Sistemas Distribuídos

Introdução

Aplicações em rede de computadores

- As redes de computadores atualmente estão em todos os lugares.
  - Ex.: Internet, redes de telefones móveis, redes corporativas, em fábricas, em campus, em casas etc.
- Motivações:
  - Muitas CPUs em um sistema operando em paralelo podem ter mais poder de processamento que 1 mainframe por um menor custo
  - Algumas aplicações são distribuídas por natureza
    - Trabalho cooperativo, jogos em rede, P2P
  - Maior confiabilidade
  - Crescimento incremental

Definição

Sistema distribuído um sistema no qual os componentes de hardware e software, localizados em computadores de uma rede, comunicam e coordenam suas ações somente pela troca de mensagens (Coulouris)

- Conseqüências desta definição:
  - Concorrência de componentes
  - Ausência de relógio global
  - Falhas independentes

Definição (cont)

Computação distribuída ou sistema distribuído é o processo de agregar o poder de várias componentes computacionais para colaborativamente executar uma única tarefa computacional de modo coerente e transparente de tal forma que elas aparentam ser um sistema único e centralizado (Wikipedia)

Definição (cont)

Coleção de computadores independentes
que se apresentam ao usuário como
um único sistema coerente (Tanenbaum)

- Essa definição implica em:
  - Máquinas autônomas (camada de software unifica e torna visão homogênea)
  - Usuários pensam que estão lidando com um único sistema

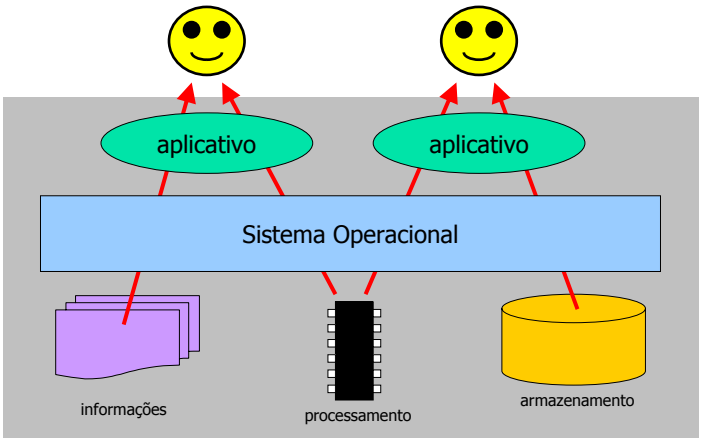
Definição (cont)

- Dois linhas de estudo
  - Sistema distribuído
    - Aplicação distribuída
  - Sistema operacional distribuído
    - Sistema operacional em uma rede

Classificação

- Sistema Operacional Centralizado
  - Sistema com um computador
  - Um usuário acessa recursos locais
- Sistema Operacional de Rede
  - Vários sistemas distintos
  - Recursos compartilhados entre usuários
  - Usuários precisam saber onde estão os recursos
- Sistema Operacional Distribuído
  - Sistemas distintos, mas visão unificada
  - Recursos estão acessíveis de forma transparente

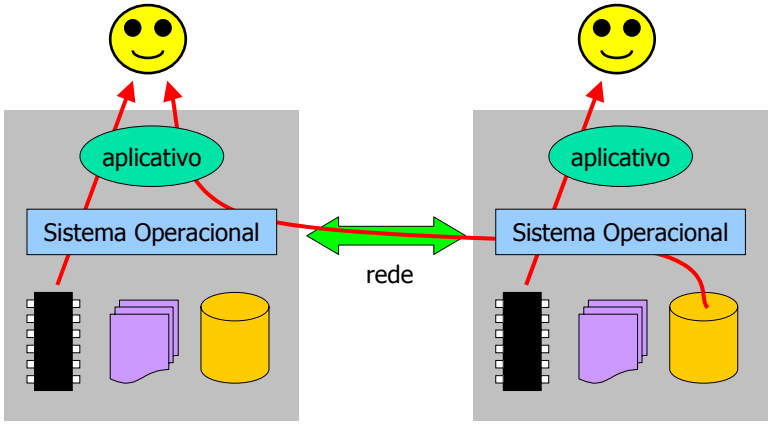
Sistema Operacional Centralizado



Sistema Operacional Centralizado

- Aplicado a sistemas convencionais
  - Recursos centralizados
  - Arquiteturas mono ou multi-processadas
  - Sistemas multi-tarefas e multi-usuários
- Principais características
  - Compartilhamento de recursos através de interrupções
  - Todos os recursos são acessíveis internamente
  - Comunicação entre processos via memória compartilhada ou através de facilidades providas pelo núcleo do sistema
- Objetivos
  - Tornar virtuais os recursos do hardware
  - Gerenciar uso dos recursos locais
  - Sincronizar atividades

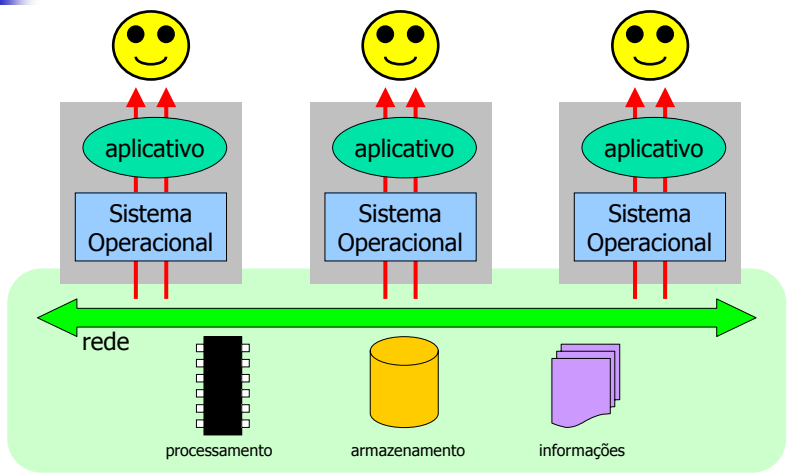
Sistema Operacional de Rede



Sistema Operacional de Rede

- Coleção de computadores conectados através de uma rede
  - Cada computador possui seu SO local
  - Cada máquina possui alto grau de autonomia
- Implementação relativamente simples
  - SOs incorporam módulos para acessar recursos remotos
  - Comunicação entre sistemas através de protocolos de transporte (Sockets ou RPC)
- Transferências explícitas
  - O usuário deve conhecer a localização dos recursos
  - Os recursos pertencem a computadores específicos
- Exemplos:
  - Compartilhamento de impressoras e arquivos
  - Web, E-mail
  - Serviços de autenticação

Sistema Operacional Distribuído



Sistema Operacional Distribuído

- Objetivos:
  - Construção de um ambiente computacional virtual
  - Localização dos recursos é abstraída
  - Localização do processamento é abstraída
  - Mecanismos transparentes de distribuição, replicação e tolerância a faltas
- O usuário vê o sistema como um ambiente virtual, e não como um conjunto de computadores conectados por uma rede
- O SO distribuído deve:
  - Controlar a alocação de recursos para tornar seu uso eficiente
  - Prover um ambiente de computação virtual de alto nível
  - Esconder a distribuição dos recursos e do processamento

Quadro comparativo

Tipo	Serviços	Objetivos
Centralizado	Gerenciamento de processos, memória, dispositivos, arquivos	Gerenciar recursos Máquina estendida Virtualização
de Rede	Acesso Remoto Troca de Informações	Compartilhar recursos Interoperabilidade
Distribuído	Visão global dos recursos (processadores, memória, arquivos, usuários, tempo) Uso do poder computacional	Unificar os computadores em uma visão global Diversas transparências

Tabela Comparativa

	Centralizado (mono ou multi-processado)	de Rede	Distribuído
Se parece com um único processador virtual ?	Sim	Não	Sim
Todas as máquinas executam o mesmo sistema operacional ?	Sim	Não	Sim
Quantas cópias do sistema operacional existem ?	1	N	N
Como a comunicação ocorre ?	Memória compartilhada	Arquivos compartilhados Protocolos de transporte	Trocas de mensagens
Há uma única fila de execução ?	Sim	Não	Não

Vantagens dos SD

- Economia
  - aproveitar recursos ociosos; é mais barato ter vários processadores interconectados do que um supercomputador
- Distribuição inerente
  - algumas aplicações são distribuídas por natureza
- Tolerância a falhas
  - em caso de falha de uma máquina, o sistema pode sobreviver, mesmo com desempenho degradado
- Crescimento incremental
  - o poder computacional pode ser aumentado através da inclusão de novos equipamentos.
- Flexibilidade
  - Maior flexibilidade na alocação dos recursos, permitindo que usuários compartilhem dados, processamento e dispositivos.

Desvantagens dos SD

- Aplicações mais complexas
  - Pouco software de alto nível disponível para sistemas distribuídos.
- Segurança
  - Necessidade de construir mecanismos para controle de acesso às informações
- Dependência da rede
  - Falhas
  - Capacidade de tráfego insuficiente

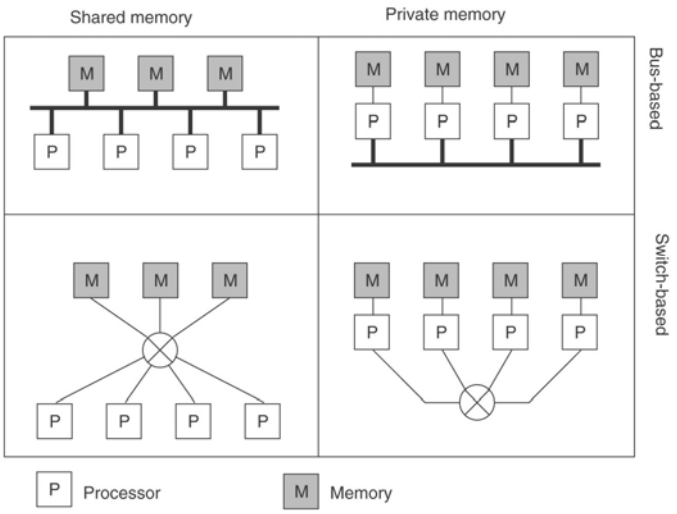
Exemplos

- Internet
  - Comunicação: troca de pacotes
  - Serviços: WWW, email, ftp etc
  - Provedor
  - Backbone
- Computação móvel
  - Redes sem fio
  - Laptop, PDA

Conceitos de hardware

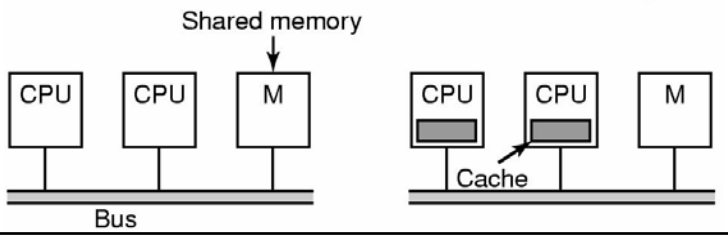
- Sistemas distribuídos consistem de várias CPUs
  - diferentes maneiras de se organizar o hardware (interconexão e comunicação)
- Classificação
  - Multiprocessador (memória compartilhada)
  - Multicomputador

Organizações de processadores e memória



Conexão de multiprocessadores

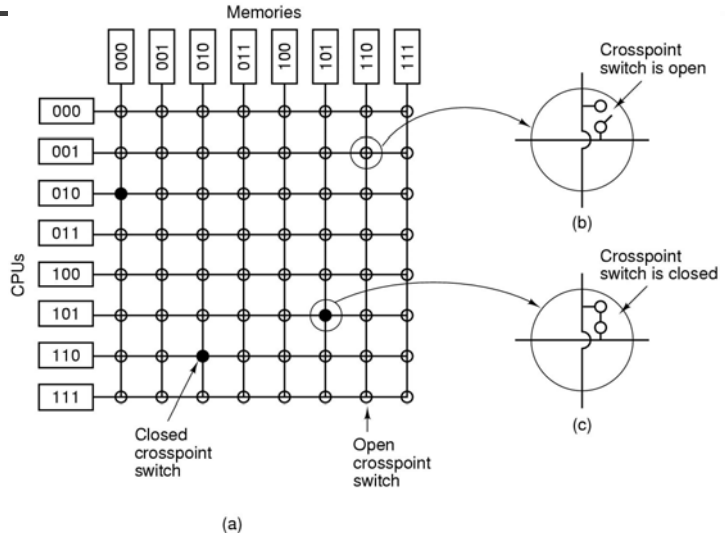
- Por Barramento
  - Uso de cache de memória para aumentar o desempenho
  - Problema de coerência de cache
    - Solução: Cache write through (com snooping cache)



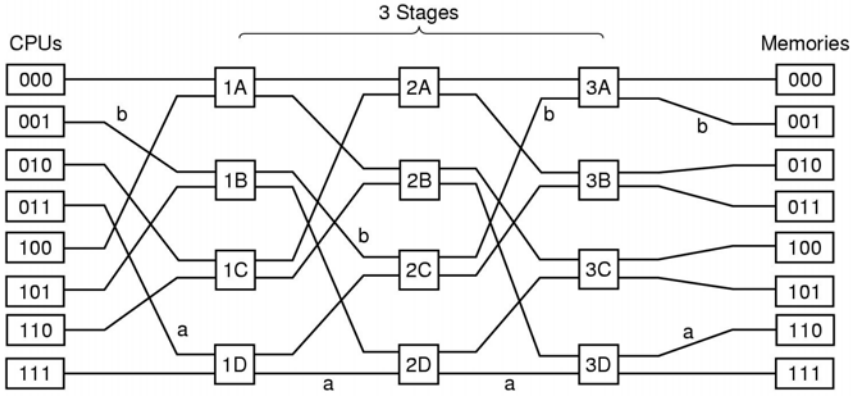
Conexão de multiprocessadores

- Por matrizes de comutação (switches)
  - Matriz de comutação memórias x CPUs
    - Matrizes com N^2 conexões
    - Uso elevado de chaves
  - Solução: uso de estágios de matrizes 2x2 (rede ômega)
    - $n*\log_2 n$ switches
    - Maior atraso

Conexão de multiprocessadores



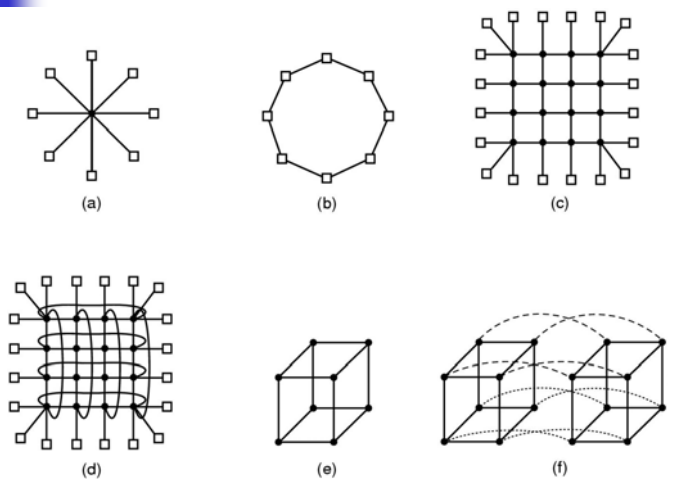
Conexão de multiprocessadores



Conexão de multicomputadores

- Barramento
  - Rede Local
- Switch
  - Grade (grid)
    - Número de hops = raiz quadrada do número de CPUs
  - Hipercubo
    - Número de Hops = Cresce de forma logaritma com o número de CPUs
    - Sistemas comerciais disponíveis com 16.384 CPUs

Conexão de multicomputadores



Sistemas operacionais para o hardware distribuído

- Classificação
  - fortemente acoplado
  - fracamente acoplado
- SOs
  - Sistemas operacionais de rede (fracamente acoplados)
  - Sistemas distribuídos reais
    - HW fracamente acoplado, SW fortemente acoplado

Questões de projeto

- Transparência
- Heterogeneidade
- Abertura
- Flexibilidade
- Confiabilidade
- Desempenho
- Escalabilidade

Transparência

- Objetivo
  - fornecer aos usuários uma imagem única e abstrata do sistema computacional
- Níveis de transparência
  - Nível de usuário: O usuário tem a impressão de estar usando um sistema centralizado.
  - Nível de programador: O programador tem a ilusão de programar um sistema centralizado.
    - Sintaxe e semântica das chamadas deve ser semelhante.

Tipos de Transparência

- Acesso
  - o acesso a recursos é idêntico
- Localização
  - os usuários não precisam conhecer a localização dos recursos
- Migração
  - os recursos podem se mover no sistema sem alterar seus nomes
- Replicação
  - os usuários não sabem quantas cópias de um recurso existem
- Concorrência
  - múltiplos usuários podem compartilhar um recurso sem o perceber (e sem conflitos)

Tipos de Transparência (cont)

- Falha
  - esconde a ocorrência de falhas
- Paralelismo
  - atividades podem ocorrer em paralelo sem que o usuário tenha de explicitá-las
- Desempenho
  - permite reconfiguração para aumentar desempenho
- Escala
  - permite expansão em escala sem alterações no sistema

Transparência de acesso

- Permite que objetos locais e remotos possam ser acessados de maneira idêntica
- Exemplo: login em uma máquina
  - Local: usuário e senha
  - Remoto: ssh ou telnet

Transparência de localização

- Os usuários não devem estar conscientes da localização física dos recursos
  - Por exemplo: o nome do recurso não deve conter o nome da máquina na qual o recurso reside
    - \\servidor1\shared\recibos.doc
    - http://www.puc-campinas.edu.br
- Os sistemas transparentes quanto à localização devem possuir um serviço de nomes, que mapeia o nome abstrato ao endereço do recurso.

Transparência de migração

- Os recursos podem trocar de lugar no sistema.
- Um sistema transparente quanto à migração é também transparente quanto à localização, mas também deve observar outras características de projeto.
- O que pode migrar ?
  - Dados
  - Computação
  - Processos
- Dependência residual
  - quando um componente do sistema migra, podem haver solicitações em andamento no sistema para ele, que não tomaram ainda conhecimento de sua nova localização. Neste caso, os nós podem guardar um histórico do movimento dos recursos, para que o processo que possua sua localização antiga (nome antigo) possa encontrá-lo.

Migração de dados

- Transferência de arquivos:
  - Quando um usuário necessita acessar um arquivo x, o arquivo x completo é transferido para a sua máquina local. Se houver alterações, o arquivo deve ser transferido de volta ao site origem
- Transferência de partes do arquivo: Somente as partes do arquivo que serão acessadas são realmente transferidas.

Migração de computação

- Quando se necessita de um grande volume de dados que se encontra em outra máquina, é mais eficiente transferir a computação do que transferir os dados.
- Migração de computação pode ser feita via RPC ou pelo envio de mensagens (geralmente no modelo cliente-servidor)

Migração de processos

- A migração de um processo, depois de iniciada a sua execução, pode ser justificada pelas seguintes razões:
  - Balanceamento de carga
  - Queda de uma máquina
  - Preferências de hardware
  - Preferências de software
  - Proximidade dos recursos
- Poucos sistemas implementam esse recurso
  - MOSIX

Transparência de replicação

- Por razões de desempenho, o sistema pode manter cópias de recursos em vários nós, sem que o usuário ou programador estejam conscientes deste fato
- Deve ser garantido pelo sistema que as múltiplas cópias do recurso serão sempre vistas como uma única cópia (coerência entre as cópias)

Transparência de concorrência

- Os usuários não devem notar que existem outros usuários no sistema. Se dois usuários acessam simultaneamente um mesmo recurso, o sistema deve garantir a coerência
- Em sistemas distribuídos, devem ser garantidas as mesmas condições de concorrência de um sistema centralizado

Transparência de falha

- Permite esconder as falhas de maneira que os usuários e programadores de aplicações possam completar suas tarefas apesar das falhas de componentes de hardware ou software
- Tarefas
  - Detecção
  - Reconfiguração
  - Recuperação

Transparência de paralelismo

- O próprio sistema operacional deve decidir que recursos (ex. processadores) alocar a uma aplicação distribuída de maneira que critérios de otimização sejam atendidos (balanceamento de carga, tempo de resposta etc).
- O usuário não deve interferir nessa escolha.
- O número de recursos alocados a uma aplicação pode variar de uma execução para outra.

Transparência de desempenho

- Permite que os sistemas possam ser reconfigurados para aumentar o desempenho com a variação da carga
- Ex.: uso de um algoritmo diferente para realização de uma determinada tarefa

Transparência de escala

- Permite que as aplicações e os sistemas pode ser expandidos sem mudanças na estrutura do sistema ou algoritmo da aplicação.
- Ex.: inclusão de novas máquinas na rede

Heterogeneidade

- Pode existir variedade e diferença em:
  - Redes de computadores
  - Hardware dos computadores
  - Sistemas operacionais
  - Linguagens de programação
  - Implementações por diferentes desenvolvedores
- Middleware: camada de software que fornece a abstração de programação e mascara a heterogeneidade. Ex.: CORBA, Java RMI
- Código móvel: código que pode ser enviado de um computador para outro. Ex. applets Java

Abertura

- O sistema pode ser estendido ou reimplementado de várias maneiras?
- A abertura de um sistema distribuído é determinada primariamente pelo facilidade de incorporação e disponibilização de novos serviços.
- Esta característica não pode ser obtida a menos que sejam tornadas públicas a especificação e documentação das interfaces dos componentes chave do sistema.
- Sistemas construídos em conformidade com padrões
  - Ex.: IETF (RFC), ANSI, ITU, IEEE,

Flexibilidade

- A inserção de novos módulos no sistema deve ser uma tarefa simples
- Duas abordagens para a estruturação de um sistema distribuído:
  - kernel monolítico (e.g. Unix distribuído)
  - micro-kernel (Mach, Chorus, Amoeba, etc)
- Um microkernel fornece somente serviços básicos
  - Mecanismo de comunicação entre processos - IPC
  - Gerência básica de memória
  - Gerência de processos de baixo nível (trocas de contexto)
  - Entrada e saída de baixo nível
- Os demais serviços (gerência de arquivos, escalonamento, etc) são providos por serviços em nível de usuário

Confiabilidade

- Em teoria
  - Se uma máquina falhar, outra pode assumir suas tarefas
  - Confiabilidade do grupo aumenta
- Na prática
  - Alguns componentes ou serviços são vitais para o sistema
  - Caso parem, todo o sistema pode cair
- Aspectos da confiabilidade
  - Disponibilidade
  - Segurança
  - Tolerância a falhas

Disponibilidade

- Fração de tempo em que o sistema está disponível para uso
- Alcançada através de:
  - redundância de componentes críticos
  - se um componente falhar, pode ser substituído
- Técnicas geralmente utilizadas:
  - redundância de hardware
    - Processadores, discos
  - redundância de software
    - dois programas distintos efetuando a mesma função

Segurança

- Autenticidade
  - Os usuários comprovam suas identidades (senhas, chaves etc)
- Autorização
  - Estabelecimento de controles de acesso aos recursos (listas de controle de acesso)
- Privacidade
  - As informações somente podem ser lidas por quem tiver direito (mecanismos de criptografia)
- Integridade
  - Os dados não podem ser destruídos ou corrompidos por terceiros
- Não-repudio
  - Todas as ações podem ser imputadas a seus autores (mecanismos de auditoria)
- Disponibilidade
  - Serviços não podem ser desativados por ação de terceiros (DoS)

Tolerância a falhas

- O que fazer em caso de falha de um servidor?
  - Sistemas distribuídos podem ser projetados para mascarar falhas
- Técnicas para tratar falhas:
  - Detecção de falhas: checksum
  - Mascaramento de falhas: retransmissão de mensagens, arquivos em duplicidade
  - Tolerância a falhas: temporizador com notificação de usuário
  - Recuperação de falhas: transações
  - Redundância: roteadores, DNS, banco de dados etc

Faltas, erros e falhas

- Faltas
  - Situações incorretas no estado interno de um sistema
  - Ex: um bit de memória inválido, um cabo de rede rompido
- Erro
  - Decorrência da falta
  - Estado interno incorreto do software
  - Ex: queda de uma conexão TCP, variável com valor errado
- Falha
  - Decorrência do erro
  - Serviço oferecido ao usuário não cumpre sua especificação
  - Ex: banco de dados fora do ar, aplicação mostrando dados incorretos
- Portanto: FALTAS → ERROS → FALHAS

Desempenho

- Métricas para medir desempenho:
  - Tempo de resposta
  - *Throughput* (número de tarefas / tempo)
  - Utilização do sistema
  - Uso da capacidade da rede
- Em um sistema distribuído:
  - + processadores, + memória, + capacidade de armazenamento
  - Pode-se distribuir os processos entre os processadores
  - + velocidade final de computação ?
  - + Custo de comunicação !

Custo de comunicação

- Componentes do custo de comunicação:
  - Tempo de processamento do protocolo
  - Tempo de latência do hardware e software de rede
  - Tempo de transmissão da mensagem
- Para obter um bom desempenho:
  - Reduzir a comunicação entre os processadores
  - Buscar manter um bom nível de paralelismo
  - Encontrar um ponto de equilíbrio entre ambos !

Granularidade das tarefas

- Granularidade
  - Tamanho do elemento básico que será distribuído
- Fina
  - pequenos conjuntos de instruções executados em paralelo
  - Muita comunicação → desempenho ruim
- Média
  - Funções executadas em paralelo (RPC)
- Grossa
  - Processos executados em paralelo
  - Grande quantidade de código para cada processo
  - Pouca comunicação → ótimo desempenho

Escalabilidade

- Noção intuitiva
  - Um sistema distribuído que opera bem com 10 máquinas também deve funcionar bem com 10.000 máquinas
  - O desempenho do sistema não deve ser degradado na medida que o número de nós cresce.
- Inimigos da escalabilidade:
  - Componentes centralizados (por exemplo, um único servidor de e-mail para todos os usuários)
  - Tabelas centralizadas (por exemplo, uma única relação on-line de telefones)
  - Algoritmos centralizados (por exemplo, o roteamento de mensagens baseado em informações completas de caminho)

Níveis de escalabilidade

- Escalabilidade de Arquitetura
  - Escalabilidade de uma arquitetura mede a parte de paralelismo inerente à aplicação que pode ser realizada sobre a arquitetura.
  - O tempo de execução do algoritmo é limitado por suas próprias características e não por características da arquitetura.
- Escalabilidade do Sistema Operacional
  - Um sistema operacional escalável também não deve limitar o desempenho de uma aplicação.
  - Adicionar processadores não vai diminuir o tempo de resposta das chamadas ao sistema, porque nós estamos introduzindo mais recursos a gerenciar.
- Linguagem de programação
  - Que permitam o uso de recursos não centralizados de forma simples
  - Exemplo: tabelas e hashes distribuídos
- Aplicação
  - Algoritmos baseados em informações descentralizadas

Melhorando a escalabilidade

- Algoritmos descentralizados com as seguintes características
  - Nenhuma máquina possui informações completas sobre o estado do sistema
  - Máquinas tomam decisões baseadas apenas nas informações disponíveis localmente
  - Falha de uma das máquinas não impede o funcionamento do algoritmo
  - Não existe um relógio global implícito
- Sistemas escaláveis
  - Servidores distribuídos: vários servidores cooperam para a execução de um serviço
  - Estruturas de dados distribuídas, divididas em partes e armazenadas em vários locais do sistema
  - Algoritmos distribuídos: cada servidor executa uma parte do algoritmo

Bibliografia

- Sistemas operacionais modernos
  A.S. TANENBAUM
  Prentice-Hall, 1995
- Distributed Systems: concept and design
  G. Coulouris, J. Dollimore e T. Kindberg
  3.ed., Addison-Wesley, 2001
- Sistemas Operacionais
  H.M. Deitel, P.J. Deitel e D.R. Choffnes
  3. Ed., Pearson Education, 2005
- Sistemas Operacionais: Conceitos
  A. Silberschatz e P. Galvin
  5a. Ed., Prentice-Hall, 2000

Exercícios

1. Cite 2 vantagens e 2 desvantagens dos sistemas distribuídos em relação aos centralizados.
2. Cite 5 tipos de recursos de hardware e 5 tipos de recursos de software que podem ser compartilhados. Dê exemplos de seu compartilhamento em sistemas distribuídos.
3. Um programa servidor escrito em uma determinada linguagem (Ex.: C++) provê a implementação de um objeto OBJ que deve ser acessado por clientes escritos em linguagens diferentes (Ex: Java). Os computadores clientes e servidor possuem diferentes arquiteturas de Hardware, e estão todos conectados a Internet. Descreva os problemas devido a cada um dos 5 aspectos de heterogeneidade (slide 23) que precisam ser resolvidos para permitir que um objeto cliente invoque um método no objeto servidor.

Exercícios

4. Suponha que a operação do objeto OBJ é dividida em 2 categorias: operações públicas disponíveis a todos usuários e protegidas disponíveis apenas a determinados usuários. Discuta os problemas envolvidos em garantir que somente os usuários determinados possam usar as operações protegidas. Suponha que o acesso às operações protegidas fornece informações que não podem ser reveladas a todos usuários, que novos problemas surgem? Defina "transparência".
5. Explique os tipos de transparência estudados.
6. É sempre importante saber se as mensagens enviadas chegaram ao seu destino de forma segura? Se sua resposta for "sim" , explique o por quê. Se a resposta for "não", dê exemplos apropriados.

Exercícios

7. Considere um sistema distribuído com duas máquinas, A e B. Determine se a máquina A pode distinguir as seguintes situações:
   a. A máquina B pára de funcionar
   b. Ocorre um defeito na conexão entre A e B
   c. A máquina B está sobre carregada e seu tempo de resposta é cem vezes maior que o normal

   Quais as implicações das suas respostas para a recuperação de falhas em um sistema distribuído?