Operação do DynamoDB

Bem-vindo, eu sou Pete. Neste módulo, falaremos sobre como operar o DynamoDB.

Comportamento do cliente resiliente

Um primeiro passo importante para o sucesso do DynamoDB é configurar os clientes para resiliência em relação a erros. Devemos lidar com erros de maneira adequada.

Os erros de nível 400 devem ser resolvidos pelo usuário, e os erros de nível 500 serão solucionados pelo DynamoDB. Alguns erros 400 (como “Alguns parâmetros necessários estavam ausentes”) sempre causarão falha, a menos que você faça uma correção da solicitação. Para “Taxa de transferência provisionada excedida” (conhecida como controle de utilização) e todos os erros de nível 500, pode-se tentar novamente.

Tentativas de ajuste

Os kits de desenvolvimento de software da AWS têm uma lógica de repetição integrada com padrões razoáveis, mas você pode ajustá-los para seu caso de uso. Você pode definir um limite no número máximo de novas tentativas, tempos limite e estratégias de recuo exponencial (incluindo jitter).

Talvez para sua aplicação seja melhor falhar rápido e seguir para um comportamento diferente. Ou você prefere esperar mais tempo e continuar tentando?

No controle de utilização, o objetivo deve ser ir para o próximo segundo de capacidade e talvez ter sucesso na próxima tentativa.

Tente evitar o problema de “thundering herd”. Várias solicitações de repetição ao mesmo tempo depois que uma situação temporária é resolvida pode resultar em mais controle de utilização e atrasos contínuos.

Tratamento de erros em operações em lote

Você pode pensar nas operações em lote como um invólucro simples em torno de GetItem e PutItem/DeleteItem. Solicitações individuais não concluídas são retornadas para você.

Você pode implementar seu próprio loop de repetição em torno disso, para a repetição de itens individuais que não foram bem-sucedidos. Certifique-se de implementar seu próprio recuo exponencial.

Auto Scaling do DynamoDB

O DynamoDB usa o serviço Application Auto Scaling da AWS para ajustar dinamicamente a capacidade da taxa de transferência provisionada em seu nome, em resposta a padrões de tráfego reais.

Isso permite que uma tabela ou um GSI aumente a capacidade de leitura e gravação provisionada para lidar com aumentos no tráfego sem controle de utilização.

Quando a carga de trabalho diminui, o Application Auto Scaling diminui a taxa de transferência para que você não pague pela capacidade provisionada não utilizada.

RCU e WCU são gerenciadas separadamente, e você define um mínimo, um máximo e uma meta de utilização (em porcentagem).

O Auto Scaling agora é habilitado por padrão, e usá-lo em todos os lugares é altamente recomendado. Muitas tabelas têm sazonalidade em suas cargas, talvez um fluxo e refluxo regular no tráfego durante um dia útil.

Quanto mais suave o padrão de carga, mais próxima sua capacidade provisionada da sua capacidade consumida e mais eficientes as operações do DynamoDB.

Auto Scaling é uma ótima garantia para o inesperado. Às vezes os clientes percebem aumentos bastante súbitos e inesperados no tráfego. O Auto Scaling pode aumentar a capacidade provisionada para lidar com qualquer limitação, provavelmente de maneira mais rápida do que quando você recebe notificações por alarme e aumenta a capacidade provisionada.

O Auto Scaling é reativo e leva um pouco de tempo para reconhecer o padrão nas métricas. Ele não é capaz de reagir instantaneamente para cobrir um pico repentino sem algum controle de utilização. A configuração da meta de utilização pode ajudar nesse caso. Observe o histórico de consumo, avalie comportamentos de pico e defina uma meta de utilização que permita ao Auto Scaling manter um buffer apropriado.

Se você sabe que tem um grande evento chegando (talvez um novo lançamento, um grande trabalho de ingestão ou uma liquidação) e seu tráfego vai aumentar drasticamente de repente (como neste gráfico), é melhor planejar com antecedência e aumentar os valores mínimos como preparação. Depois que o evento terminar, você pode reduzir os valores novamente.

Os parâmetros de Auto Scaling podem ser alterados programaticamente ou você pode agendar atividades de Auto Scaling. Talvez você queira reduzir os valores durante a noite e nos fins de semana porque sua carga de trabalho não é grande fora do horário comercial.

Tabelas globais

Agora você pode operar tabelas do DynamoDB em várias regiões. As tabelas globais são replicadas automaticamente pelo serviço. Habilitar esse recurso permite um Acordo de Nível de Serviço (SLA) com cinco noves.

Há custo e complexidade adicionais, portanto, certifique-se de que o 9 adicional seja justificável para seu caso de uso (as tabelas do DynamoDB já são altamente disponíveis e duráveis nas regiões).

Talvez uma razão mais convincente para usar tabelas globais seja fornecer latência extremamente baixa aos clientes globais.

***Uma razão mais convincente para usar tabelas globais (para muitos) é fornecer latência extremamente baixa aos clientes globais.

Tabelas globais: atualizações concorrentes

As tabelas globais tem vários mestres e os conflitos são resolvidos por um mecanismo de "última gravação prevalece" (last-write-wins). A consistência forte entre regiões não é possível. Para leituras após gravação fortemente consistentes é preciso direcionar os clientes para ler da mesma região em que estão gravando.

Para alguns casos de uso, uma segunda região pode ser usada como destino de failover ativo em caso de um desastre regional. Quando não há alta simultaneidade para um único item em várias regiões, você pode usar o roteamento geográfico para levar clientes globais a qualquer endpoint global mais próximo.

Imagine uma aplicação onde cada item representa o perfil de um usuário. Como é improvável que o usuário esteja fazendo atualizações de vários lugares ao mesmo tempo, você pode direcionar o tráfego para a cópia mais próxima da tabela. Os usuários desfrutarão de uma experiência de baixa latência em qualquer lugar de destino.

As tabelas globais em cada região precisam ter capacidade de gravação suficiente para transportar todas as gravações globais a fim de acomodar o tráfego replicado. Tenha isso em mente ao provisionar: o Auto Scaling é altamente recomendado.

Neste slide, podemos ver um exemplo de “última gravação prevalece” (last-write-wins) com consistência eventual. Estamos operando uma tabela global em duas regiões (Réplica 1 e Réplica 2). Ela contém um item que registra o status monitorado de um microsserviço operado pela empresa.

Há monitores em execução nas duas regiões. Às 11:21 e 3 segundos, o microsserviço aparece como inativo (status “vermelho”) na região Réplica 1. Às 11:21 e 4 segundos, o microsserviço aparece como ativo (status “verde”) na região Réplica 2.

Quando a atualização de status vermelho da Réplica 1 atinge a Réplica 2, ela não substitui o status verde, porque o timestamp associado mostra que ele está obsoleto. Todas as regiões convergem para a última atualização, o status “verde”.

Expiração de itens com TTL

Se os itens na tabela perderem relevância com o tempo, os itens antigos podem expirar para manter o custo de armazenamento baixo e o consumo de RCU eficiente.

Em vez de pagar pela WCU necessária para excluir os itens, o DynamoDB pode cuidar disso gratuitamente através do recurso de vida útil (TTL). Você pode configurar um nome de atributo específico como sinalizador de expiração. Qualquer item que tenha esse atributo será elegível para expiração.

O atributo deve conter um número que representa o tempo após o qual a exclusão é permitida. Esse tempo deve estar no formato epoch. Dentro de um ou dois dias após o período de expiração, o DynamoDB excluirá o item e nenhuma WCU será consumida.

Se é importante para sua aplicação ignorar imediatamente os itens que ultrapassaram o tempo de expiração, você pode comparar o tempo de epoch recuperado e o tempo atual para qualquer item e tratá-lo conforme necessário.

Um ótimo padrão com TTL é mover itens expirando para armazenamento inativo, como o S3. Isso pode ser feito com streams. A exclusão por TTL é gravada no stream com um registro do item que foi excluído. Você pode ler essas entradas de itens expirados do stream e gravá-las no S3 (por meio do Kinesis Firehose). Usar triggers para uma função do Lambda que lida com essa transição é um padrão muito popular.

Controle de acesso

O DynamoDB está totalmente integrado ao Identity and Access Management (IAM). Você pode usar controle refinado para impedir o acesso não autorizado a tabelas, itens individuais e até atributos individuais.

A prática recomendada é aplicar o princípio do privilégio mínimo: permitir aos usuários e funções apenas o acesso estritamente necessário.

Outra prática recomendada para clientes que estão todos em uma VPC é usar um VPC Endpoint. Isso fornece um endpoint de destino do DynamoDB dentro da VPC e evita que o tráfego tenha que atravessar a internet roteada publicamente usando endereçamento público. É possível manter um maior isolamento da VPC dessa maneira.

DynamoDB Accelerator (DAX)

O DynamoDB Accelerator (DAX) fornece um cache compatível com API para as tabelas do DynamoDB, usado como um endpoint separado.

O DAX é um cluster de nós altamente disponível, acessível somente na VPC. O DAX é um cache de gravação, o que significa que itens e atualizações gravados através do cache são automaticamente disponibilizados para a próxima leitura eventualmente consistente. Leituras fortemente consistentes não são armazenadas em cache.

O uso de um cache do DAX pode diminuir drasticamente a quantidade de RCUs necessárias na tabela e suaviza cargas de leitura desequilibradas e com muitos picos. Também reduz o já rápido tempo de resposta do DynamoDB de milissegundos de um dígito para submilissegundos.

Backup e restauração

As tabelas podem ser facilmente copiadas e restauradas com o DynamoDB. Dois tipos de backups estão disponíveis: sob demanda (faz um backup sempre que solicitado) e recuperação point-in-time (PITR).

A recuperação PITR mantém uma janela contínua de 35 dias de informações sobre a tabela. Você pode fazer a recuperação a qualquer segundo dentro desses 35 dias. Os backups sob demanda são quase instantâneos, e nenhum tipo de backup consome capacidade da tabela.

A restauração é feita em uma nova tabela. Se você preferir, pode excluir a tabela original primeiro. Normalmente, os clientes fazem a restauração em uma tabela separada onde é possível visualizar os dados e compará-los com os itens atuais, talvez para reparar seletivamente alterações não intencionais que foram feitas na tabela.

Alternativamente, você pode reconfigurar clientes para acessar um nome de tabela diferente. Os tempos de restauração variam de acordo com a densidade de partição, mas a maioria das restaurações é concluída em menos de 10 horas.

Esse tempo não é dimensionado linearmente com o tamanho total da tabela. Os dados particionados são restaurados em paralelo. Para a maioria das tabelas de produção, a restauração PITR é uma escolha inteligente, que pode ser complementada com backups sob demanda para armazenamento de longo prazo.

Monitoramento e solução de problemas

Aqui estão algumas recomendações básicas para monitoramento e solução de problemas.

Primeiro, verifique o código de erro da AWS retornado pelas operações. Considere incluir esses dados nos logs da aplicação.

Certifique-se de habilitar o CloudTrail para que as operações de controle do DynamoDB (criar tabela, atualizar tabela etc.) estejam disponíveis para análise posterior.

Use as métricas do CloudWatch fornecidas pelo DynamoDB para monitorar o desempenho da tabela e defina alarmes para métricas pertinentes quando estiverem fora do intervalo aceitável.

Recomenda-se alarmes para: latência de solicitação bem-sucedida (SuccessfulRequestLatency), eventos de controle de utilização, consumo de capacidade, erros do usuário e erros do sistema.

Obrigado por participar deste módulo. Em seguida, falaremos sobre designs de tabelas no DynamoDB.

Tratamento adequado de códigos de erro 400 e 500

Gerencie códigos de erro 400 e 500 adequadamente para garantir uma experiência agradável ao cliente.

Para alguns erros 400, você deve corrigir o problema antes de enviar novamente a solicitação. Por exemplo:

Houve um problema com a solicitação.

Alguns parâmetros necessários estavam ausentes.

Para outros erros 400, você pode tentar novamente até que a solicitação seja bem-sucedida. Por exemplo:

Taxa de transferência provisionada excedida.

Para erros 500, você pode tentar novamente até que a solicitação seja bem-sucedida. Por exemplo,

Ocorreu um erro de servidor interno.

O serviço não estava disponível.

Tentativas de ajustes

Os SDKs da AWS têm uma lógica de repetição integrada, com padrões razoáveis.

Faça o ajuste para seu caso de uso a fim de minimizar a visibilidade e acelerar a recuperação para:

Limites em tentativas de repetição

Tempos limite

Recuo exponencial e jitter

Tratamento de erros em operações em lote

Pense em BatchGetItem e BatchWriteItem como invólucros simples em torno de GetItem e PutItem/DeleteItem.

Use informações retornadas sobre itens não processados (BatchGetItem: UnprocessedKeys, BatchWriteItem: UnprocessedItems) no lote para criar sua própria lógica de repetição. Certifique-se de implementar recuo exponencial em sua aplicação.

As operações em lote podem ler ou gravar itens de uma ou mais tabelas, e solicitações individuais em uma operação em lote podem falhar. O motivo mais provável para falha é que a tabela em questão não tem capacidade suficiente de leitura ou gravação provisionada.

Use as informações sobre tabelas e itens com falha para repetir a operação em lote com recuo exponencial.

Melhores práticas de monitoramento e solução de problemas

Verifique o código de erro da AWS retornado pelas operações e inclua esses dados nos logs da aplicação.

Habilite o CloudTrail para que as operações de controle do DynamoDB (criar tabela, atualizar tabela etc.) estejam disponíveis para análise posterior.

Use as métricas do CloudWatch fornecidas pelo DynamoDB para monitorar o desempenho da tabela.

Defina alarmes para métricas pertinentes fora do intervalo aceitável.