**Threads**

**Fio de execução**, também conhecido como **linha ou encadeamento de execução**, (em [inglês](https://pt.wikipedia.org/wiki/L%C3%ADngua_inglesa): *Thread*), é uma forma de um [processo](https://pt.wikipedia.org/wiki/Processo_(inform%C3%A1tica)) dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrencialmente. O suporte à *thread* é fornecido pelo próprio [sistema operacional](https://pt.wikipedia.org/wiki/Sistema_operacional) no caso da linha de execução ao nível do núcleo (em [inglês](https://pt.wikipedia.org/wiki/L%C3%ADngua_inglesa): *Kernel-Level Thread (KLT)*), ou implementada através de uma biblioteca de uma determinada [linguagem](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o), no caso de uma *User-Level Thread (ULT*). Uma thread permite, por exemplo, que o utilizador de um programa utilize uma funcionalidade do [ambiente](https://pt.wikipedia.org/wiki/GUI) enquanto outras linhas de execução realizam outros cálculos e operações.

## Vantagens do uso de Threads

* **Simplificação do modelo de programação**: Em muitas aplicações ocorrem múltiplas atividades simultaneamente, e algumas delas podem ser bloqueadas de tempos em tempos. Ao decompor uma aplicação em múltiplos threads sequenciais que executam quase em paralelo, há uma simplificação do modelo de programação.
* **Threads são mais rápidos de criar e destruir** se comparado aos processos, uma vez que não possuem quaisquer recursos associados a eles.
* **Desempenho:** quando há grande quantidade de computação e de E/S, os threads permitem que essas atividades se sobreponham, acelerando a aplicação.

Da mesma forma que os processos sofrem escalonamento, os threads também têm a mesma necessidade. Quando vários processos são executados em uma CPU, eles dão a impressão que estão sendo executados simultaneamente. Com as threads ocorre o mesmo, elas esperam até serem executadas. Como esta alternância é muito rápida, há impressão de que todas as threads são executadas paralelamente.

**Arquitetura e implementação de threads -** Os sistemas operacionais podem implementar o conceito de multithread de forma diferente, considerando critérios de desempenho, concorrência e modularidade das aplicações multithread.

* Threads em Modo usuário - Threads podem ser criadas através de uma biblioteca de funções fora do núcleo do sistema operacional (modo usuário). Neste caso as threads são implementadas pela aplicação do usuário e não pelo sistema operacional. É a aplicação que deve gerenciar as threads (sincronização e comunicação).
* Thread em Modo Kernel - Threads podem ser criadas pelo próprio núcleo do sistema operacional (modo kernel) através de chamadas às rotinas do sistema que oferecem todas as funções para gerencia de threads (sincronização e comunicação). Neste caso é o sistema operacional quem cuida das threads e decide qual thread vai ser executada pelo processador em um determinado momento (escalonamento de threads).
* Thread em Modo Híbrido - Threads podem ser criadas por uma combinação de ambos (modo híbrido). Nesta implementação um processo pode ter várias threads em modo kernel e cada thread em modo kernel pode ter várias threads em modo usuário. O núcleo do sistema operacional reconhece as threads em modo kernel e pode escalona-las individualmente. O problema desta implementação é que não é possível a comunicação entre threads em modo usuário e threads em modo kernel.
* Scheduler Activations - Threads em modo Scheduler Activations usam o melhor do modo kernel e do modo usuário. Usam as facilidades do modo kernel com o desempenho e a flexibilidade do modo usuário. Nesta implementação o núcleo do sistema troca informações com a biblioteca de threads utilizando uma estrutura chamada scheduler activations. Essa implementação evita a troca de modo de acesso (usuario-kernel-usuario) pois o kernel e a biblioteca de threads se comunicam e trabalham de forma cooperativa.

**Threads Compartilham:**

* Espaço de endereçamento;
* Variáveis Globais;
* Arquivos abertos;
* Processos filhos;
* Alarmes pendentes;
* Sinais e rotinas de tratamento de sinais;
* Informações de contabilização

**Threads não Compartilham;**

* -Contador de programa;
* **-Registradores;**
* -Pilha;
* -Estado.

Após um processo ter sido criado, ele começa a ser executado e realiza o seu trabalho, qualquer que seja ele. No entanto, nada dura para sempre, nem mesmo os processos. Cedo ou tarde, o novo processo terminará, normalmente devido a uma das condições a seguir:

1. Saída normal (voluntária);
2. Erro fatal (involuntário);
3. Saída por erro (voluntária);
4. Morte por outro processo (involuntário)

**PROCESS CONTROL BLOCK - PCB** - O processo é implementado pelo sistema operacional através de uma estrutura de dados chamada "bloco de controle de processos" (PROCESS CONTROL BLOCK - PCB). A partir do PCB, o sistema operacional mantém todas as informações sobre o contexto de hardware, contexto de software e espaço de endereçamento de cada processo. No sistema operacional, o bloco de controle de processo (BCP) representa e guarda informações associadas a um processo, como, por exemplo, o seu estado pronto ou em execução.

**Thrashing** é a situação onde uma grande quantidade de recurso computacional é utilizada para fazer uma quantidade mínima de trabalho, com o sistema em um estado contínuo de contenção de recursos. Uma vez iniciado, o thrashing é tipicamente protelado num ciclo vicioso até que algo ocorra para remover a situação original que levou àquele estado.

Geralmente o thrashing se refere a dois ou mais processos acessando um mesmo recurso repetidamente de modo que há uma séria degradação de performance do sistema, porque o sistema está gastando um tempo desproporcionalmente longo somente para *acessar* o recurso compartilhado. O tempo de acesso ao recurso pode ser considerado um desperdício, uma vez que não contribui para nenhum dos processos. Este é geralmente o caso, por exemplo, quando uma [CPU](https://pt.wikipedia.org/wiki/CPU) pode processar mais informação do que aquela disponível na [RAM](https://pt.wikipedia.org/wiki/RAM), consequentemente o sistema vai passar mais tempo se preparando para executar as operações do que propriamente o fazendo.

**Preemptividade ou preempção** é a capacidade de tirar de execução um processo em detrimento de outro (prioridades - escalonamento).

**Estados de Processos**

• Há basicamente quatro estados de execução de um processo:

– Executável: O processo pode ser executado

• Apenas esperando tempo de CPU para processar seus dados

– Dormente: O processo está aguardando algum recurso

• Aguardando uma entrada de teclado ou de rede, um dado de disco, etc.

– Zumbi: O processo está tentando se destruir

• Terminou sua execução mas ainda não teve seus dados coletados.

– Parado: O processo é suspenso (não há permissão para ser executado)

• Proibido administrativamente de executar (c/ um STOP ou TSTP e são reiniciados com CONT)

## Algoritmos de escalonamento

Existem os algoritmos [preemptivos](https://pt.wikipedia.org/wiki/Preemp%C3%A7%C3%A3o" \o "Preempção) e os não preemptivos. Os preemptivos são algoritmos que permitem que um processo seja interrompido durante sua execução, quer seja por força de uma interrupção de [entrada/saída](https://pt.wikipedia.org/wiki/Entrada/sa%C3%ADda), quer seja em decorrência da politica de escalonamento adotada e aplicada por parte do escalonador de processos ou simplesmente por força do término da execução do processo. Após a interrupção deste processo, ocorre o que se chama de [troca de contexto](https://pt.wikipedia.org/wiki/Troca_de_contexto), que consiste em salvar o conteúdo dos registradores e a memória utilizada pelo processo e conceder a outro processo o privilégio de executar na CPU, restaurando assim o contexto deste ultimo processo. Cabe ressaltar que nos algoritmos não preemptivos, por serem utilizados exclusivamente em sistemas monoprocessados, esse fato não ocorre, sendo cada programa executado até o fim.

* [FIFO](https://pt.wikipedia.org/wiki/FIFO)[[1]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-1)[[2]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-2)[[3]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-3) (First in, first out) ou [FCFS](https://pt.wikipedia.org/w/index.php?title=FCFS&action=edit&redlink=1) (First come, first served), em português: "primeiro que entra, primeiro que sai": Onde como seu próprio nome já diz, o primeiro que chega será o primeiro a ser executado, não-preemptivo, ou seja, executa o processo como um todo do início ao fim não interrompendo o processo executado até ser finalizado, apenas uma fila, processos que passam para o estado de pronto vão para o final da fila e são escalonados quando chegam no início. Vantagens: o mais simples entre os processos de escalonamento, até mais do que o Round-Robin, todos os processos tendem a serem atendidos. Desvantagens: muito sensível a ordem de chegada, se processos maiores chegarem primeiro aumentarão o tempo médio de espera, não garante um tempo de resposta rápido.
* [SJF](https://pt.wikipedia.org/wiki/SJF) (Shortest Job First): Onde o menor processo ganhará a CPU e atrás do mesmo formar uma fila de processos por ordem crescente de tempo de execução, não-preemptivo. Desvantagem: baixo aproveitamento quando se tem poucos processos prontos para serem executados.
* [SRT](https://pt.wikipedia.org/wiki/SRT) (Shortest Remaining Time): Neste algoritmo é escolhido o processo que possua o menor tempo restante, mesmo que esse processo chegue à metade de uma operação, se o processo novo for menor ele será executado primeiro, preemptivo. Desvantagem: processos que consomem mais tempo de execução podem demorar muito para serem finalizados se muitos processos com curto tempo de execução chegarem.
* [Algoritmo Loteria](https://pt.wikipedia.org/w/index.php?title=Algoritmo_Loteria&action=edit&redlink=1): O Sistema Operacional distribui tokens (fichas), numerados entre os processos, para o escalonamento é sorteado um numero aleatório para que o processo ganhe a vez na CPU, processos com mais tokens têm mais chance de receber antes a CPU;
* Algoritmo de Prioridade: Como o próprio nome já diz, é um algoritmo onde cada processo no estado de pronto recebe uma prioridade, os processos com maiores prioridades são executados primeiro, prioridades que podem ser atribuídas dinâmica ou estaticamente. É um Algoritmo [preemptivo](https://pt.wikipedia.org/wiki/Preemptividade).
* [Escalonamento garantido](https://pt.wikipedia.org/wiki/Escalonamento_garantido): Este algoritmo busca cumprir promessas de alocação de CPU o mais preciso possível. Uma forma completamente diferente de tratar a questão do escalonamento é fazer certas promessas ao usuário a respeito da performance, e cumpri-las de alguma forma. Uma promessa bem realista e muito fácil de cumprir é a de que se houver N usuários ativos na rede, cada um vai receber em torno de 1/N da capacidade de processamento que um usuário usou para todos os seus processos desde o momento em que tal usuário tornou-se ativo. Como o tempo que cada usuário gastou até o momento é conhecido, é fácil calcular a razão entre o tempo realmente concedido ao usuário e o tempo prometido. A ideia do algoritmo é por para rodar o processo com razões mais baixas, diminuindo, em consequência, as razões mais altas.
* [RR](https://pt.wikipedia.org/wiki/Round-robin_(algoritmo))[[4]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-4)[[5]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-5)[[6]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-6) (Round-Robin): Inspirado na história de Robin Hood onde, na procura de justiça, Robin roubava dos ricos para entregar aos pobres, fazendo assim com que todos no seu reino tivesse o mesmo tanto de bens. Uma das mais simples e robustas entre as atuais técnicas utilizadas para problemas de distribuição de carga, nesse escalonamento o [sistema operacional](https://pt.wikipedia.org/wiki/Sistema_operacional) possui um timer, chamado de [quantum](https://pt.wikipedia.org/wiki/Quantum), onde todos os processos ganham o mesmo valor de quantum para rodarem na [CPU](https://pt.wikipedia.org/wiki/CPU), depois que o quantum acaba e o processo não terminou, ocorre uma preempção e o processo é inserido no fim da fila. Se o processo termina antes de um *quantum*, a CPU é liberada para a execução de novos processos. Em ambos os casos, após a liberação da CPU, um novo processo é escolhido na fila. Novos processos são inseridos no fim da fila. Quando um processo é retirado da fila para a CPU, ocorre uma *troca de contexto*, o que resulta em um tempo adicional na execução do processo. Esta técnica remove a necessidade de criar sistemas para monitoração dinâmica e são obviamente construídas de forma muito mais rápida e prática das que fazem balanceamento através de medições de recursos. Esta técnica foi criada antes mesmo de existirem computadores e é até hoje utilizada em larga escala por inúmeros sistemas com diferentes propósitos. Com exceção do algoritmo RR, FIFO e escalonamento garantido, todos os outros sofrem do problema de [Inanição](https://pt.wikipedia.org/wiki/Inani%C3%A7%C3%A3o_(computa%C3%A7%C3%A3o)) (*starvation*), preemptivo;
* [Múltiplas Filas](https://pt.wikipedia.org/wiki/M%C3%BAltiplas_Filas)[[7]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-7)[[8]](https://pt.wikipedia.org/wiki/Escalonamento_de_processos#cite_note-8): São usadas várias filas de processos prontos para executar, cada processo é colocado em uma fila, e cada fila tem uma política de escalonamento própria e outra entre filas, preemptivo, cada fila tem um determinado nível de prioridade, sendo um dos mais antigos agendadores de prioridade, estava presente no CTSS (Compatible Time-Sharing System - Sistema Compatível de Divisão por Tempo). No algoritmo de Múltiplas Filas, também pode ser aplicado particularmente, em cada fila, diferentes algoritmos como por exemplo, o algoritmo RR ou FCFS.
* Algoritmo Fair-Share: O escalonamento é feito considerando o dono dos processos, onde cada usuário recebe uma fração da CPU e processos são escalonados procurando garantir essa fração. Se um usuário A possui mais processos que um usuário B e os dois têm a mesma prioridade, os processos de A serão mais demorados que os do B.

Todos os algoritmos classificam os processos em estados: Iniciando, Pronto, Executando, Entrada/ Saída e Terminado.

Quando escalonar?

* Quando se cria um novo processo
* No término de um processo
* Quando um processo bloqueia para E/S
* Quando ocorre uma interrupção de E/S (liberação do processo bloqueado)

**Condições para ocorrência de Deadlock:**

Assim, nenhum processo consegue executar **recurso** que precisa, ou liberar recurso que está de posse, ou **ser acordado,** pois o recurso que precisa está ocupado.

*Vale detalhar que****recurso****é uma sequência de eventos necessários ao uso de um processo, assim pode ser dispositivos ou qualquer item compartilhado.*

As **condições para se entrar no estado de Deadlock**:

* **Exclusão Mútua**: todo recurso está ou associado a um único processo ou disponível;
* **Posse e Espera**: processos que retêm recursos podem solicitar novos recursos;
* **Não Preempção**: recursos concedidos previamente não podem ser forçosamente tomados;
* **Espera Circular**: deve haver uma cadeia circular de dois ou mais processos, na qual cada um está à espera de recursos retido pelo membro seguinte dessa cadeia.

Todas essas condições devem ocorrer simultaneamente para que ocorra Deadlock. Se uma dessas condições não ocorrer, não haverá deadlock.

**Gerencia de Memoria**

**Mapa de Bits**

* Divide a memória em **unidades de alocação**.
* Cada unidade corresponde a 1 bit no "mapa de bits" (**livre =0**, **ocupada =1**).
* Quanto **menor** a unidade de alocação, **maior** o mapa de bits (***inversamente proporcional***).
* É uma técnica **simples** em quantidade **fixa** de memória.

**Listas Encadeadas**

* **Segmentos** de memória **alocados** e **disponíveis**.
* A lista de segmentos é **ordenada** por **endereços**.
* Vantagem de **atualização** **rápida** e **simples**.
* first-fit, best-fit, worst-fit, quick-fit, next-fit são **algoritmos** de alocação deste tipo.

**Memoria virtual** emprega**PAGINAÇÃO** e**SEGMENTAÇÃO**.

* **Paginação:** a paginação é um processo da gerência de memória onde o espaço de endereçamento virtual e o espaço de endereçamento real é dividido em páginas do mesmo tamanho, onde cada página virtual possui uma entrada na tabela, com informações que permitem ao sistema localizar a página real correspondente.
* **Segmentação:** a segmentação é um procedimento da gerência de memória, onde os programas são divididos em sub-rotinas e estruturas de dados, e depois são colocados em blocos de informações na memória que possuem tamanhos diferentes com seu próprio espaço de endereçamento.

**Na memória virtual por paginação**, o espaço de endereçamento virtual e o espaço de endereçamento real são divididos em blocos de **mesmo tamanho** chamados páginas. Na **memória virtual por segmentação**, o espaço de endereçamento é dividido em blocos de **tamanhos diferentes** chamados segmentos. Na memória virtual por segmentação com paginação, o espaço de endereçamento é dividido em **segmentos** e, por sua vez, cada segmento dividido em **páginas**, o que elimina o problema da fragmentação externa encontrado na segmentação pura.

**Memory Management Unit (MMU)** tem como função mapear os endereços virtuais para endereços físicos de memória.

As funções do gerenciador de memória incluem determinar quais partes da memória estão ou não em uso; alocar memória para os processos quando dela necessitem; desalocar memória quando processos forem encerrados ou deixem de usá-la ou quando terminem. Ele pode, ainda, controlar as trocas de dados entre a memória principal e o disco rígido, quando a memória principal não é grande o suficiente para conter todos os processos.

**Substituição de páginas na memória**

       Se não houver página disponível na memória física para carga de uma nova página lógica, é necessário copiar uma página da memória principal (RAM) para a secundária (disco) e só depois fazer a carga da página requisitada do disco para a RAM. Esta página escolhida para ser copiada para o disco é chamada de "vítima".  
  
       A eleição da página vítima que será removida da memória RAM é muito importante. Esta decisão é tomada seguindo o algoritmo de substituição de páginas. O ideal é que a página removida seja uma com pouca probabilidade de ser acessada em um futuro próximo (na escala de tempo do sistema em questão).  
  
       Uma boa alternativa é adicionar outros bits de controle à tabela de páginas.

* **Bit de sujeira *(dirty bit)***: Indica quando uma página foi alterada durante a execução do processo. Se a página vítima não foi alterada, ela não precisa ser copiada para o disco, economizando um acesso a disco. Páginas de código e de constantes, por exemplo, nunca são alteradas. O acesso a disco representa quase todo o tempo gasto para atender uma falta de página.
* **Bit de referência *(reference bit)***: Indica quando uma página foi acessada pelo processo. Com o auxílio de marcas de tempo, pode-se determinar se uma página foi acessada em um intervalo e levar este dado em conta ao se escolher uma página para ser liberada na RAM.
* **Bit de tranca *(lock bit)***: Um exemplo de uso é a situação em que um processo P1 pediu leitura de dados em disco. O SO passa esta requisição à controladora de disco e informa que os mesmos devem ser gravados na página física X que corresponde à página lógica Y do espaço de endereçamento de P1. Enquanto a controladora está trabalhando, é interessante trancar a página X.

***As******formas de tratamento de Deadlock****:*

**Ignorar por completo:** Utilizar o Algoritmo Avestruz que é uma abordagem mais simples. "Enterre a cabeça na areia e finja que nenhum problema ocorrerá.".

**Detecção e Recuperação:**

**Preempção**: retirar um recurso de algum outro processo. Nesse método, cabe uma análise da natureza do recurso;

**Restauração de Estado**: armazenar pontos de verificação de processos periodicamente para que esse estado salvo sirva para restaurar o processo se este é encontrado em estado de deadlock;

**Eliminação de Processos**: forma mais grosseira, mas mais simples de quebrar um deadlock é eliminar um dos processos no ciclo de processos em deadlock, assim os outros processos conseguirão seus recursos. Para eliminar o processo, escolher um que possa ser reexecutado desde o início;

**Multiprogramação ou Multitarefa** - nome dado à característica de alguns sistemas operacionais de rodar diversas aplicações simultâneas. Na realidade, na execução multitarefa, o tempo do processador é compartilhado entre as aplicações em execução, e a execução das tarefas passa rapidamente de uma para a outra, criando a ilusão de que as aplicações ocorrem de forma conjunta.

**Time-sharing** - sistemas que compartilham o tempo de uso da CPU entre diversos programas. Também conhecidos como sistemas de tempo compartilhado, conseguem executar diversas tarefas simultaneamente, pois existe a divisão do tempo do processador em pequenos intervalos, denominados fatias de tempo. Caso a tarefa não termine durante a fatia a ela determinada, há uma interrupção e ela volta para a fila de escalonamento, aguardando novamente sua vez.

**Algoritmo FIFO (First-In-First-Out)**

- Algoritmo primeiro a entrar, primeiro a sair

- A página que primeiro foi utilizada será a primeira a ser escolhida.

**Algoritmo LFU (Least-Frequently-Used)**

- Seleciona a pagina menos referenciada (frame menos utilizado)

**Algoritmo LRU (Least-Recently-Used)**

- Seleciona a página que está há mais tempo sem ser referenciada;

- Algoritmo de página usada menos recentemente;

- Pode ser implementada através de uma **lista encadeada;**

- É pouco empregado devido ao seu elevado custo de implementação;

**Algoritmo NRU (Not-Recently-Used)**

- Algoritmos de substituição de páginas não utilizadas recentemente;

- Seleciona as páginas que não foram utilizadas recentemente e não foram modificadas;

* **Multiprocessamento simétrico**: um grupo de processadores trabalha em conjunto compartilhando uma única memória através de um único barramento. Isso torna possível qualquer processador executar uma parte do processo.  Aqui todos os processadores são tratados igualmente. Qualquer processador pode fazer o trabalho de outro processador. Sistemas como Linux, Windows, Mac OS suportam o multiprocessamento simétrico.
* **Multiprocessamento assimétrico**: há um processador mestre que controla a estrutura de dados do sistema.

**Virtualização**

**Tipo I – Bare Metal**

• Hypervisor **roda diretamente no hardware do servidor** (nativo), em vez de se instalar o sistema operacional sobre o hardware, instala-se o hypervisor, e em cima deste ficarão os guests.

• Controla o hardware e o acesso do guest em relação ao que este precisa, entregando componentes virtualizados para cada guest.

• Compartilha os recursos de hardware entre as máquinas virtuais.

Ex.: Vmware ESXi, Microsoft Hyper-V, Citrix Server e KVM.

**Tipo II - hosted**

Também denominado *hosted*, é software que se executa sobre um sistema operacional para oferecer a funcionalidade descrita. Alguns dos hipervisores tipo 2 mais utilizados são:

* Oracle: [VirtualBox](https://pt.wikipedia.org/wiki/VirtualBox" \o "VirtualBox) (grátis), VirtualBox OSE (livre)
* VMware: Workstation (comercial), Server (grátis), Player (grátis)
* [QEMU](https://pt.wikipedia.org/wiki/QEMU) (livre)
* [oVirt](https://pt.wikipedia.org/wiki/OVirt) (livre)
* DRDS (livre)
* Microsoft: [Virtual PC](https://pt.wikipedia.org/wiki/Virtual_PC), Virtual Server;

**Máquina virtual: executa um sistema operacional completo, incluindo o kernel**, exigindo mais recursos do sistema (CPU, memória e armazenamento)

**Contêineres: executa a parte do modo de usuário de um sistema operacional**e pode ser personalizada para conter apenas os serviços necessários para o seu aplicativo, usando menos recursos do sistema.

**Clustering**, em arquitetura de TI, define uma arquitetura de sistema capaz combinar vários computadores *(datacenters)* para trabalharem em conjunto ou pode denominar o grupo em si de computadores combinados. Logo, é possívelresolver o problema da falta de recursos em servidores físicos.

Cada estação é denominada “nodo” e, combinadas, formam o **cluster**. Em alguns casos, é possível ver referências como “supercomputadores” ou “computação em cluster” para o mesmo cenário, representando o hardware usado ou o software especialmente desenvolvido para conseguir combinar esses equipamentos.

**BENEFÍCIOS DA VIRTUALIZAÇÃO**

* Permite aumentar a eficiência de uso de computadores (de todos os portes);
* Permite aumentar agilidade, flexibilidade e dimensionamento do parque de tecnologia;
* Permite simplificar o gerenciamento de ambientes de centros de dados;
* Permite aumentar a produtividade da infraestrutura tecnológica;
* Permite reduzir períodos de ociosidade dos recursos de hardware;
* Permite reduzir a quantidade de servidores físicos e melhorar a disponibilidade de sistemas;
* Permite migração em tempo real de máquinas virtuais em execução de um servidor físico para outro.

**Tipos de virtualização:**

* **Virtualização completa/total:** Técnica que utiliza uma camada de abstração dos recursos físicos reais dos computadores a fim de fornecer ao sistema operacional convidado uma réplica do hardware virtualizado, dispensando a necessidade de modificações no sistema operacional convidado.
* **Paravirtualização:**Técnica que modifica o sistema operacional visitante para chamar o hypervisor sempre que precisar executar uma instrução que possa alterar o estado do sistema – uma instrução sensível/privilegiada.
* **Virtualização Assistida por Hardware:**Técnica em que se reproduz de maneira similar os componentes físicos da máquina, emulando processador, memória cache, ciclos de clock e os demais componentes e processos.
* **Virtualização baseada em contêineres:**Técnica em que os sistemas operacionais convidados compartilham o mesmo núcleo do sistema operacional. É utilizada nos sistemas operacionais modernos, permitindo aumentar a performance de execução e a escalabilidade.

Outra definição:

* **Virtualização assistida por hardware**: reproduz de maneira similar os componentes físicos da máquina, emulando processador, memória cache, ciclos de clock e os demais componentes e processos. Compreende o tipo mais complexo de virtualização, no qual permite que o sistema guests rode isolado no hardware do sistema host. Neste processo, existem circuitos no processador e no controlador de memória que possibilitam ao hardware rodar simultaneamente vários sistemas operacionais.

* **Virtualização de aplicativos**: possibilita a execução de qualquer aplicação, protegendo o sistema operacional de ações que poderiam comprometer a sua performance. Neste processo, as aplicações não usam arquivos, bibliotecas do sistema e chaves de registro, elas apenas são executadas. Acontece o isolamento da aplicação no interior da máquina virtual, tornando-se independente do sistema operacional, uma vez que pode ser executada em qualquer sistema que possua a máquina virtual instalada. Por exemplo, a Java Virtual Machine (JVM), que pode rodar aplicativos Java em qualquer sistema operacional que possua a máquina Java instalada.

* **Virtualização de Storage**: também conhecida como armazenamento. Compreende o fortalecimento de vários dispositivos físicos, reorganizando-os em grupos virtuais para um comum acesso de aplicativos e usuários finais. Esta interação das camadas físicas não interromperá o acesso aos dados na camada lógica de armazenamento. Este processo de virtualização é ramificado em armazenamento de bloco (SANs - Storage Area Network, - e NAS - Network Attached Storage) e armazenamento de arquivo.

* **Virtualização do sistema operacional**: possibilita de vários sistemas operacionais rodem em um mesmo hardware simultânea e isoladamente, fornecendo desempenho, segurança e confiabilidade aos meios englobados. Além disso, com ela é possível englobar tanto software quanto hardware, possibilitando usá-la de inúmeras maneiras, não se limitando somente a uma aplicação ou arquitetura. Para que escolha a mais correta, devem-se considerar as necessidades a serem satisfeitas e os objetivos a serem alcançados.

O **Xen** é um Hypervisor que permite a criação, execução e gerenciamento simultâneos de várias máquinas virtuais em um computador físico. A virtualização funciona abstraindo hardware e dispositivos físicos dos aplicativos em execução nesse hardware.

O  processo de virtualização gerencia e provisiona os recursos do sistema, incluindo processador , armazenamento e recursos de rede. Isso permite que o sistema hospede mais de uma carga de trabalho simultaneamente. E consequentemente, fazendo um uso mais econômico dos servidores e sistemas disponíveis em toda a organização.

**VMWare**

* **VSwitches:** tem por função a mesma de um switch físico: Permitir comunicação entre dispositivos que estejam no mesmo domínio de broadcast, filtrar e encaminhar frames, segmentação de trafego por VLAN, entre outros fatores.
* **O vRealize** é uma plataforma VIRTUAL EM NUVEM HÍBRIDA.
* **O VxRail** é uma infra-estrutura exclusiva que reúne em uma mesma plataforma as tecnologias Dell EMC e VMware e provê a maneira mais fácil e rápida de ampliar e ao mesmo tempo simplificar um ambiente virtualizado.
* **O VMware Virtual SAN™** é uma nova camada de armazenamento definida por software que amplia o VMware vSphere® Hypervisor para agrupar processamento e armazenamento conectado diretamente em pools. O Virtual SAN é incorporado ao kernel do vSphere. Ao agrupar discos e memória flash do servidor em clusters, cria um armazenamento extremamente simples, de alto desempenho e adaptável, projetado especificamente para máquinas virtuais.
* **Vmotion:**  é um característica da Vmware onde a maquina virtual pode ser migrada de um servidor para outro sem parada, ou seja, sem perda de dados ou alguma interrupção durante o procedimento.