O segundo desafio que falamos quando falamos sobre os desafios do processamento de fluxo, foi que queremos ser capaz de processar continuamente os dados à medida que eles chegavam. Em outras palavras, queríamos criar um pipeline de dados que trabalhasse em dados de streaming. Na arquitetura de referência do GCP, o que estamos vendo agora é a segunda parte, a parte do fluxo de dados em nuvem. Onde estamos obtendo alguns dados, por exemplo, uma fonte de streaming ou talvez mesmo de uma fonte ruim e estamos criando um pipeline de dados para processar esses dados.

Neste capítulo, veremos o que é o Dataflow, essa é uma rápida recapitulação o material que foi coberto no curso no servidor menos análise de dados.

Se você não fez esse curso, encorajo-o a levá-lo porque eu não vou falar muito sobre os fundamentos do Dataflow ou no próximo capítulo, o básico da consulta. Esse curso cobre o básico em termos e conceitos. E os termos de, mas no contexto de fazer o processamento em lote. Então, neste capítulo, eu estaria focado nos conceitos de fluxo que o Dataflow fornece.

Vamos falar sobre os desafios que envolvem o processamento de fluxo e como o Dataflow nos ajuda a resolver isso.

Em seguida, criaremos um pipeline de processamento de fluxo para dados de tráfego. As mãos na atividade que estamos fazendo neste curso, e então, veremos como lidar com dados atrasados. Vamos olhar para o conceito de marcas d'água, gatilhos, acumulações e

como eles são afetados por diferentes escolhas, em termos de como você faz seus gatilhos e como você faz suas janelas. E, finalmente, faremos um laboratório no qual executamos dados para um pipeline o fluxo de dados de tráfego simulado que chega ao PubSub.

Então, quando você olha para o processamento de dados em um fluxo, se tudo que você precisava fazer era processar o elemento de dados por elemento, isso é fácil, tudo bem? Algum elemento entra, você executa algum processamento e depois o envia. Isso é muito fácil, se cada elemento é independente, isso não é um problema.

No entanto, se você precisar combinar elementos, se você precisa fazer agregações, se você precisa basicamente pegar dois elementos. E porque eles são, ambos têm a mesma chave. Eu preciso basicamente pegá-los, agregá-los, calcular sua soma,

calcule sua média, calcule uma lista dessas coisas. Agora fica um pouco mais difícil e vamos falar sobre o porquê disso. Claro, o mais difícil é se você precisa fazer composição. Se você tem dados que estão chegando e eles podem ter chaves diferentes e precisam de processamento sofisticado, em termos de processamento de dados provenientes de duas fontes diferentes.

E você precisa compô-los, adicionar informações aos dados de uma fonte usando dados computados ou complicações baseadas em outra fonte, composição.

Isso é ainda mais difícil, mas precisamos ser capazes de fazer agregações e composição em dados de fluxo.

Então, o estado da arte, se você quisesse fazer isso até muito recentemente, que você escreveria dois conjuntos de código.

Você escreveria um conjunto de códigos que manipulasse dados em lote e a ideia era que você estava muito focado em ter isso o mais preciso possível.

E então você criaria outro conjunto de códigos que processa dados de streaming,

dados que chegaram. E a suposição era que porque os dados estão chegando em um fluxo, você não pode garantir precisão, vai haver latência, haverá dados que não aparecem. Então, em essência, você calcularia algo, você calcularia um resultado especulativo, preencheria seu painel.

Mas então, uma vez por dia, você voltaria seis horas depois, talvez oito horas

mais tarde, pode ser dois dias depois, e você voltará e recompilará tudo.

E isso é obtido e chamado sua fonte de dados de ouro, e você salvaria isso.

Então você construiria dois pipelines, um pipeline com baixa latência, mas

não é muito preciso, muito especulativo.

E um segundo pipeline, que é de latência muito alta

os dados não estão disponíveis até três dias depois, mas são precisos.

E você também equilibra a taxa de transferência

você equilibra a tolerância a falhas ao criar esses dois pipelines de dados.

Então, que possuem estado de arte, você faria isso duas vezes,

às vezes você faria isso muitas vezes.

Tudo bem, você pode fazer o processamento em lote após 6 horas,

após 12 horas, após 24 horas, após uma semana, etc.

4:57

Mas como você pode ver, isso é um desperdício.

5:01

A razão pela qual você faria isso era porque se você tem dados de streaming,

dados continuamente chegando, eles vão chegar fora de ordem.

Então, todos esses pontos que são circulados

são todos os valores de dados que foram produzidos às 8 horas.

Alguns dados chegam quase imediatamente às 8 horas,

alguns outros dados chegam às 8:30, mas

Há alguns dados que chegam sete horas depois.

Então, se você vai calcular um agregado,

Qual é o total de todas as coisas que foram produzidas às 8 horas?

Qualquer resultado que você calcula às 9 horas não vai

estar correto, porque você vai ter alguns dados que estão chegando tão tarde.

5:54

E por causa disso, porque os dados que chegam continuamente podem sair de ordem,

pode ser atrasado porque a latência é um fato da vida.

É por isso que as pessoas constroem dois pipelines.

6:10

Você pode pegar seu processamento e dividi-los em janelas de tempo de processamento.

Você poderia dizer, eu vou calcular às 9 horas, certo.

Vou calcular às 9 horas, todos os dados recebidos entre 8 e 9.

Vou calcular às 10 horas, todos os dados recebidos entre 9 e 10 e assim por diante.

Mas, claro, se você fizer isso, se qualquer resultado que você calcula às 10 horas.

Não vai conter este evento tardio que chegou às 13:00.

6:42

E assim o resultado que você computou às 10 horas será incorreto.

Se você usou janelas de tempo fixo, perderá informações sobre

eventos, e não apenas sobre eventos relacionados que chegam mais tarde.

Se você se lembrar de nosso diagrama, tivemos alguns dados que chegaram quase imediatamente.

Se, por exemplo, dividirmos nosso conjunto de dados em

fluxos e depois dividimos nosso conjunto de dados em fluxos,

calculamos às 10 horas, todos os dados que chegam entre 9:00 e 10:00.

O que acontece com os dados que foram produzidos às 8 horas que chegaram às 8:45?

Vai fazer parte do horário fixo das 8 horas às 9 horas.

Não será incluído nas nove a dez janelas e isso está errado.

Porque agora qualquer soma que você disser às 8 horas, o total do número de negócios

foi de 300.000 e estranho.

E agora, qualquer informação comercial que você tenha realmente chegou

menos de uma hora atrasado é um problema ou menos que um minuto atrasado, certo.

Você pode tratar isso como horas, minutos ou segundos ou como quiser pensar

dependendo do problema que você está pensando.

O conceito é o mesmo.

8:02

Então, o que queremos é que nós queremos ter um pipeline,

8:08

que permite que você realize essas trocas,

obter precisão, lidar com baixa latência, lidar com alta latência.

O que queremos é um modelo unificado para processamento de lote e string.

E algo que pode ser executado em

ambos os dados em lote e dados de fluxo, e é isso que o Apache Beam nos fornece.

Então, o Apache Beam é um produto de código aberto baseado em vários

produtos de internet no Google chamado etc.

Mas tem sido de código aberto, é gerenciado pela fundação Apache e

Ele pode ser executado em vários tempos de execução, não apenas no fluxo de dados em nuvem.

O fluxo de dados em nuvem é uma estrutura de execução que executa pipelines do Apache Beam, mas

Há outros também.

Há tempos de execução para o Spark, há tempos de execução para o Flink e assim por diante.

Mas o que vamos falar é um tempo de execução que é fluxo de dados.

9:07

Então, com o Apache Beam, o que você tem, o conceito chave é um conceito chamado de Window.

E a ideia por trás da janela é que uma janela é um shuffle baseado no tempo.

Então, em outras palavras, você basicamente computará uma janela.

Então, aqui estão seus dados de entrada que estão sendo enviados, são ilimitados.

E então você diz, eu quero uma janela de 7-8, então vamos dizer

é do 7º minuto ao 8º minuto ou do 8º minuto ao 9º minuto.

E esta janela vai conter os dados com

o timestamp entre 8 e 9.

Portanto, a principal diferença aqui é que não é que recebemos entre 8 e

9, mas que o timestamp da mensagem, foi publicado entre 8 e 9.

Então, todos os dados que foram publicados entre 8 e 9 vão ficar

embaralhado para esta janela onde quer que ocorra no fluxo de dados de entrada.

Então estamos basicamente fazendo uma distinção

entre a hora do evento e o tempo de processamento.

10:24

OK. A hora em que o evento ocorreu e

o tempo em que podemos processar este evento.

E o tempo de processamento,

o tempo em que podemos processar o evento pode ser posterior.

Por causa da latência.

Mas vamos saber o horário do evento e

Basicamente, vamos abarcar todos os nossos eventos com base no tempo do evento.

E então nós vamos realizar o processamento.

Então, o Beam apóia essa ideia de um shuffle baseado no tempo em que

você está essencialmente tirando eventos desse fluxo de entrada,

e você está colocando-os na janela certa para calcular seus resultados.

Então, o fluxo de dados é totalmente gerenciado

ambiente de escalonamento automático para tubulações de vigas.

Então a ideia é que você escreva o código Beam, envie-o para o Dataflow,

e o Dataflow executará esse código Beam em seu nome.

E acompanhará seu fluxo de dados adicionando

novos trabalhadores do necessário, removendo os trabalhadores do necessário.

Certo, ele fornece um ambiente de execução de dimensionamento automático totalmente gerenciado.

E o que vamos nos preocupar é principalmente com o código

que precisamos escrever, que é código aberto, código do Apache Beam.

11:49

>> Quando um editor publica uma mensagem,

a mensagem está associada a um registro de data e hora.

Quando uma mensagem é lida do PubSub por um assinante,

esses dados incluem esse registro de data e hora.

O registro de data e hora é a base de todas as primitivas de janelas, incluindo marcas d'água,

disparadores e monitoramento de atraso de mensagens atrasadas.

Ao executar o processamento de fluxo, o registro de data e hora ajuda a identificar uma conta

atrasos naturais no fluxo de dados, como devido a atrasos de rede.

O Dataflow é capaz de explicar e manipular mensagens e mensagens [INAUDÍVEIS]

mensagens fora de ordem usando o carimbo de data / hora PubSub.

Por padrão, o registro de data e hora é definido durante o processo de publicação e

representa o tempo real em que a mensagem é publicada no PubSub, que é

uma hora do sistema.

Há casos em que você vai querer substituir o horário do sistema pelo seu próprio tempo

carimbo.

Você pode usar um timestamp personalizado publicando-o como um atributo PubSub,

em seguida, você informa o fluxo de dados usando o rótulo de registro de data e hora, setter.

Por exemplo, ao gravar pipelines em lote, os dados estão basicamente sendo lidos

do nosso arquivo e não há registro de data e hora inerentemente associado a ele.

Métodos de saída como TextIO não suportam saída com o registro de data e hora.

Então, para associar as mensagens aos timestamps,

você tem que criar essa associação na coleção p.

Nesse caso, em vez de usar context.output do processo,

você pode usar o contexto do processo. saída com timestamp.

No exemplo mostrado, a instância do contexto do processo é chamada c, então

mostra c.outputWithTimestamp.

Isso simula o comportamento que ocorreria no pipeline se

havia um timestamp associado a cada mensagem.

Outro exemplo é o processamento de log.

Nesse caso,

o timestamp que você deseja usar já foi gerado quando o log foi gravado.

É o registro de data e hora já incorporado no registro de log.

Neste caso, você passa o conteúdo do registro de log,

extraia o registro de data e hora e use-o no contexto do processo.outputWithTimestamp

associar a mensagem a um carimbo de data / hora de log.

O restante do pipeline tratará os dados como se os logs estivessem sendo gerados

em tempo real.

13:57

Posteriormente no código, haverá um grupo por chave e cada grupo de tempo por chave é

executado, ele executa somas de contagens, grupos e agregações.

Tudo isso é feito no contexto de uma única janela.

Então, o importante é entender que esse código

altera a ação acumulativa para uma ação em janela.

Se você fosse calcular uma média por chave, sem esse código,

seria um meio por chave em todo o conjunto de dados.

Mas assim que este método de aplicação for inserido no código, a média por chave será

ser calculado repetidamente no contexto dentro da janela de dois minutos.