Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Campus Campo Mourão
Bacharelado em Ciência da Computação

Estilos de Arquitetura de Software

Diogo Rodrigues dos Santos[1], Gustavo Zanzin Guerreiro Martins[1]

[1]Departamento Acadêmico de Computação – Universidade Tecnológica Federal do Paraná (UTFPR) – 86812-460 – Campo Mourão – PR – Brasil

{diogo.2003, gustavozanzin}@alunos.utfpr.edu.br

1. Introdução

Um *estilo de arquitetura* é um grupo de arquiteturas que compartilham características em comum. Elas não estão necessariamente atreladas à alguma tecnologia, mas algumas tecnologias são naturalmente adequadas para serem usadas em determinadas arquiteturas. A seguir, serão apresentadas algumas arquiteturas utilizadas em aplicações na nuvem.

2. Big Compute

A arquitetura de *software* do tipo *big compute* é destinada a tarefas finitas que exigem **processamento intenso** em larga-escala e/ou por longos períodos de tempo, como simulações, dinâmica de fluidos e renderização de cenas 3D. O trabalho pode ser dividido em vários (geralmente cerca de centenas até milhares) de núcleos para processamento simultâneo, quando possível. Um de seus principais desafios inclui prover milhares de núcleos de forma rápida e eficiente.

3. Big Data

Conforme a quantidade e a complexidade dos dados a serem processados e analisados por um sistema aumentam em demasia, de forma a extrapolar a capacidade dos bancos de dados tradicionais, o estilo de arquitetura de software *big data* pode ser adotado. Frequentemente, envolve tipos de trabalho como processamento em lote de de *big data* estático, processamento em tempo real de *big data* dinâmico, exploração interativa, análise preditiva e aprendizado de máquina.

Ao adotar essa abordagem, as organizações podem lidar de forma mais eficiente com **grandes volumes de dados**, possibilitando a extração de *insights* valiosos e a tomada de decisões fundamentada em estatísticas.

4. Event-Driven Architecture

Em uma arquitetura orientada a eventos, normalmente é utilizado o conceito de **produtores e consumidores**, que geram e recebem eventos, respectivamente. Os produtores são independentes dos consumidores, sendo assim, eles não sabem quantos ou quais consumidores estão captando os eventos. Fato que não ocorre, no entanto, quando se diz

respeito a **consumidores concorrentes**; cada mensagem (ou evento) é processada por apenas um consumidor. Esse consumidor é escolhido de forma a garantir que cada mensagem seja processada apenas uma vez para garantir a consistência dos dados.

Esse estilo de arquitetura pode ser utilizado através de dois modelos. O primeiro modelo é o ***publish/subscribe***, no qual os consumidores assinam os eventos e os recebem quando são publicados. Além disso, quando um evento é recebido, não podem ser reproduzidos novamente. O segundo é o modelo ***event streaming***, onde o fluxo dos eventos é armazenado em *logs*, permitindo ao consumidor a leitura de qualquer parte do fluxo.

Tal arquitetura torna-se muito útil quando o sistema envolve grande volume e alta velocidade de dados (em *IoT* por exemplo), ou em casos que vários serviços precisam processar os mesmos eventos. Uma das principais vantagens da arquitetura orientada a eventos é a facilidade de **inserção de novos consumidores** sem a necessidade de alteração dos produtores, uma vez que tais entidades são dissociadas.

5. Microservices

Uma arquitetura de microsserviços implementa **serviços autônomos** delimitando seus recursos de acordo com as regras de negócio no sistema. Cada serviço deve implementar um único recurso do sistema.

Esse estilo arquitetural utiliza um **gestor/orquestrador** que possui a responsabilidade de colocar os serviços em nós, identificar falhas, e equilibrar serviços. *Kubernetes* é um exemplo de *software* orquestrador.

Outrossim, é comum que um sistema estruturado na arquitetura de microsserviços utilize ***gateways*** que **encaminham** as chamadas dos clientes. Dessa forma, serviços podem ser refatorados ou versionados sem atualizar obrigatoriamente todos os clientes.

6. N-tier Application

Nesse estilo de arquitetura o sistema é dividido em camadas, onde cada camada pode estar em uma máquina separada. Ademais, ele subdivide-se em dois modelos arquiteturais, a saber, com **camadas fechadas** e com **camadas abertas**. O primeiro pode comunicar-se apenas com a camada imediatamente inferior. Enquanto que no segundo modelo, qualquer camada abaixo pode ser invocada.

A arquitetura de aplicação n-camadas é muito utilizada em implementações de sistemas no modelo *IaaS*, no qual temos grupos de camadas sendo executadas em um conjunto de máquinas virtuais separadas. Seu principal benefício está na portabilidade entre plataformas na nuvem e entre local e nuvem. Além disso, esse estilo é aberto a ambientes heterogêneos, como *Windows* e *Linux*, por exemplo.

7. Web-Queue Worker

Trata-se de usar uma fila de mensagens composta de um produtor e vários consumidores, tornando o processamento de grandes quantidades de recursos mais veloz. Esse tipo de arquitetura é muito encontrada no *front-end* da *Web*.

O principal desafio dessa arquitetura está em implementar produtores de maneira que seja fácil a manutenção ao passo que a aplicação cresce.

Referências

Microsoft. *Azure Architecture Center*. ***Architecture Styles***. Disponível em: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/. Acesso em: 25 de mar. de 2024.