Modelos de Sistemas

1. Descreva e ilustre a arquitetura cliente-servidor de uma ou mais aplicações na Internet (por exemplo: Web e e-mail).

A arquitetura cliente-servidor é um modelo de computação em que as tarefas são distribuídas entre os clientes e os servidores em uma rede. Na Internet, essa arquitetura é amplamente utilizada em várias aplicações, como a web e o e-mail.

Para ilustrar essa arquitetura, podemos considerar o exemplo de um site de comércio eletrônico. O cliente, que é o usuário que acessa o site, utiliza um navegador web para fazer uma solicitação ao servidor. Essa solicitação pode ser para visualizar um produto, adicionar um item ao carrinho de compras ou realizar o pagamento de uma compra.

O servidor, que é responsável por processar as solicitações do cliente, recebe a solicitação e realiza a tarefa correspondente. Por exemplo, se o cliente está visualizando um produto, o servidor retorna as informações sobre o produto, como a descrição, o preço e a disponibilidade. Se o cliente está adicionando um item ao carrinho de compras, o servidor armazena essa informação em um banco de dados para que possa ser recuperada posteriormente.

Essa interação entre o cliente e o servidor ocorre através do protocolo HTTP (Hypertext Transfer Protocol). O cliente envia solicitações HTTP para o servidor, que responde com as informações solicitadas. Essas informações podem ser em formato HTML, que é interpretado pelo navegador do cliente e exibido como uma página web.

No caso do e-mail, a arquitetura cliente-servidor também é utilizada. O cliente, que pode ser um aplicativo de e-mail ou um navegador web, envia uma solicitação para o servidor de e-mail para receber ou enviar mensagens. O servidor de e-mail, por sua vez, recebe a solicitação do cliente e executa a tarefa correspondente, como enviar uma mensagem ou recuperar mensagens da caixa de entrada do cliente.

Essa interação entre o cliente e o servidor de e-mail ocorre através de protocolos de e-mail, como SMTP (Simple Mail Transfer Protocol) para enviar mensagens e POP3 (Post Office Protocol version 3) ou IMAP (Internet Message Access Protocol) para receber mensagens.

Em resumo, a arquitetura cliente-servidor é um modelo de computação distribuído que é amplamente utilizado em aplicações da Internet, como a web e o e-mail. Ela envolve a interação entre o cliente, que é o usuário, e o servidor, que é responsável por processar as solicitações do cliente e fornecer as informações solicitadas. Essa interação ocorre através de protocolos de comunicação específicos para cada aplicação.

2. Para a aplicação apresentada no exercício anterior, discuta como os servidores cooperam ao prover um serviço.

No exemplo do site de comércio eletrônico apresentado anteriormente, os servidores cooperam ao prover o serviço de várias maneiras:

Servidores web: responsáveis por fornecer as páginas web solicitadas pelo cliente. Esses servidores geralmente trabalham em conjunto para equilibrar a carga de tráfego, garantir a disponibilidade e a escalabilidade do serviço. Os servidores web podem compartilhar as informações do banco de dados para garantir que as informações exibidas em diferentes páginas sejam consistentes.

Servidores de banco de dados: responsáveis por armazenar as informações dos produtos, clientes e transações. Esses servidores geralmente trabalham em conjunto para garantir a consistência dos dados e a escalabilidade do serviço. Os servidores de banco de dados podem ser replicados para garantir que as informações sejam disponibilizadas de forma redundante e evitar falhas em um único servidor.

Servidores de pagamento: responsáveis por processar as transações financeiras realizadas pelos clientes. Esses servidores trabalham em conjunto com os servidores web e de banco de dados para garantir a segurança e a precisão das transações. Os servidores de pagamento podem se comunicar com os servidores de banco de dados para verificar a disponibilidade de fundos e atualizar as informações do cliente após a conclusão da transação.

Servidores de logística: responsáveis por gerenciar o estoque, o transporte e a entrega dos produtos. Esses servidores podem trabalhar em conjunto com os servidores de pagamento e de banco de dados para garantir que as informações sobre o estoque e a entrega sejam precisas e atualizadas. Os servidores de logística podem se comunicar com os servidores web para fornecer informações de rastreamento de entrega aos clientes.

Em resumo, os servidores cooperam ao prover um serviço de comércio eletrônico ao trabalharem em conjunto para fornecer as informações solicitadas pelo cliente, processar as transações financeiras, gerenciar o estoque e a entrega dos produtos e garantir a segurança e a disponibilidade do serviço. Cada servidor desempenha um papel específico na prestação do serviço e se comunica com outros servidores para garantir a consistência e a precisão das informações fornecidas aos clientes.

3. Como as aplicações apresentadas no exercício 1 acomodam o particionamento, replicação ou caching de dados entre os servidores.

As aplicações apresentadas no exercício 1 acomodam o particionamento, replicação ou caching de dados entre os servidores de diferentes maneiras, dependendo das suas necessidades e características específicas. A seguir, veremos como cada aplicação lida com essas técnicas:

Aplicação web: no caso de uma aplicação web, o particionamento de dados pode ser utilizado para distribuir as informações em diferentes servidores, de modo a equilibrar a carga de trabalho e melhorar o desempenho da aplicação. Por exemplo, um servidor pode ser responsável por fornecer informações sobre produtos de uma determinada categoria, enquanto outro servidor fornece informações sobre produtos de outra categoria. A

replicação de dados pode ser utilizada para garantir a disponibilidade e a redundância dos dados, permitindo que a aplicação continue funcionando mesmo que um servidor falhe. O caching de dados pode ser utilizado para armazenar em cache informações frequentemente acessadas, como páginas web, reduzindo o tempo de resposta e o tráfego na rede.

Aplicação de e-mail: no caso de uma aplicação de e-mail, o particionamento de dados pode ser utilizado para distribuir as informações dos usuários em diferentes servidores, de modo a equilibrar a carga de trabalho e melhorar o desempenho da aplicação. A replicação de dados pode ser utilizada para garantir a disponibilidade e a redundância dos dados, permitindo que a aplicação continue funcionando mesmo que um servidor falhe. O caching de dados pode ser utilizado para armazenar em cache mensagens frequentemente acessadas, como as mensagens mais recentes, reduzindo o tempo de resposta e o tráfego na rede.

Em resumo, as aplicações apresentadas no exercício 1 podem utilizar o particionamento, replicação ou caching de dados entre os servidores de diferentes maneiras para melhorar o desempenho, a disponibilidade e a redundância dos dados. Essas técnicas são utilizadas de acordo com as necessidades e características específicas de cada aplicação.

4. Sugira algumas aplicações para o modelo de processos pares (peers), distinguindo entre os casos quando o estado de todos pares necessita ser idêntico e casos que exijam menos consistência.

O modelo de processos pares (peers) é frequentemente utilizado em sistemas distribuídos para permitir que um conjunto de computadores trabalhem juntos de forma descentralizada, compartilhando informações e recursos sem a necessidade de um servidor centralizado. Abaixo, apresento algumas aplicações que podem utilizar esse modelo de processos, distinguindo entre os casos em que o estado de todos os pares precisa ser idêntico e os casos que exigem menos consistência:

Compartilhamento de arquivos: o compartilhamento de arquivos é uma aplicação comum que utiliza o modelo de processos pares. Nesse caso, o estado de todos os pares precisa ser idêntico, garantindo que todas as cópias dos arquivos estejam atualizadas e consistentes. Isso é necessário para evitar conflitos e garantir que todas as alterações sejam salvas corretamente.

Mensageiros instantâneos: os mensageiros instantâneos também podem ser implementados com o modelo de processos pares. Nesse caso, o estado de todos os pares não precisa ser idêntico o tempo todo, pois é aceitável que uma mensagem possa ser entregue a alguns pares antes de outros. No entanto, é importante que todas as mensagens sejam entregues eventualmente e que os pares mantenham uma lista atualizada dos usuários conectados.

Sistemas de recomendação: os sistemas de recomendação podem utilizar o modelo de processos pares para recomendar itens a usuários com base em suas preferências e comportamentos anteriores. Nesse caso, o estado de todos os pares não precisa ser idêntico, mas é importante que os pares troquem informações periodicamente para atualizar as recomendações e garantir que os usuários recebam as recomendações mais relevantes.

Jogos em rede: os jogos em rede também podem utilizar o modelo de processos pares. Nesse caso, o estado de todos os pares precisa ser idêntico para garantir que todos os jogadores vejam a mesma cena e que as ações de cada jogador sejam sincronizadas corretamente. É importante que as informações sejam atualizadas em tempo real para evitar atrasos e erros na jogabilidade.

Em resumo, o modelo de processos pares pode ser utilizado em uma variedade de aplicações, distinguindo entre os casos em que o estado de todos os pares precisa ser idêntico e os casos que exigem menos consistência. O modelo é especialmente útil para aplicações em que os recursos são compartilhados e precisam ser atualizados em tempo real entre os pares.

5. Liste os tipos de recursos locais que são vulneráveis para um ataque por um programa não confiável que é carregado de um local remoto e executado em um computador local.

Quando um programa não confiável é carregado de um local remoto e executado em um computador local, ele pode ter acesso a diferentes tipos de recursos locais, que podem ser vulneráveis a ataques. Alguns dos recursos mais comuns incluem:

Arquivos locais: um programa não confiável pode acessar e modificar arquivos no sistema de arquivos local, incluindo documentos pessoais, bancos de dados, senhas e outras informações confidenciais.

Conexões de rede: um programa mal-intencionado pode se conectar a outras máquinas na rede local e na Internet para acessar dados ou iniciar ataques a outras máquinas na rede.

Memória do sistema: um programa não confiável pode acessar a memória do sistema, que pode conter informações sensíveis, como senhas e chaves criptográficas.

Acesso à webcam e ao microfone: programas maliciosos podem acessar a webcam e o microfone do sistema sem o conhecimento do usuário para capturar imagens e gravações de áudio.

Informações do sistema: um programa mal-intencionado pode coletar informações do sistema, como o nome do computador, endereço IP e outros dados de identificação, que podem ser usados para fins maliciosos.

Acesso a periféricos: programas não confiáveis podem acessar periféricos conectados ao sistema, como dispositivos USB, para copiar dados ou instalar malware.

Registro do sistema: um programa malicioso pode modificar as configurações do registro do sistema para iniciar automaticamente quando o computador é iniciado, ocultar sua presença e executar outras ações maliciosas.

Em geral, é importante ter cuidado ao baixar e executar programas de fontes desconhecidas, pois esses programas podem conter malware que pode acessar recursos

locais vulneráveis e comprometer a segurança do sistema. Recomenda-se sempre utilizar softwares de antivírus atualizados e manter o sistema operacional e demais programas atualizados com as últimas correções de segurança.

6. Dê alguns exemplos de aplicações onde o uso de código móvel é interessante.

O uso de código móvel é interessante em diversas aplicações que envolvem a transferência de código de uma máquina para outra, tais como:

Atualizações de software: quando uma atualização de software é lançada, o código móvel pode ser usado para atualizar automaticamente o software em dispositivos conectados à Internet, como smartphones, tablets e computadores.

Jogos em rede: jogos em rede podem usar código móvel para enviar atualizações, novos recursos e correções de bugs para dispositivos dos jogadores em tempo real.

Sistemas distribuídos: em sistemas distribuídos, o código móvel pode ser usado para enviar funções e tarefas de um nó para outro, tornando o sistema mais flexível e escalável.

Aplicativos de compartilhamento de arquivos: aplicativos de compartilhamento de arquivos podem usar código móvel para transferir arquivos entre dispositivos, permitindo que usuários acessem seus arquivos em diferentes dispositivos.

Sistemas de gerenciamento de redes: sistemas de gerenciamento de redes podem usar código móvel para monitorar a rede e realizar ações de correção de falhas remotamente.

Computação em nuvem: em sistemas de computação em nuvem, o código móvel pode ser usado para fornecer recursos de processamento e armazenamento em dispositivos remotos, permitindo que os usuários realizem tarefas mais complexas em seus dispositivos.

Sistemas de inteligência artificial: sistemas de inteligência artificial podem usar código móvel para enviar modelos e algoritmos de um dispositivo para outro, permitindo que diferentes dispositivos contribuam para o processo de treinamento e inferência.

7. Quais fatores afetam o tempo/capacidade de resposta (responsiveness de uma aplicação que acessa dados compartilhados gerenciados por um servidor. Descreva e discuta alternativas para os problemas.

O tempo/capacidade de resposta (responsiveness) de uma aplicação que acessa dados compartilhados gerenciados por um servidor é afetado por diversos fatores, tais como:

Latência da rede: o tempo necessário para enviar e receber dados entre o cliente e o servidor pode afetar a capacidade de resposta da aplicação. Uma rede lenta pode resultar em tempos de espera prolongados para carregar dados e enviar solicitações.

Sobrecarga do servidor: se o servidor estiver sobrecarregado com muitas solicitações simultâneas, a capacidade de resposta da aplicação pode ser reduzida. Isso pode levar a tempos de espera prolongados e atrasos na entrega de dados.

Desempenho do banco de dados: a velocidade e eficiência do banco de dados usado pelo servidor pode afetar a capacidade de resposta da aplicação. Se o banco de dados estiver lento ou mal projetado, pode haver atrasos na recuperação e armazenamento de dados.

Uso excessivo de recursos do cliente: se a aplicação estiver usando muitos recursos do cliente, como memória ou processamento, isso pode afetar a capacidade de resposta da aplicação e reduzir o desempenho do sistema como um todo.

Para solucionar esses problemas, existem algumas alternativas que podem ser adotadas:

Melhorar a latência da rede: pode ser possível melhorar a latência da rede usando tecnologias como cache de conteúdo ou alocando servidores mais próximos do cliente. Também é possível reduzir a quantidade de dados que precisam ser transferidos para melhorar o desempenho.

Escalar o servidor: uma alternativa é escalar o servidor adicionando mais recursos, como mais CPUs, memória ou armazenamento. Também pode ser possível usar balanceamento de carga para distribuir a carga entre vários servidores.

Otimizar o banco de dados: pode ser possível melhorar o desempenho do banco de dados otimizando as consultas, definindo índices apropriados e ajustando as configurações do banco de dados.

Reduzir o uso de recursos do cliente: uma alternativa é reduzir a quantidade de processamento que ocorre no cliente, usando técnicas como lazy loading ou carregamento incremental de dados. Também é possível usar algoritmos de compressão para reduzir a quantidade de dados que precisam ser transferidos.

8. Diferencie buffering e caching.

Buffering e caching são dois conceitos diferentes, embora possam ser usados em conjunto para melhorar o desempenho de aplicações que precisam acessar e manipular dados.

O buffering é um mecanismo de armazenamento temporário de dados usados para otimizar a comunicação entre diferentes componentes de uma aplicação. Quando um componente de uma aplicação produz dados mais rápido do que outro componente é capaz de processá-los, o buffering pode ser usado para armazenar temporariamente esses dados até que o componente que os consome esteja pronto para processá-los. Em outras palavras, o buffering é uma técnica usada para lidar com diferenças de velocidade entre componentes de uma aplicação.

Já o caching é um mecanismo de armazenamento temporário de dados usado para reduzir o tempo de acesso a dados frequentemente usados por uma aplicação. Quando uma aplicação precisa acessar um determinado recurso, ela pode verificar se uma cópia desse recurso já está armazenada em cache, em vez de acessar o recurso original diretamente. Se a cópia em cache for encontrada, ela pode ser usada em vez de buscar o recurso original, o que pode reduzir significativamente o tempo de resposta da aplicação. O caching

é usado para melhorar o desempenho geral de uma aplicação, reduzindo o tempo de acesso a dados frequentemente usados.

Em resumo, o buffering é usado para lidar com diferenças de velocidade entre componentes de uma aplicação, enquanto o caching é usado para reduzir o tempo de acesso a dados frequentemente usados. Ambos os mecanismos podem ser usados em conjunto para melhorar o desempenho de uma aplicação, mas cada um é usado para um propósito diferente.

9. Dê alguns exemplos de falhas em hardware e software que podem ou não podem ser tolerados pelo uso de redundância em um sistema distribuído. Para quais o uso de redundância, em casos apropriados, torna o sistema tolerante a falhas.

Falhas em hardware e software são comuns em sistemas distribuídos, e podem ter diferentes níveis de impacto na operação do sistema, dependendo da sua severidade e do contexto em que ocorrem. Algumas falhas podem ser toleradas pelo uso de redundância, enquanto outras não.

Exemplos de falhas em hardware e software que podem ser toleradas pelo uso de redundância incluem:

- Falhas de hardware em um servidor, que podem ser contornadas por meio da replicação dos dados e serviços em outros servidores, permitindo que a operação continue normalmente mesmo que um ou mais servidores falhem.

- Erros de software em uma aplicação, que podem ser contornados por meio da implementação de um mecanismo de detecção de erros e recuperação, que permite que a aplicação continue operando mesmo que ocorra um erro em algum componente.

- Falhas de rede, que podem ser contornadas por meio da implementação de múltiplos caminhos de comunicação entre os componentes de um sistema distribuído, permitindo que os dados e serviços possam ser roteados por outras rotas caso uma delas falhe.

Por outro lado, existem falhas em hardware e software que não podem ser toleradas pelo uso de redundância, como por exemplo:

- Falhas catastróficas em um servidor, como uma falha na fonte de alimentação, que pode causar a perda total dos dados e serviços hospedados naquele servidor. Nesse caso, a redundância não seria suficiente para recuperar os dados e serviços perdidos.

- Erros em um sistema de segurança, como uma falha em um mecanismo de autenticação, que pode permitir que um invasor obtenha acesso indevido ao sistema distribuído. Nesse caso, a redundância não seria suficiente para prevenir a falha de segurança.

- Em resumo, o uso de redundância pode tornar um sistema distribuído tolerante a falhas em alguns casos, mas não em outros. É importante avaliar cuidadosamente os riscos e impactos das falhas potenciais e escolher as técnicas de redundância adequadas para mitigá-los.

10. Considere um servidor simples que atenda a pedidos de clientes sem acessar outros servidores. Explique por que geralmente não é possível determinar um tempo limite despendido pelo servidor para responder ao pedido do cliente. O que poderia ser feito para tornar o servidor capaz de executar pedidos dentro de um tempo delimitado? É uma opção prática?

Em geral, é difícil determinar um tempo limite para o servidor responder aos pedidos do cliente em um ambiente em que o servidor não acessa outros servidores. Isso ocorre porque existem vários fatores que podem influenciar o tempo necessário para processar um pedido, como a carga do servidor, a complexidade do pedido, a capacidade de processamento do servidor e a velocidade da rede.

Além disso, o tempo necessário para executar um pedido pode variar amplamente devido a condições imprevisíveis, como interrupções na rede, falhas de hardware ou outros problemas técnicos que possam surgir.

No entanto, é possível tornar o servidor capaz de executar pedidos dentro de um tempo delimitado por meio de técnicas como limitação de recursos e balanceamento de carga. Por exemplo, é possível limitar o número de pedidos que o servidor pode processar simultaneamente para evitar sobrecarga e garantir que o servidor possa atender a todos os pedidos dentro de um tempo razoável.

Outra opção é usar técnicas de balanceamento de carga, que distribuem os pedidos entre vários servidores para que nenhum servidor fique sobrecarregado e todos possam responder aos pedidos dentro de um tempo razoável.

Embora essas técnicas possam ajudar a tornar o servidor capaz de executar pedidos dentro de um tempo delimitado, elas nem sempre são práticas. A implementação de limitação de recursos e balanceamento de carga pode ser complexa e exigir recursos adicionais, como servidores adicionais e software especializado, que podem não estar disponíveis ou serem muito caros para algumas organizações. Além disso, algumas aplicações podem não se beneficiar dessas técnicas, especialmente aquelas que envolvem processamento intensivo de dados ou outras tarefas complexas que requerem mais tempo para serem concluídas.

11. Para cada um dos fatores que contribuem para o tempo levado para transmitir uma mensagem entre dois processos sobre um canal de comunicação, discuta quais medidas poderiam ser tomadas para configurar um limite de tempo total. Por que estas medidas não são fornecidas para os sistemas distribuídos de propósito geral?

Existem vários fatores que contribuem para o tempo levado para transmitir uma mensagem entre dois processos sobre um canal de comunicação em um sistema distribuído. Alguns desses fatores são:

Latência de rede: A latência é o tempo que uma mensagem leva para viajar pela rede entre os processos. Para configurar um limite de tempo total, pode-se definir um tempo máximo que uma mensagem pode levar para chegar ao destino. Para isso, podem ser utilizadas técnicas como a configuração de timeout nas operações de comunicação ou o uso de protocolos de transporte com garantias de tempo de entrega.

Sobrecarga do sistema: A sobrecarga do sistema pode ser causada por diversos fatores, como o alto tráfego na rede, a sobrecarga dos processadores ou a falta de recursos de memória. Para evitar a sobrecarga, podem ser adotadas medidas como a configuração de limites para o tráfego de rede, o uso de algoritmos de escalonamento de processos eficientes e a alocação adequada de recursos de hardware.

Concorrência de processos: Em um sistema distribuído, vários processos podem estar competindo por recursos, o que pode afetar o tempo de resposta. Para limitar o tempo total de resposta, podem ser adotadas técnicas de controle de concorrência, como o uso de semáforos ou de monitores.

No entanto, é importante lembrar que os sistemas distribuídos de propósito geral são projetados para suportar uma ampla variedade de aplicações e cenários de uso. Portanto, não é possível fornecer medidas padrão para configurar um limite de tempo total que atenda a todas as necessidades dos usuários. Além disso, a configuração de um limite de tempo pode afetar negativamente o desempenho do sistema, caso o tempo limite seja muito baixo ou seja configurado de forma inadequada. Por isso, é importante avaliar cuidadosamente as necessidades do sistema e as especificidades de cada aplicação antes de definir limites de tempo para a transmissão de mensagens.

12. O serviço Protocolo de Tempo de Rede (NTP) pode ser usado para sincronizar relógios de computadores. Explique por que, mesmo com este serviço, não há garantias para a diferença entre dois relógios.

O Protocolo de Tempo de Rede (NTP) é amplamente utilizado para sincronizar relógios de computadores em sistemas distribuídos. No entanto, mesmo com o uso do NTP, não há garantias para a diferença entre dois relógios.

Isso ocorre porque a precisão da sincronização depende de diversos fatores, como a latência da rede, o desempenho dos servidores NTP, a precisão dos relógios locais e a interferência de outros processos no sistema. Além disso, as diferenças nos relógios podem se acumular ao longo do tempo, especialmente em sistemas distribuídos que envolvem múltiplos servidores e dispositivos.

Mesmo quando o NTP é configurado corretamente e os relógios são sincronizados regularmente, é possível que a diferença entre os relógios seja maior do que o esperado em determinados momentos, especialmente quando ocorrem condições imprevisíveis, como interrupções na rede ou sobrecarga dos servidores NTP.

Por isso, embora o NTP seja uma ferramenta útil para sincronizar relógios em sistemas distribuídos, é importante reconhecer que não há garantias para a diferença entre dois relógios. A precisão da sincronização pode ser afetada por diversos fatores, e é importante

avaliar cuidadosamente as necessidades do sistema e as especificidades de cada aplicação antes de usar o NTP para sincronização de relógios.