Sistemas Distribuídos

Prof. Marcelo Veiga Neves
marcelo.neves@pucrs.br

Conteúdo

- Definição de sistemas distribuídos
- Histórico
- Vantagens e desvantagens
- Exemplos de sistemas distribuídos
- Requisitos/considerações de projeto
- Conceitos básicos

Definição de Sistemas Distribuídos

- Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente (TANENBAUM)

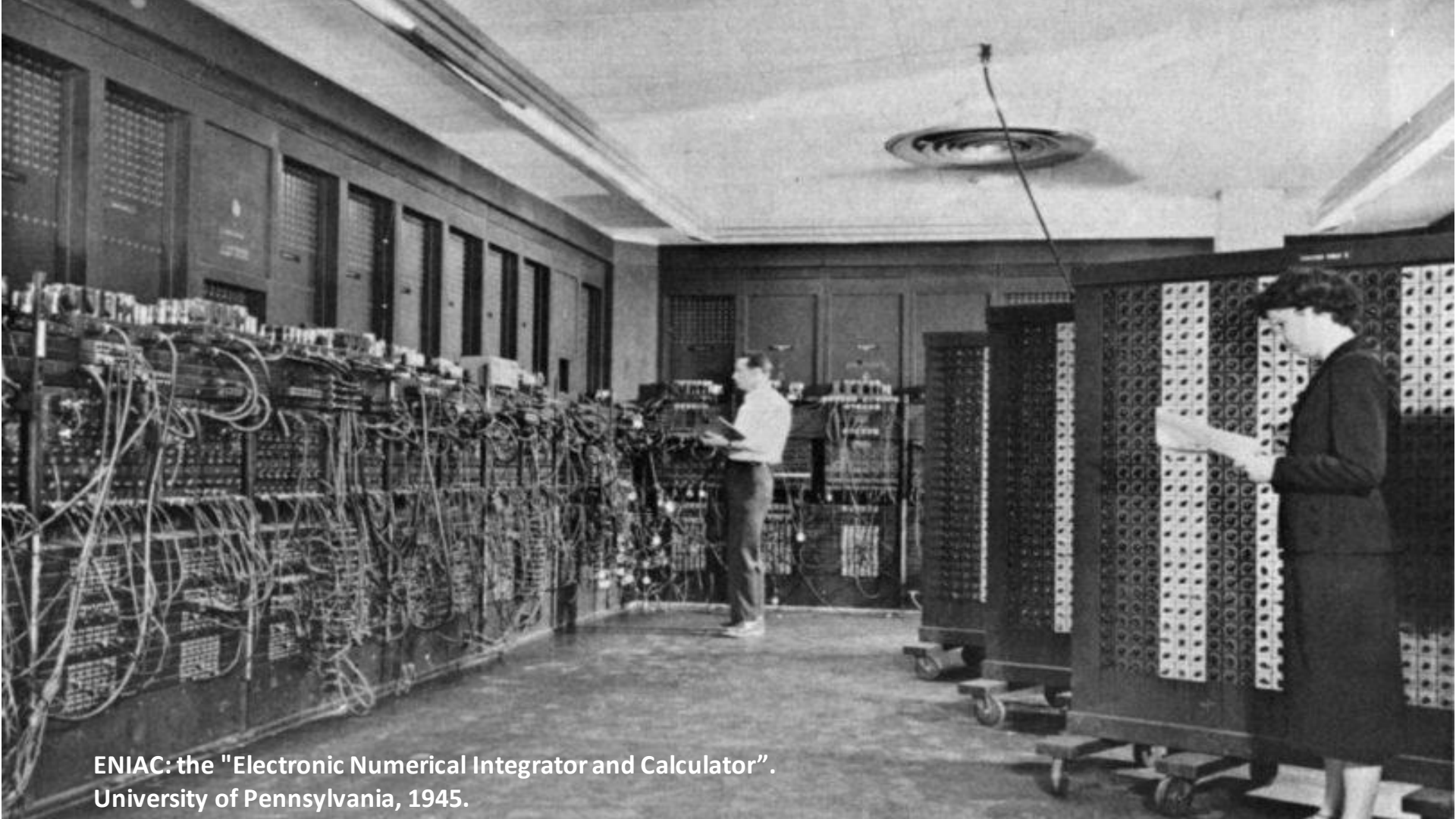
- Dois aspectos:
  - Hardware: autonomia/independência
  - Software: sistema único

Principais Caracterísiticas

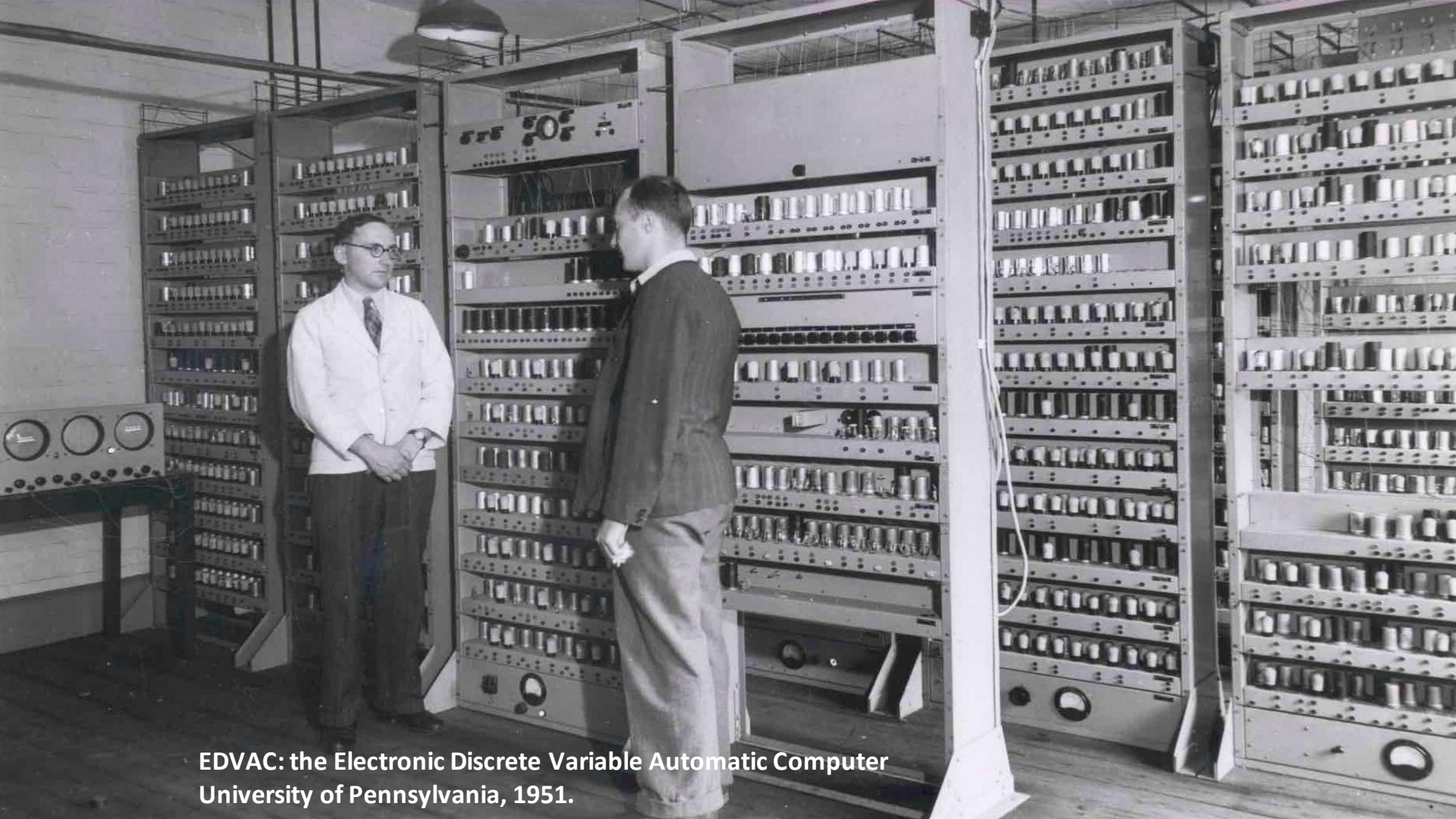
- Computadores autônomos, porém com abstrações que permitem que eles sejam vistos como um sistema único
- Execução concorrente
- Compartilhamento de recursos
- Troca de mensagens (comunicação via rede)
- Inexistência de relógio global
- Falhas independentes
- Muitas vezes componentes e recursos redundantes

Histórico – Mainframes (Anos 60 e 70)

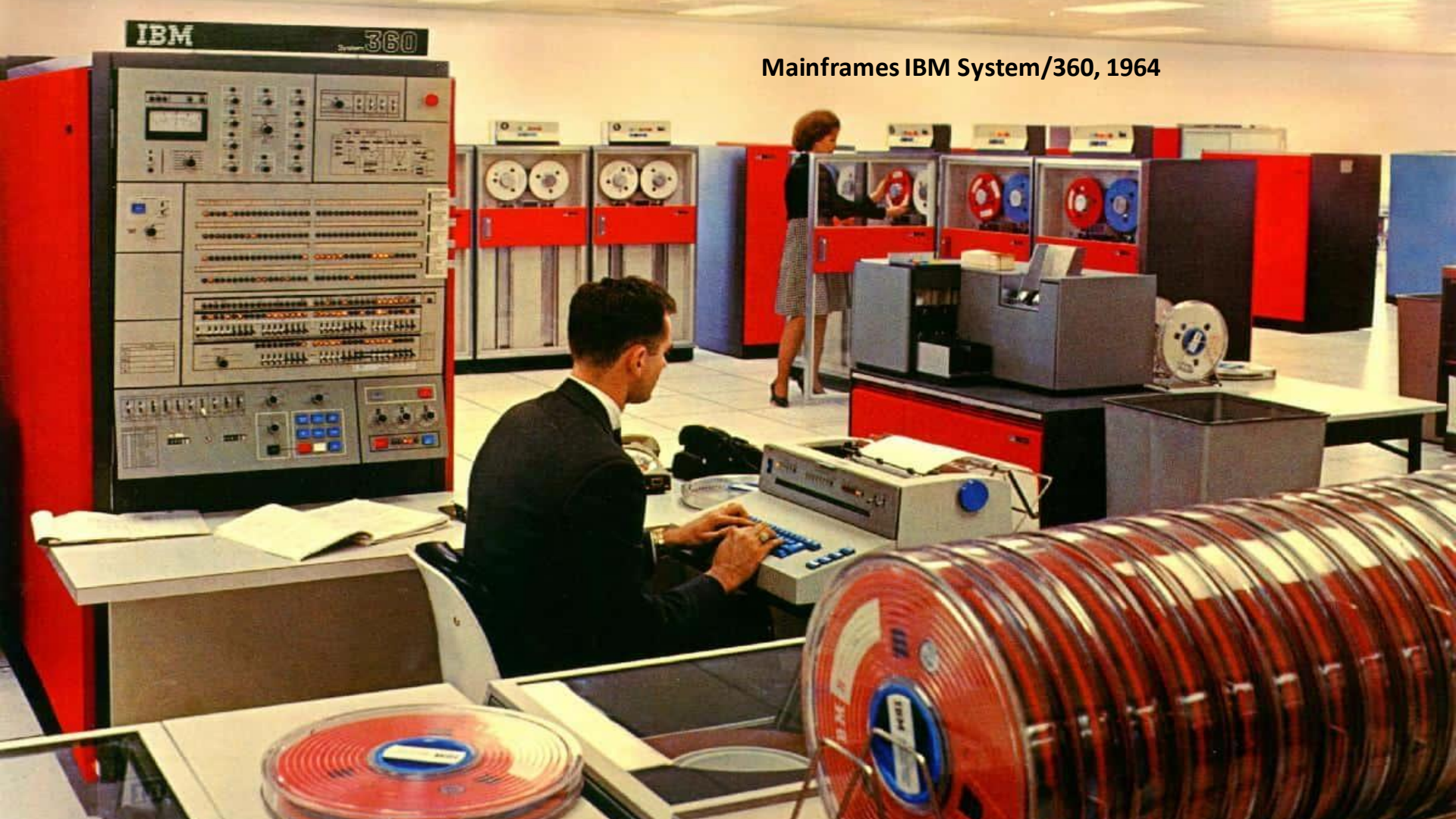
- Computadores iniciais: grandes e caros
- Mainframes IBM (anos 60 e 70)
  - Centralização do processamento de dados de uma organização
  - Time-sharing – compartilhamento da máquina entre diversos usuários através de “terminais burros”



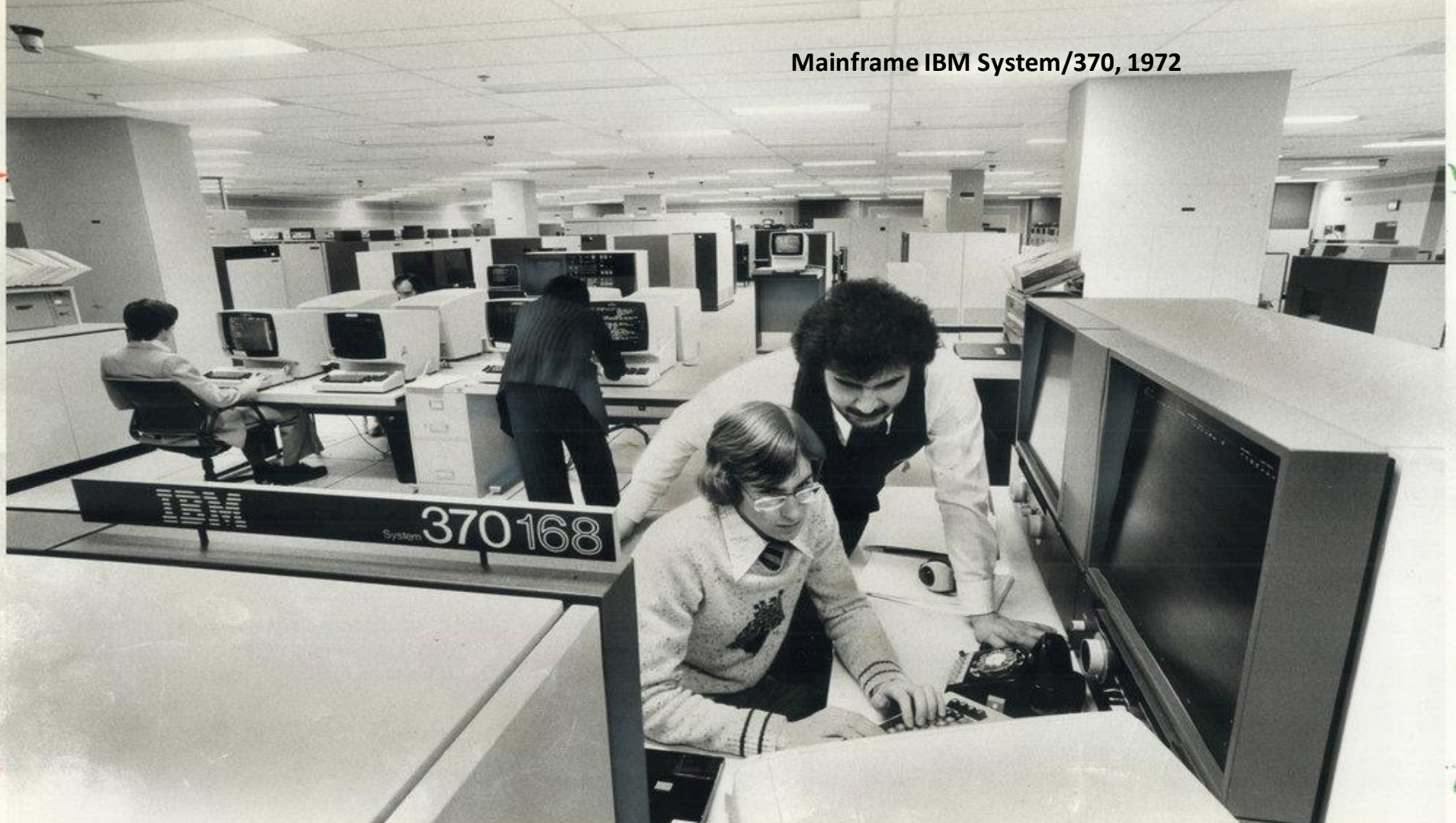
ENIAC: the "Electronic Numerical Integrator and Calculator".
University of Pennsylvania, 1945.



EDVAC: the Electronic Discrete Variable Automatic Computer
University of Pennsylvania, 1951.



Mainframes IBM System/360, 1964



Mainframe IBM System/370, 1972

Histórico – Redes (Anos 60 e 70)

- Final dos anos 60 e início dos anos 70:
  - Unix (Bell Labs, 69)
  - Surgimento das redes:
    - ARPANet (DoD, 69) – WAN
    - Ethernet (Xerox Palo Alto, 73) – LAN
- Final dos anos 70: Protocolos TCP/IP

Histórico – Downsizing (Anos 70 e 80)

- Evolução do hardware (downsizing): redução do tamanho, do preço, aumento da velocidade

- Surgimento dos microprocessadores (anos 70):
- Microprocessador Intel 8080 (Intel, 1974)
- Linguagem BASIC para 8080 (Microsoft, 1975)

- Surgimento do computador pessoal (anos 80):
- IBM PC (1981)
- Sistema operacional MS-DOS (Microsoft, 1981)

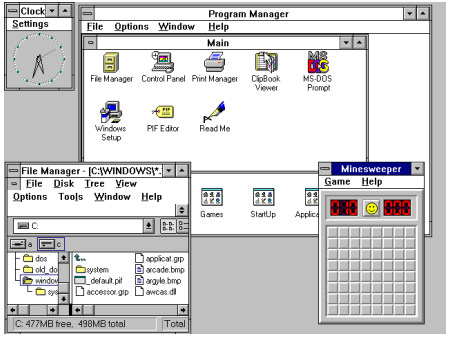
- Popularização das LANs (anos 80):
- Comunicação entre PCs, estações de trabalho e mainframes
- PCs emulavam terminal para acessar mainframe

Bill Gates e Paul Allen, Microsoft, 1981.

IBM PC, 1981

Compaq Portable Computer, 1983

Apple Macintosh, 1984



MS Windows 3.0, 1990

Intel Pentium, 1993

Sun Ultra 60 Workstation, 1997

Histórico – Internet (Anos 90)

- Surgimento do Microsoft Windows (anos 90)
  - Descentralização das aplicações (cliente-servidor)

- Surgimento do Linux (anos 90)
  - Melhor custo e desempenho em relação ao Windows
  - Tornou-se amplamente adotado em servidores

- WWW e Internet (anos 90)
  - Organizações realizando comunicação através da rede pública além das redes privadas
  - Aumento explosivo dos dispositivos conectados

Histórico – Clusters computing (Anos 90/2000)

- Anos 90: clusters Beowulf
- Clusters de computadores
  - Alto desempenho, alta disponibilidade, balanceamento de cargas

Cluster Beowulf, NASA, 1995

IBM Blue Gene/P, 2007

Histórico – Cloud computing (Anos 2000)

- Virtualização
  - Consolidação de servidores
  - Acesso compartilhado a recursos (Multi-tenant)
  - Melhoria no uso dos recursos
- Cloud computing
  - Modelo pay-per-use
  - Acesso sob demanda (provisionamento/liberação self-service)
  - Recursos compartilhados (virtualização)
  - Executa em grandes data centers


Google data center, Council Bluffs, Iowa, USA


Google data center, The Dalles, Oregon, USA


Facebook data center,
North Carolina, USA.

Vantagens

- Economia
  - Revogação da Lei de Grosh: desempenho é proporcional ao custo^2
  - Válida para mainframes
- Velocidade/desempenho
  - 10.000 CPUs x 50MIPS = 500.000 MIPS
  - Uma CPU (??) para isto deveria executar 1 instrução a cada 0.002 nanoseg (2 picoseg). (Velocidade da luz 0.6mm em 2 picoseg)

Vantagens

- Algumas aplicações são naturalmente distribuídas
- Confiabilidade (reliability)
  - Redundância: % fora do ar, % perda de performance
  - Aviação, reatores nucleares, …
- Expansibilidade/escalabilidade
  - Aumentar poder de processamento sem se desfazer daquilo que já possui de maneira gradativa

Vantagens

- Compartilhamento de dados
  - Ex: base de dados compartilhada
- Compartilhamento de periféricos
  - Economia (ex: impressoras)
- Comunicação/colaboração
  - Ex: chat, e-mail, etc.
- Flexibilidade
  - Melhor aproveitamento dos recursos

Desvantagens

- Complexidade de desenvolvimento e gerenciamento
- Dificuldade de depuração
- Dificuldade de garantia de segurança
- Rede pode causar problemas (gargalo, indisponibilidade)

Exemplos de sistemas distribuídos

- Internet
- Intranets
- Computação móvel e ubíqua
- Cluster de computadores
- Aplicações paralelas e distribuídas
- Jogos em rede
- Compartilhamento de arquivos na *web* (P2P)
- Criptomoedas (Bitcoin)
- Whatsapp, etc.

Requisitos/Considerações de Projeto

- Heterogeneidade
- Padronização/Abertura
- Segurança
- Escalabilidade
- Tratamento de falhas
- Concorrência
- Transparência

Considerações de Projeto

- Heterogeneidade
  - Um SD é formado por diferentes redes, SOs e processadores com representações internas de dados diferentes
  - Solução = *middleware*
  - Linguagens como Java são uma alternativa interessante
- Padronização/Abertura
  - Padrões abertos com especificação e documentação disponíveis, permitindo entender e estender o sistema
  - Independência de fornecedor
- Segurança
  - É preciso garantir confidencialidade, integridade e disponibilidade
  - Uso de soluções baseadas em criptografia

Considerações de Projeto

- Escalabilidade
  - O sistema deve funcionar efetivamente e eficientemente com qualquer número/tamanho de usuários, dados e recursos
  - Evitar a centralização
  - Usar estruturas de dados adequadas
- Tratamento de falhas
  - Objetivo = manter a disponibilidade do sistema mesmo na ocorrência de falhas
  - Detecção, mascaramento, tolerância, recuperação
  - Palavra-chave = redundância
- Concorrência
  - É uma propriedade natural de um SD
  - É preciso garantir exclusão mútua no acesso a recursos compartilhados

Considerações de Projeto

- Transparência
  - Esconder a serapação física de componentes de forma que o sistema seja percebido como um sistema único
  - Tipos de transparência:
    - Acesso, localização, replicação, falhas, migração, concorrência, desempenho

Considerações de Projeto

- Transparência de acesso
  - usuário não distingue entre acesso a recurso local ou remoto
  - interface do usuário - conjunto de chamadas de sistema tem que ser projetadas para fazer sentido em sistemas centralizados ou distribuídos - acessar recursos locais ou distantes
  - facilidade de esquema de nomeação global

Considerações de Projeto

- Transparência de localização
  - movimentação de recursos e usuários a vontade
  - recursos mantêm mesmo nome, independentemente de localização
  - usuários acessam mesmos recursos a partir de qualquer nodo e acesso
- Transparência de replicação
  - replicação: aumento de desempenho e confiabilidade
  - gerência de réplicas: nomeação de réplicas, mapeamento de nome dado pelo usuário para réplica apropriada do recurso, etc.

Considerações de Projeto

- Transparência de falhas
  - mascarar dos usuários as falhas parciais do sistema
  - continuidade do funcionamento, talvez de maneira degradada, em presença de falhas
- Transparência de migração
  - razões: manter desempenho, confiabilidade e segurança
  - escolha de objeto (processo) a migrar
  - comunicação deve continuar de maneira transparente

Considerações de Projeto

- Transparência de concorrência
  - competição por recursos - necessidade de:
    - ordenação de eventos
    - exclusão mútua
    - *no-starvation*
    - *no dead-lock*
- Transparência de desempenho
  - reconfiguração da carga do sistema para aumentar desempenho
  - facilidades de alocação de recursos e definição do momento da migração de "trabalho" para homogeneizar a carga dos nodos (balanceamento de cargas)

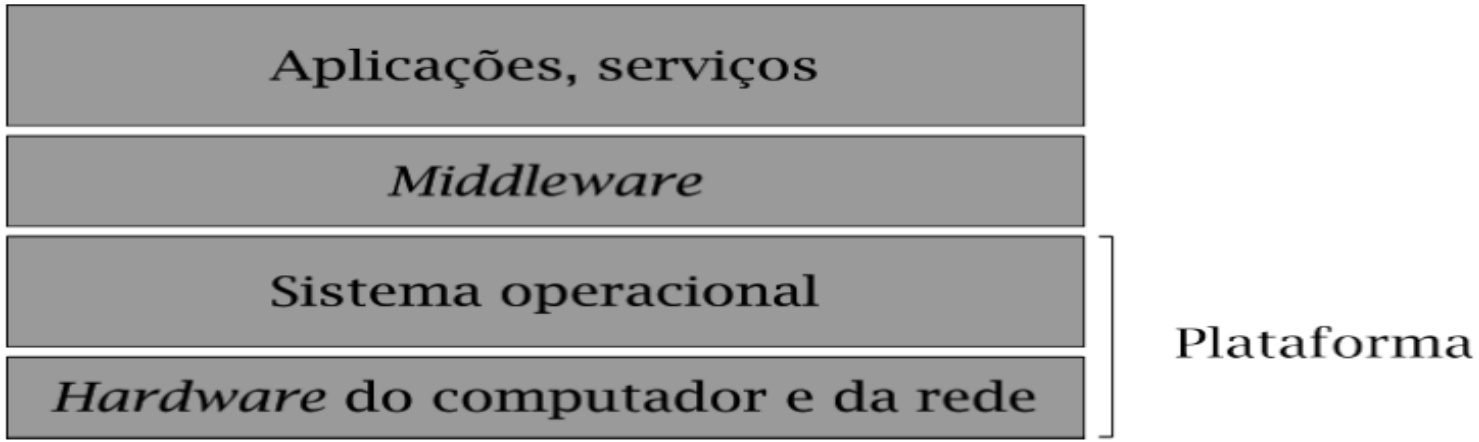
Conteúdo

- Programação distribuída
- Sistemas distribuídos
- Histórico
- Vantagens e desvantagens
- Exemplos de sistemas distribuídos
- Requisitos/considerações de projeto
- Conceitos básicos

Conteúdo

- ~~Programação distribuída~~
- ~~Sistemas distribuídos~~
- ~~Vantagens e desvantagens~~
- ~~Histórico~~
- ~~Exemplos de sistemas distribuídos~~
- ~~Requisitos/considerações de projeto~~
- Conceitos básicos

Conceitos Básicos



Conceitos Básicos

- O que é plataforma?

Conceitos Básicos

- O que é plataforma?
  - Plataforma: camadas de hardware e software de nível mais baixo
  - Exemplos:
    - Intel x86/Windows
    - Intel x86/Linux
    - Intel x86/Solaris
    - SPARC/SunOS
    - PowerPC/MacOS

Conceitos Básicos

- O que é *Middleware*?

Conceitos Básicos

- O que é *Middleware*?
- *Middleware*: camada entre o SO e as aplicações, que mascara a heterogeneidade e provê um modelo mais conveniente de programação
  - Corresponde a um conjunto de rotinas, processos e eventualmente objetos que interagem entre si para dar suporte para que as aplicações possam se comunicar e compartilhar recursos de uma forma mais conveniente
- Exemplos:
  - Sun RPC (Remote Procedure Calls), CORBA (Common Object Request Broker Architecture), Java RMI, etc.
  - Outros exemplos de middlewares modernos:
    - Microsoft .NET, Sun J2EE, Google AppEngine

Conceitos Básicos

- Serviços e interfaces?

Conceitos Básicos

- Serviços e interfaces?
- Serviço: componente que gerencia recursos e apresenta alguma funcionalidade a usuários e aplicações
- Interface: relação de operações de um serviço

Stateless vs Stateful

- *Stateless* significa que o estado do serviço não é levado em consideração entre duas invocações remotas
  - Cada requisição carrega suas próprias credenciais e é autenticada individualmente
  - Servidores não mantêm informação de estado sobre clientes
  - Não necessita estabelecer e fechar conexões
  - Uma queda no servidor não afeta os clientes
  - É simples
  - Chamadas devem conter todo o estado (mais longas)

Stateless vs Stateful

- *Stateful* significa que as requisições são orientadas à conexão, isto é, cada requisição está relacionada a alterações que foram feitas por requisições prévias
  - Servidores mantêm informação de estado sobre clientes
  - Chamadas mais curtas (contêm no mínimo um identificador)
  - Problemas com quedas de servidores e clientes

Referências

- Material baseado em slides dos Profs. Roland Teodorowitsch, Avelino Zorzo, Celso Costa, Fernando Dotti e Luiz Gustavo Fernandes
- E nos seguintes livros:
  - Distributed Operating Systems - Concepts and Design, Pradeep Sinha
  - Distributed Systems: Principles and Paradigms, Andrew S. Tanenbau