Então, o que é o Bigtable? Então, Bigtable novamente voltando ao nosso gráfico,

Bigtable é sobre o caso de dados de streaming quando o BigQuery não é suficiente.

Então, isso é novamente um caso de uso de análise de dados e

esse é o caso de uso sobre o qual estamos falando neste curso.

Estamos falando sobre streaming de dados neste curso,

então estamos falando em usar o Bigtable

quando a taxa de transferência precisa, há mais do que o BigQuery pode

manipular ou se suas necessidades de latência são quando você vai precisar de latências menores,

latências de milissegundos quando o BigQuery só pode fornecer segundas latências.

Então, com o BigQuery, o BigQuery é fácil, é barato

sua latência é da ordem de segundos,

você obtém cerca de 100.000 linhas por segundo de streaming.

Bigtable tem baixa latência, é muito alto rendimento

é mais caro que o BigQuery porque você está pagando

para o número de nós do Bigtable que você está executando.

Então, se você está executando um cluster que é dez nós, por exemplo,

você pode obter 100.000 consultas por segundo com seis milissegundos de latência.

Então, latência muito baixa, taxa de transferência muito alta.

Você paga pelo número de clusters,

esses clusters são reservados para você.

Então, nesse gráfico, Bigtable vem aqui,

trata-se de ter dados estruturados para análise de dados,

não analítica de dados de transações

que você precisa de latência de milissegundos e taxa de transferência muito alta para.

Então Bigtable é grande

é rápido, é auto-escala

e não é SQL.

Então é grande. Quão grande?

Você pode lidar com dados com mais de um terabyte.

Esses dados podem ser semi-estruturados ou podem ser estruturados.

Isso realmente significou para dados que está mudando muito rápido que tem

um throughput muito alto e você usa quando não precisa de transações,

quando você não precisa de forte semântica relacional.

Então, para que as pessoas usam o Bigtable?

Ele tende a ser muito usado para dados de séries temporais, dados financeiros,

dados do sensor, dados com essa ordem natural em termos de tempo.

Ele também é usado para operações em lote assíncronas de processamento em tempo real,

e cada vez mais está se acostumando com dados envolvendo algoritmos de aprendizado de máquina,

especialmente algoritmos de aprendizado de máquina que fazem treinamento contínuo.

Então, com a tabela Cloud você tem disponibilidade global,

é como pubsub novamente.

Então, você basicamente tem disponibilidade global,

você pode comprar serviços e dados onde quiser.

Mas, ao contrário do pubsub, ao contrário do BigQuery,

não é livre de cluster.

Então, você está pensando em termos de um cluster de nós com o Bigtable.

Bigtable como tudo no GCP,

seus dados são criptografados no voo

em repouso, no fio,

e você tem controle total dos dados,

você obtém o gerenciamento de acesso à identidade que é comum à plataforma,

e você obtém armazenamento redundante com dimensionamento automático.

Então, todos os dados que você colocou no Bigtable são duráveis,

é replicado e você pode ter acesso a ele.

Assim, como o BigQuery, o Bigtable também separa computação e armazenamento.

O que você quer dizer com isso? Eu não acabei de dizer que o Bigtable usa clusters?

Bem, ele usa clusters, mas esses clusters contêm apenas ponteiros para os dados,

eles não contêm os dados em si.

Então, os clusters consistem em nós,

esses nós contêm os metadados,

os dados em si permanecem no Colossus,

ele permanece no Google Cloud Storage.

Então, pensamos em termos de dados armazenados em linhas contíguas,

então você basicamente tem linhas de dados e aquelas linhas de dados que são armazenados contíguos,

nós falamos deles como comprimidos.

Assim, todos esses tablets de dados são armazenados no GCS e

O que os nós contêm é essencialmente ponteiros para esses comprimidos de dados.

Então, sempre que o cliente quiser fazer o processamento,

eles basicamente dizem aqui é o processamento que eu quero fazer e os nós basicamente realizar

o processamento e os dados ficam

embaralhado nesses nós para basicamente fazer o cálculo.

Mas quando os nós lêem os dados,

eles essencialmente lêem linhas contíguas de dados.

Isso será importante quando pensarmos no design do Bigtable,

como otimizar o desempenho do Bigtable.

Então agora que olhamos para o design do Bigtable,

Vejamos como inserir dados no Bigtable.

Para ingestar dados no Bigtable,

a primeira coisa é que precisamos criar um cluster Bigtable e a maneira como você cria

um cluster Bigtable é que você pode ir para

a interface do usuário da web e você pode criar um cluster,

Você pode usar o gcloud para criar o cluster.

Então, neste caso, estamos mostrando apenas o gcloud.

Assim, as instâncias do gcloud Bigtable criam

os ajustes INSTANCE e, em seguida, base, especificam qual é o seu tipo de instância.

Então, novamente, aqui estaremos especificando quais são seus trabalhadores,

você quer N\_one,

oito CPU quanta memória,

etc e, em seguida, você basicamente pode fornecer seu tipo de armazenamento de cluster,

lá, por padrão, são discos SSD porque são mais rápidos, mas você pode escolher.

Mas uma vez que você tenha seu cluster,

você sempre pode adicionar nós,

você pode remover nós,

e você pode fazer isso sem qualquer tempo de inatividade.

Então, você tem um cluster que você criou com três nós,

e você pode adicionar mais dois nós.

A razão pela qual você pode fazer isso

é que os próprios nós do cluster não guardam os dados,

eles só possuem metadados.

Então, tudo o que acontece quando você adiciona

novos clusters é que alguns metadados são copiados. É barato.

Então, você pode adicionar nós,

você pode remover nós sem ter nenhum tempo de inatividade.

Depois de ter sua instância do Bigtable,

você pode criar sua mesa,

e isso estou mostrando em Python,

você usa um cliente do Bigtable e diz "pegue-me a instância" e crie uma tabela,

e então você começa a trabalhar com uma mesa.

Mais comumente, em vez de usar o Python como estamos fazendo aqui,

onde estamos basicamente criando uma tabela e especificando o ID da coluna,

especificando o valor e comprometendo-o,

mais comumente você faria isso não do Python, mas do Dataflow.

Então, se você estiver usando o Dataflow,

então você estaria essencialmente usando BigTable IO para se comprometer com isso.

Então, se você quiser transmitir dados para o Bigtable usando o Dataflow,

você basicamente pegaria ou criaria a tabela e lembraria que no BigQuery,

o que fizemos foi que pegamos nossos dados e os convertemos

nas linhas da tabela e, em seguida, transmitimos com essas linhas da tabela para o BigQuery.

No caso do Bigtable,

o que você vai converter seu objeto em

você vai convertê-los em coisas chamadas Mutations,

os trocadores, que você vai fazer e depois

você vai escrever essa lista de mutações para o Bigtable.

Então, para criar a tabela essencialmente, você vai autenticar no GCP.

Tendo autenticado contra o GCP,

você irá em frente e criará sua mesa.

Então, isso agora cria uma tabela e então você pode ir em frente e criar uma mutação.

Então, neste caso, estamos basicamente criando uma mutação onde estamos definindo o valor de

a latitude a ser

algum valor de latitude e, em seguida

vamos fazer isso para cada coluna que vamos escrever,

vamos criar uma chave de linha e uma chave de linha é, neste caso, uma estrada,

a direção, o número da pista,

a chave do sensor e o registro de data e hora reverso.

Observe como eu estou fazendo o timestamp reverso.

Eu estou basicamente tomando o timestamp real e eu sou

subtraindo-o do valor máximo possível de um Long.

Então, eu estou recebendo o número de segundos desde 1970,

número de milissegundos desde 1970,

e subtraindo-o do valor máximo

de um longo e que lhe dá um timestamp reverso,

e estou fazendo disso a minha chave.

Então, rodovia, direção, pista,

ID do sensor e o registro de data e hora reverso.

Então eu estou criando uma lista de mutações e para

cada valor que eu preciso para escrever latitude, longitude,

sensor, etc, estou basicamente adicionando uma mutação

e então eu basicamente estou escrevendo a chave e a lista de mutações.

Então, isso é basicamente o meu objeto de mutação e, depois de ter um objeto de mutação,

Vou basicamente fazer BigTableIO.write para esta tabela.

Para ler o Bigtable novamente,

você pode fazer isso de forma muito programática usando a API do HBase,

podemos usar o cliente HBase,

você pode até usar o BigQuery para consultar o Bigtable.

Então, como projetamos para o Bigtable com esse entendimento

esse Bigtable separa computação e armazenamento.

By the way, por que separar computação e armazenamento?

Então, isso pode fazer o autoescalonamento.

Se conseguirmos um novo cluster que apareça,

o que esse novo cluster tem que fazer?

Só precisa de ponteiros para os dados

não precisa ter os dados.

Você só precisa de ponteiros para os dados.

Então, para fazer nosso design no Bigtable,

A primeira coisa é que as tabelas no Bigtable têm apenas um índice,

qual é a chave da linha.

Então, você tem linhas de dados,

cada linha tem uma chave.

Então, basicamente, tudo o que você está armazenando no Bigtable é um par de valores-chave.

Então, você tem uma chave e você tem alguns valores.

Esses valores podem ter colunas.

Então, essas colunas podem ser chamadas de famílias de colunas.

Mas há apenas uma dessas colunas é a chave.

Então, neste caso,

nós temos a chave de linha,

e todas as linhas são armazenadas em ordem crescente dessa chave de linha.

Nenhuma das outras colunas pode ser indexada.

A única coisa que é indexável é a chave da linha.

Então, o que fazemos quando projetamos o Bigtable é que pegamos

informação que pertence e grupo

-los juntos em que são chamados de famílias de colunas.

Então, neste caso, nesta tabela,

Eu tenho duas famílias de colunas.

Eu tenho uma família de colunas chamada informação do usuário,

e outra família de colunas chamada estatísticas de perfil.

Então, data de nascimento, idade e sexo são todas as informações sobre o usuário,

mas o número de mensagens que o usuário enviou,

isso é estatísticas de perfil.

Então, de certa forma, você pode pensar

as informações do usuário como esta informação muito lenta mudança,

e as estatísticas do perfil como sendo esta informação de mudança rápida.

Então, isso é outra coisa para pensar quando pensamos em

termos de quais tipos de colunas entram em uma família de colunas.

Então, isso traz à tona esse ponto

o número de seguidores que um usuário tem, onde deveria estar?

Deve estar na família da coluna de informações do usuário

ou deveria ser a família de colunas de estatísticas de perfil?

Quando você diz que as informações do usuário tendem a ser informações estáticas,

não costuma vir em streaming o tempo todo.

Mas seguidores e contagens de mensagens continuam mudando.

Sempre que o usuário basicamente atinge um like na mensagem de alguém ou no perfil de alguém,

eles se tornam um seguidor dessa pessoa,

então o número de seguidores aumenta et cetera.

Então, você pode pensar nos seguidores e no número de

mensagem conta como sendo esses dados que são constantemente atualizados.

Então, pode ser bom mover seus seguidores para as estatísticas do perfil.

Então, é assim que você pensa sobre as famílias das colunas.

Colunas relacionadas que tendem a ser atualizadas juntas.

Quando desenhamos o Bigtable,

você escolhe essencialmente entre um de dois tipos de projetos.

Esses são os dois tipos mais comuns de designs no Bigtable.

Um tipo de design é onde você tem uma tabela ampla.

Uma tabela ampla é uma tabela onde você tem um número de colunas,

e cada valor de coluna existe para cada linha.

Observe essa linha final

essa linha tem um valor para seguidores

um valor para aniversário,

um valor por idade,

um valor para sexo,

um valor para contagem de mensagens.

Então, Briana também faz Caitlyn.

Todas essas linhas,

tem um valor real para cada uma dessas colunas.

Se for esse o caso,

então você pode pensar em cada linha,

você basicamente tem todos esses valores.

Nós pensamos nisso como uma mesa ampla.

Por outro lado, você pode ter tabelas esparsas.

Tabelas nas quais muitas das colunas não terão dados nelas.

Pense, por exemplo, sobre os usuários que fornecem classificações nos produtos.

Quantos produtos um usuário avalia?

Talvez três, talvez quatro.

Então, se fôssemos fazer uma mesa ampla,

basicamente temos fila para todos os usuários

uma coluna para cada produto, primeiro

uma coluna para cada filme, por exemplo,

e apenas algumas dessas colunas terão dados nelas.

Isso é muito desperdício.

Em vez disso, o que faríamos no Bigtable é que fazemos isso como uma mesa estreita.

Portanto, uma tabela restrita é ótima para dados esparsos.

Então, neste caso,

o que faremos é que para cada usuário,

nós só armazenamos os nomes do filme,

e a classificação do filme que eles realmente avaliaram.

Então, aqui por exemplo,

onde estamos olhando para uma mesa estreita que consiste

de qual usuário segue qual outro usuário.

Mais uma vez, você pode ter milhões de usuários,

e qualquer usuário em particular,

nós não queremos armazenar um milhão de colunas para

esse outro par de usuários que esse cara poderia seguir.

Em vez disso, o que faremos é armazenar todos os nomes de usuários,

a pessoa real que eles são seguidos,

e você terá várias linhas que consistem nesse par.

Mas, claro, cada linha tem que ter uma chave única.

Então, o que faremos é que vamos fazer a chave consistir

deste par do seguidor e da pessoa que eles seguem.

Isso se torna uma chave única,

e então você pode realmente armazenar em texto simples.

Neste caso, o nome do seguidor, em seguida, os outros seguidores,

e outras informações sobre quando isso aconteceu, etc.

Então, você terá apenas aquelas chaves que realmente fazem sentido,

e o resto da coluna não teria uma entrada na tabela.

Então, essa é outra maneira de fazer seu design.

Então, pense em termos de

seus dados são densos

ou seus dados são escassos.

Dependendo se é denso ou esparso,

escolha entre um desses dois tipos de designs.

Agora, a coisa a lembrar é que, no Bigquery, as linhas são ordenadas.

Então, eles são armazenados classificados

Cada tablet vai consistir em linhas contíguas.

As linhas serão classificadas a partir do menor para o maior.

Então, eles serão classificados em ordem crescente.

6:56

Então, se você tem dados correspondentes a todas essas coisas,

eles vão ser codificados na verdade nesta ordem específica, certo?

Eles vão ser Andrew, Bob,

Jesse, Sandy e Zachary.

Eles vão estar em ordem alfabética.

Então, todas as suas linhas,

suas chaves de linha, as linhas serão armazenadas em ordem de classificação.

Então, dado esses fatos,

vamos falar sobre como otimizamos o design do Bigtable.

Então, o princípio básico é que, se você puder consultar os dados,

consultas que usam a chave de linha,

ou use um prefixo de linha,

ou usar um intervalo de linha são os mais eficientes.

Isso faz sentido porque a única coisa que estamos indexando é uma chave de linha.

Qualquer consulta que use uma chave de linha será mais eficiente porque

qualquer outra consulta essencialmente acabará lendo a tabela inteira.

Nós não queremos ler a tabela inteira.

Então, queremos usar a chave de linha o máximo possível.

Essa é a única coisa que é indexada.

Então, qualquer consulta que use a chave da linha,

mas não apenas a chave de linha

Acontece que Bigtable permitirá que você use um prefixo da chave de linha.

Portanto, você não precisa usar a chave de linha inteira.

Você pode dizer: "Pegue todas as linhas cuja chave comece com essas três letras".

Isso também é eficiente porque as linhas são armazenadas em ordem crescente.

Assim, pode ir em frente e encontrar todas as chaves que começam com algo muito rapidamente.

Então, quando você cria seu Bigtable,

a pergunta que você quer perguntar,

a primeira pergunta é:

Qual é a consulta mais comum que precisamos dar suporte?

Essa consulta mais comum que você precisa para suporte,

essencialmente ajuda você a descobrir o que sua chave de linha precisa ser.

Sua chave de linha precisa refletir essa consulta mais comum.

Então, por exemplo, se sempre vamos procurar,

me diga a velocidade do I35,

essa é a minha rodovia, no marcador de milha, algo em 347?

Então, você quer fazer esse parâmetro de consulta,

a estrada e o marcador da milha.

Você quer torná-lo o começo da chave da linha.

Você pode ter outras coisas na chave da linha,

mas você quer que alguém seja capaz de dizer

"Dê-me todas as linhas que começam com I35 e 347."

Talvez todas as perguntas sejam sobre a velocidade do I35?

Se eles apenas perguntarem a você a velocidade de uma rodovia,

você quer que a rodovia seja a primeira parte da chave de linha.

10:00

Então, a razão que isso funciona é porque você é basicamente

O Bigtable está armazenando as chaves em determinada ordem.

Então, o que significa que quando você está pensando em como você faz suas chaves,

você deseja basicamente fazer com que as entidades relacionadas sejam armazenadas de forma adjacente.

Então, você queria que quando o usuário fizesse uma consulta,

você deseja fornecer de volta a elas linhas que são contíguas.

Então, você quer basicamente fazer isso,

os usuários são muito propensos a puxar

registros que são contíguos porque isso tornará sua leitura eficiente.

É muito mais rápido se pudermos retornar resultados do mesmo tablet.

Se os resultados precisarem ser extraídos de vários tablets,

vai ser muito mais lento

do que se os resultados vierem do mesmo tablet.

A única maneira de os resultados virem do mesmo tablet é se você

pedindo registros cujas chaves são contíguas.

Use as chaves são uma após a outra.

Então, queremos basicamente fazer com que,

nós projetamos nossas chaves para que as consultas mais comuns retornem linhas adjacentes.

A segunda coisa que você pode querer fazer é fazer uma pergunta.

Será um caso de uso muito comum,

me dê os últimos recordes,

os últimos registros.

Se você vai conseguir os últimos discos o tempo todo,

então pense sobre isso.

Quais são os últimos registros?

Os últimos são essencialmente sobre timestamp.

Então, você quer adicionar o timestamp à chave,

mas você não quer adicionar o timestamp como está para a chave. Por que não?

Suponha que você tenha adicionado o timestamp à chave

pense no seu timestamp.

Digamos que um timestamp seja o ano, mês,

e dia, e se você adicionar esse carimbo de data / hora à chave,

qual vai ser a primeira parte da chave?

Então, qual será a primeira parte do timestamp?

A primeira parte do timestamp será o ano.

Anos anteriores virão depois dos últimos anos.

Então, se você começar sua coleta de dados em 2008,

os recordes de 2008 estarão na frente da sua mesa,

e seus dados mais recentes de 2017,

2018, todos os dados estarão na parte inferior.

Então, se o que queremos é recuperar os últimos registros.

Nós não devemos armazenar os dados em ordem crescente de timestamp.

Precisamos armazenar nossos dados em ordem decrescente de registro de data e hora.

Mas o Bigtable não faz isso.

O Bigtable armazena apenas as coisas em ordem crescente da chave da linha.

Então, o que vamos fazer é que na nossa chave de linha,

vamos colocar o timestamp reverso,

para que os últimos registros sejam os primeiros.

Então, é isso que vamos fazer.

Então, nossa chave de linha será a rodovia e o marcador da milha.

Isso é porque essa é a consulta mais comum.

As pessoas vão querer saber onde estou ou para onde estou indo,

qual será a velocidade? Certo então.

Eles sabem para onde estão indo por causa do software GPS.

Vai dizer: "Essa pessoa está viajando no I35,

e eles estão atualmente na milha número 345, eles estão se movendo para o norte.

Portanto, estou interessado nos marcadores de milhagem 346, 347 e 348.

Então, essa é uma consulta muito comum

e porque essa é uma consulta muito comum,

nós colocamos isso como um parâmetro de consulta,

nós fazemos isso a chave de linha.

Mas também não estamos interessados ​​em

a velocidade ontem ou no dia anterior ou três dias atrás.

Estamos interessados ​​na velocidade agora

os últimos recordes de velocidade.

Por causa disso, a terceira parte vai ser o timestamp reverso,

para que possamos obter os últimos registros de

esta rodovia e marcador de milha antes de qualquer outra coisa.

14:28

A segunda coisa que você quer pensar,

é como você faz suas leituras e como você faz suas gravações.

Nós já conversamos um pouco sobre as leituras.

A maneira como queremos fazer nossas leituras,

é que queremos linhas contíguas.

Linhas que estão próximas umas das outras para serem devolvidas de uma só vez.

Certo? Mas ao mesmo tempo,

você quer pensar sobre os ativos do Bigtable operando para

vários usuários em todo o aplicativo.

Você quer que suas leituras e gravações sejam distribuídas.

Em outras palavras, você não quer que toda a atividade aconteça em apenas um tablet.

Você quer que a atividade aconteça em vários tablets,

de modo que vários nós estão basicamente realizando o trabalho.

Então, você quer que suas leituras e gravações sejam distribuídas uniformemente,

para que a carga de trabalho em todos esses nós seja equilibrada.

Se suas leituras e gravações estiverem concentradas em um conjunto de chaves,

então vai ser um comprimido e

um nó que vai estar fazendo o peso do trabalho.

Isso não é um design muito escalável.

Então, o que precisamos é de distribuição de leituras e gravações.

Então, como distribuímos nossas leituras e gravações?

Nós queremos basicamente evitar o hotspotting.

Nós não queremos que um único nó faça um monte de atividades.

Então, para fazer isso,

Queremos pensar sobre o tipo de chaves de linha que não devemos fazer.

Nós não devemos, por exemplo,

tem uma chave de linha que é um domínio.

A razão pela qual não queremos que nossa chave de linha comece com o domínio é imaginar

que você está vendo os domínios dos quais as pessoas virão

vai ser geralmente o caso que

determinados domínios são extremamente ativos que outros domínios.

Então, digamos, por exemplo,

você tem centenas de clientes

e você está basicamente armazenando o domínio desses clientes,

é possível que dois ou três dos seus clientes sejam realmente responsáveis ​​por 30,

40% da sua empresa e os tablets correspondentes a

esses clientes vão basicamente causar o hotspotting.

Então, não queremos fazer isso.

Os domínios podem ser usados, mas você não deseja que sua chave de linha comece com o domínio.

Então, a chave da linha pode ser parte do domínio,

mas você não quer que a chave de linha comece com o domínio porque isso vai refletir

qual nó vai estar tomando o peso do processamento.

Você não deseja que os IDs do usuário sejam sua chave de linha se os IDs do usuário forem atribuídos sequencialmente.

Tudo bem se seus IDs de usuário forem atribuídos aleatoriamente

se você está apenas usando um código hash.

Então, algo muito arbitrário e algo muito aleatório.

É bom ter um ID de usuário como sua chave de linha.

Mas se seus IDs de usuário forem sequenciais,

isso pode ser um problema porque em muitas aplicações,

os usuários mais novos serão mais ativos do que os usuários criados 6, 7 anos atrás.

O que isto significa é que todos os nós,

lembre-se de que as linhas estão organizadas em ordem crescente,

então você terá alguns comprimidos que consistem em

todos os usuários antigos e alguns tablets que consistem em todos os novos usuários,

e os que correspondem a novos usuários tenderão a ser mais ativos.

Isso é um problema, isso é hotspotting.

Você quer que as leituras e gravações sejam distribuídas.

Você quer que todos os tablets estejam realmente funcionando em todos os momentos,

todos os nós para estar realmente trabalhando em todos os momentos.

Da mesma forma, você não quer ter

um identificador estático como chave

especialmente se você tiver um identificador estático que continuará sendo usado.

Então, por exemplo, se você tem uma chave de linha

uso mem ou uso da CPU ou uso de disco e você continua atualizando-o repetidamente,

poderia ser o caso desses comprimidos,

esses nós que basicamente fazem este processamento para esses dados constantemente atualizados,

eles vão basicamente se sobrecarregar.

Então o que você deveria fazer?

O que você deve fazer é distribuir essa carga de escrita.

Então, você quer basicamente distribuir a carga de escrita para que os dados estejam chegando,

Você quer escrever para todos os tablets.

Você não quer estar escrevendo para apenas um tablet.

Mas ao mesmo tempo,

você quer projetar sua chave de linha de tal forma que quando as pessoas fazem consultas,

eles basicamente retornam linhas adjacentes, linhas consecutivas.

Precisamos equilibrar isso.

Então, essa chave que fizemos anteriormente,

milhagem da rodovia e timestamp reverso

na verdade faz isso muito bem. Então, pense nisso.

Então, que tipo de consulta as pessoas fazem?

Uma única pessoa diria:

"Eu estou viajando no I35,

me dê o marcador de milhagem 347. "

Mas todo usuário no seu sistema vai viajar

I35 e vai estar no milhador 347? Não, certo?

Você terá usuários por toda a cidade.

Todos eles estarão em diferentes rodovias,

haverá milhas diferentes.

Então, em certo sentido,

as leituras serão distribuídas,

as gravações também serão distribuídas porque você tem sensores em todas as rodovias,

todos os milhadores, todos eles enviarão os dados.

Então, todos os nós vão obter uma quantidade aproximadamente igual de

trabalho e eles basicamente vão ler e escrever. Então, isso é ótimo.

Então, nós distribuímos escreve,

nós distribuímos leituras,

mas ao mesmo tempo,

para qualquer usuário em particular quando eles estão pedindo seus dados,

você começa a dar-lhes linhas consecutivas e isso é basicamente a ideia.

Agora isso funciona.

Vamos dar algumas opções

algumas alternativas para basicamente ver coisas que não funcionam.

Então, digamos, por exemplo,

nós projetamos para ser timestamp, rodovia, em seguida, milhador.

Qual é o problema aqui?

Agora, isso é um problema porque é muito semelhante à ideia de ID do usuário.

Você terá dados para um timestamp específico que está sendo gravado agora.

Então, todos os sensores

todos os dados que estão chegando vão entrar em

o mesmo tablet porque todas as linhas começam com um timestamp.

Então, isso funciona se é timestamp reverso ou timestamp regular.

Você está escrevendo para o mesmo tablet porque todos esses sensores são

envio de dados ao mesmo tempo e porque todos enviam dados ao mesmo tempo,

Se a sua chave de linha começa com o tempo, você é o anfitrião.

Você não quer que sua chave comece com o tempo porque

então é o mesmo tablet que vai ficar sobrecarregado.

Vamos pegar o segundo.

Suponha que você faça um milhagem primeiro

e depois a estrada

e então você tem o timestamp reverso. Agora, qual é o problema?

O problema acontece.

Isso parece estar bem, mas pense em um caso de uso típico.

O caso de uso típico será um gráfico plotando

software que basicamente está tentando escolher entre alguma rota,

uma rota ou outra rota e normalmente as pessoas pedem,

Qual é a velocidade média em uma rodovia particular?

Você quer basicamente dizer,

nesta estrada entre as milhas 340 e 350, porque essas são as duas interseções,

me diga a velocidade?

Esse tipo de consulta se torna extremamente difícil se você começar com um milhador porque agora

se você está pedindo todos os dados para as milhas 340 a 350,

você não está mais lendo linhas contíguas,

você está lendo linhas de talvez 10 comprimidos diferentes.

Então, o problema com isso,

começando com um milhador é que

suas consultas não retornarão linhas consecutivas.

Sua consulta, me dê a velocidade entre milhagem

340 e milhador 350 na auto-estrada I35,

isso não vai ser linhas consecutivas que vai

leia de vários tablets e isso é um problema.

Então, quando você pensa sobre o seu design,

você quer pensar em termos de como eu organizo

minha chave de linha para que eu retorne linhas adjacentes, tanto quanto

possível para que eu tenha distribuído

escrevendo o máximo possível e distribuído lê o máximo possível.

Então, como projetamos para o Bigtable com esse entendimento

esse Bigtable separa computação e armazenamento.

By the way, por que separar computação e armazenamento?

Então, isso pode fazer o autoescalonamento.

Se conseguirmos um novo cluster que apareça,

o que esse novo cluster tem que fazer?

Só precisa de ponteiros para os dados

não precisa ter os dados.

Você só precisa de ponteiros para os dados.

Então, para fazer nosso design no Bigtable,

A primeira coisa é que as tabelas no Bigtable têm apenas um índice,

qual é a chave da linha.

Então, você tem linhas de dados,

cada linha tem uma chave.

Então, basicamente, tudo o que você está armazenando no Bigtable é um par de valores-chave.

Então, você tem uma chave e você tem alguns valores.

Esses valores podem ter colunas.

Então, essas colunas podem ser chamadas de famílias de colunas.

Mas há apenas uma dessas colunas é a chave.

Então, neste caso,

nós temos a chave de linha,

e todas as linhas são armazenadas em ordem crescente dessa chave de linha.

Nenhuma das outras colunas pode ser indexada.

A única coisa que é indexável é a chave da linha.

Então, o que fazemos quando projetamos o Bigtable é que pegamos

informação que pertence e grupo

-los juntos em que são chamados de famílias de colunas.

Então, neste caso, nesta tabela,

Eu tenho duas famílias de colunas.

Eu tenho uma família de colunas chamada informação do usuário,

e outra família de colunas chamada estatísticas de perfil.

Então, data de nascimento, idade e sexo são todas as informações sobre o usuário,

mas o número de mensagens que o usuário enviou,

isso é estatísticas de perfil.

Então, de certa forma, você pode pensar

as informações do usuário como esta informação muito lenta mudança,

e as estatísticas do perfil como sendo esta informação de mudança rápida.

Então, isso é outra coisa para pensar quando pensamos em

termos de quais tipos de colunas entram em uma família de colunas.

Então, isso traz à tona esse ponto

o número de seguidores que um usuário tem, onde deveria estar?

Deve estar na família da coluna de informações do usuário

ou deveria ser a família de colunas de estatísticas de perfil?

Quando você diz que as informações do usuário tendem a ser informações estáticas,

não costuma vir em streaming o tempo todo.

Mas seguidores e contagens de mensagens continuam mudando.

Sempre que o usuário basicamente atinge um like na mensagem de alguém ou no perfil de alguém,

eles se tornam um seguidor dessa pessoa,

então o número de seguidores aumenta et cetera.

Então, você pode pensar nos seguidores e no número de

mensagem conta como sendo esses dados que são constantemente atualizados.

Então, pode ser bom mover seus seguidores para as estatísticas do perfil.

Então, é assim que você pensa sobre as famílias das colunas.

Colunas relacionadas que tendem a ser atualizadas juntas.

Quando desenhamos o Bigtable,

você escolhe essencialmente entre um de dois tipos de projetos.

Esses são os dois tipos mais comuns de designs no Bigtable.

Um tipo de design é onde você tem uma tabela ampla.

Uma tabela ampla é uma tabela onde você tem um número de colunas,

e cada valor de coluna existe para cada linha.

Observe essa linha final

essa linha tem um valor para seguidores

um valor para aniversário,

um valor por idade,

um valor para sexo,

um valor para contagem de mensagens.

Então, Briana também faz Caitlyn.

Todas essas linhas,

tem um valor real para cada uma dessas colunas.

Se for esse o caso,

então você pode pensar em cada linha,

você basicamente tem todos esses valores.

Nós pensamos nisso como uma mesa ampla.

Por outro lado, você pode ter tabelas esparsas.

Tabelas nas quais muitas das colunas não terão dados nelas.

Pense, por exemplo, sobre os usuários que fornecem classificações nos produtos.

Quantos produtos um usuário avalia?

Talvez três, talvez quatro.

Então, se fôssemos fazer uma mesa ampla,

basicamente temos fila para todos os usuários

uma coluna para cada produto, primeiro

uma coluna para cada filme, por exemplo,

e apenas algumas dessas colunas terão dados nelas.

Isso é muito desperdício.

Em vez disso, o que faríamos no Bigtable é que fazemos isso como uma mesa estreita.

Portanto, uma tabela restrita é ótima para dados esparsos.

Então, neste caso,

o que faremos é que para cada usuário,

nós só armazenamos os nomes do filme,

e a classificação do filme que eles realmente avaliaram.

Então, aqui por exemplo,

onde estamos olhando para uma mesa estreita que consiste

de qual usuário segue qual outro usuário.

Mais uma vez, você pode ter milhões de usuários,

e qualquer usuário em particular,

nós não queremos armazenar um milhão de colunas para

esse outro par de usuários que esse cara poderia seguir.

Em vez disso, o que faremos é armazenar todos os nomes de usuários,

a pessoa real que eles são seguidos,

e você terá várias linhas que consistem nesse par.

Mas, claro, cada linha tem que ter uma chave única.

Então, o que faremos é que vamos fazer a chave consistir

deste par do seguidor e da pessoa que eles seguem.

Isso se torna uma chave única,

e então você pode realmente armazenar em texto simples.

Neste caso, o nome do seguidor, em seguida, os outros seguidores,

e outras informações sobre quando isso aconteceu, etc.

Então, você terá apenas aquelas chaves que realmente fazem sentido,

e o resto da coluna não teria uma entrada na tabela.

Então, essa é outra maneira de fazer seu design.

Então, pense em termos de

seus dados são densos

ou seus dados são escassos.

Dependendo se é denso ou esparso,

escolha entre um desses dois tipos de designs.

Agora, a coisa a lembrar é que, no Bigquery, as linhas são ordenadas.

Então, eles são armazenados classificados

Cada tablet vai consistir em linhas contíguas.

As linhas serão classificadas a partir do menor para o maior.

Então, eles serão classificados em ordem crescente.

6:56

Então, se você tem dados correspondentes a todas essas coisas,

eles vão ser codificados na verdade nesta ordem específica, certo?

Eles vão ser Andrew, Bob,

Jesse, Sandy e Zachary.

Eles vão estar em ordem alfabética.

Então, todas as suas linhas,

suas chaves de linha, as linhas serão armazenadas em ordem de classificação.

Então, dado esses fatos,

vamos falar sobre como otimizamos o design do Bigtable.

Então, o princípio básico é que, se você puder consultar os dados,

consultas que usam a chave de linha,

ou use um prefixo de linha,

ou usar um intervalo de linha são os mais eficientes.

Isso faz sentido porque a única coisa que estamos indexando é uma chave de linha.

Qualquer consulta que use uma chave de linha será mais eficiente porque

qualquer outra consulta essencialmente acabará lendo a tabela inteira.

Nós não queremos ler a tabela inteira.

Então, queremos usar a chave de linha o máximo possível.

Essa é a única coisa que é indexada.

Então, qualquer consulta que use a chave da linha,

mas não apenas a chave de linha

Acontece que Bigtable permitirá que você use um prefixo da chave de linha.

Portanto, você não precisa usar a chave de linha inteira.

Você pode dizer: "Pegue todas as linhas cuja chave comece com essas três letras".

Isso também é eficiente porque as linhas são armazenadas em ordem crescente.

Assim, pode ir em frente e encontrar todas as chaves que começam com algo muito rapidamente.

Então, quando você cria seu Bigtable,

a pergunta que você quer perguntar,

a primeira pergunta é:

Qual é a consulta mais comum que precisamos dar suporte?

Essa consulta mais comum que você precisa para suporte,

essencialmente ajuda você a descobrir o que sua chave de linha precisa ser.

Sua chave de linha precisa refletir essa consulta mais comum.

Então, por exemplo, se sempre vamos procurar,

me diga a velocidade do I35,

essa é a minha rodovia, no marcador de milha, algo em 347?

Então, você quer fazer esse parâmetro de consulta,

a estrada e o marcador da milha.

Você quer torná-lo o começo da chave da linha.

Você pode ter outras coisas na chave da linha,

mas você quer que alguém seja capaz de dizer

"Dê-me todas as linhas que começam com I35 e 347."

Talvez todas as perguntas sejam sobre a velocidade do I35?

Se eles apenas perguntarem a você a velocidade de uma rodovia,

você quer que a rodovia seja a primeira parte da chave de linha.

10:00

Então, a razão que isso funciona é porque você é basicamente

O Bigtable está armazenando as chaves em determinada ordem.

Então, o que significa que quando você está pensando em como você faz suas chaves,

você deseja basicamente fazer com que as entidades relacionadas sejam armazenadas de forma adjacente.

Então, você queria que quando o usuário fizesse uma consulta,

você deseja fornecer de volta a elas linhas que são contíguas.

Então, você quer basicamente fazer isso,

os usuários são muito propensos a puxar

registros que são contíguos porque isso tornará sua leitura eficiente.

É muito mais rápido se pudermos retornar resultados do mesmo tablet.

Se os resultados precisarem ser extraídos de vários tablets,

vai ser muito mais lento

do que se os resultados vierem do mesmo tablet.

A única maneira de os resultados virem do mesmo tablet é se você

pedindo registros cujas chaves são contíguas.

Use as chaves são uma após a outra.

Então, queremos basicamente fazer com que,

nós projetamos nossas chaves para que as consultas mais comuns retornem linhas adjacentes.

A segunda coisa que você pode querer fazer é fazer uma pergunta.

Será um caso de uso muito comum,

me dê os últimos recordes,

os últimos registros.

Se você vai conseguir os últimos discos o tempo todo,

então pense sobre isso.

Quais são os últimos registros?

Os últimos são essencialmente sobre timestamp.

Então, você quer adicionar o timestamp à chave,

mas você não quer adicionar o timestamp como está para a chave. Por que não?

Suponha que você tenha adicionado o timestamp à chave

pense no seu timestamp.

Digamos que um timestamp seja o ano, mês,

e dia, e se você adicionar esse carimbo de data / hora à chave,

qual vai ser a primeira parte da chave?

Então, qual será a primeira parte do timestamp?

A primeira parte do timestamp será o ano.

Anos anteriores virão depois dos últimos anos.

Então, se você começar sua coleta de dados em 2008,

os recordes de 2008 estarão na frente da sua mesa,

e seus dados mais recentes de 2017,

2018, todos os dados estarão na parte inferior.

Então, se o que queremos é recuperar os últimos registros.

Nós não devemos armazenar os dados em ordem crescente de timestamp.

Precisamos armazenar nossos dados em ordem decrescente de registro de data e hora.

Mas o Bigtable não faz isso.

O Bigtable armazena apenas as coisas em ordem crescente da chave da linha.

Então, o que vamos fazer é que na nossa chave de linha,

vamos colocar o timestamp reverso,

para que os últimos registros sejam os primeiros.

Então, é isso que vamos fazer.

Então, nossa chave de linha será a rodovia e o marcador da milha.

Isso é porque essa é a consulta mais comum.

As pessoas vão querer saber onde estou ou para onde estou indo,

qual será a velocidade? Certo então.

Eles sabem para onde estão indo por causa do software GPS.

Vai dizer: "Essa pessoa está viajando no I35,

e eles estão atualmente na milha número 345, eles estão se movendo para o norte.

Portanto, estou interessado nos marcadores de milhagem 346, 347 e 348.

Então, essa é uma consulta muito comum

e porque essa é uma consulta muito comum,

nós colocamos isso como um parâmetro de consulta,

nós fazemos isso a chave de linha.

Mas também não estamos interessados ​​em

a velocidade ontem ou no dia anterior ou três dias atrás.

Estamos interessados ​​na velocidade agora

os últimos recordes de velocidade.

Por causa disso, a terceira parte vai ser o timestamp reverso,

para que possamos obter os últimos registros de

esta rodovia e marcador de milha antes de qualquer outra coisa.

14:28

A segunda coisa que você quer pensar,

é como você faz suas leituras e como você faz suas gravações.

Nós já conversamos um pouco sobre as leituras.

A maneira como queremos fazer nossas leituras,

é que queremos linhas contíguas.

Linhas que estão próximas umas das outras para serem devolvidas de uma só vez.

Certo? Mas ao mesmo tempo,

você quer pensar sobre os ativos do Bigtable operando para

vários usuários em todo o aplicativo.

Você quer que suas leituras e gravações sejam distribuídas.

Em outras palavras, você não quer que toda a atividade aconteça em apenas um tablet.

Você quer que a atividade aconteça em vários tablets,

de modo que vários nós estão basicamente realizando o trabalho.

Então, você quer que suas leituras e gravações sejam distribuídas uniformemente,

para que a carga de trabalho em todos esses nós seja equilibrada.

Se suas leituras e gravações estiverem concentradas em um conjunto de chaves,

então vai ser um comprimido e

um nó que vai estar fazendo o peso do trabalho.

Isso não é um design muito escalável.

Então, o que precisamos é de distribuição de leituras e gravações.

Então, como distribuímos nossas leituras e gravações?

Nós queremos basicamente evitar o hotspotting.

Nós não queremos que um único nó faça um monte de atividades.

Então, para fazer isso,

Queremos pensar sobre o tipo de chaves de linha que não devemos fazer.

Nós não devemos, por exemplo,

tem uma chave de linha que é um domínio.

A razão pela qual não queremos que nossa chave de linha comece com o domínio é imaginar

que você está vendo os domínios dos quais as pessoas virão

vai ser geralmente o caso que

determinados domínios são extremamente ativos que outros domínios.

Então, digamos, por exemplo,

você tem centenas de clientes

e você está basicamente armazenando o domínio desses clientes,

é possível que dois ou três dos seus clientes sejam realmente responsáveis ​​por 30,

40% da sua empresa e os tablets correspondentes a

esses clientes vão basicamente causar o hotspotting.

Então, não queremos fazer isso.

Os domínios podem ser usados, mas você não deseja que sua chave de linha comece com o domínio.

Então, a chave da linha pode ser parte do domínio,

mas você não quer que a chave de linha comece com o domínio porque isso vai refletir

qual nó vai estar tomando o peso do processamento.

Você não deseja que os IDs do usuário sejam sua chave de linha se os IDs do usuário forem atribuídos sequencialmente.

Tudo bem se seus IDs de usuário forem atribuídos aleatoriamente

se você está apenas usando um código hash.

Então, algo muito arbitrário e algo muito aleatório.

É bom ter um ID de usuário como sua chave de linha.

Mas se seus IDs de usuário forem sequenciais,

isso pode ser um problema porque em muitas aplicações,

os usuários mais novos serão mais ativos do que os usuários criados 6, 7 anos atrás.

O que isto significa é que todos os nós,

lembre-se de que as linhas estão organizadas em ordem crescente,

então você terá alguns comprimidos que consistem em

todos os usuários antigos e alguns tablets que consistem em todos os novos usuários,

e os que correspondem a novos usuários tenderão a ser mais ativos.

Isso é um problema, isso é hotspotting.

Você quer que as leituras e gravações sejam distribuídas.

Você quer que todos os tablets estejam realmente funcionando em todos os momentos,

todos os nós para estar realmente trabalhando em todos os momentos.

Da mesma forma, você não quer ter

um identificador estático como chave

especialmente se você tiver um identificador estático que continuará sendo usado.

Então, por exemplo, se você tem uma chave de linha

uso mem ou uso da CPU ou uso de disco e você continua atualizando-o repetidamente,

poderia ser o caso desses comprimidos,

esses nós que basicamente fazem este processamento para esses dados constantemente atualizados,

eles vão basicamente se sobrecarregar.

Então o que você deveria fazer?

O que você deve fazer é distribuir essa carga de escrita.

Então, você quer basicamente distribuir a carga de escrita para que os dados estejam chegando,

Você quer escrever para todos os tablets.

Você não quer estar escrevendo para apenas um tablet.

Mas ao mesmo tempo,

você quer projetar sua chave de linha de tal forma que quando as pessoas fazem consultas,

eles basicamente retornam linhas adjacentes, linhas consecutivas.

Precisamos equilibrar isso.

Então, essa chave que fizemos anteriormente,

milhagem da rodovia e timestamp reverso

na verdade faz isso muito bem. Então, pense nisso.

Então, que tipo de consulta as pessoas fazem?

Uma única pessoa diria:

"Eu estou viajando no I35,

me dê o marcador de milhagem 347. "

Mas todo usuário no seu sistema vai viajar

I35 e vai estar no milhador 347? Não, certo?

Você terá usuários por toda a cidade.

Todos eles estarão em diferentes rodovias,

haverá milhas diferentes.

Então, em certo sentido,

as leituras serão distribuídas,

as gravações também serão distribuídas porque você tem sensores em todas as rodovias,

todos os milhadores, todos eles enviarão os dados.

Então, todos os nós vão obter uma quantidade aproximadamente igual de

trabalho e eles basicamente vão ler e escrever. Então, isso é ótimo.

Então, nós distribuímos escreve,

nós distribuímos leituras,

mas ao mesmo tempo,

para qualquer usuário em particular quando eles estão pedindo seus dados,

você começa a dar-lhes linhas consecutivas e isso é basicamente a ideia.

Agora isso funciona.

Vamos dar algumas opções

algumas alternativas para basicamente ver coisas que não funcionam.

Então, digamos, por exemplo,

nós projetamos para ser timestamp, rodovia, em seguida, milhador.

Qual é o problema aqui?

Agora, isso é um problema porque é muito semelhante à ideia de ID do usuário.

Você terá dados para um timestamp específico que está sendo gravado agora.

Então, todos os sensores

todos os dados que estão chegando vão entrar em

o mesmo tablet porque todas as linhas começam com um timestamp.

Então, isso funciona se é timestamp reverso ou timestamp regular.

Você está escrevendo para o mesmo tablet porque todos esses sensores são

envio de dados ao mesmo tempo e porque todos enviam dados ao mesmo tempo,

Se a sua chave de linha começa com o tempo, você é o anfitrião.

Você não quer que sua chave comece com o tempo porque

então é o mesmo tablet que vai ficar sobrecarregado.

Vamos pegar o segundo.

Suponha que você faça um milhagem primeiro

e depois a estrada

e então você tem o timestamp reverso. Agora, qual é o problema?

O problema acontece.

Isso parece estar bem, mas pense em um caso de uso típico.

O caso de uso típico será um gráfico plotando

software que basicamente está tentando escolher entre alguma rota,

uma rota ou outra rota e normalmente as pessoas pedem,

Qual é a velocidade média em uma rodovia particular?

Você quer basicamente dizer,

nesta estrada entre as milhas 340 e 350, porque essas são as duas interseções,

me diga a velocidade?

Esse tipo de consulta se torna extremamente difícil se você começar com um milhador porque agora

se você está pedindo todos os dados para as milhas 340 a 350,

você não está mais lendo linhas contíguas,

você está lendo linhas de talvez 10 comprimidos diferentes.

Então, o problema com isso,

começando com um milhador é que

suas consultas não retornarão linhas consecutivas.

Sua consulta, me dê a velocidade entre milhagem

340 e milhador 350 na auto-estrada I35,

isso não vai ser linhas consecutivas que vai

leia de vários tablets e isso é um problema.

Então, quando você pensa sobre o seu design,

você quer pensar em termos de como eu organizo

minha chave de linha para que eu retorne linhas adjacentes, tanto quanto

possível para que eu tenha distribuído

escrevendo o máximo possível e distribuído lê o máximo possível.