Aula 1: Introdução às Arquiteturas de Sistemas Distribuídos

1. Definição e Características dos Sistemas Distribuídos:

- **Definição:**
  - Sistemas distribuídos são sistemas em que componentes de hardware e software, localizados em diferentes máquinas de uma rede, trabalham juntos como um único sistema coerente. Eles se comunicam e coordenam suas ações através da passagem de mensagens.
  - Segundo Tanenbaum, é um conjunto de computadores independentes que são apresentados ao usuário como um único sistema integrado.

- **Características:**
  - **Coleção de elementos de computação autônomos:** Cada elemento funciona de forma independente.
  - **Ausência de relógio global:** Não há um relógio central que sincronize todas as operações.
  - **Sistema único e coerente:** Os usuários veem o sistema como um todo unificado.

2. Middleware:

- **Serviços Disponibilizados:**
  - **Comunicação:** Como chamadas de procedimento remoto (RPC).
  - **Transações:** Suporte a operações atômicas.
  - **Composição de Serviços:** Integração de diferentes serviços.
  - **Fiabilidade:** Garantir a consistência e disponibilidade dos serviços.

3. Objetivos do Middleware:

- **Partilha de Recursos:** Facilitar a utilização compartilhada dos recursos do sistema.
- **Distribuição Transparente:** Tornar a distribuição invisível para aplicações e usuários.
- **Abertura:** Uso de componentes que podem ser integrados em outros sistemas.
- **Escalabilidade:** Capacidade de crescer em termos de número de usuários, volume de dados, etc.

4. Tipos de Transparência na Distribuição:

- **Acesso:** Esconde as diferenças de representação e acesso aos dados.
- **Localização:** Esconde onde os objetos estão localizados.
- **Relocalização:** Esconde a mudança de localização dos objetos.
- **Migração:** Esconde que os objetos podem ser movidos.
- **Replicação:** Esconde a existência de réplicas dos objetos.
- **Concorrência:** Esconde que os objetos são compartilhados por múltiplos usuários.
- **Falhas:** Esconde as falhas e recuperações dos objetos.

5. Arquiteturas de Sistemas Distribuídos:

- **Cluster Computing:** Vários computadores funcionando como um único sistema.
- **Grid Computing:** Conjunto de computadores distribuídos geograficamente que trabalham juntos.
- **Cloud Computing:** Fornecimento de serviços de computação através da internet.
- **Arquitetura por Camadas:** Estrutura tradicional de software com camadas de dados, interface e processamento (ex. MVC – Model-View-Controller).

6. Organização Centralizada e Descentralizada:

- **Cliente/Servidor:** Um servidor central atende a requisições de múltiplos clientes.
- **Peer-to-peer:** Todos os nós têm funcionalidades equivalentes e se comunicam diretamente.
- **Sistemas Híbridos:** Combinação de centralização e descentralização, como Edge-server e CDNs.

Aula 2: Processos, Threads e Virtualização

1. Processos e Threads:

- **Processos:**
  - Criados pelo sistema operacional.
  - Podem ter múltiplas threads.
  - Maior overhead e tempo para abrir e fechar.
  - A comunicação entre processos é mais lenta porque não compartilham memória.
- **Threads:**
  - Subprocessos dentro de um processo.
  - Compartilham memória e, portanto, são mais eficientes para ler e escrever nas mesmas variáveis.
  - Em Python, devido ao GIL (Global Interpreter Lock), as threads não trazem benefícios significativos em termos de paralelismo.

2. Exemplos em Python:

- **Multiprocessing:**

```
import multiprocessing

def count(prefix, n):
    for x in range(n):
        print(prefix, x)

if __name__ == "__main__":
    p1 = multiprocessing.Process(target=count, args=("a", 1000))
    p2 = multiprocessing.Process(target=count, args=("b", 1000))
    p1.start()
    p2.start()
    p1.join()
    p2.join()
```

Threading:

```
import threading

def count(prefix, n):
    for x in range(n):
        print(prefix, x)

if __name__ == "__main__":
    thread1 = threading.Thread(target=count, args=("a", 100))
    thread2 = threading.Thread(target=count, args=("b", 100))
    thread1.start()
    thread2.start()
    thread1.join()
    thread2.join()
```

3. Threads em Sistemas Distribuídos:

- **Servidor Multithread:** Organizado em modelo dispatcher/worker.
- **Modelo de Virtualização:**
    - **Bibliotecas:** Mesmo SO/HW, bibliotecas individuais.
    - **Chamadas ao Sistema:** Mesmo HW, visão do SO individual.
    - **Chamadas ao Hardware:** Hardware individual.

4. Virtualização:

- **Conceitos:**
    - Permite que um computador desempenhe o papel de vários através da partilha de recursos.
    - Facilita a execução de vários sistemas distintos no mesmo hardware, de forma separada.
- **História da Virtualização:**
    - **Mainframe IBM S/360:** Introdução do timeshare de aplicações.
    - **Bochs:** Ambientes emulados para debugging.
    - **VMWare e XEN:** Hypervisores que permitem isolamento completo e para-virtualização.

5. Arquiteturas de VMs:

- **Processo:** Ex. JVM (Java Virtual Machine).
- **Hosted:** Ex. VirtualBox.
- **Nativa:** Ex. ESXi.

6. Servidores:

- **Stateless:** Não guardam informação entre pedidos.
- **Stateful:** Guardam informação de sessão, aumentando a eficiência da comunicação.

7. Migração de Código e Recursos Locais:

- **Modelos de Migração:**
    - **Ligação por Identificador:** Ex. URL.
    - **Ligação por Valor:** Ex. C/Java.
    - **Ligação por Tipo:** Ex. Referências a tipos como monitor, impressora.

Aula 3: Comunicações em Sistemas Distribuídos

1. Pilha OSI:

- **Camadas de Baixo Nível:**
  - **Física:** Codificação de bits.
  - **Ligação:** Codificação de bits em frames com controle de erros e fluxo.
  - **Rede:** Roteamento de pacotes numa rede de computadores.

2. Camada de Transporte:

- **Protocolos Principais:**
  - **TCP:** Orientado à ligação, fiável, orientado aos fluxos de informação.
  - **UDP:** Comunicação não fiável (best-effort).

3. Camada Middleware:

- Facilita a partilha de serviços e protocolos comuns entre aplicações, incluindo cifragem, escalabilidade (replicação e caching), e (un)marshaling dos dados.

4. Tipos de Comunicação:

1. Transiente vs Persistente:

- **Transiente:** Mensagens podem ser descartadas se o destinatário não estiver disponível no momento do envio. Não há armazenamento intermediário.
- **Persistente:** Mensagens são armazenadas até que possam ser entregues ao destinatário, mesmo que ele não esteja disponível no momento do envio.

2. Assíncrono vs Síncrono:

- **Assíncrono:** O remetente envia uma mensagem e continua com outras tarefas, sem esperar pela resposta imediata do destinatário. A comunicação não requer que ambas as partes estejam ativas simultaneamente.
- **Síncrono:** O remetente envia uma mensagem e espera pela resposta do destinatário antes de continuar. Ambas as partes precisam estar ativas ao mesmo tempo para que a comunicação ocorra.

5. Modelos de Comunicação:

- **Cliente/Servidor:**
  - Comunicação transiente síncrona, onde cliente e servidor precisam estar ativos durante a comunicação.
  - Problemas incluem inatividade do cliente enquanto espera e a necessidade de lidar com falhas imediatamente.
- **Mensagens:**
  - Middleware orientado a mensagens permite comunicação persistente e assíncrona, com processos trocando mensagens em filas de espera.

6. Remote Procedure Call (RPC):

- **Modelo Procedimental:** Familiar para programadores, onde procedimentos podem ser executados noutra máquina através de empacotamento de parâmetros.

7. Sockets e Filas:

- **Sockets TCP e ZeroMQ:**
  - Facilita a criação de pares de comunicação assíncrona.
- **Filas:**
  - Comunicação assíncrona persistente através de middleware de filas, com operações como put, get, poll e notify.

8. Modelos de Multicasting e Algoritmos:

- **Application-level Multicasting:** Disseminação de dados através de redes overlay organizadas em árvore ou mesh.
- **Flooding e Algoritmos Epidemológicos:** Técnicas para a propagação de mensagens em sistemas distribuídos.

9. Chord e Hashing Consistente:

- **Protocolo de Pesquisa P2P:**
  - Facilita a localização de recursos e pares, utilizando tabelas de roteamento e hashing consistente para balanciamento de carga e gestão de entradas/saídas.

Aula 4: Designação (Naming)

1. Conceitos Básicos:

- **Nomes (Names):** Utilizados para referir-se a entidades num sistema distribuído.
- **Endereços (Addresses):** Nomes utilizados para referir-se ao ponto de acesso de uma entidade, como um endereço IP.
- **Identificadores (Identifiers):** Nomes desprovidos de significado, usados unicamente para comparar entidades.

2. Sistemas de Nomes (Naming Systems):

- **Nomes Planos (Flat Naming):** Identificadores únicos que identificam uma entidade. Exemplo: endereços IP, UUIDs.
- **Broadcasting:** Anunciar um ID e esperar uma resposta da entidade com seu endereço atual. Não escala além da rede local.
- **Forwarding Pointers:** Deixar uma referência à nova localização quando uma entidade se move. Exemplo: Mobile IP.

3. Distributed Hash Tables (DHT):

- **Chord:** Cada nó e entidade possuem identificadores únicos. Utiliza uma tabela de roteamento (Finger Table) para localizar chaves de forma eficiente.

4. Hierarchical Location Services (HLS):

- Organizam a localização de entidades em uma árvore hierárquica para facilitar a busca.

5. Nomes Estruturados (Structured Naming):

- **Exemplo:** Nomes de arquivos e servidores na internet.
- **Implementação de um Name Space:** Um grafo de nomes estruturados que facilita a resolução de nomes.

6. Resolução de Nomes:

- **Iterativa:** O cliente resolve o nome passo a passo.
- **Recursiva:** O servidor resolve o nome completo e retorna ao cliente.

7. Internet Domain Name System (DNS):

- Sistema de nomes de domínio que traduz nomes de domínio em endereços IP.

8. Attribute-based Naming (LDAP):

- Utiliza atributos para designar unicamente cada entrada de diretório, facilitando a busca de informações.

Aula 6: Coordenação

1. Relógios e Sincronização:

- **UTC (Universal Coordinated Time):** Baseado em relógios atômicos, usado para sincronizar relógios globalmente.
- **Sincronização de Relógios:**
  - **Precisão:** Manter o desvio entre relógios de diferentes máquinas dentro de um limite.
  - **Exatidão:** Manter o desvio entre o relógio e o tempo UTC.

2. Algoritmos de Sincronização:

- **Berkeley:** Ajusta a velocidade do tempo, não permitindo atrasar o relógio.
- **Reference Broadcast Synchronization (RBS):** Nós comparam seus tempos de recepção de uma mensagem de referência para determinar offsets relativos.

3. Relação Aconteceu-Antes (Happened-Before):

- **Definição:** Relaciona eventos com base na ordem de ocorrência, essencial para a ordenação parcial de eventos em sistemas concorrentes.

4. Relógios Lógicos:

- **Lamport:** Associam timestamps a eventos para manter a consistência de ordem.
- **Relógios Vetoriais:** Garantem que eventos relacionados causuisticamente são ordenados corretamente.

5. Algoritmos de Exclusão Mútua:

- **Baseados em Permissões:** Um nó central ou distribuído concede acesso.
- **Baseados em Token:** Um token é passado entre nós para controlar o acesso.

6. Algoritmos de Eleição:

- **Bullying:** O processo com o maior ID assume a posição de coordenador.
- **Anel:** Processos organizados em um anel lógico elegem o coordenador.

7. Correspondência de Eventos Distribuídos:

- **Publish-Subscribe:** Sistema distribui eventos para nós subscritos.

Aula 7: Consistência e Replicação

1. Razões para Replicar:

- **Fiabilidade:** Continuidade de operação após falhas.
- **Performance:** Cobrir áreas geográficas distantes e atender a altas solicitações.

2. Desafios da Replicação:

- **Consistência:** Manter a consistência entre réplicas pode ser oneroso e diminuir a escalabilidade.

3. Modelos de Consistência Data-centric:

- **Consistência Contínua:** Medida pelo grau de consistência entre réplicas.
- **Consistência Sequencial:** Todos os processos veem as operações na mesma ordem sequencial.
- **Consistência Causal:** Escritas relacionadas são vistas na mesma ordem por todos os processos.

4. Consistência Client-centric:

- **Leituras Monotônicas:** Leituras sucessivas retornam valores iguais ou mais recentes.
- **Escritas Monotônicas:** Escritas sucessivas são feitas na ordem correta.

5. Localização de Réplicas:

- **Selecionar as melhores localizações:** Minimizar a distância média aos clientes.
- **Replicação de Conteúdo:** Processos podem hospedar réplicas dinamicamente.

6. Distribuição de Conteúdos:

- **Push Updates:** Servidor inicia a atualização.
- **Pull Updates:** Cliente solicita atualizações.
- **Leases:** Contratos onde o servidor promete atualizar o cliente até a expiração do contrato.

7. Protocolos de Consistência:

- **Baseados em Primário:** Um nó primário coordena as atualizações.
- **Escrita Replicada:** Garante que operações de escrita e leitura sigam regras de quórum.

Aula 8: Tolerância a Falhas

1. Confiança (Dependability):

- **Componentes de Confiança:** Disponibilidade, Fiabilidade, Segurança, Manutenibilidade.
- **Fiabilidade vs Disponibilidade:**
    - **Fiabilidade (F(t)):** Probabilidade de um componente funcionar corretamente durante um intervalo de tempo.
    - **Disponibilidade (D(t)):** Fração média de tempo em que o componente está funcionando.

2. Terminologia de Falhas:

- **Fracasso/Falha (Failure):** Componente não cumpre suas especificações.
- **Erro:** Parte do programa que pode causar uma falha.
- **Falha/Culpa (Fault):** Causa do erro.

3. Modelos de Fracasso:

- **Crash:** Componente para de funcionar.
- **Omissões:** Componente não responde a mensagens.
- **Falha Temporal:** Responde fora dos limites temporais.
- **Falha Arbitrária:** Produz respostas arbitrárias em tempos arbitrários.

4. Redundância para Mascarar Falhas:

- **Informação:** Bits extras para recuperação de erros.
- **Temporal:** Ação pode ser repetida se algo der errado.
- **Física:** Equipamentos ou processos adicionais para suportar falhas.

5. Consenso:

- **Flooding:** Envio de comandos propostos para todos os processos e fusão dos comandos recebidos.
- **Paxos:** Algoritmo de consenso com servidores primários e backups.

6. Teorema CAP:

- **Consistência, Disponibilidade e Tolerância a Partições:** Em sistemas distribuídos, só é possível garantir duas dessas três propriedades simultaneamente.

Cloud Computing

1. Conceito:

- **Modelo NIST:** Acesso conveniente e sob demanda a recursos de computação compartilhados, configuráveis e rapidamente provisionados.

2. Modelos de Serviço:

- **IaaS (Infrastructure as a Service):** Infraestrutura virtualizada, como servidores e armazenamento. Ex: Amazon EC2, Google Cloud Platform.
- **PaaS (Platform as a Service):** Plataforma para desenvolvimento de aplicativos sem se preocupar com a infraestrutura subjacente. Ex: Google App Engine, MS Azure.
- **SaaS (Software as a Service):** Aplicativos acessados via web. Ex: Google Apps, Salesforce.

3. Vantagens:

- **Custo:** Pagar pelo uso, sem necessidade de grandes investimentos iniciais.
- **Escalabilidade:** Aumento ou redução instantânea de recursos.
- **Segurança e Confiabilidade:** Serviços com altos níveis de segurança e disponibilidade.

4. Modelos de Negócio:

- **Público, Híbrido e Privado:** Diferenças em termos de manutenção, eficiência, conformidade e novas questões emergentes.

Docker

1. Conceitos Básicos:

- **Containers vs VM's:** Containers compartilham o kernel do SO e são mais leves que VMs.
- **Docker:** Ferramenta que facilita a criação, deploy e execução de containers.

2. Componentes:

- **Image:** Conjunto de dados necessário para rodar um container.
- **Container:** Instância de uma aplicação em execução.
- **Engine:** Software que executa containers.
- **Registry:** Repositório de imagens Docker.

3. Arquitetura e Funcionamento:

- **Namespaces e Cgroups:** Isolamento de recursos e controle de uso.
- **Dockerfile:** Arquivo de configuração que define como um container deve ser construído.

4. Rede e Persistência:

- **Networking:** Tipos de rede como bridge, host e none.
- **Persistência:** Uso de mounts e volumes para armazenar dados.

5. Ferramentas Adicionais:

- **Docker Compose:** Orquestração de múltiplos containers.
- **Kubernetes:** Orquestrador de containers e mais.

Python asyncio

1. Conceito de Async:

- **Programação Concorrente:** Padrão de programação para lidar com operações assíncronas.

2. Funcionamento do Python:

- **GIL (Global Interpreter Lock):** Limitação de execução concorrente em múltiplos cores.
- **Asyncio:** Implementa tarefas assíncronas sem intervenção do SO, utilizando um único processo e uma thread.

3. Implementação de Async:

- **Funções Async:** Podem ser suspensas e resumidas.
- **Event Loop:** Responsável por gerenciar tarefas assíncronas, utilizando "cooperative multi-tasking".

4. Problemas e Soluções:

- **Tarefas Intensivas (CPU):** Necessidade de liberar o CPU periodicamente.
- **Bibliotecas Alternativas:** Para suportar async em funções que não são nativamente compatíveis.

5. Exemplos de Ferramentas:

- **NGINX, Flask, RabbitMQ, Redis, HAProxy:** Ferramentas populares em ambientes distribuídos que podem ser integradas em sistemas Python asyncio.

Perguntas de escolha múltipla

Aula 1: Introdução às Arquiteturas de Sistemas Distribuídos

1. **Qual é a característica principal de um sistema distribuído?**
   - a) Possui um relógio global para sincronização
   - b) Consiste em componentes de hardware e software que comunicam e coordenam suas ações através de uma rede
   - c) Utiliza um único processador central para todas as operações
   - d) Funciona apenas em um ambiente centralizado

2. **O que é middleware em um sistema distribuído?**
   - a) Um tipo de hardware que conecta diferentes sistemas
   - b) Software que facilita a comunicação e a gestão de dados em sistemas distribuídos
   - c) Um protocolo de segurança para redes locais
   - d) Um sistema operacional para servidores

3. **Qual das seguintes é uma vantagem do uso de middleware?**
   - a) Reduz a necessidade de segurança
   - b) Elimina a necessidade de redes
   - c) Suporta a partilha de recursos e transparência de distribuição
   - d) Garante latência zero em comunicações

Aula 2: Processos, Threads e Virtualização

4. **Qual a principal diferença entre processos e threads?**
   - a) Processos compartilham memória, enquanto threads não compartilham
   - b) Threads são mais lentas que processos
   - c) Threads compartilham memória e são mais leves que processos
   - d) Processos são gerenciados pelo sistema operacional, enquanto threads são gerenciadas pela aplicação

5. **O que é virtualização?**
   - a) A técnica de criar múltiplas réplicas de dados para backup
   - b) A criação de uma versão virtual (em vez de real) de algo, como hardware, sistemas operacionais, dispositivos de armazenamento
   - c) Um tipo de memória volátil usada em sistemas distribuídos
   - d) Uma forma de criptografia para proteger dados

6. **Qual das seguintes opções é um exemplo de Máquina Virtual?**
    - a) Docker
    - b) JVM (Java Virtual Machine)
    - c) NGINX
    - d) GitHub

Aula 3: Comunicações em Sistemas Distribuídos

7. **O que é a camada de transporte na pilha OSI?**
    - a) Garante que os dados são transmitidos de forma segura entre dois pontos
    - b) Garante a entrega de dados de ponta a ponta entre hosts
    - c) Controla os dispositivos de hardware conectados ao sistema
    - d) Garante que os dados são apresentados no formato correto

8. **O que é RPC (Remote Procedure Call)?**
    - a) Um método para garantir segurança em comunicações de rede
    - b) Um protocolo para sincronização de relógios em sistemas distribuídos
    - c) Uma forma de executar procedimentos em outra máquina como se fossem locais
    - d) Um sistema de armazenamento distribuído

9. **Qual é a principal diferença entre TCP e UDP?**
    - a) TCP é orientado à conexão e confiável, enquanto UDP é sem conexão e não confiável
    - b) TCP é mais rápido que UDP
    - c) UDP oferece garantias de entrega de dados, enquanto TCP não oferece
    - d) UDP é usado para transmissões de vídeo enquanto TCP é usado para e-mails

Aula 4: Designação (Naming)

10. **O que é um identificador em sistemas de nomes?**
    - a) Um nome usado para referir-se a um ponto de acesso de uma entidade
    - b) Um nome que é utilizado para comparar entidades unicamente
    - c) Um endereço IP de um dispositivo
    - d) Um método de comunicação entre diferentes redes

11. **O que é DNS (Domain Name System)?**
    - a) Um protocolo de segurança para sistemas distribuídos
    - b) Um sistema que traduz nomes de domínio em endereços IP
    - c) Uma técnica de virtualização para servidores
    - d) Um tipo de sistema de armazenamento
12. **Qual das seguintes opções é um exemplo de Distributed Hash Table (DHT)?**

- a) LDAP
- b) DNS
- c) Chord
- d) HTTP

Aula 6: Coordenação

13. **O que é a relação "aconteceu-antes" (happened-before)?**
    - a) Relaciona eventos com base na ordem de ocorrência em diferentes processos
    - b) Um protocolo de segurança em redes
    - c) Um método para sincronização de relógios
    - d) Um algoritmo para compressão de dados

14. **Qual das seguintes é uma técnica para manter uma visão global do comportamento do sistema?**
    - a) Criação de clusters
    - b) Relógios Lógicos
    - c) Uso de servidores dedicados
    - d) Criptografia de dados

15. **Qual algoritmo é usado para consenso em sistemas distribuídos?**
    - a) DNS
    - b) DHT
    - c) Paxos
    - d) RPC

Aula 7: Consistência e Replicação

16. **Qual é o principal objetivo da replicação em sistemas distribuídos?**
    - a) Reduzir o uso de memória
    - b) Aumentar a segurança dos dados
    - c) Aumentar a fiabilidade e a performance do sistema
    - d) Simplificar a administração de redes

17. **O que é consistência eventual?**
    - a) Garantia de que todas as réplicas convergem para o mesmo valor no tempo presente
    - b) Garantia de que todas as réplicas convergem para o mesmo valor eventualmente
    - c) Garantia de que não haverá inconsistências entre réplicas
    - d) Um método de compressão de dados em sistemas distribuídos

18. **Qual das seguintes é uma técnica de replicação de conteúdo?**
    - a) Push Updates
    - b) TCP/IP
    - c) Firewall
    - d) DNS

Aula 8: Tolerância a Falhas

19. **O que é redundância física em sistemas distribuídos?**
    - a) Adicionar bits extras aos dados para correção de erros
    - b) Repetir uma ação se algo der errado
    - c) Adicionar equipamentos ou processos extras para permitir falhas
    - d) Usar algoritmos de compressão para economizar espaço

20. **O que é o teorema CAP?**
    - a) Um sistema distribuído pode ter Consistência, Disponibilidade e Tolerância a Partições simultaneamente
    - b) Um sistema distribuído pode ter Consistência, Disponibilidade ou Tolerância a Partições, mas não todos simultaneamente
    - c) Um sistema distribuído deve ser sempre consistente
    - d) Um sistema distribuído deve ser sempre disponível

21. **Qual das seguintes falhas é detectável de forma confiável em sistemas síncronos?**
    - a) Falha por crash
    - b) Falha arbitrária
    - c) Falha temporal
    - d) Falha por omissão

Cloud Computing

22. **Qual é a vantagem principal do modelo de cloud computing?**
    - a) Elimina a necessidade de qualquer infraestrutura
    - b) Reduz custos ao pagar apenas pelo que é usado
    - c) Garante segurança absoluta contra falhas
    - d) Permite a operação sem internet

23. **Qual das seguintes é uma característica do SaaS (Software as a Service)?**
    - a) Fornece acesso a hardware virtualizado
    - b) Oferece uma plataforma para desenvolvimento de aplicativos
    - c) Aplicações acessadas via Web Services e Web 2.0
    - d) Fornece infraestrutura de rede como serviço

24. **Qual das seguintes é uma preocupação ao usar cloud computing?**
    - a) Necessidade de hardware físico
    - b) Dependência do provedor de serviços
    - c) Falta de escalabilidade
    - d) Custos iniciais altos

Docker

25. **Qual é o conceito principal do Docker?**
    - a) Gerenciar bancos de dados distribuídos
    - b) Facilitar a criação, deploy e execução de containers
    - c) Prover serviços de rede de alta velocidade
    - d) Garantir segurança de dados

26. **Qual componente do Docker armazena e entrega imagens?**
    - a) Container
    - b) Engine
    - c) Registry
    - d) Control Plane

27. **O que é um Dockerfile?**
    - a) Uma imagem de container
    - b) Um arquivo de configuração que define como construir um container
    - c) Uma ferramenta para monitorar containers
    - d) Um sistema de armazenamento de dados

Python asyncio

28. **O que é Asyncio no Python?**
    - a) Uma biblioteca para manipulação de arquivos
    - b) Um módulo para programação assíncrona
    - c) Uma ferramenta de teste de desempenho
    - d) Um método de criptografia de dados

29. **Qual o propósito do event loop no asyncio?**
    - a) Gerenciar o uso de memória
    - b) Executar funções assíncronas em ordem
    - c) Sincronizar relógios de sistema
    - d) Monitorar a segurança do sistema

30. **Qual das seguintes opções não é compatível com async por padrão?**
    - a) Socket
    - b) HTTP requests
    - c) Database operations
    - d) Arquivos de log

Respostas

Aula 1: Introdução às Arquiteturas de Sistemas Distribuídos

1. **Qual é a característica principal de um sistema distribuído?**
   - b) Consiste em componentes de hardware e software que comunicam e coordenam suas ações através de uma rede
2. **O que é middleware em um sistema distribuído?**
   - b) Software que facilita a comunicação e a gestão de dados em sistemas distribuídos
3. **Qual das seguintes é uma vantagem do uso de middleware?**
   - c) Suporta a partilha de recursos e transparência de distribuição

Aula 2: Processos, Threads e Virtualização

4. **Qual a principal diferença entre processos e threads?**
   - c) Threads compartilham memória e são mais leves que processos
5. **O que é virtualização?**
   - b) A criação de uma versão virtual (em vez de real) de algo, como hardware, sistemas operacionais, dispositivos de armazenamento
6. **Qual das seguintes opções é um exemplo de Máquina Virtual?**
   - b) JVM (Java Virtual Machine)

Aula 3: Comunicações em Sistemas Distribuídos

7. **O que é a camada de transporte na pilha OSI?**
   - b) Garante a entrega de dados de ponta a ponta entre hosts
8. **O que é RPC (Remote Procedure Call)?**
   - c) Uma forma de executar procedimentos em outra máquina como se fossem locais
9. **Qual é a principal diferença entre TCP e UDP?**
   - a) TCP é orientado à conexão e confiável, enquanto UDP é sem conexão e não confiável

Aula 4: Designação (Naming)

10. **O que é um identificador em sistemas de nomes?**
    - b) Um nome que é utilizado para comparar entidades unicamente
11. **O que é DNS (Domain Name System)?**
    - b) Um sistema que traduz nomes de domínio em endereços IP
12. **Qual das seguintes opções é um exemplo de Distributed Hash Table (DHT)?**
    - c) Chord

Aula 6: Coordenação

13. **O que é a relação "aconteceu-antes" (happened-before)?**
    - a) Relaciona eventos com base na ordem de ocorrência em diferentes processos
14. **Qual das seguintes é uma técnica para manter uma visão global do comportamento do sistema?**
    - b) Relógios Lógicos
15. **Qual algoritmo é usado para consenso em sistemas distribuídos?**
    - c) Paxos

Aula 7: Consistência e Replicação

16. **Qual é o principal objetivo da replicação em sistemas distribuídos?**
    - c) Aumentar a fiabilidade e a performance do sistema
17. **O que é consistência eventual?**
    - b) Garantia de que todas as réplicas convergem para o mesmo valor eventualmente
18. **Qual das seguintes é uma técnica de replicação de conteúdo?**
    - a) Push Updates

Aula 8: Tolerância a Falhas

19. **O que é redundância física em sistemas distribuídos?**
    - c) Adicionar equipamentos ou processos extras para permitir falhas
20. **O que é o teorema CAP?**
    - b) Um sistema distribuído pode ter Consistência, Disponibilidade ou Tolerância a Partições, mas não todos simultaneamente
21. **Qual das seguintes falhas é detectável de forma confiável em sistemas síncronos?**
    - a) Falha por crash

Cloud Computing

22. **Qual é a vantagem principal do modelo de cloud computing?**
    - b) Reduz custos ao pagar apenas pelo que é usado
23. **Qual das seguintes é uma característica do SaaS (Software as a Service)?**
    - c) Aplicações acessadas via Web Services e Web 2.0
24. **Qual das seguintes é uma preocupação ao usar cloud computing?**
    - b) Dependência do provedor de serviços

Docker

25. **Qual é o conceito principal do Docker?**
    - b) Facilitar a criação, deploy e execução de containers
26. **Qual componente do Docker armazena e entrega imagens?**
    - c) Registry
27. **O que é um Dockerfile?**
    - b) Um arquivo de configuração que define como construir um container

Python asyncio

28. **O que é Asyncio no Python?**
    - b) Um módulo para programação assíncrona
29. **Qual o propósito do event loop no asyncio?**
    - b) Executar funções assíncronas em ordem
30. **Qual das seguintes opções não é compatível com async por padrão?**
    - a) Socket