

Aplicações e Serviços de Computação em Nuvem

A computação em nuvem consiste na entrega de recursos e serviços de TI (como servidores, armazenamento, bancos de dados, redes e software) através da internet, sob demanda. Ela permite acesso remoto, escalabilidade, flexibilidade e pagamento por uso, eliminando a necessidade de manter infraestrutura física local.

- 1 A computação em nuvem oferece uma utilização eficiente de recursos de hardware (*p.ex.*, CPU, RAM, disco) através da **partilha** destes recursos entre multiplas aplicações (**multi-tenancy**). Indique **uma** implicação positiva e **uma** implicação negativa provenientes desta partilha de recursos. **Justifique** a sua resposta.

Implicação Positiva: Eficiência no uso de recursos e redução de custos

- **Justificativa:** A partilha permite que um único conjunto de recursos físicos (CPU, RAM, disco) seja utilizado por várias aplicações ou clientes, otimizando sua utilização e evitando desperdício de capacidade ociosa. Isso reduz os custos operacionais para os provedores e permite que os clientes paguem apenas pelo que usam, tornando o modelo mais econômico e sustentável.

Implicação Negativa: Risco de interferência entre aplicações (noisy neighbor)

- **Justificativa:** A partilha de recursos pode levar a situações em que uma aplicação consome uma quantidade desproporcional de recursos, impactando o desempenho das demais (conhecido como efeito "vizinho barulhento"). Por exemplo, uma aplicação que utiliza excessivamente a CPU pode causar lentidão em outras aplicações que compartilham o mesmo recurso, comprometendo a qualidade do serviço.

Em resumo, enquanto a partilha melhora a eficiência e reduz custos, ela também pode introduzir desafios de isolamento e estabilidade, exigindo mecanismos avançados de gerenciamento para mitigar os efeitos negativos.

Implicações Positivas da Partilha de Recursos (Multi-tenancy):

1. **Redução de Custos:** O compartilhamento otimiza o uso de hardware, permitindo preços mais baixos para os clientes.
2. **Eficiência no Uso de Recursos:** Recursos físicos ociosos são minimizados, maximizando o retorno do investimento no hardware.
3. **Escalabilidade Econômica:** Mais clientes podem ser atendidos sem a necessidade de novos investimentos em infraestrutura física.
4. **Manutenção Simplificada:** Centralizar os recursos facilita atualizações e manutenção pelos provedores de nuvem.
5. **Flexibilidade:** O modelo permite que múltiplos clientes utilizem o mesmo sistema, com personalização adequada a cada um.

Implicações Negativas da Partilha de Recursos (Multi-tenancy):

1. **Risco de Interferência (Noisy Neighbor):** Uma aplicação pode consumir recursos desproporcionalmente, afetando o desempenho de outras.
2. **Segurança e Privacidade:** Partilhar recursos pode aumentar o risco de violações de segurança se os isolamentos não forem bem geridos.
3. **Previsibilidade de Desempenho:** A performance pode ser menos previsível devido à concorrência pelos mesmos recursos.
4. **Complexidade na Gestão de Recursos:** Garantir o equilíbrio e a alocação justa entre clientes exige ferramentas avançadas e supervisão contínua.
5. **Dependência do Provedor:** A qualidade do serviço depende da capacidade do provedor de gerenciar eficientemente o ambiente multi-tenant.

Aplicações e Serviços de Computação em Nuvem

Teste

12 de dezembro de 2023

Duração: 1h30min

2 O componente **Docker Daemon** é responsável por gerir imagens, containers e redes num ambiente virtualizado. Indique qual a diferença entre uma **imagem** e um **container** na plataforma Docker.

Imagen

- Uma **imagem** no Docker é um **arquivo estático** que contém tudo o necessário para rodar uma aplicação: o código, as bibliotecas, as dependências, as variáveis de ambiente e as configurações de sistema.
- As imagens são **imutáveis**, ou seja, uma vez criadas, não podem ser alteradas. Em vez disso, você pode criar uma nova versão da imagem a partir de alterações ou atualizações.
- Elas servem como **modelos reutilizáveis** para criar containers. Por exemplo, uma imagem baseada no Ubuntu pode incluir um servidor web Nginx e as configurações necessárias para rodar um site.
- São armazenadas no Docker Registry (como o Docker Hub) ou em registros privados, de onde podem ser puxadas para criar containers.

Container

- Um **container** é uma **instância em execução** de uma imagem. Ele é dinâmico e representa um ambiente isolado onde a aplicação roda.
- Diferente da imagem, o container é **mutável** durante a execução, permitindo que dados sejam escritos, alterações sejam feitas ou aplicações sejam executadas. No entanto, essas mudanças não afetam a imagem original e, quando o container é encerrado, todas as alterações feitas nele são descartadas (a menos que sejam salvas em volumes).
- Cada container é executado em um **ambiente isolado**, compartilhando o kernel do sistema operacional host, mas mantendo seus próprios recursos (como sistema de arquivos, rede e processos).

Resumo da Relação:

- **Imagen:** O "plano" ou "modelo" que define como o container será construído.
- **Container:** O "produto final" em execução, criado a partir de uma imagem.

Aplicações e Serviços de Computação em Nuvem

Teste

12 de dezembro de 2023

Duração: 1h30min

3 Em vários sistemas de armazenamento distribuídos, a **gestão de dados** (*i.e.*, conteúdo) e **metadados** (*p.ex.*, localização, tamanho, permissões) de ficheiros é feita por componentes diferentes. Ainda, na maioria dos casos, estes componentes são instalados em conjuntos de servidores distintos. **Explique a vantagem** de considerar componentes independentes para a gestão de dados e metadados. **Justifique** a sua resposta.

A separação entre a gestão de dados e metadados em sistemas de armazenamento distribuídos traz vantagens significativas para o desempenho, escalabilidade e confiabilidade do sistema. Aqui está uma explicação detalhada:

1. Melhor Desempenho

- Vantagem:** A separação permite que operações relacionadas a dados (leitura/escrita de arquivos) e a metadados (consulta de localização, permissões, etc.) sejam processadas de forma independente, evitando gargalos em um único componente.
- Justificativa:** Consultas de metadados geralmente são pequenas e frequentes, enquanto operações de dados envolvem movimentação de grandes volumes de informações. Separar os componentes evita que acessos intensos a um tipo de informação (metadados ou dados) impactem o outro.

2. Escalabilidade

- Vantagem:** Componentes independentes podem ser escalados separadamente, de acordo com suas necessidades específicas.
- Justificativa:** Em um cenário de alta demanda por leitura de arquivos, os servidores de dados podem ser ampliados sem a necessidade de aumentar a capacidade dos servidores de metadados, e vice-versa. Isso reduz custos e melhora a eficiência do sistema.

3. Confiabilidade e Tolerância a Falhas

- Vantagem:** A separação facilita a implementação de redundância e recuperação em caso de falhas.
- Justificativa:** Se um servidor de dados falhar, os metadados permanecerão acessíveis para que o sistema identifique réplicas dos dados em outros servidores. Da mesma forma, se um servidor de metadados falhar, os dados permanecem intactos nos servidores de armazenamento.

4. Flexibilidade no Design

- Vantagem:** Componentes independentes podem ser otimizados para suas funções específicas.
- Justificativa:** Servidores de dados podem ser projetados para lidar com alta capacidade de armazenamento e throughput, enquanto os servidores de metadados podem priorizar baixa latência e alta disponibilidade para consultas rápidas.

Conclusão:

Separar a gestão de dados e metadados aumenta o desempenho, a escalabilidade e a confiabilidade do sistema, enquanto permite maior flexibilidade no design e na operação. Essa abordagem é fundamental para lidar com os desafios de armazenamento em larga escala em ambientes distribuídos.

Nome: Número:

ENGENHARIA INFORMÁTICA

Aplicações e Serviços de Computação em Nuvem

Teste

12 de dezembro de 2023

Duração: 1h30min

4 Dê **dois exemplos de métricas**, para além do débito e latência no processamento de pedidos, que possam ser úteis ao avaliar experimentalmente um dado sistema. **Justifique a relevância** das métricas escolhidas.

1. Taxa de Erro

- O que é: Mede a porcentagem de operações que falham em relação ao total de operações realizadas pelo sistema.
- Relevância: Avalia a confiabilidade do sistema, especialmente em cenários críticos onde falhas podem ter impactos graves. Uma baixa taxa de erro é essencial para sistemas que precisam garantir alta disponibilidade e integridade de dados, como bancos ou plataformas de e-commerce.

2. Utilização de Recursos

- O que é: Mede a porcentagem de uso de recursos, como CPU, memória, disco e rede, durante a operação do sistema.
- Relevância: Permite avaliar a eficiência do sistema. Um sistema que atinge alta performance sem utilizar todos os recursos disponíveis é considerado eficiente, enquanto a sobrecarga de recursos pode indicar gargalos ou má escalabilidade. Isso é crucial para ajustar a capacidade e otimizar custos em ambientes de produção.

1. Taxa de Requisições por Segundo (Throughput)

- O que mede: A quantidade de requisições processadas pelo sistema em um determinado intervalo de tempo.
- Relevância: Avalia a capacidade do sistema de lidar com altas cargas de trabalho.

2. Disponibilidade

- O que mede: A porcentagem de tempo que o sistema está operacional e acessível aos usuários.
- Relevância: Crucial para serviços que exigem alta confiabilidade, como sistemas bancários e plataformas online.

3. Tempo de Recuperação de Falhas (MTTR - Mean Time to Recovery)

- O que mede: O tempo médio que o sistema leva para se recuperar de uma falha.
- Relevância: Indica a eficiência do sistema em lidar com falhas e minimizar interrupções.

4. Taxa de Sucesso de Transações

- O que mede: A proporção de transações que foram completadas com sucesso em relação ao total de transações iniciadas.
- Relevância: Avalia a integridade e a confiabilidade do processamento de transações.
