**Processos e Comunicação em SDs**

1. O que são Processos? O que pode ser feito para diminuir o overhead do chaveamento entre eles e melhorar o desempenho de aplicações?

|  |
| --- |
| Processos são programas em execução no sistema operacional que possuem recursos próprios, como espaço de endereçamento de memória, identificadores de processo e arquivos abertos. Eles são a base da computação multitarefa, permitindo que vários programas sejam executados simultaneamente em um sistema.  Para diminuir o overhead do chaveamento entre processos e melhorar o desempenho das aplicações, podemos utilizar threads em vez de processos (as threads compartilham o mesmo espaço de endereçamento de memória, o que significa que a mudança entre elas é mais rápida e menos custosa em termos de recursos do sistema), alocar mais recursos para os processos mais importantes, usar algoritmos de escalonamento de processos eficientes, etc... |

1. De que formas Threads podem ser implementadas? Explique.

|  |
| --- |
| As Threads podem ser implementadas de duas formas diferentes:  **Nível de Usuário:**  Chaveamento entre Threads extremamente mais rápido;  As threads rodam sobre uma camada de software que gerenciam as threads (pool de threads);  Desvantagem: uma chamada de sistema bloqueadora pode acarretar no bloqueio de todas as Threads;  **Nível de Kernel:**  O gerenciamento das threads é feito pelo kernel;  Uma chamada de sistema bloqueadora não acarreta no bloqueio de outras Threads;  Desvantagem: Chavear contexto de threads neste caso é mais caro; |

1. Como Threads podem ser utilizadas para criação de Sistemas Distribuídos eficientes? Cite um exemplo do uso de Threads em Sistemas Distribuídos e explique como elas poderiam ser utilizadas.

|  |
| --- |
| Threads podem ser utilizadas em sistemas distribuídos para realizar operações concorrentes e aumentar a eficiência do sistema. Elas são muito atraentes para utilização nesses sistemas pois facilitam a comunicação na forma de manter múltiplas conexões lógicas ao mesmo tempo e permitem chamadas bloqueantes de sistema sem bloquear o processo inteiro.  Em sistemas distribuídos, há várias tarefas que precisam ser executadas simultaneamente em diferentes máquinas, e as Threads podem ajudar a garantir que as tarefas sejam executadas de forma concorrente, maximizando o uso dos recursos disponíveis.  Um exemplo é a comunicação entre um Browser (Navegador) e um Servidor. |

1. O que você entende por Máquinas Virtuais? Quais os diferentes tipos de virtualização existentes? Explique.

|  |
| --- |
| Máquinas Virtuais são ambientes de computação isolados e independentes, criados a partir de um único conjunto de recursos físicos de computador. Uma máquina virtual é essencialmente uma emulação de um sistema de computador real, que é executado dentro de um ambiente virtualizado. Esses ambientes são criados por software de virtualização, que permite que um único servidor físico execute várias máquinas virtuais independentes, possibilitando que o usuário acesse uma simulação de computador dentro de outro. Nesse sentido, o hardware principal é dividido em vários pools de recursos, o que permite uso deles para objetivos diferentes e específicos.  Os diferentes modos de virtualização são: Virtualização total, Emulação e Paravirtualização. |

1. O que são Protocolos de Comunicação? Para que são utilizados em Sistemas Distribuídos? Cite dois exemplos de protocolos bastante utilizados e suas funções.

|  |
| --- |
| Protocolos de comunicação são um conjunto de regras, formatos de mensagens e procedimentos que governam a troca de informações entre diferentes dispositivos em uma rede de computadores. Eles definem a maneira como os dispositivos se comunicam entre si, incluindo como as mensagens são estruturadas, como as informações são transmitidas e como as respostas são tratadas.  Os sistemas distribuídos dependem de protocolos de comunicação para permitir que os diferentes componentes do sistema se comuniquem e cooperem entre si. Os protocolos são usados para garantir a integridade, segurança e confiabilidade das comunicações entre os dispositivos, e para garantir que as informações sejam transmitidas de forma eficiente e precisa.  Dois exemplos de protocolos de comunicação bastante utilizados em sistemas distribuídos são:  TCP/IP: O protocolo de Internet (TCP/IP) é um conjunto de protocolos de comunicação que são amplamente utilizados em redes de computadores. Ele define as regras para a troca de dados entre computadores em uma rede e fornece uma base para a comunicação em larga escala entre diferentes sistemas distribuídos. O TCP é responsável pela divisão de dados em pacotes, enquanto o IP é responsável pela roteamento desses pacotes pela rede.  HTTP: O protocolo de transferência de hipertexto (HTTP) é usado para a transferência de informações na World Wide Web. Ele define a maneira como os navegadores da web e os servidores da web se comunicam e permite que os usuários acessem e compartilhem informações na Internet. O HTTP define os métodos para solicitar e enviar dados, bem como as respostas para essas solicitações. |

1. Explique diferentes estratégias que podem ser utilizadas por servidores usufruindo do uso de Threads para tratar requisições de usuários.

|  |
| --- |
| Thread por solicitação: cada solicitação do usuário é processada por uma thread dedicada. Isso permite que cada solicitação seja processada independentemente das outras solicitações, o que pode melhorar a capacidade de resposta do servidor.  Pool de threads: um conjunto fixo de threads é criado e gerenciado pelo servidor. Quando uma solicitação é recebida, ela é atribuída a uma thread disponível no pool. Isso limita o número de threads que podem ser criados, o que ajuda a evitar o consumo excessivo de recursos do sistema.  Thread por conexão: cada conexão de cliente é atribuída a uma thread dedicada que processa todas as solicitações feitas por essa conexão. Isso ajuda a manter a integridade da conexão, garantindo que todas as solicitações de um determinado cliente sejam tratadas pela mesma thread.  Thread por evento: em vez de criar uma nova thread para cada solicitação, o servidor usa um modelo baseado em eventos para tratar as solicitações. Isso significa que o servidor usa uma única thread para esperar por eventos, como novas conexões de clientes ou dados recebidos de uma conexão existente. Quando um evento é acionado, o servidor executa o código associado a esse evento em uma nova thread. Isso pode ser uma estratégia eficaz para lidar com muitas conexões simultâneas com um consumo mínimo de recursos do sistema. |

1. O que são Protocolos de Middlewares? Cite dois exemplos de tipos de serviços que podem ser oferecidos por estes protocolos.

|  |
| --- |
| Protocolos de middlewares são protocolos de comunicação utilizados entre diferentes aplicações para permitir a comunicação e troca de informações entre elas. Esses protocolos são intermediários entre as camadas de aplicação e de transporte e ajudam a gerenciar e otimizar o fluxo de dados entre elas.  Dois exemplos de tipos de serviços que podem ser oferecidos por estes protocolos são:  Serviços de mensageria: os protocolos de middlewares podem oferecer serviços de mensageria que permitem que as aplicações troquem mensagens de forma assíncrona, sem necessidade de uma conexão persistente. Um exemplo de protocolo de middleware de mensageria é o MQTT, amplamente utilizado em aplicações IoT.  Serviços de integração: os protocolos de middlewares também podem oferecer serviços de integração, que permitem que diferentes sistemas interajam e troquem informações de forma padronizada e automatizada. Um exemplo de protocolo de middleware de integração é o REST, amplamente utilizado em aplicações web para troca de dados entre diferentes sistemas. |

1. Explique de forma sucinta qual o papel de cada camada do Modelo OSI e como é feita uma comunicação em rede utilizando deste modelo.

|  |
| --- |
| Camada Física: É responsável por transmitir bits brutos através de um meio de comunicação, como um cabo ou uma rede sem fio.  Camada de Enlace de Dados: É responsável por organizar os bits em quadros e transmiti-los de forma confiável entre dispositivos adjacentes na rede.  Camada de Rede: É responsável por endereçar e encaminhar pacotes de dados através de uma rede de vários dispositivos.  Camada de Transporte: É responsável por garantir que os dados sejam entregues corretamente do remetente ao destinatário, controlando o fluxo de dados, a confiabilidade e a verificação de erros.  Camada de Sessão: É responsável por estabelecer, manter e encerrar sessões entre aplicativos em diferentes dispositivos.  Camada de Apresentação: É responsável por formatar e apresentar dados para que possam ser entendidos pelo destinatário, incluindo a criptografia e compressão de dados.  Camada de Aplicação: É responsável por fornecer serviços de rede para aplicativos, permitindo que os usuários acessem serviços como e-mail, navegação na web e compartilhamento de arquivos.  A comunicação em rede usando o modelo OSI envolve a transferência de dados através de todas as camadas, começando na camada de aplicação e descendo até a camada física. Cada camada adiciona um cabeçalho ou rodapé aos dados, formando um pacote que é transmitido para a próxima camada. Quando o pacote chega ao destino, as camadas reversas desempacotam os dados e os entregam ao aplicativo destinatário. O modelo OSI ajuda a garantir que a comunicação ocorra de forma eficiente e confiável entre dispositivos em uma rede. |

1. Quais são as alternativas de comunicação oferecidas por Middlewares? Como elas funcionam? Explique.

|  |
| --- |
| Mensagens assíncronas: Esse modelo de comunicação envolve a troca de mensagens entre as aplicações ou componentes, sem que haja a necessidade de uma conexão direta e constante entre eles. Cada mensagem contém informações que serão processadas pelo receptor, que pode responder com outra mensagem. O middleware responsável pela comunicação armazena as mensagens em uma fila, garantindo que elas sejam processadas na ordem correta. Esse modelo é utilizado em sistemas distribuídos e em situações em que é necessário lidar com grande volume de dados.  Chamadas de procedimento remoto (RPC): Esse modelo de comunicação permite que uma aplicação invoque um procedimento em outra aplicação, como se estivesse executando uma função localmente. O middleware é responsável por gerenciar a chamada, encaminhando a solicitação para o servidor correto e retornando a resposta para a aplicação cliente. Esse modelo é utilizado em sistemas distribuídos em que é necessário executar operações em outras máquinas.  Corba: Common Object Request Broker Architecture (CORBA) é um modelo de comunicação que permite que objetos em diferentes plataformas e linguagens de programação se comuniquem entre si. O middleware CORBA é responsável por gerenciar a comunicação entre os objetos, fornecendo uma interface de programação padronizada e transparente para os desenvolvedores.  Web services: Esse modelo de comunicação permite que diferentes aplicações se comuniquem usando padrões web. As aplicações podem enviar e receber informações por meio de requisições HTTP, utilizando XML ou JSON como formato de dados. O middleware é responsável por fornecer a infraestrutura necessária para que as aplicações possam se comunicar de forma segura e confiável. |