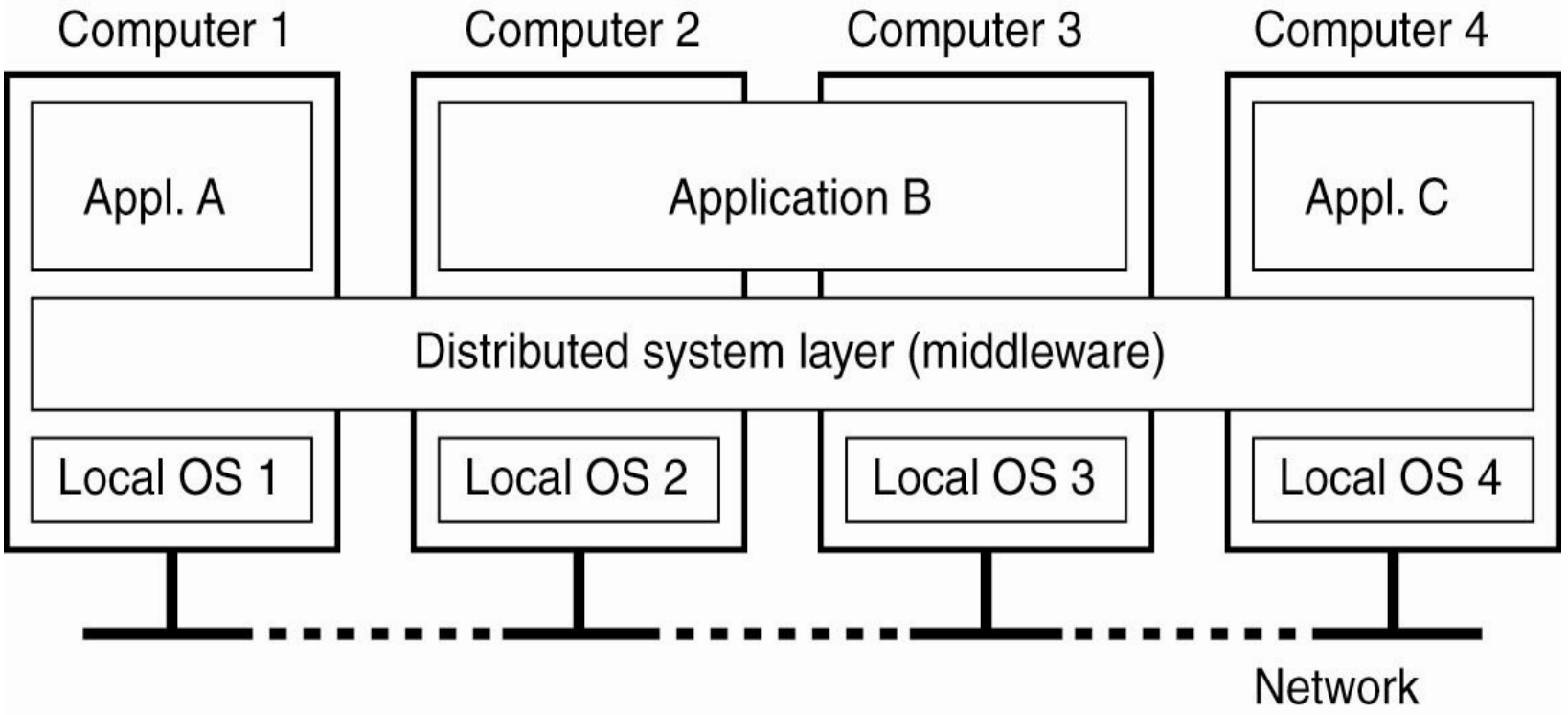
Introdução

Capítulo 1

Sistemas Distribuídos : Definição (1)

- “Sistema distribuído consiste de uma coleção de computadores independentes ligados por algum tipo de meio de comunicação e equipados com software de sistema distribuído.”
- Software distribuído permite computadores coordenar suas atividades e compartilhar recursos do sistema - hardware, software e dados.
- Características principais:
  - Vários computadores (ou processadores)
  - Redes (ou elemento de interconexão)
  - Estado compartilhado

Sistemas Distribuídos : Definição (2)



Problema fundamental: heterogeneidade

Solução: Middleware

Esconder as diferenças entre diferentes plataformas

Porque Sistemas Distribuídos

Pergunta: Porque construir e usar sistemas distribuídos?
Algumas justificativas fundamentais:

- Acessibilidade de Recursos
- Transparência
- Abertura
- Escalabilidade

Acessibilidade

- Principal objetivo:dar acesso e compartilhar recursos remotos de uma forma controlada e eficiente
- Tipos de recursos:
  - Impressoras, computadores, arquivos, dados
- Conectar utilizadores e recursos também facilita interações e trabalho cooperativo
- Aspecto complicador é a segurança
  - Abuso de recursos compartilhados
  - Roubo de informações críticas
  - Questões de privacidade

Sistemas Distribuídos - Introdução

Transparência

- Como conseguir a imagem de um sistema único? Esta é, provavelmente, a característica mais importante e permite que o sistema seja visto como um todo e não uma coleção de componentes independentes.

Transparência

Transparência	Descrição
Acesso	Esconde diferenças na representação de dados e como um recurso é acessado
Localização	Esconde onde um recurso está localizado
Migração	Esconde que um recurso pode mover-se para outra localização
Relocação	Esconde que um recurso pode ser movido para outra localização enquanto esta sendo usado
Replicação	Esconde que um recurso pode ser compartilhado por vários usuários concorrentes
Concorrência	Esconde que um recurso pode ser compartilhado por vários usuários concorrentes
Falha	Esconde a falha e recuperação de um recurso
Persistência	Esconde quando um recurso (software) esta em memória ou em disco

Exemplos:
- o nome do recurso (impressora) não inclui a sua localização
- replicar os arquivos mais acessados em vários servidores de arquivos
- acessos concorrentes a um arquivo são automaticamente protegidos por locks

Transparência

- Grau de transparência, Quanta transparência é preciso?
- Em geral é desejável.
- Em algumas situações não tem sentido:
    - Não é possível
    - Não é uma boa idéia
    - Tem consequências no desempenho

Abertura(Openess)

- Definição : um sistema é aberto se existem regras bem definidas que especificam a sintaxe e a semântica dos serviços por ele fornecidos
- dar acesso e compartilhar recursos remotos de uma forma controlada e eficiente
- Este objetivo é alcançado através de interfaces padronizadas:
  - Acordo para o formato das msgs e dados transportados
  - Biblioteca exporta um conjunto de funções bem definidas
  - Interface de um objeto descrita em IDL
- Conceitos relacionados:
  - Interoperabilidade: funcionamento em conjunto de componentes de fabricantes diferente
  - Portabilidade: funcionamento de componente sem modificação em outro sistema
  - Extensibilidade: funcionalidades de componentes podem ser trocadas ou melhoradas sem modificar o sistema como um todo

Abertura(Openess)

- Separação entre Políticas e Mecanismos
  - Tem a ver com a flexibilidade e extensibilidade
  - O sistema deve fornecer uma série de mecanismos
    - Ex. : uma série de réplicas de um serviço
  - Mas as políticas para o seu uso devem ser configuráveis
    - Ex. : qual consistência manter, fraca ou forte?
- Exemplo: Cache do Navegador Web
  - Como é:
    - Podemos escolher o tamanho da cache
    - Podemos escolher qdo um doc tem a sua consistência verificada
  - Poderia ser:
    - O navegador nos dá um mecanismo de armazenamento para cache
    - Definimos a política:
      - Tamanho, verificação de consistência, quanto tempo um doc fica na cache, que documentos serão apagados, que tipo vai para cache,
- Esta politica poderia ser descrita em um arquivo de configuração, ou através de um componente de software (escrito) e juntado ao navegador.

Escalabilidade

- Um sistema distribuído que funciona bem com 100 nodos vai funcionar bem com 100000?
- Escalabilidade em 3 dimensões
    - Tamanho do sistema
    - Distribuição geográfica
    - Em termos de administração

Problemas

Concept	Example
Centralized services	A single server for all users um servidor único de e-mail para todos
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

- Evitar centralização
- Características:
    1. nenhuma máquina tem informação completa sobre o sistema
    2. as decisões são tomadas tendo em conta apenas a informação local
    3. a falha de uma máquina não deve impedir o funcionamento do algoritmo
    4. não se baseiam na hipótese da existência de relógios sincronizados

Tecnicas de Escalabilidade (1)

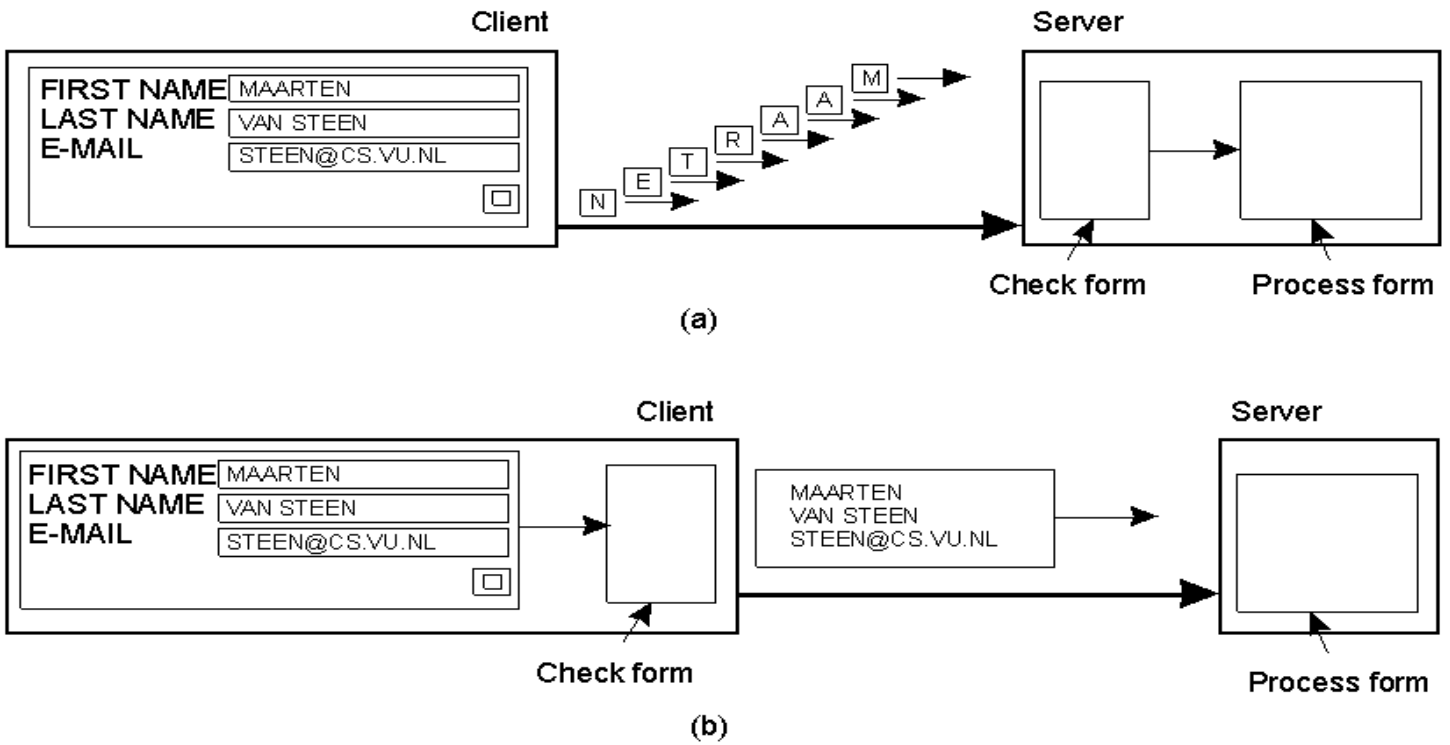
a) Esconder latência
   - Evitar “travar” o programa a espera de resposta
   - Evitar a comunicação o máximo possível

b) Distribuir

   - Dividir um componente em diversos menores e espalhar pelo sistema

c) Replicar
   - Balanceamento de carga
   - Replicar um sistema em vários locais
   - Replicar = problema?
   - Protocolos de replicação

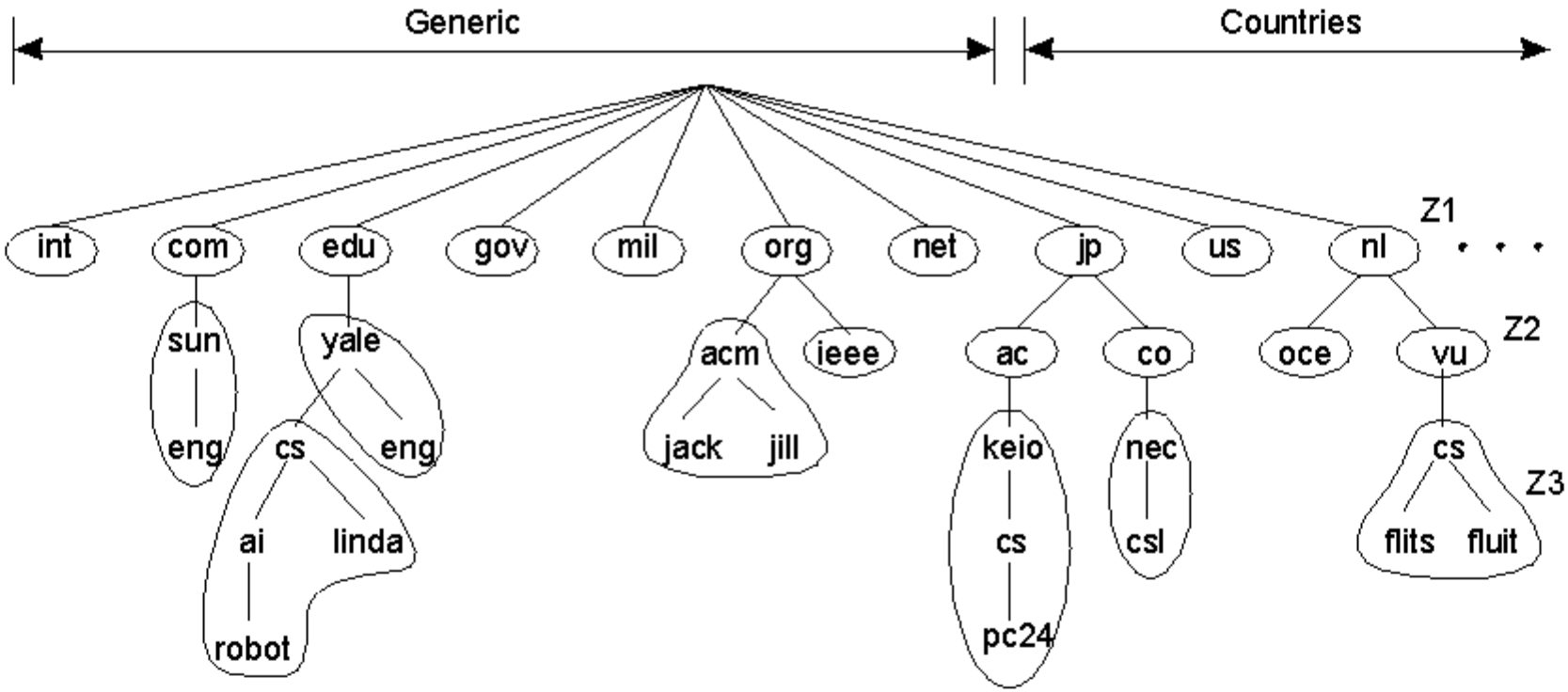
Tecnicas de Escalabilidade (2)



a) Servidor ou

b) Cliente verifica preenchimento de formulário

Tecnicas de Escalabilidade (3)



An example of dividing the DNS name space into zones.

Idéias erradas

- Segundo Peter Deutsch, ex-Sun Microsystems:
  - A rede é confiável
  - A rede é segura
  - A rede é homogenea
  - A topologia da rede não muda
  - Latência é zero
  - Largura de banda é infinita
  - Transporte custa zero
  - Existe apenas um administrador

Continua ...

CONCEITOS DE HARDWARE

- Taxonomia
- Arquitetura MIMD
  - Multiprocessadores
  - Multicomputadores

Sistemas Distribuídos - Introdução

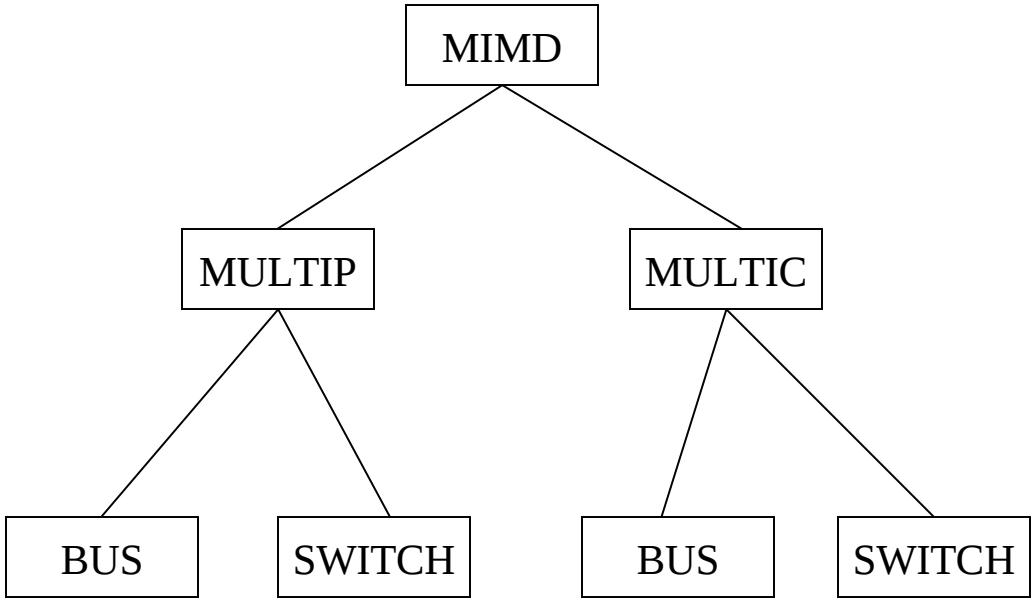
Conceitos de Hardware

- Taxonomia de Flynn (1972):
  - SISD: 1 fluxo de instruções e 1 fluxo de dados
  - SIMD: 1 fluxo de instruções e múltiplos fluxos de dados
  - MISD: múltiplos fluxos de instruções e 1 fluxo de dados
  - MIMD: múltiplos fluxos de instruções e múltiplos fluxos de dados

*Todos os sistemas distribuídos são MIMD

Sistemas Distribuídos - Introdução

Conceitos de Hardware - arquiteturas MIMD



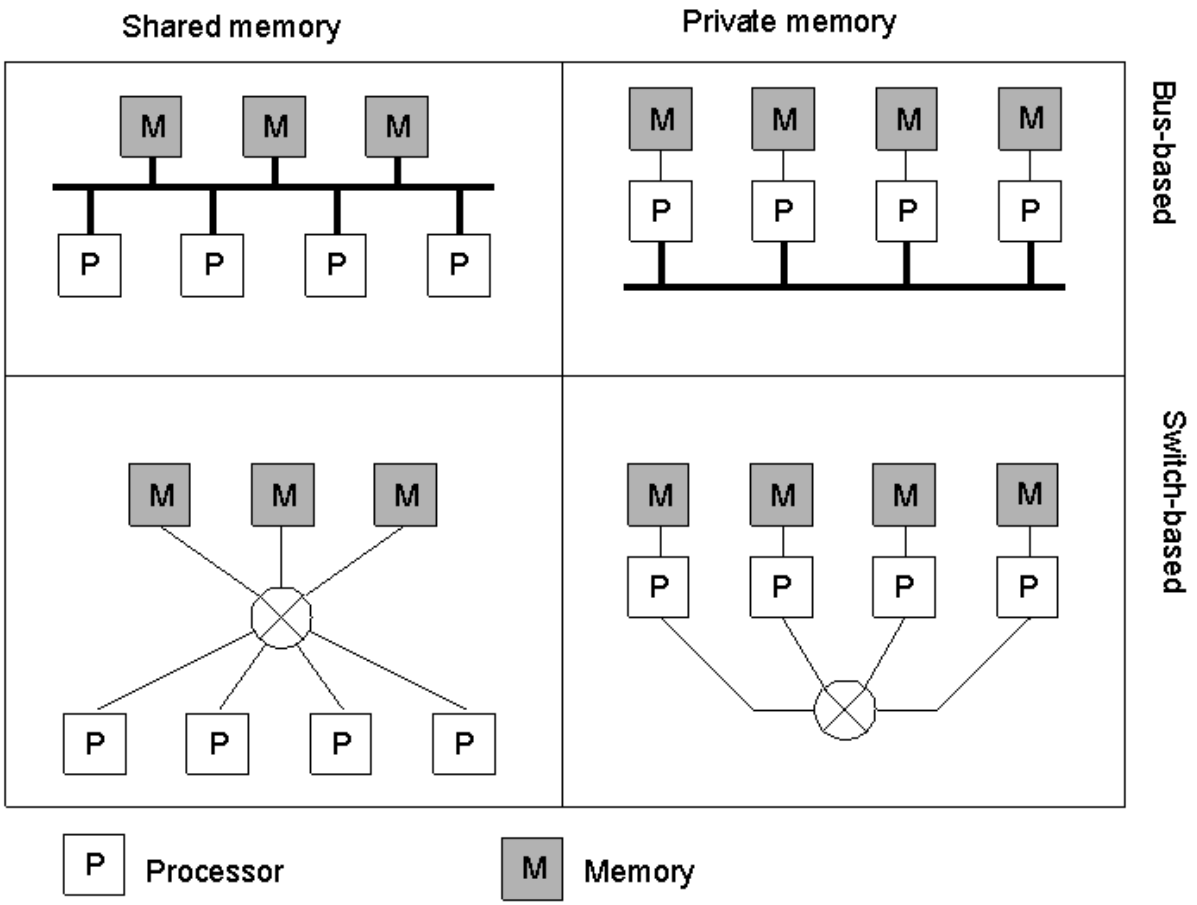
Classificação:

Memória --- multiprocessadores, multicomputadores

Rede de Interconexão --- Bus, Switch

Acoplamento --- Forte, fraco

Hardware Concepts

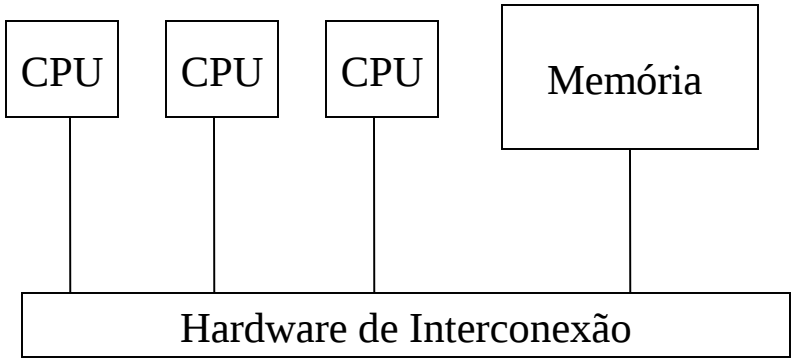


Diferentes basic organizations and memories in distributed computer systems

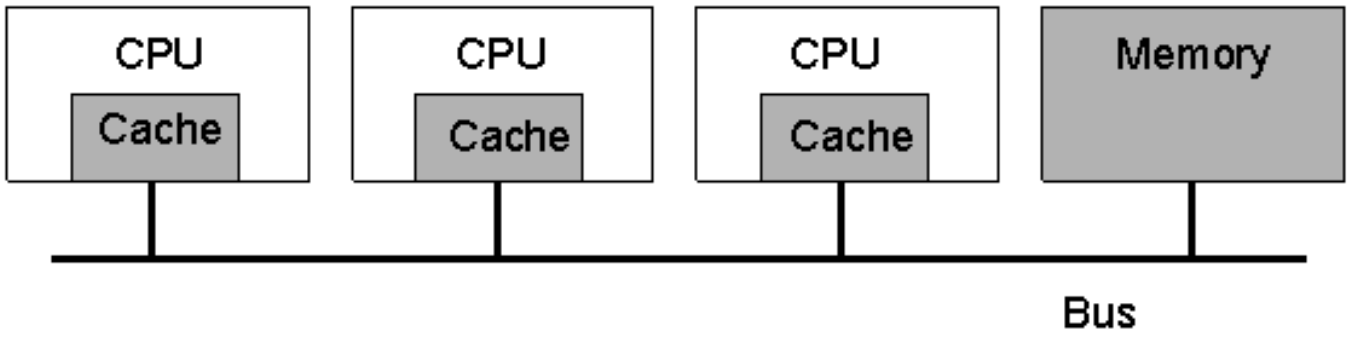
Sistemas Distribuídos - Introdução

Conceitos de Hardware - arquiteturas MIMD

- Multiprocessadores : fortemente acoplados
  memória compartilhada
- Arquitetura:

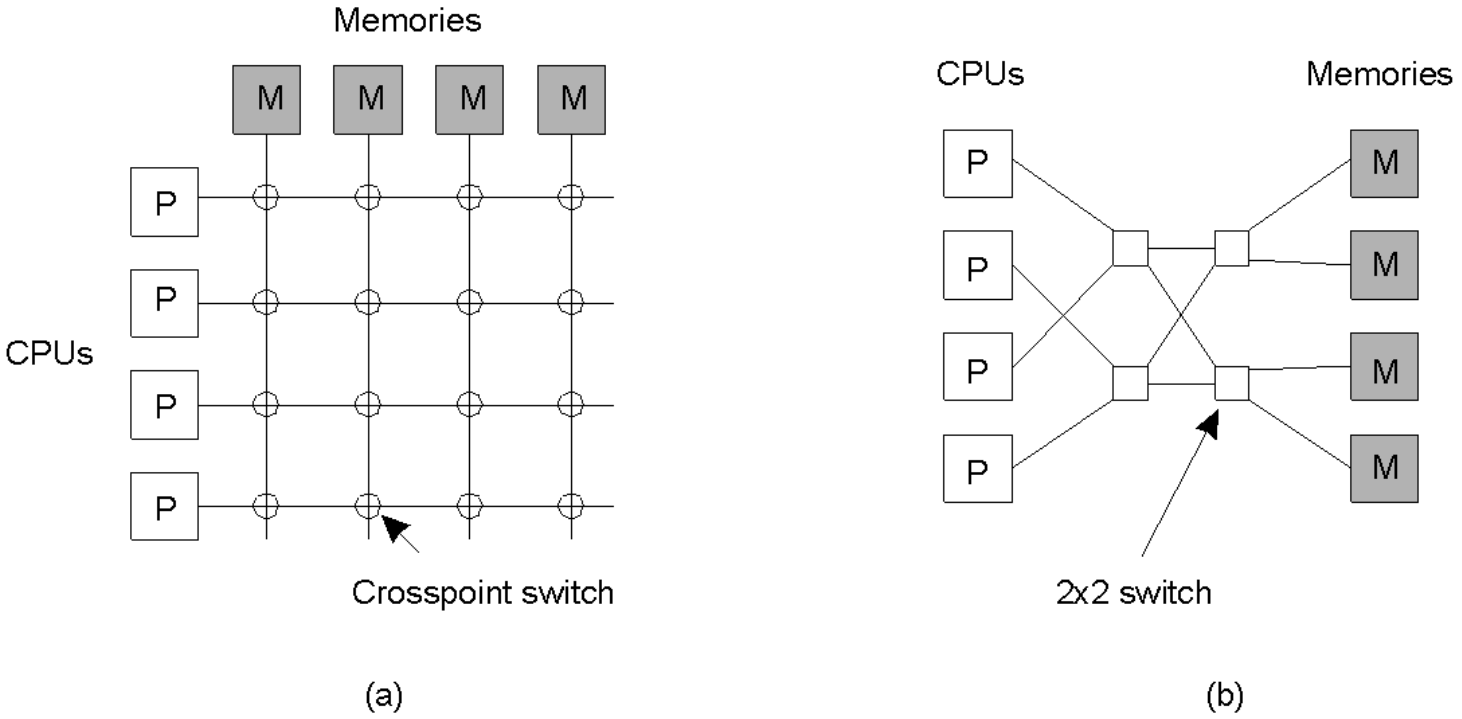


Multiprocessors (1)



A bus-based multiprocessor.

Multiprocessors (2)



a) A crossbar switch
b) An omega switching network

Sistemas Distribuídos - Introdução

Conceitos de Hardware - arquiteturas MIMD

Classificação baseada no acesso a memória

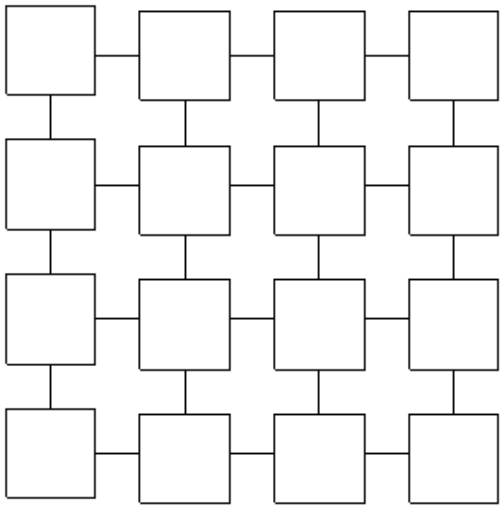
- **UMA -** Acesso uniforme a memória : a memória principal esta localizada em posição centralizada.
- **NUMA -** Acesso não-uniforme a memória : a memória principal é particionada e as partes são anexadas aos processadores. Um processador pode acessar a memória de qualquer processador (acesso é mais lento).

Sistemas Distribuídos - Introdução

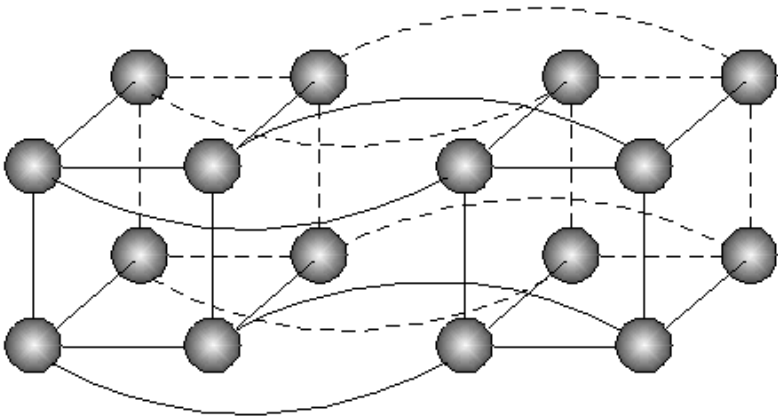
Conceitos de Hardware - arquiteturas MIMD

- Multicomputadores : fracamente acoplados
  memória distribuída
- Arquitetura:
  - Interconexão por LAN: Barramento, Anel, ...
  - Interconexão por redes chaveadas:
    - Estáticas - Hipercubo
    - Dinâmicas - Crossbar

Homogeneous Multicomputer Systems



(a)



(b)

a) Grid
b) Hypercube

Sistemas Distribuídos - Introdução

Conceitos de Software

- A meta de projeto de software na construção de um sistema distribuído é criar uma única imagem do sistema, ter uma coleção de computadores independentes como um único sistema, passada para os usuários.
- Software usado em SD podem ser classificados em 2 tipos:

  Fracamente acoplado – operam como máquinas praticamente independentes, existe compartilhamento de dispositivos e serviços (file, web)

  Fortemente acoplado – existe uma dependência forte com as outras máquinas para todos os aspectos do sistema. A interconexão e a funcionalidade são necessárias para a operação local do sistema.

Sistemas Distribuídos - Introdução

Conceitos de Software

- Sistemas distribuídos comum hoje aqueles com software fracamente acoplados e hardware fracamente acoplados. Ex. Estações de trabalho (CPU e SO local) em LAN.
  - Interação explícita *rlogin*
  - Servidores de arquivos atendem requisições
  - Alto grau de autonomia

  Poucos requisitos atingem o sistema como um todo

Sistemas Distribuídos - Introdução

Conceitos de Software

- Próximo passo: software fortemente acoplados em hardware fracamente acoplados. Ex. Rede de máquinas aparentando um único sistema timesharing, onde os usuários não estão preocupados com a distribuição.
- Neste caso, um verdadeiro sistema distribuído:
  - Um único mecanismo (global) de comunicação entre processos
  - Um esquema de proteção global
  - Esquema de nomes uniforme

  Mesma Interface de chamada de sistema em qualquer lugar

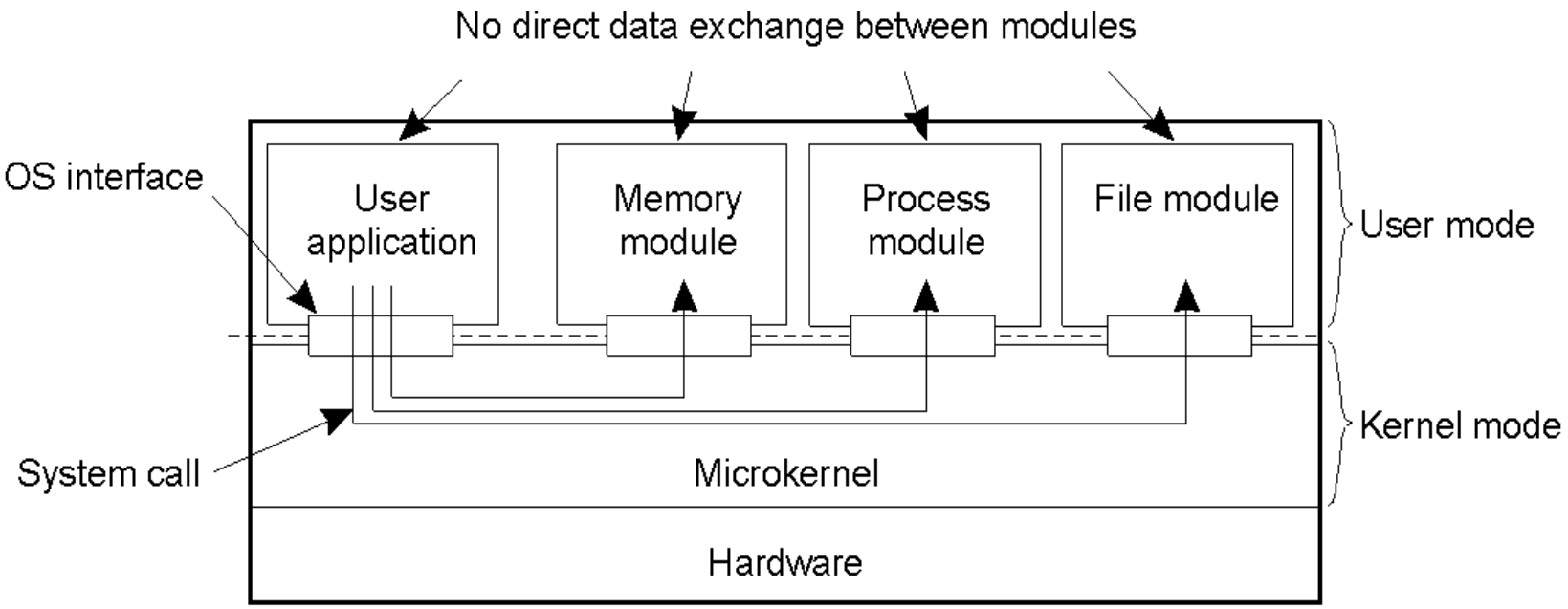
Software Concepts

System	Description	Main Goal
DOS	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

An overview between

- DOS (Distributed Operating Systems)
- NOS (Network Operating Systems)
- Middleware

Uniprocessor Operating Systems



Separating applications from operating system code through a microkernel.

Multiprocessor Operating Systems (1)

```
monitor Counter {
private:
 int count = 0;
public:
  int value() { return count;}
  void incr () { count = count + 1;}
  void decr() { count = count – 1;}
}
```

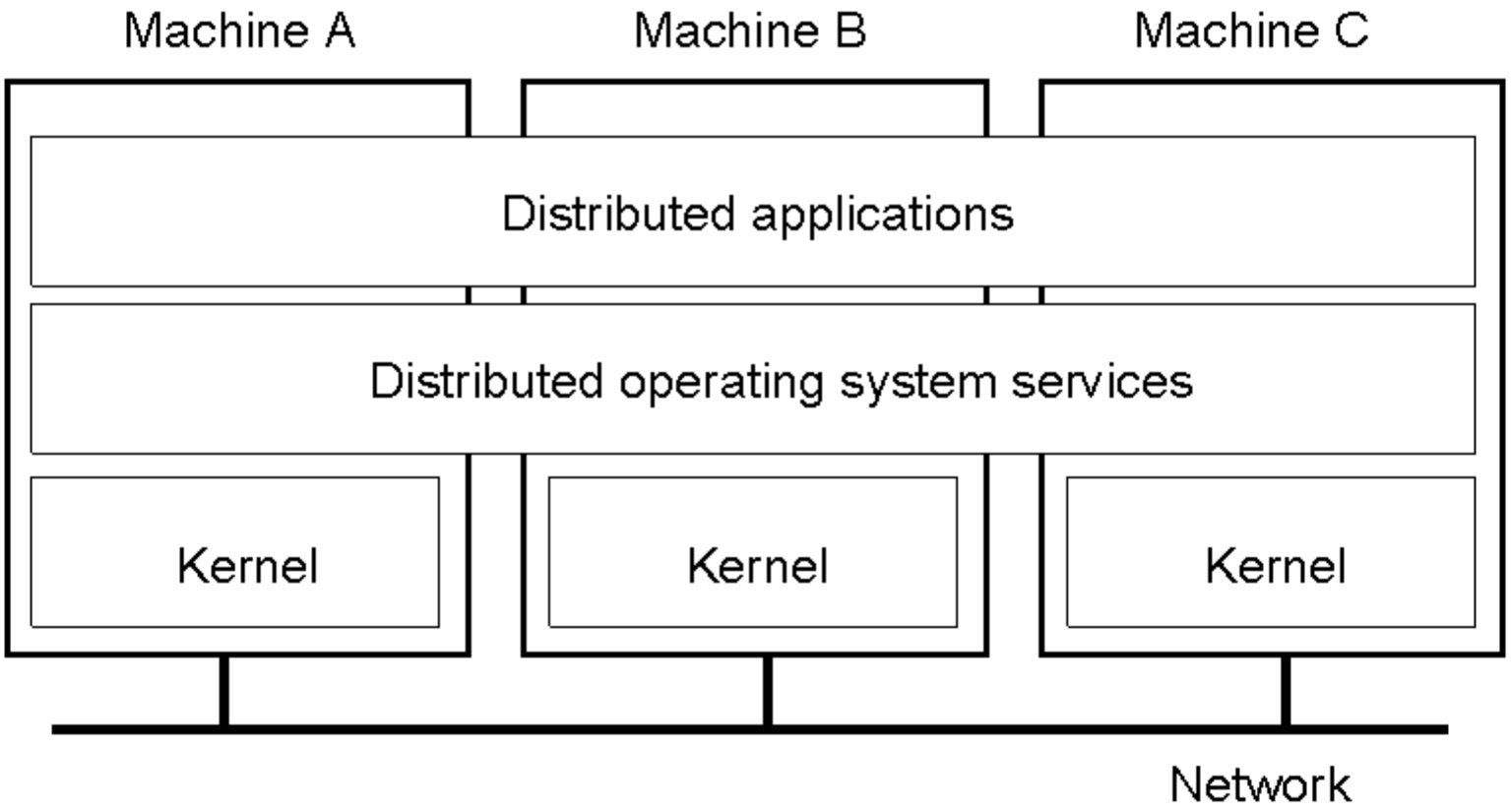
A monitor to protect an integer against concurrent access.

Multiprocessor Operating Systems (2)

```
monitor Counter {
private:
  int count = 0;
  int blocked_procs = 0;
  condition unblocked;
public:
  int value () { return count;}
  void incr () {
       if (blocked_procs == 0)
         count = count + 1;
       else
         signal (unblocked);
  }
void decr() {
if (count ==0) {
  blocked_procs = blocked_procs + 1;
  wait (unblocked);
  blocked_procs = blocked_procs – 1;
   }
  else
   count = count – 1;
 }
}
```

A monitor to protect an integer against concurrent access, but blocking a process.

Multicomputer Operating Systems (1)



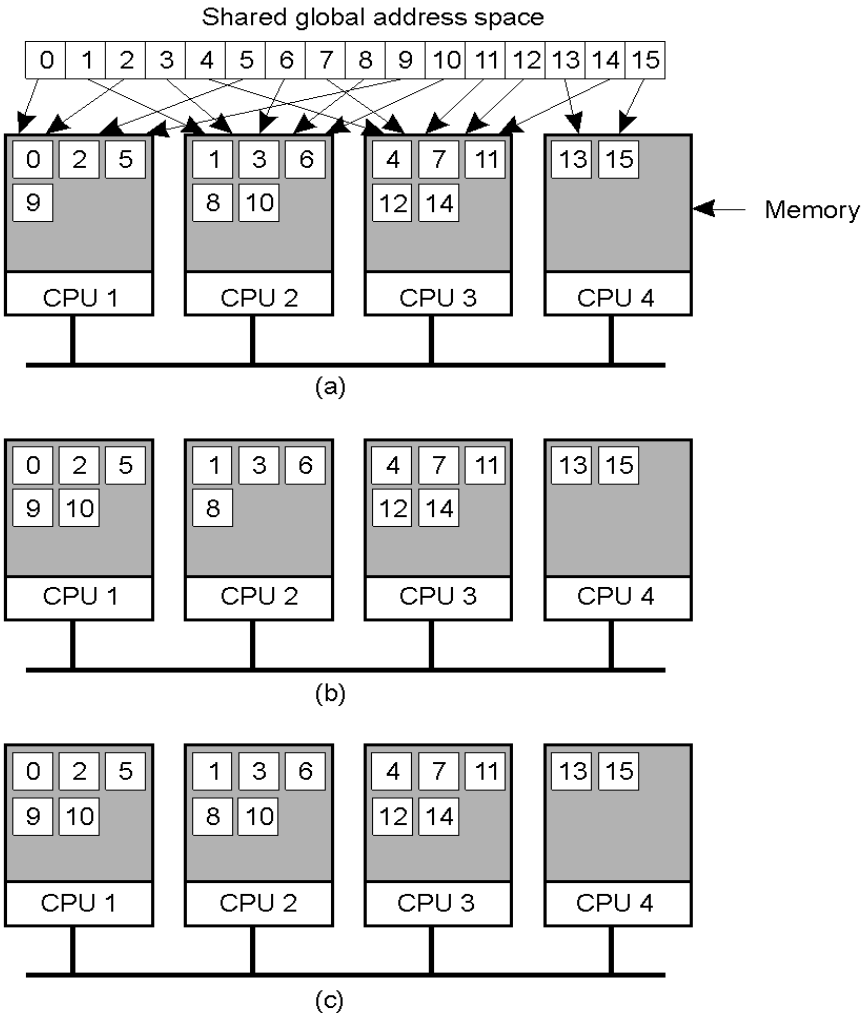
General structure of a multicomputer operating system

Distributed Shared Memory Systems (1)

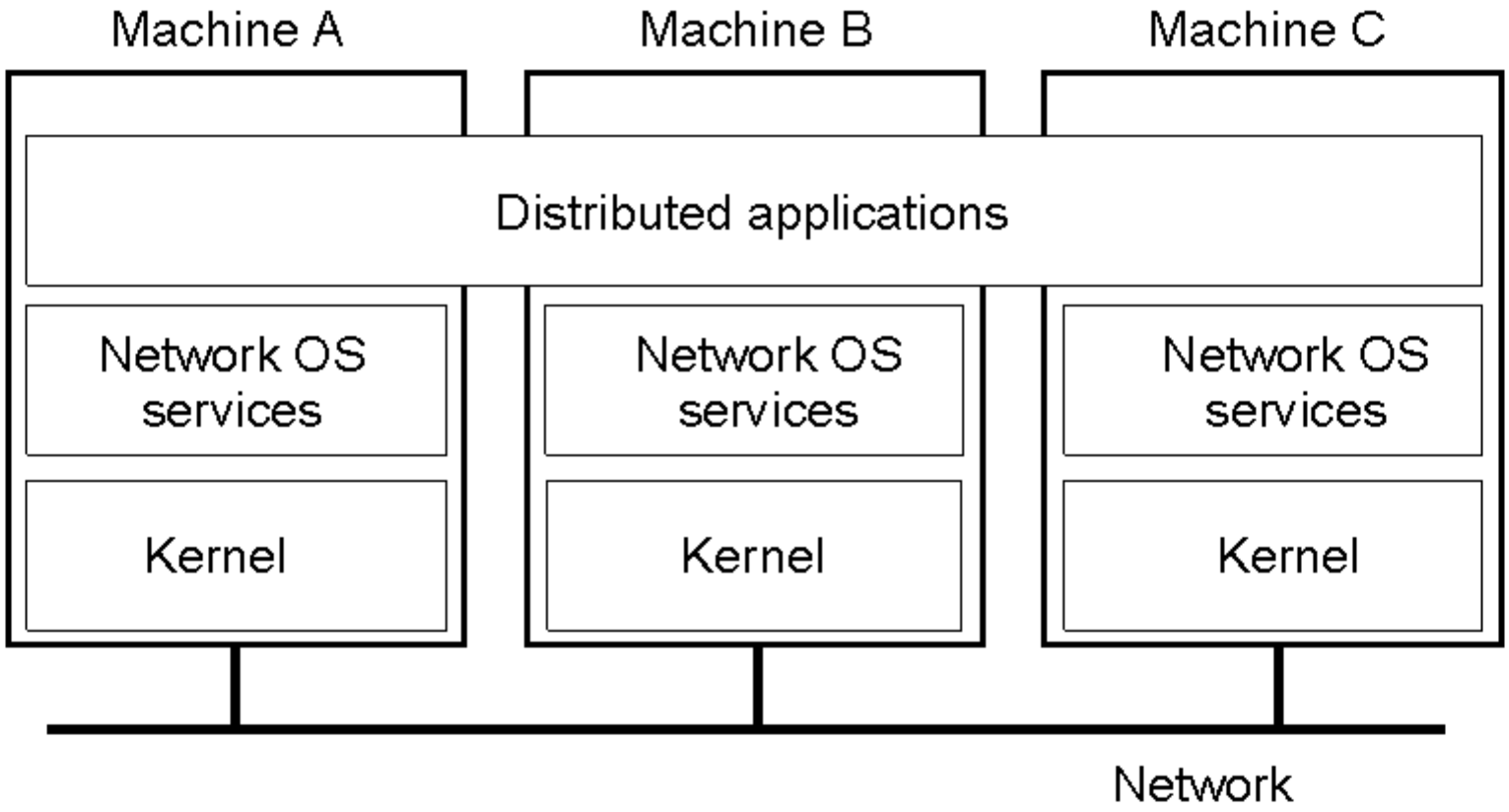
a) Pages of address space distributed among four machines

b) Situation after CPU 1 references page 10

c) Situation if page 10 is read only and replication is used

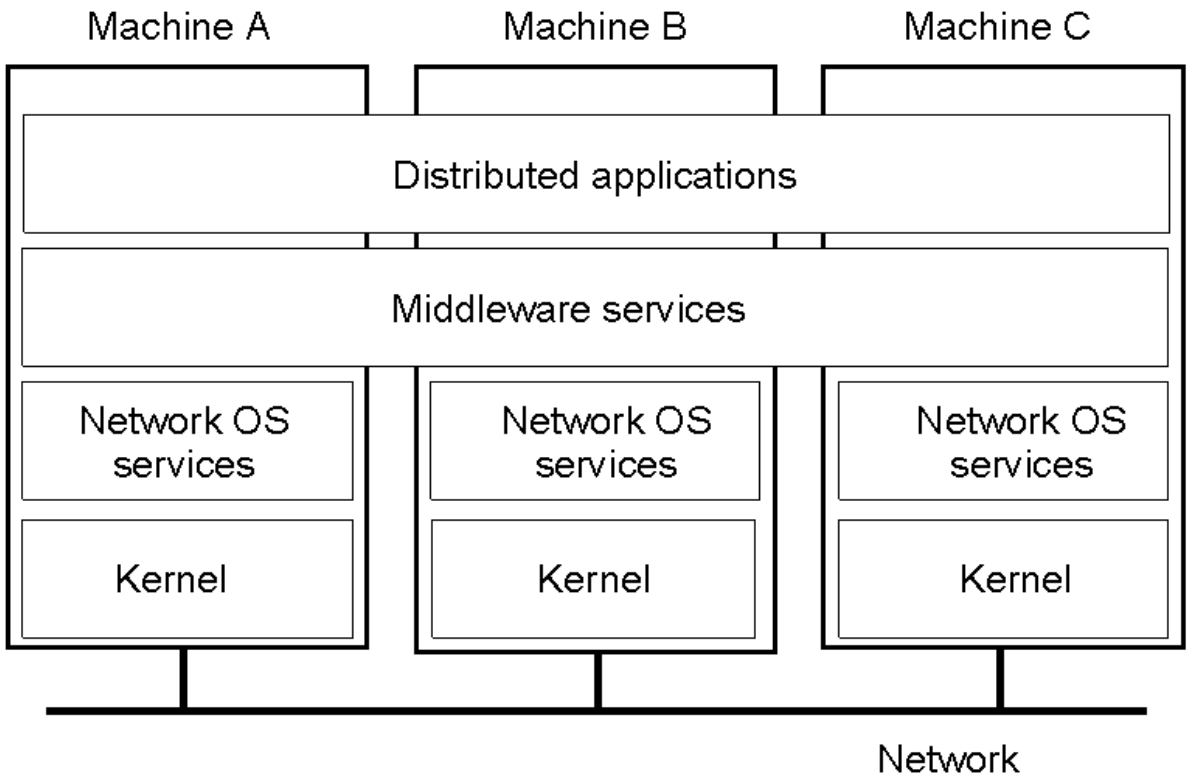


Network Operating System (1)



General structure of a network operating system.

Positioning Middleware



General structure of a distributed system as middleware.

Comparison between Systems

Item	Distributed OS		Network OS	Middleware -based OS
	Multiproc.	Multicomp.		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware based distributed systems.

Modelos de Serviço

- Cada unidade em um sistema distribuído pode ser responsável por um conjunto de funções na organização. O modelo de serviço pode ser considerado como uma classificação da configuração do sistema:
  - Modelo Centralizado
  - Modelo Cliente - Servidor
  - Modelo Pares (Peer-to-peer)
  - Modelo Pool de processadores

Modelo Centralizado

- Um modelo centralizado não existe rede. Todos os aspectos da aplicação estão hospedados em uma máquina e os usuários estão diretamente conectados a esta máquina.
  - Mainframe time-sharing
  - Pode ter várias CPUs
  - Comunicação via terminais (serial)
  - Problema: escalabilidade, contenção.

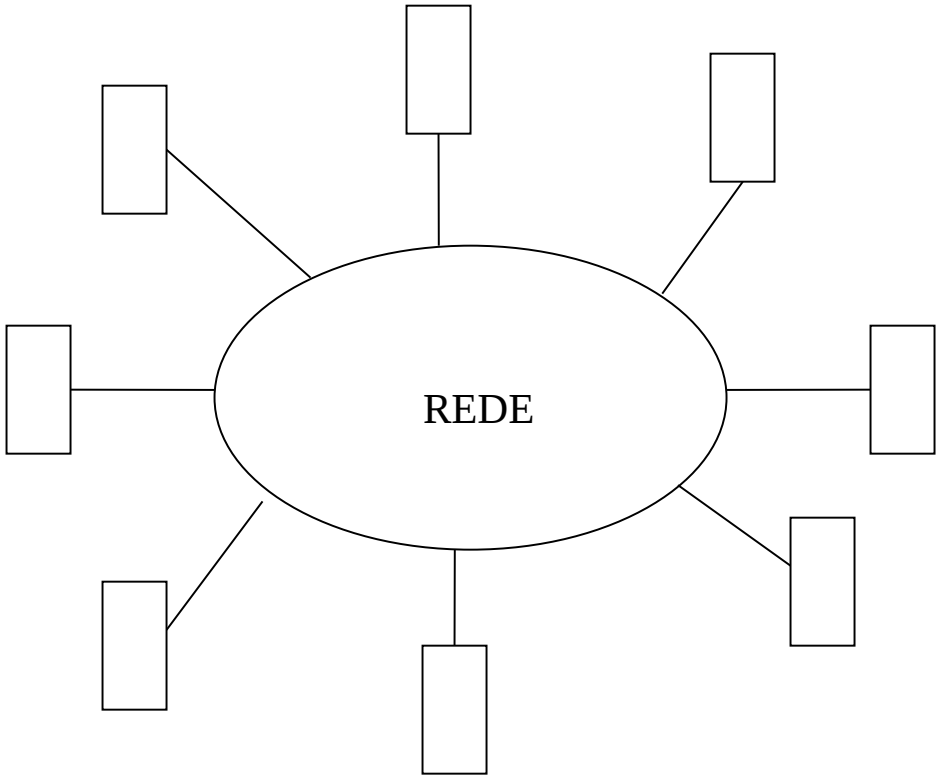
Modelo Cliente-Servidor

É o modelo de rede com 3 componentes:

Serviço: tarefa que é realizada por uma máquina particular (file).

Servidor: é a máquina que realiza a tarefa(file server)

Cliente: é uma máquina que solicita serviço.



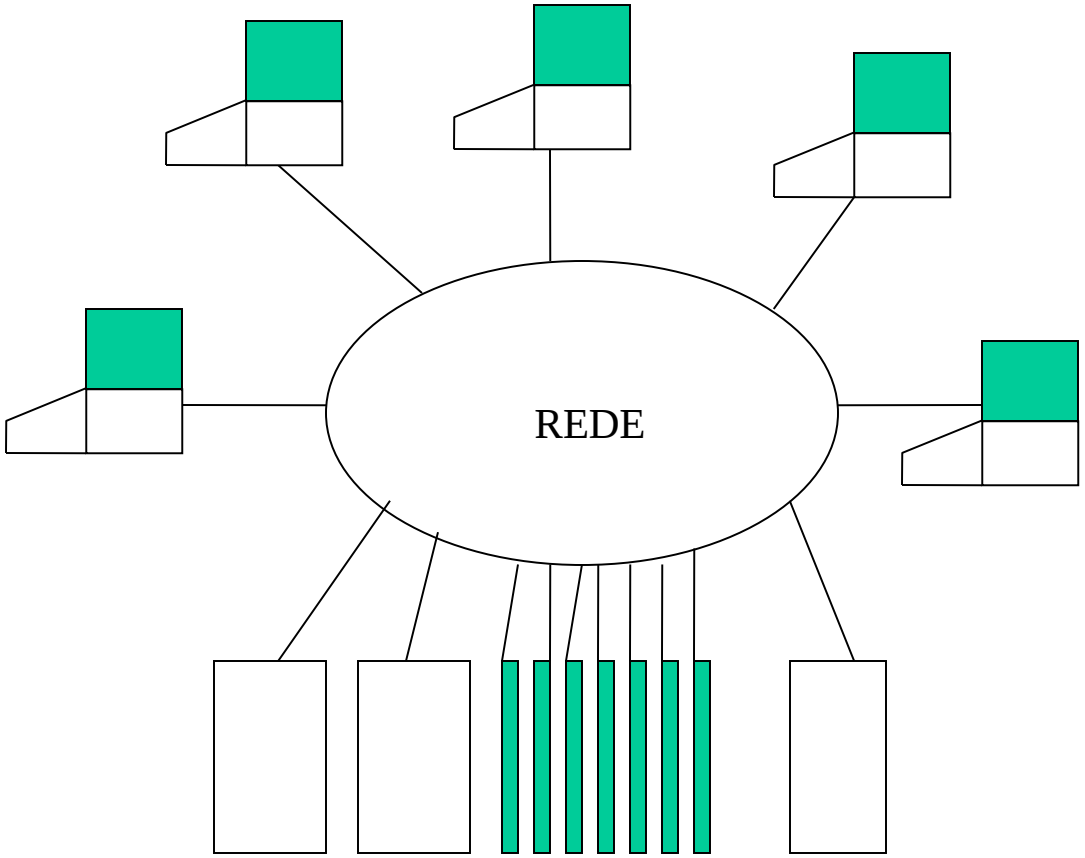
Modelo Peer-to-peer

O modelo cliente servidor certas máquinas são mais adequadas para realizar certos serviços. O modelo peer-to-peer cada máquina tem as mesmas capacidades (equivalentes), nenhuma é dedicada a servir as outras.

Sistemas Distribuídos - Introdução

Modelos de Sistemas Distribuídos

Modelo Grupo de processadores(Processor-Pool): é composto por um conjunto de PCs e workstations agrupados.



Sistemas Distribuídos - Introdução

Modelos de Sistemas Distribuídos

Terminais (X, diskless W)

Processadores (CPU, memória)

Servidores especiais (Run,FS)

- Neste modelo não existe home machine
- Permite melhor utilização dos recursos de processamento
- Expansão de serviços mais fácil
- Não adequado a aplicações interativas que utilizam recursos gráficos

Clientes Magros e Gordos

O modelo cliente servidor pode ser explorado a partir da consideração do particionamento do software entre cliente e servidor, ou seja, qual a fração de tarefas o cliente realiza antes do trabalho do servidor?

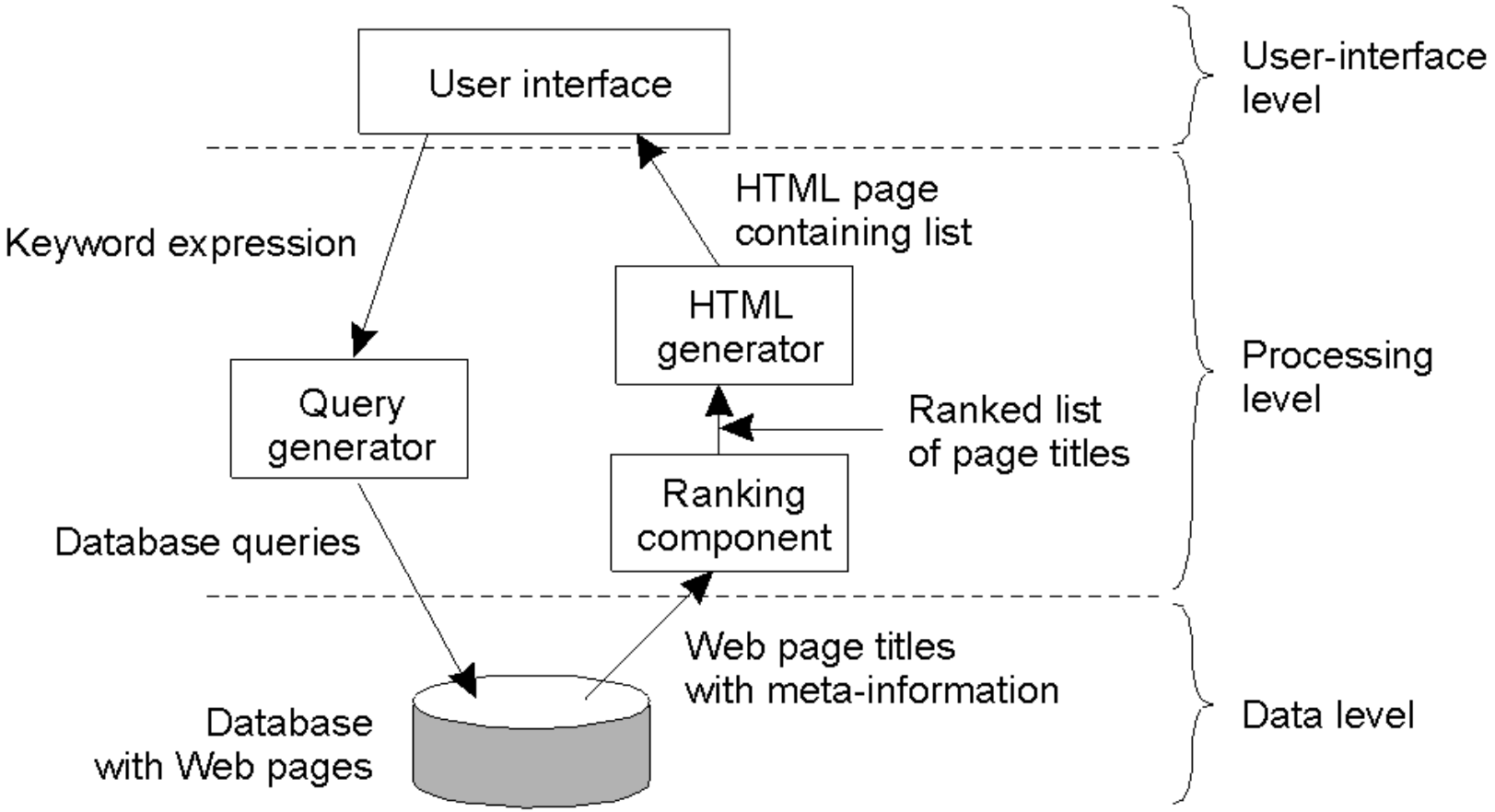
Cliente Magro:

- menor escalabilidade(servidor mais carregado)
- pior desempenho (muita comunicação)
- facilidade de gestão(centralizado)

Cliente Gordo:

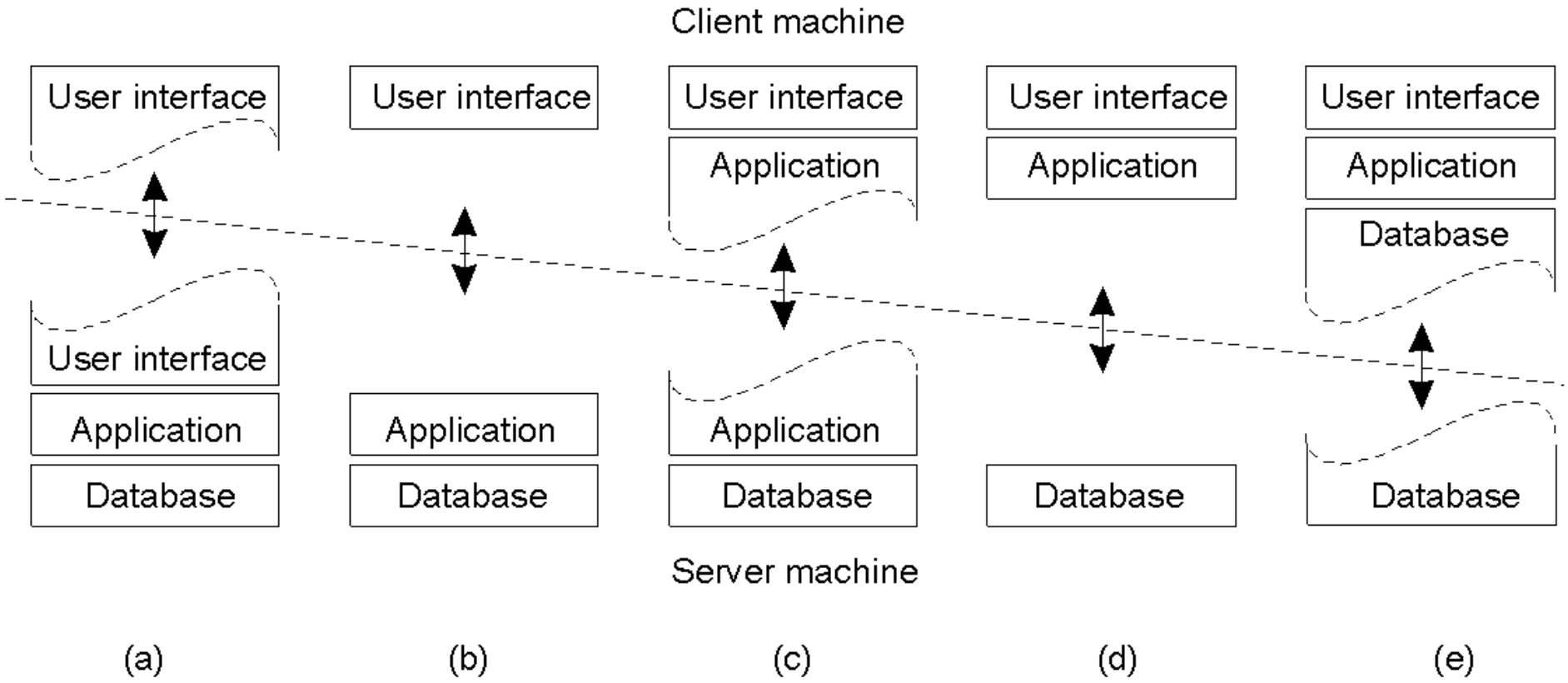
- maior escalabilidade(servidor menos carregado)
- melhor desempenho (menos comunicação)
- dificuldade de gestão(distribuída)

Processing Level



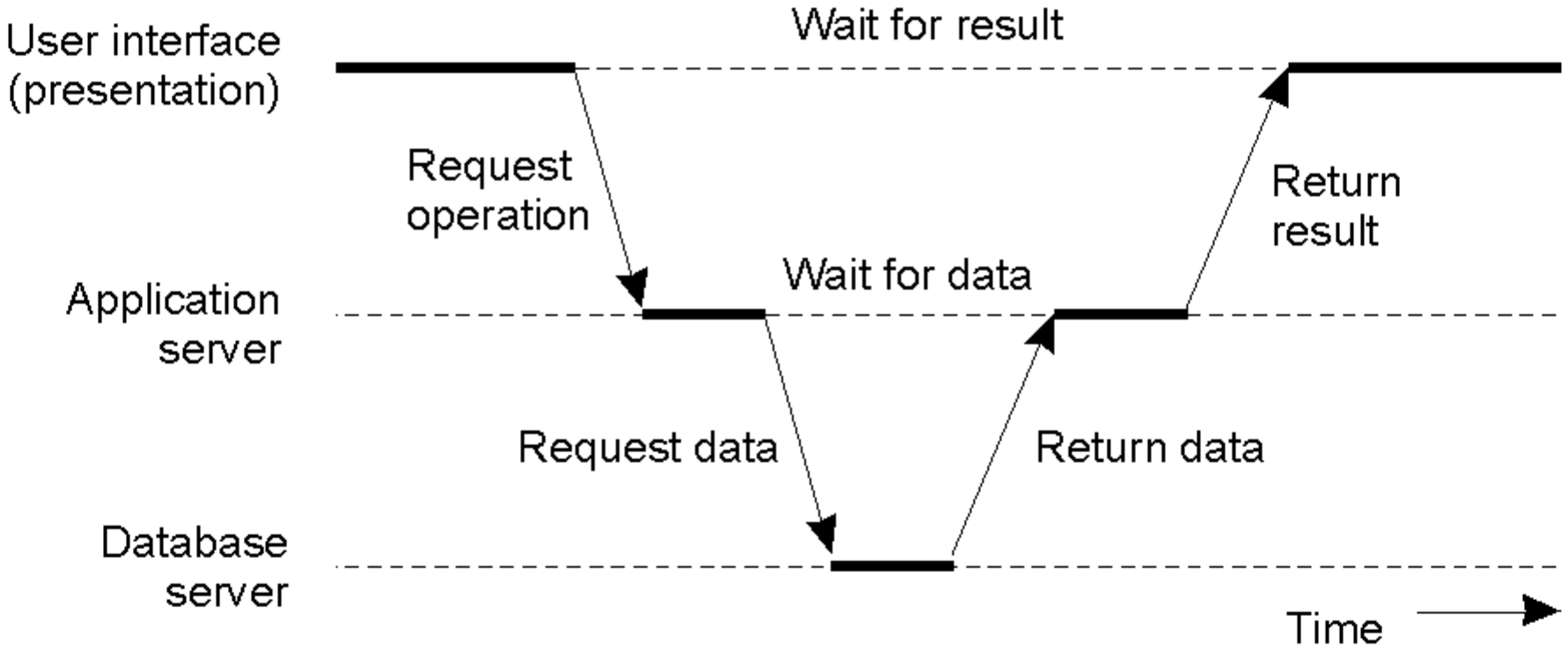
The general organization of an Internet search engine into three different layers

Multitiered Architectures (1)



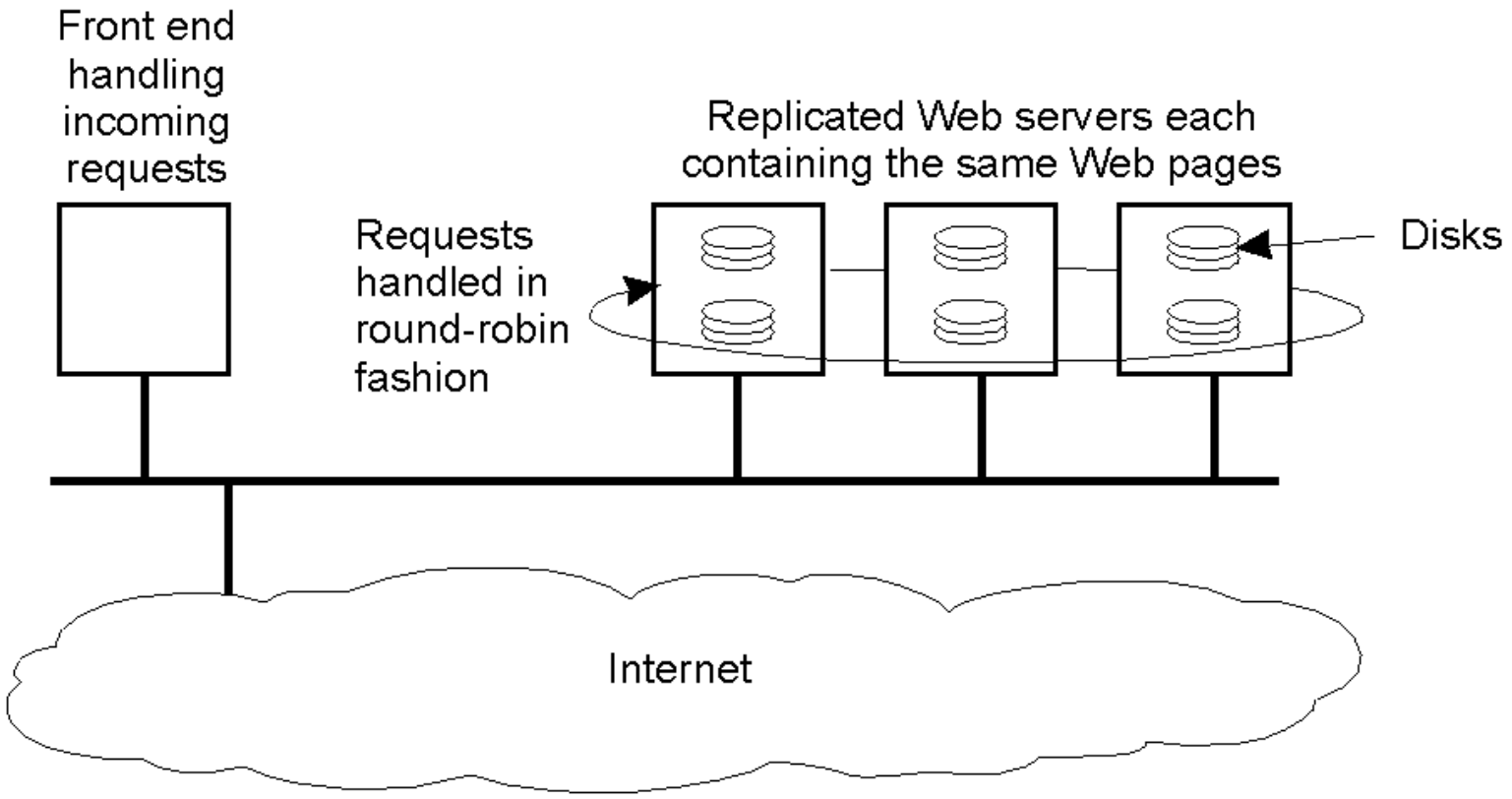
Alternative client-server organizations (a) – (e).

Multitiered Architectures (2)



An example of a server acting as a client.

Modern Architectures



An example of horizontal distribution of a Web service.