**Definição:** Sistema no qual os componentes de hardware e software, localizados em computadores interligados em rede, comunicam-se e coordenam suas ações apenas enviando mensagens entre si.

Sistema que é composto de softwares espalhados em diversas máquinas conectados através de uma rede, se comunicando por trocas de mensagens.

**Consequências:**

- Concorrência;
  Softwares concorrendo por recursos em uma rede de computadores.
- Inexistência de um relógio global;
  É complicado trabalhar com um relógio único e não existe um relógio global que sincronize todos os sistemas. **A hora nunca é igual, sempre há uma defasagem por menor que seja mas existe.**
- Falhas independentes.
  Difícil isolar falhas pois pode uma falha em um computador pode ser na memória, disco ou simplesmente uma falha de comunicação.

**Motivação:** Compartilhamento de recursos.

Por exemplo, uma impressora compartilhada por "n" usuários ou um disco com capacidade grande compartilhado por vários usuários. **Pode-se compartilhar recursos mais facilmente com sistemas distribuídos.**

Domínios de aplicação selecionados e aplicações de rede associadas.

| | |
|---|---|
| *Finanças e comércio* | eCommerce e.g. Amazon e eBay, PayPal, online banking |
| *A sociedade da informação* | Web como repositório de informações e os mecanismos de busca, ebooks, Wikipedia; redes sociais: Facebook e Instragram. |
| *Setores de criação e entretenimento* | Jogos online, musica e filme em casa, conteúdo gerado pelo usuário, e.g. YouTube, Netflix |
| *Assistência médica* | Informática na saúde, registro online de pacientes, monitoramento de pacientes; |
| *Educação* | e-learning, ambientes de aprendizado virtual; educação à distância. Ex: Khan Academy |
| *Transporte and logistica* | GPS, serviços de mapas: Google Maps, Google Earth |
| *Ciências* | A computação em grade como tecnologia para permitir a colaboração entre cientistas. |
| *Gerenciamento ambiental* | Tecnologia de sensores para monitorar terremotos, enchentes ou tsunamis |

Figura 1. Exemplos de sistemas distribuídos.

**Exemplo de um sistema de negócios financeiro**

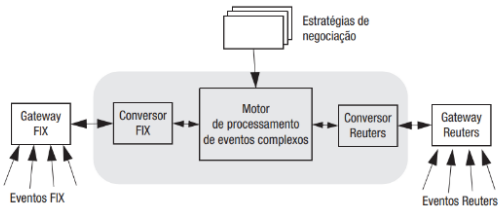


Figura 2. Um exemplo de sistema de negócios financeiros.

Grande sistema distribuído com uso de inteligência artificial que toma decisões de compra e venda de ações baseado em informações recebidas de diferentes fontes.

**Tendências**

- Interligação em rede pervasiva e a internet moderna;
- Computação móvel e ubíqua
    - Desafios
    - ubíquo = onipresente;
      Os componentes da rede realizam cálculos e tomam decisões em movimento. (UBÍQUA)
      **Computação em diferentes componentes, desde uma lâmpada até um super computador.**
    - pervasivo = entranhados;
      **Computador em todo lugar:** no ar condicionado, ventilador, lâmpada interligados em uma grande rede. Casas, cidades inteligentes.
      **Diferentes componentes espalhados em diferentes nós se comunicando através de troca de mensagens.**
- Ex: Jarvis do Zuckerberg, Alexa da Amazon
  uso de webservice.
- Serviços multimídia distribuídos - Garantir QoS
  A ideia é a possibilidade de comunicação por áudio e vídeo pela internet garantindo qualidade de serviço (Quality of service).
    - Ex: Webcasting (lives)

**Uma parte típica da internet**

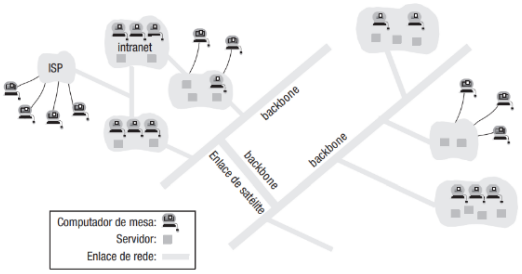


Figura 3. Uma parte típica da Internet.

**ISP:** Provedor de serviços internet para diferentes empresas e clientes (corporativos ou domésticos).
**Intranet:** Redes de computadores de acesso privativo daquela instituição onde os dados circulam apenas dentro daquela infraestrutura. Normalmente conectadas à **Internet** através dos **backbones.**
**Backbones:** ou espinhas dorsais, são estruturas de altíssimas larguras de banda geralmente providas pelos provedores de acesso a internet, permitem a interconexão entre as diferentes redes que compõem a internet. A infraestrutura pode ser baseada em fibras (atráves do oceano por exemplo) ou links de satélite em localizações de difícil acesso geográfico.

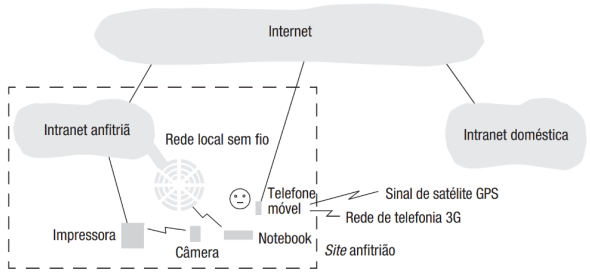


Figura 4. Equipamentos portáteis em um sistema distribuído.

O mesmo dispositivo possui diferentes tecnologias de comunicação e suas aplicações devem se adaptar a cada uma dessas formas de comunicação.

**Tendências**

- Computação distribuída como um serviço;

  Tendência de utilizar aplicações apenas usando recursos computacionais como um serviço. Esses recursos como serviço usam grandes sistemas distribuídos com virtualização.

  O usuário não precisa mais ter o aplicativo localmente, ele pode usar serviços na nuvem.

  - Virtualização;
  - Computação em nuvem;
  - Nuvem: conjunto de serviços baseados na Internet, suficientes para os usuários prescindir do armazenamento local de dados ou aplicativos.
  - Nuvens são implementadas em clusters.

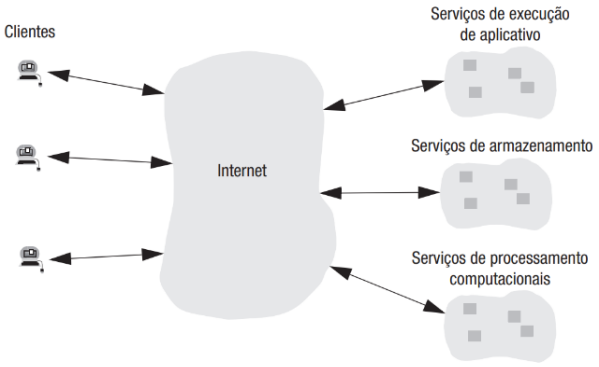


Figura 5. Computação em nuvem.

Diversos clientes conectados a internet onde é disponibilizado aplicação, armazenamento e serviços.

**Tendências**

- Computação distribuída como um serviço;

  O serviço pode ser disponibilizado em diferentes níveis.

  - IaaS, PaaS, SaaS, FaaS em nuvem;
    **Infraestrutura:** hardware + sistema operacional.
    **plataforma:** hardware +

SO e um conjunto de bibliotecas e recursos.
**Software:** software pronto para ser utilizado. (pacote office)
**Função:** recurso para que uma função seja executada. Cobrança feita com base nas chamadas a essa função que utiliza o serviço.

- **Cluster:** conjunto de computadores interligados que cooperam para fornecer um único recurso de computação integrado de alto desempenho.
  A ideia é aglomerar um conjunto de máquinas onde o SD vai unir o hardware e disponibilizar como um só componente integrado.
  Exemplo: 10 máquinas com 1G de memória e 1T de disco, o cluster disponibiliza 10G de memória e 1T de disco (e um ambiente integrado de alto desempenho).
- As soluções blade;

blade = lâmina.
Cada computador (elemento com memória, armazenamento e processamento) é organizado em uma lâmina dentro de um chassi.

O recurso é modular, pode remover e inserir uma lâmina sem prejuízos para a funcionalidade do ambiente (disponibilidade).

**blade -** hardware
**cluster -** integra o hardware em um hardware só.
**ambiente de virtualização -** como software.

- A computação em grade - precursora da computação em nuvem.

**Desafios**

Desafios em sistemas distribuídos.

- **Heterogeneidade**
  Diferentes ambientes e fabricantes concorrem por

clientes e trabalham nos seus próprios padrões para obter bons resultados, porém torna difícil a interação entre diferentes computadores de diferentes fabricantes.
Para esconder, cria-se uma abstração para a **heterogeneidade**, essa camada de software chama-se **Middleware**.

- **Middleware** - camada de software que fornece uma abstração de programação, assim como o mascaramento da heterogeneidade das redes, de hardware, dos sistemas operacionais e das linguagens de programação. Ex: CORBA e Java RMI.

- **Migração de código** - Ex: Applets, Javascript.
  Colocar um código para executar em uma máquina com padrões diferentes e ser executado corretamente.
  Exemplo: sites com formulários utilizam códigos de verificação de validação de campos na máquina do próprio cliente a partir do servidor.
- **Sistemas Abertos** - aquele que pode ser estendido e reimplementado de várias maneiras;
  A ideia é que o sistema pode ser reimplementado diversas vezes, para isso a especificação dos sistema é disponível para qualquer desenvolvedor.
  A internet é um exemplo, os padrões são disponibilizados para quem quiser implementar recursos que possibilitem a interação do sistema com a internet.
  - As RFCs;
    **request for comments**.
    Nelas são disponibilizadas as especificações detalhadamente garantindo que sejam implementados por diferentes fabricantes e permitindo um ambiente menos heterogêneo.
  - Os padrões W3C.
    A ideia é padronizar os recursos da web.
- Segurança: Confidencialidade, Integridade e Disponibilidade;
  **Confidencialidade:** só acessar o dado quem for dono daquele dado ou a quem for dado acesso.

**Integridade:** Dados inalterados independente de mudança de uma máquina para outra.
**Disponibilidade:** que não haja ataques que acabem com a disponibilidade desses serviços.

- Segurança de código móvel;
  Applets apresentaram diversas falhas nesse caso.
- Ex: Ataque DoS;
  Ataque de negação de serviço: a ideia é enviar uma quantidade de requisições maior que o servidor pode processar para que o processo trave e não seja possível receber requisições.
  Para evitar deve-se investir em segurança;

- Escalabilidade - Sistema que permanece estável mesmo quando há um aumento significativo no número de recursos e usuários;
  Independente do aumento do número de usuários, o tempo de resposta deve continuar estável.
  - Exemplo de não escaláveis:
    Esgotamento do IPV4 e DNS centralizado.
    IPV4 possui sério problema com escalabilidade pois a quantidade de endereços IP não foi pensada levando em conta como a internet iria crescer. Assim, não há mais endereços disponíveis para uso e foi necessário adaptar o sistema para servir aos novos clientes.
- T**ratamento de falhas:** Tolerância, recuperação, redundância.
  A possibilidade de falha é maior, há vários nós conectados pela rede, o nó ou a rede podem falhar.
  **redundância:** serviço hospedado em mais de uma máquina.
  **recuperação:** reinicialização automática.
- **Concorrência:** Uso de semáforos para garantir sincronização;
  Garantir que haja a pena uma máquina utilizando um recurso, utilizando semáforos para travar o semáforo sempre que for usar o recurso.

- **Qualidade de serviço:** Desempenho, adaptabilidade. Ex: Apps multimídia.
  Garantir um **desempenho** mínimo de transmissão.
  **Adaptabilidade:** se houver uma queda de velocidade o sistema deve reduzir a qualidade do vídeo.

**Desafio: Transparência**

Garantir que o usuário não saiba a complexidade do ambiente que está acessando.

- **Transparência de acesso:** permite que recursos locais e remotos sejam acessados com uso de operações idênticas.
  A forma como se abre o arquivo no google drive deve ser parecida com a forma de abrir um arquivo no próprio HD.
  - Ex: Google Drive.
- **Transparência de localização:** permite que os recursos sejam acessados sem conhecimento de sua localização física ou em rede (por exemplo, que prédio ou endereço IP).
  Os recursos devem ser acessados sem ser necessário saber onde fisicamente estes recursos estão.
  - Ex: URL.
- **Transparência de concorrência:** permite que vários processos operem concorrentemente, usando recursos compartilhados sem interferência entre eles.
  Vários processos podem operar de forma concorrente pelo uso de um recurso sem apresentar falhas no resultado final.
  - Ex: Impressora.
- **Transparência de replicação:** permite que várias instâncias dos recursos sejam usadas para aumentar a confiabilidade e o desempenho, sem conhecimento das réplicas por parte dos usuários ou dos programadores de aplicativos.
  Cria cópias de um recurso sem conhecimento do usuário ou desenvolvedor para o caso de algum recurso cair.
  - Ex: Keepalived.

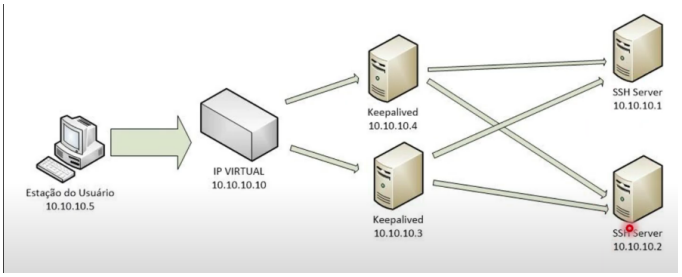


Figura 6. Exemplo de keepalived.

- **Transparência de falhas:** permite a ocultação de falhas, possibilitando que usuários e programas aplicativos

concluam suas tarefas, a despeito da falha de componentes de hardware ou software.
O usuário continua executando suas tarefas independentemente das falhas. Por exemplo, se falha o envio de um email, é visível o servidor de email tentando enviar o email.
    - Ex: Email, keepalived.
- **Transparência de mobilidade:** permite a movimentação de recursos e clientes dentro de um sistema, sem afetar a operação de usuários ou de programas.
Permite que o usuário se movimente dentro do sistema sem ocorrer falhas.
    - Ex Telefone celular.
- **Transparência de desempenho:** permite que o sistema seja reconfigurado para melhorar o desempenho à medida que as cargas variam.
Permite que o ambiente seja reconfigurado para melhorar o desempenho, ou seja aumento na quantidade de recursos, de acordo com o aumento das requisições.
    - Ex: Kubernetes.
- **Transparência de escalabilidade:** permite que o sistema e os aplicativos se expandam em escala, sem alterar a estrutura dos sistema ou os algoritmos de aplicativo.
Aumento da escala do sistema sem alterar a estrutura do sistema, do ponto de vista do desenvolvedor.
    - Ex: Kubernetes.

**Estudo de caso: a WEB**

- Arquitetura básica da Web:
    - HTML;
    Linguagem usada para descrever os objetos que vão ser utilizados na web.
    - URL;
    Tecnologia para identificar e localizar um serviço na web.
    - HTTP;
    Protocolo que garante o tráfego de dados entre cliente e servidor.
- Página dinâmica - CGI (Common Gateway Interface);
Usa a ideia de páginas dinâmicas, adapta-se às entradas do usuário. Usa-se php. asp etc...
- Serviços Web (Web Services)

A ideia é garantir a interação entre programas usando a própria infraestrutura da web.

- Programas como clientes Web;
- O HTML é insuficiente para interações por meio de programas;
- A linguagem XML;
  Garante que seja possível criar novas tags, utilizada para envio e recepção de dados em servidor.
- Utiliza os métodos GET, POST, PUT e DELETE do HTTP;
- A arquitetura REST;
  Estrutura previamente estabelecida de como as aplicações devem ser construídas para garantir a interação entre cliente e servidor.
- A web semântica;
  Dar sentido ao resultado que é retornado ao usuário.
- Uso de cache e proxies.
  Garante tempo de resposta menor para as requisições.

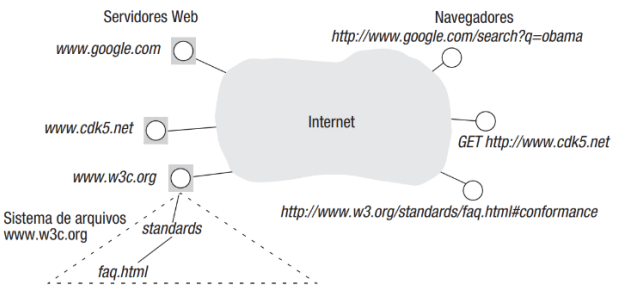


Figura 7. Servidores e navegadores Web.