O livro de receitas de engenharia de dados

Dominando o encanamento da ciência de dados

Andreas Kretz

18 de maio de 2019

v1.1

Conteúdo

Parte I

Introdução

1 Como usar este livro de receitas

O que você realmente precisa aprender para se tornar um engenheiro de dados incrível? Não procure mais, você encontra aqui.

Como usar este documento: Isto não é um treinamento! É uma coleção de habilidades que valorizo muito em meu trabalho diário como engenheiro de dados. Destina-se a ser um ponto de partida para você encontrar os tópicos para pesquisar.

Este projeto é um trabalho em andamento! Nas próximas semanas, compartilharei com vocês meus pensamentos sobre a importância de cada tópico. Também tento incluir links para recursos úteis.

Como descobrir o que há de novo? Você sempre encontrará a versão mais recente no meu Patreon https://www.patreon.com/plumbersofds

Ajude a tornar esta coleção incrível! Participe da discussão no Patreon ou escreva-me um e-mail para andreaskayy@gmail.com. Diga-me seus pensamentos, o que você valoriza, acha que deveria ser incluído ou onde estou errado.

- Dados do Twitter para prever o melhor horário para postar usando a hashtag datascience ou ai
- Encontre os principais tweets do dia
- Principais usuários
- Analisar sentimentos e palavras-chave

2 Engenheiro de dados x Cientistas de dados

2.1 Cientista de Dados

Os cientistas de dados não são como todos os outros cientistas.

Os cientistas de dados não usam jalecos brancos nem trabalham em laboratórios de alta tecnologia cheios de equipamentos de filmes de ficção científica. Eles trabalham em escritórios como você e eu.

O que os difere da maioria de nós é que eles são os especialistas em matemática. Eles usam álgebra linear e cálculo multivariável para criar novos insights a partir dos dados existentes.

Como exatamente esse insight se parece?

Aqui está um exemplo:

Uma empresa industrial produz muitos produtos que precisam ser testados antes do envio.

Normalmente, esses testes levam muito tempo porque existem centenas de coisas a serem testadas.
Tudo para garantir que seu produto não quebre.

Não seria ótimo saber com antecedência se um teste falha dez etapas abaixo da linha? Se você soubesse disso, poderia pular os outros testes e simplesmente jogar fora o produto ou consertá-lo.

É exatamente aí que um cientista de dados pode ajudá-lo em grande escala. Esse campo é chamado de análise preditiva e a técnica de escolha é o aprendizado de máquina.

Máquina o quê? Aprendizado?

Sim, aprendizado de máquina, funciona assim:

Você alimenta um algoritmo com dados de medição. Ele gera um modelo e o otimiza com base nos dados que você o alimentou. Esse modelo basicamente representa um padrão de aparência de seus dados. Você mostra a esse modelo novos dados e o modelo lhe dirá se os dados ainda representam os dados com os quais você os treinou. Essa técnica também pode ser usada para prever falhas de máquina com antecedência com aprendizado de máquina. Claro que todo o processo não é tão simples.

O processo real de treinamento e aplicação de um modelo não é tão difícil. Muito trabalho para o cientista de dados é descobrir como pré-processar os dados que alimentam os algoritmos.

Porque para treinar um algoritmo você precisa de dados úteis. Se você usar quaisquer dados para o treinamento, o modelo produzido não será confiável.

Um modelo não confiável para prever falhas de máquina diria a você que sua máquina está danificada, mesmo que não esteja. Ou ainda pior: diria que a máquina está bem mesmo quando existe uma avaria.

As saídas do modelo são muito abstratas. Você também precisa pós-processar as saídas do modelo para receber valores de saúde de 0 a 100.

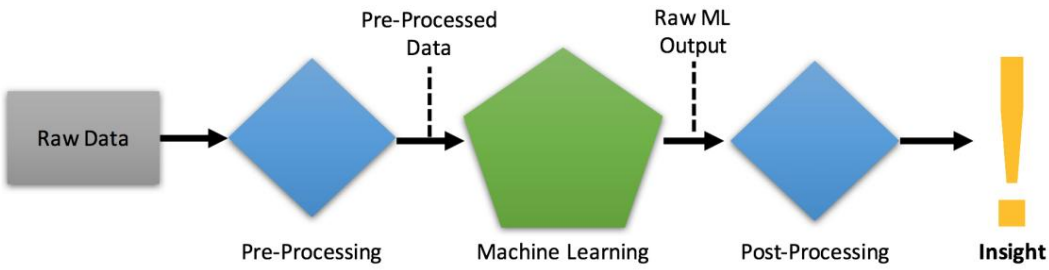

Figura 2.1: O pipeline de aprendizado de máquina

2.2 Engenheiro de Dados

Os engenheiros de dados são o elo entre a estratégia de big data da administração e os cientistas de dados que precisam trabalhar com dados.

O que eles fazem é construir as plataformas que permitem que os cientistas de dados façam sua mágica.

Essas plataformas geralmente são usadas de quatro maneiras diferentes:

- Ingestão de dados e armazenamento de grandes quantidades de dados
- Criação de algoritmo por cientistas de dados
- Automação dos modelos e algoritmos de aprendizado de máquina do cientista de dados para uso de produção
- Visualização de dados para funcionários e clientes
- Na maioria das vezes, esses caras começam como arquitetos de soluções tradicionais para sistemas que envolvem bancos de dados SQL, servidores web, instalações SAP e outros “padrão”

sistemas.

Mas, para criar plataformas de big data, o engenheiro precisa ser especialista em especificar, configurar e manter tecnologias de big data como: Hadoop, Spark, HBase, Cassandra, MongoDB, Kafka, Redis e muito mais.

O que eles também precisam é de experiência em como implantar sistemas em infraestrutura de nuvem, como Amazon ou Google, ou hardware local.

2.3 De quem as empresas precisam

Para uma boa empresa, é absolutamente importante ter engenheiros de dados bem treinados e cientistas de dados.

Pense no cientista de dados como o piloto profissional de carros de corrida. Um atleta em forma com talento e habilidades de condução como você nunca viu.

O que ele precisa para vencer corridas é alguém que lhe forneça o carro de corrida perfeito para dirigir.
É para isso que serve o arquiteto de soluções.

Como o motorista e sua equipe, o cientista de dados e o engenheiro de dados precisam trabalhar juntos. Eles precisam conhecer as diferentes ferramentas de big data por dentro e por fora.

É por isso que as empresas procuram pessoas com experiência no Spark. É um terreno comum entre ambos que impulsiona a inovação.

O Spark fornece aos cientistas de dados as ferramentas para fazer análises e ajuda os engenheiros a colocar os algoritmos do cientista de dados em produção.

Afinal, esses dois decidem quão boa é a plataforma de dados, quão boa é a visão analítica e quão rápido todo o sistema entra em um estado pronto para produção.

parte II

Habilidades básicas de engenharia de dados

3 Aprenda a codificar

Por que isso é importante: sem codificação, você não pode fazer muito em engenharia de dados. Não consigo contar quantas vezes precisei de um rápido hack Java.

As possibilidades são infinitas:

- Gravar ou obter rapidamente alguns dados de um banco de dados SQL
- Teste para produzir mensagens para um tópico Kafka
- Compreensão do código-fonte de um Webservice Java
- Lendo as estatísticas do contador de um armazenamento de valor de chave HBase

Então, qual idioma eu recomendo?

Eu recomendo fortemente o Java. Está em toda parte!

Ao entrar no processamento de dados com o Spark, você deve usar o Scala. Mas, depois de aprender Java, isso é fácil de fazer.

Além disso, o Python é uma ótima escolha. É super versátil.

Pessoalmente, no entanto, não gosto muito de Python. mas vou pesquisar sobre isso

Onde aprender? Há um curso de Java na Udemy que você pode consultar: https://www.udemy.com/java Programming-tutorial-for-beginners

- Programação orientada a objeto OOP
- O que são testes de unidade para garantir que seu código está funcionando
- Programação Funcional
- Como usar ferramentas de gerenciamento de compilação como Maven
- Testes resilientes (?)

Falei sobre a importância de aprender fazendo neste podcast:

4 Familiarize-se com o Github

Por que isso é importante: um dos maiores problemas com a codificação é acompanhar as mudanças. Também é quase impossível manter um programa do qual você possui várias versões.

Outro é o tópico de colaboração e documentação. O que é super importante.

Digamos que você trabalhe em um aplicativo Spark e suas faculdades precisem fazer alterações enquanto você estiver de férias. Sem algum gerenciamento de código, eles estão em apuros:

Onde está o código? O que você mudou por último? Onde está a documentação? Como marcamos o que mudamos?

Mas se você colocar seu código no GitHub, suas faculdades poderão encontrar seu código. Eles podem entendê-lo por meio de sua documentação (por favor, também tenha comentários em linha)

Os desenvolvedores podem extrair seu código, fazer uma nova ramificação e fazer as alterações. Após as férias, você pode inspecionar o que eles fizeram e mesclar com o código original. e você acaba tendo apenas um aplicativo

Onde aprender: Confira a página de Guias do GitHub, onde você pode aprender todos os fundamentos: https://guides.github.com/introduction/flow/

Esta ótima folha de dicas de comandos do GitHub salvou minha bunda várias vezes: https://www.atlassian.com/git git-cheatsheet

Puxar, Empurrar, Ramificar, Bifurcar

Também falei sobre isso neste podcast:

5 Desenvolvimento Ágil – disponível

Agilidade, a capacidade de se adaptar rapidamente às mudanças nas circunstâncias.

Hoje em dia todo mundo quer ser ágil. Pessoas de grandes ou pequenas empresas procuram a “mentalidade de startup”.

Muitos pensam que é a cultura corporativa. Outros acham que é o processo como criamos as coisas aquilo importa.

Neste artigo vou falar sobre agilidade e autoconfiança. Sobre como você pode incorporar a agilidade em sua carreira profissional.

5.1 Por que o ágil é tão importante?

Historicamente, o desenvolvimento é praticado como um processo bem definido. Você pensa em algo, especifica, desenvolve e depois constrói em produção em massa.

É um processo um pouco arrogante. Você assume que já sabe exatamente o que um cliente deseja. Ou como um produto deve ser e como tudo funciona.

O problema é que o mundo não funciona assim!

Muitas vezes as circunstâncias mudam por causa de fatores internos.

Às vezes, as coisas simplesmente não funcionam como planejado ou as coisas são mais difíceis do que você pensa.

Você precisa se adaptar.

Outras vezes, você descobre que construiu algo que os clientes não gostam e precisa ser alterado.

Você precisa se adaptar.

É por isso que as pessoas embarcam no trem do Scrum. Porque Scrum é a definição de desenvolvimento ágil, certo?

5.2 Regras ágeis que aprendi ao longo dos anos – disponível

5.2.1 O método está fazendo diferença?

Sim, o Scrum ou o OKR do Google podem ajudar a ser mais ágil. O segredo para ser ágil, no entanto, não é apenas como você cria.

O que me faz estremecer é que as pessoas tentam dizer que ser ágil começa na sua cabeça. Então, o problema é você?

Não!

A maior lição que aprendi nos últimos anos é esta: a agilidade vai pelo ralo quando você terceiriza o trabalho.

5.2.2 O problema da terceirização

Eu sei que no papel a terceirização parece óbvia: custos de desenvolvimento em relação ao fixo custos.

É caro vincular recursos existentes a uma tarefa. É ainda mais caro se você precisar contratar novos funcionários.

O problema com a terceirização é que você paga alguém para construir coisas para você.

Não importa quem você paga para fazer algo por você. Ele precisa ganhar dinheiro.

A agenda dele será gastar o mínimo de tempo possível em seu trabalho. É por isso que a terceirização requer contratos, especificações detalhadas, cronogramas e datas de entrega.

Ele não quer gastar mais tempo em um projeto, apenas porque você quer mudanças no meio. Cada mudança não planejada custa a ele tempo e, portanto, dinheiro.

Nesse caso, você precisa fazer outra especificação detalhada e uma alteração no contrato.

Ele não vai se concentrar em melhorar o produto durante o desenvolvimento. Em primeiro lugar porque ele não tem uma visão geral. Em segundo lugar porque ele não quer.

Ele está fazendo o que lhe é dito.

Quem pode culpá-lo? Se eu fosse o subcontratado, faria exatamente o mesmo!

Isso soa ágil para você?

5.2.3 O conhecimento é rei: uma lição de Elon Musk

Fazendo tudo internamente, é por isso que as startups são tão produtivas. Não se perde tempo esperando por outra pessoa.

Se algo não funcionar ou precisar ser alterado, há alguém na equipe que pode fazer isso imediatamente. .

Um exemplo muito proeminente que segue essa estratégia é Elon Musk.

As Gigafábricas da Tesla são projetadas para obter matérias-primas de um lado e cuspir carros do outro. Por que você acha que a Tesla está construindo Gigafábricas que custam muito dinheiro?

Por que a SpaceX está construindo seus motores espaciais? Claramente, existem outras empresas mais antigas que poderiam fazer isso por eles.

Por que Elon está construindo máquinas de perfuração de túneis em sua nova empresa de perfuração?

À primeira vista, isso não faz sentido!

5.2.4 Como você realmente pode ser ágil

Se você olhar mais de perto, tudo se resume a controle e conhecimento. Você, sua equipe, sua empresa, precisa fazer o máximo possível por conta própria. A autoconfiança é rei.

Construa seu conhecimento e, portanto, o conhecimento da equipe. Quando você tem a capacidade de fazer tudo sozinho, você está no controle total.

Você pode construir carros elétricos, motores de foguetes ou túneis de perfuração.

Não confie muito nos outros e tenha certeza de fazer as coisas sozinho.

Sonhe grande e APENAS FAÇA!

PS. Não me interpretem mal. Você ainda pode terceirizar o trabalho. Basta fazer isso de maneira inteligente, terceirizando pequenas peças independentes.

5.3 Técnicas Ágeis

5.3.1 Scrum

5.3.2 DISTRITO

Falei sobre isso neste Podcast: https://anchor.fm/andreaskayy/embed/episodes/Agile Development-Is-Important-But-Please-Dont-Do-Scrum–PoDS-041-e2e2j4

6 Aprenda como um computador funciona

6.1 CPU,RAM,GPU,HDD

6.2 Diferenças entre PCs e Servidores

Falei sobre hardware de computador e processamento de GPU neste podcast: https://anchor.fm/andreaskayy the-hardware-and-the-GPU-is-super-important–PoDS-030-e23rig

7 Redes de Computadores - Dados Transmissão

7.1 Modelo ISO/OSI

7.2 Sub-redes IP

7.3 Interruptor, Interruptor de Nível 3

7.4 Roteador

7.5 Firewalls

Falei sobre Infraestrutura e Técnicas de Rede neste podcast: https://anchor.fm/andreaskayy/ Networking-Infrastructure-and-Linux-031-PoDS-e242bh

8 Segurança e Privacidade

8.1 Certificados SSL de Chave Pública e Privada

8.2 O que é uma autoridade certificadora

8.3 Tokens da Web Java

8.4 Regulamentos GDPR

8.5 Privacidade por design

9Linux

9.1 Noções básicas do sistema operacional

9.2 Script de shell

9.3 Tarefas cronológicas

9.4 Gerenciamento de pacotes

Linux Tips é a segunda parte deste podcast: https://anchor.fm/andreaskayy/embed/episodes/IT Networking-Infrastructure-and-Linux-031-PoDS-e242bh

10 A Nuvem

10.1 Noções básicas de AWS, Azure, IBM, Google Cloud

10.2 nuvem x local

10.3 vantagens e desvantagens

10.4 Segurança

Ouça algumas reflexões sobre a nuvem neste podcast: https://anchor.fm/andreaskayy/embed/episod Be-Arrogant-The-Cloud-is-Safer-Then-Your-On-Premise-e16k9s

11 Projeto de Zona de Segurança

11.1 Como proteger um aplicativo multicamada

(UI em zona diferente do banco de dados SQL)

11.2 Segurança de cluster com Kerberos

Falei sobre design de zona de segurança e arquitetura lambda neste podcast: https://anchor.fm/andreas to-Design-Security-Zones-and-Lambda-Architecture–PoDS-032-e248q2

11.3 Tíquetes Kerberos

12 Processamento de Fluxo

12.1 Três métodos de transmissão — disponíveis

No processamento de fluxo, às vezes é permitido descartar mensagens, outras vezes não. Às vezes é bom processar uma mensagem várias vezes, outras vezes isso precisa ser evitado como o inferno.

O tópico de hoje são os diferentes métodos de streaming: No máximo uma vez, pelo menos uma vez e exatamente uma vez.

O que isso significa e por que é tão importante mantê-los em mente ao criar uma solução. É isso que você vai descobrir neste artigo.

12.2 Pelo menos uma vez

Pelo menos uma vez, significa que uma mensagem é processada no sistema uma ou várias vezes. Portanto, com pelo menos uma vez, não é possível que uma mensagem entre no sistema e não seja processada.

Não está caindo ou se perdendo em algum lugar do sistema.

Um exemplo em que pelo menos uma vez o processamento pode ser usado é quando você pensa em um gerenciamento de frota de carros. Você obtém dados de GPS de carros e esses dados de GPS são transmitidos com um registro de data e hora e as coordenadas de GPS.

É importante que você obtenha os dados do GPS pelo menos uma vez, para saber onde está o carro. Se você estiver processando esses dados várias vezes, eles sempre terão o carimbo de data/hora.

Por isso, não importa que seja processado várias vezes, por causa do timestamp. Ou que seria armazenado várias vezes, porque apenas substituiria o existente.

12.3 No máximo uma vez

O segundo método de streaming é no máximo uma vez. No máximo uma vez significa que não há problema em descartar algumas informações, para descartar algumas mensagens.

Mas é importante que uma mensagem seja processada apenas uma vez no máximo.

Um exemplo disso é o processamento de eventos. Algum evento está acontecendo e esse evento não é importante o suficiente, então pode ser descartado. Não tem nenhuma consequência quando é descartado.

Mas quando esse evento acontece, é importante que ele não seja processado várias vezes.
Então pareceria que o evento aconteceu cinco ou seis vezes em vez de apenas uma.

Pense em falhas de ignição do motor. Se isso acontecer uma vez, não é grande coisa. Mas se o sistema disser que isso acontece muito, você pensará que tem um problema com o motor.

12.4 Exatamente uma vez

Outra coisa é exatamente uma vez, isso significa que não há problema em descartar dados, não há problema em perder dados e também não há problema em processar uma mensagem que os dados disseram várias vezes

Um exemplo disso é, por exemplo, bancário. Quando você pensa em transações com cartão de crédito, não é bom desistir de uma transação.

Quando descartado, seu pagamento não está sendo processado. Também não é bom ter uma transação processada várias vezes, porque você está pagando várias vezes.

12.5 Verifique as ferramentas!

Tudo isso soa muito simples e lógico. Que tipo de processamento é feito deve ser um requisito para o seu caso de uso.

Isso precisa ser pensado no processo de design, porque nem todas as ferramentas suportam todos os três métodos. Muitas vezes, você precisa codificar seu aplicativo de maneira muito diferente com base no método de streaming.

Especialmente exatamente uma vez é muito difícil de fazer.

Portanto, a ferramenta de processamento de dados precisa ser escolhida com base se você precisa exatamente uma vez, pelo menos uma vez ou se precisa no máximo uma vez.

13 Big Data

13.1 O que é big data e onde está a diferença para ciência de dados e análise de dados?

Falei sobre a diferença neste podcast: https://anchor.fm/andreaskayy/embed/episodes/BI vs-Data-Science-vs-Big-Data-e199hq

13.2 Os 4Vs do Big Data — disponível

É um equívoco completo. O volume é apenas uma parte dos chamados quatro Vs do big data: volume, velocidade, variedade e veracidade.

Volume é sobre o tamanho. Quantos dados você tem.

A velocidade é sobre a velocidade com que os dados estão chegando até você.

A quantidade de dados em um tempo específico precisa ser processada ou está entrando no sistema.
É aqui que entra todo o conceito de streaming de dados e processamento em tempo real.

Variedade é o terceiro. Isso significa que os dados são muito diferentes. Que você tem tipos muito diferentes de estruturas de dados.

Como arquivos CSV, PDFs que você tem coisas em XML. Que você possui arquivos de log JSON ou que possui dados em algum tipo de armazenamento de valor de chave.

É sobre a variedade de tipos de dados de diferentes fontes que você basicamente deseja unir. Tudo para fazer uma análise com base nesses dados.

A veracidade é a quarta e esta é muito, muito difícil. O problema com o big data é que ele não é muito confiável.

Você não pode realmente confiar nos dados. Especialmente quando você vem da IoT, o

Lado da Internet das Coisas. Os dispositivos usam sensores para medição de temperatura, pressão, aceleração e assim por diante.

Você nem sempre pode ter cem por cento de certeza de que a medição real está correta.

Quando você tem dados que são, por exemplo, do SAP e contém dados criados manualmente, você também tem problemas. Como você sabe, nós, humanos, somos ruins em inserir coisas.

Todo mundo articula diferente. Cometemos erros, até na ortografia, e isso pode ser uma questão muito difícil para o analytics.

Falei sobre os 4Vs neste podcast: https://anchor.fm/andreaskayy/embed/episodes/4- Vs-Of-Big-Data-Are-Enough-e1h2ra

13.3 Por que Big Data? - disponível

O que sempre enfatizo é que os quatro V's são muito bons. Eles lhe dão uma direção geral.

Há uma questão muito mais importante: sucesso catastrófico.

O que quero dizer com sucesso catastrófico é que seu projeto, sua startup ou sua plataforma tem mais crescimento do que você antecipou. Crescimento exponencial é o que todo mundo está procurando para.

Porque com o crescimento exponencial existe o dinheiro. Começa pequeno e fica muito grande muito rápido. A clássica curva do taco de hóquei:

1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384. . . .ESTRONDO!

Pense nisso. Começa pequeno e bastante lento, mas fica muito grande muito rápido.

Você recebe muitos usuários ou clientes que estão pagando para usar seu serviço, a plataforma ou qualquer outra coisa. Se você tiver um sistema que não está equipado para dimensionar e processar os dados, todo o sistema falha.

Isso é um sucesso catastrófico. Você é tão bem-sucedido e cresce tão rápido que não consegue mais atender à demanda. E então você falha e está tudo acabado.

Agora é como se você pudesse inventar isso enquanto vai. Isso você pode prever em alguns meses ou semanas, o sistema atual não funciona mais.

13.3.1 Planejamento é tudo

Tudo acontece muito, muito rápido e você não consegue mais reagir. Existe um tipo necessário de planejamento e análise do potencial do seu caso de negócios necessário.

Então você precisa decidir se realmente tem big data ou não.

Você precisa decidir se usa ferramentas de big data. Isso significa que, ao conceituar toda a infraestrutura, pode parecer ridículo realmente se concentrar em ferramentas de big data.

Mas a longo prazo vai te ajudar muito. Um bom planejamento eliminará muitos problemas, especialmente se você pensar em streaming de dados e análises em tempo real.

13.3.2 O problema com ETL

Uma implantação típica de plataforma tradicional seria semelhante à imagem abaixo. Os dispositivos usam uma API de dados para carregar dados que são armazenados em um banco de dados SQL. Uma ferramenta de análise externa está consultando dados e carregando os resultados de volta no banco de dados SQL. Os usuários então usam a interface do usuário para exibir os dados armazenados no banco de dados.

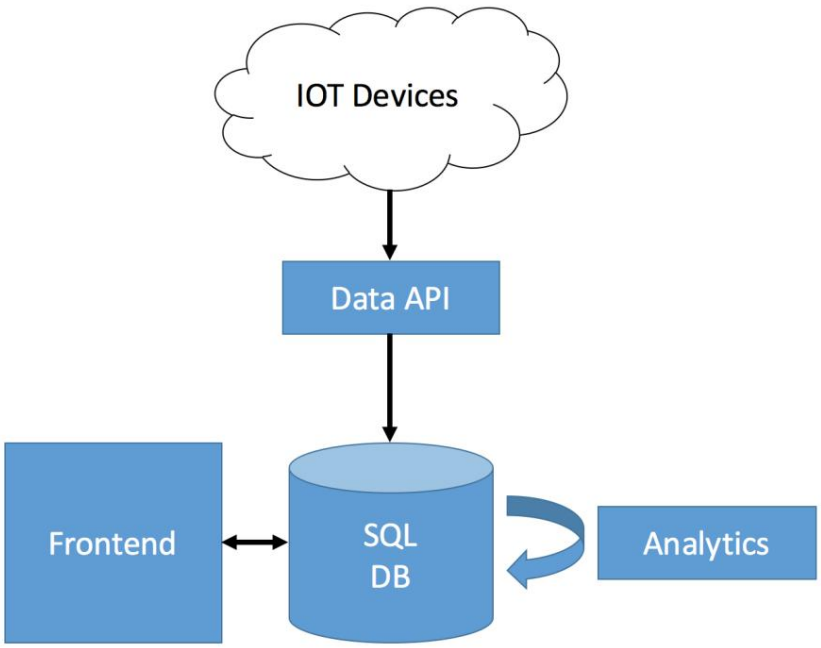

Figura 13.1: Arquitetura comum da plataforma SQL

Agora, quando o front-end consulta os dados do banco de dados SQL, acontecem as três etapas a seguir:

O banco de dados extrai todas as linhas necessárias do armazenamento. Os dados extraídos são transformados, por exemplo, classificados por carimbo de data/hora ou algo muito mais complexo.

Os dados extraídos e transformados são carregados no destino (a interface do usuário) para a criação do gráfico. Com quantidades explosivas de dados armazenados, o processo ETL começa a ser um problema real.

O Analytics está trabalhando com grandes conjuntos de dados, por exemplo, dias inteiros, semanas, meses ou mais. Os conjuntos de dados são muito grandes, como 100 GB ou Terabytes. Isso significa bilhões ou trilhões de linhas.

Isso faz com que o processo ETL para grandes conjuntos de dados leve cada vez mais tempo. Muito rapidamente, o desempenho do ETL fica tão ruim que não entrega mais resultados para análises.

Uma solução tradicional para superar esses problemas de desempenho é tentar aumentar o desempenho do servidor de banco de dados. Isso é o que se chama de aumento de escala.

13.3.3 Expansão

Para escalar o sistema e, portanto, aumentar as velocidades de ETL, os administradores recorrem a um hardware mais poderoso:

Acelerando o desempenho da extração adicionando discos mais rápidos para ler fisicamente os dados mais rapidamente. Aumentando a RAM para cache de linha. O que já está na memória não precisa ser lido por unidades de disco lentas. Usando CPUs mais potentes para melhor desempenho de transformação (mais RAM também ajuda aqui) Aumentar ou otimizar o desempenho de rede para entrega de dados mais rápida para o front-end e análise Aumentar a escala do sistema é bastante fácil.

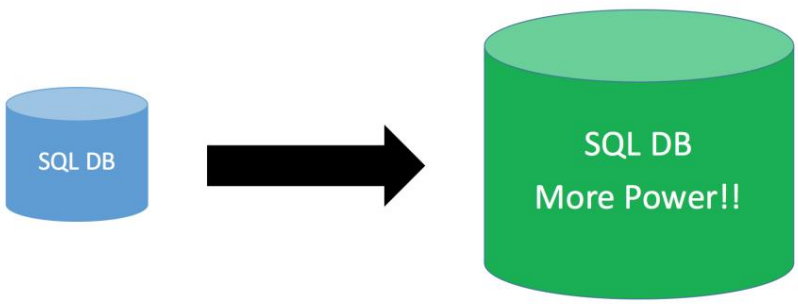

Figura 13.2: escalando um banco de dados SQL

Mas com o crescimento exponencial é óbvio que mais cedo ou mais tarde (mais cedo do que tarde) você terá os mesmos problemas novamente. Em algum momento, você simplesmente não pode mais escalar porque já possui um sistema monstro ou não pode comprar hardware mais caro.

O próximo passo que você poderia tomar seria expandir.

13.3.4 Expansão

Escalar horizontalmente é o oposto de escalar verticalmente. Em vez de construir sistemas maiores, o objetivo é distribuir a carga entre muitos sistemas menores.

A maneira mais simples de expandir um banco de dados SQL é usar uma rede de área de armazenamento (SAN) para armazenar os dados. Você pode usar até oito servidores SQL, anexá-los à SAN e permitir que eles lidem com consultas. Dessa forma, a carga é distribuída entre esses oito servidores.

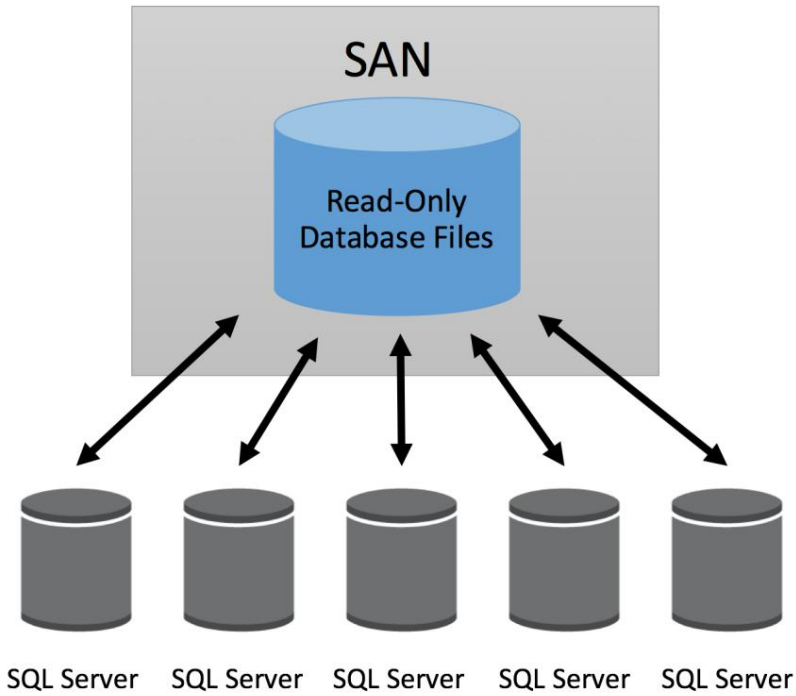

Figura 13.3: Dimensionando um banco de dados SQL

Uma grande desvantagem dessa configuração é que, como o armazenamento é compartilhado entre os servidores sql, ele só pode ser usado como um banco de dados somente leitura. As atualizações devem ser feitas periodicamente, por exemplo, uma vez por dia. Para fazer atualizações, todos os servidores SQL precisam se desconectar do banco de dados. Então, um está anexando o banco de dados no modo de leitura e gravação e atualizando os dados. Este procedimento pode demorar um pouco se muitos dados precisarem ser carregados.

Este link para uma página do MSDN da Microsoft tem mais opções de expansão de um banco de dados SQL para você.

Eu deliberadamente não quero entrar em detalhes sobre possíveis soluções de expansão. O ponto que estou tentando enfatizar é que, embora seja possível dimensionar bancos de dados SQL, é muito complicado.

Não há solução perfeita. Cada opção tem suas vantagens e desvantagens. Um grande problema comum é o esforço administrativo necessário para implementar e manter

uma solução dimensionada.

13.3.5 Por favor, não use Big Data

Se você não tiver problemas de dimensionamento, não use ferramentas de big data!

Big data é algo caro. Um cluster Hadoop, por exemplo, precisa de pelo menos cinco servidores para funcionar corretamente. Mais é melhor.

Acredite em mim, essas coisas custam muito dinheiro.

Especialmente quando você está falando sobre manutenção e desenvolvimento nas principais ferramentas de big daat em conta.

Se você não precisa, não faz absolutamente nenhum sentido!

Por outro lado: se você realmente precisa de ferramentas de big data, elas vão te salvar :)

14 Data Warehouse x Data Lake

15 plataformas Hadoop — disponíveis

Quando as pessoas falam sobre big data, uma das primeiras coisas que vêm à mente é o Hadoop. A busca do Google por Hadoop retorna cerca de 28 milhões de resultados.

Parece que você precisa do Hadoop para fazer big data. Hoje vou esclarecer por que o Hadoop está tão na moda.

Você verá que o Hadoop evoluiu de uma plataforma para um ecossistema. Seu design permite que muitos projetos Apache e ferramentas de terceiros se beneficiem do Hadoop.

Concluo com minha opinião sobre se você precisa aprender o Hadoop e se o Hadoop é a tecnologia certa para todos.

15.1 O que é Hadoop

Hadoop é uma plataforma para armazenamento distribuído e análise de conjuntos de dados muito grandes.

O Hadoop possui quatro módulos principais: Hadoop common, HDFS, MapReduce e YARN. A maneira como esses módulos são interligados é o que torna o Hadoop tão bem-sucedido.

As bibliotecas e funções comuns do Hadoop estão funcionando em segundo plano. É por isso que não vou me aprofundar neles. Eles estão lá principalmente para dar suporte aos módulos do Hadoop.

15.2 O que torna o Hadoop tão popular? - disponível

Armazenar e analisar dados tão grandes quanto você deseja é bom. Mas o que torna o Hadoop tão popular?

A funcionalidade principal do Hadoop é o impulsionador da adoção do Hadoop. Muitos projetos paralelos do Apache usam suas funções principais.

Por causa de todos esses projetos paralelos, o Hadoop se transformou mais em um ecossistema. Um ecossistema para armazenar e processar big data.

Para visualizar melhor este ecossistema, desenhei o gráfico a seguir. Mostra alguns projetos do ecossistema Hadoop que estão intimamente ligados ao Hadoop.

Não é uma lista completa. Existem muitas outras ferramentas que nem eu conheço. Talvez eu esteja desenhando um mapa completo no futuro.

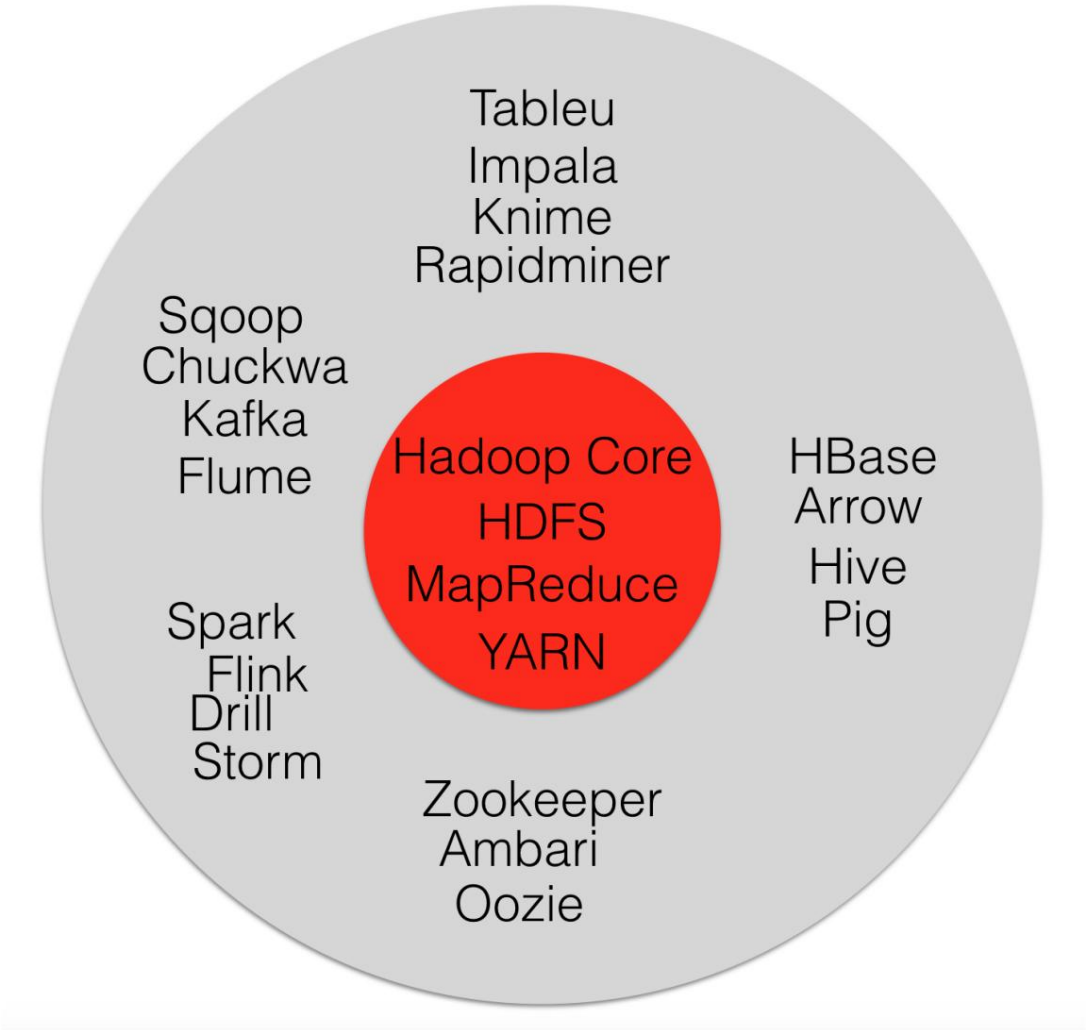

Figura 15.1: Componentes do Ecossistema Hadoop

15.3 Componentes do Ecossistema Hadoop

Lembra do meu projeto de plataforma de big data? O blueprint tem quatro estágios: ingestão, armazenamento, análise e exibição.

Por causa do ecossistema Hadoop, as diferentes ferramentas nessas etapas podem funcionar juntas perfeitamente.

Aqui está um exemplo:

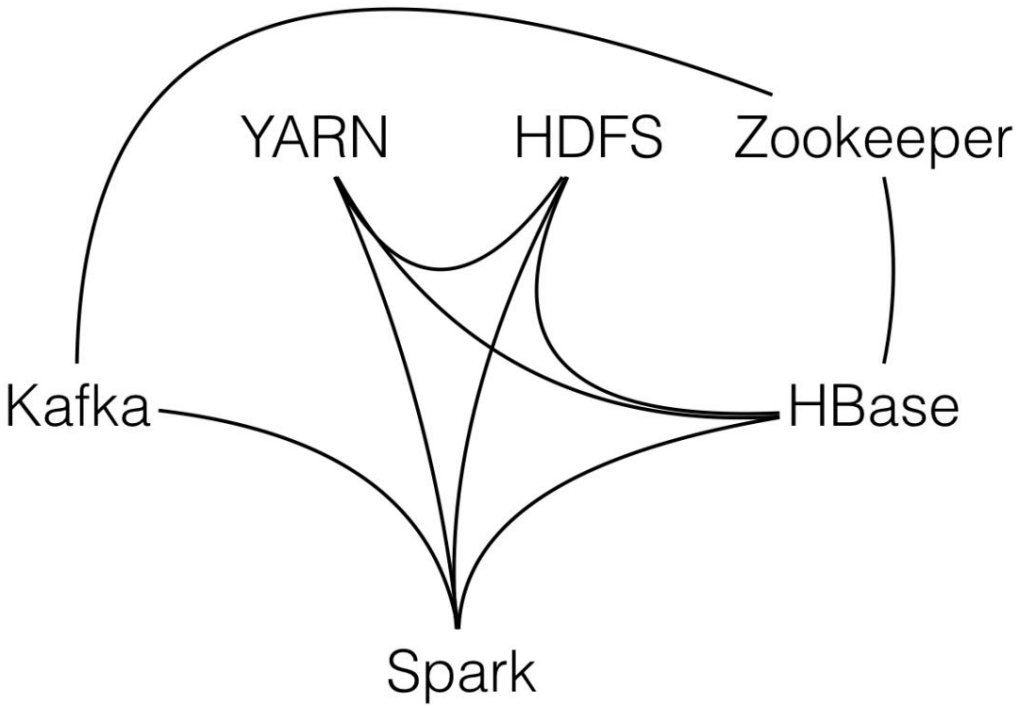

Figura 15.2: Conexões entre ferramentas

Você usa o Apache Kafka para ingerir dados e armazená-los no HDFS. Você faz a análise com o Apache Spark e, como back-end para a exibição, armazena os dados no Apache HBase.

Para ter um sistema funcionando, você também precisa do YARN para gerenciamento de recursos. Você também precisa do Zookeeper, um serviço de gerenciamento de configuração para usar Kafka e HBase

Como você pode ver na imagem abaixo, cada projeto está intimamente ligado ao outro.

O Spark, por exemplo, pode acessar diretamente o Kafka para consumir mensagens. É capaz de acessar o HDFS para armazenar ou processar dados armazenados.

Ele também pode gravar no HBase para enviar resultados analíticos para o front-end.

O legal desse ecossistema é que é fácil construir novas funções.

Deseja armazenar dados do Kafka diretamente no HDFS sem usar o Spark?

Não tem problema, existe um projeto para isso. O Apache Flume possui interfaces para Kafka e HDFS.

Ele pode atuar como um agente para consumir mensagens do Kafka e armazená-las no HDFS. Você nem precisa se preocupar com o gerenciamento de recursos do Flume.

O Flume pode usar o gerenciador de recursos YARN do Hadoop pronto para uso.

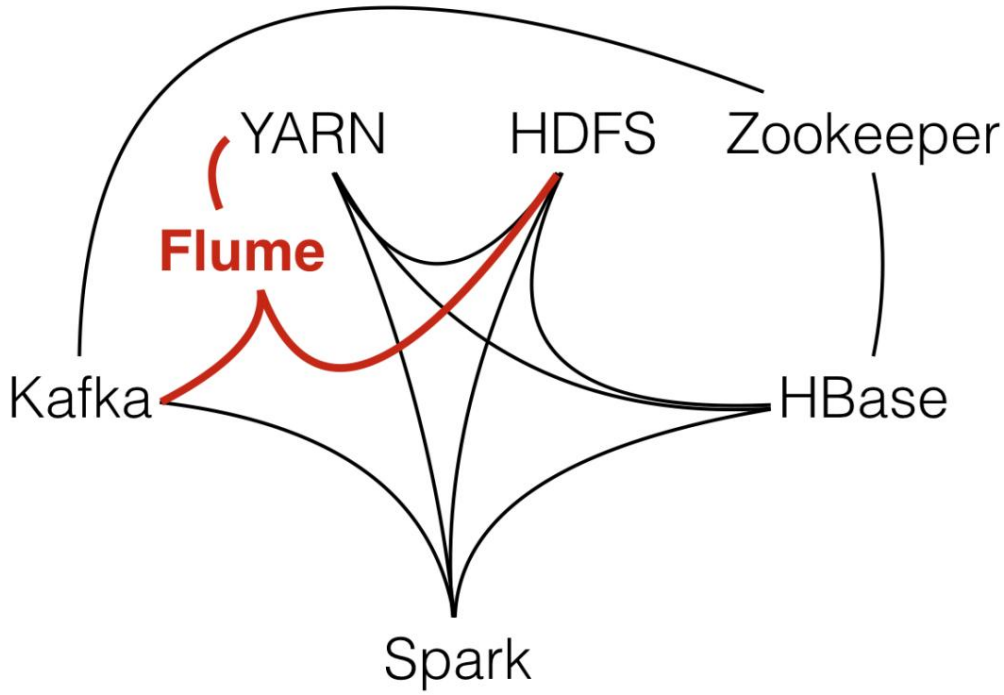

Figura 15.3: Integração do Flume

15.4 O Hadoop está em todo lugar?

Embora o Hadoop seja tão popular, não é a bala de prata. Não é a ferramenta que você deve usar para tudo.

Muitas vezes não faz sentido implantar um cluster Hadoop, porque pode ser um exagero.
O Hadoop não é executado em um único servidor.

Você basicamente precisa de pelo menos cinco servidores, melhor seis para executar um pequeno cluster. Por causa disso. os custos iniciais da plataforma são bastante elevados.

Uma opção que você tem é usar sistemas especializados como Cassandra, MongoDB ou outros bancos de dados NoSQL para armazenamento. Ou você muda para a Amazon e usa o Simple Storage Service da Amazon, ou S3.

Adivinha qual é a tecnologia por trás do S3. Sim, HDFS. É por isso que a AWS também tem o equivalente ao MapReduce chamado Elastic MapReduce.

O melhor do S3 é que você pode começar bem pequeno. Quando seu sistema cresce, você não precisa se preocupar com o escalonamento do servidor s3.

15.5 VOCÊ DEVE APRENDER HADOOP?

Sim, eu definitivamente recomendo que você entenda agora como o Hadoop funciona e como usá-lo. Como mostrei neste artigo, o ecossistema é bastante grande.

Muitos projetos de big data usam o Hadoop ou podem interagir com ele. É por isso que geralmente é uma boa ideia conhecer o máximo possível de tecnologias de big data.

Não em profundidade, mas a ponto de você saber como eles funcionam e como você pode usá-los.
Seu principal objetivo deve ser ser capaz de atingir o ponto inicial ao ingressar em um projeto de big data.

Além disso, a maioria das tecnologias é de código aberto. Você pode experimentá-los gratuitamente.

Como é a arquitetura de um sistema Hadoop

Quais ferramentas geralmente estão em um com Hadoop Cluster

Yarn Zookeeper HDFS Oozie Flume Hive

15.6 Como selecionar hardware Hadoop Cluster

16 O ETL ainda é relevante para Análise?

Falei sobre isso neste podcast: https://anchor.fm/andreaskayy/embed/episodes/Is ETL-Dead-For-Data-Science–Big-Data—PoDS-039-e2b604

17 Docker

17.1 O que é o docker e para que você o usa — disponível

Você já brincou com o Docker? Se você é um aprendiz de ciência de dados ou um cientista de dados, precisa conferir!

É incrível porque simplifica a maneira como você pode configurar ambientes de desenvolvimento para ciência de dados. Se você deseja configurar um ambiente de desenvolvimento, geralmente precisa instalar muitos pacotes e ferramentas.

17.1.1 Não estrague seu sistema

O que isso faz é basicamente bagunçar seu sistema operacional. Se você é iniciante, não sabe quais pacotes precisa instalar. Você não sabe quais ferramentas precisa instalar.

Se você quiser, por exemplo, começar com os notebooks Jupyter, precisará instalá-los no seu PC de alguma forma. Ou você precisa começar a instalar ferramentas como o PyCharmor Anaconda.

Tudo isso é adicionado ao seu sistema e você bagunça seu sistema cada vez mais. O que o Docker traz para você, especialmente se você estiver em um sistema Mac ou Linux, é a simplicidade.

17.1.2 Imagens pré-configuradas

Porque é muito fácil de instalar nesses sistemas. Outra coisa legal sobre as imagens do docker é que você pode simplesmente pesquisá-las na loja do Docker, baixá-las e instalá-las em seu sistema.

Executá-los em um ambiente totalmente pré-configurado. Você não precisa pensar sobre as coisas que você vai para a biblioteca do Docker, você procura GPU de aprendizagem profunda e Python.

Você obtém uma lista de imagens que você pode baixar. Você baixa um, inicia-o, acessa o navegador, acessa o URL e começa a codificar.

Comece a fazer o trabalho. A única outra coisa que você precisa fazer é vincular algumas unidades a essa instância para poder trocar arquivos. E então é isso!

Não há nenhuma maneira que você pode travar ou bagunçar seu sistema. Está tudo encapsulado no Docker. Isso funciona porque o Docker tem acesso nativo ao seu hardware.

17.1.3 Leve com você

Não é um ambiente completamente virtualizado como um VirtualBox. Uma imagem tem a vantagem de poder levá-la para onde quiser. Portanto, se você estiver no seu PC em casa, use-o lá.

Faça uma construção rápida, pegue a imagem e vá para outro lugar. Instale a imagem que costuma ser bastante rápida e use-a como se estivesse em casa.

É incrível!

17.2 Implantação do Contêiner Kubernetes

Eu mesmo estou entrando muito mais no Docker. Por razões um pouco diferentes.

O que estou procurando é usar o Docker com o Kubernetes. Kubernets, você pode automatizar todo o processo de implantação do contêiner.

A ideia é que você tenha um cluster de máquinas. Digamos que você tenha 10 clusters de servidor e execute o Kubernetes neles.

O Kubernetes permite ativar contêineres do Docker sob demanda para executar tarefas. Você pode configurar quantos recursos como CPU, RAM, rede, contêiner do Docker podem usar.

Basicamente, você pode ativar contêineres no cluster sob demanda. Sempre que você precisar fazer uma tarefa de análise.

Perfeito para Ciência de Dados.

17.3 Como criar, iniciar, parar um Container

17.4 Microsserviços do Docker?

17.5 Kubernetes

17.6 Por que e como fazer um contêiner Docker orquestração

Podcast sobre como os alunos de ciência de dados usam o Docker (para cientistas de dados): https://anchor.fm/andreaskay Data-Science-Go-Docker-e10n7u

18 APIs REST

Confira meu podcast sobre como as APIs dominam o mundo: https://anchor.fm/andreaskayy/embed/episodes APIs-Rule-The-World–PoDS-033-e24ttq

18.1 Postar/Obter HTTP

18.2 Projeto de API

18.3 Implementação

18.4 Segurança OAuth

19 bancos de dados

19.1 Bancos de Dados SQL

19.1.1 Projeto de banco de dados

19.1.2 Consultas SQL

19.1.3 Procedimentos Armazenados

19.1.4 Conexões do Servidor ODBC/JDBC

19.2 Armazenamentos NoSQL

19.2.1 Armazenamentos de KeyValue (HBase)

19.2.2 Armazenamento de Documentos HDFS — disponível

O sistema de arquivos distribuídos Hadoop, ou HDFS, permite armazenar arquivos no Hadoop. A diferença entre HDFS e outros sistemas de arquivos como NTFS ou EXT é que é um dis
um homenageado.

O que isso significa exatamente?

Um sistema de arquivos típico armazena seus dados no disco rígido real. É dependente de hardware.

Se você tiver dois discos, precisará formatar cada disco com seu próprio sistema de arquivos. Eles são completamente separados.

Você então decide em qual disco armazenar fisicamente seus dados.

O HDFS funciona de maneira diferente de um sistema de arquivos típico. O HDFS é independente de hardware.

Ele não apenas abrange muitos discos em um servidor. Ele também abrange muitos servidores.

O HDFS colocará automaticamente seus arquivos em algum lugar no coletivo do servidor Hadoop.

Ele não apenas armazenará seu arquivo, mas o Hadoop também o replicará duas ou três vezes (você pode definir isso). A replicação significa que as réplicas do arquivo serão distribuídas para diferentes servidores.

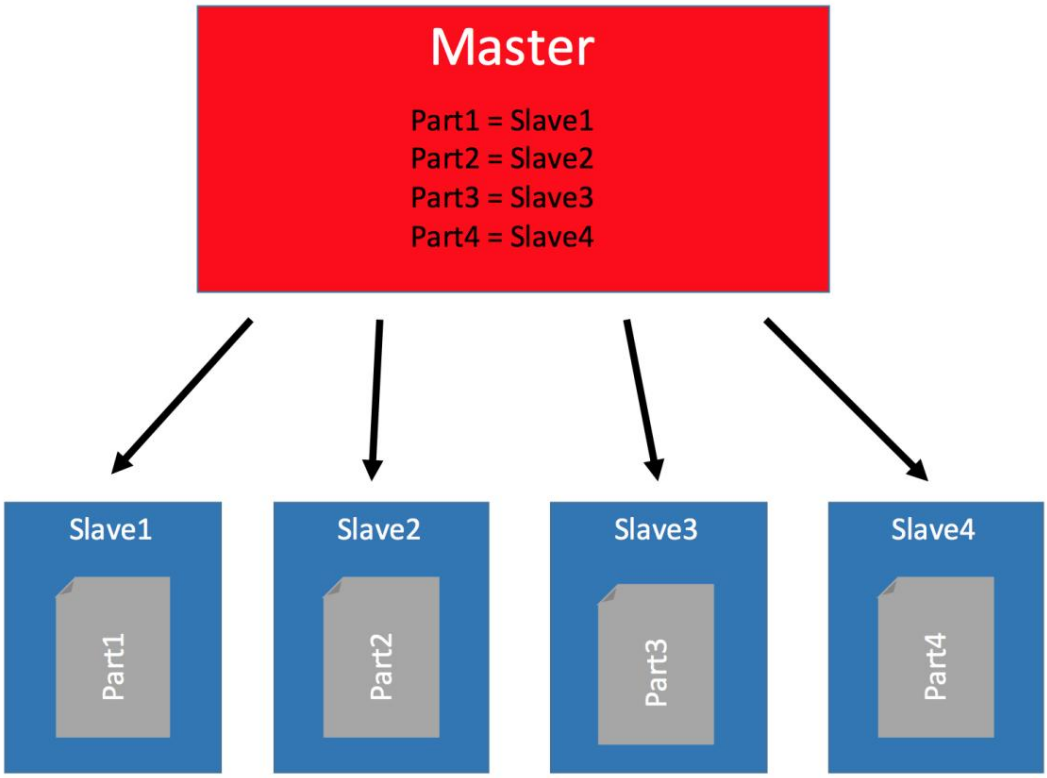

Figura 19.1: Mestre HDFS e nós de dados

Isso lhe dá tolerância a falhas superior. Se um servidor cair, seus dados permanecerão disponíveis em um servidor diferente.

Outra grande coisa sobre o HDFS é que não há limite de tamanho para os arquivos. Você pode ter arquivos de log do servidor com terabytes de tamanho.

Como os arquivos podem ficar tão grandes? O HDFS permite anexar dados aos arquivos. Portanto, você pode despejar dados continuamente em um único arquivo sem preocupações.

O HDFS armazena fisicamente arquivos diferentes de um sistema de arquivos normal. Ele divide o arquivo em blocos.

Esses blocos são então distribuídos e replicados no cluster Hadoop. A divisão acontece automaticamente.

Na configuração você pode definir o tamanho dos blocos. 128 megabytes ou 1 gigabyte?

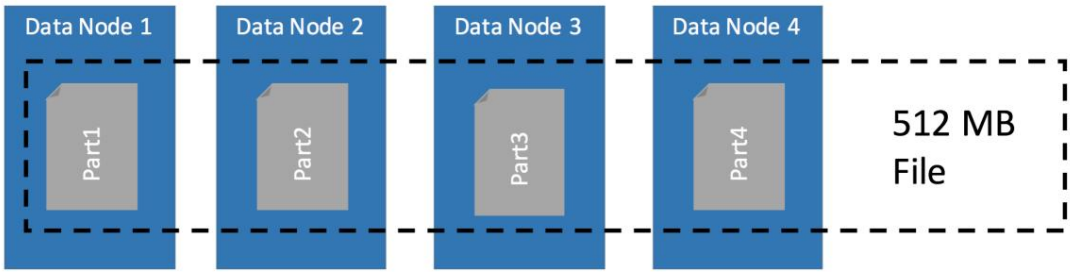

Figura 19.2: Distribuição de blocos para um arquivo de 512 MB

Não há problema algum.

Essa mecânica de dividir um arquivo grande em blocos e distribuí-los pelos servidores é ótima para processamento. Consulte a seção MapReduce para obter um exemplo.

19.2.3 Armazenamento de Documentos MongoDB

19.2.4 Armazém da Colmeia

19.2.5 Impala

19.2.6 Deve

19.2.7 Bancos de dados de séries temporais

DalmatinerDB InfluxDB Prometheus Riak TS OpenTSDB KairosDB Elasticsearch Druid

19.2.8 Bancos de Dados MPP (Greenplum)

20 Processamento de Dados/Análise - Estruturas

20.1 MapReduce

Desde os primórdios do ecosistema Hadoop, a estrutura MapReduce é um dos principais componentes do Hadoop junto com o sistema de arquivos Hadoop HDFS.

O Google, por exemplo, usou o MapReduce para analisar o conteúdo html armazenado de sites por meio da contagem de todas as tags html e todas as palavras e combinações delas (por exemplo, manchetes). A saída foi usada para criar a classificação da página para a Pesquisa do Google.

Foi quando todo mundo começou a otimizar seu site para a pesquisa do Google. A otimização séria do mecanismo de pesquisa foi suportada. Esse foi o ano de 2004.

O funcionamento do MapReduce é que ele processa os dados em duas fases: a fase de mapa e a fase de redução.

Na fase de mapa, a estrutura está lendo dados do HDFS. Cada conjunto de dados é chamado de registro de entrada.

Depois, há a fase de redução. Na fase de redução, a computação real é feita e os resultados são armazenados. O destino de armazenamento pode ser um banco de dados ou HDFS de volta ou qualquer outra coisa.

Afinal, é Java – então você pode implementar o que quiser.

A mágica do MapReduce é como as fases map e reduce são implementadas e como ambas as fases funcionam juntas.

As fases de mapa e redução são paralelizadas. O que isso significa é que você tem várias fases de mapa (mapeadores) e fases de redução (redutores) que podem ser executadas em paralelo em seu máquinas de cluster.

Aqui está um exemplo de como um processo de mapeamento e redução funciona com dados:

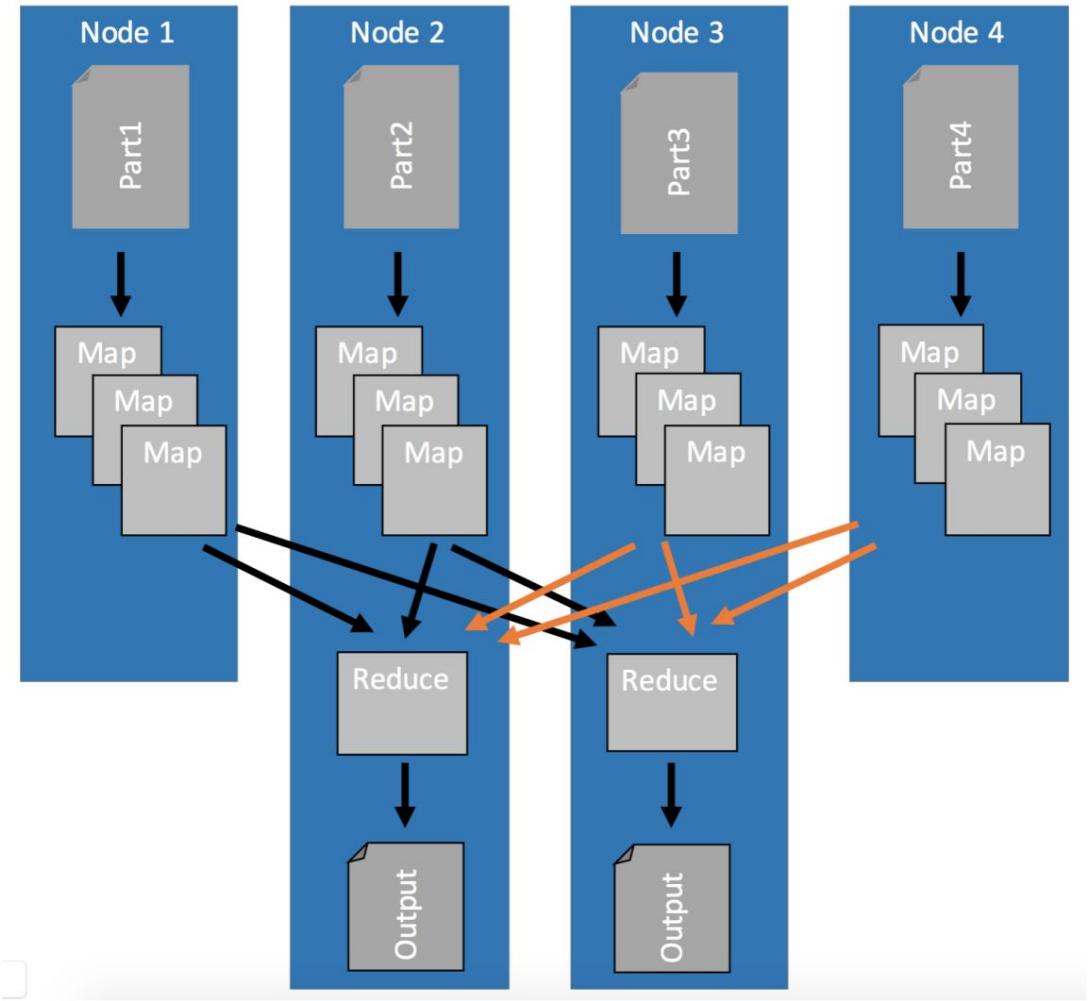

Figura 20.1: Mapeamento de arquivos de entrada e redução de registros mapeados

20.1.1 Como funciona o MapReduce – disponível

Em primeiro lugar, todo o processo de mapeamento e redução depende muito do uso de pares chave/valor. É para isso que servem os mapeadores.

Na fase do mapa, os dados de entrada, por exemplo, um arquivo, são carregados e transformados em pares chave/valor.

Quando cada fase do mapa é concluída, ele envia os pares de chave/valor criados para os redutores, onde eles são classificados por chave. Isso significa que um registro de entrada para a fase de redução é uma lista de valores dos mapeadores que possuem a mesma chave.

Em seguida, a fase de redução está fazendo o cálculo dessa chave e seus valores e exibindo os resultados.

Quantos mapeadores e redutores você pode usar em paralelo? O número de processos paralelos de mapa e redução depende de quantos núcleos de CPU você tem em seu cluster. Todo mapeador e todo redutor está usando um núcleo.

Isso significa que quanto mais núcleos de CPU você realmente tiver, mais mapeadores você pode usar e mais rápido o processo de extração pode ser feito. Quanto mais redutores você estiver usando, mais rápido a computação real será feita.

Para deixar isso mais claro, preparei um exemplo:

20.1.2 Exemplo

Como eu disse antes, o MapReduce funciona em duas etapas, mapear e reduzir. Freqüentemente, esses estágios são explicados com uma tarefa de contagem de palavras.

Pessoalmente, eu odeio este exemplo porque contar coisas é muito trivial e realmente não mostra o que você pode fazer com o MapReduce. Portanto, vamos usar um caso de uso mais real do mundo da internet das coisas (IoT).

Os aplicativos IoT criam uma enorme quantidade de dados que precisam ser processados. Esses dados são gerados por sensores físicos que fazem medições, como a temperatura ambiente às 8h.

Cada medição consiste em uma chave (o carimbo de hora quando a medição foi realizada) e um valor (o valor real medido pelo sensor).

Como você geralmente tem mais de um sensor em sua máquina ou conectado ao seu sistema, a chave deve ser uma chave composta. Chaves compostas contêm adicionalmente ao

informações de tempo de medição sobre a fonte do sinal.

Mas vamos esquecer as chaves compostas por enquanto. Hoje temos apenas um sensor. Cada medição gera pares chave/valor como: Timestamp-Value.

O objetivo deste exercício é criar valores médios diários dos dados desse sensor.

A imagem abaixo mostra como funciona o processo de mapear e reduzir.

Primeiro, o estágio do mapa carrega dados não classificados (registros de entrada) da fonte (por exemplo, HDFS) por chave e valor (chave:2016-05-01 01:02:03, valor:1).

Então, como o objetivo é obter médias diárias, as informações de hora:minuto:segundo são cortadas do carimbo de data/hora.

Isso é tudo o que acontece na fase do mapa, nada mais.

Depois que todas as fases do mapa paralelo são concluídas, cada par chave/valor é enviado para o redutor que está manipulando todos os valores para essa chave específica.

Cada registro de entrada do redutor possui uma lista de valores e você pode calcular (1+5+9)/3, (2+6+7)/3 e (3+4+8)/3. Isso é tudo.

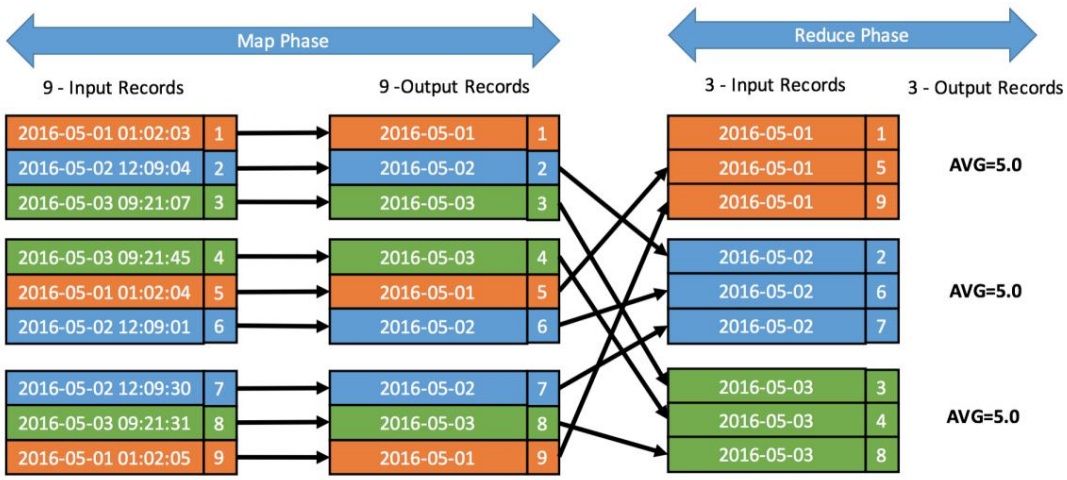

Figura 20.2: Exemplo de MapReduce de dados de séries temporais

O que você acha que precisa fazer para gerar médias de minutos?

Sim, você precisa cortar a chave de forma diferente. Você então precisaria cortá-lo assim: “2016- 05-01 01:02”. Manter as informações de hora e minuto na chave.

O que você também pode ver é por que a redução de mapa é tão boa para fazer trabalho paralelo. Neste caso, a etapa do mapa poderia ser feita por nove mapeadores em paralelo, pois cada mapa é independente de todos os outros.

O estágio de redução ainda pode ser feito por três tarefas em paralelo. Um para laranja, azul e outro para verde.

Isso significa que, se seu conjunto de dados for 10 vezes maior e você tiver 10 vezes mais máquinas, o tempo para fazer o cálculo será o mesmo.

20.1.3 Qual é a limitação do MapReduce? - disponível

O MapReduce é incrível para tarefas analíticas mais simples, como contar coisas. Ele só tem uma falha: tem apenas dois estágios, Mapear e Reduzir.

Figura 20.3: O processo de redução do mapa

Primeiro MapReduce carrega os dados do HDFS na função de mapeamento. Lá você prepara os dados de entrada para o processamento no redutor. Após a conclusão da redução, os resultados são gravados no armazenamento de dados.

O problema com MapReduce é que não há uma maneira simples de encadear vários mapas e reduzir processos juntos. Ao final de cada processo de redução, os dados devem ser armazenados em algum lugar.

Esse fato torna muito difícil realizar processos analíticos complicados. Você precisaria encadear trabalhos MapReduce juntos.

Encadear trabalhos com armazenamento e carregamento de resultados intermediários simplesmente não faz sentido.

Outro problema com o MapReduce é que ele não é capaz de transmitir análises. Os trabalhos levam algum tempo para girar, fazer as análises e desligar. Basicamente, minutos de espera são totalmente normais.

Este é um grande ponto negativo em um mundo de processamento de dados cada vez mais em tempo real.

20.2 Apache Spark

Falei sobre os três métodos de streaming de dados neste podcast: https://anchor.fm/andreaskayy/em Methods-of-Streaming-Data-e15r6o

20.2.1 Qual é a diferença para MapReduce? - disponível

O Spark é um framework de memória completo. Os dados são carregados, por exemplo, hdfs, na memória dos trabalhadores.

Não há mais um mapa fixo e um estágio reduzido. Seu código pode ser tão complexo quanto você querer.

Uma vez na memória, os dados de entrada e os resultados intermediários permanecem na memória (até que o trabalho seja concluído). Eles não são gravados em uma unidade como no MapReduce.

Isso torna o Spark a escolha ideal para fazer análises complexas. Permite, por exemplo, fazer processos iterativos. Modificar um conjunto de dados várias vezes para criar uma saída é totalmente fácil.

A capacidade de análise de streaming também é o que torna o Spark tão bom. O Spark tem nativamente a opção de agendar um trabalho para ser executado a cada X segundos ou X milissegundos.

Como resultado, o Spark pode fornecer resultados de streaming de dados em “tempo real”.

20.2.2 Como o Spark se encaixa no Hadoop? - disponível

Existem alguns artigos muito enganosos intitulados Spark ou Hadoop, Spark é melhor que Hadoop ou mesmo Spark está substituindo o Hadoop.

	Storage	Analytics	Resource Management
Hadoop	Hadoop Distributed File System HDFS	MapReduce	YARN (Yet Another Resource Negotiator)
Spark	—-	Spark	Spark Resource Management

Figura 20.4: recursos Hadoop x Spark

Então, é hora de mostrar as diferenças entre Spark e Hadoop. Depois disso, você saberá quando e para que deve usar o Spark e o Hadoop.

Você também entenderá por que Hadoop ou Spark é a pergunta totalmente errada.

20.2.3 Onde está a diferença?

Para deixar claro como o Hadoop difere do Spark, criei esta tabela de recursos simples:

Hadoop é usado para armazenar dados no Hadoop Distributed File System (HDFS). Ele pode analisar os dados armazenados com MapReduce e gerenciar recursos com YARN.

No entanto, o Hadoop é mais do que apenas armazenamento, análise e gerenciamento de recursos. Existe todo um ecossistema de ferramentas em torno do núcleo do Hadoop. Escrevi sobre esse ecossistema neste artigo: O que é Hadoop e por que ele é tão bizarramente popular. você deveria checar
fora também.

Comparado ao Hadoop, o Spark é “apenas” uma estrutura de análise. Não tem capacidade de armazenamento. Embora tenha um gerenciamento de recursos autônomo, você geralmente não usa esse recurso.

20.2.4 Spark e Hadoop são uma combinação perfeita

Então, se Hadoop e Spark não são a mesma coisa, eles podem funcionar juntos?

Absolutamente! Veja como a primeira imagem ficará se você combinar o Hadoop com o Spark:

Como armazenamento, você usa o sistema de arquivos distribuído Hadoop. A análise é feita com o Apache Spark e o Yarn está cuidando do gerenciamento de recursos.

Por que isso funciona tão bem juntos?

Do ponto de vista da arquitetura da plataforma, o Hadoop e o Spark geralmente são gerenciados no mesmo cluster. Isso significa que em cada servidor em que um nó de dados HDFS está em execução, uma faísca
thread de trabalho também é executado.

No processamento distribuído, a transferência de rede entre as máquinas é um grande gargalo.
A transferência de dados dentro de uma máquina reduz significativamente esse tráfego.

O Spark é capaz de determinar em qual nó de dados os dados necessários são armazenados. Isso permite uma carga direta dos dados do armazenamento local na memória da máquina.

Isso reduz muito o tráfego de rede.

20.2.5 Faísca no YARN:

Você precisa garantir que seus recursos físicos sejam distribuídos perfeitamente entre os serviços. Esse é especialmente o caso quando você executa Spark workers com outros serviços do Hadoop na mesma máquina.

Simplesmente não faria sentido ter dois gerenciadores de recursos gerenciando os recursos do mesmo servidor. Mais cedo ou mais tarde, eles vão se meter no caminho um do outro.

É por isso que o gerenciador de recursos autônomo do Spark raramente é usado.

Portanto, a questão não é Spark ou Hadoop. A pergunta tem que ser: você deve usar Spark ou MapReduce juntamente com HDFS e YARN do Hadoop.

20.2.6 Minha regra prática simples:

Se você estiver realizando trabalhos em lote simples, como contar valores ou calcular médias: use o MapReduce.

Se você precisar de análises mais complexas, como aprendizado de máquina ou processamento de fluxo rápido: use o Apache Spark.

20.2.7 Idiomas disponíveis - disponíveis

Os trabalhos do Spark podem ser programados em vários idiomas. Isso torna a criação de processos analíticos muito fácil de usar para os cientistas de dados.

O Spark suporta Python, Scala e Java. Com a ajuda do SparkR, você pode até conectar seu programa R a um cluster Spark.

Se você é um cientista de dados muito familiarizado com o Python, basta usar o Python, é ótimo. Se você sabe codificar Java, sugiro que comece a usar Scala.

Os trabalhos do Spark são mais fáceis de codificar em Scala do que em Java. No Scala, você pode usar funções anônimas para fazer o processamento.

Isso resulta em menos sobrecarga, é um código muito mais limpo e simples.

Com o Java 8, as chamadas de função simplificadas foram introduzidas com expressões lambda. Ainda assim, muitas pessoas, inclusive eu, preferem Scala a Java.

20.2.8 Como fazer processamento de fluxo

20.2.9 Como fazer o processamento em lote

20.2.10 Como o Spark usa os dados do Hadoop – disponível

Outra coisa é a localidade dos dados. Eu sempre enfatizo que processar dados localmente onde eles estão armazenados é a coisa mais eficiente a se fazer.

Isso é exatamente o que o Spark está fazendo. Você pode e deve executar os trabalhadores do Spark diretamente nos nós de dados do seu cluster Hadoop.

O Spark pode então identificar nativamente em qual nó de dados os dados necessários estão armazenados. Isso permite que o Spark use o worker em execução na máquina onde os dados estão armazenados para carregar os dados na memória.

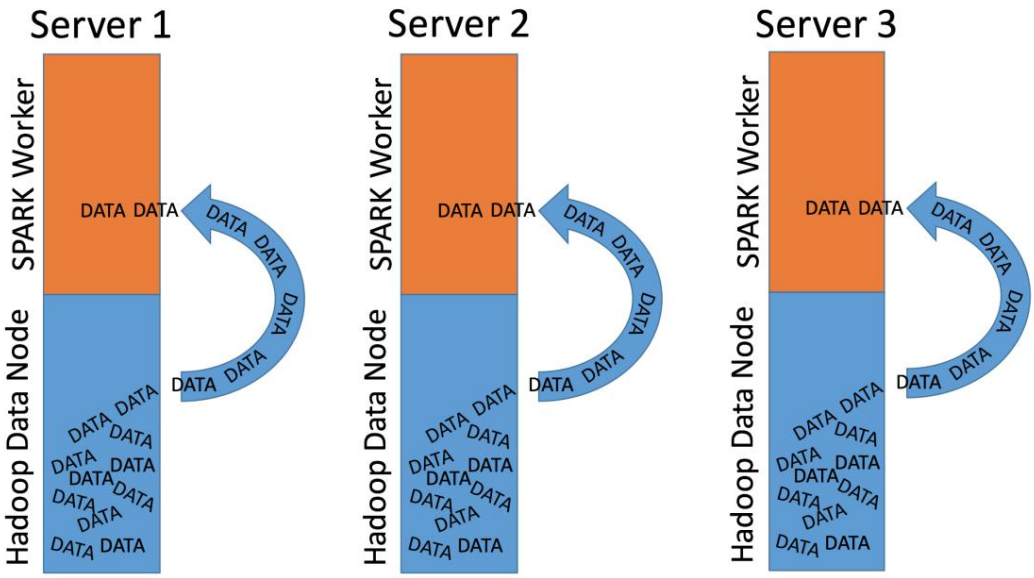

Figura 20.5: Spark usando a localidade de dados do Hadoop

A desvantagem dessa configuração é que você precisa de servidores mais caros. Porque o processamento do Spark precisa de servidores mais fortes com mais RAM e CPUs do que um Hadoop “puro”

configurar.

20.3 O que é um RDD e o que é um DataFrame?

20.4 Codificação Spark com Scala

20.5 Codificação Spark com Python

20.6 Como e por que usar o SparkSQL?

20.7 Aprendizado de máquina no Spark? (Fluxo tensor)

20,8 MLib:

A biblioteca de aprendizado de máquina MLlib está incluída no Spark, então muitas vezes não há necessidade de importar outra biblioteca.

Tenho que admitir, porque não sou um cientista de dados, não sou um especialista em aprendizado de máquina.

Pelo que vi e li, a estrutura de aprendizado de máquina MLlib é um bom presente para cientistas de dados que desejam treinar e aplicar modelos com o Spark.

20.9 Configuração do Spark – disponível

Do ponto de vista de um arquiteto de soluções, o Spark é perfeito para as plataformas de big data do Hadoop. Isso tem muito a ver com a implantação e o gerenciamento do cluster.

Empresas como Cloudera, MapR ou Hortonworks incluem o Spark em suas distribuições do Hadoop. Por causa disso, o Spark pode ser implantado e gerenciado com os clusters Hadoop gerenciados pela web.

Isso torna o processo de implantação e configuração de um cluster Spark muito rápido e fácil de administrar.

20.10 Gerenciamento de recursos do Spark - disponível

Ao executar uma estrutura de computação, você precisa de recursos para fazer computação: tempo de CPU, RAM, E/S e assim por diante. Pronto para uso, o Spark pode gerenciar recursos com sua versão autônoma gerente de Recursos.

Se o Spark estiver sendo executado em um ambiente Hadoop, você não precisa usar o próprio gerenciador de recursos autônomo do Spark. Você pode configurar o Spark para usar o recurso YARN do Hadoop agement.

Por que você faria isso? Ele permite que o YARN aloque recursos com eficiência para seus processos Hadoop e Spark.

Ter um único gerenciador de recursos em vez de dois independentes torna muito mais fácil configurar o gerenciamento de recursos.

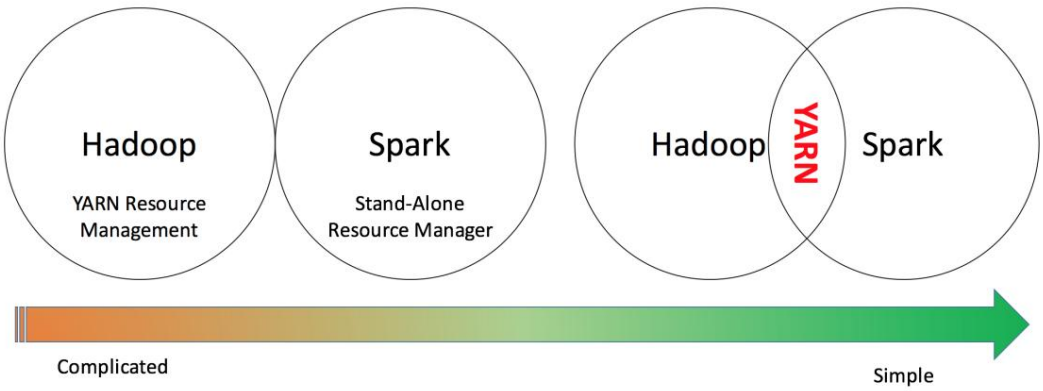

Figura 20.6: Spark Resource Management com YARN

21 Apache Kafka

21.1 Por que uma ferramenta de fila de mensagens?

21.2 Arquitetura Kakfa

21.3 O que são tópicos

21.4 O que Zookeeper tem a ver com Kafka

21.5 Como produzir e consumir mensagens

Meu vídeo do YouTube como configurar o Kafka em casa: https://youtu.be/7F9tBwTUSeY

Meu vídeo do YouTube como escrever para Kafka: https://youtu.be/RboQBZvZCh0

22 Aprendizado de Máquina

Meu podcast sobre como fazer aprendizado de máquina na produção: https://anchor.fm/andreaskayy/embed/e Learning-In-Production-e11bbk

22.1 Modelos de treinamento e aplicação

22.2 O que é aprendizado profundo

22.3 Como fazer Machine Learning na produção — disponível

O aprendizado de máquina na produção está usando processamento de fluxo e lote. Na camada de processamento em lote você está criando os modelos, pois tem todos os dados disponíveis para treinamento.

No fluxo na camada de processamento, você está usando os modelos criados, está aplicando-os a novos dados.

A ideia que você precisa incorporar é que este é um ciclo constante. Treinando, aplicando, treinando novamente, empurrando para a produção e aplicando.

O que você não quer fazer é não querer fazer isso manualmente. Você precisa descobrir um processo de retreinamento automático e envio automático para a produção de modelos.

Na fase de retreinamento o sistema avalia automaticamente o treinamento. Se o modelo não se encaixa mais, ele funciona enquanto for necessário para criar um bom modelo.

Depois que a avaliação do modelo é concluída e é bom, o modelo é enviado para a produção. No fluxo de processamento.

22.4 Por que o aprendizado de máquina na produção é mais difícil então você pensa - disponível

Como automatizar o aprendizado de máquina é algo que me motiva todos os dias.

O que você faz no desenvolvimento ou na educação é criar um modelo e ajustá-lo aos dados. Então esse modelo é basicamente feito para sempre.

De onde venho, do mundo IoT, o problema é que as máquinas são muito diferentes.
Eles se comportam de maneira muito diferente e experimentam desgaste.

22.5 Modelos não funcionam para sempre

As máquinas têm certos processos que diminuem a integridade real da máquina. O desgaste da máquina é um grande problema. Modelos construídos em cima de uma boa máquina não funcionam para sempre.

Quando a Máquina se desgasta, os modelos precisam ser ajustados. Eles precisam ser mantidos, retreinados.

22.6 Onde estão as plataformas que suportam isso?

O retreinamento e a redistribuição automáticos são um problema muito grande, um problema muito grande para muitas empresas. Porque a maioria das plataformas existentes não tem esse recurso (na verdade, não vi nenhuma até agora).

Veja o aprendizado de máquina da AWS, por exemplo. O processo é: construir, treinar, ajustar a implantação. Onde está o ciclo de retreinamento?

Você pode criar modelos e usá-los na produção. Mas este loop está quase em lugar nenhum
ser visto.

É um problema muito grande que precisa ser resolvido. Se você deseja fazer aprendizado de máquina em produção, pode começar com a interação manual do treinamento, mas em algum momento precisará automatizar tudo.

22.7 Gerenciamento de Parâmetros de Treinamento

Para treinar um modelo, você está manipulando os parâmetros de entrada dos modelos.

Veja o aprendizado profundo, por exemplo. Para treinar você está manipulando, por exemplo:

Quantas camadas você usa. A profundidade das camadas, o que significa quantos neurônios você tem em uma camada.
Qual função de ativação você usa, quanto tempo você está treinando e assim
sobre.

Você também precisa acompanhar quais dados você usou para treinar qual modelo.

Todos esses parâmetros precisam ser manipulados automaticamente, modelos treinados e testados.

Para fazer tudo isso, você basicamente precisa de um banco de dados que monitore essas variáveis.

Como automatizar isso, para mim, é como o grande segredo. Eu ainda estou trabalhando para descobrir isso.

22.8 Qual é a sua solução?

Você já teve o problema de retreinamento automático e implantação de modelos como
bem?

Você conseguiu usar uma plataforma de nuvem como Google, AWS ou Azure?

Seria muito legal se você compartilhasse sua experiência :)

22.9 Como convencer as pessoas que o aprendizado de máquina funciona - disponível

Muitas pessoas ainda não estão convencidas de que o aprendizado de máquina funciona de maneira confiável. Mas eles querem uma visão analítica e, na maioria das vezes, o aprendizado de máquina é o caminho a percorrer.

Isso significa que, quando você está trabalhando com clientes, precisa ser muito convincente.
Especialmente se eles próprios não gostam de aprendizado de máquina.

Mas na verdade é bem fácil.

22.10 Sem Regras, Sem Modelos Físicos

Muitas pessoas ainda têm a impressão de que a análise só funciona quando é baseada na física. Quando existem regras matemáticas estritas para um problema.

Especialmente em países de engenharia pesada como a Alemanha, esta é a norma:

“Sere tem que ser uma regra para cada coisa!” (imagine um sotaque alemão) Quando você está fazendo engenharia, você está calculando coisas com base na física e não em dados. Se você está construindo uma asa de avião, é melhor usar cálculos para que ela não caia.

E tudo bem.

Continue fazendo isso!

O aprendizado de máquina existe há décadas. Não funcionou tão bem quanto as pessoas esperavam. Temos que admitir isso. Mas existe esse preconceito de que ainda não funciona.

O que não é verdade: o aprendizado de máquina funciona.

De alguma forma, você precisa convencer as pessoas de que é uma abordagem viável. Que aprender com os dados para fazer previsões está funcionando perfeitamente.

22.11 Você tem os dados. USE-O!

Como cientista de dados, você tem um ás na manga, é o óbvio:

São os dados e as estatísticas.

Você pode usar esses dados e essas estatísticas para combater os preconceitos das pessoas. É muito poderoso se alguém disser: “Isso não funciona”

Você traz os dados. Você mostra as estatísticas e mostra que funciona de forma confiável.

Muitas discussões terminam aí.

Os dados não mentem. Você não pode lutar contra os dados. Os dados estão sempre certos.

22.12 Dados são mais fortes que opiniões

É também por isso que acredito que a direção autônoma será mais rápida do que muitos de nós pensamos. Porque muitas pessoas dizem, eles não são seguros. Que você não pode confiar nesses carros.

A questão é: quando você tem os dados, pode fazer as estatísticas.

Você pode mostrar às pessoas que a direção autônoma realmente funciona de forma confiável. Você verá a pergunta: Isso é permitido ou não? Terá ido embora mais rápido do que você
pensar.

Porque as agências governamentais podem começar a testar os algoritmos com base em cenários predefinidos. Eles podem executar benchmarks e pontuar o desempenho dos carros.

Todas essas opiniões, se funcionar ou não, desaparecerão.

A agência de automóveis tem as estatísticas. As estatísticas mostram às pessoas como os bons carros funcionam.

Empresas como a Tesla têm uma vida muito fácil. Porque os dados já estão lá.

Eles só precisam nos mostrar que os algoritmos funcionam. O fim.

23 Visualização de dados

23.1 Android e IOS

23.2 Como projetar APIs para aplicativos móveis

23.3 Como usar servidores Web para exibir conteúdo

Esta seção não contém nenhum texto, é por isso que a página está confusa

23.3.1 Tomcat

23.3.2 Molhe

23.3.3 NodeRED

23.3.4 Reagir

23.4 Ferramentas de Business Intelligence

23.4.1 Tabela

23.4.2 PowerBI

23.4.3 Quliksense

23.5 Gerenciamento de identidade e dispositivos

23.5.1 O que é um gêmeo digital?

23.5.2 Diretório Ativo

Parte III

Construindo um exemplo de plataforma de dados

24 My Big Data Platform Blueprint

Há algum tempo, criei um projeto de plataforma de big data simples e modular para mim. É baseado no que tenho visto no campo e lido em blogs de tecnologia em todo o mundo.
Internet.

Hoje vou compartilhar com vocês.

Por que eu acredito que será super útil para você?

Porque, ao contrário de outros blueprints, não é focado em tecnologia. Ele é baseado em quatro padrões comuns de design de plataforma de big data.

Seguir meu projeto permitirá que você crie a plataforma de big data que atenda exatamente às suas necessidades. Construir a plataforma perfeita permitirá que os cientistas de dados descubram novos insights.

Ele permitirá que você lide perfeitamente com big data e tome decisões baseadas em dados.

O PROJETO O projeto concentra-se em quatro áreas principais: ingestão, armazenamento, análise e exibição.

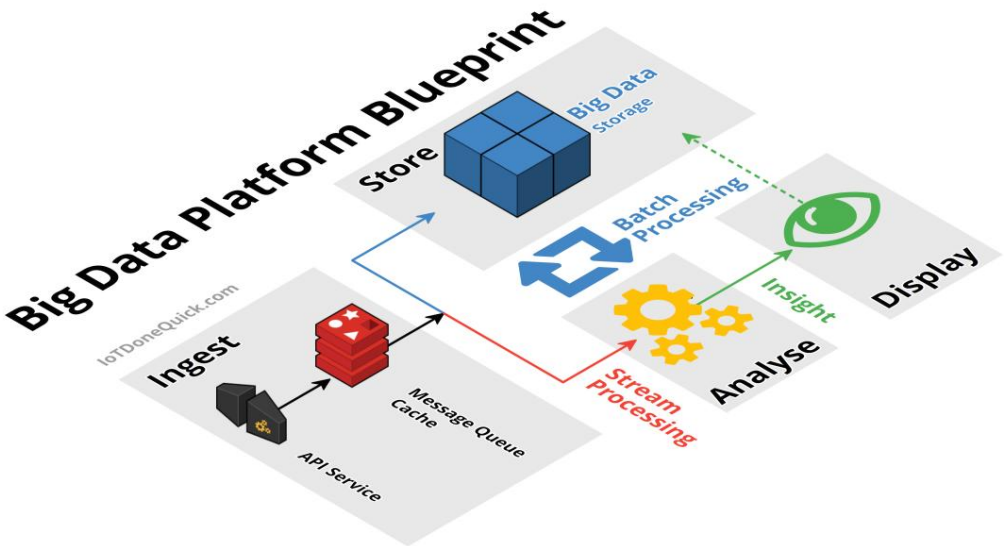

Figura 24.1: Planta da plataforma

Tendo a plataforma dividida dessa forma, ela se torna uma plataforma modular com interfaces fracamente acopladas.

Por que é tão importante ter uma plataforma modular?

Se você tem uma plataforma que não é modular, você acaba com algo fixo ou difícil de modificar. Isso significa que você não pode ajustar a plataforma às mudanças nos requisitos da empresa.

Devido à modularidade, é possível trocar todos os componentes, se necessário.

Agora, vamos falar mais sobre cada área-chave.

24.1 Ingerir

A ingestão trata de obter os dados da fonte e disponibilizá-los para estágios posteriores. As fontes podem ser tudo de tweets, logs de servidor para dados do sensor IoT como de
carros.

As fontes enviam dados para seus serviços de API. A API enviará os dados para um armazenamento temporário.

O armazenamento temporário permite que outras etapas tenham acesso simples e rápido aos dados recebidos.

Uma ótima solução é usar sistemas de fila de mensagens como Apache Kafka, RabbitMQ ou AWS Kinesis. Às vezes, as pessoas também usam caches para aplicativos especializados como o Redis.

Uma boa prática é que o armazenamento temporário siga o padrão de publicação e assinatura.
Dessa forma, as APIs podem publicar mensagens e o Analytics pode consumi-las rapidamente.

24.2 Analisar / Processar

O estágio de análise é onde a análise real é feita. Analytics, na forma de fluxo e processamento em lote.

Os dados de streaming são obtidos da ingestão e inseridos na análise. Streaming analisa os dados “ao vivo” assim, gerando resultados rápidos.

Como etapa central e mais importante, a análise também tem acesso ao armazenamento de big data.
Por causa dessa conexão, a análise pode pegar uma grande parte dos dados e analisá-los.

Esse tipo de análise é chamado de processamento em lote. Ele lhe dará respostas para as grandes questões.

Para saber mais sobre fluxo e processamento em lote, leia minha postagem no blog: Como criar produtos novos e empolgantes com auxílio de big data

O processo analítico, em lote ou streaming, não é um processo unidirecional. O Analytics também pode gravar dados de volta no armazenamento de big data.

Muitas vezes, gravar dados de volta no armazenamento faz sentido. Ele permite que você combine saídas analíticas anteriores com os dados brutos.

O insight analítico pode dar significado aos dados brutos quando você os combina. Muitas vezes, essa combinação permite que você crie insights ainda mais úteis.

Uma ampla variedade de ferramentas de análise está disponível. Variando de MapReduce ou AWS Elastic MapReduce a Apache Spark e AWS lambda.

24.3 Loja

Este é o típico armazenamento de big data onde você apenas armazena tudo. Ele permite que você analise o quadro geral.

A maioria dos dados pode parecer inútil por enquanto, mas é de extrema importância mantê-los.
Jogar dados fora é um grande não, não.

Por que não jogar algo fora quando é inútil?

Embora pareça inútil por enquanto, os cientistas de dados podem trabalhar com os dados. Eles podem encontrar novas maneiras de analisar os dados e gerar informações valiosas a partir deles.

Que tipo de sistemas podem ser usados para armazenar big data?

Sistemas como Hadoop HDFS, Hbase, Amazon S3 ou DynamoDB são perfeitos para armazenar big data.

Confira meu podcast sobre como decidir entre SQL e NoSQL: https://anchor.fm/andreaskayy/embed/ Vs-SQL-How-To-Choose-e12f1o

24.4 Exibição

Exibir dados é tão importante quanto ingeri-los, armazená-los e analisá-los. As pessoas precisam ser capaz de tomar decisões baseadas em dados.

É por isso que é importante ter uma boa apresentação visual dos dados. Às vezes, você tem muitos casos de uso ou projetos diferentes usando a plataforma.

Pode não ser possível para você construir a IU perfeita que se adapta a todos. O que você deve fazer neste caso é permitir que outras pessoas criem a interface do usuário perfeita.

Como fazer isso? Criando APIs para acessar os dados e disponibilizá-los aos desenvolvedores.

De qualquer forma, UI ou API, o truque é dar ao estágio de exibição acesso direto aos dados no cluster de big data. Esse tipo de acesso permitirá que os desenvolvedores usem resultados analíticos, bem como dados brutos, para criar o aplicativo perfeito.

25 Arquitetura Lambda

25.1 Processamento em Lote

Faça as grandes perguntas. Lembra da sua última declaração de imposto anual?

Você separa as pastas. Você corre pela casa procurando os recibos.

Todas essas coisas divertidas.

Quando você finalmente encontrar tudo, preencha o formulário e envie-o.

Fazer a declaração de impostos é um excelente exemplo de um processo em lote.

Os dados chegam e são armazenados, a análise carrega os dados do armazenamento e cria uma saída (insight):

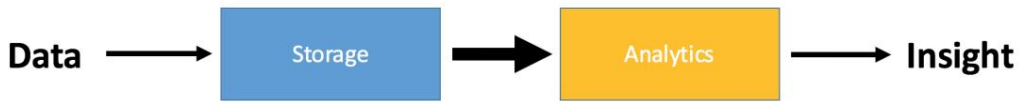

Figura 25.1: Pipeline de processamento em lote

O processamento em lote é algo que você faz sem agendamento ou agendado (declaração de imposto). É usado para fazer as grandes perguntas e obter insights olhando para o quadro geral.

Para fazer isso, as tarefas de processamento em lote usam grandes quantidades de dados. Esses dados são fornecidos por sistemas de armazenamento como o Hadoop HDFS.

Eles podem armazenar muitos dados (petabytes) sem problemas.

Os resultados dos trabalhos em lote são muito úteis, mas o tempo de execução é alto. Porque a quantidade de dados usados é alta.

Pode levar minutos ou às vezes horas até que você obtenha seus resultados.

25.2 Processamento de Fluxo

Obtenha informações instantâneas sobre seus dados.

O streaming permite que os usuários tomem decisões rápidas e executem ações com base em informações em "tempo real". Ao contrário do processamento em lote, o streaming processa os dados em tempo real, conforme eles chegam em.

Com o streaming, você não precisa esperar minutos ou horas para obter resultados. Você obtém informações instantâneas sobre seus dados.

No pipeline de processamento em lote, a análise ocorreu após o armazenamento de dados. Teve acesso a todos os dados disponíveis.

O processamento de fluxo cria insights antes do armazenamento de dados. Ele só tem acesso a fragmentos de dados à medida que chegam.

Como resultado, o escopo do insight produzido também é limitado. Porque falta o quadro geral.

Figura 25.2: Pipeline de processamento de fluxo

Somente com análise de streaming você pode criar serviços avançados para o cliente.
A Netflix, por exemplo, incorporou o processamento de fluxo no Chuckwa V2.0 e no novo pipeline Keystone.

Um exemplo de serviços avançados por meio do processamento de fluxo é o recurso "Trending Now" da Netflix.
Confira o estudo de caso da Netflix.

25.3 Você deve fazer processamento em fluxo ou em lote?

É uma boa ideia começar com o processamento em lote. O processamento em lote é a base de toda boa plataforma de big data.

Uma arquitetura de processamento em lote é simples e, portanto, rápida de configurar. A simplicidade da plataforma significa que também será relativamente barato de executar.

Uma plataforma de processamento em lote permitirá que você faça as grandes perguntas rapidamente. Eles fornecerão informações valiosas sobre seus dados e clientes.

Quando chegar a hora e você também precisar fazer análises em tempo real, adicione um pipeline de streaming à sua plataforma de big data de processamento em lote.

25.4 Arquitetura Lambda Alternativa

25.4.1 Arquitetura Kappa

25.4.2 Arquitetura Kappa com Kudu

26 pensamentos sobre a escolha do Ambiente Alvo

26.1 Nuvem x local

26.2 Fornecedores independentes ou nativos da nuvem

27 pensamentos sobre a escolha de um Ambiente de desenvolvimento

27.1 Nuvem como ambiente de desenvolvimento

27.2 Ambiente de desenvolvimento local

27.3 Arquitetura de Dados

27.3.1 Dados de Origem

27.3.2 Requisitos analíticos para streaming

27.3.3 Requisitos analíticos para processamento em lote

27.3.4 Visualização de Dados

27.4 Marco 1 - Decisões de Ferramenta

Parte IV

Estudos de caso

28 Como faço estudos de caso

28.1 Ciência de Dados @Airbnb

https://medium.com/airbnb-engineering/airbnb-engineering-infrastructure/home

Blog de engenharia do Airbnb: https://medium.com/airbnb-engineering

Infraestrutura de dados: https://medium.com/airbnb-engineering/data-infrastructure-at-airbnb

Escalando o nível de serviço: https://medium.com/airbnb-engineering/unlocking-horizontal-scalab

Druid Analytics: https://medium.com/airbnb-engineering/druid-airbnb-data-platform-601c3

Spark Streaming para registro de eventos: https://medium.com/airbnb-engineering/scaling-spark-str

-Druid Wiki: https://en.wikipedia.org/wiki/Apache_Druid

28.2 Ciência de Dados @Baidu

https://www.slideshare.net/databricks/spark-sql-adaptive-execution-unleashes-the-pow

28.3 Ciência de Dados @Blackrock

https://www.slideshare.net/DataStax/maintaining-consistency-across-data-centers-rand

28.4 Dados científicos @BMW

Big data na indústria automotiva - usando dados do veículo https://www. unibw.de/
code.../ws3_bigdata_vortrag_widmann.pdf

28.5 Ciência de dados @Booking.com

https://www.slideshare.net/ConfluentInc/data-streaming-ecosystem-management-at-booki ref=https://www.confluent.io/kafka-summit-sf18/data-streaming-ecosystem-management

https://www.slideshare.net/SparkSummit/productionizing-behavioural-features-for-mach

https://www.slideshare.net/ConfluentInc/data-streaming-ecosystem-management-at-booki ref=https://www.confluent.io/kafka-summit-sf18/data-streaming-ecosystem-management

Druida: https://towardsdatascience.com/introduction-to-druid-4bf285b92b5a

Arquitetura Kafka: https://data-flair.training/blogs/kafka-architecture/

Plataforma Confluent: https://www.confluent.io/product/confluent-platform/

28.6 Ciência de Dados @CERN

https://en.wikipedia.org/wiki/Large_Hadron_Collider

http://www.lhc-facts.ch/index.php?page=datenverarbeitung

https://openlab.cern/sites/openlab.web.cern.ch/files/2018-09/2017_ESADE_Madrid_ Big_Data.pdf

https://openlab.cern/sites/openlab.web.cern.ch/files/2018-05/kubeconeurope2018-cern-1 pdf

https://www.slideshare.net/SparkSummit/next-cern-accelerator-logging-service-with-ja

https://databricks.com/session/the-architecture-of-the-next-cern-accelerator-logging

http://opendata.cern.ch

https://gobblin.apache.org

https://www.slideshare.net/databricks/cerns-next-generation-data-analysis-platform-w

https://www.slideshare.net/SparkSummit/realtime-detection-of-anomalies-in-the-databa

28.7 Ciência de dados @Disney

https://medium.com/disney-streaming/delivering-data-in-real-time-via-auto-scaling-ki

28.8 Ciência de dados @Drivetribe

https://berlin-2017.flink-forward.org/kb_sessions/drivetribes-kappa-architecture-wit

https://www.slideshare.net/FlinkForward/flink-forward-berlin-2017-aris-kyriakos-koli

28.9 Ciência de dados @Dropbox

https://blogs.dropbox.com/tech/2019/01/finding-kafkas-throughput-limit-in-dropbox-in

28.10 Ciência de dados @Ebay

https://www.slideshare.net/databricks/moving-ebays-data-warehouse-over-to-apache-spa https://www.slideshare.net/databricks/analytical-dbms-to-apache-spark-auto -migração

28.11 Ciência de dados @Expedia

https://www.slideshare.net/BrandonOBrien/spark-streaming-kafka-best-practices-w-bran https://www.slideshare.net/Naveen1914/brandon-obrien-streamingdata

28.12 Ciência de Dados @Facebook

https://code.fb.com/core-data/apache-spark-scale-a-60-tb-production-use-case/

28.13 Ciência de Dados @@Grammarly

https://www.slideshare.net/databricks/building-a-versatile-analytics-pipeline-on-top

28.14 Ciência de Dados @ING Fraud

https://sf-2017.flink-forward.org/kb_sessions/streaming-models-how-ing-adds-models-a

28.15 Ciência de dados @Instagram

https://www.slideshare.net/SparkSummit/lessons-learned-developing-and-managing-massi

28.16 Ciência de Dados @LinkedIn

https://engineering.linkedin.com/teams/data#0

http://www.bigdatausecases.info/companies/linkedin

28.17 Ciência de dados @Lyft

https://eng.lyft.com/running-apache-airflow-at-lyft-6e53bb8fccff

28.18 Dados científicos @NASA

http://sites.nationalacademies.org/cs/groups/ssbsite/documents/webpage/ssb_ 182893.pdf

https://esip.figshare.com/articles/Apache_Science_Data_Analytics_Platform/5786421

http://www.socallinuxexpo.org/sites/default/files/presentations/OnSightCloudArchitect pdf

https://www.slideshare.net/SparkSummit/spark-at-nasajplchris-mattmann?qid=90968554-288e-454a-b63a-21a45cfc897d&v=&b=&from_search=4

https://en.m.wikipedia.org/wiki/Hierarchical_Data_Format

28.19 Data Science @Netflix – disponível

A Netflix revolucionou a forma como assistimos filmes e TV. Atualmente, mais de 75 milhões de usuários assistem a 125 milhões de horas de conteúdo Netflix todos os dias!

A receita da Netflix vem de um serviço de assinatura mensal. Portanto, o objetivo da Netflix é mantê-lo inscrito e obter novos assinantes.

Para conseguir isso, a Netflix está licenciando filmes de estúdios, bem como criando seus próprios filmes originais e séries de TV.

Mas oferecer novos conteúdos não é tudo. O que também é muito importante é mantê-lo assistindo ao conteúdo que já existe.

Para poder recomendar seu conteúdo, a Netflix está coletando dados dos usuários. E está arrecadando muito.

Atualmente, a Netflix analisa cerca de 500 bilhões de eventos de usuários por dia. Isso resulta em impressionantes 1,3 petabytes todos os dias.

Todos esses dados permitem que a Netflix crie sistemas de recomendação para você. Os recomendadores estão mostrando conteúdo de que você pode gostar, com base em seus hábitos de visualização ou no que está em alta no momento.

O pipeline de processamento em lote da Netflix Quando a Netflix começou, eles tinham uma arquitetura de sistema de processamento em lote muito simples.

Os principais componentes foram Chuckwa, um sistema de coleta de dados escalável, Amazon S3 e Elastic MapReduce.

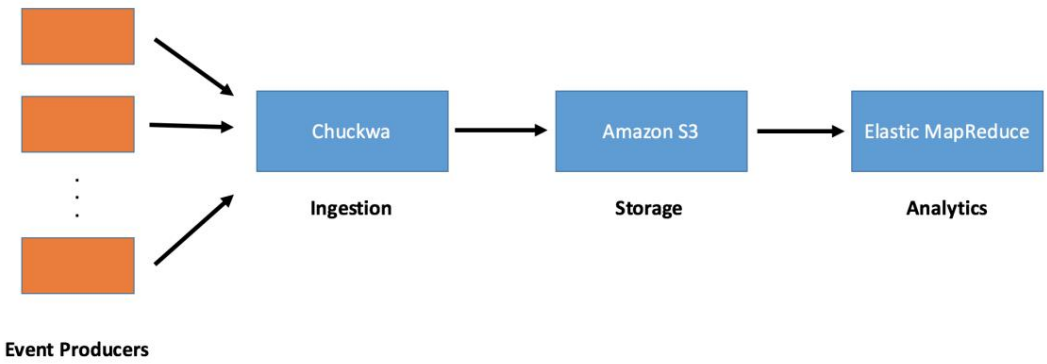

Figura 28.1: Antigo pipeline de processamento em lote da Netflix

Chuckwa escreveu mensagens recebidas em arquivos de sequência do Hadoop, armazenados no Amazon S3. Esses arquivos poderiam então ser analisados por trabalhos do Elastic MapReduce.

Os trabalhos do pipeline de processamento em lote da Netflix eram executados regularmente diariamente e a cada hora. Como resultado, a Netflix pôde aprender como as pessoas usavam os serviços a cada hora ou uma vez por dia.

Saiba o que os clientes querem: porque você está olhando para o quadro geral, você pode criar novos produtos. A Netflix usa insights de big data para criar novos programas de TV e

filmes.

Eles criaram House of Cards com base em dados. Há uma palestra muito interessante do Ted sobre isso que você deveria assistir:

Como usar dados para fazer um programa de TV de sucesso — Sebastian Wernicke:

O processamento em lote também ajuda a Netflix a saber o episódio exato de um programa de TV que o deixa viciado. Não apenas globalmente, mas para todos os países onde a Netflix está disponível.

Confira o artigo do TheVerge

Eles sabem exatamente qual programa funciona em qual país e qual programa não funciona.

Isso os ajuda a criar programas que funcionam em qualquer lugar ou selecionar os programas para licenciar em diferentes países. A Alemanha, por exemplo, não tem a biblioteca completa que os americanos têm :(

Temos que aturar apenas uma pequena porção de programas de TV e filmes. Se você tiver que selecionar, por que não selecionar aqueles que funcionam melhor?

O processamento em lote não é suficiente Como plataforma de dados para geração de insights, o pipeline Cuckwa foi um bom começo. É muito importante poder criar exibições agregadas por hora e diariamente para o comportamento do usuário.

Até hoje, a Netflix ainda realiza muitos trabalhos de processamento em lote.

O único problema é: com o processamento em lote, você basicamente olha para o passado.

Para a Netflix e empresas orientadas a dados em geral, olhar para o passado não é suficiente.
Eles querem uma visão ao vivo do que está acontecendo.

O recurso de tendências agora Um dos recursos mais recentes da Netflix é “Tendências agora”. Para o usuário médio, parece que “Trending Now” significa atualmente o mais assistido.

Isso é o que recebo como tendência enquanto escrevo isso em uma manhã de sábado às 8h na Alemanha. Mas é muito mais.

O que está sendo observado atualmente é apenas uma parte dos dados usados para gerar “Trending Now”.

"Tendência agora" é criado com base em dois tipos de fontes de dados: eventos de reprodução e impressão eventos.

Figura 28.2: Recurso Trending Now do Netflix

Quais mensagens esses dois tipos realmente incluem não são realmente comunicadas pela Netflix.
Fiz algumas pesquisas no Netflix Techblog e descobri o seguinte:

Os eventos de reprodução incluem qual título você assistiu por último, onde parou de assistir, onde usou o retrocesso de 30 e outros. Os eventos de impressão são coletados enquanto você navega na Biblioteca Netflix, como rolar para cima e para baixo, rolar para a esquerda ou para a direita, clicar em um filme e assim por diante

Basicamente, os eventos de reprodução registram o que você faz enquanto assiste. Os eventos de impressão capturam o que você faz no Netflix, enquanto não está assistindo a algo.

Arquitetura de streaming em tempo real da Netflix A Netflix usa três serviços voltados para a Internet para trocar dados com o navegador ou aplicativo móvel do cliente. Esses serviços são serviços da Web simples baseados no Apache Tomcat.

O serviço para receber eventos de reprodução é chamado de “Visualização do histórico”. Os eventos de impressão são coletados com o serviço “Beacon”.

O “Serviço de recomendação” faz recomendações com base em dados de tendências disponíveis para clientes.

As mensagens dos serviços Beacon e Viewing History são colocadas no Apache Kafka. Ele atua como um buffer entre os serviços de dados e a análise.

Beacon e Viewing History publicam mensagens em tópicos Kafka. A análise se inscreve nos tópicos e obtém as mensagens entregues automaticamente da maneira primeiro a sair.

Após a análise, o fluxo de trabalho é direto. Os dados de tendências são armazenados em um armazenamento de valor-chave do Cassandra. O serviço de recomendação tem acesso ao Cassandra e está disponibilizando os dados para o cliente Netflix.

Os algoritmos de como o sistema analítico está processando todos esses dados não são conhecidos do público. É um segredo comercial da Netflix.

O que se sabe é a ferramenta de análise que eles usam. Em fevereiro de 2015, eles escreveram no blog de tecnologia que usam uma ferramenta personalizada.

Eles também afirmaram que a Netflix substituirá a ferramenta de análise personalizada por

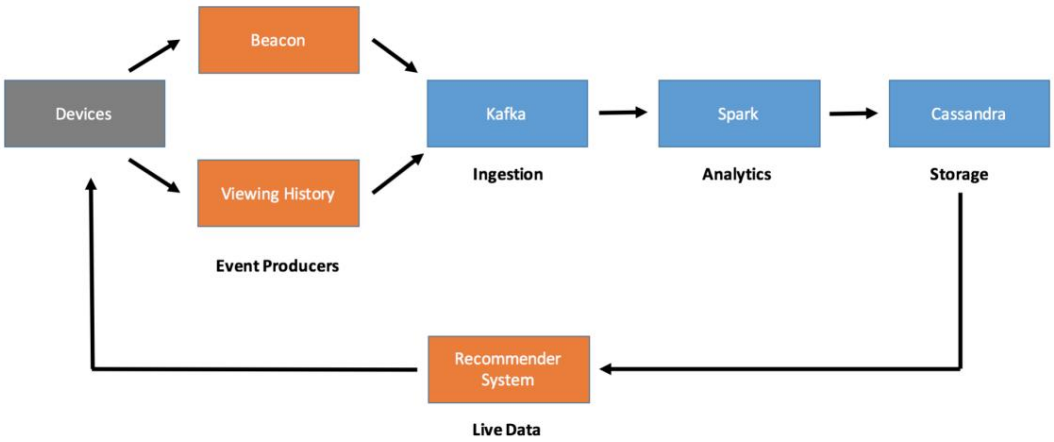

Figura 28.3: Pipeline de streaming Netflix

Apache Spark streaming no futuro. Meu palpite é que eles mudaram para o Spark há algum tempo, porque a postagem deles tem mais de um ano.

28.20 Ciência de Dados @OTTO

https://www.slideshare.net/SparkSummit/spark-summit-eu-talk-by-sebastian-schroeder-a

28.21 Ciência de Dados @Paypal

https://www.paypal-engineering.com/tag/data/

28.22 Ciência de Dados @Pinterest

https://www.slideshare.net/ConfluentInc/pinterests-story-of-streaming-hundreds-of-te ref=https://www.confluent.io/kafka-summit-sf18/pinterests-story-of-streaming -centenas

https://www.slideshare.net/ConfluentInc/building-pinterest-realtime-ads-platform-usi ref=https://www.confluent.io/kafka-summit-sf18/building-pinterest-real-time-ads -plat

https://medium.com/@Pinterest_Engineering/building-a-real-time-user-action-counting-

https://medium.com/pinterest-engineering/goku-building-a-scalable-and-high-performan

https://medium.com/pinterest-engineering/building-a-dynamic-and-responsive-pinterest

https://medium.com/@Pinterest_Engineering/building-pin-stats-25ec8460e924

https://medium.com/@Pinterest_Engineering/improving-hbase-backup-efficiency-at-pinte

https://medium.com/@Pinterest_Engineering/pinterest-joins-the-cloud-native-computing

https://medium.com/@Pinterest_Engineering/using-kafka-streams-api-for-predictive-bud

https://medium.com/@Pinterest_Engineering/auto-scaling-pinterest-df1d2beb4d64

28.23 Ciência de dados @Salesforce

https://engineering.salesforce.com/building-a-scalable-event-pipeline-with-heroku-an

28.24 Ciência de dados @Slack

https://speakerdeck.com/vananth22/streaming-data-pipelines-at-slack

28.25 Ciência de dados @Spotify

https://labs.spotify.com/2016/02/25/spotifys-event-delivery-the-road-to-the-cloud-pa

https://www.slideshare.net/InfoQ/scaling-the-data-infrastructure-spotify

28.26 Ciência de Dados @Symantec

https://www.slideshare.net/planetcassandra/symantec-cassandra-data-modelling-techniq

28.27 Ciência de dados @Tinder

https://www.slideshare.net/databricks/scalable-monitoring-using-apache-spark-and-fri

28.28 Ciência de Dados @Twitter

https://www.slideshare.net/sawjd/real-time-processing-using-twitter-heron-by-karthik

https://www.slideshare.net/sawjd/big-data-day-la-2016-big-data-track-twitter-heron-s

28.29 Ciência de Dados @Uber

https://eng.uber.com/uber-big-data-platform/

https://eng.uber.com/aresdb/

28h30 Ciência de Dados @Upwork

https://www.slideshare.net/databricks/how-to-rebuild-an-endtoend-ml-pipeline-with-da

28.31 Ciência de Dados @Woot

https://aws.amazon.com/de/blogs/big-data/our-data-lake-story-how-woot-com-built-a-se

28.32 Data Science @Zalando

https://jobs.zalando.com/tech/blog/what-is-hardcore-data-science--in-practice/ ?gh_src=4n3gxh1

https://jobs.zalando.com/tech/blog/complex-event-generation-for-business-process-mon

Bibliografia

[1] J. Ely e I. Stavrov, Analisando pó de giz e velocidades de escrita: abordagens computacionais e geométricas, BoDine Journal of Mathematics 3 (2001), 14-159.

Lista de Figuras

Lista de mesas