

Agenda

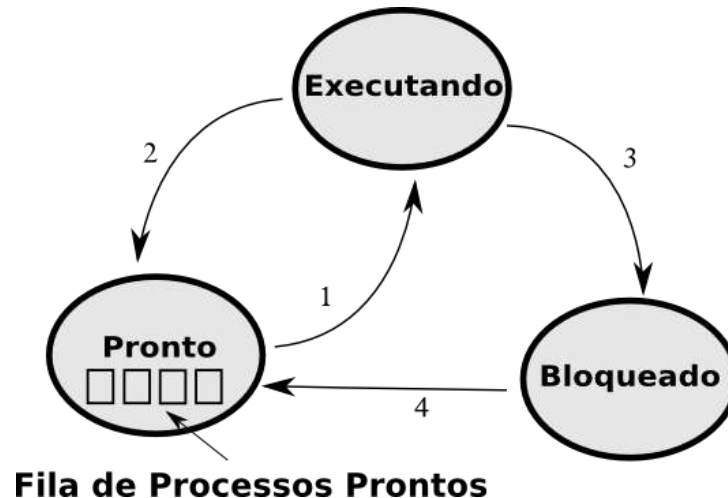
- Threads
- Virtualização
- Clientes
- Servidores
- Migração do Código

O que é um **Processo**?

O que é um **Thread**?

Processo

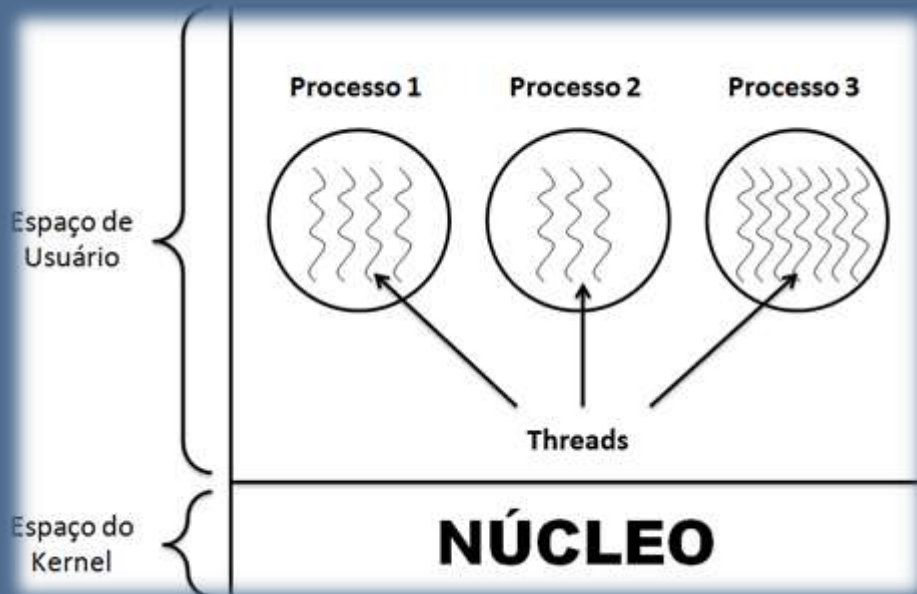
“Um processo costuma ser definido como um programa em execução, isto é, um programa que está sendo executado em um dos processadores virtuais do sistema operacional no momento em questão”



Thread

Conceito

“é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente”



Threads

Processos

Funcionamento Geral dos Processos

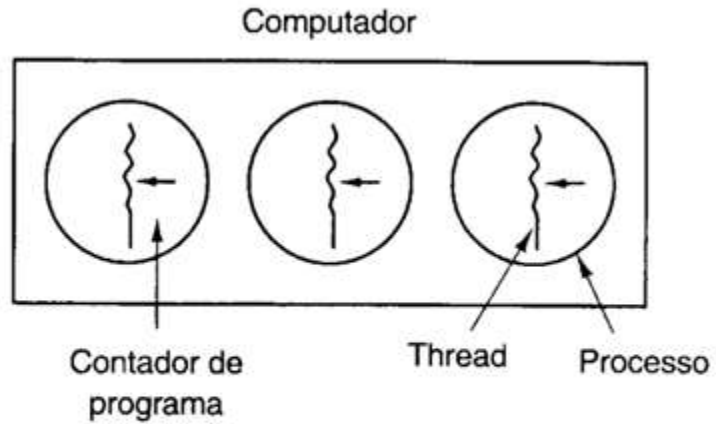
Threads Processos

- Os processos são criados a partir de uma ação qualquer;
- É reservado um espaço na memória para o determinado processo;
- Mas, os processos concorrem pelo hardware sob o qual estão rodando;
- A perda de desempenho está ligada ao chaveamento da CPU de um processo para o outro;
 - Podendo ser necessário retirar um outro processo da memória, exigindo interação com o disco rígido para guardar um processo A de modo que o processo B possa ser criado e alocado;

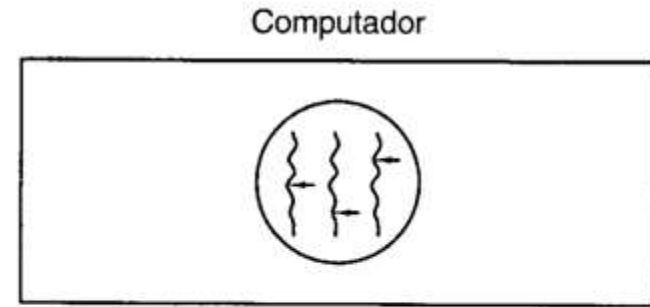


Figura 2.4: Diagrama de estados de uma tarefa em um sistema multi-tarefas.

MAZIERO, C. A. Sistemas Operacionais: Conceitos e Mecanismos. 2013.



(a)



(b)

Figura 2-6 (a) Três processos, cada um com um *thread*. (b) Um processo com três *threads*.

TANENBAUM, A. S.; WOODHULL, A. S. Sistemas Operacionais – Projeto e Implementação, 3ª ed. Bookman, 2008.

Threads

Processos

A importância da utilização de threads em sistemas distribuídos está na capacidade de obter vantagens de desempenho na execução de processos.

Implementação de Thread

Thread
Processos

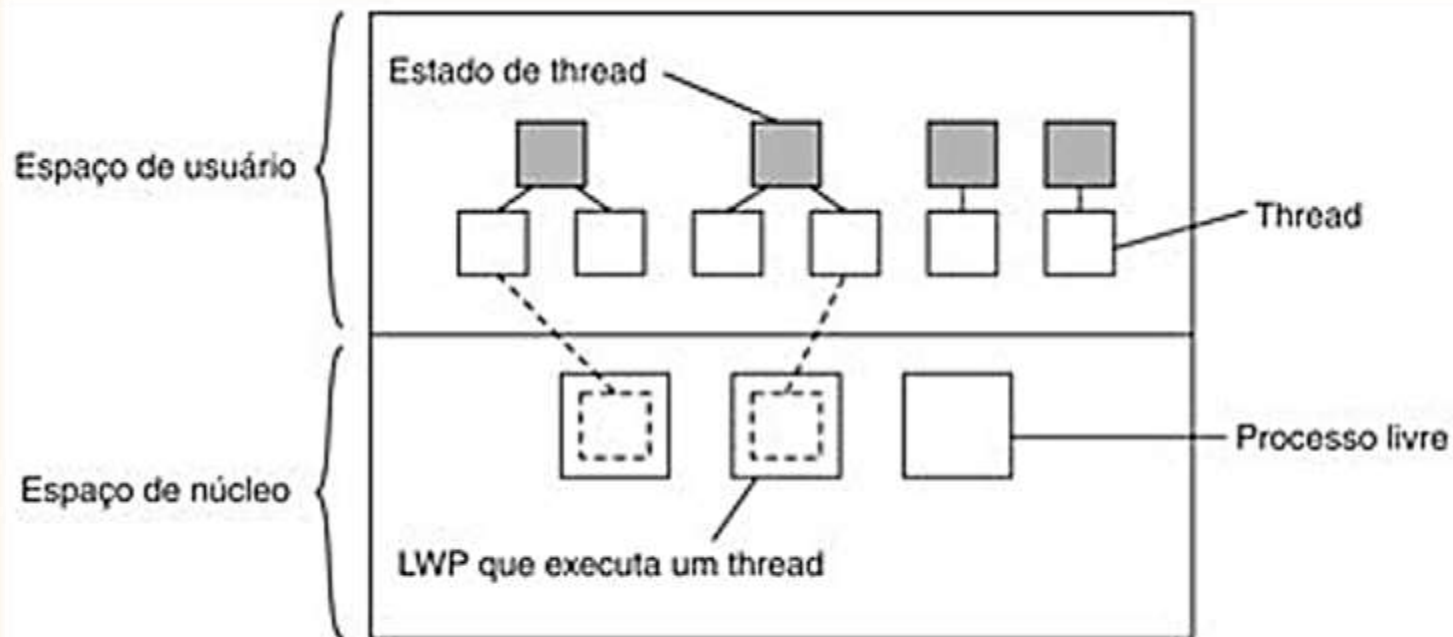


Figura 3.2 Combinação de processos leves de nível de núcleo e threads de nível de usuário.

Threads em Sistemas Distribuídos

Threads
Processos



Uma importante propriedade de threads é que eles podem proporcionar um meio conveniente de permitir chamadas bloqueadoras de sistemas sem bloquear o processo inteiro no qual o thread está executando.

Cliente Multithread

Threads em Sistemas Distribuídos

- A thread resolve o problema de precisar ficar esperando que uma ação seja concluída antes de executar outras tarefas;
- O browser executa várias tarefas de forma simultânea;
- Como funciona o browser ao acessar uma página?
 - O usuário não esperar carregar toda a página para visualizar. O que for sendo carregado vai ser mostrado, até que toda a página possa ser visualizada.

Cliente Multithread

Threads em Sistemas Distribuídos

- Outro ponto de destaque:
 - é a capacidade dos browsers poderem abrir várias conexões com o servidor;
- Como funciona?
 - Servidores replicados em uma mesmo site, por meio de alternância cíclica ou balanceamento de carga um cliente *multithread* pode ter cada execução ou chamada em um servidor diferente;



A prática mostra que o multithreading não somente simplifica consideravelmente o código do servidor, mas também facilita muito o desenvolvimento de servidores que exploram **paralelismo** para obter alto desempenho [...]

**O que é paralelismo em
computação?**

Servidores Multithread

Threads em Sistemas Distribuídos

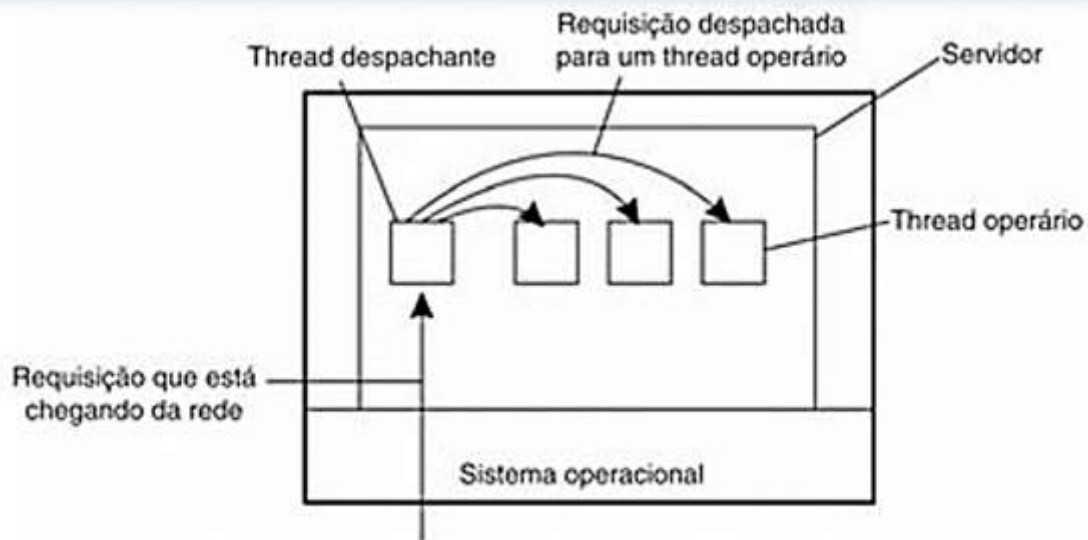


Figura 3.3 Servidor multithread organizado segundo modelo despachante/operário.

Servidores Multithread

Threads em Sistemas Distribuídos

Quais as vantagens de cada estratégia para servidores Multithread?

Três modos de construir um servidor

Modelo	Características
Threads	Paralelismo, chamadas bloqueadoras de sistema
Processos Monothread	Sem paralelismo, chamadas bloqueadoras de sistema
Máquina de Estado Finito	Paralelismo, chamadas de sistemas não bloqueadoras

Virtualização

Processos

“

Em sua essência, a virtualização trata de estender ou substituir uma interface existente de modo a imitar o comportamento de um outro sistema [...]

Virtualização em Sistemas Distribuídos

Virtualização

- A ideia central nasce na década de 1970 com o objetivo de reutilizar hardware;
- Um *case* de sucesso foi aplicado nos IBM's 370 e sucessores, no qual ofereciam uma máquina virtual;

Virtualização em Sistemas Distribuídos

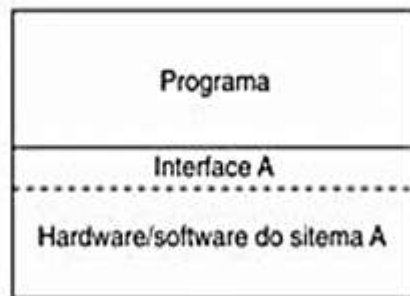
Virtualização

“A medida que o hardware ficava mais barato, os computadores ficavam mais potentes e a quantidade de tipos diferentes de sistemas operacionais diminuía, a virtualização deixava de ser um problema tão importante. Todavia, as coisas mudaram desde o final da década de 1990 por várias razões [...]”

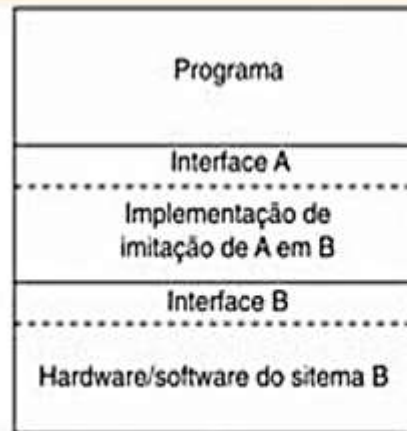
Virtualização em Sistemas Distribuídos

Virtualização

- Uma aplicação prática:
 - Exemplo do Detran;
 - Software da década de 1950;
- Virtualização é a solução para auxiliar a manutenção de *middelwares*;
- Auxiliar na permanência de softwares legados;
 - Exemplo da Secretaria de educação;



(a)



(b)

Figura 3.4 (a) Organização geral entre programa, interface e sistema. (b) Organização geral da virtualização do sistema A sobre o sistema B.

Arquiteturas de Máquinas Virtuais

Virtualização

“[...] sistemas de computadores oferecem quatro tipos diferentes de interfaces em quatro níveis diferentes”

1. Uma interface entre o hardware e o software, o qual consiste em **instruções de máquina** que possa ser invocadas por qualquer programa;
2. Uma interface entre o hardware e o software, o qual consistem em instruções de máquina que possam ser invocadas somente por programas privilegiados, como um sistema operacional;
3. Uma interface que consistem em **chamadas de sistemas** como oferecidas por um sistema operacional;
4. Uma interface que consistem em chamadas de biblioteca que, em geral, formam o que é conhecido como **interface de aplicação de programação (application programming interface – API)**.

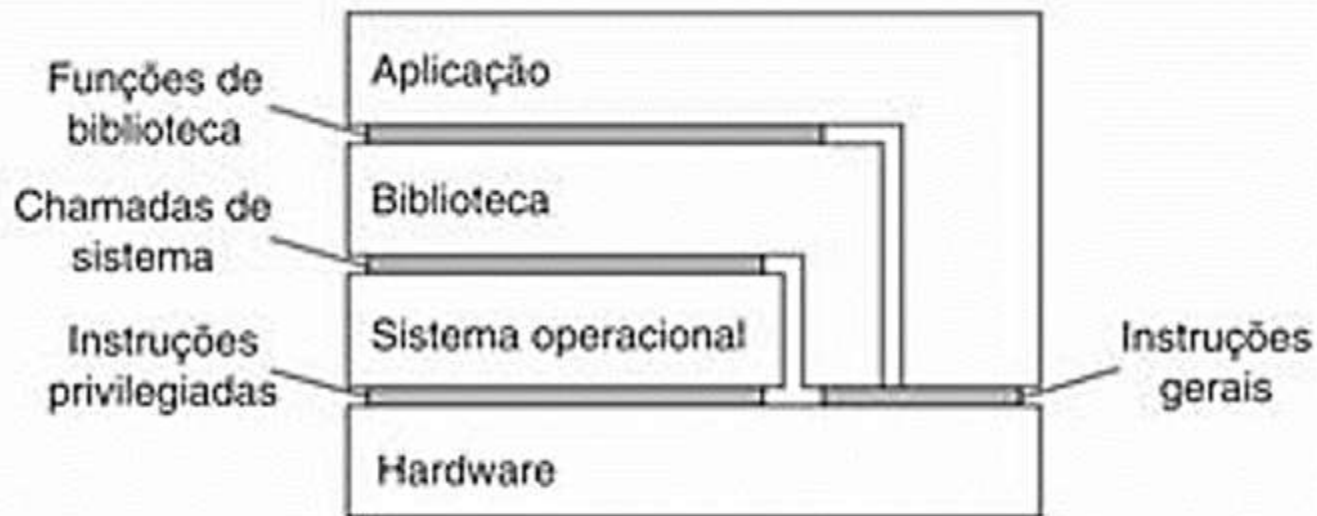


Figura 3.5 Várias interfaces oferecidas por sistemas de computadores.

Modos de Virtualização

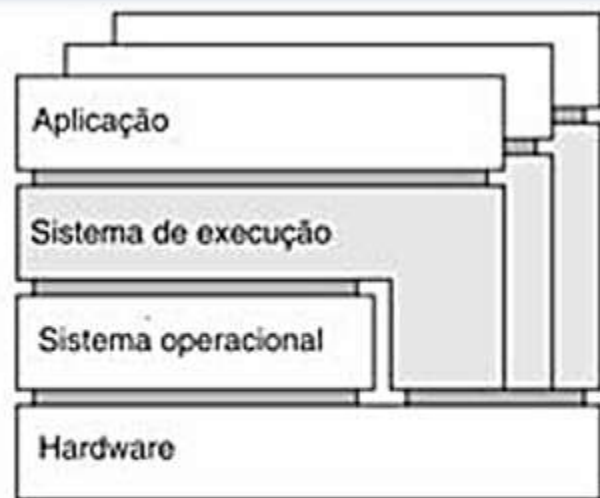
Arquiteturas de Máquinas Virtuais Virtualização

- Primeira forma relacionada com a interpretação ou emulação de um conjunto de instruções;
 - Exemplos:
 - JVM
 - Dalvik
 - Wine (Play On Linux)

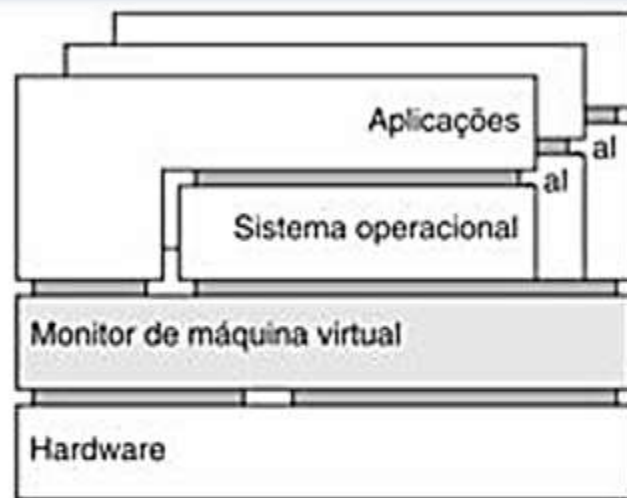
Modos de Virtualização

Arquiteturas de Máquinas Virtuais Virtualização

- A segunda abordagem fornece um sistema que é implementado sobre uma camada que protege o hardware original, mas oferece interface para o usuário;
 - Exemplo:
 - Virtualbox (sistema)
 - Docker
 - Exemplo de Interfaces de usuário:
 - Genymotion
 - Vagrant

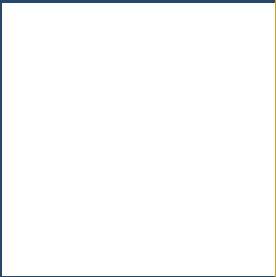


(a)



(b)

Figura 3.6 (a) Máquina virtual de processo, com várias instâncias de combinações (aplicação, execução).
(b) Monitor de máquina virtual com várias instâncias de combinações (aplicações, sistema operacional).



Apresentar sistemas de virtualização

Clientes

Processos

“

Uma tarefa importante de máquinas clientes é proporcionar aos usuários meios de interagir com servidores remotos.

Interfaces de Usuário em Rede

Clientes

Duas formas de comunicação em rede por interfaces de usuários:

1. Primeira forma: as aplicações se comunicam e sincronizam informações por meio de um protocolo implementando em nível de usuário;
2. Segunda forma: o terminal burro é utilizado apenas para enviar comandos enquanto o servidor é quem de fato executa as informações;

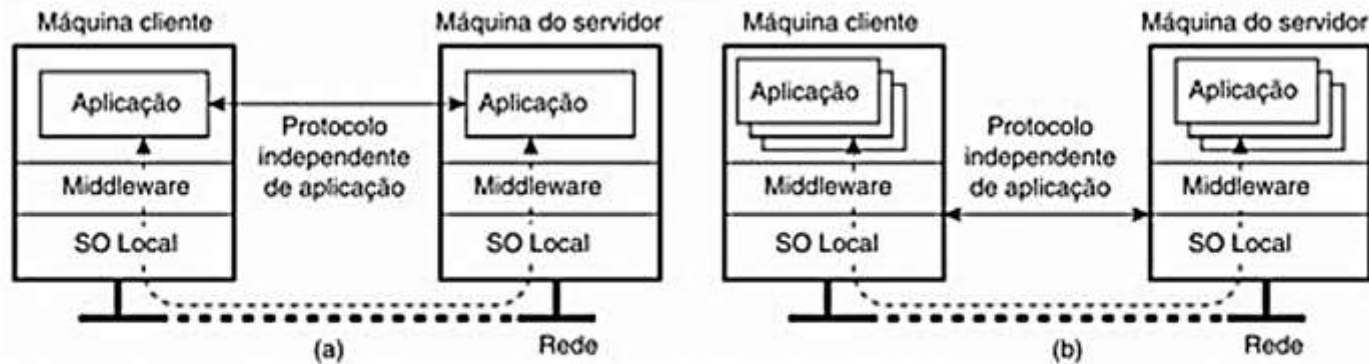


Figura 3.7 (a) Aplicação em rede com seu próprio protocolo. (b) Solução geral para permitir acesso a aplicações remotas.

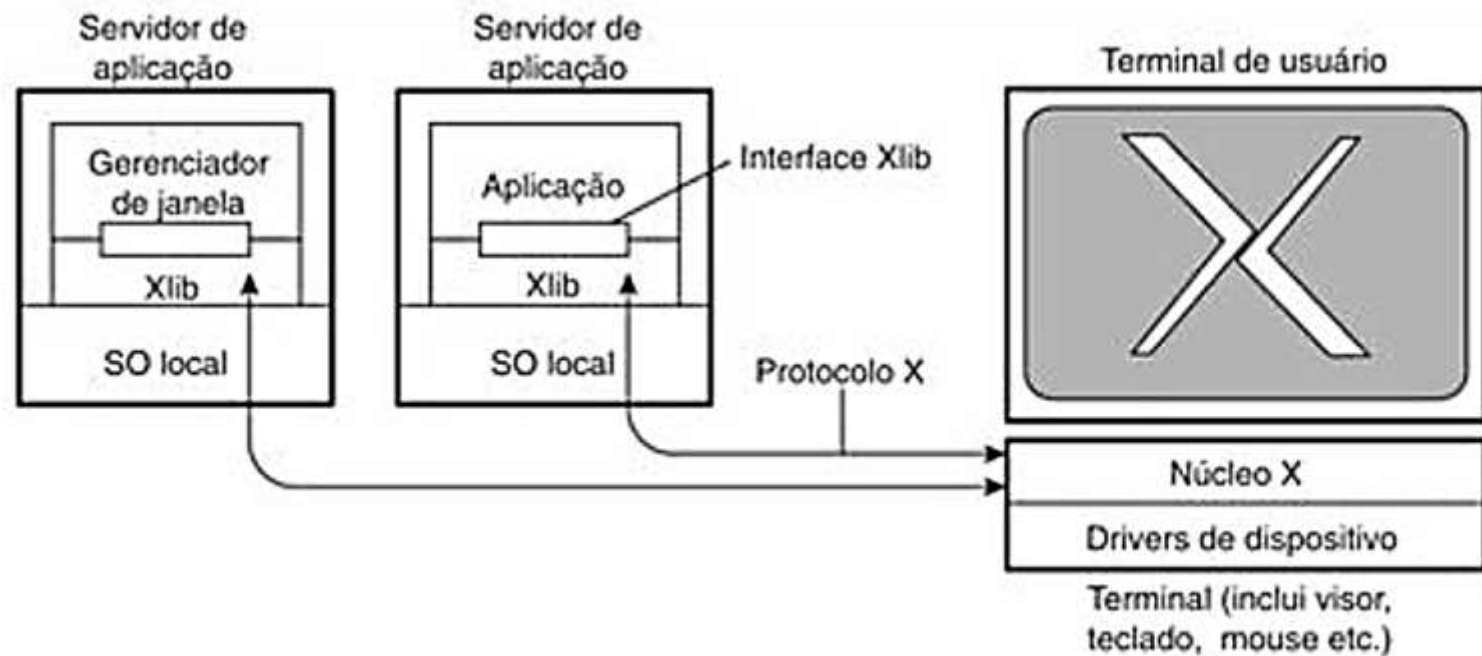


Figura 3.8 Organização básica do sistema X Window.

“

Em muitos casos, parte do nível de processamento e dados em uma aplicação cliente-servidor são executadas também do lado do cliente.

Transparência de Distribuição

**Software do lado
do cliente**

- Exemplos:
 - Caixas automáticos;
 - Caixas registradoras;
 - Algumas aplicações web com formulário;

- Neste modelo o ideal é que o cliente não saiba com quem está se conectado, mas que forneça as informações necessárias para a interface de usuário.

Transparência de Acesso

Software do lado do cliente

- É ocultado que está havendo comunicação com qual arquitetura de um computador servidor;
- A transparência de acesso busca criar uma interface que possa abstrair o hardware;
- O middleware pode ocultar a mudança de requisição de servidores, tornando transparente a conexão, e assim o usuário não percebe, no caso de uma desconexão, em qual servidor o cliente está sendo alocado;

Transparência de Replicação

Software do lado do cliente

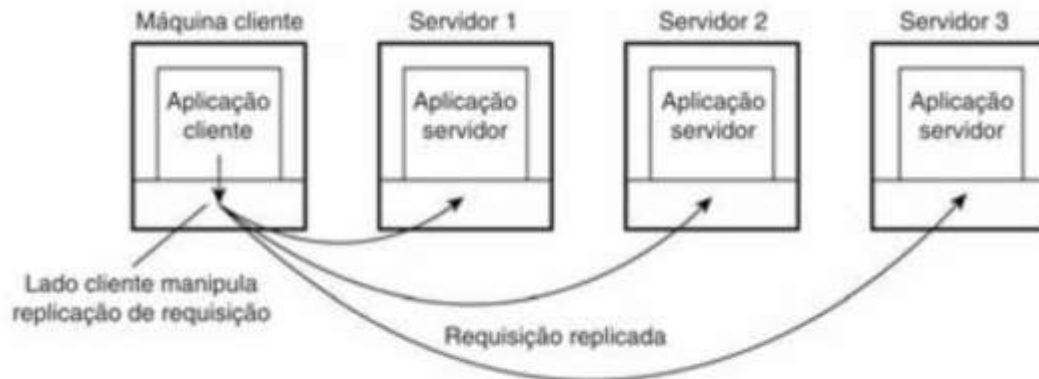


Figura 3.9 Replicação transparente de um servidor usando uma solução do lado do cliente.

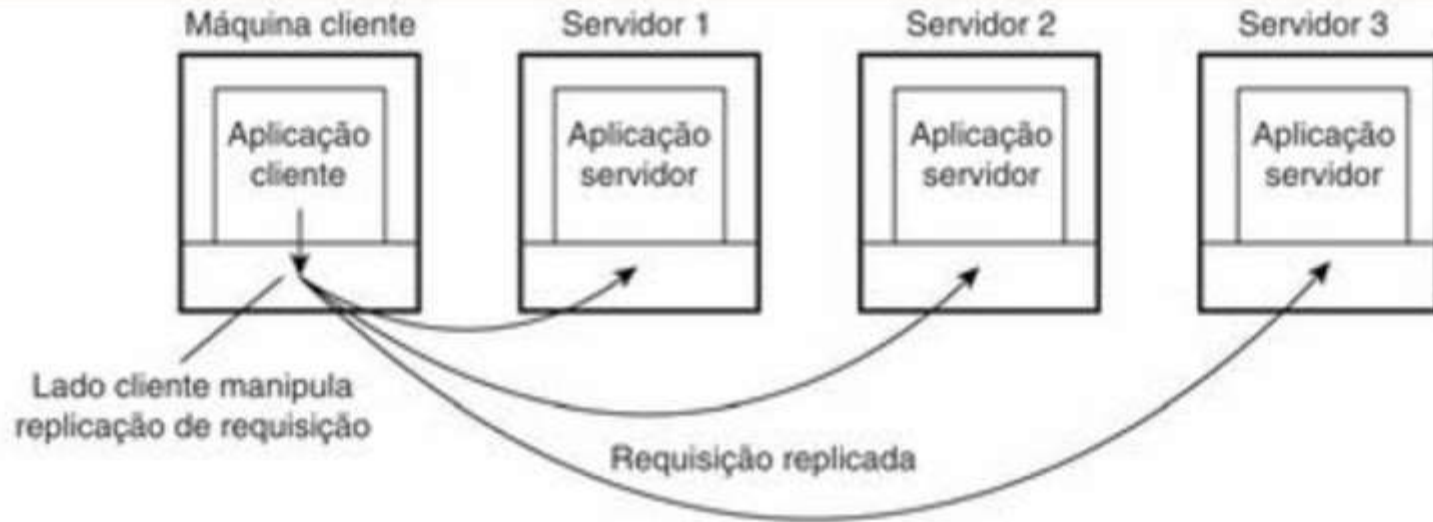


Figura 3.9 Replicação transparente de um servidor usando uma solução do lado cliente.

Transparência a falhas e a concorrência

Software do lado do cliente

- Técnicas de transparência a falhas utilizam, por exemplo, softwares que possam trocar de servidor após verificar repetidas conexões ou utilizar cache;
- Transparência de concorrência pode ser resolvido com servidores intermediários que monitoram as transações;

Servidores

Processos

“

**Um servidor é um processo
que implementa um serviço
específico em nome de um
conjunto de clientes.**

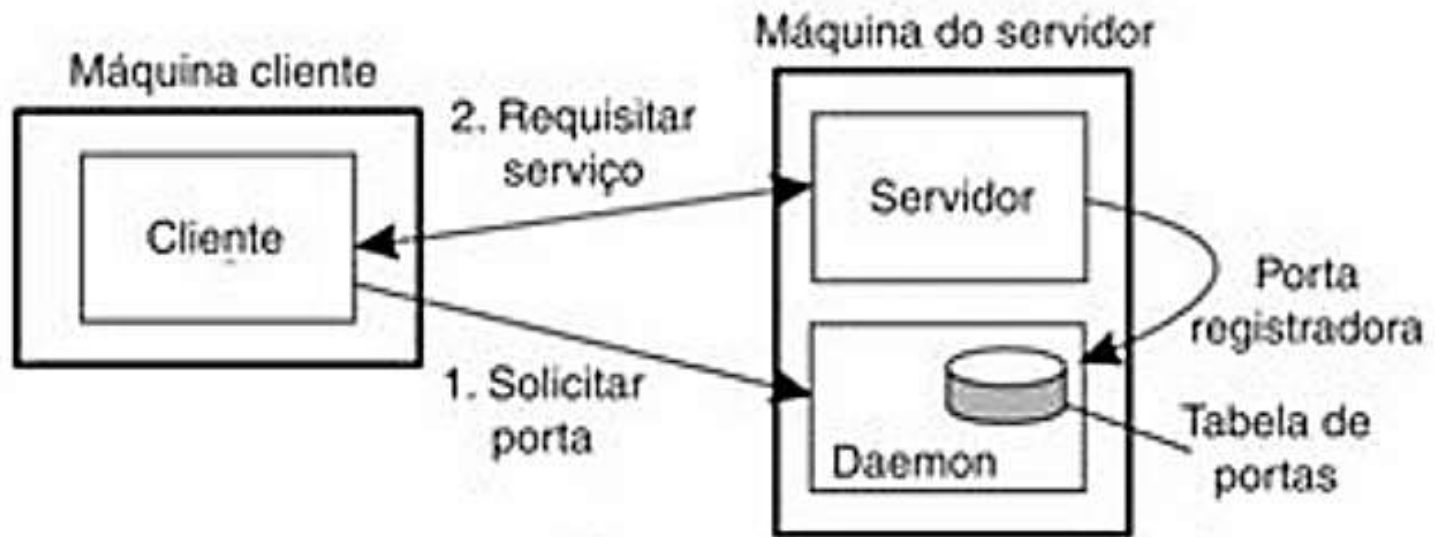
Tipos de Servidores

Servidores

- **Servidor iterativo**
 - É o próprio servidor que manipula a requisição e, se necessário, retorna uma resposta ao cliente requisitante;
- **Servidor concorrente**
 - Não manipula por si uma requisição, delega a uma thread separado ou a outro processo, enquanto aguarda por outra requisição;

Como clientes contatam
servidores?

Como clientes sabem qual a
porta de um serviço?



(a)

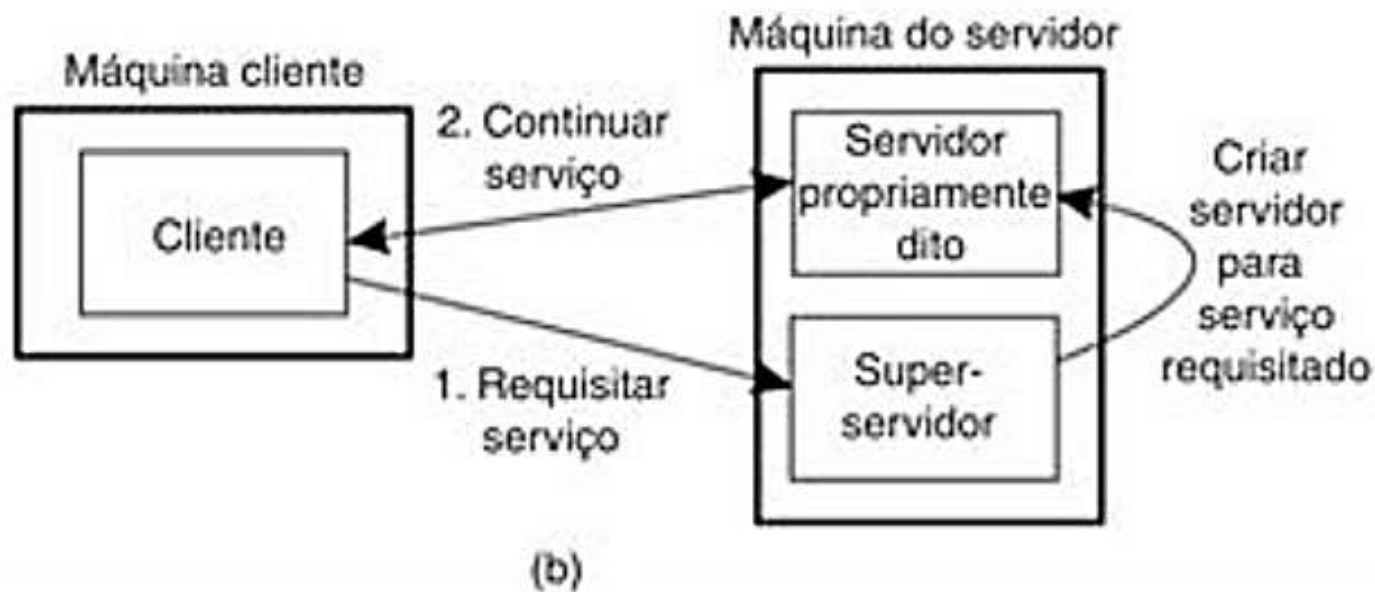


Figura 3.10 (a) Vinculação cliente-a-servidor usando um daemon. (b) Vinculação cliente-a-servidor usando um superservidor.

Um servidor pode ter sua
comunicação com o cliente
interrompida?

Qual estratégia você usaria?

Estado do Servidor

Servidores

- Servidores sem estado
 - Não mantém informações sobre o estado de seus clientes e pode mudar seu próprio estado sem ter de informar a nenhum cliente;
 - Exemplo: servidor web.
- Servidores com estado flexível
 - O servidor “promete” manter o estado em nome do cliente, mas apenas por tempo limitado;
- Servidor com estado
 - Mantém informações persistentes sobre seus clientes.
 - Problema: Se o servidor falha é preciso recuperar a tabela de entradas (cliente, arquivo);

Estado de Sessão e Estado Permanente

Servidores

- Estado de sessão
 - É mantido em arquiteturas de três camadas, no qual o servidor de aplicação precisa acessar um servidor de banco de dados por meio de um conjunto de consultas antes de responder ao cliente;
- Estado permanente
 - Informações mantidas em bancos de dados como informações de clientes, chaves associadas com software comprado, entre outras;

Cluster de Servidores

Servidores

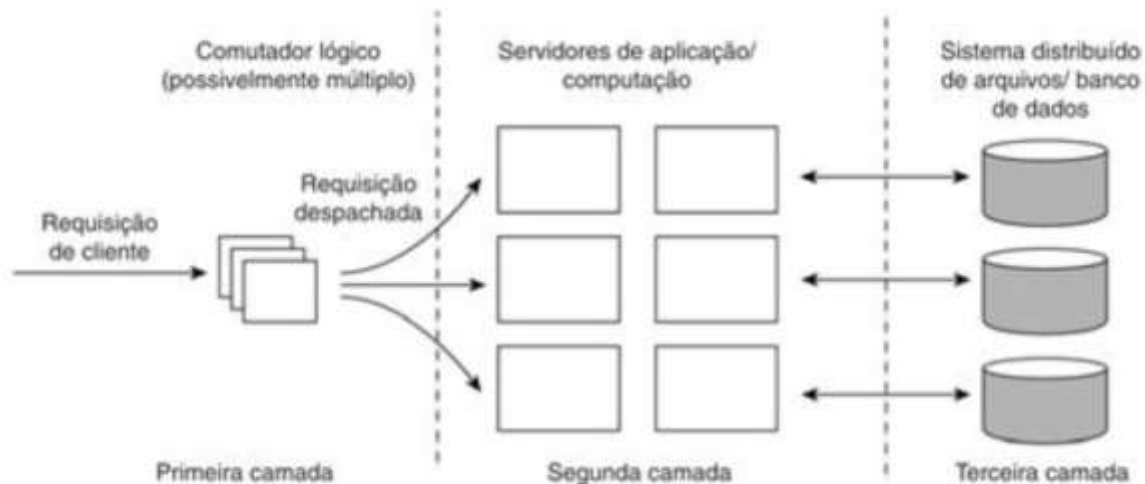


Figura 3.11 Organização geral de um cluster de servidores de três camadas.

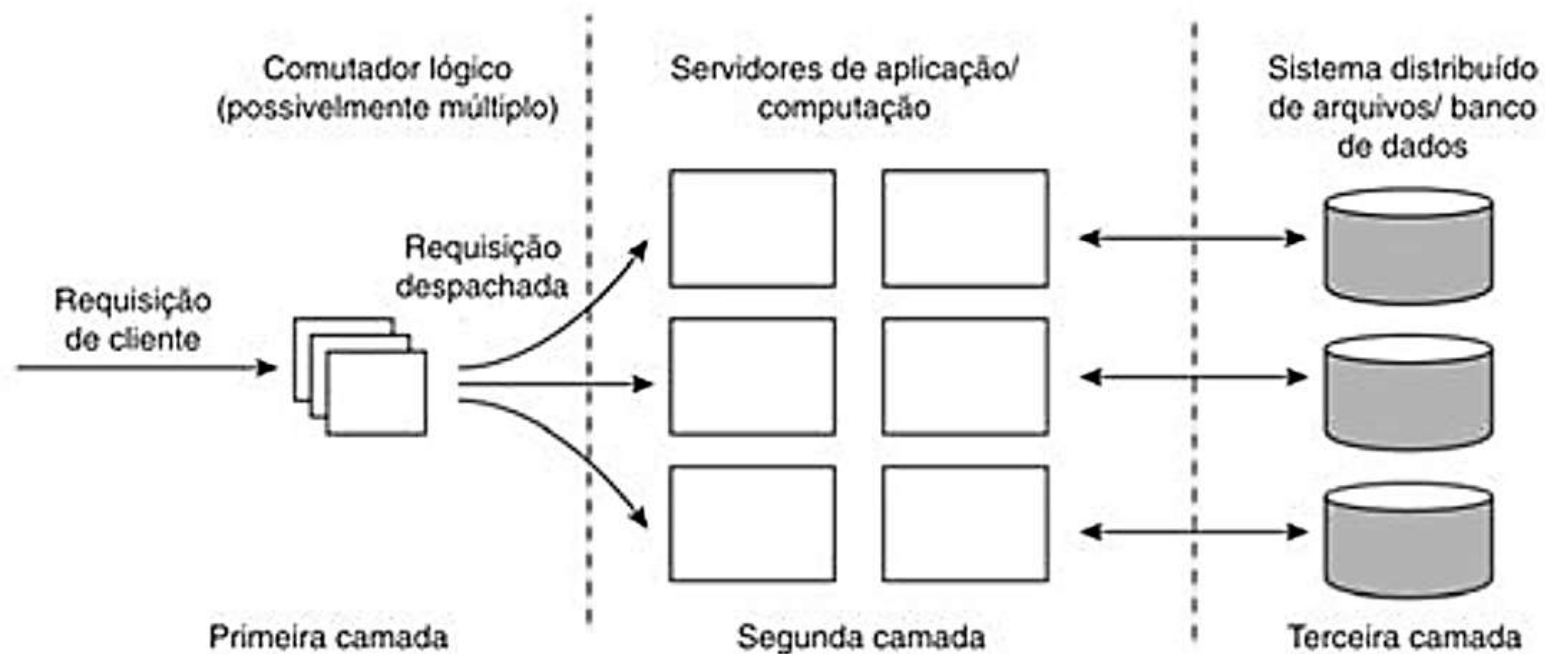


Figura 3.11 Organização geral de um cluster de servidores de três camadas.

Cluster de Servidores com Comutador

Servidores

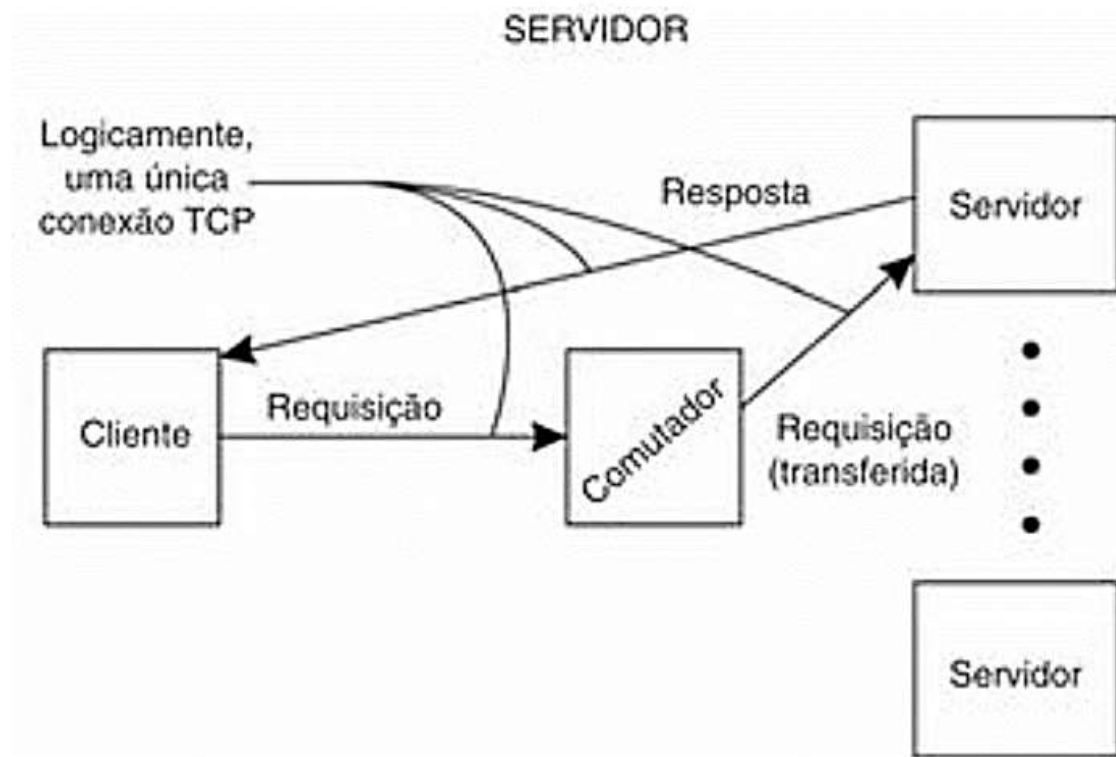


Figura 3.12 Princípio da transferência TCP



Nesses cluster, muitas vezes há uma máquina de administração separada que monitora servidores disponíveis e passa essa informação para outras máquinas conforme adequado, tal como o comutador.

Servidor Distribuído

Conceito

“É um conjunto de máquinas que possivelmente muda dinamicamente, com vários pontos de acesso também possivelmente variáveis, mas que, quanto ao mais, se apresenta ao mundo externo como uma única e poderosa máquina.”

Servidor Distribuído

Otimização de rota (IPV6)

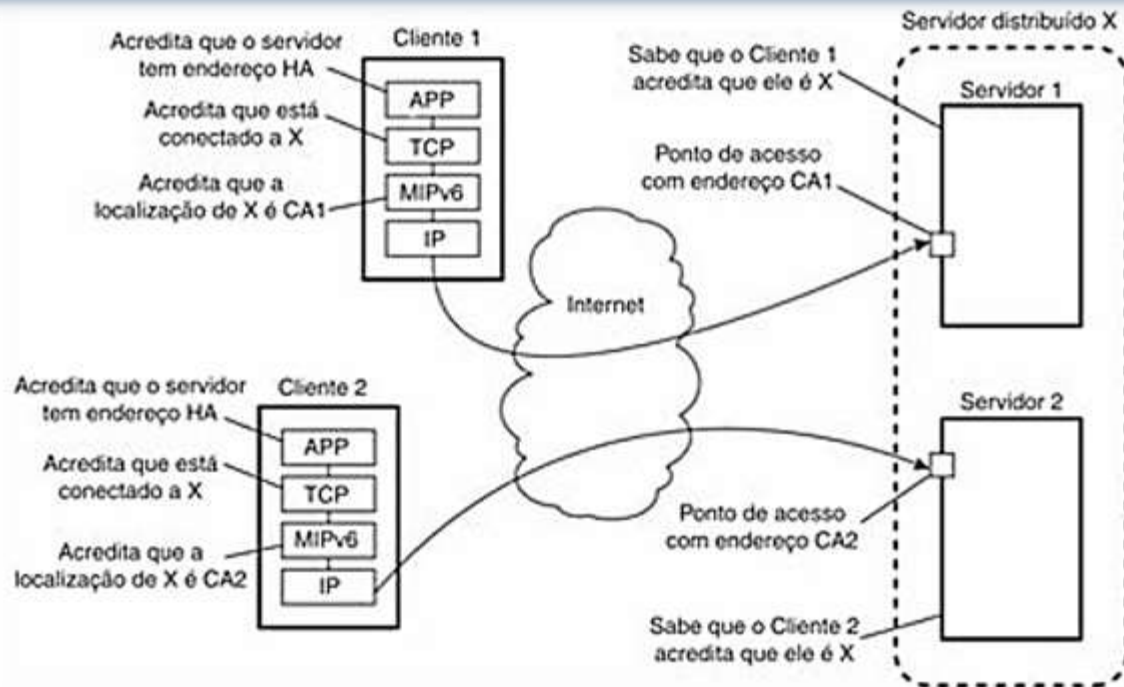


Figura 3.13 Otimização de rota em um servidor distribuído.

Migração de Código

Processos

Motivação

- Aumentar o desempenho;
- Então a migração de código, ou migração de processo, busca diminuir o impacto de uma máquina para outra menos requisitada.

“A carga costuma ser expressa em termos do comprimento da fila da CPU ou da utilização da CPU, mas outros indicadores de desempenho também são usados”

Motivação

Exemplos

- Exemplo 1
 - Uma aplicação no cliente que realiza muito acesso ao banco de dados.
 - Transferir esse código para o servidor é o ideal já que o impacto na rede é muito alto.
 - O sentido está em processar o código onde o mais próximo dos dados.

Motivação

Exemplos

- Exemplo 2
 - Tratamento de formulários no servidor;
 - O ideal é que o tratamento inicial seja realizado no cliente, buscando evitar processamento desnecessário no servidor;

Modelo de Descarga de Código

- O código a ser utilizado só será carregado na hora que o cliente precisar se comunicar com o servidor;
- Envolve um problema de segurança.
 - Acessar as informações e descarregar um código no seu disco sem você saber o que ele faz, que tipo de informação ele envia;

Modelo de Descarga de Código

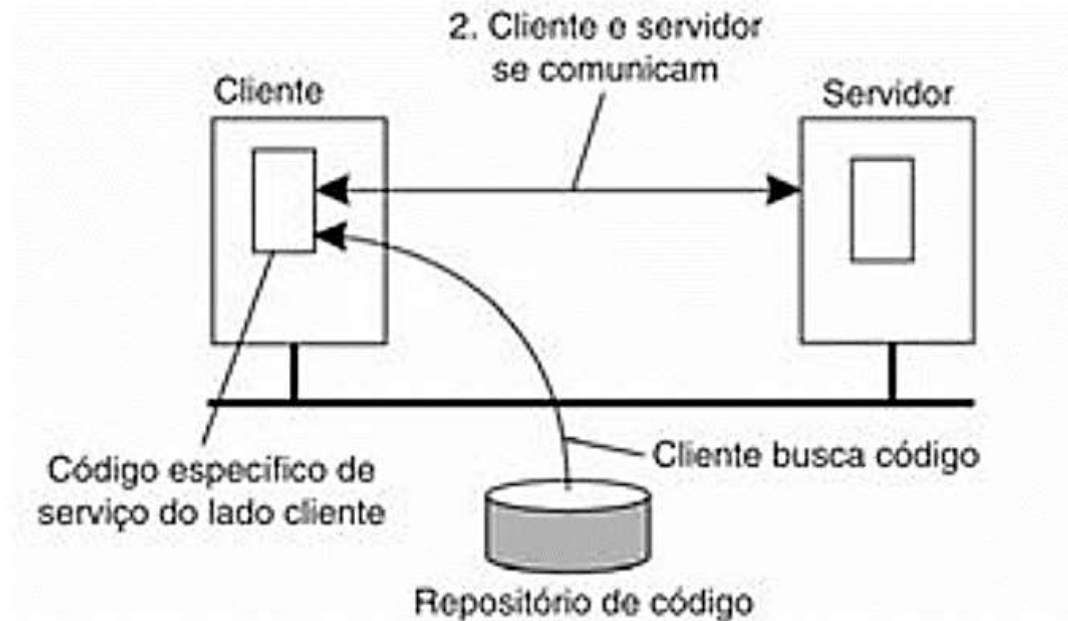


Figura 3.16 Princípio de configuração dinâmica de um cliente para se comunicar com um servidor. Primeiro o cliente busca o software necessário e então invoca o servidor.

“

**[...] migração de código trata
da movimentação de
programas entre máquinas
com a intenção de executá-
los na máquina-alvo.**

Segmentos de um Processo

Migração de Código

- Segmento de código
 - É a parte que contém as instruções que compõem o programa em execução;
- Segmento de recursos
 - Contém referências a recursos externo que o processo necessita (impressoras, arquivos...)
- Segmento de execução
 - Utilizado para armazenar o estado de execução de um processo no momento determinado;

Mobilidade

- Mobilidade fraca
 - É possível transferir somente o segmento de código;
- Mobilidade forte
 - Nesta qualificação, a mobilidade permite não apenas enviar o segmento de código, mas também o segmento em execução;
 - O processo deverá ser parado, transferido, e em seguida iniciado do exato ponto onde parou;

Alternativas para migração de código

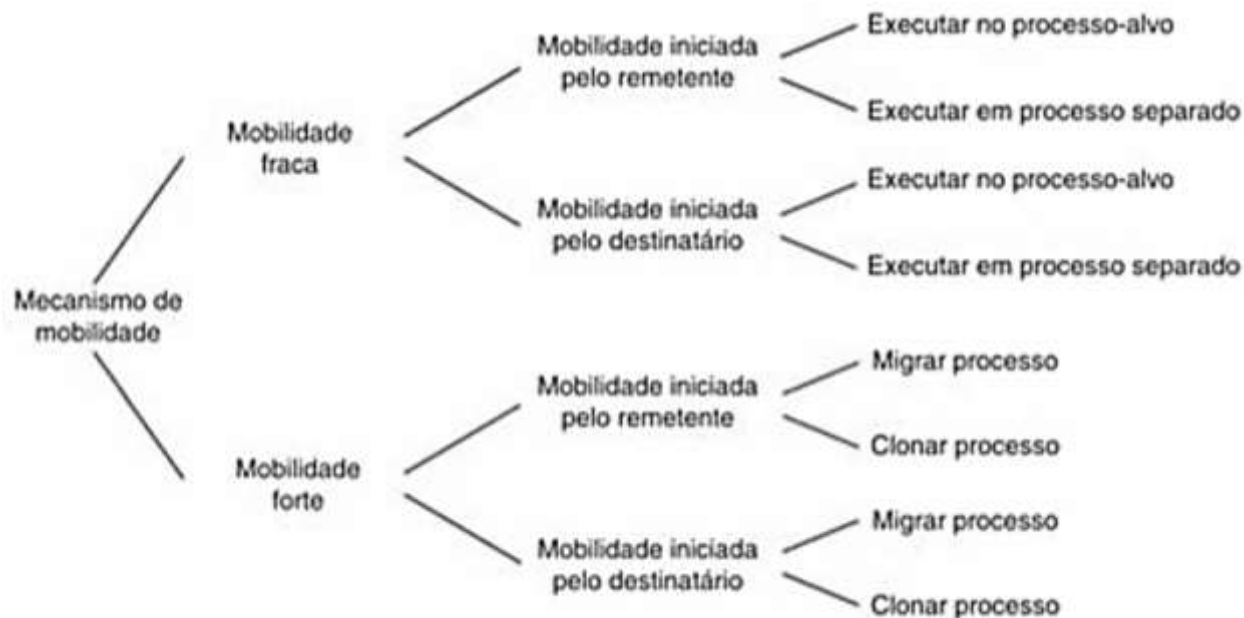


Figura 3.17 Alternativas para migração de código.

Segmento de Recurso

Migração de Código e Recursos Locais

- **Três tipos de vinculações:**
 - Vinculação mais forte, por identificador;
 - Exemplo: URL;
 - Vinculação processo-recurso mais fraca, por valor;
 - Exemplo: bibliotecas do java;
 - Vinculação mais fraca de todas, por tipo;
 - Exemplo: referências a dispositivos locais como monitores e impressoras;

Vinculação recurso-máquina				
	Não ligado	Amarrado	Fixo	
Vinculação processo- recurso	Por identificador	MV (ou GR)	GR (ou MV)	GR
	Por valor	CP (ou MV,GR)	GR (ou CP)	GR
	Por tipo	RB (ou MV,CP)	RB (ou GR,CP)	RB (ou GR)

GR	Estabelecer referência global no âmbito do sistema
MV	Mover o recurso
CP	Copiar o valor do recurso
RB	Vincular novamente o processo ao recurso disponível no local

Tabela 3.2 Ações a executar no que se refere às referências a recursos locais quando da migração de código para uma outra máquina.

Migração em Sistemas Heterogêneos

Migração de Código

- Possíveis problemas para tratar esse tipo de migração
 1. Empurrar páginas de memória para a nova máquina e reenviar as que forem modificadas mais tarde durante a migração do processo.
 2. Parar a máquina virtual corrente; migrar memória e iniciar a nova máquina virtual;
 3. Deixar que a nova máquina virtual puxe novas páginas conforme necessário, isto é, deixar que processos comecem imediatamente na nova máquina virtual e copiar páginas por demanda;

Referências

TANENBAUM, A. S.; STEEN, M. V. Sistemas Distribuídos: princípios e paradigmas. 2.ed. São Paulo, SP: Pearson Prentice Hall, 2008