Engenharia de Software
Software Concorrente e Distribuído

Universidade Federal de Goiás
Marcus Mitra Muniz Inácio - 201508741
Pedro Henrique Pires - 201515577

9 Perguntas Sobre Sistemas Distribuídos

1. Apresente cinco tipos de recursos de hardware e cinco tipos de recursos de dados ou de software que podem ser compartilhados de forma útil. Dê exemplos práticos de compartilhamento desses recursos em sistemas distribuídos.

Hardware:

1. Impressoras
2. Discos
3. Processadores
4. Memória RAM
5. Sensores

Software:

1. Arquivos
2. Banco de Dados
3. Servidores
4. Serviços(Aplicações)
5. Protocolos de Comunicação

Exemplos:

- Impressoras e sensores disponíveis na rede para que outras aplicações tenham acesso às suas funções.
- Discos de armazenamento para armazenar dados dos bancos de dados das aplicações.
- Concorrência por processamento de dados.

2. Considere diferentes estratégias para implementação de jogos online multi-player em grande escala. Quais as vantagens de se usar uma abordagem de servidor único para representar o estado do jogo? Quais os problemas dessa abordagem e como poderiam ser resolvidos?

A vantagem da abordagem de servidor único seria a melhor coesão do estado do jogo para todos os jogadores e um melhor tempo de resposta, já que não necessitaria de comunicação entre diversos serviços para o funcionamento do servidor.

Porém a grande desvantagem é que uma falha no servidor significaria uma completa falta de acesso ou o jogo se tornaria "injogável" até a falha ser resolvida. Outro grande problema seria a escalabilidade do servidor, com cada vez mais usuários, o servidor iria necessitar de cada vez mais recursos e isso poderia se tornar insustentável rapidamente.

Esses problemas poderiam ser resolvidos com o uso de sistemas distribuídos: vários servidores responsáveis por diferentes "áreas" do jogo, e consequentemente diferentes jogadores e diferentes dados; Serviços distribuídos entre diferentes aplicações assim, se uma aplicação falhar, sua falha será isolada.

3. Compare e contraste o modelo de computação em nuvem com modelos mais tradicionais de computação cliente-servidor. Qual a novidade da computação em nuvem enquanto conceito?

O modelo mais atual em nuvem gera uma maior transparência no quesito de servidores, são servidores que podem estar fisicamente muito distantes e ainda sim funcionarem como um só, caso um precise de manutenção ou fique indisponível, ele é facilmente substituído por outro servidor da nuvem.

Isso contrasta com o modelo mais tradicional de cliente-servidor, onde geralmente temos uma única instalação de servidor em um mesmo espaço físico de uma empresa, tornando mais difícil a manutenção(caso você precise derrubar um servidor para manutenção, nem sempre há a disponibilidade de recursos para ter um servidor de redundância) e dificultando a escalabilidade.

Assim, empresas especializadas em proverem servidores em nuvem acabam tendo um grande benefício em relação ao modelo tradicional de cliente-servidor. Conseguindo sempre manter os servidores funcionando e manter uma escalabilidade mais fluida.

4. Use a WWW como exemplo para ilustrar o conceito de compartilhamento de recursos usando o modelo cliente-servidor. Quais as vantagens e desvantagens do uso de HTML, URLs e HTTP como tecnologias de suporte para o compartilhamento de recursos na forma de navegação na Web? Alguma dessas tecnologias é adequada para computação cliente-servidor de propósito geral?

HTML seria basicamente indispensável caso o usuário necessite de alguma interação no meio desse compartilhamento de recursos. A nível de comunicação de sistemas é dispensável.

URLs e HTTP são viáveis para a computação cliente-servidor, o HTTP está mais próximo da camada TCP e carrega informações relevantes sobre a conexão e o uso de URLs pode ser interessante para padronizar comunicações(assim como o HTTP) e facilitar acesso aos bancos de dados.

5. Um programa servidor escrito em uma linguagem (C++, por exemplo) provê a implementação de um objeto que pode ser acessado por clientes escritos em linguagens diferentes (por exemplo, Java). Os computadores clientes e servidores podem ter hardware diferente uns dos outros, mas todos estão conectados à Internet. Descreva os problemas decorrentes dos vários aspectos de heterogeneidade e que

precisam ser resolvidos para tornar possível a um objeto cliente chamar um método em um objeto servidor.

A chave da questão é a padronização, ou seja, utilizar padrões em todos os componentes presentes em um sistema distribuído para que não haja problemas com a heterogeneidade dos sistemas.

Já em questões de hardwares diferentes, diferentes SOs, é interessante a utilização de um “Docker”(Container) para que as aplicações consigam rodar e se comunicar mesmo em hardwares e ambientes heterogêneos.

6. Discuta as possíveis falhas que podem ocorrer quando um processo cliente realiza uma chamada a um objeto servidor. O que pode ser feito para que o sistema seja tolerante a essas falhas?

Pode haver falha na comunicação(rede), podem haver falhas em diversos sistemas. É muito difícil citar todas as possíveis falhas de um sistema distribuído. Porém é possível falarmos de soluções, que são: redundância de hardware e redundância de software.

7. Considere um processo servidor que mantém um objeto compartilhado, o qual pode ser acessado por vários clientes ao mesmo tempo. Que problemas ou “interferências" podem ocorrer ao se permitir a execução concorrente de requisições de múltiplos clientes? Como prevenir essas interferências?

Caso esse objeto seja apenas acessado e não modificado, os problemas que poderiam ocorrer seriam apenas as falhas de sistema. Em caso de modificação por cada requisição, poderia-se criar um objeto inconsistente com a ordem de eventos caso não seja utilizado algum método de controle como o Relógio de Lamport.

8. Considere um serviço que pode ser implementado por vários servidores replicados. Seria suficiente o uso de comunicação multicast

das requisições dos clientes (para todos os servidores) para garantir a sincronização do estado das réplicas?

Sim, a ideia de comunicação multicast é justamente prover consistência em réplicas. Em caso de falha, a última atualização enviada é executada em todas as réplicas ou nenhuma. A atualização será realizada se as outras réplicas restantes concordarem que a réplica que falhou não pertence mais ao grupo.

Quando a réplica na qual houve uma falha se recupera, ela é validada e recebe as atualizações necessárias para sincronizar-se novamente.

9. O uso de URLs em HTTP para identificar recursos possibilita transparência de localização? Explique.

Sim, até certo ponto, geralmente os usuários não irão ver diretamente as urls ligadas às requisições e respostas dos sistemas distribuídos porém, caso consigam(o que já entra em outro problema de transparência) é mais difícil “esconder” a localização, já que muitas URLs possuem em seu endereço, o país de origem(.br, .de, .cn).