Então vamos começar falando sobre alguns os desafios associados ao processamento de fluxo, em particular sobre a realização deste tipo de processamento contínuo.

Vamos usar isso para motivar algumas das nossas discussões em torno dos conceitos de janelas e triggers etc.

Então, quando você olha para o processamento de fluxo, você tem alguns desafios,

número um, no caso de dados de tráfego. Os dados de tráfego vão continuar crescendo.

Então, teremos mais sensores eles podem começar a produzir dados com mais frequência. Então o tamanho vai crescer, precisamos que seja escalável. Nossos dados de tráfego conforme os dados continuam aumentando e porque nós distribuímos sensores,

nós temos um sensor em cada estrada, cada pista. Eles estão em todo lugar.

Então, precisamos que seja tolerante a falhas. Precisamos lidar com o caso de os sensores caírem, há um problema com as redes, então os centros estão enviando uma explosão de dados porque de repente ficou online e esse tipo de coisa.

Também queremos pensar no seu modelo de programação. Precisamos ser capazes de fazer coisas como comparar o tráfego ao longo da hora anterior com o tráfego da última sexta-feira ao mesmo tempo, nesse ponto, esse processamento de fluxo ou processamento em lote?

Bem, os dados da última sexta-feira você pode pensar nisso como lote. Está tudo pronto.

Nós já temos os dados, mas o tráfego na última hora, isso é dados ilimitados, e queremos fazer essa comparação.

Porque nós queremos dizer que isso é incomum para esta hora do dia. Qualquer tipo de processamento onde você precisa de lote e fluxo, você obviamente quer fazer isso no mesmo pipeline. Então você quer um modelo de programação que lhe permita manipular lote e fluxo dentro do mesmo pipeline. Finalmente, claro, subjacente a tudo isso é, o que acontece se os dados desse sensor atrasarem?

Existem sensores que são distribuídos por toda a cidade de San Diego.

As redes não serão idênticas, os sensores não vão estar todos ao mesmo tempo.

Então, precisamos ser resilientes. Precisamos fazer nosso processamento de fluxo em uma maneira muito resistente de lidar com todos esses problemas.

Portanto, o primeiro problema é que a quantidade de dados vai crescer com o tempo.

Nós sabemos disso, certo? Porque a maioria dos dados não cresce com o tempo.

Mas, ao mesmo tempo, sabemos também que os padrões de tráfego mudam.

Há mais tráfego na hora do rush do que há à meia noite.

Então, você basicamente tem diferentes quantidades de padrões de tráfego e você tem mais sensores que estão chegando.

Então, como você lida com essa quantidade variável de dados?

Uma maneira é decidir "onde minha carga de trabalho vai ser,

esse tipo de curva, e porque o meu pico de uso é de exigir dizer 18 trabalhadores,

então eu vou provisionar um cluster de 20 trabalhadores ".

Bem, isso é um cluster fixo

e no cluster de 20 notas é um desperdício no momento em que você pode precisar.

Bem nos vales dessa curva,

você pode precisar de apenas cinco máquinas,

cinco trabalhadores nesse cluster.

Então, se cluster fixo é um desperdício de recursos.

Ao mesmo tempo você não pode dizer

"Eu vou em frente e tomo a média e dou a você

apenas 10 trabalhadores "porque se você tem 10 trabalhadores,

então você tem sob clusters provisionados metade

o tempo e os clusters de subprovisionamento

vai lhe dar sérios problemas de escalabilidade.

Então, precisamos fazer outra coisa.

Precisamos autoscalar nossos clusters.

A segunda coisa que o Dataflow nos dá é essa ideia de janelamento, certo?

Queremos basicamente dizer "Onde no tempo do evento".

Então, novamente, a chave é no tempo do evento. Onde é isso?

Apenas uma questão de que horas vamos receber os dados,

mas qual hora do evento né?

São os dados do evento correspondentes a

oito horas ou nove horas certo? O local não produz.

Então, estamos respondendo a essa questão de onde

tempo de evento precisamos calcular essa agregação.

Quando digo o número total de operações no mercado de ações,

Eu estou interessado no número total de negócios no mercado de ações que aconteceu às 08:00

horas não o número total de

mercado de ações que eu recebi às 8:00 horas,

certo, há uma grande diferença aqui.

Queremos tirar este atraso da rede desta equação e quando computamos

nossos agregados que queríamos ser em termos de tempo de evento e é isso que a janela é.

Então, o janelamento nos dará o fluxo de dados ilimitado,

e vai dividir esse fluxo de dados ilimitado em pedaços finitos de dados.

Agora você poderia fazer isso com janelas fixas certo?

Você poderia dizer, por exemplo, que todos

os dados correspondentes aos negócios que ocorreram entre as 8:00 e as 9:00 horas,

todos os negócios que aconteceram entre 9:00 e 10:00 horas e assim por diante.

Então essas são janelas fixas.

Uma maneira mais interessante de fazer isso

é dizer que eu vou olhar para negócios de hora em hora, mas eu sou

vou atualizar meu resultado a cada cinco minutos.

Essa é uma janela de correr.

Então você tem uma hora de dados, mas você está atualizando a cada cinco minutos.

OK. Então essa é uma janela deslizante.

Outro tipo de janela também é possível,

e o ponto principal disso é dizer que um tamanho não serve para todos.

Você precisa de escolhas aqui em termos de que tipo de janelas você deseja fazer.

Então, o terceiro tipo de janela que você pode querer é talvez você seja um aplicativo da web,

você está construindo um aplicativo da web e em seu aplicativo da web,

Você quer fazer sua janela com base na atividade do usuário como sessão.

Você quer dizer que o usuário faz login

o usuário faz o log in entre quantas vezes eles fizeram x,

ou quantas páginas da Web visitaram.

Então você está basicamente tendo um fluxo de dados de cliques,

e você está processando, mas sua janela não é baseada no tempo,

Agora é baseado na atividade do usuário.

Então você está definindo uma sessão e está definindo uma sucessão.

A chave aqui poderia ser cada usuário.

Então você está basicamente dizendo para esse usuário,

certo, quantos cliques eles fizeram?

Ou poderia ser baseado em domínio.

Então, dentro deste domínio,

sobre todos os usuários, onde cada janela é definida como a hora

que esse usuário em particular logado e

esse usuário em particular desconectou, o que eles fizeram?

Então, novamente, a maneira como você faz suas janelas será diferente, mas

inevitavelmente, a maneira como você define suas janelas será em termos de tempo de evento.

Vai ser em termos do tempo em que a coisa em questão

Não aconteceu o momento em que você está processando.

Então, queremos fazer essa distinção entre tempo de evento e tempo de processamento.

O uso de janelas é sobre o tempo do evento.

Então, o modelo unificado de feixe é extremamente poderoso

porque ajuda você a lidar com diferentes tipos de paradigma de processamento.

Então, você tem seus dados chegando e você

pode decidir que você deseja processar seus dados como lote.

Então, você basicamente pegará seus dados,

você vai nos criar Beam bounded.

Está tudo lá e você quer processar dados em lote,

Feixe vai deixar você fazer isso.

Você quer pegar seus dados e quer processá-los em micro lotes,

você quer apenas separá-los

e você não quer lidar com essa ideia do tempo do evento que está processando o tempo,

você só quer lidar com este tempo de processamento?

Não tem problema, você ainda pode fazer isso.

Mas mais importante e muito interessante,

O Beam permitirá que você também faça o processamento de fluxo em termos de tempo do evento.

O Beam dá a você essa flexibilidade e permite misturar e combinar.

Então, como vamos falar sobre a situação em que você quer fazer algumas coisas em lote,

e algumas coisas no fluxo

e você quer fazer no mesmo pipeline,

e isso ainda é possível.

Além disso, quando você está construindo um pipeline,

Feixe deixa você escolher entre alta latência e

baixa latência entre esta parte do meu pipeline precisa ser em lote,

pode ser de alta latência, mas eu quero que seja muito

preciso ou essa parte do meu pipeline pode ser especulativa.

Eu quero que seja baixa latência,

Basta publicar uma quantia, mas continue atualizando a soma.

Além disso, o Beam permite processar diferentes tipos de dados,

dados estruturados, dados semi-estruturados,

dados do objeto, permite executar consultas.

Então, o modelo unificado Beam é extremamente poderoso.

Eu sei que pode ser um pouco incomum

você provavelmente não viu,

mas eu recomendo fortemente que você aprenda Beam

porque é um martelo que resolve muitos problemas,

muitos problemas, e isso acontece especialmente se você executá-lo no Cloud Dataflow,

Ele faz isso de maneira totalmente gerenciada.

Então, é uma maneira muito poderosa de fazer as coisas.

Então, como um exemplo de por que você pode querer fazer

processamento em lote e processamento de fluxo no mesmo pipeline,

plugue tão sem vergonha aqui.

Então, esse gasoduto está fora do meu livro

que vem da O'Reilly sobre o Data Science no Google Cloud Platform.

Então, a ideia por trás desse pipeline é que você quer prever o atraso de um vôo.

É esse voo que está decolando,

será atrasado ou não?

E o atraso de chegada de um voo,

então você pode usar algumas coisas para prever,

para ajudar você a prever isso.

Um deles é,

meu vôo está decolando 15 minutos atrasado, mas 15 minutos atrasado,

isso quanto tempo passei no taxi,

isso é incomum ou isso é comum?

E isso acaba por depender do aeroporto em que você está.

Então, por exemplo, se você está decolando do Aeroporto LaGuardia, em Nova York,

é muito comum

seus vôos em volta que saem na noite para sentar-se na pista por 30 minutos.

Então, você quer comparar o tempo que você passou na pista,

com o tempo típico que você passa na pista de LaGuardia.

Então, é aqui que você está computando em um lote, em dados históricos.

Então, sobre os dados históricos,

você está computando o atraso típico de partida dos voos do Aeroporto LaGuardia.

Por outro lado,

você também está interessado em coisas como atrasos do tempo e a idéia é que

se você vai aterrissar em Chicago e o tempo está passando em Chicago,

todos os vôos que estão chegando em Chicago serão

atrasado e, portanto, você também é provável que seja adiada.

Mas desta vez, este é o atraso de chegada que está acontecendo atualmente em Chicago.

Então, esse é o atraso de chegada nos últimos minutos que está acontecendo em Chicago.

Então, agora estamos falando sobre o cálculo de um atraso médio em dados de streaming.

Então, temos um atraso médio em relação aos dados históricos

sobre dados globais que são batch,

estamos falando de um atraso médio em

o aeroporto de chegada que está no tempo atual,

de modo que é streaming de dados e quando estamos

basicamente tentando dizer vamos em frente e descobrir se este voo é

vai ser atrasado você está interessado em

tanto o atraso de partida em

seu aeroporto em comparação com o atraso de partida que é típico em

este aeroporto e o atraso de chegada em

o aeroporto de chegada que está completamente acabado apenas no momento atual.

Então, você precisa fazer batch e window no mesmo pipeline.

Então, aqui está apenas um exemplo disso, mas isso é

uma coisa surpreendentemente comum e

a maioria das pessoas não pensa nisso porque acredita que não é possível.

Bem, o Dataflow com o Apache Beam é possível.

Então, você quer pensar em termos de fazer

processamento em lote e fluxo de processamento no mesmo pipeline.

Então, o outro aspecto sobre o Dataflow,

quando você está executando pipelines do Apache Beam,

é que o Dataflow oferece dimensionamento automático.

Assim, seu trabalho é constantemente reequilibrado em recursos de computação e armazenamento.

Então, seus dados chegam e há uma otimização e

Autoescalonamento e rebalanceamento de peças dentro do Dataflow.

O Dataflow também está fazendo marcas d'água,

é descobrir quais máquinas estão sobrecarregadas,

e está curando eles, está monitorando,

está coletando os logs,

e está enviando tudo isso para você.

Então, o que o Dataflow oferece é uma estrutura de execução para um pipeline de feixe.

Então, você pode escrever um pipeline de feixe e executá-lo em qualquer lugar, mas é uma boa ideia executá-lo em

um lugar onde você sabe que todos os seus pipelines serão executados muito bem.

Então, linha de fundo, então,

O processamento de fluxo é muito mais fácil com o Dataflow.

Como o tamanho é importante e o Dataflow faz o dimensionamento automático e faz o Mapreduce,

permite lidar com volumes crescentes variáveis.

Porque você precisa de tolerância a falhas,

é importante que os recursos sejam implantados sob demanda

e esse trabalho é redistribuído entre os trabalhadores,

novos trabalhadores são adicionados, se necessário.

O modelo de programação que o Apache Beam oferece a você

Ele oferece a capacidade de lidar com o lote e o fluxo.

Finalmente, podemos lidar com todos os problemas associados dados ilimitados, dando-lhe uma boa implementação escalável do Windows,

triggers, de processamento incremental, etc. Então, vamos ver como fazemos isso no código do Apache Beam.