Padrões de arquitetura Serverless

Como o DynamoDB se encaixa na arquitetura Serverless? Sou Pete Naylor, da AWS. Obrigado pela participação.

Plataforma Serverless da AWS

Estes são os blocos de construção Serverless mais comuns. Lambda para computação, API Gateway para interfaces de microsserviço, S3 para armazenamento de objetos, DynamoDB como banco de dados operacional, SNS e SQS para mensagens e filas (desacoplamento), funções de etapa para gerenciamento de fluxo de trabalho, Kinesis e Athena para streaming e análise, e vários outros para ferramentas de desenvolvimento.

As coisas mudam rápido na AWS, portanto, fique atento a atualizações e anúncios de novos serviço.

Juntas, essas ferramentas facilitam a criação de soluções para problemas complexos com baixa sobrecarga administrativa: sem gerenciamento de servidores, escalabilidade flexível, alta disponibilidade e sem capacidade ociosa.

Streams do DynamoDB e AWS Lambda

Este é um dos facilitadores mais importantes. Usar streams do DynamoDB com triggers do Lambda para alimentar dados do DynamoDB para outros serviços, talvez como parte de um pipeline que possa incluir um data lake.

O stream é distribuído para ser dimensionado à medida que a taxa de transferência cresce, e o Lambda é dimensionado automaticamente conforme necessário para processar os dados e enviá-los para a próxima etapa.

Juntos, os streams do Lambda e do DynamoDB fornecem uma entrega confiável de eventos “pelo menos uma vez”: qualquer atividade de “gravação” pode se tornar um trigger e o Lambda pode filtrar e executar ações com base na alteração.

Consultas em arquiteturas de microsserviços

Um problema comum com o gerenciamento de dados em um ambiente de microsserviços é a proliferação de dados duplicados.

Em vez de ter um banco de dados para cada serviço (que pode não abranger todas as necessidades de consulta), um banco de dados do DynamoDB pode ser usado por todos os microsserviços, e as necessidades complexas de consulta podem ser atendidas por outros serviços mais adequados para pesquisas analíticas ou textuais, por exemplo.

A solução é separar o lado operacional (ou de comando) da solução de visualizações de consultas complexas. Este conceito é conhecido como persistência poliglota ou segregação de responsabilidade de consulta de comando.

Consultas com microsserviços

Este diagrama de arquitetura mostra o potencial. À esquerda, temos o lado operacional, com tráfego em tempo real (interativo) usando DynamoDB para alta disponibilidade, durabilidade e escalabilidade.

Os streams do DynamoDB acionam funções do Lambda, que processam adequadamente as atualizações de dados e as transmitem para outros serviços. Por exemplo, uma tabela separada do DynamoDB para um painel quase em tempo real ou ElasticSearch para consulta de texto – talvez Redshift por meio de Kinesis Firehose e S3 – para business intelligence.

Todas essas possibilidades de consulta de dados são expostas por meio de diferentes microsserviços e API Gateway. Cada serviço é dimensionado de forma independente, e os requisitos operacionais críticos são isolados de potenciais complexidades.

Exemplo de dados de séries temporais

Agora vamos considerar um exemplo de dados de séries temporais. O requisito é coletar e armazenar leituras de temperatura de potencialmente milhares de sensores. Também precisamos ser capazes de recuperar rapidamente (menos de 10 ms) uma leitura para um determinado sensor e timestamp.

Elas precisam ser mantidas por 30 dias antes de qualquer exclusão.

Implementamos isso fazendo com que os sensores coloquem as leituras em um stream do Kinesis. Configuramos uma função do Lambda que pesquisa o stream e grava as leituras em uma tabela do DynamoDB com SensorID como chave de partição e timestamp como chave de classificação.

Finalmente, usamos um atributo TTL que é essencialmente o timestamp mais 30 dias. O DynamoDB excluirá esses registros automaticamente após um mês.

Arquitetura de soluções

Aqui temos o panorama geral da solução. Na esquerda, vemos os sensores enviando as leituras para o stream do Kinesis. As funções do Lambda são lidas a partir do stream e inseridas na tabela do DynamoDB.

As métricas são emitidas para o CloudWatch e usadas em nosso monitoramento operacional. Mantemos as leituras de temperatura expiradas para possíveis análises de longo prazo. À medida que as leituras recentes expiram via TTL, usamos uma função acionada do Lambda para enviá-las para o Kinesis Firehose, que as grava no S3.

Os usuários acessam um site estático (hospedado no S3) e se autenticam através do Cognito para consultar os dados de temperatura do DynamoDB.

Considerações de custo

Como funciona a questão do custo? Com 100 mil pontos de dados por segundo, os dados armazenados no DynamoDB correspondem a cerca de 2,5 TB. As funções do Lambda estariam aproximadamente entre USD 2 e 5 mil por mês.

Os fragmentos do Kinesis para cobrir essa taxa de transação seriam mais USD 5 mil por mês. E as WCUs do DynamoDB chegariam a 50 mil USD por mês.

Parece um pouco caro... Podemos fazer melhor?

Estamos armazenando cerca de 50 B por item, mas isso é arredondado para 1 KB, uma unidade de capacidade de gravação. Isso não é muito eficiente. Estamos usando menos de 5% das WCUs pelas quais estamos pagando.

Solução:

Que tal usar um nivelamento de carga baseado em fila? Podemos agrupar vários pontos de dados em um único item, economizando em WCUs.

Usando o tamanho do lote do Lambda, podemos agrupar os pontos de dados em um único atributo de lista. Também podemos usar BatchWriteItem para melhorar nossa eficiência de conexão.

Além disso, poderíamos considerar a compactação dos dados. Ainda dependeremos de TTL para a expiração.

Custo, revisado

Qual é a economia com o design revisado? Se olharmos apenas para as WCUs do DynamoDB, a agregação de 10 pontos de dados por item economiza 90% do custo.

A diferença ao longo de um ano é de mais de USD 500 mil. Podemos economizar muito armazenando vários pontos de dados em um único item.

Considerações de escalabilidade

Quais são as considerações de escalabilidade para essa solução à medida que nossa rede de sensores cresce? Para o Kinesis, precisaríamos adicionar mais fragmentos conforme necessário. O Lambda será dimensionado automaticamente.

O DynamoDB também será dimensionado automaticamente para qualquer volume de dados e qualquer capacidade de taxa de transferência de tabela.

Espero que este exemplo traga ideias e uma compreensão do potencial do DynamoDB na arquitetura Serverless.

Avaliação

A seguir temos a avaliação. Descubra se você entendeu bem o conteúdo deste curso.

Mais uma vez, sou Pete Naylor, arquiteto de soluções especialista em NoSQL da AWS. Obrigado por participar deste curso sobre o DynamoDB e arquitetura Serverless!

Plataforma Serverless da AWS: vantagens

Sem gerenciamento de servidores

Escalabilidade flexível

Alta disponibilidade

Sem capacidade ociosa