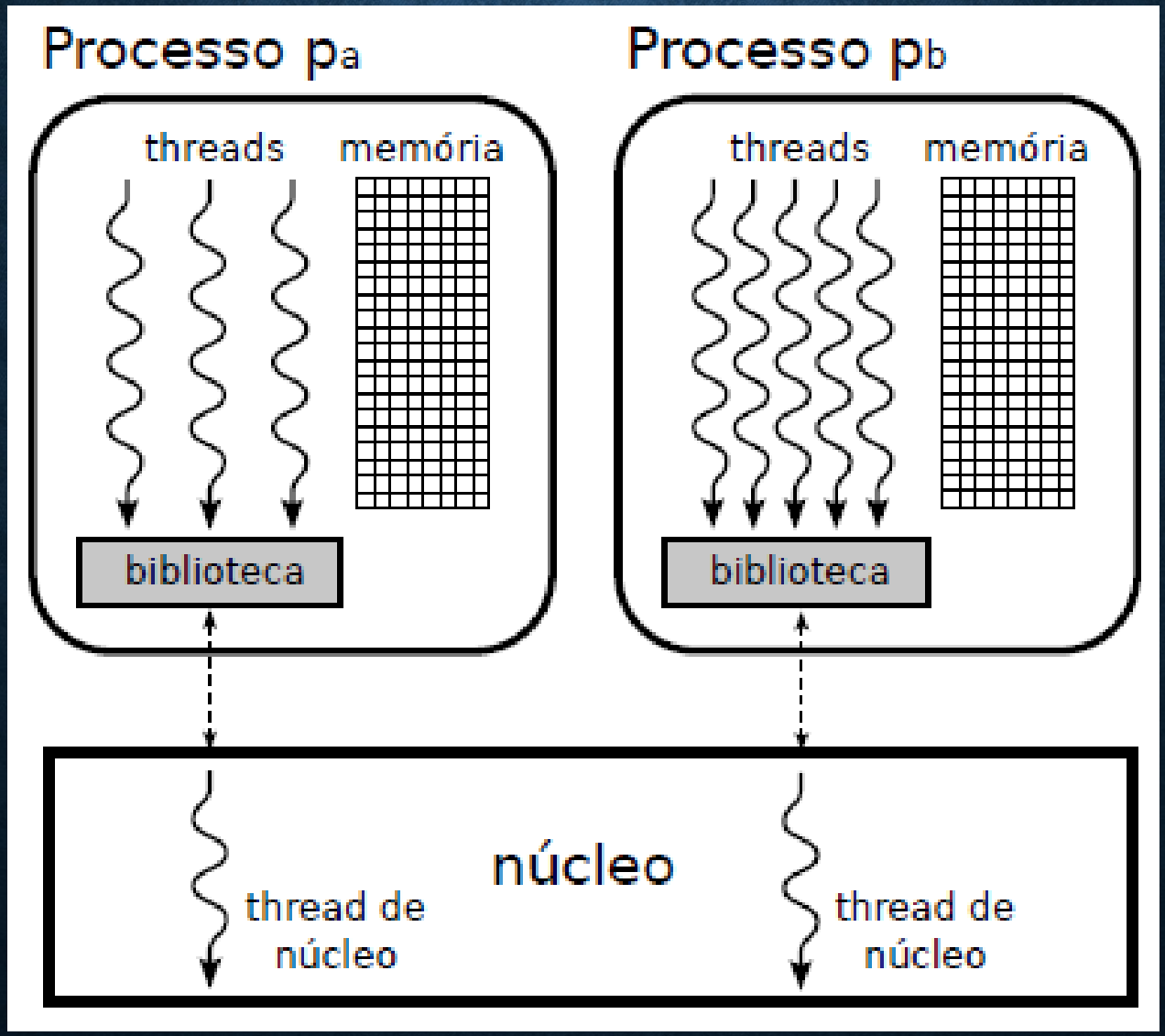
# MODELO DE THREADS N:1

O modelo de threads N:1 é muito utilizado, por ser leve e de fácil implementação.

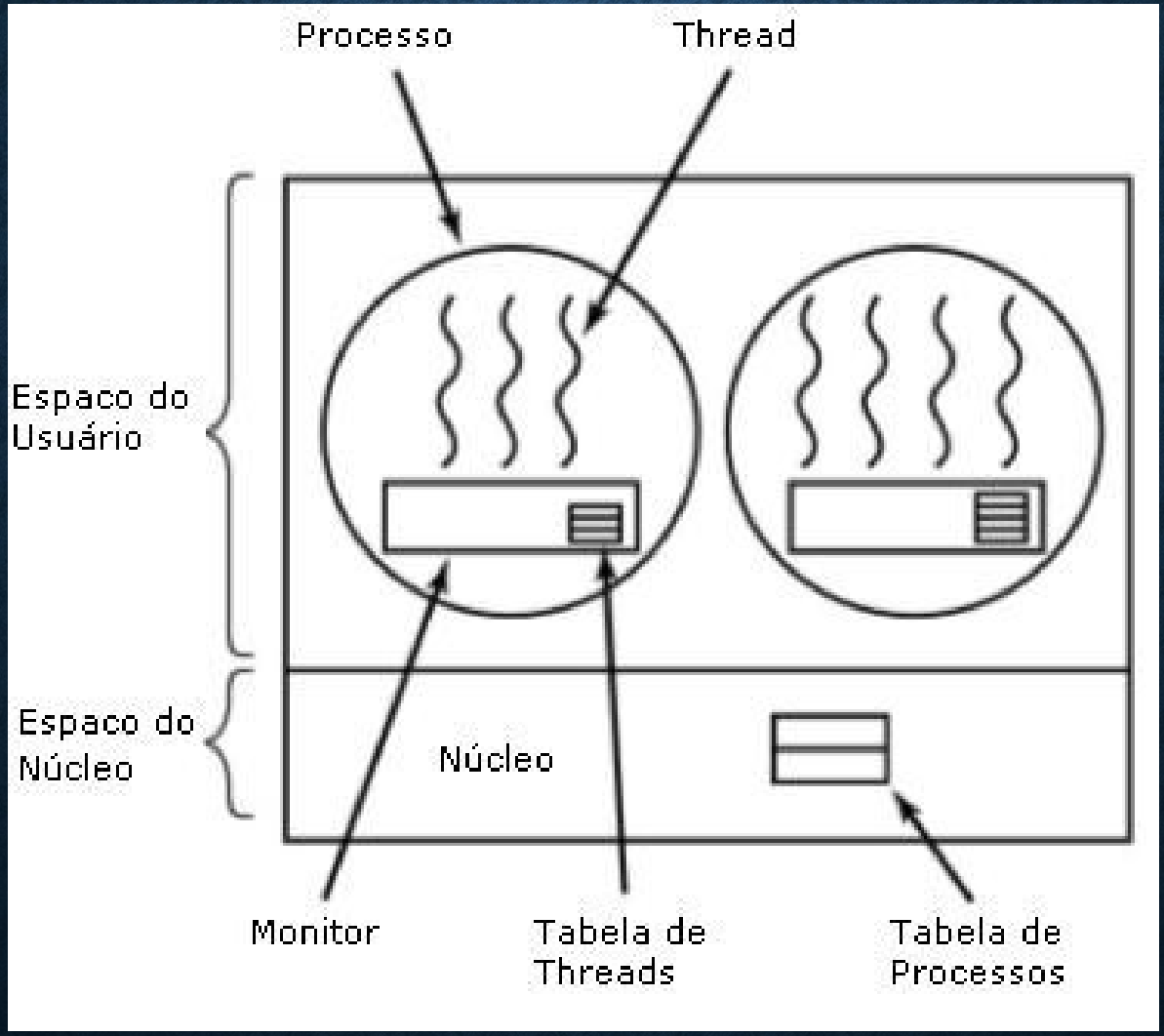
Como o núcleo somente considera uma thread, a carga de gerência imposta ao núcleo é pequena e não depende do número de threads dentro da aplicação.

Essa característica torna este modelo útil na construção de aplicações que exijam muitos threads, como jogos ou simulações de grandes sistemas (a simulação detalhada do tráfego viário de uma cidade grande, por exemplo, pode exigir um thread para

# MODELO DE THREADS
# N:1



# MODELO DE THREADS N:1




Pacote de threads no espaço do usuário

# MODELO DE THREADS N:1

Entretanto, o modelo de threads N:1 apresenta problemas em algumas situações, sendo o mais grave deles relacionado às operações de entrada/saída.

Como essas operações são intermediadas pelo núcleo, se um thread de usuário solicitar uma operação de E/S (recepção de um pacote de rede, por exemplo) o thread de núcleo correspondente será suspenso até a conclusão da operação, fazendo com que todos os threads de usuário associados ao processo parem de executar enquanto a operação não for concluída.

# MODELO DE THREADS N:1

Outro problema desse modelo diz respeito à divisão de recursos entre as tarefas.

O núcleo do sistema divide o tempo do processador entre os fluxos de execução que ele conhece e gerencia: as threads de núcleo. Assim, uma aplicação com 100 threads de usuário irá receber o mesmo tempo de processador que outra aplicação com apenas um thread (considerando que ambas as aplicações têm a mesma prioridade). Cada thread da primeira aplicação irá portanto receber 1/100 do tempo que recebe o thread único da segunda aplicação, o que não pode ser considerado uma divisão justa desse recurso.

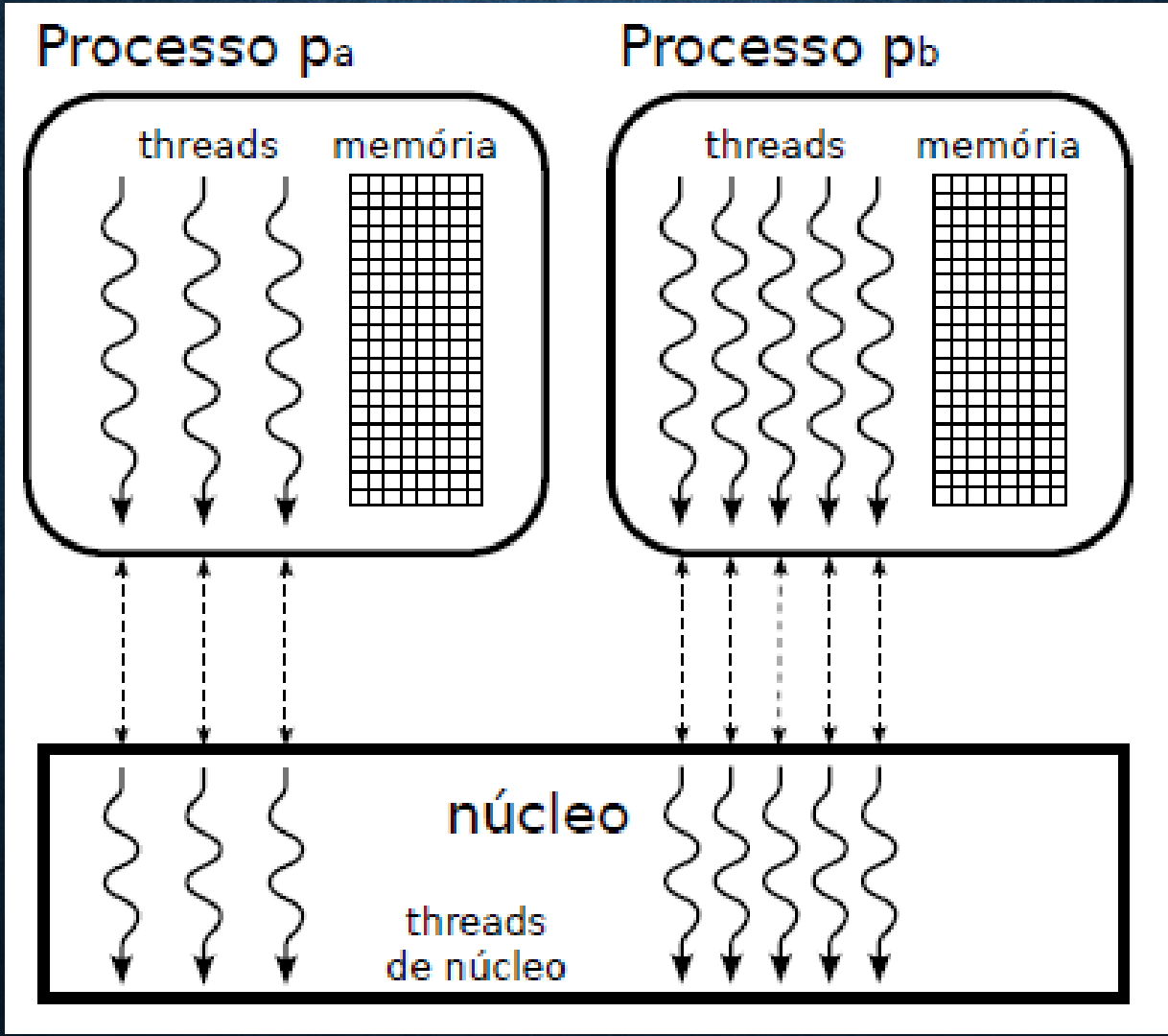
# MODELO DE THREADS 1:1

A necessidade de suportar aplicações com vários threads (*multithreaded*) levou os desenvolvedores de sistemas operacionais a incorporar a gerência dos threads de usuário ao núcleo do sistema. Para cada thread de usuário foi então definido um thread correspondente dentro do núcleo, suprimindo com isso a necessidade de bibliotecas de threads.

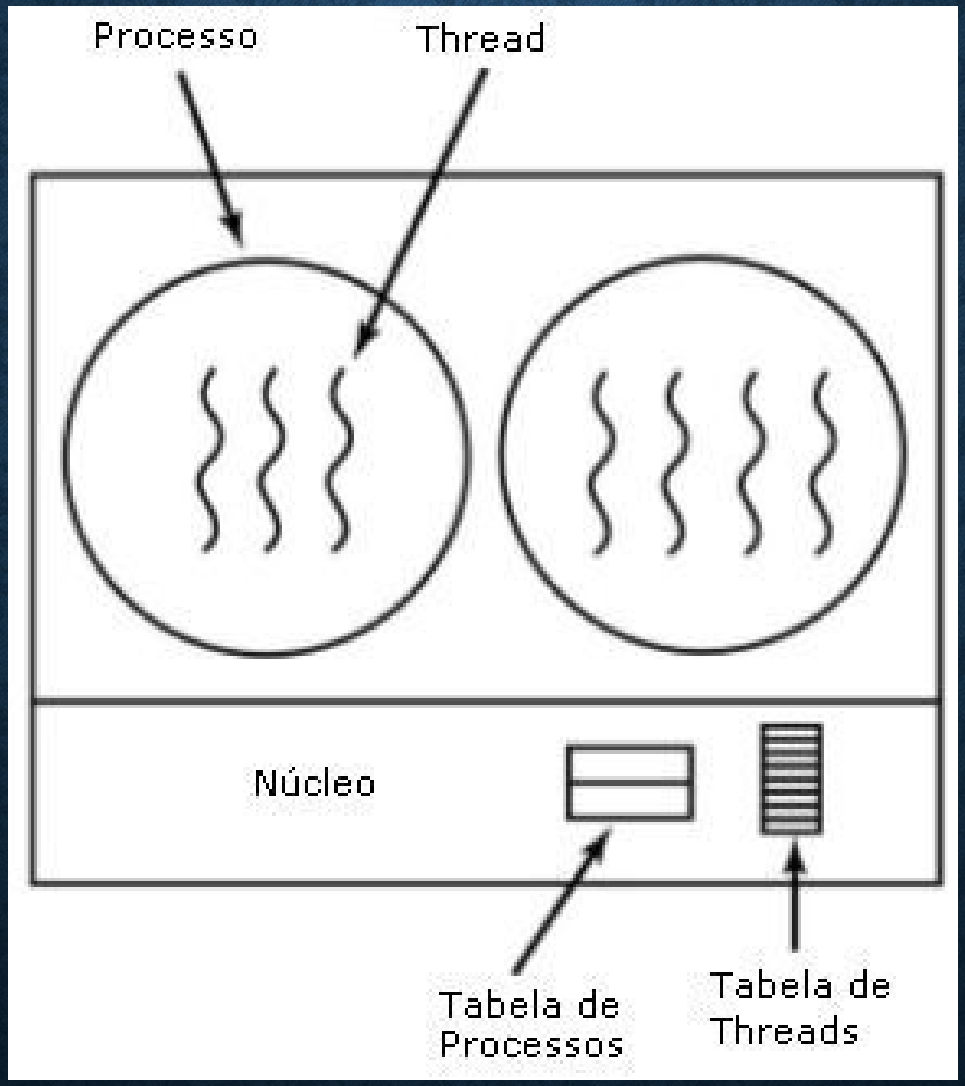
Caso um thread de usuário solicite uma operação bloqueante (leitura de disco ou recepção de pacote de rede, por exemplo), somente seu respectivo thread de núcleo será suspenso, sem afetar os demais threads.

Além disso, caso o hardware tenha mais de um processador, mais threads da mesma aplicação podem executar ao mesmo tempo, o que não era possível no modelo anterior. Essa forma de implementação, denominada **Modelo de Threads 1:1** é a mais frequente nos sistemas operacionais atuais.

# MODELO DE THREADS 1:1



# MODELO DE THREADS 1:1




Threads gerenciados pelo Núcleo

# MODELO DE THREADS N:M

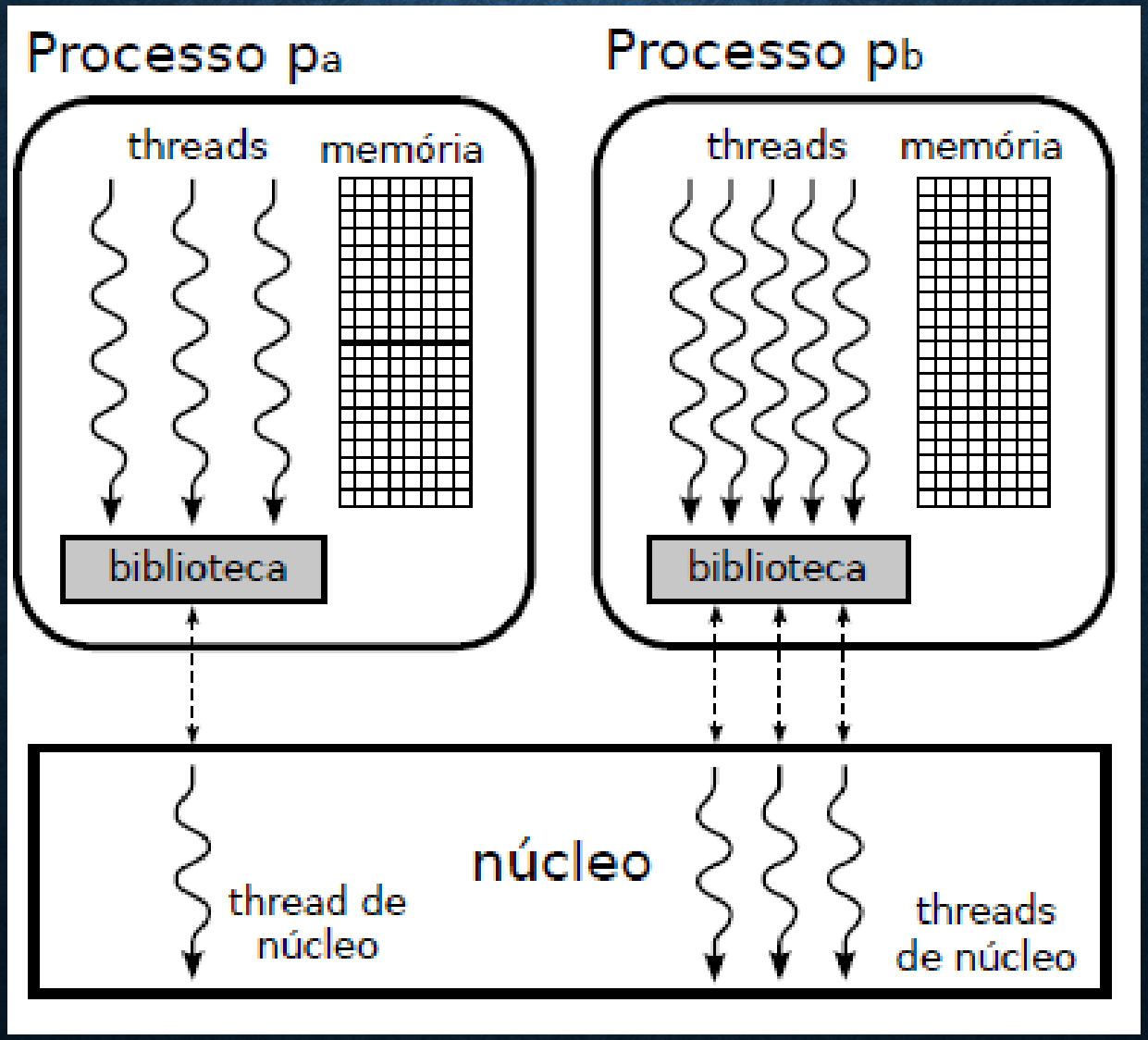
O modelo de threads 1:1 é adequado para a maioria das situações e atende bem às necessidades das aplicações interativas e servidores de rede. No entanto, é pouco escalável: a criação de um grande número de threads impõe um carga significativa ao núcleo do sistema, inviabilizando aplicações com muitas tarefas (como grandes servidores Web e simulações de grande porte).

Para resolver o problema da escalabilidade, alguns sistemas operacionais implementam um modelo híbrido, que agrega características dos modelos anteriores. Nesse novo modelo, uma biblioteca gerencia um conjunto de threads de usuário (dentro do processo), que é mapeado em um ou mais threads do núcleo.

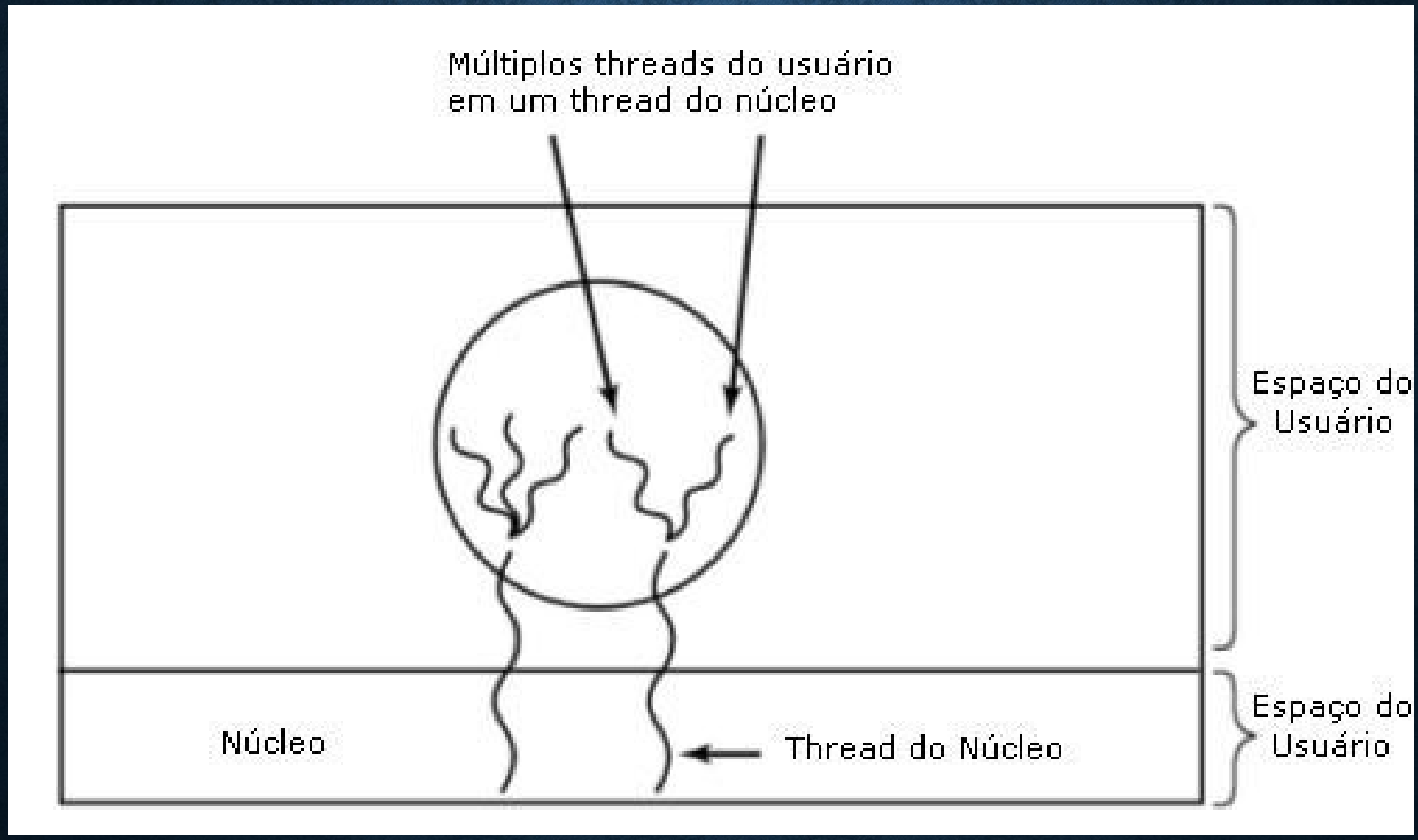
O conjunto de threads de núcleo associados a um processo é geralmente composto de um thread para cada tarefa bloqueada e mais um thread para cada processador disponível, podendo ser ajustado dinamicamente conforme a necessidade da aplicação.

Essa abordagem híbrida é denominada Modelo de Threads N:M, onde N threads de usuário são mapeados em M <= N threads de núcleo.

# MODELO DE THREADS
# N:M



# MODELO DE THREADS
# N:M OU IMPLEMENTAÇÕES HÍBRIDAS




Multiplexar threads no espaço do usuário dentro de threads do núcleo

# QUADRO COMPARATIVO DOS MODELOS DE THREADS

| Modelo | N:1 | 1:1 | N:M |
|---|---|---|---|
| Resumo | Todos os N *threads* do processo são mapeados em um único *thread* de núcleo | Cada *thread* do processo tem um *thread* correspondente no núcleo | Os N *threads* do processo são mapeados em um conjunto de M *threads* de núcleo |
| Local da implementação | bibliotecas no nível usuário | dentro do núcleo | em ambos |
| Complexidade | baixa | média | alta |
| Custo de gerência para o núcleo | nulo | médio | alto |
| Escalabilidade | alta | baixa | alta |
| Suporte a vários processadores | não | sim | sim |
| Velocidade das trocas de contexto entre *threads* | rápida | lenta | rápida entre *threads* no mesmo processo, lenta entre *threads* de processos distintos |
| Divisão de recursos entre tarefas | injusta | justa | variável, pois o mapeamento *thread*→ processador é dinâmico |
| Exemplos | GNU Portable Threads | Windows XP, Linux | Solaris, FreeBSD KSE |

# USO DE THREADS

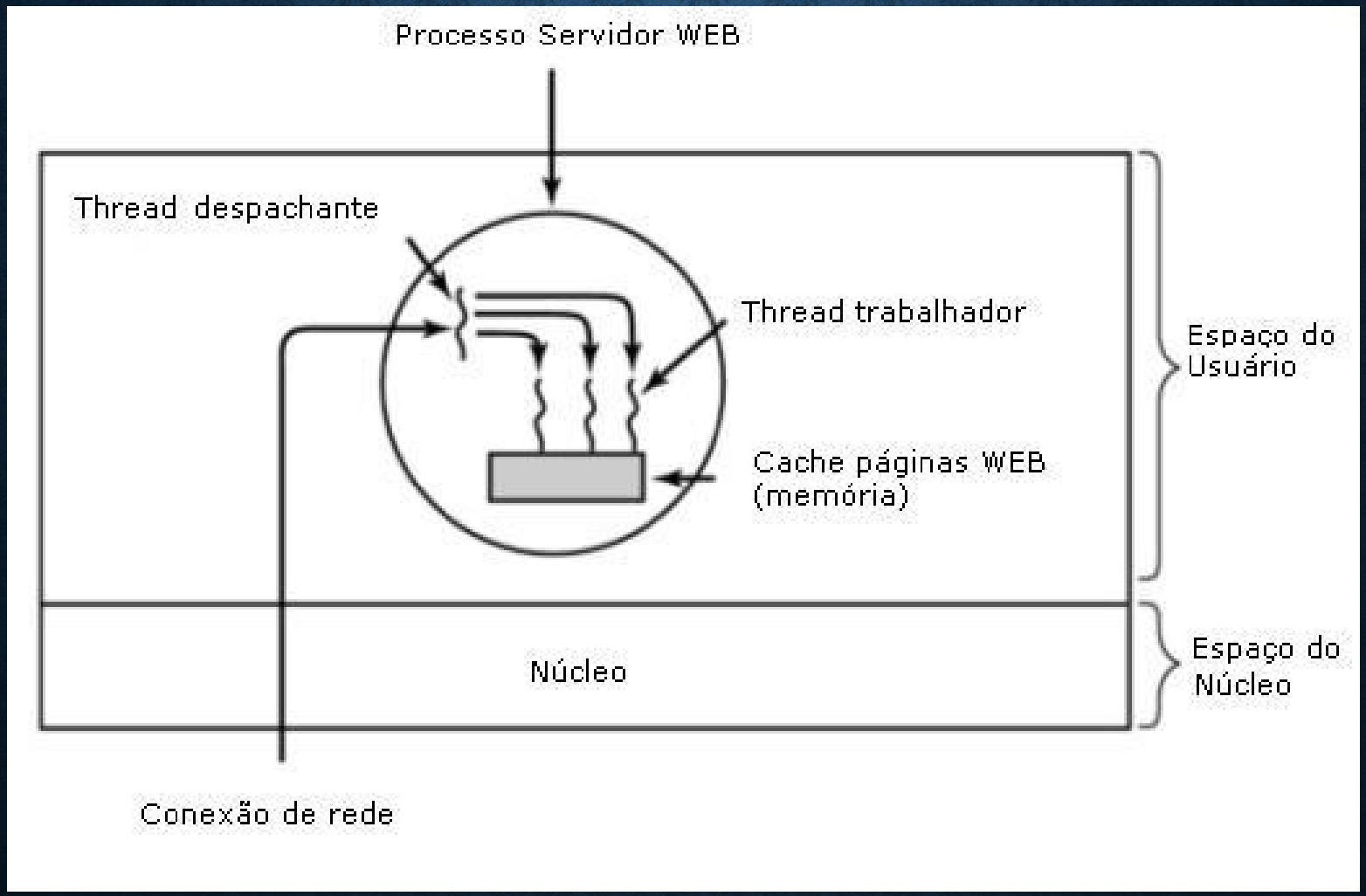
Threads são úteis nos navegadores para World Wide Web.

Páginas Web contêm várias pequenas imagens.

Para cada imagem em uma página da Web, o navegador deve configurar uma conexão separada com o site da página e requisitar a imagem.

Muito tempo é desperdiçado, estabelecendo e liberando todas essas conexões. Por ter múltiplos threads dentro do navegador, muitas imagens podem ser solicitadas ao mesmo tempo acelerando significamente o desempenho na maioria dos casos, uma vez que com imagens pequenas, o tempo de configuração é o fator limitante, não a velocidade da linha de transmissão.

# USO DE THREADS




Servidor WEB multithread

# THREADS: POSSÍVEIS PROBLEMAS

## Uso incorreto dos recursos compartilhados

- Ex: um thread fecha um arquivo e o outro tenta buscar dados no arquivo fechado.

## Comportamento na criação de processos multithreaded

- Dois processos com vários threads cada um, acessando e bloqueando algum dispositivo (ex: teclado).
  - Para quem os dados irão? Pai? Filho? Os dois?

# RESUMINDO…

## MODELO DE THREAD N:1
## OU THREADS NO ESPAÇO DO USUÁRIO

Usados através de bibliotecas de threads  Mais rápidos

- Sem chamada ao Sistema;
- Possibilidade de se usar diferentes políticas de escalonamento.

Problemas com bloqueio por E/S

- O processo inteiro é bloqueado (todos os threads);
- O núcleo não pode controlar o quantum de threads;
- O programador deve prever que haverão diversas chamadas para abrir mão da CPU (yield)

## RESUMINDO…
## MODELO DE THREAD 1:1
## OU THREADS NO NÚCLEO

### Maior custo computacional

- Grande mudança de contexto deixa execução mais lenta;
- Baixo aproveitamento da memória cache.

### Melhor granularidade

- Threads são preemptados quando seu quantum acaba;
- E/S bloqueia somente um thread, ao invés de um processo todo.