Introdução ao Processamento Paralelo e Distribuído

Desligue sua câmera e seu microfone. Utilize o chat para interagir apenas com o professor. Esta sessão está sendo gravada.

Notas dos slides

APRESENTAÇÃO

O presente conjunto de slides pertence à coleção produzida para a disciplina *Introdução ao Processamento Paralelo e Distribuído* ofertada aos cursos de bacharelado em Ciência da Computação e em Engenharia da Computação pelo Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas.

Os slides disponibilizados complementam as videoaulas produzidas e tratam de pontos específicos da disciplina. Embora tenham sido produzidos para ser assistidos de forma independente, a sequência informada reflete o encadeamento dos assuntos no desenvolvimento do conteúdo programático previsto para a disciplina.

computação
Universidade Federal de Pelotas

universidade federal de pelotas
CDTec
centro de desenvolvimento tecnológico

Sistemas Distribuídos

Modelos de Arquiteturais

Notas da videoaula

DESCRIÇÃO

Nesta videoaula são apresentados os conceitos fundamentais da construção de Sistemas Distribuídos e de seus modelos arquiteturais.

OBJETIVOS

Nesta videoaula, em continuação à aula anterior, o aluno compreenderá os principais conceitos associados aos Sistemas Distribuídos e poderá identificar diferentes modelos arquiteturais empregados para construí-los.

“ To create architecture is to put in order. Put what in order?

Function and objects.

Le Corbusier

Estilos Arquiteturais

O **estilo arquitetural** de um SD é definido em termos de componentes e conectores.

- Os componentes devem ter interfaces bem definidas, permitindo substituição
- Componentes são conectados entre si
- A conexão entre os componentes permite troca de dados
- A configuração dos componentes e conectores deve oferecer a visão de um sistema único

- O componente é uma unidade modular
- O conector é o mecanismo de comunicação, coordenação e/ou cooperação entre os componentes (como RPC, troca de mensagens, ...)

Estilos Arquiteturais

- Em camadas
- Baseado em objetos
- Baseada em recursos
- Publish/Subscribe

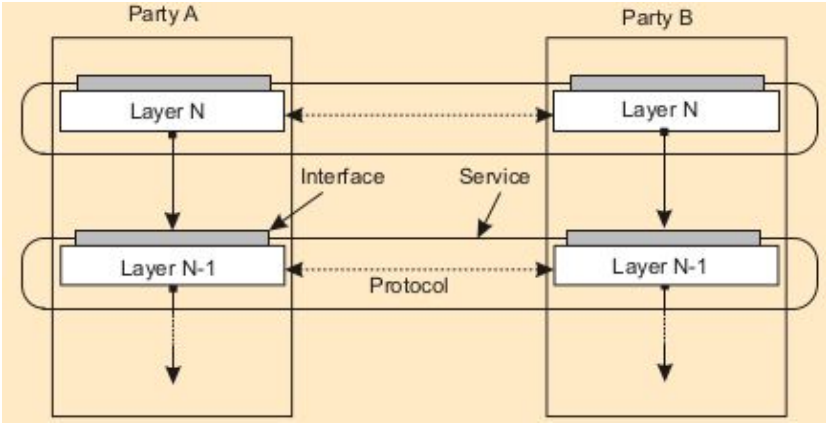
SD em Camadas

Exemplo: pilha de um protocolo de comunicação

Cada camada disponibiliza uma interface de acesso aos serviços que oferece.

Cada camada tem um protocolo para interação entre componentes distintos.

A demanda da camada N é considerada o dado a ser transmitido pela camada N-1.



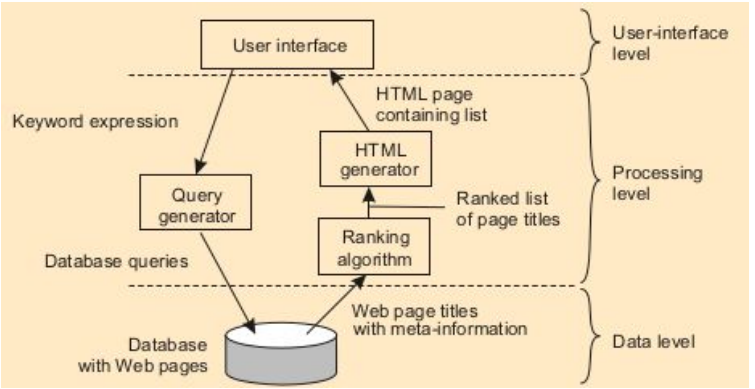
SD em Camadas

<u>Exemplo</u>: camadas de aplicação em um sistema de busca na web

A interface com usuário oferece acesso aos serviços (a usuários ou outros sistemas)

A camada de processamento realizada a operacionalização das funcionalidades

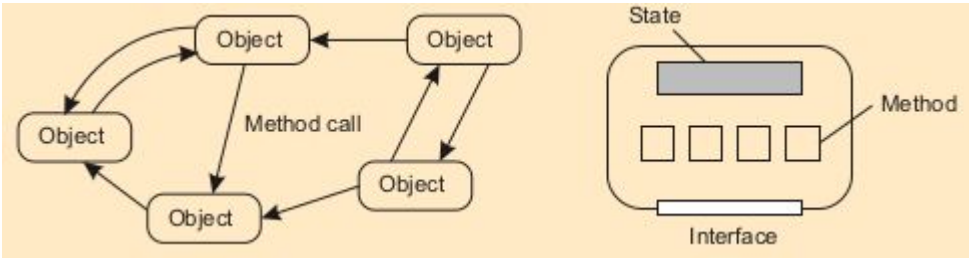
A camada de dados contém os dados a serem manipulados pelos demais componentes.



SD Baseado em Objetos

O que se tem é a abstração em termos de objetos que interagem. A essência é a mesma da programação OO: o objeto encapsula dados e serviços, sendo a chamada de serviços realizada por invocações (remotas) a métodos.

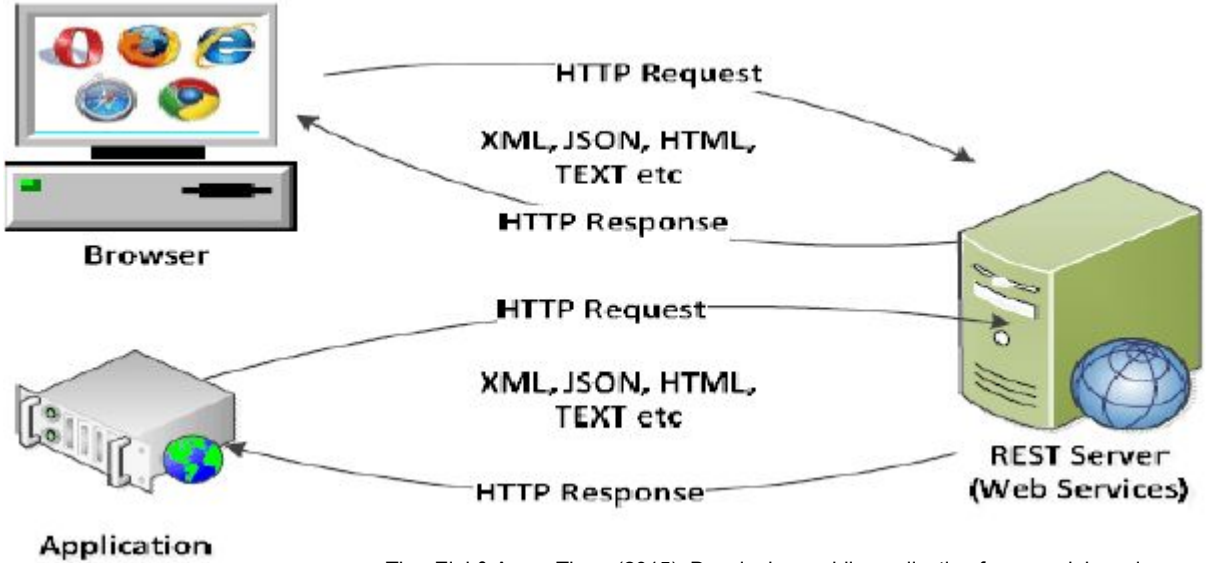
Muito utilizado para construir aplicações com Java/RMI ou Corba.



SD Baseado em Recursos

Arquiteturas RESTful

O SD é entendido como uma coleção de recursos, gerenciados individualmente pelos componentes. Os componentes podem adicionar, retirar, encontrar, modificar dinamicamente os recursos disponíveis.



Thu, Eiei & Aung, Than. (2015). Developing mobile application framework by using RESTFuL web service with JSON parser. 388.

Exemplo: Amazon Storage Service

SD Baseado em Recursos

Arquiteturas RESTful

- Recursos são identificados por um nome
- Todos os serviços oferecem a mesma interface
- As mensagens enviadas por ou a um serviço devem ser inteiramente descritas (descrever tanto dados como ações a serem executadas)
- Um componente não armazena informações sobre o componente que realizou a solicitação de um serviço

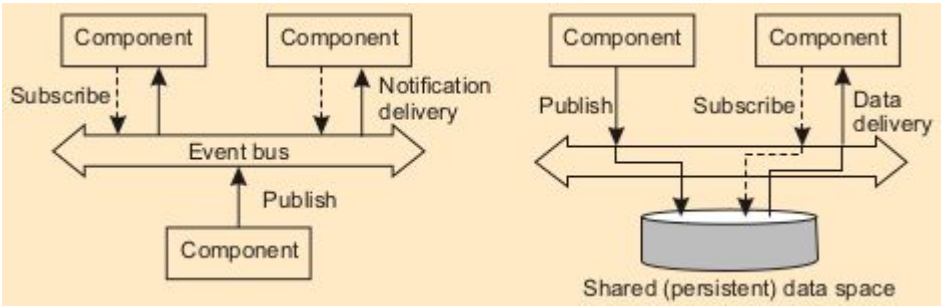
Operações básicas:

- PUT: Cria um novo recurso
- GET: Obtém informações sobre um recurso
- DELETE: Remove do sistema um recurso
- POST: Atribui a um recurso um novo estado

SD Publish/Subscribe

Valorizam o caráter dinâmico dos componentes no sistema: entrar e sair.

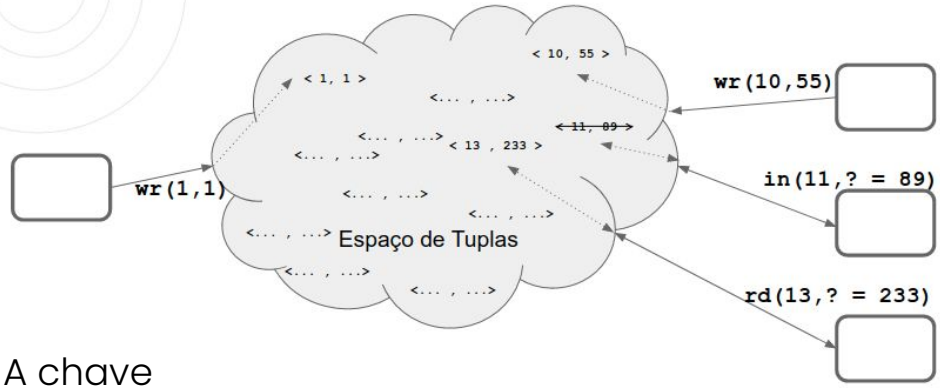
A implementação pode considerar o envio direto das publicações para cada assinante ou prever um componente (ou vários) responsável por armazenar, segundo alguma política, as postagens e reencaminhá-las aos assinantes.



SD Publish/Subscribe

Exemplo: Linda tuple space

Trata-se de uma linguagem de coordenação, na qual um conjunto de quatro operações básicas estendem o potencial de uma segunda linguagem ao acesso de um espaço de dados compartilhado.

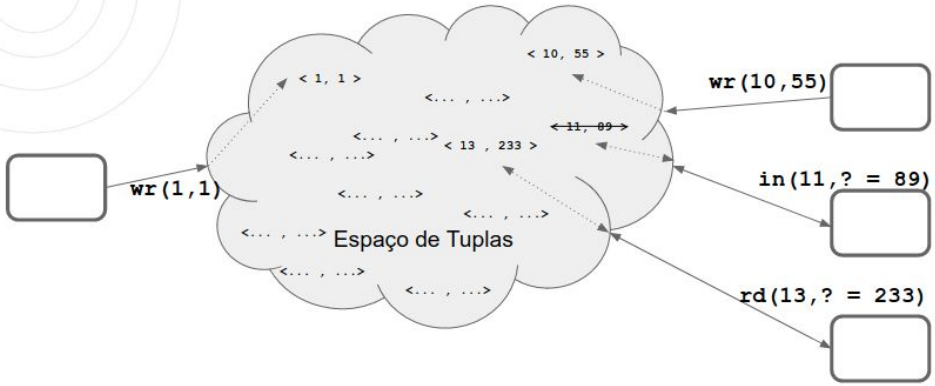


Uma tupla possui uma chave e um dado. A chave representa a identificação do dado no espaço de tuplas.

SD Publish/Subscribe

Exemplo: Linda tuple space

Trata-se de uma linguagem de coordenação, na qual um conjunto de quatro operações básicas estendem o potencial de uma segunda linguagem ao acesso de um espaço de dados compartilhado.



Operações:
RD(key,dta)`: Lê uma tupla do espaço de tuplas correspondendo a key informada
OUT(key, dta): Escreve uma nova tupla
IN(key,dta)`: Retira do espaço de tuplas a tupla que corresponder a key informada
EVAL(key_out,F,key_in): Executa o serviço F no espaço de tuplas, usando como entrada a tupla key_in e armazenando o resultado em key_out

O Problema do Middleware

A ideia é que o middleware promova a interação entre os componentes. As interfaces devem ser, portanto, deveriam ser compatíveis entre todos componentes. Na prática, não são. Entram em ação:

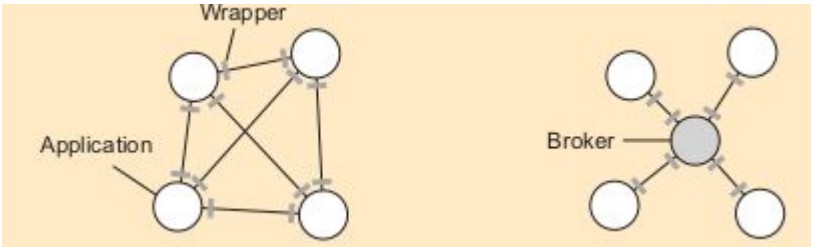
- Wrappers
- Brokers
- Interceptors

O Problema do Middleware

Wrappers e Brokers

Um wrapper é um componente que oferece uma interface aceita pelo cliente. Sua função é adaptar a comunicação entre dois componentes.

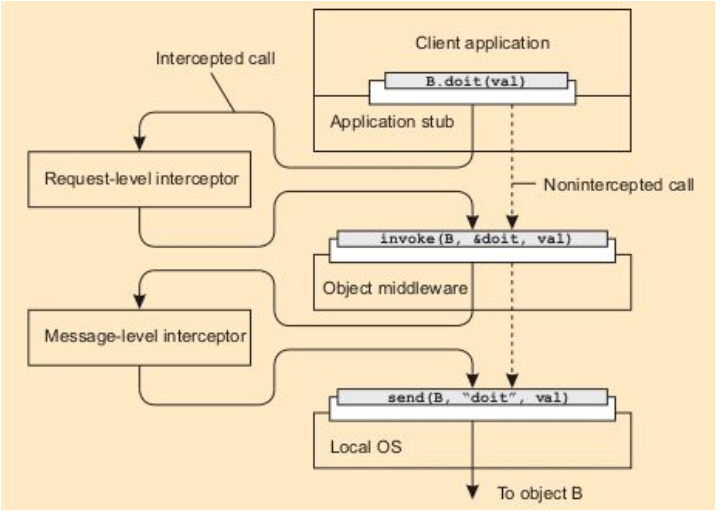
Um broker, por sua vez, permite a adaptação das comunicações entre vários componentes.



O Problema do Middleware

Interceptor

É uma peça de software que se interpõe entre dos componentes. Seu papel é realizar um conjunto de operações sem que sua ação remonte ao cliente. No exemplo, uma requisição de serviço é realizada pelo cliente ao componente gerido no middleware. Sem conhecimento do cliente, por interceptação, réplicas do componente de serviço são invocadas.

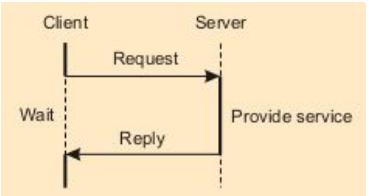


Esquemas arquiteturais

- Centralizada
  - Cliente/Servidor
  - Multicamada
- Descentralizada
  - P2P

Esquemas arquiteturais

- Centralizada
  - Cliente/Servidor
  - Multicamada



Ideia básica:

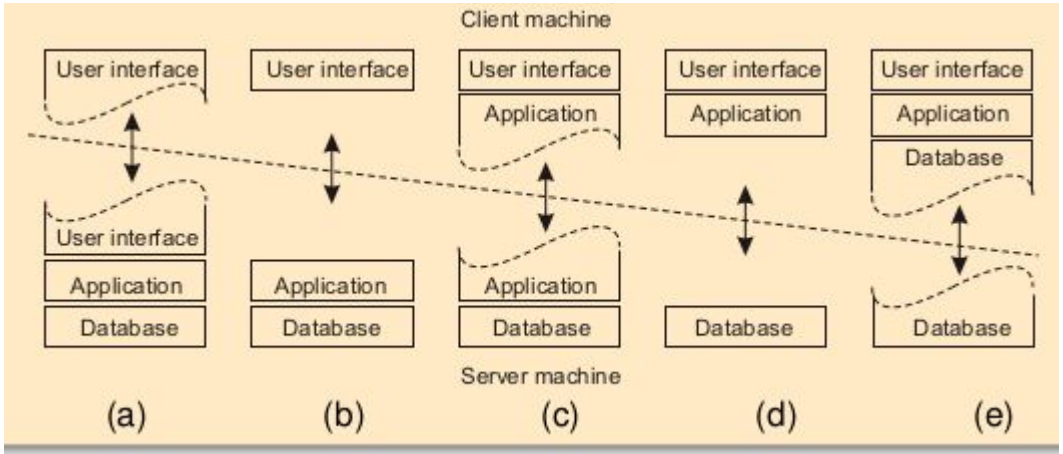
- Alguns componentes oferecem serviços
- Outros consomem

Esquemas arquiteturais

- Centralizada
  - Cliente/Servidor
  - Multicamada

Ideia básica:

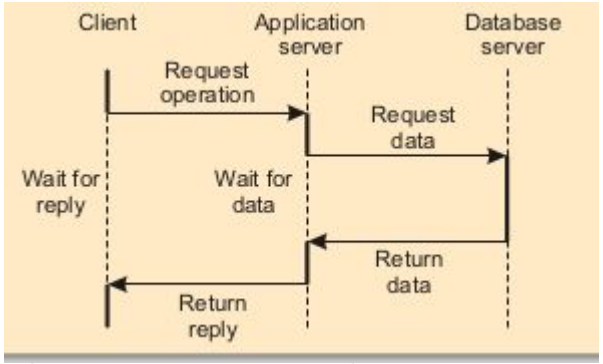
- Alguns componentes oferecem serviços
- Outros consomem



Exemplo de possibilidade de modelos cliente/servidor em duas camadas

Esquemas arquiteturais

- Centralizada
  - Cliente/Servidor
  - Multicamada



Ideia básica:

- Alguns componentes oferecem serviços
- Outros consomem

Exemplo de possibilidade de modelos cliente/servidor em ttrês camadas, onde o servidor da aplicação acessa, como cliente, o servidor de dados.

Esquemas arquiteturais

- Descentralizado
  - Redes P2P
    - Estruturadas
    - Não estruturadas

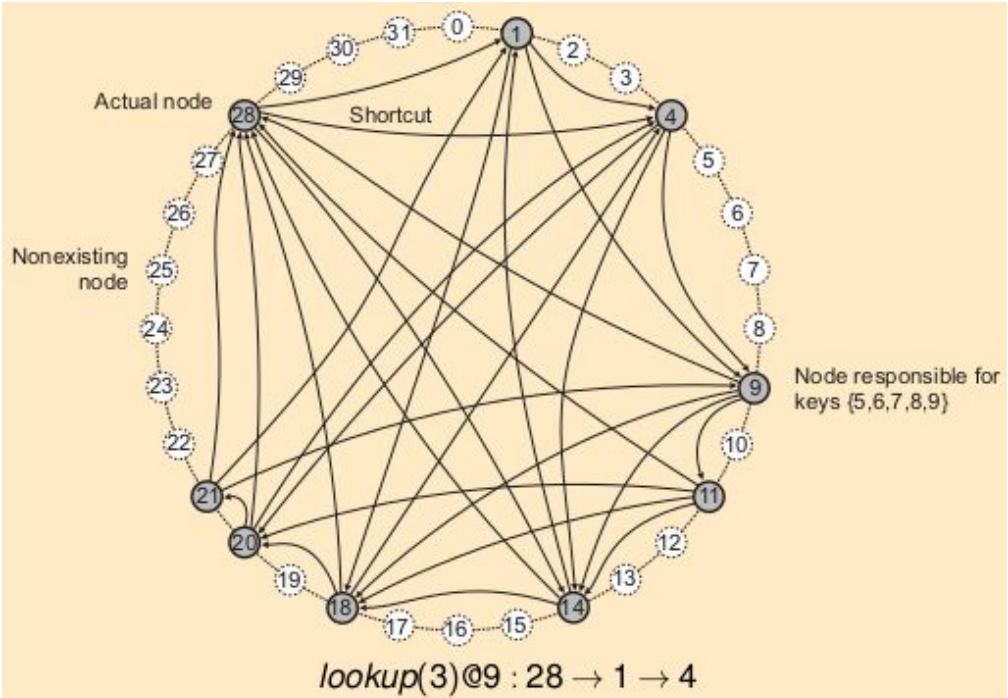
Todos os componentes tem o mesmo papel, de cliente e servidor. Existe a necessidade de endereçamento do serviço/dado.

Esquemas arquiteturais

- Descentralizado
  - Redes P2P
    - Estruturadas

Exemplo: Chord

- Os nós possuem um identificador m-bit e são organizados em um anel
- Existem atalhos no anel
- Cada item de dado possui um hash para uma chave m-bit
- Um item de dados com a chave k é armazenada no nó com o menor identificador que respeitar id ≥ k, chamado sucessor da chave k



Esquemas arquiteturais

- Descentralizado
  - Redes P2P
    - Não estruturadas

Não existe uma estrutura conhecida, cada nó mantém uma lista de vizinhos construída a medida que a computação evolui (ad-hoc). Como resultado, é criado um grafo onde as arestas (u,v) indicam a probabilidade de haver uma conexão entre os nós u e v.

Uma busca em uma rede P2P pode se dar por

- Flooding
- Random walker


Obrigado!