Modelos de programação distribuídas

São abordagens e paradigmas que permitem que sistemas distribuídos sejam projetados e implementados de maneira eficiente;

Cada modelo tem suas próprias características, vantagens e desvantagens;

O modelo adequado é escolhido com base nas necessidades específicas do sistema.

Modelo de Comunicação de Mensagens

Componentes distribuídos trocam mensagens para coordenar atividades e compartilhar dados. Frequentemente usado em sistemas baseados em mensagens e sistemas de filas.

Características

- **Assíncrono ou Síncrono**: As mensagens podem ser enviadas e recebidas de forma assíncrona ou síncrona.
- **Desacoplamento**: Os componentes não precisam estar diretamente conectados ou saber sobre o estado interno uns dos outros.
- **Resiliência**: Melhora a resiliência do sistema, pois os componentes podem falhar sem afetar o sistema global.

Exemplos

- **RabbitMQ**: Sistema de mensagens baseado em AMQP.
- **Apache Kafka**: Plataforma de streaming distribuído para processamento de eventos.

Modelo de Computação Paralela

Divide tarefas em subtarefas que podem ser executadas simultaneamente em múltiplos processadores ou máquinas.

Características

- **Divisão de Trabalho**: As tarefas são divididas e processadas em paralelo.
- **Sincronização**: É necessário coordenar e sincronizar as subtarefas para garantir a consistência dos resultados.

Exemplos

- **MapReduce**: Modelo de programação para processamento paralelo e distribuído de grandes conjuntos de dados.
- **Apache Spark**: Framework para processamento de dados em memória.

Modelo de Serviços Web

No modelo de serviços web, serviços são disponibilizados e acessados através da rede usando protocolos de comunicação padrão, como HTTP/HTTPS.

Características

- **Interoperabilidade**: Serviços web podem ser acessados por diferentes plataformas e linguagens de programação.
- **Padronização**: Utiliza padrões como SOAP (Simple Object Access Protocol) ou REST (Representational State Transfer).

Exemplos

- **SOAP**: Protocolo baseado em XML para troca de mensagens estruturadas.
- **REST**: Arquitetura que utiliza operações HTTP e formatos de dados como JSON e XML.

Modelo de Microserviços

No modelo de microsserviços, um aplicativo é composto por pequenos serviços independentes que comunicam entre si através de APIs bem definidas.

Características

- **Desacoplamento**: Serviços são independentes e podem ser desenvolvidos, implantados e escalados separadamente.
- **Escalabilidade**: Permite escalar serviços individuais conforme necessário.

Exemplos

- **Docker**: Plataforma de contêineres para empacotar e isolar microserviços.
- **Kubernetes**: Plataforma de orquestração de contêineres para gerenciar e escalar microserviços.

Modelo de Consistência Eventual

O modelo de consistência eventual garante que, após uma atualização, todos os nós do sistema eventualmente chegarão a um estado consistente, mas não garante consistência imediata.

Características

- **Alta Disponibilidade**: Permite alta disponibilidade e escalabilidade.
- **Tolerância a Falhas**: Suporta a tolerância a falhas e recuperação.

Exemplos

- **Amazon DynamoDB**: Banco de dados NoSQL que utiliza consistência eventual para alta disponibilidade.
- **Cassandra**: Banco de dados distribuído que fornece consistência eventual e alta escalabilidade.

Modelo de Dados Distribuídos

Dados são distribuídos e gerenciados em diferentes locais, mas apresentados como um único sistema lógico.

Características

- **Transparência**: Os usuários e aplicativos interagem com o sistema de dados como se fosse um único banco de dados.
- **Gerenciamento**: Requer gerenciamento complexo para garantir a integridade e consistência dos dados.

Exemplos

- **Hadoop HDFS**: Sistema de arquivos distribuído para armazenamento de grandes volumes de dados.
- **Google Bigtable**: Sistema de banco de dados distribuído para grandes volumes de dados estruturados.

Modelo de Computação em Nuvem

O modelo de computação em nuvem oferece recursos de computação, armazenamento e rede sob demanda via Internet, permitindo a construção de sistemas distribuídos escaláveis e flexíveis.

Características

- **Elasticidade**: Recursos podem ser ajustados conforme a demanda.
- **Gerenciamento Simplificado**: Fornecido como um serviço, reduzindo a necessidade de gerenciamento de hardware.

Exemplos

- **AWS (Amazon Web Services)**: Plataforma de serviços em nuvem que inclui computação, armazenamento e serviços de banco de dados.
- **Microsoft Azure**: Plataforma de nuvem que oferece uma ampla gama de serviços para computação e dados.