Universidade Federal de Goiás - UFG
Instituto de Informática - INF

Aluno: Bruno Ribeiro Cardoso de Morais

Definição de Sistemas Distribuídos e Termos

A principal motivação para construir um sistema distribuído é o compartilhamento de recursos entre os componentes envolvidos no sistema. Tais como, arquivos, impressoras, serviços, componentes entre outros. A construção de um SD envolve alguns desafios como; a concorrência de componentes onde os vários componentes envolvidos no sistema tentam acessar o mesmo recurso no mesmo instante de tempo. A falta de um relógio global gerando problemas de sincronização e coordenação entre os componentes, onde cada um baseia-se no seu próprio relógio local. As falhas de componentes independentes onde inevitavelmente algum componente do sistema irá falhar em algum momento independentes dos outros componentes. A heterogeneidade visto que o sistema é composto por componentes que apresentam diferentes tipos de hardwares, SOs, softwares e que são conectados por diferentes tipos de redes. A escalabilidade onde o sistema deve ser capaz de lidar com as variações de escala sem comprometer proporcionalmente o seu desempenho. A transparência onde o sistema torna os aspectos da distribuição (acesso, localização, realocação, replicação, migração, concorrência e falhas) ocultos para os programadores e clientes. Arquitetura de SD define a estrutura do sistema em termos de componentes, localizados em uma rede de computadores, especificados separadamente e suas interações (comunicação e coordenação). Os principais modelos de arquitetura são, cliente/servidor e peer-to-peer. Na arquitetura C/S temos os processos clientes interagindo com os processos servidores, para acessar os recursos compartilhados que este gerencia. Por sua vez os servidores podem ser clientes de outros servidores. Por exemplo os servidores web são clientes de servidores DNS. No P2P todos os processos envolvidos em uma tarefa desempenham funções semelhantes, interagindo cooperativamente como pares, sem distinção entre processos clientes e servidores. Em um SD tanto processos quanto canais de comunicação podem falhar. O modelo de falhas define como uma falha pode se manifestar em um SD, proporcionando um entendimento de seus efeitos e consequências. Podendo ser falhas por omissão, arbitrárias e de sincronização. As falhas por omissão acontece quando um processo ou canal de comunicação deixa de executar as ações que deveria. Na semântica de chamadas maybe (talvez), os clientes não sabem nada sobre o resultado da RPC. A at-least-once garante que a solicitação foi executada uma ou mais vezes. A at-most-once que a solicitação foi executada no máximo um vez. O XDR (representação externa de dados) é um padrão que é alcançado através do empacotamento (marshalling) dos dados nativos transformando-os em uma representação externa, para a trasmissão dos mesmos na rede. Já o unmarshalling é o desempacotamento dos dados na representação externa transformando-os em dados nativos. A serialização é uma técnica que permite converter objetos em bytes (colocando-os em série) podendo ser salvos em disco ou enviados através de um stream (HTTP, via socket…). Os componentes em um SD comunicam entre si por mensagens usando três tipos de paradigmas. Comunicação entre processos (IPC), comunicação remota (RMI, RPC, Request-Replay) e comunicação indireta (publish-subscriber, memória compartilhada, filas de mensagens entre outros). No paradigma IPC a comunicação entre processos ocorre em um nível de abstração relativamente baixo, as primitivas de passagem de mensagens (send, receive). Acesso direto a API de sockets e suporta multcasting. OS protocolos de requisição e resposta são um padrão imposto em serviço de passagem de mensagem para suportar C/S. Onde um cliente faz uma requisição para o servidor e este responde. Na chamada de procedimento remoto (RPC), procedimentos nos processos remotos podem ser chamados como se fosse procedimentos no

espaço de endereçamento local, ocultando aspectos importantes da distribuição. RMI (invocação remota de métodos) é muito parecida com RPC, mas voltada para objetos distribuídos. Um objeto chamador pode invocar um método remoto em um objeto remoto, usando invocações de método. Middleware é um software localizado entre os SO e camada de aplicação e serviços, abastraindo a heterogeneidade da rede, dos SOs, dos hardwares, das linguagens de programação, possibilitando a comunicação entre os componentes, fornecendo a mesma interface para todos nós. CORBA é um middleware que oferece uma linguagem de definição de interface (IDL) a qual fornece recursos para definição de (módulos, interfaces, tipos, atributos e assinaturas de módulos), que é usada para definir interface remota, que por sua vez especifica quais métodos do objeto pode ser invocado de forma remota. Referência de objetos os objetos podem ser acessados por meio de referências de objetos. Para invocar um método em um objeto é necessário a referência do objeto e o nome do método. O bind ocorre quando um nome (identificador) é associado a um objeto. O serviço de nomes faz o mapeamento do nome de um objeto remoto para sua referência através do bind. O stub do lado cliente faz o marshalling dos parâmetros e unsmarshalling do valor de retorno. O stub permite o cliente realizar chamadas de procedimentos remotos como se fosse no espaço de endereçamento local. O skeleton faz o unmarshalling dos parâmetros e o marshalling do valor de retorno. A chamada local ocorre quando um processo realiza uma chamada de um método dele mesmo quando está em execução. Enquanto chamada remota ocorre quando um processo realiza uma chamada de um método em outro processo em execução no mesmo host ou em hosts diferentes.