Sistemas Distribuídos

2016.1

PROF. MARCIAL PORTO FERNANDEZ
MARCIAL@LARCES.UECE.BR

PROF. ANDRÉ RIBEIRO CARDOSO
ANDREC@LARCES.UECE.BR

2. Processos e Threads

Sumário

Processos

- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Sumário

Processos

- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Definição de Processo (Deitel)

"Um processo é uma entidade. Cada processo tem seu próprio espaço de endereço, que normalmente consiste em uma região de texto (código), uma região de dados e uma pilha"

Espaço de endereço é a unidade de gerenciamento virtual de um processo

- Região de texto contém o código executado pelo processador
- Região de dados contém variáveis e memória alocada dinamicamente usada pelo processo durante a execução
- Região de pilha contém instruções e variáveis locais para chamadas ativas ao procedimento

Definição de Processo (Coulouris)

"Processo consiste em um ambiente de execução, junto com um ou mais threads"

Ambiente de execução é a unidade de gerenciamento de recursos locais pelo núcleo aos quais threads têm acesso

- Espaço de endereçamento (Mem. Virtual = Física + Swap)
- Recursos de sincronização e comunicação entre threads (ex. semáforos e sockets)
- Recursos de mais alto nível como arquivos e janelas abertas

Ambientes de execução representam um domínio de proteção onde threads são executadas

- Certos núcleos permitem o compartilhamento de memória física entre ambientes de execução do mesmo computador

OBS: Criação e gerenciamento de ambientes de execução são dispendiosos, porém threads podem compartilhá-los.

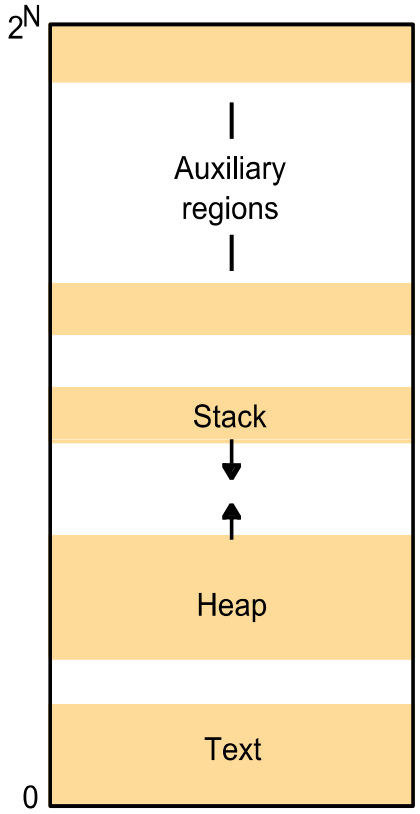
Definição de Processo (Coulouris)

Região

- Extensão
- Permissão de R-W-E para threads de processos
- Pode crescer em ambas as direções de endereçamento

Endereço Unix

- Texto (programa)
- Heap
- Região Auxiliar



Espaço de endereçamento

Definição de Processo (Tanenbaum)

“Processo é apenas um programa em execução acompanhado dos valores atuais do contador de programa, registradores e variáveis” (Tanenbaum)

Cada processo tem uma CPU virtual

- CPU troca de um processo para outro simulando paralelismo – cada processo é executado em um determinado momento

Definição de Processo (Tanenbaum)

Contador de programa contém o endereço de memória da próxima instrução a ser buscada

- 1 contador físico
- n contadores lógicos

Quando o processo executa, um contador de programa lógico é carregado no contador de programa físico (Chaveamento de contexto)

Quando acaba o tempo de CPU alocado ao processo n, o contador de programa físico é salvo no contador lógico do processo de memória

Sumário

Processos

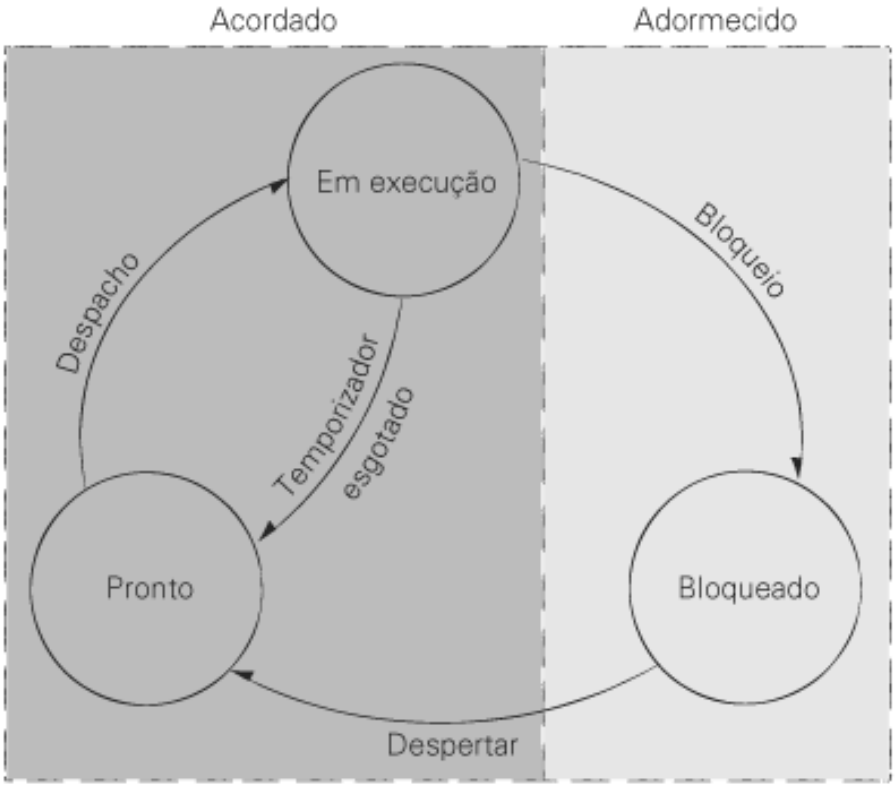
- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Estados de processo e estados de transição



Transições de estado de processo.

Blocos de Controle de Processo (PCBs)

Os PCBs mantêm as informações que o sistema operacional precisa para gerenciar o processo.

- Normalmente, eles incluem as seguintes informações:
  - Número de identificação de processo (PID)
  - Estado do processo
  - Contador de programa
  - Prioridade de escalonamento
  - Credenciais
  - Um ponteiro para o processo-pai
  - Ponteiros para os processos-filho
  - Ponteiros para localizar os dados e instruções do processo na memória
  - Ponteiros para recursos alocados

Blocos de controle de processo (PCBs)

Tabela de processos

- O sistema operacional mantém ponteiros para cada PCB do processo em uma tabela de processos no âmbito total do sistema ou por usuário.
- Permite acesso rápido aos PCBs.
- Quando um processo é encerrado, a tabela de processos retira o processo da tabela e disponibiliza todos os seus recursos.

Blocos de controle de processo (PCBs)

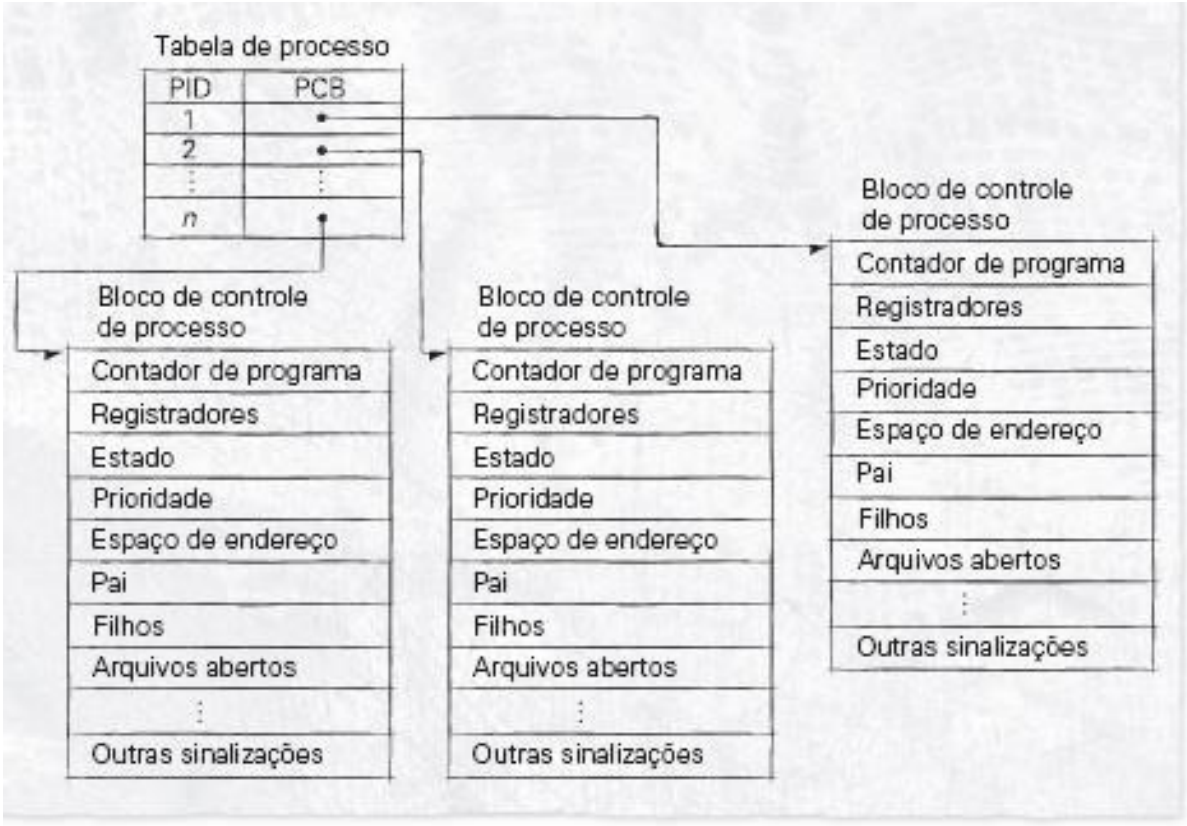


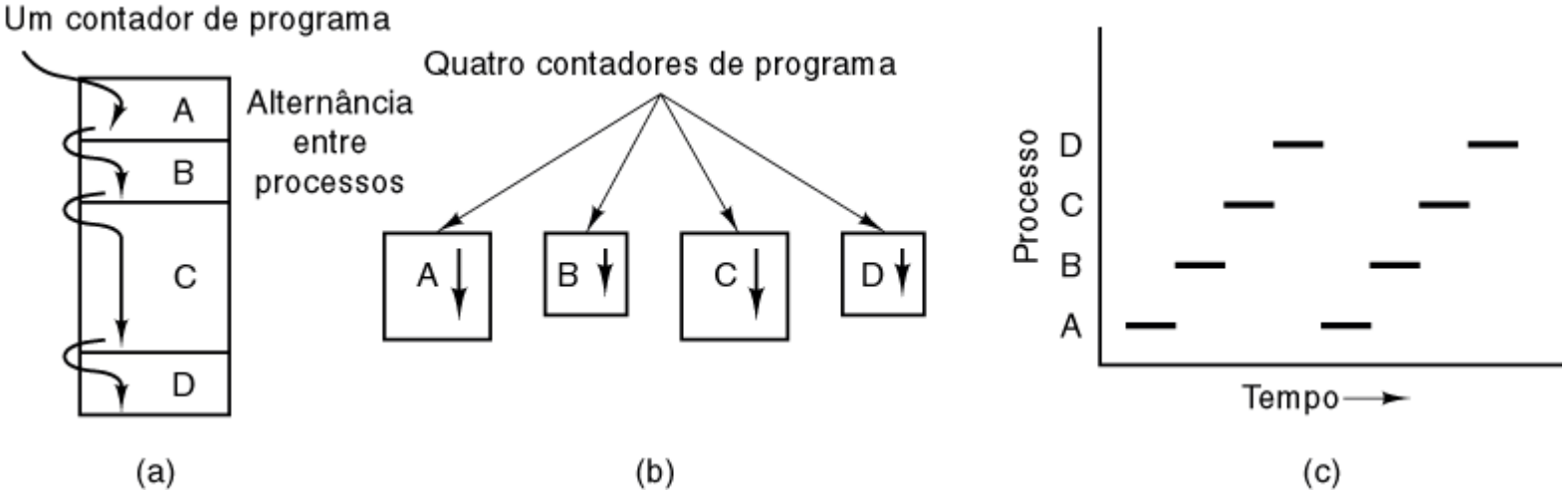
Tabela de processos e blocos de controle de processo

Processos
O Modelo de Processo

a) Modelo conceitual de 4 processos sequenciais, independentes

b) Multiprogramação de quatro programas

c) Somente um programa está ativo a cada momento



Criação/Término de Processos

Principais eventos que levam à criação de processos

- Início do sistema
- Execução de chamada ao sistema para criação de processos
- Solicitação do usuário para criar um novo processo
- Início de um programa do sistema

Condições que levam ao término de processos

- Saída normal (voluntária)
- Saída por erro (voluntária)
- Erro fatal (involuntário)
- Cancelamento por um outro processo (involuntário)

Sumário

Processos

- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Criação de Processos em SDs

Escolha de um Host de destino (transparente ao programador e ao usuário)

- Executar processos no host escolhido
- Balanceamento de carga
  - Centralizado (único componente gerenciador de carga)
  - Hierárquico (vários nós gerenciadores de carga organizados em estrutura de árvore)
  - Descentralizado (nós trocam informações diretamente uns com os outros a fim de tomar decisão de alocação)

Criação de Processos em SDs

- Políticas de Balanceamento de Carga
  - Política de transferência - processo será criado localmente ou remotamente
  - Política de localização – determina qual nó vai receber novo processo selecionado para transferência.
  - Decisão vai depender das cargas relativas dos nós, das suas arquiteturas e dos seus recursos especializados.

Criação de Processos em SDs

Criação de um novo ambiente de execução

- Espaços de endereçamento com conteúdos inicializados e, talvez, outros recursos, como arquivos padrão abertos

Duas estratégias para definir e inicializar o espaço de endereçamento

- Definido estaticamente: regiões criadas a partir de uma lista que especifica seus respectivos tamanhos (ex. Inicializadas a partir de um arquivo executável ou preenchidas com valores zero)
- Definido de acordo com um já existente: processo filho compartilha região de programa do processo pai e tem cópia das regiões de heap e pilha (ex. fork do Unix)

Sumário

Processos

- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Definição de Threads

Em SOs tradicionais, cada processo tem um espaço de endereçamento e uma só thread

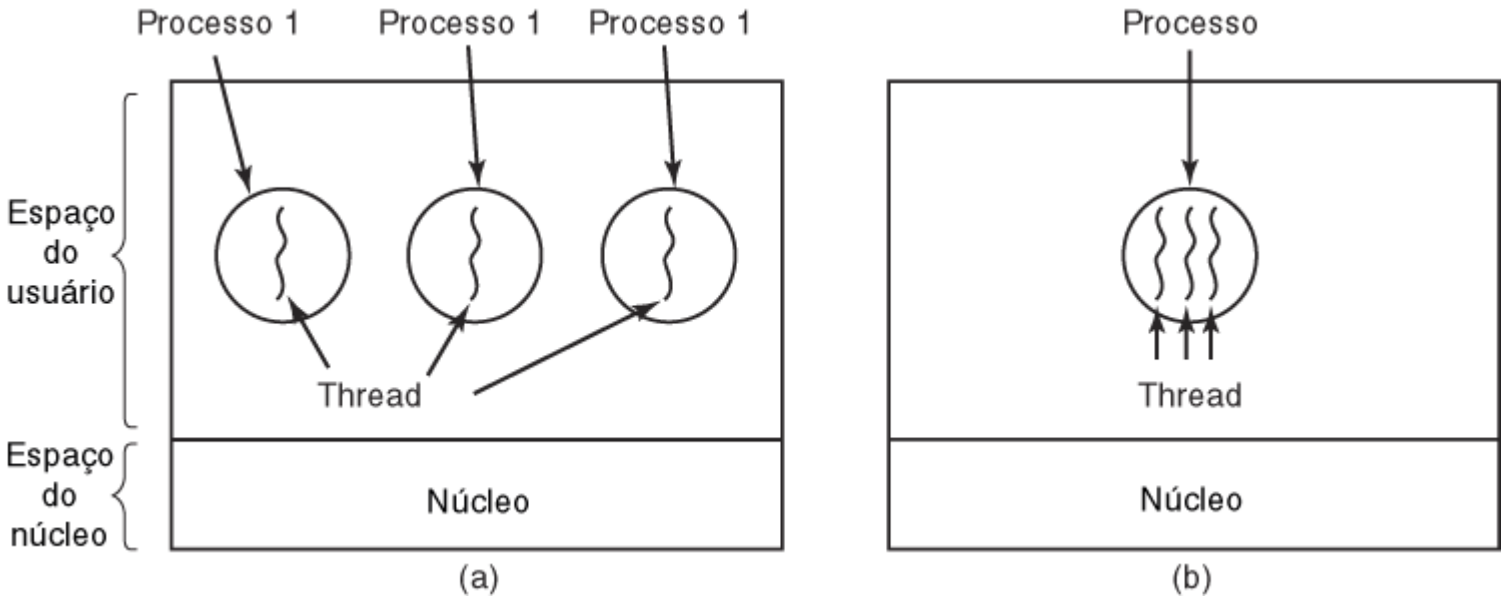
É possível ter várias threads no mesmo espaço de memória executando concorrentemente

- Threads compartilham variáveis globais
- Inexiste proteção entre threads – uma thread tem acesso ao endereço de memória dentro do espaço de endereçamento do processo (pode ler, gravar ou apagar)
- OBS: Cooperação realizada nas tarefas

Threads
O Modelo de Thread (1)

a) Três processos cada um com um thread

b) Um processo com três threads



Threads
O Modelo de Thread (2)

(a) Items compartilhados por todos os threads em um processo

(b) Itens privativos de cada thread

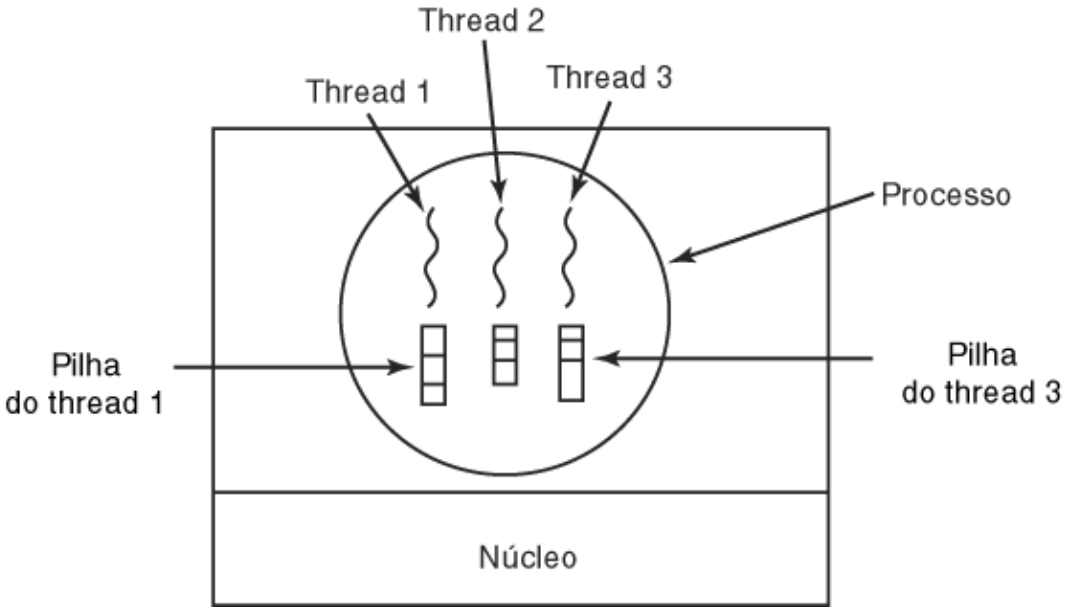
Itens por processo	Itens por thread
Espaço de endereçamento	Contador de programa
Variáveis globais	Registradores
Arquivos abertos	Pilha
Processos filhos	Estado
Alarmes pendentes	
Sinais e tratadores de sinais	
Informação de contabilidade	

(a) (b)

Threads
O Modelo de Thread (3)

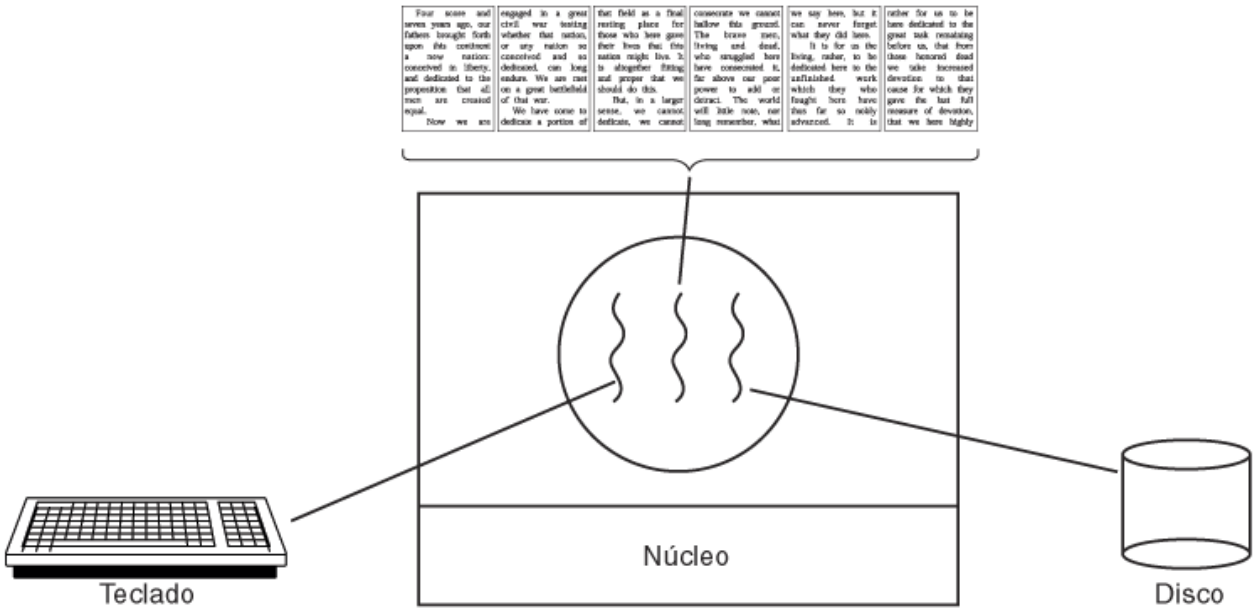
Cada thread tem sua própria pilha

- A pilha contém uma estrutura para o procedimento chamado
- A estrutura possui váriáveis locais do procedimento e o endereço de retorno



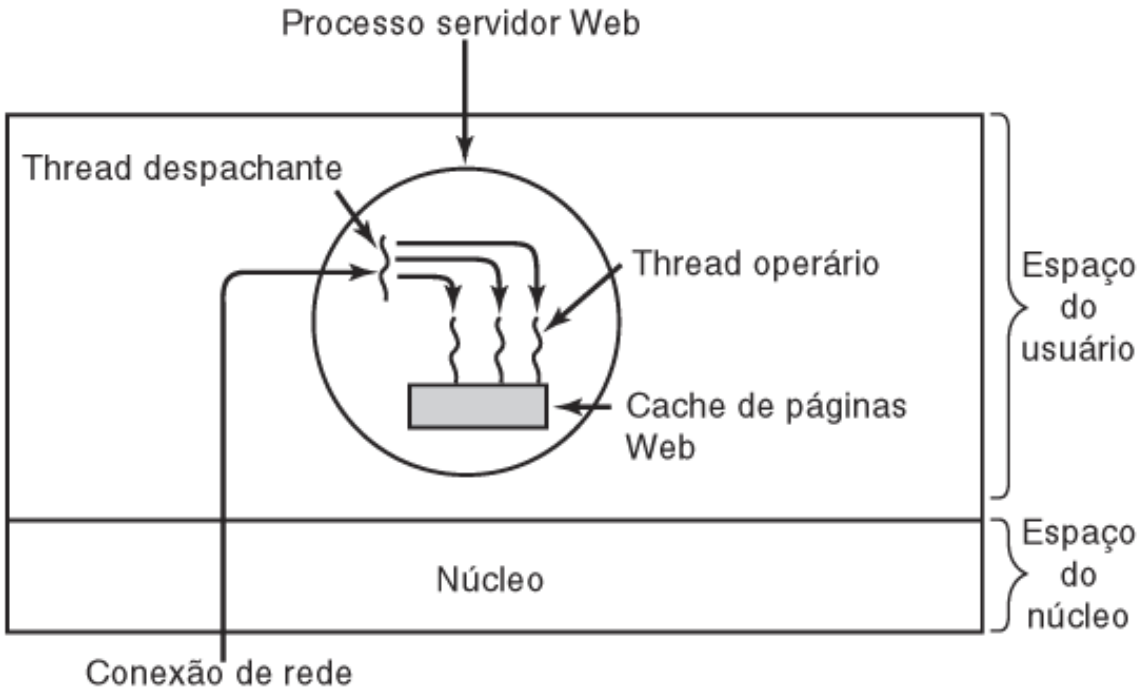
Uso de Thread (1)

Um processador de texto com três threads



Uso de Thread (2)

Um servidor web com múltiplas threads



Uso de Thread (3)

Código simplificado do slide anterior

- (a) Thread despachante
- (b) Thread operário

```
while (TRUE) {
    get_next_request(&buf);
    handoff_work(&buf);
}
```

(a)

```
while (TRUE) {
    wait_for_work(&buf)
    look_for_page_in_cache(&buf, &page);
    if(page_not_in_cache(&page))
        read_page_from_disk(&buf, &page);
    return_page(&page);
}
```

(b)

Sumário

Processos

- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Implementação de Threads

Threads de Usuário

Threads de Núcleo

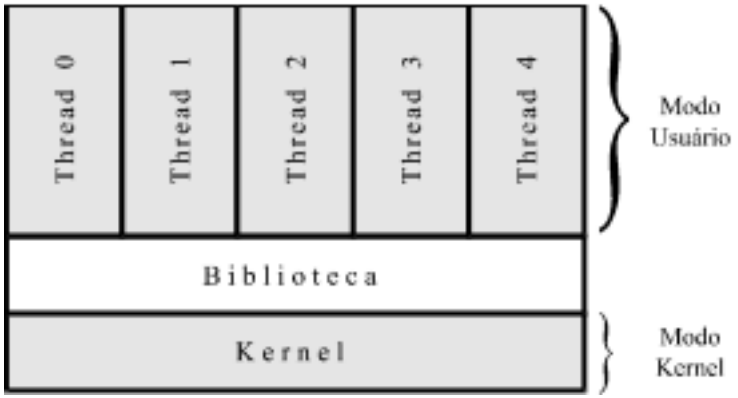
Threads Híbridas

Implementação de Threads de Usuário (1)

Chamadas a uma biblioteca de rotinas ligadas e carregadas em tempo de execução no espaço de endereçamento do processo

Biblioteca responsável por gerenciar e sincronizar as threads de usuário

Também chamado N:1 (user:processor)



Implementação de Threads de Usuário (2)

Threads são totalmente inseridas dentro do espaço de usuário

- Contém rotinas de criar/terminar threads (ex. pthread_create, pthread_exit, pthread_yield, pthread_join etc.)
- Contém operações sobre variáveis de sincronização (ex. exclusão mútua)

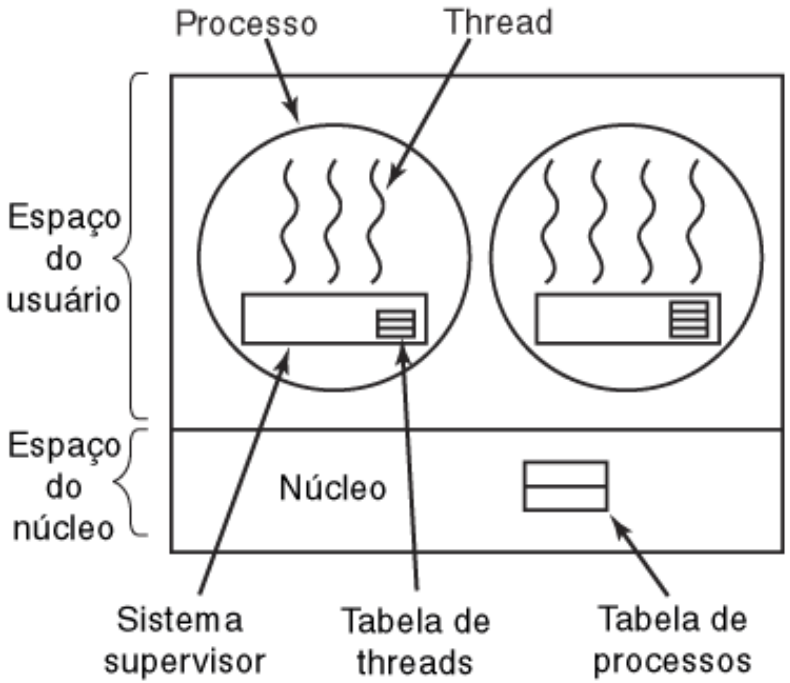
- Vantagens
  - Podem ser implementadas em SOs que não suportem threads
  - São portáveis por não precisarem de APIs de gerenciamentos de SOs

Implementação de Threads de Usuário (3)

- Executam no topo do Sistema Servidor (SS)
  - Cada processo possui sua tabela de threads
- Controle de execução de threads pelo SS
  - Ex. SS verifica se thread deve passar ao estado “bloqueado”
    - Se sim, SS armazena registradores da thread na tabela de thread, busca outra thread para executar e recarrega registradores da máquina com novos valores
  - OBS: Alternância mais rápida por não precisar desviar para controle de núcleo

Implementação de Threads de Usuário (4)

Um pacote de threads de usuário



Implementação de Threads de Usuário (5)

- Vantagens
  - Procedimento que salva o estado da thread e o escalonador são procedimentos locais
  - Desnecessário o chaveamento de contexto
  - Escalonamento eficiente
  - Cada processo apresenta próprio algoritmo de escalonamento de threads
  - Gerenciadas por bibliotecas em nível de usuário

Exemplos Threads Usuário

Exemplos:

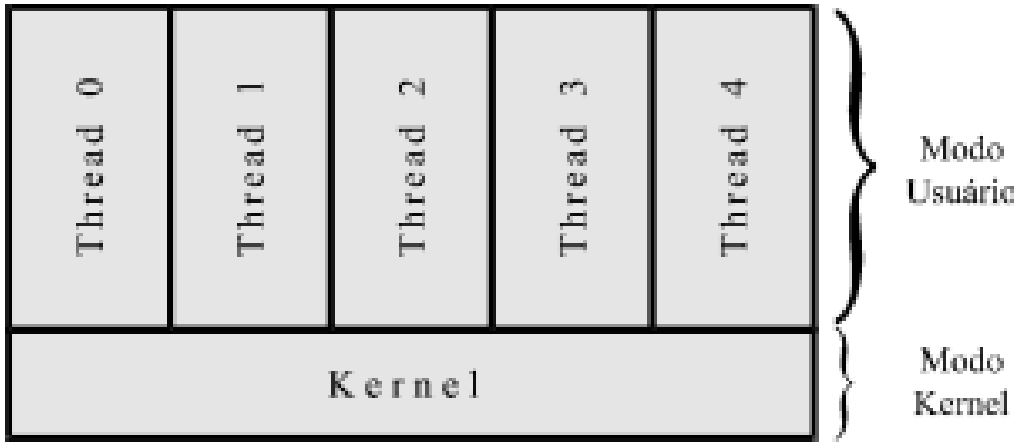
- POSIX Pthreads
- Threads Java
- Threads Win32

Implementação de Threads de Núcleo (1)

Implementadas diretamente pelo núcleo do sistema, por chamadas do sistema

SO (escalonador) sabe da existência de cada thread e pode escaloná-los individualmente.

Também chamado 1:1 (user: processor)



Implementação de Threads de Núcleo (2)

Inexistência do Sistema Supervisor

Inexistência da tabela de threads por processo

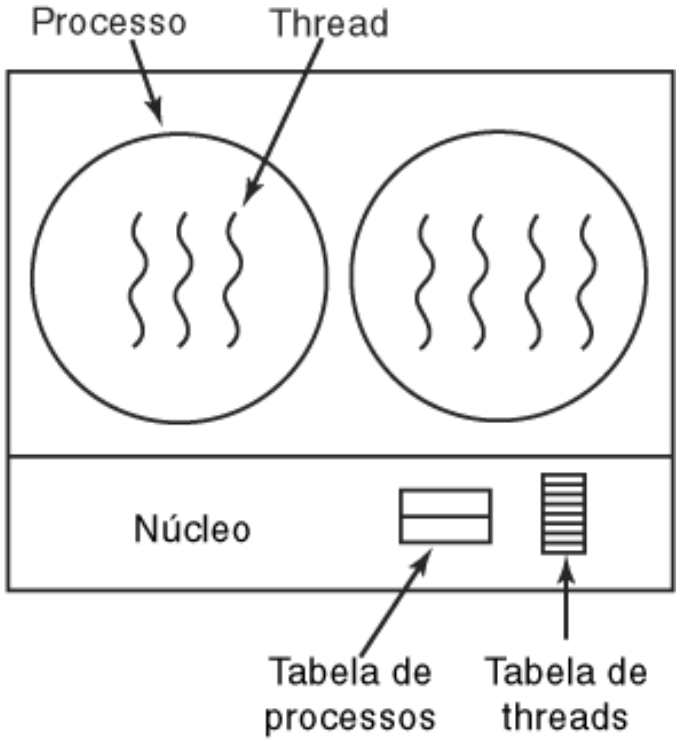
Tabela de thread única

Criação/destruição através de chamada de núcleo

- Chaveamento de contexto de threads de núcleo semelhante ao chaveamento de contexto de processos

Implementação de Threads de Núcleo (3)

Um pacote de threads gerenciado pelo núcleo



Implementação de Threads de Núcleo (4)

Vantagens

- Podem conter rotinas com instruções privilegiadas
- Em sistemas multiprocessamento, o núcleo pode despachar threads de um mesmo processo para diversas CPUs
- Uma thread com operação de E/S bloqueante não paralisa todos as threads do mesmo processo
- Ideal para ambientes com rotinas de escalonamento baseadas em prioridades (SOs tempo real)

Implementação de Threads de Núcleo (5)

Desvantagens

- O suporte a threads está no SO e depende de rotinas específicas tornando o código menos portável
- Invocam o núcleo para tomada de decisões e, com isso, há uma mudança de modo e com sobrecarga no SO
- O SO pode ser obrigado a gerenciar muitos threads de aplicativos, consumindo recursos como memória e CPU

Exemplos Threads Núcleo

Exemplos:

- Windows XP/2000
- HP-UX
- Linux
- Tru64 UNIX
- MacOS

Implementação Híbrida (1)

Combinação de threads de usuário com threads de núcleo

Uso de threads de núcleo e multiplexação de threads de usuário sobre uma ou todas threads de núcleo

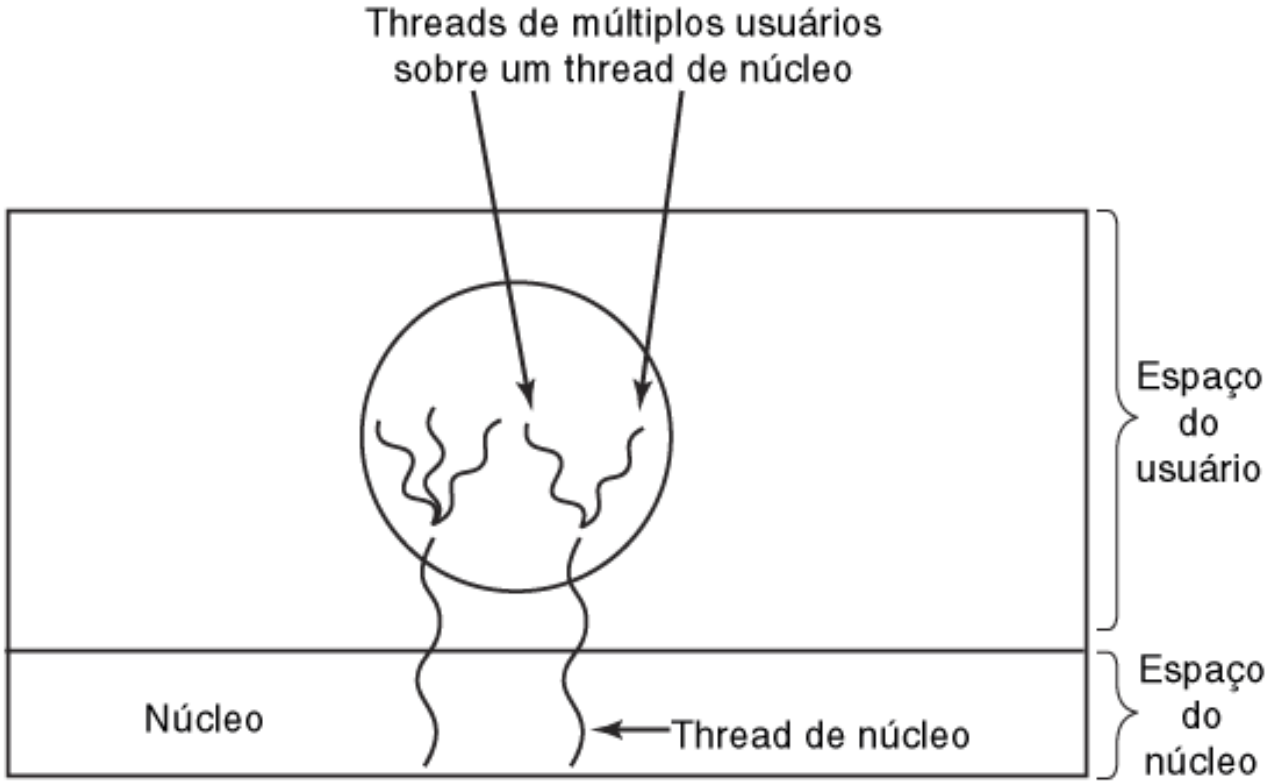
O núcleo tem conhecimento apenas das threads de núcleo e as escalona

Código em nível de usuário fornece sugestões de agendamentos de execução para escalonador de threads de núcleo

Também chamado N:M (users:processor)

Implementação Híbrida (2)

Multiplexação de threads de usuário sobre threads de núcleo



Exemplos Threads Híbrida

Exemplos:

- Windows 7
- Solaris

Sumário

Processos

- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Threads em Sistemas Distribuídos

Clientes Multithreads

Servidores Multithreads

Clientes Multithreads (1)

Clientes necessitam ocultar o delay de transporte em redes de longa distância

Técnicas de ocultação de latência

- Após início da conexão, cliente executa outras tarefas (uso de threads)

Busca de documentos web são operações bloqueadoras

Browser executa várias tarefas simultâneas

- Uso de conexões http paralelas

Servidores Multithreads (1)

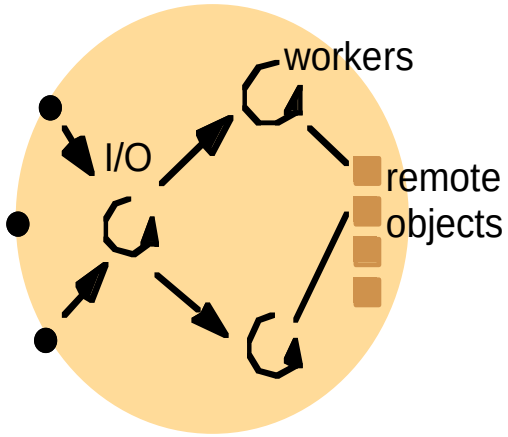
Sistemas monoprocessadores

- Problema: num processo monothread, o processo é bloqueado quando uma chamada bloqueadora do sistema for executada

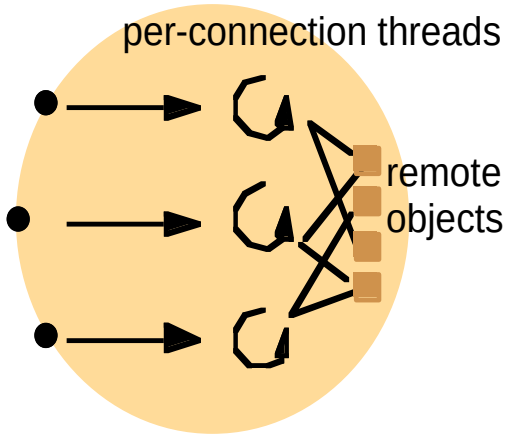
Sistemas multiprocessadores

- Exploração do verdadeiro paralelismo
- Cada thread é designada para uma CPU diferente (thread de núcleo)

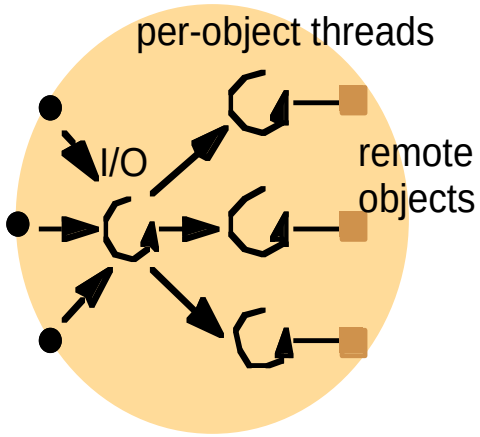
Arquitetura Baseada em Threads



a. Thread-per-request



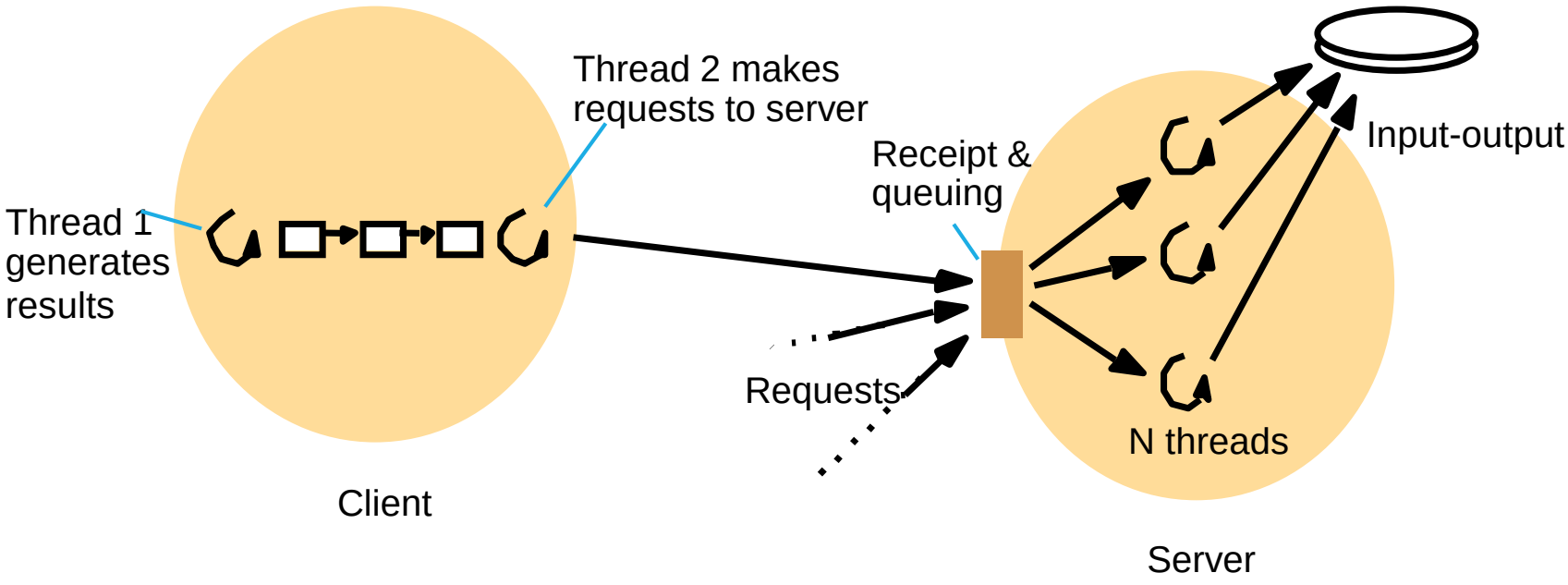
b. Thread-per-connection



c. Thread-per-object

Clientes e Servidores com threads

Servidor com *pool* de threads



Clientes e Servidores com threads

Suposição 1

- Cada thread retira um pedido da fila e o processa
- Processamento leva em média 2 milissegundos
- E/S leva 8 milissegundos
- Único processador sem cache
  - 2+8=10 milissegundos
  - Desempenho máximo (serv. manipula 100 pedidos/seg)

Clientes e Servidores com threads

Suposição 2

- Sem cache
- Servidor com 2 threads
- Escalonadas independentemente
- Uma unidade de disco ➔ ambas as threads ficam bloqueadas (E/S)
- Desempenho máximo 8 milissegundos (1000/8 = 125 pedidos/seg)

Clientes e Servidores com threads

Suposição 3

- Introdução de cache
- Servidor mantém dados que lê em buffers no seu espaço de endereçamento
- Servidor com 2 threads
- Escalonadas independentemente
- Uma thread examina a cache ao ler dados
- Taxa de acerto=75%
- Tempo médio por pedido (0,75*0 + 0,25*8=2 seg)
- Tx de rendimento = (1000/2=500 pedidos/seg)
- Pelo uso do cache, Tx de rendimento (1000/2,5= 400 pedidos/seg)

Sumário

Processos

- Definição
- Controle de Processos
- Processos em SD

Threads

- Definição
- Implementação de Threads
- Threads em SD

Comparando Processos x Threads

Processos vs Multithreads

Criar uma thread dentro de um processo é menos oneroso

O chaveamento para uma thread dentro de um mesmo processo é menos oneroso do que em processos diferentes

Threads em um processo compartilham dados e outros recursos

Ambiente de execução	*Thread*
Tabelas de espaço de endereçamento	Registradores internos do processador salvos
Interfaces de comunicação, arquivos abertos	Prioridade e estado da execução (como *BLOCKED*)
Semáforos, outros objetos de sincronização	Informações do tratamento da interrupção de software
Lista de identificadores de *thread*	Identificador do ambiente de execução
Páginas do espaço de endereçamento residentes na memória; entradas de cache em hardware	

Processos vs Multithreads

Tempo de criação/destruição de threads é inferior ao tempo de criação/destruição de processos (10 a 20 vezes)

Chaveamento de contexto entre threads é mais rápido do que entre processos (5 a 50 vezes)

Threads residem dentro dos processos que as portam, logo compartilham seus recursos

Platform	fork()			pthread_create()		
	real	user	sys	real	user	sys
AMD 2.4 GHz Opteron (8cpus/node)	41.07	60.08	9.01	0.66	0.19	0.43
IBM 1.9 GHz POWER5 p5-575 (8cpus/node)	64.24	30.78	27.68	1.75	0.69	1.10
IBM 1.5 GHz POWER4 (8cpus/node)	104.05	48.64	47.21	2.01	1.00	1.52
INTEL 2.4 GHz Xeon (2 cpus/node)	54.95	1.54	20.78	1.64	0.67	0.90
INTEL 1.4 GHz Itanium2 (4 cpus/node)	54.54	1.07	22.22	2.03	1.26	0.67

Fonte: http://www.llnl.gov/computing/tutorials/pthreads (tempo em segundos para criação 50000 processos /threads)

Processos vs Multithreads

Custo de criação de uma thread em um ambiente de execução existente

- Alocar uma pilha
- Fornecer valores iniciais para os registradores internos do processador
- Definir estado de execução (SUSPENDED OU RUNNABLE)
- Definir um valor para prioridade inicial

Custo de criação de um processo

- Criação de um ambiente de execução
- Criação de tabelas de espaço de endereçamento...

Processos vs Multithreads

Chaveamento de contexto de processos

- O sistema operacional realiza um chaveamento de contexto para interromper um processo em execução e começar a executar um processo previamente pronto.
- Salva o contexto de execução do processo em execução no PCB desse processo.

Chaveamento de contexto de threads que compartilham ambiente de execução

- Não envolve transição de domínio (é extremamente econômica)
- Não envolve uma chamada de sistema
- Escalonamento pode ser personalizado ou alterado de acordo com requisitos específicos da aplicação