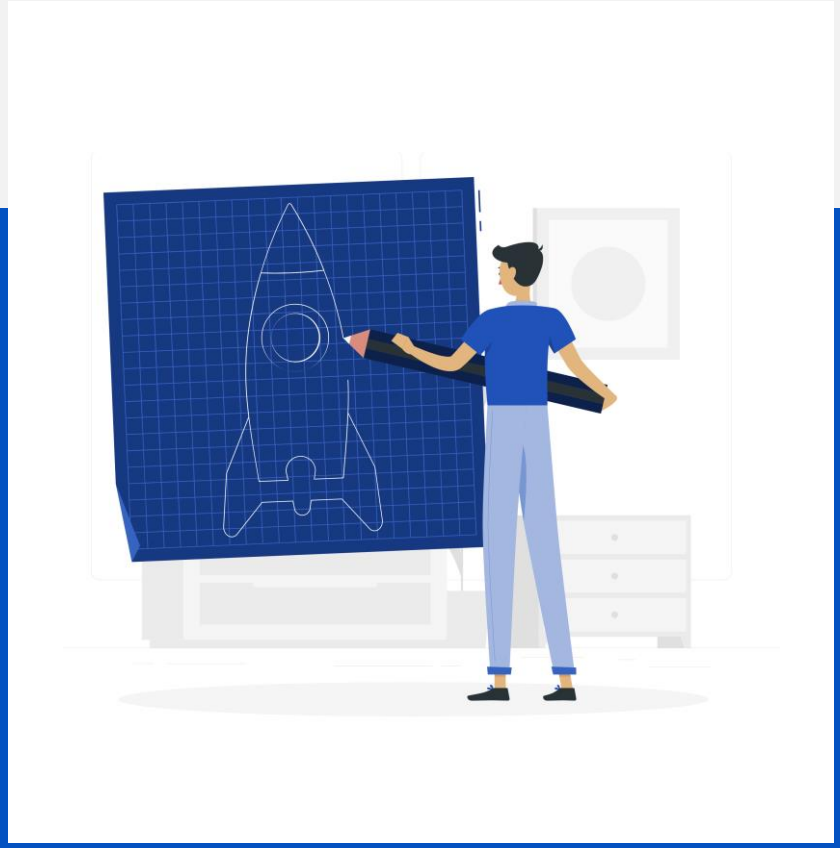
escola
britânica de
artes criativas
& tecnologia



Módulo 12

Conceitos avançados para microsserviços – Parte 3

Módulo 12

Aulas 1, 2 e 3 – Consistência eventual

- Nesta aula, falaremos de um dos maiores pilares da arquitetura de sistemas distribuídos: consistência eventual de dados.
- É um conceito relativamente difícil de se compreender e, por isso, tentarei ser o mais didático possível. Mas é algo que realmente se aprende mais na prática (também conhecido como sofrendo na própria pele) do que somente lendo a teoria.
- Com o advento de sistemas distribuídos (essencialmente, a arquitetura de microsserviços), fomos presenteados com alta performance em larga escala e baixas latências.
- Contudo, sistemas distribuídos trazem consigo alta complexidade por conta das inúmeras partes móveis. E tais sistemas também falham – mais verdade ainda para bases de dados distribuídas.

Módulo 12

Aulas 1, 2 e 3 – Consistência eventual

- Por conta de seus múltiplos componentes e diferentes níveis de abstrações, bases de dados distribuídas são muito difíceis de se depurarem.
- Por isso é tão importante entendermos as escolhas arquiteturais que temos ao construirmos sistemas distribuídos. Ao fazermos isso, fica mais fácil de manter tudo rodando.
- Uma dessas escolhas é o trade-off entre consistência forte e consistência eventual (de novo, não tem almoço grátis). Antes de mergulharmos nestes conceitos, é interessante primeiro entender como é desenhado um armazenamento distribuído.
- Basicamente, o armazenamento distribuído é alcançado através da replicação da base de dados. Esta replicação costuma ser feita através de vários nós ou servidores distintos, que se comunicam entre si através da rede e de protocolos de replicação específicos a cada base de dados.

Módulo 12

Aulas 1, 2 e 3 – Consistência eventual

- Cada réplica tem uma cópia completa dos dados e, por isso, consegue atender a todas as requisições de leitura e escrita que são feitas a elas.
- Com essa redundância, as bases de dados distribuídas fornecem sua principal vantagem: baixa latência das operações, dado que os clientes podem buscas as informações nos nós mais próximos deles.
- Além disso, também conseguimos alta performance e disponibilidade por conta da distribuição da carga entre múltiplos nós. Para aumentar a escala, basta adicionarmos mais nós.
- Tudo funciona muito bem até que problemas começam a acontecer, especialmente após a ingestão de volumes insanos de dados. A replicação pode demorar a acontecer, um ou mais nós podem cair, pode haver aumento de latência por conta do volume de dados, etc.
- Para lidar com esses cenários, temos basicamente duas escolhas de design: usar consistência forte ou consistência eventual de dados. E é exatamente isso que explicaremos nos slides a seguir.

Módulo 12

Aulas 1, 2 e 3 – Consistência eventual

- Sistemas de armazenamento distribuídos possuem três qualidades desejáveis:
  1. Consistência: qualquer dado dentro do sistema se comporta da mesma forma independente do nó onde esteja presente.
  2. Disponibilidade: o sistema sempre retornará uma resposta válida, mesmo que alguns nós não estejam rodando no momento.
  3. Tolerância a particionamento: o sistema deve se comportar de forma adequada mesmo quando partes dele são cortadas devido a falhas de rede.
- E é aqui que entra o famoso teorema CAP, que afirma que um sistema distribuído pode garantir apenas duas dessas características em todos momentos e nunca as três juntas.
- CAP vem do inglês: Consistency (consistência), Availability (disponibilidade) e Partitioning (particionamento).
- Isso significa que, quando sistemas distribuídos não funcionam como deveriam, precisamos fazer uma escolha de design sobre quais duas dessas três qualidades deveremos manter.

Módulo 12

Aulas 1, 2 e 3 – Consistência eventual

- Dado que lidar com particionamento é absolutamente essencial em bases de dados distribuídas (dado que teremos múltiplos nós), uma consequência direta do teorema CAP é que ou temos sistemas consistentes e tolerantes a particionamentos, ou temos sistemas altamente disponíveis e tolerantes a particionamentos.
- Sistemas que priorizam alta disponibilidade ao invés de consistência ficam disponíveis o tempo todo para operações de leitura e escrita. Mesmo que um nó esteja fora de sincronia por um erro de particionamento e retorne um dado antigo que não reflita o estado atual do sistema, ainda assim ele irá responder. A ideia aqui é sempre responder *alguma coisa* e não necessariamente o dado mais atual.
- Já sistemas que priorizam consistência ao invés de alta disponibilidade rejeitarão requisições de leitura e escrita caso o dado não esteja em sincronia com outros nós no sistema. Neste caso, é melhor falhar e não responder nada do que responder com dados velhos.
- Note que estas duas formas de se desenhar sistemas não são inteiramente inconsistentes. Dado que ambos toleram eventuais particionamentos, eles ainda operam com algum tipo de consistência entre os nós.
- E é aqui que veremos o conceito principal desta aula: sistemas do tipo CP usam um tipo de consistência chamado de consistência forte; já sistemas do tipo AP usam um tipo de consistência chamado de consistência eventual. Explicaremos ambos a seguir.

Módulo 12

Aulas 1, 2 e 3 – Consistência eventual

- Consistência forte: é a consistência garantida pelo teorema CAP. Os dados são consistentes através dos nós, não importa se o sistema está disponível ou se temos particionamentos de rede. Para um observador externo, todas as atualizações feitas no sistema parecem ter sido feitas de forma sequencial em um único nó. O sistema se encarregará de sincronizar todos os dados através dos nós e funcionará como se fosse composto por apenas um nó.
- Consistência eventual: prioriza disponibilidade ao invés de consistência. Vai fazer com que o sistema responda a todas as requisições de leitura e escrita mesmo que o dado esteja desatualizado naquele nó. Isso significa que poderá haver operações de leitura que trarão dados antigos e é a sua aplicação quem tem que lidar com esse cenário para evitar que o usuário final seja impactado.
- O modelo de consistência eventual garante consistência no sistema, mas nem sempre. Existe uma janela de inconsistência em que um nó pode não ter o dado mais atualizado.
- Esta janela normalmente é bem curta, na case dos milissegundos, e é determinada pela carga no sistema, o número de réplicas que a base de dados possui e na qualidade da comunicação entre os nós.
- Por fim, não devemos confundir os modelos de consistência forte e eventual com as garantias de consistência oferecidas pelas bases de dados relacionais, também conhecidas como bases de dados ACID.
- Em bases de dados ACID, as mudanças nos dados são conhecidas como transações. E estas transações possuem quatro propriedades principais: atomicidade, consistência, isolamento e durabilidade. Daí a sigla ACID.

Módulo 12

Aulas 1, 2 e 3 – Consistência eventual

- As garantias de consistência no modelo ACID significam que qualquer transação executada na base de dados a deixará em um estado válido e consistente após o commit da mesma. Com válido, queremos dizer que o dado é íntegro e obedece às restrições SQL mais comuns (not null, etc).
- Quando mencionamos consistência forte e eventual, contudo, estamos nos referindo à consistência de dados em nós (réplicas) de uma base de dados em um tempo qualquer.
- Por fim, aceitar o quão eventual a consistência pode ser é uma decisão meramente de negócio.
- Nos nossos screencasts, veremos um pouco de consistência forte e eventual na prática.

Módulo 12

Aula 4 – CDNs (Content Delivery Networks)

- Uma CDN (content delivery network, ou rede de distribuição de conteúdo) é um grupo de servidores distribuídos geograficamente que trabalham em conjunto para oferecer uma entrega rápida de conteúdo de internet, especialmente conteúdo estático, como imagens, vídeos, scripts, etc.
- Hoje, a maior parte do conteúdo de internet dos grandes sites é oferecida através de CDNs, que também podem servir para proteger seus serviços de alguns dos mais comuns ataques maliciosos (como a negação distribuída de serviço, por exemplo).

- Principais vantagens:
  1. Melhoram muito o tempo de carregamento do conteúdo estático de sites por conta de seus mecanismos de cache e de sua natureza distribuída (várias regiões).
  2. Reduzem o custo de largura de banda, principalmente por conta da já mencionada cache e de mecanismos de compactação de dados.
  3. Melhoram a disponibilidade e a redundância do conteúdo.
  4. Aprimoram a segurança dos sites e aplicações que as utilizam.

Módulo 12

Aula 4 – CDNs (Content Delivery Networks)

- Uma CDN funciona basicamente assim: servidores são colocados nos pontos de troca de tráfego na internet, também conhecidos como IXPs. Estes pontos são os principais locais onde os provedores de internet se conectam para fornecer acesso ao tráfego originado em suas diferentes redes.
- Como um provedor CDN possui conexão com estes locais de alta velocidade e altamente interconectados, o tempo de entrega dos dados acaba sendo reduzido (o que também reduz os custos com largura de banda).
- Como CDNs não costumam ser grátis e requerem uma certa configuração inicial, não teremos screencasts mostrando como usá-las. A ideia será somente falar sobre a existência das mesmas e de suas vantagens.

escola
britânica de
artes criativas
& tecnologia

Obrigado!