

# Sistemas Distribuídos

## Introdução

## Tipos de Escalonamento

### Sistemas Centralizados x Sistemas Distribuídos

#### Vertical:

Adiciona mais recursos, como memória ou RAM, no mesmo computador. Porém, é limitado pela CPU, ou seja, por um único nó.

#### Horizontal:

Adiciona novos computadores, facilitando a distribuição de recursos de forma mais barata. No entanto, os softwares devem ser adaptados para utilização distribuída, ou seja, em vários nós.

---

## Servidores

São máquinas que dão boot no sistema e ocupam um espaço acessível a grande porte, facilitando o armazenamento e o acesso.

### Exemplos:

- Servidor de 1U para rack
  - Servidor BladeSystem de 10U
- 

## Storage

Local onde as informações ficam armazenadas, ou seja, os dados em si. É voltado para armazenamento em larga escala.

---

## Objetivos de um Sistema Distribuído (SD)

### 1. Acessibilidade

Permitir que os usuários acessem de forma simples e facilitada os recursos, garantindo também a segurança e as permissões de acesso.

---

## 2. Transparência

Transparência significa esconder dos usuários que o processamento e os recursos estão distribuídos em vários computadores.

### Tipos de transparência:

- **Acesso:** Esconde diferenças de representação e manipulação dos dados em diferentes sistemas.
- **Localização:** Esconde onde os recursos estão fisicamente. Exemplo: para acessar um sistema não é necessário saber onde ele está hospedado.
- **Desempenho:** O sistema adapta-se à demanda. Exemplo: adição de servidores web para balancear carga em períodos críticos.
- **Mobilidade:** Esconde mudanças de localização dos recursos durante o acesso.
- **Replicação:** Esconde que os recursos são replicados para aumentar disponibilidade ou desempenho.
- **Concorrência:** Esconde dos usuários o fato de outros acessarem um mesmo recurso simultaneamente.
- **Falhas:** Esconde falhas ou quedas de servidores, garantindo ao usuário a continuidade dos serviços.

⚠ Em alguns casos, é necessário que o usuário perceba a distribuição, por exemplo: a localização de uma impressora para retirar um documento.

---

## 3. Abertura

Um sistema aberto garante adaptabilidade a outros sistemas, permitindo diferentes implementações.

### Exemplos:

- Computador pessoal: permite conectar periféricos de diferentes fabricantes.

- Rede TCP/IP: dispositivos com arquiteturas e sistemas operacionais distintos comunicam-se sem dificuldades.
- 

## 4. Escalabilidade

Garantir desempenho e flexibilidade, mesmo com o aumento de usuários e do uso de recursos.

**Escalabilidade pode ser analisada em três dimensões:**

- **Tamanho:** quantidade de usuários e recursos.
- **Dispersão geográfica:** distância entre nós.
- **Administração:** número de domínios administrativos.

⚠ Sistemas centralizados têm baixa escalabilidade.

Com o aumento de usuários em um servidor, a latência cresce devido à distância entre o usuário e o servidor, além da sobrecarga de processamento, armazenamento e rede.


✅ Já o sistema distribuído é muito mais escalável.

**Como projetar um sistema distribuído escalável:**

- **Custo dos recursos físicos:** garantir expansão física de acordo com a demanda.
- **Desempenho:** assegurar qualidade de rede e resposta em sistemas de alta demanda.
- **Esgotamento de recursos de software:** endereços IPv4, por exemplo, podem se esgotar; é difícil prever crescimento futuro.
- **Gargalos:** uso de algoritmos descentralizados (ex: DNS) para evitar pontos únicos de falha ou sobrecarga.

**Técnicas para garantir escalabilidade:**

- **Comunicação síncrona:** só prossegue após receber resposta do pedido.
- **Comunicação assíncrona:** evita bloquear o cliente enquanto espera a resposta.
  - ⚠ Não é adequada para aplicações interativas, pois o usuário aguarda resposta rápida.

-  Exige que o cliente tenha maior capacidade de processamento das tarefas do servidor.
  - **Distribuição das responsabilidades:** dividir um componente em partes menores espalhadas pelo sistema.
  - **Replicação dos componentes:** aumenta a disponibilidade e distribui a carga entre servidores. Réplicas próximas aos clientes reduzem a latência.
    - Exemplo: cache em servidores DNS.
    - Desafio: manter a consistência dos dados entre as réplicas (exemplo: tempo de validade no cache do DNS via *TTL*).
- 

## 5. Concorrência

Recursos distribuídos podem ser acessados simultaneamente por vários clientes.

É comum múltiplos clientes tentarem acessar o mesmo recurso ao mesmo tempo.

É necessário garantir a consistência dos dados entre réplicas e acessos concorrentes. Para isso:

- Operações devem ser sincronizadas e ordenadas.
- O conceito de **semáforo**, dos sistemas operacionais, é utilizado para controlar o acesso concorrente