FACULDADE DE
CIÊNCIAS E TECNOLOGIA

Sistemas de Computação em Cloud

Relatório

Raul Santos 72394
José Morgado 59457

13 de Novembro, 2024

1. Introdução

Este projeto tem como objetivo portar uma aplicação Tukano para a plataforma Azure Cloud, tornando-a numa solução escalável, rápida e altamente disponível. A aplicação Tukano é uma rede social onde utilizadores podem enviar e visualizar “shorts” - ou vídeos de curta duração - similar a plataformas como o TikTok. A portabilidade para a cloud foi alcançada através da integração de serviços PaaS da Azure, nomeadamente **Azure Blob Storage**, **Azure CosmosDB** e **Azure Redis Cache**, que substituíram a infraestrutura local utilizada anteriormente. Estes serviços foram escolhidos pela sua capacidade de otimização e escalabilidade para o nível mundial.

2. Visão Geral da Solução

2.1 Descrição da Solução Usada

A aplicação foi reestruturada de modo a fazer uso dos serviços cloud da Azure, mantendo a mesma arquitetura da solução original, com os serviços REST *Users*, *Shorts* e *Blobs*. A solução alcançada integra estes serviços com o Azure Blob Storage para armazenar os vídeos, CosmosDB para gestão de dados de utilizadores e metadados de vídeo e Redis Cache para acelerar o acesso a dados frequentemente lidos.

2.2 Serviços PaaS do Azure Utilizados

Os principais serviços da Azure implementados nesta solução foram:

- **Azure Blob Storage:** Responsável pelo armazenamento dos arquivos de vídeo. Permite replicação global e rápido acesso aos dados, cumprindo assim os nossos requisitos de escalabilidade e disponibilidade.
- **Azure Redis Cache:** Utilizado para caching de dados, o que permite um mais rápido acesso aos dados frequentemente utilizados, reduzindo a latência e diminuindo a carga nas bases de dados mais computacionalmente “caras”.
- **Azure CosmosDB:** Suporta o armazenamento de dados de utilizadores e de metadados dos shorts, suportando tanto PostgreSQL como NoSQL. Tem também diferentes modelos de dados, adequando-se a diferentes necessidades de performance.

2.3 Configuração e deployment

O funcionamento correto da aplicação *Tukano* é predicado não só na existência dos serviços azure utilizados, mas também na existência de várias variáveis de ambiente que contêm *Connection Strings*, *URLs de acesso, Passwords* e definições sobre que serviços utilizar. Estes recursos e variáveis são lançados a partir de um ficheiro java que automatiza quase tudo baseado no ficheiro disponibilizado TukanoMgt. É importante referir que o único ficheiro que

não ê gerado com este script é a base de dados CosmosDB PostgreSQL que terá de ser criada manualmente. As suas informações de ligação terão de ser colocadas no ficheiro de configuração do *hibernate* na diretoria WEB-INF/classes/.

No ficheiro tukanomgt.java existem algumas Strings que podem ser alteradas para definir quais recursos serão criados, em que regiões serão criados e quais recursos a aplicação irá utilizar. A variável DB_TYPE decide que tipo de base de dados será utilizada (CosmosDB Po e pode ter dois valores: SQL e noSQL enquanto que a variável REDIS_AVAILABLE decide se o cache Redis será utilizado pela aplicação, podendo ter dois valores também: NO e YES.

Após o script ser executado são gerados dois ficheiros por cada região na diretoria */resources/*, um ficheiro .props com as variáveis de ambiente explicitadas e um ficheiro .sh que regista as variáveis diretamente na aplicação a partir do azure cli. É recomendado executar o ficheiro .sh imediatamente após o primeiro deploy do webapp para carregar as variáveis de ambiente.

2.4 Replicação global

A aplicação em sí pode ser replicada em várias regiões dependendo das regiões definidas no script referido anteriormente, sendo as regiões *europenorth* e *westus* por padrão. Os outros recursos, com exceção do Redis e do CosmosDB PostgreSQL, também estão replicados globalmente. Notavelmente o serviço blobs necessita de uma *Connection String* distinta por cada região que implica a necessidade de variáveis de ambiente distintas por região. O cosmosDB noSQL utiliza um nível de consistência SESSION e utiliza multiple-write-regions para menor latência na escrita em regiões diferentes.

2.5 Detalhes importantes.

- Diferenças entre o CosmosDB noSQL e o CosmosDB PostgreSQL:

  O CosmosDB noSQL apresenta algumas diferenças notaveis em comparação ao seu equivalente PostgreSQL que condicionam a utilização de cada um. Notavelmente as queries SQL utilizadas em certas funções dos shorts não são imediatamente compativeis com os dois serviços, o que força uma complicação da lógica de serviço, especialmente em funções para apagar informação, visto que o noSQL não aceita queries “DELETE”.

- Containers no CosmosDB noSQL:

  O CosmosDB noSQL está dividido em dois containers, um para os Users, com chave de partição /userId e outro para os Shorts que também incluí Likes e Follows com chave de partição /shortId.

- Transações no CosmosDB noSQL:

  Não é possivel utilizar transações tradicionais com o CosmosDB noSQL. Utilizando a classe CosmosBatch é possivel obter um comportamento semelhante a uma transação

normal com uma limitação importante, a necessidade de esplicitar uma chave de partição a usar. A possibilidade de usar esta classe para transações está implementada, mas nem todas as situações onde uma transação tradicional é utilizada conseguem ser traduzidas corretamente.

3. Avaliação de Desempenho

NorthEurope, NoSQL, No Redis:
Tempo de resposta - médio: 74.2 ms; mediana: 71,5 ms
Throughput - approx. 7 requests/s
WestUS, NoSQL, No Redis:
Tempo de resposta: médio: 75,8 ms; mediana: 67,4 ms
Throughput - approx. 5 requests/s
…
(O script de deploy parou de funcionar e não durmo há demasiado tempo…)

3.1 Metodologia de Avaliação

Foram utilizados testes com a ferramenta **Artillery** para simular utilizadores virtuais a aceder à aplicação, testando a latência de cada uma dos recursos disponibilizados pelo API.

As métricas de desempenho que analisámos foram:

- **Throughput** - número de operações processadas por segundo
- **Latência** - tempo médio de resposta das operações

3.2 Comparação com a Solução Original

A solução original, tal como a nossa, não está a funcionar corretamente localmente.
Não conseguimos encontrar o motivo para tal.

3.3 Impacto do Cache Redis

4. Declaração de Utilização de Ferramentas de Inteligência Artificial

Durante o desenvolvimento deste projeto foram usadas ferramentas de inteligência artificial para efeitos de pesquisa, organização e geração de esqueleto de código.

Esta utilização de IA teve, principalmente, um papel de apoio e otimização.

5. Considerações finais

A portabilidade da TuKano para a Azure Cloud utilizando serviços PaaS revelou-se vantajosa em termos de escalabilidade e desempenho. Este projeto é uma base sólida para a expansão

da aplicação em cenários globais, onde a inclusão de mais funcionalidades e melhorias de segurança podem potencialmente aumentar o valor do sistema para uma base de utilizadores distribuída.