Design para carga de trabalho uniforme

Considerações sobre design

Bem-vindo, meu nome é Pete. Com o DynamoDB, sua carga operacional é muito baixa. No entanto, é importante obter o design certo para o maior sucesso à medida que suas tabelas crescem.

O tempo usado para otimizar seu design será bem gasto. Quais são algumas das considerações?

Cargas de trabalho uniformes

Primeiro, e o mais importante, você precisa escolher uma chave de partição que resulte em uma distribuição uniforme de dados e tráfego do item através do espaço de hash. Uma boa chave de partição tem alta "cardinalidade". Isso significa que há muitos valores únicos.

Por exemplo, se você estivesse armazenando o status de usuário para um jogo móvel oferecido somente nos EUA e no Canadá, o código do país seria uma escolha muito ruim de chave de partição, pois resultaria em apenas dois valores.

O ID do usuário (UserId) seria uma escolha muito melhor!

Considere anexar algum valor calculado à chave de partição para gerar um número maior de valores exclusivos e distribuir o tráfego de forma mais uniforme.

Se você escolher uma chave de partição para a qual alguns valores obtêm muito mais tráfego do que os demais, você terá uma chave ativa, e a partição onde ela se encontra também será ativa.

A capacidade adaptável pode ajudar com desequilíbrios, mas você deve buscar o padrão mais uniforme possível (em todo o espaço de hash e ao longo do tempo) para seus dados e caso de uso.

O uso de cache (como DAX) pode ajudar a suavizar a atividade de leitura ativa, mas não gravações. Para gravações, você pode considerar o uso de nivelamento de carga baseado em fila. Algo como uma fila SQS no nível da aplicação para suavizar e amortecer picos na carga de gravação.

Dados ativos e inativos

Em seguida, considere separar os dados ativos e inativos. Alguns clientes usam um design de tabela contínua, em que talvez cada tabela represente um mês ou um trimestre de atividade. As tabelas mais antigas só recebem tráfego quando a aplicação passa por elas, podendo ter capacidades provisionadas muito menores.

Uma vez que as tabelas já não são necessárias, você pode excluí-las sem ter que pagar por muitas exclusões com WCUs. Melhor ainda, considere um nível ativo que é armazenado em algum lugar mais econômico, como o S3.

Atributos grandes

Os itens no DynamoDB são limitados a 400 kB de tamanho. Ler ou gravar objetos tão grandes resulta em atividade ativa localizada em uma única partição.

O ideal seria manter os tamanhos de itens pequenos (entre 1 kB e 4 kB). Para isso, primeiro considere armazenar objetos grandes no S3 e mantenha apenas uma referência ao objeto no item do DynamoDB como metadados. Você pode considerar esse modelo do DynamoDB servindo essencialmente como um índice para os objetos do S3.

Como alternativa, considere a compactação de objetos grandes. Armazene-os como um atributo binário e mantenha atributos de metadados menores separadamente.

Finalmente, se você realmente precisa trabalhar com objetos grandes no DynamoDB e também precisa de alto desempenho e baixa latência para esses itens, divida o objeto em blocos menores e mantenha-os em vários itens separados.

Desta forma, você pode ler e gravar em paralelo para obter melhor desempenho, e o tráfego será bem distribuído no espaço de partição.

Aqui temos um exemplo dessa estratégia: observe que há um item de metadados que rastreia os objetos em um nível alto, incluindo o número de blocos. A partir disso, a chave primária pode ser calculada para cada bloco.

Threads de aplicações separadas podem ler os blocos e, em seguida, remontar o objeto quando concluído. Se você sabe que terá uma chave muito ativa e isso é inevitável (como planejar que um candidato receba milhares de votos por segundo, por exemplo), você pode se preparar com antecedência dividindo essa chave em vários fragmentos, itens separados.

Cada gravação é atribuída a um fragmento aleatório. Uma leitura obtém todos os fragmentos e constrói o resultado resumido. Isso é conhecido como endereçamento por vetores (scatter-gather) e pode ser uma técnica muito eficaz.

Índices secundários locais

Falamos sobre índices secundários no início do curso, mas vamos resumir algumas práticas recomendadas em relação aos LSIs e GSIs.

Use índices com moderação. Lembre-se de que cada índice resultará em gravações adicionais que deverão ser pagos com WCUs. Lembre-se de que os LSIs limitarão o tamanho da coleção de uma chave de partição para aproximadamente 10 GB de dados.

Projete atributos seletivamente e aproveite índices esparsos para otimizar o consumo de taxa de transferência. E lembre-se de que os LSIs têm o mesmo tempo de vida que a tabela base.

Índices secundários globais

Os GSIs não suportam leituras fortemente consistentes, mas são muito mais flexíveis do que os LSIs. Qualquer LSI também pode ser modelado como um GSI.

GSIs são geralmente recomendados, a menos que você tenha uma necessidade justificada de suportar uma consistência forte. Mais uma vez, não crie índices desnecessários, pois eles consumirão capacidade. Tenha em mente a taxa de transferência de gravação do GSI. Você precisa ter certeza de que o GSI pode acompanhar as gravações ou sua tabela base será limitada (conhecida como contrapressão de GSI).

Você também deve ser seletivo sobre as projeções e aproveitar a indexação esparsa como faria para um LSI.

Ao planejar o GSI, saiba que ele tem os mesmos requisitos de distribuição uniforme (e alta cardinalidade) que a tabela base. As chaves ativas podem ser um problema para os GSIs assim como são para a tabela base.

As leituras do GSI são independentes da tabela base, portanto, o controle de utilização do GSI pode ser mais tolerável. Isso pode ser usado para isolar atividades como varreduras pesadas.

Bloqueio otimista com número de versão

Vamos falar sobre um padrão interessante para gerenciar a simultaneidade. Isso seria importante quando você deseja atualizar um item somente se ele não foi alterado desde a última leitura.

Use o bloqueio otimista com um número de versão para garantir que um item não foi alterado desde a última leitura.

Considere um cenário em que você lê um item da tabela AccountStatus (Status da conta) e faz um processamento. Em seguida, você define o atributo accountLocked como N porque determinou que o último login com falha ocorreu há mais de 24 horas.

É possível que o usuário tenha feito outra tentativa de login com falha entre o momento em que você leu os dados e o momento em que fez a atualização. A atualização está incorreta porque se baseia em informações obsoletas. A abordagem de bloqueio otimista resolve esse problema.

Leia o item e lembre-se do número da versão (versionNum = 0). Faça a transição de estado na memória depois de validar as informações (accountLocked = N if currentLoginTime > lastFailedLoginTime + 24 hours).

Incremente o número da versão (versionNum = 1). Grave o item com os atributos atualizados (accountLocked = N e versionNum = 1).

Use uma expressão condicional para executar uma gravação somente se o item não tiver sido alterado desde a última leitura. Se a condição falhar, reinicie da etapa 1.

Essa abordagem também é conhecida como padrão de design de leitura, modificação e gravação ou controle de simultaneidade otimista. É perfeitamente normal usar leituras eventualmente consistentes para isso.

Tabelas um-para-muitos

Se a tabela tiver itens que armazenam um grande número de valores em um atributo do tipo conjunto, como conjunto de strings ou conjunto de números, considere remover o atributo de conjunto da tabela e dividi-lo como itens separados em outra tabela.

Por exemplo, em uma tabela que armazena as discussões de um fórum e as respostas como um conjunto de strings em cada item, cada item será muito grande. É provável que o tamanho do item exceda o tamanho máximo do item no Dynamo DB.

A taxa de transferência será reduzida devido à busca desnecessária de grandes quantidades de dados, mesmo que você precise apenas de informações mínimas, como o assunto da discussão. Remova as respostas da tabela de discussão do fórum.

Crie uma tabela separada para armazenar as respostas como itens individuais. Ao analisar ainda mais, você pode encontrar uma maneira de armazenar vários tipos diferentes de itens (devido à flexibilidade do esquema) em uma única tabela, talvez com índices para suportar diferentes requisitos de consulta.

Padrões de acesso variados

Se você acessa frequentemente itens grandes em uma tabela, mas não usa os valores de atributo grandes, considere armazenar atributos menores acessados com frequência em uma tabela separada.

Por exemplo, considere a tabela Company (Empresa). Ela tem atributos grandes, como informações da empresa, declaração de missão e logotipo. Esses atributos são praticamente estáticos e raramente acessados.

Para melhorar a taxa de transferência, considere dividir o atributo de preço das ações alteradas e lidas com frequência em uma tabela separada.

Mais uma vez, analise se você realmente precisa de várias tabelas. Será que a outra tabela poderia ser modelada como um GSI com apenas a projeção do atributo de preço das ações?

Migrações e operações em massa

Vamos falar brevemente sobre a migração de dados de aplicações existentes para o DynamoDB. Normalmente, a API muda, assim como o modelo de item.

Se o tempo de inatividade for uma opção, você pode exportar os dados da origem, fazer algumas alterações e importar para o DynamoDB. Em seguida, faça com que sua aplicação passe a chamar o DynamoDB.

E se o tempo de inatividade não é uma opção e a aplicação deve permanecer ativa?

Aqui está uma lista de etapas na estratégia típica.

Criar tabela(s) do DynamoDB.

Modificar a aplicação para gravar na origem e no DynamoDB.

Executar um preenchimento.

Verificar.

Modificar a aplicação para ler do DynamoDB.

Modificar a aplicação para gravar somente no DynamoDB.

Encerrar o armazenamento de dados obsoleto.

Para exportar, transformar e importar dados (como em um preenchimento), várias opções conhecidas podem ser consideradas: AWS Data Pipeline, AWS Glue e Amazon EMR com Hive (usando o conector do DynamoDB).

Database Migration Service

Outra opção de migração que pode ser especialmente útil é o AWS Database Migration Service (DMS).

O DMS é um serviço que pode mover dados de uma origem (Cassandra, MongoDB e vários bancos de dados relacionais) para o DynamoDB.

O DMS pode ser usado para fazer uma cópia inicial do conjunto de dados completo e, em seguida, continuar atualizando tabelas do DynamoDB com as alterações em andamento. Quando você tiver segurança de que a aplicação está pronta para a troca, basta implantar uma nova versão do software que usa o SDK para se conectar ao DynamoDB.

Lembre-se de que esta é uma oportunidade para otimizar o design. É recomendável remodelar seus dados para melhor se adequar ao serviço DynamoDB, especialmente se estiver migrando de um banco de dados relacional.

Modelo de dados de carrinho de compras

Vamos dar uma olhada rápida em um exemplo muito simples, um carrinho de compras. A aplicação usava um banco de dados relacional e está migrando para o DynamoDB.

Podemos ver o modelo relacional à esquerda com duas tabelas separadas, uma listando os carrinhos e outra listando os itens nos carrinhos. Esta é a forma normalizada.

No DynamoDB, agregamos as informações em um único item em uma única tabela.

Atualização do carrinho usando OCC

Agora empregamos controle de simultaneidade otimista para garantir uma experiência agradável ao cliente.

Primeiro, usamos GetItem para recuperar o carrinho atual. Em seguida, construímos uma nova versão com o produto adicional inserido no carrinho e um número de versão incrementado.

Gravamos isso novamente na tabela de forma condicional, com base no mesmo número da versão do item lido recentemente.

Se houver uma falha devido a uma atualização provisória no carrinho, nossa aplicação lida com a exceção voltando e iniciando de novo.

Isso conclui este módulo. A seguir, como o DynamoDB se encaixa na arquitetura serverless?

Migração em tempo real

As etapas típicas de migração em tempo real incluem:

Criar tabela(s) do DynamoDB.

Modificar a aplicação para gravar na origem e no DynamoDB.

Executar um preenchimento.

Verificar.

Modificar a aplicação para ler do DynamoDB.

Modificar a aplicação para gravar somente no DynamoDB.

Encerrar o armazenamento de dados obsoleto.

Para exportar, transformar e importar dados (como em um preenchimento), várias opções conhecidas podem ser consideradas: AWS Data Pipeline, AWS Glue e Amazon EMR com o Hive (usando o conector do DynamoDB).

AWS Data Migration Service (DMS)

O DMS é um serviço que pode mover dados de uma origem (Cassandra, MongoDB e vários bancos de dados relacionais) para o DynamoDB.

O DMS pode ser usado para fazer uma cópia inicial do conjunto de dados completo e, em seguida, continuar atualizando tabelas do DynamoDB com as alterações em andamento. Quando você estiver seguro de que a aplicação está pronta para a troca, bastar implantar uma nova versão do software que usa o SDK para se conectar ao DynamoDB.

Lembre-se de que esta é uma oportunidade para otimizar o design. É recomendável remodelar seus dados para melhor se adequar ao serviço DynamoDB, especialmente se estiver migrando de um banco de dados relacional.